УДК 681.3.324

Д. И. Шпилевой

РЕГЛАМЕНТАЦИЯ ДОСТУПА К ДАННЫМ В СУБД ORACLE

В основе регламентации доступа к данным в Oracle лежит парольная защита. В наиболее распространенном случае для работы с данными в своей схеме пользователь Oracle обязан указать пароль. Так, пароль указывается при выполнении соединения с СУБД (например, в SQL*Plus в команде CONNECT), в предложении SQL создании пользователя или в полном определении связи с посторонней БД (database link). Безусловно, это самый простой метод. Однако его удобство и, что более важно, безопасность оставляют желать лучшего. Ключевые контейнеры могут быть скопированы и впоследствии «взломаны» методом простого перебора паролей. Определенные неудобства вызывает и привязка ключевого контейнера к конкретной рабочей станции.

Хранение пароля

Заданный для пользователя Oracle командой CREATE/ALTER USER пароль подвергается преобразованию и попадает в словарь-справочник в виде свертки (password hash). При указании пароля в момент установления соединения с СУБД Oracle заново вычислит свертку и сравнит ее с хранимой в БД. В открытом виде пароли в БД не хранятся.

Основное место хранения свертки пароля – таблица словаря-справочника SYS.USER$. Над этой таблицей, как базовой, построена производная, SYS.DBA_USERS. Если в профиле (profile) пользователя включен параметр PASSWORD_REUSE_TIME, свертки пароля также хранятся в SYS.USER_HISTORY$. Увидеть свертки логически можно, выдав например:

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT username, password FROM dba_users;

USERNAME                       PASSWORD
------------------------------ ------------------------------
dima                           4A3BA55E08595C81
misha                           66F4EF5650C20355
roma                           7C9BA362F8314299
```

Физически свертки можно наблюдать в файлах ОС: "парольном" PWD.ORA; табличного пространства SYSTEM, где хранятся SYS.USER$ и SYS.USER_HISTORY$ (часто это SYSTEM01.DBF); полного экспорта; архивированных журналов.

Алгоритм вычисления свертки пароля

Как описано в документации, в СУБД Oracle для создания паролей может использоваться следующий набор символов:

1) цифры '0123456789',

2) буквы 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ',

3) символы !"#$%&()`*+,-/:;<=>?_.

Поскольку все символы преобразуются к верхнему регистру, то мощность алфавита паролей равна 10+26+21=57 символов. В результате допустимых паролей может быть

$$\sum_{i=1}^{i=30} 57^i \cong 4,8 \cdot 10^{52}$$

(все пароли длиной 1 символ, плюс все пароли длиной два символа и т.д. до 30 символов).

Перед помещением его в словарь-справочник БД и при проверке подлинности (аутентичности) официально фирмой-изготовителем не опубликован. Тем не менее в [1] этот алгоритм описан так:

1. Конкатенировать логин и пароль пользователя (обозначим эту операцию ||).
2. Преобразовать полученную строку к верхнему регистру (UPPER(Логин||Пароль)).
3. Если в ОС используется однобайтовая кодировка, то преобразовать каждый символ в двухбайтовый, заполнив старший байт нулями (0x00).
4. Зашифровать получившуюся строку (дополняя ее нулями до длины блока), используя алгоритм DES в режиме CBC с фиксированным ключом, значение которого есть 0x0123456789ABCDEF.
5. Зашифровать получившуюся строку еще раз с помощью DES-CBC, но используя последний блок предыдущего шага как ключ шифрования.

Графически это выглядит так, как показано на рис.1.

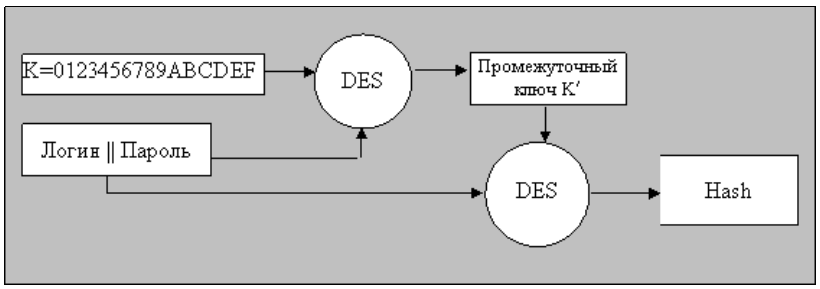

Рис. 1. Алгоритм вычисления свертки пароля

Обозначим шифрование строки X на ключе K как DES(X,K). Тогда, в аналитической форме уравнение шифрования выглядит так:

$$H = DES(UPPER(Логин||Пароль), DES(UPPER(Логин||Пароль), K)). \quad (1)$$

Если обозначить UPPER(Логин||Пароль) =X, то уравнение станет совсем простым:

$$H=DES(X,DES(X,K)). \quad (2)$$

Обратную операцию (расшифрование H на ключе K) обозначим так: $X=DES^{-1}(H,K)$.

Особенности такого алгоритма:

– Свертка не зависит от регистра букв. Например, пары SCOTT/TIGER, Scott/Tiger, scoTT/TigeR дадут одну и ту же свертку F894844C34402B67.

– Одинаковые склеенные пары *имя_пользователя/пароль* дают одинаковую свертку. Например, пары SCOTT/TIGER, SCOT/TTIGER, SCOTTTIG/ER дадут одну и ту же свертку F894844C34402B67.

– Свертка не зависит от БД. Например, где бы мы ни создавали БД Oracle, свертка для пользователя SCOTT и пароля TIGER всегда будет F894844C34402B67.

– Используется шифрование DES. Алгоритм шифрования DES определен для 56-битового ключа, но не во всех версиях СУБД Oracle выдерживается значение 56 бит. В силу ограничений экспортного законодательства США, запрещающего продавать за рубеж ПО и аппаратуру, содержащие стойкие криптографические алгоритмы, длина ключа для экспортных версий СУБД Oracle искусственно занижалась до 40 бит, что облегчает правительству США чтение шифрованной информации. В последние годы экспортный контроль Гос. Департаментом США был значительно ослаблен, поскольку производители ПО и электроники стали нести слишком большие потери (упущенная выгода) из-за этих ограничений.

Взлом пароля

С алгоритмической точки зрения при вскрытии современного шифра со случайным ключом существуют два принципиально различных направления атаки: поиск и использование некоторых уязвимостей алгоритма и перебор ключей.

Поиск уязвимостей – долгая и кропотливая аналитическая работа, позволяющая делать выводы об исходных данных на основе шифрованных данных и знания алгоритма (криптосистемы).

Перебор ключей (второе название – «силовая атака») – стандартный способ подбора пароля, заключающийся в генерации очередного значения, шифровании его и сравнении полученного результата с имеющимся хешем. Основную роль в переборе ключей играют вычислительные мощности злоумышленника, таким образом, это универсальный способ для взлома различных шифров, поскольку этот способ не зависит от внутреннего устройства шифра.

Ввиду больших затрат времени и вычислительных средств, к перебору прибегают в том случае, когда остальными методами решить проблему не получается. И даже в случае перебора все равно пытаются максимально сузить перебираемое множество за счет анализа особенностей конкретного алгоритма.

В дальнейшем будем предполагать, что злоумышленник знает логин пользователя, хеш H и ключ шифрования K.

Если бы криптосистема в Oracle строилась на однократном шифровании, при котором H=DES(X,K), то при известных H и K она бы мгновенно теряла стойкость, поскольку X= DES^{-1} (H,K), и все параметры правой части известны.

В Oracle ситуация сложнее, и при наших допущениях, если требуется вычислить пароль X, то уравнение будет выглядеть так:

$$X = DES^{-1}(H, K). \qquad (3)$$

Заметим, что для обратной операции требуется промежуточный ключ K . То есть, чтобы вычислить значение X, в данном уравнении недостает значения K= =DES(X, K), которое является промежуточным ключом и не хранится нигде. А вычислить DES(X, K) нет возможности, потому что X есть Логин||Пароль и пароль, по условиям задачи, неизвестен.

Написанное выше означает, что, даже зная ключ шифрования K, можно только вычислить всевозможные значения DES(Логин||Пароль,K) для всевозможных значений пароля, а затем найти в этом множестве подходящее значение. А сделать это

можно путем перебора всех возможных паролей, т.е. фактически только путем силовой атаки.

Для вычисления секретного пароля X необходимо вначале вычислить промежуточный ключ *DES(Логин||Пароль,K)* для всевозможных значений пароля. Иными словами, теоретическая сложность взлома этой криптосистемы при отсутствии реальных уязвимостей, приближается к сложности силовой атаки на этот шифр.

Силовая атака

Если аналитическим путем найти уязвимую точку не удастся, то всегда существует метод, называемый силовая атака.

Так, можно, игнорировав значение H, непосредственно приступить к вычислениям $X=DES^{-1}(K',K)$, взяв генератор случайных чисел в качестве источника K' (расшифровываем некоторое случайное значение K' на ключе K, затем сравниваем результат с X). Эта атака позволяет сэкономить на операции $X = DES^{-1}(H, K')$ и таким образом ускорить процесс.

На red-database-security.com приводится пример программы, подбирающей пароль перебором, отталкиваясь от известного имени пользователя и известной свертки. Разброс значений скорости перебора на Pentium-4 с частотой 3 GHz такой: код SQL генерирует до 500 паролей в секунду, код на C до 1.100.000 паролей в секунду (наилучший результат). При этом время, необходимое для полного перебора паролей, распределяется так:

5-символьные пароли - 10 секунд;
6-символьные пароли - 5 минут;
7-символьные пароли - 2 часа;
8-символьные пароли - 2,1 дня;
9-символьные пароли - 57 дней;
10-символьные пароли - 4 года.

Обладание хешем пароля позволяет выполнить атаку на пароль в off-лайне, когда противник может заранее вычислить хеши паролей для известных логинов типа sys, system и т.д. Используя заранее вычисленные хеши паролей, злоумышленник может при необходимости просматривать эти таблицы в поисках подходящего хеша.

Использование предварительно вычисленных таблиц обычно ограничено множеством алгоритмов, не использующих случайную привязку. Но привязка в СУБД Oracle не мешает злоумышленнику, поскольку он может построить хеши для заранее выбранной учетной записи (логина). Отличным кандидатом будет логин SYSTEM, который существует во всех БД и гарантирует привилегированный доступ.

Длина хешей и паролей

Длина паролей в Oracle может достигать 30 знаков, а длина хеша фиксирована и равна 8 байт, т.е. 64 бита. Таким образом, на одно хеш-значение приходится приблизительно $10^{52}/10^{19}=10^{33}$ возможных паролей в Oracle.

Вычислим эффективную длину пароля, в зависимости от длины хеш-значения. Возможных значений хеш-функции 2^{64}, следовательно, надо решить уравнение $57^x=2^{64}$. В результате получаем $x\approx11$. Этот факт показывает, что практически для всех паролей длиной больше 11 знаков будет существовать более короткий пароль. Следовательно, в результате силовой атаки злоумышленник обнаружит один из кратчайших паролей. Но поскольку хеш короткого и длинного паролей одинаковы, то пароли можно считать неразличимыми. Следовательно, при длине пароля более

11 знаков увеличение длины пароля не увеличивает время безопасности СУБД, т.е. время, необходимое злоумышленнику для его нахождения.

Если Oracle решит отказаться от принудительного преобразования символов к верхнему регистру, то в этом случае x≈10. Разница всего в 1 знак.

Для современных хеш-функций минимальная длина хеш-значения обычно находится на уровне 128 бит.

В этом случае $21^{28} \approx 57^{22} \approx 83^{20}$, что означает, что для хеш-значений 128-битовой длины потребуется 22 знака из 57-знакового множества, либо 20 знаков для 83-знакового (т.е. 20 знаков, если отказаться от преобразования к верхнему регистру).

Ну и, наконец, 30-символьному паролю должен соответствовать хеш длиной

- 175 бит, если сохраняется Upper() (57 знаков);
- 191,25 бит, если отказаться от Upper() (83 знака).

Варианты усовершенствования парольной защиты

- От DES следует отказаться в силу малой длины ключа и малой длины блока. Малая длина ключа означает успешность силовой атаки за небольшое время, а малая длина блока означает повышенный уровень коллизий.

Можно взять новый, более совершенный алгоритм AES, пришедший в 2000 г. на смену DES'у. Этот алгоритм допускает вариацию длины блока (128, 192, 256 бит), длины ключа 128, 192, 256 бит и количества раундов (чем больше раундов, тем выше качество шифрования и меньше скорость). Алгоритм AES сертифицирован национальным институтом США по стандартизации NIST и АНБ (Агентство Национальной Безопасности США).

Либо можно выбрать сертифицированную криптографически стойкую хеш-функцию, которых к сегодняшнему дню разработано несколько десятков. В России таким стандартом является ГОСТ Р 34.11, в Европе – RIPEMD, в США – SHA, MD5. Они протестированы компетентными организациями (органами) и приняты в качестве государственных стандартов. Для предотвращения коллизий и сведения к минимуму эффекта от парадокса дней рождений современную длину блока можно установить в 256 бит. По современным оценкам этой длины хватит на сотню другую лет, при сохранении современных темпов развитии вычислительной техники.

- Конкатенацию логина и пароля следует заменить другим преобразованием, например сложением. Если в базе данных заведен пользователь "ora" с паролем "cle", то в процессе силовой атаки мы можем наткнуться на логин "o" с паролем "racle", в результате чего угадать реальную пару логин||пароль будет совсем несложно, потому что для каждого пароля случайной добавкой является часть его логина. Таким образом, в целях минимизации числа коллизий следует от конкатенации отказаться.

- Зная имена стандартных пользователей sys, system, outln и т.д., противник имеет возможность заблаговременно сгенерировать словарь. Если не весь словарь целиком, то хотя бы из наиболее часто употребительных паролей. В этом случае задача расшифровывания сводится к задаче поиска в массиве заранее вычисленных значений. А в случае, например, побитового сложения логина и пароля, знание имен стандартных пользователей пользы противнику не принесет.

- «Двухэтажную» конструкцию Oracle полезно превратить в «трехэтажную», по аналогии со стандартом ANSI X9.17. Такая конструкция существенно усложнит для криптоаналитика атаку с накоплением, а также атаку с анализом промежуточных ключей.

Кроме того, «трехэтажная» конструкция замедлит скорость вычисления хеша, что замедлит скорость силовой атаки и тем самым повысит стойкость криптосистемы. Это стандартное требование к хеш-функциям, гласящее, что «хеш-функция должна вычисляться медленно». Если хеш-функция вычисляется быстро, то хакер тоже ее вычислит быстро, и быстро осуществит перебор паролей. Медленный однонаправленный алгоритм не сильно замедлит одноразовое вычисление пароля, при подключении к БД, но существенно затруднит противнику перебор всевозможных паролей. Наиболее популярное решение – это многократное итеративное использование хеш-функций. Например, в Unix-системах однонаправленная функция шифрования применяется итеративно несколько раз, чтобы обеспечить достаточно большое время отклика.

- Не игнорировать различие между заглавными и строчными символами. В этом пункте нет вообще никаких принципиальных сложностей.

В настоящий момент количество возможных паролей есть $\sum_{i=1}^{i=30} 57^i \cong 4{,}8 \cdot 10^{52}$. После отмены преобразования к верхнему регистру множество допустимых символов увеличится с 57 до 57+26=83, и в результате количество возможных паролей возрастет до величины $\sum_{i=1}^{i=30} 83^i \cong 3{,}8 \cdot 10^{57}$. Расширение множества допустимых паролей позволит скомпенсировать на десяток-другой лет возможные успехи в вычислительной технике и криптоанализе.

- В качестве варианта по построению защищенной СУБД можно генерировать уникальное значение ключа для каждой инсталляции СУБД. Дело в том, что для некоторых ИС с повышенными требованиями к безопасности очень желательно иметь возможность скомпилировать своего собственного клиента с зашитым в нем уникальным ключом, т.е. клиента, способного подключаться только к своей «родной» БД, оснащенной таким же ключом. Понятно, что обычный клиент со стандартным ключом никогда не подключится к такой БД. Несмотря на все возникающие сложности, предлагаемая конструкция дает сильную гарантию безопасности информации в БД.

Выводы

Подводя итог, следует сказать, что реальной уязвимости на сегодняшний день противники СУБД Oracle не продемонстрировали. Каких-либо уязвимостей в криптографической конструкции не предъявлено. Новых теоретических результатов по вопросу получения несанкционированного доступа к СУБД Oracle тоже нет. Результаты по подбору паролей в [1] получены за счет эффективного применения вычислительных средств и осуществляют банальную силовую атаку на хеш. При отсутствии специализированного ПО, либо специализированных аппаратных средств, осуществить такую атаку за разумное время невозможно, точнее вероятность успеха ничтожно мала, а время атаки велико.

Тем не менее стоит подчеркнуть, что в системах с повышенными требованиями к безопасности для обеспечения надежной аутентификации следует использовать более сильные средства, чем стандартная парольная защита СУБД Oracle. Возможный вариант – настроить внешнюю аутентификацию и тем самым перенести проблему из СУБД в другое место.

Пользователям можно порекомендовать выбирать пароли, состоящие из нестандартных символов, заключая их в двойные кавычки.

Сравнивая парольные защиты в Oracle и ОС Unix, следует отметить, что ОС защищены слабее. В основных промышленных ОС – Solaris 8,9, HPUX 10,11, AIX

4,5 – в качестве пароля используются только первые 8 знаков. Проведя эксперимент, окажется, что, например, для пользователя root, пароли rootroot, rootroot1, rootroot2 и т.д. являются с точки зрения ОС одинаковыми, т.е. одинаково проходными. Так что, если вы набираете 10 знаков при подключении к серверу, то фактически вы набираете только первые 8. Спрашивается, имеет ли смысл защищать СУБД, если защита самого сервера слабее? В ОС Линукс и Solaris 10 ситуация с паролями намного лучше.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Joshua Wright, Carlos Cid.* An assessment of the Oracle password hashing algorithm.
2. http://www.red-database-security.com/

УДК 004.414.2

В.А. Михеев

ОСНОВЫ ПОСТРОЕНИЯ ПОДСИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ МНОГОФУНКЦИОНАЛЬНОЙ ИНФОРМАЦИОННОЙ СИТЕМЫ

При проектировании крупной территориально-распределенной многофункциональной информационной системы (МИС) холдинга, одной из основных задач которой является построение единого информационного пространства, необходимо учитывать, что в системе предполагается циркуляция, как общедоступной информации, так и информации ограниченного доступа.

Для обеспечения защиты информации в МИС необходима разработка и внедрение подсистемы защиты информации. Таким образом, разработка основ построения подсистемы защиты информации является актуальной задачей, позволяющей уже на этапе проектирования МИС заложить требуемый уровень защиты.

Подсистема защиты информации является неотъемлемой составной частью МИС и к ней могут предъявляются требования, аналогичные требованиям, предъявляемым к автоматизированным системам в защищенном исполнении. Общие требования по безопасности информации и порядок создания автоматизированных систем в защищенном исполнении определен в национальных стандартах [1,2].

Основные цели защиты информации в МИС должны предусматривать:

- предотвращение утечки информации по техническим каналам (в том числе за счет возможно внедренных в технические средства специальных устройств негласного получения информации);
- предотвращение несанкционированного уничтожения, искажения, копирования, блокирования информации;
- сохранение возможности управления процессом обработки, хранения и использования информации в условиях несанкционированных воздействий на защищаемую информацию.

Подсистема защиты информации должна обеспечивать комплексную защиту информации, передаваемой, накапливаемой и обрабатываемой в МИС, в том числе:

- конфиденциальность информации;
- целостность информации;
- достоверность информации;
- доступность информации;