Проектирование безопасности SQL Server

1. Конечные точки TCP

Конечные точки управляют возможностью подключения к экземпляру SQL Server и определяют допустимые способы коммуникации.

Подобно брандмауэрам в сети, конечные точки обеспечивают безопасность на границе между приложениями и экземпляром SQL Server.

1.1. Типы конечных точек и полезные данные

У конечной точки есть два важных компонента: протокол передачи данных (*TCP* или *HTTP*) и полезные данные, определяющие поддерживаемый тип трафика.

Существуют следующие типы полезных данных:

- SOAP;
- TSQL;
- SERVICE_BROKER;
- DATABASE_MIRRORING.

1.1. Типы конечных точек и полезные данные

В Таблице 1 представлены допустимые сочетания протоколов передачи данных типов полезной нагрузки для конечных точек.

Таблица 1 – Протоколы передачи данных и типы полезной нагрузки для конечных точек

Протокол передачи данных	**Тип полезной нагрузки**
TCP	TSQL
TCP	SERVICE_BROKER
TCP	DATABASE_MIRRORING
HTTP	SOAP

1.1. Типы конечных точек и полезные данные

Протокол передачи данных	Тип полезной нагрузки
TCP	TSQL
TCP	SERVICE_BROKER
TCP	DATABASE_MIRRORING
HTTP	SOAP

С помощью различных комбинаций протоколов передачи данных и типов полезной нагрузки выполняется предварительная фильтрация допустимого трафика еще до поступления команды на экземпляр SQL Server.

Пример. Пусть для конечной точки задан протокол TCP и тип полезной нагрузи TSQL.

В этом случае при попытке передачи через данную конечную точку трафика HTTP, SERVICE_BROKER, DATABASE_MIRRORING соединение будет отключено даже без проверки подлинности запроса.

1.1. Типы конечных точек и полезные данные

Протокол передачи данных	Тип полезной нагрузки
TCP	TSQL
TCP	SERVICE_BROKER
TCP	DATABASE_MIRRORING
HTTP	SOAP

Этот процесс во многом подобен работе брандмауэров в сети.

Администраторы сети настраивают брандмауэры так, чтобы разрешить передачу трафика только через заданные UDP- и TCP-порты.

Любые попытки передачи запросов через заблокированные порты брандмауэр отклоняет.

Точно также запросы, формат которых не соответствует определению конечной точки, отклоняются.

1.2. Доступ к конечным точкам

Даже если в поступающем на конечную точку трафике используются «правильные» протоколы передачи данных и тип полезной нагрузки, соединение все равно будет отключено, если не будет разрешений на доступ к данной конечной точке.

Доступ к конечной точке состоит из двух уровней:

1-й уровень. Управление доступом определяется состоянием конечной точки.

Конечная точка может находиться в одном из трех состояний:

- STARTED (запущено)
- STOPPED (остановлено)
- DISABLED (отключено)

1.2. Доступ к конечным точкам

2-й уровень управления доступом – разрешение на подключение к конечной точке.

Чтобы подключиться к конечной точке, у используемого приложением имени входа должно быть разрешение CONNECT на подключение к конечной точке.

1.3. Конечные точки TCP

У конечных точек TCP возможен один из трех типов полезной нагрузки: TSQL, DATABASE_MIRRORING и SERVICE_BROKER.

1.3.1. Аргументы конечных точек

Конечные точки TCP можно настроить на прослушивание трафика по определенным IP-адресам и портам.

Для любых конечных точек TCP можно задать следующие два аргумента:

- ✔ LISTENER_PORT
- ✔ LISTENER_IP

Аргумент LISTENER_PORT обязательный.

Конечная точка TCP для полезной нагрузки TSQL, создаваемая для каждого экземпляра в ходе установки, настроена по умолчанию на TCP-порт 1433 или другой порт для данного экземпляра.

Номера портов.

Поскольку в конечных точках DATABASE_MIRRORING по умолчанию используется TCP-порт 5022, а в конечных точках TSQL – TCP-порт 1433, то имеет смысл задавать в этих конечных точках другие номера портов.

Это позволит предотвратить возможные атаки или хотя бы затруднит их, при этом злоумышленникам придется воспользоваться сканером портов, а не просто вслепую подключиться к порту 1433 или 5022 для организации DoS-атаки («отказ в обслуживании») или другой попытки взлома.

Зачем так много усилий нужно приложить, чтобы создать соединение с экземпляром SQL Server еще до того, как начнется проверка подлинности пользователя?

В предыдущих версиях SQL Server любое приложение могло подключиться и передать любой запрос на сервер SQL Server.

Никто не проверял корректность запросов приложений, поэтому злоумышленникам было проще атаковать серверы SQL Server.

В SQL Server 2008 до постановки запроса в очередь на обработку выполняется проверка допустимости каждого запроса и отправившего его пользователя.

При попытке компрометации сервера SQL Server администраторы также могут немедленно закрыть доступ путем перевода соответствующей конечной точки в состояние DISABLED.

Аргумент LISTENER_IP является необязательным и позволяет организовать очень мощный уровень безопасности в приложении.

В конечной точке можно задать IP-адрес, который будет прослушиваться.

По умолчанию у данного аргумента значение ALL, то есть конечная точка прослушивает все соединения, поступающие на все действительные IP-адреса на данном сервере.

Задав значение аргумента LISTENER_IP, можно ограничить запросы на соединение отдельным сетевым интерфейсом.

При этом конечная точка будет прослушивать запросы, отправляемые лишь на указанный IP-адрес.

Пример. Код на языке T-SQL создает конечную точку *Database_Mirroring*:

```
CREATE ENDPOINT [Mirroring]
AS TCP (LISTENER_PORT = 5022)
FOR DATA_MIRRORING (ROLE = PARTNER, ENCRYPTION = REQUIRED);
ALTER ENDPOINT [Mirroring] STATE = STARTED;
```

При выполнении этого кода создается конечная точка, обслуживающая зеркальное сеансы зеркального отображения базы данных через порт 5022 и реагирующая на запросы с любых действительных IP-адресов.

Параметр ROLE = PARTNER означает, что в сеансах зеркального отображения, обслуживаемых данной конечной точкой, могут принимать участие лишь базы данных на текущем экземпляре SQL Server с ролью основной или зеркальной базы данных и при использовании алгоритма шифрования RC4.

Как получить информацию об имеющихся конечных точках?

Откройте среду SQL Server Management Studio, подключитесь к своей базе данных, откройте новое окно запроса и выполните следующий код:

```
SELECT * FROM sys.endpoints
SELECT * FROM sys.tcp_endpoints
SELECT * FROM sys.http_endpoints
SELECT * FROM sys.database_mirroring_endpoints
SELECT * FROM sys.service_broker_endpoints
```

1.4. Создание участников

Участники (principals) предназначены для прохождения проверки подлинности и идентификации в экземпляре сервера или в базе данных.

Участники подразделяются на две основные категории:

- Имена входа/пользователи
- Группы как на уровне экземпляра, так и на уровне базы данных.

1.4.1. Имена входа

Для доступа к экземпляру пользователь должен пройти проверку подлинности, предоставив свои учетные данные серверу SQL Server для проверки.

Чтобы пользователи могли проходить проверку подлинности, необходимо создать имена входа для соответствующего экземпляра.

В SQL Server 2008 различают пять типов имен входа:

- Стандартное имя входа SQL Server (sa);
- Имя входа Windows;
- Группа Windows;
- Сертификат;
- Ассиметричный ключ.

Администраторы баз данных создают стандартные имена входа и настраивают для них имя и пароль, которые и должны предъявлять пользователи при проверке подлинности.

Имя входа хранится в главной базе данных и сопоставляется локальному идентификатору безопасности (SID) в SQL Server.

Имя входа SQL Server также может быть сопоставлено имени входа Windows или группе Windows.

Для создания имени входа используют синтаксис:

CREATE LOGIN loginName { WITH <список параметров1> | FROM <источники>}
< список параметров 1> : : =
PASSWORD = { ‘пароль’ | hashed_password HASHED} [MUST CHANGE]
[, <option_list2> [….]]

<список параметров2> : : =
SID = sid
| DEFAULT_DATABASE = *база_данных*
| DEFAULT_LANGUAGE = *язык*
| CHECK_EXPIRATION = { ON | OFF }
| CHECK_POLICY = { ON | OFF}
| CREDENTIAL = имя_учетных данных

<sources> : : =
WINDOWS [WITH <параметры_windows> [….]]
| CERTFICATE имя_сертификата
| ASYMMETRIC KEY имя_ассиметричного_ключа

<параметры_windows> : : =
| DEFAULT_DATABASE = база_данных
| DEFAULT_LANGUAGE = язык

На время обслуживания базы данных, например пр развертывании нового кода или изменении структуры базы данных, необходимо запретить пользователям доступ к базе данных.

Для этого можно отозвать разрешения у имени входа, но впоследствии потребуется восстановить эти разрешения.

Чтобы закрыть доступ без изменений в разрешения для имени входа, можно отключить это имя входа, выполнив код:

ALTER LOGIN <имя_входа> DISABLE

1.4.2. Пользователи базы данных

Система безопасности SQL Server работает по принципу отсутствия доступа по умолчанию.

Если пользователю явно не предоставлено разрешение, он не сможет выполнить соответствующее действие.

Чтобы предоставить доступ к базе данных, нужно добавить имя входа к базе данных как пользователя, выполнив команду:

```
CREATE USER имя_пользователя
    [ { { FOR | FROM }
        { LOGIN имя_входа
            | CERTIFICATE имя_сертификата
        | ASYMMETRIC KEY имя_ассиметричного_ключа }
        | WITHOUT LOGIN ]
     [ WITH DEFAULT_SCHEMA = имя_схемы ]
```

1.4.2. Пользователи базы данных

В базе данных можно создавать пользователя, не связанного с именем входа.

Он так и называется – пользователь без имени входа (loginless user).

1.4.3. Пример создания имен входа и пользователей базы данных

1. В меню Пуск (Start) щелкните правой кнопкой Мой компьютер (My Computer) и выберите Управление (Manage).
2. Щелкните правой кнопкой узел Пользователи (Users) в папке Локальные пользователи и группы (Local Users and Groups) и выберите Новый пользователь (New User). Создайте учетную запись Windows и назовите ее TestAccount. Закройте консоль Управление компьютером (Computer Management).
3. Выполните следующий код, чтобы добавить учетную запись Windows как имя входа в свой экземпляр, указав вместо <имя_компьютера> имя компьютера, на котором работает SQL Server.

```
-- Скобки необходимы !!!
CREATE LOGIN [ <имя_компьютера>\TestAccount ] FROM WINDOWS
GO
```

1.4.3. Пример создания имен входа и пользователей базы данных

4. Выполните следующий код, чтобы создать два собственных имени входа SQL Server, указав вместо <Введите_здесь_надежный_пароль> надежный пароль, и добавив созданные учетные записи как пользователей в базу данных Firma.

```
CREATE LOGIN Test WITH PASSWORD = '<Введите_здесь_надежный_пароль> '
CREATE LOGIN Test 2 WITH PASSWORD  = '<Введите_здесь_надежный_пароль> '
GO
USE Firma
GO
CREATE USER Test FOR LOGIN Test
CREATE USER Test2 FOR LOGIN Test2
GO
```

5. Выполните следующий код, чтобы создать пользователя без имени входа в базе данных Firma:

```
USE Firma
GO
CREATE USER TestUser WITHOUT LOGIN
GO
```

1.4.3. Пример создания имен входа и пользователей базы данных

6. Выполните следующий код, чтобы просмотреть конечные точки, а также участников экземпляра и базы данных:

```
-- Участники на уровне экземпляра
SELECT * FROM sys.asymmetric_key
SELECT * FROM sys.certificate
SELECT * FROM sys.credentials
SELECT * FROM sys.linkenlogins
SELECT * FROM sys.remote_logins
SELECT * FROM sys.server_principals
SELECT * FROM sys.sql_logins
SELECT * FROM sys.endpoints
GO

-- Участники на уровне базы данных
SELECT * FROM sys.database_principals
SELECT * FROM sys.database_role_members
GO
```

7. Выполните следующий код, чтобы переименовать учетную запись sa:

```
ALTER LOGIN sa WITH NAME MySaAccount
GO
```