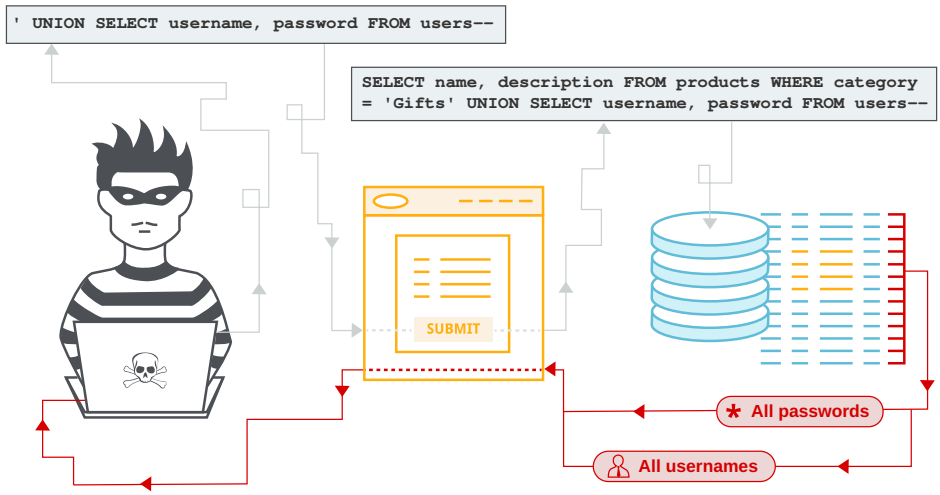
SQL-инъекция

В этом разделе мы объясним, что такое SQL-инъекция, опишем несколько распространенных примеров, объясним, как найти и использовать различные виды уязвимостей, связанных с SQL-инъекцией, и подведем итоги, как предотвратить SQL-инъекцию.



Лабораторации

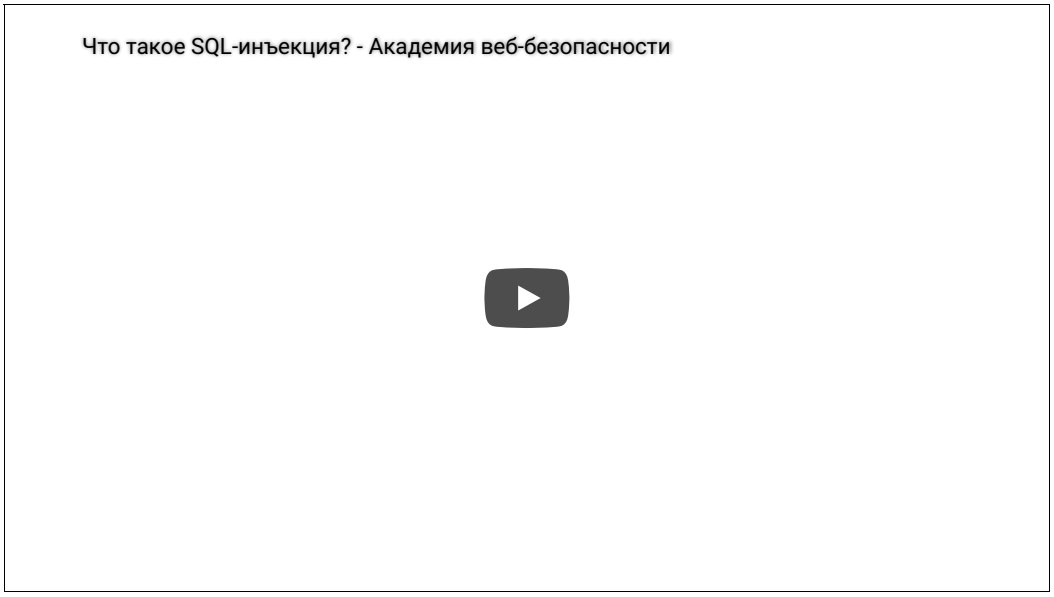
Если вы уже знакомы с основными понятиями, лежащими в основе уязвимостей SQL-инъекций, и просто хотите попрактиковаться в их использовании на реальных, преднамеренно уязвимых целях, вы можете получить доступ ко всем лабораторным работам в этом разделе по ссылке ниже.

Посмотреть все лабораторные работы по внедрению SQL >>

Что такое SQL-инъекция (SQLi)?

SQL-инъекция — это уязвимость веб-безопасности, которая позволяет злоумышленнику вмешиваться в запросы, которые приложение делает к своей базе данных. Обычно это позволяет злоумышленнику просматривать данные, которые он обычно не может получить. Это могут быть данные, принадлежащие другим пользователям, или любые другие данные, к которым может получить доступ само приложение. Во многих случаях злоумышленник может изменить или удалить эти данные, что приведет к постоянным изменениям содержимого или поведения приложения.

В некоторых ситуациях злоумышленник может эскалировать атаку путем внедрения SQL-кода, чтобы скомпрометировать базовый сервер или другую внутреннюю инфраструктуру, или выполнить атаку типа «отказ в обслуживании».



Каковы последствия успешной атаки SQL-инъекцией?

Успешная атака с внедрением SQL-кода может привести к несанкционированному доступу к конфиденциальным данным, таким как пароли, данные кредитной карты или личная информация пользователя. Многие громкие утечки данных в последние годы были результатом атак с использованием SQL-инъекций, что привело к ущербу для репутации и штрафам со стороны регулирующих органов. В некоторых случаях злоумышленник может получить постоянный черный ход в системах организации, что приведет к долгосрочной компрометации, которая может оставаться незамеченной в течение длительного периода времени.

Примеры SQL-инъекций

Существует множество уязвимостей, атак и методов SQL-инъекций, которые возникают в разных ситуациях. Некоторые распространенные примеры SQL-инъекций включают в себя:

- Извлечение скрытых данных , где вы можете изменить запрос SQL, чтобы получить дополнительные результаты.
- Подрыв логики приложения , когда вы можете изменить запрос, чтобы он мешал логике приложения.
- Атаки UNION , где вы можете получить данные из разных таблиц базы данных.
- Изучение базы данных , откуда можно извлечь информацию о версии и структуре базы данных.
- Слепая инъекция SQL , при которой результаты запроса, которым вы управляете, не возвращаются в ответах приложения.

Получение скрытых данных

Рассмотрим приложение для покупок, которое отображает товары в разных категориях. Когда пользователь нажимает на категорию «Подарки», его браузер запрашивает URL-адрес:

```
https://insecure-website.com/products?category=Gifts
```

Это заставляет приложение выполнять SQL-запрос для получения сведений о соответствующих продуктах из базы данных:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

Этот SQL-запрос просит базу данных вернуть:

- все подробности (*)
- из таблицы продуктов
- где категория Подарки
- и выпущено 1.

Ограничение `released = 1`используется для сокрытия невыпущенных продуктов. Для невыпущенных продуктов, предположительно `released = 0`.

Приложение не реализует никакой защиты от атак SQL-инъекций, поэтому злоумышленник может построить атаку следующим образом:

```
https://insecure-website.com/products?category=Gifts'--
```

Это приводит к SQL-запросу:

```
SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1
```

Ключевым моментом здесь является то, что последовательность двойного тире `--`является индикатором комментария в SQL и означает, что остальная часть запроса интерпретируется как комментарий. Это эффективно удаляет оставшуюся часть запроса, поэтому он больше не включает `AND released = 1`. Это означает, что отображаются все продукты, включая невыпущенные продукты.

Идя дальше, злоумышленник может заставить приложение отображать все продукты в любой категории, включая категории, о которых он не знает:

```
https://insecure-website.com/products?category=Gifts'+OR+1=1--
```

Это приводит к SQL-запросу:

```
SELECT * FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1
```

Модифицированный запрос вернет все товары, где либо категория «Подарки», либо 1 равно 1. Поскольку `1=1`всегда верно, запрос вернет все элементы.

ЛАБОРАТОРИЯ APPRENTICE Уязвимость

SQL-инъекции в предложении WHERE, позволяющая извлекать скрытые данные »

Подрыв логики приложения

Рассмотрим приложение, которое позволяет пользователям входить в систему с именем пользователя и паролем. Если пользователь отправляет имя пользователя `wiener`и пароль `bluecheese`, приложение проверяет учетные данные, выполняя следующий SQL-запрос:

```
SELECT * FROM users WHERE username = 'wiener' AND password = 'bluecheese'
```

Если запрос возвращает сведения о пользователе, то вход выполнен успешно. В противном случае оно отклоняется.

Здесь злоумышленник может войти в систему как любой пользователь без пароля, просто используя последовательность комментариев SQL. `--`убрать проверку пароля из `WHERE`пункт запроса. Например, отправка имени пользователя `administrator'--`и пустой пароль приводит к следующему запросу:

```
SELECT * FROM users WHERE username = 'administrator'--' AND password = ''
```

Этот запрос возвращает пользователя, чье имя пользователя `administrator`и успешно регистрирует злоумышленника как этого пользователя.

ЛАБОРАТОРИЯ APPRENTICE Уязвимость

SQL-инъекции, позволяющая обойти вход »

Получение данных из других таблиц базы данных

В случаях, когда результаты SQL-запроса возвращаются в ответах приложения, злоумышленник может использовать уязвимость SQL-инъекций для извлечения данных из других таблиц в базе данных. Это делается с помощью `UNION`ключевое слово, которое позволяет выполнить дополнительный `SELECT`запрос и добавить результаты к исходному запросу.

Например, если приложение выполняет следующий запрос, содержащий пользовательский ввод «Подарки»:

```
SELECT name, description FROM products WHERE category = 'Gifts'
```

то злоумышленник может отправить ввод:

```
' UNION SELECT username, password FROM users--
```

Это заставит приложение вернуть все имена пользователей и пароли вместе с названиями и описаниями продуктов.

Прочитайте больше

UNION-атаки SQL-инъекций »

Изучение базы данных

После первоначальной идентификации уязвимости SQL-инъекций обычно бывает полезно получить некоторую информацию о самой базе данных. Эта информация часто может проложить путь для дальнейшей эксплуатации.

Вы можете запросить сведения о версии для базы данных. То, как это делается, зависит от типа базы данных, поэтому вы можете сделать вывод о типе базы данных, исходя из того, какой метод работает. Например, в Oracle вы можете выполнить:

```
SELECT * FROM v$version
```

Вы также можете определить, какие таблицы базы данных существуют и какие столбцы они содержат. Например, в большинстве баз данных вы можете выполнить следующий запрос для получения списка таблиц:

```
SELECT * FROM information_schema.tables
```

Прочитайте больше

Проверка базы данных при атаках »
SQL-инъекциями Шпаргалка »

Уязвимости слепых SQL-инъекций

Многие экземпляры SQL-инъекций представляют собой слепые уязвимости. Это означает, что приложение не возвращает результаты SQL-запроса или сведения об ошибках базы данных в своих ответах. Слепые уязвимости по-прежнему можно использовать для доступа к несанкционированным данным, но используемые методы, как правило, более сложны и трудны в исполнении.

В зависимости от характера уязвимости и используемой базы данных для использования слепых SQL-инъекций могут использоваться следующие методы:

- Вы можете изменить логику запроса, чтобы инициировать обнаруживаемую разницу в ответе приложения в зависимости от истинности одного условия. Это может включать введение нового условия в некоторую логическую логику или условный запуск ошибки, такой как деление на ноль.
- Вы можете условно активировать временную задержку при обработке запроса, что позволит вам сделать вывод об истинности условия на основе времени, которое требуется приложению для ответа.
- Вы можете инициировать внешнее сетевое взаимодействие, используя OAST . Эта техника чрезвычайно эффективна и работает в ситуациях, когда другие техники не работают. Часто вы можете напрямую эксфильтровать данные через внеполосный канал, например, поместив данные в поиск DNS для домена, который вы контролируете.

Прочитайте больше

Слепая SQL-инъекция »

Как обнаружить уязвимости SQL-инъекций

Большинство уязвимостей SQL-инъекций можно быстро и надежно найти с помощью веб-сканера уязвимостей .

Внедрение SQL можно обнаружить вручную с помощью систематического набора тестов для каждой точки входа в приложение. Обычно это включает:

- Отправка символа одинарной кавычки `'` и поиск ошибок или других аномалий.
- Отправка определенного синтаксиса SQL, который оценивает базовое (исходное) значение точки входа и другое значение, и поиск систематических различий в результирующих ответах приложений.
- Отправка логических условий, таких как `OR 1=1` и `OR 1=2` и поиск различий в ответах приложения.
- Отправка полезных данных, предназначенных для инициирования временных задержек при выполнении в SQL-запросе, и поиск различий во времени, необходимом для ответа.
- Отправка полезных данных OAST, предназначенных для запуска внеполосного сетевого взаимодействия при выполнении в запросе SQL, и отслеживание любых результирующих взаимодействий.

Внедрение SQL в разные части запроса

Большинство уязвимостей SQL-инъекций возникают в `WHERE` пункт `SELECT` запрос. Этот тип SQL-инъекций обычно хорошо понимают опытные тестировщики.

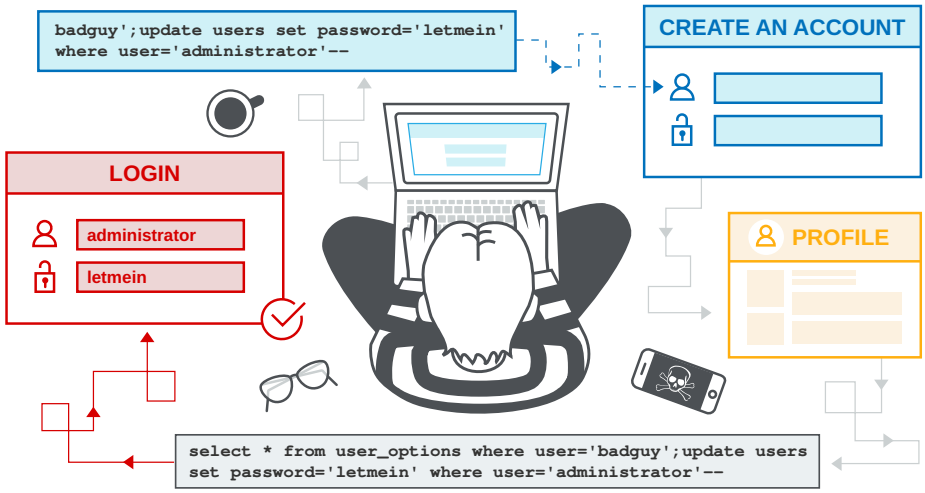
Но уязвимости SQL-инъекций в принципе могут возникать в любом месте внутри запроса и в разных типах запросов. Наиболее распространенными другими местами, где возникает SQL-инъекция, являются:

- В `UPDATE`заявления, в пределах обновленных значений или `WHERE`пункт.
- В `INSERT`операторы внутри вставленных значений.
- В `SELECT`операторов в пределах имени таблицы или столбца.
- В `SELECT`заявления, в рамках `ORDER BY`пункт.

SQL-инъекция второго порядка

SQL-инъекция первого порядка возникает, когда приложение получает пользовательский ввод из HTTP-запроса и в ходе обработки этого запроса включает ввод в SQL-запрос небезопасным способом.

В SQL-инъекциях второго порядка (также называемых хранимыми SQL-инъекциями) приложение получает пользовательский ввод из HTTP-запроса и сохраняет его для будущего использования. Обычно это делается путем помещения входных данных в базу данных, но в месте хранения данных уязвимости не возникает. Позже, при обработке другого HTTP-запроса, приложение извлекает сохраненные данные и включает их в SQL-запрос небезопасным способом.



SQL-инъекция второго порядка часто возникает в ситуациях, когда разработчики знают об уязвимостях SQL-инъекций и поэтому безопасно обрабатывают начальное размещение входных данных в базе данных. Когда данные впоследствии обрабатываются, они считаются безопасными, поскольку ранее они были безопасно помещены в базу данных. На этом этапе данные обрабатываются небезопасным образом, поскольку разработчик ошибочно считает, что им можно доверять.

Факторы, специфичные для базы данных

Некоторые основные функции языка SQL реализованы одинаковым образом на популярных платформах баз данных, поэтому многие способы обнаружения и использования уязвимостей SQL-инъекций одинаково работают в разных типах баз данных.

Однако между общими базами данных также есть много различий. Это означает, что некоторые методы обнаружения и использования SQL-инъекций работают по-разному на разных платформах. Например:

- Синтаксис для объединения строк.
- Комментарии.
- Пакетные (или сложенные) запросы.
- API для конкретных платформ.
- Сообщения об ошибках.

Прочитайте больше

Шпаргалка по SQL-инъекциям »

Как предотвратить внедрение SQL

Большинство случаев SQL-инъекций можно предотвратить, используя параметризованные запросы (также известные как подготовленные операторы) вместо объединения строк в запросе.

Следующий код уязвим для SQL-инъекций, поскольку пользовательский ввод объединяется непосредственно с запросом:

```
String query = "SELECT * FROM products WHERE category = '"+ input + "'";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
```

Этот код можно легко переписать таким образом, чтобы пользовательский ввод не мешал структуре запроса:

```
PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHER
statement.setString(1, input);
ResultSet resultSet = statement.executeQuery();
```

Параметризованные запросы можно использовать в любой ситуации, когда ненадежные входные данные отображаются как данные в запросе, включая `WHERE`пункт и значения в `INSERT`или `UPDATE`утверждение. Их нельзя использовать для обработки ненадежных входных данных в других частях запроса, таких как имена таблиц или столбцов или `ORDER BY`пункт. Функциональность приложения, которая помещает ненадежные данные в эти части запроса, должна будет использовать другой подход, например, внести в белый список разрешенные входные значения или использовать другую логику для обеспечения требуемого поведения.

Чтобы параметризованный запрос был эффективным для предотвращения SQL-инъекций, строка, используемая в запросе, всегда должна быть жестко закодированной константой и никогда не должна содержать никаких переменных данных из любого источника. Не поддавайтесь искушению решать в каждом конкретном случае, является ли элемент данных безопасным, и разрешать динамическим данным влиять в возможном строк в запросе для случаев, которые считаются безопасными. Слишком легко происхождении данных или чтобы изменения в другом коде нарушили предположения о том, какие данные испорчены.

Истории из Daily Swig о SQL-инъекциях

Радар поиска ошибок

Последние программы вознаграждения за обнаружение ошибок на июнь 2022

31 мая 2022

Программа Linkedin по поиску ошибок становится общедоступной

26 мая 2022

Проблемы на стороне сервера

Серьезная ошибка Snipe-IT, которую можно использовать для отправки электронных ловушек для сброса пароля

мая 2022

Радар поиска ошибок

Последние программы вознаграждения за обнаружение ошибок на май 2022

29 апреля 2022

Прочитайте больше

Найдите уязвимости SQL-инъекций, используя веб-сканер уязвимостей Burp Suite. »