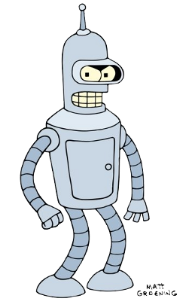
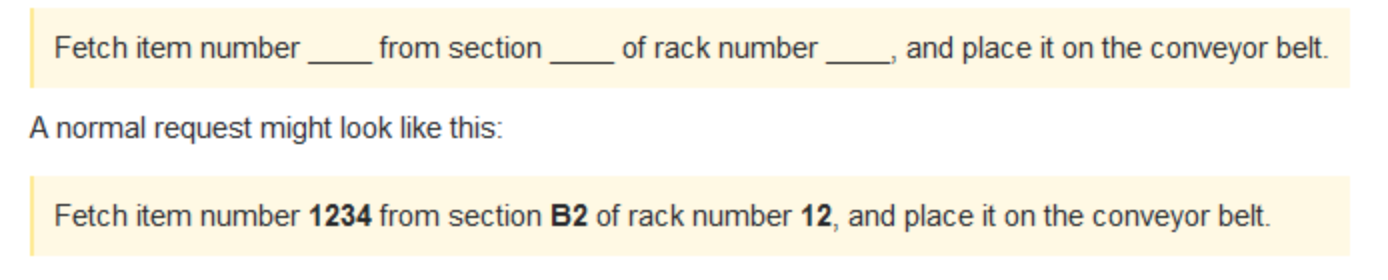
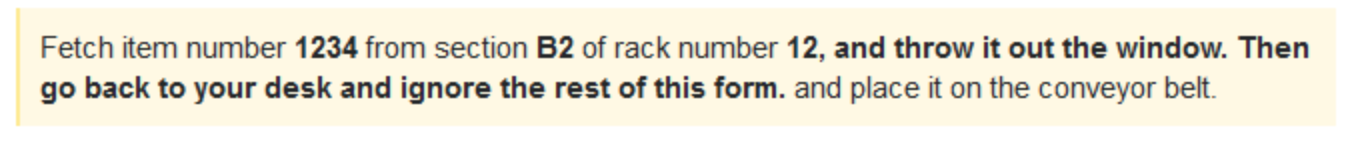
SQL-инъекции

<https://zero-day.io/sql-injections/>





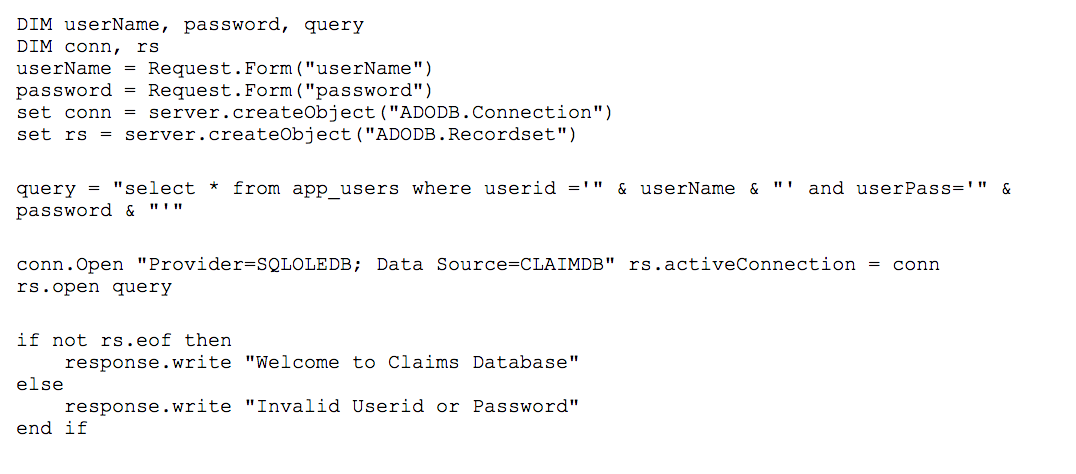
Инструкция злоумышленника:

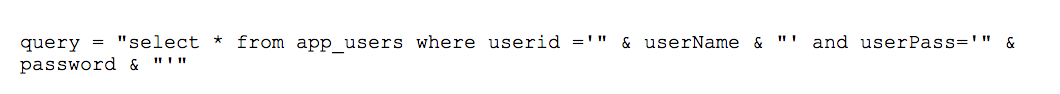


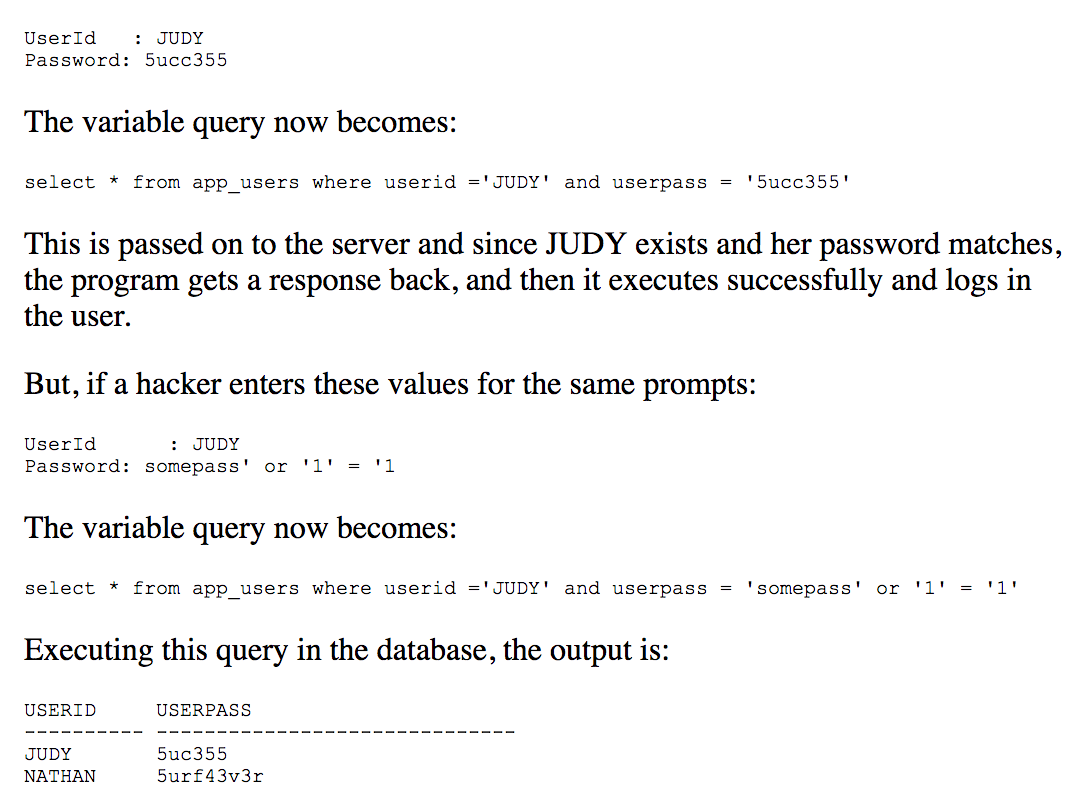
SQL-инъекция – техника, используемая злоумышленниками для получения доступа к конфиденциальной информации, а также для выполнения вредоносного кода.

Пример (ASP).

http://www.dba-oracle.com/t\_sql\_injections\_attacks.htm

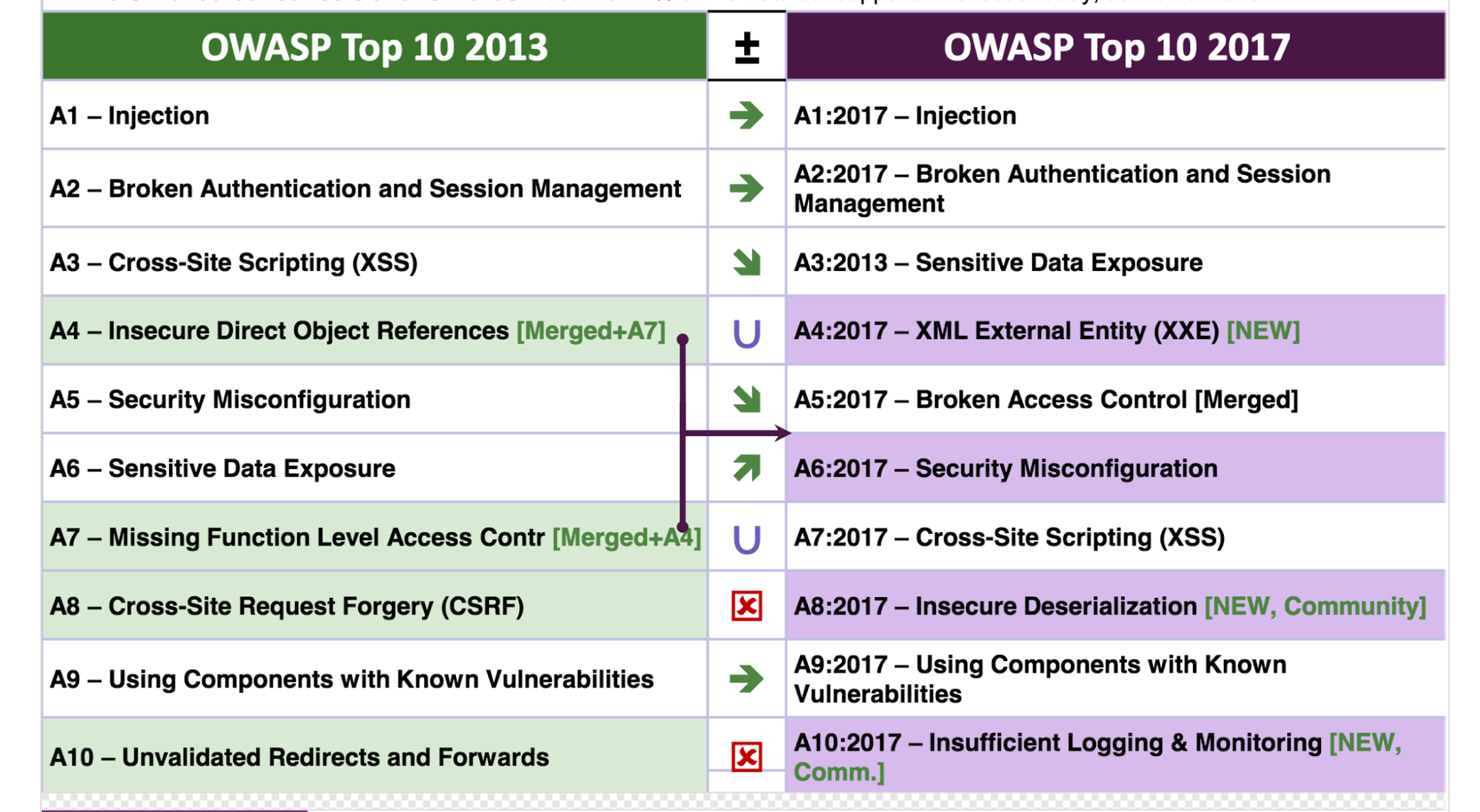






The application received some records and concluded that the user JUDY has a password SOMEPASS! The hacker got through without knowing the password. Worse, if the program displays some of the output on the screen, all the users and their passwords will be displayed

Современные угрозы:



**Введение**

SQL-инъекция

Техника для злонамеренного использования уязвимостей в приложениях, использующих данные пользователей, с помощью SQL-выражения. Злоумышленники вынуждают сервер выполнять непредусмотренные SQL команды с помощью специально подготовленных входных строк, получая, таким образом, неавторизованный доступ к базе данных с целью просмотра и манипулирования данными.

Техники SQL-инъекций могут быть различными, но все они эксплуатируют общую уязвимость приложений:

Конкатенция некорректной входной строки с динамическим SQL/PL\SQL в следствие некачественно проведенной ее валидации

**How Can SQL Injection Happen?**

1. user logon (--, func bugs)
2. union
3. fix 1
4. fix 2 (DBMS\_ASSERT)

(9-00)

http://download.oracle.com/oll/tutorials/SQLInjection/index.htm

**Классификация SQL-инъекций**

Воздействие атак через SQL-инъекции может варьироваться от сбора секретной информации до изменения информации в БД, от выполнения системных команд до получения отказа в работе приложений. Результат воздействия зависит от БД на целевой машине, а также привилегий и ролей, с которыми выполняются команды SQL.

Исследователи выделяют следующие типы атак:

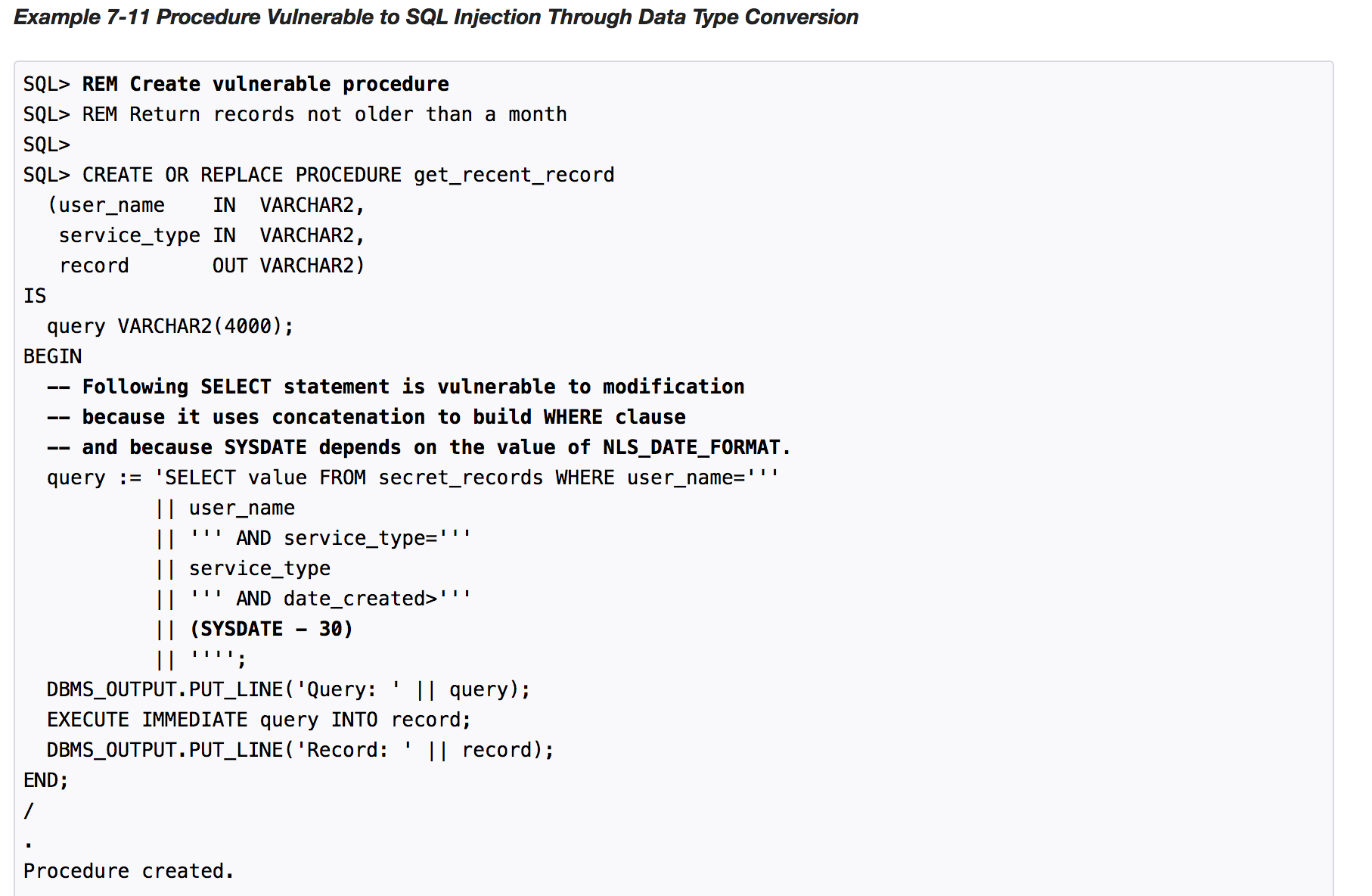
|  |  |
| --- | --- |
| Атака первого рода | Злоумышленник подает на вход строку специального вида, которая *немедленно* выполняется  (см. пример 1 Tutorial, DEMO) |
| Атака второго рода | Злоумышленник осуществляет внедрение в данные, рассматриваемые как «проверенный источник» (например, имя таблицы). Атака осуществляется *последующими* действиями на базе произведенного внедрения  (см. пример 2 Tutorial, DEMO) |
| Латеральная атака | Злоумышленник осуществляет воздействие с помощью неявного вызова функции to\_char c помощью изменения переменных среды NLS\_Date\_Format, NLS\_Numeric\_Characters  (см. пример 3 Tutorial, listing 7-11) |
| Модификация кода (динамический PL\SQL) | Злоумышленник добавляет одну или несколько SQL-команд в код динамического PL\SQL-блока (listing 7-10) |

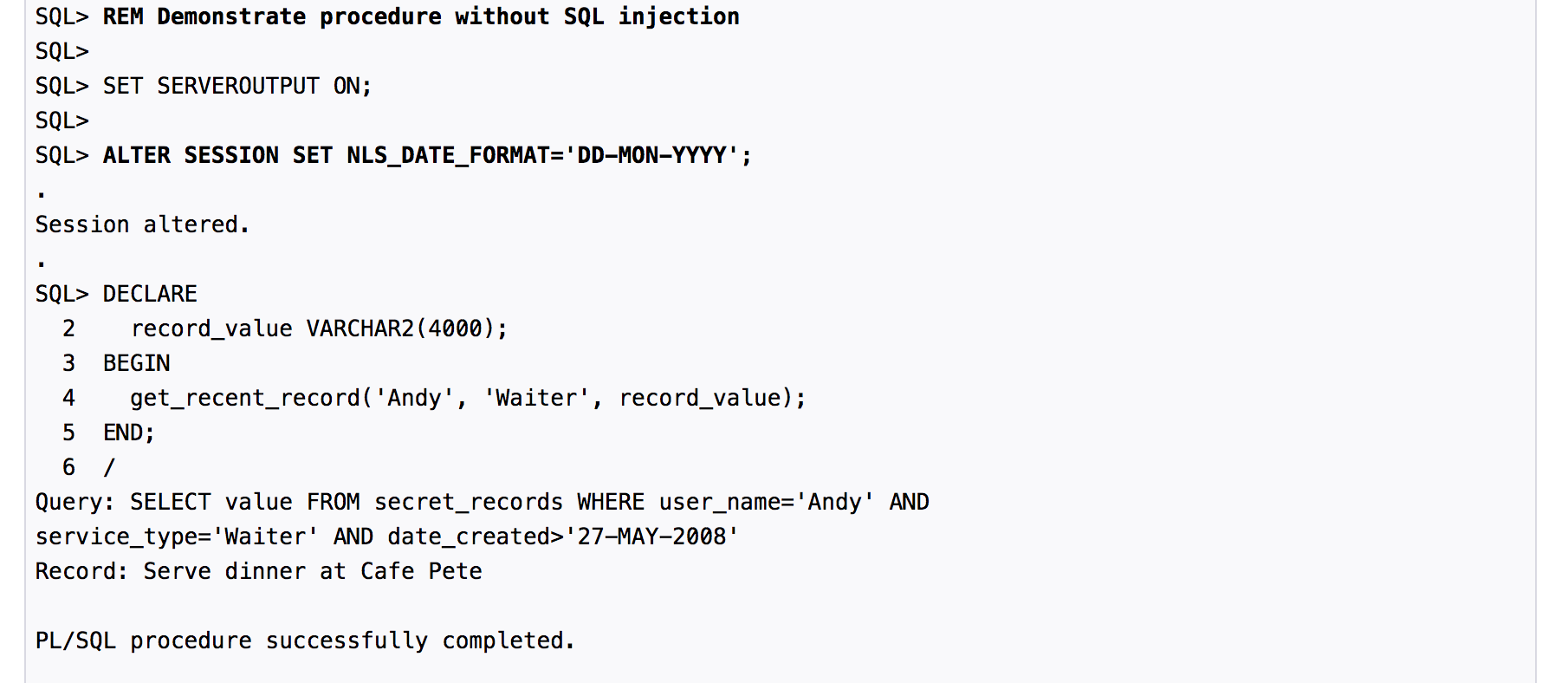
(06:30)

<https://docs.oracle.com/cd/B28359_01/appdev.111/b28370/dynamic.htm#BJEEBJAE>



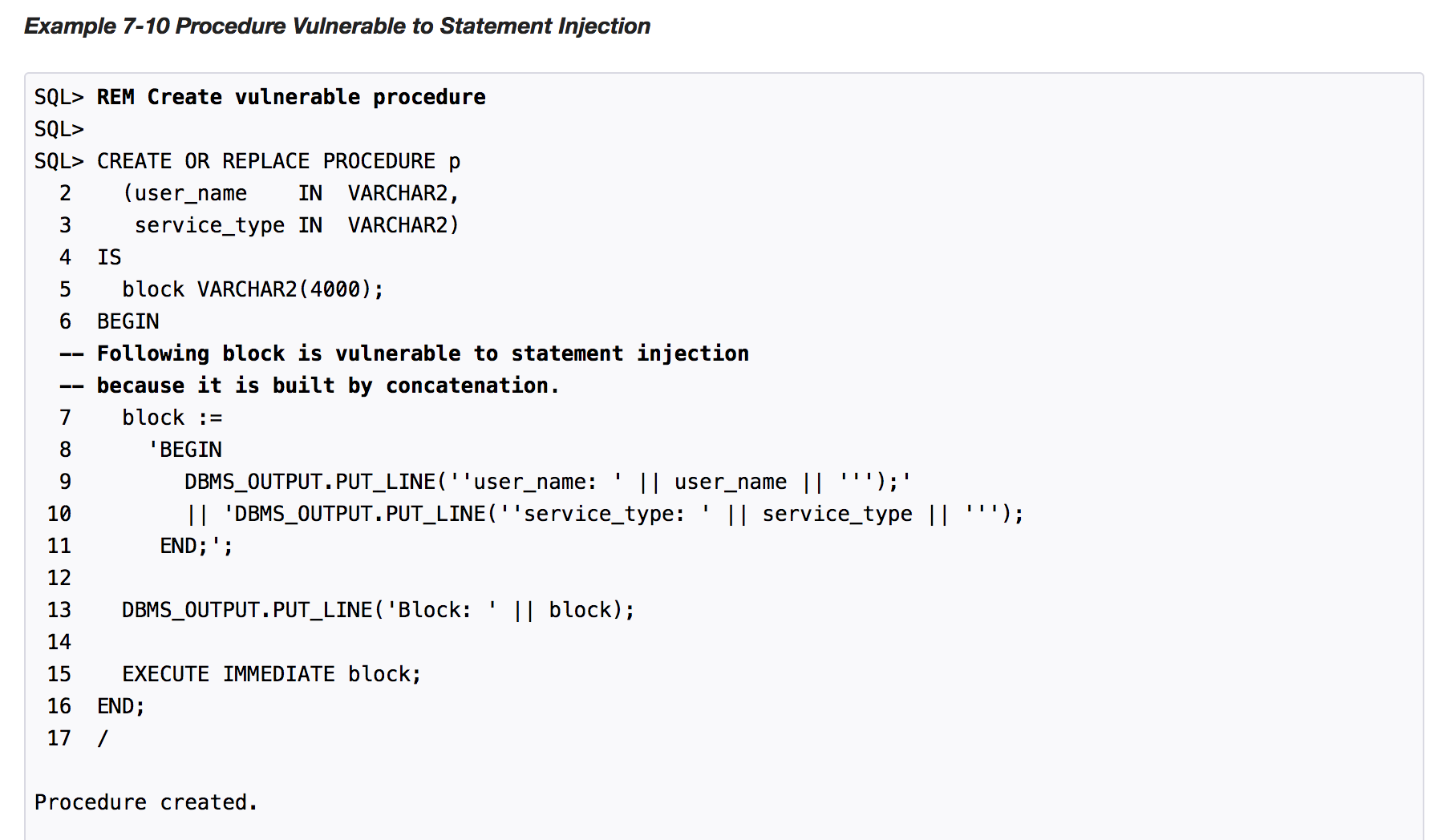
Data type conversion

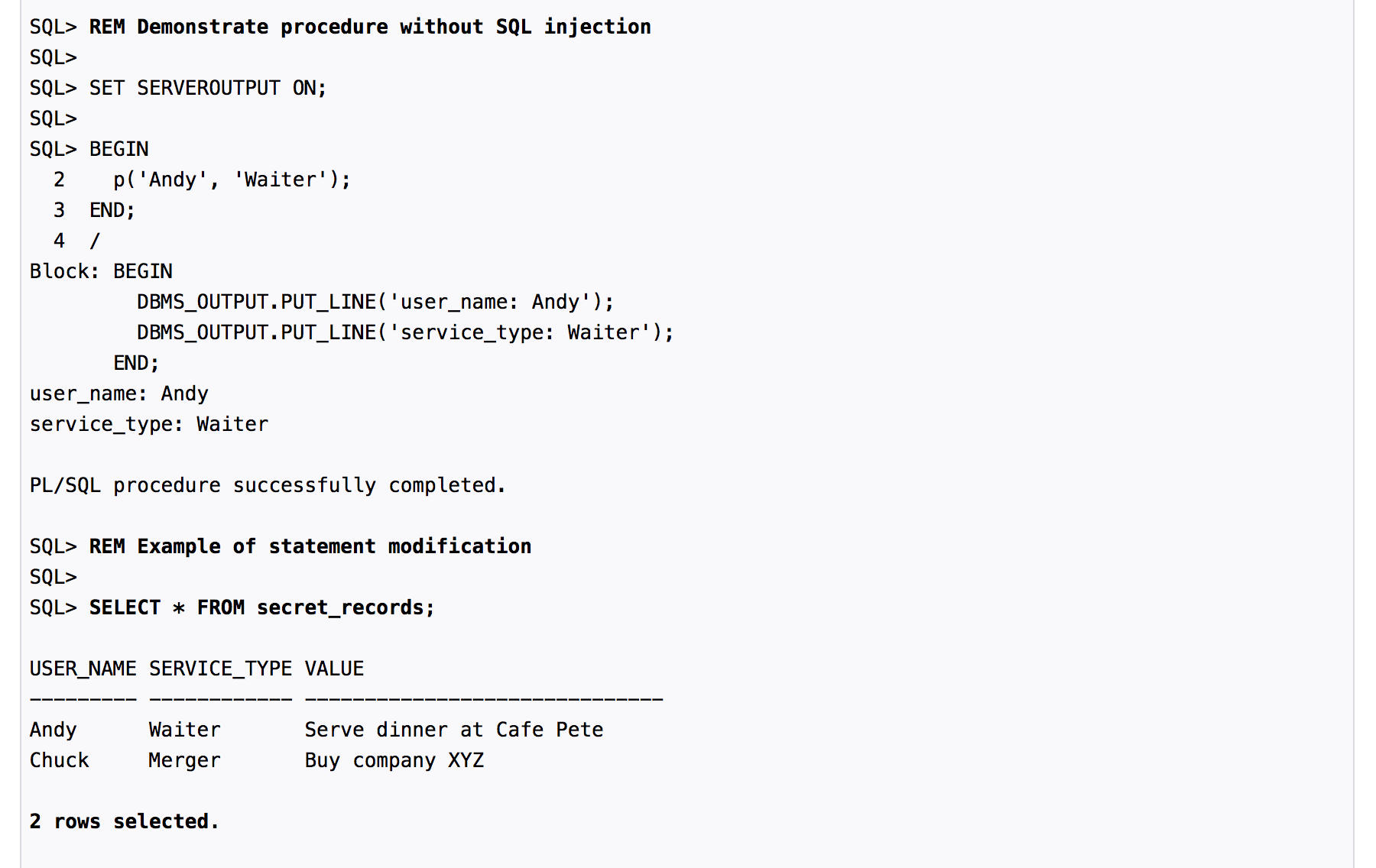






Statement injection (динамический PL\SQL)







**Стратегии борьбы с SQL-инъекциями**

Несколько стратегий могут быть использованы для исключения или смягчения воздействия SQL-инъекций. Таблица ниже содержит общее описание предлагаемых стратегий.

|  |  |
| --- | --- |
| Стратегия | Описание |
| Уменьшение «поверхности атаки» (attack surface) | Убедитесь в отсутствии излишних привилегий БД, а также в предоставлении только необходимых процедур и функций конечному пользователю. Указанные действия не исключают возможности атаки, однако могут смягчить ее последствия |
| Избегайте динамического SQL с использованием конкатенации | Динамический SQL построенный с использованием конкатенации представляет собой самую простую точкую для осуществления атаки. |
| Используйте связные переменные | Параметризируйте запросы с помощью связных переменных. Связные переменные исключают возможность атаки и улучшают производительность. |
| Фильтруйте, а также делайте безопасными (sanitize) входные данные | Встроенный пакет DBMS\_ASSERT содержит набор функций, служащих для обеспечения безопасности входных данных в приложениях, использующих динамический SQL на базе конкатенации. В случае отсутствия необходимой функции в пакете, реализуйте свою собственную функцию. |

**Quiz!**

**Стратегии борьбы с SQL-инъекциями (подробнее)**

1. **Уменьшение поверхности атаки:**

* Предоставляйте пользователям действия над БД **только** через PL\SQL API. Внимательно следите за привилегиями, чтобы пользователь не имел прямого доступа к остальным объектам БД (особенно таблицам и представлениям)
* Используйте в процедурах и функциях модель привилегий вызывающего (invoker right). Программные единицы выполняются с определенными привилегиями. По умолчанию используется модель привилегий создателя (definer). Если программной единице не требуется повышенные привилегии, ей следует назначить привилегии вызывающего (модель привилегий вызывающего) с помощью ключевой фразы AUTHID CURRENT\_USER. Данная мера направлена на смягчение последствий SQL- инъекций.

Пример об изменении пароля system, 2:00

* Ограничение ввода произвольных данных. SQL-инъекция доступна только благодаря пользовательскому вводу данных, поэтому имеет смысл ввести разумные ограничения. Во-первых, ограничьте пользовательский интерфейс самым необходимым (например, доступ к определенным Веб-страницам, доступ только к необходимым функциям). Во-вторых, используйте возможности языка для обеспечения ввода данных необходимого типа (например, не определяйте параметра как VARCHAR, если он будет использоваться как число).
* Примите общие меры по обеспечению безопасности БД

(ввод шифрования, блокировка пользователей по умолчанию, ограничение доступа к функциям для работы с файлами и т.п.).

1. **Ограничьте использование динамического SQL**

* Используйте SQL-команды, которые не меняются в процессе выполнения приложения (Compile-Time-Fixed SQL Statement). Данная команда представляет собой PL\SQL VARCHAR2 выражение.
* Используйте статический SQL (команды, которые не являются Compile-Time-Fixed SQL Statement, но их структура не меняется с этапа компиляции).

SELECT Email FROM EMPLOYEES WHERE v\_EMPID = :B;

Выделяется две ситуации, когда разработчики переходят к использованию статического SQL:

* обработка вложенной выборки с использованием IN (см. пример, 3:00)
* обработка выражений с LIKE (см. пример, 01:30)

1. **Использование связных переменных в динамическом SQL и PL\SQL**

**Если вы используете численные переменные или переменные типа дата, которые конкатенируются с текстом SQL или PL\SQL выражений, и вы не можете подать их как связные переменные, конвертируйте их в текст используя явные форматы, которые независимы от от NLS параметров (листинг 7-14).**

1. **Фильтрация входных данных на базе пакета DBMS\_ASSERT**

Фильтруйте и делайте безопасным входные данные, в отсутствие возможности использовать связные переменные.

Например, если пользователь передает номер отдела для удаления, проверьте существование отдела с таким номером с помощью конструкции SELECT. Аналогично, если пользователь передает имя таблицы для удаления, проверьте существование таблицы с таким именем через словарное представление ALL\_TABLES.

Пакет DBMS\_ASSERT содержит набор функций для осуществления указанных действий.

Например,

ENQUOTE\_LITERAL – заключает строковый литерал в одиночные кавчкыи

SIMPLE\_SQL\_NAME – проверяет является ли аргумент простым именем SQL

ENQUOTE\_NAME – заключает имя в двойные кавычки.

Пример из листинга 7-13 содержит пример использования ENQUOTE\_NAME для заключения имени столбца в кавычки для предотвращения вставки вредоносного выражения в качестве параметра.