

Università degli Studi di Salerno



Dipartimento di Informatica

# Penetration Testing & Ethical Hacking

---

## Target Discovery

Arcangelo Castiglione  
arcastiglione@unisa.it

# Outline

---

- Concetti Preliminari
- Identificare le Macchine Target Attive
- Operating System (OS) Fingerprinting

# Outline

---

- **Concetti Preliminari**
- Identificare le Macchine Target Attive
- Operating System (OS) Fingerprinting

# Target Discovery

## Obiettivi e Motivazioni

- Dopo aver raccolto informazioni sull'asset in esame (tramite la fase di *Information Gathering*) il passo successivo è quello di scoprire ed analizzare le macchine target (host) attive (*alive*) in tale asset



# Target Discovery

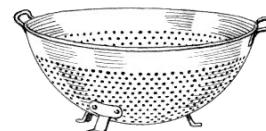
## Obiettivi e Motivazioni

---

**1. Individuare quali macchine** ( dette *target*) risultano essere «attive» («raggiungibili») all'interno dell'asset (*rete target*)

- Se una macchina target non risulta attiva, allora **non sarà possibile** effettuare il **penetration testing** su di essa
  - Pertanto sarà necessario considerare un'altra macchina

**2. Individuare il Sistema Operativo delle macchine** che risultano essere **attive**



# Outline

---

- Concetti Preliminari
- **Identificare le Macchine Target Attive**
- Operating System (OS) Fingerprinting

# Identificare le Macchine Target

## Principali Strumenti

---

- Gli strumenti (comandi) inclusi in questa categoria permettono di «identificare» le **macchine target attive**
  - Tali macchine saranno poi analizzate durante fasi successive del processo di penetration testing
- 
- **N.B.** Prima di iniziare il processo di identificazione è fondamentale conoscere i termini e gli accordi stipulati con chi ha commissionato il test

# Identificare le Macchine Target

## Principali Strumenti

---

- ping e ping6
- fping
- nmap
- arping
- arp-scan
- hping3
- nping

**Principali comandi per  
l'identificazione delle macchine  
target attive**

# Identificare le Macchine Target

## Principali Strumenti

---

➤ **ping e ping6**

➤ fping

➤ nmap

➤ arping

➤ arp-scan

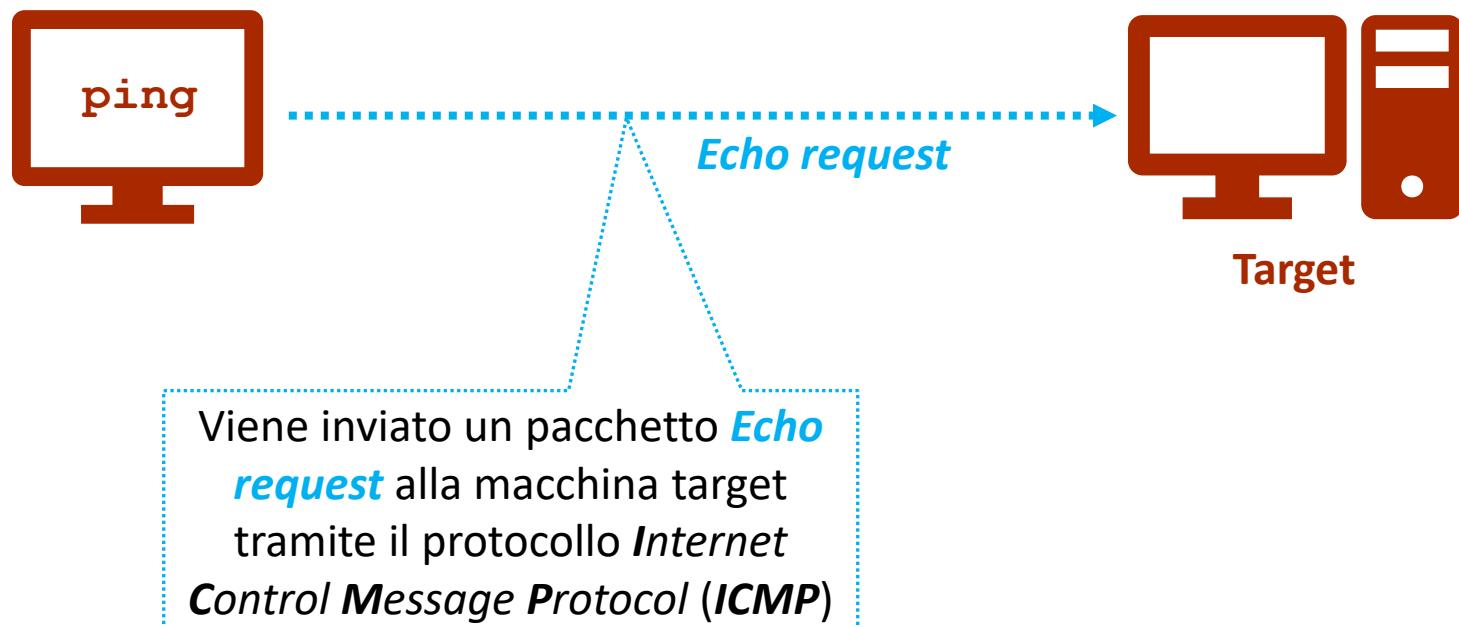
➤ hping3

➤ nping

# Il comando ping

## Logica di Funzionamento

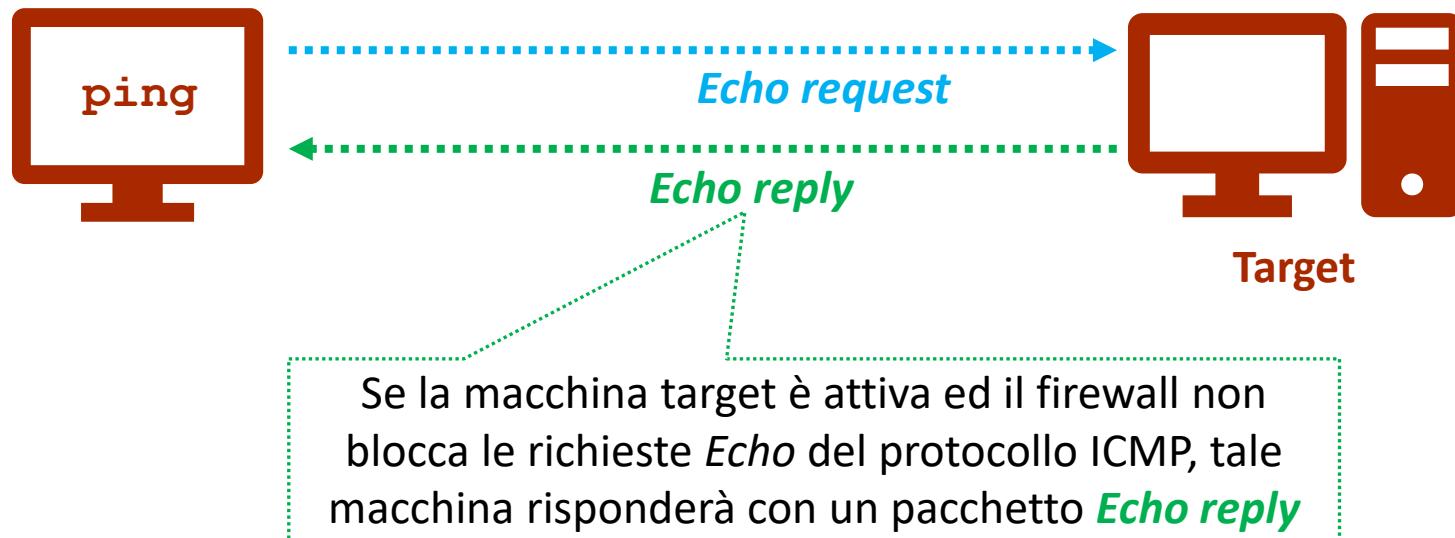
- Il **ping** è lo strumento più noto per verificare se una determinata macchina target è attiva



# Il comando ping

## Logica di Funzionamento

- Il **ping** è lo strumento più noto per verificare se una determinata macchina target è attiva



# Il comando ping

## Esempio di Esecuzione

```
root@kali:~# ping www.google.it
PING www.google.it (64.233.176.94) 56(84) bytes of data.
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=1 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=2 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=3 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=4 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=5 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=6 ttl=41 time=145 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=7 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=8 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=9 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=10 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=11 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=12 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=13 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=14 ttl=41 time=146 ms
^C
--- www.google.it ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 30ms
rtt min/avg/max/mdev = 142.538/143.799/145.661/0.922 ms
root@kali:~#
```

Per eseguire il comando **ping** è sufficiente digitare **ping** e l'indirizzo (o l'hostname) della macchina target

# Il comando ping

## Esempio di Esecuzione

```
root@kali:~# ping www.google.it
PING www.google.it (64.233.176.94) 56(84) bytes of data.
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=1 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=2 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=3 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=4 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=5 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=6 ttl=41 time=145 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=7 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=8 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=9 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=10 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=11 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=12 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=13 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=14 ttl=41 time=146 ms
^C
--- www.google.it ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 30ms
rtt min/avg/max/mdev = 142.538/143.799/145.661/0.922 ms
root@kali:~#
```

Per eseguire il comando **ping** è sufficiente digitare **ping** e l'indirizzo (o l'hostname) della macchina target

# Il comando ping

## Esempio di Esecuzione

```
root@kali:~# ping www.google.it
PING www.google.it (64.233.176.94) 56(84) bytes of data.
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=1 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=2 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=3 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=4 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=5 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=6 ttl=41 time=145 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=7 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=8 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=9 ttl=41 time=144 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=10 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=11 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=12 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=13 ttl=41 time=143 ms
64 bytes from yw-in-f94.1e100.net (64.233.176.94): icmp_seq=14 ttl=41 time=146 ms
^C
--- www.google.it ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 30ms
rtt min/avg/max/mdev = 142.538/143.799/145.661/0.922 ms
root@kali:~#
```

Di default il comando **ping** viene eseguito ripetutamente fino all'interruzione da parte dell'utente mediante **CTRL+C**

# Il comando ping

## Possibili Opzioni

---

- Esistono diverse opzioni per il comando **ping**
- Le opzioni principalmente utilizzate sono le seguenti
  - **-c <N>**
  - Specifica il numero **N** di *Echo request* che verranno inviate

# Il comando ping

## Possibili Opzioni

---

- Esistono diverse opzioni per il comando **ping**
- Le opzioni principalmente utilizzate sono le seguenti
  - **-I <argomento>**
    - Specifica l'interfaccia di rete dell'indirizzo sorgente
      - L'argomento può essere di due tipologie
        1. Un indirizzo IP (ad esempio, **192.168.14.123**)
        2. Il nome di un'interfaccia di rete (ad esempio, **eth0**)

# Il comando ping

## Possibili Opzioni

---

- Esistono diverse opzioni per il comando **ping**
- Le opzioni principalmente utilizzate sono le seguenti
  - **-s <numero\_di\_byte>**
    - Specifica il numero di byte che devono essere inviati per ciascuna richiesta echo
    - Di default, in Kali Linux il valore è impostato a 56 byte

# Il comando ping

## Possibili Opzioni

---

- Esistono diverse opzioni per il comando **ping**
- Le opzioni principalmente utilizzate sono le seguenti
  - **-s <numero\_di\_byte>**
    - Specifica il numero di byte che devono essere inviati per ciascuna richiesta echo
    - Di default, in Kali Linux il valore è impostato a 56 byte

### Osservazione

**ping Macchina\_Target** equivalente a **ping -s 56 Macchina\_Target**

# Il comando ping

## Possibili Opzioni

- Esistono diverse opzioni per il comando **ping**
- Ulteriori informazioni sul comando **ping** possono essere ottenute mediante la relativa man page
  - **man ping**

```
PING(8)                               iputils                               PING(8)

NAME
    ping - send ICMP ECHO_REQUEST to network hosts

SYNOPSIS
    ping [-aAbBdDfhLnOqrRUvV46] [-c count] [-F flowlabel] [-i interval]
          [-I interface] [-l preload] [-m mark] [-M pmtudisc option]
          [-N nodeinfo option] [-w deadline] [-W timeout] [-p pattern] [-Q tos]
          [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp option] [hop ... ]
          {destination}

DESCRIPTION
    ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an
    ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams
    ("pings") have an IP and ICMP header, followed by a struct timeval and
    then an arbitrary number of "pad" bytes used to fill out the packet.
```

# Il comando ping

## Esempio 1

---

- Supponiamo di
  - Possedere la lista degli indirizzi IP (o degli hostname) delle macchine da analizzare (*macchine target*)
    - Lista ottenuta mediante la fase di *Information Gathering*
- **Obiettivo:** controllare quale di queste macchine sarà analizzabile dal pentester
  - Controllare quindi se tali macchine sono attive e raggiungibili dal pentester

# Il comando ping

## Esempio 1

---

- Supponiamo che
  - La macchina target (ad es., Metasploitable 2) abbia indirizzo IP **10.0.2.5**
  - La macchina Kali Linux abbia indirizzo IP **10.0.2.15**
- Per verificare la disponibilità della macchina target è possibile utilizzare il seguente comando
  - **ping -c 1 10.0.2.5**

# Il comando ping

## Esempio 1

---

- Per verificare la disponibilità della macchina target è possibile utilizzare il seguente comando
  - `ping -c 1 10.0.2.5`

```
File Edit View Search Terminal Help
root@kali:~# ping -c 1 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.645 ms

--- 10.0.2.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.645/0.645/0.645/0.000 ms
root@kali:~#
```

# Il comando ping

## Esempio 1

```
File Edit View Search Terminal Help
root@kali:~# ping -c 1 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.645 ms

--- 10.0.2.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.645/0.645/0.645/0.000 ms
root@kali:~#
```

### ➤ Osservazioni

# Il comando ping

## Esempio 1

```
File Edit View Search Terminal Help
root@kali:~# ping -c 1 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.645 ms

--- 10.0.2.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.645/0.645/0.645/0.000 ms
root@kali:~#
```

### ➤ Osservazioni

- È stata inviata una sola *Echo request*

# Il comando ping

## Esempio 1

```
File Edit View Search Terminal Help
root@kali:~# ping -c 1 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.645 ms

--- 10.0.2.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.645/0.645/0.645/0.000 ms
root@kali:~#
```

### ➤ Osservazioni

- È stata inviata una sola *Echo request*
- È stata ricevuta un'unica *Echo reply*

# Il comando ping

## Esempio 1

```
File Edit View Search Terminal Help
root@kali:~# ping -c 1 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.645 ms

--- 10.0.2.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.645/0.645/0.645/0.000 ms
root@kali:~#
```

### ➤ Osservazioni

- È stata inviata una sola *Echo request*
- È stata ricevuta un'unica *Echo reply*
- Nessun pacchetto è andato perso

# Il comando ping

## Esempio 1

```
File Edit View Search Terminal Help
root@kali:~# ping -c 1 10.0.2.5
PING 10.0.2.5 (10.0.2.5) 56(84) bytes of data.
64 bytes from 10.0.2.5: icmp_seq=1 ttl=64 time=0.645 ms

--- 10.0.2.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.645/0.645/0.645/0.000 ms
root@kali:~#
```

### ➤ Osservazioni

- È stata inviata una sola *Echo request*
- È stata ricevuta un'unica *Echo reply*
- Nessun pacchetto è andato perso

Utilizzando il network sniffer *Wireshark* vediamo cosa transita in rete

# Il comando ping

## Esempio 2 – Ping e Wireshark

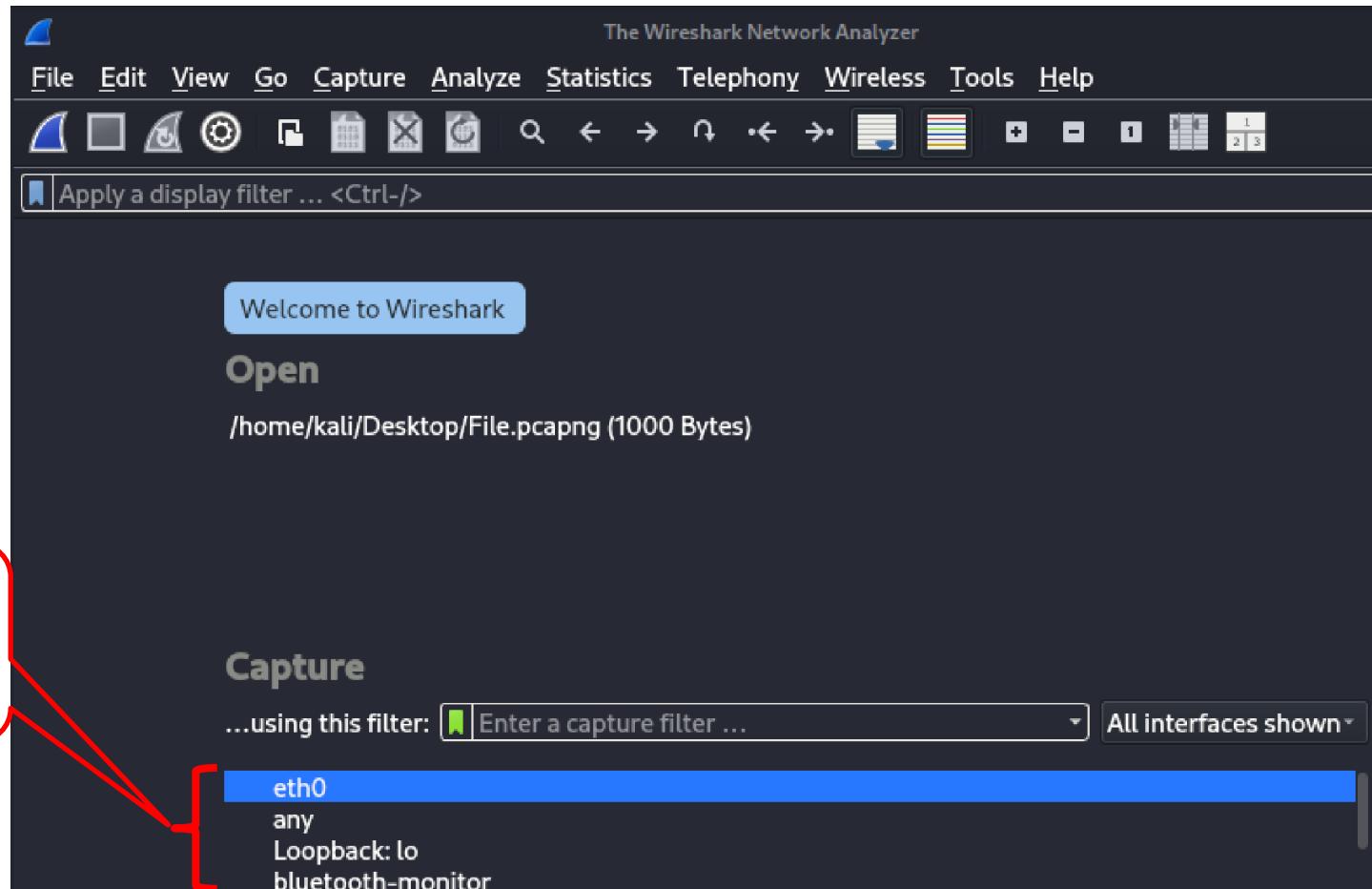
---

- **Wireshark:** Popolare analizzatore di protocolli di rete
  
- Strumento già presente in Kali Linux e ben documentato
  - <https://www.wireshark.org/download/docs/user-guide.pdf>
  
- Utilizzabile in due modalità
  - Grafica, attraverso la sezione «09 – Sniffing & Spoofing di Kali Linux» di Kali Linux
  - Da Terminale, digitando **wireshark**



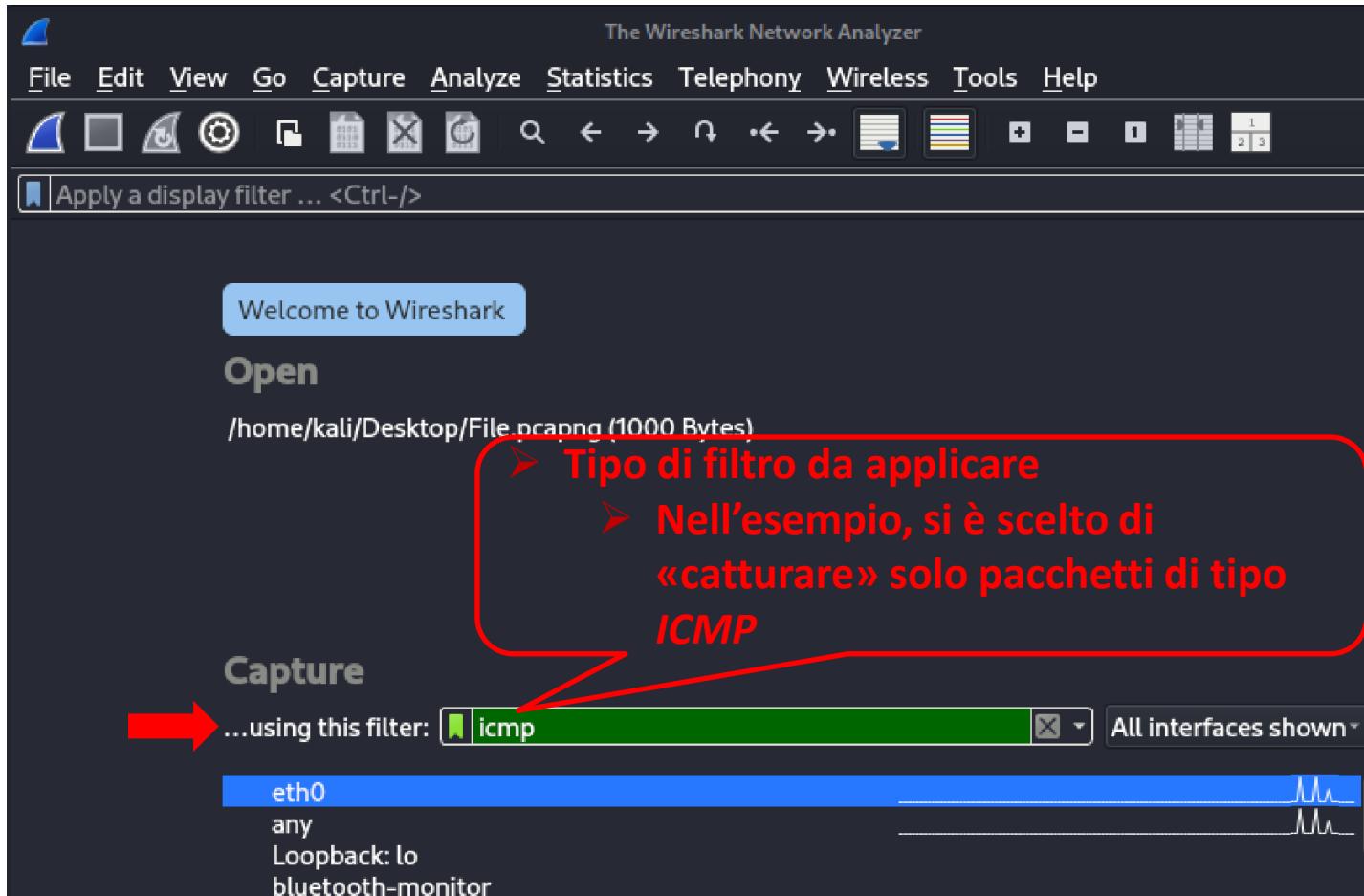
# Il comando ping

## Esempio 2 – Ping e Wireshark



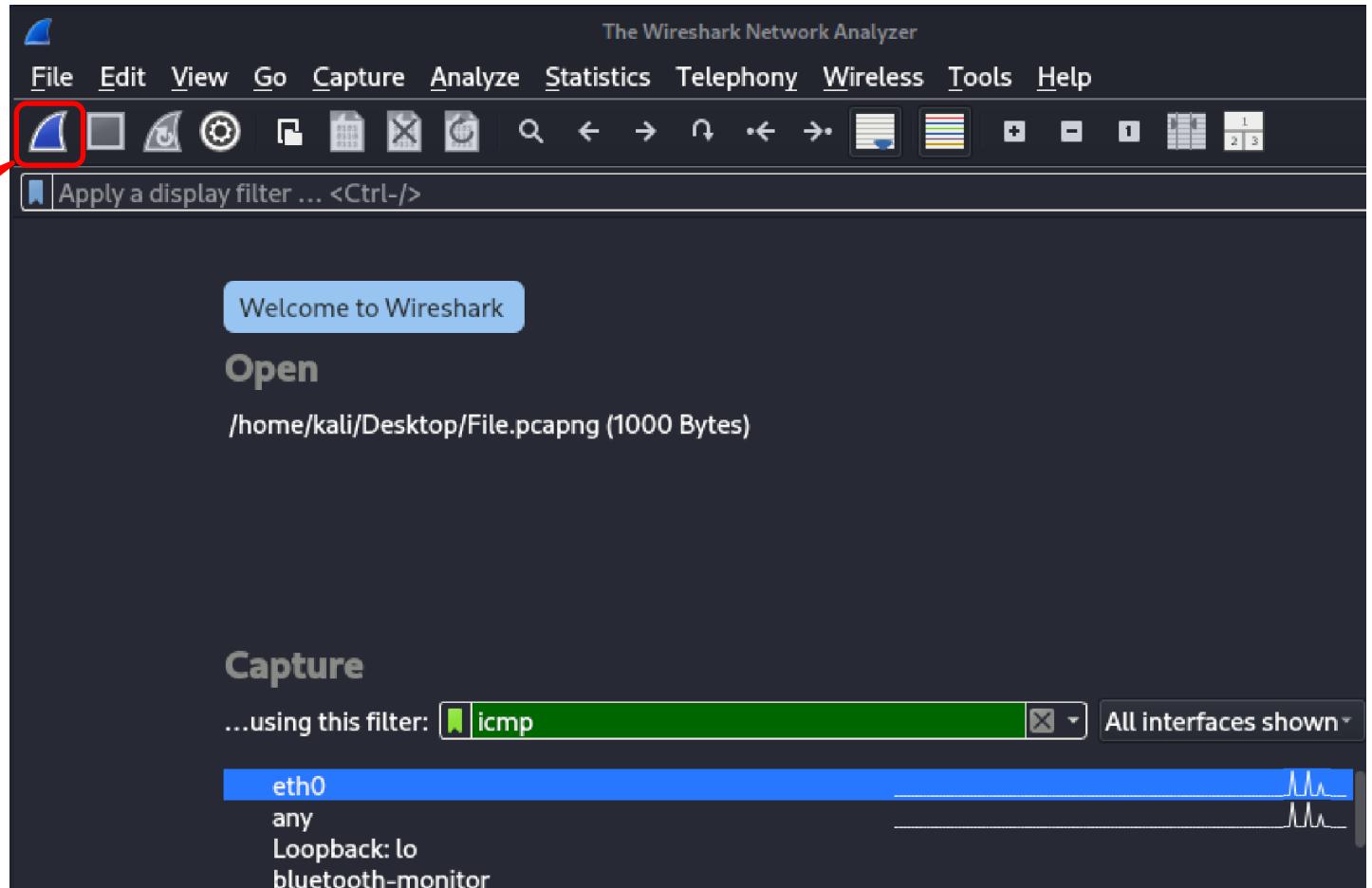
# Il comando ping

## Esempio 2 – Ping e Wireshark



# Il comando ping

## Esempio 2 – Ping e Wireshark

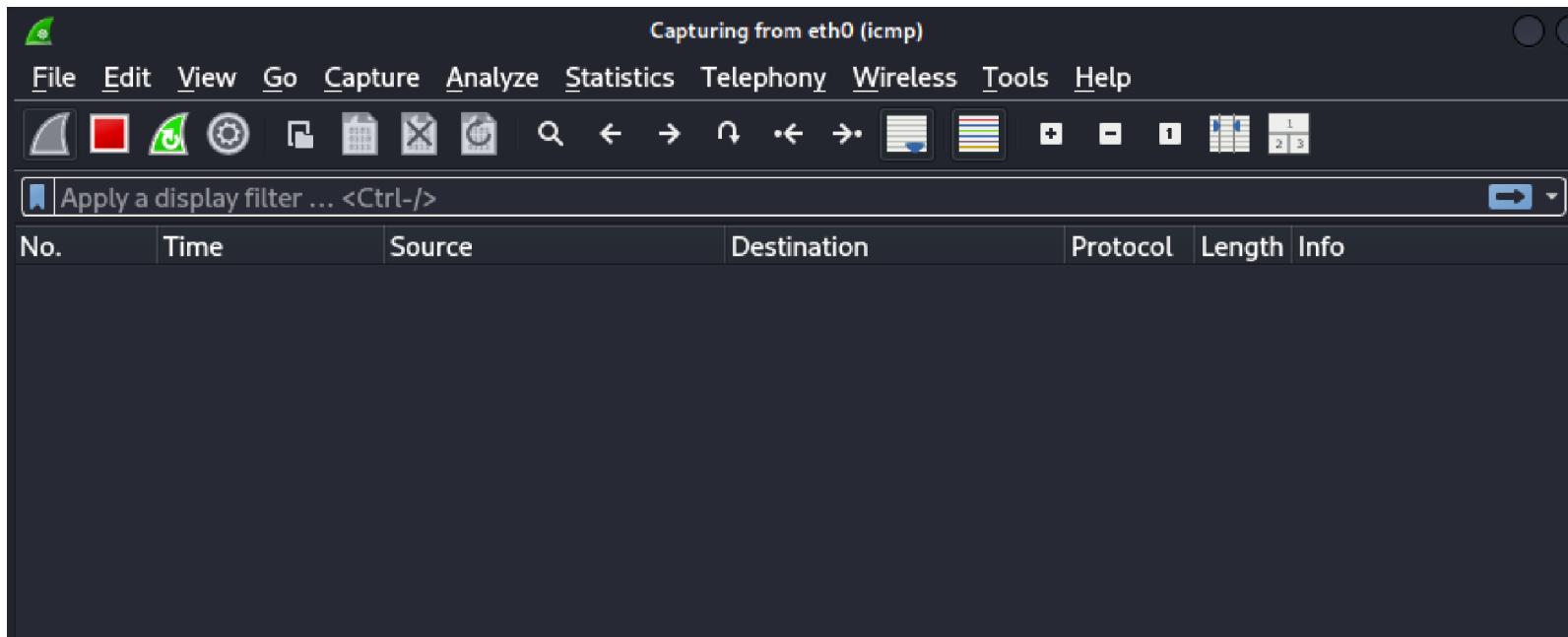


Per avviare la cattura dei pacchetti

Target Discovery

# Il comando ping

## Esempio 2 – Ping e Wireshark



**Wireshark resta in attesa di pacchetti di tipo *ICMP***

# Il comando ping

## Esempio 2 – Ping e Wireshark

---

- Eseguiamo il seguente comando dal Kali

- `ping -c 1 10.0.2.4`

```
(kali㉿kali)-[~]
└─$ ping -c 1 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=7.84 ms

--- 10.0.2.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.835/7.835/7.835/0.000 ms
```

# Il comando ping

## Esempio 2 – Ping e Wireshark

Source	Destination	Protocol	Length	Info
10.0.2.15	10.0.2.4	ICMP	98	Echo (ping) request
10.0.2.4	10.0.2.15	ICMP	98	Echo (ping) reply

- Analizzando i pacchetti tramite Wireshark è possibile osservare che
  - È stata inviata una richiesta **Echo (ping) request** dall'indirizzo IP sorgente **10.0.2.15**, mediante il protocollo **ICMP**
  - È stata ricevuta una risposta **Echo (ping) reply** dall'indirizzo IP di destinazione **10.0.2.4**
- **Osservazione:** Poiché la macchina target è attiva e non filtra i pacchetti *ICMP Echo request*, essa invierà un pacchetto *ICMP Echo Reply* alla macchina Kali



# Il comando ping

## Esempio 2 – Ping e Wireshark

The screenshot shows a Wireshark capture window. At the top, there is a table with columns: Source, Destination, Protocol, Length, and Info. Two rows of ICMP traffic are listed:

Source	Destination	Protocol	Length	Info
10.0.2.15	10.0.2.4	ICMP	98	Echo (ping) request id=0x001c, seq=1/256,
10.0.2.4	10.0.2.15	ICMP	98	Echo (ping) reply id=0x001c, seq=1/256,

A red arrow points to the first row. Below the table, the details pane shows the ICMP request message:

- Type: 8 (Echo (ping) request)
- Code: 0
- Checksum: 0x84a9 [correct]
- [Checksum Status: Good]
- Identifier (BE): 28 (0x001c)
- Identifier (LE): 7168 (0x1c00)
- Sequence Number (BE): 1 (0x0001)
- Sequence Number (LE): 256 (0x0100)
- [Response frame: 2]
- Timestamp from icmp data: Apr 14, 2025 14:54
- [Timestamp from icmp data (relative): 0.0000]

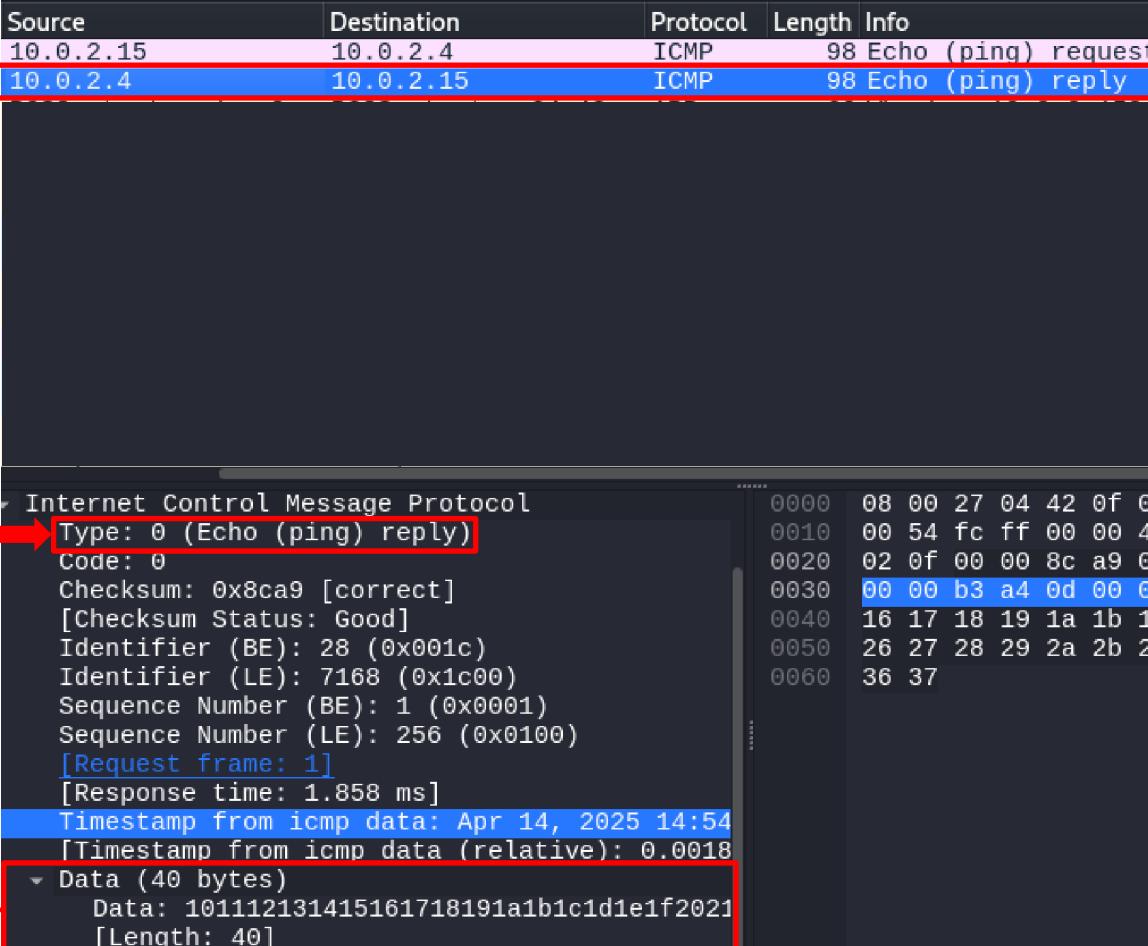
A red arrow points to the "Type: 8 (Echo (ping) request)" line. Another red arrow points to the details pane. The bottom right corner of the window has the Wireshark logo.

# Il comando ping

## Esempio 2 – Ping e Wireshark

Source	Destination	Protocol	Length	Info
10.0.2.15	10.0.2.4	ICMP	98	Echo (ping) request
10.0.2.4	10.0.2.15	ICMP	98	Echo (ping) reply

Internet Control Message Protocol  
Type: 0 (Echo (ping) reply)  
Code: 0  
Checksum: 0x8ca9 [correct]  
[Checksum Status: Good]  
Identifier (BE): 28 (0x001c)  
Identifier (LE): 7168 (0x1c00)  
Sequence Number (BE): 1 (0x0001)  
Sequence Number (LE): 256 (0x0100)  
[Request frame: 1]  
[Response time: 1.858 ms]  
Timestamp from icmp data: Apr 14, 2025 14:54  
[Timestamp from icmp data (relative): 0.0018  
Data (40 bytes)  
Data: 101112131415161718191a1b1c1d1e1f2021  
[Length: 40]



# Il comando ping

## Esempio 2 – Ping e Wireshark

```
ping -c 1 10.0.2.4
```

I 56 byte di dati ICMP sono partizionati come segue:

[ICMP Header (Type, Code, Checksum, ID, Seq): 8 byte]

- Type: 8 (Echo Request)                          -> 1 byte
- Code: 0    -> 1 byte
- Checksum: 0xXXXX                                -> 2 byte
- Identifier (BE): 0xXXXX                        -> 2 byte
- Sequence number (BE): 0xXXXX                   -> 2 byte

[Timestamp: 8 byte, aggiunti dal ping come primi 8 byte del payload]

[Payload: 40 byte]

Source	Destination	Protocol
10.0.2.15	10.0.2.4	ICMP
10.0.2.4	10.0.2.15	ICMP

Internet Control Message Protocol

Type: 0 (Echo (ping) reply)

Code: 0

Checksum: 0x8ca9 [correct]  
[Checksum Status: Good]

Identifier (BE): 28 (0x001c)  
Identifier (LE): 7168 (0x1c00)

Sequence Number (BE): 1 (0x0001)  
Sequence Number (LE): 256 (0x0100)

[Request frame: 1]

[Response time: 1.858 ms]

Timestamp from icmp data: Apr 14, 2025 14:54  
[Timestamp from icmp data (relative): 0.0018]

▼ Data (40 bytes)

Data: 101112131415161718191a1b1c1d1e1f2021  
[Length: 40]

# Il comando ping

## Esempio 3 – Ping e Wireshark

---

➤ Eseguendo il seguente comando

➤ `ping -s 128 -c 1 10.0.2.4`

➤ Possiamo osservare in Wireshark che

I 128 byte di dati ICMP sono partizionati come segue:

[ICMP Header (Type, Code, Checksum, ID, Seq): 8 byte]

- Type: 8 (Echo Request)	-> 1 byte
- Code: 0	-> 1 byte
- Checksum: 0xXXXX	-> 2 byte
- Identifier (BE): 0xXXXX	-> 2 byte
- Sequence number (BE): 0xXXXX	-> 2 byte

[Timestamp: 8 byte, aggiunti dal ping come primi 8 byte del payload]

[Payload: 112 byte]

# Il comando ping6

- Assumiamo che la macchina target abbia il seguente indirizzo IPv6
  - **fe80::78cc:bcf:fae5:5bf8**

```
[user@parrot]~$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
      inet 10.0.2.5 netmask 255.255.255.0 broadcast 10.0.2.255  
        →inet6 fe80::78cc:bcf:fae5:5bf8 prefixlen 64 scopeid 0x20<link>  
          ether 08:00:27:7a:1a:c2 txqueuelen 1000 (Ethernet)  
            RX packets 1012975 bytes 1521223791 (1.4 GiB)  
            RX errors 0 dropped 0 overruns 0 frame 0  
            TX packets 321498 bytes 18803976 (17.9 MiB)  
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Mediante il comando **ifconfig** è possibile conoscere l'indirizzo IPv6 (`inet6`) associato ad una determinata interfaccia di rete

# Il comando ping6

---

- Eseguiamo il seguente comando
  - `ping6 fe80::78cc:bcf:fae5:5bfb`
- Analizzando tramite Wireshark lo scambio di informazioni possiamo notare che

Source	Destination	Protocol	Length	Info
<code>fe80::a00:27ff:fe95:8c5e</code>	<code>fe80::78cc:bcf:fae5:5bfb</code>	ICMPv6	118	Echo (ping)
<code>fe80::78cc:bcf:fae5:5bfb</code>	<code>fe80::a00:27ff:fe95:8c5e</code>	ICMPv6	118	Echo (ping)

- **Osservazione:** Poiché la macchina target è attiva e non filtra i pacchetti *ICMPv6 echo request*, essa invierà un pacchetto *ICMPv6 Echo reply* alla macchina Kali



# Il Comando Osservazioni

---

- **Osservazione 1:** Le richieste inviate tramite il comando **ping** potrebbero essere bloccate
  - Ad esempio configurando il firewall in modo da accettare *ICMP Echo request* solo da determinati indirizzi IP
  
- **Osservazione 2:** È possibile utilizzare il comando **ping** sia verso indirizzi IP pubblici che privati

# Identificare le Macchine Target

## Principali Strumenti

---

- ping e ping6
- **fping**
- nmap
- arping
- arp-scan
- hping3
- nping

# Il comando fping

## Logica di Funzionamento

---

- Il comando **fping** permette di inviare una *ICMP Echo request* a più macchine target (host) contemporaneamente
- È possibile specificare direttamente la lista delle macchine target
  - Mediante terminale
  - Usando un file contenente la lista delle macchine da analizzare

# Il comando fping

## Help

---

- L'help del comando **fping** è ottenibile mediante

**fping -h**

- Schermata dell'Help (*Parziale*)

```
root@kali:~# fping -h
Usage: fping [options] [targets...]

Probing options:
  -4, --ipv4      only ping IPv4 addresses
  -6, --ipv6      only ping IPv6 addresses
  -b, --size=BYTES amount of ping data to send, in bytes (default: 56)
  -B, --backoff=N set exponential backoff factor to N (default: 1.5)
  -c, --count=N   count mode: send N pings to each target
  -f, --file=FILE  read list of targets from a file ( - means stdin)
  -g, --generate   generate target list (only if no -f specified)
                   (give start and end IP in the target list, or a CIDR address)
                   (ex. fping -g 192.168.1.0 192.168.1.255 or fping -g 192.168.1.0/24)
  -H, --ttl=N     set the IP TTL value (Time To Live hops)
  -I, --iface=IFACE bind to a particular interface
  -l, --loop       loop mode: send pings forever
  -m, --all        use all IPs of provided hostnames (e.g. IPv4 and IPv6), use with -A
```

# Il comando fping

## Help

---

- Ulteriori informazioni sul comando **fping** possono essere ottenute mediante la relativa *man page*

**man fping**

- Schermata del man (*Parziale*)

```
fping [ options ] [ systems... ]
```

**DESCRIPTION**

fping is a program like ping which uses the Internet Control  
Message Protocol (ICMP) echo request to determine if a target  
host is responding. fping differs from ping in that you can  
specify any number of targets on the command line, or specify  
file containing the lists of targets to ping. Instead of  
sending to one target until it times out or replies, fping will  
send out a ping packet and move on to the next target in a

Manual page fping(8) line 1 (press h for help or q to quit)

# Il comando fping

## Logica di Funzionamento

---

- Di default il comando **fping** invia tre *ICMP Echo request*
  - Il comportamento di default del comando può essere modificato usando l'opzione **-r**
  
- **fping** monitora le risposte da parte di ciascuna macchina target
  - Se una macchina target invia una risposta (*ICMP Echo reply*), tale macchina sarà etichettata come «attiva» (*alive*)
  - Altrimenti, se una macchina target non risponde entro un certo periodo di timeout, tale macchina sarà etichettata come «non raggiungibile» (*unreachable*)

# Il comando fping

## Esempio Grafico

---



**Lista dei Target:**

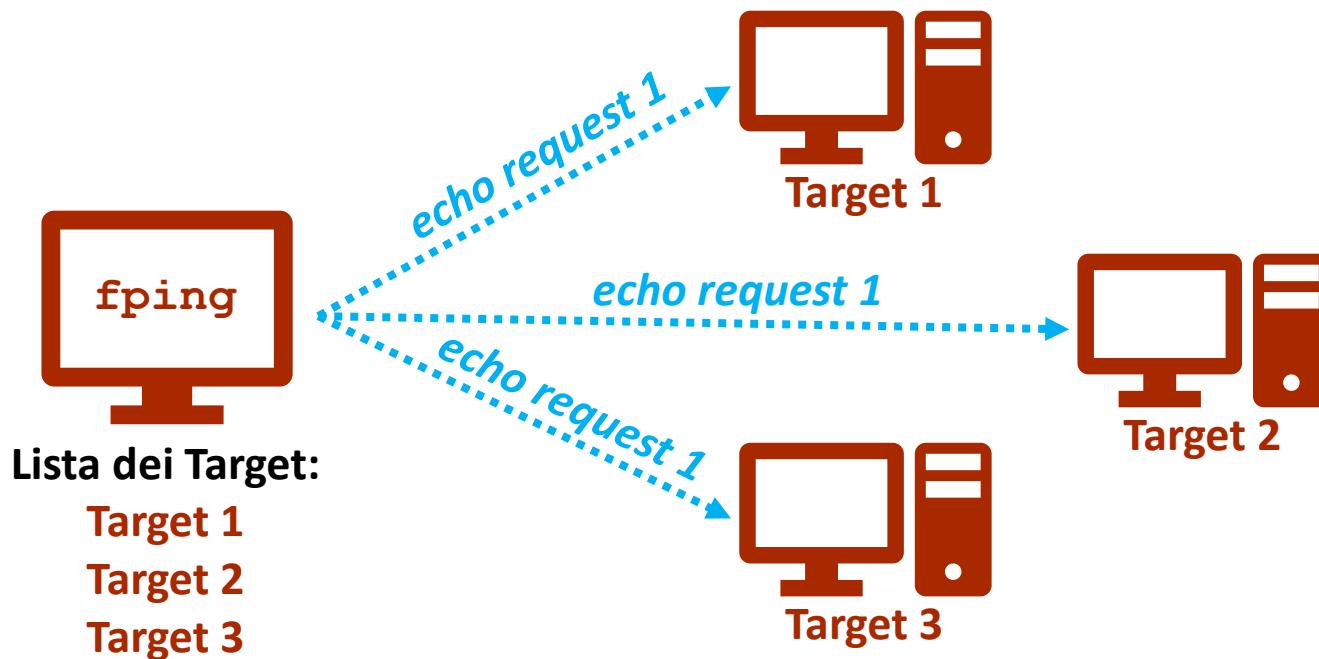
**Target 1**

**Target 2**

**Target 3**

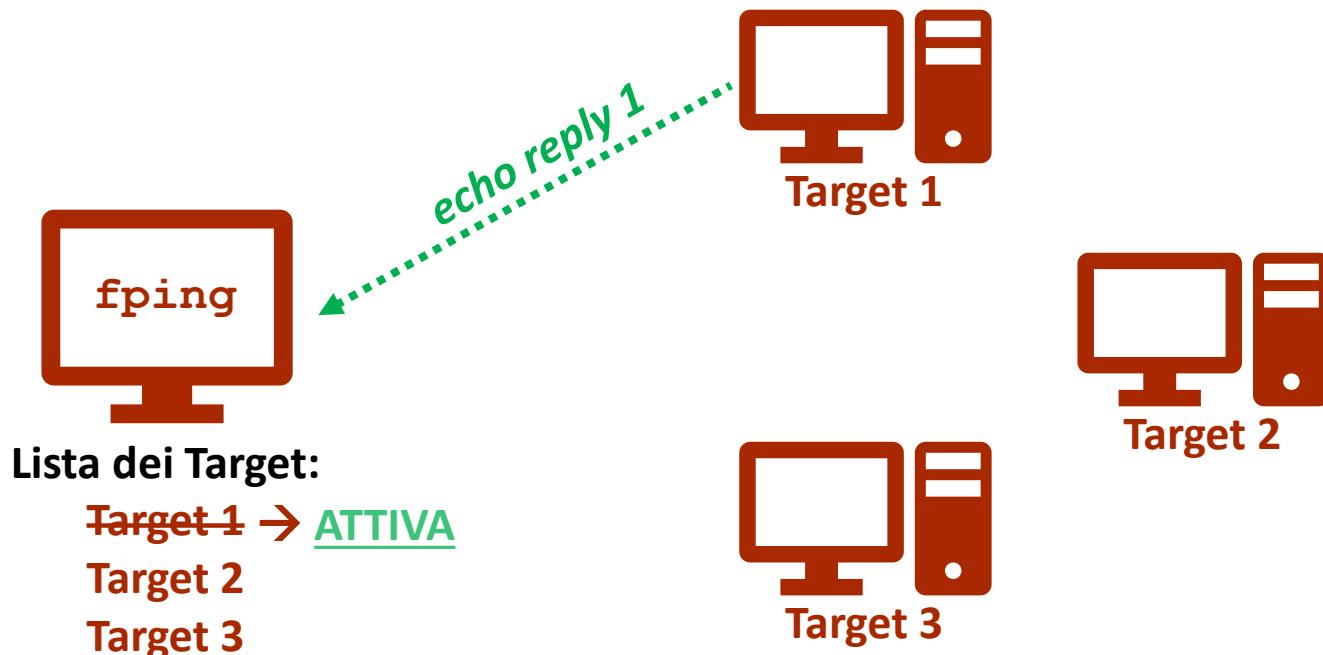
# Il comando fping

## Esempio Grafico



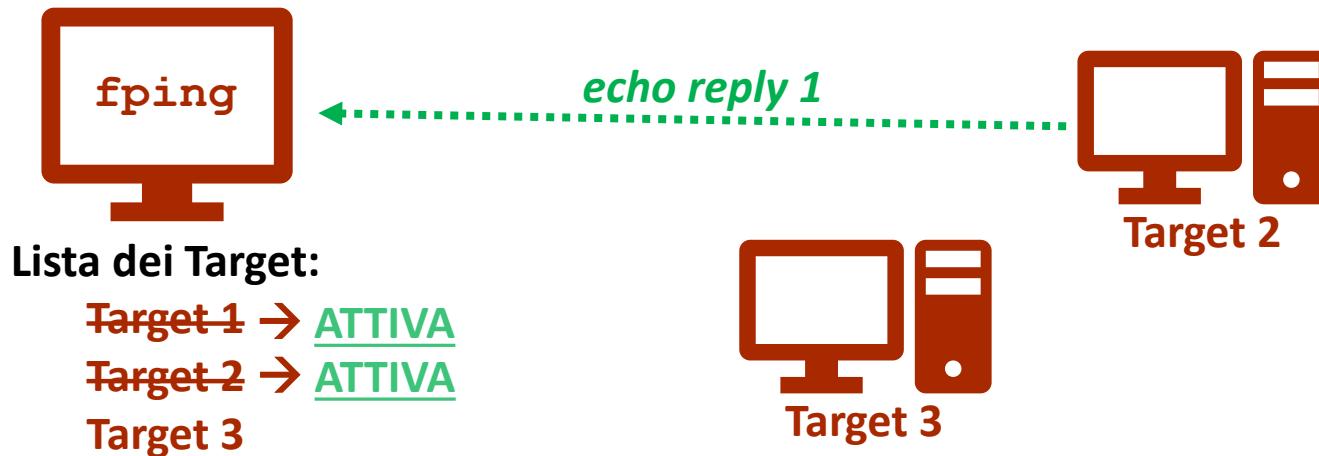
# Il comando fping

## Esempio Grafico



# Il comando fping

## Esempio Grafico



# Il comando fping

## Esempio Grafico



Lista dei Target:

~~Target 1~~ → ATTIVA

~~Target 2~~ → ATTIVA

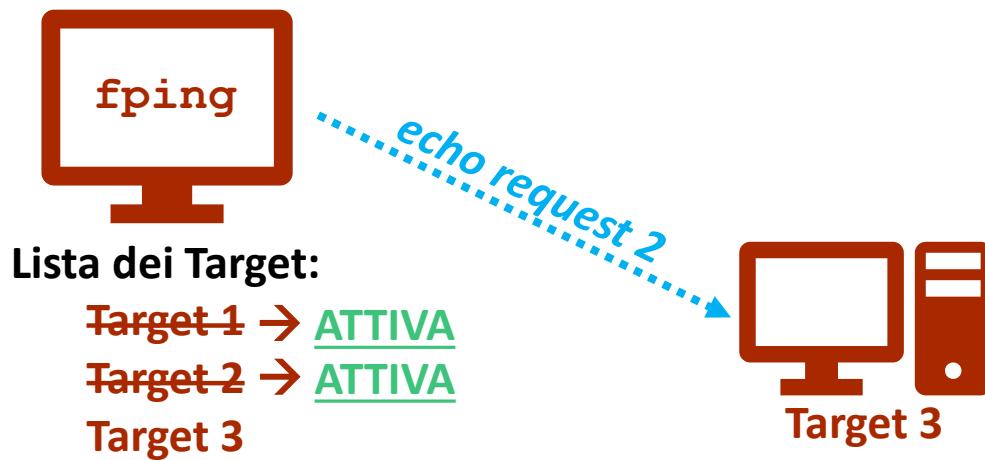
Target 3



Nessuna risposta  
all'echo request 1,  
dopo il timeout

# Il comando fping

## Esempio Grafico



# Il comando fping

## Esempio Grafico

---



**Lista dei Target:**

~~Target 1~~ → ATTIVA

~~Target 2~~ → ATTIVA

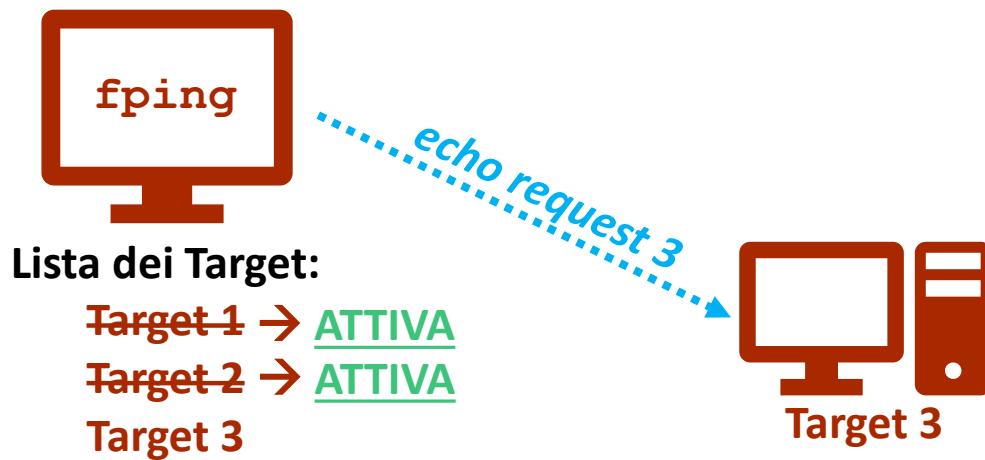
Target 3



Nessuna risposta  
all'echo request 2,  
dopo il timeout

# Il comando fping

## Esempio Grafico



# Il comando fping

## Esempio Grafico



Lista dei Target:

Target 1 → ATTIVA

Target 2 → ATTIVA

Target 3



Nessuna risposta  
all'echo request 3,  
dopo il timeout

# Il comando fping

## Esempio Grafico

---



**Lista dei Target:**

**Target 1 → ATTIVA**

**Target 2 → ATTIVA**

**Target 3 → NON RAGGIUNGIBILE**

# Il comando fping

## Esempio 1

---

- Utilizzo del comando **fping** su tre macchine target

- **64.233.176.94**
  - **8.8.8.8**
  - **83.4.123.45**

- **Esempio**

```
fping 64.233.176.94 8.8.8.8 83.4.123.45
```

# Il comando fping

## Esempio 1

---

- `fping 64.233.176.94 8.8.8.8 83.4.123.45`

```
root@kali:~# fping 64.233.176.94 8.8.8.8 83.4.123.45
8.8.8.8 is alive
64.233.176.94 is alive
83.4.123.45 is unreachable
root@kali:~# █
```

- Le prime due macchine target (**64.233.176.94** e **8.8.8.8**) sono attive (*alive*)
- La terza (**83.4.123.45**) è non raggiungibile (*unreachable*)

# Il comando fping

## Esempio 2

- **fping** consente di valutare una lista di host senza specificarli tutti in modo manuale
- Supponiamo di voler individuare gli host attivi all'interno della rete **10.15.21.0/24**
  - È possibile utilizzare l'opzione **-g** del comando **fping** nel modo seguente
  - **fping -g 10.15.21.0/24**
- *Output (Parziale)*

```
kali@linux# fping -g 10.15.21.0/24
10.15.21.0 is alive
10.15.21.1 is alive
...
...
10.15.21.100 is unreachable
...
10.15.21.120 is unreachable
...
10.15.21.160 is alive
kali@linux#
```

# Il comando fping

## Esempio 3

- Mediante l'opzione **-s** è anche possibile ottenere alcune statistiche
  - **fping -s 64.233.176.94 8.8.8.8 83.4.123.45**

```
root@kali:~# fping -s 64.233.176.94 8.8.8.8 83.4.123.45
8.8.8.8 is alive
64.233.176.94 is alive
83.4.123.45 is unreachable

  3 targets
  2 alive
  1 unreachable
  0 unknown addresses

  1 timeouts (waiting for response)
  6 ICMP Echos sent
  2 ICMP Echo Replies received
  0 other ICMP received

  22.8 ms (min round trip time)
  83.3 ms (avg round trip time)
  143 ms (max round trip time)
  4.103 sec (elapsed real time)
```

Statistiche ottenute  
usando l'opzione **-s**

# Identificare le Macchine Target

## Principali Strumenti

---

- ping e ping6
- fping
- **nmap**
- arping
- arp-scan
- hping3
- nping

# Il comando nmap

---

- Strumento Open Source multifunzione per l'analisi di rete
- Può essere usato per molteplici finalità
  - Ad esempio, per verificare se una macchina target è attiva
    - Utilizzando l'opzione **-sP** (*Ping Scan*)
- Esempio 1 (Singola Macchina)
  - **nmap -sP 10.0.2.13**

```
root@kali:~# nmap -sP 10.0.2.13
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-23 23:26 CET
Nmap scan report for 10.0.2.13
Host is up (0.00047s latency).
MAC Address: 08:00:27:63:2C:EB (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 5.69 seconds
```

# Il comando nmap

---

- Si può usare **nmap** per determinare quali host sono attivi in un determinato intervallo di rete
- **Esempio 2 (Sottorete)**
  - `nmap -sP 10.0.2.0/24`

```
root@kali:~# nmap -sP 10.0.2.1/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-23 23:29 CET
Nmap scan report for 10.0.2.1
Host is up (0.00043s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00037s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00035s latency).
MAC Address: 08:00:27:77:FE:A0 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.00034s latency).
MAC Address: 08:00:27:B2:42:60 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.6
Host is up (0.00069s latency).
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
```

# Il comando nmap

- Si può usare **nmap** per determinare quali host sono attivi in un determinato intervallo di rete
- **Esempio 2 (Sottorete)**
  - **nmap -sP 10.0.2.0/24**

```
root@kali:~# nmap -sP 10.0.2.1/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-23 23:29 CET
Nmap scan report for 10.0.2.1
Host is up (0.00043s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00037s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00035s latency).
MAC Address: 08:00:27:77:FE:A0 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.00034s latency).
MAC Address: 52:42:60 (Oracle VirtualBox virtual NIC)
.0.2.6
tency).
E:29:E1 (Oracle VirtualBox virtual NIC)
```

Per maggiori informazioni

<https://technology.amis.nl/2018/07/27/virtualbox-networking-explained/>

Indirizzi IP sempre presenti,  
poiché utilizzati dall'architettura  
di virtualizzazione di Virtual Box

# Il comando nmap

- Si può usare **nmap** per determinare quali host sono attivi in un determinato intervallo di rete
- **Esempio 2 (Sottorete)**
  - `nmap -sP 10.0.2.0/24`

```
root@kali:~# nmap -sP 10.0.2.1/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-23 23:29 CET
Nmap scan report for 10.0.2.1
Host is up (0.00043s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00037s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00035s latency).
MAC Address: 08:00:27:77:FE:A0 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.00034s latency).
MAC Address: 08:00:27:B2:42:60 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.6
Host is up (0.00069s latency).
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
```

Macchine reali  
presenti nella rete

# Identificare le Macchine Target

## Principali Comandi

---

- ping e ping6
- fping
- nmap
- **arping**
- arp-scan
- hping3
- nping

# Perché usare ARP?

---

- ARP (Address Resolution Protocol)
  
- Funziona
  - Solo all'interno di reti locali (LAN)
  - Anche se il firewall blocca ICMP o altre tipologie di scansione basate su protocolli di livello superiore rispetto ad ARP
  - Anche con «host silenziosi»: Anche se un host non genera traffico visibile o ha servizi disattivati, se ha una scheda di rete attiva, risponderà ad ARP
  
- È molto veloce

# Il comando arping

## Logica di Funzionamento

---

- **arping** è usato per verificare la presenza di un host in una rete **LAN (Local Area Network)**
  - Basato sul protocollo **ARP (Address Resolution Protocol)**
- È possibile usare **arping** verso la macchina target specificando il suo
  - Indirizzo IP
  - Hostname
  - Indirizzo **MAC (Media Access Control)**

# Il comando arping

## Logica di Funzionamento

---

- **arping** può essere usato solo nel contesto di una rete locale (LAN)
- **arping** usa il protocollo ARP, che non può essere instradato tramite router o gateway

**ARP**  
*Address  
Resolution  
Protocol*

# Il comando arping

## Logica di Funzionamento

- Invocando il comando **arping** senza argomenti è possibile visualizzare una schermata sintetica dell'help
  - In cui sono indicate tutte le sue possibili opzioni

```
root@kali:~# arping
ARPing 2.19, by Thomas Habets <thomas@habets.se>
usage: arping [ -0aAbdDeFpPqrRuUv ] [ -w <sec> ] [ -W <sec> ] [ -S <host/ip> ]
          [ -T <host/ip> ] [ -s <MAC> ] [ -t <MAC> ] [ -c <count> ]
          [ -C <count> ] [ -i <interface> ] [ -m <type> ] [ -g <group> ]
          [ -V <vlan> ] [ -Q <priority> ] <host/ip/MAC | -B>
For complete usage info, use --help or check the manpage.
```

**N.B.** Per poter funzionare richiede i privilegi di root

# Il comando arping

## Logica di Funzionamento

- Per ottenere l'help completo è possibile utilizzare il seguente comando

**man arping**

- Schermata dell'Help (*Parziale*)

```
arping(8)                                     arping(8)

NAME
    arping - sends arp and/or ip pings to a given host

SYNOPSIS
    arping [-0aAbBdDeFhpqrRuUv] [-S host/ip] [-T host/ip] [-s MAC]      [-t MAC]
    [-c count] [-i interface] [ -w seconds ] [ -W seconds ] [ -V vlan ] [ -Q
    priority ] [ -g group ] <host | -B>

    arping --help

DESCRIPTION
    The arping utility sends ARP and/or ICMP requests to the specified host
    and displays the replies. The host may be specified by its hostname, its
```

# Il comando arping

## Esempio 1

---

- È possibile utilizzare **arping** per ottenere l'indirizzo MAC della macchina target

- **arping 10.0.2.5 -c 1**

```
root@kali:~# arping 10.0.2.5 -c 1
ARPING 10.0.2.5
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=0 time=9.185 msec

--- 10.0.2.5 statistics ---
1 packets transmitted, 1 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 9.185/9.185/9.185/0.000 ms
root@kali:~#
```

# Il comando arping

## Esempio 1

- È possibile utilizzare **arping** per ottenere l'indirizzo MAC della macchina target

➤ **arping 10.0.2.5 -c 1**

```
root@kali:~# arping 10.0.2.5 -c 1
ARPING 10.0.2.5
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=0 time=9.185 msec

--- 10.0.2.5 statistics ---
1 packets transmitted, 1 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 9.185/9.185/9.185/0.000 ms
root@kali:~#
```

- Possiamo osservare che la macchina target ha indirizzo  
**MAC 08:00:27:7a:1a:c2**

# Il comando arping

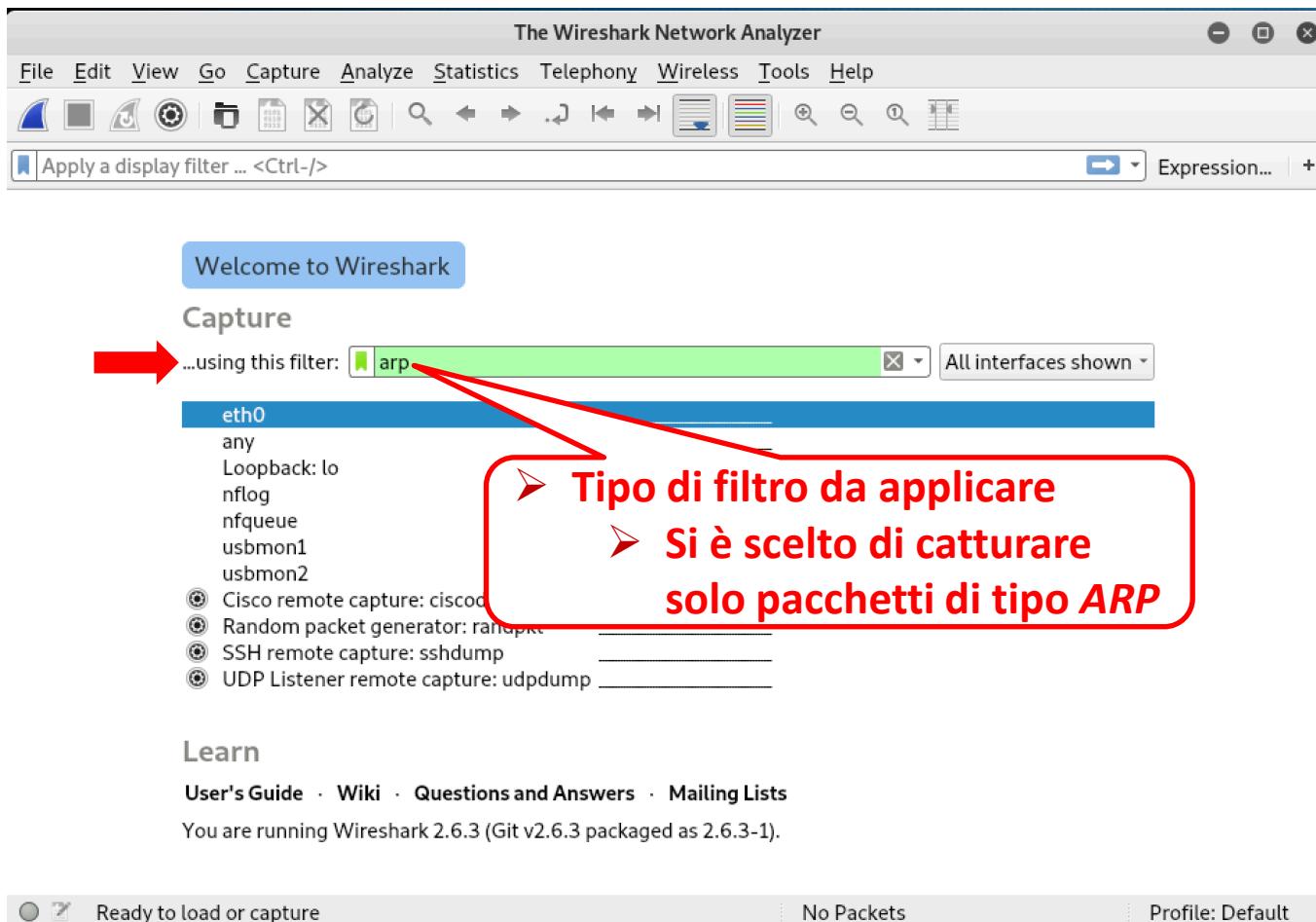
## Esempio 2

---

- È possibile utilizzare **arping** per ottenere l'indirizzo *MAC* della macchina target
  - `arping 10.0.2.5 -c 1`
  
- Ripetiamo l'Esempio 1 utilizzando Wireshark

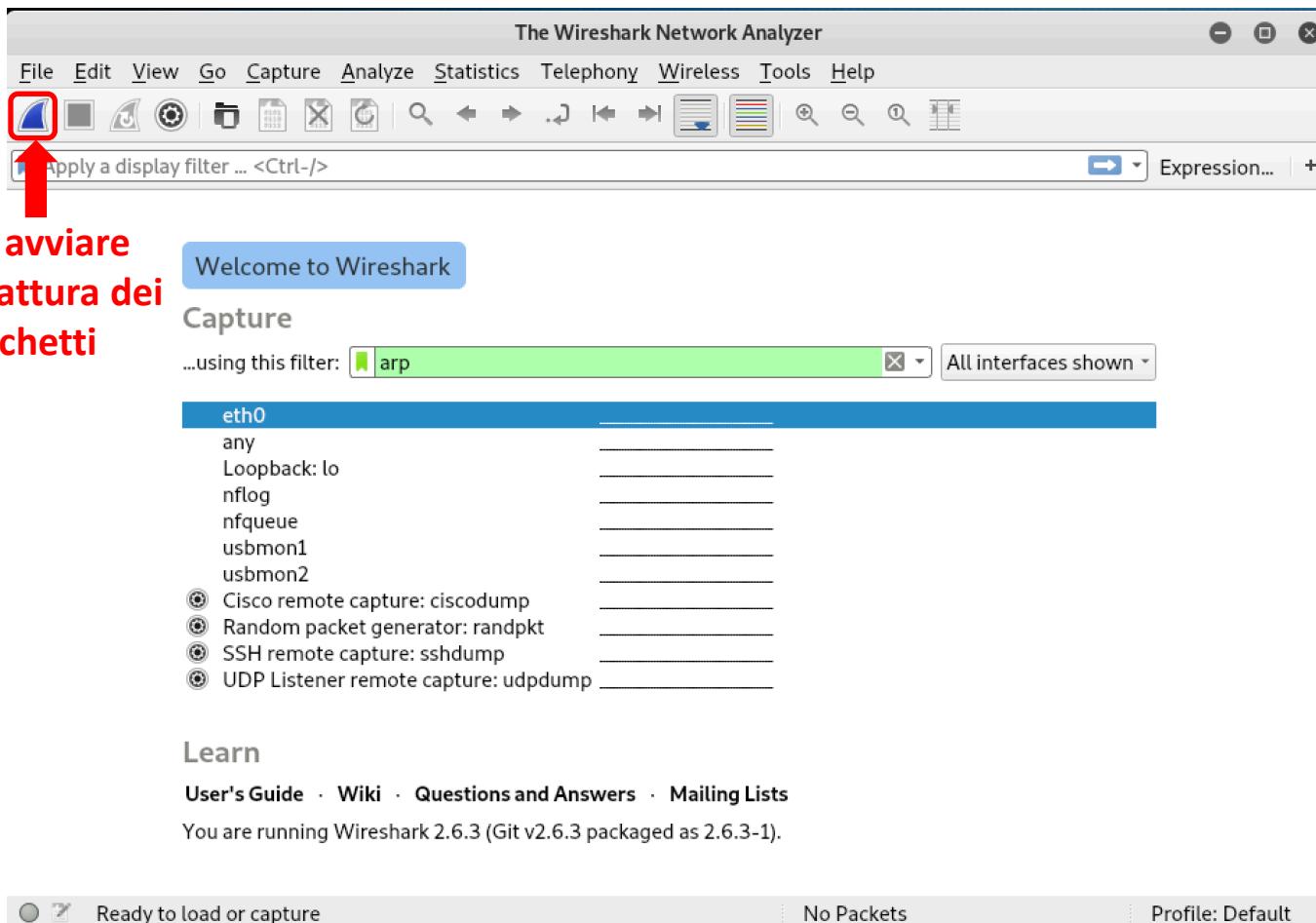
# Il comando arping

## Esempio 2



# Il comando arping

## Esempio 2



# Il comando arping

## Esempio 2

---

- Invochiamo nuovamente il comando **arping** con la stessa sintassi usata in precedenza

- **arping 10.0.2.5 -c 1**

```
root@kali:~# arping 10.0.2.5 -c 1
ARPING 10.0.2.5
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=0 time=9.185 msec

--- 10.0.2.5 statistics ---
1 packets transmitted, 1 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 9.185/9.185/9.185/0.000 ms
root@kali:~#
```

# Il comando arping

## Esempio 2 – Richiesta ARP

- Osserviamo i pacchetti catturati da Wireshark durante l'esecuzione del comando **arping**

```
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
  ▶ Interface id: 0 (eth0)
  Encapsulation type: Ethernet (1)
  Arrival Time: Mar 22, 2019 18:33:49.801170494 CET
    [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1553276029.801170494 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
  Frame Number: 1
  Frame Length: 58 bytes (464 bits)
  Capture Length: 58 bytes (464 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:arp]
    [Coloring Rule Name: ARP]
    [Coloring Rule String: arp]
```

# Il comando arping

## Esempio 2 – Richiesta ARP

- Osserviamo i pacchetti catturati da Wireshark durante l'esecuzione del comando **arping**

```
▼ Ethernet II, Src: PcsCompu_95:8c:5e (08:00:27:95:8c:5e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
    Type: ARP (0x0806)
    Trailer: 00000000000000000000000000000000
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
  Sender IP address: 10.0.2.15
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.0.2.5
```

# Il comando arping

## Esempio 2 – Richiesta ARP

- Osserviamo i pacchetti catturati da Wireshark durante l'esecuzione del comando **arping**

```
▼ Ethernet II, Src: PcsCompu_95:8c:5e (08:00:27:95:8c:5e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
    Type: ARP (0x0806)
    Trailer: 00000000000000000000000000000000
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
  Sender IP address: 10.0.2.15
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.0.2.5
```

Indirizzo MAC della  
macchina Kali  
08:00:27:95:8c:5e

# Il comando arping

## Esempio 2 – Richiesta ARP

- Osserviamo i pacchetti catturati da Wireshark durante l'esecuzione del comando **arping**

```
▼ Ethernet II, Src: PcsCompu_95:8c:5e (08:00:27:95:8c:5e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
    Type: ARP (0x0806)
    Trailer: 00000000000000000000000000000000
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: PcsCompu_95:8c:5e (08:00:27:95:8c:5e)
  Sender IP address: 10.0.2.15
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.0.2.5
```

La macchina Kali invia una richiesta ARP ad un indirizzo MAC di broadcast specificando l'indirizzo IP 10.0.2.5

# Il comando arping

## Esempio 2 – Risposta ARP

- Osserviamo i pacchetti catturati da Wireshark durante l'esecuzione del comando **arping**

```
Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
  ▶ Interface id: 0 (eth0)
    Encapsulation type: Ethernet (1)
    Arrival Time: Mar 22, 2019 18:33:49.802439707 CET
      [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1553276029.802439707 seconds
      [Time delta from previous captured frame: 0.001269213 seconds]
      [Time delta from previous displayed frame: 0.001269213 seconds]
      [Time since reference or first frame: 0.001269213 seconds]
    Frame Number: 2
    Frame Length: 60 bytes (480 bits)
    Capture Length: 60 bytes (480 bits)
      [Frame is marked: False]
      [Frame is ignored: False]
      [Protocols in frame: eth:ethertype:arp]
      [Coloring Rule Name: ARP]
      [Coloring Rule String: arp]
```

# Il comando arping

## Esempio 2 – Risposta ARP

- Osserviamo i pacchetti catturati da Wireshark durante l'esecuzione del comando **arping**

- Ethernet II, Src: PcsCompu\_7a:1a:c2 (08:00:27:7a:1a:c2), Dst: PcsCompu\_95:8c:5e (08:00:27:95:8c:5e)
  - Destination: PcsCompu\_95:8c:5e (08:00:27:95:8c:5e)
  - Source: PcsCompu\_7a:1a:c2 (08:00:27:7a:1a:c2)
  - Type: ARP (0x0806)
  - Padding: 00
- Address Resolution Protocol (reply)
  - Hardware type: Ethernet (1)
  - Protocol type: IPv4 (0x0800)
  - Hardware size: 6
  - Protocol size: 4
  - Opcode: reply (2)
  - Sender MAC address: PcsCompu\_7a:1a:c2 (08:00:27:7a:1a:c2)
  - Sender IP address: 10.0.2.5
  - Target MAC address: PcsCompu\_95:8c:5e (08:00:27:95:8c:5e)
  - Target IP address: 10.0.2.15

Se l'indirizzo IP esiste (come nel nostro caso), l'host relativo a tale indirizzo IP invierà una risposta ARP contenente il proprio indirizzo MAC

Se l'indirizzo IP esiste (come nel nostro caso), l'host relativo a tale indirizzo IP invierà una risposta *ARP* contenente il proprio indirizzo *MAC*

# Il comando arping

## Esempio 3 – Rilevazione Indirizzi IP Duplicati

- **arping** permette di rilevare indirizzi IP duplicati in una rete locale
- Utilizziamo **arping** per rilevare se è stato utilizzato più di una volta l'indirizzo IP 10.0.2.5
  - **arping -d -i eth0 10.0.2.5 -c 2**
  - **echo \$?**

```
root@kali:~# arping -d -i eth0 10.0.2.5 -c 2
ARPING 10.0.2.5
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=0 time=633.605 usec
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5): index=1 time=7.050 msec
2.3-
-2- 10.0.2.5 statistics ---
2 packets transmitted, 2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.634/3.842/7.050/3.208 ms
root@kali:~# echo $?
0
root@kali:~#
```

# Il comando arping

## Esempio 3 – Rilevazione Indirizzi IP Duplicati

- **arping** permette di rilevare indirizzi IP duplicati in una rete locale
- Utilizziamo **arping** per rilevare se è stato utilizzato più di una volta l'indirizzo IP 10.0.2.5
  - **arping -d -i eth0 10.0.2.5 -c 2**
  - **echo \$?**

```
root@kali:~# arping -d -i eth0 10.0.2.5
ARPING 10.0.2.5
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5)
60 bytes from 08:00:27:7a:1a:c2 (10.0.2.5)
2.3ms
-- 10.0.2.5 statistics --
2 packets transmitted, 2 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.034/3.842/7.050/3.208 ms
root@kali:~# echo $?
0
root@kali:~#
```

\$? rappresenta il codice restituito dall'ultimo processo eseguito

- 0 indica che non si è verificato alcun errore
- Altri valori (diversi da 0) rappresentano possibili situazioni anomale o di errore

# Il comando arping

## Esempio 3 – Rilevazione Indirizzi IP Duplicati

- **arping** permette di rilevare indirizzi IP duplicati in una rete locale
  - Utilizziamo **arping** per rilevare se è stato utilizzato più di una volta l'indirizzo IP 10.0.2.5
    - **arping -d -i eth0 10.0.2.5 -c 2**
    - **echo \$?**

```
root@kali:~# arping -d
ARPING 10.0.2.5
60 bytes from 08:00:27
60 bytes from 08:00:27
2.3-
--- 10.0.2.5 statistics ---
2 packets transmitted, 2 packets received, 0% unanswered (0 extra)
rtt min/avg/max = 0.634/3.842/7.050/3.208 ms
root@kali:~# echo $?
0
root@kali:~#
```

➤ 0 indica che il processo **arping** è terminato con successo  
➤ Non si è verificato alcun errore

# Il comando arping

## Esempio 3 – Rilevazione Indirizzi IP Duplicati

---

- **arping** permette di rilevare indirizzi IP duplicati in una rete locale
- Avviamo due VM, ad esempio
  - VM1 = Metasploitable 1 (Avente indirizzo IP **10.0.2.12**)
  - VM2 = Metasploitable 2 (Avente indirizzo IP **10.0.2.6**)
- Assegnamo alla VM1 lo stesso indirizzo IP della VM2
  - **ifconfig eth0 10.0.2.6**
- Utilizziamo **arping** per rilevare se è stato utilizzato più di una volta l'indirizzo IP **10.0.2.6**
  - **arping -d -i eth0 10.0.2.6 -c 2**
  - **echo \$?**

# Il comando arping

## Esempio 3 – Rilevazione Indirizzi IP Duplicati

```
➤ arping -d -i eth0 10.0.2.6 -c 2  
➤ echo $?
```

```
root@kali:~# arping -d -i eth0 10.0.2.6 -c 2  
ARPING 10.0.2.6  
60 bytes from 08:00:27:ae:29:e1 (10.0.2.6): index=0 time=453.061 usec  
60 bytes from 08:00:27:d3:79:ab (10.0.2.6): index=1 time=528.885 usec  
60 bytes from 08:00:27:d3:79:ab (10.0.2.6): index=2 time=391.331 usec  
60 bytes from 08:00:27:ae:29:e1 (10.0.2.6): index=3 time=429.999 usec  
- pippo.txt jkakavas-  
:zip 10.0.2.6-statistics creepy-plugins...  
2 packets transmitted, 4 packets received, 0% unanswered (2 extra)  
rtt min/avg/max/std-dev = 0.391/0.451/0.529/0.050 ms  
root@kali:~# echo $?  
1
```

# Il comando arping

## Esempio 3 – Rilevazione Indirizzi IP Duplicati

```
➤ arping -d -i eth0 10.0.2.6 -c 2  
➤ echo $?
```

```
root@kali:~# arping -d -i eth0 10.0.2.6 -c 2  
ARPING 10.0.2.6  
60 bytes from 08:00:27:ae:29:e1 (10.0.2.6): index=0 time=453.061 usec  
60 bytes from 08:00:27:d3:79:ab (10.0.2.6): index=1 time=528.885 usec  
60 bytes fr ➤ 1 indica che il processo arping non  
60 bytes fr ➤ è terminato con successo  
- p ➤ Si è verificato un errore  
zip 10.0.2.6 2 packets transmitted, 1 packets received, 0% unanswered (2 extra)  
rtt min/avg/max/std-dev = 0.391/0.451/0.529/0.050 ms  
root@kali:~# echo $?  
1
```

# Identificare le Macchine Target

## Principali Strumenti

---

- ping e ping6
- fping
- nmap
- arping
- **arp-scan**
- hping3
- nping

# Il comando arp-scan

---

- Permette di conoscere gli host attivi sulla rete locale
- Utilizza il protocollo *ARP*
- Per poterlo eseguire è necessario disporre dei permessi di root
- Per maggiori informazioni sul comando **arp-scan**
  - **man arp-scan**

```
ARP-SCAN(1)          General Commands Manual          ARP-SCAN(1)

NAME
    arp-scan - The ARP scanner

SYNOPSIS
    arp-scan [options] [hosts...]
                                         [creepy-plugins...]

Target hosts must be specified on the command line unless the
--file option is given, in which case the targets are read from
the specified file instead, or the --localnet option is used,
in which case the targets are generated from the network inter-
face IP address and netmask.
```

# Il comando arp-scan

## Esempio 1

➤ **arp-scan 10.0.2.0/24**

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:77:fe:a0      Cadmus Computer Systems
10.0.2.4      08:00:27:b2:42:60      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems
10.0.2.9      08:00:27:97:0f:ef      Cadmus Computer Systems
10.0.2.12     08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.13     08:00:27:63:2c:eb      Cadmus Computer Systems
10.0.2.14     08:00:27:f1:65:3c      Cadmus Computer Systems

9 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.367 seconds (108.15 hosts/sec).
9 responded
```

# Il comando arp-scan

## Esempio 1

➤ **arp-scan 10.0.2.0/24**

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:77:fe:a0      Cadmus Computer Systems
10.0.2.4      08:00:27:b2:42:60      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems
10.0.2.9      08:00:27:97:0f:ef      Cadmus Computer Systems
10.0.2.12     08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.13     08:00:27:63:2c:eb      Cadmus Computer Systems
10.0.2.14     08:00:27:f1:65:3c      Cadmus Computer Systems

9 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.367 seconds (108.15 hosts/sec).
9 responded
```

# Il comando arp-scan

## Esempio 1

➤ **arp-scan 10.0.2.0/24**

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:77:fe:a0      Cadmus Computer Systems
10.0.2.4      08:00:27:b2:42:60      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems
10.0.2.9      08:00:27:97:0f:ef      Cadmus Computer Systems
10.0.2.12     08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.13     08:00:27:63:2c:eb      Cadmus Computer Systems
10.0.2.14     08:00:27:f1:65:3c      Cadmus Computer Systems

9 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.367 seconds (108.15 hosts/sec).
9 responded
```

# Il comando arp-scan

## Esempio 1

➤ **arp-scan 10.0.2.0/24**

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:77:fe:a0      Cadmus Computer Systems
10.0.2.4      08:00:27:b2:42:60      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems
10.0.2.9      08:00:27:97:0f:ef      Computer Systems
10.0.2.12     08:00:27:d3:79:ab      Computer Systems
10.0.2.13     08:00:27:63:2c:eb      Computer Systems
10.0.2.14     08:00:27:f1:65:3c      Computer Systems
9 packets received by filter, 0 packets discarded
Ending arp-scan 1.9.5: 256 hosts scanned in 0:00:00
  9 responded
```

Indirizzi IP sempre presenti, che fanno  
parte dell'architettura di virtualizzazione  
utilizzata da Virtual Box

# Il comando arp-scan

Esempio 2 (Presenza di Indirizzi IP Duplicati)

➤ **arp-scan 10.0.2.0/24**

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-
scan)
          creepy-plugins...
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:04:c0:93      Cadmus Computer Systems
10.0.2.6      08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems (DUP: 2)
10.0.2.5      08:00:27:7a:1a:c2      Cadmus Computer Systems

6 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.289 seconds (111.84 hosts/
sec). 6 responded
```

# Il comando arp-scan

Esempio 2 (Presenza di Indirizzi IP Duplicati)

➤ **arp-scan 10.0.2.0/24**

```
root@kali:~# arp-scan 10.0.2.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-
scan)
          creepy-plugins...
10.0.2.1      52:54:00:12:35:00      QEMU
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.3      08:00:27:04:c0:93      Cadmus Computer Systems
10.0.2.6      08:00:27:d3:79:ab      Cadmus Computer Systems
10.0.2.6      08:00:27:ae:29:e1      Cadmus Computer Systems (DUP: 2)
10.0.2.5      08:00:27:7a:1a:c2      Cadmus Computer Systems

6 packets received by filter, 0 packets dropped
Ending arp-scan 1.9.5: 256 hosts scanned in 1.00 sec). 6 responded
```

Indirizzi IP duplicati

# Identificare le Macchine Target

## Principali Strumenti

---

- ping e ping6
- fping
- nmap
- arping
- arp-scan
- **hping3**
- nping

# Il comando hping3

## Logica di Funzionamento

---

- **hping3** è uno strumento multifunzione che consente di generare ed analizzare arbitrari pacchetti di rete
  
- **hping3** può essere utilizzato per
  - Rilevare gli host attivi, tramite varie tecniche
  - Interagire con il protocollo *TCP/IP*
  - Valutare firewall e IDS
  - Valutare le prestazioni di una rete (RTT, perdita di pacchetti, throughput UDP, jitter, etc)
  - Portscanning
  - Simulare attacchi
  - E molto altro ancora...

<http://wiki.hping.org/25>

# Il comando hping3

## Logica di Funzionamento

---

- Per utilizzare il comando **hping3** è sufficiente digitarlo da terminale
- È possibile fornire istruzioni al comando **hping3** in diversi modi
  - Tramite linea di comando
  - Tramite una shell interattiva fornita da **hping3** stesso
  - Tramite script
- Per poter funzionare richiede i privilegi di root

# Il comando hping3

## Logica di Funzionamento

- Di default **hping3** invia pacchetti *TCP* vuoti (*NULL packet*) sulla porta 0
- Se **hping3** viene usato utilizzando *TCP*, è possibile usare tale comando senza alcun flag (comportamento di default) oppure con uno o più dei seguenti flag

Opzione	Nome del Flag
-S	<b>syn</b>
-A	<b>ack</b>
-R	<b>rst</b>
-F	<b>fin</b>
-P	<b>psh</b>
-U	<b>urg</b>
-X	<b>xmas</b> : setta i flag <b>fin</b> , <b>urg</b> e <b>psh</b>
-Y	<b>ymas</b>

# Il comando hping3

## Logica di Funzionamento

- Per modificare il comportamento di default di **hping3** è possibile utilizzare le seguenti opzioni

Opzione (formato breve)	Opzione (formato lungo)	Descrizione
-0	--raw-ip	<b>Invio di pacchetti raw IP</b>
-1	--icmp	<b>Invio di pacchetti ICMP</b>
-2	--udp	<b>Invio di pacchetti UDP</b>
-8	--scan	<b>Modalità «scan»</b>
-9	--listen	<b>Modalità di ascolto («listen»)</b>

# Il comando hping3

## Esempio 1

---

- Invio di un pacchetto *Echo request* tramite il protocollo *ICMP*

```
hping3 -1 64.233.176.94 -c 1
```

- *Output*

```
root@kali:~# hping3 -1 64.233.176.94 -c 1
HPING 64.233.176.94 (eth0 64.233.176.94): icmp mode set, 28 headers + 0 data bytes
len=46 ip=64.233.176.94 ttl=41 id=16172 icmp_seq=0 rtt=150.3 ms

--- 64.233.176.94 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 150.3/150.3/150.3 ms
root@kali:~# █
```

- La macchina target è attiva poiché ha risposto alla richiesta *ICMP* (**1 packets received**)

# Il comando hping3

## Esempio 2 – Valutazione Firewall

---

- È possibile utilizzare **hping3** per analizzare le regole di un firewall presente su una macchina target (ad es., Metasploitable 2)
  
- Supponiamo che sulla macchina target sia in esecuzione un firewall con le seguenti regole (*firewall policy*)
  - **Accetta** tutti i pacchetti *TCP* diretti alla porta 22 (Servizio **SSH**)
  - **Accetta** tutti i pacchetti *TCP* relativi ad una connessione stabilita
  - **Scarta** tutti gli altri pacchetti

# Il comando hping3

## Esempio 2 – Valutazione Firewall

---

- Indirizzo IP della macchina target: 10.0.2.5
- Mediante le seguenti istruzioni configuriamo il **firewall (iptables)** sulla **macchina target** affinché esso
  - Cancelli eventuali **politiche di filtro** definite precedentemente
    - `iptables -F`
    - `iptables -t nat -F`
    - `iptables -X`
  - Accetti tutti i **pacchetti** relativi a connessioni sulla **porta TCP 22** e scarti tutti gli altri
    - `iptables -P FORWARD DROP`
    - `iptables -P INPUT DROP`
    - `iptables -P OUTPUT ACCEPT`
    - `iptables -A INPUT -p tcp --dport 22 -j ACCEPT`

# Il comando hping3

## Esempio 2 – Valutazione Firewall

---

- Indirizzo IP della macchina target: **10.0.2.5**
- Mediante le seguenti istruzioni configuriamo il **firewall (iptables)** sulla **macchina target** affinché esso
  - Cancelli eventuali **politiche di filtro** definite precedentemente
    - **iptables -F**
    - **iptables -t nat -F**
    - **iptables -X**
  - **Accetti** tutti i **pacchetti** relativi a connessioni sulla **porta TCP 22** e **scarti** tutti gli altri
    - **iptables -P FORWARD DROP**
    - **iptables -P INPUT DROP**
    - **iptables -P OUTPUT ACCEPT**
    - **iptables -A INPUT -p tcp --dport 22 -j ACCEPT**

Per maggiori informazioni sul comando **iptables**, digitare **man iptables**

# Il comando hping3

## Esempio 2 – Valutazione Firewall

- I comandi **iptables** possono essere inseriti in uno script
  - Ad esempio, **iptables.sh**

```
iptables -F
iptables -t nat -F
iptables -X
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Contenuto dello script **iptables.sh**

- È necessario impostare i permessi di esecuzione sullo script prima di eseguirlo (**chmod 755 iptables.sh**)

Eseguiamo lo script: **./iptables.sh**

# Il comando hping3

## Esempio 2 – Valutazione Firewall

- Dalla macchina Kali inviamo un pacchetto di *ICMP Echo request* alla macchina target

- `hping3 -1 10.0.2.5 -c 1`

```
root@kali:~# hping3 -1 10.0.2.5 -c 1
HPING 10.0.2.5 (eth0 10.0.2.5): icmp mode set, 28 headers + 0 data bytes

--- 10.0.2.5 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali:~#
```

- La macchina target non risponde all'*ICMP Echo request* (**0 packets received**)
  - Tale richiesta è stata bloccata («*dropped*») dal firewall `iptables`

# Il comando hping3

## Esempio 2 – Valutazione Firewall

- Inviamo invece un pacchetto *TCP* di tipo *SYN* sulla porta 22

```
hping3 10.0.2.5 -c 1 -S -p 22
```

```
root@kali:~# hping3 10.0.2.5 -c 1 -S -p 22
HPING 10.0.2.5 (eth0 10.0.2.5): S set, 40 headers + 0 data bytes
len=46 ip=10.0.2.5 ttl=64 DF id=0 sport=22 flags=RA seq=0 win=0 rtt=7.3
ms
...
--- 10.0.2.5 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 7.3/7.3/7.3 ms
```

- Il firewall della macchina target consente al pacchetto *TCP SYN* di raggiungere la porta 22 (**1 packets received**)

# Il comando hping3

## Esempio 3 – Valutazione Firewall

- Inviamo un pacchetto *UDP* sulla porta 22

```
hping3 -2 10.0.2.5 -c 1 -p 22
```

```
root@kali:~# hping3 -2 10.0.2.5 -c 1 -p 22
HPING 10.0.2.5 (eth0 10.0.2.5): udp mode set, 28 headers + 0 data bytes
--- 10.0.2.5 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

- Possiamo osservare che il firewall del target non accetta pacchetti *UDP* sulla porta 22 (0 **packets received**)

# Identificare le Macchine Target

## Principali Strumenti

---

- ping e ping6
- fping
- nmap
- arping
- arp-scan
- hping3
- **nping**

# Il comando nping

## Logica di Funzionamento

---

- **nping** è uno strumento che permette di
  - Generare pacchetti appartenenti ai protocolli
    - *TCP*
    - *UDP*
    - *ICMP*
    - *ARP*
  - Modificare i campi dell'header dei pacchetti appartenenti a tali protocolli
    - Ad esempio, porta di destinazione per i protocolli *TCP* e *UDP*
  - Specificare più host di destinazione e porte

# Il comando nping

## Logica di Funzionamento

---

- Può essere utilizzato per
  - Inviare *ICMP Echo request* (similmente al comando **ping**)
  - Effettuare stress testing della rete
  - Effettuare *Address Resolution Protocol (ARP) poisoning*
  - Effettuare attacchi di tipo *Denial of Service (DoS)*
  - Etc
- Kali Linux include il comando **nping** all'interno del package Nmap

# Il comando nping

## Logica di Funzionamento

- Il comando **nping** supporta diverse *modalità di probing* configurabili mediante le seguenti opzioni

Modalità	Descrizione
--tcp-connect	Connessione <i>TCP</i> , non necessita dei privilegi di <i>root</i>
--tcp	Modalità <i>TCP</i>
--udp	Modalità <i>UDP</i>
--icmp	Modalità <i>ICMP</i> (default)
--arp	Modalità <i>ARP/RARP</i>
--tr	Modalità di traceroute (utilizzabile nelle seguenti modalità: <i>TCP</i> , <i>UDP</i> e <i>ICMP</i> )

# Il comando nping

## Esempio 1

---

- Invio di una *ICMP Echo request* ad un insieme di macchine target
  
- Indirizzi IP coinvolti
  - 10.0.2.5 [Macchina Parrot]
  - 10.0.2.6 [Macchina Metasploitable 2]
  - 10.0.2.7 [Macchina Metasploitable 3]

# Il comando nping

## Esempio 1

- Eseguiamo sulle macchine target `10.0.2.5` e `10.0.2.6` lo *script* contenente le istruzioni del firewall `iptables` (`./iptables.sh`)

```
iptables -F
iptables -t nat -F
iptables -X
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Contenuto dello script `iptables.sh`

- Prima di eseguire lo script vanno impostati i relativi permessi di esecuzione (`chmod 755 iptables.sh`)

# Il comando nping

## Esempio 1

- Invio di una *ICMP Echo request* ad un insieme di macchine target

```
nping -c 1 10.0.2.5-7
```

Sintassi per specificare gli indirizzi IP da 10.0.2.5 a 10.0.2.7

- Indirizzi IP coinvolti
  - 10.0.2.5 [Macchina Parrot]
  - 10.0.2.6 [Macchina Metasploitable 2]
  - 10.0.2.7 [Macchina Metasploitable 3]

**La macchina Parrot e la macchina Metasploitable 2 sono protette da firewall**

# Il comando nping

## Esempio 1

```
root@kali:~# nping -c 1 10.0.2.5-7
```

```
Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2019-03-23 17:12 CET
SENT (0.0365s) ICMP [10.0.2.15 > 10.0.2.5 Echo request (type=8/code=0) id=40985 seq=1]
| IP [ttl=64 id=64549 iplen=28 ]
```

```
SENT (1.0951s) ICMP [10.0.2.15 > 10.0.2.6 Echo request (type=8/code=0) id=2989 seq=1]
| IP [ttl=64 id=64549 iplen=28 ]
```

```
SENT (2.0954s) ICMP [10.0.2.15 > 10.0.2.7 Echo request (type=8/code=0) id=15590 seq=1]
| IP [ttl=64 id=64549 iplen=28 ]
```

```
RCVD (2.0967s) ICMP [10.0.2.7 > 10.0.2.15 Echo reply (type=0/code=0) id=15590 seq=1] I
| P [ttl=64 id=3055 iplen=28 ]
```

```
Statistics for host 10.0.2.5:
```

```
| Probes Sent: 1 | Rcvd: 0 | Lost: 1 (100.00%)
| Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
```

```
Statistics for host 10.0.2.6:
```

```
| Probes Sent: 1 | Rcvd: 0 | Lost: 1 (100.00%)
| Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
```

```
Statistics for host 10.0.2.7:
```

```
| Probes Sent: 1 | Rcvd: 1 | Lost: 0 (0.00%)
| Max rtt: 1.176ms | Min rtt: 1.176ms | Avg rtt: 1.176ms
```

```
Raw packets sent: 3 (84B) | Rcvd: 1 (46B) | Lost: 2 (66.67%)
```

```
Nping done: 3 IP addresses pinged in 2.13 seconds
```

La macchina target con indirizzo  
IP 10.0.2.5 non ha risposto alla  
ICMP Echo request

# Il comando nping

## Esempio 1

```
root@kali:~# nping -c 1 10.0.2.5-7
```

```
Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2019-03-23 17:12 CET
SENT (0.0365s) ICMP [10.0.2.15 > 10.0.2.5 Echo request (type=8/code=0) id=40985 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
SENT (1.0951s) ICMP [10.0.2.15 > 10.0.2.6 Echo request (type=8/code=0) id=2989 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
```

```
SENT (2.0954s) ICMP [10.0.2.15 > 10.0.2.7 Echo request (type=8/code=0) id=15590 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
RCVD (2.0967s) ICMP [10.0.2.7 > 10.0.2.15 Echo reply (type=0/code=0) id=15590 seq=1]
IP [ttl=64 id=3055 iplen=28 ]
```

```
Statistics for host 10.0.2.5:
```

```
| Probes Sent: 1 | Rcvd: 0 | Lost: 1 (100.00%)
| Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
```

```
Statistics for host 10.0.2.6:
```

```
| Probes Sent: 1 | Rcvd: 0 | Lost: 1 (100.00%)
| Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
```

```
Statistics for host 10.0.2.7:
```

```
| Probes Sent: 1 | Rcvd: 1 | Lost: 0 (0.00%)
| Max rtt: 1.176ms | Min rtt: 1.176ms | Avg rtt: 1.176ms
```

```
Raw packets sent: 3 (84B) | Rcvd: 1 (46B) | Lost: 2 (66.67%)
```

```
Nping done: 3 IP addresses pinged in 2.13 seconds
```

La macchina target con indirizzo  
IP 10.0.2.6 non ha risposto alla  
ICMP Echo request

# Il comando nping

## Esempio 1

```
root@kali:~# nping -c 1 10.0.2.5-7
```

```
Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2019-03-23 17:12 CET
SENT (0.0365s) ICMP [10.0.2.15 > 10.0.2.5 Echo request (type=8/code=0) id=40985 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
SENT (1.0951s) ICMP [10.0.2.15 > 10.0.2.6 Echo request (type=8/code=0) id=2989 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
SENT (2.0954s) ICMP [10.0.2.15 > 10.0.2.7 Echo request (type=8/code=0) id=15590 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
RCVD (2.0967s) ICMP [10.0.2.7 > 10.0.2.15 Echo reply (type=0/code=0) id=15590 seq=1]
IP [ttl=64 id=3055 iplen=28 ]
```

```
Statistics for host 10.0.2.5:
```

```
| Probes Sent: 1 | Rcvd: 0 | Lost: 1 (100.00%)
| Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
```

```
Statistics for host 10.0.2.6:
```

```
| Probes Sent: 1 | Rcvd: 0 | Lost: 1 (100.00%)
| Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
```

```
Statistics for host 10.0.2.7:
```

```
| Probes Sent: 1 | Rcvd: 1 | Lost: 0 (0.00%)
| Max rtt: 1.176ms | Min rtt: 1.176ms | Avg rtt: 1.176ms
```

```
Raw packets sent: 3 (84B) | Rcvd: 1 (46B) | Lost: 2 (66.67%)
```

```
Nping done: 3 IP addresses pinged in 2.13 seconds
```

La macchina target con indirizzo  
IP 10.0.2.7 ha risposto alla  
ICMP Echo request

# Il comando nping

## Esempio 1

```
root@kali:~# nping -c 1 10.0.2.5-7

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2019-03-23 17:12 CET
SENT (0.0365s) ICMP [10.0.2.15 > 10.0.2.5 Echo request (type=8/code=0) id=40985 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
SENT (1.0951s) ICMP [10.0.2.15 > 10.0.2.6 Echo request (type=8/code=0) id=2989 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
SENT (2.0954s) ICMP [10.0.2.15 > 10.0.2.7 Echo request (type=8/code=0) id=15590 seq=1]
IP [ttl=64 id=64549 iplen=28 ]
RCVD (2.0967s) ICMP [10.0.2.7 > 10.0.2.15 Echo reply (type=0/code=0) id=15590 seq=1] IP [ttl=64 id=3055 iplen=28 ]

Statistics for host 10.0.2.5:
| Probes Sent: 1 | Rcvd: 1 | Lost: 0 (0.00%)
| Max rtt: N/A | Min rtt: 1.0951ms | Avg rtt: 1.0951ms
Statistics for host 10.0.2.6:
| Probes Sent: 1 | Rcvd: 0 | Lost: 1 (100.00%)
| Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
Statistics for host 10.0.2.7:
| Probes Sent: 1 | Rcvd: 1 | Lost: 0 (0.00%)
| Max rtt: 1.176ms | Min rtt: 1.176ms | Avg rtt: 1.176ms
Raw packets sent: 3 (84B) | Rcvd: 1 (46B) | Lost: 2 (66.67%)
Nping done: 3 IP addresses pinged in 2.13 seconds
```

Solo 1 macchina target su 3 ha risposto alla ICMP Echo request

# Il comando nping

## Esempio 2

---

- Anche se una macchina target non risponde alla *ICMP Echo request* è ancora possibile scoprire se essa è attiva
  - Inviando un pacchetto *SYN TCP* ad una porta aperta su tale macchina
- Invio un singolo pacchetto *TCP* (opzione **--tcp**) sulla porta 22 (opzione **-p 22**) della macchina **10.0.2.5**  
**nping --tcp -c 1 -p 22 10.0.2.5**

# Il comando nping

## Esempio 2

- Invio un singolo pacchetto *TCP* (opzione **--tcp**) sulla porta 22 (opzione **-p 22**) della macchina **10.0.2.5**

```
nping --tcp -c 1 -p 22 10.0.2.5
```

```
root@kali:~# nping --tcp -c 1 -p 22 10.0.2.5

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2019-03-23 17:37 CET
SENT (0.0462s) TCP 10.0.2.15:23317 > 10.0.2.5:22 S ttl=64 id=46054 iplen=40 seq=28836
45003 win=1480
RCVD (0.0510s) TCP 10.0.2.5:22 > 10.0.2.15:23317 RA ttl=64 id=0 iplen=40 seq=0 win=0

Max rtt: 4.718ms | Min rtt: 4.718ms | Avg rtt: 4.718ms
Raw packets sent: 1 (40B) | Rcvd: 1 (46B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 1.08 seconds
```

# Il comando nping

## Esempio 2

- Invio un singolo pacchetto *TCP* (opzione **--tcp**) sulla porta 22 (opzione **-p 22**) della macchina **10.0.2.5**

```
nping --tcp -c 1 -p 22 10.0.2.5
```

```
root@kali:~# nping --tcp -c 1 -p 22 10.0.2.5

Starting Nping 0.7.70 ( https://nmap.org/nping ) at 2019-03-23 17:37 CET
SENT (0.0462s) TCP 10.0.2.15:23317 > 10.0.2.5:22 S ttl=64 id=46054 iplen=40 seq=28830
45003 win=1480
RCVD (0.0510s) TCP 10.0.2.5:22 > 10.0.2.15:23317 RA ttl=64 id=0 iplen=40 seq=0 win=0

Max rtt: 4.718ms | Min rtt: 4.718ms | Avg rtt: 4.718ms
Raw packets sent: 1 (40B) | Rcvd: 1 (46B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 1.08 seconds
```

- La macchina target ha risposto alla richiesta TCP sulla porta 22

# Target Discovery in IPv6

## THC-IPv6

---

- ***The Hacker Choice's IPv6 Attack Toolkit (THC-IPV6)***
  - Suite di comandi per effettuare numerose operazioni di rete su *IPv6*
  - In Kali (ed in tutti gli altri sistemi Debian-based) i nomi dei comandi hanno come prefisso **atk6-**
- Per maggiori informazioni
  - **man thc-ipv6**

# Target Discovery in IPv6

## THC-IPv6

---

`man thc-ipv6`

```
THC-IPv6(8)           System Manager's Manual           THC-IPv6(8)

NAME
    The Hacker Choice's IPv6 Attack Toolkit (aka thc-ipv6)

SYNOPSIS
    tool [options] ...

DESCRIPTION
    This manual page briefly documents each of the attack-toolkit6 tools.
    Not all options are listed here, to see the full list of options of
    each tool please invoke them with -h.

    Note that on Debian (if you read this on Debian) command names are
    prefixed with atk6-, so for example the tool alive6 should be in-
    volved as atk6-alive6. This is a Debian-only modification.
```

Output parziale

# THC-IPv6

atk6-alive6

---

- Scoprire le macchine attive in ambiente *IPv6* è estremamente oneroso
  - Necessario eseguire la scansione di una rete dove lo spazio degli indirizzi è enorme
- Utilizzo del protocollo ***ICMPv6 Neighbor Discovery***
  - Consente ad un host *IPv6* di rilevare gli indirizzi di tutti gli altri host *IPv6* sulla rete locale
  - E quindi di rilevare gli host attivi

# THC-IPv6

## atk6-alive6

---

- Il comando **atk6-alive6** consente di
  - Inviare richieste (*probe*) *ICMPv6* e di ottenere le relative risposte
  - Trovare gli host *IPv6* attivi sulla rete *IPv6* locale

# THC-IPv6

## atk6-alive6 – Esempio 1

---

- Nel seguente esempio assumiamo che
  - Siano attive le VM Metasploitable 1, Metasploitable 2 e Parrot
  - L'interfaccia **eth0** sia collegata alla rete LAN
- Mediante il seguente comando possiamo rilevare gli host attivi
  - **atk6-alive6 -p eth0**

```
root@kali:~# atk6-alive6 -p eth0
Alive: fe80::a00:27ff:fed3:79ab [ICMP echo-reply]
Alive: fe80::a00:27ff:feae:29e1 [ICMP echo-reply]
Alive: fe80::78cc:bcf:fae5:5fb [ICMP echo-reply]

Scanned 1 address and found 3 systems alive
```

# THC-IPv6

## detect-new-ip6

---

- Il comando **detect-new-ip6** permette di rilevare un nuovo indirizzo *IPv6* che si «unisce» alla rete locale

### ➤ Esempio

1. Digitiamo il seguente comando in Kali Linux

➤ **atk6-detect-new-ip6 eth0**

2. Avviamo Metasploitable 1

```
root@kali:~# atk6-detect-new-ip6 eth0
Started ICMP6 DAD detection (Press Control-C to end) ...
Detected new ip6 address: fe80::a00:27ff:feae:29e1
```

# Il comando nbtscan

---

- Durante un **penetration testing** su rete locale in ambiente **Windows** può essere utile ottenere informazioni sul protocollo **NetBIOS**
  - *Network Basic Input/Output System*
- Il protocollo *NetBIOS* consente di accedere a servizi di condivisione «aperta» forniti da macchine Windows-based su rete locale
  - Cartelle
  - Stampanti
  - Altri dispositivi
  - Etc

# Il comando nbtscan

---

- **nbtscan** permette di ottenere per ciascuna macchina che «espone» (non filtra) il protocollo *NetBIOS* varie informazioni
  - Indirizzo IP
  - Nome del NetBIOS
  - Servizi disponibili
  - Nome utente registrato
  - Indirizzo MAC
  - Etc

Per maggiori informazioni sul comando **nbtscan**, digitare **man nbtscan**

# Il comando nbtscan

---

```
root@kali: ~
File Edit View Search Terminal Help
nbtscan(1)      scan networks searching for NetBIOS information      nbtscan(1)

NAME
    nbtscan -- program for scanning networks for NetBIOS name information

SYNOPSIS
    nbtscan [-v] [-d] [-e] [-l] [-t timeout] [-b bandwidth] [-r] [-q]
            [-s separator] [-h] [-m retransmits] [-f filename | target]

DESCRIPTION
    NBTscan is a program for scanning IP networks for NetBIOS name information. It sends NetBIOS status query to each address in supplied range and lists received information in human readable form. For each responded host it lists IP address, NetBIOS computer name, logged-in user name and MAC address (such as Ethernet).

    NBTscan produces a report like that:

    IP address          NetBIOS Name        Server      User           MAC addr
ess
    -----
    192.168.1.2        MYCOMPUTER         JDOE        00-a0-c9

Manual page nbtscan(1) line 1 (press h for help or q to quit)
```

Output Parziale

Per maggiori informazioni sul comando **nbtscan** digitare **man nbtscan**

# Il comando nbtscan

## Esempio 1

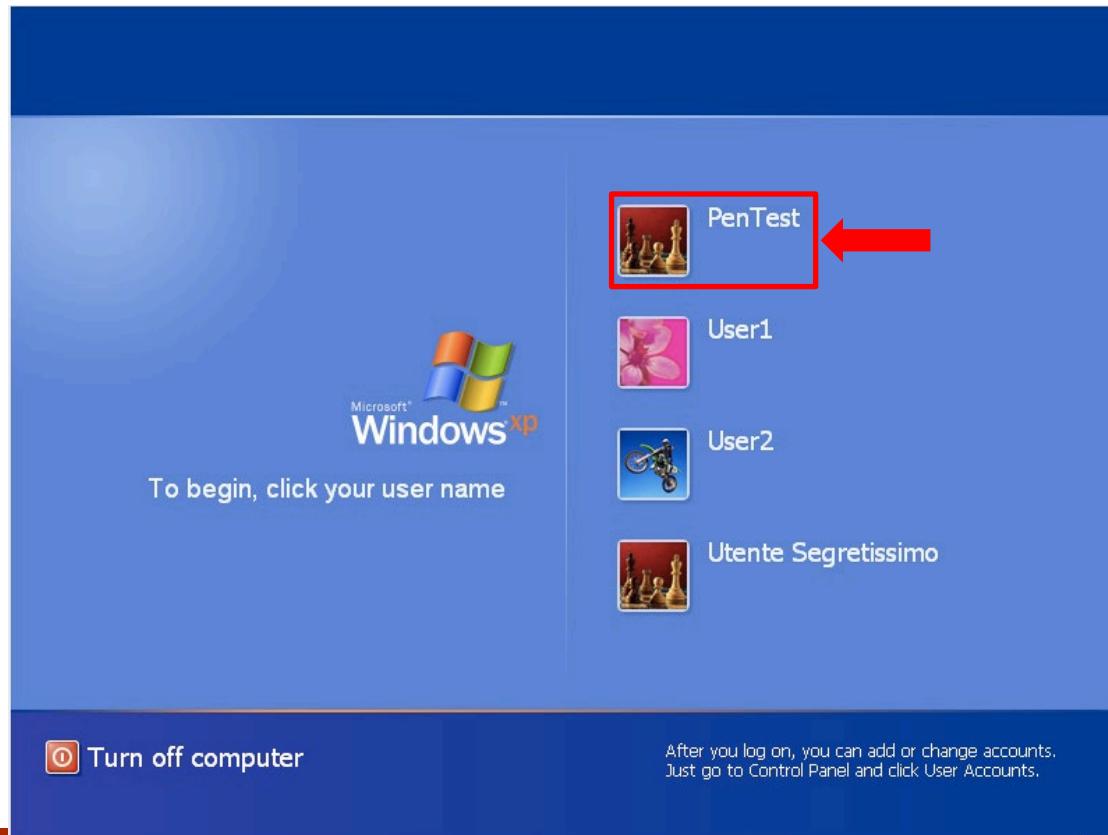
---

- Utilizziamo come asset le VM Windows XP, Metasploitable 3 e Windows 10
  - Disabilitiamo i firewall presenti in tali VM
- Eventualmente è possibile utilizzare ulteriori VM Windows-based

# Il comando nbtscan

## Esempio 1

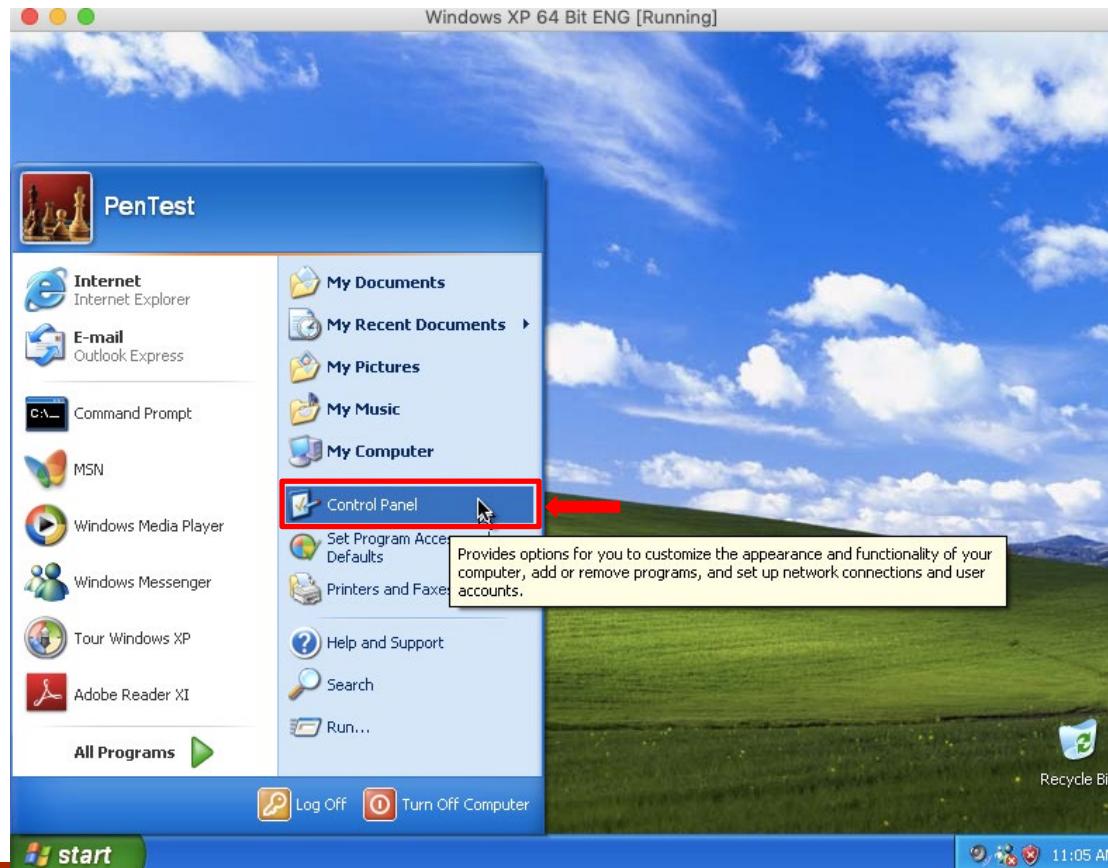
- Disabilitiamo il firewall presente in Windows XP



# Il comando nbtscan

## Esempio 1

- Disabilitiamo il firewall presente in Windows XP...

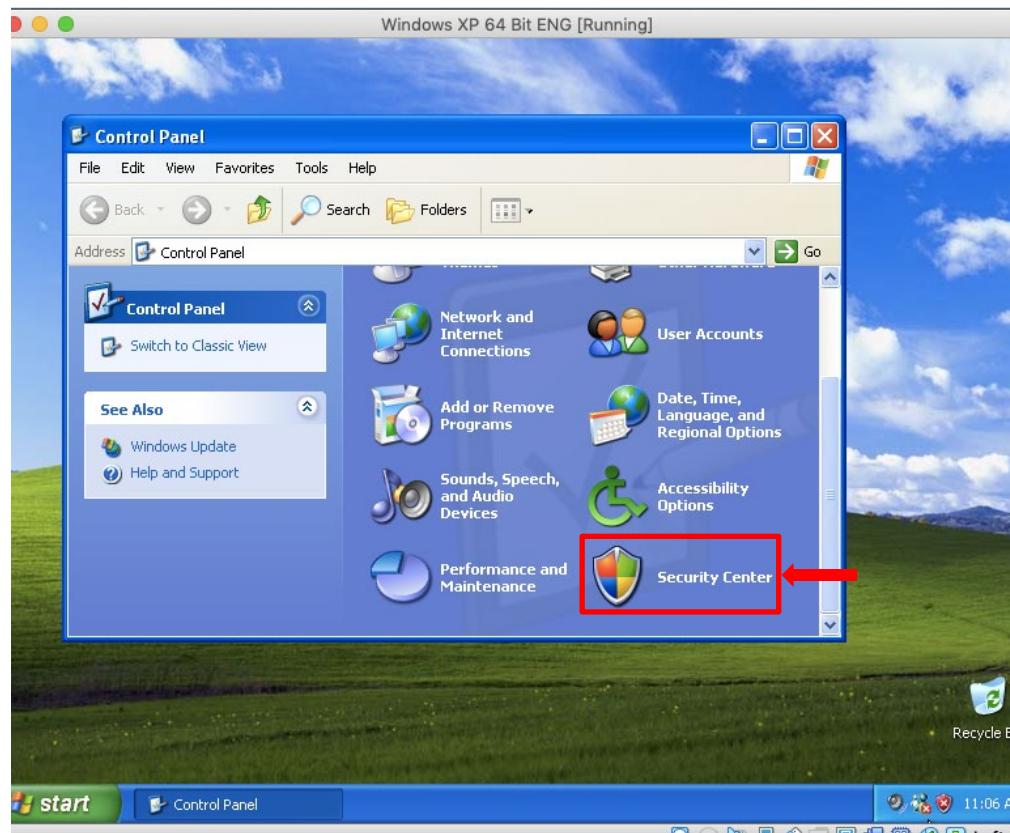


Target Discovery

# Il comando nbtscan

## Esempio 1

- Disabilitiamo il firewall presente in Windows XP...



# Il comando nbtscan

## Esempio 1

- Disabilitiamo il firewall presente in Windows XP...

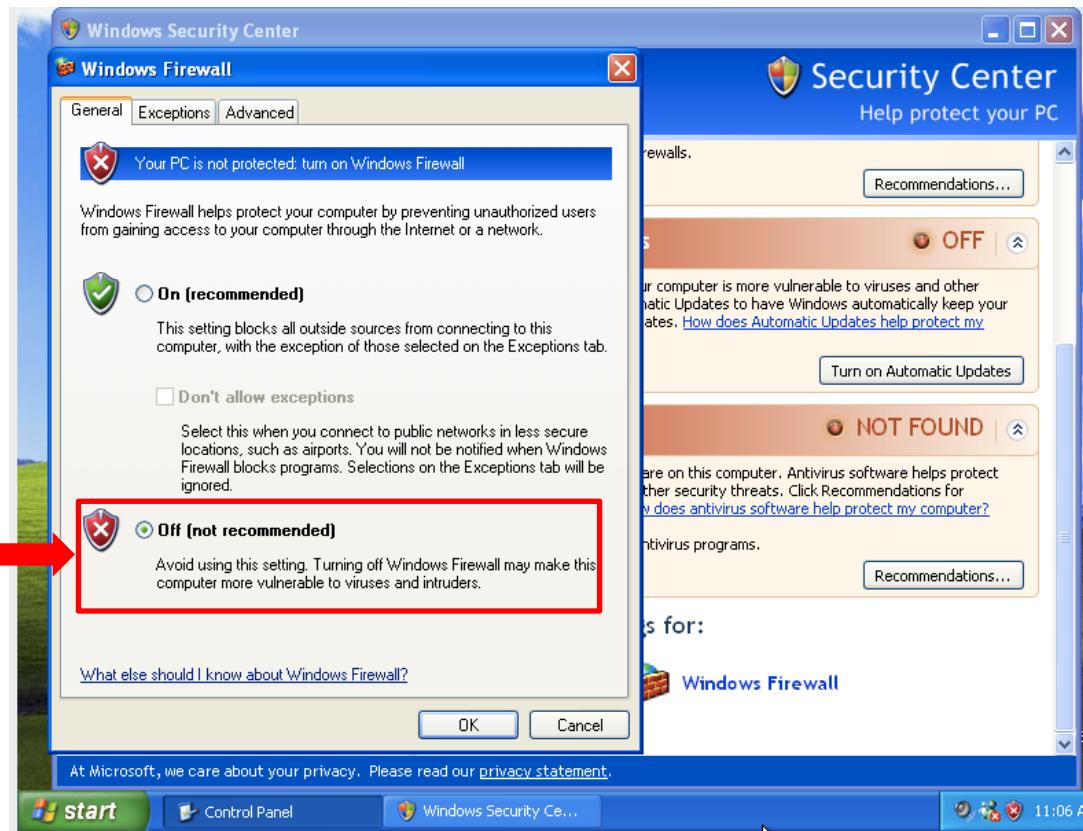


Scorrere fino a fine pagina

# Il comando nbtscan

## Esempio 1

- Disabilitiamo il firewall presente in Windows XP...



# Il comando nbtscan

## Esempio 1

- Otteniamo le informazioni sul NetBIOS delle macchine che si trovano nella rete **10.0.2.0/24**
  - **nbtscan 10.0.2.1-254**

```
root@kali:~# nbtscan 10.0.2.1-254
Doing NBT name scan for addresses from 10.0.2.1-254

IP address  NetBIOS Name    Server      User          MAC address
---  
10.0.2.7      METASPLOITABLE3 <server>   <unknown>    08:00:27:39:14:b4
10.0.2.14     WAINAKH        <server>   <unknown>    08:00:27:f1:65:3c
10.0.2.13     PRIVATE-2417C11 <server>   <unknown>    08:00:27:63:2c:eb
10.0.2.16     MSEDGEWIN10   <server>   <unknown>    08:00:27:04:18:04
```

# Il comando nbtscan

## Esempio 1

- Otteniamo le informazioni sul NetBIOS delle macchine che si trovano nella rete **10.0.2.0/24**
  - **nbtscan 10.0.2.1-254**

```
root@kali:~# nbtscan 10.0.2.1-254
Doing NBT name scan for addresses from 10.0.2.1-254

IP address  Permessi NetBIOS Name    Server      User          MAC address
-----+-----+-----+-----+-----+-----+-----+
10.0.2.7      METASPLOITABLE3 <server>  <unknown>    08:00:27:39:14:b4
10.0.2.14     WAINAKH        <server>  <unknown>    08:00:27:f1:65:3c
10.0.2.13     PRIVATE-2417C11 <server>  <unknown>    08:00:27:63:2c:eb
10.0.2.16     MSEDGEWIN10   <server>  <unknown>    08:00:27:04:18:04
```

Nell'esempio sono stati usati **due host** con **Windows XP** (uno senza Service Pack, un altro con SP3), **un host** con **Metasploitable 3** ed **un host** con **Windows 10**. Su tutti gli host è stato disabilitato il firewall

# Il comando nbtscan

## Esempio 2

- Mediante le opzioni **-hv** di **nbtscan** è possibile ottenere ulteriori informazioni sul NetBIOS, mostrandole in formato *human-readable*

➤ `nbtscan -hv 10.0.2.1-254`

Host 10.0.2.7 (Metasploitable 3)

```
NetBIOS Name Table for Host 10.0.2.7:  
  
Incomplete packet, 155 bytes long.  
Name          Service      Type  
-----  
WORKGROUP     Domain Name  
METASPOITABLE3 Workstation Service  
METASPOITABLE3 File Server Service  
  
Adapter address: 08:00:27:39:14:b4
```

Servizio di Condivisione  
File Remota

# Il comando nbtscan

## Esempio 2

- Mediante le opzioni **-hv** di **nbtscan** è possibile ottenere ulteriori informazioni sul NetBIOS, mostrandole in formato *human-readable*
  - `nbtscan -hv 10.0.2.1-254`

```
NetBIOS Name Table for Host 10.0.2.14:  
  
Incomplete packet, 155 bytes long.  
Name          Service      Type  
-----  
WAINAKH       Workstation Service  
WORKGROUP     Domain Name  
WAINAKH       File Server Service  
WORKGROUP     Browser Service Elections  
  
Adapter address: 08:00:27:f1:65:3c
```

Host 10.0.2.14 (Windows XP)

Servizio di Condivisione  
File Remota

# Il comando nbtscan

## Esempio 2

- Mediante le opzioni **-hv** di **nbtscan** è possibile ottenere ulteriori informazioni sul NetBIOS, mostrandole in formato *human-readable*

➤ `nbtscan -hv 10.0.2.1-254`

NetBIOS Name Table for Host 10.0.2.18:		
Name	Service	Type
PENTESTINGXP	Workstation Service	
WORKGROUP	Domain Name	
PENTESTINGXP	File Server Service	
WORKGROUP	Browser Service Elections	
WORKGROUP	Master Browser	
MSBROWSE	Master Browser	

Adapter address: 08:00:27:db:d8:91

Host 10.0.2.13 (Windows XP SP3)

Servizio di Condivisione  
File Remota

# Il comando nbtscan

## Esempio 2

- Mediante le opzioni **-hv** di **nbtscan** è possibile ottenere ulteriori informazioni sul NetBIOS, mostrandole in formato *human-readable*

➤ `nbtscan -hv 10.0.2.1-254`

Host 10.0.2.16 (Windows 10)

```
NetBIOS Name Table for Host 10.0.2.16:  
  
Incomplete packet, 155 bytes long.  
Name           Service      Type  
-----  
MSEdgeWin10    File Server Service  
MSEdgeWin10    Workstation Service  
WORKGROUP      Domain Name  
  
Adapter address: 08:00:27:04:18:04
```

**Servizio di Condivisione  
File Remota**

# Outline

---

- Concetti Preliminari
- Identificare la Macchine Target Attive
- Operating System (OS) Fingerprinting

# OS Fingerprinting

---

- Se la macchina target è in esecuzione ed è raggiungibile possiamo anche individuare il Sistema Operativo che essa utilizza
  - L'individuazione del Sistema Operativo è nota come **Operating System (OS) Fingerprinting**
  - Operazione che può essere estremamente importante per condurre in maniera efficace la fase di Enumerating Target
- Due tipologie di OS Fingerprinting
  - Attivo
  - Passivo

# OS Fingerprinting

Attivo

---

- Vengono inviati pacchetti verso la macchina target
  - Si determina il Sistema Operativo in base all'analisi delle risposte ricevute da tale macchina
  
- **Pro e Contro**
  - **Pro:** Velocità del metodo e risultati molto accurati
  - **Contro:** La macchina target potrebbe individuare il tentativo di ottenere informazioni riguardanti il suo Sistema Operativo

# OS Fingerprinting

## Passivo

---

- Concetto introdotto da Michal Zalewsky mediante lo strumento **p0f** (*maggiori dettagli in seguito*)
  
- Si basa sull'analisi dei pacchetti (tipicamente TCP) inviati durante le normali attività di rete
  
- **Pro e Contro**
  - **Pro:** Meno probabile che la macchina target si accorga che si sta cercando di ottenere informazioni sul Sistema Operativo
  - **Contro:** Lentezza del metodo e risultati meno accurati

# OS Fingerprinting Attivo

nmap

---

- Port scanner estremamente utile, potente e versatile
  - Sarà descritto in maniera dettagliata durante le prossime lezioni
  
- Può essere utilizzato per individuare il Sistema Operativo della macchina target
  - Mediante tecniche di **OS Fingerprinting Attivo**

# OS Fingerprinting Attivo

nmap – Esempio 1 (Metasploitable 2)

---

- Per individuare il Sistema Operativo della macchina target è possibile utilizzare l'opzione **-O** nel modo seguente

```
nmap -O 10.0.2.6
```

- *Output [Metasploitable 2]*

```
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 7.89 seconds
```

# OS Fingerprinting Attivo

nmap – Esempio 1 (Metasploitable 2)

- Per individuare il Sistema Operativo della macchina target è possibile utilizzare l'opzione **-O** nel modo seguente

```
nmap -O 10.0.2.6
```

- *Output [Metasploitable 2]*

```
MAC Address: 08:00:27:AE:29:E1 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
```



```
OS detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 7.89 seconds
```

# OS Fingerprinting Attivo

nmap – Esempio 2 (Metasploitable 3)

---

- Per individuare il Sistema Operativo della macchina target è possibile utilizzare l'opzione **-O** nel modo seguente

```
nmap -O 10.0.2.7
```

- *Output [Metasploitable 3]*

```
MAC Address: 08:00:27:39:14:B4 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 7
OS CPE: cpe:/o:microsoft:windows_7::sp1
OS details: Microsoft Windows 7 SP1
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 8.59 seconds
```

# OS Fingerprinting Attivo

nmap – Esempio 2 (Metasploitable 3)

- Per individuare il Sistema Operativo della macchina target è possibile utilizzare l'opzione -O nel modo seguente

```
nmap -O 10.0.2.7
```

- *Output [Metasploitable 3]*

```
MAC Address: 08:00:27:39:14:B4 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 7
OS CPE: cpe:/o:microsoft:windows_7::sp1
OS details: Microsoft Windows 7 SP1
Network Distance: 1 hop
```



```
OS detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 8.59 seconds
```

# OS Fingerprinting Passivo

## p0f – Logica di Funzionamento

---

- Strumento per l'**OS Fingerprinting Passivo**
  
- Utilizzato per identificare il Sistema Operativo delle
  - Macchine che si connettono alla macchina di testing (Kali)
  - Macchine alle quali si connette la macchina di testing (Kali)
  - Macchine alle quali la macchina di testing (Kali) tenta di connettersi
    - Ma non ci riesce, ottenendo un pacchetto **RST** come risposta
  - Macchine di cui è possibile osservare le comunicazioni
  
- Non è installato di default in Kali Linux
  - `apt-get install p0f`

# OS Fingerprinting Passivo

## p0f – Logica di Funzionamento

---

- **p0f** si basa sull'analisi dei pacchetti TCP inviati durante le normali attività di rete
  
- Sfrutta informazioni (chiamate *Informazioni Caratterizzanti*) nei pacchetti che non sono quelle di default e non seguono regole standard
  - Il loro comportamento varia in base al Sistema Operativo
  - Tali informazioni (memorizzate nel file **p0f.fp**) sono utilizzate da **p0f** per determinare il Sistema Operativo della macchina target

# OS Fingerprinting Passivo

p0f – Logica di Funzionamento – Info. Caratterizzanti

---

## ➤ Informazioni Caratterizzanti: Esempio 1

- Sistemi Linux-based
  - Linux di solito utilizza pacchetti di 56 (o 64) byte per il ping
- Sistemi Windows-based
  - Windows di solito utilizza pacchetti di 32 byte per il ping

## ➤ Informazioni Caratterizzanti: Esempio 2

- Sistemi Linux-based
  - TTL variabile (in base alla distribuzione ed alla versione del kernel Linux)
- Sistemi Windows-based
  - TTL di solito pari a 128

# OS Fingerprinting Passivo

## p0f – Help

---

- Sfruttando le informazioni caratterizzanti, **p0f** è in grado di individuare il Sistema Operativo della macchina target
- Per ottenere informazioni sul comando **p0f** è possibile consultare la relativa *man page*

**man p0f**

- *Output (Parziale)*

```
P0F(1)          General Commands Manual          P0F(1)

NAME
      p0f - identify remote systems passively

SYNOPSIS
      p0f [ -f file ] [ -i device ] [ -r file ] [ -o file ] [ -s socket ] [ 
      -u user ] [ -S limit ] [ -t c.h ] [ -m c.h ] [ -pdL ] [ 'filter rule' ] 

DESCRIPTION
      p0f uses a fingerprinting technique based on analyzing the structure of a
      TCP/IP packet to determine the operating system and other configuration
      properties of a remote host. The process is completely passive and does
      not generate any suspicious network traffic. The other host has to either:
      - connect to your network - either spontaneously or in an induced manner,
        for example when trying to establish a ftp data stream, returning a
        bounced mail, performing auth lookup, using IRC DCC, external html mail
        image reference and so on,
      - or be contacted by some entity on your network using some standard means
        (such as a web browsing); it can either accept or refuse the connection.
```

# OS Fingerprinting Passivo

## p0f – Esempio 1 (Metasploitable 2)

- Per avviare lo strumento è sufficiente digitare **p0f**

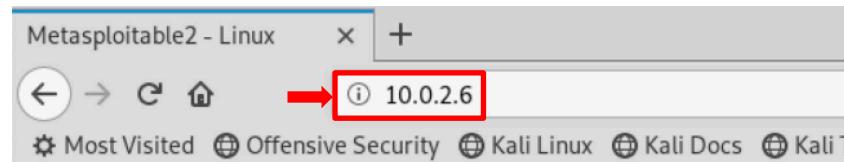
```
root@kali:~# p0f
--- p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> ---
[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on default interface 'eth0'.
[+] Default packet filtering configured [+VLAN].
[+] Entered main event loop.
```

- **p0f** resta in «attesa» di attività di rete da o verso la macchina target Metasploitable 2

# OS Fingerprinting Passivo

## p0f – Esempio 1 (Metasploitable 2)

- Per generare attività di rete stabiliremo una connessione *HTTP* ed interagiremo con il Web Server in esecuzione su Metasploitable 2
- L'indirizzo IP di Metasploitable 2 nell'esempio è **10.0.2.6**



Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

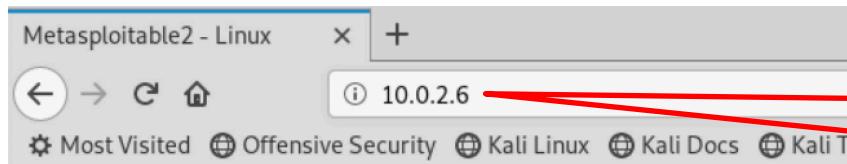
Login with msfadmin/msfadmin to get started

- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

# OS Fingerprinting Passivo

## p0f – Esempio 1 (Metasploitable 2)

- Per generare attività di rete stabiliremo una connessione *HTTP* ed interagiremo con il Web Server in esecuzione su Metasploitable 2
- L'indirizzo IP di Metasploitable 2 nell'esempio è **10.0.2.6**



Tramite browser effettuiamo una connessione all'indirizzo IP 10.0.2.6



Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

# OS Fingerprinting Passivo

## p0f – Esempio 1 (Metasploitable 2)

- Tra le informazioni generate da **p0f** possiamo osservare che è stato individuato il Sistema Operativo ed anche la (probabile) versione kernel da esso utilizzato

```
.-[ 10.0.2.15/34090 -> 10.0.2.6/80 (syn+ack) ]-  
  
server = 10.0.2.6/80  
os = Linux 2.6.x  
dist = 0  
params = none  
raw_sig = 4:64+0:0:1460:mss*4,6:mss,sok,ts,nop,ws:df:0  
  
----
```

- Possiamo verificare la correttezza di tali informazioni digitando il comando **uname -a** sulla macchina target Metasploitable 2

```
root@metasploitable:/home/msfadmin# uname -a  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

# OS Fingerprinting Passivo

p0f – Esempio 2 (Metasploitable 3)

- Per avviare lo strumento è sufficiente digitare **p0f**

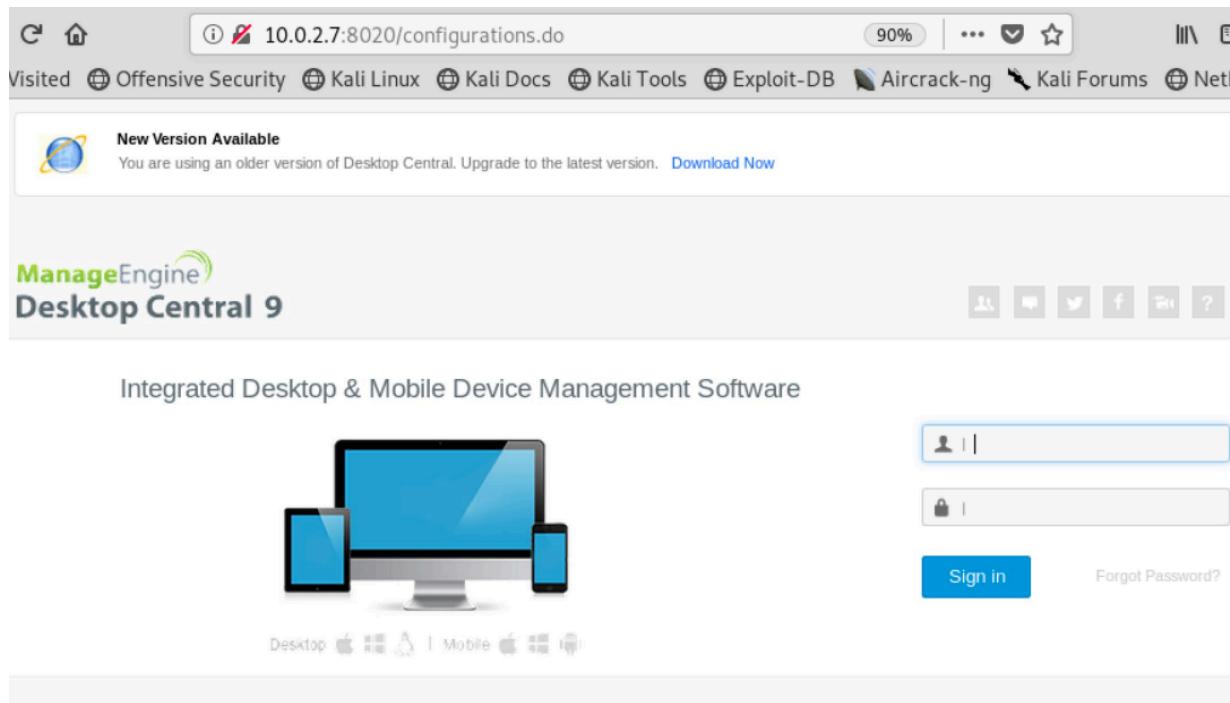
```
root@kali:~# p0f
--- p0f 3.09b by Michal Zalewski <lcamtuf@coredump.cx> ---
[+] Closed 1 file descriptor.
[+] Loaded 322 signatures from '/etc/p0f/p0f.fp'.
[+] Intercepting traffic on default interface 'eth0'.
[+] Default packet filtering configured [+VLAN].
[+] Entered main event loop.
```

- **p0f** resta in attesa di attività di rete da o verso la macchina target

# OS Fingerprinting Passivo

## p0f – Esempio 2 (Metasploitable 3)

- Per generare attività di rete stabiliremo una connessione *HTTP* (ed interagiremo) con il Web Server in esecuzione su Metasploitable 3
- L'indirizzo IP di Metasploitable 3 nell'esempio è **10.0.2.7**, porta **8020**

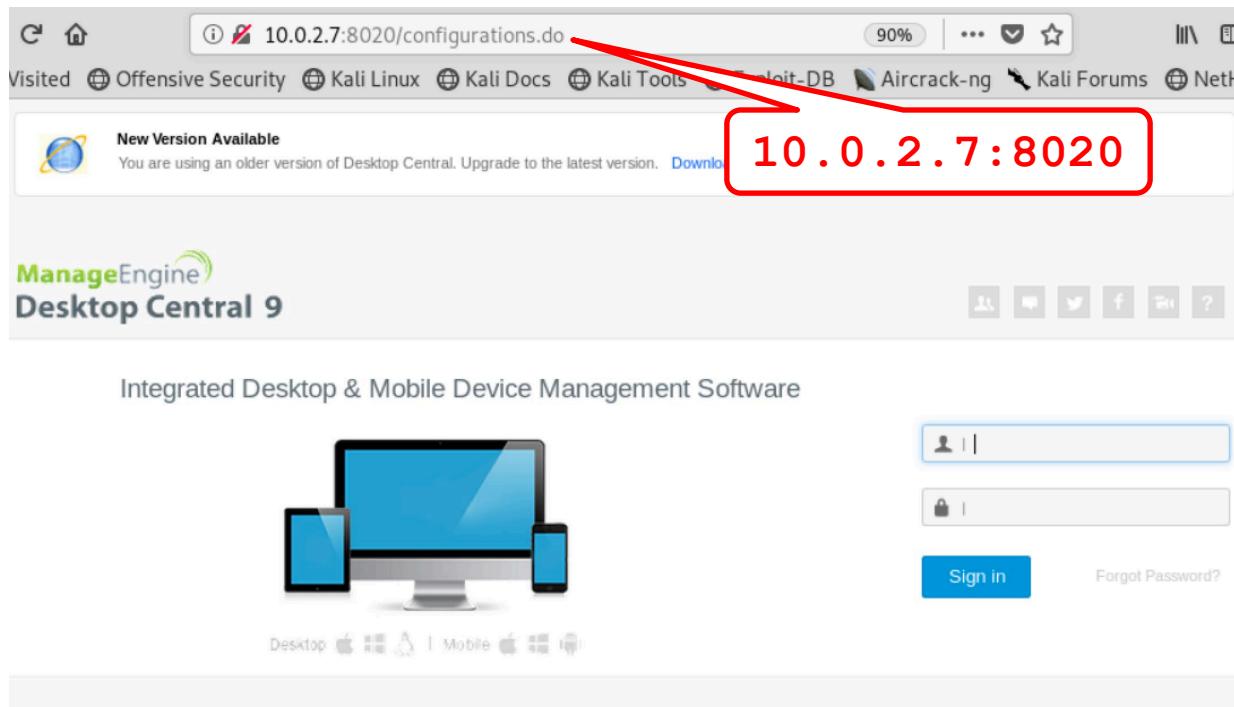


Target Discovery

# OS Fingerprinting Passivo

## p0f – Esempio 2 (Metasploitable 3)

- Per generare attività di rete stabiliremo una connessione *HTTP* (ed interagiremo) con il Web Server in esecuzione su Metasploitable 3
- L'indirizzo IP di Metasploitable 3 nell'esempio è **10.0.2.7**, porta **8020**



# OS Fingerprinting Passivo

## p0f – Esempio 2 (Metasploitable 3)

- Tra le informazioni generate da **p0f** possiamo osservare che è stato individuato correttamente il tipo di Sistema Operativo (Windows) ma non è stata individuata la corretta versione
  - Vengono suggeriti come possibili Sistemi Operativi Windows 7 o Windows 8
  - Ma Metasploitable 3 è basata su Windows 2008 Server R2

```
.-[ 10.0.2.15/59046 -> 10.0.2.7/8020 (syn+ack) ]-  
|  
| server    = 10.0.2.7/8020  
| os          = Windows 7 or 8  
| dist      = 0  
| params    = none  
| raw_sig   = 4:128+0:0:1460:8192,8:mss,nop,ws,sok,ts:df,id+:0  
|  
|-----|
```

# OS Fingerprinting Passivo

## p0f – Esempio 2 (Metasploitable 3)

- Tra le informazioni generate da **p0f** possiamo osservare che è stato individuato correttamente il tipo di Sistema Operativo (Windows) ma non è stata individuata la corretta versione
  - Vengono suggeriti come possibili Sistemi Operativi Windows 7 o Windows 8
  - Ma Metasploitable 3 è basata su Windows 2008 Server R2

```
.-[ 10.0.2.15/59046 -> 10.0.2.7/8020 (syn+ack) ]-  
|  
| server    = 10.0.2.7/8020  
| os          = Windows 7 or 8  
| dist      = 0  
| params    = none  
| raw_sig   = 4:128+0:0:1460:8192,8:mss,nop,ws,sok,ts:df,id+:0  
|  
|-----|
```

Tuttavia...

# OS Fingerprinting Passivo

## p0f – Esempio 2 (Metasploitable 3)

➤ Metasploitable 3 è basata su Windows 2008 Server R2

Windows 7	Blackcomb, Vienna	October 22, 2009	NT 6.1	<ul style="list-style-type: none"><li>Windows 7 Starter</li><li>Windows 7 Home Basic</li><li>Windows 7 Home Premium</li><li>Windows 7 Professional</li><li>Windows 7 Enterprise</li><li>Windows 7 Ultimate</li><li>Windows Thin PC</li></ul> <p>See <a href="#">Windows 7 editions</a></p>	7601 (Service Pack 1)	<ul style="list-style-type: none"><li>Mainstream support ended on January 13, 2015</li><li>Extended support ends on January 14, 2020</li></ul>
-----------	-------------------	------------------	--------	--	--------------------------	--

Windows Server 2008 R2	October 22, 2009	NT 6.1	<ul style="list-style-type: none"><li>Windows Server 2008 R2 Enterprise</li><li>Windows Server 2008 R2 Datacenter</li><li>Windows Server 2008 R2 for Itanium-based Systems</li><li>Windows Web Server 2008 R2</li><li>Windows Storage Server 2008 R2</li><li>Windows HPC Server 2008 R2</li><li><a href="#">Windows Small Business Server 2011</a></li><li>Windows MultiPoint Server 2011</li></ul>	7601	<ul style="list-style-type: none"><li>Mainstream support ended on January 13, 2015</li><li>Extended support ends on January 14, 2020</li></ul>
------------------------	------------------	--------	---	------	--

[https://en.wikipedia.org/wiki/List\\_of\\_Microsoft\\_Windows\\_versions](https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_versions)

# OS Fingerprinting Passivo

## p0f – Esempio 2 (Metasploitable 3)

➤ Metasploitable 3 è basata su Windows 2008 Server R2

Windows 7	Blackcomb, Vienna	October 22, 2009	NT 6.1 ↑	<ul style="list-style-type: none"><li>Windows 7 Starter</li><li>Windows 7 Home Basic</li><li>Windows 7 Home Premium</li><li>Windows 7 Professional</li><li>Windows 7 Enterprise</li><li>Windows 7 Ultimate</li><li>Windows Thin PC</li></ul> <p>See <a href="#">Windows 7 editions</a></p>	7601 (Service Pack 1) ↑	<ul style="list-style-type: none"><li>Mainstream support ended on January 13, 2015</li><li>Extended support ends on January 14, 2020</li></ul>
-----------	-------------------	------------------	-------------	--	-------------------------------	--

Windows Server 2008 R2	October 22, 2009	NT 6.1 ↑	<ul style="list-style-type: none"><li>Windows Server 2008 R2 Enterprise</li><li>Windows Server 2008 R2 Datacenter</li><li>Windows Server 2008 R2 for Itanium-based Systems</li><li>Windows Web Server 2008 R2</li><li>Windows Storage Server 2008 R2</li><li>Windows HPC Server 2008 R2</li><li><a href="#">Windows Small Business Server 2011</a></li><li>Windows MultiPoint Server 2011</li></ul>	7601 ↑	<ul style="list-style-type: none"><li>Mainstream support ended on January 13, 2015</li><li>Extended support ends on January 14, 2020</li></ul>
------------------------	------------------	-------------	---	-----------	--

**Windows 7 e Windows Server 2008 R2 utilizzano la stessa versione di «Kernel Windows»: NT 6.1 e la stessa build**

# Bibliografia

---

- **Kali Linux 2 - Assuring Security by Penetration Testing.**  
**Third Edition.** Gerard Johansen, Lee Allen, Tedi Heriyanto, Shakeel Ali. Packt Publishing. 2016

- Capitolo 5

