

# **Nmap for Pentester** **Ping Scan**

[WWW.HACKINGARTICLES.IN](http://WWW.HACKINGARTICLES.IN)

## Contents

Introduction.....	3
Ping Sweep .....	3
Demonstrating working of Ping Sweep using Wireshark .....	4
Block Ping Sweep Scan .....	4
Bypass Ping Sweep Filter using TCP SYN Ping .....	5
Block TCP SYN Ping Scan.....	6
Bypass TCP SYN Ping using TCP ACK Ping .....	6
Block TCP ACK Ping Scan .....	7
Bypass TCP ACK Ping using ICMP Echo .....	7
Block ICMP Echo Ping Scan .....	8
Bypass ICMP Echo Ping using ICMP Timestamp Ping .....	8
Block ICMP Ping Scan .....	9
Bypass ICMP Ping Scan using UDP Ping .....	9
Block UDP and Ping Sweep .....	10
Bypass UDP and Ping Sweep using Protocol Scan .....	11
Block IP Protocol Ping Scan .....	11
Bypass IP protocol Ping using No Ping Scan .....	12

## Introduction

We are going to scan the target machine with different Nmap ping scans, and the response packets of different scans can be confirmed by analysis of Nmap traffic through Wireshark.

A ping scan in Nmap is done to check if the target host is alive or not. As we know, ping by default sends the ICMP echo request and gets an ICMP echo reply if the system is alive. Ping scan by default sends an ARP packet and gets a response to check if the host is up.

Nmap scans change their behaviour according to the network they are scanning.

- Scanning Local Network with Nmap where nmap sends an ARP packet with every scan
- If an external network is to be scanned; Nmap sends following request packets:
  1. ICMP echo request
  2. ICMP timestamp request
  3. TCP SYN to port 443
  4. TCP ACK to port 80

For this, we are using the **--disable-arp-ping** attribute to change the behaviour of nmap scans to treat a local network as a public network.

Let's Start!!

## Ping Sweep

In order to identify a live host without using an ARP request packet, Nmap utilises the **-sP option**, which is known as Ping Sweep Scan. We can use the **-sn flag**, which means no port scan, also known as a **ping scan**.

```
nmap -sP 192.168.1.104 --disable-arp-ping
or
nmap -sn 192.168.1.104 --disable-arp-ping
```

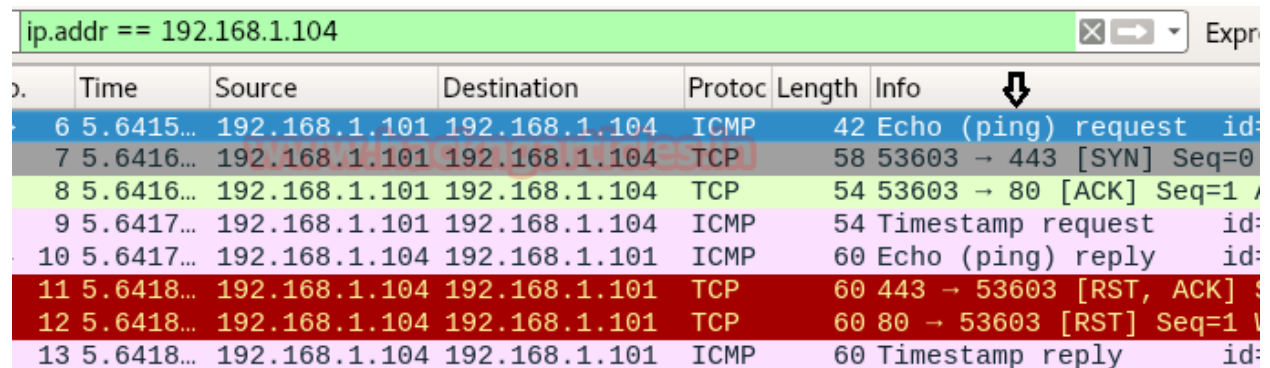
From the given below image, you can observe that it finds **1 host is up**. Because we disabled Arp request packets for local network scans with the parameter **--disable-arp-ping**, it will treat this as an external network and behave as described above.

```
root@kali:~# nmap -sP 192.168.1.104 --disable-arp-ping ↩
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 05:04 EST
Nmap scan report for 192.168.1.104
Host is up (0.00020s latency).
MAC Address: 00:0C:29:6B:71:A7 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds
root@kali:~#
```

## Demonstrating working of Ping Sweep using Wireshark

From given below image you can observe following packet of request and reply between both network IP.

1. ICMP echo request
2. TCP SYN to port 443
3. TCP ACK to port 80
4. ICMP timestamp request
5. ICMP echo reply
6. TCP RST, ACK to port 443
7. TCP RST to port 80
8. ICMP Timestamp Reply



No.	Time	Source	Destination	Protoc	Length	Info
6	5.6415...	192.168.1.101	192.168.1.104	ICMP	42	Echo (ping) request id:
7	5.6416...	192.168.1.101	192.168.1.104	TCP	58	53603 → 443 [SYN] Seq=0
8	5.6416...	192.168.1.101	192.168.1.104	TCP	54	53603 → 80 [ACK] Seq=1
9	5.6417...	192.168.1.101	192.168.1.104	ICMP	54	Timestamp request id:
10	5.6417...	192.168.1.104	192.168.1.101	ICMP	60	Echo (ping) reply id:
11	5.6418...	192.168.1.104	192.168.1.101	TCP	60	443 → 53603 [RST, ACK]
12	5.6418...	192.168.1.104	192.168.1.101	TCP	60	80 → 53603 [RST] Seq=1
13	5.6418...	192.168.1.104	192.168.1.101	ICMP	60	Timestamp reply id:

## Block Ping Sweep Scan

Now let's put some firewall rules in IPTABLES to drop ICMP packets, TCP SYN packets on port 443 and TCP ACK on port 80, which will block Ping Sweep scans.

```
sudo iptables -I INPUT -p ICMP -j DROP
sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK --dport 80 -j DROP
sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN --dport 443 -j DROP
```

```
pentest@ubuntu:~$ sudo iptables -I INPUT -p ICMP -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK --dport 80 -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN --dport 443 -j DROP
pentest@ubuntu:~$
```

Now repeat the ping sweep scan again to identify the state of the live host. From the given below image, you can observe that this time it shows that **0 hosts are up**, which means the firewall has blocked packets sent by this scan.

```
root@kali:~# nmap -sP 192.168.1.104 --disable-arp-ping
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 05:06 EST
Note: Host seems down. If it is really up, but blocking our ping probes,
Nmap done: 1 IP address (0 hosts up) scanned in 3.09 seconds
root@kali:~#
```

If you look at the image below, you will notice that it has not received any reply packets while demonstrating request packets of a Ping Sweep scan with Wireshark.

ip.addr == 192.168.1.104

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
2	0.3833...	192.168.1.101	192.168.1.104	ICMP	42	Echo (ping) request id=0xf772, s
3	0.3834...	192.168.1.101	192.168.1.104	TCP	58	39937 → 443 [SYN] Seq=0 Win=1024
4	0.3834...	192.168.1.101	192.168.1.104	TCP	54	39937 → 80 [ACK] Seq=1 Ack=1 Win=
5	0.3834...	192.168.1.101	192.168.1.104	ICMP	54	Timestamp request id=0x4a55, s
11	2.3860...	192.168.1.101	192.168.1.104	ICMP	54	Timestamp request id=0x1d7b, s
12	2.3862...	192.168.1.101	192.168.1.104	TCP	54	39938 → 80 [ACK] Seq=1 Ack=1 Win=
13	2.3863...	192.168.1.101	192.168.1.104	TCP	58	39938 → 443 [SYN] Seq=0 Win=1024
14	2.3864...	192.168.1.101	192.168.1.104	ICMP	42	Echo (ping) request id=0x06a5, s

## Bypass Ping Sweep Filter using TCP SYN Ping

Now, we'll try to bypass the firewall rules by using a ping scan with **TCP SYN packets**. For that, we'll use the **-PS attribute**. -PS sends TCP SYN packets on port 80 by default; we can change it by specifying the ports with it, like -PS443.

```
nmap -sP -PS 192.168.1.104 --disable-arp-ping
```

From the given below image, you can observe that it finds **1 host is up**.

```
root@kali:~# nmap -sP -PS 192.168.1.104 --disable-arp-ping
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 05:08 EST
Nmap scan report for 192.168.1.104
Host is up (0.00027s latency).
MAC Address: 00:0C:29:6B:71:A7 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
root@kali:~#
```

From the given below image, you can observe that it is showing the result, which is similar to the NMAP stealth scan. Here it is following the TCP Half connection mechanism where a SYN packet is sent on port 80 and received a SYN, ACK from port 80 and then an RST packet to reset the connection.

The difference between -sP packet on port 80 and -PS packet on port 80 is as following:

- Ping sweep scan [-sP] send TCP ACK packet on port 80 and hex value of ACK flag is 10, as the reply from host machine it receives RST packet whose hex value is 4.

- TCP SYN Ping scan send TCP SYN packet on port 80 and its hex value is 2, as a reply it received SYN, ACK packet whose value is some of their hex value i.e.  $2 + 10 = 12$  and able to bypass above firewall rule applied on port 80 for TCP ACK packet.

ip.addr == 192.168.1.104							
Time	Source	Destination	Protoc	Length	Info		
4 2.6045...	192.168.1.101	192.168.1.104	TCP	58	64604 → 80 [SYN] Seq=0 Win=1024 Le		
5 2.6048...	192.168.1.104	192.168.1.101	TCP	60	80 → 64604 [SYN, ACK] Seq=0 Ack=1		
6 2.6048...	192.168.1.101	192.168.1.104	TCP	54	64604 → 80 [RST] Seq=1 Win=0 Len=0		

## Block TCP SYN Ping Scan

Sometimes network administrators use Iptables on TCP SYN packets to drop all SYN packets in order to initiate TCP connections with all TCP ports in their network.

```
sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN -j DROP
```

As a result, it blocks the NMAP TCP SYN Ping probes so that it cannot identify the state of the live host.

```
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN -j DROP
pentest@ubuntu:~$
```

Now repeat again TCP SYN Ping for identifying the state of the live host. From given below image you can observe this time it shows that **0 host is up** which means the firewall has blocked packets send by this scan.

```
root@kali:~# nmap -sP -PS 192.168.1.104 --disable-arp-ping

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-19 05:16 EST
Note: Host seems down. If it is really up, but blocking our ping probes,
Nmap done: 1 IP address 0 hosts up scanned in 2.07 seconds
```

## Bypass TCP SYN Ping using TCP ACK Ping

In order to bypass this, we'll use a ping scan using TCP ACK packets. For that, we'll use the -PA attribute. -PA sends TCP ACK packets on port 80 by default. We can change it by specifying the ports with it, like -PA443

```
nmap -sP -PA 192.168.1.104 --disable-arp-ping
```

From the given below image, you can observe that it has found **1 host is up**.

```
root@kali:~# nmap -sP -PA 192.168.1.104 --disable-arp-ping

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-19 05:25 EST
Nmap scan report for 192.168.1.104
Host is up (0.00031s latency).
MAC Address: 00:0C:29:48:22:FD (VMware)
Nmap done: 1 IP address (1 host up) scanned in 13.12 seconds
```

When you notice the given below packets captured by Wireshark, you will find that an ACK packet is sent on port 80 as a reply to a received RST packet on port 80.

Source	Destination	Protocol	Length	Info
192.168.1.103	192.168.1.104	TCP	54	61300 → 80 [ACK] Seq=1 Ac...
192.168.1.104	192.168.1.103	TCP	60	80 → 61300 [RST] Seq=1 Wi...

## Block TCP ACK Ping Scan

Sometimes network administrators use the iptables filter on TCP ACK packets to drop all ACK packets in order to establish a TCP connection with all TCP ports in their network.

```
sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK -j DROP
```

As a result, it blocks the NMAP TCP ACK Ping probes so that it cannot identify the state of the live host.

```
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK -j DROP
pentest@ubuntu:~$
```

Now repeat TCP ACK Ping to identify the state of the live host. From the given below image, you can observe that this time it shows that **0 hosts are up**, which means the firewall has blocked packets sent by this scan.

```
root@kali:~# nmap -sP -PA 192.168.1.104 --disable-arp-ping

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 03:50 EST
Note: Host seems down. If it is really up, but blocking our ping probes,
Nmap done: 1 IP address (0 hosts up) scanned in 2.12 seconds
root@kali:~#
```

## Bypass TCP ACK Ping using ICMP Echo

In some scenarios, network admins apply a firewall filter on the TCP flag to resist unwanted TCP communication in the network. Let's consider that the network admin has blocked TCP communication by applying the filter on SYN as well as the ACK flag.

In order to bypass this rule, we'll use ping scan with ICMP packets. For that, we'll use the **-PE attribute**. **-PE** sends an ICMP echo request packet [ICMP type 8] and receives an ICMP echo reply packet [ICMP type 0].

```
nmap -sP -PE 192.168.1.104 --disable-arp-ping
```



From the given below image, you can observe that it finds **1 host is up**.

```
root@kali:~# nmap -sP -PE 192.168.1.104 --disable-arp-ping

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 11:24 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled
Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.1.104
Host is up (0.00054s latency).
MAC Address: 00:0C:29:73:D9:9A (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
```

## Block ICMP Echo Ping Scan

Usually, most network admins apply an ICMP filter on their network so that other systems or networks are not able to Ping their network.

```
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

As a result, it blocks the NMAP ICMP echo Ping probes so that it cannot identify the state of the live host.

```
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Now repeat TCP ICMP Ping to identify the state of the live host. From the given below image, you can observe that this time it shows that **0 hosts are up**, which means the firewall has blocked packets sent by this scan.

```
root@kali:~# nmap -sP -PE 192.168.1.104 --disable-arp-ping

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 11:16 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled
Try using --system-dns or specify valid servers with --dns-servers
Note: Host seems down. If it is really up, but blocking our ping probes,
Nmap done: 1 IP address (0 hosts up) scanned in 2.14 seconds
```

Demonstrating NMAP ICMP echo Ping with Wireshark shows only the ICMP request packet in the network and didn't receive any reply packet from the host network, as shown in the given below image.

ip.addr == 192.168.1.104						Expression.
No	Time	Source	Destination	Protocol	Length	Info
31	6171...	192.168.1.103	192.168.1.104	ICMP	42	Echo (ping) request id=0x1b73,
32	6189...	192.168.1.103	192.168.1.104	ICMP	42	Echo (ping) request id=0x9a71,

## Bypass ICMP Echo Ping using ICMP Timestamp Ping

In order to bypass this rule, we'll use a ping scan with ICMP packets. For that, we'll use the -PP attribute. -PP sends an **ICMP timestamp** request packet [ICMP type 13] and receives an **ICMP timestamp reply** packet [ICMP type 14].

```
nmap -sP -PP 192.168.1.104 --disable-arp-ping
```



From the given below image, you can observe that it finds **1 host is up**.

```
root@kali:~# nmap -sP -PP 192.168.1.104 --disable-arp-ping

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 11:22 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.1.104
Host is up (0.0023s latency).
MAC Address: 00:0C:29:73:D9:9A (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
```

Demonstrating NMAP ICMP timestamp Ping with Wireshark shows an ICMP timestamp request packet sent on the network and received any timestamp reply packet from the host network as shown in the given below image.

ip.addr == 192.168.1.104							
Time	Source	Destination	Protocol	Length	Info		
10.6110...	192.168.1.103	192.168.1.104	ICMP	54	Timestamp request	id=0xf493,	
10.6133...	192.168.1.104	192.168.1.103	ICMP	60	Timestamp reply	id=0xf493,	

## Block ICMP Ping Scan

It might be possible that the network admin had blocked entire types of ICMP messages by dropping all ICMP packets using the following iptables filter.

```
sudo iptables -I INPUT -p ICMP -j DROP
```

As a result, it blocks the NMAP ICMP Ping probes so that it cannot identify the state of the live host.

```
pentest@ubuntu:~$ sudo iptables -I INPUT -p ICMP -j DROP
```

Now repeat it again. ICMP Ping either -PP or PE to identify the state of the live host. From the given below image, you can observe that this time it shows that **0 hosts are up**, which means the firewall has blocked packets sent by this scan.

```
root@kali:~# nmap -sP -PP 192.168.1.104 --disable-arp-ping

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 22:06 EST
Note: Host seems down. If it is really up, but blocking our ping probes,
Nmap done: 1 IP address (0 hosts up) scanned in 2.17 seconds
```

## Bypass ICMP Ping Scan using UDP Ping

We have seen multiple ways to check if the system is live. Now, you can determine whether a system is up or not, whether it is on the local network or public network.

For example, ping scan with ICMP ping does not work, or even if a TCP packet filter is enabled on the host network, it becomes difficult to identify the live host. To bypass such types of rules, we'll use ping scan with **UDP packets**. For that, we'll use the **-PU attribute**.

```
nmap -sP -PU 192.168.1.104 --disable-arp-ping
```

When no ports are specified, PU sends UDP packets; the default is 40125. A reply received ICMP message such as "ICMP destination unreachable," indicating that the host is alive.

From the given below image, you can observe that it finds **1 host is up**.

```
root@kali:~# nmap -sP -PU 192.168.1.104 --disable-arp-ping
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-19 05:30 EST
Nmap scan report for 192.168.1.104
Host is up (0.00040s latency).
MAC Address: 00:0C:29:48:22:FD (VMware)
Nmap done: 1 IP address (1 host up) scanned in 13.11 seconds
```

Using Wireshark to demonstrate NMAP UDP Ping shows a UDP **request packet** sent on **40125** in the network and receiving an **ICMP destination unreachable** as a reply packet from the host network, as shown in the image below.

Source	Destination	Protocol	Length	Info
192.168.1.103	192.168.1.104	UDP	42	41307 → 40125 Len=0
192.168.1.104	192.168.1.103	ICMP	70	Destination unreachable

## Block UDP and Ping Sweep

Now let's put some firewall rules in IPTABLES to drop ICMP packets, TCP SYN packets on port 443 and TCP ACK on port 80, which will block Ping Sweep scans as well as drop UDP packets. It's possible that the network administrator had blocked the entire TCP packet.

```
sudo iptables -I INPUT -p ICMP -j DROP
sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK --dport 80 -j DROP
sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN --dport 443 -j DROP
sudo iptables -I INPUT -p UDP -j DROP
```

As a result, it will resist NMAP by making TCP Ping, ICMP Ping, and UDP ping so that it cannot identify the state of the live host.

```
pentest@ubuntu:~$ sudo iptables -I INPUT -p ICMP -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK --dport 80 -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN --dport 443 -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p UDP -j DROP
pentest@ubuntu:~$
```

Now repeat UDP Ping to identify the state of the live host. From the given below image, you can observe that this time it shows that **0 hosts are up**, which means the firewall has blocked packets sent by this scan.

```
root@kali:~# nmap -sP -PU 192.168.1.104 --disable-arp-ping
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 06:06 EST
Note: Host seems down. If it is really up, but blocking our ping probes
Nmap done: 1 IP address (0 hosts up) scanned in 2.09 seconds
```

## Bypass UDP and Ping Sweep using Protocol Scan

Using a Protocol Ping scan, we can identify live hosts when ICMP, TCP, and UDP have been blocked. For that, we'll use the **-PO attribute**. -PO sends IP packets with the particular protocol number placed in their IP header, if no protocols are precise, the default is to send multiple IP packets for ICMP (protocol 1), IGMP (protocol 2), and IP-in-IP (protocol 4).

```
nmap -sP -PO 192.168.1.104 --disable-arp-ping
```

From the given below image, you can observe that it finds **1 host is up**.

```
root@kali:~# nmap -sP -PO 192.168.1.104 --disable-arp-ping
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 06:07 EST
Nmap scan report for 192.168.1.104
Host is up (0.00010s latency).
MAC Address: 00:0C:29:6B:71:A7 (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

From given below image of Wireshark we can observe the following mechanism followed by Protocol ping scan.

- Send ICMP Echo to host network
- Send IGMP query to host network
- Send IPv4 (IP-in-IP) to host network
- Received ICMP Destination unreachable as the reply from Host

Source	Destination	Protocol	Length	Info
192.168.1.103	192.168.1.104	ICMP	42	Echo (ping) request id=0..
192.168.1.103	192.168.1.104	IGMPv1	42	Membership Query
192.168.1.103	192.168.1.104	IPv4	34	
192.168.1.104	192.168.1.103	ICMP	62	Destination unreachable (..

## Block IP Protocol Ping Scan

Now let's put some firewall rules in iptables to drop ICMP packets, TCP SYN packets on port 443 and TCP ACK on port 80, which will block Ping Sweep scans as well as drop UDP packets and IP protocol too on the network to prevent any kind of Ping scan. It's possible that the network administrator had blocked the entire TCP packet.

```
sudo iptables -I INPUT -p ICMP -j DROP
sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK --dport 80 -j DROP
sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN --dport 443 -j DROP
sudo iptables -I INPUT -p UDP -j DROP
sudo iptables -I INPUT -p IP -j DROP
```

As a result, it will resist NMAP by making TCP Ping, ICMP Ping, UDP ping, and protocol ping so that it cannot identify the state of the live host.

```
pentest@ubuntu:~$ sudo iptables -I INPUT -p ICMP -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL ACK --dport 80 -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p tcp --tcp-flags ALL SYN --dport 443 -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p UDP -j DROP
pentest@ubuntu:~$ sudo iptables -I INPUT -p IP -j DROP
```

Now repeat Protocol Ping to identify the state of the live host. From the given below image, you can observe that this time it shows that **0 hosts are up**, which means the firewall has blocked packets sent by this scan.

```
root@kali:~# nmap -sP -P0 192.168.1.104 --disable-arp-ping
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 06:12 EST
Note: Host seems down. If it is really up, but blocking our ping probes,
Nmap done: 1 IP address (0 hosts up) scanned in 3.09 seconds
```

## Bypass IP protocol Ping using No Ping Scan

When all of the Ping scans fail to determine whether the host is up or down, we select the last and best option, "No Ping," because we will use **-PN/-P0/-Pn** and basically perform a TCP port scan for the top 1000 ports.

If you want to prevent port scan and ping scan, use sweep ping with no ping as given below to identify whether the state of the host is up or down.

```
nmap -sP -PN 192.168.1.104 --disable-arp-ping
```

From the given below image, you can observe that it finds **1 host is up**.

```
root@kali:~# nmap -sP -PN 192.168.1.104 --disable-arp-ping
Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 06:12 EST
Nmap scan report for 192.168.1.104
Host is up.
Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds
```

# JOIN OUR TRAINING PROGRAMS

