

DDoS Attack in a Lab Setting

Alex, Erik, and Lelo | COMP482: Cybersecurity





Disclaimer

This project is for educational purposes only and was done by trained professionals. It was performed on a local, private network, hosted by Virtual Machines, and within the bounds of a lab setting. We do not advise you to replicate this type of attack.

Table of Contents

- **Project Overview**
- **Background and Environment Setup**
- **Problems and Troubleshooting**
- **The Attack**
- **Observations and Results**
- **Potential Improvements**
- **Reflection**



Overview

Conduct a successful DDoS attack on a local network of VMs, hosted on multiple machines. The attack will target a website, hosted locally, to try and disrupt access to it by using the Low-Orbit Ion Cannon tool.

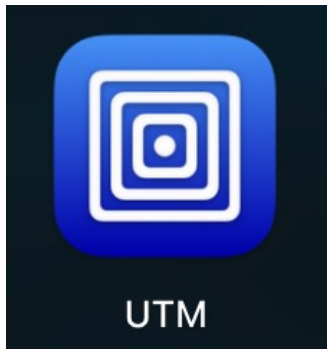




Project Background



Environment Setup



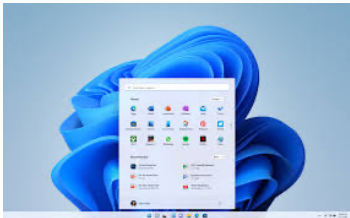
For Mac, UTM (Universal Transverse Mercator), because it is supported by M2 chips.



VirtualBox works for both Mac and PC. It is a popular choice, but Windows VMs are experimental on Mac.

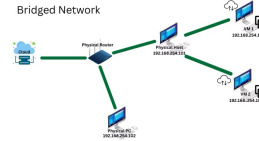


VMs and Settings



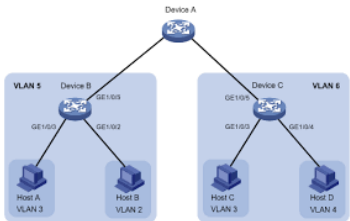
Windows 11 VM

For software, GUI and ease of usage



Bridged Adapter

For VM to appear as another device in the same network



Promiscuous

Allowed all so that we can receive all types of packets



Cable Connect

To all network (internet) into the VM

Tools Used

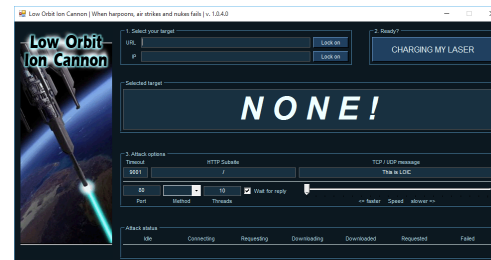
Wireshark

network packet analyzer -
lets you capture and
inspect all traffic flowing
through a network
interface



LOIC

Low Orbit Ion Cannon - Sends
massive amounts of TCP,
UDP, or HTTP requests to
a target IP

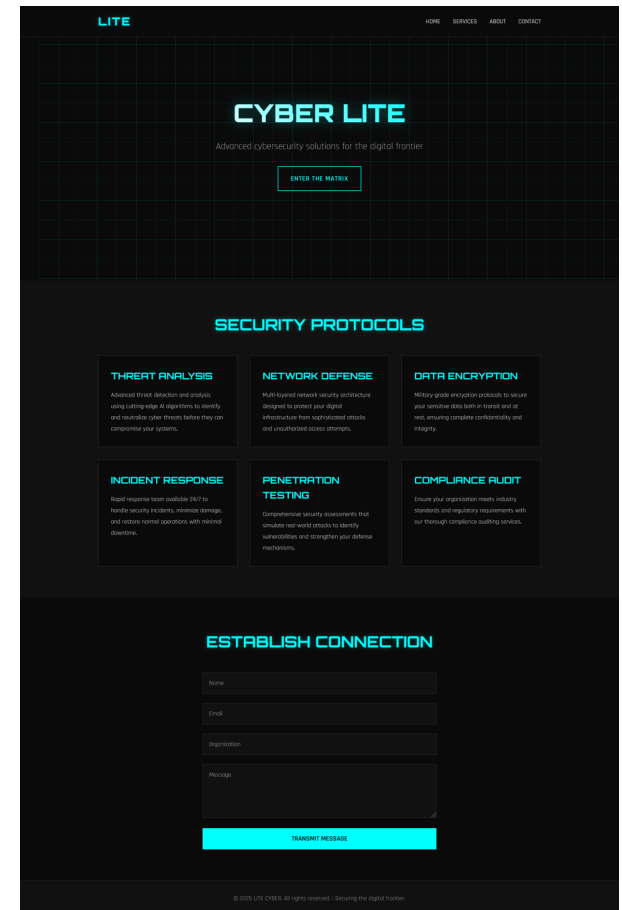


Website and Hosting

- Made a website using HTML and CSS
- Hosted locally using python

```
python -m http.server 8000
```

You need to be in the same directory





Troubleshooting



Problems and Troubleshooting



Valid IP Address

- 169.XXX... vs. 192.XXX...
- Restart the VM/Host Machine
- Restart the Wi-Fi router
- Resetting DHCP and IPv4 Protocols
- Setting a manual IP

```
Administrator: C:\WINDOWS\
Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\thor>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::9abe:2cfb:3e78:33f5%11
    IPv4 Address. . . . . : 192.168.1.228
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

C:\Users\thor>
```

Problems and Troubleshooting



Pinging another VM

- NAT Network vs Bridged Adapter
- Unique MAC Addresses
- Disabling Windows Firewall

The screenshot shows a Windows 11 desktop environment. In the background, a window titled 'Administrator: C:\WINDOWS\' displays network configuration for an 'Ethernet adapter Ethernet'. The configuration includes:

- Connection-specific DNS Suffix
- Link-local IPv6 Address : fe80::9abe:2cfb:3e78:33f5%11
- IPv4 Address. : 192.168.1.228
- Subnet Mask : 255.255.255.0
- Default Gateway : 192.168.1.1

In the foreground, a command prompt window shows the execution of a ping command:

```
C:\Users\thor>ping 192.168.1.41
```

The output of the ping command is as follows:

```
Pinging 192.168.1.41 with 32 bytes of data:
Reply from 192.168.1.41: bytes=32 time=3ms TTL=128
Reply from 192.168.1.41: bytes=32 time<1ms TTL=128
Reply from 192.168.1.41: bytes=32 time=1ms TTL=128
Reply from 192.168.1.41: bytes=32 time=3ms TTL=128

Ping statistics for 192.168.1.41:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 3ms, Average = 1ms
```

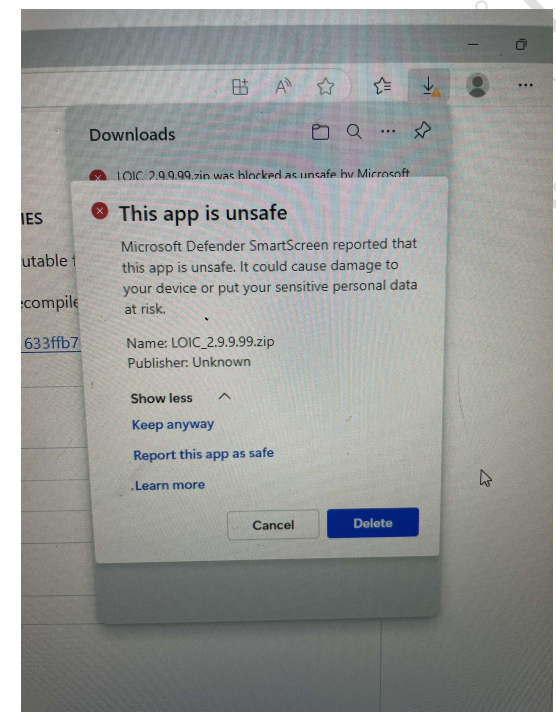
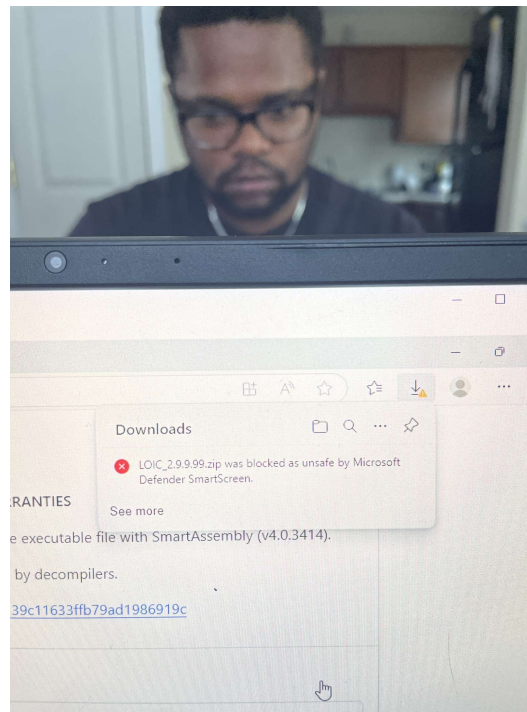
The taskbar at the bottom shows the time as 1:33 PM on 6/1/2025.

Problems and Troubleshooting

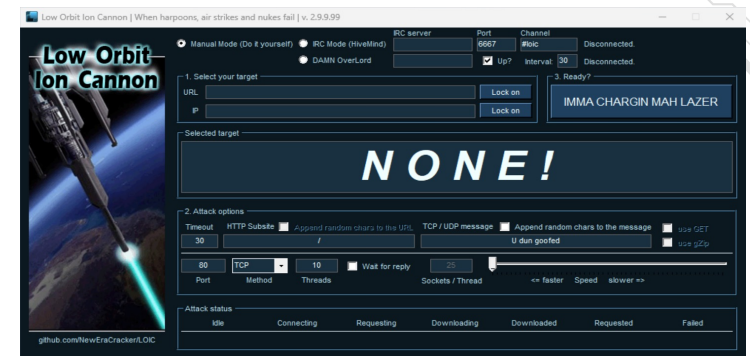
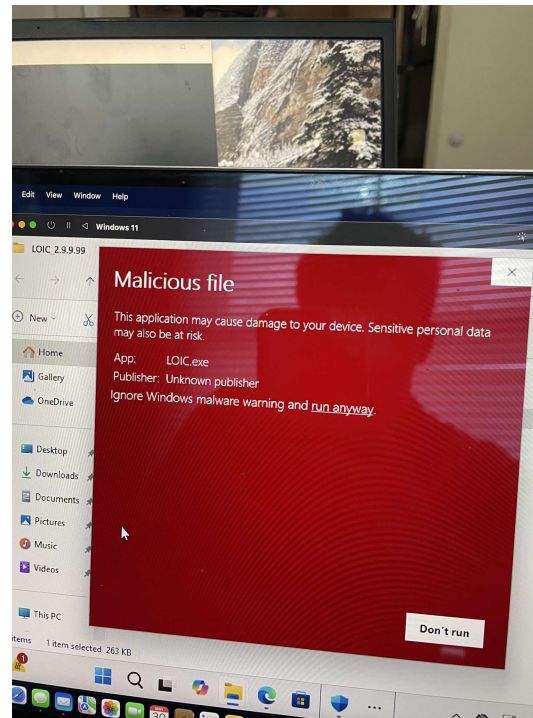
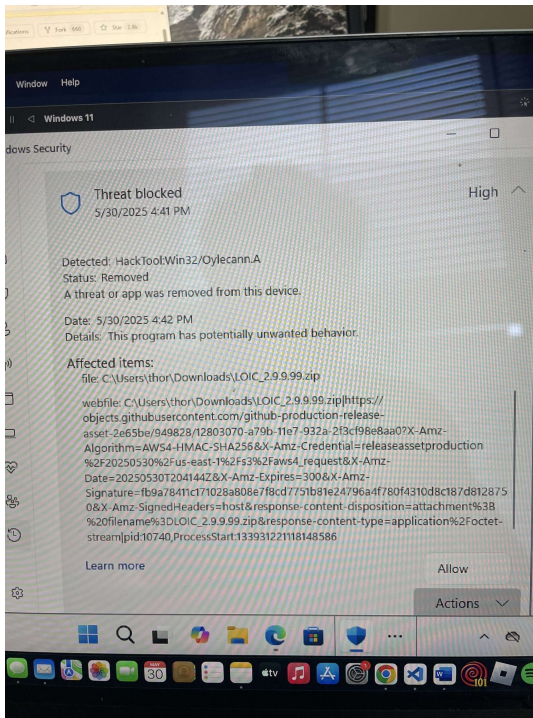


Downloading LOIC

- GitHub download
- Circumventing Windows Firewall and Microsoft Detection Antivirus
- Machine/VM errors



Problems and Troubleshooting





Execution



Low Orbit Ion Cannon

- Developed in 2010 by Praetox Technology
- Network Stress-Testing Application
- Written in C#
- Become open-source (around 2015/2016)
- Last release in October 2017
- Made popular by Anonymous
- Very easy to use



Low Orbit Ion Cannon



github.com/NewEraCracker/LOIC

☒ Manual Mode (Do it yourself) ☐ IRC Mode (HiveMind) ☐ DAMN OverLord

IRC server

Port

Channel

Disconnected.

6667

#loic

☐ Up?

Interval:

30

Disconnected.

1. Select your target

URL

Lock on

IP

Lock on

3. Ready?

IMMA CHARGIN MAH LAZER

Selected target

NONE!

2. Attack options

Timeout

30

HTTP Subsite

☐ Append random chars to the URL

/

TCP / UDP message

☐ Append random chars to the message

U dun goofed

☐ use GET

☐ use gZip

80

Port

TCP

Method

10

Threads

☐ Wait for reply

25

Sockets / Thread

<= faster Speed slower =>

Attack status

Idle

Connecting

Requesting

Downloading

Downloaded

Requested

Failed

Attack Simulation Details

- 7 Total VMs (5 Attackers, 1 Target, 1 Observer)
- All attackers were identical
- Independently controlled
- Goal: Deny a legitimate user access to the site



Step 1: Identify Target Address

1. Select your target

URL	<input type="text"/>	Lock on
IP	<input type="text" value="192.168.1.253"/>	Lock on

```
C:\Users\thor>ipconfig

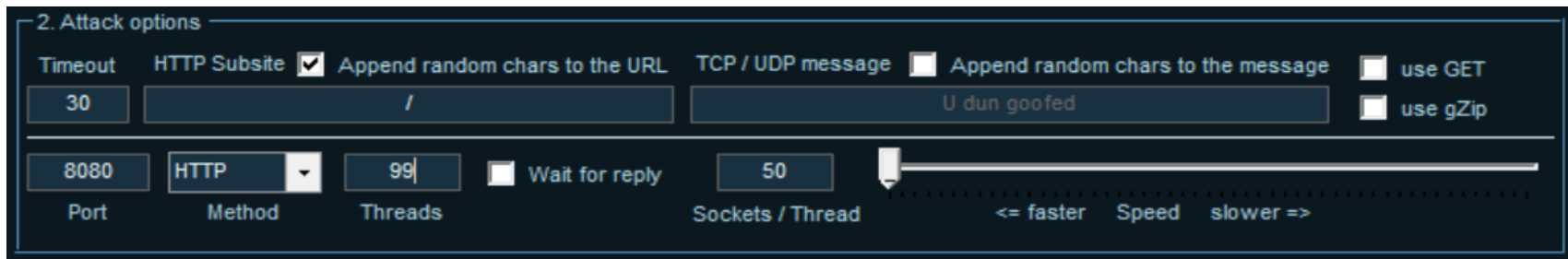
Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::9abe:2cfb:3e78:33f5%11
    IPv4 Address. . . . . : 192.168.1.228
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1
```



Step 2: Customize Attack



2. Attack options

Timeout: 30

HTTP Subsite: ☒ Append random chars to the URL: ☒ TCP / UDP message: ☐ Append random chars to the message: ☐ use GET: ☐ use gZip: ☐

Port: 8080 Method: HTTP Threads: 99 Wait for reply: ☐ Sockets / Thread: 50

Speed: <= faster Speed slower =>

URL: / Message: U dun goofed

Options:

- Port: 8080
- Threads: 99 (Max)
- Sockets/Thread: 50
- Request Type: ?

METHOD

TCP - Transmission Control Protocol

Transport Layer Protocol

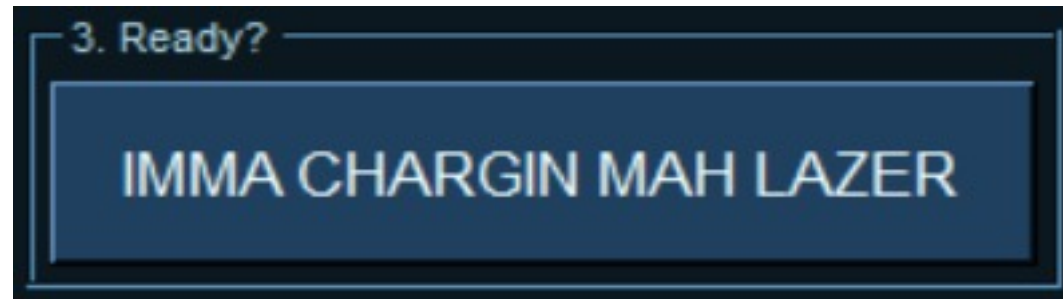
UDP - User Datagram Protocol

Internet Layer Protocol

HTTP - Hypertext Transfer Protocol

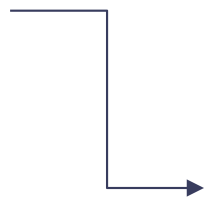
Application Layer Protocol

Step 3: CHARGE THE LASER



Options:

- DO something illegal (in most scenarios)
- DON'T do something illegal



BOOOOOOOO!

We were very careful to not do anything illegal, we promise



Low Orbit Ion Cannon



github.com/NewEraCracker/LOIC

☒ Manual Mode (Do it yourself)

☐ IRC Mode (HiveMind)

☐ DAMN OverLord

IRC server

Port

Channel

Disconnected.

6667

#loic

☒ Up?

Interval: 30

Disconnected.

1. Select your target

URL

Lock on

IP

192.168.1.253

Lock on

3. Ready?

IMMA CHARGIN MAH LAZER

Selected target

192.168.1.253

2. Attack options

Timeout

30

HTTP Subsite

☒ Append random chars to the URL

TCP / UDP message

☐ Append random chars to the message

☐ use GET

☐ use gZip

/

U dun goofed

8080

HTTP

99

☐ Wait for reply

50

Port

Method

Threads

Sockets / Thread

<= faster Speed slower =>

Attack status

Idle

Connecting

Requesting

Downloading

Downloaded

Requested

Failed

10

0

0

0

0

0

0



RESULTS

Observer VM



DDOS TEST 3.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length	Info
17	2.376911	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
22	2.382381	192.168.1.56	192.168.1.253	HTTP	194	HEAD / HTTP/1.1
33	2.393846	192.168.1.56	192.168.1.253	HTTP	194	HEAD / HTTP/1.1
34	2.393846	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
39	2.397339	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
50	2.406664	192.168.1.56	192.168.1.253	HTTP	187	HEAD / HTTP/1.1
55	2.407941	192.168.1.56	192.168.1.253	HTTP	193	HEAD / HTTP/1.1
56	2.407941	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
68	2.424634	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
71	2.429434	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
74	2.438285	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
77	2.439059	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
80	2.441288	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
83	2.444392	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
86	2.447507	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
89	2.452864	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
92	2.454946	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
101	2.578741	192.168.1.56	192.168.1.253	HTTP	193	HEAD / HTTP/1.1
104	2.578741	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
106	2.579388	192.168.1.56	192.168.1.253	HTTP	187	HEAD / HTTP/1.1
111	2.584831	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
114	2.589280	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
122	2.594200	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
123	2.594200	192.168.1.56	192.168.1.253	HTTP	193	HEAD / HTTP/1.1
129	2.602968	192.168.1.56	192.168.1.253	HTTP	193	HEAD / HTTP/1.1
139	2.611843	192.168.1.56	192.168.1.253	HTTP	193	HEAD / HTTP/1.1
141	2.611843	192.168.1.56	192.168.1.253	HTTP	193	HEAD / HTTP/1.1
143	2.612586	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
149	2.616505	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
158	2.627750	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
160	2.664039	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
162	2.669259	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
165	2.678850	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
168	2.682290	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
171	2.688846	192.168.1.253	192.168.1.56	HTTP	242	HTTP/1.0 200 OK
176	2.728054	192.168.1.56	192.168.1.253	HTTP	186	HEAD / HTTP/1.1
182	2.743435	192.168.1.56	192.168.1.253	HTTP	193	HEAD / HTTP/1.1

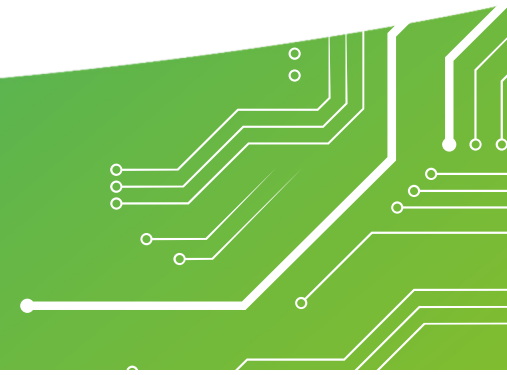
> Frame 324585: 239 bytes on wire (1912 bits), 239 bytes captured (1912 bits) on interface \Device\NPF... (200002...) | 0000 08 00 27 0c e5 33 08 00 27 31 ed f4 08 00 45 00 ...:3... '1...E-

HyperText Transfer Protocol: Protocol | Packets: 326114 - Displayed: 31756 (9.7%) | Profile: Default



~16,000+

Requests sent in 2.5 minutes



Success Metrics

1. Low Orbit Ion Cannon: Failed Requests

Attack status						
Idle	Connecting	Requesting	Downloading	Downloaded	Requested	Failed
0	82	0	0	1905	1905	4683

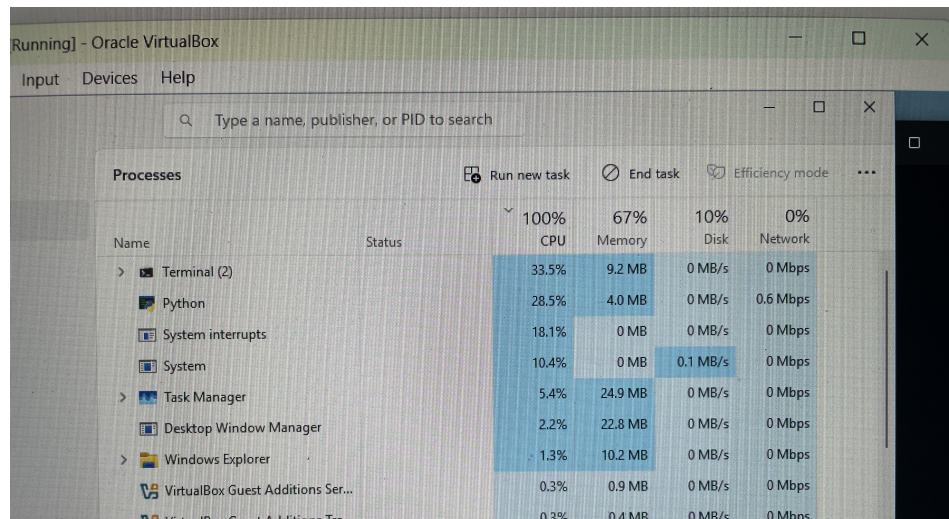
Success Metrics

2. Wireshark: Retransmission Requests

3251...	152.603287	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53458 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.603287	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53408 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.603287	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53401 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.604467	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53387 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.604467	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53367 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.604467	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53406 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.604467	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53388 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.604467	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53383 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.604467	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53386 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.605016	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53385 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.605016	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53400 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.605016	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53390 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.605016	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53405 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.605572	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53395 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3251...	152.605572	192.168.1.56	192.168.1.253	TCP	66 [TCP Port numbers reused] 53403 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM

Success Metrics

3. Target VM Task Manager: Fully-utilizing host resources

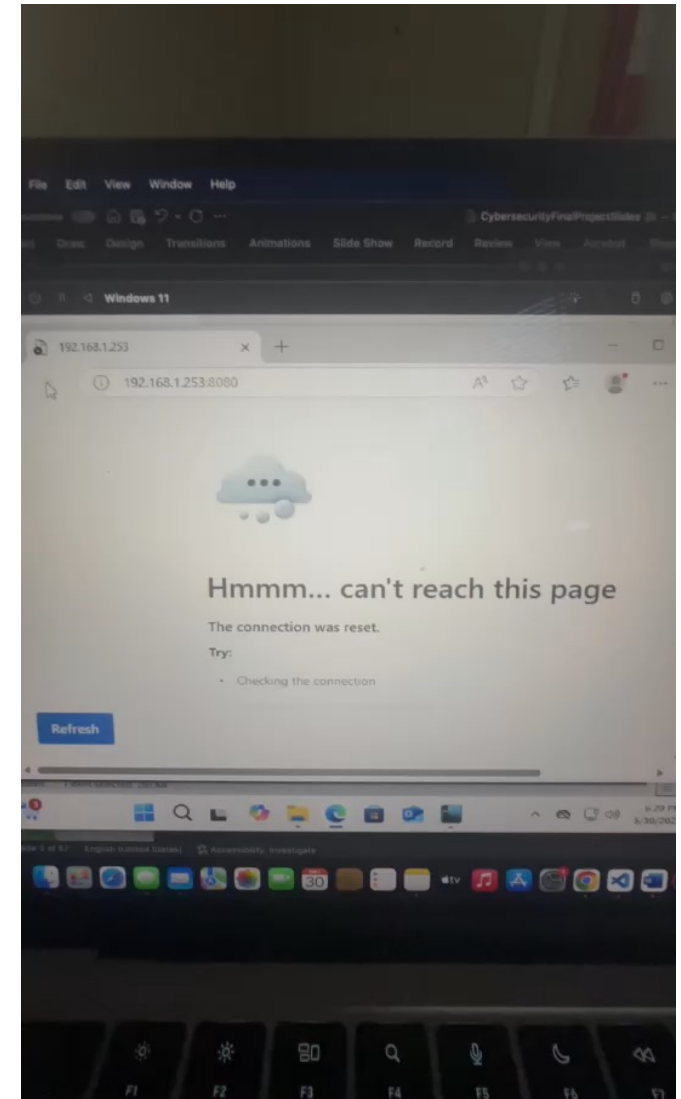
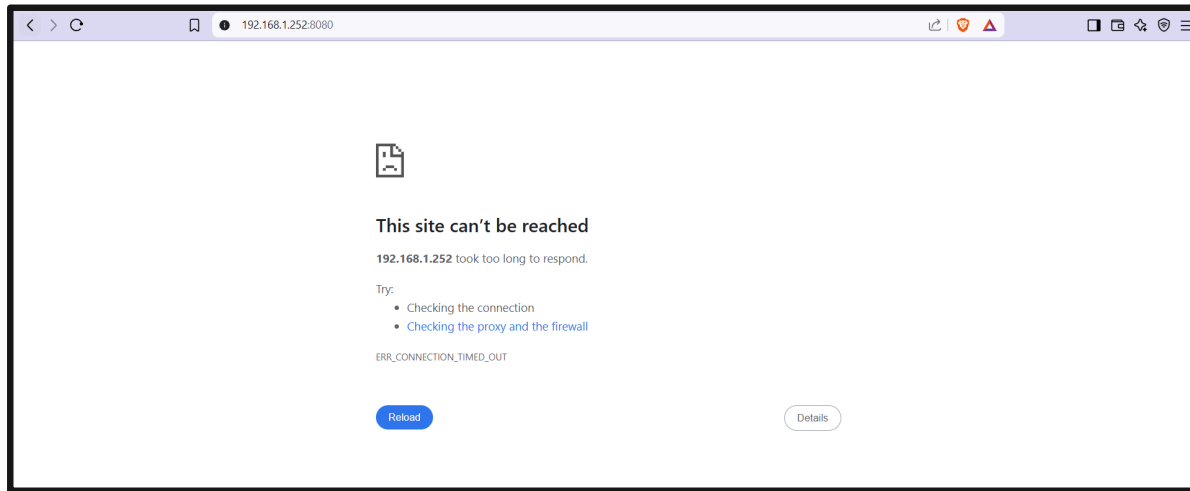


The screenshot displays the Oracle VM Task Manager interface. At the top, there's a search bar with the placeholder text 'Type a name, publisher, or PID to search'. Below this, a 'Processes' window is open, showing a list of running processes with columns for Name, Status, CPU, Memory, Disk, and Network. The 'System' process is highlighted, showing 10.4% CPU usage and 0.1 MB/s disk activity. Other processes like 'Terminal (2)', 'Python', and 'Task Manager' are also visible.

Name	Status	CPU	Memory	Disk	Network
Terminal (2)		33.5%	9.2 MB	0 MB/s	0 Mbps
Python		28.5%	4.0 MB	0 MB/s	0.6 Mbps
System interrupts		18.1%	0 MB	0 MB/s	0 Mbps
System		10.4%	0 MB	0.1 MB/s	0 Mbps
Task Manager		5.4%	24.9 MB	0 MB/s	0 Mbps
Desktop Window Manager		2.2%	22.8 MB	0 MB/s	0 Mbps
Windows Explorer		1.3%	10.2 MB	0 MB/s	0 Mbps
VirtualBox Guest Additions Ser...		0.3%	0.9 MB	0 MB/s	0 Mbps
VirtualBox Guest Additions Tra		0.3%	0.4 MB	0 MB/s	0 Mbps

Success Metrics

4. Website unavailable to Legitimate Users



Discussion Questions

- If we had a Web Application Firewall or rate limiting in place, how many of you think the attack would still succeed? Why or why not?
- What do you think would've been different if we launched this attack at Layer 3 or 4 instead of the application layer (Layer 7)?
- Do you think using tools like LOIC makes these attacks too easy? If so, should they be more restricted or harder to access?



Future Improvements

- We could have attacked the network layer
 - We attacked layer 7, which is typical, we could have shown deeper understanding by attacking layer 3 or 4
- Our own attack scripts
 - Writing our own scripts could have shown understanding and mastery
- Secured the app, so that we can find loopholes around
 - Attacking a rate limited system and one with Web Application Firewall (WAF) could have improved the robustness of our attack.
- Tried on a full-stack app
 - Instead of packet overload, we could focus on cutting down services:
 - API hangs - servers takes long to respond
 - Database overloads - database receives more queries than it can handle
 - UI timeouts - database receives more queries than it can handle



Final Thoughts

Reassurances

- Environment and attack set-up: Complicated and time consuming
- One machine is not strong enough
- Some level of knowledge required to implement the attack

Concerns

- Simplicity of the attack: Easy to replicate
- The resources are open and obtainable
- Organizations with the resources will have the means to conduct large-scale attacks



Sources

- <https://www.cloudflare.com/learning/ddos/ddos-attack-tools/low-orbit-ion-cannon-loic/>
- <https://openhub.net/p/LOIC>
- <https://github.com/NewEraCracker/LOIC/releases>
- https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Tools_and_setup/set_up_a_local_testing_server
- <https://www.geeksforgeeks.org/differences-between-tcp-and-udp/>
- <https://www.akamai.com/glossary/what-is-a-layer-3-attack>
- <https://developers.cloudflare.com/waf/rate-limiting-rules/best-practices/>
- <https://www.catchpoint.com/api-monitoring-tools/api-gateway-timeout>
- <https://www.virtualbox.org/wiki/Downloads>
- <https://mac.getutm.app/>
- <https://www.microsoft.com/en-us/software-download/windows11>
- <https://www.wireshark.org/download.html>





Questions?