BYPASSING NGFW'S LAYER 7 APPLICATION POLICY

Ali Efe - @0xaefe

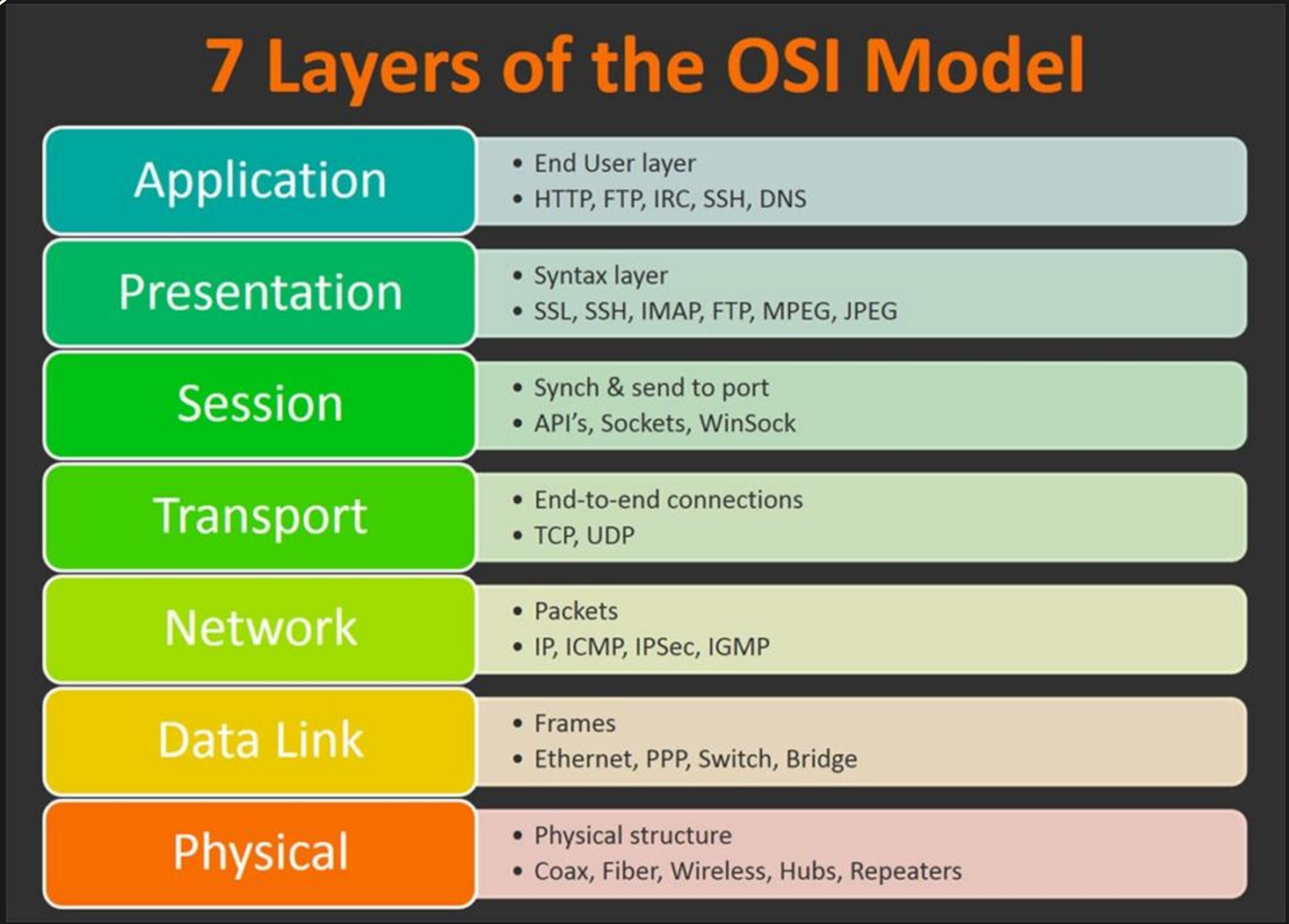
BSides Vancouver 2024

AGENDA

WHOAMI

- Penetration Tester at IBM X-Force Red
- Testing applications and networks
- Cybersecurity enthusiast
- Former x (instructor, web developer, IT guy)
- OSCP, CRTP, GWAPT

INTRODUCTION



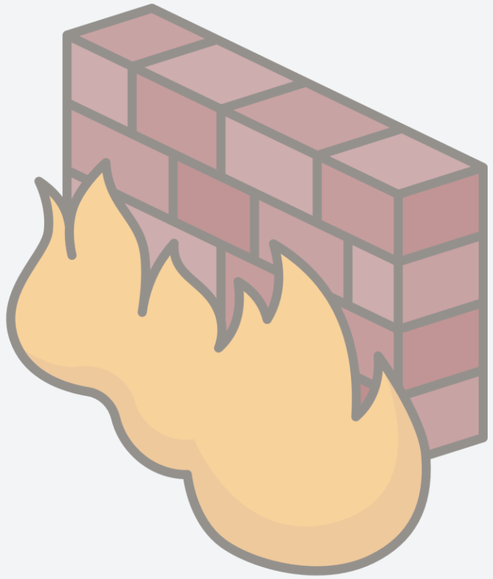
** Source: https://int0x33.medium.com/day-51-understanding-the-osi-model-f22d5f3df756*

INTRODUCTION

Traditional Firewall

- Packet filtering, NAT, stateful inspection, VPN
- Block/Allow: TCP/UDP ports, IP addresses

" ... more than 80% of all new malware and intrusion attempts are exploiting weaknesses in applications, as opposed to weaknesses in networking components and services." *



Next Generation Firewall (NGFW)

- Block/Allow: IP addresses, TCP/UDP ports, application in use, content
- More OSI layers
- Integrated IDS/IPS capabilities
- This talk's focus: Layer 7

* Source: https://en.wikipedia.org/wiki/Next-generation_firewall

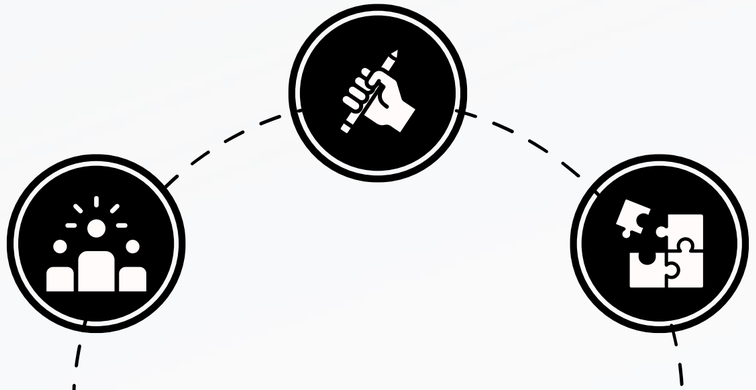
SOMETHING LOOKED WEIRD

Questions led me to research - #1

An interesting but weird case my friend in IR told me...

"... a repeating egress traffic, from the same source to the same target over the same port..."

"... but terminating TCP session without sending much data, just a few packets..."



SOMETHING LOOKED WEIRD

Questions led me to research - #2

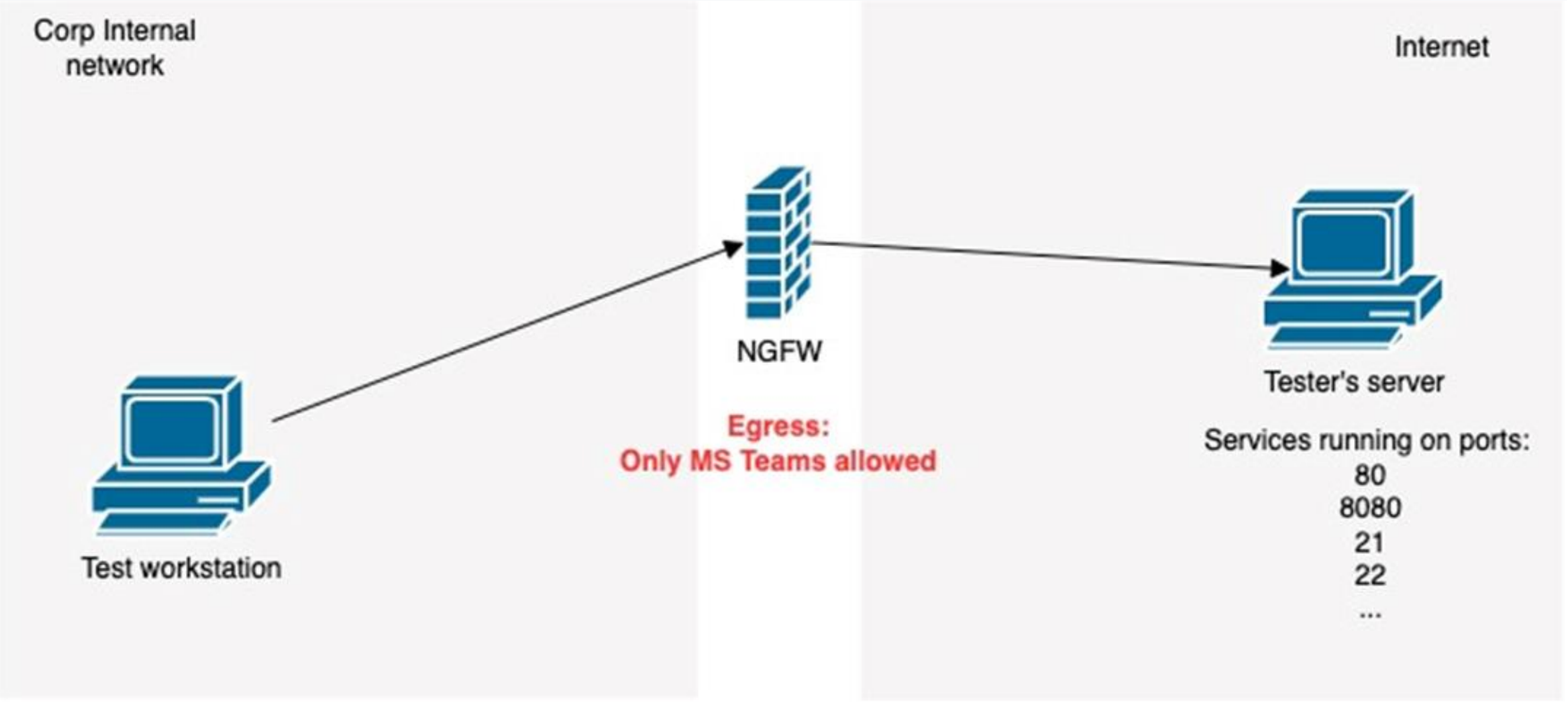
Another ordinary day as a pentester, testing firewall egress traffic

Expected:
Only MS Teams should be allowed (TCP ports 80, 443)

Found:
Many ports open

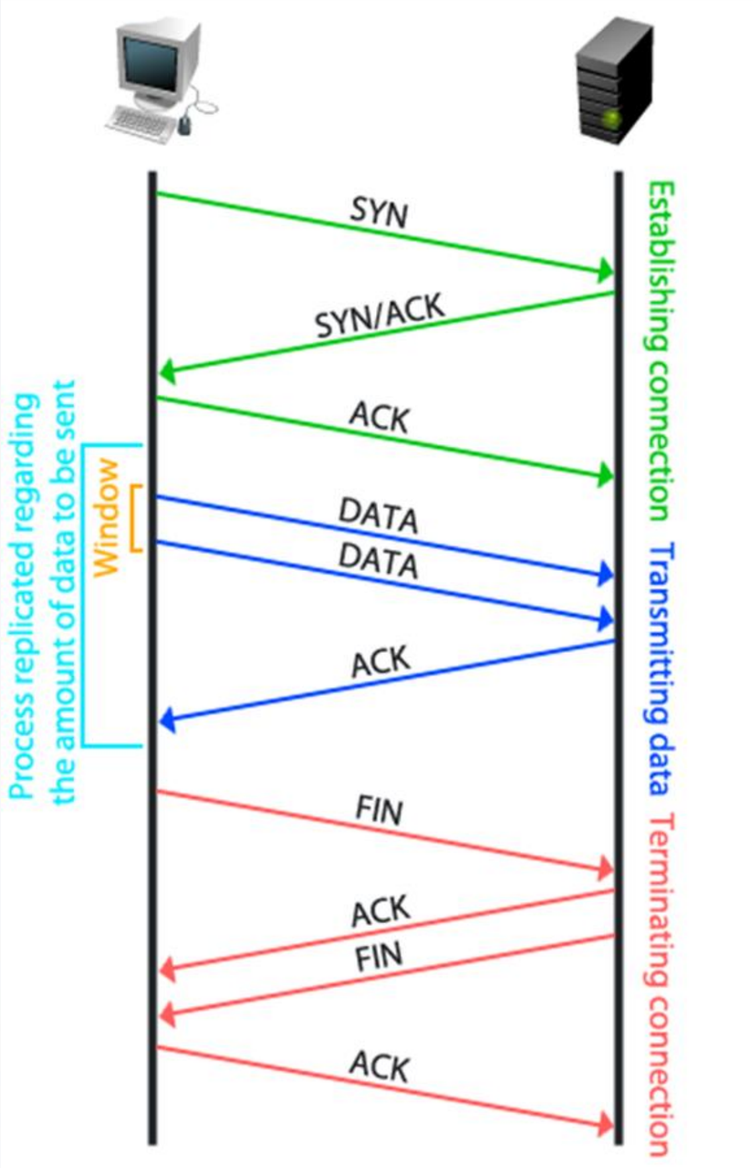
Connect using client apps:
Ex. ssh,ftp etc., but no success!

...Bug or something else?



SOMETHING LOOKED WEIRD

Questions led me to research - #2



*Source: https://toschprod.wordpress.com/2012/01/30/osi-model-layer-4-transport/

SOMETHING LOOKED WEIRD

```
ali@linux-victim:~$ sudo nmap -sS -v linux-attacker
Starting Nmap 7.70 ( https://nmap.org ) at 2022-04-20 18:32 UTC
Initiating Ping Scan at 18:32
Scanning linux-attacker
Completed Ping Scan at
Initiating SYN Stealth
Scanning linux-attacker
Discovered open port 80
Discovered open port 80
Discovered open port 21
Discovered open port 22
Increasing send delay f
Increasing send delay f
Increasing send delay f
Increasing send delay f
Increasing send delay f
SYN Stealth Scan Timing
Debugging Increased to
Debugging Decreased to
Verbosity Increased to
Stats: 0:01:19 elapsed;
SYN Stealth Scan Timing
Completed SYN Stealth S
Nmap scan report for li
Host is up (0.0024s lat
Scanned at 2022-04-20 1
Not shown: 994 closed p
PORT      STATE    SERVI
21/tcp    open     ftp
22/tcp    open     ssh
53/tcp    filtered domai
80/tcp    open     http
515/tcp   filtered print
8080/tcp  open     http-

Read data files from: /
Nmap done: 1 IP address
           Raw packets
ali@linux-victim:~$
```

ip.addr == 10.1.1.4

No.	Time	Source	Destination	Protocol	Length	Info
112	5.082731	10.1.1.4	10.1.1.5	TCP	60	63100 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
113	5.082760	10.1.1.5	10.1.1.4	TCP	54	443 → 63100 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
114	5.082766	10.1.1.4	10.1.1.5	ICMP	60	Echo (ping) request id=0xe6eb, seq=0/0, ttl=39 (reply in 115)
115	5.082772	10.1.1.5	10.1.1.4	ICMP	42	Echo (ping) reply id=0xe6eb, seq=0/0, ttl=64 (request in 114)
116	5.082774	10.1.1.4	10.1.1.5	ICMP	60	Timestamp request id=0xb0a2, seq=0/0, ttl=54
117	5.082777	10.1.1.5	10.1.1.4	ICMP	54	Timestamp reply id=0xb0a2, seq=0/0, ttl=64
118	5.287380	10.1.1.4	10.1.1.5	TCP	60	15720 → 110 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
119	5.287408	10.1.1.5	10.1.1.4	TCP	54	110 → 15720 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
120	5.287414	10.1.1.4	10.1.1.5	TCP	60	58787 → 554 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
121	5.287417	10.1.1.5	10.1.1.4	TCP	54	554 → 58787 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
122	5.287419	10.1.1.4	10.1.1.5	TCP	60	30970 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
123	5.287431	10.1.1.5	10.1.1.4	TCP	58	80 → 30970 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
124	5.287433	10.1.1.4	10.1.1.5	TCP	60	2899 → 8080 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
125	5.287438	10.1.1.5	10.1.1.4	TCP	58	8080 → 2899 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
126	5.287440	10.1.1.4	10.1.1.5	TCP	60	53192 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
127	5.287442	10.1.1.5	10.1.1.4	TCP	54	443 → 53192 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
128	5.287444	10.1.1.4	10.1.1.5	TCP	60	31464 → 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
129	5.287446	10.1.1.5	10.1.1.4	TCP	54	135 → 31464 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
130	5.287448	10.1.1.4	10.1.1.5	TCP	60	55567 → 3389 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
131	5.287451	10.1.1.5	10.1.1.4	TCP	54	3389 → 55567 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
132	5.287453	10.1.1.4	10.1.1.5	TCP	60	34977 → 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
133	5.287455	10.1.1.5	10.1.1.4	TCP	54	25 → 34977 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
134	5.287457	10.1.1.4	10.1.1.5	TCP	60	25298 → 5900 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
135	5.287459	10.1.1.5	10.1.1.4	TCP	54	5900 → 25298 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
136	5.289229	10.1.1.4	10.1.1.5	TCP	60	30970 → 80 [RST] Seq=1 Win=0 Len=0
137	5.289236	10.1.1.4	10.1.1.5	TCP	60	2899 → 8080 [RST] Seq=1 Win=0 Len=0
138	6.389136	10.1.1.4	10.1.1.5	TCP	60	50125 → 199 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
139	6.389167	10.1.1.5	10.1.1.4	TCP	54	199 → 50125 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
140	6.389174	10.1.1.4	10.1.1.5	TCP	60	60392 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
141	6.389178	10.1.1.5	10.1.1.4	TCP	54	23 → 60392 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
142	6.389180	10.1.1.4	10.1.1.5	TCP	60	59282 → 5959 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
143	6.389182	10.1.1.5	10.1.1.4	TCP	54	5959 → 59282 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
144	6.389184	10.1.1.4	10.1.1.5	TCP	60	3851 → 1025 [SYN] Seq=0 Win=1024 Len=0 MSS=1368
145	6.389187	10.1.1.5	10.1.1.4	TCP	54	1025 → 3851 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

SOMETHING LOOKED WEIRD

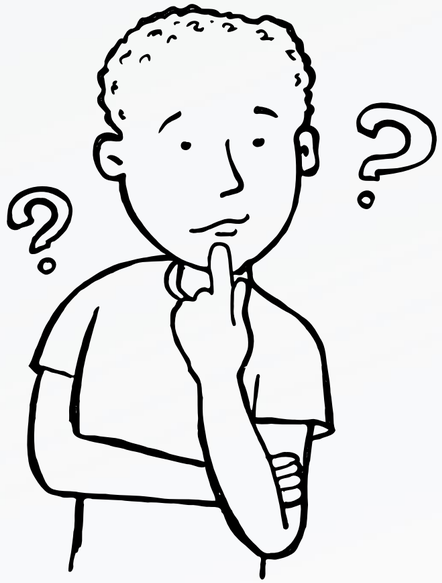
Questions led me to research - #2

Digging more...

Connect using Netcat was successful!

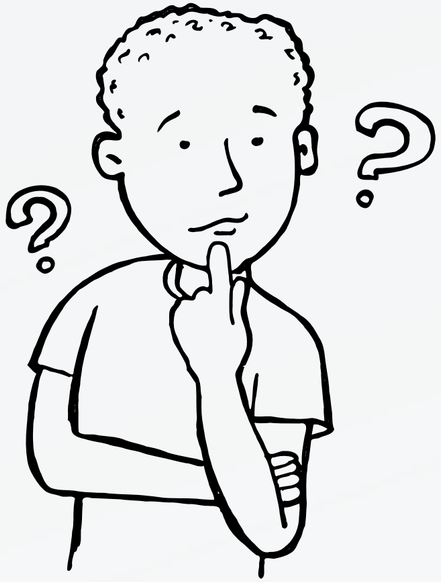
Reverse shell using Netcat was successful but got blocked shortly after

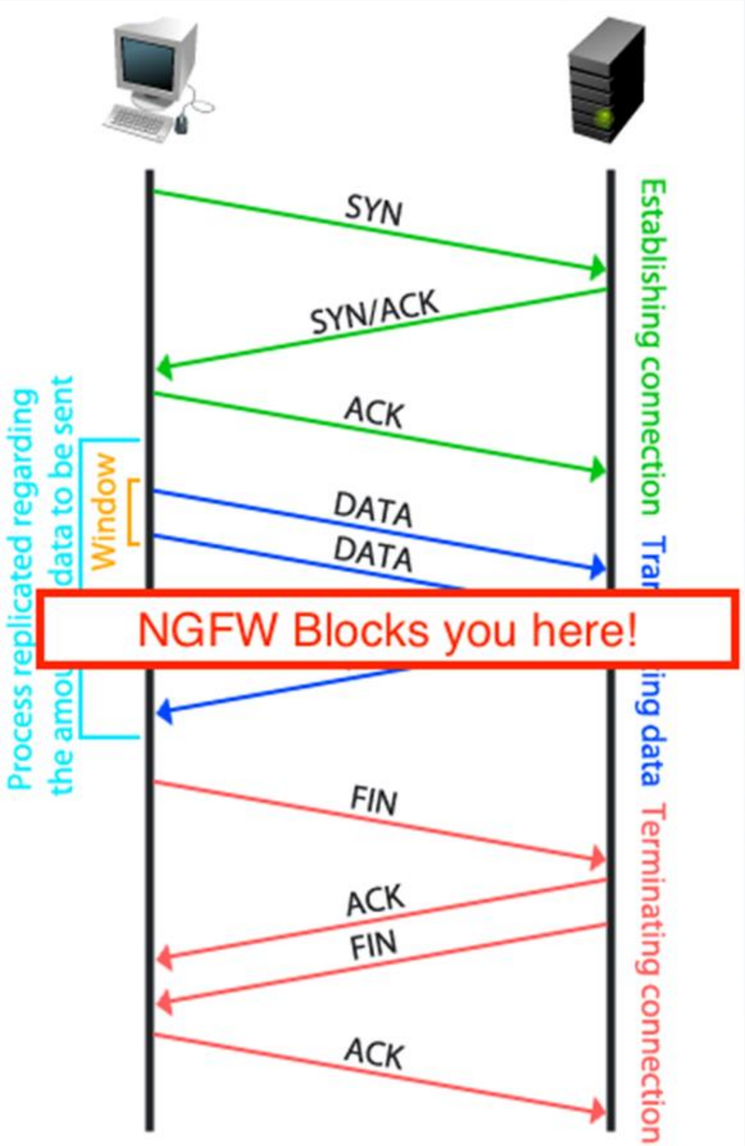
Still, some data made to the target server!



SOMETHING LOOKED WEIRD

Questions led me to research - #2





SOMETHING LOOKED WEIRD

WORKING AS INTENDED

Cisco Firepower (FTD uses Snort) documents had a section explaining what I was looking for.

cisco.com/c/en/us/support/docs/security/firepower-ngfw/212321-clarify-the-firepower-threat-defense-acc.html#anc10

work links | resources | IDOR | open tabs

Appid 676:1 = HTTP

The deployed policy in LINA.

Note: The rule is pushed as a **permit** action because LINA cannot determine that the session uses HTTP. On FTD the Application Detection mechanism is in Snort engine.

```
firepower# show access-list
…
access-list CSM_FW_ACL_ line 9 remark rule-id 268435461: L7 RULE: Rule1
access-list CSM_FW_ACL_ line 10 advanced permit ip host 192.168.1.40 host 192.168.2.40 rule-id 268435461 (hitcnt=0) 0xb788b786
```

For a Block Rule that uses Application as a condition, the trace of a real packet shows that the session is dropped by the LINA due to Snort engine verdict.

Note: In order for the Snort engine to determine the application it has to inspect a few packets (usually 3-10 which depends on the application decoder). Thus a few packets are allowed through the FTD and they make it to the destination. The allowed packets are still subject to the Intrusion Policy check based on the **Access Policy > Advanced > 'Intrusion Policy used before Access Control rule is determined'** option.

Verify Behavior:

When host-A (192.168.1.40) tries to establish an HTTP session with host-B (192.168.2.40) the LINA ingress capture shows:

Source: https://www.cisco.com/c/en/us/support/docs/security/firepower-ngfw/212321-clarify-the-firepower-threat-defense-acc.html#anc10

WORKING AS INTENDED

IPS/IDS engine would allow some packets first until the engine can figure out if this is bad/malicious traffic or not.

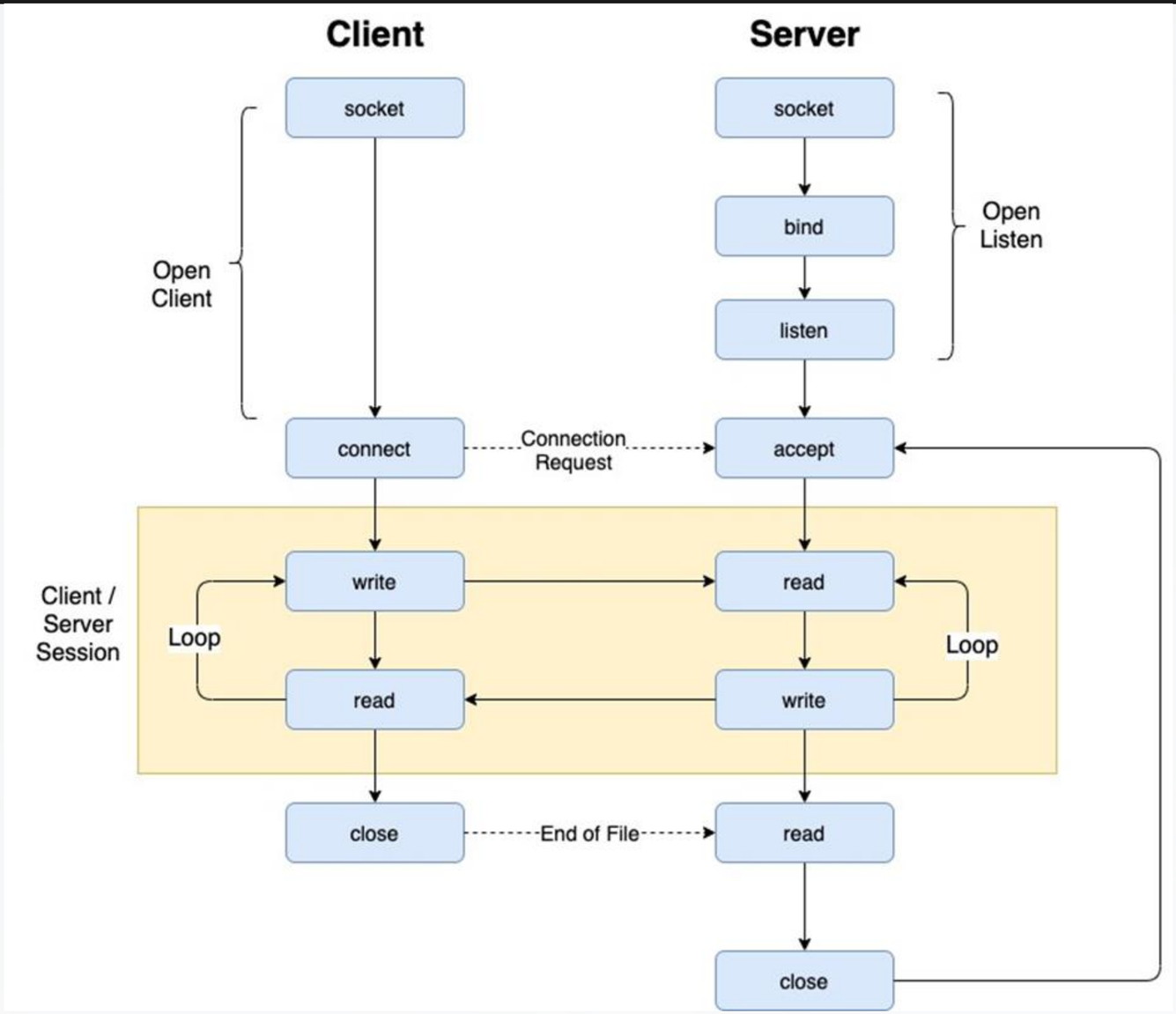
It explains why port scans showed the ports were open, but trying to connect those services were failing.

Design Flaw

WHAT IF..?

...the app doesn't follow the common programming practice with socket programming?

BACK TO FUNDAMENTALS



*Source: https://www.oreilly.com/library/view/distributed-computing-in/9781787126992/02dd04be-0dbb-4732-8bc5-1961644e8875.xhtml

WHAT IF..?

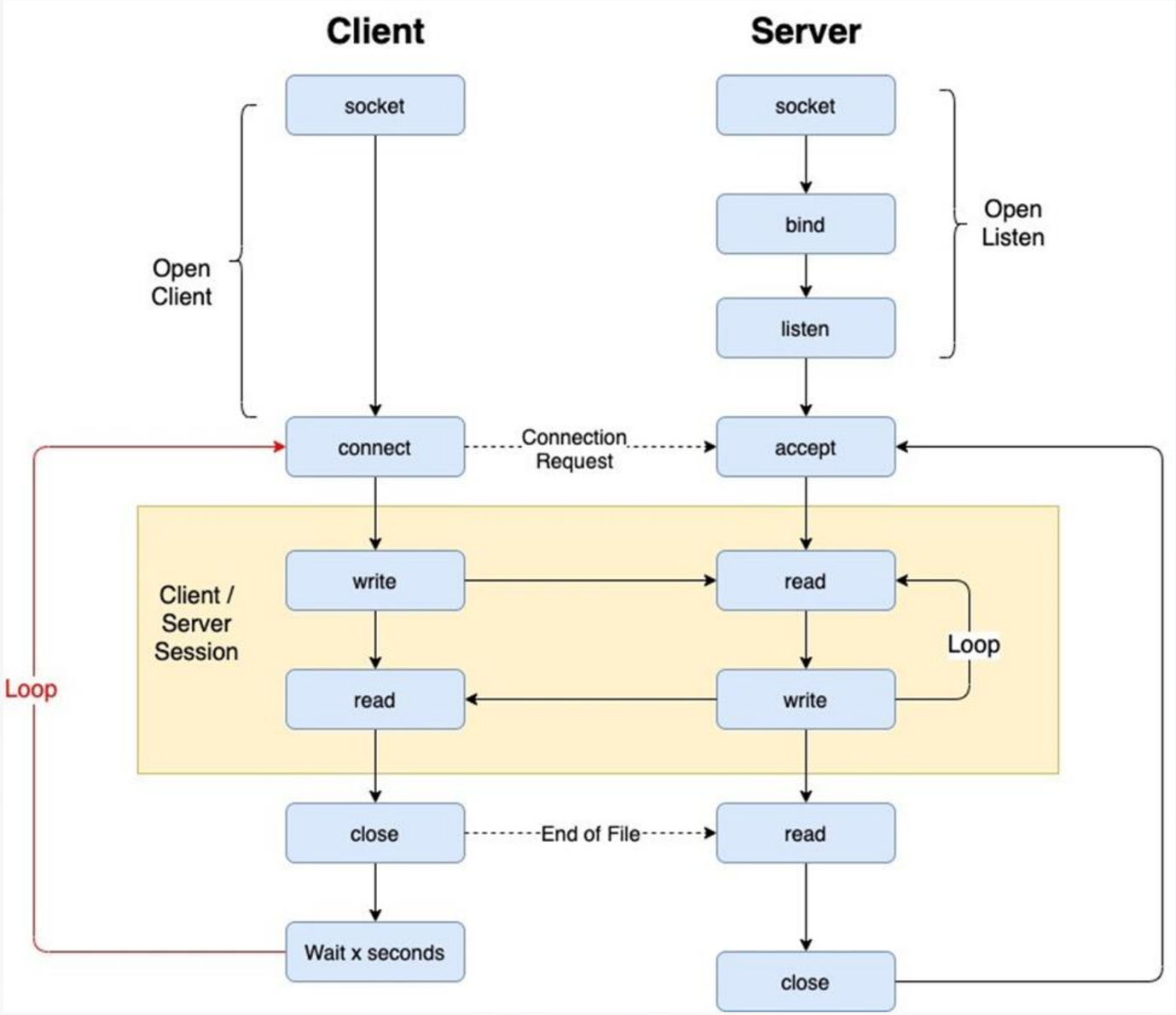
...the app doesn't follow the common programming practice with socket programming?

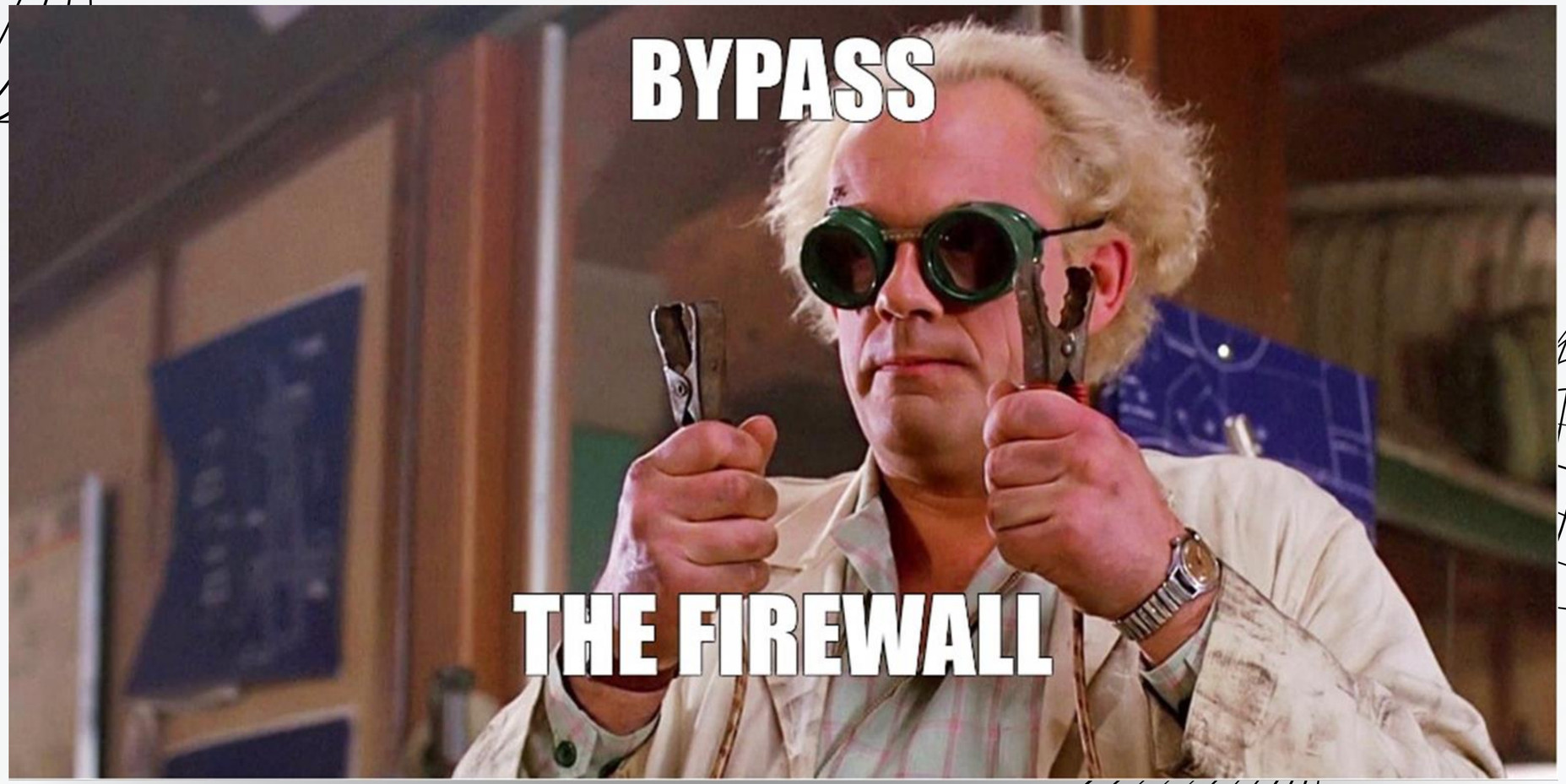
What if the client just connects, write, read, close the socket and just keep repeating this

Client closes the connection before IDS/IPS engine detects your application and blocks the connection

Every time you can send about 1-1.5 KB (ex.)

WHAT IF..?


Client
Server
socket
socket
bind
listen
Open
Client
Open
Listen
connect
Connection
Request
accept
Client /
Server
Session
write
read
read
write
Loop
Loop
close
End of File
read
Wait x seconds
close



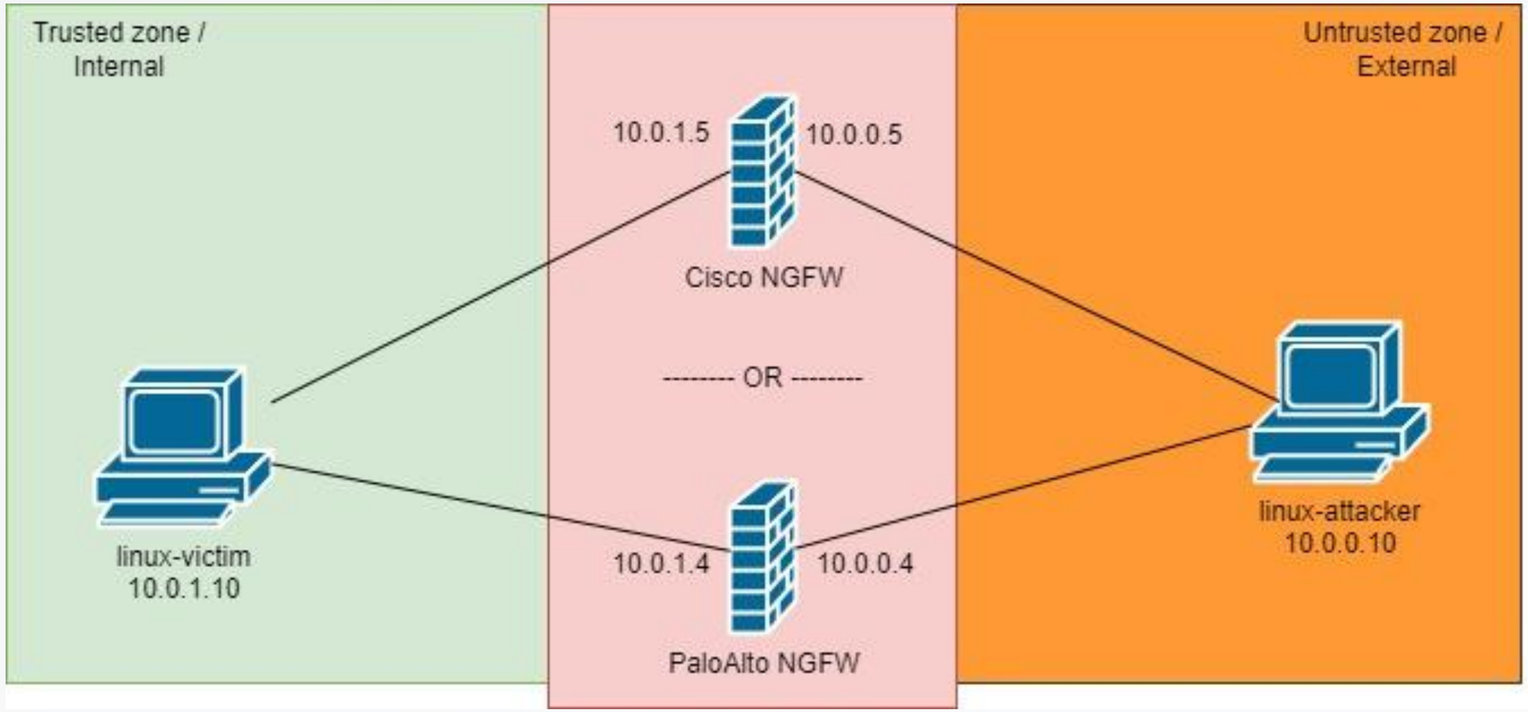
* Source: https://www.meme-arsenal.com/create/meme/3434715

DEMO SCENARIO

Example: A phishing attack with a malicious payload

Firewall rules:

- Only allow web browsing and/or MS Teams (TCP 80, 443)
- Default - Block everything else



LET'S BYPASS

Fragtunnel.py

- A TCP tunnel tool written in Python
- Not a proxy, not a socks proxy and not your regular tunnel
- You can tunnel your application's traffic to the target server, bypassing NGFW's along the way
- Data received from application get sliced into smaller fragments
- Each fragment gets sent one by one, each in different TCP session
- Data coming out from tunnel gets merged to make original data
- Original data gets sent to the final target
- Has current shortcomings, definitely needs improvement
  - Tunnel traffic is not threaded therefore, overall slow speed
  - Support for SSL/TLS
  - Tested with only limited set of tools/apps yet

DEMO

Live demo bypassing NGFWs from well known vendors

SOME NOTES

Firewalls and their Layer 7 application policy rules are not the only defense we have

Vendors very well knows this behavior

SUGGESTIONS

It is suggested by vendors that we should not only rely on Layer 7 (application level) policies. Instead, we should block anything unwanted on Layer 3 & 4 if possible, then have rules on Layer 7.

If possible, set and use more granular Layer 7 policies

- Allow-list an application, if possible, allow domains it should talk to
- Allow-listed server IPs that application should talk to (Layer 3)
- Protocols that application should use (Layer 4-7)

For blue teams

- Heavily repeated TCP handshakes from same source to same destination with same port in use may be an indicator of compromise and possible exfiltration

FUN FACTS

Probably I wasn't the only one thought about this, was I?

Initially I couldn't find anything, so started my research, put time and effort and built some PoC.

Accidentally, last year I found a tool and some presentations already discussed on this topic.

Of course, there were other researchers reported this way back, for example:

- Network Application Firewalls: Exploits and Defenses, Defcon 2011, Brad Woodberg
- Bypassing Next-Gen Firewall Rules, Nolasec 2012, Dave Laselle
- Sinking the Next Generation Firewall, Derbycon 2016, Russel Butturini
- Chunky Cookies: Smashing Application Aware Defenses, BSides Nashville 2017, Russel Butturini
- Fireaway by Russel Butturini (different PoC tool leveraging same idea)

FUN FACTS

The issue still exists even though it was reported first ~12 years ago.

QA

THANK YOU

Ali Efe

LinkedIn
Twitter: 0xaefe
Github: https://github.com/efeali