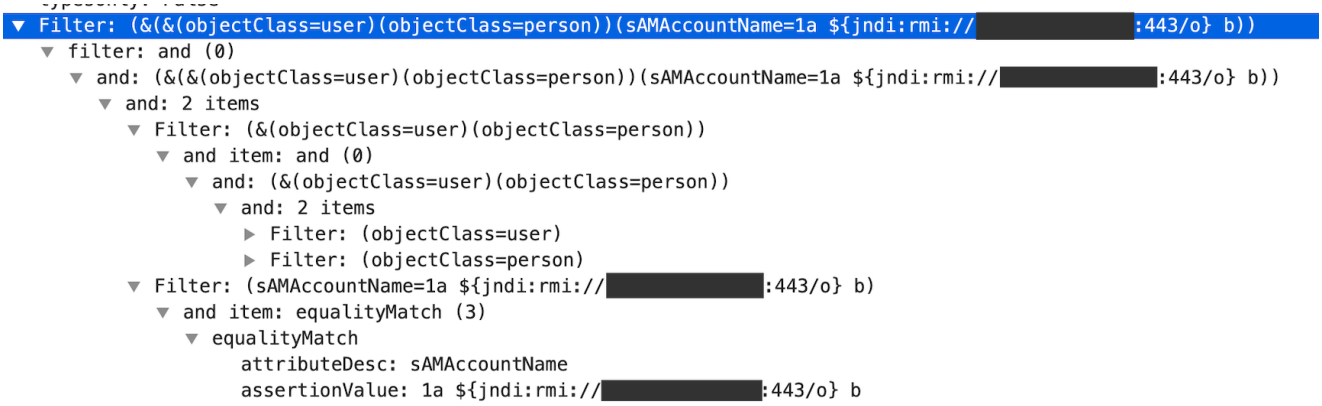
Log4Shell: Reconnaissance and post exploitation network detection

research.nccgroup.com/2021/12/12/log4shell-reconnaissance-and-post-exploitation-network-detection/

December 12, 2021



RIFT: Research and Intelligence Fusion Team December 12, 2021December 29, 2021 9 Minutes

Note: This blogpost will be live-updated with new information. NCC Group's RIFT is intending to publish PCAPs of different exploitation methods in the near future – last updated December 15th at 17:30 UTC

tl;dr

In the wake of the `CVE-2021-44228`, `CVE-2021-45046` and `CVE-2021-44832` (a.k.a. Log4Shell) vulnerability publication, NCC Group's RIFT immediately started investigating the vulnerability in order to improve detection and response capabilities mitigating the threat.

This blog post is focused on detection and threat hunting, although attack surface scanning and identification are also quintessential parts of a holistic response. Multiple references for prevention and mitigation can be found included at the end of this post.

This blogpost provides Suricata network detection rules that can be used not only to detect exploitation attempts, but also indications of successful exploitation. In addition, a list of indicators of compromise (IOC's) are provided. These IOC's have been observed listening for incoming connections and are thus a useful for threat hunting.

Update Wednesday December 29th, 09:00 UTC

CVE-2021-44832

A further vulnerability was disclosed on December 28th and is tracked under CVE-2021-44832. NCC Group assesses this vulnerability to be lower priority than the original due to the requirement of pre-existing privileged access to underlying hosts in order to exploit.

From the disclosure:

"*where an attacker with permission to modify the logging configuration file can construct a malicious configuration using a JDBC Appender with a data source referencing a JNDI URI which can execute remote code.*"

This vulnerability may be used as a persistence technique but is unlikely to be used as an initial entry mechanism due to the need to modify configuration.

Update Friday December 24th, 14:50 UTC

Log4Shell PCAPS and Network Coverage

Since the publication of the Log4Shell exploit there have been a lot of developments surrounding the Log4j CVE, leading to several new versions of the package to fix the workarounds that people found for the mitigations. During this time, there were also many people focusing their efforts on finding evasive methods to bypass mitigations put in place that block exploitation by monitoring for the exploitation string.

Because of the variety of the evasive methods, and the different protocols that can be used to exploit the vulnerability, we have created pcaps and an overview to assist security engineers in their endeavours to check their current detection coverage.

Setup

RIFT has used an environment to test different scenarios with the purpose of automatically creating pcaps and testing network coverage for the Remote Code Execution (RCE) vectors of Log4Shell using `LDAP` and `RMI`.

We tested different vectors that attackers could use in real-world scenarios, focusing on the HTTP protocol as this has been observed being used in the wild. Please keep in mind that HTTP is by no means the only protocol attackers can use to trigger the vulnerability in applications using a vulnerable version of Log4j. Any string that is logged by a vulnerable Log4j is subject to exploitation. We have also seen different evasion techniques, so these have also been tested for coverage.

We want to emphasize that we already observed attackers using encoded variants of the available protocols (HTTP Basic Authorization) and that there are plentiful other encoding methods that might still be logged decoded by the application using a vulnerable Log4j package.

We have used the following tools for testing the exploitation:

- JNDIExploit for LDAP github repo down
- JNDI-Exploit-Kit for LDAP and RMI

For web applications that is vulnerable to log4shell we have used:

- christophetd/log4shell-vulnerable-app
- docker-vuln-log4j-webapp

Log4Shell PCAPs and Coverage Tracking

The tables displayed below give an overview of the different evasion methods and their respective coverage. The PCAP filenames contain the `ev` string to mark the evasion ID.

EV	Payload
1	`${${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://` `${${::-j}${::-n}${::-d}${::-i}:${::-r}${::-m}${::-i}://`
2	`${${lower:jndi}:${lower:ldap}://` `${${::-j}ndi:rmi://`
3	`${${lower:${lower:jndi}}:${lower:ldap}://` `${${lower:jndi}:${lower:rmi}://`
4	`${${lower:j}${lower:n}${lower:d}i:${lower:ldap}://` `${${lower:${lower:jndi}}:${lower:rmi}://`
5	`${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:l}d${lower:a}p://` `${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}://`
6	`${j${env:DOESNOTEXIST:-}ndi:ldap://` `${j${env:DOESNOTEXIST:-}ndi:rmi://`
7	`${${: : : : ::: :: :: : :::-j}ndi:ldap://` `${${: : : : ::: :: :: : :::-j}ndi:rmi://`
8	`${${:::::::::::::-j}ndi:ldap://` `${${:::::::::::::-j}ndi:rmi://`

PCAPS etc. can be found here https://github.com/fox-it/log4shell-pcaps

Signatures

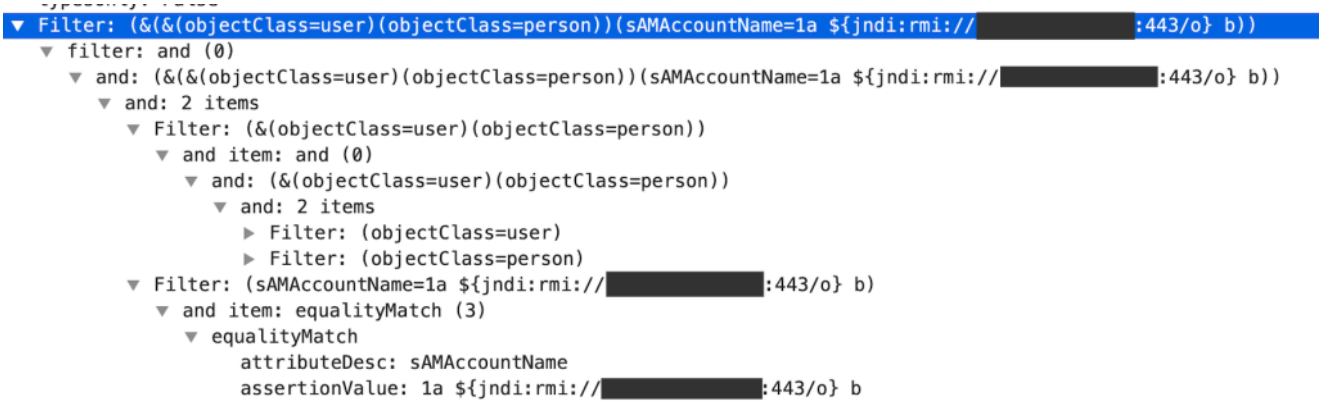
Our Log4Shell Suricata signatures can be found here: log4shell-suricata.rules

We have found that our signatures for outgoing `LDAP` and `RMI` packets are the best indicators (sids 21003738 and 21003739) of detecting a successful Log4Shell detonation. This also covers the situation where the malicious JNDI string is not always detected, for example due to TLS, but the IDS still monitors outgoing traffic.

Furthermore, the exploit chain itself might not always succeed, for example, due to Java versions or hardening of the system and or network. However, when these signatures trigger, a vulnerable Log4j version performed the callback and should be further investigated to determine which application caused it.

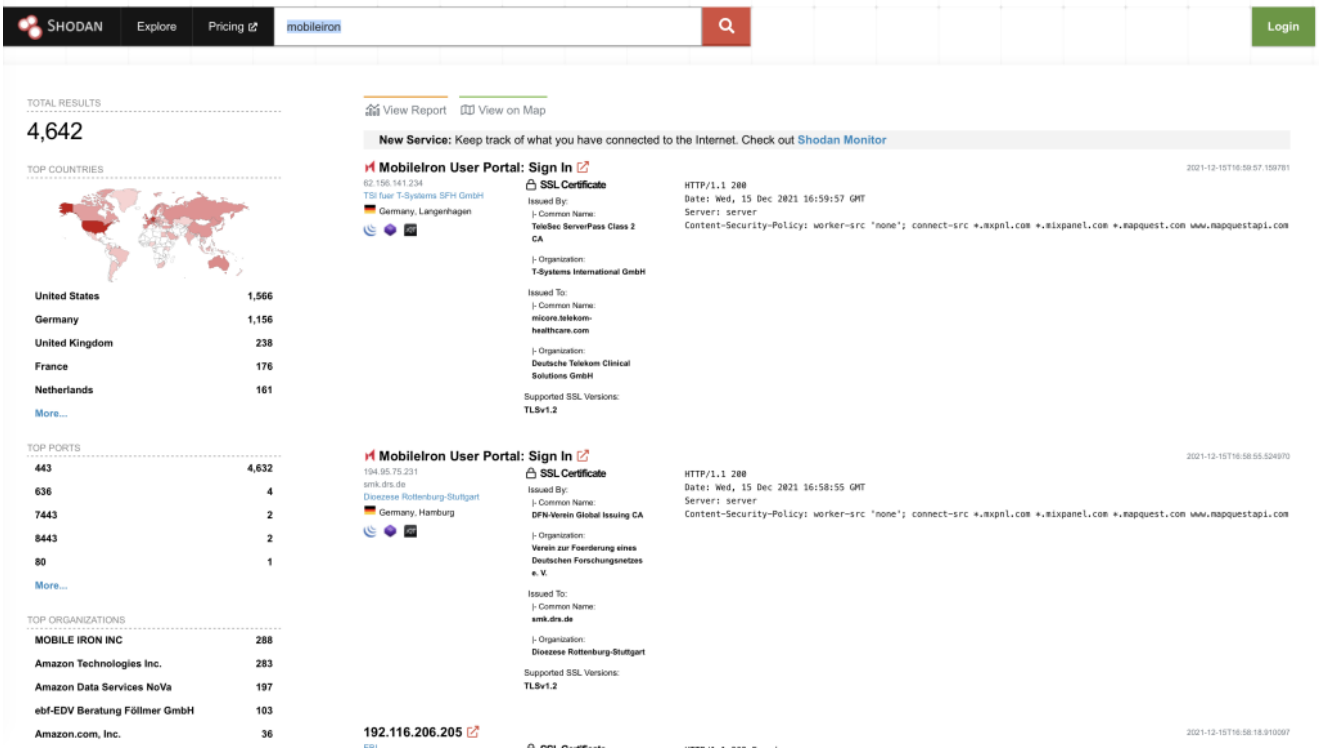
Update Wednesday December 15th, 17:30 UTC

We have seen 5 instances in our client base of active exploitation of Mobile Iron during the course of yesterday and today.



Our working hypothesis is that this is a derivative of the details shared yesterday – https://github.com/rwincey/CVE-2021-44228-Log4j-Payloads/blob/main/MobileIron.

The scale of the exposure globally appears significant



We recommend all Mobile Iron users updated immediately.

Ivanti informed us that communication was sent over the weekend to MobileIron Core customers. Ivanti has provided mitigation steps of the exploit listed below on their Knowledge Base. Both NCC Group and Ivanti recommend all customers immediately apply the mitigation within to ensure their environment is protected.

Update Tuesday December 14th, 13:00 UTC

Log4j-finder: finding vulnerable versions of Log4j on your systems

RIFT has published a Python 3 script that can be run on endpoints to check for the presence of vulnerable versions of Log4j. The script requires no dependencies and supports recursively checking the filesystem and inside JAR files to see if they contain a vulnerable version of Log4j. This script can be of great value in determining which systems are vulnerable, and where this vulnerability stems from. The script will be kept up to date with ongoing developments.

It is strongly recommended to run host based scans for vulnerable Log4j versions. Whereas network-based scans attempt to identify vulnerable Log4j versions by attacking common entry points, a host-based scan can find Log4j in unexpected or previously unknown places.

The script can be found on GitHub: https://github.com/fox-it/log4j-finder

JNDI ExploitKit exposes larger attack surface

As shown by the release of an update JNDI ExploitKIT it is possible to reach remote code execution through serialized payloads instead of referencing a Java `.class` object in LDAP and subsequently serving that to the vulnerable system. While `TrustURLCodebase` defaults to `false` in newer Java versions (6u211, 7u201, 8u191, and 11.0.1) and therefore prevents the LDAP reference vector,depending on the loaded libraries in the vulnerable application it is possible to execute code through Java serialization via both rmi and ldap.

Beware: Centralized logging can result in indirect compromise

This is also highly relevant for organisations using a form of centralised logging. Centralised logging can be used to collect and parse the received logs from the different services and applications running in the environment. We have identified cases where a Kibana server was not exposed to the Internet but because it received logs from several appliances it still got hit by the Log4Shell RCE and started to retrieve Java objects via LDAP.

We were unable to determine if this was due to Logstash being used in the background for parsing the received logs, but this stipulates the importance of checking systems configured with centralised logging solutions for vulnerable versions of Log4j, and not rely on the

protection of newer JDK versions that has `com.sun.jndi.ldap.object.trustURLCodebase` `com.sun.jndi.rmi.object.trustURLCodebase` set to `false` by default.

A warning concerning possible post-exploitation

It is therefore advised to apply the patches provided by Microsoft in the November 2021 security updateAlthough largely eclipsed by Log4Shell, last weekend also saw the emergence of details concerning two vulnerabilities (`CVE-2021-42287` and `CVE-2021-42278`) that reside in the Active Directory component of Microsoft Windows Server editions. Due to the nature of these vulnerabilities, an attackers could escalate their privileges in a relatively easy manner as these vulnerabilities have already been weaponised.

It is therefore advised to apply the patches provided by Microsoft in the November 2021 security updates to every domain controller that is residing in the network as it is a possible form of post-exploitation after Log4Shell were to be successfully exploited.

Background

Since Log4J is used by many solutions there are significant challenges in finding vulnerable systems and any potential compromise resulting from exploitation of the vulnerability. JNDI (Java Naming and Directory Interface™) was designed to allow distributed applications to look up services in a resource-independent manner, and this is exactly where the bug resulting in exploitation resides. The nature of JNDI allows for defense-evading exploitation attempts that are harder to detect through signatures. An additional problem is the tremendous amount of scanning activity that is currently ongoing. Because of this, investigating every single exploitation attempt is in most situations unfeasible. This means that **distinguishing scanning attempts from actual successful exploitation is crucial.**

In order to provide detection coverage for `CVE-2021-44228` and `CVE-2021-45046` , NCC Group's RIFT first created a ruleset that covers as many ways as possible of attempted exploitation of the vulnerability. This initial coverage allowed the collection of Threat Intelligence for further investigation. Most adversaries appear to use a different IP to scan for the vulnerability than they do for listening for incoming victim machines. **IOC's for listening IP's / domains are more valuable than those of scanning IP's.** After all a connection from an environment to a known listening IP might indicate a successful compromise, whereas a connection to a scanning IP might merely mean that it has been scanned.

After establishing this initial coverage, our focus shifted to detecting successful exploitation in real time. This can be done by monitoring for rogue JRMI or LDAP requests to external servers. Preferably, this sort of behavior is detected in a port-agnostic way as attackers may choose arbitrary ports to listen on. Moreover, currently a full RCE chain requires the victim machine to retrieve a Java class file from a remote server (caveat: unless exfiltrating sensitive environment variables). For hunting purposes we are able to hunt for inbound Java

classes. However, if coverage exists for incoming attacks we are also able to alert on an inbound Java class in a short period of time after an exploitation attempt. The combination of inbound exploitation attempt and inbound Java class is a high confidence IOC that a successful connection has occurred.

This blogpost will continue twofold: we will first provide a set of suricata rules that can be used for:

1. Detecting incoming exploitation attempts;
2. Alerting on higher confidence indicators that successful exploitation has occurred;
3. Generating alerts that can be used for hunting

After providing these detection rules, a list of IOC's is provided.

Detection Rules

Some of these rules are redundant, as they've been written in rapid succession.

```
# Detects Log4j exploitation attempts
```

```
alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Apache Log4J RCE Request Observed (CVE-2021-44228)"; flow:established, to_server; content:"${jndi:ldap://"; fast_pattern:only; flowbits:set, fox.apachelog4j.rce; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:web-application-attack; priority:3; reference:url, http://www.lunasec.io/docs/blog/log4j-zero-day/; metadata:CVE 2021-44228; metadata:created_at 2021-12-10; metadata:ids suricata; sid:21003726; rev:1;)
```

```
alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Apache Log4J RCE Request Observed (CVE-2021-44228)"; flow:established, to_server; content:"${jndi:"; fast_pattern; pcre:"/\$\{jndi\:(rmi|ldaps|dns)\:/"; flowbits:set, fox.apachelog4j.rce; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:web-application-attack; priority:3; reference:url, http://www.lunasec.io/docs/blog/log4j-zero-day/; metadata:CVE 2021-44228; metadata:created_at 2021-12-10; metadata:ids suricata; sid:21003728; rev:1;)
```

```
alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Defense-Evasive Apache Log4J RCE Request Observed (CVE-2021-44228)"; flow:established, to_server; content:"${jndi:"; fast_pattern; content:!"ldap://"; flowbits:set, fox.apachelog4j.rce; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:web-application-attack; priority:3; reference:url, http://www.lunasec.io/docs/blog/log4j-zero-day/; reference:url, twitter.com/stereotype32/status/1469313856229228544; metadata:CVE 2021-44228; metadata:created_at 2021-12-10; metadata:ids suricata; sid:21003730; rev:1;)
```

alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Defense-Evasive Apache Log4J RCE Request Observed (URL encoded bracket) (CVE-2021-44228)"; flow:established, to_server; content:"%7bjndi:"; nocase; fast_pattern; flowbits:set, fox.apachelog4j.rce; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:web-application-attack; priority:3; reference:url, http://www.lunasec.io/docs/blog/log4j-zero-day/; reference:url, https://twitter.com/testanull/status/1469549425521348609; metadata:CVE 2021-44228; metadata:created_at 2021-12-11; metadata:ids suricata; sid:21003731; rev:1;)

alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Apache Log4j Exploit Attempt in HTTP Header"; flow:established, to_server; content:"${"; http_header; fast_pattern; content:"}"; http_header; distance:0; flowbits:set, fox.apachelog4j.rce.loose; classtype:web-application-attack; priority:3; threshold:type limit, track by_dst, count 1, seconds 3600; reference:url, http://www.lunasec.io/docs/blog/log4j-zero-day/; reference:url, https://twitter.com/testanull/status/1469549425521348609; metadata:CVE 2021-44228; metadata:created_at 2021-12-11; metadata:ids suricata; sid:21003732; rev:1;)

alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Apache Log4j Exploit Attempt in URI"; flow:established,to_server; content:"${"; http_uri; fast_pattern; content:"}"; http_uri; distance:0; flowbits:set, fox.apachelog4j.rce.loose; classtype:web-application-attack; priority:3; threshold:type limit, track by_dst, count 1, seconds 3600; reference:url, http://www.lunasec.io/docs/blog/log4j-zero-day/; reference:url, https://twitter.com/testanull/status/1469549425521348609; metadata:CVE 2021-44228; metadata:created_at 2021-12-11; metadata:ids suricata; sid:21003733; rev:1;)

Better and stricter rules, also detects evasion techniques

alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Apache Log4j Exploit Attempt in HTTP Header (strict)"; flow:established,to_server; content:"${"; http_header; fast_pattern; content:"}"; http_header; distance:0; pcre:/(\$\{\w+:.*\}|jndi)/Hi; xbits:set, fox.log4shell.attempt, track ip_dst, expire 1; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:web-application-attack; reference:url,www.lunasec.io/docs/blog/log4j-zero-day/; reference:url,https://twitter.com/testanull/status/1469549425521348609; metadata:CVE 2021-44228; metadata:created_at 2021-12-11; metadata:ids suricata; priority:3; sid:21003734; rev:1;)

```
alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Apache Log4j Exploit Attempt in URI (strict)"; flow:established, to_server; content:"${"; http_uri; fast_pattern; content:"}"; http_uri; distance:0; pcre:/(\$\{\w+:.*\}|jndi)/Ui; xbits:set, fox.log4shell.attempt, track ip_dst, expire 1; classtype:web-application-attack; threshold:type limit, track by_dst, count 1, seconds 3600; reference:url,www.lunasec.io/docs/blog/log4j-zero-day/; reference:url,https://twitter.com/testanull/status/1469549425521348609; metadata:CVE 2021-44228; metadata:created_at 2021-12-11; metadata:ids suricata; priority:3; sid:21003735; rev:1;)
```

```
alert http any any -> $HOME_NET any (msg:"FOX-SRT – Exploit – Possible Apache Log4j Exploit Attempt in Client Body (strict)"; flow:to_server; content:"${"; http_client_body; fast_pattern; content:"}"; http_client_body; distance:0; pcre:/(\$\{\w+:.*\}|jndi)/Pi; flowbits:set, fox.apachelog4j.rce.strict; xbits:set,fox.log4shell.attempt,track ip_dst,expire 1; classtype:web-application-attack; threshold:type limit, track by_dst, count 1, seconds 3600; reference:url,www.lunasec.io/docs/blog/log4j-zero-day/; reference:url,https://twitter.com/testanull/status/1469549425521348609; metadata:CVE 2021-44228; metadata:created_at 2021-12-12; metadata:ids suricata; priority:3; sid:21003744; rev:1;)
```

view raw log4shell-exploitation-attempts.rules hosted with ❤ by GitHub

Detecting outbound connections to probing services

Connections to outbound probing services could indicate a system in your network has been scanned and subsequently connected back to a listening service. This could indicate that a system in your network is/was vulnerable and has been scanned.

```
# Possible successful interactsh probe
```

```
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"FOX-SRT – Webattack – Possible successful InteractSh probe observed"; flow:established, to_client; content:"200"; http_stat_code; content:"<html><head></head><body>"; http_server_body; fast_pattern; pcre:"/[a-z0-9]{30,36}<\/body><\/html>/QR"; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:misc-attack; reference:url, github.com/projectdiscovery/interactsh; metadata:created_at 2021-12-05; metadata:ids suricata; priority:2; sid:21003712; rev:1;)
```

```
alert dns $HOME_NET any -> any 53 (msg:"FOX-SRT – Suspicious – DNS query for interactsh.com server observed"; flow:stateless; dns_query; content:".interactsh.com"; fast_pattern; pcre:"/[a-z0-9]{30,36}\.interactsh\.com/"; threshold:type limit, track by_src, count 1, seconds 3600; reference:url, github.com/projectdiscovery/interactsh; classtype:bad-unknown; metadata:created_at 2021-12-05; metadata:ids suricata; priority:2; sid:21003713; rev:1;)
```

```
# Detecting DNS queries for dnslog[.]cn
```

```
alert dns any any -> any 53 (msg:"FOX-SRT – Suspicious – dnslog.cn DNS Query Observed"; flow:stateless; dns_query; content:"dnslog.cn"; fast_pattern:only; threshold:type limit, track by_src, count 1, seconds 3600; classtype:bad-unknown; metadata:created_at 2021-12-10; metadata:ids suricata; priority:2; sid:21003729; rev:1;)

# Connections to requestbin.net

alert dns $HOME_NET any -> any 53 (msg:"FOX-SRT – Suspicious – requestbin.net DNS Query Observed"; flow:stateless; dns_query; content:"requestbin.net"; fast_pattern:only; threshold:type limit, track by_src, count 1, seconds 3600; classtype:bad-unknown; metadata:created_at 2021-11-23; metadata:ids suricata; sid:21003685; rev:1;)

alert tls $HOME_NET any -> $EXTERNAL_NET 443 (msg:"FOX-SRT – Suspicious – requestbin.net in SNI Observed"; flow:established, to_server; tls_sni; content:"requestbin.net"; fast_pattern:only; threshold:type limit, track by_src, count 1, seconds 3600; classtype:bad-unknown; metadata:created_at 2021-11-23; metadata:ids suricata; sid:21003686; rev:1;)
```

view raw log4shell-probes.rules hosted with ❤ by GitHub

Detecting possible successful exploitation

Outbound LDAP(S) / RMI connections are highly uncommon but can be caused by successful exploitation. Inbound Java can be suspicious, especially if it is shortly after an exploitation attempt.

```
# Detects possible successful exploitation of Log4j

# JNDI LDAP/RMI Request to External

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"FOX-SRT – Exploit – Possible Rogue JNDI LDAP Bind to External Observed (CVE-2021-44228)"; flow:established, to_server; dsize:14; content:"|02 01 03 04 00 80 00|"; offset:7; isdataat:!1, relative; threshold:type limit, track by_src, count 1, seconds 3600; classtype:bad-unknown; priority:1; metadata:created_at 2021-12-11; sid:21003738; rev:2;)

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"FOX-SRT – Exploit – Possible Rogue JRMI Request to External Observed (CVE-2021-44228)"; flow:established, to_server; content:"JRMI"; depth:4; threshold:type limit, track by_src, count 1, seconds 3600; classtype:bad-unknown; priority:1; reference:url, https://docs.oracle.com/javase/9/docs/specs/rmi/protocol.html; metadata:created_at 2021-12-11; sid:21003739; rev:1;)

# Detecting inbound java shortly after exploitation attempt
```

```
alert tcp any any -> $HOME_NET any (msg: "FOX-SRT – Exploit – Java class inbound after CVE-2021-44228 exploit attempt (xbit)"; flow:established, to_client; content: "|CA FE BA BE 00 00 00|"; depth:40; fast_pattern; xbits:isset, fox.log4shell.attempt, track ip_dst; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:successful-user; priority:1; metadata:ids suricata; metadata:created_at 2021-12-12; sid:21003741; rev:1;)
```

view raw log4shell-success.rules hosted with ❤ by GitHub

Hunting rules (can yield false positives)

Wget and cURL to external hosts was observed to be used by an actor for post-exploitation. As cURL and Wget are also used legitimately, these rules should be used for hunting purposes. Also note that attackers can easily change the User-Agent but we have not seen that in the wild yet. Outgoing connections after Log4j exploitation attempts can be tracked to be later hunted on although this rule can generate false positives if victim machine makes outgoing connections regularly. Lastly, detecting inbound compiled Java classes can also be used for hunting.

```
# Outgoing connection after Log4j Exploit Attempt (uses xbit from sid: 21003734) – requires `stream.inline=yes` setting in suricata.yaml for this to work
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"FOX-SRT – Suspicious – Possible outgoing connection after Log4j Exploit Attempt"; flow:established, to_server; xbits:isset, fox.log4shell.attempt, track ip_src; stream_size:client, =, 1; stream_size:server, =, 1; threshold:type limit, track by_dst, count 1, seconds 3600; classtype:bad-unknown; metadata:ids suricata; metadata:created_at 2021-12-12; priority:3; sid:21003740; rev:1;)
```

```
# Detects inbound Java class
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "FOX-SRT – Suspicious – Java class inbound"; flow:established, to_client; content: "|CA FE BA BE 00 00 00|"; depth:20; fast_pattern; threshold:type limit, track by_dst, count 1, seconds 43200; metadata:ids suricata; metadata:created_at 2021-12-12; classtype:bad-unknown; priority:3; sid:21003742; rev:2;)
```

view raw log4shell-hunting.rules hosted with ❤ by GitHub

Indicators of Compromise

This list contains **Domains and IP's that have been observed to listen for incoming connections.** Unfortunately, some adversaries scan and listen from the same IP, generating a lot of noise that can make threat hunting more difficult. Moreover, as security researchers are scanning the internet for the vulnerability as well, it could be possible that an IP or domain is listed here even though it is only listening for benign purposes.

```
# IP addresses and domains that have been observed in Log4j exploit
attempts
134[.]209[.]26[.]39
199[.]217[.]117[.]92
pwn[.]af
188[.]120[.]246[.]215
kryptoslogic-cve-2021-44228[.]com
nijat[.]space
45[.]33[.]47[.]240
31[.]6[.]19[.]41
205[.]185[.]115[.]217
log4j[.]kingudo[.]de
101[.]43[.]40[.]206
psc4fuel[.]com
185[.]162[.]251[.]208
137[.]184[.]61[.]190
162[.]33[.]177[.]73
34[.]125[.]76[.]237
162[.]255[.]202[.]246
5[.]22[.]208[.]77
45[.]155[.]205[.]233
165[.]22[.]213[.]147
172[.]111[.]48[.]30
133[.]130[.]120[.]176
213[.]156[.]18[.]247
m3[.]wtf
poc[.]brzozowski[.]io
206[.]188[.]196[.]219
185[.]250[.]148[.]157
132[.]226[.]170[.]154
flofire[.]de
45[.]130[.]229[.]168
c19s[.]net
194[.]195[.]118[.]221
awsdns-2[.]org
2[.]56[.]57[.]208
158[.]69[.]204[.]95
45[.]130[.]229[.]168
163[.]172[.]157[.]143
45[.]137[.]21[.]9
bingsearchlib[.]com
45[.]83[.]193[.]150
165[.]227[.]93[.]231
yourdns[.]zone[.]here
eg0[.]ru
dataastatistics[.]com
log4j-test[.]xyz
79[.]172[.]214[.]11
152[.]89[.]239[.]12
67[.]205[.]191[.]102
ds[.]Rce[.]ee
```

```
38[.]143[.]9[.]76
31[.]191[.]84[.]199
143[.]198[.]237[.]19

# (Ab)use of listener-as-a-service domains.
# These domains can be false positive heavy, especially if these services
are used legitimately within your network.
interactsh[.]com
interact[.]sh
burpcollaborator[.]net
requestbin[.]net
dnslog[.]cn
canarytokens[.]com

# This IP is both a listener and a scanner at the same time. Threat
hunting for this IOC thus requires additional steps.
45[.]155[.]205[.]233
194[.]151[.]29[.]154
158[.]69[.]204[.]95
47[.]254[.]127[.]78
```

view raw log4shell-iocs.md hosted with ❤ by GitHub

References

General references

- Fox-IT / NCC Group actively participates in a continuously updated reddit thread: https://www.reddit.com/r/blueteamsec/comments/rd38z9/log4j_0day_being_exploited/
- https://nvd.nist.gov/vuln/detail/CVE-2021-44228

Mitigation:

https://github.com/OllieJC/aws-log4j-mitigations

Attack surface:

Known vulnerable services / products which use log4j:

- https://github.com/YfryTchsGD/Log4jAttackSurface
- https://mvnrepository.com/artifact/log4j/log4j/usages

Hashes of vulnerable products:

https://gist.github.com/olliencc/8be866ae94b6bee107e3755fd1e9bf0d

Published by RIFT: Research and Intelligence Fusion Team

RIFT leverages our strategic analysis, data science, and threat hunting capabilities to create actionable threat intelligence, ranging from IoCs and detection capabilities to strategic reports on tomorrow's threat landscape. Cyber security is an arms race where both attackers and defenders continually update and improve their tools and ways of working. To ensure that our managed services remain effective against the latest threats, NCC Group operates a Global Fusion Center with Fox-IT at its core. This multidisciplinary team converts our leading cyber threat intelligence into powerful detection strategies. View all posts by RIFT: Research and Intelligence Fusion Team

Published December 12, 2021December 29, 2021

Post navigation

Previous Post Announcing NCC Group's Cryptopals Guided Tour!
Next Post log4j-jndi-be-gone: A simple mitigation for CVE-2021-44228