sonatype

Our Top 5 Vulnerable Open Source Components

Overview

Sonatype operates at the forefront of open source software contribution and maintenance. With over a decade of active involvement in projects such as Maven Central, we continually work to be a leader in the open source community. Our pivotal role in managing the Central Repository, servicing billions of component releases to developers globally, demonstrates our commitment to supporting today's software-driven world.

Open source components form the foundation of modern software development. Our research findings, outlined in the 2022 State of the Software Supply Chain report, reveal remarkable growth in the supply of open source component releases. Download volume of components across the four major ecosystems projected to surpass 3 trillion downloads overall last year. But with this boom in open source component usage, there has been a **742%** average annual increase in software supply chain attacks over the past three years.

In terms of open source components, software dependencies can equal open source vulnerabilities. In 2022, the number of open source dependencies downloaded and integrated into software grew by an estimated average of 33% across all monitored ecosystems (Maven, PyPI, NuGet, and npm).

Our research also surfaced notable statistics such as:

- **6 out of every 7 project vulnerabilities come from transitive dependencies**
- **1.2 billion vulnerable dependencies are downloaded every month**
- **96% of known vulnerable open source components already have a fix available**

To empower your organization and developers in leveraging open source components to drive innovation while mitigating risk, you need precision and complete intelligence. Tools lacking visibility and accuracy fall short in today's fast-paced software development environment. Inaccurate or incomplete data exposes organizations to vulnerabilities, licensing challenges, and quality issues, leading to increased costs and diminished innovation.

At Sonatype, we've harnessed the power of precise identification to enable organizations to error-proof their software supply chains. By leveraging our 10+ years of experience and conducting 118,000 hours of data security research, we provide development and security teams with high-quality scan results for critical decision-making regarding version selection and remediation. Our customers have expressed great interest in our data capabilities, urging us to provide insights into the most popular open source components and conduct in-depth security analyses of the vulnerabilities affecting them.

In response to these demands, Sonatype has successfully identified five popular open source components used worldwide and their associated vulnerabilities.

In this report, we delve into the most vulnerable version for each, based on a CVSS rating of 6.5 or higher.

We take immense pride in the accuracy and reliability of our data. As you read this report, we encourage you to engage with us, ask questions, and provide feedback. Sonatype is committed to driving secure and innovative software supply chains, and we value your partnership in achieving this shared goal.

Our Top 5 Popular Open Source Components and Their Vulnerabilities

The VM-escape paradox

Name	Vulnerability	Description
vm2	CVE-2023-30547	Critical sandbox escape vulnerabilities
redis-py	CVE-2023-28859	A bug in ChatGPT
Chromium	CVE-2022-2294	Heap buffer overflow in WebRTC
WildFly Elytron	CVE-2020-10714	A session fixation attack
Apache Chainsaw	CVE-2022-23307	A vulnerability in an end-of-life version of Log4j

Name of component: **vm2**

Name of vulnerability: CVE-2023-30547

Associated CVEs: CVE-2023-29199, CVE-2023-29017, CVE-2022-36067

Type of vulnerability: Sandbox escape

Versions affected: <3.9.17

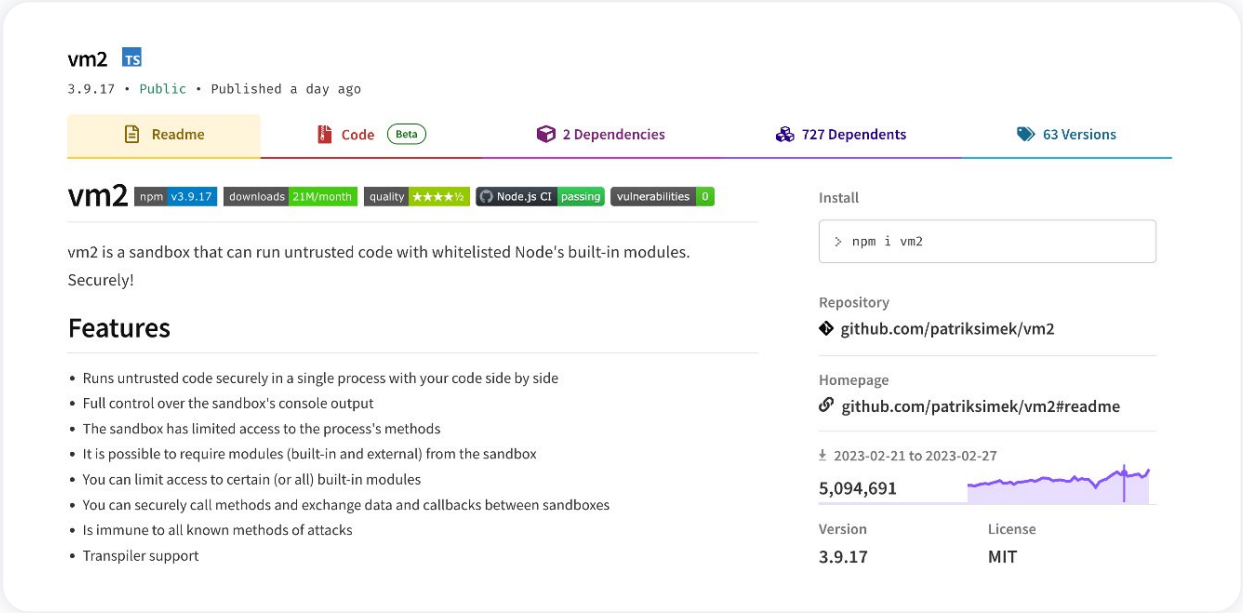
Criticality:

- CVSS 3.0 Score: 10.0
- CVSS 3.0 Metrics: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

DESCRIPTION

The **vm2** npm package is no stranger to sandbox escape vulnerabilities that keep piling on. To attackers, they are a gift that keeps on giving. Sandbox escape vulnerabilities also lead to an interesting paradox as the whole point of a virtual machine (VM) is to serve as a sandbox — a "safe" zone insulated from the host system/environment. In this manner, any untrusted code run within a VM — for example, by a malicious attacker — does not risk affecting the applications and data present outside of the VM. That is one reason why the **vm2** sandbox escape vulnerabilities described here are deemed "critical" in severity.

By no means is **vm2** unheard of either. The popular npm component has been downloaded more than 315 million times (at the time of this writing) over the course of its lifetime and rakes in around 5 million weekly downloads.



ATTACK MECHANICS

The sandbox escape vulnerabilities in **vm2** are essentially identical at their core. Of the four CVEs mentioned below representing this flaw, each subsequent one resulted from an incomplete fix issued for a previous one — three being disclosed within the same month. The insufficient or incomplete error handling in the **vm2** project is what enables attackers to "break out" of the sandbox protections provided by the package.

What makes these CVEs pressing is proof-of-concept (PoC) exploits [1, 2] which became available on the internet almost around the same time as the vulnerability report was filed and often publicly on the **vm2** GitHub:

1. CVE-2023-30547, disclosed April 17, 2023, affects **vm2** versions prior to 3.9.17 and concerns the `handleException()` function improperly handling errors. Attackers can trick the error-handling mechanism to escape the sandbox and run arbitrary code in the context of the host.

2. CVE-2023-29199, disclosed April 11, 2023, is much the same, except it affects versions 3.9.15 and below. It was supposed to be resolved in 3.9.16, but obviously that's not the case considering CVE-2023-30547.

3. CVE-2023-29017, disclosed April 6, 2023, affects versions 3.9.15 and below. Instead of targeting `handleException()` though, the PoCs for this flaw target `Error.prepareStackTrace` to break out of the sandbox.

4. CVE-2022-36067, disclosed August 30, 2022, affects versions below (and not including) 3.9.11. It also concerns `prepareStackTrace`.

The PoC for the latest **CVE-2023-30547** is provided by SeungHyun Lee, aka Xion, of KAIST Hacking Lab, who is credited with reporting the vulnerability.

```
const {VM} = require("vm2");
const vm = new VM();

const code = `
err = {};
const handler = {
    getPrototypeOf(target) {
        (function stack() {
            new Error().stack;
            stack();
        })();
    }
};

const proxiedErr = new Proxy(err, handler);
try {
    throw proxiedErr;
} catch ({constructor: c}) {
    c.constructor('return process')().mainModule.require('child_process').execSync('touch pwned');
}
`

console.log(vm.run(code));
```

As evident, the exploit leverages insufficient error handling within the package. An attacker can effectively write a program that throws an error message and later catches the error to execute arbitrary code *outside* of the sandbox.

REMEDIATION RECOMMENDATION

At the time of writing, the latest **vm2** version **3.9.17** has resolved all these sandbox escape vulnerabilities — upgrade to that version or higher to fix. However, given the ongoing trend with **vm2**, it won't be surprising if even the latest fix can be bypassed by variations of the exploit.

Critical vulnerabilities in prominent projects like **vm2** further entice bug bounty hunters, ethical hackers, and threat actors to hunt for similar flaws in smaller projects with identical functionality. For example, a now-abandoned npm project `**safe-eval**` draws researchers reporting sandbox escape type flaws [1, 2, 3]. These contain similar PoCs used in vulnerability reports for **vm2**.

CONCLUSION

While virtual machines are designed to provide a secure sandbox environment, the presence of these critical vulnerabilities highlights the need for robust security practices. The sandbox escape vulnerabilities in the **vm2** npm package serve as compelling cases for implementing a comprehensive software composition analysis (SCA) product like Sonatype IQ Server, which powers multiple Sonatype solutions.

Sonatype IQ Server offers powerful automation capabilities, enabling organizations to proactively identify and address vulnerabilities in their software supply chain. By leveraging Sonatype IQ Server's advanced SCA features, such as component analysis and vulnerability scanning, developers can detect issues like the ones found in **vm2**, ensuring the integrity and security of their code.

The ability to automate scanning and analysis processes significantly reduces the risk of incomplete fixes and subsequent vulnerabilities, providing you with peace of mind. Furthermore, Sonatype IQ Server's comprehensive reports and actionable insights empower teams to prioritize and remediate vulnerabilities efficiently. By adopting Sonatype IQ Server, you can enhance security posture, streamline vulnerability management, and safeguard software assets against emerging threats, ensuring delivery of secure and reliable software to your customers.

ChatGPT spills out payment information due to Redis bug

Name of component: **redis-py**

Name of vulnerability: CVE-2023-28859

Associated CVEs: CVE-2023-28858

Type of vulnerability: Race Condition

Versions affected: Before 4.4.4, and 4.5.x before 4.5.4

Criticality:

- ▶ CVSS 3.0 Score: 6.5
- ▶ CVSS 3.0 Metrics: CVSS 3.0 Metrics: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

DESCRIPTION

In March 2023, several ChatGPT users were surprised to see their chat histories showing other people's chat queries. OpenAI later disclosed it had to temporarily take ChatGPT offline as an unpatched bug (or possible vulnerability — more on that below) in an open source component caused the data leak of some of its subscribers' payment-related information, along with users' chat queries. The library in question is called Redis — a popular in-memory data structure store often used for distributed caching and large-scale noSQL databases.

The bug, initially tracked by us as **sonatype-2023-1621**, was later assigned **CVE-2023-28858** and **CVE-2023-28859**. The bug is a race condition in Redis, an open source component available in the PyPI repository. Because the "bug" poses a security risk impacting a system's confidentiality and resource availability potentially opening doors for exploitation, it effectively becomes a vulnerability.

NOTE: *bug* vs *vulnerability*

As explained in this article, "a bug is when the system isn't behaving as it's supposed to, whereas a vulnerability is a bug that manifests itself as an opportunity for exploitation."

ATTACK MECHANICS

The vulnerability itself is quite straightforward, but it concerns a scenario that would typically occur only in extremely rare circumstances. In this case, OpenAI admitted to inadvertently introducing a server change that caused a spike in Redis request cancellations, thereby bumping up the probability of this race condition triggering a whole lot more than it normally would have.

That is why multiple users, as opposed to a singular random person here and there, had their chat queries leak into other users' chat histories. For about 1.2% of ChatGPT Plus subscribers, their

name, email address, payment address, and partial credit card data (last four digits, expiration date) were also leaked.

OpenAI states that it uses Redis to cache user information across its servers, so it doesn't need to query its database for every request. OpenAI further uses Redis Cluster to fairly distribute load across multiple Redis instances.

"We use the **redis-py** library to interface with Redis from our Python server, which runs with asyncio," said OpenAI. "The library maintains a shared pool of connections between the server and the cluster, and recycles a connection to be used for another request once done."

The Redis PyPI library uses asyncio to implement its cluster and client classes. But due to insufficient error handling in Redis for extremely rare conditions that can occur in large scale context-dependent applications like ChatGPT, unintended consequences may happen.

REMEDIATION RECOMMENDATION

Despite Redis releasing a fix in version 4.5.3 and some backports, sharp-witted testers were able to reproduce the flaw, deeming it unfixed. As such, a second identifier, **CVE-2023-28859** was assigned to track the flaw in insufficiently fixed versions (e.g. 4.5.3, 5.0.0b1, etc.).

12 redis/asyncio/cluster.py

```
@@ -1002,10 +1002,18 @@ async def execute_command(self, *args: Any, **kwargs: Any) -> Any:
1002 1002          await connection.send_packed_command(connection.pack_command(*args), False)
1003 1003
1004 1004          # Read response
     1005 +        return await asyncio.shield(
     1006 +            self._parse_and_release(connection, args[0], **kwargs)
     1007 +        )
     1008 +
     1009 +    async def _parse_and_release(self, connection, *args, **kwargs):
1005 1010          try:
1006      -            return await self.parse_response(connection, args[0], **kwargs)
     1011 +            return await self.parse_response(connection, *args, **kwargs)
     1012 +        except asyncio.CancelledError:
     1013 +            # should not be possible
     1014 +            await connection.disconnect(nowait=True)
     1015 +            raise
1007 1016          finally:
1008      -            # Release connection
1009 1017              self._free.append(connection)
1010 1018
1011 1019      async def execute_pipeline(self, commands: List["PipelineCommand"]) -> bool:
```

This case study particularly demonstrates how commonly used open source components, when deployed across large-scale systems and combined with specific edge-case misconfigurations, will lead to amplification of bugs and vulnerabilities that should otherwise (at least in theory) not occur at all.

The race condition described here would normally not have triggered to such an extent had OpenAI not made server changes to its Redis instances.

Moral of the story is, if a bug or vulnerability is exploitable under theoretical circumstances that should never occur in the real world, sooner or later they *will* find a way to become exploitable — an opportunity threat actors will not want to miss.

CONCLUSION

The incident involving the Redis bug and its impact on ChatGPT highlights the critical need for a comprehensive SCA tool like Sonatype IQ Server. This case study exemplifies the potential risks associated with open source components when deployed at scale and combined with specific misconfigurations. Sonatype IQ Server offers a powerful suite of features that can help you prevent and mitigate such vulnerabilities.

By leveraging Sonatype IQ Server's advanced SCA capabilities, including component analysis, vulnerability scanning, and configuration management, your organization can proactively identify and address potential issues in your software supply chain. The automation and comprehensive reporting provided by Sonatype IQ Server enables your developers to prioritize remediation efforts effectively and ensure the security of your applications.

Furthermore, Sonatype IQ Server's licensing features such as policy enforcement and vulnerability intelligence empower your organization to enforce security best practices and make informed decisions about component usage. By adopting Sonatype IQ Server, you can enhance vulnerability management, minimize the risk of data breaches, and safeguard software assets against emerging threats.

Google Chrome aka Chromium hackers in the wild: A heap buffer overflow in WebRTC

Name of component: **Chromium**

Name of vulnerability: CVE-2022-2294

Type of vulnerability: Out-of-bounds write

Versions affected: <103.0.5060.114

Criticality:

- CVSS 3.0 Score: 8.8
- CVSS 3.0 Metrics: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

DESCRIPTION

From novices to the most technically savvy, almost everyone has heard of Google Chrome. While Chrome itself may not be an open source development "component," it's largely built off of Google's open source **Chromium**, which is. Available on the Fedora repository, **Chromium** is built using open source software (OSS) components like Skia, v8, WebKit, and WebRTC.

As such, a project as extensive and complex as **Chromium** frequently finds and patches vulnerabilities — many of which have exploits circulating in the wild, according to Google. Interestingly, Apple's Safari web browser, also built using WebKit, often also patches some of the same CVEs that impact Chrome. Of the many, one such vulnerability is **CVE-2022-2294** — a heap buffer overflow fixed in WebRTC within both Google Chrome and Safari.

ATTACK MECHANICS AND REMEDIATION

The vulnerability, reported by Jan Vojtesek from Avast's Threat Intelligence team, concerns a memory corruption flaw within WebRTC which can be exploited by threat actors via malicious web content. For example, a web browser user when visiting a malicious web page that exploits this flaw can potentially be subjecting their machine to arbitrary code execution by the attacker.

The one-line fix for the flaw simply changes a function call — from RTC_DCHECK_EQ() to RTC_CHECK_EQ in the rtp_sender.cc file:

```
Disallow invalid arguments in RestoreEncodingLayers.

Changing DCHECK into CHECK for good measure.

Bug: chromium:1343889
Change-Id: I2cede85dc2d2a4238739f73afe25275047f4aa50
Reviewed-on: https://webrtc-review.googlesource.com/c/src/+/268460
Reviewed-by: Ilya Nikolaevskiy <ilnik@webrtc.org>
Commit-Queue: Henrik Boström <hbos@webrtc.org>
Cr-Commit-Position: refs/heads/main@{#37511}
```

```
diff --git a/pc/rtp_sender.cc b/pc/rtp_sender.cc
index b152bcb..9a8f45c 100644
--- a/pc/rtp_sender.cc
+++ b/pc/rtp_sender.cc
```

```
@@ -74,8 +74,8 @@
     const RtpParameters& parameters,
     const std::vector<std::string>& removed_rids,
     const std::vector<RtpEncodingParameters>& all_layers) {
-  RTC_DCHECK_EQ(parameters.encodings.size() + removed_rids.size(),
-                all_layers.size());
+  RTC_CHECK_EQ(parameters.encodings.size() + removed_rids.size(),
+               all_layers.size());
   RtpParameters result(parameters);
   result.encodings.clear();
   size_t index = 0;
```

Without understanding additional context about what the simple code change does, it would be inappropriate to comment on the finer intricacies of the vulnerability. However, public resources like documentation and code comments help us better understand what these functions are used for and how using one of them over another poses a security risk.

According to code comments on the WebRTC project, RTC_CHECK and RTC_DCHECK are nearly identical functions, each used for assertions or to evaluate if a provided condition is true.

NOTE

Assertions are simple statements that help programmers test their assumptions made within a program. They can "help a programmer read the code, help a compiler compile it, or help the program detect its own defects," as this Wikipedia article describes.

But that is not to say all assertion functions are the same. RTC_DCHECK* function calls, for example, are relevant to and will only be evaluated in debug builds, whereas RTC_CHECK* is appropriate for use in production environments. In other words, presence of RTC_DCHECK* function calls will be ignored in a production build of WebRTC.

RTC_DCHECK_EQ and RTC_CHECK_EQ are simple extensions of these functions and another form of assertions but instead of accepting one condition as an argument, these accept two arguments. The RTC_DCHECK_EQ function will return **true** if both arguments passed to it are equal.

```
The macros here print a message to stderr and abort under various
// conditions. All will accept additional stream messages. For example:
// RTC_DCHECK_EQ(foo, bar) << "I'm printed when foo != bar.";
```

But, the use of RTC_DCHECK_EQ as opposed to RTC_CHECK_EQ in WebRTC production releases gives rise to a bug, which turns into a potentially exploitable vulnerability. This subtle but important distinction can spell out the difference between a program that is safe and another one which can become a security hazard.

CONCLUSION

The heap buffer overflow vulnerability in WebRTC underscores the critical importance of adopting a comprehensive solution for software supply chain management. The Sonatype Platform offers a suite of powerful tools and capabilities that enable your organization to proactively manage open source components and vulnerabilities.

With Sonatype solutions powered by Sonatype IQ Server, you can:

- gain complete visibility into your software supply chain;
- identify and address vulnerabilities early in the software development life cycle (SDLC); and
- enforce security policies across your entire software ecosystem.

The Sonatype Platform combines advanced SCA with continuous monitoring and intelligent automation, empowering your development teams to make more informed decisions about component selection, detect vulnerabilities, and remediate them efficiently.

By leveraging Sonatype's robust features and expertise, you can mitigate risks, enhance application security posture, and ensure compliance with industry regulations. Embracing the Sonatype Platform is a strategic investment that enables you to fortify your software supply chain, defend against emerging threats, and deliver secure and reliable products.

The cursed session: WildFly Elytron

Name of component: **WildFly Elytron**

Name of vulnerability: CVE-2020-10714

Type of vulnerability: Session fixation

Versions affected: 1.11.3.Final and before

Criticality:

- CVSS 3.0 Score: 7.5
- CVSS 3.0 Metrics: CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H

DESCRIPTION

In the wide spectrum of vulnerabilities, the most dangerous ones are those that are easily exploitable. The so-called “low-hanging fruit” are vulnerabilities that can be abused by even the most inexperienced hackers and make applications targets for all kinds of security enthusiasts and threat actors.

Such is the case with **WildFly Elytron**, a security framework found to be vulnerable to session fixation attacks. This component, which is meant to replace PicketBox, shipped with WildFly since version 10 and Red Hat JBoss Enterprise Application Platform (EAP) 7.1. The above, and the fact that it can be used as a stand-alone library, are key indicators that this issue could potentially affect numerous Java server environments.

ATTACK MECHANICS AND REMEDIATION

The vulnerability, found in v1.11.3.Final and previous versions, allows attackers to trick victims into accessing affected applications with predefined session IDs. The image below shows how easily an attacker can get a valid session by placing a known session ID in the login URL of an application that uses FORM authentication. Once the victim logs in, a new session is assigned to the supplied session ID, enabling the attacker to supplant the victim and act on their behalf.

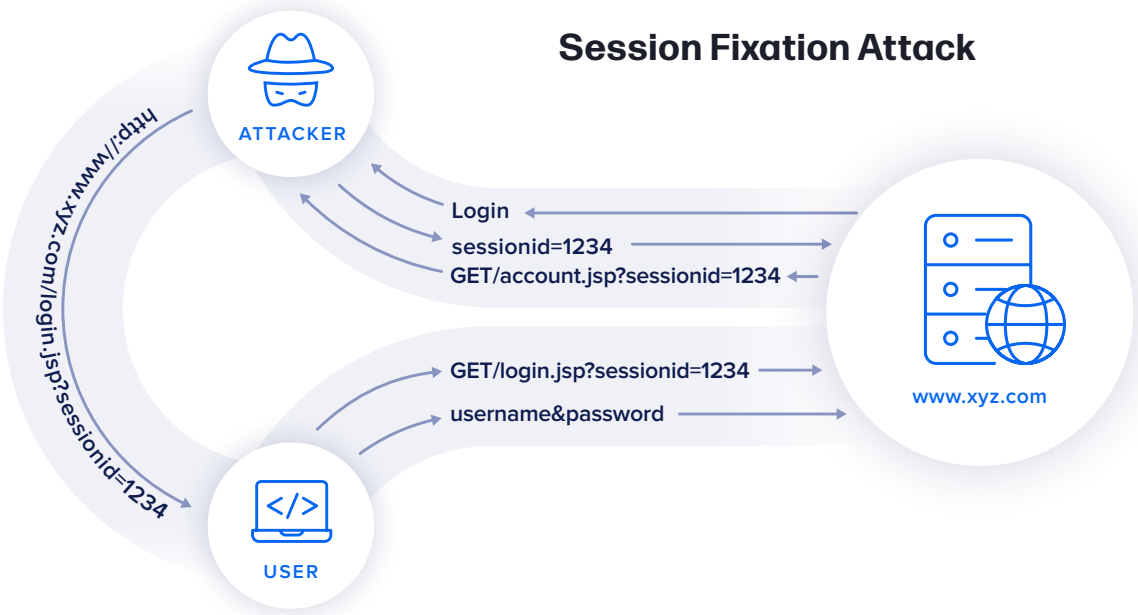
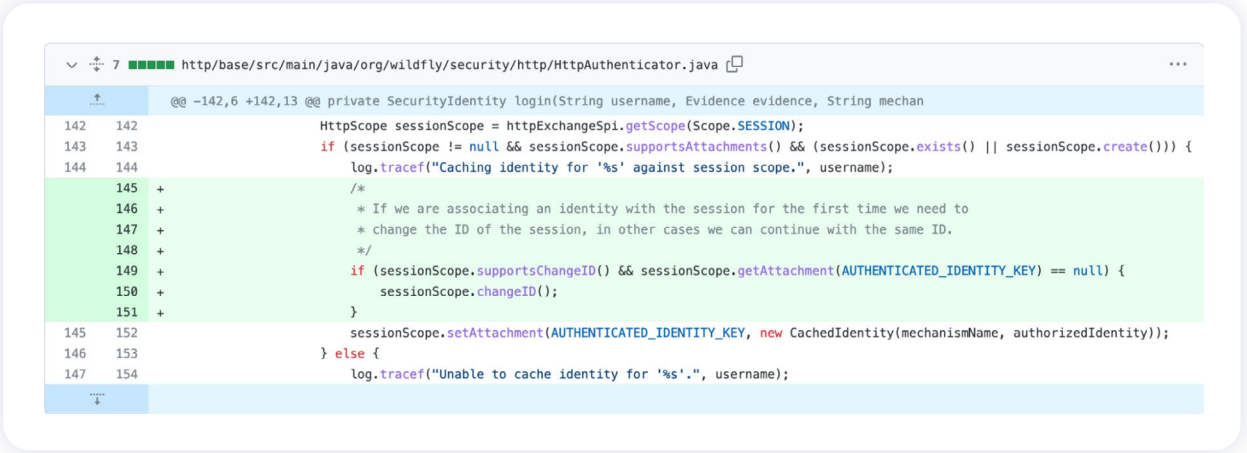


Image adapted from: https://www.geeksforgeeks.org/session-fixation-attack/

The issue usually arises from improper session management that fails to regenerate IDs on logins, logouts, password changes, and any other actions that might alter the session state.

The maintainers chose to fix this security issue by regenerating the session ID after a successful login, thus making the attack vector ineffective.



This fix was also applied to the SPNEGO authentication mechanism which also depends on sessions to persist the current identity, but it's less common as it tends to be used on internal networks rather than the internet.

The attack relies on the ability of the attacker to easily place an ID in the URL and keep the session active. Even though sessions have a 15-minute timeout by default, additional measures can be taken to mitigate this vulnerability. Servers that support session tracking by URL and Cookies can be adjusted to only allow Cookies by modifying the configuration value in the web.xml file, thus also making the attack vector ineffective as it cannot be exploited by sharing the link to the login page.

Additionally, the following mitigation suggestion comes from Red Hat Bugzilla:

From:

```
~~~
  <session-config>
    <tracking-mode>URL</tracking-mode>
  </session-config>
~~~
```

To:

```
~~~
  <session-config>
    <tracking-mode>COOKIE</tracking-mode>
  </session-config>
~~~
```

CONCLUSION

The session fixation vulnerability in **WildFly Elytron** illustrates the significance of addressing even seemingly simple vulnerabilities, which can still be easily exploited. To tackle this challenge, Sonatype offers comprehensive software supply chain management solutions that:

- streamline vulnerability management processes;
- enforce best practices; and
- proactively mitigate risks posed by open source components.

By incorporating Sonatype solutions into your workflows, you can bolster your software supply chain, ensure the integrity of your applications, and reduce security risks.

Keeping track of every component within large-scale applications can be a daunting task. But with Sonatype IQ Server powering your defense system, you can identify and address vulnerabilities early in the SDLC and more effectively and efficiently secure your applications.

Sonatype stands as a trusted partner in the face of a rapidly evolving threat landscape. By embracing Sonatype's solutions, your organization can enhance its overall security posture and deliver secure and reliable products.

Beyond Log4Shell: An end-of-life version of Log4j with a Chainsaw vulnerability

Name of component: **Apache Chainsaw**

Name of vulnerability: CVE-2022-23307

Associated CVEs: CVE-2020-9493

Type of vulnerability: Deserialization of untrusted data

Versions affected: <2.1.0

Criticality:

- CVSS 3.0 Score: 8.8
- CVSS 3.0 Metrics: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

DESCRIPTION

When talking about vulnerable components from an impact perspective, we need to talk about Log4j. The latest 1.x release was made way back in 2012, due to it being an end-of-life product; however, it is still used by over 100,000 projects, heavily downloaded, and has seven known vulnerabilities which can be seen in Sonatype OSS Index. This project is a Java logging framework used everywhere from games such as Minecraft to being a core component in applications built by industry giants, making it hard to ignore when new critical vulnerabilities arise affecting just about anything that uses Java.

It's been almost two years since the critical Log4Shell vulnerability hit — primarily impacting 2.x versions like 2.15. The security industry was upside-down for a bit trying to fully understand the impact this had and what other issues would stem down from this major discovery. Then, when we thought it might all be over, more critical vulnerabilities arose — this time in `**Log4j 1.x**`. Despite the age of this version, in a fast-evolving industry, it's not always easy to keep all of your dependencies updated to the latest and safest versions. We see this clearly in our vulnerable download dashboard, where we observe millions of downloads a week of vulnerable versions.

ATTACK MECHANICS AND REMEDIATION

Enter **CVE-2022-23307** (Sonatype's CVSS rating: 9.8 / Critical). This remote code execution (RCE) flaw is applicable to default instances, making it noteworthy. The vulnerability itself lurks in the **Apache Chainsaw** component, which is included within Log4j 1.x versions. **Chainsaw** versions prior to 2.1.0 were vulnerable to untrusted deserialization; therefore, the inclusion of this version in Log4j 1.x makes the latter vulnerable too.

Reported by a pseudonymous researcher with the moniker *@kingkk*, **CVE-2022-23307** is essentially the same issue as **CVE-2020-9493**, with the newer identifier assigned specifically for Log4j. We have

to also remember that Logback is a successor to Log4j, “picking up where log4j 1.x leaves off.” Which begs the question, is `**Logback**` also impacted by this?

Firstly, there is no indication that Logback utilizes **Chainsaw**, so we cannot jump to any conclusions. But consider that when an OSS project forks into many others, critical vulnerabilities might be forked along with it. Even vulnerabilities later discovered in future forks can often impact its predecessors. Understanding the code we utilize and import into our applications is a must. We have to know exactly what we are working with and have a way to manage it. Here is where a software bill of materials (SBOM) comes in handy.

On peeking inside one of the Log4j 1.x JARs using “tar -tvf,” one can trivially spot the presence of **Chainsaw** classes:

```
log4j_log4j % tar tvf 1.2.6/log4j-1.2.6.jar| grep -i chainsaw
  0           0  1 Aug  2002 org/apache/log4j/chainsaw/
  0        1173  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel$1.class
  0        1356  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel$2.class
  0        1345  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel$3.class
  0        1340  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel$4.class
  0        1344  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel$5.class
  0         904  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel$6.class
  0        1162  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel$7.class
  0        5345  1 Aug  2002 org/apache/log4j/chainsaw/ControlPanel.class
  0        5199  1 Aug  2002 org/apache/log4j/chainsaw/DetailPanel.class
  0        2313  1 Aug  2002 org/apache/log4j/chainsaw/EventDetails.class
  0        1478  1 Aug  2002 org/apache/log4j/chainsaw/ExitAction.class
  0        4307  1 Aug  2002 org/apache/log4j/chainsaw/LoadXMLAction.class
  0        2371  1 Aug  2002 org/apache/log4j/chainsaw/LoggingReceiver$Slurper.class
  0        2888  1 Aug  2002 org/apache/log4j/chainsaw/LoggingReceiver.class
  0         806  1 Aug  2002 org/apache/log4j/chainsaw/Main$1.class
  0        6625  1 Aug  2002 org/apache/log4j/chainsaw/Main.class
  0         784  1 Aug  2002 org/apache/log4j/chainsaw/MyTableModel$1.class
  0        2337  1 Aug  2002 org/apache/log4j/chainsaw/MyTableModel$Processor.class
  0        8389  1 Aug  2002 org/apache/log4j/chainsaw/MyTableModel.class
  0        3882  1 Aug  2002 org/apache/log4j/chainsaw/XMLFileHandler.class
```

Now let’s take a look at the actual root cause and our main interest, which is an untrusted deserialization. For this, we can decompile one of the `**LoggingReceiver.class**`, which processes an InputStream within the run() method of its Slurper nested class:

Line 54 is where the Serializable interface’s readObject() method is invoked, without verifying the object type or restricting deserialization to only certain types of objects. As a result, an attacker who is able to send a malicious payload to the vulnerable Log4j 1.x version can trigger arbitrary code execution.

As mentioned above, Log4j 1.x versions have been end-of-life for quite some time now. As such, upgrading to the latest Log4j 2.x version is the recommended upgrade route.

CONCLUSION

Log4j 1.x vulnerabilities, such as **Apache Chainsaw**, show the risks associated with outdated components. Upgrading to Log4j 2.x is crucial to mitigate these risks. With next-generation solutions powered by Sonatype IQ Server, you have access to comprehensive security data and detection capabilities, including binary scanning, enabling effective vulnerability identification.

Sonatype's solutions also go beyond detection by providing dependable remediation guidance. By leveraging our expertise, your organization can proactively address vulnerabilities and secure your software supply chain.

By embracing proactive vulnerability mitigation, you can effectively manage your dependencies, track threats, and implement timely upgrades. Sonatype's solutions empower you to protect critical assets from potential attacks and stay vigilant in the face of an expanding attack surface. Our commitment to continuous improvement and comprehensive security measures makes us a valuable partner to defend against evolving threats and ensure the integrity of your SDLC practices.

Prioritizing Open Source Security: Sonatype's Data-Driven Approach

In this whitepaper, we highlighted our top five vulnerable open source components, shedding light on the importance of security in the open source landscape. As the use of open source components continues to dominate software development practices, so does the need for precise identification and comprehensive intelligence to mitigate risks.

Sonatype's extensive scanning and data research revealed alarming statistics regarding the vulnerabilities in popular open source components. The findings emphasize the significance of addressing security vulnerabilities proactively. Our commitment to providing accurate and reliable data enables you and your organization to make informed decisions about version selection and remediation.

The vulnerabilities outlined in this whitepaper serve as a wake-up call for the software industry to prioritize security in the open source landscape. With Sonatype's expertise and data-driven solutions, organizations can navigate the complex world of open source software with confidence, ensuring their software supply chains remain secure, reliable, and innovative.

Headquarters
8161 Maple Lawn Blvd, Suite 250
Fulton, MD 20759
USA • 1.877.866.2836

European Office
168 Shoreditch High St, 5th Fl
London E1 6JE
United Kingdom

APAC Office
60 Martin Place, Level 1
Sydney 2000, NSW
Australia

Sonatype Inc.
www.sonatype.com