AWS's Log4Shell Hot Patch Vulnerable to Container Escape and Privilege Es

By Yuval Avrahami
April 19, 2022 at 3:00 PM
Category: Cloud, Vulnerability
Tags: Apache Log4j, AWS, container escape, containers, CVE-2021-3100, CVE-2021-3101, CVE-2021-44228, CVE-2022-0070, CVE-2022-0071, log4j, privilege escalation

This post is also available in: 日本語 (Japanese)

Executive Summary

Following Log4Shell, AWS released several hot patch solutions that monitor for vulnerable Java applications and Java containers and patch them on the fly. Each solution suits a different environment, covering standalone servers, Kubernetes clusters, Elastic Container Service (ECS) clusters and Fargate. The hot patches aren't exclusive to AWS environments and can be installed onto any cloud or on-premises environment.

Unit 42 researchers identified severe security issues within these patching solutions and partnered with AWS to remediate them. After installing the patch service to a server or cluster, every container in that environment can exploit it to take over its underlying host. For example, if you installed the hot patch to a Kubernetes cluster, every container in your cluster can now escape until you either disable the hot patch or upgrade to the fixed version. Aside from containers, unprivileged processes can also exploit the patch to escalate privileges and gain root code execution.

Containers can escape regardless of whether they run Java applications, or whether their underlying host runs Bottlerocket, AWS's hardened Linux distribution for containers. Containers running with user namespaces or as a non-root user are affected as well. Unit 42 assigned CVE-2021-3100, CVE-2021-3101, CVE-2022-0070 and CVE-2022-0071 to track the vulnerabilities.

AWS released a fixed version for each hot patch solution on April 19:

1. Version 1.1-16 of the `log4j-cve-2021-44228-hotpatch` package, which bundles the hot patch service.
2. Version 1.1-16 of the `kubernetes-log4j-cve-2021-44228-node-agent` Daemonset, which installs the updated package.
3. Version 1.02 of Hotdog, a hot patch solution for Bottlerocket hosts based on Open Container Initiative (OCI) hooks.

Unit 42 advises anyone who installed any of these hot patches to upgrade to a fixed version. Note that starting from Dec. 17, 2021, JDK packages (Java installations) on Amazon Linux automatically installed the `log4j-cve-2021-44228-hotpatch` package. Alternatively, users who are confident their applications are patched against Log4Shell can disable the hot patch service following the instructions in the Mitigations section below.

Prisma Cloud detects the hot patch package and will alert on hosts running a vulnerable version.

CVEs Assigned	CVE-2021-3100, CVE-2021-3101, CVE-2022-0070, CVE-2022-0071
Related Unit 42 topics	Container escape, privilege escalation, cloud, Apache log4j

Table of Contents

Overview of AWS Log4Shell Hot Patches

Log4Shell proved itself as one of the worst vulnerabilities of recent times. To help users combat the issue at scale, AWS open-sourced several hot patch solutions, each cover
the process of injecting a fix to a vulnerable running application. It's meant to serve as a short-term solution until a new, fixed version of the application can be deployed.

AWS released three hot patching solutions that detect processes and containers running vulnerable Java applications and patch them on the fly:

1. A hot patch service bundled in an RPM package. Starting from Dec. 17, 2021, this service is automatically installed with Amazon Linux JDK (Java) packages. Fargate custor
installed on the hosts running their containers.
2. A hot patch Daemonset for Kubernetes clusters, which installs the aforementioned hot patch service on all nodes.
3. Hotdog, a hot patch solution bundled as a set of OCI hooks. Hotdog is primarily intended for Bottlerocket hosts.

These solutions cover most compute environments, from Kubernetes clusters to ECS clusters, Fargate containers and standalone servers. They aren't exclusive to AWS envir
cloud environments or on-premises.

Unit 42 researchers discovered these patches can be exploited for container escape and privilege escalation. After any one of the patches is installed to a host or cluster, new
and compromise their underlying host. On hosts that installed either the hot patch service or the hot patch Daemonset, existing containers can escape as well. Aside from co
exploit the patch service to escalate privileges and gain root code execution. AWS has now mitigated these vulnerabilities and released a fix for each solution.

Root Cause Analysis

AWS's hot patch solutions continuously search for Java processes and patch them against Log4Shell on the fly. Any process running a binary named "java" – inside or outside of a container – is considered a candidate for the hot patch.

To patch Java processes inside containers, the hot patch solutions invoke certain container binaries. For example, they run the container's "java" binary twice: once to retrieve the Java version, and again to inject the hot patch. The issue was that they invoked container binaries without properly containerizing them. That is, the new processes would run without the limitations normally applied to container processes.

For example, the "java" binary was invoked in the container namespaces via the nsenter command (excluding the user namespace). But aside from that, it was spawned with all Linux capabilities, and without the isolation technologies that normally confine containers, such as seccomp and cgroups. It also ran as the root user regardless of the container's user.

A malicious container therefore could have included a malicious binary named "java" to trick the installed hot patch solution into invoking it with elevated privileges. The malicious "java" process could then abuse its elevated privileges to escape the container and take over the underlying host. The fixed hot patch solutions now properly containerize container binaires before running them.

Aside from containers, the hot patch service also patched host processes in a similar manner. A malicious unprivileged process could have created and run a malicious binary named "java" to trick the hot patch service into executing it with elevated privileges. The fixed hot patch service now spawns "java" binaries with the same privileges as the Java process being patched.

Container Escape Demo

To verify the vulnerability is exploitable, we built a proof of concept (PoC) container image. When deployed to a cluster or VM that runs a vulnerable version of a hot patch solution, the container exploits the vulnerabilities to escape and gain root code execution on the underlying host. It then sends a reverse shell to an attacker-controlled server.

In the demo video below, a user installed the hot patch Daemonset to an EKS cluster. The demo then simulates a supply chain attack by showing what happens when the user inadvertently runs a malicious container image that exploits the hot patch.

Video 1. CVE-2021-3100 exploit demo.

While the demo showcases a supply chain attack, existing containers that were compromised (e.g. by a network payload) can also exploit the issues to escape and take over their underlying host. We've decided not to share the exploit's implementation details at this time to prevent malicious parties from weaponizing it.

Impact

Given the urgency surrounding Log4Shell, users may have deployed hot patches at scale, inadvertently putting container environments at risk. Even after Java applications were patched against Log4Shell, users may have kept the hot patch running for defense-in-depth as there isn't a strong incentive to remove it.

Containers are often used as a security boundary between applications running on the same machine. A container escape allows an attacker to extend a campaign beyond a single application and compromise neighboring services. In Kubernetes clusters, a single container escape is unfortunately sometimes enough to take over the entire cluster.

Mitigations

AWS released a fix for each hot patch solution. Once a host runs a fixed version, container escape and privilege escalation are no longer possible.

1. In Kubernetes clusters, you can install the fixed hot patch version by deploying the latest Daemonset provided by AWS. Note that only deleting the hot patch Daemonset (your nodes. **Updated April 25**: Currently, there isn't a fixed Daemonset version for Debian-based hosts (Debian and Ubuntu). See this GitHub thread for more details. **Update** Debian-based hosts was released. Note that the fixed version for the Debian `log4j-cve-2021-44228-hotpatch` package is **1.1.17**.

2. On standalone hosts, you can upgrade by running `yum update log4j-cve-2021-44228-hotpatch`.

3. Hotdog users need to upgrade to the latest version.

Alternatively, if you're confident that your environment is patched against Log4Shell, you can disable the hot patch service on a host by running `sudo touch /etc/log4j` disable Hotdog, run `apiclient set oci-hooks.log4j-hotpatch-enabled=false`.

Prisma Cloud customers can identify affected hosts under the Vulnerabilities tab. The platform detects the hot patch packages and alerts customers on VMs running a vulner vulnerabilities, use the Amazon Linux Security Advisories (ALAS) IDs associated with them: ALAS-2021-1554, ALAS-2021-1732, ALAS-2022-1580 and ALAS-2022-1773.

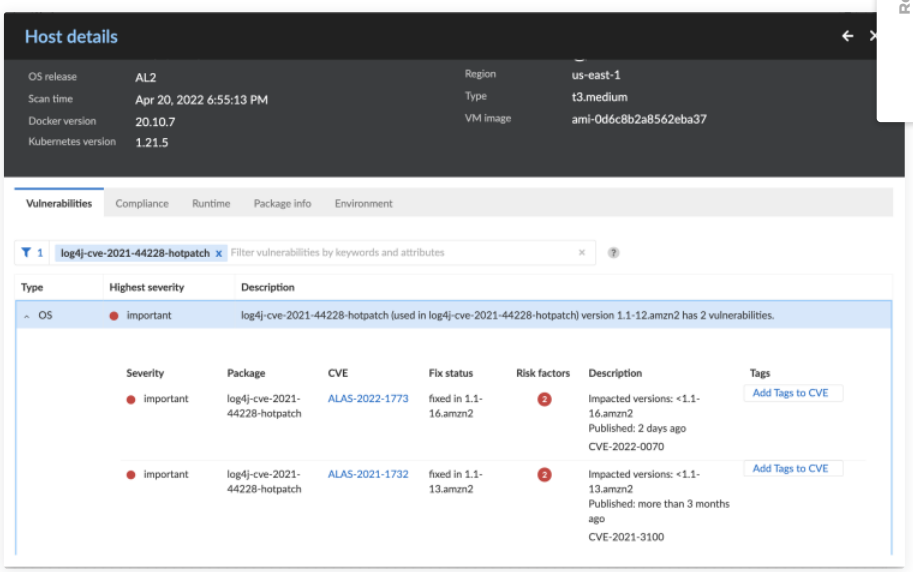

Figure 1. Prisma Cloud detects and alerts on vulnerable log4j-cve-2021-44228-hotpatch versions.

Palo Alto Networks Prisma Cloud, Cortex XDR and Next-Generation Firewalls (NGFWs) can detect follow-on attacker activities and disrupt command and control communications like the reverse shell used in the demo.

Safely Interacting With Containers

CVE-2021-3100, CVE-2021-3101, CVE-2022-0070 and CVE-2022-0071 add to a long list of container escape vulnerabilities that arise from a host process directly interacting with a running container. Simple tasks like copying files or spawning a new containerized process can have surprising outcomes when the container is malicious.

If you're building software around containers, defer to an established container runtime like runc for operations involving a container's processes or filesystem. Although they have also had their share of vulnerabilities, container runtimes are by far the most vetted and mature programs for safely interacting with containers.

Conclusion

Given the urgency surrounding Log4Shell, users may have deployed hot patches at scale, inadvertently putting container environments at risk. We encourage users to upgrade to the fixed hot patch version as soon as possible. Multitenant container environments and clusters running untrusted images are especially at risk.

If you're still patching against Log4Shell, prioritize that effort first. While the presented issues can lead to severe attacks against container environments, Log4Shell has rightfully earned its spot as one of the worst vulnerabilities of all time and is still being actively exploited.

We'd like to thank AWS for their partnership and coordination in remediating this vulnerability efficiently. As Log4Shell exploitation peaked, AWS's hot patch helped the community stop countless attacks. With these vulnerabilities fixed, it's now possible to use the hot patch to address Log4Shell while also keeping container environments secure.

Additional Resources

- Unit 42 analysis of Log4Shell
- AWS advice on mitigating Log4Shell in container enviroments
- Prisma Cloud Mitigations for Log4Shell

Disclosure Timeline

- **Dec. 14:** AWS releases hot patch package with support for containers.

- **Dec. 27**: Unit 42 reports bypasses for the initial fixes to AWS.
- **Feb. 9**: Unit 42 researchers meet with AWS security to discuss fixes.
- **April 1**: AWS shares fixed versions for Unit 42 review.
- **April 4**: Unit 42 points out a few remaining issues.
- **April 19**: AWS releases final fixes and advisories; Unit 42 discloses the vulnerabilities publicly.

Updated May 5, 2022, at 11:10 a.m. PT.