Microsoft Security Response Center

Microsoft's Response to CVE-2021-44228 Apache Log4j 2

MSRC / By MSRC Team / December 11, 2021

Published on: 2021 Dec 11

SUMMARY

Microsoft continues our analysis of the remote code execution vulnerability (CVE-2021-44228) related to Apache Log4j (a logging tool used in many Java-based applications) disclosed on 9 Dec 2021. As we and the industry at large continue to gain a deeper understanding of the impact of this threat, we will publish technical information to help customers detect, investigate, and mitigate attacks, as well as guidance for using Microsoft security solutions to increase resilience against related attacks. We will update this blog with information and protection details as they become available.

In addition to monitoring the threat landscape for attacks and developing customer protections, our security teams have been analyzing our products and services to understand where Apache Log4j may be used and are taking expedited steps to mitigate any instances. If we identify any customer impact, we will notify the affected party. Our investigation to date has identified mitigation steps customers could take in their environments as well as on our services.

Apply the Latest Security Updates

To address this vulnerability, Microsoft recommends customers apply the latest security updates to remediate this vulnerability. Please review the Apache CVE and the Apache security advisory for further details:

- Apache CVE: CVE-2021-44228
- Apache security advisory: Apache Log4j Security Vulnerabilities

All systems, including those that are not customer facing, are potentially vulnerable to this exploit, so backend systems and microservices should also be upgraded. The recommended action is to update Log4j 2 to 2.15.0. A service restart will be required.

Workarounds

To help mitigate the risk of this vulnerability until the more complete security update can be applied, customers should consider the following mitigations steps. A service restart will be required for these changes to take

effect. These workarounds should not be considered a complete solution to resolve this vulnerability:

- In case the Log4j 2 vulnerable component cannot be updated, Log4J 2 versions 2.10 to 2.14.1 support the parameter log4j2.formatMsgNoLookups to be set to 'true', to disable the vulnerable feature. Ensure this parameter is configured in the startup scripts of the Java Virtual Machine: -Dlog4j2.formatMsgNoLookups=true.
- Alternatively, customers using Log4j 2.10 to 2.14.1 may set the LOG4J_FORMAT_MSG_NO_LOOKUPS="true" environment variable to force this change.
- Kubernetes administrators may use "kubectl set env" to set the LOG4J_FORMAT_MSG_NO_LOOKUPS="true" environment variable to apply the mitigation across Kubernetes clusters where the Java applications are running Log4j 2.10 to 2.14.1, effectively reflecting on all pods and containers automatically.
- For releases from 2.0-beta9 to 2.10.0, the mitigation is to remove the JndiLookup class from the classpath: zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class

Background of Log4j

The vulnerability, tracked as CVE-2021-44228 and referred to as "Log4Shell," affects Java-based applications that use Log4j 2 versions 2.0 through 2.14.1. Log4j 2 is a Java-based logging library that is widely used in business system development, included in various open-source libraries, and directly embedded in major software applications. The scope of impact has expanded to thousands of products and devices, including Apache products such as Struts 2, Solr, Druid, Flink, and Swift. Because this vulnerability is in a Java library, the cross-platform nature of Java means the vulnerability is exploitable on many platforms, including both Windows and Linux. As many Java-based applications can leverage Log4j 2, organizations should contact application vendors or ensure their Java applications are running the latest up-to-date version. Developers using Log4j 2 should ensure that they are incorporating the latest version of Log4j into their applications as soon as possible in order to protect users and organizations.

Analysis of the CVE-2021-44228 vulnerability

The vulnerability is a remote code execution vulnerability that can allow an unauthenticated attacker to gain complete access to a target system. It can be triggered when a specially crafted string is parsed and processed by the vulnerable Log4j 2 component. This could happen through any user provided input.

Successful exploitation allows for arbitrary code execution in the targeted application. Attackers do not need prior access to the system to log the string and can remotely cause the logging event by using commands like curl against a target system to log the malicious string in the application log. When processing the log, the vulnerable system reads the string and executes it, which in current attacks is used to execute the code from the malicious domain. Doing so can grant the attacker full access and control of the affected application.

Given the fact that logging code and functionalities in applications and services are typically designed to process a variety of external input data coming from upper layers and from many possible vectors, the biggest

risk factor of this vulnerability is predicting whether an application has a viable attack vector path that will allow the malformed exploit string to reach the vulnerable Log4j 2 code and trigger the attack.

A common pattern of exploitation risk, for example, is a web application with code designed to process usernames, referrer, or user-agent strings in logs. These strings are provided as external input (e.g., a web app built with Apache Struts). An attacker can send a malformed username or set user-agent with the crafted exploit string hoping that this external input will be processed at some point by the vulnerable Log4j 2 code and trigger code execution.

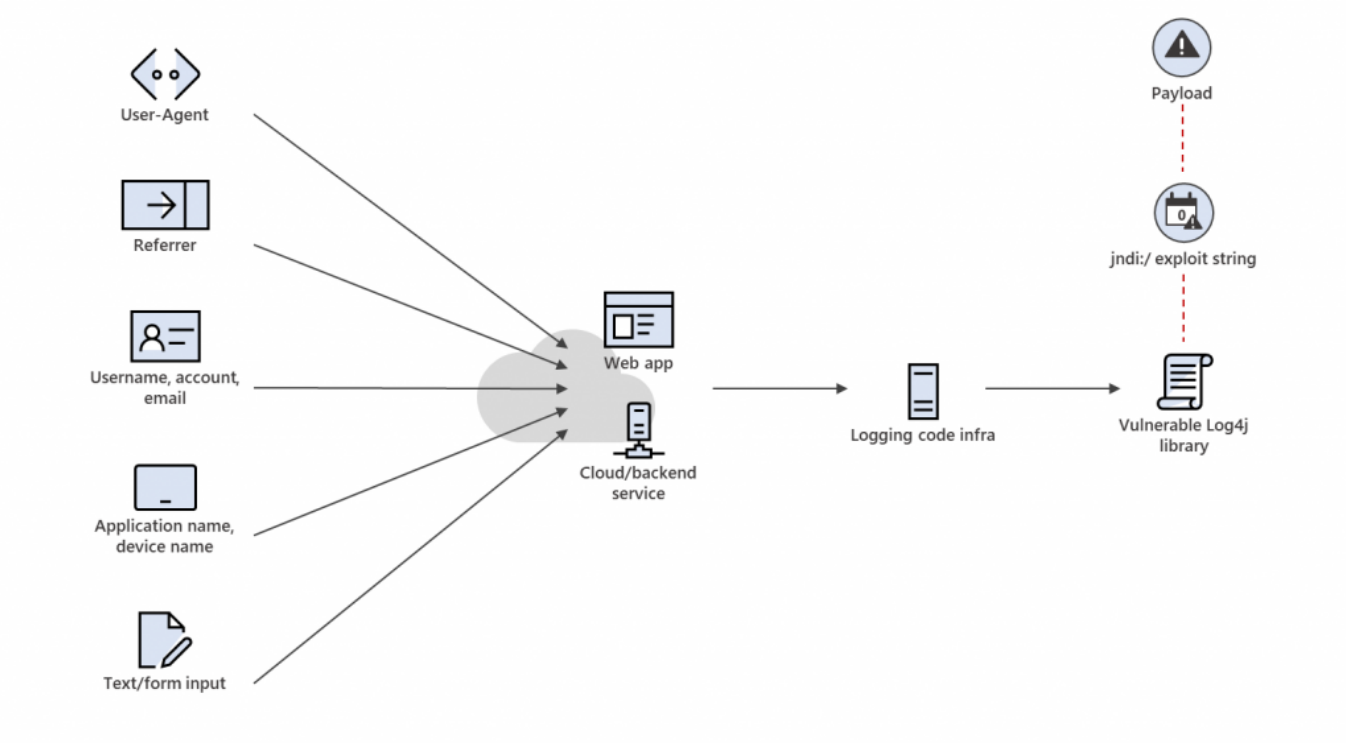


Figure 1. CVE-2021-44228 exploit vectors and attack chain

Mitigation Guidance for Microsoft Services

After further analysis of our services and products, below are a few mitigation strategies given by various Microsoft services.

Microsoft Azure AD

While the industry is determining and mitigating overall exposure, attackers are probing all endpoints for vulnerability. Applying rigorous least privilege access policies to all resources in your environment is critical. If you use Azure Active Directory for single-sign on in your environment, we recommend you do the following with a special focus on applications you deploy or manage directly (SaaS apps, including those deployed by Microsoft, must be secured by their vendors). Note that log4j2 usage may be pre-auth for some of your applications, but these steps will help prevent post-authentication exploitation. Templates and examples for these policies are built in to facilitate deployment:

- To facilitate these steps and minimize business impact, you may wish to use application tagging capabilities to identify those applications which you haven't validated as patched and target stricter policies to them.
- Enable MFA for all access to these resources to prevent probing using accounts with compromised passwords.
- If you are using Azure AD Identity Protection, enable blocking on risky logins (we recommend blocking on medium or higher). This will prevent access via ToR exit nodes and anonymizing VPNs.
- If you use Azure AD Conditional Access, restrict access to your resources to known/trusted locations or networks.
- Require Azure AD joined or better, MDM managed devices to access these resources.
- Monitor the risky sign in reports or use the risk workbook to track anomalous logins to your applications to help focus your investigations.
- While ADFS, as a Windows service, does not use the impacted libraries, other federation providers do. If you use a non-Microsoft federation provider (for example, for SAML 2.0), watch for token anomalies which indicate compromise of those systems.

For key guidance on securing your identity deployment, see https://aka.ms/securitysteps.

Azure Application Gateway, Azure Front Door, and Azure WAF

In our investigation so far, we have not found any evidence that these services are vulnerable however customer applications running behind these services might be vulnerable to this exploit. We highly recommend customers to follow mitigations and workarounds mentioned in this blog to protect their applications. In parallel we are also working on enhancing WAF managed RuleSets to help protect against this vulnerability.

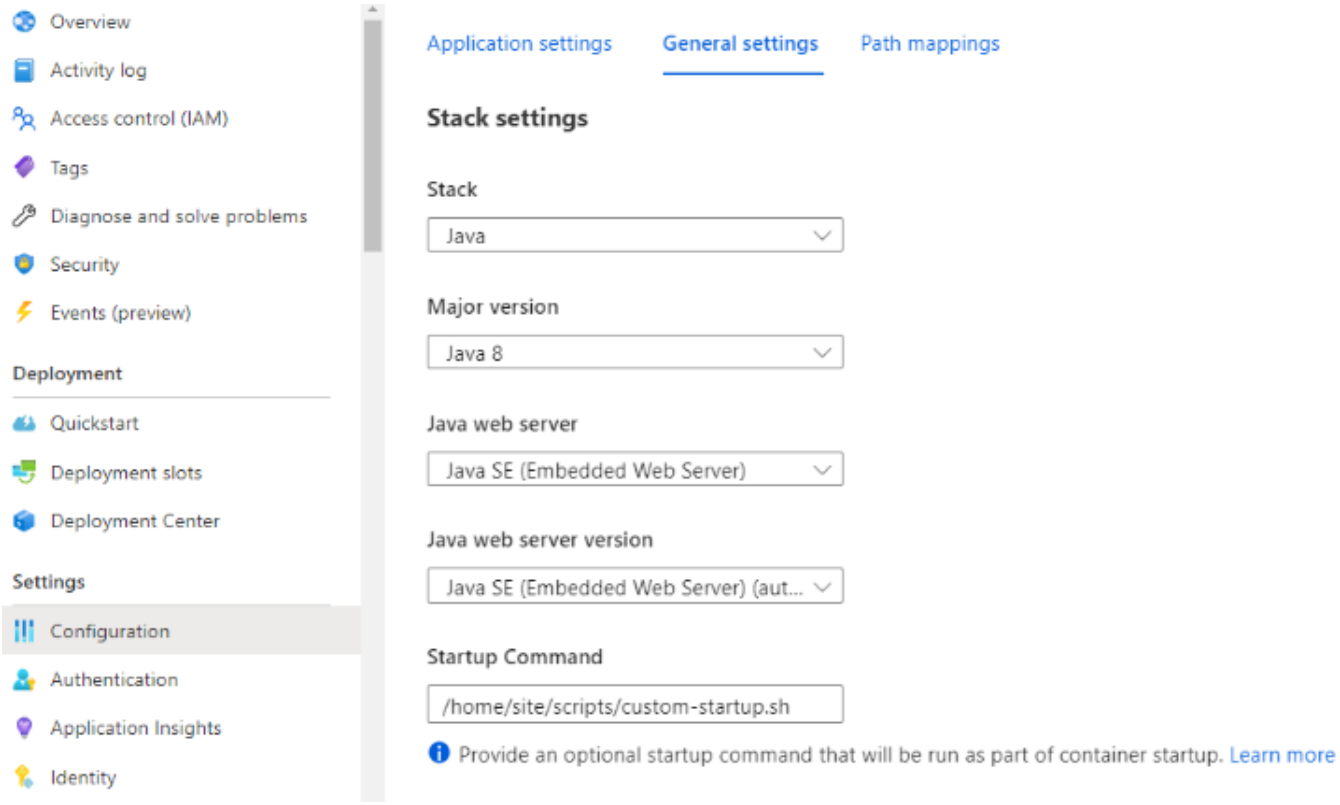
Azure App Service (Windows and Linux)

Azure App Service and Functions does not distribute Log4J in the managed runtimes such as Tomcat, Java SE, JBoss EAP, or the Functions Runtime. However, your applications may use Log4J and be susceptible to this vulnerability.

If possible, customers should upgrade Log4j to version v2.15.0 and re-deploy applications. This is the primary recommended mitigation. If you are not able to re-deploy your application, then in Log4j versions 2.10 and greater, you can mitigate this behavior by setting the system property "-Dlog4j2.formatMsgNoLookups=true". On App Service, you can set this property by creating an app setting named JAVA_OPTS with a value of "-Dlog4j2.formatMsgNoLookups=true". The JAVA_OPTS app setting is passed to your Java application as it starts. If you already have the JAVA_OPTS app setting set, just append "-Dlog4j2.formatMsgNoLookups=true" to the existing value. If you are using Log4J version 2.9 or lower, this system property mitigation will not work and you should upgrade to v2.15.0.

There are a few cases on App Service where you will need to set this system property differently:

If you are using a custom startup script to launch your applications, you will need to add this system property to the startup script.



On Windows App Service, if you use HttpPlatformHandler to start your Java applications, then you will need to add this configuration in your web.config file. If you are not familiar with these terms, then it likely does not apply to your app.

Azure App Service for Containers

If you are deploying Linux or Windows containers to App Service, then you will need to set the system property in your container image. Most likely this will be in the entry point of your Dockerfile.

Azure Functions

Configuring the system property will depend on your choice of hosting option: dedicated, premium or consumption. As a reminder, the primary recommended mitigation is upgrading Log4J to 2.15.0 and re-deploying your application. If you cannot do that for whatever reason, then you can apply the system property.

- **Dedicated and Premium Functions**: Create an app setting named JAVA_OPTS with a value of "-Dlog4j2.formatMsgNoLookups=true". If you already have the JAVA_OPTS app setting set, just append "-Dlog4j2.formatMsgNoLookups=true" to the existing value.
- **Consumption Functions:**
  - **Linux**: Create an app setting named "languageWorkers__java__arguments" with a value of "-Dlog4j2.formatMsgNoLookups=true".
  - **Windows**: Create an app setting named "languageWorkers:java:arguments" with a value of "-Dlog4j2.formatMsgNoLookups=true".

Note that updating app setting will restart your Web and Function apps, which could affect cold start performance. If you are using Log4J version 2.9 or lower, this system property mitigation will not work and you should upgrade to v2.15.0.

Information for Security Operations and Hunters

Microsoft security teams have put together the following guidance and resources to help customers understand this vulnerability and to help detect and hunt for exploits:

- Microsoft Security blog describing the nature of current attacks Microsoft is observing. The blog also contains guidance on how to use Microsoft security products to detect and hunt for malicious activity, and apply protections: Guidance for preventing, detecting, and hunting for CVE-2021-44228 Log4j 2 exploitation
- RiskIQ (acquired by Microsoft in August 2021) published threat intelligence article to the community portal with information about the vulnerability and exploitation of it, as well as detections and mitigations: CVE-2021-44228 Apache Log4j Remote Code Execution Vulnerability
- Microsoft 365 Defender threat analytics article with detection information and potential impacts to customer environments: Threat Insights: CVE-2021-44228 Log4j active exploitation (sign in is required)

We will further update this guidance as we continue to learn from our investigation.

The MSRC Team

← Previous Post Next Post →

Search …

Categories

BlueHat (179)

Japan Security Team (937)

MSRC (970)

Security Research & Defense (369)

Tags

advisory (60) ANS (47) Attack (43) Attack Vector (68) Black Hat (33) BlueHat Security Briefings (55) Community-based Defense (90) Defense-in-depth (38) EcoStrat (34) EMET (68) Exploitability (77) Internet Explorer (IE) (156) malware (59) Microsoft Office (81) Microsoft Windows (106) Mitigations (126) monthly bulletin release (48) rating (48) Risk Asessment (104) security (79) Security Advisory (134) Security Bulletin (133) security bulletin release (44) Security Bulletins (39) Security Conference Engagement (56) Security Ecosystem (52) Security Engineering (42) Security Research (77) Security Update (139) Security Update Webcast (46) Security Update Webcast Q & A (70) Update Tuesday (63) Webcast (37) Windows Update (68) Workarounds (74) Zero-Day Exploit (36) アドバイザリ (147) セキュリティ (53) セキュリティ情報 (454) セキュリティ更新 (79) ワンポイント (39) 啓発 (44) 展開 (45) 時事ネタ (42) 脆弱性 (244)

Recent Posts

Archives

Select Month