Preliminary Draft[1]

How Does CVE-2023-24998 Impact ESAPI?

Kevin W. Wall <kevin.w.wall@gmail.com>

Summary

Category:	Apache Commons FileUpload before 1.5 does not limit the number of request parts to be processed resulting in the possibility of an attacker triggering a Denial of Service (DoS) with a malicious upload or series of uploads. Note that, like all of the file upload limits, the new configuration option (FileUploadBase#setFileCountMax) is not enabled by default and must be explicitly configured.
Module:	Apache Commons FileUpload – a direct compile-time dependency used by ESAPI to support "file uploading" over HTTP.
Announced:	In this security bulletin.
Credits:	GitHub Dependabot, Snyk, Martin Bektchiev
Affects:	All versions of ESAPI 2.x prior to 2.5.2.0 and all versions of ESAPI 1.x.
Details:	**Exploitable as used by ESAPI, but see discussion below for important details.**
GitHub Issue #:	None.
Related:	CVE-2023-ABCDE (TBD); no other ESAPI security bulletins.
CWE:	CWE-770 (Allocation of Resources Without Limits or Throttling)
CVE Identifier:	CVE-2023-24998
CVSS Severity (version 3.1)	CVSS v3.1 Base Score: 7.5 (High) Impact Subscore: 3.6 Exploitability Subscore: 3.9 CVSS Vector CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

1 We are waiting on feedback from the author of the ESAPI WAF. If it does not have other mechanisms that attempt to address DoS attacks, then it probably isn't worth trying to address this CVE there as the remediation is in reality not very effective. Hopefully, we will know more about that no later than April 10, 2023. Unfortunately, if Apache Commons FileUpload would have remediated this CVE so that it was secure-by-default, we wouldn't even probably have this footnote.

Background

OWASP ESAPI (the OWASP Enterprise Security API) is a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications. The ESAPI for Java library is designed to make it easier for programmers to retrofit security into existing applications. ESAPI for Java also serves as a solid foundation for new development.

One of the security controls provided by ESAPI for Java is its provision for (relatively) "safe file uploads". ESAPI leverages the functionality of Apache Commons FileUpload to achieve this functionality.

Problem Description

According to the description in NIST's National Vulnerability Database (NVD), the current description for CVE-2023-24998 states:

> "Apache Commons FileUpload before 1.5 does not limit the number of request parts to be processed resulting in the possibility of an attacker triggering a DoS with a malicious upload or series of uploads. Note that, like all of the file upload limits, the new configuration option (FileUploadBase#setFileCountMax) is not enabled by default and must be explicitly configured."

So the real question that everyone is asking is will using ESAPI leave my application code exposed to CVE-2023-24998 in a manner that makes this CVE exploitable? That is the question this analysis attempts to answer, but the TL;DR answer for those of you not interested in the details is, "Yes".

ESAPI only uses Apache Commons FileUpload all of the HTTPUtilities.getFileUploads() methods and in the ESAPI WAF class (ESAPIWebApplicationFirewallFilter), specifically an internal implementation class that the ESAPI WAF uses (InterceptingHTTPServletRequest). **Thus, if you are not using one of those, you are not even interacting with Apache Commons FileUpload at all and thus are *NOT* affected.** However, if you are using any of the HTTPUtilities.getFileUploads() on have the ESAPI WAF enabled, you likely are affected.

The (original[2]) description for CVE-2023-24998 implies that the "problem" was with the FileUploadBase class. Indeed, a cursory examination of this class between the 1.4 and 1.5 releases shows that the 1.5 version added a counter (fileCountMax) for the maximum allowed files *per single HTTP request* and if that counter is exceeded, a FileCountLimitExceededException is thrown. There were also setter / getter

2 By the time you read this, hopefully the original CVE description will have been updated as per my request to note all the affected Apache Commons FileUpload classes rather than only the single abstract class.

methods for that counter added and as noted in the CVE description one must explicitly call FileUploadBase.setFileCountMax() before doing the uploads, otherwise no upper limit is enforced. But what is easy to miss is that FileUploadBase is an *abstract* class and that in turn implies that other classes must be impacted as well. In other words, one cannot simply recursively grep one's source code for FileUploadBase to see if you are using it and if not, conclude that your code is not affected. Instead, you must also look for all these other Apache Commons FileUpload classes that are subclasses of FileUploadBase and check to see if you are using them as well. As it turns out, the Apache Commons FileUpload classes that exend FileUploadBase are:

- org.apache.commons.fileupload.diskFileUpload
- org.apache.commons.fileupload.FileUpload
- org.apache.commons.fileupload.portlet.PortletFileUpload
- org.apache.commons.fileupload.servlet.ServletFileUpload

So, minimally one needs to search for the use of any of these subclasses as well as the abstract FileUploadBase class.

As it turns out, ESAPI uses the ServletFileUpload class, which makes the DoS vulnerability described in CVE-2023-24998, exploitable in ESAPI.

Thus, our conclusion is, this CVE *is* exploitable via ESAPI. Because of this, the ESAPI team is working with GitHub Security team to file a CVE specific to ESAPI. That will allow us to make the CVE description specific to the affected classes in ESAPI.

Related Concerns

Please see CVE-2023-ABCDE (TBD) for additional details.

Impact

The underlying problem is that an attacker can keep uploading so many new uniquely named files that eventually your system resources (inodes for *nix file systems) become exhausted and no more files will be able to be able to be created on that file system. Hence the DoS aspect.

If your application is *not* using the ESAPI WAF or one of the HTTPUtilities.getFileUploads() method,s then your application are not impacted by the DoS vulnerability in CVE-2023-24998. If that is the case with any of application using ESAPI, Software Composition Analysis (SCA) tools and/or services like OWASP Dependency Check, BlackDuck, Snyk, Veracode's SourceClear, GitHub Dependabot, etc.

will continue to give you warnings that you may be required to explain to your management in order to justify continue using ESAPI in your application. And while we would like you to update to 2.5.2.0 or whatever the latest, you can use the Workarounds section to quiet your SCA tools and point your management to this Security Bulletin as justification for doing so.

However, if your applications ***are*** using one of the HTTPUtilities.getFileUploads() methods or (possibly) the ESAPI WAF, then you can be subject to a DoS attack. Specifically, a successful attack could result in an attacker creating files on your file system that you are using for uploading files until no more additional files can be added. (That is, on *nix systems, this is a DoS attack against inodes on a file system or possibly against disk quota restrictions.) I believe that Apache Commons Files should have addressed this CVE so that it was secure-by-default so that it would have been sufficient to simply upgrade to if version 1.5 or later and simply set the file count limit via a system property rather than requiring that developers make explicit calls to FileUploadBase.setFileCountMax() (which would still be allowed and would override that said system property). But regardless, the Apache Commons FileUpload patch for CVE-2023-24998 is not likely to be very effective unless it wasn't an issue in your application in the first place. To be truthful, if you wish to support file uploads at all and implement it so that it is not subject to DoS attacks related repeated uploading of numerous files, then it is not sufficient to simply add a mechanism that imposes an limit on the number of uploaded files on a per HTTP request. That upper bound must apply per authenticated user , not per anonymous HTTP request attempts. If you want to address that, I suggest using a separate independent WAF (the poor man's ESAPI WAF does not attempt to address DoS type attacks via request throttling, etc); it or perhaps a RASP solution similar to OWASP AppSensor and impose limits per user or at least rate limiting per IP address. For a determined attacker who wishes to cause a DoS attack by making your server run out of inodes or run out out of file system space, placing restrictions only on a per HTTP request basis rather than taking a more holistic approach is not going to do anything but slow them down a bit. It could even make the situation worse. If you have (say) only 20,000 inodes available and you allow only 20 files to be uploaded per request a determined DoS attacker cares little if they can use all those inodes with a single HTTP request or if then need to make 1000 HTTP requests to do so. And the 1000 HTTP requests may be worse because then may do all of those requests in parallel and spike your server's CPU resources as well. So, let's not pretend this is a real solution. It is but a Band-Aid on a potentially gushing artery. An real solution to preventing DoS attacks almost certainly is more complicated than something that can be placed in some library. But to do that, library provides such as Apache Commons FileUpload or OWASP ESAPI would have to take much stricter measures, such as requiring that users uploading files are authenticated and to do that would make that functionality much less useful to the others who had a legitimate use case for anonymous file uploads and who don't wish to consider DoS attacks against it as part of their threat model.

Workaround

If you must use an ESAPI release prior to 2.5.2.0 and you are not using any of the HTTPUtilities.getFileUploads() methods or the ESAPI WAF (see the “Problem Description” section), then you can use the following (or similar) workaround when building your project to keep your SCA tools from complaining.

First, in your application’s pom.xml, reference your dependency on the ESAPI jar in this manner, to exclude the vulnerable version of Commons FileUpload:

```
<dependency>
 <groupId>org.owasp.esapi</groupId>
 <artifactId>esapi</artifactId>
 <version>2.5.1.0</version>  <!-- Or whatever ESAPI version you are using. -->
 <exclusions>
     <exclusion>
        <groupId>commons-fileuplad</groupId>
        <artifactId>commons-fileupload</artifactId>
     </exclusion>
 </exclusions>
</dependency>
```

When you then build your project, this should exclude the Commons FileUpload jar from your classpath (Note that you do not have to specify a ‘version’ where you are *excluding* a jar.)

You can also exclude specific transitive dependencies using Gradle. If you use Gradle, follow these general instructions.

Acknowledgments

GitHub Dependabot, Snyk, Martin Bektchiev

References

https://nvd.nist.gov/vuln/detail/CVE-2023-24998