Local Information Disclosure Vulnerability in Netty on Unix-Like systems due temporary files

High **normanmaurer** published **GHSA-5mcr-gq6c-3hq2** on Feb 8, 2021

Package

io.netty:netty-codec-http (Maven)

Affected versions	Patched versions
< 4.1.59.Final	4.1.59.Final

Description

Impact

When netty's multipart decoders are used local information disclosure can occur via the local system temporary directory if temporary storing uploads on the disk is enabled.

The CVSSv3.1 score of this vulnerability is calculated to be a 6.2/10

Vulnerability Details

On unix-like systems, the temporary directory is shared between all user. As such, writing to this directory using APIs that do not explicitly set the file/directory permissions can lead to information disclosure. Of note, this does not impact modern MacOS Operating Systems.

The method `File.createTempFile` on unix-like systems creates a random file, but, by default will create this file with the permissions `-rw-r--r--`. Thus, if sensitive information is written to this file, other local users can read this information.

This is the case in netty's `AbstractDiskHttpData` is vulnerable.

netty/codec-http/src/main/java/io/netty/handler/codec/http/multipart/AbstractDiskHttpData.java
Lines 80 to 101 in e5951d4

```
80      private File tempFile() throws IOException {
81          String newpostfix;
82          String diskFilename = getDiskFilename();
83          if (diskFilename != null) {
84              newpostfix = '_' + diskFilename;
85          } else {
86              newpostfix = getPostfix();
87          }
88          File tmpFile;
89          if (getBaseDirectory() == null) {
90              // create a temporary file
91              tmpFile = File.createTempFile(getPrefix(), newpostfix);
```

`AbstractDiskHttpData` is used as a part of the `DefaultHttpDataFactory` class which is used by `HttpPostRequestDecoder` / `HttpPostMultiPartRequestDecoder`.

You may be affected by this vulnerability your project contains the following code patterns:

```
channelPipeline.addLast(new HttpPostRequestDecoder(...));


channelPipeline.addLast(new HttpPostMultiPartRequestDecoder(...));
```

Patches

This has been patched in version `4.1.59.Final`.

Workarounds

Specify your own `java.io.tmpdir` when you start the JVM or use `DefaultHttpDataFactory.setBaseDir(...)` to set the directory to something that is only readable by the current user.

References

- CWE-378: Creation of Temporary File With Insecure Permissions
- CWE-379: Creation of Temporary File in Directory with Insecure Permissions

Similar Vulnerabilities

Similar, but not the same.

- JUnit 4 - GHSA-269g-pwp5-87pp
- Google Guava - google/guava#4011
- Apache Ant - https://nvd.nist.gov/vuln/detail/CVE-2020-1945
- JetBrains Kotlin Compiler - https://nvd.nist.gov/vuln/detail/CVE-2020-15824

For more information

If you have any questions or comments about this advisory:

- Open an issue in netty
- Email us here

Original Report

Hi Netty Security Team,

I've been working on some security research leveraging custom CodeQL queries to detect local information disclosure vulnerabilities in java applications. This was the result from running this query against the netty project:
https://lgtm.com/query/7723301787255288599/

Netty contains three local information disclosure vulnerabilities, so far as I can tell.

One is here, where the private key for the certificate is written to a temporary file.

netty/handler/src/main/java/io/netty/handler/ssl/util/SelfSignedCertificate.java
Lines 316 to 346 in e5951d4

```
316     // Encode the private key into a file.
317     ByteBuf wrappedBuf = Unpooled.wrappedBuffer(key.getEncoded());
318     ByteBuf encodedBuf;
319     final String keyText;
320     try {
321         encodedBuf = Base64.encode(wrappedBuf, true);
322         try {
323             keyText = "-----BEGIN PRIVATE KEY-----\n" +
324                     encodedBuf.toString(CharsetUtil.US_ASCII) +
325                     "\n-----END PRIVATE KEY-----\n";
326         } finally {
327             encodedBuf.release();
```

One is here, where the certificate is written to a temporary file.

netty/handler/src/main/java/io/netty/handler/ssl/util/SelfSignedCertificate.java
Lines 348 to 371 in e5951d4

```
348     wrappedBuf = Unpooled.wrappedBuffer(cert.getEncoded());
349     final String certText;
350     try {
351         encodedBuf = Base64.encode(wrappedBuf, true);
352         try {
353             // Encode the certificate into a CRT file.
354             certText = "-----BEGIN CERTIFICATE-----\n" +
355                     encodedBuf.toString(CharsetUtil.US_ASCII) +
356                     "\n-----END CERTIFICATE-----\n";
357         } finally {
358             encodedBuf.release();
359         }
```

The final one is here, where the 'AbstractDiskHttpData' creates a temporary file if the getBaseDirectory() method returns null. I believe that 'AbstractDiskHttpData' is used as a part of the file upload support? If this is the case, any files uploaded would be similarly vulnerable.

netty/codec-http/src/main/java/io/netty/handler/codec/http/multipart/AbstractDiskHttpData.java
Line 91 in e5951d4

```
91      tmpFile = File.createTempFile(getPrefix(), newpostfix);
```

All of these vulnerabilities exist because `File.createTempFile(String, String)` will create a temporary file in the system temporary directory if the 'java.io.tmpdir' system property is not explicitly set. It is my understanding that when java creates a file, by default, and using this method, the permissions on that file utilize the umask. In a majority of cases, this means that the file that java creates has the permissions: `-rw-r--r--`, thus, any other local user on that system can read the contents of that file.

Impacted OS:

- Any OS where the system temporary directory is shared between multiple users. This is not the case for MacOS or Windows.

Mitigation.

Moving to the `Files` API instead will fix this vulnerability.
https://docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html#createTempFile-java.nio.file.Path-java.lang.String-java.lang.String-java.nio.file.attribute.FileAttribute...-

This API will explicitly set the posix file permissions to something safe, by default.

I recently disclosed a similar vulnerability in JUnit 4:
GHSA-269g-pwp5-87pp

If you're also curious, this vulnerability in Jetty was also mine, also involving temporary directories, but is not the same vulnerability as in this case.
GHSA-g3wg-6mcf-8jj6

I would appreciate it if we could perform disclosure of this vulnerability leveraging the GitHub security advisories feature here. GitHub has a nice credit system that I appreciate, plus the disclosures, as you can see from the sampling above, end up looking very nice.
https://github.com/netty/netty/security/advisories

This vulnerability disclosure follows Google's 90-day vulnerability disclosure policy (I'm not an employee of Google, I just like their policy). Full disclosure will occur either at the end of the 90-day deadline or whenever a patch is made widely available, whichever occurs first.

Cheers,
Jonathan Leitschuh

Severity

High

CVE ID

CVE-2021-21290

Weaknesses

No CWEs

Credits

JLLeitschuh