

[New issue](#)[Jump to bottom](#)

[SECURITY] Fix Temporary Directory Hijacking or Information Disclosure Vulnerability #1617

🔒 Closed JLLeitschuh wants to merge 1 commit into `samtools:master` from `JLLeitschuh:fix/JLL/temporary_directory_hijacking_or_temporary_directory_information_disclosure` [🔗](#)

Conversation 5 Commits 1 Checks 5 Files changed 1



JLLeitschuh commented on Jul 27

Contributor

Security Vulnerability Fix

This pull request fixes either 1.) Temporary Directory Hijacking Vulnerability, or 2.) Temporary Directory Information Disclosure Vulnerability, which existed in this project.

Preamble

The system temporary directory is shared between all users on most unix-like systems (not MacOS, or Windows). Thus, code interacting with the system temporary directory must be careful about file interactions in this directory, and must ensure that the correct file permissions are set.

This PR was generated because the following chain of calls was detected in this repository in a way that leaves this project vulnerable.

```
File.createTempFile(..) -> file.delete() -> either file.mkdir() or file.mkdirs().
```

Impact

This vulnerability can have one of two impacts depending upon which vulnerability it is.

1. Temporary Directory Information Disclosure - Information in this directory is visible to other local users, allowing a malicious actor co-resident on the same machine to view potentially sensitive files.
2. Temporary Directory Hijacking Vulnerability - Same impact as 1. above, but also, the local users can manipulate/add contents to this directory. If code is being executed out of this temporary directory, it can lead to local privilege escalation.

Temporary Directory Hijacking

This vulnerability exists because the return value from `file.mkdir()` or `file.mkdirs()` is not checked to determine if the call succeeded. Say, for example, because another local user created the directory before this process.

```
File tmpDir = File.createTempFile("temp", ".dir"); // Attacker knows the full path of the directory that will be later created
// delete the file that was created
tmpDir.delete(); // Attacker sees file is deleted and begins a race to create their own directory before the java code.
// and makes a directory of the same name
// SECURITY VULNERABILITY: Race Condition! - Attacker beats java code and now owns this directory
tmpDir.mkdirs(); // This method returns 'false' because it was unable to create the directory. No exception is thrown.
// Attacker can write any new files to this directory that they wish.
// Attacker can read any files created within this directory.
```

Other Examples

- [CVE-2021-20202](#) - Keycloak/Keycloak
- [CVE-2020-27216](#) - eclipse/jetty.project

Temporary Directory Information Disclosure

This vulnerability exists because, although the return values of `file.mkdir()` or `file.mkdirs()` are correctly checked, the permissions of the directory that is created follows the default system `umask` settings. Thus, the directory is created with everyone-readable permissions. As such, any files/directories written into this directory are viewable by all other local users on the system.

```
File tmpDir = File.createTempFile("temp", ".dir");
tmpDir.delete();
if (!tmpDir.mkdirs()) { // Guard correctly prevents temporary directory hijacking, but directory contents are everyone-readable.
    throw new IOException("Failed to create temporary directory");
}
```

Other Examples

- [CVE-2020-15250](#) - junit-team/junit
- [CVE-2021-21364](#) - swagger-api/swagger-codegen
- [CVE-2022-24823](#) - netty/netty
- [CVE-2022-24823](#) - netty/netty

The Fix

The fix has been to convert the logic above to use the following API that was introduced in Java 1.7.

```
File tmpDir = Files.createTempDirectory("temp dir").toFile();
```

The API both created the directory securely, ie with a random, non-conflicting name, with directory permissions that only allow the currently executing user to read or write the contents of this directory.



👉 Vulnerability disclosure is a super important part of the vulnerability handling process and should not be skipped! This may be completely new to you, and that's okay, I'm here to assist!

First question, do we need to perform vulnerability disclosure? It depends!

1. Is the vulnerable code only in tests or example code? No disclosure required!
2. Is the vulnerable code in code shipped to your end users? Vulnerability disclosure is probably required!

Vulnerability Disclosure How-To

You have a few options options to perform vulnerability disclosure. However, I'd like to suggest the following 2 options:

1. Request a CVE number from GitHub by creating a repository-level [GitHub Security Advisory](#). This has the advantage that, if you provide sufficient information, GitHub will automatically generate Dependabot alerts for your downstream consumers, resolving this vulnerability more quickly.
2. Reach out to the team at Snyk to assist with CVE issuance. They can be reached at the [Snyk's Disclosure Email](#).

Detecting this and Future Vulnerabilities

This vulnerability was automatically detected by GitHub's [LGTM.com](#) using this [CodeQL Query](#).

You can automatically detect future vulnerabilities like this by enabling the free (for open-source) [GitHub Action](#).

I'm not an employee of GitHub, I'm simply an open-source security researcher.

Source

This contribution was automatically generated with an [OpenRewrite refactoring recipe](#), which was lovingly hand crafted to bring this security fix to your repository.

The source code that generated this PR can be found here:

[UseFilesCreateTempDirectory](#)

Opting-Out

If you'd like to opt-out of future automated security vulnerability fixes like this, please consider adding a file called `.github/GH-ROBOTS.txt` to your repository with the line:

```
User-agent: JLLeitschuh/security-research
Disallow: *
```

This bot will respect the [ROBOTS.txt](#) format for future contributions.

Alternatively, if this project is no longer actively maintained, consider [archiving](#) the repository.

CLA Requirements

This section is only relevant if your project requires contributors to sign a Contributor License Agreement (CLA) for external contributions.

It is unlikely that I'll be able to directly sign CLAs. However, all contributed commits are already automatically signed-off.

The meaning of a signoff depends on the project, but it typically certifies that committer has the rights to submit this work under the same license and agrees to a Developer Certificate of Origin (see <https://developercertificate.org/> for more information).

- [Git Commit Signoff documentation](#)

If signing your organization's CLA is a strict-requirement for merging this contribution, please feel free to close this PR.

Sponsorship & Support

This contribution is sponsored by HUMAN Security Inc. and the new Dan Kaminsky Fellowship, a fellowship created to celebrate Dan's memory and legacy by funding open-source work that makes the world a better (and more secure) place.

This PR was generated by [Moderne](#), a free-for-open source SaaS offering that uses format-preserving AST transformations to fix bugs, standardize code style, apply best practices, migrate library versions, and fix common security vulnerabilities at scale.

Tracking

All PR's generated as part of this fix are tracked here: [JLLeitschuh/security-research#10](#)

🔍 vuln-fix: Temporary Directory Hijacking or Information Disclosure ...

✓ 269ba3f

lbergelson commented on Jul 28

Member

@JLLeitschuh Thanks for finding this and submitting a fix.

👍 1

codecov-commen... commented on Jul 28


Codecov Report

Merging [#1617](#) ([269ba3f](#)) into [master](#) ([489c419](#)) will decrease coverage by `0.030%`.
The diff coverage is `100.000%`.


	Coverage	Diff	
##	master	#1617	+/-
=====			
- Coverage	69.840%	69.810%	-0.030%
+ Complexity	9684	9680	-4


Files	703	703	
Lines	37752	37741	-11
Branches	6131	6105	-26
=====			
- Hits	26366	26347	-19
- Misses	8932	8936	+4
- Partial	2454	2458	+4

Impacted Files	Coverage Δ	
src/main/java/htsjdk/samtools/util/IOUtil.java	59.367% <100.000%> (+0.473%)	↑
...sjdk/samtools/util/htsget/HtsgetErrorResponse.java	73.333% <0.000%> (-6.667%)	↓
...tools/cram/encoding/core/GammalIntegerEncoding.java	86.667% <0.000%> (-6.667%)	↓
...mtools/cram/encoding/core/BetalIntegerEncoding.java	91.304% <0.000%> (-4.348%)	↓
...am/encoding/core/CanonicalHuffmanByteEncoding.java	52.941% <0.000%> (-2.941%)	↓
...va/htsjdk/beta/codecs/variants/vcf/VCFDecoder.java	62.651% <0.000%> (-2.410%)	↓
...va/htsjdk/samtools/util/htsget/HtsgetResponse.java	89.706% <0.000%> (-1.471%)	↓
src/main/java/htsjdk/io/AsyncWriterPool.java	72.222% <0.000%> (-1.389%)	↓
...ava/htsjdk/beta/codecs/reads/cram/CRAMDecoder.java	71.277% <0.000%> (-1.355%)	↓
...va/htsjdk/samtools/util/AsyncBufferedIterator.java	68.132% <0.000%> (-1.099%)	↓
... and 11 more		

  **Ibergelson** mentioned this pull request on Sep 23

Fix temporary directory hijacking or temporary directory information disclosure #1621

 Merged

 **Ibergelson** closed this in [4a4024a](#) on Sep 23

JLLeitschuh commented on Oct 8

Contributor Author

Hi @Ibergelson,

Do you believe this fixed a valid security vulnerability? Do you need assistance with vulnerability disclosure and CVE issuance?

Ibergelson commented on Oct 12

Member

@JLLeitschuh Thanks for following up. I think this did fix a real issue. I've never tried to issue a vulnerability disclosure before. I might bug you for help if I get stuck.

JLLeitschuh commented on Oct 13

Contributor Author

If you've never done a vulnerability disclosure before, Snyk can assist here.

CC: Benji @benjifin, Leeya @leeyashalti

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

3 participants

