CVE-2022-26612: Apache Hadoop: Arbitrary file write in FileUtil#unpackEntries on Windows

Posted to dev@hadoop.apache.org (list.html?dev@hadoop.apache.org)

Gautham Banasandra - Thursday, April 7, 2022 9:42:53 AM EDT

Severity: high

Description:

The unTar function [1] uses unTarUsingJava function on Windows and the built-in tar utility on Unix and other OSes:

```
if(Shell.WINDOWS) {
// Tar is not native to Windows. Use simple Java based implementation for
// tests and simple tar archives
unTarUsingJava(inFile, untarDir, gzipped);
}
else {
// spawn tar utility to untar archive for full fledged unix behavior such
// as resolving symlinks in tar archives
unTarUsingTar(inFile, untarDir, gzipped);
}
```

The function verifies that the extracted TAR entry is under the expected targetDirPath[2]:

```
if (!outputFile.getCanonicalPath().startsWith(targetDirPath)) {
throw new IOException("expanding " + entry.getName()
+ " would create entry outside of " + outputDir);
}
```

However it doesn't apply the same restriction to the target of an extracted symlink[3]:

```
if (entry.isSymbolicLink()) {
// Create symbolic link relative to tar parent dir
Files.createSymbolicLink(FileSystems.getDefault()
.getPath(outputDir.getPath(), entry.getName()),
FileSystems.getDefault().getPath(entry.getLinkName()));
return;
}
```

As a result, a TAR entry may create a symlink under the expected extraction directory which points to an external directory. A subsequent TAR entry may extract an arbitrary file into the external directory using the symlink name. This however would be caught by the same targetDirPath[4] check on Unix because of the getCanonicalPath call. However on Windows, getCanonicalPath doesn't resolve symbolic links, which bypasses the check.

unpackEntries during TAR extraction follows symbolic links which allows writing outside expected base directory on Windows.

[1]=https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-pr

oject/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L850 (https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L850)
[2]=https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L964-L967 (https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L964-L967)
[3]=https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L983-L989 (https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L983-L989)
[4]=https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L964-L967 (https://github.com/apache/hadoop/blob/125e3b616040b4f98956aa946cc51e99f7d596c2/hadoop-common-project/hadoop-common/src/main/java/org/apache/hadoop/fs/FileUtil.java#L964-L967)

Credit:

This issue was reported by a member of GitHub Security Lab, Jaroslav Lobačevski (https://github.com/JarLob (https://github.com/JarLob)).
