# CWE Detail – CWE-1386

## Description

The product opens a file or directory, but it does not properly prevent the name from being associated with a junction or mount point to a destination that is outside of the intended control sphere.

## Extended Description

Depending on the intended action
 being performed, this could allow an
 attacker to cause the product to read,
 write, delete, or otherwise operate on
 unauthorized files. In Windows, NTFS5 allows for file
 system objects called reparse points.
 Applications can create a hard link from one
 directory to another directory, called a
 junction point. They can also create a
 mapping from a directory to a drive letter,
 called a mount point. If a file is used by a
 privileged program, but it can be replaced
 with a hard link to a sensitive file (e.g.,
 AUTOEXEC.BAT), an attacker could excalate
 privileges. When the process opens the file,
 the attacker can assume the privileges of
 that process, tricking the privileged
 process to read, modify, or delete the
 sensitive file, preventing the program from
 accurately processing data. Note that one
 can also point to registries and
 semaphores.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2021-26426: Privileged service allows attackers to delete unauthorized files using a directory junction, leading to arbitrary code execution as SYSTEM.

**•** CVE-2020-0863: By creating a mount point and hard links, an attacker can abuse a service to allow users arbitrary file read permissions.

**•** CVE-2019-1161: Chain: race condition (CWE-362) in anti-malware product allows deletion of files by creating a junction (CWE-1386) and using hard links during the time window in which a temporary file is created and deleted.

**•** CVE-2014-0568: Escape from sandbox for document reader by using a mountpoint [REF-1264]

## Modes of Introduction

**•** Implementation: The developer might not consider that when a program in Windows operates with different permissions than the executing user, the use of links, mount points, and junctions might cause the program to access files or directories that are outside of the intended storage location.

## Common Consequences

**•** Impact: Read Files or Directories — Notes: Read arbitrary files by replacing a user-controlled folder with a mount point and additional hard links.

**•** Impact: Modify Files or Directories — Notes: Modify an arbitrary file by replacing the rollback files in installer directories, as they can have the installer execute those rollbacks.

**•** Impact: Modify Files or Directories — Notes: Even if there is no control of contents, an arbitrary file delete or overwrite (when running as SYSTEM or admin) can be used for a permanent system denial-of-service, e.g. by deleting a startup configuration file that prevents the service from starting.

## Potential Mitigations

**•** Architecture and Design: When designing software that will have different rights than the executer, the software should check that files that it is interacting with are not improper hard links or mount points. One way to do this in Windows is to use the functionality embedded in the following command: "dir /al /s /b" or, in PowerShell, use LinkType as a filter. In addition, some software uses authentication via signing to ensure that the file is the correct one to use. Make checks atomic with the file action, otherwise a TOCTOU weakness (CWE-367) can be introduced. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Notes

**•** Terminology: Symbolic links, hard links, junctions, and mount points can be confusing terminology, as there are differences in how they operate between UNIX-based systems and Windows, and there are interactions between them.

**•** Maintenance: This entry is still under development and will continue to see updates and content improvements.