# CWE Detail – CWE-61

## Description

The product, when opening a file or directory, does not sufficiently account for when the file is a symbolic link that resolves to a target outside of the intended control sphere. This could allow an attacker to cause the product to operate on unauthorized files.

## Extended Description

A product that allows UNIX symbolic links (symlink) as part of paths whether in internal code or through user input can allow an attacker to spoof the symbolic link and traverse the file system to unintended locations or access arbitrary files. The symbolic link can permit an attacker to read/write/corrupt a file that they originally did not have permissions to access.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-1999-1386: Some versions of Perl follow symbolic links when running with the -e option, which allows local users to overwrite arbitrary files via a symlink attack.

**•** CVE-2000-1178: Text editor follows symbolic links when creating a rescue copy during an abnormal exit, which allows local users to overwrite the files of other users.

**•** CVE-2004-0217: Antivirus update allows local users to create or append to arbitrary files via a symlink attack on a logfile.

**•** CVE-2003-0517: Symlink attack allows local users to overwrite files.

**•** CVE-2004-0689: Possible interesting example

**•** CVE-2005-1879: Second-order symlink vulnerabilities

**•** CVE-2005-1880: Second-order symlink vulnerabilities

**•** CVE-2005-1916: Symlink in Python program

**•** CVE-2000-0972: Setuid product allows file reading by replacing a file being edited with a symlink to the targeted file, leaking the result in error messages when parsing fails.

**•** CVE-2005-0824: Signal causes a dump that follows symlinks.

**•** CVE-2015-3629: A Libcontainer used in Docker Engine allows local users to escape containerization and write to an arbitrary file on the host system via a symlink attack in an image when respawning a container.

**•** CVE-2020-26277: In a MySQL database deployment tool, users may craft a maliciously packaged tarball that contains symlinks to files external to the target and once unpacked, will execute.

**•** CVE-2021-21272: "Zip Slip" vulnerability in Go-based Open Container Initiative (OCI) registries product allows writing arbitrary files outside intended directory via symbolic links or hard links in a gzipped tarball.

## Related Attack Patterns (CAPEC)

* CAPEC-27

## Modes of Introduction

**•** Implementation: These are typically reported for temporary files or privileged programs.

## Common Consequences

**•** Impact: Read Files or Directories, Modify Files or Directories — Notes:

## Potential Mitigations

**•** Implementation: Symbolic link attacks often occur when a program creates a tmp directory that stores files/links. Access to the directory should be restricted to the program as to prevent attackers from manipulating the files. (Effectiveness: N/A)

**•** Architecture and Design: Follow the principle of least privilege when assigning access rights to entities in a software system. Denying access to a file can prevent an attacker from replacing that file with a link to a sensitive file. Ensure good compartmentalization in the system to provide protected areas that can be trusted. (Effectiveness: N/A)

## Applicable Platforms

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

## Notes

**•** Research Gap: Symlink vulnerabilities are regularly found in C and shell programs, but all programming languages can have this problem. Even shell programs are probably under-reported. "Second-order symlink vulnerabilities" may exist in programs that invoke other programs that follow symlinks. They are rarely reported but are likely to be fairly common when process invocation is used [REF-493].