# CWE Detail – CWE-59

## Description

The product attempts to access a file based on the filename, but it does not properly prevent that filename from identifying a link or shortcut that resolves to an unintended resource.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## 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: Window manager does not properly handle when certain symbolic links point to "stale" locations, which could allow local users to create or truncate arbitrary files.

**•** 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-2001-1494: Hard link attack, file overwrite; interesting because program checks against soft links

**•** CVE-2002-0793: Hard link and possibly symbolic link following vulnerabilities in embedded operating system allow local users to overwrite arbitrary files.

**•** CVE-2003-0578: Server creates hard links and unlinks files as root, which allows local users to gain privileges by deleting and overwriting arbitrary files.

**•** CVE-1999-0783: Operating system allows local users to conduct a denial of service by creating a hard link from a device special file to a file on an NFS file system.

**•** CVE-2004-1603: Web hosting manager follows hard links, which allows local users to read or modify arbitrary files.

**•** CVE-2004-1901: Package listing system allows local users to overwrite arbitrary files via a hard link attack on the lockfiles.

**•** CVE-2005-1111: Hard link race condition

**•** CVE-2000-0342: Mail client allows remote attackers to bypass the user warning for executable attachments such as .exe, .com, and .bat by using a .lnk file that refers to the attachment, aka "Stealth Attachment."

**•** CVE-2001-1042: FTP server allows remote attackers to read arbitrary files and directories by uploading a .lnk (link) file that points to the target file.

**•** CVE-2001-1043: FTP server allows remote attackers to read arbitrary files and directories by uploading a .lnk (link) file that points to the target file.

**•** CVE-2005-0587: Browser allows remote malicious web sites to overwrite arbitrary files by tricking the user into downloading a .LNK (link) file twice, which overwrites the file that was referenced in the first .LNK file.

**•** CVE-2001-1386: ".LNK." - .LNK with trailing dot

**•** CVE-2003-1233: Rootkits can bypass file access restrictions to Windows kernel directories using NtCreateSymbolicLinkObject function to create symbolic link

**•** CVE-2002-0725: File system allows local attackers to hide file usage activities via a hard link to the target file, which causes the link to be recorded in the audit trail instead of the target file.

**•** CVE-2003-0844: Web server plugin allows local users to overwrite arbitrary files via a symlink attack on predictable temporary filenames.

**•** 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-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.

**•** CVE-2020-27833: "Zip Slip" vulnerability in container management product allows writing arbitrary files outside intended directory via a container image (.tar format) with filenames that are symbolic links that point to other files within the same tar file; however, the files being pointed to can also be symbolic links to destinations outside the intended directory, bypassing the initial check.

## Related Attack Patterns (CAPEC)

* CAPEC-132
* CAPEC-17
* CAPEC-35
* CAPEC-76

## Attack TTPs

**•** T1027.009: Embedded Payloads (Tactics: defense-evasion)

**•** T1547.009: Shortcut Modification (Tactics: persistence, privilege-escalation)

**•** T1574.005: Executable Installer File Permissions Weakness (Tactics: persistence, privilege-escalation, defense-evasion)

**•** T1574.010: Services File Permissions Weakness (Tactics: persistence, privilege-escalation, defense-evasion)

**•** T1564.009: Resource Forking (Tactics: defense-evasion)

**•** T1027.006: HTML Smuggling (Tactics: defense-evasion)

## Modes of Introduction

**•** Implementation: REALIZATION: This weakness is caused during implementation of an architectural security tactic.

## Common Consequences

**•** Impact: Read Files or Directories, Modify Files or Directories, Bypass Protection Mechanism — Notes: An attacker may be able to traverse the file system to unintended locations and read or overwrite the contents of unexpected files. If the files are used for a security mechanism then an attacker may be able to bypass the mechanism.

**•** Impact: Execute Unauthorized Code or Commands — Notes: Windows simple shortcuts, sometimes referred to as soft links, can be exploited remotely since a ".LNK" file can be uploaded like a normal file. This can enable remote execution.

## Potential Mitigations

**•** 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

**•** Theoretical: Link following vulnerabilities are Multi-factor Vulnerabilities (MFV). They are the combination of multiple elements: file or directory permissions, filename predictability, race conditions, and in some cases, a design limitation in which there is no mechanism for performing atomic file creation operations. Some potential factors are race conditions, permissions, and predictability.