# CWE Detail – CWE-271

## Description

The product does not drop privileges before passing control of a resource to an actor that does not have those privileges.

## Extended Description

In some contexts, a system executing with elevated permissions will hand off a process/file/etc. to another process or user. If the privileges of an entity are not reduced, then elevated privileges are spread throughout a system and possibly to an attacker.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2000-1213: Program does not drop privileges after acquiring the raw socket.

**•** CVE-2001-0559: Setuid program does not drop privileges after a parsing error occurs, then calls another program to handle the error.

**•** CVE-2001-0787: Does not drop privileges in related groups when lowering privileges.

**•** CVE-2002-0080: Does not drop privileges in related groups when lowering privileges.

**•** CVE-2001-1029: Does not drop privileges before determining access to certain files.

**•** CVE-1999-0813: Finger daemon does not drop privileges when executing programs on behalf of the user being fingered.

**•** CVE-1999-1326: FTP server does not drop privileges if a connection is aborted during file transfer.

**•** CVE-2000-0172: Program only uses seteuid to drop privileges.

**•** CVE-2004-2504: Windows program running as SYSTEM does not drop privileges before executing other programs (many others like this, especially involving the Help facility).

**•** CVE-2004-0213: Utility Manager launches winhlp32.exe while running with raised privileges, which allows local users to gain system privileges.

**•** CVE-2004-0806: Setuid program does not drop privileges before executing program specified in an environment variable.

**•** CVE-2004-0828: Setuid program does not drop privileges before processing file specified on command line.

**•** CVE-2004-2070: Service on Windows does not drop privileges before using "view file" option, allowing code execution.

## Modes of Introduction

**•** Architecture and Design: N/A

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

**•** Operation: N/A

## Common Consequences

**•** Impact: Gain Privileges or Assume Identity — Notes: If privileges are not dropped, neither are access rights of the user. Often these rights can be prevented from being dropped.

**•** Impact: Gain Privileges or Assume Identity, Hide Activities — Notes: If privileges are not dropped, in some cases the system may record actions as the user which is being impersonated rather than the impersonator.

## Potential Mitigations

**•** Architecture and Design: Compartmentalize the system to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area. Ensure that appropriate compartmentalization is built into the system design, and the compartmentalization allows for and reinforces privilege separation functionality. Architects and designers should rely on the principle of least privilege to decide the appropriate time to use privileges and the time to drop privileges. (Effectiveness: N/A)

**•** Architecture and Design: Very carefully manage the setting, management, and handling of privileges. Explicitly manage trust zones in the software. (Effectiveness: N/A)

**•** Architecture and Design: Consider following the principle of separation of privilege. Require multiple conditions to be met before permitting access to a system resource. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** Constraining the process inside the application's home directory before opening any files is a valuable security measure. However, the absence of a call to setuid() with some non-zero value means the application is continuing to operate with unnecessary root privileges. Any successful exploit carried out by an attacker against the application can now result in a privilege escalation attack because any malicious operations will be performed with the privileges of the superuser. If the application drops to the privilege level of a non-root user, the potential for damage is substantially reduced.

## Notes

**•** Maintenance: CWE-271, CWE-272, and CWE-250 are all closely related and possibly overlapping. CWE-271 is probably better suited as a category.