# CWE Detail – CWE-273

## Description

The product attempts to drop privileges but does not check or incorrectly checks to see if the drop succeeded.

## Extended Description

If the drop fails, the product will continue to run with the raised privileges, which might provide additional access to unprivileged users.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2006-4447: Program does not check return value when invoking functions to drop privileges, which could leave users with higher privileges than expected by forcing those functions to fail.

**•** CVE-2006-2916: Program does not check return value when invoking functions to drop privileges, which could leave users with higher privileges than expected by forcing those functions to fail.

## Modes of Introduction

**•** Implementation: REALIZATION: This weakness is caused during implementation of an architectural security tactic. This issue is likely to occur in restrictive environments in which the operating system or application provides fine-grained control over privilege management.

## 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)

**•** Implementation: Check the results of all functions that return a value and verify that the value is expected. (Effectiveness: High)

**•** Implementation: In Windows, make sure that the process token has the SeImpersonatePrivilege(Microsoft Server 2003). Code that relies on impersonation for security must ensure that the impersonation succeeded, i.e., that a proper privilege demotion happened. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** The call to ImpersonateNamedPipeClient may fail, but the return value is not checked. If the call fails, the code may execute with higher privileges than intended. In this case, an attacker could exploit this behavior to write a file to a location that the attacker does not have access to.