# CWE Detail – CWE-270

## Description

The product does not properly manage privileges while it is switching between different contexts that have different privileges or spheres of control.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2002-1688: Web browser cross domain problem when user hits "back" button.

**•** CVE-2003-1026: Web browser cross domain problem when user hits "back" button.

**•** CVE-2002-1770: Cross-domain issue - third party product passes code to web browser, which executes it in unsafe zone.

**•** CVE-2005-2263: Run callback in different security context after it has been changed from untrusted to trusted. \* note that "context switch before actions are completed" is one type of problem that happens frequently, espec. in browsers.

## Related Attack Patterns (CAPEC)

* CAPEC-17
* CAPEC-30
* CAPEC-35

## Attack TTPs

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

**•** T1055.003: Thread Execution Hijacking (Tactics: defense-evasion, 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

**•** 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: A user can assume the identity of another user with separate privileges in another context. This will give the user unauthorized access that may allow them to acquire the access information of other users.

## Potential Mitigations

**•** 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: Run your code using the lowest privileges that are required to accomplish the necessary tasks [REF-76]. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations. (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)

## Notes

**•** Research Gap: This concept needs more study.