# CWE Detail – CWE-15

## Description

One or more system settings or configuration elements can be externally controlled by a user.

## Extended Description

Allowing external control of system settings can disrupt service or cause an application to behave in unexpected, and potentially malicious ways.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Related Attack Patterns (CAPEC)

* CAPEC-13
* CAPEC-146
* CAPEC-176
* CAPEC-203
* CAPEC-270
* CAPEC-271
* CAPEC-579
* CAPEC-69
* CAPEC-76
* CAPEC-77

## Attack TTPs

**•** T1574.007: Path Interception by PATH Environment Variable (Tactics: persistence, privilege-escalation, defense-evasion)

**•** T1547.014: Active Setup (Tactics: persistence, privilege-escalation)

**•** T1112: Modify Registry (Tactics: defense-evasion, persistence)

**•** T1574.006: Dynamic Linker Hijacking (Tactics: persistence, privilege-escalation, defense-evasion)

**•** T1547.004: Winlogon Helper DLL (Tactics: persistence, privilege-escalation)

**•** T1647: Plist File Modification (Tactics: defense-evasion)

**•** T1562.003: Impair Command History Logging (Tactics: defense-evasion)

**•** T1547.001: Registry Run Keys / Startup Folder (Tactics: persistence, privilege-escalation)

## Modes of Introduction

**•** Implementation: Setting manipulation vulnerabilities occur when an attacker can control values that govern the behavior of the system, manage specific resources, or in some way affect the functionality of the application.

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

## Common Consequences

**•** Impact: Varies by Context — Notes:

## 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: Because setting manipulation covers a diverse set of functions, any attempt at illustrating it will inevitably be incomplete. Rather than searching for a tight-knit relationship between the functions addressed in the setting manipulation category, take a step back and consider the sorts of system values that an attacker should not be allowed to control. (Effectiveness: N/A)

**•** Implementation: In general, do not allow user-provided or otherwise untrusted data to control sensitive values. The leverage that an attacker gains by controlling these values is not always immediately obvious, but do not underestimate the creativity of the attacker. (Effectiveness: N/A)

## Demonstrative Examples

**•** Although a process must be privileged to successfully invoke sethostid(), unprivileged users may be able to invoke the program. The code in this example allows user input to directly control the value of a system setting. If an attacker provides a malicious value for host ID, the attacker can misidentify the affected machine on the network or cause other unintended behavior.

**•** In this example, an attacker could cause an error by providing a nonexistent catalog name or connect to an unauthorized portion of the database.