# CWE Detail – CWE-637

## Description

The product uses a more complex mechanism than necessary, which could lead to resultant weaknesses when the mechanism is not correctly understood, modeled, configured, implemented, or used.

## Extended Description

Security mechanisms should be as simple as possible. Complex security mechanisms may engender partial implementations and compatibility problems, with resulting mismatches in assumptions and implemented security. A corollary of this principle is that data specifications should be as simple as possible, because complex data specifications result in complex validation code. Complex tasks and systems may also need to be guarded by complex security checks, so simple systems should be preferred.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2007-6067: Support for complex regular expressions leads to a resultant algorithmic complexity weakness (CWE-407).

**•** CVE-2007-1552: Either a filename extension and a Content-Type header could be used to infer the file type, but the developer only checks the Content-Type, enabling unrestricted file upload (CWE-434).

**•** CVE-2007-6479: In Apache environments, a "filename.php.gif" can be redirected to the PHP interpreter instead of being sent as an image/gif directly to the user. Not knowing this, the developer only checks the last extension of a submitted filename, enabling arbitrary code execution.

**•** CVE-2005-2148: The developer cleanses the $\_REQUEST superglobal array, but PHP also populates $\_GET, allowing attackers to bypass the protection mechanism and conduct SQL injection attacks against code that uses $\_GET.

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

**•** Operation: N/A

## Common Consequences

**•** Impact: Other — Notes:

## Potential Mitigations

**•** Architecture and Design: Avoid complex security mechanisms when simpler ones would meet requirements. Avoid complex data models, and unnecessarily complex operations. Adopt architectures that provide guarantees, simplify understanding through elegance and abstraction, and that can be implemented similarly. Modularize, isolate and do not trust complex code, and apply other secure programming principles on these modules (e.g., least privilege) to mitigate vulnerabilities. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** N/A

**•** N/A