# CWE Detail – CWE-470

## Description

The product uses external input with reflection to select which classes or code to use, but it does not sufficiently prevent the input from selecting improper classes or code.

## Extended Description

If the product uses external inputs to determine which class to instantiate or which method to invoke, then an attacker could supply values to select unexpected classes or methods. If this occurs, then the attacker could create control flow paths that were not intended by the developer. These paths could bypass authentication or access control checks, or otherwise cause the product to behave in an unexpected manner. This situation becomes a doomsday scenario if the attacker can upload files into a location that appears on the product's classpath (CWE-427) or add new entries to the product's classpath (CWE-426). Under either of these conditions, the attacker can use reflection to introduce new, malicious behavior into the product.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2018-1000613: Cryptography API uses unsafe reflection when deserializing a private key

**•** CVE-2004-2331: Database system allows attackers to bypass sandbox restrictions by using the Reflection API.

## Related Attack Patterns (CAPEC)

* CAPEC-138

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

## Common Consequences

**•** Impact: Execute Unauthorized Code or Commands, Alter Execution Logic — Notes: The attacker might be able to execute code that is not directly accessible to the attacker. Alternately, the attacker could call unexpected code in the wrong place or the wrong time, possibly modifying critical system state.

**•** Impact: DoS: Crash, Exit, or Restart, Other — Notes: The attacker might be able to use reflection to call the wrong code, possibly with unexpected arguments that violate the API (CWE-227). This could cause the product to exit or hang.

**•** Impact: Read Application Data — Notes: By causing the wrong code to be invoked, the attacker might be able to trigger a runtime error that leaks sensitive information in the error message, such as CWE-536.

## Potential Mitigations

**•** Architecture and Design: Refactor your code to avoid using reflection. (Effectiveness: N/A)

**•** Architecture and Design: Do not use user-controlled inputs to select and load classes or code. (Effectiveness: N/A)

**•** Implementation: Apply strict input validation by using allowlists or indirect selection to ensure that the user is only selecting allowable classes or code. (Effectiveness: N/A)

## Applicable Platforms

**•** Java (Class: None, Prevalence: Undetermined)

**•** PHP (Class: None, Prevalence: Undetermined)

**•** None (Class: Interpreted, Prevalence: Sometimes)

## Demonstrative Examples

**•** A programmer might refactor this code to use reflection as follows: