# CWE Detail – CWE-643

## Description

The product uses external input to dynamically construct an XPath expression used to retrieve data from an XML database, but it does not neutralize or incorrectly neutralizes that input. This allows an attacker to control the structure of the query.

## Extended Description

The net effect is that the attacker will have control over the information selected from the XML database and may use that ability to control application flow, modify logic, retrieve unauthorized data, or bypass important checks (e.g. authentication).

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Modes of Introduction

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

## Common Consequences

**•** Impact: Bypass Protection Mechanism — Notes: Controlling application flow (e.g. bypassing authentication).

**•** Impact: Read Application Data — Notes: The attacker could read restricted XML content.

## Potential Mitigations

**•** Implementation: Use parameterized XPath queries (e.g. using XQuery). This will help ensure separation between data plane and control plane. (Effectiveness: N/A)

**•** Implementation: Properly validate user input. Reject data where appropriate, filter where appropriate and escape where appropriate. Make sure input that will be used in XPath queries is safe in that context. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** The Java code used to retrieve the home directory based on the provided credentials is:

## Notes

**•** Relationship: This weakness is similar to other weaknesses that enable injection style attacks, such as SQL injection, command injection and LDAP injection. The main difference is that the target of attack here is the XML database.