# CWE Detail – CWE-182

## Description

The product filters data in a way that causes it to be reduced or "collapsed" into an unsafe value that violates an expected security property.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2004-0815: "/.////" in pathname collapses to absolute path.

**•** CVE-2005-3123: "/.//..//////././" is collapsed into "/.././" after ".." and "//" sequences are removed.

**•** CVE-2002-0325: ".../...//" collapsed to "..." due to removal of "./" in web server.

**•** CVE-2002-0784: chain: HTTP server protects against ".." but allows "." variants such as "////./../.../". If the server removes "/.." sequences, the result would collapse into an unsafe value "////../" (CWE-182).

**•** CVE-2005-2169: MFV. Regular expression intended to protect against directory traversal reduces ".../...//" to "../".

**•** CVE-2001-1157: XSS protection mechanism strips a <script> sequence that is nested in another <script> sequence.

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Bypass Protection Mechanism — Notes:

## Potential Mitigations

**•** Architecture and Design: Avoid making decisions based on names of resources (e.g. files) if those resources can have alternate names. (Effectiveness: N/A)

**•** Implementation: Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. (Effectiveness: N/A)

**•** Implementation: Inputs should be decoded and canonicalized to the application's current internal representation before being validated (CWE-180). Make sure that the application does not decode the same input twice (CWE-174). Such errors could be used to bypass allowlist validation schemes by introducing dangerous inputs after they have been checked. (Effectiveness: N/A)

**•** N/A: Canonicalize the name to match that of the file system's representation of the name. This can sometimes be achieved with an available API (e.g. in Win32 the GetFullPathName function). (Effectiveness: N/A)

## Applicable Platforms

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

## Notes

**•** Relationship: Overlaps regular expressions, although an implementation might not necessarily use regexp's.