# CWE Detail – CWE-158

## Description

The product receives input from an upstream component, but it does not neutralize or incorrectly neutralizes NUL characters or null bytes when they are sent to a downstream component.

## Extended Description

As data is parsed, an injected NUL character or null byte may cause the product to believe the input is terminated earlier than it actually is, or otherwise cause the input to be misinterpreted. This could then be used to inject potentially dangerous input that occurs after the null byte or otherwise bypass validation routines and other protection mechanisms.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2008-1284: NUL byte in theme name causes directory traversal impact to be worse

**•** CVE-2005-2008: Source code disclosure using trailing null.

**•** CVE-2005-3293: Source code disclosure using trailing null.

**•** CVE-2005-2061: Trailing null allows file include.

**•** CVE-2002-1774: Null character in MIME header allows detection bypass.

**•** CVE-2000-0149: Web server allows remote attackers to view the source code for CGI programs via a null character (%00) at the end of a URL.

**•** CVE-2000-0671: Web server earlier allows allows remote attackers to bypass access restrictions, list directory contents, and read source code by inserting a null character (%00) in the URL.

**•** CVE-2001-0738: Logging system allows an attacker to cause a denial of service (hang) by causing null bytes to be placed in log messages.

**•** CVE-2001-1140: Web server allows source code for executable programs to be read via a null character (%00) at the end of a request.

**•** CVE-2002-1031: Protection mechanism for limiting file access can be bypassed using a null character (%00) at the end of the directory name.

**•** CVE-2002-1025: Application server allows remote attackers to read JSP source code via an encoded null byte in an HTTP GET request, which causes the server to send the .JSP file unparsed.

**•** CVE-2003-0768: XSS protection mechanism only checks for sequences with an alphabetical character following a (<), so a non-alphabetical or null character (%00) following a < may be processed.

**•** CVE-2004-0189: Decoding function in proxy allows regular expression bypass in ACLs via URLs with null characters.

**•** CVE-2005-3153: Null byte bypasses PHP regexp check (interaction error).

**•** CVE-2005-4155: Null byte bypasses PHP regexp check (interaction error).

## Related Attack Patterns (CAPEC)

* CAPEC-52
* CAPEC-53

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Unexpected State — Notes:

## Potential Mitigations

**•** N/A: Developers should anticipate that null characters or null bytes will be injected/removed/manipulated in the input vectors of their product. Use an appropriate combination of denylists and allowlists to ensure only valid, expected and appropriate input is processed by the system. (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)

## Applicable Platforms

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

## Notes

**•** Relationship: This can be a factor in multiple interpretation errors, other interaction errors, filename equivalence, etc.