# CWE Detail – CWE-184

## Description

The product implements a protection mechanism that relies on a list of inputs (or properties of inputs) that are not allowed by policy or otherwise require other action to neutralize before additional processing takes place, but the list is incomplete.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2024-4315: Chain: API for text generation using Large Language Models (LLMs) does
 not include the "\" Windows folder separator in its denylist (CWE-184)
 when attempting to prevent Local File Inclusion via path traversal
 (CWE-22), allowing deletion of arbitrary files on Windows systems.

**•** CVE-2008-2309: product uses a denylist to identify potentially dangerous content, allowing attacker to bypass a warning

**•** CVE-2005-2782: PHP remote file inclusion in web application that filters "http" and "https" URLs, but not "ftp".

**•** CVE-2004-0542: Programming language does not filter certain shell metacharacters in Windows environment.

**•** CVE-2004-0595: XSS filter doesn't filter null characters before looking for dangerous tags, which are ignored by web browsers. MIE and validate-before-cleanse.

**•** CVE-2005-3287: Web-based mail product doesn't restrict dangerous extensions such as ASPX on a web server, even though others are prohibited.

**•** CVE-2004-2351: Resultant XSS when only <script> and <style> are checked.

**•** CVE-2005-2959: Privileged program does not clear sensitive environment variables that are used by bash. Overlaps multiple interpretation error.

**•** CVE-2005-1824: SQL injection protection scheme does not quote the "\" special character.

**•** CVE-2005-2184: Detection of risky filename extensions prevents users from automatically executing .EXE files, but .LNK is accepted, allowing resultant Windows symbolic link.

**•** CVE-2007-1343: Product uses list of protected variables, but accidentally omits one dangerous variable, allowing external modification

**•** CVE-2007-5727: Chain: product only removes SCRIPT tags (CWE-184), enabling XSS (CWE-79)

**•** CVE-2006-4308: Chain: product only checks for use of "javascript:" tag (CWE-184), allowing XSS (CWE-79) using other tags

**•** CVE-2007-3572: Chain: OS command injection (CWE-78) enabled by using an unexpected character that is not explicitly disallowed (CWE-184)

**•** CVE-2002-0661: "\" not in list of disallowed values for web server, allowing path traversal attacks when the server is run on Windows and other OSes.

## Related Attack Patterns (CAPEC)

* CAPEC-120
* CAPEC-15
* CAPEC-182
* CAPEC-3
* CAPEC-43
* CAPEC-6
* CAPEC-71
* CAPEC-73
* CAPEC-85

## Modes of Introduction

**•** Implementation: Developers often try to protect their products against malicious input by checking against lists of known bad inputs, such as special characters that can invoke new commands. However, such lists often only address the most well-known bad inputs. As a quick fix, developers might rely on these lists instead of addressing the root cause of the issue. See [REF-141].

**•** Architecture and Design: The design might rely solely on detection of malicious inputs as a protection mechanism.

## Common Consequences

**•** Impact: Bypass Protection Mechanism — Notes: Attackers may be able to find other malicious inputs that were not expected by the developer, allowing them to bypass the intended protection mechanism.

## Potential Mitigations

**•** Implementation: Do not rely exclusively on detecting disallowed inputs. There are too many variants to encode a character, especially when different environments are used, so there is a high likelihood of missing some variants. Only use detection of disallowed inputs as a mechanism for detecting suspicious activity. Ensure that you are using other protection mechanisms that only identify "good" input - such as lists of allowed inputs - and ensure that you are properly encoding your outputs. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** Because the code only checks for the lower-case "script" string, it can be easily defeated with upper-case script tags.

## Notes

**•** Relationship: Multiple interpretation errors can indirectly introduce inputs that should be disallowed. For example, a list of dangerous shell metacharacters might not include a metacharacter that only has meaning in one particular shell, not all of them; or a check for XSS manipulations might ignore an unusual construct that is supported by one web browser, but not others.