# CWE Detail – CWE-178

## Description

The product does not properly account for differences in case sensitivity when accessing or determining the properties of a resource, leading to inconsistent results.

## Extended Description

Improperly handled case sensitive data can lead to several possible consequences, including: case-insensitive passwords reducing the size of the key space, making brute force attacks easier bypassing filters or access controls using alternate names multiple interpretation errors using alternate names.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

**•** CVE-2000-0499: Application server allows attackers to bypass execution of a jsp page and read the source code using an upper case JSP extension in the request.

**•** CVE-2000-0497: The server is case sensitive, so filetype handlers treat .jsp and .JSP as different extensions. JSP source code may be read because .JSP defaults to the filetype "text".

**•** CVE-2000-0498: The server is case sensitive, so filetype handlers treat .jsp and .JSP as different extensions. JSP source code may be read because .JSP defaults to the filetype "text".

**•** CVE-2001-0766: A URL that contains some characters whose case is not matched by the server's filters may bypass access restrictions because the case-insensitive file system will then handle the request after it bypasses the case sensitive filter.

**•** CVE-2001-0795: Server allows remote attackers to obtain source code of CGI scripts via URLs that contain MS-DOS conventions such as (1) upper case letters or (2) 8.3 file names.

**•** CVE-2001-1238: Task Manager does not allow local users to end processes with uppercase letters named (1) winlogon.exe, (2) csrss.exe, (3) smss.exe and (4) services.exe via the Process tab which could allow local users to install Trojan horses that cannot be stopped.

**•** CVE-2003-0411: chain: Code was ported from a case-sensitive Unix platform to a case-insensitive Windows platform where filetype handlers treat .jsp and .JSP as different extensions. JSP source code may be read because .JSP defaults to the filetype "text".

**•** CVE-2002-0485: Leads to interpretation error

**•** CVE-1999-0239: Directories may be listed because lower case web requests are not properly handled by the server.

**•** CVE-2005-0269: File extension check in forum software only verifies extensions that contain all lowercase letters, which allows remote attackers to upload arbitrary files via file extensions that include uppercase letters.

**•** CVE-2004-1083: Web server restricts access to files in a case sensitive manner, but the filesystem accesses files in a case insensitive manner, which allows remote attackers to read privileged files using alternate capitalization.

**•** CVE-2002-2119: Case insensitive passwords lead to search space reduction.

**•** CVE-2004-2214: HTTP server allows bypass of access restrictions using URIs with mixed case.

**•** CVE-2004-2154: Mixed upper/lowercase allows bypass of ACLs.

**•** CVE-2005-4509: Bypass malicious script detection by using tokens that aren't case sensitive.

**•** CVE-2002-1820: Mixed case problem allows "admin" to have "Admin" rights (alternate name property).

**•** CVE-2007-3365: Chain: uppercase file extensions causes web server to return script source code instead of executing the script.

**•** CVE-2021-39155: Chain: A microservice integration and management platform compares the hostname in the HTTP Host header in a case-sensitive way (CWE-178, CWE-1289), allowing bypass of the authorization policy (CWE-863) using a hostname with mixed case or other variations.

## 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)

## Applicable Platforms

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

## Demonstrative Examples

**•** The code only works when the "script" tag is in all lower-case, forming an incomplete denylist (CWE-184). Equivalent tags such as "SCRIPT" or "ScRiPt" will not be neutralized by this method, allowing an XSS attack.

## Notes

**•** Research Gap: These are probably under-studied in Windows and Mac environments, where file names are case-insensitive and thus are subject to equivalence manipulations involving case.