# CWE Detail – CWE-179

## Description

The product validates input before applying protection mechanisms that modify the input, which could allow an attacker to bypass the validation via dangerous inputs that only arise after the modification.

## Extended Description

Product needs to validate data at the proper time, after data has been canonicalized and cleansed. Early validation is susceptible to various manipulations that result in dangerous inputs that are produced by canonicalization and cleansing.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2002-0433: Product allows remote attackers to view restricted files via an HTTP request containing a "\*" (wildcard or asterisk) character.

**•** CVE-2003-0332: Product modifies the first two letters of a filename extension after performing a security check, which allows remote attackers to bypass authentication via a filename with a .ats extension instead of a .hts extension.

**•** CVE-2002-0802: Database consumes an extra character when processing a character that cannot be converted, which could remove an escape character from the query and make the application subject to SQL injection attacks.

**•** CVE-2000-0191: Overlaps "fakechild/../realchild"

**•** CVE-2004-2363: Product checks URI for "<" and other literal characters, but does it before hex decoding the URI, so "%3E" and other sequences are allowed.

**•** CVE-2002-0934: Directory traversal vulnerability allows remote attackers to read or modify arbitrary files via invalid characters between two . (dot) characters, which are filtered and result in a ".." sequence.

**•** CVE-2003-0282: Directory traversal vulnerability allows attackers to overwrite arbitrary files via invalid characters between two . (dot) characters, which are filtered and result in a ".." sequence.

## Related Attack Patterns (CAPEC)

* CAPEC-3
* CAPEC-43
* CAPEC-71

## Modes of Introduction

**•** Implementation: Since early validation errors usually arise from improperly implemented defensive mechanisms, it is likely that these will be introduced more frequently as secure programming becomes implemented more widely.

## Common Consequences

**•** Impact: Bypass Protection Mechanism, Execute Unauthorized Code or Commands — Notes: An attacker could include dangerous input that bypasses validation protection mechanisms which can be used to launch various attacks including injection attacks, execute arbitrary code or cause other unintended behavior.

## Potential Mitigations

**•** 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 problem with the above code is that the validation step occurs before canonicalization occurs. An attacker could provide an input path of "/safe\_dir/../" that would pass the validation step. However, the canonicalization process sees the double dot as a traversal to the parent directory and hence when canonicized the path would become just "/".

**•** While the script attempts to screen for '..' sequences, an attacker can submit a directory path including ".~.", which will then become ".." after the filtering step. This allows a Path Traversal (CWE-21) attack to occur.

## Notes

**•** Research Gap: These errors are mostly reported in path traversal vulnerabilities, but the concept applies whenever validation occurs.