# CWE Detail – CWE-472

## Description

The web application does not sufficiently verify inputs that are assumed to be immutable but are actually externally controllable, such as hidden form fields.

## Extended Description

If a web product does not properly protect assumed-immutable values from modification in hidden form fields, parameters, cookies, or URLs, this can lead to modification of critical data. Web applications often mistakenly make the assumption that data passed to the client in hidden fields or cookies is not susceptible to tampering. Improper validation of data that are user-controllable can lead to the application processing incorrect, and often malicious, input. For example, custom cookies commonly store session data or persistent data across sessions. This kind of session data is normally involved in security related decisions on the server side, such as user authentication and access control. Thus, the cookies might contain sensitive data such as user credentials and privileges. This is a dangerous practice, as it can often lead to improper reliance on the value of the client-provided cookie by the server side application.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2002-0108: Forum product allows spoofed messages of other users via hidden form fields for name and e-mail address.

**•** CVE-2000-0253: Shopping cart allows price modification via hidden form field.

**•** CVE-2000-0254: Shopping cart allows price modification via hidden form field.

**•** CVE-2000-0926: Shopping cart allows price modification via hidden form field.

**•** CVE-2000-0101: Shopping cart allows price modification via hidden form field.

**•** CVE-2000-0102: Shopping cart allows price modification via hidden form field.

**•** CVE-2000-0758: Allows admin access by modifying value of form field.

**•** CVE-2002-1880: Read messages by modifying message ID parameter.

**•** CVE-2000-1234: Send email to arbitrary users by modifying email parameter.

**•** CVE-2005-1652: Authentication bypass by setting a parameter.

**•** CVE-2005-1784: Product does not check authorization for configuration change admin script, leading to password theft via modified e-mail address field.

**•** CVE-2005-2314: Logic error leads to password disclosure.

**•** CVE-2005-1682: Modification of message number parameter allows attackers to read other people's messages.

## Related Attack Patterns (CAPEC)

* CAPEC-146
* CAPEC-226
* CAPEC-31
* CAPEC-39

## Attack TTPs

**•** T1539: Steal Web Session Cookie (Tactics: credential-access)

## Modes of Introduction

**•** Implementation: OMISSION: This weakness is caused by missing a security tactic during the architecture and design phase.

## Common Consequences

**•** Impact: Modify Application Data — Notes: Without appropriate protection mechanisms, the client can easily tamper with cookies and similar web data. Reliance on the cookies without detailed validation can lead to problems such as SQL injection. If you use cookie values for security related decisions on the server side, manipulating the cookies might lead to violations of security policies such as authentication bypassing, user impersonation and privilege escalation. In addition, storing sensitive data in the cookie without appropriate protection can also lead to disclosure of sensitive user data, especially data stored in persistent cookies.

## Potential Mitigations

**•** 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

**•** N/A

**•** An attacker can intercept and alter hidden fields in a post to the server as easily as user input fields. An attacker can simply parse the HTML for the substring:

## Notes

**•** Relationship: This is a primary weakness for many other weaknesses and functional consequences, including XSS, SQL injection, path disclosure, and file inclusion.

**•** Theoretical: This is a technology-specific MAID problem.