# CWE Detail – CWE-1021

## Description

The web application does not restrict or incorrectly restricts frame objects or UI layers that belong to another application or domain, which can lead to user confusion about which interface the user is interacting with.

## Extended Description

A web application is expected to place restrictions on whether it is allowed to be rendered within frames, iframes, objects, embed or applet elements. Without the restrictions, users can be tricked into interacting with the application when they were not intending to.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2017-7440: E-mail preview feature in a desktop application allows clickjacking attacks via a crafted e-mail message

**•** CVE-2017-5697: Hardware/firmware product has insufficient clickjacking protection in its web user interface

**•** CVE-2017-4015: Clickjacking in data-loss prevention product via HTTP response header.

**•** CVE-2016-2496: Tapjacking in permission dialog for mobile OS allows access of private storage using a partially-overlapping window.

**•** CVE-2015-1241: Tapjacking in web browser related to page navigation and touch/gesture events.

**•** CVE-2017-0492: System UI in mobile OS allows a malicious application to create a UI overlay of the entire screen to gain privileges.

## Related Attack Patterns (CAPEC)

* CAPEC-103
* CAPEC-181
* CAPEC-222
* CAPEC-504
* CAPEC-506
* CAPEC-587
* CAPEC-654

## Attack TTPs

**•** T1036.004: Masquerade Task or Service (Tactics: defense-evasion)

**•** T1548.004: Elevated Execution with Prompt (Tactics: privilege-escalation, defense-evasion)

**•** T1056: Input Capture (Tactics: collection, credential-access)

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Gain Privileges or Assume Identity, Bypass Protection Mechanism, Read Application Data, Modify Application Data — Notes: An attacker can trick a user into performing actions that are masked and hidden from the user's view. The impact varies widely, depending on the functionality of the underlying application. For example, in a social media application, clickjacking could be used to trik the user into changing privacy settings.

## Potential Mitigations

**•** Implementation: The use of X-Frame-Options allows developers of web content to restrict the usage of their application within the form of overlays, frames, or iFrames. The developer can indicate from which domains can frame the content. The concept of X-Frame-Options is well documented, but implementation of this protection mechanism is in development to cover gaps. There is a need for allowing frames from multiple domains. (Effectiveness: N/A)

**•** Implementation: A developer can use a "frame-breaker" script in each page that should not be framed. This is very helpful for legacy browsers that do not support X-Frame-Options security feature previously mentioned. It is also important to note that this tactic has been circumvented or bypassed. Improper usage of frames can persist in the web application through nested frames. The "frame-breaking" script does not intuitively account for multiple nested frames that can be presented to the user. (Effectiveness: N/A)

**•** Implementation: This defense-in-depth technique can be used to prevent the improper usage of frames in web applications. It prioritizes the valid sources of data to be loaded into the application through the usage of declarative policies. Based on which implementation of Content Security Policy is in use, the developer should use the "frame-ancestors" directive or the "frame-src" directive to mitigate this weakness. Both directives allow for the placement of restrictions when it comes to allowing embedded content. (Effectiveness: N/A)