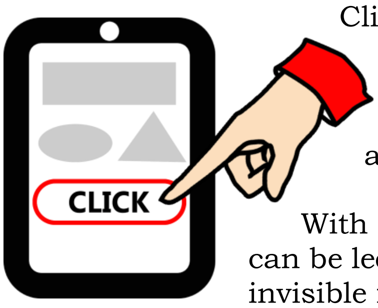
Cyber Security Centre of Excellence
West Bengal

Department of Information Technology & Electronics
Government of West Bengal

Advisory on Clickjacking

Overview:

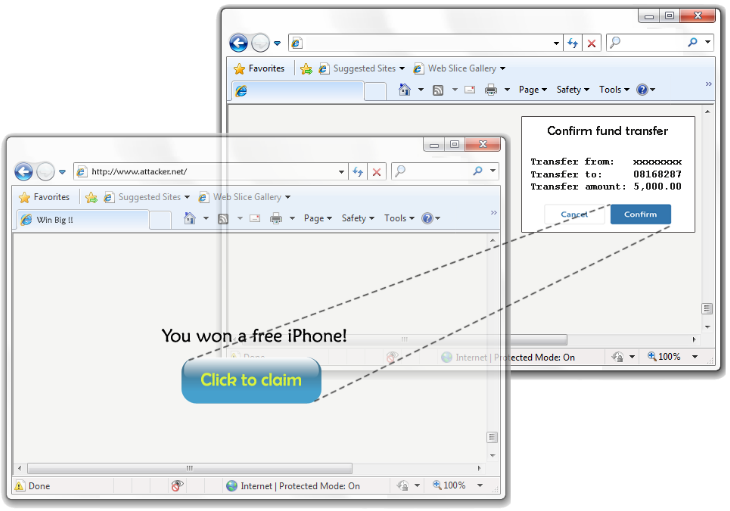


Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are clicking on the visible button, but are instead clicking on an invisible frame controlled by the attacker.

In other words, Clickjacking is an attack that occurs when an attacker uses a transparent iframe in a window to trick a user into clicking on a Calls-to-Action (CTA), such as a button or link, to another server in which they have an identical looking window. The attacker in a sense hijacks the clicks meant for the original server and sends them to the other server.

Examples:



For example, imagine an attacker who builds a web site that has a button on it that says "click here for a free iPod". However, the attacker has loaded an iframe with your mail account, and lined up exactly the "delete all messages" button directly on top of the "free iPod" button. The victim tries to click on the "free iPod" button but instead actually clicked on the invisible "delete all messages" button. In essence, the attacker has "hijacked" the user's click, hence the name "Clickjacking".

Clickjacking categorized to the following:

Classic, Likejacking, Nested, Cursorjacking, MouseJacking, Browserless, Cookiejacking, Filejacking, Password manager attack.

Impact:

Social engineering attacks are very successful by utilizing the Clickjacking also. Websites vulnerable to clickjacking may indirectly allow the Social engineering attacks. This leads leaking of Credentials, browser hijacking and system hijacking.

Solution:

Implement X-Frame-Options and Content Security Policy at server side. X-Frame-Options provide Clickjacking protection by not allowing rendering of a page in a frame.

The X-Frame-Options header has three different directives in which you can choose from.

1. **deny directive**
   The deny directive completely disables the loading of the page in a frame, regardless of what site is trying.
   X-Frame-Options: deny

2. **sameorigin directive**
   The sameorigin directive allows the page to be loaded in a frame on the same origin as the page itself.
   X-Frame-Options: sameorigin

3. **allow-from uri directive**
   The allow-from uri directive allows the page to only be loaded in a frame on the specified origin and or domain
   X-Frame-Options: allow-from https://www.example.com

Enable on Nginx

Please add it to your server block config.

```
add_header X-Frame-Options "sameorigin" always;
```

Enable on Apache

Please add it to your httpd.conf file (Apache config file).

```
header always set X-Frame-Options "sameorigin"
```

Enable on IIS

Please add it to your site's Web.config file.

```
<system.webServer>
  ...

  <httpProtocol>
    <customHeaders>
      <add name="X-Frame-Options" value="sameorigin" />
    </customHeaders>
  </httpProtocol>

  ...
</system.webServer>
```

Content-Security-Policy: frame-ancestors 'none'

The page cannot be displayed in a frame, regardless of the site attempting to do so.

Content-Security-Policy: frame-ancestors 'self'

The page can only be displayed in a frame on the same origin as the page itself.

Content-Security-Policy: frame-ancestors uri

The page can only be displayed in a frame on the specified origins.

Testing Clickjacking vulnerability:

Execute the following HTML page "**ASD-clickjack.html**" by changing the **URL**.

```
<html>
  <head>
    <title>Clickjack test page</title>
  </head>
  <body>
    <p>Website is vulnerable to clickjacking!</p>
    <iframe src="<https://URL>" width="500" height="500" sandbox ></iframe>
  </body>
</html>
```

If the page loads, the website/page is vulnerable to Clickjacking.

Source: **Application Security Division, NIC**

References:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
- https://owasp.org/www-community/attacks/Clickjacking
- https://security.nic.in/docs/[SOP]_clickjacking.pdf