**Day 3: The ClickJacking Attack**

Clickjacking, also known as a UI (User Interface) redress attack or a UI manipulation attack, is a type of malicious technique used by attackers to deceive users into clicking on something different from what they perceive on the screen. This technique is typically employed to trick users into performing actions they didn't intend to, such as clicking on a disguised button or link that leads to unintended consequences.

Here's how a clickjacking attack typically works:

1. Attack Setup: The attacker creates a website or web page with malicious intentions. This page contains some content that they want users to interact with, such as a button, video, game, or any other appealing element.

2. Overlay: The attacker then overlays this content with a transparent layer, essentially creating a hidden frame or layer on top of the visible content. Users cannot see this hidden layer.

3. Deceptive User Interaction: When a user visits the attacker's website, they believe they are interacting with the visible content (e.g., clicking a button), but in reality, their clicks are intercepted by the transparent layer.

4. Unintended Actions: The attacker can control the actions performed by the user without their knowledge. This can include actions like making the user like a social media page, follow a Twitter account, submit a form with sensitive information, or perform other malicious activities.

Clickjacking can be used for various malicious purposes, including stealing user data, spreading malware, or performing actions on the user's behalf without their consent. It often relies on the user's trust in the visible content and their lack of awareness of the hidden layer.

To defend against clickjacking attacks, web developers and websites can implement security measures like:

1. Frame Busting Code: This code prevents a website from being embedded within an iframe, making it more difficult for attackers to overlay their content.

2. X-Frame-Options Header: Setting the X-Frame-Options HTTP header can specify whether a web page can be embedded within an iframe. Options include denying all framing or allowing framing from specific domains.

3. Content Security Policy (CSP): Implementing a CSP can restrict which domains are allowed to load content on a webpage, reducing the risk of clickjacking.

4. User Education: Educating users about the dangers of clicking on unfamiliar or suspicious links and ensuring they verify the content they interact with can also help mitigate the risk.

It's important for web users to be cautious when interacting with websites and links, especially if they appear suspicious or unfamiliar. Clickjacking is just one of many techniques used by attackers to deceive users, so practicing good cybersecurity habits is essential for staying safe online.

**Blocking top-navigation**

We can block the transition caused by changing top.location in [beforeunload](https://javascript.info/onload-ondomcontentloaded#window.onbeforeunload) event handler.

The top page (enclosing one, belonging to the hacker) sets a preventing handler to it, like this:

window.onbeforeunload = function() {

  return false;

};

When the iframe tries to change top.location, the visitor gets a message asking them whether they want to leave.

In most cases the visitor would answer negatively because they don’t know about the iframe – all they can see is the top page, there’s no reason to leave. So top.location won’t change!

**Sandbox attribute**

One of the things restricted by the sandbox attribute is navigation. A sandboxed iframe may not change top.location.

**X-Frame-Options**

The server-side header X-Frame-Options can permit or forbid displaying the page inside a frame.

It must be sent exactly as HTTP-header: the browser will ignore it if found in HTML <meta> tag. So, <meta http-equiv="X-Frame-Options"...> won’t do anything.

The header may have 3 values:

**DENY**

Never ever show the page inside a frame.

**SAMEORIGIN**

Allow inside a frame if the parent document comes from the same origin.

**ALLOW-FROM domain**

Allow inside a frame if the parent document is from the given domain.

For instance, Twitter uses X-Frame-Options: SAMEORIGIN.

Here’s the result:

<iframe src="https://twitter.com"></iframe>

Depending on your browser, the iframe above is either empty or alerting you that the browser won’t permit that page to be navigating in this way.

**Showing with disabled functionality**

The X-Frame-Options header has a side-effect. Other sites won’t be able to show our page in a frame, even if they have good reasons to do so.