**Download Your Daily Notes**

**My own views on Introduction to Clickjacking**

 The attacker creates an attractive page which promises to give the user a free trip to Tahiti.  
In the background the attacker checks if the user is logged into his banking site and if so, loads the screen that enables transfer of funds, using query parameters to insert the attacker’s bank details into the form.  
The bank transfer page is displayed in an invisible iframe above the free gift page, with the “Confirm Transfer” button exactly aligned over the “Receive Gift” button visible to the user.  
The user visits the page and clicks the “Book My Free Trip” button.  
In reality the user is clicking on the invisible iframe, and has clicked the “Confirm Transfer” button. Funds are transferred to the attacker.  
The user is redirected to a page with information about the free gift (not knowing what happened in the background)

**Daily Notes - Introduction to Clickjacking**

 The attacker creates an attractive page which promises to give the user a free trip to Tahiti.  
In the background the attacker checks if the user is logged into his banking site and if so, loads the screen that enables transfer of funds, using query parameters to insert the attacker’s bank details into the form.  
The bank transfer page is displayed in an invisible iframe above the free gift page, with the “Confirm Transfer” button exactly aligned over the “Receive Gift” button visible to the user.  
The user visits the page and clicks the “Book My Free Trip” button.  
In reality the user is clicking on the invisible iframe, and has clicked the “Confirm Transfer” button. Funds are transferred to the attacker.  
The user is redirected to a page with information about the free gift (not knowing what happened in the background)

**Daily Notes - Old school defenses (weak)**

 There are several old-school defense mechanisms that can be used to secure JavaScript code, although they may not be as effective as modern security techniques. Here are some examples:  
  
Avoid using the 'eval' function: The 'eval' function is often used to execute arbitrary code in JavaScript, but it can also be used to inject malicious code into a web page. To avoid this, it's best to avoid using 'eval' wherever possible.  
  
Use variable scoping: Declaring variables with 'var' will limit their scope to the function or block where they are defined. This helps prevent global variables from being accessed or modified by external code.  
if (top != window) {  
  
top.location = window.location;  
  
}

**Daily Notes - X-Frame-Options**

 X-Frame-Options is a HTTP header that is used to protect web applications against clickjacking attacks. The header instructs the browser whether the web page can be displayed in an iframe or not. If a web page is loaded inside an iframe on another domain without the X-Frame-Options header, it becomes vulnerable to clickjacking attacks.  
  
There are three possible values for the X-Frame-Options header:  
  
DENY: The web page cannot be displayed in an iframe on any domain.  
  
SAMEORIGIN: The web page can only be displayed in an iframe on the same origin as the web page.  
  
ALLOW-FROM uri: The web page can only be displayed in an iframe on the specified domain.  
  
Here's an ex

**Daily Notes - Samesite cookie attribute**

 In JavaScript, you can set the SameSite attribute on a cookie by adding it to the cookie string. The attribute can have one of three values:  
  
SameSite=Lax: The cookie is allowed to be sent with top-level navigations and will be included with GET requests initiated by third-party websites, but not with POST requests, which can be used to perform CSRF attacks.  
  
SameSite=Strict: The cookie is only sent in requests originating from the same site that set it.  
  
SameSite=None: The cookie can be sent with cross-site requests, but only if the request is secure (i.e., uses HTTPS) and includes the Secure attribute. This is commonly used for cross-site single sign-on (SSO) cookies.  
  
Here's an example of setting the SameSite attribute on a cookie in JavaScript

**Daily Notes - Activity 1 - Clickjacking presentation**

 Slide 1: Introduction  
  
Begin by introducing the topic of clickjacking and why it's important to be aware of this type of attack.  
Slide 2: What is clickjacking?  
  
Explain that clickjacking is a type of web attack that tricks users into clicking on a button or link that is not what it appears to be.  
Provide a basic definition of clickjacking: it's a technique used by attackers to overlay an invisible or disguised element on a legitimate website, which tricks the user into clicking on something they didn't intend to.  
Slide 3: Why is clickjacking done?  
  
Explain the motives behind clickjacking attacks: to steal sensitive information, to gain access to user accounts, or to trick users into performing actions that are not in their best interest.  
Provide some examples of how clickjacking can be used to achieve these goals.  
Slide 4: How is clickjacking done?  
  
Explain the two main methods of clickjacking: UI redress attack and UI overlay attack.  
UI redress attack: In this type of attack, the attacker creates a fake UI element that looks identical to the original one but is placed over the original element. When the user clicks on the fake element, the action is actually performed on the original element, and the user is unaware of what happened.  
UI overlay attack: In this type of attack, the attacker places an invisible element over a legitimate website element. When the user clicks on the legitimate element, they are actually clicking on the invisible element, which can perform any action the attacker desires.  
Slide 5: Examples of clickjacking attacks  
  
Provide some real-world examples of clickjacking attacks: e.g., a user thinks they are clicking on a legitimate download button, but in reality, they are agreeing to install malware.  
Show how these attacks are executed by providing screenshots or a short demo video.  
Slide 6: How to protect against clickjacking attacks  
  
Explain some of the ways in which clickjacking can be prevented, such as implementing Content Security Policy (CSP) headers, using the X-Frame-Options header, or using the frame-busting code.  
Provide some best practices that users can follow to protect themselves against clickjacking attacks, such as being careful about what links they click on, keeping their browser up to date, and using ad-blockers.  
Slide 7: Conclusion  
  
Summarize the key points of the presentation and emphasize the importance of being aware of clickjacking attacks.  
Provide some additional resources that users can use to learn more about clickjacking, such as OWASP's guide on clickjacking.  
Slide 8: Q&A  
  
Open up the floor to questions and answer them to the best of your ability.  
Provide some additional resources or reading material for those who are interested in learning more.

**My Views on the Day**

*No answer yet.*

**Daily Notes - Day 3 Reflections**

 it was more informative .i leant a lot