CSCI 331 Final Project: Clickjacking - A UI Redress Attack

Faisal Al-Saif and Michelle Wang

December 2023

Abstract

For an unguarded website, it is possible for an attacker to create their own website with the facade of the unguarded site. In doing so, an attacker can masquerade their website – with a similar but slightly different URL – as the real website in order to have unsuspecting visitors click on a hidden button or link. This attack is called clickjacking, or a "UI redress attack". In this paper, we show how some seemingly innocent websites can redirect a user's activity from the attacker's imposter site and be exploited for malicious intent. The range of severity associated with this attack can range from innocently pushing web users to a site for purposes of increasing traffic to tricking users into giving away sensitive information.

1 What is the Vulnerability?

Clickjacking, also known as a "UI redress attack", is a broad family of attacks. These attacks usually involve tricking a user into clicking a web page element and intercepting or redirecting those mouse clicks to an element on a different web page that the user is unaware of. Clickjacking attacks often fool users into thinking they are clicking on one thing when they are actually clicking on another. Users think they are using a web page's normal UI, but in fact there is a hidden UI in control, and in other words, the UI has been redressed, hence the name. This attack is done by manipulating the web page so an element is

invisible or disguised as another element, and this attack can cause users to visit a harmful link, unknowingly/unwillingly download malware, visit malicious web pages, etc.

2 What resources are affected?

This attack affects the host website web page that the clickjacking attack is being performed on, and it affects the integrity of the resources, as the attacker is modifying information without the user's knowledge or consent. When a user clicks on an element in a clickjacking attack, they are not aware that their clicks are actually performing a different function than what they had originally intended.

3 Who is to Gain?

Potential attackers using clickjacking can have many different motivations, which can vary in extremity. For example, in a less extreme example, an attacker may use clickjacking to get more visits to their own website or increase the number of likes on a social media page – which is also known as likejacking – by redirecting clicks from the host website to their own. In a more extreme example, a different attacker may use clickjacking to steal user login credentials by tricking a user to unknowningly submit their credentials. Another example of a way clickjacking can be used for the attacker's gain is if the attacker creates a page with a button that promises a user prizes or a free gift, but in reality, the button is exactly aligned under an invisible "transfer money" button from the user's bank account. If the user is signed into their bank account, when the user clicks the fake button, money is transferred to the attacker. This attack is particularly malicious because the attack cannot be traced back to the attacker since the

user performed it while being legitimately signed into their bank account. Some additional examples of the goals of clickjacking include boosting ad revenues on sites, gaining likes on Facebook, increasing views of YouTube videos, or forcing users to install malware.

4 Is the Attack Difficult?

Clickjacking is not an overly complicated attack for a motivated assailant. The difficult part is finding a worthy website susceptible to the attack. If a website allows sensitive web pages to be included in iframes, then a clickjacking attack can be performed on that web page. A basic way to test if a site is vulnerable to clickjacking is to create an HTML page and attempt to include a sensitive page from the website in an iframe. This can be done using code like the following, which is a part of the OWASP Testing Guide:

```
<html>
<head>
<title>Clickjack test page</title>
</head>
<body>
<p>Website is vulnerable to clickjacking!</p>
<iframe src="http://www.yoursite.com/sensitive-page" width="500"
    height="500"></iframe>
</body>
</html>
```

Clickjacking also does not require special skills or resources, as attackers can simply exploit vulnerabilities related to HTML iframes and prevention centers targeting page framing to perform these attacks. Furthermore, the issue of

clickjacking is not just related to the iframes, but about deceiving the user and exploiting their trust in what they see in the browser window. Despite the simplicity of the attack, the consequences of clickjacking can range from mild to severe.

5 What are the Consequences?

As stated previously, attackers' motivations can vary in extremity, but attacks such as stealing login credentials or installing malware can have severe consequences for confidentiality, authenticity, and integrity of resources. Because of the potential severity of the consequences of this attack, it is of utmost importance that security measures be implemented to protect online web pages from these security threats. We will discuss potential countermeasures against clickjacking later on in this paper.

6 A Brief History of Clickjacking

"Clickjacking" was first documented as early as 2002, when Jesse Ruderman published a paper regarding the behavior of transparent elements inside web browsers. However, this work went largely unnoticed until 2008. Ruderman began his research by loading web pages in iframes to perform actions on the web page within the iframe without being noticed. Therefore in 2002, Ruderman first displayed the security vulnerability susceptible to "clickjacking." In 2008, two researchers, Grossman and Hansen, displayed the problems clickjacking can cause on a much higher level than Ruderman had in 2002. Grossman and Hansen showed how Adobe Flash Player could be used as a way to gain access to a user's microphone and camera without the user's permission. This use of UI redressing ("Clickjacking") on a large public scale brought the vulnerability to light. Some

more recent instances of this attack "in the wild" have been documented on a smaller scale, resulting in the loss of one's personal information.

7 How does it work?

Clickjacking is made possible by HTML frames (otherwise known as iframes), which gives attackers the ability to display web pages within other web pages through these frames. If a web page allows for other elements to be embedded in iframes, an attacker can cover the original web page with a hidden, transparent layer with its own JavaScript and UI elements. The attacker then tricks users into visiting the malicious page, which looks just like a site users know and trust but is actually redressed. Users could have no idea that this is occurring, as there is no indication of a hidden UI layered over the original site. Then, users click a link or a button expecting a particular action from the original site, but the attacker's script runs instead. Furthermore, in some clickjacking attacks, the attacker's script can even conceal the fact that the attack occurred by executing the expected action, making it appear as if nothing has gone wrong.

We have designed a clickjacking attack that shows users a web page that looks like they can click to win free Chipotle burritos, but in reality, we trick them into clicking a button that increases to a total dollar amount that is meant to be sent to an attacker's bank account. Our decoy web page displays a message that reads "Congratulations! Your neighborhood Chipotle is offering you a one time chance to win as many burritos as you can! You've won 0 free Chipotle burritos! CLICK THE BUTTON FOR MORE!" with a button that says "MORE BURRITOS!". The goal of this web page is to get the user to click on the button in an effort to win "more burritos". Each time the user clicks the button, it appears as if the number of free burritos that they are winning is increasing, but in reality, their clicks trigger the clicking of a different button

on a hidden target web page – one that "sends money" from the user to the attacker. For demonstration purposes, this money is not real – it is just a counter that increases each time the user clicks the "More Money!" button. Hence, our attack utilizes a redressed web page – the free burritos page – to steal money from an unsuspecting user by redirecting their clicks on the decoy page to perform an unintended action on the target page.

8 Countermeasures

There are two general ways to defend against clickjacking – client-side methods and server-side methods. The most common client-side defense method is called Frame Busting. This protection is a piece of JavaScript to check that the domain of the page is the same as the domain of the browser window. The domains of the two do not match when a page is embedded in another iframe. Client-side methods can be effective in some cases, but are considered not to be a best practice, because they can be easily bypassed by adding a line of JavaScript that prevents the frame busting prevention from being loaded. The most common server-side defense method is X-Frame-Options. Using HTTP headers, website developers can ensure their websites cannot be used for clickjacking by specifying who can embed their web page. This method asks the browser to block attempts to load the website within another iframe. Another server side protection mechanism is "Content-Security-Policy" (CSP). Using the CSP HTTP header, web developers can choose between 'none,' 'self,' or *uri* as "frame-ancestors" options. Used together, this header looks like:

```
Content-Security_policy: frame-ancestors <source 1> <source 2> ...
    <source n>
```

where <source 1> to <source n> are the different approved embedding pages.

The sources can be replaced with the 'none' and 'self' options. The 'none' option declares that the page cannot be displayed in a frame, regardless of the web page. The 'self' option declares that the page can only be displayed in a frame that has the same origin as the page itself. This HTTP header is the most up-to-date, having replaced the older X-Frame-Options header. Yet, many web pages still use the X-Frame-Options header. Server-side methods are recommended by security experts as an effective way to defend against clickjacking. As a last line of defense, many modern browsers have built in clickjacking checking algorithms to make sure that web pages are performing the way they are designed to perform.

However, it is important to note that these defense mechanisms are not enough to protect from clickjacking completely. With the majority of web browsing traffic now coming from mobile devices, the potential for creating misleading user interfaces is enormous and securing traditional web browser access is no longer enough.

9 Conclusion

Clickjacking is a serious risk for institutions that protect intellectual property or private and sensitive data. Invisible yet invasive, clickjacking allows attackers to gain unauthorized access and control over a victim's computer, giving access to sensitive data like passwords, usernames, and credit card numbers or tricking web users to install malware or viruses. The long-term impacts of this attack include identity theft, financial loss, and reputation damage. Furthermore, clickjacking can lead to weakening trust in online platforms, which can negatively impact businesses' online presence. Although seemingly harmless, the clickjacking attack demands serious security measures in order to mitigate the risks that it poses.

10 Bibliography

Advocate, A. C. D., Andrea ChiarelliPrincipal Developer AdvocateI have over 20 years of experience as a software engineer and technical author. Throughout my career. (2020, October 30). Clickjacking attacks and how to prevent them. Auth0. https://auth0.com/blog/preventing-clickjacking-attacks/

Banach, Z. (2022, December 21). Clickjacking attacks: What they are and how to prevent them. Invicti. https://www.invicti.com/blog/web-security/clickjacking-attacks/

Clickjacking defense cheat sheet. Clickjacking Defense - OWASP Cheat Sheet Series.(n.d.).https://cheatsheetseries.owasp.org/cheatsheets/ClickjackingDefenseCheatSheet.htmlDefendingwithX-Frame-Options4ResponseHeaders

Clickjacking. Hacksplaining. (n.d.). https://www.hacksplaining.com/exercises/click-jacking

Imperva. (2019, December 29). What is clickjacking: Attack example: X-frame-options Pros Cons: Imperva. Learning Center. https://www.imperva.com/learn/application-security/clickjacking/

UI redressing attacks on Android devices - black hat briefings. (n.d.). https://media.blackhat.com/ad-12/Niemietz/bh-ad-12-androidmarcusniemietz-WP.pdf

What is clickjacking? tutorial examples: Web security academy. What is Clickjacking? Tutorial Examples — Web Security Academy. (n.d.). https://portswigger.net/web-security/clickjacking