IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

IDENTIFICATION OF CLICK JACKING ATTACK USING IFRAME

[1]Tanmayi Nagale

[1]Assistant Professor

[1] Department of Computer Engineering,

[1]Thakur College of Engineering and Technology, Kandivali, Mumbai, 400101

Abstract: In recent years, cyber assaults have increased in frequency. Attackers have developed new vulnerabilities at this period to conduct harmful online actions. The attackers have targeted both the clients and the servers. Attackers have used clickjacking as one of their tactics to trick innocent internet users into taking an action. Clickjacking is seen as a form of social engineering that preys on people's naivety to defend against web threats The clickjacking attack takes use of a flaw in the online applications. With the use of user-initiated clicks, this assault employs a method that enables cross-domain attacks and executes undesired activities. This study identifies the flaws that make a website susceptible to clickjacking attacks and suggests a fix for them. This paper focuses on the idea of clickjacking, as well as defenses against it and useful real-world examples.. There are several preventative measures, but none are completely effective because there are many workarounds. However, three key defence mechanisms are presented: a built-in browser feature that respects specific HTTP headers; a client-side script technique; and a site that is completely resistant to clickjacking. The study acknowledges clickjacking as a novel and potentially deadly attack in its conclusion.

Index Terms - clickjacking, frame busting, iframe, X-Frame-Options.

I. INTRODUCTION

A malicious assault known as "clickjacking" involves the hijacking of a user interface element on a website. Technically, a clickable component on the website is placed above an invisible iframe, which causes the attacker's iframe to be run [1] rather than the user's intended action, leading to an entirely different outcome. Clickjacking is a problem on all websites that use graphical elements and across all browsers. Websites that facilitate connections between users of other websites, such as those that allow users to "like" Facebook pages, are particularly vulnerable to the assault. A clickjacking attack involves tricking a web user into interacting with a user interface element coming from an untrusted source. This interaction is intended to set off an unexpected incident that will allow an unreliable source to obtain private or sensitive information from the user. Since 2008, when a few researchers discovered an exploit leveraging Adobe Systems Flash programs that might grant the attacker remote access to a victim's web camera and microphone, the attack has been well-known. Many websites and browser developers have built defenses against clickjacking after acknowledging the issues. However, numerous websites remain vulnerable to this form of attack. This exploit affects all browsers and is not specific to any one of them. The live demonstration attack will be made by building a web GUI and overlaying it with an iframe. We will use scientific articles and comparable materials to obtain information for the report. This paper will look at the issues and dangers posed by various clickjacking attacks, as well as the barriers and countermeasures. The risks associated with combining clickjacking with other potent attacks are another concern this article raises. In a clickjacking assault, several techniques can be used to attack UI elements, such as the Facebook like button. The website's sensitive component, such as an invisible like button above another button that, when pressed, likes a specific page without the user ever knowing it, might be directly covered by an iframe that the attacker could insert. The attacking script would be executed regardless of where the user clicked by placing the iframe under the mouse cursor [4]. Another strategy is UI redressing [2], which involves making the iframe visible and appear to be a legitimate component of the website. On a bank website, this attack might be set up to request the user's bank credentials.

II. LITERATURE SURVEY

According to Lin-Shung Huang, the main cause of clickjacking is when an attacker program tricks a user into acting inappropriately by hiding or making transparent a sensitive user interface element of the target application. To address this root cause, we suggest a new defence called InContext, in which web sites (or applications) mark UI elements that are sensitive and browsers (or OSes) enforce context integrity of user actions on these sensitive UI elements. This defense makes sure that a user sees all the information she needs to see before acting and that the timing of the action matches her intent. They conducted user research using 2064 volunteers on Amazon Mechanical Turk to assess the efficiency of our offence and defense. We demonstrate that the effectiveness of our attacks, which range from 43% to 98% successful, and the effectiveness of our In Context defense against clickjacking attacks, where the usage of clickjacking is more successful than social engineering, are both quite high.[1]

Although there are clickjacking mitigations available, there aren't many online applications using them. A clickjacking vulnerability is also considerably harder to demonstrate or evaluate than more conventional client-side vectors, even though the potential for an attacker to frame a website is simple to spot. The few demonstrations of clickjacking exploitation that are made often employ unique JavaScript and HTML for each specific vulnerability because there are currently no tools that can be relied upon to do so. Even worse, frequently this secret code is never made available, forcing everyone to start from begin. Judith LundeenAnd Alves-Foss, JimA program called Browser Exploitation Framework, developed by Jim Alves-Foss, is intended to make it simple for expert penetration testers to show the effects of client-side security flaws. In this work, we provide a plugin module for BeEF that enables penetration testers to quickly show how clickjacking vulnerabilities affect systems.[2]

J. A. Shamsi, S. Hameed, W. Rahman, F. Zuberi, K. Altaf and A. Amjad have implemented methods for avoiding clickjacking attack i.e ClickSafe. The practice of "click jacking" involves taking advantage of user clicks to carry out malicious tasks that benefit the assailant. To strengthen security and dependability against click jacking attempts, we suggest Click safe, a browser-based utility. There are three main parts to click safe. A web page's harmful components that route users to external links are found by the detecting unit. The mitigation unit intercepts user clicks and issues informed warnings to users, who can then decide whether or not to proceed. Additionally, Click Safe has a feedback component that keeps track of user behaviors, translates them into ratings, and enables more informed interactions in the future. Since the identification and mitigation of threats are based on a thorough framework that makes use of the detection of dangerous web components and takes user feedback into account, Click Safe stands out from other solutions of a similar nature. We describe the functioning of click safe, its mechanism, and its ability to offer protection from click jacking to a sizable user base. [4]

Web security is a problem that D. Pawade, D. Reja, A. Lahigude, and E. Johri address and offer a solution to so that users can enjoy secure web browsing. We created the "ClickProtect" browser plugin to offer a general defence against various Clickjacking assaults. While the user is browsing the internet, ClickProtect is running in the browser. It is intended to stop clickjacking attacks by alerting the user before the risky action is taken. The dangerous, hidden web components are made accessible here along with a pop-up warning notice. This will lessen the likelihood that user information will be stolen and promote secure browsing. [5]

The user is misled into acting inappropriately as a result of the attacker application presenting a target application's sensitive UI element to the user out of context (for example, by making the sensitive UI transparent). To address this root cause, we suggest a new defence called In Context, in which websites (or applications) mark UI elements that are sensitive and browsers (or OSes) enforce context integrity of user actions on these sensitive UI elements. This defense makes sure that a user sees all the information she needs to see before acting and that the timing of the action matches her intent. To assess the efficacy of our offence and defence, Hossain Shahriar and Hisham M. Haddad conducted user studies on Amazon Mechanical Turk with 2064 participants. We demonstrate that the effectiveness of our attacks, which range from 43% to 98% successful, and the effectiveness of our In Context defence against clickjacking attacks, where the usage of clickjacking is more successful than social engineering, are both quite high. [6]

Shahriar, Hossain, The detection of clickjacking assaults, a developing problem with web application security, is covered in this work. To identify clickjacking attacks, we suggest a system for analyzing web application request and response pages. In addition to scrutinizing frame-related visual elements, our technology carefully examines JavaScript code patterns to compare them to known attack signatures. The suggested method can recognize sophisticated clickjacking assaults such cursorjacking, double-clicking, and object-based attacks against the history. We use a collection of trustworthy and fraudulent websites to assess the recommended strategy. The outcomes show that our method has low rates of false positive and false negative outcomes. The proposed method has very little overhead. [7]

A web-based attack called clickjacking has recently attracted a lot of media attention. A malicious page is designed in a clickjacking assault to fool victims into clicking on a scarcely (or not at all) visible feature of another page. An attacker could coerce the victim into taking an action that is advantageous to the attacker by using the victim's clicks, such as starting an online transaction. Although clickjacking has been the focus of numerous conversations and concerning reports, it is presently unknown how frequently attackers utilize clickjacking in actual attacks and how crucial the attack is for Internet users' security. A creative approach is put out by Christopher Kruegel, Davide Balzarotti, Manuel Egele, Engin Kirda, Marco Balduzzi, and them for the rapid and effective detection of clickjacking attacks. We outline the system we created, put into place, and used to evaluate more than a million distinct web pages. The results of the experiments demonstrate the viability of our strategy. Additionally, our empirical analysis on a large number of well-known websites reveals that clickjacking has not yet been widely adopted by online attackers. [8]

Internet technologies are the centre of the modern world. Users have access to a wide range of online tools and services to make their lives easier. In this research, we suggest creating a reliable map between the user interface and the online application in order to detect and avoid clickjacking assaults. The use of agents is used to achieve this. In this study, three agents are proposed. They are agents for tracking, detecting, and taking action. The tracking agent counts the number of clicked controls and records their positional coordinates in local memory. The detecting agent checks the number of clickable controls and their spatial coordinates between the loaded webpage and local memory. The action agent issues a warning if there is a mismatch. This project's major goal is to offer a secure web application through the use of a secure browser. The suggested work's outcomes are deemed to be satisfactory [9].

The development of new technologies, such mobile phones, GSP, WiFi, and 3G, has promoted the sharing of information via social media. The volume of information published on social media frequently results in security problems that users are unaware of. Sensitive information, such a person's name, address, and marital status, may be included in this information and may be deadly if it falls into the wrong hands. In this study, we evaluate the many types of social media ads and discuss the dangers that social media now poses. Then, we concentrate on two distinct threats, namely clickjacking and location leakage, and make recommendations for each of them along with an assessment of the solutions. [10]

Smartphones provide users a ton of ease by incorporating all necessary functionalities. Have consumers ever wondered whether the phone they are interacting with is spoofing them when they spend more time on their phones? In-depth research is done on mobile clickjacking assaults in this work. They explained the operation of the clickjacking assault and the crucial details to avoid detection. Then, we investigate the viability of launching clickjacking attacks on several UIs, including system app windows, 3rd-party app windows, and other system UIs, in order to assess its potential dangers. Finally, suggested a system-level defense mechanism for the Android platform against clickjacking assaults that requires neither user nor developer effort and is compatible with already released apps. Extensive experiments are used to evaluate the countermeasure's effectiveness. The findings demonstrate that our plan can successfully stop clickjacking assaults while having a minimal negative effect on the system. [11]

ClickDetector tool, which is based on the following three steps, and is proposed by Benmerzoug, A., and Saoudi as a way to stop attacker attempts to perform clickjacking attacks. It detects all advanced clickjacking attack techniques reported by OWASP. Analysis of the Request, Response Header, and Response Page. By incorporating user feedback, future interactions will be more knowledgeable for new users. For added security, users can use the Google Safe Browsing option. The collected experimental findings show that ClickDetector successfully identifies all created assaults without producing any false positives, demonstrating its efficacy without affecting browser speed. [12]

III. METHODOLOGY

Attacks using clickjacking are possible in a number of ways. Imagine, for instance, that a website is created by an attacker with a button that reads, "Click here for a free trip." An iframe with opacity of zero has been loaded by the attacker. A user's attempt to click "free trip" actually clicks on the invisible frame. The user won't be aware that credit card purchases are about to be made in actuality. Because it employs some fundamental HTML and JavaScript features to frame a page or generate layers on a single page, clickjacking may appear simple to execute at first. Clickjacking attacks can be done in various ways. For example, imagine an attacker who builds a website with a button that says “click here for a free trip.” The attacker has loaded an iframe with opacity set to zero. When a user tries to click on “free trip” it actually clicks on the invisible frame. The user won’t see that he or she is actually about to be charged purchases to their credit card. Clickjacking may see easy to execute at first because it uses some basic HTML and JavaScript features to frame a page or create layers on a single page. However, because there are no standardized techniques to evaluate a clickjacking attack like any other flaw or vulnerability, it is difficult to identify and prevent them. also makes use of scripting languages' and style sheets' avoidable characteristics [1]. Various clickjacking defense strategies exist, including client-side defenses and server-side defenses. Although there are treatments to lessen this onslaught, none of them offer a comprehensive remedy. Various clickjacking prevention strategies are explored in this study.

For identification of clickjacking attack, following 3 steps to be followed.

Step 1: Find Vulnerable Url

Step 2: Put url in the code of iframe, which is given below

<iframe src=" **https://monolith.xyz**/" height="550px" width="700px"></iframe>

Step 3: Open html in browser. If webpage is successfully open in browser then websie is vulnerable to Clickjacking attack.

IV. RESULTS AND DISCUSSION

Clickjacking (also known as User Interface redress attack, UI redress attack, or UI redressing) is a malicious technique used to trick a Web user into clicking on something other than what they think they are clicking on, potentially disclosing private information or taking control of their computer while they are clicking on what appear to be innocent web pages. Because the server failed to return an X-Frame-Options header, this website may be vulnerable to clickjacking attacks. A browser's ability to render a website in a frame or an iframe can be controlled using the X-Frame-Options HTTP response header. By making sure that their material is not embedded on other websites, websites can utilize this to protect themselves from clickjacking assaults. Web Server is vulnerable as a result.

Following is an demonstration of identification of vulnerable website for clickjacking attack

Example 1

Step 1: Find Vulnerable Url

For experimentation I have taken one website i.e https://monolith.xyz/

Step 2: Put url in the code of iframe, which is given below and save file using html extension

```
<html lang="en-US">
<head>
<meta charset="UTF-8">
<title>I Frame</title>
</head>
<body>
<h3>clickjacking vulnerability</h3>
<iframe src=" https://monolith.xyz/" height="550px" width="700px"></iframe>
</body>
</html>
```

Step 3: Open html file in browser. If webpage is successfully open in browser then websie is vulnerable to Clickjacking attack.

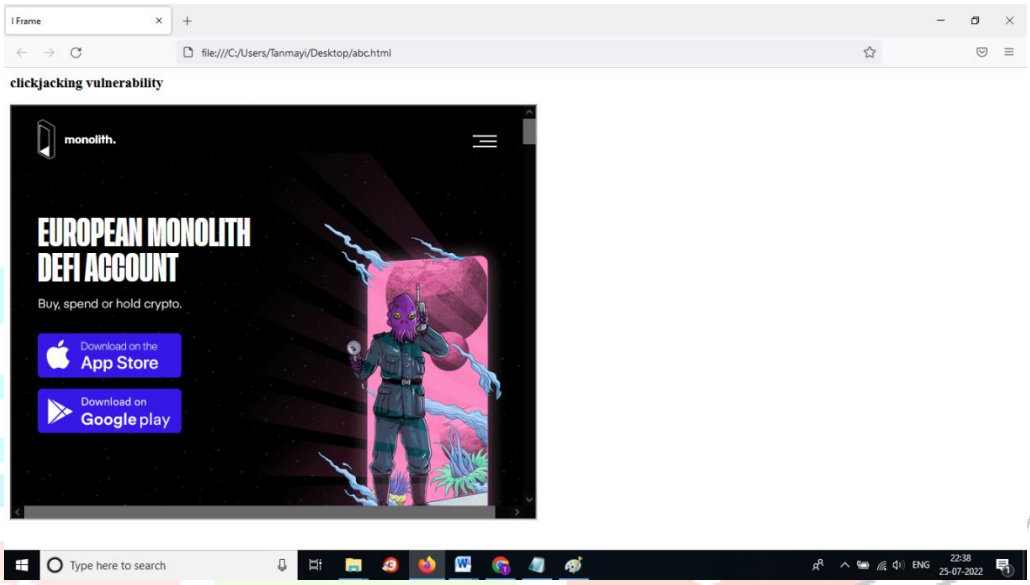

Fig 1: Demonstration of Clickjacking Attack 1

Example 2:

Step 1: Find Vulnerable Url

For experimentation I have taken one website i.e. **https://data.rarible.com**

Step 2: Put url in the code of iframe, which is given below and save file using html extension

```
<html lang="en-US">
<head>
<meta charset="UTF-8">
<title>I Frame</title>
</head>
<body>
<h3>clickjacking vulnerability</h3>
<iframe src="https://data.rarible.com/" height="550px" width="700px"></iframe>
</body>
</html>
```

Step 3: Open html file in browser. If webpage is successfully open in browser then website is vulnerable to Clickjacking attack.

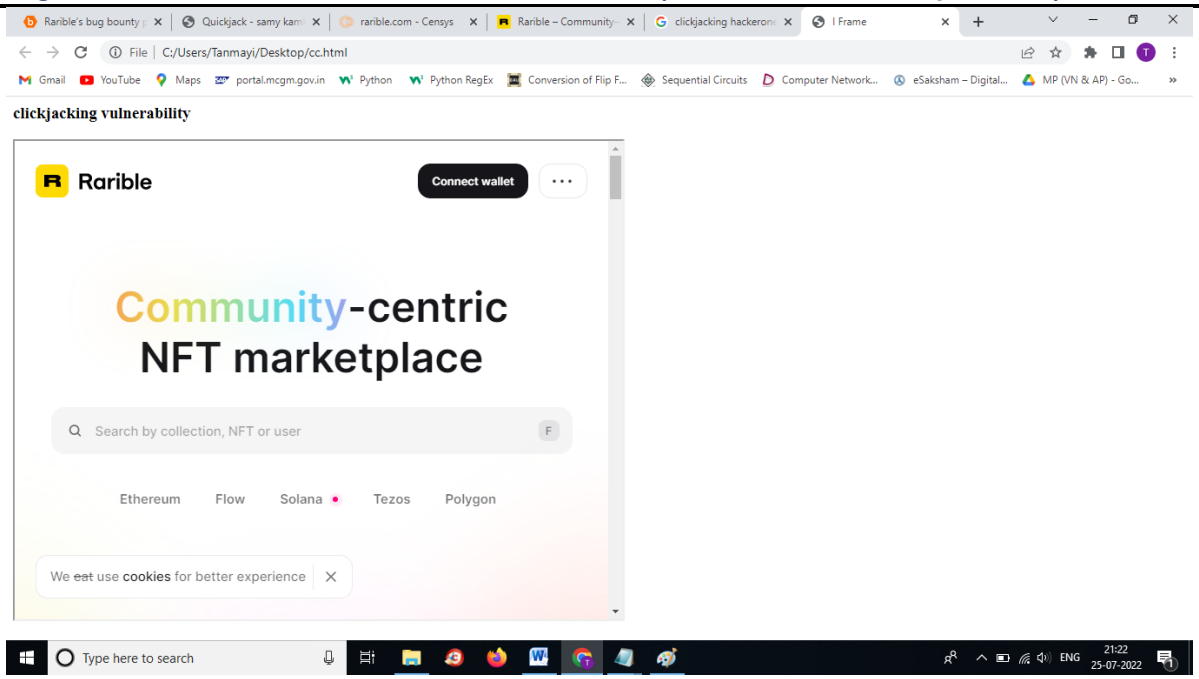

Fig 2: Demonstration of Clickjacking Attack 2

REFERENCES

[1] Huang, Lin-Shung, et al. "Clickjacking: attacks and defenses." Proceedings of the 21st USENIX Security Symposium. 2012.

[2] B. Lundeen and J. Alves-Foss, "Practical clickjacking with BeEF," *2012 IEEE Conference on Technologies for Homeland Security (HST)*, Waltham, MA, USA, 2012, pp. 614-619, doi: 10.1109/THS.2012.6459919.

[3] Hansen, Robert, and Jeremiah Grossman. "Clickjacking." SecTheory Retrieved 5.03 (2008): 2012.

[4] J. A. Shamsi, S. Hameed, W. Rahman, F. Zuberi, K. Altaf and A. Amjad, "Clicksafe: Providing Security against Clickjacking Attacks," 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering, Miami Beach, FL, USA, 2014, pp. 206-210, doi: 10.1109/HASE.2014.36.

[5] D. Pawade, D. Reja, A. Lahigude and E. Johri, "Implementation of extension for browser to detect vulnerable elements on web pages and avoid Clickjacking," *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, Noida, India, 2016, pp. 226-230, doi: 10.1109/CONFLUENCE.2016.7508118.

[6] Hossain Shahriar and Hisham M. Haddad. 2015. Client-Side Detection of Clickjacking Attacks. Int. J. Inf. Sec. Priv. 9, 1 (January 2015), 1–25. https://doi.org/10.4018/IJISP.2015010101

[7] Shahriar, Hossain, et al. "Request and Response Analysis Framework for Mitigating Clickjacking Attacks." IJSSE vol.6, no.3 2015: pp.1-25. http://doi.org/10.4018/IJSSE.201507010

[8] Marco Balduzzi, Manuel Egele, Engin Kirda, Davide Balzarotti, and Christopher Kruegel. 2010. A solution for the automated detection of clickjacking attacks. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10). Association for Computing Machinery, New York, NY, USA, 135–144. https://doi.org/10.1145/1755688.1755706

[9] Bala, Kiran and Dr. E. Babu Raj. "EFFECTIVE APPROACH TO DETECT CLICKJACKING ATTACKS." (2016).

[10] Lim Chin Nei, Loo Yow Cherng, Manmeet Mahinderjit Singh "A Case Study on Clickjacking Attack and Location Leakage", International Journal of Scientific & Engineering Research, Volume 5, Issue 7, July-2014 ISSN 2229-5518

[11] L. Wu, B. Brandt, X. Du and Bo Ji, "Analysis of clickjacking attacks and an effective defense scheme for Android devices," *2016 IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, PA, USA, 2016, pp. 55-63, doi: 10.1109/CNS.2016.7860470.

[12] Benmerzoug, A., Saoudi, L. (2020). Implementation of Web Browser Extension for Mitigating Clickjacking Attack. In: Bouhlel, M., Rovetta, S. (eds) Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'18), Vol.2. SETIT 2018. Smart Innovation, Systems and Technologies, vol 147. Springer, Cham. https://doi.org/10.1007/978-3-030-21009-0_39