Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 8, August 2015, pg.406 – 413

RESEARCH ARTICLE

Clickjacking - Safe Guarding User from Client Side Attacks and Vulnerabilities

[1] Rupesh K, [2] Madhusekhar Y, [3] Sujilatha T

[1]MTech Student, Department of Computer Science & Engineering, Gokula Krishna college of Engineering, Sullurpet under JNT University , Ananthapur, Andhra Pradesh, India

[2]Associate Professor, Department of Computer Science & Engineering, Gokula krishna college of Engineering, Sullurpet under JNT University, Ananthapur, Andhra Pradesh,, India

[3]Associate Professor, Department of Computer Science & Engineering, Gokula krishna college of Engineering, Sullurpet under JNT University, Ananthapur, Andhra Pradesh,, India

Abstract- In a clickjacking attack, effort has been taken into the research in the client side attacks that includes the vulnerabilities that are like the cross-site scripting and cross-site forgery and many other types recently and known as Clickjacking. In this context the clickjacking uses the vulnerability that the browser exploits the weakness in a different domain that is isolated and having the same origin policy. It performs this action by making the user to click on the frame the user wants but actually that is not the user need. In many of the cases, this malicious vulnerability makes the unsuspecting user to consider their details that are compromised in a click. Anyway there are some protections measures are present for the clickjacking, these applications of the web are making them to mitigate for long and between. Along with them there are many possibility for the attacker to make a page to frame that can be easy to make it detect, and it is more difficult to measure the effects of the clickjacking that have vulnerability than the client side traditional vectors.

Index Terms — clickjacking, vulnerabilities, Frame bursting, I-Frame, X-Frame, cross-site forgery

I. INTRODUCTION

Despite more review and check, click jacking still trails with the exact and clear definition. The process of manipulating the frame is also way of clickjacking attack. A click can also be stolen by covering the frame containing the victim page. In filtration process we can easily get the IFRAME existing in URL or not.

If IFRAME exist in URL that means that particular URL can be affected via Clickjacking attack. Although clickjacking is considered as the dangerous effort, it is now not precise to verify up to how much extent the clickjacking is mainly used by the malicious users in the world, and unable to know how important the attack is intended for the purpose of security of the users in the internet.

The frame busting that is the prevalence of the web site that is assessed during the experiment we conducted .We should first prepare a web page that takes the only one parameter that denotes the URL that can be embedded within the IFRAME .If the contents in the web page that is the IFRAME complete its rendering and then the loading, we can conclude and verify that the IFRAME still exits. The content that is totally in the window web browser will perform frame busting in the pages and thus the IFRAME in the page is removed.

To automatically do this experiment, we implement a extension of the Firefox that takes all the unified resource locators (URLs) that are to be visited. Whenever a page gets loaded the Firefox has the extensions that waits for a some specified time and then it concludes the IFRAME presence .If the document DOM-tree does not have the IFRAME ,we can conclude frame busting is performed on the embedded page This may lead to a message DENY. We surveyed the frame busting practices of the top 500websites. Using both known and novel attack techniques, we found that all of the clickjacking defenses we encountered could be circumvented in one way or another. These are beneficial to by performing these functionalities.

1. We surveyed the frame busting practices of the top 500websites.
2. We found that all of the clickjacking defenses we encountered could be circumvented in one way or another.

II. IMPLEMENTATION

1. Anti CSRF Token Module:

A Cross-site request forgery hole is when a malicious site can cause a visitor's browser to make a request to your server that causes a change on the server. The server thinks that because the request comes with the user's cookies, the user wanted to submit that form.

Depending on which forms on your site are vulnerable, an attacker might be able to do the following to your victims:

- Log the victim out of your site. (On some sites, "Log out" is a link rather than a button!)
- Change the victim's site preferences on your site. (Example: Google)
- Post a comment on your site using the victim's login.
- Transfer funds to another user's account.
- Attacks can also be based on the victim's IP address rather than cookies:
- Post an anonymous comment that is shown as coming from the victim's IP address.
- Modify settings on a device such as a wireless router or cable modem.
- Modify an intranet wiki page.
- Perform a distributed password-guessing attack without a botnet. (This assumes they have a way to tell whether the login succeeded, perhaps by submitting a second form that isn't protected against CSRF.)

CSRF attacks usually involve JavaScript to submit the cross-site form automatically. It is possible for a malicious site to make a user submit a form to another site even without JavaScript, however: form fields can be hidden and buttons can be disguised as links or scrollbars.

a) Preventing CSRF:

Make sure form submissions that cause server-side changes use your own forms. There are two ways you can do this:

1. Check the referrer header. If it is not present, or if it does not show the correct URL as the referrer, reject the submission. This has the advantage of being simple and sane, but the disadvantage that users who have told their browsers to omit the referrer header (out of concern for privacy) or lie about the referrer (in order to gain access to porn sites that use the referrer for

the wrong purpose) will have trouble. This strategy doesn't work if the form uses GET and the page can contain user-generated content with links.

2. Include a hidden field in the form and check its value when the form is submitted. A simple scheme is to use an MD5 hash of the login cookie, some information about the form, and a secret on the server. Another possibility is to use a randomly generated one-time key for every form you serve, assuming you have a sufficiently unpredictable source of random numbers. This has the disadvantage of making it unsafe for users to save the HTML for forms and upload them to, say, bug databases.

2. Iframe Module:

When you allow websites to frame you, you basically give them full permission to decide, what part of JavaScript of your very own script can be executed and what cannot. That sounds crazy right? So, let's say you have three script blocks on your website. The website that frames you doesn't mind two of them - but really hates the third one. may be a frame buster, maybe some other script relevant for security purposes. So the website that frames you just turns that one script block off - and leave the other two intact. Now how does that work?

Well, it's easy. The entire framing website is doing, is using the browser's XSS filter to selectively kill JavaScript on your page. This has been working in IE some years ago but doesn't anymore - but it still Work's perfectly fine in Chrome

3) X –Frame Module:

The X-Frame-Options HTTP response header was introduced by Microsoft in 2008. It can be used to indicate if the web page can be loaded in a "frame" or "iframe" tag. There are two possible values for X-Frame-Options. They are called "DENY" and "SAMEORIGIN". By using "DENY" the web page can not be loaded by a frame. In contrast, "SAMEORIGIN" can be used to display the web page in a frame when the origin of the top-level browsing context is not different. The problem with the necessity of JavaScript does not exists in this case. This is because there is no need to use JavaScript code. Possible problems can be to use X-Frame-Options with proxies, which strip the HTTP header, or multi-domain sites Nevertheless, the advantages outweigh the disadvantages. One should therefore decide for this variant of the frame Bursting.

Browser	Lowest version
Internet Explorer	8.0
Firefox (Gecko)	3.6.9 (1.9.2.9)
Opera	10.50
Safari	4.0
Chrome	4.1.249.1042

Table 1: Browser compatibility of X-Frames

III. Architecture diagram:

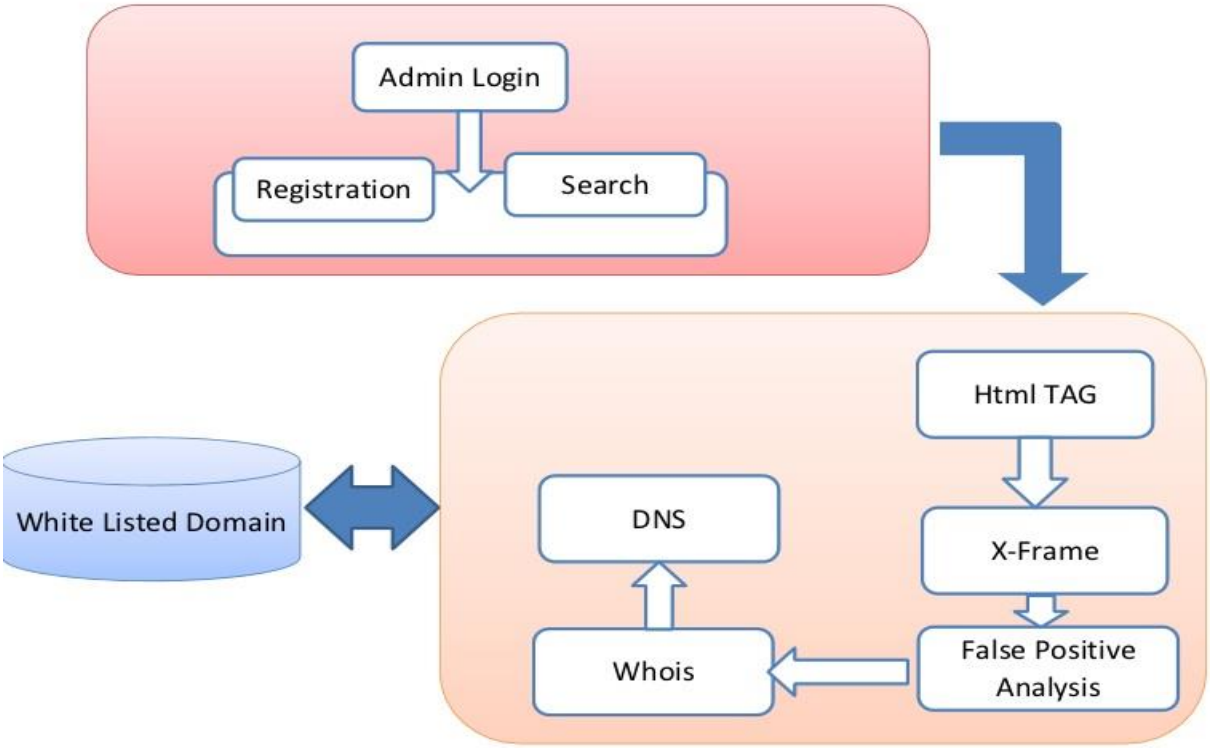

Fig 1::X-Frame verification and listing the domain

The admin register and then login to search the webpages to get the HTML tags and the number of other attributes of the HTML tags like X-Frame, I-Frame count and then a test is performed to get the Black listed domain and a alert is displayed that the site is black listed and if it is white listed then the user is prompted to continue with the current site.

IV. RELATED WORK

Despite extensive discussions and reports, clickjacking still lacks a formal and precise definition Informally, it is a technique to lure the victim into clicking on a certain element of a page, while her intention is to interact with the con- tent of a different site. That is, even though the victim is under the impression of clicking on a seemingly harmless page, she is actually clicking on an element of the attacker's choice.

The typical scenario, as described by Grossman and Hansen , involves two different websites: A target site is a website accessible to the victim and important for the attacker. Such sites include, for example, online-banking portals, auction sites, and web mail services. The goal of the attacker is to lure the victim into unsuspectingly clicking on elements of the target page, on the other hand, is under control of the attacker. Commonly, this page is created in a way so that a trans- parent IFRAME containing overlays the content of . Since the victim is not aware of the invisible IFRAME, by correctly aligning over , an attacker can lure the victim into clicking elements in , while she is under the impression of clicking on A successful clickjacking attack, for example, might result in the victim deleting all messages from her web mail inbox, or generating artificial clicks on advertisement banners. shows a real-world clickjacking attack that has been used to propagate a message among Twitter users In this attack, the malicious page embeds Twitter.com on a transparent IFRAME. The status-message field is initialized with the URL of the malicious page itself. To provoke the click, which is necessary to publish the entry, the malicious page displays a button labeled "Don't Click." This button is aligned with the invisible "Update" button of Twitter. Once the user performs the click, the status message (i.e., a link to the malicious page itself) is posted to her Twitter profile. While clickjacking attacks can be performed in plain HTML, the use of JavaScript can be used to create more sophisticated attacks.

For instance, JavaScript allows the attacker to dynamically align the framed content with the user's mouse cursor, thus making it possible to perform at- tacks that require multiple clicks. Note that manipulating the frame's opacity level (e.g., making it transparent) is not the only way to mount a click- jacking attack. A click can also be "stolen" by covering the frame containing the victim page with opaque elements, and then leaving a small hole aligned with the target element on the underlying page. Another possible approach consists of resizing and/or moving

the IFRAME in front of the mouse just before the user performs a click. Unlike other common web vulnerabilities such as cross- site scripting and SQL injection, clickjacking is not a consequence of a bug in a web application (e.g., a failure to properly sanitize the user input). In contrast, it is a consequence of a misuse of some HTML/CSS features (e.g., the ability to create transparent IFRAMEs), combined with the way in which the browser allows the user to interact with invisible, or barely visible, elements.

A number of techniques to mitigate the clickjacking problem have been discussed on security-related blogs One approach proposes to extend the HTTP protocol with an optional, proprietary X-FRAME-OPTIONS header. This header, if evaluated by the browser, prevents the content to be rendered in a frame in cross-domain situations. A similar approach was proposed to enhance the CSS or HTML languages to allow a page to display different content when loaded inside a frame.

V. CONCLUSION:

This paper presents a new approach to counter clickjacking. The solution takes user feedback to create dynamic black and white lists and overcome limitations posed by previous solutions. Though these add-ons are open-source, and the client-side code can be easily modified for malicious content, secure protocols are allowed for secure transmission can be considered for the security of the users.

REFERENCES:

[1] Robert Hansen and Jeremiah Grossman. (2008, Sep.) Explanation of Clickjacking. http://www.sectheory.com/clickjacking.htm

[2] 2. Michal Zalewski. (2008, September) Dealing with UI redress vulnerabilities inherent to the current web page of the site http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2008-September/016284.html

[3] Eric Lawrence. (2009, January) IE8 Security Part VII: ClickJacking Defenses. http://blogs.msdn.com/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking

[4] The Register. (2009, February) Twitter attack exposes awesome power of clickjacking http://www.theregister.co.uk/2009/02/13/twitter_clickjack_attack/

[5] Joey Tyson. (2009, November) Facebook Worm Uses Clickjacking in the Wild. http://theharmonyguy.com/2009/11/23/facebook-worm-uses-clickjacking-in-the-wild/

[6] Marco Balduzzi, Manuel Egele, Davide Balzarotti, Engin Kirda, and Christopher Kruegel, "A Solution for the Automated Detection of Clickjacking Attacks," in ASIACCS'10, Beijing.

[7] Jeremiah Grossman. (2009, June) Clickjacking. http://jeremiahgrossman.blogspot.com/2009/06/clickjacking-2017.html

[8] W3C. HTML 4.01 Specification, Introduction to links and anchors. http://www.w3.org/TR/REC-html40/struct/links.html#anchors

[9] W3C. HTML5 Specification, Session history and navigation. [Online]. http://www.w3.org/TR/html5/history.html#scroll-to-fragid.

