Volume 6, Issue 12, December 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey of Clickjacking Attack and Countermeasures in Web Environment

R. P. Seenivasan*
Department of Computer Science,
Pondicherry University, Puducherry, India

K. Suresh Joseph
Department of Computer Science,
Pondicherry University, Puducherry, India

Abstract— An important and timely attack technique on the web is Clickjacking. Web framing attacks such as clickjacking use iframes to hijack a user's web session. In clickjacking attack, a malicious page is constructed, that tricks the victim to click on malicious page without the knowledge of the victim. This attack can be used alone or in combination with other attacks. This attack is also responsible for making attacks such as CSRF possible. Clickjacking can cause severe damages, including compromising a user's private webcam, email or other private data, and web surfing anonymity. Once the requirements are fulfilled, an attacker will ensnare the victim to perform at least one action on this user authenticated web page that the attacker is not allowed to perform. This consequently leads to the fact that the access control mechanism will fail and would not work as expected. There are many techniques available to prevent this attack but frame busting [5] technique has got most priority. This paper discusses on the survey of different possible basic clickjacking attacks and the possible countermeasures to overcome the attack.

Keywords— Clickjacking, iframes, CSRF (Cross Site Request Forgery), Web surfing anonymity, frame busting

I. INTRODUCTION

There are many attacks generated day by day and protective mechanisms are also provided. One of the attacks is Clickjacking. This technique is also known under the previously used term "UI redressing". A clickjacking attack was introduced by Robert Hansen and Jeremy Grossman in 2008. Most websites still have not implemented effective protection against Clickjacking. In this attack, attacker will construct a malicious webpage, by using this page attacker will steal all the confidential information like passwords, spam mails, cookies, etc. Victim will click on the element of a malicious page without the knowledge about where they are actually clicking. This attack will not only steal valuable information but also install number of software's like harmful viruses, spyware which will corrupt the files in the systems. In section 2 the description about basic clickjacking and how it is implemented. In section 3 the description about existing anti-clickjacking defences. In section 4 the description about related clickjacking attacks are discussed. In section 5, proposed general solutions for the clickjacking attacks and conclusion are discussed.

II. BASIC CLICKJACKING

This attack will use two nested iframes for positioning an element from target website. The inner frame should be large enough to display elements in target page such that without scrolling elements on which user will click is visible. Next, the outer frame will be a window for the page loaded in the inner frame. Users may think that they are clicking on the website they see but they are clicking on the invisible website.

The following method is usually known under the name "clickjacking", but it is not the only type of click hijacking attack. For this reason, the attack will be called "basic clickjacking". From the perspective of an attacker, at first one has to create a web page that includes a large iframe containing the target web page. The HTML code of the iframe, which is in the example inside the file "inner.html", is displayed in Listing 2.1[1].

Listing 2.1: Iframe code that opens the web site of "google.com" (filename: inner.html)

```
<iframe id="inner" src="http://www.google.com" width="2000" height="2000" scrolling="no" frameborder="none">
</iframe>
```

This iframe opens the web site of "google.com". The output of the web site interpreted by the web browser "Mozilla Firefox 3.6.3". It shows that the user is automatically authenticated against the web page with his or her name [1]. This authentication method is a typical characteristic that an attacker is looking for [2].

In the second step, a new web page is created. It is exemplary called "clickjacking.html". This page loads the web page "inner.html" with an iframe. So, the inner.html will not be visible to the user. However, there is a crucial difference in the HTML code of the "clickjacking.html" iframe, as shown in Listing 2.2[1].

Listing 2.2: If a

Listing 2.2: Iframe and CSS code to open the web page of the file "inner.html" (filename: clickjacking.html)

```
<iframe id="inner" src="inner.html" width="2005" height="290" scrolling="no" frameborder="none">
</iframe>
```



```
<style type="text/css"><!--
#inner {position: absolute; left: -1955px; top: -14px ;}
//--></style>
```

It should be clear that the focus is on the text, or rather the link “Sign out”, which is given at the top right of the web page of “google.com” or included in the “inner.html” document [2].

Fig 1.1.: Truncated and masked output of the web site “google.com” interpreted by the web browser Mozilla Firefox 3.6.3

A comparison Of Listing 2.2 and Listing 2.1 shows that there is a new attribute “id” inside the “iframe” tag. The attribute “id” holds the value “inner”. This value is addressed by CSS code given with the last three lines of Listing 2.2. “#inner” specifies that the position of the element that holds the “id” value “inner” is absolute and that it should be moved 1,955 pixels to the left and 14 pixels up. Furthermore, there are new height and width specifications. These values are defined according to the layout of the web site and especially to the “Sign out” link. It should be noted that it may be an advantage to keep these values small to focus on the essential content, but in this case it does not play an important role. The web browser output of “Mozilla Firefox 3.6.3” is given in Fig 1.2[2].

Fig 1.2.: Output of the file “clickjacking.html” interpreted by the web browser Mozilla Firefox 3.6.3

A single click on the “Go” button will enables the user to visit the web site of the “Chair for Network and Data Security”. It is used a form here to demonstrate the practical relevance. At this point it should be noted that the codes of the aforementioned Listings 2.1 and 2.2 include only the most essential HTML elements and do not correspond to a valid W3C HTML structure as in Listing 2.3[2].

Listing 2.3: HTML form web page (part of the file: trustedPage.html)

Listing 2.3: HTML form web page (part of the file: trustedPage.html)

```
1 <! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2 "http://www.w3.org/TR/html4/loose.dtd">
3 <html>
4 <head>
5 <title>Trusted web page</title>
6 </head>
7 <body>
8 <h1>www.nds.rub.de</h1>
9 <form action="http://www.nds.rub.de">
10 <input type="submit" value="Go">
11 </form>
12 </body>
13 </html>
```

The generated HTML document is now supplemented with clickjacking code. Like Listing 2.2, Listing 2.4 also uses an iframe with CSS. This iframe loads the web page“clickjacking.html” in a 50 pixel width and 300 pixel height-iframe [4]. For this reason, a victim will click on the “Sign out” hyperlink by clicking on the “Go” button. This action is not visible to the user, because of the defined CSS property “opacity: 0.0”. It makes the iframe transparent and thus hides it fro-m the user. This shows clearly that clicking on a link will forward to a “google.com” web page [2].

Listing 2.4: Iframe code with CSS for clickjacking (part of the file: trustedPage.html)

```
1 <iframe id="clickjacking" src="clickjacking.html" width="50" height="300"
```



```
scrolling="no" frameborder="none">
2 </iframe>
3
4 <style type="text/css"><!--
5 #clickjacking {position: absolute; left: 7px; top: 81px; opacity: 0.0}
6 //--></style>
```

The output of the web page with "Mozilla Firefox 3.6.3" is displayed in Fig 1.3. Furthermore, the status bar is shown in the web browser. It holds information about the click, because the mouse pointer is placed over the link of the current web page [2].

Fig 1.3.: Output of the file "trustedPage.html" interpreted by the web browser Mozilla Firefox 3.6.3

III. EXISTING CLICKJACKING ATTACKS

Existing clickjacking attacks are classified into: (1) compromising target display integrity, waiting until users can fully recognize elements before action is done by the user (2) compromising pointer integrity, depending on the cursor feedback and (3) compromising temporal integrity, providing enough time before clicking [4].

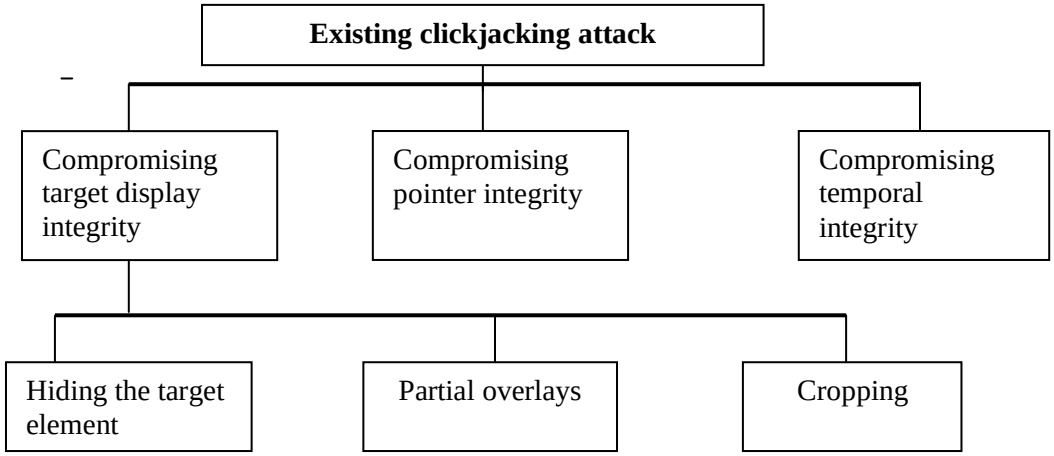

Fig 2.1.: Block diagram for existing clickjacking attack

A. Compromising target display integrity

Hiding the target element: Many browsers will support HTML/CSS features that help the attackers to visually hide the target element but not mouse events. This can be done by making the target element transparent by wrapping that in a div container with opacity value zero and by using lower CSS z-index [4].

Partial overlays: It is possible to visually confuse a victim by obscuring only a part of the target element. For example, attackers could overlay their own information on top of a PayPal checkout iframe to cover the recipient and amount fields while leaving the "Pay" button intact, the victim will thus have incorrect context when clicking on "Pay". This overlaying can be done using CSS z-index or using Flash Player objects that are made topmost with Window Mode property set to wmode=direct. Furthermore, a target element could be partially overlaid by an attacker's popup window [4].

Cropping: The attacker may crop the target element to only show a piece of the target element, such as the "Pay" button, by wrapping the target element in a new iframe that uses carefully chosen negative CSS position offsets and the Pay button's width and height. An extreme variant of cropping is to create multiple 1x1 pixel containers of the target element and using single pixels to draw arbitrary clickable art [4].

B. Compromising pointer integrity

To be fully visible and authentic, pointer feedback is required [10]. An attacker may violate it by displaying a fake cursor away from pointer known as cursor jacking. This will make the victim to click on the fake cursor. Using the CSS cursor property, an attacker can easily hide the default cursor and programmatically draw a fake cursor several pixels away from the original position [4].

C. Compromising temporal integrity

The UI element is manipulated after the user decided to click on the element, but before the actual click occurs. Humans will take few hundred milliseconds to react to changes, attacker will use this time to change the element [4]. For example, an attacker could move the target element (via CSS position properties) on top of a decoy button shortly after the victim hovers the cursor over the decoy, in anticipation of the click. To predict clicks more effectively, the attacker

could ask the victim to repetitively click objects in a malicious game or to double-click on a decoy button, moving the target element over the decoy immediately after the first click [4].

IV. EXISTING ANTI-CLICKJACKING DEFENSES

There are both client-side and server-side approaches to protect user against clickjacking attack. Same–origin policy is used to protect from distrusting websites, it is failed after any of the clickjacking attacks are introduced. Some of the anti-clickjacking defenses are

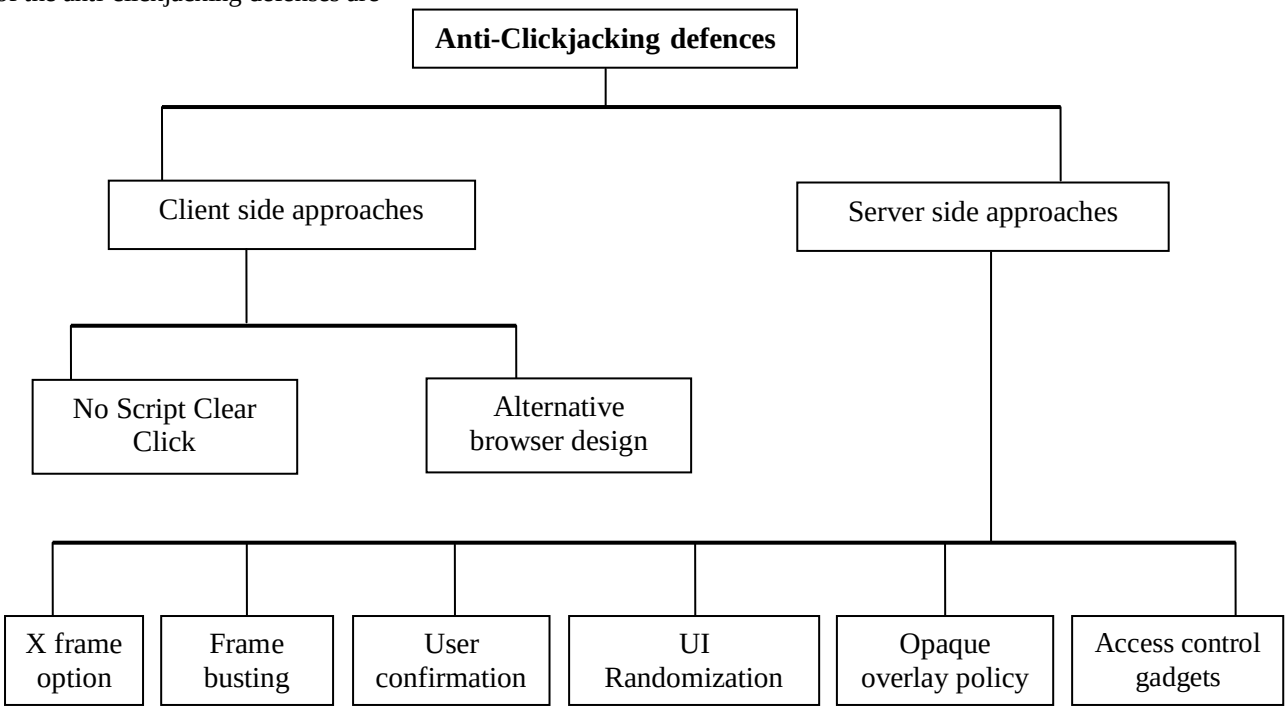

Fig 3.1.: Block diagram for existing anti-clickjacking defenses

D. Client-side approaches

No Script ClearClick: No Script is an extension for Mozilla-based web browsers like Firefox. It allows JavaScript, Java, Flash and other plugins to be executed only by trusted web sites of the user's choice. In addition, XSS and clickjacking protection is integrated. Concerning clickjacking, the extension has provided a plugin called "ClearClick" since version 1.8.2[6]. This feature recognises clicking and typing on hidden elements and clearly reveals the element by showing a "Clear Click Warning" window as it is displayed in below figure. Furthermore, the extension protects the user inter alia against the usage of IFRAMEs [35]. It blocks such frames in untrusted web pages and also trusted web pages, which try to load content from untrusted web pages [37]. This extension is highly recommended as a client-side solution for clickjacking with a Mozilla-based web browser. At this point it should be supplemented prophylactic ally, that the web browser Google Chrome has a plugin such as "No Script", too. This plug-in is called "Not Scripts" and it does not provide advance protection for clickjacking [38].

Fig 4.1: ClearClick Warning during clickjacking

Alternative Browser Design: Latest browsers support clickjacking attack using iframe. There are alternative browser [43] designs like Gazelle or secure browsers which gives priority to security [42].

E. Server-side approaches

User Confirmation: One straight forward way is to provide authentication for a click i.e. sending conformation popup after the click has been done on the target element. Facebook currently using this approach for like button in order to avoid clickjacking and like jacking. This approach degrades user experience and victim may be tricked to click on both target element and a confirmation popup. So, this approach is not used by everyone [4].

UI Randomization: Another technique to protect the target element is to randomize its UI layout. For example, PayPal could randomize the position of the Pay button on its express checkout dialog to make it harder for the attacker to cover it with a decoy button. This is not robust, since the attacker may ask the victim to keep clicking until successfully guessing the Pay button's location [4].

Opaque Overlay Policy: The Gazelle web browser forces all cross-origin frames to be rendered opaquely. It will check whether opacity zero is present in the source code since attacker will make some web pages invisible [4].

Visibility Detection on Click: Another method is to block mouse clicks if the browser finds any frames that are not fully visible [4]. Adobe has added such protection to Flash Player's webcam access dialog in response to webcam clickjacking attack, but this defense is not available for web content. The ClearClick module of the Firefox extension No-Script also uses this technique. Although ClearClick is reasonably effective at detecting visual context compromises, its on-by-default nature must assume that all cross-origin frames need clickjacking protection, which results in false positives on some sites. Due to these false positives, ClearClick prompts users to confirm their actions on suspected clickjacking attacks, posing a usability burden. An extension called ClickIDS was proposed to reduce the false positives of ClearClick by alerting users only when the clicked element overlaps with other clickable elements. Finally, a fundamental limitation of techniques that verify browser-rendered bitmaps is that cursor icons are not captured. Thus, pointer integrity is not guaranteed. Unfortunately, cursor spoofing attacks can still be effective against some users even if the default cursor is visible over the target element [4].

Access Control Gadgets: Access control gadgets (ACG) were recently introduced as a new model for modern OSes to grant applications permissions to access user-owned resources such as camera or GPS. An ACG is a privileged UI which can be embedded by applications that need, access to the resource and application permission is need to access the corresponding resource. The notion of ACGs is further generalized to application-specific ACGs, allowing applications to require authentic user actions for application-specific functionality. Application-specific ACGs precisely capture today's web widgets that demand a clickjacking defense.

Protecting temporal context: One common way to give users enough time to comprehend any UI change is to impose a delay after displaying a dialog, so that users cannot interact with the dialog until the delay expires. This approach has been deployed in Flash Player's webcam access dialog.

Frame busting: Latest defense used by many browsers is clickjacking [5]. A small JavaScript code is used to check if the page that contains the script is currently framed or not.

```
<script type="text/java script">
If (top! =self)
{
Top. Location=self. Location;
}
</script>
```

X-Frame Options: This approach is introduced by Microsoft to counter clickjacking attacks. Similar to frame busting it also avoid unauthorized framing. X-Frame-Options header values are SAMEORIGIN and DENY [34]. While DENY prevents the browser from rendering the document within a frame completely, SAMEORIGIN allows the browser to display a resource within a frame whenever the top frame was served by the same origin. IE will support ALLOW_FROM also [4].

V. RELATED WORK

Including clickjacking there are many types of attacks, of this type some of them are UI redressing, Stroke jacking, like jacking, Event jacking, Class jacking, Double click attack.

A. UI redressing

The discussed "Sign out" link, or rather the iframe of the above section is, without the CSS design specification "opacity", visible. If the link which is invisible in the webpage of clickjacking is shown directly in the webpage then the attack is called UI redressing. This user interface (UI) redressing method is especially useful when there are buttons with nonspecific text like "Go", "Click here" or for example "Send". Thus, it is not always necessary to make elements invisible to the user. It is important to know that "Clickjacking" is regularly used as another term for "UI redressing". The validity of "UI redressing" purely depends on the observers [19] view.

B. Strokejacking

The term "Strokejacking" was introduced by Michal Zalewaski in 2010. Zalewski used the fact that there are web pages available that make use of the focus functionality of web browsers. The example chosen for this approach is

"google.com" search engine. By loading the web page, the cursor will be automatically placed into the search field and thus it will be focussed. The "Strokejacking" attack uses this property by showing an iframe, which includes the search engine web page, together with the text field of the attacker. This text field is also focussed automatically. Therefore, the web browser has to choose between these two elements, or more precisely, between the "iframe" and the "input" tag. This decision takes a JavaScript code of the attacker indirectly [25].

The interesting fact is that an attacker could make the iframe transparent, as in the case of "basic clickjacking", and that the victim can key in e.g.: Malicious code without prior intimation of the host [26].

C. Likejacking

Likejacking [29] is a special kind of "basic clickjacking". This term was officially introduced by the security company "Sophos" in 2010. In essence, there is no difference between "basic clickjacking" and "likejacking", because the same method with frames is used.

The special feature is that the characteristic of social networking is taken into account. The victim, in form of a user of Facebook, has to click on the button "Like", which is offered by Facebook as a service. By clicking on this button, a status message will be printed on Facebook and also under the web page category "Like". So one is able to hover web pages with malicious code by using clickjacking on the "Like" button, but the code is hidden to the user [2].

D. Eventjacking

Depending on the XSS filtering, it is possible to append the "onclick" event handler up to all links or the whole document. That gives an attacker the option to perform actions between a click and a request of the next web page [2]. As the name "eventjacking" says, this type of attack hijacks events to expand the functionality of XSS by clickjacking. This can be used to read the cookie, to get a CSRF token or to perform many other dangerous actions by using the victim as a click generator [29].

E. Classjacking

CSS offers the attribute "class" as a selector to style a group of HTML elements. It can be said that it is possible to combine different actions as a consequence of a click on an element by using the CSS "class" attribute and "jQuery" as a JavaScript library. For this reason, the attack is called "Classjacking" [2]. This attack is usually a combination of XSS and the attack will be more dreadful when java script [4] is incorporated.

F. Double-click attack to steal user private data

Today's browsers do not protect temporal integrity for web sites [6]. We show in our second attack that even if a security-critical web page (such as an OAuth dialog page) successfully employs frame busting [19] (refusing to be embedded by other sites), our attack can still successfully click jack such a page by compromising temporal integrity for popup windows. We devised a bait-and-switch double-click attack against the OAuth dialog for Google accounts, which is protected with X-Frame-Options [34]. The attack is shown in Figure. First, the attack page baits the user to perform a double-click on a decoy button. After the first click, the attacker switches in the Google OAuth pop-up window under the cursor right before the second click (the second half of the double-click). This attack can steal a user's emails and other private data from the user's Google account [2].

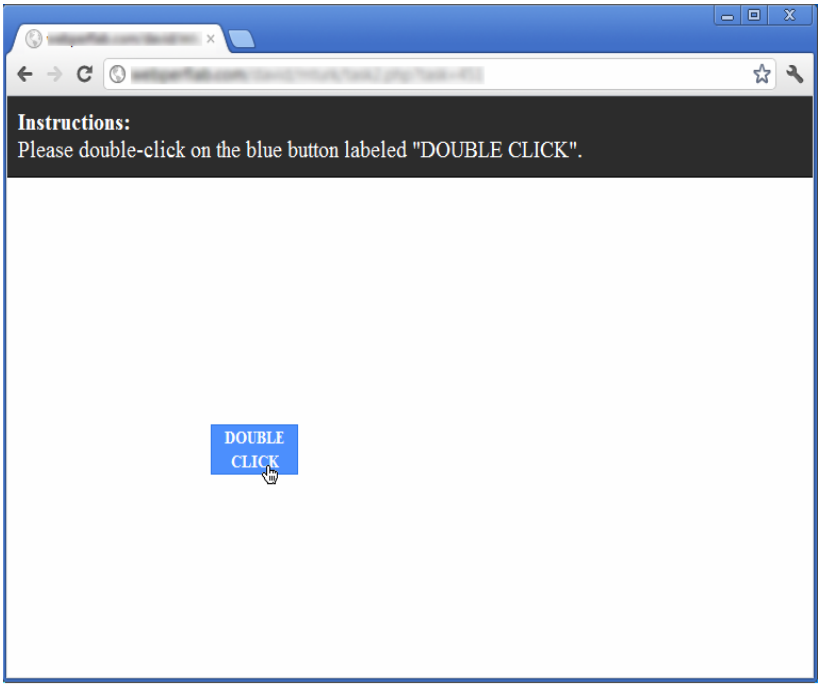

Fig 5.1: Screenshot of window of the attacker page

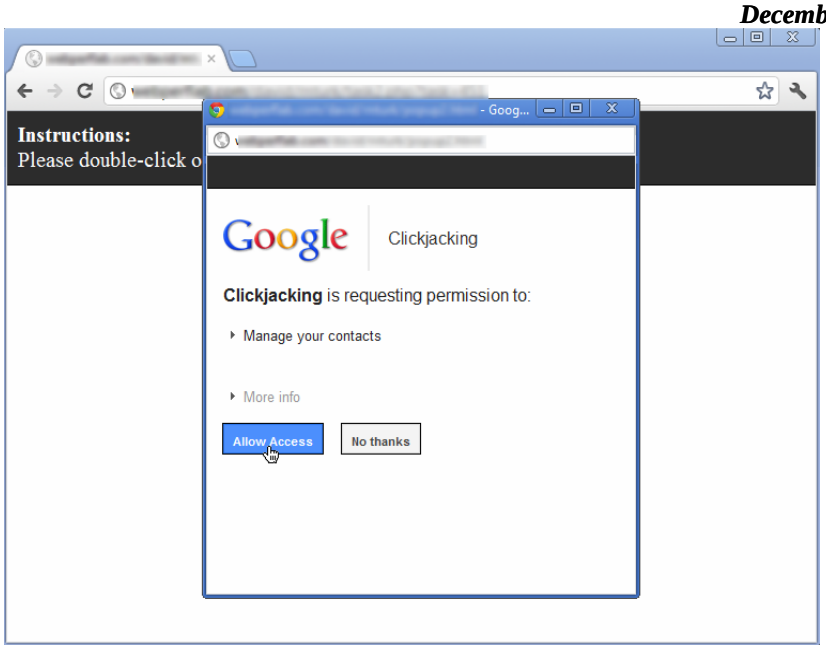

Fig 5.2: Double-click attack page. The target OAuth dialog popup window appears underneath the pointer immediately after the first click on the decoy button.

VI. FUTURE WORK AND CONCLUSION

Clickjacking attacks are an emerging threat on the web. Clickjacking can cause several damages, including compromising a user's private webcam, email, privacy of data used , and web anonymity. In this paper both existing clickjacking attacks and anti-clickjacking attacks are listed. Many sites protect themselves from clickjacking and many websites are still susceptible to attacks due to the identified bypasses and weaknesses of the presented countermeasures. Novel countermeasures are needed that take these more complex scenarios and attacks into consideration to successfully counter the threats imposed by clickjacking.

The proposed system is evaluated with the metrics namely security and cost. The results obtained by incorporating anti-clickjacking technique, clearly proves that it can be evidently used in almost all web services in near future. So that the privacy of the users data can be preserved if this attack is not taken into account in near future it can even create a national threat resulting in stealing of all military and confidential data of the governing bodies.

REFERENCES

[1] A. Sankara Narayana, "Clickjacking Vulnerability and Countermeasures," International Journal of Applied Information System (IJAIS) – ISSN: 2249-0868, Foundation of Computer Science FCS, New York, USA Volume 4 – No.7, December 2012.

[2] M.Niemietz. UI Redressing: Attacks and Countermeasures Revisited. In CONFidence, January 2011.

[3] Macro Balduzzi, Manuel Egele, Engin Kirda, Davide Balzarotti, Christopher Kruegel, " A Solution for the Automated Detection of Clickjacking Attacks," p.3,April 2012.

[4] Lin-Shung Huang, Alex Moshchuk, Helen J. Wanj, Stuart Schechter, Collin Jackson, "Clickjacking: Attacks and Defenses".

[5] Gustav Rydstedt, Elie Bursztein, Dan Boneh, Collin Jackson, "Busting Frame Busting: A Study of Clickjacking Vulnerabilities on Popular Sites," Web 2.0 Security and Privacy, 2010.

[6] Sebastian Lekies, Mario Heiderich, Dennis Appelt, Thorsten Holz, Martin Johns, "On the fragility and limitations of current Browser-provided Clickjacking protection schemes".

[7] G. Aharonovsky, "Malicious camera spying using ClickJacking," http://blog.guya.net/2008/ 10/07/malicious-camera-spying-using- clickjacking/, 2008.

[8] M. Balduzzi, M. Egele, E. Kirda, D. Balzarotti, and C. Kruegel, "A solution for the automated detection of clickjacking attacks," In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, 2010.

[9] D. Baron, "Preventing attacks on a user's history through CSS: visited selectors," http://dbaron.org/mozilla/visited-privacy, 2010.

[10] L. C. Aun, "Clickjacking with pointer-events," http://jsbin.com/imuca.

[11] Gustav Rydstedt, Elie Bursztein, Dan Boneh, Collin Jackson, "Busting Frame Busting: A Study of Clickjacking Vulnerabilities on Popular Sites," Web 2.0 Security and Privacy, 2010.

[12] Egele, Kirda, Balzarotti, Kruegel, "New Insights into Clickjacking," OWASP Foundation AppSec Research, 2010.

[13] Bikash Dash, "Introduction and Prevention to Clickjacking Attack Eg Hacking", 2011.

[14] Barth, A.Caballero, J. Song, "D: Secure Content Sniffing For Web Browsers, or How to stop papers from reviewing themselves," In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland (2009).

[15] B. Hill, "Adaptive user interface randomization as an anti-clickjacking strategy. http://www.thesecuritypractice. com/the_security_practice/papers/AdaptiveUserInterfaceRandomization.pdf, May 2012.
[16] R. Hoffmann, P. Baudisch, and D. S. Weld, "Evaluating visual cues for switching windows on large screens," In Proceedings of the 26th annual SIGCHI conference on Human factors in computing systems, 2008.
[17] L.S.Huang and C.jackson, "Clickjacking attacks unresolved" http://mayscript.com/blog/david/clickjacking-attacks-unresolved,2011
[18] C. Jackson, "Improving browser security policies," PhD thesis, Stanford University, 2009.
[19] "Preventing Frame Busting and Click Jacking (UI Redressing)," [Online]. Available: http://coderrr.wordpress.com/2009/02/13/preventing-frame-busting-and-click-jacking-ui-redressing/,2009.
[20] Mozilla firefox extension, https://developer.mozilla.org/enUS/docs/Creating_XPI_Installer_Modules
[21] E. Bordi, Proof of Concept - CursorJacking (noScript). http://static.vulnerability.fr/noscript-cursorjacking.html.Wikipedia. Likejacking. http://en.wikipedia.org/wiki/Clickjacking#Likejacking.
[22] Robert Hansen, Jeremiah Grossman, "Clickjacking," [Online]. Available: http://www.sectheory.com/clickjacking.htm,Dec 2008.
[23] Michal Zalewski, "Arbitrary page mashups (UI redressing)," [Online]. Available:
[24] http://code.google.com/p/browsersec/wiki/Part2#Arbitrary_page_mashups_ (UI_redressing), Nov. 2010.
[25] "World Wide Web Consortium (W3C)," World Wide Web Consortium, [Online]. Available:http://www.w3.org,2010.
[26] "Strokejacking," seclists.org, [Online].Available:http://seclists.org/ fulldisclosure/ 2010/ Mar/232, 2010. "Stroke triggered XSS and Strokejacking," andlabs.org, [Online]. Available: http: //blog.andlabs.org/2010/04/stroke-triggered-xss-and-strokejacking_06.html,2010.
[27] "Clickjacking with AJAX and Flash," Blog of gnucitizen.org, [Online]. Available: http:// www.gnucitizen.org/ blog/ more-advanced-clickjacking-ui-redress-attacks/,2008
[28] "Introduction to links and anchors," World Wide Web Consortium, [Online]. Available:http://www.w3.org/TR/REC-html40/struct/links.html#anchors,2010.
[29] Paul Stone, "Next Generation Clickjacking (Slides)," Black Hat Europe, [Online]. Available: https://media.blackhat.com/bh-eu-10/presentations/Stone/BlackHat-EU-2010-Stone-Next-Generation-Clickjacking-slides.pdf, 2010.
[30] Larry Chaffin, Anton Chuvakin, Champ, III Clark, "Infosecurity 2008 Threat Analysis," SyngressMedia, 2008, pp. 109–111.
[31] Marcus Niemietz, "Authentication Web Pages with Selenium: Vulnerability Analysis and Exploitation of Authentication Web Pages with Selenium," AVM, 2010, pp. 22–23.
[32] "Same origin policy for JavaScript," [Online]. Available:https://developer.mozilla.org/En/Same_origin_policy_for_JavaScript, Mozilla, 2010.
[33] "Clickjacking Tool," Context Information Security Limited, [Online]. Available: http://contextis.com/resources/tools/clickjacking-tool/2010.
[34] "The X-Frame-Options response header," Mozilla Corporation, [Online]. Available:https:// developer.mozilla.org /en/the_x-frame-options_response_header,2010.
[35] "No Script Changelog," Inform Action, [Online]. Available :http://noscript.net/changelog,2010.
[36] "No Script Firefox extension," Inform Action, [Online]. Available: http://noscript.net, 2010.
[37] "Clear Click and Clickjacking," Inform Action, [Online]. Available: http://noscript.net/faq#faqsec7 ,2010.
[38] "Not Scripts Limitations," [Online]. Available: http://optimalcycling.com/other-projects/notscripts/limitations/,2010.
[39] "Preventing Frame Busting and Click Jacking (UI Redressing)," [Online]. Available: http://coderrr.wordpress.com/2009/02/13/preventing-frame-busting-and-click-jacking-ui-redressing/ ,2009.
[40] "IE8 Security Part IV: The XSS Filter," Microsoft, [Online]. Available: http: //blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-iv-the-xss-filter.aspx, 2008.
[41] "Google Chrome - Stable Channel Update," Google, [Online]. Available: http: //googlechromereleases.blogspot.com/2010/03/stable-channel-update.html,2008.
[42] "defineSetter," Mozilla, [Online]. Available:https://developer.mozilla.org/ en/ JavaScript/ Reference/ Global _Objects/Object/defineSetter,2010.
[43] Marcus Niemietz, "Authentication Web Pages with Selenium: Vulnerability Analysis and Exploitation of Authentication Web Pages with Selenium," AVM, 2010, p. 35.

