On Detection and Prevention of Clickjacking Attack for OSNs

Ubaid Ur Rehman, Waqas Ahmad Khan
School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan.
{ 12msccsurehman, 12msccswkhan } @seecs.edu.pk

Nazar Abbas Saqib, Muhammad Kaleem
College of Electrical and Mechanical Engineering
National University of Sciences and Technology
Islamabad, Pakistan.
nazar.abbas @ceme.nust.edu.pk,
kaleemmuk@yahoo.co.uk

Abstract—**Clickjacking attacks are the emerging threats to websites, especially to online social networks (OSNs). In this paper, we describe some new attacks to online websites. The new Clickjacking attacks cause serious damage to users by stealing their personal credentials or by sharing their personal information on social networks bringing moral degradation to them. The attacker applications are hidden behind the sensitive user interface to steal the clicks of the user and use them for the illegal purposes. To detect and prevent Clickjacking attacks, we propose a browser-based solution referred to as Cursor Spoofing and Clickjacking Prevention (CSCP). CSCP ensures protection Cursor spoofing attack with high effectiveness and also the Likejacking attacks, other variation of Clickjacking attacks which associate malicious code to Facebook *Like* buttons. We have conducted our studies on 442 participants to evaluate the effectiveness of our attacks and also defenses. Results show that our attack success rate falls between 76% and 78%.**

Keywords- Clickjacking, Cursor Spoofing, Cascading Style Sheet (CSS), Frame Busting, Likejacking

I. Introduction

Clickjacking attack was introduced by Robert Hansen and Jeremiah Grossman in 2008, to steal user initiated mouse clicks to perform actions that the user is not interested in. The attacker achieves the goal by choosing a clickable region on a web page e.g. the region where the *login button* on the webpage is located and user is asked to enter his or her username and password. On clicking, malicious web page loads from the website inside an *iframe*, which makes use of *Cascading Style Sheets (CSS)* to hide all the content on the website except the targeted region. In that region, Clickjacking is used to trick the user on the button or routing him to another web page. It may be transparent or overlapped by another element on the website. Technically, both the JavaScript and CSS are used to place the *iframe* under the mouse cursor to make user click in the targeted region resulting a malicious action the attacker is intended to do [1]. The vulnerability can occur in all the browsers to embed the code or a script of Clickjacking, which executes without the user's knowledge. Clickjacking attack can cause several threats like stealing personal data such as bank account information, credit card information and social security numbers or installing software applications on a computer.

Clickjacking has diverse types of attacks, among them the most popular are Likejacking and Cursorjacking. Likejacking is a malicious Facebook enabled technique of tricking users of websites into clicking web links, which are included in Facebook *likes* and appears in user's profile. Cursorjacking is a user interface redressing technique to change the location of the cursor user perceives [2]. The attacker replaces the real cursor by the fake cursor to perform click on a targeted region also known as Cursor Spoofing.

According to the ranking of the social networking website, we checked top fifteen networking websites. Among them, eight websites are still vulnerable to Clickjacking attack [15]. To avoid such attacks, browser-based solution can be used effectively. Rest of this paper is organized as follows.

Section II explains the state of the art work in the field of Clickjacking. Section III summarizes our contribution. We explain our proposed attack on social networking websites in Section IV. Section V describes the prevention of existing and proposed attack. The details of our experimental setup are provided in Section VI. Section VII illustrates the conclusion.

II. Related Work

This Section explains the work done in the field of Clickjacking. That identifies different attacks and their prevention techniques.

A. Vulnerability and Countermeasures

Now a day, many websites are vulnerable to Clickjacking attack. The targets for such attacks are social networking websites such as *Facebook* and *Twitter*. Clickjacking uses iframe to hijack the user's web session. A malicious page has constructed to cheat the victim to click on different elements of the webpage which are not visible to the user. Security experts describe the tricks of an attacker to do certain action on webpage by hiding clickable elements inside an invisible iframe. Some popular websites are still vulnerable to such kinds of attacks. The suspense at attacker perspective is to what extent of Clickjacking attack is being used and how significant it is for the security of Internet user [5].

B. Automated Detection of Clickjacking Attacks

In [6], authors proposed solution for automated detection of Clickjacking attack. The attacker tries to steal the victim clicks to do an unintended action that provides the benefit to an attacker in form of initiating an online money transaction.

978-1-4799-2293-2/13 $31.00 © 2013 IEEE
DOI 10.1109/FIT.2013.37

The proposed solution for automated and efficient detection of Clickjacking attacks is reported by the analysis of a million of unique web pages by system design, implementation and results. The detection approach divides into *ClickIDS* and *NoScript*. *ClickIDS* is a browser plug-in that intercepts the mouse click events. *NoScript* is a Firefox add-on that provides protection against *Cross Site Scripting (CSC)*. The testing units simulate the behavior of human user to interact with the web page. The limitation in the applied system is in interaction with the clickable elements of the page. The result of the implemented system was evaluated in terms of False Positive, True Positive, Borderline case and False Negative. Message spamming and Click fraud are used as a proof of concept for Clickjacking. The borderline attacks are also identified during the analysis.

C. Browser Based Protection Scheme

The limitations of current anti-Clickjacking approaches and nested Clickjacking help in Clickjacking attack against social networks such as Google+. The protection is needed on both the client and server sides. *NoScript* and *ClearClick* are options to protect the client side while *Frame Busting* and *X-Frame*-approaches can be applied on the server side. The issues of double and nested Clickjacking are also discussed in detail with a real-world example of exploiting Google+ via Google's image. The analysis of the specific website based on *X-frame-options*, *Frame Busting* and *CSP* mechanism is also presented [4].

D. Framebusting on Popular Websites

Framebusting is a technique to prevent Clickjacking attack on the websites. Top 500 websites use *JavaScript* based Clickjacking prevention techniques to secure the websites. Some are browser and *cross browser* specific. Certain generic Clickjacking attacks include *double framing*, *onBeforeUnload Event* that exploits *XSS filter*, *referrer checking problem*, *clobbering top.location*, *IE restricted zone* and *sandbox attributes*. The three defenses against these kinds of attack are: X-*Frame-Options* for Microsoft, *Content Security Policy* for Mozilla and an improved version of *JavaScript* based protection technique. [9]

E. Practical Technique

Clickjacking technique cheats users of the website into clicking some hidden button or link that someone does not intend to click. Thus perform some malicious activity as stealing sensitive information or taking over the account. This technique exploits the weakness of web browsers *cross domain isolation* and *same origin policy*. Clickjacking rely on custom JavaScript, CSS and HTML. *BeEF* is a browser exploitation framework that security professionals use to demonstrate effect of client side security vulnerabilities [7]

F. Attacks Against Framed Websites

Clickjacking allows attacker to inject arbitrary text into *forms* that extract contents from a webpage. A new technique that allows information leaked from an iframe to be used for log-in detection. Several attacking methods are described to carry out successful Clickjacking attack: Positioning Methods, Cross Site Request Forgery, Text Field injection, Content Extraction, HTML source extraction, forced drag and drop with Java Applets and Anchor element position detection. Two techniques are describe to prevent Clickjacking are *Framebusting* and *X-Frame-Options*. The Framebusting is not efficient prevention. *X-Frame-Options* were first introduced for IE8. To prevent Clickjacking in previous browsers, websites should use both *X-Frame-Options* and *JavaScript* based method [10].

G. Attacking on Web Application

There are numbers of attacks on Web apps as Clickjacking, Cross Site Request Forgery, Cross Site Scripting and XML based attacks. But here our focus is on Clickjacking only. Many attacks exist similar to Clickjacking as *Cookiejacking* and *Likejacking.* Cookiejacking attack use drag and drop to steal cookies from IE during the time that *Likejacking* attack is counter to *Facebook* that cheats user to click a hidden *like* button of Facebook. There are number of techniques use to bypass Facebook defensive measures that includes *disabling JavaScript and HTML5 sandbox attribute*. By using these techniques the attacker may steal user information or compromise user's account. These techniques may be prevented by using *Framebusting* and *X-Frame-Options*. Each of these having some deficiencies: *Framebusting* is based on JavaScript and problem is that it may be difficult to get right and can be bypassed easily. The *X-Frame-Options* supports by newer browsers as IE8 & above [8].

H. Surviving Attacks

The present-day Clickjacking attacks are mainly classified into three classes: *compromising display integrity, compromising pointer integrity* and *compromising temporal integrity*. Other variants attacks are: A *Cursor spoofing attack, double-click attack and whacks a mole attack* [3].

i. Compromising Display Integrity: The targeted element is wrapped with transparent cover that attacker desires the user to click on that button to do clickjacking attack. Some time to confuse the user attacker obscure the partial overlaying on the targeted button as on top of *PayPal* button the *Pal* is wrapped by the targeted element and the user just seen to have click on the *Pay* button but actually the click made on the *PayPal* button [17, 18].

ii. Compromising Pointer Integrity: The attacker violates the integrity of pointer by showing a fake cursor instead of the real cursor. The victim assume the fake cursor as an actual one and perform clicks that help attacker to achieve his/her targets from the user. The keyboard focus stroke jacking attack is the example of compromising pointer integrity. The stroke jacking attack simulates an input field getting focus but actual focus is on the keyboard targeted element. User is force to type unwanted information in the targeted element. [19, 20]

iii. Compromising Temporal Integrity: The attacker cheats the user in the short interval of time that manipulates the *UI* element. The user decides to click on *UI* element. The targeted *UI* was overlapped by a *button/UI element* and the actual click perform on the overlapped *button/element*. Humans typically require a few hundred milliseconds to react to the visual changes and attacker gets benefits of the slow reaction to launch timing attacks. [21, 22]

iv. Cursor Spoofing Attack: This attack compromises the pointer integrity. Attacker hides the webcam permission dialog box inside an advertisement and compromise the integrity of cursor. The fake cursor provides false feedback of pointer location to the user. A loud video ad play automatically that compels the user to click on the skip ad link. The user clicks on the link and the real cursor gets permission of user's webcam from adobe flash player webcam setting. [3]

v. Double-Click Attack: The double-click attack is based on compromising temporal integrity. The browser do not protect against *Framebusting*. The double-click attack do *Bait and switch* that the user do a double click *right after the first click*. The attacker switches user focus to the Google popup window under the right before the second click. [3]

vi. Wack-a-Mole Attack: The whack-a-mole attack is the hybrid of *pointer and temporal integrity*. The attacker asks the user to click as fast as possible and suddenly switch *Facebook Like button* and gets the profile of the victim. [3]

I. Surviving Defenses

According to our research the following are the Clickjacking defenses up till now. The defenses are based on JavaScript, CSS, and X-Frame-Option.

i. FrameKiller: The framekiller uses *JavaScript* to detect the target inside *iframe* which do not work on Facebook *like* button.

ii. User Confirmation: The users are asking on each click. The re-verify of each event degrades user experience.

iii. User Interface Randomization: The position of the button randomize on the website as *Pay button* position changes at each visit. The attacker may ask the victim to keep clicking until get succeed.

iv. Opaque Overlay Policy: The defense force all the cross origin frames to render opaquely, may break the flow of other legitimate websites.

v. Framebusting: The Framebusting check the target inside an *iframe* using *http header X-Frame-Options* to ask browser not to render this page inside an *iframe*. This technique not works on *Facebook Like*. The *Like* button must be within an iframe.

vi. Visibility Detection on Click: The mouse clicks are block if the browser detect that the clicked cross origin frame is not fully visible.

a) ClearClick compare the bitmap of target element rendered in the isolation and real target element on the page.

b) ClickIDS alerts when click overlaps with clickable elements.

Neither *ClearClick* nor *ClickIDS* assure pointer integrity.

vii. Give User Interface Delays: Give enough time to the user to realize what is happening. The user cannot interact with the target element until the delay expires as *Firefox add-ons NoScript.*

The *NoScript* provide a tradeoff between user experience penalty and protection from timing attack.

III. Our Contribution

The major contributions of this paper are in assessing the effectiveness of existing attacks as well as designing and evaluating a new browser based defense. We also introduce new attacks that are based on Likejacking and Cursor spoofing, describe in the next section. We build an extension for Google-chrome to provide a client side solution to protect against such attacks.

IV. Our Proposed Attacks

Our proposed attacks are based on *Likejacking* and *Cursor spoofing*. They mostly affect the users who are very sensitive about their personal information. The attacks may also be modified to steal the user credential in form of username, password, pictures, and any private information that has more value for the users. The proposed attacks are launched into two different scenarios.

A. Use of CAPTCHA

CAPTCHA is a type of challenge response in computing to ensure that the response is generated by users. *CAPTCHA* is used to avoid bots. The computer asks the user to complete a word or retype some irregular letters due to which computer authenticates either its human or bots. The proposed attack is a type of human authentication scheme in which user is asked to follow a certain pattern to allow the user access to the actual website. The user needs to visit the *human-check* web site [13], may follow the pattern in order to verify and redirected to the actual website. The pattern of buttons is at specific location, which shows that behind each button, some function is performed. The user when click on 1 makes a click on *Facebook like* button. A social network page is embedded under the *human-check* website. When click on *2* the *liked* page is shared on the user wall without knowing to the user. When click on 3, continue button appears and proceeds to the actual website.

The front view and embedded frame inside the Human Check web page are shown in *Figure 1.*The problem arises when the attacker embeds some obscene social network link and it is shared with the user's friends causing shame for the user in his or her community.

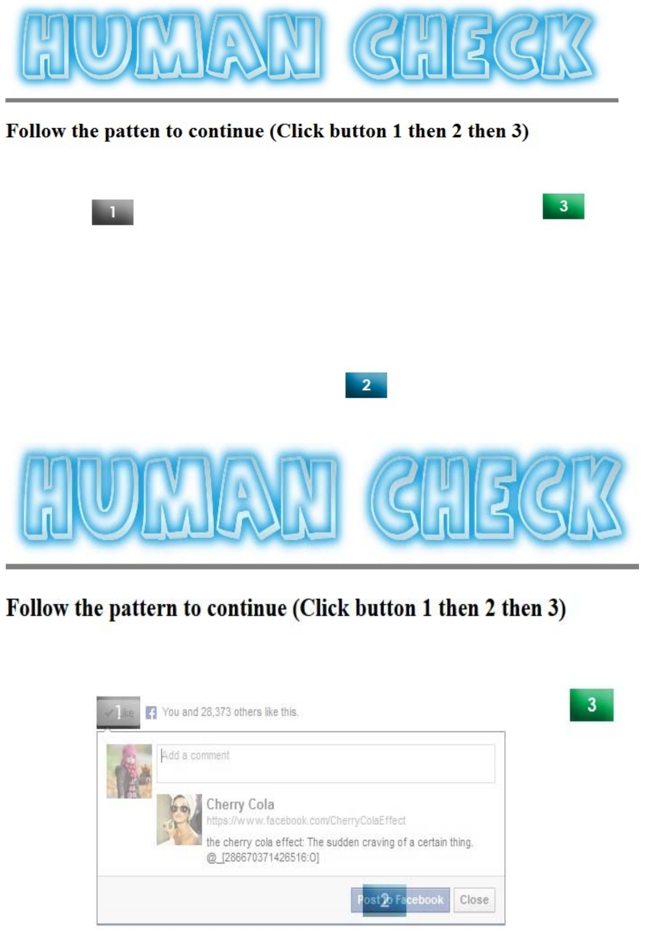

HUMAN CHECK
Follow the patten to continue (Click button 1 then 2 then 3)
1
3
2
HUMAN CHECK
Follow the pattern to continue (Click button 1 then 2 then 3)
You and 28,373 others like this.
Add a comment
Cherry Cola
https://www.facebook.com/CherryColaEffect
the cherry cola effect: The sudden craving of a certain thing.
@_[286670371426516:0]
Post to Facebook
Close
3

Figure 1: Use of CAPTCHA. Likejacking Attack in which a frame was embedded underneath the buttons

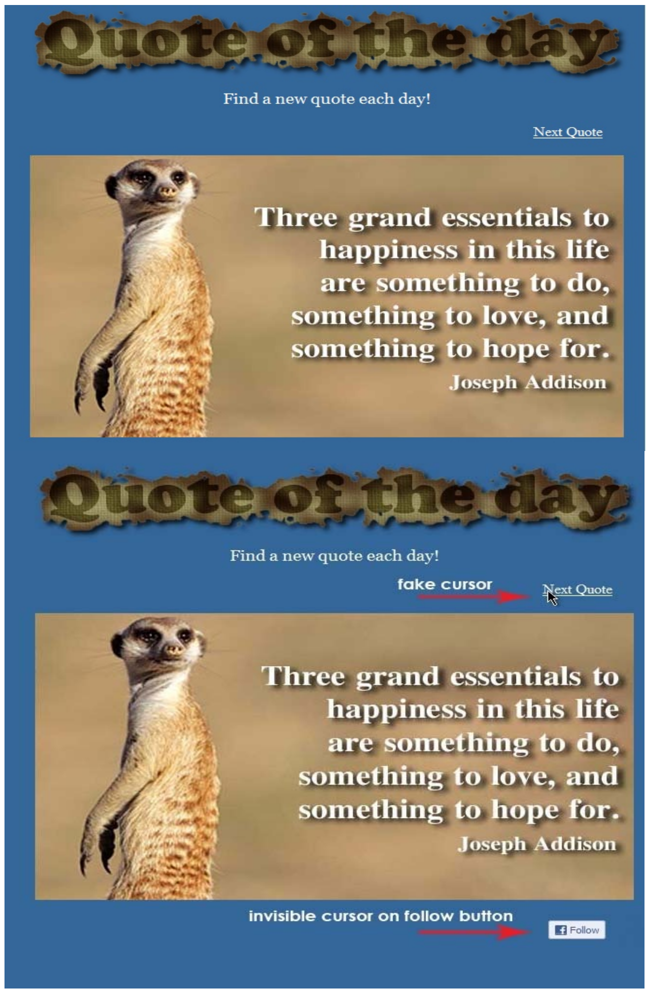

Quote of the day
Find a new quote each day!
Next Quote
Three grand essentials to happiness in this life are something to do, something to love, and something to hope for.
Joseph Addison
Quote of the day
Find a new quote each day!
fake cursor
Next Quote
Three grand essentials to happiness in this life are something to do, something to love, and something to hope for.
Joseph Addison
invisible cursor on follow button
Follow

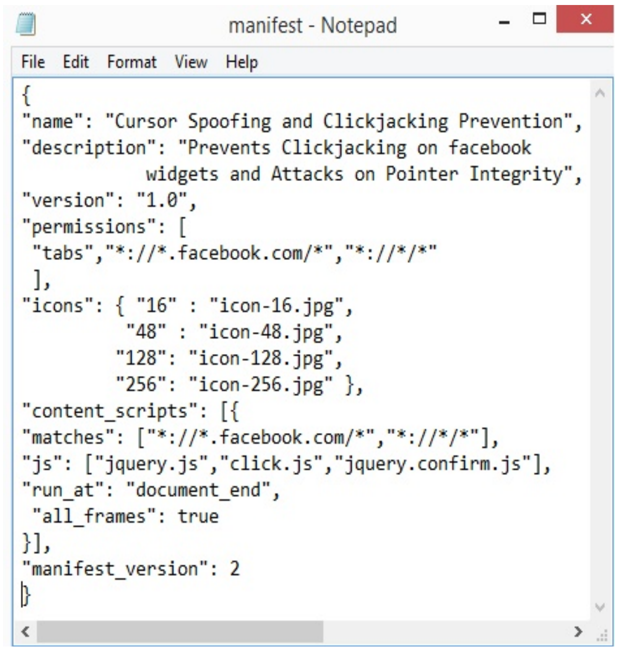
Figure 2: User Interest. Cursor Spoofing Attack to use the follow button

B. User Interest

The *user interest* attack is normally launched by the people who know you personally, have knowledge about your hobbies and area of your interest. In technical words social engineering is required to launch such kind of attack because people have different interests. Normally the areas of interest are identified from the daily activities on social networking websites as *Facebook* and *Twitter*. The attack is based on cursor spoofing. A web site is created keeping victim in mind as genuine web page. The user when enter the web page as *Daily Quote* [14], *A quote of the day* will be displayed. The user may be interested to read next quote due to which he or she clicks on the *next quote* button. The cursor that clicks on the *next quote* button is a fake cursor. The actual cursor is spoofed and invisible, clicks on the *follow* button of the *Facebook*. The user follows person or page on the *Facebook* which he or she may not interest to *follow*. *Figure 2* shows the *Daily Quote* website that contains embedded follow button, real and fake cursor. The Daily Quote website allow the user to see only fake cursor and the actual cursor was hidden, use for malicious purposes.

V. OUR PROPOSED DEFENSE

The basic cause of Clickjacking attack is that the attacker application is present on sensitive user interface which the user needs to click. The attacker hides the frame beside the actual webpage and compromises visual and pointer integrity. For this reason a browser-based solution was proposed that prevent user against Clickjacking attacks.



```
{
"name": "Cursor Spoofing and Clickjacking Prevention",
"description": "Prevents Clickjacking on facebook
            widgets and Attacks on Pointer Integrity",
"version": "1.0",
"permissions": [
 "tabs","*://*.facebook.com/*","*://*/*"
 ],
"icons": { "16" : "icon-16.jpg",
          "48" : "icon-48.jpg",
         "128": "icon-128.jpg",
         "256": "icon-256.jpg" },
"content_scripts": [{
"matches": ["*://*.facebook.com/*","*://*/*"],
"js": ["jquery.js","click.js","jquery.confirm.js"],
"run_at": "document_end",
 "all_frames": true
}],
"manifest_version": 2
}
```

Figure 3: Manifest File of CSCP Extension

We created *Google Chrome extension* to prevent user against *Likejacking* and *Cursor Spoofing* attacks. Google Chrome was selected because it has just two extensions for the prevention of Clickjacking attack.

A. *Zscaler Likejacking Prevention*

The *Zscaler Likejacking Prevention* detects hidden Facebook widgets and warns users about Likejacking.

B. *I'd like to confirm*

Adds a confirm dialog to every Facebook *Like* button in order to prevent Clickjacking.

Our proposed defense covers the functionality of both the existing extensions and also ensures pointer integrity. Therefore we name it as *Cursor Spoofing* and *Clickjacking Prevention (CSCP). CSCP* has the functionality of detecting and preventing Clickjacking attacks on the *Facebook social plug-in*. When the pointer clicks on *like or follow* button, *pop-up* appear to the user that *you click on like or follow button*. When a *cursor spoofing* is detected on the websites, it displays both the fake and real cursors and warns the user that the website is compromised. *Figure 3* shows the manifest file of the proposed add-on.

VI. Experiment

This section provides details about our experimental setups. We hosted the *HumanCheck* [13] and *CursorSpoofing* [14] website online and allow different user to visit these websites. We also deployed these websites as a home page on each and every computer in a lab of 20 *PCs* using default web browser as Google chrome. Among them 10 *PCs* browser was set to default home page of *HumanCheck* webpage. Remaining 10 PCs default home page was set as *CursorSpoofing webpage*. There was 442 participant age between 16 years to 30 years. The experiment was analyzed for ten days. *Table 1* shows the success rate of both the attacks without any defenses. The log of each web site was maintained using File Transfer Protocol (FTP), which generate two files for each web page. One file contains the counter that increment when a user visits the web page [23, 24]. Another file contains the counter that increments when a user clicks on the *like* or *follow* button [25, 26].

Table 1: Success Rate of Attack Before CSCP

Attack	Total Number Of Visit	Number of People Clickjacked	Attack Success Rate	CSCP Extension Enable
Human Check (Likejacking)	*241*	*189*	*78.9%*	×
Daily Quote (Cursor Spoofing)	*202*	*155*	*76.7%*	×

After analysis of attack for five days we then enable the *CSCP* extension in each computer and start analyzing the log of each website for the next five days. That results a low success rate of attacks as compared to our previous analysis. The lower success rate is due to adding two features of *CSCP* extension.

First, when a user clicks on a hidden button inside an *iframe* it provides a confirmation box with the message, *you have clicked a hidden button! Are you sure you want to continue?* If the user clicks on *cancel,* the behavior of the hide button will be discarded. Otherwise if user clicks on the *yes* button then the default actions of the social plug-in take place. Secondly the extension checks each website for *cursor spoofing*. If cursor spoofing detected then the *CSCP* shows the actual and fake cursor both and thus the user is informed that the website are compromised. *Table 2* shows the success rate after enabling CSCP.

Table 2: Success Rate of Attack After CSCP

Attack	Total Number Of Visit	Number of People Clickjacked	Attack Success Rate	CSCP Extension Enable
Human Check (Likejacking)	*233*	*53*	*22.7%*	✓
Daily Quote (Cursor Spoofing)	*209*	*21*	*10.0%*	✓

The *CSCP* extension for Google chrome creates a big difference. The attack success rate reduces from 78.4% to 22.7% for *HumanCheck* website. And for *CursorSpoofing* attack the success rate reduces from 76.7% to 10.0%.

VII. Conclusion

We have proposed new variants of attacks which are harmful than the existing Clickjacking attacks. The attacks cause the loss of human credential and also reputation of the user in the community by sharing illegal material on OSNs. In order to defend against those attack. We have proposed a web based solution in the form of *CSCP Google Chrome extension* that ensures defense against clicking on the embedded sensitive user interface. The extension provides protection against visual integrity and pointer integrity. The *CSCP* has effective prevention rate of 56% to 67% for the current and newly proposed Clickjacking attack.

Acknowledgment

The research is sponsored by SecureDial Project funded by Higher Education Commission (HEC), Islamabad-Pakistan.

References

[1] Paul Stone, 2010, Next Generation Clickjacking: New attacks against framed web pages. [Online] Black Hat Europe. Available at: <*https://www.blackhat.com/html/bh-eu-10/bh-eu-10-archives.html#Stone*> [Accessed February 19, 2013]

[2] Clickjacking <*http://en.wikipedia.org/wiki/Clickjacking*> [Accessed Dec 01, 2012]

[3] Lin-Shung Huang, Alex Moshchuk, Helen J. Wang, Stuart Schechter, Collin Jackson, 2012, Clickjacking: Attacks and Defenses. [Online] *Available at:* <*http://dl.acm.org/citation.cfm?id=2362815*> [Accessed February 19, 2013]

[4] Sebastian Lekies, Mario Heiderich, Dennis Appelt, Thorsten Holz, Martin Johns, 2012, On the fragility and limitations of current Browser-provided Clickjacking protection schemes. [Online] Available at: <*https://www.usenix.org/system/files/conference/woot12/woot12-final16.pdf*> [Accessed January 15, 2013]

[5] A.Sankara Narayanan, 2012, Clickjacking vulnerability and countermeasures. [Online] New York: International Journal of Applied Information Systems. Available at: *http:// research.ijais.org/volume4/number7/ijais12-450793.pdf*

[6] Marco Balduzzi, Manuel Egele, Engin Kirda, Davide Balzarotti, Christopher Kruegel, 2010, A Solution for the Automated Detection of Clickjacking Attacks. [Online] Available at: <*http:// www.iseclab.org/papers/asiaccs122-balduzzi.pdf*> [Accessed January 15, 2013]

[7] Brigette Lundeen, Dr. Jim Alves-Foss, 2012, Practical Clickjacking with BeEF. [Online] Homeland Security: IEEE Conference on Technologies. Available at: <*http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6459919&tag=1*> [Accessed February 19, 2013]

[8] Rich Lundeen, Jesse Ou, Travis Rhodes, 2011, New Ways I'm Going to Hack Your Web App. [Online] Black Hat Abu Dhabi. Available at: <*https://www.blackhat.com/html/bh-ad-11/bh-ad-11-archives.html#Lundeen*> [Accessed January 15, 2013]

[9] Gustav Rydstedt, Elie Bursztein, Dan Boneh, Collin Jackson, 2010, Busting Frame Busting: A Study of Clickjacking Vulnerabilities on Popular Sites. [Online] Oakland: IEEE Web 2.0 Security and Privacy. Available at: <*http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.182.5269*> [Accessed January 15, 2013]

[10] Paul Stone, 2010, Next Generation Clickjacking: New attacks against framed web pages. [Online] Black Hat Europe. Available at: <*https://www.blackhat.com/html/bh-eu-10/bh-eu-10-archives.html#Stone*> [Accessed January 15, 2013]

[11] Advance Cursor jacking <*http://podlipensky.com/examples/clickjacking/advanced-cursorjacking.html*> [Accessed Dec 01, 2012]

[12] Cursor jacking <*http://podlipensky.com/examples/clickjacking/cursorjacking-js.html*> [Accessed Dec 01, 2012]

[13] HumanCheck <*http://chekkers.atwebpages.com/humancheck.php*> [Accessed June 06, 2013]

[14] CursorSpoofing <*http://chekkers.atwebpages.com/cursor.php*> [Accessed June 06, 2013]

[15] Popular Social Networking Sites Ranking <*http://www.ebizmba.com/articles/social-networking-websites*> [Accessed May 15, 2013]

[16] Clickjacking <*https://www.owasp.org/index.php/Clickjacking*> [Accessed Dec 01, 2012]

[17] Adobe Flash Object and Embed tag attributes <*http://helpx.adobe.com/flash/kb/flash-object-embed-tag-attributes.html*> [Accessed March 13, 2013]

[18] E. Vela, CSS Attacks. <*http://sirdarckcat.blogspot.com/2008/10/aboutcss-attacks.html*> [Accessed March 13, 2013]

[19] M. Zalewski, Firefox vulnerability <*http://seclists.org/fulldisclosure/2007/Feb/226*> [Accessed March 13, 2013]

[20] M. Zalewski, The strokejacking <*http://lcamtuf.blogspot.com/2010/06/curse-of-inverse-strokejacking.html*> [Accessed March 13, 2013]

[21] J. Ruderman. Security Dialogs <*http://www.squarefree.com/2004/07/01/race-conditions-in-security-dialogs/*> [Accessed March 13, 2013]

[22] M. Zalewski, User Interfaces for non-robots <*http://lcamtuf.blogspot.com/2010/08/ondesigning-uis-for-non-robots.html*> [Accessed March 13, 2013]

[23] Log of Users Visited Human Check Website. Available at: < http://chekkers.atwebpages.com/count/access_h.txt>

[24] Log of Users Visited Daily Quote Website. Available at: < http://chekkers.atwebpages.com/count/access.txt>

[25] Log of Users having CSCP extension and Likejacked. Available at: < http://chekkers.atwebpages.com/count/counter_h.txt>

[26] Log of Users having CSCP extension and became Victum of Cursor Spoofing. Available at: <http://chekkers.atwebpages.com/count/counter.txt>