Int. J. Advanced Networking and Applications
Volume: 10 Issue: 01 Pages: 3735-3740 (2018) ISSN: 0975-0290

Clickjacking Attack: Hijacking User's Click

Kokila Jamwal
M.Tech. Student (4th Sem.)
Department of Computer Science & IT, University of Jammu, J & K, India
Email: kokilajamwal@gmail.com
Lalit Sen Sharma
Professor
Department of Computer Science & IT, University of Jammu, J & K, India
Email: lalitsen.sharma@gmail.com

ABSTRACT

The cyber attacks have become most prevalent in the past few years. During this time, attackers have discovered new vulnerabilities to carry out malicious activities on the internet. Both the clients and the servers have been victimized by the attackers. Clickjacking is one of the attacks that have been adopted by the attackers to deceive the innocuous internet users to initiate some action. Clickjacking attack exploits one of the vulnerabilities existing in the web applications. This attack uses a technique that allows cross domain attacks with the help of user-initiated clicks and performs unintended actions. This paper traces out the vulnerabilities that make a website vulnerable to clickjacking attack and proposes a solution for the same.

Keywords - **clickjacking, cursorjacking, frame busting, iframe, X-Frame-Options.**

Date of Submission: May 29, 2018 Date of Acceptance: June 23, 2018

1. INTRODUCTION

Along with the growth of Internet, web applications have also evolved from simple collections of static HTML documents to complex dynamic pages. Web applications with highly sophisticated user interfaces have become possible due to client-side and server-side scripting languages. But on the other hand, these technologies have been exploited by the attackers as well. Attackers have managed to perform different attacks on the web applications using these technologies, necessitating focus on the web security. Attackers always keep looking for bugs, vulnerabilities and loopholes in the web applications, browser, servers and other components of the web to carry out a malicious activity. Internet Security Threat Report 2011 by Symantec [1] concludes that approximately 95 new vulnerabilities are discovered every week. Website Security Statistics Report 2015 by WhiteHat [2] shows that 64% of the Public Administration sites, 55% of the Transportation and Food services sites are always vulnerable. 86% of web applications had serious issues with authentication, access control, and confidentiality as per HP 2015 Cyber Risk Report [3]. According to 2016 Vulnerability Statistics Report by Edgescan [4], 61% of the web application vulnerabilities lead to browser attacks. Exploitation of one such vulnerability leads to an attack, called the *clickjacking* attack.

The term '*Clickjacking*' was coined by Jeremiah Grossman and Robert Hansen in the year 2008. According to them, *Clickjacking* is a technique where cross-domain attacks are perpetrated by hijacking user-initiated clicks to perform unintended actions [5]. *Clickjacking* attack is a technique that entices the victim into clicking on a specific element of a webpage, while the victim intends to interact with the content of a different website. The victim clicks on an element of the attacker's choice under the misconception of clicking on an apparently harmless page. In order to perform the attack, a malicious website (attacker's website) loads a page from the target website inside an *iframe*. Cascading Style Sheets (CSS) can be used to hide everything except the targeted region of the page. The targeted region can either be displayed as a part of the attacker's page, known as User Interface redressing or made fully transparent and placed on top of another element on the attacker's page [6]. All the major browsers like Firefox, Chrome, Opera, etc have been the victims of *clickjacking* attack. Social networking websites have been one of the mostly attacked groups of web applications. "Twitter bomb" is known to be one of the worst *clickjacking* attacks, apart from the attack on Adobe Flash. 30% of Alexa Top 10 web sites, 70% of Top 20 bank web sites, and 80% of 5 popular open-source web applications have no defense against *clickjacking* in 2012 [7].

1.1 Types of Clickjacking

The *clickjacking* attack aims to trick the victim to perform an action which he/she doesn't intend to. Existing *clickjacking* attacks are classified as [8]:

1.1.1 Compromising target display integrity

- Hiding the target element- Attacker hides the target element using HTML/CSS styling, but mouse events still work. Attacker can carry out this attack by making the target element transparent by placing it within a division and setting the opacity value to zero.
- Partial overlays- Attacker baffles a victim by making only a part of the target element opaque. The *z-index* property of CSS can be used for this overlaying thus compromising target display integrity.
- Cropping- Attacker crops the target element to show only a piece of the target element.

1.1.2 Compromising pointer integrity

Attacker violates pointer integrity by displaying a fake cursor and hides the actual default cursor, known as *cursorjacking*. This way victim misinterprets a click's target.

1.1.3 Compromising temporal integrity

Attacker manipulates a UI element after the user decides to click. But before the actual click occurs the attacker could move the target element on top of a button shortly after the victim hovers the cursor over the button, in anticipation of the click.

Attackers have compromised the context integrity of web applications in the past and continue to do so; providing new terms for different variants of the attack such as likejacking, strokejacking, drag-and-drop *clickjacking*, etc.

2. AIMS AND OBJECTIVES

The objective of this study is to find out the vulnerabilities leading to *clickjacking* attack in the web applications. The study aims to study the mitigation techniques of the attack and propose a solution for the same.

3. RELATED WORK

The goal of *clickjacking* attack is to trick the victim into performing unintended actions by hijacking the victim's clicks. Various researchers have worked in the field of detection and prevention of *clickjacking* attack. Study in this domain is still being carried out by many researchers. Some of them are listed as:

Lin-Shung Huang and et al. [9] devised new variants of *clickjacking* attacks and demonstrated the inefficiency of existing countermeasures in mitigating those attacks. They evaluated the existing techniques of the attack prevention and proposed a mechanism called "*InContext*" to ensure visual and temporal integrity of websites.

R. P. Seenivasan and K. Suresh Joseph [10] proposed a system to prevent *clickjacking* attacks. They evaluated the proposed anti-*clickjacking* technique with security and cost metrics.

Dipti Pawade and et al. [11] developed a JavaScript based browser extension called "*ClickProtect*" for Google Chrome which provides a solution for multiple *clickjacking* attacks. It warns the user before he/she proceeds towards an unsafe action (click) with a pop-up warning message.

G. Rydstedt and et al. [12] surveyed *frame busting* code the Alexa Top-500 websites including banks, social networks, online merchandise, trading and gaming sites. They proposed a JavaScript-based defense against *clickjacking*.

Brigette Lundeen and Jim Alves-Foss [13] presented a plug-in module for BeEF (Browser Exploitation Framework) which provides a way for penetration testers to easily demonstrate the impact of *clickjacking* vulnerabilities. It is a tool designed to help professional penetration testers easily demonstrate the impact of client-side security vulnerabilities.

Daehyun Kim and Hyoungshick Kim [14] investigated how vulnerable Korean websites were to *clickjacking* attacks by performing real attacks on the top 500 most popular Korean websites as well as all of the financial websites. They also investigated top 500 global websites for *clickjacking* vulnerability. They identified the type of websites most vulnerable to *clickjacking* attack. They used two browsers to check the vulnerability: Internet Explorer8 and Google Chrome. They analyzed the attack failure rates based on the type of website and also on the different countries of the world. The investigation showed that 99.6 % of the Korean websites were vulnerable to *clickjacking* attack which was more than the global websites, with vulnerability as 78%.

A. Sankara Narayanan [15] provided a clear view of *clickjacking* attack using various code examples. Study provided the numeric figures for the *clickjacking* google results. He compared the numeric figures with other browser-based attacks on the basis of google results in the past few years. He implemented *clickjacking* attack using various available online tools and mitigated the attack using available anti-*clickjacking* tools.

Yusuke Takamatsu and Kenji Kono [16] proposed an automated tool called "*Clickjuggler*" for checking defenses against visual *clickjacking* during development. They implemented Clickjuggler as a plug-in for Firefox 20.0.1 and 3.6.8 using Firefox plug-in interface. They applied Clickjuggler to four real-world web applications including Joomla, WordPress, MediaWiki and Roundcube. They compared Clickjuggler to other tools like CJTool and BeEF plug-in.

M. Balduzzi and et al. [17] proposed a solution for the automated and efficient detection of *clickjacking* attacks. They developed a browser plug-in consisting of a detection unit and a testing unit. They analysed more than a million unique Internet web pages to estimate the prevalence of *clickjacking* attacks. The analysed to what extent *clickjacking* defending techniques have been adopted. As a result 7% of visited web pages did not contain any clickable elements. 37.3% of visited websites contained at least one *iframe* tag making the web pages vulnerable to *clickjacking* attack.

4. EXPERIMENTAL SETUP

In this study, a web application, named SOCIO has been developed in php. SOCIO is a sample social networking web application developed for attack purpose; which provide features like instant messaging and public posts. This web application has been hosted on the local host using XAMPP server. Various variants of the *clickjacking* attack like basic *clickjacking*, multiple-*iframe clickjacking* and *cursorjacking* have been carried out on this web application to perform unintended actions. Visual integrity and pointer integrity of the web applications have been compromised. Other experiments such as checking top websites of J&K for *clickjacking* vulnerability and study on behavior of the mitigation techniques on different browsers have also been performed. Modern browsers like Google Chrome64, IE11, UC Browser7 and Firefox59 have been used to carry out these experiments. The Fig. 1 shows the architecture of *clickjacking* for carrying out *clickjacking* on the target web application on the local host.

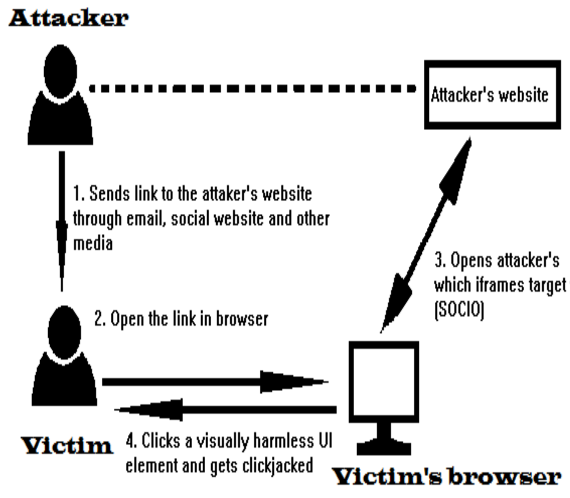

Attacker
Attacker's website
1. Sends link to the attaker's website through email, social website and other media
2. Open the link in browser
3. Opens attacker's website which iframes target website (SOCIO)
Victim
4. Clicks a visually harmless UI element and gets clickjacked
Victim's browser

Fig. 1: Architecture of clickjacking attack

Apart from the basic *clickjacking* attack, other variants of the attack have been performed through the target web application.

5. EXPERIMENTAL WORK

The experiments have been carried out on the sample social networking web application. The attacks that have been performed on SOCIO are explained below:-

- Basic *Clickjacking*-In SOCIO, basic *clickjacking* has been performed by sending a link to the victim through both, public post and message. The attack is used by the attacker to make the victim change his/her profile picture.
- *Clickjacking* on nested *iframes*- Nested *iframes* has been used by the attacker to carry out the attack.
- *Cursorjacking*- Introduced by Eddy Bordi, *cursorjacking* deceives the victim by using a fake cursor to perform the attack. In order to make the attack successful, original mouse cursor has been made invisible. In SOCIO, *cursorjacking* leads to download a malicious file.
- *Cookiejacking*- In SOCIO, victim's cookies have been stolen using this variant of *clickjacking*. This is done using cross-site scripting along with *clickjacking*.

These attack variants have been successfully performed on the developed web application using all the major modern browsers.

After performing the attacks successfully, the available solutions to mitigate the attack have been applied to the SOCIO web application one-by-one. These have been listed below:-

- *X-Frame-Options*- Introduced by Microsoft, *X-Frame-Options* HTTP header is a response header that prevents framing of a web application. Setting *X-Frame-Options* allows a web application to specify if it can be iframed or not. The possible values for *X-Frame-Options* are:-
  DENY: Application cannot be displayed from any web page.
  SAMEORIGIN: Application can be displayed from web pages with the same origin.
  ALLOW-FROM: Application can be displayed from specified URI only.

To mitigate the attack on SOCIO, following line was added in the .htaccess file in the directory of the web application:-

Header set X-Frame-Options DENY

The SAMEORIGIN value for *X-Frame-Options* was also implemented by using the following line:-

Header set X-Frame-Options SAMEORIGIN

- *Frame busting*- *Frame busting* is a technique that uses JavaScript code to prevent *clickjacking*. In SOCIO, different *frame busting* codes have been implemented to mitigate the *clickjacking* attack.

Experiments have also been carried to check the web sites vulnerable to the *clickjacking* attack among the top 50 websites of J&K, as shown in Table 1. Another experiment involving the behavioural study of different browsers for the mitigation solutions has been performed, the results of which has been summarized in Table 2.

6. PROPOSED SOLUTION

Most of the *clickjacking* attacks are carried out using the transparent layers of *iframe*. So, the proposed solution detects such transparent *iframes* and makes them opaque and warns the user about possible *clickjacking* attack. The solution also works to detect the *cursorjacking* attack by checking if the default mouse pointer is made hidden. The proposed solution has been implemented as a JavaScript-based extension. This extension consists of two units: Detection Unit and Action Unit. Detection unit follows pre-defined steps to detect the presence or absence of the attack and the action unit takes certain actions to successfully mitigate the attack. The flowchart of the proposed solution is explained in Fig. 2.

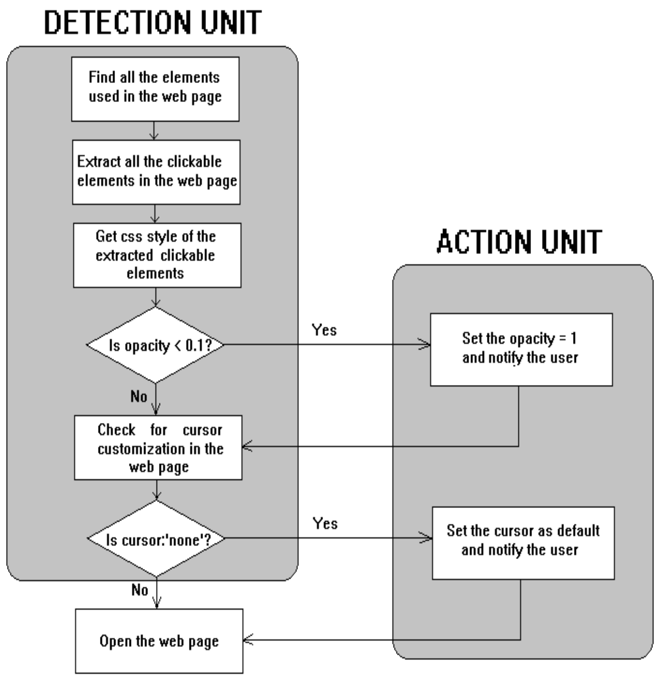

DETECTION UNIT
Find all the elements used in the web page
Extract all the clickable elements in the web page
Get css style of the extracted clickable elements
Is opacity < 0.1?
Yes
No
Check for cursor customization in the web page
Is cursor:'none'?
Yes
No
Open the web page
ACTION UNIT
Set the opacity = 1 and notify the user
Set the cursor as default and notify the user

Fig. 2: Flowchart of the proposed solution

7. RESULT AND DISCUSSION

By performing these experiments, various results have been gathered. The experiment to find out the number of websites from Jammu and Kashmir (J&K) vulnerable to *clickjacking* attack has been carried out on all the major browsers. The result of the experiment has been summarized in the TABLE 1 below.

Table 1: Number of websites vulnerable to *clickjacking*

Browser	Number of websites (out of 50) vulnerable to clickjacking
Google Chrome 64	44
Internet Explorer 11	44
Mozilla Firefox 59	43
UC Browser 7	44

The result of this experiment points to the huge possibility of *clickjacking* attacks on the websites of J&K. This is alarming that only 4 out of the 50 websites are completely safe (in all the tested browsers) from *clickjacking* attack. Among those three, one website has *X-Frame-Options* set to SAMEORIGIN and other two have implemented *frame busting* codes.

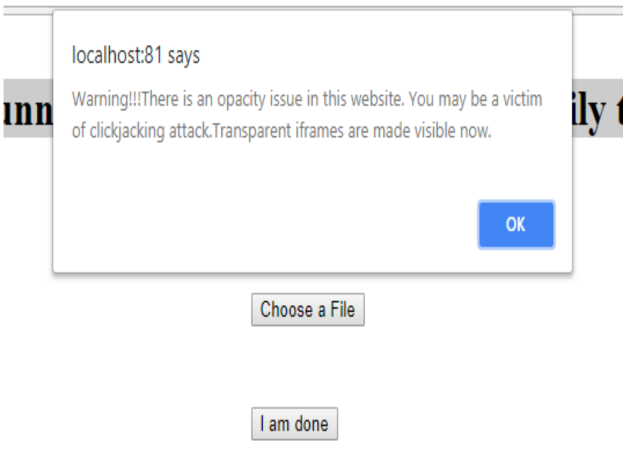
Another experiment involving the study of behaviour of browsers for different values of *X-Frame-Options* header has been carried out successfully. TABLE 2 shows whether the attack has been successfully mitigated or not.

Table 2: Behaviour of browsers for different values of *X-Frame-Options* header

Browser	*X-Frame-Options* set to DENY	*X-Frame-Options* set to SAMEORIGIN	*X-Frame-Options* set to SAMEORIGIN for nested *iframes*
Google Chrome 38.0	Yes	Yes	No
Google Chrome 64	Yes	Yes	Yes
Internet Explorer 11	Yes	Yes	No
Microsoft Edge 16	Yes	Yes	No
Mozilla Firefox 58	Yes	Yes	No
Mozilla Firefox 59	Yes	Yes	Yes
UC Browser 7	Yes	Yes	No

Apart from these experiments, the proposed solution has been implemented to check the effectiveness of the solution.

The web pages with *clickjacking* attack have been checked with the JavaScript-based extension, the results of which have been shown in the Fig. 3, Fig. 4, Fig. 5 and Fig.6.


localhost:81 says

Warning!!!There is an opacity issue in this website. You may be a victim of clickjacking attack.Transparent iframes are made visible now.

Fig.3: Alert generated by the proposed extension regarding opacity issue

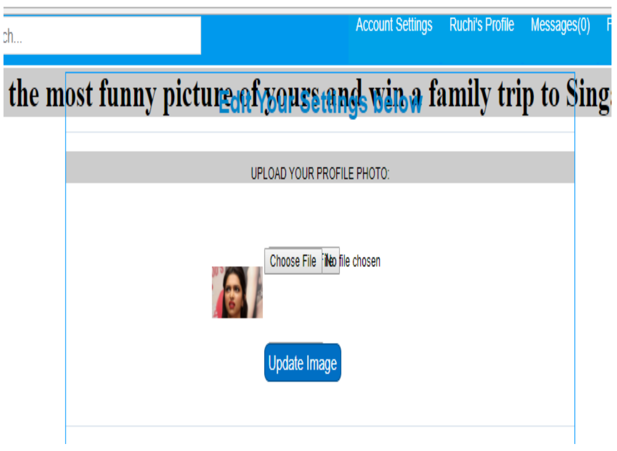
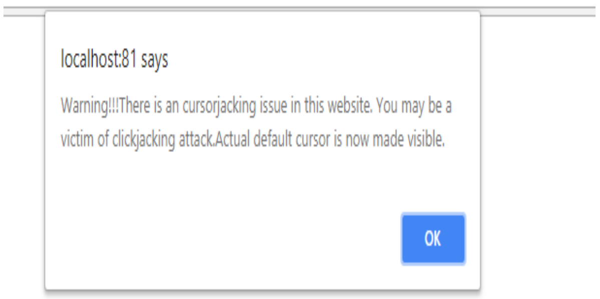

Account Settings
Ruchi's Profile
Messages(0)
the most funny picture of yours and win a family trip to Sing
Edit Your Settings Below
UPLOAD YOUR PROFILE PHOTO:
Choose File No file chosen
Update Image

Fig.4: Transparent element (iframe) is made visible by the proposed extension


localhost:81 says

Warning!!!There is an cursorjacking issue in this website. You may be a victim of clickjacking attack.Actual default cursor is now made visible.

Fig.5: Alert generated by the proposed extension regarding cursorjacking

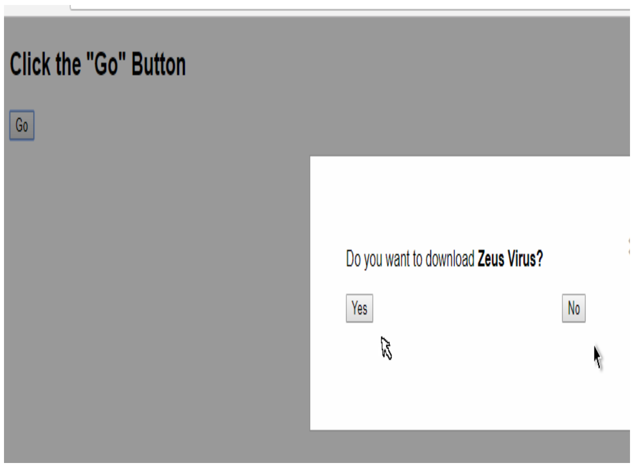


Fig.6: Actual cursor is made visible by the proposed extension

The results show that the proposed solution for *clickjacking* attack successfully predicts the attack and then prevents the attack by alerting the user. The proposed solution has been successfully implemented in Chrome, Firefox, UC Browser and Opera. The effectiveness of the proposed solution has been checked on 20 web pages, from both local host and on internet. The confusion matrix for the same is presented in TABLE 3 below.

Table 3: Confusion Matrix for the proposed solution

Confusion Matrix		Predicted Class By the Proposed Extension	
		Clickjacking Present	Clickjacking Absent
Actual Class	Clickjacking Present	13	1
	Clickjacking Absent	0	6

The confusion matrix gives 13 True Positive (TP), 1 False Negative (FN), 0 False Positive (FP) and 6 True Negative (TN) cases. Therefore, precision, recall and F-measure for the proposed solution are 1, 0.929 (approx.) and 0.963 (approx.) respectively. With 0.963 (which is close to 1) F-measure value, the proposed solution works effectively against *clickjacking* attack.

8. CONCLUSION

The probability of *clickjacking* attacks is quite high due to lack of awareness about the attack among a large group of users. Even though there have been a variety of techniques to mitigate *clickjacking* attack, but the attackers have already bypassed those techniques in one way or the other. *Clickjacking* is one of the emerging attacks that can result in serious problems in the field of web security, in the near future. Attackers are always in a constant search of methods and techniques that can be used to carry out malicious activities on internet. There is a huge need of developing efficient techniques, both on client-side and server-side to protect the web users from *clickjacking* attack and it ever-emerging variants.

REFERENCES

[1] Symantec Corporation, Internet Security Threat Report, 2012. [Online]. Available: http://www.symantec.com/threatreport/

[2] WhiteHat Security, Inc., Website Security Statistics Report 2015, Santa Clara, CA 95054, 2015.

[3] HP Security Research, “Cyber Risk Report”, 2015.

[4] BCC Risk Advisory Ltd., 2016 Vulnerability Statistics Report Edgescan, 2016. [Online]. Available: http://www.edgescan.com

[5] Robert Hansen and Jeremiah Grossman, Explanation of Clickjacking. [Online]. Available: http://www.sectheory.com/clickjacking.htm

[6] Context Information Security Ltd, Next Generation Clickjacking, London, 2010. [Online]. Available: http://www.contextis.co.uk

[7] Dingjie Yang, Clickjacking: An Overlooked Web Security Hole, 2012. [Online]. Available: https://blog.qualys.com/securitylabs/2012/11/29/clickjacking-an-overlooked-web-security-hole

[8] Lin-Shung Huang, Alex Moshchuk, Helen J. Wang, Stuart Schechter and Collin Jackson, Clickjacking: Attacks and Defenses, *Proc. USENIX Security Symposium*, Bellevue, WA, 2012, 413-428.

[9] Hanqing Wu and Liz Zhao, Clickjacking in *Web Security: A WhiteHat Perspective* (New York, NY, USA: CRC Press, 2015) 141-156.

[10] R. P. Seenivasan and K. Suresh Joseph, A Survey of Clickjacking Attack and Countermeasures in Web Environment, *International Journal of Advanced Research in Computer Science and Software Engineering*, *6*(12), 2016, 206-213.

[11] Dipti Pawade, Era Johri, Divya Reja and Abhilasha Lahigude, Implementation of Extension for Browser to Detect Vulnerable Elements on Web Pages and Avoid Clickjacking, *Proc. 6th IEEE International Conf. on Cloud System and Big Data Engineering*, Noida, India, 2016, 226-230.

[12] G. Rydstedt, E. Bursztein, D. Boneh and C. Jackson, Busting frame busting: a study of clickjacking vulnerabilities at popular sites, *Proc. IEEE Web 2.0 Security and Privacy*, Oakland, CA, 2010, 1-13.

[13] Brigette Lundeen and Jim Alves-Foss, Practical Clickjacking with BeEF, *Proc. IEEE Conf. on Technologies for Homeland Security (HST)*, Massachusetts, USA, 2012, 614-619.

[14] Daehyun Kim and Hyoungshick Kim, Performing clickjacking attacks in the wild: 99% are still vulnerable!, *Proc. IEEE 1st International Conf. on Software Security and Assurance*, Suwon, South Korea, 2015, 25-29.

[15] A. Sankara Narayanan, Clickjacking Vulnerability and Countermeasures, *International Journal of Applied Information Systems*, *4*(7), 2012, 7-10.

[16] Yusuke Takamatsu and Kenji Kono, Detection of Visual Clickjacking Vulnerabilities in Incomplete Defenses, *IEEE Journal of Information Processing, 23*(4), 2015, 513-524.

[17] M. Balduzzi, M. Egele, E. Kirda, D. Balzarotti and C. Kruegel, A Solution for the Automated Detection of Clickjacking Attacks, *Proc. 5th ACM Symposium on Information, Computer and Communications Security*, Beijing, China, 2010, 135–144.

AUTHORS BIOGRAPHY

Kokila Jamwal is a Master of Technology (MTech) student in the Department of Computer Science & IT, University of Jammu. She received her undergraduate Degree in Information Technology (B.E (IT)) from Mahant Bachittar Singh College of Engineering and Technology, Jammu in the year 2016. Her research interests are in the field of Networking and Network Security.

Lalitsen Sharma received his doctorate in Computer Science and Engineering from Guru Nanak Dev University Amritsar, Punjab, India. He also holds master's degree in Mathematics and Computer Applications from the same university. He has been teaching to post graduate students in computer applications of University of Jammu for more than 20 years. He is a life member of India Science Congress Association, Computer Society of India, Institute of Electronics and Communication Engineers and National HRD Network, India. He is specialized in Data Communication and Network, Internet and WWW and Data Structures. He has completed one major research project to trace out vulnerabilities in network applications funded by University Grants Commission, Ministry of Human Resource Development, Govt. of India. He has produced one PhD and is currently supervising Five PhD scholars.