# A Brief Survey of Detection and Mitigation Techniques for Clickjacking and Drive-by Download Attacks

Tatwadarshi P. Nagarhalli
P. G. Student
Shree L.R. Tiwari College of Engineering
Mumbai, India

J.W. Bakal, PhD
Principal,
Shivajirao S. Jondhale College of Engineering,
Mumbai, India

Neha Jain
Asst. Prof.
Shree L.R. Tiwari College of Engineering
Mumbai, India

## ABSTRACT
With the advent of the World Wide Web the whole world became closure to each other. Further it has provided a medium to socialise over long distances. This has further abated the growth of many social media platforms. The social media platforms have brought many, even the non tech-savvy user on the internet. So, social media platform users have become an easy targets of the attackers and hackers who exploit the vulnerabilities of users, including the web browsers. Clickjacking and drive-by downloads have become a popular tools through which the attackers try to exploit the users. This paper takes a look at the different systems that have been proposed to detect, mitigate and prevent clickjacking and drive-by download attacks.

## Keywords
Clickjacking, Drive-by Download, Social Media Attacks.

## 1. INTRODUCTION
The rapid proliferation of internet usage has made interaction between people, even living very far apart, possible on click of a button. One of the most popular medium for interaction is the social media. Social networking has become a popular way for users to meet and interact online. Users spend a major amount of time on different social network platforms such as Facebook, Twitter, Google Plus, etc. Intentionally or unintentionally sharing a significant amount of their private and often sensitive information on the social network [9]. Because of this social networks have become a treasure chest of private and sensitive data. Also, it has been seen that social network users tend to have a very high level of blind trust toward other social network users. They friend requests rather easily, and also trust thing, attachments that their friends send to them with closed eyes. This information of the blind faith shown by the users on their social networks and the kind of information that they share attracts the interest of cybercriminals. Attackers uses a several modern attack techniques on social networking sites like clickjacking, malicious browser extensions via drive-by-downloads, URL shorteners and socially engineered script injection.

In clickjacking an attacker uses transparent layers to trick the user into clicking on a button or a link on another page when in fact they wanted to click on a top level page which was not supposed to be harmful. In Drive-by download attacks just by visiting a malicious web site will lead to the download and subsequent execution of malicious and harmful software on the visitor's computer. After downloading, the application is invoked and is free to perform its immoral purposes. URL shortening services replaces long and difficult to remember URLs with shorter ones, which can be easily remembered, and subsequently redirects all requests for this short- ended URL to the original long URL. On the internet there are number of free and open source software's that provide this service. And, source code for any software that is created under this license is open for anyone to modify, hack or build upon. In Cross Site Scripting (XSS) attack an attacker manages to inject Java script code or sometimes other code into a website causing it to execute the code.

Many solutions have been proposed to identify and mitigate these kind of attacks.

## 2. DIFFERENT ATTACKS THROUGH SOCIAL NETWORK

### 2.1 Clickjacking
Clickjacking is also known as UI-redressing. It is one of the most popular attacks on the internet. In Clickjacking, the attackers trick users to click on targets, this targets might to a different page. The Users assume that they clicked on buttons and they know then next intended actions, but they are unaware that they have actually clicked on invisible buttons which might be mapped to a remote page, set up by the attacker. This is generally done by using some invisible frames in overlaid frames in web page [1]. This is how the attackers try to steal a genuine mouse clicks and utilize it to perform some malicious tasks which are advantageous to them but harmful to the user. It has been found out that the attackers generally make use of the HTML Iframe element. An html <iframe> tag helps in embedding another HTML page into the current web page [2]. This property of the <iframe> tag is exploited by the attacker for clickjacking.

### 2.2 Drive-by download and Malicious Browser extensions
A drive-by download attacks one of the most easy and common attack used to spread malware or malicious codes over the internet. In drive-by download the malware is downloaded from the Internet into the user's computer without user's knowledge or alarm. Generally, in the recent times, the drive-by download target the add-on or plugins which are generally developed by any third party [4, 5]. One of the reasons for taking advantages of these add-ons and plugins might be that these third party softwares are less tested against different type of attacks that might take place on the said add-on and plugin. The attacker takes advantage of this and targets these unsuspecting small softwares. In some cases it is possible that the user, through ignorance, authorizes the malicious downloads without fully understanding the consequence of his actions.

### 2.3 URL Shorteners
URL shortening is a process of reducing the length of the Uniform Resource Locator (URL) as many a times it becomes difficult to keep in mind long URLs. In a very short time it

has become a very popular way escape from the often long URLs. URL shortening is technique through which the often long URLs which are difficult to remember are made substantially shorter, but still they redirect to the in intended or required page [7]. But on the flip side for any users it is difficult to determine where the URLs will eventually take the users to. The attackers generate post which might be based on some popular news or events, and then rather than using the real link, they use the fake shortened URL. When the user clicks on this link he is redirect to malicious web sited triggering a chain of actions. Also, in the social network people tend to blindly trust their co social network users. This trust is also exploited by the attackers, by posting a shortened URL on a friend's message board. This greatly increases the chances of the malicious link being clicked.

## 2.4 Socially engineered XSS

Cross-site scripting (XSS) is one of the most popular application layer attack techniques.

Generally a web page does not contain only a static page. A static page will have full control over its contents and hence not vulnerable to XSS attacks. To make the look and feel better a web page in most cases will contain many dynamic contents in it. If a web site contains dynamic pages the server has no control as to how the information is interpreted by the client's web browser.

With the help of the XSS the attacker will be able to inject malicious client side scripts into the victims' browser. The can further information about the client to the attacker, thus compromising the user's computer.

When one user is infected with XSS it spread fast to other users with the help of the web browser of the already infected user itself. And since the social networks heavily depends on the connectivity of the users the XSS virus have enough options to spread itself exponentially like a chain from one computer to another computer.

The Detection and Mitigation Techniques for Clickjacking and Drive-by Download attacks are:

# 3. CLICK JACKING

Lin-Shung Huang et. al. [10], have classified clickjacking into three type according to the type of integrity that has been compromised. These include compromising target display, compromising cursor integrity and compromising temporal integrity.

Compromising target display: The target display is compromised by misusing the feature of the <iframe> tag that the HTML provides. The existence is completely made invisible by making the div container opacity zero in the CSS. It is also possible for the attacker to crop some part of the genuine target element and embed the malicious code.

There many proposals to detect and mitigate attacks on target displays. For blacklisted domains Facebook has introduced conformation schemes for like buttons [11]. This technique works well in the specified domain. But the problem with this technique is that it is domain specific and it would have to be repeated each and every website specifically, which is a cumbersome job. Also, the user experience gets degraded and this is still vulnerable to double click attacks.

B. Hill [12] has proposed a method where the user interface is randomized. That is, the clickable areas in the web page will be different every time the page loaded. The major drawback of this approach is that the attacker may force the user to keep on clicking till the desired result has been achieved.

H. J.Wang et. al. [13], developed a new web browser, the Gazelle web browser. This web browser forces the opaque rendering of all the cross-origin frames. The advantage of this approach is that it makes all cross-origin frames to become visible and the user will be aware of where he is clicking. On the other hand the whole idea of <iframe> tag is defeated as the invisibility of all the frames is snatched.

Rather than disallowing the working of the <iframe> tag completely it is better to disable mouse click if the browser detects any cross-origin invisible content. The same concept is used by ClearClick, an extension available for Firefox browser which was introduced into the NoScript plugin [14, 16]. It tries to detect any mouse click event redirects to any invisible content or frame. If it does then the ClearClick prompts a warning to the user whether he wants to continue or not. Only after confirmation the invisible frame is rendered. But was observed that ClearClick generated many false positives, this is because ClearClick assumes that all cross-origin frames are a Clickjacking attempts, which in most cases are not. Thus making the user experience cumbersome.

To reduce the false positives Marco Balduzzi et. al. [15] introduce a new plug-in called the ClickIDS. The idea behind this plug-in is quite simple, this plugin checks whether there exists two or more clickable elements where the user intends to click or where the frames overlap on a particular page. For detection purpose this plug-in checks all the clickable elements, that is all the <a>, <embed>, menu, text fields, checkbox and radio buttons are scanned.

In ClickIDS whenever a page is loaded a page-handler routine is executed and it initiates a new routine to handle clicks called the click-handler routine. And whenever in a page the user clicks his click-handler routine registers the coordinates of the mouse click. After registering the coordinates the web page including the FRAME's are scanned and checked whether there is another clickable element at the same coordinates. If a clickable element is found at the same place then an alert is generated.

The ClickIDS works very well when attacks are based on overlapping elements. But, attacks based on partially obstructed pages is not identified by ClickIDS, unlike NoScript. Also, the number of false positives generated by is plug-in is rather high. So the authors propose to use the ClickIDS along with a slightly modified NoScript plugin. The modification that has been is that rather than generating a popup when an attack is detected the event is registered for further use of ClickIDS. When used both ClickIDS and NoScrip together the number false positives decreases drastically. One flaw that still remains is that this technique does not take into consideration the cursor spoofing attacks, through which Clickjacking attacks can succeed. Also, the system only checks for clickable elements; the attacker can develop a transparent FRAME over a text field.

To counter the cursor spoofing attacks Lin-Shung Huang [10], propose a new way called the InContext through which this can be mitigated. In InContext like the ClearClick the systems compares the bitmap image of the system. The only difference is that the InContext compares the bitmap of the Operating System lever screenshot with the screen shot after the click when invisible FRAMS might start working. If the two compared image differ then the user action is cancelled. This system ensures visual integrity and also reduces the number of

false positives. But this system enforces that at the host level the CSS transforms are not applied. Due to this the visual experience and the user flexibility is compromised.

Another system proposed by Krishna Chaitanya T. et. al. [3], enforces a simple policy change in the ClearClick. The ClearClick checks for clickjacking only when the IFRAMS originate from a different place. But if the hacker used the same space as the origin the ClearClick extension was not able to detect the attack. So, to counter this problem a simple policy change was proposed where even the same origin IFRAMS are also supposed to be scanned for any probable attacks. This simple policy change ensures that even if the attacker used the same origin space it is scanned for any malicious activities. This system improves the performance of ClearClick but still the number of false positives do not reduce that drastically. Also this will need extension specific approval.

i. Compromising courser integrity: To mitigate attacks which compromise the integrity of the courser almost all the updated web browser enforce that all the cross-origin customization of the coursers are disallowed [10]. Further enforces a rule where, if any sensitive elements are present than no customization is allowed altogether. Also this system disables any type of sound as it is believed that any type of sound distracts the user. This ensures the systems performance but the user experience is hampered.

ii. Compromising temporal integrity: To ensure that temporal integrity is maintained [10] enforce many new policies like delaying the user interface, delaying the user interface after the pointer entry, ensuring that the pointer has to be reentered into the sensitive space if any sensitive elements newly become visible, padding the area around the sensitive elements. The major impact of these policies is that the memory overhead is large apart from affecting the user experience.

# 4. DRIVE-BY DOWNLOAD

Generally the drive-by download involves the following steps [6, 17].

i. The attacker compromises a legitimate web application, add-ons, or plugins.

ii. The attacker inserts malicious codes into the compromised legitimate web applications, thus compromising the web server.

iii. Now when the user sends a request to the compromised web server the web server send the malicious code to the user.

iv. After download the malicious codes executes in the user's computer, thus compromising the user's computer. This might give the complete control of the computer to the attacker

Many techniques have been proposed to detect, mitigate and prevent drive by download attacks. At each stages of the drive-by download attack many techniques have been proposed. Detecting and mitigating drive-by downloads at has its own challenges. Table 1 lists the challenges faced at each stages.

**Table 1: Challenges**

| Stages | Challenges |
|---|---|
| i | Even a legitimate web pages can be compromised |
| ii | Cloaking, obfuscation |
| iii | Availability of vulnerability, missing zero-day attack, time consumption |
| iv | Once the malicious code has been executed the user may face exploitation including delay exploitation, there is a possibility that user might detect virtual environment, hooked function might be detected. |

Marco Cova et. al. [18], have proposed no less than ten features that need to be included into the web browser to detect the anomalies while download is taking place and monitor those download closely to detect and prevent malicious codes from getting downloaded onto the user systems. The features include the tracking of different browser components like the ActiveX controls and plugins, keeping track of the parameters passed by the functions, also keeping track of the number of times the browser has been redirected to a different URLs. All these features does help in reducing the number of drive-by download attacks but rather than using a full-fledged browser the system has been tested on an HtmlUnit emulator. The integration of all the said features into a web browser seems a task and also the memory requirements have not been mentioned considering the amount of monitoring has is required.

Pratik Upadhya et. al. [19], extend the concept of Browser Guard initially proposed by Fu-Hau Hsu et. al. [20], in 2009. In the proposed solution to detect and mitigate drive-by download the Browser Guard sets several check points on a browser and the windows kernel. This helps in detecting a download content and further blocking downloaded malicious codes at runtime, without needing user prompting. The blocking is done as the Browser Guard maintains a set of lists, White list and Black List. All the download initiated by the URLs present in the White list is allowed continue and all downloads initiated by the URLs present in the Black list is blocked, this is known as the filtration phase of the Browser Guard. The next phase is the prohibition phase, here the hash value of the executable file is calculated and all the Black listed hash value files are blocked from executing. The kernel component of the Browser Guard makes sure that the check points are not bypassed by the multiple downloads initiated. The system helps in detecting and mitigating drive-by downloads attacks but, the system fails in the case of shadow attacks.

Krishna Chaitanya T et. al. [3], observed that the web servers have no say with regards to whether the users can or cannot download any extensions or executable files. It was further observed that if the web servers were given the authority to say whether the user can download the extensions many drive-by downloads can be mitigated. As, the web servers can detect whether the download is initiated by itself or third malicious party has initiated the download. So, it suggests that a declarative security policy should be set where the web servers have an option to allow or deny the download of any extensions.

The declarative policy is implemented by the administrator or developer of the web page. The Administrator or the developer declares a new HTTP response header, 'X-Extension-Permit'. When the browsers detect this 'X-Extension-Permit' HTTP header, they execute the requested security policy, thus giving the web browser a say in the

download extensions. But this verification from the source site is also hinders the user experience and autonomy.

# 5. ANALYSIS

Table 2 provides a brief analysis of three clickjacking attack detection and mitigating techniques that have been discussed.

**Table 2: Analysis of Clickjacking Papers**

| **Parameters** | **Analysis and detection of modern spam techniques on social networking sites [3]** | **A solution for the automated detection of clickjacking attack [15]** | **Clickjacking: Attacks and Defenses [10]** |
|---|---|---|---|
| **Input** | WebPages | WebPages | WebPages |
| **Existing solution** | NoScript extension for Clickjacking | NoScript extension for Clickjacking | oScript extension for Clickjacking. |
| **Proposed Solution** | Sample Chrome extension for Clickjacking | Automated System | Automated System that compares operating system level screenshots |
| **Limitations** | Extension specific approval | Testing unit only interacts with the clickable elements. | Enforcement of not applying CSS transforms |
| **Output** | Checks transparent Iframes from cross origin as well as from same origin | Generate report which contains pages having clickjacking attempts and transparent Iframes. | Checks transparent Iframes by comparing the screenshots taken at the operating system level and after the invisible frames have initiated. Also prevents courser spoofing. |

Table 3 provides a brief analysis of three drive-by download attack detection and mitigation techniques that have been discussed.

**Table 3: Analysis of Drive-By Download Papers**

| **Parameters** | **Analysis and detection of modern spam techniques on social networking sites [3]** | **Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code [18]** | **Runtime Solution for Minimizing Drive-By-Download Attacks [19]** |
|---|---|---|---|
| **Input** | WebPages | WebPages | WebPages |
| **Existing solution** | Machine learning algorithms for malicious browser extensions | Machine learning algorithms for malicious browser extensions | Tools like Patch Exe, NOP sleds and Shell code, IMC. |
| **Proposed Solution** | Extension GateKeeper for malicious browser extension. | Ten features to monitor and mitigate drive by download attacks | BrowserGuard |
| **Limitations** | Verification from source sites | Tested on HtmlUnit emulator. Memory requirements are unknown | Not applicable in case of shadow attacks |
| **Output** | Execute the requested security policy against malicious browser extensions. | Mitigate threats which arise from drive by download attacks | Mitigate threats which arise from drive by download attacks |

# 6. CONCLUSION

With popularization of the social media many not to tech-savvy users have been added into the World Wide Web. Also, the attackers have been innovating and inventing newer techniques to fool the user into diverging their secrets which can be exploited. And, clickjacking and drive-by downloads attack have become popular tools through which the attackers exploit the vulnerabilities of the user.

Due to these reasons it is very important that extensive research is conducted and automated systems are developed in from time to time that can cope with eminent and ever emerging threats from clickjacking and drive-by download attacks. The future systems should be easily integrated into the web browsers, it should be easy to use and its working should be transparent to the user, it should be of capable handling shadow attacks and should not enforce unreasonable restrictions on the user. The future systems should ensure that the vulnerabilities that exists in the whole process of web interaction are not exploited by the attackers.

## 7. REFERENCES

[1] Clickjacking Attack Lab, Laboratory for Computer Security Education, http://www.cis.syr.edu/~wedu/seed/Labs/Vulnerability/ClickJacking/ClickJacking.pdf

[2] https://developer.mozilla.org/en/docs/Web/HTML/Element /iframe. Last accessed 16th, Dec, 2015

[3] Chaitanya, K. T., Ponnapalli, H., Herts D. and Pablo, J. 2012. "Analysis and detection of modern spam techniques on social networking sites", Third International Conference on Services in Emerging Markets, pp. 147-152

[4] Provos, N., Mavrommatis, P., Rajab, M. A., and Monrose, F. 2008. "All your iframes point to us", USENIX Security Symposium.

[5] Provos, N., McNamee, D., Mavrommatis, P., Wang, K., and Modadugu, N. 2007. "The Ghost In The Browser Analysis of Web-based Malware", First Workshop on Hot Topics in Understanding Botnets (HotBots '07).

[6] Egele, M., Wurzinger, P., Kruegel, C., and Kirda, E., "Defending Browsers against Drive-by Downloads: Mitigating Heap-spraying Code Injection Attacks", https://www.iseclab.org/papers/driveby.pdf

[7] https://en.wikipedia.org/wiki/URL_shortening. Last accessed 16th, Dec, 2015

[8] http://www.acunetix.com/websitesecurity/cross-site-scripting/. Last accessed 16th, Dec, 2015

[9] Gunatilaka, D. "A Survey of Privacy and Security Issues in Social Networks", http://www.cse.wustl.edu/~jain/cse571-11/ftp/social.pdf

[10] Huang, L., Moshchuk, A., Wang, H. J., Schechter, S. and Jackson, C. 2012. "Clickjacking: Attacks and Defenses", 21th USENIX (The Advanced Computing Systems Association) Security Symposium.

[11] Wisniewski, C. 2011 "Facebook adds speed bump to slow down likejackers". http://nakedsecurity.sophos.com/2011/03/30/facebook-adds-speed-bump-to-slow-down-likejackers/.

[12] Hill, B. "Adaptive user interface randomization as an anti-clickjacking strategy". http://www.thesecuritypractice.com/the_security_practice/papers/AdaptiveUserInterfaceRandomization.pdf

[13] Wang, H. J., Grier, C., Moshchuk, A., King, S. T., Choudhury, P. and Venter H. 2009. "The Multi-Principal OS Construction of the Gazelle Web Browser". In Proceedings of the 18th Conference on USENIX Security Symposium.

[14] Maone G. 2008. "Hello ClearClick, Goodbye Clickjacking!" http://hackademix.net/2008/10/08/hello-clearclick-goodbye-clickjacking/. Last accessed 16th, Dec, 2015

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

[16] Narayanan, A. S. 2012. "Clickjacking Vulnerability and Countermeasures", International Journal of Applied Information Systems (IJAIS) Foundation of Computer Science FCS, New York, USA Volume 4– No.7, December 2012, pp. 7-10.

[17] Le, V. L., Welch, I., Gao X. and Komisarczuk, P. 2013. "Anatomy of Drive-by Download Attack", Australian Computer Society, Inc. This paper appeared at the 11th Australasian Information Security Conference (AISC 2013), Adelaide, South Australia, January-February 2013, pp. 49-58.

[18] Cova, M., Kruegel, C. and Vigna, G. 2010. "Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code", International World Wide Web Conference Committee (IW3C2), Raleigh, North Carolina, USA WWW 2010, April 26–30.

[19] Upadhya, P., Meer, F., Dmello, A. and Dmello, N. 2013. "Runtime Solution for Minimizing Drive-By-Download Attacks", International Journal of Modern Engineering Research (IJMER) Vol.3, Issue.2, March-April. 2013 pp-1019-1021

[20] Hsu, F., Tso, C., Yeh, Y., Wang, W. and Chen, L. 2011. "Browser Guard: A Behavior-Based Solution to Drive-by-Download Attacks", Ieee Journal On Selected Areas In Communications, Vol. 29, No. 7, August 2011