UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

Hybrid Method on Clickjacking Detection and Prevention in Modern Advertisements

Kirit Shashank Dhurandhar, Maheyzah Md Siraj
School of Computing, Faculty of Engineering
Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Johor, Malaysia
Email: dhurandharkirit@yahoo.in, maheyzah@utm.my

Submitted: 9/07/2019. Revised edition: 29/09/2019. Accepted: 1/10/2019. Published online: 28/11/2019
DOI: https://doi.org/10.11113/ijic.v9n2.231

***Abstract*—In modern advertisements, clickjacking attacks can be delivered through a vulnerability in web application. To overcome this, web application security is required that will prevent malvertisement. In this study, prevention of clickjacking in the modern web advertisements are implemented. Vulnerability checks on the potentially malicious website were conducted. Implementation of hybrid prevention method of clickjacking into new developed website were carried out. Among top 500 websites, 50 websites were chosen as a dataset in this study out of which 4 case studies were selected. Website with server privileges were required to implement the hybrid prevention method, consisting opacity, Z-Index and X-Frame option policy. A new website was developed to satisfy the requirements for the method implementation. The results show, among 50 selected websites, about 19 websites were vulnerable to clickjacking. When the hybrid prevention method were implemented in the developed website, it increases the security by mitigating the vulnerability of web application to clickjacking attack.**

***Keywords*—Clickjacking, Detection, Prevention, Cyber Security, Opacity, Z-index, X-Frame option**

I. INTRODUCTION

When people talk about advertisements, it is thought of it as a way organizations and business owners use to sell more product and to popularize it by sharing them on the largest platform in the world which is the internet. Malvertiesing is a malware that use programmatic advertisement exchanges and deploy the malicious content. In other word, the advertisement is tricking programmatic exchanges into thinking that they are legitimate instead of a traditional publisher reviewing an advertisement and place it directly onto the web page. Later, they use this exchange to redirect the user without their knowledge.

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 [1]. Clickjacking in simple terms is hijacking user's clicks by using transparent or opaque layers by ticking them into clicking on a button or a link. It can also redirect the user to another page by not letting the user click on the uppermost page as wished. Clickjacking was called as "UI redress attack" is a where an attacker makes several invisible layers that confuses the client. When they are redirected to another page, the page is mostly managed by third party application, domain or both. The attacker achieves the goal by smartly setting a trap at a clickable region on a web page e.g. the region where the login button on the web page is located and the 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 make the targeted region transparent. In this region, different flavors of Clickjacking are used to trick the user like deploying fake cursor, transparent buttons, et cetera. The region might also be 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 the user click in the targeted region resulting in a malicious action the attacker is intended to do. 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.

There are many researches that have implemented their respective techniques in clickjacking prevention using different scripts and tools. The concept of Same-origin policy is discussed to create a better prevention compared to the limitation of previous research. Several clickjack mitigation techniques have been implemented and applied for browser, but they all have weaknesses. Internet being the biggest platform to show advertisements, attackers are getting successful in satisfying their malicious behaviors by finding vulnerability and exploiting it so gain private information from users which are mostly unaware of such happenings. In several scenarios, people might be unaware as to why the ads are popping up, what makes users redirect to a new website, or possible solution that must stop these advertisements from displaying. Recent researches had been deployed to address the issue of clickjacking exploit through prevention and detection techniques and in fact most of these anti-clickjacking techniques depend on numerous web application vulnerability which are fixable. However, clickjacking is still a threat to most users on the internet by the means for social networking or online movie streaming or unlicensed software providing websites. Therefore, these research focus on malicious websites to detect and prevent malicious clickjacking or redirects by making a hybrid prevention method.

II. Literature Survey

In paper [2] they have devised new clickjacking attack variants, which bypass existing defenses and cause more harm than previously known, such as compromising webcams, user data, and web surfing anonymity. To defend against clickjacking in a fundamental way, they have proposed InContext, a web browser or OS mechanism to ensure that a user's action on a sensitive UI element is in context, having visual integrity and temporal integrity. The concept of context integrity is introduced and is used to define and characterize clickjacking attacks and their root causes. They have designed, implemented, and evaluated InContext, a set of techniques to maintain context integrity and defeat clickjacking.

The authors in [3] have proposed attacks 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. For example, Zscaler Likejacking Prevention, detects hidden Facebook widgets and warns users about Likejacking. Where, it tries to confirm the password, pictures, and any private information that has more value for the users.

The proposed attacks are launched into two different scenarios such as Use of CAPTCHA and Use of Interest. The proposed attack is a type of human authentication scheme in which the users were asked to follow a certain pattern to allow the user to access the actual website. This paper [3] has proposed defense by creating 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 which adds a confirm dialog to every Facebook Like button in order to prevent Clickjacking. The proposed defense covers the functionality of both the existing extensions and ensures the pointer integrity. Hence the name given to it is Cursor Spoofing and Clickjacking Prevention (CSCP). CSCP has the functionality of detecting and preventing Clickjacking attacks on the Facebook. When the pointer clicks on like or follows button, a pop-up appears to the user that is clicked. 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.

Completely hidden: The actual clickjacking attack consists of loading a victim piece of content into a 1x1 iframe which affects the end-user by preventing them by not able to see the victim content. The attacker then aligns the 1x1 iframe at the center under the cursor so that the end-user clicks it. Thus, the end-user cannot make a difference between the 1x1 iframe beneath the mouse pointer, people can be tricked easily in clicking on such content.

Transparent overlay: In this scenario the attacker may work on making the trusted windows transparent. The attacker will then use this to overlay the trusted window over something that the user wishes to click. This will cause the end-user to trust that they are clicking on the content aligned beneath the legitimate window. This scenario would register the click by the transparent window since it is aligned over the content at the time the click was made.

Rapid content replacement: Like 'Content overlay' attack, this variation lets an attacker to try and obscure the content over the click where the user wishes to click. The attacker waits for the end-user to click, as soon as the user is believed to click, the attacker rapidly takes away the content that is obscuring the victim dialogue box. Formerly the end-user clicks on the victim dialogue box, the attacker puts the content overlay to obscure the dialogue. The process hardly takes more than few milliseconds. This gives the attacker the freedom to ask the user to perform a double-click. The click takes away the malicious overlay and the next click would be passed to the legitimate dialogue beneath. The whole scenario explains how the attacker uses this technique to bypass the screen scraping security by making sure the dialogue box is completely visible when the user wishes to click. The cycle of this technique only makes the dialogue visible till the time the click has been registered and again hidden back to gain.

Content overlays: The most common way to exploit clickjacking involves obscuring a legitimate and trusted dialogue by overlaying malicious scripts or contents. There are many variants to this attack.

Why Clickjacking Exists?

There could be a several reasons which depends upon the type of attack to the vulnerability of a web page. This particular

attack gradually evolves which makes it easy to prevent in a website that can be insecure to clickjacking assault. There are many detection and mitigation methods overall. They range from prevention method both client side to the detection method server side.

Most clickjacking works when the affected user is already logged in a particular webpage similar to socializing networks. The victim is then tricked by attacker into performing unwished process on a legitimate site. Social networking sites are engineered to scatter data, info or links quickly such as viral media and clickjacking uses this platform to spread the attack.

How Clickjacking Works?

The affected website loads something called as iFrame in its target website. The attacker makes sure the alignment of the target website is accurately positioned in the affected website. [4-6]. Mouse movement is also followed using Javascript. [5, 7]. Mouse movement is also followed using JavaScript as shown in Fig. 1, Fig. 2, Fig. 3, Fig. 4, Fig. 5 [7].

```
1  <iframe src = "[target website]" width = "1000" height = "500"
scrolling = "no" frameborder = "none">
2  < / iframe>
```

Fig. 1. Sample code frame

```
1  <iframe src = "[target website]" width = "1000" height = "500"
scrolling = "no" frameborder = "none">
2  < / iframe>
```

Fig. 2. Sample code clickjacking

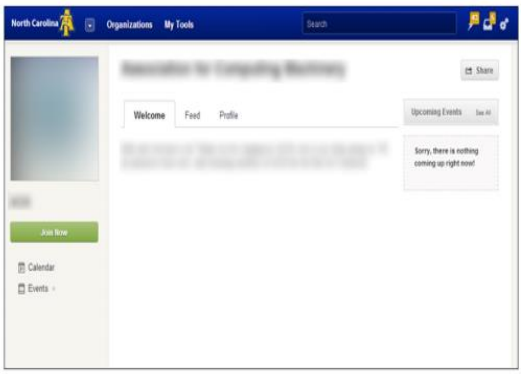

Fig. 3. Inner.html

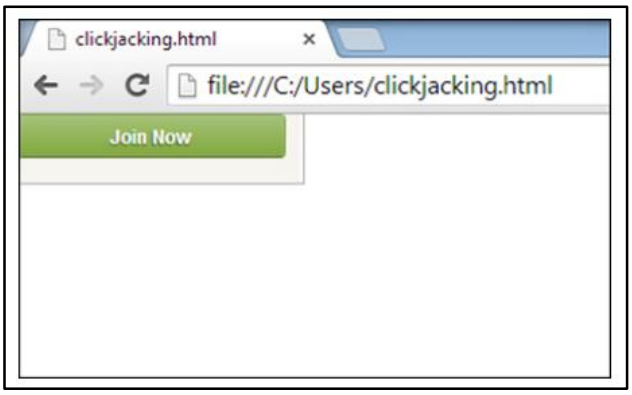

Fig. 4. Clickjacking.html

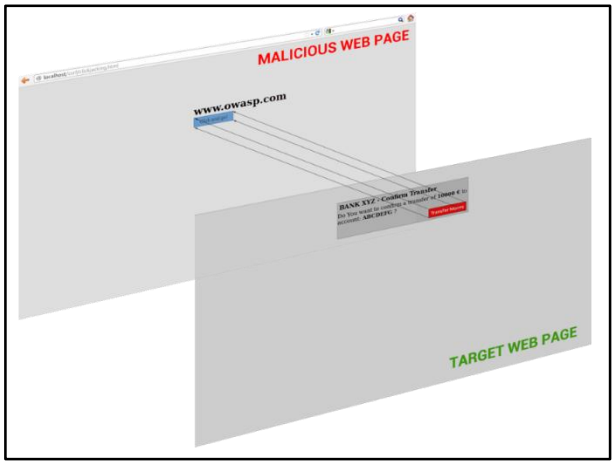

Fig. 5. Example of layers

Fig. 1 is an example of clickjacking using series of iframes with absolute positioning. It shows the source code for inner.html which puts the target page in an iFrame shown in Fig. 3. While, Fig. 2 shows the source code for clikjakcing.html where it puts inner.html into an iFrame that means the target page is now inside the two levels of iFrames. It will be resulted in the clickjacking.html showing the Join Now button instead of the entire page, shown in Fig. 4 The layers after the website that is put in an iFrame can be seen in Fig. 5.

Malvertisement

An advertisement which includes malicious content or used to download malicious software on a user's computer is known as malvertisement (malicious advertisement). It can be used to attack the user's computer with malicious software. Blue Coat systems Inc which is a well-known network security company says that malvertising is the latest way to hijack a computer. This technique is preferred choice for organized crime. Affected devices can be used to create stronger botnets that may be made to use as identity theft, corporate espionage etc.

III. METHODOLOGY

This tool informs clickjacking vulnerability from the website. X-Frame-Option is used for server-side implementation that can intercept and analyze requests coming through browser from response page (received from remote web servers).The advantages from such proxy level analysis. First, advanced types of clickjacking attackers mostly rely on sophisticated JavaScript code. If it can analyze the structure of JavaScript code for potential malicious activities (e.g., clobbering object, defining event handler), then attacks can be identified early. Second, the approach does not depend on the enabling or disabling of JavaScript code at the client side. Third, clickjacking attacks due to stripping special HTTP headers (X-Frame-Options) by other proxy servers can be addressed easily. Finally, advanced attack techniques can be detected without breaking legacy websites, and with less performance overhead. Fig. 6 shows the flow of the research in prevention of clickjacking.

When a response page is received, it should be checked for prevention which will performs several checks to identify the symptoms of a clickjacking attack in the page. Fig. 6 also shows the flowchart of detecting attacks based on three modules: Transparent Iframe, Z - Index, and HTTP header policy [8].

Manual detection for vulnerability was conducted on websites using online tools such as Appsec and Geek Flare. Appsec is a tool used to test if a website is farmable in an iframe of a different website. If the websites is frameable, it will loads in Appsec iframe demonstrating vulnerability to clickjacking. If the website fails to load in Appsec iframe, there is a possibility of clickjacking prevention implemented on that website. To further analyze if the website contains any clickjacking prevention implemented, Geek Flare tool is used. Geek Flare is a tool to read a websites header information that displays the presence of X-Frame option policy. If the website is equipped with an X-Frame option policy, then the website is not vulnerable to clickjacking. In contrast, if the X-Frame option is absent, the website is vulnerable to clickjacking attack. Thus, requires implementation of clickjacking prevention technique. Further testing can be carried out using testing methods mentioned below to find hidden iframes that are potentially malicious and could lead to a clickjacking attack.

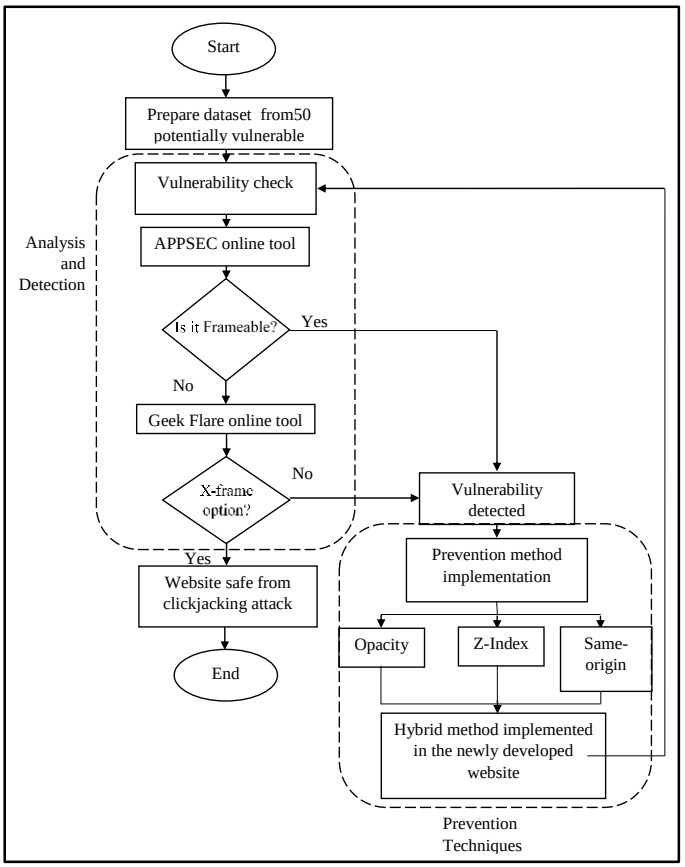

Fig. 6. Research framework

Prevention of clickjacking using a hybrid of opacity, Z-Index, X-Frame option (same origin policy) techniques and uses a web-based tool to detect vulnerability of potential webpage to get the required data for the prevention. Transparent Iframes are proven to be the most effective way to make a user click on a link or button or frame without their intention. By changing the background colour of the background and font colour, it will make any transparent Iframes visible (if there are any). This will further let us inspect the source code to check if there are any element which have its background color and font color transparent. If that is the case, then the technique mentioned can be used to make transparent Iframes visible and the user can see the malicious website loaded in that Iframe which can help prevent accidental or unintentional clicks by the user.

Z-index basically defines which layer of the webpage is closer to the human eye. First, user would shortlist all elements which have position attribute not set as static as z-index is not defined for such elements. Then it will filter out elements closest to users' eye, i.e. having max z-index. If these filtered elements are found to have transparent background colour and font colour, then it will make them visible.

From Fig. 7, it shows a website with X-Frame-Option set to same-origin which states that this website could not be loaded in an Iframe of another website if the origin of that website is

different, hence preventing a clickjacking attack occurrence. This method is used to compare the websites and test them for clickjacking vulnerability. Manual checking of the websites will be conducted as a proof of concept to demonstrate the effectiveness of Same-origin policy.

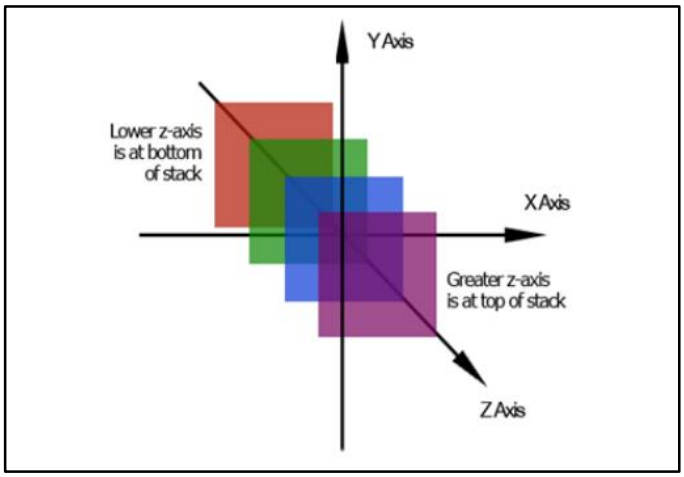

Fig. 7. Z-Index [9]

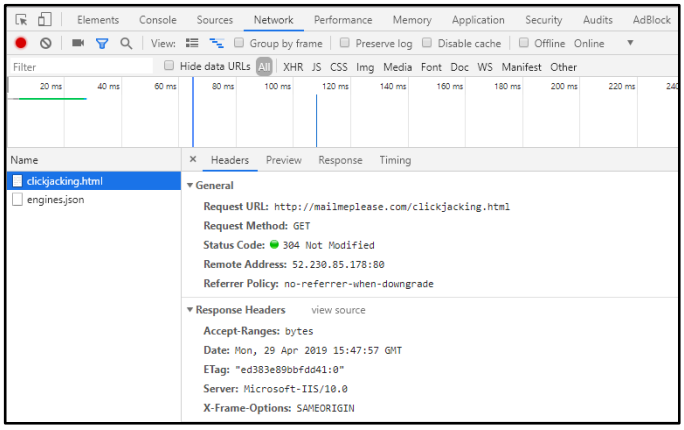

Fig. 8. Example policy X-Frame-Option (Same-origin)

From Fig. 8, it shows a header option that can be read using browser developer settings. For this example, a new website was created for the testing of the effectiveness of this method. As you may see in the figure above, the developer option can read the header configurations of a website and shows an X-Frame option set to same-origin. This is the option that is discussed and one of the methods used to prevent clickjacking vulnerability.

IV. Design And Implementation

In previous section, it was discussed that a set of websites to test for potential vulnerability for clickjacking from The Moz top 500 website. They provide websites that are among top 500 sites in the world as seen in Fig. 10, which will provide legitimate websites for further testing for clickjacking vulnerability.

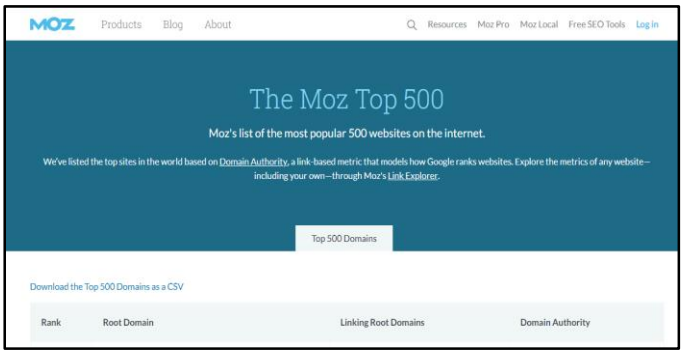

Fig. 10. Website interface

The Moz lists top 500 websites in the world based on domain activity and google ranking. They sort the list of websites by their rank, root domain, linking root domain and domain authority which shows the website legitimacy and provide domains that are active. These websites benefit this research by providing legitimate websites for testing possible vulnerability for clickjacking.

Vulnerability testing were primarily done using two online tools which play an important role. As it provides crucial data for demonstrating vulnerability of websites by reading information from websites. Which gives this research a backbone by providing a concept of this study. Tool for adding a website into an iframe is called Appsec. While, tool for reading HTTP header information on the websites is known as Geek Flare.

A. Appsec

Clickjacking which is also known as UI redressing which manipulates an iframe to load a legitimate website on a malicious or attackers' website as shown in Fig. 11. This tool loads a potential website to its iframe by proving the concept of this study. Appsec website provides a simple tool where a potential vulnerable website link can be pasted on its website to test the vulnerability to clickjacking.

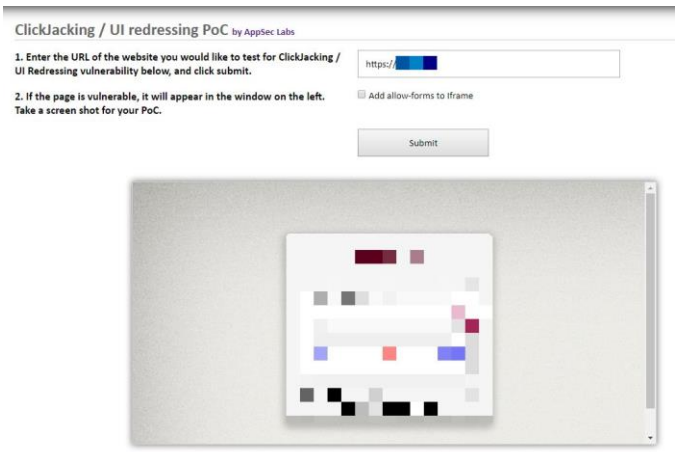

Fig. 11. Appsec interface. Image is blurred to protect the identity of the website

To test the vulnerability of a website to clickjacking attack could be proven when the websites is loaded in Appsec iframe. Which is primarily how clickjacking is carried out by an attacker. If the website is successfully loaded in Appsec iframe, it provides this study a clear proof that the given website is vulnerable to clickjacking. If the website does not load in Appsec iframe, this demonstrates that the website is not vulnerable to clickjacking and no further testing or implementation can be conducted on the study.

B. Geek Flare

Geek-Flare is a Netspark web application security scanner which is the only scanner that delivers automatic verification of vulnerabilities. Proof-Based scanning. Websites that are proven to have a vulnerability of clickjacking are further verified if there are any HTTP header option such as X-Frame option present in the website header as shown in Fig. 12. 100% of the times if X-frame option policy is not present in the website header, it is vulnerable to clickjacking.

xFrame Information

Name	Value
date	Wed, 01 May 2019 20:48:45 GMT
server	Apache
content-length	3699
connection	close
content-type	text/html; charset=UTF-8

Fig. 12. X-Frame information

Geek-Flare also gives a score to the websites for vulnerability to clickjacking as shown in Fig. 13. Where, 'A' is the highest score and 'C' being the lowest score. Websites that scores an 'A' in Geek-Flare is considered as not vulnerable to clickjacking attack or UI redressing attack. However, websites that are scored a 'C' are considered to be vulnerable to clickjacking attack. The tool scores on the webpage works by analysing if the website header option present the X-Frame. If the tool reads the webpage header and couldn't find the X-Frame option, it scores the page with 'C'. On the other hand, if the tools read the X-Frame option present in the webpage. It scores it with 'A' which also means the webpage is secured from a possible clickjacking attack. The X-Frame option is a policy that prevents a website from getting clickjacked by not allowing it to load in other iframe depending on the policy configuration of the website.

The results obtained from the vulnerability identification using this tool will require further investigation to implement the prevention method. This method can be used to prevent such attacks in modern advertisements by making such websites mitigate a possible clickjacking attack entirely.

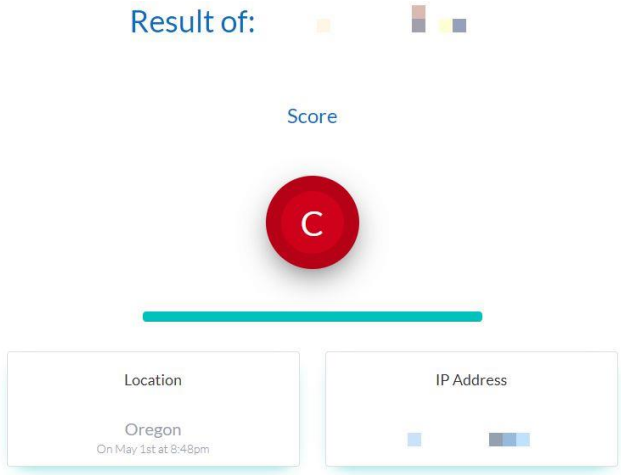

Fig. 13. Vulnerability testing. Image is blurred to protect the identity of the website

There are two existing techniques used in preventing clickjacking such as opaque transparent iframe and Z-Index check as discussed in previous chapters. However, this technique was not very effective for modern day malicious advertisements which rely on clickjacking vulnerability. They hijacked user's clicks and redirected them to a malicious website.

When the attacker manipulates transparent iframes to hide in the websites interface by reducing the opacity level, this prevention technique can be applied to make the transparent iframes more visible to the user's eye and preventing the user to perform clickjack which will redirect them to a malicious website. The opacity levels of the hidden iframe implemented by the attacker is shown in Fig. 14.

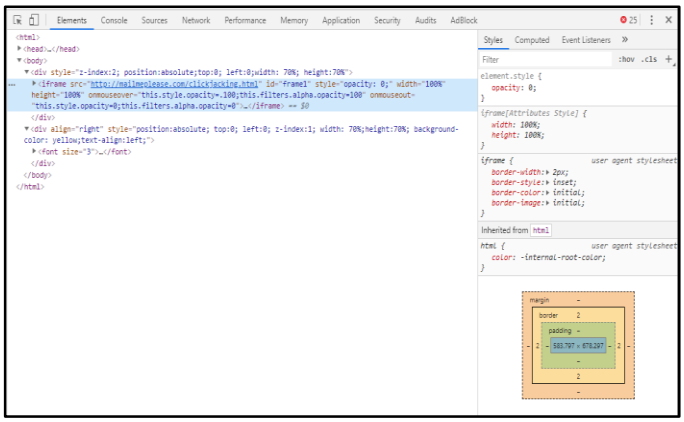

Fig. 14. Opacity level set by the attacker

In Fig. 14, it shows the attacker's line of code that can be overridden and making it visible to the user's eye. Further investigation can be carried out on the source code, by detecting the element background and the font colour.

Z-index basically defines which layer of the webpage is closer to the human eye. First, it will shortlist all the elements which have position attribute that is not set as static as z-index and not defined for such elements. Then it will filter out elements which closest to the user's eye, in which having maximum z-index. If these filtered elements are found to have transparent background color and font colour, an alert will be generated as shown in Fig. 15.

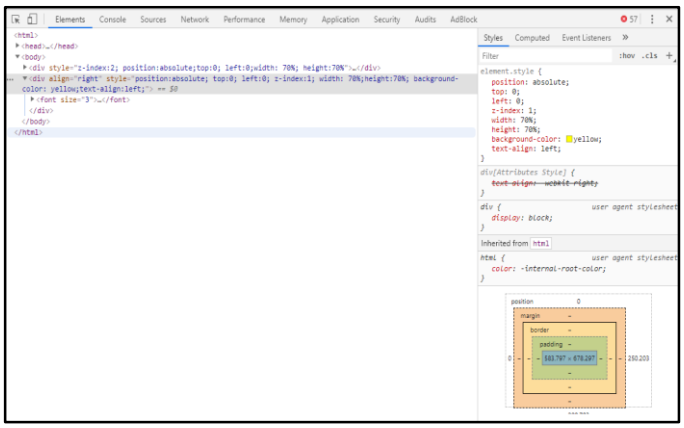

Fig. 15. Z-Index check implementation

This prevention implementation sets the Z-Index of a website to the uppermost layer and eliminates the iframe from appearing above a website which in return will alerts the user to avoid any unintentional clicks and can theoretically prevent a clickjacking attack. Z-index only takes effect if the position of the element are set explicitly. By setting it to be fixed, absolute, or relative.

X-Frame option is a HTTP response header that can be used to indicate whether the browser should be allowed to render a page in a frame or an iframe. When it implemented on any websites, it could avoid clickjacking attacks by making sure that their content is not embedded into another sites. By using the x-frame-options directive to protect sensitive anti-cross-site request forgery pages, web developers can immediately help mitigate the web application attacks. If the X-FRAME-OPTIONS value contains the token 'DENY' browser will prevent the page from rendering since it can be contained within an iframe. If the value contains the token 'SAMEORIGIN', the browser will block rendering only if the origin of the top level-browsing-context is different than the origin of the content containing the x-frame-options directive. For instance, if http://mailmeplease.com/clickjacking.html contains a DENY directive, that page will not render in a subframe, no matter where the parent frame is located. In contrast, if the x-frame-options directive contains the SAMEORIGIN token, the page may be framed by any page from the exact http://mailmeplease.com origin.

For this research, windows server 2016 was implemented for server-side operation. Internet information service (IIS) manager holds the windows server sites that have been deployed by the user. X-Frame-Option must be configured in the http response header tab as highlighted in Fig. 16. Once in the tab, the user could implement X-Frame option to prevent clickjacking attack.

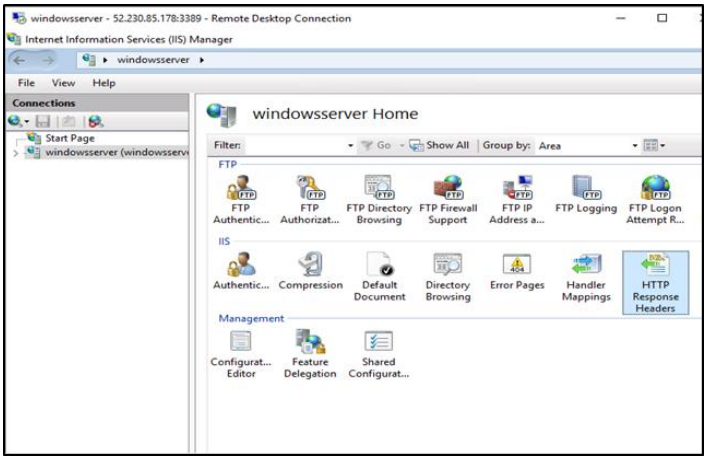

Fig. 16. Windows server Implementation

V. Results And Discussion

Several case studies are selected based on the popularity worldwide websites which potentially contains malicious advertisements. Each case study will be analyzed using the online tools such as Appsec and Geek Flare and discussed in previous chapter. Results from the vulnerability check will be presented in each section below.

Vulnerability Check

A. Case Study 1: Website A

This website was selected from dataset created using The Moz Top 500 website. It was ranked as number 1 out of the 500 websites listed in Moz.com. website A is known for the blog publishing service that allow any user to post time-stamped entries. It is globally used and developed since 1999, which was later bought by Google. The domain for website A can be owned by the user and direct the domain to Google servers. Since users can have their own domain, the chances of clickjacking vulnerability are high. By conducting the vulnerability check on the website using Appsec, the results are shown in Fig. 17.

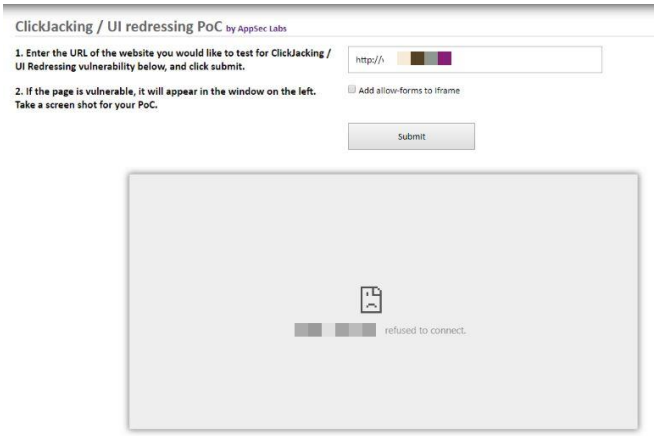

Fig. 17. Appsec results. Image is blurred to protect the identity of the website

Figure 17 shows the results obtained by pasting the website link on the Appsec interface. It shows website A refused to connect which means the website is not frameable on the Appsec iframe. The result might indicate the website A have implemented clickjacking prevention. Further investigation carried out in Geek Flare web tool as shown in Fig. 18.

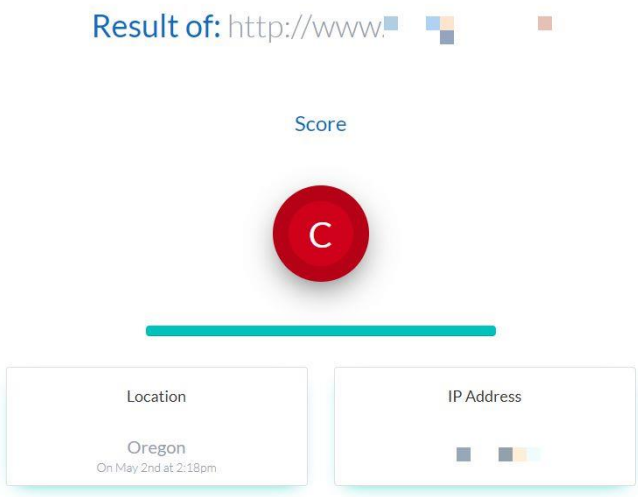

Fig. 18. Geek Flare scoring result. Image is blurred to protect the identity of the website

Geek Flare scores website A with a 'C', which indicates the websites is prone to clickjacking vulnerability. This tool gives the score by reading the HTTP header information of website A, it shows no presence of clickjacking prevention implemented in the website. Details of the analysis of the websites HTTP header information is shown in Fig. 19.

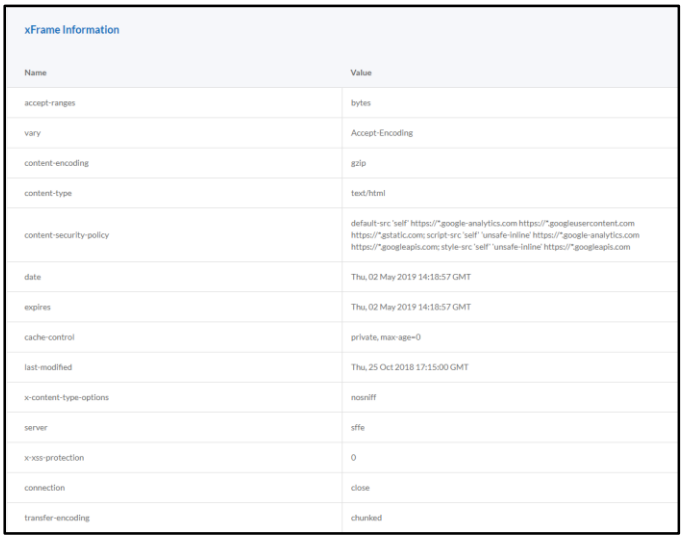

Fig. 19. HTTP header information

The HTTP header information shown in Fig. 19 indicates the absence of X-Frame option policy on website A. This suspects that website A relies on old frame busting technique which gives a false positive result which confuses the vulnerability testing process. However, this website is probably still vulnerable to clickjacking attack if an attacker exploits this vulnerability.

B. Case Study 2: Website B

This online movie streaming website is chosen due to its popularity in providing latest movies worldwide. The users frequently visit this site for free and latest movies since it has multiple streaming servers for uninterrupted entertainment. Due to its high traffic of visitors, making it a good opportunity for the attacker to carry out successful exploit of clickjacking vulnerability. By running the vulnerability check on website B using Appsec, the result is shown in Fig. 20

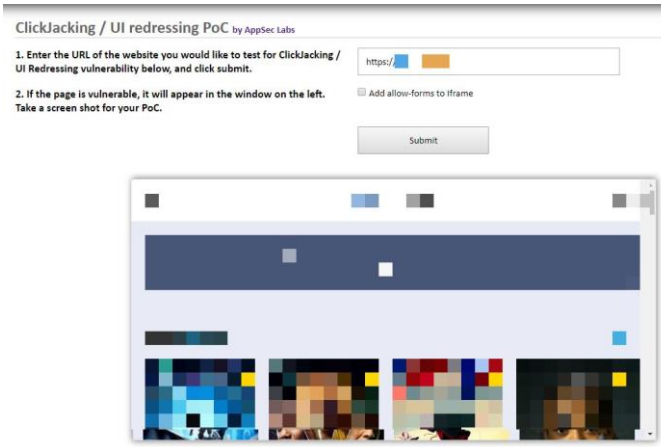

Fig. 20. Appsec result. Image is blurred to protect the identity of the website

In Fig. 20, the result indicates that website B is vulnerable to clickjacking. Appsec analyses the website by framing it on its iframe to detect possible vulnerability. Since website B loads in Appsec iframe, which indicates the website can be exploited by the attacker. Attacker uses these vulnerable websites to spread malicious advertisements to redirect users to malicious website. No Further testing needs to be conducted using Geek Flare, as the website shows clear indication of vulnerability to clickjacking.

C. Case Study 3: Website C

This higher educational institution website was chosen to spread awareness among students and security experts about the vulnerability to clickjacking. This website is a good example for demonstrating clickjacking vulnerability since the attacker can frame this website into another malicious iframe to make the malicious website look legitimate website. Users could be exposed to such malicious website to give away their credential information to the attacker without user ever knowing about it. Analysis of website C was carried out using Appsec web tool and the results can see seen in Fig. 21.

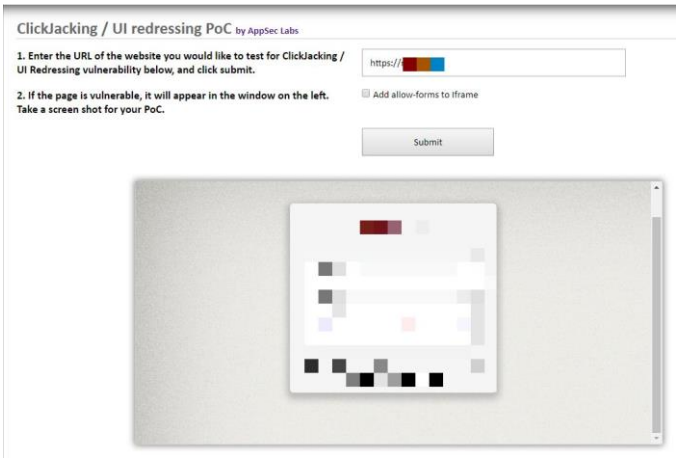

Fig. 21 Appsec result. Image is blurred to protect the identity of the website

Figure 21 shows vulnerability check results indicating website C indeed vulnerable to clickjacking attack. Geek Flare analysis will not be required since Appsec web tool enough to analyse the clickjacking vulnerability. HTTP header stores the X-Frame option information if it was implemented to website C.

D. Case Study 4: mailmeplease.com/clickjacking.html

New website was developed to support the implementation of the prevention method using windows sever and a domain to make the website live on the world wide web. The website was created using simple html coding and was further prepared to hosting using Microsoft Internet Information Services (IIS).

The website was hosted using mailmeplease.com domain. Windows IIS was used so that the manager function can be used to implement the prevention methods. The mailmeplease.com/clickjacking.html will be analyzed using the same tools and will undergo the same procedure for vulnerability check as above. The results from Appsec are shown in Fig. 22.

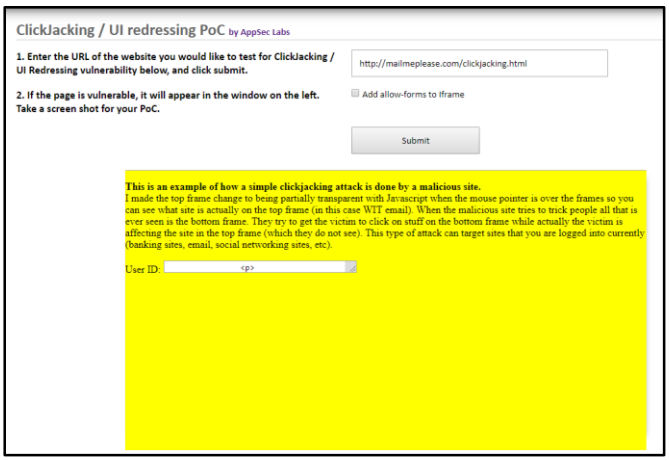

Fig. 22. Appsec result

In Fig. 22, mailmeplease.com/clickjacking.html is loaded in Appsec webtools interface which indicates the possibility of this website to be vulnerable to clickjacking. Appsec interface loads a given website in an iframe only when no prevention methods are implemented for clickjacking vulnerability. Since this is a testing website, further testing is conducted on Geek Flare to demonstrate and later compare the before and after implementation of X-Frame- option policy. Fig. 22 shows the analyzed results by Geek Flare interface.

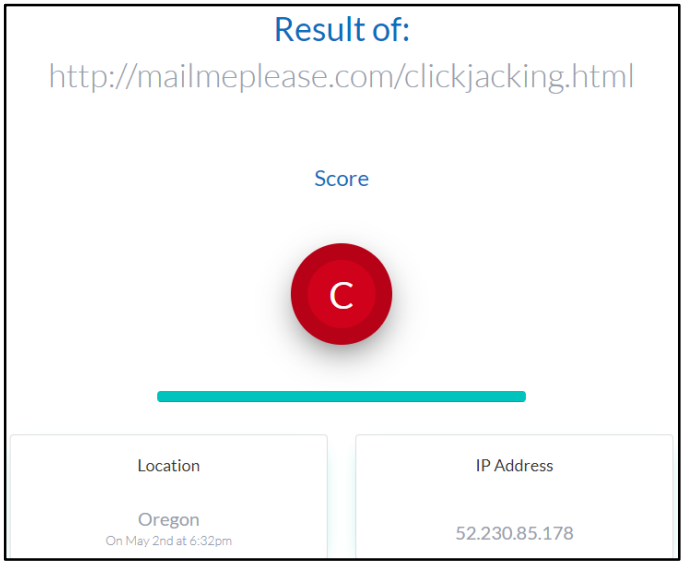

Fig. 23 Geeek Flare scoring result

Figure 23 shows score result of mailmeplease.com /clickjacking.html before the prevention methods were implemented the score shows 'C' which implies that the website is vulnerable to clickjacking attack. The HTTP header information is shown in Fig. 24.

.

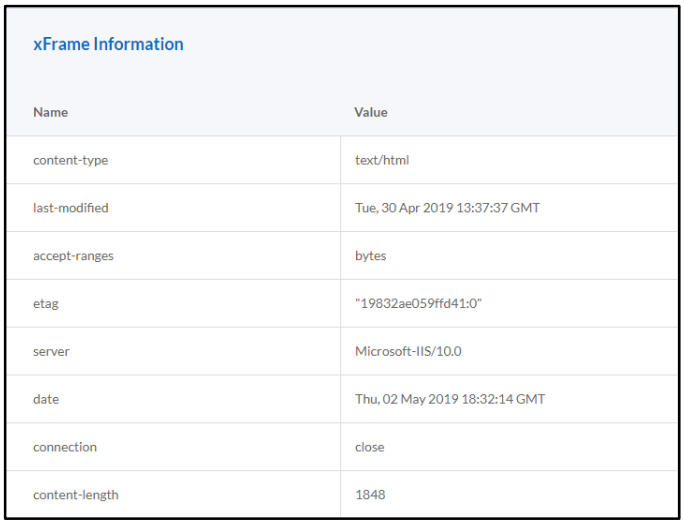

Fig. 24. HTTP header information

HTTP header information as seen in Fig. 24, shows no signs of X-Frame option policy present in the websites configuration. This analysis was carried out to give a clear example of the websites X-Frame option policy and will benefit further comparison of before and after implementation of prevention methods.

Hybrid Implementation

A. Opacity and Z-Index Implementation Result

To overcome transparency of the hidden iframe implemented by the attacker on the website, a line of code is applied on the newly developed website by increasing the opacity level from 0.0 to 0.1 Fig. 25. Z-Index also plays an important role in hiding malicious iframe that are commonly altered by the attacker. The website also demonstrates Z-Index value to mimic a vulnerable website altered by an attacker. Another set of code is implemented in the same developed website to change the value of Z-index in such a way that the website is displayed closest to the user's eyes shown in Fig. 26.

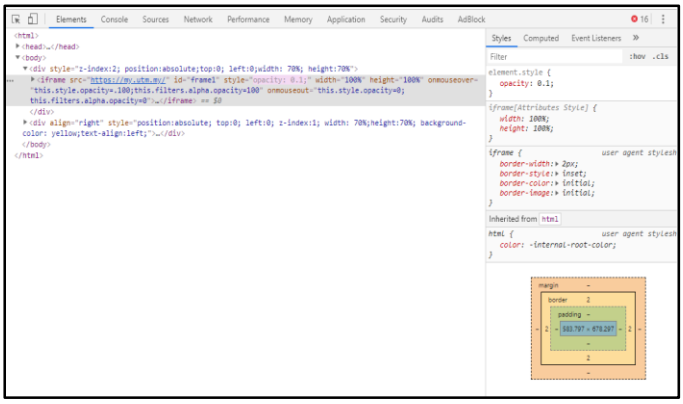

Fig. 25. Opacity change result

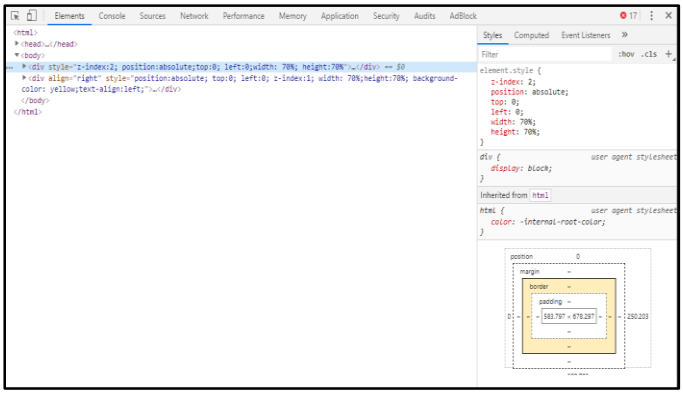

Fig. 26. Z-Index change result

Fig. 25 shows the opacity level that have been changed on the developed website. In the previous chapter a hidden iframe was mimicked as an example to show how an attacker makes

the iframe transparent. In Fig. 27, the website response on the prevention methods is demonstrated.

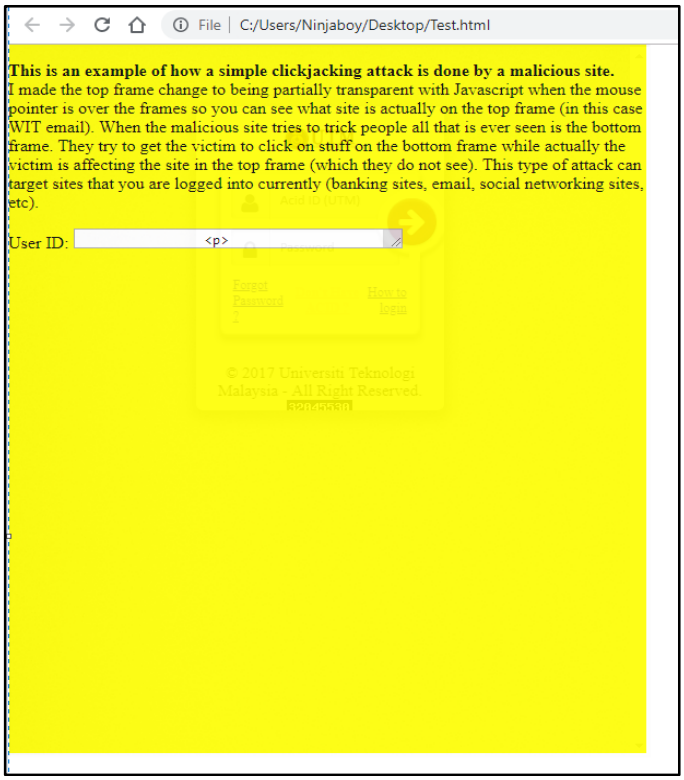

Fig. 27. Implementation result

Fig. 27 shows the result after the opacity and Z-Index implementation using the line of code shown in Fig. 25 and 26. The code is implemented in such a way that it works after the mouse is hovered over the hidden iframe. When the mouse is hovered the opacity of the hidden iframe is set to change and make it visible during mouse hovering. While , the Z-Index layers the legitimate website closest to the user eyes. This prevention method could be used for protection user from filling sensitive information such as bank details and website credentials, online shopping and others.

B X-Frame Option Implementation Result

HTTP header stores the configuration of X-Frame option implemented on a website. This information can be read using the browser developer settings or in this case Geek Flare web application security tool. When X-Frame option is implemented on a website, it blocks the website from rendering on the attacker's website preventing clickjacking attack. Server-side implementation was taken place to demonstrate the working of X-Frame option on a website. Fig. 28 shows the resulting change after implementation of X-frame option

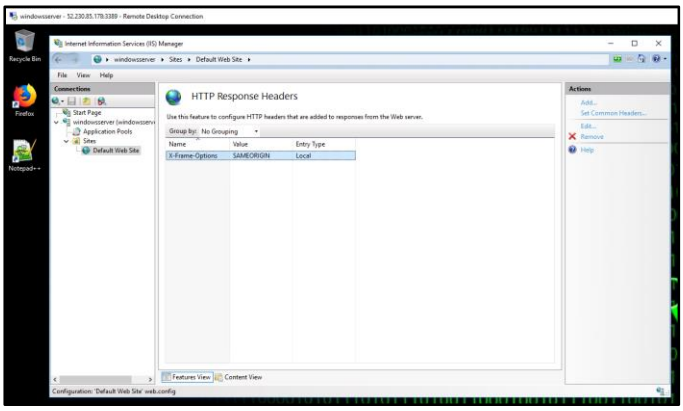

Fig. 28. X-Frame option implementation result

Implementation of Same Origin in X-Frame option was carried out in the windows IIS manager. The website which is supposed to be applied this prevention method is selected and configured according to the user's preference. In this demonstration the use of Same-Origin policy was conducted since it lets a website load into iframe if the website has a same origin. This website would not be rendered in any other websites iframe , thus preventing a clickjacking attack. Since it is a server-side configuration a website had to be developed to support the prevention method in this study.

Once the X-Frame option was applied to the website's configuration, the testing for clickjacking vulnerability was carried out to analyze if the implemented prevention methods works or not. Fig. 29 shows the results from Appsec web tools implying that the developed website failed to render on another website, thus preventing clickjacking attack. In other words, an attacker will not be able to exploit this website to hijack user click for its malicious use anymore.

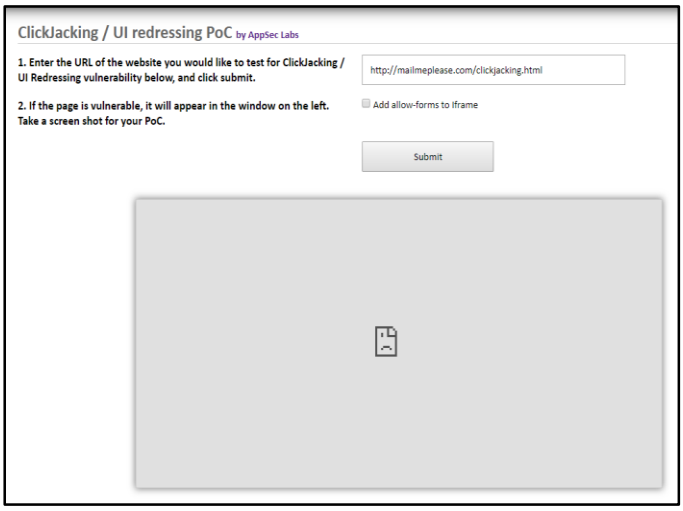

Fig. 29. Appsec result

Fig. 29 was achieved after configuring the web configuration of the website in IIS server. X-Frame option was applied and set to same-origin to demonstrate the effectiveness of proposed prevention methods. Further investigation was conducted to show if X-Frame option does indeed readable in HTTP header information, hence Geek Flare tool was used to check the score and read header information form the website. Fig. 30 shows the results after implementing X-Frame option from Geek Flare analysis of the developed website.

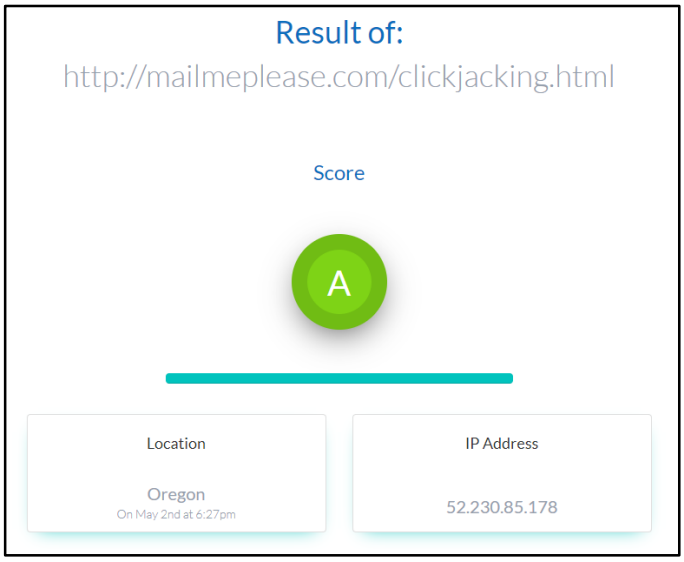

Fig. 30. Geek Flare results

This study also demands to read the HTTP header information of the website developed to prevent clickjacking; hence Fig. 30 satisfies those needs by using Geek Flare web tool to show X-Frame option present in HTTP header information.

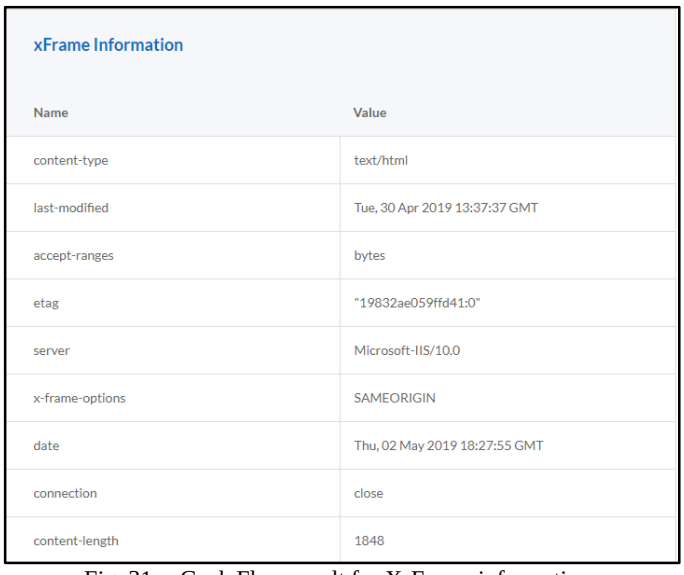

Fig. 31. Geek Flare result for X-Frame information

In Fig. 31, the X-Frame option can be read to be set as same-origin which concludes the prevention method and not further study needs to be conducted on this website since it fulfils all the required implementation and results to prevent a potential clickjacking attack to this website.

Discussion

There are many ways to prevent clickjacking on the internet, which varies depending on the studies. The prevention method provided by other studies may or may not work depends on the attacks. The methods for preventing clickjacking mentioned in this paper more relied on the X-Frame option more than the opacity and Z-Index. As the attacker may use the easiest way to attack users as many as possible, free movie streaming websites are a major threat for clickjacking since their revenue are depends on the advertisements shown in an iframe on the movie streaming websites. Iframe will displays advertisement and the attacker will take advantages and use it to spread malicious attacks to the users by exploiting the websites iframes and framing the malicious websites.

There are certain tools provided in Google chrome browser extension, which is an automated application to prevent the clickjacking vulnerability on websites through browser extension. After downloading and running this app in the browser extension, no security was prided by this apps for clickjacking vulnerability. This made this study to focus on manual implementation for this vulnerability, since the automated application are unstable and tend to stop working if when a proper design testing and implementation were not carried out by the developer. The X-Frame option automated application was under testing phase and was developed completely few months back. Since it is a new application, they only support some features provided by the X-Frame option.

VI. Conclusion

With an increase in the usage of the internet, protection against Clickjacking will become a necessity in coming days to protect users from malicious attackers. Many solutions came and became obsolete with time. However, while designing the prevention methods for this study extra caution were taken to make it more robust and providing a solution which will be easy to apply manually. This system will check for any anomalies pertaining to Clickjacking attacks present in web pages. Since it is a server-side implementation, developers may have to apply it manually to the websites they want to protect from clickjacking attack.

Several prevention methods for clickjacking have been implemented in the study, such as Opacity, Z-Index, X-Frame option. Each of these methods have their own limitations, which can be overcome by combining these 3 methods implemented to a website. Several websites were tested for vulnerability check and few of those were selected as case study in previous chapter. The results of vulnerability check from Appsec and Geek Flare are shown and discussed. New

website was developed to implement the hybrid prevention methods proposed by this study.

The prevention methods are server-side implementation which will contribute to the information security developers and less toward the clients or users. The results obtained after implementing the prevention methods to developed website concludes that by combining the 3 prevention methods, the prevention for clickjacking is successfully implemented.

REFERENCES

[1] Hansen, R. and J. Grossman. (2008). *Clickjacking*. Available from: http://www.sectheory.com/clickjacking.htm.

[2] Huang, L.-S., *et al.* (2012). Clickjacking: Attacks and Defenses. *USENIX Security Symposium.*

[3] Rehman, U. U., *et al.* (2013). On Detection and Prevention of Clickjacking Attack for Osns. *Frontiers of Information Technology (FIT), 2013 11th International Conference on*. IEEE.

[4] Callegati, F. and M. Ramilli. (2009). Frightened by Links. *IEEE Security & Privacy,* **7**(6).

[5] Niemietz, M. (2011). Ui Redressing: Attacks and Countermeasures Revisited. *CONFidence, 2011*.

[6] Stone, P. (2010). *Next Generation Clickjacking.* BlackHat Europe.

[7] Gall, M. (2010). *Facebook/Flattr/… Clickjacking Examples And How To Avoid It*. Available from: https://digitalbreed.com/2010/07/18/facebook-flattr-clickjacking-and-how-to-avoid-it/.

[8] Ross, D. and T. Gondrom. (2012). *HTTP Header X-Frame-Options.* Work in Progress.

[9] Bradley, S. (2009). *Z-Index and the CSS Stack: Which Element Displays First?* Available from: http://vanseodesign.com/css/css-stack-z-index/.