Case Study on Clickjacking

Shaeq Khan
SWE 681 Secure Software Engineering
Topic Paper - 2nd April, 2014
George Mason University
skhan27@gmu.edu

Abstract

Clickjacking attacks are an emerging threat to the web as it hijacks the users web session. This paper outlines major attacks that happened with user based web applications and discusses solutions to protect users from this attack. It also outlines the challenges faced by current solutions to re-iterate the need of better browser security and research directed in this field of study.

Introduction

Clickjacking is a malicious technique of tricking a web user into clicking on something different from what the user thinks they are clicking on. In simple terms it can be described as privilege escalation for attacker or unintentional misuse of authority by user.

The concept was compiled and introduced to the world first by Robert Hansen and Jeremy Grossman in Sept 2008 in a paper "Clickjacking". This is a browser security issue and not a bug in the web application. It results from the misuse of some html and css features combined with no form of effort made by the browser to restrict users from interacting with invisible or barely visible elements on a web page.

Clickjacking involves a Target site (T) accessible by the victim and of importance to the attacker like a banking or email website and a Malicious site (M) which is under the attacker control. A parent frame containing T is overlaid on M and presented to the victim. The victim thinks that they are making clicks on M but since T is aligned on M and is invisible, the victim unsuspectingly clicks on elements of T.

We can use three essential tags to redress a UI. The html tag iframe lets us embed a website in a frame. The css tag opacity lets us change the transparency of html elements. The css tag z-index lets us stack html elements on top of one another. Keep in mind that an element with a greater stack order is always in front of an element with a lower stack order.

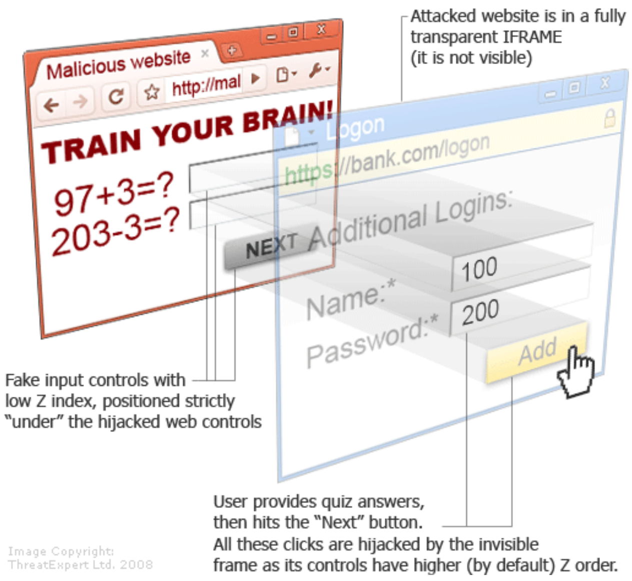

Malicious website
http://mal
TRAIN YOUR BRAIN!
97+3=?
203-3=?
NEXT
Logon
https://bank.com/logon
Additional Logins:
Name:*
Password:*
100
200
Add
Attacked website is in a fully transparent IFRAME (it is not visible)
Fake input controls with low Z index, positioned strictly "under" the hijacked web controls
User provides quiz answers, then hits the "Next" button. All these clicks are hijacked by the invisible frame as its controls have higher (by default) Z order.
Image Copyright: ThreatExpert Ltd. 2008

Figure 1 – Stealing user inputs

Clickjacking is considered to be Instance of the Confused Deputy Problem. A confused deputy is a computer program that is fooled by another party into misusing its authority.

Another example of the Confused Deputy Problem is Cross Site Request Forgery (CSRF) which is a common form of attack if the web application uses cookies to authenticate requests by a browser. However Clickjacking doesn't require access to cookies and instead misguides the user into giving access to the application over the web by using their clicks.

Clickjacking is an attack where the user acts as the confused deputy. In this attack a user thinks they are harmlessly browsing a website (an attacker-controlled website) but they in fact tricked into performing sensitive actions on another website. To perform the attack, a malicious website will load a page from the website inside an iframe made completely transparent and layered on top of another element of the malicious site. Clickjacking is also referred to as User Interface Redress Attack, UI Redress Attack, UI redressing. It can take the form of an embedded code or a script that can execute without the user's knowledge.

REAL WORLD ATTACKS

Click Fraud - Click fraud is a type of fraud that occurs on the Internet in pay per click online advertising when a person, automated script or computer program imitates a legitimate user of a web browser clicking on an ad, for the purpose of generating a charge per click without having actual interest in the target of the ad's link. The click fraud for facebook was targeted to generate likes for fan pages by admins or share a link to generate revenue from user clicks. The malicious sites tricked users into clicking on a transparent iframe that contained a concealed banner.

Figure 2 – Link on the victim's newsfeed

In the case of the malicious website http://fb.59.to, clicking on the link redirected the user to an external website -

```
http://www.facebook.com/l.php?u=http%253A%2
52F%252Ffb.59.to%252F%253F4ff11a526ae73e9f1
70bbe6702ebb93c&h=
..somehash...&ref=nf
```

This site gives user's a fake test to trick them into clicking on buttons. The facebook share button was hidden as an iframe over the several buttons that appeared on the page and that made the user share the malicious link with their friends list.

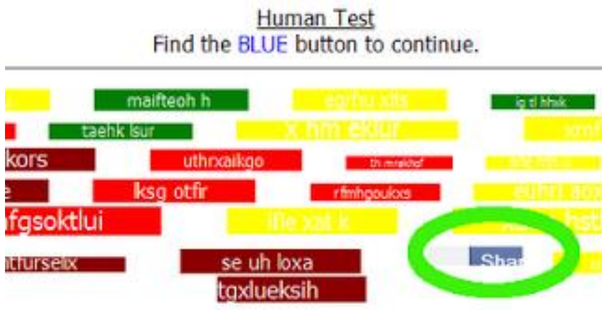


Figure 3 – Victim is redirected to the malicious site with the user's session details

In the page source, there is an iframe element -

```
<iframe frameborder=0 scrolling=no
height=25 width=100
src="2.php?u=http://fb.59.to/?...somehash..
.."></iframe>
<span style=background-color:yellow;>
<font style=font-size:13;color=white>
```

The target URL (2.php) has another iframe which in turn has yet another one with the target page

```
<div style="left:-90px; top:-386px;
position:absolute;">
<iframe height=400 width=250
src="http://www.facebook.com/sharer.php?u=h
ttp://fb.59.to/?hash" frameborder=0
scrolling=no> </iframe></div>
```

The Twitter Bomb – Don't Click!

Twitter saw a tiny url link that spread around by Clickjacking. Clicking on the link redirected the user to a website with a single button titled "Don't Click" and as we are quite aware of the human nature, people did click it.

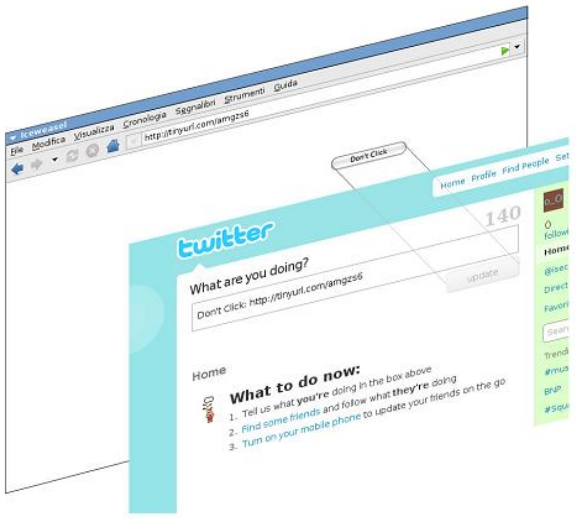


Figure 4 – Twitter Clickjack attack

The position of the twitter iframe and the button on the malicious site was in such a way that Twitter's update button (in the iframe) was positioned on top of the ordinary button on the page, but the iframe was hidden from view, so people thought they were clicking the ordinary button underneath it, but they were really clicking Twitter's update button.

The code used on the malicious site used iframe in the following way –

```
<iframe style={width:550px; height:228px;
top:-170px; left:-400px; position:absolute;
z-index:2; opacity:0;
filter:alpha(opacity=0);}
scrolling="no"
src="http://twitter.com/home?status=Don’t
Click: http://tinyurl.com/amgzs6"></iframe>
```

And the button on the malicious site was styled in the following way –

```
<BUTTON style={width:120px; top:10px;
left:10px; position:absolute;
z-index:1;}>
Don’t Click</BUTTON>
```

Note how the z-index of the iframe is more than that of the button which put it above the button and the opacity of the iframe was set to 0 which meant it was transparent.

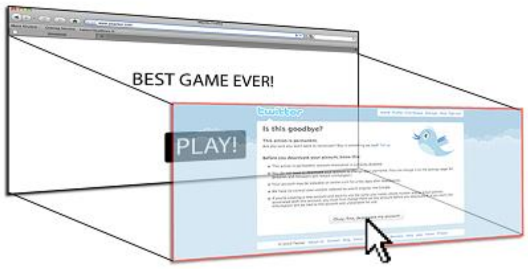
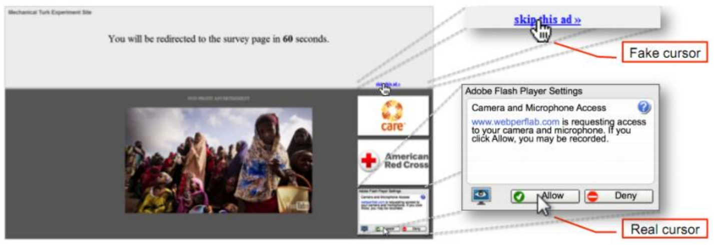


Figure 5 – Another twitter attack to make users deactivate their accounts to free strategic or valuable usernames

The Adobe Flash attack

Malicious sites used a fake cursor that was rendered to provide false feedback of pointer location. A loud video ad played automatically that made the user click on the “skip this ad” link. The real click landed on the Adobe Flash Player webcam settings dialog that granted the site permission to access the user’s webcam. Cursor hiding is achieved by setting the CSS cursor property to none, or a custom cursor icon that is completely transparent, depending on browser support.


Mechanical Turk Experiment Site
skip this ad »
Fake cursor
care
American Red Cross
Adobe Flash Player Settings
Camera and Microphone Access
Allow
Deny
Real cursor

You will be redirected to the survey page in 60 seconds.

www.webperflab.com is requesting access to your camera and microphone. If you click Allow, you may be recorded.

Figure 6 – The Adobe Flash attack

Countering Clickjacking using Frame Busting

This is a client side protection mechanism and is a piece of JavaScript code that prevents a web page from being displayed in a frame. A page using this method will detect that it has been framed by another website and attempt to load itself in place of the site trying to frame it (thus busting out of the frame).

The JavaScript code has a

- condition and
- counter action (to navigate the top of the page to the correct place)

A simple frame busting code used by websites -

```
if (top.location != location)
        top.location = self.location;
```

But this piece of code does not do the job because it can be overridden or defeated easily. Most of the Alexa Top-500 sites use frame busting code in JavaScript. But many do not frame bust on their front page. Instead, they only frame bust on a login page or on a password reset page.

The following are conditions and counter actions used in the Alexa Top-500 sites –

unique sites	conditional statement
38%	`if (top != self)`
22.5%	`if (top.location != self.location)`
13.5%	`if (top.location != location)`
8%	`if (parent.frames.length > 0)`
5.5%	`if (window != top)`
5.5%	`if (window.top !== window.self)`
2%	`if (window.self != window.top)`
2%	`if (parent && parent != window)`
2%	`if (parent && parent.frames && parent.frames.length>0)`
2%	`if((self.parent&&!(self.parent===self))&&(self.parent.frames.length!=0))`

Figure 7 – Popular frame busting conditional statements

unique sites	counter-action
7	top.location = self.location
4	top.location.href = document.location.href
3	top.location.href = self.location.href
3	top.location.replace(self.location)
2	top.location.href = window.location.href
2	top.location.replace(document.location)
2	top.location.href = window.location.href
2	top.location.href = "URL"
2	document.write('')
2	top.location = location
2	top.location.replace(document.location)
2	top.location.replace('URL')
1	top.location.href = document.location
1	top.location.replace(window.location.href)
1	top.location.href = location.href
1	self.parent.location = document.location
1	parent.location.href = self.document.location
1	top.location.href = self.location
1	top.location = window.location
1	top.location.replace(window.location.pathname)
1	window.top.location = window.self.location
1	setTimeout(function(){document.body.innerHTML='';},1);
1	window.self.onload = function(evt){document.body.innerHTML='';}
1	var url = window.location.href; top.location.replace(url)

Figure 8 – Popular frame busting counter action

COUNTERING CLICKJACKING USING X-FRAME-OPTIONS

This is a server side protection mechanism where the HTTP response header can be used to indicate whether or not a browser is allowed to render a page in a <frame> or <iframe> tag. This was first proposed by Proposed by Microsoft and adopted by IE8, Chrome, Opera, Safari. Sites use this to avoid Clickjacking by ensuring their content is not embedded into other sites. The HTTP X-FRAME-OPTIONS header code is used by only THREE of the Alexa Top-500.

Possible arguments

- **DENY**: webpage cannot be loaded by a frame
- **SAMEORIGIN**: display web page in a frame if the parent is same
- **ALLOW-FROM uri**: page can only be displayed in a frame on the specified origin

Clickjacking protection can be used with Java EE web applications. One way to do it is to go to every servlet or jsp and add the following piece of code –

```
response.addHeader("X-FRAME-OPTIONS",
"DENY");
```

The second argument can vary as per requirement. You can also implement a simple JavaEE filter to add the X-FRAME-OPTIONS header to some or all parts of your application by downloading the source code for ClickjackFilter from the OWASP site and adding it to the class path of your web project and then adding the filter definition and mapping to the deployment descriptor web.xml. Detailed instructions are provided on the provided link.

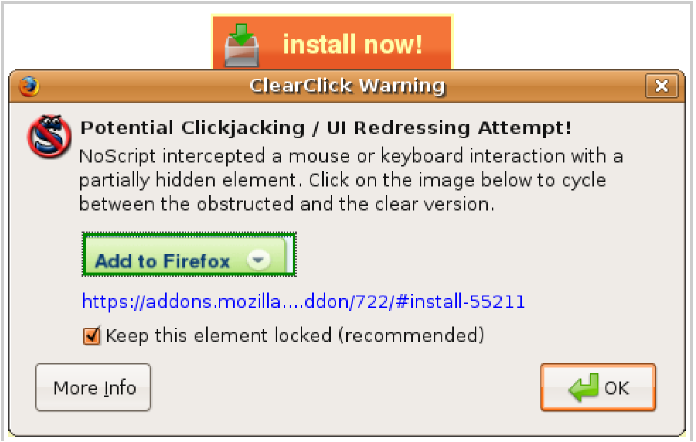
Possible problems with X-FRAME-OPTIONS

- policy needs to be specified for every page
- white listing of domains not supported in current implementation
- proxies add and strip off headers

COUNTERING CLICKJACKING USING BROWSER PLUGINS

This is a client side protection mechanism where the user can install plugins available for free of cost to the browser. Two such popular ones exist.

NoScript is an extension for mozilla based web browsers like Firefox. In October 2008, an anti-clickjacking feature was integrated. It allows JavaScript, Java, Flash and other plugins to be executed only by trusted web sites of your choice. It employs a whitelist based pre-emptive script blocking approach.



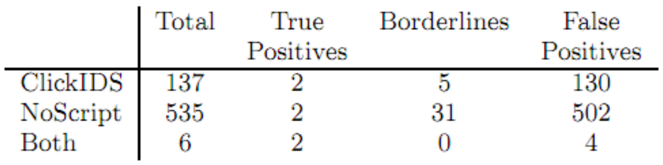
Potential Clickjacking / UI Redressing Attempt!
NoScript intercepted a mouse or keyboard interaction with a partially hidden element. Click on the image below to cycle between the obstructed and the clear version.

https://addons.mozilla....ddon/722/#install-55211

☑ Keep this element locked (recommended)

Figure 11 – NoScript interception as shown on official website

ClickIDS utilizes two browser plugins to operate in parallel in order to analyze the automated clicks. This was a result of a published paper that concentrated on finding defenses against Clickjacking attacks. This works with NoScript and experimental results show that the combination of the two different detection techniques greatly reduces the number of false positives.



	Total	True Positives	Borderlines	False Positives
ClickIDS	137	2	5	130
NoScript	535	2	31	502
Both	6	2	0	4

Figure 12 – Results for using both plugins to detect Clickjacking attacks

The basic idea behind the working of this plugin is that a suspicious behavior is reported when two or more clickable elements of different pages overlap at the coordinates of the mouse click.

Challenges for Current Solutions

In **Mobile Applications**, m.example.com or mobile.example.com deliver full or significant subsets of functionality, do not frame bust their mobile sites. This coupled with the many sites that do not differentiate sessions between the regular and the mobile site; that is, if you are logged in at www.example.com you are also logged in at mobile.example.com is dangerous because it ends up giving the attacker complete access to the full website from a simple mobile site.

Modern browsers treat the ***location* variable** as a special immutable attribute across all contexts. However, this is not the case in IE7 and Safari 4.0.4 where the location variable can be redefined as shown below –

Frame busting code on victim.com

```
if (top.location != self.location) {
      top.location = self.location ;
}
```

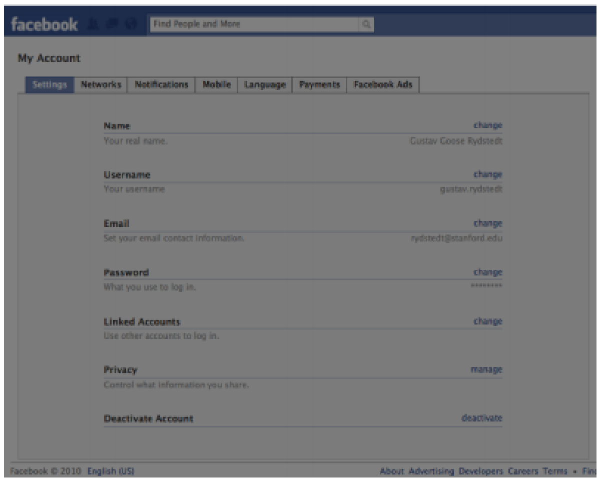
```
<script> var location="dummy"; </script>
<iframe
src="http://www.victim.com"></iframe>
```

Other issues like double framing, onunload event, restricted frames, referrer checking exist as well.

Breaking facebook's frame busting script

Facebook's technique to battle Clickjacking using frame busting script is radically different from popular techniques. Their script inserts a gray semi-transparent div that covers all of the content when a profile page is framed. This disables the user from further clicking on any active clickable area inside the iframe and when the user clicks anywhere on the div, Facebook busts out of the frame. The following is the script used –

```
if (top != self) {
      window.document.write("<div
      style='background:black; opacity:0.5;
      filter:alpha(opacity=50);
      position:absolute; top:0px; left:0px;
        width:9999px; height:9999px;
        z-index:1000001'
      onClick='top.location.href=window.loc
      ation.href'></div>");
}
```



My Account

Name — change
Your real name. — Gustav Goose Rydstedt

Username — change
Your username — gustav.rydstedt

Email — change
Set your email contact information. — rydstedt@stanford.edu

Password — change
What you use to log in. — ********

Linked Accounts — change
Use other accounts to log in.

Privacy — manage
Control what information you share.

Deactivate Account — deactivate

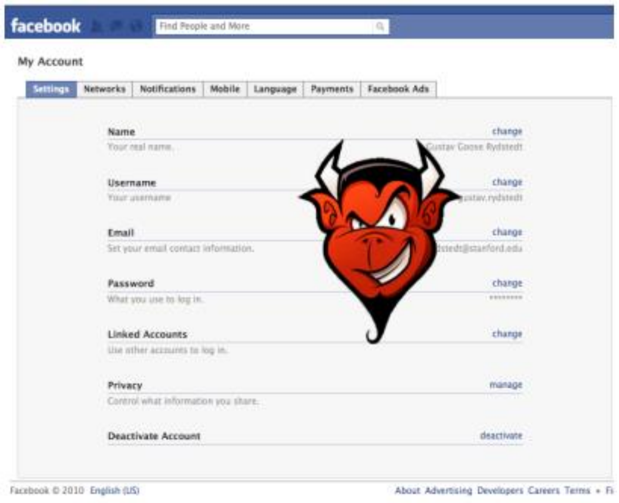
Facebook © 2010 English (US)

Figure 9 – semi-opaque div covers the page

But this code is not completely secure. Since all Facebook's content except this div is centered in the frame, this framing defense can be defeated by making the enclosing frame sufficiently large so that the center of the frame is outside the dark div area. The content naturally moves to the center of the frame and is shown to the user without the dark overlay. Using the following iframe defeats this frame busting technique. Note that the scrollTo function dynamically scrolls to the center of the frame where the content appears in the clear.

```
<body style="overflow-x:hidden; border:0px;
margin:0px;">
<iframe width="21800px" height="2500px"
      src="http://facebook.com/"
      frameborder="0" marginheight="0"
      marginwidth="0"></iframe>

<script>window.scrollTo(10200,0);</script>
```



Name — change
Your real name. — Gustav Goose Rydstedt
Username — change
Your username — gustav.rydstedt
Email — change
Set your email contact information. — dstedt@stanford.edu
Password — change
What you use to log in. — ********
Linked Accounts — change
Use other accounts to log in.
Privacy — manage
Control what information you share.
Deactivate Account — deactivate

Facebook © 2010 English (US) About Advertising Developers Careers Terms • Fi

Figure 10 – busting facebook's frame busting

As proposed by Rydstedt, Bursztein, Boneh and Jackson in their paper "Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites" the following is a secure frame busting script –

```
<style>
html {display:none;}
</style>
<script>
      if(self == top)
      {document.documentElement.style.displ
      ay='block';
      } else {
            top.location=self.location;
      }
</script>
```

Conclusion

Clickjacking isn't big enough right now but is predicted to feature in the next version of The Open Web Application Security Project (OWASP)Top 10 list which features problems like injection, broken authentication and session management, XSS etc.

iframes are now largely adopted on internet and have overcome traditional frames which has made it a possible new attack vector.

Clickjacking is not a preferred attack by attackers because it is complicated to set up and not easily portable (different browsers/configurations render a page differently).

Be careful of links that open in a new window. Check your social networking apps and revoke their access to other accounts if not in use.

More research is needed in this area on how to increase the browser security against Clickjacking. The purpose of including the "Challenges for Current Solutions" section in the paper was to make the reader realize that solutions do exist for Clickjacking attacks but they all have their limitations. Web applications are only secure to a certain degree against this attack. As a user, the best way to protect you from this problem is to use the browser plugins. These have been tested well against the most recent attacks and are under constant improvement from the community. As a developer, the best protection from this attack is to add X-Frame-Options header to your web projects for now.

Sources

"Clickjacking", Robert Hansen (SecTheory) & Jeremiah Grossman (WhiteHat Security) [SecTheory]
http://www.sectheory.com/clickjacking.htm

[Wikipedia.org]
http://en.wikipedia.org/wiki/Clickjacking

"Stealing Mouse Clicks for Banner Fraud", Larry Suto [ha.ckers]
http://ha.ckers.org/blog/20070116/stealing-mouse-clicks-for-banner-fraud/

"Cookie re-use in Office 365 and Other Web Services", Sam Bowne [City College of San Fansisco]
http://samsclass.info/123/proj10/cookie-reuse.htm

[Wikipedia.org]
http://en.wikipedia.org/wiki/Confused_deputy_problem

"A Solution for Automatic Detection of Clickjacking Attacks", Balduzzi, Egele, Kirda, Balzarotti, Kruegel [International Secure Systems Lab]
http://www.iseclab.org/people/embyte/papers/asiaccs122-balduzzi.pdf

"Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites", Rydstedt, Bursztein, Boneh, Jackson [Web 2.0 Security and Privacy 2014]
http://w2spconf.com/2010/papers/p27.pdf

"New Insights into Clickjacking", Marco Balduzzi [Slideshare]
http://www.slideshare.net/embyte/new-insights-into-clickjacking

"Facebook Clickjack Script", Amit Barfa [ktechie.com]
http://ktechie.com/facebook/facebook-clickjack-script/

"UI Redressing and Clickjacking", Marcus Niemietz [Slideshare]
http://www.slideshare.net/DefconRussia/marcus-niemietz-ui-redressing-and-clickjacking-about-click-fraud-and-data-theft

"ClickJacking Niagara" [protecht]
http://www.protecht.ca/blog/clickjacking-niagara

"Clickjacking: Attacks and Defenses", Huang, Moshchuk, Wang, Schechter, Jackson [usenix]
https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final39.pdf

"Clickjacking – UI Redressing", Ronan Dunne [Slideshare]
http://www.slideshare.net/RonanDunne1/click-jacking

"Developer Conference 2011", Krishna [Slideshare]
http://www.slideshare.net/novogeek/clickjacking-devcon2011

[http://blog.kotowicz.net]
http://blog.kotowicz.net/2009/12/new-facebook-clickjagging-attack-in.html

"Clickjack Protection for JavaEE", OWASP
https://www.owasp.org/index.php/ClickjackFilter_for_Java_EE

"Twitter don't click exploit", Chris Shiflett
http://shiflett.org/blog/2009/feb/twitter-dont-click-exploit

Examples

http://www.mniemietz.de/demo/cursorjacking/cursorjacking.html#

http://koto.github.io/blog-kotowicz-net-examples/cursorjacking/