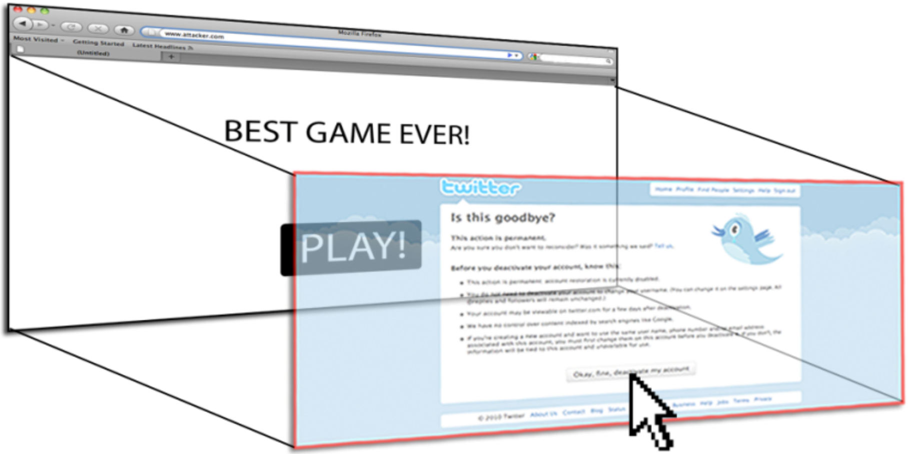
Clickjacking (UI Redressing)


www.attacker.com
Mozilla Firefox
Most Visited
Getting Started
Latest Headlines
(Untitled)
BEST GAME EVER!
PLAY!
twitter
Home Profile Find People Settings Help Sign out
Is this goodbye?
This action is permanent.
Before you deactivate your account, know this:
Okay, fine, deactivate my account
© 2010 Twitter About Us Contact Blog Status

Clickjacking

- portmanteau of “click hijacking”
- attacker overlays multiple transparent or opaque frames
    - trick a user into clicking a button or link on another page
- circumvents same-origin policy
    - malicious page cannot click the link itself
    - requires a user action
        - but all XSRF defences are gone if that click happens

Twitter Clickjack

Users send out tweets against their will.

Twitter Clickjack

Users send out tweets against their will.

Users are tricked into clicking a post-to-twitter link.

Twitter Clickjack

Users send out tweets against their will.

Users are tricked into clicking a post-to-twitter link.

Works if they are logged in

Likejacking: clickjacking in the context of the Facebook like button.

But wait: how isn't this just XSRF?

Clickjacking attack: when a user's mouse click is used in a way that was not intended by user.

Simple Example

- <a
    - onMouseDown=window.open(http://www.evil.com)
    - href=http://www.google.com/>
        - anchor text
- </a>
- anchor goes to evil.com
- why the google.com?

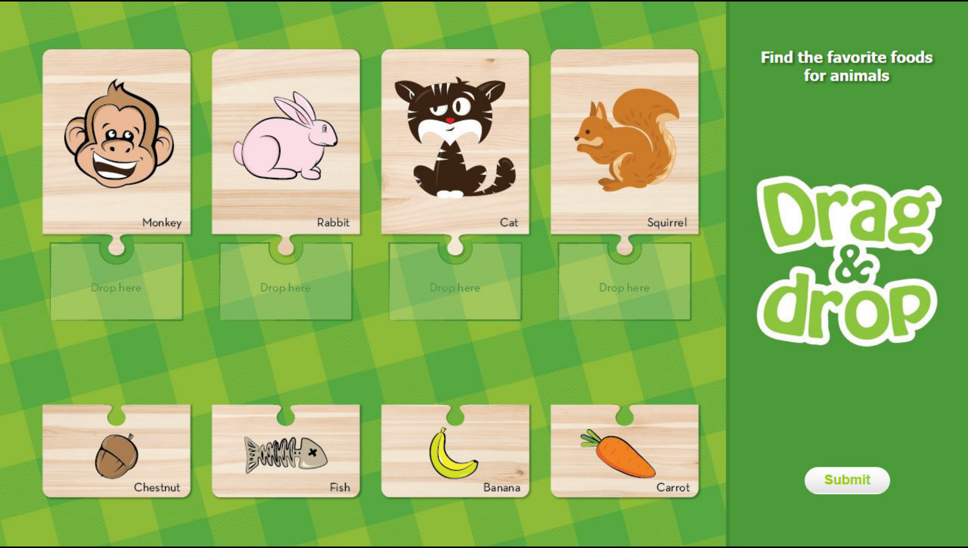
iframes

- any website can frame any other website
  - have a subwindow or such that shows its content
- main frame does not need to handle all the logic of managing two things
  - subframe can be its own session, links clicking, changing page, etc.
- <IFRAME SRC="HTTP://WWW.GOOGLE.COM/..."> </IFRAME>
  - HTML attributes include OPACITY (percentage visible)
    - 1.0: totally visible
    - 0.0: totally invisible
  - Z-INDEX: position on the stack (top gets clicks)
  - POINTER-EVENT: set to NONE to say ignore click (goes to next)

Drag-and-Drop Abuse

- same origin policy stops the html page to “see” what the user selects in an iframe
    - e.g., iframe_text_field.textContents throws an exception
- but selected text can be dragged into an object despite same origin
    - motive is that user does this deliberately
    - i.e., mouse events cannot be spoofed

How can this be exploited?



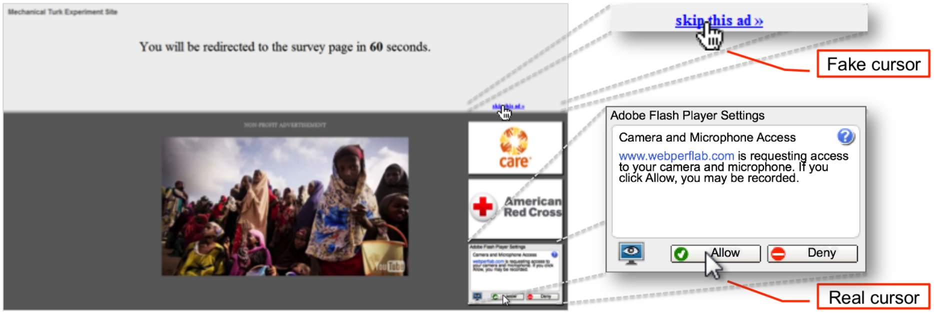
Find the favorite foods for animals

Abusing Drag-n-Drop

- only need to get the user to drag and drop for **any reason**
- hidden iframes will load the data that the evil site wants
- destination will be an HTML object within the evil site's control
- user is tricked into circumventing same origin policy

Cursorjacking

- mouse cursor can be turned off in the web browser
  - CSS CURSOR property supports “none”
- then create another cursor in javascript that follows the mouse movement
  - different looking cursors won’t necessary be suspicious
  - though different cursor physics will be noticable


Mechanical Turk Experiment Site

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

NON-PROFIT ADVERTISEMENT

skip this ad »

Adobe Flash Player Settings

Camera and Microphone Access

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

Strokejacking: suppose that bank.com needed the user to enter in numbers for an amount to do a bank transfer.

Strokejacking: suppose that bank.com needed the user to enter in numbers for an amount to do a bank transfer. That is, clicks aren't enough: the user has to hit keys.

Strokejacking: suppose that bank.com
needed the user to enter in numbers
for an amount to do a bank transfer.
That is, clicks aren't enough:
the user has to hit keys.
SOP stops this from being faked.

Strokejacking

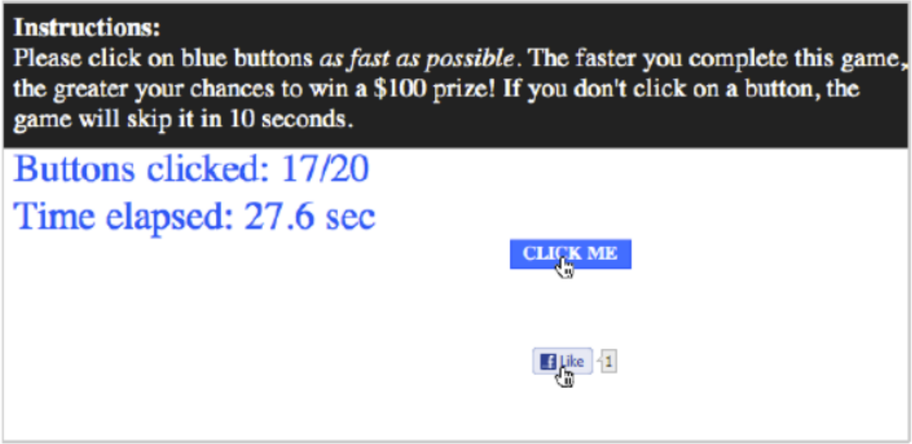
- site convinces the user to type some keystrokes on a simulated input field
- actual keystrokes being sent to the iframe that needs it
- e.g., numbers become the amount to send.
- how could the user be tricked?

All these attacks conspire to break SOP.

All these attacks conspire to break SOP.
They require human effort to click or type
and the user is being tricked into doing that.

Compromise Temporal Integrity

- temporal integrity refers to the state remaining the same in time
  - security issue involving something changing after security check is done but before something being allowed by that check is done
  - TOCTTOU: time of check to time of use
- for clickjacking, it means changing the UI after the user decides to click but before the click occurs
  - e.g., if logic executes on ONCLICK, then change UI on MOUSEDOWN
  - e.g., bait the user to double click, and swap the UI between them

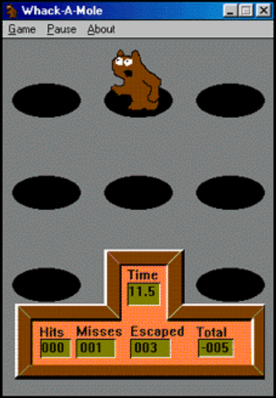

Instructions:
Please click on blue buttons *as fast as possible*. The faster you complete this game, the greater your chances to win a $100 prize! If you don't click on a button, the game will skip it in 10 seconds.

Buttons clicked: 17/20

Time elapsed: 27.6 sec

Whack-A-Mole Attack

- bait the user to click as fast as possible
- switch to a different UI button when appropriate



How do we stop this?

Solution: user confirmation

Solution: user confirmation
Good site pops up dialogue box with info about what it is about to do and confirms

Solution: user confirmation
Good site pops up dialogue box with info about what it is about to do and confirms
awful user experience

Solution: UI Randomization

Solution: UI Randomization
Good site embeds form
elements at random locations
so it is hard to overlay

Solution: UI Randomization
Good site embeds form
elements at random locations
so it is hard to overlay
e.g., paypal pay button
always in different location

Solution: UI Randomization

Good site embeds form elements at random locations so it is hard to overlay

e.g., paypal pay button always in different location

awful user experience

multi-click attack

Solution: Opaque Policy

Solution: Opaque Policy
no element can be transparent

Solution: Opaque Policy
no element can be transparent
each pixel belongs to a single element

Solution: Opaque Policy

no element can be transparent

each pixel belongs to a single element

any problems?

Partial Overlaps and Cropping

- don't completely cover the target
- instead hide the important parts
    - e.g., message that you mean to post
    - e.g., amount that your credit card is charged

Solution: Frame Busting

Solution: Frame Busting
I am the page owner (what gets put in iframe)

Solution: Frame Busting
I am the page owner (what gets put in iframe)
I insist that I am never loaded in an iframe

Solution: Frame Busting

I am the page owner (what gets put in iframe)

I insist that I am never loaded in an iframe

```
if (top != self) top.location.href = location.href;
if (top != self)
if (top.location != self.location)
if (top.location != location)
if (parent.frames.length > 0)
if (window != top)
if (window.top !== window.self)
if (window.self != window.top)
if (parent && parent != window)
if (parent &&
parent.frames &&
parent.frames.length>0)
if((self.parent&&
!(self.parent===self))&&
(self.parent.frames.length!=
```

top.location = self.location
top.location.href = document.location.href
top.location.href = self.location.href
top.location.replace(self.location)
top.location.href = window.location.href
top.location.replace(document.location)
top.location.href = window.location.href
top.location.href = "URL"
document.write('')
top.location = location
top.location.replace(document.location)
top.location.replace('URL')
top.location.href = document.location
top.location.replace(window.location.href)
top.location.href = location.href
self.parent.location = document.location
parent.location.href = self.document.location
top.location.href = self.location
top.location = window.location
top.location.replace(window.location.pathname)

Frame Busting

- conditional check for iframing
  - take counter-action if iframing is detected
  - then no user behaviour on site is result of clickjacking
- doesn’t work for embedded stuff like facebook “like” buttons

So clickjacking is (somewhat) solved!

Frame Busting in the Wild

- researchers surveyed the Alexa top 500 websites and all top US banks
- 14% use framebusting
- found 100% of framebusting can be circumvented one way or another
    - some browser specific
    - some cross browser

Frequently it was in the code to allow their own iframes

Frequently it was in the code to allow their own iframes
i.e., I don't want to be an iframe,
but I want to have my own things as iframes

Frequently it was in the code to allow their own iframes
i.e., I don't want to be an iframe,
but I want to have my own things as iframes
and they are okay with being iframes
as long as I'm still the main frame.

Frequently it was in the code to allow their own iframes
i.e., I don't want to be an iframe,
but I want to have my own things as iframes
and they are okay with being iframes
as long as I'm still the main frame.
This policy can be hard to implement.

Walmart’s Framebusting

- if (top.location != location)
  - if (document.referer &&
  - document.referer.indexOf(”walmart.com”) == -1)
    - top.location.replace(document.location.href);
- document.referer is a string
- indexOf returns -1 for string not found

Error in Referrer Checking:
website http://www.attacker.com/walmart.com.html has the iframe

The New York Times's Framebusting

- if (window.self != window.top && !document.referer.match(/https?://[^?\/]+\.nytimes\.com\//))
- self.location = top.location;

Error in Referer Checking:
website
http://eve.com/a.html?b=https://www.nytimes.com/
has the iframe

US Bank's Framebusting

- if (self != top)
    - var domain = getDomain(document.referer);
    - var okDomains = /usbank|localhost|usbnet/;
    - var matchDomain = domain.search(okDomains);
    - if (matchDomain == -1)
        - // frame bust

Error in Referer Checking:
website http://usbank.attacker.com has the iframe

Error in Referer Checking:
website http://usbank.attacker.com has the iframe
or the Norwegian State House Bank
http://www.husbanken.no

Error in Referer Checking:
website http://usbank.attacker.com has the iframe
or the Norwegian State House Bank
http://www.husbanken.no
or the Rusbank http://www.rusbank.org
(its actually Rosbank, but still)

Typical Frame Busting code:

```
if (parent.location != self.location)
parent.location = self.location
```

Double Framing Attack:
main frame has <iframe src="frame2.html">

Double Framing Attack:
main frame has <iframe src="frame2.html">
frame2.html has <iframe src="victim.com">

A site may trust another site
to allow them in iframes

A site may trust another site
to allow them in iframes
but what if that site does
not implement framebusting?

A site may trust another site
to allow them in iframes
but what if that site does
not implement framebusting?
e.g., main frame has <iframe src="trusted.com">
that frame has <iframe src="victim.com">

A site may trust another site
to allow them in iframes
but what if that site does
not implement framebusting?
e.g., main frame has <iframe src="trusted.com">
that frame has <iframe src="victim.com">
victim avoid framebusting for trusted

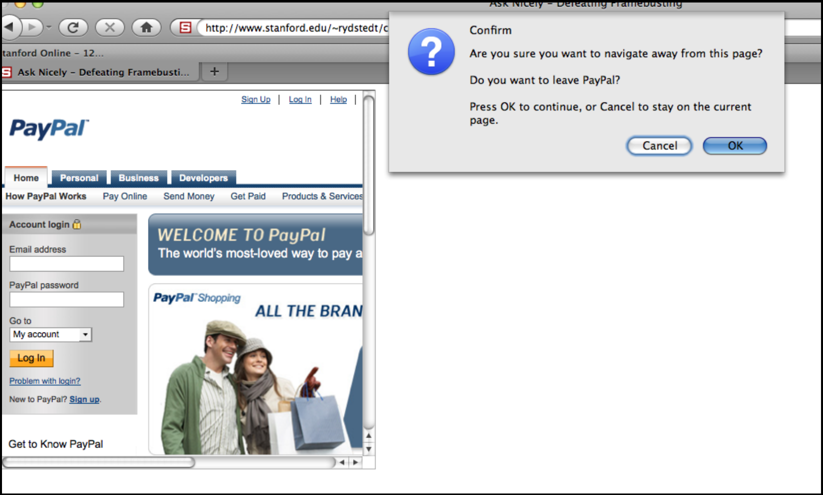
A fix?

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

Location Clobbering

- IE7: var location="clobbered";
- Safari: window.__defineSetter__("location", function(){})

Asking Nicely



Confirm

Are you sure you want to navigate away from this page?

Do you want to leave PayPal?

Press OK to continue, or Cancel to stay on the current page.

Frame busting from Paypal will be cancelled if the user clicks cancel.

Frame busting from Paypal will be cancelled if the user clicks cancel. The pop-up is actually the iframer's onbeforeunload() function.

Best at the time

- style html's body as "display: none"
- try to framebust if "self !== top"
- change style to "display: block" if "self === top"

Then introduced

- X-Frame-Options HTTP header sent with page
- two possible values: **DENY** and **SAMEORIGIN**
- DENY: page will not render if framed
- SAMEORIGIN: page will only render if top frame has same origin
- addresses the main issue about ad hoc anti-framebusting that allowed first party

Content Security Policy

- standardized in 2012
- meant to protect against XSS and clickjacking more comprehensively
    - not in the ad hoc ways
- implemented as HTTP response header
    - semicolon-space separated list of directives

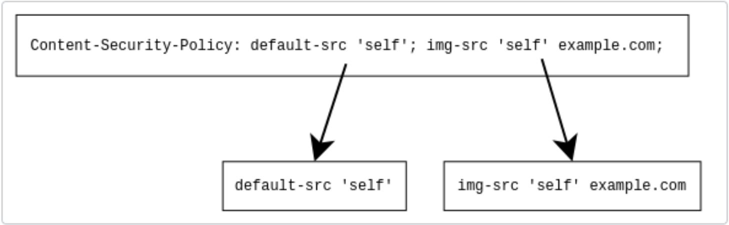
HTTP

Copy

```http
Content-Security-Policy: default-src 'self'; img-src 'self' example.com
```

It sets two directives:

- the `default-src` directive is set to `'self'`
- the `img-src` directive is set to `'self' example.com`.



Content-Security-Policy: default-src 'self'; img-src 'self' example.com;
default-src 'self'
img-src 'self' example.com

Content Security Policy

- script-src nonce
  - includes a random number in the response header
    - Content-Security-Policy: script-src 'nonce-kjshdf87sd'
  - server puts that number with all script tags
    - <script nonce="kjshdf87sd">console.log("will run");</script>
  - browser refuses to run any JavaScript that does not have that nonce
  - attacker cannot read the DOM to see the nonce
  - attacker cannot guess a large random nonce

Content Security Policy

- script-src hash
  - include the sha256 of the JavaScript file that will be loaded
  - browser loads the JavaScript and checks that the hash matches
  - browser refuses to run JavaScript that doesn't match
    - supply chain compromise
- default-src https:
  - browser will insist to do all resource loads via HTTPS

Content Security Policy

- if there is script-src or default-src headers then dangerous JavaScript is disabled
  - inline JavaScript is disabled
    - <script>console.log("hi")</script>
    - <img src="missing.png" onerror=console.log("hi") />
    - <a href="javascript:console.log('hi')" />
  - eval function is disabled
  - Function constructor is disabled
    - e.g., const sum = new Function("a", "b", "return a + b");

Content Security Policy

- frame-ancestor directive addresses clickjacking
    - replaces X-Frame-Options
- Content-Security-Policy: frame-ancestors 'none'
    - do not permit iframing
- Content-Security-Policy: frame-ancestors 'self' https://www.example.org
    - allow self and www.example.org to iframe
        - does not allow others even if example.org would
    - self is same-origin
- Content-Security-Policy: frame-ancestors https://⋆.samesite.com
    - if samesite.com is the domain, implements same-site allowed

Summary

- clickjacking / UI redress takes many forms
- tricks user into violating the same origin policy on themselves
    - nothing can stop a user from actually typing out numbers and clicking send money
    - primary reason for doing banking and then logging out
- framebusting meant to avoid iframing
    - hacks to allow self iframing fraught
- content security policy now provides comprehensive fix