

# **UI-based attacks**

# Clickjacking attacks

- Exploitation where a user's mouse click is used in a way that was not intended by the user

# Simple example

```
<a  
  onMouseDown=window.open(http://www.evil.com)  
  href=http://www.google.com/>  
Go to Google</a>
```

What does it do?

- Opens a window to the attacker site

Why include href to Google?

- Browser status bar shows URL when hovering over as a means of protection

# Recall: Frames

- A frame is used to embed another document within the current HTML document
- Any site can frame another site
- The `<iframe>` tag specifies an inline frame

# What happens in this case?

The screenshot shows a web browser window with the address bar set to `funnycats.com`. The browser's address bar and search bar are visible. Below the browser window, the text "Funny cats website" is displayed. A red arrow labeled "JavaScript" points to a red sign-in form on the Bank of America website. The form contains the text "Secure Sign-in" and two input fields, both containing the word "secret". A "Sign In" button is visible next to the second input field. Below the input fields, there are links for "Save Online ID", "Security & Help", "Forgot ID", "Forgot Passcode", and "Enroll". The Bank of America logo is visible at the top of the page. The browser's address bar shows the URL `funnycats.com`. The browser's search bar contains the text "Search". The browser's address bar also shows the text "Gmail", "News", "PreVeil Email", "Safeway - Groceri...", "Instacart - Whole ...", "CS 294, Fall 2011", and "Bear Facts Faculty...".

Same-origin policy prevents this access

# How to bypass same-origin policy for frames?

Clickjacking

# Clickjacking using frames

Evil site frames good site

Evil site covers good site by putting dialogue boxes or other elements on top of parts of framed site to create a different effect

Inner site now looks different to user

# Compromise visual integrity – target

- Hiding the target
- Partial overlays

Lin-Shung Huang  
[Not you?](#) | [Log out](#)

PayPal

**You are about to pay**

Receiver	Amount
Adblock Plus	<b>\$0.15</b>
Total	

Pay with:

[My PayPal Balance](#) [View PayPal policies.](#)

BANK OF AMERICA, N.A. XXX

**\$0.15**

Memo: Contribution for Adblock Plus

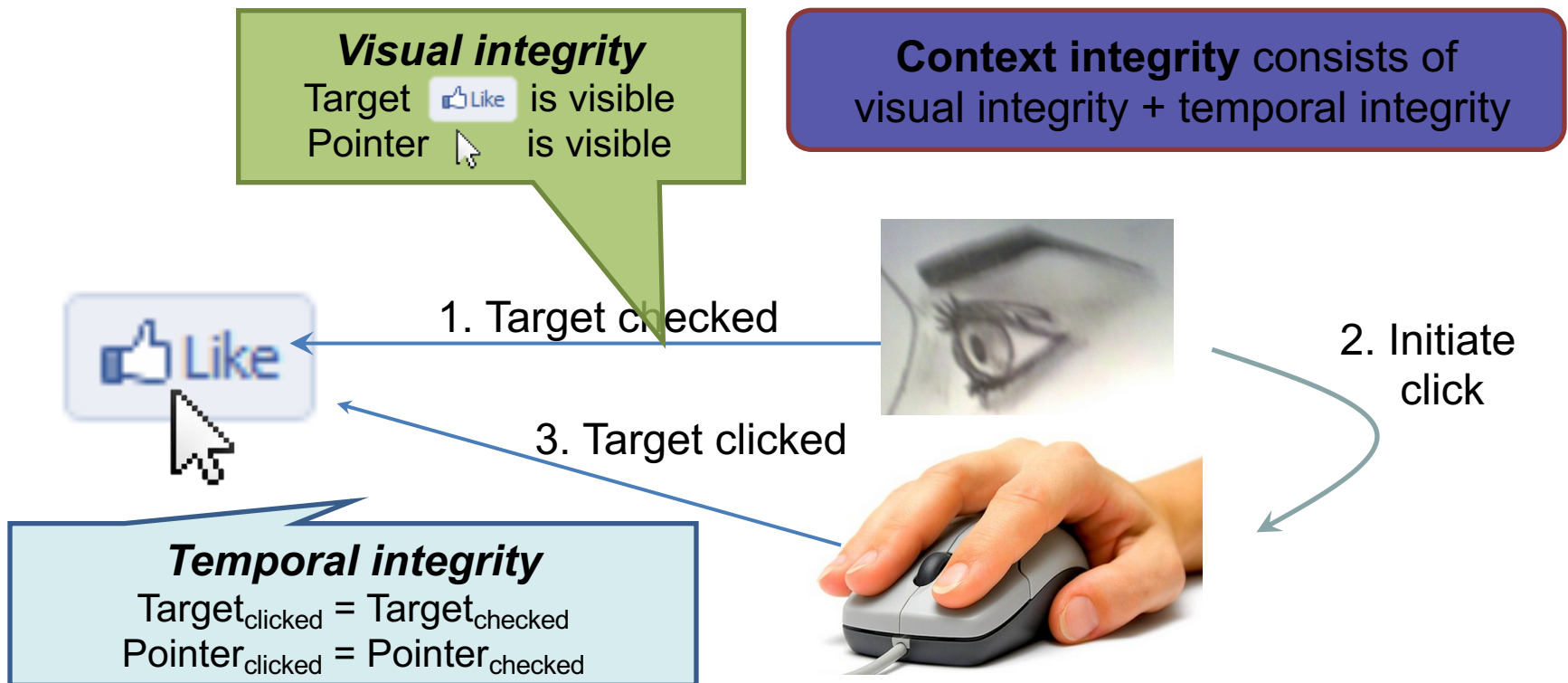
[Pay](#) [Cancel](#)

PayPal protects your privacy and security. [+]



# UI Subversion: *Clickjacking*

- An attack application (script) compromises the *context integrity* of another application's **User Interface** when the user acts on the **UI**



# Compromise visual integrity – target

- Hiding the target
- Partial overlays

Lin-Shung Huang  
[Not you?](#) | [Log out](#)

PayPal

**You are about to pay**

Receiver	Amount
Adblock Plus	<b>\$0.15</b>
Total	

Pay with:

[My PayPal Balance](#) [View PayPal policies.](#)

BANK OF AMERICA, N.A. XXX

**\$0.15**

Memo: Contribution for Adblock Plus

[Pay](#) [Cancel](#)

PayPal protects your privacy and security. [+]

# Compromise visual integrity – pointer: cursorjacking

- Can customize cursor!

CSS example:

```
#mycursor {  
  cursor: none;  
  width: 97px;  
  height: 137px;  
  background: url("images/custom-cursor.jpg")  
}
```

- Javascript can keep updating cursor, can display shifted cursor



**Fake cursor, but more  
visible**

**Real cursor**

# Compromise visual integrity – pointer: cursorjacking

Cursorjacking deceives a user by using a custom cursor image, where the pointer was displayed with an offset



**Fake, but more visible**

**real**

# Clickjacking to Access the User's Webcam



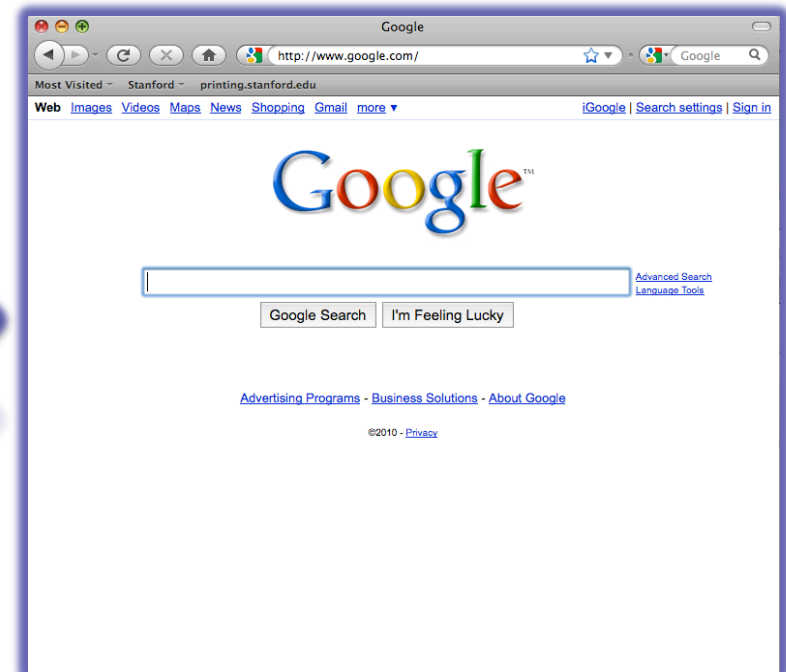
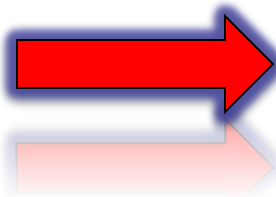
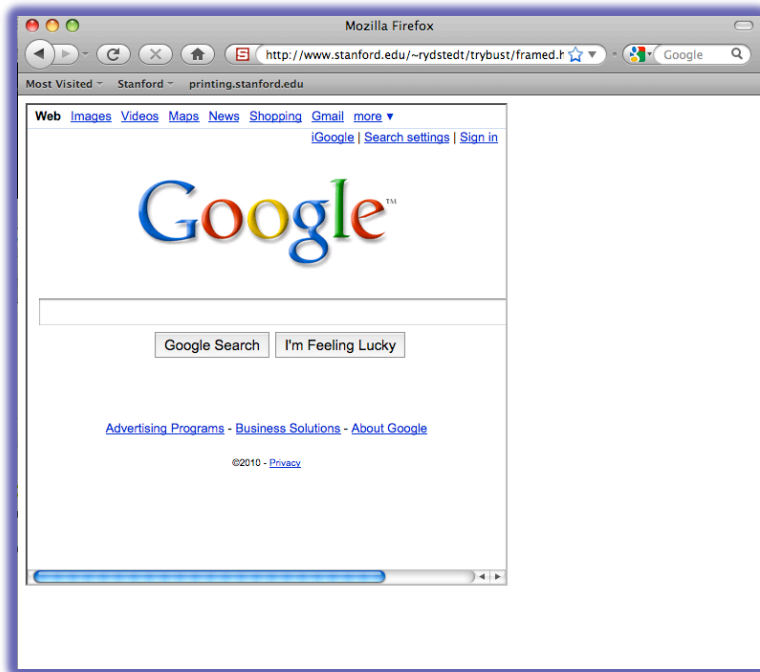
**How can we defend against  
clickjacking?**

# Defenses

- **User confirmation**
  - Good site pops dialogue box with information on the action it is about to make and asks for user confirmation
  - Degrades user experience
- **UI randomization**
  - good site embeds dialogues at random locations so it is hard to overlay
  - Difficult & unreliable (e.g. multi-click attacks)

# Defense 3: Framebusting

Web site includes code on a page that prevents other pages from framing it





# What is framebusting?

Framebusting code is often made up of

- a conditional statement and
- a counter action

Common method:

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

# A Survey

Framebusting is very common at the Alexa Top 500 sites

[global traffic rank of a website]

Sites	Framebusting
Top 10	60%
Top 100	37%
Top 500	14%

# Many framebusting methods

## Conditional Statements

```
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!=0))
```

# Many framebusting methods

## Counter-Action Statements

```
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
```

Most current framebusting  
can be defeated

# Easy bugs

Goal: bank.com wants only bank.com's sites to frame it

**Bank runs this code to protect itself:**

```
if (top.location != location) {  
    if (document.referrer &&  
        document.referrer.indexOf("bank.com") == -1)  
    {  
        top.location.replace(document.location.href);  
    }  
}
```

**Problem:** <http://badguy.com?q=bank.com>

# Defense: Ensuring visual integrity of pointer

- Remove cursor customization
  - Attack success: 43% -> 16%



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

[skip this ad >](#)

NON-PROFIT ADVERTISEMENT



Adobe Flash Player Settings

Camera and Microphone Access  
webportlab.com is requesting access to  
your camera and microphone. If you click  
Allow, you may be recorded.

Allow  Deny

# Ensuring visual integrity of pointer

- Freeze screen outside of the target display area when the real pointer enters the target
  - Attack success: 43% -> 15%
  - Attack success (margin=10px): 12%
  - Attack success (margin=20px): 4% (baseline:5%)



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

[skip this ad >](#)

NON-PROFIT ADVERTISEMENT



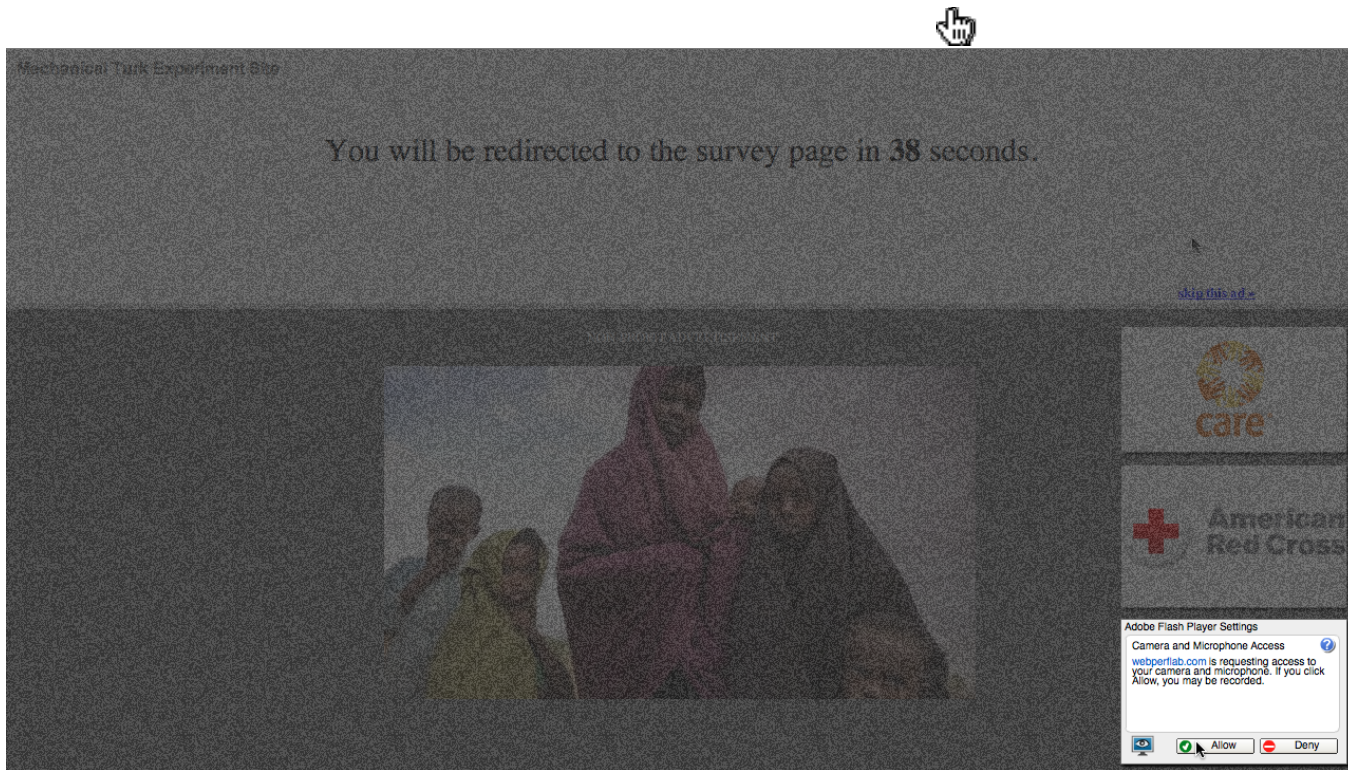
Margin=20px





# Ensuring visual integrity of pointer

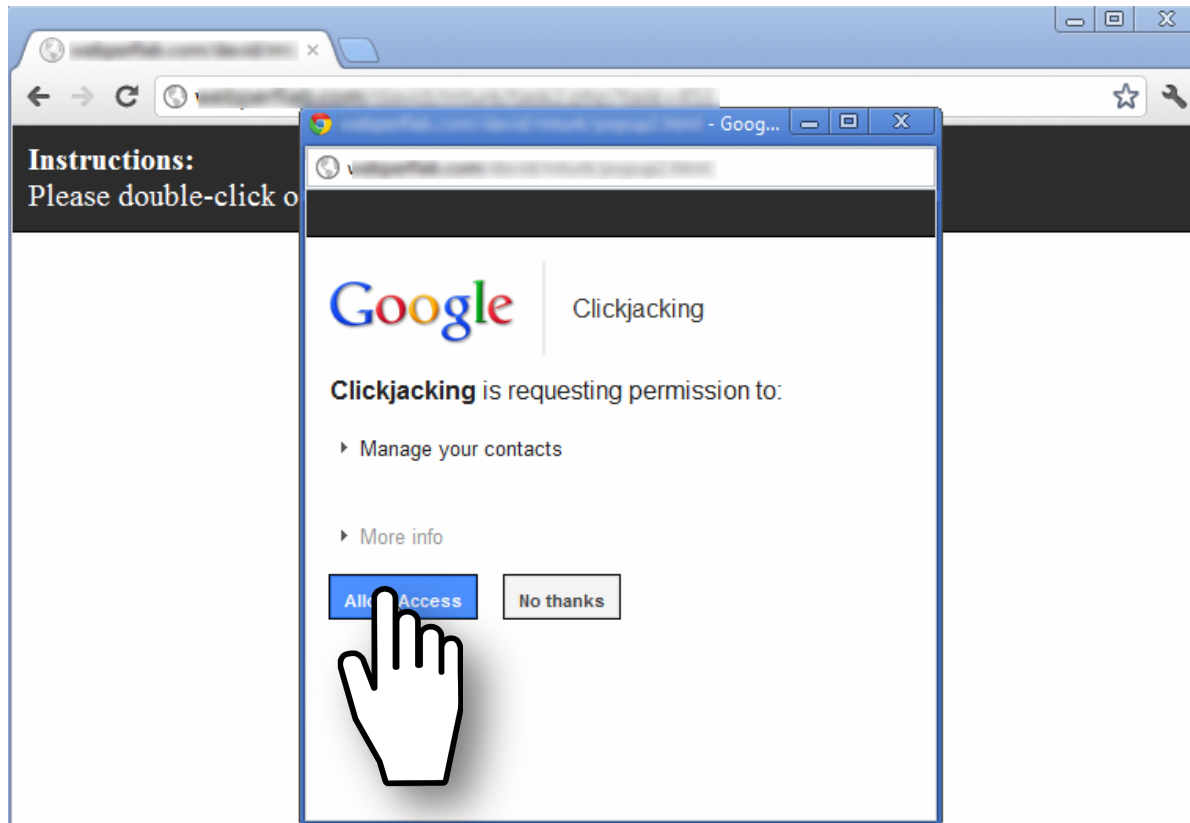
- Lightbox effect around target on pointer entry
  - Attack success (Freezing + lightbox): 2%



**How about a temporal integrity attack  
example?**

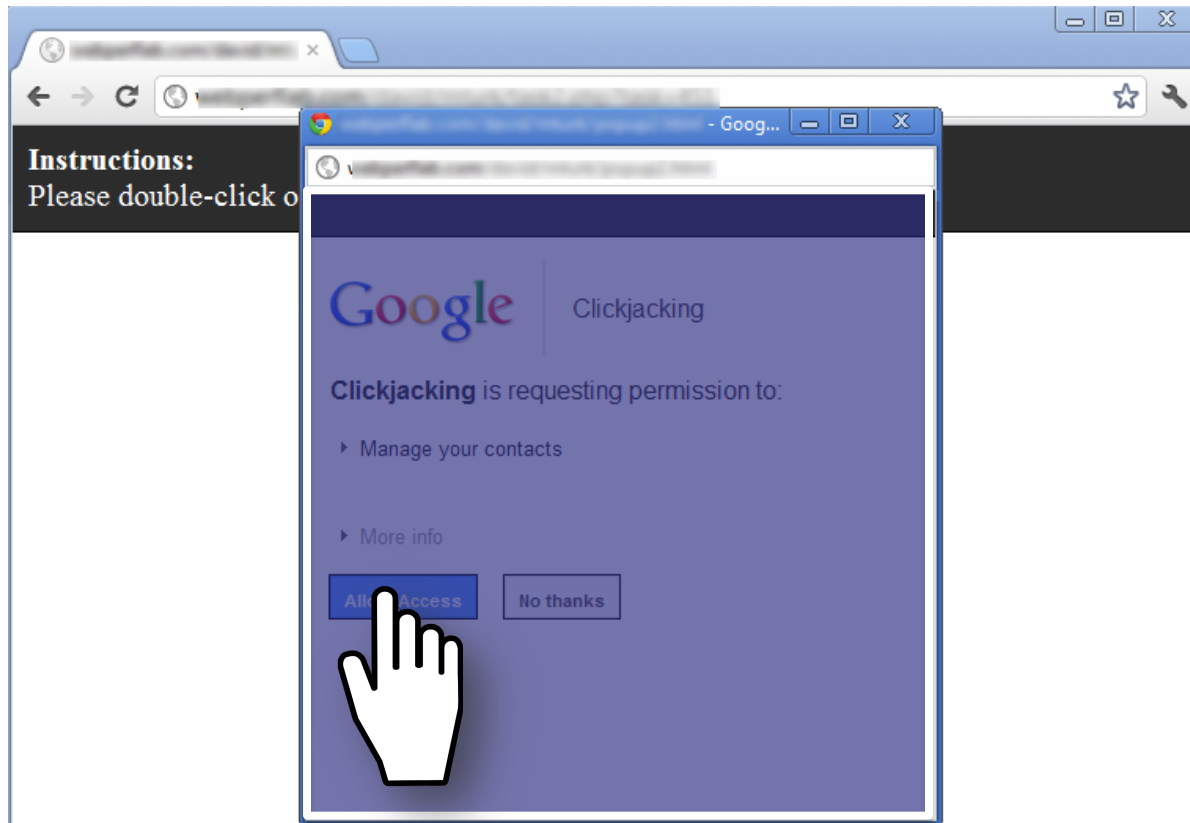
# Temporal clickjacking

- ◆ As you click on a button for an insensitive action, a button for a sensitive action appears overlaid and you click on it by mistake



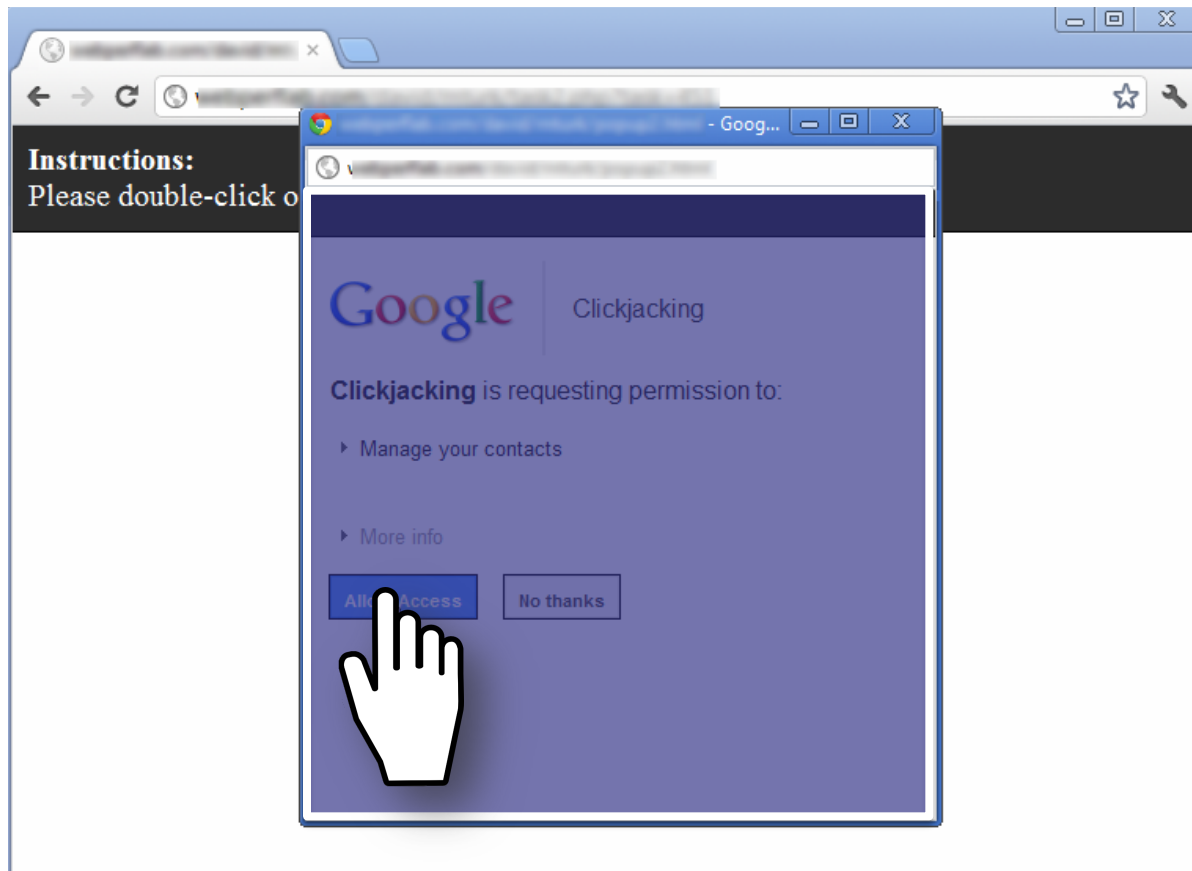
# Enforcing temporal integrity

- UI delay: after visual changes on target or pointer, invalidate clicks for X ms
  - Attack success (delay=250ms): 47% -> 2% (2/91)
  - Attack success (delay=500ms): 1% (1/89)

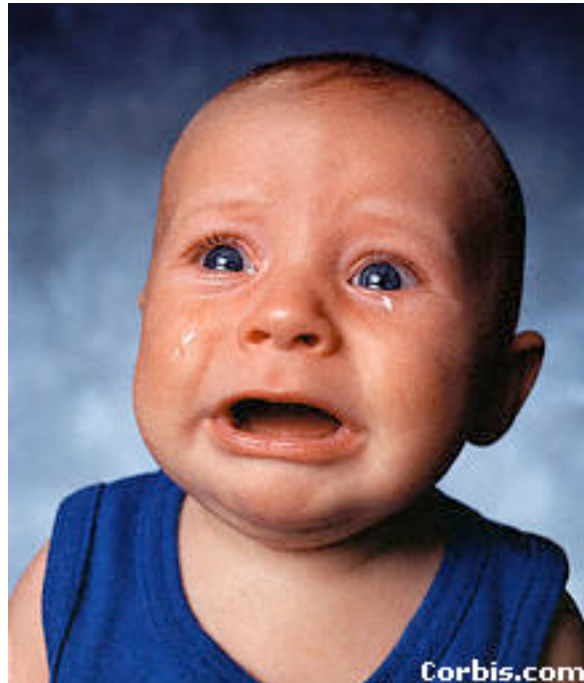


# Enforcing temporal integrity

- Pointer re-entry: after visual changes on target, invalidate clicks until pointer re-enters target
  - Attack success: 0% (0/88)



Is there any hope?



# Other defense: X-Frames-Options

(IE8, Safari, FF3.7)

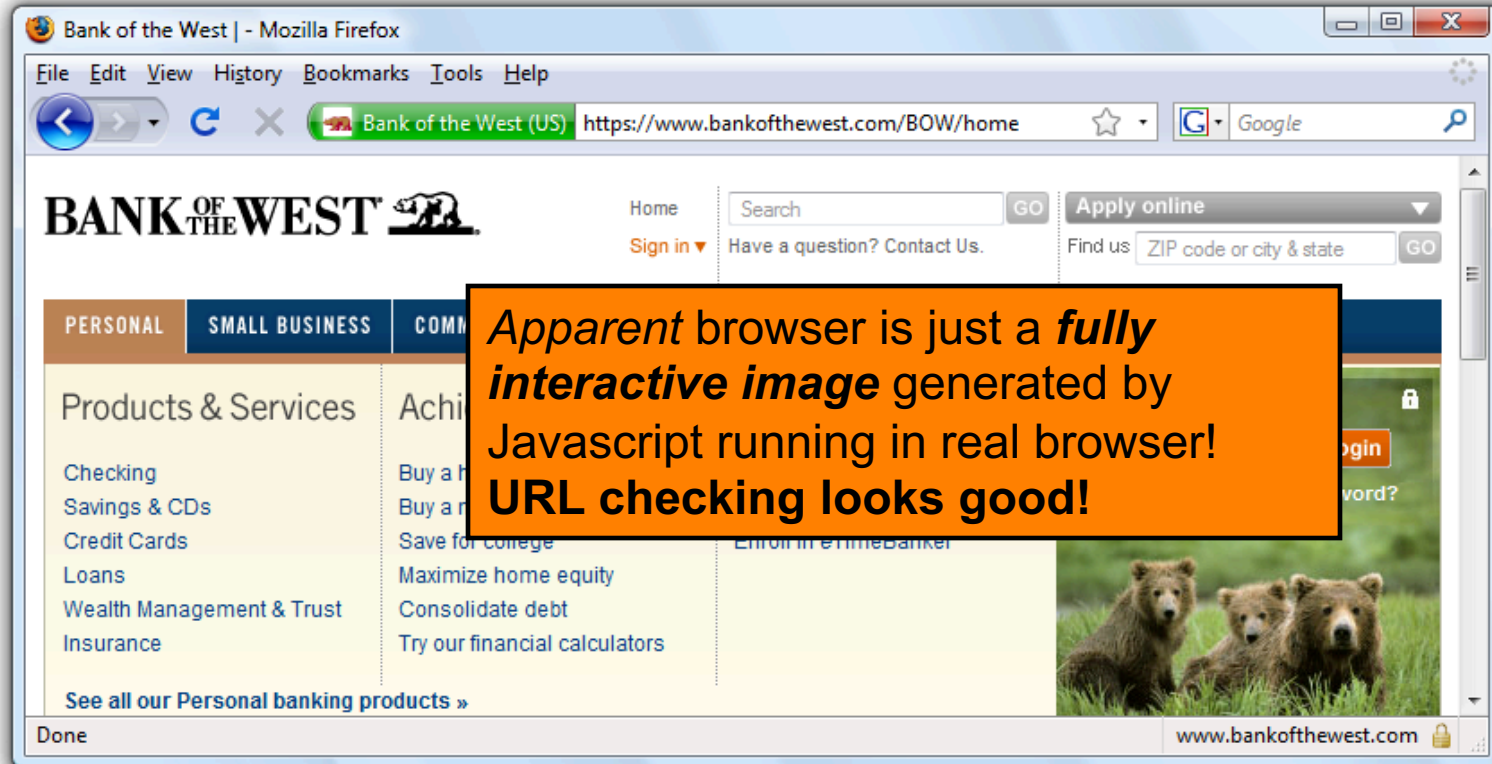
- Web server attaches HTTP header to response
- Two possible values: **DENY** and **SAMEORIGIN**
  - **DENY**: browser will not render page in framed context
  - **SAMEORIGIN**: browser will only render if top frame is same origin as page giving directive
- Good defense ... but poor adoption by sites (4 of top 10,000)
- Coarse policies: no whitelisting of partner sites, which should be allowed to frame our site

# Other Forms of UI Sneakiness

- Users might find themselves living in *The Matrix* ...



# “Browser in Browser”



# Summary

- Clickjacking is an attack on our perception of a page based on the UI
- Framebusting is tricky to get right
  - All currently deployed code can be defeated
- Use X-Frame-Options