# 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

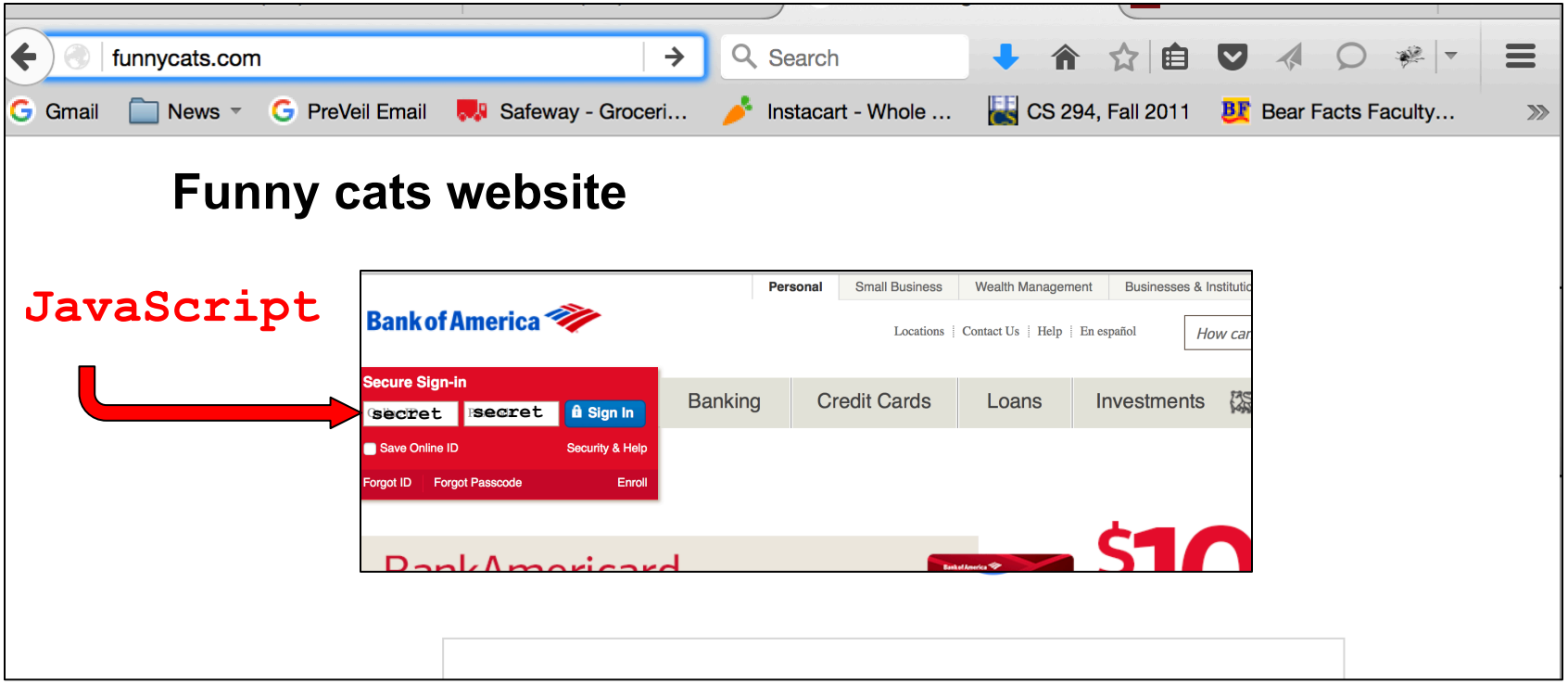
Why include `href` to Google?

- 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?




Same-origin policy prevents this access

# How to bypass same-origin policy for frames?

Clickjacking

# Clickjacking using frames

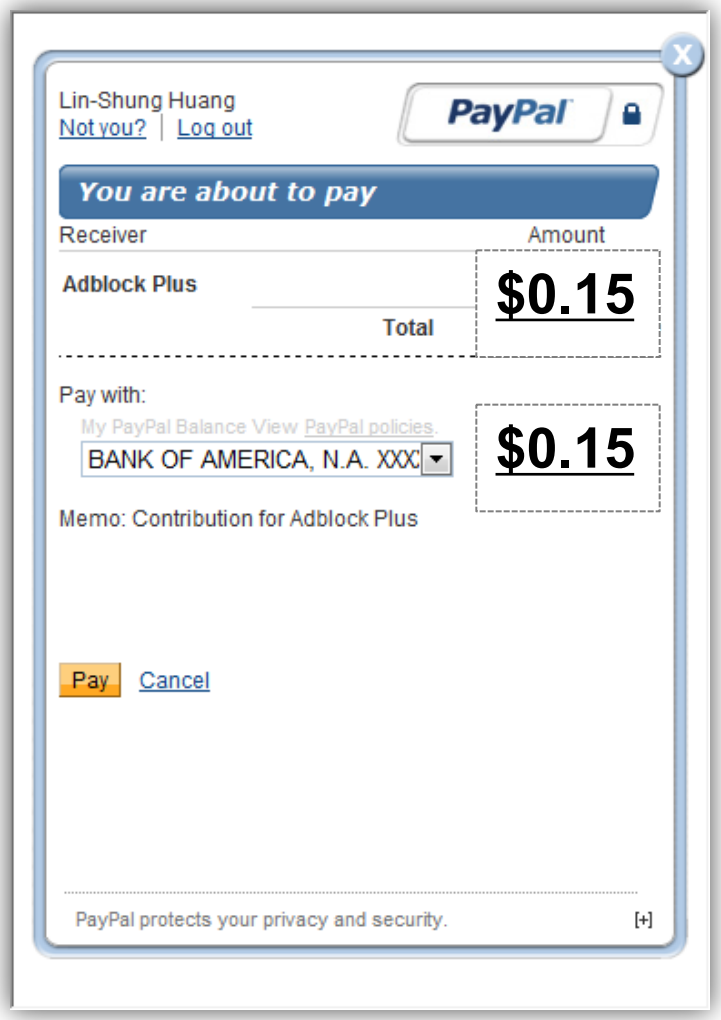
Evil site frames good site

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

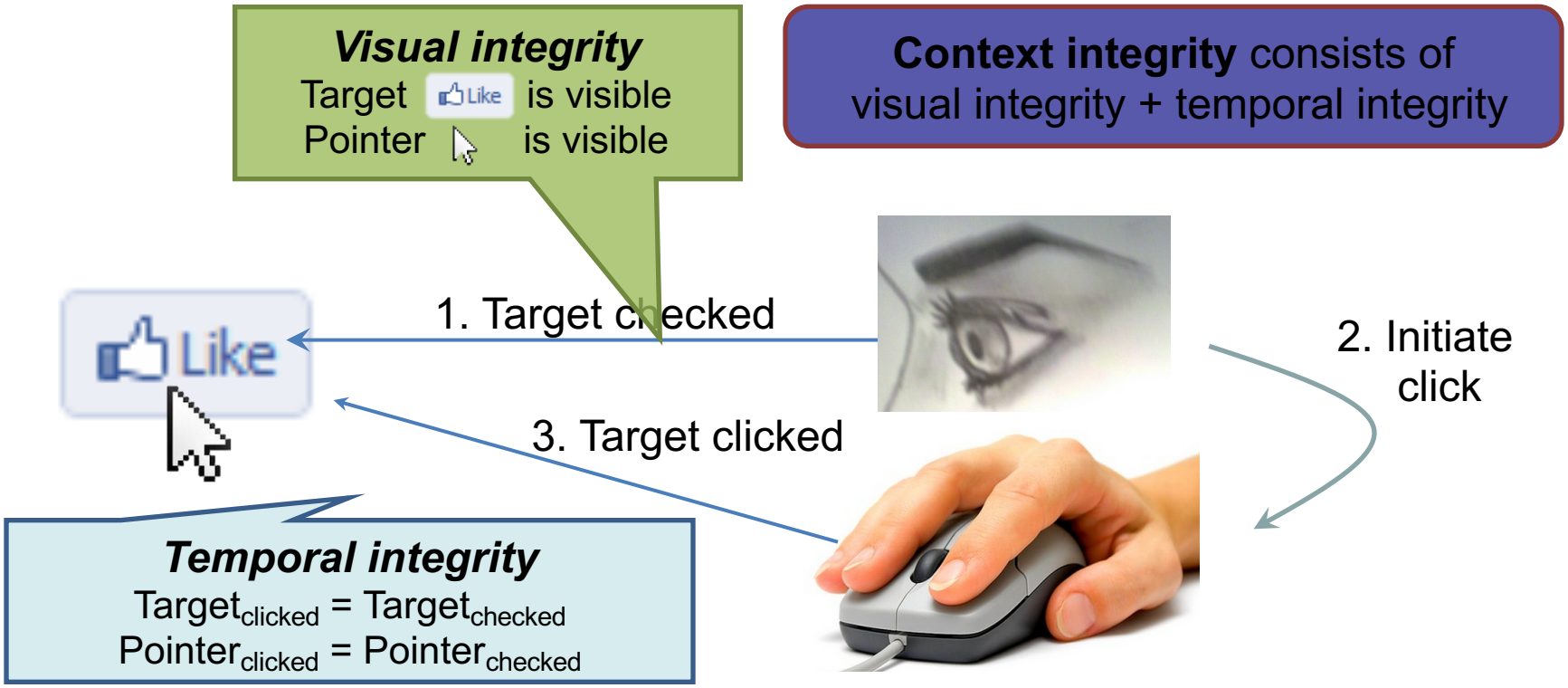
# Compromise visual integrity – target

- Hiding the target
- Partial overlays



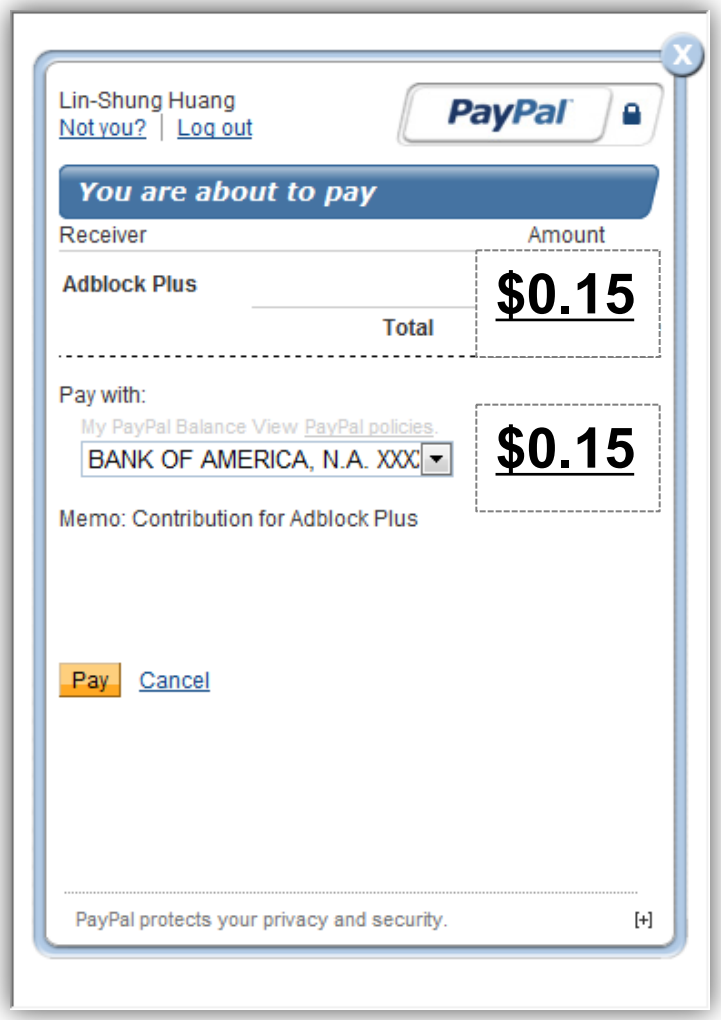
# UI Subversion: *Clickjacking*

- 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



# 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

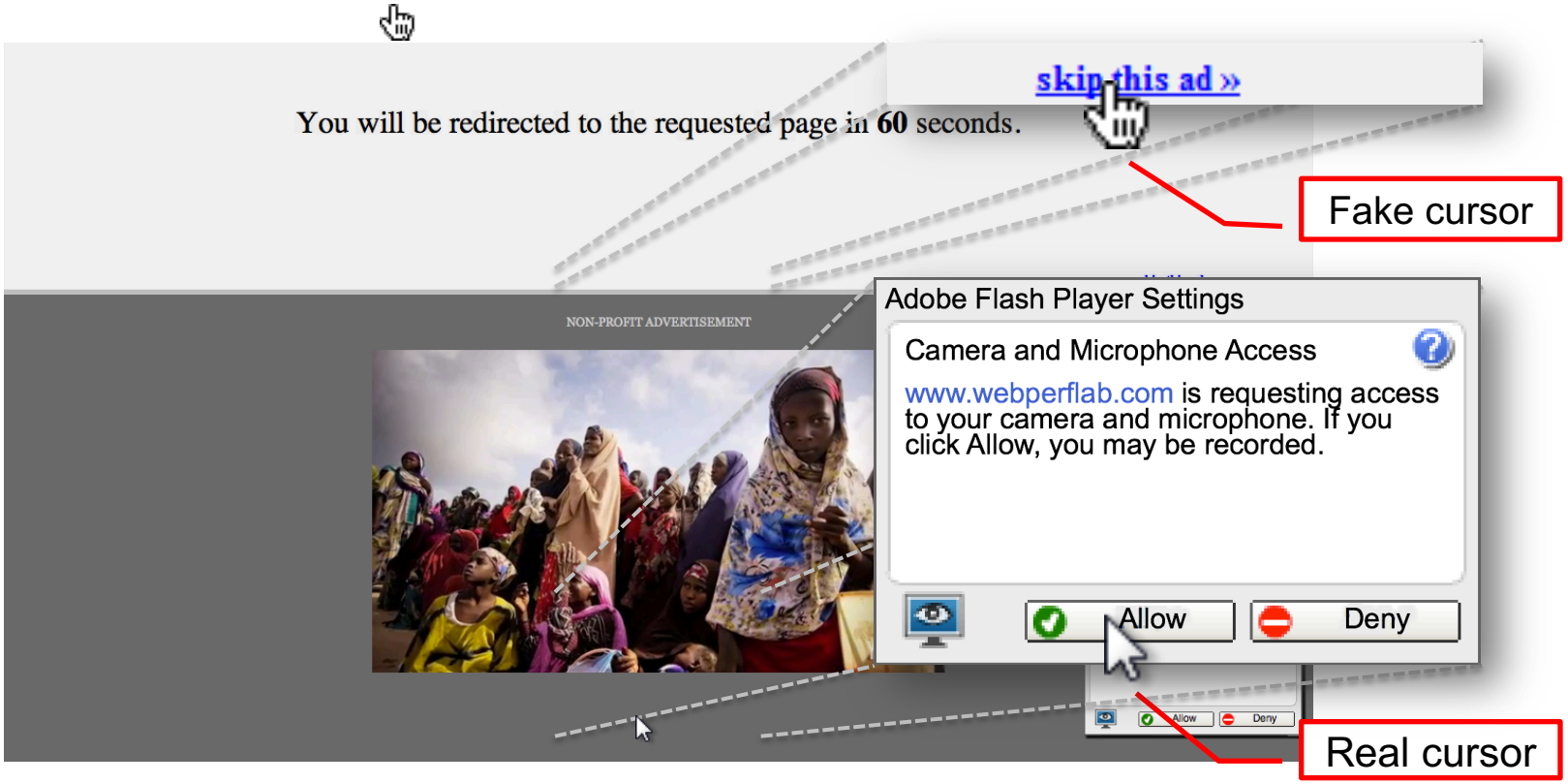
Cursorjacking deceives a user by using a custom cursor image, where the pointer was displayed with an offset



**Fake, but more visible**

**real**

# Clickjacking to Access the User's Webcam



# How can we defend against clickjacking?

# Defenses

- User confirmation

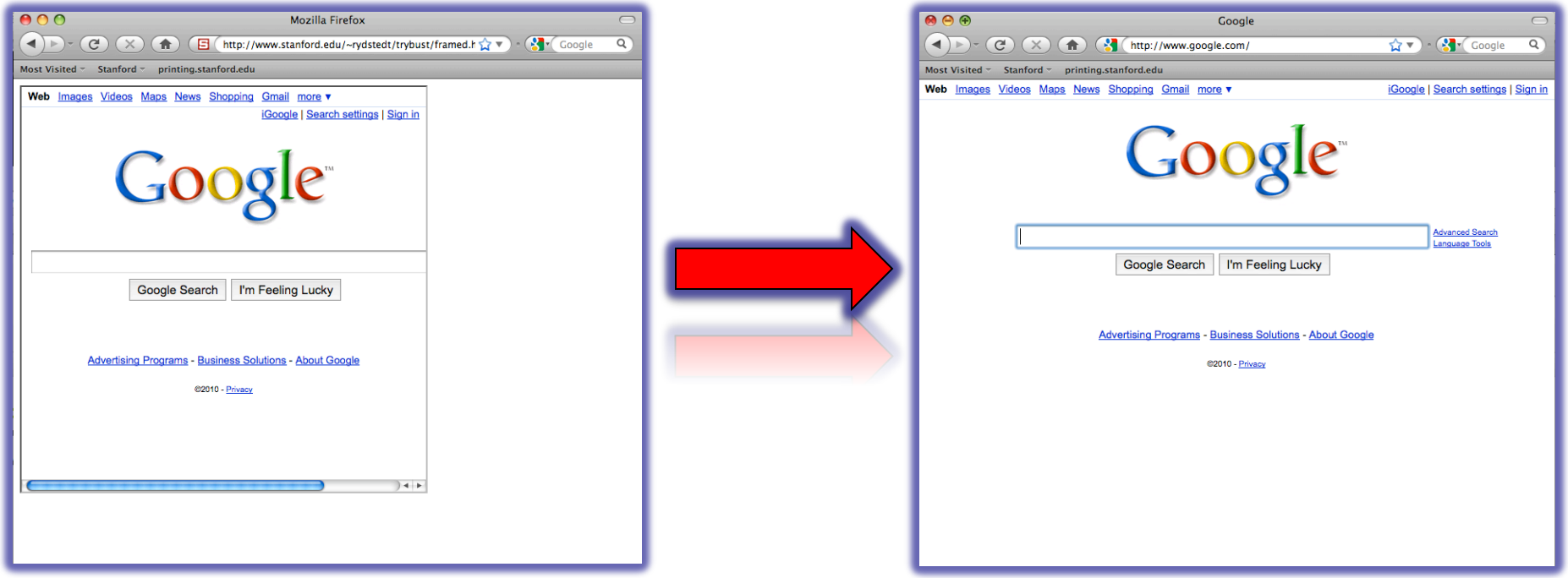
- Good site pops dialogue box with information on the action it is about to make and asks for user confirmation
- Degrades user experience

- 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% |

credit: Gustav Rydstedt

# 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

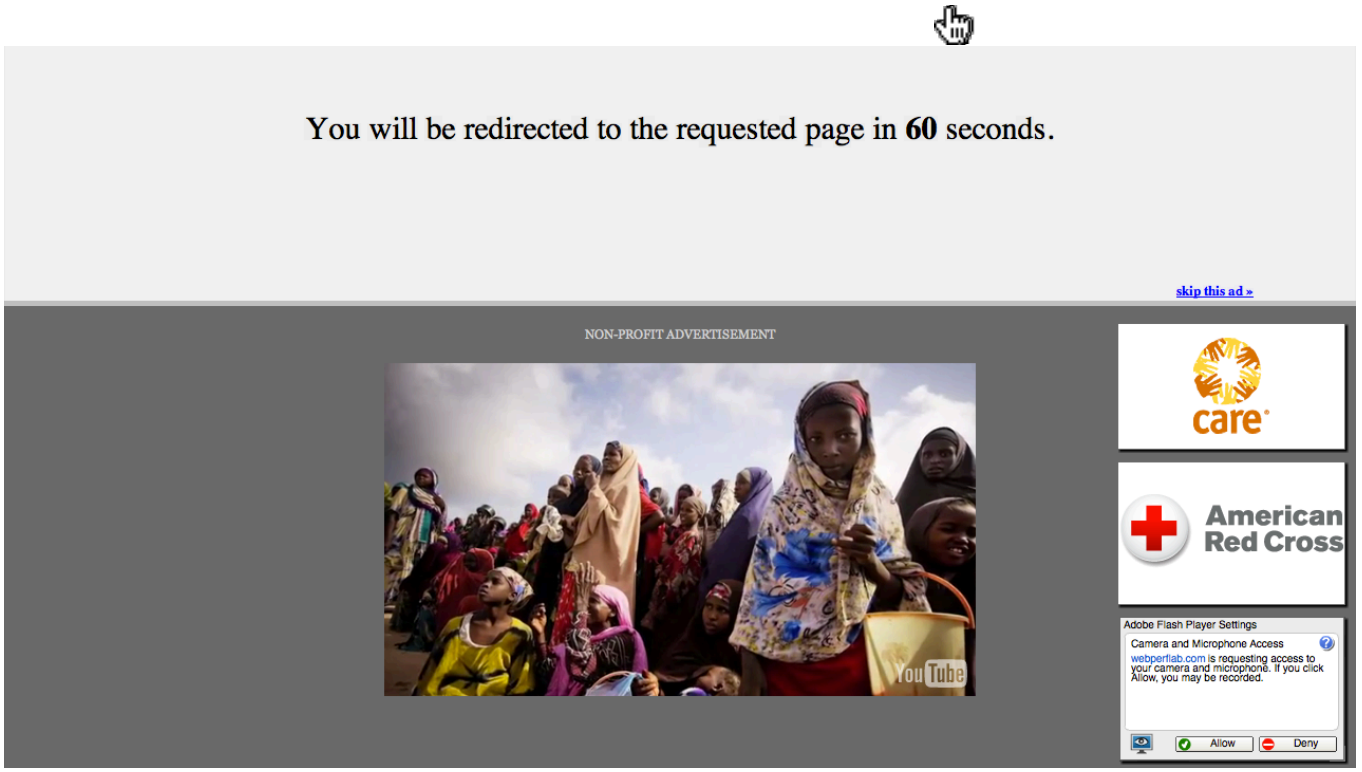
**Bank runs this code to protect itself:**

```
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

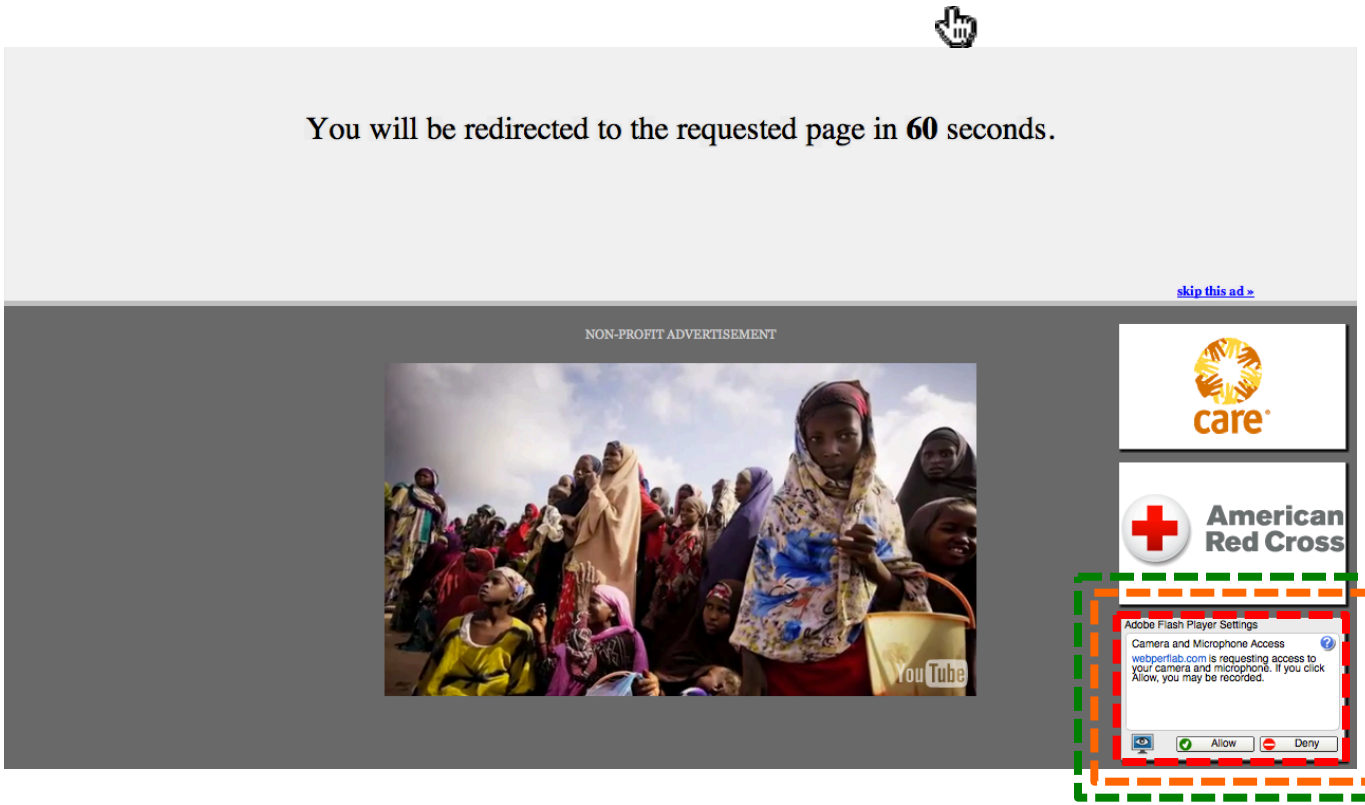
# Defense: Ensuring visual integrity of pointer

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



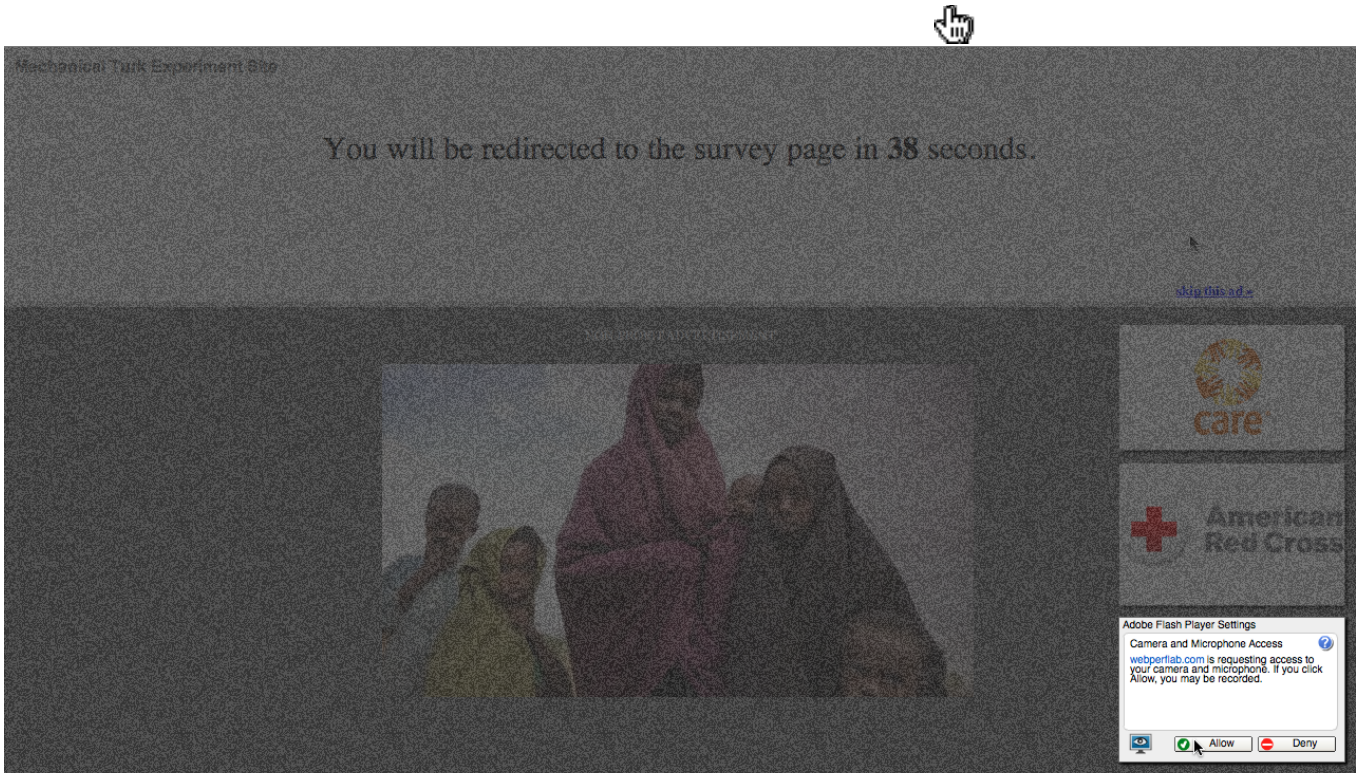
# Ensuring visual integrity of pointer

- 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%)



# Ensuring visual integrity of pointer

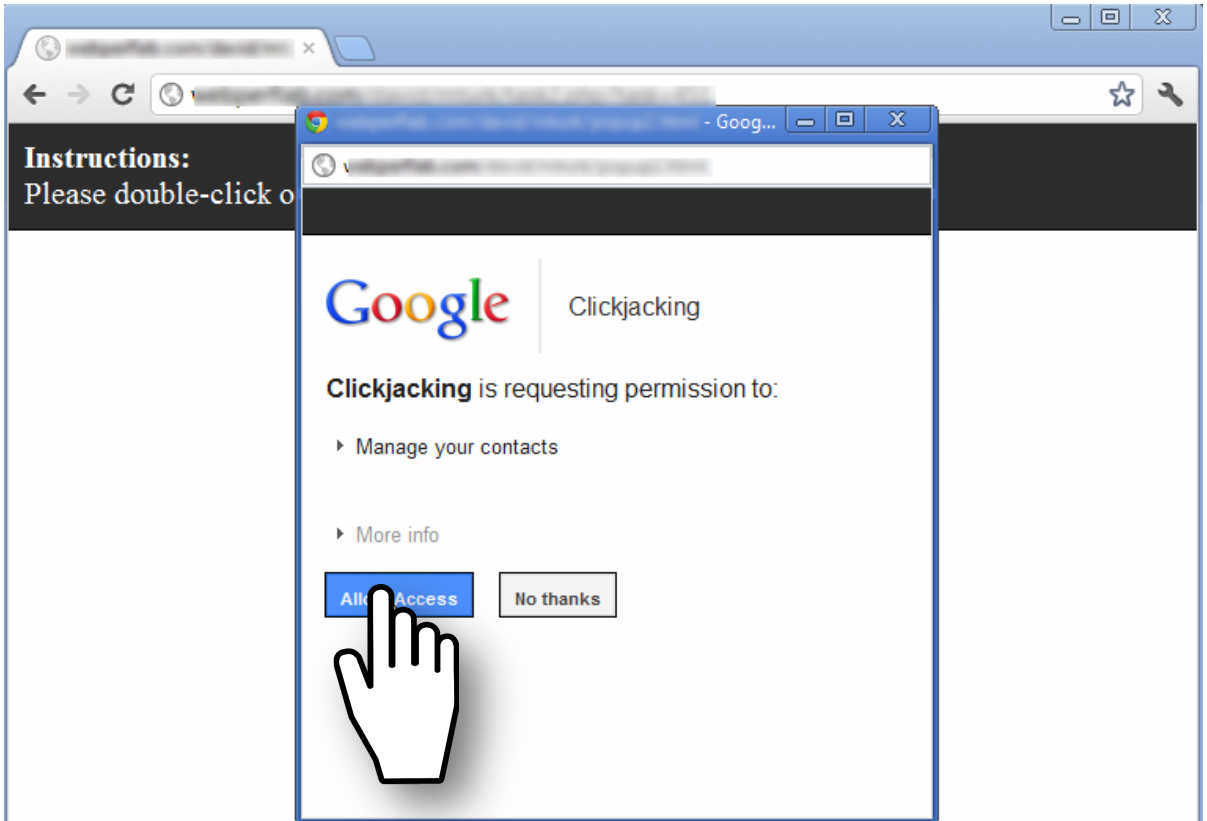
- Lightbox effect around target on pointer entry
  - Attack success (Freezing + lightbox): 2%



# How about a temporal integrity attack example?

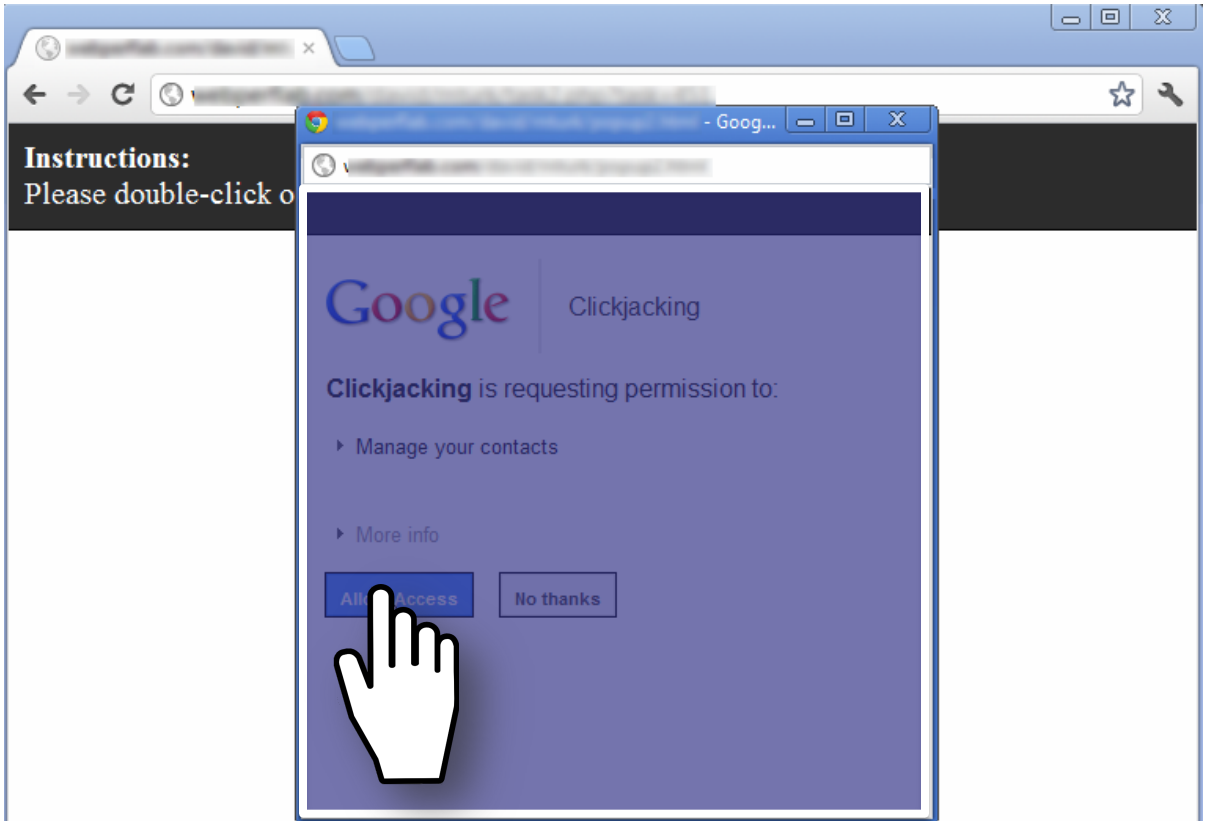
# Temporal clickjacking

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



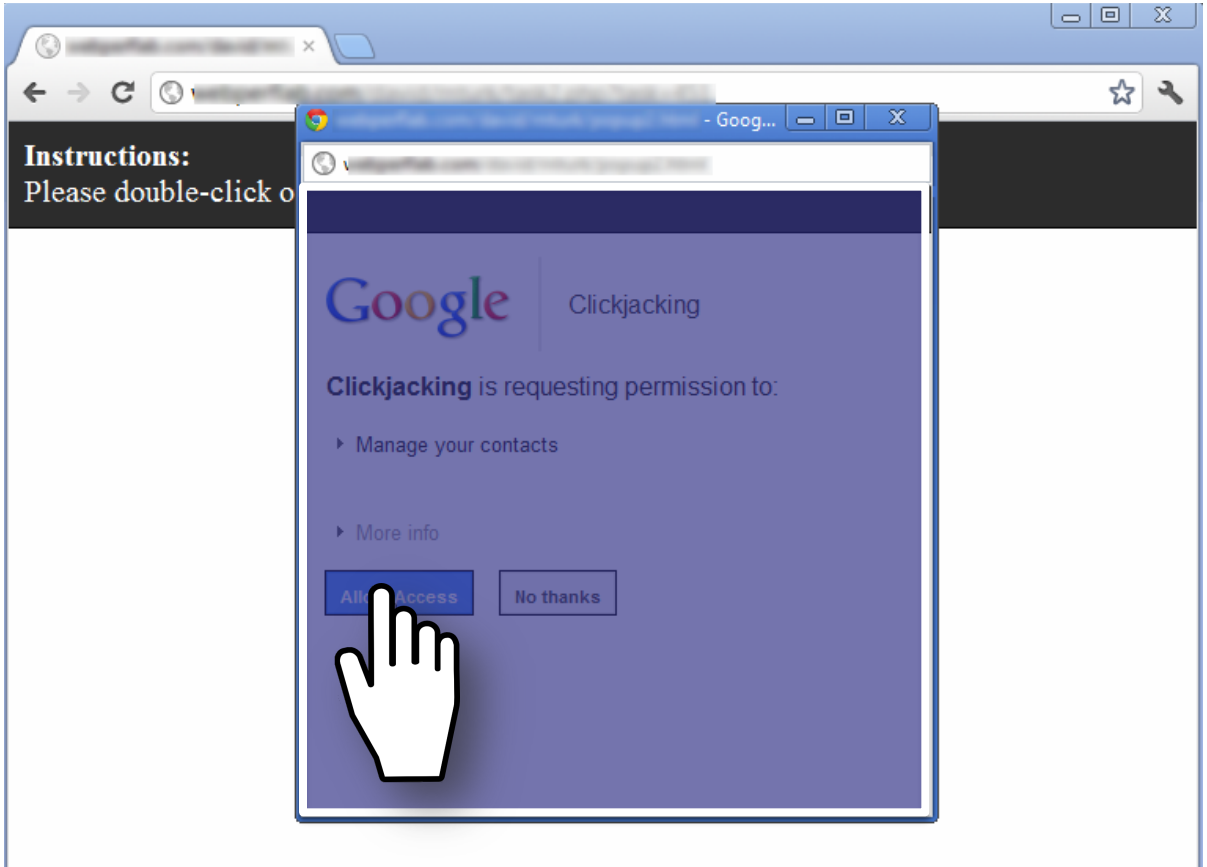
# Enforcing temporal integrity

- 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)

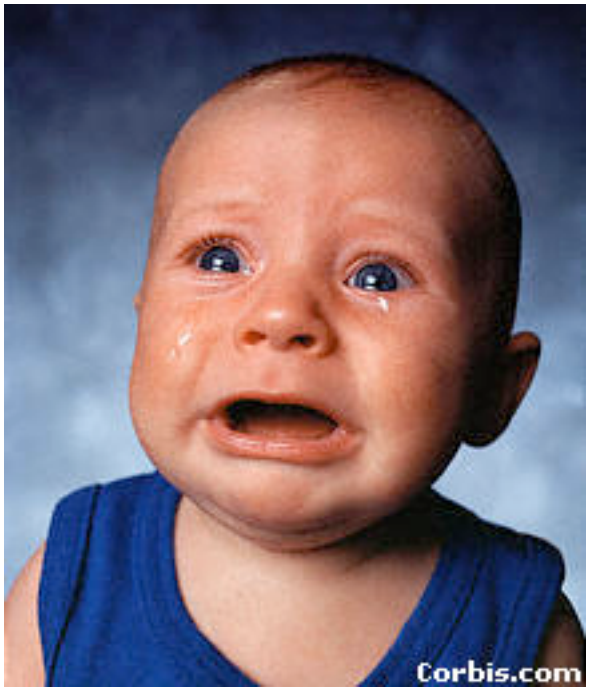


# Enforcing temporal integrity

- 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

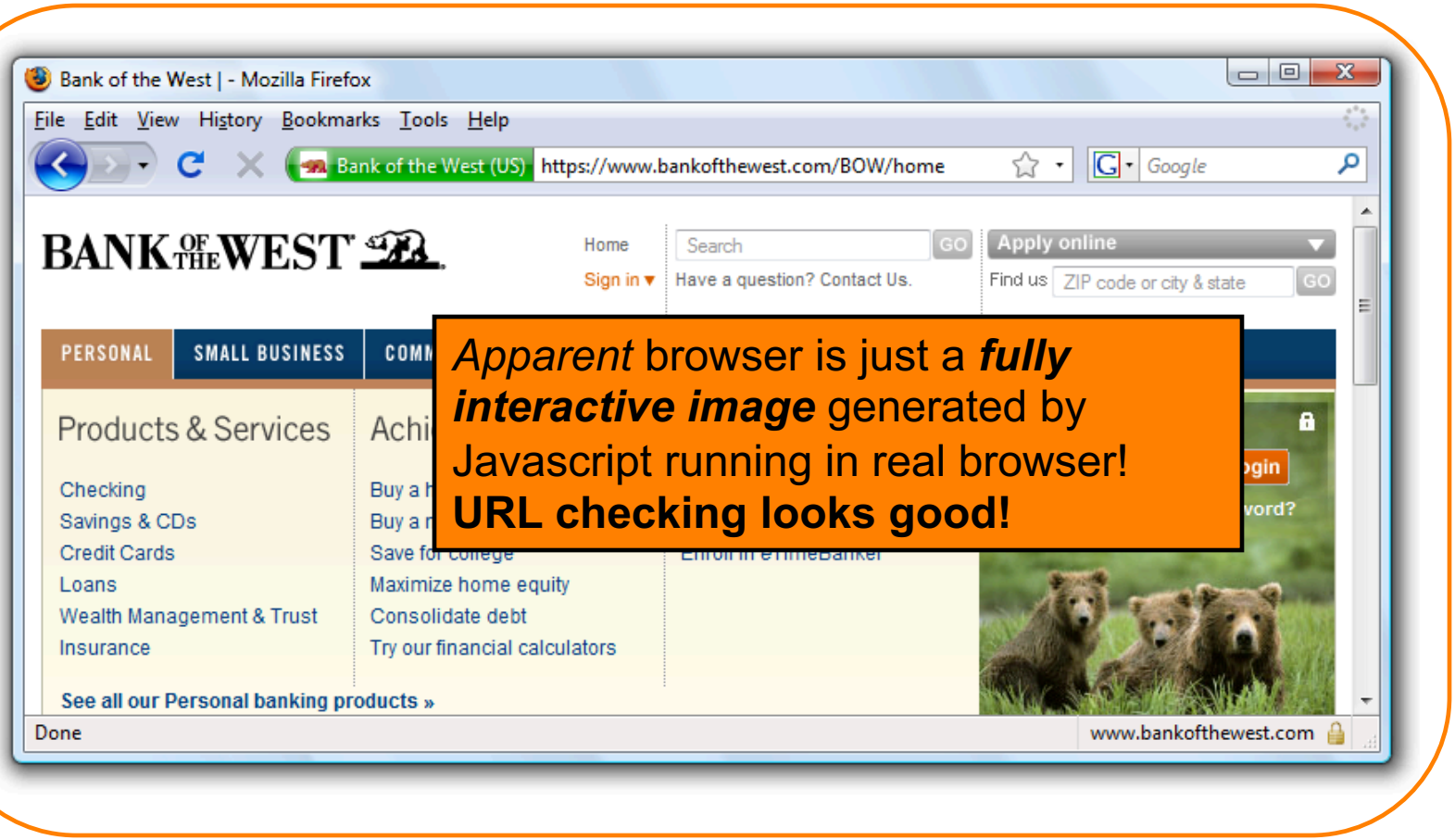
- Good defense … but poor adoption by sites (4 of top 10,000)

- 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