

CSE 484 / CSE M 584
Computer Security:
Lab 2 & Click Jacking

TA: Adrian Sham
adrsham@cs

Thanks to Franzi & Vitaly Shmatikov for Clickjacking slides

Logistics / Reminders

- Submit account info for Lab #2 by 5pm **today**.
 - Link: <https://catalyst.uw.edu/webq/survey/neyp/270183>
- Homework #2 due **tomorrow** (5pm).
- Next office hour:
 - Michael and Adrian: 9:30-10:30am, CSE 218
- Lab #2: Web security
 - Should be out tomorrow / earlier

XSS review

- Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications.
- Allows the attacker to inject JavaScript into web pages viewed by other users.
- JavaScript can do a lot of things, like reading cookies and ex-filtrating them.
- Solutions
 - Sanitize/validate your input
 - Browser detection

XSSI: Cross-Site Script Inclusion

- **Idea:** **Include scripts** (e.g., libraries) to run in context of current domain.

Example:

```
<head> <script  
src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jq  
uery.min.js"></script> </head>
```

- **Threat:** Attacker provides malicious library, can execute code **in your domain's context**.
- **Solution:** Make sure included code comes from **trusted site**.

Same-Origin Policy

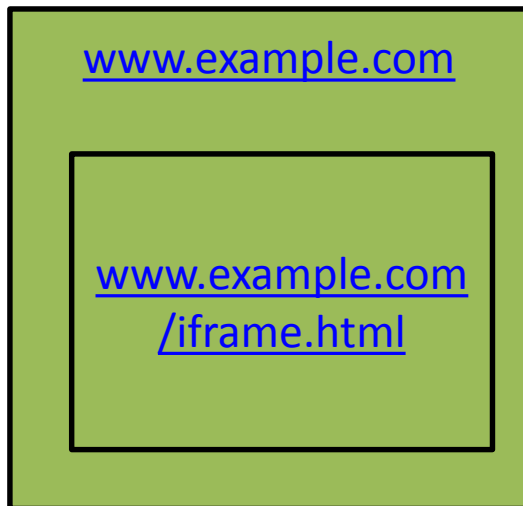
Website origin = (scheme, domain, port)

| Compared URL | Outcome | Reason |
|---|---------|---|
| http://www.example.com/dir/page.html | Success | Same protocol and host |
| http://www.example.com/dir2/other.html | Success | Same protocol and host |
| http://www.example.com:81/dir/other.html | Failure | Same protocol and host but different port |
| https://www.example.com/dir/other.html | Failure | Different protocol |
| http://en.example.com/dir/other.html | Failure | Different host |
| http://example.com/dir/other.html | Failure | Different host (exact match required) |
| http://v2.www.example.com/dir/other.html | Failure | Different host (exact match required) |

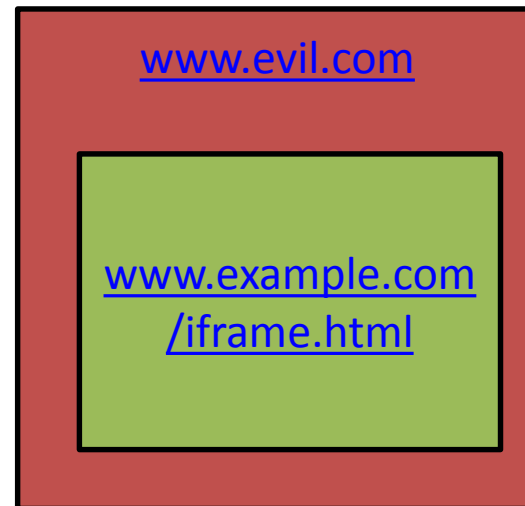
[Example thanks to Wikipedia.]

Same-Origin Policy (DOM)

- Only code from same origin can **access HTML elements** on another site (or in an iframe).



www.example.com (the parent) **can** access HTML elements in the iframe (and vice versa).



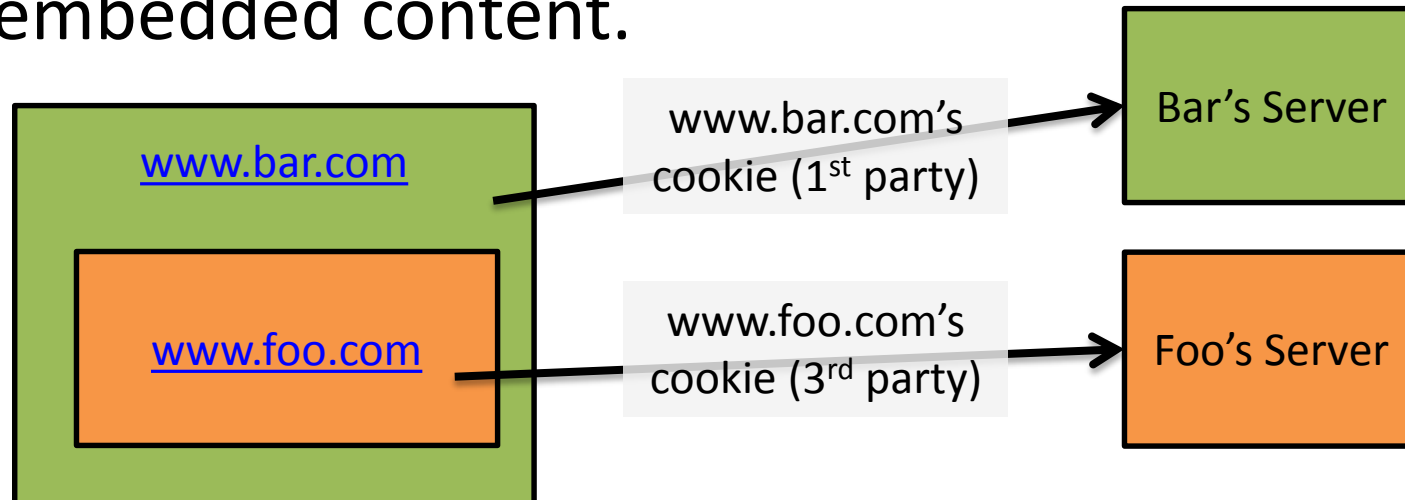
www.evil.com (the parent) **cannot** access HTML elements in the iframe (and vice versa).

Same-Origin Policy (Cookies)

- **For cookies:** Only code from same origin can **read/write cookies** associated with an origin.
 - Can be set via Javascript (`document.cookie=...`) or via `Set-Cookie` header in HTTP response.
 - Can narrow to subdomain/path (e.g., <http://example.com> can set cookie scoped to <http://account.example.com/login>.)
 - **Secure cookie:** send only via HTTPS.
 - **HttpOnly cookie:** can't access using JavaScript.

Same-Origin Policy (Cookies)

- Browsers **automatically include cookies** with HTTP requests.
- **First-party cookie:** belongs to top-level domain.
- **Third-party cookie:** belongs to domain of embedded content.



Same-Origin Policy (Scripts)

- When a website **includes a script**, that script runs in the context of the embedding website.

```
www.example.com  
  
<head>  
<script  
  src="http://otherdomain.com/library.js"></script> </head>
```

The code from <http://otherdomain.com> **can** access HTML elements and cookies on www.example.com.

- If code in the script sets a cookie, under what origin will it be set?

PHP review

- A **server**-side programming language
- File extension is .php
- Before a webpage is sent to you, PHP code is executed by the server
- You won't see the PHP code, only html
- PHP can be use to set and read cookies for authentication
- You will need a basic PHP script to receive captured cookies

Quick demo of XSS

Back story to Lab #2

- You finally decide to show your click-happy Computer Security TAs who's boss.
- Use XSS attacks to steal your TA's cookies, and therefore access your gradebook to change your grade.
- Use a SQL Injection to add yourself to Franzzi's good list.

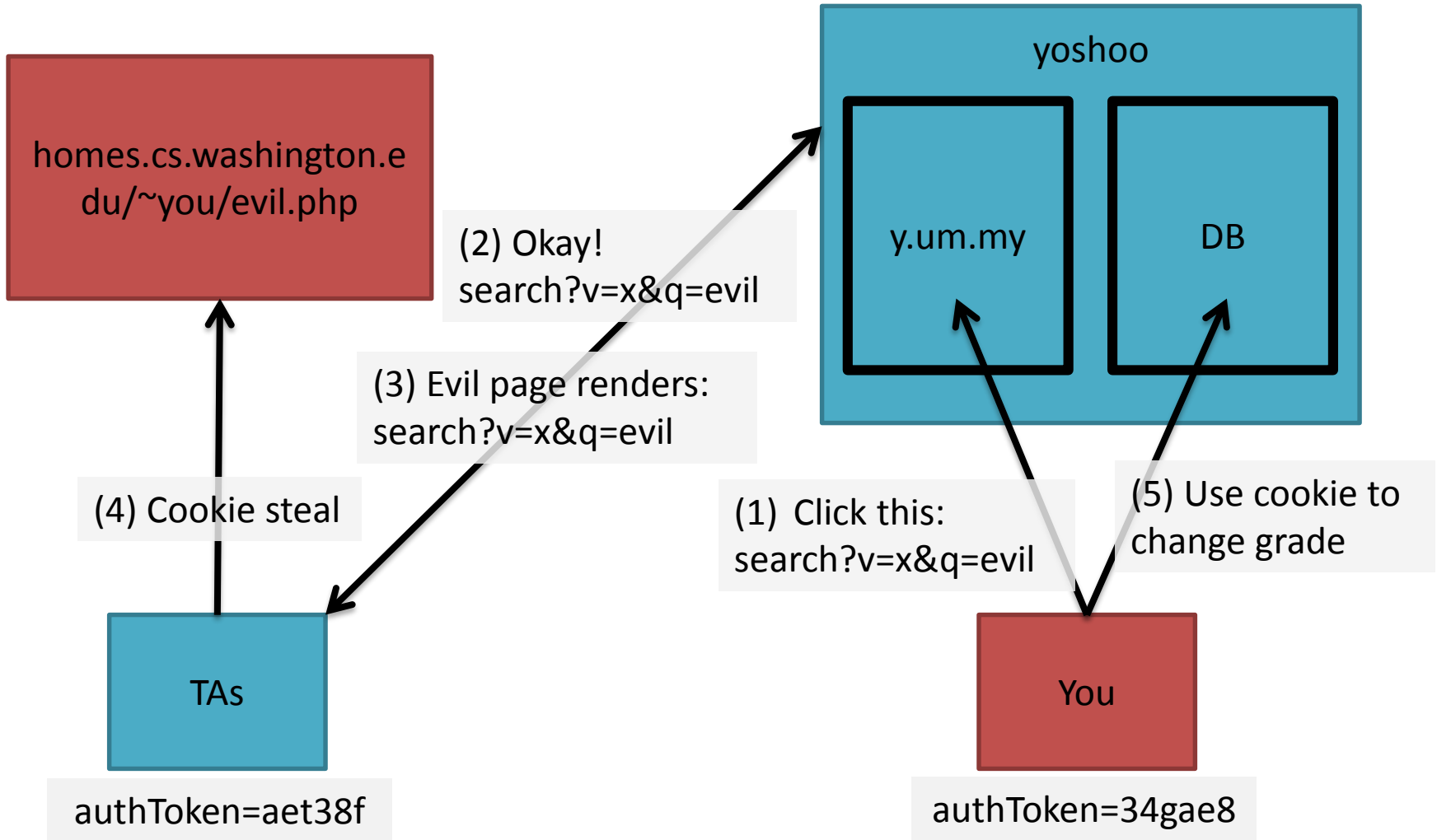
Basic setup

- Give the TAs (y.um.my) a link with a XSS vulnerability.
- TAs will 'visit' this link, and their cookie will be stolen.
- The process of stealing cookie involves sending it to a place you control.
- Save the cookie, read it, and use it to log in and change your grade.
- Easy!

What you will need

- [Firefox](#), latest version should be OK
 - Chrome **might** probably won't work
- [Firebug](#) add-on for Firefox
- Setup a location to collect your ~~stolen~~ liberated cookies
 - Good place is homes.cs, FAQ [here](#)
- URL encoder (converts characters into a format for transmission over the internet)
 - Such as [this](#)

Lab #2 Explained



Tips

- Be mindful of Same Origin Policy
 - Don't redirect
- Run JavaScript locally before sending to y.um.my
- When URL encoding, be careful of new-lines in XSS
 - Browser might stop executing at newline
- Talk to us if something feels wrong / confusing

SQL Injection

- SQL Injection allows the attacker to insert malicious SQL statements
- Usually caused by incorrect filtering of user input

SQL Injection Lab #2

- Franzini keeps a list of students she really *really* likes
- Use SQL Injection to inject your username into the list
- Hint: There are different syntax(s) for inserting into a table, try all of them

Click Jacking

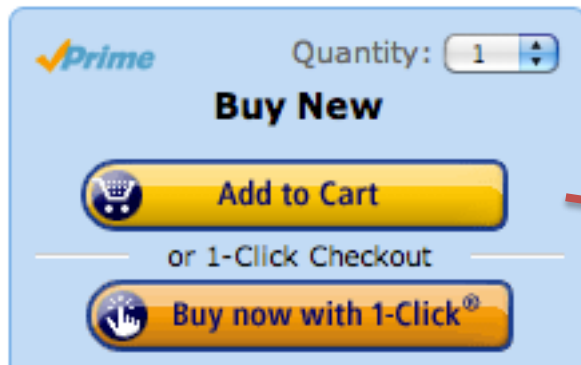
- Clickjacking happens when an attacker uses different techniques to hijack clicks meant for their page and routing them to another
- Multiple techniques
 - Transparent UI elements on top of a button or link
 - Timing based attacks

Example

- Video of click jacking
- https://www.youtube.com/watch?v=9V4_emKyAg8
- User is asked to play a game
- Button is quickly switched to a 'save' button

Clickjacking

- **Trick users** into interacting with sensitive user interfaces in another domain.
 - Using invisible iframes:



Clickjacking using the Cursor



Figure 1: Cursor spoofing attack page. The target Flash Player webcam settings dialog is at the bottom right of the page, with a “skip this ad” bait link remotely above it. Note there are two cursors displayed on the page: a fake cursor is drawn over the “skip this ad” link while the actual pointer hovers over the webcam access “Allow” button.

- Following slides by Vitaly Shmatikov
- <http://www.cs.utexas.edu/~shmat/courses/cs361s/clickjack.ppt>

Clickjacking (UI Redressing)

[Hansen and Grossman 2008]

- Attacker overlays multiple transparent or opaque frames to trick a user into clicking on a button or link on another page



- Clicks meant for the visible page are hijacked and routed to another, invisible page

Clickjacking in the Wild

- Google search for “clickjacking” returns 624,000 results... this is not a hypothetical threat!
- Summer 2010: Facebook worm superimposes an invisible iframe over the entire page that links back to the victim's Facebook page
 - If victim is logged in, automatically recommends link to new friends as soon as the page is clicked on
- Many clickjacking attacks against Twitter
 - Users send out tweets against their will

It's All About iFrame

- Any site can frame any other site

```
<iframe
```

```
  src="http://www.google.com/...">
```

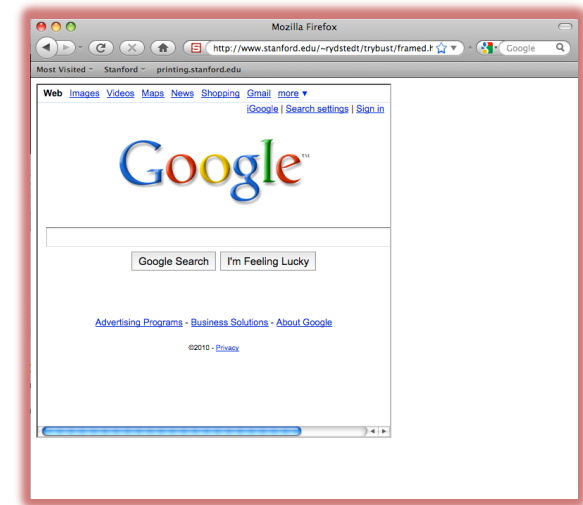
```
</iframe>
```

- HTML attributes

- Style

- **Opacity** defines visibility percentage of the iframe

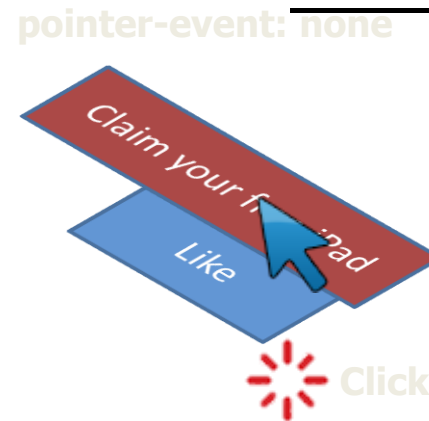
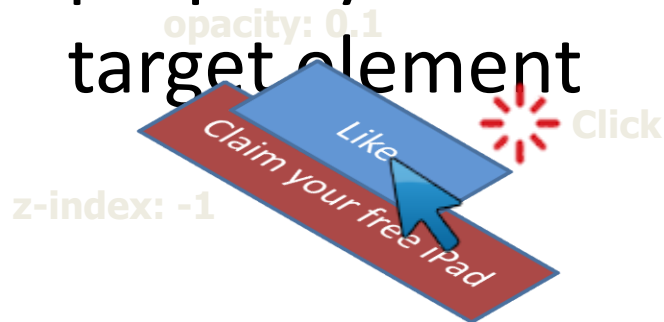
- 1.0: completely visible
- 0.0: completely invisible



Hiding the Target Element

[“Clickjacking: Attacks and Defenses”]

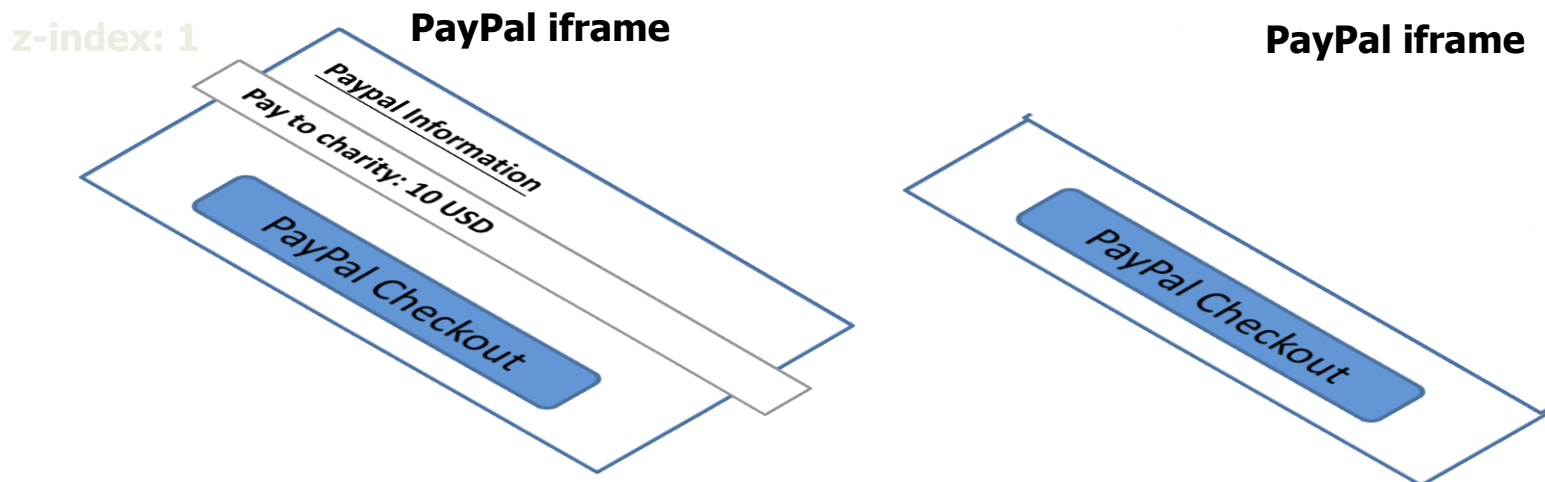
- Use CSS `opacity` property and `z-index` property to hide target element and make other element float under the target element
- Using CSS `pointer-events: none` property to cover other element over the target element



Partial Overlays and Cropping

[“Clickjacking: Attacks and Defenses”]

- Overlay other elements onto an iframe using CSS `z-index` property or Flash Window Mode `wmode=direct` property
- Wrap target element in a new iframe and choose CSS position offset properties



Drag-and-Drop API

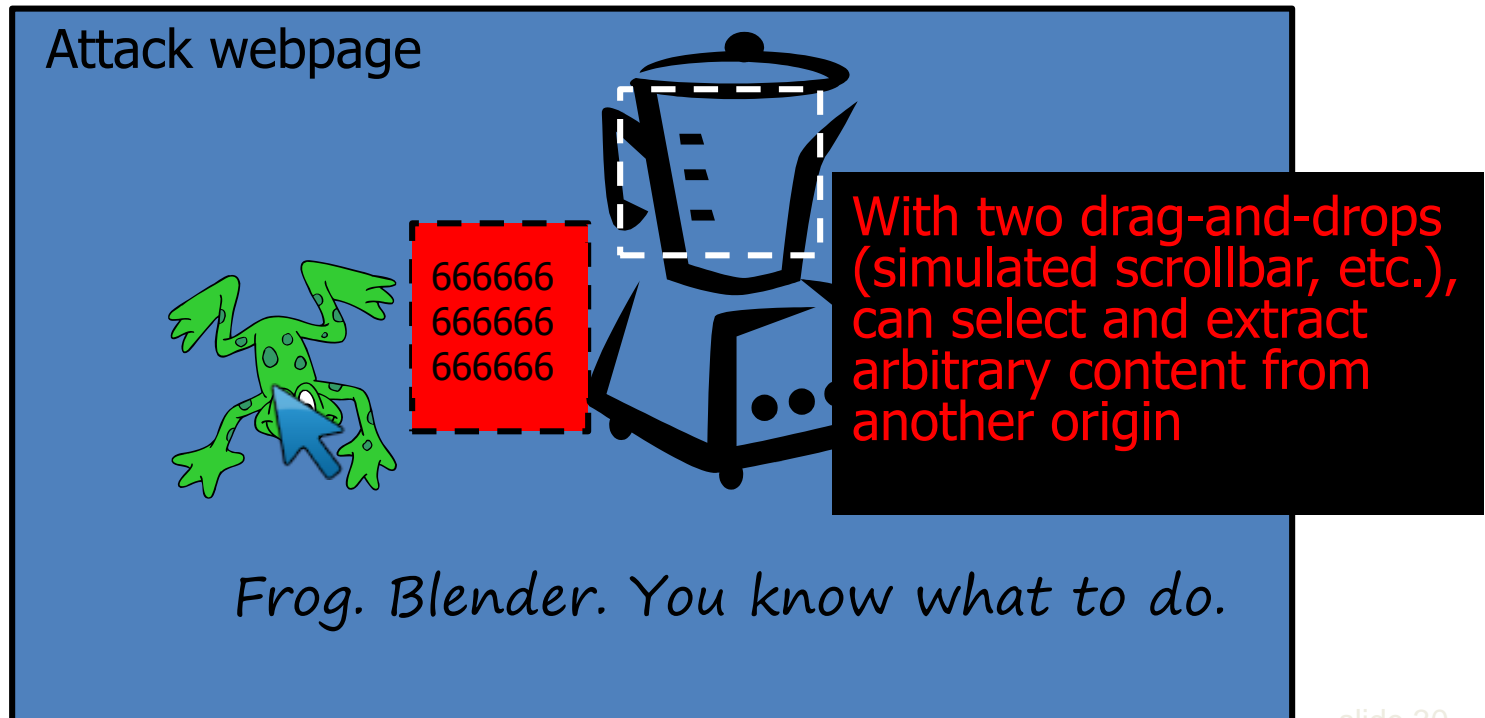
[“Next Generation Clickjacking”]

- Modern browsers support drag-and-drop API
- JavaScript can use it to set data being dragged and read it when it's dropped
- Not restricted by the same origin policy: data from one origin can be dragged to a frame of another origin
 - Reason: drag-and-drop can only be initiated by user's mouse gesture, not by JavaScript on its own

Abusing Drag-and-Drop API

[“Next Generation Clickjacking”]

1. Bait the user to click and start dragging
2. Invisible iframe with attacker’s text field under mouse cursor, use API to set data being dragged
3. Invisible iframe from another origin with a form field



Fake Cursors

[“Clickjacking: Attacks and Defenses”]

- Use CSS `cursor` property and JavaScript to simulate a fake cursor icon on the screen

Real cursor icon

`cursor: none`



Fake cursor icon



Keyboard “Strokejacking”

[“Clickjacking: Attacks and Defenses”]

- Simulate an input field getting focus, but actually the keyboard focus is on target element, forcing user to type some unwanted information into target element

Attacker’s page

Typing Game
Type whatever screen shows to you


Xfpog95403poigr06=2kfpX

[]



Hidden iframe within attacker’s page

Bank Transfer
Bank Account: 9540
Amount: 3062 USD



Solution: Frame Busting

- I am a page owner
- All I need to do is make sure that my web page is not loaded in an enclosing frame ...

Clickjacking: solved!

– Does not work for FB “Like” buttons and such, but

Ok

- How

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

Frame Busting in the Wild

- Survey by Gustav Rydstedt, Elie Burzstein, Dan Boneh, Collin Jackson



Following slides shamelessly jacked from Rydstedt