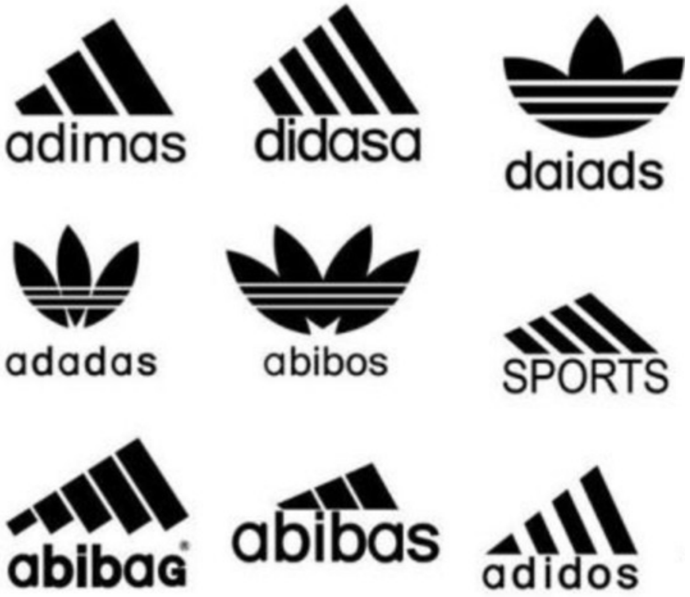
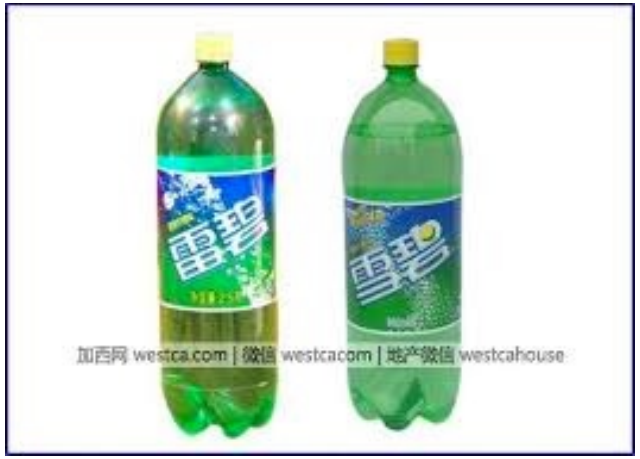
CS5331: Web Security

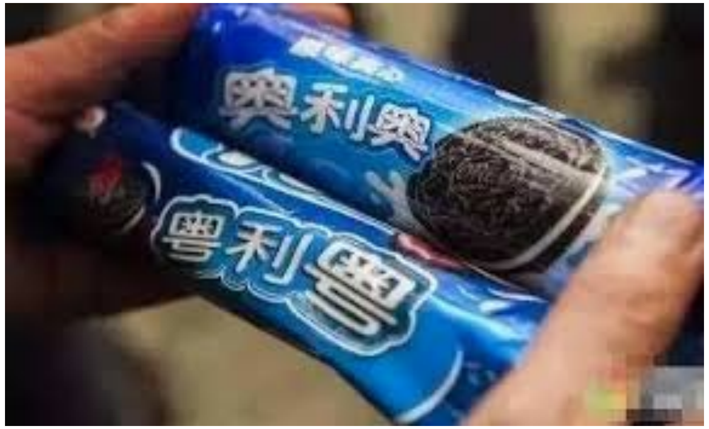
Lecture 8: UI Attacks and Password attacks

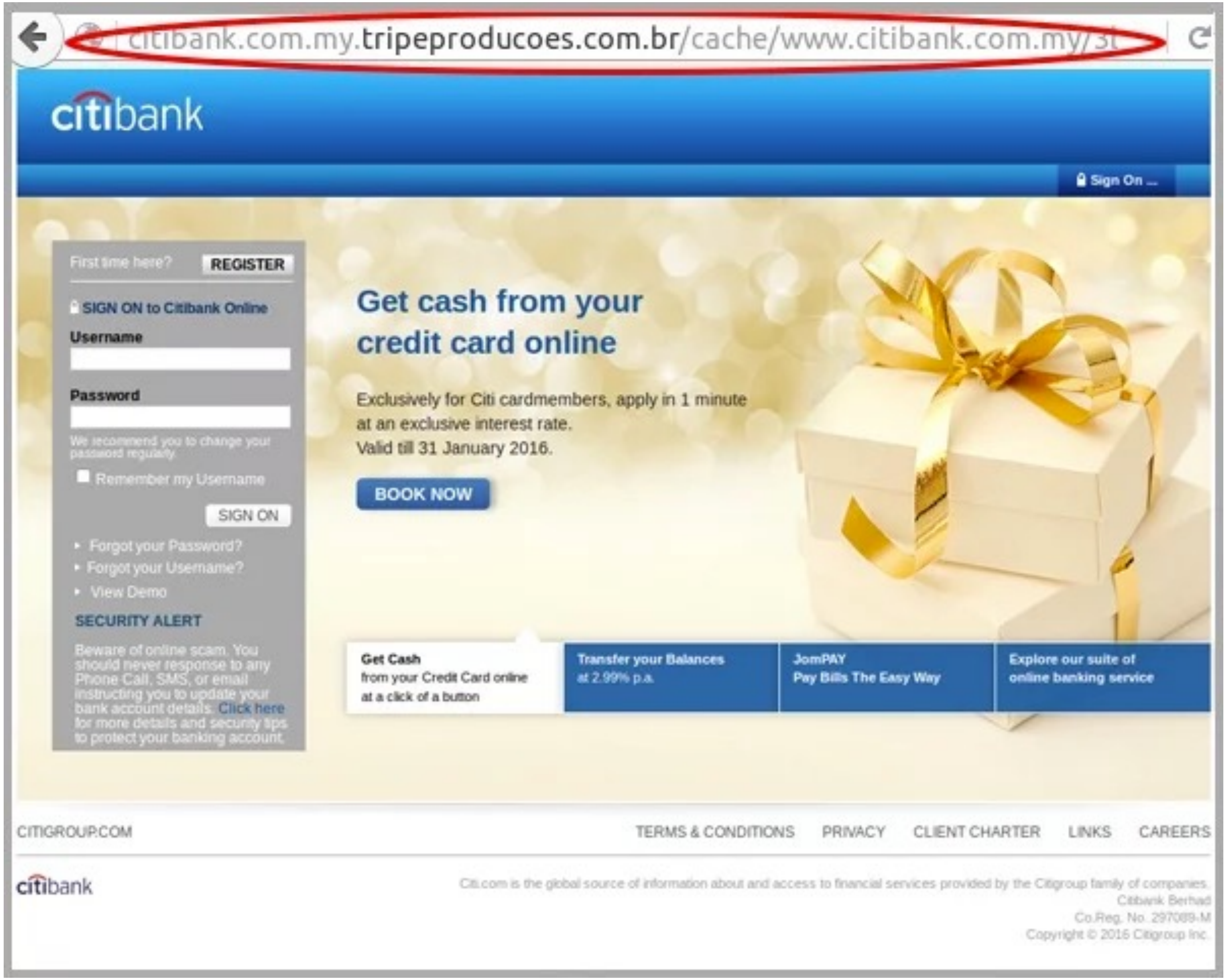
Web UI Attacks

adidas
adidas


adimas
didasa
daiads
adadas
abibos
SPORTS
abibaG
abibas
adidos


雷碧
雪碧
加西网 westca.com | 微信 westcacom | 地产微信 westcahouse


奥利奥
粤利粤



First time here?

SIGN ON to Citibank Online

We recommend you to change your password regularly.

□ Remember my Username

- Forgot your Password?
- Forgot your Username?
- View Demo

SECURITY ALERT

Beware of online scam. You should never response to any Phone Call, SMS, or email instructing you to update your bank account details. Click here for more details and security tips to protect your banking account.

Get cash from your credit card online

Exclusively for Citi cardmembers, apply in 1 minute at an exclusive interest rate.
Valid till 31 January 2016.

Get Cash from your Credit Card online at a click of a button

Transfer your Balances at 2.99% p.a.

JomPAY Pay Bills The Easy Way

Explore our suite of online banking service

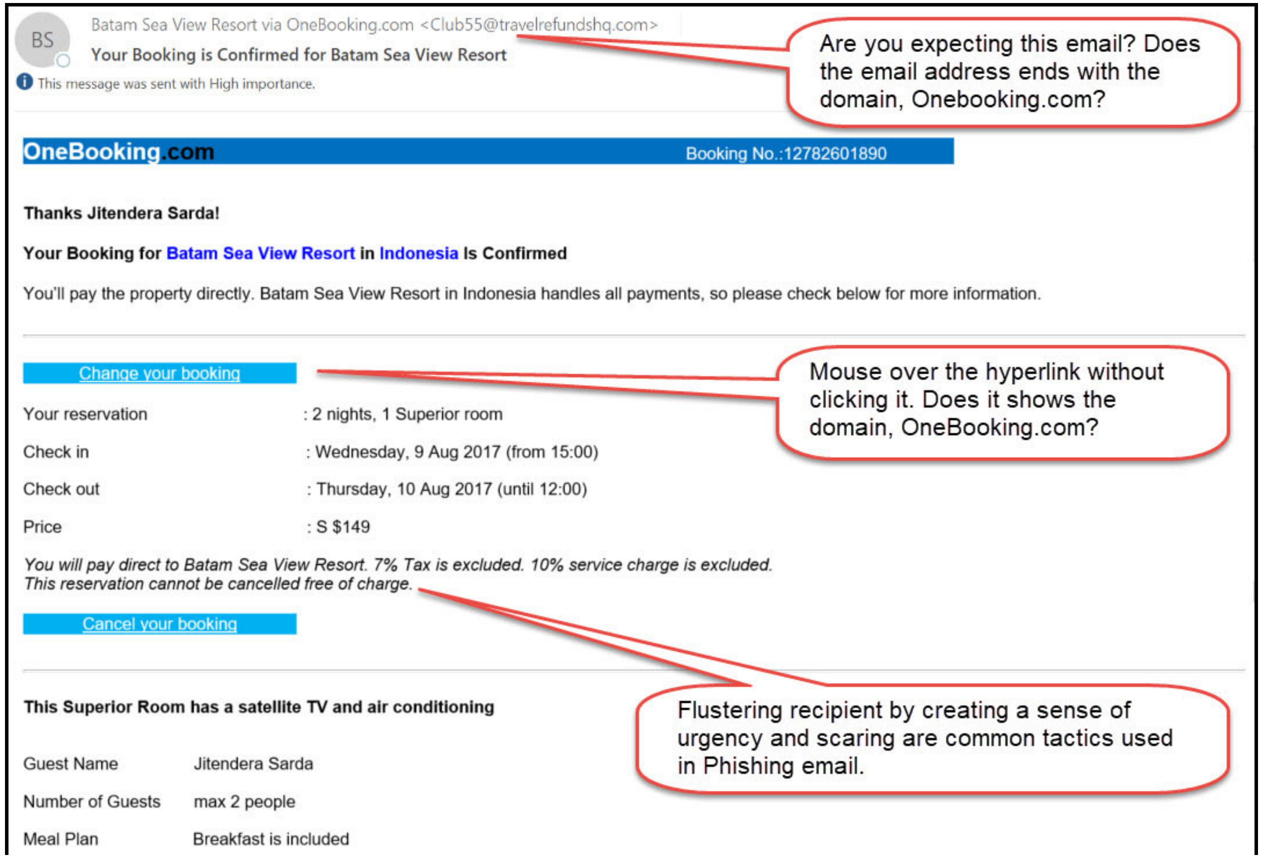
Citi.com is the global source of information about and access to financial services provided by the Citigroup family of companies.
Citibank Berhad
Co.Reg. No. 297089-M
Copyright © 2016 Citigroup Inc.

Some Phishing Attacks

- Homograph attack:
  - A phishing web page with a domain name that is only *a few pixels different* from the legitimate site's domain name
- Picture-in-picture attack:
  - A phishing web page that shows a *fake browser window* that appears to be *showing the real site*
- Hostname-and-path boundary confusion:
  - A phishing web page with hostname that contains the targeted hostname followed by a character resembling "/"
  - Occurs if there is no clear distinction between hostname and path in URL bar: some browsers have addressed this!
- Typosquatting

Preventing Phishing

- User education: phishing drill



Batam Sea View Resort via OneBooking.com <Club55@travelrefundshq.com>

Your Booking is Confirmed for Batam Sea View Resort

This message was sent with High importance.

OneBooking.com Booking No.:12782601890

Thanks Jitendera Sarda!

Your Booking for Batam Sea View Resort in Indonesia Is Confirmed

You'll pay the property directly. Batam Sea View Resort in Indonesia handles all payments, so please check below for more information.

Change your booking

Your reservation	: 2 nights, 1 Superior room
Check in	: Wednesday, 9 Aug 2017 (from 15:00)
Check out	: Thursday, 10 Aug 2017 (until 12:00)
Price	: S $149

You will pay direct to Batam Sea View Resort. 7% Tax is excluded. 10% service charge is excluded.
This reservation cannot be cancelled free of charge.

Cancel your booking

This Superior Room has a satellite TV and air conditioning

Guest Name	Jitendera Sarda
Number of Guests	max 2 people
Meal Plan	Breakfast is included

Are you expecting this email? Does the email address ends with the domain, Onebooking.com?

Mouse over the hyperlink without clicking it. Does it shows the domain, OneBooking.com?

Flustering recipient by creating a sense of urgency and scaring are common tactics used in Phishing email.

From: NUS IT Care

User Education

We need to verify your credit card number.
Don't take the bait. Protect your financial information.

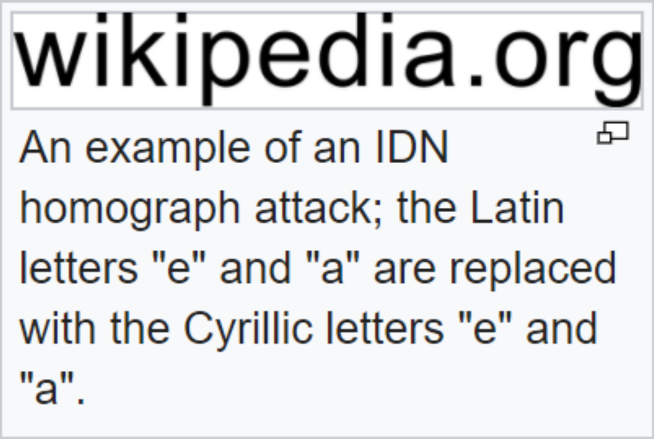
From: U.S. Federal Trade Commission

Preventing Phishing

- Phishing repository site:
  - Example: phishtank.com (submit suspected phishes, track the status of your submissions,
    verify other users' submissions)
- However, it can be tricky to accurately determine if an unsolicited email is a phishing
  - Example: SonicWall Phishing IQ Test
    https://www.sonicwall.com/phishing/
  - *You can test your own phishing-spotting skill!*

Some Browser Defenses

- The *internationalized domain name (IDN) homograph* attack:
  - Don’t support IDN sites? Usability issue!
  - Block IDNs that mix scripts for different languages?
  - Block TLD from choosing letters that could resemble an existing Latin TLD



wikipedia.org

An example of an IDN homograph attack; the Latin letters "e" and "a" are replaced with the Cyrillic letters "e" and "a".

Some Browser Defenses

- Display IDNs in punycode in the URL bar:
  - Punycode: representation of Unicode with the limited ASCII character subset consisting of letters, digits, and hyphen (Letter-Digit-Hyphen/LDH subset)
  - bücher.tld → xn--bcher-kva.tld
  - See https://en.wikipedia.org/wiki/Punycode
- Blacklisting phishing URLs:
  - https://en.wikipedia.org/wiki/Google_Safe_Browsing
  - Is it scalable?
  - How about short-lived URLs?
  - Possibility of filter evasion?

Defenses Against Phishing Attacks?

- Attacks on “human perception”
- Human can be the weakest link in the system
- Still a big security problem
- *Any good defenses?*
- Let’s have a discussion…!

Clickjacking

Background: Iframe and Screen Access

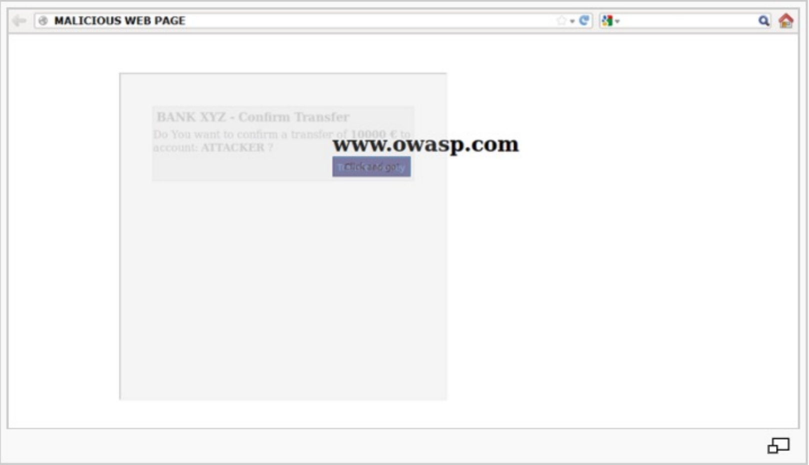
- Pages can embed iframes from 3rd-party
  - Any site can host another in <iframe>:

```
<iframe id="newframe" src="http://www.cnn.com" style="opacity:0.0;
position:absolute; top:195px; left:10px; width:1000px;
height:200px;"></iframe>
```

- “Pixel delegation” policy (To double-check):
  - An embedded iframe:
    - Can draw pixels within its iframe’s box: Yes
    - Can draw pixels outside of its iframe’s box: No
  - The embedding iframe/page:
    - Can draw pixels within the embeded iframe’s box: Yes
  - Child/descendant policy?

Background: Iframe and Screen Access

- Frames can overlap:
    - Outer iframe/page can create *another overlapping iframe* positioned over an embeded iframe
    - Sample page: https://www.owasp.org/index.php/Testing_for_Clickjacking_(OTG-CLIENT-009)



BANK XYZ - Confirm Transfer

Do You want to confirm a transfer of 10000 € to account: ATTACKER ?

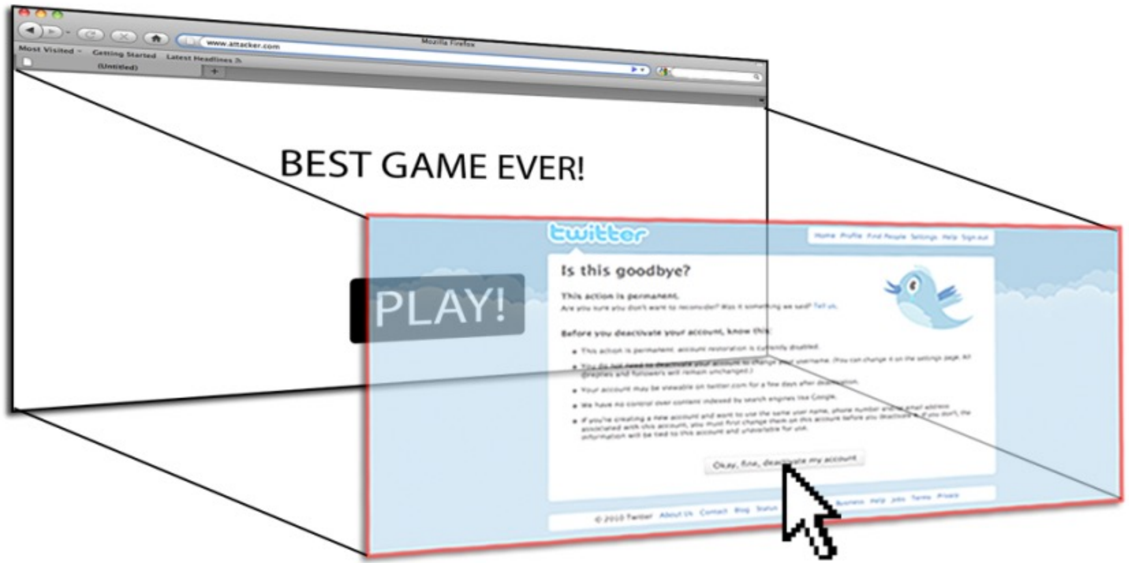
- CSS controls the transparency, location of frames

Background: Iframe and Screen Access

- Some CSS features are useful for clickjacking
- Properties of iframe’s style attribute:
  - Positioning properties: `position, top, left`
  - `opacity:`
    - Defines visibility percentage of the iframe
    - Value 1.0: completely visible
    - Value 0.0: *completely invisible*
  - `z-index:`
    - Specifies the *stack order* of a positioned element
    - An element with greater stack order (1) is always in front of another element with lower stack order (0)
- `pointer-events: none`
  - The element is never the target of mouse events

Clickjacking Techniques

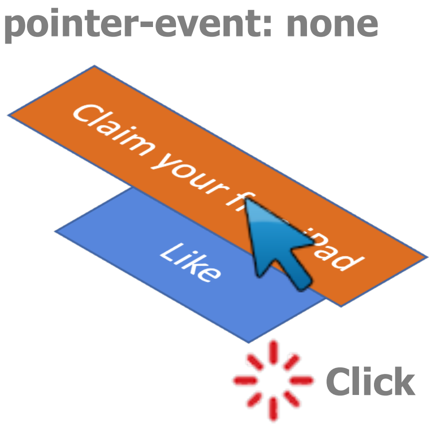
- How to trick users?
- One technique: *hide the target element*
- Mechanism #1: use CSS `opacity` and `z-index` properties to hide an invisible UI-target element *over* a visible (enticing) UI-bait element
- Result: can fool a user into taking unintended action on the invisible (damaging) UI-target element



Clickjacking – Hansen, Grossman 2008

Clickjacking Techniques

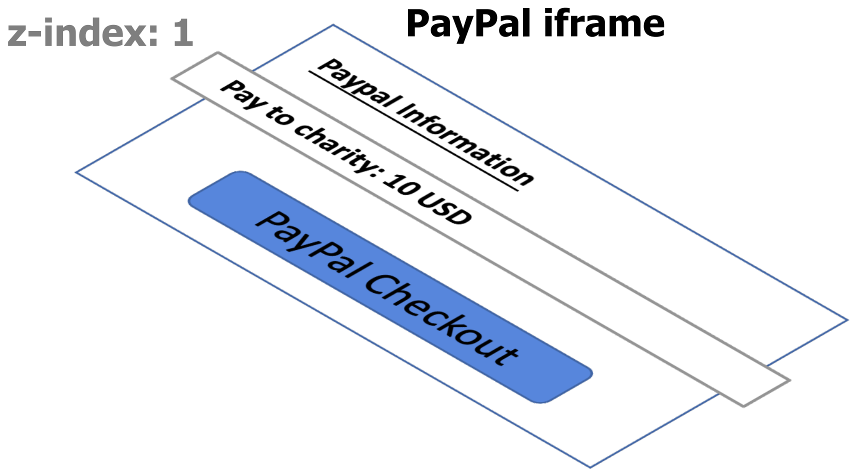
- Mechanism #2: use `pointer-events: none` to hide/cover an invisible UI-target element *under* a visible (enticing) UI-bait element
- Result: can fool a user into taking unintended action on the invisible (damaging) UI-target element
- An example:



Clickjacking – Hansen, Grossman 2008

Clickjacking Techniques

- Other technique: partial overlays
  - Obscure *only a part of* the target element
  - Example: cover Paypal's recipient or amount field only, and leave the "Pay" button intact



Clickjacking Attacks & Defenses – Huang et al.

Clickjacking Techniques

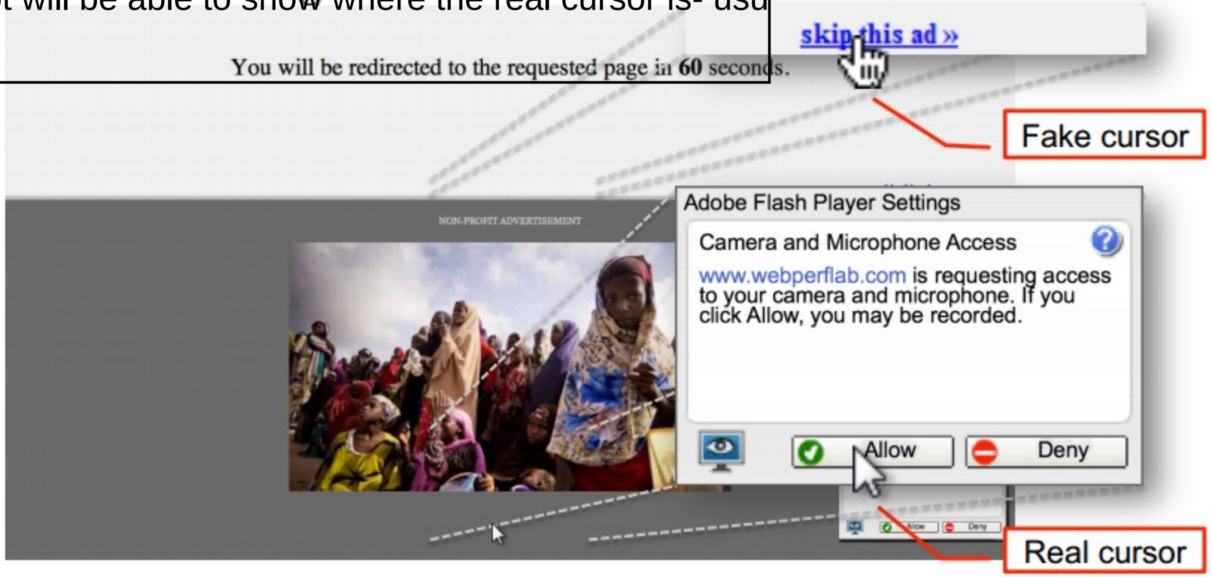
- What have we seen so far?
  - Clickjacking attacks targeting *target display integrity*
- Can we attack others?
  - Yes, we can attack *mouse pointer integrity*
- **Cursor-jacking**:
  - Display *a fake cursor icon* away from the actual pointer

Clickjacking Attacks & Defenses – Huang et al.

Cursor-Jacking

- Cursor-jacking
  - E.g. using CSS `cursor: url(x.gif)`

successful but any different OS interrupt will be able to show where the real cursor is- usu



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

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.

Attack technique: cursor-spoofing

Attack success: 43% (31/72)

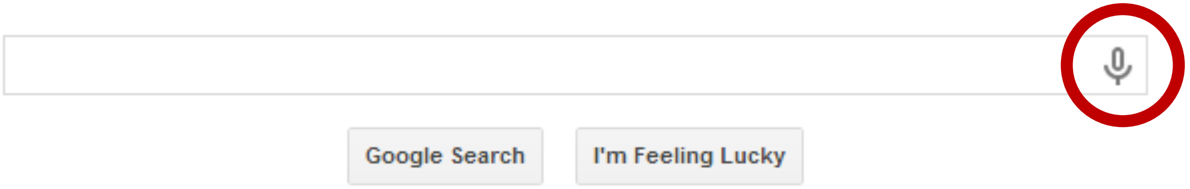
13

Live examples:

http://koto.github.io/blog-kotowicz-net-examples/cursorjacking/

Clickjacking
(UI Redressing Attacks)

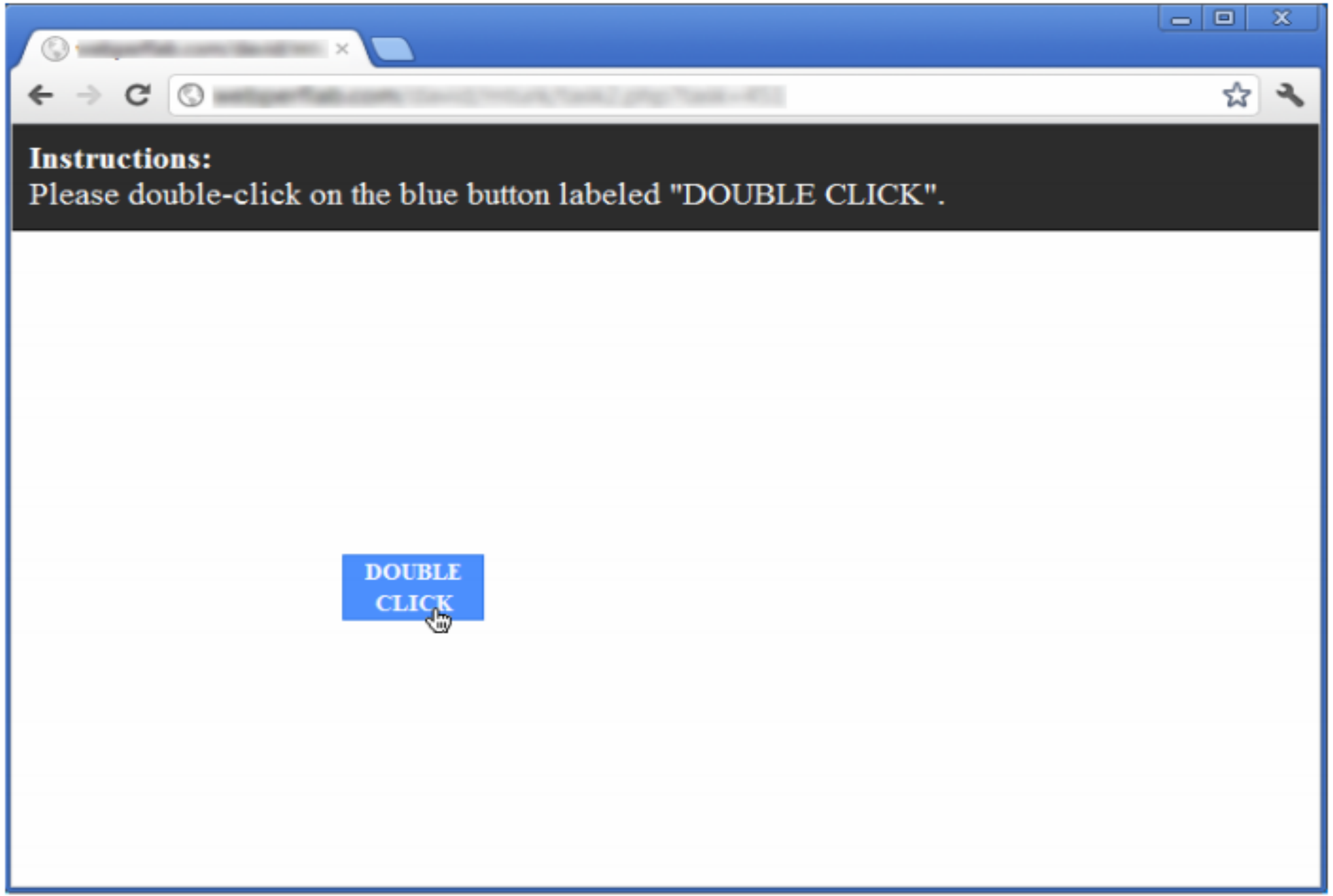
- *Clickjacking*: one principal may trick the user into interacting with (e.g., clicking, touching, or voice controlling) UI elements of another principal, triggering actions not intended by the user
- Variations:
  - Likejacking
  - Camjacking
  - Tapjacking



  - Micjacking?

Timing Attack: Bait-and-Switch

- User is asked to *double-click* the blue button:

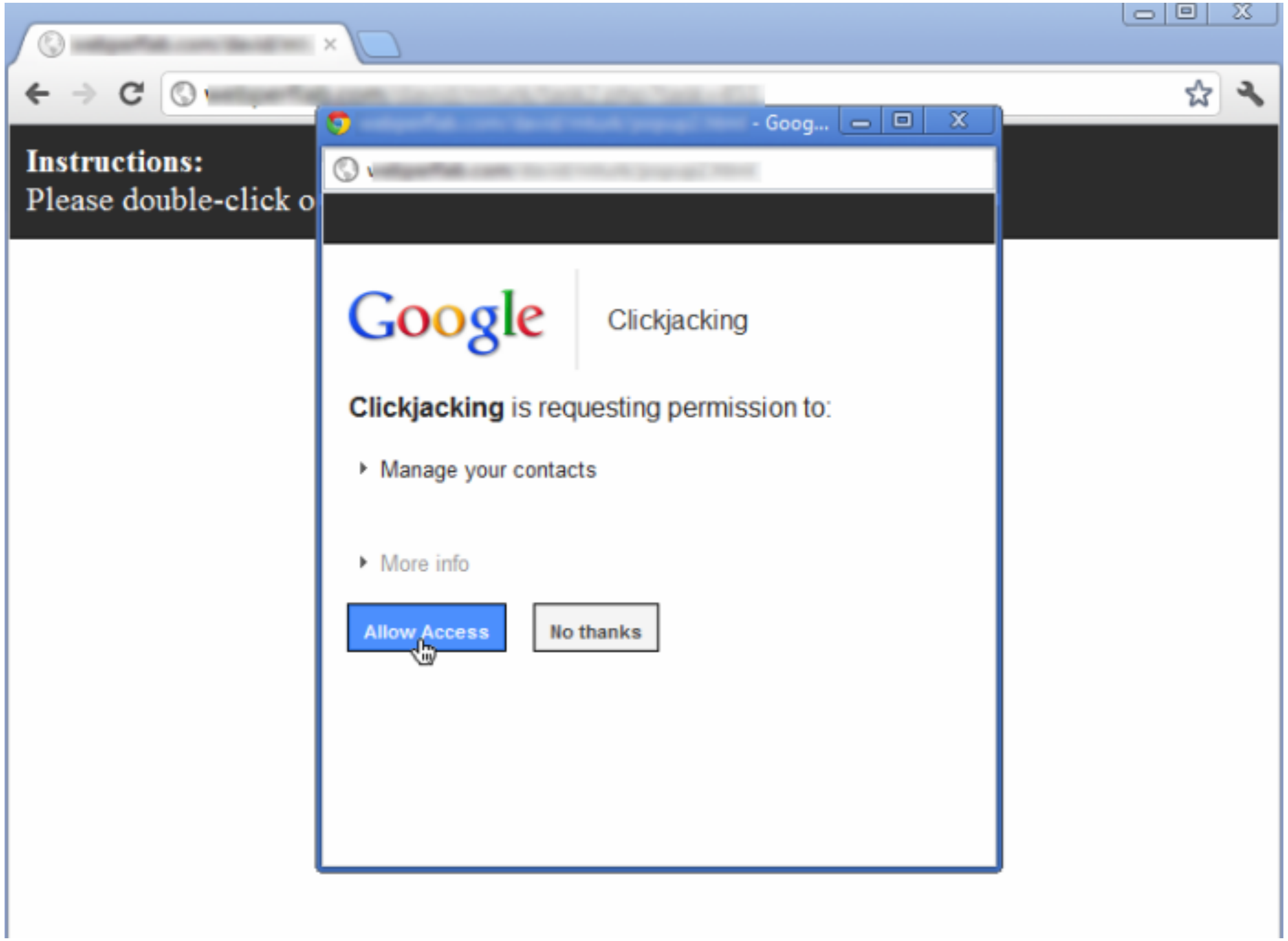


Instructions:
Please double-click on the blue button labeled "DOUBLE CLICK".

Clickjacking Attacks & Defenses – Huang et al.

Timing Attack: Bait-and-Switch

- After a click?



- Goog...

Instructions:
Please double-click o

Google | Clickjacking

Clickjacking is requesting permission to:

- Manage your contacts
- More info

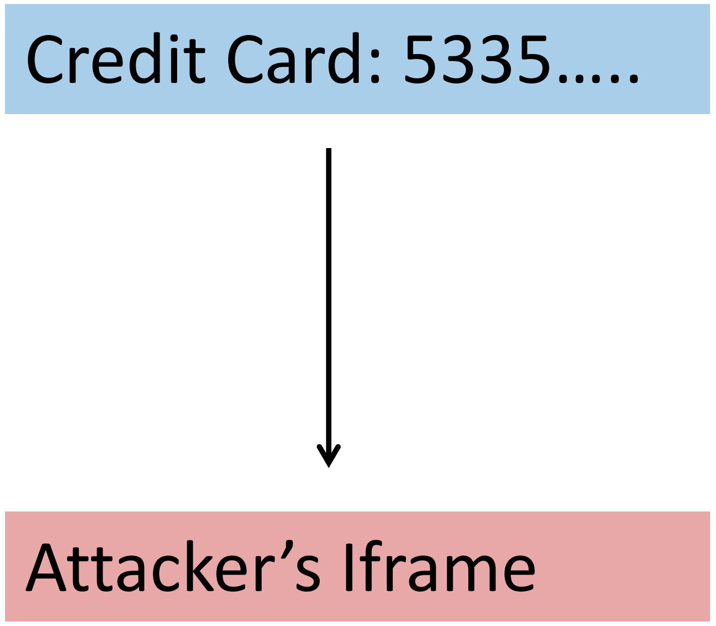
Clickjacking Attacks & Defenses – Huang et al.

Timing Attack: Bait-and-Switch

- Also known as a “double-click UI attack”:
  - Bait the user to perform a double-click,
    then switch focus to a popup window under the cursor right between the two clicks
- General attack strategy:
  - Manipulate UI elements after the user has decided to (double-) click, but before the actual (double-) click occurs
- The attack compromise *temporal integrity*

Drag-and-Drop Attacks

- Drag-and-drop API



Credit Card: 5335…..

Attacker’s Iframe

Drag-and-Drop Attacks

- Modern browsers support drag-and-drop API
- Used to set data when an element is being dragged and read it when it's dropped
- Can data from one origin be dragged to a frame of another origin ?
  - Yes!
- Why/reasoning?
  - Drag-and-drop *can only be initiated* by user's mouse gesture, not by JavaScript on its own
- Any chances of enticing web users into drag-and-dropping?

Drag-and-Drop Attacks

- *Who like to play web games?*

DRAG THIS BALL TO THE BASKET !

Read: Paul Stone, “Next Generation Clickjacking”

Defenses: Framebusting

- Prevent victim site from being “framed”
  - A.k.a. “frame-busting”
  - Idea: test whether I am a page owner
  - Easy, right?

Condition

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

Counter-Action

IE 7:

var location = “clobbered”;

Safari:

window.__defineSetter__("location", function(){});

top.location is now undefined. ☹

If My Frame Is Not On Top …

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!=

All of these methods are broken

… Just Move Mine To the Top

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
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)

Busting Framebusting, also see: OWASP Defense Cheat Sheet

Defenses:
Other Clickjacking defenses

- User confirmation
  - degrades user experience
- UI randomization
  - unreliable (e.g. multi-click attacks)
- Framebusting (`X-Frame-Options`)
  - incompatible with embedding 3rd-party objects
- Opaque overlay policy (Gazelle browser)
  - breaks legitimate sites
- Visibility detection on click (NoScript)
  - false positives

Defenses: Framebusting

- More recently, a more robust defense HTTP Header
- Tell browser not to render a page in a frame:
    - `X-Frame-Options: DENY`
- What about same-origin attacks?
    - Victim meant to be included w/o frames
        - E.g. Facebook Like buttons
    - Outer and inner frame of same origin
    - `X-Frame-Options: SAMEORIGIN`
- Or for code that needs to be framed?
    - E.g. Google Maps
    - `X-Frame-Options: ALLOW-FROM https://example.com/`

X-Frame-Options HTTP Header

- Potential limitations:
  - Per-page policy specification
  - The ALLOW-FROM option is a relatively recent: browser support issue
  - The current implementation does not allow for a whitelist of domains that are allowed to frame the page
- Ref: https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet#Defending_with_X-Frame-Options_Response_Headers

Using Content Security Policy (CSP)

- `frame-ancestors` directive:
  - Can be used in a CSP HTTP response header to indicate whether or not a browser should be allowed to render a page in an iframe

- Some usage examples:
  - `Content-Security-Policy: frame-ancestors 'none’;`
  - `Content-Security-Policy: frame-ancestors 'self’;`
  - `Content-Security-Policy: frame-ancestors 'self' '*.somesite.com' 'https://myfriend.site.com';`

Using Content Security Policy (CSP)

- Potential limitations:
  - CSP frame-ancestors is not supported by all the major browsers yet
  - X-Frame-Options takes priority:
    CSP Spec says X-Frame-Options should be ignored if CSP frame-ancestors is specified, but Chrome 40 & Firefox 35 ignore the frame-ancestors directive and follow the X-Frame-Options header instead
- Ref: https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet#Defending_with_Content_Security_Policy_.28CSP.29_frame-ancestors_directive

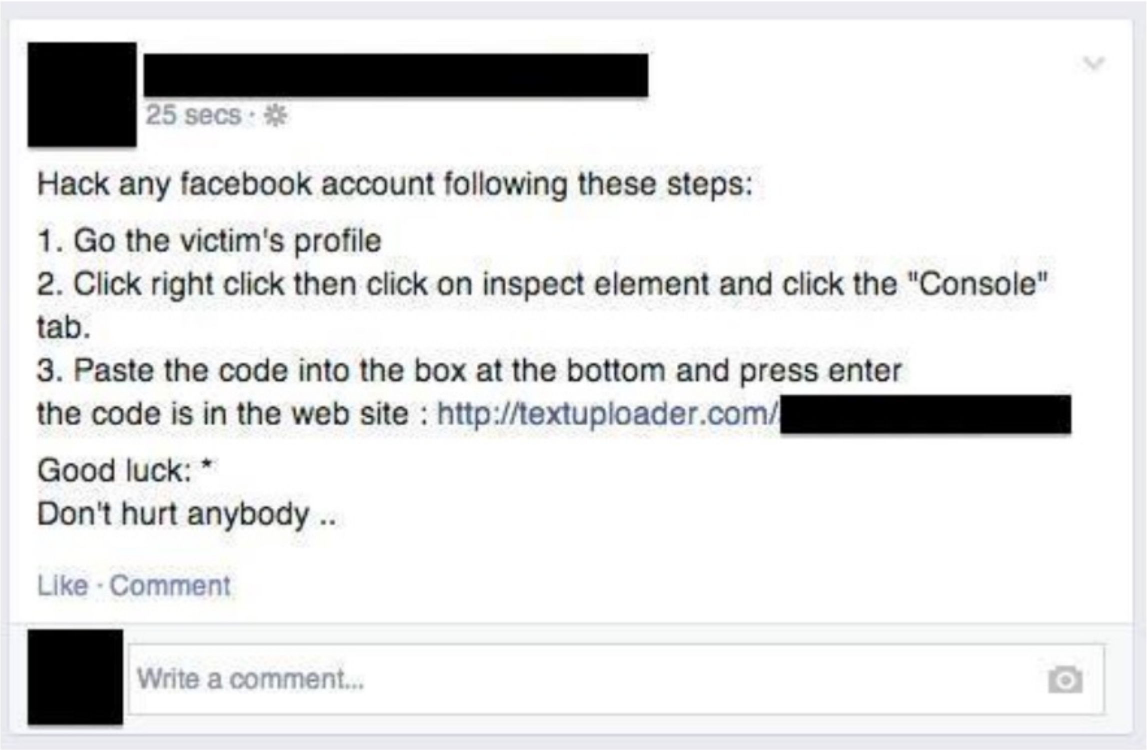
Copy-Paste XSS or Self-XSS

Self-XSS

- Coercing users to copy-and-paste…
- Example:
  - [Facebook Self-XSS]
  - Entice a user into copypasting text (parts of a webpage) into his/her browser's *URL bar*
  - Applied counter measures:
    - https://bugzilla.mozilla.org/show_bug.cgi?id=656433
    - https://bugs.chromium.org/p/chromium/issues/detail?id=82181

More Self-XSS

- How about this copypasting into browser’s *developer tool’s console* instead?



[redacted] [redacted]

25 secs ·

Hack any facebook account following these steps:

1. Go the victim's profile
2. Click right click then click on inspect element and click the "Console" tab.
3. Paste the code into the box at the bottom and press enter

the code is in the web site : http://textuploader.com/[redacted]

Good luck: *
Don't hurt anybody ..

[redacted]

From: https://www.facebook.com/notes/facebook-security/dont-be-a-self-xss-victim/10152054702905766/

More Self-XSS

- Possible defenses?
  - User education: https://www.facebook.com/help/543344735779134/
  - Browser support: https://bugzilla.mozilla.org/show_bug.cgi?id=994134 https://bugs.chromium.org/p/chromium/issues/detail?id=345205

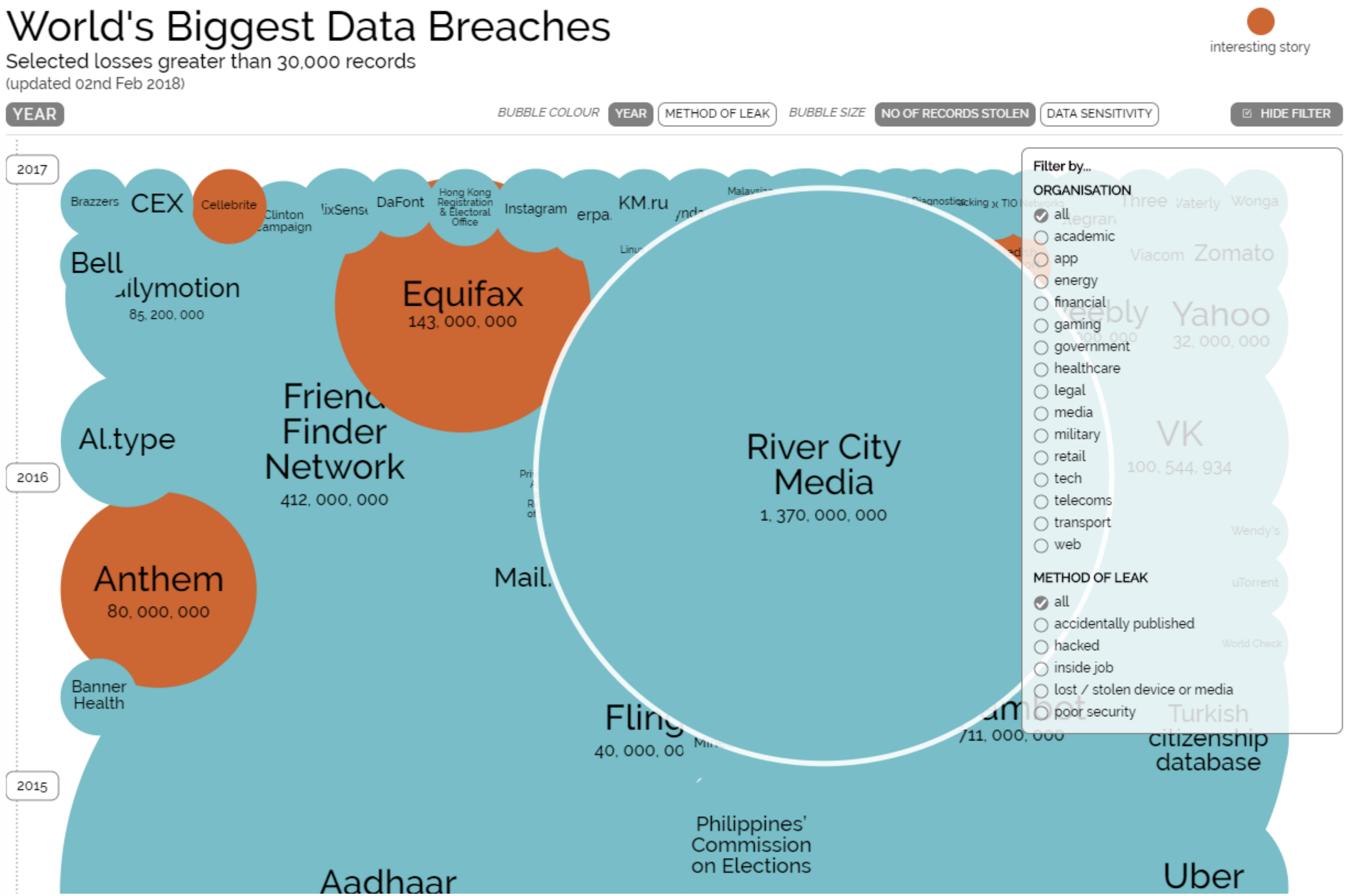
Common Theme

- Common theme of discussed UI attacks:
  - Entice user to perform a UI action:
    - Click
    - Double-click
    - Drag-and-drop
    - Copy-and-paste
  - Provide enough incentive for the user to do so
  - Ultimately perform unintended operations that are detrimental to the user!

Password Theft & Cracking

"I memorized all of my e-mail addresses, passwords, and PIN numbers...but now I don't remember my name!"

Password Data Breaches



World's Biggest Data Breaches

Selected losses greater than 30,000 records

(updated 02nd Feb 2018)

BUBBLE COLOUR: YEAR | METHOD OF LEAK

BUBBLE SIZE: NO OF RECORDS STOLEN | DATA SENSITIVITY

Filter by...

ORGANISATION

- ☑ all
- ○ academic
- ○ app
- ○ energy
- ○ financial
- ○ gaming
- ○ government
- ○ healthcare
- ○ legal
- ○ media
- ○ military
- ○ retail
- ○ tech
- ○ telecoms
- ○ transport
- ○ web

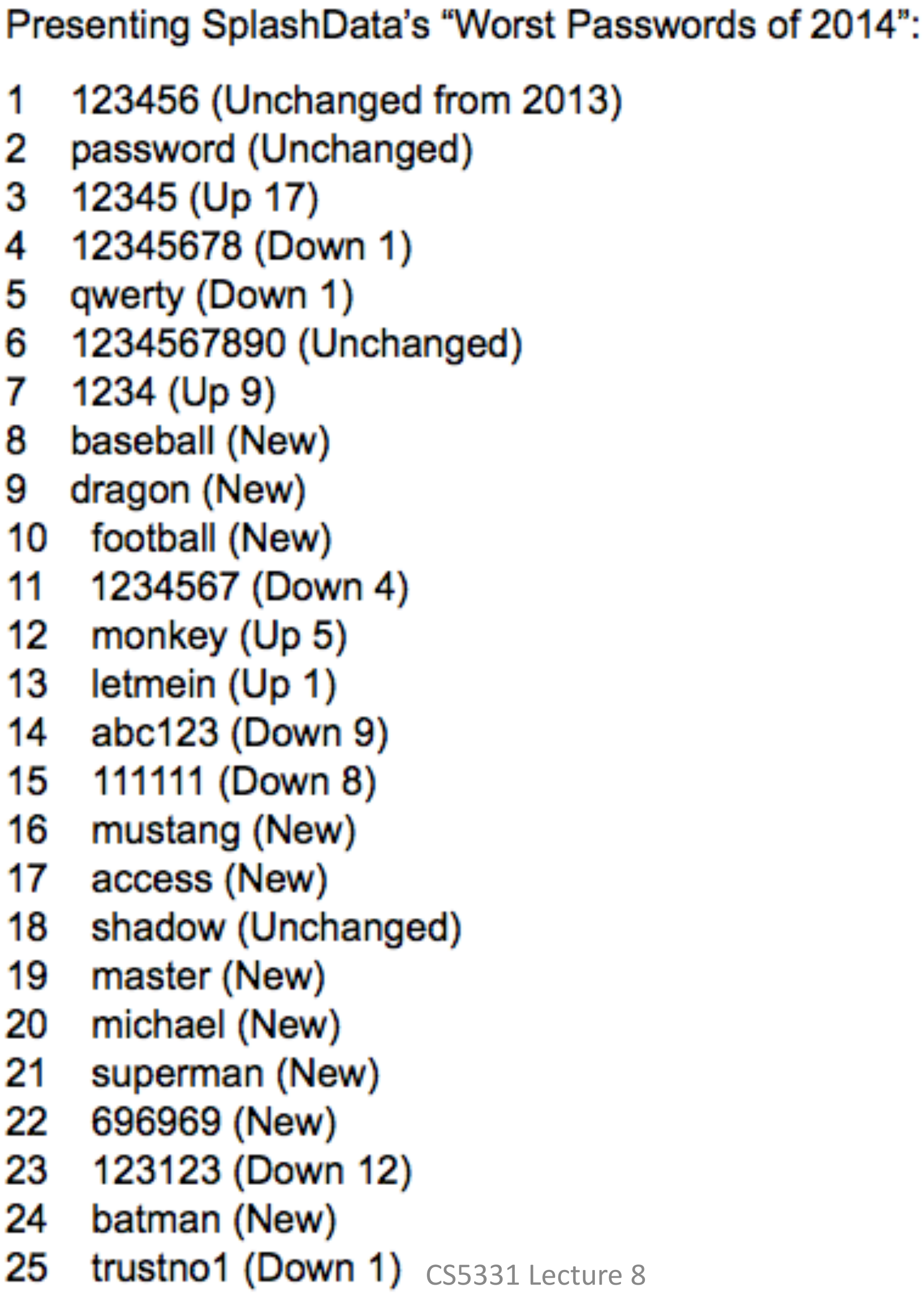
METHOD OF LEAK

- ☑ all
- ○ accidentally published
- ○ hacked
- ○ inside job
- ○ lost / stolen device or media
- ○ poor security

DataBreaches (Information is Beautiful [Website])

Password Cracking

- Passwords shouldn't be stored in plaintext
- One solution: store ***H*(*pwd*)**
  - Why hash?
    - Hash functions have a 'one-way' property
- Weakness: How many attempts for 32-bit hash?
  - To crack one password?
  - To crack any one from a list of 4M passwords?
- Solution: Salt and hash each password
  - *H* (*r* | | *pwd*) // *r* can be stored in plaintext
  - Same effort to crack one password
  - Dictionary-based attacks are harder on a list (can't reuse guesses)
- BTW, many password cracking tools exist [e.g. JTR]
  - Start with a known set of words
  - Combine them using rules
    - E.g. concatenate, replace "e" with "3", etc.
  - Brute-force them



Presenting SplashData's "Worst Passwords of 2014":

1 123456 (Unchanged from 2013)

2 password (Unchanged)

3 12345 (Up 17)

4 12345678 (Down 1)

5 qwerty (Down 1)

6 1234567890 (Unchanged)

7 1234 (Up 9)

8 baseball (New)

9 dragon (New)

10 football (New)

11 1234567 (Down 4)

12 monkey (Up 5)

13 letmein (Up 1)

14 abc123 (Down 9)

15 111111 (Down 8)

16 mustang (New)

17 access (New)

18 shadow (Unchanged)

19 master (New)

20 michael (New)

21 superman (New)

22 696969 (New)

23 123123 (Down 12)

24 batman (New)

25 trustno1 (Down 1)

Password Recovery

- Self-service password reset as a fallback authentication mechanism:
  - Enhancing usability: a user can still login even if password is lost
  - Reducing cost: reduces operating cost of helpdesk
- Common: “Secret” questions?
  - Name your pet, aunt’s middle name, movie…
  - Problem: not really secret!
- Two-factor Authentication
  - Varieties: What you know, what you have, who you are, where you are, what you do
  - Pros and cons?

Password Protection Measures

- **Limited login attempts:**
  - Add delay into login session
  - Add security questions
  - Lock the account after a few failed attempts
- **Password checker or metering**:
  - System checks for weak password when user registers/ changes password (for e.g. using password dictionary)
  - Password metering indicates weak, average, strong passwords
- **Password usage policy**:
  - Users must use strong passwords, and minimize password loss
- **Password ageing**:
  - Users must regularly change passwords

Summary

- Web UI attacks
  - Phishing
  - Clickjacking
  - Self-XSS
- Password attacks
  - Password cracking
  - Password storage