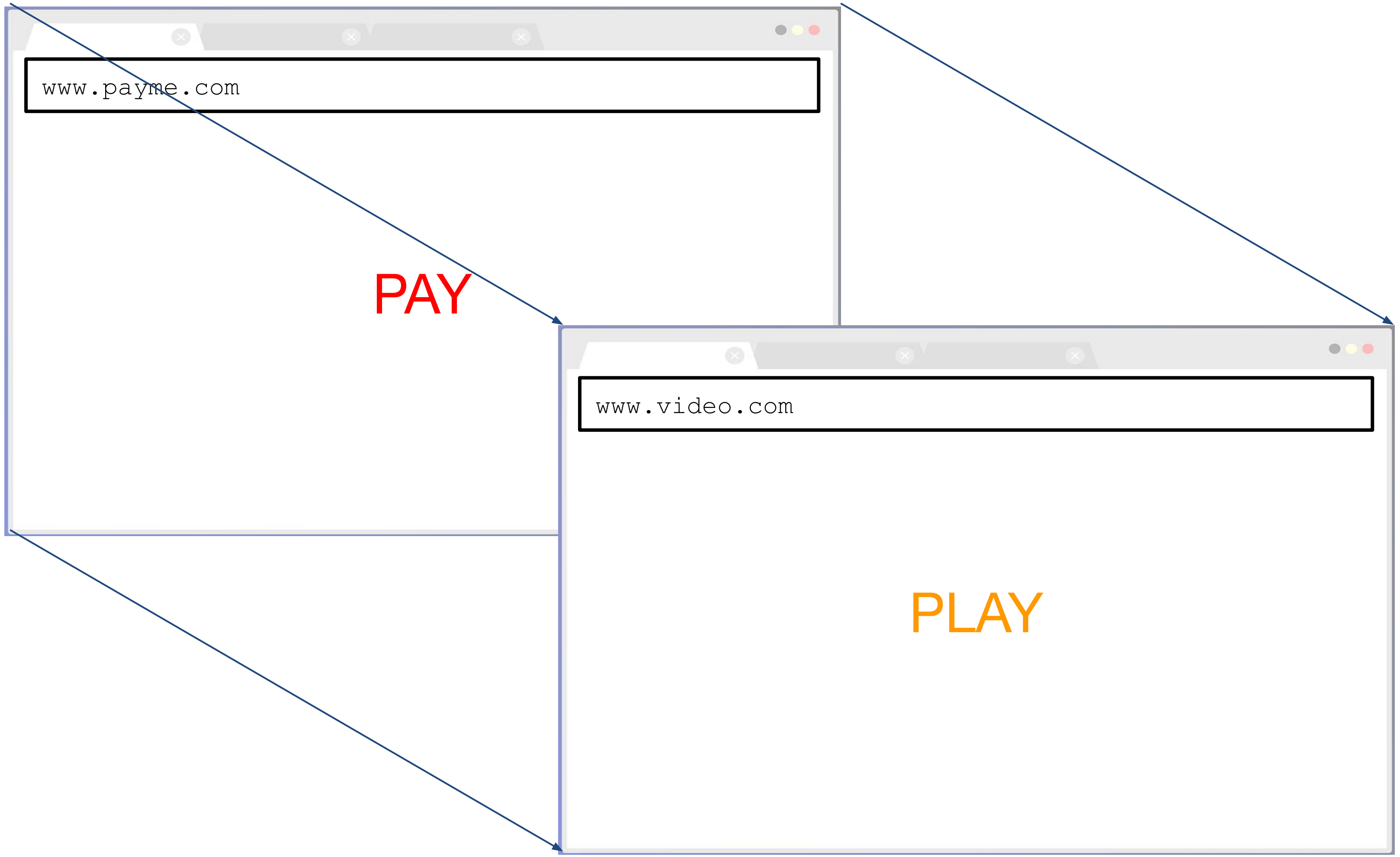
ClickJacking

Kameswari Chebrolu

Clickjacking

- An interface-based attack
- Tricks a user into clicking a webpage element
  - Is invisible or disguised as another element
  - What is achieved?
    - Can unwittingly download malware
    - Visit malicious web pages
    - Provide sensitive information
    - Purchase products online etc
    - Launch other attacks like XSS


www.payme.com
PAY
www.video.com
PLAY

Variants

- Likejacking: Like button is manipulated causing users to “like” a page without intending
  - E.g facebook or twitter post
- Cursorjacking: change position of cursor
  - Relies on vulnerabilities in Flash which have now been fixed!

Attack-1

- Redirect users to other pages via Exploiting user’s mouse clicks

```
<a onMouseUp=window.open("http://www.malicious.com")
href="http://www.trusted.com/">Claim your gift coupon!</a>
```

  - onMouseUp is a javascript function
  - mouseup event: Occurs when a mouse button is released after being pressed down over an element
  - window.open() function in JavaScript opens a new browser window or tab with a specified URL

- Why redirect?
  - Help increase traffic to a website can result in higher search engine ranking!
  - Facilitate drive-by-download
    - exploits vulnerabilities in the web browser or browser plugins (such as Adobe Flash or Java), or operating system

Attack-2

- Steal login credentials
- Any site can frame another site via HTML frames
- If an attacker frames a bank website in his website, and victim enters credentials, who gets the bank login information ?
  - Bank due to same-origin policy

- Malicious site frames good site
  - Login box of good site is framed by another **invisible** frame
  - Victim sees login of good site and enters login
  - But entered login goes to invisible frame of malicious-site!

Iframe Demo

- Embedding links
- Positioning
- Opacity
- Z-index

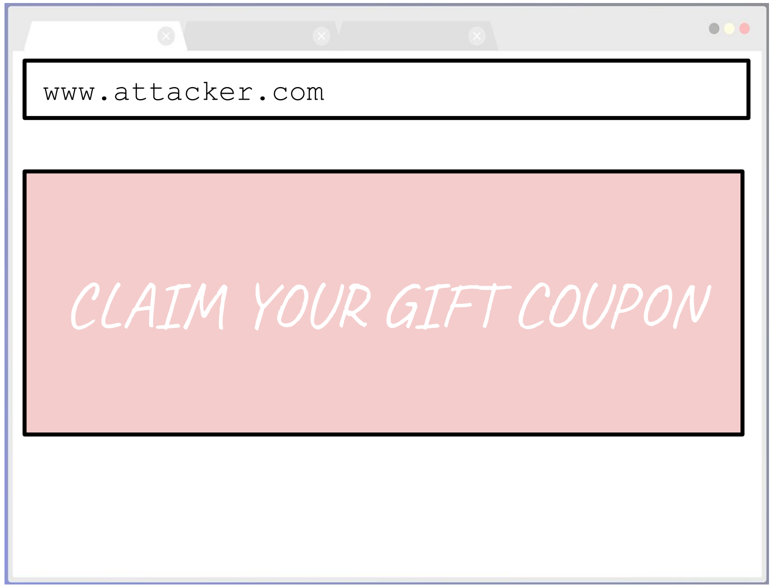
```
<head>
  <style>
   #top-evil {
       position:relative;
       width:128px;
       height:200px;
       opacity:0.0001;
       z-index:2;
       }
   #bottom-victim {
       position:absolute;
       width:200px;
       height:300px;
       z-index:1;
       }
  </style>
</head>
<!-- ... -->
<body>
  <iframe id="bottom-victim"
src="https://www.example.com">
  </iframe>
  <div id="top-evil">
  ...Attacker has to include relevant login fields
and align properly via styling...
  </div>

</body>
```

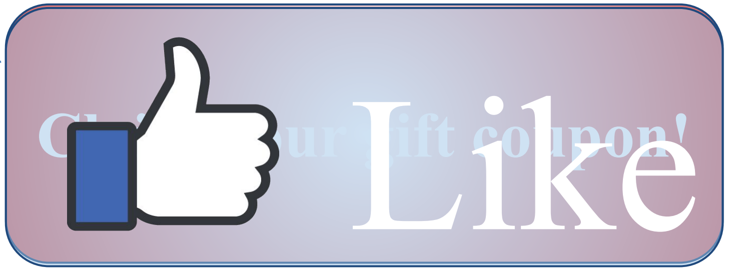
Construction

- z-index determines the stacking order
  - Vulnerable site is bottom, attacker site on top
  - Elements with a higher z-index will be placed on top of elements with a lower index
- Evil site opacity⟶ make it transparent (0.0 or close to 0)
- Evil site position so that there is precise overlap of the actions with the victim website
  - Managed using appropriate width and height position value
  - Absolute and relative positions ensure proper overlap regardless of screen size, browser type and platform

Like Jacking



Opacity: 0%; invisible Facebook frame is on top of it



Opacity: 50%; frame is visible

Multistep Click Jacking

- Hijacking one click is relatively easy
- To realize some attacks may need multiple clicks on same or different pages
  - E.g To trick a user into changing email address
    - Email address and password has to be typed in page1
    - “Yes” has to be clicked to confirm change in page2
- Attacker may need to design some activity/game to lure user to click on multiple positions
  - These attacks require considerable precision to be effective!

Point to Note

- Invisible frames (evil frame on top) will not allow users to interact with the bottom frame → user may get suspicious
  - Attacker can make top frame small and use it to only cover the relevant portion

Cursorjacking

- Deceives user by using a fake but more prominent cursor
  - Can be made to move via javascript to resemble real cursor
- Fake cursor points to something desirable (win a gift coupon!) while real cursor points to something malicious (download malware)
- User thinks he is clicking fake cursor but is clicking real cursor

Example

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

Defense: Frame Busting Script

- Earlier days, no support from browsers against this attack
- Web developers wrote Javascript code to avoid being framed by others!
  - E.g. check and enforce that the current application window is the main or top window

- Sample code:

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

  - If a page is in an iframe, windows.top and windows.self will be different (else same)
  - If page displayed is being farmed by another website, above code redirects the entire browser window to the same page outside of that frame
- Javascript protection is not reliable
  - e.g. if a user disabled javascript in browser, no protection possible!
- Outdated defense, no longer used!

Defense: X-Frame-Options

- Best protection: Server tells browser what to do
- Originally introduced as an unofficial response header in Internet Explorer 8
  - Rapidly adopted within other browsers
- Header is used to control whether a browser should be allowed to render a page in a <frame>, <iframe>, <embed>, or <object>.

- X-Frame-Options: DENY
  - Tells browser not to render the page in a frame under any circumstances
  - Page cannot be embedded in any other site
- X-Frame-Options: SAMEORIGIN
  - Allows page to be framed only by pages from the same origin (i.e., the same domain)
- X-Frame-Options: ALLOW-FROM URI
  - Allows the current page to be displayed in a frame, but only in a specific URI
  - E.g. X-Frame-Options: ALLOW-FROM https://trustedsite.com

- In apache, this can be configured as

```
<IfModule mod_headers.c>
        Header always set X-Frame-Options "SAMEORIGIN"
</IfModule> (or)

<IfModule mod_headers.c>
        Header always set X-Frame-Options ALLOW-FROM
https://example.com
</IfModule>
```

(mod_headers is an Apache HTTP Server module; helps manipulate/add HTTP headers in the server's responses)

Limitations

- To enable SAMEORIGIN option across a website, X-Frame-Options header needs to be returned with every HTTP response for each individual page
  - Does not apply cross-site by default
- X-Frame-Options does not support a whitelist of allowed domains
- ALLOW-FROM option is not supported by all browsers
- X-Frame-Options is a deprecated option in most browsers

Defense: Content Security Policy (CSP)

- Can protect against clickjacking also
- Incorporate frame-ancestors directive
- frame-ancestors 'none'
  - Same as X-Frame-Options deny directive
- frame-ancestors 'self'
  - Prevents framing altogether
  - Equivalent to X-Frame-Options sameorigin directive

HTTP header

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Security-Policy: default-src 'self'; script-src 'self'
https://cdnjs.cloudflare.com; style-src 'self'
https://fonts.googleapis.com; font-src 'self'
https://fonts.gstatic.com; frame-ancestors 'self';
```

- CSP is more flexible
  - Can specify multiple domains and use wildcards
  - E.g. frame-ancestors 'self' https://example.com https://*.robust-website.com
- CSP also validates each frame in the parent frame hierarchy
  - X-Frame-Options only validates the top-level frame.

Clickjacking + XSS

- Historically, clickjacking has been used mostly to do like jacking!
- Clickjacking combined with XSS can lead to powerful attack!
- XSS exploit is combined with iframe target URL
  - User clicks button/link, executes XSS attack!

Clickjacking vs CSRF

- Clickjacking: user is required to perform an action such as a button click
  - Not mitigated through CSRF token
- CSRF: forge an entire request without the user's knowledge or input

Real Life Examples

- Clickjacking at ylands.com: https://hackerone.com/reports/405342
- Clickjacking with reflected XSS: https://hackerone.com/reports/1221942

Summary

- Tricks a user into clicking a webpage element or entering some data
  - Redirect to other links, steal credentials, like comments/tweets etc
  - Managed via iframe opacity, z-index and javascript
  - Clickjacking + XSS can lead to powerful attacks
- Defenses: Frambuster, X-Frame-Options, CSP (frame-ancestors)
  - CSP is the latest and best option

References

- https://portswigger.net/web-security/clickjacking
- Computer Security: A Hands-on Approach by Wenliang Du, 2022 https://books.google.co.in/books?id=DI8szwEAC