Lecture 28

Phishing and Denial-of-Service

ECE 422: Reliable and Secure Systems Design

UNIVERSITY OF ALBERTA

Instructor: An Ran Chen
Term: 2024 Winter

Schedule for today

- Key concepts from last classes
- Phishing
  - Typosquatting
  - IDN Homograph attack
  - Pharming
- Denial-of-Service attack
  - Client-side: UI attack
  - The Annoying Site “features” and prevention

CSP nonces

A CSP nonce is a randomly generated token that is used exactly one time.

- Generate random numbers to give allow specific scripts when CSP is enabled

Why? Analogous as session IDs but for scripts

- As long as nonce is valid, trusted scripts can be loaded and executed
- Generated on every page load, so attackers cannot reuse the same token

How does it work?

- Generate nonce for every request to web server
- Declare nonce in the CSP header `script-src`
- Add it to scripts tags

strict-dynamic

strict-dynamic is a possible value inside script-src directive

- Used in combination with nonces

Why? Trust propagation to all the scripts loaded by the root script

- Allows any script to be included by any script with nonce attribute
- Solution to nested scripts inside nested scripts inside …

How does it work?

- Declare strict-dynamic in script-src with nonce as part of the CSP header

Content-Security-Policy-Report-Only

Content-Security-Policy-Report-Only allows developers to experiment with policies by monitoring (but not enforcing) their effects.

- Server sends Content-Security-Policy-Report-Only header instead of Content-Security-Policy
- Violation of policies presented in a report

Why? To test a policy without breaking the application

- Problem with testing CSP: If we miss something (e.g., attribute events), the website will break → unhappy customers
- Report-only mode offer a solution to this problem

Final take-homes on CSP

- XSS are still relevant in real-world web applications
- XSS: convert user’s data into code
- Always sanitize user’s data: Escaping the input based on the context
- Use CSP to prevent almost all XSS attacks
- CSP nonces and strict-dynamic make it easier to implement CSP
- CSP report-only mode makes it easier to test CSP

Schedule for today

- Key concepts from last classes
- Phishing
    - Typosquatting
    - IDN Homograph attack
    - Pharming
- Denial-of-Service attack
    - UI attack
    - The Annoying Site “features” and prevention

Phishing

Phishing is a form of social engineering that deceive victims into sharing sensitive information such as login credentials or account details.

- Social engineering attacks rely on human error
- Tricking users with fraudulent emails, text messages, phone calls or websites

Cyber security is a "people problem":

- Security solutions have a technological component that we can make as secure as possible
- But it is often easier to trick people than attacking the security of a system

Phishing

There are three common types of phishing on the web:

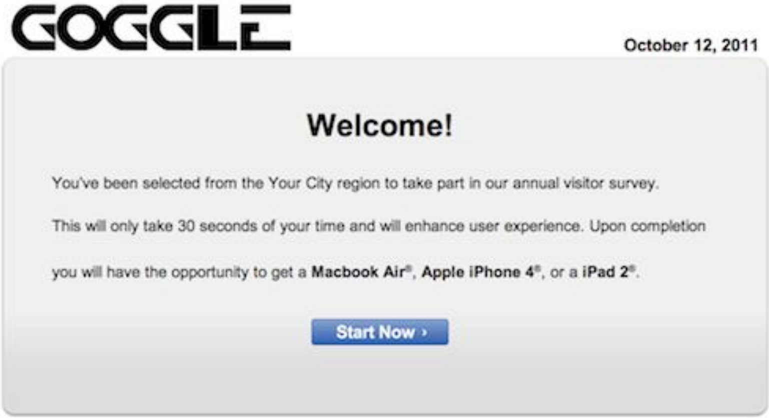
- Typosquatting: Use similar-looking domain to trick the victim
- IDN Homograph attack: Rely on alternate character sets used in other countries to produce similar-looking domain to trick the victim
- Pharming: Redirect users to a malicious website by injecting entries into Domain Name System
- Picture-in-picture attack: Use a fake browser address bar as an image inside the actual browser (rarely used nowadays)

MOVIE HACKING...
IF I CAN JUST OVERCLOCK THE UNIX DJANGO, I CAN BASIC THE DDOS ROOT. DAMN. NO DICE. BUT WAIT... IF I DISENCRYPT THEIR KILOBYTES WITH A BACKDOOR HANDSHAKE THEN... JACKPOT.
REAL HACKING...
HI, THIS IS ROBERT HACKERMAN. I'M THE COUNTY PASSWORD INSPECTOR.
HI BOB! HOW CAN I HELP YOU TODAY?
VIA 9GAG.COM

Typosquatting

Typosquatting (also known as URL hijacking) targets victim who incorrectly type a website address into their web browser.

- Users are tricked into thinking that they are in fact in the real site
- Example: goggle.com vs google.com



Welcome!

You've been selected from the Your City region to take part in our annual visitor survey.

This will only take 30 seconds of your time and will enhance user experience. Upon completion you will have the opportunity to get a **Macbook Air®**, **Apple iPhone 4®**, or a **iPad 2®**.

Internationalized Domain Names (IDN)

Internationalized Domain Names (IDN) are domain names which use a wide range of Unicode characters used in different languages.

- Including letters or characters from non-Latin scripts (e.g., Arabic or Chinese characters) in domains
    - Japanese .jp domain registry services: 日本語.jp
    - Starbucks Korea: 스타벅스코리아.com

Why? To allow more web users to navigate in their preferred language

- Most domain names are registered in ASCII characters (A to Z, 0 to 9, and the hyphen "-")
- However, languages that require diacritics cannot be rendered in ASCII
- Also useful for brand localization

Internationalized Domain Names (IDN)

How it works? This is done by transcoding Unicode characters to punycode

- Punycode is a representation of Unicode with the limited ASCII character subset used for Internet host names
- Starting with the prefix "xn--" to signal the domain name is using punycode

Example of IDNs:

- 日本語.jp → https://xn--wgv71a119e.jp
- 스타벅스코리아.com → http://xn--oy2b35ckwhba574atvuzkc.com

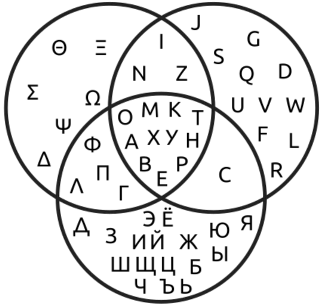
Try it yourself:

- 日本語.jp
- https://xn--wgv71a119e.jp

IDN homograph attack

IDN homograph attack exploits the fact that different characters from different writing systems look alike.

- Users may not notice subtle differences in characters from different writing systems (e.g., Latin, Cyrillic or Greek)
- Often happening on IDNs that allow non-ASCII characters
- Font family can also cause issues (e.g., m vs rn vs rri)



IDN homograph attack

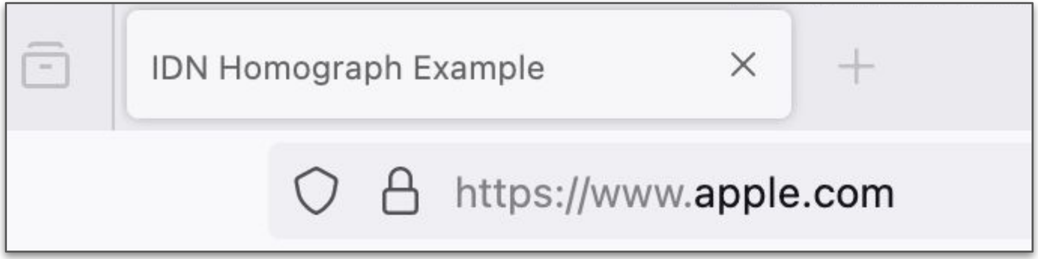
Example: Can you tell which one is the phishing site?

- https://www.apple.com
- https://www.apple.com

Try visiting https://www.apple.com/ in Firefox vs Chrome

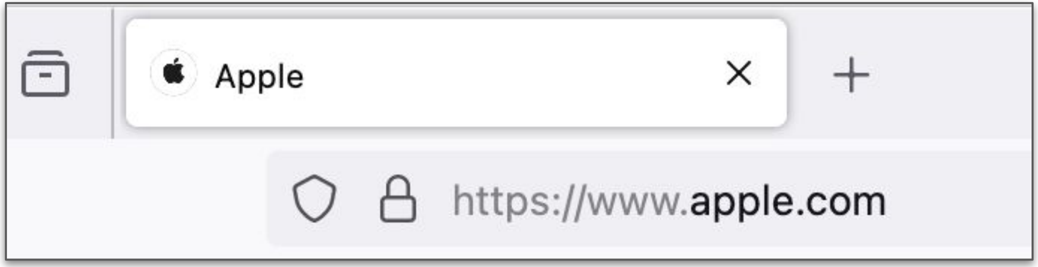
- First ‘apple.com’ uses Cyrillic characters rather the ASCII characters
  - E.g., Cyrillic ‘a’ (U+0430) vs ASCII “a” (U+0041)
- Two websites returns different hashes
  - Recall hash function: same input produces same output
  - First ‘apple.com’ produces 7a9f74
  - Second ‘apple.com’ produces 15fb0e
  - Try it yourself: UTF-8 to SHA256

IDN homograph attack



IDN Homograph Example

https://www.apple.com



Apple

https://www.apple.com

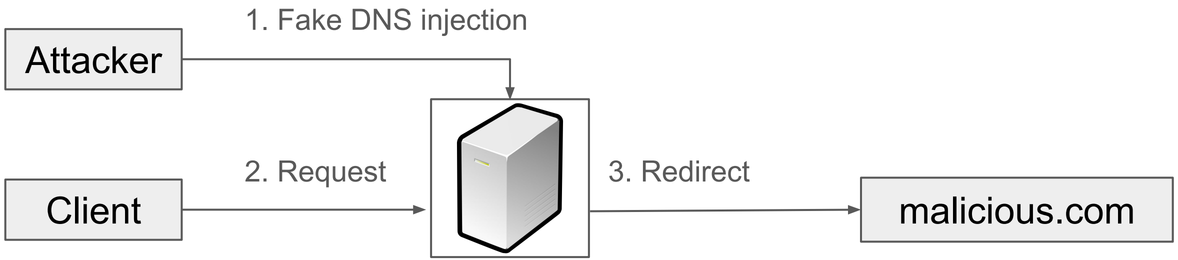
Pharming attack

Pharming attack redirects users to a fake website that mimics a real website.

- Manipulate the Domain Name System to redirect user’s request without their knowledge

How it works? Attacker injects fake DNS entry on the DNS server

- Request to the DNS server is redirected to attacker’s malicious website
- Attacker gets access to user credentials



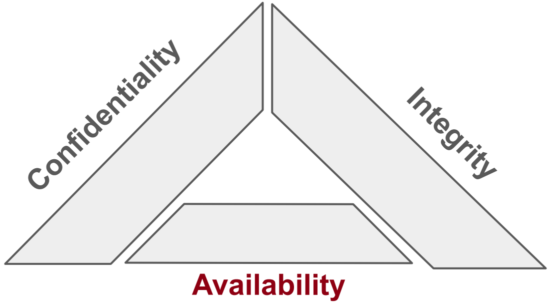
Schedule for today

- Key concepts from last classes
- Phishing
  - Typosquatting
  - IDN Homograph attack
  - Pharming
- Denial-of-Service attack
  - UI attack
  - The Annoying Site “features” and prevention

Denial-of-Service attack

Denial-of-Service attack, also known as DoS attack, is designed to render a service inaccessible to its users.

- Attack on the availability of the system



There are typically two types of DoS attack:

- Server-side DoS attack: Network or server attacks
  - Make the server or network resource unavailable by flooding it with superfluous requests to overload the system
- Client-side DoS attack: UI attacks (or clickjacking attacks)
  - Trick a victim into inadvertently clicking on an attacker-supplied input.

Note that DoS is different from Distributed Denial-of-Service (DDoS) attack

UI attacks

UI attacks (or clickjacking attacks) are a category of attacks that try to fool a victim into inadvertently clicking on an attacker-supplied input.

UI attacks can have different goals:

- Override browser defaults to disorient users on site
  - E.g., Advertisement pretending to be part of the website UI
- Scareware to intimidate the user into trapping them on an unwanted site
  - E.g., Pop-ups windows with “Warning! Virus has been detected!”
- Annoy the user, mostly harmless but annoying

UI attacks

Often, attacker can compromise a vulnerable website by:

- Including malicious advertisements
- Injecting malicious scripts into third-party widgets
- Injecting malicious scripts as user-generated content (i.e., Stored XSS)

UI attacks

Example of UI attacks: Combining UI attacks with phishing + web scraping + XSS

- Big idea: “steal” a click from the user, so that the user loads something malicious

Possible scenario:

- Scraping for contact information
- Phishing users into vulnerable websites with XSS and UI attacks
  - Malicious download button added through stored XSS or as a paid advertisement
  - Buttons redirect users to malicious websites
- Ask users to login and steal their credentials

Infinite alert loop

```
const messages = [
 'Once upon a time...',
 'There is a lecture…',
 'About UI attack…',
 'Where the instructor discusses about…',
]
while (true) {
 messages.forEach(message => alert(message))
}
```

- Block any user’s input to the tab
- Force user to quite the browser

Intuition: Find a way to break the user out of infinite alert loops without needing to quit their browser.

Infinite alert loop

Different browsers have came up with different defense mechanisms:

- Chrome: multiprocess allowing close button on the tab from working
- Firefox: checkbox on popup, also multiprocess

But users still loses their session…

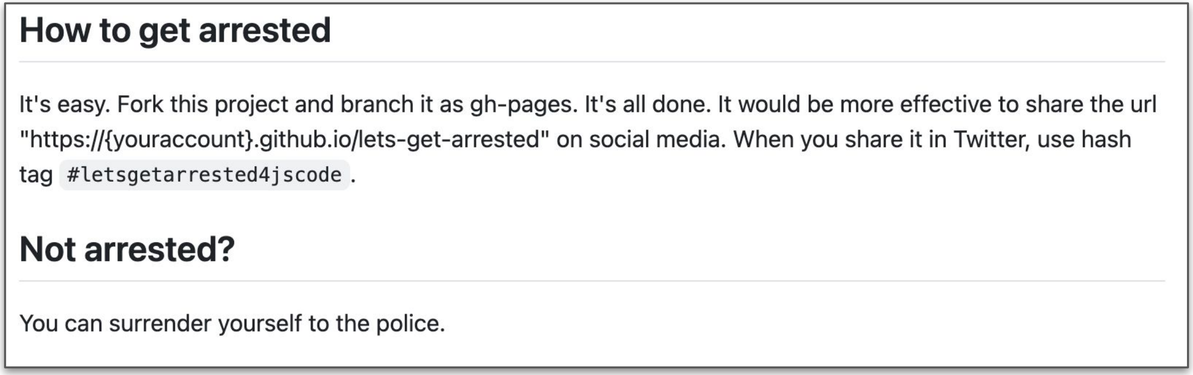
- Alternative solution available, but not for end-users
- Chrome - Source - Pause button available

Infinite Pop-up Incident

Infinite Pop-up Incident: In 2019, a 13-year-old Japanese girl was arrested for posting a infinite pop-up loop prank on a forum.

- Modern browser could close the popup
- However, majority of mobile browsers couldn't closed it

Lets-get-arrested project was launched three days later to protest against the incident:

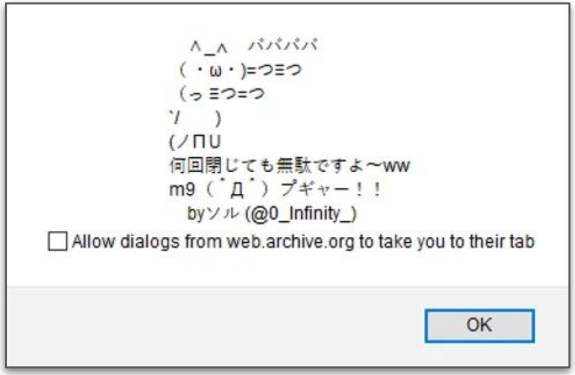


How to get arrested

It's easy. Fork this project and branch it as gh-pages. It's all done. It would be more effective to share the url "https://{youraccount}.github.io/lets-get-arrested" on social media. When you share it in Twitter, use hash tag `#letsgetarrested4jscode`.

Not arrested?

You can surrender yourself to the police.



```
  ∧_∧   ﾊﾟﾊﾟﾊﾟﾊﾟ
 （・ω・)=つ≡つ
 (っ ≡つ=つ
`/    )
(ノΠU
何回閉じても無駄ですよ～ww
m9（ﾟДﾟ）プギャー！！
   byソル (@0_Infinity_)
```

☐ Allow dialogs from web.archive.org to take you to their tab

Example of UI attacks

The Annoying Site is an example of harmless UI attack:

- https://www.theannoyingsite.com (Github@ feross/TheAnnoyingSite.com)
- Warning: Avoid opening the link in the main browser, use an alternative one that you can force to quit

Some “features” on The Annoying Site:

- Log user out
- Embarrassing searches
- Tabnabbing
- Trigger a file download

API Level

API Level	Restrictions	Examples
Level 0	No restrictions	window.move() File download CSS
Level 1	User interaction required (e.g., click or keypress)	window.open() Copy text to clipboard
Level 2	User “engagement” required	Autoplay sound
Level 3	User permission required	Camera, microphone, USB

Log user out

```
window.onload = function() {
  doSites(document.getElementById("sitelist"), [
    ["Apple", get("https://appleid.apple.com/account/signout")],
    ["GitHub", get("https://github.com/logout")],
    ["GMail", get("http://mail.google.com/mail/?logout")],
};
```

Script from SuperLogout

- Force users out of their session

What is happening behind the scenes?

Log user out

```
window.onload = function() {
  doSites(document.getElementById("sitelist"), [
    ["Apple", get("https://appleid.apple.com/account/signout")],
    ["GitHub", get("https://github.com/logout")],
    ["GMail", get("http://mail.google.com/mail/?logout")],
};
```

Script from SuperLogout

- Force users out of their session

Behind the scene:

- User lands on the website
- Website sends a HTTP request from user's browser to popular sites
- User's browser helpfully attaches user session cookies
- SOP passes, CSP passes; User logged out of their own account without knowing it …

Embarrassing searches

```
const searches = [
 'where should i bury the body',
 'why does my eye twitch',
 'why is my poop green',
 'why do i feel so empty',
]
// for each entry in search do:
let searchIndex = 1
window.location = 'https://www.bing.com/search?q=' +
encodeURIComponent(searches[searchIndex]);
searchIndex += 1
```

- Force users to search for something embarrassing

What is happening behind the scenes?

Embarrassing searches

```
const searches = [
 'where should i bury the body',
 'why does my eye twitch',
 'why is my poop green',
 'why do i feel so empty',
]
// for each entry in search do:
window.location = 'https://www.bing.com/search?q=' +
encodeURIComponent(searches[searchIndex])
};
```

- Force users to search for something embarrassing

Behind the scene:

- User lands on the website
- Website sends a HTTP request from user’s browser to popular sites
- SOP passes, CSP passes; User searches for something embarrassing …

Tabnabbing

Tabnabbing is UI attack that targets the inactive tabs in victim’s browser.

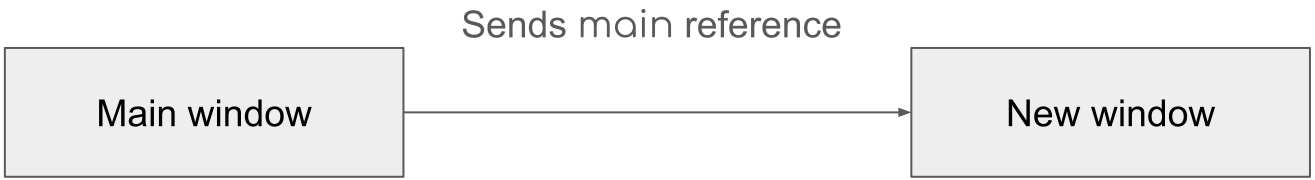
For example:

- When user clicks on a link on reddit.com with:

```
<a href='https://malicious.com' target='_blank'>External Website</a>
```

- malicious.com gets a reference to reddit.com window window.opener
- malicious.com redirects the user to a fake reddit on the main window

Window opener property returns a reference to the window that created the window



Tabnabbing prevention

To prevent tabnabbing, add rel='noopener' to all links with target='_blank'

- The opened site's window.opener will be null
- From 2021, all browsers treat target="_blank" as implying rel="noopener"
- Tabnabbing is rare today

Next class: Bitcoin