FA2022 Week 03

Web Hacking II

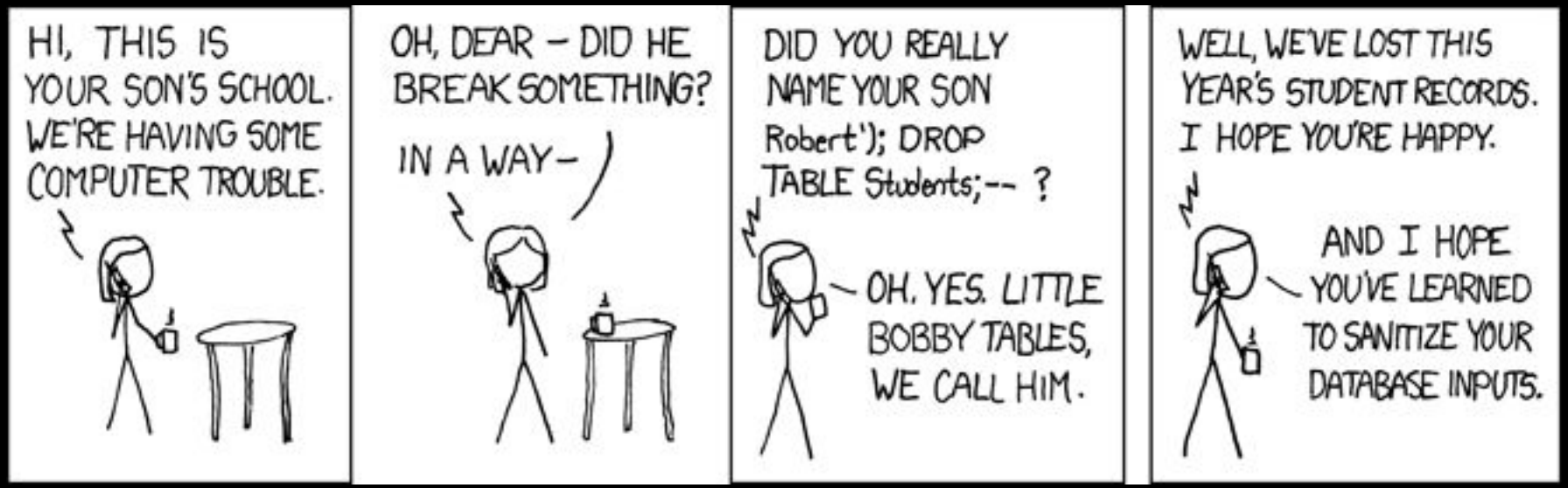
Pete and Kevin

Announcements

- CSAW Results
  - We placed 9th overall and 8th in our region, so we qualified for finals in New York!

- Fall CTF
  - Next weekend (not this weekend!)
  - September 24th at 12 - 6PM
  - **CIF 3039 (NOT SIEBEL CS 1404!!)**

sigpwny{is_it_pronounced_sql_or_sql}



Overview for Today

SQL Injection (SQLi)

- SQL Overview
- Injection
- Example

Cross site scripting (XSS)

- Javascript recap
- Injection
- Example

SQL Injection

Malicious user input that changes a SQL statement

SQL Overview

- Used to retrieve or store things in a database

- SQL "queries" are run on the server
  - Search for products

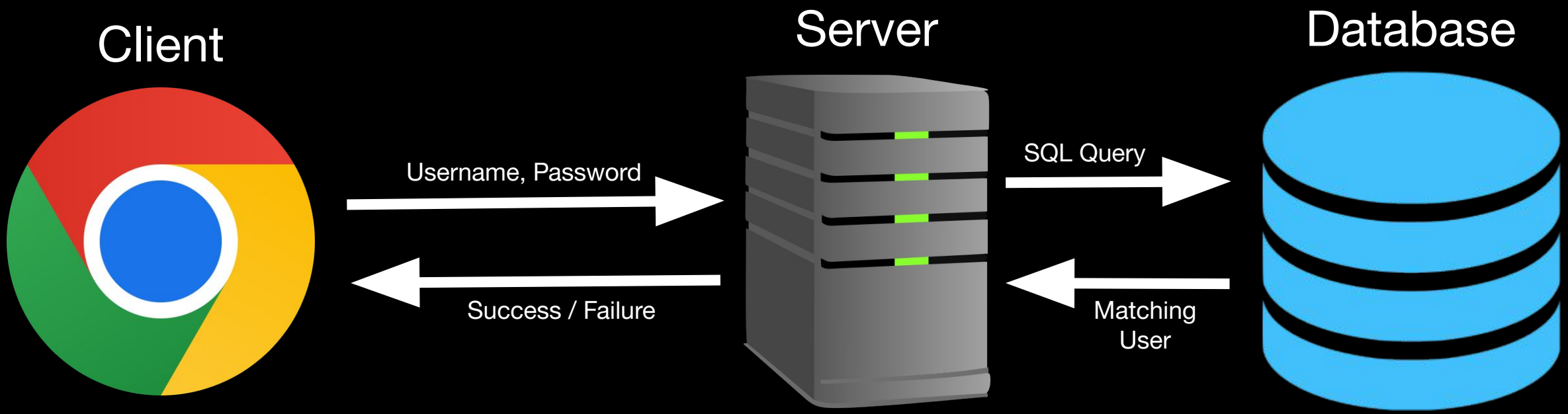
```
SELECT * FROM products WHERE name CONTAINS 'orange'
```

  - Add accounts

```
INSERT INTO users (username, pass)
VALUES ('lastpass_dev', 'password')
```

  - User logins

User Login Flow



User Login Query

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

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

the table called "users"

such that the following conditions are true...

- the username column (field) is "admin"
- the password column is "password"

Server Code

```php
<?php
    $username = $_POST['username'];
    $password = $_POST['password'];

    //Actual SQL query is here V
    $query = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";

    $results = $db->query($query);
    $row = $results->fetchArray();

    echo 'Welcome', $row['username'];
?>
```

It puts our username input directly into the query!

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

```
SELECT * FROM users WHERE username = '$username' AND password = '$password'
```

↓

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

← `--` is a line 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 any password!

SQL Injection Techniques

- Basic
  - Login as other users by changing clause
  - SQL 1 challenge
- Union
  - Exfiltrate additional data from SQL database (users, passwords, other tables, etc)
  - SQL 2 challenge
- 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'?

Additional Learning

sqlmap

- Automated SQL Injections

portswigger

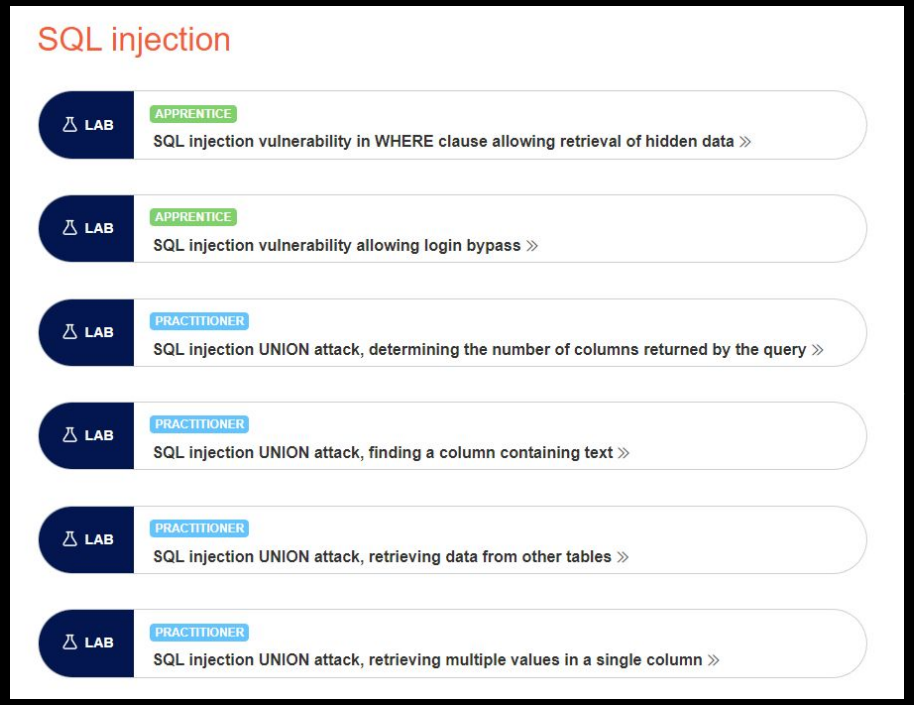
- Tutorials and practice for SQL Injections

```
$ 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')
```



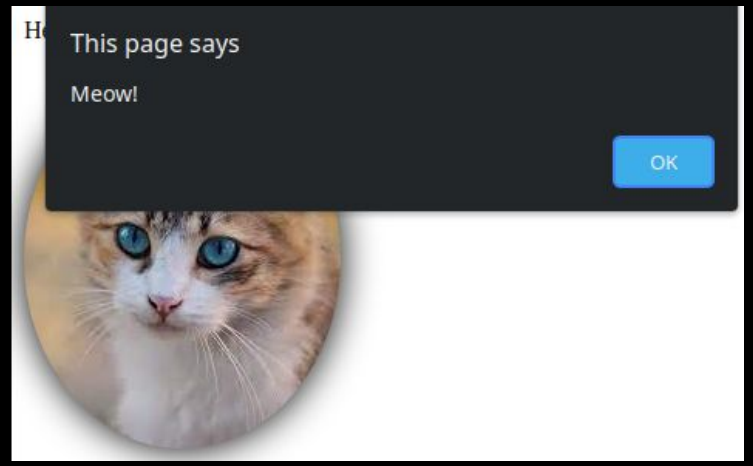
Cross-Site-Scripting (XSS)

Attacker's javascript on web pages that other clients execute!

JavaScript Recap

- Programming language that interacts with website
- Runs in **browser** (client side!)

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



Simple View Message App

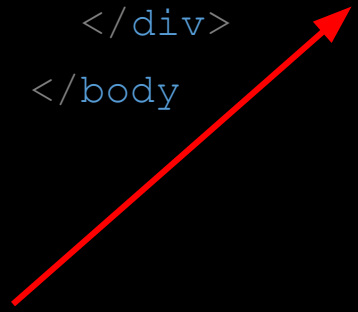
Server Code (app.js)

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

- Can 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 put directly in HTML

Message link

View message

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

/view?message=hello

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

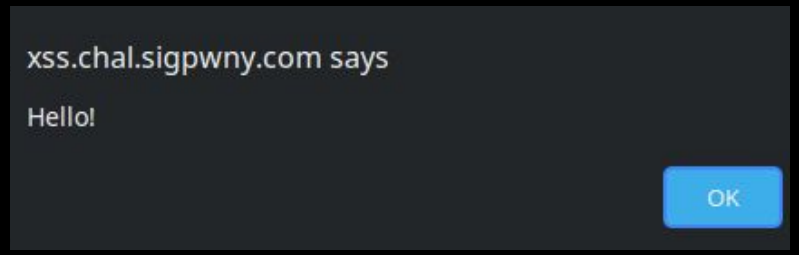
/view?message=bold

```
<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 Techniques

- <script>alert(1)</script>

-
  - also onerror=...

- SVG XSS!

- hacktricks.xyz
  - Extremely detailed list of XSS attack types

XSS Scope

- Valuable information can be stored client-side
  - Cookies, Page Contents
- Imagine if twitter had this vulnerability
  - Everyone who views a malicious tweet instantly follows attacker on twitter

XSS Post-Exploitation

- Send link with XSS to the victim
- Steal cookies, read page contents, perform action as user
- Why is XSS necessary?
  - Same-Origin-Policy (SOP)
  - Javascript can only read site info from the same domain
    - What happens without this?

Go try for yourself!

https://ctf.sigpwny.com

- 2 SQL Challenges
- 3 XSS Challenges

Next Meetings

2022-09-18 - This Sunday

- Reverse Engineering Setup
- We will setup Ghidra, pwntools, and other rev tools!

2022-09-22 - Next Thursday

- Rev I
- Learn how to use Ghidra and how to reverse engineer apps!

2022-09-24 - Next Saturday

- Fall CTF 2022
- 12PM - 6PM in CIF 3039