CodePath

Week 2

CS 4760 F18
Christopher Raley

Topics

Week 2

Readings available on CodePath Course website

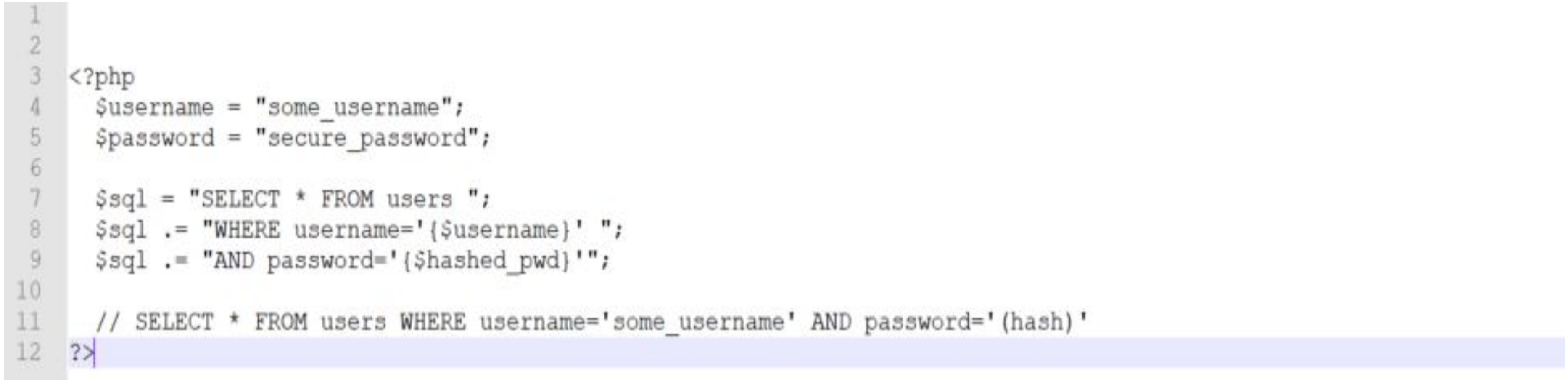
- **Intro to SQL**
- **SQL Injection (SQLi)**
- File Upload Abuse
- Remote Code Execution

Data Query Languages

- Languages that manage databases
  - XPath/Xquery, LDAP, **SQL**, etc.
  - Also known as database/query languages
- CodePath uses SQL consistently throughout all activities

SQL

- Data Query Language
- Used to store, update, and retrieve information in databases
- Stores data in tables
    - Commands like “SELECT”, “WHERE”, “FROM”, etc. are used to query data from such tables
    - SQL also has logical operators (AND, OR)
- Let’s do some high level examples



```


<?php
  $username = "some_username";
  $password = "secure_password";

  $sql = "SELECT * FROM users ";
  $sql .= "WHERE username='{$username}' ";
  $sql .= "AND password='{$hashed_pwd}'";

  // SELECT * FROM users WHERE username='some_username' AND password='(hash)'
?>
```

* Assuming that this is a database that actually hashes its passwords

SQL Injection

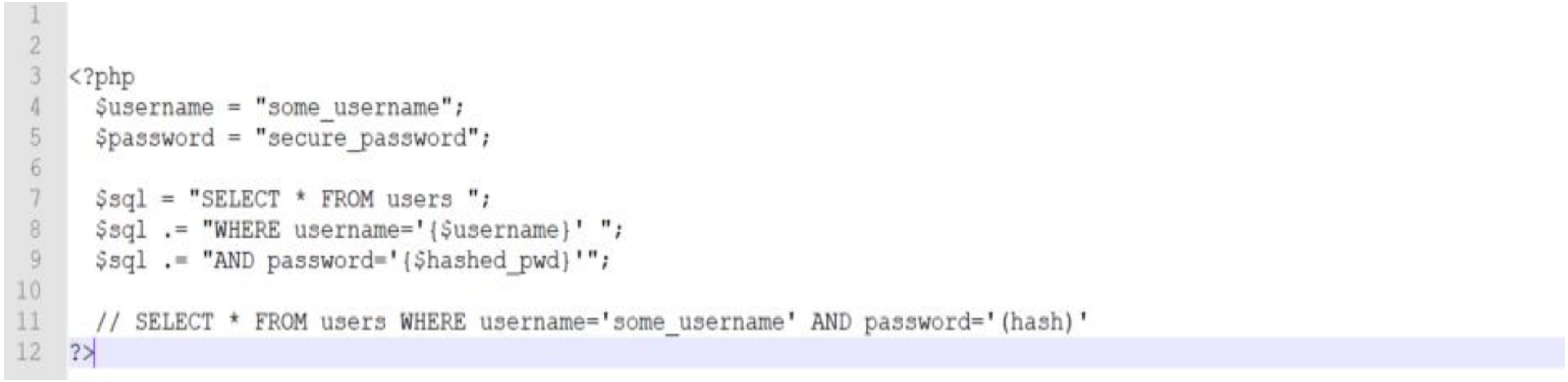
- SQL Injection (SQLI) is when untrusted data is used to construct an SQL query. The data is inserted (or "injected") into the SQL query string.”
- Ranked as #1 threat by OWASP
    - Easy for attackers to exploit

Why is this relevant?

Put simply…
It’s dangerous

Around July 2012, a hacker group was able to use a “union-based SQL injection” in order to retrieve passwords that were stored in **plaintext**, compromising ~450,000 login credentials.

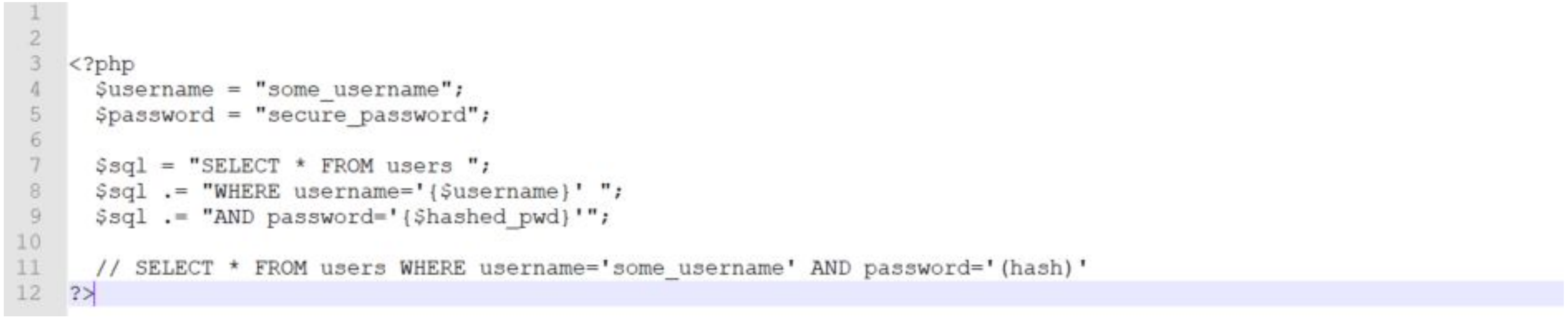
Yahoo reportedly hacked: Is your account safe?



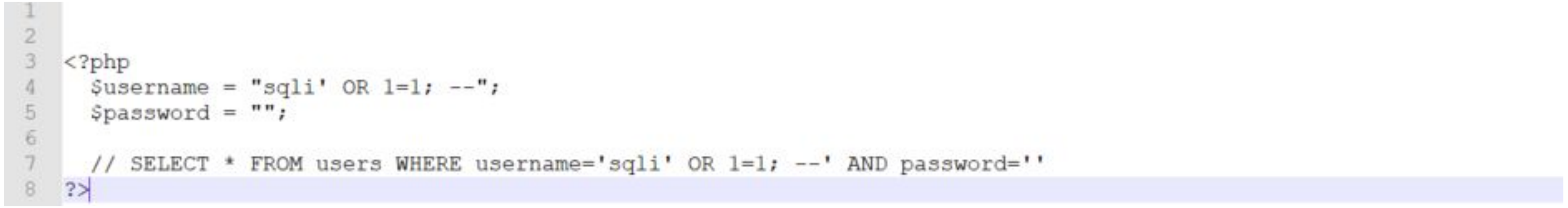
```


<?php
  $username = "some_username";
  $password = "secure_password";

  $sql = "SELECT * FROM users ";
  $sql .= "WHERE username='{$username}' ";
  $sql .= "AND password='{$hashed_pwd}'";

  // SELECT * FROM users WHERE username='some_username' AND password='(hash)'
?>
```

Recall previous SQL

```
SELECT * FROM users WHERE username=‘$username’ AND password=’$hashed_pwd’
```

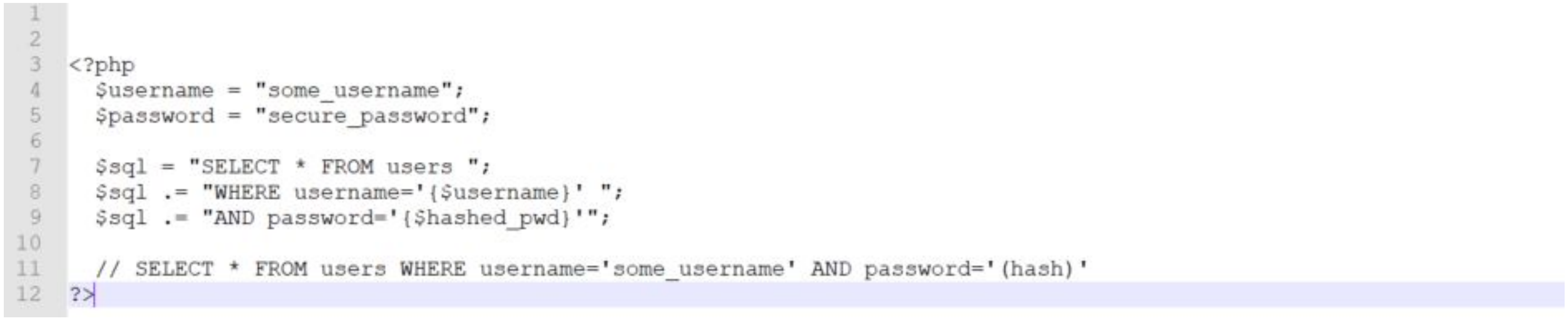


```php


<?php
  $username = "some_username";
  $password = "secure_password";

  $sql = "SELECT * FROM users ";
  $sql .= "WHERE username='{$username}' ";
  $sql .= "AND password='{$hashed_pwd}'";

  // SELECT * FROM users WHERE username='some_username' AND password='(hash)'
?>
```

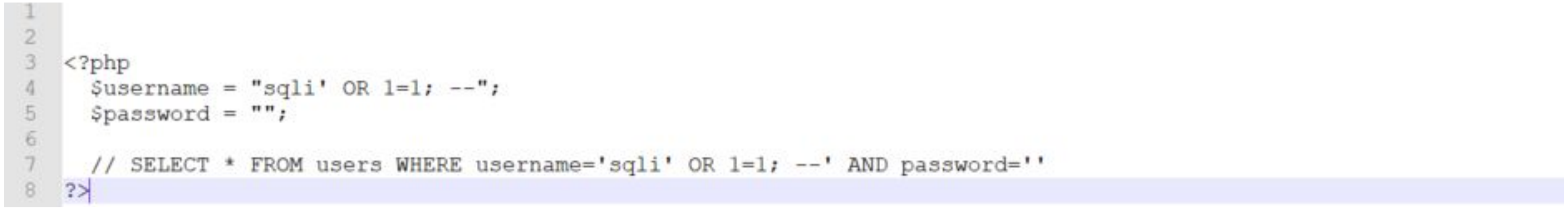


```php


<?php
  $username = "sqli' OR 1=1; --";
  $password = "";

  // SELECT * FROM users WHERE username='sqli' OR 1=1; --' AND password=''
?>
```

Do you see the vulnerability?



```


<?php
  $username = "some_username";
  $password = "secure_password";

  $sql = "SELECT * FROM users ";
  $sql .= "WHERE username='{$username}' ";
  $sql .= "AND password='{$hashed_pwd}'";

  // SELECT * FROM users WHERE username='some_username' AND password='(hash)'
?>
```



```


<?php
  $username = "sqli' OR 1=1; --";
  $password = "";

  // SELECT * FROM users WHERE username='sqli' OR 1=1; --' AND password=''
?>
```

There’s a tautology!

How do you know what injections to use?

You don’t…
trial and error!

Techniques

- Try enter just “ ’ ” and see what happens
- Try using ” instead of ’ to close off your queries
- Don’t forget about “;” and “--”
- Look up SQL Cheat Sheets
- Don’t get frustrated… it’s tough!

Lab 2

Code Injection