

EECS 388: Lab 4

Review: Project 1 Part 1

Introduce Project 2 (Web Security)

SQL Introduction

SQL Injection

Current Assignments

- Lab 2: Available Now (Thursday, Sept 7!)
 - **Due Thursday, Sept. 26 at 6 PM**
- Project 2: Web Security
 - Released: Thursday, Sept. 19
 - **Due Thursday Oct. 3 at 6 PM**
 - Coverage:
 - SQL Injection
 - CSRF Attack
 - XSS Attack

Partners are optional!

Reminder: Weekly Canvas Quizzes

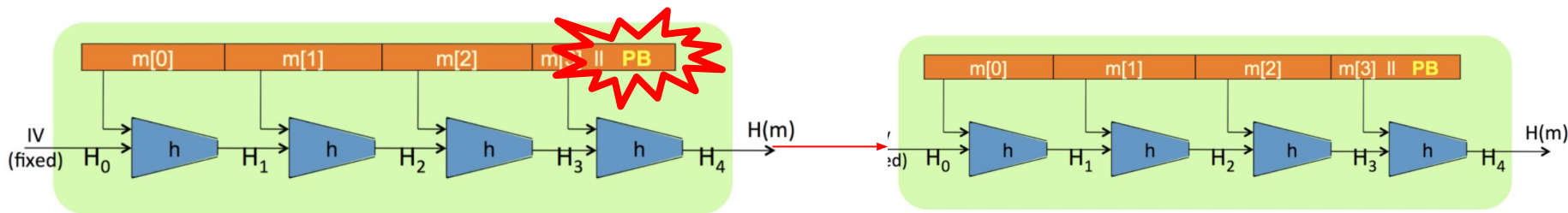
Review:

Project 1, Part 1

“Key” Takeaways

Length Extension

- Padding from original message must remain in the extended message!



Hash Collisions

- Differentiate blobs using their SHA-256 hashes!

```
sha256(blob.encode("latin-1")).hexdigest()
```

→ `print("Use SHA-256 instead!")`
→ `print("MD5 is perfectly secure!")`

Intro to Web Project

Project 2 Preview

- Part 1: SQL Injection Attacks
 - Lecture 8, today's lab!
- Part 2: XSS Attacks
 - Lecture 8, next week's lab
- Part 3: CSRF Attacks
 - Lecture 8, next week's lab
- Skills you'll need
 - Basic JavaScript, HTML, knowledge of DOM, basic SQL (Lecture 7)
 - Using basic webdev tools (Lab Assignment 2)
 - Running a basic Python webserver (Project 2 spec)
 - Ability to Google basic questions as needed



But Wait!

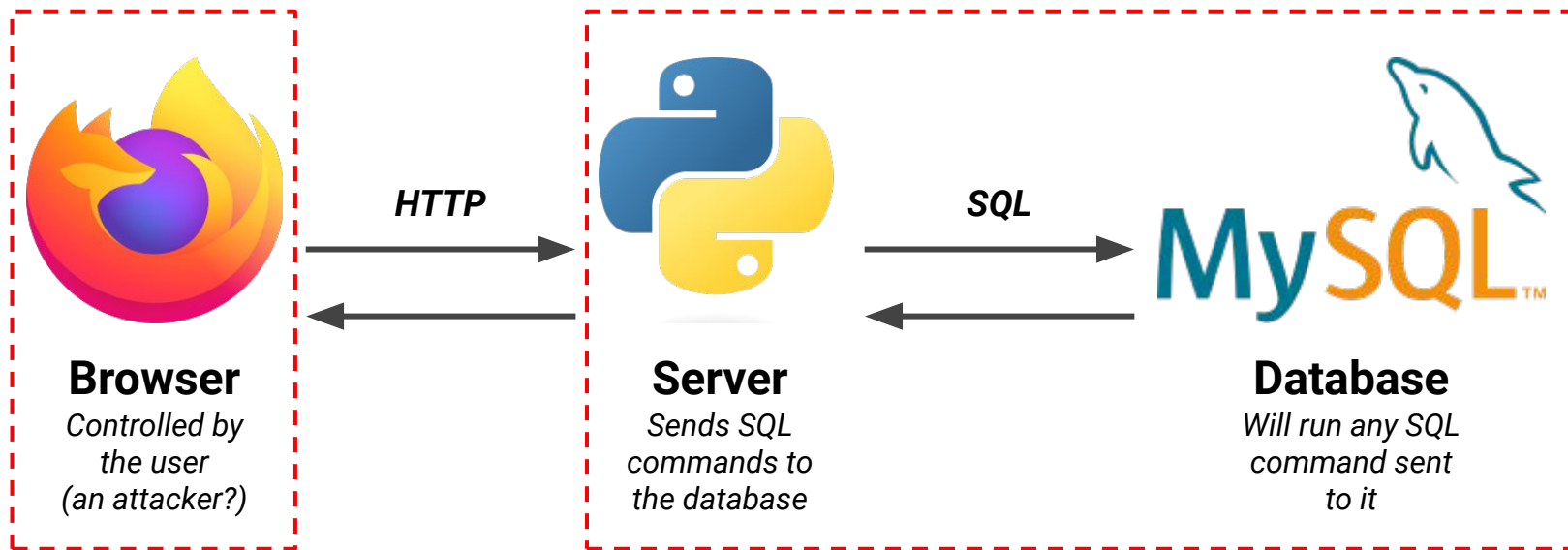
***NEVER* use an important
password for an account on
an insecure site!**

(Also, never reuse a password at all)



SQL 101

Databases



Very important that users (attackers) can't send any command that they want to the database!

Structured Query Language (SQL)

- Allows you to interact with a database
- Can store and retrieve data
- Built to look kinda (???????) like English

users

username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000
danlliu	x1w41dd5\$	05-10-2000

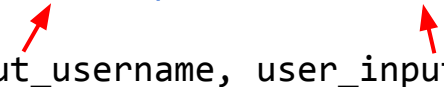
`SELECT * FROM users WHERE username = "danlliu"`

(Get all data rows from the table "users" where the "username" column contains exactly "danlliu")

Good SQL tutorial by W3: <https://www.w3schools.com/sql/>

Example Code - SQL in Python using SQLite3

```
import sqlite3
...
con = sqlite3.connect('user.db')
sql_statement =
    "SELECT birthday FROM users WHERE username = ? AND password = ?"
result = con.execute(sql_statement, (user_input_username, user_input_password))
if result is not empty:
    # print the user's birthday
```



- SQL = Blue
- SELECT = choose the column
- birthday = the attribute to be selected
- FROM = choose the table
- WHERE = specify logical conditions
- AND = logical operator that requires both conditions to be true



Example Code - SQL in Python

...

```
sql_statement =  
    "SELECT birthday FROM users WHERE username = ? AND password = ?"  
result = sql_execute(sql_statement, (user_input_username, user_input_password))  
if result is not empty:  
    # print the user's birthday
```

users

username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000
danlliu	x1w41dd5\$	05-10-2000

{leslie, haxor}

{danlliu, x1w41dd5\$}

no user found

05-10-2000

SQL - Select Statement

```
SELECT [col1, col2, col3] FROM table_name [additional conditions];
```

- * can be used in the place of col_names to select all columns
SELECT * FROM table_name;
- SELECT birthday FROM users WHERE username = 'leslie' AND password = 'haxor';
- SELECT birthday FROM users WHERE username = 'danlliu' AND password = 'x1w41dd5\$';

users

username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000
danlliu	x1w41dd5\$	05-10-2000

{leslie, haxor}

{danlliu,x1w41dd5\$}

no user found

05-10-2000

SQL - Delete Statement

`DELETE FROM table_name WHERE [condition];`

- `DELETE FROM users WHERE username = 'danlliu';`

users

username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000
danlliu	x1w41dd5\$	05-10-2000



users

username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000

SQL - Insert Statement

```
INSERT INTO table_name (col1, col2, ...) VALUES(val1, val2, ...);
```

- (col1, col2, ...) not necessary, but will assume you provide values for all columns
- Columns that are NOT NULL must be assigned a value
- `INSERT INTO users VALUES('danliu', 'x1w41dd5$', '05-10-2000');`

users		
username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000



users		
username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000
danliu	x1w41dd5\$	05-10-2000

SQL - Installing SQLite + Demo

Project 2 uses MySql

- [SQLite Install](#)
- [Online SQLite](#)
- [SQLite Syntax](#)

users

username	password	birthday
chrdeng	b1234	01-31-2016
iallada	password	06-11-2000
danlliu	x1w41dd5\$	05-10-2000

SQL Injections

A Main Vulnerability: Data vs. Code

- What is the difference?
 - The context in which they are used
 - You can treat a file of source code as just a long string, and vice versa
 - If we can confuse the context, we can get data to be executed as code
 - SQL Injection
 - XSS



SQL Injections - Example Code

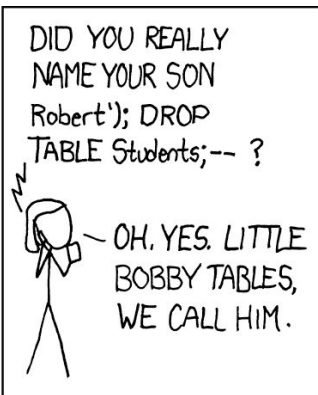
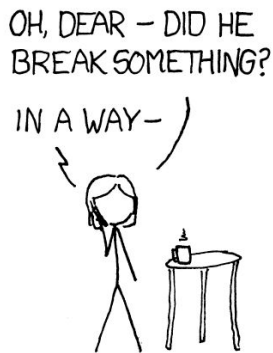
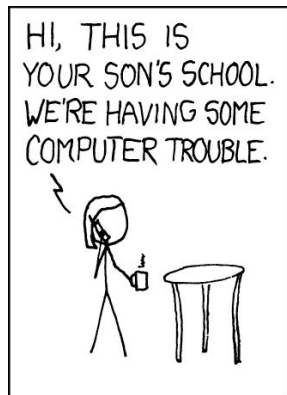
Not using prepared
statements 😱

```
...
sql_statement =
    "SELECT * FROM users WHERE username = '" + user_input_username +
    "' AND password = '" + user_input_password + "'"
result = con.execute(sql_statement)
if result is not empty:
    # let the user log in to the site
```

- * means to select all
- Double quotes - surround the SQL code snippet
- Single quotes - surround the user input within the SQL code
- Where is the vulnerability? How could an attacker exploit this?

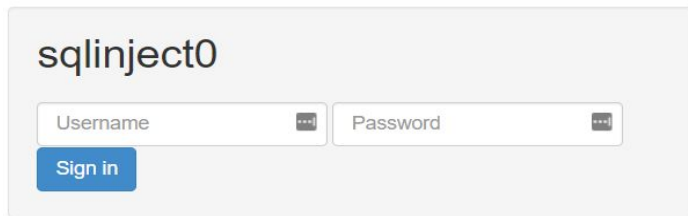
SQL Injections: Exercise

- Navigate to <https://www.hacksplaining.com/exercises/sql-injection>
- Complete the demo
- Let us know if you have questions!



Project 2: SQL Injections

- Manipulate the SQL by getting your string input to be interpreted as SQL
- Do not need to be in Docker's Firefox browser for this part
 - <https://project2.eecs388.org/sqlinject/0>
 - Can change defense level with number at end
- Situation
 - You are breaking into the account of username victim
 - Come up with a password that will allow you to manipulate the SQL behind the scenes



The image shows a web application interface for 'sqlinject0'. It features a light gray rectangular box containing the title 'sqlinject0' in a bold, black, sans-serif font. Below the title are two input fields: 'Username' and 'Password'. Each field has a small, dark gray icon to its right, likely representing a password toggle or a search function. Below the 'Username' field is a blue button with the text 'Sign in' in white. The background of the slide is white, with a decorative geometric pattern of pink and red triangles in the bottom right corner.

Real Examples of SQL Injections

- SQL Injection Attack on Heartland Payment Systems, 7-Eleven
 - <https://www.wired.com/2010/03/heartland-sentencing/>
- GhostShell SQL Attacks
 - <https://www.darkreading.com/attacks-breaches/ghostshell-haunts-websites-with-sql-injection>
- 2016 Russian-government Attacks Against Voter Registration Systems
 - <https://rollcall.com/2019/04/22/mueller-report-russia-hacked-state-databases-and-voting-machine-companies/>



Lab Assignment 2

Useful for Project 2!

Lab 2 Information and Spec Walkthrough

- Lab 2 seeks to gently introduce you to:
 - The Firefox GUI, you'll need this for Project 2
 - Dev Tools, this will assist you in completing the XSS / CSRF attacks
 - JavaScript
- Quick walkthrough of the Lab 2 Spec!



See you next week!

