

# Web Exploitation

An Introduction

# WEB EXPLOITATION WORKSHOP

Location : ILC S311  
Date : Tuesday 24th  
Sept, 2024 @ 7pm



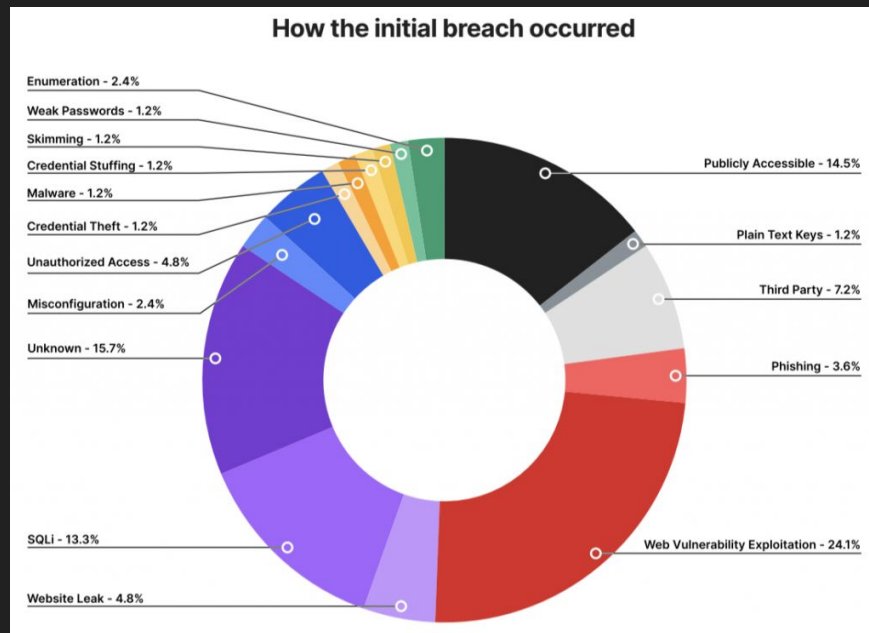
# Disclaimer!

H4ck1ng is 1lleg4l 4nd b4d

**Don't do this stuff without  
explicit permission. YOU WILL  
GET CAUGHT...**

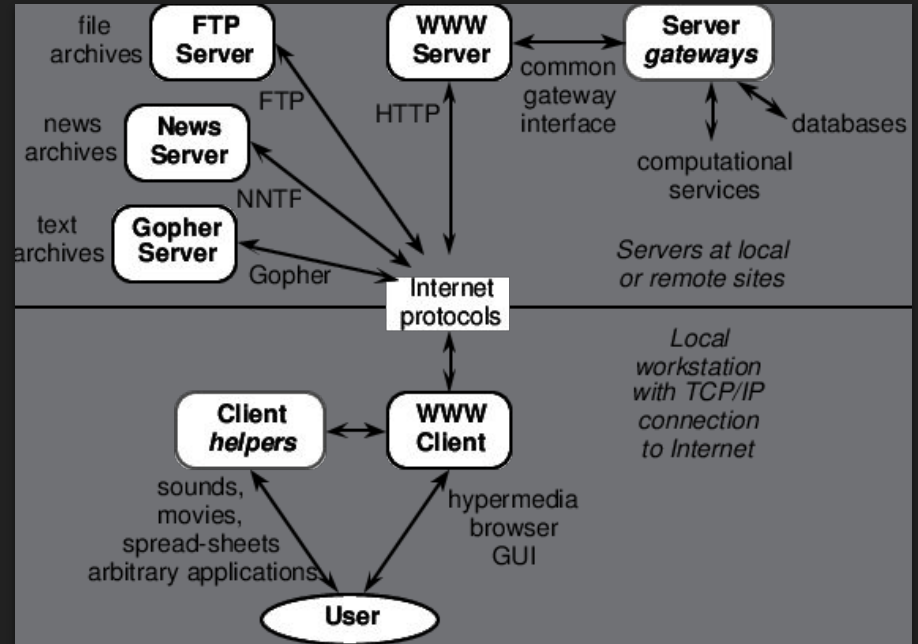
# What is Web Exploitation?

- Finding and exploiting vulnerabilities in web-based application
- Some common web vulnerabilities:
  - SQL Injection
  - Cross Site Scripting
  - Local File Inclusion
  - **Command Injection**
- Like the web itself, it can feel like a complicated mess
  - Don't worry! Everybody starts somewhere



# What is the Web?

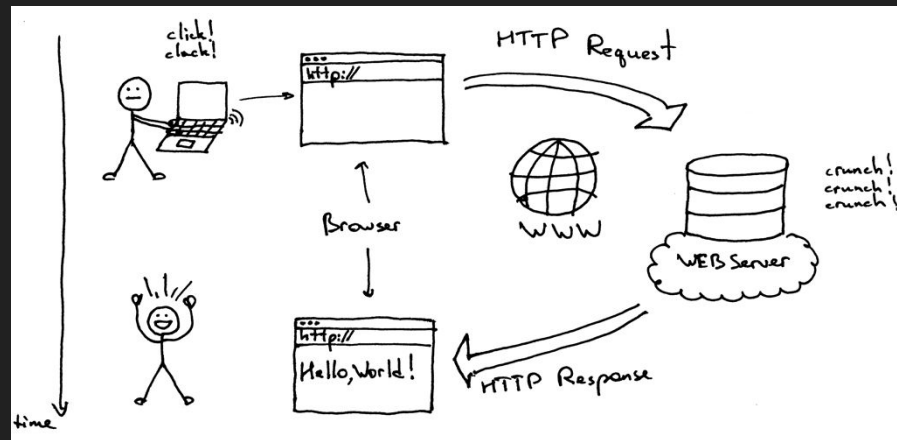
- **World Wide Web (WWW):** “an interconnected system of public webpages accessible through the Internet” - [mdn web docs](#)
- The web is NOT the internet, it is just an application built on top of the internet
- Protocols
  - A standard (“language”) for communication between computers
- If you want to learn more about the internet take CS 453!



# Client-Server Model

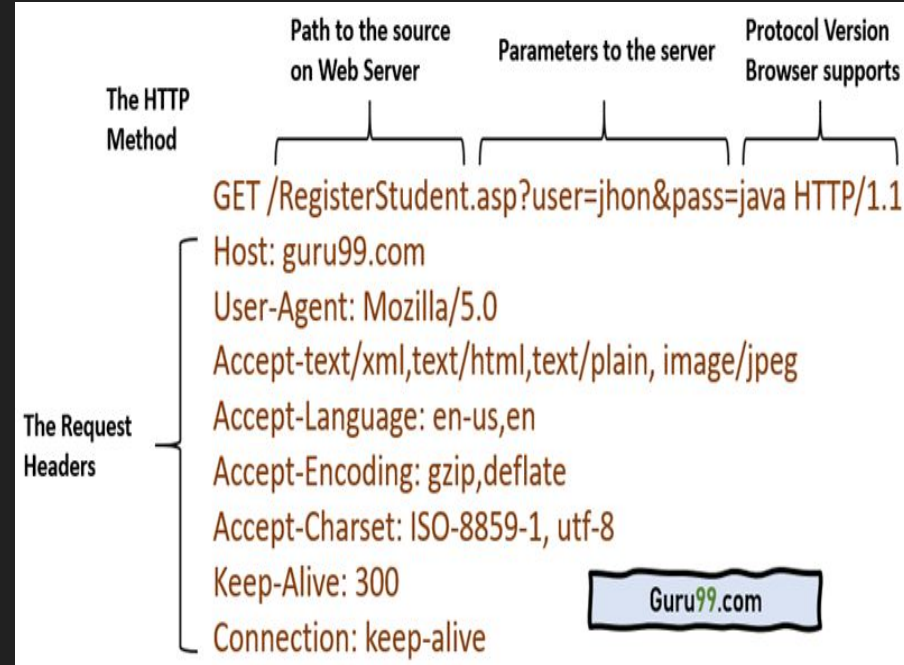
- **Client:** System/program that connects to a remote server to retrieve content
  - For most users: the browser
- **Server:** A local or remote system that provides data to a user
  - Can be local or remote
- You can set up a local file server of your own using:

`python3 -m http.server`



# Hypertext Transfer Protocol (HTTP)

- **HTTP:** A special protocol designed for communicating between web client and servers
  - Follows the client server model we mentioned earlier
- We send an HTTP request from our client and receive a response from the server
- An HTTP request consists of:
  - **HTTP Method:** GET, POST, etc..
  - **URI:** what location you're requesting
  - **HTTP version:** HTTP 1.1/2
  - **HTTP Request headers**
    - Other pieces of helpful information



# HTTP Headers - How are we requesting our data.

- Headers are sent by both the server and client: to tell the client and the server information about each other
- Useful Request Headers:
  - **Cookie:** Cookies are used for tracking and/or authenticating users
    - We can edit and resend cookies to exploit logic bugs in developer code!
  - **Host:** What server are we requesting a resource on
  - **Content-Type:** Ensures our content is sent correctly to the server, examples are JSON, form-data, etc...
- A response header to look for:
  - **Server or x-powered-by:** Tells us what server is running in the background, useful for researching vulnerabilities
  - Read more about security with response headers [here](#)

# HTTP methods - What are we requesting?

Common HTTP Request Method	Meaning
<b>OPTIONS</b>	Requests the server to tell us the available methods on an endpoint
<b>GET</b>	Requests the resource from a filename provided on the host, we receive the content in the body
<b>HEAD</b>	Same as GET, but body is not given
<b>POST</b>	Submits an entity to the specified resource, often causing a change in state or side effects on the server

- Can read more about different headers [here](#)



# Burp Suite and HTTP Proxies

- Burp Suite is a pentesting tool to find, enumerate, and exploit vulnerable web applications
- Burp Suite is a proxy that receives all HTTP/s traffic on local port 8080 and forwards or intercepts based on settings on our settings



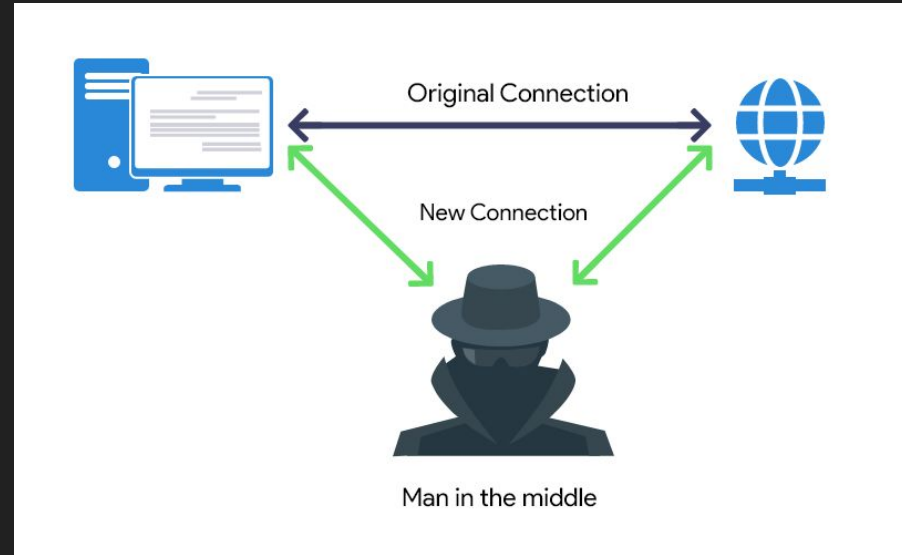
# Burp Suite and HTTP Proxies

- Today we're primarily going to use three features of Burp Suite:
  - **Target:** Let's us see the sitemap of a website we visit: this sitemap will grow as we visit more of the website
  - **Proxy:** Our proxy will intercept all HTTP requests it receives where we can the see and modify the contents while they are being sent
  - **Repeater:** We can send any HTTP request we intercept to repeater where we have a view of our request and response to debug an endpoint



# HTTPS - A Side Note

- You may have noticed most websites you connected to are using HTTPS
- This is a more secure version of HTTP encrypted using Transport Layer Security (TLS)
- This is used to prevent man in the middle attacks (MITM)
- You won't need to for the demo, but if you want Burp Suite to work over HTTPS you may need to follow this [tutorial](#)




Let's intercept a request with Burp Suite.

# HTTP Status Codes - How is the Server Responding?

- Typically only seen in requests, for HTTP responses we get a status code:

Status Code	Meaning
200	OK
301	Moved permanently
302	Found
400	Bad/Invalid request

Status Code	Meaning
403	Unauthorized
404	Not found
418	I'm a teapot 
500	Internal Server Error

- Sometimes we can find exploits because improper methods were allowed

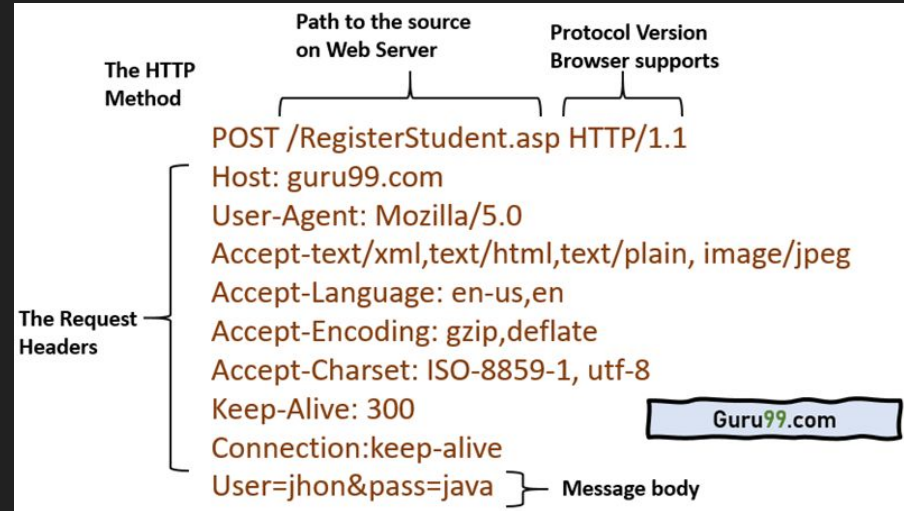
You can now try Challenges 1 and 2.

<https://training.umasscybersec.org>



# HTTP Body - What is the data we are sending?

- With **POST** requests we can send data to the server via our HTTP body
  - Below all our headers of a post request we can add a new line and begin adding data
  - This data can be in multiple forms, we can specify what form in our request with the
- We can also send data to the server with **GET** requests using URL Parameters
  - [https://www.google.com/search?q=hello\\_world](https://www.google.com/search?q=hello_world)



# Command Injection

What's wrong with the following code? Discuss it with the people around you.

```
import os

def index(user_input):
    #Operating system executes ping -c 3 user_input
    return os.popen(f'ping -c 3 {user_input}').read()
```



# How can we run multiple commands without a new line? Discuss with the people around you!

```
import os
def index(user_input):
    #Operating system executes ping -c 3 user_input
    return os.popen(f'ping -c 3 {user_input}').read()
```

# How can we run multiple commands without a new line?

- `echo "hi" ; ls`
- `echo "hi" | ls`
- `echo "hi" && ls`
- We can find a cheat sheet [here](#)

You can now attempt Challenge 3.



## What now?

- We will have future talks (this friday and more!), diving deep into web exploits
- In the meantime, you can:
  - Try the web challenges on our [Training Platform](#)
  - [Play PicoCTF](#)
  - [Try the PortSwigger labs](#)
  - Play weekly CTFs with the club!

# Hivestorm 2024



- Cyber defense competition
- Teams of 4 (+2 Alts)
- Virtual competition open to all students with no team limit per school - Great way to get experience!
- Auditing a simulated corporate network for security issues
- Applications due Sept 27, more info in our Discord.
- Competition held Wednesday, October 16

<https://www.hivestorm.org/event.html> - Info

<https://forms.gle/BViUNekAobFckLNh7> - Signup

