LAB: Cross Site Request Forgery (CSRF) Attack (Ubuntu 20.04 VM)
****

HTTP Background Information
****

Same-site request: request to a website comes from one of its own pages

Cross-site request: request to a website comes from a page of a different website

Cross-site requests have many uses:
- a webpage from one website embeds an image from another website
- homepage on GVSU website with a link to Facebook, Twitter, Instagram, etc.
- Websites with online advertisements
- a mashup website

Without cross-site requests, each website would just display its own pages and would have no ability to connect with other websites.

Browsers know whether a request is a cross-site or not based on which page a request is initiated from, but they do not convey that information (using the "Referrer" HTTP header in the HTTP request) to the server due to privacy concerns.

Cookies: An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with later requests to the same server. Cookies help websites remember stateful information using a protocol that is stateless (HTTP). An example usage of cookie is the session ID, a unique ID assigned to the user by a web server.

Cookies are maily used for:
- tracking
- session management (a series of HTTP transactions from one client)
- personalization

The browser attaches all the cookies belonging to a website whether a request to the site is same-site or cross-site. The website cannot tell the difference between a cross-site request and a same-site request unless the browser or the website uses other approaches.

HTTP Messages: how data is exchanged between a server and a client
(client is typically a Web browser)

HTTP Requests - messages sent by the client to initiate an action on the server

HTTP Responses - server responses to client's requests

HTTP requests and responses share similar structure and are composed of:
- Start line (a single line)
- Optional set of headers
- A blank line separating meta-information and the body
- Optional body

Two ways to inspect HTTP requests and responses in Firefox browser:

1. Using the "HTTP Header Live" add-on

2. Web Developer Tools
   Open application menu -> More tools -> Web Developer Tools -> Network tab

CSRF Background Information
****

In a CSRF attack, a victim (with an active authenticated session with the target website) is tricked into viewing an attacker's webpage. While viewing the attacker's webpage, a forged cross-site request is sent to the target website without victim's knowledge. The target website processes the forged request as if it was submitted by the authenticated user.

A CSRF attack executes unwanted actions on behalf of a user on a website where the

user is already authenticated.

CSRF attacks exploit "the trust that a website has in a user's browser" as the user is authenticated and has an active session with the target website.

CSRF attacks target state-changing requests (such as funds transfer, adding a friend and editing a user's profile on a social network platform, ...).

A successful CSRF attack causes unauthorized state changes for a victim user.

Cross-Site Request Forgery (CSRF) attack is also known as session riding or one-click attack.

CSRF attacks require the following:

1) A target website with CSRF vulnerability
2) A victim user with an active session with the target website
3) A malicious website that is controlled by an attacker
4) A web page (crafted by the attacker) on the malicious website that injects a forged cross-site HTTP GET/POST request to the target website into victim's user session
5) The attacker needs to attract the victim user to visit the crafted web page on the malicious website (usually via an email/message sent to the victim with a link to the URL for the malicious web page)

Defending against CSRF attacks:

To defend against CSRF attacks, a website must make sure that every state-changing request comes from one of its own pages. This can't be done if a website relies just on cookies (which typically include a unique session ID set by the server) attached to the request sent by a client because cookies are attached to same-site as well as cross-site requests.

CSRF Lab Instructions
****

Lab files:
    Web_CSRF_Elgg.pdf (lab specifications - do not use this file)
    CSRF-Lab-Instructions.pdf (use this file to do the lab)
    Labsetup.zip (files required for docker containers)
    CSRF.pdf

Lab Setup and Information:

1. Download CSRF-Lab-Files.zip file from Blackboard to home folder on SEED VM.
    Unzip the file.

2. Open Terminal app from the sidebar.

    It is useful to have two terminal windows (as two tabs in the Terminal app) open.
    I will refer to these tabs as "Terminal 1" and "Terminal 2" in this document.

    Terminal 1: is used to run the docker containers for this lab
    Terminal 2: is for all other uses

3. From Terminal 1: open /etc/hosts file on the VM using the following command:

    $ sudo gedit /etc/hosts

    Delete the lines from 11 to end of the file and add the following lines/mappings to the file starting at line 11.

    # For SQL Injection Lab
    10.9.0.5   www.seed-server.com

    # For XSS Lab
    10.9.0.5   www.seed-server.com
    10.9.0.5   www.example32a.com
    10.9.0.5   www.example32b.com

```
    10.9.0.5        www.example32c.com
    10.9.0.5        www.example60.com
    10.9.0.5        www.example70.com

    # For CSRF Lab
    10.9.0.5        www.seed-server.com
    10.9.0.5        www.example32.com
    10.9.0.105      www.attacker32.com
```

Save the file and exit the editor.

4. From Terminal 1:

To use the "HTTP Header Live" add-on, we should update Firefox Browser in the SEED VM.

Enter the following two commands at the command prompt ($):

```
$ sudo apt update           # this will couple of minutes

$ sudo apt install --only-upgrade firefox       # this will take couple of minutes

$ clear

// now install the "HTTP Header Live" add-on in Firefox
```

Start Firefox browser
Open application menu -> Add-ons and themes
Enter "HTTP Header Live" in the finder/search box
Click "HTTP Header Live" item, click "Add to Firefox" and "Add" to install it

// add "Show sidebars" button to the browser toolbar
Right click on the browser toolbar and select "Customize Toolbar..."
Drag "Sidebars" item to the browser toolbar and drop it

NOTE: If "Software Updater" window opens and asks you to install updates, just cancel.

5. From Terminal 1:

Make sure you are in the CSRF-Lab-Files folder.
Unzip Labsetup.zip file using the command: unzip Labsetup.zip
This will create a folder named Labsetup.

This lab uses a social networking web application called Elgg.
Web application URL is http://www.seed-server.com (after we start the docker containers)

This web application is purposely made to be vulnerable to CSRF attacks (certain built-in countermeasures in the app are temporarily turned off for this lab).

This lab is setup using three docker containers:
- one for hosting the Elgg website (vulnerable website)
- one for hosting the attacker's website (malicious website)
- one for hosting the MySQL database for Elgg application

In the Elgg container, the code for the vulnerable website is in folder /var/www/elgg (don't look for it now - we haven't started the docker container yet!)

In the Attacker container, the code for the website is in the folder /var/www/attacker (don't look for it now - we haven't started the docker container yet!)

6. From Terminal 1: Change to Labsetup folder, remove any existing docker containers and images, and start the docker containers using the commands below:

```
$ cd Labsetup   // to move to the Labsetup folder in the CSRF-Lab-Files folder

$ docker rm -vf $(docker ps -aq)        // remove any running containers
                                        // ignore any errors/warnings

$ docker rmi -f $(docker images -aq)    // remove any docker images
                                        // ignore any errors/warnings
```

```
    $ dcup      // download and start three docker containers
                // CAUTION: "may" raise errors if this command is run with VPN turned on
                // (on my laptop, this command caused errors with VPN on; it worked
                //  when I turned off the VPN)

    You will see some output generated and three docker containers started.
    It will take few minutes to download the docker images and start the containers.
    So, be patient!!

    After minutes of downloading and installing, you should see the following text
    in Terminal 1 window:
    "3306  MySQL Community Server - GPL."

    You don't have to do anything within this window. Just leave this window alone.

7.  From Terminal 2: you can verify the three docker containers are running with
    the command:

    $ docker ps       // you should see three docker containers running


Attack Task 1: Observing HTTP Requests using "HTTP Header Live" add-on in Firefox
*********************************************************************************

Start Firefox browser in the SEED VM

Clear browser history:
    Open application menu -> Settings -> Privacy & Security -> History
    Click "Clear History..." and click "Clear" button

While we are on the Privacy and Security page, let's do the following as well:
    Scroll to the top of page
    Click on "Manage Exception..."
    Add "www.attacker32.com" in the "Address of website" box
    Click "Add Exception" and "Save Changes"
    Close this browser tab

The Elgg website (www.seed-server.com) is configured with the following user accounts
(username and password) for the lab tasks:

        admin       seedadmin
        alice       seedalice
        boby        seedboby
        charlie     seedcharlie
        samy        seedsamy

Let's display "HTTP Header Live" add-on:
    Click on "Show sidebars" button on the top right corner in the browser
    A sidebar opens on the left side of the browser window
    Select "HTTP Header Live" from the dropdown, if it is not already selected
    Uncheck "autoscroll" checkbox at the bottom of this window
    You may also want to use the "Clear" button before each action on a web page

Vist www.seed-server.com and inspect HTTP GET request(s) generated using the
"HTTP Header Live" tool.

Click "Clear" button in the "HTTP Header Live" to clear the window contents.

Login to Alice's account and inspect HTTP POST request sent to server
    username: alice     password: seedalice

Logout from Alice's account.

You may close "HTTP Header Live" window now. Just note that "HTTP Header Live" can
be launched anytime to observe the HTTP request(s) sent to the web applicaiton.
```

Attack Task 2: CSRF Attack by Forging HTTP GET Request
****

Elgg website (with CSRF vulnerability): http://www.seed-server.com

Attacker's website: http://www.attacker32.com

For this lab,

    Victim: Alice (username: alice   password: seedalice)

    Attacker: Samy (username: samy   password: seedsamy)

Attack scenario: Samy wants to become a friend to Alice, but Alice refuses to
add him as her friend. So, Samy decides to use CSRF attack to achieve his goal.

First, let's see what a legitimate "Add Friend" HTTP GET request looks like.

- Display "HTTP Header Live" add-on. It is useful to clear the contents of this add-on
  before each action on the website. Uncheck "autoscroll" checkbox.

  FYI: Sometimes, it is helpful to hide the display of this add-on to see whole page
  of the site and all menu options available to the user on the website.

- Samy logs into his account on www.seed-server.com and searches for "Alice" and
  clicks on Alice

- Samy adds Alice as a friend to see what "Add Friend" HTTP GET request looks like:

    http://www.seed-server.com/action/friends/add?friend=56

- What is ID (known as GUID in Elgg) of Alice?
    - Look at the value of "friend" parameter in the HTTP GET request above
    - It is 56

- But, we need ID/GUID of Samy if we want to make Alice add Samy as her friend:
    - View the page source of any page (when Samy is logged in)
        - Right click on the page and select "View Page Source"
    - Look for "guid" or "owner_guid" in the <script> section at the bottom
      of the page source
    - You will notice the Samy's ID/GUID is 59
    - Close the tab

Next, Samy constructs a malicious web page (addfriend.html) and uploads the page
to malicious website at www.attacker32.com. This is already done for you (see the
note below).

The addfriend.html file is already placed in the right folder in the attacker's
website when the docker container was started. You can view the addfriend.html file
in the CSRF-Lab-Files/Labsetup/attacker folder.

Samy then sends the following message to Alice to trick her to click on the URL in the
message to the malicious webpage (http://www.attacker32.com/addfriend.html).

    Message Subject: Win Free Electronic Gadgets

    Message Body:alice

    Hi Alice,

    You can win a free iPad by clicking on this link: http://www.attacker32.com/addfriend.html

    - Samy

Samy logs out.

Alice logs in, sees the message from Samy, and clicks on the URL in the message hoping
to win a free electronic gadget.

CSRF attack is launched and Samy is added to Alice's friends list.

Visit Alice's friends list to verify if Samy is added as a friend.

Alice logs out.

Attack Task 3: CSRF Attack by Forging HTTP POST Request
****

Samy wants to do something more interesting using CSRF attacks.

Samy wants to edit Alice's profile to say "Samy is my Hero!!"

Samy logs into his account.

Samy edits his own profile to place "I love SEED projects" in the "Brief description" field to inspect the HTTP POST request sent to the website www.seed-server.com

    http://www.seed-server.com/action/profile/edit

Next, Samy constructs a malicious web page (editprofile.html) and uploads the page to malicious website at www.attacker32.com. This is done for you (see the note below).

The editprofile.html file is already placed in the right folder in the attacker's website when the docker container was started. You can view the editprofile.html file in the CSRF-Lab-Files/Labsetup/attacker folder.

Samy then sends a message to Alice to trick her to click on the URL in the message to the malicious webpage (http://www.attacker32.com/editprofile.html).

    Message Subject: Chance to win a free trip to Paris

    Message Body:

    Hi Alice,

    To win a trip to Paris for free, visit: http://www.attacker32.com/editprofile.html

    - Samy

Samy logs out.

Alice logs in, sees the second message from Samy, and clicks on the URL.

CSRF attack is launched and Alice's profile is changed to say "Samy is my Hero!!".

Look at Alice's profile now to verify the change made to her profile.

Alice logs out.

Task 4. Defense: Enabling Elgg's Countermeasure
****

Let's now turn on the countermeasure by removing/commenting the following statement in the validate() function in "Csrf.php" file in the Elgg web application:

    return;   # Added for SEED Labs (disabling the CSRF countermeasure)

file path: /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security/Csrf.php

Do the following to accomplish this task:

From Terminal 2, start a shell on the Elgg container using this command:

$ docker ps   # to get a list of running containers

$ docksh elgg-10.9.0.5   # start a shell on Elgg container

cd /var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security/

```
# nano Csrf.php     // start nano editor

    - scroll to validate() function and place // in front of return; statement
    - save the file by CTRL^O and press enter
    - exit the editor by CTRL^X

# exit      // to exit the container shell
```

The countermeasure is based on two fields (__elgg_token and __elgg_ts) that Elgg website generates randomly and uniquely for every user and adds them to each page it sends to the client and checks for their presence in each request made to the website from that page.

If these two value are present in the request and match with what is sent to the client, then the request is a same-site request while its absence signifies that the request is cross-site.

Login to Alice account and do the following:
- Remove Samy from friends list
- Edit the Alice's profile to clear the "Brief description" field

Next, Alice visits her messages box and clicks on the following link:

http://www.attacker32.com/addfriend.html

You will notice error message(s): "Form is missing __token or __ts fields"

Check Alice's friends list to confirm that Samy is not added as a friend.

Next, Alice visits her messages box and clicks on the following link:

http://www.attacker32.com/editprofile.html

You will notice error message(s): "Form is missing __token or __ts fields"

Check Alice's profile to confirm it does not say "Samy is my Hero!!"

After verifying that the attak has failed, logout and close the browser.

Task 5: Experimenting with the SameSite Cookie Method

****

Read Section 4.2 in the lab on "Task 5: Experimenting with the SameSite Cookie Method".

From SEED Ubuntu VM, open Firefox browser and visit http://www.example32.com

Click on "Link A" on the main page and experiment with sending GET and POST requests.
- Take time to understand the output produced

Click on "Link B" on the main page and experiment with sending GET and POST requests.
- Take time to understand the output produced

Close Firefox

Clean up and wrap up the lab

****

1. In Terminal 2:

```
    $ cd ~/CSRF-Lab-Files/Labsetup   // to make sure you are in the Labsetup folder
    $ docker ps         // shows the list of docker contains running
    $ dcdown            // to stop and remove the docker containers
    $ docker rmi -f $(docker images -aq)    // remove the docker images
    $ exit              // to close terminal 2 window
```

2. In Terminal 1:

```
    $ exit              // to close terminal 1 window
```

********** END OF CSRF ATTACK LAB **********