CSC 223

Title and Experiment #	Lab 5: Clickjacking
Name	Gianna Galard
Date Performed	20-Nov-22
Date Submitted	20-Nov-22

The student pledges this work to be their own *Gianna Galard*

Overview:

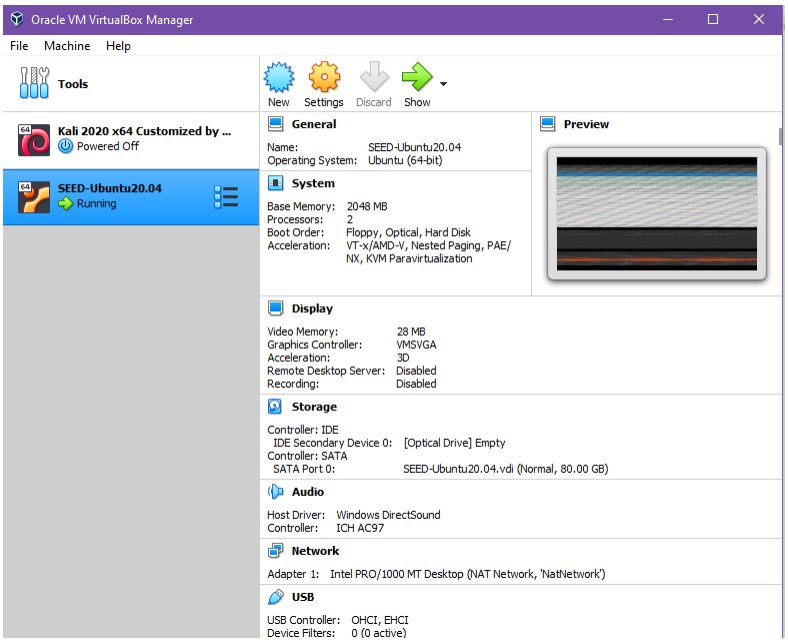
Clickjacking, also known as a “UI redress attack,” is an attack that tricks a user into clicking on something they do not intend to when visiting a webpage, thus “hijacking” the click. In this lab, we will explore a common attack vector for clickjacking: the attacker creates a webpage that loads the content of a legitimate page but overlays one or more of its buttons with an invisible button that triggers malicious actions. When a user attempts to click on the legitimate page’s buttons, the browser registers a click on the invisible button instead, triggering the malicious action.

This lab covers the following topics:

• Clickjacking attack
• Countermeasures: frame busting and HTTP headers
• Iframes and sandboxes
• JavaScript

Lab Environment Setup:

- Launched the Virtual Machine



- Downloaded and used Labsetup as a shared folder between Linux VM and Host PC

- Ran the following commands in the following order in terminal:
  - **dcbuild** to build the container
  - **dcup** to start the container
- Added DNS configuration in **/etc/hosts/**

Lab Tasks:

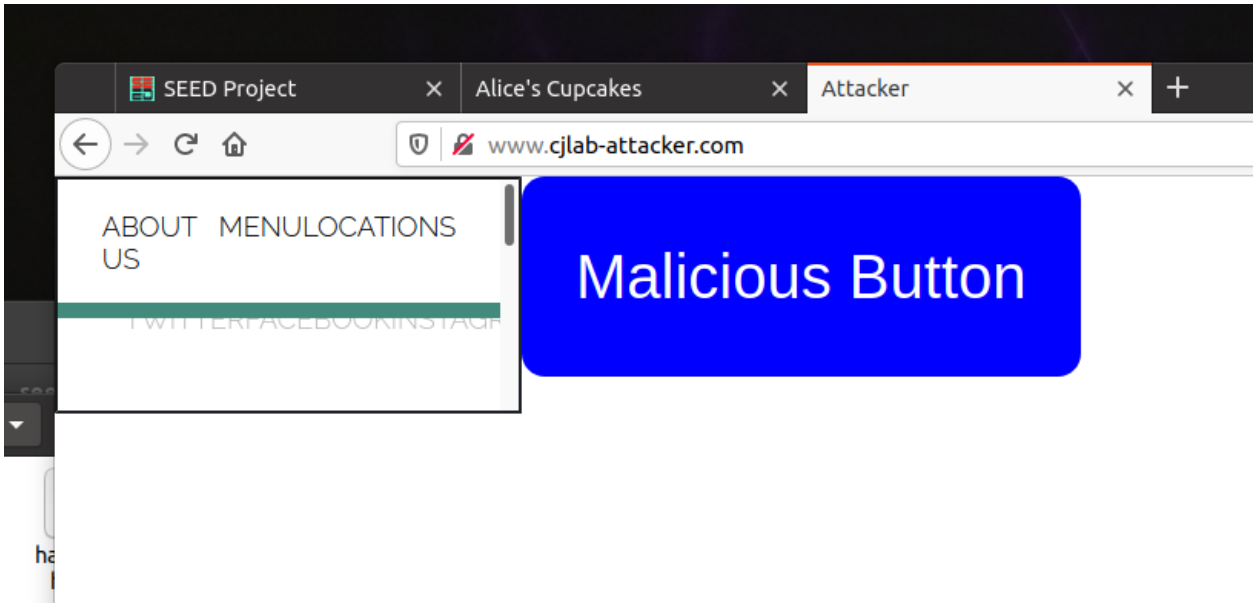
Task 1: Copy that site!

Place the iframe within the attacker file “attacker.html”

```
body>
   <!-- TODO: place your iframe HERE (Task 1) -->
   <iframe src="http://www.cjlab.com/index.html" title = "alice cloned site"></iframe>
   <!-- The malicious button's html code has already been provided for you.
       Note that the button code must come after iframe code-->
```

The next step is to modify the CSS in attacker.css using the height, width, and position attributes to make the iframe cover the whole page and the button overlay the iframe.

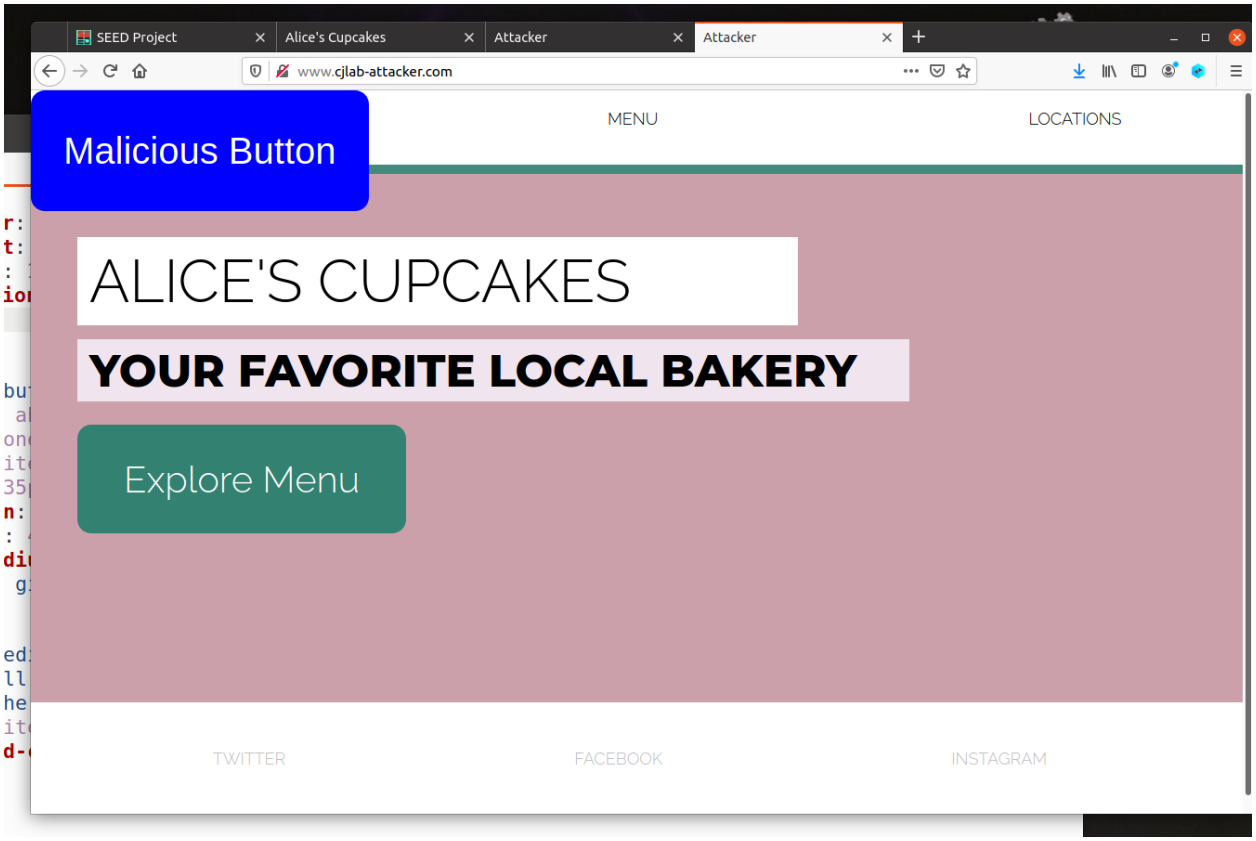
After navigating to www.cjlab-attack.com, this is what is shown before adjusting the dimensions.



After opening attacker.css, adding the css styles for iframe, and refreshing the cache:

attacker.css

```
1 iframe {
2         border: 0;
3         height: 100vh;
4         width: 100vw;
5         position: absolute;
6 }
7
8 button{
9     /* Given button code for size and shape
```



1. With the iframe inserted, what does the attacker's website look like?

   - The website looks like an exact clone to Alice's cupcakes except we see the bright blue malicious button in the upper left corner of the site.

Task 2: Let's Get Clickjacking!

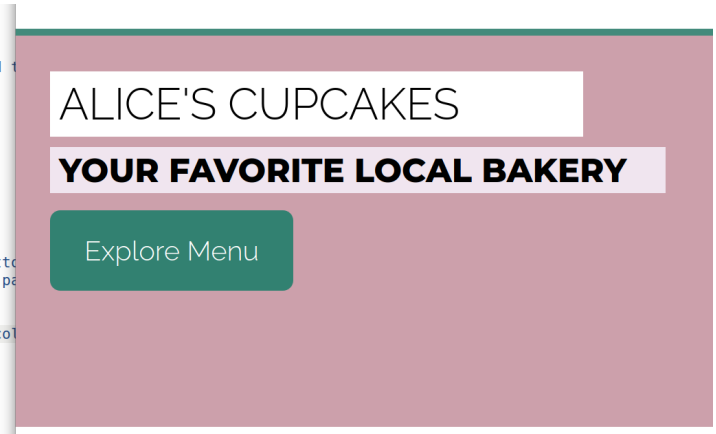
The next step is to add code to the CSS specification of a "button" object given in attacker.css to make the malicious button in attacker.html invisible. First, position the button to cover the "Explore Menu" button within the iframe added in the previous task.

Before making the background transparent, I kept it blue to show the positioning of the malicious button on the page:

```
button{
    /* Given button code for size and shape. You do not need t
    position: absolute;
    border: none;
    color: white;
    padding: 35px 35px;
    text-align: center;
    font-size: 40px;
    border-radius: 15px;
    /* end of given button code */


    /* TODO: edit/add attributes below for the malicious butto
    /* You will want to change the button's position on the pa
       make the button transparent */
    color: transparent;  /* font color */
    background-color: blue;  /* button's background color */
    margin-left: 45px;
    margin-top: 350px;


  }
```

After making the background transparent, the webpage now looks like:

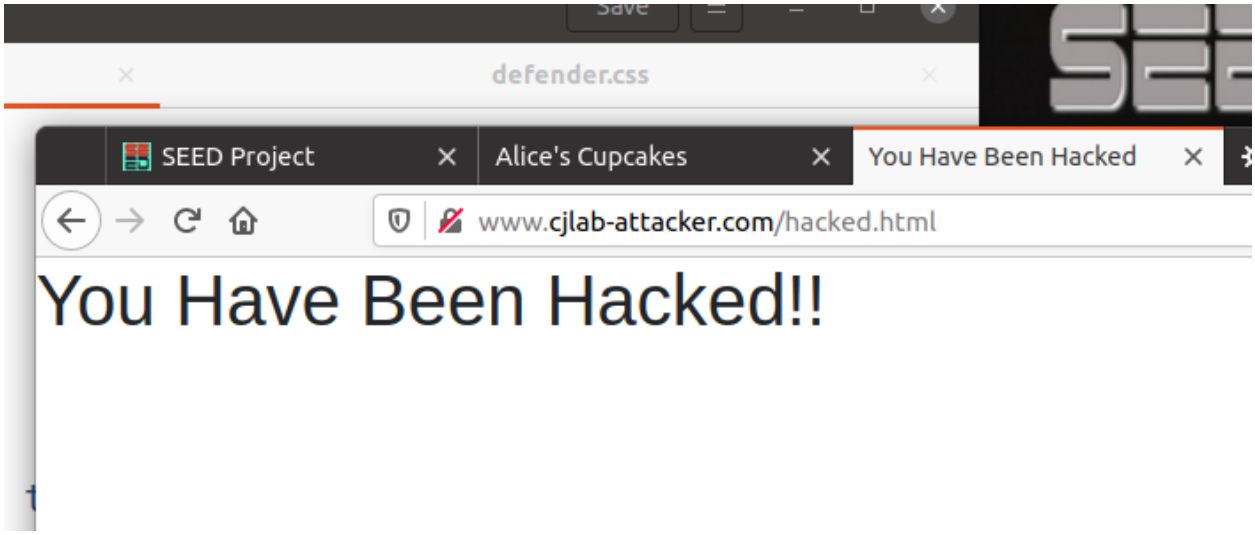
```
}

button{
    /* Given button code for size and shape. You do not need t
    position: absolute;
    border: none;
    color: white;
    padding: 35px 35px;
    text-align: center;
    font-size: 40px;
    border-radius: 15px;
    /* end of given button code */


    /* TODO: edit/add attributes below for the malicious butto
    /* You will want to change the button's position on the pa
       make the button transparent */
    color: transparent;  /* font color */
    background-color: transparent;  /* button's background col
    margin-left: 45px;
    margin-top: 350px;


  }
```



2. How does the appearance of the attacker's site compare to that of the defender's site?

- Aside from the tab name and url difference, it looks completely identical to the defenders site.

3. What happens when you click on the “Explore Menu” button on the attacker’s site?

- When you click the explore menu on the attackers site:
- 

4. Describe an attack scenario in which the style of clickjacking implemented for this Task leads to undesirable consequences for a victim user.

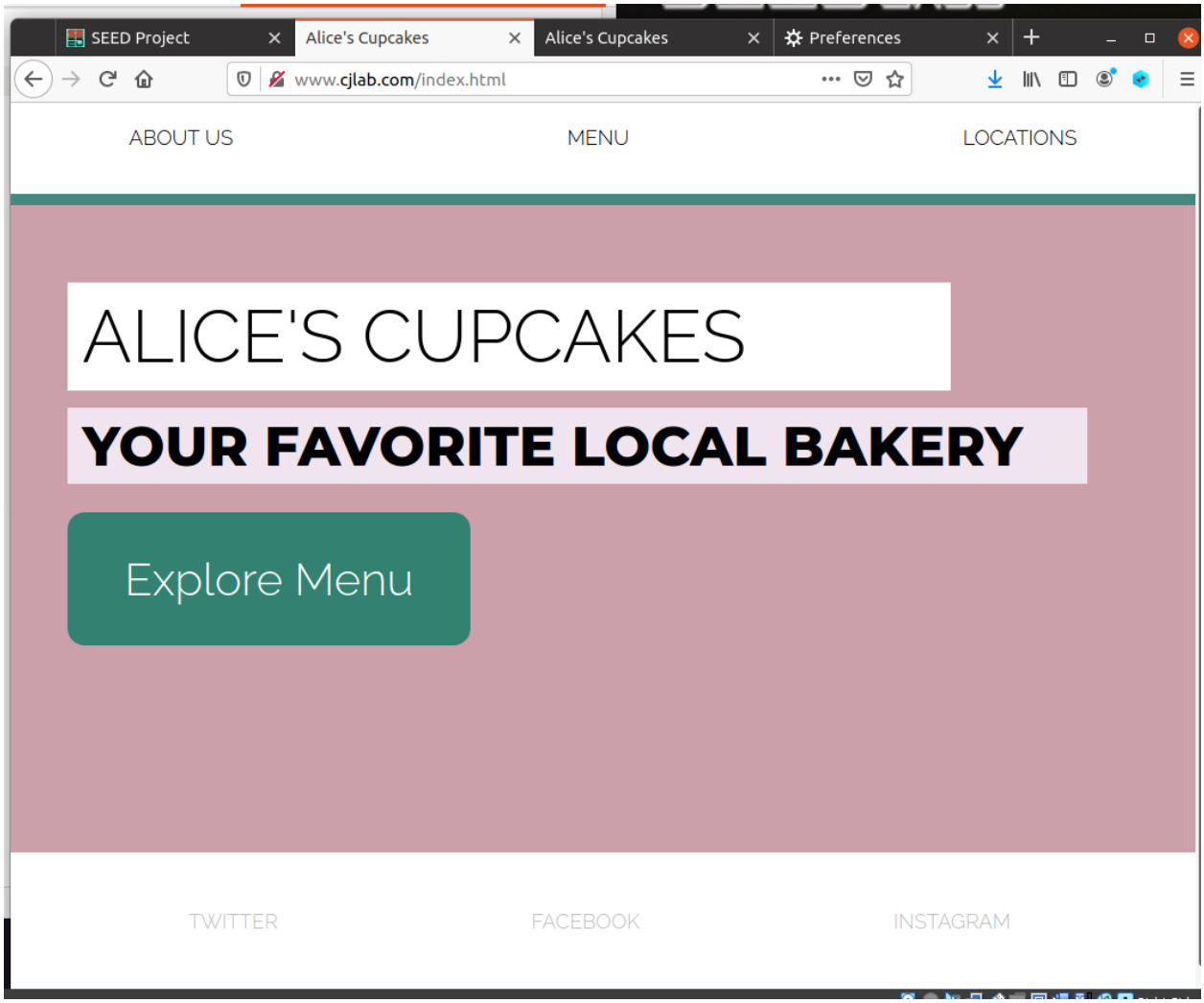
- If a user tried logging in to a private account like Elgg, it could record their login info. If a user wanted to buy a premium membership on the service, they would also have access to their credit card information and any saved data on the user.

Task 3: Bust That Frame!

My next task is to fill in the Javascript method called makeThisFrameOnTop(). My code should compare window.top and window.self to determine if the top window and the current site's window are the same object (as they should be). To achieve this, I wrote an if statement to check if window.top compares to window.self, and if they are not equal, the location object is set as the location of the top of the window.

```
    <!-- Frame Busting script to prevent clickjacking -->
    <script>
        window.onload = function() {
            makeThisFrameOnTop();
        };

        function makeThisFrameOnTop() {
            // TODO: write a frame-busting function according to
            // instructions (Task 3)
            if(window.top !== window.self)
            { window.top.location = window.self.location }

        }

    </script>

</html>
```

After clearing the browser's cache and refreshing the page:



5. What happens when you navigate to the attacker's site now?

- It is now shown that the attacker's site is even more similar to the original site. If you look at the title name and URL, they are almost entirely identical.

6. What happens when you click the button?

- Nothing happens when you click the button, the page appears to refresh but you are no longer being redirected to the You Have Been Hacked!! message, it stays on this landing page.

Task 4: Attacker Countermeasure (Bust the Buster)

My next task as an attacker is to create a workaround for front-end clickjacking defenses like frame busting. My objective is to add the sandbox attribute to the malicious iframe.

```
</head>

<body>
    <!-- TODO: place your iframe HERE (Task 1) -->
    <iframe class="iframe" src="http://www.cjlab.com/index.html" title = "alice cloned site" sandbox></iframe>
    <!-- The malicious button's html code has already been provided for you.
        Note that the button code must come after iframe code-->
    <button onclick="window.location.href = 'hacked.html';">Malicious Button</button>
</body>
```

7. What does the sandbox attribute do? Why does this prevent the frame buster from working?

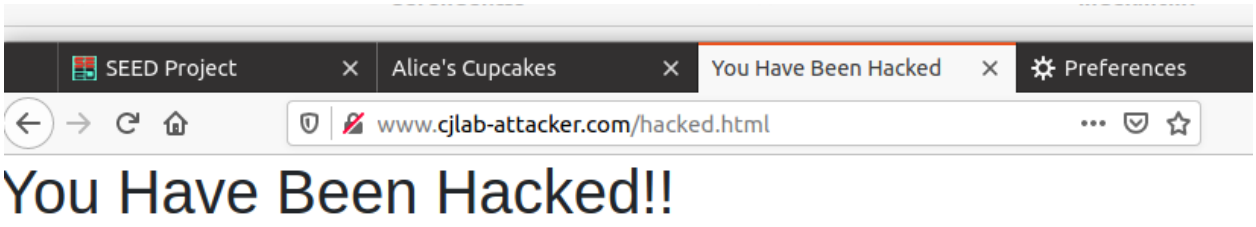
- The sandbox attribute turns off all JavaScript, preventing the frame buster from working. Scripts are disabled within the iframe, and links to other browsing contexts are disabled within the iframe.

8. What happens when you navigate to the attacker's site after updating the iframe to use the sandbox attribute?

- After updating the iframe to use the sandbox attribute, it redirects back to the Attacker title and URL but still looks like a replica of Alice's site.

9. What happens when you click the button on the attacker's site?

- You Have Been Hacked!! is displayed.



-

Task 5: The Ultimate Bust

My next task is to modify the defender's response headers. I achieved this goal by navigating to the image_defender and launching apache_defender.conf in my text editor. I then set the XFO header to the value "DENY" and the CSP header to contain the directive "frame-ancestors 'none'"

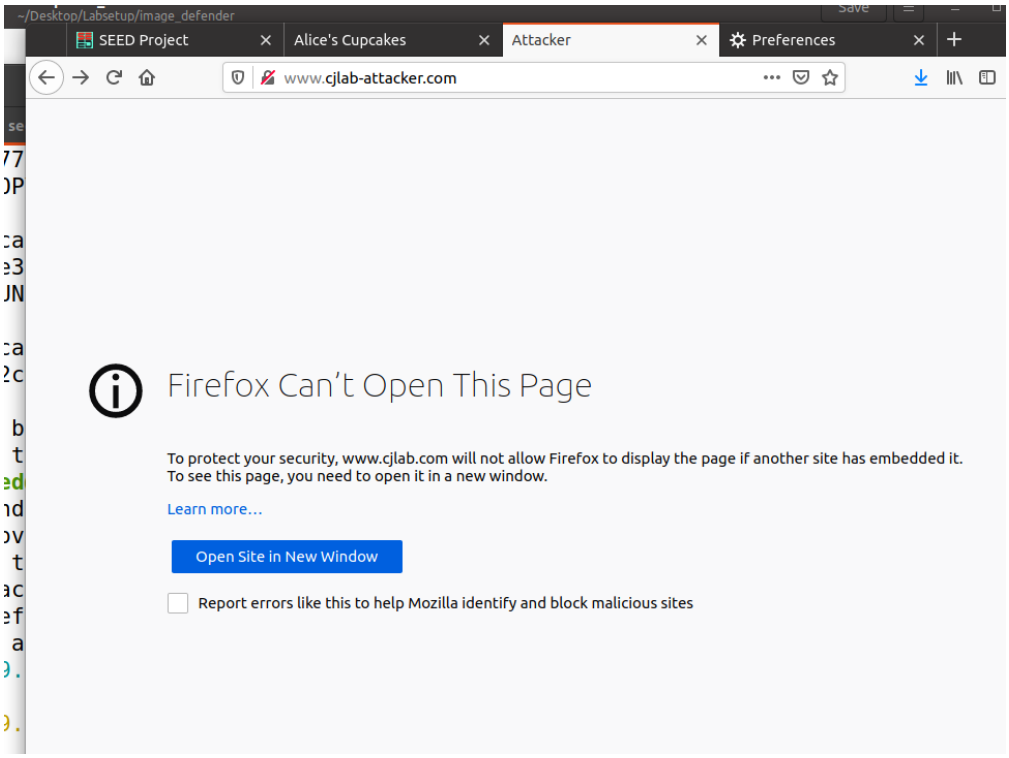
```
<VirtualHost *:80>
    DocumentRoot /var/www/defender
    ServerName www.cjlab.com
    Header set X-Frame-Options "DENY";
    Header set Content-Security-Policy " \
             frame-ancestors 'none'; \
          "
</VirtualHost>
```

Since I'm making a change to the server config, I need to rebuild and restart the containers for the change to take effect. First, I force quit the server by pressing CTRL+C in the terminal I ran the containers on. Then, I rebuilt the container using dcbuild and started the containers with dcup.

```
Attaching to attacker-10.9.0.105, defender-10.9.0.5
attacker-10.9.0.105 |  * Starting Apache httpd web server apache2
 *
defender-10.9.0.5 |  * Starting Apache httpd web server apache2
 *
^CGracefully stopping... (press Ctrl+C again to force)
Stopping attacker-10.9.0.105 ...
Stopping defender-10.9.0.5   ...
Killing attacker-10.9.0.105  ... done
Killing defender-10.9.0.5    ... done
[11/20/22]seed@VM:~/.../Labsetup$
```

```
Step 2/3 : COPY apache_attacker.conf server_name.conf /etc/apache2/sites-availab
le/
 ---> Using cache
 ---> 2262aae38751
Step 3/3 : RUN  a2ensite server_name.conf        && a2ensite apache_attacker.con
f
 ---> Using cache
 ---> efe1e12c0c24

Successfully built efe1e12c0c24
Successfully tagged seed-image-attacker-clickjacking:latest
[11/20/22]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (mysql-10.9.0.6, www-10.9.0.5) for this project
. If you removed or renamed this service in your compose file, you can run this
command with the --remove-orphans flag to clean it up.
Starting attacker-10.9.0.105 ... done
Recreating defender-10.9.0.5 ... done
Attaching to attacker-10.9.0.105, defender-10.9.0.5
attacker-10.9.0.105 |  * Starting Apache httpd web server apache2
 *
defender-10.9.0.5 |  * Starting Apache httpd web server apache2
 *
```

The original site is still accessible, however the attack site can not be launched



10. What is the X-Frame-Options HTTP header attribute, and why is it set to “DENY” to prevent the attack?

- XFO is an HTTP response header, and setting it to DENY does not allow rendering a page in a frame.

11. What is the Content-Security-Policy header attribute, and why is it set to “frame-ancestors ‘none’ ” to prevent the attack?

- CSP is an HTTP response header, and setting frame-ancestors to none prevents any domain from framing the content.

12. What happens when you navigate to the attacker’s site after modifying each response header (one at a time)? What do you see when you click the button?

- After modifying each response header one at a time, the attack site cannot be launched. Therefore I cannot click the button.