

Part 2 (click jacking)

Before we begin, what is **click jacking**? – Clickjacking also known as a “**UI redress attack**”, is when an attacker uses multiple transparent layer to trick users to clicking on a button or link on another page when they were intending to click on top level pages.

.....

Today, we’re going to take a look at the hands-on-lab for seed to get broad understanding of this topic.

First, if you have seed install in your **virtual machine**, common, lets’ begin go on and create a new folder and download the package from here =>

https://seedsecuritylabs.org/Labs_20.04/Web/Web_Clickjacking_Cupcakes/ This will help you get started for the run ^_^ . OR if you haven’t downloaded virtual machine neither seed, visit this website and then follow the steps to download. : <https://www.kali.org/get-kali/#kali-virtual-machines> .

.....

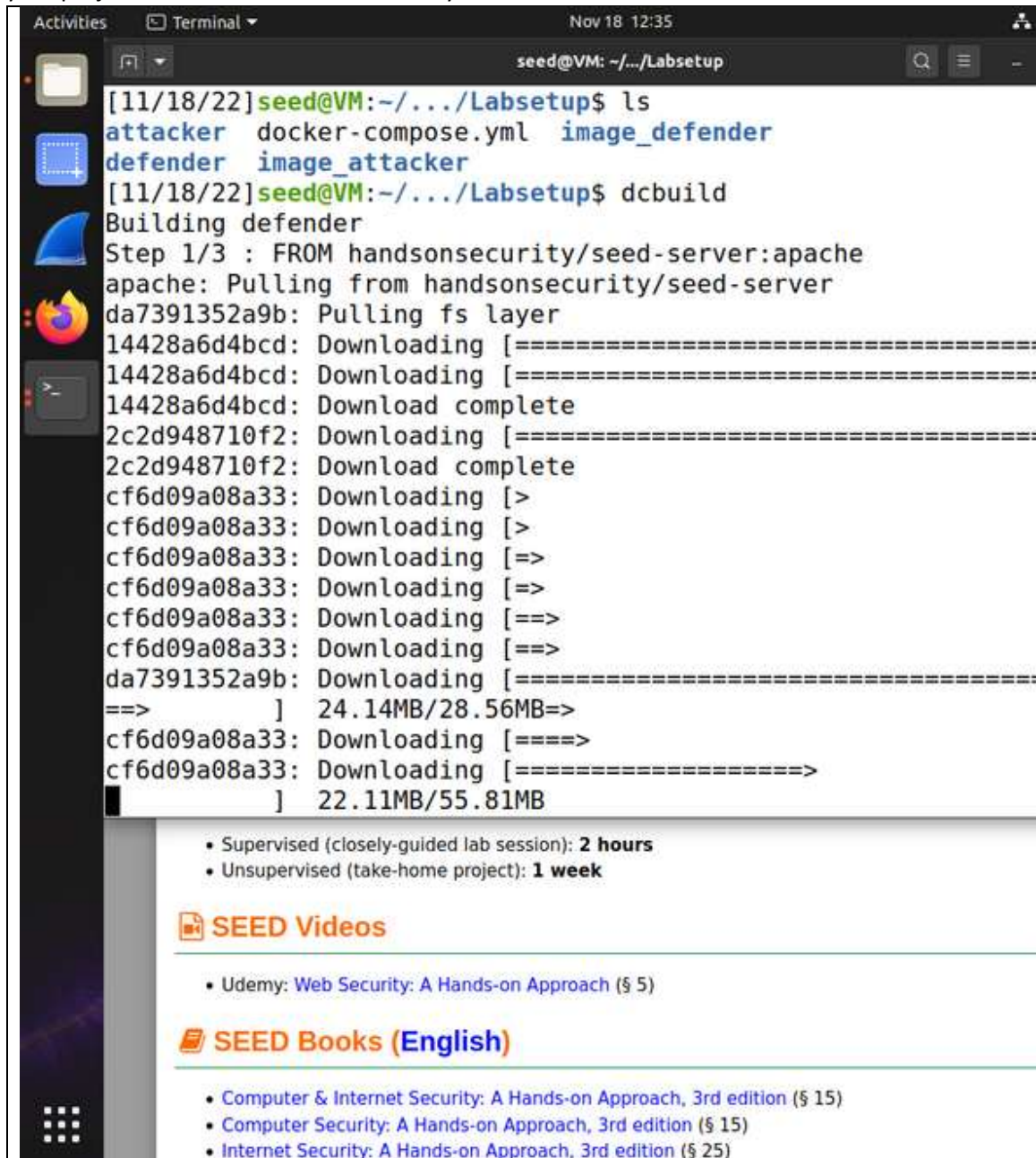
Open you LabSetup in you terminal then follow these instructions.

1. Enter ls (list) to make sure that you’re in the right directory. You should see something like this

attacker docker-compose.yml image_defender defender image_attacker



- Be sure to **build** you project before you can use. To do this, you should enter **dcbuild** to build your project. It should look like this after you've built it...



```

[11/18/22] seed@VM: ~/.../Labsetup$ ls
attacker  docker-compose.yml  image_defender
defender  image_attacker
[11/18/22] seed@VM: ~/.../Labsetup$ dcbuild
Building defender
Step 1/3 : FROM handsonsecurity/seed-server:apache
apache: Pulling from handsonsecurity/seed-server
da7391352a9b: Pulling fs layer
14428a6d4bcd: Downloading [=====]
14428a6d4bcd: Downloading [=====]
14428a6d4bcd: Download complete
2c2d948710f2: Downloading [=====]
2c2d948710f2: Download complete
cf6d09a08a33: Downloading [>]
cf6d09a08a33: Downloading [>]
cf6d09a08a33: Downloading [=>]
cf6d09a08a33: Downloading [=>]
cf6d09a08a33: Downloading [==>]
cf6d09a08a33: Downloading [==>]
da7391352a9b: Downloading [=====]
==> ] 24.14MB/28.56MB=>
cf6d09a08a33: Downloading [====>]
cf6d09a08a33: Downloading [=====>]
] 22.11MB/55.81MB

```

- Supervised (closely-guided lab session): **2 hours**
- Unsupervised (take-home project): **1 week**

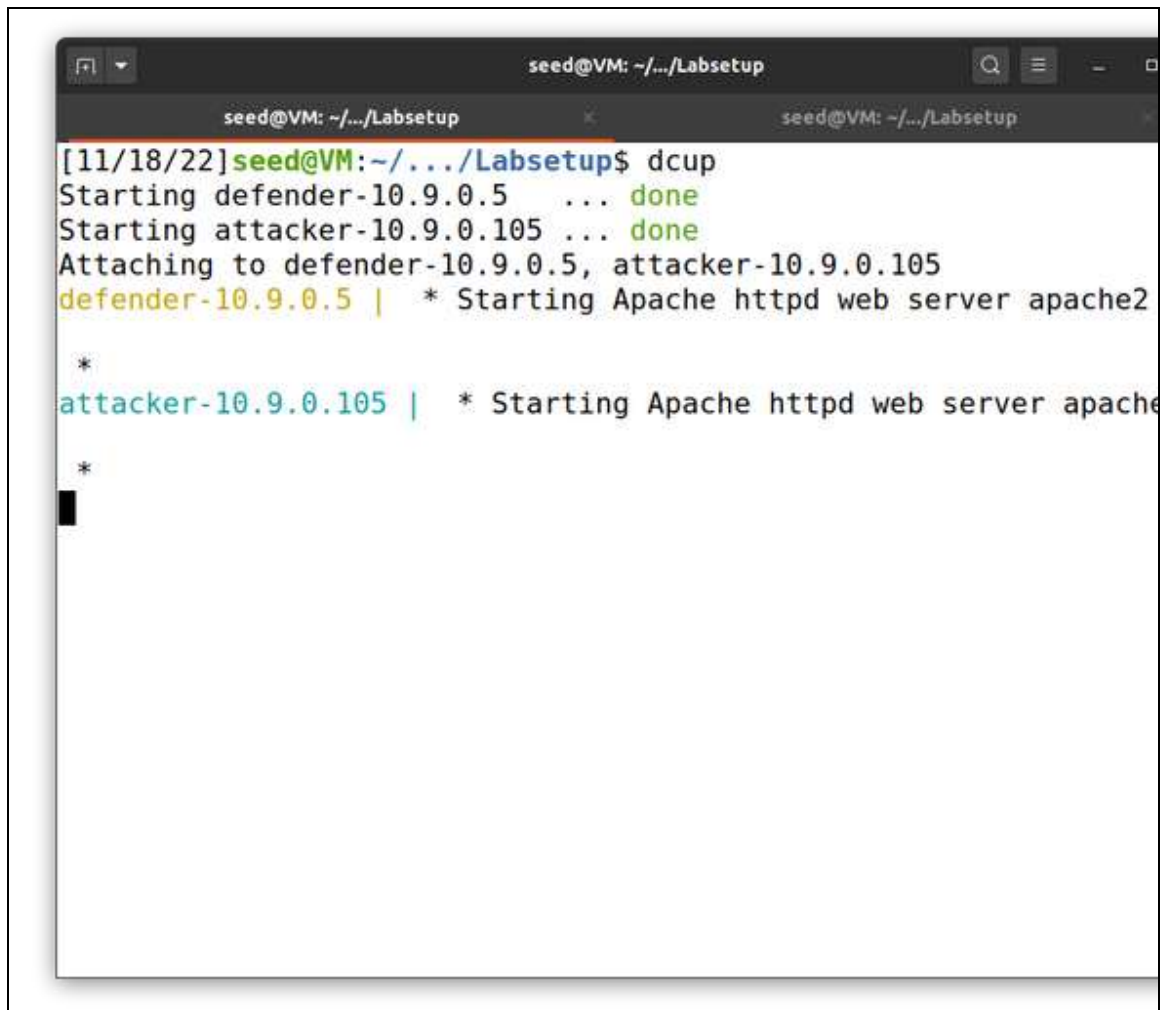
SEED Videos

- Udemy: [Web Security: A Hands-on Approach](#) (\$ 5)

SEED Books (English)

- Computer & Internet Security: A Hands-on Approach, 3rd edition (\$ 15)
- Computer Security: A Hands-on Approach, 3rd edition (\$ 15)
- Internet Security: A Hands-on Approach, 3rd edition (\$ 25)

- Now that you've built you project, where're the furniture to make you project look nice. That's why you should enter **dcup** to buy all the furniture needed. This command will make sure you project is hosted and running. This too should look like this.

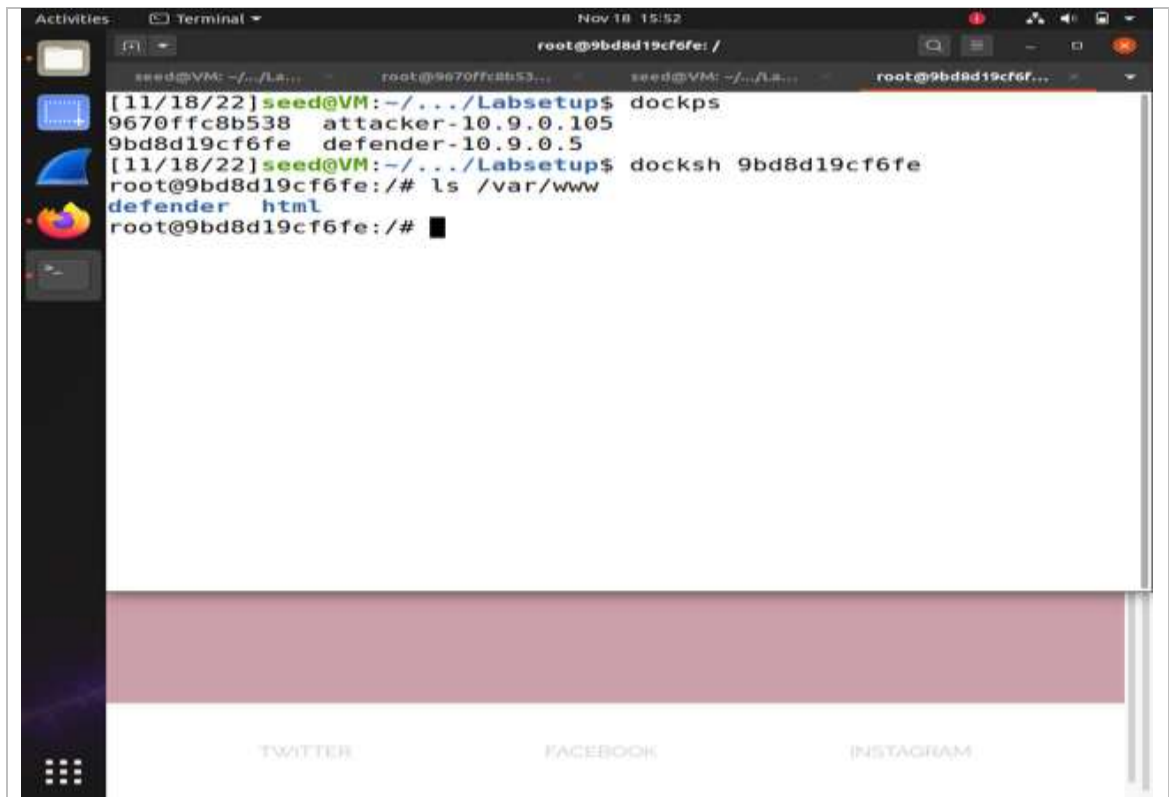


```
seed@VM: ~/.../Labsetup
[11/18/22]seed@VM:~/.../Labsetup$ dcup
Starting defender-10.9.0.5 ... done
Starting attacker-10.9.0.105 ... done
Attaching to defender-10.9.0.5, attacker-10.9.0.105
defender-10.9.0.5 | * Starting Apache httpd web server apache2
*
attacker-10.9.0.105 | * Starting Apache httpd web server apache2
*
█
```

4. Open a new tab after entering **dcup**, do not stop the program from running i.e. entering [ctrl + c] this will make your server to stop working. Ok, after you've opened a new tab on you terminal, type this command, **[dockps]** This will show your [id's] needed. The attackers [id] and the defender's [id]... This should look like this.

```
9670ffc8b538 attacjer – 10.9.0.105
9bd8d19cf6fe defender – 10.9.0.5
```

CLICK-JACKING-SEED-LAB



The screenshot shows a terminal window with a dark background and a light-colored text area. The terminal is running a series of Docker commands. The first command is `dockps`, which lists the running containers. The output shows two containers: `9670ffc8b538` (attacker-10.9.0.105) and `9bd8d19cf6fe` (defender-10.9.0.5). The second command is `docksh 9bd8d19cf6fe`, which opens a shell inside the `defender` container. The prompt changes to `root@9bd8d19cf6fe:/#`. The third command is `ls /var/www`, which lists the contents of the `/var/www` directory. The output is `defender html`. The terminal window has a title bar that says "Activities" and "Terminal". The date and time "Nov 18 15:52" are displayed in the top right corner. The terminal window is part of a desktop environment with a sidebar on the left containing icons for a file manager, a web browser, and a terminal. At the bottom of the screen, there are social media links for "TWITTER", "FACEBOOK", and "INSTAGRAM".

```
root@9bd8d19cf6fe: /
[11/18/22]seed@VM:~/.../Labsetup$ dockps
9670ffc8b538  attacker-10.9.0.105
9bd8d19cf6fe  defender-10.9.0.5
[11/18/22]seed@VM:~/.../Labsetup$ docksh 9bd8d19cf6fe
root@9bd8d19cf6fe:/# ls /var/www
defender  html
root@9bd8d19cf6fe:/#
```

5. Have you forgotten cat? How would you know the server where you project is being hosted to? Common lets type `[cat /etc/hosts]` to display your host on the terminal. That the main job for cat displaying things on the termina. ^_^. This should look like this.

```

seed@VM: ~/.../Labsetup
[11/18/22]seed@VM:~/.../Labsetup$ dockps
9670ffc8b538  attacker-10.9.0.105
9bd8d19cf6fe  defender-10.9.0.5
[11/18/22]seed@VM:~/.../Labsetup$ ls
attacker  docker-compose.yml  image_defender
defender  image_attacker
[11/18/22]seed@VM:~/.../Labsetup$
[11/18/22]seed@VM:~/.../Labsetup$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      VM

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters

# For DNS Rebinding Lab
192.168.60.80  www.seedIoT32.com

# For SQL Injection Lab
10.9.0.5       www.SeedLabSQLInjection.com

```

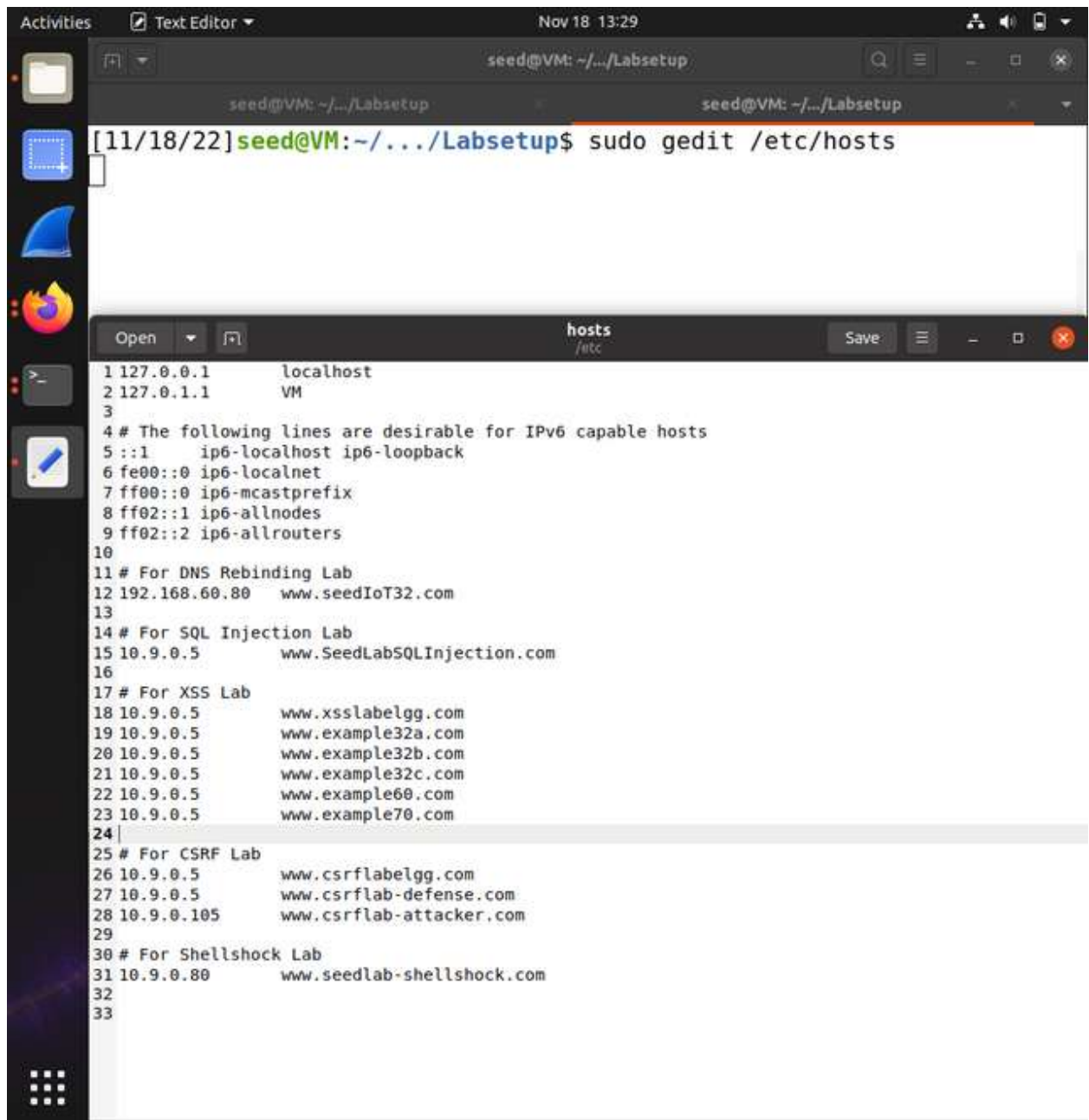
6. One more step. Type this command [sudo gedit /etc/hosts]. This will open a note pad for you and allow you to edit the host platform. Be sure to add this host location. DNS configuration.
 [

 "10.9.0.5 www.cjlab.com",

 "10.9.0.105 www.cjlab-attacker.com"

] This command should be added in the /etc/host file.... You opened with [sudo gedit /etc/hosts] and then save it [ctrl + s].

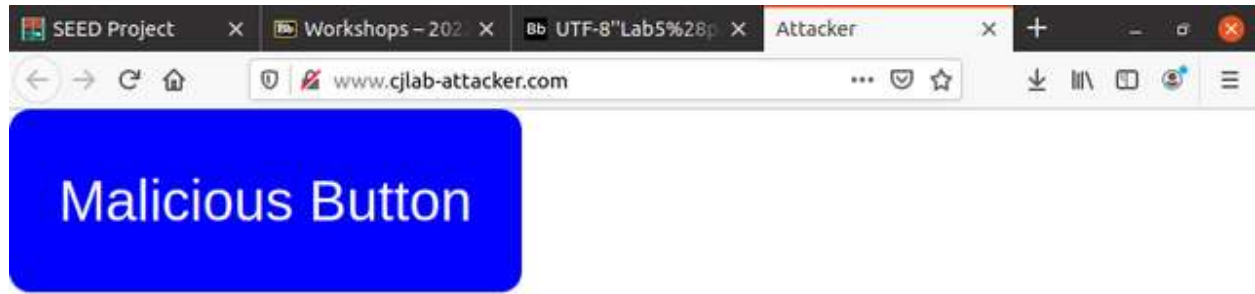
CLICK-JACKING-SEED-LAB



The screenshot shows a Linux desktop environment. At the top, a terminal window displays the command `sudo gedit /etc/hosts` being executed. Below the terminal, a text editor window titled `hosts /etc` is open, showing the contents of the `/etc/hosts` file. The file contains a list of IP addresses and their corresponding hostnames, including `localhost`, `VM`, and various domains for different labs like `www.seedIoT32.com`, `www.SeedLabSQLInjection.com`, `www.xsslabelgg.com`, `www.csrflabelgg.com`, and `www.seedlab-shellshock.com`.

```
1 127.0.0.1    localhost
2 127.0.1.1    VM
3
4 # The following lines are desirable for IPv6 capable hosts
5 ::1         ip6-localhost ip6-loopback
6 fe00::0     ip6-localnet
7 ff00::0     ip6-mcastprefix
8 ff02::1     ip6-allnodes
9 ff02::2     ip6-allrouters
10
11 # For DNS Rebinding Lab
12 192.168.60.80 www.seedIoT32.com
13
14 # For SQL Injection Lab
15 10.9.0.5      www.SeedLabSQLInjection.com
16
17 # For XSS Lab
18 10.9.0.5      www.xsslabelgg.com
19 10.9.0.5      www.example32a.com
20 10.9.0.5      www.example32b.com
21 10.9.0.5      www.example32c.com
22 10.9.0.5      www.example60.com
23 10.9.0.5      www.example70.com
24
25 # For CSRF Lab
26 10.9.0.5      www.csrflabelgg.com
27 10.9.0.5      www.csrflab-defense.com
28 10.9.0.105    www.csrflab-attacker.com
29
30 # For Shellshock Lab
31 10.9.0.80     www.seedlab-shellshock.com
32
33
```

After doing that, Go to this website. <http://www.cjlab-attacker.com> you should see a single button.
Here's a preview.



3 LAB TASKS.

3.1 TASK 1: COPY THAT SITE.

In the defender folder, you will find the files comprising the website for Alice's Cupcakes: index.html and defender.css. In the attacker folder, you will find the files comprising the attacker's website: attacker.html and attacker.css. You will be making changes to all of these files except defender.css throughout the lab. Our first step as the attacker is to add code to attacker.html so that it mimics the

Alice's Cupcakes website as closely as possible. A common way to do this is with an HTML Inline Frame element ("iframe"). An iframe enables embedding one HTML page within another. The src attribute of the iframe specifies the site to be embedded, and when the iframe code is executed on a page, the embedded site is loaded into the iframe

.....SOLUTION.....

- Add an iframe HTML element in `attacker.html` that pulls from `http://www.cjlab.com`.

First, let's navigate to the attacker folder then enter `[docksh 9670ffc8b538]`. Your output will be like

[

```
root@9670ffc8b538:/#
```

]

[

```
Enter => ls /var/www. To view the files available in the attackers folder.
```

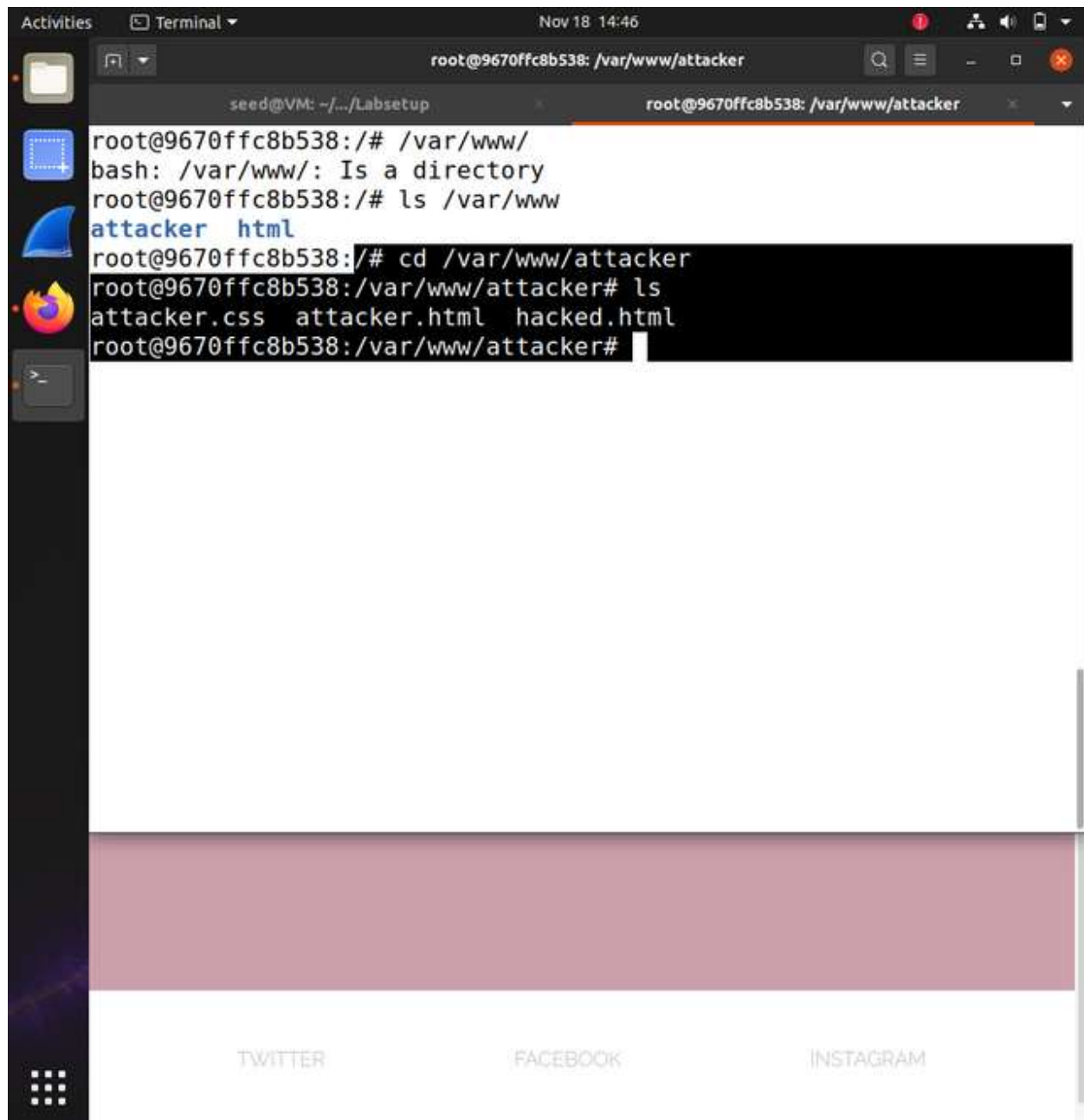
```
//image required
```

```
Then enter => cd /var/www/attacker.
```

```
//image required
```

]

for you to be able to edit your html and add the defenders's page in the iframe, enter ***[nano attacker.html]***



The screenshot shows a terminal window with the following commands and output:

```
root@9670ffc8b538: /var/www/attacker
seed@VM: ~/.../Labsetup
root@9670ffc8b538: /var/www/
bash: /var/www/: Is a directory
root@9670ffc8b538: /var/www
attacker.html
root@9670ffc8b538: /var/www/attacker
root@9670ffc8b538: /var/www/attacker# ls
attacker.css  attacker.html  hacked.html
root@9670ffc8b538: /var/www/attacker#
```

The terminal window is titled "root@9670ffc8b538: /var/www/attacker". The background of the terminal shows a web page with social media links for TWITTER, FACEBOOK, and INSTAGRAM.

- 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`.

..SOLUTION...

This is just the same... [nano `attacker.css`] to edit the file. Or you can just open the file in to text editor.

Make sure to just edit the (`iframe` css) it looks like this.

Do this in the css...

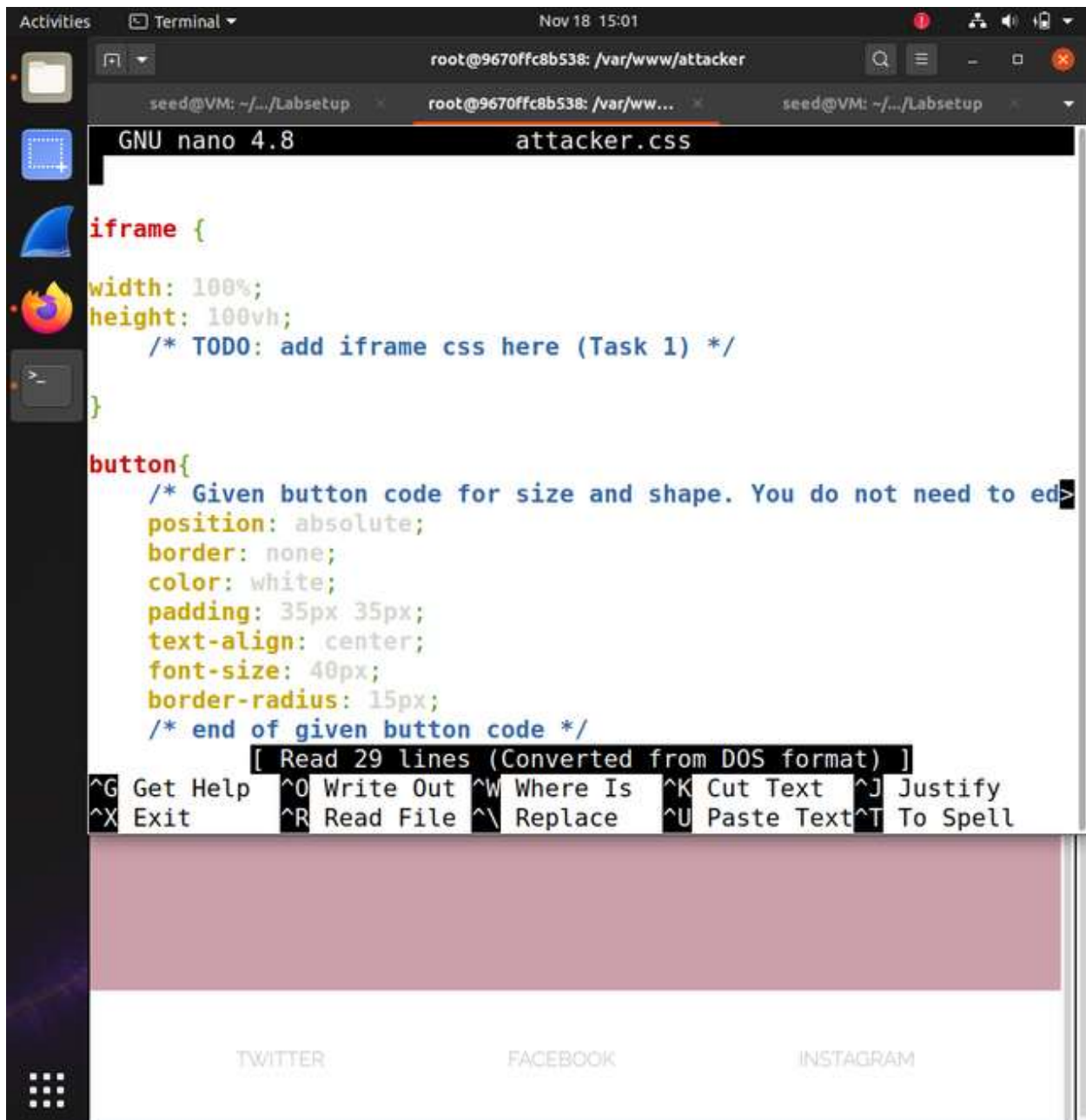
```
iframe {
```

CLICK-JACKING-SEED-LAB

Width: 100%;

height: 100vh;

}



```
GNU nano 4.8 attacker.css

iframe {
width: 100%;
height: 100vh;
/* TODO: add iframe css here (Task 1) */
}

button{
/* Given button code for size and shape. You do not need to ed
position: absolute;
border: none;
color: white;
padding: 35px 35px;
text-align: center;
font-size: 40px;
border-radius: 15px;
/* end of given button code */
[ Read 29 lines (Converted from DOS format) ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

TWITTER FACEBOOK INSTAGRAM
```

OUTPUT



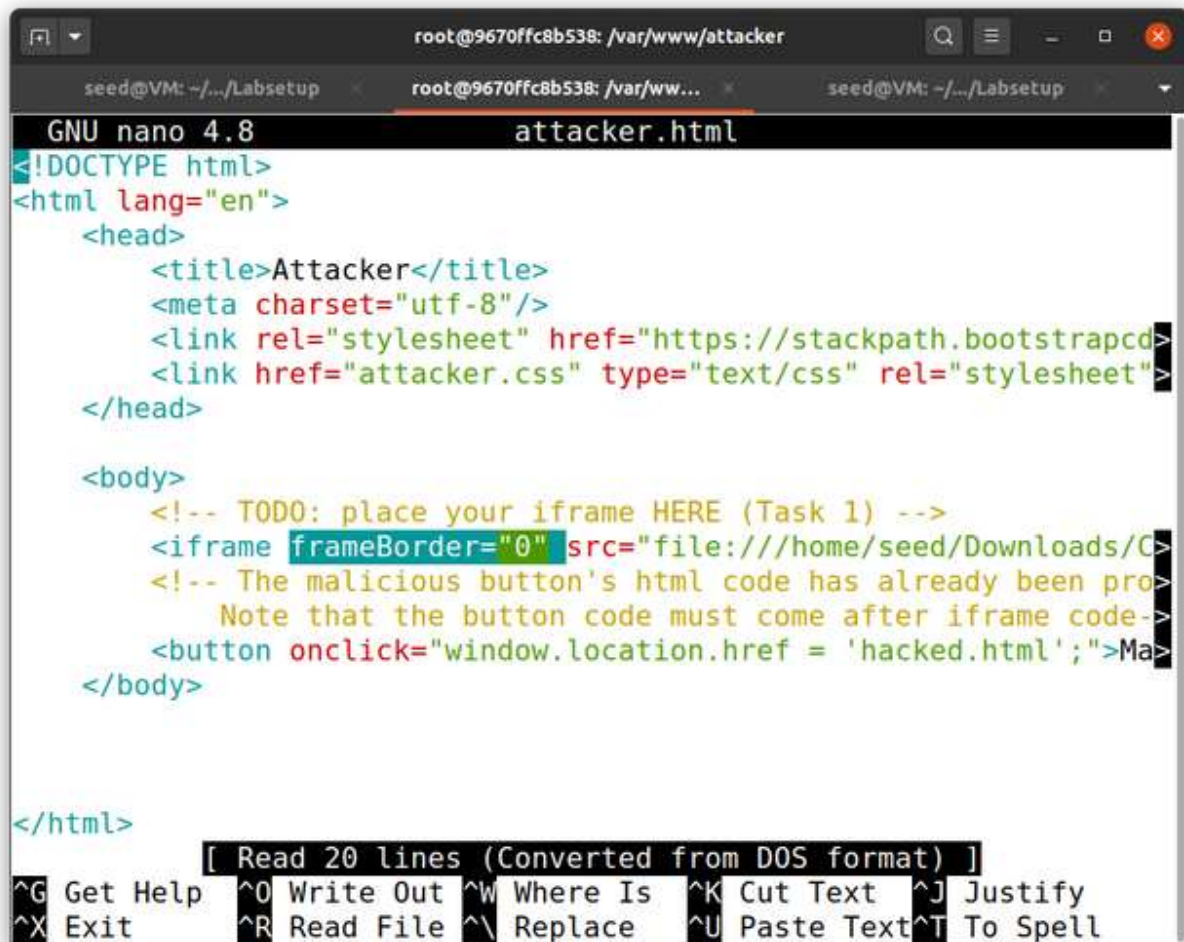
- Hints:

- Explicitly set the iframe to have no border.

...Solution...

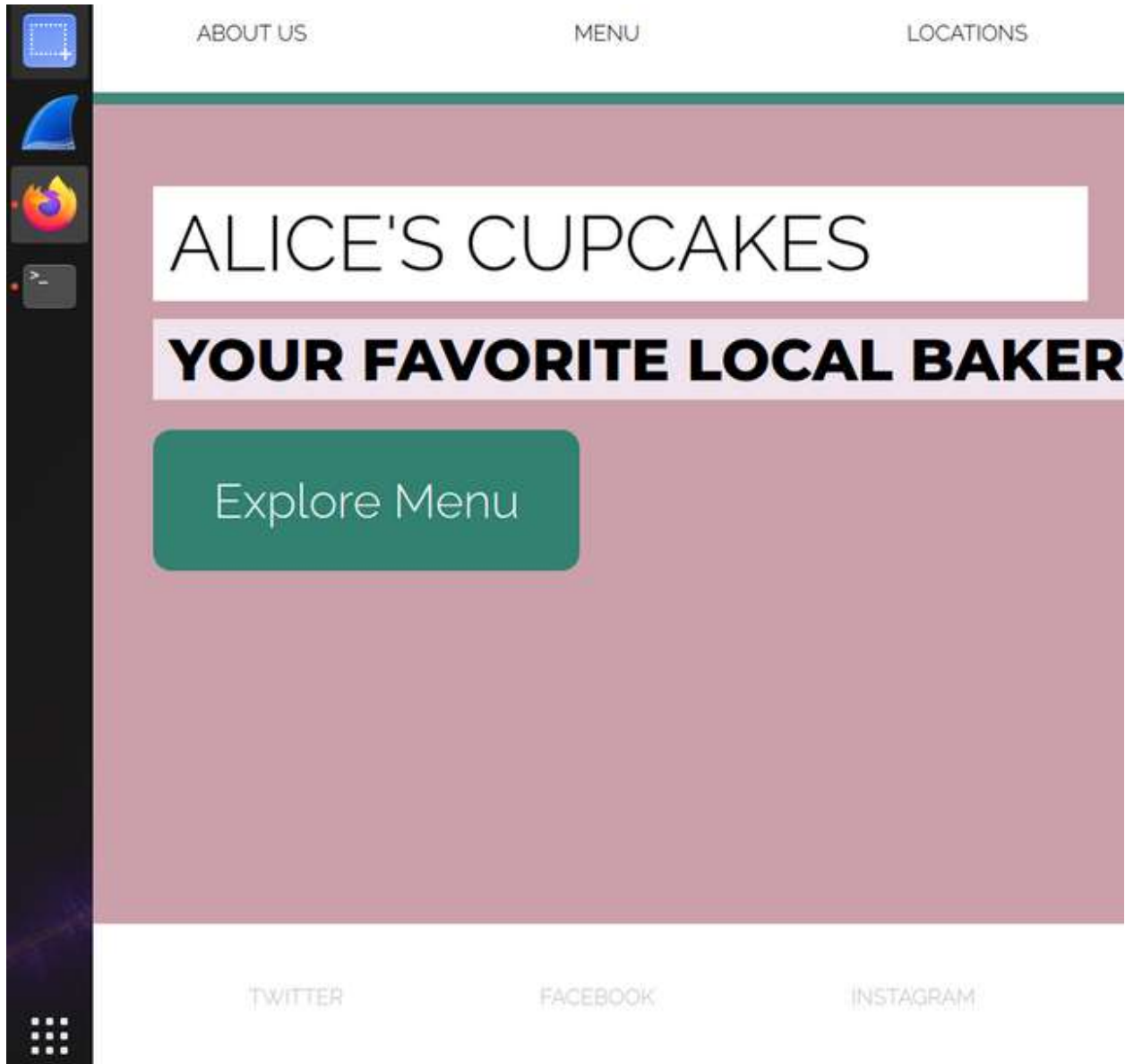
Add [frameborder="0"] in the iframe.

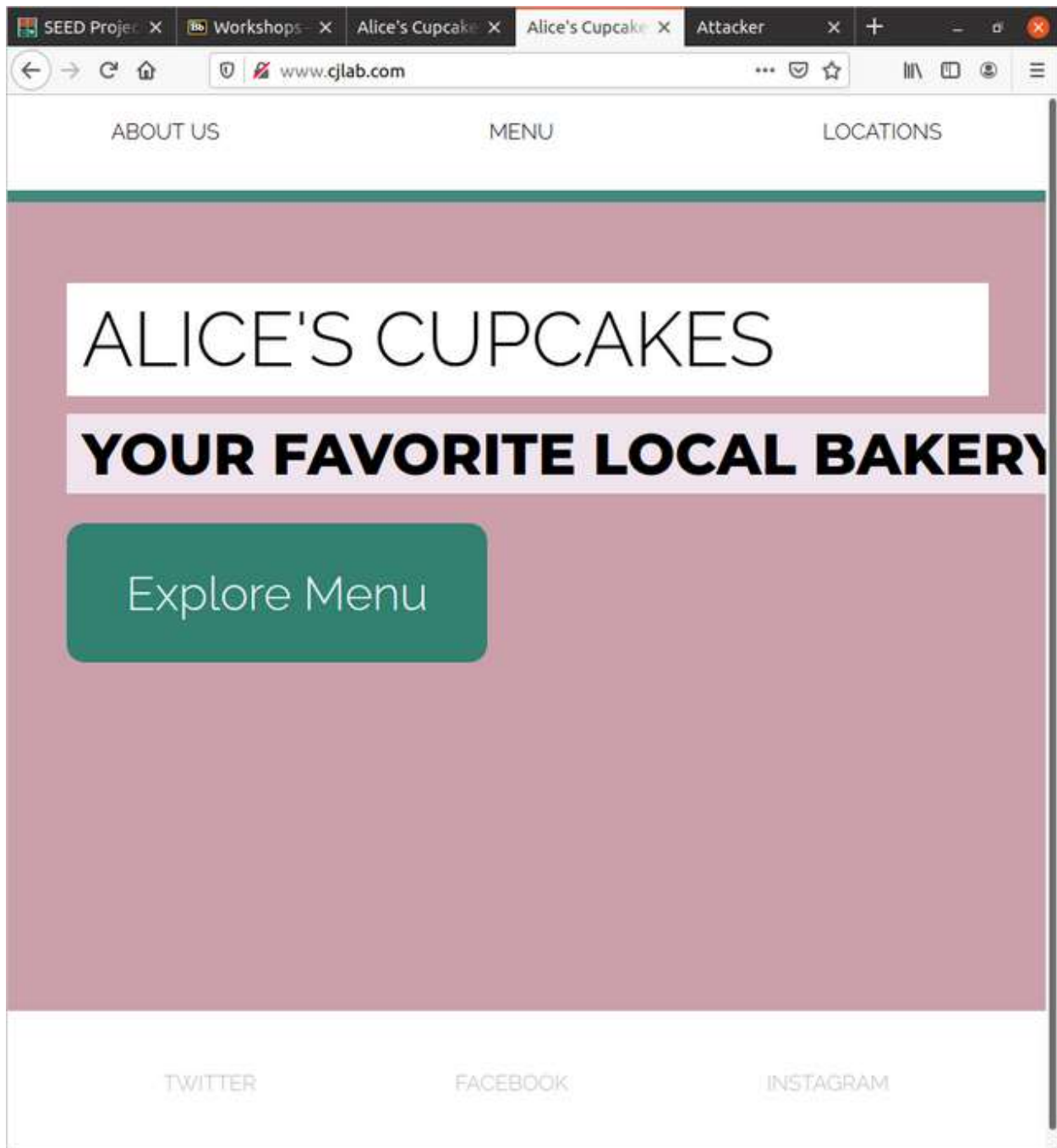
```
<iframe src="..." frameborder="0">
```



```
GNU nano 4.8 attacker.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Attacker</title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link href="attacker.css" type="text/css" rel="stylesheet">
  </head>
  <body>
    <!-- TODO: place your iframe HERE (Task 1) -->
    <iframe frameborder="0" src="file:///home/seed/Downloads/C
    <!-- The malicious button's html code has already been pro
    Note that the button code must come after iframe code-
    <button onclick="window.location.href = 'hacked.html';">Ma
  </body>
</html>

[ Read 20 lines (Converted from DOS format) ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell
```





3.2 TASK 2: LET'S GET CLICKJACKING!...

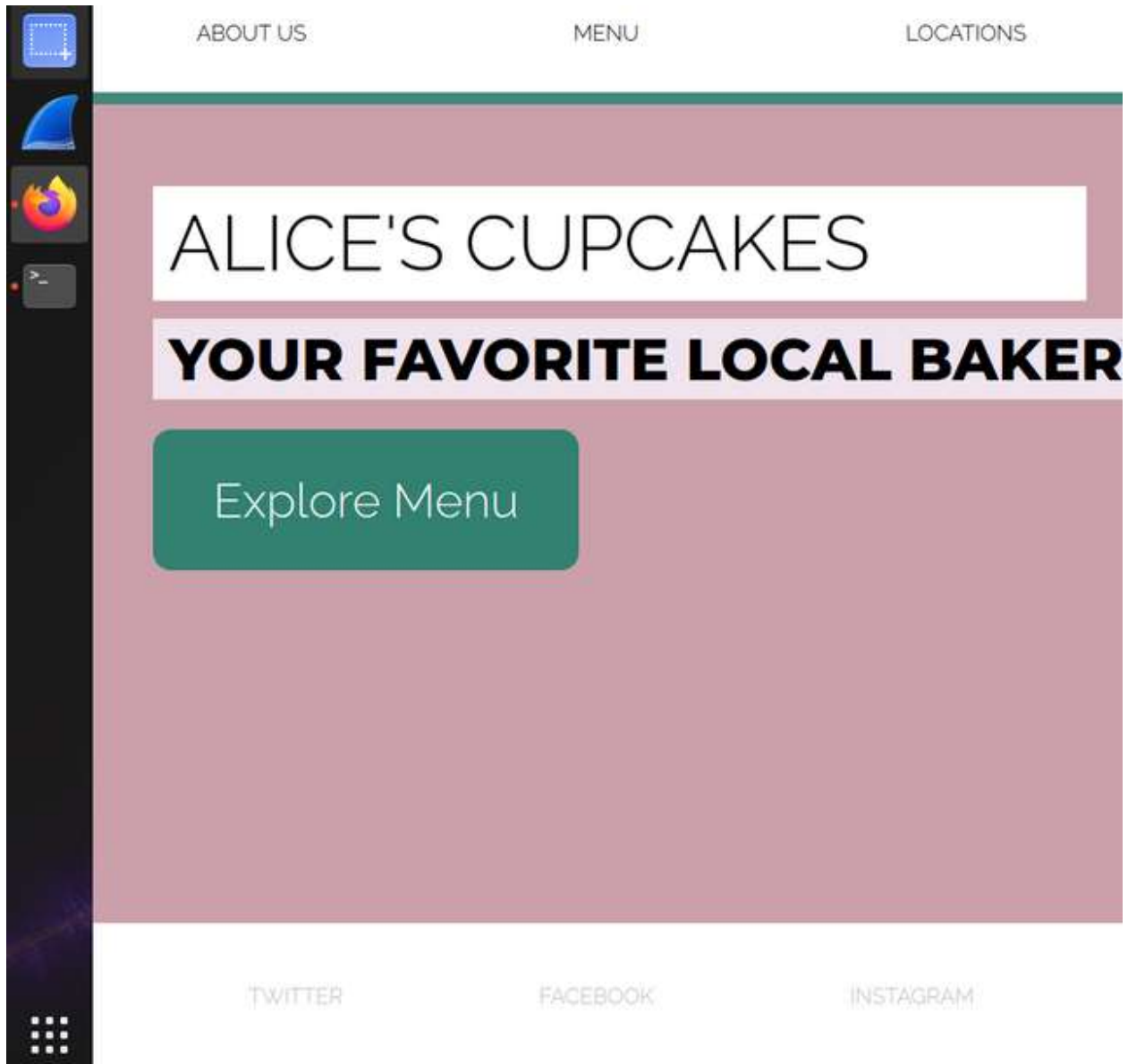
Basic clickjacking attack. Add code to the CSS specification of a “button” object given in `attacker.css` to make the malicious button in `attacker.html` invisible. Position the button so that it covers the “Explore Menu” button within the iframe added in the previous Task. There are a variety of ways to accomplish this Task; we recommend using the CSS attributes `margin-left`, `margin-top`, `color`, and `background-color`. When this Task is complete, you will have a functioning clickjacking attack.

..... SOLUTION.....

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

Ans => {

Same appearance



}

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

ANS => {

The page reloads. But nothing else happens.

}

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

Ans => {

Using This click jacking attack, when a user clicks on a particular part of an iframed page. Example you can use youtube video, facebook post, videos etc.. iframe. To hack someone. If the hacker position the attack button in the right position, when a user click on the invinsible button provided by the attacker instead for the user to be directed to the right page, they will be redirected to the attacker's page of choice.

}

3.3 TASK 3: BUST THE FRAME!

“Frame busting” is the practice of preventing a web page from being displayed within a frame, thus defending against the type of attack implemented in the previous Task. One way to bust frames is to include script code in the webpage source that prevents the site from being framed – that is, it prevents other sites from opening the webpage in an iframe. In this Task we will add script code to the defender's webpage that ensures it is the topmost window on any page where it is being displayed, thus preventing buttons on an attacker's page from being overlaid on top of it.

...SOLUTION...

This defence requires JAVASCRIPT SKILL to protect

First... Navigate to the defenders folder by using [docksh defenders_id]

[

```
root@9bd8d19cf6fe:/#
```

]

```
root@9bd8d19cf6fe:/# ls /var/www
```

```
Defender.html
```

```
root@9bd8d19cf6fe:/# cd /var/www/defender
```

```
root@9bd8d19cf6fe:/#/var/www/defender
```

```
root@9bd8d19cf6fe:/#/var/www/defender nano index.html
```

Now that you've opened nano, add the following code. On the bottom of the body

//

Disclaimer! The code below will stop the iframe attack but now in all cases. In some device the prevent **popups** which will prevent this defence from working. Well, This defence will automatically change the url or the website where the attacker entered the defence site. The url will be changed from <http://www.cjlab-attacker> to <http://www.cjlab.com> soon as the user reloads their page.

Code

{

```
<script>
    if(window.top !== window.self){
        window.top.location = window.self.location
    }
</script>
```

}

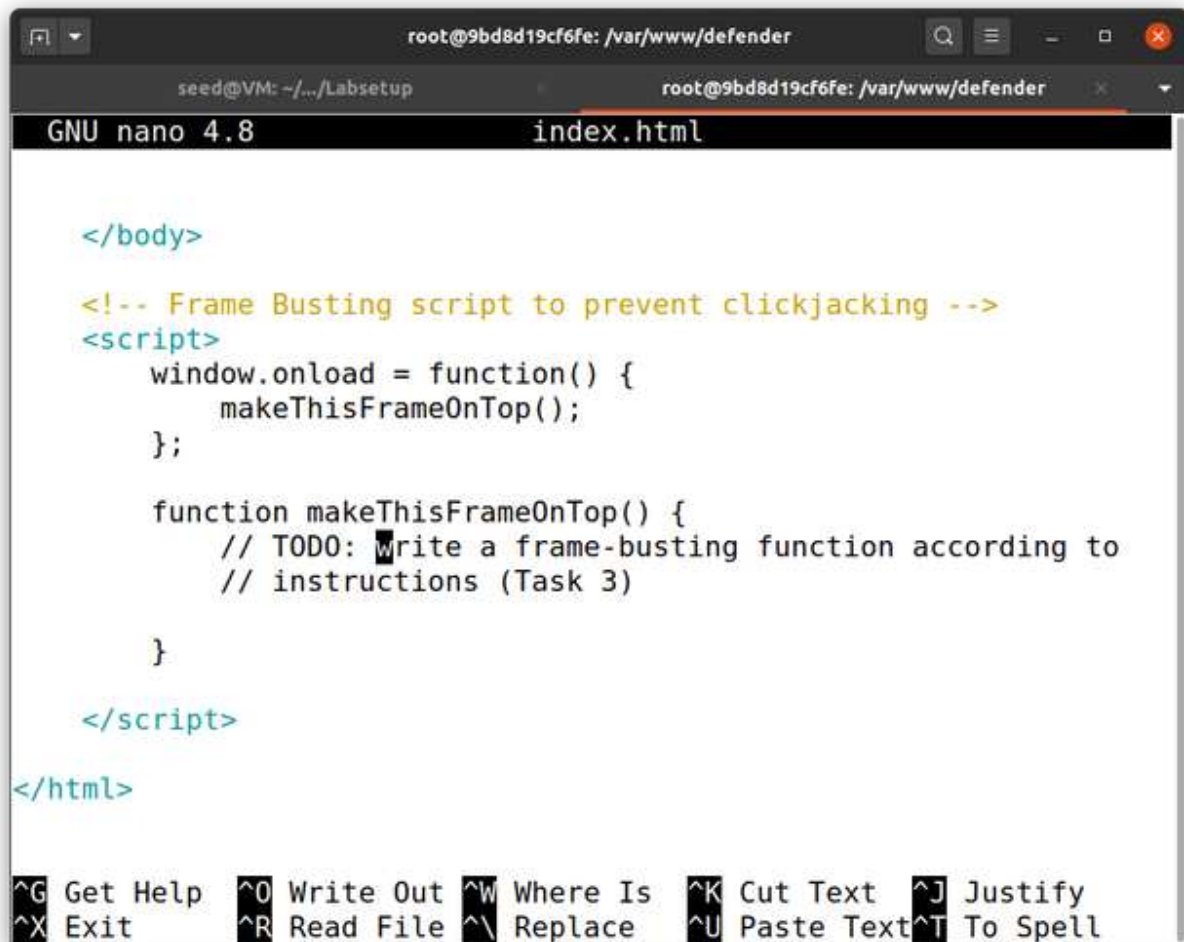
OR

{

```
<script>
    if(window.top !== window.self){
        window.location = window.self
    }
</script>
```

}

BEFORE



```
root@9bd8d19cf6fe: /var/www/defender
seed@VM: ~/.../Labsetup root@9bd8d19cf6fe: /var/www/defender
GNU nano 4.8 index.html

</body>

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

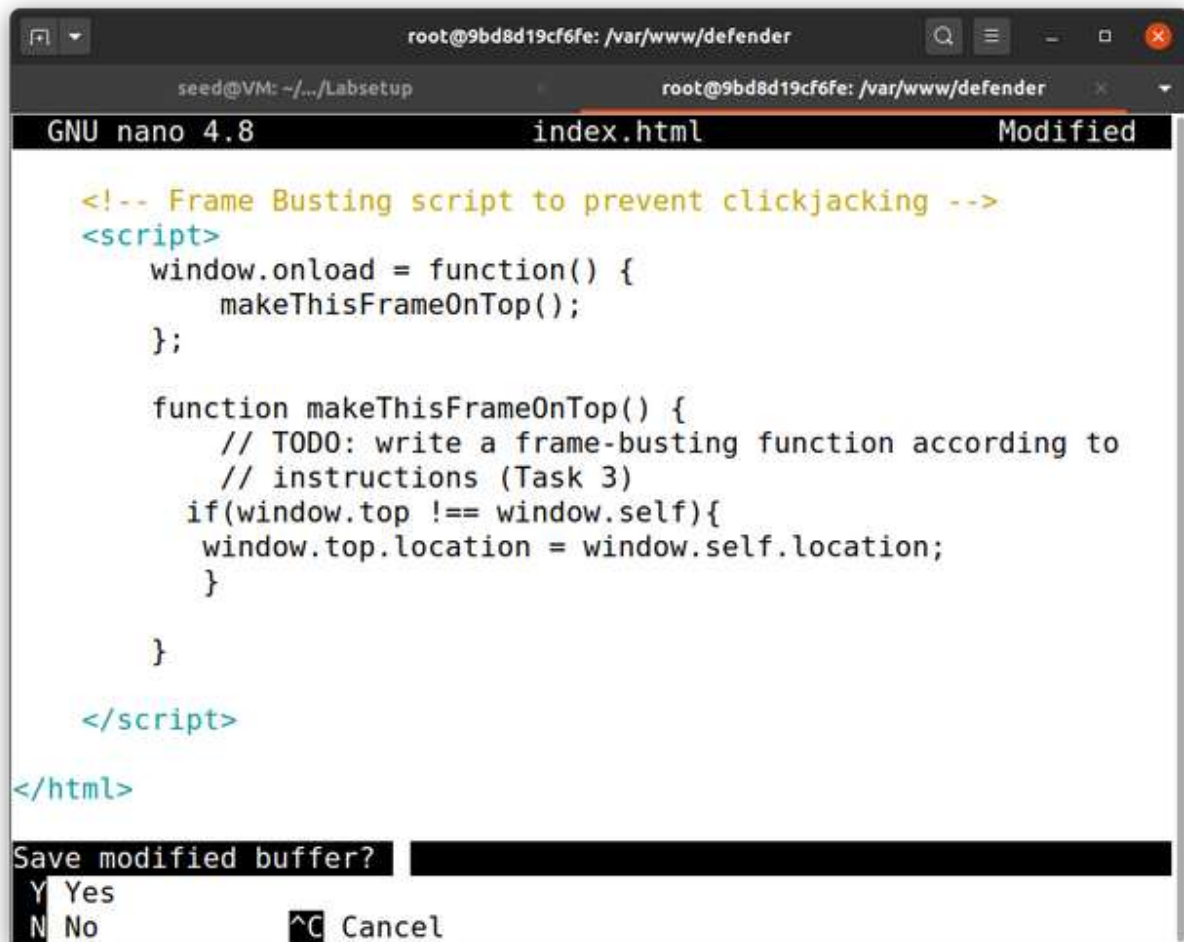
    function makeThisFrameOnTop() {
        // TODO: Write a frame-busting function according to
        // instructions (Task 3)

    }

</script>
</html>

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell
```

AFTER



```

root@9bd8d19cf6fe: /var/www/defender
seed@VM: ~/.../Labsetup root@9bd8d19cf6fe: /var/www/defender
GNU nano 4.8 index.html Modified

<!-- 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>

Save modified buffer?
Y Yes
N No      ^C Cancel

```

If you'd like, you can open 10,000, 000 browser tab and windows to make the hacker's machine crash.

Example of this is to add the following code..

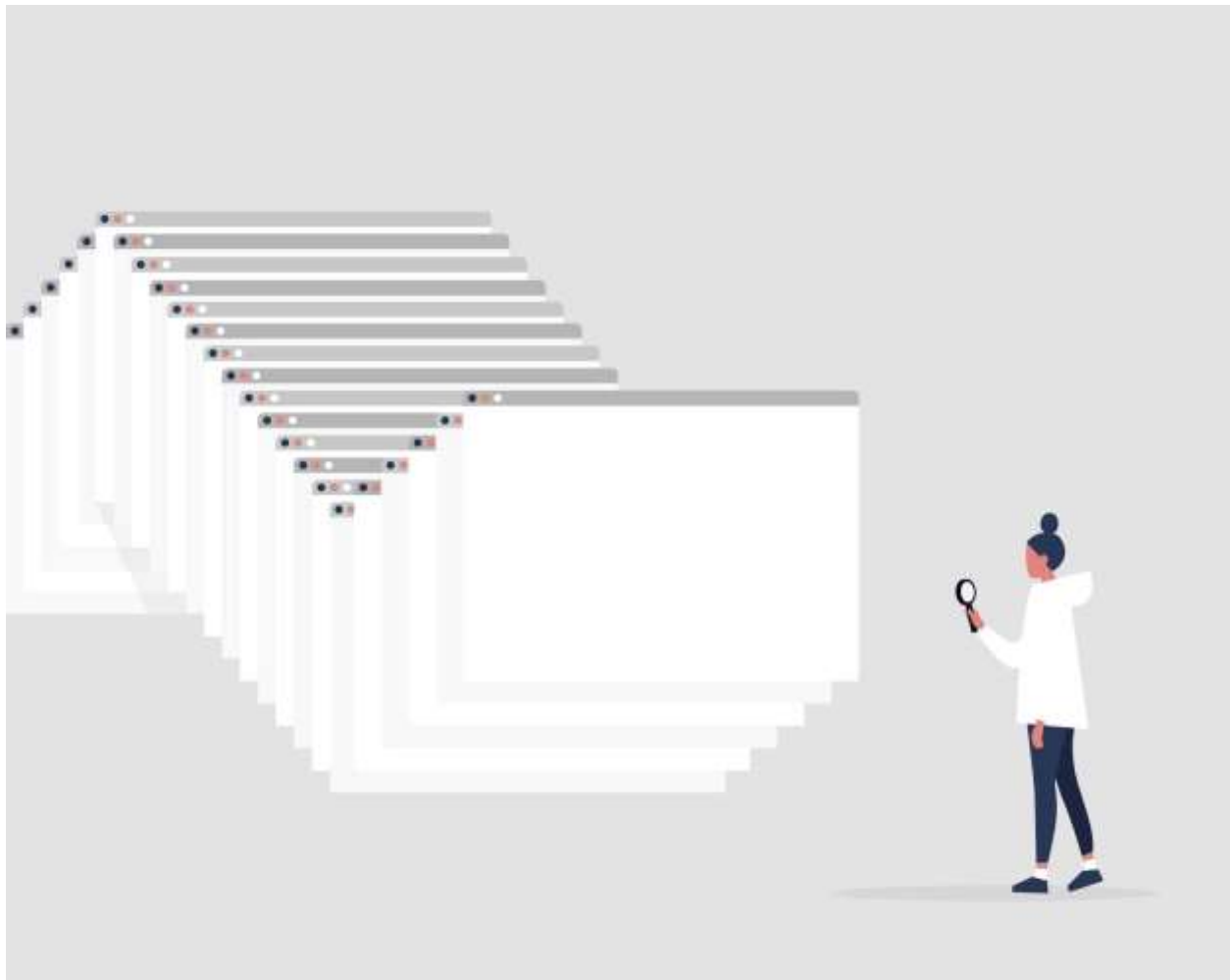
Code

```

{
  <script>
  If(window.top !== window.self){
  let hacker = {
    alert("wrong path hacker. If you click ok on this alert you device is gonna crash")
    setInterval(() => {
    let l = 0;
    let random_po = Math.floor(Math.random() * 100000)
    while(l < 10000000){

```

```
window.open("http://hack.com", "_blank", "left="+ random_po +", top="+ random_po +", width="+  
random_po +", height="+ random_po +")  
}  
})  
}  
}  
</script>  
}
```



QUESTIONS.

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

ANS => {

I get automatically redirected to the defenders page...

}

6 What happens when you click the button?

ANS => {

NOTHING HAPPENS. I JUST RELOADS

}

3.4 TASKK 4: ATTACKER COUNTERMEASURE (BUST THE BUSTER)

Disable the frame-busting script. Now let's explore how an attacker can create a workaround for frontend clickjacking defenses like frame busting. There are multiple workarounds, but one of the simplest in the current scenario is to add the sandbox attribute to the malicious iframe. Read more about the sandbox attribute on this page about iframes, then add the sandbox attribute to the iframe in attacker.html and answer the following questions.

// Would you like a simple knowledge about sandbox visite.

https://www.w3schools.com/tags/att_iframe_sandbox.asp

....SOLUTION....

Ok now add sandbox attribute in to your iframe in the attacker.html

With sandbox attribute added, you have the ability to disable, allow javascript to load in the iframe. So This means that the above code for the javascript won't work if the hacker disables javascript.

If you're an actual hacker. Do not try this on a website being built with react Js, like Facebook. Netflix etc... and my website. Cuz nothing will show. The reason is, Every thing being built with react is javascript.

<iframe src="..." sandbox/>

The screenshot shows a terminal window with the nano 4.8 editor open, editing a file named `attacker.html`. The code is as follows:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Attacker</title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <link href="./attacker.css" type="text/css" rel="stylesheet">
  </head>
  <body>
    <!-- TODO: place your iframe HERE (Task 1) -->
    <iframe sandbox width="100%" style="height: 100vh" frameborder="1">
    <!-- The malicious button's html code has already been provided in the attacker.css file.
    Note that the button code must come after iframe code-->
    <button style="position: absolute; top: 30%; left: 30%;" onclick="alert('XSS Successful!')">Click Me!
  </body>
</html>

```

Below the editor, a terminal prompt shows the command `cat attacker.html` and its output, which is the same HTML code. At the bottom of the terminal window, a browser window is visible, displaying a website with a Black Friday advertisement for Adobe Creative Cloud.

Now The 3.3 task has been disabled ...

QUESTIONS.

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

ANS => {

It disable javascript from running in the iframe...

}

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


```
ANS => {
  The website still loads....
}
```

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

```
Ans => {
  You get redirected to the You have been hacked page.
}
```

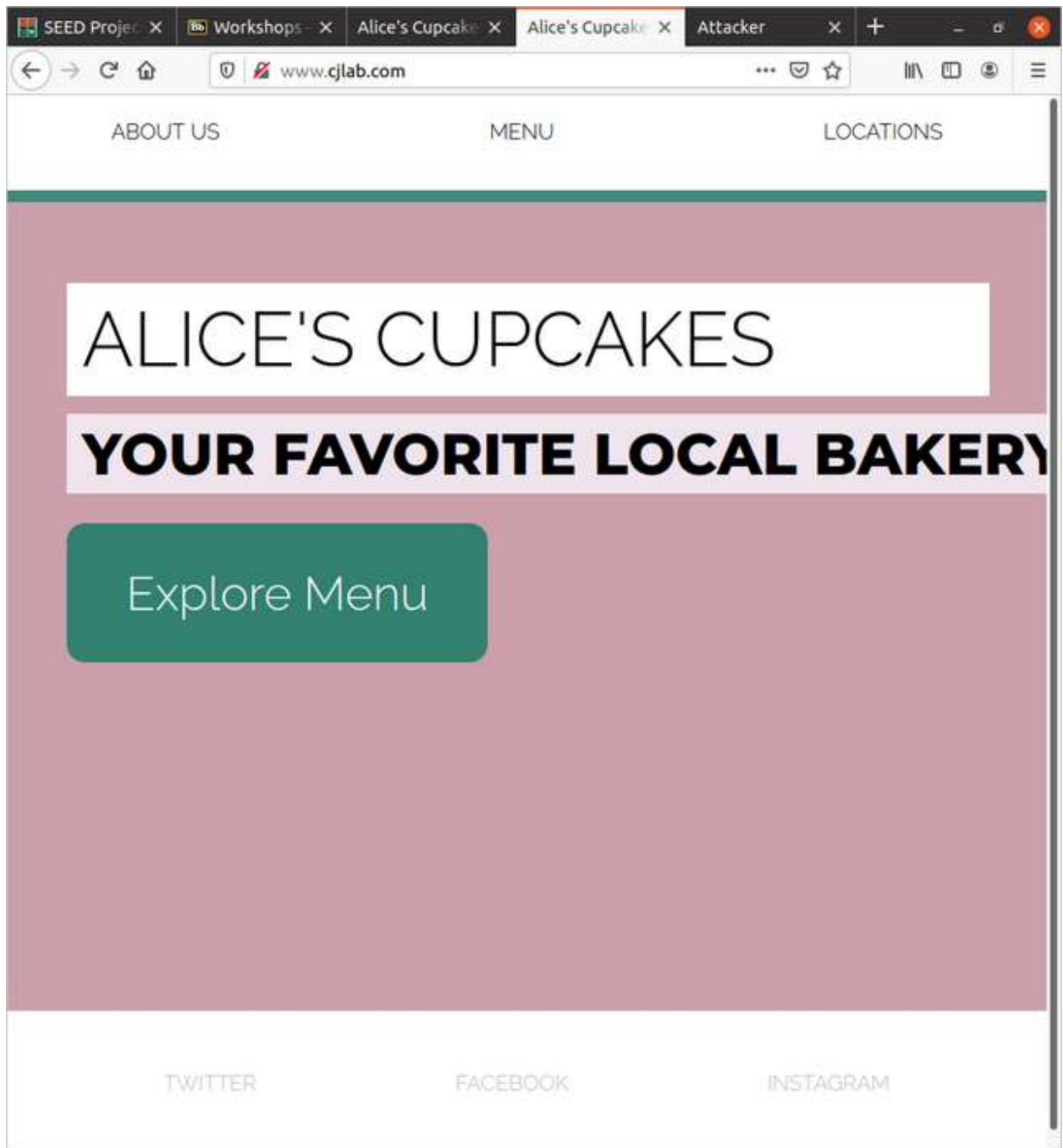
The screenshot shows a terminal window with the nano editor open, editing a file named `attacker.html`. The code is as follows:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Attacker</title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link href="./attacker.css" type="text/css" rel="stylesheet">
  </head>
  <body>
    <!-- TODO: place your iframe HERE (Task 1) -->
    <iframe sandbox width="100%" style="height: 100vh" frameborder="1">
    <!-- The malicious button's html code has already been provided in the previous task.
    Note that the button code must come after iframe code -->
    <button style="position: absolute; top: 30%; left: 30%;" onclick="location.href='https://www.example.com/hacked';">Click here
  </body>
</html>

```

Below the editor, a terminal prompt shows the command `cat attacker.html` being executed, displaying the same code. The browser window below shows a website with a button that triggers a redirect to a 'You have been hacked' page.



3.5 TASK 5: THE ULTIMATE BUST.

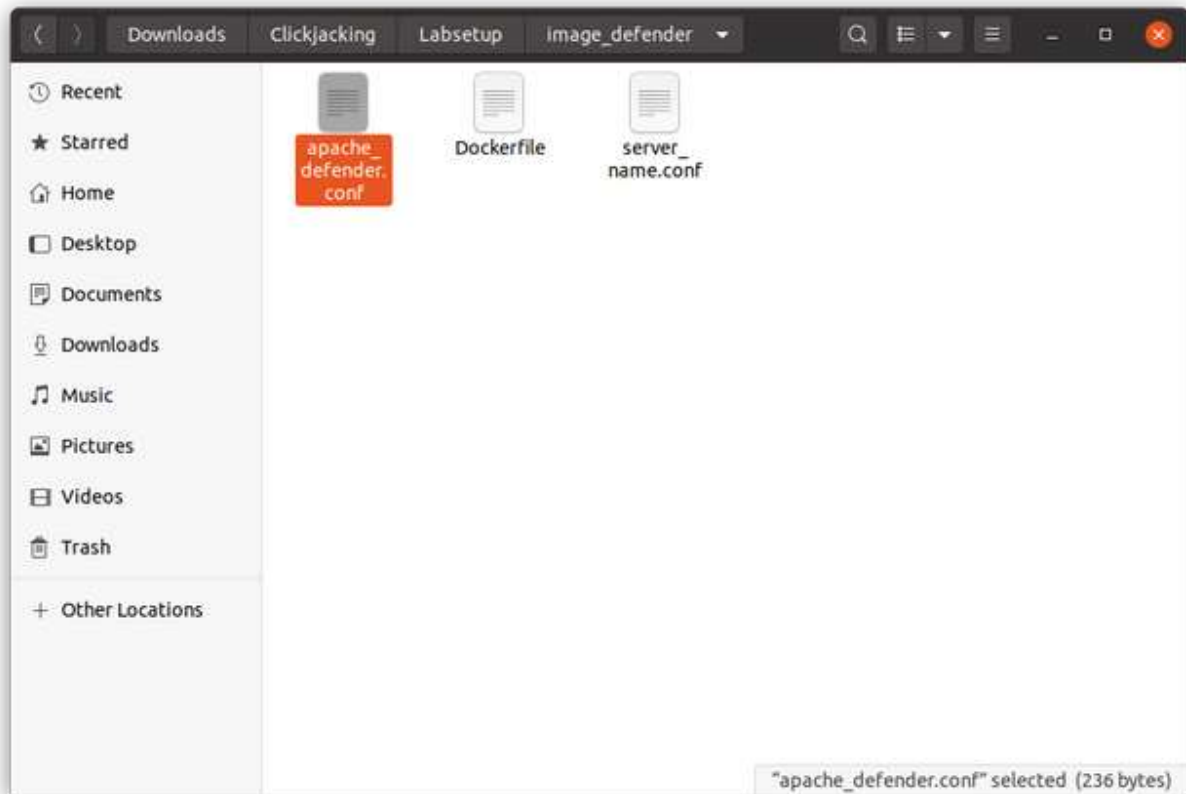
As we saw in the previous Task, front-end defenses such as frame busting can be directly circumvented by other front-end settings on the attacker's webpage and are therefore not sufficient to prevent clickjacking. To prevent clickjacking attacks more comprehensively, we need to set up back-end (i.e. server-side) defenses. Modern websites can cooperate with common browsers to provide such

defenses. The front-end attacks presented in previous Tasks all rely on the ability of an attacker's webpage code (running in a victim user's browser) to fetch a benign website's content before the benign webpage code has a chance to execute any front-end defenses. To block this capability, special HTTP headers have been created that specify to browsers the circumstances under which a website's content should or should not be loaded. By providing one of these HTTP headers with its response to a request for content, the website can instruct browsers to only display the content according to specific matching rules designed to exclude clickjacking attack scenarios. These headers are not part of the official HTTP specification, but are processed by many common browsers. One such header is called "X-Frame-Options", and a newer, more popular one is called "ContentSecurity-Policy". This header can include a diverse set of key-value directives to implement a site's Content Security Policy (CSP) with the intention of stopping many common attacks while perserving the desired content-sharing behavior of the site. The CSP header directive relevant for preventing clickjacking is "frame-ancestors", which specifies the valid parents that may embed a page in a frame. You can read more about the XFO attribute [here](#), and more about CSPs [here](#). Modify the defender's response headers. Open the Apache configuration file for the defender's website (apache defender.conf in the defender's image folder). Uncomment the lines that specify the HTTP response headers served with the page, and substitute appropriate text in order to prevent the clickjacking attack. Specifically, set the X-Frame-Options (XFO) header to the value "DENY" and the Content-SecurityPolicy (CSP) header to contain the directive "frame-ancestors 'none'".

...SOLUTION...

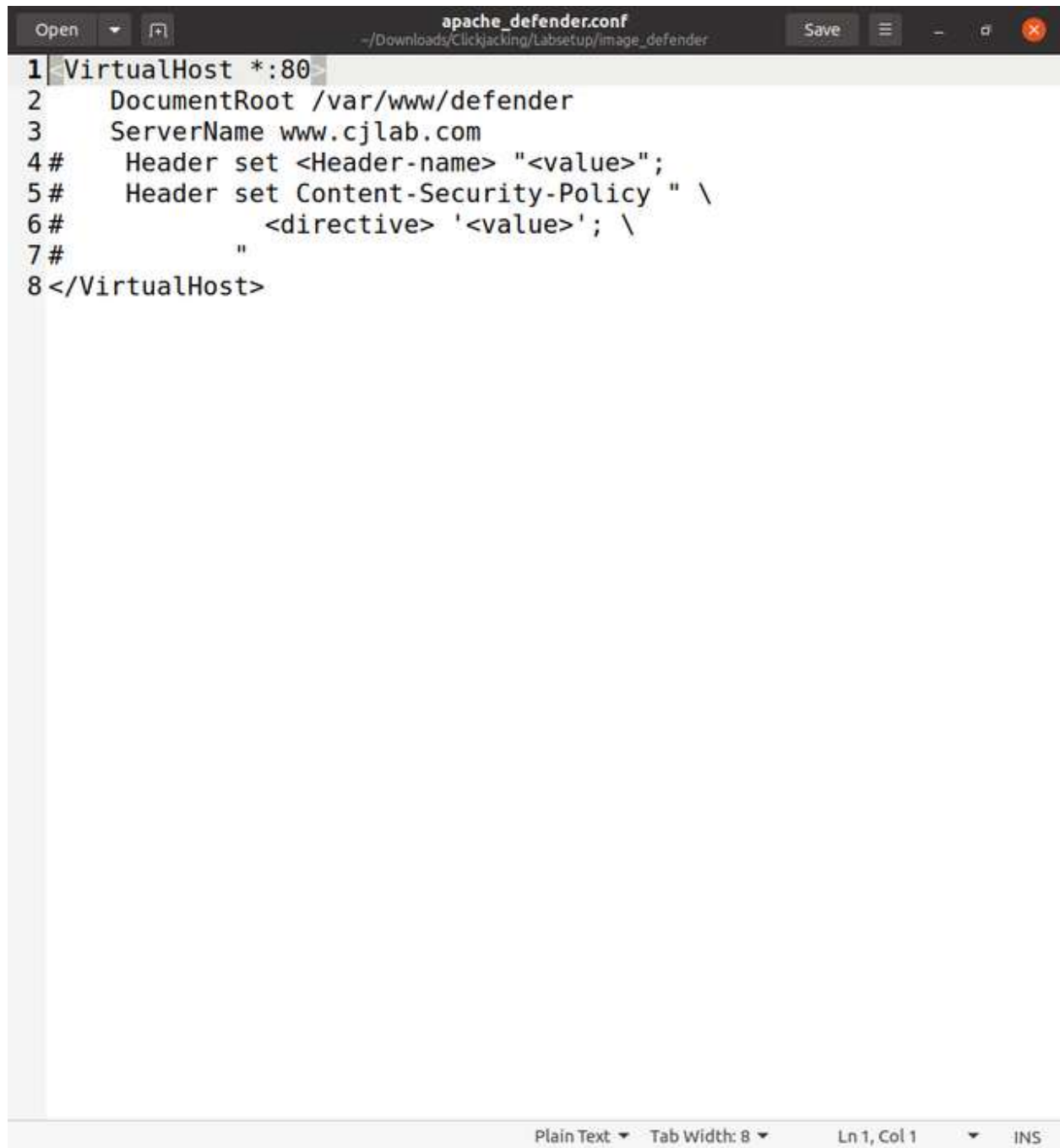
I know it's hard to find **the apache_defenders.config** file. It's very simple to find. Just open you file manager in you virtual machine and click on **labsetup** and click on the **Image_defender** folder. **Ans woala** ... you saw it right?

Open the apache with texteditor.. i.e. right click on the apache_defender.config file and open it in your text editor..



{PROFESSOR, YOU KNOW I CAN BY PASS IFRAME-OPTIONS RIGHT, CHECK THIS LINK OUT AND SEE HOW <https://www.npmjs.com/package/x-frame-bypass> this can by pass both sameorigin & deny }

BEFORE



The image shows a text editor window titled 'apache_defender.conf' with a file path of '~/.Downloads/Clickjacking/Labsetup/image_defender'. The editor contains the following configuration for a VirtualHost:

```
1<VirtualHost *:80>
2    DocumentRoot /var/www/defender
3    ServerName www.cjlab.com
4#    Header set <Header-name> "<value>";
5#    Header set Content-Security-Policy " \
6#        <directive> '<value>'; \
7#        "
8</VirtualHost>
```

The status bar at the bottom indicates 'Plain Text', 'Tab Width: 8', 'Ln 1, Col 1', and 'INS'.

After

Just copy this code here and paste it in the apache

```
{
```

CLICK-JACKING-SEED-LAB

<VirtualHost *:80>

DocumentRoot /var/www/defender

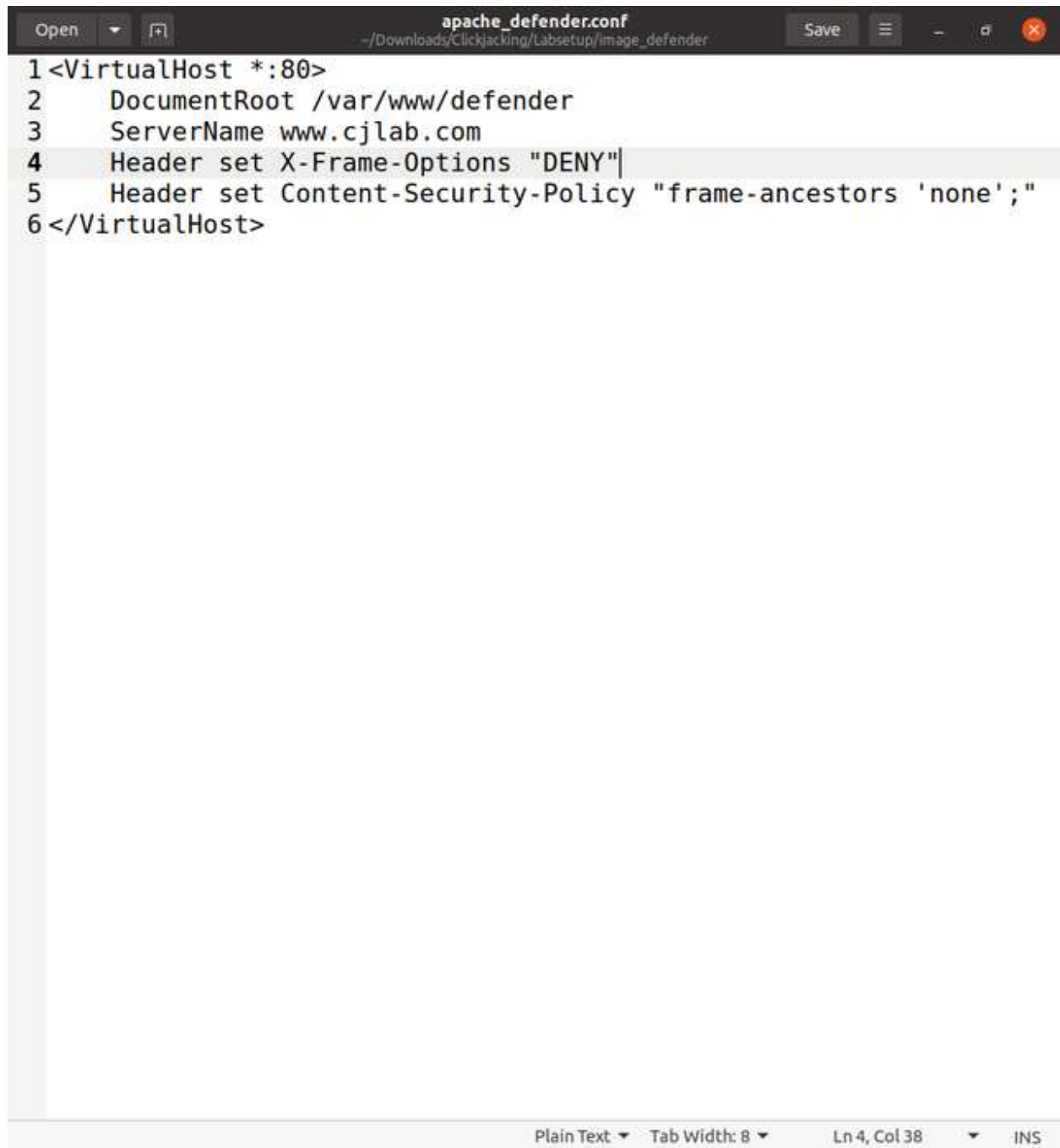
ServerName www.cjlab.com

Header set X-Frame-Options "DENY"

Header set Content-Security-Policy "frame-ancestors 'none';"

</VirtualHost>

}



The image shows a text editor window titled 'apache_defender.conf' with a file path of '~/.Downloads/Clickjacking/Labsetup/image_defender'. The editor contains the following configuration for a VirtualHost:

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

The status bar at the bottom indicates 'Plain Text', 'Tab Width: 8', 'Ln 4, Col 38', and 'INS'.

QUESTIONS...

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

ANS => {

These attributes are SAMEORIGIN, **ALLOW-FROM** AND DENY, It is set to DENY to stop it from showing in any website even in the website that hosted it.

}

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

ANS => {

THE CONTENT POLICY ARE **NONE, SELF, STRICT-DYNAMIC, REPORT-SAMPLE, UNSAFE-INLINE, UNSAFE-EVAL, UNSAFE-HASHES, UNSAFE-ALLOW-REDIRECT**. It is turned to NONE so it won't allow loading of any resources.

}

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?

ANS => {

YOU DON'T NEED TO CLICK THE BUTTON, THE USER HAS ALREADY SEEN A BROKEN PAGE, AND WILL NOT CLICK ANYTHING ELSE ON THAT PAGE. Ok let's answer this question. If you click on the button, nothing happens.

}



Blocked by X-Frame-Options Policy

An error occurred during a connection to www.helpmegeek.com.

Firefox prevented this page from loading in this context because the page has an X-Frame-Options policy that disallows it.