Workshop on
Web Security

Nov 29, 2023

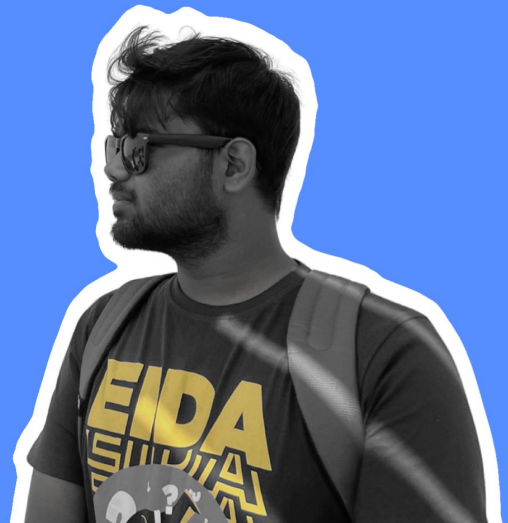
Introduction

Hi, I am **Ataf**

- BUET CSE'17 Batch, Ex-BUETSEC
- Secure Software Dev. @ OpenRefactory
  - Find & Fix Bugs in Open Source Projects
- Contributor/Member @ RevoltZero
  - Share & Practice Knowledge

GitHub : @fazledyn, @fazledyn-or

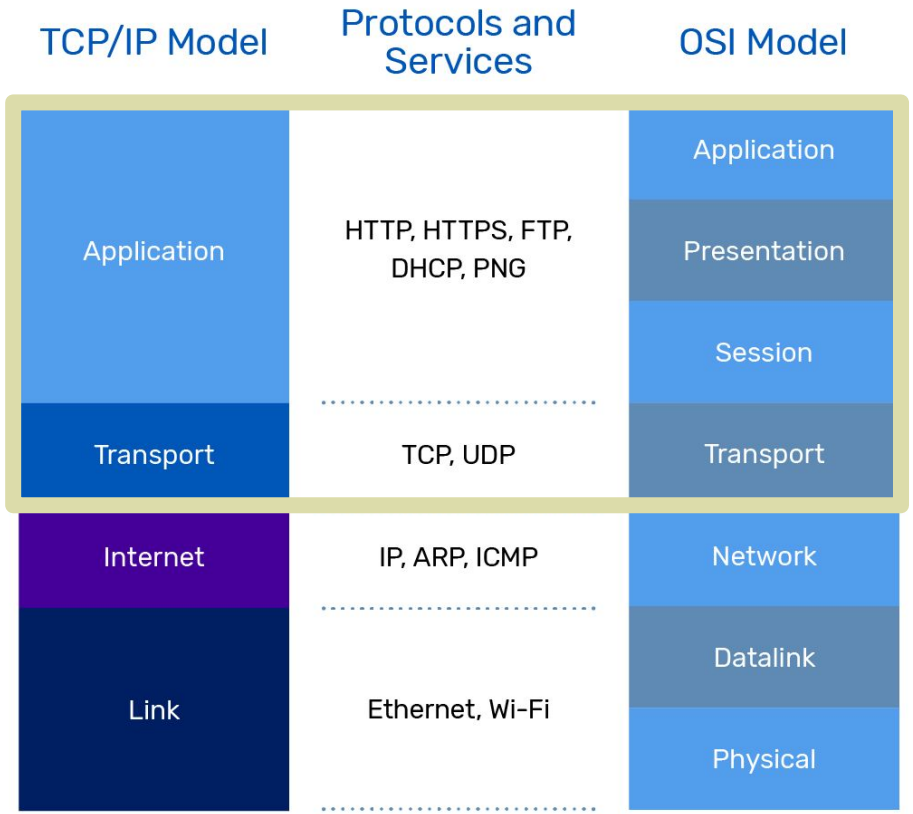
Email : rabidahamed@gmail.com



We’ll Be Doing Today

- Different Types of HTTP Requests
- The Browser Environment
- App #1: Cookie
- App #2: Flask Debug Mode
- App #3: Fuzzing
-

Network Model



For Web Security

- HTTP/HTTPS
- TCP, UDP
- FTP

# of Ports
65536

0-65535

Different Types of HTTP Requests

Mainly FOUR types of request (methods)

- GET
- POST
- PUT
- DELETE

Others

- PATCH
- HEAD
- OPTIONS and so on

Read more: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Different Types of HTTP Requests

Mainly FOUR types of request (methods)

- GET → to GET some data from server
- POST → to POST some data to server
- PUT
- DELETE

Others

- PATCH
- HEAD
- OPTIONS and so on

Read more: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Different Types of HTTP Requests

Mainly FOUR types of request (methods)

- GET → to GET some data from server
  - Parameters are in URL
  - No payload/body
  - Limited URL length

  - https://store.com/find_products?name=Tshirt&size=XL

- POST → to SEND some data to server
  - Parameters are in URL
  - Data in payload/body

  - https://store.com/add_product
    Data in payload

The Browser Environment

Let’s start with a sample code - Example #1

Let’s run the code

1) In terminal using NodeJS
2) In browser

```
1
2  const fetch = require("node-fetch");
3  const url = 'https://example.com';
4
5  fetch(url)
6  .then((res) => {
7      return res.text();
8  })
9  .then((data) => {
10     console.log(data);
11 })
12 .finally(() => {
13     console.log("fetch completed");
14 })
15
```

The Browser Environment

Let’s start with a sample code -

Example #1

Let’s run the code

1) In terminal using NodeJS

2) In browser

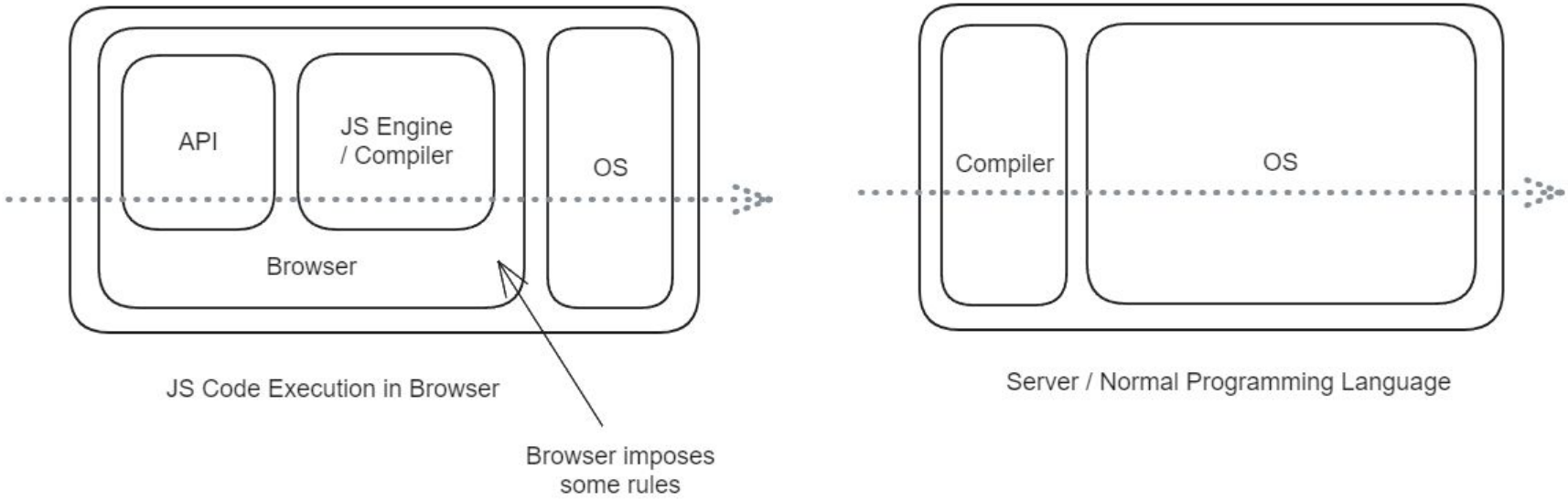
Same code but

why different output?

```
1
2   const fetch = require("node-fetch");
3   const url = 'https://example.com';
4
5   fetch(url)
6   .then((res) => {
7       return res.text();
8   })
9   .then((data) => {
10      console.log(data);
11  })
12  .finally(() => {
13      console.log("fetch completed");
14  })
15
```

The Browser Environment

Depends on How JavaScript Actually Works -



The Browser Environment

1) So, Different Environment = Different Output

2) But what are those rules?

- No File System Access *
- No Socket Connection
- No Threading/Subprocess
- Follow CSP, Same Origin Policy, etc.

3) Browser Plugins

Read More: https://infosec.mozilla.org/guidelines/web_security

The Browser Environment

Let’s start with a sample code -

Example #2

Let’s run these codes

```python
import requests

url = "http://localhost:5000/"
res = requests.get(url)
print(res.text)

```

```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/")
def index():
    print("****************")
    print(request.headers)
    print("****************")
    return "Hello World"

if __name__ == "__main__":
    app.run(debug=True)
```

The Browser Environment

Let’s start with a sample code -

Example #2

Let’s run these codes

Different outputs here too!

```python
import requests

url = "http://localhost:5000/"
res = requests.get(url)
print(res.text)

```

```python
from flask import Flask, request

app = Flask(__name__)

@app.route("/")
def index():
    print("****************")
    print(request.headers)
    print("****************")
    return "Hello World"

if __name__ == "__main__":
    app.run(debug=True)
```

Port Scanning Using `nmap`

1) Follow On Screen
    a) Server 1
    b) Server 2

2)

Example App #1: Cookie

Example App #1: Cookie

1) Challenge URL: http://23.251.153.217:6000/

2) Follow On Screen

Example App #1: Cookie

1) Challenge URL: http://23.251.153.217:6000/

2) Follow On Screen

3) JWT: JSON Web Tokens
   - Format: {scheme/algo}.{payload/data}.{signature}
   - If you find anything, try breaking it

Example App #1: Cookie

1) Challenge URL: http://23.251.153.217:6000/

2) Follow On Screen

3) JWT: JSON Web Tokens
   - Format: {scheme/algo}.{payload/data}.{signature}
   - If you find anything, try breaking it

4) Don’t trust JWTs blindly! Always verify the signature!

Read more: https://jwt.io/

Example App #2: Flask Debug Mode

Example App #2: Flask Debug Mode

1) Challenge URL: http://23.251.153.217:3000/

2) Follow On Screen

Example App #2: Flask Debug Mode

1) Challenge URL: http://23.251.153.217:3000/

2) Follow On Screen

3) Follow these resources, solve on your own

https://www.bengrewell.com/cracking-flask-werkzeug-console-pin/

https://www.youtube.com/watch?v=jwBRgaIRdgs

Example App #3: Fuzzing

1) Challenge URL: http://23.251.153.217:8000/
2) Follow On Screen

- Fuzzing Tools:
  - dirb
  - gobuster
  - nikto

Example App #3: Fuzzing

1) Challenge URL: http://23.251.153.217:8000/
2) Follow On Screen

- Fuzzing Tools:
  - $ dirb http://23.251.153.217:8000 /usr/share/dirb/wordlists/common.txt
  - $ gobuster
  - $ nikto
- Difference
  - `dirb` can search iteratively
  - `gobuster` is multithreaded

GitHub Codespaces

GitHub Codespaces

1) Install GitHub CLI
2) Install VSCode (not Codium)
3) Login to your account
4) Follow On Screen

GitHub Codespaces

1) Install GitHub CLI
2) Install VSCode (not Codium)
3) Login to your account
4) Follow On Screen

Use it as a VPS!

- Run Scripts, Fuzzers
- Host your webapps

The Hacker's Approach

a) Perform automation, save time
   i) nmap, fuzzers, scripts, ZAP

b) Do it on a server than your PC
   i) Better bandwidth
   ii) Better performance

c) Perform recon
   i) Censys, Shodan for finding info
   ii) Get IP address !

The Developer’s Approach

a) Use SSL/TLS Certificates - *Free*
   i) ZeroSSL, Cloudflare, etc.

b) Use DDoS Protection
   i) Cloudflare - *Free*
   ii) Never expose your IP address
      - If HTTP exposed, can perform DDoS on IP
      - DDoS Protection saves only requests coming to domain

c) Don’t expose unnecessary ports. Follow HSTS.

d) Use Encryption, JWTs carefully

The Developer’s Approach

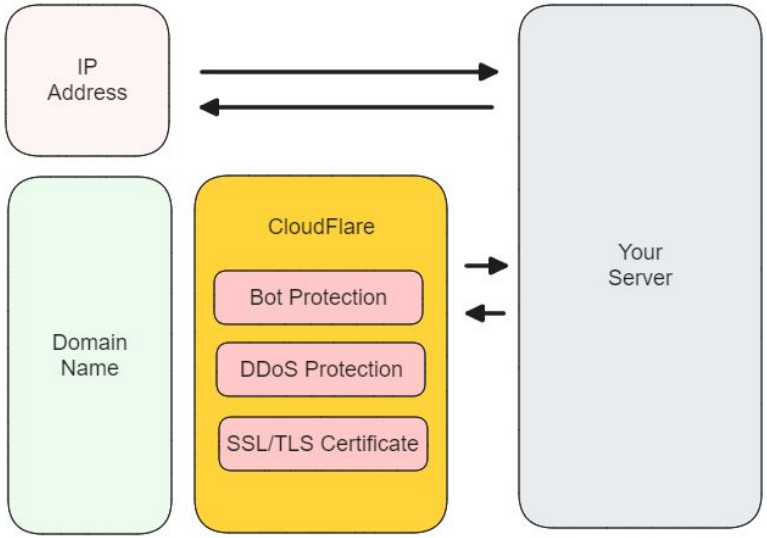
b) Use DDoS Protection

i) Cloudflare - *Free*

ii) Never expose your IP address

- If HTTP exposed, can perform DDoS on IP

- DDoS Protection saves only requests coming to domain



Questions?

- Knock Anytime on Discord
- BUET Cyber Drill Practice Zone @ Facebook

- Resources To Study
  - CS 253: Web Security by Stanford CS
  - PortSwigger Labs
  - MDN Docs

- Contents: https://github.com/buetsec/web-security-workshop-2023

End Of Slide

Thank You