Web Application Security

Build.Break.Learn

About Me

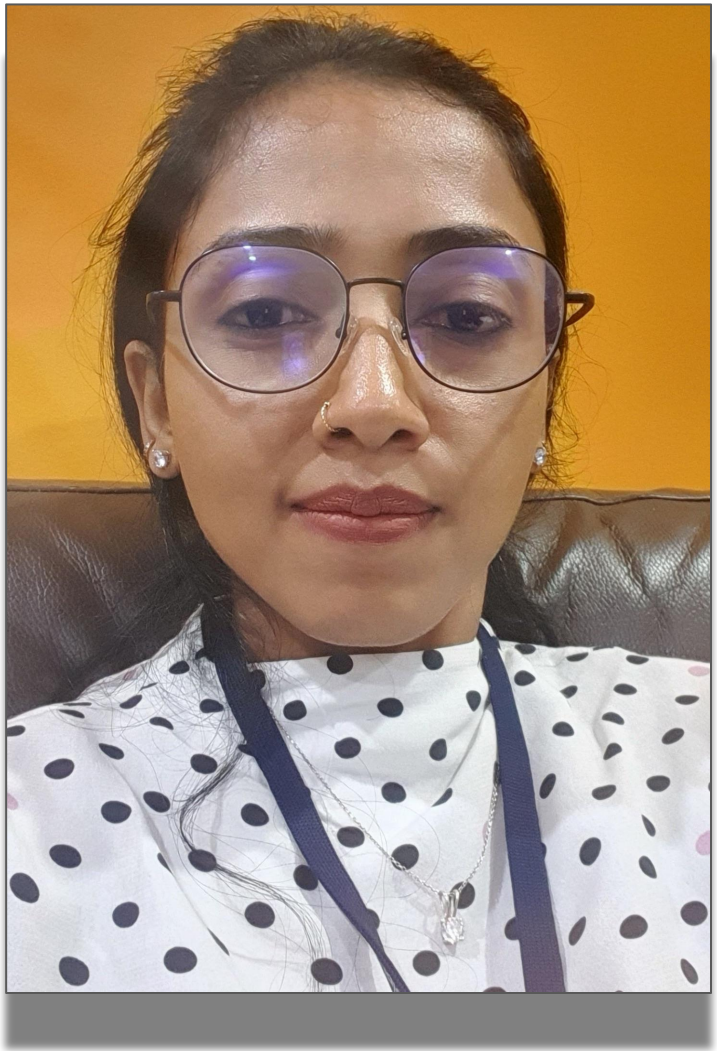
Riddhi Shree

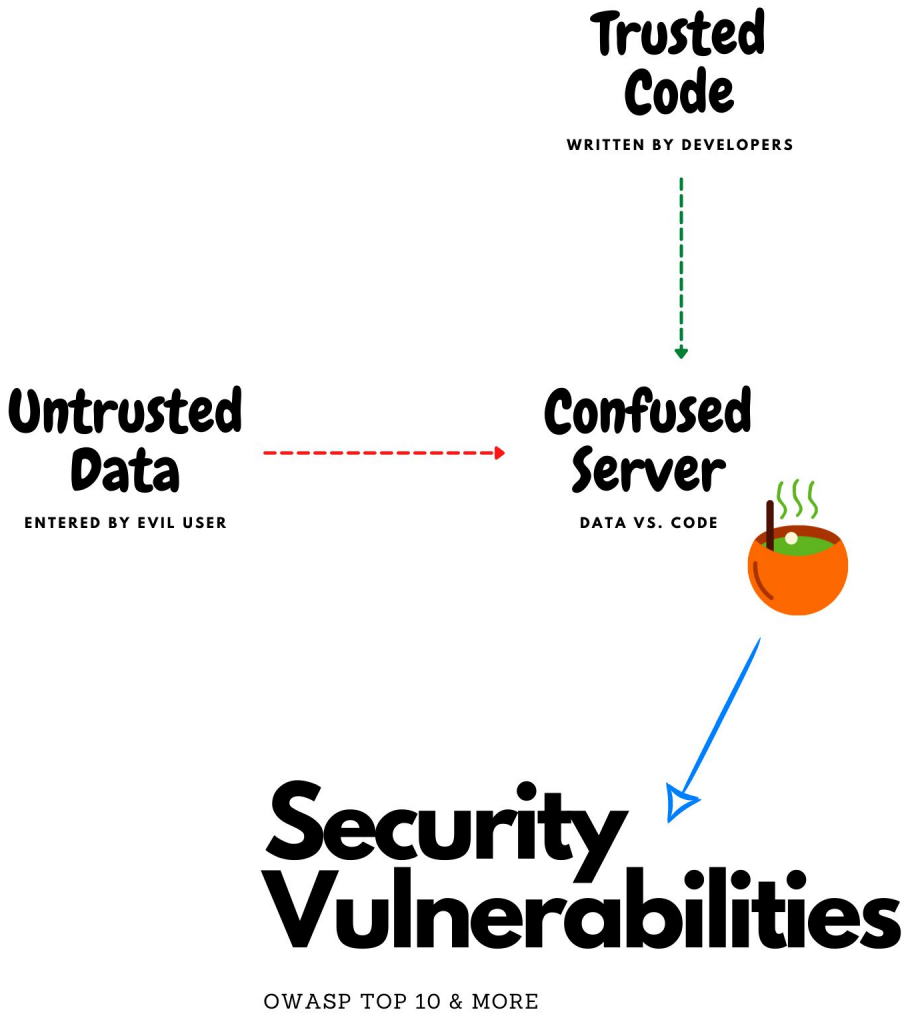
Twitter: @_riddhishree

GitHub: github.com/riddhi-shree

Medium: riddhi-shree.medium.com

Website: riddhishree.com




Trusted Code
WRITTEN BY DEVELOPERS
Untrusted Data
ENTERED BY EVIL USER
Confused Server
DATA VS. CODE
Security Vulnerabilities
OWASP TOP 10 & MORE

DAY-2

Vulnerability Scanning

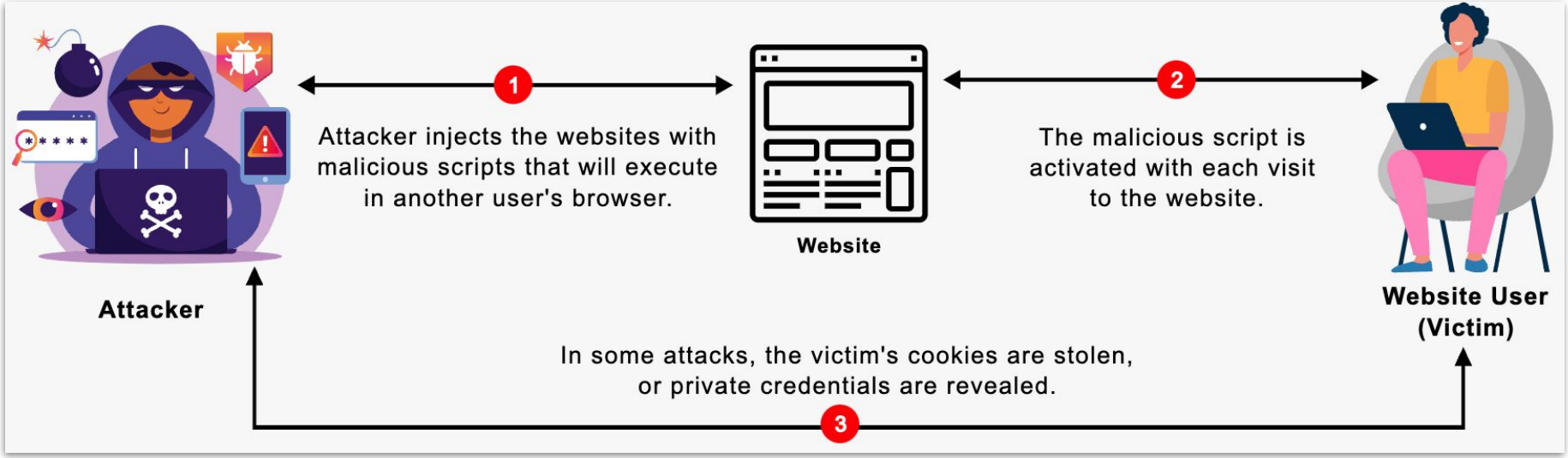
What to Expect?

Now that we have gained an understanding about common web security vulnerabilities, let’s understand:

- Client-Side Vulnerabilities
  - Cross-Site Scripting
  - Unvalidated Redirects and Forwards
  - Cross-Site Request Forgery
- Web Vulnerability Scanning Tools

Client-Side Vulnerabilities

Cross-Site Scripting



Hands-On: XSS

- Follow instructions given here: “web-app-security-nullcon2023-lab/vulnerabilities/**xss**/README.md”

Subresource Integrity

1. Create an **HTML** file:

```
<html>
<head>
<script=”http://www.external.com/demo.js”>
</head>
<body onload=”Execute()”>
</body>
</html>
```

2. Create a **javascript** file:

```
function Execute(){
alert(“This is non-malicious… so far!”);
}
```

Subresource Integrity

- Obtain a **hash** value:

```
openssl dgst -sha384 -binary demo.js | openssl base64 -A
```

- Use this hash value in the “**<script>**” tag:

```
<script src=”http://www.external.com/demo.js”
integrity=”sha384-theentirehashvalue”
crossorigin=”anonymous”></script>
```

- Modify the contents of the javascript file, and notice what happens.

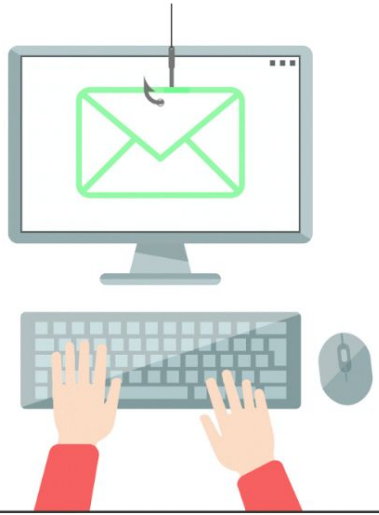
Demo: Subresource Integrity

Either try this simple example on your local system, or, watch a nice demo here:

https://www.youtube.com/watch?v=x5wX88YUf68

Unvalidated Redirects and Forwards

1) User receives phishing mail with a link pointing to a known page, with a redirect parameter containing a link to the attacker site.

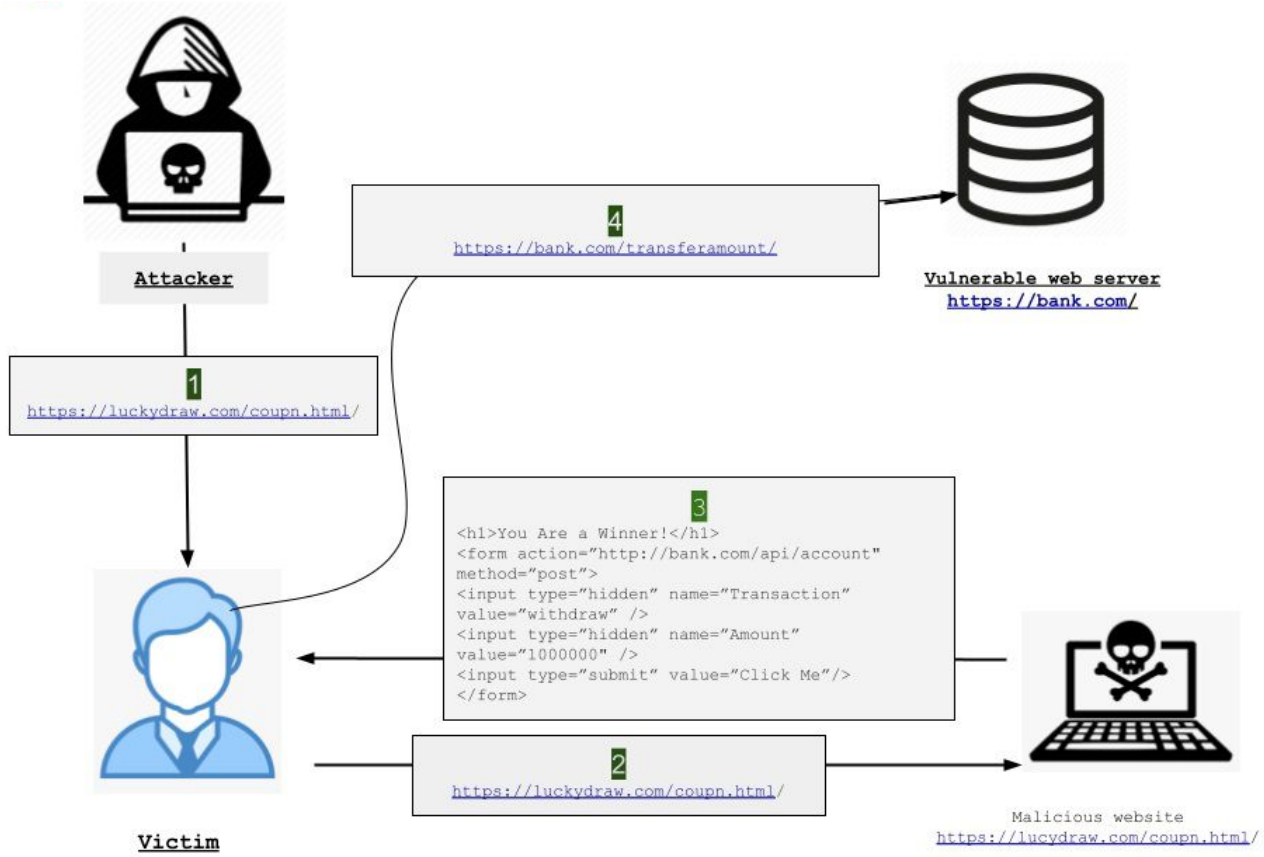


2) User clicks on the link and is directly being redirected to the attacker's page.

3) Page looks like the trusted page and has a login prompt, but is actually the website under the attacker's control.

- Read this blog - https://www.riddhishree.com/posts/breaking_hashes/

Cross-Site Request Forgery



Hands-On: CSRF

- Follow instructions given here:
  “web-app-security-nullcon2023-lab/vulnerabilities/**csrf.md**”

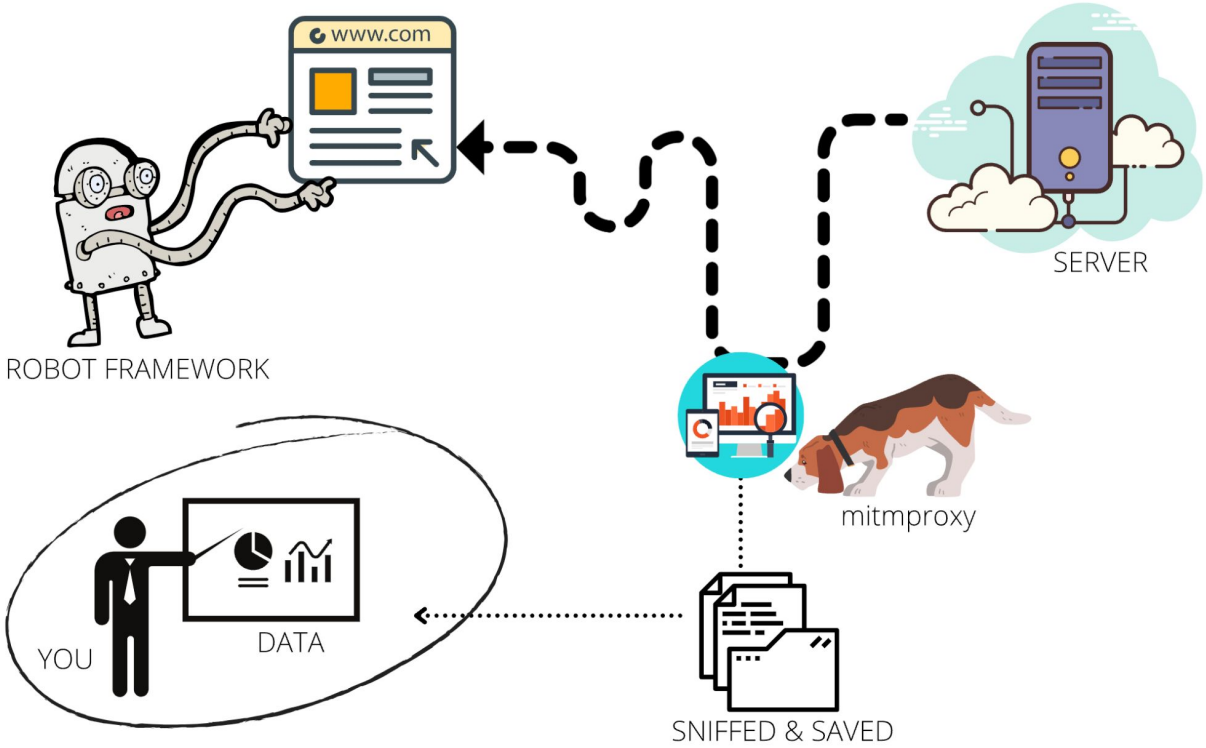
CSRF Tokens

- Make the requests unpredictable by including randomness, i.e., by using CSRF tokens.
  - Tie it to user session
  - Should be unpredictable
  - Perform strict server-side validation before executing a sensitive request on the server
- According to Portswigger:
  - “You should use a cryptographically secure pseudo-random number generator (CSPRNG), seeded with the timestamp when it was created plus a static secret.”
- https://portswigger.net/web-security/csrf/bypassing-samesite-restrictions

CORS Misconfiguration

- Same-origin policy (SOP) limits the ability for a website to interact with reaources outside of the source domain.
- It presents challenges to APIs and microservices which have legitimate use cases for accessing and sharing information between domains.
- Cross-origin resource sharing (CORS) is a controlled relaxation of SOP.
- CORS rules allow domains to specify which domains can request information from them by adding specific HTTP headers in the response.
- **Access-Control-Allow-Origin**: This header specifies the allowed domains to read the response contents. The value can be either a wildcard character (*), which indicates all domains are allowed, or a comma-separated list of domains.
- **Access-Control-Allow-Credentials**: This header determines whether the domain allows for passing credentials — such as cookies or authorization headers in the cross-origin requests.
  The value of the header is either True or False. If the header is set to “true,” the domain allows sending credentials. If it is set to “false,” or not included in the response, then it is not allowed.

Man-in-the-Middle (MitM)



Setup Your Local Lab (Linux / macOS)

- `cd web-app-security/TrainingContent/target/local_playground`
- `./setup.sh`
- `netstat -AaLlnW`
- `./open_apps.sh`

```
mirage@192 local_playground % netstat -AaLlnW
Current listen queue sizes (qlen/incqlen/maxqlen)
Socket           Flowhash Listen         Local Address
26dda9ff6a0129e1        0 0/0/5          *.5555
26ddaa0903ca54e9        0 0/0/128        *.3000
26ddaa0903c7bce9        0 0/0/128        *.8899
26ddaa0903c7ace9        0 0/0/128        *.9090
26ddaa0903c79ce9        0 0/0/128        *.4443
26ddaa0903c7cce9        0 0/0/128        *.6001
26ddaa0903ca84e9        0 0/0/128        *.6602
26ddaa0903cb84e9        0 0/0/128        *.4444
26ddaa0903ca7ce9        0 0/0/128        *.4280
26ddaa0903cb54e9        0 0/0/128        *.3333
26ddaa0903ca6ce9        0 0/0/128        *.3001
26ddaa0903ca5ce9        0 0/0/128        *.2222
26ddaa0903ca64e9        0 0/0/128        *.1234
26ddaa0903ca74e9        0 0/0/128        *.8080
26ddaa0903cb8ce9        0 0/0/128        *.5432
26ddaa0903cb74e9        0 0/0/128        *.80
26ddaa0903cb6ce9        0 0/0/128        *.8443
26ddaa0903cb64e9        0 0/0/128        *.5000
26dda9ff6a3db751        0 0/0/128        *.5000
26ddaa0903cb5ce9        0 0/0/128        *.7000
26dda9ff6a3dc261        0 0/0/128        *.7000
26ddaa0903cb7ce9        0 0/0/128        *.8834
26dda9ff6a3dcd71        0 0/0/128        *.8834
```

Check if test applications can be accessed locally.

1. API (CRUD) - http://127.0.0.1:1234/
2. Misconfiguration - http://127.0.0.1:2222/
3. Juice Shop - http://127.0.0.1:3001/#/
4. RCE - http://127.0.0.1:3333/
5. DVWA - http://127.0.0.1:4280/login.php
6. SSRF - http://127.0.0.1:4444/
7. **XSS** - http://127.0.0.1:5555/
8. SQLi - http://127.0.0.1:6001/
9. Security Ninja - http://127.0.0.1:8899/
10. Shellshock - http://127.0.0.1:9090/
11. Security Shepherd - https://127.0.0.1:443/setup.jsp
12. Heartbleed - https://127.0.0.1:4443/

Tool Check

- Nuclei
- OWASP ZAP
- Burp Suite
- OpenVAS
- Robotframework


nuclei

Vulnerability Scanning Using Nuclei

- https://docs.nuclei.sh/getting-started/install
- Run using Docker image:

```
docker run --rm projectdiscovery/nuclei -u
http://localhost -jsonl
```

- Scan multiple targets:

```
nuclei -l targets.txt
```

- `nuclei -u http://127.0.0.1 -tags cve -severity critical,high`

OWASP

Zed Attack Proxy

Run OWASP ZAP

- **docker-compose.yml**

```
version: '3'
services:
  zap:
    image: ghcr.io/zaproxy/zaproxy:stable
    user: "zap"
    ports:
      - "8888:8080"
      - "8090:8090"
    command: zap-webswing.sh
    volumes:
      - ./wrk:/zap/wrk/:rw
```

- Start ZAP

  `docker-compose up -d –build`

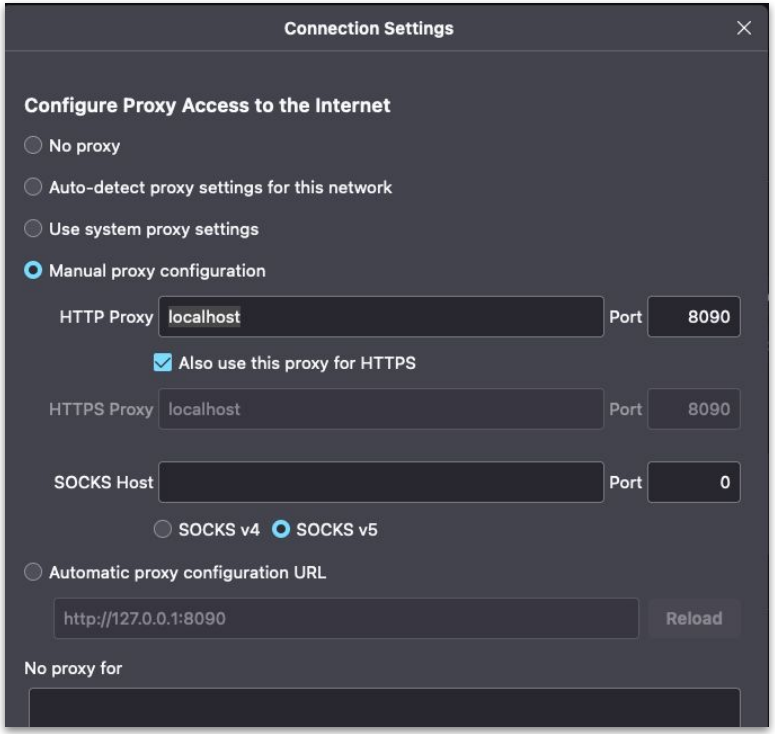
- Alternatively, you can also run

  `docker run -v $(pwd)/wrk:/zap/wrk/:rw -u zap -p 8888:8080 -p 8090:8090 --name zap -i ghcr.io/zaproxy/zaproxy:stable zap-webswing.sh `

- Navigate to **`http://localhost:8888/zap/`**

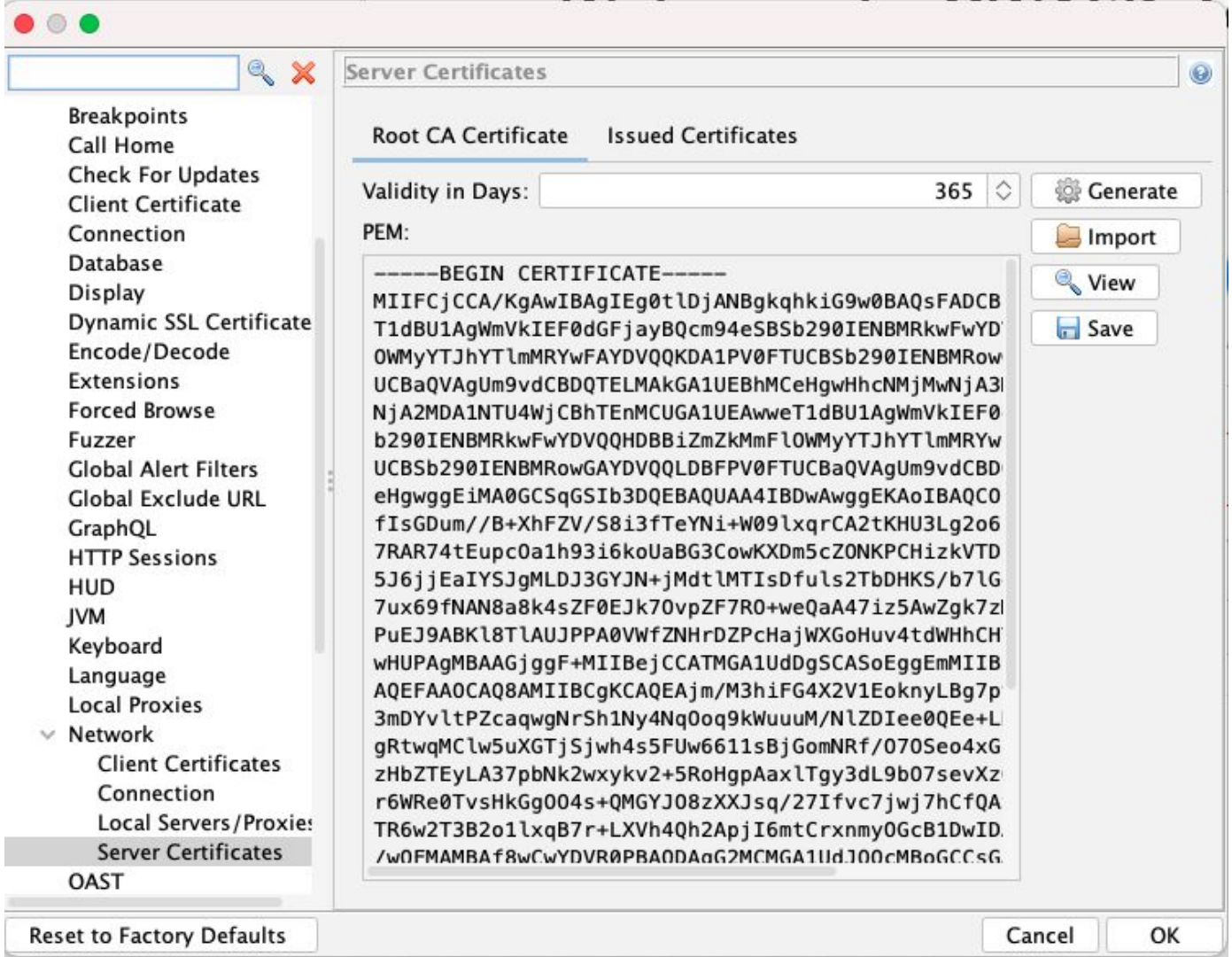
Configure Browser Proxy

- Create a new Firefox browser profile for running OWASP ZAP:
  - `firefox -P`
  - `/Applications/Firefox.app/Contents/MacOS/firefox -P`
  - `firefox.exe -P`
- Open a different browser/ browser profile, and configure it to use ZAP as a proxy (127.0.0.1:8090).
- https://www.zaproxy.org/docs/desktop/start/proxies/



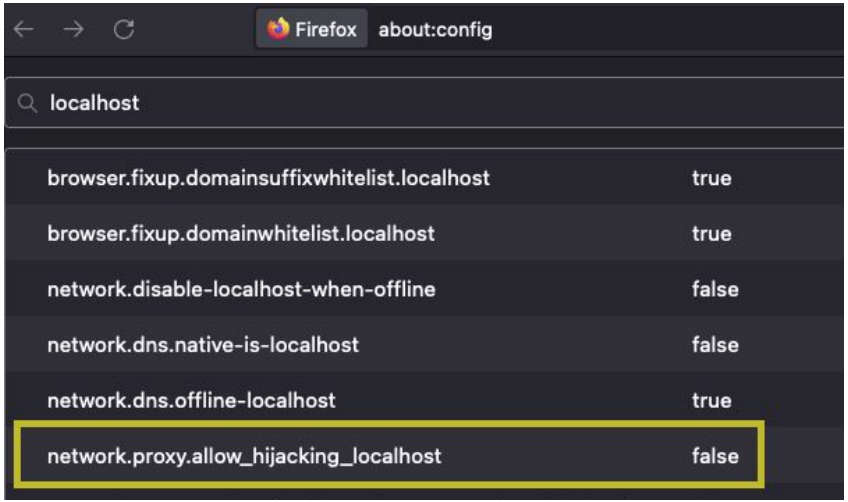
Import CA Certificate

- In your browser, import `zap_root_ca.cer` certificate from "`$(pwd)/wrk`" folder.
- Select the checkbox "`Trust this CA to identify websites.`"


Breakpoints
Call Home
Check For Updates
Client Certificate
Connection
Database
Display
Dynamic SSL Certificate
Encode/Decode
Extensions
Forced Browse
Fuzzer
Global Alert Filters
Global Exclude URL
GraphQL
HTTP Sessions
HUD
JVM
Keyboard
Language
Local Proxies
Network
Client Certificates
Connection
Local Servers/Proxies
Server Certificates
OAST
Server Certificates
Root CA Certificate
Issued Certificates
Validity in Days:
365
Generate
Import
View
Save
PEM:
-----BEGIN CERTIFICATE-----
MIIFCjCCA/KgAwIBAgIEg0tlDjANBgkqhkiG9w0BAQsFADCB
T1dBU1AgWmVkIEF0dGFjayBQcm94eSBSb290IENBMRkwFwYD
OWMyYTJhYTlmMRYwFAYDVQQKDA1PV0FTUCBSb290IENBMRow
UCBaQVAgUm9vdCBDQTELMAkGA1UEBhMCeHgwHhcNMjMwNjA3
NjA2MDA1NTU4WjCBhTEnMCUGA1UEAwweT1dBU1AgWmVkIEF0
b290IENBMRkwFwYDVQQHDBBiZmZkMmFlOWMyYTJhYTlmMRYw
UCBSb290IENBMRowGAYDVQQLDBFPV0FTUCBaQVAgUm9vdCBD
eHgwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCO
fIsGDum//B+XhFZV/S8i3fTeYNi+W09lxqrCA2tKHU3Lg2o6
7RAR74tEupcOa1h93i6koUaBG3CowKXDm5cZONKPCHizkVTD
5J6jjEaIYSJgMLDJ3GYJN+jMdtlMTIsDfuls2TbDHKS/b7lG
7ux69fNAN8a8k4sZF0EJk7OvpZF7RO+weQaA47iz5AwZgk7z
PuEJ9ABKl8TlAUJPPA0VWfZNHrDZPcHajWXGoHuv4tdWHhCH
wHUPAgMBAAGjggF+MIIBejCCATMGA1UdDgSCASoEggEmMIIB
AQEFAAOCAQ8AMIIBCgKCAQEAjm/M3hiFG4X2V1EoknyLBg7p
3mDYvltPZcaqwgNrSh1Ny4NqOoq9kWuuuM/NlZDIee0QEe+L
gRtwqMClw5uXGTjSjwh4s5FUw6611sBjGomNRf/O7OSeo4xG
zHbZTEyLA37pbNk2wxykv2+5RoHgpAaxlTgy3dL9b07sevXz
r6WRe0TvsHkGgOO4s+QMGYJO8zXXJsq/27Ifvc7jwj7hCfQA
TR6w2T3B2o1lxqB7r+LXVh4Qh2ApjI6mtCrxnmyOGcB1DwID
/wOFMAMBAf8wCwYDVR0PBAQDAgG2MCMGA1UdJQQcMBoGCCsG
Reset to Factory Defaults
Cancel
OK

Allow ZAP to intercept “localhost” applications

- In ZAP interface, disable HUD by unchecking checkboxes in `Tools` > `Options` > `HUD` section.
- In Firefox browser, visit `` `about:config` `` and set `` `network.proxy.allow_hijacking_localhost` `` to `true`.



Vulnerability Scanning Using OWASP ZAP

- `docker run -v $(pwd):/zap/wrk/:rw -t ghcr.io/zaproxy/zaproxy:stable` **`zap-full-scan.py`** `-t https://192.168.1.6 -g gen.conf -r testreport.html`
- `docker run -v $(pwd):/zap/wrk/:rw -t ghcr.io/zaproxy/zaproxy:stable zap-full-scan.py -t https://192.168.1.6 -g gen.conf -r testreport.html -a -j -s -l INFO` **`--hook=hook1.py`**

hook1.py

```
# Change the zap_ajax_spider target to hit admin path

# Change the crawl_depth to 2

def zap_ajax_spider(zap, target, max_time):

    zap.ajaxSpider.set_option_max_crawl_depth(2)

    return zap, target + '/admin.html', max_time
```


BURPSUITE

Configure Burp Suite

- Refer https://github.com/riddhi-shree/web-app-pentesting-using-burp-suite

OpenVAS

Open Vulnerability Assessment Scanner

Configure OpenVAS

- Follow instructions given here: “web-app-security-nullcongoa2023/Day2/**openvas.md**”