iGNITE Technologies

WEB PENTEST LAB SETUP

DOCKER

www.hackingarticles.in

Contents

Introduction

In the world of cybersecurity, penetration testing and vulnerability assessment are crucial steps in identifying and mitigating potential security threats. With the increasing number of web applications, the need for secure and reliable testing environments has become more important than ever. In this article, we will explore the world of vulnerable web applications, their importance in penetration testing, and how to set them up using Docker.

Introduction to Vulnerable Web Applications

Vulnerable web applications are intentionally designed to be insecure, allowing penetration testers and security researchers to test and improve their skills in a safe and controlled environment. These applications are crucial in identifying and mitigating potential security threats and are widely used in the cybersecurity industry.

Prerequisite

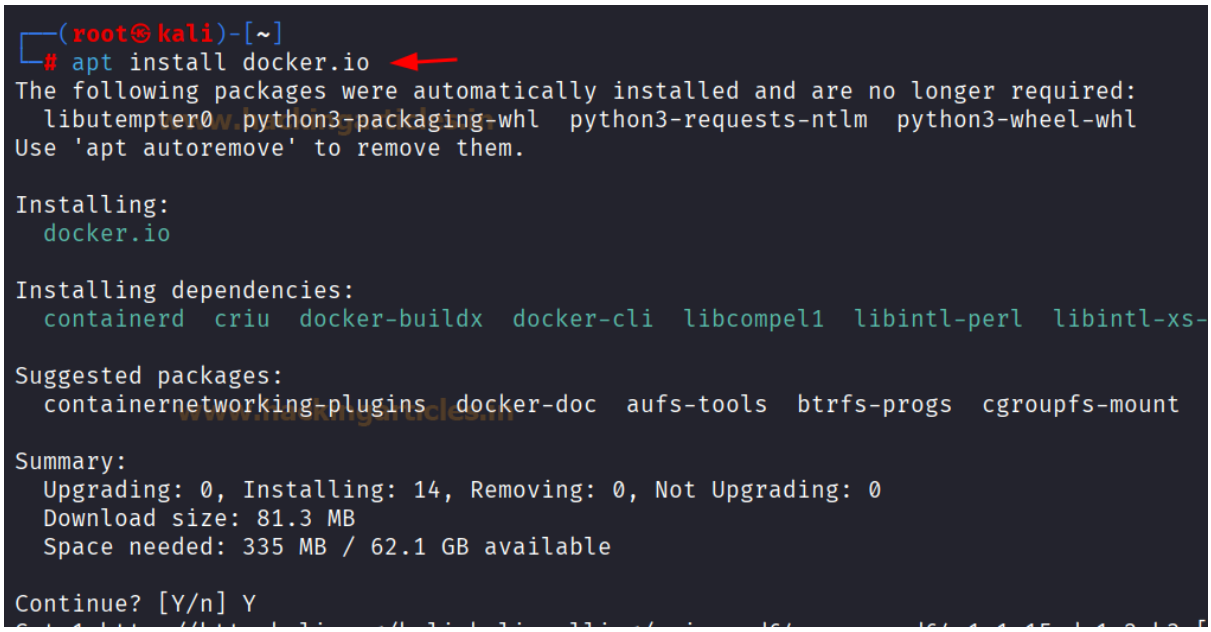
- Basic understanding of Docker and its commands
- Familiarity with Linux and web applications

Setting Up Docker on Your Machine

Docker is a platform that allows you to run applications in lightweight, isolated containers — perfect for deploying vulnerable apps without affecting your main system.

To install Docker and Docker Compose on a Debian/Ubuntu-based system, run:

```
apt install docker.io
```




```
apt install docker-compose
```

```
┌──(root㉿kali)-[~]
└─# apt install docker-compose
The following packages were automatically installed and are
  libutempter0  python3-packaging-whl  python3-requests-ntl
Use 'apt autoremove' to remove them.

Installing:
  docker-compose

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading:
  Download size: 13.0 MB
  Space needed: 64.4 MB / 61.8 GB available
```

Setting up Vulnerable Web Applications using Docker

Here are the instructions for setting up various vulnerable web applications using Docker:

DVWA (Damn Vulnerable Web Application)

DVWA is a PHP/MySQL web application that simulates a vulnerable website, allowing users to practice exploiting common web vulnerabilities such as SQL injection and cross-site scripting (XSS).

To configure DVWA on Docker, you can use the following commands:

```
docker pull sagikazarmark/dvwa
```

```
┌──(root㉿kali)-[~]
└─# docker pull sagikazarmark/dvwa
Using default tag: latest
latest: Pulling from sagikazarmark/dvwa
693502eb7dfb: Pull complete
e6c91bb380b4: Pull complete
e111b9773d58: Pull complete
55f12e04cfae: Pull complete
8f1b50e10184: Pull complete
Digest: sha256:1224167ccb59ad64751d52d7beb75fd445a252ae3c136
Status: Downloaded newer image for sagikazarmark/dvwa:latest
docker.io/sagikazarmark/dvwa:latest
```

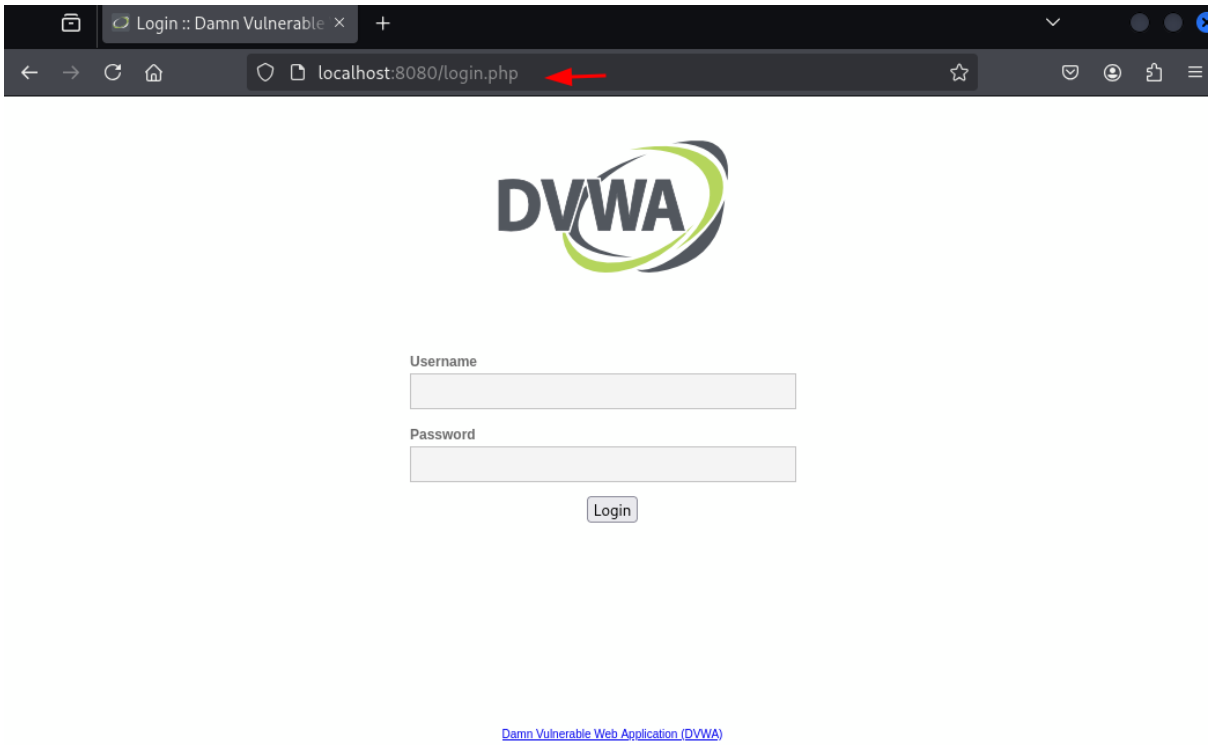
```
docker run --rm -it -p 8080:80 sagikazarmark/dvwa
```

```
┌──(root㉿kali)-[~]
└─# docker run --rm -it -p 8080:80 sagikazarmark/dvwa

[ ok ] Starting MySQL database server: mysqld ..
[info] Checking for tables which need an upgrade, are corrupt or were
not closed cleanly..
[....] Starting web server: apache2AH00558: apache2: Could not reliably dete
. ok
⟹ /var/log/apache2/access.log ⟸
tail: unrecognized file system type 0x794c7630 for '/var/log/apache2/access.

⟹ /var/log/apache2/error.log ⟸
[Thu Jun 05 11:51:46.779714 2025] [mpm_prefork:notice] [pid 545] AH00163: Ap
[Thu Jun 05 11:51:46.779763 2025] [core:notice] [pid 545] AH00094: Command
```

DVWA configuration & Login Steps

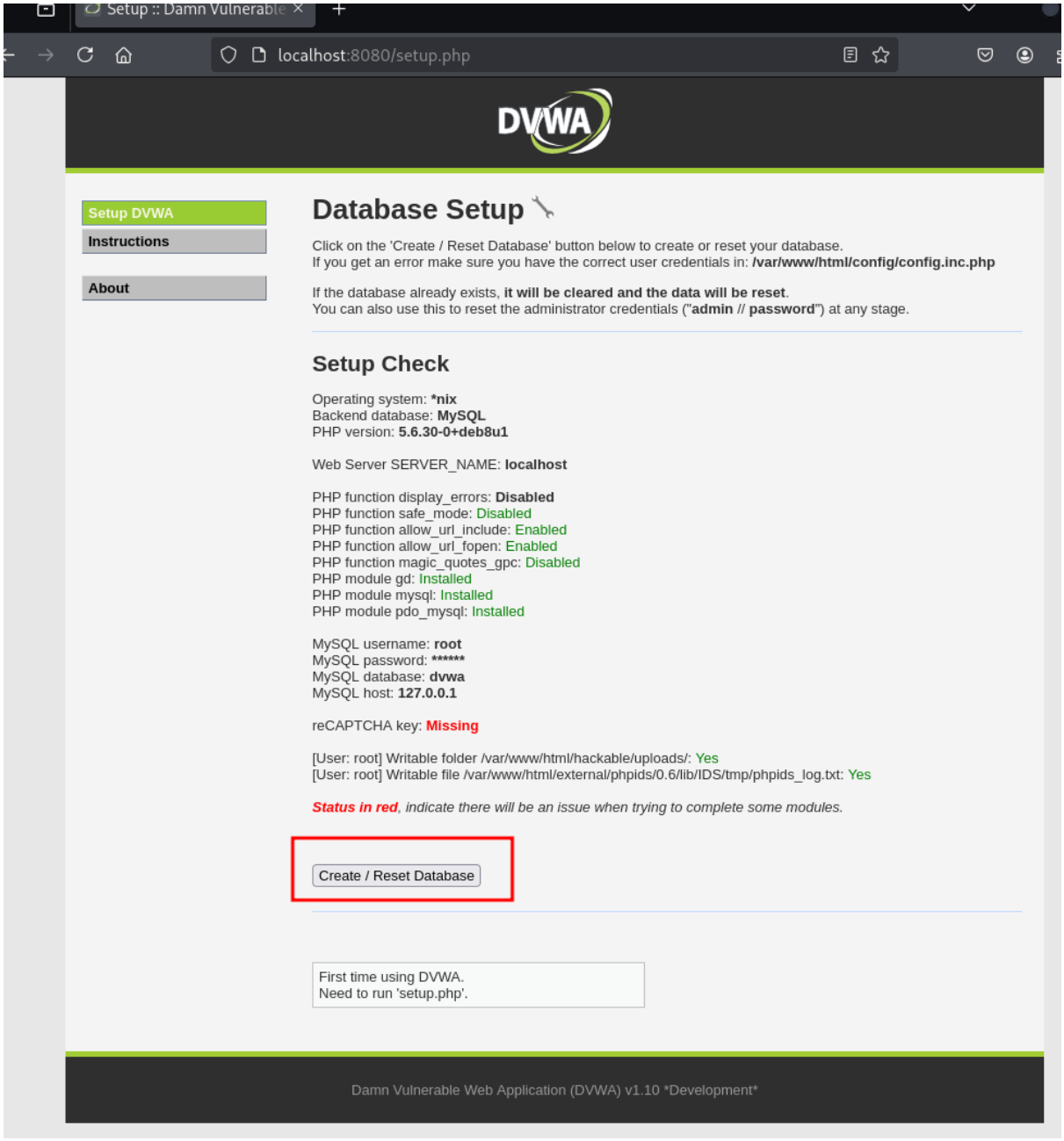
Access the Login Page

- Open DVWA in your browser (**http://localhost:8080/login.php**).
- Default Credentials:
  - Username: admin
  - Password: password
- Click Login.



Create/Reset the Database

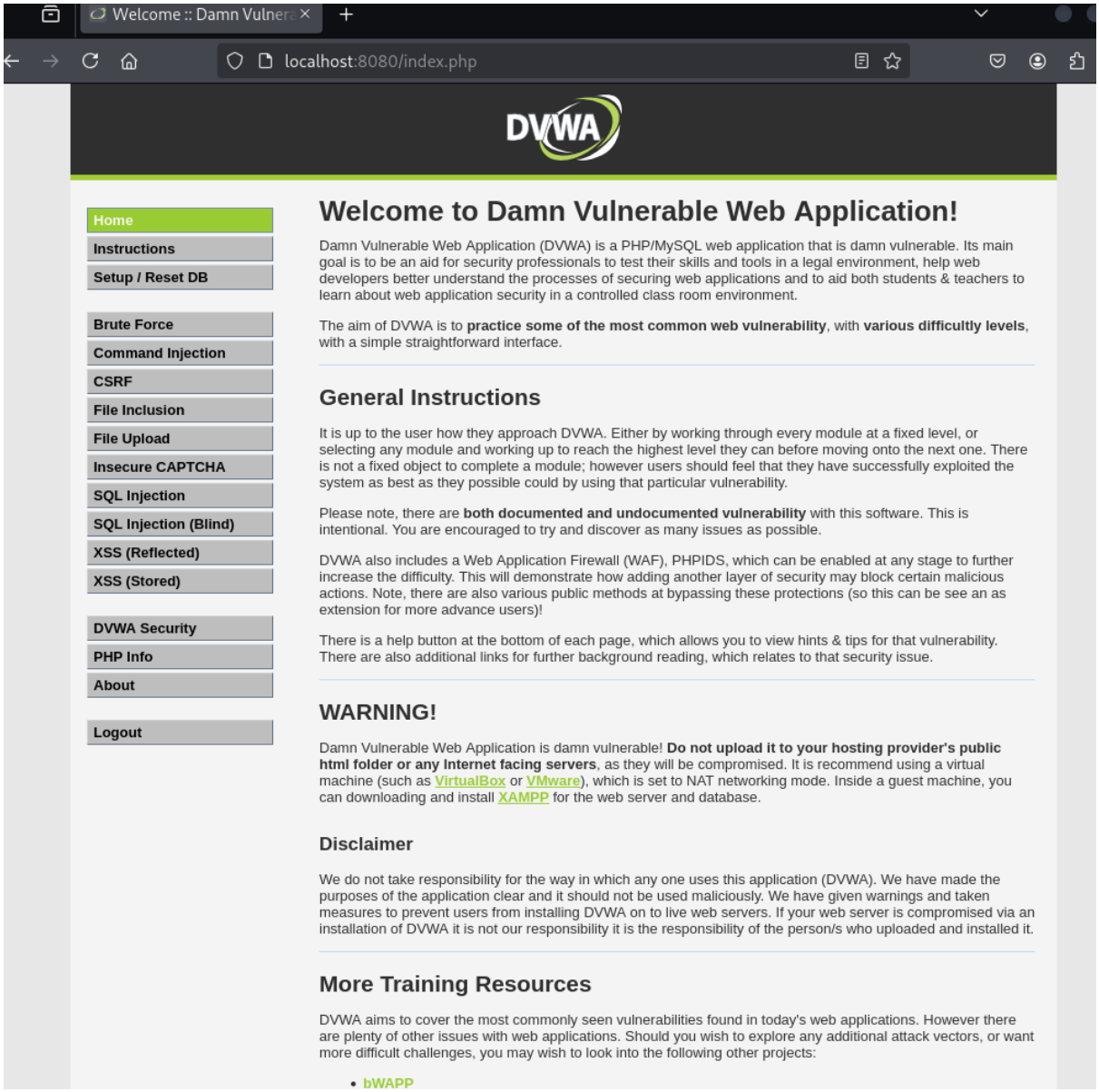
- After logging in, you'll see a "Database configuration" page.
- Click the "Create / Reset Database" button to:
  - Initialize the DVWA database (if first-time configuration).
  - Clear existing data and reset to default (if resetting).



***Note**: Ensure MySQL credentials in /var/www/html/config/config.inc.php are correct.*

Main Page Loads Automatically

- After database configuration, you'll be redirected to the DVWA Welcome Page.
- From here, you can:
    - Select security levels (Low/Medium/High/Impossible).
    - Access vulnerability modules (e.g., SQL Injection, XSS).



OWASP Juice Shop

OWASP Juice Shop is an open-source web application that is designed to be insecure, with over 30 vulnerabilities and challenges for users to discover and exploit.

To configure Juice Shop on Docker, you can use the following commands:

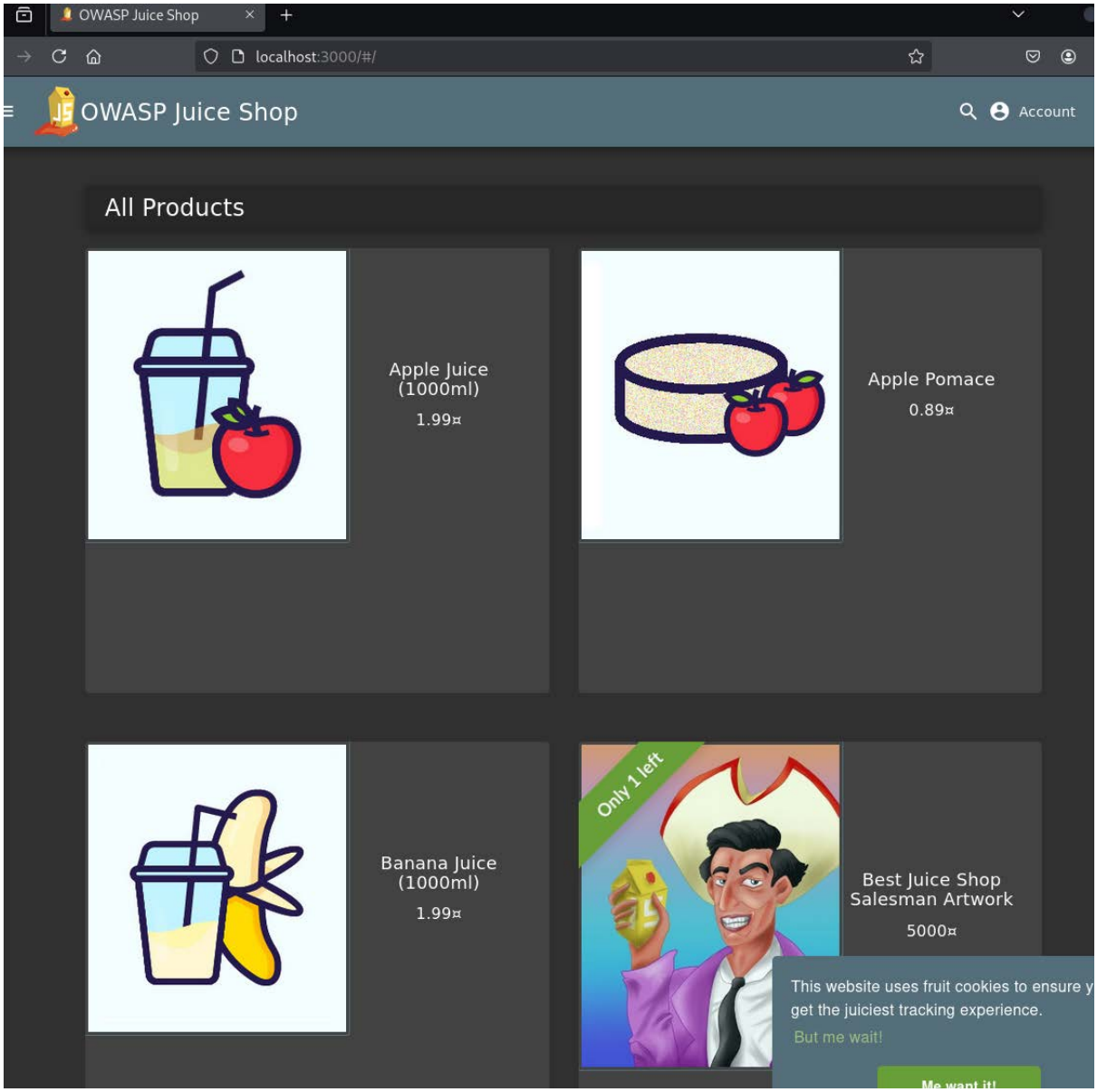
```
docker pull bkimminich/juice-shop
```

```
┌──(root㉿kali)-[~]
└─# docker pull bkimminich/juice-shop

Using default tag: latest
latest: Pulling from bkimminich/juice-shop
3d78e577de35: Pull complete
bfb59b82a9b6: Pull complete
4eff9a62d888: Pull complete
a62778643d56: Pull complete
7c12895b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
0bab15eea81d: Pull complete
4aa0ea1413d3: Pull complete
da7816fa955e: Pull complete
9aee425378d2: Pull complete
d00c3209d929: Pull complete
221438ca359c: Pull complete
ab0f6cad3051: Pull complete
6f971e93c4e2: Pull complete
c83c31ce41af: Pull complete
0cb5c07f8edd: Pull complete
3137de975d0a: Pull complete
```

```
docker run --rm -p 3000:3000 bkimminich/juice-shop
```

```
┌──(root㉿kali)-[~]
└─# docker run --rm -p 3000:3000 bkimminich/juice-shop

info: Detected Node.js version v20.19.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Entity models 19 of 19 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file styles.css is present (OK)
info: Required file index.html is present (OK)
info: Required file main.js is present (OK)
info: Required file tutorial.js is present (OK)
info: Required file runtime.js is present (OK)
```

Once you have started the Juice Shop container, you can access it by visiting **http://localhost:3000** via your web browser. From there, you can begin your hands on testing journey.



OWASP VulnApp

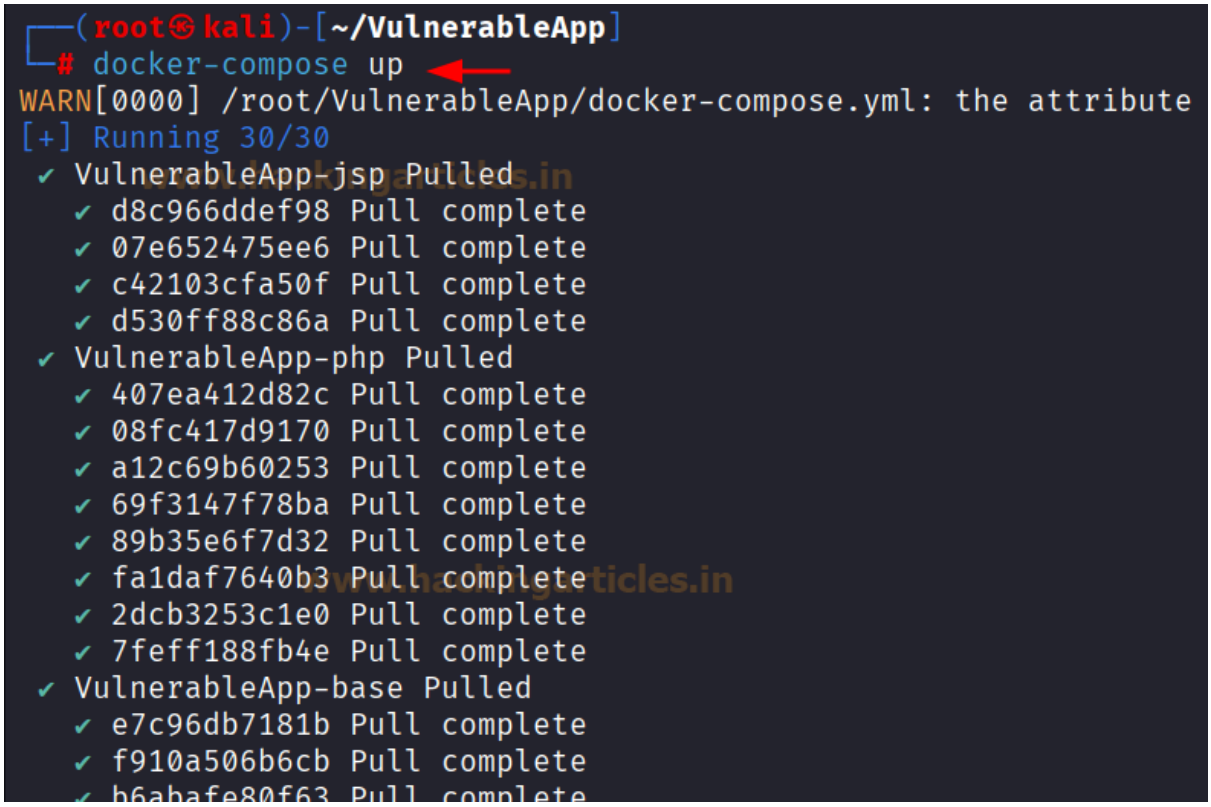
OWASP VulnApp is a web application that is intentionally insecure to various types of attacks, offering an environment for users to learn about web security and practice exploiting vulnerabilities in a safe and controlled environment. It covers a range of vulnerabilities, including SQL injection, XSS, and file inclusion.

To configure OWASP VulnApp on Docker, you can use the following commands:

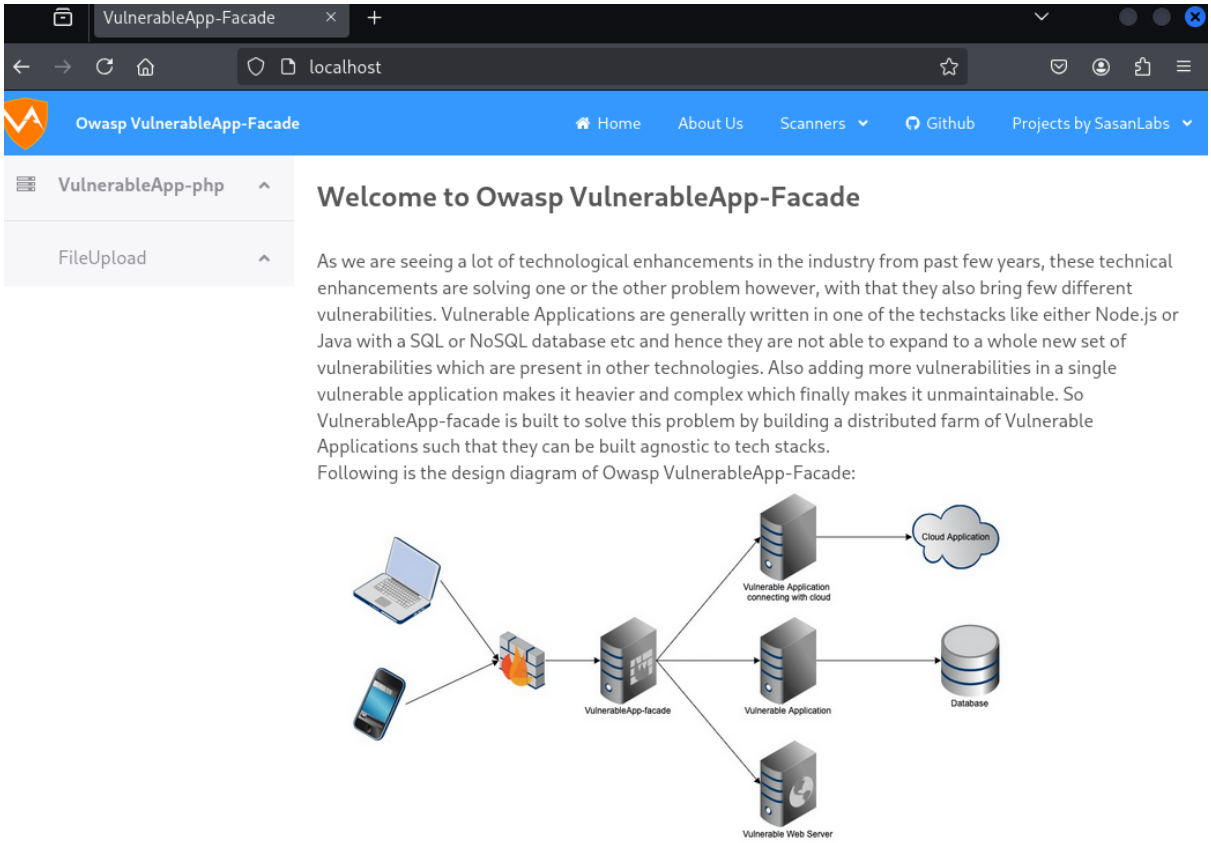
```
git clone https://github.com/SasanLabs/VulnerableApp.git
cd VulnerableApp
```

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/SasanLabs/VulnerableApp.git

Cloning into 'VulnerableApp' ...
remote: Enumerating objects: 7741, done.
remote: Total 7741 (delta 0), reused 0 (delta 0), pack-reused 7741 (from 1)
Receiving objects: 100% (7741/7741), 42.39 MiB | 11.57 MiB/s, done.
Resolving deltas: 100% (3919/3919), done.

┌──(root㉿kali)-[~]
└─# cd VulnerableApp

┌──(root㉿kali)-[~/VulnerableApp]
└─# ls -al
total 108
drwxr-xr-x  9 root root  4096 Jun  5 08:00 .
drwx------ 24 root root  4096 Jun  5 08:00 ..
-rw-r--r--  1 root root  5381 Jun  5 08:00 build.gradle
-rw-r--r--  1 root root   815 Jun  5 08:00 .classpath
-rw-r--r--  1 root root  3353 Jun  5 08:00 CODE_OF_CONDUCT.md
-rw-r--r--  1 root root  2196 Jun  5 08:00 CONTRIBUTING.md
-rw-r--r--  1 root root   610 Jun  5 08:00 docker-compose.yml
drwxr-xr-x  6 root root  4096 Jun  5 08:00 docs
drwxr-xr-x  8 root root  4096 Jun  5 08:00 .git
drwxr-xr-x  4 root root  4096 Jun  5 08:00 .github
-rw-r--r--  1 root root   334 Jun  5 08:00 .gitignore
drwxr-xr-x  3 root root  4096 Jun  5 08:00 gradle
-rwxr-xr-x  1 root root  5766 Jun  5 08:00 gradlew
-rw-r--r--  1 root root  2763 Jun  5 08:00 gradlew.bat
-rw-r--r--  1 root root 11357 Jun  5 08:00 LICENSE
-rw-r--r--  1 root root   608 Jun  5 08:00 .project
-rw-r--r--  1 root root  9831 Jun  5 08:00 README.md
drwxr-xr-x  3 root root  4096 Jun  5 08:00 scanner
drwxr-xr-x  2 root root  4096 Jun  5 08:00 .settings
drwxr-xr-x  4 root root  4096 Jun  5 08:00 src
-rw-r--r--  1 root root    44 Jun  5 08:00 .travis.yml
```

```
docker-compose up
```



Once you have started the OWASP VulnApp container, you can access it by visiting **http://localhost:80** via your web browser. From there, you can begin your hands on testing journey.



OWASP WebGoat

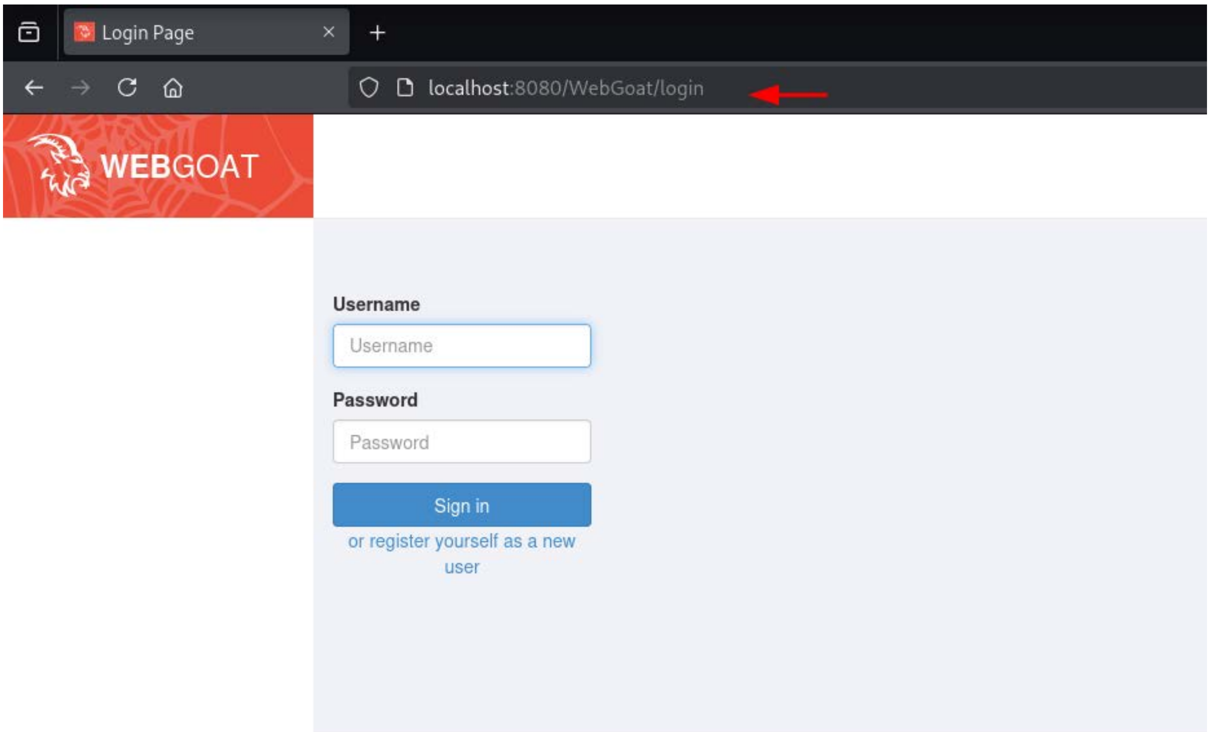
Web Goat is a web application that is designed to teach web application security through a series of lessons and exercises, covering topics such as authentication, authorization, and input validation.

To configure WebGoat on Docker, you can use the following commands:

```
docker pull webgoat/webgoat
docker run -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 -e TZ=Europe/Amsterdam
webgoat/webgoat
```

```
┌──(root㉿kali)-[~]
└─# docker pull webgoat/webgoat

Using default tag: latest
latest: Pulling from webgoat/webgoat
5a7813e071bf: Pull complete
d1504bee8985: Pull complete
7b5a3507f742: Pull complete
4f4fb700ef54: Pull complete
ea28f6f7f0aa: Pull complete
7b2e4df376c9: Pull complete
307ac9d4e999: Pull complete
Digest: sha256:3101bd9e7bcfe122d7ef91e690ef3720de36cc4e86b3d06763a1ddf2e2751a4b
Status: Downloaded newer image for webgoat/webgoat:latest
docker.io/webgoat/webgoat:latest

┌──(root㉿kali)-[~]
└─# docker run -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 -e TZ=Europe/Amsterdam webgoat/webgoat

2025-06-05T14:23:41.936+02:00  INFO 1 --- [           main] org.owasp.webgoat.server.StartWebGoat    : Sta
2025-06-05T14:23:41.940+02:00  INFO 1 --- [           main] org.owasp.webgoat.server.StartWebGoat    : No
2025-06-05T14:23:42.662+02:00  INFO 1 --- [           main] org.owasp.webgoat.server.StartWebGoat    : Sta

2025-06-05T14:23:42.744+02:00  INFO 1 --- [           main] org.owasp.webgoat.server.StartWebGoat    : No
2025-06-05T14:23:43.251+02:00  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Boo
2025-06-05T14:23:43.291+02:00  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Fin
2025-06-05T14:23:44.176+02:00  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tom
2025-06-05T14:23:44.186+02:00  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Sta
```

Once you have started the WebGoat container, you can access it by visiting **http://localhost:8080/WebGoat/login** via your web browser. From there, you can begin your hands on testing journey.

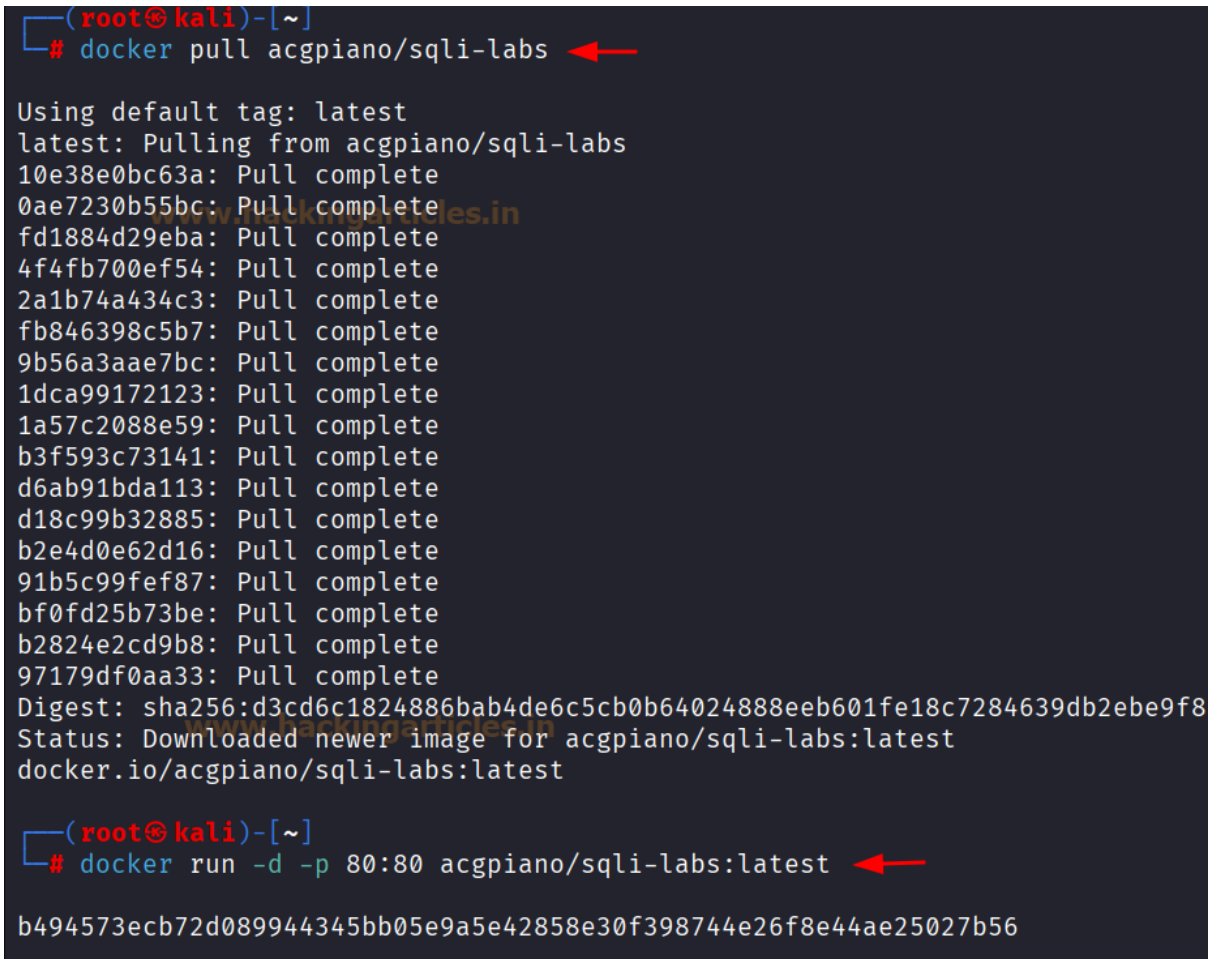


SQLi-Labs

SQLi-Labs is a web application that is specifically designed to teach SQL injection attacks, providing a range of challenges and exercises for users to practice exploiting SQL injection vulnerabilities.

To configure SQLi-Labs on Docker, you can use the following commands:

```
docker pull acgpiano/sqli-labs
docker run -d -p 80:80 acgpiano/sqli-labs:latest
```



Once you have started the SQLi-Labs container, you can access it by visiting **http://localhost** via your web browser. From there, you can begin your hands-on testing journey.

SQLi-LABS configuration & Challenges

Database configuration

- Click "configuration/reset Database for labs" to:
  - Purge old database (SECURITY) if it exists.

SQLi-LABS Page-1(*Basic Challenges*)

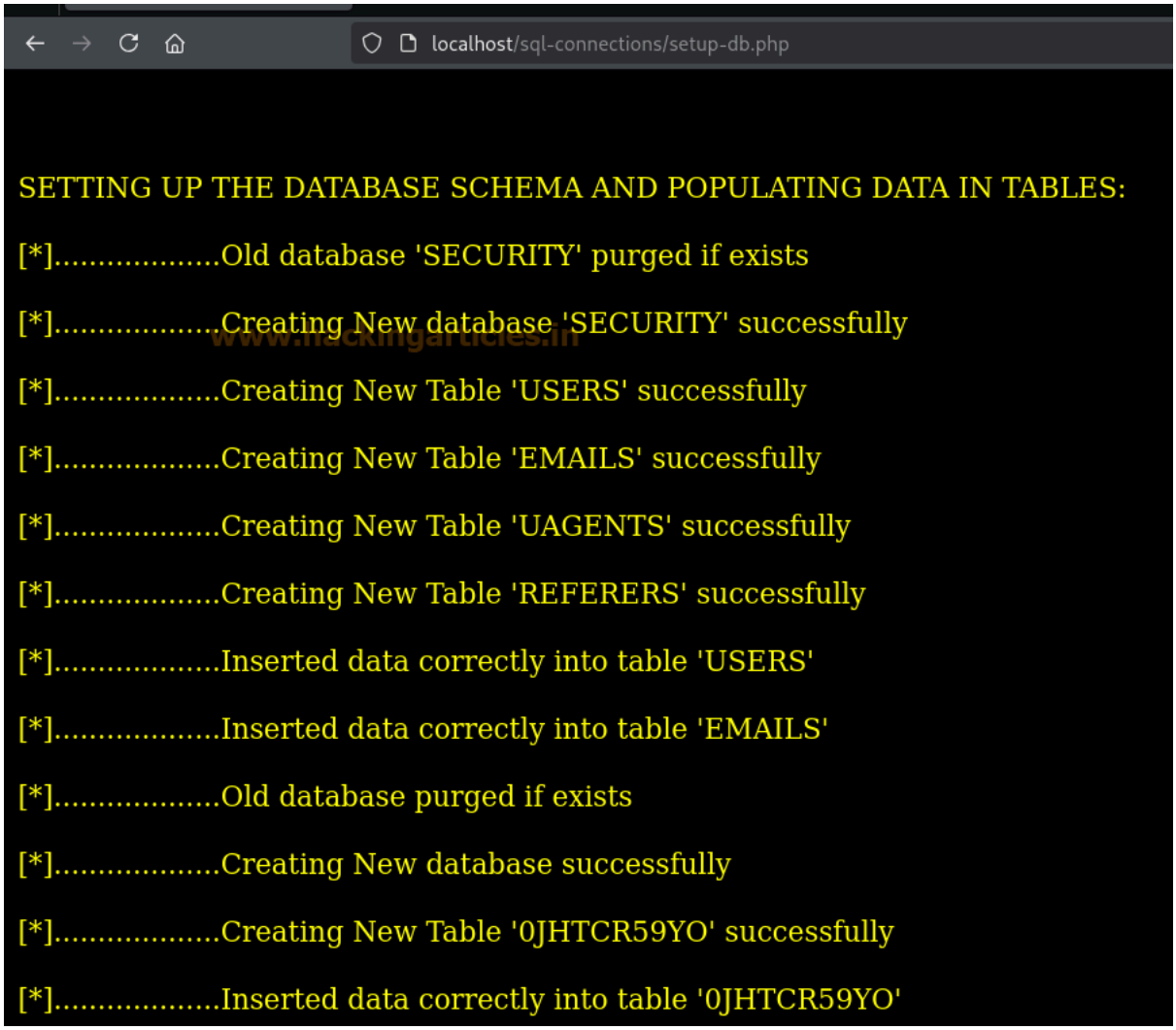
Setup/reset Database for labs

Page-2 (Advanced Injections)

Page-3 (Stacked Injections)

Page-4 (Challenges)

- Create new tables (USERS, EMAILS, etc.) and populate them with test data.



- Output confirms successful creation of databases/tables (e.g., [*].....Creating New Table 'USERS' successfully).

Challenge Categories

- Page-1 (Basic): Error-based, blind, and time-based SQL injections (e.g., GET - Error based - Single quotes).
- Page-2 (Advanced): Complex injections like double injections, outfile dumping.
- Page-3 (Stacked): Multi-query injections (e.g., ST - Double Injection).
- Page-4 (Challenges): Mixed difficulty levels (Less-1 to Less-22).

bWAPP

BWAPP is a web application that is intentionally insecure to various types of attacks, offering an environment for users to learn about web security and practice exploiting vulnerabilities in a safe and controlled environment. It covers a range of vulnerabilities, including SQL injection, XSS, and file inclusion.

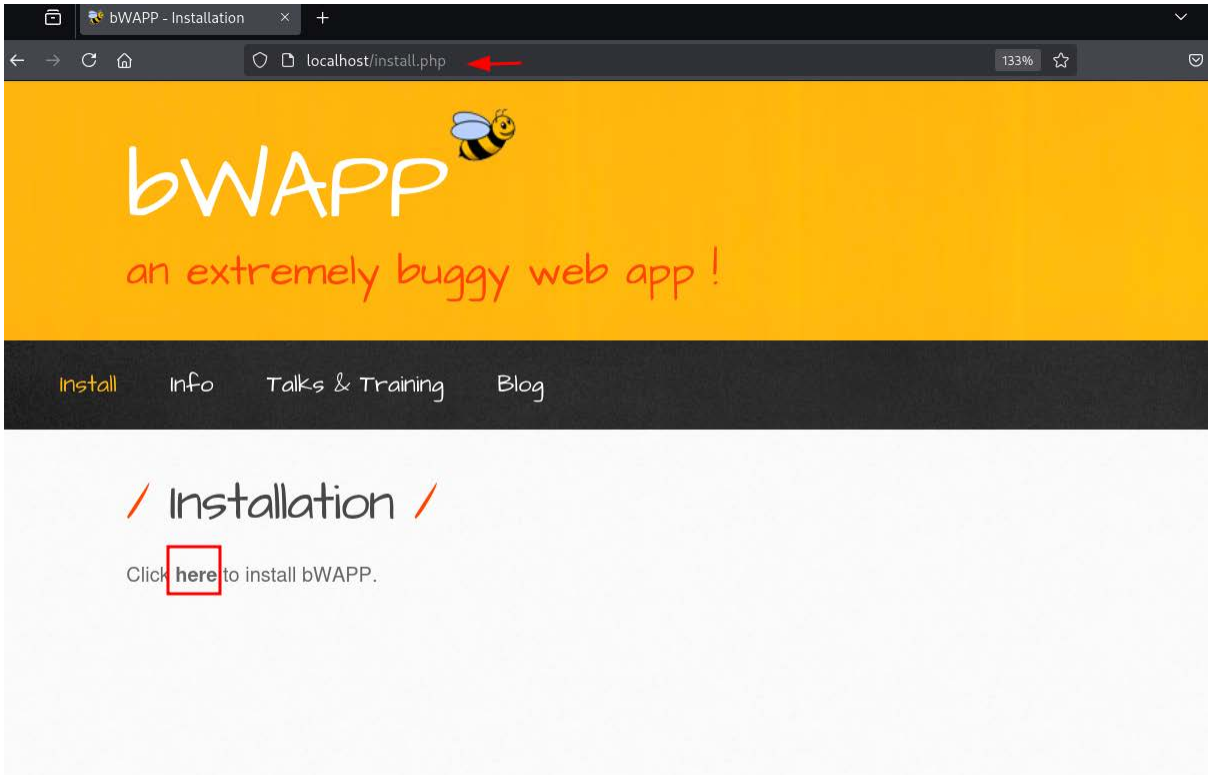
To configure bWAPP on Docker, you can use the following commands:

```
docker pull raesene/bwapp
docker run -d -p 80:80 raesene/bwapp
```

```
┌──(root㉿kali)-[~]
└─# docker pull raesene/bwapp

Using default tag: latest
latest: Pulling from raesene/bwapp
Digest: sha256:2f41183ea9f9e8fb36678d7a2a0c8a9db9a59f4569cee02fe6664b419b2600ee
Status: Image is up to date for raesene/bwapp:latest
docker.io/raesene/bwapp:latest

┌──(root㉿kali)-[~]
└─# docker run -d -p 80:80 raesene/bwapp

3eebeae39756914a5b8f16311d1c8b094dde9a24e7bc7e5bf4012f5bfedb4624
```

Once you have started the bWAPP container, you can access it by visiting **http://localhost/install.php** via your web browser.

Click “here” to install bWAPP.



After Installation, the main page will open.

OWASP Mutillidae II

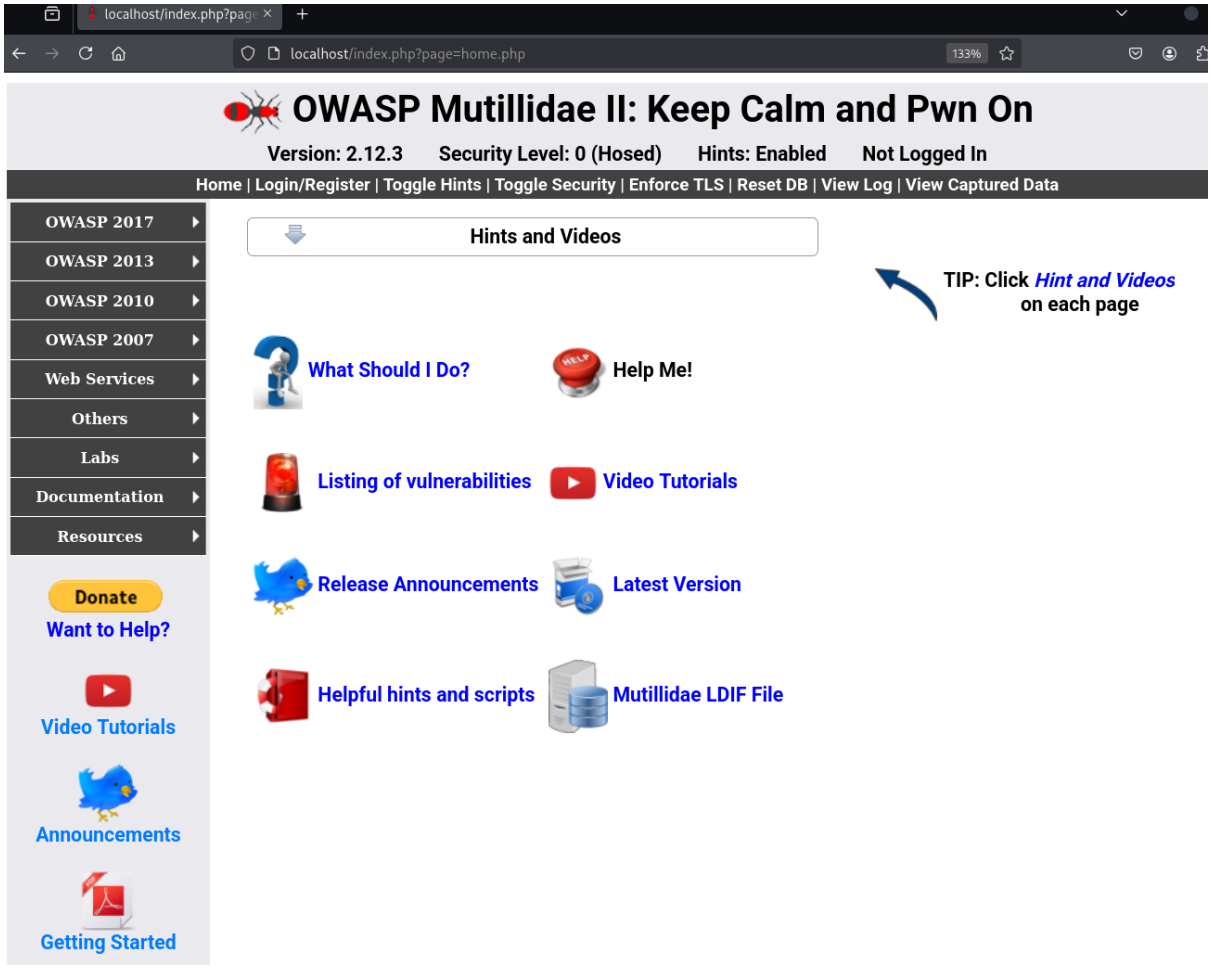
Mutillidae is a web application that is intentionally insecure to various types of attacks, offering an environment for users to learn about web security and practice exploiting vulnerabilities in a safe and controlled environment. It covers a range of vulnerabilities, including SQL injection, XSS, and file inclusion, and is known for its realistic and challenging scenarios.

To configure Mutillidae on Docker, you can use the following commands:

```
git clone https://github.com/webpwnized/mutillidae-docker.git
cd mutillidae-docker
docker compose -f.build/docker-compose.yml up --build –detach
```

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/webpwnized/mutillidae-docker.git
Cloning into 'mutillidae-docker' ...
remote: Enumerating objects: 1169, done.
remote: Counting objects: 100% (61/61), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 1169 (delta 55), reused 49 (delta 49), pack-reused 1108 (from 2)
Receiving objects: 100% (1169/1169), 178.38 KiB | 1.00 MiB/s, done.
Resolving deltas: 100% (505/505), done.

┌──(root㉿kali)-[~]
└─# cd mutillidae-docker

┌──(root㉿kali)-[~/mutillidae-docker]
└─# docker compose -f .build/docker-compose.yml up --build --detach

[+] Building 6.9s (6/9)
 ⇒ [database internal] load build definition from Dockerfile
 ⇒ ⇒ transferring dockerfile: 2.08kB
 ⇒ [database internal] load metadata for docker.io/library/mariadb:latest
 ⇒ [directory internal] load build definition from Dockerfile
 ⇒ ⇒ transferring dockerfile: 2.50kB
 ⇒ [directory internal] load metadata for docker.io/osixia/openldap:latest
 ⇒ [database internal] load .dockerignore
 ⇒ ⇒ transferring context: 2B
 ⇒ [database 1/2] FROM docker.io/library/mariadb:latest@sha256:fcc7fcd7114adb5d41f
 ⇒ ⇒ resolve docker.io/library/mariadb:latest@sha256:fcc7fcd7114adb5d41f14d116b8a
```

Once you have started the Mutillidae container, you can access it by visiting **http://localhost:80** via your web browser. From there, you can begin your hands on testing journey.



Damn Vulnerable GraphQL Application

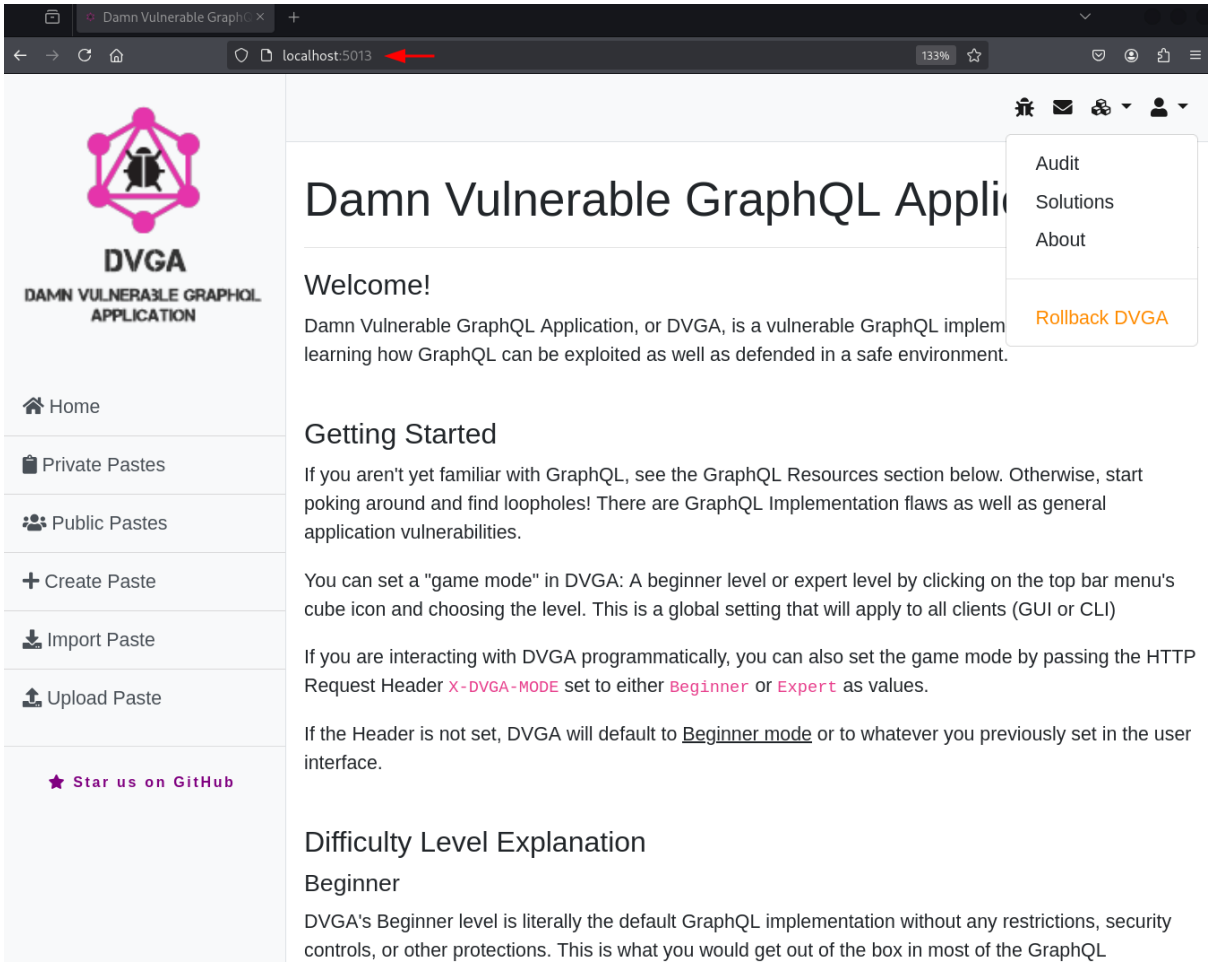
Damn Vulnerable GraphQL Application is a GraphQL-based web application that is intentionally insecure to various types of attacks, offering an environment for users to learn about GraphQL security and practice exploiting vulnerabilities in a safe and controlled environment.

To configure Damn Vulnerable GraphQL Application on Docker, you can use the following commands:

```
docker pull dolevf/dvga
docker run -t -p 5013:5013 -e WEB_HOST=0.0.0.0 dolevf/dvga
```

```
┌──(root㉿kali)-[~]
└─# docker pull dolevf/dvga

Using default tag: latest
latest: Pulling from dolevf/dvga
Digest: sha256:040aa33c199d99f3380c9ff9a1ee5d725e9abca7b189c63a35a2a73bda79c957
Status: Image is up to date for dolevf/dvga:latest
docker.io/dolevf/dvga:latest

┌──(root㉿kali)-[~]
└─# docker run -t -p 5013:5013 -e WEB_HOST=0.0.0.0 dolevf/dvga

DVGA Server Version: 2.1.2 Running ...
```

Once you have started the Damn Vulnerable GraphQL Application container, you can access it by visiting **http://localhost:5013** via your web browser. From there, you can begin your hands on testing journey.



Damn Vulnerable RESTurant API

Damn Vulnerable Resturant API is a RESTful API that is intentionally insecure to various types of attacks, offering an environment for users to learn about API security and practice exploiting vulnerabilities in a safe and controlled environment. It covers a range of vulnerabilities specific to APIs, such as authentication and authorisation flaws.

To configure Damn Vulnerable RESTurant API on Docker, you can use the following commands:

```
git clone https://github.com/theowni/Damn-Vulnerable-RESTaurant-API-Game.git
cd Damn-Vulnerable-RESTaurant-API-Game
./start_app.sh
```

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/theowni/Damn-Vulnerable-RESTaurant-API-Game.git

Cloning into 'Damn-Vulnerable-RESTaurant-API-Game' ...
remote: Enumerating objects: 645, done.
remote: Counting objects: 100% (168/168), done.
remote: Compressing objects: 100% (68/68), done.
remote: Total 645 (delta 122), reused 100 (delta 100), pack-reused 477 (from 1)
Receiving objects: 100% (645/645), 849.92 KiB | 2.79 MiB/s, done.
Resolving deltas: 100% (345/345), done.

┌──(root㉿kali)-[~]
└─# cd Damn-Vulnerable-RESTaurant-API-Game

┌──(root㉿kali)-[~/Damn-Vulnerable-RESTaurant-API-Game]
└─# ./start_app.sh

[+] Running 8/9
 ⠸ db [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿] 89.22MB / 93.78MB Pulling
   ✔ 96526aa774ef Pull complete
   ✔ 86da063037d7 Pull complete
   ✔ a93c97be5f7f Pull complete
   ✔ 9737661d941c Download complete
```

Once you have started the Damn Vulnerable RESTurant API container, you can access it by visiting **http://localhost:8091/docs** via your web browser. From there, you can begin your hands on testing journey.

Damn Vulnerable RESTaurant 1.0.0 OAS 3.1

/openapi.json

An intentionally vulnerable API service designed for learning and training purposes for ethical hackers, security engineers, and developers.

Servers

http://localhost:8091 - Local API server

Authorize

healthcheck

GET /healthcheck Healthcheck

menu

GET /menu Get Menu

PUT /menu Create Menu Item

DELETE /menu/{item_id} Delete Menu Item

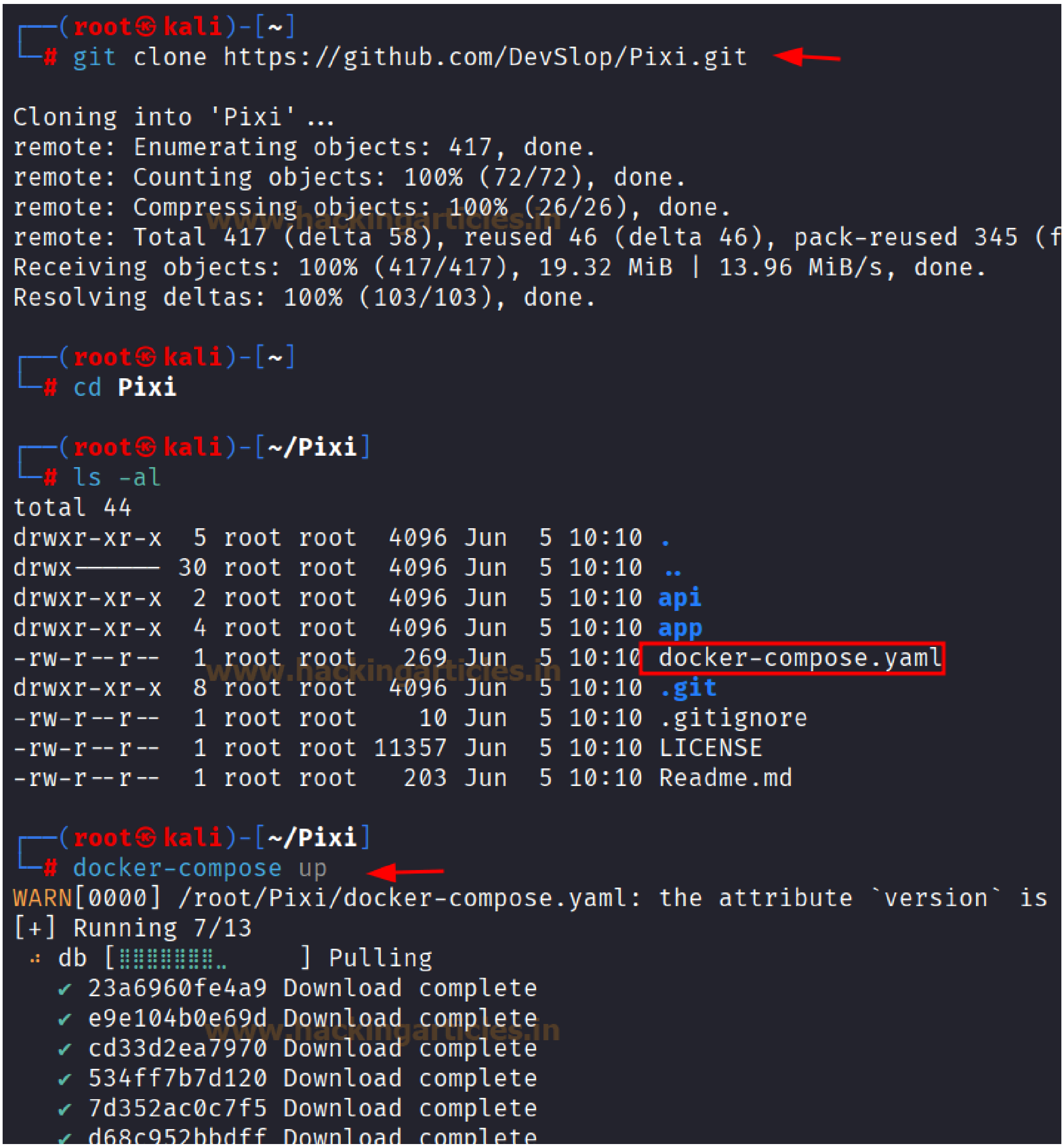
PUT /menu/{item_id} Update Menu Item

Pixi

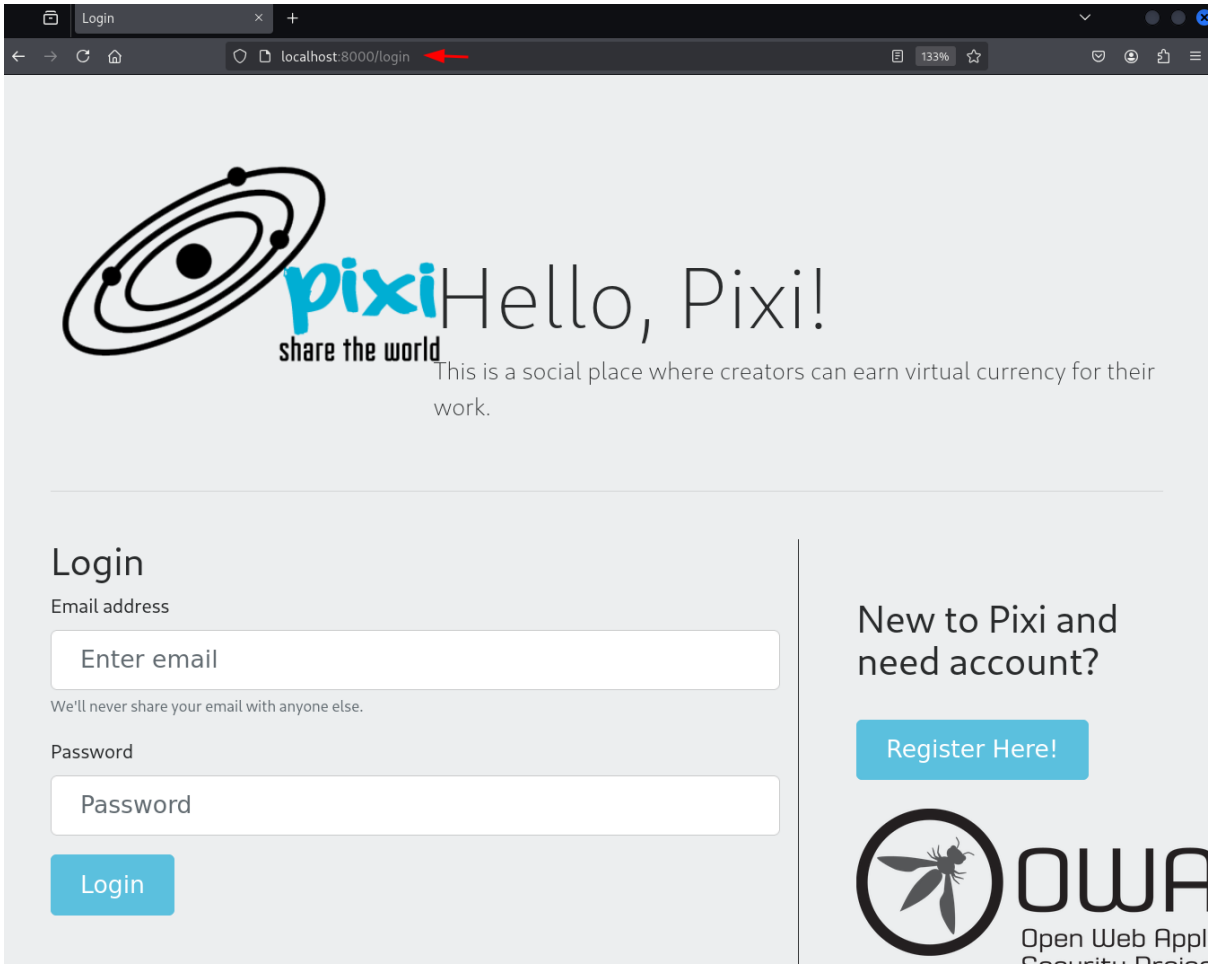
Pixi is a web application that is intentionally insecure to various types of attacks, offering an environment for users to learn about web security and practice exploiting vulnerabilities in a safe and controlled environment.

To configure Pixi on Docker, you can use the following commands:

```
git clone https://github.com/DevSlop/Pixi.git
cd Pixi
docker-compose up
```



Once you have started the Pixi container, you can access it by visiting **http://localhost:8000/login** via your web browser. From there, you can begin your hands on testing journey.



PyGoat

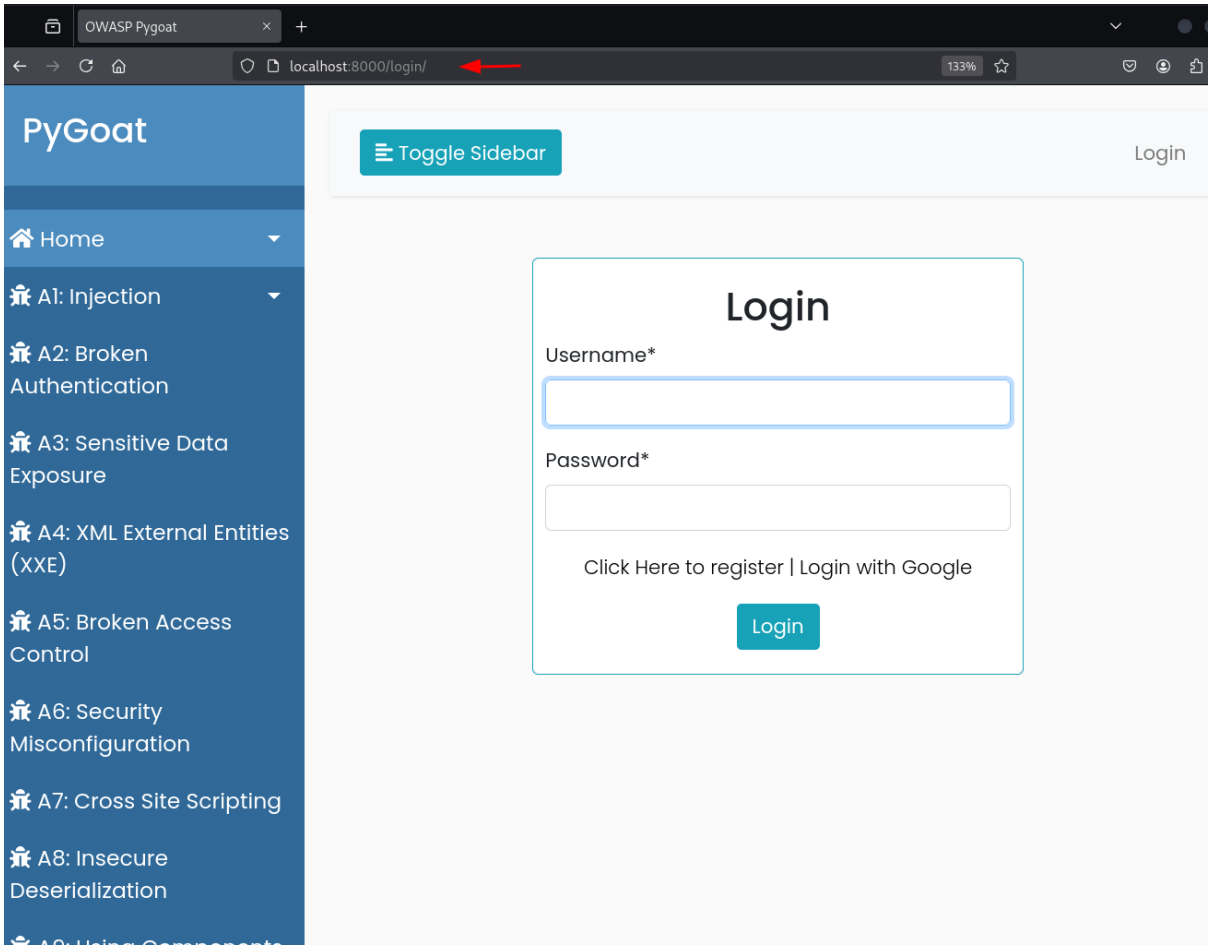
PyGoat is a web application that is designed to teach web application security through a series of lessons and exercises, covering topics such as authentication, authorisation, and input validation. It provides a hands-on environment for users to learn about web security and practice exploiting vulnerabilities, with a focus on Python-based web applications.

To configure PyGoat on Docker, you can use the following commands:

```
docker pull pygoat/pygoat:latest
docker run --rm -p 8000:8000 pygoat/pygoat:latest
```

```
┌──(root㉿kali)-[~]
└─# docker pull pygoat/pygoat:latest

latest: Pulling from pygoat/pygoat
6aefca2dc61d: Pull complete
967757d56527: Pull complete
c357e2c68cb3: Pull complete
c766e27afb21: Pull complete
32a180f5cf85: Pull complete
1535e3c1181a: Pull complete
ca398dbb0a27: Pull complete
fc3fb1727276: Pull complete
13ca01dc6e0b: Pull complete
85487682e9bf: Pull complete
89f3fa55b841: Pull complete
8b15550bfa89: Pull complete
809c6a69c05b: Pull complete
32171777ff27: Pull complete
351b5d9c4e1a: Pull complete
dd84b3b61f2f: Pull complete
1b5ef8302d20: Pull complete
a3dde43128dc: Pull complete
Digest: sha256:26a75f7cd023a9c77158f55c54dbb405ab2b77aaf46dfb888f209735fa0d0b54
Status: Downloaded newer image for pygoat/pygoat:latest
docker.io/pygoat/pygoat:latest

┌──(root㉿kali)-[~]
└─# docker run --rm -p 8000:8000 pygoat/pygoat:latest
www.hackingarticles.in
Watching for file changes with StatReloader
Performing system checks ...

System check identified no issues (0 silenced).
June 05, 2025 - 14:17:35
Django version 4.0.4, using settings 'pygoat.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

Once you have started the PyGoat container, you can access it by visiting **http://localhost:8000/login** via your web browser. From there, you can begin your hands-on testing journey.



Server-Side Request Forgery

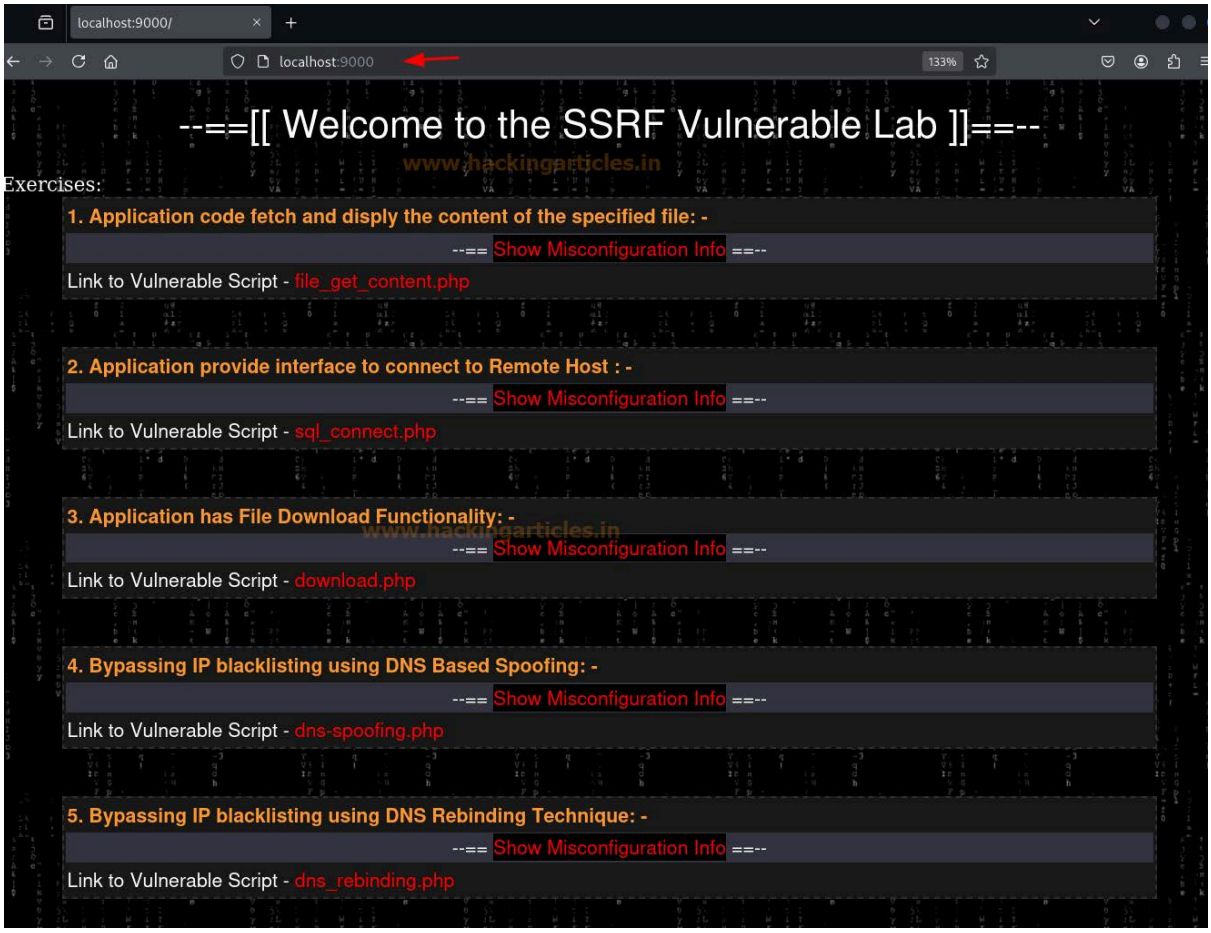
SSRF is a web application that is designed to teach Server-Side Request Forgery (SSRF) attacks, providing a range of challenges and exercises for users to practice exploiting SSRF vulnerabilities.

To configure SSRF on Docker, you can use the following commands:

```
docker pull youyouorz/ssrf-vulnerable-lab
docker run -d --name ssrf-lab -p 9000:80 youyouorz/ssrf-vulnerable-lab
```

```
┌──(root㉿kali)-[~]
└─# docker pull youyouorz/ssrf-vulnerable-lab

Using default tag: latest
latest: Pulling from youyouorz/ssrf-vulnerable-lab
5e6ec7f28fb7: Pull complete
cf165947b5b7: Pull complete
7bd37682846d: Pull complete
99daf8e838e1: Pull complete
ae320713efba: Pull complete
ebcb99c48d8c: Pull complete
9867e71b4ab6: Pull complete
936eb418164a: Pull complete
bc298e7adaf7: Pull complete
ccd61b587bcd: Pull complete
b2d4b347f67c: Pull complete
56e9dde34152: Pull complete
9ad99b17eb78: Pull complete
87d97eadcdf9: Pull complete
a5d9c413828a: Pull complete
Digest: sha256:655ed05a5e690dcff30eb2f81299d8b22847cb516bd597a13d2bd0b2310efdef
Status: Downloaded newer image for youyouorz/ssrf-vulnerable-lab:latest
docker.io/youyouorz/ssrf-vulnerable-lab:latest

┌──(root㉿kali)-[~]
└─# docker run -d --name ssrf-lab -p 9000:80 youyouorz/ssrf-vulnerable-lab

dcbf369f7bc8342a533f8670930662ceb055a1622618a67cf36a01ac89d9b353
```

Once you have started the SSRF container, you can access it by visiting **http://localhost:9000** via your web browser. From there, you can begin your hands-on testing journey.



VulnBank

VulnBank is a web application that is designed to simulate a vulnerable online banking system, allowing users to practice exploiting common web vulnerabilities such as SQL injection and cross-site scripting (XSS).

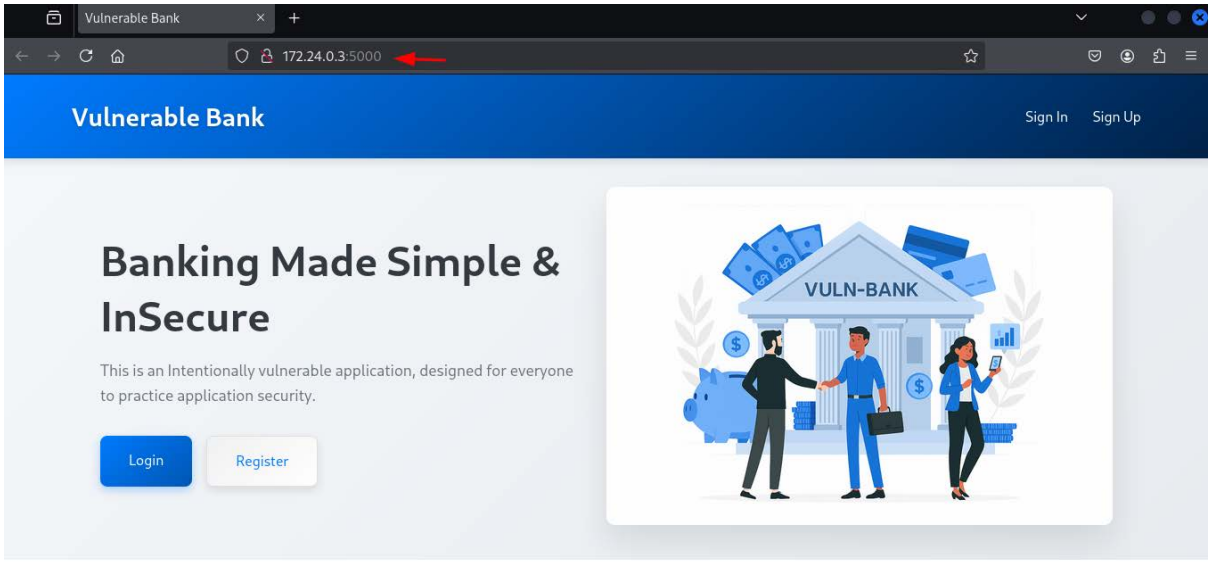
To configure VulnBank on Docker, you can use the following commands:

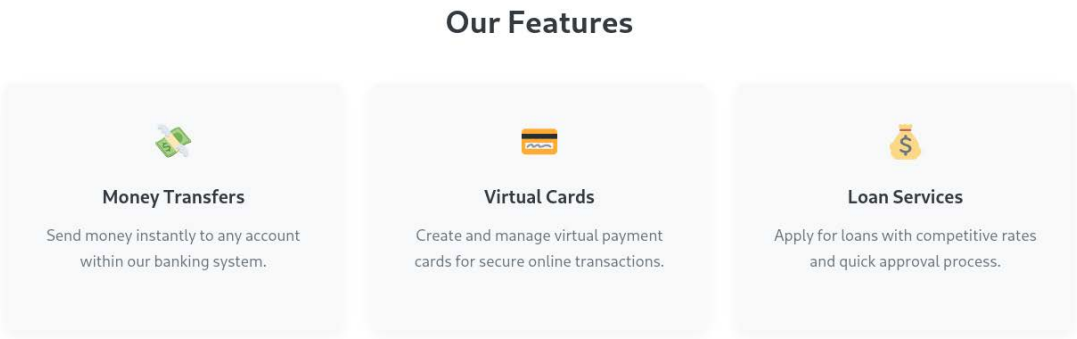
```
git clone https://github.com/Commando-X/vuln-bank.git
cd vuln-bank
docker-compose up
```

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/Commando-X/vuln-bank.git

Cloning into 'vuln-bank' ...
remote: Enumerating objects: 175, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 175 (delta 26), reused 15 (delta 15), pack-reused 139 (from 1)
Receiving objects: 100% (175/175), 1.63 MiB | 2.80 MiB/s, done.
Resolving deltas: 100% (94/94), done.

┌──(root㉿kali)-[~]
└─# cd vuln-bank

┌──(root㉿kali)-[~/vuln-bank]
└─# docker-compose up
WARN[0000] /root/vuln-bank/docker-compose.yml: the attribute `version` is obsol
[+] Running 14/15
 ⠧ db [##################                ] 117.8MB / 122.8MB Pulling
   ✔ 61320b01ae5e Already exists
   ✔ 3db9b37be7c3 Pull complete
   ✔ b094358e9765 Pull complete
   ✔ 6bae5fff5835 Pull complete
   ✔ ce5633da08fe Pull complete
   ✔ 72f83ee82e85 Pull complete
   ✔ bd1fa28722bb Pull complete
   ✔ 410cd7ec9a40 Pull complete
   ✔ f503177ee6ac Download complete
   ✔ 4485e9416430 Download complete
   ✔ 89ba8b615fa9 Download complete
   ✔ 5dd960f6dbc6 Download complete
   ✔ 82726bd8bf4c Download complete
   ✔ 2bb588ce4e67 Download complete
```

```
web-1  |    Use a production WSGI server instead.
web-1  |  * Debug mode: on
web-1  |  * Running on all addresses.
web-1  |    WARNING: This is a development server. Do not use it in a production deployment.
web-1  |  * Running on http://172.24.0.3:5000/ (Press CTRL+C to quit)
web-1  |  * Restarting with stat
web-1  |  * Debugger is active!
web-1  |  * Debugger PIN: 123-556-094
web-1  | 172.24.0.1 - - [05/Jun/2025 14:22:03] "GET / HTTP/1.1" 200 -
web-1  | 172.24.0.1 - - [05/Jun/2025 14:22:03] "GET /static/style.css HTTP/1.1" 200 -
web-1  | 172.24.0.1 - - [05/Jun/2025 14:22:03] "GET /static/uploads/banking-app.png HTTP/1.1" 200 -
web-1  | 172.24.0.1 - - [05/Jun/2025 14:22:03] "GET /favicon.ico HTTP/1.1" 404 -
```

Once you have started the VulnBank container, you can access it by visiting **http://172.24.0.3:5000** via your web browser. From there, you can begin your hands-on testing journey.



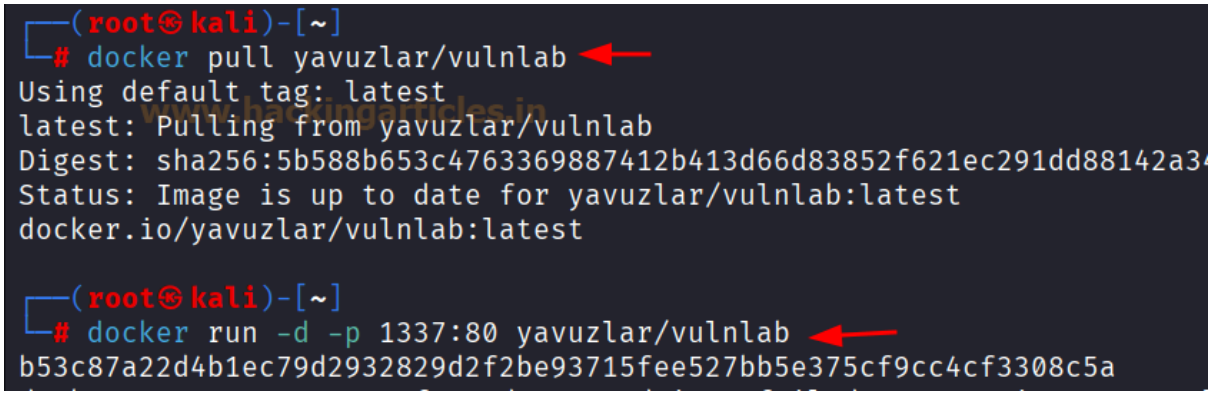


VulnLab

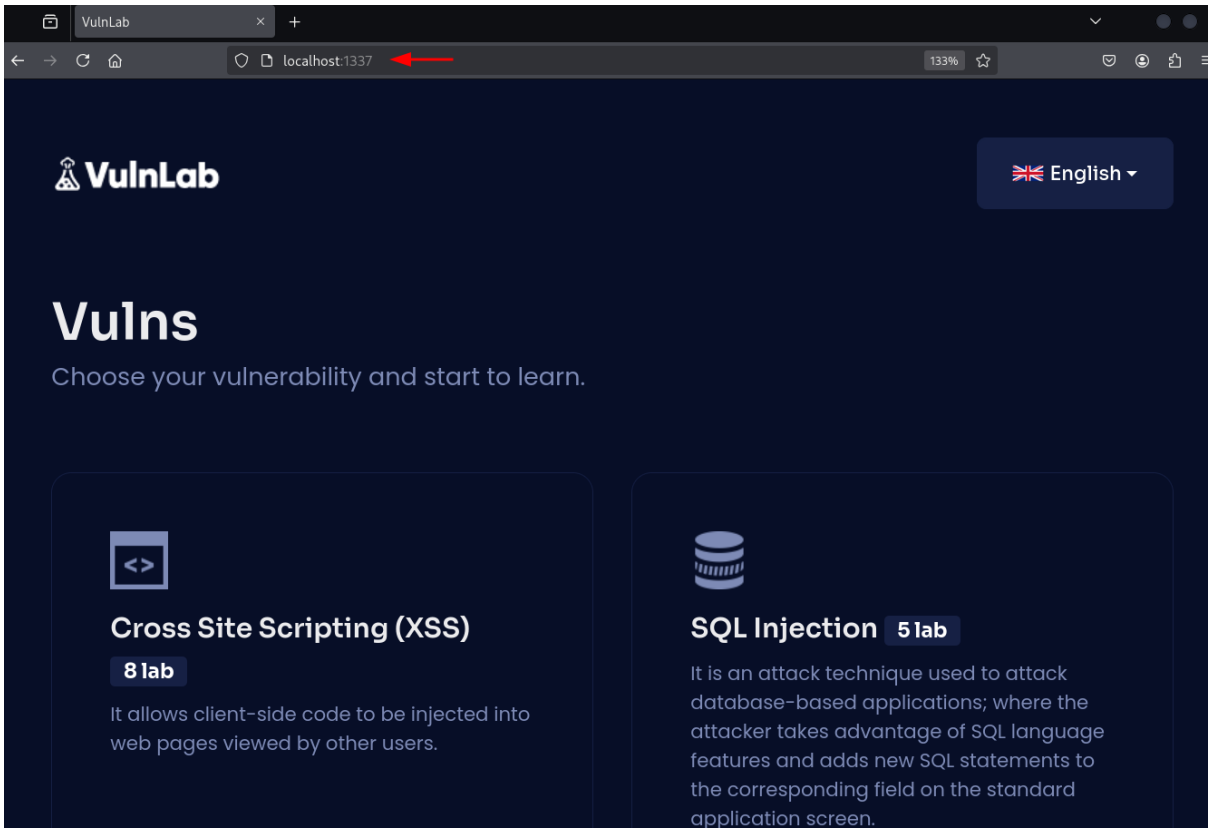
VulnLab is a web application that is designed to provide a hands-on environment for users to learn about web security and practice exploiting vulnerabilities. It covers a range of vulnerabilities, including SQL injection, XSS, and file inclusion, and is known for its realistic and challenging scenarios.

To configure VulnLab on Docker, you can use the following commands:

```
docker pull yavuzlar/vulnlab
docker run -d -p 1337:80 yavuzlar/vulnlab
```



Once you have started the VulnLab container, you can access it by visiting **http://localhost:1337** via your web browser. From there, you can begin your hands-on testing journey.



OWASP WrongSecrets

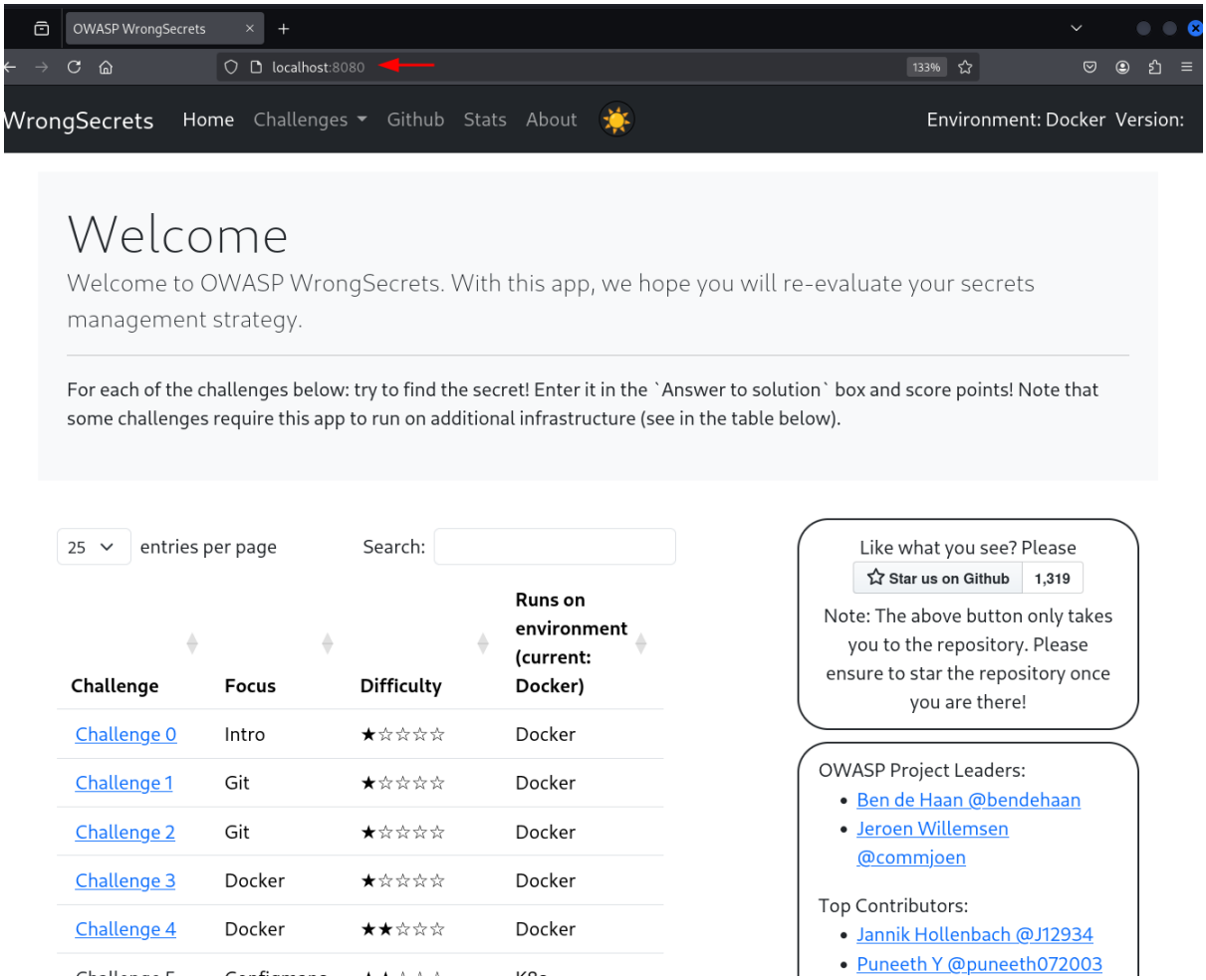
WrongSecrets is a web application that is designed to teach users about secrets management and how to properly store and manage sensitive data.

To configure WrongSecrets on Docker, you can use the following commands:

```
docker run -p 8080:8080 jeroenwillemsen/wrongsecrets:latest-no-vault
```

```
┌──(root㉿kali)-[~]
└─# docker run -p 8080:8080 jeroenwillemsen/wrongsecrets:latest-no-vault

Unable to find image 'jeroenwillemsen/wrongsecrets:latest-no-vault' locally
latest-no-vault: Pulling from jeroenwillemsen/wrongsecrets
f18232174bc9: Already exists
06b4ddc782f3: Pull complete
956bc91adb8d: Pull complete
2bcb7041227e: Pull complete
b6d8cc948f02: Pull complete
12116a1697f7: Pull complete
4f4fb700ef54: Pull complete
be5303f2858b: Pull complete
9c173956a718: Pull complete
9934a6577103: Pull complete
c49197982637: Pull complete
3266911ebc73: Pull complete
5873fddd6690: Pull complete
f5e62c2ba977: Pull complete
cc4f1c6a1008: Pull complete
c2a50e42cb17: Pull complete
69d177738dc3: Pull complete
1e46747d5a81: Pull complete
d8ea8c4447f4: Pull complete
681f5f06162d: Pull complete
cfece9b6c5b6: Pull complete
1054103af020: Pull complete
f42f45713f0f: Pull complete
Digest: sha256:8bf8f473a507c8722f360d9d761c1eebf7a3f5be8f18df7865cca708e626fdb8
Status: Downloaded newer image for jeroenwillemsen/wrongsecrets:latest-no-vault

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.4.4)
```

Once you have started the WrongSecrets container, you can access it by visiting **http://localhost:8080** via your web browser. From there, you can begin your hands-on testing journey.



Yrprey

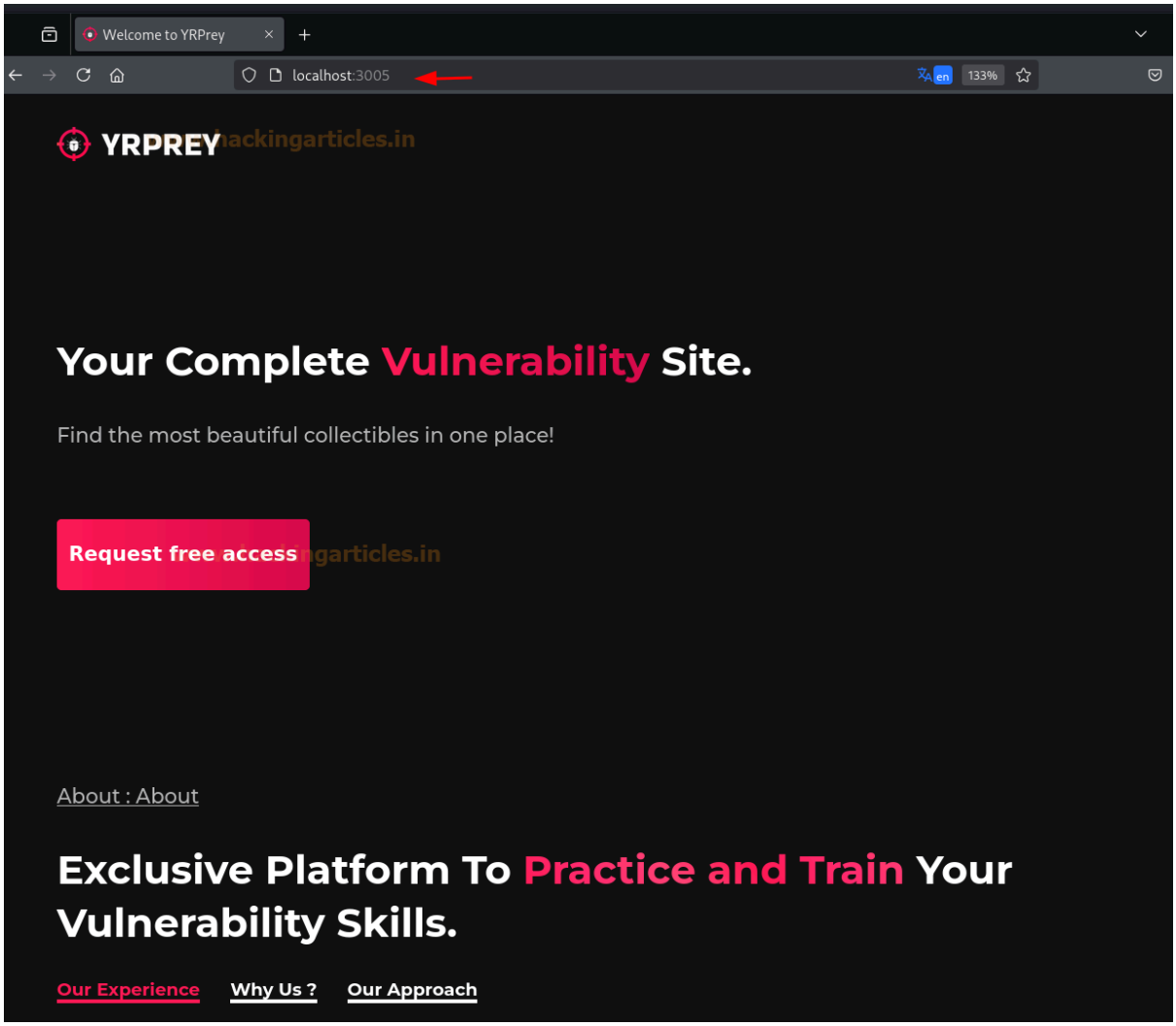
Yrprey is a web application that is designed to simulate a vulnerable web application, allowing users to practice exploiting common web vulnerabilities such as SQL injection and cross-site scripting (XSS).

To configure Yrprey on Docker, you can use the following commands:

```
git clone https://github.com/yrprey/yrprey-application.git
cd yrprey-application
docker-compose -f.docker/dev/app.yaml -p yrprey up -d
```

```
┌──(root㉿kali)-[~/yrprey-application]
└─# git clone https://github.com/yrprey/yrprey-application.git
Cloning into 'yrprey-application'...
remote: Enumerating objects: 324, done.
remote: Counting objects: 100% (324/324), done.
remote: Compressing objects: 100% (232/232), done.
remote: Total 324 (delta 66), reused 314 (delta 60), pack-reused 0 (from 0)
Receiving objects: 100% (324/324), 1.52 MiB | 4.28 MiB/s, done.
Resolving deltas: 100% (66/66), done.

┌──(root㉿kali)-[~/yrprey-application]
└─# cd yrprey-application

┌──(root㉿kali)-[~/yrprey-application/yrprey-application]
└─# docker-compose -f .docker/dev/app.yaml -p yrprey  up -d
WARN[0000] /root/yrprey-application/yrprey-application/.docker/dev/app.yaml: the
[+] Running 50/50
 ✔ backend Pulled
   ✔ ca7dd9ec2225 Pull complete
   ✔ 1b78b4fe0ca1 Pull complete
   ✔ 9d6040f2a28f Pull complete
   ✔ 0e2e66b89284 Pull complete
   ✔ 3b1be5f02bec Pull complete
   ✔ 96243f515dda Pull complete
   ✔ e427679e7c26 Pull complete
   ✔ 759eb390abda Pull complete
   ✔ 9820b782a628 Pull complete
   ✔ a9e6097c1efa Pull complete
```

Then, once you have started the Yrprey container, you can access it by visiting **http://localhost:3005** via your web browser. From there, you can begin your hands-on testing journey.



Zero Health

Zero Health is a web application that is designed to simulate a vulnerable healthcare system, allowing users to practice exploiting common web vulnerabilities such as SQL injection and cross-site scripting (XSS).

To configure Zero Health on Docker, you can use the following commands:

```
git clone https://github.com/aligorithm/zero-health.git
cd zero-health
docker-compose up
```

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/aligorithm/zero-health.git

Cloning into 'zero-health' ...
remote: Enumerating objects: 346, done.
remote: Counting objects: 100% (346/346), done.
remote: Compressing objects: 100% (204/204), done.
remote: Total 346 (delta 205), reused 259 (delta 128), pack-reused 0 (f
Receiving objects: 100% (346/346), 718.94 KiB | 3.18 MiB/s, done.
Resolving deltas: 100% (205/205), done.

┌──(root㉿kali)-[~]
└─# cd zero-health

┌──(root㉿kali)-[~/zero-health]
└─# docker-compose up
WARN[0000] /root/zero-health/docker-compose.yml: the attribute `version
[+] Running 17/17
 ✔ ollama Pulled
   ✔ 13b7e930469f Pull complete
   ✔ 97ca0261c313 Pull complete
   ✔ f6a9ed9582e4 Pull complete
   ✔ f6b71baa717c Pull complete
 ✔ db Pulled
   ✔ fe07684b16b8 Pull complete
   ✔ ef2c06670672 Pull complete
   ✔ db999d1f8078 Pull complete
   ✔ 42965b0a5926 Pull complete
   ✔ 84add0d959f4 Pull complete
   ✔ 47e14f06c3ba Pull complete
```

Then, once you have started the Zero Health container, you can access it by visiting **http://localhost:3000** via your web browser. From there, you can begin your hands-on testing journey.

Conclusion

Vulnerable web applications offer hands-on opportunities for both Red and Blue Teams to sharpen their skills. By mapping these apps to the MITRE ATT&CK framework, security professionals can simulate real-world attack chains and improve their defense posture. Docker makes deploying these labs easy and repeatable. Whether testing XSS, SQLi, or API abuse, always follow ethical guidelines and use them in isolated environments.

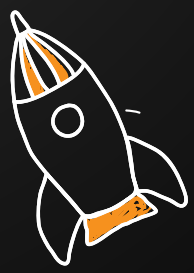
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

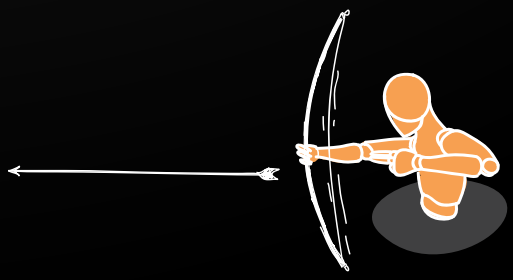
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux