2. Cross-site Scripting (XSS)

Objective: To learn about XSS attacks and how to mitigate them.

Tools: DVWA

What is XSS?

Cross-Site Scripting (XSS) is a type of security vulnerability commonly found in web applications. It allows attackers to inject malicious scripts into content that other users will see. This can lead to a variety of harmful outcomes, such as stealing session cookies, redirecting users to malicious sites, or defacing web pages. Here are the main types of XSS attacks:

1. **Stored XSS:** The malicious script is permanently stored on the target server, such as in a database, and is served to users who visit the affected page. For example, if a user posts a comment containing a malicious script, and this comment is displayed to others, it's a stored XSS attack.

2. **Reflected XSS:** The malicious script is reflected off a web server, usually via a URL or request parameter. This type of XSS is often used in phishing attacks. The malicious payload is sent to the server and then immediately reflected back to the user's browser.

3. **DOM-based XSS:** The vulnerability exists in the client-side code rather than on the server. In this case, the malicious script manipulates the DOM (Document Object Model) of the page, potentially altering how content is displayed or how the page interacts with the user.

 # How XSS Attacks Work

1. Injection: The attacker injects a malicious script into a web application.
2. Execution: The injected script is executed in the context of the victim's browser, often with the same permissions as the user.
3. Impact: The script can perform actions such as capturing sensitive data, manipulating the user's view of the site, or redirecting the user to malicious sites.

 # Preventing XSS

1. Input Validation: Ensure all user inputs are validated and sanitized. Reject or encode dangerous characters before storing or processing user input.
2. Output Encoding: Encode data before rendering it on web pages to prevent it from being executed as code. For example, use HTML entity encoding to display user input safely.
3. Use Security Libraries and Frameworks: Many modern frameworks and libraries include built-in protection against XSS. Utilize these tools to mitigate risks.
4. Content Security Policy (CSP): Implement a CSP to restrict the sources from which scripts can be loaded and executed. This helps to minimize the impact of XSS attacks.

Staying informed about best practices and continuously reviewing and testing your applications are crucial steps in maintaining security against XSS vulnerabilities.

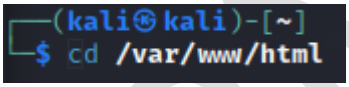
TOOL DVWA: DVWA, which stands for Damn Vulnerable Web Application, is a deliberately insecure web application designed to be used for educational and training purposes in the field of cybersecurity and ethical hacking. Created to provide a safe and controlled environment for learning about web application vulnerabilities and practicing penetration testing techniques, DVWA is not intended for use in production environments.

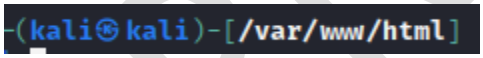
Installation of DVWA

Required Kali Linux, MySQL, Web Server Apache

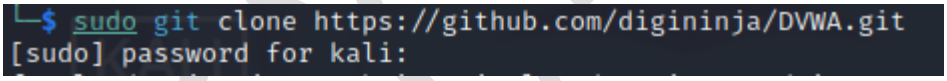
Step1: Download DVWA goto browser Google and type DVWA github then copy URL https://github.com/digininja/DVWA.git

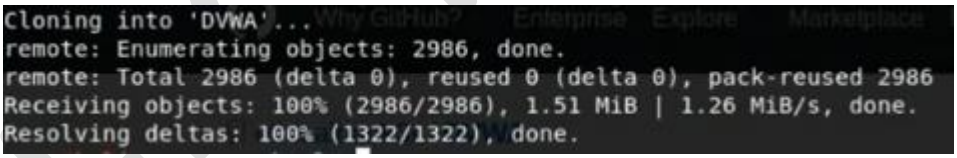
Step2: Goto terminal to download DVWA on local system and change the directory to run all web application

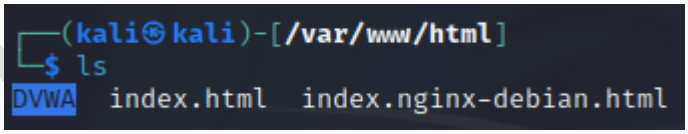


The default directory of web server 

Step3: if you're already a root user then no need to give sudo, otherwise provide the sudo at the beginning of the DVWA git link







Step3: Change the downloaded DVWA into executable, give all the permission

```
┌──(kali㉿kali)-[/var/www/html]
└─$ sudo chmod -R 777 DVWA

┌──(kali㉿kali)-[/var/www/html]
└─$ ls
DVWA  index.html  index.nginx-debian.html

┌──(kali㉿kali)-[/var/www/html]
└─$ cd DVWA

┌──(kali㉿kali)-[/var/www/html/DVWA]
└─$ ls
about.php       dvwa              phpinfo.php     README.md      security.txt
CHANGELOG.md    external          php.ini         README.pt.md   setup.php
compose.yml     favicon.ico       README.ar.md    README.tr.md   tests
config          hackable          README.es.md    README.vi.md   vulnerabilities
COPYING.txt     index.php         README.fa.md    README.zh.md
database        instructions.php  README.fr.md    robots.txt
Dockerfile      login.php         README.id.md    SECURITY.md
docs            logout.php        README.ko.md    security.php
```

```
┌──(kali㉿kali)-[/var/www/html/DVWA]
└─$ cd config
```

Config.inc.php.dist got the default configuration, just create the copy of this file with **.php** in future if have any issues then copy of the default values will be available

```
┌──(farhan㉿kali)-[/var/www/html/DVWA/config]
└─$ ls
config.inc.php.dist

┌──(farhan㉿kali)-[/var/www/html/DVWA/config]
└─$ cp config.inc.php.dist config.inc.php
```

Step4: Open config file using (nano[nano config.inc.php edit ctrl O to save and ctrl x] or vi editor) and hit enter

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo mousepad config.inc.php
```

Step5: After opening the config file, make changes with **dbuser=admin, dbpassword=password**, save and close the file

```
17 $_DVWA = array();
18 $_DVWA[ 'db_server' ]   = getenv('DB_SERVER') ?: '127.0.0.1';
19 $_DVWA[ 'db_database' ] = 'dvwa';
20 $_DVWA[ 'db_user' ]     = 'admin';
21 $_DVWA[ 'db_password' ] = 'password';
22 $_DVWA[ 'db_port']      = '3306';
```

Step6: Configure the database- first start the mysql (service mysql start)

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo systemctl start mysql

┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo systemctl status mysql
```

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo systemctl status mysql
● mariadb.service - MariaDB 10.11.2 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; disabled; preset: disabl>
     Active: active (running) since Wed 2024-09-04 12:11:27 EDT; 1min 0s ago
```

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo su
┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.11.2-MariaDB-1 Debian n/a
```

U-user default user name of sql is root

Step7: Login is done, after that fill the database

```
MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)
```

Create the user: The user name and password should be same as config file and 127.0.0.1 is the loop back address

```
MariaDB [(none)]> create user'admin'@'127.0.0.1' identified by 'password';
Query OK, 0 rows affected (0.088 sec)
```

Provide the previleges to this user

```
MariaDB [(none)]> grant all privileges on dvwa.* to 'admin'@'127.0.0.1';
Query OK, 0 rows affected (0.022 sec)
```

Then provide exit and clear the screen

Step8: Configure and start web server (or use this command to start service apache2 start)

```
┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# systemctl start apache2

┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; preset: disabl>
     Active: active (running) since Wed 2024-09-04 12:27:20 EDT; 56s ago
```

Step9: Configure the server with php file

```
┌──(root㉿kali)-[/var/www/html/DVWA/config]
└─# cd /etc/php

┌──(root㉿kali)-[/etc/php]
└─# ls
8.2

┌──(root㉿kali)-[/etc/php]
└─# cd 8.2

┌──(root㉿kali)-[/etc/php/8.2]
└─# cd apache2

┌──(root㉿kali)-[/etc/php/8.2/apache2]
└─# ls
conf.d  php.ini
```

Make the changes with php file with editor (or you can also use gedit php.ini)

```
┌──(root㉿kali)-[/etc/php/8.2/apache2]
└─# mousepad php.ini
```

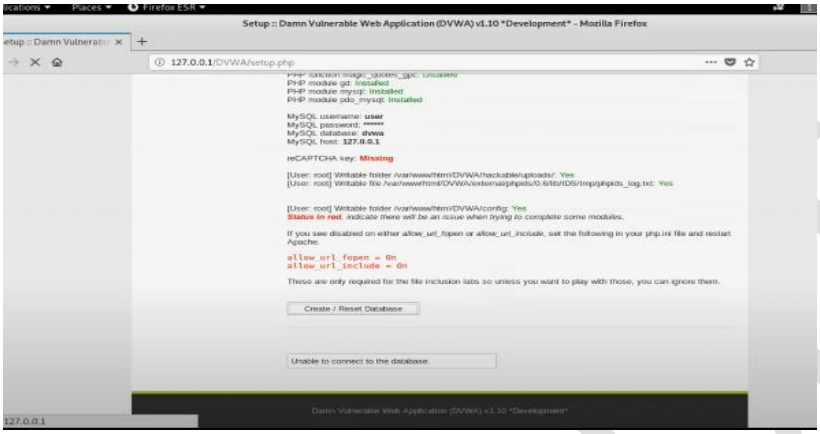
The fopen and include should be **on** condition

```
866 allow_url_fopen = On
867
868 ; Whether to allow include/require to open URLs (like https:// or ftp://) as
    files.
869 ; https://php.net/allow-url-include
870 allow_url_include = On
```

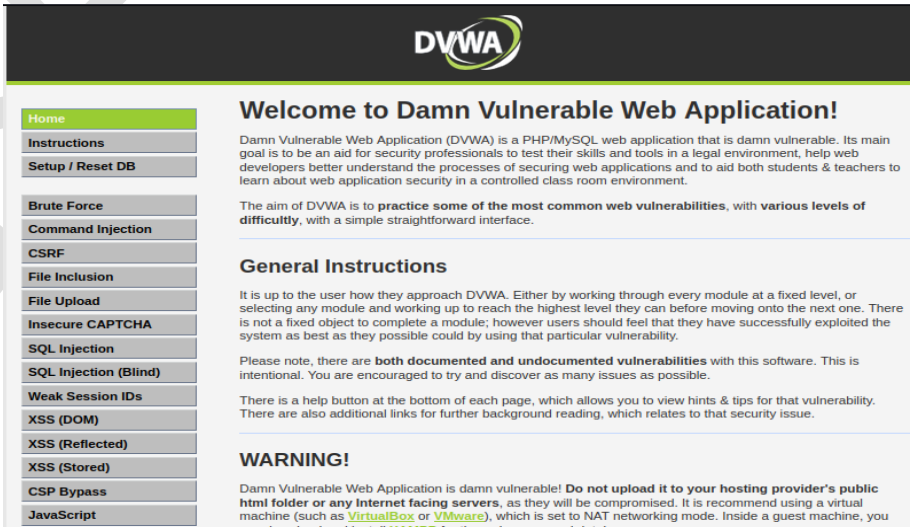
Step10. Restart the apache2 server (or use this command to start apache **service apache2 start**)

```
┌──(root㉿kali)-[/etc/php/8.2/apache2]
└─# systemctl restart apache2

┌──(root㉿kali)-[/etc/php/8.2/apache2]
└─# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; preset: disabl>
     Active: active (running) since Wed 2024-09-04 12:34:30 EDT; 10s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 33055 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCC>
   Main PID: 33060 (apache2)
      Tasks: 6 (limit: 2267)
     Memory: 10.5M
        CPU: 122ms
     CGroup: /system.slice/apache2.service
             ├─33060 /usr/sbin/apache2 -k start
             ├─33062 /usr/sbin/apache2 -k start
             ├─33063 /usr/sbin/apache2 -k start
             ├─33064 /usr/sbin/apache2 -k start
             ├─33065 /usr/sbin/apache2 -k start
             └─33066 /usr/sbin/apache2 -k start

Sep 04 12:34:30 kali systemd[1]: Starting apache2.service - The Apache HTTP Server>
Sep 04 12:34:30 kali apachectl[33059]: AH00558: apache2: Could not reliably determ>
Sep 04 12:34:30 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-20/20 (END)
```

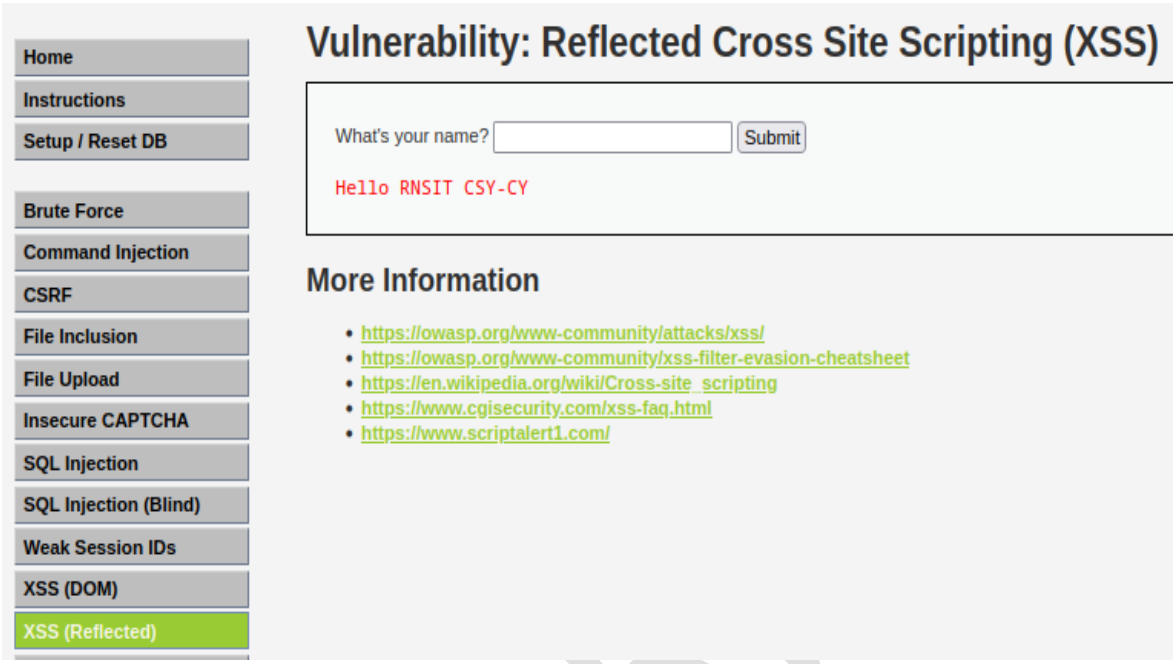
Step11: Goto browser and type **url as 127.0.0.1\DVWA**
Set the DVWA For the first time when it is used click on create



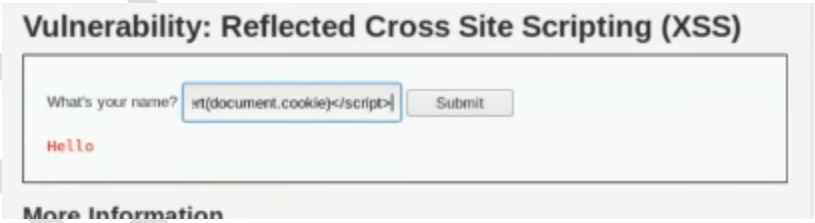
Provide username as **admin** and password as **password**
Next click on create/open db, and login again

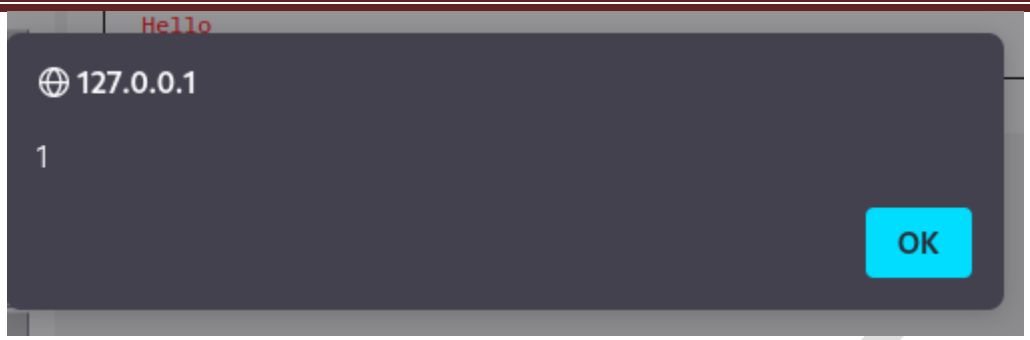


Example: XSS Reflected

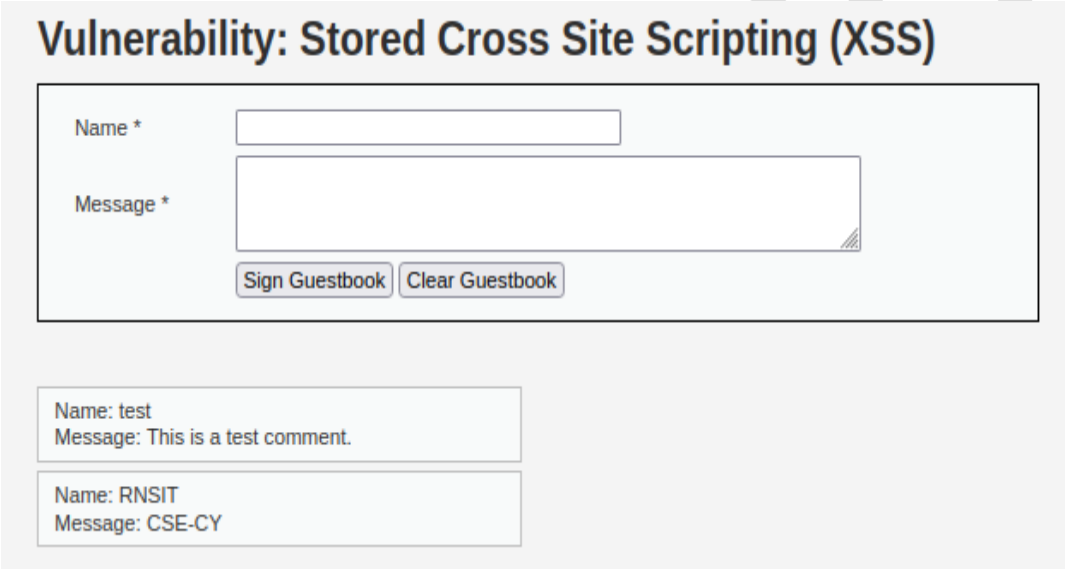


Example: Get cookies information: Session ID(Using Burp you can login to account without credentials) with low security level.
Try with different security level- medium, high



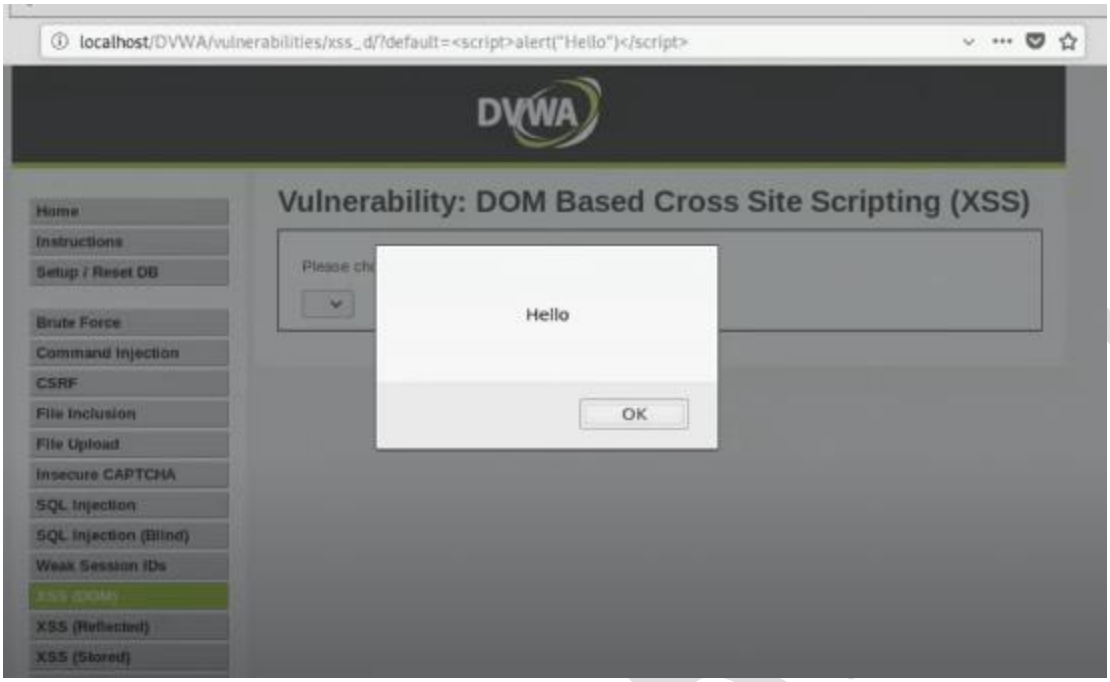


Stored: Example 1)



Example 2) Malicious script

DoM: no text box is available ,only we can use here is url



Ref link: https://www.youtube.com/watch?v=EoaDgUgS6QA
https://www.youtube.com/watch?v=cWu_FJUrH5Y&t=334s&ab_channel=edureka%21