

Traccia Esercizio:

Nella lezione pratica di oggi vedremo come configurare una DVWA – ovvero damn vulnerable web application in Kali Linux.

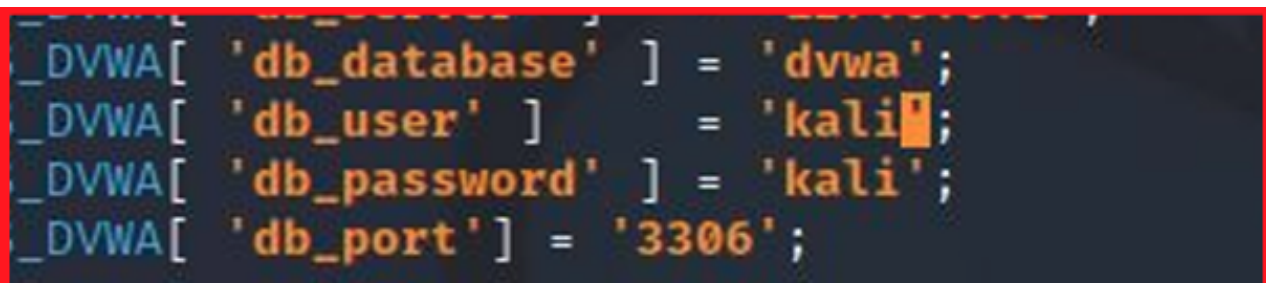
Esecuzione:

Requisiti: OS Kali Linux (OK), Database MySQL, Web Server Apache

Installazione:

```
cd /var/www/html
git clone https://github.com/digininja/DVWA
chown -R www-data:www-data DVWA/
cd DVWA/config
cp config.inc.php.dist config.inc.php
nano config.inc.php
```

Una volta aperto **NANO**, utilizzando le funzioni in basso suggerite cambiamo la configurazione utilizzando le credenziali fornite nel materiale



```
_DVWA[ 'db_database' ] = 'dvwa';
_DVWA[ 'db_user' ] = 'kali';
_DVWA[ 'db_password' ] = 'kali';
_DVWA[ 'db_port' ] = '3306';
```

```
service mysql start
mysql -u root -p
create user 'kali'@'127.0.0.1' identified by 'kali';
grant all privileges on dvwa.* to 'kali'@'127.0.0.1' identified by 'kali' ;
cd /etc/php/8.2/apache2 (modificate 8.2 con la versione corrente istallata)
service apache2 start
```

Apriamo il browser **127.0.0.1/DVWA/setup.php**

Selezioniamo alla fine **CREATE / RESET DATABASE**

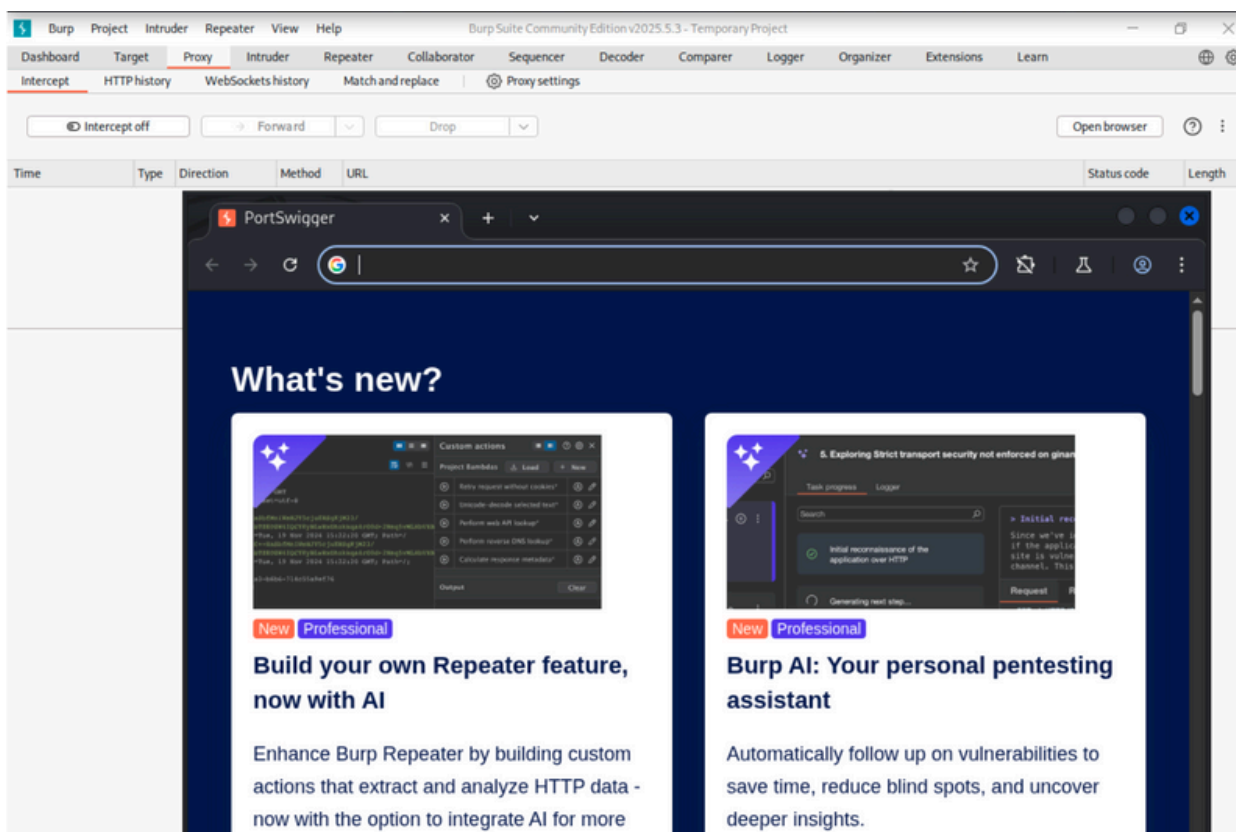
Se abbiamo seguito bene le indicazioni non avremmo problemi ad avviare il DVWA

Traccia Esercizio:

Ora che la nostra app è attiva, facciamo un po' di pratica con Burpsuite.

Lanciamo Burpsuite, scegliamo un progetto temporaneo ed apriamo un browser, inserendo l'indirizzo della nostra DVWA: 127.0.0.1/DVWA e inseriamo nei campi login e password i valori «admin» e «password» rispettivamente. Intercettiamo la richiesta con burp e vediamo come possiamo modificarla. Guardate i parametri di login, possiamo modificarli a nostro piacimento prima di inviare la richiesta all'app.

Apriamo **Burp Suite**, selezioniamo **Proxy**, apriamo il browser (**top dx**)

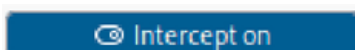


Attiviamo **"Intercept"** (top sx)

Ora la suite è pronta a **intercettere il traffico** e dal suo browser ci colleghiamo a **http://127.0.0.1/DVWA/index.php**

11:13:31 16 Jul 2025	HTTP	→ Request	GET	http://192.168.50.101/
11:13:42 16 Jul 2025	HTTP	→ Request	GET	http://127.0.0.1/DVWA/index.php

Vicino ad **"Intercept"** che ora sarà visualizzato così poichè attivato.



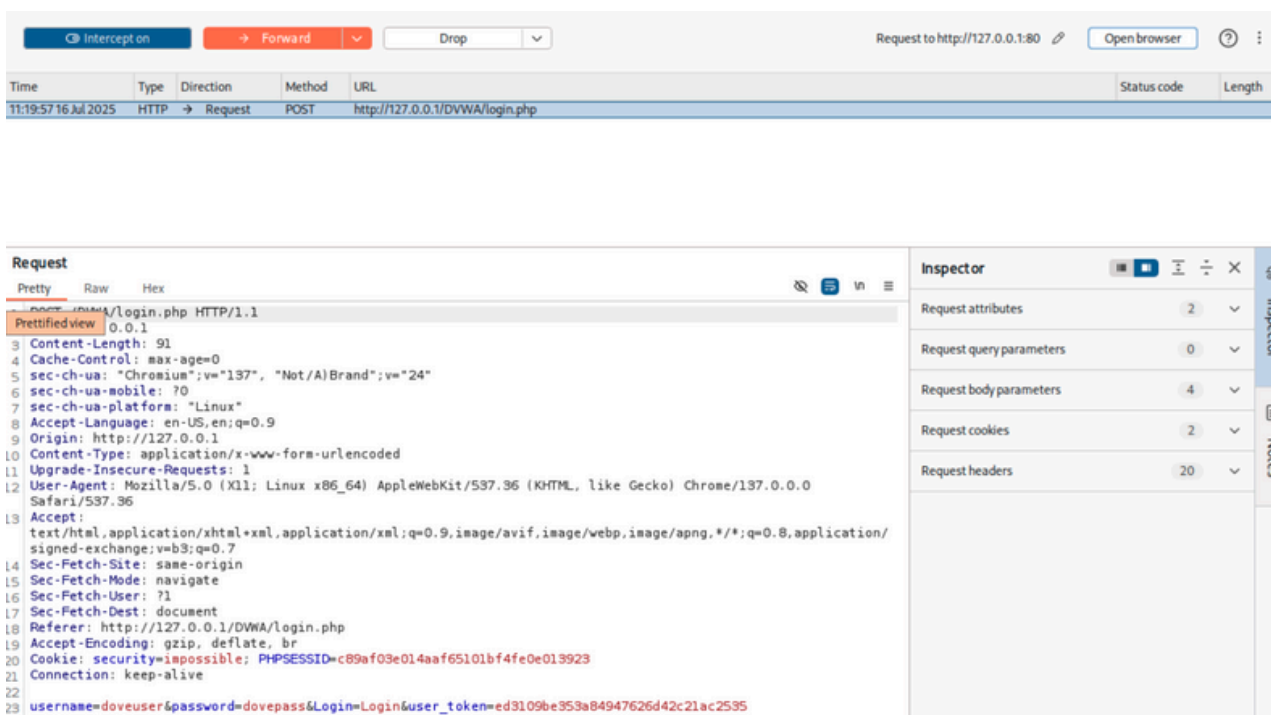
Abbiamo la funzione **"FORWARD"**.

Una volta che interagiamo con il browser, dovremmo tornare su **"Proxy"** e per **mandare la richiesta avanti** usiamo **"FORWARD"** (avanti di uno step la comunicazione).

Riaprendo il browser di burp ora ci troviamo di fronte alla pagina di richiesta di login.

Inseriamo delle **credenziali** a caso per vedere cosa succede.

Tornando su **Burp Proxy** possiamo osservare la seguente situazione



Request to http://127.0.0.1:80

Time	Type	Direction	Method	URL	Status code	Length
11:19:57 16 Jul 2025	HTTP	→ Request	POST	http://127.0.0.1/DVWA/login.php		

Request

Pretty Raw Hex

Prettified view 0.0.1

```

1 POST http://127.0.0.1/DVWA/login.php HTTP/1.1
2 Host: 0.0.1
3 Content-Length: 91
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Accept-Language: en-US,en;q=0.9
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://127.0.0.1/DVWA/login.php
19 Accept-Encoding: gzip, deflate, br
20 Cookie: security=impossible; PHPSESSID=c89af03e014aaf65101bf4fe0e013923
21 Connection: keep-alive
22
23 username=doveuser&password=dovepass&Login=Login&user_token=ed3109be353a84947626d42c21ac2535
  
```

Inspector

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 4
- Request cookies: 2
- Request headers: 20

Possiamo notare in basso

```
username=doveuser&password=dovepass&Login=Login&user_token=ed3109be353a84947626d42c21ac2535
```

Facciamo un test! Prendiamo **la richiesta da proxy**, tasto dx, e mandiamola al **repeater** con il comando **"Send to Repeater"**.

URL	
http://127.0.0.1/DVWA/login.php	<div> <div>http://127.0.0.1/DVWA/login.php</div> <div>Add to scope</div> <div>Forward</div> <div>Drop</div> <div>Add notes</div> <div>Highlight ></div> <div>Don't intercept requests ></div> <div>Do intercept ></div> <div>Scan</div> <div>Send to Intruder Ctrl+I</div> <div>Send to Repeater Ctrl+R</div> </div>
ind" ; v= " 24"	

Usiamo il **Repeater** per testare come risponde il server a delle **richieste manipolate**.

Request

	Pretty	Raw	Hex
1	POST /DVWA/login.php HTTP/1.1		
2	Host: 127.0.0.1		
3	Content-Length: 91		
4	Cache-Control: max-age=0		
5	sec-ch-ua: "Chromium";v="137", "Not/A)Brand";v="24"		
6	sec-ch-ua-mobile: ?0		
7	sec-ch-ua-platform: "Linux"		
8	Accept-Language: en-US,en;q=0.9		
9	Origin: http://127.0.0.1		
10	Content-Type: application/x-www-form-urlencoded		
11	Upgrade-Insecure-Requests: 1		
12	User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36		
13	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7		
14	Sec-Fetch-Site: same-origin		
15	Sec-Fetch-Mode: navigate		
16	Sec-Fetch-User: ?1		
17	Sec-Fetch-Dest: document		
18	Referer: http://127.0.0.1/DVWA/login.php		
19	Accept-Encoding: gzip, deflate, br		
20	Cookie: security=impossible; PHPSESSID= c89af03e014aaf65101bf4fe0e013923		
21	Connection: keep-alive		
22			
23	username=admin&password=password&Login=Login& user_token=ed3109be353a84947626d42c21ac2535		

Modifichiamo la **password** e l'**utente** e mandiamo la richiesta selezionando sopra la funzione **"SEND"**.

Ora a destra dello schermo possiamo vedere la risposta del server.

Ora andiamo in **"Proxy"** e dopo aver modificato user e password mandiamo avanti le **richieste** con **"FORWARD"**

Se apriamo il browser di **Burp**, possiamo notare che il **login** è stato **eseguito**

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript
- Authorisation Bypass
- Open HTTP Redirect
- Cryptography
- API
- DVWA Security
- PHP Info
- About
- Logout

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficulty**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerabilities** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- [Mutillidae](#)
- [OWASP Vulnerable Web Applications Directory](#)

Ora siamo liberi di testare le varie funzioni del programma e approfittare delle volute vulnerabilità per prendere confidenza con lo strumento