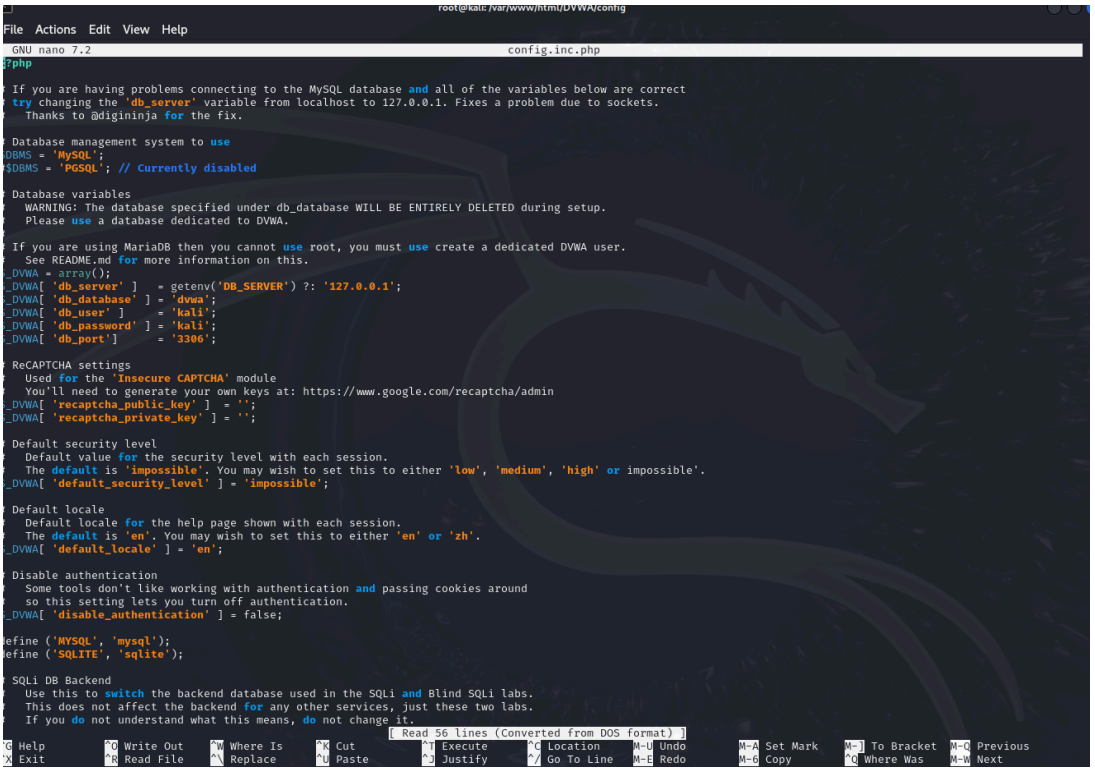
S3.L2 Web APP

1) Prima di tutto, acquisisco i privilegi di root tramite il comando sudo su nel mio terminale Kali Linux. Successivamente, mi sposto nella directory dei progetti web usando il percorso /var/www/html e clono il repository Damn Vulnerable Web Application (DVWA) utilizzando il comando git clone.
Dopo aver completato il download, assegno i permessi di lettura, scrittura ed esecuzione a tutti gli utenti per la directory DVWA utilizzando il comando chmod -R 777.
Successivamente, accedo alla directory di configurazione di DVWA e copio il file di configurazione di distribuzione nel file di configurazione attivo.
Per modificare il file di configurazione config.inc.php, avvio l'editor di testo nano.
All'interno del file, aggiorno le credenziali del database con l'username e la password 'kali', quindi salvo le modifiche e chiudo l'editor.
Ora sono pronto per utilizzare DVWA con le nuove credenziali di accesso!



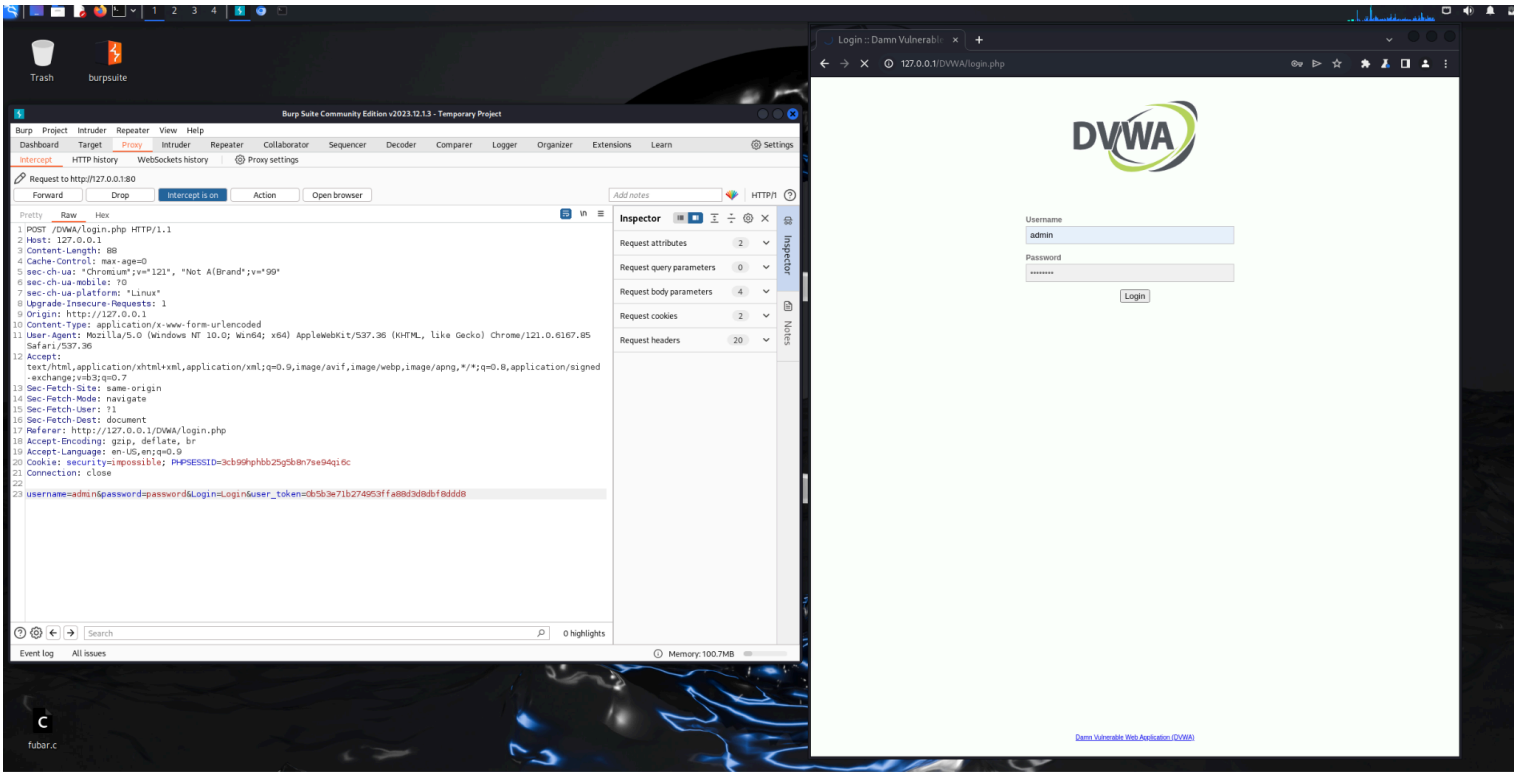
2) Ho appena avviato MariaDB sul mio sistema Kali Linux usando il comando sudo mariadb.
Ora, per accedere al server MariaDB come root, ho eseguito il comando mysql -u root -p.
Una volta dentro, ho creato un nuovo utente chiamato 'kali' che può accedere solo dall'indirizzo 127.0.0.1. Ho assegnato anche una password per questo utente con il comando:CREATE USER 'kali'@'127.0.0.1' IDENTIFIED BY 'kali';
Dopodiché, ho garantito all'utente 'kali' tutti i privilegi sul database 'dvwa', necessari per gestire il database della Damn Vulnerable Web Application, utilizzando il comando:GRANT ALL PRIVILEGES ON dvwa.* TO 'kali'@'127.0.0.1';
Ho controllato attentamente che non ci fossero errori dopo l'esecuzione dei comandi.
Infine, per uscire dal prompt di MariaDB, ho digitato exit;.
Ora sono pronto per utilizzare il database 'dvwa' con l'utente 'kali' che ho appena creato!

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 33
Server version: 10.11.6-MariaDB-2 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT User, Host FROM mysql.user WHERE User = 'kali';
+------+-----------+
| User | Host      |
+------+-----------+
| kali | 127.0.0.1 |
+------+-----------+
1 row in set (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON dvwa.* TO 'kali'@'127.0.0.1' IDENTIFIED BY 'kali';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> exit
Bye

┌──(root㉿kali)-[/home/kali/Desktop]
└─#
```

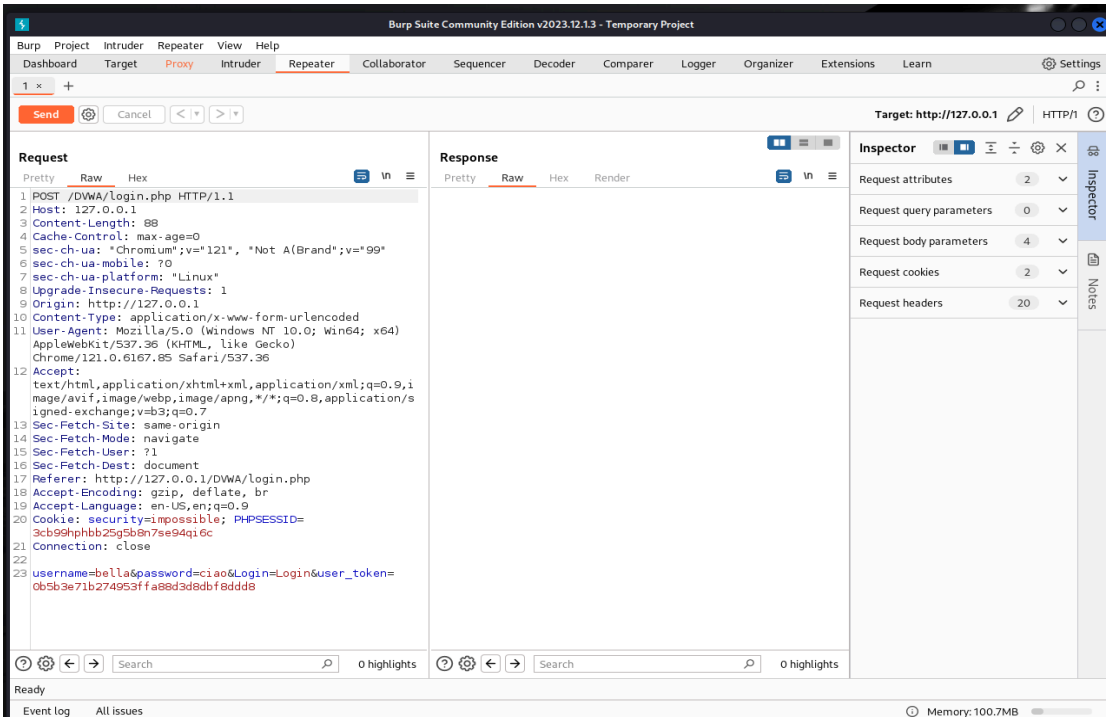
3) Ho avviato il servizio Apache digitando service apache2 start nel terminale. Poi mi sono spostato nella directory /etc/php/8.2/apache2 con il comando cd /etc/php/8.2/apache2. Per assicurarmi di lavorare sulla versione corretta di PHP, ho eseguito prima il comando ls nella directory /etc/php per controllare le sottodirectory presenti. Ho aperto il file php.ini con il mio editor di testo preferito per modificare le impostazioni allow_url_fopen e allow_url_include assicurandomi che fossero entrambe impostate su ON. Dopo aver salvato le modifiche, ho eseguito nuovamente il comando service apache2 start per riavviare il servizio Apache con la nuova configurazione.

```
root@kali: /etc/php/8.2/apache2
File  Actions  Edit  View  Help
  GNU nano 7.2                                  /etc/php/8.2/apache2/php.ini *
; RFC2616 compliant header.
; Default is zero.
; https://php.net/cgi.rfc2616-headers
;cgi.rfc2616_headers = 0

; cgi.check_shebang_line controls whether CGI PHP checks for line starting with #!
; (shebang) at the top of the running script. This line might be needed if the
; script support running both as stand-alone script and via PHP CGI<. PHP in CGI
; mode skips this line and ignores its content if this directive is turned on.
; https://php.net/cgi.check-shebang-line
;cgi.check_shebang_line=1

;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;

; Whether to allow HTTP file uploads.
; https://php.net/file-uploads
file_uploads = On

; Temporary directory for HTTP uploaded files (will use system default if not
; specified).
; https://php.net/upload-tmp-dir
;upload_tmp_dir =

; Maximum allowed size for uploaded files.
; https://php.net/upload-max-filesize
upload_max_filesize = 2M

; Maximum number of files that can be uploaded via a single request
max_file_uploads = 20

;;;;;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; https://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like https:// or ftp://) as files.
; https://php.net/allow-url-include
allow_url_include = On

; Define the anonymous ftp password (your email address). PHP's default setting
; for this is empty.
; https://php.net/from
;from="john@doe.com"

; Define the User-Agent string. PHP's default setting for this is empty.
; https://php.net/user-agent
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark     M-] To Bracket   M-Q Previous
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy         ^Q Where Was     M-W Next
```

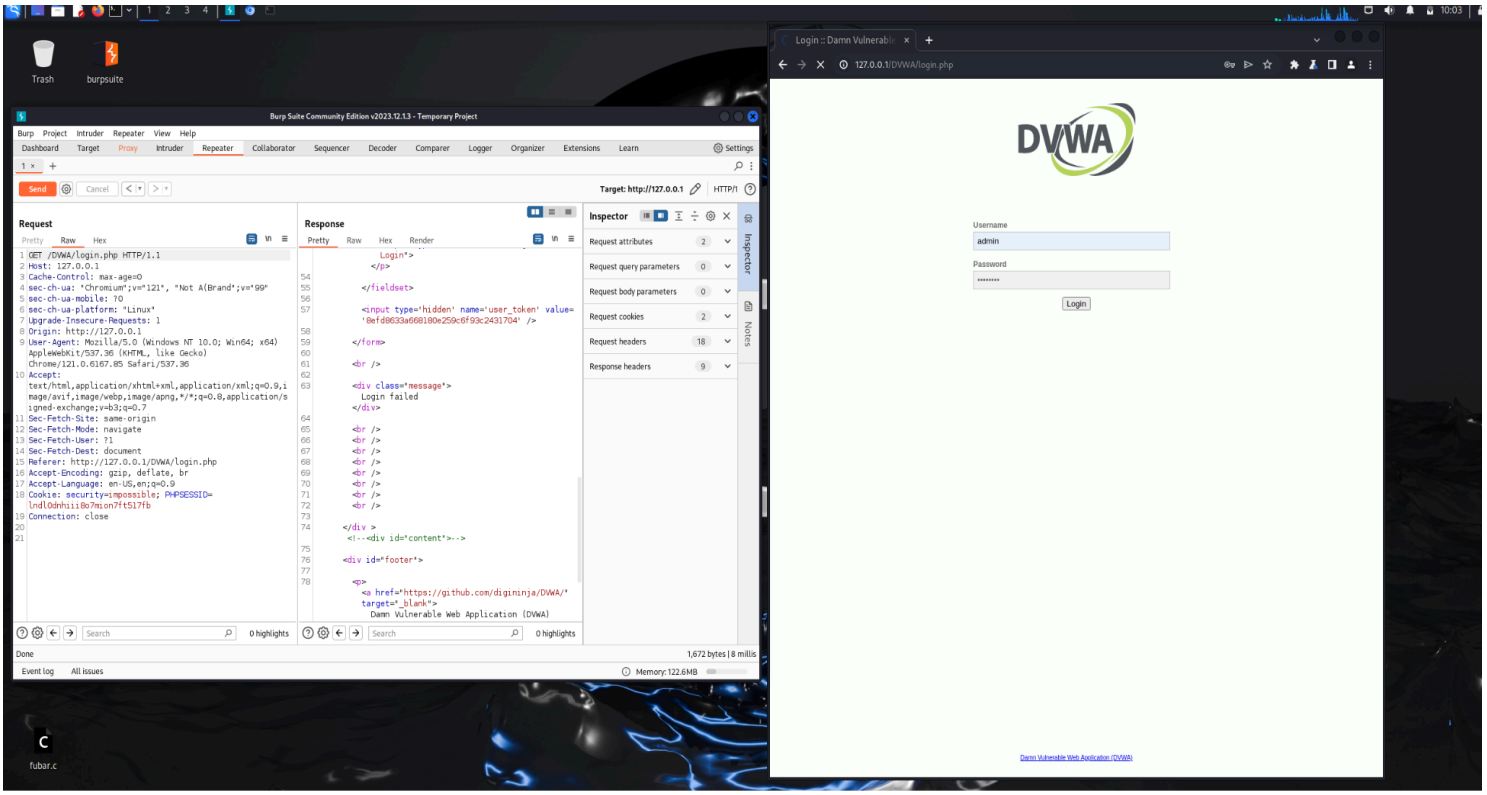
4) Successivamente alla configurazione della DVWA vado ad aprire Burpsuite per fare delle prove di sniffing. Ho avviato Burp Suite e impostato il browser per passare attraverso il proxy di Burp Suite. Dopo aver inserito le credenziali di accesso nella Damn Vulnerable Web Application (DVWA), ho attivato l'intercettazione delle richieste in Burp Suite. Nel momento in cui ho tentato di effettuare il login nell'applicazione DVWA, Burp Suite ha intercettato la richiesta di login. Come risultato, sono riuscito a visualizzare e raccogliere le credenziali utilizzate nel tentativo di accesso. La richiesta intercettata mostrava chiaramente i parametri POST con l'username admin e la password che avevo inserito, che era lo scopo finale: lo sniffing delle credenziali di accesso.



5) Dopo aver intercettato la richiesta di login con Burp Suite, ho modificato le credenziali inserendo volutamente dei valori errati. Prima di inoltrare la richiesta modificata, ho utilizzato il tasto destro del mouse per selezionare l'opzione “send to repeater” per avere maggiore controllo sull'invio delle richieste. Ho quindi inviato la richiesta errata di login al server premendo il pulsante "send" in Repeater e successivamente ho cliccato su "follow redirection" per seguire automaticamente qualsiasi reindirizzamento risposto dal server.



6) Come ultimo passaggio vado a dimostrare come dall'opzione Repeater nella riga 63 mi darà come messaggio d'errore 'login failed'.



IN CONCLUSIONE:

Ho coperto diversi aspetti fondamentali della sicurezza delle applicazioni web, dalla configurazione di un ambiente di sviluppo sicuro con DVWA e MariaDB su Kali Linux, fino all'utilizzo di strumenti come Apache, PHP e Burp Suite per testare e analizzare la sicurezza delle applicazioni.
Ho imparato come configurare correttamente DVWA e MariaDB, incluso l'assegnare correttamente i privilegi agli utenti e la gestione delle impostazioni di sicurezza. ho anche esplorato l'importanza di configurare correttamente il server Apache e PHP per garantire una corretta gestione delle vulnerabilità potenziali.
Inoltre, ho sperimentato l'utilizzo di Burp Suite per il testing di sicurezza, compreso lo sniffing delle credenziali e la manipolazione delle richieste per individuare potenziali vulnerabilità e problemi di sicurezza.