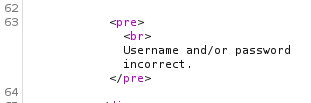
**BW1/L4**

Un “Brute force” è come dice la parola stessa un attacco di forza bruta che mira al recupero delle credenziali di accesso.   
In particolare, tale attacco cerca di trovare la giusta combinazione di username e password che ricava da dizionari o liste, che contengono i suddetti dati più comuni.

Ora andremo ad analizzare un attacco brute force sul sito di DVWA che ci mette a disposizione vari livelli di sicurezza, partiremo dal “low” per poi passare al “medium” e “high”, questi sono i livelli che prenderemo in considerazione per oggi  
Il primo passo per poter effettuare il login è la richiesta “GET”. Questa richiesta è uno dei metodi utilizzati nel protocollo http per richiedere dati da un server web. Quando si effettua una richiesta GET, nel caso del login, il client (ad esempio un browser web) include in essa dei parametri: username, password e login. Il server quando riceve questa request, confronterà i parametri della get con quelli salvati nel database, se c’è corrispondenza sarà effettuato l’accesso, altrimenti avremo come risposta un errore.

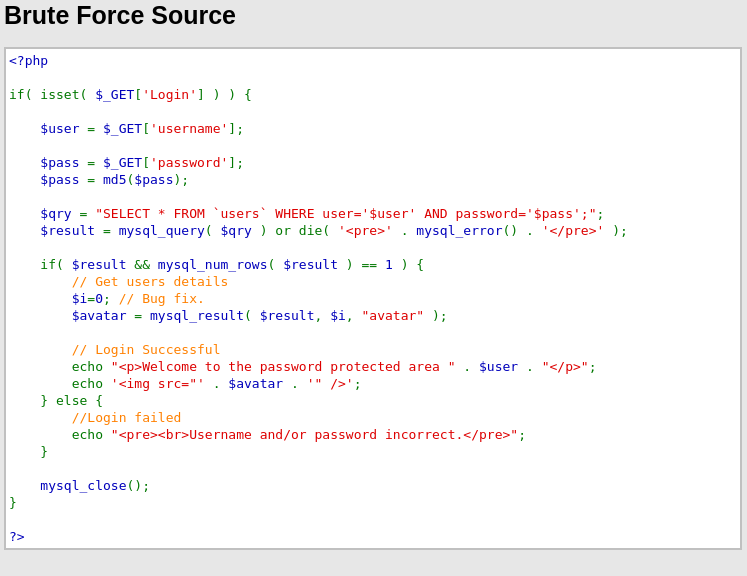
**Credenziali errate Credenziali corrette**

****

il server web DVWA non impone un limite di richieste http da poter fare, quindi possiamo eseguirne infinite con l’attacco brute force finché non avremo la combinazione giusta di username e password.

**Low**

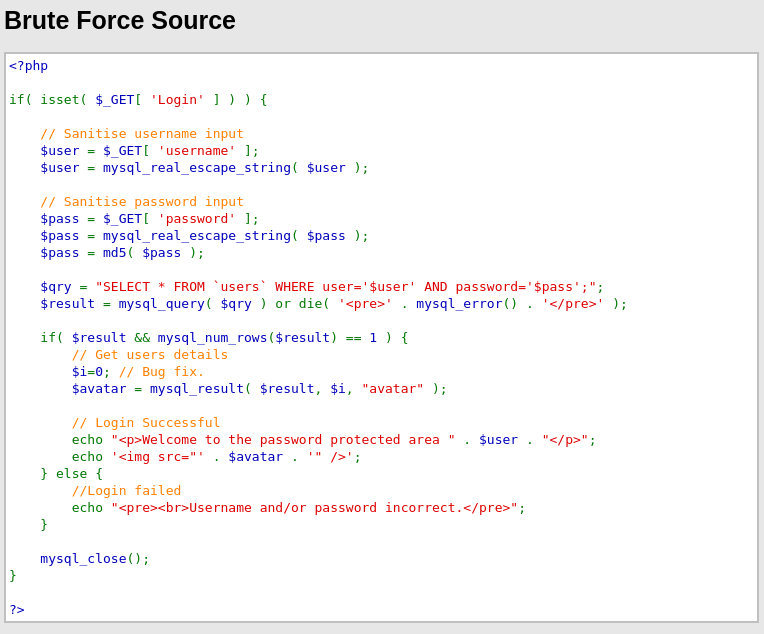
Possiamo vedere il codice **php** lato server legato al servizio di login della pagina DVWA



**Sicurezza:** l'unica cosa che il server richiede dall'utente che prova loggarsi è l’inserimento delle correte credenziali. In particolare il server confronterà i campi di interesse immessi dall'utente, come username e password, con i dati salvati nel proprio database, e se i dati sono corretti permetterà l'accesso.

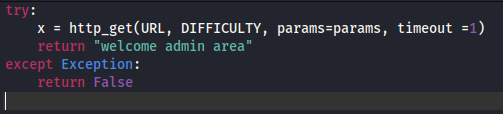
**Come aggirarla:** Dovremmo creare un codice che continua ad inserire varie combinazioni di credenziali possibili per risalire a quelle corrette; le librerie o liste svolgono un ruolo importante nel fornirci varie scelte.

**Medium**

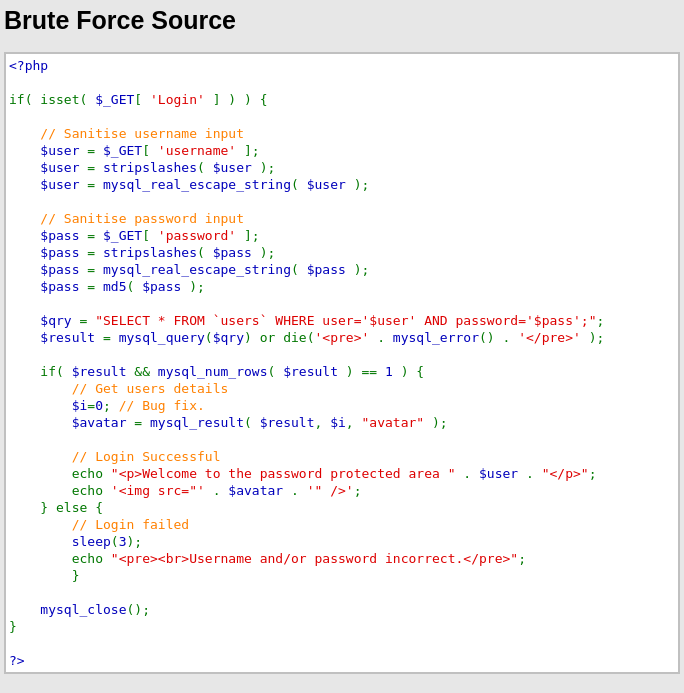
A questo livello il codice **php** lato server si è evoluto aggiungendo il **sanitise** e lo **sleep**

**Sicurezza:** in questo livello di difficoltà si possono notare due differenze rispetto al livello precedente: la sanitizzazione e lo sleep. La sanitizzazione dei dati è un processo utilizzato per pulire e proteggere i dati dalle minacce di sicurezza. Quando si accettano input dagli utenti, è essenziale applicare la sanitizzazione per evitare che questi possano essere utilizzati per compromettere la sicurezza del sistema. Lo sleep è il periodo di tempo che il server attende prima di inviare la risposta al client nel caso in cui la prova di accesso abbia esito negativo. Questa tecnica serve per mitigare gli attacchi di forza bruta, perché aggiungendo questo sleep tra una richiesta e l’altra l’attacco rallenta significativamente e a volte inefficace

1. **Come aggirarla:** Un attaccante può bypassare questa attesa basando i suoi attacchi sul tempo di risposta del server. Ad esempio, con burpsuite possiamo controllare quanto tempo impiega il server a dare una risposta in caso di esito positivo. Supponiamo che questo tempo sia di 0,2 secondi, ogni richiesta che supererà questi secondi per avere una risposta sarà una richiesta fallita. di seguito mostriamo la parte di codice usata per aggirare il problema dei tempi di risposta del server. Infatti configurando il parametro timeout il nostro codice non attenderà la risposta del server per la verifica delle corrette credenziali, ma andrà a provare l combinazione di credenziali successiva se la precedente supera il timeout di 1 secondo. A seconda di quanto veloce ci risponde il server abbiamo, sappiamo subito se abbiamo preso o no le credenziali corrette



**High**



**Sicurezza:** Nella modalità high lo sleep non sarà più un periodo di tempo statico, ma ad ogni richiesta potrà variare in un certo intervallo di tempo dinamico, viene infatti definito sleep randomico. Viene introdotta un’altra fondamentale differenza: il csfr token. Nel codice della richiesta da parte del client verrà aggiunto un'altra variabile (type=Hidden) e cioè il token, il cui valore sarà un codice alfanumerico casuale generato dal server ad ogni richiesta http di accesso dell’utente. il server verifica quindi che il token csfr inviato corrisponda a quello memorizzato per l’utente e per la sessione in corso. Se il token non è corretto o è assente, il server rifiuterà la richiesta.

**Come aggirarla:** Ancora una volta gli attaccanti hanno trovato il modo per aggirare le tecniche di sicurezza implementate. In questo caso, per poter effettuare un brute force a questo livello, le richieste di get saranno due: la prima servirà per chiedere al server il codice del token, senza inserire le credenziali d’accesso; la seconda sarà quella effettiva, con l’inserimento sia dei dati che del token. In questo modo le richieste da eseguire per ogni tentativo raddoppiano aumentandone il tempo richiesto per completare l’attacco.   
(Aggiungere il token da burpstuit)

