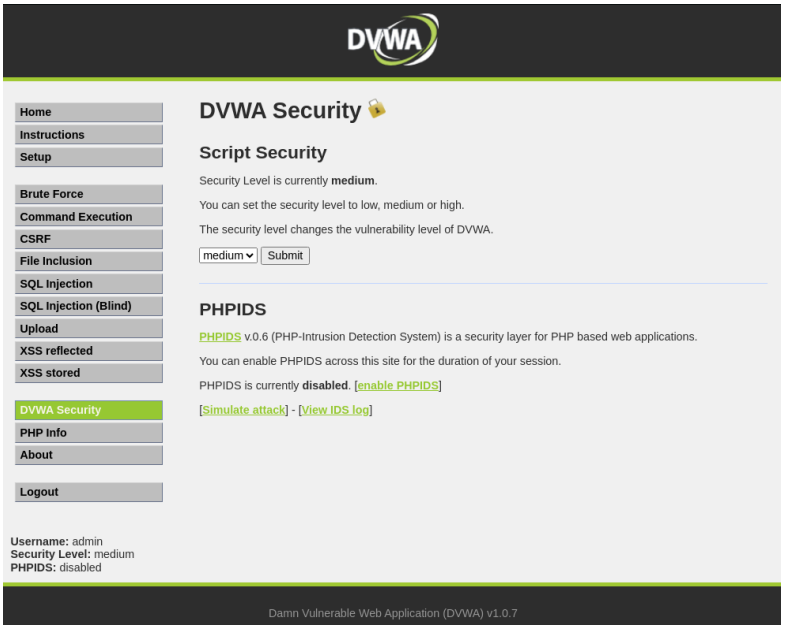
Extra S6-L2
CSS e SQL Injection su DVWA livello MEDIUM

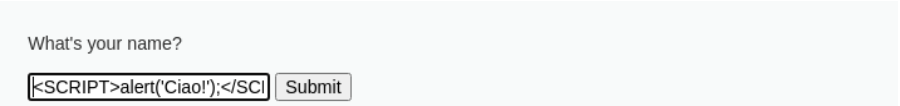
- Per prima cosa ho impostato il livello di sicurezza della **dvwa** su **MEDIUM**



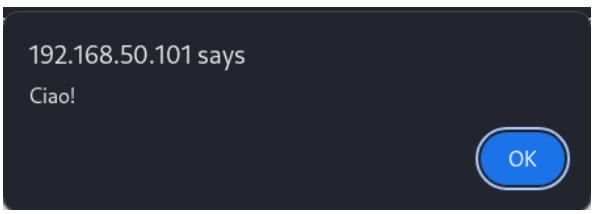
La prima cosa di cui mi sono occupato è stata inserire uno script in **XSS Reflected**. Per capire come fare ho analizzato il codice sorgente della pagina

```php
<?php

if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){

 $isempty = true;

} else {

 echo '<pre>';
 echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
 echo '</pre>';

}

?>
```

Pur non conoscendo il linguaggio **PHP**, ho capito, grazie a delle prove effettuate, che ciò che si limitava a fare tale codice era sostituire la stringa **“<script>”** con **“”**.
Considerando la differenza tra maiuscole e minuscole presente in Python, ho provato ad applicare qui lo stesso principio, scrivendo il comando in questo modo: **<SCRIPT>**.



Il risultato è che lo script è stato letto correttamente restituendo:

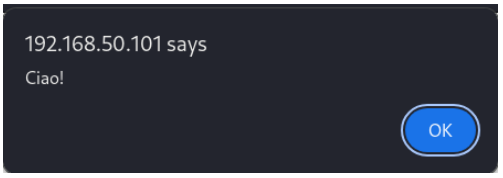


- Successivamente sono passato ad analizzare il codice sorgente della pagina **XSS Stored**:

Stored XSS Source

```php
<?php

if(isset($_POST['btnSign']))
{

    $message = trim($_POST['mtxMessage']);
    $name    = trim($_POST['txtName']);

    // Sanitize message input
    $message = trim(strip_tags(addslashes($message)));
    $message = mysql_real_escape_string($message);
    $message = htmlspecialchars($message);

    // Sanitize name input
    $name = str_replace('<script>', '', $name);
    $name = mysql_real_escape_string($name);

    $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";

    $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>' );

}

?>
```

Analizzando il codice ho capito che nel corpo del messaggio, la sanitizzazione, avveniva in modo differente rispetto a quanto visto prima. Essa infatti avviene grazie all'istruzione **"htmlspecialchars"** (vista nelle slides) che modifica alcuni caratteri speciali in modo da impedire l'inserimento di script.
Tuttavia ho notato anche che nella sezione **"name"** il discorso era diverso e che il blocco presente qui era uguale a quello presente in **XSS Reflected**.
Pertanto ho inserito lo stesso script con le maiuscole nella sezione **"name"** ed effettivamente lo script è stato letto.



- Infine mi sono occupato della **SQL Injection**. Il primo passo è stato sempre quello di analizzare il codice sorgente:

```php
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i=0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Inizialmente non ho trovato nulla di strano in tale codice, così ho pensato di confrontarlo con il codice della stessa pagina ma al livello **LOW**.
L'unica differenza tra i due, risiede nell'istruzione **"mysql_real_escape_string"**.
Informandomi ho scoperto che tale funzione blocca alcuni caratteri speciali, tra cui le virgolette (' o "), responsabili quindi del blocco delle mie richieste.
Così ho iniziato a documentarmi per individuare un modo alternativo per utilizzare tali caratteri "senza utilizzarli".
Ho scoperto che è possibile scrivere ' scrivendo al suo posto **CHAR (39)**, infatti MySQL in questo modo, andrà a leggere il carattere che corrisponde al **39** in **ASCII**.
Consapevole di ciò ho formulato una richiesta per verificare il funzionamento:

CHAR(39) OR 1=1 #

E la richiesta è passata, restituendomi il seguente risultato: