SQL Injection

Daniele Napolitano

SQL Injection Elements

- **Teoria**
  - ***Che cosa è?***
  - ***Come si verifica?***
  - ***Chi è vulnerabile?***
- **Esempio di attacco**
- **Come difendersi (usando PHP)**
  - ***Filtrare l'input***
  - ***Cosa filtrare***
  - ***Come filtrare***
  - ***Applicazioni pratiche***
- **Conclusioni**
- **Links**

Che cosa è?

- L'SQL Injection è una tecnica di attacco che permette ad un malintenzionato di modificare una query SQL di uno script dall'esterno.

Come si verifica?

- Per effettuare un attacco SQL Injection, su una applicazione web vulnerabile, è necessario immettere una stringa opportunamente costruita in un campo di un form. Questo permette di iniettare (da qui il nome) del codice SQL all'interno di una query usata per gestire i dati dell'utente.

Chi è vulnerabile?

- Sono vulnerabili tutti i siti che utilizzano un database SQL e che non filtrano l'input dell'utente.

Esempio di attacco

- Mettiamo caso che un sito web, utilizzante PHP, presenti un form di login in cui i dati immessi vanno direttamente nella query SQL.

*SELECT * FROM test_table WHERE user = '$var_user' AND pass = '$var_psw';*

- *$var_user* e *$var_psw* sono prese direttamente dalla variabile globale *$_POST[nome_var]* che è un array dei valori ricevuti dal form.

Nota: D'ora in poi segneremo in rosso l'input dall'utente.

Esempio di un attacco

Impostiamo alle variabili le seguenti stringhe:

$var_user = “pippo” e $var_psw = “123”

La query diventerà:

*SELECT * FROM test_table WHERE user = '*pippo*'*
*AND pass = '*123*';*

Quindi restituirà 1 se nella tabella esiste una riga che contiene al campo user la stringa “pippo” e al campo psw la stringa “123” o una riga vuota se non ci sono corrispondenze.

Esempio di un attacco

- Ora invece vediamo cosa succede inserendo una stringa qualsiasi nel campo nome e nel campo psw la seguente stringa: “qualsiasi' OR 1; --”

La query diventerà:

*SELECT * FROM test_table WHERE user = 'qualsiasi' AND pass = 'qualsiasi' OR 1; --';*

Questa query darà sempre 1 come risultato poiché la condizione per essere vera è sempre costante, qualsiasi siano le corrispondenze nella tabella.

Esempio di un attacco

- Ecco quindi che si è scoperto che un qualsiasi utente che abbia accesso ad un form non filtrato, possa immettere del codice SQL. Oltre all'esempio illustrato è possibile concatenare altre query con “;” e ricordandosi di finire la stringa con “--” che commenta il codice seguente.

Esempio:

*SELECT * FROM test_table WHERE user = 'qualsiasi'*
AND pass = 'qualsiasi' OR 1; INSERT INTO test_table ('user', 'pass') VALUES ('io', 'my_pass'); --';

Filtrare l'input

- L'unico rimedio è quello di filtrare l'input dell'utente che andrà nella query SQL.

Cosa filtriamo?

- Abbiamo visto come introducendo dei caratteri speciali come gli apici “ ' “ possiamo modificare la query a nostro piacimento.

Vedremo quindi come filtrarli un maniera opportuna

Come filtriamo?

- In primo luogo se la variabile input è un numero basta fare il type-casting:

$numeric_var = (int) $_POST[“var_input”];

- In questo modo si forza la conversione della variabile var_input a numero intero.

Come filtriamo?

- Esistono anche altre funzioni del PHP per ottenere lo stesso risultato

settype(var,”int”);

intval(var);

- Queste funzioni costringono la variabile ad essere un numero.

Cosa filtriamo?

- Il problema viene quando la varibile deve essere una stringa. Qui bisognerà eliminare gli apici, o meglio, segnalare che sono apici del contenuto informativo della stringa anzichè carattere di controllo che indica la fine della stringa della query.
- Questo è possibile grazie agli slash che definiscono i caratteri speciali.
- Applicando gli slash a questa stringa *“dell'uomo”* diventerà *“dell\'uomo”*. In questa maniere l'apice non viene più considerato come fina stringa bensì come normale carattere.

Come filtrare?

- Vediamo come applicare questo in pratica:

- PHP offre una comodità, la direttiva **magic_quotes_gpc** permette di avere le variabili $_POST, $_GET e $_COOKIE già filtrate in modo tale che in presenza degli apici, troviamo degli slash “\”.

Come filtrare?

- Altra soluzione è usare la funzione addslashes(). Così:

  $var_sicura = addslashes($var);

- Volendo essere ancora più precisi possiamo utilizzare le funzioni specifiche del database usato.
- Per chi usa MySQL ci sono due funzioni: *mysql_escape_string()* e *mysql_real_escape_string()*, quest'ultima tiene conto anche del charset usato dal database.
- PostgreSQL invece ha *pg_escape_string()*.

Applicazione

- Con le tecniche illustrate vediamo di stendere un po' di codice di esempio.
- L'esempio che segue è una funzione adibita al filtraggio delle stringhe che dovranno essere immesse in una query.

Applicazione

```
function filter_string($string)
{
    $string  = htmlspecialchars($string); // filtro anche il codice HTML
    $string  = nl2br($string); // filtro i ritorni a capo con \n\r nel tag html <br>
    if (!get_magic_quotes_gpc()) // controllo che non sia già attivata la direttiva magic_quotes_gpc
    {
        $string  = mysql_escape_string($string); // filtro la stringa con la funzione specifica del db
    }
    return $string; // ritorno la stringa filtrata
}
```

Applicazione

- **Utilizzo della funzione filter_string():**

```
$username = filter_string($_POST[“username”]); //
   applico il filtro

$password = filter_string($_POST[“password”]); // idem

$sql = "SELECT * FROM users WHERE user = '$username'
   AND pass = '$password'”;

$result = mysql_query($sql);


if(mysql_affected_rows($result)>0)

{

// se autenticato esegue il codice qui dentro

}
```

Conclusioni

- Un sito web vulnerabile potrebbe fare danni in tutto il server che lo ospita se abbiamo tutti i diritti sul db. Ci sono molte parole chiave SQL che permetto anche di chiudere il server SQL o cancellare interi db.
- In conclusione bisogna sempre controllare tutto quello che viene dall'esterno del nostro script.

Link utili

- Link utili:

  http://www.php.net/manual/it/index.php

  Manuale PHP dove reperire informazioni su tutte le funzioni illustrate.

  http://it.wikipedia.org/wiki/SQL_injection

  Ottima spiegazione della SQL Injection