Vulnerabilità WordPress SEO by Yoast

Vulnerabilità SEO, ma come risolvere i danni fatti?

Come scritto nel blog ufficiale (LINK) il blasonato plugin SEO by Yoast soffre di una gravissima vulnerabilità: **Blind SQL Injection,** file interessato sarebbe il ***class-bulk-editor-list-table.php***.

Cos'è una **Blind SQL Injection** e come possiamo sfruttarla?

Un hacker inserisce una query SQL non valida in un'applicazione, nel nostro caso **WordPress** che avendo un autore, un admin o un editor già autenticati che visitano un URL malformato, il malintenzionato riesce ad accedere e modificare il database **WordPress.**

Vediamo cosa è successo proprio a noi che scriviamo questo articolo.

Il furbo di turno ha sfruttato la vulnerabilità per modificare il database, creare un nuovo utente, concedergli i privilegi amministrativi ed aggiungere delle widget con codice javascript.

Tale codice servire a modificare i link del sito per rimandare a pubblicità, il modo più veloce per monetizzare. Fortunatamente l'hacker aveva un suo scopo ben preciso, quello di monetizzare, per questo motivo non ha fatto danni.

Questo serve a riflettere su quanti usano WordPress per scopi professionali senza affidarsi ad aziende o professionisti del settore.

Come suggerito da hostingtalk.it:

> In casi come questi, il consiglio è di **aggiornare immediatamente** il plugin WordPress SEO by Yoast all'ultima versione disponibile o di **affidarsi a servizi di hosting gestiti**, che eseguono in automatico per l'utenza gli upgrade di sicurezza necessaria. Altra alternativa è **l'autoaggiornamento di WordPress**, sempre che non sia stato disabilitato.

In alternativa un contratto di manutenzione può salvare il proprio business.

MySQL UDF Perl Regular Expression

Nel realizzare nuovi scraper per g4play.it Emanuele si è reso conto che la nostra istanza MySQL non supporta le espressioni regolari, a lui non servono solo query di ricerca ma manipolazioni di dati complesse. Con estrema semplicità mi chiede di installare la libreria lib_mysqludf_preg, non è complicato, ma neanche così banale. Iniziamo subito con l'installazione dei pacchetti che ci serviranno:

```
[root@mysqllbit lib_mysqludf_preg]# yum install pcre pcre-devel
[root@mysqllbit lib_mysqludf_preg]# yum install make gcc gcc-c++
[root@mysqllbit lib_mysqludf_preg]# yum install mysql-devel
```

Questo per evitare tutti gli errori relativi al compilatore, a pcre e mysql. Scarichiamo il paccheto da GitHub:

```
[root@mysqllbit lib_mysqludf_preg]# wget
https://github.com/mysqludf/lib_mysqludf_preg/archive/testing.zip
[root@mysqllbit lib_mysqludf_preg]# unzip testing.zip
```

ora lanciamo il configuratore

```
[root@mysqllbit lib_mysqludf_preg]# ./configure
```

ci siamo risparmiati gli errori avendo installato preventivamente i pacchetti, l'unico messaggio a video con la parola ERROR è

```
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using
password: NO)
```

Possiamo rilassarci, avendo settato la password di root è normale che non riesca ad accedere. Ora installiamo:

```
[root@mysqllbit lib_mysqludf_preg]# make
[root@mysqllbit lib_mysqludf_preg]# make install
[root@mysqllbit lib_mysqludf_preg]# make installdb

ERROR 1548 (HY000) at line 5: Cannot load from mysql.proc. The table is
probably corrupted
make: *** [uninstalldb] Error 1
```

Sull’ultimo passaggio ho ricevuto errore di tabella corrotta, per questo ho dovuto prima “sistemare” le tabelle MySQL e poi rilanciare il make installdb

```
[root@mysqllbit lib_mysqludf_preg]# make installdb
/usr/bin/mysql -p <./uninstalldb.sql
Enter password:
cat installdb.sql | sed 's/\.so/.dll/g' >installdb_win.sql
if test -f .libs/lib_mysqludf_preg.dll; then  \
                /usr/bin/mysql -p <./installdb_win.sql; \
        else \
                /usr/bin/mysql -p <./installdb.sql;\
        fi
Enter password:
[root@mysqllbit lib_mysqludf_preg]# make test
cd test; make test
make[1]: Entering directory `/usr/local/lib/lib_mysqludf_preg/test'
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-fi...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
make[1]: Leaving directory `/usr/local/lib/lib_mysqludf_preg/test'
```

Finito, ora Emanuele potrà usare le espressioni regolari per manipolare i dati di g4play.it.