

## Forum:[Hack Forums](#)

Thread: [This ain't your standard \\$GET request! TUT on Injection via POST](#)

**Hooded Robin** - May 9, 2011, 3:34 pm

Today I will present my third tutorial show you how to use SQLMAP to perform SQL injections via \$POST with a little help from the Burpsuite. I should point out that this is not all that common of a vulnerability in the wild these days, but it is very good to know and should be tested before you give up all hopes on your target site (and some of this can be used for other types of research and attacks). I would rate this as Intermediate in nature, due to complex nature of orchestrating things to work, but it is not all that hard to do if you give it a few tries. Here we go, try to keep up...

Start off by making sure you have the following requirements: Burpsuite (latest copy - free version is fine)

Download from Homepage: <http://portswigger.net/burp/download.html>

Just download and run burpsuite\_v1.3.03.jar file (version number may change)

Requires JRE Environment to be installed to run, platform independant

SQLMAP (latest copy - Open Source=FREE)

Download from Homepage: <http://sourceforge.net/projects/sqlmap/files/sqlmap/>

Requires Python to be installed, platform independant although not all features capable on Windows platform

Site with \$POST data being submitted for us to find and inject (i.e. Login page or search fields)

Example Google Dork: "inurl:admin/login.php" get creative and get better results

Now ow it is time to find a site where we can start doing some testing, let us use Google to see what we can find using a simple dork:

Code ext:php inurl:admin/login site:US NOTE: There are variations to this, so use your own creativity to increase your odds

Resulting Target Found: <http://www.site.com/Admin/login.php> (take note, write it down, and close browser)

Now it is time to use Burpsuite to help us find out the needed fields for injecting. Run the "burpsuite\_v1.x.jar" file to get up and running, just double click it OR start it from the command line by using Code java -jar burpsuite\_v1.x.jar (replace "x" with your version number). Bursuite is configured by default to operate on port 8080, so you will need to configure your favorite browser to use this as the proxy address so that Brupsuite can pick up all of the traffic for analysis and examining. For most browsers you can open and then go into options or internet options>>connections>>LAN Settings>>PROXY Settings>>and then set to: 127.0.0.1:8080 >>save>>and your done.

Now any pages you go to in your web browser will be parsed by Burpsuite, simply remove proxy setting to go back to normal surfing. By default Burpsuite has the option to "Intercept Data" set to "ON". This is helpful if you want to tamper with data for EVERY request traveling on the line. You can quickly and easily turn this off if you don't want as much interaction as it will still save the logs for requests sent and received to the HISTORY tab. I prefer to leave it ON as I see more of what is actually going on and can simply DROP any I don't want or FORWARD to allow without very much hassle. Leave it setup for now, we will come back to this in a few...

NOTE: you will have to do some back and forth to see the results while we test so I suggest you just minimize or adjust your screens so you can run side by side (my preference) to save your self time.

Now that we have a potential target and Burpsuite running we will start testing it with our browser to see what all is captured. Surf to the login page found with our Google Dorks earlier and watch as the Burpsuite starts to blink for user interaction. Assuming you have default behavior you will need to "FORWARD" or "DROP" the requests as they are sent and received. This equates to allow or drop in simple terms. You can review the actual details of what is being sent in the RAW tab window (you change tabs to view different aspects of what is being sent/received). Let us try entering "logintest" in the login field and "passtest" in the password field, then submit. Obviously this is not going to get us into the site, unless you are just a really lucky guesser, BUT it will help us to get one step closer to accomplishing that task. You will see the request come through Burpsuite again, FORWARD the request to allow it to run through the full authentication cycle. Now let's review the results from that simple request we just sent...

In Burpsuite navigate to the HISTORY tab, two to the right of the main INTERCEPTOR tab we have been working from since we started. The HISTORY tab will outline all of the details that were captured when we sent the login request to our target site.

You should see one or two \$GET requests which were sent from you when you were actually navigating to the target site, and then you should find a \$POST request from when we actually submitted the login page/form. This is the one we are actually interested in. Highlight the \$POST request in the history tab and you can view the specifics down below in the RAW tab window. This is a view of the \$POST request sent, if you look through it (likely near bottom) you will notice the actual post parameters sent for the form fields we entered (testlogin & passtest).

It should look something like this:

```
Code POST /Admin/login.php HTTP/1.1
Host: http://www.site.com/
Proxy-Connection: keep-alive
Referer: http://www.site.com/Admin/login.php
Cache-Control: max-age=0
Origin: http://www.ostergottesdienste.de/
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.24 (KHTML, like Gecko)
Chrome/11.0.696.65 Safari/534.24
Content-Type: application/x-www-form-urlencoded
Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: sid=a3b362a522444335b693ab04007ebbed
Content-Length: 133
_auth_username=testlogin&_auth_password=testpass&Submit_button=Submit
Notice the "_auth_username=testlogin"&"_auth_password=testpass". These are the $POST fields
```

we just sent, these field names may vary from site to site so please stay with me if yours a little different.

Note the cookie field to see if this is of any value, it and user-agent fields also injectable (to be covered in next TUT)

PRO TIP\*: If you use tools like Cain and Able you can use this method to add to the "HTTP\_PASS\_FIELDS.LST"&"HTTP\_USER\_FIELDS.LST" files to increase your chances of catching credentials while pulling off MiTM attacks as you can add the form fields it looks for based on your findings in Burpsuite (coughFacebookGmailMyspaceetc).

OK now that we know our \$POST variables we will be working with, it is time to startup SQLMAP. Open command prompt and type in 'sqlmap.py', see previous posts regarding how to set environment variables to avoid having to navigate to the install folder. Now that SQLMAP is up we will start feeding it the data we found from our previous steps above (\_auth\_username=&\_auth\_password=&Submit\_button=). We will need to let identify we want to use \$POST method to inject our website by issuing the '--data' argument followed by our \$POST fields we found using Burpsuite "\_auth\_username=" & "\_auth\_password=".

NOTE: \$GET requests are sent by SQLMAP as default but by issuing the '--data' argument it is smart enough to know to change to \$POST - realized by you supplying the those \$POST parameters with the '--data' argument in command. The fields passed are then used to test for SQL injection as well as any

provided GET parameters that might normally be tested for.

Code sqlmap.py -u http://www.site.com/Admin/login.php --data

"\_auth\_username=test&\_auth\_password=test\$Submit\_button=Submit" \*PRO TIP If you know or only want to test certain fields then only include the ones you want (i.e. \_auth\_password only). If you started a scan and it doesnt look like it is an injectable field you can hit 'CTRL+C' to pause and then you can hit 'N+ENTER' to skip to the next field to start testing. This can save you from spending unnecessary time.

Once confirmed that you have found an injection point, we can add arguments we learned from our previous introductions to find crucial information. We transform the above command into the following:

Code sqlmap.py -u http://www.site.com/Admin/login.php --data "\_auth\_password=test"

--current-user --current-dba --users --dbs --is-dbaOnce we have processed results we will check for passwords, using the '--passwords' and see if we can get any...

Code sqlmap.py -u http://www.site.com/Admin/login.php --data "\_auth\_password=test" --passwords

NOTE: this is assuming you have gone in order, as it will read the session file already created to skip things like (--users) which normally accompanies this command argument...

You can continue enumerating the database for more details on tables, columns, etc but that was already covered in my two original posts on SQLMAP so I will not go into any more details on how to dump the database. If you are super lazy and this was too complex for you, then you can use the easy way out as covered in my mini-tutorial at the end of SQLMAP Tips & Tricks article (--forms).

I hope you have enjoyed this little twist to your standard injection techniques, and hope you walk away remembering to leave no rock unturned in your search for vulnerabilities and injection points. I will try to include more in depth coverage on Burpsuite and its other functionalities and how else it can be used in future articles, but I encourage you in the meantime to please dig around in it and try to get creative. Enjoy!

Later...  
H.R.

**hich** - May 9, 2011, 3:55 pm

Nice tut bro, keep posting as long as you can ;)

**Hooded Robin** - May 9, 2011, 4:26 pm

thanks mate! I have one more to cover use with Metasploit and true full database takeover and then I will be done posting about sqlmap...

**nch.sniper** - July 14, 2011, 5:34 pm

Nice tutorial Bro i am working on it but i have a little problem am stuck some where

**Gnohter** - July 14, 2011, 6:53 pm

You can also do it by hand using Firefox's addon Live HTTP Headers.

**Suit®** - July 14, 2011, 7:16 pm

Word nice one mate. I stumbled upon burpsuite and had no idea how to use it. Cheers. I will defs be integrating this into my SQL arsenal

**Hooded Robin** - July 14, 2011, 9:39 pm

Thanks and glad you liked it!

**PlickaMaterna** - July 15, 2011, 5:15 am

OR simply use HackBar and send POST requests easily.

**Hooded Robin** - July 15, 2011, 6:03 am

sure, but you can not automate the HackBar, LiveHTTP Headers, or Tamper Data which is part of what makes it so the Burpsuite so valuable. It also has the ability to help identify common weaknesses in cookies fields as well as referrer fields which I know none of those tools do either. This was just a basic intro to show how it can be used to get started, as this is 100 times more advanced than most realize...

**Nuevo** - February 22, 2017, 6:08 pm

Thanks for you tutorial :)

**Kashmiri Wolf** - February 23, 2017, 12:56 am

Thanks For Posting ?