

Forum:[Hack Forums](#)

Thread: [\[TUT\] BURP SUITE - PART II: SQL Authentication Bypass \[TUT\]](#)

Hooded Robin - November 30, 2011, 4:46 pm

Today I am going to extend my previous tutorial where I introduced you to [Burp Suite tools](#) so that we can now audit login forms for SQL Authentication Bypass vulnerabilities. This is a common problem found daily in the wild where PHP & MySQL are being used as the Authorization mechanism based on dynamic SQL Query's which are completed from user input supplied through login form. This can be time consuming and repetitive to conduct manually but it is relatively easy to audit - and after reading this it will be even easier, especially with Burp! I will first provide an explanation of the problem and what is going on which allows the vulnerability to be exploited followed by an example, so here goes...

As mentioned above Authentication Bypass vulnerabilities often occurs due to a lack of filtering of user supplied input. If we review a quick example of code from a PHP/MySQL authentication page we will begin to see things more clearly. You can find some test samples with quick Google dork "file:php/asp inurl:admin/login", but here is sample login to keep it easy:

```
Code <?php
$sql = "SELECT * FROM users WHERE username=" . $_POST['username'] . " AND password=" .
$_POST['password'] . """;
response = mysql_query($sql);
?>
```

This code fails to filter or check the user supplied input. The target system reads like so as result:

Code SELECT * FROM users WHERE user=" AND password="

Since this is not filtered or checked we will pick a username and use SQL Injection on the secondary field. Let us assume we choose the username "webadmin" and the SQL Injection: ' OR 'x' = 'x'

This is now how the query looks that will get passed through:

Code SELECT * FROM users WHERE user='webadmin' AND password=" OR 'x' = 'x'

This is how the target system actually reads & parses the request:

Code SELECT * FROM users WHERE user='webadmin' AND TRUE

The ultimate goal here is to use our SQL injection to alter the SQL query so that it is fooled into letting us in. This works as you can see above as 'x' is always going to be equal to 'x' so it will always return as TRUE, and thus the system will grant us webadmin user access to walk right in through the front door without any real password.

Now we could use alternative injections above, like: ' or '1'='1

```
' or 'x'='x
') or ('x'='x
') or ('1'='1
...etc
```

Some customization may be needed to fit your specific need (Sometimes injections need to be placed in both fields; sometimes it needs more complex syntax, HEX or Char encoding may be needed, etc). You could easily build your own list, but I have gone ahead and put together a decent list to help get you started. The list can be found in the "AuthBypass/auth-bypass.txt" file which is included in my full download here: [HR's Burp Starter Pack](#).

Now with Burp Suite:

Now rather than try each potential SQL statement or injection we place all of our potential injections into a single file and then let Burp run the requests in an automated fashion. This will help to save you time, as well as allow easier parsing of the results in a systematic way. In order to do this we simply submit a bogus request with arbitrary data submitted (user: foo, pass: bar).

[img]http://img812.imageshack.us/img812/1753/62672264.png[/img]

[img]http://img713.imageshack.us/img713/6742/28054744.png[/img]

[img]http://img221.imageshack.us/img221/9762/21137854.png[/img]

We then load up the request and send it to the Intruder tool to build our attack. We clear the default injection points and specifically set them around our password field or both user and password fields. You will choose the Sniper or Battering Ram attacks for this method, depending on how you want to test the form. You will then need to set the payload to run your auth-bypass.txt file. You can add some additional inspection items to the grep field if you want. I find it is often helpful to add some basic text you might see upon failed or successful login attempts (welcome, success, failed, wrong, etc).

[img]http://img202.imageshack.us/img202/2039/16933739.png[/img]

Once you run the Intruder tool it is time to sit back and interpret the results. You will need to check time, length, grep, etc and manually review any of the requests that appear to show a noticeable variance in responses.

[img]http://img338.imageshack.us/img338/4830/45015881.png[/img]

Once you find one, re-test it manually in the browser to see if it works.

[img]http://img43.imageshack.us/img43/5240/89323078.png[/img]

If it works you are in, if not keep analyzing and inspecting further.

[img]http://img408.imageshack.us/img408/1195/28104046.png[/img]

Rinse, wash, and repeat as necessary ?

BONUS-VIDEO:

Here is a quick video to show how things should work for those of you that needed the real time visual aid (I know I usually do):

[Video: <http://www.youtube.com/watch?v=2ub55R6p7FY>]

I hope you have enjoyed another tutorial on the Burp Suite and how it can be used to perform quick and accurate audits of login forms to see if they are vulnerable to Authentication Bypass via SQL Injection. I plan to have a follow up tutorial stemming from this one which adapts the methods slightly to perform dictionary attacks against login forms. Please check back often and soon for more updates and more tutorials to come. As always and until next time, Enjoy!

Shouts to Zer0Lulz crew!

blackperuvian - November 30, 2011, 6:09 pm

Cool for learners but I have not lagy method

But thanks for contribution

Limboland™ - December 14, 2011, 1:38 pm

men i love your tutorial's ...you are not the best on HF but second place on HF .

I 'm a big fan of your's .

?????

I love burp ???

Hooded Robin - December 14, 2011, 1:41 pm

I still feel I have a LONG way to go, BUT thanks for the positive feedback - it is much appreciated!

PS - Burp is the best framework ever!

BeggFoMercy - December 14, 2011, 2:13 pm

Nice TUT bro keep up good working ..!

VipVince - December 14, 2011, 3:25 pm

Awesome tutorial man. Was a good read. I may start using burp again.

Zhen. - December 17, 2011, 11:40 am

Been using Burp ever since you started a tutorial about it and now I'm addicted to using it lol

Thanks for the tutorial bro, looking forward to more Burp tutorials of yours

Keep up the good work ?

Hooded Robin - December 17, 2011, 12:00 pm

Thanks for the feedback guys, it is always appreciated! I do plan to do some more Burp tutorials just working on a few others lately, but trust me there will be many more to come...If anyone has a request for one just shoot me a PM and I will see what I can do :)

Que - January 13, 2012, 8:59 am

hey nice tutorial! ^^ haven't even thought about using sql injection like this but it is pretty nice :) i will check out the other tutorials too ^^

Hooded Robin - January 13, 2012, 11:31 am

You can also use this method with a few minor tweaks to use Burp as a Authentication Bruteforcer tool, but I think this may be another tut another time...Glad you enjoyed it!

th3.g4m3_0v3r - January 13, 2012, 11:49 am

i like Your Tutorial

You remember Me Bro ???? :P

And You have gud Knowledge in web hacking...

Can I add You

Thanks

honey3 - February 8, 2012, 7:10 am

hi bro i cant download the link please give another link

Hooded Robin - February 8, 2012, 9:21 am

Link updated, but here you go anyways: http://uppit.com/6fj3c4vxi4xk/HR-BURP-PACK_1.23.12.rar

stock1337 - March 7, 2012, 10:57 pm

Is there a way to do a binary tree search in burp's intruder ? I google a bit but didn't find a way to do it on burp.

For the 'normal' blind sqli, sometime we can just do boolean test on `substring(user(),1,1)> 97` and keep guessing till the right one for a character. Anyway to do that on binary search on burp instead of test all asciii character?

Hooded Robin - March 8, 2012, 7:54 am

@stock1337 - i dont exactly follow what your asking when you say "binary search". Can you please provide some examples of manual request for this? If you are talking about making forked decisions then you cant do that with Burp unless you write your own python plugin for it to use. It would most likely take a combination of intruder tool and repeater tool with user interaction at set points to make certain modifications to allow the forking to occur. If you can code this then you dont need to use Burp for it so your probably better off just writing your own custom python, perl, or Php script to do that, sorry. If I am way off then provide some examples of what you mean and I will work with you to try and figure something out.

stock1337 - March 8, 2012, 10:00 pm

Pardon my poor english :S Below is a example I'd like to batch process:

```
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<200 -- True.  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<100 -- False  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>100 -- True  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>150 -- false  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<150 -- True  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>125 -- False  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<125 -- True  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>112 -- False  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<112 --True  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>106 --False  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<106 --True  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>103 --True  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>105 -- False  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<105 -- True  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))>104 -- False  
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<104 -- False  
http://site/join.php?id=6//and//ascii(substring(user(),1,1)) =104 --True ==>104 = h
```

For the normal blind sqli, one can use the above boolean test to manually figure out a character's decimal value is 104 (thus the character is 'h'). I am looking for a way to batch process those tests in BURP so I don't have to manually perform it and repeat the same for multiple parameters throughout the entire site. If I undertand you correctly, burp don't seems have that features yet....

My programming knowledge only goes as far as basic bash script....Is there a good tutorial for python showing me how to achieve what I need(as comprehensive as yours preferably ^_^)?

Hooded Robin - March 8, 2012, 10:43 pm

Download my pack or make your own list with all the numbers of the ascii chart, we will use this as our payload in a minute (one per line to work right). Next we open our link in our browser to capture it in the proxy tab of Burp. Next we sent it to the repeater tab. On the repeater tab we frame our

BLIND injection by starting with our first request:

```
http://site/join.php?id=6//and//ascii(substring(user(),1,1))<200 --
```

Now to make Burp useful we skip the entire TRUE/FALSE test and simple brute our way to TRUE. We send the request to the INTRUDER tab now, where we change it as follows:

```
http://site/join.php?id=6//and//ascii(substring(user(),0,1))=$200$--
```

We mark flip to "=" comparison and then we mark the char value as our injection point (clear any others that may be loaded). Next you need to choose your payload, choose Sniper and runtime file. Point it to the file we made earlier which is full of numbers (from ascii chart). Now you can add text from the TRUE page results to the regex check if you like but it is not really necessary for this process. Run INTRUDER tool when setup. Now simply interpret the results each time until you find the TRUE request. You can use the response size in 99% of the cases to judge whether your request came back TRUE or FALSE. Once you find the TRUE request, mark that char down.

Go back to the INTRUDER tool beginning, move the LIMIT statement in injection up position to get the next character position from our target table/column/password and repeat the process again, adjusting injection syntax with repeater tool as needed and then sending back to intruder again.

It is deffinatly faster than doing it by hand and it send about the same amount of requests if you keep an eye on it. You could play with your payload list to come up with an optimized way to sort it but pure bruteforce in logical systematic order does the trick for me. This method also works really well for dumping large databases when you have an error based injection (same principal, just frame it and enumerate the LIMIT or CHAR positions).

Hope this helps clarify things up a bit if we need to discuss further please send my way via PM to get this thread back on track.

To everyone else hope this has not taken away from the thread at hand which is focused on using SQL Injection techniques to bypass weak authentication schemes typically found at login panels. If anyone else has questions please feel free to post away, but please keep on thread topic or shoot me a PM.

FMtheHackerVenkat - May 3, 2012, 6:01 am

thank u for poviding tutorial ..???:

Jesse Pinkman™ - May 3, 2012, 7:41 am

HQ as always, keep up the good work mate!

??¢??? - May 3, 2012, 7:53 am

HR i love you . Awesome work :) bro

Th3PonyWizard - May 3, 2012, 7:56 am

(May 3, 2012 06:01 AM)FMtheHackerVenkat Wrote: [\[CITE\]](#) thank u for poviding tutorial ..???:And (no) thanks for gravedigging tutorials. Meh.

A Pretty nice tutorial, but sadly I can't use it, being on Linux. (Is there any linux version of that? :\)

Hooded Robin - May 3, 2012, 10:07 am

Burp Suite is Java based and therefore is platform independent. If you don't like Burp Suite everything works with the new OWASP Zed Attack Proxy (a.k.a. ZAP) as well. I use both on Linux regularly :)

PS - thanks for the comments, I don't mind you gravedigging any of my tutorials. They need since

they get buried by some much BS these days :(

Zixem - May 13, 2012, 10:34 am

Awsome TUT .. Nice work TNX [= ?

MISTUGAN - August 24, 2012, 5:05 pm

Oh! another great tutorials...
thanks for sharing...

hrga787 - August 24, 2012, 11:02 pm

Nice post. thanks for this.. very usefull

FinalDespair - December 1, 2012, 1:35 am

Thanks for the tutorial. just what i wanted.

Avinash Thapa - April 30, 2014, 8:32 am

one of the nice tutorial for burp...
Keep it up man..

ShadowRi5ing - April 30, 2014, 4:29 pm

Good tutorial, but this is not an SQL authentication bypass, it's just a very classic example of an SQLi.

http://www.cvedetails.com/cve-details.php?t=1&cve_id=2012-2122

Hooded Robin - May 8, 2014, 6:41 am

This is authentication bypassing via SQLi which is a common occurrence in the wild and very easy to test with Burp which is the main point here to this thread. The actual MySQL Authentication Bypass is another vulnerability as you have referenced in your link. I wrote something for that which is included in my creep3r tool already which you can find here:

<https://github.com/Hood3dRob1n/creep3r/blob/master/classes/dbtools.rb> (see lines 42 - 90, for specific code)

Mr.Plasma - May 8, 2014, 10:59 am

Nice tutorial its i have enjoyed it, It helped me btw Alot :D

hacker625 - March 4, 2015, 7:20 am

Yes.Now i am working with burp suit. I need to learn more just only burp tutorials :D

idwarner - March 4, 2015, 7:30 am

Verry nice tutorial man !
Nice share .

Nuevo - January 29, 2017, 6:43 pm

Thanks bro for tutorial. Nice share :D

Nuevo - February 14, 2017, 8:59 pm

Thank you so much for you tutorial :)

mr.vulnerable - April 18, 2017, 1:05 am

nice tutorial bro keep posting like this im new here