# Blind SQL injection –WAF HTB challenge

## 

1. What the WAF function does is check the given SQL statement from the user for any of the listed characters or phrases and will ignore the statement if it is true.
2. If none are found, then the statement is passed through
3. Takes POST request of raw bodied json data (need to send post request in json format {“xxx”:”xxx”})
4. The public function takes an SQL statement ($SQL) and our statement ($args) and passes through our argument into %S.
5. We need to include some illegal characters in the argument through encoding.
6. if our argument bypasses the WAF function then it is put into ‘json\_decode’ and passes it into the SQL statement (to search the notes file for an assignee but we want to escape out of this).
7. Uses vsprintf to replace the %s with our argument
8. We can send Unicode characters in our post request in the json format. These Unicode characters will be sent into json\_decocde which will be decoded to their ascii format and read as normal.
9. This means we can put the Unicode equivalent of an apostrophe (‘) to escape the SQL statement and continue writing our own command to return data
10. However, this is a blind SQL injection, so we need to delay the response from the server if our statement is true. If the statement is false, the response will not be delayed which gives us Boolean information on the data.



* This is the data in the POST request. It sends a POST request for a user admin and then encodes the statement “ 'WAITFOR DELAY '0:0:5'--# ”
* Which bypasses the WAF function except the “in” inside “admin”.
* We need to use SQLmap to automate the SQL injection.
  + We need to create a custom tamper script that will take our payload and convert it into its Unicode equivalent



What this script does is loops through the payload SQLmap will generate and turn it into its Unicode equivalent.

We can now tell SQLmap to perform a POST request (using -u) at the target address (<http://docker.hackthebox.eu:31324/>). Because we are sending over json data we need to tell sqlmap how to send the data by specifying the –data parameter with a json format e.g.

python2 /usr/bin/sqlmap -u 'http://docker.hackthebox.eu:31324' --data '{"user":"\*"}'

We already know from the source code and from our tests earlier this is how the application receives its data. The best feature from SQLmap is being able to add the “\*” to specify where the program should Inject the payload.

In addition, to this we can set two parameters (level and risk) to specify how much detail we want the injection to be. Level 5 and risk 3 are the highest priority. (this is optional)

To specify the tamper script we want to use we use the –tamper parameter like so

python2 /usr/bin/sqlmap -u 'http://docker.hackthebox.eu:31324' --data '{"user":"\*"}' —level=5 —risk=3 --tamper="WAFBypass.py"

Because we are carrying out a blind SQL injection this needs to be specified with the –technique parameter (we used BUTS to specific Blind, Union, Time-based and )

python2 /usr/bin/sqlmap -u 'http://docker.hackthebox.eu:31324' --data '{"user":"\*"}' --tamper="WAFBypass.py" --technique BUTS –dump

The -dump parameter dumps the information it finds from the injections on screen

A screenshot of text

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Debugging:**

A close up of text on a black surface

Description automatically generatedIn the tamper script we can print the payload to see what is being sent to be encoded

A picture containing standing, water, man, white

Description automatically generatedOr we can print the retVal to see the Unicode payload being sent to the application.