ssooking

征途漫漫、步履铿锵，无畏、无惧、无阻！

CTF线下防御战 — 让你的靶机变成“铜墙铁壁”

本文首发安全客，未经允许禁止转载。 原文链接

一. 前言

随着CTF的普及，比赛的形式也有了越来越多的花样，对于线下赛来说，开始出现了安全加固或者防御战之类的环节，亦或者因为拿下靶机后不希望其他攻击者进入而进行“争夺”，无论什么形式，这些都需要我们对于服务器的防护工作有所了解。对于线下赛，笔者虽说没有什么很高超的攻防技巧，但也是有着一些自己的心得。本文总结了一些CTF线下赛中常用的服务器加固姿势，希望能对各位CTF朋友们有所帮助。环境针对目前常见线下赛常见的linux Web服务器，但是因为CTF毕竟与真实网络环境有很大区别，本文的涉及的大部分姿势不具有普遍适用性。本文涉及到的相关代码github下载地址： CTFDefense 。

二. 常用姿势

1. 提权

在开始正文之前，需要先提一下提权，这个要根据自己的比赛过程中的需要而定。有些比赛就有专门的防御加固环节，但安全加固的很多操作都会涉及到root权限，如果直接给root权限最好，但一般一开始会给一个普通权限账号，或者干脆什么都不给，需要我们自己通过漏洞拿下服务器，这样往往就需要提权了。关于提权，通常我们要根据kernel版本号找到对应的poc，平时我们可以收集测试一些比较新的提权poc，以备不时之需。这里有一个网站： http://exploit.linuxnote.org/ ，里面有许多linux本地提权的poc。github上有一个挺全的提权exp项目： https://github.com/SecWiki/linux-kernel-exploits 。网上也有人分享的一些打包搜集的poc,比如 这个 ，有兴趣的朋友可以多下载看看。

下面分享几个最近两年并且影响范围比较大的：

CVE-2017-6074（DCCP双重释放漏洞 < 2.6.18 ）
描述：DCCP双重释放漏洞可允许本地低权限用户修改Linux内核内存，导致拒绝服务（系统崩溃）或者提升权限，获得系统的管理访问权限
用法：./pwn

CVE-2016-5195（脏牛，kernel 2.6.22 < 3.9 (x86/x64)）
描述：低权限用户可修改root用户创建的文件内容，如修改 /etc/passwd，把当前用户的 uid 改成 0 即可提升为root权限
用法： ./dirtyc0w file content

CVE-2016-8655（Ubuntu 12.04、14.04，Debian 7、8）
描述：条件竞争漏洞，可以让低权限的进程获得内核代码执行权限
用法：./chocobo_root
POC： https://www.seebug.org/vuldb/ssvid-92567

CVE-2017-1000367（sudo本地提权漏洞 ）
CVE-2017-1000364

Page： ssooking.ml
Email：
ssooking@qq.com
Github：
github.com/ssooking

昵称： ssooking
园龄： 1年8个月
粉丝： 35
关注： 2
+加关注

我的标签

Webshell (11)
XSS (10)
木马后门 (9)
python (9)
内网渗透 (9)
Web请求响应 (8)
程序代码 (7)
Shell (6)
Trojan (6)
热门分享 (6)
更多

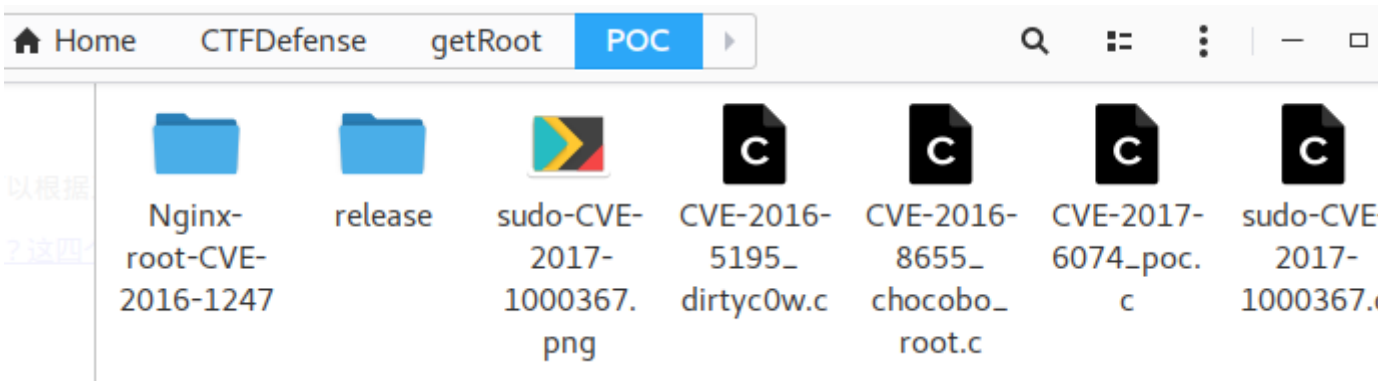
描述：Linux Kernel Stack Clash安全漏洞。该漏洞是由于操作系统内存管理中的一个堆栈冲突漏洞，它影响Linux，FreeBSD和OpenBSD，NetBSD，Solaris，i386和AMD64，攻击者可以利用它破坏内存并执行任意代码 。

CVE-2016-1247（Nginx权限提升漏洞）

描述：Nginx服务在创建log目录时使用了不安全的权限设置，可造成本地权限提升，恶意攻击者能够借此实现从 nginx/web 的用户权限 www-data 到 root 用户权限的提升。

POC： https://legalhackers.com/advisories/Nginx-Exploit-Deb-Root-PrivEsc-CVE-2016-1247.html

提权相关代码在GetRoot目录，POC中是上面提到的几个本地提权源代码，release中分别是编译好的32位和64位程序。



实用脚本

Linux_Exploit_Suggester.pl ，它可以根据系统内核版本号返回一个包含了可能exploits的列表。还有一个检查linux安全状况的脚本： 原文链接

还有几个详见： Linux提权？这四个脚本可以帮助你

2. 常用操作命令

linux操作有很多命令，但是线下赛的防护工作中常用的也就那么一些，我们平时可以留意并总结起来，便于我们比赛使用。

ssh操作

```
ssh <-p 端口> 用户名@IP    //登录
scp 文件路径   用户名@IP:存放路径    //向ssh服务器上传输文件
```

备份web目录

```
tar -zcvf web.tar.gz /var/www/html/
```

用户管理

```
w    //查看当前用户
pkill -kill -t <用户tty>    //踢掉当前登录用户
```

进程管理

```
ps aux | grep pid或者进程名    //查看进程信息
```

查看已建立的网络连接及进程

```
netstat -antulp | grep EST
```

查看指定端口被哪个进程占用

```
lsof -i:端口号 或者 netstat -tunlp|grep 端口号
```

结束进程命令

```
kill PID
killall <进程名>
kill -9 <PID>
```

iptables命令

封杀某个IP或者ip段，如：123.4.5.6

```
iptables -I INPUT -s 123.4.5.6 -j DROP
iptables -I INPUT -s 123.4.5.1/24 -j DROP
```

禁止从某个主机ssh远程访问登陆到本机，如123.4.5.6

```
iptable -t filter -A INPUT -s 123.4.5.6 -p tcp --dport 22 -j DROP
```

随笔分类(163)

CTF(7)
Linux(18)
安全基础(8)
程序代码(19)
大学课程(10)
攻防渗透(20)
技术随笔(6)
竞赛游戏(3)
漏洞分析(2)
木马后门(7)
前端安全(6)
前沿关注(3)
神兵利器(24)
思维视角(3)
无线技术(7)
系统运维(16)
硬件相关(4)

随笔档案(130)

2018年4月 (1)
2018年3月 (3)
2018年1月 (3)
2017年12月 (1)
2017年11月 (2)
2017年10月 (3)
2017年9月 (4)
2017年8月 (1)
2017年7月 (2)
2017年6月 (1)
2017年5月 (2)
2017年4月 (1)
2017年3月 (7)
2017年2月 (8)
2017年1月 (7)
2016年12月 (13)
2016年11月 (37)
2016年10月 (12)
2016年9月 (18)
2016年8月 (4)

文章分类(90)

Trojan(13)
安全时事(4)
二进制(6)
漏洞聚焦(2)
奇门技巧(11)

Mysql数据库操作

备份mysql数据库

```
mysqldump -u 用户名 -p 密码 数据库名 > back.sql    //备份指定数据库
mysqldump --all-databases > bak.sql       //备份所有数据库
```

还原mysql数据库

```
mysql -u 用户名 -p 密码 数据库名 < bak.sql
```

安全检查

```
find / *.php -perm 4777      //查找777的权限的php文件
awk -F: '{if($3==0)print $1}' /etc/passwd    //查看root权限的账号
crontab -l    //查看计划任务
```

检测所有的tcp连接数量及状态

```
 netstat -ant|awk '{print $5 "\t" $6}' |grep "[1-9][0-9]*\."|sed -e 's/::ffff://' -e 's/:[0-9]*//'|sort|uniq -c|sort -rn
```

查看页面访问排名前十的IP

```
cat /var/log/apache2/access.log | cut -f1 -d " " | sort | uniq -c | sort -k 1 -r | head -10
```

查看页面访问排名前十的URL

```
cat /var/log/apache2/access.log | cut -f4 -d " " | sort | uniq -c | sort -k 1 -r | head -10
```

再推荐两篇篇安全应急排查手册： 应急排查手册 ， Linux应急响应姿势浅谈

3. 文件监控防webshell

防御webshell，我们可以监控我们的web目录，对文件的增加或修改等操作进行限制等，粗暴一点的话，就禁止任何文件产生变化，杜绝被传webshell的可能性。

（1）使用系统 chattr +i 命令

linux下的文件有着隐藏属性，可以用lsattr命令查看。其中有一个i属性，表示不得更动任意文件或目录。如果你已经有root或者sudo权限了，那么你可以使用"chattr +i 命令"修改文件隐藏属性，这样所有用户都不能对该文件或者目录进行修改删除等操作（包括root），如果想进行修改，必须用命令"chattr -i"取消隐藏属性。

Linux文件保护禁止修改、删除、移动文件等,使用chattr +i保护

```
>> :/var/www/html# touch 1.txt
>> :/var/www/html# ls
1.txt
>> :/var/www/html# chattr +i 1.txt
>> :/var/www/html# echo "Hello" >> 1.txt
-bash: 1.txt: Permission denied
>> :/var/www/html# _
```

例子：

用chattr命令防止系统中某个关键文件被修改：

```
chattr +i /etc/profile
```

将/var/www/html目录下的文件设置为不允许任何人修改：

```
chattr -R +i /var/www/html
```

```
>> :/var/www/html# chattr -R +i /var/www/html/aaa/
>> :/var/www/html# lsattr aaa/
----i----------- aaa/aab
----i----------- aaa/2.txt
>> :/var/www/html# _
```

（2）自己动手丰衣足食

python的第三方库pyinotify可以让我们很方便地实现这些功能。但是由于是第三方库，线下赛中通常没法联网安装库，所以我们可以手工把库文件传到靶机里python库中：/usr/lib/pythonXXX/site-packages，但是更方便的做法是借用pyinstaller等工具将其打包成linux可执行文件。

安装了pyinotify库之后，我们仅仅运行在机器上： "python -m pyinotify 监控目录路径" 这条简单的命令，就可以看到对这个目录以及该目录下所有进行任何操作的的监控日志。

```
→  ~ python -m pyinotify /var/www/html
<Event dir=False mask=0x20 maskname=IN_OPEN name=index.html path=/va
/var/www/html/index.html wd=1 >
<Event dir=False mask=0x10 maskname=IN_CLOSE_NOWRITE name=index.html
pathname=/var/www/html/index.html wd=1 >
<Event dir=True mask=0x40000020 maskname=IN_OPEN|IN_ISDIR name=shell
pathname=/var/www/html/shell wd=1 >
<Event dir=True mask=0x40000001 maskname=IN_ACCESS|IN_ISDIR name=she
l pathname=/var/www/html/shell wd=1 >
<Event dir=True mask=0x40000010 maskname=IN_CLOSE_NOWRITE|IN_ISDIR r
www/html pathname=/var/www/html/shell wd=1 >
<Event dir=True mask=0x40000020 maskname=IN_OPEN|IN_ISDIR name=shell
pathname=/var/www/html/shell wd=1 >
<Event dir=True mask=0x40000001 maskname=IN_ACCESS|IN_ISDIR name=she
l pathname=/var/www/html/shell wd=1 >
<Event dir=True mask=0x40000010 maskname=IN_CLOSE_NOWRITE|IN_ISDIR r
www/html pathname=/var/www/html/shell wd=1 >
```

但由于监控事件太过杂，很多并不是我们关注的，并且我们不仅仅要监控，还需要对某些操作进行自动处理，因此我们可以自己编程，针对性地实现我们需要的功能，下面是一段代码示例。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# ** Author: ssooking
import os
import argparse
from pyinotify import WatchManager, Notifier,ProcessEvent
from pyinotify import IN_DELETE, IN_CREATE,IN_MOVED_TO,IN_ATTRIB

class EventHandler(ProcessEvent):
        """事件处理"""
        #创建
        def process_IN_CREATE(self, event):
            print "[!] Create : " + event.pathname
            DeleteFileOrDir(event.pathname)

        #删除
        def process_IN_DELETE(self, event):
            print "[!] Delete : " + event.pathname

        #文件属性被修改，如chmod、chown命令
        def process_IN_ATTRIB(self, event):
            print "[!] Attribute been modified:" + event.pathname

        #文件被移来，如mv、cp命令
        def process_IN_MOVED_TO(self, event):
            print "[!] File or dir been moved to here: " + event.pathname
            DeleteFileOrDir(event.pathname)

def DeleteFileOrDir(target):
    if os.path.isdir(target):
        fileslist = os.listdir(target)
        for files in fileslist:
            DeleteFileOrDir(target + "/" + files)
        try:
            os.rmdir(target)
            print "     >>> Delete directory successfully: " + target
        except:
```

```
            print "     [-] Delete directory failed: " + target

    if os.path.isfile(target):
        try:
            os.remove(target)
            print "     >>> Delete file successfully" + target
        except:
            print "     [-] Delete file filed:  " + target


def Monitor(path):
        wm = WatchManager()
        mask = IN_DELETE | IN_CREATE | IN_MOVED_TO | IN_ATTRIB
        notifier = Notifier(wm, EventHandler())
        wm.add_watch(path, mask,rec=True)
        print '[+] Now Starting Monitor:  %s'%(path)
        while True:
                try:
                        notifier.process_events()
                        if notifier.check_events():
                                notifier.read_events()
                except KeyboardInterrupt:
                        notifier.stop()
                        break

if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        usage="%(prog)s -w [path]",
        description='''
            Introduce: Simple Directory Monitor!  by ssooking''')
    )
    parser.add_argument('-w','--watch',action="store",dest="path",default="/var/www/html/",help="d
    args=parser.parse_args()
    Monitor(args.path)
```

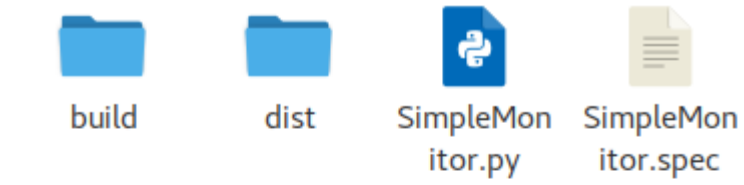
关于pyinotify 库的用法不再赘述，可以看到我在上述代码中创建了一个事件监控处理的类EventHandler，在这个示例中，我们仅仅关注创建、删除、修改属性、移动操作事件，并且我定义了一个DeleteFileOrDir方法用于自动删除增加的目录或者文件。运行测试截图：

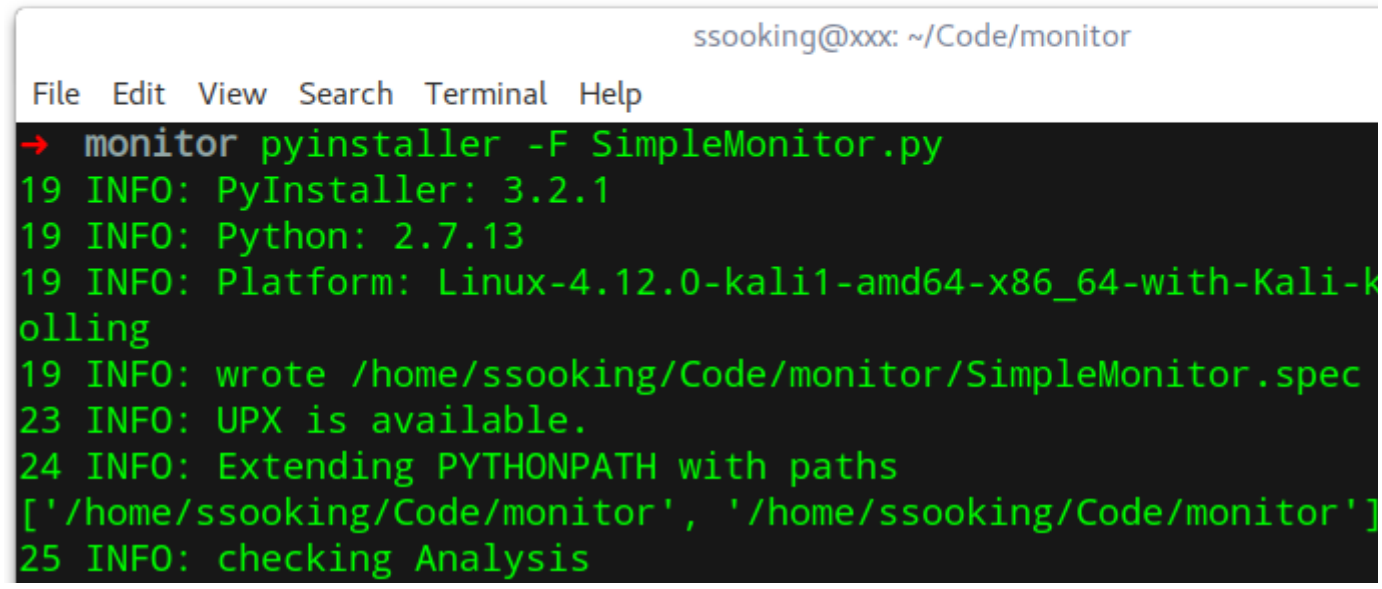
```
→  Tmp cd testdir
→  testdir ls
1.php  abc  bbb  test.py
→  testdir chmod +x test.py
→  testdir rm 1.php
→  testdir touch aaa.txt
→  testdir cd ..
→  Tmp mv aabb testdir/
```

```
python SimpleMonitor.py -w /home/ssooking/Tmp/testdir
File  Edit  View  Search  Terminal  Help
→  ~ python SimpleMonitor.py -w /home/ssooking/Tmp/testdir
[+] Now Starting Monitor:  /home/ssooking/Tmp/testdir
[!] Attribute been modified:/home/ssooking/Tmp/testdir/test.py
[!] Delete : /home/ssooking/Tmp/testdir/1.php
[!] Create : /home/ssooking/Tmp/testdir/aaa.txt
     >>> Delete file successfully/home/ssooking/Tmp/testdir/aaa
[!] Attribute been modified:/home/ssooking/Tmp/testdir/aaa.txt
[!] Delete : /home/ssooking/Tmp/testdir/aaa.txt
[!] File or dir been moved to here: /home/ssooking/Tmp/testdir/
     >>> Delete directory successfully: /home/ssooking/Tmp/test
[!] Delete : /home/ssooking/Tmp/testdir/aabb
```

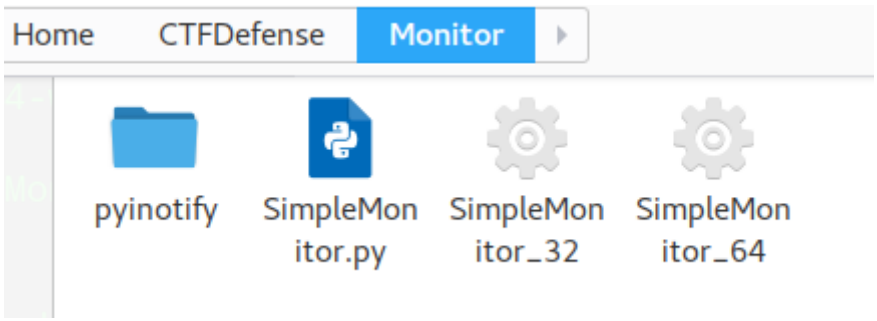
我们可以编写功能更加细化的程序，实现如监控文件变更，禁止创建、修改、删除任何文件或目录，自动删除新增文件，把被修改的文件改回去，删除畸形隐藏文件等功能。我们使用pyinstaller把我代码打包为linux的elf可执行文件。-F参数表示打包为独立可运行文件，命令

执行完之后自动生成：build、dist文件夹和SimpleMonitor.spec文件，你可以在dist目录里找到生成的elf程序。





打包的文件在CTFDefense项目的Monitor目录下



4. 网络监控断异常连接

linux安全防护一定少不了 iptables了，使用iptables需要有管理员权限。对于比赛环境，我们完全可以配置一个近乎苛刻的配置防火墙策略。

具体我们可以做哪些工作呢，举一些例子：

（1）关闭所有网络端口，只开放一些比赛的必要端口，也可以防止后门的连接

```
#开放ssh
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

#打开80端口
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT

#开启多端口简单用法
#iptables -A INPUT -p tcp -m multiport --dport 22,80,8080,8081 -j ACCEPT

#允许外部访问本地多个端口 如8080, 8081, 8082,且只允许是新连接、已经连接的和已经连接的延伸出新连接的会话
iptables -A INPUT -p tcp -m multiport --dport 8080,8081,8082,12345 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p tcp -m multiport --sport 8080,8081,8082,12345 -m state --state ESTABLISHED -j ACCEPT
```

（2）限制ssh登陆，进行访问控制

```
iptable -t filter -A INPUT -s 123.4.5.6 -p tcp --dport 22 -j DROP     //禁止从123.4.5.6远程登陆到本机
iptables -A INPUT -s 123.4.5.6/24 -p tcp --dport 22 -j ACCEPT    //允许123.4.5.6网段远程登陆访问ssh
```

（3）限制IP连接数和连接速率

我们可以限制IP的网络连接数和速度等，限制过快的连接频率，这样可以在一定程度上限制对方的扫描器。狠一点的话，甚至可以让对方只能以手工点网页的速度与访问+_+

单个IP的最大连接数为 30

```
iptables -I INPUT -p tcp --dport 80 -m connlimit --connlimit-above 30 -j REJECT
```

单个IP在60秒内只允许最多新建15个连接

```
iptables -A INPUT -p tcp --dport 80 -m recent --name BAD_HTTP_ACCESS --update --seconds 60
--hitcount 15 -j REJECT
iptables -A INPUT -p tcp --dport 80 -m recent --name BAD_HTTP_ACCESS --set -j ACCEPT
```

允许外部访问本机80端口，且本机初始只允许有10个连接，每秒新增加2个连接，如果访问超过此限制则拒接 （此方式可以限制一些攻击）

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 2/s --limit-burst 10 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
```

再猥琐一点，可以定时断开已经建立的连接，让对方只能断断续续的访问~~

（4）数据包简单识别，防止端口复用类的后门或者shell

假设病毒木马程序通过22，80端口像服务器外传送数据，这种方式发向外发的数据不是我们通过访问网页请求而回应的数据包。我们可以禁止这些没有通过请求回应的数据包。

```
iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEP
```

（5）限制访问

如果对方来势太凶，我们可以限制或者封杀他们的ip段。

```
iptable -t filter -A FORWARD -s 123.4.5.6 -d 123.4.5.7 -j DROP    //禁止从客户机123.4.5.6访问
123.4.5.7上的任何服务
```

封杀123.4.5.6这个IP或者某个ip段

```
iptables -I INPUT -s 123.4.5.6 -j DROP
iptables -I INPUT -s 123.4.5.1/24 -j DROP
```

（6）过滤异常报文

iptables有一个TCP匹配扩展协议--tcp-flags，功能是过滤TCP中的一些包，比如SYN包，ACK包，FIN包，RST包等等。举个例子，我们知道SYN是建立连接，RST是重置连接，如果这两个同时出现，就知道这样的包是有问题的，应该丢弃。下面的例子是利用--tcp-flags参数，对一些包进行标识过滤，扔掉异常的数据包。

```
iptables -A INPUT -p tcp --tcp-flags SYN,FIN,ACK,RST SYN                #表示 SYN,FIN,ACK,RST
的标识都检查，但只匹配SYN标识
iptables -A INPUT -p tcp --syn                                          #匹配SYN标识位
iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP            #检查所有的标识位，匹配
到FIN URG PSH的丢弃
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP                   #丢弃没标志位的包
iptables -A INPUT -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP  #匹配到SYN ACK FIN URG
的丢弃
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN,RST -j DROP            #匹配到SYN ACK FIN
RST的丢弃
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN,PSH -j DROP           #匹配到SYN FIN PSH的丢
弃
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN,RST,PSH -j DROP        #匹配到SYN FIN RST
PSH的丢弃
iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP           #匹配到 SYN,RST的丢弃
iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP           #匹配到 SYN,FIN的丢弃
```

（7）防DDOS攻击

```
iptables -A INPUT -p tcp --dport 80 -m limit --limit 20/minute --limit-burst 100 -j ACCEPT
```

-m limit: 启用limit扩展
–limit 20/minute: 允许最多每分钟10个连接
–limit-burst 100: 当达到100个连接后，才启用上述20/minute限制

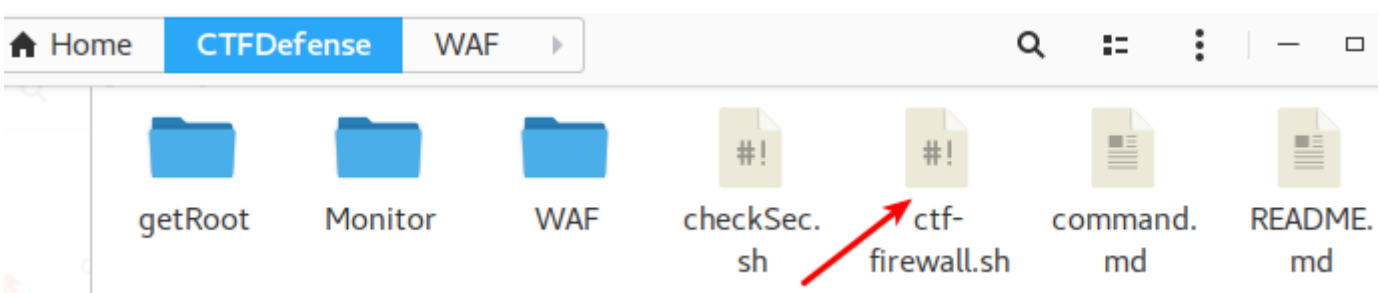
丢弃陌生的TCP响应包,防止反弹式攻击

```
iptables -A INPUT -m state --state NEW -p tcp ! --syn -j DROP
iptables -A FORWARD -m state --state NEW -p tcp --syn -j DROP
```

更多的姿势，需要打开我们的脑洞了，下面是一个通用的firewall脚本，我们可以传到服务器上一键执行，相关参数可以查阅资料详细了解：

```
#!/bin/bash

#Allow youself Ping other hosts , prohibit others Ping you
iptables -A INPUT -p icmp --icmp-type 8 -s 0/0 -j DROP
iptables -A OUTPUT -p icmp --icmp-type 8 -s 0/0 -j ACCEPT


#Close all INPUT FORWARD OUTPUT, just open some ports
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

#Open ssh
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

#Open port 80
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT


#Open multiport
#iptables -A INPUT -p tcp -m multiport --dport 22,80,8080,8081 -j ACCEPT


#Control IP connection
#The maximum number of connections for a single IP is 30
iptables -I INPUT -p tcp --dport 80 -m connlimit --connlimit-above 30 -j REJECT

#A single IP allows up to 15 new connections in 60 seconds
iptables -A INPUT -p tcp --dport 80 -m recent --name BAD_HTTP_ACCESS --update --seconds 60
--hitcount 15 -j REJECT
iptables -A INPUT -p tcp --dport 80 -m recent --name BAD_HTTP_ACCESS --set -j ACCEPT


#Prevent port reuse
iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT


#Filter abnormal packets
iptables -A INPUT -i eth1 -p tcp --tcp-flags SYN,RST,ACK,FIN SYN -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL FIN,URG,PSH -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN,RST -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN,PSH -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN,RST,PSH -j DROP
iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

#Prevent DoS attacks
iptables -A INPUT -p tcp --dport 80 -m limit --limit 20/minute --limit-burst 100 -j ACCEPT

#Discard unfamiliar TCP response packs to prevent rebound attacks
iptables -A INPUT -m state --state NEW -p tcp ! --syn -j DROP
iptables -A FORWARD -m state --state NEW -p tcp --syn -j DROP
```

注意，对于不同的iptables版本，一些参数的用法可以会有略微的差异，使用时我们可能要根据需要进行修改。



5. 综合分析控阻溢出类攻击

关于溢出类攻击，我还没有总结出一些很实用的姿势，这里提供一些思路。

一般来说，溢出攻击成功后，会建立shell通道和网络连接，我们可以配合前面提到的命令，从这两方面入手进行检测和阻隔：

（1）检测高权限的进程

（2）检测sh，bash等进程

（3）检测建立的网络连接

（4）检查开放的端口

例子：通过端口和bash发现可疑进程

```
# netstat -antulp
 Internet connections (servers and established)
Recv-Q Send-Q Local Address           Foreign Address         Stat
     0      0 0.0.0.0:22              0.0.0.0:*               LIST
     0      0 127.0.0.1:3306          0.0.0.0:*               LIST
     0      0 0.0.0.0:80              0.0.0.0:*               LIST
     0    344 [redacted]:22           [redacted]:36774        ESTA
     0      0 [redacted]:9999         [redacted]:46482        ESTA
     0      0 [redacted]:22           [redacted]:34284        ESTA
     0      0 [redacted]:44795        [redacted]:9988         ESTA
     0      0 :::22                   :::*                    LIST
     0      0 0.0.0.0:69              0.0.0.0:*
# ps -aux | grep bash
    1049  0.0  0.2  11624  2680 ?        S    08:29   0:00 /bin/ba
   10359  0.0  0.4  23264  4892 pts/0    Ss   16:57   0:00 -bash
   17537  0.0  0.4  23204  4940 pts/1    Ss   18:24   0:00 -bash
   18836  0.0  0.2  13224  2968 pts/0    S+   18:40   0:00 bash
   18994  0.0  0.0  12732   956 pts/1    S+   18:42   0:00 grep ba
```

如果我们怀疑某个进程正在是受到溢出攻击后创建的shell进程，我们可以分析这个进程是否有socket连接，linux中查看指定进程socket连接数的命令为：

```
ls /proc/<进程pid>/fd -l | grep socket: | wc -l
```

比如我们查看ssh进程的socket连接。如果我们检测的程序有socket连接，说明它正在进行网络通信，我们就需要进行进一步判断。

```
root@VM-247-51-debian:/var/www/html/sXh# netstat -tunlp|grep 22
tcp        0      0 0.0.0.0:22              0.0.0.0:*
tcp6       0      0 :::22                   :::*
root@VM-247-51-debian:/var/www/html/sXh# ls /proc/372/fd -l | grep
4
root@VM-247-51-debian:/var/www/html/sXh# ls /proc/372/fd -l | grep
lrwx------ 1 root root 64 Sep 22 08:26 1 -> socket:[11033]
lrwx------ 1 root root 64 Sep 22 08:26 2 -> socket:[11033]
lrwx------ 1 root root 64 Sep 22 08:26 3 -> socket:[11408]
lrwx------ 1 root root 64 Sep 22 08:26 4 -> socket:[11417]
```

我们还可以检测可疑进程开启的管道。linux下查看进程管道数的命令类似：

```
ls /proc/<进程pid>/fd -l | grep pipe: | wc -l
```

```
 >> :~# ps -ef | grep apache2
root     17266     1  0 18:21 ?        00:00:00 /usr/sbin/apac
www-data 17267 17266  0 18:21 ?        00:00:00 /usr/sbin/apac
www-data 17268 17266  0 18:21 ?        00:00:00 /usr/sbin/apac
www-data 17269 17266  0 18:21 ?        00:00:00 /usr/sbin/apac
www-data 17270 17266  0 18:21 ?        00:00:00 /usr/sbin/apac
www-data 17271 17266  0 18:21 ?        00:00:00 /usr/sbin/apac
root     17572 17537  0 18:24 pts/1    00:00:00 grep apache2
 >> :~# ls /proc/17266/fd -l | grep pipe:
lr-x------ 1 root root 64 Sep 22 18:21 4 -> pipe:[275057]
l-wx------ 1 root root 64 Sep 22 18:21 5 -> pipe:[275057]
 >> :~# _
```

典型的一个例子是： Apache模块后门mod_rootme， 它复用了webserver的80端口，mod_rootme通过管道和bash交互数据，但是由于开启了额外的管道，我们从这个变化上便能察觉到。

详细内容可以参考： http://t.qq.com/p/t/330573116082464 。

总体来说，我们主要可以关注进程情况和网络连接情况，综合分析进程，阻断溢出攻击创建的shel的。

6. 漏洞修复简单粗暴

CTF比赛中修复漏洞主要就是为了防止其他队伍的入侵了。

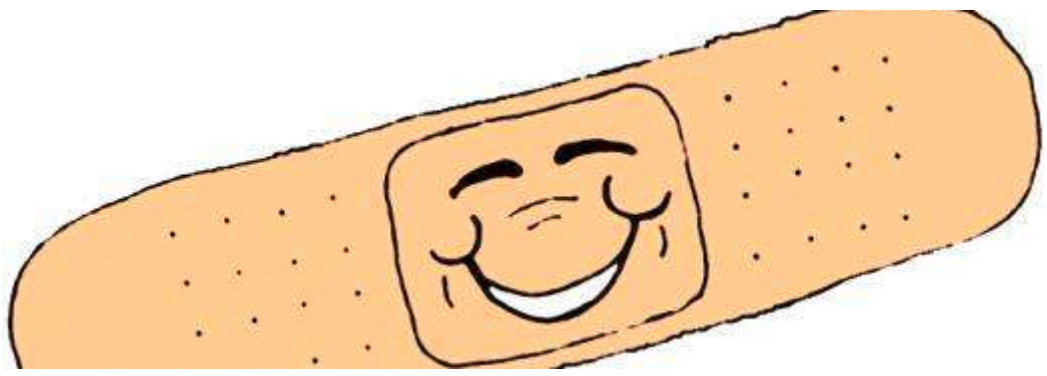
1．删站：如果赛组没有明确禁止，这是最粗暴的姿势，只留自己的webshell，参加过几场比赛确实遇到了这种尴尬的事情，web攻防最后都演变成了拼手速的“GetShell+留后门+删站”。

2．删漏洞页面：大部分举办方还是会明确禁止删除网站的，通常赛组会定期访问网站主页（一般来说），从而确定网站是否正常运行。其实我们没必要删除整个网站，只要删掉有漏洞的页面就行了，比如删后台登录页面、注册页面、上传页面等等。

3．破坏正常功能：如果明确不能删除任何页面，可以选择让这些漏洞点的功能函数（或者其依赖的功能函数）失效。比如上传点，如果考虑过滤挺麻烦，又不能删页面，那么我们可以找到这个漏洞网页，改掉或者删掉文件里对应的类似upload等这种功能调用函数。

上面这三种其实都算不上修补漏洞了，真实环境下哪能这么干。

4．采用正常修补手段：规则限定很严的情况下，我们还是采用正常手法吧，修改服务配置、安装补丁、下载更新的软件版本、加过滤等等。



>>>>>_<<<<<<

谈到那个删站，我忍不住想吐槽几句，我个人是不赞成这种做法的，因为这种操作在比赛中很容易拉仇恨并且产生连锁反应，也不利于个人的进步。有些人认为：我先拿下这台服务器说明我有能力，你没进去就说明你没本事，所以我先进去删了站不让其他人进也无可厚非，有能耐你就先拿下它，那你删了我也没意见。也有人说，真正的对抗里，敌人不会对你仁慈～～没错，挺有道理的，而且强者也应该拥有发言权。但是我们不能只从一个角度考虑问题，换个角度去考虑，CTF竞赛虽然是向着真实的网络环境靠拢，但是它的根本目的是提高竞技者的安全技能和知识水平。对于大部分切磋竞技的玩家来说，参加比赛也都是为了让自己获得提高，大家在同一个平台上进行进行切磋对抗，认识到自己和对手身上的优点与不足，这才是竞技。但是这种行为其实从某种意义上来说已经破坏了比赛的公平性，毕竟如果是因为技术不到位，那当然没什么话说，但是如果网站删了，让别人发挥的地方都没有，这种切磋也没有意义。举个不太恰当的例子，这就像两个人打架，你说你比我强，咱们比比再说，但是你都不让我跟你打，这算什么。再换个角度，其实我一直坚信真正具有强者姿态的人，不畏惧挑战、不怕被人超越，不屑于通过这种手段

巩固自己的地位。相反，我们只有将自己至于狂风大浪中，才能成长和蜕变，最终成为一个强者。

无论从什么角度考虑，我们应该敢于挑战自身、挑战别人，不断强大自己，不断去征服，无畏无惧、步履铿锵!

7. 安全软件锦上添花

可以使用第三方软件的话，装个WAF，安全狗之类的吧。这个我没什么话要说，附个linux安全狗的链接吧：
安全狗linux版： http://www.safedog.cn/website_safedog.html

我们平时也可以搜集或者自己实现一些脚本或者工具备用。

这里有waf一枚： http://hackblog.cn/post/75.html

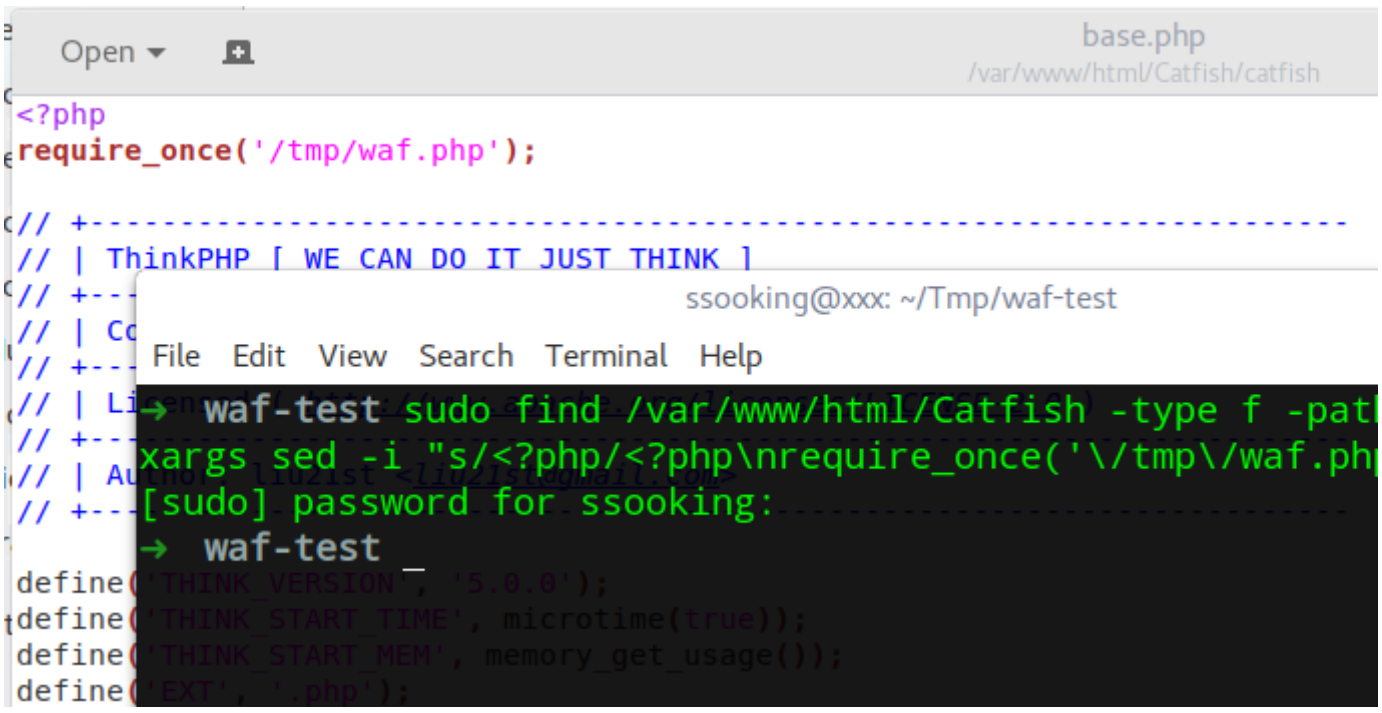
如果我们想给web目录文件添加自定义waf脚本，其实可以用一条命令解决,以php为例：

```
find /var/www/html -type f -path "*.php" | xargs sed -i "s/<?php/<?php\nrequire_once('\/tmp\/waf.php');\n/g"
```

命令的意思就是查找/var/www/html目录下所有php文件，在头部添加一句，用require函数引入/tmp/waf.php文件。因为sed命令利用 / 区分文件中的原字符串和修改的字符串，所以我们要对 / 进行转义。类似于在单引号中再次使用单引号时我们也要用反斜杠转义：\'，命令转换过来就是这样：

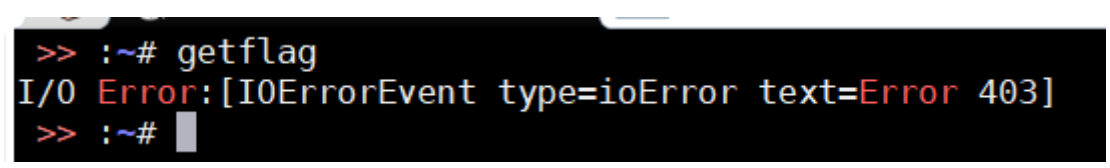
```
find /var/www/html -type f -path "*.php" | xargs sed -i "s/<?php/<?php\nrequire_once('/tmp/waf.php');\n/g"
```

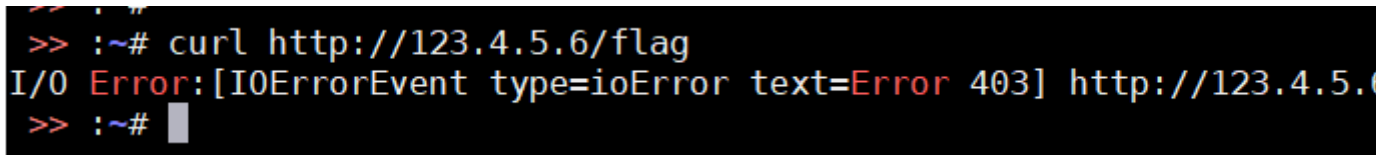
这样，再次打开时我们就会发现已经引入了自定义waf文件。



8. 我可能get了假的flag

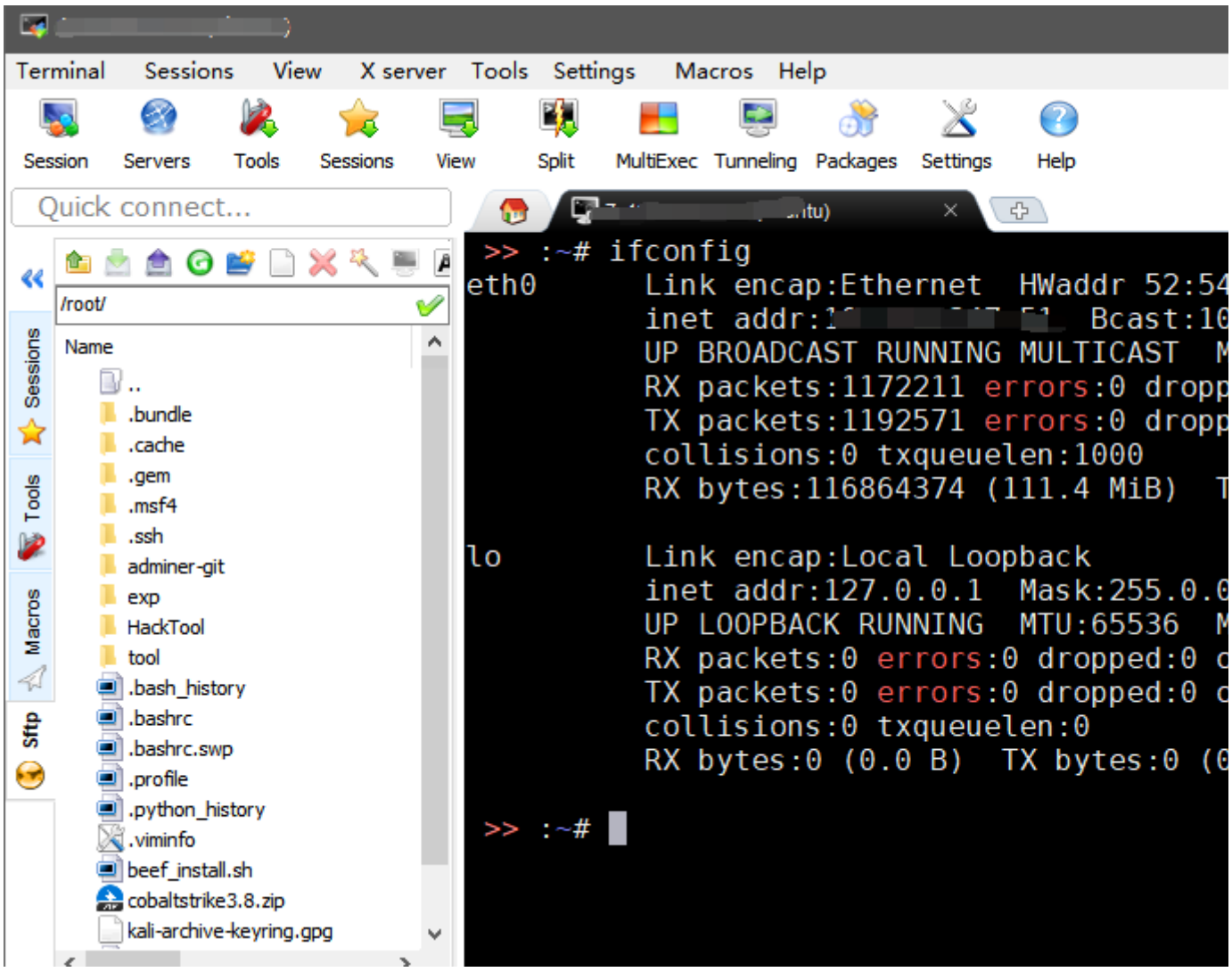
如果说很不幸，我们前面的关卡都被突破了（实际上我都感觉前面那些设置都有点“搅屎”的味道了，不过还是希望师傅们能一起来讨论讨论有没有什么骚姿势，以及绕过它们的方法）。假设真的被突破了，对于CTF线下赛来说，我们最终的目的都是拿到flag。通常我们会在服务器上执行类似于"getflag"命令，或者"curl"访问某个url获取flag，然后获取到一个字符串，然后在答题平台上提交这段字符串即可获取分数。就拿前之前的ISCC来说，这个也是我赛后想到的。这个getflag是一个elf的程序，在/usr/bin/下，顺手给下载了，有兴趣的同学可以去逆向一波。重点在这，有几次我getflag的时候因为webshell丢了，服务器显示了Error。后来想想，我们是不是可以故意利用这种报错来欺骗不细心的竞争对手呢，当然我不知道是不是已经有师傅们用了这个手法。这是模拟的效果：



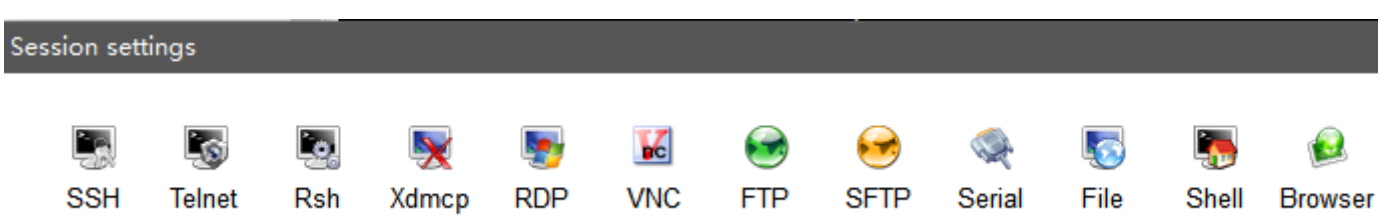


怎样实现？比如我们可以添加alias别名，或者我们可以把这些命令更改或者替换掉，换成一些伪装命令程序。再换一层想想，接着上面的思路，如果我们替换或者伪装了系统命令，对方getshell之后，进来发现cd，ls等命令都没法用，会怎么样呢？然而这样会不会不太好～～

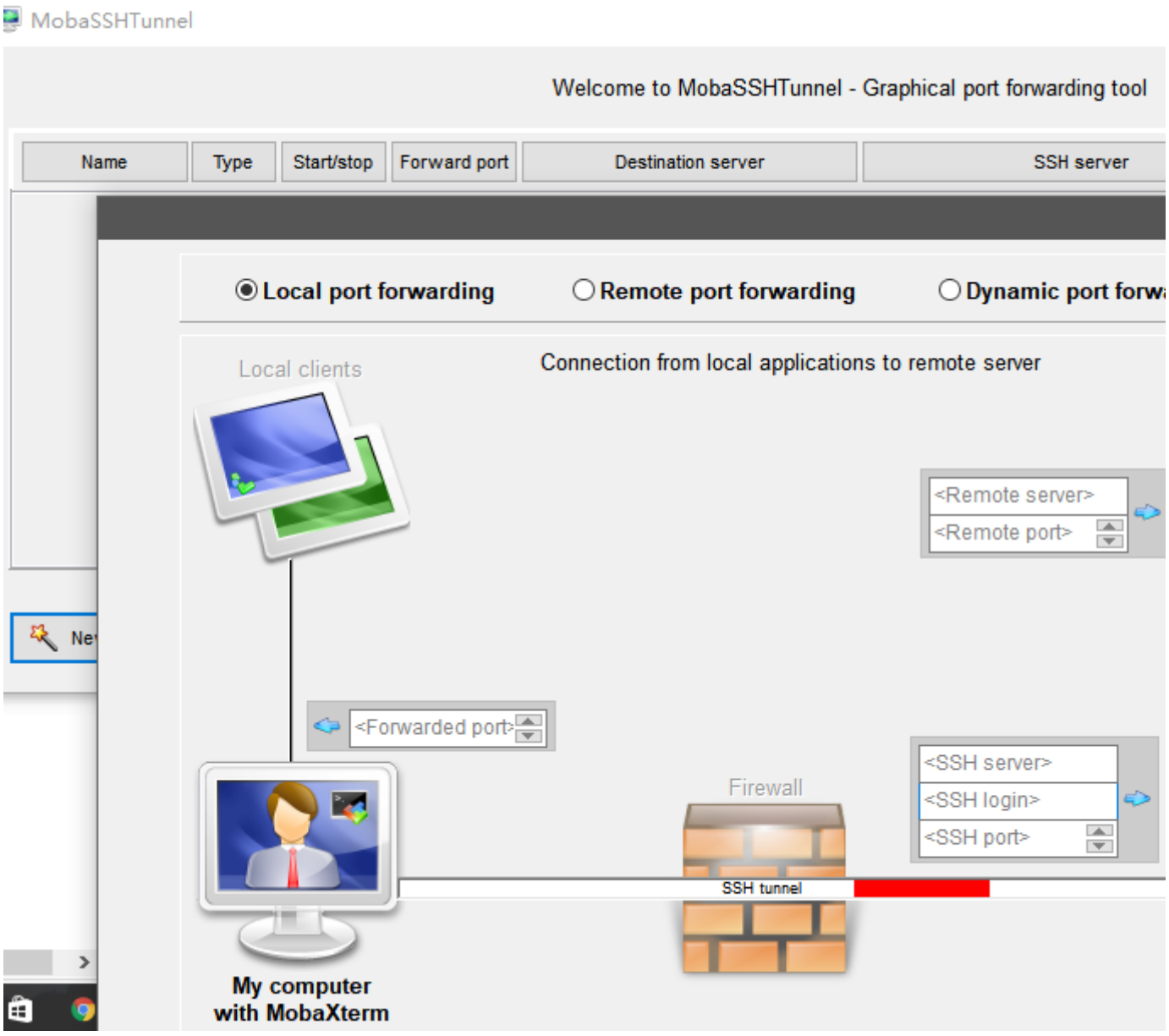
最后推荐一个感觉挺实用功能很强的远程连接管理工具，可以极大方便我们的工作：MobaXterm 。（不是打广告~）



它支持多种连接方式，可以拖拽进行文件管理。支持在打开的会话一键批量执行命令



还有一个非常方便的ssh端口转发功能，支持本地、远程、动态转发。



还有很多其他功能貌似很厉害，不过我没用过，就不说了...

真不是打广告。

三. 对CTF举办的一点小小建议

如今CTF越来越火，对于这些比赛的举办方，我有着一些不成熟想法和小建议，如果您觉得有什么不合适的地方，纯当娱乐：

（1）扩展竞技形式：目前线下赛web攻防占绝大多数，有些小比赛甚至只有若干web服务器，上面放几个不同类型的站点，形式有些单一了，其实可以增加多种对抗模式，甚至可以让参赛选手走出比赛场地。去年曾有幸聆听了诸葛建伟博士关于打破XCTF现有格局的讲座，他提出了体系化攻防演练，认为CTF可以引入实地wifi渗透、门禁系统突破、无人机攻防、GPS信号对抗等，增加比赛多样性与趣味性，让线下赛不再只是局限于小小的机房~~

（2）重视安全分析与防护。安全不仅仅只是网络攻防对抗，数据分析、取证分析、应急响应、追踪溯源等技能也相当重要，并且在安全人才圈里这方面缺口也比较大。今年六月份，启明星辰主办的2017”信息安全铁人三项赛"（分为”个人逆向赛“、”数据分析赛“、”企业攻防赛“三个赛事），其中”数据分析赛“便是一个典型代表，参赛选手需要分析真实企业网络环境下受网络攻击的流量数据，从中找出攻击者、分析其网络攻击行为，如欺骗劫持、爆破、webshell连接操作等，找到并分析攻击者的后门或者恶意软件。这种模式，有助于参赛者接触到相对更加真实的网络攻击流量的对抗与防御。

（3）完善竞技模式的具体细节，尽量避免取巧或者粗暴姿势。比如拿修补漏洞举例子，现在CTF线下赛中绝大部分参赛者为了维持加固自己的shell，往往都会采用删除部分页面的方法，比如登陆、注册页面，因为采用正常打补丁、修改配置等操作都比较费时费事。但在比赛中这种方式是对于学习真正的安全加固、漏洞修补知识没有太多提高。玩CTF不应该仅仅为了比赛而比赛，或者只是为了拿个奖、拿几张证书，还是要注重从中学到点东西，不过有证书对以后就业还是有些帮助的。

虽然说这些会增加举办方的负担，给选手增加难度，但是这也是一种趋势。CTF必然要经历从普及到提高的转变，并且随着参赛选手水平的提高，我们确实需要一些更有意思的玩法，这是一个相互促进的关系。当然，对于入门级的CTF选手来说，题目难度过大反而会降低比赛体验，对于不同级别的玩家，可以设置不同级别的赛事。从形式上奖，像引入门禁系统突破、无人机攻防等，对于大部分CTF举办方来说实现起来有些难度，毕竟涉及到不同的环境、设备、人员维护等问

题，所以这个不应该强求，但是对网络攻防来说增加如windows 服务器、邮件服务器、路由设备等还是可行的。以后的CTF规格和水平会越来越高，对于参赛选手的挑战难度也会越来越大，这对于举办方和选手来说都是挑战，但是挑战亦是机遇，我们应时刻准备好投入战斗！

四. 最后的话

虽说上面提到的这些姿势不可能让我们的靶机变得无懈可击，但是至少能在某种程度上提高它的防御值，希望能对大家有所帮助。最后，祝愿各位CTF参赛选手在比赛中勇创佳绩，同时也祝愿各单位的大佬们都能把CTF筹办的越来越好！

分类： CTF

« 上一篇： 攻击流量的清洗
» 下一篇： 60cms Cookies欺骗漏洞审计

posted @ 2017-10-11 16:29 ssooking 阅读(212) 评论(0) 编辑 收藏