2019年7月11日和12日

首先，今天遇到了很多问题。比如物理机无法通过ssh连接到ubandu（虚拟机）。然后我之前的ubandu又不知道为什么直接变成了dos界面。快照也没有。然后文件下载了一个别人的pwn专用的虚拟机，不是很会用16版本的。首先python2.7不会装。然后我不知道为什么，apt-get install python 它默认是下载python3.然后今天这么多麻烦的原因的gdb的问题。不会使用gdb。然后又因为gdb的黑白文字根本看不了。所以准备看看有没有语法高亮的插件。结果就准备装pwngdb和peda。我也不是很清楚这个东西的原理。但是，我看到很多教程他们都在用这个东西，语法高亮而且很方便。

这两天学习到了 ps -e|grep XXX 查看是否在后台运行。如果有就kill 对应的PID。以此来释放系统锁。<用apt-get install 安装软件的时候，如果在未完成下载的情况下将terminal close。此时 apt-get进程可能没有结束。结果，如果再次运行apt-get install 命令安装软件时，出现如上错误。即有另一个程序正在占用apt-get install进程，由于它在运行时，会占用软件源更新时的系统锁（简称‘系统更新锁’），此时资源被锁>还有一种方法是强制解锁（sudo rm /var/cache/apt/archives/lock）（sudo rm /var/lib/dpkg/lock）。

配置Linux的ssh服务，sudo apt-get install ssh openssh下载ssh和openssh，接下来修该/etc/ssh/sshd\_config。无论用nano还是vim。

service ssh restart。重启ssh服务。update-rc.d ssh enable修改为开机自启。

然后docker用到了windows pro版的hyper-V的虚拟化功能，但是docker轻量，方便，软件级别的虚拟化。不想vmware直接是硬件级别的虚拟化。由于我要用vm虚拟机，所以我把hyper-V禁用了。然后放弃了windows上用docker。然后值得说一下的是hyper-V的虚拟网卡DockerNAT，默认交换机，桥接网络要禁用。不然用vm的时候使用桥接或者别的会导致没网的。

然后桥接，就勾选直接复制物理机选项。（相当于内网的一台真实电脑），然后NAT的话，就是和主机一样的IP，可以用来做端口映射。

各有各的好。

然后还学习到了git和github的怎么邀请别人一起改代码。怎么上传。怎么向github上拉取代码。

然后是Linux上面的脚本是./MOMO.sh和Windows上面的.bat文件差不多。

学了一点wirte（）和read（）函数的知识，还没有掌握怎么使用漏洞来pwn。然后32位的用p32，64位的用p64来写payload.

今天懂了一点点pwn专用虚拟机的一些配置，

1. 比如要安装（为64位系统提供32位运行环境支撑）。因为有些程序是32位的。不是64位操作系统兼容32位的程序吗？为什么还要安装？难道因为Linux不同吗？ <dpkg --add-architecture i386>< sudo apt-get -y install lib32z1 lib32ncurses5>
2. gdb的插件peda.<git clone https://github.com/longld/peda.git ~/peda><echo "source ~/peda/peda.py" >> ~/.gdbinit>
3. 用来调试堆的插件pwndbg. <cd ~/> <git clone <https://github.com/scwuaptx/Pwngdb.git>> <cp~/Pwngdb/.gdbinit ~>
4. ropgadget 做题的工具，用来找gadget的.< 先安装Capstone是一个轻量级的多平台架构支持的反汇编架构，支持包括ARM\ARM64、MIPC和x64/x86平台。 sudo apt-get install python-capstone 然后，下载好ROPgadget解压进入文件夹中 python setup.py install $ ROPgadget >
5. libcsearcher用来泄露libc库中函数的偏移的库.< git clone https://github.com/lieanu/libc. ><cd libc><git submodule update --init –recursive><sudo python setup.py develop>
6. one\_gadget用来寻找libc库中的execve('/bin/sh', NULL, NULL)可以一个gadget就可以getshell的东西.< 先安装gem sudo apt install ruby 然后 gem install one\_gadget>

换pip和系统的源。

发现它将flag读取到栈上了，结合它开的防护机制NX，可以想到smash the stack这种攻击手法，利用 \_\_stack\_chk\_fail 来打印想要的信息，因为flag在栈上，所以要先泄露出栈的地址，然后将argv[0]覆盖成flag的地址，通过触发 \_stack\_chk\_fail来将flag打印出来。栈的地址可以通过libc中的一个变量 \_environ变量泄露出来。因为在libc中的全局变量 environ储存着该程序环境变量的地址，而环境变量是储存在栈上的，所以可以泄露栈地址，进而计算出flag在栈上的地址。<新思路>

然后我还在苦苦在栈溢出的libc初级那里挣扎。