CTF环境搭建

1. 安装linux环境

A 虚拟机需使用Ubuntu16.04 amd64 Desktop或者Server版

镜像下载地址

B 虚拟机软件选择

- VisualBox (Win&Mac, 开源)
- VMvare (Win&Mac)
- Parallels Desktop (Mac用户强烈推荐，pt上有破解版)

2. 软件安装

默认系统安装python，如果没有：

```
sudo apt-get install python2.7
```

其他软件安装env.sh

```
#!/bin/bash
cd $HOME

# 更换apt源镜像加速
{
echo "deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial main restricted univ
erse multiverse";
echo "deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-updates main restric
ted universe multiverse";
echo "deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-backports main restr
icted universe multiverse";
echo "deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-security main restri
cted universe multiverse";
} | sudo tee /etc/apt/sources.list 1>/dev/null

# 安装软件- vim, tmux, pip, 32位执行文件依赖库, ssh
sudo apt update
sudo apt -y install vim python-pip binutils \
    nasm gcc-multilib g++-multilib \
    libc6-dev-i386 \
    libc6-dbg libc6-dbg:i386 \
    libssl-dev gdb curl nmap \
    git tmux binwalk \
    openssh-server

# 更换pip源镜像加速
mkdir .pip && echo -e "[global]\nindex-url = https://pypi.tuna.tsinghua.edu.cn/sim
ple" > .pip/pip.conf

# 安装pwntools
sudo pip install --upgrade pip
sudo pip install --upgrade capstone
sudo pip install --upgrade pwntools

# 安装gdb-peda
git clone https://github.com/longld/peda.git ~/peda
echo "source ~/peda/peda.py" >> ~/.gdbinit
echo "DONE! debug your program with gdb and enjoy"
```

3. pwntools学习

Pwntools的作用是屏蔽细节，方便快速编写exploit脚本

教程:

六星尚红泽同学写的简单教程(需要翻墙)

4. IDA- 强大的反编译工具

A.功能

反编译与反汇编32位与64位可执行文件，得到C语言源代码，方便找漏洞。

注：不建议用IDA做拆炸弹实验

B 安装

下载地址(内网)

C. 常用功能

- IDA初始刚打开时是汇编代码，点击右键可以切换Graph View和Text View
- F5：反编译
- n： 修改函数或变量名
- r： 修改常量类型
- y： 修改变量类型
- g：跳转到指定地址
- x： 查看引用当前函数的位置

D. 注意

Mac OS Mojave系统，在中文输入法下，IDA7.0会crash。 如果受不了就试试这个解决方案

5. PEDA-GDB插件

A 简介

peda(Python Exploit Development Assistance for GDB)为gdb提供了很多人性化的功能，比如高亮显示反汇编代码、寄存器、内存信息，提高了debug的效率。如果不了解gdb请参考这个链接,CSAPP(深入理解计算机系统)第三章也有介绍。

B peda常用命令：

- aslr -- Show/set ASLR setting of GDB
- checksec -- Check for various security options of binary
- vmmap -- Get virtual mapping address ranges of section(s) in debugged process
- dumpargs -- Display arguments passed to a function when stopped at a call instruction
- dumprop -- Dump all ROP gadgets in specific memory range
- elfheader -- Get headers information from debugged ELF file

- elfsymbol -- Get non-debugging symbol information from an ELF file
- lookup -- Search for all addresses/references to addresses which belong to a memory range
- patch -- Patch memory start at an address with string/hexstring/int
- pattern -- Generate, search, or write a cyclic pattern to memory
- procinfo -- Display various info from /proc/pid/
- pshow -- Show various PEDA options and other settings
- pset -- Set various PEDA options and other settings
- readelf -- Get headers information from an ELF file
- ropgadget -- Get common ROP gadgets of binary or library
- ropsearch -- Search for ROP gadgets in memory
- searchmemlfind -- Search for a pattern in memory; support regex search
- shellcode -- Generate or download common shellcodes.
- skeleton -- Generate python exploit code template
- xormem -- XOR a memory region with a key