1、 GDB 常用命令

命令	选项（注释）	功能
$vim ~/.gdbinit	编辑gdb配置文件	切换pwndbg和peda
$gdb <executablefile>	以指定目标程序的方式打开gdb	进入gdb
pwndbg>list/l	line/func/start,end/+/-	查看程序保留的代码信息（要求gcc编译时加上-g选项）
pwndbg>checksec		查看程序保护措施
pwndbg>break/b	line/func	设置断点
pwndbg>run/r	遇到断点停止	运行程序
pwndbg>start/star		开始执行程序，在main函数第一条语句前停止
pwndbg>next/n pwndbg>step/s	C语言级别的单步调试	执行下一条源代码语句
pwndbg>nexti/ni pwndbg>stepi/si	汇编语言级别的单步调试	执行下一条汇编语句
pwndbg>info	frame/stack/r/b etc	查看程序信息
pwndbg>x	示例:x /40wx $sp	查看内存内容
gdb-peda>pattern	create size/offset value	产生特定长度字符串/计算偏移

2、 pwntools 库

常用命令	功能
p = process('executablefile')	导入本地文件
p = remote(url/ip)	连接远程主机
p.send(data)	发送数据
p.sendline(data)	发送一行数据（末尾加'\n'）
p.recv(numb=4096, timeout=default)	接收指定字节的数据，指定超时
p.recvuntil(delims, drop=False)	指定接收的pattern
p.recvline(keepends=True)	接收一行数据(末尾为'\n')
p.recvall()	接收到EOF
p.interactive()	进入交互界面
e = ELF('file')	读取ELF文件
p32、p64	打包数据（转换成二进制数据）
u32、u64	解包数据（转化成整数）

.

实验一：栈溢出漏洞的利用

一、实验目标

1. 栈溢出漏洞利用，编写 python 脚本 exp_1.py 获取 shell；

2. 掌握栈溢出漏洞的原理；

3. 掌握 ubuntu 系统中调试工具 gdb 及其插件 pwndbg、peda 的使用、windows 系统中的逆向静态分析工具 IDA 的使用、python 的 pwntools 库的使用。

二、内容与要求

1. 目标程序:pwn_1

2. 反汇编：使用 IDA Pro 7.0(32 位)反汇编源程序 pwn_1,使用 F5 插件功能查看伪代码，阅读汇编代码，理解源程序的运行流程。

系统环境：Windows 系统

3. 漏洞分析：对源程序反汇编后，寻找程序的漏洞（可利用点）并确定 payload(shellcode)。切换至 Ubuntu 虚拟机环境中，使用 pwndbg 调试程序。

4. 综合以上内容，编写 exp.

基本要求：获取本地 shell

格式要求：文件名:exp_1.py

所用库:pwntools

代码尽量简洁、标明 shellcode。

5. ctf-wiki: https://ctf-wiki.github.io/ctf-wiki/pwn/linux/stackoverflow/stack-intro-zh/

6. 建议学时：1 学时。

实验二：整数溢出与 ROP 技术

一、实验目标

1. 整数溢出漏洞利用，利用面向返回编程 ROP（Return-Oriented Programming）技术劫持控制流；

2. 掌握整数溢出原理与 ROP 技术原理；

3. 理解程序的执行流程。

二、内容与要求

1. 目标程序：pwn_2

2. 反汇编：使用 IDA Pro 7.0(32 位)反汇编目标程序 pwn_2,阅读汇编代码，使用 F5 插件查看伪代码，理解源程序的运行流程。

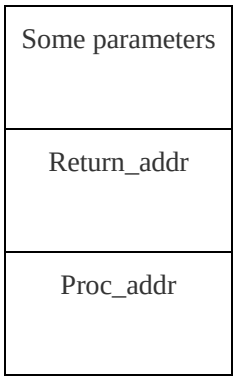
系统环境：Windows 系统

3. 漏洞分析：对源程序反汇编后，寻找程序的漏洞（可利用点）并确定 payload(shellcode)。

切换至 Ubuntu 虚拟机环境中，使用 pwndbg 调试程序。

4. ROP 技术利用：

ROP 技术建立在栈溢出漏洞原理之上，通过 ROP 技术可以构造一个控制流。



pwn_2 程序并没有后门函数 getshell,这里需要用到 libc 文件获取系统函数 system 和 /bin/sh 的地址从而构造出 system("/bin/sh")命令（/lib/i386-linux-gnu/libc.so.6）。

预备知识：PLT、 GOT 表和动态链接工程

构造思路：system 和 binsh 在内存中已装载，其实际地址等于其在 libc.so.6 的局部地址+libc.so.6 的首地址。于是我们需要确定 libc.so.6 的首地址，这里我们可以利用

pwn_2 和 libc.so.6 共有函数 puts 来泄露 libc.so.6 的首地址。

Libc.so.6 的首地址 = puts@got – puts 在 libc.so.6 中的局部地址。

（1）于是可以构造控制流：puts@plt + vlun_addr + puts@got

其含义为：Puts(puts_addr)+return_addr(vlun_addr),即执行了一次 puts 函数，

打印出 puts 函数在内存中的地址，并把返回地址设置为 vlun_addr,劫持控制流。

（2）知道了 puts 函数的地址，便可以计算出 libc.so.6 的首地址,顺理成章可以知道 system 和‘/bin/sh’字符串的地址。于是再构造控制流：system_addr+vlun_addr+binsh_addr，即可获取 shell。

5. 综合以上内容，编写 exp

基本要求：获取 shell

格式要求：文件名:exp_2.py

所用库：pwntools

代码尽量简洁、标明 shellcode。

6. Ctf-wiki: https://ctf-wiki.github.io/ctf-wiki/pwn/linux/stackoverflow/basic-rop-zh/

7. CSDN:https://blog.csdn.net/qq_18661257/article/details/54694748

8. 建议学时：2 学时。

实验三：Canary 保护的绕过

一、实验目标

1. 了解程序的保护机制- Canary、ALSR（PIE）、NX（DEP）等机制

2. 重点掌握栈保护 Canary 机制；

3. 理解 Canary 保护的绕过方法并完成实验。

二、内容与要求

1. 目标程序：pwn_3

2. 反汇编：使用 IDA Pro 7.0(32 位)反汇编源程序 pwn_3,使用 F5 插件查看伪代码，阅读汇编代码，理解源程序的运行流程。

系统环境：Windows 系统

3. Canary 保护：

该程序与第一题类似，不同的是在编译时开启了栈保护 Canary 机制。通常的缓冲区溢出保护会修改栈内存的结构，使其在栈底附近包含一个 Canary 值，当栈缓冲区溢出时，该 Canary 值能够指示出内存中的缓冲区溢出。 Canary 位于栈上的缓冲区和控制数据之间，当发生缓冲区溢出时，Canary 将会是第一个遭到破坏的数据，因此当对 Canary 数据验证失败时，系统会认定发生了栈缓冲区溢出，从而通过一定的操作对栈缓冲区进行处理。

绕过思路：Canary 的结构是固有的 4 字节数据其第四个字节是固定的 0x00,因此通过覆盖该字节将可能使程序在打印字符串的同时泄露出 canary 值。

4. 漏洞分析：对源程序反汇编后，寻找程序的漏洞（可利用点）并确定 payload(shellcode)。切换至 Ubuntu 虚拟机环境中，使用 pwndbg 调试程序。

Tip:pwndbg>canary 命令可以查看 canary 内容。

5. 综合以上内容，编写 exp

基本要求：获取 shell

格式要求：文件名:exp_3.py

所用库：pwntools

代码尽量简洁、标明 shellcode。

6. ctf-wiki:https://ctf-wiki.github.io/ctf-wiki/pwn/linux/mitigation/canary-zh/

7. 建议学时：1 学时。

实验四：格式化字符串漏洞利用

一、实验目标

1. 格式化字符串漏洞利用，实现内存读写；

2. 掌握格式化字符串漏洞原理。

二、内容与要求

1. 目标程序：pwn_4

2. 反汇编：使用 IDA Pro 7.0(32 位)反汇编源程序 pwn_4,使用 F5 插件查看伪代码，阅读汇编代码，理解源程序的运行流程。

系统环境：Windows 系统

3. 格式化字符串漏洞：

格式化字符串输入与输出函数是 C 语言的基本函数，格式化字符串通过%argc 来控制输入输出的格式，然而，当提供的参数少于%argc 的个数时，程序将泄露出栈中的内容；而用户非法使用%n 将可能导致内存非法写入。

4. 漏洞分析：对源程序反汇编后，寻找程序的漏洞（可利用点）并确定 payload(shellcode)。切换至 Ubuntu 虚拟机环境中，使用 pwndbg 调试程序。

5. 综合以上内容，编写 exp

基本要求：获取 shell

格式要求：文件名:exp_4.py

所用库：pwntools

6. ctf-wiki:https://ctf-wiki.github.io/ctf-wiki/pwn/linux/mitigation/canary-zh/

7. 建议学时：1 学时。