Review the assignment due date

栈溢出攻击实验

> "Exploiting is an art, but understanding the principles behind it is science." —— Anon

目录

简介

栈溢出攻击是一种常见的安全漏洞利用技术，它通过向程序的栈内存区域写入超出预期的数据量，导致覆盖相邻内存区域的数据，从而可能改变程序的执行流程。这种攻击手法可以用来执行恶意代码，获取系统权限，或者使服务崩溃。

栈溢出攻击原理

在计算机科学中，函数调用时会创建一个称为“栈帧”或“活动记录”的数据结构，用于存储局部变量、函数参数、返回地址等信息。栈是后进先出（LIFO）的数据结构，这意味着最后压入栈的数据将最先被弹出。当发生函数调用时，新的栈帧会被压入栈顶；当函数返回时，该栈帧从栈顶弹出。

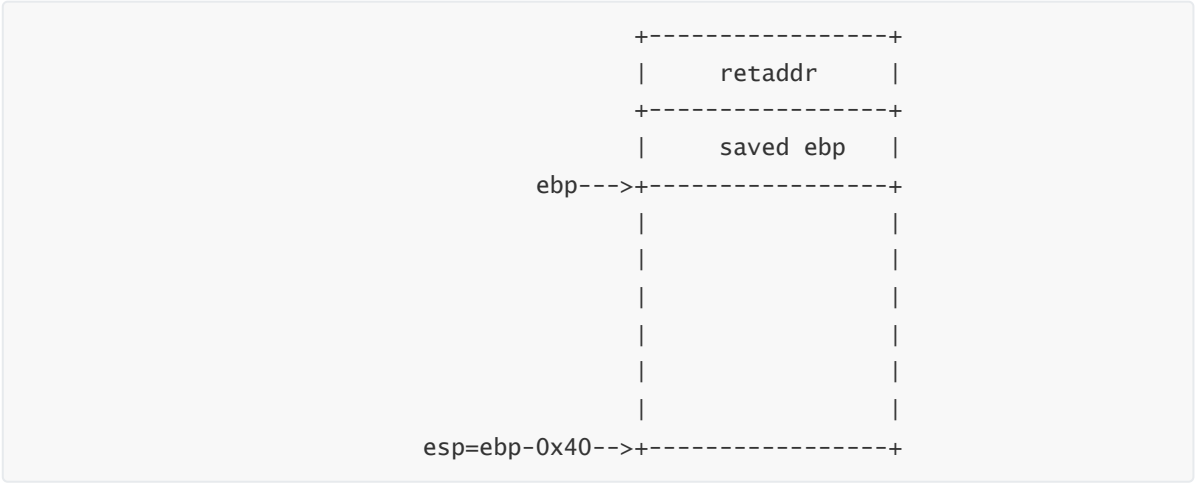
栈溢出的过程

1. **缓冲区分配**：当函数调用发生时，为局部变量分配一定的栈空间。
2. **用户输入**：如果函数接收了来自用户的输入，并且没有正确地检查输入长度，就可能导致写入的数据超过局部变量所分配的空间。
3. **栈帧覆盖**：过长的数据将覆盖栈上的其他信息，包括函数的返回地址。
4. **控制流劫持**：攻击者可以通过精心构造的输入来覆盖返回地址，使得函数返回时跳转到攻击者指定的代码位置，通常是恶意代码所在的位置。

栈溢出的一个简单例子

```
// 比方说现在有人向你暴露了一个操作接口，你还通过一些手段知道了这个操作背后是这样的代码
char s[24];
get(s);
```

实际上这个代码相当的不安全，这是因为get(s)可以让用户输出很长的字符串，从而影响了函数栈本身。
比方说这个这段代码所调用的函数栈如下



我们将你的输入定义为payload，那么此时payload会放在esp所指向的地址(注意，这是因为在32位中参数会放在栈上决定的，64位中不一定会放在rsp所指向的地址上)
但是如果你想要搞破坏的话，你可以通过设计你的payload来让函数最后跳转到别的函数上去，即修改retaddr的内容
比如说你的payload是：

```
payload='A'*0x40(覆盖esp指向的地址到ebp指向地址之间的内容)+'A'*4(覆盖saved ebp内容)+0xdeadbeef
#在python中,payload的形式更有可能是b'A'*0x44+b'\xef\xbe\xad\xde'，在后面会解释为什么格式是这个样子。
(覆盖retaddr，当函数执行结束后会跳转到0xdeadbeef执行下一条指令)
#payload指的是你的输入内容，后面我们将延续此含义
```

请想一下，如果0xdeadbeef上函数是system('bin/sh')这种系统函数调用的话，那岂不是就可以劫持到你电脑的权限了？（实际上第一代蠕虫病毒就是用的栈溢出的漏洞）

攻击手段

1. 可读写栈，即在函数栈内部注入一段转化为二进制的汇编代码，然后跳转到栈上执行
2. Ropgadget，合理利用程序内部的代码碎片进行传参、攻击。

实验目的

本实验旨在让参与者理解栈溢出攻击的基本原理，学会如何检测潜在的栈溢出漏洞，并掌握基本的防护措施。

实验环境

- 操作系统：Linux
- 编译器：GCC
- 其他工具：gdb, objdump

实验启动

```
git clone git@github.com:RUCICS/attacklab.git
cd attacklab
# let's get started
```

一般的实验步骤

以problem1为例

1. 首先你需要使用`objdump -d file`去获得 problem1 的实验代码，并认真阅读其中的关键部分
2. 确定好该题目要求你做什么，并设计相关payload输入来完成实验
    1. 你需要将你的payload保存为一个.txt文件(比如文件名为ans1.txt)，且.txt文件的内容是一个二进制流，也就是你要放在栈上的内容。之后运行./problem1 ans1.txt，便可以看到你的输出
    2. 这里提供一种将保存二进制流文件的方法，你需要使用python去运行下面的代码(注意 这只是一种写法，你可以按照自己的习惯去写)

```
# 比如你发现你可以使用'A'去覆盖8个字节，然后跳转到0x114514地址就可以完成任务，那么你可以这么写你的payload并保存
padding = b"A" * 16
func1_address = b"\x14\x45\x11\x00\x00\x00\x00\x00"  # 小端地址
payload = padding+ func1_addr ess
# Write the payload to a file
with open("ans.txt", "wb") as f:
    f.write(payload)
print("Payload written to ans.txt")
#解释一下，为什么要将函数地址写成这个样子
#比方说你希望将字节0xA放在栈上时，如果你的txt文件是可见字符'A'的话，实际上放到栈上的是字节0x41(可见字符'A'对应的ASCll码值)
#但很明显，这不符合我们的预期，因此需要用关键字b去保证是0xA，比如b'A'，此时就是0xA，而不是可见字符'A'
#再之后就是地址的问题了，比如地址0x114514,由于大部分人的机器是小端存储，在python中最低有效字节应该放在前面，因此最后结果为上面代码的结果
#当然，如果你不喜欢python的话，可以尝试其他多种写法，只需要保证结果正确就行。
```

题目介绍

题目名称	保护类型	提示	注意事项	输出要求
Problem1	无	无	无	输出'Yes!I like ICS!'
Problem2	Nxenabled	注意传参方法与题目本身的代码片段	无	输出'Yes!I like ICS!'
Problem3	无	注意你能够使用的字节长度，以及你的栈地址变化情况	本题目难度较大，你可以在选择在gdb模式下输出正确结果或体现结果正确，除此之外，你可以根据你的策略去选择是否关闭内核全局栈随机化	输出幸运数字'114'
Problem4	Canary保护	想一想 你真的需要写代码吗	由于题目较为简单，请在报告中明确指出canary保护的机制以及是如何体现在汇编代码中的	输出通关提示

- 提示：请认真阅读汇编代码，所有的信息全在汇编代码中，我们基本上不会让你硬造函数。上述所要求输出的结果都被保存在汇编代码中，你要想办法“调出来”。

如何提交

1. 这次实验只需要提交报告。报告格式需按照./reports中的格式，将md文件和转换后的pdf文件放在./reports文件夹下提交。
2. 你需要在报告中体现你每道题目的思路是什么，payload是什么，攻击后的输出是什么。problem4请务必指出程序是在哪里设置canary保护的。

注意事项

- 不要自己尝试在自己的系统上做更多超出题目本身的操作，这并不安全。
- 学习和实践安全知识的目的是提高系统的安全性，而不是用于非法目的。
- 报告中参考的资料应放在报告后面
- 询问问题前请多思考，一些bug的问题可以自己百度或上csdn解决，你已经是一个成熟的计算机系大学生了，应该学会自己修bug了~
- 一些不懂的概念请自己百度、上csdn、上github或通过chatgpt进行解决，助教有时候回复的也不太准确。实在找不到答案后可以询问助教。
- 关于题目本身的思路请不要询问助教，你需要保证是自己独立完成实验。
- 如果思路出现堵塞或者无法理解题目，请移步到CTF wiki查看pwn题目中stack overflow的做法（https://ctf-wiki.org/pwn/linux/user-mode/stackoverflow/x86/stack-intro/）。
- ddl为一周左右，毕竟是baby-attack，相比于原汁原味的attacklab难度降低很多了。
- 实验流程同datalab保持一致，你需要从obe上接受作业链接。