缓冲区溢出漏洞实验 实验报告

实验目的：

理解缓冲区溢出原因，并通过实际操作向程序缓冲区写入精心设计且超长的内容，破坏堆栈段的存储实现程序的跳转，执行恶意代码达成攻击目的（夺取root权限）。

实验环境：

实验楼 courses 231

实验内容以及操作：

预备工作：安装32位编译环境和更新。

实验正式开始。

我们需要关闭Linux中堆与栈的地址空间随机化以猜测内存地址。执行命令：sudo sysctl -w kernel.randomize\_va\_space=0.

由于系统保护，调用shell的程序会自动放弃root权限。这个保护在 /bin/bash 中实现。我们调用另一个程序 zsh 来实现无此保护的情况。

进入32位环境，使用 /bin/bash。

编辑shellcode，并得到它的汇编版本（二进制码）：

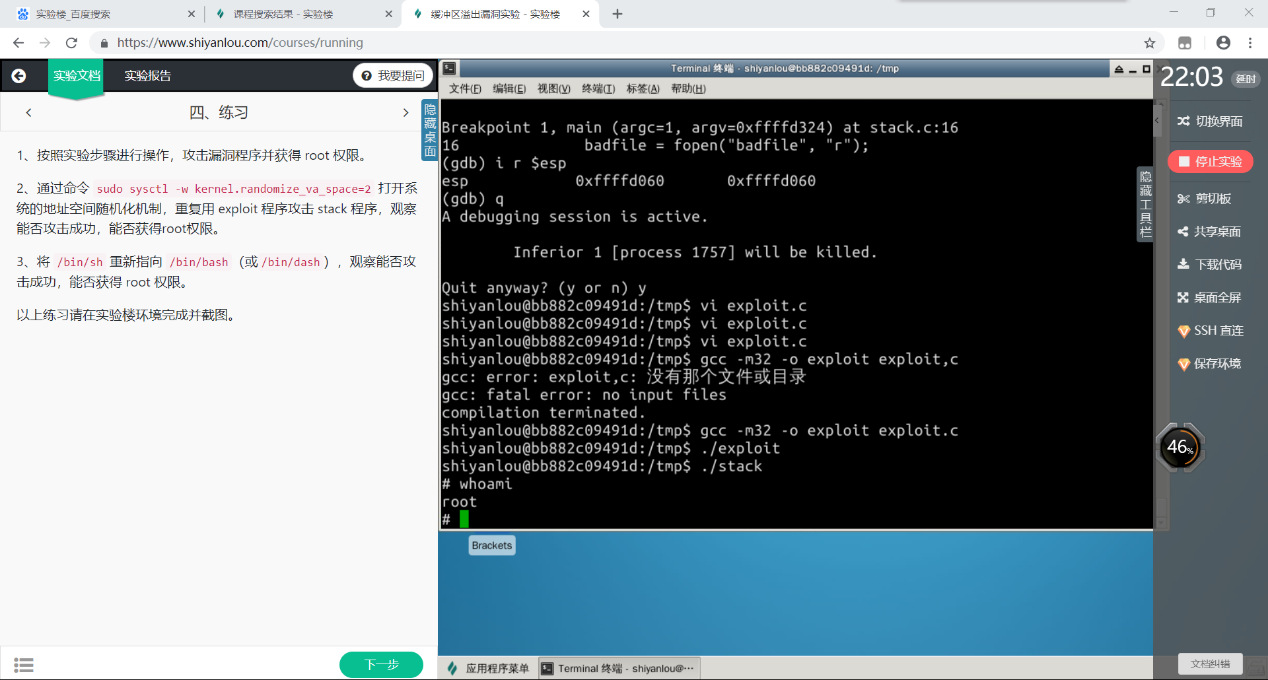
\x31\xc0\x50\x68"//sh"\x68"/bin"\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80”

编辑堆栈漏洞程序。基本操作为：从外部读取一个文件，将A文件中的第一个字符串写入str。调用子过程，将str的内容拷贝至一个远小于str的buffer中，实现堆栈段的覆盖。如果覆盖地恰到好处，就可以在子过程的返回地址位置跳出，并执行shellcode。

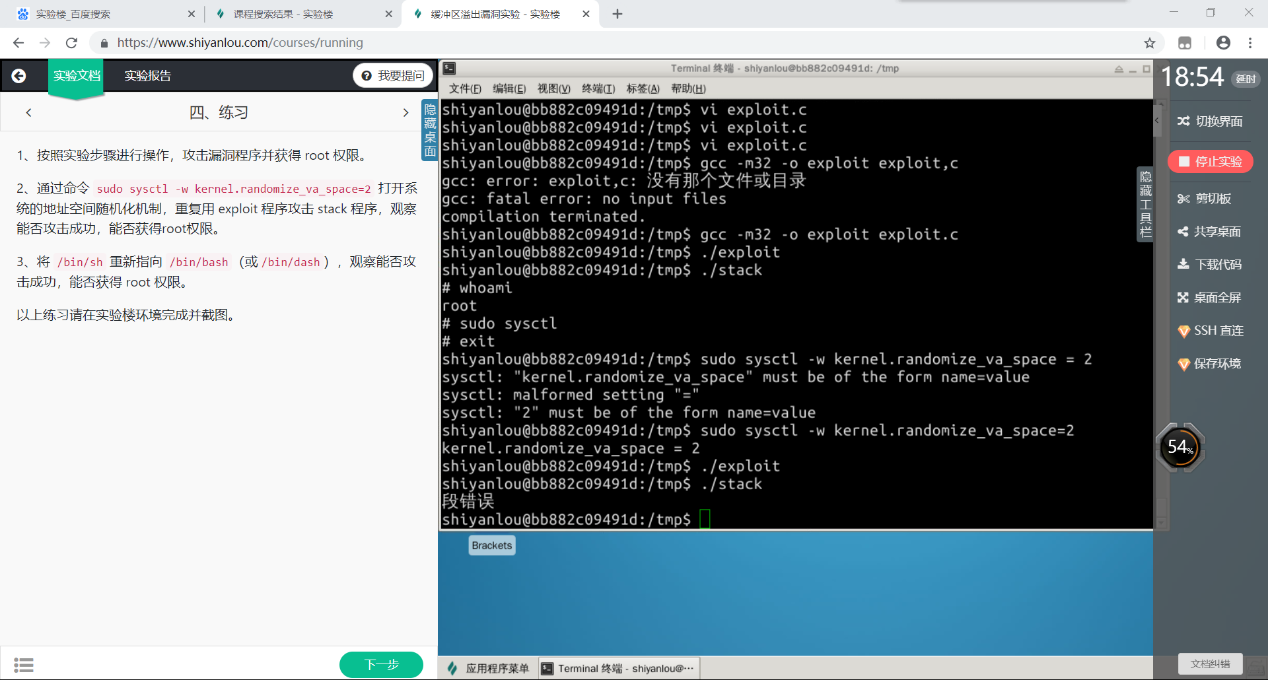
当然，我们需要在之后关闭堆栈内程序禁止执行的保护开关。

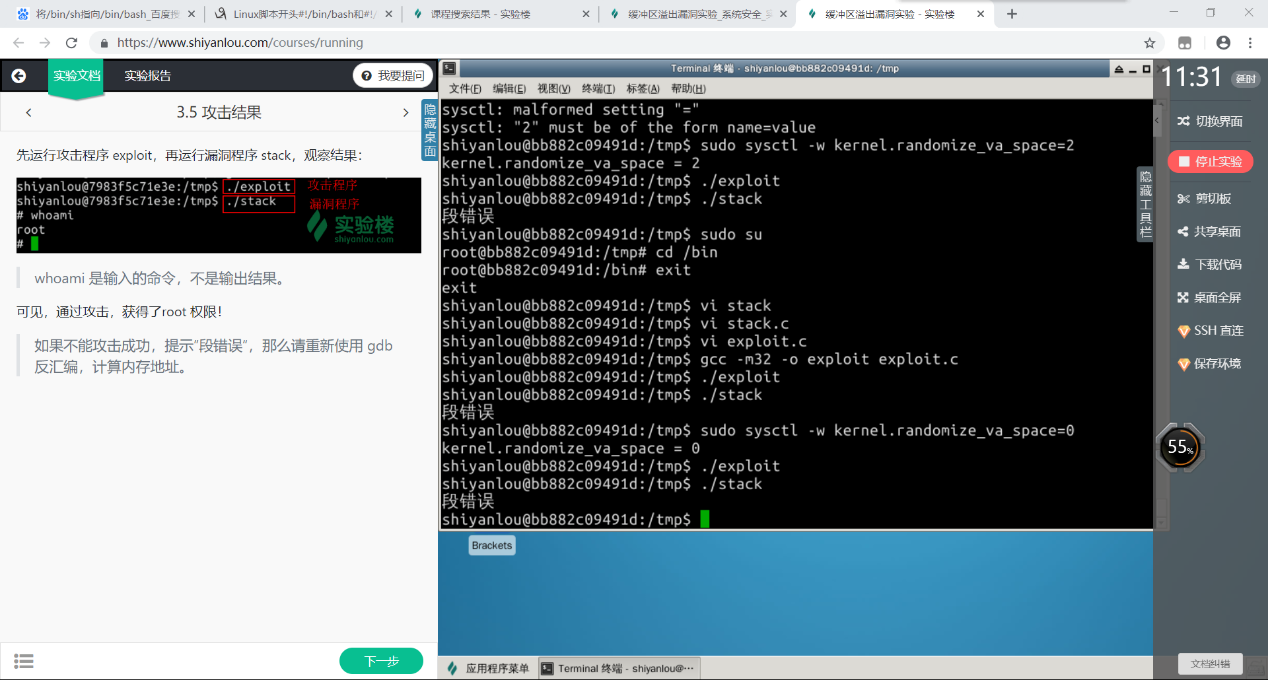
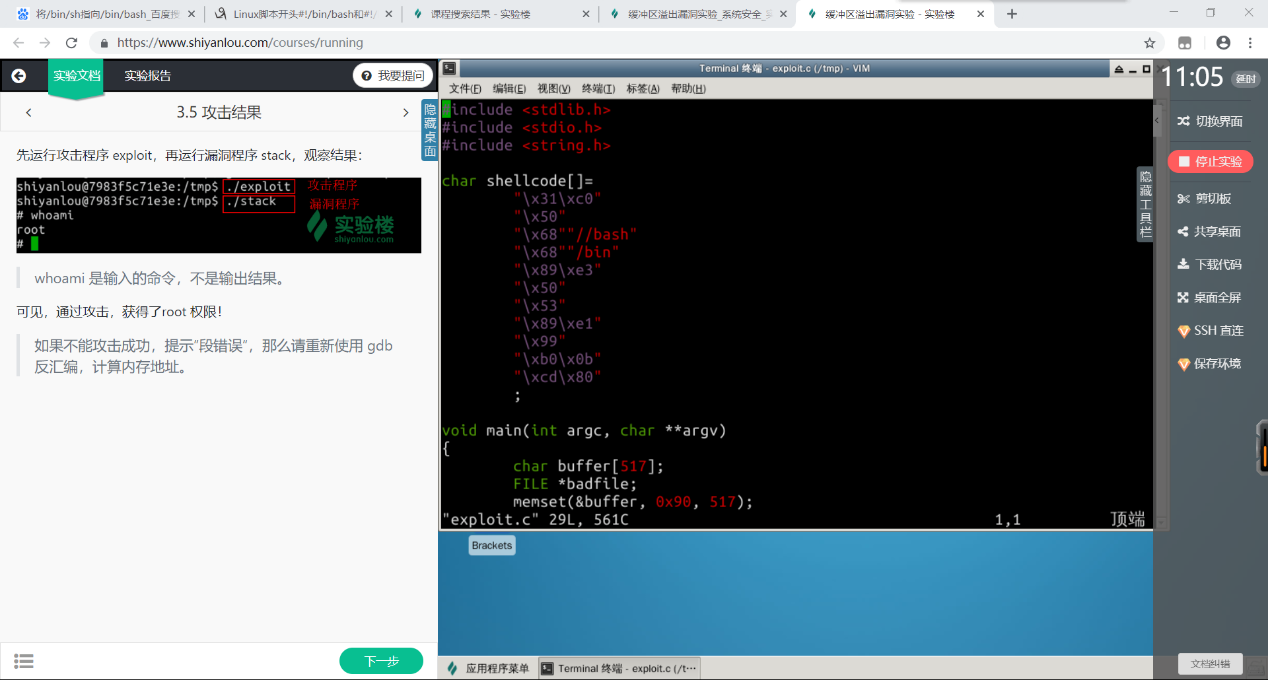
编辑堆栈漏洞程序的攻击程序。攻击程序很简单，就是向A文件写入一个精心设计的字符串。采用实验楼的方案，原堆栈漏洞程序子过程中buffer大小为12字节，再多覆写12字节（由漏洞程序写法得出），也就是第25字节开始覆盖子过程返回地址。只要将这个地址填写为shellcode的地址（在攻击程序中，为buffer+100的位置），就可以跳转并执行恶意代码。

现在我们要计算，shellcode 在内存中的位置。很简单，gdb调试堆栈漏洞程序，在合适的位置设置断点找到buffer的起始地址，增加100就是shellcode的位置，并将其写入字符串25字节开始的位置。所有代码至此准备完毕。

运行堆栈漏洞程序的攻击程序，再运行堆栈漏洞程序，效果如下：

至此，成功获得系统的root权限。

如果未关闭地址空间随机化，那么程序是无法进行攻击的。

如果 /bin/sh 被指向 /bin/bash，程序仍然无法进行攻击。

至此，实验完成。

实验结论：

这虽然是很久以前的攻击方法，但对于计算机底层的程序实现的理解仍然有着极大的帮助。至少学习程序底层运行之前，我是完全不了解这套理论。现在掌握程序运行与存储原理，通过程序内写入代码与跳转，能做到夺取root权限（关掉了一大堆开关的前提下……）。