

**实验报告**



**题目： 缓冲区溢出攻击实验**

**班 级： 2021211321**

**学 号： 2021212171**

**姓 名： 杨晨**

**学 院： 计算机学院**

**2022年 11 月 20 日**

一、实验目的  
1. C语言程序的机器级表示。  
2. 掌握GDB调试器的用法。  
3. C编译器生成的x86-64机器代码，理解不同控制结构生成的基本指令模式，过程的实现。

4. 掌握两种缓冲区攻击方法，进一步理解软件漏洞的危害。

1. 实验环境
2. 服务器：10.120.11.12（关机时间：2022年11月25日00：00）
3. Linux
4. Objdump命令反汇编
5. GDB调试工具
6. 积分榜：http://10.120.11.13:19340/scoreboard

三、实验内容

登录bupt1服务器，在home目录下可以找到一个targetn.tar文件，解压后得到如下文件：

README.txt；

ctarget；

rtarget；

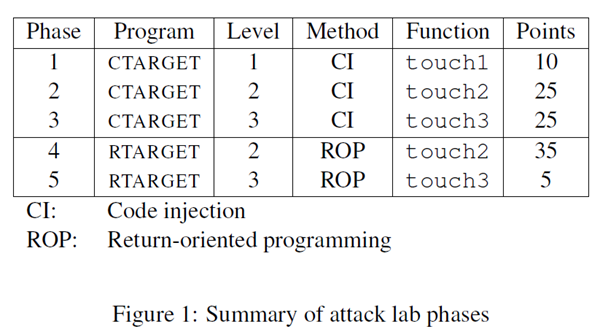
cookie.txt；

farm.c；

hex2raw。

ctarget和rtarget运行时从标准输入读入字符串，这两个程序都存在缓冲区溢出漏洞。通过代码注入的方法实现对ctarget程序的攻击，共有3关，输入一个特定字符串，可成功调用touch1，或touch2，或touch3就通关，并向计分服务器提交得分信息；通过ROP方法实现对rtarget程序的攻击，共有2关，在指定区域找到所需要的小工具，进行拼接完成指定功能，再输入一个特定字符串，实现成功调用touch2或touch3就通关，并向计分服务器提交得分信息；否则失败，但不扣分。因此，本实验需要通过反汇编和逆向工程对ctraget和rtarget执行文件进行分析，找到保存返回地址在堆栈中的位置以及所需要的小工具机器码。实验的具体内容见实验说明，尤其需要认真阅读各阶段的Some Advice提示。

本实验包含了5个阶段（或关卡），难度逐级递增。各阶段分数如下所示：



四、实验步骤及实验分析

建议按照：准备工作、阶段1、阶段2、…等来组织内容

各阶段需要有操作步骤、运行截图、分析过程的内容

**准备工作**

解压文件（**tar -xvf target556.tar**）

查看README文件的内容（**cat REANME.txt**）

文本

描述已自动生成

确认实验分为五个部分，1~3位于文件ctarget中，4~5位于文件rtarget中

**阶段一**

任务是让 CTARGET 在 getbuf 执行其返回语句时执行 touch1 的代码，而不是返回test

首先观察test函数的c代码

文本

中度可信度描述已自动生成

函数调用了getbuf函数，题目要求我们通过缓冲区注入的方式使getbuf执行后要返回到touch1函数中

touch1函数的c代码如下

文本

描述已自动生成

进入gdb调试（**gdb ctarget**）

文本

描述已自动生成

查看test的汇编代码（**disas test**）

图形用户界面, 文本

描述已自动生成

给rsp分配栈帧，接下来调用了getbuf函数，

使用disas命令，反汇编getbuf函数（**disas getbuf**）

文本

中度可信度描述已自动生成

分配了了40个字节的栈帧，之后栈顶位置rsp作为参数rdi，调用Gets函数读入字符串

反汇编touch1函数（**disas touch1**）

电脑萤幕画面

描述已自动生成

touch1函数的起始地址为0x4018c9，所以需要用0x4018c9来覆盖返回地址

图示

描述已自动生成

因此，我们可以输入48个字节，前40个字节将getbuf的栈空间填满，最后8个字节将返回地址覆盖为0x4018c9，即touch1的地址，这样在getbuf函数执行retq指令后，程序就会返回到touch1函数，执行touch1函数

由于x86-64机器采用小端法存储，因此需要在缓冲区填满后，用c9 18 40 00覆盖内存

创建attack1文件存储原始数据（**vi attack1**）

无效数据使用00填充

图片包含 户外, 街道, 绿色, 标志

描述已自动生成

采用管道输入，将数据通过hex2raw程序转化后输入ctarget程序（**./hex2raw <attack1> answer1**）

查看转化后的answer1文件（**cat answer1**）



进入gdb调试，注入攻击（**r < answer1**）

攻击成功

文本

描述已自动生成

**阶段二**

任务是让CTARGET执行touch2的代码，而不是返回test。但是，在这种情况下，还必须使其将 cookie 作为其参数传递。

首先观察touch2的c代码

手机屏幕的截图

描述已自动生成

除了需要改变函数的返回地址外，还要将cookie作为参数传递进去

反汇编touch2函数（**disas touch2**）

touch2的地址为0x4018fd

图形用户界面, 文本, 应用程序

描述已自动生成

Cookie的值为0x4c9688c3



容易想到，阶段二需要完成3步工作：

第一步，将cookie的值传给寄存器rdi，即将参数的值修改为cookie

第二步，在栈中压入touch2的代码地址

第三步，ret指令，调用返回地址，即执行touch2

新建一个汇编代码文件，写入汇编代码并保存（**vi code.s**）（**Esc ：wq**）

文本

描述已自动生成

反汇编getbuf函数（**disas getbuf**）

并在0x4018b7处设置断点，以便getbuf在分配栈帧后，栈顶指针rsp的值（**b \*0x00000000004018b7**）

文本

描述已自动生成

运行程序（**r**）

查看rsp的值（**p/x $rsp**）

得到栈顶指针的值0x556845a8，即为我们需要修改的返回地址

文本

描述已自动生成

这样，我们可以让程序继续回到栈顶0x556845a8处，执行我们所编写的代码

综上，阶段二的逻辑思路如下

1. getbuf将我们的代码读入，同时用0x556845a8覆盖掉返回地址

2. getbuf在ret的时候再次回到了我们注入代码的位置（图中绿色箭头）

3. 执行我们的代码，首先将cookiec传给寄存器rdi，然后将touch2的地址（0x4018fd）压入栈中

4. 再次ret的时候，从栈中弹出的将是0x4018fd，即touch2函数的地址，成功跳转到touch2函数

图示

描述已自动生成

可以使用 OBJDUMP 作为反汇编器，进而方便地生成指令序列的字节码。

编译code.s文件（**gcc -c code.s**）

目标文件code.o的反汇编结果输出到文件code.d中，同时显示源码（**objdump -d code.o > code.d**）



打开code.d（**vi code.d**）

文本

描述已自动生成

将这段代码放到attack2中的开头，代码地址放到末尾（**vi attack2**）

图片包含 键盘, 街道, 一群, 橙子

描述已自动生成

采用管道输入，将数据通过hex2raw程序转化后输入ctarget程序（**./hex2raw <attack2> answer2**）

查看转化后的answer1文件（**cat answer2**）

文本

描述已自动生成

注入攻击（**r < answer2**）

文本

描述已自动生成

**阶段三**

任务是让 CTARGET 执行touch3 的代码，而不是返回test。同时必须令touch3将 cookie 的字符串表示形式作为其参数传递

反汇编touch3函数（**disas touch3**）

图形用户界面, 文本

描述已自动生成

得到了touch3函数的地址0x401a16

注意到touch3函数调用了hexmatch函数，hexmatch的有两个参数，第一个参数是cookie，第二个参数是touch3接受到的参数

touch3的c代码如下

屏幕上有字

描述已自动生成

反汇编hexmatch函数（**disas hexmatch**）

文本

中度可信度描述已自动生成

haxmatch的c代码

手机的屏幕截图

描述已自动生成

注意到s的位置是随机的，所以我们在getbuf栈中写入的字符串有可能会被覆盖，而一旦被覆盖，那么无就无法完成正常匹配

考虑到我们需要把cookie转为字符串来进行匹配

因此，可以把cookie的字符串数据存储在test的栈上

反汇编test函数（**disas test**）

图形用户界面, 文本, 应用程序

描述已自动生成

在0x401a90处设置断点（**b \*0x0000000000401a90**）



运行（**r**）

查看rsp的值，得到test栈顶指针的位置0x556845d8（**p/x $rsp**）

文本

描述已自动生成

所以我们的cookie字符串应该放在0x556845d8

从之前的汇编代码，可以知道touch3的地址是0x401a16

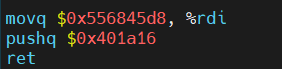
容易想到，阶段三需要完成3步工作：

第一步，将cookie串的值传给寄存器rdi，即将参数的值修改为cookie

第二步，在栈中压入touch3的代码地址

第三步，ret指令，调用返回地址，即执行touch3

注入代码如下



综上

阶段三的逻辑思路如下

1. getbuf读入后，将返回地址覆盖为0x556845a8，test栈顶位置处覆盖为cookie串的值

2. ret执行后，从栈中弹出返回地址，接下来执行我们的代码（绿色箭头处）

3. 先将存在0x556845d8处的cookie串传递给寄存器rdi，然后将touch3的地址压入栈中

4. 再次执行ret，从栈中弹出touch3的地址，成功跳转到touch3函数

图示

描述已自动生成

将汇编代码保存到code.s中（**vi code.s**）（**Esc :wq**）

使用 OBJDUMP 作为反汇编器生成指令序列的字节码。

编译code.s文件（**gcc -c code.s**）

目标文件code.o的反汇编结果输出到文件code.d中，同时显示源码（**objdump -d code.o > code.d**）

上述汇编代码的字节级表示如下

文本

描述已自动生成

Cookie的值0x4c9688c3作为字符串转为ascii码为：34 63 39 36 38 38 63 33

由于在test栈帧中利用了8个字节来存放cookie，所以需要输入56个字节

注入代码放在开头，41~48个字节放置注入代码的地址用来覆盖返回地址，最后8个字节存放cookie的ascii（**vi attack3**）

文本

描述已自动生成

采用管道输入，将数据通过hex2raw程序转化后输入ctarget程序（**./hex2raw <attack3> answer3**）

查看转化后的answer3文件（**cat answer3**）

文本

描述已自动生成

注入攻击（**r < answer3**）

文本

描述已自动生成

**阶段四**

对于阶段 4，需要重复阶段 2 的攻击，但是由于采用了栈随机化，即段程序分配的栈的位置在每次运行时都是随机的，这就使我们无法确定在哪里插入代码限制可执行代码区域，而且它限制栈上存放的代码是不可执行的

阶段四需要采用ROP攻击方式，即使用farm中的gadget在程序 RTARGET 上执行此操作（可以认为是“就地取材”）。可以使用由以下指令类型组成的gadgets来实现解决方案，并且仅使用前八个 x86-64 寄存器 （%rax~%rdi）。

movq：这些代码如图 3A 所示。

popq ：这些代码如图 3B 所示。

ret ：此指令由单字节0xc3编码。

nop ：此指令（发音为“no op”，是“no operation”的缩写）由单字节0x90编码。它的唯一效果是使程序计数器递增 1。

需要的所有小工具都可以在由函数start\_farm和mid\_farm划分的 rtarget 代码区域中找到

表格

描述已自动生成

我们在阶段二用到的注入代码为

图片包含 文本

描述已自动生成

显然，我们无法找到这样带有立即数的指令，因此需要考虑将cookie的值存放在栈中

首先，我们需要把cookie的值传递给寄存器rdi，可以用指令



但是在farm.c中找不到这两条指令，所以考虑使用其他寄存器进行中转

考虑2个gadget

手机屏幕的截图

描述已自动生成

查表，得知上述4条指令的机器代码依次为

58

c3

48 89 c7

c3

反汇编farm.c源文件，汇编代码保存到farm.s中

编译farm.s文件（**gcc -c farm.s**）

目标文件farm.o的反汇编结果输出到文件farm.d中，同时显示源码（**objdump -d farm.o > farm.d**）

文本

描述已自动生成

用vi编辑器打开farm.d文件（**vi farm.d**）

查找第一个指令58（**/58 Enter**）

手机屏幕的截图

描述已自动生成

发现指令在函数addval\_464中

调试rtarget（**gdb rtarget**）

反汇编addval\_464函数（**disas /r addval\_464**）

文本

描述已自动生成

得到指令的地址为0x401ae5

同理，在farm.d中查找指令48 89 c7（**/48 89 c7 Enter**）

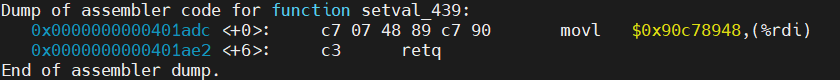
文本

中度可信度描述已自动生成

发现指令在setval\_439函数中

调试rtarget（**gdb rtarget**）

反汇编addval\_350函数（**disas /r setval\_439**）



得到指令的位置0x401ade

所以，阶段四的逻辑思路如下

1. getbuf读入数据，执行ret后跳转到gadget①

2. gadget①执行pop，将cookie串传给rax寄存器，然后ret跳转到gadget②

3. gadget②执行，将rax传给rdi，即完成了cookie的值传给rdi寄存器，然后ret跳转到touch2函数

图示, 示意图

描述已自动生成

所以注入的指令为（**vi attack4**）

电脑萤幕

低可信度描述已自动生成

采用管道输入，将数据通过hex2raw程序转化后输入ctarget程序（**./hex2raw <attack4> answer4**）

查看转化后的answer4文件（**cat answer4**）

文本

描述已自动生成

注入攻击（**r < answer4**）

文本

描述已自动生成

**阶段五**

本阶段要求对 RTARGET 进行 ROP 攻击，使用指向 cookie 字符串表示形式的指针，调用函数 touch3，和阶段三的工作相同

而在阶段三中用到的注入代码为

文本

中度可信度描述已自动生成

其中0x556845d8是栈中存放cookie的地址

由于这一阶段栈的位置是随机的，考虑获取rsp的地址，然后根据偏移量来确定存放cookie的地址

反汇编可执行程序rtarget（**objdump -d rtarget > rtarget.s**）



在rtarget.s中寻找我们需要的gadget

通过反复试探，最后我们需要的将cookie串传给rdi的指令如下

位置：0x401b0b

movq %rsp, %rax    48 89 e0

ret

位置：0x401ade

movq %rax, %rdi    48 89 c7

ret

位置：0x401ae5

popq %rax          58

ret

位置：0x401b99

movl %eax, %edx    89 c2

ret

位置：0x401b65

movl %edx, %ecx    89 d1

ret

位置：0x401bd1

位置：movl %ecx, %esi    89 ce

ret

位置：0x401af7

lea    (%rdi,%rsi,1),%rax

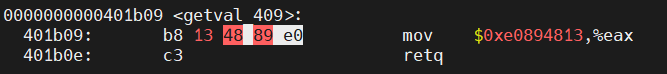
ret

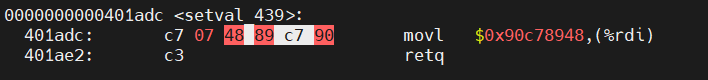
位置：0x401ade

movq %rax, %rdi    48 89 c7

ret

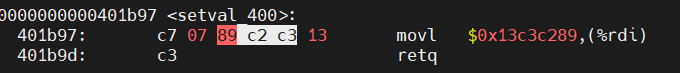
下面是一些查找截图

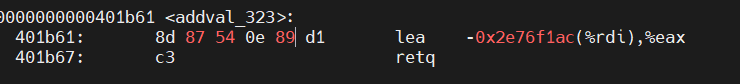


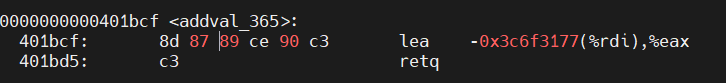


图形用户界面

低可信度描述已自动生成









综上，阶段五的逻辑思路如下

1. 首先获取到rsp的地址，并用寄存器rdi保存

2. 将偏移量0x48（十进制72）传给rax寄存器

3. 经过一系列中间寄存器，偏移量0x48最后传给了rsi寄存器

4. rax = rdi + 偏移量，即得到了cookie的地址

5. rax传给rdi，即将cookie串作为touch3的参数

6. 调用touch3函数

图示

描述已自动生成

注意，之所以偏移量是72（十进制）而不是80（十进制），是由于getbuf在执行完毕后，retq会令rsp = rsp + 8，所以我们第一次捕获到rsp的地址时候，实际上捕获到的是test栈顶位置+8的位置（图中黄色箭头处），因此从这里到cookie串的位置，需要加的偏移量是72

所以注入的指令为（**vi attack5**）

图片包含 图示

描述已自动生成

采用管道输入，将数据通过hex2raw程序转化后输入ctarget程序（**./hex2raw <attack5> answer5**）

查看转化后的answer5文件（**cat answer5**）

文本

描述已自动生成

注入攻击（**r <answer5**）

文本

描述已自动生成

五、总结体会

总结心得（包括实验过程中遇到的问题、如何解决的、过关或挫败的感受、实验投入的时间和精力、意见和建议等）

刚开始没有看说明文档直接开始做，导致没有思路，走了很多弯路。后来重新浏览了一遍说明思路，发现里面有很多提示和技巧，从而对整个实验有了一个清晰地认识。

对于每个阶段，首先需要理清思路，分步考虑，在构建输入字符的时候，可以采用画图的方法，便于理解。在输入的时候，要注意x86-64位机器是小端存储方法 。

通过整个实验，对缓冲区溢出溢出和函数调用的理解更加深入。知道了2种缓冲区攻击方式，也知道了如何预防，写出更加安全稳定的程序

意见和建议：

希望老师在讲这个实验的时候，将说明文档里的内容也一并介绍给同学们，或者告诉同学们要仔细看说明文档，这样在做实验的时候，会更容易上手。

六、诚信声明（不签扣10分）

需要填写如下声明，并在底部给出手写签名的电子版。

在完成本次实验过程中，我曾分别与以下各位同学就以下方面做过交流：

1、简单描述交流内容，例如：来自\*\*\*的建议，采用\*\*\*方式\*\*\*

2、全部独立完成

此外，我还参考了以下资料：

1. 网址等
2. 除说明文档外，未参考其他资料

在我提交的程序中，还在对应的位置以注释形式记录了具体的参考内容。

我独立完成了本次实验除以上方面之外的所有工作，包括分析、设计、编码、调试与测试。

我清楚地知道，从以上方面获得的信息在一定程度上降低了实验的难度，可能影响起评分。

我从未使用他人代码，不管是原封不动地复制，还是经过某些等价转换。

我未曾也不会向同一课程（包括此后各届）的同学复制或公开我这份程序的代码，我有义务妥善保管好它们。

我编写这个程序无意于破坏或妨碍任何计算机系统的正常运行。

我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按照0分计。

（签名）卡通人物

低可信度描述已自动生成