|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程编号 1502760001-07  题目类型 实验4 | |  |  |  | | --- | --- | --- | | **得分** | **教师签名** | **批改日期** | |  | 冯禹洪 |  | |

**深 圳 大 学 实 验 报 告**

**课 程 名 称： 计算机系统(2)**

**实验项目名称： 缓冲区溢出攻击实验**

**学 院： 计算机与软件学院**

**专 业： 软件工程（腾班）**

**指 导 教 师： 冯禹洪**

**报告人： 黄亮铭 学号： 2022155028 班级： 腾班**

**实 验 时 间： 2024年05月11日-2024年06月07日**

**实验报告提交时间： 2024年06月07日**

**教务处制**

**一、 实验目标：**

1. 理解程序函数调用中参数传递机制；
2. 掌握缓冲区溢出攻击方法；
3. 进一步熟练掌握GDB调试工具和objdump反汇编工具。

**二、实验环境：**

1. 计算机（Intel CPU）
2. Linux 64位操作系统
3. GDB调试工具
4. objdump反汇编工具

**三、实验内容**

本实验设计为一个黑客利用缓冲区溢出技术进行攻击的游戏。我们仅给黑客（同学）提供一个二进制可执行文件bufbomb和部分函数的C代码，不提供每个关卡的源代码。程序运行中有3个关卡，每个关卡需要用户输入正确的缓冲区内容，否则无法通过管卡！

要求同学查看各关卡的要求，运用**GDB调试工具和objdump反汇编工具**，通过分析汇编代码和相应的栈帧结构**，**通过缓冲区溢出办法在执行了getbuf()函数返回时作攻击，使之返回到各关卡要求的指定函数中。第一关只需要返回到指定函数，第二关不仅返回到指定函数还需要为该指定函数准备好参数，最后一关要求在返回到指定函数之前执行一段汇编代码完成全局变量的修改。

实验代码bufbomb和相关工具（sendstring/makecookie）的更详细内容请参考“实验四 缓冲区溢出攻击实验.pptx”。

本实验要求解决关卡1、2、3，给出实验思路，通过截图把实验过程和结果写在实验报告上。

|  |
| --- |
| 因为本次实验用到的可执行文件是32位，而实验环境是64位的，需要先安装一个32位的库，在root权限下安装如下所示：**因为lib32ncurses5安装过程中出现错误（无法找到相应文件），所以通过网络查阅得知可以在库名的后面添加-dev来安装，最后成功安装。**    还需要安装sendmail    首先利用反汇编命令查看getbuf函数的汇编代码，以便分析getbuf在调用<Gets>时的栈帧结构，汇编代码如下： |

**四、实验步骤和结果**

**步骤1 返回到smoke()**

* 1. **题目描述**

本实验中，bufbomb中的test()函数将会调用getbuf()函数，getbuf()函数再调用gets()从标准输入设备读入字符串。

系统函数gets()未进行缓冲区溢出保护。其代码如下：

|  |
| --- |
| *int getbuf()*  *{*  *char buf[12];*  *Gets(buf);*  *return 1;*  *}* |

我们的目标是使getbuf()返回时，不返回到test()，而是直接返回到指定的smoke()函数。

* 1. **解题过程**

|  |
| --- |
| 分析getbuf()函数的汇编代码，可以发现，首先压栈，然后将%esp减去0x28来分配额外的40个字节的地址空间。字符数组buf的位置用%ebp-24个字节来计算。然后调用Gets（）函数，读取的字符串返回到%ebp-24，字符串会向上占用空间。    我们只需要输入长度为32个字节的字符串即可更改返回地址（后4个字节影响返回地址，其余的无任何影响）。因此，**我们需要输入64个十六进制数，前56个数字任意填写，后8个数字需要填写smoke()函数的起始地址**。  由上述分析可知，我们现在需要知道的是smoke()函数的地址。使用命令查看smoke()函数的地址（见下图）。    然后我们新建一个txt文件，内容为任意56个数字加上刚刚查找到的smoke()函数的地址，考虑到x86架构为小端存储，因此地址需要按字节反序。    再输入命令确认文件中的内容是否正确。    最后通过sendstring将0.txt转换成二进制格式，再通过管道输入到bufbomb中。相应的命令为。    发现得到正确的结果。 |

* 1. **最终结果截图**

|  |
| --- |
| 通过上述分析进行操作，发现得到正确的答案，最终结果如下图所示。 |

**步骤2 返回到fizz()并准备相应参数**

**2.1 题目描述**

这一关要求返回到fizz()并传入自己的cookie值作为参数，破解的思路和第一关是类似的，构造一个超过缓冲区长度的字符串将返回地址替换成fizz()的地址，只是增加了一个传入参数，所以在读入字符串时，要把fizz()函数读取参数的地址替换成自己的cookie值。

**2.2 解题过程**

|  |
| --- |
| 首先利用objdunp查看并分析fizz()函数的汇编代码：    从汇编代码和第一个任务中地分析可知，栈帧的结构和fizz()函数读取的参数的位置（%ebp+8）。    我们只需要输入长度为40个字节的字符串即可更改返回地址和参数值（第40个字节到第43个字节影响返回地址，后4个字节影响参数值，其余的无任何影响）。因此，**我们需要输入80个十六进制数，前56个数字任意填写，接下来的8个数字需要填写smoke()函数的起始地址，再随意填写8个数字，最后填写自己名字的cookie值**即可。  由上述分析可知，我们此时需要知道fizz()函数的地址，使用命令得到fizz()函数的起始地址。    然后使用makecookie工具得到我的名字的cookie。相应的命令为。    再新建一个txt文件，内容为任意56个数字加上刚刚查找到的fizz()函数的地址，再加上8个任意数字和上面得到的cookie值，考虑到x86架构为小端存储，因此地址和cookie值需要按字节反序。    其次输入命令确认文件中的内容是否正确。    最后通过sendstring将0.txt转换成二进制格式，再通过管道输入到bufbomb中。相应的命令为。    发现得到正确的结果。 |

**2.3 最终结果截图**

|  |
| --- |
| 通过上述分析进行操作，发现得到正确的答案，最终结果如下图所示。 |

**步骤3 返回到bang()且修改global\_value**

* 1. **题目描述**

这一关要求先修改全局变量global\_value的值为自己的cookie值，再返回到band()。为此需要先编写一段代码，在代码中把global\_value的值改为自己的cookie后返回到band()函数。将这段代码通过GCC产生目标文件后读入到buf数组中，并使getbuf函数的返回到buf数组的地址，这样程序就会执行我们写的代码，修改global\_value的值并调用band()函数。

* 1. **解题过程**

|  |
| --- |
| 首先，为了能精确地指定跳转地址，先在root权限下关闭Linux的内存地址随机化：    用objdump查看bang()函数的汇编代码（见下图）。bang()函数首先读取0x804a1c4和0x804a1d4的地址的内容并进行比较，要求两个地址中的内容相同，否则会给出错误信息：    用gdb调试命令查看相应地址的内容，发现两个位置分别被打上了global\_value、cookie的标签。至此，我们的第一个任务内容明了：**自行编写代码，将地址0x804a1d4的内容存储到地址0x804a1c4中**。    利用objdump得到bang()函数的入口地址为0x08048e10:    现在，我们知道了该任务的全部内容。**首先是将global\_value的值设置为cookie的值，然后将bang()函数的入口地址压入栈中。当函数返回的时候，会直接取栈顶作为返回地址，从而调用bang()函数**。  我们输入的字符串应该为一段汇编代码加上随意的一段数字再加上字符串缓冲区的首地址（buf的首地址）。**栈帧结构同任务1。**  根据上述分析，首先创建文本文件insert.txt，内容如下图，然后将其后缀名修改为.s的汇编代码文件。    然后将其编译为可重定位目标文件，并通过objdump工具进行反汇编，得到结果（操作及结果见下图）。    我们只需要将对应的二进制代码输入到字符缓冲区中即可。此时，返回地址应该被重写为buf开始的地址，为此我们使用GDB调试工具查找buf的起始地址。由关卡1分析可知，buf的起始地址为%ebp-0x18，即：0xffffb1e0。    然后我们新建一个txt文件，内容为上述二进制代码加上任意12个数字加上刚刚查找到的buf的地址，考虑到x86架构为小端存储，因此地址需要按字节反序。    再输入命令确认文件中的内容是否正确。    最后通过sendstring将0.txt转换成二进制格式，再通过管道输入到bufbomb中。相应的命令为。    最后发现答案正确。 |

**3.3 最终结果截图**

|  |
| --- |
| 通过上述分析进行操作，发现得到正确的答案，最终结果如下图所示。 |

**五、实验总结与体会**

1. 通过本次实验，我理解了程序函数调用中参数传递机制。
2. 通过这次实验，我对于数据溢出有了更深刻的了解，知晓了黑客是如何通过注入字符串，修改函数返回地址，跳转到想要运行的函数的入口，从而进行攻击的。
3. 通过本次实验，我明白了我们应该在读入数据的时候多加注意缓冲区溢出的情况，防范缓冲区溢出攻击。
4. 此外，通过本实验，我进一步熟练掌握GDB调试工具和objdump反汇编工具。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字：冯禹洪    2024年 月 日 |
| 备注： |