**电子科技大学信息与软件工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 网络安全技术**

**电子科技大学教务处制表**

**实 验 报 告**

**学生姓名：陆圣珩 学 号： 2017221302009 指导教师：赵洋**

**实验地点：信软学院楼西303 实验时间：11-21**

**一、实验室名称：**

**二、实验项目名称：缓冲区溢出实验**

**三、实验学时： 4 学时**

**（一）实验目的**

了解缓冲区溢出的相关概念；明确缓冲区溢出的危害；理解栈溢出、堆溢出、整型溢出、格式化字符串溢出及文件流溢出的原因；掌握安全编程技术。C/C++之类的高级语言通过指针直接对内存地址中所蕴藏的数据进行修改，因此可以利用C语言修改堆栈内容而修改程序的执行方式。本实验通过对缓冲区溢出利用的实践，理解缓冲区溢出的利用方法和带来的危害后果。

**（二）实验内容**

1. 初步认识栈溢出实验
2. 利用栈溢出漏洞实验
3. 溢出编程实验

**四、实验原理**

缓冲区溢出是目前最常见的一种安全问题，操作系统以及应用程序大都存在缓冲区溢出漏洞。缓冲区是一段连续内存空间，具有固定的长度。缓冲区溢出是由编程错误引起的，当程序向缓冲区内写入的数据超过了缓冲区的容量，就发生了缓冲区溢出，缓冲区之外的内存单元被程序“非法”修改。

一般情况下，缓冲区溢出导致应用程序的错误或者运行中止，但是，攻击者利用程序中的漏洞，精心设计出一段入侵程序代码，覆盖缓冲区之外的内存单元，这些程序代码就可以被CPU所执行，从而获取系统的控制权。

目前，操作系统（Windows、Linux、Unix）、数据库以及应用软件主要采用C/C++语言开发，但C/C++语言缺乏数组边界条件检查、程序执行不受控制等特点，因此，这些软件不可避免地存在缓冲区溢出漏洞，成为安全隐患。

**五、实验器材（设备、元器件）**

1. 实验人数50～80人，每人1台计算机；独立完成本实验。
2. 拓扑：（A、B范围中的主机分别简称为A主机和B主机）



**A**

**B**

1. 设备：以太网交换机2～4台；计算机50～80台
2. 软件：Visual Studio软件

**六、实验步骤**

1. **栈溢出的跟踪与解析实验**
2. 使用VS打开stack.dsw文件。
3. 在push语句前设置断点，按F5键进入调试界面。
4. 打开Watch、Variables、Registers、Memory、Disassembly五个窗口。
5. 记录EAX、ESP、EBP、EIP的值，并计算栈的大小为多少字节？
6. 按F11键执行push语句，观察栈、 ESP和EIP的值是否变化？为什么？
7. 继续执行pop语句，观察EAX、ESP、EIP的值是否变化？为什么？
8. 继续执行下面的语句，观察寄存器值的改变，最后按shift+F5结束调试。
9. 使用VS打开hanshu.dsw文件。
10. 在ourfunction语句前设置断点，按F5键进入调试界面。
11. 记录ESP、EBP、EIP的值，按F11键执行ourfunction语句，观察ESP、EIP的值是否变化，以及现在栈顶存放的4个字节是什么地址？
12. 继续执行，直到“int our=0”语句，观察每一步寄存器值的改变。
13. 执行“int our=0”语句，观察0值被存放在栈中的哪个位置？为什么？
14. 继续执行，直到“ret”语句，记录ESP，EBP的值。
15. “ret”语句执行后，什么值弹出到EIP，这时程序跳转到什么位置？ESP和EBP的值是否和步骤10时相同？
16. 按shift+F5结束调试。
17. 使用VS打开overflow.dsw文件。
18. 在“char longbuf[100]=……”语句前设置断点，F5调试，F11执行。
19. 记录EAX、EBP、ESP的初始值，查看longbuf的存放地址，计算此地址与EBP相差多少字节？
20. 继续执行“overflow(longbuf)”直到“push eax”语句后，EBP、ESP、EAX的值是否变化？栈顶现在存放的是什么数据？

lea是将源操作数的地址传到目的操作数中，那么“lea eax,[ebp-64h]”是将什么地址赋给了EAX寄存器？

1. 继续执行到“strcpy (des,buf)”指令后，mov是将数据从源操作数传到目的操作数中，那么“mov edx, dword ptr[ebp+8]”是将什么数据传到EDX中？

“lea eax,[ebp-8]”又是将什么数据传到EAX中？

1. 继续执行到“ret”语句，查看将弹出什么值到EIP中？
2. 利用栈溢出漏洞实验
3. 使用VS打开liyong.dsw文件。
4. 在“char longbuf[100]=……”语句前设置断点，F5调试，执行完赋值语句后查看longbuf的存放起始地址和结束地址？记录EBP、ESP的值。
5. 继续执行，“lea eax,[ebp-5fh]；push ecx”是将什么地址压栈？
6. “call @ILT+1175(overflow)”语句是调用overflow函数，在这之前要将EIP入栈，这时记录压入堆栈的EIP值是多少，存放在哪个地址处？
7. 调用overflow函数后，新分配给它的栈顶和栈底地址分别是多少？栈的大小是多少？
8. “char des[5]=””语句执行完后，是将那段地址分配给des变量，大小是多少字节？
9. 继续执行”strcpy(des,buf)”语句，mov语句是将什么值传递给edx? Lea语句是将什么值传递给eax?
10. Strcpy完成后，查看des变量空间是否有改变？
11. Ret语句执行完后，查看EBP，ESP，EIP的值分别是多少？程序将跳到哪里执行？
12. **溢出编程实验**

1、试自己编写一个缓冲区溢出的程序，通过溢出覆盖返回地址，从而跳转到一个指定的程序。运行结果截图。

参考代码：

#include <stdio.h>

#include <stdlib.h>

void why\_here(void)

{

printf("why u here !n\n");

printf("you are traped here\n");

system("pause");

\_exit(0);

}

int main(int argc,char \* argv[])

{

int buff[1];

buff[2] = (int)why\_here;

system("pause");

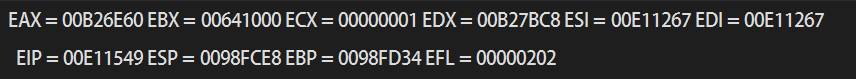
return 0;

}

1. 试自己编写一个缓冲区溢出的程序，通过溢出覆盖返回地址和植入Shellcode，从而跳转到Shellcode执行，给出源码和运行结果截图。（附加题）

**七、实验数据及结果分析：**

1. **栈溢出的跟踪与解析实验**
2. 使用VS打开stack.dsw文件。
3. 在push语句前设置断点，按F5键进入调试界面。
4. 打开Watch、Variables、Registers、Memory、Disassembly五个窗口。
5. 记录EAX、ESP、EBP、EIP的值，并计算栈的大小为多少字节？



EAX=00B26E60 ESP=0098FCE8 EBP=0098FD34 EIP=00E11549

栈大小=EBP-ESP=4C

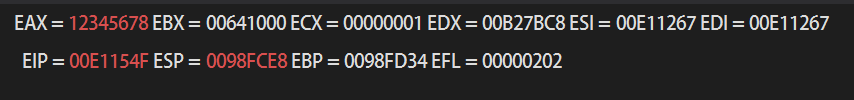
1. 按F11键执行push语句，观察栈、 ESP和EIP的值是否变化？为什么？



发生了变化

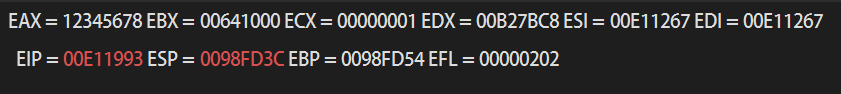
EIP存储着下一条指令的地址，每执行一条指令，该寄存器变化一次，而ESP指向栈顶，当压栈或出栈时，栈顶发生变化，ESP也就发生变化

1. 继续执行pop语句，观察EAX、ESP、EIP的值是否变化？为什么？

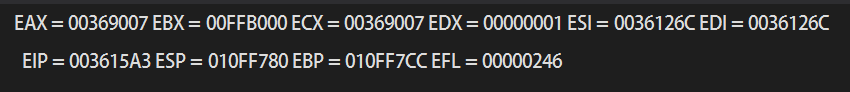
 发生了变化

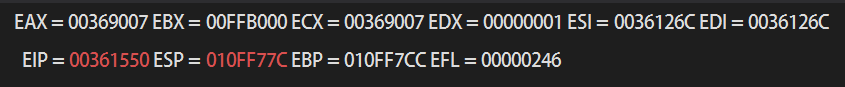
EAX是返回值，在函数结束时返回值改变

1. 继续执行下面的语句，观察寄存器值的改变，最后按shift+F5结束调试。

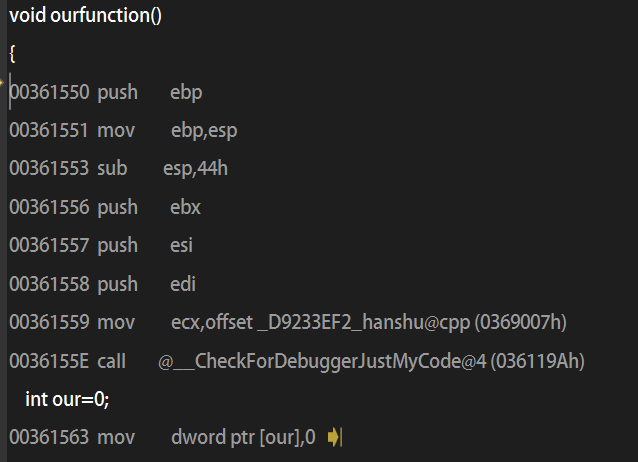


1. 使用VS打开hanshu.dsw文件。
2. 在ourfunction语句前设置断点，按F5键进入调试界面。
3. 记录ESP、EBP、EIP的值，按F11键执行ourfunction语句，观察ESP、EIP的值是否变化，以及现在栈顶存放的4个字节是什么地址？

按F11后：

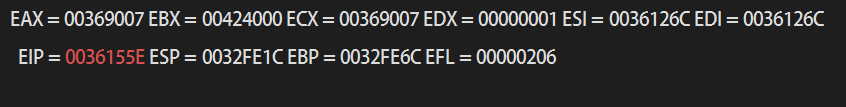


ESP、EIP发生变化，栈顶存放的四个字节是下一个要执行的语句的地址

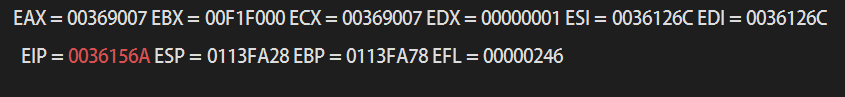


1. 继续执行，直到“int our=0”语句，观察每一步寄存器值的改变。

(因为多次尝试寄存器值与上图有所变化)

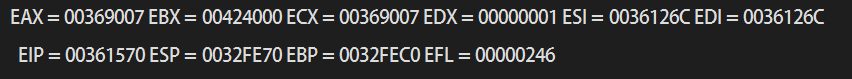


1. 执行“int our=0”语句，观察0值被存放在栈中的哪个位置？为什么？

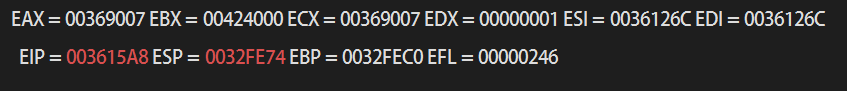




1. 继续执行，直到“ret”语句，记录ESP，EBP的值。

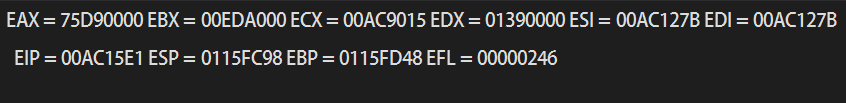
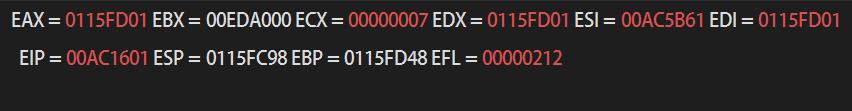


1. “ret”语句执行后，什么值弹出到EIP，这时程序跳转到什么位置？ESP和EBP的值是否和步骤10时相同？

将要执行的下一条语句弹出到EIP，程序跳转到主函数结尾

此时ESP和EBP和步骤10时相同（因为多次执行步骤12导致截图有变化）

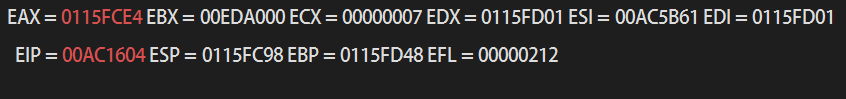
1. 按shift+F5结束调试。
2. 使用VS打开overflow.dsw文件。
3. 在“char longbuf[100]=……”语句前设置断点，F5调试，F11执行。
4. 记录EAX、EBP、ESP的初始值，查看longbuf的存放地址，计算此地址与EBP相差多少字节？

longbuf的存放地址为ESI （00AC5B61）

此地地址与EBP相差69A1E7

1. 继续执行“overflow(longbuf)”直到“push eax”语句后，EBP、ESP、EAX的值是否变化？栈顶现在存放的是什么数据？



lea是将源操作数的地址传到目的操作数中，那么“lea eax,[ebp-64h]”是将什么地址赋给了EAX寄存器？

是将数组元素所在栈的最后一位地址赋值给了EAX，即该数组的栈底地址

1. 继续执行到“strcpy (des,buf)”指令后，mov是将数据从源操作数传到目的操作数中，那么“mov edx, dword ptr[ebp+8]”是将什么数据传到EDX中？

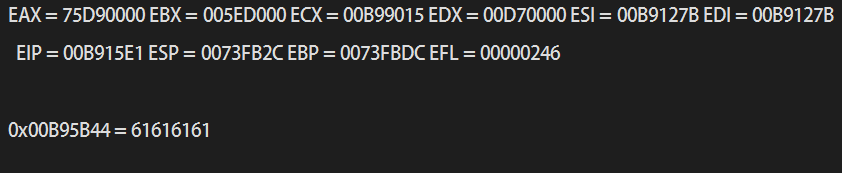
“lea eax,[ebp-8]”又是将什么数据传到EAX中？

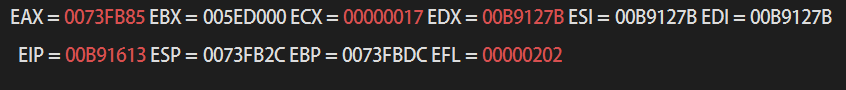
mov edx, dword ptr[ebp+8]是将ECX的地址传到EDX中

1. 继续执行到“ret”语句，查看将弹出什么值到EIP中？

EIP变为62626262，即bbbb

1. 利用栈溢出漏洞实验
2. 使用VS打开liyong.dsw文件。
3. 在“char longbuf[100]=……”语句前设置断点，F5调试，执行完赋值语句后查看longbuf的存放起始地址和结束地址？记录EBP、ESP的值。

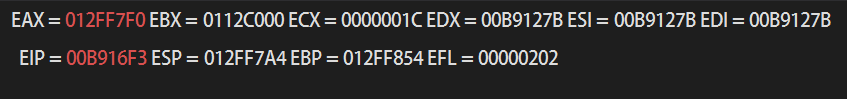




1. 继续执行，“lea eax,[ebp-5fh]；push ecx”是将什么地址压栈？

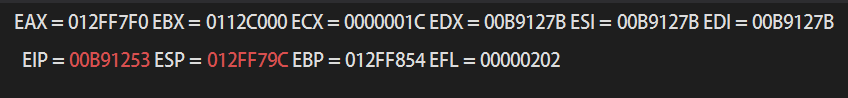
将保存的要初始化数组的内容地址压栈

1. “call @ILT+1175(overflow)”语句是调用overflow函数，在这之前要将EIP入栈，这时记录压入堆栈的EIP值是多少，存放在哪个地址处？



存放在函数局部变量申请栈空间的后四个位置

1. 调用overflow函数后，新分配给它的栈顶和栈底地址分别是多少？栈的大小是多少？



新分配的栈顶地址即为ESP存储的地址：012FF79C

新分配的栈底地址即为EBP存储的地址：012FF854

栈的大小为48h

1. “char des[5]=””语句执行完后，是将那段地址分配给des变量，大小是多少字节？

5个字节

1. 继续执行”strcpy(des,buf)”语句，mov语句是将什么值传递给edx? Lea语句是将什么值传递给eax?

第一个mov指令是将函数的全局变量地址传入eax

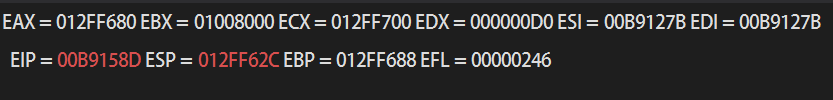
第二个lea指令是将函数的局部变量地址传入ecx

1. Strcpy完成后，查看des变量空间是否有改变？



改变了

1. Ret语句执行完后，查看EBP，ESP，EIP的值分别是多少？程序将跳到哪里执行？



EBP的值为 “aaaa”的ASCII码。

ESP的值为溢出后地址处。

EIP的值为初始化量后面的值

程序将会跳转到EIP指向的地方运行。

1. **溢出编程实验**
2. 试自己编写一个缓冲区溢出的程序，通过溢出覆盖返回地址，从而跳转到一个指定的程序。运行结果截图。

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  void test()  {  printf("溢出了！");  return;  }  int main(void)  {  int str[10] = { 0 };  str[11] = (int)test;  return 0;  } |

**八、实验结论、心得体会和改进建议：**

本次实验，在 VS2019 的帮助下，理解并尝试操作了 windows 下汇编语言的实现过程，深刻理解了 EAX EBX ESI EIP ESP EBP以及栈等常用概念，并学习了栈溢出的基本原理。通过尝试并亲自编写示例性栈溢出程序，更好的掌握了栈溢出的基本原理，对软件安全技术这一部分的内容有了深刻的理解。

**九、对本实验过程及方法、手段的改进建议**

**无**

**报告评分：**

**指导教师签字：**