# 《漏洞利用及渗透测试基础》实验报告

林晖鹏 2312966

March 2025

## 1 实验名称

IDE 反汇编实验

## 2 实验要求

根据第二章示例 2-1，在 XP 环境下进行 VC6 反汇编调试，熟悉函数调用、栈帧切换、CALL 和 RET 指令等汇编语言实现，将 call 语句执行过程中的 EIP 变化、ESP、EBP 变化等状态进行记录，解释变化的主要原因。

## 3 实验过程

1. 进入 VC 反汇编

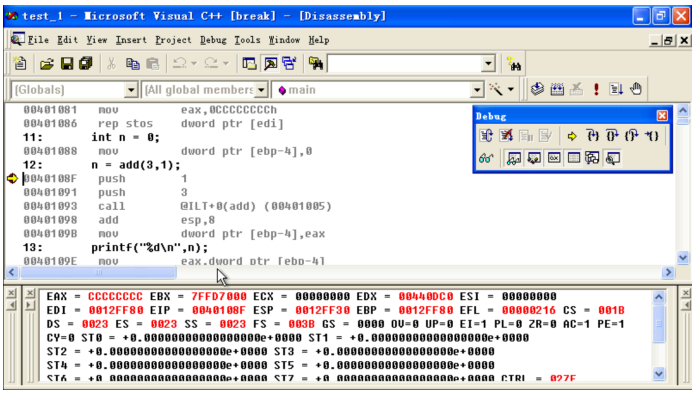


图 1: 反汇编界面

2. 观察 add 函数调用前后语句下面是调用 add 函数前后的汇编语句，并注释写上分析。

```
12: n = add(3,1);
0040108F push 1  //从右到左将参数压入栈
00401091 push 3
00401093 call @ILT+0(add) (00401005)  //调用 add 函数，函数返回地址入栈
00401098 add esp,8                    //esp 栈帧切换回主函数
0040109B mov dword ptr [ebp-4],eax  //将返回值赋值给 n 的地址
}
```

3. add 函数内部栈帧切换等关键汇编代码

下面来分析 add 函数内部的汇编代码，先跳转指令执行区域。

```
@ILT+0(?add@@YAHHH@Z):
00401005 jmp add (00401030)
```

然后到 add 函数位置执行指令，下面是汇编代码以及分析

```
#include <iostream>
int add(int x, int y)
{
401030 push ebp  //将 ebp 地址压入栈进行保存
401031 MOV ebp,esp  //将 ebo、esp 位置转移到 add 函数栈帧上
401033 sub esp, 44h
401036 push ebx  //保存现场
401037 push esi
401038 push edi
401039 lea edi, [ebp-44h]
40103C MOV ecx, 11h
401041 M  eax, OCCCCCCCCh
401046 rep stos dword ptr [edi]  //循环对栈区数据初始化
 int z = 0;
401048 MOV dword ptr [ebp-4],0  //给 z 初始化为 0
  z = x+y;
40104F mov eax,dword ptr [ebp+8]  //将 x 赋值到 eax 寄存器上
401052 add eax,dword ptr [ebp+0Ch]  //将 y 加到 eax 上
401055 mov dword ptr [ebp-4],eax  //将 eax 的值赋值给 z
```

```
20   return z;
21 401058 mov eax,dword ptr [ebp-4]  //将 z 的值存到寄存器 eax 中
22  }
23 40105B pop edi  //下面都是恢复现场
24 40105C pop esi
25 40105D pop ebx
26 40105E mov esp,ebp
27 401060 pop ebp
28 401061 ret //返回主函数
```

4. 重点分析并解释 EIP、ESP、EBP 的变化。

   **EIP**:

   在 add 函数调用前的 call 部分，跳跃寻址将 EIP 转移到 jump 处；然后在执行 jump 后，EIP 再次跳跃寻址转移到 add 函数的地址处；最后 ret 处再次跳跃寻址回到主函数 add 调用的 1 地方。在其余部分 EIP 都是顺序寻址。

   **ESP、EBP**:

   在调用前将指针压入栈以及保存 EBP 地址的时候，ESP 都往低地址扩展；然后再将 ESP 赋值给 EBP，此时 EBP 高地址方向的都是函数的初始参数，低地址方向都是局部变量。

   再对 ESP 往低地址扩展栈区内存，用来存函数调用时产生的局部变量。

   在函数执行完之后，将局部变量 pop 清空，然后再将 ESP 变回此时 EBP 的地址，再 pop 之前压入栈的 ebp（同时 EBP 恢复之前地址）以及加八个字节（因为这里存了两个形参 x、y）。

# 4 心得体会

- 通过实验，掌握了如何查看函数汇编代码，并通过 debug 逐步查看代码执行。
- 对 EIP、ESP、EBP 的理解加深，进一步熟悉了在函数执行基本流程中，这三个寄存器的变化。
- 基本掌握学习了基础汇编代码的知识，能看懂汇编代码并理解使用。