《漏洞利用及渗透测试基础》实验报告

姓名：梁婧涵 学号：2112155 班级：1120

实验名称：IDE反汇编实验

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。

实验过程：

1. 进行VC反汇编

打开反汇编窗口，在函数调用的位置下断点。运行程序，等待程序运行到断点处。

1. 观察add函数调用前后语句

查看当前指令，如果是一个Call指令，则说明当前函数正在被调用。在执行Call指令之前，当前函数的下一条指令的地址会被压入堆栈，这个地址即为当前函数执行完毕之后要返回的地址，这个地址可以通过ESP寄存器得到。Call指令执行时，EIP寄存器中的值会被压入堆栈，保存当前函数执行的下一条指令的地址。Call指令会跳转到被调用函数的第一条指令处执行。

1. Add函数内部栈帧切换等关键汇编代码：

在被调用函数执行的第一条指令处，当前函数的EBP值会被压入堆栈，作为当前函数的栈帧指针，指向当前函数的栈帧。在被调用函数执行的第一条指令处，ESP的值会被减去当前函数需要的栈空间，用于为当前函数的局部变量和参数分配空间。在被调用函数中，EBP指向当前函数的栈帧指针，ESP指向当前函数的栈顶。当被调用函数执行完毕，使用RET指令返回到调用函数的下一条指令处时，ESP的值会恢复为调用函数时的值，即当前函数的下一条指令的地址。

心得体会：

通过实验，掌握了RET指令的用法；RET指令实际就是执行了Pop EIP；此外，通过本实验，掌握了多个汇编语言的用法。

具体来说，在调用函数时，程序会将函数的返回地址压入栈中，同时将当前栈顶指针（ESP）向下移动，以便为函数调用分配局部变量的空间。函数调用结束后，程序会使用返回地址将EIP寄存器的值设置为返回地址，并将ESP和EBP寄存器恢复为调用函数之前的值。

因此，在执行call语句时，EIP会指向函数的入口地址，ESP会向下移动，EBP则不会变化。在函数执行过程中，ESP会继续向下移动以分配局部变量的空间，并在函数返回时恢复为调用函数之前的值。在函数返回时，程序会使用返回地址将EIP寄存器的值设置为调用函数的下一条指令，同时恢复ESP和EBP寄存器的值。

总之，在调用函数时，程序会将返回地址压入栈中，并分配局部变量的空间；在函数返回时，程序会使用返回地址将EIP寄存器的值设置为调用函数的下一条指令，并恢复ESP和EBP寄存器的值。通过记录这些寄存器的变化，我们可以了解程序的执行过程和堆栈的变化情况。