1. 缓冲区溢出攻击：

目标：

在正常情况下，main 函数和 overflow 函数中均不存在对 fun 函数的调用。通过对函数 overflow 输入的溢出攻击，使得在函数 overflow 返回时，函数 fun 被调用。

原理：

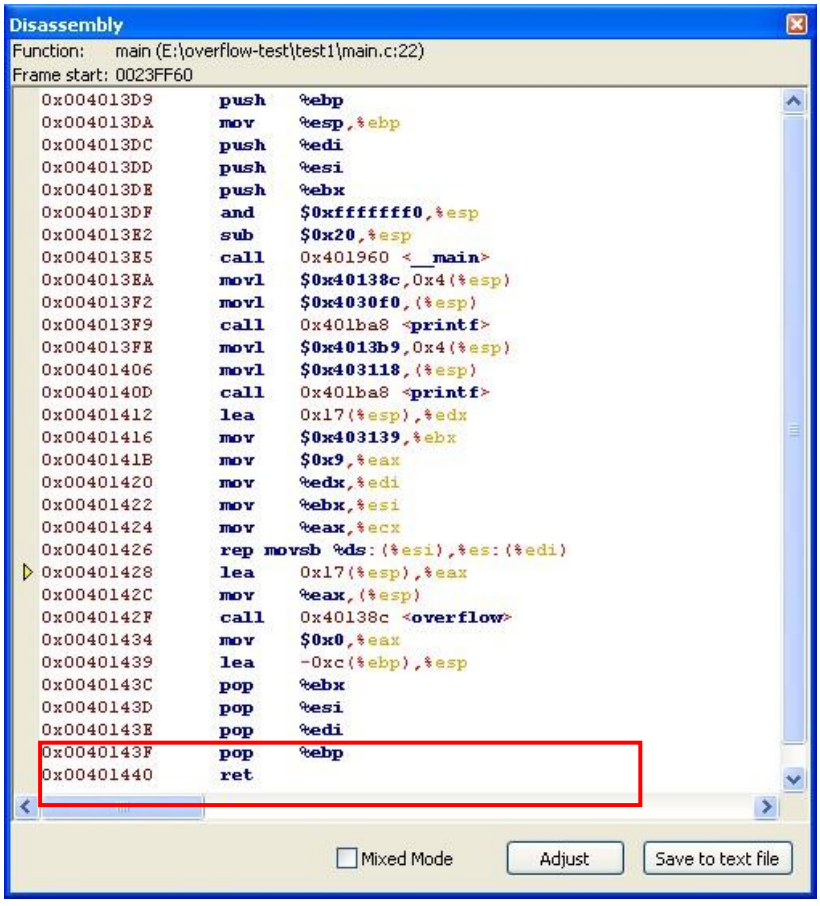
构造函数 overflow 的输入值（input），使得函数 overflow 函数中的局部变量 buf 溢出；溢出部分的值为函数 fun 的入口地址，并将覆盖函数 overflow 被调用时栈空间的返回地址。**重点在于找到栈空间中 buf 与返回地址间的相对位置。**

攻击方法与过程如下：

1.1 在 main 函数中“overflow(input);”语句处和 overflow 函数中“strcpy(buf,input);”设置断点，并启动调试。此时，观察命令行输出，其中 fun 函数的入口地址为：Ox004013B9。

1.2 选择“Debug->Debugging windows->Disassembly”窗口，查看 main 函数执行时相应的汇编语言和对应逻辑地址。

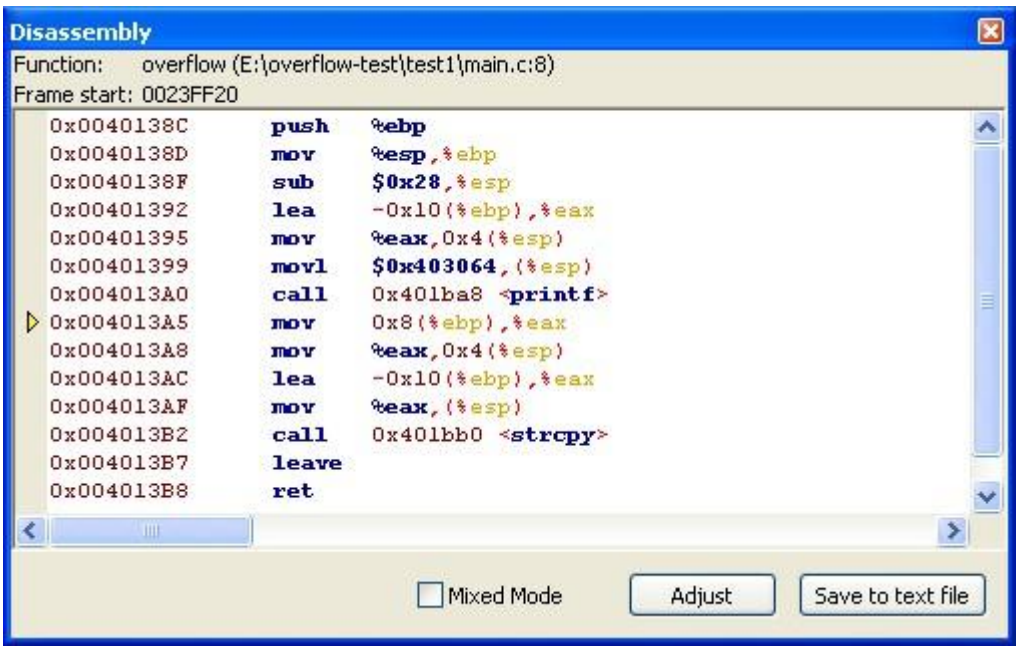
在该窗口中，找到类似“Call Ox40138c <overflow>”的语句块，如图所示：



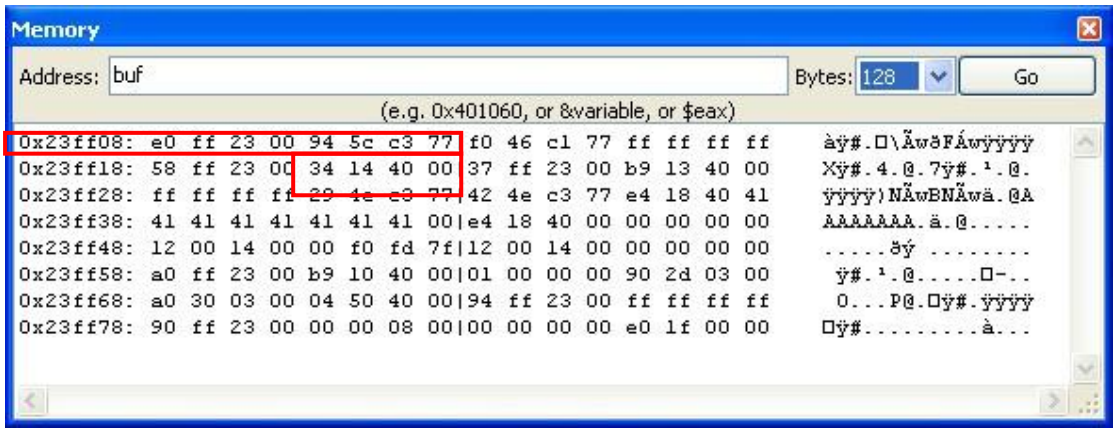
该语句块说明，在函数 overflow 返回时，逻辑地址为 Ox00401434 的语句：

“mov $Ox0, %eax”被执行。因此，函数 overflow 调用时的栈空间中，返回地址应该为“Ox00401434”。

1.3 继续调试，进入函数 overflow 执行过程中。此时，查看汇编语言窗口：



选择“Debug->Debugging windows->Memory dump”窗口，查看此时靠近 buf 变量的栈空间的内存状态（在 address 栏输入：buf）。



由图可见 buf 的地址为 Ox23ff08。因此 **windows 系统的栈空间向下生长（低位地址）**，因此函数 overflow 调用时压入栈的返回地址应该在比 buf 高的地址位。在图中查找返回地址“Ox00401434”，如图所示。由图可见，返回地址距离 buf 首地址为 **20 个字符**，返回地址占 4 个字符，因此 input 为 24 个字符时，刚好覆盖返回地址。为了使用 fun 函数的入口地址（**Ox004013B9**）覆盖返回地址，input 的地 21 至 24 位必须分别为：OxB9, Ox13, Ox40 和 Ox00。

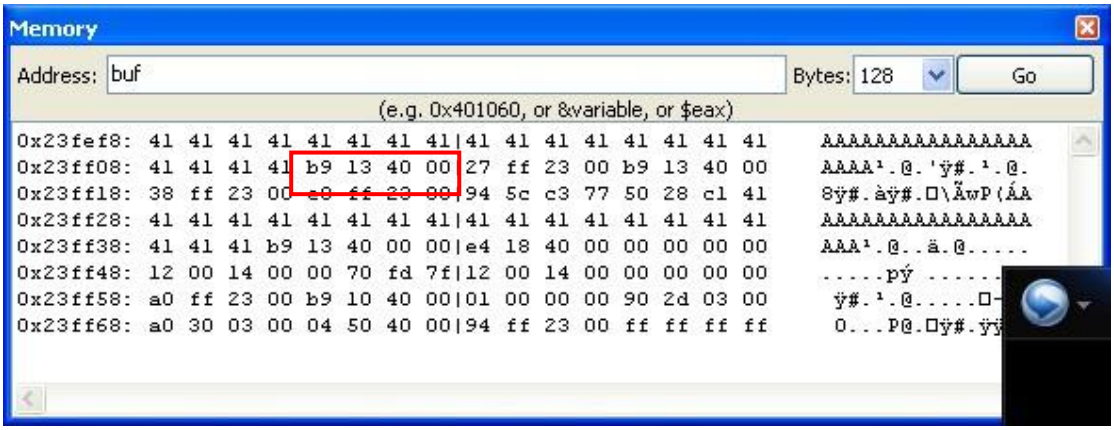
1.4 退出调试。对程序中的 input 做出如下修改：

“char input[]="AAAAAAAA";”

改为

“char input[]="AAAAAAAAAAAAAAAAAAAA\xB9\x13\x40\x00";”

1.5 保持上述断点，启动调试。观察函数 overflow 函数中语句“strcpy(buf,input);”被执行后的栈空间内存状态，如下图：



此时，返回地址位置已经被修改成为函数 fun 的入口地址 **Ox004013B9**。因此，在函数 overflow 返回时，函数 fun 将被调用!

1.6 继续调试模式下单步执行，将发现程序执行进入 fun 函数中，说明溢出攻击成功，函数 fun 被调用。