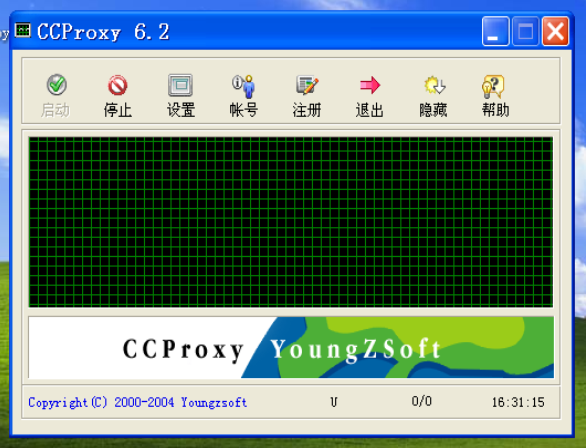
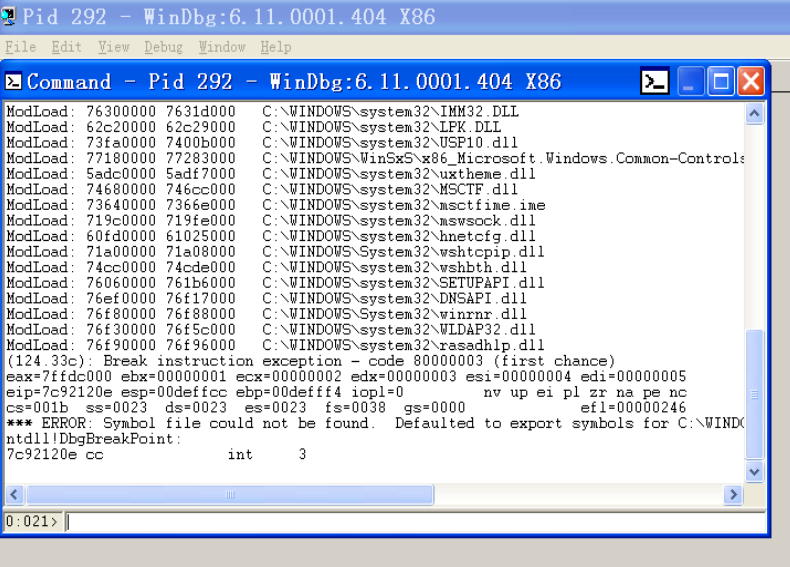
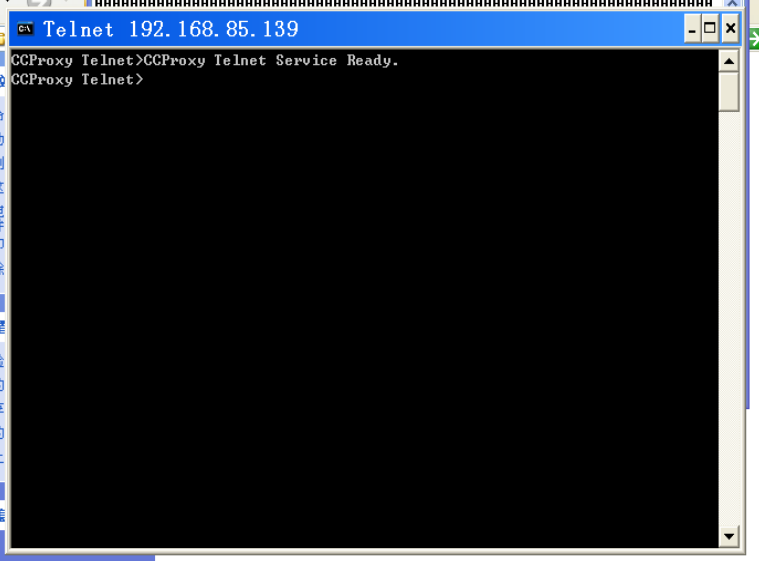
首先启动CCProxy程序，并打开WinDbg程序监控之。

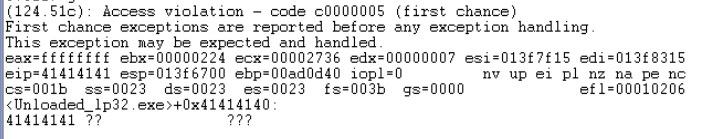




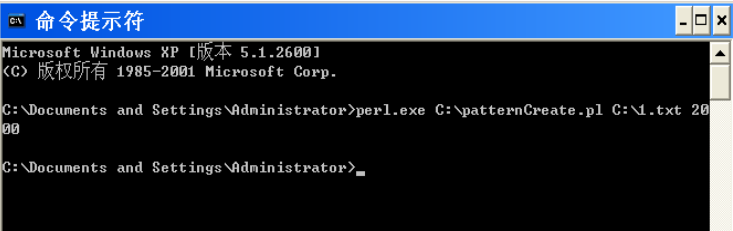
在WinDbg的窗口下输入g使程序得以继续运行。在另一个窗口内，使用telnet远程登陆CCProxy。

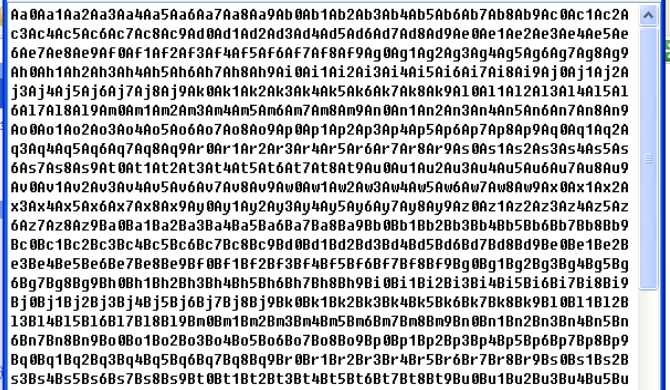


不断使用不同长度的A字符串进行ping测试，最终发现在输入2000个A时，程序崩溃。此时WinDbg程序提醒我们发生了缓冲区溢出，EIP寄存器的值为0x41414141，即ASCII字符“AAAA”。

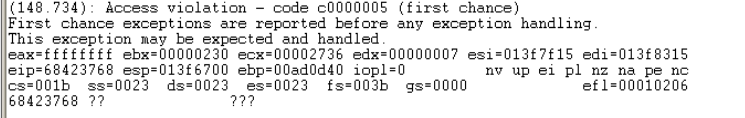


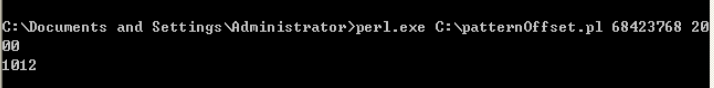
因为在A为1000个时，并没有产生此错误，因此可以判断，程序发生缓冲区溢出的条件为A在1000-2000之间。使用由perl语言编写的，自动生成不重复字符串程序patternCreate.pl，可以生成指定长度的字符串，在这里生成2000个字符的串。





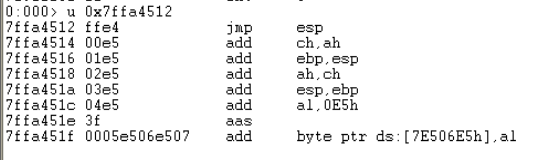
将这段字符串输入到CCProxy中，同样发生了异常，此时EIP的值为68423768，即，我们可以通过查找此子串在源字符串中的位置，来敲定RET相对缓冲区的偏移量。





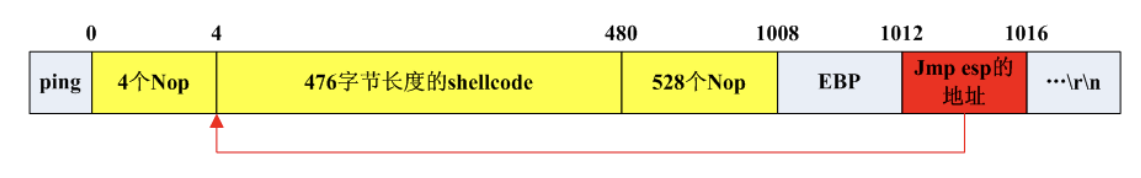
可以看到，偏移量为1012.

我们需要获取程序中，最终跳转的jmp esp指令所在的位置。由经验加上实验可最终确认，该指令所在的位置为0x7ffa4512。可以通过查询地址来验证这一点。我们在后面要在内存中填充这个地址，让程序去运行这条指令。



接下来要获取ESP在缓冲区中的相对偏移量。在发生崩溃时，输入dd esp查看ESP寄存器指向的数据，可以看到为61413161，再次使用pl程序可以得到偏移量为4.

因此，以接收ping指令字符串首地址为开始的第1012个字节的后4个字节，为RET；第4字节后的4个字节为ESP。因此，我们可以在前者的位置输入0x1245fa7f（小端对齐），程序运行到此，会跳转到jmp esp处，去执行ESP寄存器指向的内存对应的指令；并在后者输入shellcode，这样可以让程序执行我们自己的shellcode，如下图所示。这个shellcode的作用是创建一个名叫“a”的受限用户。



编写一个python文件，用其远程登陆ccproxy，并通过Ping的方式输入shellcode，最终发现CCProxy崩溃。查看系统用户，发现系统里多了一个叫“a”的用户，证明我们的攻击成功。

