Global Information Assurance Certification Paper

Elevating Security for our Clusters
Jason Tedford
GSEC Practical Assignment Version 1.4b
Option 2
December 26th 2003

ABSTRACT

Network and Systems security has always been only mildly important at various times in our company. Our team has always had security stuck in the backs of their minds somewhere and it rarely made it to the forefront of our operations. Although there are some good security solutions designed in certain pockets of technology at our enterprise, for the most part it seems as though we have had a salt-and-pepper approach securing our infrastructure. Only when large-scale issues arise does network and systems security really come into focus. Even then, it has been only for a very limited time horizon. Luckily, in the past 10 months, IS&T's internal reporting structures have changed and our technical teams now report to our "parent" company. The leadership of the executive team, especially our COO, has a sincere interest in making sure that IS&T develop a comprehensive security plan for all entities that are supported. One of the primary focuses has been to get various individuals trained in the security arena. It was paramount that the training be industry specific and not platform or vendor specific. We found that SANS offered the best all-around security training and did not concentrate on specific products. As such, we have committed a substantial amount of resources to the training of our staff through SANS and other training providers, as well as concentrating on a new found commitment to the implementation of an enterprise-wide security strategy.

After attending the SANS Security Essentials curriculum, I was shocked and amazed at how readily available software tools are and the ability to acquire knowledge and take advantage of exploits against systems in any given environment. People with even a rudimentary knowledge of how systems and networks function can inflict critical damage to various systems and environments. Therefore, the focus of this paper is to look at one of our mission-critical system's current state and implement appropriate security fixes for our environment. The areas that I will focus on are operating system security as well as certain aspects of network security to derive an overall "box-hardening" approach. This paper does not get into the security related to the central application that communicates with the player terminals on our gaming floor. This application is coded by a partner of the organization and therefore not directly under the control of the IS&T department.

BEFORE

The system that I have chosen to secure for this paper is our Account Transaction Center. This system is a set of four RS/6000 nodes operating as

two two-node clusters. They are running AIX 4.3.3 maintenance level 11 for the operating systems as well as utilizing HACMP 4.4.1. HACMP is a set of licensed program products that allows clustering of systems as well as hardware takeover in the event of some type of failure. The operating system was installed without any customization with the exception of installing additional file-sets off of the AIX media which makes it a relatively vanilla install. Our development partner for the application running on the box has done some minor security enhancements by assigning different port numbers for commonly used ports in a clustered environment as well as shutting off certain services not utilized. However, this was just a cursory look at the box from a security perspective. As was pointed out consistently in the GSEC training, the defaults of an operating system installation are highly insecure. The hardware consists of four RS/6000 model H70's with redundant fibre channel connectivity to an Enterprise Storage Server for all storage except for the root volume groups. The root volume group contains the operating system and is on internal disk. These machines in the cluster serve account information for our patrons as well as keep track of all player transactions on our gaming LAN. The LAN consists of five Cisco 6500 series switches as well as a number of 3500 series switches. These switches allow for connectivity of our player terminals on the gaming floor. These terminals are from various different manufacturers that include Sigma, Konami, Leisure Time, and Oneida II. The Sigma, Konami, and Leisure time terminals are all serial based terminals that connect to a terminal server to transmit account activity across the network. The Oneida II terminals are Linux terminals that provide direct Ethernet connectivity. Patrons use a card to slide into a terminal in order to login to the terminal. After the card is in the terminal the patron is presented with a cipher-pad in order to input their PIN. This PIN is verified with the Transaction Center and upon success, the player's balance is displayed at the terminal and the patron can start gaming. The gaming transactions consist of debits and credits to the player's account. When the patron is finished playing at the terminal, the card is removed and a logout transaction is sent to the Transaction Center.

DURING

In order to assess and secure the Transaction Center, I decided to utilize for the most part the tools that are available to anyone who would want to exploit our environment. Many of these tools were presented in the GSEC training material. However, I will also utilize some that were not. After utilizing these tools, I matched up what I found to how the system was configured in order determine the appropriate changes that needed to be implemented.

One of the first areas that anyone would look at is what services are available on the target system. In order to determine what services the Transaction Center was running I wanted to do a port scan on the four nodes in the cluster. In the SANS GSEC training program, Nmap was referenced many times as a tool of choice for network scanning. I downloaded Nmap version 3.48 in gzip compressed format and installed it on my Linux box [1]. After reading some documentation, I decided that I would run a TCP SYN Scan as well as a

UDP Scan. I first utilized Nmap to TCP SYN scan all well-known ports (ports 1-1024) on all four nodes and the results are listed below. In a normal TCP connection the initiator of the communications sends an initial SYN packet to the target machine. If there is an available listener on the target port, then a corresponding SYN/ACK is sent to the originator. The originator then sends an ACK back to the target. This process is known as the TCP Three-Way Handshake. The TCP SYN scan or Half-Open scan sends an initial SYN packet to the destination host. If a corresponding SYN/ACK is received by the target, then a RST or reset packet is sent back to the target to terminate the session immediately [2]. For UDP, there is no corresponding Three-Way Handshake since UDP is a best effort protocol. A packet is sent from a source ip and port to a destination ip and port address pair. If the destination machine has a listener configured to accept packets for UDP, the packet is received and no reply is sent back to the source. If there is no available listener, the target machine sends back an ICMP Port Unreachable message. In the UDP scan, a packet is sent to each port to be scanned in the list. Again, I utilized only ports 1-1024. If an ICMP Port Unreachable message is sent back to the source address, then the port is not listening for UDP traffic and is assumed to be closed. However, if nothing is sent back from the target host, then the target is assumed to have received the UDP Packet. Since UDP does not need an acknowledgement at the source that a packet has been received at the target, then the port is assumed to be open [2]. Between both of these scans, I was able to determine which services were running on each host. All four nodes were running the same services for both the TCP SYN scan as well as the UDP scan. The following list shows the ports that were available for the scans:

TCP SYN Scan

7 – echo	9 – discard	13 – daytime	19 – chargen	21 – FTP
23 – telnet	25 – smtp	37 – time	80 – http	

UDP Scan

7 – echo	9 – discard	13 – daytime	19 – chargen	37 – time
111 – sunrpc	123 – ntp	161 – snmp		

Armed with this information, I was able to look at which ports were required for the normal operation of the system as well as any administrative or management ports that were required for day to day operations. The first services I looked at were echo, discard, daytime and chargen. I had an idea of what these services were from past experience, but I decided to research them just to make sure.

The echo daemon is a service that sends any information it receives back to the originating source. This service is generally used as a tool to troubleshoot connectivity issues with a remote system by measuring round trip times as well as if communications exist at all [3]. The discard daemon is another troubleshooting service that receives information on its port after a session has been established and simply discards any information received [3]. The chargen

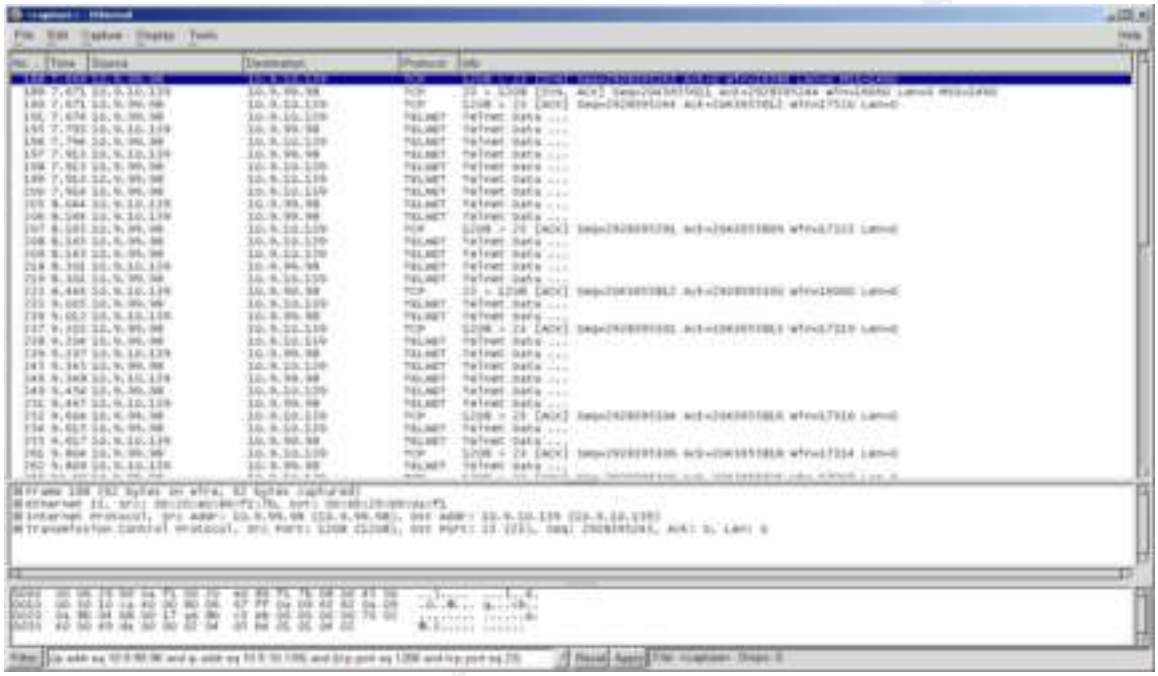
service is a character generator that sends back a character stream to the originator when it receives activity on its port. This service is useful in order to find delay or packet loss in the network since it sends a continuous stream of data back to the originator. The Daytime service simply responds back to the sender the current date and time in ASCII format [3]. These services are there simply as a means to test and troubleshoot different types connectivity issues. In our cluster environment, we do not utilize any of these services for testing any part of our network or applications. As with anything that can be used for good, these tools can also be used for evil.

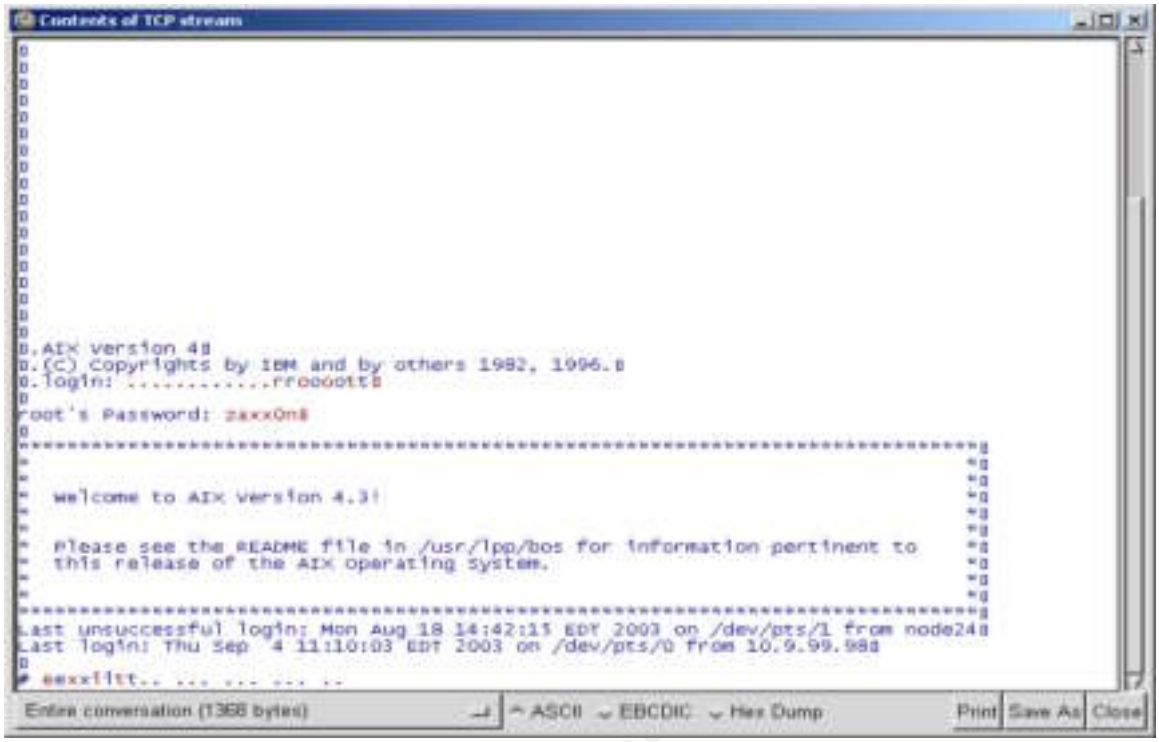
I next went out on the Internet to find any specific vulnerability associated with these services. I found CERT Advisory CA-1996-01 UDP Port Denial-of-Service Attack which points out that if a packet is forged such that the packet's source port is the echo service and its destination port is the chargen service, a potential denial-of-service condition may arise [4]. The original packet would cause the chargen service on the remote host to send a constant stream of data to the spoofed address which would then echo back all of the data it received on the echo port. This forged traffic would cause constant network traffic between the two hosts. This race condition could saturate the network as well as cause the two machines to concentrate their processing on network I/O and CPU interrupts. Given the potential for outage based on this CERT advisory, I decided that the risk was too great in order to continue to leave these services running, especially since they are not utilized. In order to stop these services from starting at boot time, I commented out the lines that pertain to these services in the /etc/inetd.conf file. Since the inetd process is controlled through the System Resource Controller in AIX, In order to make these changes take effect right away, I issued the following command to refresh inetd and re-read the inetd.conf file:

```
refresh –s inetd
```

The next service I looked at was the telnet service. Telnet allows a remote user to open up a terminal session to a host as if they were sitting right in front of the console [5]. All of our administrators and support personnel utilize the telnet service to remotely log into the cluster. The problem with the telnet service is that all of the traffic is transmitted in clear text [6]. As most administrators who have ever utilized telnet to manage a server know, the userid and password will be transmitted in clear-text as well. To show this I utilized a network sniffing program called Ethereal to capture packets off of our network between my machine and one of our nodes in the cluster. Ethereal is a network protocol analyzer available from http://www.ethereal.com/distribution/win32. I utilized the Windows version of the tool. The results of the packet capture show that in fact the root password can be retrieved in clear-text. All that needs to happen is to find the originating packet to setup the telnet session with the remote host. One way to do this is to take the packet capture and filter by either the source or destination IP address. The filter that would be entered is ip.addr == x.x.x.x where x.x.x.x is one of the IP addresses in question. The first packet in the

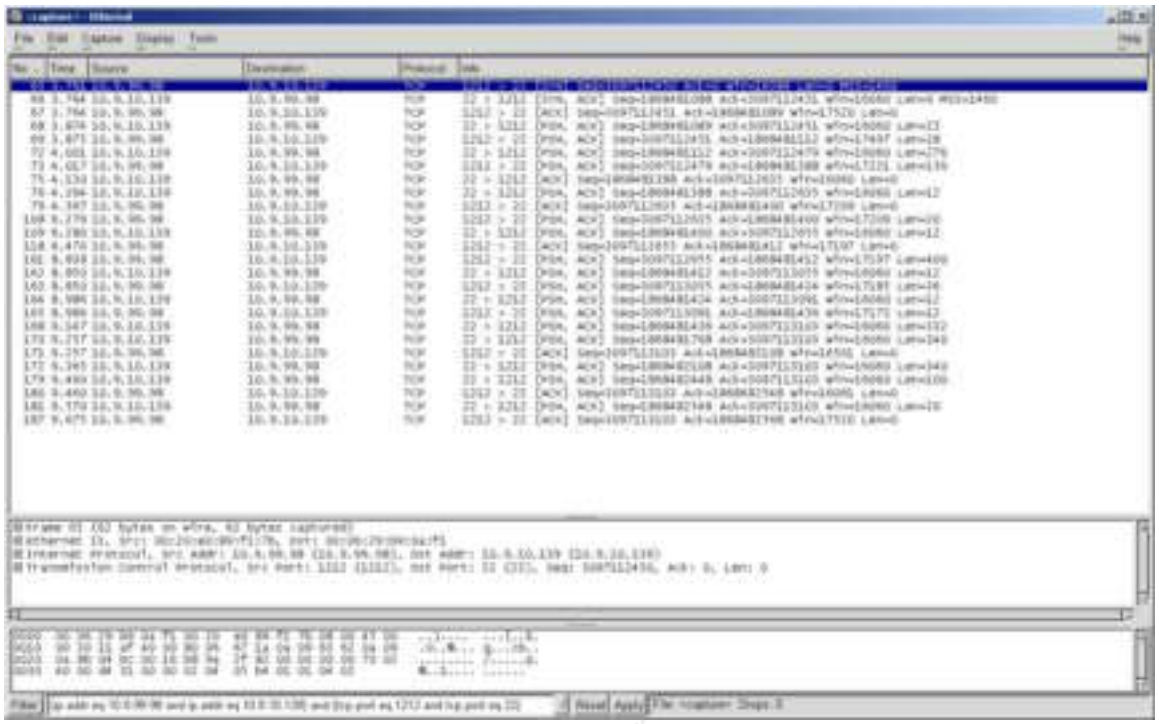
Ethereal window should be the initial SYN packet requesting to set up the connection with the remote system. Now all that needs to happen is to right-click on one of the packets in the window related to the TCP connection and select Follow TCP Stream from the popup menu. The following two windows show the Ethereal packet capture filtered by destination IP address equal to one of our remote test nodes and then the TCP Stream window showing exactly the information sent across the wire. As you can see in the stream window, the root password is readily available.

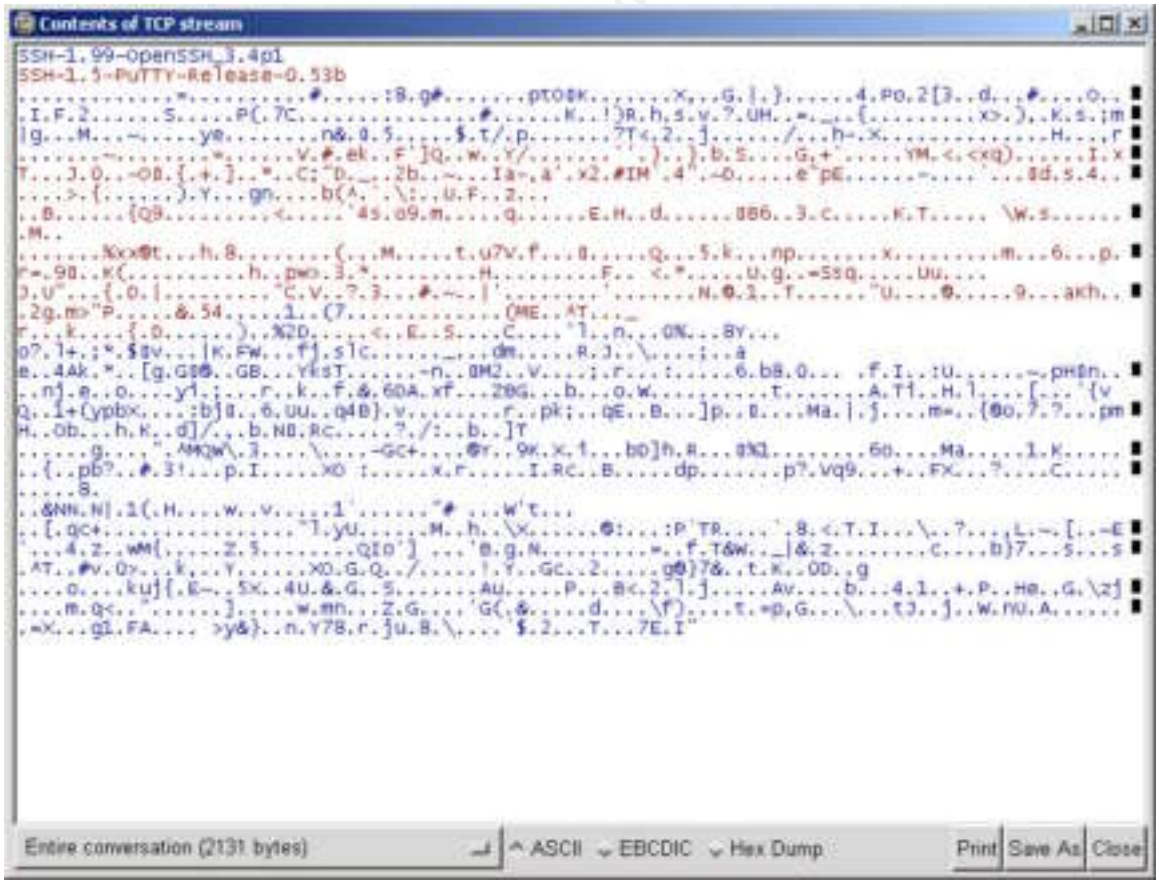




For this cluster, this one example of retrieving passwords from sniffing the network is so severe that it was paramount to get to an alternative solution for remote terminal access. The solution that was chosen is OpenSSH. OpenSSH is a free implementation of the SSH protocol that encrypts all of the traffic between two connections to prevent various network style attacks. For AIX, I chose to utilize the distribution that is included on the AIX Toolbox for Linux Applications CD. I found a great paper entitled "Securing Remote Access on AIX 4.3.3 using OpenSSH" in the GIAC practical repository written by David Randell [7]. Randell's paper gave a detailed look at obtaining OpenSSH and the process of installing it into our environment. I followed Randell's instructions carefully and downloaded all of the filesets that were indicated in his paper. After playing around with the install for awhile and testing some connections for myself, I was finished with the install. Kudos to Mr. David Randell for an informative paper with great instructions.

Now that OpenSSH has been installed, you can see in the following dump and trace that when a connection has been made to the remote cluster, the data portion of the packets has been encrypted. I utilized an SSH client installed on my Windows machine called PuTTY. This program will be described in a moment.





I next looked at the FTP service. This service allows a user with an FTP client to attach to the port in order to do file transfers either up to or down from the system. The cluster utilizes the FTP client services to transfer summary records between the transaction center and our ratings database that sits on an AS/400. The service is also used by administrators who need to be able to move scripts and other data that has been collected to and from the system. Therefore, FTP is not a service that will be able to be stopped from running in the cluster without a suitable alternative. As it happens, the OpenSSH suite also provides a means of secure file transfers over an encrypted connection. Since we are now utilizing the OpenSSH suite to connect to remotely for terminal sessions on our cluster, it makes sense to utilize this same service as the means to transfer files to and from the cluster. For both the remote terminal connections and the secure file transfers, I utilized a suite of free client software called PuTTY. This package is available at www.chiark.greenend.org.uk/~sgtatham/putty/download. You can download a zip file of all the binaries which includes all of the available PuTTY clients for secure communications or you can select only the client programs needed for a particular environment. In our environment, I have distributed the PuTTY terminal emulator to all of the administrators in order for them to have a consistent tool to manage the cluster. Our applications group had to take some time to make sure that any of the other departments that utilize a terminal emulator for remote access were able to connect with the PuTTY client via SSH. For the few administrators that need to transfer files to and from the cluster, the PuTTY PSFTP client was utilized in order to conduct transfers via the SSH protocol as well.

As far as any automated transfers that occur out of the clustered environment, I left the ftp client programs on the box in order for the scripts to send information to our AS/400. Currently, I was unable to find a suitable SSH service to run on our AS/400 systems in order to remove our reliance on any of the ftp protocol for our entire gaming environment. The cluster initiates communications to another machine in order to transfer data automatically so it is really acting as a client in this scenario. Therefore, the ftp server service has been commented out of our inetd.conf file and the init process read the inetd.conf file in again.

Next I looked at the sendmail daemon. Sendmail is the defacto standard for routing email messages in a UNIX environment. I have always heard that there are many critical issues with the sendmail daemon. I found numerous CERT Advisories on sendmail including the following:

1) CA-2003-12 Buffer Overflow in Sendmail – This advisory explains that there is vulnerability with the sendmail daemon that can possibly allow an attacker to gain access to a machine and run code at the user level of the sendmail daemon [8].
2) CA-2003-07 Remote Buffer Overflow in Sendmail – This is also an advisory that explains that a remote user could potentially exploit a

vulnerability that would allow code execution and the user level of the sendmail daemon [9].

3) CA-2002-28 Trojan Horse Sendmail Distribution – This advisory explains that some versions of the sendmail distribution had their source modified in order to contain a Trojan which could potentially allow remote execution of code [10].

After reading these three CERT Advisories as well as many other documents related to sendmail, I started to investigate how our cluster was utilizing the service. It turns out that the only thing that the cluster utilized sendmail for was to send email to a remote system. The remote system simply allowed the aggregation of email from the four node cluster into one mailbox for easier viewing of the mail.

I found an interesting article called "Improving Sendmail Security by Turning It Off" [11]. This article points out that most systems running sendmail are probably running it by default. It also points out that there are really only two reasons to need to run sendmail. The first reason is to listen on port 25 for messages destined for the machine from the outside and the second reason is to flush the local mail queue if unsent mail on a periodic basis. In our cluster, the machines simply send email to a different machine. Therefore, there is no reason for sendmail to be listening on port 25 of these machines for incoming mail. Since these machines need to email out, sendmail needs to be invoked in order to clear the mail queue on the machines. After working with the operations team as well as our administrators, we decided to invoke sendmail out of the crontab every 15 minutes. The following entry was added to the crontab for root on each system in the four node cluster:

```
0,15,30,45 * * * * * /usr/sbin/sendmail –q
```

This crontab entry will spawn sendmail to deliver queued message four times per hour. Also, since the machines do not receive mail, the decision was made to stop sendmail from listening every time the machines reboot. In AIX, the sendmail daemon controlled through the System Resource Controller. The System Resource Controller allows an administrator to create and control subsystems or groups of programs that run as services. Subsystems can be started, stopped, added or removed. The System Resource Controller starts in the inittab and then all of the subsystems are started. I removed the sendmail subsystem from the resource controller of each of the four nodes by issuing the following command:

```
rmssys –s sendmail
```

The next two services I looked at were time and ntp. The time service sets the clock of the local machine to the time of another machine running the time service in master mode. The master server controls time for all of the machines that check in with it utilizing the time protocol. NTP is the network time

protocol. This service also sets the local clock to that of a master server. Our network utilizes a time server that gets time from satellite as well as another server in our DMZ that gets time from the Internet. This time server utilizes listens for ntp requests from devices on our network and sends out ntp replies. Since ntp is used for time in our environment, there is no need to run the time service on port 37. The four node cluster is pointed to both of our NTP servers in order to synchronize time across the cluster.

The next protocol looked at was http or port 80. We are not utilizing port 80 for anything in our environment currently. There has been some development work going on in order to view application specific statistical information in the future. This development has only been looked at in our test and development environments. Since we currently are not utilizing any http services for this cluster, simply shutting the service down is appropriate for our environment.

The last protocol I looked at was SNMP. SNMP is called the Simple Network Management Protocol. It our environment we utilized SNMP as a tool to manage and collect information from our systems and networking devices. We utilize Tivoli Netview as our SNMP network manager. After learning about SNMP vulnerabilities in the SANS curriculum as well as viewing a large amount of the CERT advisories on SNMP, I knew this was an area that needed to be addressed. First and foremost, our entire infrastructure utilized the public and private community names. These default community names allow anyone with a MIB browser or snmpget tools to query and write information to our systems and networking devices. Since SANS class the entire infrastructure has changed both the public and private community strings to something else. To change the community names for the cluster the community lines in the /etc/snmpd.conf file were changed. The Netview system also had to have its configuration changed in order to allow all of the polling that takes place to continue. One other piece that was added to the /etc/snmpd.conf file on each community line was the addition of the ip addresses of the network management stations that are allowed to issue requests to the read/write and read-only communities. This now allows our snmp environment to be slightly more secure that it previously was. Also, since our network management stations are connected to the same subnet as the cluster, I requested for our network team to block all incoming and outgoing SNMP requests on the routers and firewalls for this segment.

Now that the cluster has been looked at from the perspective of services that were available from the network, I decided to look at a couple of other things. Since it is possible for someone to spoof an ip address of one of the network stations, I decided it was important to attempt to keep track of this. I utilized a program called arpwatch which is available from http://www-nrg.ee.lbl.gov/nrg.html. Arpwatch is a tool that keeps a database of ip address to Ethernet address mappings. The arpwatch man-page describes the tool pretty well. The database is a flat file with the default name of arp.dat. The information stored in the flat file is MAC Address, IP Address, epic time, and hostname. The program keeps track of changes by references changes with either the MAC address or IP address and time since the epic. Any changes in the network mappings are logged in the database and an email is generated. This email gets

sent to the mailbox of the user who launched the application. I set up a machine (a pc acting as a server) with RedHat 8.0 with the arpwatch program and connected it to our network. Obviously I disabled nearly every port on the machine and also installed ssh in order for me to get to the machine remotely. I also configured the crontab to start sendmail to empty the mail queue every 10 minutes. When the program (arpwatch) is started, it references a file in the /etc/sysconfig directory named arpwatch. This file contains any options to utilize when running the program. By default, there is an entry OPTIONS="-u pcap" which causes the program to run as the pcap user instead of root. I also configured a .forward file on this machine to my own internal email address. The .forward file will take any mail destined for the user on the local machine and send those emails to the email address in the file. Now, anytime an ip-MAC pair is changed in our network for whatever reason, an email is generated and various people are alerted. We have come up with some new internal procedures for normal maintenance and service of computers on our network so certain emails received are ok to get. If there is not a corresponding maintenance request to match up to an email that has been received, the networking team will investigate in order to see why the mapping has changed.

AFTER

This paper shows that with a very limited amount of effort, our cluster environment was able to go from a state of excessive exposure to a relatively secure system. Much of the effort was to determine if the services that were running on the system were actually needed for the proper operation of the environment. In cases where services were not needed, simply shutting them down is one of the best alternatives. A next step for these unneeded services will be to remove the actual binaries themselves in order to further limit exposure in the event that the system is compromised. Although this step was not part of the process for this paper, we will be identifying the filesets for each service that was disabled and remove them in the future.

For the protocols that were being utilized within the clustered environment, a simple investigation into alternatives to the less secure protocols yielded great alternatives. Migrating from the unencrypted Telnet and FTP services to OpenSSH makes eavesdropping or sniffing on the wire a much more difficult task since the traffic on the network is encrypted. As everyone knows there have been recent security issues with certain implementations of SSH. Even though we are now much more secure with SSH installed, the recent issues in certain versions of the protocol point out that security architecture and implementation is an ongoing task. All of the people responsible for security will need to continue their education and training as well as stay abreast of new alerts and issues in order to minimize our exposure to all of the systems in the environment.

References

[1] “Downloading NMAP”. URL: http://www.insecure.org/nmap/nmap_download.html (5 August 2003).

[2] Fyodor. “The Art of Port Scanning.” 6 September 1997. URL: http://www.insecure.org/nmap/nmap_doc.html (5 August 2003).

[3] Haden, Rhys. “IP Small Services”. URL: http://www.rhyshaden.com/ip_small.htm (6 August 2003).

[4] “CERT Advisory CA-1996-01 UDP Port Denial-of-Service Attack”. 24 September 1997. http://www.cert.org/advisories/CA-1996-01.html. (6 August 2003).

[5] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994. 401.

[6] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994. 417.

[7] Randell, David. “Securing Remote Access on AIX 4.3.3 using OpenSSH.” http://www.giac.org/practical/GSEC/David_Randell_GSEC.pdf. (September 2003).

[8] “CERT Advisory CA-2003-12 Buffer Overflow in Sendmail.” 29 May 2003. http://www.cert.org/advisories/CA-2003-12.html. (8 August 2003).

[9] “CERT Advisory CA-2003-07 Remote Buffer Overflow in Sendmail.” 9 June 2003. http://www.cert.org/advisories/CA-2003-07.html. (8 August 2003).

[10] “CERT Advisory CA-2002-28 Trojan Horse Sendmail Distribution.” 25 March 2003. http://www.cert.org/advisories/CA-2002-28.html. (8 August 2003).

[11] Pomeranz, Hal. “Improving Sendmail Security by Turning It Off” Sys Admin Magazine June 2003 (2003): 8-11.