"Improving Security in a Public Library for Patrons and Staff"

Brett Charbeneau

GSEC Practical Assignment Ver.1.4b
Option 2
June 24, 2003

ABSTRACT

Public libraries present an interesting set of security challenges to administrators. While there are servers and workstations dedicated to staff use which require the usual safe practices and protection that we think of in commercial networks, there is also the mission to supply workstations for the public that make in-house servers accessible and provide unfiltered Internet access. It is not uncommon for libraries to possess servers that are used by both staff and patrons which require open access at public levels with very controlled administrative access for senior staff. Added to the fray for security is a careful balance that must be achieved with a vendor who maintains the library's catalog server with little consideration for pro-active security measures.

The concept of "defense in depth" provides a specific and thorough approach to network devices, servers, and workstations. Host-based intrusion detection on servers, using Tripwire, offers a way to monitor system file integrity while stateful iptables-based firewalls give protection at the network level by separating public from staff networks, filtering packets, and providing Network Address Translation (NAT) addresses for workstations and servers. The deployment of a central logging server gives the ability to monitor network traffic and provides a baseline of normal activity and the knowledge that things are working the way they should via log audits. Finally, confidentiality for archived institutional data and patron activities present two different facets of this concept: one through careful preservation via correct administration and another through complete deletion to protect privacy.

INTRODUCTION – Life on a Public Library Network

While the security mantra of data confidentiality, integrity, and availability certainly applies, there are some curiously different applications of the standard in a public library environment. Our mission is to provide services and programs to the public for free – and this includes access to online resources to which we subscribe as well as the Internet in general. For many patrons, the only access they have to the online world is through our workstations and these computers see almost continuous use while we are open.

The advantage of operating in a non-commercial environment is that we are unfettered by concerns about accurate billing, industrial espionage, losing contracts to competition, distant markets, and other elements of business networks. However, part of our mission involves getting the public on our network - something business and government entities are generally loathe to do and usually have explicit policies against. Internet Service Providers (ISP's) are somewhat similar to what we are about because they allow the paying public onto their systems, but there is a distinct difference between a customer who agrees to an acceptable use policy and a patron who may come in off the street once and who we will never see again.

For us, unrestricted and confidential access for patrons is absolutely paramount. We had strict policies in place to insure patron privacy long before our institution came online. There are books in our collection that not everyone agrees with and some people find offensive, and yet we procure and circulate them in the interest of free access to information for all. Part of free access is the knowledge that, as a patron, your research habits can be satisfied in complete privacy. Occasionally this works against our ability to enforce our online policies: because we don't keep track of them, patrons in violation are hard to catch in the act.

At the same time, we cannot carry out our mission without a network and that infrastructure has many of the same security needs of a business. Leading the list is our catalog database. This dataset contains an entry for every book, cassette, VHS, CD, and DVD that we circulate as well as the contact information of some 50,000 patrons. We cannot afford to have this data compromised and if it becomes unavailable much of our work grinds to a halt: books can't be checked in or out, fines cannot be paid, catalog searches cannot be done. We also have a diverse group of other databases and internal documents that our staff manages and creates to make our many programs possible.

Like the critical data used by a business, our resources are worthy of thoughtful, in-depth protection. Commercial networks are for the exclusive use of staff for business activities, and this limitation is usually enforced through required logins at each workstation and limits on physical access. The device on the perimeter of the network that separates the LAN from the Internet is a serious line of demarcation between "us" and "them". You develop default drop policies on the gateway device and monitor what happens on the internal network in search of misbehaving machines or people – the latter of which are usually reigned in by a Human Resources Department's disciplinary procedures.

ISP's operate their businesses more in line with what we see in the library world: the line of demarcation between "us" and "them" is blurred along several fronts. As an ISP, you have to protect your resources from Internet attacks. Because these resources are likely to be used by both your customers and your staff, you also have to know and practice the difference between the needs of the customers and the needs of administrators. Customers need to have limited access to machines that serve both customers and staff and no access to staff-only servers and network devices like switches. This is usually achieved through strong authentication and access control lists (ACL's). ISP's are also compelled to keep detailed logs of network activities as a way of monitoring what happens and to find culpable parties if policies are disobeyed.

However, ISP's aren't always concerned about user privacy since it may or may not be a part of their company's mandate. ISP's may choose to filter

Internet content. They also have the ability, via their well-kept logs, to find and ultimately eject “bad” users by deleting their accounts if violations warrant it.

Our strict policies on preserving patron privacy are never far from our thoughts. When checked out material is returned, any connection between the person and whatever they took home is severed. The system has no mechanism for archiving this information even if we wanted it. We will not give out any information on any patron to any individual, even to family members, with the exception of parents asking about their own children or law enforcement agencies with subpoenas. Not surprisingly, this approach to patron privacy is extended to online activities. A library card is not required to use one of our workstations and we have a great amount of out-of-town visitors who use our resources only while they are visiting. We do not record which machine a patron chooses to use nor what they do while they are using our workstations. We do not filter Internet content in any way. (As of this writing, the Supreme Court of the United States has seen fit to require filtering in public libraries that use federal monies to fund their Internet access. Because my institution has never used federal grants to pay ISP fees, we are exempt from this requirement.) Many monitors have privacy screens so only the user can see the display. Up until recently, every workstation had its own printer so that patrons could make hard copies discreetly. (This may seem excessive, but particularly with health issues a patron may, quite reasonably, not want their research to be viewed by anyone else.)

In some ways this goal of maximizing patron privacy works against us, especially in terms of enforcing an acceptable use policy. Patrons are constantly coming and going. Connecting a specific patron with a specific action on the network is extremely difficult and even if it wasn’t, patrons are not employees that can be taken to task very easily. Our best recourse against repeat offenders, if we’re somehow able to catch them red-handed, is to ban them from the library which rarely happens.

In short, the approach to network security in a public library is similar to that of a business network but with a very different context that requires a twist on some of the typical concepts and procedures. The bottom line is that as the Network Administrator for this facility I am charged with keeping the whole lot as secure as possible, protecting our data and resources, while protecting the privacy of the public at the same time.

BEFORE - A LAN on the Internet with Full Access to and From – Eeek!

The main problem staring me in the face, and made more apparent as my GSEC (Global Security Essentials Certification) class went on, was that I had an institutional network connected to the Internet that was rife with opportunities for crackers both remotely and locally. (Please see and refer to Appendix II for a diagram of the network before any security enhancements were performed.)

The first device on the perimeter is a Cisco 2501 router. This gateway was set up with no ACL’s or packet filtering, just the routing tables it needed to operate. My GSEC class enumerated the risks this kind of open access presents.

My instructor and our reading painted vivid images of crackers freely scanning and mapping my network in my head. Once an attacker had mapped my network they would quickly learn the number and locations of my workstations, the status of their Windows shares, and the number and kinds of services my servers were running, among other things. This information would give an attacker an excellent vantage point from which to research and try known vulnerabilities against my network. My class additionally pointed out that having decent ACL's defined on the router would allow me to limit staff access to inappropriate and risky Internet resources like Instant Messaging (IM) and Peer-to-peer (P2P) file sharing services like KaZaa.

The more I learned in GSEC about network infrastructure the more I realized how vulnerable our network was. All my workstations and servers were linked together via 10baseT hubs and the class section on network sniffers revealed how easy it would be for someone to eavesdrop on all the traffic and capture passwords and other critical data. Compounding this threat was the fact that my network existed as a single subnet making it even more conducive to successful sniffing. Floppy-based sniffers on public workstations became an obvious threat in my circumstances.

The servers I am responsible for, both Linux and NT, also presented much room for improvement. There was no access control mechanism in place so I had no way to block public access to staff servers nor to limit public access to shared servers. My GSEC class reading listed tools like TCPWrappers, IPSec, and iptables as a way to gain some control and why this was a good way to enforce access limitations. The servers were also operating very much as out-of-the-box installations with many unnecessary services running. Not knowing exactly which services were running where made keeping up with security patches almost impossible and therefore opened up vulnerabilities to these servers. Our in-class discussions on removing unused services made it clear that limiting services makes it easier to keep up with important patches. Limiting the services installed also gives you a great start on defining access rules in iptables so only those services you explicitly allow through are exposed. Additionally, I learned that separating services over several servers allows for greater redundancy – when a server goes down you only lose one or two services as opposed to everything.

Passwords on these servers were not being rotated regularly and the lack of a password policy meant that some users were getting away with typing the name of their favorite baseball team at login. The GSEC reading makes it plain how trivial it is to determine a dictionary-based password with a program like Crack. Finally, I had no secure way to administrate these servers as I was using either regular telnet or WinVNC. With the aforementioned ease of using sniffers on my network, these methods of accessing the server pushed administrative passwords onto the network in clear text. During our class we talked at length about tools like stunnel and SSH and all the flexibility they provide by encrypting data on the fly.

The Windows workstations on my network were also in trouble. Staff and public workstations were on the same subnet. The configurations on these computers were easy to change and users were installing downloaded plug-ins and illegal software. The fact that all these workstations were assigned routable IP addresses and were not behind any sort of firewall opened them to direct attack from the Internet. Because of the network configuration, malevolent patrons, if they knew at all what they were doing, could have complete access to our local servers and staff workstations on all levels.

Along with the issues GSEC pointed out to me on extant devices, I also learned about the importance of some things that were completely absent from my network. Over and over we revisited the importance of system logs and how a central logging server makes collecting and simultaneously examining these logs possible. Becoming familiar with your router and servers logs allows you to define what “normal” traffic an behavior is. Once you know what is typical, the abnormal and possibly nefarious activity starts to jump out at you. There are also many benefits to being familiar with the daily goings-on of the network you are responsible for, and system logs can give you a cold splash of reality as they reveal what is really happening on the network. The idea of a central log server made sense to me, particularly if I could set up one in a secure way so that the integrity of the data it collected could be preserved and archived.

As GSEC progressed, it was also becoming apparent that my network would benefit from one or more stateful firewalls. Deployed correctly, they could protect both my servers and workstations from attacks originating from the Internet. A firewall configured to use NAT offered some appealing qualities for the machines on the internal network. I also learned that I could also use these sorts of firewalls as a way to separate my public and staff subnets.

DURING - Making with the Security Spackle

Since the whole central logging server concept offered such an effective way of monitoring not only network behavior but also the effects of my security enhancements, I decided to make this a priority.

I decided to place my logging server on my network with a routable IP address and not behind any firewall except for the installation of iptables on the host itself. In order to strengthen this system against attack from the Internet I blocked all routes to this address at the gateway router. My goal was to make the logging server difficult to reach from the Internet and cutting off *all* traffic to it at the gateway seemed more effective than just denying all port 514 traffic to a firewall at this address (which would then forward log traffic on to the logging server on an internal network). The less exposed a server is to the outside, the less likely it is to get compromised. To protect against attacks from the inside, access to the two ports that actively listen was source-address limited to my network with iptables and TCPWrappers.

Once I determined where to place this server on the network, I found an excellent article by Eric Hines on the Linux Security website[1] that contained detailed guidance on setting up the central logging server itself and good advice on how to configure syslogd on the servers you wish to monitor in a way that obscures the fact that the logs are being sent somewhere else. After installing the OS (RedHat Linux) the article walks the reader through the steps of eliminating all inetd services, disabling RPC services and non-administrative accounts, and installing and configuring OpenSSH. Hines also describes setting up a packet filter, using ipchains, so that all the server effectively does is listen to UDP port 514 for the log data coming in from other servers and TCP port 22 for access to administrative SSH.

I chose to deviate from Hines' advice in substituting iptables for ipchains. Iptables came into common use since the article was published and is built into the Linux 2.4 kernel. I was also very interested in setting up rules with a default deny policy and iptables has the ability to do stateful packet inspection which allows it to carefully monitor protocols that use ephemeral ports (like FTP). The flexibility iptables possesses with logging statements, on input and output interfaces as well as on forwarded traffic, made the central logging server that much more valuable in analyzing my network. I was also attracted to the idea of setting iptables to do stealthy things like return host unreachable packets and to totally ignore pings. It's a defense in depth principle to be sure, but if and when an attacker gets on my network somehow I want this host in particular to be hard to get to. (Please see APPENDIX I for an excellent chart comparing the features of ipchains and iptables directly.)

I altered the listening port for SSH to the non-standard port 60 to help obscure administrative access to this box if it is scanned. I also chose to install Tripwire on this server as a means of monitoring the monitor, as it were – I definitely wanted to know if this server had any unauthorized file changes. Finally I made sure the iptables rules included an allowance for the Network Time Protocol (NTP) so that the server's clock would be spot on for accurate timestamps. All of my servers use NTP clients as it is vital to have this synchronization to insure that entries are recorded accurately - incidents must be considered in the correct order.

From this point on it is safe for the reader to assume that all Linux servers and firewalls have iptables installations which are set up in a similar way and for the very same reasons. Iptables is a very useful tool.

Hines goes on to walk the reader through a method of recompiling syslogd on the clients so that the syslogd.conf file the daemon reads is in a non-standard location. He recommends leaving the standard conf in its original location as a dummy once this is done. I followed this advice on the syslogd clients as a way to confuse attackers should they compromise one of my servers. If an attacker examines the original syslogd.conf file the mention of a central logging server will not appear. Since part of the idea of keeping logs is to figure out how successful compromises take place, I definitely wanted as much log info on the compromise itself to reach the logging server as possible – and as clandestinely as possible. An attacker would either have to carefully examine the new binary syslogd file to discover where I put the real conf file (I gave it a very obscure name and location) or would have to examine the traffic leaving the server to determine that port 514 traffic was indeed being beamed out.

One of the really wonderful things about my mentor class was the way our instructor encouraged us all to share other useful resources with the class. One of the gems I learned about this way was a program called NTsyslog[2], which allows the audit information in the NT Event Viewer to be sent to a central logging server via UDP port 514. I installed this on my NT box (after turning on auditing and adjusting the log settings to include all successes and failures) so that I would have the events on that server sent to the same place all my Linux log info would be kept. I was pleased to learn about this utility, particularly since it was not in our reading and is certainly very useful in this context.

The next missing piece on my network that GSEC helped me recognize was the lack of any sort of firewall. While there was certainly an option to purchase a commercial solution for this, budgetary constraints kept me from considering these for long. I figured I could set up a few extra PC's I had as dual-homed Linux boxes and mitigate a number of threats for both servers and workstations at one fell swoop: lessen attacks from the Internet, give all computers non-routable IP addresses by using NAT, and separate the public and staff subnets to help prevent sniffing.

Rather than set up a single firewall just behind the router I decided that multiple firewalls would best suit our situation. The staff and public firewalls have different requirements - it's okay if the public wants to use IM, for instance – and a firewall for each would separate the subnets nicely as well as add some redundancy. Because of the critical nature of our mail and catalog servers I wanted to have the option of getting logs from those servers *and* their own respective firewalls in order to compare for possible weaknesses. In addition, with this setup the complete failure of one firewall would not bring down my entire internal network but only a portion of it.

I started out with two Pentium PC's, each with two network cards. After installing RedHat Linux and configuring the server with NTP so that their clocks were in sync with the central logging server, I recompiled syslogd and used a dummy conf file, a la Hines. I then set about the process of installing Tripwire on these firewalls. This can be a confusing process for a first-timer and I found an article[3] by Carla Schroder to be extremely helpful. As with all installations of Tripwire that I will be discussing, the reports are run daily or at most weekly and emailed to myself and the root account of my email server so that I can analyze them remotely and have a backup copy should I accidentally delete a report. (My staple Tripwire install for all servers involves the binaries for the program running from a writable CD in a non-writable CD-ROM drive, and the Tripwire database being stored on a write-protected floppy disk. The former keeps intruders from altering the Tripwire program itself that I use to run reports and the latter allows me to update the database easily when patches are released, but keeps this dataset from being altered unless I'm there to physically remove the write protection.)

After I installed the OS and insured that install with Tripwire, I set about the task of defining a default deny policy for these firewalls with iptables. I have a saying that "flexibility breeds contempt," and iptables definitely possesses this characteristic for someone new to this powerful filtering tool. Originally, I looked around for a script that would write a default deny policy based on a reasonably flexible conf file, and I found that in gShield.[4] After configuring this script to create the ruleset I had something to study and watch in action before writing my own from scratch. The gShield script was invaluable as a guide for developing logging statements as these made it easier to debug my homegrown iptables rules.

Gateway Router

My attention now turned to my gateway router. No traffic passing through this router was ever filtered or subjected to an ACL. There was no means of keeping track of traffic at the gateway to determine what “normal” activity was or what kind of attacks we were experiencing. One option to remedy this situation was to simply replace the router with something “better” but the fact of the matter was that the hardware was already adequate for the job and some kind of network appliance was going to be very costly. There was also a great opportunity for me to become familiar with Cisco IOS by learning how to reconfigure the unit.

My goal was to try to reconfigure the router to specifically block and log at least the most popular attacks being seen on the Internet with the ultimate objective being a default deny policy at the perimeter. I decided against such a strict policy at the outset for two reasons. First, the vendor we contract with to maintain our catalog server was not forthcoming on their access needs for this computer. They had sent a very unspecific document on firewall needs, but it was incomplete, as testing quickly proved. I just knew if I was missing that critical rule to allow their traffic in that as soon as I left town on vacation the server would crash and my colleagues would be stuck with a tech support team that couldn’t do their work.

In addition, the catalog server is the bailiwick of a dedicated system administrator for that machine. Before enumerating a specific policy of what does and does not get in, I wanted the opportunity to show my coworker the value of secure access to servers from remote locations. I didn’t know specifically what her needs were, and I wasn’t sure she could give me routes and port numbers even if I asked. I also wanted enough time to either convince or compel our vendor to practice secure server administration and to verify that they possessed the proper client programs (like SSH) and understood how they worked. So, I designed my initial set of ACL’s as a stepping stone towards a better-defined set of rules once everyone, including my administration, was on board with the idea. This effectively moved a portion of my perimeter back to the internal firewalls, but it was a (hopefully temporary) compromise I really had to make in our specific situation.

Since the central logging server was all set to go I thought that blocking and logging the most popular attacks at the perimeter would provide some great documentation to carry to my constabulary to show them that we were under attack all the time. I learned about dshield.org in class and headed over there to see what sorts of attacks were making the rounds. Initially, I set the router to drop and log incoming packets destined for port 445 and 137 (Netbios), port 25 (for the recent Sendmail vulnerabilities), and port 21 (FTP).

Not surprisingly I started getting log data on these denials almost immediately. After a few weeks of watching these I discovered that a vast majority of attacks originated from Asia and Eastern Europe. Because we don't serve these areas I met with my director and proposed that we can block ALL traffic from these parts of the globe. I visited registries (APNIC, RIPE) and learned what the non-North American IP blocks were. The class section on extended ACL's showed how to use wildcard masks and with this knowledge I configured the router to drop and log all packets from these blocks. I did this in order to limit the exposure of my network to the sorts of constant attacks, probes, and scans that were emanating from these areas.

Since our class reading enumerated other things that are smart to automatically filter at the gateway I decided to block traffic that an Internet-facing interface should never legitimately see: in-bound packets with my network's source address, packets with flags set in illegal ways, and certain outbound traffic (like UDP port 1434 packets in an age of the SQL Slammer worm). I also learned that some services on the router should be deactivated, such as snmp, ip redirects, ntp and the like. There's no reason packets like these should ever be allowed to cross the perimeter interface and heading trouble off like this at the pass should help keep that trouble off my network in the first place.

On my way to a default deny list, I specifically allowed all in-bound port 25 traffic (email) to my mail server and all port 53 (DNS) to my nameserver, but denied traffic on these ports to any other address in my network. By developing these specific "allow" rules at this stage, I could test and make sure they work in my context.

Network Infrastructure

The basic network infrastructure and subnet were the next areas I tackled. Having the public and staff workstations on the same subnet presented some real problems with sniffing. Even with the firewalls deployed in such a way as to separate these two groups, I was still very concerned about the risk of sniffing on either of the divided subnets. The option that made the most sense was upgrading all my hubs to managed switches. Switches themselves were not nearly as prone to successful sniffing as hubs and the more expensive managed switches would allow me to configure virtual LAN's (VLAN's) to completely separate the subnets. Other benefits were the ability to monitor the amount of traffic on any given port, port mirroring if I wanted to get into intrusion detection down the road, and that the newer switches would bring my internal network to 100baseT speed. So, in the most expensive move in security I asked for and received, we purchased a sufficient number of managed switches to replace all of my 10baseT hubs. I then configured a separate VLAN for public and staff workstations.

Servers

The Linux and NT servers on my network had no access control or packet filtering, no password policies, no way to insure system file integrity, and were running lots of unneeded services. I had a special problem with our catalog server in this vein – it really wasn't mine to administrate. The last thing in the world I wanted to happen was for me to tweak the security on that server (a Sun box) and then have the vendor turn around and indicate that all of a sudden they couldn't do their work or found that parts of their software ceased to function.

I felt I could manage access control and packet filtering on the Linux servers with iptables and added an additional layer of security by configuring TCPWrappers. On servers shared by the public and staff I limited access (with both tools) to North American IP addresses on public service ports and strictly to my network on administrative services. Servers that were exclusively for staff use I set to be unreachable from any address outside my network – or from the public firewall. I did this to limit the exposure of these machines to only those source addresses that were in my institution's mandated service area, which greatly helped mitigate the probes, scans, and attacks to which these servers would be subjected and to offer my staff some protection from patrons with a thirst for hacking. I decided to place my NT box behind the staff firewall on the same subnet with those workstations because I could not find an NT-based tool that came close to the power and flexibility of iptables. This little trick also allowed me to assign a NAT'd address to that NT server which was an added bonus.

I then went around to each Linux server and did an audit of the services installed to identify what was required and uninstall or disable what was not. On the NT box I made sure to uninstall Microsoft Internet Information Server straight away since that server package has such a bad history of exploits. I made sure to block anything but required access to this server at the staff firewall as well, in the same effort to reduce exposure to that box.

Because passwords are one of the last lines of defense to a server I then set about implementing a scheme to rotate passwords and defined a policy to insure only strong passwords are used. On the Linux servers I made sure to have the most recent version of Pluggable Authentication Modules (PAM) and then configured the /etc/login.defs file to only allow passwords of sufficient length, with a good combination of non-alphanumeric characters, and that cannot be found in any dictionary. For the NT box I established similar password policy using the Microsoft password filter.

I installed Tripwire or Tripwire for Windows on these servers as a way to insure file integrity and intrusion detection. On the Linux servers (and the firewalls as well) I installed and configured PortSentry to give these machines some real-time defense against port scans and particularly obnoxious attacks like SYN floods. Each of these servers was set up to dutifully beam all the log data they collected to the central server.

I upgraded the web, cache, and name servers so that they could have their own host-based iptables and TCPWrappers rules and left their routable IP addresses in place. I did not place these behind firewalls because I view these servers as somewhat sacrificial – I leave basic protection to iptables and the detection of intruders to the Tripwire packages that are installed on these boxes.

Because email is so critical to our interdepartmental communication and our mission in general, I chose to set up a third stateful firewall dedicated to this server. Between the recent plethora of sendmail vulnerabilities and the fact that there is no source address limit on port 25 traffic to this server, I definitely wanted the opportunity to compare the logs of the firewall to the logs of the server itself. Also, the idea of running a mail server with a NAT'd address seemed like a smart way to proceed.

That Nutty Catalog Server

Finally, I created a fourth iptables-based firewall and put it in front of my catalog server. Rather than take the risk that one of my security tweaks on the server itself would disturb the functionality of that system or raise the ire of the vendor contracted to care for it, I decided to carry my security measures right up to that machine's network interface and leave the rest alone. The only host-based work I did on that server was to assign it a NAT'd address and install Tripwire for the sake of file integrity. While I couldn't uninstall unneeded services I could definitely block all access to the server except to those ports where I absolutely knew a relevant service lived. I also logged every connection, successful or not, to this firewall and every connection that was forwarded to the server. The idea here was to record everything I could so that if and when that host was ever compromised I'd be able to present this evidence to the vendor and my constabulary in an effort to sway the vendor to a more secure way of thinking – and administrating.

Workstations

I'm grateful to say that the installation of stateful firewalls mitigated many of the risks to the workstations. Because they were all using non-routable (NAT'd) IP addresses now, they were not connecting directly to the Internet. If any of those Windows machines managed to contract a server-based worm, it wouldn't be very effective since the firewall would block any ingress to it. The stateful firewalls also dropped any traffic to the workstations that wasn't specifically related to established internal connections, so many port scans and outright attacks would not reach the workstations themselves. By using a firewall on both the public and staff subnets each was separate from the other and never saw any traffic unrelated to its subnet. My coworkers discovered two powerful software packages for the public workstations that I gave the big thumbs up to as host-based intrusion detection tools. The first, WinSelect[5], acts as a third-party policy editor for WIN9X computers. This program makes it easy to define access limitations to the Control Panel and such via a password. The second, called DeepFreeze[6], and is an amazing host-based tool for Windows. In a nutshell, as DeepFreeze is installed it takes a snapshot image of the hard disk. No matter what a user does to the disk, including a complete reformat, when that machine is rebooted it restores the image it recorded earlier.

AFTER

(Please see Appendix III for a diagram of the network after security enhancements were put into place.)

The new additions of a central logging server provided me with accurate way to monitor my gateway, servers, and firewalls, all from one spot. I now know for certain that many attacks are being stopped at the gateway; and when things get through, the logs of the firewalls give me enough detail to figure out WHY things got through and what I can do to tighten up the rules. Because I keep these logs archived for a year, I'll have a means to determine how a machine gets compromised when that happens in the future.

Stateful firewalls brought a measurable level of protection to my network. Servers and workstations are shielded from most attacks from the Internet – I know this from comparing logs. The NAT'd addresses that servers and workstations now have make many worms ineffective, which is something I can tell from the firewall logs. I know that staff cannot get to IM and P2P services because the paths are blocked at the staff firewall, while the public has access to these services on their workstations because their firewall is more permissive.

The purchase of network switches and the separation of staff and public subnets, both through VLAN's and separate firewalls, has virtually eliminated the dangers presented by network sniffing. Sniffers on the public network only see traffic from other public workstations while staff workstations are not visible, which I can prove by running a floppy-based sniffer at any client. The managed switches give me tremendous flexibility for adding future VLAN's as well as the ability to add an intrusion detection sensor by mirroring a port if I want.

Because of the work I did on my Linux servers and NT box they are subjected to fewer network attacks. The logs indicate this, as do the additions PortSentry makes to the hosts.deny file in reaction to port scans. The services my servers run are easier to patch because there are fewer of them. Connections to these machines are logged giving me a much better idea of how these resources are used by staff and what mischief slips through the gateway router. Tripwire informs me of any unauthorized file changes on these severs so if and when they *are* compromised in the future I stand a good chance of catching it. Finally, users must choose strong passwords and rotating them on a regular basis because of the way I configured the password policy on these servers.

On my public workstations, WinSelect helps keep patrons from messing with the configurations of the workstations. If a patron somehow manages to get past WinSelect, any software they install or viruses, worms, or trojans they download will be eradicated when the computer is restarted. This struck me as the ultimate not in intrusion detection, but in intrusion *elimination.* In addition, DeepFreeze allowed me to completely preserve patron confidentiality through total deletion of their activities when the machine is restarted.

My staff workstations now use NAT'd addresses that make P2P much harder to operate and worm infections with servers in them ineffective and easier to catch through log analysis. These workstations are shielded from the public network and from most Internet attacks on Windows shares and the like by their stateful firewall. I learned in GSEC that I could also deploy some sort of host-based firewall like Zone Alarm on these workstations and even go so far as installing Tripwire for Windows on the workstations, which I may do in the future.

Conclusion – Where Do We Go From Here

My network still has some areas of vulnerability. I would like to continue working towards a default deny policy on the gateway router. Doing this would block a lot of noise right at the perimeter of my network and without such a policy, we risk new exploits getting through. I'm presently logging only dropped packets and the default deny would give me a better picture of what we are being subjected to than the next layer (the stateful firewalls) may be telling me in logs.

I also want to find a way to convince our catalog vendor to use more secure methods of administration. I hope to gently require that they use SSH to administer our server and limit access to administrative services based on source address when they provide a complete list. Even as I write this, my vendor has decided to route all their tech support access through a single NAT'd address and is playing with a Windows-based SSH client, so the future looks good.

It's one thing to make an attempt to better an insecure situation through some prudent actions, but the ability to monitor these changes to check for effectiveness – or make sure your actions are actual improvements – makes for a very satisfying and effective experience. My GSEC class did more than expose me to the threats, it showed me a variety of tools and techniques I could use to mitigate the problems.

APPENDIX I

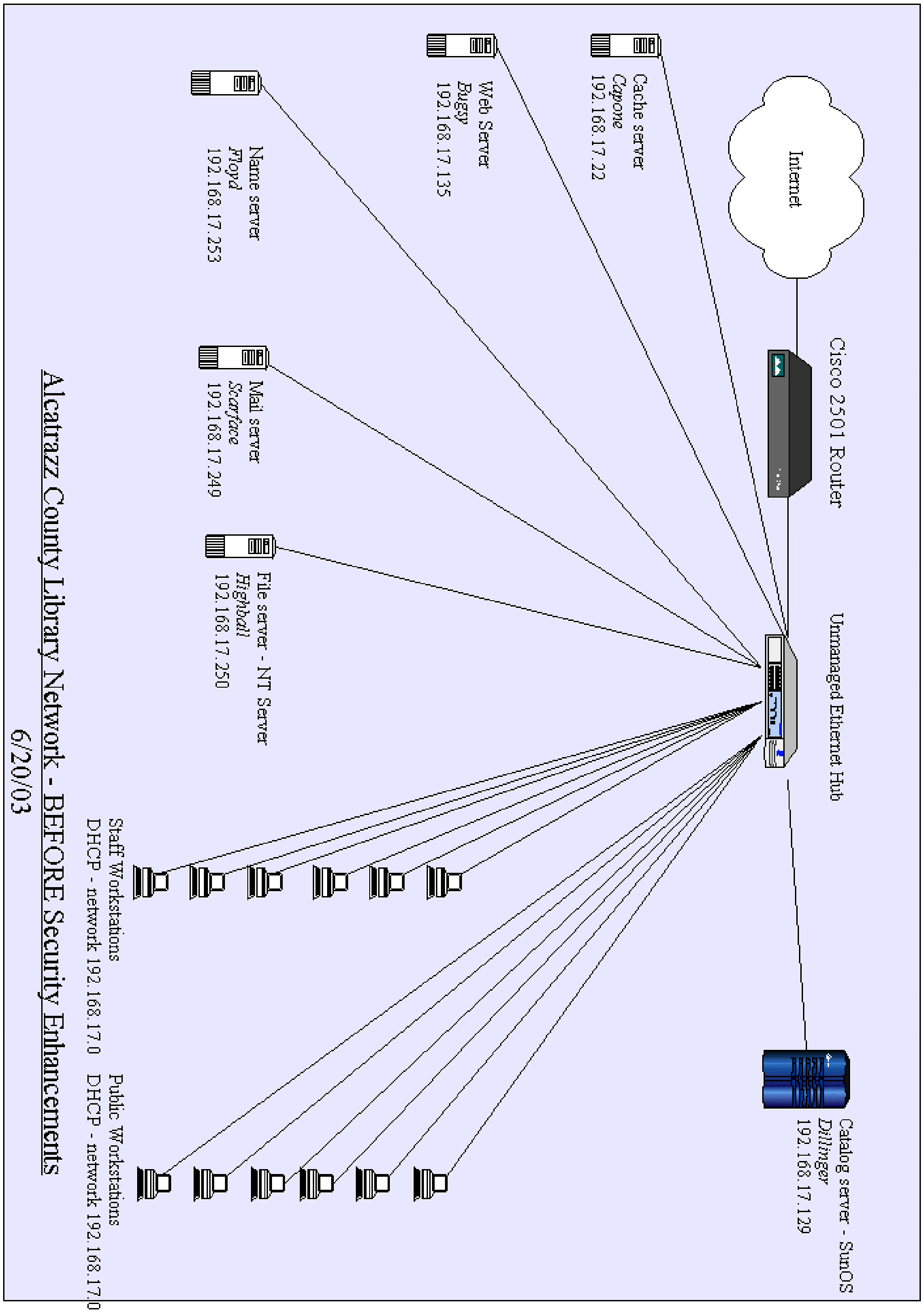
IPTables vs. IPChains Comparison[7]

Here is a basic table of comparison between iptables and ipchains.

	IPTables	**IPChains**
Match Packets based on:		
Src/Dest Address	*	*
Src/Dest Port	* (Single Port, Port Range, Multiple Port+)	* (Single Port, Port Range++)
TCP Flags	*	*
Input Interface	*	*
Output Interface	*	
Protocol (TCP,UDP, ICMP)	*	*
ICMP Type	*	*
Fragmented Packets	*	
Packet Match Limits	*	
Packet Marking	*	*
Output source†	*	
Connection State††	*	
Type of Service bit	*	*
Packet Destinations:		
ACCEPT (Pass)	*	*
DROP	*	*
REJECT with †††	*	*
ICMP Net Unreachable	*	
Host Unreachable	*	
Port Unreachable	*	
Protocol Unreachable	*	
Network Prohibited	*	
Host Prohibited	*	
Echo Reply	*	
TCP Reset	*	
LOG	*	*††††
On Log Level x	*	
prefixed by text	*	
Include TCP sequence	*	
include TCP options	*	
include IP options	*	
MARK	*	*††††
MIRROR•	*	
REDIRECT	*	*
Many other targets not available in ipchains	*	

+ = Multiple Ports: A set of up to 15 nonconsecutive ports (i.e. 1,3,5)
++ = Port Range: A set of consecutive ports (i.e. 1 to 10)
† = Valid only on the output chain from the machine running iptables.
†† = States: Established, Related, New, and Invalid
††† = Valid Rejection types in iptables: Network Unreachable, Host Unreachable, Port Unreachable, Protocol Unreachable, Network Prohibited, Host Prohibited, Echo Reply, TCP Reset
†††† = Available as a command line switch, not as a packet target.

APPENDIX II

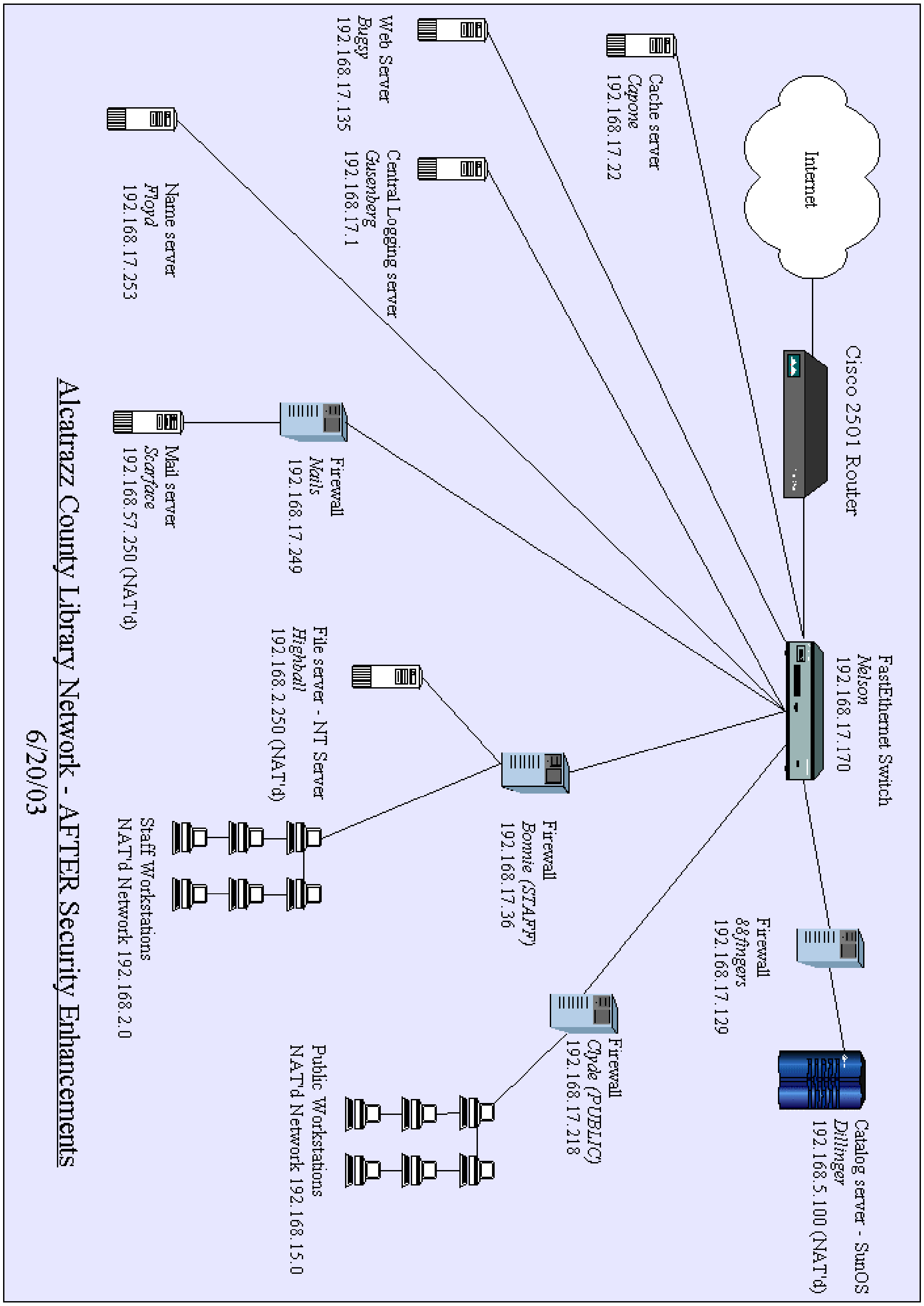


Alcatrazz County Library Network - BEFORE Security Enhancements

6/20/03

(Unless otherwise noted, all servers run RedHat Linux)

APPENDIX III



Alcatrazz County Library Network - AFTER Security Enhancements

6/20/03

(Unless otherwise noted, all servers run RedHat Linux)

REFERENCES:

[1] Hines, Eric. “Complete Reference Guide to Creating a Remote Log Server.” Linuxsecurity.com. 22 August 2000. URL: http://linuxsecurity.com/feature_stories/feature_story-64.html (18 June 2003).

[2] “Project Info - Ntsyslog.” Sourceforge.net. 21 October 2002, URL: http://sourceforge.net/projects/ntsyslog/ (18 June 2003).

[3] Schroder, Carla. “Who's Got Root? Installing and Configuring Tripwire” Earthweb.com. 31 December 2002, URL: http://networking.earthweb.com/netsecur/article.php/10952_1562771_2 (18 June 2003).

[4] Godot. “gShield.” URL: http://muse.linuxmafia.org/gshield.html (June 19 2003).

[5] Hyper Technologies, Inc. “Winselect.” URL: http://www.winselect.com/ (24 June 2003).

[6] Ibid., http://www.winselect.com/pages/deepfreeze/dpfrz_info.htm?B13=More+Info

[7] Ballard, Josh. “IPTables vs. IPChains Comparison.” Oofle.com URL: http://testweb.oofle.com/iptables/comparison.htm (18 June 2003).