GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

Case Study: Home Network Redesign

GIAC Security Essentials Certification (GSEC)
Practical Assignment
Version 1.4c
Option 2

Submitted by: Nate Wilson

February 18, 2005

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.

Table of Contents

© SANS Institute 2000 - 2005, Author retains full rights.

Abstract:

Six years ago, I signed up for an ADSL account with GTE. At the time I thought, if I'm going to be constantly connected to the internet, I'd better do something to secure my computer. My solution was to download a Linux distribution (Red Hat 6.0, if I recall) and research how to setup a firewall. In the six years that have passed, my network has been patched and added onto so many times that…well…let's just say security isn't at the level that it should be. When it comes to security, I have the typical hard candy shell and soft gooey center. What I've really needed to do (for a number of years now) is to wipe the slate clean and redesign my network from the ground up. So my goal with this practical is to take a fresh look at my network to make it secure, while maintaining functionally and to do so for as little money as possible.

Before:

As I said earlier, my network has evolved over the past six years. Before we get into how to organize and secure my network, I'd like to take a little time to explain how it looked at the beginning of this project. Figure 1 shows a picture of my network.

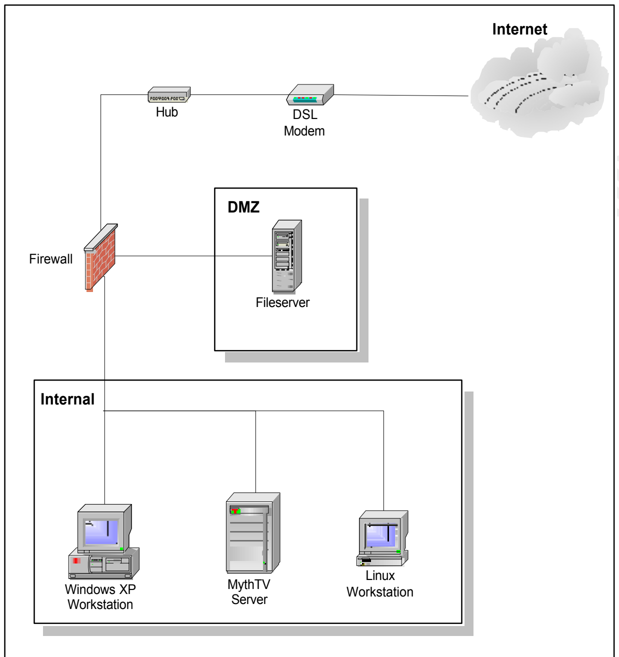


Figure 1

Let's take this one piece at a time, starting with the firewall and I'll explain how the servers are configured.

Firewall

The firewall is one of the only machines that I've had in place since the beginning, therefore, it's the one in the biggest need of an overhaul. First of all, the operating system on the firewall is Red Hat 9. The firewall has three real network interfaces (Internet, DMZ, and internal) and three virtual interfaces (used for VPN tunnels). The firewall I use is iptables. When I first set it up, the rule set was pretty simple. Internal workstations were allowed to go out to the internet on any port. All ports in from the internet were denied. Lastly, the firewall was allowed to get out to the internet on any port. It was never the most secure rule set, but at the time, it was a fairly standard/simple setup. It started out at about half a page, but over the years my firewall rule set has grown into three and a half pages of firewall rules that were needed at one time or another. Specific ports are forwarded to my internal workstation to support peer to peer software that I haven't used in two years. I still have rules for VPN tunnels that no longer exist. Traffic is still dropped from IPs that were infected with nimda years ago. The list goes on and on. My rule set still works, but it's hardly efficient or secure by today's standards (more on that later). The firewall also handles VPN functions for me. I use VPN tunnels to connect to the home networks of various friends and family members. To create these tunnels I use both FreeS/Wan and openvpn. FreeS/Wan is a more solid and reliable VPN application, but it only runs on Linux operating systems and it's more difficult to configure. This is where openvpn really shines. Openvpn can be installed on Windows, Unix (BSD & Solaris), Linux, and OS X. It is very easy to setup and it supports road warrior (client/server) setups. So, I mainly use openvpn to connect to people who don't have a Linux gateway installed. The only other function of my firewall is to act as a DHCP server for my internal network.

Fileserver

Next let's take a look at the file server. It is connected to the DMZ interface on the firewall via a crossover cable. This machine's OS is Fedora Core 1. The file server is my primary file store and runs servers for both NFS and SMB traffic. It also serves as my web server, mail server and an internal DNS server. The firewall forwards web (http/ssl) and sendmail requests from the internet to this machine. For web services I use Apache and OpenSSL. I run both an IMAP server and sendmail server for my e-mail. My internal DNS was originally setup before I owned any domain names, so I chose to use wilsonnet.com (knowing that I'd never be able to get to the real wilsonnet.com on the internet). For external DNS, my DNS server just forwards requests to my ISP's DNS servers.

MythTV Server

The only other server on my network is my MythTV box. Its OS is also Fedora Core 1. Its only job is it runs MythTV, which is an open source PVR application. This machine records TV programs and serves them up to other MythTV front end boxes on my network. This machine also runs apache web server for web administration of MythTV.

Workstations

Internally on my network, I have two workstations. One of these has Windows XP for its OS and the other has Fedora Core 2. There's nothing special about either of their configurations, they're just your run-of-the-mill desktop machines. They are primarily used for playing games, web surfing, and checking email.

© SANS Institute 2000 - 2005, Author retains full rights.

During:

Now that I've described my network's setup, let's go through the problems with its design and some of the ways I hope to fix these issues. I'll start off talking about some general policy changes that I have implemented across my entire network. Then I'll cover machine specific configuration problems.

General Policy Issues

Choices that I made years ago have slowly, over time, become policy on my network. Many of these choices were made because they were the easy thing to do, not because they were the most secure option. As part of my redesign, I want to revise a few of these and help eliminate the soft gooey middle of my network.

The first major policy I want to implement on my network is the policy of least privileged access. This basically means by default, all traffic is blocked. Only necessary traffic will be permitted. When writing firewall rules, I won't consider any networks to be trusted as I did before. This includes traffic coming from or going to any internal machine, which leads to the next problem in my previous design.

None of my machines (with the obvious exception of my firewall) are running a firewall. Host based firewalls have started growing in popularity over the last couple of years. With that in mind, I'll be using firewalls on all of my internal hosts, which, along with the concept of least privileged access will really help in hardening my internal network.

Another decision that was made years ago, for ease of implementation was using passwords for SSH and shared secrets for VPNs rather than using keys. My goal is to convert all SSH servers and VPN tunnels to use keys.

Using keys for SSH sessions, turned out to be so simple that I can't believe I never bothered to do it before. All you have to do (on Linux) is run ssh-keygen –t rsa (or dsa for a DSA key) and enter a passphrase for the key. This will generate a key pair in your home directory under .ssh. Then you just take your public key and move it to all the other machines you want to connect to under your home/.ssh directory and name it authorized_keys2. Restart sshd and you're ready to go. Now you can ssh to the machine using keys instead of passwords. As an added security measure, I also went into the sshd_config file and set PasswordAuthentication to no. This disables logging in to SSH using passwords.

Setting up keys for VPN tunnels proved to be a little more complicated than SSH. Basically, a key pair needs to be generated for each end of the tunnel. With freeS/Wan the command to generate a key pair looks like this: "ipsec newhostkey --output /etc/ipsec.secrets". Next you have to export your public key (the machine you're connecting to needs to have your public key). In freeS/Wan the command to export the public key is: "ipsec showhostkey --left > /tmp/left.pub". FreeS/Wan uses left and right to signify the two ends of the tunnel. Once you have your public key exported, you need to get the other end's public key and put both of these into your ipsec.conf file under the appropriate connection settings. The only other configuration change needed to convert from a pre-shared secret to keys is to change the authby configuration under ipsec.conf from secret to rsasig. Restart the tunnel and it will come back up using keys to authenticate the end points.

As I said earlier, at the time I started setting up my network, I didn't own any domain names so I used an existing domain name for my internal network. This isn't really that big of a security concern, but it is just bad practice. In hind sight, I should've at least used an invalid top level domain (I've seen many examples using cxm instead of com), that way, if the request did make its way on to the internet the root DNS servers would reject it. There have been a couple times, in the past few years, when my internal requests made it on to the internet and received valid IPs from the wilsonnet.com servers (it seems their DNS servers are configured to respond to any request with one IP address). Depending on what application was making a DNS request, this could have resulted in a security breach (sensitive data could have been sent to the real wilsonnet.com server rather than my internal server). So to avoid all these issues, I have reconfigured my network to use the domain dotfarm.org because I own it and control its IP addresses.

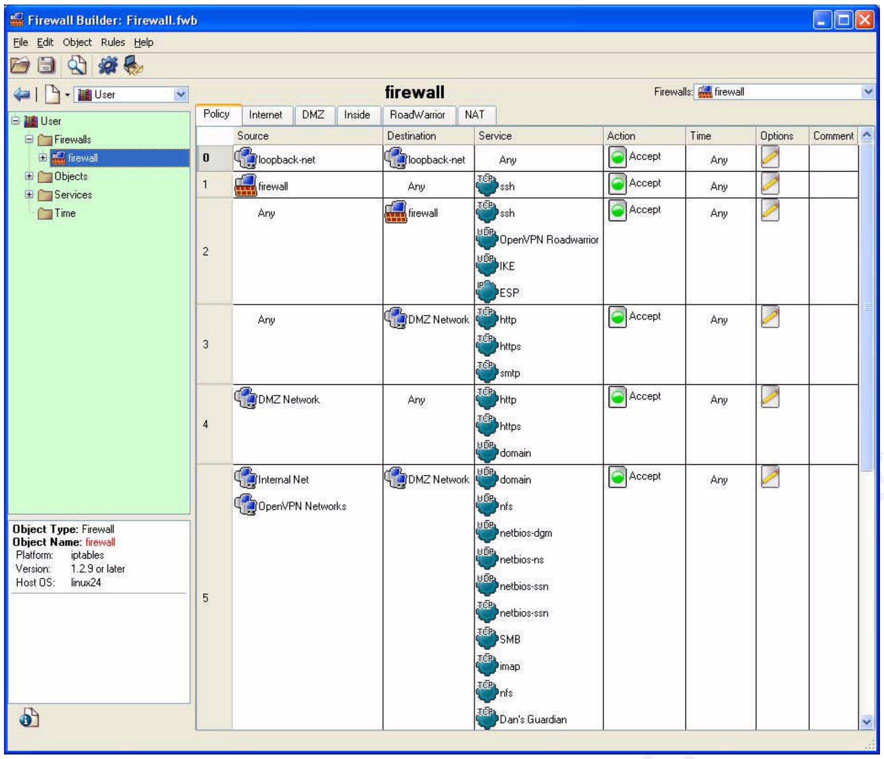
The last major policy change I made was to the network settings of the workstations. Previously, all of my machines could route out to the internet. They all had the firewall listed as their default gateway and the firewall was configured to perform IP masquerading for all internal hosts. I decided to change this, at least for internal workstations. By eliminating the default route from an internal workstation, it's unable to talk to servers that are not on its subnet. This significantly increases security on the most likely source of infection on the network. I'll get into how this is done and what the pros and cons of it are when I discuss workstations later on.

Firewall

Now that I've covered the general policy changes that I plan on making, I'll move on to the firewall and take a look at what changes I need to make to it. First, let's look at the OS, Red Hat 9. It's been unsupported for almost a year now and needs to be upgraded. I've been using Red Hat since version 5.1 and I'm very familiar with their distribution, so I'll stick to the Red Hat Fedora line. Up to this point, I've avoided using the 2.6 kernel on my servers, but it's been out for long enough now for most of the kinks to be worked out so, the time seems right to make the jump. That said, I decided to go with Fedora Core 3 on my firewall. Next let's discuss the firewall application itself. Since I'm running a Linux OS, it only made sense to stick with iptables, but my rule set could definitely use an overhaul. Six years ago, when I originally implemented the rule set, I didn't see anything wrong with allowing the firewall itself and all of the internal machines out to the internet on all ports. Trojan horses and spy ware just weren't as big of an issue back then, but times have changed and it's about time my rule set did too. My solution was to use the concept of least privileged access. Basically, I've allowed traffic only if it's needed, but by default I've denied everything. This will definitely result in more administration on my part (I'll need to add rules for all valid traffic), but it will also stop a lot of traffic that I don't want going across the wires. This concept is used for all traffic that crosses zones of trust (in other words, anytime traffic goes between two interfaces on the firewall). I've come up with the following matrix to show which ports are permitted between two given interfaces.

From:	->eth0 (Internet)	->eth1 (DMZ)	->eth2 (Internal)	->FreeS/Wan (VPN)	->openvpn (SSL-VPN)
eth0 (Internet)	XXXXXXXX	25, 80, 443	XXXXXXXX	XXXXXXXX	XXXXXXXX
eth1 (DMZ)	25, 53, 80, 443	XXXXXXXXX	XXXXXXXX	22, 80, 137, 138, 139, 445, 2049	XXXXXXXX
eth2 (Internal)	XXXXXXXX	22, 25, 53, 80, 137, 138, 139, 143, 443, 445, 2049, 8000	XXXXXXXX	22, 80, 137, 138, 139, 445, 2049	22, 137, 138, 139, 445
FreeS/Wan Interfaces (VPN)	XXXXXXXX	22, 25, 53, 80, 137, 138, 139, 143, 443, 445, 2049, 8000	22, 80, 137, 138, 139, 445, 2049	XXXXXXXX	XXXXXXXX
openvpn Interfaces (802.11 VPN)	XXXXXXXX	22, 25, 53, 80, 143, 137, 138, 139, 445, 443, 2049, 8000	22, 80, 137, 138, 139, 445, 2049	XXXXXXXX	XXXXXXXX

In this matrix, the left column represents the interface that traffic is coming in on. The other columns represent the destination interfaces and the cell where the two interfaces intersect shows what ports are permitted. So, for example traffic coming in from the internet is permitted to talk to the DMZ (file server) on ports 25, 80, and 443. When it came time to write the firewall rules that this table represents, I found out pretty quickly that the rules were going to be very complicated and there were going to be quite a few of them. It turns out accounting for all network traffic is more difficult than I first imagined. Still, I think this policy is worth the effort of figuring out what all the rules should be. So, rather than writing five or more pages of firewall rules by hand I decided to look for a GUI that would help give me a handle on my rule set. A friend of my recommended fwbuilder, it's a GUI frontend for building firewall rules for iptables, ipfilter and Cisco PIX firewalls. The GUI will run on Linux, Windows, or OS X and it works with command-line SSH clients to copy the firewall script down to the firewall and execute it for you. Fwbuilder comes with standard services for the most commonly used protocols and it allows you to add hosts, networks, and IP ranges. Once you have all of your objects, building rules is a simple matter of dragging and dropping objects and services to build rules. It is a great application for newbies and I also highly recommend it for very complicated rule sets. Since the rules are much easier to read in the GUI, future administration will be quick and simple. To give you an idea of what fwbuilder looks like, here is a screenshot of my firewall policy:


Firewall Builder: Firewall.fwb
File Edit Object Rules Help
User
Firewalls
firewall
Objects
Services
Time
firewall
Firewalls: firewall
Policy Internet DMZ Inside RoadWarrior NAT

	Source	Destination	Service	Action	Time	Options	Comment
0	loopback-net	loopback-net	Any	Accept	Any		
1	firewall	Any	ssh	Accept	Any		
2	Any	firewall	ssh OpenVPN Roadwarrior IKE ESP	Accept	Any		
3	Any	DMZ Network	http https smtp	Accept	Any		
4	DMZ Network	Any	http https domain	Accept	Any		
5	Internal Net OpenVPN Networks	DMZ Network	domain nfs netbios-dgm netbios-ns netbios-ssn netbios-ssn SMB imap nfs Dan's Guardian	Accept	Any		

Object Type: Firewall
Object Name: firewall
Platform: iptables
Version: 1.2.9 or later
Host OS: linux24

This screenshot only shows about half of the total policy, but by comparison, when I started to write out the rules by hand, I was up to well over two pages of firewall rules (with comments) and I was just getting started. Had I continued writing out the rules manually, I would've been left with a rule set that was very large and difficult to manage. With fwbuilder, I can quickly find what I'm looking to modify.

Fileserver

The file server was in a lot better shape than the firewall. I'd reloaded it fairly recently and, at the time I set up sendmail and apache, I tried my best to make sure they were secure. There is, however, still room for improvement. As with all other machines, getting the firewall running is a good place to start. The file server needed to accept web and sendmail traffic from everyone and SSH, DNS, IMAP, NFS, and SMB traffic from internal hosts. So, I configured iptables on the fileserver with a rule set to allow only these ports and deny everything else. The rules here are short and simple, so I wrote them out by hand (as opposed to using fwbuilder).

Next, I rechecked the sendmail configuration to make sure that it was still secure and would not allow unauthorized relaying of mail. I found a web site (http://www.abuse.net/relay.html), a few years ago, that does a pretty thorough check of your email server. All you have to do is enter the address of your email server into the web form and their script will try a lot of different methods of relaying off of your mail server. This site is an easy way to make sure your sendmail configuration is secure against spammers, who would like nothing more than to have your server do their dirty work.

The last measure I took on my fileserver was to make sure the applications on the file server were fully patched and up to date.

© SANS Institute 2000 - 2005, Author retains full rights.

Myth TV server

Much like the file server, my MythTV server has been reloaded within the past six months, so it started off in pretty good shape. Once again, the rules here are pretty simple, so I wrote them out manually. They are configured to allow web traffic from anywhere and mysql, NFS, SMB, and MythTV streaming from internal hosts. The MythTV application was already fully up to date, but I had to update all other applications on the box.

Windows Workstation

My internal workstations underwent the biggest changes of all. First of all, my host based firewall policy applies to workstations as well as servers. On my Windows XP workstation, I used Microsoft's firewall that's built into service pack 2. In addition to blocking all inbound requests, I wanted the Windows firewall to be an application aware firewall. Basically, I wanted it to block applications that I had not permitted to talk on the network. Microsoft's SP2 firewall does a good job of prompting whenever a new application tries to make a request over the network. This way, if a trojan or virus tries to generate network traffic, Windows will let me know. The one problem with this is, some trojan and virus writers have become clever enough to replace iexplorer.exe with their trojan, knowing that iexplorer.exe will be allowed out through the application firewall.

This is where my next change helps out. Before, my internal machines had their default route set to the firewall's internal IP address (192.168.12.65). The default route tells a machine where to go if it wants to talk to a server that's not on its subnet. I removed this default route, so internal workstations don't know how to reach machines that are not on their subnet. I've added a static route to the 192.168.12.0 network, so internal machines know how to get to the file server, but to talk to the internet, workstations have to use a proxy server (squid) running on the fileserver. This way trojans and viruses that try to masquerade as iexplorer.exe will be unable to find any servers on the internet.

This leads to my next step for securing my Windows workstation. I have, in every way possible, disabled Internet Explorer on the workstation. Recently there have simply been too many high profile, high risk vulnerabilities in Internet Explorer. I just can't see going to all this work to secure my network and leaving that big of a hole on a key workstation. Sticking with my bias towards open source and my goal of doing this as cheaply as possible, I'm using Mozilla Firefox as my default web browser. There has been a lot of speculation that, given time, there will be just as many problems with Firefox as there have been with Internet Explorer. Only time will tell, but for now, I think it's the way to go. As I said, I've configured it to proxy web requests off of my fileserver on port 8000. I chose to put the proxy server on my fileserver, rather than on the firewall, because the fileserver is a faster machine and it has a lot more disc space for caching web pages. I've left Internet Explorer without a proxy server configured, so if anything does try to use it to get out to the internet it won't be able to find its way out.

Antivirus software is something that every user (at least every Windows user) should be running. I'm almost embarrassed to say, however, that my Windows machine didn't have antivirus software installed. I've used many different antivirus programs over the years (they usually came free with hardware I purchased), but every time the subscription ran out, I'd just stop using antivirus software until another free copy came my way. Accepting the fact that I'm unwilling to pay for antivirus software (admitting the problem is the first step in finding a solution), I figured I'd better find a free antivirus program, if I wanted to have any hope of having it operational at this time next year. Enter AVG Antivirus from Grisoft. AVG Antivirus is free to home users and I've heard some very positive things about it, so I thought I'd give it a try (hey, it's got to be better than nothing).

<u>Linux Workstation</u>

The Linux workstation has a similar configuration to the Windows workstation. It has no default route, and its browser has been set up to proxy off of the fileserver on port 8000. Its firewall, however, isn't application aware. I'm just using iptables and allowing ports 22, 53, 8000 and 2049 out bound and blocking all inbound ports. I looked around and I was unable to find an application aware firewall that I was happy with. I suspect that this is because trojans and viruses aren't as prevalent in the Linux world as they are on the Windows side, so this shouldn't be a problem. If malicious Linux software does become a problem, antivirus and application firewalls will follow shortly thereafter to fill the void.

<u>Wireless (802.11b/g):</u>

There are a couple of additional devices that I've added to my network during this redesign. The first is a wireless access point (WAP). My original plan was to add another real interface to the firewall (to create another DMZ) and use it to connect the access point to. The more I thought about this, the less sense it made. The access point has a built in firewall, so, assuming I installed a host based firewall on my wireless machines I would be triple firewalled. Maybe it's just me, but this seemed like overkill. I was already planning to limit the wireless machines access to my internal network. My plan was to give them basically the same access as someone connecting in from the internet. I'd then use openvpn in road warrior mode to allow my wireless machines to connect back into the internal network. I thought, rather than add the complexity of another interface and more firewall rules, why not just put the access point outside of my firewall and have it pull its own internet routable IP address. This way, if anyone manages to get connected to my access point, the only access they'll have is to the internet.

This gets into the second part of setting up wireless…securing the access point. In the past, I've considered setting up an access point with internet access and leaving it wide open for anyone to connect to. Putting the access point outside my firewall would be the best possible scenario for doing this. Unfortunately, there are too many legal implications in today's world for me to consider giving out free internet access (The last thing I want is to have to pay a $3000 fine because my neighbor downloaded a Metallica song off of the Kazaa network over my DSL line). So, instead of going the easy route and leaving it wide open, I did my best to lock it down tight. My plan was to use a layered approach in securing the access point. The first measure I used was MAC address filtering to limit what network cards my access point will talk to. This is only marginally secure because MAC addresses are easy to sniff and spoof. To really

© SANS Institute 2000 - 2005, Author retains full rights.

lock down the access point, I set it up to use Wi-Fi Protected Access (WPA) encryption. This encryption incorporates rekeying and is much more secure than WEP (the previous method of securing 802.11x wireless traffic). To get WPA to work with Windows XP, you have to either install service pack 2 or "Hotfix (SP2) Q815485".

Since I'm using the firewall that comes with service pack 2, I already had it installed, so I didn't need the Hotfix. Next I just had to update my router (a Linksys WRT54G) to the latest firmware version. It's now running on version 3.01.3 which supports WPA. Then under Wireless Security on my access point, I picked WPA Pre-Shared Key for the security mode and TKIP for the WPA Algorithm. For home users a pre-shared key is about the only option, unless you feel like setting up a radius server just to authenticate your wireless users. After that, I picked a long shared key and moved on to configuring the laptop. On the laptop, I went to the Wireless Network Connections Properties window and chose to add a network. Then I typed in the SSID for my access point and chose WPA-PSK for my network authentication and TKIP for my data encryption. All that was left to do was to type in the key and I was connected.

Proxy Server

When I built my network, my wife and I were the only users of it. However, now we have a two year old daughter and a son on the way. Even at only two years old, my daughter likes to go out to the Sesame Street homepage to play a little Elmo game. This got me thinking about the future when my kids will start using the internet more and more. I've used the web enough to know that bad stuff is only a couple clicks away. I've already talked about how I'm using a proxy server to help secure my internal workstations against trojans and viruses, but I'm also using it to help protect my kids against the bad side of the internet. I'm using squid as my proxy server. It is an excellent proxy server that supports a number of different plug-ins, and best of all, it's free. In my setup I'll be taking advantage of two add-on products.

Dan's Guardian is an open source content filtering program that works with squid. It looks at the content of web pages and uses that content to determine if the page is allowed or denied. It's filtering is based on meta data, MIME filtering, as well as phrase filtering that looks for profanity and pornographic phrases. This allows it to catch new web sites as well as previously known ones. Many commercial products, such as Cyber Patrol and Net Nanny, use a database of lists to categorize known sites. The problem with this method is similar to the problem antivirus companies currently have. If new sites come out that they haven't added to their list yet, the sites will be permitted. Dan's Guardian doesn't use lists, so it doesn't have this problem. The one problem it does have is a greater likelihood of false positives, but in my scenario, I'm better safe than sorry. Dan's Guardian has quite a few other features, such as preventing uploading to the internet and file extension blocking, but the main function I want it for is blocking pornographic sites. Setting up Dan's Guardian is pretty simple. You download the source code and run a standard install (configure, make, make install). After that you can edit the config file if you wish (I changed the port mine runs on to 8000 and removed file extension and mime type blocking, but aside from that I left it at the defaults). Assuming you already had squid installed, you just start up the Dan's Guardian daemon and point a browser at it's port and start surfing.

The other squid add-on I'm installing on my fileserver is Ad Zapper. This is a very simple to install squid redirect script. The squid proxy has the ability to redirect web pages to add-on products before they are passed back to the client machine. This allows the add-on to modify the page, in this case remove the advertisements. Ad Zapper is a simple perl script that you download, chmod to make it executable and then point squid to in your squid.conf file. That's really all there is to setting it up. From then on, when you pull down web pages through your proxy server the ads are replaced by an image that says "this ad zapped". One could argue that zapping ads doesn't really have anything to do with security, but in addition to ads, Ad Zapper will also remove links to known web trackers, so I thought I should mention it here. The less people on the internet know about my machines the safer I feel.

After

My main goals, when I started this effort, were to harden the center of my network, to strengthen the border and to do it all at as little monetary expense as possible.

I have worked to harden my network against attacks from within. I have done this by treating all networks as untrusted. Outdated operating systems have been upgraded to newer, supported ones. I have checked to make sure all applications are up to date and fully patched. My internal domain name has been changed to a name I own and control the name servers for. SSH sessions have been hardened by requiring keys with passphrases. Finally, I have taken many steps to make my network more resistant to malicious software (mainly trojans and viruses). Internal workstations can't find the internet (except through a proxy server), so malicious software won't be able to phone home. I patched my workstations to remove known security vulnerabilities. My Windows workstation now has an application firewall, which will limit network access to a few applications that I have permitted. I have also installed Mozilla Firefox which, at this point in time, is much less susceptible to malicious software. Finally, I have antivirus software installed which will detect and remove trojans and viruses that do make their way in.

In addition to hardening the center, a lot of the changes have also made my network more resistant to external attacks. Some of these are the same changes that helped harden my network against internal threats, like upgrading OS versions, patching applications, and requiring keys for SSH sessions. In addition, I have also converted all of my VPN tunnels over from using preshared secrets to using RSA keys. Lastly, I have rebuilt my firewall rules to eliminate the dead wood. Before my firewall had a lot of holes punched through it from the internet, and most of these holes were no longer used or needed. In this effort, I have wiped the slate clean and limited access to only the necessary ports.

There have also been some changes that I have made that increase the functionality and usability of my network without compromising the security of it. The biggest of these was adding a wireless access point to my network. This allows me to securely access my network from anywhere in the house through the use of WPA and openvpn. The other functionality improvements were made on the newly installed proxy server. These are Dan's Guardian and Ad Zapper. Both of these help add security to my network, while also increasing its usability.

All of these changes have helped increase the security of my network, but there are always trade offs to increased security. As a result of these changes, some applications simply won't run on my internal workstations. Some will work, but only after additional firewall rules or static routes have been added. By locking down my network, I have definitely increased the amount of administration that will be required in the future. Luckily, I have also found tools, such as fwbuilder and Ad Zapper that help balance out the cost/benefit equation. Even after all these changes, my network is by no means bullet proof. There are still holes that could be exploited . . . there will always be holes that can be exploited. For instance, a Firefox exploit could install a trojan that masquerades as firefox.exe and makes connections out to the internet over an HTTP proxy server. A trojan like this could send sensitive data out of my internal network. For the most part, however, I think I've eliminated the major holes in my design and I have definitely raised the bar for would-be crackers.

When I set out to secure my network, I didn't plan on going as far as I did. It was a lot of work getting to this point, but in the end I think it will be well worth it. One of the most interesting parts of this project, to me, is that I was able to drastically improve the security of my network at no cost (other than my time). I think this shows, good network security is available to any person/business, if you're willing to invest the time. I've heard the joke that, "Linux/open source software is free…if your time is worthless. Personally, I like to think of it this way…knowledge is priceless. Six years ago when I embarked on my first journey into the world of network security, by building a firewall, I set myself on a course where I would end up learning more than I did in four years of computer classes in college. For the past few years, I have simply spent too much time at the rest area. This project helped me get back in the drivers seat and continue the journey.

© SANS Institute 2000 - 2005, Author retains full rights.

References

Aaldering, Dave "SSH with Keys HOWTO" Dave Aaldering's Personal Home Page
18 Feb. 2004 <http://www.puddingonline.com/~dave/publications/SSH-with-Keys-HOWTO/document/html-one-page/SSH-with-Keys-HOWTO.html>

Andreasson, Oskar "Iptables Tutorial 1.1.19" Internet FAQ Archives
18 Feb. 2004 <http://www.faqs.org/docs/iptables>

Barron , Daniel "DansGuardian Brief Installation Guide" DansGuardian True Web Content Filtering for All
18 Feb. 2004 <http://dansguardian.org/downloads/installation2.html>

Bowman , Barb "WPA Wireless Security for Home Networks" Microsoft
18 Feb. 2004 <http://www.microsoft.com/windowsxp/using/networking/expert/bowman_03july28.mspx>

"Firewall Builder User's Guide" Firewall Builder
18 Feb. 2004 <http://www.fwbuilder.org>

Gibson , Steve "GRC Port Authority – Interactive Internet Database" Home of Gibson Research Corporation
18 Feb. 2004 <http://www.grc.com/PortDataHelp.htm>

Harrison, Peter "Configuring Linux VPNs" Linux Home Networking
18 Feb. 2004 <http://www.siliconvalleyccie.com/linux-adv/vpn-linux.htm>

Laverty, Denis "WPA vs WEP: Why is WPA better than WEP?" OpenXtra-Network Management Services Training
18 Feb. 2004 <http://www.openxtra.co.uk/articles/wpa-vs-wep.php>

© SANS Institute 2000 - 2005, Author retains full rights.

© SANS Institute 2000 - 2005 Author retains full rights.