Global Information Assurance Certification Paper

Scanning for viruses

GSEC CERTIFICATION

version 1.4b

option 2

by

Daniel Boyd

9 February 2003

Abstract

My first job position after college was with a company that needed someone to do some IT work. I was hired to design and implement a firewall as well as a virus scanning mail solution. We wanted all mail sent in and out of the company to be scanned. The challenge for me was not only lack of experience but gaining the trust of management.

SANS was my guide throughout this project. My goal was to do a risk evaluation taking into account business need then implementing the solution. Throughout the course of the project I learned the value of several security tools. I will step through the process I went through that increased security at this company. Part of the process included: interviews with employees, firewall selection through business need, firewall setup based on security and business requirements, implementing an IDS, and setting up a mail hub that would scan incoming and outgoing mail for viruses. Even after meeting the requirements of this security project, security was not guaranteed. This was my first project and some mistakes were made. I will address the mistakes and how they can be fixed. Given the limitations of the project and the learning curve the security of the network was increased.

Before

The company I worked for (called Company Widget) was not an IT company. It is a small manufacturing company that did not have a need or desire for remote access. After having problems with their ISP my supervisor decided it would be in the company's best interest to handle its own mail hosting. At the time I was hired I replaced a system administrator who was let go for hacking systems. It would be in Widget's best interest to host its own mail because it was using one drop off mailbox for the whole company at the ISP (around 50 employees). Fetchmail would request the mail from the ISP and then deliver it locally via Sendmail. Everything worked great using this system until somebody was carbon copied or blind carbon copied on an email. The ISP refused to add the necessary envelope headers in order to sort the mail properly. One solution would have been to get fifty email boxes at the ISP, for each employee. In the end it was cheaper to get a nailed up connection and have Widget do its own mail hosting.

At this point, Widget did have a firewall. The firewall ran Linux with ipchains. Nobody at the company was sure what policy was on the firewall. The rules had been downloaded from the Internet. All together the rule set was around forty rules long. We did not know what the firewall let through and what it did not let through.

RISK ASSESMENT

In order to build a firewall policy I would have to know what the risk is. Widget was a small company. In terms of the US market it is not well know such as

Microsoft or Boeing. I established what was going to be needed to show management what the risk was. The following is a list of compiled issues and problems I addressed to determine risk at Company Widget. I used Chris Brenton's Active Defense book as a guide when developing questions

1. How computer literate are the employees?
   Do most employees have trouble using a mouse or have some of them built a Beowulf cluster? When I started we had to beg employees to run a virus scan. Many of the employees where intimidated by running a virus signature update program.

2. What services need to be run on the network?
   Was there a need to ftp or telnet out? If so, who should have access to the services? The firewall was running the portmapper, rpc, and nfs demons. There was no need for a mounted network file system. This needed to be fixed.

3. What physical security is needed?
   Who has access to the server room and wiring closets? Who has root access and why? Many of the employees passed out their key code to open the entry door to anyone who requested it.

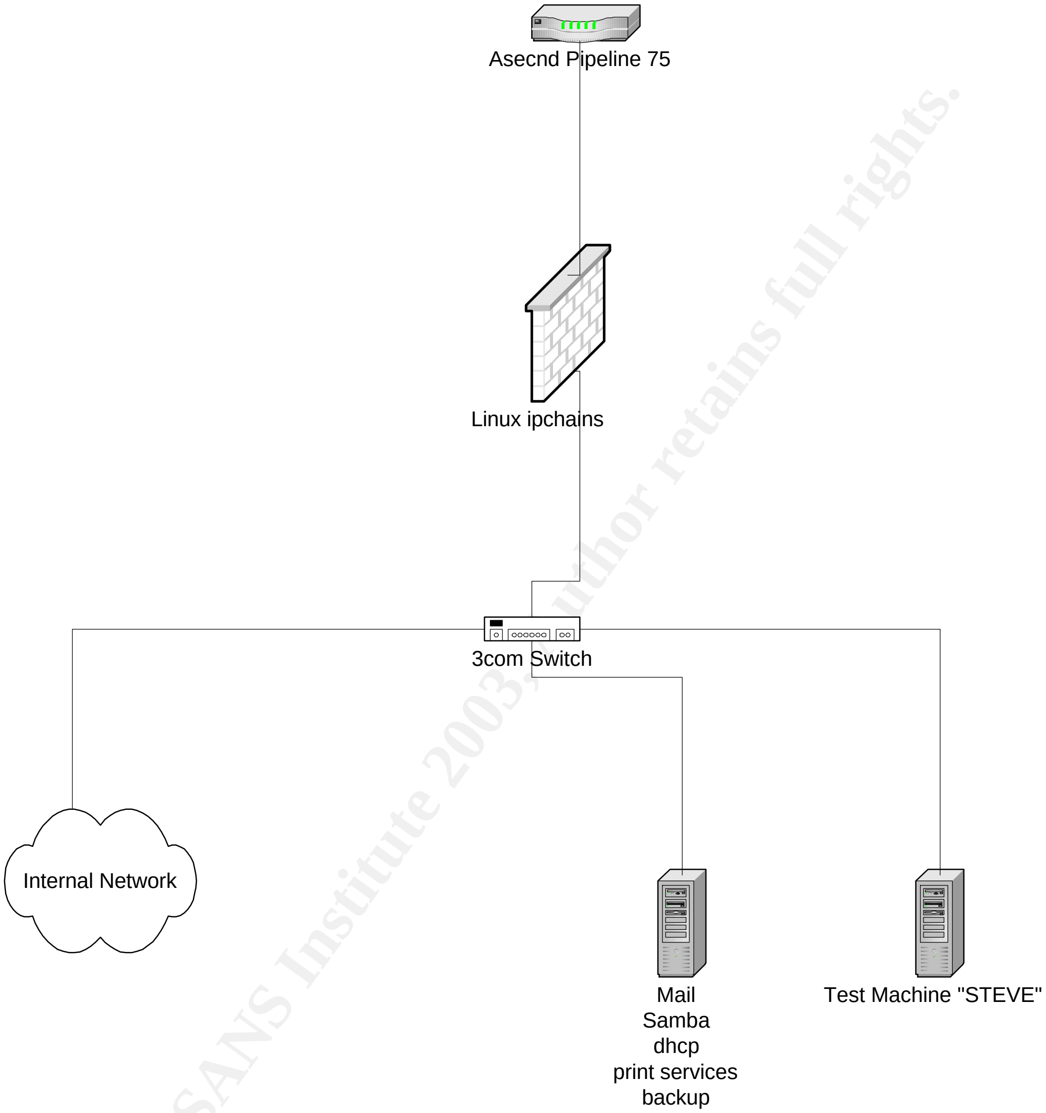
4. What is the value of data being secured?
   Should the servers be behind one or two firewalls? One server for the whole company handled dhcp, DNS, email, print services, and acted as a file server.

PROBLEMS

One of the limitations was a lack of experience on my part. I had to learn how to secure a network and justify it to management. It was for this reason that many of the issues with Widget's network were dealt with in phases. My priorities were the firewall and the mail relay, however internal auditing needed to be implemented. A few of the design and implementation decisions were flawed. Security was increased, however not as much as it could have been.

The router was another limitation. My supervisor had trouble with the ISP setting up the router. After many set backs, the router finally was configured by the ISP. When I was brought on board, the ISDN connection seemed to fail daily. When this was repaired we wanted to take no chances at the connection failing again. My recommendations and work were affected only in that I had to treat the firewall as the first line of defense. I knew that the router had security features however I would not be able to leverage it.

Layout of Network



At the beginning the firewall and the fileserver ran Red Hat Linux 6.2. Neither system had been updated with security fixes. My supervisor was computer savvy and liked the idea of Linux being free. The firewall was a 950 MHZ machine with 96 MB of RAM. It had two interfaces; one for the internal network, and the other for the Internet. The router was an Acsend Pipeline 75 router performing Network Address Translation. NAT allowed for many machines to be added to the network. Since I did not work directly with the fileserver or STEVE I do not have the specifications on their hardware. The main server ran dhcp, samba, sendmail, and print services. STEVE was an extra server on the network my supervisor used in case there was testing needed.

In summary, there was not a huge initial outside threat to this network. They did not have a permanent IP address. They did not host any particular service. They even had a firewall. The threat came because they were changing the network. They were going to have a permanent IP address. They were going to host their own email. My job was to ensure that there was a competent firewall protecting their network and that viruses were kept to a minimum. I would also have to point out any holes or problems I saw in the network.

During

I set out to answer my questions for the risk assessment. First item was to interview employees to see what their level of computer literacy was. Together with Human Resources I spoke with the CEO/VP and most users of the network.

Interviews allowed me to accomplish several things. The Risk Assessment called for a physical audit of the area. If I could gain the trust of the employees I will gain an extra set of eyes around the building. Security awareness would increase. Interviewing allowed me to see at what level the employees were at in understanding computers. Through interviews I could see which employees might try to play with the network. Not only did the interviews introduce me to the employees it also demonstrated to management that I was willing to step through a process and not make a wild recommendation.

I invited Human Resources to interviews for a few reasons. One was to have someone with interviewing experience in order to help read body language. Just graduating from college, it would help to have someone the employees might be able to relate with. By inviting Human Resources I have given the management a chance to be part of the decision making. Being fresh out of school this was a way to build trust and to show that the solution will fit business requirements. After reviewing the interviews I was ready to start building the firewall.

Firewall

Before we picked a firewall we needed to answer the questions to the risk assessment. First, through the interviews we decided that the employees as a whole were not computer literate. We could use this to our advantage. If we used a firewall that was not windows or did not have a GUI most of the employees would shy away.

Second we only needed to have one service available to the outside. Email was the only service that was needed from the outside in (port 25). Outgoing services needed were http (port 80/443), ftp (port 20,21), dns (port 53). and outgoing email (port 25). There was also a need to be able to upload updates to the firewall securely so we had to have ssh (port 22). Ssh was closely watched and only used when doing updates. It was addressed in the firewall policy given to management.

Physical security for the firewall was already in place. The firewall was in a server room secured with a keypad for entry. Only those with permission could enter the room and even fewer would have an account on the firewall (or would know what to do with it if they had it). Since we only managed one firewall the only way to access it would be through the console. Except for updates there was no need to be able to remote administer to the firewall.

Management followed my recommendation on the choice of firewall: iptables. My recommendation was based on several factors.

1. Linux/Iptables is free.
   My supervisor, having used Linux before, liked this idea. It also fit into the budget nicely.
2. The company currently used ipchains.
   Although Iptables is statefull, the interface and most of the syntax is the same.
3. There was no need for remote access.
   This company was small. No one needed to have access to the network off-site, hence, there was no need for a VPN at this time.
4. Based on the interviews there were not a lot of computer savvy employees.
   Linux is a little more difficult to understand than Windows. If we used a product that was not in mainstream there would be security through ignorance.
5. There was support on the Internet for iptables.
   If I needed support because I was still new to Linux it would be there.

I did look into a Cisco PIX firewall. The thought I had would be that an application firewall might be more secure than a full OS firewall, because all the application (black box) firewall will do is be a firewall. This idea was laid to rest rather quickly because of the cost and the fact that iptables would work fine. At that point in my career, never having experience with Cisco products caused the learning curve to be too large.

I had to decide where to place the firewall on the network. Unfortunately, I could only place one firewall and given the project scope that was going to be at the perimeter. I will discuss this later as a limitation to the project. I did decide given the one firewall that the Services network should have its own subnet directly off of the firewall. The internal network would be off of another interface on the firewall. The internal data should be behind at least two firewalls (it was not).

The next step was the firewall install. I only installed the packages that were needed for the firewall. This included: iptables, sendmail, ssh, and tripwire. . If a service is not installed on a system it cannot be exploited. Ssh was installed in order to allow uploads via the scp command for patches and bug fixes. Sendmail was installed just for host delivery of system messages. Sendmail was not configured as a mail server on the firewall. The firewall did not receive any messages through email from outside sources.

After the install I decided to make the default run level two. Level two shuts down all networking services and starts Linux in a single-user mode. By running the firewall in level two it allowed me another layer of security. If a service does find a way on to the firewall it would not start up by default.

To do this edit the /etc/inittab file

The line:

Id:3:initdefault:

Should be changed to:

Id:2:initdefault:

One basic user account was added to the firewall The account would be used to copy updates and patches to the system via scp.

The firewall has three interfaces: one going out to the router (the Internet), one for the Services network, and one for the internal network.

eth0: Internet interface: A.B.C.D

eth1: Service network interface: 192.168.2.1/24

eth2: Internal Network: 192.168.3.1/24

(A.B.C.D is a public IP address).

I needed a definitive list of what the firewall will and will not let in. I decided that the firewall will have to route incoming mail delivery to the mail relay on the Services network. Since iptables is stateful, established connections need to be allowed in. The firewall will need to allow the mail relay on the Services network to send mail to the internal mail server.

The internal DNS server would have to make DNS request out. HTTP and HTTPS needed to be let out too. FTP needed to be allowed out. A problem I had was that I had to let FTP out connected to a passive FTP server. A passive FTP server will change the data port during the FTP session. This is a problem because the port keeps changing throughout the transfer, meaning the firewall has to allow the transferring port out. Users connected to the mail relay directly for outbound mail. This decision was made because I had complete access to the mail relay but not the internal mail server. The firewall had to allow users to connect to the mail relay via port 25 for outgoing mail.

After I had made a list of services to let in and out I was ready to implement the firewall. The policy I ran, based on Zieglar's recommendation was a DEFAULT DENY policy(Zieglar, 32). This policy means that if a service is not specifically allowed in to the network it is dropped. By dropped it means that no response is sent to the user or system to signify that the packet is not allowed in. Default Deny policies are harder to write than Default allow because if the service is not specifically mentioned it will be dropped, Default Allow is the exact opposite. Default Allow is less secure because if a service is left out (not dropped) it is allowed in. There is more access control with Default Deny.

If one is setting up a rule set remotely he/she should not change the default policy to default deny first. It should be the last step. This is because the firewall will block the connection the user is using to change the rule set. I had console access so I changed the default policy right away. I used Ziegler's book as a guide in setting up a script that would apply the rule set on startup.

```
iptables –P OUTPUT DENY
iptables –P INPUT DENY
iptables –P FORWARD DENY
```

The next lines will drop unexpected packets. This will be a packet that the firewall should never see. A scanner or an attacker is trying to see what the firewall will do based on an invalid packet. The way to read these lines is to look at the first argument given to tcp-flags and then assume that the next argument is set from those flags. If the flags in the second argument are set in the first argument do the action, "DROP".

```
iptables –A FORWARD –i eth0 –p tcp –tcp-flags ALL NONE –j DROP
iptables –A FORWARD –i eth0 –p tcp –tcp-flags SYN,FIN SYN,FIN –j DROP
iptables –A FORWARD –i eth0 –p tcp –tcp-flags SYN,RST SYN,RST –j DROP
iptables –A FORWARD –i eth0 -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,FIN FIN -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,PSH PSH  -j DROP
iptables -A FORWARD -i eth0 -p tcp --tcp-flags ACK,URG URG  -j DROP
```

These were my first rules because if it is not a valid packet I do not want to bother looking at the source or the destination service, drop it right away.
The next rules are to drop packets that are known invalid addresses. A bad address would be a spoofed address. I should not see an address coming from my firewall's address. I should not see an address from a private address coming from the Internet interface into the network. On the other hand, I should not see and address from the internal side that is not part of the 192.168.2.0/24 addresses. If there was an address coming from the internal network it could be an attack attempt launched from an internal user. I wanted to guard the Internet from any attacks as best I could so I added a rule on the internal interface.

```
iptables -A FORWARD -i eth0 -s 10.0.0.0/8 -j DROP
iptables -A FORWARD -i eth0 -s 127.0.0.0/8 -j DROP
iptables -A FORWARD -i eth0 -s 172.16.0.0/12 -j DROP
iptables -A FORWARD -i eth0 -s 192.168.0.0/16 -j DROP
iptables -A FORWARD -i eth0 -s 255.255.255.255 -j DROP
iptables -A FORWARD -i eth0 -s 0.0.0.0 -j DROP
iptables -A FORWARD -o eth1 -s ! 192.168.3.0/24 -j DROP
```

The last rule listed prevents addresses from the internal network from being spoofed. Next, I allowed through any established connection.

```
iptables -A FORWARD -i eth0 -o eth1 -m state --state \
ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth0 -o eth2 -m state --state\
ESTABLISHED,RELATED -j ACCEPT
```

I wanted the established related rules early in the rule set because, if a connection is already established it would optimize the connection by not having to run through all of the rules.

Notice that I have not added service rules yet. All of the rules thus for have been source address rules and malformed packets. Although someone could request a connection on a valid port their address might be spoofed. Next I needed to let DNS request out of the network, only to a specific ISP server though.

```
iptables -A FORWARD -i eth1 -o eth0 -p udp -s 192.168.3.45 -d A.B.C.E \
--dport 53 -j ACCEPT
iptables -A FORWARD -i eth2 -o eth0 -p udp -s 192.168.2.3 -d A.B.C.E \
--dport 53 -j ACCEPT
```

Web services have to be allowed out.

```
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 443 -j ACCEPT
```

Email has to be allowed out from the services network.

```
iptables -A FORWARD -i eth2 -o eth0 -p tcp -s 192.168.2.3 --dport 25 -j ACCEPT
iptables -A FORWARD -i eth2 -o eth1 -p tcp -s 192.168.2.3 --d 192.168.3.45 \
--dport 25 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth2 -m state \
--state ESTABLISHED,RELATED -j ACCEPT
```

The previous two rules allow for the mail relay to communicate with the internal mail server. The next set of rules allows for an internal user to connect to a passive ftp server. Please see the limitations section of this paper for comment. I had to allow connections out on ports higher than 1024 to get the connection to work. I needed to know who was trying to ftp out.

```
iptables -A FORWARD -i eth1 -o eth0 -p tcp  --dport 21 -LOG --log-prefix “ftp
access”
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport  21 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 20 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0  -j LOG --log-prefix “internal access”
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 1024:65535 -j ACCEPT
```

The previous five rules was what I created to let ftp out. If I saw an internal access alert in the log, I could check above it for an ftp access alert. This was not the best way to accomplish this task. A better way would have been to use connection tracking. By logging internal attempts at this point I could see if anyone was trying to use a service that was blocked.

I had to add a few rules to the NAT tables in order to allow a connection from the outside to the mail relay machine. There needed to be a rule that would replace the internal private addresses of the network with one public address. I did this through the MASQUERADE target

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0  -d A.B.C.G --dport 25 -j DNAT \
 --to-destination 192.168.2.3:25
```

(A.B.C.G was the public address given to the ISP for the MX record.)

I did not allow any icmp messages. This was not the right thing to do, however I never received any complaints. Source-quenching should be allowed. All of the firewall rules were placed in a script that would execute on startup. Directly from Ziegler there were several kernel support options that were used in the script.

```
echo 1> /proc/sys/net/ipv4/icmp_echo_ignore_broadcast
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
     echo 0 > $f
done
for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
     echo 0 > $f
done
```

The first command ignores broadcast pings. This action is redundant because the firewall’s rule set does not allow echo pings out. If someone was using the network as a launching point of a smurf attack this would prevent it. The second command disables source routing. Source routing is when a packet has been told what routers to use. Tcp_syncookies is a tool used to handle syn flood attacks. The last two commands prevent accepting and sending redirects.

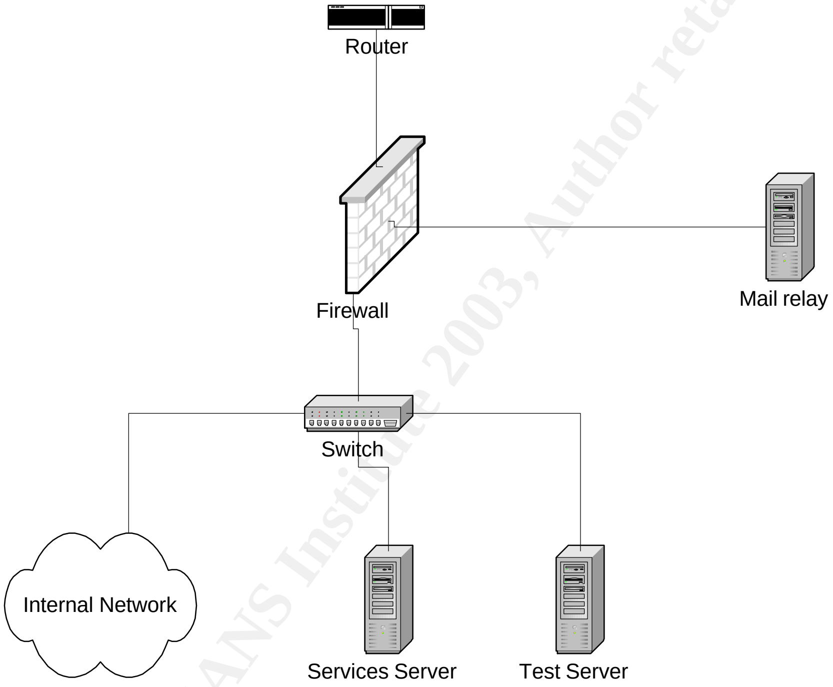
According to SANS Firewalls 101: Perimeter Protection with Firewalls, forwarding should not be allowed by default(6-8). If the firewall would happen to crash this

would allow the network to be accessible without a firewall filter. It would be better to turn on forwarding manually. The next line enables ip forwarding.
echo 1 > /proc/sys/net/ipv4/ip_forward

I added this rule to the startup script. This was not the right thing to do. I discussed this concern with my supervisor. If the firewall crashed he wanted the ability to send a non-computer person into the server room to reboot the firewall. Rebooting and starting up the firewall in his opinion should be a push of a button. I knew it was insecure however business need came before security need.

Mail Server

The mail relay would accept mail and scan it for viruses. It would then deliver to an internal mail server. Employees would then check for mail from the internal mail server. The new network layout would look like this:



The Mail Relay would run Red Hat Linux 7.2. The decision to have a mail relay and its location in the network was based on several factors. The budget of this project did not allow for another firewall so I had to use a single firewall. My supervisor and I came to the conclusion that we needed to have access to email before the end user. This would allow us to scan the email for viruses before the end user could open up the file in the internal network. If management ever had a reason to capture email before an end user received it, the mail relay would be a tap. If our mail was under attack at least the mail relay would provide another layer before the internal network.

Amavis, which stands for A Mail Virus Scanner, is a program that will work with a mail transfer application that will use a virus scanning engine to scan for viruses. Its website is located at www.amavis.org. There were several reasons for using Amavis. Amavis is licensed under the GPL therefore it is free to use. Amavis allows several antivirus engines to execute on the same email. Amavis runs on Linux. I did not find any other email software that would use a normal antivirus scanner to scan email that was free.

Amavis is written in perl. The milter amavis program is the latest stable release (at the time of the project). Earlier versions of Amavis needed the setuid root on /etc/procmail. This posed a security risk because it allows any user to execute procmail as root. The amavis milter did not require the setuid set. One of the benefits of using a milter program is that we can execute any number of milter programs on an email. I later added a mimedefang milter using spamassassin for spam. Mimedefang may be found here http://www.roaringpenguin.com/mimedefang/. This executed completely separate of Amavis in the same mail transfer program.

Sendmail was used as the mail transfer program. After checking with SANS (http://www.sans.org/top20/#U8), I downloaded the source code and then compiled it instead of using the distribution install and rpm. There are advantages of downloading the source is that it allowed me to install certain features I would need. It also allowed me to check the signing key to verify that it was a legitimate copy of Sendmail. In order to use milters, such as Amavis, Sendmail needs to be compiled with the -DMILTER flag set. With the source I could decide what flags I needed set.

After the install of Sendmail there is one file that needs attention, /etc/mail/relay-domains. I placed a single entry of 192.168.3. This allows the relay of internal mail but does not allow outsiders to use the mail relay for spam.

Now was the time to select a virus engine for Amavis to use. Although the cost of the Amavis program is free, a third party vendor is needed for the actual virus search engine. At first I wanted to go with RAV(Reliable AntiVirus). Unfortunately I have used other virus scanners in the past and found that the company did not have good customer service. Amavis contains a series of test after the install that I will cover that will test the ability of the virus engine to scan email. RAV, although supported by Amavis, failed the test. I had to meet a deadline with my supervisor so I reluctantly went with McAffee's uvscan program for Linux. Uvscan passed all of the Amavis tests and comes with a free trial period. The virus engine worked, unfortunately, their customer service was not great.

Installing Amavis

When I installed Linux on the mail relay, just like the firewall, I installed only the packages that I needed. Unlike the firewall, I needed perl on the mail relay for Amavis.

With McAffee there is a simple install script that will install the virus engine.

After Amavis is downloaded and the signature is checked I unpacked it into a separate directory. There are several perl modules that are needed for the install. I did not trust CPAN installing software on my system so I downloaded the source code and installed each module separately. The list of perl modules is located in the Amavis documentation. To install the perl modules I went to the directory of the perl module and executed:

```
make
make test
make install
```

There is a list of programs that Amavis uses that will need to be installed.

I am now ready to install the Amavis program. In the Amavis source directory I ran

```
./configure  --enable-milter  --with-amavisuser=amavis  --with-sendmail-
source=/src/sendmail-a.b.c
```

The configure program will return with error if missing a perl mod. or the virus scanner.

A user named amavis needs to be added to the system
I added a line to the sendmail.mc file in order to get Amavis to work.
(from the amavis documentation)

```
define(`milter’,`1’)
input_mail_filter(`milter-amavis, `S=local:/var/amavis/amavis-milter.sock,
T=S;S:10m;R:10m;E:10m’)
```

One of the big advantages to running Sendmail and Amavis this way is that both are not run by the root user. They are even run by different users. I used the amavis user account to retrieve updates to the virus scanner program.

In order to relay email from the relay machine to the internal server I used the /etc/aliases file. The permissions were set to 644 and the owner and group were root and root. A typical entry looked like this:

```
joe shmoe:      joe@internalserver.widget.net
```

internalserver.widget.net was given an entry in /etc/hosts. This prevented the need for DNS.
Amavis sends alerts by default to a user called virusalerts. I added an alias for this.

```
virusalerts:   dan@internalserver.widget.net
```

This would forward virus notifications to the internal server.
Amavis will stop an email if it contains a virus. The virus will be placed in a special directory on the mail relay. Since the mail is not forwarded on the virus does not get to the internal network. This was an increase in security because before we had to rely on the end user to update virus patterns, and avoid opening suspect email. Now we controlled at least one scan on the email and could prevent sending a virus message to another business.

Tripwire

Tripwire was used on both the firewall and the mail relay. Since I was new I did not have much time to configure tripwire extensively. What I did do was a recommendation from Linux Journal (http://www.linuxjournal.com/article.php?sid=4718). I added a entry to the default configuration file that I knew would cause a false positive. This way I could tell if a report was fake or not.
In order to do this, I set a watch on the /var/log/messages file on both the firewall and mail relay.

```
 # Dan’s log file
{
  rulename=”Danlogfile”,
  severity=100
}
{
/var/log/messages -> $(ReadOnly);
}
```

I should see this tripped every time tripwire ran. The default policy file had references to several files that my systems did not have. They were not installed. The rules concerning these files caused tripwire to print out file system errors. I considered this a security feature because I knew how many file system errors there were. Tripwire ran once a night on both the mail relay and the firewall.

After

From outside the network I ran nmap against the firewall to see if anything was open.

```
nmap -sS A.B.C.D
nmap -sT A.B.C.D
nmap -sU A.B.C.D
```

The same scans of the address assigned to the email relay revealed the same as the firewall except for the stealth scan. I was able to get the up time of the mail relay machine and port 25 was open.

Nmap returned stating that it could not find a host. I tried to ping the firewall, the ping program simply waited until the ping program timed out. I tried to telnet to the firewall, no response. The telnet program hung. I checked the logs and could see the attempt.

After implementing the firewall I did not have anything I could go to management with as saying "we are more secure". This is simply because they did not notice it. They had the same network access as before the firewall. The firewall had been installed on the same machine that had the old firewall. There was nothing I could physically show them for my time and effort. The check from nmap reveled that nothing seemed to be getting through (Nmap results did not mean much to management). Not even a ping. I knew that it was better than before. After the mail relay went into production and started catching viruses, I had something solid I could show management. The way management seemed to view the "viruses caught" report was that by preventing a virus from attacking the network they have saved time and money by not having to have me clean their system. One of the company's vendors called warning not to open any email from that vendor. The vendor was attacked with a virus. Management noted that our company did not have to make a call to anyone avoiding embarrassment. My work was justified. I was asked what guarantee I could give the company based on the work that I have accomplished. My response was that I could not guarantee that we would never see a virus. I could guarantee that no matter what the end user did or did not do their email would be scanned for viruses with an update virus signature file. I could also guarantee that if a virus was caught in email it would not propagate throughout the network. It would be stopped by a machine "separated" from the internal network.

Limitations

I am new to the security field. This project was my first security project. I know that there were mistakes along the way.

"In fact, some studies state that as much as 70 percent of all attacks come from someone within ..."(Brenton, 6). First, this company is not a big company its threat is more of an internal threat than an external. According to SANS there should be a firewall between the internal network and the fileserver(3-39). This would allow the file server to be behind two firewalls. The second firewall should not be Linux based. Second, the mail relay ran the same software as the firewall and the internal mail server. It ran the same mail transfer program, Sendmail. If I could do this again the mail relay would run a completely different operating system as well as a different mail program. As it is explained, this would be a form of hardware stacking. If a hacker could compromise one system he/she could use the same vulnerability to get to the second system running the same program. This project can also give one a false sense of security. Although we could stop inbound and outbound email viruses, we could still get a virus. Someone could get a virus from a web site or bring one in from a floppy or c.d. We later implemented an antivirus program for each user to help prevent this from happening. Given the limitations of the budget and my experience this project was a success.

References:

Brenton, Chris; Hunt, Cameron. Active Defense A Comprehensive Guide to Network Security. Alameda: Sybex, Inc, 2001.

Northcutt S. Firewalls 101: Perimeter Protection with Firewalls, 2001.

Ziegler, Robert L. Linux Firewalls. Indianapolis: New Ryders, 2002.

"SANS/FBI The Twenty Most Critical Internet Security Vulnerabilities"
http://www.sans.org/top20/ 9 February 2003.

"Compiling Sendmail" 28 December 2002.
http://www.sendmail.org/compiling.html 10 February 2003.

"Linux download: software for your box - uvscan"
http://www.linuxdownloads.org/article.php?sid=1151 10 February 2003.

"Paranoid Penguin: Intrusion Detection for the Masses"
http://www.linuxjournal.com/article.php?sid=4718 . 10 February 2003

Renier, Link. "AMaVIS - A Mail Virus Scanner" 31 October 2000.
http://www.amavis.org/amavis.html#require 9 February 2003.

Renier. Link. "AMaVIS - Download" 10 February 2003.
http://www.amavis.org/download.php3 . 10 February 2003.

"Roaring Penguin Software (MIMEDefang)". 10 February 2003
http://www.roaringpenguin.com/mimedefang/. 10 February 2003

"RAV AntiVirus - Business/Corporate Anti-Virus Solutions"
http://www.ravantivirus.com/pages/business.php 10 February 2003.

"Sendmail - 8.12"
http://www.sendmail.org/8.12.7.html . 10 February 2003.