Protecting Your Network with OpenBSD and PF

David Arnett

April 15, 2004

SANS\GIAC Practical – GSEC Certification
Version 1.4c

Abstract

The purpose of this paper is to identify a common network security deficiency, evaluate alternatives and implement a solution.

Initially this paper will describe a security deficiency in a small office network. It will then focus on various options in dealing with the deficiency. This paper will then describe in detail the implementation of the chosen solution, specifically the installation and configuration of an OpenBSD, PF (packet filter) firewall. Finally, this paper will measure the impacts of the firewall within a small office network.

Introduction

Many Windows users are unfamiliar or are intimidated with the seemingly hundreds of open source Unix like operating systems available today but I intend to show that any average computer user can successfully install and configure OpenBSD and PF (packet filter) as a network based firewall. Although the use of a firewall should not be your only security measure it is an important part of an overall defense in depth strategy.

> "Defense in depth is the proposition that multiple layers of security are better than a single protection mechanism. The layers may be technological, procedural, or policy." (Defense 1)

In this paper I will first describe a real world business scenario with little or no information security measures. Next, I will discuss alternatives to the problem including defense in depth strategies. Finally, I will walk through the implementation of the security measures detailing the installation, configuration and testing of a OpenBSD and PF firewall.

Problem

The scenario for this installation is a small business with five (5) personal computers. The owner of the business advised that their current systems were approximately four (4) to five (5) years old and that they were now experiencing system crashes and poor performance. He stated that he was concerned about loss or damage to his accounting records and customer databases and wanted to upgrade his systems mainly due to the unreliability of the current systems.

A quick review showed that the five (5) systems were Pentium II 400mhz machines with 32 megabytes of RAM and 8.4 gigabyte hard drives. Three (3) of the systems were running Microsoft Windows 98 and two (2) had been upgraded to Windows ME. None of the systems had been patched since installation and there were no personal firewalls installed. Only two (2) of the systems had any sort of antivirus software installed, which had not been updated since installation.

Additionally only two (2) of the systems required passwords; and the main "Accounting" computer had an open file share with no authentication required.

The office computers are generally used for basic office applications and email received through a POP (post office protocol) account. The office does not host a web site or its own email server and has no need to accept any connections originating from the internet. The office connects to the internet through a DSL (digital subscriber line) from a local service provider and a Cisco 675 router. The router provides NAT (network address translation) services to the 10.0.0.0/8 network and is connected to a unmanaged 10 port hub that allows connectivity and file sharing to the office computers.

Alternatives

The first alternative included upgrading the existing systems with additional memory; new hard drives (for reliability not size), Microsoft XP Pro, as well as, antivirus and personal firewall software. It was additionally recommended that a network based firewall be installed and a tape backup system be implemented to preserve pertinent business records.

The second recommendation included replacing the existing hardware with entirely new systems. This option was presented because the original hardware vendor would not guarantee compatibility of the original hardware and Windows XP. Additionally, this solution provided a three (3) year in house warranty on all the hardware.

Decision

It was decided that the old systems would be replaced with new Pentium 4 systems running Windows XP Pro. The systems were installed with new antivirus and personal (host-based) firewall software. Due to the fact that the office had no dedicated IT staff, the systems were configured for automatic Windows and antivirus updates. Additionally a new switch replaced the older hub and a tape backup unit was installed on the main "Accounting" computer. The tape was configured to automatically backup important business records and enough tapes were purchase so that the owner could maintain a copy offsite. A strict password policy was enforced requiring strong passwords and periodic password changes. Windows XP Pro was configured to provide authentication to the shared resources drive.

The theory of Defense in Depth is that you put into place as many layers of security measures as are feasible under the existing conditions. If one or more layers fail or is not designed to handle the threat, one of the remaining layers will protect your systems, network and information. In this scenario I have put into place multiple layers from a network and host based firewalls to patch management, authentication and disaster recovery. The key is that the measures were feasible and appropriate for the existing work environment. The addition of an OpenBSD Firewall was a cost effective way to add another layer of security.

Many options are available for installing a network based firewall. They include hardware firewalls, Linux IPSockets/IPTables, and other BSD-based operating systems running PF (packet filter). These are all reasonable and effective choices when implemented properly. For the purpose of this scenario, I chose the OpenBSD operating system and PF.

> “The OpenBSD project produces a freely available, multi-platform 4.4BSD-based UNIX-like operating system. Our goals place emphasis on correctness, security, standardization, and portability. (Obsd 1)

OpenBSD requires very little disk space (less then 300 megabytes), runs on multiple hardware platforms and, by default, has many of its available services disabled. This is important because a common technique for hackers is to exploit a vulnerable service running on a system.

> “Packet Filter (from here on referred to as PF) is OpenBSD's system for filtering TCP/IP traffic and doing Network Address Translation. PF is also capable of normalizing and conditioning TCP/IP traffic and providing bandwidth control and packet prioritization, and can be used to create powerful and flexible firewalls”. (6 - Networking par 6.3)

Due to the recent computer upgrades at this business I chose to use one of the original computers for the firewall system. This allowed me to provide an additional layer of security without any additional hardware cost. The original computers are Pentium II 400Mhz with 32mb of RAM, a 8Gb hard disk drive with one (1) 10/100 ethernet card. I added an additional 32mb of RAM and installed a second 10/100 ethernet card from one of the original systems. The dual Ethernet cards are required to separate the internal network from the internet.

One of the basic principals of Network Security is the idea of defense in depth. Each of the above-listed precautions provides a layer of security. The addition of a network based firewall will add another layer of security to the network. Additionally a network based firewall can provide flexibility in its configuration and the ability to log activity matching the firewall rules set.

Installation

The “Installation” portion of this paper was designed to walk the reader through the installation and configuration of an OpenBSD PF firewall while showing its simplicity, efficiency and effectiveness.

The Installation of OpenBSD is fairly straight forward and can be accomplished in many ways. The basic idea is that the system is booted from a bootable floppy disk, hard disk or from a network boot if your hardware supports it. After the boot process completes the install scripts walk the user through the setting up of the disks, hostname, root password, network, and selection of installation files. The OpenBSD installation files consist of multiple .tgz (tar gzipped) compress files. The installation scripts can retrieve these files from an OpenBSD ftp site, local disk, local cdrom, or network drive.

The current release for OpenBSD at the time of this paper was version 3.5. I accessed one of the OpenBSD “mirror” sites listed from a link at www.OpenBSD.org. From the mirror site, I entered the subdirectory labeled “3.5” for the current release and I downloaded both the “tools” and “I386” subdirectories to my local machine. The “tools” subdirectory contains several utilities including a program that writes the boot floppy image to a floppy diskette as well as a utility for disk partitioning. The “I386” subdirectory contains the installation files for OpenBSD. In reality, the entire “I386” subdirectory is not required for this installation. For this installation I chose not to run X Windows on the server so the installation files for X Windows were not needed. So if you have a slow internet connection you only need to download the boot image you are going to use and the “base35.tgz, comp35.tgz, etc35.tgz, misc35.tgz, bsd and man35.tgz” files.

The next step is to create the boot floppy. From a Unix-like computer, use the “dd” command and write the file to the disk using the following command:
dd if=floppy35.fs of=/dev/rfd0c (substituting your floppy drive for rfd0c).
From a DOS environment use the rawrite.exe command located in the \tools subdirectory that was downloaded. If true a DOS or Unix-like operating system is not available, use the Windows version of “rawrite.exe”, “rawritewin.exe” available from http://uranus.it.swin.edu.au/~jn/linux/rawwrite.htm. Using the Windows version, I created a bootable floppy diskette by pointing the “rawritewin.exe” to the “floppy35.fs” boot floppy image located in the “\i386” subdirectory. If the system would boot from a CD-ROM then I could used the “\i386\cd35.iso” image and written it to a CD-ROM. Using either method will result in the same outcome.

Next insert the bootable floppy or CD-ROM and boot the system. While the system is booting a large volume of text will scroll across the screen. This is the boot kernel locating hardware devices on the system and is completely normal. Once the system has booted, select from one of the following options: (I)nstall, (U)pgrade or (S)hell. Place the CD-ROM containing the ”\i386” subdirectory tree into the CD-ROM drive and enter “I” for install.

The next two prompts request a “Terminal type?” and keyboard encoding. The default [vt220] and keyboard encoding should be sufficient unless there are specific needs. Press enter for prompts both.

The following screen warns of data loss and asks if the user really wants to modify the hard drive and proceed with the install. In my situation, I was installing to a single empty hard disk, so the answer was “yes”. If your situation is different, this may require additional consideration. The install script will locate available disks on the system and asks which one should be designated as the “root” disk, or where the “/” root file system will be installed. In my situation I had only one disk installed so the default [wd0] (Western Digital drive 0) was entered. The next prompt asks if all of disk “wd0” should be used for OpenBSD. Again I entered “yes” and continued.

Included with the hard drive selection an advisory stating that “To enable all available security features you should configure the disk(s) to allow the creation of separate file systems for /, /tmp, /var, /usr and /home. This means that you should create separate partitions and therefore separate files systems on the disk or disks for each of these directories. This will keep the \var directory from overfilling with logs and filling up your system drive”. For additional information read the “installation” and “readme” files located on the www.OpenBSD.org site as well as those located in the “\3.5” and “\3.5\i386” subdirectories.

According to the OpenBSD faq page:

> “If you say "yes" to this question, the entire disk will be allocated to OpenBSD. This will result in a standard Master Boot Record and partition table being written out to disk -- one partition, the size of the entire hard disk, set to the OpenBSD partition type, and flagged as the bootable partition. This will be a common choice for most production uses of OpenBSD; however, there are some systems this should not be done on. Many Compaq systems, many laptops, some Dell and other systems use a "maintenance" or "Suspend to Disk" partition, which should be kept intact. If your system has any other partitions of any type you do not wish to erase, do not select "yes" to the above question”. (4-OpenBSD 3.5. par. 4.5.2)

At the prompt, I entered “yes”, thus the entire disk was allocated to OpenBSD. Next I partitioned the disk for the files systems. For my application, I selected the following partition scheme:

Wd0a	200M	/(root)
Wd0b	300M	(swap)
Wd0d	200M	/tmp,
Wd0e	1G	/usr,
Wd0f	300M	/home
Wd0g	6G	/var

I intentionally made the /var partition rather large to allow adequate storage for the firewall log files. The /user partition was made large enough to add applications later, if needed. The /home partition size was selected with the assumption that I will have no other users on the system.

Next, the startup script activates the disklabel program. At the “>” prompt, the user should enter one of the following commands:

?	A question mark for command help
p	Prints the current disklabel to the screen. Modifiers k, m and g display kilobytes, megabytes or gigabytes
d	Clears any existing disklabel and creates a new one
m	Modifies an existing entry

At the ”>” prompt, I chose “a”. The next prompt will requested the beginning offset of the partition and displayed the default sector offset:

```
Offset: [63] _ <enter> for default
```

The following prompt asked for the partition size and displayed available sectors. Input will create a 200 megabyte “/” root partition.

```
Size: [16406187] _ <200m><enter>

(Creates a 200 megabyte partition.)

FS type: [4.2BSD] <enter>

Mount point: [none] _ “ /” <enter>
```

The remainder of the prompts and input was as follows:

```
>
> a b
offset: [409248] <enter>
size: [1503787] _ “300m” <enter>
FS type: [swap] _ <enter>
>

>a d
offset: [1024128] _ <enter>
size: [1350187] _ “200m” <enter>
FS type: [4.2BSD] _ <enter>
Mount point: [none] _ “/tmp”<enter>
>
```

```
>a e
offset: [1433376] _ <enter>
size: [1247787] _ “1g” <enter>
FS type: [4.2BSD] _ <enter>
Mount point: [none] _ “/usr”<enter>
>

> a f
offset: [3531024] _ <enter>
size: [735787] _ “300m” <enter>
FS type: [4.2BSD] _ <enter>
Mount point: [none] _ “/home” <enter>
>

>a g
offset: [4145904] _ <enter>
size: [582187] _ <enter> (for rest of disk space)
FS type: [4.2BSD] _ <enter>
Mount point: [none] _ “/var” <enter>

> w     <enter>
>q      <enter> (to write the disklabel and quit)
```

The next step was to verify the mount points. This prompt listed which partition the listed directories will be mounted too. This should be the same as the above listed “partition scheme”. I typed “done” <enter>, and the install script formatted the partitions. Another warning message appeared indicating that all existing data would be erased. I typed “yes” <enter>, to continue.

The next prompt requested a system hostname. I chose the name “testfw”, and pressed <enter>. The next prompt asked if the user wished to configure the network. For my situation, I selected “no” because I planned on configuring the network at a later time. The next prompt asked for a root password and a confirmation of the password. I entered a password and pressed <enter> to continue.

The next step was to load the install sets. I entered “c” for CD-ROM “cd0” and entered the path “/i386”, because that was the directory that I had burned to the CD-ROM disk. Next, a list of the available install sets was displayed. Since I did not wish to install X Windows, I only selected bsd, base35.tgz, etc35.tgz, misc35.tgz, comp35.tgz and man35.tgz. Any set can be selected by entering the complete name or deselected from the checked sets by placing a “-“ in front of the name. After pressing <enter> twice to choose and confirm my selection, the selected sets were installed. This process took only a few minutes and the progress was shown via text progress bars. After the selected sets were installed, I was prompted again for additional sets. I pressed <enter>, since no additional sets were required

Next, the script asked if “sshd” (secure shell daemon) should start automatically. Because I wished to access the firewall remotely, I did opt for “sshd” to start by default. The next prompt asked if the X Windows System was expected to run. I entered “no”. Next the script requested the local time zone. By typing a question mark, “?”, available time zones are displayed. I select the appropriate time zone and pressed enter.

This completed the installation. I entered “halt” on the command line, removed the boot floppy and then reset the computer to reboot into the newly installed operating system. The system rebooted, generated the ssh keys and presented a login prompt. I logged in as “root” with the password and pressed enter for terminal type. The prompt “hostname#” was displayed.

Note: Do read the afterboot(8) man page by typing “man afterboot” at the prompt and do create a standard user account for future logins.

Configuration

Configuration of a newly installed OpenBSD system as a firewall is actually easier that it might seem. For me the first order of business was to disable all unnecessary services and patch the operating system. “The first step to lock down the firewall box is to disable all unnecessary running services. Luckily, OpenBSD out of the box is really secure even with ident, comsat, daytime, time, rstatd, and rusersd enabled in /etc/inetd.conf. Comment out mentioned services in /etc/inetd.conf and edit /etc/rc.conf and make sure portmap, sendmail and ntpd daemons are disabled as well” (Tran 1).

Next, I connected one Ethernet connection from the computer to the Cisco 675 DSL router and the other to the internal network. The two networks were then bridged internally and the data was filtered using PF (packet filter). The fun part was that the adapters did not require an assigned IP address as they were both brought up in promiscuous mode. Both adapters quietly listen to all traffic on the cable. The bridge moved the data to the other adapter and therefore passed all traffic through. I ran PF (packet filter) to block and pass data according to my established rules. Since traffic was passed through both adapters it was only necessary to filter traffic on one interface, the one that pointed to the internet.

Next, I identified the network interface cards. In OpenBSD the cards are named by their manufacturer. In my situation, I installed two Intel 10/100 ethernet adapters which OpenBSD named “fxp0” and “fxp1”. To find this information I ran the “dmesg” command followed by the “|more”

```
testfw# dmesg |more
```

Displayed were the hardware devices that the system had located. It appeared in lines similar to these:

```
fxp0 at pci0 dev 8 function 0 "Intel 82557" rev –x08: irq 12, address
00:02:b3:07:xx.xx

fxp1 at pci0 dev 9 function 0 "Intel 82557" rev –x08: irq 10, address
00:02:b3:07:xx.xx
```

With the network adapters identified, it was necessary to bring them up and make them active at boot time. This was done by creating a file in the "/etc" subdirectory named "/etc/hostname.fxp0" and "/etc/hostname.fxp1" where "fxp0" was the name of my network adapter. There were many options available for these network configuration files, but all that was needed for my configuration was to add the word "up" in each file so that the interfaces will be brought up at boot time. For example:

```
echo up > /etc/hostname.fxpo
echo up > /etc/hostname.fxp1
```

With the network adapters activated, the next step was to create a bridge between the two (2) network adapters. This was also done by a simple configuration file in the "/etc" directory named "/etc/bridgename.bridge0". This file contained the command "add fxp0 add fxp1 up blocknonip fxp0". This command told the system to add these two interfaces to the bridge, bring it up and block all non ip traffic on interface "fxp0", our external interface. For example:

```
Echo add fxp0 add fxp1 up blocknonip fxp0 > /etc/bridgename.bridge0
```

Next, I enabled IP forwarding so that the traffic would be passed between the two networks. This was done by editing a line in "/etc/sysctl.conf" and removing the comment symbol "#" from the line "net.inet.ip.forwarding=1".

At this point, the network adapters had been identified and configured to come up at boot time. A bridge has been created between the two adapters, and ip forwarding has been enabled so that packets would be passed between the networks. The next step was to enable PF at boot time and write the rules for the pf.conf file (the firewall rules).

In my scenario, the Cisco 675 router provided network address translation from its external IP to the internal 10.0.0.0/8 network. The Cisco's IP was 10.0.0.1 and each of the workstations were set up with static IP numbers in the 10.0.0.0/8 range. The remaining steps for the configuration included: rebooting the system, verifying that the network was working properly, reviewing the “/etc/pf.conf” file and writing the rules set for the firewall.

I connected interface fxp0 to the Ethernet “LAN” port on the Cisco 675 router, which connected to the internet. Next, I connected interface fxp1 to the internal network switch and booted the new firewall system.

After bootup, I executed “ifconfig –a” and observed that the two interfaces were up and in promiscuous mode. I then executed “brconfig –a” (bridge configuration program) to verify that interface fxp0 and fxp1 were functioning properly. Finally, I verified that each workstation had internet connectivity. At this time, PF was operational. However, no rules had been defined so it was just passing all traffic.

PF itself is not an application but a pseudo-device and all filtering takes place in the kernel. PF is controlled by a simple configuration file \etc\pf.conf.

To enable PF at boot, I edited the “/etc/rc.conf” file to read “pf=YES” from “pf=NO”,

The included “/etc/pf.conf” provides some basic examples of defining macros, NAT (network address translation), tables, redirection or traffic forwarding and basic rules. For reference purposes the “/etc/pf.conf” file from the OpenBSD installation has been included below:

```
#	$OpenBSD: pf.conf,v 1.27 2004/03/02 20:13:55 cedric Exp $
#
# See Pf.conf(5) and /usr/share/PF for syntax and examples.

#ext_if="ext0"
#int_if="int0"

#table <spamd> persist
#table <spamd-white> persist

#scrub in

#nat on $ext_if from !($ext_if) -> ($ext_if:0)
#rdr pass on $int_if proto tcp to port ftp -> 127.0.0.1 port 8021
#rdr pass on $ext_if proto tcp from <spamd> to port smtp \
#	-> 127.0.0.1 port spamd
#rdr pass on $ext_if proto tcp from !<spamd-white> to port smtp \
#	-> 127.0.0.1 port spamd

#block in
#pass out keep state
```

```
#pass quick on { lo $int_if }
#antispoof quick for { lo $int_if }

#pass in on $ext_if proto tcp to ($ext_if) port ssh keep state
#pass in on $ext_if proto tcp to ($ext_if) port > 49151 user proxy keep
state
#pass in log on $ext_if proto tcp to ($ext_if) port smtp keep state
#pass out log on $ext_if proto tcp from ($ext_if) to port smtp keep state
```

Note: The basic syntax for PF is very simple. PF will read down the rules until it finds a match for the packet that it has received. It will continue to read the rest of the rules and will "act" on the last rule that it has read that matches the condition. Example: If the rules state: "block all" then "pass all" PF will pass the traffic based on the last entry that matched the condition. To stop this behavior the "quick" term is used. When PF reads the quick on a packet that matched that rule, then PF will quit looking for additional rules to match. The "all" statement states to act on all interfaces, or on all IP addresses or ports.

Below is a brief explanation of the PF syntax:

"*action direction* [log] [quick] on *interface* [*af*] [proto *protocol*] \
from *src_addr* [port *src_port*] to *dst_addr* [port *dst_port*] \
[*tcp_flags*] [*state*]" (PF: Packet Filtering 1)

"*action direction*":

Block in	block traffic entering the interface
Block out	block traffic exiting the interface
Pass in	pass traffic entering the interface
Pass out	pass traffic exiting the interface
Block all	block all traffic entering or exiting the interface
Pass all	pass all traffic entering or exiting the interface

"log":

create entry in log for packet matching rule

"quick"

if rule matches condition, stop searching for additional matches

"*interface*"

which interface to act on (fxp0)

"*protocol*"

which protocol to match (tcp udp icmp)

"from *src_addr* [port *src_port*]"

match traffic coming from IP address and source port

"to *dst_addr* [port *dst_port*]"
match traffic going to IP address and destination port

"tcp_flags"
TCP flags to match

"state"
whether to "keep state" on the connection

In creating the rules set, I first defined the external (ext_if=) and internal interfaces (int_if=) in the Pf.conf file. The external interface was changed to "fxp0" and internal interface to "fxp1". Since the "ext_if" and "int_if" are examples of macros that PF uses, declaring the external interface (ext_if) as "fxp0" and the internal interface (int_if) as "fxp1" allowed the rest of the rules to call on the variable "ext_if" and "int_if". This makes the writing and later administration of the rules easier then rewriting the actual interface. This also helps if an interface card breaks and must be replaced with a different card. There is no need to rewrite the entire rules set to replace each instance of "fxp0", instead just change the variable to reflect the new interface card.

Next, I uncommented the "#scrub in" line and changed it to read "scrub in all random-id fragment reassemble". The alterations to the "#scrub in" line allowed the firewall to perform packet normalization, reassembly to all interfaces and randomize the IP sequence numbers.

> "'Scrubbing' is the normalization of packets so there are no ambiguities in interpretation by the ultimate destination of the packet. The scrub directive also reassembles fragmented packets, protecting some operating systems from some forms of attack, and drops TCP packets that have invalid flag combinations." (PF: Scrub 1)

I purposely left the nat, redirection (rdr) and spamd sections commented out (#) because they were not needed in my application. But they are definitely worth the research and implementation if they fit your needs.

Next, I defined a "block all" policy as the default for all traffic if no other rules apply. Below is an example from my "/etc/pf.conf":

```
Block all
```

Next, I chose to allow all traffic into and out of my internal interface. This was done because I only needed to filter traffic on one interface; the other was just a bridge to the internal network. The "quick" statement told PF that if the rule applies stop looking for any further rules to match. In other words, placing a "block all quick" statement PF will block all traffic on all interfaces and stop looking for other rules. Below is an example from my "/etc/pf.conf":

```
pass in quick on fxp1 all
pass out quick on fxp1 all
```

The next block line instructed PF to block and log all incoming traffic to the ext_if. this line stated that PF should not allow any incoming traffic from the internet and to log any packet that tries to enter. Next, I instructed PF to "pass out" or allow to exit all traffic from the internal network to the internet. I also wanted to log this traffic and keep state. Keeping state tells PF to keep track of the connection so that it will allow in the response from a connection on the internal network.

```
block in log on $ext_if all
pass out log on $ext_if from $internal_net to any keep state
pass out log on $ext_if proto {udp, icmp } from any to any keep state
```

Although UPD is a connectionless protocol, the next rule told PF to remember that the internal network sent out a UDP packet and allows a return.

The next rule simply blocked all traffic originating from the internal network that is not "!" from the 10.0.0.0/8 network. This was to prevent any packets with a source IP number not from my internal network to exit the firewall.

```
Block out log quick on $ext_if from ! 10.0.0.0/8 to any
```

The last three block lines blocked traffic from the internal network to some commonly exploited ports in the event that one of my machines is infected and attempts to locate additional victims.

```
Block out log quick on $ext_if proto tcp from any to port { 25 135 136 137
138 139 445 593 665 >< 766 4444 }
Block out log quick on $ext_if proto udp from any to port { 25 69 135 139
445 8998 }
Block out log quick on $ext_if proto { tcp udp } from any port { 54320
31337 27573 27574 } to any
```

My completed /etc/pf.conf files for the firewall read as follows:

```
ext_if=fxp0
int_if=fxp1
internal_net=”10.0.0.1/8”
Block all
pass in quick on fxp1 all
pass out quick on fxp1 all

block in log on $ext_if all
pass out log on $ext_if from $internal_net to any keep state
pass out log on $ext_if proto {udp, icmp } from any to any keep state
Block out log quick on $ext_if from ! 10.0.0.0/8 to any
Block out log quick on $ext_if proto tcp from any to port { 25 135 136 137
138 139 445 593 665 >< 766 4444 }
Block out log quick on $ext_if proto udp from any to port { 25 69 135 139
445 8998 }
Block out log quick on $ext_if proto { tcp udp } from any port { 54320
31337 27573 27574 } to any
```

With the rules set in place, I restarted the system to see how it all worked.

Logging

Parsing through log files can certainly be time consuming and is often the most overlooked and dreaded of all security functions. However, the viewing of log files is essential so that the examiner is familiar with the logging format and has a understanding of what normal logging traffic looks like in case of an intrusion. OpenBSD PF provides a very strong logging format that should not be overlooked.

Logging in PF is done by the PFlog daemon “PFlogd” which writes packets to the “/var/log/PFlog” file. The PFlog daemon listens on the “PFlog0” interface and writes the packets in “tcpdump” binary format. This means that the logs cannot be read with a text editor. However, tcpdump or any other utility that recognizes the tcpdump format can be used to decipher the output. This is a very powerful feature and allows the examiner greater flexibility in parsing the log files. To view the log file enter the command:

```
# tcpdump -n -e -ttt -r /var/log/PFlog
```

To view the log in real time you can specify the log interface in your tcpdump command line instead of the log file:

```
# tcpdump -n -e -ttt -i PFlog0”
```

Here the –r for "read file" is replaced by the –i switch for interface.
(PF: Logging 1)

Measure Impacts

In order to measure the impact of the firewall, I asked three questions. First, is the firewall blocking traffic that it is suppose to block? Second, is the firewall allowing authorized traffic out? And lastly, does the firewall cause significant slowing of internet traffic?

Note: Scanning a network for vulnerabilities and testing network performance are two (2) complete subjects worthy of their own papers. I am simply documenting basic steps taken to ensure the functionality of my firewall, not a complete guide for doing either. Additionally, it is important to obtain prior permission, and have the authority to do scanning of any network.

To measure the effectiveness of the firewall I first connected a small network hub between the firewall and the internet gateway (Cisco 675 router). I then connected my computer to the small hub, giving my test machine a open 10.0.0.0/8 static IP and verified that I could "ping" the router interface (10.0.0.1) from my "test" computer. Once this was accomplished I disabled "PF" on the firewall by issuing the command "pfctl –d", which stopped the firewall filtering, essentially making it a pass through to the internal network. I then was able to "ping" one of the internal systems from my "test" computer and likewise "ping" my "test" system from an internal system. I also verified that I had normal internet connectivity from the office systems.

After completing this step I enabled the firewall packet filter "PF" by issuing the command "PFctl –e". This caused PF to reread the rules set from the "/etc/Pf.conf" file. Next, I verified that I still had internet connectivity from the office systems and that I could still "ping" my "test" computer.

Note: Remember that the rules above allow outgoing icmp "ping" packets but do not allow them to enter.

I next issued the command "tcpdump –n –e –ttt –I PFlog0" on the firewall machine so that I could review the log entries in real time.

To test the firewall I tried to "ping" the internal office systems from my "test" system. This failed showing that the firewall was at least partially working. A review of the display showed that the "ping" packets had matched a rule and had been successfully logged as a "match block in on fxp0".

From my "test" computer, I then ran "Nessus" scanner (available at www.nessus.org) and various "nmap" (available at www.insecure.org") port scans in an attempt to identify any of the internal office systems. The net result was that I was only able to see my test machine and the gateway router.

In order to test the efficiency of the firewall, I executed three separate tests. The test was to time a download from a site under three (3) conditions: (1) With a system connected directly to the router bypassing the firewall system altogether, (2) With the firewall in place but with PF disabled, (3) With PF enabled. The net result was that there was no significant difference in the download speed between the three setups.

In my final test, I monitored the firewall logs and solicited user feedback over a period of time to ensure that the end users were able to conduct business without interference from the firewall.

Conclusion

In this paper, I have presented a case involving a real world business. The business, typical of many small businesses had a limited budget and no technical staff. The computer systems in this business, like many other businesses, have been added and updated on an as-needed basis with little or no planning for information security. I have discussed and implemented principles of defense in depth by applying multiple layers of security. This included upgrading the computer systems, installing patches, installing antivirus software, installing host based firewalls, establishing user and resource authentication, creating backup procedures, and the installing of a network based firewall.

I expanded the discussion on network based firewalls by examining in detail the process of installing, configuring and testing an OpenBSD and PF network based firewall. Through this process I have shown the reader that using OpenBSD and PF as a network based firewall is cost effective, efficient, secure, flexible and offers ease of installation and ease of configuration. Finally through testing, I was able to determine that the new firewall was a viable solution for the original scenario.

List of References

"4 – OpenBSD 3.5 Installation Guide". Sept 9, 2004, Chap.4 , par. 4.5.3
4.5.3 – Setting up disks
http://www.openbsd.org/faq/faq4.html)

"6 - Networking". Sept 9, 2004, chap. 6 par. 6.3
OpenBSD Faq, How do I filter and firewall with OpenBSD?
http://www.openbsd.org/faq/faq6.html#PF

"Defense-in-depth". Sept 14, 2004, pg.1)
Free-definition
http://www.free-definition.com/Defense-in-depth.html

"Hardware" Sept 9, 2004
OpenBSD Documentation
ftp://ftp3.usa.openbsd.org/pub/OpenBSD/3.5/HARDWARE

"obsd-faq". Sept 9, 2004 pg. 1
OpenBSD Documentation and Frequently Asked Questions
ftp://ftp3.usa.openbsd.org/pub/OpenBSD/doc/obsd-faq.txt

"PF: Example: Firewall for Home or Small Office". Pgs 1-6
www.openbsd.org/faq/PF/example1.html

"PF-faq". Sept 9, 2004
PF: The OpenBSD Packet Filter
ftp://ftp3.usa.openbsd.org/pub/OpenBSD/doc/PF-faq.txt

"PF: Getting Started". Pgs 1-2
www.openbsd.org/faq/PF/config.html

"PF: Logging". Sept 14, 2004, pg.1
http://www.openbsd.org/faq/PF/logging.html

"PF: Packet Filtering". Sept 9, 2004, pg. 1
OpenBSD Faq,
http://www.openbsd.org/faq/PF/filter.html

"PF: Scrub (Packet Normalization)". Sept 9, 2004 pg. 1
OpenBSD Documentation and Frequently Asked Questions
http://www.openbsd.org/faq/PF/scrub.html

"PF: Shortcuts for Creating Rulesets". Sept 9, 2004. pg.1
OpenBSD Documentation and Frequently Asked Questions
http://www.openbsd.org/faq/PF/shortcuts.html

"RawWrite for windows version 0.7 for NT & 95". Sept 14, 2004
written by John Newbigin
http://uranus.it.swin.edu.au/~jn/linux/rawwrite.htm

"README". Sept 9, 2004
OpenBSD Documentation
ftp://ftp3.usa.openbsd.org/pub/OpenBSD/README

Tran, Hoang Q. "OPENBSD PF". Sept 13, 2004. pg. 1
www.unixcircle.com/features/openPFnat.php