Global Information Assurance Certification Paper

Penguin Perimeter Protection
Chris Krumme
October 1, 2000

Introduction

With more and more people connecting to the Internet on a daily basis for both business and pleasure, it doesn't take much to draw the conclusion that our systems are exposed to a constantly increasing number of threats. As the number of compromised systems grows and lessons are learned the hard way, it will become apparent that protecting your systems from the dangers of the wild should become commonplace – if not standard. With the increasing popularity and stability of Linux, corporate, SOHO, and home network administrators have a viable and economical solution when it comes to protecting their networks from the dangers of cyberspace. Although your systems will never be 100% secure, between Linux and that dated PC collecting dust in the corner you can set up a powerful guard at the front door to your network.

There are plenty of detailed reference materials available on the Internet and at your local bookstore on how to harden Linux systems. But for this paper I am hoping to summarize the major points in each and eliminate the points that are not necessary for these types of configurations so that you can apply the resulting information specifically to firewall and/or gateway machines.

Prerequisites

It should be noted that this document provides a roadmap when setting up a secure firewall/gateway system to your internal network, but many details of the actual implementation (as well as listing every possible thing you can do) are beyond the scope of this paper. The best source for filling in the gaps in this paper can be found in the Linux HOWTOs on your local system or on the 'net.

Due to the high level scope of this document several major points are assumed to be the case:

1. Some flavor of Linux is already installed on your firewall/gateway machine. NOTE: A good installation is half the battle, it would be extremely wise to search around for hints and tips on how to partition your drive(s), what to install and what not to install, etc. Less is more in our case, so *only* install what you need – for a firewall/gateway that's not much!
2. You have at least 2 NICs installed on the firewall/gateway machine – one facing the Internet and the other facing your internal network, IP Masquerading (a.k.a. NAT) configured.
3. You have some form of Internet connection.
4. You are comfortable enough with flavors of UNIX to be able to configure your system and use the available tools.

Once your firewall machine is up and running, start by making sure that you have physical security taken care of, strong passwords/password shadowing in place, and TCP Wrappers installed. Most Linux distributions these days will take care of password shadowing and TCP Wrappers by default, but if yours was not set up this way take the time to do it now!

Configure TCP Wrappers

TCP Wrappers is a binary that wraps itself around the inetd services, such as ftp and telnet. A request to connect to an inetd service is first sent to tcpd which logs the connection attempt and checks it against the access control list. If the connection is permitted it launches the appropriate binary, otherwise the connection is dropped. TCP Wrappers comes stock on almost all Linux distributions these days, but the catch is that it doesn't come configured. The first thing you should do is strengthen your access control by telling TCP Wrappers that nobody but the localhost and the machines in your private address space are allowed to connect to the firewall machine. This is accomplished by editing the */etc/hosts.deny* file to contain this line only:

ALL: ALL

And your */etc/hosts.allow* file to contain these lines:

ALL: 127.0.0.1

ALL: 192.168.1.

This assumes that you have chosen 192.168.1.xxx as your private address space. There are plenty of other options available, so just make sure to replace 192.168.1. with whatever address range you have chosen for your subnet.

Eliminating Services

Another step you should take prior to connecting to the Internet is to shut down as many services as you can. These processes can be found by running "netstat –at" (or "netstat –apt", if logged in as root, will show you what programs are using the open ports). These processes can be started from several places at boot time, but most of the nasty ones (telnet, ftp, etc.) can be found in one of two places:

/etc/rc.d/rc[0-5].d/ - where 0 - 5 is a runlevel number

- The entries in these directories are symbolic links that can be removed by either deleting the file, renaming it (replacing "S" with "s" for example), or in 2.2.x versions of the Linux kernel typing "chkconfig <service name> off".

/etc/inetd.conf - this is the place where services like ftp, telnet, auth, etc are started

- The entries that do not contain a "#" at the start of the line are started up when the system boots up. To disable a service from coming up at boot time simply place a "#" at the beginning of the line and save the file. The next time you reboot the system the service will not be started. In most cases you can comment out everything in */etc/inetd.conf*. If you do need to start services make sure that services started by this file are wrapped by tcpd. You can tell if this is the case if you see something like "usr/sbin/tcpd in.<service>" in the second half of the line of the service you plan on running.

In most situations, you will not want any services listening to the outside world on a machine that is a dedicated firewall, perhaps with the exception of SSH if you want to be able to access your system remotely. The bottom line here is that if you don't need it, don't run it. Every service that you offer up to the outside world gives intruders another possible door into your network, so weigh your options carefully when it comes to running externally available services on your firewall machine. One way to accomplish this safely is to *chroot* the application. This will force the application to run in it's own "chroot jail", a directory where all of the resources for a given program are stored separately from the rest of the system. This prevents the program from having access to the rest of your system. The down side is that when you update your system, if the chrooted program relies upon one of the updated files you will have to update it in both the chroot environment as well as wherever it belongs for the rest of the system.

As mentioned above, if you do require access to the firewall machine remotely it is strongly recommended that you replace the typical ftp, telnet, and other remote access services with SSH (Secure Shell). This provides you with an encrypted method of administering your system as well as a secure way to transfer files from remote locations. It also encrypts passwords preventing cleartext passwords being broadcast over the Internet. SSH requires that the Secure Socket Layer (SSL) be installed on your system. You can get open source versions of SSL from: http://www.openssl.org/ and SSH from http://www.openssh.org/

Permission Checking

The next item on the list is to check for SUID/SGID files on the system. The reason for this is that if your system is compromised there is a possibility that the attacker could exploit weaknesses in SUID/SGID programs in order to gain root privileges. This, of course, means that someone has already gained access to your system (which hopefully won't happen in the first place!) You can run this command to identify SUID/SGID files on your system:

```
find / \( -perm -4000 -o -perm -2000 \) -exec ls -ldb {} \; >> /tmp/suidfile
```

This will output all of your system SUID/SGID files to */tmp/suidfile*. Once this information is available to you, you can go and set more acceptable permissions on these files. Some programs will require SUID/SGID access in order to function properly, but most programs won't argue with taking permissions down a bit. Again, a firewall/gateway should have a minimalist configuration so you probably won't run into any major problems.

Firewall Rules (ipchains)

There is simply no way to write a paper on securing Linux without mentioning *ipchains*. Ipchains is the replacement for *ipfwadm* spotted in kernels prior to 2.2.x. It is very easy to use and is an extremely powerful way to directly specify what traffic is authorized to pass through your network, what should be rejected, what should be dropped,

and what should be logged. Just like with TCP Wrappers, the first thing to do is set your default policies to deny everything. This is accomplished by setting the input, output, and forward chains to deny traffic by default:

ipchains input –P DENY

ipchains output –P DENY

ipchains forward –P DENY

Beyond this, it is your specific needs that dictate how ipchains should be configured. Make sure that you don't forget about your subnet and local host when you set it up. Also note that the chains work in a hierarchical nature. Each packet is compared sequentially to each rule in the chain until it matches one, if it doesn't match any of the rules then the default policy is honored (which is why you set it to deny by default!). It is critical to keep this in mind when writing firewall rules because you could wind up appending rules at the end of a chain that are already handled by a previous rule. Ipchains has an "insert" argument that allows you to specify a location in the chain to address this problem.

The rules that are executed at startup can, in most cases, be found in */etc/rc.d/rc.firewall*. Do a search on the 'net and you will find that there is plenty of reference material for ipchains. It is not difficult to use once you understand how it works. Learn it, know it, live it – it's the heart of your firewall.

Update!

Before going live, you must upgrade your system for the latest security patches and bug fixes. If you have access to a machine already connected to the Internet you can download the appropriate patches and copy them to floppies, otherwise burn a CD with the latest patches. Do your best to upgrade before you put your machine on a live connection. Your machine can be compromised literally within minutes of establishing a connection, so be prepared! Keep in mind that updates are released about as often as new exploits, so make sure that upgrading is a *very* regular part of system maintenance.

Logging and Auditing

All flavors of UNIX are rich in logging resources. Just about everything that happens on a system can be logged somewhere. By learning a little about how to use *syslog* and *ipchains* you can script a decent intrusion detection system (IDS), or at least generate log data that you can audit. For a 'pre-packaged' IDS system you can use a combination of a number of available programs, such as *snort* and *swatch*. Snort is a packet-filter, sniffer, and logger that provides basic intrusion detection capabilities. It is rule-based and can either alert you to specific traffic patterns, log the traffic, or let it pass. It requires configuration, but is worth the effort. Swatch, or the System Watcher, is a supplement to your normal syslog logging capabilities, and is much more powerful. Swatch provides real-time monitoring, logging, and reporting, and has several unique features including "Backfinger" which will try to grab finger information from an attacking host, and instant paging.

You should also monitor who is logging in, or attempting to log in, to your system. *last* and *lastlog* read your */var/log/wtmp* and */var/log/lastlog* files, respectively, to see who has been logging in to your system and when. This isn't foolproof on its own however, as this will be one of the first places that an intruder would look to start erasing his or her tracks. There are plenty of log sweepers and cleaners out there that can circumvent your default logging systems, so make sure you have good logging practices. Things like logging remotely if possible, otherwise to read-only media, and setting up redundant logging just in case your main logging system winds corrupted, should be high on the list.

Verification

The next thing that you should do before putting your connection live is to test it out by scanning for vulnerabilities. There are many tools out there to do this. Some of the more famous are *nmap*, *Nessus*, and *SATAN* (Security Administrator's Tool for Analyzing Networks), but there are quite a few choices. There are also commercial-grade solutions such as CyberCop by Network Associates, Inc (http://www.nai.com), and Internet Security Systems (ISS) System Scanner, Internet Scanner, and RealSecure IDS products (http://www.iss.net). Whichever tool you choose, make sure that you *do* scan your system for vulnerabilities. Doing this will alert you to security holes that you may have overlooked in your system configuration, and it will also show you what a potential intruder might see when they scan your network. Obviously useful information!

In addition to the above, some form of file integrity checker should be put in place. Make sure that when you do this

you baseline a *fresh* system install, not a dated one, and not one that has had exposure to a live Internet connection. If your machine has already been exposed to the Internet, especially with inadequate security measures in place, you could wind up with a corrupt baseline of your system. Although you can make your own sort of "poor man's tripwire", there is a very good open source application out there for Linux called, you guessed it, TripWire. TripWire is a package that takes a "snapshot" of important system binaries and stores their digital fingerprints in a TripWire database. This database should be stored offline or on read-only media to ensure integrity. TripWire will notify you if the fingerprints on your current system don't match those stored in the database. Unless you have changed the file(s) yourself, you should know at this point that there is something amiss within your system.

Alternatives

If you are protecting a small home network or SOHO setup and aren't game to the major commitment of configuring and maintaining a firewall, there could be another solution for you – packet-filtering routers. Routers are external devices that are operating system and application neutral. They are there to handle accepting or denying traffic before it gets to your machines. Some advanced routers are even able to defeat DoS attacks and spoofing, as well as make your system invisible to the outside world. There are drawbacks to this solution however. First, many router-based firewalls *are* susceptible to spoofing and other types of attacks, especially the bargain ones. Second, network performance can really decline if you choose to use strict packet-filtering rules – this may or may not be an issue for you depending on how much incoming traffic you expect. And lastly, a good router-based firewall is *expensive*. You really do get what you pay for when it comes to these little marvels.

I should also mention a very nice software tool for hardening your Red Hat or Mandrake system called Bastille-Linux. Bastille basically does a once-over on your system and looks for things to tighten down. It will write firewall rules (ipchains) for you, it will shut down unnecessary services, and it will find and correct SUID/SGID files, among many other things. The GUI is an interactive script that asks you questions about your system and generates output based on your answers. One of the nicest features of Bastille is that when it is asking you questions it tells you *why* it is asking them, making running the program a good learning experience. It is available from http://www.bastille-linux.org/ and can be run by untaring in your */root* directory (tar –zxvf Bastille-version), running the interactive script ./InteractiveBastille.pl, and finally running ./BackEnd.pl to actually make the changes.

Summary

What has been mentioned in this paper is really the tip of the iceberg and should be done as a minimum before you take your Linux firewall/gateway machine live. There are NO guarantees when it comes to security and there is no single solution – good security is a series of layers making it so that if part of your system is breached the intruder will just come up against another obstacle, one after the other. The same principal applies to car alarms as computer systems – if someone wants in bad enough they *will* eventually get in, no matter how far you have gone to beef up your security. At the same time, there is no point in making it easy. By tightening your system down you will dissuade script kiddies and those who have too much time on their hands (and are low on ambition) from targeting your system in the first place. Take the time to do some research and see what others have done to their systems before taking them live and learn from those experiences. There are many great resources on the Internet as well as books that can aid in this process and take you through securing a system in much more detail. If you value your privacy and your systems, take security seriously and learn how to protect yourself. Put the penguin at your front door.

Resources

Coddington, Dale. "Securing Linux". 30 MAR 2000.
URL: http://www.securityfocus.com/focus/linux/articles/linux-securing.html

Vertes, Peter. "Securing Your Linux Box". NOV 1998. Linux Gazette. #34
URL: http://www.linuxgazette.com/issue34/vertes.html

"Securing Linux Mandrake". 21 AUG 2000
URL: http://rootprompt.org/article.php3?article=821

Jasen, John E. "SYLS101: Securing Your Linux System 101".
URL: http://userpages.umbc.edu/~jjasen1/unix/linux.html

Boran, Se‡n. "Hardening Red Hat Linux With Bastille". 30 APR 2000
URL: http://www.securityportal.com/cover/coverstory20000501.html

Anonymous. "Maximum Linux Security". JUN 2000. SAMS Publishing.
ISBN: 0-672-31670-6