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 (Security 401)"
at http://www.giac.org/registration/gsec

Using open source to create a cohesive firewall/IDS system.

Thomas Dager

GSEC practical (v 1.2e)

Overview

Defense in Depth is a basic concept, wherein the defender seeks to apply designated, concentric layers of defense in an effort to detect and deter an enemy. Attackers are faced with breaking through or bypassing each layer without being detected, a difficult task. Another benefit is that a flaw in one layer can be covered by other layers, thus mitigating a mistake in the implementation of a particular layer of defense. There are many components that make up the defensive layers: ip firewalling, tcp wrappers, application access control, intrusion detection, encryption and many more. In this paper I will be discussing what are arguably the two main components of the layered defense, a firewall and intrusion detection system. More importantly I will show how to use existing open source technologies to combine these into a comprehensive whole that, while not providing a total solution, can go a long way in fulfilling the defense in depth strategy.

So where do we start?

First we must install a secured version of Linux…any flavor is fine. The best place to start in securing your system is during OS installation. You should not have any previous installations. You will want to start with a clean installation where you can guarantee the system integrity. Place your system in an isolated network, or do not even attach it to a network until the hardening is done. NEVER connect this box to an active network nor the Internet, until it is hardened. There are many, many things that need/can be done to a standard Linux installation to harden it. Since this is beyond the scope of this paper you may want to take a look here for a more in-depth look at hardening your Linux system: http://www.enteract.com/~lspitz/linux.html.

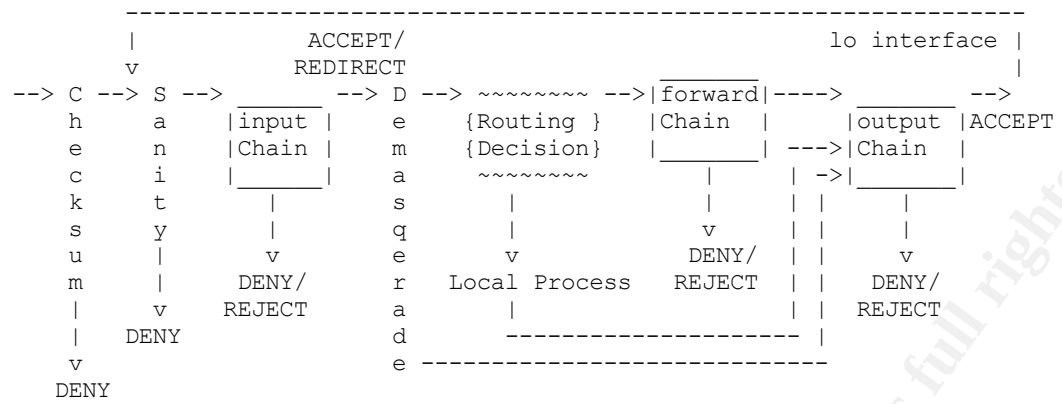
OK now what?

Next you need to make sure that your kernel has the IPCHAINS firewall in it. Look for the file '/proc/net/ip_fwchains', if you see it your kernel has it, if not you will need to make a kernel that has them. That is beyond the scope of this paper, however the Kernel HOWTO can help. The good news though is that if your kernel is numbered 2.1.102 or higher then you are in luck, it is in the mainstream kernel.

Yippee! It's there!

Next you will be using the tool IPCHAINS to configure the firewall. Basically IPCHAINS interacts with the kernel and tells it what packets to filter. Understand that in a strict sense IPCHAINS is a simple packet filter. In other words it has a set of rules (which you create) that controls all traffic incoming and outgoing. So if you set up the chains wrong, such as having your input policy set to accept, then your firewall will provide little if any protection! Now, before we set up IPCHAINS we need to know how they work. Here is a simple diagram of the path a packet takes as it enters the network taken directly from the Linux IPCHAINS-HOWTO by Rusty Russell (Russell, 13):

© SANS Institute 2000 - 2005 Author retains full rights.



As you can see this can get somewhat complicated. However, the thing to remember is that there are several levels of security that you can provide. The main thing should be to set a policy of DENY on the input chain. If you do not, and any input does NOT match a rule…well, it will be accepted anyway, thus negating your whole input chain. You may set the output chain for DENY as well. However, this will entail significantly more work on your part, as you must then determine ALL the outbound services that you need and what ports they may operate on. Of course, this does provide some protection against Trojans that may be on your systems by blocking their outbound path. I suggest you set an output chain policy of ACCEPT until you understand what services may be blocked (you CERTAINLY do not want to block your employees' AIM connection, now would you?!). Here is a simple, yet effective setup:

Chain input (policy DENY):

^-As you can see, the policy is set to DENY which is GOOD!!

target	prot	opt	source	destination	ports
ACCEPT	all	------	127.0.0.1	0.0.0.0/0	n/a
ACCEPT	all	------	192.168.30.0/24	0.0.0.0/0	n/a

^-Trusted host, use ONLY if you are SURE that the host can be FULLY trusted.

ACCEPT	icmp	------	0.0.0.0/0	0.0.0.0/0	4 -> *
ACCEPT	icmp	------	0.0.0.0/0	0.0.0.0/0	12 -> *
ACCEPT	icmp	------	0.0.0.0/0	0.0.0.0/0	3 -> *

^-Accepts only SPECIFIC ICMP (in this case 3 - Destination Unreachable, 4 - Source Quench, 12 - Parameter Problem) traffic, all others are blocked.

Now the above settings are VERY restrictive. Basically unless something is from the machine itself or the trusted host or one of the three specific ICMP types, it is BLOCKED. Remember that as you set yours up.

Chain forward (policy DENY):

target	prot	opt	source	destination	ports
routes	all	------	0.0.0.0/0	0.0.0.0/0	n/a
portfw	all	------	0.0.0.0/0	0.0.0.0/0	n/a
MASQ	all	------	192.168.17.0/24	0.0.0.0/0	n/a

Chain output (policy ACCEPT):

target	prot	opt	source	destination	ports

© SANS Institute 2000 - 2005 Author retains full rights.

```
ACCEPT     all     ------  0.0.0.0/0          127.0.0.1          n/a
ACCEPT     all     ------  0.0.0.0/0          192.168.30.0/24          n/a
ACCEPT     all     ------  0.0.0.0/0          10.10.23.0/25      n/a
ACCEPT     icmp    ------  0.0.0.0/0          0.0.0.0/0          4 ->   *
ACCEPT     icmp    ------  0.0.0.0/0          0.0.0.0/0          12 -> *
ACCEPT     icmp    ------  0.0.0.0/0          0.0.0.0/0          3 ->   *
```

Can you see what is wrong with the above ruleset (chain output)? Since the output policy is set to ACCEPT, the other accepts within the chain are superfluous. Make sure that when you set up your rules that you pay close attention to policies and chain rules and how they will interact. What should be in the chain would be any DENY's that you wanted, since once you reach the end of the chain it will default to its policy if nothing is matched within that chain. This is where the different views in firewall construction come into play. Some prefer to start restrictive and loosen things up as needed and others prefer to start open and restrict as necessary. The decision on which one to use will be dictated by both the corporate environment and the individual administrators.

The simplest way to add a rule to a chain is with the following command:

ipchains –A(add) input(chain) –s x.x.x.x/x(source IP) –d x.x.x.x/x(destination IP) –j DENY (what action this particular rule is to do)
*note that the items in brackets are comments for clarity

So here is an example:

ipchains –A input –s 192.168.4.0/24 –d 0.0.0.0/0 –j DENY

So you have entered your ruleset and everything is fine, right? Wrong. There is one final thing you need to do. As your firewall setup is stored in the kernel, upon reboot you will lose these settings. Horrors, no!!! So what we need to do is make these rules permanent. After you have created a ruleset run (as root):

ipchains-save > /etc/ipchains.rules

Once you have that you will then need to create a script that runs every time the system boots that will load your ruleset. There is a simple and effective script listed in the IPCHAINS-HOWTO.

For a more in-depth look at setting up your ruleset take a look at the IPCHAINS-HOWTO or Ipchains and IP Masquerading by George Bakos. Once you have your ruleset in place you then need to setup the IDS portion of this tag-team.

Let's see some intrusion attempts!

We will be using one of the best open source Intrusion Detection Systems available, SNORT! According to Snort's website:

> Snort is a lightweight network intrusion detection system, capable of

© SANS Institute 2000 - 2005 Author retains full rights.

> performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealthport scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture. Snort has a real-time alerting capability as well, incorporating alerting mechanisms for syslog, a user specified file, a UNIX socket, or WinPopup messages to Windows clients using Samba's smbclient (What is Snort?)

Wow, what great program! So where do I get it?? Well you can get it here! You will also need to have libpcap installed, and that can be found here.

Setting it up.

Now that you have it you need to install it. Well it is simple. The following commands are taken directly from the install file that comes with the snort package (Roesch, Install file):

1.) *** Make sure you have libpcap installed!!! ***
2.) ./configure
3.) make
4.) make install
5.) Create a sample rules file (if you want to use rules, check out the included snort.conf file)
6.) snort -?
7.) If you've used previous versions of Snort, you may need to rewrite your rules to make them compliant to the rules format.
See http://www.snort.org/writing_snort_rules.htm for more information.
8.) Have fun!

Now remember that this is a GENERIC install. Check the Snort.org website for possible precompiled packages for your particular Linux distribution. If you need to do unusual things to compile the package you will have to figure out how `configure' could check whether to do them. Also by default, `make install' will install the package's files in `/usr/local/bin', `/usr/local/man', etc.

Once you have it fully installed it is time to become familiar with the usage of snort. I will cover some of the basics; for more in-depth information please see the Snort FAQ or Mark D. Tollison's An Analysis of the Snort Network intrusion.

The Basics

The first thing we want to do is get snort running. To run Snort in sniffer mode type:

snort -dvi eth0 (making sure that the interface listed is one that can see the traffic)

and make sure it can see the packets. Then run it with the HOME_NET set appropriately for the network you're defending in your rules file.

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

Cool but what the heck are –d this and –v that for?

Here is a small sample of the various switches and their meanings. More information can be found in the Snort FAQ.

```
Command line:

      snort -[options] <filters>

Options:

    -A <alert>  Set <alert> mode to full, fast or none.  Full mode
            does normal "classic Snort"-style alerts to the alert
            file.  Fast mode just writes the timestamp, message,
            IPs, and ports to the file.  None turns off alerting.
            There is experimental support for UnixSock alerts
            that allow alerting to a sepreate process.  Use the
            "unsock" argument to activate this feature.

    -d      Dump the application layer data

    -i <if> Sniff on network interface <if>.


    -v      Be verbose.  Prints packets out to the console.  There
            is one big problem with verbose mode: it's still kind
            of slow.  If you are doing IDS work with Snort, don't
            use the -v switch, you WILL drop packets (not many, but
            some).
```

Of course one of the most important switches for our purposes is the '-c' switch. With this switch Snort enters IDS mode when a configuration file is specified with the "-c" switch. Output formats, rules, preprocessor configuration, etc are all specified in the configuration file.

More rules??

Yes. Just like everything else in Linux, Snort is based on configuration files. You must either create or download a ruleset to use. For information on creating your own rules look here.

Now to bring it all together.

That was the whole point of this, right? Well here it is: to combine these two components into a functional whole. To do this we will use a third piece of code called Guardian. So what exactly is Guardian? Well, it is a stand-alone Perl script which watches the output of snort, and will add rules to IPChains on the fly as snort detects and reports an attack.

© SANS Institute 2000 - 2005 Author retains full rights.

Of course before it does this, it checks to make sure it is allowed to block the attackers IP address. This does prevent Guardian from blocking spoofed packets which are crafted to look like they originated from a trusted machine, but that is an inherent risk with Snort itself.

Setup

It is imperative that both Snort and Guardian run as root. Guardian needs to run as root so that it can issue ipchains commands and Snort must run as root so that it can set the network card in promiscuous mode.

- The first thing you will need to do is edit the guardian.conf file.
- Now you need to copy the guardian.conf file to /etc/guardian.conf (the default place it looks), or you can run Guardian with the -c option and tell it where the config file is located at.
- Next, you will want to set up your ignore file. This is the file you defined in the guardian.conf file with the IgnoreFile keyword.. It is a good idea to put your DNS servers, gateway, and any other remote machine's IP you access often.
- Once you have that in place, the next thing you need to do is create your log file, just a simple 'touch /var/log/guardian.log' should work.
- You need to set the timeLimit value in the config. This will allow Guardian to delete IP's from the rules table if the age of the entry is greater than the timeLimit value.

Now run Guardian!! You will want to have Guardian run as a part of the script you created earlier for Snort. This script will allow both Snort and Guardian to run at boot up and protect your system fully.

Conclusion

So what do you have after reading through this? You have a robust firewall that is mated to an effective intrusion detection system. These two, when combined, can form the core of any layered defense system. You can then institute other security measures that can only add to the level of security that is provided. Just remember that once your firewall/ids hybrid is set up your work is not done, especially in regards to the Snort rules. Since new exploits are found every day new rules are also created. You can either download the latest rules from http://www.snort.org/ or even write one yourself. Just make sure that you submit it to the Snort database. After all, we need every advantage we can get when we are facing the barbarians at the gates!

References

Bakos, George. "Ipchains and IP Masquerading. The Semi-Stateful Packet Filtering

© SANS Institute 2000 - 2005 Author retains full rights.

Firewall for All of Us Poor Folks." 2 October 2000.
URL: http://www.sans.org/infosecFAQ/firewall/ipchains.htm (24 June 2001)

Roesch, Marty. Snort install file. 2 January 2001. URL: (Part of the Snort package)
http://www.snort.org/Files/snort-1.7.tar.gz (25 June 2001)

Russell, Rusty. "Linux IPCHAINS-HOWTO." v1.0.8. 4 July 2000. URL:
http://www.ibiblio.org/pub/Linux/docs/HOWTO/IPCHAINS-HOWTO (23 June 26, 2001)

Spitzner, Lance. "Armoring Linux." 19 September 2000. URL:
http://www.enteract.com/~lspitz/ (23 June 2001)

Snort FAQ. URL: http://www.snort.org/FAQ.html (26 June 2001)

Tollison, Mark D. "An Analysis of the Snort Network intrusion" 10 December 2000.
URL:
http://www.sans.org/infosecFAQ/intrusion/snort2.htm (24 June 2001)

What is Snort? URL: http://www.snort.org/what_is_snort.htm (26 June 2001)

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

© SANS Institute 2000 - 2005 Author retains full rights.