Global Information Assurance Certification Paper

When Time And Money Are Not On Your Side

Jason E. High, MCP
GSEC Practical Version 1.2f
January 9, 2002

Introduction

Time and money are two things that every organization always needs more of. This paper is designed to give systems administrators a guide to quickly deploying a relatively secure information security infrastructure. I am going to cover several of the free tools available that can be combined to provide a base level of security with very low time and money investment.

Please note that this paper is not designed to be an exhaustive reference on any of the tools covered. There are a lot of things that you can do with Snort, for example, that I'm not going to cover. The goal here is to allow systems administrators to assess and secure their environment quickly without having to spend thousands of dollars on equipment, training, etc. I'm going to be making some assumptions that are potentially dangerous. It is important for each organization to decide what an acceptable level of risk is to them and to alter their security plan and policy accordingly. This guide is supposed to be a starting point that will get some security in place for sites that have either no security or a very negligible amount of security.

Vulnerability Assessment

One very important aspect of information security is the concept of vulnerability assessment. Vulnerability assessment is basically you finding the security holes in your organization before the attackers do. Please make sure you have written and signed permission from the appropriate authorities in your organization before doing any vulnerability assessment.

Nessus is a very robust and effective vulnerability assessment too available from http://www.nessus.org. The Nessus daemon runs on the UNIX/Linux platform, with clients available for UNIX/Linux as well as Windows and others. It is a good idea to install the Nmap port scanner before installing and using Nessus. Nessus will take advantage of Nmap if it is present.

The first thing to do before starting the scan is to decide what hosts you will scan. You can scan a single host, multiple hosts, an entire subnet, or have Nessus read a list of hosts from a file that you create. The best idea is to start off scanning only one or two non-critical systems so that you can learn the tool. From there you can start scanning entire subnets and critical systems. I have only one subnet in my organization, so the order of scanning that I followed was:

- A Windows 2000 computer that held no mission-critical data or applications
- All non-critical hosts. I created a file named non.critical.hosts and listed the IP addresses there.
- All critical hosts, one at a time.

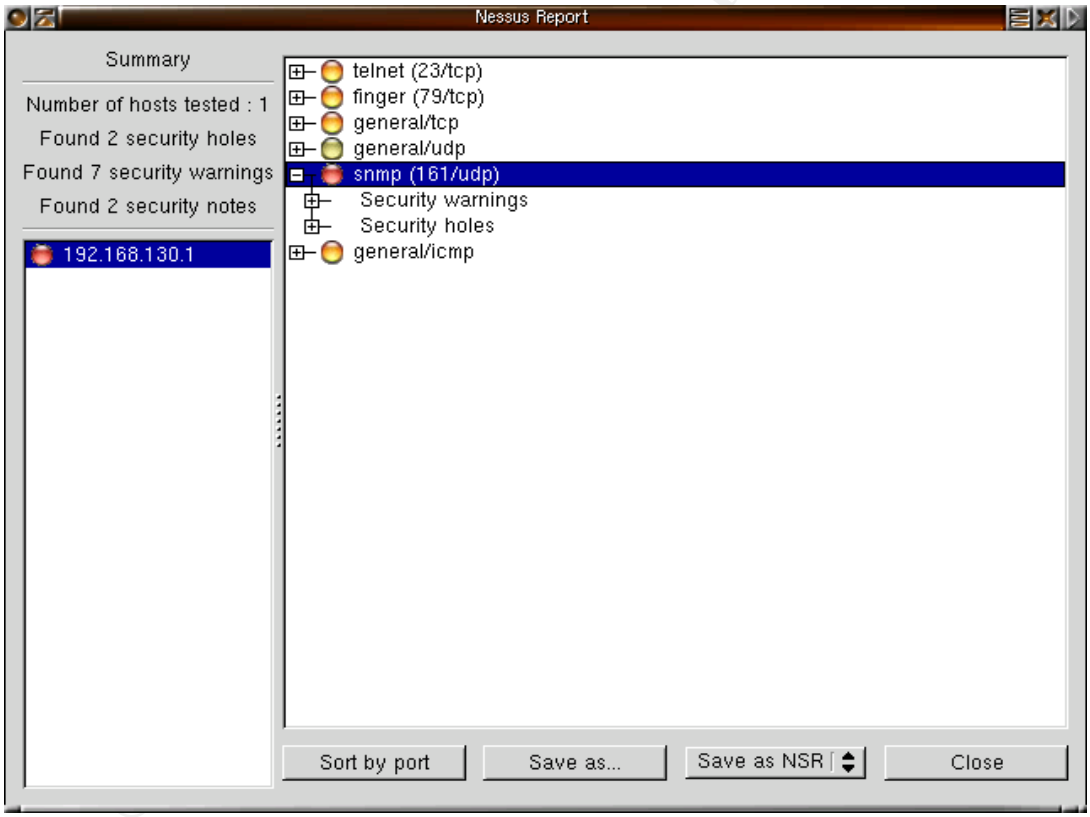
I like this approach because at no time am I in danger of bringing down more than one

mission-critical system at a time.

You also need to choose what plugins you will enable. The best idea is to start off by clicking the Enable All But Dangerous button. After you've run the tool like that, you can go back and enable any Dangerous plugins that you wish. While enabling dangerous plugins will provide for the most comprehensive scan, it can also mean downtime for critical systems. They're called Dangerous Plugins for a reason. Use your discretion.

You can now go through and enable any Preferences that you want enabled. The default preferences will provide a fairly thorough scan, so go with them to start. You can also set any options that you wish. I scan all ports (1-65535), leave the max threads set to 8, tell Nessus not to do Reverse Lookups, and leave everything else as is. When you've set all that (or at least looked at it to see what is available), click Start Scan.

Nessus will take a while to run and then produce a screen similar to the following:



Nessus will give you three types of warnings, each more severe than the next. The types are:

- Security hole - This is the most severe. Nessus found a security hole that you need to patch or fix.
- Security warning - This means that Nessus found a potential vulnerability, but that there is not necessarily a hole. You should investigate all security warnings to determine whether or not they are applicable.

- Security note - A security note is more like an FYI. For example, Nessus will give you the trace to the host that you scanned in the form of a security note.

You should now take the output of Nessus, close any security holes discovered, and take all security notes into consideration. This is a great first step toward securing your infrastructure, as well as a great way to learn more about your network. You should also set up a schedule to run Nessus at various times to ensure that no new vulnerabilities are introduced onto your network.

Intrusion Detection

The next step in securing our network is going to be intrusion detection. Not only can intrusion detection do intrusion detection, it also helps as a vulnerability and risk assessment tool. It's difficult to justify spending money on information security to management without providing them with some indication that there is actually a security risk. Snort, a free, lightweight intrusion detection system will help you do just that.

Snort runs on a variety of platforms (including Windows) but prefers the UNIX/Linux environment. Also, since Linux is so cheap and runs on just about anything, it is more cost-effective to run Snort on Linux. Snort is available as a free download from http://www.snort.org. Note that Snort can also be used as a packet sniffer and logger.

To start Snort in intrusion detection mode, try: `/path_to_snort_binary/snort -d -h 192.168.130.0/24 -l ./snort.logs -c snort.conf.` This starts Snort using the snort.conf rules file, the default configuration file that comes with Snort. It will collect application data as well as TCCP/IP/UDP/ICMP headers. Snort will log to the ./snort.logs directory and will log relative to the network 192.168.130.0/24 (you should replace this network and mask with your own.

Snort will begin logging any attempts at intrusion based on the rules specified in your snort.conf. The default configuration that comes with Snort should suit your needs initially, and is definitely better than not having intrusion detection at all. If anything you will need to tweak it to keep the number of false positives down. If you have the time and resources to do more research it is a good idea to learn how to write your own rules for Snort to respond to specific events that occur within your organization.

Snort will log any alerts in the alert file, and will log more detailed information in a directory named after the intruding host. Snort also maintains a portscan.log file that will log all portscan attempts. Set up a schedule for reviewing these files and possibly establishing firewall rules that will filter any IP addresses, subnets, domain names, etc. that seem to be frequently attacking your site. Please be aware, however, that Snort can and will generate false positives. There's nothing worse than performing a Denial Of Service attack on yourself because you misinterpreted some Snort logs.

If you're in a switched environment and have no way of enabling port forwarding or something similar, a good place to put Snort is either just outside the firewall (between your firewall and the rest of the world), or just inside (between the firewall and your

switches). This would allow Snort to see all traffic entering and leaving your network, which is definitely better than nothing. If you can, put Snort in both places for maximum benefit. If you have to choose one, I prefer to put Snort between my network and firewall, rather than between the firewall and the Internet. This allows Snort to see traffic that is actually making it into my network. Hopefully your firewall is stopping many of the intrusions that Snort is likely to detect if placed outside the firewall. Placing Snort outside the firewall is, however, good for research, risk assessment, etc.

Firewall

And now we come to the subject everyone always thinks of first when they hear the phrase information security...the firewall. Your firewall is your perimeter defense, so it is definitely a very important piece of the puzzle. For the firewall we'll be looking at Iptables, a utility included with the 2.4 Linux kernel.

To use firewall support in the 2.4 kernel, you have to compile certain modules into the kernel. Check out the Iptables Tutorial at http://www.linuxsecurity.com/resource_files/firewalls/IPTables-Tutorial/iptables-tutorial/iptables-tutorial.html for a pretty complete list of modules as well as a very informative and in-depth discussion of Iptables. For most distributions of Linux, however, you can choose to install firewall support at install time, and this will give you the necessary support.

I want to stress that there is a lot that you can do with Iptables that I will not be covering here. I'm going to cover some of the commands to set up and maintain a basic, packet filtering firewall. I will not be covering state, ip masquerading, nat, etc. I want to help you learn how to setup a firewall quickly and easily, and I want you to be able to maintain it. Look in the List Of References to see some more in-depth papers on Iptables.

Use the command `iptables` to access Iptables (please note that you must be logged in as root). Iptables takes the form of: `iptables <command> <match> <target or jump>`. An example would be: `iptables -t filter -A INPUT -p tcp -j DROP`. This tells Iptables to use the filter table, to append a rule to the INPUT chain that says to DROP any packets that match the TCP protocol. Iptables currently uses three tables: filter, nat, and mangle. I'm only going to cover the filter table, which is the default. The nat table is used for rules related to Network Address Translation (NAT), and the mangle table is used to alter packets before they are sent out. If you don't specify a table Iptables will use the filter table by default.

The filter table consists of three built-in chains:

- INPUT - these rules are applied to packets coming into the firewall
- OUTPUT - these rules are applied to packets that originate at the firewall
- FORWARD - these rules are applied to packets that are being routed by the firewall to another host.

The commands that you'll most often use to manipulate Iptables are:

- `iptables -L <chain>` - list all of the rules in a chain. If you specify just

`iptables -L` (with no chain specified), Iptables will list all rules in all chains.

- `iptables -A <chain>` - appends the rule to the bottom of the chain specified.
- `iptables -D <chain> <rule>` - deletes a rule from the chain. You specify the rule either numerically (the first rule listed is 1) or by typing out the rule.
- `iptables -R <rulenum>` - replace the rule at the specified number in the chain.
- `iptables -I <rulenum>` - insert the rule at the specified number in the chain.
- `iptables -P <chain> <target>` - tells Iptables what the default policy for the chain is. The valid targets are either DROP or ACCEPT.

Those are the basic commands that Iptables recognizes. There are more commands that Iptables uses to manipulate tables, but they're outside the scope of this paper. Read the man page for more information.

Now we'll talk about matching. The concept of matching is just giving Iptables criteria that the packets must meet in order for the rule to apply to them. Some of the criteria you can use are:

- `iptables -A INPUT -p <protocol>` - the packet must match the given protocol for the rule to be applied. Iptables recognizes all protocols listed in /etc/protocols.
- `iptables -A INPUT -s <address></mask>` - packets must match the specified source address for the rule to be applied. Addresses take the form of address/mask. For example, 192.168.130.0/24.
- `iptables -A INPUT -d <address></mask>` - packets must match the specified destination address for the rule to be applied. Addresses use the same format as for the source address.

Those are probably the three most common criteria used. There are many other options for each, some of which we are going to see in my examples. So now we've learned how to specify a table, chain, and criteria...but what will Iptables do with the packet? Well, that's our target specification.

There are four built-in targets:

- ACCEPT - means to let the packet through the firewall
- RETURN - means to stop traversing the current chain and to return to the next rule in the previous (calling) chain.
- QUEUE - means to pass the packet to userspace.
- DROP - means to drop the packet.

You specify these targets with the -j option, as in: `iptables -A OUTPUT 192.168.130.0/24 -j ACCEPT`. This command tells Iptables to accept all outbound traffic from 192.168.130.0/24. You can also jump to another chain, but I'm not going to cover that here. Now I'm going to give some real-world examples. Rather than showing you how to actually set up a firewall, which could take volumes and would be subject to lots of debate, I'm going to show you how to secure an individual host. This should give you an idea of how to use the commands effectively and give you a good starting point for deploying your own Linux firewall.

First things first, I want the default policy on all three chains to be set to DROP. I issue

the following three commands: `iptables -P INPUT -j DROP`, `iptables -P OUTPUT -j DROP`, `iptables -P FORWARD -j DROP`. Then use `iptables -L` to make sure that all three chains now have their default policy set to DROP. Realize that this is a very extreme starting point. If you're testing this on your computer as you go (like I am), you now have no connection to the outside world. For most of us, this is an unacceptable condition. The two most common tasks that require connectivity are sending and receiving email, and surfing the internet. But, there's something that we must do first: `iptables -A OUTPUT -p tcp --destination-port 53 -j ACCEPT`, `iptables -A OUTPUT -p udp --destination-port 53 -j ACCEPT`. These two commands will allow DNS to function for host name resolution. Now that we can resolve hosts, let's see if we can set up our email. The TCP port for SMTP is 25 by default, so we'll use this command to send mail: `iptables -A OUTPUT -p tcp --destination-port 25 -j ACCEPT`. Try sending a test message to make sure that it works. Note that you won't receive it until we open up POP traffic, so why don't we do that now. POP uses port 110 by default, so we'll use this command to finish opening up our email: `iptables -A INPUT -p tcp --source-port 110 -j ACCEPT`. Now you should be able to receive the message that you sent to yourself. We now have one final service to open up, and that's http. Issue this command to open up web browsing: `iptables -A OUTPUT -p tcp --destination-port 80 -j ACCEPT`. You should now be able to perform DNS lookups, send and receive email, and browse the web. As you can see, it would be fairly easy to open up just about any service. If you don't like setting the default policy on all three chains to DROP initially, you've got a lot more work to do. You'll have to issue commands that not only allow traffic through, but also deny certain traffic. For instance, to deny all ICMP packets, use `iptables -A INPUT -p icmp -j DROP`. Again, you'll have to issue a lot more commands, and having the default policy set to DROP is probably the best and safest way to go. Either way, you can now use these commands to set up your own Linux firewall.

Summary

I realize that I didn't cover a lot of what these tools can do. However, the goal here was to give systems administrators a quick-start guide to getting a security infrastructure up and running. The three of these tools together can form a very strong information security base for any organization. You can take the basics that you learned here and expand upon them with these tools, or add new tools such as proxy servers, host-based intrusion detection systems, etc. Either way, you're well on your way to securing your organization's assets.

List Of References

Roesch, Martin. "Snort Users Manual\Snort Release 1.8.3"
URL: http://www.snort.org/docs/writing_rules

Stark, Vernon. "Nessus – A Very Capable Security Auditing Tool" (August 6, 2000)
URL: http://rr.sans.org/audit/nessus.php

Andreasson, Oskar. “IP Tables Tutorial” (November 11, 2001)
URL: http://www.linuxsecurity.com/resource_files/firewalls/IPTables-Tutorial/iptables-tutorial/iptables-tutorial.html

Tollison, Mark. “An Analysis of the Snort Network Intrusion Detection System” (December 10, 200)
URL: http://rr.sans.org/intrusion/snort2.php

Sutherland, John. “4th Generation of Linux Based Firewalls” (April 11, 2001)
URL: http://rr.sans.org/firewall/4th_gen.php

Drake, Joshua. “10 minutes to an iptables-based Linux firewall” (September 24, 2001)
URL: http://www.linuxworld.com/site-stories/2001/0920.ipchains.html

LeCunt, David. “Linux.com Security: Firewalls, IPtbles, and Rules” (June 13, 2001)
URL: http://www.linux.com/enhance/newsitem.phtml?sid=1&aid=12431