Global Information Assurance Certification Paper

Defense In Depth: Company B's Open-Source Network Security Strategy

GSEC Practical Version 1.4b
Jeff Wolfe
April 17, 2003

1 Introduction

The importance of network security is well established and heavily emphasized. Despite this, many businesses, small and large, maintain environments seeming to invite a network attack. Some of the reasons for this include lack of security knowledge, lack of human resources, and/or lack of funding. Considering the latter, network security costs can be minimized without compromising security quality and reliability through open-source network security tools. Utilizing a few of these adds layers to a company's defense in depth strategy at minimal costs. This discussion focuses on implementing multiple open-source utilities in order to provide network-based security through a strategy of defense, detection, and prevention. The necessary security components are firewalls, intrusion detection systems, logging, and vulnerability scanning.

Simply obtaining and implementing these free utilities alone, however, does not yield a secure network. A solid understanding of network and security concepts is also necessary; including host-based security measures, security policies, updating and patching, and more. Certain aspects of these areas are mentioned briefly throughout this discussion but details are outside the scope of this paper.

2 Open-source vs. Commercial

Battles for and against open-source solutions as an alternative to commercial solutions for network security are still being waged. Open-source programs endure unequalled source code scrutiny that serves to provide quality levels equal to if not better than commercial products. The development process of open-source continues into the distribution process with many consumer eyes focused on bug recognition. These bugs are therefore often rapidly reported and fixed.[1] This theory of quality assurance however becomes questionable when reading the following excerpt found in Dennis Fisher's article on eWeek's website: "Of the 29 advisories issued through October [2002] by the CERT Coordination Center at Carnegie Mellon University in Pittsburgh, 16 of them addressed vulnerabilities in open-source or Linux products." However, the article does continues to state that CERT tends to focus advisories toward higher risk vulnerabilities and that Microsoft published eleven lower risk vulnerabilities of its own in October 2002 alone.[2] Another forte of commercial products is a dedicated source of customer support. Open-source products however provide abundant support through numerous mailing lists, message boards, and dedicated product-specific sites. Open-source programs tend not to be as graphically elaborate, entailing a greater understanding of the underlying concepts. Commercial products typically employ intuitive GUI's with advanced auto-configurations which can speed implementation time. The point is that there are pro's and con's of both.

There may never be a clear winner in this debate but these are some of the real issues affecting company security decisions and at least deserve mentioning. Finally, a clear strength of open-source is that many utilities are free and highly customizable. For organizations of smaller size or extremely tight IT budgets, cost may be the factor that decides the issue. If this is the case, open-source has the solution. The only product cost incurred is the hardware itself.

3 Defining Company B

Security should not be an afterthought once a network is built but rather treated as an actual component necessary to the network and built into the design. Company B must therefore first define the network components necessary to support their services. Company B sells products and services and orders are processed and shipped at the company office by a sales team or via online ordering. The office site includes approximately 40 employees. Three devices compose the DMZ: a mail relay, web server, and an external DNS server. The three servers that are within the internal network server farm include an internal DNS server, DHCP server, and mail server.

Administrative workstations such as accounting, human resources, etc. are to be segregated from the shipping and sales teams' workstations with each group being connected via a switch to their respective network segment. Both groups will have limited access to the Internet.

Figure 3.1 shows the physical layout of Company B's service network along with the logical layout of the implemented security components presented within this discussion. It is the result of the planning discussed within this section. Also shown are network segment address scheme and the interface numbers of the security components and the central router. All servers, switches, hubs, and security devices will be stored in a central, physically secure location under lock and key at minimum. Other security measures, such as logging physical access to machines, host hardening, etc., are necessary but beyond this scope.

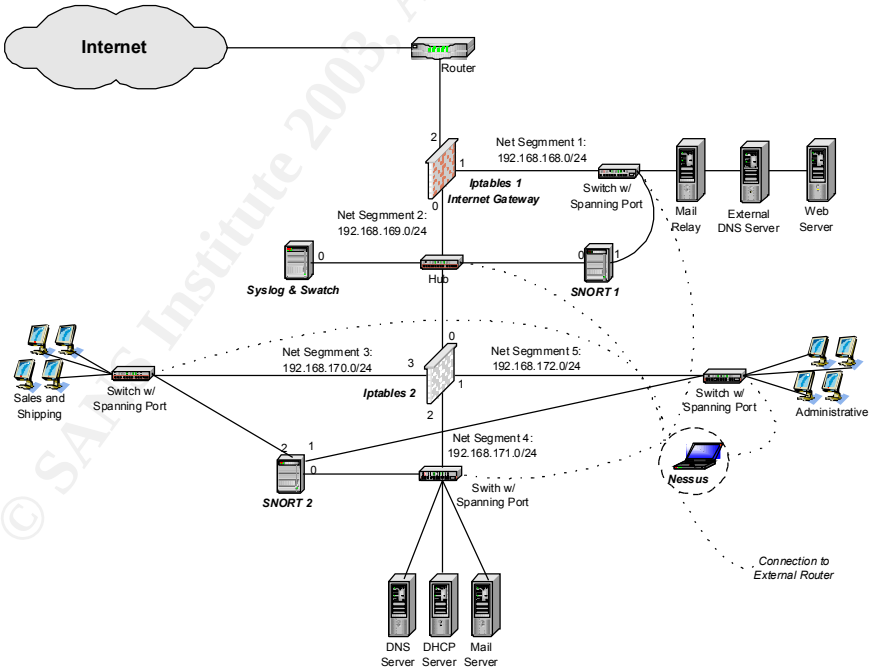

Figure 3.1: Company B's Network

3.a Security Components

Now that a macro view of Company B's service is defined, the planning of security deployment begins. The deployment of the security components requires

multiple computers in order to have manageability, scalability, and non-dependency. The solution to secure the network must be versatile and reliable. Planning the implementation of defense in depth stages can be the difference between success and failure of network security.

The defense in depth strategy involves three stages: defense, detection, and prevention. With these stages implemented, attackers must first devise a way to break the network defenses. While doing this, they must attempt to reach their goal undetected and also hope the target has done a poor job of prevention. If any stage has known shortcomings, compromise will be inevitable. The components needed consist of two network firewalls using Iptables, two network intrusion detection systems using SNORT, and a vulnerability-scanning device using Nessus. Iptable sand SNORT use Syslog to log security alerts to files on their hard drives and also to a central Syslog server running Swatch to filter through the log files. All security devices will run on the Mandrake 9.1 operating system. Nessus will be located on a dedicated laptop for mobility purposes allowing vulnerability scanning from a variety of network locations.

The order of implementation will be as follows:

1. Syslog/Swatch: set up the device that all other devices will log to.
2. Iptables: set up the main defense, the Internet gateway, to protect against outside attacks, and the internal default gateways for internal segments. The various server setups and configurations in accordance to the firewalls are not discussed here but quick Mandrake 9.1 HOWTO's are located in the documentation rpm on Mandrake 9.1 CD 1
3. SNORT: implement the intrusion detection mechanism to detect attempted exploits as they traverse the network.
4. Nessus: complete the strategy by implementing prevention to find and patch vulnerabilities before the attacker discovers them.

When implementing these stages, each component should be configured and verified as working before continuing onto the next component or stage. In all, six security boxes and two additional small hubs are required for the network.

3.b IP Addressing

Having five network segments and six network components dispersed among them can create IP confusion. Further adding to possible confusion is the fact that Iptables components require multiple interfaces serving as default gateways. This confusion is obviously amplified when the implementing organization size increases and the number of network segments increases. Defining and documenting locally developed IP address conventions can aid in resolving future IP address issues.

As shown in figure 3.1 on page 2, the network consists of five separate segments. In order to maintain segregation, these networks need to use different network addressing. Once the private addressing scheme for each segment is decided upon, considering which interface connects to which network segment will help determine the IP address for that interface. In this scenario, eth 0 of each security device is reserved to be the interface that logs to the central

Syslog/Swatch device. All other interfaces on SNORT machines will not be assigned IP addresses so that they can be in sniffer mode. Also, Iptables device interfaces serve as default gateways and will have ending addresses lower than five. The following table summarizes the Ethernet interface IP address assignments according to these local conventions. This addressing scheme will also allow remote administration of all security devices via openssh in Mandrake.

Component	Interface	Network Segment	IP Address	Default Gateway
Iptables 1	Eth 0	192.168.169.0/24	192.168.169.1	
	Eth 1	192.168.168.0/24	192.168.168.1	IP Address of External router
	Eth 2	ISP Assigned Address Block		
Iptables 2	Eth 0	192.168.169.0/24	192.168.169.2	
	Eth 1	192.168.172.0/24	192.168.172.1	192.168.169.1
	Eth 2	192.168.171.0/24	192.168.171.1	
	Eth 3	192.168.170.0/24	192.168.170.1	
SNORT 1	Eth 0	192.168.169.0/24	192.168.169.50	192.168.169.2
	Eth 1	192.168.168.0/24		N/A - Sniffing Only
SNORT 2	Eth 0	192.168.171.0/24	192.168.171.50	192.168.171.1
	Eth 1	192.168.172.0/24		N/A - Sniffing Only
	Eth 2	192.168.170.0/24		N/A - Sniffing Only
Syslog	Eth 0	192.168.169.0/24	192.168.169.30	192.168.169.2

Table 3.1 Interface Configurations

4 Mandrake 9.1 and Syslog

The following Mandrake 9.1 section covers a typical installation that should be performed for each security device. Next, configuring the Syslog/Swatch device is detailed. While logging and log filtering are technically a form of detection within the defense in depth strategy, building the Syslog device is done before any other device because all other security components will log to it. With this method, each subsequent device built can be completely tested for connectivity and logging to the Syslog system.

4.a Mandrake 9.1: Configuaration

Mandrake Web Site: http://www.mandrake.com
Current Version: 9.1 **Source:** http://www.mandrake.com

4.a.i Mandrake Setup

The package selections described here represent the packages needed for all security components. On the first package selection screen, deselect all selected options, select the individual package selection option, and continue. On the next screen, select “Truly Minimal Installation” and continue. In the individual package selection window, click the toggle button to view all packages alphabetically. Select the openssh server and client packages to be used for remote administration in the future and the tcpdump package. Select the additional packages Swatch, Iptables, and SNORT for each respective security device. Syslog is a mandatory package installation that is installed automatically. When configuring the network, have Mandrake auto detect the interfaces but statically assign the IP addresses and default gateways according to Table 3.1. Do not allow Mandrake to start X interface at boot. Make a boot diskette and reboot.

4.a.ii Harden the system

Minimizing unneeded services on the security devices adds another layer to the defense by minimizing known and unknown exploitable vulnerabilities. These steps will harden the systems by removing any unnecessary services to start at boot.

1. Command to see a listing of all services running after boot:
    a. *chkconfig --list | grep :on*
2. Command to turn off desired service for run levels 1-5:
    a. *chkconfig --level 12345 <service> off*
    b. Repeat this command until only kheader, syslog, network, random, rawdevices, crond, and sshd are left with an *on* status. Of course leave the service needed that pertains to the individual security device such as Iptables.

One final note for the operating system installations is to synchronize the times on all boxes using the date command. Otherwise, log times can be a cause of confusion when diagnosing network issues.

4.a.iii Routing

Iptables1 firewall, which is the Internet gateway, must have static routes to be able to route to the internal networks. One way to accomplish this is to perform the following *route* commands

route add -net 192.168.170.0 netmask 255.255.255.0 gw 192.168.169.2 eth0
route add -net 192.168.171.0 netmask 255.255.255.0 gw 192.168.169.2 eth0
route add -net 192.168.172.0 netmask 255.255.255.0 gw 192.168.169.2 eth0

and then create a *static-routes* file in */etc/sysconfig* and manually add the route statements

eth0 net 192.168.170.0 netmask 255.255.255.0 gw 192.168.169.2
eth0 net 192.168.171.0 netmask 255.255.255.0 gw 192.168.169.2
eth0 net 192.168.172.0 netmask 255.255.255.0 gw 192.168.169.2

The same results can be achieved by simply adding the above *route* command lines to the end of */etc/rc3.d/S99local* file: [7]

Similar routing must be done on the internal Iptables2 device to achieve internet access and limited DMZ access. On both systems, IP forwarding must also be enabled but this is accomplished in the Iptables section.

4.b Syslog: Configuration

Syslog Web Site: http://www.syslog.org/
Current RPM: sysklogd-1.4.1-4mdk.i586.rpm **Source: Mandrake 9.1 CD 1**

Servers, firewalls and intrusion detection systems can generate countless alerts kept in log files. Without auditing capabilities, firewalls and intrusion detection can prove useless if those lines of defense and detection are thwarted. Logged information can aid in the aftermath forensics following a compromise and can also be used to flag compromise attempts while they are happening. Both objectives can be accomplished by using the standard Linux Syslog daemon to log information locally and externally to a dedicated device. If a system is compromised by an attacker, external logging serves two purposes in that it

serves as a backup log source and it also makes it more difficult for attackers to cover their tracks.

Syslog logs messages from services to a specified location according to a facility tag and a priority tag. A facility is the subsystem or program generating the event and the priority is the level of urgency for the event. The following table summarizes the facility names and descriptions and also describes the priorities and corresponding meanings.

Name	Facility	Priority	Meaning
kern	Kernel	*emerg*	Emergency condition, such as an imminent system crash, usually broadcast to all users
user	Regular user processes	*alert*	Condition that should be corrected immediately, such as a corrupted system database
mail	Mail system	*crit*	Critical condition, such as a hardware error
lpr	Line printer system	*err*	Ordinary error
auth	Authorization system, or programs that ask for user names and passwords (login, su, getty, ftpd, etc.)	*warning*	Warning
daemon	Other system daemons	*notice*	Condition that is not an error, but possibly should be handled in a special way
news	News subsystem	*info*	Informational message
uucp	UUCP subsystem	*debug*	Messages that are used when debugging programs
local0... local7	Reserved for site-specific use	*none*	Do not send messages from the indicated facility to the selected file. For example, specifying **.debug;mail.none* sends all messages except mail messages to the selected file.
mark	A timestamp facility that sends out a message every 20 minutes		

Table 5.1: Syslog Facilities and Priorities[3]

The priority level associated with a facility is a minimum level. For example, if a facility logs messages with a priority of warning, then all events of warning, err, crit, alert, and emerg will be syslogged to the specified location. This information Syslog uses for handling event messages is stored in the */etc/syslog.conf* file and is processed early in the boot cycle so that other starting services can log information.[4] The format of a line in *syslog.conf* to handle system events is: *<facility>.<priority><tab><location>*. For example,

**.debug /var/log/messages*

is the one line used in Company B's Syslog device's *syslog.conf* file and will log all facility messages of any priority to the */var/log/messages* file.

While Syslog comes with the *syslog.conf* file full of preconfigured facilities, levels and locations, the strategy implemented for this discussion erases these rules and establishes a maximum sensitivity for logging with all messages going to one file and delegates the sorting and alerting to Swatch. Swatch will notify the system administrator of all messages that are deemed important to the network environment while Syslog will maintain all messages for troubleshooting and forensics if necessary. On the Syslog system, all preexisting rules in the *syslog.conf* file are deleted and the above rule is added. On other systems such as Iptables, SNORT, and network servers, Company B will insert the above line while also adding the line:

**.debug @syslog.companyb.com*

Each system as a result will send all messages to the local system's messages file and also to the Syslog server. [4]

Before the logging across the network can happen, a few more steps must be performed.

1. In the */etc/sysconfig/syslog* file of the syslog host, the *–r* option must be added within the quotes in the first uncommented line to allow logging from remote hosts
2. The devices involved in syslogging should have their entries in the */etc/hosts* file of the Syslog device in the following format:
   <ip-address> <fully-qualified-domain-name> alias1 alias2 alias3 etc.
   a. Example: *192.168.169.1 iptables1.companyb.com iptables1*
   This will prevent logged files on the Syslog device from having an IP address as the source but rather the logs will have the device name of the source.
3. Syslog must be restarted to activate the changes with the following command: */etc/init.d/syslog restart* [5]

4.b.i Rotating Logs with Logrotate

Easily overlooked, log files themselves can grow to enormous sizes and fill the hard drive. Current hard drive size standards make this possibility a minor concern for a company of this size but the precautions should be in place from day one. A separate partition is created on all security devices soley for the */var/log/messages* file. This will prevent the system from crashing due to the log files filling up the primary partition. Keeping the log files rotated will also prevent the drive partition from being filled.

Logrotate is a script that by default runs weekly within the */etc/cron.daily* directory and references the */etc/logrotate.conf* file to determine if a rotation is needed. The */etc/logrotate.d/syslog* file holds the information for which Syslog files to rotate and */var/log/messages* is already present in the list. Modifying *logrotate.conf* rotate logs daily instead of weekly and also to keep daily logs for 31 days sets the desired log rotation scheme for company B.[6] This configuration should be performed on all network servers and security components.

4.c Swatch: Configuration

Swatch Web Site: http://swatch.sourceforge.net/
Current RPM: swatch-3.0.4-2mdk.noarch.rpm **Source: Mandrake 9.1 CD 1**

Logging capabilities will be implemented for all servers and security components and all is secure. Wrong! Logging all facility messages of any priority to a single file can quickly become much too overwhelming to be of use simply by manually scouring the lines for a certain event or events. Finding a particular piece of information can be like finding a needle in a haystack. Fortunately, there are open-source tools such as Swatch to aid in finding specific entries in log files and alerting the system administrator as events are triggered and logged to the central log file.

Swatch is a tool that filters log message files for particular patterns called triggers as the file(s) are updated and performs designated actions. Actions can be ignore, echo, bell (audio alert), e-mail, and execution of system commands or scripts such as notifying a pager. These triggers and actions are kept in the */etc/swatchrc* configuration file. A text file containing regular expressions,

swatchrc has the following rule syntax: */pattern/pattern/ action,action [throttle]*. Patterns are regular expressions and can contain a single text pattern or multiple patterns to cover a variety of messages with one rule. The actions can also be multiple and consist of a variety such as echoing to the console, printing alerts, sounding an audible alarm, emailing messages, executing other programs or scripts, and more. When echoing to the console, Swatch also has the ability to assign colors, bold print, and blinking characteristics to particular alerts sent to the screen to aid in determining the severity of each message. Another aid in this area is the audible alert which can be assigned one a variety of sounds. The throttle field is optional and is used to prevent a large quantity of identical alerts within a specified time period in the format *HH:MM:SS*.[8]

An example line within the script to alert on bad login attempts could appear as follows:

```
watchfor      /INVALID|REPEATED|INCOMPLETE|[Ff]ail/
        echo bold
        bell 3
        mail addresses=root@localhost, subject="Bad login attempt"
        throttle 05:00
```

When one of the four words in the first line is located in */var/log/messages*, the message is echoed to the screen in bold, a bell sound is made, an e-mail alert is sent to the specified address with the accompanying subject heading, and only one alert is generated within a five minute period.

The *swatchrc* file that comes with Swatch is a good starting point but it needs editing to alert for other messages such as those from SNORT and Iptables devices. Fine-tuning, a necessary and continual process in log filtering, must then be done to prevent a denial of service (DoS) from due to purposeful, frequently occurring messages and to suppress common alerts such as standard successful user logins and so forth.

The final step in configuring Swatch is ensuring that it is started in run level three at boot up. Since the security devices are booting into run level 3, edit the */etc/rc3.d/S99local* file and append the following command to the file: *swatch –c /etc/swatchrc –t /var/log/messages &*. This starts Swatch as a background process using the *swatchrc* configuration file to alert on the specified patterns within */var/log/messages* as the file is updated via the Syslog service.

The following three sections contain details for implementing the tools used within the defense in depth strategy. Each component subsection contains the Internet home page of the indicated security component, the latest rpm or program version as of April 2003, a product overview, and installation and configuration instructions pertaining to Company B.

5 Defense

5.a Iptables

Iptables Web Site: http://www.netfilter.org
Current RPM: iptables-1.2.7a-2mdk.i586.rpm **Source: Mandrake 9.1 CD 3**

The first step in securing a network is defending the perimeter. What exactly does Company B need to defend against? Company B mainly wants to protect their network from intrusions, denial of service, and information theft that may be directed from any type of attacker including joy riders, vandals, spies, thieves, etc. There are many ways to design a firewall implementation just as there are many ways to design a network infrastructure. The key to reliable and effective firewalls is configuration. Misconfiguring a firewall can cut off one or more segments within a network and/or expose data sensitive networks. Some causes of this are creating unnecessary rules and not creating the rules that should be mandatory for any company. Another cause may be ignorance of all aspects of the particular firewall software. Again, planning is imperative and obviously much detail is involved. So much detail that numerous books and articles have been written on the subject. With this in mind, meeting the basic firewall needs of Company B utilizing Iptables will be discussed without delving into the vast detail of a complete rule set.

Iptables, the successor of Linux's ipchains, comes standard with most Linux distributions and offers improved features and options such as a wide variety of packet filtering capabilities, packet mirroring, packet mangling, network address translation (NAT), and more. The capabilities to perform these actions are defined in rule lists that are kept in the kernel. This portion is actually separate from Iptables and is called netfilter. The program that allows configuration of the rule lists and runs the firewall is the userland application Iptables.[9]

Iptables is a progression of ipchains and still uses the initial chains, specifically INPUT, FORWARD, and OUTPUT chains, to filter packets. Other chains are PREROUTING and POSTROUTING. These chains, which dictate specifically how packets are handled through rules, are components of the default tables FILTER, NAT, and MANGLE. When a packet is inspected by netfilter, i.e. it originates from the local host or is received on an interface, netfilter qualifies the packet according to one of three routing scenarios: incoming destined for the local system, incoming destined for another host, or originating from the local system destined for another host. The packet is then processed by a set of the above tables accordingly. For instance, the packet may come in destined for the local host; traverse the MANGLE-PREROUTING chain, NAT-PREROUTING chain, MANGLE-INPUT chain, and then the FILTER-INPUT chain before being processed. Within each table, the above chains inspect the packet information for matches specific to one or multiple packet fields as specified by the rules implemented for that chain. If a matching option between the rule and packet information is found, the packet is handled according to one of many actions or targets such as accept, drop, reject, queue, dnat, or return target assigned with the rule.[10] An action can also be to jump to another chain within the same table. If no rule is matched, the commonly accepted approach to firewall implementation

is to drop all packets not matching a rule. These default rules, called policies, are usually in the INPUT, OUTPUT, FORWARD chains. This detailed process could eat up tremendous amounts of CPU cycles if many unnecessary rules were defined within these chains.

Perhaps, the most important enhancement of Iptables over ipchains is that it performs stateful inspection of packets. This is done by state matching within the packet and allows for a much more efficient firewall. Stateful inspection keeps unused ports above 1024 for return traffic closed and provides a balance between vulnerable packet filtering and lower performance application gateways. Another improvement of Iptables is the separation of the three initial chains during packet inspection. Previously in ipchains, all three chains could potentially examine packets regardless of their routing scenario. Now in Iptables, upon entry into netfilter, the packets are directed toward the one appropriate chain. In other words, a packet will only visit one of the three basic chains.[11] The packet will traverse other chains within other tables but the chains within each table may have no rules. One last interesting note on Iptables' versatility, as there are many more examples, is that it allows the capability to define custom chains within a table that can be used as targets. A chain, such as a one to handle catching all bad TCP packets, could be created to inspect all TCP packets and dropping the bad ones before continuing on to the standard tables and chains. This could save precious CPU cycles on machines suspect to extremely high traffic, e.g. an Internet gateway.

With its versatile rule list capabilities and streamlined, stateful packet filtering, Iptables provides substantial initial defense for the network.

5.b Defining the Firewall

The first step in building a firewall is planning and the planning starts with a documented security policy. Reviewing Company B's architecture is necessary to define the security policy as the resulting firewall enforces the documented policy. Company B has the following general architecture:

1. External firewall serving as an Internet gateway connected to a DMZ and internal network.
2. Internal firewall that segregates three internal sub-networks: administration, shipping and sales, and a server farm, and is connected to the external firewall.

With the company architecture reviewed, the following security policy will be enforced via firewall rule sets.

1. All employees will have limited Internet access.
2. The system administrator will have internal and external remote access through SSH
3. The DMZ will have Internet access through the external firewall.
4. Access between the administration clients and shipping and handling clients will be restricted to SSH through the system administrator.

5. IP address specific restrictions will be used to prevent unnecessary Internet traffic from the working internal departments. (e.g. ICQ, Internet Radio, etc.).
6. Universally necessary firewall rules will be implemented (e.g. deny all that is not implicitly matched, bad packet handling, logging, etc.)
7. The default policy for the INPUT, OUTPUT, and FORWARD chains will be to drop packets that are not implicitly accepted.

A quick guide for additional necessary rules can be found at http://www.spitzner.net/rules.html.

5.c Writing Firewall Rules With Iptables

The following is the standard syntax of Iptables' rules.

iptables <-t table> <command> <chain> <match> <target or jump>

One or more of these fields may not be necessary depending on the command. Most packet filtering will be done on the INPUT, OUTPUT, and FORWARD chains and not specifying a table will default to the FILTER table. Since Company B would want to allow icmp packets for troubleshooting, the rules

iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

would accomplish this. A more complex rule using a user-defined chain and demonstrating the LOG action is a rule to catch bad tcp packets that have a NEW state without the SYN bit set.

*IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG *
--log-prefix "New not syn:" [12]

As mentioned earlier, a cause of firewall exploits is misconfiguration. One rule to follow is to minimize the rule set as much as possible without hampering security. The more rules added to the rule set, the more confusing troubleshooting could be in the future, not to mention the possibility of unknowingly creating an exploit opportunity.

A crucial aspect in implementing Iptables is that rule order matters. Rules are traversed in order of their creation and once a matching rule is found, the specified action is performed. Note: a rule can be created and inserted at a specific position in the rule set. Taking the above log example, Company B would obviously want to drop that packet.

IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

If the log rule was placed after the drop rule, Company B would not receive the alert that these packets were present because the packet would be dropped and therefore never reach the log rule. This is just one case of handling odd packets. Staying updated on the latest security issues is part of network security management and as new security alerts emerge, modifications to the firewall rule set will be necessary.

5.d Logging

If the final policy of the chains is to drop, one might wonder why have any DROP rules since these will be dropped by default if not accepted. The answer lies in

logging. On each firewall, the last rule of each chain will be similar to the following:

*IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG *
--log-level DEBUG --log-prefix "iptables1 INPUT packet died: "

This will log all packets that do not match any rules with 3 alerts per minute as a limit just before they are dropped by the default DROP policy. This can be helpful in detecting a worm or Trojan from within that continuously tries to communicate outside the firewall on a port that is not allowed. Some packets that should not be logged that are dropped are broadcasts and NetBIOS to name two. Otherwise the log files will become cluttered as these packets are harmless and are frequently encountered. In order to prevent this, there must be a rule in place to drop these packets so that they do not flow to the rule above. Also, it is wise to be aware of certain packets being received such as the bad TCP packets, which could indicate an attack. These will be dropped by default but the message in the log will not be specific to that particular type of event. Therefore the rule set should contain rules for the specific packets to drop with a preceding log rule with a specific message. This will also allow for the specification in Swatch on certain Iptables logs to alert on. This brings up the last issue on logging. It is necessary to specify on the Syslog/Swatch server which logs should be alerted on and their urgency level. Following the rule example in the Swatch section, this can be accomplished. A good strategy for this is to start with a liberal alerting policy in place and tune down to an acceptable level. Care must be taken not to tune the alerts down too far as this could allow an exploit attempt to not be detected until too late.

5.e Iptables Setup

Since it is not only impractical but also impossible to be able to type in the rules each time the device reboots, which hopefully is not a common occurrence, scripting will be used to load the rules at system boot up. Iptables' scripting structure typically consists of the following sections.

1. Configuration: define script variables such as those for interfaces, network addresses (DMZ, shipping, etc) that will reference throughout the script.
2. Module loading – load modules that Iptables requires.
3. Proc configuration - ensure any special configuration requirements are met such as enabling IP forwarding with the line *echo "1" > /proc/sys/net/ipv4/ip_forward*.
4. Rules set up - Insert rule-set:
    a. Filter table
        i. Set policies - Set up all the default policies for the system chains such as the default drop policies specifically accept desired services coming in.
        ii. Create user specified chains
        iii. Create rules in user specified chains
        iv. INPUT chain
        v. FORWARD chain
        vi. OUTPUT chain

b. Nat table
    i. Set policies
    ii. Create user specified chains
    iii. Create rulesin user specified chains
    iv. PREROUTING chain
    v. POSTROUTING chain
    vi. OUTPUT chain
c. Mangle table
    i. Set policies
    ii. Create user specified chains
    iii. Create content in user specified chains
    iv. PREROUTING chain
    v. INPUT chain
    vi. FORWARD chain
    vii. OUTPUT chain
    viii. POSTROUTING chain[13]

Sample scripts that are detailed by network architecture can be downloaded and modified but should serve as a starting model. Close scrutiny must be used to ensure the implemented script covers all areas that the implemented network requires. Also, scripts will be somewhat different between the internal and external firewall as their requirements are different. For instance, the number of interfaces and the different connected networks will make a difference in variables declarations and NAT rules. Rules must also be put in place to restrict access between the two departmental workgroups.

Once the script is finalized, it must be started at boot every time. If Company B names their Iptables1 firewall script *iptables1fw*, this can be performed with the following sequence of commands:

chmod 700 /etc/rc.d/init.d/iptables1fw	-executable permissions
chown 0.0 /etc/rc.d/init.d/iptables1fw	-change ownership
chkconfig --add iptables1fw	-create symbolic link
chkconfig --level 345 iptables1fw on	-have script executer at specified levels at boot
service iptables1fw start	-start the script

Many more detailed documents, including HOWTO's and tutorials, are available through Iptables' home site. An extremely detailed and well-written resource for information pertaining to Iptables can be located in Oskar Andreasson's "Iptables Tutorial 1.1.17." Included in this are multiple Iptables firewall scripts, troubleshooting, along with command and option explanations.

6 Detection

6.a SNORT

SNORT Web Site: http://www.snort.org/
Current RPM: snort-1.9.1-1mdk.i586.rpm **Source: Mandrake 9.1 CD 3**

Firewalls are the main line of defense and hopefully this will be the stopping point for most attacks. Attacks are going to happen though, many different methods

will be used, and many will not be stopped by firewall rules. Attacks normally try to exploit some service or application vulnerability to gain access to a system to use as a launching point in the attack chain, to steal information, or to simply cause havoc on the system itself. For those attempts that get past the firewall or for those originating from within the firewalled network, intrusion detection systems can provide the means to detect and stop attacks or malicious activity in real time.

Intruders can gain access via a number of different methods. Some of these include software flaws, misconfigured systems (e.g. unneeded vulnerable services running), and password cracking. These methods of attack can fall within one of three categories: reconnaissance, probing to become familiar with the target system and possible exploitable services and programs used; exploits, intruders using bugs or hidden features to gain access to and/or control of a system; and denial of service attacks (DoS), slow down or crash the system through repetitive packet requests.[14] Particular vulnerability exploits can include: CGI script, web server attacks, web browser attacks, Sendmail attacks, IP spoofing, DNS attacks.[15] DoS attacks can include the Ping of Death, SYN flooding, Land attack and more.[16]

Attackers have many different options; the next question is how to stop these attacks that will get through the firewall or originate from within. The attacks will generally have byte patterns within the packets and sometimes over a number of packets. IDS signatures, which contain these patterns, are referenced to compare packet headers and payload for certain patterns. If a match is found, an alert can be sent to the administrators to notify of a possible attack and hopefully stop the attack or by a designed auto-response action. The key to efficient detection is minimizing false positives (alerts that are generated due to a general signature match but in reality are legitimate traffic) while not missing legitimate attacks or minimizing false negatives due to signatures being too general.[17] These are the packets that Company B's administrator should be alerted on; the packets that are legitimate attacks and are not false positives.

SNORT, an open-source intrusion detection program packaged with Mandrake, has extreme versatility and is widely supported by the open-source community. Due to this support, it has a very large signature base to detect possible reconnaissance, exploit, and DoS attempts. Company B will have two SNORT devices, SNORT1 and SNORT2, and both will log to the Syslog server through their eth0 promiscuous interfaces. All other interfaces on the SNORT boxes will non-addressable interfaces in promiscuous mode to sniff traffic only.

6.b Signatures and Rules

As mentioned above, signatures are patterns within network traffic. Rules contain the signatures SNORT uses to compare packet data for matches. Some of these signatures could be patterns within the packet headers (e.g. SYN and FIN flags both set), or DNS buffer overflow attempt within the payload of the packet, or DoS attempts through numerous command executions, just to name a few[18] and an identified signature in network traffic can help stop a malicious attack. SNORT comes with numerous signatures already. If abnormal network traffic occurs that

Company B wants to be alerted for in the future. For this, SNORT also provides the ability to write custom signatures to match for the pattern within future traffic.

Rules, similar to firewall rules, check the traffic for a match. IDS rules have two sections; the rule header and the rule options. These are kept in multiple files in the */etc/snort/rules* directory. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. The rule option section, located within parenthesis, contains the alert message that specifies which parts of the packet should be inspected. This information will determine if the specified action should be taken.[19] Standard syntax is as follows:

<rule action><protocol(s)><src IP(s)><src port(s)> -><><dest IP(s)><dest port(s)> \

(<keyword>:<args>;)

There may be multiple protocols, source and destination IP's, ports, keywords, and arguments. This complexity provides the ability to be very specific in the traffic that is detected. Caveats exist though. Rules too general will cause false positives, which can actually conceal legitimate signature alerts within noisy, unnecessary alerts. Rules too specific will produce false negatives, which may not be detected until a compromise has already occurred. SNORT's web site provides detailed information on the options and arguments for each field. Also Security Focus' website provides a series of articles on IDS signatures, practice and theory.

Obviously Company B's system administrator is not going to write all the required meticulous rules solely for their network structure from scratch. This is an ongoing task as future signatures are created as new vulnerabilities and attack methods are discovered. Community support is an enormous strength of open-source. With this, as new required signatures are published, they can be downloaded from a number of sites to help further secure networks. SNORT's website provides a searchable database of the latest signatures. Subscribing to security mailing lists and being familiar with the CVE and CERT websites can greatly aid in the signature updating process. Again, staying informed is the key to success.

6.c SNORT Setup

Once the SNORT rpm is installed, the next step in setting up SNORT is reviewing and editing the */etc/snort/snort.conf* file to be configured for the networks being monitored. The *snort.conf* file contains information such as variables that are used in the individual rules, preprocessor directives, and output plug-ins. The file provides adequate instruction as to the purpose and methods of defining all items within each section. The first section defines the network segment being monitored including the addresses of subnets and more specifically the IP addresses of particular servers being monitored on those networks. The preprocessor directives serve purposes such as detecting scans, packet fragmentation and reassembly, and other exploits while output plug-ins allow specification of output options. The last section is also well documented

within the file and is reserved for user-defined rule sets. Most directives come preconfigured to be included at SNORT startup but can be commented out.

Fortunately for Company B, SNORT is almost ready to run from unpacking the package and installing it. This very well may not be the case for more complex network environments as SNORT can be considered the Swiss army knife of open-source intrusion detection with its vast number of configuration options.

6.d Starting SNORT

SNORT can be started in three different modes: sniffer mode, packet logger mode, and network intrusion detection mode. The first, sniffer mode, reads packets as received and outputs to the console. Packet logger mode logs all received packets to the specified directory. Network intrusion detection mode is the mode that Company B will run in. With this mode, the *snort.conf* specific to each interface file will be used to configure SNORT to match signatures to packet content and send alerts to Syslog, which in turn will send those alerts to the Syslog server.

SNORT needs configured to run on multiple interfaces, as each interface will monitor a separate network segment. This will be accomplished by running multiple instances of SNORT. This requires that multiple *snort.conf* files be used so that each SNORT interface instance can have their own respective configurations such as network variables. Therefore the SNORT1 device will have two *snort.conf* files and the SNORT2 device will have three. The following table shows the variable assignments made for each SNORT device and the default commented lines in *snort.conf* to uncomment. The various *snort.conf* files can be created with the following command syntax:

cp /etc/snort/snort.conf <rules file name per SNORT device interface>

This file is created for each interface in each SNORT system according to Table 6.1 below. This table also reflects the modified variable assigned values and the preprocessor directives to uncomment in the files.

Rules Files Variables				
Rules File Name	*Device*	*Monitoring Interface*	*Variable*	*Value*
snort1eth0.conf	SNORT1	eth0	RULES_PATH	/etc/snort/rules
snort1eth1.conf	SNORT1	eth1	DNS_SERVERS	192.168.168.5
			SMTP_SERVERS	192.168.168.4
			HTTP_SERVERS	192.168.168.3
			RULES_PATH	/etc/snort/rules
snort2eth0.conf	SNORT2	eth0	DNS_SERVERS	192.168.171.5
			SMTP_SERVERS	192.168.171.4
			RULES_PATH	/etc/snort/rules
snort2eth1.conf	SNORT2	eth1	RULES_PATH	/etc/snort/rules
snort2eth2.conf	SNORT3	eth2	RULES_PATH	/etc/snort/rules
Preprocessor Directives				
Directives to Uncomment		*Reason*		*Value*
preprocessor portscan2:		More granular detection of portscans		scanners_max 3200, targets_max 5000, target_limit 5, port_limit 20, timeout 60
preprocessor portscan-ignorehosts:		Useful for preventing noisy alerts from DNS Servers		$DNS_SERVERS
output alert_syslog:		Enable syslog abilities		LOG_AUTH LOG_ALERT
All commented include lines in the rule section		Start with liberal detection and tune down to acceptable level according to environment		

Table 6.1: SNORT Rule Files and Modifications

With the configuration files complete, the following commands should be placed at the end in the */etc/init.d/snortd* on SNORT1 in order to start up SNORT at boot:
snort -i eth0 -s -b -c /etc/snort/snort1eth0.conf &
snort -i eth1 -s -b -c /etc/snort/snort1eth1.conf &

These commands are placed in the same location on SNORT2:
snort -i eth0 -s -b -c /etc/snort/snort2eth0.conf &
snort -i eth1 -s -b -c /etc/snort/snort2eth1.conf &
snort -i eth2 -s -b -c /etc/snort/snort2eth2.conf &

SNORT only requires a reboot and it will be running.

6.e Logging

The above commands use the –s option to configure snort to use the Syslog service for logging alerts. The alerts go to the local hard drive in */var/log/messages* and are then sent to the Syslog server via the modifications made in the *syslog.conf* file. As with all other logging, the policy begins with a very liberal status; logging almost everything. The process needed now is fine-tuning the logging level in Swatch via the *swatchrc* file. Rules must be added to this file to alert in proper ways for different alerts. Such a rule might be

watchfor /Denial of Service/
echo bold
bell 1
mail addresses=sysadmin@companyb.com, subject="Denial of Service Detected"

As some alerts are investigated, some of these added rules could be eliminated if not necessary. This can be a tedious and time consuming task but once at an acceptable level, the pay off will be seen once the first exploit is stopped.

7 Prevention

7.a NESSUS

Nessus Web Site: http://www.nessus.org/ **Source:**
Current Version: 2.0.3 http://www.nessus.org/nessus_2_0.html

The last phase of Company B's defense in depth strategy is prevention. Prevention is not only preventing what is known but also what is unknown. As new vulnerabilities are discovered, there will surely be exploit attempts to follow. The primary ways to defend against these attacks are staying informed of the latest vulnerabilities, keeping systems updated and patched, and utilizing tools such as Nessus Security Scanner to test the targeted systems. Nessus is a vulnerability scanner that identifies vulnerabilities in network computers and determines if and where a system can be exploited and/or threatened. With tools such as this, known exploits can be identified and the necessary remedies applied. Company B's system administrator will be the only person authorized to use this tool.

Nessus, the open-source solution to vulnerability scanning, provides not only complex, granular scanning capabilities but also a detailed reporting utility describing specific vulnerabilities, CVE advisory references, and suggestions to

remedy the vulnerability. These reports can be saved so that results future scans can be compared to ensure the proper solutions were applied. The reason for single administrator authorization is that Nessus' scanning capabilities include testing for DoS and other malicious vulnerabilities that can take down systems. In the wrong hands this can also be a tool used against Company B.

Some of Nessus' features include plug-in architecture, multiple port service testing, unlimited or targeted host testing, safe testing mode, and full test mode. Plug-in architecture allows Nessus to be updated as new vulnerabilities are exposed. By going to Nessus' plug-in page, http://cgi.nessus.org/plugins, Company B can stay updated with the latest plug-ins to detect new exploit opportunities. (As a note, Nessus provides an automatic update utility that connects to the Nessus web site and downloads new plug-ins via the Linux lynx utility. However this connection is not secure so it is recommended to go to the site and manually download the new plug-ins.) Users can also create custom plug-ins. Multiple port service testing is unique in that it doesn't test for a specific service running on a specific port. In other words if there are multiple web server services running, one on port 80 and another on some other port, both ports will be detected as running that service. Therefore, any vulnerabilities within that service will be detected. Nessus has the option of testing just the local host, a single targeted remote host, or a whole range of addresses to detect multiple hosts. Safe testing specifies testing for vulnerabilities that will not bring the system(s) down such as useless services running. Full mode testing is considered dangerous and can exploit a vulnerability to a point of crashing a device.[20]

7.b Installation

The Nessus machine will be a laptop that has sole purpose to perform vulnerability scanning. Notice the dotted lines in Figure 3.1 flowing from the Nessus machine to various locations. This indicates that the connections are not permanent. Each time the administrator performs a vulnerability scan he will manually connect to the network switch, hub, or router. When not doing this job, the machine will be disconnected from the network. The standard hardening procedures described earlier will be followed for the Nessus device. However, since a graphical environment will be installed, services pertaining to the KDE interface such as xfs cannot be turned off.

Nessus requires the four source code files to compiled and installed. At the time of this document, no rpm package was available that accomplished this task in one installation. Nessus does provide a shell script that handles the installation processes. When initially running the script, there will be compile errors due to a conflict between the versions of GTK from the GIMP packages that are installed in Mandrake 9.1. The Mandrake 9.1/KDE 3.1 installation contains GTK 2.0 packages while Nessus tries to compile using GTK 1.2 packages. There are a number of ways to work around this issue but these are not detailed here.

Once installed, the script informs the user of four necessary steps to finalize the installation. Most important of these is that it requires a certificate to be created for the user along with a username and password. This is significantly important

to prevent malicious doers from utilizing this tool even if the laptop is stolen or compromised. Finally the Nessus daemon is started and then the *nessus* command is issued to start the program. Once the username and password are entered, scanning can begin.

7.c Reporting

Nessus' graphical interface is intuitive and also provides pop-up tips to guide the user. To test the installation, it would be wise to test on a non-critical system such as the Syslog server. Note that testing notices should be given to the appropriate personnel. Recall the Syslog server IP address is 192.168.169.30. All testing will be enabled including dangerous tests.

The initial reporting interface allows a quick overview of the scanned system(s) vulnerabilities. The scan against the Syslog server only showed two security alerts, both assessed as low risk factors. Selecting one of these in the port window, a security warning and security note are available to view. The selected port, general (icmp), and corresponding security warning display the vulnerability description, solution, risk factor, CVE's (Common Vulnerabilities and Exposures) related document information, and the Nessus plug-in ID that detected the vulnerability. The scan also detected the operating system as well.

The scan is then saved for future assessment against future scans. When saving the report, different formats are possible, the most user friendly probably being the HTML format. There is an HTML version with various charting and various text formats as well. These HTML formats provide convenient hyperlinks to the CVE related document and Nessus ID. As mentioned earlier, staying informed and updated is the strongest defense. Using Nessus quickly highlights the areas of each system that need attention while completing the three stages of defense in depth.

8 Additional Concerns and Conclusion

The above systems and configurations were built to model Company B in a test environment in order to verify all information presented. There are a number of concerns that still need attention within Company B's network. One of these includes defining security policies. Policies need to be documented along with making these policies readily available to personnel. Host hardening of all workstations will also be necessary. Finally, implementation of host based firewalls and IDS should be explored.

It has been shown that open-source does have the versatile and reliable tools necessary and available with more than adequate support to implement network security. Much open-source community effort is the reason for this to be possible. Company B started with defense, which flowed into detection, which in turn flowed into prevention. This prevention, upon realizing vulnerabilities, leads right back to the first and second stages through being updated and informed and applying fixes. So, it is clear the while it initially appears that the defense in depth strategy is composed of three distinct stages, these stages are actually intertwined to form a solid foundation for a secure network infrastructure.

9 References

1 Fisher, Dennis. “Security Fueling Open-Source Adoption.” EWeek. 29 Oct. 2003. URL: http://www.eweek.com/article2/0,3959,655054,00.asp. (16 Apr. 2003).

2 Fisher, Dennis. “Open-Source Comes Under Fire.” EWeek. 22 Nov. 2003. URL: http://www.eweek.com/article2/0,3959,720477,00.asp. (16 Apr. 2003).

3 Garfinkel, Simson., and Gene Spafford. “Auditing and Logging.” Practical UNIX & Internet Security. O’Reilly, 1996. URL: http://www.plys.dk/docs/bookshelf/puis/ch10_05.htm. (16 Apr. 2003).

4 “syslog: The UNIX System Logger.” CPL Systems. URL: http://www.cplsystems.net/syslog.htm. (16 Apr. 2003).

5 Harrison, Peter. “Syslog.” Linux Home Networking. URL:http://www.siliconvalleyccie.com/logging.htm#_Toc36811062. (16 Apr. 2003).

6 Harrison, Peter. “Logrotate.” Linux Home Networking. URL:http://www.siliconvalleyccie.com/logging.htm - _Toc36811063. (16 Apr. 2003).

7 Harrison, Peter. “How To Configure Two Gateways.” Linux Home Networking. URL:http://www.siliconvalleyccie.com/network-linux.htm#_Toc33893560. (16 Apr. 2003).

8 Spitner, Lance. “Watching Your Logs.” Lance’s Security Papers. 19 July 2000. URL:http://www.spitzner.net/swatch.html (16 Apr. 2003).

9 netfilter: firewalling, NAT and packet filtering for Linux 2.4. URL:http://www.netfilter.org/. (16 Apr. 2003).

10 Andreasson, Oskar, “Iptables Tutorial 1.1.17.” Frozen Tux. 6 Apr. 2003. URL:http://iptables-tutorial.frozentux.net/iptables-tutorial.html - TARGETS. (16 Apr. 2003).

11 Red Hat Linux 8.0: The Official Red Hat Linux Reference Guide. 2002. URL:http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/pdf/rhl-rg-en-80.pdf. 167-169. (16 Apr. 2003).

12 Andreasson, Oskar, “Iptables Tutorial 1.1.17.” Frozen Tux. 6 Apr. 2003. URL:http://iptables-tutorial.frozentux.net/iptables-tutorial.html#NEWNOTSYN. (16 Apr. 2003).

13 Andreasson, Oskar, “Iptables Tutorial 1.1.17.” Frozen Tux. 6 Apr. 2003. URL:http://iptables-tutorial.frozentux.net/iptables-tutorial.html - THESTRUCTURE. (16 Apr. 2003).

14 Graham, Robert. “FAQ: Network Intrusion Detection Systems"?.” RobertGraham.com. 21 Mar. 2000. URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html - 1.7. (16 Apr. 2003).

15 Graham, Robert. “FAQ: Network Intrusion Detection Systems"?.” RobertGraham.com. 21 Mar. 2000. URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html - 1.8. (16 Apr. 2003).

16 Graham, Robert. “FAQ: Network Intrusion Detection Systems"?.” RobertGraham.com. 21 Mar. 2000. URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html - 1.9. (16 Apr. 2003).

17 “Description --> Intrusion Detection System.” LINUXsecure: Issues on Linux and Security. 9 Dec. 2002. URL: http://www.linuxsecure.de/index.php?action=24 - sig. (16 Apr. 2003).

18 Frederick, Karen Kent. “Network Intrusion Detection Signatures, Part One.” Security Focus. 19 Dec. 2001. URL: http://www.securityfocus.com/infocus/1524. (16 Apr. 2003).

19 Roesch, Martin, and Chris Green. “Snort Users Manual Snort Release: 2.0.0.” Snort™. 2003. URL: http://www.snort.org/docs/writing_rules/chap2.html - tth_chAp2. (16 Apr. 2003).

20 Nessus. 2 Apr. 2003. URL: http://www.nessus.org/doc/datasheet.pdf. (16 Apr. 2003).