HOWTO: Debian Etch Security Appliance Firewall

Firewall how-to: port level security, intrusion detection, web caching, web content filtering, logging
Suitable for: Enterprise, Corporate, SOHO, Small Business, public access networks, schools, libraries, etc.

Version (by date): June 26, 2008
Written By: Daryl Caudill, aka drokmed
Linux distribution: Debian Etch 4.0 (STABLE)
Prerequisites: Experience with GNU/Linux, and a strong understanding of TCP/IP is required
Difficulty level: Beginners will find this hard and challenging
Maintained at: forums.debian.net

Summary:

This how-to is written to help you set up a multi-function GNU/Linux firewall. It covers advanced techniques, and is far from trivial. This document is more of a tutorial than a quick how-to. It is intentionally long, to help educate the beginner.

Features and Capabilities:

Firewall:
- Network Address Translation (NAT)
- Per user/port/interface/direction control and logging
- Layer 4 software control and logging (same thing, just worded differently)
- Port Forwarding (DNAT)
- Port Knocking
- Traffic Shaping
- Rate Limiting
- Multi-attempt attack suppression
- Active attack monitoring and notification
- Automated response to known attack types
- Web traffic performance caching proxy server
- Dynamic real-time web content access filtering
- Per user/group web access control per site/type/content
-

Featured Packages:

Debian Etch (STABLE) GNU/Linux
- shorewall - robust firewall configuration tool
- dnsmasq - simple DNS and DHCP server
- squid - robust web caching server
- dansguardian - robust web content filtering server
- webmin - remote web-based graphical management interface
- psad - port scan attack detection
- fwsnort - iptables-based attack detection and active response
- nmap - robust text-based port scanner
- iftop - real-time network interface traffic monitor
- ntop - web-based network traffic sampling and reporting
- and many others, like ntp, openssh-server, ddclient, etc.

About:

Services provided include: firewall, network address translation (NAT), web caching proxy, web content filtering, DNS server, DHCP server, web-based graphical management interface, lots and lots of utilities, and tips on how to manage and troubleshoot it. This how-to gives the foundation for a firewall that can be built upon with other more advanced services, such as an MTA redirector like postfix, a vpn service, etc. Advanced users should feel free to add to this document, and share it with everyone. I'd appreciate a copy.

Building the firewall is the easy part. The hard part is knowing how to use it, and troubleshoot the services. Even though a graphical management system is provided, you still need to have a good understanding of the theory behind each service, and should utilize the mentioned reference material in each section.

I intentionally wrote this for the “Beginner linux user”, because this is almost always the first server every linux fan tries to build. The scenario is always the same: You have a wimpy cheap firewall device you bought at the computer store, and it does NAT, DNS and DHCP well enough (usually), but lacks in many other capabilities. I see questions posted in the forums all the time: How do I set up web caching, or content filtering, or see who is attacking my firewall, or what sites users are accessing? The answer is always: build your own firewall. For most beginning linux fans, who don't really understand things like how to configure IP routing in linux, building a firewall turns out to be much harder than they realize. This document should help them get started.

I have intentionally over-simplified this as much as possible, to help the new person. I am employing the KISS principle: KEEP IT SIMPLE STUPID, or what I like to call KEEP IT STUPIDly SIMPLE. Experienced linux users will still find this useful. They can just scan through the details and read just the commands at each step. Any beginner skipping the details, then later asking for help, risk getting yelled at, and told to READ the details!

Some people will argue over which packages I chose to install. Whatever. Honestly, it doesn't matter. For each category, there are many packages to choose from. Over the years, I have settled on these packages for my firewalls. As you get better at building firewalls, you will try other packages, and I do encourage that. The beauty of linux is there are many ways to accomplish the same thing! This how-to just gets you started.

Firewall packages explained:

What are we installing on this firewall? Here are the major components:

Operating System: Debian GNU/Linux

Homepage: http://www.debian.org/
Forums: http://forums.debian.net/
Wiki: http://wiki.debian.org/
Mailing Lists: http://www.debian.org/MailingLists/

Debian GNU/Linux is considered by many VETERAN linux system administrators as the most robust, reliable, stable, solid, seasoned, business class, production caliber, truly open source free distribution available. Many businesses run their production servers on Debian. Debian does assume you know your way around a linux system. If you are new to linux, and “need your hand held”, there are beginner linux distributions out there you can learn on first. Once you have become proficient, and no longer consider yourself a “noob”, you would be wise to try Debian, especially if you intend to run linux on your business servers. You'll be glad you did.

Why choose Debian? See for yourself: http://wiki.debian.org/WhyDebian

This particular how-to is written for Debian Etch 4.0, the current STABLE version. It can easily be modified to work with any linux distribution by the more advanced linux user. If you do re-write this for another distribution/version, please share it with us. I'd appreciate a copy.

As for Debian Etch (current stable branch), you should ALWAYS build a firewall using the STABLE branch of Debian, period! Debian STABLE has frozen the version of each package installed, so no bugs will sneak in from newer versions. As new bugs are discovered in the older frozen package, the Debian maintainers do an

excellent job of quickly providing security patches to fix them. You should never have to worry if your firewall will be hacked because of bugs in “too new” packages.

Firewall configuration software: Shorewall

Homepage: http://www.shorewall.net/
Features: http://www.shorewall.net/shorewall_features.htm
Wiki: http://wiki.shorewall.net/
Mailing Lists: http://dir.gmane.org/gmane.comp.security.shorewall

Shorewall is considered by many experts as the most advanced open source router/firewall/gateway configuration tool available, for configuring netfilter/iptables. Netfilter is the packet filtering framework inside the Linux 2.4.x and 2.6.x kernel series.

Netfilter Homepage: http://www.netfilter.org/
Netfilter Wiki: http://en.wikipedia.org/wiki/Netfilter
Iptables Wiki: http://en.wikipedia.org/wiki/Iptables

There are many packages available for configuring the firewall functionality. I have chosen to install and configure the shorewall package, not because it is simple (it is quite advanced), but because it is very easy to install, configure, and manage via the command line (and via webmin if you want a GUI). It is also designed to easily implement many advanced features, only some of which we will be using. You can easily implement the many other advanced features later if you choose.

Some people will argue it is better to use other methods and/or packages, saying you learn more, such as writing shell scripts to modify IPTABLES directly. This is certainly true, but requires a lot more time, learning, troubleshooting, and or course, experience.

Graphical management interface: Webmin

Homepage: http://www.webmin.com/
Forums: http://sourceforge.net/forum/forum.php?forum_id=600155
Mailing Lists: http://www.webmin.com/mailing.html

Webmin is a very popular web-based graphical management system for linux and unix systems. It has a huge following, and is actively supported on pretty much every linux and unix version known. In fact, there are many companies that actively contribute money and resources to ensure Webmin stays up-to-date:

Supporters: http://www.webmin.com/partners.html

There are many people that do not like webmin for many reasons. It has had a long history of security problems, bugs that mess up your system, etc. If you explore it, and click on the wrong thing, you can sometimes break your server! However, for beginning linux users, it is an incredibly useful educational tool to show you all of the advanced options that exist on a linux server, including many you didn't even know about. Everyone should try it at least once.

Beginning linux users always try to find the easiest way to manage things that are new. That is to be expected. It shouldn't surprise you that advanced linux users ALSO like to keep management simple. I always install webmin on all of my firewalls and servers. I don't use it very often, but it is there if I need it.

For those of you who are security conscience and wondering, webmin will NOT be accessible from the Internet. If you need that, set up an SSH tunnel or VPN!

DNS/DHCP Server: Dnsmasq

Homepage: http://www.thekelleys.org.uk/dnsmasq/doc.html
Mailing Lists: http://lists.thekelleys.org.uk/mailman/listinfo/dnsmasq-discuss

Dnsmasq is a very popular DNS and DHCP server for home and small business use. It is incredibly simple to

install and configure. For our purposes, we just install it, “turn it on”, and enable a few features. Another reason I chose Dnsmasq is because it also includes a nice DHCP server built-in, that is very easy to configure.

Web cache/Proxy Server: Squid

Homepage: http://www.squid-cache.org/
Wiki: http://wiki.squid-cache.org/
Mailing Lists: http://www.squid-cache.org/Support/mailing-lists.dyn

Squid is a very popular advanced web caching proxy server. It improves performance and saves bandwidth by caching frequently-used content. It supports robust configurations, and easily integrates with third party tools for blocking known bad sites, performing dynamic content filtering (such as Dansguardian), etc.

Squid can be configured to support users in a variety of ways, including transparent mode (users don't even know it's there), and per-user authentication, which provides per-user robust control and logging, and can even be configured to authenticate logins against your NT Domain controller, or even an LDAP server.

For many years, squid has been the standard. It is quite advanced, yet is simple to set up. It has many advanced features, and is also easily managed via webmin.

Web Content Filtering: DansGuardian

Homepage: http://dansguardian.org/
Mailing Lists: http://dansguardian.org/?page=mailinglist
Wiki: http://en.wikipedia.org/wiki/DansGuardian

DansGuardian is by far the most advanced web content filtering system available in open source. Unlike other content filters, DansGuardian examines the *content* in real time (not just URL addresses), and actually reads the web page words before forwarding it to the user. If DansGuardian detects any naughty or otherwise unwanted words in the page, it will block it! DansGuardian can filter using a robust combination of multiple methods, including: URL and domain filtering, content phrase filtering, PICS filtering, MIME filtering, file extension filtering, and POST limiting (limit uploads). DansGuardian can even block advertisements!

DansGuardian offers a HUGE list of features, way too many to list here.

DansGuardian Features: http://dansguardian.org/?page=introduction

In fact, when this firewall is done, everyone in your organization will refer to it as the “DansGuardian Firewall”, because of the block pages they get when trying to access any inappropriate sites!

For many years, DansGuardian has been the standard. There are easier packages to setup, but they are not nearly as robust and flexible. DansGuardian is also partially manageable from webmin (old module unfortunately).

Intrusion Detection, Prevention and Response: psad & fwsnort

Homepage: http://www.cipherdyne.org/
psad homepage: http://cipherdyne.org/psad/
fwsnort homepgae: http://www.cipherdyne.org/fwsnort/

To detect attacks, and automate responses to them, we will be installing psad and fwsnort, two very popular packages, both by Michael Rash, who also recently wrote a book called 'Linux Firewalls' which covers psad, fwsnort and other great utilities in detail. I highly recommend this book.

Port Scan Attack Detector (psad) monitors the firewall logs in real time, to detect port scans and other suspect traffic. Depending on the configurable danger threshold, you can have it generate email alerts, and even block offending IP addresses automatically. Psad incorporates many of the signatures from snort, and can detect attack scans, DDoS attacks, and other known threats such as nmap advanced port scans. Psad can also passively fingerprint the remote operating system from which the scan originates. Psad is an excellent tool used to detect and thwart system level attacks.

Fwsnort, which works hand-in-hand with psad, takes it a step further, and protects your application servers, such as your web server, email server, database server, etc. Unlike psad, which only looks at the layer three and four information such as IP addresses, ports, packet types, etc., fwsnort actually examines the content of the payload as well, meaning it looks at the data inside the packet. This can be used to detect malicious worms and viruses, and any other types of application layer exploits. Fwsnort takes many snort signatures, and implements them directly into netfilter rules, to detect known application layer exploits.

For those of you not familiar with snort (I have mentioned it several times), snort is a well known IDS that is quite robust, and is considered the best in the open source world. We do not run it because unlike psad and fwsnort, snort usually requires a dedicated machine that is very cpu intensive. It's overkill for our purposes, especially since psad and fwsnort cover pretty much everything we need. For more information about snort, check it out:

Snort Homepage: http://www.snort.org/

Miscellaneous utilities:

No firewall would be complete if it didn't include lots of useful utilities to check things out. There are many utilities (too many), and I try to cover some of the more popular. We could add to this section a zillion more options.

If you do add more utilities, please share with me so I can update this document.

Utilities added:

- nmap (text-based security scanner, scans systems for open ports)
- iftop (text-based real-time network traffic monitor)
- ntop (web-based traffic sampling and reporting of network traffic)
- logrep (web-based syslog viewer) vs php-syslog-ng vs logtool vs SMT vs phplogcon (freshmeat.net)
- mutt (text-based email client, for checking mail from ssh session)
- wireshark (text-based packet sniffer)

Prerequisites and assumptions:

Even though this how-to is written for the beginner level, you still must have some experience with Linux, TCP/IP and firewalls. If you are very new to any of these, I doubt you will be able to successfully implement this firewall in a production environment.

If you want to be successful at building this firewall for use in a production environment, you must have:

- A strong understanding of TCP/IP. This is required. You must understand the first four layers of the OSI model: physical, data link, network, and *especially* transport. Almost everything in this guide focuses on layer 4 ports.

- A strong understanding of the advanced features that modern firewalls provide. You need to understand the inner workings of NAT, port forwarding, port redirection, port knocking, etc.

- A sufficient working knowledge of linux, using the command line. You need to know your way around the linux command prompt, including editing files, restarting services, using ssh, etc.

- You need to know how to install Debian linux. I do not cover the actual Debian install. There are plenty of other how-to's out there that cover this.

- Experience with the Debian distribution of linux. You should already have experience building Debian linux servers in text mode. If you know linux, but are coming from a different linux distribution, I highly recommend the book “The Debian System” by Martin F. Krafft, which is written for people experienced with linux, but new to Debian... a must have for any administrator.

- A good understanding of ALL the services being installed on this firewall. For each service, I identify the main configuration files, as well as the documentation that is installed with the package. READ THEM! Familiarize yourself with them. Make sure you understand how each package works.

Other Useful Links:

Intended Audience:

- Linux enthusiasts, who really want to learn this stuff. You may get a working firewall, but unless you read and learn all the recommended documentation, the firewall will be of little use to you. Use in a production environment at your own risk MUAHAHAHAHAHA!!!!! Don't get fired! :)
- Systems Administrators, who would like to get a clean system up and running quickly.
- Systems Integrators, who want to build and support GNU/Linux firewalls for their clients.

Network Setup Requirements:

Your network must have:

- ✔ A direct internet connection. Can be a static or dynamic IP address provided by your ISP, doesn't really matter which.
- ✔ A relatively new pc with moderate speed and lots of RAM, to install the server software. It doesn't have to be new. The cheapest box from your local computer store costing under $500us is more than sufficient. Since we are installing all the services on this one box, the more RAM the better. The basic firewall should have 256MB RAM. If you intend to install squid/dansguardian, a minimum of 512MB RAM is needed to avoid hitting the swap file. This should suffice for a small office of less than 100 users. More users means more RAM… you'll have to figure RAM usage yourself.
- ✔ A management workstation, used to remote test and remote configure the server. Can be Windows or Linux based (dual-boot very useful). I use my laptop, running Debian Lenny, of course.
- ✔ Additional workstations to test connectivity are extremely useful, but not required for testing.
- ✔ An external (on other side of Internet) linux workstation, to perform remote testing. I have a linux firewall at work and at home, to test to/from each.
- ✔ An optional registered internet domain. You must subscribe to one of the providers (ex: dyndns), and register your domain name address on their site. Don't be cheap, get your own personal domain. If you intend to also run an email server, you need a custom account that supports MX records (email server identification). If your ISP provides a dynamic IP address, we will install a service on the firewall to automatically update your domain IP address information whenever it changes.

Hardware requirements:

It may surprise you that this firewall can be installed on a older slower machine. It does not require much in the way of resources (squid needs the most). An old Pentium PC with 256MB RAM will do nicely. Obviously, a faster machine with more RAM will be better, but is not necessary for small LAN's.

You also need a 2nd network interface card (NIC).

NOTE: Even though I don't cover adding a third NIC and setting up a DMZ, if you know what that is, you shouldn't have any trouble adding one. Maybe I'll add an appendix that covers this.

What NOT to install on a firewall:

This machine is a firewall. It is placed DIRECTLY on the Internet, with nothing in between. It is not a server. Your firewall should ONLY have services that are best located on the firewall. If you are thinking of adding services to this firewall, you must ask yourself: Is the service firewall related?

Things never to install on your firewall:
- Web Server
- File Server
- Email Server
- Print Server
- User shell accounts
- any other non-firewall type of service

Every service running on a server opens up that server to attack. If you run non-firewall services on your firewall, each and every one opens another way for hackers to penetrate your server, take over your network, steal and then destroy all of your personal information, set up horrible services of their own that are now hosted by YOU (porn, denial of service attacks, stolen software archives, etc).

I have personally had servers hacked:
- I once had a poorly configured ftp site hacked, and became a host of HUGE amounts of stolen software and porn, filling my hard disk over night.
- I used to install VNC on my firewall so I could remote control it, only to find somebody hacked that, installed software of their own, and used it as their personal remote workstation to visit nasty websites, and actually host an IRC server on my machine.
- My logs show my firewall is port scanned a lot, sometimes many times a day.

I know, you are thinking, nahhhhh nobody cares about me, my server isn't interesting enough to be hacked. I used to think that way too. It does happen. It's embarrassing, it's infuriating, it's frustrating, it's easy to avoid. Don't be a schmuck... build a separate server for non-firewall services. Some of you will decide to do it anyways. Go ahead! What do I know?

Firewall Theory:

Due to popular demand, I've decided to clarify some of the specifics we intend to implement on this firewall. This will help immensely when both configuring AND troubleshooting it. You must understand this section.

Personal Firewall vs. Stand-alone Firewall:

This is a common misunderstanding. There is a huge difference between the two.

A basic firewall (also known as personal firewall), is usually something you install on your workstation, and blocks traffic in just ONE direction, and that is FROM “the world” TO your workstation (some can also block from the workstation to the world). It is designed for a workstation that has just one NIC. That is all you need on your workstation. However, personal firewalls were never meant to be used as a stand-alone firewall. There is a huge difference.

A stand-alone firewall is much more advanced. It will be configured to INITIATE traffic in SIX different directions. I'll explain:

INITIATING traffic one-way explained:

This is the **CORRECT** way to represent traffic INITIATED through a stand-alone firewall:

WAN → Firewall
WAN → LAN
Firewall → WAN
Firewall → LAN
LAN → Firewall
LAN → WAN

You MUST understand that each of these six different directions serve completely different purposes.

This representation is **WRONG**:

WAN ↔ Firewall
Firewall ↔ LAN
WAN ↔ LAN

This representation is also **WRONG**:

WAN ↔ Firewall ↔ LAN

This is **WRONG** too:

WAN <----------> LAN
WAN ↔ Firewall ↔ LAN

This is **CORRECT**, shown a different way, but basically the same thing as the other correct example:

WAN <---------- LAN
WAN ----------> LAN
WAN → FIREWALL → LAN
WAN ← FIREWALL ← LAN

You never configure traffic for bi-directional. If you want a specific type of traffic to be *initiated* from both directions, you must configure it for EACH direction separately.

You always configure traffic to be allowed or INITIATED from one side to another. Once a session is initiated, by default, the firewall will ALLOW the RESPONSE to be received in the other direction. You do NOT have to configure the firewall to allow responses coming from the other direction.

This is a global parameter we turn on, which is applied to all initiated sessions. Basically, the firewall builds an internal table that tracks all sessions, and when reply traffic is received on a valid session, the firewall permits the reply to go through. We will be examining this session tracking information later, in the troubleshooting section, using the 'netstat' command.

Make sense? You'll get it. Read on, I explain specific examples next.

Here is what we are going to setup on this firewall:

WAN → Firewall

- Nothing!! You NEVER want anyone on the Internet INITIATING a session to talk directly to your firewall! Everything gets dropped!
- The only exception is TOP SECRET! You enable Knock, which will allow one specific person (YOU) from a specific remote host to temporarily gain ssh access to the firewall directly. This can only be done by YOU, the firewall administrator. It is an optional “back door”, and is a closely guarded secret. There are even more advanced and secure ways than knock, such as single packet authorization. Some administrators don't even allow this. Your call.

WAN → LAN

- This is where you allow traffic from the Internet to go *through* your firewall to your internal servers: web server, email server, database server, etc.
- This uses what is called destination network address translation (DNAT), to port forward Internet traffic to your internal servers. Each type of traffic must be configured individually. For example, to allow access to your internal web server, both ports 80 and 443 must be forwarded to the IP address of your internal web server. For mail, ports pop and/or smtp and/or imap to your email server, etc.
- By default, nothing is forwarded. However, I have included commented out sections in the config file,

for you to use port forwarding, with examples for each type of traffic. To enable, simply uncomment each line, and enter the correct IP address of the internal server to be accessed.
- Another example is if you want to allow an employee to access their desktop machine remotely via the Internet. For example, if they wanted to use remote control software such as VNC or pcAnywhere from home, this is where you would enable this. You would keep the port closed, but give them their own unique (and top secret!) knock to temporarily open a unique hole, so only they can get in, and be port forwarded (and port redirected) to their own machine. Pretty cool, huh? I gave examples of this in the config file too.

Firewall → WAN

- This one confuses people. Why does the firewall need to talk to anyone on the Internet? LOL yes it does, actually. Not only does the Debian machine need to talk to the WAN (apt-get/aptitude to install packages, get software updates, security patches, etc.), but some of the packages we are installing on the firewall need to talk to the WAN.
- Squid, the proxy service that fetches and caches web pages for your internal users, is running on the firewall, and needs to *initiate* requests onto the WAN.
- The network time protocol (NTP) server is also running on the firewall, and needs to talk to the WAN.
- The DNS server is also running on the firewall, and needs to talk to the WAN.
- The ssh *client*, so YOU (and only you) can ssh from inside the firewall to the Internet.
- ICMP, so YOU can ping from inside the firewall to the Internet.

Firewall → LAN

- Not much here really. If you did allow yourself to knock into the ssh port, you will probably want to allow yourself to ping and ssh into the local LAN. That's about it.

LAN → Firewall

- This one confuses people. You will NEVER allow your users direct access to the Internet. Instead, they must talk to proxies on the firewall, then the firewall talks to the Internet.
- All LAN user web browsers must be reconfigured to point to the IP address of the firewall, and talk to port 8080, which is the DansGuardian server. Please note: DansGuardian will forward all web traffic requests to squid, which is port 3128 on the local host. Another note: any traffic from one service to another on the same host, goes through the internal interface, and does not go through the firewall.
- The DNS port must be open to the LAN, so all users can access the DNS server on the firewall.
- The NTP port must be open to the LAN, so all users can access the time server on the firewall.
- ICMP is typically opened up to the LAN, so all users can ping the firewall.

LAN → WAN

- Nothing! You don't want ANYONE accessing anything directly through the firewall to the Internet.
- You will probably make exceptions for yourself, and maybe your boss and other demanding management (which must remain top secret, of course!).
- I do include two other examples in the config file, to allow windows machines to get updates through the firewall, and to allow windows machines to access bitdefender for the free online virus scan/clean.

Site Preparation Documentation:

Before you build the firewall, you need to have your documentation ready. Many beginners skip this step, and wind up rebuilding the firewall from scratch! Don't you dare ask me how to change any of the names or addresses after you build the firewall. I WILL yell at you, then laugh and tell you to rebuild it. Don't be a schmuck, get your documentation ready, and keep it with the firewall or a secure location.

Determine all name and IP addressing:

Does your ISP provide you a dynamic IP address, via DHCP? Or is it static? You have to know ahead of time.

Things you must identify and/or decide on, BEFORE building the firewall:

- ✔ Firewall name
- ✔ Domain name
- ✔ WAN IP address *
- ✔ WAN network mask *
- ✔ WAN default route *
- ✔ ISP DNS server IP address #1 *
- ✔ ISP DNS server IP address #2 *
- ✔ ISP DNS server IP address #3 *
- ✔ LAN IP address
- ✔ LAN network mask

NOTE for dynamic WAN links (*): If your ISP provides a dynamic IP address (not static), then you do not need to document the WAN IP, WAN network mask, WAN default route, or ISP DNS servers. This information will be gathered by the firewall automatically via DHCP.

For the Firewall name, unless you have something specific in mind, I recommend you name it 'firewall'.

For the Domain name, make sure you predetermine this. I highly recommend you subscribe to dyndns.org, and register your domain name, BEFORE building this firewall. If you do it after you build the firewall, there's a HUGE chance the domain name you chose has ALREADY been registered by somebody else. You have been warned!

Internal LAN IP addressing plan:

You need a plan!

Your network should follow a predefined network addressing plan. I always make a spreadsheet, keep a printout in my desk, and as I add or change IP addresses, I pull out the sheet and write on it. That way, I always know who is using what IP address. Whenever I need to modify the DNS server, figure out the next available address etc., I always have the printout with me.

For your convenience, I have created a spreadsheet for you. Please download it, and fill-in all relevant information. Modify as desired.

DOWNLOAD SAMPLE IP ADDRESS SPREADSHEET HERE: www.abazaba.org/debian/lanipdoc.ods

Note: I assume you have fewer than 250 workstations in your organization. If you have more, than you are probably using the 10.x.x.x network, and should already know how to do this.

For those of you who need a plan, the spreadsheet uses the following assumed configuration:

EXAMPLE LAN IP NETWORK:

LAN IP network: 192.168.1.0
LAN IP address: 192.168.1.1
LAN IP broadcast: 192.168.1.255
LAN network mask: 255.255.255.0
LAN default route: 192.168.1.1
LAN DNS Server: 192.168.1.1

LAN IP address assignments:

- 1: firewall
- 2-9 : routers and switches

- 10-19 : file servers, print servers, game servers (LOL just kidding)
- 20-29 : privileged administration workstations (me, boss, etc.)
- 30-39 : administrative/executive staff
- 40-49 : department managers/leads
- 50-199 : standard users, assigned by firewall DHCP server
- 200-250: VPN

Wireless IP address assignments:

If you have a wireless network, simply put the wireless access points on their own IP network, and route between wired and wireless. For example, a wireless network can use the 192.168.2.0 network.

Wired IP address: 192.168.1.x (this is a router, use 2-9 range)
Wired network mask: 255.255.255.0
Wired default route: 192.168.1.1
Wired DNS server: 192.168.1.1

Wireless IP network: 192.168.2.0
Wireless IP address: 192.168.2.1
Wireless IP broadcast: 192.168.2.255
Wireless network mask: 255.255.255.0
Wireless default route: 192.168.2.1

Your wireless router will probably have a built-in DNS and DHCP server. Go ahead and use them.

Your needs may differ. Adjust all of the above as required.

Note: For those of you who are wondering why I am starting with the 192.168.1.0 network, and not the 192.168.0.0 network, it's because I'm *old school*. Back in the day, some routers (like Wellfleet, before Bay, before Nortel) couldn't handle the 0 network, so to be safe, we always started at 1. Feel free to use the 0 network if you prefer, it's up to you.

Prepare your workstation:

Before we build the server, we want to make sure your workstation has all of the necessary programs we will be using during the remote installation, configuration and testing phase. There are several programs you will find incredibly useful on the workstation.

Linux workstation:

If you are running Debian linux on your workstation, like me :) you want to prepare it for use as a remote configuration workstation.

Your linux workstation already has secure shell installed. We will be using it a lot. Make sure you have an easy way of launching it. Whether you are using GNOME or KDE (doesn't matter), I usually add a “quick launch” icon to the toolbar. Get used to doing things from the command line.

You might also want to install Krusader. It's the linux equivalent to the Windows WinSCP utility. It is a great secure remote file browsing/editing utility.

```
# apt-get install krusader md5deep cfv krename arj lha unrar unrar-free rar rpm unace unzip p7zip
```

Windows workstation:

If you are running windows on your management workstation, you also need to install some remote connectivity software.

Install SSH utilities:

The most popular is putty: http://www.chiark.greenend.org.uk/~sgtatham/putty/

Download putty.exe and psftp.exe. Putty is the DOS equivalent to ssh, providing a secure remote shell to the server. Psftp is the DOS equivalent to sftp, providing secure file transfer to the server.

We will be remote connecting a LOT to the server. It is best to create a quick launch icon that runs putty, and connects you to the server (by IP address for now, will use DNS name later). Go ahead and do that now.

Install remote file manager utility:

WinSCP is one of the most popular utilities for accessing a remote file system. It looks just like the Windows file manager, in a split screen format, showing your local file system on the left, and the remote linux file system on the right. Nifty.

Download and install WinSCP: http://winscp.net/eng/docs/screenshots

Put that on your quick launch bar.

Prepare the Firewall:

Before we start installing Debian, we need to prepare the firewall machine for installation.

Upgrade the BIOS: (optional)

It's a good idea to upgrade the BIOS before building the firewall, especially if it is not a new machine. This is probably the only chance you will get. After you build the firewall, you should NEVER upgrade the BIOS, because it may break the firewall!

Note: Upgrading the BIOS is dangerous to the machine, if you don't know how. One false move, and the machine is rendered useless (you cannot fix it). Only perform this if step if you are experienced. If you have never upgraded a BIOS before, well, now is not the time to learn LOL.

Download and burn the NETINSTALL CD:

- ✔ Download the latest **Debian Stable NETINSTALL ISO** cd, currently Etch 4.0r3 as of this writing. Do NOT get an ISO with a GUI such as GNOME or KDE. It is EXTREMELY inadvisable to install a GUI, which will be wasting disk space, and more importantly, a LOT of precious memory. You do NOT need a GUI on the server. We will do all tasks remotely. We also provide a graphical management system (webmin) you can run remotely.
- ✔ Burn the ISO to a CD, and make sure it burned okay.

Setup the firewall hardware:

- ✔ Do NOT install the 2nd NIC at this time. Make sure there is only one NIC installed. For people new to linux, configuring multiple NICs can be very difficult to get right. We will make sure the first NIC is working correctly BEFORE adding the second one.
- ✔ Go ahead and connect the server DIRECTLY to the Internet (this is to avoid IP/DNS/routing/resolver problems for those not familiar with customizing and troubleshooting these things).

Install the Operating System:

OK, let’s get started!

- ✔ If you haven't completed the Site Preparation Documentation, NOW is the time. You need it now.
- ✔ Boot the CD, and start the server build.
- ✔ Don’t install any of the application servers such as web, mail, samba, etc. Do not install the desktop environment. The only thing that should be selected for installation is 'base system'.
- ✔ Make sure you select the Internet sources, and update the sources. Don’t worry about upgrading anything yet, we’ll do that later.

- ✔ Use the server name and domain name you registered with your domain service.
- ✔ Purely optional: For the local user account (non-root), I usually name it something like 'linuxadmin' or 'administrator', and not my normal login name, because in a business setting, someone else may eventually be managing this firewall. You need to decide that now LOL.
- ✔ New linux users should use the entire hard disk as one partition, select: guided, use entire disk, all files in one partition
- ✔ When installation is done, reboot and test to your liking. Make sure it's ready. Make sure you can ping everywhere.

Configure the Package Management System:

For a fresh install, you should always finish configuring the package management system, and perform an upgrade, to make sure all available patches are installed, and everything is current.

Edit the sources.list file:

By default, the NETINSTALL will leave the CDROM media enabled as a source. We need to disable this, to make sure you can safely remove the NETINSTALL CD, and all updates are fetched from the Internet.

```
# vi /etc/apt/sources.list
```

Your /etc/apt/sources.list file should look something like this. Make sure you comment out the installation CD (there are usually two lines, and I always comment out the first one, and delete the second one):

```
#
# deb cdrom:[Debian GNU/Linux 4.0 r2 _Etch_ - Official i386 NETINST Binary-1 20080103-00:44]/
etch contrib main

deb http://ftp.debian.org/debian/ etch main
deb-src http://ftp.debian.org/debian/ etch main

deb http://security.debian.org/ etch/updates main
deb-src http://security.debian.org/ etch/updates main
```

Update the system and apply all patches:

Before we continue, we want to run the upgrade process, to ensure we are using the latest Debian Etch kernel patch, and have installed all package updates.

A note on apt-get vs aptitude: Although they can be used interchangeably, it is a good idea to stick to just one of them, because there are some subtle differences (I won't get into). Aptitude is a program that runs on top of apt-get. I tend to use aptitude because it is a little bit smarter, and better at resolving dependency issues. However, for this firewall, it is safe to use either one.

```
# aptitude update
# aptitude dist-upgrade
```

Install all updates. If an upgraded kernel patch is also installed, after it is done installing, reboot:

```
# reboot
```

Automate the update/upgrade process:

Just kidding!

Even though it is possible, you should NEVER automate the upgrade process on any production machine, whether it's a firewall, server or even just a workstation. Even though Debian Stable is fantastic with it's reliable upgrades, there is always the remote chance an upgrade will break something on your firewall. You

should always perform upgrades on your production firewall manually.

How often should I upgrade?

Debian Stable doesn't get a lot of updates. Most people manually run the update once a week. You can go once a month and be fine. Just remember though, the longer you wait, the more updates there will be, and the greater the chances a badly needed bug fix is waiting to be applied.

Every weekend, usually on Sunday, I will remote into our corporate firewall (from home) and run the update:

```
# aptitude update
# aptitude dist-upgrade
```

If the kernel gets patched, make sure you reboot the firewall. When I need to reboot the firewall, I usually wait until either late at night, or early in the morning, when I know nobody is on the network. If you live near work, you can usually safely reboot it remotely from home (I do). Just make sure you reconnect after a few minutes, to make sure it came back up ok, and everything is running.

Install Miscellaneous Services:

Before we begin configuring the main firewall packages, lets go ahead and install some needed miscellaneous services.

Install SSH Server:

At-a-glance:
Main directory: /etc/ssh
Main configuration file: /etc/ssh/sshd_config
Documentation: /usr/share/doc/openssh* (multiple directories)

We need to install the OpenSSH secure shell server, so we can remote connect to the server.

```
# apt-get install openssh-server
```

Disable ssh root login:

We want to beef up the default security for the ssh server. By default, it listens on port 22 of all interfaces, and allows root to login. At a minimum, we want to disable root login. Lets do that now.

Edit the **/etc/ssh/sshd_config** file, and make sure you set the 'PermitRootLogin' option to NO:

```
PermitRootLogin no
```

and restart ssh server:

```
# /etc/init.d/ssh restart
```

Note: Later, when we configure the shorewall firewall, we will be CLOSING port 22 access from the WAN, and configuring 'port knocking' to open it.

Install Time Synchronization (NTP) service:

At-a-glance:
Main directory: none

Main configuration file: /etc/ntp.conf
Documentation: /usr/share/doc/ntp, /usr/share/doc/ntp-doc

We want the date/time on this server to always be correct. The NTP service will synchronize with a public NTP server on the Internet. This service is auto-configured for you upon installation, and requires no configuration (in most cases).

```
# apt-get install ntp ntp-doc
```

The time on your firewall will now always be correct.

Install ddclient: (optional)

At-a-glance:
Main directory: none
Main configuration file: /etc/ddclient.conf
Documentation: /usr/share/doc/ddclient

If you have a dynamic IP address, and you registered a domain name with dyndns.org, you need to install this client. It will detect when your IP changes, and automatically update your A-record with dyndns.

NOTE: Make sure you already have the A-record set up on dyndns before installing ddclient.

```
# aptitude install ddclient
```

The post-installation script will ask you:

- fqdn of host: firewall.abazaba.org
- interface to monitor: eth0
- dyndns login name: name
- dyndns login password: password

That's it. Ddclient will monitor your WAN IP address, and notify dyndns.org every time it changes.

Configure 2nd NIC:

Before we actually install the 2nd NIC, we can go ahead and configure it. Take a look at the /etc/networking/interfaces file:

```
# cat /etc/network/interfaces
```

For a static configuration, it will look something like this:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo

auto eth0
auto eth1

iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
        address 72.91.54.192
```

```
        netmask 255.255.255.0
        network 72.91.54.0
        broadcast 72.91.54.255
        gateway 72.91.54.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 72.91.54.192 68.238.112.12 68.238.96.12
        dns-search abazaba.org

iface eth1 inet static
        address 192.168.1.1
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
```

For a DHCP configuration, it will look something like this:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet dhcp

# I added this stuff:

# The LAN network interface
iface eth1 inet static
        address 192.168.1.1
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255

auto eth0
auto eth1
```

Install 2nd NIC:

Shutdown the server:

halt

and install the other NIC. Make sure it is a NIC that is supported in linux. Most NICs older than a year are automatically supported.

Note: this how-to does not cover hardware issues. You need to have experience installing hardware.

Before you turn the computer back on, you need to finish connecting all networking hardware.
- ➔ connect the LAN port to an Ethernet switch
- ➔ connect another PC to the Ethernet switch

If the Ethernet switch has any server capabilities, such as DNS server, DHCP server, etc., make sure you turn that stuff off!!!

Make sure the Ethernet switch is powered up, so the LAN NIC in the firewall comes up at boot time.

Verify 2nd NIC is functioning:

Power up the server, and login as root.

There are a number of commands you should familiarize yourself with:

ifconfig
ifconfig eth0
ifconfig eth1
route
netstat -nr
arp -a
dig yahoo.com
ping www.yahoo.com

ifconfig eth1 should produce something like:

```
eth1      Link encap:Ethernet  HWaddr 00:1B:2F:34:57:C0
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21b:2fff:fe34:57c0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2608168 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3883163 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:561724795 (535.7 MiB)  TX bytes:370550585 (353.3 MiB)
          Interrupt:201 Base address:0xb800
```

If for some reason you think the server didn't detect the NIC, check this file:

cat /etc/udev/rules.d/z25_persistent-net.rules

Please note: Every time you install another NIC that the server detects, it adds that NIC's hardware information to the /etc/udev/rules.d/z25_persistent-net.rules file. If the ethx goes higher than eth0 and eth1 (you have eth2 eth3 etc) just delete those entries and reboot to get the count back down.

Other files you can check to troubleshoot the server:

more /var/log/dmesg (this is the boot log, the stuff you see scroll on the screen at boot time).

Test from your workstation:

Please note: the firewall is not yet done. We still cannot send packets through it to the Internet yet. Your workstation cannot access the Internet yet. That's okay. It CAN access the firewall.

At this point, you should be able to connect to the firewall from your workstation. You should be able to ping the firewall, and connect to it with ssh, using a secure shell. You should also be able to connect using the secure remote file manager.

- ➔ Make sure you can ping the IP address of the firewall from your workstation.
- ➔ Make sure you can open a secure shell to the firewall's IP address from your workstation.

Remote Installation and Configuration:

From this point forward, all of the remaining firewall tasks should be done remotely from your workstation. There is no need to work at the firewall console anymore.

From now on, we will be doing all firewall configuration from your remote management workstation.

- ➔ From your management workstation (Linux or Windows), open a secure shell to the firewall, and login as root.
- ➔ I usually open multiple ssh windows, to install from one, and test from the others.

Install dnsmasq:

At-a-glance:
Main directory: none
Main configuration file: /etc/dnsmasq.conf
Documentation: /usr/share/doc/dnsmasq

At this point, both network interfaces are up. Workstations on the local LAN can only ping and ssh into the firewall. They cannot ping anything past the firewall yet. Before we turn on the NAT/routing, lets get the DNS and DHCP services up and running.

```
# aptitude install dnsmasq
```

Make sure you read through the main configuration file for dnsmasq, it is: /etc/dnsmasq.conf

Restrict to LAN interface:

Dnsmasq should only listen to DNS and DHCP requests from the local LAN, and ignore from the WAN. Note: the firewall will also be configured to block DNS and DHCP requests from the WAN.

Edit the **/etc/dnsmasq.conf** file, find and uncomment the 'interface' line, and set it to 'eth1':

```
interface=eth1
```

Expand Hosts:

We want to enable the expand hosts option, so the DNS server understands each LAN workstation name is part of the local domain, and will add the domain name to them automatically.

Edit the **/etc/dnsmasq.conf** file, and uncomment the 'expand-hosts' line:

```
expand-hosts
```

Set Domain Name:

We need to tell the DNS/DHCP server the name of our local domain.

Note: Make sure you set the domain name to the exact same thing as you did when you installed Debian. If you don't remember the exact name you used, from the command line, enter: **dnsdomainname**

The "domain=" line should read something like this (use your domain name):

```
domain=abazaba.org
```

Configure DHCP Server (optional):

If you want a DHCP server available for your network, you need to turn it on, and configure the address

range. The configuration we are using allocates 100 addresses for DHCP. If you need more, modify as desired. The first DHCP address starts at 50. Most networks will reserve the first portion of addresses for various things, including: firewall, routers, switches, servers, and machines with static IP addresses.

TIP: I tend to give important people static IP addresses, so they can bypass some firewall restrictions placed on people who receive DHCP addresses. However, this only works on networks where none of the users are computer savvy. More on this later.

Edit the **/etc/dnsmasq.conf** file, uncomment the first 'dhcp-range' line, and make it look something like:

```
dhcp-range=192.168.1.50,192.168.1.150,12h
```

That's it. Dnsmasq should be configured correctly. Restart dnsmasq:

/etc/init.d/dnsmasq restart

Note: At this time, do NOT reconfigure your workstation to use the firewall as the DNS server. Will will test it after we finish configuring shorewall.

Install Shorewall:

At-a-glance:
Main directory: /etc/shorewall, /usr/share/shorewall
Main configuration file: ALL FILES IN MAIN DIRECTORY (see below)
Documentation: /usr/share/doc/shorewall* (multiple directories)

At this point, the two network cards are up and operational, but no packets can pass between them until we install and configure routing/NAT. We will do this with the shorewall package.

```
# aptitude install shorewall
```

Note: after shorewall installs, it will display a message:

```
#### WARNING ####
The firewall won't be started/stopped unless it is configured
```

Don't worry about this, we will address it shortly.

Make sure you read the shorewall documentation in the **/usr/share/doc/shorewall/** directory, such as the README.Debian.gz file.

Note: to uncompress the readme file, use: gunzip README.Debian.gz

Also make sure you read through the main configuration file for shorewall:

more /etc/shorewall/shorewall.conf

Copy example configuration files:

By default, when shorewall is installed, nothing is configured. This is intentional. The installer cannot read your mind. You have to decide how shorewall will be configured. However, it does come with a variety of sample configuration files, for various 'typical' firewall scenario's. We are going to use one of them. Take a look at the examples provided in the '/usr/share/doc/shorewall/examples' directory. We will be using the files in the 'two-interfaces' directory.

Also, take a look at the default files in: /usr/share/doc/shorewall/default-config

If you actually looked in them, you will see that they are all empty. Well, not exactly empty. They are full of comments. However, there are no actual configuration commands.

First, copy these default files to our configuration directory:

cp /usr/share/doc/shorewall/default-config/* /etc/shorewall

Next, uncompress the files in the example two-interface directory:

gunzip /usr/share/doc/shorewall/examples/two-interfaces/*

Now, copy the example files from the 'two-interfaces' directory:

cp /usr/share/doc/shorewall/examples/two-interfaces/* /etc/shorewall

Last, change file permissions so only root can access these files. Even though we will never let any users in this box, we should still follow server hardening guidelines:

chmod 600 /etc/shorewall/*

Key configuration files:

You need to understand the purpose and contents of the key files we copied from the 'two-interfaces' directory. These, along with the main config file (/etc/shorewall/shorewall.conf) are the most important. The others are for optional stuff that you may or may not use.

These are the primary configuration files, listed in ORDER OF PRECEDENCE:

/etc/shorewall/shorewall.conf	<-- main configuration file
/etc/shorewall/zones	<-- fw, net, loc
/etc/shorewall/interfaces	<-- net=eth0, fw=self, loc=eth1
/etc/shorewall/masq	<-- another word for NAT, masq everything from eth0 (loc) to eth1 (net)
/etc/shorewall/policy	<-- defines to default ACCEPT, REJECT or DROP to/from all interfaces
/etc/shorewall/rules	<-- overrides policy with specific port/application exceptions to allow traffic

Note: PLEASE READ these files! You should have some background or at least understanding of how firewalls are configured. You first configure in general overall terms, then slowly drill down into the details, making exceptions along the way. That way, in the end, the only things that can pass through a firewall are those things you specifically allow, mostly defined in the rules file. Everything else, by default, is blocked.

The zones file:

The default zones file is fine. No changes needed. You need to read the notes in this file, and understand the three zones defined: local (LAN), net (Internet), and firewall (within the firewall itself, including both NICs).

/etc/shorewall/zones

fw	firewall
net	ipv4
loc	ipv4

If you are going to add a DMZ, or VPN, etc., you would add an entry for each of those in the zones file.

The interfaces file:

The default interface file needs to be modified. The default does not allow a DHCP server to run on the firewall, which services the local (eth1) segment. To allow this, we need to edit the file, and add the ‘dhcp’ parameter at the end of the ‘eth1’ line.

/etc/shorewall/interfaces

Net	eth0	detect	dhcp,tcpflags,norfc1918,routefilter,nosmurfs,logmartians
loc	eth1	detect	tcpflags,detectnets,nosmurfs,dhcp

By adding ‘dhcp’ to the ‘eth1’ line, shorewall will now allow DHCPDISCOVER packets to be detected.

The masq file:

The default masq is fine. No changes needed. This is the file some people miss. This is where you turn on NAT (already turned on for us from this example file).

/etc/shorewall/masq

eth0	eth1

The policy file:

This is the second most important file. You had better understand it. The policy file needs some tweaking.

By default, the firewall is fairly locked down. For home use, it is probably locked down too much. However, for business use, it is not locked down enough.

Traffic from the LAN:

loc	net	**REJECT**	**info**
loc	$FW	REJECT	info
loc	all	REJECT	info

By default, all traffic from the LAN is allowed to access the Internet, through the firewall, which is BAD. It's too much freedom. In a business environment, we do not want the users directly accessing anything on the Internet. We will force them to use our proxy server for web access, and any other traffic will have to be specifically allowed in the the rules file, such as myspace chat clients, ssh sessions, ftp sessions, email sessions, streaming video/music sessions, etc. Those things, IF allowed, will be defined in the rules file.

We need to change the loc->net rule to REJECT, and turn on logging with the 'info' option. We want the firewall to log all unauthorized attempts to access the Internet (more on this later).

Now, people on the LAN cannot access ANYTHING on the Internet, by default. I know what you are thinking, that's not what I want. In a business setting, we want to SPECIFICALLY define each type of traffic we will allow users to access the Internet. We will do that soon in the rules file.

Notice the second line. All local traffic TO the firewall is rejected, which is GOOD. We don't want people poking around with our firewall. Again, we will define specific exceptions to the firewall in the rules file.

Traffic from the firewall:

$FW	net	REJECT	info
$FW	loc	REJECT	info
$FW	all	REJECT	info

By default, we do not want the firewall talking to ANYTHING. You may wonder why we don't want the firewall able to talk to anything. This is a security precaution. There are very few things the firewall needs to talk to directly, and we will make those specific exceptions later in the rules file.

Traffic from the Internet:

net	$FW	DROP	info
net	loc	DROP	info
net	all	DROP	info

This one should be obvious. By default, we don't want anyone on the Internet accessing ANYTHING on our firewall OR our local network. Exceptions, such as accessing our web or email server will be made in the rules file. Please note, with the DROP all, people on the Internet cannot even PING our firewall, which is what we want. In fact, they can't even detect the existence of our firewall, which is even better.

The rules file:

The rules file is **THE MOST IMPORTANT FILE**. This is where you define, usually by port, what specific traffic you will allow. This file is very flexible, and can grow to be quite long. You MUST have a solid understanding of the contents in this file. If you don't, you will never get the firewall working as you want.

The rules file is where you make specific exceptions, such as controlling if pings to the firewall from the Internet should be dropped or not (default is drop them). This is where you will ENABLE specific ports, to allow support for specific services. This is also where you port forward, like port 80 to a web server, or port

25 to your email server, etc.

Since by default everything is either REJECTed or DROPped, we need to add some rules to the rules file to accept or allow specific traffic types.

My default rules file:

Here is a good default rules file. We will tweak it more later. Please study it. As you can see, it uses some unique commands. For now, make yours look like this (don't worry about the tabs/spacing):

```
# From the Internet:
Ping/DROP       net     $FW     # turned off logging of pings
Auth/DROP       net     $FW     # breaks IRC clients, don't care
SSH/ACCEPT      net     $FW     # we will close this & add portknock later

# From the Internet to internal computers (PORT FORWARDING):
# DNAT          net     loc:192.168.1.15        tcp     25      # smtp server
# DNAT          net     loc:192.168.1.16        tcp     80      # web server
# DNAT          net     loc:192.168.1.16        tcp     443     # web server
# DNAT          net     loc:192.168.1.14        tcp     5631    # pcanywhere
# DNAT          net     loc:192.168.1.14        tcp     5632    # pcanywhere
# DNAT                  net     loc:192.168.1.20        tcp     5901    # vnc server
# DNAT                  net     loc:192.168.1.21:5900 tcp     32163  # vnc server (alter port #)
# DNAT                  net     loc:192.168.1.22:22     tcp     41484  # ssh server (alter port #)

# From the firewall to the Internet:
Ping/ACCEPT     $FW     net
SSH/ACCEPT      $FW     net
DNS/ACCEPT      $FW     net
NTP/ACCEPT      $FW     net
HTTP/ACCEPT     $FW     net     # apt-get uses 80
ACCEPT          $FW     net     icmp

# From the firewall to the local LAN:
Ping/ACCEPT     $FW     loc
SSH/ACCEPT      $FW     loc
ACCEPT          $FW     loc     icmp

# From the LAN (all users) to the firewall:
Ping/ACCEPT     loc     fw
DNS/ACCEPT      loc     fw
ACCEPT          loc     fw      tcp     8080    # dansguardian
ACCEPT          loc     fw      tcp     3128    # squid (close this after testing dansguardian)

# From the LAN (privileged machines) to the firewall:
SSH/ACCEPT              loc:192.168.1.1-192.168.1.29  fw
Webmin/ACCEPT           loc:192.168.1.1-192.168.1.29  fw
HTTP/ACCEPT             loc:192.168.1.1-192.168.1.29  fw
HTTPS/ACCEPT            loc:192.168.1.1-192.168.1.29  fw
ACCEPT                  loc:192.168.1.1-192.168.1.29  fw      tcp     3000    # ntop

# From the LAN (all users) to the Internet:
Ping/ACCEPT     loc     net
# DNS/ACCEPT    loc     net     # we can close this, since they get dns from this firewall
# NTP/ACCEPT    loc     net     # give specific rules to servers that need NTP access
# ACCEPT        loc     net:67.15.127.0/24      all     # specific remote hosts

# allow access to bitdefender.com for free online virus scan/clean
HTTP/ACCEPT     loc     net:66.223.50.0/24
HTTP/ACCEPT     loc     net:193.164.155.0/24
HTTP/ACCEPT     loc     net:66.40.145.0/24
HTTP/ACCEPT     loc     net:216.207.68.0/24
HTTP/ACCEPT     loc     net:168.215.74.0/24
HTTP/ACCEPT     loc     net:209.170.75.0/24
```

```
HTTP/ACCEPT   loc       net:209.51.185.0/24
HTTP/ACCEPT   loc       net:8.17.64.0/24
HTTP/ACCEPT   loc       net:199.7.51.0/24
HTTP/ACCEPT   loc       net:199.7.54.0/24

# allow access to microsoft for windows updates
HTTP/ACCEPT   loc       net:207.46.0.0/16
HTTPS/ACCEPT  loc       net:207.46.0.0/16
HTTP/ACCEPT   loc       net:65.55.184.0/24
HTTPS/ACCEPT  loc       net:65.55.184.0/24
HTTP/ACCEPT   loc       net:207.68.160.0/24
HTTP/ACCEPT   loc       net:69.147.114.0/24
HTTP/ACCEPT   loc       net:65.55.200.0/24
HTTPS/ACCEPT  loc       net:65.55.200.0/24
HTTP/ACCEPT   loc       net:68.142.118.0/24
HTTP/ACCEPT   loc       net:192.221.99.0/24
HTTP/ACCEPT   loc       net:209.73.188.0/24
HTTP/ACCEPT   loc       net:216.109.127.0/24
HTTP/ACCEPT   loc       net:4.23.40.0/24
HTTP/ACCEPT   loc       net:206.33.52.0/24
HTTP/ACCEPT   loc       net:204.160.126.0/24
HTTP/ACCEPT   loc       net:198.78.220.0/24
HTTP/ACCEPT   loc       net:209.18.38.0/24
HTTPS/ACCEPT  loc       net:65.55.192.0/24
HTTP/ACCEPT   loc       net:198.78.200.0/24
HTTP/ACCEPT   loc       net:131.107.115.0/24
HTTP/ACCEPT   loc       net:64.4.52.0/24

# first block managers getting to specific sites like myspace, youtube, etc.
REJECT        loc       net:204.16.33.0/24    all
REJECT        loc       net:216.178.32.0/24   all
REJECT        loc       net:208.65.153.0/24   all
REJECT        loc       net:66.151.149.0/24   all
REJECT        loc       net:209.10.40.0/24    all
REJECT        loc       net:38.98.192.0/24    all
REJECT        loc       net:207.188.21.0/24   all
REJECT        loc       net:69.31.48.0/24     all
REJECT        loc       net:66.7.181.0/24     all
REJECT        loc       net:207.138.234.0/24  all
REJECT        loc       net:65.59.234.0/24    all
REJECT        loc       net:216.17.104.0/24   all
REJECT        loc       net:216.218.248.0/24  all
REJECT        loc       net:68.142.219.0/24   all
REJECT        loc       net:216.241.160.0/24  all
REJECT        loc       net:208.82.7.22       all
REJECT        loc       net:208.82.5.22       all
REJECT        loc       net:208.82.6.22       all
REJECT        loc       net:208.72.33.0/24    all
REJECT        loc       net:208.72.34.0/24    all

# then allow managers direct access via general protocols
HTTP/ACCEPT   loc:192.168.1.1-192.168.1.29  net
HTTPS/ACCEPT  loc:192.168.1.1-192.168.1.29  net
SSH/ACCEPT    loc:192.168.1.1-192.168.1.29  net
FTP/ACCEPT    loc:192.168.1.1-192.168.1.29  net
POP3/ACCEPT   loc:192.168.1.1-192.168.1.29  net
POP3S/ACCEPT  loc:192.168.1.1-192.168.1.29  net
IMAP/ACCEPT   loc:192.168.1.1-192.168.1.29  net
IMAPS/ACCEPT  loc:192.168.1.1-192.168.1.29  net
SMTP/ACCEPT   loc:192.168.1.1-192.168.1.29  net
SMTPS/ACCEPT  loc:192.168.1.1-192.168.1.29  net
```

Turn off logging to the console:

By default, shorewall will send log lines to the console. If you never use the console, you will never see these lines. If you do use the console, these lines will annoy the !#$%&! out of you! I guess some people like it, but I don't. Here's how to turn it off:

edit /etc/default/klogd

set last line to: KLOGD="-c 4 -x"

Remove inhibitors:

The Debian package maintainers add a safeguard in the configuration file to prevent startup without configuration. Since we are aware of this, and know how to configure shorewall, go ahead and remove these inhibitors now.

Edit **/etc/shorewall/shorewall.conf** and set these lines to:

```
STARTUP_ENABLED=Yes
IP_FORWARDING=On
```

ALSO

Edit **/etc/default/shorewall** and change startup to 1:

```
startup=1
```

That's it! We should be ready to go.

Test configuration files:

First check the firewall configuration files:

```
# shorewall check
```

It should check everything, and at the end, say: Shorewall configuration verified. If it gives an error, make sure you fix the error.

Start firewall:

When ready, start the firewall:

```
# shorewall start
```

NOTE: Do NOT use **/etc/init.d/shorewall start** to start, stop or restart the firewall. Always use the **shorewall** command directly.

If it starts okay, it is now time to reconfigure your workstation to use the firewall.

If you want to see what rules (chains in iptables speak) are actively applied in the kernel, use the shorewall show command:

```
# shorewall show
# shorewall show loc2net
# shorewall show net2fw
# shorewall show capabilities
# shorewall help
```

Note: this output is best viewed on a wide screen (more than 80 characters) ssh session, window maximized.

Advanced Shorewall Configuration:

Please note: You can skip these now, and come back later to add them. If you are new to shorewall, I suggest you get comfortable with the default configuration we have built, before trying to add these advanced features.

If you are experienced with both Shorewall and these features, then feel free to add them now.

Use rate-limiting to drop SSH packets after 3 failed login attempts:

I don't use the rate limiting method to thwart SSH attackers. A much better method is to use port knocking, which keeps the SSH port closed to everyone except a valid knocker. However, if you want to implement this, here it is.

To prevent a hacker from using an automated script to brute force into your SSH server, you can use the rate-limiting feature. In the rules file, simply replace the SSH/ACCEPT line with this:

```
Limit:info:SSHA,3,60    net    $FW    tcp    22
```

This line tells the server to only allow three new SSH connection attempts per minute, and log all dropped attempts.

Add Port Knocking to protect the SSH port:

If you used the above defaults, the SSH port is open to both the LAN and WAN. We do not want to leave it open to the WAN, for anyone to connect to. Within 24-48 hours, you will discover in your logs that people are already trying to brute force login to your SSH server! It's ridiculous. We need to add port knocking.

First, we need to create the dummy action file. This file adds the action of SSHKnock to available actions.

touch /usr/share/shorewall/action.SSHKnock

Leave that file empty.

Decide what port # you want for your port knock:

The remaining configuration enables port knocking on port 1600, purely as an example. DO NOT USE PORT 1600. Pick a random number, between 1025 and 65535. Make sure you modify the below examples to use your new number.

Next, create the /etc/shorewall/SSHKnock file, and make it look like this:

```
if [ -n "$LEVEL" ]; then
    log_rule_limit $LEVEL $CHAIN SSHKnock ACCEPT "" "$TAG" -A -p tcp --dport 22   -m recent --rcheck --name SSH
    log_rule_limit $LEVEL $CHAIN SSHKnock DROP   "" "$TAG" -A -p tcp --dport ! 22
fi
run_iptables -A $CHAIN -p tcp --dport 22   -m recent --rcheck --seconds 60 --name SSH          -j ACCEPT
run_iptables -A $CHAIN -p tcp --dport 1599 -m recent                       --name SSH --remove -j DROP
run_iptables -A $CHAIN -p tcp --dport 1600 -m recent                       --name SSH --set    -j DROP
run_iptables -A $CHAIN -p tcp --dport 1601 -m recent                       --name SSH --remove -j DROP
```

If you studied the script, you will see that it will open port 22 upon receiving any TCP packet on port 1600. It will also close port 22 if it receives any TCP packet on port 1599 or 1601. If somebody port scans your computer, usually sequentially, this will prevent them from opening the port after a full scan.

Make sure you change 1600 to whatever number you have chosen. Change the 1599 and 1601 to the correct new numbers, -1 and +1 from your new number.

Next, edit the /etc/shorewall/actions file, and add this line before the last line:

SSHKnock

Next, edit the /etc/shorewall/rules file, and add this line (put it in the 'from Internet to FW'):

SSHKnock net $FW tcp 22,1599,1600,1601

You should be ready. Test the configuration files:

shorewall check

If everything checks out, then restart the firewall:

shorewall restart

Reconfigure workstation:

Now that the firewall is fully operational (hopefully), with routing/NAT enabled, and with the DNS/DHCP server enabled, it is now time to reconfigure your workstation to use the firewall as the default route and primary DNS server, and optionally the DHCP server as well.

Note: You should know how to do this. I will not cover it.

Make sure you use one of the privileged IP addresses, in the range 192.168.1.2-29. I use the admin range 20-29. I'm 20, the owner 21, etc.

Test and make sure your workstation has full connectivity, and can get to the Internet correctly.

Test the firewall:

Use nmap to test ports not blocked by the firewall:

To see what ports your workstation can use through the firewall, use nmap to scan an Internet site, from your linux workstation that is behind the firewall. Please note: port scanning is NOT acceptable by most sites, and you can get reported as a hacker. Make sure you scan a site that you control, or at least are friends with the administrator, or you know they are retards that don't check logs :)

If you don't have nmap installed, just install it:

aptitude install nmap

To see what ports are not blocked by your firewall, run nmap to port scan a remote site, from your workstation, through the firewall:

```
# nmap -P0 -sT some.internet.host.com -p1-65535
```

Your ISP may block some ports, but the available ports should still show up as filtered.

You can also port scan the firewall directly, to see what SERVICES are OPEN and available to the LAN.

Remote test the port knock:

NOTE: You can only perform this test from a remote Internet host, on the other side of the firewall.

SSH to a remote Internet host, then turn around and try to SSH back, trying to connect to the firewall.

Remotehost # ssh administrator@firewall.yourdomain.com

That should fail. We have to knock on port 1600, to open the ssh port 22.

ssh -p 1600 administrator@firewall.yourdomain.com

or:

nmap -P0 -sS -p1600 firewall.yourdomain.com

The port is now open. You have 60 seconds to successfully ssh into the firewall from your IP address. If you are going to open multiple ssh sessions, you have to do it in 60 seconds! After 60 seconds, you will have to knock again to open the port for another 60 seconds.

ssh administrator@firewall.yourdomain.com

Success! You have knocked the port open, then connected. When you are done, logout, then close the hole with:

nmap -P0 -sS -p1601 firewall.yourdomain.com

NOTE: You do not have to close the hole. It will automatically close after 60 seconds from the time of the ORIGINAL knock.

Other ways to knock the port open:
- install 'knocker'
- use your web browser to contact the server: http://firewall.yourdomain.com:1600
- other ways

Install psad:

aptitude install psad

Please read the configuration file. It is heavily documented. We need to tweak the main configuration file:

/etc/psad/psad.conf
- EMAIL_ADDRESSES root@localhost; (you can have multiple addresses here)
- HOSTNAME server; (put your local server name here)
- HOME_NET 72.91.54.0/24, 192.168.1.0/24; (put your LAN & WAN net's here)
- EMAIL_ALERT_DANGER_LEVEL 3; (you don't want to be flooded with emails, set to 3)
- IMPORT_OLD_SCANS Y; (keep history of attackers)

NOTE: *** I need to explain everything in this section ***
- explain alert levels, avoiding trivial email alerts, etc.

Install fwsnort:

NOTE: *** I need to add this section ***

Install squid:

aptitude install squid

Read the main configuration file: /etc/squid/squid.conf, it is HEAVILY documented.

Configuring Squid:

You have to decide how you want squid to be used: Transparent vs. Authenticated

In a transparent configuration, the users do not know it is there. They think they are accessing the Internet directly, but in reality, the squid server intercepts all of their Internet traffic, and caches it, thereby speeding up Internet access for everyone. This method is used when everyone has the same access rights, and don't need to control or log on a per-person basis. This may be fine for home use, but not for a work environment. We want authenticated control.

In an authenticated configuration, when the user first opens a browser window, the browser will ask for a login name and password to access the Internet. If they do not have a valid account, then Internet access is denied. This method is used when different types of people need different access levels, and/or logging needs to identify each person on each access attempt, whether successful for not.

Squid can authenticate users, using different methods (methods can be combined for fall-back purposes):

ncsa_auth	Authentication against an NCSA/Apache style password file
smb_auth	Authentication against an SMB (for example NT) server
getpwnam_auth	Authentication using the getpwnam() library call.
pam_auth	Authentication using standard PAM services.
ldap_auth	Authentication against an LDAP database.

Here is an actual configuration file, for using squid, with authentication, to a password file local to the server:

```
server:/etc/squid# cat squid.conf
http_port 3128
hierarchy_stoplist cgi-bin ?
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
acl apache rep_header Server ^Apache
broken_vary_encoding allow apache
access_log /var/log/squid/access.log squid
hosts_file /etc/hosts
refresh_pattern ^ftp:          1440    20%     10080
refresh_pattern ^gopher:       1440    0%      1440
refresh_pattern .              0       20%     4320
coredump_dir /var/spool/squid

auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/squid_passwd
auth_param basic children 5
auth_param basic realm Access to the Internet requires Authentication
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive off

acl all src 0.0.0.0/0.0.0.0
acl domainusers proxy_auth REQUIRED
http_access allow domainusers
http_access deny all
```

NOTE: For this configuration file, I have chosen to use the ncsa_auth mechanism, which is basically an encrypted password file stored on the firewall.

Create the password file:

touch /etc/squid/squid_passwd

We will later use webmin to add/modify/delete users from this file.

Install dansguardian:

aptitude install dansguardian

The main configuration file is /etc/dansguardian/dansguardian.conf file. It is HEAVILY commented. READ IT! There are a few lines you need to be aware of:

filterport = 8080 (this is the port dansguardian listens to, open it in the firewall, and point browsers to it)

proxyip = 127.0.0.1 (this is the IP address dansguardian will FORWARD traffic to, can be remote server)

proxyport = 3128 (this is the port dansguardian will FORWARD traffic to)

filtergroups = 5 (in this example, we will be defining five different groups in dansguardian)

filtergroupslist = '/etc/dansguardian/filtergroupslist' (this points to the file which assigns users to groups)

Install webmin:

Webmin used to be in the Debian archives, the last version supported was Sarge. It was removed by request from the Debian archives, due to too many bugs, including serious security violations. For a highly secure firewall, webmin is not recommended. However, for home and small businesses, webmin is fine. I include it here for those that wish to try it.

Before installing webmin, we first have to install some required perl packages:

#aptitude install libauthen-pam-perl libio-pty-perl libmd5-perl

Note: As of this writing, the current version of webmin is 1.400. Before installing, check the sourceforge web site, to identify the most current version available, then update the wget line accordingly.

wget http://internap.dl.sourceforge.net/sourceforge/webadmin/webmin_1.400_all.deb

dpkg -i webmin_1.400_all.deb

That's it! There is nothing to configure. Webmin is now installed and operational!

From your management workstation, point your web browser to:

https://firewall:10000/

Login as root, and take a look around. Do not 'click to install' anything. Don't make any changes yet. We are not yet done building this server. We'll come back to using webmin later.

Install miscellaneous:

iftop
etc.

What else?

SystemRescueCD

Managing the Firewall:

List active TCP connections:
- from command line: netstat -NT | grep EST
- from ntop: IP->Local->Active TCP/UDP Sessions

To tear down a TCP connection:
- 1st, add rule to firewall to block destination IP address (and optionally port)
- aptitude install cutter
- then from the command line: cutter IP

Netstat-nat

Great utility for listing NAT connections.

Other stuff:

rootkit detection

vnstat – network traffic logger and monitor