Using a Custom LiveCD and Firewall Builder to Provide Enterprise Level Security on a Budget

Jim Gadrow
GIAC Security Essentials Certification (GSEC)
Practical Assignment v1.4b, Option 1
February 26, 2004

Contents

Abstract

One of the most challenging requirements of security professionals is to implement projects with a minimum of risk in the “real-world” environment where the security budget is not unlimited and staffing levels are low. This paper describes and discusses the process of successfully completing one of those projects, an implementation of a wireless network segment in a remote office. Most importantly from a business perspective, is that the project was implemented with an acceptable level of risk, at an extremely low cost. The goal of this paper is not to provide a command by command script, but rather to provide an experienced guide to the sources and reference material used. It is hoped that it will provide a valuable starting point for others with similar needs or a desire to build upon the information presented.

In this paper, analysis of the project requirements lends itself to the proposal of using a customized Knoppix LiveCD system on read-only media as a distributed firewall. Advantages of remotely deploying the LiveCD are considered. Preferring a distributed firewall architecture for ease of management by a small staff, but commercial ones being cost prohibitive, a comparable Open Source application is chosen. Research is done into the procedures that will be required to accomplish the task. Stage 1 of the project itself will be configuring the WAP, and is only briefly discussed. Stage 2 of the implementation is done by mastering a LiveCd and is discussed in much greater detail. Stage 3, where the Firewall Builder software is configured to securely push a policy to a remote firewall over SSH using public key authentication is also discussed.

Background Info

My employer is a software development company. In one of our remote VPN-connected offices, our engineers are developing applications that will run on multiple wireless devices, including handhelds. In order to test their application development, they needed a small lab to be set up that allowed them to use a Wireless Access Point to connect their test devices to a server they are using on their local LAN.

My role as the Security Administrator for the network means it was my task to determine how to implement that test network, mitigating any risk as much as possible. In addition, my responsibility would include any moves/adds/changes, upgrades, replacements, etc. Not having a security team, everything must be able to be managed from remote, quickly and effectively. I also currently manage multiple other firewalls using Check Point’s Firewall-1 using a distributed architecture, with the enforcement points separate from the management server. Something similar was to be preferred in this instance. In addition, it needed to be done as inexpensively as possible as the budget for this small office is already stretched to the max and any additional dollar spent on this project means less that can be spent on others.

Determining the deployment method

Requirements

The following were the requirements for the project from our software engineers.

- A limited number of people need to test less than 5 wireless devices.
- Access only needs to be allowed from those devices.
- The devices need a single access point to access a development server on the internal LAN.
- Ports 137-139 are needed for Samba access.

I added the following requirements.

- The Wireless Access Point must be able to be securely configured.
- The solution chosen must be able to be managed from a remote office.
- A distributed system would be preferable as it would be more easily and quickly managed by 1 person.

Of course, the standard requirement from management was that it be done for as low of a cost as humanly possible.

Considerations

Risk

At the present time, wireless LANs "are widely viewed as inherently insecure"[1] because they broadcast radio signals that can pass beyond the physical bounds of the office. This means that if your office is in a rather densely populated area (as this one is), there are potentially thousands of people passing within range of the broadcast daily any one of whom has the ability to intercept and examine the data without your knowledge. Without looking any more deeply than that into the matter I already knew the risk was high, but in this instance the needs of the business required it to be deployed anyway so the best option was to mitigate the risk as much as possible by following best practices when deploying the WAP, briefly described later in this paper.

Cost

As one of those best practices, it was obvious that I needed to block unauthorized access between the LAN and wireless segment using a firewall. If funds for this project were unlimited, we could simply have purchased another FW-1 module, thrown it in place within minutes and managed it along with our other ones. In this case though, the cost was too prohibitive. Since I do use FW-1 in other locations in the network and am familiar with it, I know that we could have easily ended up spending thousands of dollars on yearly subscriptions for software and support for this module alone.

---

[1] Brewin, Bob. Verton, Dan., DiSabatino, Jennifer. "Wireless LANs" Trouble in the Air". Computerworld Magazine, January 14, 2002.
URL http://www.computerworld.com/securitytopics/security/story/0,10801,67344,00.html

It will be immediately understood by many people reading this paper that once other offices learn that something has been put in place here (the project is current and was just completed as the paper was being written), we will be asked to put something similar in most of them as well. I can guarantee that within a month, requests will be received from other offices saying basically, “They have wireless in their office, why can't we?”, and at least some will demonstrate a justifiable need. Firewall software and support costs can quickly become astronomical in such a case, with multiple remote offices.

Manageability

In addition to the cost, the solution needed to be easily managed since there's limited staff available. Assuming that we will be doing the same thing in other offices in the foreseeable future, the difficulty of managing multiple enforcement points becomes prohibitive just as quickly as cost factors do. A quick and easy stand-alone solution like Smoothwall might have been acceptable in this one instance, but would rapidly have become more difficult to manage as the number of similar installations grew. Having to access separate websites for each firewall is not nearly as easily managed as being able to use a distributed firewall, where policy can be pushed out from a management server. Demonstrating another reason why it’s so popular, a FW-1 module would have again been ideal since we already have such a setup, but I needed to try finding something less costly.

Assessment

Software

Fortunately, an article in the October, 2003 issue of Information Security Magazine[2] had caught my attention when it favorably compared a Linux iptables firewall to Check Point’s FW-1.

When a piece of software that costs thousands of dollars for the application and support is compared to a freely available open source application, it’s definitely interesting and worth taking note of, so I had bookmarked it to pursue it in more depth when the opportunity or need arose. That time appeared to have come. The article also made mention of another open source application called Firewall Builder that sounded very close to meeting my needs. From the same article:

> Firewall Builder (www.fwbuilder.com), an open source tool, implements an attractive GUI for iptables that allows drag-and-drop operations to edit policies in a manner similar to Check Point. Additionally, Firewall Builder can securely deploy a new iptables policy on a remote firewall over SSH via a simple shell install script. (Rash, 2003)

In its comparison, the article mentioned that iptables’ “initial installation, as well as the care and feeding, is more complex than that of a commercial product.” In my situation, I was quite willing to trade a bit of a difficulty during the setup if necessary in exchange for the greatly reduced cost. The article also discussed

[2] Rash,Michael. “Doing It Yourself”. Information Security Magazine. October 2003 (2003): 70 - 75.

the paid support available from Check Point vs. the freely available open source support. However, in this installation at least (and hopefully in the bulk of future similar ones), a need for outside support should be negligible, as it should be a fairly static system.

The one important point in the article that I disagreed with was where it said "iptables is strictly a one-to-one architecture, making it less scalable in large, distributed environments". While this comment may be correct for iptables alone, further research on the Firewall Builder website into its ability to deploy policies to remote systems proved that (at least with the current version) single or multiple policies can be created from a central policy server and securely pushed out to multiple remote firewalls over SSH. This would provide the distributed firewall architecture which fits our needs quite well. As a matter of fact, it can also manage remote firewalls "built on a variety of platforms including, but not limited to, Linux using iptables, ipfilter on FreeBSD or Solaris and pf on OpenBSD".[3] That gives me even more flexibility.

In short, what I learned from the comparison and related research perfectly answered our situation. It showed that an open source solution could be provided at little to no cost, that (at least in our situation) might require slightly more effort to set up, but would perform well and be reasonably scalable for similar setups. Using Firewall Builder and a Linux iptables firewall would give me the flexibility to save costs in future projects like this one when feasible. Of course, if a future project was a more difficult environment I could always fall back to using FW-1. In such an event the cost would be justified by the higher level of complexity. Since this solution appeared to meet all the requirements, I decided to pursue it.

Choosing the distribution

Although I tend to be more familiar with Red Hat Linux, I've been intrigued with the idea of a bootable LiveCD-styled system being used as a firewall because using read-only media immediately limits some of the damage that can be caused in the event of a compromise. The read-only media prevents an attacker from making any permanent changes in your firewall configuration. A reboot is all that's required to completely restore the system to a pre-attack state. Granted, if an attack is successful, a pre-attack state is still vulnerable but at least any rootkits, backdoors, other added software or configuration changes the attacker was able to make to the system are eliminated. In addition, if a new LiveCD can be easily created that repairs the vulnerability used by the attacker (or even just a simple configuration change), it can be quickly deployed to the remote office via next-day shipment and installed by anyone in the office who has the ability to reboot the system. Downtime due to reboots is not as much of a factor in a development test environment as it would be in a production environment, so that is not as much of a concern. Testing and reboots can usually be scheduled around each other.

[3] Kurland, Vadim. "Firewall Builder Frequently Asked Questions". Rev 1.42 November 25, 2003 URL: http://www.fwbuilder.org/archives/cat_faq.html

Since most of the LiveCD-type systems I've run across are based on the Knoppix distribution, I decided to look more deeply into using that as my base. I quickly found some instructions on how to remaster the Knoppix disks[4] which allow you to customize it to suit your own needs. In addition, Knoppix has excellent auto-configuration upon startup which recognizes a wide variety of hardware. I tried a couple of otherLiveCD systems I found, most notably Devil-Linux (www.devil-linux.org). Interestingly, from the introduction on the homepage I found that Devil-Linux is "directly supported by Firewall Builder".[5] What made it especially intriguing to me was that it takes the /etc directory and tars and bzips it up and places it all on a floppy, then extracts the info upon boot. However, I had difficulty bringing it up in VMWare which I use heavily as my testing environment, and also had problems trying to recognize as large a variety of hardware as Knoppix did, including PCMCIA. So I passed on it for the moment as I was under a deadline and couldn't afford to spend much time troubleshooting it. In this situation, it actually helped illustrate why Knoppix's ability to auto-detect, identify and configure hardware is one of its major benefits; one that will give us extra flexibility if and when we add on other systems in the future.

Hardware

The ability to recognize the PCMCIA Ethernet cards was highly desirable to me, as I was considering using a laptop system as the firewall. Since this was a remote engineering office with no IT presence, it wasn't unreasonable to expect that the firewall box and/or a replacement would have to be shipped back and forth as hardware failed, upgraded equipment was required, etc. Using a laptop system would help ease shipping, and would also take up less physical space in a small office. In addition, with the Wireless Access Point connected to the laptop with a cross connect cable, the system would be more portable within the office. This not only makes it more convenient to be moved around on different floors of the building as needed, but also allows it to be kept further away from outside walls to limit the range of the wireless network. This helps keep the system more secure by keeping the number of potential eavesdroppers to a minimum, and perhaps occasionally frustrates them in the event that the WAP moves to a different location, thus placing it back out of range if it hasn't been.

Minimum requirements of the project were much closer to being a doorstop than a state-of-the-art system. There would be less than 10 engineers using the system, testing less than 10 devices which would not put much of a load on the firewall. From the Knopper.net site[6], minimum requirements for running Knoppix are:

- Intel-compatible CPU (i486 or later),

[4] Knoppix.net. "Knoppix Remastering Howto".
URL: http://www.knoppix.net/docs/index.php/KnoppixRemasteringHowto
[5] Zuerker, Heiko. "Introduction". Devil-Linux homepage. URL: http://devil-linux.org
[6] Knopper, Klaus. Knoppix Homepage (English)
URL: http://www.knopper.net/knoppix/index-en.html

- 20 MB of RAM for text mode, at least 96 MB for graphics mode with KDE (at least 128 MB of RAM is recommended to use the various office products),
- bootable CD-ROM drive, or a boot floppy and standard CD-ROM (IDE/ATAPI or SCSI),
- standard SVGA-compatible graphics card,
- serial or PS/2 standard mouse or IMPS/2-compatible USB-mouse.

Meeting those minimums were obviously not a problem, and since the article in Information Security Magazine had given the nod to iptables' performance as compared to FW-1, I was confident that I did not require heavy-duty hardware to handle the intended light usage on this project. There was a laptop handy that had been turned in by one of the users when they upgraded to a newer system. That would serve quite nicely (and had no additional cash outlay), so I pressed it back into service.

Summary

To sum up the deployment I decided upon, I determined to create a custom Knoppix-based LiveCD running a Linux iptables firewall, and use it on a laptop that would serve as a gateway between a wireless segment and our development LAN. Firewall Builder would be used to create and push policy to it. This would allow me the flexibility to add additional enforcement points as needed, for little to no cost, using multiple platforms if necessary.

Using a customized LiveCD based on Knoppix:

- Protects against permanent damage from a potentially high-risk segment of the network.
- Allows the firewall to be upgraded in a remote office with no IT presence necessary.
- Can be remastered when and as needed.
- Allows backups to be made by simply copying the CD.
- Allows the firewall to be run on a large variety of hardware, even including laptops.
- Has no associated software costs.

Using Firewall Builder:

- Allows a simple drag-and-drop GUI interface to be used to create policy.
- Allows policies to be securely pushed to distributed firewalls.
- Allows policy objects to be reused for multiple firewalls.
- Allows flexibility of platform choice.
- Has no associated software costs.

Using a laptop for the hardware:

- Reduces anticipated shipping costs.

- Provides additional security by the ability to be placed in less exposed area.
- Provides additional convenience for developers.

Setting up the Wireless Access Point

Using Best Practices

The following table of best practices are taken from the WiFi Alliance's whitepaper entitled, "Enterprise Solutions for Wireless LAN Security" [7].

WLAN Security Best Practices

To protect your wireless LAN network from attack, the following best practices are recommended:

1. Educate employees about WLAN risks, with focus on threats from
   - Unauthorized attachment of APs (rogue APs)
   - Use of WLAN cards in ad hoc mode, especially when in public areas or any building with perimeter less than the WLAN broadcast range
   - Connect only to known APs; masquerading APs are more likely in unregulated public spaces
2. Deploy personal firewalls on all mobile machines. Use corporate network security policy to enforce their continuous use.
3. Actively and regularly scan for rogue APs on the corporate network, using available WLAN management tools, such as NetStumbler, AirMagnet, or AirDefense.
4. Change default management passwords on APs.
5. Change the default SSID on all APs, and allow the APs to broadcast their SSIDs. This enables users to easily identify the AP to which they are connecting and only present the necessary credentials.
6. Turn on and use WEP. It provides basic-level protection against the drive by snooper or unintentional visitor. WEP should always be used with other measures in a corporate environment.
7. When deploying 802.1X infrastructure to implement dynamic WEP, configure the session key update for at least once per hour to minimize the chance of key repetition.
8. Avoid placing APs against exterior walls or windows.

[7] Wi-Fi Alliance. "Enterprise Solutions for Wireless LAN Security". Whitepaper. February 6, 2003 URL: http://www.wi-fi.org/OpenSection/pdf/Whitepaper_Wi-Fi_Enterprise2-6-03.pdf

9. Reduce the broadcast strength of the AP when possible to keep it within the necessary area of coverage, and avoid coverage of unintended areas such as parking lots.
10. When planning network design, use 802.1X-based port authentication for wired switches and hubs to inhibit future addition of unauthorized, user-attached APs.
11. When using a VPN/firewall solution to protect campus WLANs, use IPsec-based VPNs with secondary authentication.

Setting up the Wireless Access Point was fairly straightforward. The particular one I used was a Linksys WAP54G, because it appeared to have the necessary configuration options that would allow me to deploy it securely. In example, unlike a couple of other models I looked at, this one did allow me to disable the SSID broadcast (Note: I disagreed with that part of #5 the Best Practices listed above, or at least in our situation the developers would all know the SSID and not need to easily identify it). The Linksys one also supports both 802.11b and 802.11g, 128bit WEP as well as WPA, and allows filtering by MAC address. Configuration of the WAP is done via a browser.

Mitigating the risk

I would like to have utilized the strong authentication provided by WPA. A PC Magazine article[8] I found on www.extremetech.com said, "Preshared key mode is appropriate for small offices and homes that don't have existing authentication infrastructures", and also "with WPA, the keys are changed so frequently that it becomes all but impossible to hack into a WPA-enabled wireless network". This sounded just right for this situation, as the devices would not be authenticating against our infrastructure, but just to the one local server. Unfortunately, in this particular instance at least some of the specific devices being tested did not support WPA. The WAP is capable of it though so we'll be able to use it when we eventually upgrade the test devices.

A mitigating factor in this setup is that the firewall blocks access to all but the previously mentioned server which is a development system and not a member of the internal domain. Since this wireless network is being used only for testing development, some additional security may also be gained by the simple method of turning off the WAP when not in use. It should not be assumed that this will always happen in real life, but properly educated users of the system may make the effort enough to deter an attacker from gathering enough packets to crack the WEP encryption. Between that possibility, the disabled SSID broadcasts, the firewalled access to the wired LAN, the ability to move the WAP away from the exterior walls, filtered MAC addresses, 128 bit WEP encryption, etc., we should have created enough layers of security to feel reasonably comfortable with this deployment. If nothing else, if an attacker was not deterred enough by the

[8] Ellison, Craig. "Keeping Your Wireless Network Secure". PC Magazine, October 6, 2003. URL: http://www.extremetech.com/article2/0,3973,1313018,00.asp

measures put in place, we get the satisfaction of knowing they'll have invested a bit of time and energy to find a lack of useful value in the possible target systems.

Remastering a disk

Once the method was decided upon, I needed to create the LiveCD. The Requirements and Procedures which follow are mostly based on the Remastering HOWTO[4] that I found on the www.knoppix.net website. (Don't confuse this one which is knoppIX.net and the previously mentioned knoppER.net, which are separate sites.)

Requirements

- Knoppix boot disk. This is not really required but the instructions given in the HOWTO assume you'll be using it, and doing so therefore guarantees compatibility with those instructions.
- 1GB available total of swap space + physical memory. (512 MB of physical memory plus 512 MB swap space will be just short of what you need, since you'll already be using some. You can see how much is available by typing 'free -otm'.
- Available hard drive partitions of at least 3 GB. You may need more if you need more swap space.

Also, if you have the drive space, go ahead and give it 6 GB or more. If you'll be doing this on multiple occasions you may want to give it the extra space to allow you to keep a local copy of the original files on the drive rather than to copy them from the slower CD, and to test your disk before you burn it. There is a very nice guide[9] at the O'Reilly www.linuxdevcenter.com site by Robert Bernier that gives step by step instructions to doing this. I personally found that those didn't port as well for me under VMWare, but since CD's are cheap, I just followed the Remastering HOWTO and made a few extra coasters. The development environment that Bernier describes however, appears to me to be superior for multiple builds and tweaks. Time and deadline was an issue for me, so I took the quick way rather than the more elegant one, but I'll likely set up that environment before I start deploying too many of these.

Procedure

Set up the development environment

- Boot from the Knoppix disk and ensure you have an available internet connection (can ping google.com for instance).
- Open up a shell as the root user.
- Mount the working partition read-write. It's very important that you note to mount it read-write, as depending how you mount it, Knoppix may default to read-only.
- Create your top levels of directory structure on the partition.

[9] Bernier, Robert. "Using and Customizing Knoppix". November 20, 2003
URL: http://www.linuxdevcenter.com/pub/a/linux/2003/11/20/knoppix.html

- Copy the uncompressed KNOPPIX files to your source directory. (this will take quite a while!) As mentioned, if you're going to do this multiple times, it's best to copy this to an 'original' directory on the local drive to save time the next time you need to start from scratch.
- Copy everything from /cdrom/KNOPPIX to your master directory except for the large file containing the compressed filesystem (that's the 'KNOPPIX' file which is around 700 MB).
- Chroot into your source directory. This will simulate the environment you'll be mastering to a new disk.
- Mount your /proc filesystem, so that the internet will be available. (Note: Don't forget to unmount it when you're done with the chroot session!)

Harden and configure your installation

At this point, you're in the environment that your firewall system will use. The first time you do this, you'll be customizing the firewall by hardening and configuring it. Everyone's system will obviously have different needs at different times, so it's well beyond the scope of this paper to describe specifically what you want to do here. In general, you'll want to make sure the system is updated with the most recent software and patches, remove any unnecessary services, and also any users and groups that aren't needed. You'll be able to change passwords as well while in the chroot'ed environment. Don't forget root and the default Knoppix user. Remember that "less" here is better as this is a firewall box, so if you don't need it get rid of it. There's an entire manual online at http://www.debian.org/doc/manuals/securing-debian-HOWTO/ which covers quite a bit of ground on securing Debian GNU/Linux, which is what Knoppix is based upon. You may also want to check out Bastille Linux (www.bastille-linux.org), which scripts the hardening process and does support Debian, though I didn't personally try it with Knoppix.

A couple of other papers that I found critically useful need to be mentioned here. The Knoppix-STD site (www.knoppix-std.org) has an excellent customizing document[10] which includes help on how to edit the boot floppy and miniroot file. Another very useful customizing document that was linked to from the Knoppix-STD site was Sunil Thomas Thonikuzhiyil's customization guide[11]. Sunil even suggests a list of some packages to remove that I cut and pasted into my own kick list, to provide a good starting point for removing the unnecessary ones. These papers give info on editing the /etc/init.d/knoppix-autoconfig file and commenting out the things you won't need. Some more interesting tweaks, such as changing the default user configuration and how to make Knoppix stop automatically logging on as a root user at bootup (in the inittab file), can be found

[10] Knoppix-STD.org. "How To Customize Knoppix-STD". Knoppix-STD website URL: http://www.knoppix-std.org/docs/customize.html
[11] Thonikuzhiyil, Sunil Thomas. "How I built a Custom Debian CD From Knoppix". URL: http://gnubox.dyndns.org:8080/~sunil/knoppix.php

in a tutorial by Alain Coron entitled “Knoppix Based Free Internet Access Installation Tutorial”[12].

In order to push the policy over from the policy server, you’ll need to create a group and user for ‘fwadmin’ and add the user to the sudo group. Create a directory structure under fwadmin’s /home directory to include a .ssh directory and copy the public key files (which you’ll create on the policy server in a later step) into there. While you’re at it, you may want to create an account that someone in the remote office can use to log on locally and reboot the machine. Also make sure to configure sshd_config as well, as you’ll be using that to push policy. At a minimum, change “PasswordAuthentication”, and “PermitRootLogin” to “no”. Make sure it only listens on your internal interface and accepts RSA authentication. Disable any rhost authentication. Start it from /etc/init.d/ssh so that it will create the host keys.

Once you’re done and satisfied with your system, don't forget to unmount your chrooted /proc filesystem before exiting the chrooted environment! Yes, that is the voice of experience you’re hearing. You’ll notice why you should have unmounted /proc when you try to build the compressed filesystem. Also make sure you clean up any old .bash_history, .vim_info, or tmp files that may have been created, plus the '.rr_moved' which is created as part of the ISO9660 process when the number of directories exceeds 8 levels.

Creating the disk

Now that you have your system ready to go, you need to create the compressed filesystem that KNOPPIX uses. This is what will allow KNOPPIX to have over 1GB and still run off of 1 bootable CD. (To be honest though, if you need that much space, you probably should go back and harden the system a bit more.) You'll probably want to script most commands to avoid retyping and the possibility of typos as they tend to be fairly long command lines with lots of options. You’ll create the compressed filesystem first. This part of the process takes a little bit of time, as it’s the part that uses up all your swap file.

After you do that, mount the boot.img and the miniroot.gz using the loop device. I found the Knoppix-STD instructions to be most helpful here. Check out syslinux.cfg in the boot.img, for changing the default boot to use your preferred cheatcodes automatically. Pay particular attention to the file linuxrc inside the gzipped miniroot file. This is where the directory structure such as the /home directories are created, and reading through the file will show you which directories are static or loaded fresh from scratch each time the system boots. Use this file to copy your fwadmin files from /KNOPPIX/home/fwadmin into the freshly created /home directory. At least that’s how I hard-coded it into the system. Your mileage may vary, and probably will. Once you’re done editing the

[12] Coron, Alain. “Knoppix Based Free Internet Access Installation Tutorial”. October 18, 2003. URL: http://alain.coron.free.fr/WCUFreeInternetAccess/

boot.img files, unmount them and copy the changed file into your mastering directory. Then you can update the md5 hashes and finally, create the ISO image file that you'll burn to CD.

Hopefully it worked and you now have a file called knoppix.iso that you can burn to disk to make your LiveCD firewall. Test it and see how it works. If you're like me, during the first run through I touched only enough to see if I could change something and have the process work. After the initial success, I went back afterwards and broke it by making too many changes. Eventually, I found one one that worked in my environment. Plan on making a few coasters. The good news is, after you've done it 3-4 times tweaking this and that, it becomes obvious that with a few scripts in place to do the grunt work, this can become a pretty efficient way to make changes and redeploy a disk.

Setting up the Policy Server

Preparing the policy box

To set up the policy server, I used a Red Hat v9.0 box rather than Knoppix simply because, a) I had it available and, b) Firewall Builder has pre-compiled binaries for Red Hat available for download from the fwbuilder.org site. This made it almost too easy to get started. I just downloaded the firewall builder packages and used rpm to install them. The one thing to watch for is to make absolutely sure you get the correct versions of the files that fwbuilder is dependent upon. I did find that it was a bit picky with just slight version changes, so make sure to get the exact file versions that are listed on the Firewall Builder site[13].

Once started, you'll have a drag-and drop interface that you can use to create the firewall policy. It's very intuitive and easy to use. On the Firewall Builder website there's a very nice short "Getting Started with Firewall Builder"[14] guide that will get you going, in addition to the 110+ page User's Guide. Firewall policies are beyond the scope of this paper, but there are some very nice examples in the "Firewall Builder's Cookbook"[15], as well as tips and tricks, HOWTO's and troubleshooting guides.

I found that it was a bit more effort to find documentation explaining the process of getting SSH and the install script to work with a distributed firewall, so I'll touch upon that. The clearest explanation I found was in some instructions at http://nil59.pisem.net/fwbuilder-relative/[16] that were for deploying a policy on an ipfilter firewall on a FreeBSD box. Not exactly my setup, but reading through the

[13] Kurland, Vadim. "Installation Guide". Firewall Builder website.
URL: http://www.fwbuilder.org/archives/cat_installation.html
[14] Barmala, Christian. "Getting Started With Firewall Builder". May 11, 2002
URL: http://www.fwbuilder.org/pages/cd/contrdoc3.html
[15] Firewall Builder. "Firewall Builder Cookbook" October 1, 2003
URL: http://www.fwbuilder.org/archives/cat_cookb.html
[16] Podolsky, Daniel. MiniHOWTO "Deploy fwbuilder-generated policy to remote FreeBSD-and-ipfilter-based firewall." URL: http://nil59.pisem.net/fwbuilder-relative/

instructions was very helpful in explaining what needs to happen for the policy to be installed on the remote box.

First off, note that Firewall Builder shouldn't be using password authentication for SSH as it's difficult if not impossible to supply one without having to put it in clear text somewhere, so it's best to use public key authentication. I cheated though, and until I was sure it worked, I left it turned on. If you do not already have a key pair, create one using 'ssh-keygen'. Make sure you're NOT logged into the shell as the root user, but use an account with less permissions. Assuming you use the defaults, your public (id_dsa.pub) and private (id_dsa) keys will be found in the .ssh subdirectory of fwadmin's (or whatever user you used) home directory. These are the ones I mentioned before that you'll need to copy over to the firewall box, or make accessible via a floppy, etc. The critical point here is to make sure the private key is secure, as anyone with the key will be able to log into your firewall box!

Test your configuration by trying to SSH to the firewall box. You should be able to connect without being prompted for a password. Be careful though, and during your testing and setup (while your firewall is NOT acting as a gateway), just set up a policy to allow Any -> Any on Any port to see if you have it configured correctly to push the policy out over SSH. Otherwise, it's quite easy to accidently deny the SSH connection from your policy server and lock yourself out of the box. No, that's not the voice of experience this time, but it could easily have been.

Scripting the Policy install

Also on the policy server, you'll want to create an install script to push the policy over to the firewall box once you get it working correctly. You can pass information such as the name of the user to the script by using the "Command line parameters for this script" box in the Compile / Install tab on the firewall object. The install script is another thing that may differ greatly from one setup to another, and can be as fancy or as simple as you want.

Not being overly familiar with the program yet, I searched for some resources to provide me with some ideas on what kind of a script I'd need to push a policy out to the box. I had already tested it by creating a policy, copying it over to the firewall using scp, and then using ssh to execute the policy on that box. But I wasn't sure if it would actually be as simple as just combining those commands together in an install script. Using Google, I found a post in the Debian-Firewall mailing list that said:

> If you've got key-based SSH authentication from your machine to your firewall box, a distribution script can be as trivial as:
>
> ```
> #!/bin/sh
> afile=firewall1.fw
> scp $afile root@192.168.0.1:/etc/firewall/
> echo "SCPd $afile to firewall1"
> ```

```
ssh root@192.168.0.1 /etc/firewall/firewall1.fw
echo "Executed new firewall script"
```

List that as the installation script in the firewall profile in fwbuilder and you're set for point-n-click activation.[17]

That answered my question, and following the thread yielded even more suggestions for more elegant scripts. At this point, you should understand the process and have enough of the procedures down to be able to build your own box.

Summary

As a professional in an ever-changing field, our job requires the ability to use the resources of the Internet to gather the information needed to accomplish the tasks set before us. This paper has attempted to demonstrate how this information was used and built upon to accomplish one of those tasks in a fairly unusual manner. The basic requirement of providing a secure Wireless Access Point for testing application development allowed me the opportunity to demonstrate how it could be done in a flexible manner that keeps risk low, without being too dificult to administer, and does so at an extremely low cost. Grand total cash outlay for the project ended up being less than $100 (for the WAP), since I appropriated an unused laptop, the policy server was already an existing machine that I use daily, and the entire project was implemented using open source software. It is hoped that the material presented has been useful to others, and that they will improve upon it.

Often in today's environment, individuals are required to do the job that teams used to do, usually with less financial resources than previously available to them. Many vendors are quite able to assist you in solving problems for fairly large sums of money, but many work environments demand that the same problem be solved for much, much less. Hopefully this paper has demonstrated how the requirement of low cost but acceptable levels of risk became a reality in this project. Although we all have a goal of the most secure possible environment, that may be secondary to the priorities of the businesses that employ us, and the ability to complete projects at or below budget is what enables us to get closer to our goal.

[17] Oxer, Jonathan. Subject: "Re: Firewall script builders". Debian-Firewall mailing list. September 5, 2003. URL: http://lists.debian.org/debian-firewall/2003/debian-firewall-200309/msg00021.html

References

[1] Brewin, Bob. Verton, Dan., DiSabatino, Jennifer. “Wireless LANs” Trouble in the Air”. Computerworld Magazine, January 14, 2002. URL:http://www.computerworld.com/securitytopics/security/story/0,10801,67344,00.html

[2] Rash, Michael. “Doing It Yourself”. Information Security Magazine. October 2003 (2003): 70 - 75.

[3] Kurland, Vadim. “Firewall Builder Frequently Asked Questions”. Rev 1.42 November 25, 2003 URL:http://www.fwbuilder.org/archives/cat_faq.html

[4] Knoppix.net. “Knoppix Remastering Howto”. URL:http://www.knoppix.net/docs/index.php/KnoppixRemasteringHowto

[5] Zuerker, Heiko. “Introduction”. Devil-Linux website. URL:http://devil-linux.org

[6] Knopper, Klaus. Knoppix Homepage (English) URL:http://www.knopper.net/knoppix/index-en.html

[7] Wi-Fi Alliance. “Enterprise Solutions for Wireless LAN Security”. Whitepaper. February 6, 2003 URL:http://www.wi-fi.org/OpenSection/pdf/Whitepaper_Wi-Fi_Enterprise2-6-03.pdf

[8] Ellison, Craig. “Keeping Your Wireless Network Secure”. PC Magazine. October 6, 2003. URL:http://www.extremetech.com/article2/0,3973,1313018,00.asp

[9] Bernier, Robert. “Using and Customizing Knoppix”. November 20, 2003 URL:http://www.linuxdevcenter.com/pub/a/linux/2003/11/20/knoppix.html

[10] Knoppix-STD.org. “How To Customize Knoppix-STD”. Knoppix-STD website. URL:http://www.knoppix-std.org/docs/customize.html

[11] Thonikuzhiyil, Sunil Thomas. “How I built a Custom Debian CD From Knoppix”. URL:http://gnubox.dyndns.org:8080/~sunil/knoppix.php

[12] Coron, Alain. “Knoppix Based Free Internet Access Installation Tutorial”. October 18, 2003. URL:http://alain.coron.free.fr/WCUFreeInternetAccess/

[13] Kurland, Vadim. “Installation Guide”. Firewall Builder website. URL:http://www.fwbuilder.org/archives/cat_installation.html

[14] Barmala, Christian. “Getting Started With Firewall Builder”. May 11, 2002 URL:http://www.fwbuilder.org/pages/cd/contrdoc3.html

[15] Firewall Builder. “Firewall Builder Cookbook” October 1, 2003 URL:http://www.fwbuilder.org/archives/cat_cookb.html

[16] Podolsky, Daniel. MiniHOWTO “Deploy fwbuilder-generated policy to remote FreeBSD-and-ipfilter-based firewall.” URL:http://nil59.pisem.net/fwbuilder-relative/

[17] Oxer, Jonathan. Subject: “Re: Firewall script builders”. Debian-Firewall mailing list. September 5, 2003. URL:http://lists.debian.org/debian-firewall/2003/debian-firewall-200309/msg00021.html

APPENDIX A – iptables vs FW-1

The following table which is taken from the referenced article in Information Security Magazine, illustrates their conclusions.

iptables vs. FW-1 How well does the open source Linux tool stack up against a leading commercial firewall?			
Category	***Iptables***	***Check Point FireWall-1***	***Advantage***
Packet Filtering	Port, IP, network, protocol, state tracking and content inspection.	Port, IP, network, protocol, encryption, state tracking and content inspection.	FW-1 is more flexible in setting rule specifications, but iptables runs a close second in functionality.
Policy Management	Command-line interface and shell scripts. Firewall Builder ad-on provides GUI functionality.	Easy-to-use GUI. Complicated rules management made simple by management console and/or Provider-1.	FW-1's GUI makes it superior. But Firewall Builder provides iptables with drag-and-drop functionality.
Integration of VPN, NAT and Routing Protocols	FreeSWAN provides an IPSec-compliant VPN; iptables has full NAT implementation; routing protocols provided by Zebra.	IPSec-compliant VPN. FW-1 has full NAT implementation; routing protocols not provided by Check Point (ipsrd on Nokia IPSO).	Add-ons give iptables comparable functions with FW-1, but they're not as easy to use.

Performance & Scalability	Up to 100 Mbps link capacity	Up to 100 Mbps link capacity	Equally matched, but iptables mau have an advantage in terms of maximum throughput on similar hardware.
Maintenance & Support	No dedicated support available, but open source community does provide ample free support. Open source code allows users to apply quick, custom fixes.	Upgrades, patches and technical support available through paid services.	User's discretion. Iptables has flexibility for do-it-yourselfers, while FW-1 has the backing of a well-established vendor.
Verdict	Iptables functionality and reliability are excellent, but requires more knowledge to deploy and maintain.	Functionality, support, and ease of administration are FW-1's strengths.	FW-1 gets the nod, but iptables is catching up.

A couple of things mentioned in the table should be pointed out. Encryption is mentioned as an extra function in FW-1 that iptables does not provide, but it does not mention that it comes at an additional cost to the base system. Similarly Provider-1 is mentioned, but that also comes at a substantial cost. Neither of these were required in this implementation in any case. We also didn't need VPN, NAT or routing protocols.

APPENDIX B – Scripts

From the Knoppix.net Knoppix Remastering Howto

```
##########################################################
```

Commands

```
##########################################################
#mount -rw /dev/hda1 /mnt/hda1

#mkdir /mnt/hda1/knx

#cd /mnt/hda1/knx ; dd if=/dev/zero of=swapfile bs=1M
count=750 ; mkswap swapfile ; swapon swapfile

#mkdir /mnt/hda1/knx/master; mkdir /mnt/hda1/knx/source

#mkdir /mnt/hda1/knx/source/KNOPPIX

#cp -Rp /KNOPPIX/* /mnt/hda1/knx/source/KNOPPIX

#mkdir /mnt/hda1/knx/master/KNOPPIX

#cp /cdrom/index.html /mnt/hda1/knx/master/

#cd /cdrom/KNOPPIX;find . -size -10000k -type f -exec cp -p
--parents {} /mnt/hda1/knx/master/KNOPPIX/ \;

#chroot /mnt/hda1/knx/source/KNOPPIX

#mount -t proc /proc proc

#rm -rf /mnt/hda1/knx/source/KNOPPIX/.rr_moved

#mkisofs -R -U -V "KNOPPIX.net filesystem" -P "KNOPPIX
www.knoppix.net" -hide-rr-moved -cache-inodes -no-bak -pad
/mnt/hda1/knx/source/KNOPPIX | nice -5
/usr/bin/create_compressed_fs - 65536 >
/mnt/hda1/knx/master/KNOPPIX/KNOPPIX

#cd /mnt/hda1/knx/master

#rm -f KNOPPIX/md5sums; find -type f -not -name md5sums -
not -name boot.cat -exec md5sum {} \; >> KNOPPIX/md5sums

#mkisofs -pad -l -r -J -v -V "KNOPPIX" -b KNOPPIX/boot.img
-c KNOPPIX/boot.cat -hide-rr-moved -o
/mnt/hda1/knx/knoppix.iso /mnt/hda1/knx/master
##########################################################
```

Remaster script

```
##########################################################
```

```
#!/bin/bash -x
 # This script builds a new KNOPPIX ISO image.
 # Copyright (C) 2004 by Marc Haber <mh+knoppix-
remaster@zugschlus.de>
 # License: GPL V2
 ROOT="$PWD"
 SOURCE="$ROOT/source/KNOPPIX"
 MASTER="$ROOT/master"
 CLOOPTARGET="$ROOT/master/KNOPPIX/KNOPPIX"
 TARGET="$ROOT"
 EXCLUDELIST="$ROOT/source/excludelist"
 rm -rf $SOURCE/.rr_moved
 cd $SOURCE
 mkisofs -R -U -V "KNOPPIX.net filesystem" \
         -P "KNOPPIX www.knoppix.net" \
         -hide-rr-moved -cache-inodes -no-bak -pad \
         -exclude-list $EXCLUDELIST \
         . | nice -5 /usr/bin/create_compressed_fs - 65536
> $CLOOPTARGET
 cd $MASTER
 rm -f KNOPPIX/md5sums
 find -type f -not -name md5sums -not -name boot.cat -exec
md5sum {} \; >> KNOPPIX/md5sums
 mkisofs -pad -l -r -J -v -V "KNOPPIX" -b KNOPPIX/boot.img
-c KNOPPIX/boot.cat -h ide-rr-moved -o $TARGET/knoppix.iso
$ROOT/master
##############################################################
```

Boot-Floppy script

```
##############################################################
#!/bin/bash -x
 # This script will loop-mount boot floppy and initrd image
 # Copyright (C) 2004 by Marc Haber <mh+knoppix-
remaster@zugschlus.de>
 # License: GPL V2
 unset CDPATH || true
```

```
# if not root, re-invoke self as root
if ?"`id -u`" -ne 0; then
       export LOCUSER="$USER"
       export LOCHOME="$HOME"
       if ?"${SHELLOPTS/xtrace/}" != "$SHELLOPTS"; then
               sudo bash -x $0 $@
               exit $?
       else
               sudo $0 $@
               exit $?
       fi
else
       LOCUSER="${LOCUSER:-$USER}"
       LOCHOME="${LOCHOME:-$HOME}"
fi
set -e
KNOPPIXDIR="KNOPPIX"
BOOTIMGFILE="$KNOPPIXDIR/boot.img"
BOOTIMGFS="vfat"
BOOTIMGDIR="boot.img"
INITRDGZ="$BOOTIMGDIR/miniroot.gz"
INITRDFILE="$KNOPPIXDIR/miniroot"
INITRDFS="ext2"
INITRDDIR="miniroot"
mountbootimage() {
  if ! modprobe loop; then
    echo >&2 "ERR: cannot load loop module"
    exit 1
  fi
  if ! ?-e "$BOOTIMGFILE"; then
    echo >&2 "ERR: no $BOOTIMGFILE found"
    exit 1
  fi
```

```
   for nofile in $BOOTIMGDIR $INITRDGZ $INITRDFILE
$INITRDDIR; do
     if ?-e "$nofile"; then
       echo >&2 "ERR: $nofile already exists"
       exit 1
     fi
   done
   mkdir -p $BOOTIMGDIR
   mount -o loop,uid=$LOCUSER -t $BOOTIMGFS $BOOTIMGFILE
$BOOTIMGDIR
   < $INITRDGZ gunzip > $INITRDFILE
   mkdir -p $INITRDDIR
   mount -o loop -t $INITRDFS $INITRDFILE $INITRDDIR
 }
 umountbootimage() {
   dd if=/dev/zero of=$INITRDDIR/nullfile || true
   sync
   rm $INITRDDIR/nullfile
   umount $INITRDDIR
   rmdir $INITRDDIR
   < $INITRDFILE gzip --best > $INITRDGZ
   rm -f $INITRDFILE
   umount $BOOTIMGDIR
   rmdir $BOOTIMGDIR
   syslinux KNOPPIX/boot.img
 }
 case "`basename $0`" in
   mountbootimage)
     mountbootimage
     ;;
   umountbootimage)
     umountbootimage
     ;;
```

```
   *)
     echo >&2 "ERR: called with unknown name `basename $0`"
     exit 1
    ;;
 esac
```

From Daniel Podolsky's MiniHOWTO

```
##########################################################
```

deploy-ipf-remote.sh

```
##########################################################
#!/bin/sh

if test -z "$5"
    then
    echo Insufficient parameters! >&2
    exit 1
elif test -z "$6"
    then
    XMLfile=$2
    WorkDir=$4
    SshUser=$USER
    FrwHost=$5
else
    XMLfile=$2
    WorkDir=$4
    SshUser=$5
    FrwHost=$6
    fi

echo "$0 called: Username: $SshUser, Firewall: $FrwHost,
Directory: $WorkDir"

CurTime=`date +%Y%m%d-%H%M%S`
```

```
ssh -C $SshUser@$FrwHost "echo \"Creating directory
\\\"$FrwHost:\$HOME/$CurTime\\\"\" && /bin/mkdir -p
\"\$HOME/$CurTime\"" || exit 2


scp -C $XMLfile
$SshUser@$FrwHost:$CurTime/policy.xml  || exit 3

scp -C $WorkDir/$FrwHost.fw
$SshUser@$FrwHost:$CurTime/script.fw   || exit 4

scp -C $WorkDir/$FrwHost-ipf.conf
$SshUser@$FrwHost:$CurTime/ipf.rules   || exit 5

scp -C $WorkDir/$FrwHost-nat.conf
$SshUser@$FrwHost:$CurTime/ipnat.rules || exit 6


ssh -C $SshUser@$FrwHost "/usr/local/bin/sudo
/usr/local/sbin/deploy-ipf-freebsd.pl \"\$HOME/$CurTime\""
|| exit 7
```