A Secure DNS Infrastructure

GIAC Security Essentials Certification (GSEC)
Practical Assignment Version 1.4c - Option 2
Brian Squadrito
February 23, 2005

Abstract:

This paper will attempt to demonstrate the building of a secure Domain Name Service (DNS) infrastructure. I will describe the original environment, the choices I made to get from that environment to where I wanted it to be, and an assessment of this implementation as it relates to security, reliability, and manageability.

This infrastructure is built on static DNS with BIND 9 on Solaris 8 (Sparc), in a stealth configuration, which means it does not accept queries from the internet. Public DNS records are hosted by an outside entity.

Environment:

The infrastructure I inherited had been designed as part of the evolution from a DOS/NetWare IPX-based network to a mixed operating system, IP-based environment with internet connectivity. Because the corporation was less than 5 years old, this infrastructure was built on recycled DOS workstations running RedHat linux and BIND 8. I had been taught how to add new subnet's to it's zone files (like a hosts file, it's a list of machine names and associated IP addresses) with a bash script, but had not done any further maintenance, as it had been hardened in ways I did not understand. With an absolute 0% downtime, it was a tough act to follow. Eventually, I was assigned to replace this infrastructure, with management's goals being that it run on supported hardware, and that I would be able to maintain it's complete functionality, as major changes to both the WAN and our application server infrastructure were on the way.

I had also inherited the management and monitoring of the firewall, mail relay server, and web proxy server, which obviously had been a higher priority for replacement/upgrade, as they directly connect to the internet. I had already replaced each of these machines with a Core Install of Solaris 8 on new hardware, hardened and scanned before being connected to the network, and each had already been through at least one penetration test and security review by an outside party. This experience had taught me to consider the security aspects of server deployment, and monitoring the firewall logs and the proxy server had taught me that with static IP addresses for every machine, I could easily coordinate information in the logs to get a more complete picture of what was going on. This meant it was easier to distinguish true security events from errors, proxy-ignorant applications trying to auto-update, or my developing paranoia. It had become obvious that the internet was becoming more hostile, that security was becoming more important, and that this trend was accelerating.

I had learned through these exercises how to secure a machine for a hostile network without sacrificing manageability. I could back it up to tape, mount and unmount a compact disk (CD), reboot, get to single-user mode, patch the operating system, and even restore it from tape. Most of these procedures I could not complete successfully on the inherited DNS machines because I didn't understand all the details of their hardening.

When the time came to prepare my first DNS server, some of the answers to the big questions of operating system and application software seemed obvious. I could easily reuse a retired but still supported Sparc server, with Solaris 8, running BIND. This simplifies procedure preparation for common tasks and training for my backup. Most of the challenges would revolve around learning the differences between version 8 and 9, and any interoperability bugs between the versions during the transition. My supervisor disagreed with these assumptions, instead suggesting another retired but still supported x86 server with NetWare as the operating system. Every person in the department had at least some experience administering NetWare, including my backup, and my supervisor had used NetWare as both DNS and DHCP servers concurrently in a previous job, which would allow us to easily add DHCP services for the small fleet of mobile users and for consultants/guests. I agreed and set up this server on NetWare 5.1, but it wasn't easy. There was no easy way to migrate the existing zone files (text) to the NetWare GUI administered server, meaning I had to re-create the entire zone file in a days-long procedure I like to refer to as a click-a-thon, as you would only willingly participate if you were being paid by the mouse-click. Add to this the complete retyping of the details of each entry when I already had a script to prepare these same files for me as text, and you'll see why I was beginning to regret this decision. I was hoping I could avoid this procedure with the backup DNS server, but now I had my doubts. When it came time to move the MIS department to a new subnet (the intended time of deployment), I found that the primary DNS server entries could not be changed on this platform. I was starting over, and my supervisor did not fault me for wanting to return to Solaris and BIND.

The one question everyone always asks when they find out I selected static DNS on BIND is, why not DHCP? The assumption that seems most common, and that I disagree with, is that management and installation of DHCP is easier than static DNS. This DNS server is effectively one giant centrally-managed hosts file, with several nice features added.

First, static DNS is more secure. I also manage the firewall, the intrusion detection system (IDS), and the web proxy server. I can trace all events back to an IP address to see the most complete picture of network events possible, when deciding if it's a false positive, a spyware/trojan victim, or a real threat to the network, that I've just been alerted to. Frequently, I'll call the user to ask them if they purposefully downloaded Firefox (UPX-EXE) or if a pile of mysterious pop-ups has just appeared while surfing. If we used DHCP, it would involve a more complicated install/configuration routine and additional maintenance to be able to provide this same level of security. In the very least, long-term leases based on mac addresses (which can easily be spoofed) would be required for the workstations, and the servers would still need static IP addresses.

If you did have an intruder on the inside, and you ended up going to court, wouldn't it be argued that unsecured DHCP services could be seen as an invitation (no intent required) to join the network? If a login banner says “Welcome” you may not be able to successfully convict an intruder, as the defense need only prove existence of doubt. Isn't it better to make them sniff the network before that guest laptop takes it down with a virus/worm/trojan? If they casually connect a DHCP configured client (the majority) unknowingly, isn't it better they only infect the PC's plugged into the same switch, than to give them the proper default gateway to infect your entire WAN?

On my employers static DNS network, consultants request permission to use network services or they are supervised in the use of a laptop configured in accordance with our network policy, whose IP address we already know is associated with the laptop commonly used in the executive boardroom, for example. When the request is made, the MIS representative asks which services they require, usually internet connectivity and/or printing, and comes back in a few minutes with an IP address we know is associated with that consultants laptop. We offer to make the changes for them, including the proxy server entry for their browser (which has all known-by-us malware distributors in a deny list), and verify that they at least have anti-virus running with current definitions. These days, guest laptops are generally locked down more than any workstation on our network. We've been able to get away with a more relaxed policy for laptops than most, as our file, print, e-mail, and proxy servers are not susceptible to any known virus, worm or trojan.

A discussion of the vulnerabilities in BIND 9.2.2 will be very brief. As of this writing, Security Focus' database has 1 vulnerability, NXDomain Denial of Service, which affects Microsoft Windows 2000 Server, Advanced Server, and Datacenter Server, at Service Packs 1, 2, and 3. I put this version into production almost 2 years ago, and it still has no vulnerabilities related to the application.

A second reason to choose static DNS over DHCP is reliability. When we were adding IP addresses to our servers and workstations, it was in preparation for Windows NT application servers, internet connectivity, and improved remote control for managing workstations with Novell ZenWorks. Our research told us that each of these uses would be more reliable with static IP addresses, especially ZenWorks remote control, which was brand new. All new IP-based services seemed to be designed for and tested first on static IP addressing.

A third reason to choose static DNS over DHCP is scalability. Static DNS on BIND easily scales to a network the size of the internet which grows at the rate of the internet.

A fourth reason to choose static DNS over DHCP is network efficiency/performance. Static DNS servers don't waste bandwidth broadcasting their existence. It's clients don't waste bandwidth broadcasting requests for an IP address, subnet mask, name server entry(s), or default gateway. To have any semblance of security, a discussion of the clients mac address (which can be easily spoofed) must ensue, only to end up with the same (static) address. Even if you settle for a long-term lease on the client side, your servers need a static IP address for the life of the application/service.

A fifth reason to choose static DNS over DHCP is manageability. The only entries we've needed to add to our zone files are for new servers or remotely manageable switches. You would still need to manually add these entries with DHCP (if you want it to be reliable and secure). When a workstation is replaced, DHCP requires a new mac address to be associated with an old IP address, and either requires that broadcasts be allowed across the WAN, or that a DHCP server be available at every site/subnet. With static DNS, we merely duplicate the IP address, and default gateway entries on the new workstation. Nothing needs to be done on the server side and there is no performance degradation for anyone but the user of the replaced workstation. With a WAN and Directory Services, we need to go into the workstations network properties anyway to set the proper context.

Implementation:

Now that we've selected BIND 9 on Solaris 8, it's time to plan the secure configuration of Solaris. The one biggest decision that affects the security of this installation/hardening of Solaris, is the choice of a Core install. It adds to the complexity of the installation and to the configuration in terms of usability, but it makes the hardening much easier as most of the unnecessary vulnerabilities are already gone, it makes required security patching far less frequent, and it greatly improves performance in every way, including backups and patching. As even Solaris patching requires at least one visit to single-user mode and at least one reboot, it is usually performed off hours to minimize visible down-time. If at all possible, do not connect this machine to a network until at least the hardening and patching are complete.

In order to keep this paper to a reasonable length, I will assume a familiarity with installing Solaris and will concentrate on those parts of the procedure that affect it's usability and security.

As this hardware was being recycled, the first choice to affect security is the choice of an Initial install at the Solaris Interactive Installation prompt. In this case, the integrity of the operating system wasn't the only reason for this choice. I was going to re-partition in preparation for a chroot jail with it's own partition.

When you get to the prompt to Select Software, select Core System Support. We are going to add a few packages, but without java being installed, the install cannot continue to the second CD to install additional packages on its own. We will mount the second CD manually, copy the packages to the hard disk, and then install them manually.

When you get to the Automatically Layout File Systems prompt, select Autolayout, /opt, /usr, and /var. At File System & Disk Layout, select Customize. Here, we backspace over the huge number in /export/home, so we can redistribute that disk space. I distributed the space on a 9 GB disk as follows:

/	1024 MB
/var	2048 MB
swap	1536 MB
/opt	2048 MB
/usr	1024 MB
/export/home	999 MB

This procedure installs all application software as packages from www.sunfreeware.com, which means all applications will be installed to the /usr partition (/usr/local). The /opt partition will serve as our chroot jail, and will be symbolically linked (like a shortcut) to /dns. None of these partitions need anywhere near this amount of disk space, as an entire Core install of Solaris is less than 400 MB, but the disk doesn't spin any faster by leaving unpartitioned space. The primary reason for separating the chroot jails partition is to isolate the filesystem whose permissions we'll customize later in this procedure.

When the install is complete and the machine has rebooted, you will log in as root, with no password. The next step is to make root's environment easier to use. Open /etc/passwd in vi, and change root's shell to /usr/bin/ksh. Now open /.profile in vi, and add the following lines to configure roots environment:

```
export PATH=/usr/sbin:/usr/bin:/usr/ccs/bin:/usr/local/sbin:/usr/local/bin
alias -x dns='/etc/init.d/dns'
alias -x __A=`echo “\020”` <tab> # up arrow = ^p = previous command
alias -x __B=`echo “\016”` <tab> # down arrow = ^n = next command
alias -x __C=`echo “\006”` <tab> # right arrow = ^f = fwd 1 character
alias -x __D=`echo “\002”` <tab> # left arrow = ^b = back 1 character
set -o emacs
```

Now add the executable bit for root's user to /.profile, with chmod. Use passwd to give root a password, exit, log back in and test your new environment. If you type a command and hit <Enter>, does the up arrow key on your keyboard show you the previous command? Does the left and right arrow key now move you back and forth through that line from your command history? If these things work and roots password behaved as expected, you now have a usable shell for your root user.

Now to install those other packages we need. First step (we'll need this again) is to echo the following mount command and pipe it to a script name of your choosing, preferably to somewhere in your path, like /usr/sbin:

```
mount -F hsfs /dev/dsk/c0t6d0s0 /mnt
```

(Note: This mount command assumes you burned the CD from a Microsoft Windows PC, you may need to burn it at 1X speed, and the filenames will be chopped down to 8 characters) Next, be sure to add the executable bit to this script for root. Now make a directory in /export/home to copy these packages into and change your directory to it. I usually call it syspat and then gunzip all the system patches there. Put Solaris Software CD #1 into the CDROM drive and test out your mount script. If it worked, you should be able to recursively copy the SUNWdoc directory from /mnt/Solaris_8/Product/ to your current directory (.). Unmount the CD with umount, and replace it with Software CD #2, then run your mount script again. You should be able to recursively copy the SUNWman, and SUNWgzip directories from /mnt/Solaris_8/Product/ to your current directory (.). Unmount this CD and install all three packages with a "pkgadd -d .".

Now we're ready to harden Solaris. This procedure will make it more likely to survive an attack and would still be considered hardened enough for most internal networks, but this is not how I would do it if it were going to be receiving queries from the internet. (My employers public DNS records are hosted by an outside party)

If you are preparing a public DNS server to host your mail's MX records and web site's address(es), I suggest you look into the Center for Internet Security's Level-1 Benchmark for Solaris. The web site is www.cisecurity.com , the fees for corporate membership are extremely high, and the Terms of Use Agreement for non-members is weird. Their Level-1 Benchmark for Solaris, however, is excellent, and you can copy, paste and edit their scripts to automate your hardening.

The following hardening procedure, while not as complete as the Center for Internet Security's and not ready for IPV6, was compiled by my reading of many a web page while preparing 5 Core Solaris machines for 5 separate applications over 3 years time.

First, we'll make Solaris' TCP Initial Sequence Number's harder to predict, by opening /etc/default/inetinit in vi, and changing TCP_STRONG_ISS=1 to TCP_STRONG_ISS=2. Next we'll remove /etc/rc2.d/S72inetsvc and replace it with a symbolic link to /etc/init.d/inetsvc. Open /etc/init.d/inetsvc and make the last line (used to start inetd) a remark by inserting a "#" in front of it. Now move /etc/inet/inetd.conf to /etc/inet/inetd.conf.ORIG to guarantee these services never start. To complete the hardening of Solaris' networking, add the following lines to the end of /etc/init.d/inetinit:

```
### Set kernel parameters for /dev/ip
ndd -set /dev/ip ip_respond_to_echo_broadcast 0
ndd -set /dev/ip ip_forward_directed_broadcasts 0
ndd -set /dev/ip ip_respond_to_timestamp 0
ndd -set /dev/ip ip_respond_to_timestamp_broadcast 0
ndd -set /dev/ip ip_forward_src_routed 0
ndd -set /dev/ip ip_ignore_redirect 1
ndd -set /dev/ip ip_respond_to_address_mask_broadcast 0
ndd -set /dev/ip ip_send_redirects 0
#
set tcp: tcp_conn_hash_size=8192
ndd -set /dev/tcp tcp_slow_start_initial 2
ndd -set /dev/tcp tcp_close_wait_interval 60000
ndd -set /dev/tcp tcp_conn_req_max_q 1280
ndd -set /dev/tcp tcp_conn_req_max_q0 10240
```

Next, we'll add the following lines to the end of /etc/system to prevent stack smashing:

```
set noexec_user_stack=1
set noexec_user_stack_log=1
```

Now, to disable the startup scripts of vulnerable services, rename each of the following by moving them to the same name with an underscore as the first character:

```
/etc/rc2.d/S71ldap.client
/etc/rc2.d/S71rpc
/etc/rc2.d/S73cachefs.daemon
/etc/rc2.d/S73nfs.client
/etc/rc2.d/S76nscd
/etc/rc2.d/S80PRESERVE
/etc/rc2.d/S88sendmail
```

```
/etc/rc2.d/S93cacheos.finish
/etc/rc3.d/S15nfs.server
```

Now to prevent system users (root, nobody, etc.) from remotely accessing your machine in insecure ways, execute the following commands:

```
touch /var/adm/loginlog
touch /var/adm/sulog
touch /.rhosts /.netrc /etc/hosts.equiv
chmod 640 /var/adm/loginlog
chmod 640 /var/adm/sulog
chmod 0 /.rhosts /.netrc /etc/hosts.equiv
cat /etc/passwd | cut -f1 -d: > /etc/ftpusers
```

Now to set up some warning banners. The purpose of these is to make sure that when/if you bring an intruder to court there is no way they can claim they didn't know they didn't belong on your server. The console login (/etc/issue) banner will appear on the screen above the login prompt, and should include at least the following warning:

```
All unauthorized access to this machine is strictly prohibited.
All violators will be prosecuted to the fullest extent of the law.
All access may be monitored and logged, and constitutes
acceptance of these conditions.
```

You may be tempted to say all access will be monitored and logged, instead of may be. Unfortunately, this may be interpreted by the courts as a burden to monitor every keystroke typed at that console. A camera may be required to prosecute if you do.

The following banners for telnet and ftp will suffice, as we've already assured these services will not run earlier in this hardening procedure:

In /etc/default/telnetd
```
  BANNER=" "
```

In /etc/default/ftpd
```
  BANNER="WARNING:  Authorized use only"
```

At this point, we should reboot to our hardened state and get ready for patching. The right way to do this from command line is to execute the following command:

```
shutdown -y -g30 -i6
```

For this install's first patching, I downloaded the Sun Recommended and Security patch cluster from SunSolve and burned it on CD. I put the CD in the CDROM, mounted the CD with the script we created and used earlier, copied the cluster to that /export/home/syspat directory, unzipped it, shutdown to single-user mode with a shutdown -y -g30, changed into the directory it unzipped to, and executed a ./install_cluster. As usual, when the patch completes, we reboot with a shutdown -y -g30 -i6. When I check that all our startup scripts for vulnerable services we don't need are still disabled and see that the patch has turned sendmail back on, I have a choice. I can continue to rename the "/etc/rc2.d/S88sendmail" script after every patch, or I can remove the sendmail packages altogether so the patch won't update it. Eventually, I looked up the names of the two sendmail packages with pkginfo -i and removed those packages with pkgrm.

Next, I prepared a backup script using ufsdump, since we certainly can't use any of those fancy and expensive GUI tools on a Core install. Besides, ufsdump is part of the operating system, and is therefore supported by Sun's standard support contract. First, this script assumes we've made a directory (mkdir) called /export/home/bkup. Then we would type the following into our new backup script, called bkmeup, with vi:

```
#!/usr/bin/ksh
# by InsertNameHere, for InsertCompanyNameHere, InsertDateHere
# Usage:  /usr/sbin/bkmeup 20050223
# where 20050223 is today's date
mt -f /dev/rmt/0 rewind
mkdir /export/home/bkup/$1
/usr/sbin/ufsdump 0uaf /export/home/bkup/$1/ufsarch$1a /dev/rmt/0n
/dev/dsk/c0t0d0s0 /
/usr/sbin/ufsdump 0uaf /export/home/bkup/$1/ufsarch$1b /dev/rmt/0n
/dev/dsk/c0t0d0s6 /usr
/usr/sbin/ufsdump 0uaf /export/home/bkup/$1/ufsarch$1c /dev/rmt/0n
/dev/dsk/c0t0d0s1 /var
/usr/sbin/ufsdump 0uaf /export/home/bkup/$1/ufsarch$1d /dev/rmt/0n
/dev/dsk/c0t0d0s5 /opt
/usr/sbin/ufsdump 0uaf /export/home/bkup/$1/ufsarch$1e /dev/rmt/0n
/dev/dsk/c0t0d0s7 /export/home
mt -f /dev/rmt/0 rewind
/usr/sbin/ufsrestore -tf /dev/rmt/0n > /export/home/bkup/$1/ufsarch$1a_index
/usr/sbin/ufsrestore -tf /dev/rmt/0n > /export/home/bkup/$1/ufsarch$1b_index
/usr/sbin/ufsrestore -tf /dev/rmt/0n > /export/home/bkup/$1/ufsarch$1c_index
/usr/sbin/ufsrestore -tf /dev/rmt/0n > /export/home/bkup/$1/ufsarch$1d_index
```

```
/usr/sbin/ufsrestore -tf /dev/rmt/0n > /export/home/bkup/$1/ufsarch$1e_index
mt -f /dev/rmt/0 rewind
echo $1 “Job's Done”
```

Be sure to make this script executable for root. I shutdown to single-user mode before running this script, and put a tape in the tape drive.

Now we're ready to begin preparations for this machine to serve DNS. To insure insecure services are not used to poison your DNS servers cache, rename /etc/nsswitch.conf to nsswitch.conf.ORIG, copy /etc/nsswitch.files to /etc/nsswitch.conf, and modify the hosts entry within to look like the following:

```
hosts: <tab> files <tab> dns
```

Next, we need to properly configure /etc/resolv.conf. On your primary DNS server, be sure to include a nameserver entry with the IP address of any secondary (backup) DNS servers, or they will not receive updates from the primary when you make changes, no matter how well you configure BIND. Your /etc/resolv.conf on your primary DNS server should include the following lines:

```
domain <tab> <mycompanyname>.com
search <tab> <mycompanyname>.com
nameserver <tab> 127.0.0.1
nameserver <tab> <IP address of secondary>
```

This next part is optional, but beneficial both for the ability to securely manage this server remotely and because it encourages you to read a README.privsep file which explains OpenSSH's use of privilege separation. In brief, privilege separation allows an application to have root privileges where it needs them, while allowing the parts of the application that interact with network clients (the dangerous part) to run as a separate, unprivileged user. Privilege separation is the new feature of BIND 9 that most influenced it's selection.

I will not detail all of the steps of this optional installation of OpenSSH, as I followed the steps of the “Installing OpenSSH Packages for Sparc and Intel/Solaris 8” document, from sunfreeware (http://www.sunfreeware.com/openssh8.html), so closely as to reference it in my notes and put checkmarks next to each step as I performed it. For what it's worth, I do recommend the optional step of “Setting up tcp_wrappers,” and would add only two additional considerations. After you set up a maintenance user and sudo later in this procedure and have tested his/her ability to perform actions with root privileges, I recommend opening /usr/local/etc/sshd_config and changing “PermitRootLogin yes”, to “PermitRootLogin no”, and executing

"/etc/init.d/sshd restart". This adds logging, via sudo, of remote actions performed with root privilege, as you will not be allowed to log in (directly) as root, via SSH. I also like to insure the ability to remotely access this machine, even while DNS services might be unavailable, by adding the IP address to the ListenAddress line, while in /usr/local/etc/sshd_config. If you make a mistake configuring your DNS server, you will still be able to recover remotely, by referring to the machine by it's IP address with your SSH client.

It's finally time to install and configure BIND. Download the package from sunfreeware, gunzip it, and install it by pointing "pkgadd -d" at it. As I had no idea how long it would be before vulnerabilities were found in it, I followed the instructions I found in a document by Sean Boran called "Running the BIND9 DNS Server securely", which can be found at http://www.boran.com/security/sp/bind9_20010430.html. Again, I have a copy of this document with checkmarks on the steps I followed exactly, so I will only point out those steps where my procedure varied or where a choice was to be made.

Step 1 was to compile from source. I do compile all services exposed to the internet from source, so that I know all the choices that have been made and options that have been added, but for this Stealth DNS server, I chose to install from a package.

In item 1, step 2, Setting up chroot and Installing BIND, keep in mind that this procedure assumes your jail will be located at /opt/dns, rather than the instructions selection of /home/dns. Once the symbolic link to /dns is created, you can follow this procedure exactly.

Instead of item 4, step 2, Install the bind distribution, perform the following:

```
mkdir /opt/dns/usr/local/sbin
cd /opt/dns/usr/local/sbin
cp /usr/local/sbin/named .
find / xfer* | grep xfer*
cd /opt/dns/usr
mkdir sbin
cd sbin
cp /usr/sbin/named-xfer .
```

Note that although item 5's use of the ldd command failed for me, copying the "for example on Solaris 8" worked.

In step 3, Setting up DNS Data Files, execute the following modification to your LD_LIBRARY_PATH environmental variable first, to avoid a fatal error starting with "ld.so.1: named-checkconf: fatal":

```
LD_LIBRARY_PATH=.:/usr/local/ssl/lib; export LD_LIBRARY_PATH
```

In step 4, Setting Jail Permissions, consider following the permission recommendation labeled “For secondaries”, even on your primary DNS server. This looser permission setup will be required if you later add Microsoft's Active Directory Services (ADS) to your network, as ADS “domain” controllers require the ability to update your forward zone file.

Before step 5, Running BIND, copy the libraries in /usr/local/ssl/lib to /dns/usr/lib and change ownership of /dns/usr/lib/* to root:named, to avoid an error “/etc/rndc.key not found”. BIND 9 was running with clean logs at this point, in it's chroot jail. It's a bit of typing, but I created the startup script /etc/init.d/dns to mirror Sean Boran's example at http://www.boran.com/security/sp/bind/dns and added the executable bit for root.

To test this automation of the startup and shutdown of your chroot'd DNS server without actually rebooting, execute the following commands:

```
dns start
dns restart
dns reload
dns stop
```

As I was not ready to put this DNS server into production without secure, logged, remote access, I chose this time to download sudo, again from sunfreeware. Follow the, hopefully now familiar, gunzip and pkgadd routine, and create yourself a maintenance user (let's call him maint) with the following command:

```
useradd -d /export/home/maint -m -s /usr/bin/ksh maint
```

Give him a password with the passwd command, as we're about to give him root privileges, and execute /usr/local/sbin/visudo. Add the following line to the user privilege specification:

```
maint <tab> ALL=(ALL) ALL
```

Next, touch /var/log/sudo to create an empty log file, and copy /.profile into “/export/home/maint/” . Now open /etc/syslog.conf with vi and add the following line, keeping in mind how syslog hates any whitespace that isn't tab:

```
local2.debug <tab><tab><tab><tab><tab> /var/log/sudo
```

Issuing the commands “/etc/init.d/syslog stop”, and then “/etc/init.d/syslog start” will allow you to test without rebooting. Test your maintenance user's environment, that sudo prompts for a password (maint's password, not root's), and that whatever command was sudo'd is entered into the log file at “/var/log/sudo”. If everything works, it's time to change PermitRootLogin to no in “/usr/local/etc/sshd_config”.

At this point in the procedure, I verified that netstat only showed DNS and SSH as listening and I plugged in the network connection. Then I tested that I could connect to it via SSH, from a workstation allowed by tcp_wrappers, so that I could configure BIND and migrate the zone files while sitting at my desk.

Although the zone files were copied unmodified from an existing DNS server, I still had questions about those settings at the top of each, especially those relating to the expiration of records in the cache. I found most of my answers explained most clearly in the “DNS & BIND Cookbook”, written by Cricket Liu and published by O'Reilly, which had made Chapter 2, Zone Data, available on the internet. If you have to build your zone files or your named.conf from scratch, you'll definitely want this book.

With the zone files already built, I was able to move on to configuring “/dns/etc/named.conf”. Still starting from Sean Boran's example, I was able to copy the pointers to the zone files from an existing DNS server exactly. This left only those settings related to security and to the relationships between the DNS servers for me to configure. These I will describe here.

In the access control lists section, is an acl labeled “nameservers”. Add the IP addresses of your other DNS servers, seperated by semi-colons. There are examples to follow already in there, with double slashes to reduce them to comment status.

For security, there is a line with version “DNS server”;, which is commented that it's purpose is to “hide BIND version”. This means an attacker has to work to see which vulnerabilities your DNS server might have, hopefully triggering alerts before they find any good ones.

Immediately following, is “allow-query”. Within the curly brackets, add the private address ranges in use on your network and the public address range that your DMZ machines are NAT'd to, so they have permission to query your DNS server.

Within your primary forward zone file reference, type will be master on your primary DNS server or slave on your secondary DNS servers. To allow Microsoft

ADS domain controllers to update this file, add an “allow-update” line, with the domain controllers IP addresses in curly brackets, seperated by semi-colons, and don't forget the semi-colon after the close curly bracket. Also recommended in any type “master” zones, is a “notify yes;”, if you'd like it to notify your secondary DNS servers of changes you've made in the primary DNS server zone files. On a secondary DNS server with type slave, add a “masters” line, with the IP address of the master in a similarly formatted set of curly brackets.

Assessment:

This solution has been extremely reliable for us. All of it's downtime has been attributed to either the backup/patching routine in single-user mode, or to administrator error (don't mistakenly remove a semi-colon from the required entries at the top of your zone files). It didn't receive a single suggestion for improvement through an internal security review conducted by an outside auditor, even with a Symantec Enterprise Security Manager agent evaluating it from the inside. This configuration of Solaris has also been very good to us, in that it's both secure and manageable.

I would like to add that the security of this configuration was reduced when we gave Microsoft's ADS domain controllers permission to update the forward zone file. This is currently the weakest link in our DNS infrastructure, as an error with the domain controllers or an intruder on the domain controllers could wipe out our forward DNS resolution in a way that would propogate to even the secondaries, almost immediately.

References:

Security Focus, Vulnerabilities, 2005
URL: http://www.securityfocus.org/bid/vendor/
Select Vendor ISC, Title BIND, Version 9.2.2

Boran, Sean, Running the BIND9 DNS Server securely, Boran Consulting, 2004
URL: http://www.boran.com/security/sp/bind9_20010430.html
§URL: http://www.boran.com/security/sp/bind/dns
URL: http://www.boran.com/security/sp/bind/named.conf.secondary

Liu, Cricket, DNS & BIND Cookbook, O'Reilly, 1st Edition, October 2002
URL: http://www.oreilly.com/catalog/dnsbindckbk/
URL: http://www.oreilly.com/catalog/dnsbindckbk/chapter/ch02.pdf

Steven M. Christensen and Associates, Inc., Installing OpenSSH Packages – Sparc and Intel/Solaris 8, 2005
URL: http://www.sunfreeware.com/openssh8.html
URL: http://www.sunfreeware.com/README.tcpwrappers

Center for Internet Security
URL: http://www.cisecurity.org/bench_solaris.html

SunSolve Online, Sun Microsystems, 2005
URL: http://au.sunsolve.sun.com/pub-cgi/show.pl?target=home

Nominum for Internet Software Consortium, ISC BIND 9 Administrator Manual, 2001
URL: http://www.nominum.com/content/documents/bind9arm.pdf

Nemeth, Evi, et.al, Unix System Administration Handbook (3rd Edition), Prentice Hall PTR, 3rd edition (August, 2000), Upper Saddle River NJ, 07458

Spitzner, Lance, Armoring Solaris
URL: http://www.spitzner.net/armoring.html

Spitzner, Lance, Armoring Solaris: II
URL: http://www.spitzner.net/armoring2.html

Bailey, Scott A.

Richer, Jacques P.