Securing Control Consoles and Clients
Ravila Helen White
June 17, 2001

Introduction

This paper will outline the process that may be used to provide a greater level of security to the various computers that act as control consoles to security devices (e.g. firewalls, tripwire, IDS etc). Implementing and deploying a strong infrastructure is futile when console/terminal devices controlling these devices can be accessed with little effort whether the intent is curious or malicious. The same is true of clients that are allowed to connect to control consoles. A secure control console and client, coupled with a secure infrastructure is an example of the concept of Defense in Depth.

Defense in Depth is the practice of layering multiple security measures to build a secure environment. Application of the Defense in Depth methodology is inherent to building a complete security infrastructure.

This paper is should not be regarded as a "How-To" guide but rather a source of information to provoke thought and innovation to the reader. The information contain herein can been applied in a business setting as well as at home.

The goal of this paper is:

- Goal 1 – Expose vulnerabilities in control consoles
- Goal 2 – Expose vulnerabilities in clients connecting to control consoles
- Goal 3 – Examine tools that will secure both your control console and client
- Goal 4 – Encourage the reader to combine a variety of security measures

The following assumptions will be made in order to provide a well-rounded paper.

- Assumption 1 – the reader has a basic understanding of network topology
- Assumption 2 – the network topology support multiple subnets
- Assumption 3 – remote access is controlled via SSH
- Assumption 3 – the environment is mixed (e.g. *nix and WinX)
- Assumption 4 – the media used for installation is trusted
- Assumption 5 – there is a basic security policy in place

This paper was written with the following constraints:

- Constraint 1 – References to SSH refer to SSH1 under OpenSSH and Cygwin
- Constraint 2 – Mention of *nix file locations refers to Solaris
- Constraint 3 – Mention of Windows refers to Microsoft Windows NT 4

Table of Contents

1 What Lies Beneath (securing the OS)

The system you choose as your control console for the management of security devices (e.g. Firewall, IDS or Tripwire) should be secured as well as your security devices. Just as you hardened the various security devices in the infrastructure so to should your control console be hardened according to their role as well as their placement in the network. Any host or client, which is allowed to connect to your device, must be considered a risk and will become a risk when not properly secured. The first step to take is host hardening. In this case the host will be the server control console.

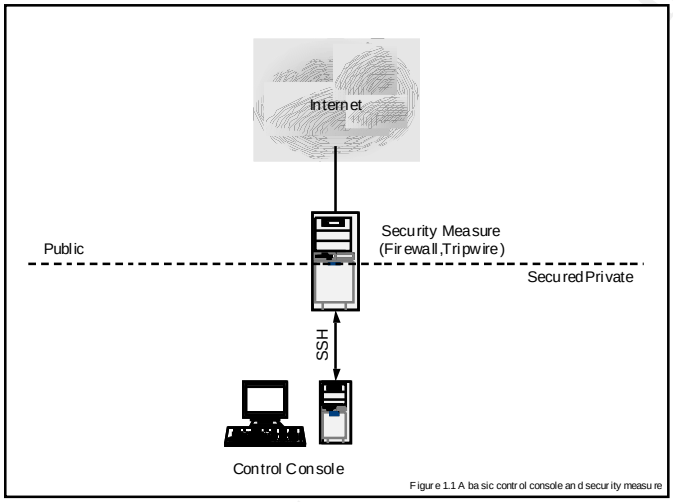

Figure 1.1 A basic control console and security measure

**Note: Multiple interfaces are not illustrated in the drawing and should be assumed.*

1.1 NT Server Hardening

Much of the security hardening of NT can be simplified through the use of the Microsoft Management Console (MMC) snap-in Security Configuration Manager (SCM). The SCM provides for the creation of security configuration templates, the comparison of the local machines settings against a trusted template and for the configuration of the local machine's settings to match the trusted template. The SCM comes with a predefine set of templates that can be modified for compliance to a organizations security policy. What ever the method used the following steps should be taken to secure the NT OS against exploitation or catastrophic failure:

1.1.1 Physical Measures

1.1.1.1 Physically secure the server - place in a access controlled area

1.1.1.2 Implement sequential booting of devices and password protect the BIOS password (if there is not a secured area to place your server you may consider disabling CD-ROM and floppy drives.

1.1.1.3 Verify that system can be recovered in the event of a disaster (e.g. backup tapes or hot-swap equipment).

1.1.2 File System Considerations

1.1.2.1 Format the drives as NTFS rather than FAT and create a system and data partition. Fat cannot be used to NTFS security.

1.1.2.2 Install as Member Server rather than Domain Controller
1.1.2.3 Create an Emergency Repair Disk (ERD) and store in a safe location as the SAM can be pulled from ERDs and password crackers run.

1.1.3 Adjusting Registry Keys

1.1.3.1 Hide drive letters
1.1.3.2 Enforce Strong Passwords
1.1.3.3 Remove unnecessary global domain groups from servers local groups
1.1.3.4 Remove Netware authentication
1.1.3.5 Remove print drivers
1.1.3.6 Enable auditing of backs and restores (optional depending on build process)
1.1.3.7 Disable anonymous logon
1.1.3.8 Disable remote access to registry
1.1.3.9 Disable remote access by anonymous to registry and named pipes
1.1.3.10 Restrict command scheduler access to authorized local users and groups only (no domain users or groups)
1.1.3.11 Secure the registry (the SCM will help with this)
1.1.3.12 Disable 8.3 file naming convention

1.1.4 Strong Passwords and Secure Account Policies

1.1.4.1 Change Administrator account name
1.1.4.2 Establish separate accounts for users with administrator privileges
1.1.4.3 Enable account lockout after specified number of bad password attempts has been reached
1.1.4.4 Secure administrator password (try and PCP encrypted file)
1.1.4.5 Remove the Everyone group and disable the guest account
1.1.4.6 Set up appropriate service accounts rather than use administrator to run services
1.1.4.7 Secure and review Event logs
1.1.4.8 Do not allow shared accounts
1.1.4.9 Encrypt the SAM password database with 128-bit encryption (this will be a security console and require high security)

1.1.5 Auditing

1.1.5.1 Turn on appropriate auditing
1.1.5.2 Monitor the audit logs

1.1.6 Networking and Internet Security Settings

1.1.6.1 Remove unnecessary protocols
1.1.6.2 Disable unnecessary services (e.g. FTP, RAS etc)
1.1.6.3 Configure Hosts file with a list of trusted hosts (optional)
1.1.6.4 Configure your hst for default routes (where applicable)
1.1.6.5 Protect vulnerable ports through a firewall or TCP Wrappers
1.1.6.6 If the security device requires a web interface, secure the web server according to vendor recommendations.

1.1.6.7 Restrict access to ports via the Advanced Security Configuration

1.1.7 Miscellaneous Suggestions

1.1.7.1 Install virus protection

1.1.7.2 Check and remove Rollback

1.1.7.3 Run penetration and vulnerability assessment against build

1.2 **Unix Hardening (Solaris)**

Hardening of Solaris can be accomplished through the use of custom scripts or through the scripts provided by SANs (YASSP) and SUN (JASS). There are other well respected works as well which may be referenced and used. Below are the basic steps toward building a secure Solaris 2.6 server. We will not cover the install of SSH as this will be covered in a separate section. However SSH can be installed when running YASSP:

1.2.1 *Installation Preparation*

1.2.1.1 Determine what applications and services will be supported.

1.2.1.2 Do not try to configure your server to control all the security devices in your environment

1.2.1.3 Remove keyboard, screen and framebuffer when appropriate

1.2.1.4 Download software from trusted sources and verify trust has not been broken

1.2.1.5 Disconnect from network (software can be configured via the console)

1.2.1.6 Make copies of all media and software and store in a safe place

1.2.2 *Installing the OS*

1.2.2.1 Partition disk manually

1.2.2.2 Set a strong password for root

1.2.2.3 Create accounts for applications and strong passwords for these accounts

1.2.2.4 Install SSH

1.2.2.5 Install VNC (please reference Section 3 for more information)

1.2.2.6 Apply recommended patches

1.2.3 *Run Hardening Script of Choice*

1.2.4 *Adjust Disk Mounting Options*

1.2.4.1 Allow /tmp to use 100mb of swap space

1.2.4.2 Disable execution

1.2.5 *Enable trusted services used by system and application*

1.2.6 *Configure the Hosts file with a list of trusted hosts (optional)*

1.2.7 *Configure your host for default routes (where applicable)*

1.2.8 *Install TCP wrappers*

1.2.9 *Install Tools and Scripts from a trusted source*

1.2.10 *Install Security scripts*

1.2.11 *Configure Logging*

1.2.12 *Tighten File Permissions*

1.2.13 *Edit login Banners (need for prosecution)*

1.2.14 Run penetration and vulnerability assessment against build

Mentioned in one of the hardening steps is configuring trusted routes. Where this is implemented throughout your security chain, greater security is insure. To illustrated consider the figure 1.2 below (partial routing table). Where the security device has multiple interfaces. One interface is connected to the Internet, while interfaces A, S and B are to be used to route to private internal networks. The A interface is all mail related services. The S interface will route to a secure network and interface B will route all other traffic destined for the corporate network. Note the route to the control host is explicit to the host rather than the network. This serves a choke point to insure that routing to a particular host is enforced rather than routing to an entire subnet. The control console may route explicitly to the security device and to the secure corporate subnet.

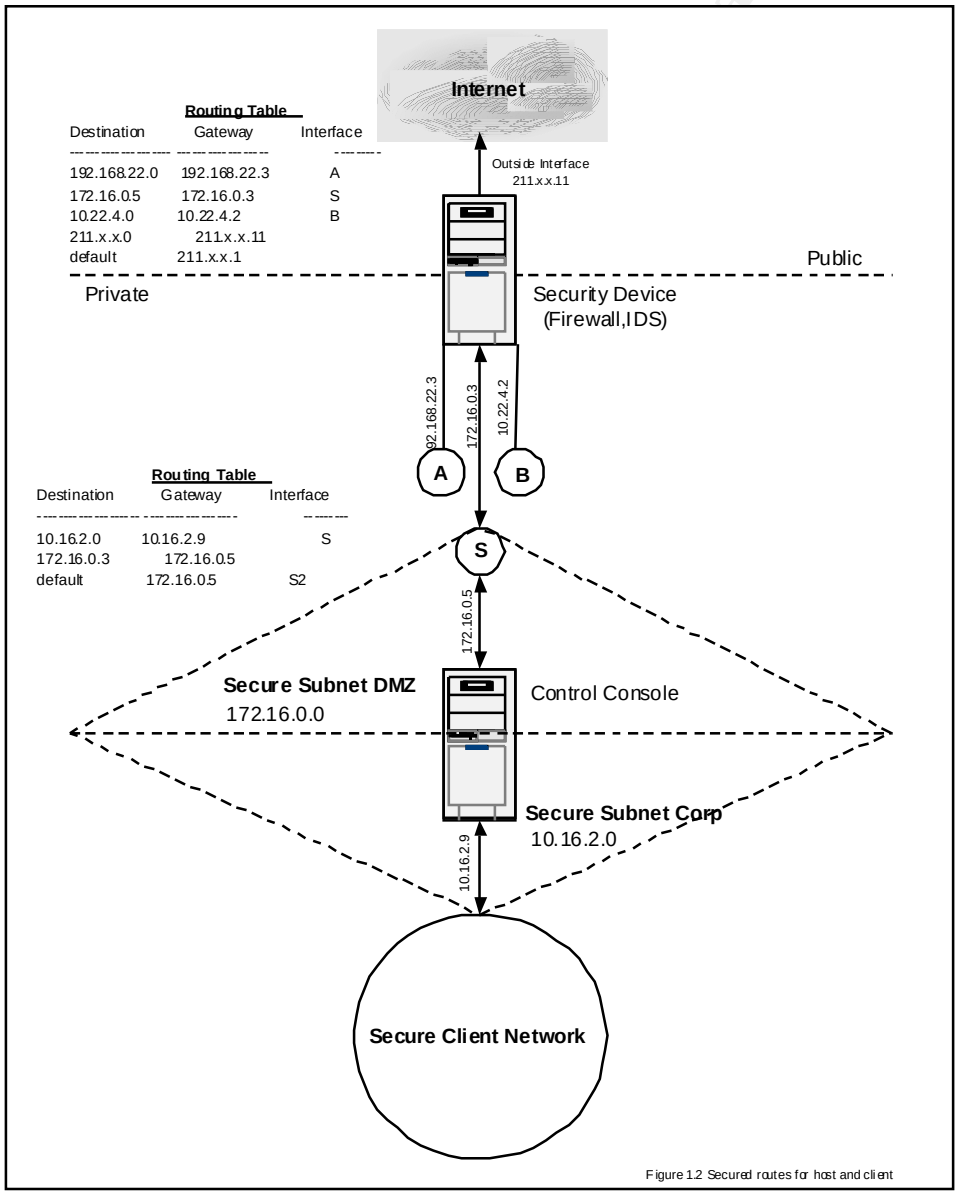

Figure 1.2 Secured routes for host and client

2 Securing the Display Terminal

Just as it is of great importance to secure your OS, so to is it important to secure the display terminal of your security control console. While enabling a locking screensaver is a start, there are other measures that must be taken to properly secure a display terminal. A locking screensaver will not prevent someone from taking control of an unsecured display and sniffing keystrokes for passwords, disabling applications, reading data which has been written to the screen or taking control of your session. The is particular of concern if you are the sole administrator of such a system as all damage resulting from such an intrusion will lay with you.

Perhaps at greatest risk are those systems that run X Windows. Windows NT to has vulnerabilities as well however the lack built-in client server support reduces the vulnerabilities associated with X. Below are steps to take to secure your terminal when running X Windows and Microsoft Windows NT server.

2.1 Securing X Windows

X Windows refers to the library of graphic routines developed at MIT in the 1980s that allows for the creation of a Graphical Users Interface (GUI). As opposed to Microsoft Windows and Apple Macintosh, X Windows is not hardware specific. While most implementations are seen on Unix platforms, X Windows also operates on Microsoft Windows NT, and VMS. Beyond X windows portability, it offers a client server architecture. This very strength when poorly configured will become vulnerable for exploitation. When poorly configured an X Windows servers windows can be dumped and then fed into a program which reads the contents of dump. When we understand that the windows of an X Windows server can be dumped we can arrive at the conclusion that keystrokes as well can be captured and keystrokes can be sent to the X Window server. Obviously, this would be catastrophic and probably embarrassing as well, to secure the OS of a security control host only to have it compromised through the display. A bit of time, care and consideration will reduce this type of risk. An understanding of X windows access methods should be obtained to insure that only trusted clients are able to access the display console. X Windows supports two types of authentication, host and token-based authentication.

2.1.2 *Host Based Authentication*

Host based authentication is configured through the xhost program. Through xhost a list of trusted hosts is maintained. If a client PC is listed amongst the trusted hosts then anyone with access to that particular PC may access your remote host. Host access is based on ip address or a host name depending on the configuration of your host and network. One might immediately jump to the conclusion that a trust client host could be spoofed thereby allowing unauthorized access to a host. While this could be a likely scenario there is a much simpler way to gain access. As stated before once a client has been authorized through xhost anyone on the client may access the host.

Now only the client has to be compromised to gain access. Xhost itself is not considered a privileged program, meaning that anyone who has access to the host may add or delete the entries held by xhost. While xhost may be tempting to use to control access it much to permissive to be trusted when the goal is securing a host that controls the security or an organization. It would be wise and advisable to simply disable xhost access completely on hosts that serve as security consoles. At least there is one less vulnerability to be considered. As xhost has been disabled this takes us to token based authentication.

2.1.1 *Token Based Authentication*

Magic Cookies on the other hand does provide a measure of control through the use of token-based authentication. Depending on the host configuration, a unique token is created at logon and stored in the .Xauthority file. Where automatic generation has not been configured, the user can provide the host with the necessary information to manually generate the Magic Cookie. Whatever the method, once the token has been generated it is stored in the .Xauthority file. The token within the. Xauthority file is machine-readable and sense can only be made of it through the use of the xauth program. While the use of MIT-MAGIC COOKIES is advisable they can be compromised through sniffing techniques during the transfer of the Magic Cookie to the trusted client. To alleviate this vulnerability, the xhost should be configured to generate a new Magic Cookie each time a client connects to the xserver. The transfer of the cookie can be handled via the SCP program in SSH which will be discussed later in this paper.

2.1.2 *Other Considerations for X*

One of the possible threats identified with X Windows and its components is keyboard sniffing. To mitigate this use of the Secure Keyboard option should be employed. This will prevent the capture of keyboard strokes from other clients.

It stands to reason that if the goal is one of high security access to the console will be granted via a token based method as discussed earlier. This would indicate that the use of and any reference to xhost should be disabled or removed. In particular all xhost commands should be removed from the .Xsession file. Finally a search through applications or shell scripts should be performed to identify any reference to xhost. These references should be removed. All options in the xdm-config file should be considered. Those that are not useful in a secure environment should be commented out and options that will increase security should be adjusted with the proper values. Finally the xdm-config file should be secured from all but the root account and where a tripwire is available tripwire the file.

2.1.3 *Securing CDE*

In the event your Unix control console is running CDE steps should be taken to secure this environment as well. The standard greeting banner should be modified to reflect a warning. Additionally, XCMCP connections should be disabled as well. Finally, permissions on the Xaccess file should be set to 444. This will render the file read only. This is file that should be set on a tripwire as well if tripwire will be installed.

With respect to a console which actually has a display terminal attached a locking screensaver should be enabled. Remote access to the CDE as user root should also be disabled.

2.2 **Securing a Microsoft Windows Display Terminal**

The methods employed to secure a Microsoft Windows display terminal is much simpler as the Windows architecture does not support a client/server architecture. To secure your Windows display terminal the following should be done

2.2.1 Disable display of last logged on user to protect user Ids

2.2.2 Disable caching of logon information. This should not cause a problem with domain authentication as the console should be a member server and local accounts will be used for authentication.

2.2.3 Implementation of logon message banner to warn potential intruders of risk.

2.2.4 Restrict console logons to allow only those who will need access, meaning remove all domain groups and users and rely upon the local users and groups

2.2.5 *Enable a password-protected screensaver to a low time-out value.*

3 Tools That Support Secure Remote Consoles

Now that we've chocked down access to our security console consideration on accessing the environment will be given. In most cases we will need to have gui access, command line access, file transfer capabilities and mail. Two tools which will allow us to attain our aforementioned goal are SSH and VNC. The nice thing about both of these tools is that there are open source (public domain) offerings for Windows and Unix platforms.

3.1 SSH (Secure Shell)

SSH is the conduit through which one can connect to a security host in a secure manner. Section two of this paper recommended removal of unsafe services. Security professionals usually understand this to mean TELNET, FTP, rlogin or any service permits remote connections. SSH solves the problem of not having these services enabled and accessible on our security host.

While transparent in nature, When a client initiates a ssh session/connection their user information is encrypted before it leaves the client PC. It is decrypted

by the destination SSH server host for verification. The encryption and decryption of data across a network protects against network (username and password) sniffing and Man-in-the-Middle attacks. Installing SSH is quite simple. It is the configuration of SSH and in particular SSHD (OpenSSH and Cygwin SSH), which will determine the security of a security host, as SSH will be the portal to your security host, which in turn is the controller of your security measures.

The SSH server can be configured through one of three methods: Server-wide Configuration, compile-time configuration and per-account configuration. More than likely on a machine that requires high security, either Server-wide or compile-time configurations will be chosen rather than per-account configuration. Why? Remembering that our ultimate goal is security and even a bit of simplicity we understand the per-account configuration allows the connecting user control over the behavior of their SSH session. A well intended but less experienced user or even administrator may not make the best choice regarding who or what my have access to the SSH server through their account. Whichever method is chosen, the pre-configured defaults should be adjusted (e.g. listenaddress) to compliment the security of your environment. This will also insure that would-be intruders will be met with a higher degree of difficulty when attempting to gather information when attempting to exploit and abuse a host system.

The SSH server reads the sshd_config (ssh_config2 for SSH2) file to determine its configuration. This file should be restricted to read only access. Once a reasonable configuration has been determined it would be wise to copy this file for safe keeping to trusted location. In the event of a disaster there would be one less thing to consider. Additionally, it is advisable to create different configuration files based on the host role (e.g. the sshd_config file for your firewalls will not allow agent forwarding whereas it may be allowed on a host which is the control console). As already mention, the first rule to remember is do not accept the defaults found in the sshd_config file. The keywords below correspond the ssh implementation of OpenSSH or CygwinSSH.

3.1.1 SSHD_Config (value is in bold and reference OpenSSH1)

- 3.1.1.1 AllowGroups Group list (Access is controlled by Unix group)
- 3.1.1.2 AllowTcpForwarding Yes (Enable TCP port forwarding)
- 3.1.1.3 AllowUsers User List (Access is controlled by username)
- 3.1.1.4 CheckMail Yes (Check new mail on login)
- 3.1.1.5 DenyGroups Group List (Access is controlled by Unix group)
- 3.1.1.6 Deny Users User List(Access is controlled by username)
- 3.1.1.7 HostKey /etc/ssh_host_key (location of the private host key)
- 3.1.1.8 IgnoreRhosts (Ignore. rhosts files)
- 3.1.1.9 KeepAlive Yes (Send keepalive packets)

3.1.1.10 KerberosAuthentication No (Permit Kerberos authentication)
3.1.1.11 KerberosOrLocalPasswd No (Kerberos fallback authentication)
3.1.1.12 KerberosTgtPassing No (Support ticket-granting-tickets)
3.1.1.13 KeyRegenerationInterval 3600 (how log in seconds a server should wait before automatically regenerating its key)
3.1.1.14 ListenAddress 0.0.0.0(restricts what interface ssh listens on)
3.1.1.15 LoginGraceTime 30 (time user is allowed to enter name and password)
3.1.1.16 LogLevel Syslog level (Set syslog level)
3.1.1.17 PasswordAuthentication yes (Permit password authentication)
3.1.1.18 PermitEmptyPasswords No (Permit empty passwords)
3.1.1.19 PermitRootLogin No (allows SSH login to via root account)
3.1.1.20 PidFile (Location of pid file)
3.1.1.21 Port 22 (this is port which the SSH server will listen on)
3.1.1.22 PrintMotd Yes (Print message of the day)
3.1.1.23 RhostsAuthentication No (Permit .rhosts authentication)
3.1.1.24 RhostsRSAAuthentication No (Permit combined authentication
3.1.1.25 RSAAuthentication No (Permit public-key authentication)
3.1.1.26 ServerKeyBits 768 (512-1024 amount of bits to use in server key)
3.1.1.27 StrictModes no (Strict file/directory permissions)
3.1.1.28 SyslogFacility Syslog level (Set syslog level)
3.1.1.29 UserLogin No (select login program)
3.1.1.30 X11Forwarding No (Enable X forwarding)
3.1.1.31 X11DisplayOffset 10 (Limit X display for SSH)
3.1.1.32 XauthLocation Filename (Location of xauth)

3.1.2 Authentication

Authentication can be based on the cryptographic keys generated by ssh-keygen, local OS username or a third party application. The keys generated by ssh-keygen are either DSA or RSA and the length is determined by the keyword value of ServerKeyBits. Where the keyword RSAAuthentication is found in sshd_config then a RSA must be generated. The same if true of DSA. Rhost authentication is dependant upon the information found in the rhost file. For this feature to work the r-commands must be enabled, which would be counterproductive on a secure host. Kerbos authentication is available when compile-time configuration. Kerbos authentication can be combined with Passwordauthentication to form the keyword KerberosOrLocalPasswd. This provides a fallback for users when either Kerberos cannot authenticate them or they are not using Kerberos. Where the goal is security through obscurity, SkeyAuthentication is the best choice. This method enforces the use of

one-time passwords. Inputing three sequence and key numbers into the s/key calculator generates the password. When the goal is complete security the options should be limited to few and used by all who connect. When multiple methods are chosen the user can configure his/her client to determine the authentication. Limits on supported authentication will enforce how a client authenticates.

3.1.3 The sshd_config file need not contain all of the above keywords to secure your SSH server. This lengthy list does represent how highly configurable and flexible SSH implementation can be for environments. When the *nix host you are configuring contains multiple network interfaces you can limit the access of SSH to those interfaces which you consider trusted with the ListenAddress value (e.g. don't allow connections on interfaces exposed to the internet). Furthermore, changing the port the SSH server listens on will rather than accepting the default port 22 will further obscure how the host has been configured. While will this will not defeat an aggressive NMAP scan it does deter the random script kiddie from obtaining SSH information by telneting to the default tcp port 22. Changing defaults serves to tighten the security of your host deter those individuals who aren't interested in hardcore hacking/cracking. Also worthy of consideration are the keyword options of AllowUsers and PrintMotd. Where the host is controlled by a select group of people it is highly advisable to choke access to SSH by limiting who can connect. This also serves to make those who are allowed access responsibility for securing their accounts and workstations. Some may consider including PrintMotd a waste of time. However, if this is the only way to access the server then we must consider the possibility that it could be compromised. When the host has been configured with Motd or a banner message displaying during the SSH login, this will serve as warning if the system should be compromised and litigation is pursued.

Where SSH is implemented on a Windows NT server notice must be taken with regard to how authentication will be handled. The Cygwin SSHD server and utilities allow for authentication from a Windows domain and the local server or Cygwin. It may be very tempting to allow authentication through the domain. However, for this to take place the host PC must participate in browsing and net bios communications to allow for interaction with the DCs of the domain. This means that anyone accessing the host could enumerate all domain accounts through the use of the mkpasswd utility included in the release of Cygwin. If your NT host is to be truly hardened, extra time and reasonable consideration of how authentication will be accomplish must be taken.

Whichever keyword options are chosen. A thorough understanding of these options should be considered. As new technology becomes available a periodic review of your sshd_config file should be performed and adjustments made. Finally, before implementing SSH understanding the pros and cons of SSH1 and SSH2 must be understood. SSH2 has a number of offerings that can ease the task of authentication such as PGP. Hopefully support of PGP will be incorporated in the OpenSSH distribution at some point.

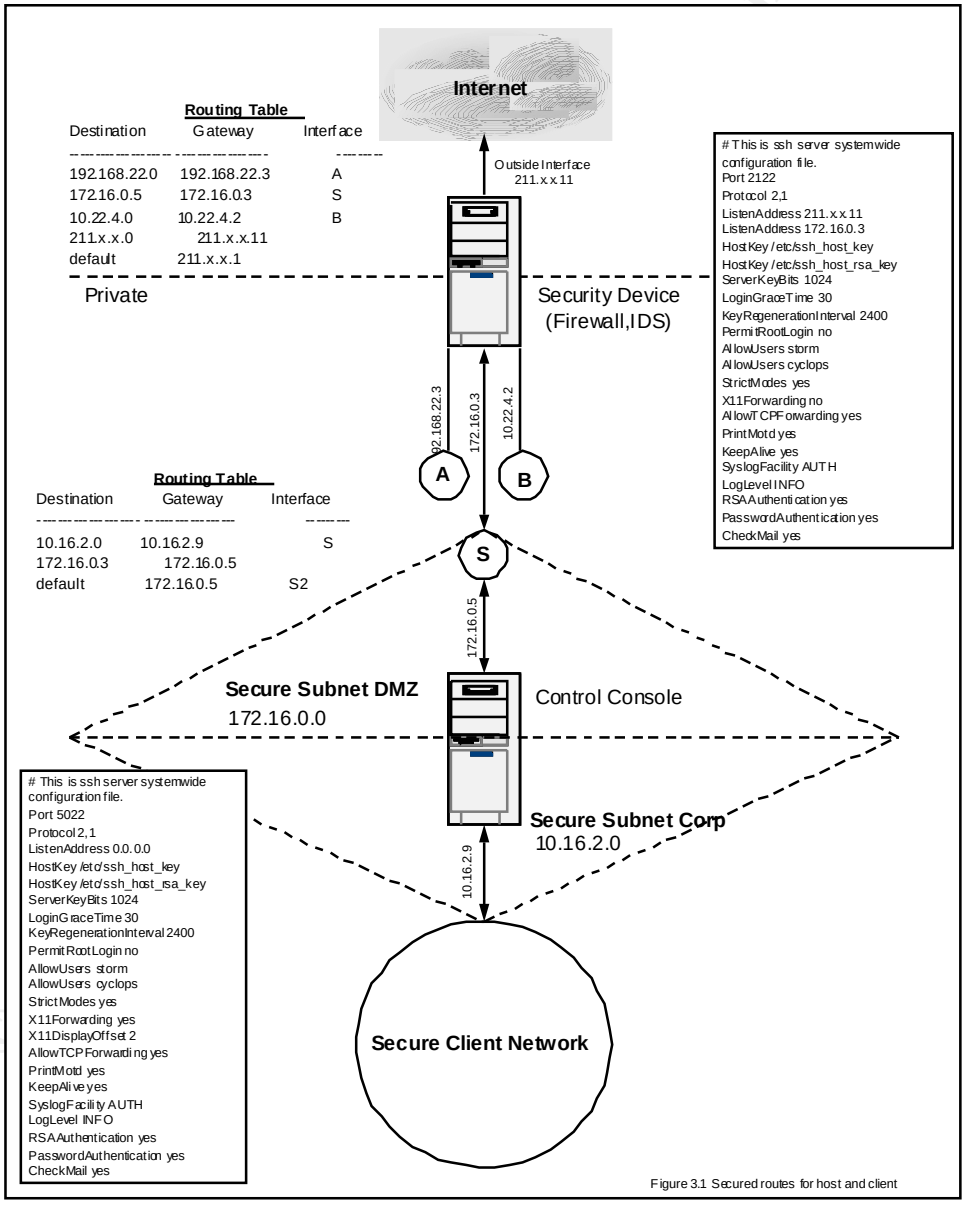

Figure 3.1 Secured routes for host and client

The above diagram (figure 3.1) shows how the restrictions placed on routing as well as the SSHD configuration reduces the level of vulnerability to the security measure as well as the control console. SSH

does not listen on the default port and only permit SSH client connections on the interfaces where access is a must. The decision to enable forwarding of TCP packets should be considered and implemented with great caution. TCP forwarding has been allowed in this case because only trusted networks or hosts are allowed access to the SSH port. On a firewall where SSH has been allowed for all outgoing connections TCP forwarding could compromised the network where a user understands how to tunnel a disallowed service or protocol through SSH. This is a scenario that can turn ugly should the destination tunnel be that of a trojaned host. The explicit use of user names is debatable. However it does insure that only trusted users may connect and where the trusted user may need emergency access they may obtain access through verification of their user name. Where this is deemed necessary, a generic account using S/key authentication may be the best approach. S/key generation will not work well on a host that is configured to relay mail alerts to the control console. In this case an account for the mail relay will be created. To avoid the use of PermitEmptyPasswd the user account should be configured to use the ssh-agent. The ssh-agent will save private keys and eliminate password phrase prompting. The sshd_config files above are but a small representation of the flexibility and strength of SSH. Use of all features may never occur, however a security professional who wants to reduce the risk of exploitation, risk and cost would do well to familiarize themselves with all features the various implementations SSH has to offer.

During our console/terminal hardening access to the X Windows environment was disabled. The tool which can be used to support the local Windows environment of both *nix and Windows NT is VNC.

3.2 VNC

VNC is a client/server application that provides the display of a remote desktop. Use of VNC has become quite prolific due to its flexibility, ease of use and 0$ cost. In an environment where support of Microsoft Windows desktops as well as *nix desktops is essential VNC can ease the cost of administration. While this papers primary focus and references are related to Microsoft Windows NT 4.0 and Solaris it should be noted that VNC is supported the following platforms: Macintosh, DEC Alpha, Linux and Windows CE 2.x.

Configuration of VNC is very simple. A password to protect the VNC server should be enabled. While this may seem sufficient where VNC is installed for use on a security control console, strong recommendations of tunneling VNC over SSH is advisable. Why? VNC authentication is based on a challenge response scheme. This is the only part of the VNC communication that is encrypted. The rest of the session is not, which can open the path to session hi-jacking or man-in-the-middle attacks. Where VNC is not tunneled and you are connected to a control console and then ssh to a remote security device, the opportunity to sniff the security devices password has just been opened.

Tunneling VNC insures that privileged communications and data will remain secure.

Further security for VNC can be accomplished through integration of TCP Wrappers. Where the host resides on a non-secure network (e.g. one where there are no routing restrictions or segmenting etc) this can insure that only those clients who are trusted may access the VNC server. When the host resides on a secure network it is still advisable to integrate TCP wrappers to guard against a breach should someone manage to access that network and began attempting the exploit the hosts of that network.

Recommendation has been made to avoid default setting found in the various software distributions we rely upon. The same is true of the VNC listening port 5901 or 5902. Changing the listening port will dissuade casual or novice abusers. Configuring VNC to listen on a random port (e.g. 7944) we can then issue the command (figure 3.2 below) to encrypted our VNC session.

```
ssh -L 6622:10.16.2.9:7944
```

Figure 3.2 Setting up portforwarding on SSH client

This command initiates the ssh session and tells the client to listen explicitly on port 6622 of the client then connect to the ssh host of 10.16.2.9 where the VNC server is listening on port 7944. Once the ssh connection has been made VNC viewer can be launch (figure 3.3). As the local host is listening on port 6622 VNC viewer is configured to connect to our local host. There now exist a display or the remote host on the client display encrypted by ssh.

```
vncviewer localhost:22
```

Figure 3.3 Connect to tunneled VNC server

3.3 TCP Wrappers

Just as SSH and VNC are based on a client server architecture, so to is the concept of TCP Wrappers. TCP Wrappers acts as an arbiter between a host offering services and the client who is requesting access to those services. Depending on the credentials of the client access will either be permitted or denied. Where a client is permitted access to a particular service wrapper launches the application called. The initial response seen from the client side is the wrapper responding not the actually service. Whether a client is denied access or granted access this information is logged by TCP Wrappers which in turn can be used when assessing hacker/cracker activity. The information provided during logging results from a reverse finger tool. The reverse finger is

initiated during the initial client call and logs the time stamp, the name of the local host, the name of the requested service (the network server process name), and the name of the remote host.

While it may seem quite simple to integrate TCP Wrappers on a control console running minimal services, a thoughtful implementation will include Wrappers that act as honey pots. The wrapper honey pots might be configured for those services that are highly exploitable (e.g. telnet, ftp, rlogin etc). Enabling wrappers for these services would expose a reconnaissance probe that would otherwise be missed if we only monitored SSH (applies if compile-time configured) and VNC connection activity.

Mention was made earlier of TCP Wrappers and logging. The logging information is sent to the syslog daemon. Depending on the configuration of syslogd messages will be written to the console, a file or forwarded to a log host. All connections should be logged. However, it is advisable to specify a priority of urgent to those attempts that are caught in the honey pot and untrusted clients who are not permitted access.

Just as SSH and VNC are open to vulnerabilities so to is TCP Wrappers. If sources IP and host name are spoofed, TCP Wrappers will not detect this action. Additionally, once a connection is made to a particular service TCP Wrappers will not examine future requests to the same service. The latter risk can be fairly low when the user base is limited and the host is located on a somewhat secure network.

4 You ARE the Weakest Link

At this point some may assume that their control console is fairly secure and the degree of risk has been reduced. This is not true when we consider that our model is one of client server. This paper has examined at length how to secure a host. It is only logical to take the same care in securing the client(s) that will connect to our control host. After all, with the control host being adequately secured the next vulnerability would lie with those who are permitted access. Some of the same techniques that were used in the host hardening can be applied to client hardening. After all the work of configuring and installing a firewall or IDS host, then the control console it would be negligible for these devices to be exploited through an unsecured client.

4.1 Windows Client Hardening

4.1.1 Locking Screen Saver

4.1.2 Secure your client against remote access from other clients and hosts

4.1.3 Move to same network as control console

4.1.4 At a minimum install TCP Wrappers to alert you to reconnaissance efforts

4.1.5 If you have two NIC cards configure one for the secure network and the other for the corporate network. Then set explicit routing on the client

4.1.6 Configure your client with a static address rather than DHCP

4.2 <u>*nix Client Hardening</u>

4.2.1 Secure the X environment
4.2.2 Secure the desktop environment
4.2.3 Locking screen saver
4.2.4 Disable remote access
4.2.5 Move to same network as control console
4.2.6 At a minimum install TCP Wrappers to alert you to reconnaissance efforts
4.2.7 If you have two NIC cards configure one for the secure network and the other for the corporate network. Then set explicit routing on the client
4.2.8 Configure your client with a static address rather than DHCP

Summary

There is no Panacea where network security is involved. Careful thought combined with a layering of security products and the elimination of weak links will allow a security professional to build a secure and low risk environment. Implicit to this goal is and understanding more than one OS. Most of us have a favorite OS. However more value is added when traversing multiple Oses are a part of our skill set. A holistic approach to security will result in the practice of supporting the Defense in Depth concept and reducing the risk of exploitation regardless of the systems we are instructed to secure and protect.

Bibliography

Harold F.Tipton & Micki Krause – "Information Security Management Handbook, 4th Edition"

Arthur E. Hutt, Seymour Bosworth, & Douglas B.Hoyt – "Computer Security Handbook, Third Edition"

Daniel J. Barrett & Richard E. Silverman – "SSH, The Secure Shell – The Definitive Guide"

SANS Security Basics Curriculum – "Windows NT Security"

SANS Security Basics Curriculum – "Information Assurance Foundations"

WWW Sources

John Fisher – "Securing X Windows
http://ciac.llnl.gov/ciac/documents/ciac2316.html

Runeb – "Crash Course in X Windows Security"
http://the.wall.riscom.net/books/unix/rootsh/docs/X.security

Sean Boran – "IT Security Cookbook"
http://www.boran.com/security/it13-applications.html

"Securing Window NT – Microsoft"
http://www.microsoft.com/technet/security/tools.asp

Software and Script Sources

SSH for *nix
http://www.openssh.org/

SSH for NT
www.cygwin.com

VNC
http://www.uk.research.att.com/vnc/index.html

TCP Wrappers for Unix
ftp://ftp.porcupine.org/pub/security/

TCP Wrappers for NT
http://www.cotse.com/CotseLabs/winetd/

YASSP
http://www.yassp.org/tintro.html

JASSP
http://www.sun.com/blueprints/tools/;$sessionid$HIIBCPYAADPPXAMTA1LU5YQ

Vendor(s)