Virtual Network Computing and Secure Shell

Damian Koziel

Full Name: R. Damian Koziel
Version of Assignment: GIAC Level One Security Essentials 1.2e
Descriptive Title: Secure Shell Virtual Network Computing. A tutorial on the installation, configuration and use of Virtual Network Computing under the native and SSH protocols.

Virtual Network Computing and Secure Shell
R. Damian Koziel
July 20, 2001

Introduction

Computing professionals of yesteryear worked in an environment that was concentric in nature. They all arrived at the same building at roughly the same time, all using the same tools in the same computing environment. The system administrator at such a site was responsible for a clearly defined body of knowledge and had a consistent platform on which to work. Times have changed. What was a centralized environment of time-shared terminals to a mainframe on a single operating system has evolved into a heterogeneous mixture of UNIX and Windows workstations. It's not just the computing environment that has changed; the physical environment has changed as well. The growth of a high-tech company may quickly surpass the available space resulting in the need for additional buildings that may reside on a campus or even across town. With the advent of powerful laptop computers, it is now possible for many high-tech professionals to work from home increasing the system administrator's physical realm. How does today's system administrator face the challenges of maintaining and troubleshooting a company's heterogeneous and sprawling computing system from a central location? Enter Virtual Network Computing.

Virtual Network Computing: An Overview

Virtual Network Computing (VNC) is a process by which a system's desktop can not only be viewed but also engaged in an interactive session as well. The use of such a tool gives the system administrator the ability to administer and troubleshoot a system remotely. In this way, a target system on the next floor, the next building or even at an employee's home is within reach. There are other methods available of establishing this type of remote viewing. The obvious example is Symantec's *PCAnywhere* (http://www.symantec.com/pcanywhere/Consumer/). There are, however, notable differences:

- VNC is platform-independent. One can interact with a Solaris desktop from a Windows NT machine, or from a Linux machine, interact with a Windows NT desktop.

- VNC is available on a large number of platforms well beyond the popular choices of Linux, Solaris and Windows NT. A complete list of completed ports can be found at http://www.uk.research.att.com/vnc/platforms.html.

- It's small; the Windows VNC client is 172 kB; the UNIX version even smaller at 77 kB.

- It's available at no cost. It can be downloaded, used and redistributed under the terms of the GNU Public License.

This remote access to graphical user interfaces is accomplished through the use of the VNC protocol based on the concept of a *remote framebuffer* or *RFB*. It is beyond the scope of this

document to explore the details of this protocol. Inquisitive readers are referred to "The RFB Protocol" which can be obtained at http://www.uk.research.att.com/vnc/rfbproto.pdf. Briefly, because VNC operates at the framebuffer level, any device that will support TCP/IP will most likely work with VNC. This includes all operating systems, window systems and applications, including UNIX, Windows and Macintosh. When a VNC client-server connection is established, authentication is accomplished through a challenge-response technique that results, at the client end, in a password request. Once authentication is completed, framebuffer parameters such as desktop size and pixel format are exchanged and the client requests an update for the entire screen, thus beginning the session. This document will concern itself with the installation, configuration and use of the VNC client and server for the Windows NT 4 and Solaris 2.6 platforms.

Installation

The first step in the installation of VNC is to obtain the client and server software. Connecting to http://www.uk.research.att.com/vnc/download.html provides the opportunity to download the binaries or the sources. The page requests a name, e-mail address and organization be provided before a download can commence. From the list that follows, select the type of installation (binary/source) and the desired platform(s). The *Proceed to download* button will then allow the selection of a form of compression (GNU gzip, WinZip, UNIX compress) as well as the transfer mode (ftp/http). Installation instructions for both binary and source downloads follow.

The Windows binary installation is straightforward and mirrors a standard Windows program installation. After unpacking the compressed archive, from the winvnc folder run **setup.exe**. The installation places on *Program Files \ ORL \ VNC* the VNC executables as well as some supporting .DLL files.

The Solaris binary installation is not as automated but nonetheless straightforward. After unpacking the compressed tar archive, place the VNC programs in a standard location (e.g. /usr/local/bin) with the command **cp vncviewer vncserver vncpasswd vncconnect Xvnc <destinationDir>**. The reader is encouraged to consult the distribution's README for additional information that may prove relevant for their environment.

If security reasons preclude the installation of binaries not built locally on a site, then the source versions of VNC can be downloaded. The Windows source installation requires a C compiler. This document will describe the use of Microsoft Windows Visual C++ 5.0 though other Windows C compilers should work equally well. After unpacking the compressed archive, open the VNC client project workspace file ...\vnc-wincsrc\vncviewer*vncviewer.dsw*.

.

- From the **Build** menu, select **Batch Build** and clear all choices.
- **Enable** *omnithread – Win32 Release*
- **Enable** *vncviewer – Win32 Release*
- Click **Build**.

Building the Windows VNC server is similar. Open the VNC server project workspace file …\vnc-winsrc\winvnc*WinVNC.dsw*.

- From the **Build** menu, select **Batch Build** and clear all choices.
- **Enable** *VNCHooks – Win32 Release*
- **Enable** *WinVNC – Win32 No_CORBA*
- **Enable** *omnithread – Win32 No_CORBA*
- Click **Build.**

When the VNC server compilation is completed, place the following files into a new folder:

- …\vnc_winsrc\vncviewer\Release\vncviewer.exe
- …\vnc_winsrc\winvnc\No_CORBA\WinVNC.exe
- …\vnc_winsrc\winvnc\No_CORBA\omnithread_rt.dll
- …\vnc_winsrc\winvnc\No_CORBA\VNCHooks.dll

This assumes the existence of the .DLLs *msvcirt.*dll and *msvcrt.dll* normally found in %SystemRoot%\sytem32.

The Solaris source installation is packaged with Make and requires a C compiler as well. This document assumes the use of the GNU C Compiler that can be obtained at http://www.gnu.org/software/gcc/gcc.html, although the Sun Microsystems' C compiler should work equally well. After unpacking the compressed tar archive, the VNC client and server are built with the following commands:

- host% **/usr/openwin/bin/xview/xmkmf**
- host% **/usr/ccs/bin/make World**
- host% **/usr/bin/cd Xvnc**
- host% **/usr/ccs/bin/make World**
- host% **/usr/bin/cd ..**
- host% **./vncinstall /usr/local/bin**

Upon completion of the installation, either through compilation or binary download, the target machine now has the capability to act as a VNC server or view another remote desktop through the use of the VNC client. The UNIX and NT installations do not differ in their functionality but the layout, as one would expect, is slightly different.

The UNIX installation places into the specified destination directory the following files:

- vncviewer: X Client; Used to interact with the remote system.
- vncserver: Wrapper script that calls XVnc.
- vncpasswd: Allows the VNC access password to be set/changed.
- vncconnect: Allows a reverse connection by which a VNC server connects to a VNC client that is in listening mode.
- Xvnc: The actual X VNC server.

The NT installation differs from the UNIX installation in that the installed directory contains the VNC client and server executables and supporting .DLLs. The password function is accessible through the WinVNC properties sheet and the listen function is made available by invoking *vncviewer –listen*. The binary installation installs a VNC program group providing a method to invoke these programs through the Windows *Start Menu*.

Configuration and Use

The next step in the use of VNC is the configuration of the VNC server. The VNC NT client connection to a Solaris VNC server will be discussed first. As VNC is client-server based, a VNC server needs to be running on the target Solaris machine. As indicated above, *vncserver* is a wrapper script that calls the actual X VNC server. This script requires the presence of Perl on the machine. The installation of Perl will not be covered in this document and the reader is referred to http://www.perl.com/pub/a/language/info/software.html for information on the download and installation of Perl. Invoking *vncserver* for the first time on the target Solaris machine will prompt "You will require a password to access your desktops." The password, when confirmed, will be stored and called for by the VNC client when connecting to the VNC server. Note that the password is stored under ~/.vnc/passwd and is independent of the user account password (i.e. NIS) so paying special attention to the UNIX login used when starting the server initially or changing the password interactively will eliminate password related connection problems. The password can be changed at any time by invoking the *vncpasswd* command. When the VNC server is started, the files <hostname>:<displayNumber>.log and <hostname>:<displayNumber>.pid are created in ~/.vnc. UNIX displays are numbered, starting with the console, at zero. The standard VNC implementation does not provide for remote access to display:0 under UNIX though the reader is encouraged to consult the resources section of this document for information on enabling this capability. Like many UNIX servers, the process ID (PID) is stored in a file and VNC follows that convention. The .pid file is used to determine which process to kill when *vncserver –kill <hostname>:<displayNumber>* is issued thereby killing the VNC server. It is possible to start more than one VNC server and this will be reflected in the numbering of the .log files. The last remaining file in the ~/.vnc directory is *xstartup*. This contains the directives for the TWM (Tab Window Manager) that control the windows (size, color, placement) as well as mouse/keyboard controls. The topic of X window managers, of which there are several, is too large to be covered in this document and the reader is referred to the resources section of this document for referrals. Now that the Solaris VNC server is running, all that remains is to connect to it via the VNC client on the Windows machine. Running *vncviewer* will prompt for the host and display number on which to connect. In the example below, the VNC client will attempt a connection to *someHost* on display number one.

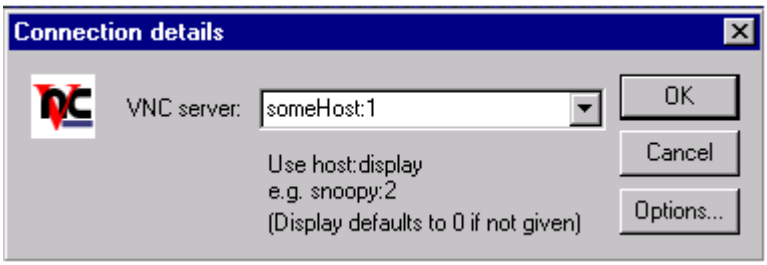

Figure 1

When the server is contacted, the password request is made as shown below.

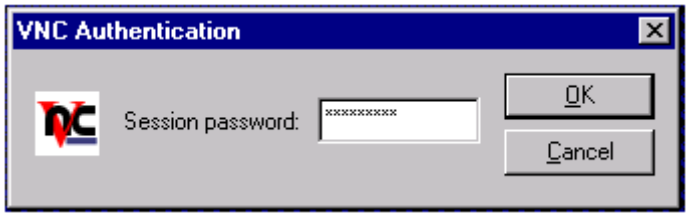

Figure 2

Once authenticated, the X window directives contained in ~/.vnc/xstartup are invoked and the VNC client window opens on the NT desktop as shown in Figure 3. This is a fully interactive login so any command or tool that can be run locally can be run under the VNC. For example, note that an *ls –lart /dev | more* is being executed in one window with *admintool* invoked in another.

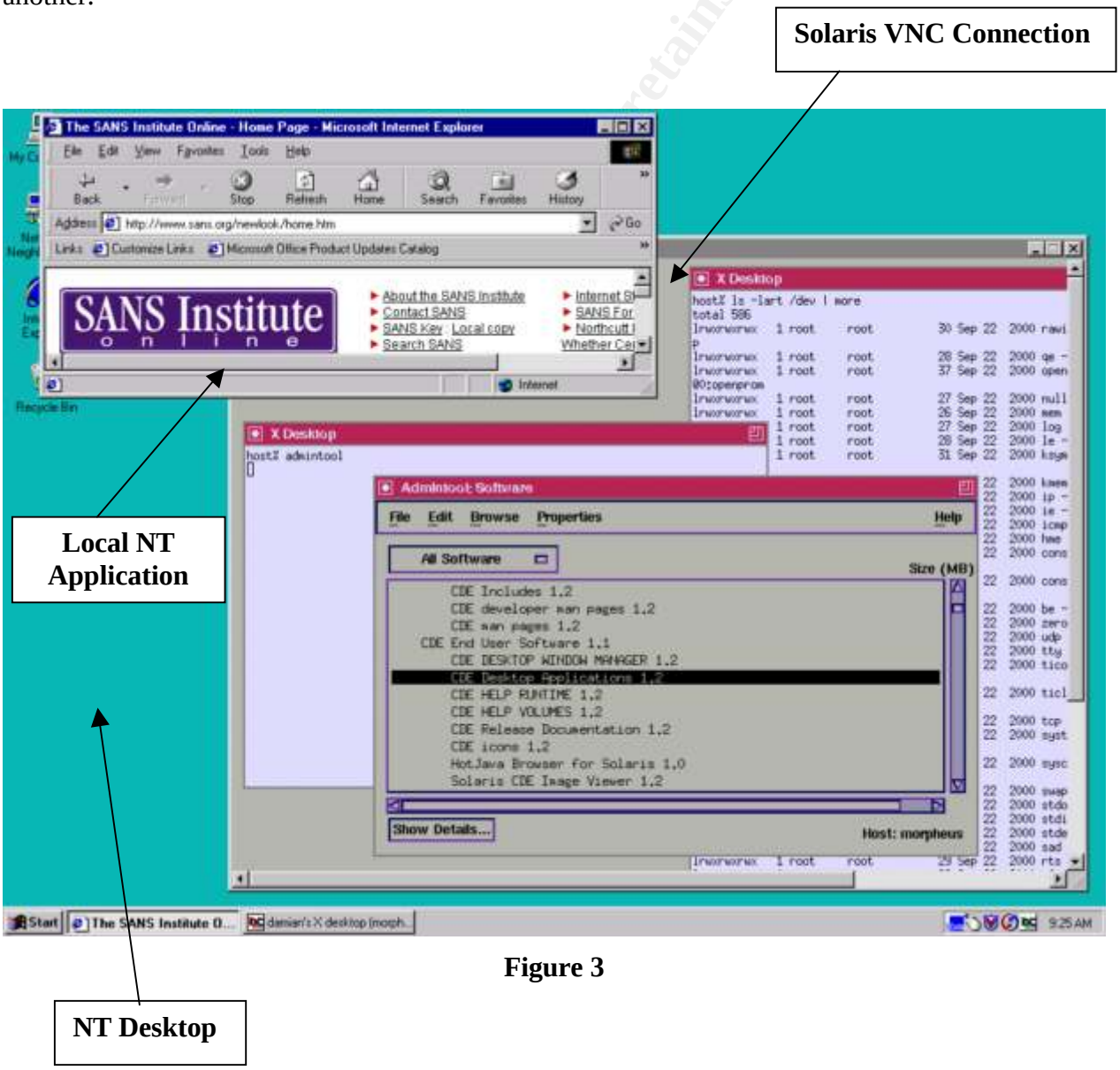

Figure 3

The VNC Solaris client connection to a NT VNC server works similarly in that the VNC server must be running on the target NT machine. To start the NT VNC server, run *WinVNC.exe* which will then appear as an icon in the system tray. Like the UNIX implementation, when run for the first time WinVNC demands that a password be set and will not allow incoming connections unless one is set. Also note that the VNC password is independent of the domain account password and is not stored on the domain controllers. Unlike UNIX, it is possible to connect to the Windows console so the first VNC connection is <hostname>:0. The server is shut down with the *Close VNC* option from WinVNC in the system tray.

Connecting to this NT VNC server from a Solaris VNC client is accomplished through invoking *vncviewer*. When prompted, enter the host and display number on which to connect and the password. Once authenticated, the VNC session begins as shown in Figure 4. This too is a fully interactive session, so any command or tool that can be run locally can be run under the VNC.

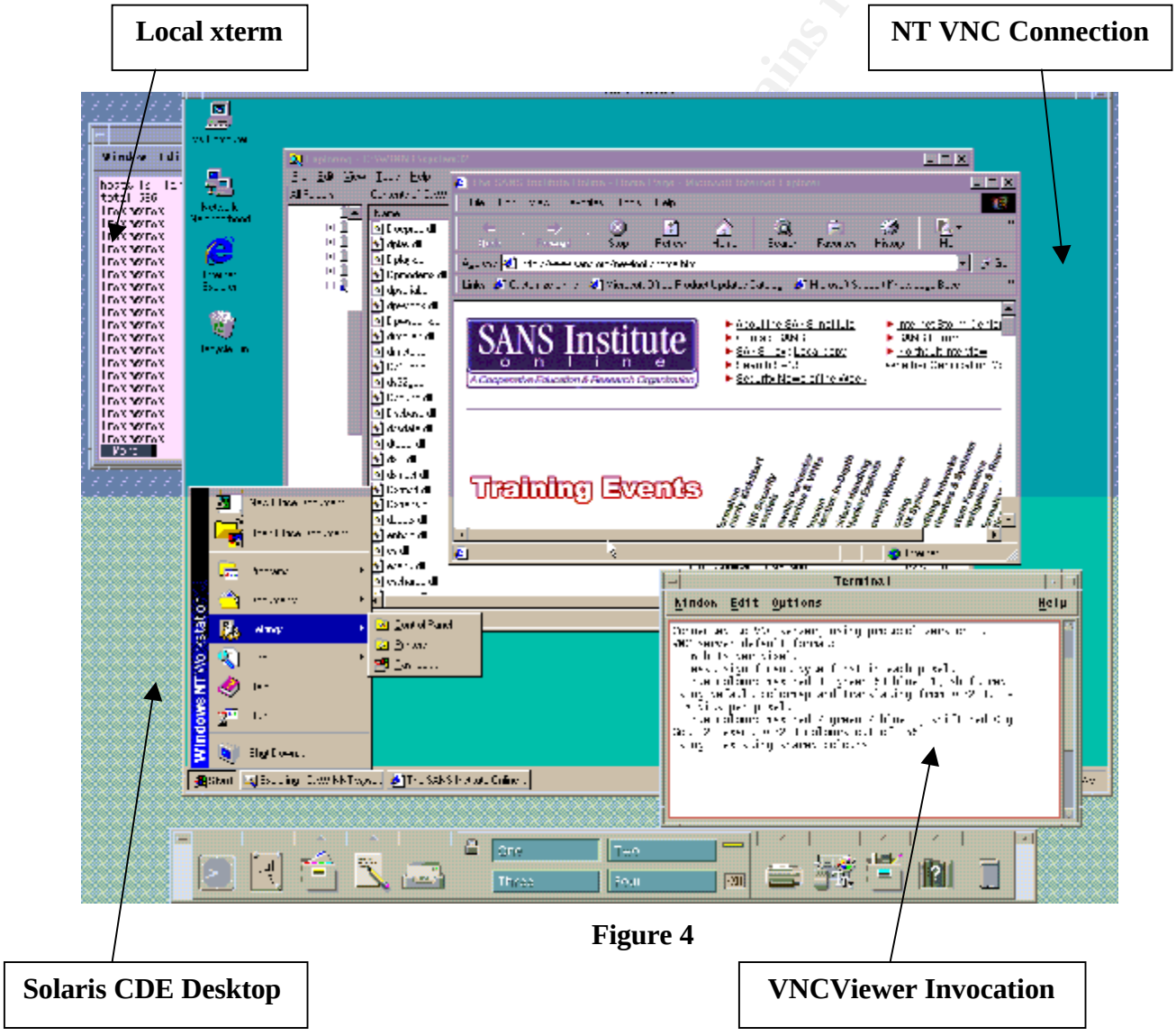

Figure 4

The ability to establish a remote GUI connection to a machine on a different platform can certainly be a perk to a system administrator whose workspace might be more aptly named a territory. This type of connection works fine when the machines in question are on the LAN, but what happens when the route to the target machine is traveling over an insecure channel? Enter VNC through Secure Shell (SSH).

Secure Shell

SSH is a vast topic. It is not the intent of this document to explore the inner workings of SSH though the reader is encouraged to consult the multitude of resources for information on SSH, some of which are listed in the resources section of this document. The aim here is to provide the reader with a cookbook approach to establishing a secure Windows VNC client connection to both Solaris and Windows VNC servers over an insecure channel. Briefly, SSH is "a popular, powerful, software-based approach to network security. Whenever data is sent by a computer to the network, SSH automatically encrypts it. When the data reaches its intended recipient, SSH automatically decrypts (unscrambles) it. The result is *transparent* encryption; users can work normally, unaware that their communications are safely encrypted on the network." [1] Although there are commercial implementations of SSH (http://www.ssh.fi), this document will explore the use of OpenSSH. OpenSSH is a freely available version of the secure shell. Its features include strong encryption (3DES, Blowfish), interoperability with SSH Protocol 1 and 2, data compression and strong authentication (Public Key, One-Time Password and Kerberos).

The first step in this process is to build Secure Sockets Layer (SSL). The Internet Engineering Task Force defines SSL as "a security protocol that provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery." [2] The following will result in a generic SSL installation though the reader is encouraged to consult the OpenSSL distribution INSTALL document for information that may be relevant to their site.

- If necessary, obtain and install Perl.
- Obtain the source for SSL from http://www.openssl.org/source/
- host% ./**config**<eol>
- host% **/usr/ccs/bin/make**<eol>
- host% **/usr/ccs/bin/make test**<eol>
- host% **/usr/ccs/bin/make install**<eol>

The next step is to build and install OpenSSH. Again, the INSTALL document should be consulted for specifics but the following will result in a generic SSH installation:

- If necessary, obtain and install Perl.
- Obtain and install Zlib (ftp://ftp.freesoftware.com/pub/infozip/zlib/index.html)
- Obtain the source for OpenSSH from http://www.openssh.org/portable.html

[1] Daniel J. Barrett, "SSH, The Secure Shell", 2001, 1:2

[2] Alan O. Freier , "The SSL Protocol Version 3.0", 18 Nov 1996 <http://www.netscape.com/eng/ssl3/draft302.txt>

- host% **./configure**<eol>
- host% **/usr/ccs/bin/make**<eol>
- host% **/usr/ccs/bin/make install**<eol>
- If the target system does not provide a method of random collection (i.e. /dev/random), optionally install Lutz Jaenicke's Pseudo Random Number Generator Daemon (http://www.lothar.com/tech/crypto).
- Configure SSH (http://www.openbsd.org/cgi-bin/man.cgi?query=sshd#C)

SSH VNC Connection

The VNC protocol normally uses port 59xx where *xx* is the display number. For example, in Windows it is possible to remotely display the console (i.e. display:0), hence the first connection is on port 5900. The standard UNIX VNC installation does not provide that capability so the first connection to a Solaris VNC server is on port 5901. SSH typically operates on port 22 unless otherwise configured. The question then becomes how to direct data destined for port 59xx on the remote end over SSH. This is accomplished through the use of the port forwarding feature of SSH. Port forwarding allows the transmission of TCP/IP traffic to be forwarded to SSH where it is encrypted before transmission. At the remote end, the data is decrypted and then directed to the original destination port and ultimately handled by the receiving application.

The last remaining step is the installation of a Windows SSH client. There are several Win32 SSH clients. This document will illustrate the use Cedomir Igaly's *SSH32* available at http://uncle-enzo.linuxmafia.com/pub/ms-windows/igaly-ssh/LATEST-IS-ssh32-1999-03-23.zip which allows a convenient way of establishing an SSH connection with port forwarding. SSH32 requires the cryptographic library *crypt32.dll* obtainable at http://uncle-enzo.linuxmafia.com/pub/ms-windows/igaly-ssh/crypt32.dll. OpenSSH is capable of supporting multiple SSH Protcols however *SSH32* is compliant only with the SSH Protocol 1 standard. Verifying that the OpenSSH configuration file *sshd_config* contains the value of *1* and optionally *2*, in the *Protocol* specification will insure that SSH32 and the SSH server can communicate. Simon Tatham's excellent Win32 SSH client *PuTTY* (http://www.chiark.greenend.org.uk/~sgtatham/putty) is SSH Protocol 2 standard compliant but port forwarding is not yet implemented.

The *SSH Options* sheet is presented when the *Connect* item of the *Action* menu is selected. The following are obligatory in enabling VNC over SSH.

- **Host Name**: The name of the host on which the SSH connection will be made.
- **Port**: The port number used by SSH; normally 22.
- **Cipher Type**: The type of encryption algorithm to be used in the SSH connection. Note that OpenSSH does not support IDEA as it is still covered by patent in many countries.
- **Authentication Type**: Select the desired authentication method (e.g. *Password*, *RSA*, *Challenge-Response*).
- **Local Forwards**: Data on the specified local port is forwarded to the specified port on the specified remote host.

A profile under construction, shown in Figure 5, establishes an SSH (i.e. Port 22) connection to a UNIX machine (i.e. Host Name) running an SSH server which will authenticate the user (i.e. User ID) using the specified authentication method and forward data arriving on the local port 5954 (i.e. Local Port) to the remote Windows machine (i.e. Host) on port 5900 (i.e. Remote Port). Note that if the remote connection was a UNIX machine, the remote port would be 5901, not 5900, as remote UNIX displays start at one.

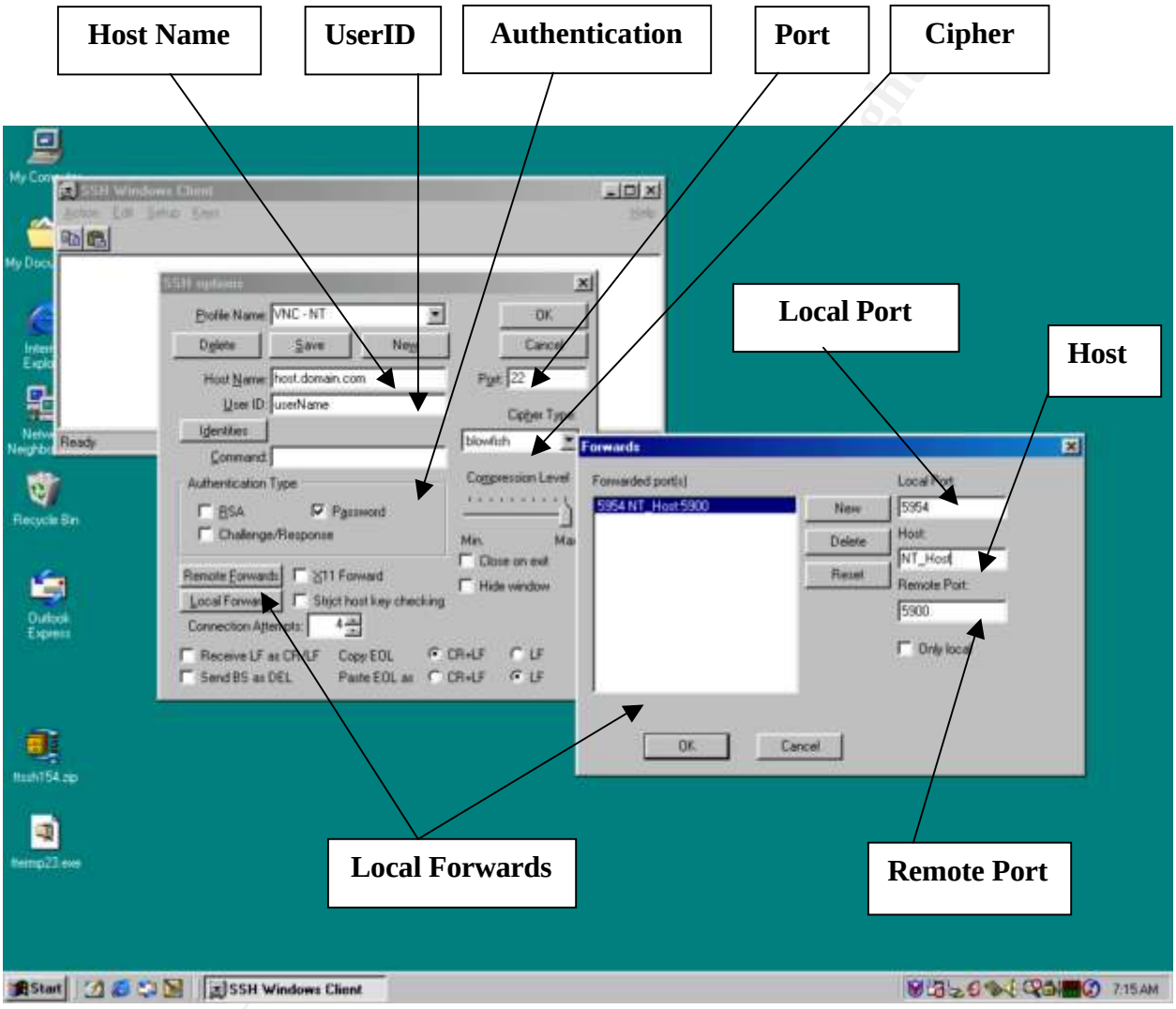

Figure 5

SSH32 does provide for the ability to use RSA as an authentication method. Cedomir Igaly provides directions for creating an RSA key at http://uncle-enzo.linuxmafia.com/pub/ms-windows/igaly-ssh/readme.igaly.

When a connection is made to the SSH server for the first time, SSH32 warns that the host is not known as shown in Figure 6. This is a safeguard against man-in-the-middle attacks. When OpenSSH was installed, the public host key */usr/local/etc/ssh_host_key.pub* file was created. If

the key presented by SSH32 does not match the public host key, there is no guarantee the connection is in fact to the desired machine. If the key is incorrect, the connection should be rejected and the cause investigated. If correct, the key can be accepted for one time or permanent use. Permanently accepted keys are stored in a *known.hosts* in the SSH32 directory.

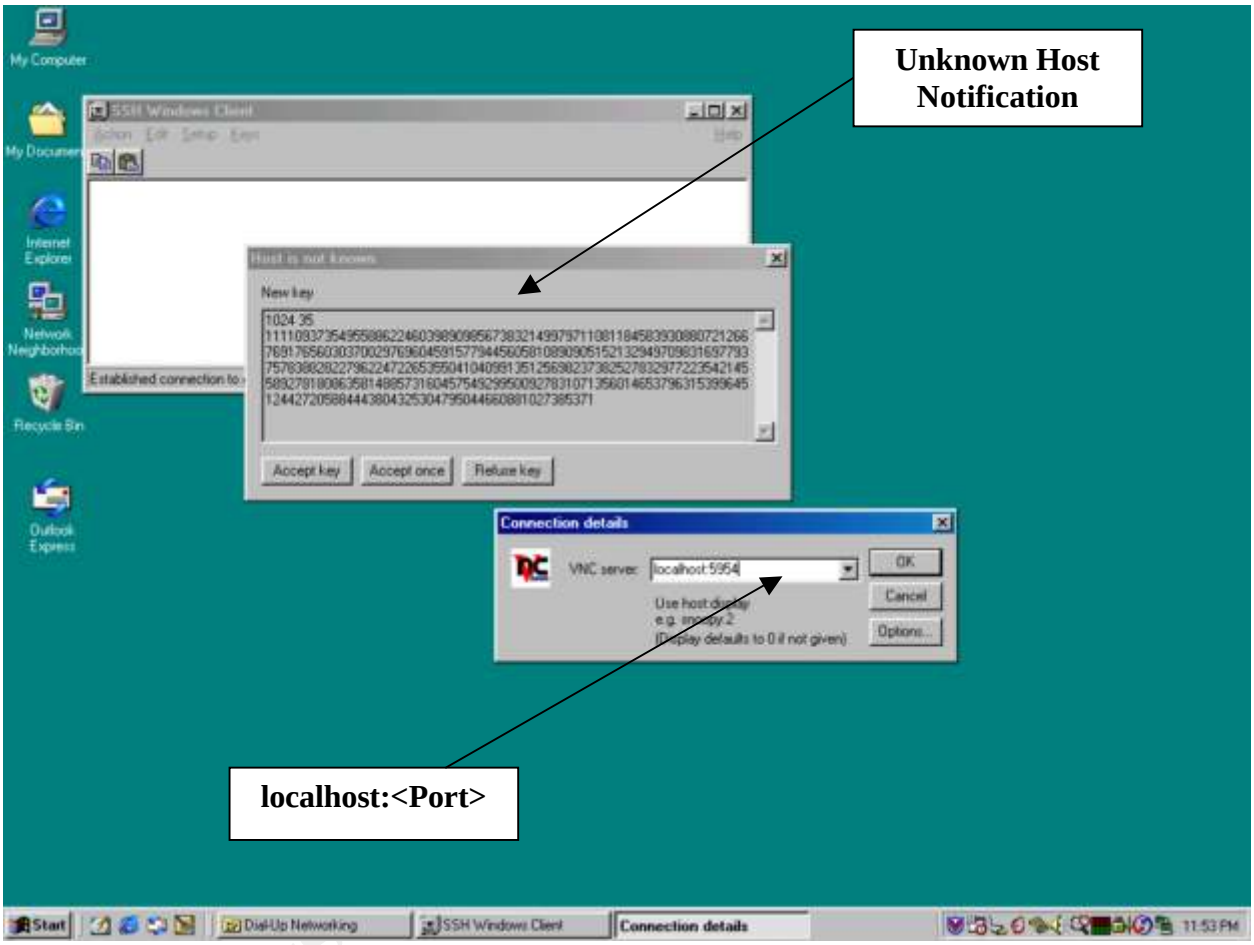

Figure 6

If the key is accepted and the user authenticated, a standard UNIX login ensues and the VNC connection can be invoked with *vncviewer* as before. However, because this SSH session was invoked with port forwarding enabled, the VNC server to connect to must be adjusted.
The SSH32 profile local forwards section was configured to take data arriving on the local port *5954* and forward it to the remote port *5900*. Consequently, specify *localhost:5954* as the VNC server. When the VNC connection is established, like before, authentication is required from the VNC server and the session begins as shown in Figure 7.

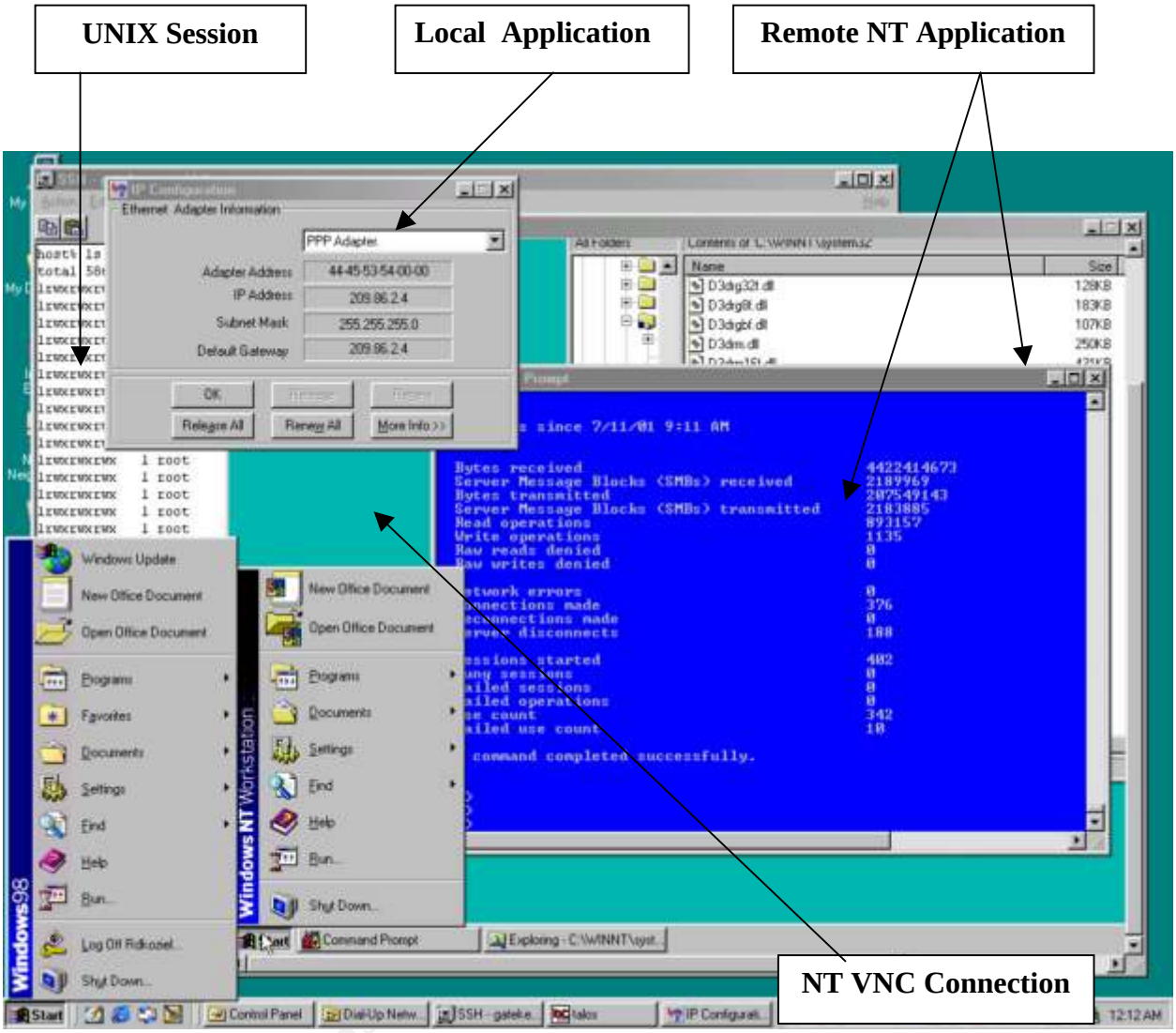

Figure 7

Privacy Issues

The prior examples of connecting to a Windows NT VNC server illustrated a connection to an existing login. In this instance the Windows NT VNC server was invoked interactively and as a result, the current user exerts controls when VNC connections will be made. In addition, since the properties sheet is available to the current user, local control exists over the options that allow the remote or local keyboard and pointer to be disabled. However, the Windows NT VNC server may also be installed as a service (i.e. *WinVNC.exe –install)* and then be set to start at boot time. A service installation has the advantage of not requiring a previous login to interact with the machine as the VNC client can send a Ctrl-Alt-Del thereby enabling a remote login to a Windows NT machine. The manner in which the Windows NT VNC server is invoked has significant privacy implications. With an interactive VNC server start, the user has control over

whether sessions will be allowed and if so, how the local and remote input devices will be allowed to interact with the machine. With the service install there is no system tray icon to indicate that the server is active, or even enabled. Depending on the privilege level of the current user and the account used to run the service, the current user may not have the ability to start/stop the VNC service. If a VNC server is installed as a service with local keyboard and pointer remaining enabled, an important question is raised. If the remote user establishes a session with no input, is this monitoring or eavesdropping? The answer lies in the organization's security policy. Does the IT staff have the right to connect to a user's machine? If so, is the IT staff obligated to announce their presence and intent? Must this type of connection be configured to disable the local keyboard and pointer or allow the local user to work unobserved? These are not simple questions and an organization must take pains in carefully answering them before incorporating this capability into a computing environment

Resources

Virtual Network Computing

VNC is a highly configurable program and the reader is encouraged to consult the VNC FAQ at http://www.uk.research.att.com/vnc/faq.html and the technical documentation at http://www.uk.research.att.com/vnc/docs.html for vital information on getting started, running the programs, and source compilation. There are a number of community contributions to VNC and the following are just a few. A complete list can be found at http://www.uk.research.att.com/vnc/extras.html.

- Firewalls: Enabling the UNIX VNC client to reach external servers through a SOCKS firewall. (http://www.uk.research.att.com/vnc/contrib/socks-patch.txt)

- RFB Server: Remote frame buffer server that provides remote control of an X desktop (http://www.hexonet.de/software/rfb)

- VNC Monitor: Enables a single window to contain multiple displays. (http://www.wilson.co.uk/Software/vnc/VncMonitor.htm)

- VNC Scan: Scans a network and lists the machines running VNC servers. (http://tgcs.web-it.com/)

- VNC Tight Encoder: Optimizes both UNIX and Windows VNC low-bandwidth connections. (http://www.ce.cctpu.edu.ru/vnc/)

- Zlib Compression: Enabling the X server and viewer to make use of Zlib compression. (http://www.uk.research.att.com/vnc/archives/1998-08/0039.html)

Secure Shell

The Internet is rich in resources for secure shell.

- Win32 Clients: Summary of referrals to Win32 SSH clients. (http://www.sorted.org/~chris/ssh/ and http://www.db.toronto.edu/~djast/ssh.html

- Windows NT SSH Server: Installation of an SSH server to enable secure remote access of an NT machine. (http://v.iki.fi/nt-ssh.html, http://www.certaintysolutions.com/tech-advice/ssh_on_nt.html, http://www.onlinemagic.com/~bgould/sshd.html)

- SANS contains a large number of resources on SSH. (http://www.sans.org/searchsans?p=1&lang=en&mode=any&q=SSH+Secure+Shell)

Any list of resources to Secure Shell would be lacking if it did not include Daniel J. Barrett and Richard E. Silverman's, *SSH The Secure Shell*.

Window Managers

- Introduction to window managers. (http://ugrad-www.cs.colorado.edu/X/WinMan.html)

- Tab Window Manager configuration (http://www.cs.ucsb.edu/facilities/software/twm.shtml, http://espc22.murdoch.edu.au/~stewart/guide/subsection2_11_2_1.html)

Conclusion

Virtual Network Computing is a valuable addition to the system administrator's toolbox. It adds a degree of clarity to the help desk enabling support personnel to understand precisely the nature of a problem. The report of an inability to print changes from "Tell me what you did." to "Show me what you did." Such a change in the resolution approach is invaluable. Taken to the next level where a support request is from an employee in a physical location that is either inconvenient or out of the area, VNC over SSH can then become a savior. However, as noted, when the VNC runs as a service under Windows NT, the possibility for invasion of privacy exists. A security policy that is well constructed will result in a set of parameters that delineate for the IT staff under what conditions this tool will be used. Doing so will help to maintain a clear channel of communication between the IT staff and the users they support.

Bibliography

Barrett, Daniel J. and Richard E. Silverman. SSH The Secure Shell. Sebastopol, CA:O'Reilly and Associates, 2001.

"Internet Assigned Numbers Authority." 24 Jun. 2001 Port Numbers. 11 Jul 2001 http://www.iana.org/assignments/port-numbers>

"Linuxmafia.com." 25 Jun. 2000. Index of /pub/ms-windows/igaly-ssh. Linuxmafia.com 11 Jul. 2001 <http://uncle-enzo.linuxmafia.com/pub/ms-windows/igaly-ssh/>

"OpenSSH." Features. 23 Feb. 2001. OpenBSD. 7 Jul. 2001 < http://www.openssh.org/features.html>

"OpenSSH." INSTALL. 8 May 2001. OpenBSD. 7 Jul. 2001. <ftp://ftp.ca.openbsd.org/pub/OpenBSD/OpenSSH/portable/INSTALL>

"OpenSSL." INSTALL. 11 Sep. 2000. OpenSSL. 7 Jul 2001. <ftp://ftp.opensll.org/source/openssl-0.9.6.tar.gz>

Stajano, Frank. "Virtual Network Computing." SSH-protected VNC: the case of the Windows client and the Unix server. 1999. AT&T Laboratories Cambridge. 8 Jul. 2001 <http://www.uk.research.att.com/vnc/sshwin.html>

Tatham, Simon. "PuTTY: A Free Win32 Telnet/SSH Client." PuTTY Known Bugs and Wish List 1 Feb 2001 GreenEnd Organisation. 10 Jul. 2001 < http://www.chiark.greenend.org.uk/~sgtatham/putty/wishlist.html>

"Transport Layer Security Working Group." The SSL Protocol Version 3.0. 18 Nov. 1996. Netscape Communications. 8 Jul. 2001 <http://www.netscape.com/eng/ssl3/draft302.txt>

"Virtual Network Computing." Making VNC more secure using SSH. 1999. AT&T Laboratories Cambridge. 8 Jul. 2001 <http://www.uk.research.att.com/vnc/sshvnc.html>

"Virtual Network Computing." VNC – How it works. 1999. AT&T Laboratories Cambridge. 5 Jul. 2001 <http://www.uk.research.att.com/vnc/howitworks.html>

"Virtual Network Computing." What makes it different from other systems?, 1999. AT&T Laboratories Cambridge. 5 Jul. 2001 <http://www.uk.research.att.com/vnc/index.html>

Last Updated: September 25th, 2020

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS October Singapore 2020	**Singapore, SG**	**Oct 12, 2020 - Oct 24, 2020**	**Live Event**
SANS Community CTF	**,**	**Oct 15, 2020 - Oct 16, 2020**	**Self Paced**
SANS SEC504 Rennes 2020 (In French)	**Rennes, FR**	**Oct 19, 2020 - Oct 24, 2020**	**Live Event**
SANS SEC560 Lille 2020 (In French)	**Lille, FR**	**Oct 26, 2020 - Oct 31, 2020**	**Live Event**
SANS Tel Aviv November 2020	**Tel Aviv, IL**	**Nov 01, 2020 - Nov 06, 2020**	**Live Event**
SANS Sydney 2020	**Sydney, AU**	**Nov 02, 2020 - Nov 14, 2020**	**Live Event**
SANS Secure Thailand	**Bangkok, TH**	**Nov 09, 2020 - Nov 14, 2020**	**Live Event**
APAC ICS Summit & Training 2020	**Singapore, SG**	**Nov 13, 2020 - Nov 21, 2020**	**Live Event**
SANS FOR508 Rome 2020 (in Italian)	**Rome, IT**	**Nov 16, 2020 - Nov 21, 2020**	**Live Event**
SANS Community CTF	**,**	**Nov 19, 2020 - Nov 20, 2020**	**Self Paced**
SANS Local: Oslo November 2020	**Oslo, NO**	**Nov 23, 2020 - Nov 28, 2020**	**Live Event**
SANS Wellington 2020	**Wellington, NZ**	**Nov 30, 2020 - Dec 12, 2020**	**Live Event**
SANS OnDemand	**OnlineUS**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**