

Frank Isaacs

Dr. Phil Lunsford

ICTN 4040.601

22 April 2008

A Comparison of VNC Connection Methods

Abstract

VNC (Virtual Network Computing) is an open-source, cross-platform protocol for viewing GUI desktops on remote machines within a LAN or over a WAN/Internet connection. This paper discusses different methods of deploying VNC with an emphasis on the security considerations of each method, and the tradeoffs associated with the convenience of each method. The methods discussed include an open connection and a connection tunneled over ssh (Secure Shell). Includes information regarding the platform-independence of VNC and ssh implementations, so that solutions presented can be applied to Windows, Linux, Mac, and even other operating systems – securely and with open-source software.

VNC Background

Virtual Network Computing, or VNC, is an open-source, cross-platform software product which enables viewing and controlling GUI desktops on remote machines on a network, either within a LAN or over a WAN/Internet connection. The machine that the user is sitting at is referred to as the client, and the remote machine being controlled is referred to as the server. Viewing entails sending screen updates from the server to the client; controlling sends mouse and keyboard from the client to the server.

The nature of VNC connections is such that any machine – running any operating system – may act as either client or server simply by running the appropriate executable program, and the client and server do not need to be running the same operating system. Although a number of configuration options exist for both server and client, for most installations, a simple install of the software is all that is required to get started with VNC. Since it is an open-

source software product, there are several widely-used implementations available, including TightVNC¹ and RealVNC.² The availability of source code provides the ability to extend and customize the code, which has resulted in additional implementations. Notable among these is UltraVNC,³ which adds the ability to use a Windows username and password to restrict access to the remote machine, rather than requiring an administrator to specifically create (and subsequently maintain) a password for VNC access.

In its simplest form, VNC is a frame-buffer update program. Normally, a screen update is sent to a video driver (software) or directly to some hardware. This so-called “framebuffer” update is redirected or copied by VNC to its own driver, sending it across the network to the client. There are various compression schemes included for increasing performance of these updates, but the basic mechanism is the updating of the framebuffer on the remote machine.

SSH Background

Secure SHell, or SSH, is a secure means of establishing a command-line connection from one machine to another over otherwise insecure channels. By using authentication and encryption, the remote connection is highly secure and does not transfer data or passwords across the wire in clear text. This is in contrast to the technologies it replaced, such as rlogin, rsh, and telnet, which are all insecure. Since its initial release in 1995, SSH has undergone a number of code updates and branches to its development tree, leading to the current SSH-2 protocol. The version of the libraries used is incremented with new releases, but the underlying architecture and protocol are stable, and the SSH-2 protocol was codified and standardized under a number of RFCs by the Internet Engineering Task Force in 2005 and 2006.^{4,5}

Like the VNC protocol, SSH is typically operated in a client-server paradigm. The server uses the well-known port of 22 (one less than the telnet port it essentially replaces), and the client establishes its port when making the connection to the server. Aside from the static nature of the port selection on the server (which is configurable), there is no practical difference between the client and server side of the SSH connection. Each side encrypts the traffic before

sending it over the wire. SSH also shares with VNC the open-source nature of its code base; it also has multiple implementations available in the open-source community, and there are commercial versions available.⁶

In addition to providing security and confidentiality of communication, SSH offers several other features to users. Chief among them is the ability to tunnel ports from the client machine to the server machine. This presents interesting – and secure – possibilities, such as the ability to connect to a remote database over a secure connection, and securing traffic for remote screen display, which includes both VNC and X sessions in *nix-like operating systems.

Security

VNC was not designed to be a secure protocol; rather, it was designed to be lightweight and low-overhead. In fact, a VNC viewer is available that can be downloaded and run on Windows platforms without any installation, and it is under 400 KB in size. This lack of security can compromise an installation, because the session itself can be sniffed, and because a compromised password may be reused elsewhere by a user. The passwords VNC sends are encrypted, but the password is limited to 8 characters, and the DES key is only 56 bits.⁷ This level of security was cracked in 1999 in less than a day⁸ and in 2001 was already considered inadequate as a security encryption standard.⁹ Although VNC does encrypt passwords, unlike predecessor technologies such as telnet and rlogin, this weakness renders it essentially insecure. A web search for “VNC password decrypt” results in plenty of hits for programs which can decrypt a password in several ways: from a captured data stream (i.e. ethereal or tcpdump), by capturing the data stream real-time as the connection is made, or if the server machine is accessible, directly from the Windows registry.

VNC has not been advertised as secure, so this weakness is not necessarily a flaw in its design. Rather, as most overall security strategies rely on processes and procedures as much as technology itself, a better approach – using existing technologies – is available.

VNC over SSH

Although still not ubiquitous in the remote-control computing arena, the concept of using SSH to secure a VNC connection is not new. It was reported at least as early as the January 2000 issue of *Boardwatch*,¹⁰ and has been increasingly used since then. This method of accessing VNC connections over an SSH connection provides an excellent means of securing an otherwise insecure protocol. In much the same way that a VPN connection uses encryption to securely traverse an insecure medium such as the Internet, SSH enables VNC to be secured by encrypting the entire data stream. Following is a discussion

Technical Details

There are two primary mechanisms by which an SSH-secured VNC session greatly improves security of a network installation: strong encryption of the data stream, and reduced network footprint. A discussion of the two methods follows. The connections were monitored by capturing and analyzing network traffic at the client machine when connecting to the VNC server machine both without and with SSH encryption. The open-source packet analyzer Wireshark was used on the client machine, running Windows XP Professional SP2.

Non SSH-secured connection

When connecting over an open network, Wireshark is fully cognizant of the VNC packets as they traverse the network. As shown in Figure 1, several of the packets are identified as being VNC packets, and the payload of the packets is known and can be displayed. In particular, note the "authentication challenge from server," "authentication response from client," and "authentication result" packets. Since these packets represent a password of a known maximum length, and they are known to use an encryption scheme that is no longer secure, this represents a communication that can be compromised using readily available technology.

.102	VNC	Security types supported
.204	VNC	Authentication type selected by client
.102	VNC	Authentication challenge from server
.204	TCP	29357 > 5901 [ACK] Seq=14 Ack=31 Win=65505 Len=0
.204	VNC	Authentication response from client
.102	VNC	Authentication result
.204	VNC	Share desktop flag
.102	VNC	Server framebuffer parameters
.204	VNC	Client key event

Figure 1 - Wireshark session capture – open connection

In addition to the weakness associated with the compromise of the password itself, the use of SSH over an open connection presents an additional security vulnerability. For best security, especially on server machines, it is common practice to secure machines by turning off all unused services and closing ports associated with them, only opening ports and services that are known to be required.¹¹ This reduces the chance that a potential attacker can gain information about the machine simply by scanning it and attempting connections. The use of VNC's well-known ports beginning at 5900 (standard) and 5800 (java/web based) gives a would-be attacker valuable information about what is in use on a target server. Knowing that VNC is in use might lead an attacker to zero in on a particular vulnerability, such as the weakness of passwords described above, and leading to potentially catastrophic results for the network.

SSH-secured connection

The SSH-secured method of connecting is relatively straightforward. The client performs the additional step of starting an SSH connection to the target machine first. This connection must be configured to allow a *tunnel* between the client and the server. A tunnel is so-named because it creates a link between a local port (on the client) and a remote port (on the server). These ports may or may not be the same; the important point is that when a request is made on the local machine on the configured port (for example, 500), the tunnel would redirect it to the remote machine on port 5901 for the first VNC session. This can be accomplished by GUI configuration in a Windows client such as PuTTY (shown in Figure 2), or on the command line when establishing the connection in Linux. The configuration shown in Figure 2 specifies that

when a local request is made on port 5901, it is forwarded to the remote machine, also on port 5901. A similar command in Linux might look like this:

```
ssh -R 5901:remote.host.com:5901 remote.host.com
```

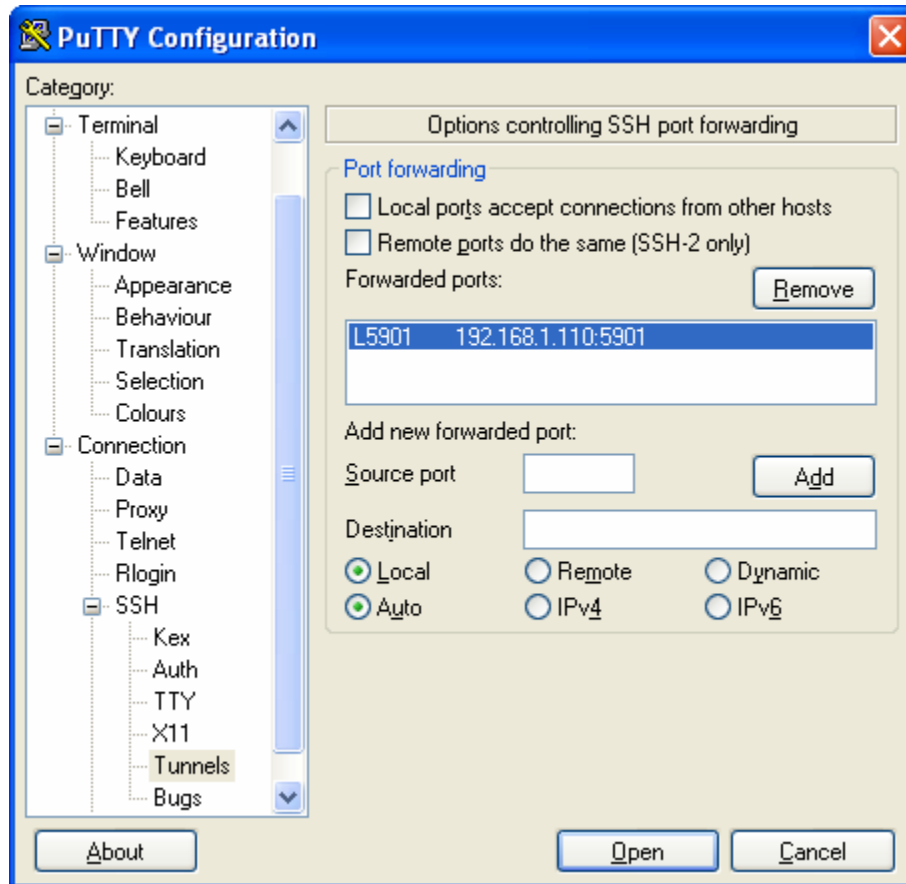


Figure 2 - Configuring an SSH Tunnel in PuTTY

Regardless of which platform is used as the client, the resulting connection will present packets only to and from port 22 on the server machine, as shown in Figure 3. This disguises the nature of the connection, and protects the transport with encryption technology that is much more secure than VNC's 56-bit key (which is used only for passwords anyway). An added benefit of this method is that only port 22 need be allowed through a firewall. A machine that is allowing any connection is likely to need this port open anyway, so no additional information is made available to a potential intruder scanning the system on its external address. It is better to tell

the world only that SSH is available and running than to reveal what services are actually running on the machine, as would be the case with an open VNC server.

2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=53 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52
2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=105 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52
2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=157 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52
2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=209 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52
2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=261 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52
2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=313 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52
2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=365 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52
2.102	TCP	ssh > 29313 [ACK]	Seq=1 Ack=417 Win=16080 Len=0
2.204	SSH	Encrypted request	packet len=52

Figure 3 - Wireshark session capture - SSH-secured connection

More info on SSH

It should be noted that the use of certificates as part of the SSH authentication process can further increase security of the connection, requiring that proper certificates exist in order to authenticate the connection in the first place. Certificates can be arbitrarily segmented, so that individual users can have access to particular resources without giving them more than they need to access in order to perform their job duties.¹² In addition, when responsibilities change, certificates can be revoked at the server side of a connection, so users can be restricted even if they are remotely located.

SSH Pitfalls

Note that SSH is not without its security vulnerabilities. Although it is open source and flaws are quickly patched, nevertheless, flaws have been found. For example, Saito, et. al. listed vulnerabilities with key exchange, and user authentication.¹³ The recommendations given to cure these two architectural defects, presented in 2002, which include not transmitting the password over the wire, have been remedied in subsequent versions of SSH. A search of the archives of

the OpenSSH mailing list for either "exploit" or "vulnerability" yields hits – even into the current year. Many of them are "potential" vulnerabilities, and patches are issued to protect against them, but this does show that it is important to keep up on the latest patches, especially for software which is critical to information security in the enterprise, such as SSH.

Conclusion

The VNC protocol provides an excellent, lightweight means of remotely controlling GUI desktops across a network connection. It is available on multiple platforms and allows different operating systems to be used to control remote desktops – even on dissimilar operating systems. It has a number of performance features, including enable detection of line speed and selection of compression appropriate to the connection. Some implementations offer integration into the host operating system for user authentication purposes.

However, VNC is not a secure protocol, and care must be taken to avoid compromise of a machine running VNC, and possibly the entire network once a single machine is compromised. A secure solution was presented, which combines VNC with the open-source SSH protocol, and which has been used in the open-source community for several years, with the same effect as protecting a VPN connection over insecure channels, such as the Internet. While the SSH protocol itself – like all software – is potentially subject to exploit, it is well-regarded and was created specifically to secure communications between two hosts, and it continues to be maintained for this purpose. The solution presents a highly secure means of remote control, using two products that are widely supported in the open-source community.

References

-
- ¹ "Introduction to TightVNC." TightVNC. 10 Apr 2008 <<http://www.tightvnc.com/intro.html>>.
- ² "VNC - the original cross-platform remote control solution." RealVNC. RealVNC Ltd.. 05 Feb 2008 <<http://www.realvnc.com/vnc/index.html>>.
- ³ "UltraVNC FAQ." UltraVNC. UltraVNC. 05 Feb 2008 <<http://www.uvnc.com/general/faq.html>>.
- ⁴ "Secure Shell (secsh) Charter." IETFInternet Engineering Task Force. 08 Sep 2005. The Internet Society (ISOC). 20 Apr 2008 <<http://www.ietf.org/html.charters/OLD/secsh-charter.html>>.
- ⁵ "Secure Shell (Concluded WG)." IETFInternet Engineering Task Force. 08 Sep 2005. The Internet Society (ISOC). 20 Apr 2008 <<http://tools.ietf.org/wg/secsh/>>.
- ⁶ Emery, Van. "VNC over SSH2 - A TightVNC Tutorial." vanemery.com. Jun 2003. 20 Apr 2008 <<http://www.vanemery.com/Linux/VNC/vnc-over-ssh.html>>.
- ⁷ "TightVNC Frequently Asked Questions." TightVNC. TightVNC. 19 Apr 2008 <<http://www.tightvnc.com/faq.html#howsecure>>.
- ⁸ Nelson, Matthew. "56-bit DES algorithm broken in record time." InfoWorld 25 Jan. 1999: 45. ABI/INFORM Global. ProQuest. East Carolina University Joyner Library, Greenville, NC. 17 Apr. 2008 <<http://www.proquest.com.jproxy.lib.ecu.edu/>>.
- *⁹ Rueda, Andres. "The Implications of Strong Encryption Technology on Money Laundering." Albany Law Journal of Science & Technology 12(2001).
- ¹⁰ Carl, Jeffrey. "Freenixes by remote control with virtual network computing." Boardwatch 1 Jan. 2000: 100-102. General Interest Module. ProQuest. East Carolina University Joyner Library, Greenville, NC. 21 Apr. 2008 <<http://www.proquest.com.jproxy.lib.ecu.edu/>>.
- ¹¹ Taylor, Laura. "Guidelines for configuring your firewall rule-set." ZDNet. 12 Apr 2001. 21 Apr 2008 <<http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2707159,00.html>>.
- ¹² Bailey, Dave. "Enterprise week - OpenSSH for remote security." ZDNet. 12 Apr 2001. 21 Apr 2008 <<http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2707159,00.html>>.
- *¹³ Saito, Takamichi, Toshiyuki Kito, Kentaro Umesawa, and Fumio Mizoguchi. "Architectural Defects of the Secure Shell." Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA '02). (2002): 1-7.