Information Security Management Handbook
Edited by Harold F. Tipton and Micki Krause
Boca Raton: CRC Press LLC, 2003

The Reality of Virtual Computing

Chris Hare, CISSP, CISA

A major issue in many computing environments is accessing the desktop or console display of a different graphical-based system than the one you are using. If you are in a homogeneous environment, meaning you want to access a Microsoft Windows system from a Windows system, you can use applications such as Timbuktu, pcAnywhere, or RemotelyPossible.

In today's virtual enterprise, many people have a requirement to share their desktops or allow others to view or manipulate it. Many desktop-sharing programs exist aside from those mentioned, including Microsoft NetMeeting and online conferencing tools built into various applications.

The same is true for UNIX systems, which typically use the X Windows display system as the graphical user interface. It is simple matter of running the X Windows client on the remote system and displaying it on the local system.

However, if you must access a dissimilar system (e.g., a Windows system from a UNIX system) the options are limited. It is difficult to find an application under UNIX allowing a user to view an online presentation from a Windows system using Microsoft PowerPoint. This is where Virtual Network Computing, or VNC, from AT&T's United Kingdom Research labs, enters the picture.

This chapter discusses what VNC is, how it can be used, and the security considerations surrounding VNC. The information presented does get fairly technical in a few places to illustrate the protocol, programming techniques, and weaknesses in the authentication scheme. However, the corresponding explanations should address the issues for the less technical reader.

WHAT IS VNC?

The Virtual Network Computing system, or VNC, was developed at the AT&T Research Laboratories in the United Kingdom. VNC is a very simple

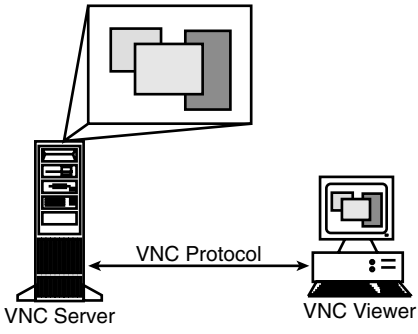

Exhibit 40-1. The VNC components.

graphical display protocol allowing connections from heterogeneous or homogeneous computer systems.

VNC consists of a server and a viewer, as illustrated in Exhibit 40-1. The server accepts connection requests to display its local display on the viewer.

The VNC services are based upon what is called a *remote framebuffer* or RFB. The framebuffer protocol simply allows a server to update the framebuffer or graphical display device on the remote viewer. With total independence from the graphical device driver, it is possible to represent the local display from the server on the client or viewer. The portability of the design means the VNC server should function on almost any hardware platform, operating system, windowing system, and application.

Support for VNC is currently available for a number of platforms, including:

- Servers:
  - UNIX (X Window system)
  - Microsoft Windows
  - Macintosh
- Viewers:
  - UNIX (X Window System)
  - Microsoft Windows
  - Macintosh
  - Java
  - Microsoft Windows CE

VNC is described as a *thin-client* protocol, making very few requirements on the viewer. In this manner, the client can run on the widest range of

Exhibit 40-2. The X Windows VNC client.

hardware. There are a number of factors distinguishing VNC from other remote display systems, including:

- VNC is stateless, meaning you can terminate the session and reconnect from another system and continue right where you left off. When you connect to a remote system using an application such as a PC X Server and the PC crashes or is restarted, the X Window system applications running terminate. Using VNC, the applications remain available after the reboot.
- The viewer is a thin client and has a very small memory footprint.
- VMC is platform independent, allowing a desktop on one system to be displayed on any other type of system, including Java-capable Web browsers.
- It can be shared, allowing multiple users the ability to view and share a single desktop at the same time. This can be useful when needing to perform presentations over the network.
- And, best of all, VNC is free and distributed under the standard GNU General Public License (GPL).

These are some of the benefits available with VNC. However, despite the clever implementation to share massive amounts of video data, there are a few weaknesses, as presented in this chapter.

HOW IT WORKS

Accessing the VNC server is done using the VNC client and specifying the IP address or node name of the target VNC server as shown in Exhibit 40-2.

The window shown in Exhibit 40-2 requests the node name or IP address for the remote VNC server. It is also possible to add a port number with the address. The VNC server has a password to protect unauthorized access to the server. After providing the target host name or IP address, the user is prompted for the password to access the server, as seen in Exhibit 40-3.

Exhibit 40-3. Entering the VNC server password.

Exhibit 40-4. The UNIX VNC client displays the password.

The Microsoft Windows VNC viewer does not display the password when the user enters it, as shown in Exhibit 40-4. However, the VNC client included in Linux systems does not hide the password when the user enters it. This is an issue because it exposes the password for the server to public view. However, because there is no user-level authentication, one could say there is no problem. Just in case you missed it, *there is no user-level authentication*. This is discussed again later in this chapter in the section entitled "Access Control."

The VNC client prompts for the password after the connection is initiated with the server and requests authentication using a challenge–response scheme. The challenge–response system used is described in the section entitled "Access Control."

Once the authentication is successful, the client and server then exchange a series of messages to negotiate the desktop size, pixel format, and the encoding schemes. To complete the initial connection setup, the client requests a full update for the entire screen and the session commences. Because the client is stateless, either the server or the client can close the connection with no impact to either the client or server.

Exhibit 40-5. The Windows desktop from Linux.

Actually, this chapter was written logged into a Linux system and using VNC to access a Microsoft Windows system that used VNC to access Microsoft Word. When using VNC on the UNIX- or Linux-based client, the user sees the Windows desktop as illustrated in Exhibit 40-5.

The opposite is also true — a Windows user can access the Linux system and see the UNIX or Linux desktop as well as use the features and functionality offered by the UNIX platform (see Exhibit 40-6). However, VNC is not limited to these platforms, as mentioned earlier and demonstrated later.

However, this may not be exactly what the Linux user was expecting. The VNC sessions run as additional displays on the X server, which on RedHat Linux systems default to the TWM Window Manager. This can be changed; however, that is outside the topic area of this chapter.

NETWORK COMMUNICATION

All network communication requires the use of a network port. VNC is a connection-based TCP/IP application requiring the use of network ports. The VNC server listens on two ports. The values of these ports depend upon the access method and the display number.

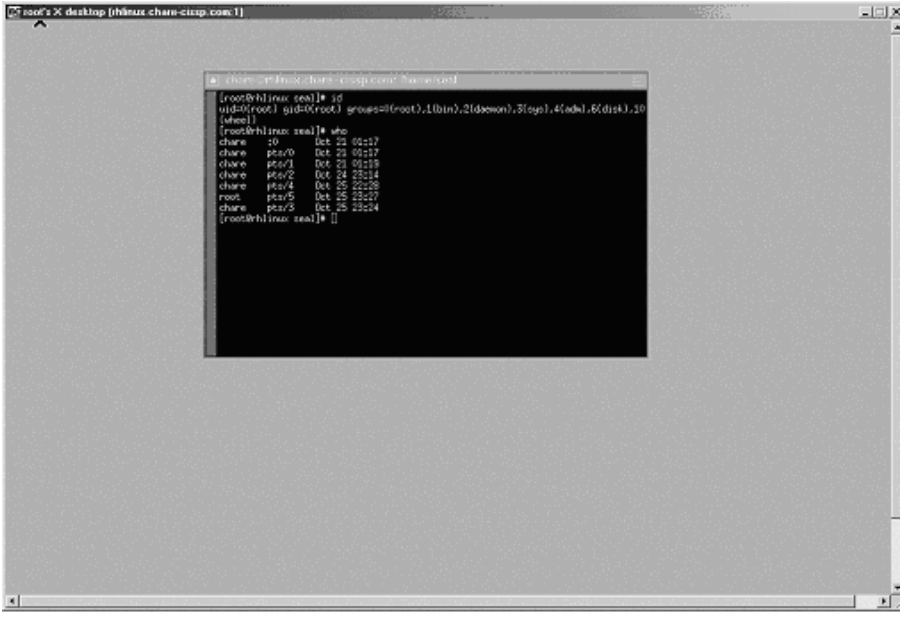

Exhibit 40-6. The TWM Window Manager from Windows.

The VNC server listens on port 5900 plus the display number. WinVNC for Microsoft Windows defaults to display zero, so the port is 5900. The same is true for the Java-based HTTP port, listening at port 5800 plus the display number. This small and restrictive Web server is discussed more in the section entitled "VNC and the Web."

If there are multiple VNC servers running on the same system, they will have different port numbers because their display number is different, as illustrated in Exhibit 40-7.

There is a VNC server executed for each user who wishes to have one. Because there is no user authentication in the VNC server, the authentication is essentially port based. This means user chare is running a VNC server, which is set up on display 1 and therefore port 5901. Because the VNC server is running at user chare, anyone who learns or guesses the password for the VNC server can access chare's VNC server and have all of chare's privileges.

Looking back to Exhibit 40-6, the session running on the Linux system belonged to root as shown here:

```
[chare@rhlinux chare]$ ps -ef | grep vnc
root   20368      1  0 23:21 pts/1  00:00:00 Xvnc :
   1 -desktop X -httpd/usr/s
```

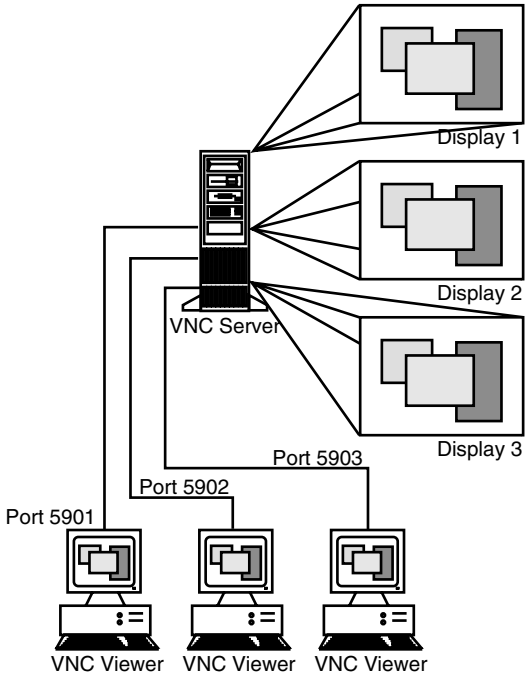

Exhibit 40-7. Multiple VNC servers.

```
chare 20476  20436  0 23:25 pts/3  00:00:00 grep vnc
[chare@rhlinux chare]$
```

In this scenario, any user who knows the password for the VNC server on display 1, which is port 5901, can become root with no additional password required. Because of this access control model, good-quality passwords must be used to control access to the VNC server; and they must be kept absolutely secret.

As mentioned previously, the VNC server also runs a small Web server to support access through the Java client. The Web server listens on port 58xx, where xx is the display number for the server. The HTTP port on the Web server is only used to establish the initial HTTP connection and download the applet. Once the applet is running in the browser, the connection uses port 59xx. The section entitled "VNC and the Web" describes using the VNC Java client.

There is a third mode, where the client listens for a connection from the server rather than connecting to a server. When this configuration is selected, the client listens on port 5500 for the incoming connection from the server.

ACCESS CONTROL

As mentioned previously, the client and server exchange a series of messages during the initial connection setup. These protocol messages consist of:

- ProtocolVersion
- Authentication
- ClientInitialization
- ServerInitialization

Once the *ServerInitialization* stage is completed, the client can send additional messages when it requires and receive data from the server.

The protocol version number defines what level of support both the client and server have. It is expected that some level of backward compatibility is available because the version reported should be the latest version the client or server supports. When starting the VNC viewer on a Linux system, the protocol version is printed on the display (standard out) if not directed to a file.

Using a tool such as *tcpdump*, we can see the protocol version passed from the client to the server (shown in bold text):

```
22:39:42.215633 eth0 < alpha.5900 > rhlinux.chare-cissp.com.1643:
P 1:13(12) ack 1 win 17520 <nop,nop,timestamp 37973 47351119>
                        4500 0040 77f0 0000 8006 4172 c0a8 0002
                        c0a8 0003 170c 066b 38e9 536b 7f27 64fd
                        8018 4470 ab7c 0000 0101 080a 0000 9455
                        02d2 854f 5246 4220 3030 332e 3030 330a

                        E^@ ^@ @ w.. ^@^@ ..^F A r.... ^@^B
                        .... ^@^C ^W^L ^F k 8.. S k ^¿ ' d..
                        ..^X D p  .. | ^@^@ ^A^A ^H^J ^@^@.. U
                        ^B.. .. O R F B 0 0 3. 0 0 3^J
```

and then again from the server to the client:

```
22:39:42.215633 eth0 > rhlinux.chare-cissp.com.1643
> alpha.5900: P 1:13(12) ack 13 win 5840 <nop,nop,time
stamp47351119 37973> (DF)
 4500 0040 e1b5 4000 4006 d7ac c0a8 0003
 c0a8 0002 066b 170c 7f27 64fd 38e9 5377
 8018 16d0 d910 0000 0101 080a 02d2 854f
 0000 9455 5246 4220 3030 332e 3030 330a

 E^@  ^@ @ .... @^@  @^F  .... .... ^@^C
 .... ^@^B ^F k ^W^L ^¿ ' d..  8..  S w
 ..^X ^V.. ..^P ^@^@ ^A^A ^H^J ^B.. .. O
 ^@^@ .. U R F  B    0 0  3.   0 0  3^J
```

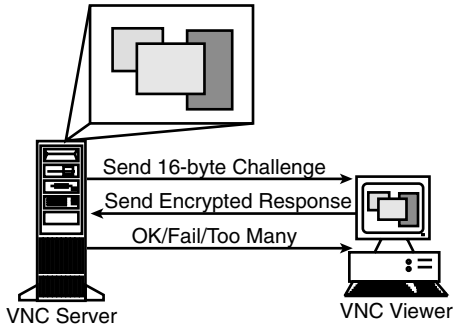

Exhibit 40-8. The VNC authentication challenge–response.

With the protocol version established, the client attempts to authenticate to the server. The password prompt shown in Exhibit 40-3 is displayed on the client, where the user enters the password.

There are three possible authentication messages in the VNC protocol:

1. *Connection Failed.* The connection cannot be established for some reason. If this occurs, a message indicating the reason the connection could not be established is provided.
2. *No Authentication.* No authentication is needed. This is not a desirable option.
3. *VNC Authentication.* Use VNC authentication.

The VNC authentication challenge–response is illustrated in Exhibit 40-8.

The VNC authentication protocol uses a challenge–response method with a 16-byte (128-bit) challenge sent from the server to the client. The challenge is sent from the server to the client in the clear. The challenge is random, based upon the current time when the connection request is made. The following packet has the challenge highlighted in bold.

```
14:36:08.908961 < alpha.5900 > rhlinux.chare-cissp.com.
2058: P 17:33(16) ack 13 win 17508 <nop,nop,timestamp
800090 8590888>
               4500 0044 aa58 0000 8006 0f06 c0a8 0002
               c0a8 0003 170c 080a ae2b 8b87 f94c 0e34
               8018 4464 1599 0000 0101 080a 000c 355a
               0083 1628 0456 b197 31f3 ad69 a513 151b
               195d 8620

               E^@ ^@ D .. X ^@^@ ..^F ^O^F .... ^@^B
               .... ^@^C ^W^L ^H^J .. + .... .. L ^N 4
               ..^X D d ^U.. ^@^@ ^A^A ^H^J ^@^L 5 Z
               ^@.. ^V ( ^D V .... 1.. .. I ..^S ^U^[
               ^Y] ..
```

The client then encrypts the 16-byte challenge using Data Encryption Standard (DES) symmetric cryptography with the user-supplied password as the key. The VNC DES implementation is based upon a public domain version of Triple-DES, with the double and triple length support removed. This means VNC is only capable of using standard DES for encrypting the response to the challenge. Again, the following packet has the response highlighted in bold.

```
14:36:11.188961 < rhlinux.chare-cissp.com.2058 >
alpha.5900: P 13:29(16) ack 33 win 5840
<nop,nop,timestamp 8591116 800090> (DF)
                4500 0044 180a 4000 4006 a154 c0a8 0003
                c0a8 0002 080a 170c f94c 0e34 ae2b 8b97
                8018 16d0 facd 0000 0101 080a 0083 170c
                000c 355a 7843 ba35 ff28 95ee 1493 caa7
                0410 8b86

                E^@  ^@ D ^X^J @^@  @^F .. T .... ^@^C
                .... ^@^B ^H^J ^W^L .. L ^N 4 .. + ....
                ..^X ^V.. .... ^@^@ ^A^A ^H^J ^@.. ^W^L
                ^@^L 5 Z  x C .. 5 .. ( .... ^T.. ....
                ^D^P....
```

The server receives the response and, if the password on the server is the same, the server can decrypt the response and find the value issued as the challenge. As discussed in the section "Weaknesses in the VNC Authentication System" later in this chapter, the approach used here is vulnerable to a man-in-the-middle attack, or a cryptographic attack to find the key, which is the password for the server.

Once the server receives the response, it informs the client if the authentication was successful by providing an *OK*, *Failed*, or *Too Many* response. After five authentication failures, the server responds with *Too Many* and does not allow immediate reconnection by the same client.

The *ClientInitialization* and *ServerInitialization* messages allow the client and server to negotiate the color depth, screen size, and other parameters affecting the display of the framebuffer.

As mentioned in the "Network Communication" section, the VNC server runs on UNIX as the user who started it. Consequently, there are no additional access controls in the VNC server. If the password is not known to anyone, it is safe. Yes and no. Because the password is used as the key for the DES-encrypted response, the password is never sent across the network in the clear. However, as we will see later in the chapter, the challenge–response method is susceptible to a man-in-the-middle attack.

The VNC Server Password

The server password is stored in a password file on the UNIX file system in the ~/.vnc directory. The password is always stored using the same 64-bit key, meaning the password file should be protected using the local file system permissions. Failure to protect the file exposes the password, because the key is consistent across all VNC servers.

The password protection system is the same on the other supported server platforms; however, the location of the password is different.

The VNC source code provides the consistent key:

```
/*
•We use a fixed key to store passwords, since we assume
•that our local file system is secure but nonetheless
•don't want to store passwords as plaintext.
*/

unsigned char fixedkey[8] = {23,82,107,6,35,78,88,7};
```

This fixed key is used as input to the DES functions to encrypt the password; however, the password must be unencrypted at some point to verify authentication

The VNC server creates the ~/.vnc directory using the standard default file permissions as defined with the UNIX system's umask. On most systems, the default umask is 022, making the the ~/.vnc directory accessible to users other than the owner. However, the password file is explicitly set to force read/write permissions only for the file owner; so the chance of an attacker discovering the password is minimized unless the user changes the permissions on the file, or the attacker has gained elevated user or system privileges.

If the password file is readable to unauthorized users, the server password is exposed because the key is consistent and publicly available. However, the attacker does not require too much information, because the functions to encrypt and decrypt the password in the file are included in the VNC source code. With the knowledge of the VNC default password key and access to the VNC server password file, an attacker can obtain the password using 20 lines of C language source code.

A sample C program, here called attack.c, can be used to decrypt the VNC server password should the password file be visible:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <vncauth.h>
```

```
#include <d3des.h>
main( argc, argv)
      int argc;
      char **argv;
{
   char *passwd;
   if (argc < = 1)
      {
      printf ("specify the location and name of a VNC
         password file\n");
      exit(1);
      }
   /* we might have a file */
   passwd = vncDecryptPasswdFromFile(argv[1]);
   printf ("passowrd file is%s\n," argv[1]);
   printf ("password is%s\n," passwd);
   exit(0);
}
```

Note: Do not use this program for malicious purposes. It is provided for education and discussion purposes only.

Running the attack.c program with the location and name of a VNC password file displays the password:

```
[chare@rhlinux libvncauth]$./attack $HOME/.vnc/passwd
passowrd file is/home/chare/.vnc/passwd
password is holycow
```

The attacker can now gain access to the VNC server. Note however, this scenario assumes the attacker already has access to the UNIX system.

For the Microsoft Windows WinVNC, the configuration is slightly different. While the methods to protect the password are the same, WinVNC uses the Windows registry to store the server's configuration information, including passwords. The WinVNC registry entries are found at:

- *Local machine-specific settings:*
  HKEY_LOCAL_MACHINE\Software\ORL\WinVNC3\
- *Local default user settings:*
  HKEY_LOCAL_MACHINE\Software\ORL\WinVNC3\Default
- *Local per-user settings:*
  HKEY_LOCAL_MACHINE\Software\ORL\WinVNC3\<*username*>
- *Global per-user settings:*
  HKEY_CURRENT_USER\Software\ORL\WinVNC3

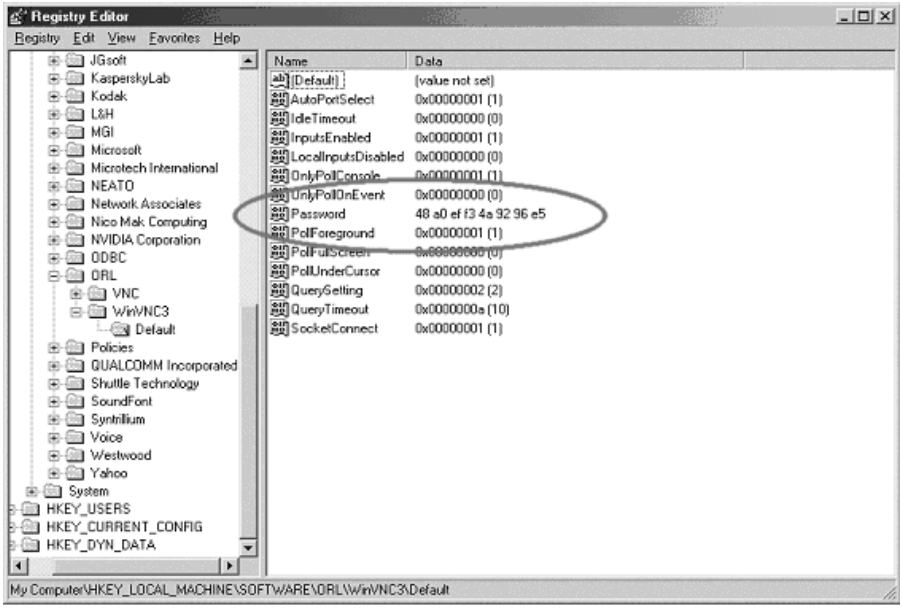

Exhibit 40-9. WinVNC Windows registry values.

The WinVNC server password will be found in the local default user settings area, unless a specific user defines his own server. The password is stored as an individual registry key value as shown in Exhibit 40-9.

Consequently, access to the registry should be as controlled as possible to prevent unauthorized access to the password.

The password stored in the Windows registry uses the same encryption scheme to protect it as on the UNIX system. However, looking at the password shown in Exhibit 40-9, we see the value:

```
48 a0 ef f3 4a 92 96 e5
```

and the value stored on UNIX is:

```
a0 48 f3 ef 92 4a e5 96
```

Comparing these values, we see that the byte ordering is different. However, knowing that the ordering is different, we can use a program to create a binary file on UNIX with the values from the Windows system and then use the attack.c program above to determine the actual password. Notice that because the password values shown in this example are the same, and the encryption used to hide the passwords is the same, the passwords are the same.

Additionally, the VNC password is limited to eight characters. Even if the user enters a longer password, it is truncated to eight. Assuming a good-quality password with 63 potential characters in each position, this represents only 63^8 possible passwords. Even with this fairly large number, the discussion thus far has demonstrated the weaknesses in the authentication method.

RUNNING A VNC SERVER UNDER UNIX

The VNC server running on a UNIX system uses the X Window System to interact with the X-based applications on UNIX. The applications are not aware there is no physical screen attached to the system. Starting a new VNC server is done by executing the command:

```
vncserver
```

on the UNIX host. Because the vncserver program is actually written in Perl, most common problems with starting vncserver are associated with the Perl installation or directory structures.

Any user on the UNIX host can start a copy of the VNC server. Because there is no user authentication built into the VNC server or protocol, running a separate server for each user is the only method of providing limited access. Each vncserver has its own password and port assignment, as presented earlier in the chapter.

The first time a user runs the VNC server, they are prompted to enter a password for the VNC server. Each VNC server started by the same user will have the same password. This occurs because the UNIX implementation of VNC creates a directory called .vnc in the user's home directory. The .vnc directory contains the log files, PID files, password, and X startup files. Should the users wish to change the password for their VNC servers, they can do so using the `vncpasswd` command.

VNC Display Names

Typically the main display for a workstation using the X Window System is display 0 (zero). This means on a system named *ace*, the primary display is ace:0. A UNIX system can run as many VNC servers as the users desire, with the display number incrementing for each one. Therefore, the first VNC server is display ace:1, the second ace:2, etc. Individual applications can be executed and, using the DISPLAY environment variable defined, send their output to the display corresponding to the desired VNC server.

For example, sending the output of an xterm to the second VNC server on display ace:2 is accomplished using the command:

```
xterm -display ace:2 &
```

Normally, the `vncserver` command chooses the first available display number and informs the user what that display is; however, the display number can be specified on the command line to override the calculated default:

```
vncserver :2
```

No visible changes occur when a new VNC server is started, because only a viewer connected to that display can actually see the resulting output from that server. Each time a connection is made to the VNC server, information on the connection is logged to the corresponding server log file found in the $HOME/.vnc directory of the user executing the server. The clog file contents are discussed in the "Logging" section of this chapter.

VNC as a Service

Instead of running individual VNC servers, there are extensions available to provide support for VNC under the Internet Super-Daemon, inetd and xinetd. More information on this configuration is available from the AT&T Laboratories Web site.

VNC AND MICROSOFT WINDOWS

The VNC server is also available for Microsoft Windows, providing an alternative to other commercial solutions and integration between heterogeneous operating systems and platforms. The VNC server under Windows is run as a separate application or a service. Unlike the UNIX implementation, the Windows VNC server can only display the existing desktop of the PC console to the user. This is a limitation of Microsoft Windows, and not WinVNC. WinVNC does not make the Windows system a multi-user environment: if more than one user connects to the Windows system at the same time, they will all see the same desktop.

Running WinVNC as a service is the preferred mode of operation because it allows a user to log on to the Windows system, perform his work, and then log off again.

When running WinVNC, an icon as illustrated in Exhibit 40-10 is displayed. When a connection is made, the icon changes color to indicate there is an active connection.

The WinVNC properties dialog shown in Exhibit 40-11 allows the WinVNC user to change the configuration of WinVNC. All the options are fully discussed in the WinVNC documentation.

With WinVNC running as a service, a user can connect from a remote system even when no user is logged on at the console. Changing the properties for WinVNC when it is running as a service has the effect of changing

Exhibit 40-10. WinVNC system tray icons.

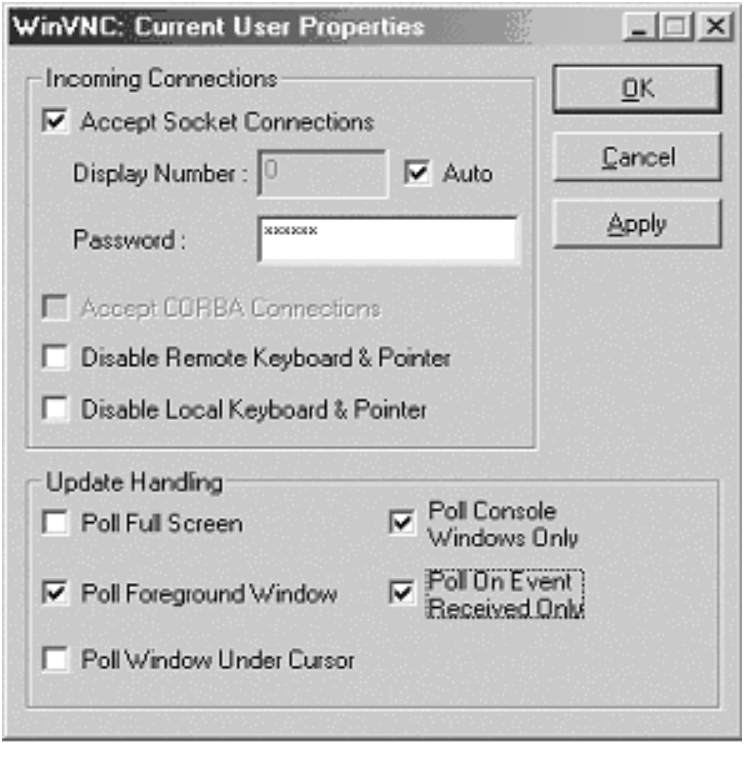

Exhibit 40-11. The WinVNC Properties dialog.

the service configuration, also known as the default properties, rather than the individual user properties. However, running a nonservice mode WinVNC means a user must have logged in on the console and started WinVNC for it to work correctly. Exhibit 40-12 illustrates accessing WinVNC from a Linux system while in service mode.

Aside from the specific differences for configuring the WinVNC server, the password storage and protocol-level operations are the same, regardless of the platform. Because there can be only one WinVNC server running

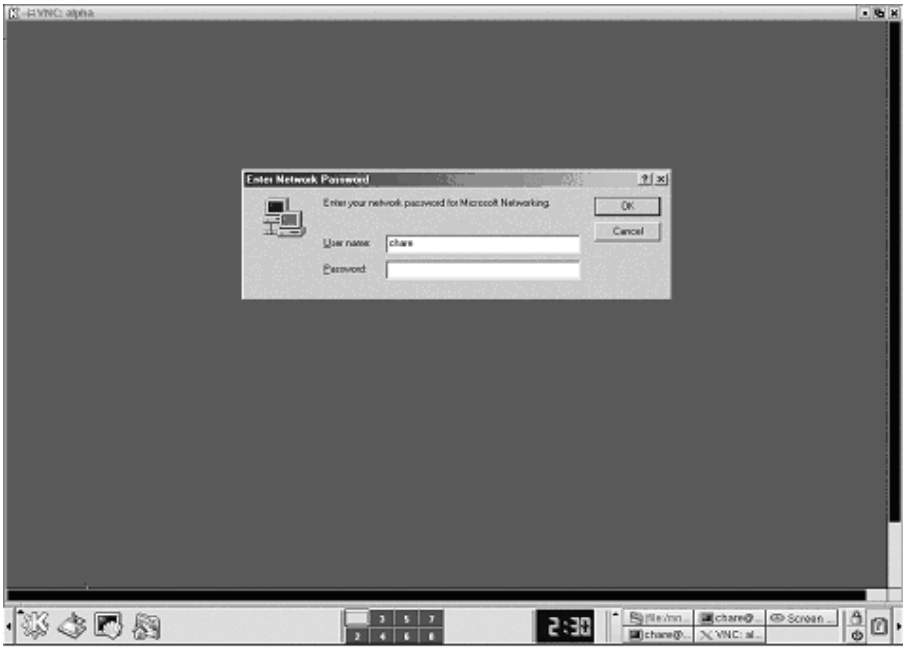

Exhibit 40-12. Accessing WinVNC in service mode.

at a time, connections to the server are on ports 5900 for the VNC viewer and 5800 for the Java viewer.

VNC AND THE WEB

As mentioned previously, each VNC server listens not only on the VNC server port but also on a second port to support Web connections using a Java applet and a Web browser. This is necessary to support Java because a Java applet can only make a connection back to the machine from which it was served.

Connecting to the VNC server using a Java-capable Web browser to:

```
http://ace:5802/
```

loads the Java applet and presents the log-in screen where the password is entered. Once the password is provided, the access controls explained earlier prevail. Once the applet has connected to the VNC server port, the user sees a display resembling that shown in Exhibit 40-13.

With the Java applet, the applications displayed through the Web browser can be manipulated as if they were displayed directly through the VNC client or on the main display of the workstation.

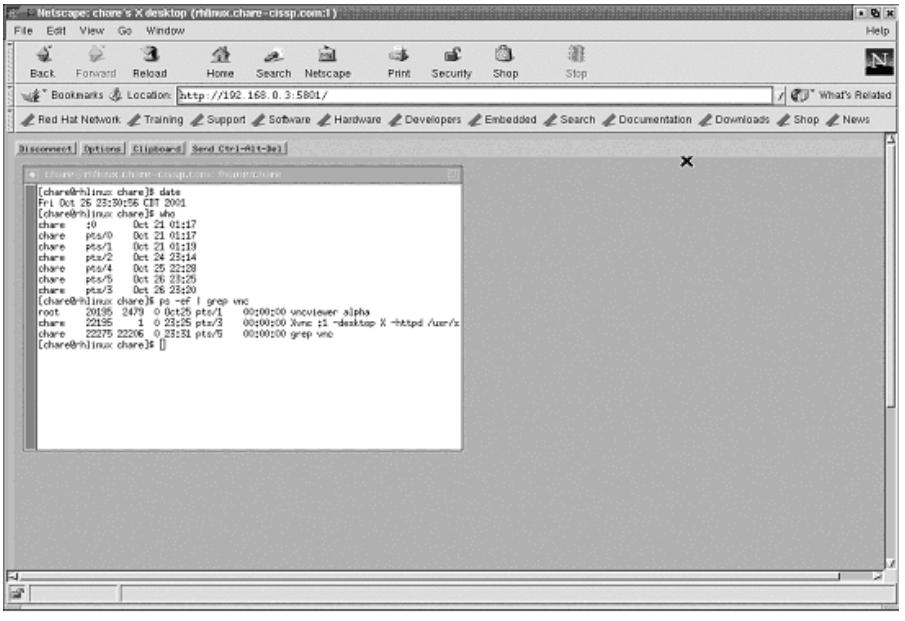

Exhibit 40-13. A VNC connection using a Java-capable Web browser.

LOGGING

As with any network-based application, connection and access logs provide valuable information regarding the operation of the service. The log files from the VNC server provide similar information for debugging or later analysis. A sample log file resembles the following. The first part of the log always provides information on the VNC server, including the listing ports, the client name, display, and the URL.

```
26/10/01 23:25:47 Xvnc version 3.3.3r2
26/10/01 23:25:47 Copyright © AT&T Laboratories
  Cambridge.
26/10/01 23:25:47 All Rights Reserved.
26/10/01 23:25:47 See http://www.uk.research.att.com/
  vnc for information on VNC
26/10/01 23:25:47 Desktop name 'X' (rhlinux.chare-
  cissp.com:1)
26/10/01 23:25:47 Protocol version supported 3.3
26/10/01 23:25:47 Listening for VNC connections on TCP
  port 5901
26/10/01 23:25:47 Listening for HTTP connections on TCP
  port 5801
26/10/01 23:25:47  URL http://rhlinux.chare-
  cissp.com:5801
```

The following sample log entry shows a connection received on the VNC server. We know the connection came in through the HTTPD server from the log entry. Notice that there is no information regarding the user who is accessing the system — only the IP address of the connecting system.

```
26/10/01 23:28:54 httpd: get `` for 192.168.0.2
26/10/01 23:28:54 httpd: defaulting to `index.vnc'
26/10/01 23:28:56 httpd: get `vncviewer.jar' for
  192.168.0.2
26/10/01 23:29:03 Got connection from client 192.168.0.2
26/10/01 23:29:03 Protocol version 3.3
26/10/01 23:29:03 Using hextile encoding for client
  192.168.0.2
26/10/01 23:29:03 Pixel format for client 192.168.0.2:
26/10/01 23:29:03  8 bpp, depth 8
26/10/01 23:29:03  true colour: max r 7 g 7 b 3, shift
  r 0 g 3 b 6
26/10/01 23:29:03 no translation needed
26/10/01 23:29:21 Client 192.168.0.2 gone
26/10/01 23:29:21 Statistics:
26/10/01 23:29:21  key events received 12, pointer
  events 82
26/10/01 23:29:21  framebuffer updates 80, rectangles
  304, bytes 48528
26/10/01 23:29:21    hextile rectangles 304, bytes 48528
26/10/01 23:29:21  raw bytes equivalent 866242,
  compression ratio 17.850354
```

The log file contains information regarding the connection with the client, including the color translations. Once the connection is terminated, the statistics from the connection are logged for later analysis, if required.

Because there is no authentication information logged, the value of the log details for a security analysis are limited to knowing when and from where a connection was made to the server. Because many organizations use DHCP for automatic IP address assignment and IP addresses may be spoofed, the actual value of knowing the IP address is reduced.

WEAKNESSES IN THE VNC AUTHENTICATION SYSTEM

We have seen thus far several issues that will have the security professional concerned. However, these can be alleviated as discussed later in the chapter. There are two primary concerns with the authentication. The first is the man-in-the-middle attack, and the second is a cryptographic attack to uncover the password.

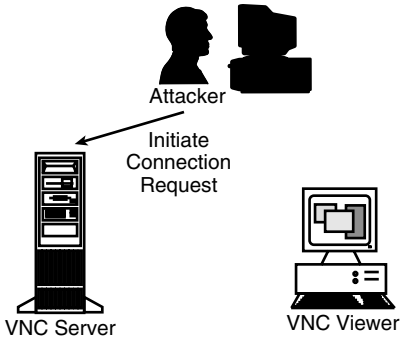

Exhibit 40-14. Attacker opens connection to VNC server.

The Random Challenge

The random challenge is generated using the rand(3) function in the C programming language to generate random numbers. The random number generator is initialized using the system clock and the current system time. However, the 16-byte challenge is created by successive calls to the random number generator, decreasing the level of randomness on each call. (Each call returns 1 byte or 8 bits of data.)

This makes the challenge predictable and increases the chance an attacker could establish a session by storing all captured responses and their associated challenges. Keeping track of each challenge–response pair can be difficult and, as discussed later, not necessary.

The Man-in-the-Middle Attack

For the purposes of this illustration, we will make use of numerous graphics to facilitate understanding this attack method. The server is system S, the client is C, and the attacker, or man in the middle, is A. (This discussion ignores the possibility the network connection may be across a switched network, or that there are ways of defeating the additional security provided by the switched network technology.)

The attacker A initiates a connection to the server, as seen in Exhibit 40-14. The attacker connects, and the two systems negotiate the protocols supported and what will be used. The attacker observes this by sniffing packets on the network.

We know both the users at the client and server share the DES key, which is the password. The attacker does not know the key. The password is used for the DES encryption in the challenge–response.

The server then generates the 16-byte random challenge and transmits it to the attacker, as seen in Exhibit 40-15. Now the attacker has a session established with the server, pending authorization.

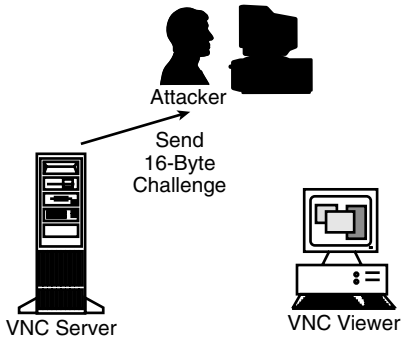

Exhibit 40-15. Server sends challenge to attacker.

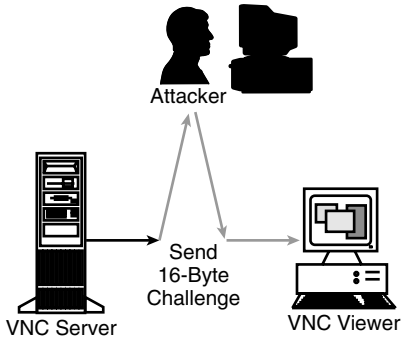

Exhibit 40-16. Attacker captures and replaces challenge.

At this point, the attacker simply waits, watching the network for a connection request to the same server from a legitimate client. This is possible as there is no timeout in the authentication protocol; consequently, the connection will wait until it is completed.

When the legitimate client attempts a connection, the server and client negotiate their protocol settings, and the server sends the challenge to the client as illustrated in Exhibit 40-16. The attacker captures the authentication request and changes the challenge to match the one provided to him by the server.

Once the attacker has modified the challenge, he forges the source address and retransmits it to the legitimate client. As shown in Exhibit 40-17, the client then receives the challenge, encrypts it with the key, and transmits the response to the server.

The server receives two responses: one from the attacker and one from the legitimate client. However, because the attacker replaced the challenge

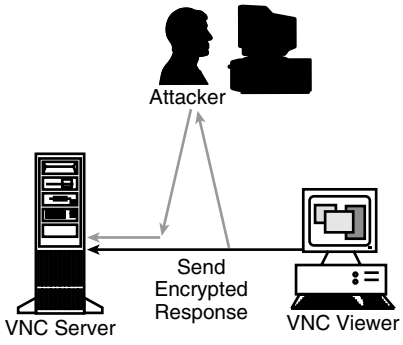

Exhibit 40-17. Attacker and client send encrypted response.

sent to the client with his own challenge, the response sent by the client to server does not match the challenge. Consequently, the connection request from the legitimate client is refused.

However, the response sent does match the challenge sent by the server to the attacker; and when the response received from the attacker matches the calculated response on the server, the connection is granted. The attacker has gained unauthorized access to the VNC server.

Cryptographic Attacks

Because the plaintext challenge and the encrypted response can both be retrieved from the network, it is possible to launch a cryptographic attack to determine the key used, which is the server's password. This is easily done through a brute-force or known plaintext attack.

A brute-force attack is the most effective, albeit time-consuming, method of attack. Both linear cryptanalysis, developed by Lester Mitsui, and differential cryptanalysis, developed by Biham and Shamir, are considered the two strongest analytic (shortcut) methods for breaking modern ciphers; and even these have been shown as not very practical, even against Single-DES.

The known plaintext attack is the most advantageous method because a sample of ciphertext (the response) is available as well as a sample of the plaintext (the challenge). Publicly available software such as *crack* could be modified to try a dictionary and brute-force attack by repeatedly encrypting the challenge until a match for the response is found. The nature of achieving the attack is beyond the scope of this chapter.

Finding VNC Servers

The fastest method of finding VNC servers in an enterprise network is to scan for them on the network devices. For example, the popular *nmap*

scanner can be configured to scan only the ports in the VNC range to locate the systems running it.

```
[root@rhlinux chare]# nmap -p "5500,5800-5999"
  192.168.0.1-5
Starting nmap V. 2.54BETA29 (www.insecure.org/nmap/)
All 201 scanned ports on gateway (192.168.0.1) are:
  filtered
Interesting ports on alpha (192.168.0.2):
(The 199 ports scanned but not shown below are in state:
  closed)
Port       State   Service
5800/tcp   open    vnc
5900/tcp   open    vnc

Interesting ports on rhlinux.chare-cissp.com
  (192.168.0.3):
(The 199 ports scanned but not shown below are in state:
  closed)
Port       State   Service
5801/tcp   open    vnc
5901/tcp   open    vnc-1

Nmap run completed — 5 IP addresses (3 hosts up) scanned
  in 31 seconds
[root@rhlinux chare]#
```

There are other tools available to find and list the VNC servers on the network; however, *nmap* is fast and will identify not only if VNC is available on the system at the default ports but also all VNC servers on that system.

Improving Security through Encapsulation

To this point we have seen several areas of concern with the VNC environment:

- There is no user-level authentication for the VNC server.
- The challenge–response system is vulnerable to man-in-the-middle and cryptographic attacks.
- There is no data confidentiality built into the client and server.

Running a VNC server provides the connecting user with the ability to access the entire environment at the privilege level for the user running the server. For example, assuming root starts the first VNC server on a UNIX system, the server listens on port 5901. Any connections to this port where the remote user knows the server password result in a session with root privileges.

We have seen how it could be possible to launch a man-in-the-middle or cryptographic attack against the authentication method used in VNC.

Additionally, once the authentication is completed, all the session data is unencrypted and could, in theory, be captured, replayed, and watched by malicious users. However, because VNC uses a simple TCP/IP connection, it is much easier to add encryption support with Secure Sockets Layer (SSL) or Secure Shell (SSH) than, say, a Telnet, rlogin, or X Window session.

Secure Shell (SSH) is likely the more obvious choice for most users, given there are clients for most operating systems. SSH encrypts all the data sent through the tunnel and supports port redirection; thus, it can be easily supported with VNC. Furthermore, while VNC uses a very efficient protocol for carrying the display data, additional benefits can be achieved at slower network link speeds because SSH can also compress the data.

There are a variety of SSH clients and servers available for UNIX, although if you need an SSH server for Windows, your options are very limited and may result in the use of a commercial implementation. However, SSH clients for Windows and the Apple Macintosh are freely available. Additionally, Mindbright Technology offers a modified Java viewer supporting SSL.

Because UNIX is commonly the system of choice for operating a server, this discussion focuses on configuring VNC with SSH using a UNIX-based system. Similar concepts are applicable for Windows-based servers, once you have resolved the SSH server issue. However, installing and configuring the base SSH components are not discussed in this chapter.

Aside from the obvious benefits of using SSH to protect the data while traveling across the insecure network, SSH can compress the data as well. This is significant if the connection between the user and the server is slow, such as a PPP link. Performance gains are also visible on faster networks, because the compression can make up for the time it takes to encrypt and decrypt the packets on both ends.

A number of extensions are available to VNC, including support for connections through the Internet superserver inetd or xinetd. These extensions mean additional controls can be implemented using the TCP Wrapper library. For example, the VNC X Window server, Xvnc, has been compiled with direct support for TCP Wrappers.

More information on configuring SSH, inetd, and TCP Wrappers is available on the VNC Web site listed in the “References” section of this chapter.

SUMMARY

The concept of thin-client computing will continue to grow and develop to push more and more processing to centralized systems. Consequently, applications such as VNC will be with the enterprise for some time. However, the thin-client application is intended to be small, lightweight, and

easy to develop and transport. The benefits are obvious — smaller footprint on the client hardware and network, including support for many more devices including handheld PCs and cell phones, to name a few.

However, the thin-client model has a price; and in this case it is security. While VNC has virtually no security features in the protocol, other add-on services such as SSH, VNC, and TCP Wrapper, or VNC and xinetd provide extensions to the basic VNC services to provide access control lists limited by the allowable network addresses and data confidentiality and integrity.

Using VNC within an SSH tunnel can provide a small, lightweight, and secured method of access to that system 1000 miles away from your office. For enterprise or private networks, there are many advantages to using VNC because the protocol is smaller and more lightweight than distributing the X Window system on Microsoft Windows, and it has good response time even over a slower TCP/IP connection link. Despite the security considerations mentioned in this chapter, there are solutions to address them; so you need not totally eliminate the use of VNC in your organization.

References

1. CORE SDI advisory: weak authentication in AT&T's VNC, http://www.uk.research.att.com/vnc/archives/2001–01/0530.html.
2. VNC Computing Home Page, http://www.uk.research.att.com/vnc/index.html.
3. VNC Protocol Description, http://www.uk.research.att.com/vnc/rfbproto.pdf.
4. VNC Protocol Header, http://www.uk.research.att.com/vnc/rfbprotoheader.pdf.
5. VNC Source Code, http://www.uk.research.att.com/vnc/download.html.

ABOUT THE AUTHOR

Chris Hare, CISSP, CISA, is an information security and control consultant with Nortel Networks in Dallas, Texas. A frequent speaker and author, his experience includes from application design, quality assurance, systems administration and engineering, network analysis, and security consulting, operations, and architecture.