

Remote Desktop Management Solution for Microsoft

Artur Maj 2003-03-18

One of the many challenges facing Microsoft administrators is how to manage remote systems in a secure manner? In the world of the UNIX the answer is quite simple: using the SSH protocol is sufficient. Thanks to the SSH, we can manage remote systems not only in the text mode, but we can also run remote X-Window applications by using the protocol tunneling technique. And all of that by using strong cryptography, which protects transmitted data from unauthorized access.

Unfortunately, providing secure remote access to the MS Windows systems is not as easy. Why? First of all, only the NT Terminal Server, 2000 Server and XP are equipped with remote management services (Terminal Services). Secondly, the solutions that offer remote MS Windows management possibilities either don't encrypt transmitted data (like VNC) or their implementation often comes hand in hand with the additional, significant costs.

This article will describe the universal method of remote management that can be used to manage almost all versions of MS Windows systems: from Windows 95 up to XP. This method is characterized not only by minimal costs, but also by a relatively high level of security.

The Solution

What features should a remote management solution have? First of all, the solution must be functional. Although in the case of Unix systems, access to the emulated text terminal is often sufficient, the use of such methods to manage MS Windows is far from ideal. Because the MS Windows is a system based on a graphics environment, remote management should be also realized in a graphics mode. Besides being functional, remote management must also be secure. The solution must not only provide user authentication, but must also assure confidentiality and integrity of the transmitted data.

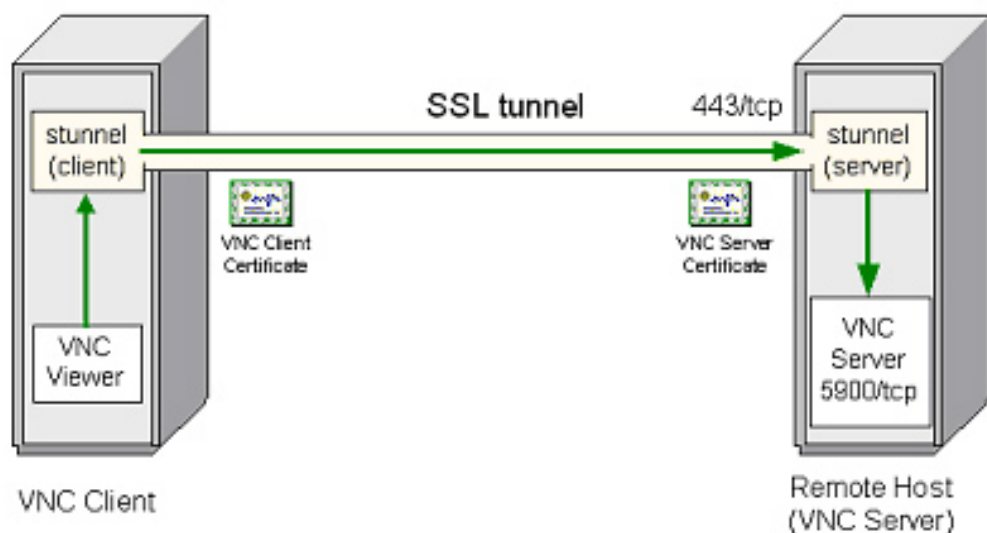
In the remote management solution that will be presented in this discussion, all the above requirements will be met by using the following open-source software:

- VNC - VNC (Virtual Network Computing) provides graphics management of remote systems. In our case, the VNC software will be the "core" of the whole solution. It will provide a graphics console to the remote MS Windows system.
- Stunnel - The main purpose of the Stunnel utility is to create SSL tunnels that can be used to transmit other, often non-encrypted protocols in a secure manner. In the described

solution, this tool will be used to secure the VNC protocol. Thanks to the Stunnel, it will be possible to assure not only confidentiality and integrity of the transmitted data, but also to authenticate VNC clients and servers by certificates.

- OpenSSL - OpenSSL is a library of cryptographic functions that can be used to enrich applications by data encrypting functions. By using OpenSSL we can also generate, sign and revoke certificates that can be used in solutions based on a public key infrastructure (PKI). In the method presented below this tool will be used to generate and sign certificates needed to authenticate both VNC clients and servers.

The following picture shows the way the software mentioned above will be used to provide secure management of remote desktops:



Now, let's proceed to the practical implementation of the described solution.

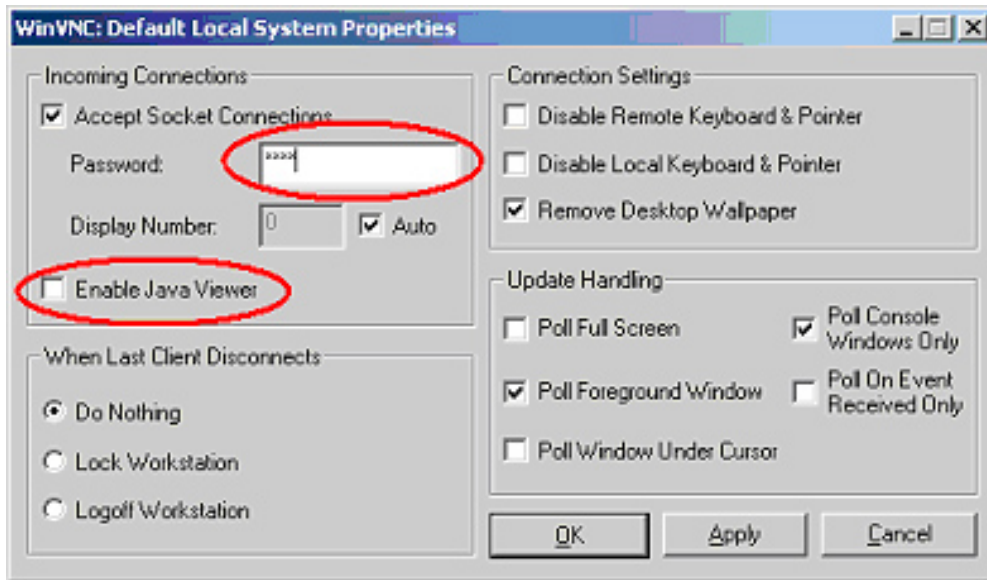
Installing the Software

The first stage of implementing this secure remote management solution is the installation of the software.

VNC

In order to use the VNC, we must [download](#) it and install it on the host that we want to manage remotely, which we will hereafter refer to as the VNC server. Next, we must register the VNC service (Start Menu ® RealVNC ® VNC Server ® Register VNC Server Service) and reboot the system.

After rebooting the system, we must set up the basic parameters of the VNC service. The most important thing is to enter an effective password, which will protect the VNC service against unauthorized access. The next step is turning off the "Enable Java Viewer" option (since this option requires two separate SSL tunnels, it won't be used), as is shown in the image below.



After we finish configuring the VNC server, we should download the VNC client software (vncviewer.exe) and place it on the host that will be the client of the VNC.

At this point we should check if the VNC client can establish a connection to the VNC server. If the programs are able to communicate with each other, we can finish the configuration.

Because the VNC server should be accessible only by a locally installed Stunnel utility, the following entry should be added to the Windows Registry on the VNC server:

```
Key:      HKEY_LOCAL_MACHINE\Software\ORL\WinVNC3Name:      LoopbackOnly
Type:     REG_DWORD
Value:    1
```

The above entry makes it possible to use loopback connections, and it limits listening on the 5900/tcp port only to the localhost (127.0.0.1). Thanks to that, the VNC server will not be directly accessible from the computer network. In addition, if we don't want users to shut down the VNC service on the VNC server host, the following entry should be added to the Registry:

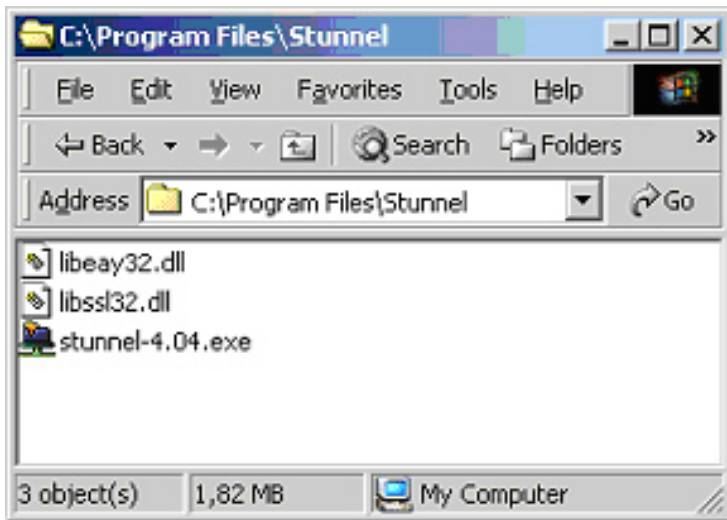
```
Key:      HKEY_LOCAL_MACHINE\Software\ORL\WinVNC3\Default
Name:     AllowShutdown
Type:     REG_DWORD
```

Value: 0

In order to activate the above changes, we must restart the VNC service.

Stunnel

The next step is installing the Stunnel utility. In order to perform that, we should [download it](#) and place it on the VNC server and client, in the directory: C:\Program Files\Stunnel. We should also download two libraries that are required by Stunnel: libeay32.dll, libssl32.dll.



If we want the Stunnel to start automatically when the system boots up, the following entry should be added to the Windows Registry:

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Name: Stunnel

Type: REG_SZ

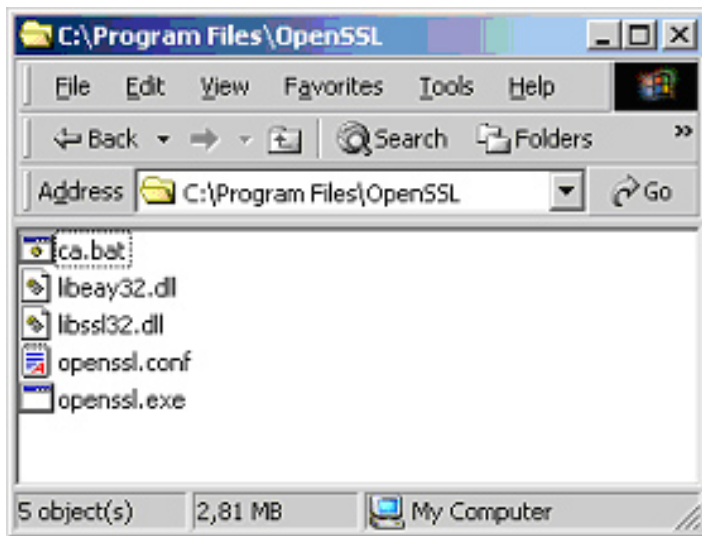
Value: "C:\Program Files\Stunnel\stunnel-4.04.exe"

OpenSSL

At the present times the OpenSSL library is installed by default in most Linux distributions, mainly because it is required by OpenSSH. However, a few people know that there is a version of OpenSSL ported to MS Windows that has almost identical functionality. Because the article is devoted to the MS Windows platform, we'll use this version of OpenSSL.

In order to install and configure the OpenSSL software, we must perform the following steps:

1. On the additional, trusted host (Windows 2000, NT or XP) - if possible, not connected to the computer network at all - we should install the OpenSSL software. The binary version of the OpenSSL (openssl.exe) can be downloaded from the [Stunnel](#) Web site. Just like in case of the Stunnel program, we also have to download two libraries: libeay32.dll and libssl32.dll. The downloaded software must be placed in the C:\Program Files\OpenSSL directory.
2. Two other files must also be downloaded: the configuration file, [openssl.conf](#) and the [ca.bat script](#), which will be used to generate certificates. The above files should be placed in C:\Program Files\OpenSSL directory. The final content of that directory should be similar to the following:



The next step is to generate certificates, which will be used to authenticate VNC servers and clients.

Keys and Certificates Generation

Certification Authority

The process of generating certificates should be started by generating a private/public key pair and certificate for the trusted third party, or CA (Certification Authority). The CA's private key will be used later to sign the VNC server's and client's certificates. The CA's certificate will be placed on all VNC servers and clients. Because the CA's private key is one of the most important elements of every PKI implementation, the key must be protected by strong pass phrase and kept away from regular users.

In order to generate the public/private key pair and certificate for the CA, we should run the ca.bat script in the following manner:

C:\progra~1\OpenSSL\ca genca

```

Command Prompt

C:\>c:\progra~1\openssl\ca genca

Simple CA utility
Written by Artur Maj (artur.maj@seccure.net)

The script will create the C:\CA directory structure needed by
OpenSSL, and generate CA Certificate.

Press CTRL-C to break, or ENTER to continue...

Creating C:\CA...
Creating C:\CA\certs...
Creating C:\CA\crl...
Creating C:\CA\newcerts...
Creating C:\CA\private...
Creating C:\CA\temp...
Creating CA index file...
Creating CA serial file...

-----
Generating the CA's certificate...
-----

Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'C:\CA\private\CAkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PL
State or Province Name (full name) [Some-State]:Warsaw
Locality Name (eg, city) []:Warsaw
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Seccure
Organizational Unit Name (eg, section) []:Seccure Labs
Common Name (eg, YOUR name) []:CA
Email Address []:ca@seccure.lab

C:\>_

```

After performing the above steps, the CA's certificate will be stored in the C:\CA\CAcert.pem file, and the private/public key pair will be stored in the C:\CA\private\CAkey.pem file.

VNC Server

The next step is to generate private/public key pair and certificate for the VNC server:

C:\progra~1\OpenSSL\ca server

```

Command Prompt

C:\>c:\progra~1\openssl\ca server

Simple CA utility
Written by Artur Maj (artur.maj@seccure.net)

-----
Step 1: Generate the keys and the certificate request
-----

Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'C:\CA\temp\vnc_server\server.key'

-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----
Country Name (2 letter code) [AU]:PL
State or Province Name (full name) [Some-State]:Warsaw
Locality Name (eg, city) []:Warsaw
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Seccure
Organizational Unit Name (eg, section) []:Seccure Labs
Common Name (eg, YOUR name) []:UNC Server
Email Address []:bilbo@seccure.lab

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

-----
Step 2: Sign the certificate
-----

Using configuration from C:\Progra~1\OpenSSL\openssl.conf
Loading 'screen' into random state - done
Enter pass phrase for C:\CA\private\CAkey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:' PL'
stateOrProvinceName   :PRINTABLE:' Warsaw'
localityName          :PRINTABLE:' Warsaw'
organizationName      :PRINTABLE:' Seccure'
organizationalUnitName:PRINTABLE:' Seccure Labs'
commonName            :PRINTABLE:' UNC Server'
emailAddress          :IA5STRING:' bilbo@seccure.lab'
Certificate is to be certified until Feb 22 20:49:30 2004 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Data Base Updated

C:\>

```

As the result, the following files will be created in the C:\CA\temp\vnc_server directory:

- server.key - private/public key pair
- server.crt - server's certificate

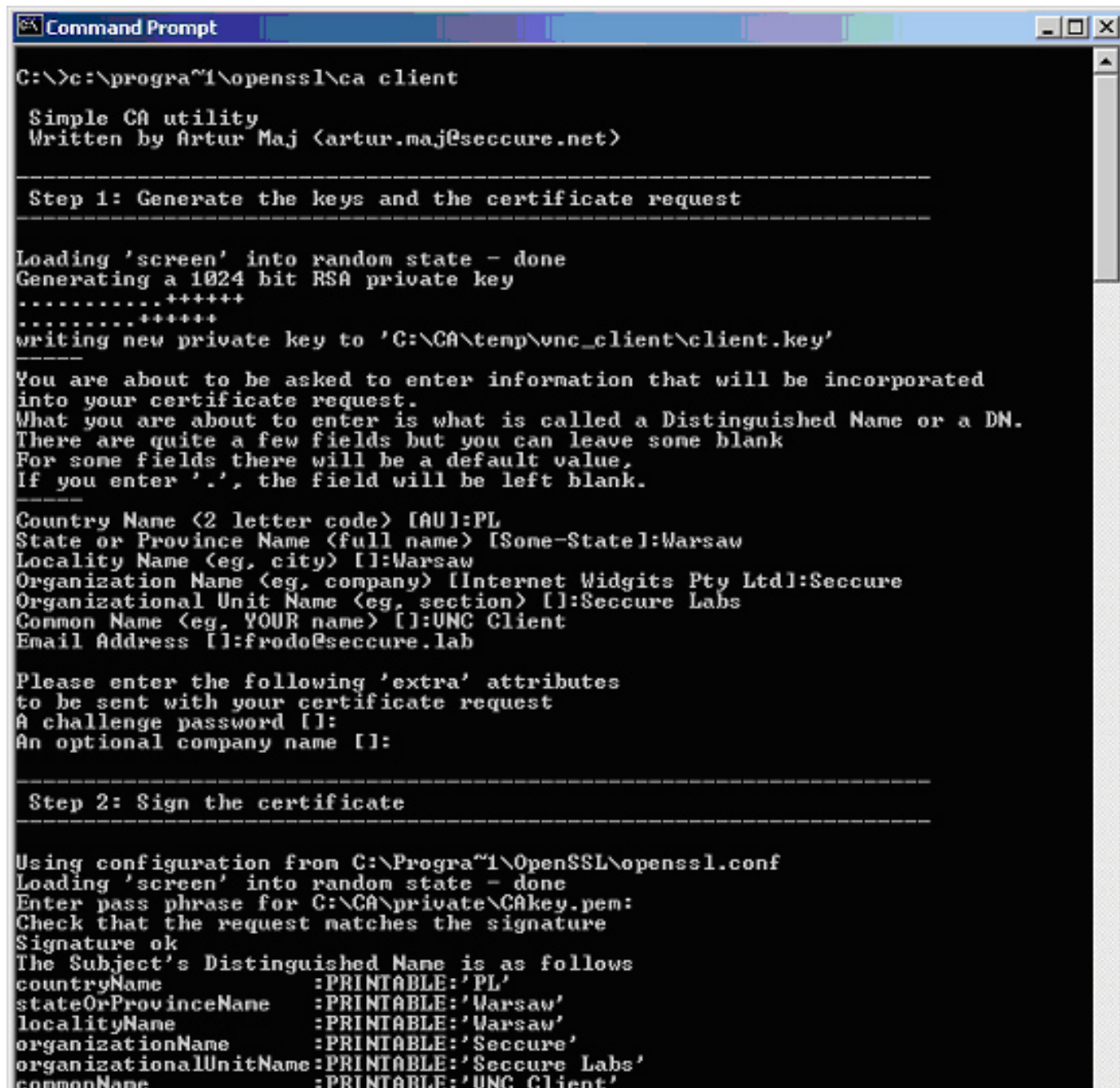
- server.pem - server.key + server.crt (required by Stunnel)

It is worth to emphasize that the server's private key is not secured by the pass phrase. The option of protecting the private key by the pass phrase hasn't been used at this point, mainly because of the Stunnel, which doesn't have the possibility to supply pass phrases. Thus, private keys secured by pass phrases cannot be used by the Stunnel utility.

VNC Client

The last step is to generate the public/private key pair and certificate for the VNC client:

```
C:\progra~1\OpenSSL\ca client
```



```
Command Prompt
C:\>c:\progra~1\openssl\ca client

Simple CA utility
Written by Artur Maj <artur.maj@seccure.net>

-----
Step 1: Generate the keys and the certificate request
-----

Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'C:\CA\temp\vnc_client\client.key'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----
Country Name (2 letter code) [AU]:PL
State or Province Name (full name) [Some-State]:Warsaw
Locality Name (eg, city) []:Warsaw
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Seccure
Organizational Unit Name (eg, section) []:Seccure Labs
Common Name (eg, YOUR name) []:UNC Client
Email Address []:frodo@seccure.lab

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

-----
Step 2: Sign the certificate
-----

Using configuration from C:\Progra~1\OpenSSL\openssl.conf
Loading 'screen' into random state - done
Enter pass phrase for C:\CA\private\CAkey.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName          :PRINTABLE:'PL'
stateOrProvinceName  :PRINTABLE:'Warsaw'
localityName         :PRINTABLE:'Warsaw'
organizationName     :PRINTABLE:'Seccure'
organizationalUnitName :PRINTABLE:'Seccure Labs'
commonName           :PRINTABLE:'UNC Client'
```

```

organizationName      :PRINTABLE:'Seccure'
organizationalUnitName:PRINTABLE:'Seccure Labs'
commonName            :PRINTABLE:'UNC Client'
emailAddress          :IA5STRING:'frodo@seccure.lab'
Certificate is to be certified until Feb 22 20:55:12 2004 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

C:\>

```

Just like in the previous step, the following files will be created in the C:\CA\temp\vnc_client directory:

- client.key - private/public key pair
- client.crt - client's certificate
- client.pem - client.key + client.crt (required by Stunnel)

Stunnel Configuration

VNC Server

Before we try to establish a secure connection between the VNC server and client, we must configure the Stunnel utility, and equip it with all the required keys and certificates.

In order to perform that, we should create a "C:\Program Files\Stunnel\stunnel.conf" file with the following content:

```

CAfile = CAcert.pem
CApath = certificates
cert = server.pem
client = no
verify = 3

[vnc]
accept = 443
connect = 127.0.0.1:5900

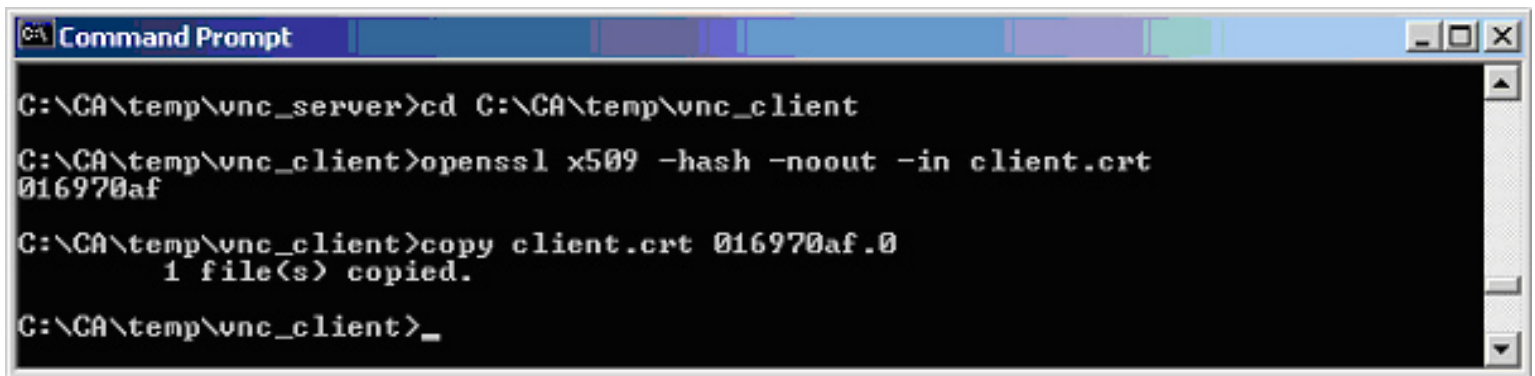
```

The above configuration will cause all incoming connections to the 443/tcp port to be forwarded to the local port 5900/tcp. Of course, this will be done only when the client proves his identity by presenting a valid, signed certificate, which must also be present in the local "certificates" directory ("verify = 3" enforces certificate authentication of both sides).

The next step is to place both the CA's certificate (C:\CA\CAcert.pem) and VNC server's private/public key pair and certificate (C:\CA\temp\vnc_server\server.pem) in the C:\Program Files\Stunnel directory.

Finally we must also load the VNC client's certificate. In order for the Stunnel utility to find the certificate during the authentication process, we must change its name as follows (the following commands must be run on the server on which the certificates was generated; the *value* should be replaced by the result of the "openssl x509" command):

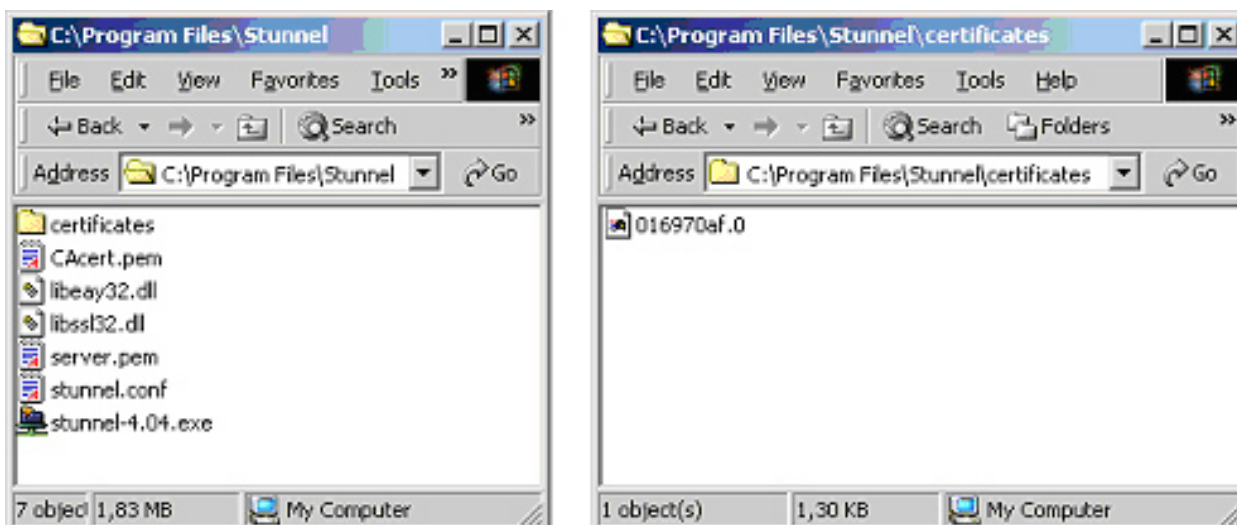
```
cd C:\CA\temp\vnc_client
C:\progra~1\openssl\openssl x509 -hash -noout -in client.crt
value
copy client.crt value.0
```



```
Command Prompt
C:\CA\temp\vnc_server>cd C:\CA\temp\vnc_client
C:\CA\temp\vnc_client>openssl x509 -hash -noout -in client.crt
016970af
C:\CA\temp\vnc_client>copy client.crt 016970af.0
1 file(s) copied.
C:\CA\temp\vnc_client>_
```

The file *value.0* should be placed in the C:\Program Files\Stunnel\certificates directory.

The final content of the C:\Program Files\Stunnel directory should be similar to the following:



VNC Client

Generally, the configuration process of the Stunnel utility installed on the VNC client host is very similar to the one, described in the previous step.

First, we must create a "C:\Program Files\Stunnel\stunnel.conf" file with the following content:

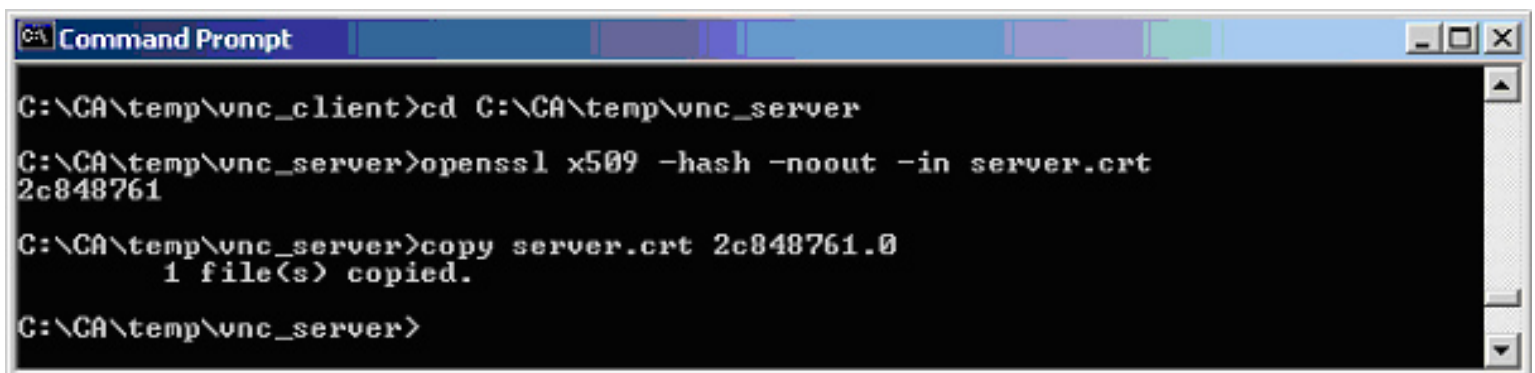
```
CAfile = CAcert.pem
CApath = certificates
cert = client.pem
client = yes
verify = 3

[vnc]
accept = 127.0.0.1:5900
connect = VNC_server_IP_address:443
```

The next step is to store both the CA's Certificate (C:\CA\CAcert.pem) and VNC client's private/public key pair and certificate (C:\CA\temp\vnc_client\client.pem) in the C:\Program Files\Stunnel directory.

Finally, we must change the name of the VNC server's certificate file in the way as follows:

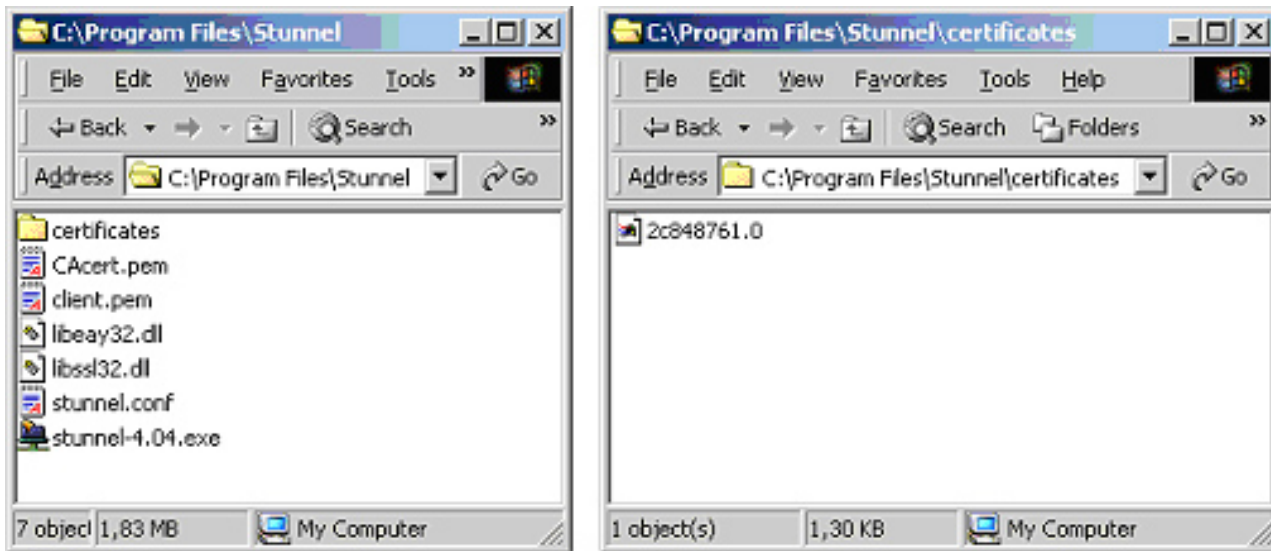
```
cd C:\CA\temp\vnc_server
C:\progra~1\openssl\openssl x509 -hash -noout -in server.crt
value
copy server.crt value.0
```



```
Command Prompt
C:\CA\temp\vnc_client>cd C:\CA\temp\vnc_server
C:\CA\temp\vnc_server>openssl x509 -hash -noout -in server.crt
2c848761
C:\CA\temp\vnc_server>copy server.crt 2c848761.0
1 file(s) copied.
C:\CA\temp\vnc_server>
```

and store it into the C:\program files\Stunnel\certificates directory.

The final content of the C:\Program Files\Stunnel directory should be similar to the following:



Testing the Connection

At this point all the software is configured and ready to use. In order to test it, we must run the Stunnel utility both on the server and client, and run the VNC service.

Then, on the VNC client host we must run vncviewer.exe. As the remote address we should enter: 127.0.0.1. If everything is configured correctly, the connection with the VNC server should be established, and both Stunnel tools should show the following information:

On the VNC server:

```

stunnel 4.04 on Win32
File Help
.5299]: stunnel 4.04 on x86-pc-mingw32-gnu WIN32 with OpenSSL 0.9.7 31 Dec 2002
!0983]: Peer certificate location certificates
!0983]: WIN32 platform: 30000 clients allowed
!1571]: vnc connected from 10.10.12.50:1250
!1571]: VERIFY OK: depth=1, /C=PL/ST=Warsaw/L=Warsaw/O=Seccure/OU=Seccure Labs/CN=CA/emailAdd:
!1571]: VERIFY OK: depth=0, /C=PL/ST=Warsaw/L=Warsaw/O=Seccure/OU=Seccure Labs/CN=UNC Client/

```

On the VNC client:

```

stunnel 4.04 on Win32
File Help
l:676]: stunnel 4.04 on x86-pc-mingw32-gnu WIN32 with OpenSSL 0.9.7 31 Dec 2002
l:664]: Peer certificate location certificates
l:664]: WIN32 platform: 30000 clients allowed
l:1040]: unc connected from 127.0.0.1:1257
l:1040]: VERIFY OK: depth=1, /C=PL/ST=Warsaw/L=Warsaw/O=Seccure/OU=Seccure Labs/CN=CA/emailAdd
l:1040]: VERIFY OK: depth=0, /C=PL/ST=Warsaw/L=Warsaw/O=Seccure/OU=Seccure Labs/CN=UNC Server/

```

If, for some unknown reason, the attempt to establish a connection fails, we should increase the Stunnel log level, and try to find the reason of failure. In order to perform that, the following global directive should be added to the stunnel.conf file:

```
debug = 7
```

Then we should restart Stunnel and try to establish connection again.

Reverse Connections

The above method works fine, but only when the VNC server host has a valid, external IP address or is placed in the same LAN, as the VNC client host. But what if the VNC server is placed beyond the NAT or incoming connections to this host are dropped by the firewall?

It appears that, thanks to the "/listen" option of the VNC client, it is possible to omit such limitations. In the traditional client-server technology, the client initiates the TCP/IP connection. However, nothing stands in the way. Not the client, but the server initiates the connection. The only condition is that the server must be able to connect to the client. In practice, this means that the client host cannot be placed beyond the NAT, and eventual firewalls must not block the incoming connection. Of course, the software must also be written in a way that makes it possible to perform such operations.

As I have mentioned before, the VNC has the possibility to establish reverse connections. In order to make a use of that option, the following changes should be applied to the stunnel.conf file on the VNC server:

```
CAfile = CAcert.pem
CApath = certificates
cert = server.pem
client = yes
```

```
verify = 3
```

```
[vnc]
```

```
accept = 127.0.0.1:5500
```

```
connect = VNC_client_IP_address:443
```

and on the VNC client:

```
CAfile = CAcert.pem
```

```
CApath = certificates
```

```
cert = client.pem
```

```
client = no
```

```
verify = 3
```

```
[vnc]
```

```
accept = 443
```

```
connect = 127.0.0.1:5500
```

It is worth noting that roles of Stunnel utilities are now inverted. The Stunnel on the server side becomes an SSL client, and the Stunnel on the client side - an SSL server.

There is also a change in the way of establishing a connection by the VNC software. In this method, the vncviewer.exe should be run first, in listening mode (Start Menu ® RealVNC ® VNC Viewer ® Run Listening VNC Viewer). Then, on the VNC server, we must use the "Add New Client" option as follows:

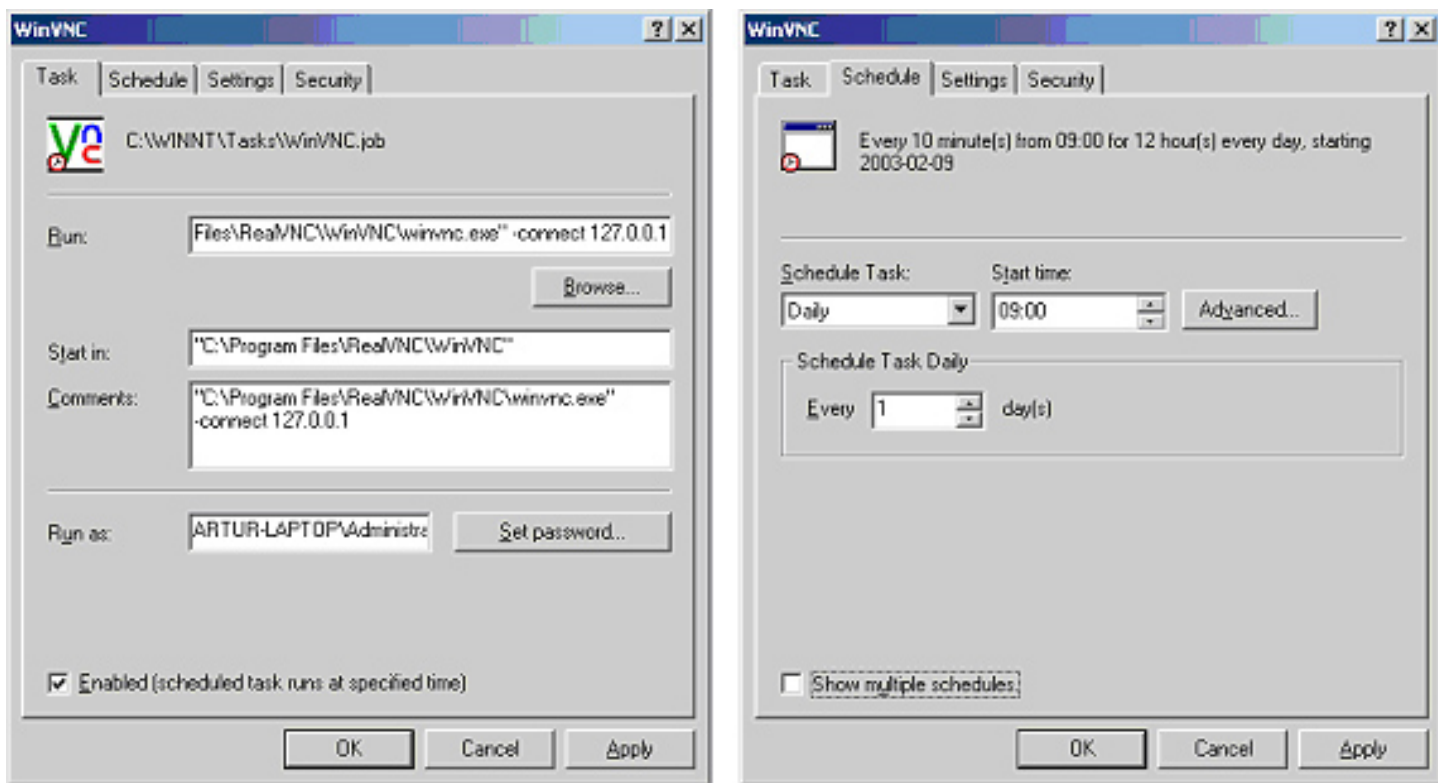


After performing these steps, the connection between the VNC server and the client should be established.

The above solution is a very effective way of omitting NAT limitations; however it has one very

important disadvantage: in order to establish a reverse connection, manual intervention on the server side is required. The question arises whether there is a way to establish such a connection without manual intervention?

It appears that it is possible. In order to perform that, it suffices to use the built into MS Windows operation system "Task Scheduler Service", to solve the problem of manual intervention. The screenshot below shows the example configuration of the "Task Scheduler", in which the VNC server tries to establish a connection with the VNC client every day between 9 a.m. and 9 p.m, in 10-minute intervals. If we want to establish a connection with the VNC server, all we need to do is run the VNC in listening mode and wait until the server connects. In 10 minutes at the most, the graphics console will be "sent" to us.



The method described above is undoubtedly very limited and disadvantageous. Apart from them, however, the method presented is an interesting way to manage a host, to which we cannot establish a direct connection.

Summary

There are a lot of programs for MS Windows remote management. Unfortunately, a large number of them either don't secure transmitted data, or their implementation comes with additional, often significant costs. The method outlined in the discussion above is a free solution for secure remote

MS Windows management. Thanks to the SSL protocol and authentication based on certificates, the described solution has a chance to compete with commercial solutions not only in affordability, but also in effective security.

Relevant Links

[OpenSSL](#)

[Stunnel](#)

[VNC](#)

[Sample openssl.conf](#)

[Simple CA utility](#)

About the author

[Artur Maj](#) works as a Principal Software Engineer for Oracle Corporation, in the EMEA Mobile, Wireless & Voice Center of Expertise. He is experienced in designing computer systems, performing security audits as well as providing security training. He is also author of many articles and publications devoted to securing computer systems and software against intruders.

View [all articles](#) by Artur Maj on SecurityFocus.

Comments or reprint requests can be sent to the [editor](#).

[Privacy Statement](#)

Copyright 2006, SecurityFocus