Global Information Assurance Certification Paper

SECURE COMMUNICATION THROUGH SSH

By Nahzatulshima Binti Zainuddin

30th August 2003

Version 1.4b

Practical Assignment Option 1

Table of Contents

1 Abstract

This paper emphasizes on SSH as an alternative for secure communication. It will discuss on how SSH works and will adapt OpenSSH structure as the default configuration. Besides that, it will also cover the SSH security features as well as its vulnerability. At the end of this paper, I will present a secure email communication plan using OpenSSH and other supporting tools to develop an option of secure communication through email.

SSH is not a new utility in security area. In fact, it has become a talk years ago. However, I find out that in my environment, SSH is normally used for login purposes or transferring files. Other functions of SSH is not fully utilized until recently when the need to develop secure email communication comes to our attention. Then, we start to manipulate SSH for this purposes. This paper aims at providing an alternative of using the public network as the basis of sharing information or as a platform of confidential discussion securely.

2 Introduction

In this era of modern technology, the use of internet has become a popular medium of communication. Organizations, companies, and businesses are now looking for distributed computing environment and the networks continue to grow while the number of users in the network becomes so huge. The need to transfer information across the network becomes essential for everyone.

Nowadays, information is considered as the most valuable business asset and people are very conscious to protect its integrity, availability, and confidentiality. The concern is that transmitting the information through the network is not secure. Someone might eavesdrop on the network to grab, manipulate, or alter the information in transit.

Out of this scenario, we are still in need to use the network to transfer information especially when it involves a wide spread geographical area. Moreover, the network is one of the fastest way of sending information. Therefore, we start to find an alternative in order to use the network in a secure way.

There are many ways available to protect in-transit information from eavesdropper:

- Physically secure the network to prevent eavesdropping
- Hide the valuable information within other medium such as picture, text or etc that appears trivial such as Steganography
- Encrypt the information so that only the owner of the key can decode the message

SSH is an alternative method that can be used to setting up a secure communication using encryption methodology. In fact, encryption is the most practical solution for a large-scale public network.

SSH is the abbreviation used for Secure Shell. It is a protocol developed by Tatu Ylonen which provides secure remote connection to a server [scott]. Unlike, ftp, telnet or rlogin, SSH preserves the privacy of its user by encrypting the traffic between the user station and the server. Thus, it protects the client-server communication from eavesdropping, connection hijacking, and other network-level attacks [open].

There are two different versions of SSH known as SSH protocol version 1(SSH1) and SSH protocol version 2(SSH2). The major difference of SSH2 from SSH1 is its mechanism for message integrity. SSH2 uses either SHA-1 or MD5 where else SSH1 uses the insecure CRC [scott]. Other than that, there are conceptually the same. SHA-1, MD5 and CRC are algorithms used to encrypt the message being transit. However, CRC has been proven to have vulnerability in security aspect.

There are several types of SSH tool available on the internet. One popular tool is OpenSSH which is based on the OpenBSD version of SSH [scott]. This can be freely downloaded at http://www.openssh.org. According to the site, *"OpenSSH is a **FREE** version of the SSH protocol suite of network connectivity tools that increasing numbers of people on the Internet are coming to rely on".* This might be true considering the fact that it is free. Another version of SSH is the commercial one which can be found at www.ssh.com.

3 Authentication Methods

They are 3 common methods of authentication provided by SSH:

a. Password Authentication

This type of authentication is enabled by configuring the /etc/ssh/ssh_config file. You just simply edit the part that contains "AllowedAuthentication" and add the word "password".

Ex:

```
AllowedAuthentication       password
```

The password used to login into server is the username password (kept in the /etc/shadow file). When user request to connect to server, it will prompt the user for a password. The password will be transmitted over the network for authentication in an encryption form. Thus, nobody can sniff the password.

b. *Public Key Authentication*

Another useful feature provided by SSH is the use of public key authentication. This means that no password is needed to connect to server. All you need is the public key and private key [rich]. How secure it is? Well, having used to use password authentication for years, it is a little awkward when accessing server without one. However, public key authentication is said to be much safer than using classical password authentication [marcel]. How does public key function? All we need is a pair of key which are Public Key and Private Key. Client may generate the keys and keep the private key is his personal station (pc, notebook or etc) and send a copy of his public key to be kept in the server. The only thing that allows user to connect to the server is the private key. So, user has to take careful action to protect his key and shall always bring it wherever he goes.

c. *Host-based Authentication*

This is a passwordless method of authentication [brianH]. Usually this method is combined with RSA-based host authentication in order to close the security vulnerabilities in regards to /etc/hosts.equiv protocol such as IP spoofing, DNS spoofing and routing spoofing [man]. This method involves two level of authentication.

The first level of authentication is based on trust. When a user wants to make a connection, the server that he wants to connect to will check the /etc/hosts.equiv file for the host name. If it is available, then the first level of authentication is granted [brianH]. However, the user still cannot access to the server as this method alone is not secure enough, which lead to the need of second level of authentication.

The next authentication level involves client's host key verification. The client needs to answer the "challenge" of proving itself as the legitimate client to the server. This process is best described with the following illustration:

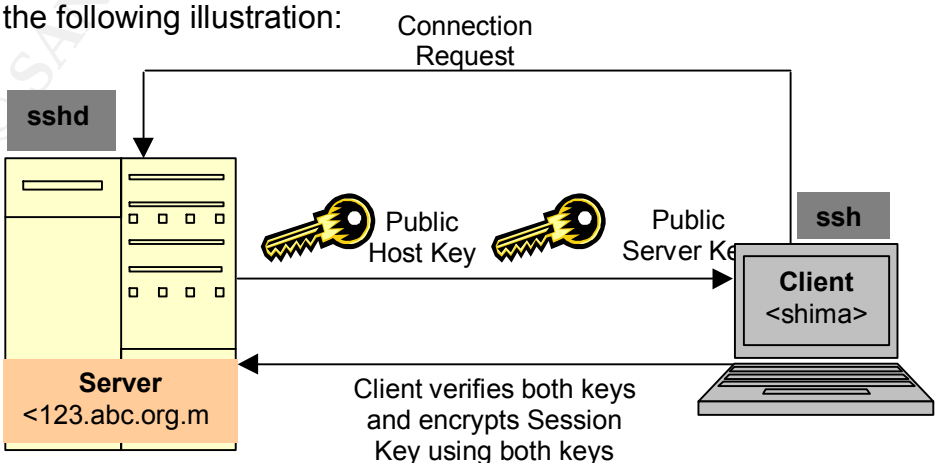


Figure 1: RSA Host-based Authentication

"ssh" is the client-side program(shima) that requests connection to server 123.abc.org.my. "sshd" is the SSH daemon running on the server. Once the server receives the connection request, it will send the Public Host Key and Public Server Key. Public Host Key is the key that is generated at install time and is kept in /etc/ssh_host_ key.pub or /etc/ssh/ssh_host_key.pub. Where else, the Public Server Key is the RSA key that is generated at run time by sshd. This key is generated every hour after the first use [scott].

The client will then verify the Public Host Key by looking in /etc/ssh_known_host and in the user home directory $HOME/ .ssh/known_hosts. If the key is there, the client will proceed of generating a random number which is known as Session Key. This key is encrypted by using both the Public Host Key and Public Server Key [scott]. Thus, the user is granted the access to the remote host. This is just a brief explanation. In fact, there are more processes involved behind the scene.

4 Some of SSH Capabilities

a. Remote Login :

Having the capability to provide an encrypted tunnel between client-server environment, SSH allows user to remotely login to the intended server securely even through the internet. Users even can execute commands from remote locations. The connection created through SSH is like an iron-based sound-proof tunnel between the two hosts. The eavesdropper is like someone who tries to listen the sound or to capture the image across the tunnel. Thus, it is quite impossible that people can break into this connection. Example:

```
% ssh -l shima 123.abc.org.my
```

The syntax is:

```
% ssh -l <username> <hostname>
```

We are also able to login remotely from Windows-based terminal to UNIX-based server. There are softwares that allow us to do so:

a. Putty
   This is a freely available utility in the internet at http://www.chiark.greenend.org.uk/~sgtatham/putty/ . According to the site, "*PuTTY is a free implementation of Telnet and SSH for Win32 platforms, along with an xterm terminal emulator".* It is a common practise to have UNIX-based servers while most of the clients are windows-based. Just like the environment I am in.

This utility assists us to make life easier by allowing us to connect to the server remotely even between two different OS based.

b. SecureCRT
This is the commercial version of utility that allows the connection to remote system using SSH. This can be found at:
http://www.vandyke.com/products/securecrt/index.html

b Transfer Files :

Another usage of SSH is to allow the transfer of files between two hosts securely. This can be done through 'scp' (Secure Copy Protocol).

Example:

% scp data.txt shima@123.abc.org.my:.

The syntax is:

% scp <filename> <username>@<hostname>:.

Referring to the example given, **data.txt** is the filename on localhost which to be copied to server **123.abc.org.my**. Where else, **shima** is the existing user on server **123.abc.org.my**. The command will copy the **data.txt** file from localhost to the target server at **123.abc.org.my**. The symbol "**:.**" indicates that the file will be dropped at user's home directory.

This command is used to transfer file between UNIX-based machines. Is it possible to do so between UNIX-based machine and Window-based machine? Well, it is possible using third party utility called WinSCP. According to http://winscp.sourceforge.net/eng/ , a website where we can freely download this software, *"WinSCP is an open source SFTP (SSH File Transfer Protocol) and SCP (Secure CoPy) client for Windows using SSH (Secure SHell). Its main function is safe copying of files between a local and a remote computer ".* This utility adopts the SSH protocol to provide an encrypted way of transferring files.

c Port Forwarding :

Other than that, SSH also has the capability of developing port forwarding. This is a unique criterion of SSH that provides a secure connection to known non-encrypted services such as POP3, NNTP, Telnet and etc. This method will create an encrypted connection from client host to the server using a specified port.

Where else, the client software will be configured to connect to the client's local port. Therefore, any data is transmitted in an encryption format rather than in a raw form where people can always sniff through the network [eskimo]. Example:

```
$ ssh 123.abc.org.my -R 110:localhost:1110
```

The syntax is:

```
$ ssh <target host> -R
remotePort:destinationHost:destinationPort
```

-R	To forward a given port on a remote host to a particular port on a local host.
remotePort	The port that the /usr/sbin/sshd will listen to and be forwarded to the local host. This can be a port number or a service name.
destinationHost	The destination host (name or IP address) from the perspective of the SSH client machine.
destinationPort	The port on the destination_host to which it should connect

[brian]

Besides configuring the port between UNIX platforms, this is also applicable between UNIX and windows-based machine. There are third party utilities available in the internet such as Putty (free) and commercial SecureCRT as mentioned before.

d Key Pair Generation :

OpenSSH comes with the key generation. There are three types of keys allowed which are RSA for SSH1, RSA for SSH2 and DSA for SSH2. The command used:

```
% ssh-keygen -t rsa 1      - RSA for SSH1

% ssh-keygen -t dsa        - DSA for SSH2

% ssh-keygen -t rsa        - RSA for SSH2
```

There are two files that will be created once the key is generated. One is the public key with an extension .pub and the other is the private key.

Both key will be kept in the user home directory $HOME/ .ssh/ directory. During the key generation process, the program will ask for a passphrase. This passphrase is not similar to a password. It can be longer and it also can be in the form of sentence. This passphrase is needed to login into remote host using public and private key. Its function is to protect the private key.

5 Is SSH secure enough?

SSH is an alternative of security utility. It was first initialized for the purpose of hiding data and password. However, sooner after the SSH-1 protocol was introduced, it had been proven that the protocol carried several vulnerabilities and limitation. Most of the security holes exist in SSH-1 are in regards to the use of RC4, an algorithm used for encryption [hel]. Thus, users are highly encouraged to not use RC4 algorithm for encryption. This site provides details of this: http://www.ssh.com/company/newsroom/article/212/. Vendors have done much to repair this in the later versions.

Known as having insecurity features, SSH-1 is not really being used in this decade. Users have moved to SSH-2 which has been released with more secure and advanced features than SSH-1 such as both RSA and DSA support, stricter encryption such as MD5 and etc. Still, there exists vulnerability in the use of SSH-2. As stated in the CERT-advisory, *"Multiple vendors' implementations of the secure shell (SSH) transport layer protocol contain vulnerabilities that could allow a remote attacker to execute arbitrary code with the privileges of the SSH process or cause a denial of service. The vulnerabilities affect SSH clients and servers, and they occur before user authentication takes place".* Details can be found at: http://www.cert.org/advisories/CA-2002-36.html

Well, a utility that was created for security purposes also contain vulnerabilities and has been used as medium of security breach. How do we want to rely on it? There are things we shall consider before using this utility:

- The need to apply it. Is it critical enough to use this utility?
- Access restriction. Only allow the minimum entity to gain authorization through this.
- Keep update of latest information. Ensure to patch the service as soon as possible.
- SSH execution in CHROOT environment
- Apply TCPWrapper for additional control

[will]

So, evaluation shall be made before applying this service and ensure proper security configuration has been made.

Setting up secure communication

Working in IT based company; our communication relies heavily on the email. This is more important when staff need to work outstation regularly. To be able to retrieve emails from outside of the organization is an alternative to ensure that communication continues smoothly. However, allowing access from the internet to our server is not a good security practise unless the transaction involves security protection. Thus, this is what SSH may provide us. It allows us to access our mails without other people can sniff the traffic.

The next section of this chapter, I shall describe the step-by-step action for setting email using port forwarding. The scenario is that all users are using windows-based platform where else the mail server is UNIX-based operating system. This example will use SecureCRT as the windows-based utility to setup connection to the UNIX-based server. This configuration manipulates some SSH capabilities in term of Remote login and Port forwarding.

1. Setting up server environment:

   The mail server shall be installed with SSH and the SSH shall be configured to apply public key authentication method. The server also shall allow access from the internet as described the diagram below.

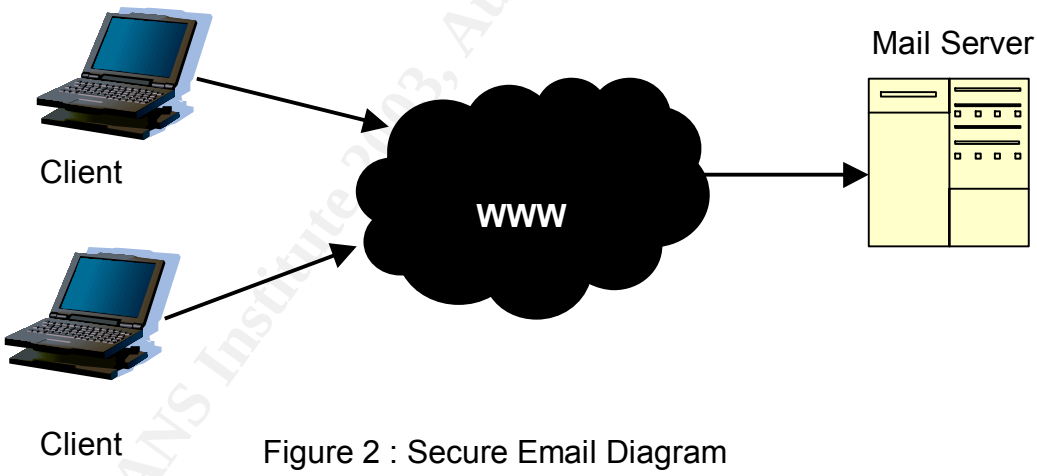
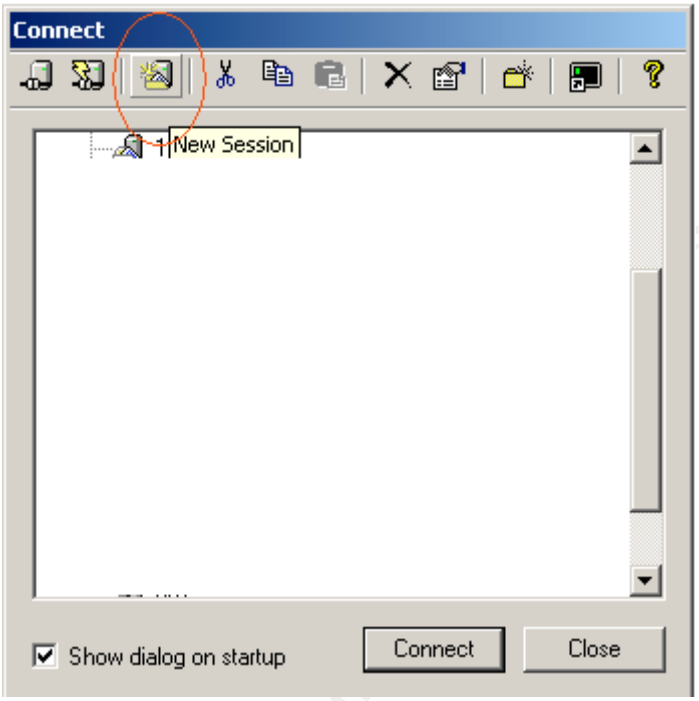


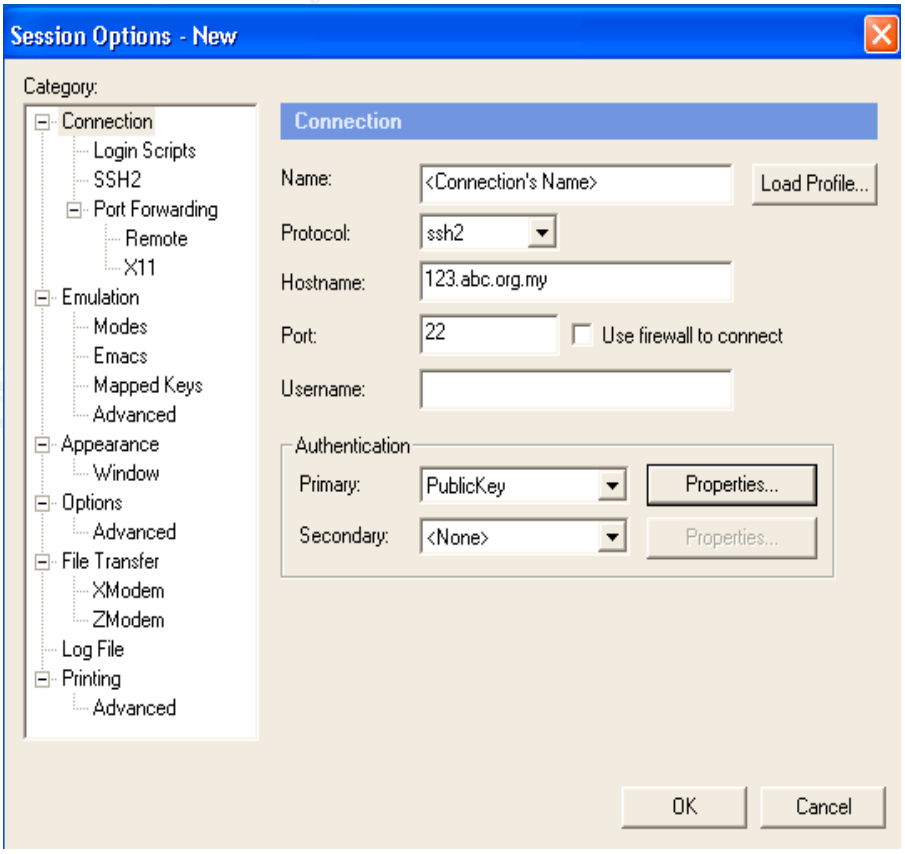
Figure 2 : Secure Email Diagram

2. Setting up the client environment (Configure SecureCRT):

i. Open SecureCRT and select new session which is the third button from the left as highlighted with red circle.

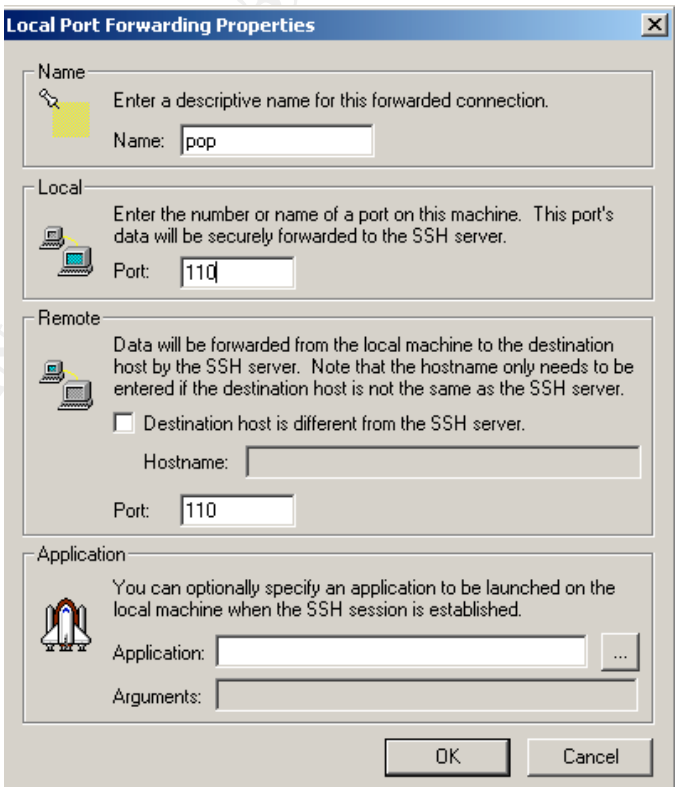


ii. Fill in the necessary information to build the connection.



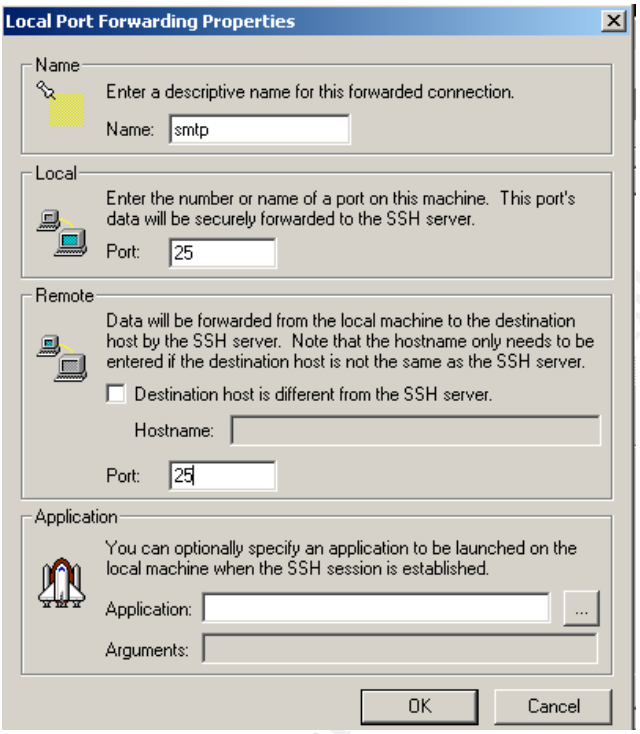
Name :	A name to represent the connection. It can be any name that user prefers to.
Protocol :	Which protocol that user wants to depend on. I would prefer to use SSH-2 due to the vulnerability image of SSH-1 as described before
Hostname :	The name of the server that user wants to connect to
Port :	The port to be used for the connection which is 22 for SSH
Username :	The username that is available in the server (if any) but in this case, I would prefer to not use one as I will use public key as access method
Authentication :	The authentication method that user wants to use to connect to the server. In this setting, I will use public key authentication as another security layer

iii. Select Port Forwarding option and configure the ssh client to create port forwarding



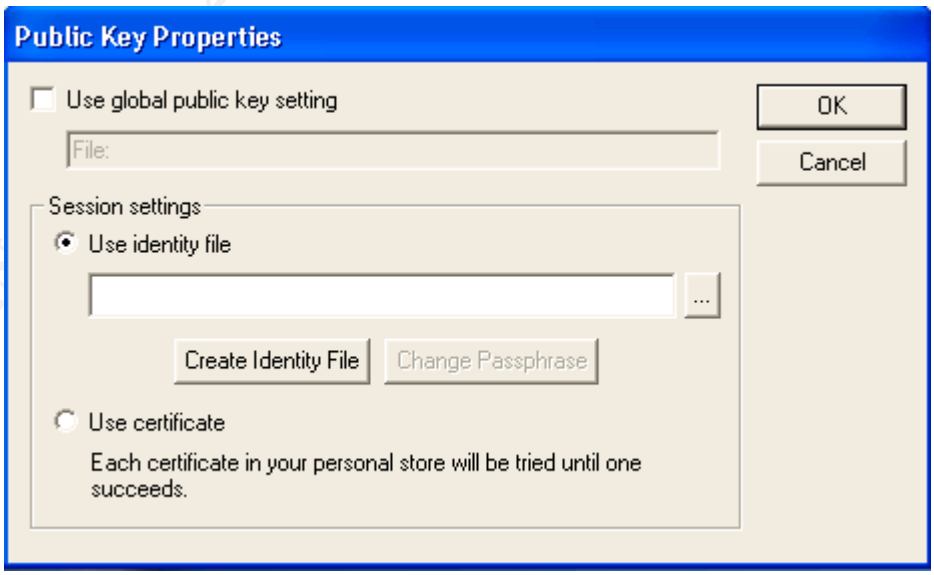
** 110 is port for pop which is used to retrieve emails from the mail server

For smtp services, add the field with the following information.

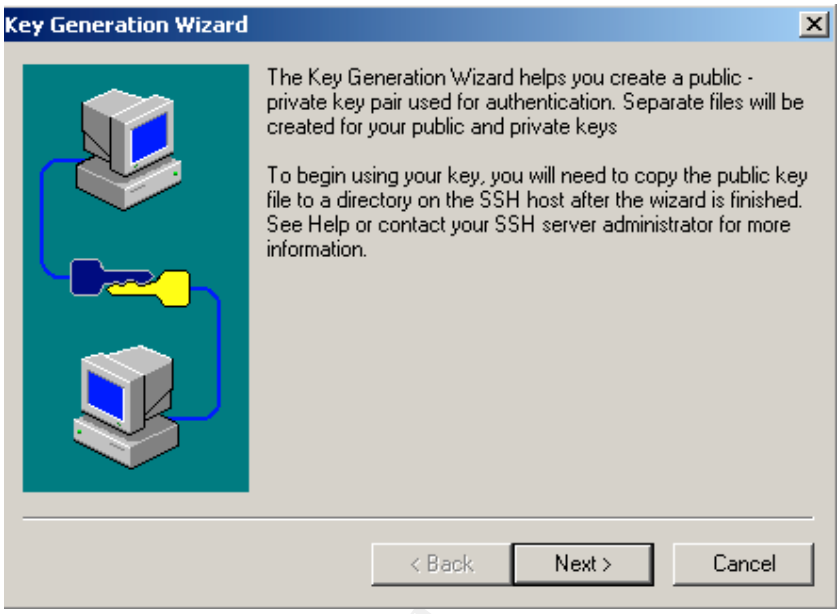


** 25 is port for smtp which is used to send emails

iv. Generate a key pair. Select the property button in the Authentication options.

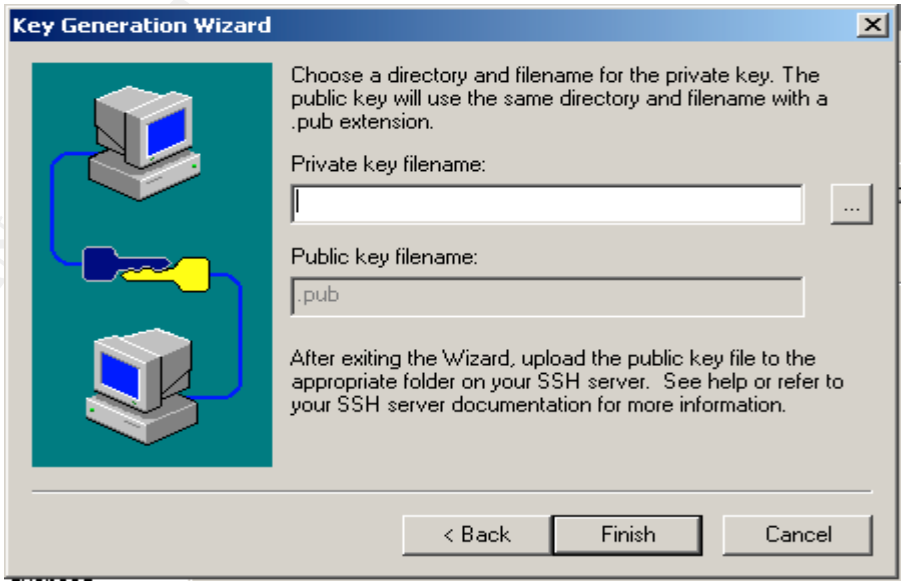


v. Click the Create Identity File button. You will be guided through the key generation wizard. Click on the Next button to continue.



vi. Next the wizard will prompt for the default storage location of generated key-pairs. Click next to select the default. Following this 2 files will be created:

- identity (private key)
- identity.pub (public key)



vii. A copy of public key file (identity.pub) shall be sent to the mail server and kept in the user home directory in /.ssh directory.

After setting up the port forwarding, configure the email client in the server names to localhost. Ensure that before sending and retrieving mails, the port forwarding has been established.

6 Conclusion

SSH provides another layer of security towards our system. However, it is not a total security solution. Even it provides multi functions in regards to security, there are also limitations such as inability to create encrypted tunnel for UDP-based communications port forwarding [scott].

Allowing access from the internet to our server needs concrete measures of its parameter protection. SSH alone without the support of other configuration will not promise anything. In fact, the concept 'Defend in Depth' shall be applied which involves the security of every aspect in our system. After everything is ready and successfully put on production, do not ever be satisfied and leave the machine as if everything will be just fine. It is a big mistake to not keep ourselves updated with the latest issues. We can be well run by subscribing to hot mailing list such as Bugtraq and always be aware of any vulnerability appears. Be remember to patch any service or utility as soon as possible before becoming the next victim.

In a nutshell, "*Do It Right Before Someone Does It Wrong For You*".

7 References

[brian] Hatch, Brian. “SSH Tunneling part 2 - Remote Forwarding”. 9 March 2003. URL: http://www.hackinglinuxexposed.com/articles/20030309.html. (18 August 2003).

[brianH] Hatch, Brian. “Secure Passwordless Logins with SSH Part 1”. 12 November 2002. URL : http://www.hackinglinuxexposed.com/articles/20021211.html. (18 August 2003).

[cert] CERT Coordination Centre. “CERT® Advisory CA-2002-36 Multiple Vulnerabilities in SSH Implementations “. 16 December 2002. URL : http://www.cert.org/advisories/CA-2002-36.html. (16 August 2003).

[eskimo] EskimoNorth, “SSH Port Forwarding”. URL: http://www.eskimo.com/support/ssh-forwarding.html. (20 August 2003).

[hel] SSH Communication Security. “Secure Shell version 1 vulnerabilities reported by CERT”. 7 November 2001. URL: http://www.ssh.com/company/newsroom/article/212/ . (23 August 2003).

[man] OpenBSD. “Manual Pages”. 25 September 1999. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1 . (18 August 2003).

[marcel] Gagné, Marcel. “Doing It All with OpenSSH, Part 2, 2003”. 24 June 2003. URL: http://www.linuxjournal.com/article.php?sid=6960&mode=&order=&thold= . (19 August 2003).

[open] OpenSSH. “OpenSSH 3.6.1 released April 1, 2003. Contains support for SSH1 and SSH2 protocols”. URL: http://www.openssh.com/. (19 August 2003).

[rich] Rich, Amy. “Secure Shell: Part 1”. http://www..com/bigadmin/features/articles/sec_shell_1.html. (20 August 2003)

[scott] Mann, Scott. L. Mitchell, Ellen. Krell, Mitchell. Linux Security System. New Jersey: Prentice Hall PTR, 2003.

[will] Pfeifer, William. “Security Implications of SSH”. 18 April 2003. http://www.sans.org/rr/paper.php?id=1180 . (21 August 2003)

Last Updated: May 23rd, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event