EmTec

http://www.emtec.com/

ELEMENTS OF AN SSH SESSION

Table of Contents:

Introduction

SSH, or secure shell, is a secure protocol of safely controlling remote servers through a shell. Using various encryption technologies, SSH provides a way to establish an encrypted connection between two computers, authenticating each side to the other, and passing commands and output back and forth. The connecting server is called the ssh client, the other is called ssh server.

Basic Terms

In order to secure the transmission of information, SSH employs a number of cryptographic techniques that need to be understood in order to understand SSH.

Symmetric Encryption

Symmetrical encryption is a form of encryption where a key can be used to encrypt messages to the other party, and also to decrypt the messages received from the other participant. What makes the encryption *symmetric* is the fact that the same key is used for encryption and decryption.

Symmetric encryption usually requires little computing power and is hence used to encrypt larger blocks of data. With SSH, it is used to encrypt the whole data stream.

Asymmetric Encryption

Asymmetrical encryption differs from symmetrical encryption in the fact that two different keys are used. One (any) of those two is used to encrypt the data and then the *other* is used to decrypt it. The benefit of this technique is that you can give the other party a key to encrypt messages to you, but anyone knowing that key will still not be able to decrypt the message again. Such a key is called the *public key*. The other key, which is not made public and which is used to decrypt the messages is called the *private key*.

This also works in the other direction. A message encrypted using the private key can only be decrypted using the public key. With SSH this fact can be used to prove identity. If a message is decryptable using the public key, it proves that whoever encrypted the message, is in possession of the private key.

Hashes

Another form of data manipulation that SSH takes advantage of is hashing. A hashing function is a method of creating some sort of fingerprint of a set of data. The hash is much shorter than the data itself, but is still different for each data set.

The data can not be created from the hash, but looking at the hash, you can prove that you have the same originating data. Thus comparing hashes is a way to prove that the originating data was the same, i.e. that it was not tampered with.

Initial Steps of an SSH Connection

The SSH protocol uses a client-server model to authenticate both sides and encrypt the data between them. The server listens on a given port for connections. It handles the details of the ssh connection, user-authentication and runs the remote shell.

A SSH session is established in two separate stages. The first is to agree upon and establish encryption to protect future communication. The second stage is to authenticate the user and discover whether access to the server should be granted.

The Negotiation Phase of the Connection

After a ssh client initiates a TCP connection to the server, the server sends a string indicating which protocol it supports. These days, by way the most common protocol version is SSH2. The server also sends its public host key, which the client uses to verify the authenticity of the host (in order to detect a possible man-in-the-middle attack). The ssh client can do this by computing a hash on the server key and comparing it to a previously stored hash.

They also exchange a list of features they support, e.g. key-exchange methods, encryption protocols, and if each group of features overlap by at least one common method, the communication can continue.

Now appears one of the most critical and most challenging parts of the communication. Both sides, which may have never met before, need to agree on an encryption key for symmetric encryption, knowing that the connection so far is not yet encrypted and an adversary could listen to the negotiation.

The negotiation is done through a process called the Diffie-Hellman algorithm. This method makes it possible for each party to combine some of their private data with public data from the other system and compute a secret byte sequence that is identical for both sides. The key will then be used to encrypt the

session. (Note: The public and private keys used in this method are completely separate from the SSH keys used to authenticate a client to the server).

The basic steps of the Diffie-Hellman method are:

1. Both parties agree on a large prime number, which will serve as a seed value.
2. Both parties agree on an encryption generator (typically AES), which will be used to manipulate the values in a predefined way.
3. Independently, each party comes up with another prime number which is kept secret from the other party. This number is used as the private key for this interaction (different than the private SSH key used for authentication).
4. The generated private key, the encryption generator, and the shared prime number are used to generate a public key that is derived from the private key, but which can be shared with the other party.
5. Both participants then exchange their generated public keys.
6. The receiving entity uses their own private key, the other party's public key, and the original shared prime number to compute a shared secret key. Although this is independently computed by each party, using opposite private and public keys, it will result in the *same* shared secret key.
7. The shared secret is then used to encrypt all communication that follows.

The generated secret is a symmetric key, meaning that the same key used to encrypt a message can be used to decrypt it on the other side, using a commonly supported encryption method (e.g. AES256-CTR). All further communication will then be encrypted and will thus be hidden from any party listening or intercepting the communication.

Authenticating the User

In the next step the user needs to prove to the ssh server who he is, upon which the ssh server decides the level of access (if any).

There are various authentication methods. The three most commonly used are password, keyboard-interactive challenge, and public private key authentication.

Password authentication

This form of authentication is the simplest one. The user specifies the username (on Unix/Linux systems this is usually system-wide username as specified in /etc/passwd) and corresponding password. Such authentication lets the user have only one set of credentials necessary for authentication.

Public key authentication

Public key authentication method is the only method that each software (both client and server) is required to implement. This method expects each client to have a key pair (key pair is a pair of keys, properly generated using one of asymmetric encryption algorithms, either RSA or DSA). The client first sends a public key to the server. If the server finds the key in the list of allowed keys, the client encrypts certain data packet using private key and sends the packet to the server together with the public key.

Keyboard challenge

Keyboard authentication is the advanced form of password authentication, aimed specifically at the human operator as a client. During keyboard authentication zero or more prompts (questions) are presented to the user. The user should give the answer to each question. The use of keyboard challenges is very flexible, they range from asking the user for his password, to answer the number which a code generator device, which is in his possession, displays at any given time.

Passing Control to the Shell

Once the parameters are negotiated and the user is authenticated, the ssh server launches a shell (or another command that the user requested when making the connection). These processes are executed on the server with the access rights the user has on the server's user list.

At this point the ssh connection becomes a transparent full duplex data channel, i.e. the software on the user side and the software on the server exchange data (keystrokes and command output) through the ssh connection, oblivious of the fact that the transmission passes over a network and each character is encrypted.

SSH Channels

With SSH version 2 the concept of channels was added to the ssh protocol. Channels are a way of splitting the established ssh communication connection into logical sub-connections.

While data is transmitted over a single network TCP connection, each packet gets a channel number and is treated by sender and receiver as if they were independent connections.

All terminal sessions, forwarded connections, etc., are channels. Either side may open a channel. Multiple channels are multiplexed into a single connection.

The ssh protocol allows the opening and closing of additional channels during the connection. While normally at least one channel is used for the user's shell access, other channels can be added for

different purposes, e.g. for file transfer, which means that a file can be transmitted over the same connection while the user is still active on the shell.

Channel Numbers and Types

Channels are identified by numbers at each end. The number referring to a channel may be different on each side. Requests to open a channel contain the sender's channel number. Any other channel-related messages contain the recipient's channel number for the channel.

Each channel has a type. Usually, you will use “session” channels, but there are also “x11” channels, “forwarded-tcpip” channels, and “direct-tcpip” channels. The “forwarded-tcpip” and “direct-tcpip” channels are managed internally via the port-forwarding interface of the client.