EmTec

http://www.emtec.com/

ZOC SSH-CLIENT FEATURES

© EmTec Innovative Software, Markus Schmidt

Table of Contents:

What is a SSH-Client

Secure Shell (SSH) is a cryptographic protocol that securely transports data over an unsecured network. As the name suggests, its main purpose is to establish a secure connection to a remote shell account.

The SSH client is the software, which the user runs on his local computer to connect to the remote server. Once connected, the ssh-client enables the user to enter commands and perform tasks through the shell of the remote computer.

ZOC SSH Specific Features

- Based on industry reference implementation OpenSSH
- Client supports latest encryptions like ED25519 SHA256, SHA2 or AES-256ctr
- SSH Public key, keyboard interactive or password authentication
- Dynamic port forwarding (client as a SOCKS proxy)
- X11 forwarding (lets you run X-Windows applications from the remote session)
- Static port forwarding (tunneling of connections to remote destinations)
- Proxy support (client connects to server through http/socks4/socks5)
- SSH Agent forwarding between client and server
- Client side SSH key generator
- SFTP file transfer
- SSH Keep-Alive

SSH Feature Details

Key Exchange

An especially difficult part of encrypted communication is the need to negotiate a shared secret (the key to use for encryption) over a public channel that could already be compromised.

The negotiation is performed through the so called Diffie-Hellman exchange or a variant thereof. ZOC supports all official diffie-hellman group exchanges, as well as the more modern ecdsa-sha2 and curve25519-sha256 protocols.

Authentication

Authenticating describes the process, where the user presents proof of who he is and the server deciding, if the user should be allowed access. The SSH protocol describes various methods that can be used for authentication.

Of those, ZOC supports password authentication, pukey exchange and keyboard-interactive challenge. Public-key exchange comes in various flavors. The ZOC ssh-client understands RSA, DSA, ECDSA and ED25519 keys. Hardware (smart card) based key authentication is also possible.

Encryption

Over time, the SSH protocol has seen a plethora of methods to be used to encrypt the communication (using shared secret was negotiated during the KEX phase as a cryptograhpic key). Some ciphers were phased out over time, especially after Edward Snowden revealed how powerful possible listeners like the NSA are, and new ones were introduced. ZOC supports the whole list, starting with aes256-ctr and going down to older ciphers like aes256-cbc or arcfour (these older ones may still be necessary to connect to older servers which have not been updated in a while).

Port-Forwarding

An important part of the secure shell protocol is a feature called *port-forwarding*. This feature allows the user to create a connection from the client computer to the server network, which can be used by other programs and where all the connection data is encrypted. This feature is sometimes called *tunneling*.

Programs and protocols which do not use data encrpytion (e.g. ftp or rsh) can connect to the tunnel's port on the client computer and the ssh client will transmit their data through the encrypted ssh connection to the final destination (and vice versa).

For example, a user can set up a port-forwarding (also called ssh-tunnel) on the client software, listening on the client port 5514 and forwarding traffic to the address of an older device on the remote network that only supports the unencrypted rsh protocol.

The user can then use a non-encrypting rsh client, connect it to localhost port 5514 and thus will get connected via the secure shell client to the rsh server on the remote network. A normal rsh client will not encrypt its data, but the ssh client will encrypt all traffic before sending it through the ssh tunnel to the host on the other side (and vice versa), essentially creating an encrypted rsh connection.

Dynamic Port-Forwarding

The standard port-forwarding feature requires the client to set up the tunnel source port and destination before making the connection. This means that there is limited flexibility and that for each possible destination, a separate ssh tunnel needs to be set up.

This problem is addressed by secure shell's dynamic port forwarding. With dynamic port forwarding, the client sets up a listening port (as with normal port fowarding), but when a software connects to the port, it can tell the client which host and port it wants to connect to. This is done in the same way that client software can request connections from SOCKS proxies.

The ssh client will then forward the connection request to the secure shell server which makes the connection to the destination host. This way, the ssh client could let an unencrypted ftp software access ftp servers on the remote network through an encrypted data channel.

SSH Connection via Proxy

In some environments, end user computers are not allowed to access the outside internet directly. In those cases, connection and data exchange is made by way of a ssh proxy which handles the actual connection to the outside network (internet). There are various type of proxies, which mainly differ in how the ssh client requests a connection to the outside world. Most common types are SOCKS-4, SOCKS-5 and HTTP. ZOC supports connections through those types, as well as connections made thorugh ssh-jumpservers.

X11 Forwarding

X11 is a communication protocol which allows a remote computer to run programs with a graphical user interface on a remote computer (normally, the remote computer can only show text in a terminal client). SSH supports a way to tunnel this type of communication between ssh client and server, thus enabling the user to run an X11 command like xeyes on the remote shell and get the window for that displayed on the local computer.

SSH Agent Forwarding

When a user authenticates an SSH session using a public/private key pair, ZOC supports the SSH agent forwarding technique to provide the key for authentication in secondary ssh sessions (ssh connections to a third server, made from typing a ssh command in the remote shell in the initial connection). If all the servers allow authetication through this specific ssh key pair, it is not necessary to provide the passphrase again for secondary ssh connections.