

Re: Explanation of SSH

Source: <http://www.derkeiler.com/Newsgroups/comp.security.ssh/2004-06/0211.html>

From: Bill Unruh (unruh_at_string.physics.ubc.ca)

Date: 06/26/04

Date: Sat, 26 Jun 2004 17:45:35 +0000 (UTC)

oscar@jbexNOSPAM.com writes:

]Having looked at the July issue of Computer Shopper (UK) magazine,
]page 269, I am still unclear on how SSH works exactly. I have googled
]and have not found a clear explanation.

]Please either point me to a web page; or enable my understanding of
]the following; or even both.

]Steps in SSH connection setup;

]1. Client issues SSH command and names server

]2. "Shopper" says "server sends back its public host and server keys

]to client". You what? Surely there is only one public key it sends

]back, and "host and server keys" means just one public key?

Usually,

]3. Client adds server key(s?) to its knownhosts file if it isn't there

]already.

And tells the user it is doing so. Also if it already has a different key
in the knownhosts file for that server, it complains "loudly".

]4. Shopper says, "Client generates 256 bit random number which it

]encrypts using server and client public keys." What is the 256 bit

]random number? Is this the client's private key? How do you get client

]private and public keys, are they randomly generated once for each SSH

]client installation, or are new client public/private keys generated

]each time SSH is run?

No, the user is the client. The user generated their own public and private keys. The client has the public key of the server. The client and server negotiate on which symmetric encryption algorithm they will use. The client generates a random number which will be the key for that symmetric algorithm. The client encrypts that random number with the public key of the server. The server decrypts it with its private key. The two now share a secret password. From then all all encryption is done using that symmetric algorithm (IDEA, Blowfish, DES, 3DEs,

comp.security.ssh: Re: Explanation of SSH

]5."Both the client and server then use this number to generate private session keys".... you what? Why do you need private session keys if the server only has one private key always, and the client already has a public/private key pair from step (4)?

For the symmetric algorithm. Public key encryption and especially decryption are horribly slow. All the traffic is encrypted using the symmetric algorithm. The only use of the public key encryption is to exchange that symmetric algorithm key (and for use as authentication)

]Other questions;

]I have to login to my SSH provider using loginname/password. Do those relate at all to the encryption element of the SSH transaction, ie. is the username/password somehow a passphrase unlocking the private key from the encrypted private key which the server stores; or is username/password simply login authentication with no encryption relevance?

No. the password username are to authenticate you to the server, just as you do when you log in while sitting in front of it.

]Is it the case that the encrypted SSH link is established first, and then username/password are transferred over the encrypted link? So if an attacker is watching the link, is the username apparent to them, or ciphered?

It is ciphered. All traffic apart from the initial contacts are encrypted.

]Is client public key generated afresh each time SSH runs, or is it generated just the once and then the same client public/private key used for all time?

Just once. By the user using ssh-keygen or by the system on installation of ssh using ssh-keygen for the system (storing it in /etc/ssh)

The symmetric algorithm key is generated each time the two systems. contact each other.

]Sorry for all the questions, I'm curious how all this works and would like some clarity.