Diagramme de séquence de l'établissement d'une session SSH

<uml> @startuml group Etablissement d'une session 3-Way Handshake ClientSSH → Serveur : demande session avec paquet SYN (Synchronize) Serveur → ClientSSH : approuve connexion avec paquet SYN-ACK (acknowledgement) ClientSSH → Serveur : confirme avec paquet ACK end group Etablissement de la session ClientSSH → Serveur : annonce la version de SSH utilisée Serveur → ClientSSH : annonce la version de SSH utilisée Serveur → ClientSSH : envoie sa clé publique ClientSSH → ClientSSH : génèrent paires de clés\nprivées - publique\ntemporaires Serveur → Serveur : génèrent paires de clés\nprivées - publique\ntemporaires ClientSSH ↔ Serveur : algorithme d'échange \nde clés Diffie-Hellman\npour déterminer\nclé partagée\n(symétrique) note left : clé de session symétique note right : clé de session symétique ClientSSH → ClientSSH : Vérification de l'identité du serveur\nAcceptation de l'identité du serveur end group Etablissement d'un canal sécurisé end group Authentification par mot de passe Serveur → ClientSSH : demande saisie login ClientSSH → Serveur : envoi login Serveur → ClientSSH : demande saisie mot de passe ClientSSH → Serveur : envoi mot de passe Serveur → Serveur : vérification login et mot de passe Serveur → ClientSSH : affichage du prompt end

group Authentification par clé SSH ClientSSH → Serveur : fait enregistrer sa clé publique\ndans authorized*keys Serveur → ClientSSH : demande saisie login ClientSSH → Serveur : envoi login Serveur → Serveur : envoie un défi (challenge) ClientSSH → ClientSSH : signe le défi|navec sa clé privée ClientSSH → Serveur : envoi réponse au challenge Serveur → Serveur : vérifie la signature avec la clé publique\nenregistrée dans authorized*keys Serveur → ClientSSH : affichage du prompt end

group Communication chifrée entre le client et le serveur durant toute la session

end group Deconnexion ClientSSH → Serveur : déconnexion avec paquet RST,ACK (Reset) Serveur → ClientSSH : approuve déconnexion avec paquet ACK end @enduml </uml>

Compléments

- Les connexions TCP
- l'algorithme **chacha20_poly1305** (RFC 7539)

L'agorithme de chiffrement symétrique **ChaCha20** est utilisé à la place de **AES 256**.

- **ChaCha20** : algorithme de chiffrement symétrique plus rapide qu'AES sur un matériel générique (mise en œuvre purement en logiciel)
- **Poly1305** : c'est un **MAC** (message authentication code) qui permet d'assurer l'**intégrité des données** en vérifiant qu'elles n'ont subi aucune modification après une transmission. C'est une fonctionnalité semblable à une fonction de hachage.

La combinaison de ces deux algortihmes permet de faire du chiffrement intègre.

Ressources

- https://fr.wikipedia.org/wiki/%C3%89change_de_cl%C3%A9s_Diffie-Hellman
- https://www.adikts.io/mieux-comprendre-ssh-et-lauthentification-par-cles/
- https://www.bortzmeyer.org/7539.html
- https://fr.wikipedia.org/wiki/Code_d%27authentification_de_message
- https://serverfault.com/questions/586638/understand-wireshark-capture-for-ssh-key-exchange
- https://serveur.ipgirl.com/comprendre-la-capture-wireshark-pour-lchange-de-keys-ssh.html