ICA0002: IT Infrastructure Services

SSH Basics

Roman Kuchin
Juri Hudolejev
2021

SSH: Secure Shell

Remote shell operated securely over insecure network

Replaced **`telnet`**, **`rsh`**, **`rlogin`** and **`rexec`**

De-facto standard tool to operate remote machines

Default connection protocol in Ansible

More info: https://www.ssh.com/academy/ssh

SSH: Secure Shell

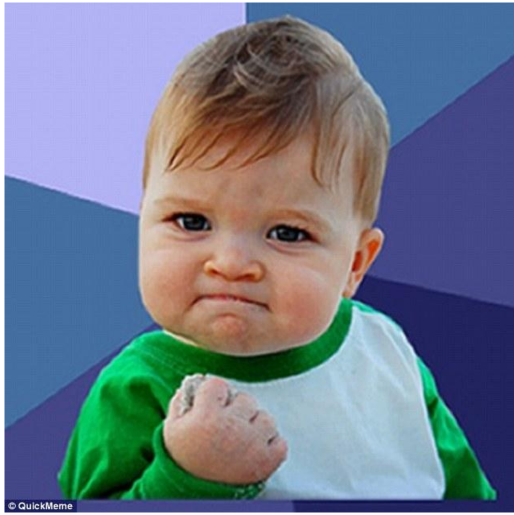
Remote shell operated **securely over insecure network**

Replaced `telnet`, `rsh`, `rlogin` and `rexec`

De-facto standard tool to operate remote machines

Default connection protocol in Ansible

More info: https://www.ssh.com/academy/ssh



Encryption

Symmetric encryption: DES, AES etc.

- Same key for encryption and decryption -- shared secret

Asymmetric (public key) encryption: DSA, RSA etc.

- Public key (openly distributed) + private key (kept secret)
- Message encrypted with one key from the pair
  can only be decrypted with the other key from the same pair

Encryption

Symmetric encryption: DES, AES etc.

Requires secure channel to exchange the shared secret :(

Asymmetric (public key) encryption: DSA, RSA etc.

Is unacceptably inefficient on large data :(

Let's use a **public key** to create a secure channel for a shared key exchange, and then use that **shared key** to encrypt the data...

That would work!

SSH session initialization

Client sends the connection request to the server

Then client and server both:

- agree algorithms for key exchange, symmetric and public key encryption
- generate shared session key using Diffie-Hellman method (SSH v2)

Then server performs the client authentication

If all good, connection is established

SSH client authentication

Authentication options:

- User password based
- User key based: RSA, DSA, ECDSA etc.
- Host key based
- Interactive -- for one time passwords
- GSSAPI -- for external authentication services such as Kerberos

SSH client authentication

Authentication options:

- User password based ← avoid at all costs
- User key based: RSA, DSA, ECDSA etc. ← we only use this on this course
- Host key based
- Interactive -- for one time passwords
- GSSAPI -- for external authentication services such as Kerberos

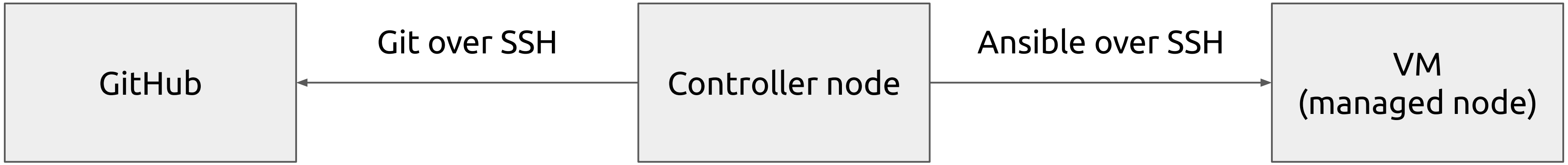
SSH public key based client authentication

In very simple terms:

- Client encrypts (signs) certain data with its **private** key
- **Public** key and signature are sent to SSH server
- SSH server checks if the public key is acceptable (authorized)
- SSH server verifies signature
- If all checks passed -- client is authenticated

More detailed info: https://tools.ietf.org/html/rfc4252

SSH in this course



Your SSH keys in this course

Your public key:

- **`~/.ssh/id_rsa.pub`** file on Controller node (connect from)
- **`~/.ssh/authorized_keys`** file on your VMs (connect to)
- In your GitHub account: https://github.com/<username>.keys (lab 1)

Your private key:

- **`~/.ssh/id_rsa`** file on Controller node -- should never leave your machine!

Public key may also be extracted from the private key file, but not vice versa!

Important!

If your private key is lost or compromised,

1. Delete the corresponding public key from your GitHub account immediately!
2. Generate a new key pair (see lab 1)
3. Contact the teachers to reset the keys on your VMs

Questions?