Secure system access

HPC Café, 9. Juni 2020
Secure authentication

- **Authentication factors**
  - **Knowledge factors**: e.g. Password, Passphrase, PIN
  - **Ownership factors**: e.g. ID card, cell phone, hardware token, software token
  - **Inherence factors**: Fingerprint, signature, retinal pattern, face, location

- **Multi-factor** authentication is mandatory

- Compromise between **security** and **convenience**

- In HPC environments **secure shell (SSH)** is the de-facto standard for authentication of access to remote systems
Common security guidelines

- Never share a password or key for different hosts/systems
- Do not store passwords clear text anywhere in any form
- Use strong passwords (15-20 characters, can contain all character classes, can include words and sentences)

If you want to stay sane you need a password manager:
- Manages large number of passwords
- Can generate long secure passwords for you
- Prevents to enter passwords all over again
- SSO password to password manager is single point of failure!
Password manager options

- Built-in solutions:
  - Google Password manager
  - Apple Keychain
  - Firefox Password manager

- Commercial offerings:
  - 1Password (3.15 Euro per month)
  - LastPass (free option, 2.67 Euro per month)
  - Bitwarden (Open Source with commercial support)

- Open Source
  - KeePass (portable, cross platform, with GUI)
  - Pass and clones (CLI based, the hackers choice, based on GPG for encryption, can use git for synchronization)

It is difficult to give a recommendation!
Who do you trust?
What do I use? gopass

- **Gopass** - The slightly more awesome standard unix password manager
  - Improved *reimplementation* of pass in **Golang**
  - **GPG** for *encryption* and **git** for *synchronization*
  - Support for *password sharing* (teams)
  - **Clean** and accessible **CLI interface**
  - Very *good support* and large active community
  - **Cross platform** (Linux, Apple Mac, BSD, Windows)

Websites: [https://www.gopass.pw/](https://www.gopass.pw/)  
[https://github.com/gopasspw/gopass](https://github.com/gopasspw/gopass)
SSH overview

**SSH** is a cryptographic **network protocol**

- Designed in 1995 by Tatu Ylönen (researcher at Helsinki University of Technology, Finland)
- SSH is standardized by an **Internet Engineering Task Force (IETF)** working group
- OpenSSH (an OpenBSD project) is the most common Open Source implementation
SSH: How does it work?

- “Secure” means
  - User is authenticated to the system
  - System is authenticated to the user
  - All transmitted data is encrypted

- Technology
  - Asymmetric encryption algorithm („Public Key“) for authentication and determination of a Session Key

  - Symmetric encryption of data transfer using Session Key
SSH: How does it work?

1. **Connects to server**
2. **Sends server's public key**
3. **Negotiates and opens a secure channel**

**User authentication**
- **Symmetric encryption with session key**
- **Asymmetric encryption with public key algorithm**

**Client**
- Connects to server
- Sends server's public key
- Negotiates and opens secure channel
- **Symmetric encryption with session key**
- **Asymmetric encryption with public key algorithm**

**Server**
- **Symmetric encryption with session key**
- **Asymmetric encryption with public key algorithm**
Public key authentication

- Use a cryptographic **key-pair** as **password**
- Keys are **not used** for **encryption**
- Was initially intended for **automation**
- Provides **two-factor** authentication if used with **passphrase**
- Implements **SSO solution** if using **ssh-agent**

Optional: ssh-agent holds decrypted key in memory

Encrypted private key

<table>
<thead>
<tr>
<th></th>
<th>Authorized public key</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>decrypted</strong></td>
<td><strong>passphrase</strong></td>
</tr>
<tr>
<td><strong>private key</strong></td>
<td><strong>in memory</strong></td>
</tr>
</tbody>
</table>

Decrypted private key
Public key authentication

- Use a cryptographic **key-pair** as password
- Keys are **not used** for encryption
- Was initially intended for **automation**
- Provides **two-factor** authentication if used with **passphrase**
- Implements **SSO solution** if using **ssh-agent**
Agent-forwarding

- Default a ssh-agent does not provide **single-sign-on (SSO)** on remote host

User perspective:
- **Agent-forwarding** enables **SSO** also on **remote hosts**
- **Private keys** do **not** need to be **deployed** on remote hosts

Admin perspective:
- **Authentication** can be hijacked and is **forwarded** to a potentially **untrusted** remote environment

Our advice: Do not use agent-forwarding!
Recommended options for key generation:

$ ssh-keygen -t rsa -b 4096 [-f outfile]

Command to transfer public keys to server:

$ ssh-copy-id -i ~/.ssh/mykey user@host

Configure per host settings in ~/.ssh/config file.

See next slide for details.

Keychain persistent frontend for ssh-agent (use on Client)

More Information: https://www.funtoo.org/Keychain
ssh config file

- **Location:** `~/.ssh/config`
- **Allows to** create shortcuts to hosts and **adjust ssh settings** on a per host base (**Caveat:** Settings are implicit!)
- **Documentation:** `$ man ssh_config`
- **Example entry:**

```plaintext
Host rrze
  ForwardAgent no
  ForwardX11 no
  HostName cshpc.rrze.fau.de
  User unrz999
  IdentityFile /home/john/.ssh/id_rsa_rrze
```
Security hints for ssh clients

- Keep the **private key files** secret!
- The following files should be read-only:
  - `authorized_keys`, `known hosts file` and `config file`
- Use a **4096bit RSA key** protected by a **passphrase**
- Use a **strong passphrase** (at least 15 characters long)
- Use a **separate key** for every client
- Disable **agentForwarding** and **X11Forwarding** in config
- Do not leave **open external logins** in running `Tmux/screen`
- Keep your ssh client installation **up to date**

And I mean secret!
Consequences

Private keys should be only placed on single-user systems, best using an encrypted harddisk

A multi-user terminal server is an untrusted host

- NO private keys on untrusted hosts
- NO ssh-agent on untrusted hosts
- NO agent forwarding to untrusted hosts
Use-case for RRZE HPC systems

Home directory is NFS share

- If **one system** is hacked **all systems** are hacked
- Use a **RRZE only keypair** for internal logins

Recommendations:

- Use **separate keypair** on every client (Laptop, Desktop)
- Create single **keypair** for internal RRZE use
- This key **may** be also used to access **external systems**
Welcome to ProxyJump

- **To access hosts** behind a terminal server (bastion host) since OpenSSH 7.3 the **ProxyJump** functionality was added.
- The connection is **tunneled** through the bastion host but the connection to final target host is made by the initial client.
- `ssh` will give warning if a **man in the middle attack** occurs.

Add this to `~/.ssh/config`:

```
Host emmy
  HostName emmy
  User unrz999
  IdentityFile ~/.ssh/id_rsa_rrze
  ProxyJump cshpc
```

```
Host cshpc
  Hostname cshpc.rrze.uni-erlangen.de
  ForwardAgent no
  User unrz999
```

All Keys have to be on originating host.
Use case: Automated file synchronization

- **Automated rsync over ssh** is a common use case
- If using **passphrase less keys** you should **restrict access** in authorized keys
- There is a **dedicated script** for this: `rrsync`

In `.ssh/authorized_keys` add:

```
command="rsync /rsync/base/path",no-agent-forwarding,no-port-forwarding,no-pty,no-user-rc,no-X11-forwarding ssh-rsa AAAA[...]
```

Path below server base path

Config entry:
```
Host rrze-sync
Hostname cshpc.rrze.uni-erlangen.de
IdentityFile ~/.ssh/id_rsa_sync
User unrz999
```

Calling rsync on client:
```
$ rsync -e ssh -av path/ rrze-sync:subdir/
```
New developments in ssh security

- As we learned to **increase security** you have to **increase factors** involved

- Add **hardware token**: e.g. FIDO keys, **YubiKey**, can be subkey of GPG key

- Use **One Time Passwords (OTP)** as second factor to extend passphrase, e.g. generated by YubiKey

- Send **additional pin** per text message to **mobile device**
DEMO

- Use *gopass* password manager
- Create ssh key-pair
- Transfer public key to remote host
- Setup secure file sync
- Create ProxyJump entry in `.ssh/config`