Contents

ssh

Keys Generation

Basic SSH-Keys Generation

NIST's standard recommendations for ssh keys encryption algorithms

Encryption Algorithm	Key length	Key generation command
ECDSA, EdDSA, DH, MQV	f=224-255 (and above)	`ssh-keygen -t ed25519`
RSA	k=2048 (and above)	`ssh-keygen -t rsa -b 4096`

```
# generate keys using ed25519
ssh-keygen -t ed25519

# generate keys using rsa
ssh-keygen -t rsa -b 4096

# copying over keys to remote system
cat $HOME/.ssh/id_ed25519.pub | ssh USERNAME@remote.system.ip "cat >> $HOME/.ssh/authorized_keys"
# alternatively one would use the ssh-copy-id command
ssh-copy-id -i $HOME/.ssh/id_ed25519.pub USERNAME@remote.system.ip
```

A typical output of the process of keys' generation would look like this,

```
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/USERNAME/.ssh/id_ed25519): $HOME/.ssh/system1_id_ed25519
Enter passphrase (empty for no passphrase):  ****************************************
Enter same passphrase again:  ****************************************
Your identification has been saved in system1_id_ed25519.
Your public key has been saved in system1_id_ed25519.pub.
The key fingerprint is:
SHA256:VrKTrH2EX+52KoKVbtIcPnzRlf21jo0vZU/Wf4IysjI local.USERNAME@local.machine
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|                 |
|        . .    o |
|       . *    o o|
|        S.....  =|
|       +++.o.  +=|
|      .O..o...Boo|
|      E @.=.o+oo+|
|       =.* =oooo.|
+----[SHA256]-----+
```

The next step would be to transfer the public part of recently generated key to the remote system,

```
$ ssh-copy-id -i $HOME/.ssh/system1_id_ed25519.pub USERNAME@remote.system.ip
```

Keys Management

```
# transfer/copy keys to remote system
ssh-copy-id  -i $HOME/.ssh/id_ed25519.pub  USERNAME@remote.system.ip

# use of an specific key to access clusterX
ssh -i $HOME/.ssh/USER_clusterX_ed25519 USERNAME@clusterX.IP.address

# use of an specific key to access clusterY
ssh -i $HOME/.ssh/USER_clusterY_ed25519 USERNAME@clusterY.IP.address
```

```
# use of an specific key for remote access to clusterZ
ssh -i $HOME/.ssh/USER_clusterZ_ed25519 USERNAME@clusterZ.IP.address
```

Adding Comments to your keys One useful fearure offered by the key generation comamand is the capability to associate comments to key, so that they can be used to remind us what a given key is being used fo . Comments can be added to the key when created using the `-C` flag. For instance,

```
# key generation with comments and specified location
ssh-keygen -t ed25519 -C "USER@laptop cluster-X" -f $HOME/.ssh/USER_clusterX_ed25519
```

in this case, both the comment and the name for the keys files is being specified by the respective `-C` and `-f` flags. If one would like to modify the comment of an existent key, the `-c` (lower-case "c") flag can be used instead.

Using `ssh-agent` to remember your keys Keys are quite powerful, they can substantially improve security and efficiency at the moment of connecting to work in remote systems. One really useful feature to help with productivity is requesting an ssh-agent program to recall our keys/passphrases combinations, in this way when a key is used to connect to a given system the *ssh-agent* will remember the passphrase entered for a given period of time avoiding to repeatedly prompt for it. The way to trigger this feature is to use `ssh-add key-file` . It is also possible to specify a timeout period (lifetime) for how long to remember the passphrase, using the `-t` flag.

Some additional ssh-agent commands

ssh-agent command	description
`ssh-add -l`	Lists fingerprints of all identities currently represented by the agent
`ssh-add -D`	Delete all identities from the agent
`ssh-add -t life`	Sets the maximum time the agent will keep the given key

Customizing SSH keys names It is possible to specify the name of the file where to store the keys when generating them. By default ssh will search for predefined file names, such as `id_ed25519` or `id_rsa` . But if we are using a different name, then we should indicate ssh which file we are using as keys. For doing so, we will use the `-i` flag followed by the location (which is also standardized under `$HOME/.ssh`) and the actual filename. E.g.

```
# ssh using specific key file
ssh -i $HOME/.ssh/USER_clusterX_ed25519 USERNAME@clusterX.IP.address
```

Configuration Details The preevious process can be simplified even a bit more, by adding some of theese details to the configuration file used by ssh. Such a configuration file resides in the `$HOME/.ssh/` directory and is named `config` . An example of an entry in this file is shown below,

Single Host

```
HOST clusterX
     HostName clusterX.IP.address
     User USERNAME
     IdentityFile ~/.ssh/USER_clusterX_ed25519
```

Multiple Hosts One could even envision the possibility of including mutliple hosts by adding entries in the `~/.ssh/config` file.

```
HOST clusterX
     HostName clusterX.IP.address
     User USERNAMEonclusterX
     IdentityFile ~/.ssh/USER_clusterX_ed25519

HOST clusterY
     HostName clusterY.IP.address
     User USERNAMEonclusterY
```

```
     IdentityFile ~/.ssh/USER_clusterY_ed25519

HOST clusterZ
     HostName clusterZ.IP.address
     User USERNAMEonclusterZ
     IdentityFile ~/.ssh/USER_clusterZ_ed25519
```

in this way, the user would just use the commands `ssh clusterX` or `ssh clusterY` for connecting to any of the corresponding remote systems.

Troubleshooting Keys Configurations

A couple of options to consider when finding troubles with the keys setup in a remote system can be generically considered.

Firstly, if for what ever reason the ssh-copy-id command is not found or available in the local system where the keys were generated, an alternative way to transfer the public key to the remote system could be achieved by using the following command,

```
cat $HOME/.ssh/id_ed25519.pub | ssh USERNAME@remote.system.ip "cat >> $HOME/.ssh/authorized_keys"
```

this assumes that the public key named `id_ed25519.pub` is located at `$HOME/.ssh/` directory and it will be placed in the `remote.system.ip` space of a user named `USERNAME` .

Permission Attributes

Another issue that may arise when transferring the private keys, is related to an improper setup of the file permissions. The `$HOME/.ssh` directory must only be accessible by the owner, and the various key files must not be writable (or in some cases, readable) by anyone else. This is how the `$HOME/.ssh` directory should look like,

```
# look at ~/.ssh permissions
ls -ld $HOME/.ssh

drwx------ 2 USERNAME GROUPNAME 7 Aug  9 15:43 /home/USERNAME/.ssh
```

To fix improper set permissions, the following command may be used:

```
# fix permissions in ~/.ssh
chmod -R go= $HOME/.ssh/
```

Debugging/Verbose Mode

Additionally, if we find problems when trying to ssh into a system which we either know the authentication procedure (either keys, password, or MFA) is not behaving or working as expected, we can use the following options when using the ssh command to obtain more verbose detail of the connections,

```
# -v activates the "verbose mode": resulting in printing debugging messages
# helpful in diagnosing connection, authentication, and configuration problems
# Multiple -v options increase the verbosity, the maximum is 3.

ssh -v USERNAME@remote.system.ip
ssh -vv USERNAME@remote.system.ip
ssh -vvv USERNAME@remote.system.ip
```

Multiplexing: `ControlMaster`

Multiplexing is the ability to send more than one signal over a single line or connection. In OpenSSH, multiplexing can re-use an existing outgoing TCP connection for multiple concurrent SSH sessions to a remote SSH server, avoiding the overhead of creating a new TCP connection and **reauthenticating each time**.

ssh has an option called `ControlMaster` that enables the sharing of multiple sessions over one single network connection. This means that you can connect to the remote system once, enter your credentials, and have all other subsequent ssh sessions utilizing the initial connection without need for re-authentication. It is possible to establish this `ControlMaster` setup manually each time on the command line, but instead it's easiest

to put it in the ssh client configuration file so that it applies every time that a connection is launched to the corresponding system.

```
HOST clusterX
     HostName clusterX.IP.address
     User USERNAMEonclusterX
     IdentityFile ~/.ssh/USER_clusterX_ed25519
     ControlMaster auto
     ControlPath ~/.ssh/controlmasters/%r@%h:%p
```

When ssh is instructed to use `ControlMaster` , it will look for the special file (a socket) in the `~/.ssh/controlmasters/` directory that is maintaining a connection to the cluster. If it already exists and is open, it'll use it to create a connection without re-authenticating; if it doesn't exist, it'll authenticate and create the file for subsequent use.

Note that all subsequent connections are dependent on the initial connection — if you exit or kill the initial connection all other ones die, too. This can obviously be annoying if it happens accidentally. It's easily avoided by setting up a master connection in the background:

```
ssh -CX -o ServerAliveInterval=30 -fN remoteServer
```

in this case the `-fN` flag puts the process in background and sit idle, after authenticating; `-C` is for using compression and `X` for X-forwarding, `ServerAliveInterval` is used to specify a time interval to keep the sessions open when inactive.

Tunneling

Tunneling is a method for transporting data across a network using protocols that are not supported by that network. Tunneling works by *encapsulating* packets, i.e. wrapping packets inside of other packets. There are many different ways of tunneling, for instance, VPN, ssh tunneling, etc.

In tunneling, or port forwarding, a local port is connected to a port on a remote host or vice versa. So connections to the port on one machine are in effect connections to a port on the other machine.

Typically the options `-f` (go to background), `-N` (do not execute a remote program) and `-T` (disable pseudo-tty allocation) can be useful for connections that are used only for creation of tunnels.

In regular port forwarding, connections to a local port are forwarded to a port on a remote machine. This is a way of **securing an insecure protocol or of making a remote service appear as local**.

```
ssh -L localPortNbr:localhost:remotePortNbr -l username remote.server.ip
```

In that way connections on the local machine made to the forwarded port will in effect be connecting to the remote machine.

An application of this is the usual utilization of ssh-tunnels to establish VNC connections to remote locations.

Graphics Forwarding

Also known as X11 forwarding or just X-forwarding, is the ability to forward graphical output from the remote system to the local host connected via ssh. This sometimes can be handy and useful but in most of the cases could also be slow.

`ssh` offers two options to enable X-forwarding: `-X` and `-Y` , and this alternatives are related to security concerns.

From `ssh` 's documentation (see https://man.openbsd.org/ssh.1#X):

```
    -X      Enables X11 forwarding.  This can also be specified on a per-host basis in a configuration

            X11 forwarding should be enabled with caution.  Users with the ability to bypass file perm
            remote host (for the user's X authorization database) can access the local X11 display thr
            connection.  An attacker may then be able to perform activities such as keystroke monitori

            For this reason, X11 forwarding is subjected to X11 SECURITY extension restrictions by def
            to the ssh -Y option and the ForwardX11Trusted directive in ssh_config(5) for more informa

     -x     Disables X11 forwarding.
```

```
-Y      Enables trusted X11 forwarding.  Trusted X11 forwardings are not subjected to the X11 SECU
```

References

- ssh summary
- https://en.wikibooks.org/wiki/OpenSSH

Last Modified: Oct. 12, 2022 -- v 0.1

Multi-Factor Authentication (MFA)

Multi-Factor Authentication (MFA) is a technique employed to strength the process of authenticating against a service. In a typical situation you would use a set of credentials, such as a username and a passwords or keys to validate your identity. By adding an additional way of confirming the identity of an user we can more securely of authenticating that such an user holds the identity is attempting to validate.

There are a few different ways to implement this, for instance, financial institutions --such as banks-- would use a code sent to your cell-phone or older technologies included the utilization of pre-distributed codes in the form cards or usb devices generating specific sequences of codes assigned to the corresponding user.

Another way to implement this strategy is by the use of the so-called *One Time Password* (OTP) following the same premises as described above.

The way this technique works is by sharing a common ”root” or source of information, and then based on a predefine prescription generate sequences of codes using this initial ’state’; pretty similar to how *pseudo-random number generation* (PRNG) works as well. Not so surprisingly then, at the core of many MFA implementations is a PRNG algorithm, some of them will use an ”initial” value (aka seed), some will use the actual time as such, making them a *time-based* token.

Among some of the most used MFA tools are:

- Google Authenticator -- https://github.com/google/google-authenticator An open-source, time-based implementation for MFA.
- DUO -- https://duo.com A propietary implementation, offering different ways to authentication.
- YubiKeys -- https://www.yubico.com Hardware propietary tokens.
- PrivacyIdea -- https://www.privacyidea.org/

Last Modified: Oct. 12, 2022 -- v 0.1

Virtual Network Computing (VNC)

- is a graphical desktop-sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer
- more efficient and selective way to display graphics from a remote connection
- it requires the implementation of a *tunnel* or the remote host to allow connections to given ports.

Set up a VNC connection

1. ssh into the remote server, e.g.
   `ssh userName@remote.server.ip`
2. launch a VNC server in the remote host, e.g. `vncserver`
3. Check with `vncserver -list` , and take note of the port number, usually denoted as `:XXXX`
4. set a password using `vncpasswd` -- **do NOT leave a password-less VNC setup**!!!
5. ssh into remote server creating a *tunnel to the local machine*, i.e.
   `ssh -fN -L5904:localhost:XXXX userNAME@remote.server.ip`
6. in your local machine launch a VNC client, eg. remina, tigervnc, etc. In MacOS, you can type in the terminal, `open vnc://localhost:5904`

Other Remote-Access Protocols

VNC is one protocol used to remotely access computational resources but there are many other ones. For instance, communication applications may use their own ones to give users access to control remote resources, or specialized applications may entitle users to control remote computers using proprietary protocols. Another example of such is the *Remote Desktop Protocol* (RDP) mostly used in Windows machines.

Independently of the specific protocol used, one should note that these are a liability form cyber-security considerations. Extreme caution must be followed when using these type of tools, and awareness that they may weaken the cybersecurity perimeter of an organization or device by enabling weak or blind spots when accessing the remote resource.

References:

- https://datatracker.ietf.org/doc/html/rfc6143

Last Modified: Jun. 13, 2023 -- v 1.0

Virtual Private Network (VPN)

VPN is a secure, encrypted connection over a publicly shared network. Tunneling is the process by which VPN packets reach their intended destination, which is typically a private network.

Last Modified: Oct. 12, 2022 -- v 0.1

Although not strictly linked to *remote computing* many different aspects in our day-to-day operations involve remote transferring of information, as well as, remote access, trust, etc. Among one of the most relevant tools and commodities employed nowadays is electronic-mail, e-mail.

Firewalls

In security, a firewall is a *system* that controls the incoming and outgoing information into a system. They offer a way to guard, inspect and filter the information flow into the system. Firewalls can be implemented in hardware or software. Hardware firewalls are mostly used at the level of data centers or Internet service providers, they suppose to act as a general and broad filter of generic attacks, such as denial of service or brute force attacks. Software firewalls can be also configured by centralized authorities, as well as by users in their own local systems. Examples of this, are the usual firewall settings at level of the OS -- it is strongly recommended to adjust these settings consciously to minimize the inflow or outflow of unnecessary information.

Email

In the same way that it is important to validate the integrity and validity of our connections to remote systems, it would also be for other types of communications, such as electronic messages, or email. In particular, for *email* a tool called **Pretty Good Privacy** (PGP) can be used to ensure the confidentiality %of the messages exchanged, as well as validate the identity of the sender. PGP is an encryption method that provides cryptographic privacy and authentication for data communication. The basic idea is to encrypt the communication (similar to how using *ssh-keys* would do it) and sign it so that the message received at the other end of the communication channel can be decrypted, validated and authenticated.

Refs.

- https://en.wikipedia.org/wiki/Pretty_Good_Privacy

Last Modified: Jun 15, 2023 -- v 1.0

Cyber-security Checklist

- In your *local system*:
  - ☐ use an anti-virus
  - ☐ keep software up-to-date with the latest patches, including the ones for the Operating System (OS)
  - ☐ be mindful of emails, malicious attachments and links:
    - ☐ do not enter sensitive data in unknown websites,
    - ☐ verify for https connections and SSL certificates
  - ☐ do not plug any type of device of unkown origin or source, e.g. USB-devices, memory sticks, memory cards, etc.
  - ☐ use a password manager, do not store passwords in plain-text and use a different password for each service
  - ☐ encrypt sensitive data
- When connecting to *remote systems*:
  - ☐ use ssh keys, with passphrases
  - ☐ use MFA
  - ☐ use VPN
  - ☐ check the information provided by the remote system (usually at the moment of logging in), about when have you connected and from which locations
  - ☐ consider using "private browsing" and set restrictions on *cookies* policies in your web browser and when visiting websites with tracking and third party cookies

---

Cyber-security *checklist*: main elements to take into consideration to enhance the cyber-security in your local and remote work spaces.

Last Modified: Oct. 10, 2022 -- v 0.1

Action	Description
keep software up-to-date	keep your devices updated with all software updates, including OS and applicat
use ssh to connect to remote systems	de-facto tool to connect to remote systems using asymmetric encryption
use ssh-keys	more efficient and convenient way to authenticate
use ssh-keys + MFA	enhanced way to authenticate
verify fingerprint of remote system	checks validity and authenticity of remote system by comparing system's fingerp
connect through VPN	improves network protection and privacy by creating an encrypted channel over
firewall	"middle-ware" (hardware and/or software) to intercept and filter potential attac
use an antivirus	local protection against wide spectrum of malware
use a passwords manager	specialized tool to more securely (i.e. using encryption) store passwords and ger
encrypted signed email (e.g. using PGP)	enhance authenticity and validity of email communications

Summary of some best practices for end users to enhance cybersecurity in remote computing.

`ssh` Summary

A summary of the most relevant aspects is available here.

Last Modified: Oct. 10, 2022 -- v 0.1

SSH Summary

Connections, forwarding and tunneling

Connections, forwarding and tunneling	
connection to remote system	`ssh username@remote.system.IP`
	`ssh username@remote.system.IP -p PORTnbr`
with graphics-forwarding	`ssh -X username@remote.system.IP`
	`ssh -Y username@remote.system.IP`
tunneling	`ssh -R remPort:remote_host:locPort username@remote.system.IP`
	`ssh -L locPort:remote_host:remPort username@remote.system.IP`
	`ssh -fN -[R _or_ L] port:remote_host:port username@remote.system.IP`
remote execution	`ssh username@remote.system.IP "remote_cmd_to_exec"`

Keys

Generation

```
ssh-keygen -t ed25519

ssh-keygen -t rsa -b 4096

# key generation with comments and specified location
ssh-keygen -t ed25519 -C "USER@laptop cluster-X" -f $HOME/.ssh/USER_clusterX_ed25519

# ssh using specific key file
ssh -i $HOME/.ssh/USER_clusterX_ed25519 USERNAME@clusterX.IP.address
```

Transfer

```
ssh-copy-id  -i $HOME/.ssh/id_ed25519.pub  USERNAME@remote.system.ip

# copying over keys to remote system
cat $HOME/.ssh/id_ed25519.pub | ssh USERNAME@remote.system.ip "cat >> $HOME/.ssh/authorized_keys"
```

Agent to recall key

```
ssh-add key-file
ssh-add key-file -t life
```

Troubleshooting

Debugging (verbose mode)

```
# -v activates the "verbose mode": resulting in printing debugging messages
# helpful in diagnosing connection, authentication, and configuration problems
# Multiple -v options increase the verbosity, the maximum is 3.

ssh -v  USERNAME@remote.system.ip
ssh -vv USERNAME@remote.system.ip
ssh -vvv USERNAME@remote.system.ip
```

More about ssh

Last Modified: Oct. 12, 2022 -- v 0.1

Further References and Resources

- ”High-Performance Computing (HPC) Security: Architecture, Threat Analysis, and Security Posture”, https://csrc.nist.gov/publications/detail/sp/800-223/draft Dated: Feb. 2023
- ”Secure Code Game” by GitHub, https://github.com/skills/secure-code-game Dated: Mar. 30, 2023
- CitizenLab, https://citizenlab.ca
- ”Are Your Passwords in the Green?”, Hive Systems, https://www.hivesystems.io/blog/are-your-passwords-in-the-green Dated: Apr. 18, 2023

CyberSecurity Glossary

Cyber-Security/Defense Matrix

Under the premise that security attacks, vulnerabilities and eventually breaches will occur; cybersecturity experts develop strategies and protocols to follow in the occurrence of these events. One way to organize and categorize these is by employing the so-called *cyber defense matrix* where among others, elememts such as, risk, layer or stratum at which the incident occur or was detected, action to be taken, etc. are arranged.

Social Engineering

One of the weakest element in any cybersecurity infrastructure is the so-called *human factor*, i.e. the employment of individuals to actually infiltrate and geopardize systems. This can indeed take different forms, e.g. via email manipulation or impersonation --e.g. spam, phishing attempts--, or even convincing individuals of pluging physical devices such as USB sticks of unkown procedence in their computers, etc. The most direct form of these attack techniques even target specific individuals by learning and correspondingly acting based on someone's typical preferences or behaviors. The generic term employed to describe this type of attacks is *social engineering*, as they are designed and calibrated to the specific idiosyncrasy of the victim's target. These attacks are still responsible for large number of cybersecurity breaches, and has been reported that even robust implementations, such as the utilization of MFA combied with other authentication mechansims, could be overcome by convincing and carefully crafted manipulation. One of the best advices agaisnt this type of manipulative attacks is to always remain vigilant and suspicious, in particular do NOT trust any sources of unknown procedence or origin.

- "Social engineering in cybersecurity: The evolution of a concept", J.M.Hatfield; Computers & Security 73 (2018) https://doi.org/10.1016/j.cose.2017.10.008

PQC - Post Quantun Cryptography

With the recent advances of quantum computers, one of the potential applications and immediate concers is the capability for "powerful" enough quantum devices to *break* the encryption alogorithm used nowadays. At the beggining of 2017, the NIST launched a request for propossals to develop standars to develop Post-Quantum Cryptographic (PQC) algorithms. Further information about this ongoing effort can be found in the following links:

- "NIST / Post-Quantum Cryptography" - Accessed: June 14, 2023 https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
- "Post-quantum cryptography", Bernstein, Lange; Nature 549 (2017) https://www.nature.com/articles/nature23461
- "Cisco / Post Quantum Security Brief" - Accessed: June 14, 2023 https://www.cisco.com/c/en/us/products/collateral/c networking/solution-overview-c22-743948.html

---

Terms and Acronyms

- `AES` Advanced Encryption Standard (symmetric)
- `ARC` Advanced Research Computing
- `CVE` Common Vulnerability and Exposure
- `DES` Data Encryption Standard -- **unsecure** deprecated, replaced by AES (https://www.rfc-editor.org/rfc/rfc4772.txt)
- `DH` Diffie-Hellman algorithm
- `DDoS` Distributed DoS attack
- `DoS` Denial of Service attack
- `ECC` Elliptic Curve Cryptography
- `ECDH` Elliptic Curve Diffie-Hellman
- `ECDSA` Elliptic Curve Digital Signature Algorithm
- `EdDSA` Edwards-curve Digital Signature Algorithm
- `ECMQV` Elliptic Curve MQV
- `HPC` High Performance Computing, aka supercomputing
- `HTTP` | `HTTPS` HyperText Transfer Protocol, secure hypertext transfer protocol
- `MFA` | `2FA` Multi-Factor (or Two-Factor) Authentication, see MFA
- `MQV` Menezes-Qu-Vanstone, authentication protocol for key agreement based on DH
- `OTP` One Time Password
- `PGP` Pretty Good Privacy
- `PKC` Public Key Cryptography, aka asymmetric cryptography
- `PQC` Post-Quantum Cryptography
- `RSA` Rivest-Shamir-Adelman algorithm (asymmetric)
- `SHA` Secure Hash Algorithm
- `ssh` Secure Shell, see ssh
- `SSL` Secure Sockets Layer
- `TLS` Transport Layer Security protocol, security protocols applied to communications within a computer network
- `VPN` Virtual Private Network, see VPN
- `VNC` Virtual Network Computing, see VNC

Last Modified: Jun. 14, 2023 -- v 1.0