# Security



# Infrastructure Security

## Basic Security Principals

- Access to all project instances, running ONAP, Kubernetes, openstack etc, shall go through secured jumphost/admin servers
  - This includes but not limited to ssh, http(s) and so on. For http(s) access, port forwarding/tunneling through jumphost can be employed.
- The instances shall not have direct ssh access from anywhere but jumphost/admin servers
- The instances shall be put in right/existing security groups depending on the purpose and that have only the required ports open and never into **de fault** security group
- If new rules are required to be added to the existing security groups or new security groups need to be created, this shall only be done in a controlled manner by submitting a ticket to Nordix JIRA, Infra project
- Everyone shall have and use their own accounts on admin/jumphost with only key based authentication - no shared keys
- If access to development instances are required, everyone shall use their own ssh key to access them
- Access to the OpenStack API shall be given on a need basis and a record of this access shall be kept
- Access to the CityControl panel shall be given on a need basis and a record of this access shall be kept
- People with CityControlPanel access shall not share their credentials with others
- People with OpenStack API access shall not share their credentials/openrc files with others
- All access requests including but not limited to SSH shall be done on Nordix JIRA, Infra project
- Infrastructure shall automatically be scanned for potential issues such as checking security groups, operating system vulnerabilities, user access and so on
- <addme>

## What If

**The resources you use are compromised**

- If the resources such as instances you created on Nordix Public Cloud Tenants are compromised, it is possible that the stacks you installed on them such as Kubernetes could be compromised as well
- **Remove all access to the instances in your project on the tenant you are working on to block both incoming and outgoing traffic as soon as possible.** You can do this by removing associated security groups from the instances.
- The reason to cut the access for all the instances is that some of the attacks are contagious and it is possible that all the instances are infected so an analysis needs to be done in your project to see the scale of the problem.
- Contact Infra Core Team privately and discuss the issue with them. **Never have this type of conversations on public maillists or IRC channels.**
- **Do not remove or turn off the instances** without consulting the Infra Core Team first as it is possible that the attackers left traces of what they have done on the instances and logs could contain valuable information to help us with further investigation and action.
- Once the analysis is done, root cause is identified, and required actions are taken, the instances can be removed.
- On top of the damages security incidents cause both for us and others, you should also expect at least 2 days to recover and go back to day to day work so everyone must be security conscious to begin with to prevent this happening to you or your teammates.

**Your SSH keys are stolen**

- **Remove all access to the instances in your project on the tenant you are working on to block both incoming and outgoing traffic as soon as possible.** You can do this by removing associated security groups from the instances.
- Once the access is removed, you can place the instance to a new security group with an IP you trust in order to login to the instance to take backup of your data. It is probably faster to remove compromised instance and start fresh on a new one as investigation of what the attacker did and cleaning it up would take more time than starting on a new instance. Also you might never be sure about if you cleaned it up enough.
- Contact Infra Core Team privately if the stolen ssh keys are used for accessing instances created on Nordix Tenants.
- Generate a new SSH keypair for your personal use, lock its filesystem access properly, and never share.
- **Do not use stolen SSH keypair again.**

**You shared your password accidentially**

- Immediately reset your password on the system you use it and contact Infra Core Team privately if the compromised account is used for Nordix Services
- Never use the same password on multiple systems

**<addme>**

# Code Security Checks

Relevant information regarding scanning code that's about to be committed to git repositories and catching potential security issues like

- accidental inclusion of private keys and credentials
- including/downloading files and artifacts from unknown and potentially dangerous locations
- virus scanning
- shell history
- <addme>

More info TBD

# Artifact Signing and Vulnerability Scanning

Relevant information regarding signing the artifacts that are made available via Nordix (either by building or caching) and scanning them for security issues and vulnerabilities

- ISO, RPM, Deb, container images and so on
- <addme>

More info TBD

# Generic Checklist for Host/Server Hardening

**HOST:-**

1. Document the host information

Each time you work on a new Linux hardening job, you need to create a new document that has all the checklist items listed in this post, and you need to check off every item you applied on the system. Furthermore, on the top of the document, you need to include the Linux host information:

- Machine name
- IP address
- Mac address
- Name of the person who is doing the hardening (most likely you)
- Date
- Asset Number (If you're working for a company, then you need to include the asset number that your company uses for tagging hosts.)

2. BIOS protection

You need to protect the BIOS of the host with a password so the end-user won't be able to change and override the security settings in the BIOS; it' s important to keep this area protected from any changes. Each computer manufacturer has a different set of keys to enter the BIOS mode, then it's a matter of finding the configuration where you set the administrative password.

Next, you need to disable the booting from external media devices (USB/CD/DVD). If you omit to change this setting, anyone can use a USB stick that contains a bootable OS and can access your OS data.

The latest servers' motherboards have an internal web server where you can access them remotely. Make sure to change the default password of the admin page or disable it if it's possible.

3. Hard disk encryption (confidentiality)

Most of the Linux distributions will allow you to encrypt your disks before installation. Disk encryption is important in case of theft because the person who stole your computer won't be able to read your data if they connect the hard disk to their machine.

In the image below, choose the third option from the list: Guided-use entire disk and set up encrypted LVM (LVM stands for logical volume manager.)

If your Linux distribution doesn't support encryption, you can go with a software like *TrueCrypt.*

4. Disk protection (availability)

Backups have so many advantages in case of a damaged system, bugs in the OS update. For important servers, the backup needs to be transferred offsite in case of a disaster. Backup needs to be managed as well. For example, how long will you keep the old backups? When do you need to backup your system (every day, every week …)?

Critical systems should be separated into different partitions for:

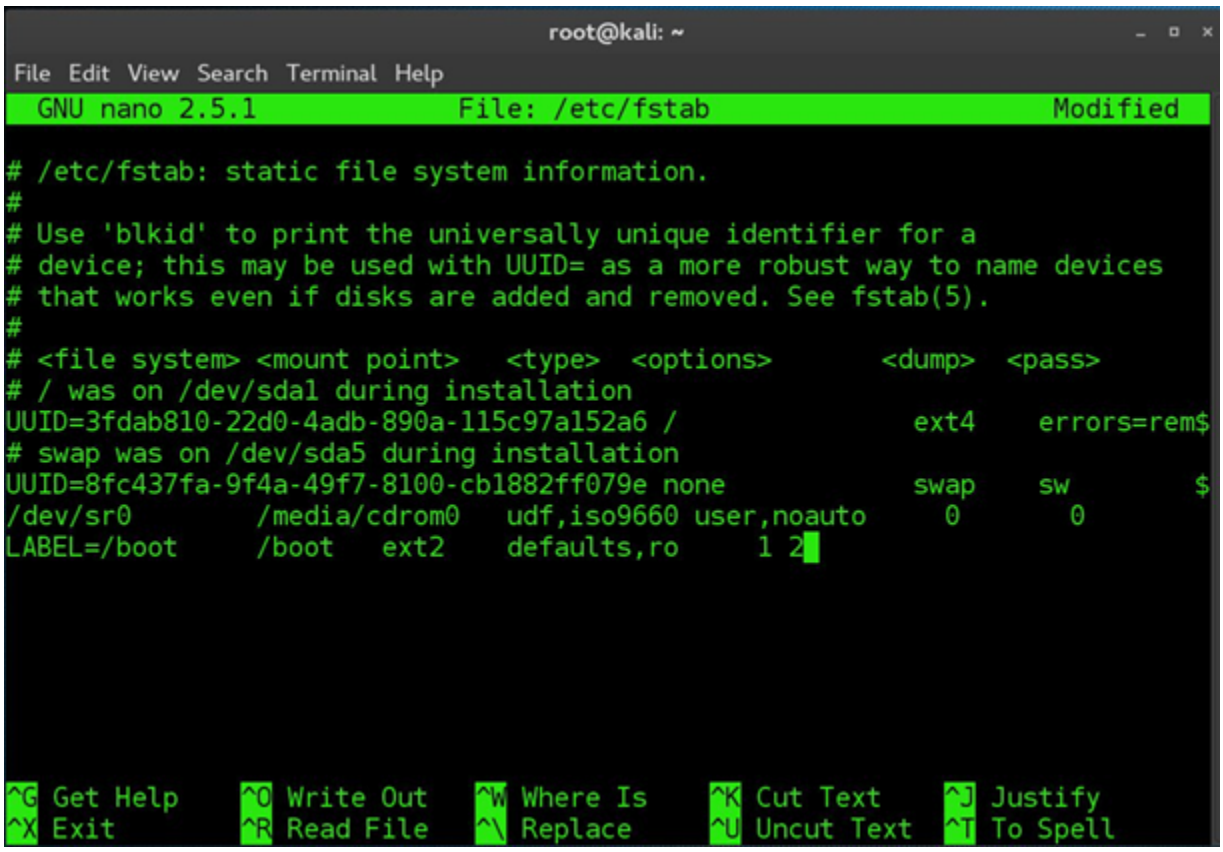
- */*
- */boot*
- */usr*
- */home*
- */tmp*
- */var*
- */opt*

Portioning disks gives you the opportunity of performance and security in case of a system error. In the picture below, you can see the option of how to separate partitions in Kali Linux during the installation.

5. Lock the boot directory

The boot directory contains important files related to the Linux kernel, so you need to make sure that this directory is locked down to read-only permissions by following the next simple steps. First, open the “fstab” file.

Then, add the last line highlighted at the bottom.




When you finish editing the file, you need to set the owner by executing the following command:

*#chown root:root /etc/fstab*

Next, set few permissions for securing the boot settings:from: eoin

- Set the owner and group of /etc/grub.conf to the root user:
  *#chown root:root /etc/grub.conf*
- Set permission on the /etc/grub.conf file to read and write for root only:
  *#chmod og-rwx /etc/grub.conf*
- Require authentication for single-user mode:
  *#sed -i "/SINGLE/s/sushell/sulogin/" /etc/sysconfig/init*
  *#sed -i "/PROMPT/s/yes/no/" /etc/sysconfig/init*

6. Disable USB usage

Depending on how critical your system is, sometimes it's necessary to disable the USB sticks usage on the Linux host. There are multiple ways to deny the usage of USB storage; here's a popular one:

- Open the "blacklist.conf" file using your favorite text editor:
  #nano /etc/modprobe.d/blacklist.conf
- When the file opens, then add the following line at the end of the file (save and close):
  blacklist usb_storage
- After this, open the rc.local file:
  #nano /etc/rc.local
- Finally, add the following two lines:
  modprobe -r usb_storage
  exit 0

**SERVER**

1. System update

The first thing to do after the first boot is to update the system; this should be an easy step. Generally, you open your terminal window and execute the appropriate commands.

2. Check the installed packages

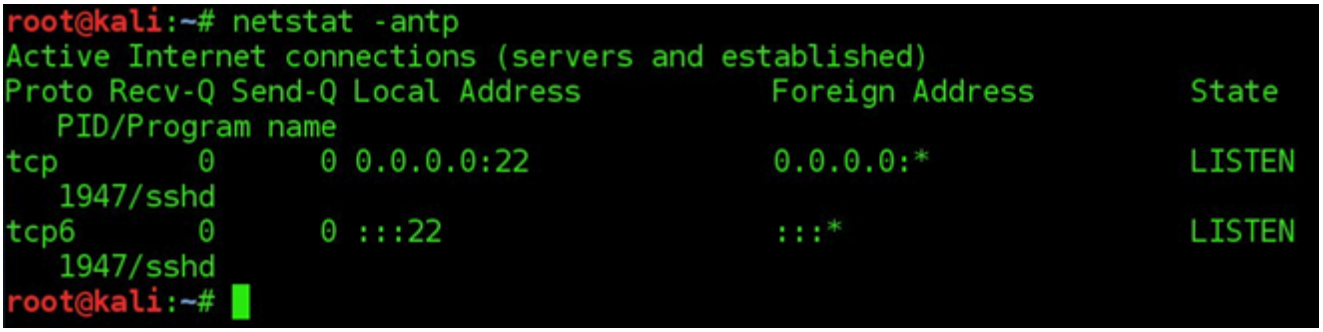
List all packages installed on your Linux OS and remove the unnecessary ones. You need to be very strict if the host you're trying to harden is a server because servers need the least number of applications and services installed on them.

Remember that disabling unnecessary services will reduce the attack surface, so it is important to remove the following legacy services if you found them installed on the Linux server:

- Telnet server
- RSH server
- NIS server
- TFTP server
- TALK server

3. Check for open ports

Identifying open connections to the internet is a critical mission. Use natstat for this process.




4. Secure SSH

Yes, indeed SSH is secure, but you need to harden this service as well. First of all, if you can disable SSH, that's a problem solved. However, if you want to use it, then you have to change the default configuration of SSH. To do it, browse to /etc/ssh and open the "sshd_config" file using your favorite text editor.


- Change the default port number 22 to something else e.g. 99.
- Make sure that root cannot login remotely through SSH:
  *PermitRootLogin no*
- Allow some specific users:
  *AllowUsers [username]*

The list can go on and on, but these should be enough to start with. For example, some companies add banners to deter attackers and discourage them from continuing further.

Here are some additional options that you need to make sure exist in the “sshd_config” file:

- Protocol2
- IgnoreRhosts to yes
- HostbasedAuthentication no
- PermitEmptyPasswords no
- X11Forwarding no
- MaxAuthTries 5
- Ciphers aes128-ctr,aes192-ctr,aes256-ctr
- ClientAliveInterval 900
- ClientAliveCountMax 0
- UsePAM yes

Finally, set the permissions on the sshd_config file so that only root users can change its contents:

```
#chown root:root /etc/ssh/sshd_config

#chmod 600 /etc/ssh/sshd_config
```

5. Enable SELinux

Security Enhanced Linux is a Kernel security mechanism for supporting access control security policy. The SELinux has three configuration modes:

- Disabled: Turned-off
- Permissive: Prints warnings
- Enforcing: Policy is enforced

- Using a text editor, open the config file:

```
#nano /etc/selinux/config
```

- And make sure that the policy is enforced:

```
SELINUX=enforcing
```

6. Network parameters

Securing your Linux host network activities is an essential task. Don’t always assume that your firewall will take care of everything. Here are some important features to consider for securing your host network:

- **Disable the IP Forwarding** by setting the net.ipv4.ip_forward parameter to 0 in *“/etc/sysctl.conf”*
- **Disable the Send Packet Redirects** by setting the net.ipv4.conf.all.send_redirects and net.ipv4.conf.default.send_redirects parameters to 0 in *“/etc/sysctl.conf”*
- **Disable ICMP Redirect Acceptance** by setting the net.ipv4.conf.all.accept_redirects and net.ipv4.conf.default.accept_redirects parameters to 0 in *“/etc/sysctl.conf”*
- **Enable Bad Error Message Protection** by setting the net.ipv4.icmp_ignore_bogus_error_responses parameter to 1 in *“/etc/sysctl.conf”*

Its Strongly recommended for using the Linux Firewall by applying the iptable rules and filtering all the incoming, outgoing and forwarded packets. Configuring your iptables rules will take some time, but it’s worth the pain.

7. Password policies

- People often reuse their passwords, which is a bad security practice. The old passwords are stored in the file “/etc/security/opasswd”. We are going to use the PAM module to manage the security policies of the Linux host. Under a debian distro, open the file “/etc/pam.d/common-password” using a text editor and add the following two lines:

```
auth sufficient    pam_unix.so likeauth nullok

password sufficient pam_unix.so remember=4    ----------Will not allow users to reuse the last four passwords.----------
```

- Another password policy that should be forced is strong passwords. The PAM module offers a pam_cracklib that protects your server from dictionary and brute-force attacks. To accomplish this task, open the file /etc/pam.d/system-auth using any text editor and add the following line:

```
/lib/security/$ISA/pam_cracklib.so retry=3 minlen=8 lcredit=-1 ucredit=-2 dcredit=-2 ocredit=-1
```

Linux will hash the password to avoid saving it in cleartext so, you need to make sure to define a secure password hashing algorithm SHA512.

- Another interesting functionality is to lock the account after five failed attempts. To make this happen, you need to open the file “/etc/pam.d /password-auth” and add the following lines:

    auth required pam_env.so

    auth required pam_faillock.so preauth audit silent deny=5 unlock_time=604800

    auth [success=1 default=bad] pam_unix.so

    auth [default=die] pam_faillock.so authfail audit deny=5 unlock_time=604800

    auth sufficient pam_faillock.so authsucc audit deny=5 unlock_time=604800

    auth required pam_deny.so

Open the file “/etc/pam.d/system-auth” and make sure you have the following lines added:

    auth required pam_env.so

    auth required pam_faillock.so preauth audit silent deny=5 unlock_time=604800

    auth [success=1 default=bad] pam_unix.so

    auth [default=die] pam_faillock.so authfail audit deny=5 unlock_time=604800

    auth sufficient pam_faillock.so authsucc audit deny=5 unlock_time=604800

    auth required pam_deny.so

After five failed attempts, only an administrator can unlock the account by using the following command:

    # /usr/sbin/faillock --user <userlocked>  --reset

- Also, another good practice is to set the password to expire after 90 days, to accomplish this task you need to:
    - Set the PASS_MAX_DAYS parameter to 90 in “/etc/login.defs”
    - Change the active user by executing the following command :
    - *#chage --maxdays 90 <user>*
    -
- The next tip for enhancing the passwords policies is to restrict access to the su command by setting the pam_wheel.so parameters in “/etc/pam.d /su”:

    auth required pam_wheel.so use_uid

- Disable the system accounts for non-root users by using the following bash script:

    #!/bin/bash

    for user in `awk -F: '($3 < 500) {print $1 }' /etc/passwd`; do

    if [ $user != "root" ]

    then

    /usr/sbin/usermod -L $user

    if [ $user != "sync" ] && [ $user != "shutdown" ] && [ $user != "halt" ]

    then /usr/sbin/usermod -s /sbin/nologin $user

    fi

    fi

    done

8. Permissions and verifications

    Permissions is one of the most important and critical tasks to achieve the security goal on a Linux host.

    - Set User/Group Owner and Permission on “/etc/anacrontab”, “/etc/crontab” and “/etc/cron.*” by executing the following commands:

#chown root:root /etc/anacrontab

#chmod og-rwx /etc/anacrontab

#chown root:root /etc/crontab

#chmod og-rwx /etc/crontab

#chown root:root /etc/cron.hourly

#chmod og-rwx /etc/cron.hourly

#chown root:root /etc/cron.daily

#chmod og-rwx /etc/cron.daily

#chown root:root /etc/cron.weekly

#chmod og-rwx /etc/cron.weekly

#chown root:root /etc/cron.monthly

#chmod og-rwx /etc/cron.monthly

#chown root:root /etc/cron.d

#chmod og-rwx /etc/cron.d

- Set the right and permissions on “/var/spool/cron” for “root crontab”

  #chown root:root <crontabfile>

  #chmod og-rwx <crontabfile>

- Set User/Group Owner and Permission on “passwd” file

  #chmod 644 /etc/passwd

  #chown root:root /etc/passwd

- Set User/Group Owner and Permission on the “group” file

  #chmod 644 /etc/group

  #chown root:root /etc/group

- Set User/Group Owner and Permission on the “shadow” file

  #chmod 600 /etc/shadow

  #chown root:root /etc/shadow

- Set User/Group Owner and Permission on the “gshadow” file

  #chmod 600 /etc/gshadow

  #chown root:root /etc/gshadow

9. Additional process hardening

   Restrict Core Dumps by:

   - Adding hard core 0 to the “/etc/security/limits.conf” file
   - Adding fs.suid_dumpable = 0 to the “/etc/sysctl.conf” file

   Configure Exec Shield by:

   - Adding kernel.exec-shield = 1 to the “/etc/sysctl.conf” file

   Enable randomized Virtual Memory Region Placement by:

   - Adding kernel.randomize_va_space = 2 to the “/etc/sysctl.conf” file

More info TBD