🖥️

BORN2BEROOT

Partitions:
# Al crear la VM, seleccionar la iso de debian y marcar “Skip unattended Installation”; establecer el tamano del disco duro de 30.8GB
# Ir a “Install”
# Al llegar a la seccion ”Partition disks”, seleccionar “Manual”
# Seleccionar el disco (SCSI1 (0,0,0) (sda) - 33.1 GB ATA VBOX HARDDISK)
# Crear la partition table en el disco (yes)
# (documentation: https://massive.io/es/transferencia-de-archivos/gb-vs-gib/)
# Seleccionar “pri/log 33.1 GB FREE SPACE” y crear una particion de 0.536870912GB PRIMARY at BEGINNING; El mount point es /boot → Done setting up the partition
#Seleccionar “Configure encrypted volumes” → (the following partitions are going to be formatted…) YES! → “Create encrypted volume“ → marcar “/dev/sda free #1 (32534MB; FREE SPACE)” → Continue →Erase data: No” → “Done setting up the partition?” Yes → Create encrypted volumes → Marcar “/dev/sda5 (32533MB: crypto)” → continue Finish
#Nos preguntara por el password de encriptacion
#Nos aparecera ahora un “Encrypted volume (sda5_crypt) - 32.5 GB Linux device-mapper (crypt)”
# y un “#1 32.5 GB f ext4”
#Seleccionar #1 y cambiar el “Use as:” a “Physical volume for LVM” → Done setting up the partition
#vamos a “Configure the Logical Volume Manager” → “Write the changes to disk and configure LVM?” YES → “Create Volume Group” → y poner el nombre “LVMGroup” → Seleccionar el dispositivo “/dev/mapper/sda5_crypt” y CONTINUE #Crearemos ahora los Logical volume para cada particion:
  - root (10.73741824GB)
  - swap (2.469606195GB)
  - home (5.36870912GB)
  - var (3.221225472GB)
  - srv (3.221225472GB)
  - tmp (3.221225472GB)
  - var-log (4.294967296GB)
#El paso es seleccionar “Create Logical Volume” → Seleccionarl el VG “LVMGroup” que hemos creado antes → ponerle el nombre que toca (p. ej. root) → y el tamano que tenemos en la tabla de arriba. Hay que hacerlo por todos en stack.
#Cuando los tengamos creados todos, “Finish”
#Ahora por cada Logical Volume, tenemos que asignar que SISTEMA DE FICHEROS implementa y que PATH monta. P.ej, “root le pondremos use as: ext4 y el Mount point: /”, “swap use as: swap area” o “var-log use as: ext4 y el Mount point en ‘enter manually’ y /var/log”
#Una vez tengamos todas las particiones listas, “Finish partitioning and write changes to disk” → “Write changes to disks?” YES!

- Differences between aptitude/apt and SELinux/Apparmor
(source: https://juncotic.com/apt-vs-apt-get-vs-aptitude-algunas-notas/)
aptitude tiene gui y es mas “user-friendly” que apt.
apt es mas “completo”
apt es una mejora de apt-get.
(source: https://phoenixnap.com/kb/apparmor-vs-selinux)
Mandatory Acces Control (MAC) systems, like AppArmor and SELinux, allow sysadmins to ***grant or deny access to resources and control systems built into the Linux kernel****. While both perform the same tasks, these system work differently and offer various features.*
Both provide security tools that isolate applications and limit access to an attacker that has compromised one part of the system.
AppArmor *works by* ***granting access first, then applying restrictions****.* ***SELinux****, however,* ***restricts acess to all applications by default and grants access only to users that present the proper certifications****.*
(**Apparmor**) sudo apparmor_status
(**SELinux**) sudo sestatus

- **Firewall** (UFW)
  - apt install ufw
  - ufw default deny incoming (*all INCOMING blocked by default*)
  - ufw default allow outgoing (*all OUTCOMING allowed by default*)
  - ufw allow 4242 (*only port 42 open*)
  - ufw enable (*activamos el firewall*)
  - ufw status verbose (*show status*)

- **Hostname:** <username>42
  - vi /etc/hostname
  - reboot *#para que lo aplique*

- **Password policy**:
  (*expire every 30 days, minimum days allowed before to modification = 2, warning message 7 das before password expire, at least 10 caracters long, must contain 1 uppercase, 1 lowercase, 1 number, it must not containt more than 3 consecutive identical characters, must not include the name of the user, the password must have at least 7 characters that are not part of the former password (not applied to root)*)
  - apt install libpam-pwquality *#instalamos la libreria que nos permite config adicional*
  - vi /etc/logins.defs
    - PASS_MAX_DAYS 30 *#expire every 30 days*
    - PASS_MIN_DAYS 2 *#minimum number of days avaliable of password*
    - PASS_WARN_AGE 7 *#Days for warnings before expiration*
  - vi /etc/security/pwquality.conf
    - difok = 7 *#the password must have at least 7 characters that are not part of the former password*
    - minlen = 10 *#10 characters long*
    - dcredit = -1 *#at least 1 number*
    - ucredit = -1 *#at least 1 uppercase*
    - lcredit = -1 *#at least 1 lowercase*
    - maxrepeat = 3 *#not contain more than 3 consecuitve identical characters*
    - usercheck = 1 *#must not include the name of the user*
    - enforcing = 1 *#the new password is rejected if it fails the check*
    - enforce_for_root *#apply this configuration to root account*

    - chage -l <username> @ #check password expiration for user
    - chage -M <num_days> <username> #Set expiration date in <num_days>
    - passwd --expire <username> #Expire the password for user
    - passwd #change password for actual user

- **SUDO** *(https://www.server-world.info/en/note?os=Debian_11&p=initial_conf&f=8)*:
  (*3 attemtps; custom message for wrong password; each action logged in "/var/log/sudo/", TTY mode enabled for security reasons, restricted to paths: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin*)
  - apt install sudo
  - su - *#login as root*
  - visudo #entramos a editar la configuracion de sudo
    - (tras "Defaults mail_badpass") Defaults   secure_path="…"
    - Defaults requiretty #forzamos a que sudo pueda SOLO ser ejecutado desde una sesion, no desde un daemon o servicio
    - Defaults passwd_tries=3
    - Defautls badpass_message="<CUSTOM_MESSAGE>"
    - (al final) Defaults syslog=local1
  - vi /etc/rsyslog.conf
    - (sobre la linea "auth,authpriv.*   /var/log/auth.log") local1.* /var/log/sudo/sudo.log
  - systemctl restart rsyslog #reiniciamos el servicio de rsyslog para que cargue la config
  - usermod -aG sudo <USERNAME> *#anadimos el usuario al grupo de sudo*

- **USER:** <username>
  (*in the groups user42 and sudo*)
  - adduser <NEW_USERNAME>

- groupadd user42 *#Creamos el grupo “user42”*
- usermod -aG sudo <username> #Anadimos <username> al grupo sudo
- usermod -aG user42 <username> #Anadimos <username> al grupo user42
- getent group <GROUP_NAME> *#verificar que usuarios hay en el grupo*

- **SSH service** (apt install openssh-server)
  (exposed at 4242; blocked root login)
  - apt install openssh-server
  - vi /etc/ssh/sshd_config
    - Cambiar “Port 22” por “Port 4242”
    - Cambiar “PermitRootLogin prohibit-password” a “PermitRootLogin no”
  - systemctl restart sshd
  - systemctl enable sshd

- **Monitoring** (script)
  (*architecture of OS and kernel version, # of pyshical processors, # of virtual processors, current available RAM and its utilization rate as percentage, current available memory an utilization rate ase percentage, current utilization rate of processors as percentage, last reboot date-time, LVM active?, number of active connections, number of users using the server, IPV4 address and MAC address, # of commands executed with sudo*)
  - lo copiamos en /root/monitoring.sh
  - le damos permisos de ejecucion: chmod +x /root/monitoring.sh
  - probamos a ejecutarlo: bash /root/monitoring.sh
  - **crontab -e** # editamos el crontab como root
    - */10 * * * * bash /root/monitoring.sh | wall #en mi caso el script lo tengo ubicado en /root/monitoring.sh, pero puede encontrarse en otro lugar.

```
#!/bin/bash

ARCHITECTURE=$(dpkg --print-architecture)
KERNEL_VERSION=$(uname -vrm)
PHYSICAL_PROCESSORS=$(awk '/^physical id/ && s[$NF]++==0' /proc/cpuinfo | wc -l)
VIRTUAL_PROCESSORS=$(awk -F: '/^physical id/ {ph=$NF} /^core id/ && a[ph,$NF]++==0' /proc/cpuinfo | wc -l)
CURRENT_RAM=$(free --mega | grep "Mem" | awk '{print $3}')
TOTAL_RAM=$(free --mega | grep "Mem" | awk '{print $2}')
CURRENT_RAM_PER=$(((100*${CURRENT_RAM}/${TOTAL_RAM})))
STORAGE=$(df -h | grep "^/dev" | awk '{print $6" - "$3"/"$2 " ("$5")"}')
CURRENT_CPU_PER=$(grep 'cpu ' /proc/stat | awk '{usage=($2+$3+$4+$6+$7+$8)*100/($2+$4+$5+$6+$7+$8)} END {print usage "%"}')
LAST_BOOT=$(uptime -s)
LVM_ACTIVE=$(if grep -Pq '/dev/(mapper/|disk/by-id/dm)' /etc/fstab || mount | grep -q /dev/mapper/; then echo "active"; else echo "ina
ACTIVE_CONNECTIONS=$(ss -tunlp | grep tcp | grep LISTEN | wc -l)
USERS_ACTIVE=$(who | wc -l)
DEVICE=$(ip route get 8.8.8.8 | awk -- '{printf $5}')
IPV4=$(ip route get 8.8.8.8 | awk -- '{printf $7}')
MAC=$(ip link show ${DEVICE} | tail -n 1 | awk '{print $2}')
COMMANDS_AS_SUDO=$(cat /var/log/sudo/sudo.log | grep -v "incorrect password"| wc -l)

echo "## SYSTEM STATUS ##"
echo "#Architecture: ${KERNEL_VERSION}"
echo "#CPU physical: ${PHYSICAL_PROCESSORS}"
echo "#vCPU: ${VIRTUAL_PROCESSORS}"
echo "#Memory usage: ${CURRENT_RAM}/${TOTAL_RAM}MB (${CURRENT_RAM_PER}%)"
echo "#Disk usage:"
echo "${STORAGE}"
echo "#CPU load: ${CURRENT_CPU_PER}%"
echo "#Last Boot: ${LAST_BOOT}"
echo "#LVM use: ${LVM_ACTIVE}"
echo "#Connections TCP: ${ACTIVE_CONNECTIONS} ESTABLISHED"
echo "#Users logged: ${USERS_ACTIVE}"
echo "#Network: IP ${IPV4} (${MAC})"
echo "#Sudo executions: ${COMMANDS_AS_SUDO} (more information at /var/log/sudo/sudo.log)"
```

- **WORDPRESS**
  - **MariaDB**
    - apt update
    - apt install mariadb-server
    - systemctl start mariadb
    - systemctl enable mariadb

- systemctl status mariadb *#get status for laughts*
  - mysql_secure_installation *#script inicial para definir una clave para root en la DB, todo "y" hasta el password*
  - mysql -u root -p *#nos conectamos a la DB con los credenciales creados antes*
    - CREATE DATABASE wordpress;
    - GRANT ALL PRIVILEGES on wordpress.* TO 'wordpress_user'@'localhost' IDENTIFIED BY 'Born2beroot';
    - FLUSH PRIVILEGES;
    - EXIT;
- **PHP**
  - apt install php php-mysql php-fpm php-curl php-gd php-intl php-mbstring php-soap php-xml php-xmlrpc php-zip
  - systemctl start php7.4-fpm *#Started by default at installation*
  - systemctl enable php7.4-fpm *#It's good in order to assure the boot at start of the server*
  - systemctl status php7.4-fpm *#get status for laugths*
- **Wordpress**
  - apt install wget
  - wget -O /tmp/wordpress.tar.gz https://wordpress.org/latest.tar.gz
  - mkdir -p /var/www/html
  - tar -xzvf /tmp/wordpress.tar.gz -C /var/www/html
  - cd /var/www/html/wordress
  - mv wp-config-sample.php wp-config.php *#creamos el archivo de config*
  - vi wp-config.php *#editamos*
    - define ('DB_NAME', 'wordpress'); *#basandonos en el nombre que le hemos puesto arriba en la config de MARIADB*
    - define ('DB_USER', 'wordpress_user'); *#como antes*
    - define ('DB_PASSWORD', 'Born2beroot');
  - sudo chown -R www-data.www-data /var/www/html/wordpress *#Cambiamos los permisos*
  - sudo chmod -R 755 /var/www/html/wordpress
- **Lighttpd**
  - apt install lighttpd -y *#Instalamos lighttpd*
  - systemctl start lighttpd *#Iniciamos lighttpd*
  - systemctl enable lighttpd *#Cargamos lighttpd "at boot"*
  - ufw allow 80 *#Abrimos el puerto 80 en el firewall*
  - vi /etc/lighttpd/lighttpd.conf *#Editamos la configuracion del lighttpd*
    - "server.document-root" ⇒ /var/www/html/wordpress *#Porque tenemos la carpeta de wordpress en aquella ubicacion*

  #Configuramos PHP-FPM para que funcione con lighttpd
  - vi /etc/php/7.4/fpm/pool.d/www.conf
    - reemplazar "listen = /run/php/php7.4-fpm.sock" con "listen = 127.0.0.1:9000"
  - vi /etc/lighttpd/conf-available/15-fastcgi-php.conf
    - reemplazar este bloque:

      "bin-path" ⇒ "/usr/bin/php-cgi",
      "socket" ⇒ "/var/run/lighttpd/php.socket",
    - por:

      "host" ⇒ "127.0.0.1",
      "port" ⇒ "9000",
  - lighty-enable-mod fastcgi *#recargamos los modulos*
  - ligthy-enable-mod fastcgi-php
  - systemctl restart php7.4-fpm *#reiniciamos los servicios*
  - systemctl restart lighttpd
  - deberiamos poder acceder a traves de la ip del nodo por el puerto 80

SERVICIO EXTRA: docker & docker-compose & traggo
# Instalamos docker & docker-compose:
- sudo apt-get update
- sudo apt-get install ca-certificates curl gnupg
- sudo install -m 0755 -d /etc/apt/keyrings
- curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
- sudo chmod a+r /etc/apt/keyrings/docker.gpg
- echo \

```
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
```

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
- sudo apt-get update
- sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-compose
- sudo usermod -aG docker <USER>

# Creamos el fichero compose del servicio de traggo
vi /home/<username>/traggo/docker-compose.yml

```
version: "3.7"
services:
  traggo:
    image: traggo/server:latest
    restart: unless-stopped
    ports:
      - 3030:3030
    environment:
      TRAGGO_DEFAULT_USER_NAME: "admin"
      TRAGGO_DEFAULT_USER_PASS: "mynewpassword"
    volumes:
      - ./traggodata:/opt/traggo/data
```

# Abrimos el puerto del firewall
- ufw allow 3030

# Iniciamos el servicio daemonizado
- (desde la carpeta) sudo docker-compose up -d

# Generamos el fichero signature.txt con el contenido del sha1 (shasum <NOMBRE_VM>.vdi)

Evaluation TIPS