Arch Install

Eben Rogers

October 4, 2018

1 Introduction

The purpose of this paper is to create a log of how I tend to install Arch Linux as well as to keep a record in order to refine the install process in future installs due to forgetfulness between install. Since

2 Pre-installation

2.1 Verify Boot Mode

We can verify EFI boot mode by running:

```
# ls /sys/firmware/efi/efivars
```

and making sure that directory exists. If it does, then you are good to go, otherwise you are likely booted in BOIS mode, refer to your motherboard manual to see if it is possible to boot using UEFI mode.

2.2 Internet

If you are connected via ethernet (or a VM) make sure you have internet by running:

```
# ping archlinux.org
```

If the ping has a response, it means you are connected to the internet and you are good to go. If you are not connected, you either need to configure your wireless interface of find some other way of connecting to the internet (I defer to the ArchWiki here. ArchWiki is your friend).

2.3 System Clock

Make sure your system clock is accurate by using timedatectl:

```
# timedatectl set-ntp true
```

You can check the service status by running:

```
# timedatectl status
```

3 Disk Setup

This is where this document will deviate from the installation tutorial from the ArchWiki. This guide will attempt to document the installation of an LVM on LUKS encrypted system with an on-disk boot partition as shown here.

3.1 Partitioning

Start by running lsblk to see your partition layout:

```
# lsblk
```

Select the partition you want to use from the output (eg. sda) and run the following command for it:

```
# gdisk /dev/sda
```

From there use the ? command to see a list of gdisk commands and figure out all of the needed commands. Start by making the first partition (using the 'n' command) within gdisk of size 512M for /boot of type EF00 (which should be the default type).

Then fill the rest with a second partition of type 8E00 (which will be our encrypted LVM volume) then use the w command within gdisk to write all the GTP data to the disk and exit.

3.2 The Encryption

Create the LUKS encrypted container using the command:

```
# cryptsetup luksFormat --type luks2 /dev/sda2
```

After filling out your password twice, the encrypted partition should now be set up. (Note: /dev/sda2 refers to the partition which will be the 'system' partition)

Now mount the encrypted partition so that it can be set up with the LVM and all needed partitions:

```
# cryptsetup open /dev/sda2 crypt
```

Now the decrypted container should be available at /dev/mapper/crypt

3.3 The Logical Volumes

Start by creating a physical volume on top of the LUKS container:

```
# pvcreate /dev/mapper/crypt
```

Followed by the creation of the volume group with the name 'volume':

```
# vgcreate volume /dev/mapper/crypt
```

Now create all the needed volumes (subpartitions if you will) for the system:

```
# lvcreate -L 8G volume -n swap
# lvcreate -L 32G volume -n root
# lvcreate -l 100%FREE volume -n home
```

You can modify the sizes '8G' and '32G' for whatever desired size the partitions will be.

3.4 Formatting the Filesystems

Format the filesystems for all the respective partitions on the system volumes:

```
# mkswap /dev/volume/swap
# mkfs.ext4 /dev/volume/root
# mkfs.ext4 /dev/volume/home
```

After formatting the filesystems, mount them:

```
# swapon /dev/volume/swap
# mount /dev/volume/root /mnt
# mkdir /mnt/home
# mount /dev/volume/home /mnt/home
```

3.5 Boot Partition

Since we don't have to do any fancy magic for the boot partition we can start right away by formatting (where sda1 is the desired boot partition):

```
# mkfs.fat -F32 /dev/sda1
```

Followed by making the needed directory:

```
# mkdir /mnt/boot
```

Followed by mounting:

```
# mount /dev/sda1 /mnt/boot
```

4 Installation

4.1 Initial Setup

Start by selecting the proper mirror servers from /etc/pacman.d/mirrorlist by moving nearby server to the top in order to speedup downloads for this and future installations:

```
# vim /etc/pacman.d/mirrorlist
```

4.2 Installation

Run lsblk one more time to make sure that all partitions are mounted properly.

```
# lsblk
```

The moment we've all been waiting for, installing base Arch (as well as vim) onto the system:

```
# pacstrap /mnt base vim
```

4.3 fstab

Since we have a custom partition setup the fstab setup is a little bit more involved than for the usual installation run both of the following:

```
# genfstab -U /mnt >> /mnt/etc/fstab
# genfstab -L /mnt >> /mnt/etc/fstab
```

Then execute

```
# vim /mnt/etc/fstab
```

And use vim to delete the three sets of lines referencing the lvm-based partitions by UUID and delete the one line referencing the boot partition by label.

An example fstab file is shown below as specified above:

```
# Static information about the filesystems.
# See fstab(5) for details.

# <file system> <dir> <type> <options> <dump> <pass>
# /dev/mapper/volume-root UUID=7e069bbc-322e-4a2e-9b3e-d1d773d16f0b
/dev/mapper/volume-root        /                ext4             rw,relatime        0 1

# /dev/mapper/volume-home UUID=a498084b-ec89-4211-905d-8dae6d7dbdd0
/dev/mapper/volume-home        /home            ext4             rw,relatime        0 2

# /dev/mapper/volume-swap UUID=db0dc117-df8d-405d-80c4-f840e50ab53a
/dev/mapper/volume-swap        none             swap             defaults,pri=-2        0 0

# /dev/sda1
UUID=3718-A53C              /boot            vfat             rw,relatime,fmask=0022,dmask=0022,codepage=437,ioch
```

4.4 Chroot and Timezone

Chroot into the new system:

```
# arch-chroot /mnt
```

Set the timezone:

```
# ln -sf /usr/share/zoneinfo/Region/City /etc/localtime
```

Which in the case of Amsterdam is:

```
# ln -sf /usr/share/zoneinfo/Europe/Amsterdam /etc/localtime
```

Then run hwclock to generate /etc/adjtime:

```
# hwclock --systohc
```

(This assumes the hardware clock is set to UTC)

4.5 Localization and Network

Uncomment en_US.UTF-8 UTF-8 and other needed locales in /etc/locale.gen then generate the locales with:

```
# locale-gen
```

Then set the LANG variable in locale.conf:

```
/etc/locale.conf
================================================================
LANG=en_US.UTF-8
```

Create the hostname file:

```
/etc/hostname
=============================================================================
myhostname
```

And corresponding hosts to the hosts file:

```
/etc/hosts
=============================================================================
127.0.0.1   localhost
::1         localhost
127.0.1.1   myhostname.localdomain        myhostname
```

5 Final Setup

5.1 Initramfs and Password

Edit the mkinitcpio.conf file to add the needed hooks: keyboard, keymap, encrypt, lvm2. Change:

```
/etc/mkinitcpio.conf
=============================================================================
HOOKS=(base udev autodetect consolefont modconf block filesystems fsck)
```

To:

```
/etc/mkinitcpio.conf
=============================================================================
HOOKS=(base udev autodetect keyboard keymap consolefont modconf block encrypt lvm2 filesystems fsck)
```

And add the following below the HOOKS next to all the commented out compression examples:

```
/etc/mkinitcpio.conf
=============================================================================
COMPRESSION="cat"
```

In order to disable ramdisk compression in order to speedup booting times. After setting the proper hooks and settings, generate the initramfs:

```
# mkinitcpio -p linux
```

Set the root password by running:

```
# passwd
```

5.2 Bootloader

We will be installing systemd-boot. Since the EFI system partition (esp) will be located at /boot any instances of 'esp' below can be replaced with /boot:

```
# bootctl --path=esp install
```

Whenever there is a new version of systemd-boot, the boot manager must be updated manually by the user. This is a safeguard because the user may not have the boot partition mounted to the system by default. However, since we will always have our beet partition mounted, we can set this up to be done automatically by creating a pacman hook at /etc/pacman.d/hooks/:

```
# mkdir /etc/pacman.d/hooks
# vim /etc/pacman.d/hooks/systemd-boot.hook
```

```
/etc/pacman.d/hooks/systemd-boot.hook
=====================================================================
[Trigger]
Type = Package
Operation = Upgrade
Target = systemd

[Action]
Description = Updating systemd-boot
When = PostTransaction
Exec = /usr/bin/bootctl update
```

Now that that is setup we need to configure the bootloader as well which requires changing two files:

```
# vim /boot/loader/loader.conf
```

```
/boot/loader/loader.conf
=====================================================================
default arch
editor 0
```

the 'default arch' refers to the 'arch' in the next filenameand the'editor' 0' stops people from being able to modify the boot paramaters from the boot screen. The next file is:

```
# vim /boot/loader/entries/arch.conf
```

```
/boot/loader/entris/arch.conf
=====================================================================
title Arch Linux
linux /vmlinuz-linux
initrd /initramf-linux.img
options cryptdevice=UUID=d6e55816-b96a-443c-bc0f-9d34a1c02dc6:crypt root=/dev/volume/root rw
```

If you are in doubt of any of the above paramaters for your system you can always use the vim :r command to read in text from a command like so ':r !command' where command can be anything. Useful commands here include: 'ls -l /dev/disk/by-uuid', 'ls /boot', and 'lsblk'.

The ':crypt' is the name of the encrypted system that will be mounted to /dev/mapper. So it should then mount /dev/mapper/crypt

The 'root=/dev/volume/root' parameter specifies the root partition so be sure to get that right for your system.

'rw' is there to make sure that the root partition gets mounted in read-write mode.

5.3 Enabling Microcode Updates

Depending on your processor you will want to execute one of the following two commands:

```
# pacman -S amd-ucode
# pacman -S intel-ucode
```

(amd-ucode for amd systems, intel-ucode for Intel systems, duh)
And then add the corresponding line file to the arch.conf file for the case of Intel:

```
/boot/loader/entries/arch.conf
=====================================================================
initrd /intel-ucode.img
```

Be sure to add the line above before the other initrd line and after the linux line. Your corresponding .img file (in the case you have an AMD system) can be found by running ls /boot or ':r !ls /boot' within vim.

5.4 Rebooting

Start by exiting the chroot environment by executing exit:

```
# exit
```

You may optionally unmount all partitions by executing:

```
# umount -R /mnt
```

Then reboot the system with:

```
# reboot
```

Don't forget to remove your installation media and then reboot and log in using your root password that you set earlier.

6 Setting Up the Arch Installation

6.1 Internet Connection

This still assumes that you are connected to ethernet since we haven't installed any NetworkManager-like utilities. Start by running:

```
# ip link
```

Which should produce an output something like this:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 08:00:27:ad:3c:69 brd ff:ff:ff:ff:ff:ff
```

On most computers the lo or 'loopback' device is used to sending messages to the computer and can be ignored when setting up initial internet access. In my case the only other device is enp0s3. In most cases the device that you are looking for is 'eth0'. Using this device name execute the following with ethernet connected:

```
# dhcpcd eth0
```

Or in my case:

```
# dhcpcd enp0s30
```

This should connect with the router its connected to and attain an ip address. If all goes well and you execute:

```
# ping archlinux.org
```

You should get responses to you pings.

6.2 Create a Non-Root User and Installing sudo

We will start by adding a simple non-root user by executing:

```
# useradd -m -G wheel -s /bin/bash username
```

Where -m specifies to create the home directory in the default location, '-G wheel' specifies the name of any additional groups the user should be in, where '-s /bin/bash' specifies the location of the login shell for the user and where username is the actual name of the user.

The above wheel group is the most commonly used group for system administrators and the /bin/bash login shell should be enough for now unless you have already installed another shell and want to use it in which case you should already know what you are doing.

After creating the use we of course also need to set a password for it:

```
# passwd username
```

Since we will still want to be able to some commands as root (like when installing packages) we should also install sudo to be able to do so without having to use 'su' or substitute user:

```
# pacman -S sudo
```

Then execute visudo and uncomment the line:

```
# %wheel ALL=(ALL) ALL
```

Such that it reads:

```
%wheel ALL=(ALL) ALL
```

Once this is done the user you just created (the one in the wheel group) should be able to run any command using sudo. You can check by executing:

```
# sudo -lU username
```

Where username is the name of the user. If it says username may run (ALL) ALL commands on the system, you are set to go.

6.3 Disabling Root Login and Enabling Hibernate

Once you are sure that you can use sudo using your default user (login to your username account and test a command like "sudo pacman -S") we can disable the use of the root account to make the system more secure by forcing an attacker to guess a username as well as a password.

Once you are 100% sure you can access root user privileges with your username account you may execute this knowing that if you mess it up that you will permanently lock yourself out of root privileges:

```
# passwd -l root
```

If you ever need to you can also unlock the root user using:

```
$ sudo passwd -u root
```

Again, keep in mind that if you are unable to use sudo when locking root and you log out of root, you can say goodbye to your root priveleges. YOU HAVE BEEN WARNED.

Next log out of your root user and log in to your username account where we will enable hibernation.

If you look above you will remember the /boot/loader/entries/arch.conf file. Within there there were a set of kernel parameters set after the option flag. If we want to have hibernation enabled (suspend to disk) we have to add one more parameters specifying the swap partition.

So if your arch.conf file is still the same change the line that use to look like:

```
/boot/loader/entries/arch.conf
==============================================================================
options SOME_PARAMETERS_HERE
```

To this:

```
options SOME_PARAMETERS_HERE resume=/dev/volume/swap
```

Using the command (Since we aren't root anymore):

```
$ sudo vim /boot/loader/entries/arch.conf
```

Where the last paramater specifies the location of the swap partition (in this case /dev/volume/swap). Next add the resume hook to the initramfs where:

```
/etc/mkinicpio.conf
==============================================================================
HOOKS=(base udev autodetect keyboard keymap modconf block encrypt lvm2 filesystems keyboard fsck)
```

Gets changed to:

```
HOOKS=(base udev autodetect keyboard keymap modconf block encrypt lvm2 filesystems keyboard resume fsck)
```

Next regenerate the initramfs:

```
$ sudo mkinitcpio -p linux
```

Hibernation should now be enabled. Reboot your system, log in, and execute:

```
$ sudo systemctl hibernate
```

Then start up you system again. If all is well, you system should now be in the state slightly before you hibernated it. NOTE: Don't forget to reconnect to the internet using dhcpcd as is described in **??**

6.4 AUR Helper and AUR Tutorial

Before you go any further I recommend reading up on the following four topics:

- Service Management: `https://wiki.archlinux.org/index.php/Systemd#Basic_systemctl_usage`
- System Maintenance: `https://wiki.archlinux.org/index.php/System_maintenance`
- Security: `https://wiki.archlinux.org/index.php/Security`
- Arch User Repository (AUR): `https://wiki.archlinux.org/index.php/Arch_User_Repository`

Knowing these four concepts is core to doing basic system maintenance and tuning in the future and contain plenty of links to other articles that are just as useful. For this section pay additional attention to the fourth link. Knowing how to install AUR packages yourself wihtout and AUR helper is crucial to being to install anything outside the official repositories.

For this next part we will also need the base-devel and git packages installed:

```
$ sudo pacman -S base-devel git
```

Hit enter on the prompt such that it will do the default which is to install all of the base-devel packages.

For this guide I will be installing the trizen package manager mostly because I'm used to it but also because IMO it's one of the better AUR helpers out there. Start by finding the trizen AUR page (yes, do this on your own I need to make sure you are paying attention) and use the Git Clone URL to clone the repo within the /tmp folder (which happens to be a ramdisk folder):

```
$ git clone https://link_to_git_repo
```

cd into the directory that gets created and read the contents ot PKGBUILD using 'vim' or 'less' to make sure there is no malicious code and then run the following command to build and install trizen:

```
$ makepkg -si
```

After this AUR packages can easily be installed or updated by invoking trizen like you would pacman:

```
$ trizen -S package_name
$ trizen -Syu
```

NOTE: make sure you do not execute trizen as root user as this can be extremely dangerous because the build process of AUR applications could (in rare cases) contain code that could cause damage to your system either maliciously or carelessly when allowed root user privilages.

NOTE: I don't know if this makes a big difference in the real world but I recommend always updating all you packages using pacman before doing so with trizen.

7 Installing Needed Applications and Desktop

7.1 Basic Internet Installation

For the internet connection manager I will be installing NetworkManager because it supports both wired and wireless connection management as well as having many possible gui applications that can be run on top of it to make its configuration easier.

Fore more information visit: `https://wiki.archlinux.org/index.php/NetworkManager`

To install it:

```
$ sudo pacman -S networkmanager
```

Then enable its system service/daemon:

```
$ sudo systemctl enable --now NetworkManager.service
```

This command enables the NetworkManager service while the –now flag starts it without having to invoke a second command or reboot to do so. After starting the NetworkManager service it is possible to see the network status by executing :

```
$ nmcli
```

And it is possible to start a graphical interface from the command line to manually create new connections (YES! Even Wifi connections as long as you know its interface **??**) by executing:

```
$ nmtui
```

7.2 Install and Setup Command Line Tools

Next install a large group of needed command line utilities that are useful during development and monitoring systems:

```
$ sudo pacman -S thefuck tldr entr htop tig zsh sl tmux ranger borg lshw
efibootmgr python openssh
$ trizen -S drive-bin entr
```

As well as removing a package that I personally don't need or want on my system:

```
$ sudo pacman -Rnscu nano
```

NOTE: The '-Rnscu' flag recursively remove packages both that are not needed and packages that dependent on the package in question. If you execute this command with this flag on the wrong package you could brick or remove large chunks of your system BE CAREFUL.

7.3 Installing i3/sway

7.4 Installing and Setting up a Display Manager

8 Setting up i3

8.1 Final Internet Installation

8.2 Commonly Used Applications

8.3 VPN: Setup Wireguard up and Down and openVPN

8.4 Installing Bluetooth Management Software

9 TODO

DONE

1.

TODO

- Install proper internet management packages
- Delete nano, I consider it bloat
- Maybe enable retaining of boot messages (Disable clearing of them)
- Install i3(MAYBE INSTALL Sway instead?) (without installing xinit since we will be using a display manager)
- Install a display manager (lightdm with lightdm-mini-greeter)
- Start customization of i3/sway (including installing yosemite san fransico font?)
- Install the following:
    1. maven, gradle, docker?
    2. java-openjdk, rust, LaTeX
    3. wireguard?
    4. firefox, keepassxc, filelight
    5. bcm20702a1-firmware (for Lenovo T430)
    6. Bluetooth Utilities
    7. redshift
    8.
-