GNU/Linux Administration - Support #447

The Paranoid Developer Arch Linux Setup

08/31/2014 03:21 PM - Daniel Curtis

Status:	Closed	**Start date:**	08/31/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Workstation	**Estimated time:**	8.00 hours
Target version:	Arch Linux	**Spent time:**	6.50 hours

Description

Like many developers, I like Linux; particularly Arch Linux. And like many sysadmins I like BSD, particularly FreeBSD. This is a guide of how I setup my recent developer laptop, it consists of a few goodies like:

1. ZFS
2. BlackArch PenTesting Distro
3. LUKS Emergency Self-Destruct
4. USB Boot Loader

Additional enhancements may come later. However getting all of this goodness onto a computer takes a little bit of patience and understanding.

Securely Wipe the Hard Drive

- Once booted into an Arch Live ISO, run the following to find the drive to erase:

```
fdisk -l
```

  - Now erase the primary hard drive, this guide uses /dev/sda for the primary hard drive:

```
dd if=/dev/zero of=/dev/sda bs=4M
```

  - Now erase the USB drive, this guide uses /dev/sdc for the USB Boot Loader drive:

```
dd if=/dev/zero of=/dev/sdc bs=1M
```

Adding the repository and install ZFS

The maintainer of ZFS on Arch has a signed repository that you can add to the /etc/pacman.conf.

- Add the **[demz-repo-archiso]** repo:

```
echo "[demz-repo-archiso]" >> /etc/pacman.conf
echo "Server = http://demizerone.com/$repo/$arch" >> /etc/pacman.conf
```

- Now repo key needs to be received and locally signed:

```
pacman-key -r 0EE7A126
pacman-key --lsign-key 0EE7A126
```

- Now update the repository information:

```
pacman -Sy
```

- Its time to install ZFS:

```
pacman -S zfs base-devel
```

- Load the ZFS kernel module:

```
modprobe zfs
```

- Check to see that the module was loaded:

```
lsmod | grep zfs
```

Install the patched cryptsetup

- Install the base-devel and libutil-linux packages:

```
pacman -S base-devel libutil-linux
```

- Grab the patch cryptsetup from the AUR

```
mkdir ~/src && cd ~/src
wget https://aur.archlinux.org/packages/cr/cryptsetup-nuke-keys/cryptsetup-nuke-keys.tar.gz
tar xzf cryptsetup-nuke-keys.tar.gz
cd cryptsetup-nuke-keys
```

- Install cryptsetup

```
makepkg -si PKGBUILD
y
y
```

Preparing the USB Boot Loader

- Find where the USB drive is by running:

```
fdisk -l
```

NOTE: Since I am using an Arch ISO from a USB drive, this guide will use /dev/sdc for the USB Boot Loader.

- Open cfdisk on the UDB drive:

```
cfdisk /dev/sdc
```

    - Erase all partitions, create a small partition for the bootloader, then add a partition for the rest of the drive, for storage:

```
[New]
primary
512
[Bootable] (make sure to have sda1 selected)
(Select Free Space)
[New]
```

```
primary
(Rest of the USB space)
[Write]
yes
[Quit]
```

- Make an ext3 partition for /boot:

```
mkfs.ext3 /dev/sdc1
```

- Make a FAT partition for general storage on the USB drive:

```
mkfs.fat /dev/sdc2
```

Setting up the encrypted hard drive

- Create a LUKS volume on /dev/sda

```
cryptsetup -i 15000 -c aes-xts-plain:sha512 -y -s 512 luksFormat /dev/sda
```

Enter passphrase:
Verify passphrase:

- Add the LUKS Emergency Self-Destruct passphrase:

```
cryptsetup luksAddNuke /dev/sda
```

Enter any existing passphrase: (existing password)
Enter new passphrase for key slot: (set the nuke password)
Verify passphrase: (verify the nuke password)

- Open the LUKS volume:

```
cryptsetup luksOpen /dev/sda root
```

NOTE: This will create the mapped device /**dev**/**mapper**/**root**. This is where the ZFS root will be installed.

1. (Optional) Create a backup of the LUKS Header

```
luksHeaderBackup /dev/sda --header-backup-file /path/to/backup-luksHeader.img
```

2. (Optional) Restore the LUKS Header from a backup
luksHeaderRestore /dev/sda --header-backup-file /path/to/backup-luksHeader.img

Preparing the encrypted primary hard drive

- Open cfdisk on the primary hard drive:

```
fdisk /dev/mapper/root
```

    - Add the primary partition for ZFS

```
g
n
1
[Enter]
[Enter]
t
39
w
```

Setting up the ZFS filesystem

- Create the zpool:

```
zpool create zroot /dev/mapper/root
```

WARNING: Always use id names when working with ZFS, otherwise import errors will occur.

- Create necessary sub-filesystem mount points such as /home and /vms can be created with the following commands:

```
zfs create zroot/home -o mountpoint=/home
zfs create zroot/vms -o mountpoint=/vms
```

NOTE: That if you want to use other datasets for system directories (/var or /etc included) your system will not boot unless they are listed in /etc/fstab! We will address that at the appropriate time in this tutorial.

Swap partition

ZFS does not allow the use swapfiles, but it is possible to use a ZFS volume as swap partition. It is important to set the ZVOL block size to match the system page size; for x86 and x86_64 systems that is 4k.

- Create a 2 GB (or whatever is required) ZFS volume:

```
zfs create -V 2G -b 4K zroot/swap
```

- Initialize and enable the volume as a swap partition:

```
mkswap /dev/zvol/zroot/swap
swapon /dev/zvol/zroot/swap
```

- Make sure to unmount all ZFS filesystems before rebooting the machine, otherwise any ZFS pools will refuse to be imported:

```
zfs umount -a
```

Configure the ZFS root filesystem

- First, set the mount point of the root filesystem:

```
zfs set mountpoint=/ zroot
```

1. and optionally, any sub-filesystems:

```
zfs set mountpoint=/home zroot/home
zfs set mountpoint=/vms zroot/vms
```

- Set the bootfs property on the descendant root filesystem so the boot loader knows where to find the operating system.

```
zpool set bootfs=zroot zroot
```

- Turn off swap, if enabled:

```
swapoff -a
```

- Export the pool:

```
zpool export zroot
```

WARNING: Do not skip this, otherwise you will be required to use -f when importing your pools. This unloads the imported pool.
NOTE: This might fail if you added a swap partition above. Need to turn it off with the swapoff command.

- Finally, re-import the pool:

```
zpool import -d /dev/mapper -R /mnt zroot
```

NOTE: -d is not the actual device id, but the /dev/mapper directory containing the symbolic links.

If there is an error in this step, you can export the pool to redo the command. The ZFS filesystem is now ready to use.

- Be sure to bring the zpool.cache file into your new system. This is required later for the ZFS daemon to start.

```
mkdir -p /mnt/etc/zfs
cp /etc/zfs/zpool.cache /mnt/etc/zfs/zpool.cache
```

1. If you don't have /etc/zfs/zpool.cache, create it:

```
zpool set cachefile=/etc/zfs/zpool.cache zroot
```

Installing Arch

- Start by mounting the boot partition

```
mkdir /mnt/boot
mount /dev/sdc1 /mnt/boot
```

- Now change the repository to **demz-repo-core**

```
vi /etc/pacman.conf
```

  - And change [demz-repo-archiso] to the following

```
[demz-repo-core]
Server = http://demizerone.com/$repo/$arch
```

- Then install the base system

```
pacstrap -i /mnt base base-devel grub openssh zfs wpa_supplicant
```

- Generate the fstab for filesystems, use:

```
genfstab -U -p /mnt | grep boot >> /mnt/etc/fstab
```

- Edit the /etc/fstab. If you chose to create datasets for system directories, keep them in this fstab!

```
vi /mnt/etc/fstab
```

  - Comment out the lines for the /, /root, and /home mountpoints, rather than deleting them. You may need those UUIDs later if something goes wrong. Anyone who just stuck with the guide's directions can delete everything except for the swap file and the boot/EFI partition. It seems convention to replace the swap's uuid with /dev/zvol/zroot/swap.
  - Edit /mnt/etc/fstab to ensure the swap partition is mounted at boot:

```
vi /mnt/etc/fstab
```

/dev/zvol/zroot/swap none swap defaults 0 0

- Setup the initial environment:

```
arch-chroot /mnt
```

  - Set a root password

```
passwd
```

  - Set a hostname

```
echo "archzfs" > /etc/hostname
```

  - Set a local time

```
ln -s /usr/share/zoneinfo/America/Los_Angeles /etc/localtime
```

  - Set a local language by uncommenting **en_US.UTF-8** in /etc/locale.gen, then running:

```
locale-gen
```

  - Set a wired network connection

```
cp /etc/netctl/examples/ethernet-dhcp /etc/netctl/wired
netctl enable wired
```

  - Set SSH to start at boot

```
systemctl enable sshd.service
```

LXDE

- Install the LXDE desktop

```
pacman -S lxde xorg xorg-xinit dbus gvfs gvfs-smb
echo 'exec startlxde' >> ~/.xinitrc
startx
```

Add an adminstive user

- It is generally a good idea not to run command directly as root, but rather as an administrative user using the sudo wrapper command
- First install sudo:

```
pacman -S sudo
```

- And create a user:

```
useradd -m -g users -s /bin/bash bob
```

- Set the user password:

```
passwd bob
```

- Edit the sudoers file:

```
visudo
```

  - And uncomment the following line:

```
%sudo   ALL=(ALL) ALL
```

- Create the sudo group:

```
groupadd -r sudo
```

- Add the bob user to the sudo group:

```
usermod -aG sudo bob
```

Setup the bootloader and initial ramdisk

When creating the initial ramdisk, first edit /etc/mkinitcpio.conf and add **zfs**before**filesystems**. Also, move **keyboard** hook before**zfs** so you can type in console if something goes wrong; and also put **usb**before**keyboard** and **encrypt**before**zfs**. You may also remove ~~fsck~~ (if you are not using Ext3 or Ext4).

- The HOOKS line should look something like this:

  HOOKS="base udev autodetect modconf block **usb keyboard encrypt zfs** filesystems"

- Regenerate the initramfs with the command:

```
mkinitcpio -p linux
```

Install and configure GRUB

- Install GRUB to the primary hard drive:

```
grub-install --target=i386-pc --recheck --debug /dev/sdc
```

Edit GRUB to boot off of the zroot pool

grub-mkconfig does not properly detect the ZFS filesystem, so it is necessary to edit grub.cfg manually.

- Edit the GRUB config:

```
/boot/grub/grub.cfg
```

  - Add or modify it similar to the following

```
set timeout=2
set default=0

# (0) Arch Linux
menuentry "Arch Linux" {
    set root=(hd0,msdos1)
    linux /vmlinuz-linux cryptdevice=/dev/sda:root root=/dev/mapper/root zfs=zroot rw
    initrd /initramfs-linux.img
}
```

Finish the setup process

- Exit the chroot environment:

```
exit
```

*Unmount all ZFS mount points:
zfs umount -a

- Unmount the bootloader partition:

```
umount /mnt/boot
```

- Export the zpool:

```
zpool export zroot
```

- Reboot:

```
reboot
```

After the first boot

If everything went fine up to this point, your system will boot. Once. For your system to be able to reboot without issues, you need to enable the zfs.target to auto mount the pools and set the hostid.

- For each pool you want automatically mounted execute:

```
zpool set cachefile=/etc/zfs/zpool.cache <pool>
```

- Enable the target with systemd:

```
systemctl enable zfs.target
```

When running ZFS on root, the machine's hostid will not be available at the time of mounting the root filesystem. There are two solutions to this. You can either place your spl hostid in the kernel parameters in your boot loader. For example, adding **spl.spl_hostid=0x00bab10c**, to get your number use the hostid command.

- The other, and suggested, solution is to make sure that there is a hostid in /etc/hostid, and then regenerate the initramfs image. Which will copy the hostid into the initramfs image. To do write the hostid file safely you need to use a small C program:

```
#include <stdio.h>
#include <errno.h>
#include <unistd.h>

int main() {
    int res;
    res = sethostid(gethostid());
    if (res != 0) {
        switch (errno) {
            case EACCES:
            fprintf(stderr, "Error! No permission to write the"
                        " file used to store the host ID.\n"
                        "Are you root?\n");
            break;
            case EPERM:
            fprintf(stderr, "Error! The calling process's effective"
                            " user or group ID is not the same as"
                            " its corresponding real ID.\n");
            break;
            default:
            fprintf(stderr, "Unknown error.\n");
        }
        return 1;
    }
    return 0;
}
```

- Copy it, save it as writehostid.c and compile it with:

```
gcc -o writehostid writehostid.c
```

    - Finally execute it and regenerate the initramfs image:

```
./writehostid
mkinitcpio -p linux
```

You can now delete the two files writehostid.c and writehostid. Your system should work and reboot properly now.

Installing BlackArch

Add the [Multilib] repository

- Make sure to uncomment the **[Multilib]** repo, similar to the following:

  [Multilib]
  Include = /etc/pacman.d/mirrorlist

- Refresh pacman:

```
pacman -Syy
```

Setting up as an Unofficial User Repository

BlackArch is compatible with normal Arch installations. It acts as an unofficial user repository.

1. Run the strap.sh script from http://blackarch.org/strap.sh as root:

```
curl -s http://blackarch.org/strap.sh | sudo sh
```

2. Run the following to add the BlackArch reposiroty to /etc/pacman.conf:

```
echo "[blackarch]" >> /etc/pacman.conf
echo "Server = http://mirror.team-cymru.org/blackarch/\$repo/os/\$arch" >> /etc/pacman.conf
```

3. Now run:

```
pacman -Sy
pacman -S blackarch
```

Installing other developer tools and packages

There are a few more packages that I use in my day-to-day tasks, for brevity I will refer over to Issue #410.

Optimizing and Tweaking

ZFS offers many useful features like snapshotting, replication, and dataset customization.

Virtual Machine Optimizations

Since I will be running virtual machines from my developer laptop, I will need a ZFS dataset that will enable/disable certain features like compression to allow VMs to run more smoothly. This is why the zroot/vms dataset was created during the initial setup

- Options for zfs can be displayed using the zfs command:

```
sudo zfs get all zroot/vms
```

  - This will return:

```
NAME        PROPERTY              VALUE                  SOURCE
zroot/vms   type                  filesystem             -
zroot/vms   creation              Sun Aug 31 14:47 2014  -
zroot/vms   used                  30K                    -
zroot/vms   available             29.5G                  -
zroot/vms   referenced            30K                    -
zroot/vms   compressratio         1.00x                  -
zroot/vms   mounted               yes                    -
zroot/vms   quota                 none                   default
zroot/vms   reservation           none                   default
```

```
zroot/vms  recordsize            128K                   default
zroot/vms  mountpoint            /vms                   local
zroot/vms  sharenfs              off                    default
zroot/vms  checksum              on                     default
zroot/vms  compression           on                     default
zroot/vms  atime                 on                     default
zroot/vms  devices               on                     default
zroot/vms  exec                  on                     default
zroot/vms  setuid                on                     default
zroot/vms  readonly              off                    default
zroot/vms  zoned                 off                    default
zroot/vms  snapdir               hidden                 default
zroot/vms  aclinherit            restricted             default
zroot/vms  canmount              on                     default
zroot/vms  xattr                 on                     default
zroot/vms  copies                1                      default
zroot/vms  version               5                      -
zroot/vms  utf8only              off                    -
zroot/vms  normalization         none                   -
zroot/vms  casesensitivity       sensitive              -
zroot/vms  vscan                 off                    default
zroot/vms  nbmand                off                    default
zroot/vms  sharesmb              off                    default
zroot/vms  refquota              none                   default
zroot/vms  refreservation        none                   default
zroot/vms  primarycache          all                    default
zroot/vms  secondarycache        all                    default
zroot/vms  usedbysnapshots       0                      -
zroot/vms  usedbydataset         30K                    -
zroot/vms  usedbychildren        0                      -
zroot/vms  usedbyrefreservation  0                      -
zroot/vms  logbias               latency                default
zroot/vms  dedup                 off                    default
zroot/vms  mlslabel              none                   default
zroot/vms  sync                  standard               default
zroot/vms  refcompressratio      1.00x                  -
zroot/vms  written               30K                    -
zroot/vms  logicalused           15K                    -
zroot/vms  logicalreferenced     15K                    -
zroot/vms  snapdev               hidden                 default
zroot/vms  acltype               off                    default
zroot/vms  context               none                   default
zroot/vms  fscontext             none                   default
zroot/vms  defcontext            none                   default
zroot/vms  rootcontext           none                   default
zroot/vms  relatime              off                    default
```

- The above output shows that compression is turned on, to disable the compression feature, while still keeping it active for all other datasets in the zpool, run:

```
zfs set compression=off zroot/vms
```

Snapshotting

The zfs-auto-snapshot-git package from AUR provides a shell script to automate the management of snapshots, with each named by date and label (hourly, daily, etc), giving quick and convenient snapshotting of all ZFS datasets. The package also installs cron tasks for quarter-hourly, hourly, daily, weekly, and monthly snapshots. Optionally adjust the --keep parameter from the defaults depending on how far back the snapshots are to go (the monthly script by default keeps data for up to a year).

To prevent a dataset from being snapshotted at all, set com.sun:auto-snapshot=false on it. Likewise, set more fine-grained control as well by label, if, for example, no monthlies are to be kept on a snapshot, for example, set com.sun:auto-snapshot:monthly=false.

- Install zfs-auto-snapshot-git from the AUR

```
cd ~/src
wget https://aur.archlinux.org/packages/zf/zfs-auto-snapshot-systemd-git/zfs-auto-snapshot-sys
temd-git.tar.gz
tar xzf zfs-auto-snapshot-systemd-git.tar.gz
cd zfs-auto-snapshot-systemd-git
makepkg -si PKGBUILD
```

- Enable and start snapshot timers as needed:

```
systemctl enable zfs-auto-snapshot-daily.timer
```

  - Additionally you can set the following to disable specific snapshots only (if unset, then it is "true"):

```
zfs set com.sun:auto-snapshot:daily=true zroot
zfs set com.sun:auto-snapshot:frequent=true zroot/home
zfs set com.sun:auto-snapshot:hourly=true zroot/vms
zfs set com.sun:auto-snapshot=false zroot/swap
```

  - Available intervals: **frequent**, **hourly**, **daily**, **weekly**, **monthly**

Conclusion

This may not the best solution for everyone, however for my use case it does. With LUKS, I have security for my system in a powered-off state and all the safety and features of ZFS on an operating system that is highly changeable configurable for the growth and adaptability of future projects.

Resources

- https://wiki.archlinux.org/index.php/ZFS
- https://wiki.archlinux.org/index.php/Installing_Arch_Linux_on_ZFS
- http://blackarch.org/download.html
- http://zfsonlinux.org/
- http://en.wikipedia.org/wiki/ZFS
- http://www.zfsbuild.com/2010/05/26/zfs-raid-levels/
- http://nex7.blogspot.ch/2013/03/readme1st.html
- http://wintelguy.com/raidcalc.pl
- https://calomel.org/zfs_raid_speed_capacity.html
- https://pthree.org/2012/04/17/install-zfs-on-debian-gnulinux/
- http://docs.oracle.com/cd/E19253-01/819-5461/

Related issues:

Related to GNU/Linux Administration - Support #410: GNet Developer User Arch ...	**Closed**	**07/08/2014**

History

#1 - 08/31/2014 07:02 PM - Daniel Curtis

- Description updated

- Status changed from New to Resolved

- % Done changed from 30 to 60

#2 - 09/01/2014 09:01 AM - Daniel Curtis

- Description updated

#3 - 09/01/2014 09:26 AM - Daniel Curtis

- Description updated

#4 - 09/04/2014 11:19 AM - Daniel Curtis

- Description updated

- % Done changed from 60 to 90

#5 - 09/04/2014 11:20 AM - Daniel Curtis

- Related to Support #410: GNet Developer User Arch Installation added

#6 - 09/04/2014 08:22 PM - Daniel Curtis

- Description updated

#7 - 09/05/2014 10:53 AM - Daniel Curtis

- Description updated

- % Done changed from 90 to 100

#8 - 09/05/2014 12:34 PM - Daniel Curtis

- Description updated

#9 - 12/12/2014 09:36 AM - Daniel Curtis

- Description updated

#10 - 12/12/2014 10:04 AM - Daniel Curtis

- Status changed from Resolved to Closed

#11 - 02/20/2015 06:14 PM - Daniel Curtis

- Description updated

- Category set to Workstation

#12 - 02/21/2015 12:04 PM - Daniel Curtis

- Description updated

#13 - 03/26/2015 05:11 PM - Daniel Curtis

- Description updated

- Target version set to Arch Linux

#14 - 04/19/2016 08:00 PM - Daniel Curtis

- Description updated