Switch Debian from legacy to UEFI boot mode

20.7.2023

Table of Contents

This note explains how to switch a legacy boot Debian/Ubuntu system into a UEFI boot system. Typical use case:

- switch a legacy boot installation into an UEFI one,
- reinstall a broken UEFI boot loader on Debian 7-12.

> This manual has been tested on Debian 7-12.

Please help to keep this manual up to date. It is hosted on GitLab[1]. There you can file issues[2] and pull requests[3].

[1] https://gitlab.com/getreu/legacy-to-uefi-boot
[2] https://gitlab.com/getreu/legacy-to-uefi-boot/issues

Before you start

1. Download and burn A recent Debian live CD or DVD[4]
2. Download and burn A Debian installation disk with rescue mode[5]
3. Back up your data!

1. Modify the hard-disk partitions

1.1. Boot a Debian live system

1. Enable UEFI in BIOS.
2. Boot a recent Debian live[6] system on USB or DVD.

1.2. Identify Debian's "/boot" partition

My legacy boot system had a 243 MiB ext2 partition mounted on `/boot`. This partition is never encrypted. It is where the grub files and Linux kernels reside. Check by double clicking on the partition icon on the live-disk-desktop and have a look inside.

```
# ls -l
total 21399
-rw-r--r-- 1 root root   155429 Sep 28 00:59 config-3.16-0.bpo.2-amd64
drwxr-xr-x 3 root root     7168 Nov  5 08:03 grub
-rw-r--r-- 1 root root 15946275 Nov  5 16:28 initrd.img-3.16-0.bpo.2-amd64
drwx------ 2 root root    12288 Nov 24  2012 lost+found
-rw-r--r-- 1 root root  2664392 Sep 28 00:59 System.map-3.16-0.bpo.2-amd64
-rw-r--r-- 1 root root  3126096 Sep 28 00:48 vmlinuz-3.16-0.bpo.2-amd64
```

```
# df -h
Filesystem             Size  Used Avail Use% Mounted on
...
/dev/sdb1              234M   28M  206M  13% /boot
```

As you can see in the following partition table of the Debian legacy boot system my `/boot` partition is number 1 (`/dev/sdb1`). On newer systems, SSD-disks might be listed as `/dev/nvme0n1p1`, where `p1` designates the first partition.

[3] https://gitlab.com/getreu/legacy-to-uefi-boot/pulls
[4] https://www.debian.org/CD/live/
[5] https://www.debian.org/distrib/netinst
[6] https://www.debian.org/CD/live/

> Although 1 is the default value for standard debian installations better check!

> The live system has identified this partition as `/dev/sdb`. The debian system on your hard-disk could reference it differently.

Partition table of the Debian legacy boot system

```
# fdisk -l /dev/sdb

Device      Boot   Start    End          Blocks    Id  System
/dev/sdb1   *      2048     499711       44032     7   HPFS/NTFS/exFAT
/dev/sdb5          501760   976771071    488134656 83  Linux
```

In legacy boot mode the `/boot` partition must have the `boot` -flag (*) set. This confirms our assumption: the `/boot` filesystem is on: `/dev/sdb1`.

```
# gdisk -l /dev/sdb
GPT fdisk (gdisk) version 0.8.5

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: not present
...
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048          499711   243.0 MiB   8300  Linux filesystem
   5          501760       976771071   238.2 GiB   8300  Linux filesystem
```

1.3. Create a GPT partition table

Transform the partition table from MBR to GPT with

```
  #gdisk /dev/sdb

  r       recovery and transformation options (experts only)
  f       load MBR and build fresh GPT from it
```

1.4. Create an UEFI partition

A good graphical tool is the Gnome Partition Editor `gparted`:

```
# gparted /dev/sdb
```

1. Shrink the `/boot` partition to 200 MB in order to free 43 MB (see partition 1 below).
2. Create a new 43 MB partition for efi using `gparted` with partition code `EF00` (EFI system) [7] and flag it **bootable**. Format the partition with a *fat32* [8] filesystem (see partition 2 below).
3. Grub2 needs additionally a *not* formatted 1 MB partition to store its stage two files [9]. The partition is referred to as *core.img* [10]. 1 MB is more than enough. Tag the partition as type `EF02` [11].

Leave the other partitions untouched (see partition 5 below).

Here the result:

Partition table of the Debian UEFI boot system

```
# gdisk -l /dev/sdb
GPT fdisk (gdisk) version 0.8.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sdb: 976773168 sectors, 465.8 GiB

...

Number Start (sector) End (sector)  Size       Code  Name
   1           2048       411647   200.0 MiB   8300  Linux filesystem
```

[7] You can change the partition code later

[8] fat32=vfat in `/etc/fstab`

[9] If you do not have an EF02 partition, Grub2 tries to find unallocated space between your partitions for its stage tow code. Since the space is unallocated in the GPT table, you may have a problem later on when you decide to adjust any partition boundaries. Omitting this step the following error message may occur: `GPT detected. Please create a BIOS-Boot partition (>1 MB, unformatted filesystem, bios_grub flag). This can be performed via tools such as Gparted. Then try again.`

[10] Wikipedia article about Grub2 [https://en.wikipedia.org/wiki/GNU_GRUB#Version_2_(GRUB_2)]

[11] See: GUID Partition Table (GPT) specific instructions [https://wiki.archlinux.org/index.php/GRUB#GUID_Partition_Table_.28GPT.29_specific_instructions]

```
2          411648       499711  43.0 MiB     EF00  Efi partition
3          499712       501759  1024.0 KiB   EF02  BIOS boot partition
5          501760    976771071   465.5 GiB   8300  Linux filesystem
```

If you haven't changed the partition code to `EF00` (or `EF02`) you can do it now within `gdisk` with the command `t`, followed by the partition number `2` (or `3`) and the partition code `EF00` (or `EF02`). Write the changes to disk and exit with the command `w`.

2. Mount the Debian filesystem

> The recovery mode of the Debian >= 9 Stretch installer disk automates all following subsections in this section. Try this first.

1. Reboot and enable UEFI in BIOS.
2. Insert a Debian installation disk.
3. Reboot again into the Debian installer disk and select **Advanced options → Rescue mode**
4. Configure keyboard, hostname, domain and network.
5. Unlock encrypted hard-disks.
6. Chose device to use as root system, e.g. `/dev/koobue1-vg/root` (for hostname `koobue1`, yours is different).
7. Answer: **Mount separate /boot partition?** with `yes`.
8. Choose **Execute a shell in** `/dev/koobue1-vg/root`.
9. Jump directly to section **Update Debian's /etc/fstab** hereafter in this manual.

If the above does not work work for you, follow the rest of this section.

The next step differs whether the `root`-filesystem is encrypted or not.

2.1. Mount a non-encrypted "root"-filesystem

1. Mount the `/` (root) filesystem.
   - For non-encrypted root filesystems a simple `mount` will do.

```
# mount -t ext4 /dev/sdb5 /mnt
```

2.2. Mount an encrypted “root”-filesystem

- For encrypted root file systems the mounting procedure can be a little tricky especially when the root filesystem resides inside a logical volume which is encrypted. This section shows how to mount and unmount an encrypted `root`-filesystem.

Find the device and partition of the to be mounted logical volume

1. Connect the disk with `host-system` and observe the kernel messages in `/var/log/syslog`

```
root@host-system:~# tail -f /var/log/syslog
sd 3:0:0:0: [sdb] 976773168 512-byte logical blocks: (500 GB/465 GiB)
sd 3:0:0:0: [sdb] Write Protect is of manually.
sd 3:0:0:0: [sdb] Mode Sense: 43 00 00 00
sd 3:0:0:0: [sdb] Write cache: enabled, read cache: enabled, doesn't
support DPO or FUA
 sdb: sdb1 sdb2 sdb3 sdb5
sd 3:0:0:0: [sdb] Attached SCSI disk
```

   The to be mounted device is `/dev/sdb`.

2. Find the partition

```
root@host-system:~# gdisk -l /dev/sdb
GPT fdisk (gdisk) version 0.8.5
...
Number Start (sector) End (sector)  Size       Code  Name
   1           2048       411647   200.0 MiB   8300  Linux filesystem
   2         411648       494821   43.0 MiB    0700
   3         494822       501759   1024.0 KiB  8300  Linux filesystem
   5         501760    976771071   465.5 GiB   8300  Linux filesystem
```

   The to be mounted logical volume of `disk-system` resides on `/dev/sdb5`.

Mount encrypted logical volume

1. Open decryption layer.

```
root@host-system:~# lvscan
  ACTIVE            '/dev/host-system/root' [231.03 GiB] inherit
```

```
  ACTIVE            '/dev/host-system/swap_1' [7.20 GiB] inherit
```

Logical volume is not registered yet. Do so.

```
root@host-system:~# cryptsetup luksOpen /dev/sdb5 sdb5_crypt
Enter passphrase for /dev/sdb5:
```

Enter disk password.

```
root@host-system:~# lvscan
  inactive          '/dev/disk-system/root' [457.74 GiB] inherit
  inactive          '/dev/disk-system/swap_1' [7.78 GiB] inherit
  ACTIVE            '/dev/host-system/root' [231.03 GiB] inherit
  ACTIVE            '/dev/host-system/swap_1' [7.20 GiB] inherit
```

Logical volume of `disk-system`is registered now. It contains one `root` partition (line 1) and one `swap` partition (line 2).

2. Activate logical volumes

```
root@host-system:~# lvchange -a y disk-system
```

Check success.

```
root@host-system:~# lvscan
  ACTIVE            '/dev/disk-system/root' [457.74 GiB] inherit
  ACTIVE            '/dev/disk-system/swap_1' [7.78 GiB] inherit
  ACTIVE            '/dev/host-system/root' [231.03 GiB] inherit
  ACTIVE            '/dev/host-system/swap_1' [7.20 GiB] inherit

root@host-system:~# ls /dev/mapper
control  disksystem-root  disksystem-swap_1  hostsystem-root
 hostsystem-swap_1  mymapper  sdb5_crypt
```

3. Mount logical volume

```
root@host-system:~# mount -t ext4 /dev/mapper/disksystem-root /mnt
```

Check success.

```
root@host-system:~# ls /mnt
```

```
bin  etc    initrd.img.old lib64      mnt  proc sbin    sys var
boot home        lib       lost+found mnt2 root selinux tmp vmlinuz
dev  initrd.img  lib32     media      opt  run  srv     usr
 vmlinuz.old
```

Unmount encrypted logical volume

This subsection is only for completeness. Skip it.

```
root@host-system:~# umount /mnt

root@host-system:~# lvscan
  ACTIVE            '/dev/disk-system/root' [457.74 GiB] inherit
  ACTIVE            '/dev/disk-system/swap_1' [7.78 GiB] inherit
  ACTIVE            '/dev/host-system/root' [231.03 GiB] inherit
  ACTIVE            '/dev/host-system/swap_1' [7.20 GiB] inherit

root@host-system:~# lvchange -a n disk-system
root@host-system:~# lvscan
  inactive          '/dev/disk-system/root' [457.74 GiB] inherit
  inactive          '/dev/disk-system/swap_1' [7.78 GiB] inherit
  ACTIVE            '/dev/host-system/root' [231.03 GiB] inherit
  ACTIVE            '/dev/host-system/swap_1' [7.20 GiB] inherit

root@host-system:~# cryptsetup luksClose sdb5_crypt
root@host-system:~# lvscan
  ACTIVE            '/dev/host-system/root' [231.03 GiB] inherit
  ACTIVE            '/dev/host-system/swap_1' [7.20 GiB] inherit
```

2.3. Mount the remaining filesystems

Either this way…

```
# mount /dev/sdb1 /mnt/boot
# mkdir /mnt/boot/efi
# mount /dev/sdb2 /mnt/boot/efi
# mount -t efivarfs efivarfs /sys/firmware/efi/efivars
# for i in /dev/ /dev/pts /proc /sys ; do mount -B $i /mnt/$i ; done
```

or this way, both commands do the same…

```
# mount /dev/sdb1 /mnt/boot
```

```
# mkdir /mnt/boot/efi
# mount /dev/sdb2 /mnt/boot/efi
# mount -t efivarfs efivarfs /sys/firmware/efi/efivars
# mount --bind /sys /mnt/sys
# mount --bind /proc /mnt/proc
# mount --bind /dev /mnt/dev
# mount --bind /dev/pts /mnt/dev/pts
```

Internet access

For internet access inside chroot:

```
# cp /etc/resolv.conf /mnt/etc/resolv.conf
```

3. Inside the “chroot” environment

3.1. Execute a shell in a “chroot” environment

Enter with:

```
# chroot /mnt
```

3.2. Update Debian’s “/etc/fstab”

Update the entries in `/etc/fstab` to reflect the partition changes above. We need to add the new **43.0 MiB EF00 Efi partition**:

```
# ls /dev/disk/by-uuid

040cdd12-8e45-48bd-822e-7b73ef9fa09f
19F0-4372
```

The UUID we are looking for is the only short 8-hex-digit ID, here: `19F0-4372`.

We add one line in `/etc/fstab` to mount the new partition persistently:

```
# echo "UUID=19F0-4372 /boot/efi vfat defaults 0 2" >> /etc/fstab
```

Check last line in `/etc/fstab`.

```
# cat /etc/fstab
```

```
# <file system> <mount point>  <type> <options>       <dump>  <pass>
/dev/mapper/koobue1-root /     ext4   errors=remount-ro 0       1
# /boot was on /dev/sdb1 during installation
UUID=040cdd12-8e45-48bd-822e-7b73ef9fa09f  /boot   ext2  defaults 0  2
/dev/mapper/koobue1-swap_1 none swap   sw              0       0
/dev/sr0        /media/cdrom0  udf,iso9660 user,noauto 0       0
#Jens: tmpfs added for SSD
tmpfs           /tmp           tmpfs  defaults,nodev,nosuid,size=500m 0 0
tmpfs           /var/lock      tmpfs
 defaults,nodev,nosuid,noexec,mode=1777,size=100m 0 0
tmpfs           /var/run       tmpfs
 defaults,nodev,nosuid,noexec,mode=0775,size=100m 0 0
UUID=19F0-4372  /boot/efi     vfat  defaults         0       2
```

> I use `/dev/mapper` for the encrypted file system and `tmpfs` because I have an SSD disk.

3.3. Mount remaining disks

```
# cat /etc/fstab
```

for not yet mounted entries and mount them manually e.g.

```
# mount /tmp
# mount /run
# mount /var/lock
...
```

3.4. Install grub-efi

```
# apt-get remove grub-pc
# apt-get install grub-efi
```

```
# grub-install /dev/sdb
```

Check presence of the efi file:

```
# file /boot/efi/EFI/debian/grubx64.efi
/boot/efi/EFI/debian/grubx64.efi: PE32+ executable (EFI application)
 x86-64 (stripped to external PDB), for MS Windows
```

A Debian entry should be listed here:

```
# efibootmgr
BootCurrent: 0000
Timeout: 0 seconds
BootOrder: 0000,2001,2002,2003
Boot0000* debian
Boot2001* EFI USB Device
Boot2002* EFI DVD/CDROM
Boot2003* EFI Network
```

Exit chroot environment.

```
exit
```

Reboot the system.

4. Validate the Debian bootloader in UEFI Bios

> From Debian 10 Buster on, the following should not be required any more. Skip it! More information can be found in this article about SecureBoot[12].

The BIOS will not accept the bootloader by default, because `/EFI/debian/grubx64.efi` is not the default path and because the file has no Microsoft signature.

This is why `grubx64.efi` has to be validated manually in the UEFI BIOS setup. In my InsydeH20 BIOS I selected:

Security → Select an UEFI file as trusted → Enter

Then browse to

```
/EFI/debian/grubx64.efi
```

in order to insert the grub boot loader in the trusted bootloader BIOS database.

> On my Acer E3-111 the BIOS menu entry was disabled by default. To enable it I had to define first a supervisor password.
> Security → Set Supervisor Password → Enter

[12] https://wiki.debian.org/SecureBoot

5. References

Tanguy

Tanguy: *Debian: switch to UEFI boot.* http://tanguy.ortolo.eu/blog/article51/debian-efi. April 2012.

Vulcan

Vulcan, Silviu: *Linux on the Acer E3-111 – Aspire E3-111-C5FN*. http://www.sgvulcan.com/linux-on-the-acer-e3-111-aspire-e3-111-c5fn/ . 09/2014.