Troubleshooting

Things will break. Start here when they do.

Restore Secure Boot Keys, Bootloader, LUKS TPM Key after Firmware update

After a firmware upgrade the firmware settings might get reset to their default values, including bootloater entries and custom secure boot keys.

Restore secure boot keys

First, you should restore any custom secure boot keys that might have been lost.

If you already had secure boot keys before the update you should be able to simply restore them by pushing them into the firmware again like you did on initial setup. Keep in mind that you still need to put the secure boot state into **setup mode** (e.g. by deleting all keys from storage) or else the keys will not be writable and restore will fail.

Boot the Arch Linux install media, mount your drives (especially the EFI system partition) and `arch-chroot` into it.

If you only boot Arch Linux:

```
sbctl enroll-keys
```

If you dual-boot Windows:

```
sbctl enroll-keys --microsoft
```

Restore boot loader

Depending on which boot loader you use you can probably restore it by just installing it again.

See the Boot Loader section for install instructions.

After restoring the boot loader, make sure to sign it with your keys and regenerate and re-sign the initrd as well!

```
sbctl sign-all
```

Regenerate TPM-based LUKS key

Since the firmware code changed the PIN you set up for a TPM-based LUKS key will probably stop validating (e.g. if you sealed against PCR 0).

You will need to re-enroll the TPM-based key into a free LUKS key slot in order to restore TPM-based PIN unlocking.

First, clear any TPM-based key from the LUKS device:

```
systemd-cryptenroll --wipe-slot=tpm2
```

Then, enroll a new key as described on Trusted Platform Module.

WARNING: Possibly missing firmware for module during initrd generation

During initrd generation mkinitcpio might output the following messages:

```
==> Starting build: '6.2.8-arch1-1'
  -> Running build hook: [base]
  -> Running build hook: [systemd]
  -> Running build hook: [sd-plymouth]
  -> Running build hook: [keyboard]
==> WARNING: Possibly missing firmware for module: 'xhci_pci'
  -> Running build hook: [sd-vconsole]
  -> Running build hook: [modconf]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: 'aic94xx'
==> WARNING: Possibly missing firmware for module: 'bfa'
==> WARNING: Possibly missing firmware for module: 'qed'
==> WARNING: Possibly missing firmware for module: 'qla1280'
==> WARNING: Possibly missing firmware for module: 'qla2xxx'
==> WARNING: Possibly missing firmware for module: 'wd719x'
  -> Running build hook: [sd-encrypt]
==> WARNING: Possibly missing firmware for module: 'qat_4xxx'
  -> Running build hook: [lvm2]
  -> Running build hook: [filesystems]
  -> Running build hook: [fsck]
```

These messages indicate that the firmware the mentioned kernel modules use are likely not installed, so the hardware these modules are intended for might not be functioning properly.

You can check which firmware files a module expects with `modinfo`:

```
modinfo xhci_pci
```

Which prints the following:

```
filename:       /lib/modules/6.2.8-arch1-1/kernel/drivers/usb/host/xhci-pci.ko.zst
license:        GPL
description:    xHCI PCI Host Controller Driver
firmware:       renesas_usb_fw.mem
srcversion:     2136F2C840FEFEEBE2620AB
alias:          pci:v*d*sv*sd*bc0Csc03i30*
alias:          pci:v00001912d00000015sv*sd*bc*sc*i*
alias:          pci:v00001912d00000014sv*sd*bc*sc*i*
depends:        xhci-pci-renesas
retpoline:      Y
intree:         Y
name:           xhci_pci
vermagic:       6.2.8-arch1-1 SMP preempt mod_unload
sig_id:         PKCS#7
signer:         Build time autogenerated kernel key
sig_key:        32:CA:80:C4:B5:BA:12:59:45:12:81:28:04:EF:9C:56:42:A8:A1:65
sig_hashalgo:   sha512
signature:      30:65:02:30:30:C2:EB:28:BB:C1:F4:09:1B:F8:94:7D:D6:6D:42:89:
□□□□2B:8C:74:4C:89:2C:F9:4F:6A:0C:92:64:B5:1C:97:76:15:DC:96:D6:
□□□□59:3B:6F:C9:E3:8F:89:16:2C:D9:36:AC:02:31:00:83:C4:FE:BF:75:
□□□□C5:8D:A7:82:01:08:79:3D:FF:8D:3C:54:41:95:6D:2C:5E:8B:C9:3B:
□□□□76:B0:1E:FE:5C:BA:23:66:30:A4:EA:D3:11:FF:7B:E4:93:67:DA:66:
□□□□02:16:6D
```

You can check to see whether the module gives any indication of whether intervention is required. If it lists hardware that is not present on the system, it can be safely ignored.

Getting rid of the warnings anyway

> **NOTE:** Also see the Arch Wiki article for `mkinitcpio` on the topic. At the time of writing, the firmware files for the `qat_4xxx` kernel module have not been made publicly available yet, so you will still receive a warning about this module in particular.

If you wish to not be warned about missing firmware files you can install the `mkinitcpio-firmware` meta package from the AUR:

```
yay -S mkinitcpio-firmware
```

This will install additional firmware files on your system to suppress these warnings.

Python Applications Stop Working

After a new minor release of Python (e.g. 3.9 -> 3.10) some Python packages from the AUR might stop working. This can be fixed by rebuilding the packages.

To get a list of the Python packages installed on your system you can run:

```
pacman -Qoq /usr/lib/python3.*
```

This gives you a list of packages that have files installed under the given directory (`/usr/lib/python3.{x..y}`). Since the directory changes when a new version of Python is released you will need to rebuild some or all of these packages.

To initiate a rebuild take your AUR Helper, e.g. with `yay`, issue the following command:

```
yay -S $(pacman -Qoq /usr/lib/python3.*) --answerclean All
```

This will pass all of the packages from the previous list as an argument to `yay`'s install command (`-S`) and rebuild the packages from scratch (`--answerclean All`). This will move them to the new directory structure.

However, if any of the packages needing to be rebuilt are not yet compatible with the new version of Python, the rebuild might still fail. Should this happen, pass a list of remaining package names by hand or use the `--ignore` option and pass it the package names that failed to rebuild (comma-separated, glob patterns are supported).

Discover does not offer system package updates

Sometimes Discover fails to delete the lockfile for PackageKit. In this case, delete the lockfile yourself.

```
sudo rm /var/lib/PackageKit/alpm/db.lck
```

Fonts in GNOME Flatpak apps are not anti-aliased on non-GTK desktops

Under Wayland GNOME apps get their anti-aliasing settings from XDG Portals. To make fonts look nice in GNOME apps on Wayland and a non-GTK desktop (e.g. KDE Plasma) you need the appropriate portal installed:

```
pacman -S xdg-desktop-portal-gtk
```

Then, restart the `xdg-desktop-portal` and `xdg-desktop-portal-gtk` user unit:

```
systemctl --user restart xdg-desktop-portal xdg-desktop-portal-gtk
```

After that restart your GNOME flatpak app and fonts should now be anti-aliased.