

A/B Updates and Secure Boot

ELCE 2026

Michael Opdenacker

Root Commit

Oct. 8, 2026



© 2024-2026 Root Commit. Licensed under CC BY-SA 4.0.

Embedded Linux consultant and trainer

- <https://rootcommit.com/about/michael-opdenacker/>
- Former founder of [Bootlin](#)
- New founder of [Root Commit](#)
- Free Software enthusiast and advocate (member of [April.org](#))
- Official Ambassador for the Yocto Project

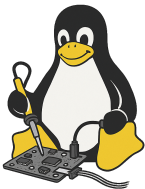


- Please take pictures of my talk and send them to me.



Consulting and engineering work

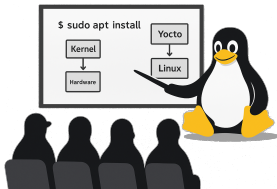
- Yocto Project
Project reviews, system implementation, new features
- Linux Kernel
Driver development, board bring-up, debugging
- Embedded Linux
Boot time, bug fixing, security and other optimizations



Training — <https://rootcommit.com/training>

- Yocto Project and OpenEmbedded — Free Materials!
- Linux kernel, board support, driver development
- Embedded Linux
- On-site or in [Amarula Solutions](#) offices in Prague, Amsterdam or Italy (Treviso, Carpi)

What's special: focus on practical activities, interactivity and learning techniques.



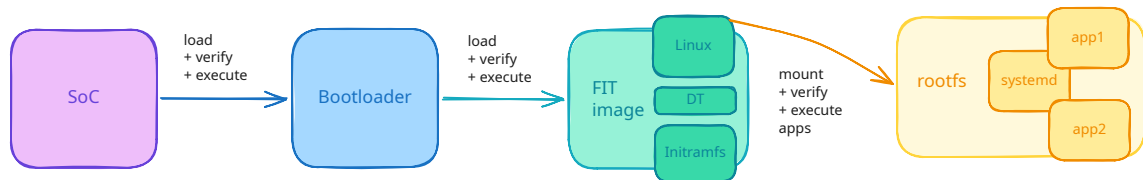


A human-like being without a soul, created by Rabbi Loew in Prague to protect the local Jewish community. However, the creature became uncontrollable and turned against its creators 😊.

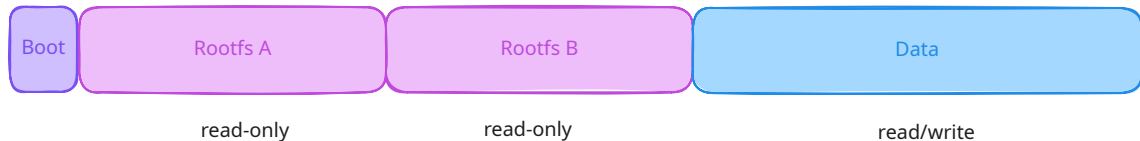
Image source:

https://commons.wikimedia.org/wiki/File:Mdm._Tussauds_Prag_8.jpg

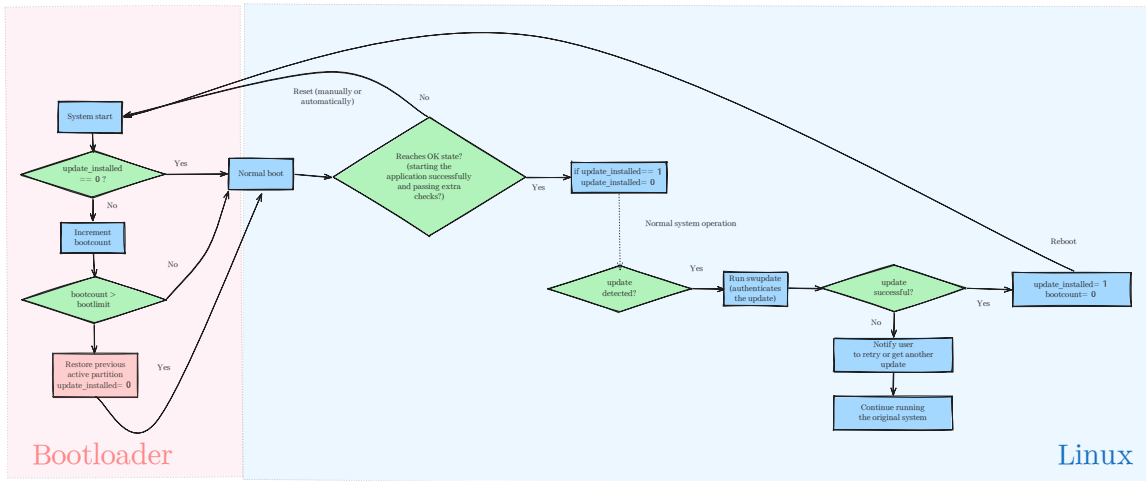
Introduction



To implement updates and recovery



A/B Update and Recovery Flow



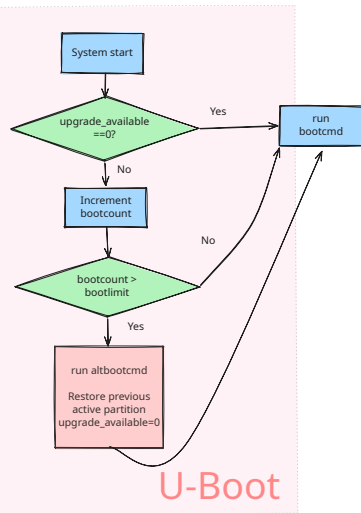
- A/B updates classical documentation suggests the use of a watchdog to reboot if an update turns out to be broken
- My customer **didn't want** a watchdog
- You can rely on the human user to reboot the machine when an update is wrong.

To allow A/B switching and updates in general...

- You can no longer have everything read-only!
- Need to change the kernel `root=` setting
- Need to keep the flexibility to change other kernel parameters in an update
- Need to have a read-write U-Boot environment

... all this without to open the doors to unwanted changes!

Implementation Bits



Turned out to be easier than expected

- It's all automated!
- Need to enable `CONFIG_BOOTCOUNT_LIMIT` in U-Boot
- Then U-Boot automatically increases bootcount every time it starts, **only when** `upgrade_available == 1`.
- Also necessary to define `bootlimit` and `altbootcmd` in U-Boot's environment

<https://docs.u-boot-project.org/en/latest/api/bootcount.html>

U-Boot patch

```
diff --git a/include/configs/verdin-imx8mm.h b/include/configs/verdin-imx8mm.h
...
+ "bootpart=2\0" \
+ "bootlimit=3\0" \
+ "altbootcmd=echo UPGRADE FAILED - RESTORING PREVIOUS VERSION;
    setenv bootcount 1; setenv upgrade_available 0;
    if test \"${bootpart}\" = \"2\"; then setenv bootpart 3;
    else setenv bootpart 2; fi; saveenv;
    run bootcmd \0" \
+ "bootcmd=setenv bootargs root=/dev/mmcblk0p${bootpart} console=ttymx0;
    load mmc 0:1 40000000 boot-${bootpart}/fitImage; bootm 40000000\0"
```

- Using libbootenv to update the U-Boot environment variables after an upgrade.

```
/etc/fw_env.config
```

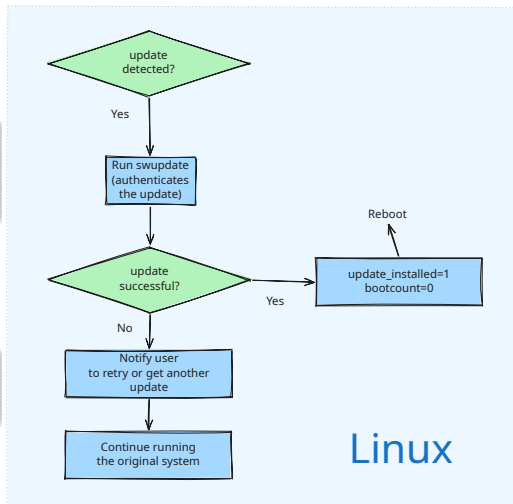
```
# Block device name Device offset Env. size  
/dev/mmcblk0boot0 -0x2200 0x2000
```

- swupdate can automatically set a variable after a successful upgrade:

```
swupdate configuration
```

```
CONFIG_UPDATE_STATE_BOOTLOADER="upgrade_available"
```

- Setting bootcount=0 is also part of the upgrade description.



- The U-Boot environment is only used from `libbootenv` (C library)
- `u-boot-fw-utils` (`fw_setenv`, `fw_getenv`) is not installed in the image
- This makes it harder for an attacker to alter the U-Boot environment

- U-Boot allows to lock the environment (thanks Ayoub Zaki for the tip!):

U-Boot configuration

```
CONFIG_ENV_WRITEABLE_LIST=y
```

- Then you can declare variables as read-only (r) or read-write(w) and with a specific type (s: string, d: integer)

U-Boot hard-coded environment patch

```
diff --git a/include/configs/verdin-imx8mm.h b/include/configs/verdin-imx8mm.h
...
+ #ifdef CONFIG_ENV_WRITEABLE_LIST
+ #define CFG_ENV_FLAGS_LIST_STATIC \
+ "bootlimit:dr,bootcmd:sr,altbootcmd:sr,bootcount:dw,bootpart:dw,upgrade_available:dw"
+ #endif
```

```
define CFG_ENV_FLAGS_LIST_STATIC \  
"bootlimit:dr,bootcmd:sr,altbootcmd:sr,bootcount:dw,bootpart:dw,upgrade_available:dw"
```

- bootlimit, bootcmd, altbootcmd are protected
- Kernel arguments (bootargs) are forced by bootcmd, so any other setting will be ignored.
- Observation: had to explicitly declare all read-write variables as read-write, otherwise changes to them are ignored. This means variables are read-only by default.
- Observation: U-Boot will forbid changes from the U-Boot shell
- An attacker on Linux may get root access and might manage to modify the environment, but U-Boot will reject changes to read-only variables.
- bootpart can be given an incorrect number (3, 4, 5...). The system will fail to boot, or be forced to use the old value (rollback attack, need protection against this).

- bootargs are hardcoded in U-Boot
- This offers protection against booting from another partition (USB, MMC) or from starting a root shell on the console.
- **But** an update may need to tweak the kernel parameters without having to update U-Boot too.
- Fortunately, a kernel image can embed its own additional settings

- Let the kernel embed and add this own parameters
- Good that root= is set outside. This way, the generated FIT image doesn't depend on the A or B partition.

```
linux-stable_%.bbappend
```

```
FILESEXTRAPATHS:prepend := "${THISDIR}/${PN}:"
```

```
do_configure:prepend() {  
    install -m 0644 ${UNPACKDIR}/cmdline.bootconfig ${S}/  
}
```

```
SRC_URI += " \  
    file://defconfig \  
    file://cmdline.bootconfig \  
"
```

```
defconfig
```

```
...  
CONFIG_BOOT_CONFIG=y  
CONFIG_BOOT_CONFIG_EMBED=y  
CONFIG_BOOT_CONFIG_EMBED_FILE="cmdline.bootconfig"
```

```
cmdline.bootconfig
```

```
kernel {  
    page_alloc.shuffle=1  
    hash_pointers=always  
    nosmt  
}
```

- Our U-Boot scripts are locked down
- But don't believe your system is safe!
- If someone can cause your scripts to fail, U-Boot will fall back to a shell
- I did that a few times to hack commercial boards and load my own kernel and DTB instead.

The ultimate solution to block the U-Boot shell and avoid information leakage

- Use this U-Boot configuration setting: `CONFIG_DISABLE_CONSOLE=y`

Parameter description

`config DISABLE_CONSOLE`

`bool "Add functionality to disable console completely"`

`help`

`Disable console (in & out).`

- In Yocto, easy to create a config fragment (`.cfg` file):
 - `bitbake -c menuconfig u-boot`
add the configuration setting(s)
 - `bitbake -c diffconfig`
generate a config fragment from the changes

- Takes care of generating all the parts from you (some details can depend on your BSP layer)
- Makes it easy to share or differentiate settings between development and production images.
- Strong community has your back



- A read-write U-Boot environment is very useful to implement A/B updates
- In U-Boot, you can protect the environment 🌳🌍😏
- With `CONFIG_ENV_WRITEABLE_LIST=y`,
you can make just a few specific variables read-write
- U-Boot's management of `bootcount`, `bootlimit`, `upgrade_available` and `altbootcmd` is automatic. No need to implement your own scripts!
- The Linux command-line is critical too, but you can protect it without sacrificing the ability to extend it, using a built-in *bootconfig*
- The various techniques I shared don't promise perfect security, but they make the system harder to compromise.
- My implementation is not perfect, but it hardens the system
- Security is made of multiple mechanisms, you need to understand what you're doing. Don't let the Golem do it for you, it will backfire!

- A read-write U-Boot environment is very useful to implement A/B updates
- In U-Boot, you can protect the environment 🌳🌍😄
- With `CONFIG_ENV_WRITEABLE_LIST=y`,
you can make just a few specific variables read-write
- U-Boot's management of `bootcount`, `bootlimit`, `upgrade_available` and `altbootcmd` is automatic. No need to implement your own scripts!

- The Linux command-line is critical too, but you can protect it without sacrificing the ability to extend it, using a built-in *bootconfig*
- Do this by setting `CONFIG_BOOT_CONFIG=y` and `CONFIG_BOOT_CONFIG_EMBED=y`
- The various techniques I shared don't promise perfect security, but they make the system harder to compromise.
- Security is made of multiple mechanisms, you need to understand what you're doing. Don't let the Golem do it for you, it will backfire!

Questions? Comments?

- mo@rootcommit.com
-  <https://fosstodon.org/@MichaelOpdenacker>
- Signal: rootcommit.01
- Matrix: @michael.o:matrix.org
- XMPP: omichael@conversations.im
- Slides available under the CC-BY-SA 4.0 license
<https://rootcommit.com/pub/conferences/2026/elce/a-b-updates-and-secure-boot/>
- Sources ($\LaTeX$):
<https://gitlab.com/rootcommit/ab-updates-and-secure-boot/>

