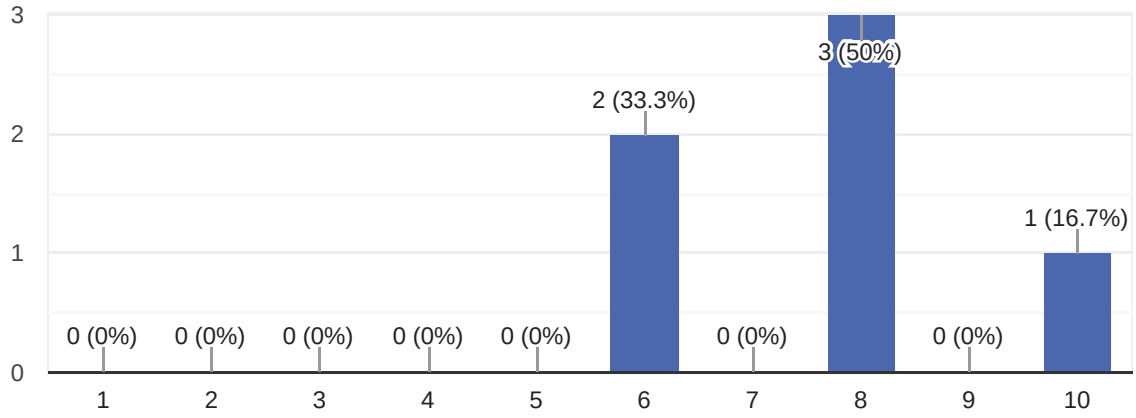
Bootlin training course evaluation

6 responses

Overall rating of the course

Copy chart

6 responses



Comments and suggestions

4 responses

Overall is good. It's better if having some more examples because the content is not easy and sometime a concrete example may help to better understand.

This is overall a good course, because of the contents: it tries to give an overview of the security tools available in embedded linux. While there were a lot of topics handled, for me it still did not end with like a nice session where everything comes together.
If I look at the structure of the course compared to how projects are handled in our company, then I would tackle the topic of updating (with RAUC) earlier in the course, e.g. immediately after creating the keys and talking about vulnerabilities...

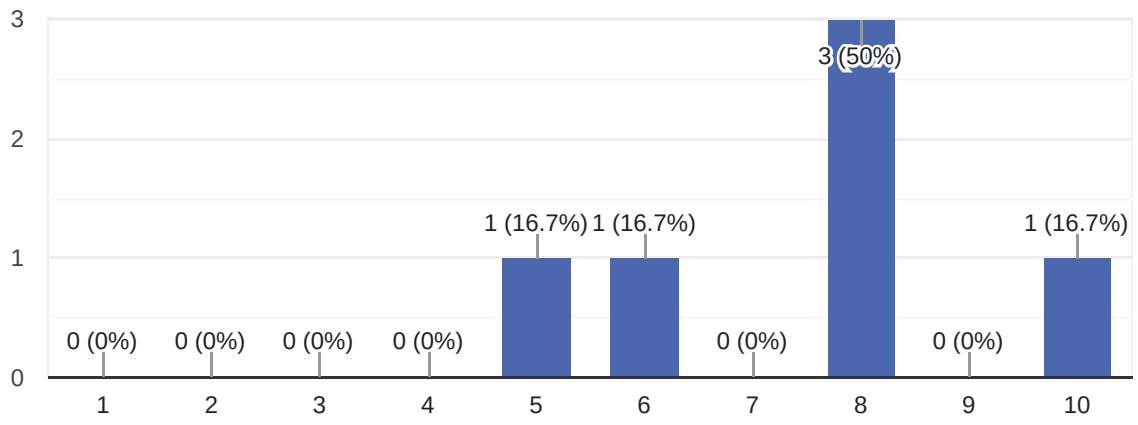
Overall overview as reminder and introduction was good.

Overall content was good was good. Some topics, the level of detail was off. For example, a lot of time (+- 1 whole training day) was spent on the "mathematical / internal workings" of certain types of crypto. A small intro + a focus on the pro's / con's and use-cases would be more useful, less boring and give some more time for important topics.

How useful were the lectures?

Copy chart

6 responses



Comments and suggestions

4 responses

Many key points on all the essentials of security aspects for linux embedded system. The lectures allows next step to deep dive into point by point. It takes time but it worth.

The lectures were ok, but maybe a bit too slow. The first afternoon was solely explaining about cryptography, but for me, it would have been beneficial to go less theoretical about that topic and to add e.g. a primer on Yocto and how the yocto build is structured for the labs, just so you have a reference to get back to when you encounter difficulties during the lab.

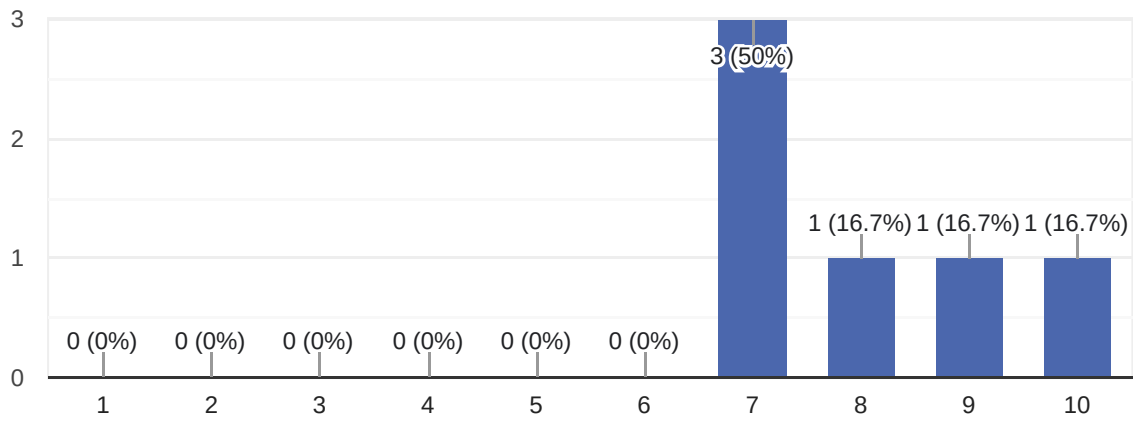
Level of deepness was satisfying for me

Usefulness of the lectures varied heavily from topic to topic. See previous questions for an example about crypto. Lectures are (somewhat understandable) very tied to NXP due to the devkit choice. However, topics like secure boot would benefit heavily from a) making it generic, b) explaining more than 1 vendor implementation. From our experience, NXP by far has on of the simplest secure boot chains, giving a pretty incomplete picture. UEFI secure boot for example was mentioned briefly but not much more due to it being "mainly for x86/64 laptops and servers". However many new AARCH64 chips are standardizing on UEFI as well, Qualcomm, Xilinx, R-Pi,.. and UEFI secure boot is significantly more complex than NXP's AHAB. In the OP-TEE chapter, writing a hello world or printing the EL level in OP-TEE and ATF is fun but realistically this is not something a lot embedded engineers will ever need to do. Topics like, secure storage REE or RPMB, common pitfalls in configuration (CFG_RPMB_WRITE_KEY, CFG_RPMB_TESTKEY) are so much more important when trying to teach people how to use these systems and mechanisms.

How useful were the practical demos?

Copy chart

6 responses



Comments and suggestions

4 responses

Demo are interesting but not easy to follow step by step if the trainer has at least a minimal on yocto, linux and so on. The solution is helpful when thing gets really stuck.

Practical demo's are ok, in the sense that the exercises in the labs touched upon various techniques. The difficulty here lies in the fact that the trainer also did the labs during the online session, and I could not keep my focus on that type of session. I have redone the labs myself, and got almost everything working. However, the solutions sections was insufficiently detailed, if you ask me. And at the end of the kernel FIT verification lab, there is this sentence "Once you're done, and you have flashed you new signed FIT image, your U-Boot boot log should look like this:"... But up until that point we only flashed a complete wic image to the sd card, and now all of a sudden a single .itb file should land somewhere. Unfortunately, the info of how the sd card is built up in the security-training-image.freheit93.wks.in file is only coming in the section about A/B updates... Which is again a reason why I would like to see this being handled earlier in the course.

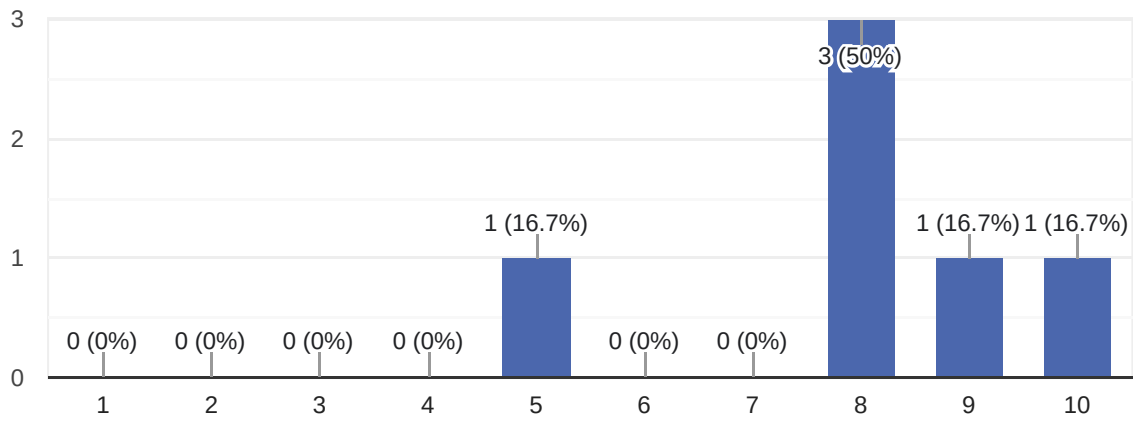
Labs need improvements based on how the lecturer struggled with it as well.

Demo's themselves are useful, solutions still have quite some gaps / missing steps.
For example: generating keys for u-boot FIT images signatures, manual steps for signing FIT image are demonstrated, then it just mentioned "once you've flashed this image". But for people not very familiar with Yocto or NXP tooling, this is very unclear.

How would you rate the overall organization of the course?

Copy chart

6 responses



Comments and suggestions

4 responses

Overall is good. It's better if having some more examples because the content is not easy and sometime a concrete example may help to better understand. It not easy to put all the into only 5 half-day.

The organization of the course was good. Registration went well, information provided is ok. The matrix chatroom seems to be a bit overrated, because the google meet sessions also have a chat in there.
I would (for online trainings) reconsider how to divide the time between lectures and demo's. I also think that more interaction is welcome during the sessions.

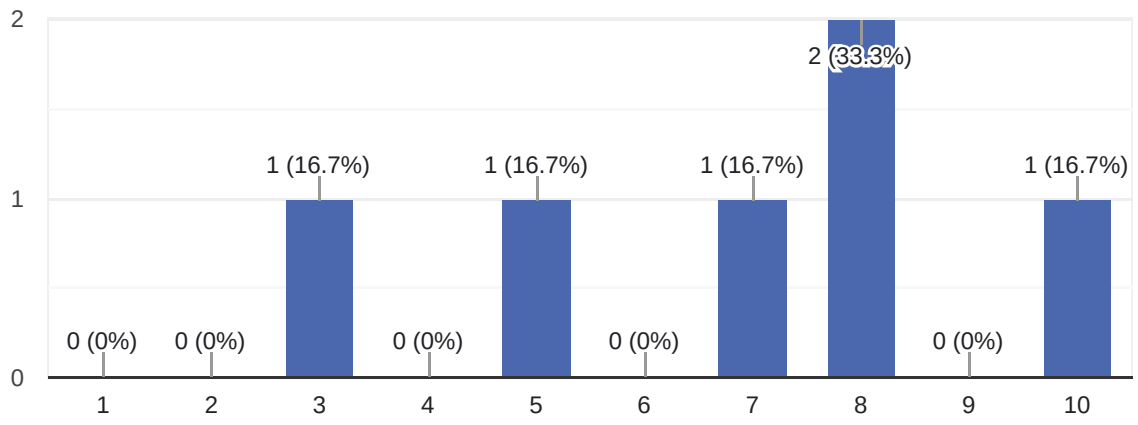
Well organized and given the age of the course pretty good.

Everything around the course was organized pretty good, Like we are used to when it comes to Bootlin trainings. However, the course itself, the organization could improved, watching the trainer read and try to execute the labs live is a bit tedious.

How would you rate the trainer?

Copy chart

6 responses



Comments and suggestions

4 responses

As this is the second courses of the trainer (as he said), it's not easy to do present, and do live exercise at the same time and especially around the complex things like Yocto and Sec linux. In the future it will go better. The trainer is solid with his knowledge and experience. This is his domain.

The trainer was ok, but I do think he can
- work on his own experience (too many topics he said were also to him not well known)
- work on his style, to make the sessions somewhat more varied

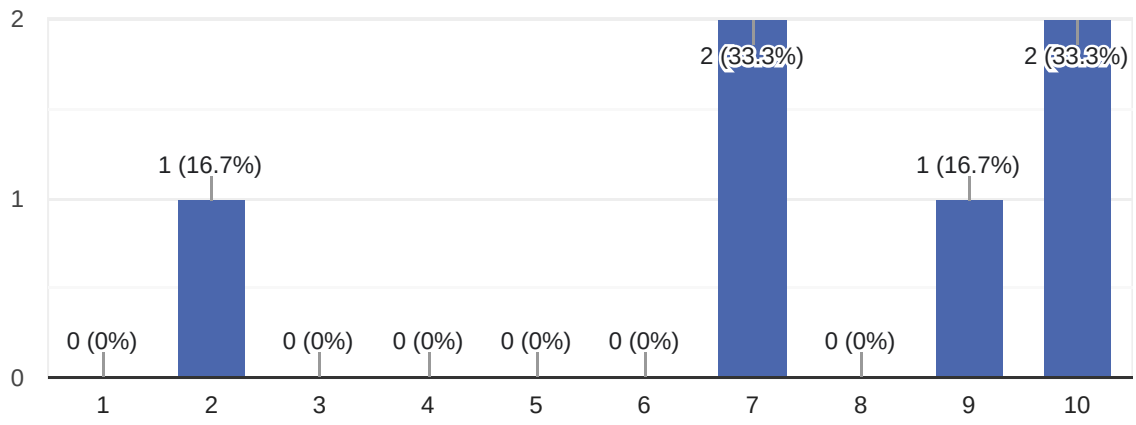
Mathieu did a good job, was prepared but requires some more trainings to be really fluent in the topic.

This is not what we are accustom to from Bootlin. While the trainer clearly demonstrated much knowledge in Yocto, the domain knowledge to properly give this training was lacking. Many topics were prefixed with the disclaimer of never or rarely having used the technology or methods leading to incomplete or even incorrect explanations of technologies. For example during the measured boot section when prompted for use-cases, no proper use-cases could be provided. dm-crypt TPM/PCR based decryption was completely missed as a very important usecase for embedded systems with measured boot. IMA and EVM topics, the slides were read but no further details.

How did the course meet your learning objectives?

Copy chart

6 responses



Comments and suggestions

4 responses

Allow me to continue deep dive into this domain based on the key points delivered from course. It's just the beginning of what I must continue to improve

learning objectives are ok, but like I already mentioned, I would have expected the session to end in a grand finale ;-)

Reminding training on the Embedded Linux Security topics

Sadly not, we were looking to reinforce our internal knowledge about embedded security , measure the solutions we build against domain expertise and potentially learn about techniques and methods to further strengthen the products we build. Having followed multiple trainings over the years by Bootlin this is far from the quality and deep level of knowledge we were expecting.

What part(s) of the course did you like most?

5 responses

All

Secure Boot

I like the subjects touched upon

RAUC and AT-F part

crypto sboms secure boot

What part(s) of the course did you like least?

3 responses

None

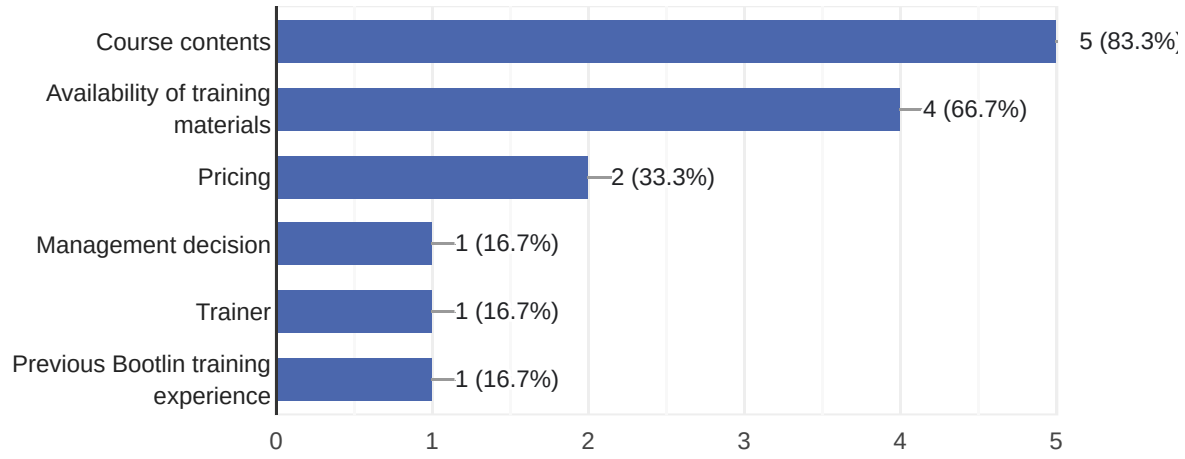
I would have loved to see a more enthusiastic trainer

S/MIME and TLS (TLS should be pretty standard for everybody -> refer to Ed Harmoush course here should be enough)

What reasons prompted you to choose a Bootlin course?

Copy chart

6 responses



Comments

2 responses

Here is some note I write down during exercise to better understand

BUILD YOCTO IMAGE FOR I.MX93 FRDM BOARD

Fix security-training-imx93-frdm.yaml to able to build :

add "path: ." to .yml file

meta-security-training:
path: .
layers:
meta-security-training:

PKI

NOTE : can explain the difference between ? !!!!!!!

host => target : encrypt host_message and then sign host_message before sending to target
target => host : sign target_message and then encrypt targe_message before sending to host

EXPLORING THE SECURE WORLD

Observing Exception Levels

Using the patches proprosed from solution

Writing a small Trusted Application (TA)
2-secure-world/meta-security-training/recipes-security/test_ta/files/ta/test-ta.c

NOTE: in this excercice, can explain that example will crash due to the access to the exception registers at the wrong level. This helps to know that this is the goal of exercice.

It take me a lot of time to finally see : test-ta crash due to the TA app is TrustApp runing on the EL0. At this level EL0, app can not acces to registers that required APP must be in EL1 or EL2.

SETUP AHAB ON I.MX93

Really difficult to follow the whole exercices. It takes me 2 nights to finish this.

NOTE : the "Detailed example: i.MX93 - AHAB container template" in slide 115-116-117/310 doesn't have the same file name regarding the solution name.

PRACTICAL LAB - EXPLORING SECURE BOOT LATER STAGES

This is a plus to the previous exercice for manually build and flash FIT on uboot part (Activating signature verification in U-Boot)

To build the whole image with FIT.
"board_setup/meta-security-training/recipes-kernel/linux/linux-kiss_%.bbappend"
=====
inherit kernel-fitimage

FILESEXTRAPATHS:prepend := "${THISDIR}/linux-kiss:"

# reuse the kernel_fitimage recipe fit_signing_key.pem
FIT_GENERATE_KEYS = "0"

UBOOT_SIGN_ENABLE = "1"
#UBOOT_SIGN_KEYDIR = "${TOPDIR}/../keys/fit-signing"
UBOOT_SIGN_KEYDIR = "${TOPDIR}/../../FIT-verification"

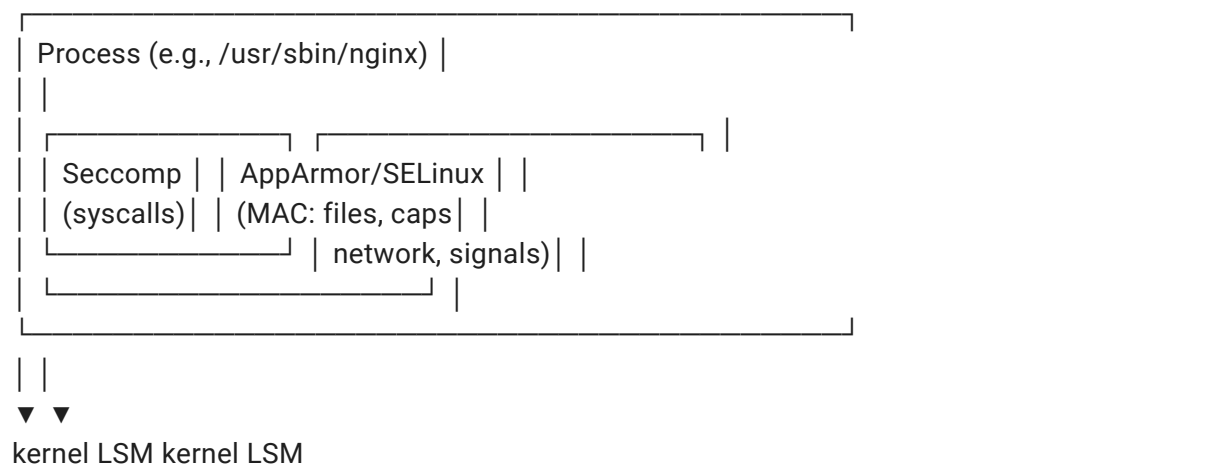
THE CONFIGURATION SIGNATURE IS NOT NECESSARY
no need the signature for config to avoid delivery 2 private keys
#UBOOT_SIGN_KEYNAME = "fit_signing_key"
UBOOT_SIGN_IMG_KEYNAME = "fit_signing_key"

FIT_HASH_ALG = "sha256"
FIT_SIGN_ALG = "rsa4096"
FIT_SIGN_INDIVIDUAL = "1"
=====

USERLAND

Can add, example this graph to highlight the relation between diffrent mecanism

Relation between Seccomp - SELinux - AppArmor



Each One's Role

Mechanism	What it controls	Granularity	Philosophy

Seccomp	Syscall access	Per-process (BPF filters)	"Only allow these specific syscalls"
AppArmor	Files, caps, network, signals, mount	Per-path profile	"Path-based confinement — simple to write"
SELinux	Same as AppArmor + labels	Per-label policy	"Label-based — more flexible but harder"

Layered Defense (All Three Together)

For a hardened process, you'd typically combine:

1. Seccomp — narrow syscall surface (e.g., only allow read, write, epoll_wait, mmap)
2. AppArmor or SELinux — restrict file access, capabilities, network
3. Linux capabilities — drop all unnecessary privs (e.g., cap_setpcap=0 for nginx worker)

Summary Table

	Seccomp	AppArmor	SELinux
Scope	Syscall only	Path-based objects	Label-based objects
Kernel hook	seccomp_run_filters() (syscall entry)	LSM hooks per-object type	Same LSM hooks as AppArmor, different policy engine
Granularity	Per-process BPF filter	Per-binary/profile	Per-label/context
Write complexity	Hard (BPF instructions)	Medium (path rules)	Hard (label + policy language)

Bypass risk	Can be bypassed via prctl() without	Breaks if binary moved outside	Survives file moves, but label mapping is
	seccomp	profile scope	complex
Typical use	Container isolation	Ubuntu/Debian hardening	RHEL/Fedora enterprise security
case			

Thanks for the great course.

Further training needs?

1 response

Embedded Linux Driver and refresher on Yocto and Embedded Linux in general