A Common Infrastructure for PKCS#11-Based Code Signing

OpenEmbedded Workshop 2020 @ Brussels

Jan Lübbe – jlu@pengutronix.de

https://github.com/jluebbe/meta-code-signing

About Me

- embedded software engineer
- co-author of FOSS tools RAUC (updating) and labgrid (testing)
- at Pengutronix since 2012

- Embedded Linux consulting & support since 2001
- > 5000 patches in Linux kernel

Why Sign Code?

- ensure that deployed systems run the expected software
- recover from run-time compromise
- soon: regulatory requirements

common use-cases

- during development: test signing and verification
- before release: authenticate released software

Existing Support in OE

- uboot-sign.bbclass in oe-core
- bundle.bbclass in meta-rauc
- meta-secure-core

secret keys in local files
⇒ key compromise difficult to detect
⇒ different roles not well supported

PKCS#11?

the standard C API to access keys on “tokens” (smart cards, YubiKeys, TPMs, TEEs, HSMs and SoftHSM)

- keys (“objects”) identified using URIs (RFC 7512)

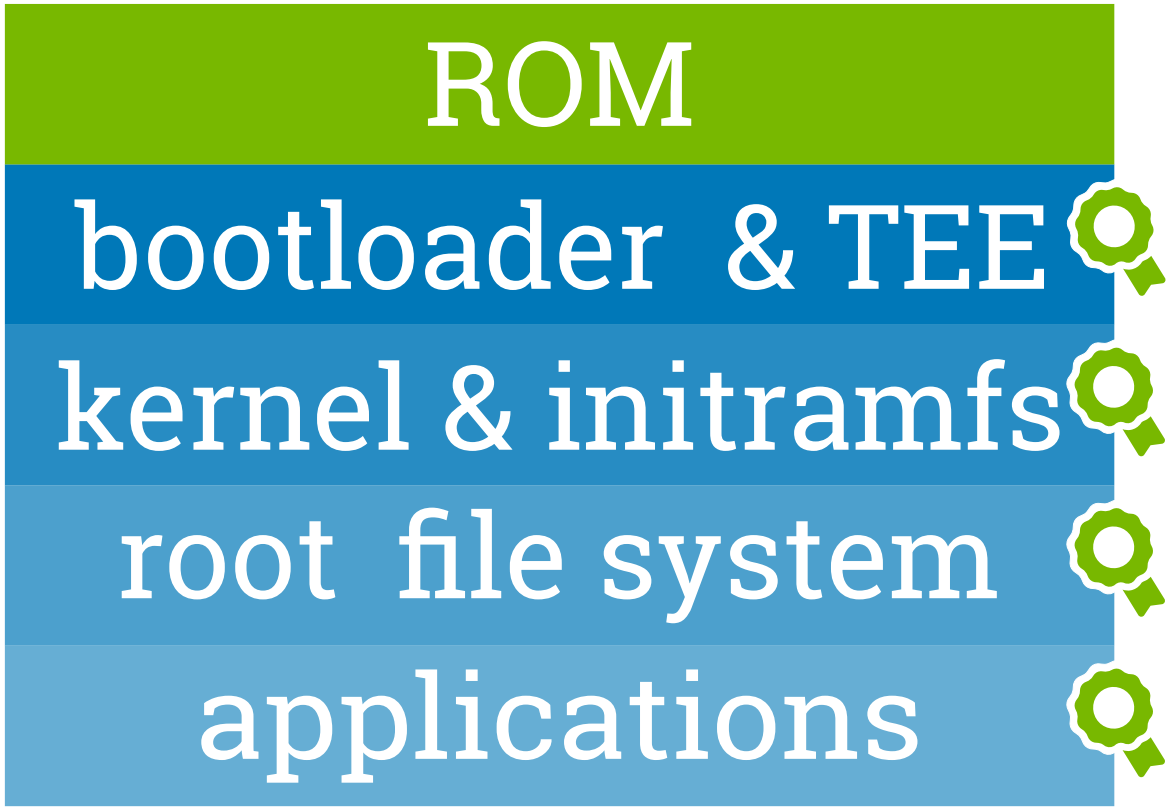
```
pkcs11:token=Crypto%20Token;manufacturer=CyberCorp;
       object=my-certificate;type=private;id=%69%95;?pin-source=file:/etc/token_pin
```

- many different implementations (open & closed)
- widely supported (OpenSSL, GnuTLS, OpenVPN, …)
- usage only, no access to key material

Why PKCS#11 in OE?

- support hardware tokens
  ⇒ better security
- decouple key configuration from key usage
  ⇒ simplifies reuse
  ⇒ consistent configuration for many recipes
- use same code-paths for development and release
  ⇒ better testing

Authenticated Components



Implementation

- https://github.com/jluebbe/meta-code-signing
- Infrastructure: signing.bbclass
- Users:
  - barebox
  - linux (fit-images)
  - rauc public key and bundles

Implementation – classes/signing.bbclass

- Infrastructure for key providers and users
- Keys stored and accessed by “role”
- local.conf can override with other PKCS#11 URIs
- Uses SoftHSMv2 internally for development keys

Implementation – providers

- Define a role and create a native package containing SoftHSM DB
- Provide virtual/<role>-signing package

Implementation – providers

```
inherit native signing
PROVIDES = "virtual/fit-signing …"
do_compile() {
    signing_import_prepare
    signing_import_define_role fit
    signing_import_cert_from_pem fit "${S}/fit-4096-development.crt"
    signing_import_key_from_pem fit "${S}/fit-4096-development.key"
    signing_import_finish
}
do_install() {
    signing_import_install
}
```

Implementation – users

- Examples for barebox, FIT and rauc
- Selects a role and uses generated `$PKCS11_URI` env var
- For extracting certificates and creating signatures
- Always “`inherit signing`”

Implementation – users – verification (rauc)

```
inherit signing
…
DEPENDS += "extract-cert-native virtual/rauc-signing"

do_configure_prepend() {
    signing_prepare
    signing_use_role rauc
    extract-cert "$PKCS11_URI" ${WORKDIR}/${RAUC_KEYRING_FILE}.der
    openssl x509 -inform der -in ${WORKDIR}/${RAUC_KEYRING_FILE}.der \
        -out ${WORKDIR}/${RAUC_KEYRING_FILE}
}
```

Implementation – users – signing (rauc)

```
inherit bundle signing
…
DEPENDS += "virtual/rauc-signing"

RAUC_KEY_FILE = "$(signing_get_uri 'rauc')"
RAUC_CERT_FILE = "$(signing_get_uri 'rauc')"

do_bundle_prepend() {
    signing_prepare
    signing_use_role rauc
}
```

Implementation – users – signing (FIT)

```
inherit signing
…
DEPENDS += "virtual/fit-signing"
…
do_compile_append() {
    signing_prepare
    signing_use_role fit
…
    uboot-mkimage -N pkcs11 -k "${PKCS11_URI#pkcs11:}" -r \
        -f ${B}/boot_fit_image.its ${B}/${MACHINE}.bin
}
```

Discussion

- further use-cases?
- simplification options?
- separate layer? meta-oe?

Thanks!

Questions?
Comments?