Using Yocto to Secure Your Device: From Development to Production

Kevin Chau, Timesys Corporation

Yocto Project Summit, 2021.11

Overview: Security in Stages

- **Early Development Analysis**
  - Threat Modelling
- **Pre-production**
  - Integrating Security
  - Supporting Infrastructure
- **Production and Ongoing Support**

Overview: Security in Stages

- Early Development Analysis
  - Threat Modelling
- Pre-production
  - Integrating Security
  - Supporting Infrastructure
- Production and Ongoing Support

Threat Modelling

Lots of different modelling methods available:

- STRIDE
- DREAD
- OWASP
- PASTA

Scope is important, what is critical for protection?

STRIDE

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

STRIDE: A Simplified Example

- Spoofing
- Tampering
- ~~Repudiation~~
- ~~Information Disclosure~~
- Denial of Service
- ~~Elevation of Privilege~~

Spoofing/Tampering: Concerns

- **What are we concerned about in the Yocto software ecosystem?**
  - Authenticity - We pull from online repositories; want to make sure that we don’t pull from bad actors
  - Repeatability - At some point, we want whatever we pull from upstream to always be the same; AUTOREV is a particular concern

Denial of Service: Concerns

- If upstream servers go down, we don’t want that to prevent us from building an image for a release
  - More malicious: DDoS attack against your organization or one of the key software sources necessary for your build

How do Offline Builds Help?

- **Offline Builds - Build using a local set of files, no network access required**
  - Requires us to do at least one build with network access to pull down all of the required sources
  - Keep that set of files and use that as our “master copy” of the required sources
  - Allows us to be certain that any future builds will all start with the same sources

Offline Builds: Creating Your Source Archive

Fetching the sources

Setting your configuration to no network

Run your build as normal

You do need external network access to download all of the sources necessary for your project. These will be collected in your downloads folder, DL_DIR.

BB_GENERATE_MIRROR_TARBALLS = "1"
bitbake *harden-image-minimal* --runonly=fetch

SOURCE_MIRROR_URL ?= "file:///home/your-download-dir/"
INHERIT += "own-mirrors"
BB_NO_NETWORK = "1"

You can remove BB_GENERATE_MIRROR_TARBALLS = “1”.

bitbake <image>

Offline Builds: Dealing with AUTOREV

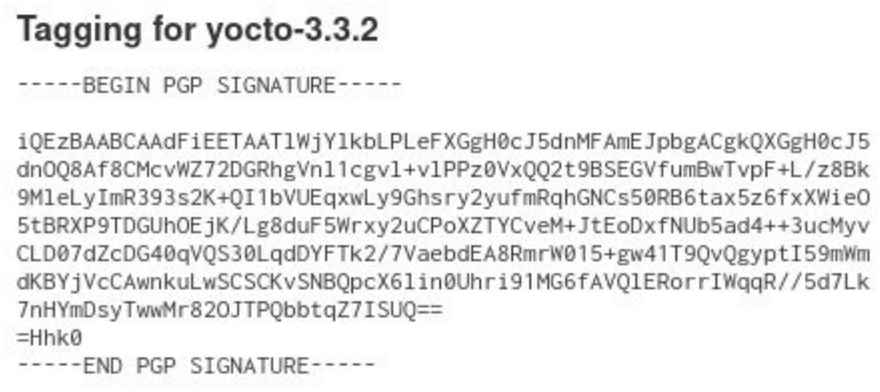
- **Don’t use it!**
  - If you do, you’ll find issues when building offline

Offline Builds: Your Obligations

- **The previous benefits come at the expense of putting the onus of traceability on you**
  - You have to keep this directory full of sources somewhere and ensure its integrity
  - Depending on your security requirements, may need to independently audit the downloaded source

Offline Builds: Validating Your Sources

Yocto has PGP signed tags


Tagging for yocto-3.3.2

```
-----BEGIN PGP SIGNATURE-----

iQEzBAABCAAdFiEETAATlWjYlkbLPLeFXGgH0cJ5dnMFAmEJpbgACgkQXGgH0cJ5
dnOQ8Af8CMcvWZ72DGRhgVnl1cgvl+vlPPz0VxQQ2t9BSEGVfumBwTvpF+L/z8Bk
9MleLyImR393s2K+QI1bVUEqxwLy9Ghsry2yufmRqhGNCs50RB6tax5z6fxXWieO
5tBRXP9TDGUhOEjK/Lg8duF5Wrxy2uCPoXZTYCveM+JtEoDxfNUb5ad4++3ucMyv
CLD07dZcDG40qVQS30LqdDYFTk2/7VaebdEA8RmrW015+gw41T9QvQgyptI59mWm
dKBYjVcCAwnkuLwSCSCKvSNBQpcX6lin0Uhri91MG6fAVQlERorrIWqqR//5d7Lk
7nHYmDsyTwwMr82OJTPQbbtqZ7ISUQ==
=Hhk0
-----END PGP SIGNATURE-----
```

```
[host poky]$ git verify-tag hardknott-3.3.4
gpg: Signature made Thu 18 Nov 2021 05:00:03
PM EST
gpg:                using RSA key
4C00139568D89646CB3CB7855C6807D1C2797673
gpg: Can't check signature: No public key
[host poky]$ gpg2 --search-keys
0x4C00139568D89646CB3CB7855C6807D1C2797673
gpg: data source: https://162.213.33.9:443
(1)    Yocto Build and Release
<releases@yoctoproject.org>
      4096 bit RSA key 87EB3D32FB631AD9,
created: 2014-10-30
Keys 1-1 of 1 for
"0x4C00139568D89646CB3CB7855C6807D1C2797673".
Enter number(s), N)ext, or Q)uit >
```

Offline Builds: Validating Your Sources

PGP signatures or checksums for your software sources

- Linux Kernel
- Mesa

Offline Builds: Validating Your Sources

Yocto can enforce checksums:

- May be helpful if your project uses internally released software

Overview: Security in Stages

- Early Development Analysis
  - Threat Modelling
- Pre-production
  - Integrating Security
  - Supporting Infrastructure
- Production and Ongoing Support

Security Features: Image Contents

- **Yocto recipes are customizable, can add security oriented features easily with meta-layers and bbappends**
  - meta-security
- **Features added by meta-security**
  - dm-verity
  - IMA/EVM
  - Kernel Module Signing

Setting Up a Basic Build with meta-security

```
host:~/example/$ git clone git://git.yoctoproject.org/poky
host:~/example/$ git clone https://git.openembedded.org/meta-openembedded
host:~/example/$ . poky/oe-init-build-env
host:~/example/build$ bitbake-layers add-layer ../meta-openembedded/meta-oe
NOTE: Starting bitbake server...
host:~/example/build$ bitbake-layers add-layer ../meta-openembedded/meta-python
NOTE: Starting bitbake server...
host:~/example/build$ bitbake-layers add-layer
../meta-openembedded/meta-networking
NOTE: Starting bitbake server...
host:~/example/build$ bitbake-layers add-layer ../meta-openembedded/meta-perl
NOTE: Starting bitbake server...
host:~/example/build$ bitbake-layers add-layer ../meta-security
```

Setting Up a Basic Build with meta-security

Meta-security provides a “harden-image-minimal” image with basic security changes to “core-image-minimal”

```
host:~/example/build$ bitbake-layers add-layer ../meta-security/meta-hardening
NOTE: Starting bitbake server...
host:~/example/build$ echo "DISTRO_FEATURES += \" security \"" >> conf/local.conf
host:~/example/build$ bitbake harden-image-minimal
host:~/example/build$ ls tmp/deploy/images/qemux86-64/
bzImage
harden-image-minimal-qemux86-64.ext4
```

Key Management / Signing Infrastructure

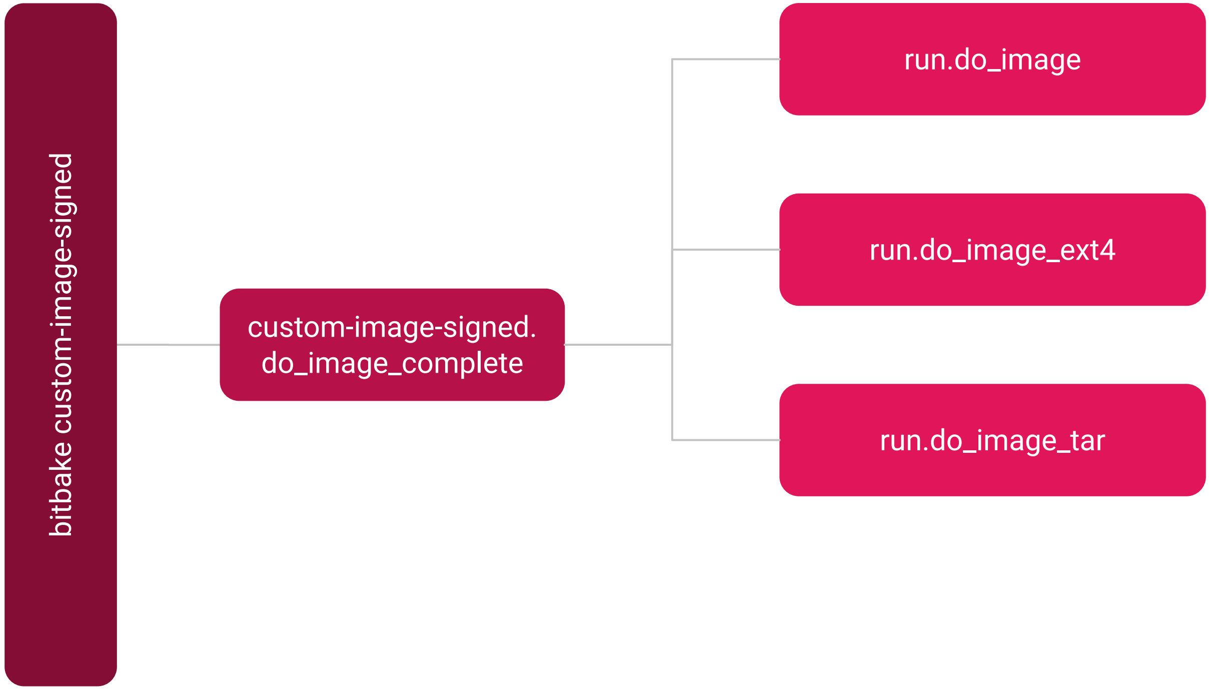
- **We’ve generated the system image and root file system, now how do we provide authenticity?**
  - Public Key Cryptography
- **How do we keep the private key private?**
  - Secure Build Machine
  - Signing Server

Implementing the “Secure Build Machine” Method

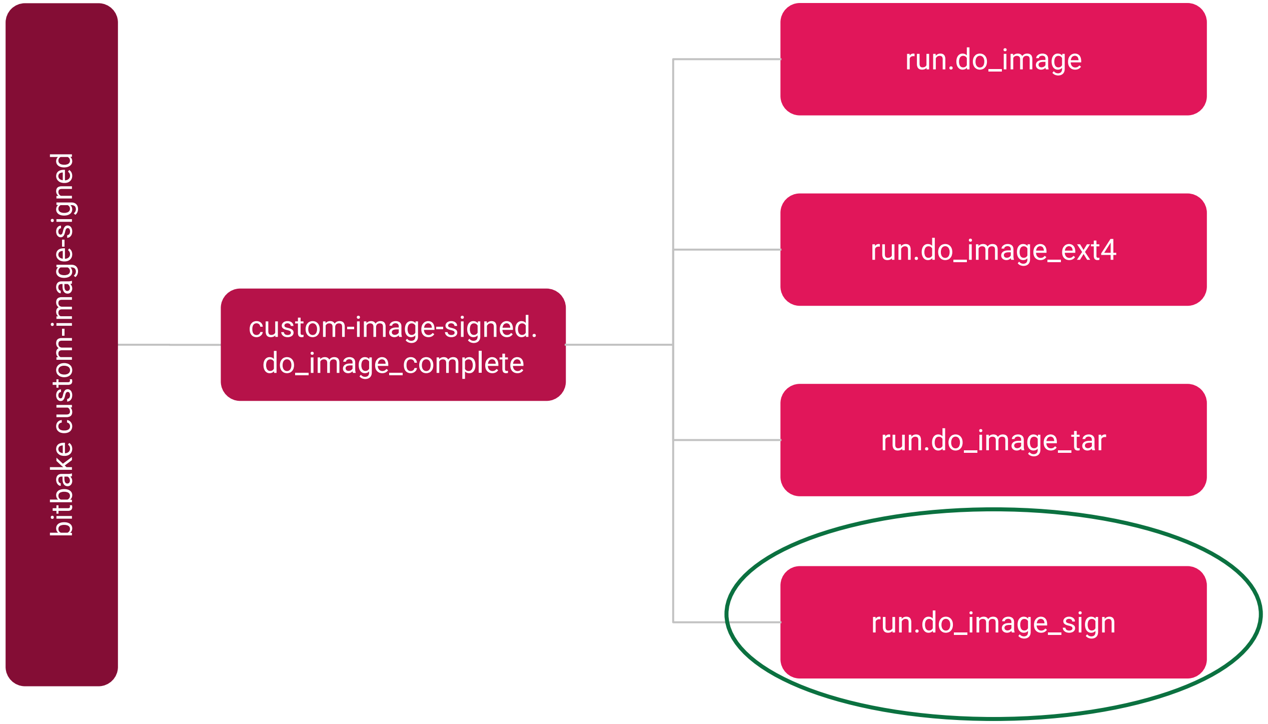
Let’s assume that we have our secure build machine

We want to create our own meta-layer that will integrate the signing steps into our Yocto build so we only have to run one command to get our desired output files

Creating a Custom Image with Signing Tasks



Creating a Custom Image with Signing Tasks



Create Our Custom Output Image Recipe

```
require recipes-core/images/harden-image-minimal.bb

python __anonymous () {
     bb.build.addtask('do_image_sign', 'do_image_complete', 'do_image_ext4', d)
}

PRIVATE_KEY = "${TOPDIR}/private.pem"

do_image_sign() {
     cd ${WORKDIR}/
     openssl dgst -sha256 deploy-${PN}-image-complete/${PN}-${MACHINE}.ext4 > hash
     openssl rsautl -sign -inkey ${PRIVATE_KEY} -keyform PEM -in hash >
deploy-${PN}-image-complete/${PN}-${MACHINE}.ext4.sig
}
```

Create Our Custom Output Image Recipe

```
bitbake custom-image-signed
wc -c tmp/deploy/images/qemux86-64/custom-image-signed-qemux86-64.ext4.sig
 256 tmp/deploy/images/qemux86-64/custom-image-signed-qemux86-64.ext4.sig
```

Create Our Custom Output Image Recipe

```
cat tmp/work/qemux86_64-poky-linux/custom-image-signed/1.0-r0/temp/log.task_order
do_prepare_recipe_sysroot (3787003): log.do_prepare_recipe_sysroot.3787003
do_rootfs (3787019): log.do_rootfs.3787019
do_flush_pseudodb (3793153): log.do_flush_pseudodb.3793153
do_write_qemuboot_conf (3793154): log.do_write_qemuboot_conf.3793154
do_image_qa (3793159): log.do_image_qa.3793159
do_image (3793166): log.do_image.3793166
do_image_ext4 (3793173): log.do_image_ext4.3793173
do_image_tar (3793174): log.do_image_tar.3793174
do_image_sign (3793214): log.do_image_sign.3793214
do_image_complete (3793217): log.do_image_complete.3793217
do_populate_lic_deploy (3793231): log.do_populate_lic_deploy.3793231
do_image_sign (3793891): log.do_image_sign.3793891
do_image_complete (3793906): log.do_image_complete.3793906
do_populate_lic_deploy (3793918): log.do_populate_lic_deploy.3793918
```

Overview: Security in Stages

- Early Development Analysis
  - Threat Modelling
- **Pre-production**
  - Integrating Security
  - **Supporting Infrastructure**
- Production and Ongoing Support

Deploying Image Files: Secure Build Machine

- **We’ve generated the signature file for our rootfs and our kernel; now we need some way for our developers to get the files**
  - Integrate with an external CI platform
  - SFTP server
  - SSH/SCP

Deploying Image Files: Signing Server

Yocto Build

In this workflow, we generate the relevant kernel image and RFS with a typical bitbake command.

```
bitbake <image>
```

Sending the Files to the Server

We would send it to our server in the manner expected, along with our developer credentials.

Deployment to Device or to Developers

Push to an automated test server and/or to a location developers can access.

Overview: Security in Stages

- Early Development Analysis
  - Threat Modelling
- Pre-production
  - Integrating Security
  - Supporting Infrastructure
- **Production and Ongoing Support**

Support: Choosing an LTS Release

Yocto has special LTS releases; use them if possible:

- Dunfell
- Kirkstone

https://wiki.yoctoproject.org/wiki/Releases

Over-the-air Firmware Updates, New Releases

Different strategies for new firmware images based on other security choices:

- **File-based Authentication: fs-verity, IMA/EVM**
  - meta-swupdate
  - meta-rauc
  - meta-updater
- **Block-based Authentication**
  - A/B schemes

A/B Schemes

- **The target system must validate the binary files against their signatures:**
  - (Kernel) Image.sig -> Image
  - (RFS) RFS.sig -> custom-image-signed.ext4
- **We can package these together into an archive, which is our update bundle**

CVE Management

- **Maintaining support means tracking and addressing vulnerabilities**
  - Yocto project maintains its own CVE Checker
  - Timesys provides one as well, meta-timesys

References

For further reading:

- https://www.timesys.com/pdf/Timesys-Security-Primer-for-IoT-Embedded-Devices.pdf
- https://insights.sei.cmu.edu/blog/threat-modeling-12-available-methods/
- https://owasp.org/www-community/Threat_Modeling
- https://www.yoctoproject.org/docs/current/mega-manual/mega-manual.html
- https://www.yoctoproject.org/docs/current/brief-yoctoprojectqs/brief-yoctoprojectqs.html
- https://elinux.org/images/3/31/Comparison_of_Linux_Software_Update_Technologies.pdf
- Designing OSTree based embedded Linux systems with the Yocto Project

Timesys Security Survey

https://docs.google.com/forms/d/e/1FAIpQLSf4LlAZ0rhEvrRcSBATs36FJx9Daop1q5w50-4PLIZ6nwloGQ/viewform yocto PROJECT | THE LINUX FOUNDATION