Security Hardening with OpenEmbedded / Yocto Project

Scott Murray, Konsulko Group

Yocto Project DevDey *Virtual*, North America, 2020

Agenda

- Security Hardening?
- Basic hardening with OE/YP
- meta-security
- meta-selinux
- Updater layers
- meta-sca

Caveats

- I do not consider myself a security expert
- This presentation was spurred by an interest in seeing what is available in the OE/YP ecosystem, so it is high-level and is not exhaustive
- Your security requirements will be dependent on product requirements and usecases, intent is to showcase some of the available tools/options

Security Hardening?

- Securing a system by reducing its attack surface
- Remove unnecessary software/services, users
- Control network access, e.g. firewall
- Intrusion detection
- Remove/improve default passwords/users
- Updates to remove vulnerabilities
- etc.

Why?

- Everything is becoming Internet connected
  - Internet of Things (IoTs)
- Attackers are becoming more aware of Linux devices
  - Scans of all of IPv4 are a thing, e.g. shodan.io
  - Customers cannot be relied upon to not attach devices directly to the Internet
  - uPnP may make device services visible unexpectedly
- Attacks may not be direct
  - Using device (mis)behavior as part of a DDoS attack

Why? (continued)

- Some presentations this week with more detailed discussion
- "Introduction to Embedded Linux Security" - Sergio Prado, Embedded Labworks
  - https://ossna2020.sched.com/event/c3XR/introduction-to-embedded-linux-security-sergio-prado-embedded-labworks#
- "IoT Developer's Guide to Building Secure IoT Devices" - Yogesh Ojha, Tata Consultancy Services
  - https://ossna2020.sched.com/event/c3TI/iot-developers-guide-to-building-secure-iot-devices-yogesh-ojha-tata-consultancy-services#

OWASP IoT Top Ten Vulnerabilities

- Open Web Application Security Project® (owasp.org)
- Internet of Things working group Top Ten vulnerabilities have been surveyed and published every few years
- Top Ten 2018 at:
  - https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf
- Also do a web application Top Ten that may be useful for evaluation of web-based interfaces

OWASP IoT Top Ten Vulnerabilities (2018)

1. Weak, Guessable, or Hardcoded Passwords
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Use of Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage
8. Lack of Device Management
9. Insecure Default Settings
10. Lack of Physical Hardening

OE/YP Hardening?

- Read the Fine Manual
  - https://www.yoctoproject.org/docs/3.1.1/dev-manual/dev-manual.html#making-images-more-secure
- Provides some useful high-level guidelines
- Has some more detailed guidance around disabling debug features, adding users and passwords, and security related compile flags
- Mentions meta-security and meta-selinux
- Useful, but mostly a starting point

Expanding on the FM

- Check image manifest(s) for surprises
- oe-pkgdata-util useful for finding what package files come from
- Check/Prune DISTRO_FEATURES
  - If using or basing off of poky, note it includes a lot of things you may not want (e.g. NFS)
- Note that poky includes "debug-tweaks" in IMAGE_FEATURES by default
  - No root password useful for early testing, but should be removed or explicitly added only to debug/dev builds

Expanding on the FM (2)

- Review kernel configuration
  - Security options, but also things like hardware RNG, architecture specific address space randomization
  - Some more ideas in https://www.whonix.org/wiki/Hardened-kernel
- Make sure CONFIG_DEVMEM is disabled if at all possible
  - Typically used to access device registers as a workaround
  - Somewhat better now with default values of STRICT_DEVMEM and IO_STRICT_DEVMEM, but using/fixing drivers and disabling is safer

Expanding on the FM (3)

- It's common for BSP layers to not enable desired features…
  - e.g. cgroup, namespace, netfilter, BPF support
  - These become more visible when using systemd or container runtimes
- …and to enable a lot of things you do not need
  - Usually err on the side of enabling a lot of driver subsystems and drivers
  - May enable DEBUG options that are problematic

Expanding on the FM (4)

- User and password management beyond the probably undesirable baking in of fixed root/admin passwords is going to take local development
  - Tooling/examples for schemes like generating device-specific passwords would probably be helpful (pointers welcome!)
- passwdqc library and PAM module for password strength checking in meta-oe may be useful for vetting user provided passwords

Expanding on the FM (5)

- ROOTFS_READ_ONLY feature is worth considering
  - Increase difficulty for attackers
  - Seconday benefit of also being useful for implementing reset to factory default schemes
- May require development effort
  - Locally developed applications, or packages from outside oe-core may not work out of the box
  - Combining with MAC schemes such as SELinux will require some work (as labelling is done on boot)

Expanding on the FM (6)

- cve-check class can be used to check packages or images for known CVEs
- See meta/classes/cve-check.bbclass
- Uses NVD CVE database, results are data dependent and may not be complete
- You will likely need to process the output if using it as input for your own maintenance or LTS
  - There are CVEs for some packages that are not whitelisted that need to be handled
- SRTool (https://wiki.yoctoproject.org/wiki/SRTool_User_Page) may be useful if you need to set up an issue tracker

meta-security

- Bit of a Swiss Army knife or toolbox layer
- Maintained by Armin Kuster
- Recipes for packages related to:
  - Support Libraries
  - Security compliance
  - Secure boot
  - Integrity/Attestation
  - Intrusion detection
  - Runtime security scanners
  - Mandatory Access Control (MAC)
- docs/overview.txt describes some packages

meta-security – Support Libraries

- libseccomp (https://github.com/seccomp/libseccomp)
  - Provides access to the kernel's syscall filtering mechanism
  - Highly recommended for enabling better sandboxing with systemd and container runtimes
  - Need to add "seccomp" to PACKAGECONFIG for e.g. systemd, runc, etc.
- google-authenticator-libpam
  - PAM module for MFA with Google authenticator
- libdhash, libmhash, libmspack
  - Potentially useful hashing and compression libraries

meta-security – Compliance

- Recipes in meta-security-compliance layer
- Lynis (https://cisofy.com/lynis) runtime system auditor
- OpenSCAP (https://www.open-scap.org)
  - Implementation of Security Content Automation Protocol
  - In simple terms, a specification of standardized naming for interaction with tools and databases
  - oscap and oscap-daemon tools for checking NIST or other databases for vulnerabilities
- These seem likely to be overkill in a lot of embedded usecases
  - But perhaps still useful in a QA role

meta-security – Secure Boot/Integrity

- Trusted Platform Module (TPM) recipes in meta-tpm
  - https://en.wikipedia.org/wiki/Trusted_Platform_Module
- TPM 1.x and TPM 2.0 tools
- Kernel configuration for linux-yocto driven by "tpm" and "tpm2" MACHINE_FEATURES
- Sample images that use TPM or TPM2
- Provides a starting point
  - Using for runtime integrity checking, key storage, etc., will require custom development

meta-security – Secure Boot/Integrity (2)

- Note that support for secure boot on ARM SoCs is typically vendor specific and is hopefully available via the vendor BSP layer
- The commonly used trusted firmware (https://www.trustedfirmware.org) component (TF-A) has tended to have recipes for forked versions in vendor BSP layers, but rationalization on a recipe in the new meta-arm layer is in progress
- Setting up things like key storage and image encryption will typically take custom integration

meta-security – Secure Boot/Integrity (3)

- Integrity Measurement Architecture (IMA) and Extended Verification Module (EVM) recipes in meta-integrity
  - https://sourceforge.net/p/linux-ima/wiki/Home/
  - Extends secure ("measured") boot up into userspace
  - Appraisal support for doing runtime remote attestation
  - Can be unwieldy to implement in practice
- Tool and sample image recipes
- See meta-integrity layer in https://github.com/jiazhang0/meta-secure-core for an alternate implemententation

meta-security – Secure Boot/Integrity (4)

- Support for dm-verity recently added
- Integrity measurement at block device block level
  - Simpler to implement than the file-oriented approach of IMA
- Originally developed for Android
  - https://source.android.com/security/verifiedboot/dm-verity
- Class for generating image with hash information
- Sample configuration for building and testing on BeagleBone Black
- Integration with platform secure boot mechanism requires development

meta-security – Intrusion Detection

- Samhain
  - http://www.la-samhna.de/samhain
  - Highly configurable filesystem scanning, rootkit detection, etc.
- Suricata
  - https://suricata-ids.org
  - Network intrustion detection via traffic inspection
- Tripwire
  - https://github.com/Tripwire/tripwire-open-source
  - Filesystem scanning
  - Widely used due to long history (created in 2000)

meta-security – Runtime Scanners

- Collection of scanners that are more configuration checking than intrusion detectors
- buck-security
  - Collection of configuration and filesystem checks
  - Project seems dead since 2013, Lynis is likely a better choice
- checksec, checksecurity
  - Simple configuration checkers, potentially more useful for QA than production use
- chkrootkit
  - Root kit detector, releases are somewhat sporadic so potential benefit would need to be evaluated

meta-security – Runtime Scanners (2)

- Bastille (https://sourceforge.net/projects/bastille-linux)
  - Hardening and reporting/auditing tool
  - Support is only for an informational reporting mode as opposed to the further ability to e.g. disable services on other distributions
  - Upstream development seems to have stopped in 2016, some evaluation would be required as to current usefulness…
  - Similarly to Lynis or OpenSCAP, some consideration required as to usefulness in a production image

meta-security – MAC

- Recipes for AppArmor, SMACK, and Tomoyo MAC systems
- SELinux support is in separate meta-selinux layer
- Application profiles for AppArmor in the default install are somewhat limited
  - Ubuntu or Debian may serve as a resource for other profiles
- Similarly, the default SMACK policies are probably insufficient and development will be required
  - SMACK policy development is simpler than SELinux, but the userbase is small at this point, so support may be hard to find
- Due to the larger userbases and active development SELinux or AppArmor are likely better choices for a new project

meta-selinux

- Recipes for SELinux MAC support
  - Tools, packagegroups, sample minimal images
- Maintained by WindRiver and Siemens developers
- Reference policy recipes for several types of policy setup (e.g. minimal, targeted, full multi-level)
- Note that ATM in dunfell and master the layer defaults do not produce a booting image with enforcing enabled
  - May take a while, but intent is to get this fixed
- SELinux policy development and maintenance is involved...
  - ...and you will likely need to do some policy development, as the reference policy is unlikely to cover everything you want to use

Why consider SELinux?

- Typically considered too much effort for traditional embedded usecases, outside of commercially supported distros
- AppArmor and SMACK are considered easier to configure and use in a targeted fashion
- But...
  - Has become relied upon to improve container security
  - Docker/runc CVE-2019-5736 container escape blocked by SELinux
  - Long time usage by RHEL/Centos/Fedora make support perhaps the best of the MAC systems

Updater layers

- There are several actively maintained updater tools with layers
  - Will point out swupdate, Mender, RAUC, Aktualizer
  - There are others, e.g. meta-swupd, that have smaller userbases
- Rolling your own mechanism is possible with e.g. OSTree recipe
  - But the ones mentioned all have support for already existing server mechanisms, and some potential for turnkey hosting with a provider

Updater layers – meta-swupdate

- https://github.com/sbabic/meta-swupdate
- Integrates swupdate support
  - https://github.com/sbabic/swupdate
- Documentation at http://sbabic.github.io/swupdate/swupdate.html
- Some discussion this week in "Secure Boot and Over-the-Air Updates - That's Simple, No?" - Jan Kiszka, Siemens AG
  - https://ossna2020.sched.com/event/c3Wx/secure-boot-and-over-the-air-updates-thats-simple-no-jan-kiszka-siemens-ag

Updater layers – meta-mender

- https://github.com/mendersoftware/meta-mender
- Integrates Mender support
  - https://mender.io
- Documentation at https://docs.mender.io/artifacts/yocto-project/building

Updater layers – meta-rauc

- https://github.com/rauc/meta-rauc
- Integrates RAUC support
  - https://rauc.io
- Documentation at https://rauc.readthedocs.io/en/latest

Updater layers – meta-updater

- https://github.com/advancedtelematic/meta-updater
- Integrates OSTree update mechanism and aktualizer client support
  - https://github.com/ostreedev/ostree
  - https://github.com/advancedtelematic/aktualizr
- Documentation at:
  - https://ostree.readthedocs.io/en/latest
  - aktualizer github page (see above)

meta-sca

- https://github.com/priv-kweihmann/meta-sca
- Collection of static analysis tools maintained by Konrad Weihmann
- Static analysis for C, C++, python, etc.
- Classes to enable per package or per image scanning (some limits depending on specific tools)
- Significant documentation
- Actively maintained

Summary

- As mentioned at the start, a non-exhaustive survey
  - No discussion of network / firewall tools
  - Some things skipped in meta-security
- Let me know if I've missed something useful!
  - Or if a particular area warrants a focused follow up presentation
- Contact info:
  - scott.murray@konsulko.com
  - smurray on Freenode.net IRC (#oe, #yocto channels)