IoT Security

Advanced Internet Security

Adrian Dabrowski
Christian Kudera
Georg Merzdovnik
Aljosha Judmayer

News from the Lab

- Challenge 4 ends today
  - Fastest solve: “Octal Kansis” in 1 day, 3:32:52
- Challenge 5 starts tomorrow

News from the Field

- Hacker Infects Node.js Package to Steal from Bitcoin Wallets[1]
  - library called “event-stream” (work with streams), over two million downloads
  - original maintainer handed over maintenance several months ago
    - did not use the library himself anymore
    - got a mail which asked him about the maintenance
  - New maintainer released update, which included a dependency “flatmap-stream”
    - obfuscated code, encrypted payload
    - attack was undetected for over two months
  - The malicious code tried to steal bitcoins from “Copay wallets”

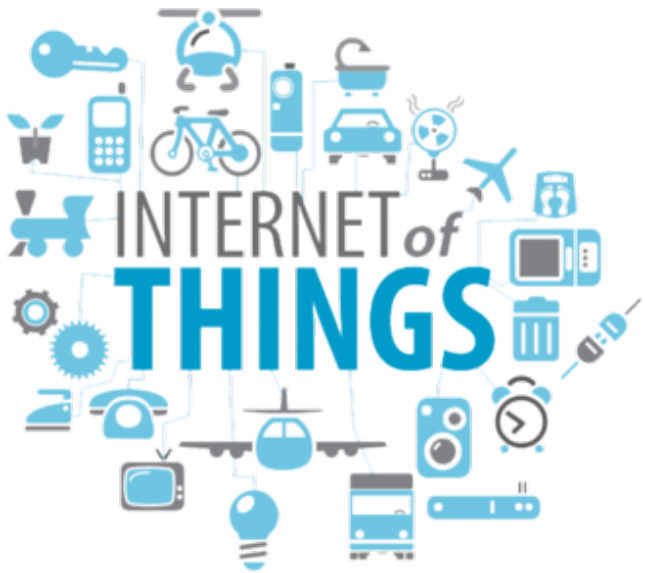
[1] Hacker Infects Node.js Package to Steal from Bitcoin Wallets

News from the Field (contd.)

- Marriott Data Leak (500 Million customers)[2]
  - Through Starwood guest reservation system
    - Acquired by Marriott in 2016
  - Timeline:
    - On September 8 alert from internal security tool about database access attempt
    - On November 19th the Security investigation concluded that there was unauthorized access
    - they were able to decrypt the content and identified that it came from the Starwood guest reservation database
  - During investigation, they learned the unauthorized access began in **2014**

[2] http://news.marriott.com/2018/11/marriott-announces-starwood-guest-reservation-database-security-incident/

Internet of Things Security Basics



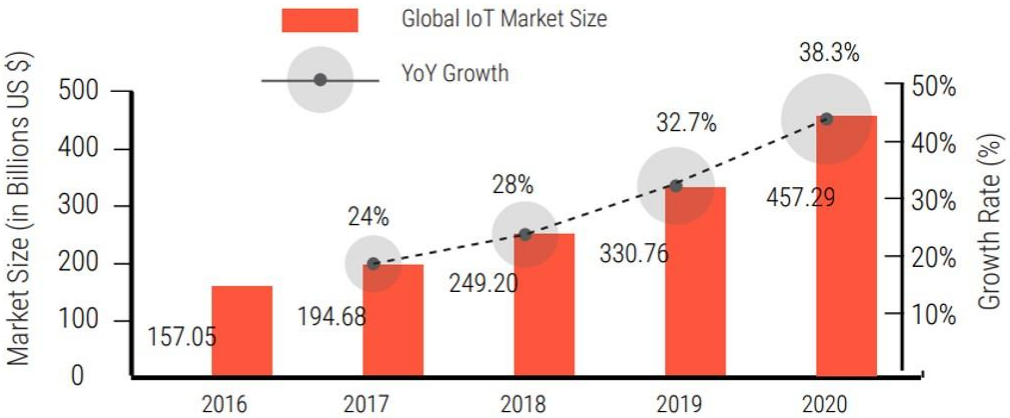

WHAT IF I TOLD YOU
THE 'S' IN IOT STANDS FOR SECURITY
imgflip.com

Internet of Things (IoT)

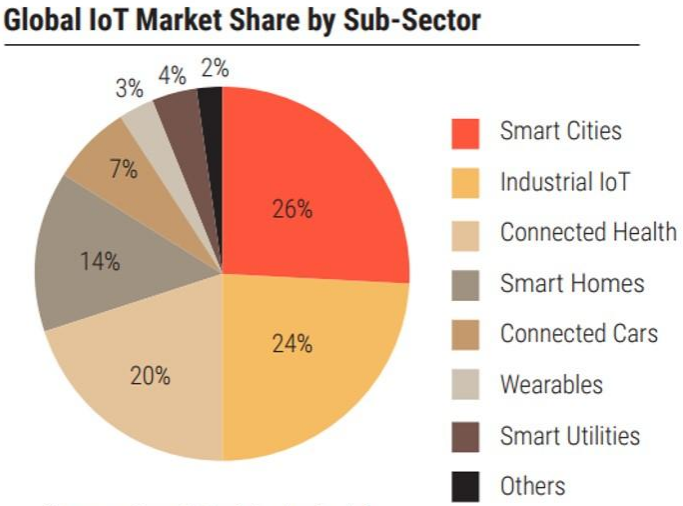
Wikipedia: The Internet of things (IoT) is the network of devices, vehicles, and home appliances that contain electronics, software, actuators, and connectivity which allows these things to connect, interact and exchange data.

- Basically everything connected to the Internet
  - High-end devices (PC, laptop, smartphone, tablet)
  - Low-end devices (embedded systems, sensors, actuators)

Forecast



[Sources: GrowthEnabler Analysis/MarketsandMarkets]



[Source: GrowthEnabler Analysis]

Why so many problems in the IoT

- Today, everything is getting connected
  - Time to market is important (first to market wins)
  - However, nobody has time (or knowledge) to implement security
- Providers lack incentives for maintenance, security patches
  - cheap devices (just make a new one)
  - have not been responsible for incidents
  - No consequences for breaches
    - some changes on the horizon (e.g. California)

IoT System Heterogeneity I

- Different Hardware Architectures
  - Other processors (than plain old x86)
    - e.g. ARM, AVR, MIPS, …
  - Peripheral device interactions
    - Sensors/Actuators
- Communication Heterogeneity
  - Wired communication (Ethernet, CAN, …)
    - Yes, cars are also part of the IoT now
  - Wide Area Networks
    - WiFi, 3G/4G/5G, LoraWAN
  - Low Range Connections
    - Bluetooth (LE) / Zigbee / 6LoWPAN

IoT System Heterogeneity II

- Different Protocols
  - Device2Device and Cloud communications
    - e.g. BLE
    - CoAP
    - MQTT
    - Websockets
    - …

IoT Security - What is all the fuzz about?

- Connected devices:
  - grow in numbers
  - control everything from homes to smart factories
    - and often rely on internet connectivity
  - collect sensitive information
- Easy to discover such devices on the internet
  - Shodan[3] *Search engine Internet-connected devices*
  - or scan yourself (zmap, masscan,…)

[3] https://www.shodan.io/

What could possibly go wrong?

Denial of Service attacks **against** devices

- DDoS Attack Takes Down Central Heating System Amidst Winter In Finland[4]
- According to the company website:
  ”Over 90 percent of the [remote systems] in the area of terraced houses or larger buildings will not send an alarm at the moment, even if the heat is switched off or radiator pressure disappears,” as the systems are designed to shut down for safety. ”The systems must be actively monitored and adjusted.”
- Solution was to place the system behind a firewall
- However, other systems also rely on constant connectivity

[4] https://thehackernews.com/2016/11/heating-system-hacked.html

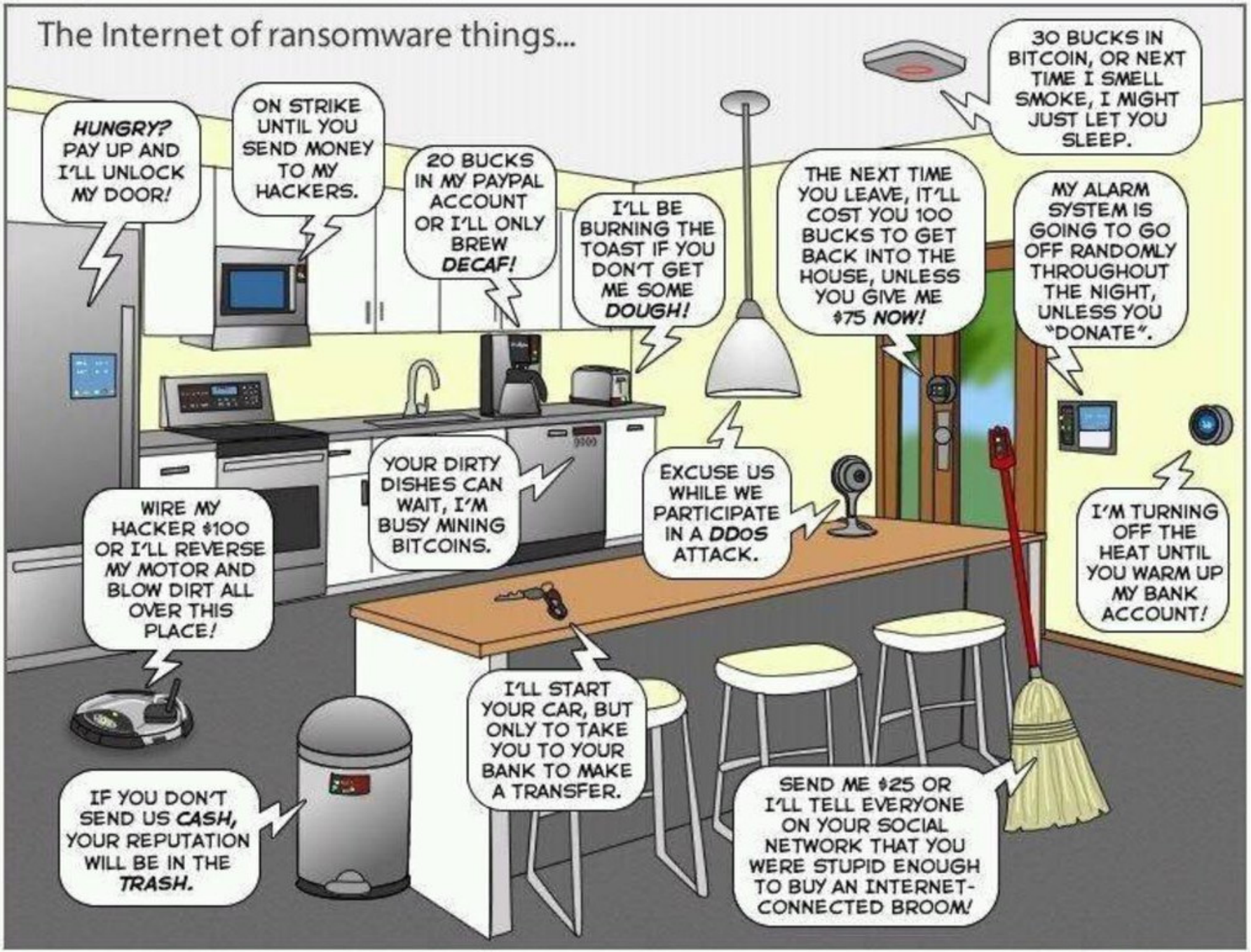
Denial of Service attacks **from** devices

- E.g. IoT Botnets used for large scale attacks
  - e.g. Mirai[5] which was used to launch major DDoS attacks
- These infect devices either through default credentials or known vulnerabilities

[5] https://en.wikipedia.org/wiki/Mirai_(malware)

Direct attacks against devices

- IoT Ransomware proof-of-concept demonstration
  - Exploited a Smart Thermostat, turned on heating and demanded 1 Bitcoin
  - performed local attack
- But could happen as well with remotely connected devices



Leak of sensitive information

- *Strava* offers an application for fitness tracking by means of smart phones' location
  - released a heat map showing the activities of its users worldwide

[6] http://www.wired.co.uk/article/strava-heat-maps-military-app-uk-warning-security

Leak of sensitive information

- *Strava* offers an application for fitness tracking by means of smart phones' location
  - released a heat map showing the activities of its users worldwide
- Eventually revealing:
  - not only individual habits,
  - but also structure of secret military bases[6].

[6] http://www.wired.co.uk/article/strava-heat-maps-military-app-uk-warning-security

Leak of sensitive information (contd.)

- Bluetooth devices were able to connect to the microphone of *My Friend Cayla*
  - interactive doll for children
  - enabling eavesdropping of its users.
- Additionally everything is recorded and sent to the companies servers
- Under German law the doll was even considered a surveillance device
  - parents were obliged to destroy *Cayla*[7].

[7] http://www.bbc.com/news/world-europe-39002142

IoT Security - Common Misconception

It’s only about the devices!

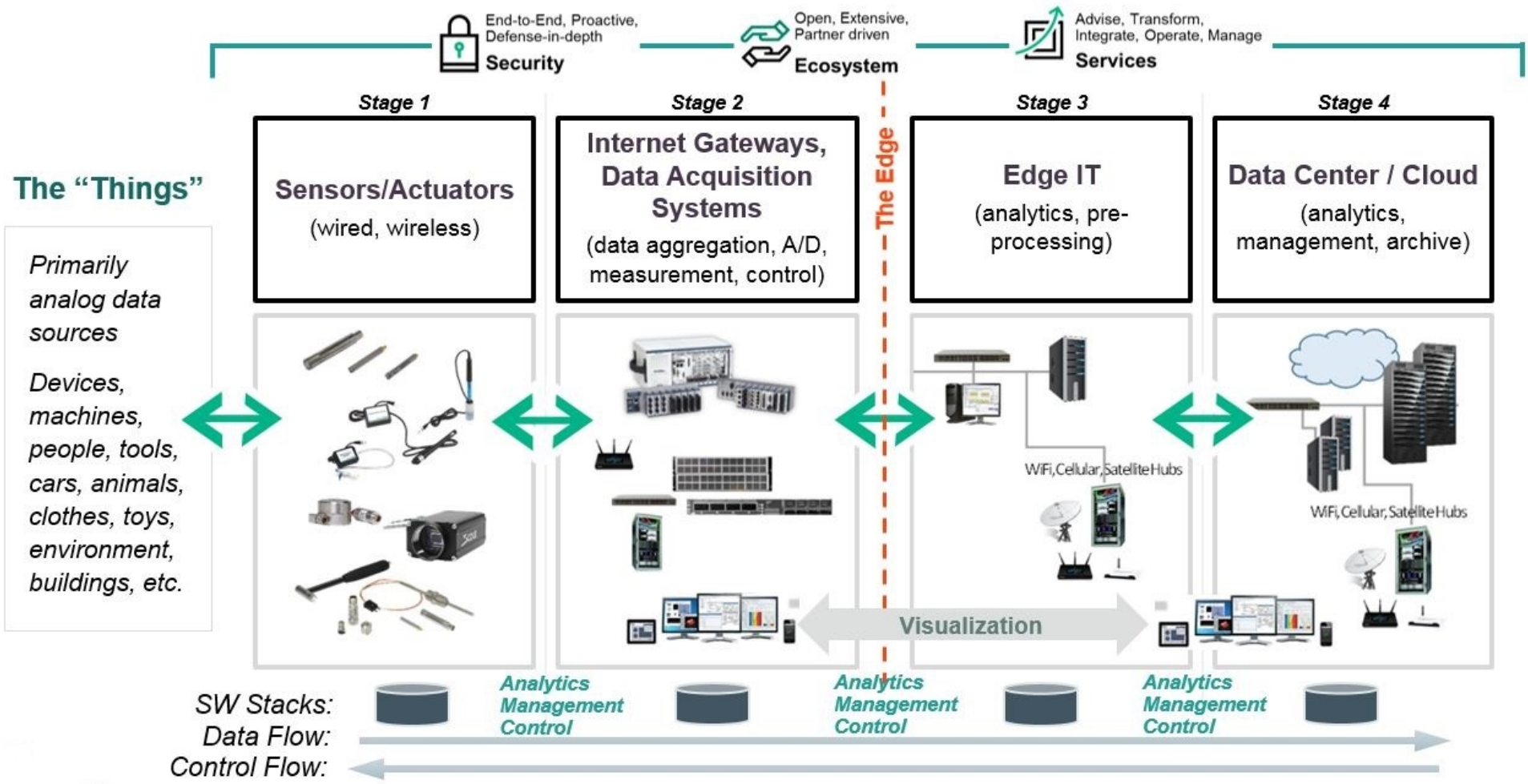
IoT Security - Common Misconception

It’s only about the devices!

IoT Security - Reality

- We also need to take care about
  - Gateways (e.g. smartphones, smart hubs)
  - Cloud backend
  - Network communication
  - …

The 4 Stage IoT Solutions Architecture



https://analyticks.wordpress.com/2016/11/07/how-to-design-an-iot-ready-infrastructure-the-4-stage-architecture/

Device Challenges

- Lot’s of small devices
- Limited resources
  - Processing power
  - Internal Memory
  - Power Consumption

Device Challenges

- Lot’s of small devices
- Limited resources
  - Processing power
  - Internal Memory
  - Power Consumption
- -> Small amount of resources for Security

Example: Google's Office Doors (What happened?)

- Employee analysed the door controller communication of the encrypted messages on the internal network
  - Realized that the messages seemed to be non-random
  - Further analysis revealed **hardcoded key** used by all devices

https://www.forbes.com/sites/thomasbrewster/2018/09/03/googles-doors-hacked-wide-open-by-own-employee/

Example: Google's Office Doors (What happened?)

- Employee analysed the door controller communication of the encrypted messages on the internal network
  - Realized that the messages seemed to be non-random
  - Further analysis revealed **hardcoded key** used by all devices

- Now he could craft his own messages to control door controller
  - or he could also have replayed the original messages

- He could also lock out legitimate users

https://www.forbes.com/sites/thomasbrewster/2018/09/03/googles-doors-hacked-wide-open-by-own-employee/

Example: Google's Office Doors (The fix?)

- Quick Fix:
  - Google segmented the network to provide protection for vulnerable systems

Example: Google's Office Doors (The fix?)

- Quick Fix:
  - Google segmented the network to provide protection for vulnerable systems
- The device company came up with a solution
  - v2 of the board uses TLS for encryption of messages

Example: Google's Office Doors (The fix?)

- Quick Fix:
  - Google segmented the network to provide protection for vulnerable systems
- The device company came up with a solution
  - v2 of the board uses TLS for encryption of messages
- **Con:** The upgrade requires a hardware change since the original board does not have enough memory to run/upgrade to the new firmware

How to deal with Security

Four levels:

- Device
  - In-Secure devices
- Communications
  - Security Problems in connections
- Cloud (Back-End)
  - Security/privacy breaches because of data leaks
- Lifecycle Management
  - e.g. Update Problems

IoT OWASP Top 10

OWASP Top 10

- Open Web Application Security Project
- You (should) know the OWASP Top 10 (for Web application security)[8]

[8] Open Web Application Security Project

IoT OWASP Top 10

- They also have resources on IoT [9]
- And also prepare to release a new Top 10 for IoT[10]
  - Not directly *vulnerabilities* but *things to avoid*
  - The following is based on this draft version

[9] https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project
[10] https://danielmiessler.com/blog/preparing-to-release-the-owasp-iot-top-10-2018/

I1 - Weak, Guessable, or Hardcoded Passwords

- Use of
  - easily bruteforced
  - hardcoded
  - publicly available
  - and/or unchangeable passwords

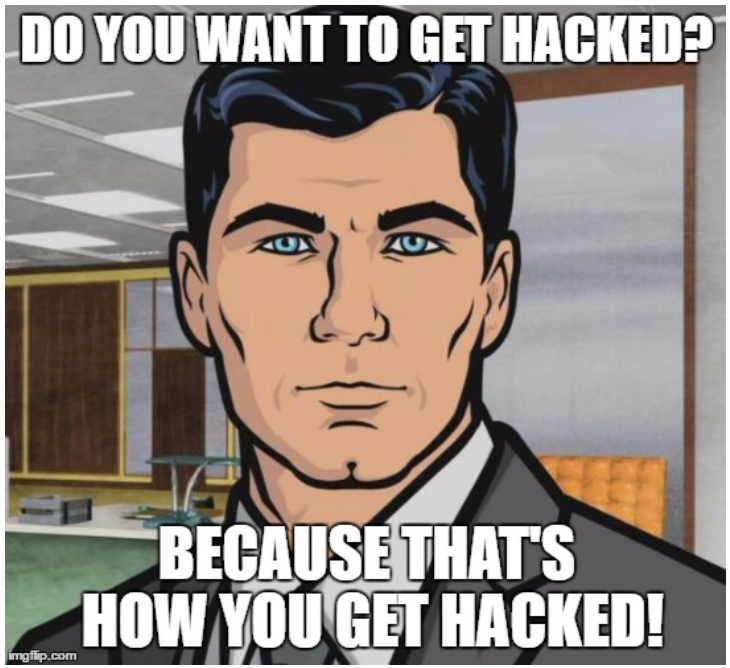
  in client-side software/firmware that can grant unauthorized access to deployed systems

I2 - Insecure Network Services / Protocols

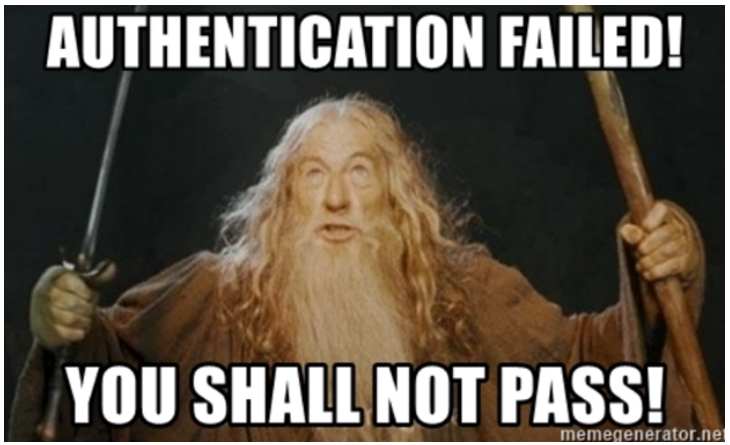
- Unneeded and/or insecure listening network services (especially internet facing) that compromise
  - confidentiality
  - integrity or
  - availability/authenticity

  of information or

- allow unauthorized remote control
  - e.g. Telnet, FTP, SSH, UPnP, etc.



I3 - Insecure Access Interfaces



- Insecure
  - web
  - backend API
  - cloud or
  - mobile interfaces

  that allow compromise of the product and /or its ecosystem.

- Common issues include:
  - a lack of authentication/authorization,
  - lacking or weak encryption,
  - and a lack of input and output filtering

I4 - Use of Insecure or Outdated Components

- Use of deprecated and insecure software components/libraries.
- Insecure customization of operating systems,
- and use of third-party software or hardware components from compromised supply chain

https://threatpost.com/bad-code-library-triggers-devils-ivy-vulnerability-in-millions-of-iot-devices/126913/

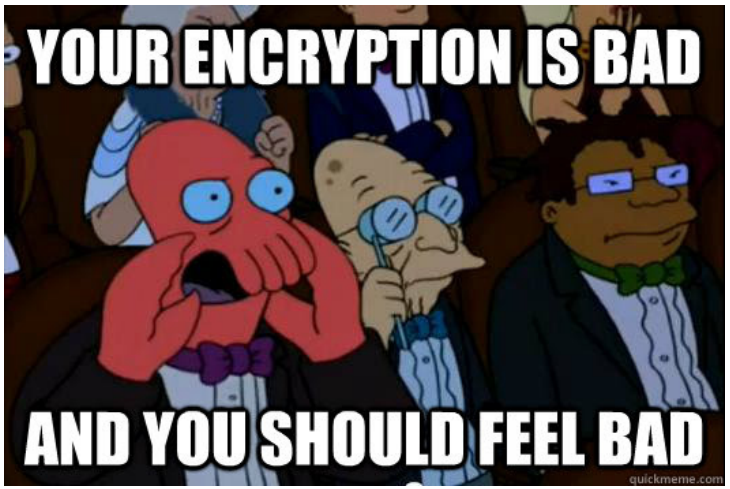
I5 - Lack of Secure Update Mechanism

- Lack of:
  - ability to securely update the device/ecosystem
  - firmware validation on device
  - secure delivery (un-encrypted in transit)
  - anti-rollback mechanisms
  - notifications of security changes due to updates

I6 - Insufficient Privacy Protection

- User's personal information
  - stored insecurely on device
  - is used insecurely, improperly, and/or without permission in logs and other artifacts
  - is transmitted insecurely over the network or the internet
  - or the system lacks adequate privacy disclosure before usage

I7 - Insecure Data Transfer and Storage

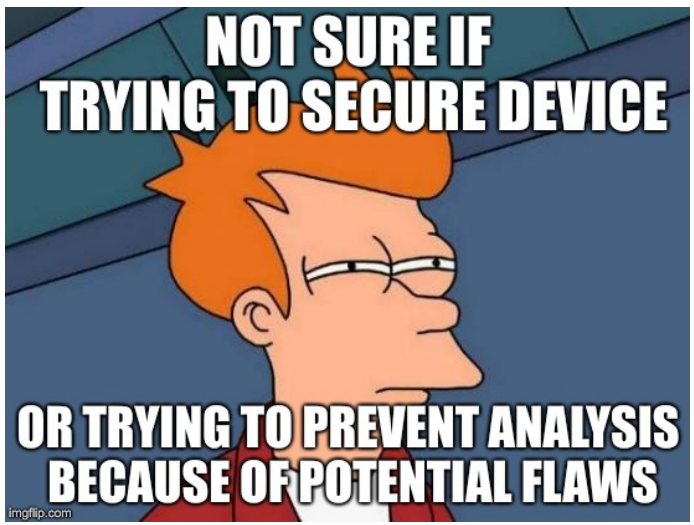


- Lack of security of sensitive data
  - at rest
  - in transit
  - or during processing
- e.g.,
  - weak or lacking cryptography,
  - mismanagement of keys
  - inefficient platform access controls
  - insufficient key rotation
  - absence of secure hardware backed storage

I8 - Lack of Physical Hardening

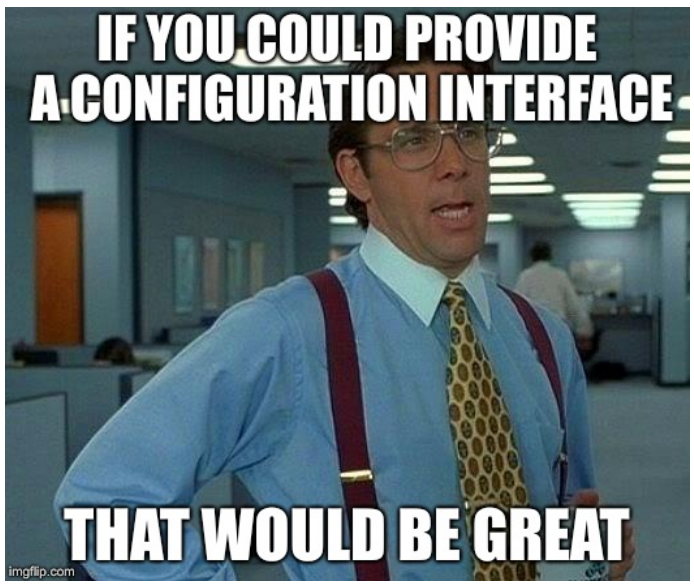
- Lack of
  - physical anti-tempering defenses and/or
  - system integrity checking

  that allows potential attackers to gain sensitive information that can help with a future remote attack



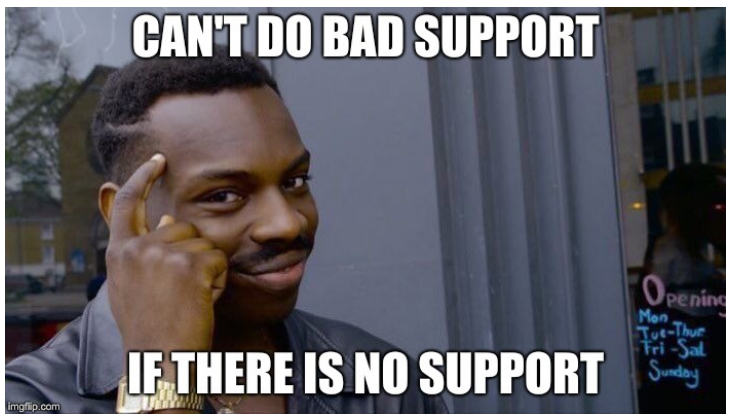
I9 - Insufficient Security Configurability

- A lack of vendor-provided product features to help the user secure the device through configuration, e.g.,
  - stronger authentication
  - logging and monitoring
  - encryption strength management
  - granular policy management
  - etc.



I10 - Lack of Device Management

- Lack of security support on existing devices deployed in production
  - including asset management
  - update management
  - and secure decommissioning



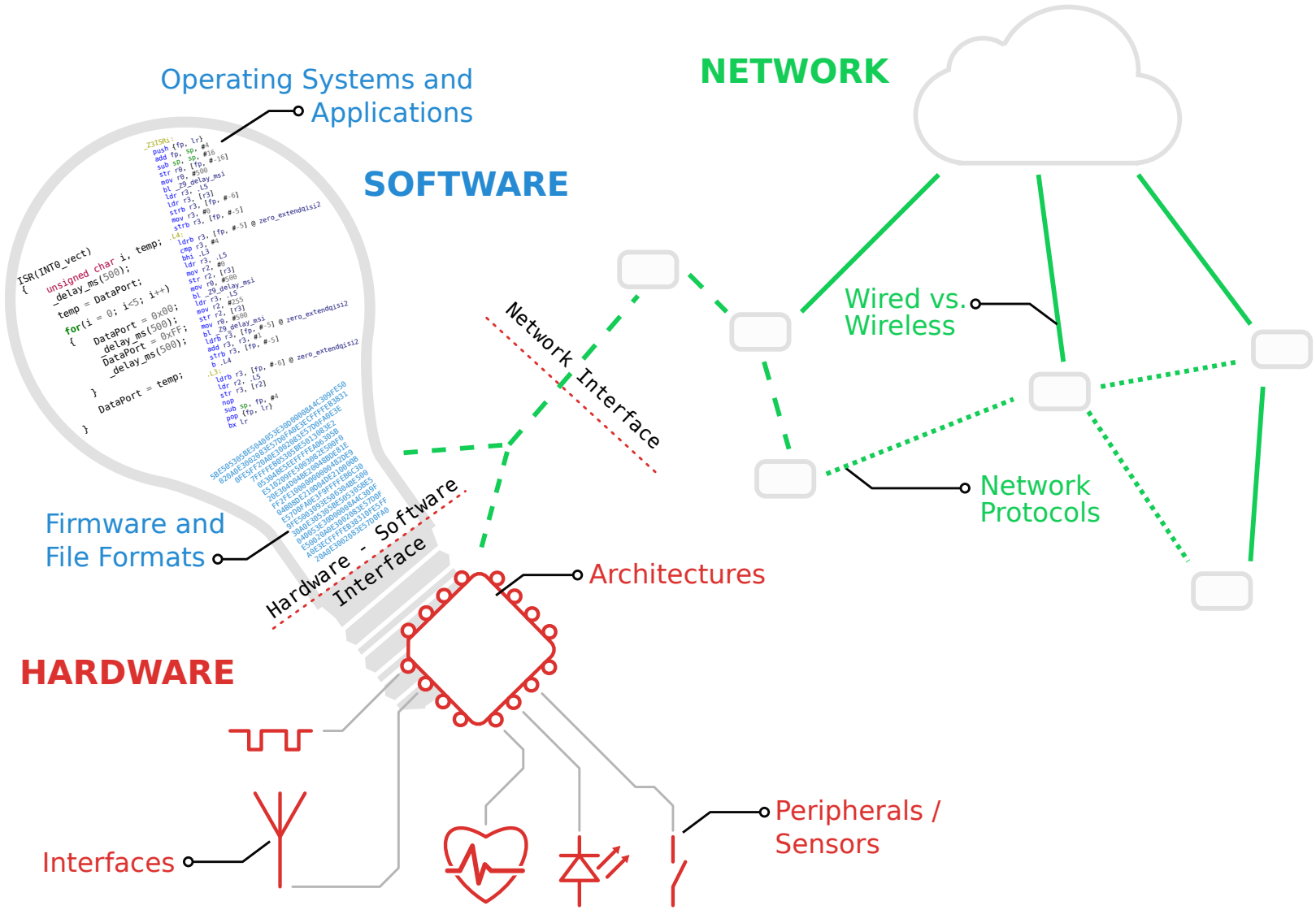
Top 10 Summary

- IoT Security is currently lacking behind
- But What can we do against it?
  - Make devices more secure
  - Find vulnerabilities and expose flaws

IoT Device Basics

Based on: https://media.blackhat.com/us-13/US-13-Zaddach-Workshop-on-Embedded-Devices-Security-and-Firmware-Reverse-Engineering-Slides.pdf

Basic Overview



What is Firmware

- Ascher Opler coined the term “firmware” in a 1967
- Originally meant contents of writable control store containing microcode (not Hardware, not Standard CPU instructions)

[11] https://en.wikipedia.org/wiki/Firmware

What is Firmware

- Ascher Opler coined the term “firmware” in a 1967
- Originally meant contents of writable control store containing microcode (not Hardware, not Standard CPU instructions)

“In electronic systems and computing, firmware is a specific class of computer software that provides the low-level control for the device’s specific hardware.”[11]

- Might provide interface for high level Software
- Or act as the operating system directly for less complex devices

[11] https://en.wikipedia.org/wiki/Firmware

Architectures and Firmware

- Usually around some micro CPU / Microcontroller
  - e.g. PIC, AVR, Intel, MIPS, ESP, RISC-V, ARC, …
- Very common cores are ARM Cortex M0, M3 and M4 based
- Those devices come with a lot of peripherals to support virtually any need
- Need to develop both hardware and software side

Different Available Operating Systems

- Provide Basic functionality
  - e.g. (Simple) Memory Management, Multithreading, network stacks, standard protocolls (e.g. TCP, UDP, …)
- Examples:
  - Linux
  - RTOS
  - RIOT
  - Contiki
  - Zephyr
  - TinyOS
  - VxWorks
  - Windows IoT
  - …

Embedded Bootloaders

- Provide Loading of Code from different interfaces
  - e.g. SD Card, NOR Flash (through SPI or I²C) of NAND Flash
  - Could also be a staged loader
- Might also have additional functionality
  - e.g. Firmware updates
- Examples:
  - u-boot
  - RedBoot
  - BareBox
  - …

Firmware Challenges

- Non-standard formats
  - Encrypted chunks
  - Non-standard update channels
  - Firmwares come and go, vendors quickly withdraw them from support/ftp sites
- Non-standard update procedures
  - Printer’s updates via vendor-specific Printer Job Language (PJL) hacks
  - different others

Updating the Firmware

- Firmware Update built-in functionality
  - Web-based upload
  - Socket-based upload
  - USB-based upload
- Firmware Update function in the bootloader
- USB-boot recovery
- Rescue partition, e.g.:
  - New firmware is written to a safe space and integrity-checked before it is activated
  - Old firmware is not overwritten before new one is active
- JTAG/ISP/Parallel programming

Firmware Update Pitfalls

- TOCTOU attacks
- Non-mutual-authenticating update protocols
- Non-signed packages
- Non-verified signatures
- Incorrectly/inconsistently verified signatures
- Leaking signature keys

Firmware Formats

- Firmware comes in different kinds formants
- Usually you need to identify and unpack internals to analyse the firmware

Firmware Formats - Typical Objects

- Bootloader (1st/2nd stage)
- Kernel
- File-system images
- User-land binaries
- Resources and support files
- Web-server/web-interface

Firmware Formats - Component Category View

- Full-blown (full-OS/kernel + bootloader + libs + apps)
- Integrated (apps + OS-as-a-lib)
- Partial updates (apps or libs or resources or support)

Firmware Formats – Packing Category View

- Pure archives (CPIO/Ar/Tar/GZip/BZip/LZxxx/RPM)
- Pure filesystems (YAFFS, JFFS2, extNfs)
- Pure binary formats (SREC, iHEX, ELF)
- Hybrids (any breed of above)

Firmware Analysis

Based on: https://media.blackhat.com/us-13/US-13-Zaddach-Workshop-on-Embedded-Devices-Security-and-Firmware-Reverse-Engineering-Slides.pdf

Overview

- Get the firmware
- Reconnaissance
- Unpacking
- Reuse engineering (check code.google.com and sourceforge.net)
- Localize point of interest
  - e.g. embedded application or firmware entry points
- Decompile/compile/tweak/fuzz/pentest/fun!

Getting the Firmware

- Use Low-Level Methods [12]
- Sometimes it might be easier to achieve:
  - Present on the product CD
  - Download from manufacturer FTP/HTTP site
    - Use Google • FTP index sites (mmnt.net, ftpfiles.net)
  - Wireshark traces (manufacturer firmware download tool or device communication itself)
  - Analyse Mobile Application

[12] You already had the hardware security lecture

Getting the Firmware - Common Hardware Issues

- JTAG & UART ports available
- Exposed buses (I²C, Serial)
- Unprotected external storages (FLASH, sdcard…)
- Unprotected radio interfaces (WiFi, Bluetooth, ZigBee…)
- Debugging consoles left active
- Unprotected bootloader and FW updates
- Fuses not set to make internal flash unreadable

Reconnaissance

- Use *strings* to identify specific Operating Systems, Manufacturers,… (match against DB)
- Read the Specification
  - Including the specification of the CPU/Microcontroller in case you want to do static or dynamic analysis
- Use tools like *binwalk* to extract known file formats/file systems or binary blobs from the dump

Unpacking

- Depending on the firmware it might be necessary to unpack information or convert the dump
  - e.g. convert iHEX files to ELF files
- This might be a problem for automatic analysis systems, since they first need to identify the file
- Also, vendors start encrypting their updates
  - However, sometimes their encryption schemes are flawed
  - e.g. Static Symmetric Key inside the Android Application to decrypt FW Updates before deploying them on the Device

Static Analysis

- When you have the unpacked firmware use the “usual” tools to do
  - disassembly
  - decompilation
  - CFG creation
  - extraction of static information
    - e.g. strings
  - …
- Examples:
  - IDA Pro
  - radare2
  - …

Dynamic Analysis

- IoT Devices come in different flavours, architectures, file systems
- How can we analyse these dynamically

Debugging

- Attach a debugger to the Device/Embedded OS
  - OCD (On chip debugging)
  - e.g. through JTAG, PDI (Programming Debugging Interface)
  - Software debugger (e.g. GNU stub or ARM Angel Debug monitor)
  - OS debug capabilities (e.g. KDB/KGDB)
- Or emulate the target for further analysis
  - QEMU is very common here

Emulation - Prerequisites

- Kernel image with a superset of kernel modules
- QEMU compiled with embedded device CPU support (e.g. ARM, MIPS)
- Firmware - most usually split into smaller parts/FS-images which do not break QEMU

Emulation - Peripherals

- IoT Devices might also make use of sensors and actuators
- How can we access these devices during analysis?

Emulation - Peripherals

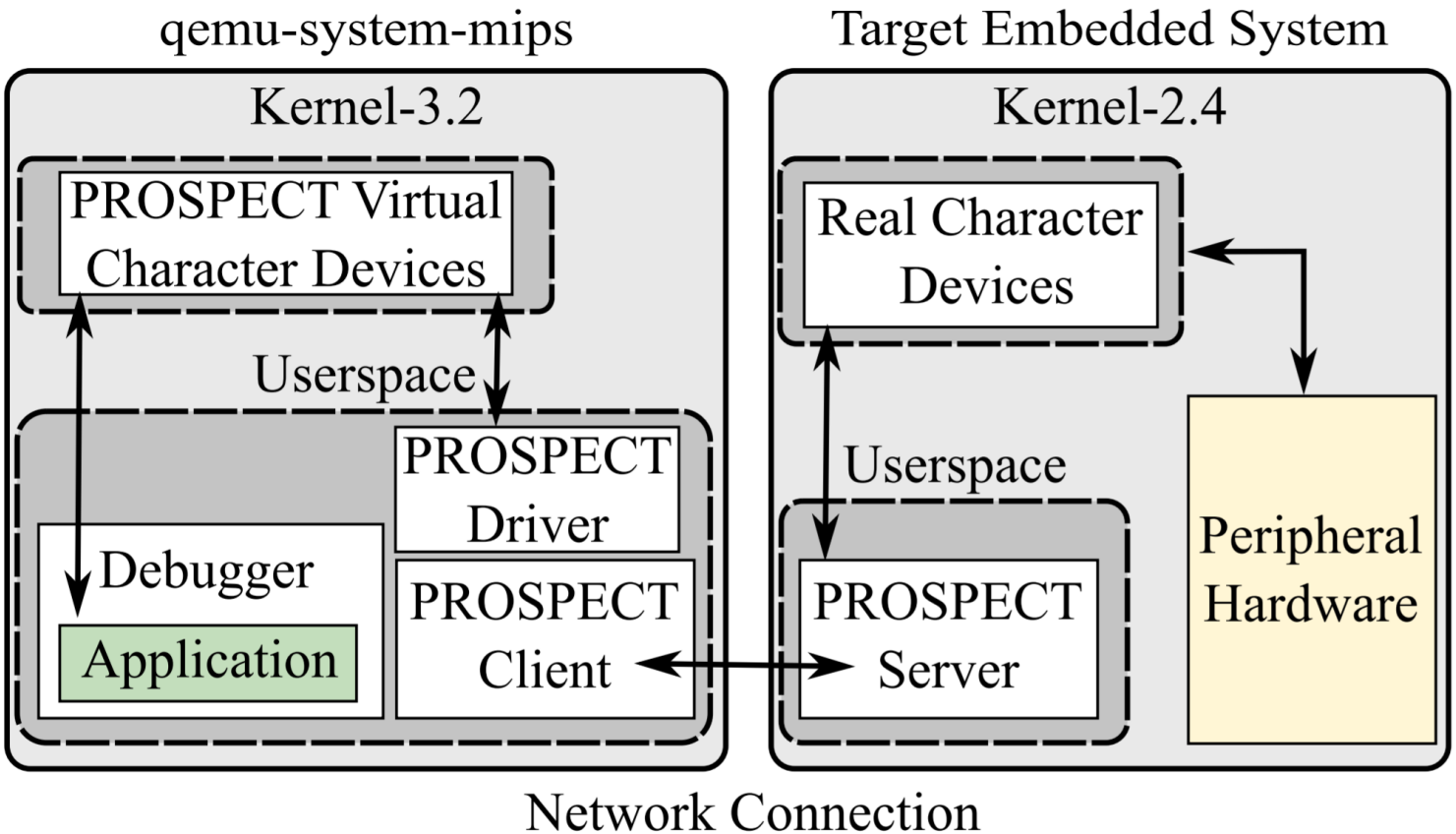
- IoT Devices might also make use of sensors and actuators
- How can we access these devices during analysis?
- Some solutions have been proposed, e.g.:
  - Emulate the peripherals
  - Change/Adapt the Firmware
  - Bridge the peripherals during analysis
- Several (research) projects working on this…

Prospect

- Allows to tunnel peripheral access from device to host PC
- Works on embedded Linux
- Currently requires TCP/IP stack/communication for communication with the target under Test

Kammerstetter, Markus, Christian Platzer, and Wolfgang Kastner. ”Prospect: peripheral proxying supported embedded code testing.” Proceedings of the 9th ACM symposium on Information, computer and communications security. ACM, 2014.

Prospect Architecture



Avatar

- Orchestrate Emulator execution together with real hardware
  - e.g. tunneling of I/O operations
- Allow S2E to work on target environments
- Different target backends:
  - GDB serial protocol (e.g. JTAG GDB server or debugger stub)
  - Low Level access to the OpenOCD's (Open On-Chip Debugger) JTAG debugging interface
  - custom binary protocol which talks to custom *Avatar* debugger

Zaddach, Jonas, et al. "AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares." NDSS. 2014.

Avatar² - A Multi-target Orchestration Platform I

- Evolution of *Avatar* to a more generalized binary analysis framework based on Avatars following concepts:
  - **Target Orchestration:**
    - includes transfer of execution from one tool to another
  - **Sepration of Execution and Memory**
    - allows *remote memory*: execution and memory reads/writes can happen in different targets
  - **State transfer and synchronization**
    - transfer states between targets
- Allows to interconnect
  - debuggers
  - emulators
  - and other dynamic binary instrumentation frameworks

Avatar² - A Multi-target Orchestration Platform II

- Structured into several components
  - Avatar core for orchestration
  - Targets to abstract endpoints
  - Protocols divided by purpose
    - e.g. memory protocol, execution protocol, …
  - Endpoints (anything that should be orchestrated for analysis)
- Under the Hood:
  - Architecture independent
  - Internal memory representation (to synchronize tools)
  - Peripheral Modeling (and bridging like Avatar)
  - Plugin System
- Open Source
  - `https://github.com/avatartwo/avatar2`

Muench, Marius, et al. ”Avatar 2: A Multi-target Orchestration Platform.” Workshop on Binary Analysis Research (colocated with NDSS Symposium)(February 2018), BAR. Vol. 18. 2018.

Avatar² - Supported Targets

- The Gnu Debugger (GDB)
- OpenOCD
  - Open source tool which is able to control debug dongles attached to e.g. s Joint Action Test Group (JTAG) or Serial Wire Debug (SWD) ports
- Quick Emulator (QEMU)
  - dynamic binary translation to allow cross-architecture emulation
- PANDA (Platform for Architecture-Neutral Dynamic Analysis)
  - dynamic analysis framework with focus on reverse engineering (based on QEMU)
- angr
  - symbolic execution and program analysis framework

Other Research Directions I

- Costin, Andrei, et al. “A Large-Scale Analysis of the Security of Embedded Firmwares.” USENIX Security Symposium. 2014.
- Shoshitaishvili, Yan, et al. “Firmalice-Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware.” NDSS. 2015.
- Costin, Andrei, Apostolis Zarras, and Aurélien Francillon. “Automated dynamic firmware analysis at scale: a case study on embedded web interfaces.” Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. ACM, 2016.

Other Research Directions II

- Muench, Marius, et al. “What you corrupt is not what you crash: Challenges in fuzzing embedded devices.” NDSS 2018, Network and Distributed Systems Security Symposium, 18-21 February 2018, San Diego, CA, USA. 2018.
- …

Secure Device Architecture

Based on: https://cansecwest.com/slides/2017/CSW2017_ScottKelly_SecureBoot.pdf

Important IoT security architecture features

- Chip security in the form of TPMs (Trusted Platform Modules)
  - act as a root of trust by protecting sensitive information and credentials
  - i.e., not releasing encryption keys outside the chip
- Secure booting can be used to ensure only verified software will run on the device.
- Even physical security protection
  - e.g., full metal shield covering all internal circuitry
  - can be employed to guard against tampering if an intruder gains physical access to the device.

Why Secure Boot?

- It will *always* be difficult to secure rich applications and remove all flaws
  - We have to accept the possibility for exploits happening
- But, we don't have to allow malware to replace trusted code (e.g. bootloaders, OS, system software, etc.)
  - Not if we correctly implement secure boot.

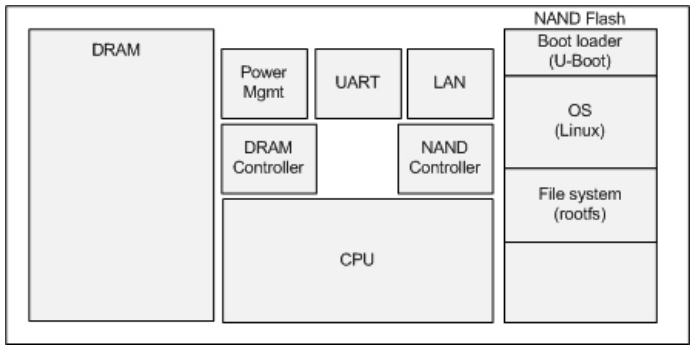
What is Secure Boot?

- Means that only authorized *system* code runs
  - If image is corrupted, or you try to install your own (unauthorized) code, system will not run.
- Wikipedia: Some devices implement a feature called “verified boot”, “trusted boot” or “secure boot”, which will only allow signed software to run on the device, usually from the device manufacturer. This is considered a restriction unless users either have the ability to disable it or have the ability to sign the software.[13]

[13] https://en.wikipedia.org/wiki/Hardware_restriction#Secure_boot

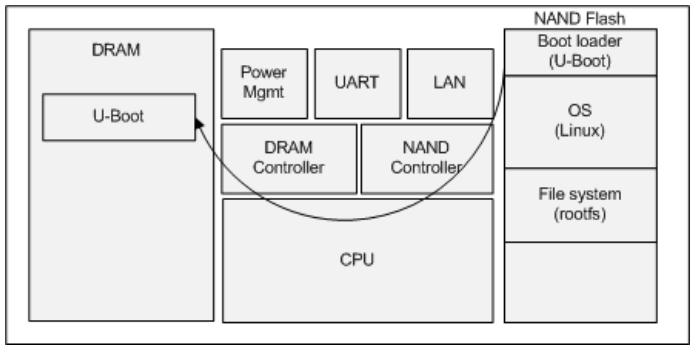
Embedded Systems Startup

- Embedded systems generally include
  - NAND/NOR Flash
    - non-volatile memory in \\ which firmware is stored
  - CPU
    - processor for OS/apps
  - DRAM
    - random access memory \\ (just like your PC)
  - Interfaces
    - Wifi, ethernet, etc.



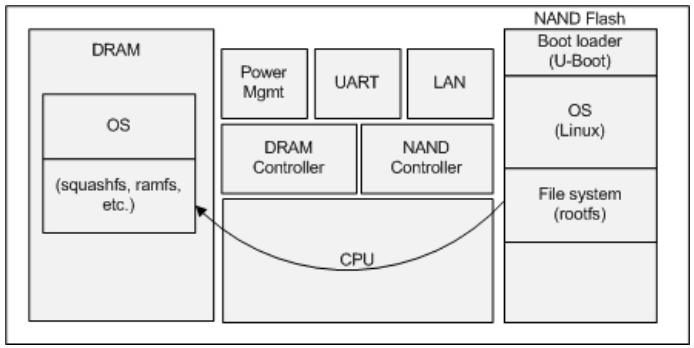
Embedded Systems Startup (contd.)

- At power-on
  - Processor comes out of reset
  - Begins running code from ROM or flash
- Boot Loader (BL) is first non-ROM firmware to run
- ROM/BL initializes HW (memory, etc.)
- BL copied (by ROM or self) into DRAM before continuing



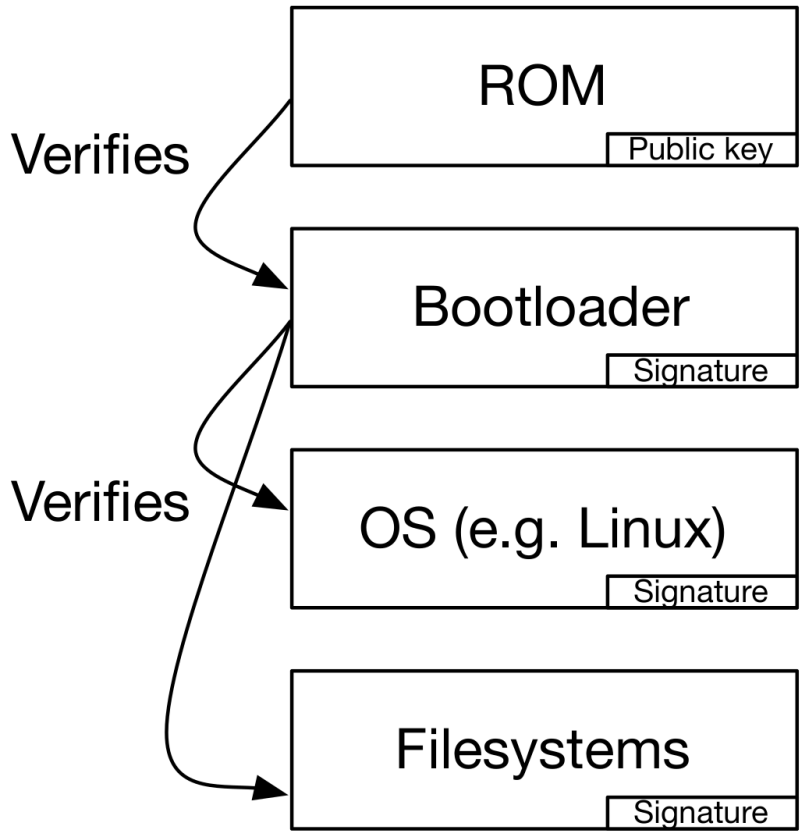
Embedded Systems Startup (contd.)

- BL continues hardware initialization from DRAM
- BL validates, loads, and jumps into OS kernel
- OS finishes init, goes to runtime steady state



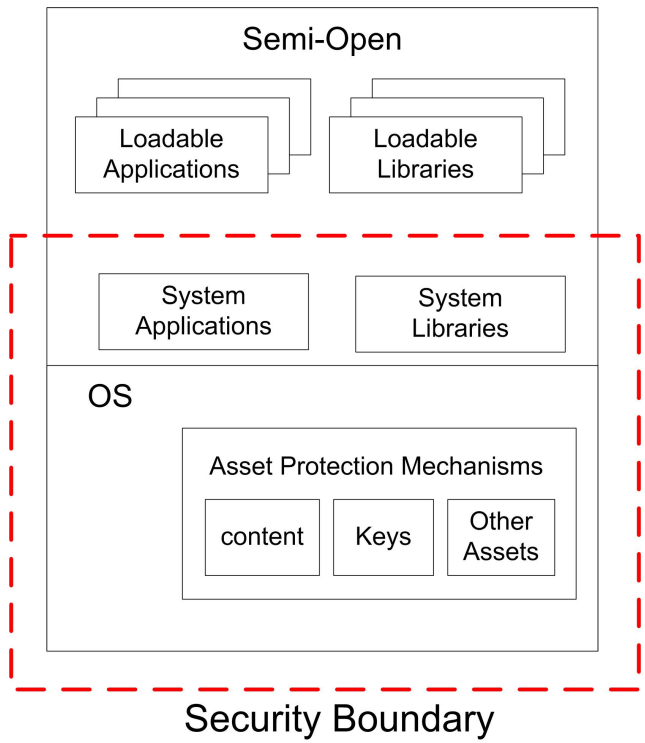
Simplistic View of Secure Boot

- On reset, processor starts from ROM
  - (Practically) immutable code
- ROM code loads/verifies bootloader
  - If invalid, halt.
- Bootloader loads/verifies OS and r/o filesystem(s)
  - If invalid, halt.
- Only verified (authorized) firmware is allowed to run.



Post-boot Secure Execution Environment

- Security protections set up during boot
  - Can go forward (set more protections) but should never be able to go back
- Secure environment established
  - Everything inside of boundary is in known state
  - Can “trust” this system
    - It will behave in a predicable way, as expected



Subverting the Boot Process

- Malware often circumvents boot process
  - Attack replaces some part of early boot code
  - Takes control of the system early on
  - Robust secure boot can prevent this.
- Of course, application may exploit system bug (later)
  - But robustly configured system can still protect some assets, operations
- Compromise at later phase can’t undo previous “locks”
  - System can potentially be recovered by reboot

How to attack Secure Boot

- Glitching
  - Cause system to erroneously accept (or ignore!) invalid signature
- Compromise signing key
  - Then, you can sign your own firmware images!
- Break crypto (e.g. factor RSA modulus)
  - This should only be possible if the crypto is used/implemented badly
- Find bug/flaw in validation code
  - Buffer overflow, integer {under,over}flow, etc.
  - Incorrect crypto implementation
- Take advantage of flawed design.

Summary

IoT Security?

- IoT Security is not so easy to get right
  - Lot's of things that can break
  - see OWASP IoT Top 10
- Today we just got a glimpse at the problem
- Analysis of IoT devices concerning Security Issues still poses problems

Questions?

Nintendo Switch - BootROM Vulnerability

https://www.youtube.com/watch?v=L3PPWVPg2WI