Security for the Rest of Us

What Matters and Where to Start

Mark Omo

Embedded Online Conference

Marcus Engineering LLC

www.embeddedonlineconference.com

AGENDA

1. What is Security?
2. How Risky is my Device?
3. Know your Product
4. Know your Software
5. The Rest of the Owl
6. Business needs

AGENDA

Mark Omo

Director of Cat Herding at Marcus Engineering

Focus: Embedded systems with teeth, where life, limb, or convenience are on the line.

Mark Omo, Director of Cat Herding at Marcus Engineering, leads the engineering team with a focus on embedded systems and safety-critical product development.

He has a background in embedded systems, security, and consumer products, and experience spanning medical, industrial, aerospace, and consumer markets.

AGENDA

1 **What is Security?**

What is Security?

Security is Risk Reduction

Fundamentally **preventing harm**

Preventing devices from **doing things** they are **not supposed to do**

- Share private information
- Work in a way they are not supposed to
- Functioning wrong or not at all
- Propagating harm

Why do People Hack Devices?

To make money

That means:

Effort < Reward

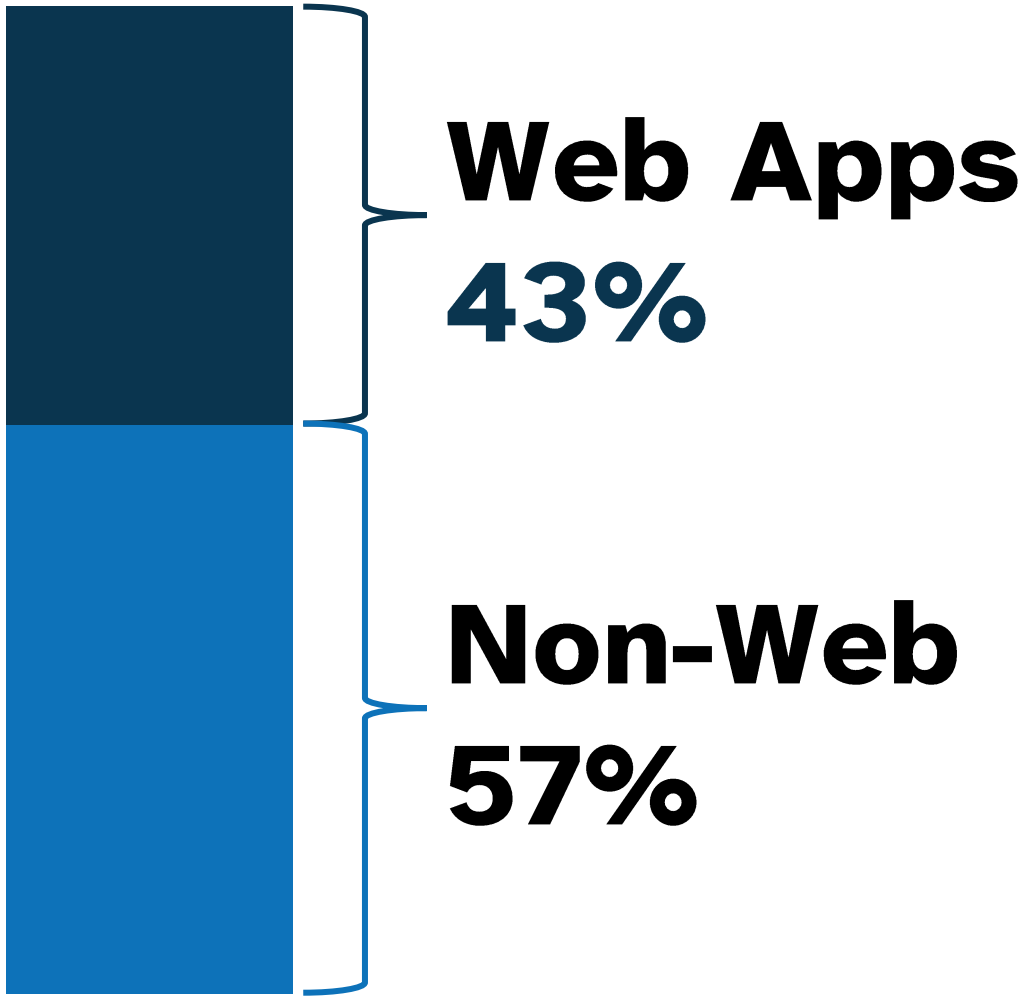
Security is **increasing effort** → **above reward**

How do devices really break?

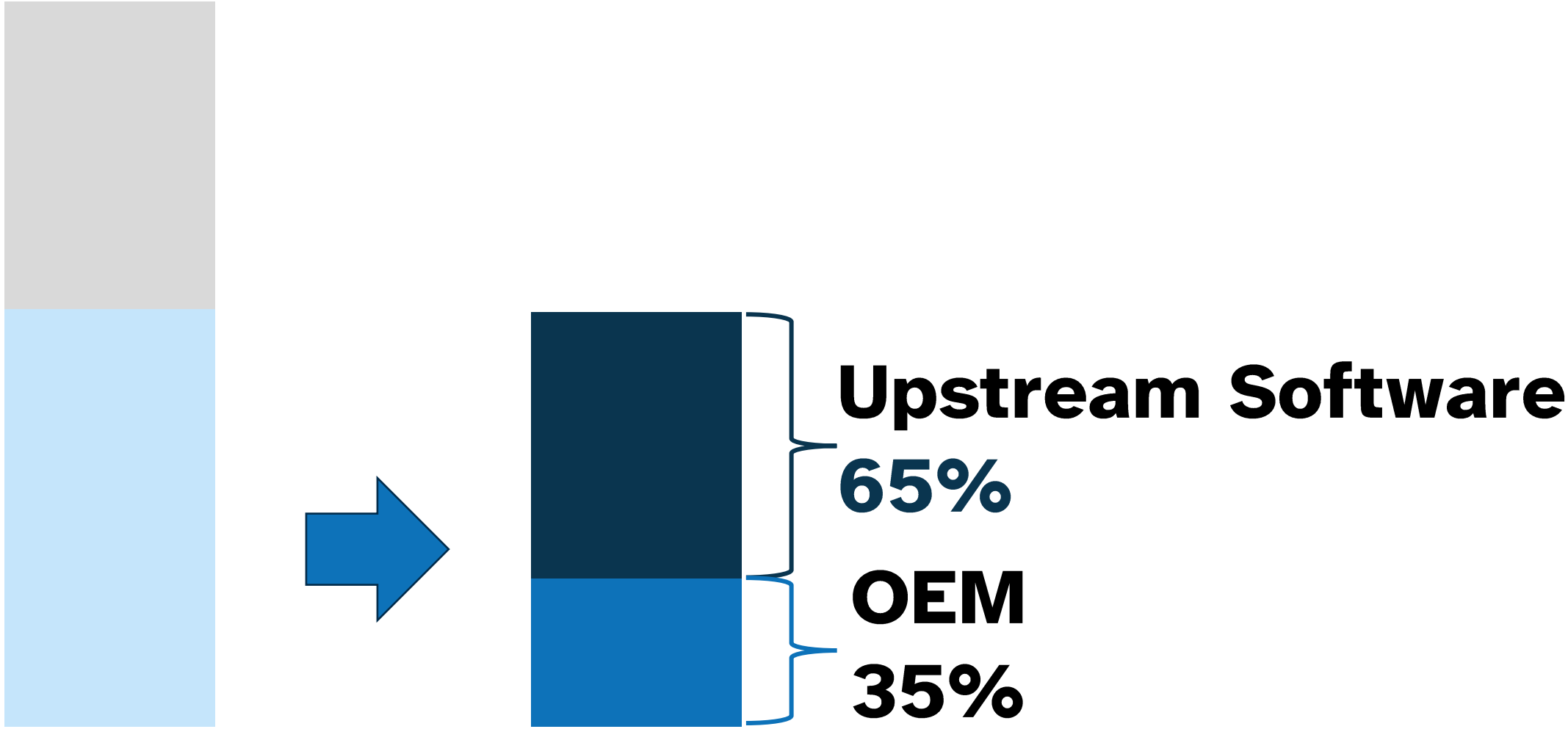
NIST keeps the National Vulnerability Database that tracks vulnerabilities in devices and software

Here is a breakdown of 25,000 IoT vulnerabilities (Over the last 5 years)

IoT Vulnerabilities



IoT Vulnerabilities



IoT Vulnerabilities

Other 2%

Cloud Communication 14%

Boot/Update Integrity 14%

Default Creds/Backdoors 22%

Custom Protocols

49%

Why Me?

Why would people hack my automated fish counter?

2 Categories of Attackers:

- **“Dragnet”** attackers that perform **mass scans** for **vulnerable devices**

- Targeted attackers, who look for **low-hanging fruit**
    - Or very ripe fruit in the form of **wide deployment**

2 How Risky is my Device?

Device Risk Categories

1. Unconnected Devices
2. Locally connected devices
3. Gateway connected devices
4. Directly connected devices
5. Web servers

More Risk

Unconnected Devices

Devices with no direct outside data connection

Unconnected devices are the lowest risk devices

No connectivity ports at all

Locally connected devices

Devices with no indirect internet connection

Devices that only transfer data over a local bus. They don't send data to or receive commands from external systems over the internet.

Devices that just talk to one another (if you have a cloud, this is NOT you)

Simple Industrial Automation

Camera lens

Gateway connected devices

Devices with an indirect internet connection

Devices that send data to another device (such as a phone, PC, or gateway) that has internet access.

Most BLE devices, USB devices, Smart Home, etc.

Networked Industrial Automation (Devices)

USB Connected Instruments/Products

Smart Home Devices (ZigBee/Z-wave)

BLE Devices

Directly connected devices

Devices with an direct internet connection

Devices that connect to the internet directly.

Anything with WiFi or Ethernet

Networked Industrial Automation (Gateways/Controllers)

Smart Home Devices (WiFi)

(Almost) Anything running Linux (like a RasPi)

Cellular or Satellite connected devices

Web servers

Devices with a web server

Please don't

Networked Industrial Automation (Gateways/Controllers)

Things running Linux

Network equipment like routers, switches, etc.

What do I do Now?

3

Know your Product

Understand your System

- **What does it do?**
  - Administer Drugs? Ring to alert customers? Show if you watered your pants? Track Shipments? Send vehicle data to a cloud? Track steps?

- **Who uses this device?**
  - Hospitals? Small Businesses? Consumers? Paper Logistics trackers?

- **What does it need to protect?**
  - Control functions - What is the worst thing it could do?
  - Data – What is the most sensitive data this controls?
    - Personally Identifying Information (PII), keys. etc.

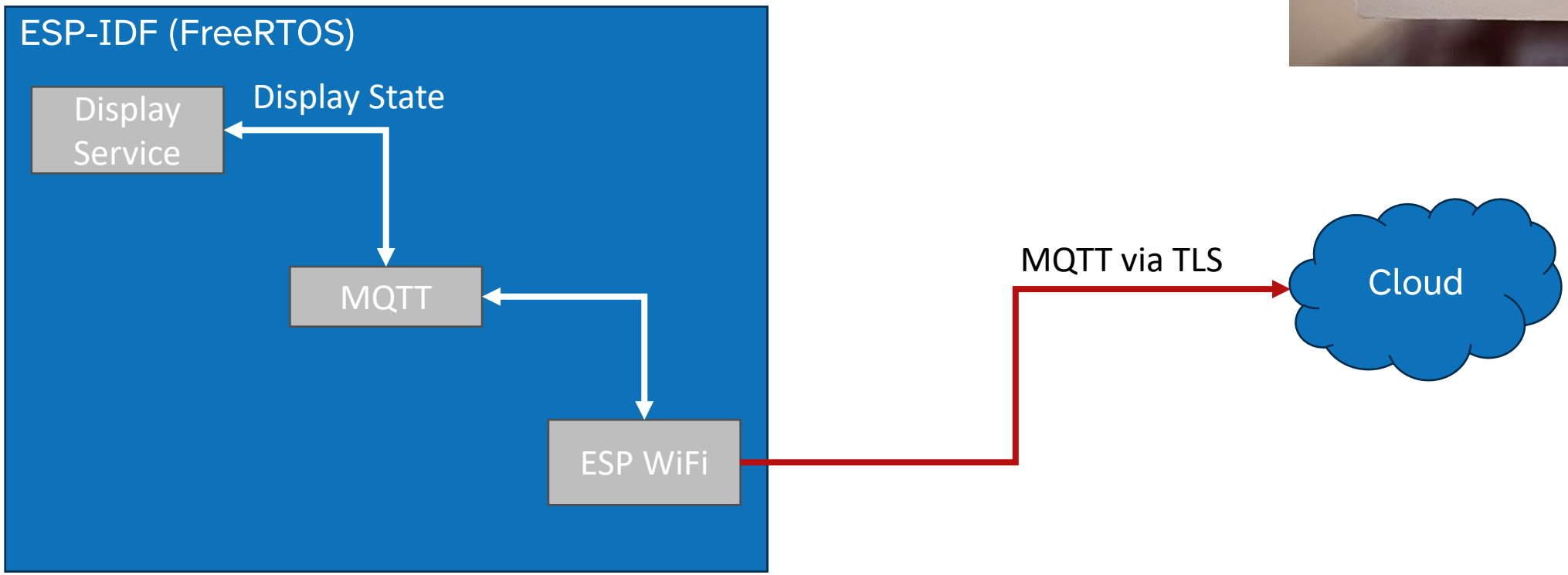
Understand your System

An ESP32 e-ink display

- **What does it do?**
  - Shows the current status and next meeting
- **Who uses this device?**
  - Medium business with 25-500 people with one or a few buildings
- **What does it need to protect?**
  - Control functions?
    - Likely none
  - Data?
    - Calendar access keys

Understand your System

Draw a picture



4 Know your Software

What’s inside your system?

The number one source of vulnerabilities is someone else's code (65%)

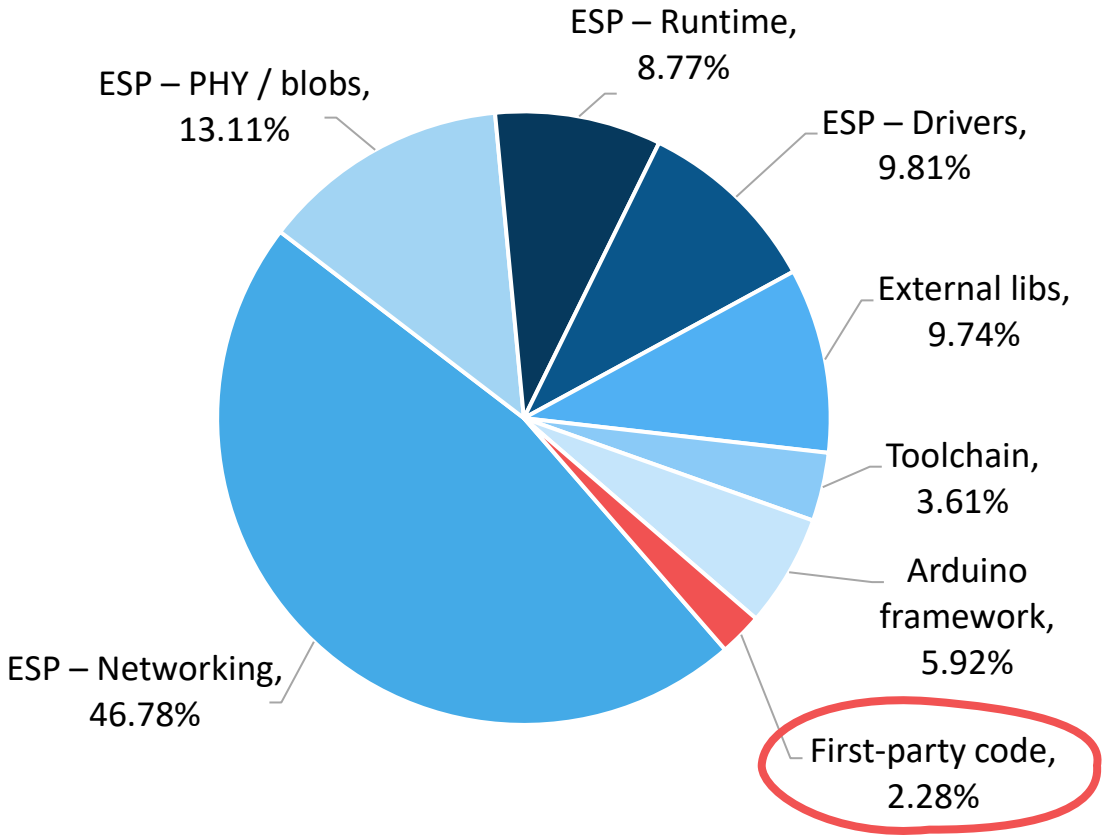
That’s the way it should be!

If you write it yourself, that means you don’t know the vulnerabilities

Breakdown of our example product (based on a real commercial project).

6,261 lines of application code

Vs. 253k lines of third-party code



How do we document all the other software?

A Software Bill of Materials (SBOM) is a list of all the 3rd party software included in your project.

Usually stored as a JSON file in one of two common formats:

- CycloneDX
- SPDX

Machine-generated and readable files that document all the software used in a project.

Keep it up to date!


```
{
  "components": [
    {
      "bom-ref": "CMSIS_PACK@6.2.0",
      "name": "CMSIS_PACK",
      "scope": "required",
      "supplier": {
        "name": "Microchip Technology Inc."
      },
      "type": "framework",
      "version": "6.2.0"
    },
    {
      "bom-ref": "SAMV71Q21B_DFP@4.12.237",
      "name": "SAMV71Q21B_DFP",
      "scope": "required",
      "supplier": {
        "name": "Microchip Technology Inc."
      },
      "type": "framework",
      "version": "4.12.237"
    },
    {
      "bom-ref": "ST LSM6DSV16X Drivers@4.2.0",
      "name": "ST LSM6DSV16X Drivers",
      "scope": "required",
      "supplier": {
        "name": "ST Microelectronics"
      },
      "type": "library",
      "version": "4.2.0"
    }
  ],
  "dependencies": [
```

Step 2: What's inside your system?

If you are in one of the following cases, there is tooling!

- Zephyr – `west spdx`
- Buildroot - `cyclonedx-buildroot`
- Yocto - `create-spdx`
- Esp-idf - `esp-idf-sbom`
- nRF Connect - `west ncs-sbom`
- Etc.

You still need to manually add dependencies you pull in yourself.

Step 2: What's inside your system?

If you are not in a nice ecosystem, then you need to do it yourself:

Core drivers

3rd party driver

Core FW

Dependency registration

Export

```python
import cyclonedx.model, cyclonedx.model.bom, cyclonedx.model.component, cyclonedx.model.contact, cyclonedx.output.json

sbom = cyclonedx.model.bom.Bom()
version = "1.1.0"

samv_dfp = cyclonedx.model.component.Component(
    supplier=cyclonedx.model.contact.OrganizationalEntity(
        name="Microchip Technology Inc."
    ),
    name="SAMV71Q21B_DFP",
    version="4.12.237",
    type=cyclonedx.model.component.ComponentType.FRAMEWORK,
    bom_ref="SAMV71Q21B_DFP@4.12.237",
    scope=cyclonedx.model.component.ComponentScope.REQUIRED,
)
sbom.components.add(samv_dfp)

st_driver = cyclonedx.model.component.Component(
    supplier=cyclonedx.model.contact.OrganizationalEntity(name="ST Microelectronics"),
    name="ST LSM6DSV16X Drivers",
    version="4.2.0",
    type=cyclonedx.model.component.ComponentType.LIBRARY,
    bom_ref="ST LSM6DSV16X Drivers@4.2.0",
    scope=cyclonedx.model.component.ComponentScope.REQUIRED,
)
sbom.components.add(st_driver)

root = cyclonedx.model.component.Component(
    supplier=cyclonedx.model.contact.OrganizationalEntity(
        name="Marcus Engineering",
        urls=[cyclonedx.model.XsUri("https://www.marcusengineering.com/")],
    ),
    name="Marcus_Engineering_Controller_Firmware",
    version=version,
    type=cyclonedx.model.component.ComponentType.FIRMWARE,
    bom_ref=f"Marcus_Engineering_Controller_Firmware@{version}",
    scope=cyclonedx.model.component.ComponentScope.REQUIRED,
)
sbom.metadata.component = root
sbom.register_dependency(root, [samv_dfp, st_driver])

Path(f"sbom_{version}.json").write_text(
    cyclonedx.output.json.JsonV1Dot6(sbom).output_as_string(indent=2),
    encoding="utf-8",
)
```

How do I make an SBOM?

What's vulnerable?

Scanning the SBOM for vulnerabilities

Use tools to scan your SBOM for vulnerable versions of code
I like cve-bin-tool

Most of the work is massaging dependencies to get scanned and fixing links
Then, check if the vulnerabilities actually affect you

The tooling out there is **geared towards systems that run Linux**

What's vulnerable?

What if I have an embedded system?

Lots of embedded libraries have vulnerabilities that automated tooling **will not pick up**

You need to **manually check the dependencies** yourself as well

Check network/outward-facing libraries like:

- Network stacks (BLE drivers, Wi-Fi drivers, etc.)
- RTOS (FreeRTOS, Zepher, etc.)
- Crypto libraries (mbedTLS, wolfSSL, etc,)
- Web servers (lwIP httpd, Mongoose, etc.)
- Web clients (MQTT Client, HTTP client, etc.)

What's vulnerable?

Rely on your vendors (sometimes)

- Espressif
  - https://www.espressif.com/en/support/documents/advisories
- Nordic Semi
  - https://docs.nordicsemi.com/bundle/struct_sa/page/struct/sa.html

- Limited central list:
  - Microchip
  - Renesas

- No central list:
  - Silicon Labs
  - STMicroelectronics
  - NXP
  - TI

Repeat

Check your product's SBOM as often as you can!

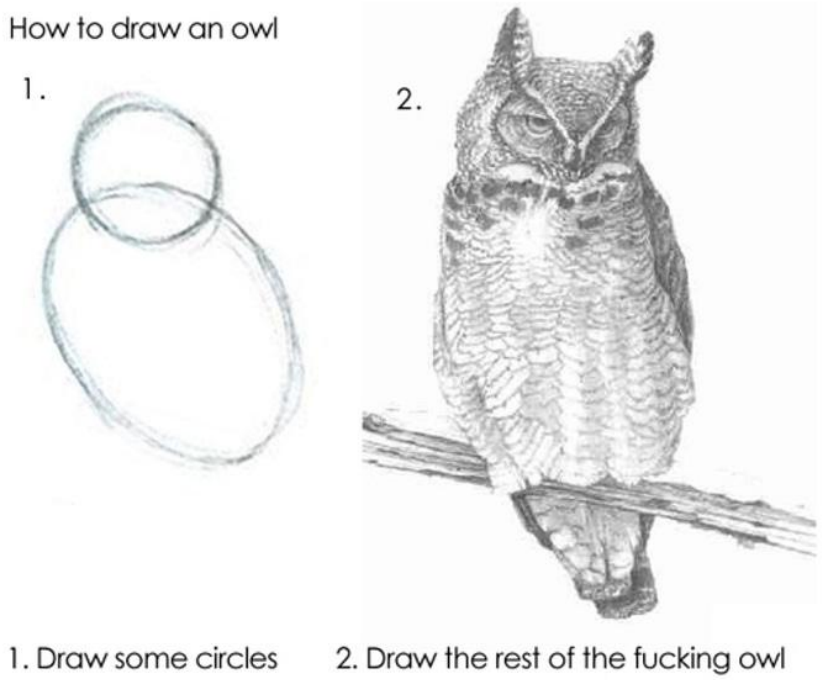
Make it part of your CI tooling if you can

- *Best practice is "continually" for Linux-based systems*

For non-Linux systems, check at least twice a year manually

- *Best practice is quarterly or more often*

5 The Rest of the Owl

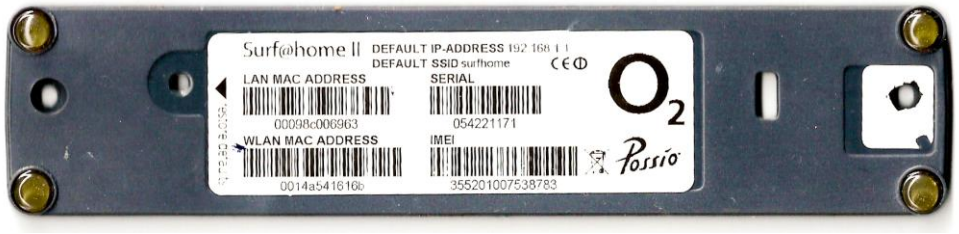


What about the rest of my product?

What do I need to design into my device to be secure?

6 Things

1: Device Identification



Every device needs a way to be recognized and told apart from others

Devices need to have a unique ID so you can tell them apart

- Logical ID for data connections (ex. to a server)
  - For some standards, this needs to NOT be based on anything you can observe
  - Best in class: Unique Public Key
  - Can also be a factory UUID or SN

- Physical ID you can see when it's off
  - Best in class: MAC address on a sticker
  - Can also be a factory SN

You can buy pre-provisioned Secure Elements with certificates that you can bulk upload, Ex. TECC608C-TNGTLS

2: Device Configuration

Review your configuration; what should be secured?

1. **List your device's configuration**
    1. Ex. WiFi passwords, BLE Bonding, Cloud keys, Passwords
    2. Device settings (brightness, runtime, voltage range, patient type, calibration)
    3. Users (types, permissions, logins, etc.)
    4. Etc.

2. **Which ones should be secured from change?**
    1. Ex. Log in to change? Physical button?

3. **Restoring a device to a secure state**
    1. Clearing ALL the volatile state to recover from compromise
    2. Ex. Pinhole Reset to factory default

3: Data Protection

Keep your device's data private, unaltered, and erasable

1. **Use proven, industry-standard encryption**
    1. Encrypt and authenticate data in transit; TLS generally covers this
    2. Encrypt private data at rest; verify nothing changed and encrypt sensitive data

    Encryption with AES-128-GCM or ChaCha20-Poly1305
    Signatures with X25519 or secp256r1
    Hashes with SHA-256 or BLAKE2s

2. **Provide a way to erase all sensitive data**
    1. Securely wipe storage and destroy the keys that unlock encrypted data
    2. No one, not even former admins, should be able to recover it

4: Logical Access to Interfaces

Control access to your device's interfaces

1. **Disable interfaces you don't need**
    1. Turn off any local or network interface not required for core functionality
        1. Debug ports, USB ports, unused interfaces (BLE, serial, etc.)
        2. **Web servers**, background processes, functions, etc.
        3. Fewer entry points = smaller attack surface for adversaries

2. **Restrict remaining interfaces to authorized entities only**
    1. Require authentication, login, or physical access to enable
    2. Access could change with state (e.g., no network until properly provisioned)

5: Software Update

Keep your device patched, verified, and rollback-ready

1. **Support secure, authenticated updates**
    1. Remote (network download, ideal) or local update path
        1. Verify and authenticate every update before installing it
        2. Only authorized entities can initiate or approve an update

        Signing proves the update came from you (authenticity)
        Hashing proves it has not been changed (integrity)
        In practice, you sign a hash; see Data Protection for algorithms

2. **Provide rollback and configuration options**
    1. Roll back to a known-good version if an update causes problems
    2. Configure auto vs. manual updates, and enable/disable update notifications

This is how you deal with vulnerabilities in 3rd or 1st party code

6: Cybersecurity State Awareness

Surface your device’s security state to users

1. **Let users see the device's cybersecurity state**
    1. Differentiate normal operation from a degraded or compromised state
        1. Report **firmware version** and alert when it falls behind the latest update
        2. Log events to a persistent record that can’t be tampered with
        Ideally, logging goes to the cloud; for unconnected devices, this can be local
        Logging should include state changes, blocked actions, updates, verification fails etc.

2. **Restrict state and logs to authorized users**
    1. Only logged-in or authorized entities can view the state indicator and logs
    2. Only entities responsible for maintaining the device state can clear it

6 Things

These 6 items are common across almost all standards

They are from:

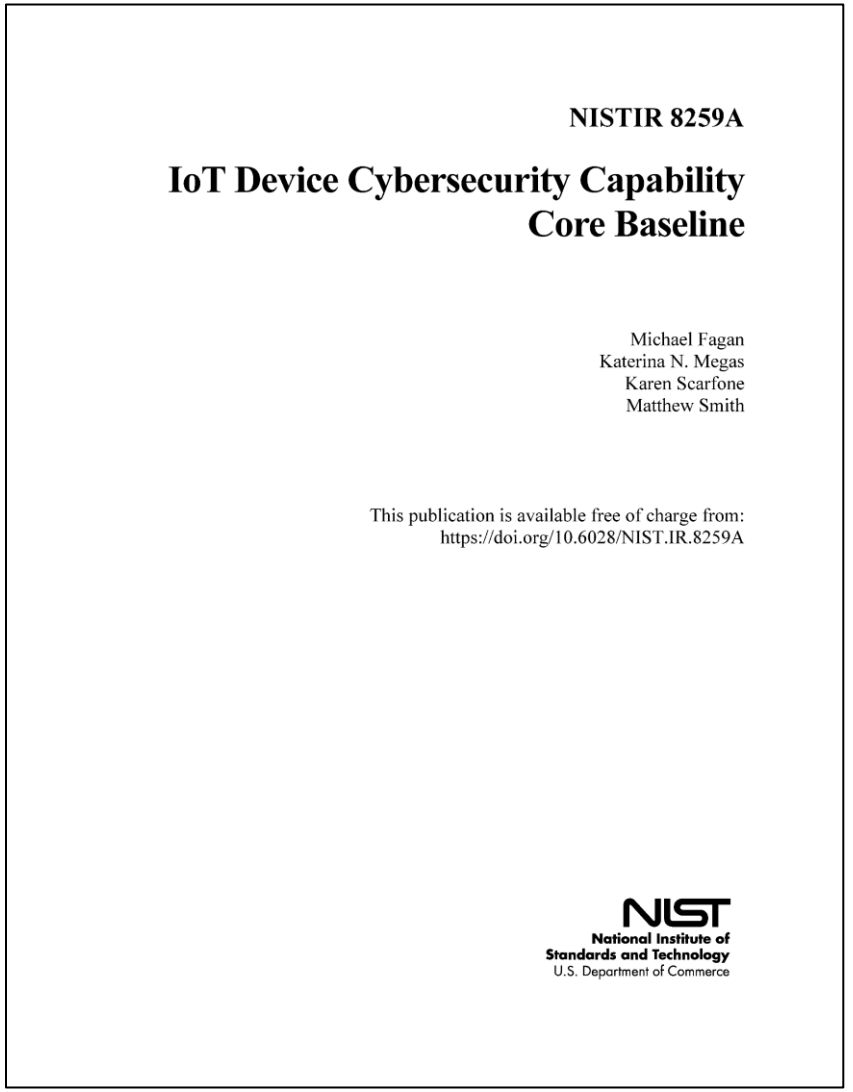
IoT Device Cybersecurity Capability Core Baseline
NISTIR 8259A

The whole standard is free and excellent

- NISTIR 8259 outlines the overview
- NISTIR 8259A is the technical checklist
- **NISTIR 8259B is the nontechnical checklist (post-shipment/maintenance)**

The US Cyber Trust Mark is a derived standard with specific callouts for Consumer IoT devices

- *NIST IR 8425 IoT Core Baseline for Consumer Products*


NISTIR 8259A

IoT Device Cybersecurity Capability Core Baseline

Michael Fagan
Katerina N. Megas
Karen Scarfone
Matthew Smith

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8259A

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Pre-Purchase

Setting expectations & commitments

1. **Publish datasheets defining the device's supported security capabilities**

   A smart lock packaging clearly lists that it uses AES-256 encryption and features hardware anti-tamper alerts

2. **Clearly define maintenance plans, including patch frequency and definitive end-of-life (EOL) dates**

   The product website states, "We guarantee quarterly security firmware updates for this device until December 2028"

3. **Document shared security responsibilities between the manufacturer and the customer**

   A smart baby monitor manual notes: "We are responsible for secure connection to the cloud, but you are responsible for keeping your passwords safe and securing your home Wi-Fi"

Deployment

Enabling secure operation & education

1. **Provide setup guides that teach customers how to configure access controls and identifiers**

   The setup app forces the user to change the default "admin" password to a unique, strong password before the smart thermostat will connect

2. **Document how to properly apply, validate, and roll back software updates**

   The user manual explains that a solid blue LED means "updating" and warns the customer never to unplug the smart speaker during this process

3. **Provide clear instructions for securely reprovisioning or disposing of the device**

   The companion app includes a "Prepare for Sale" button that securely wipes all stored Wi-Fi credentials and account data from the camera

Ongoing Support

Receiving reports & queries (inbound)

1. **Accept vulnerability reports & bug submissions. Customers & researchers need a way to tell you something is wrong**

   Provide a clear channel like a bug bounty program, a security@ email, or a web form

2. **Implement a Coordinated Vulnerability Disclosure (CVD) process**

   Base your process on the CERT Guide to CVD and IoT and CVD guidelines

3. **Offer support for cybersecurity queries and track recurring issues to improve the product**

   Customer service notices 50 users asking which firewall ports the hub needs open; the team tracks this and updates the online FAQ

Incident & Maintenance

Informing customers proactively (outbound)

1. **Maintain communication lists to proactively reach your customers**

   The smart TV requires users to register an email address during setup specifically so the manufacturer can send emergency security bulletins

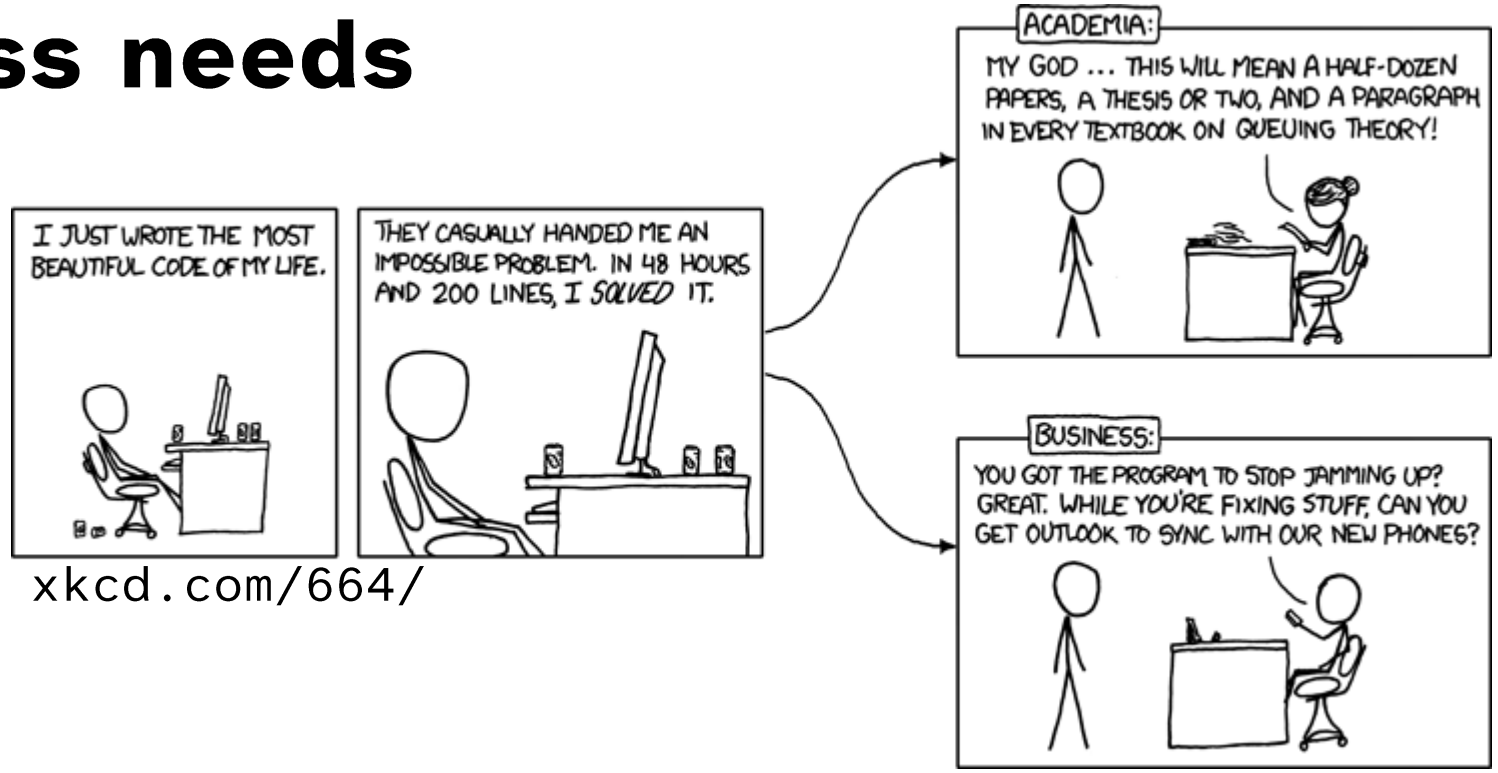
2. **Disclose new vulnerability discoveries and issue security advisories with mitigation steps**

   The company emails all users saying, "A vulnerability was found in the Bluetooth pairing. Please disable Bluetooth in settings until the patch is released"

3. **Alert customers to new software update availability and remind them of upcoming EOL dates**

   The companion mobile app sends a high-priority push notification stating: "Critical security update available. Tap here to install patch now"

6 Business needs



xkcd.com/664/

Cybersecurity Regulation

Cybersecurity regulations across industries and the world force compliance

Regional:

- **Cyber Resilience Act** (CRA) – **EU** (Dec 2027)
- **Product Security and Telecommunication Infrastructure** (PSTI) – **UK**
- **Cyber Trust Mark** – **US** (Jan 2027)

Industry Specific:

- **Medical Devices** (ex. FDA Cybersecurity Premarket/Postmarket)
- **Motor Vehicles** (ex. NHTSA Cybersecurity Best Practices/SAE 21434)
- **Aviation** (ex. FAA/EASA DO-326A)

Bad cybersecurity causes real exposure

Abbott RF pacemakers

Could have been solved by Update / Patch

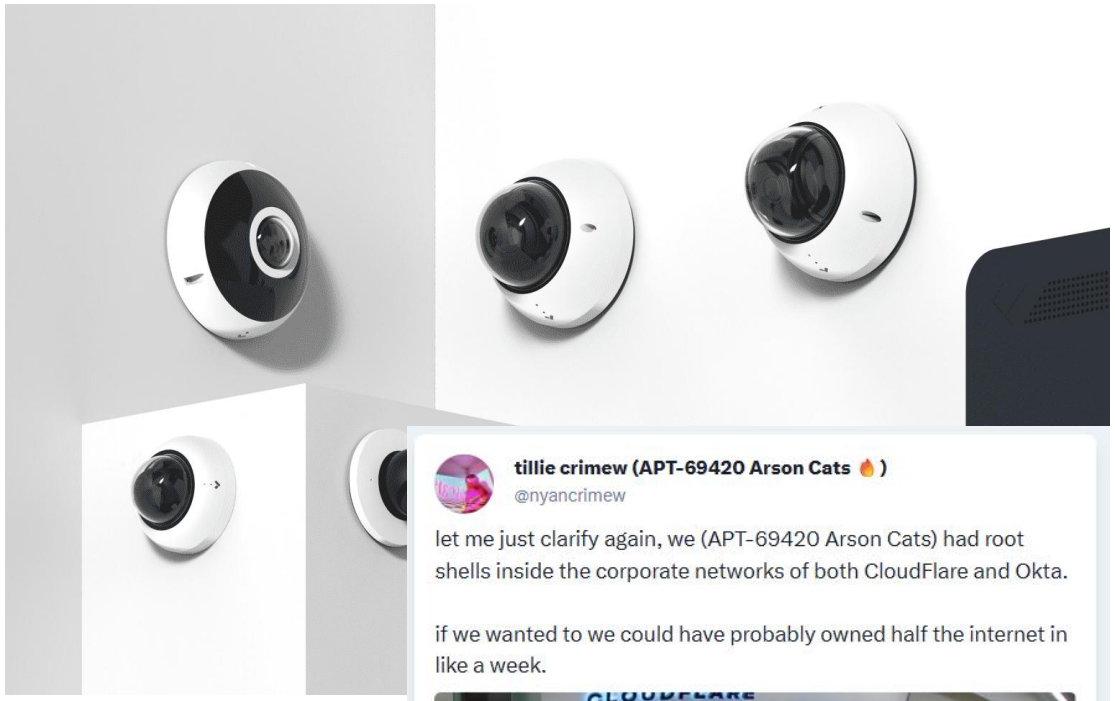
- Shipped without secure-update; Vulnerabilities surfaced

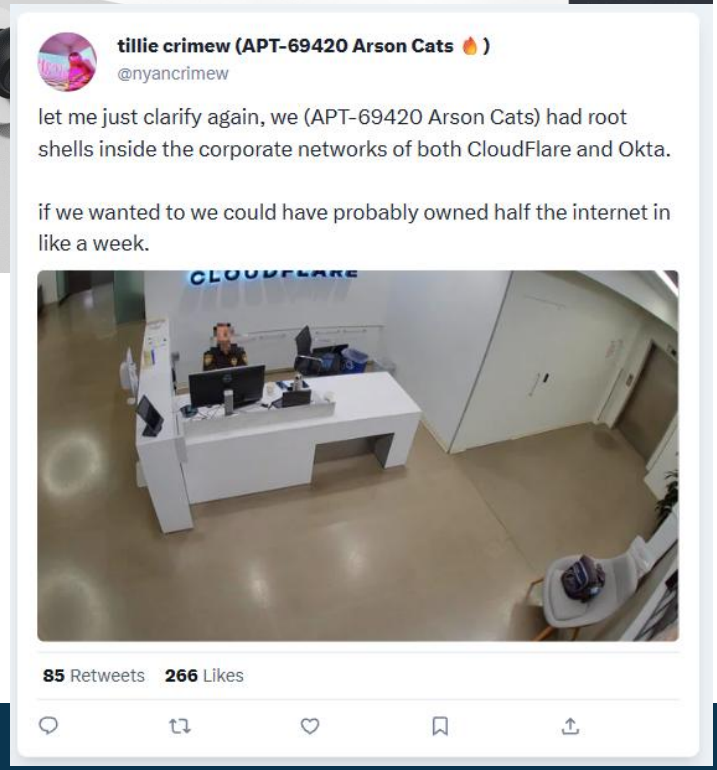
- 465,000 **implanted pacemakers** had to be updated **while patients were on a cardiac monitor**
  - 853 incomplete updates
  - 122 losses of programmed settings
  - 16 devices with a complete loss of functionality

Verkada's camera cloud

Logging/Intrusion Detection

- Attackers accessed more than 150,000 cameras
- **Lack of anomaly alerts** meant it was noticed only **after screenshots hit Twitter**





Medtronic MiniMed insulin pump

Authentication

- Remote pumps had an unauthenticated RF interface
- Attackers could silently overdose insulin
- FDA recall with 64,000 units affected

Contec CMS8000

Access Control

- Network users permissions were not locked down allowed PHI export and remote control
- Subject of urgent coordinated CISA and FDA disclosure

Hikvision IP cameras

Integrity

- A hidden “magic string” bypassed authentication and firmware signing
- Hundreds of thousands of cameras were hijacked for botnets until owners manually flashed fixed firmware; many remained vulnerable years later.

THANK YOU

Embedded
Online
Conference

Marcus Engineering LLC

www.embeddedonlineconference.com

momo@MarcusEngineering.com