Tower defense for hackers: Layered (in-)security for microcontrollers

Milosch Meriac
meriac.com
@FoolsDelight

hardwear.io

2

ABOUT
ME

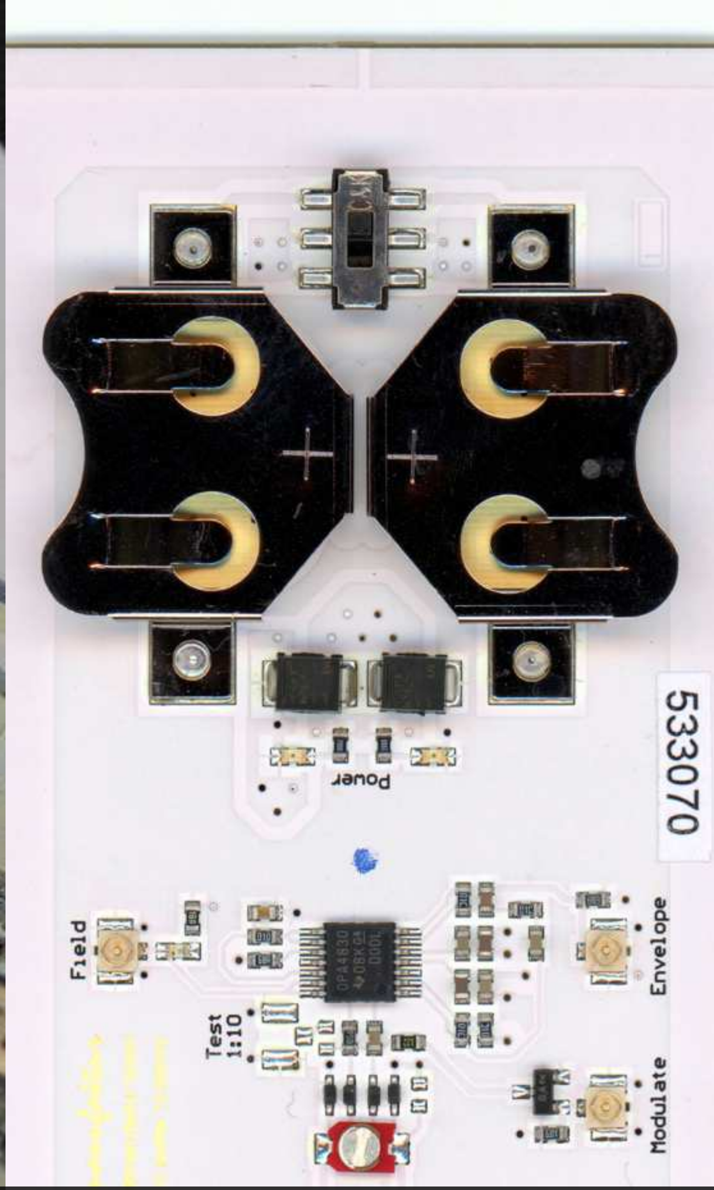
My Open Software & Hardware Projects

SHARED FUN IS TWICE THE FUN

I created a range of open hardware designs and software tools around RF(ID)/BLE security research and electronic art projects. You can find a more information on my work at meriac.com

OpenPCD.org

Open 13.56 MHz
RFID NFC Reader



OpenPICC.org

Open 13.56 MHz
RFID Sniffer Design
with HID iClass
decoder

OpenBeacon.org

Realtime 2.4GHz
Localization & Human
Interaction Analysis

Blinkenlights Stereoscope

960 x Realtime 2.4GHz
Wireless Halogen Dimmers
for Toronto City Hall

Xbox Linux Core Team

Breaking the first serious computing
platform for consumers

ABOUT
ME

My project at ARM Ltd.

I am working on the ARMmbed uVisor – a security-enabled hypervisor for MMU-less microcontrollers targeting IoT devices like wireless sensor nodes.
An early alpha-version of the uVisor is available under Apache License on github.

Principal Security Engineer

Working in the IoT Unit
on ARM mbed in Cambridge, UK

5

INEVITABILITY

Security
+ Time
= Comedy

IF A PRODUCT IS SUCCESSFUL,
IT WILL BE HACKED

Who can't suppress snickering on seemingly dumb security bugs in other peoples product?
It's easy and delightful to declare developers of failed products as incompetent.
As old stuff has old bugs – we won't run out of entertainment anytime soon.

DEVICE LIFETIME

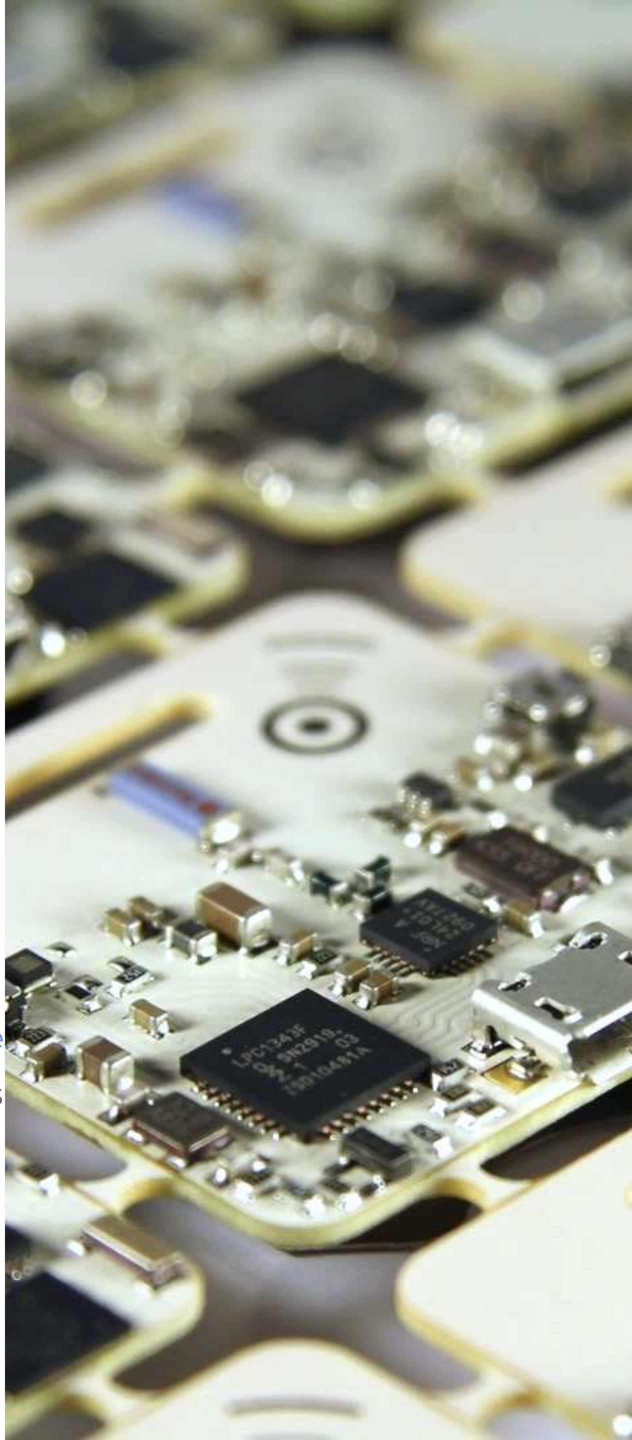
The security of a system is dynamic over its lifetime. Lifetimes of home automation nodes can be 10+ years.

ATTACKS SCALE WELL

The assumption of being hacked at some point requires a solid mitigation strategy for simple, reliable and inexpensive updates.
Do our defenses scale equally well as our attacker do?

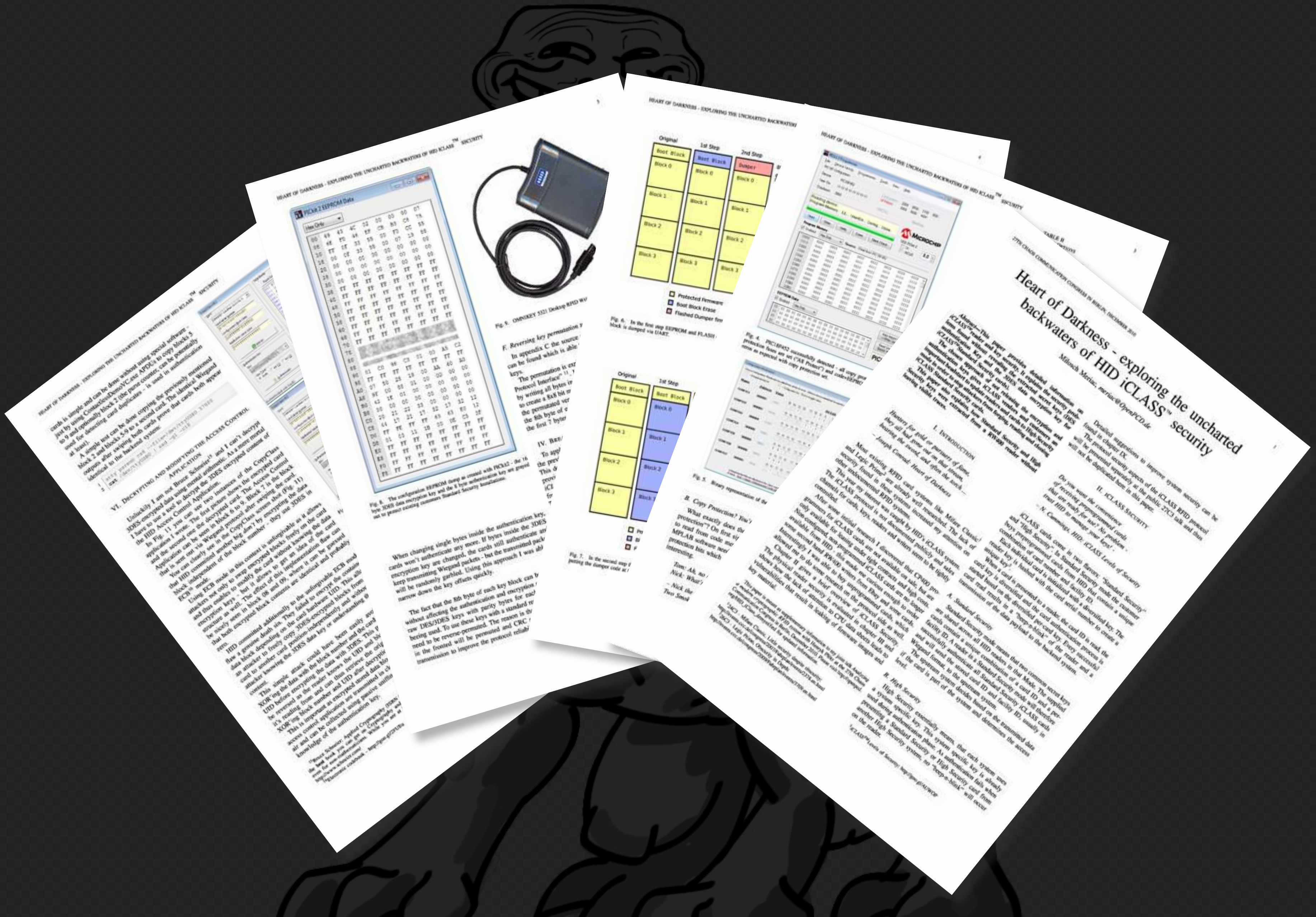
YOU CAN'T STOP IT

Value of bugs is expressed by value = number_of_installations x device_value
Increased value and large deployments drive attackers - especially in the IoT.
Massively parallelized security researchers/attackers vs. limited product development budgets and time frames.



It’s insane fun to be a security troll.

BEEN THERE, DONE THAT!



My favourite: "Heart of Darkness - exploring the uncharted backwaters of HID iCLASS™ security"

If we believe that security requires a sound architecture from the start, we must stop trolling the result, and start trolling the architecture.

BE A GOOD CITIZEN!
SHOW THEM HOW TO DO IT RIGHT
CREATE BEST-PRACTICE IoT SOLUTIONS RUNNING ON UNTRUSTED CLOUD SYSTEMS
AND EXERCISE END-TO-END ENCRYPTION

Why is Microcontroller Security so hard?

MICROCONTROLLERS ARE DIFFERENT

The ugly truth™ is that makers must find all flaws – attackers only have to find one.

BREAKING A SYSTEM IS EASY.
FIXING A SYSTEM IS HARD.

Security from the 80's for today's threats

MMU-LESS ARCHITECTURES

Flat memory models enable escalation & persistence of bugs.

LIMITED COMPUTING POWER & MEMORY

Vendors commonly use shortcuts to scale public key cryptography down to microcontroller memory limitations..

RANDOM NUMBER GENERATION

New wireless microcontrollers have HRNGs. Existing systems either don't have them or don't use them. PRNGs are hard to get right and make promising targets..

INTERNAL STORAGE

Reliable and tamper-proof storage of data is a core security requirement for most security systems. It's very hard to get internal storage right when assuming local attackers.

“It ain’t what you don’t know that gets you into trouble. It’s what you know for sure that just ain’t.”

MARK TWAIN

WRITER, ENTERPRENEUR & PUBLISHER

MMU-LESS ARCHITECTURES

13

Flat memory models

THE 80'S CALLED, THEY WANT THEIR SECURITY MODEL BACK

A flat address spaces as the result of lack of a memory management units (MMU) does not justify the absence of security. Many microcontrollers like ARM Cortex-M3/M4 provide a hardware memory protection unit (MPU) instead.

NO SEPARATION

Flat memory models and ignorance of MPUs blocks vital security models like “least privilege”.

ESCALATION

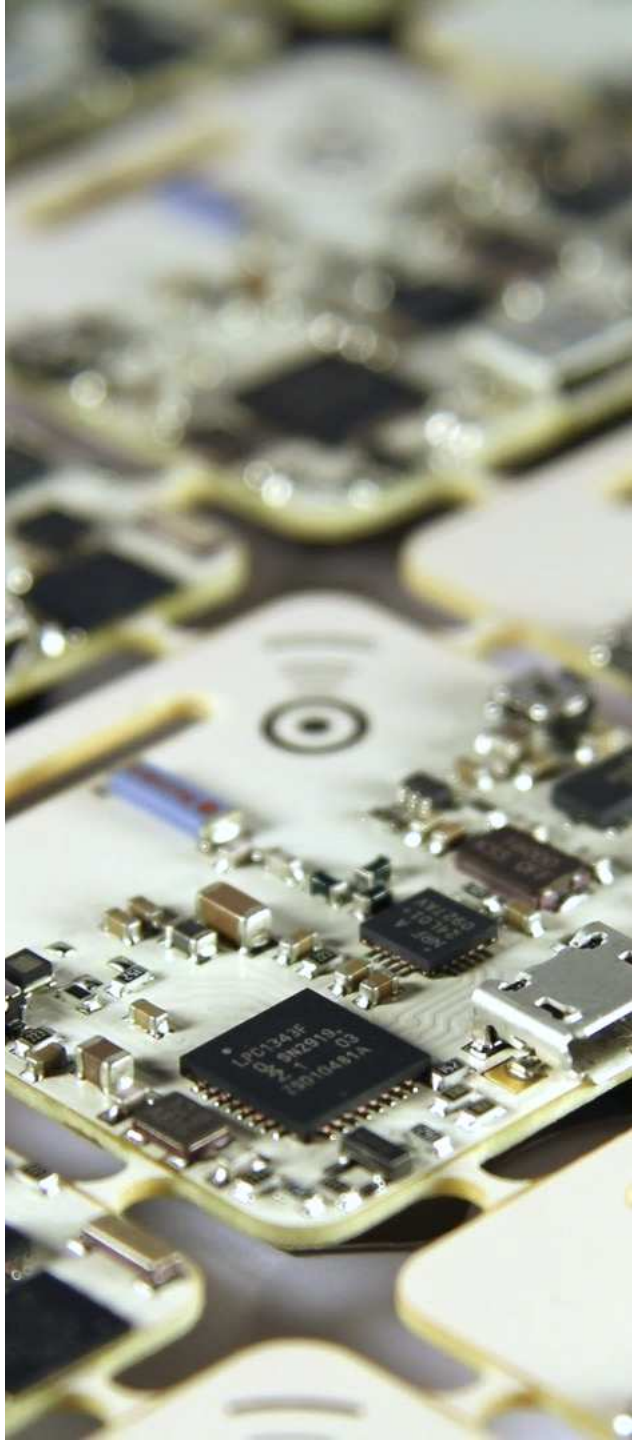
Flat memory models enable escalation & persistence of bugs by uncontrolled writing to flash memories.

VERIFICATION

Security verification impossible due to the immense attack surface and lack of secure interfaces between system components.

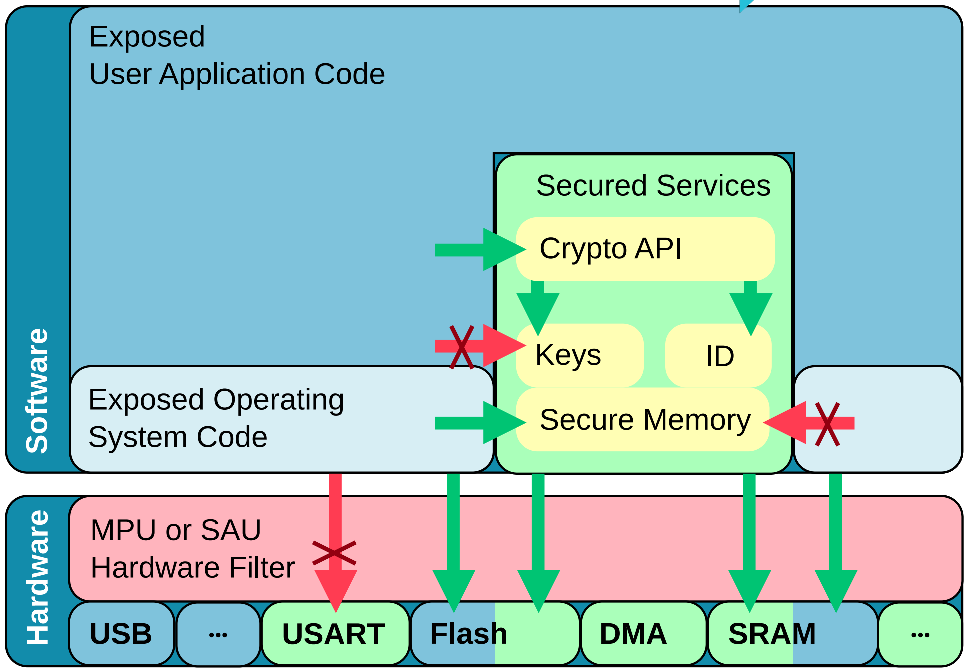
LEAKAGE

All your bases are belong to us – thanks to leakage of device secrets like keys



Example: uVisor for microcontrollers

- Hypervisor with hardware-enforced security sandboxes using MPU virtualization – no MMU needed.
- Targeting ARM Cortex-M3/M4 microcontrollers
- Apache Licensed github project in development – integrated with ARM mbed and Keil RTX, (also Apache-licensed)
- Mutually distrustful security model:
  - Principle of Least Privilege
  - Boxes are protected against each other and drivers
  - Enforces API entry points across boxes
  - Box-API functionality can be restricted to specific boxes: “Box caller ID”
  - Per-box access control lists (ACL)
    - Restrict access to selected peripherals like Flash to avoid malware persistence
  - Remote Procedure Call API (RPC) for secure box-box calls



LIMITED COMPUTING POWER & MEMORY

15

Resources matter

ABSTRACTIONS ARE EXPENSIVE

Point-solutions are easy to build for microcontrollers, but hard to maintain and to reason about. Abstractions are hard to build, but enable to amortize security verification across a large range of devices. For crypto API’s this is understood now – but not yet for other system components.

PUBLIC KEY CRYPTO

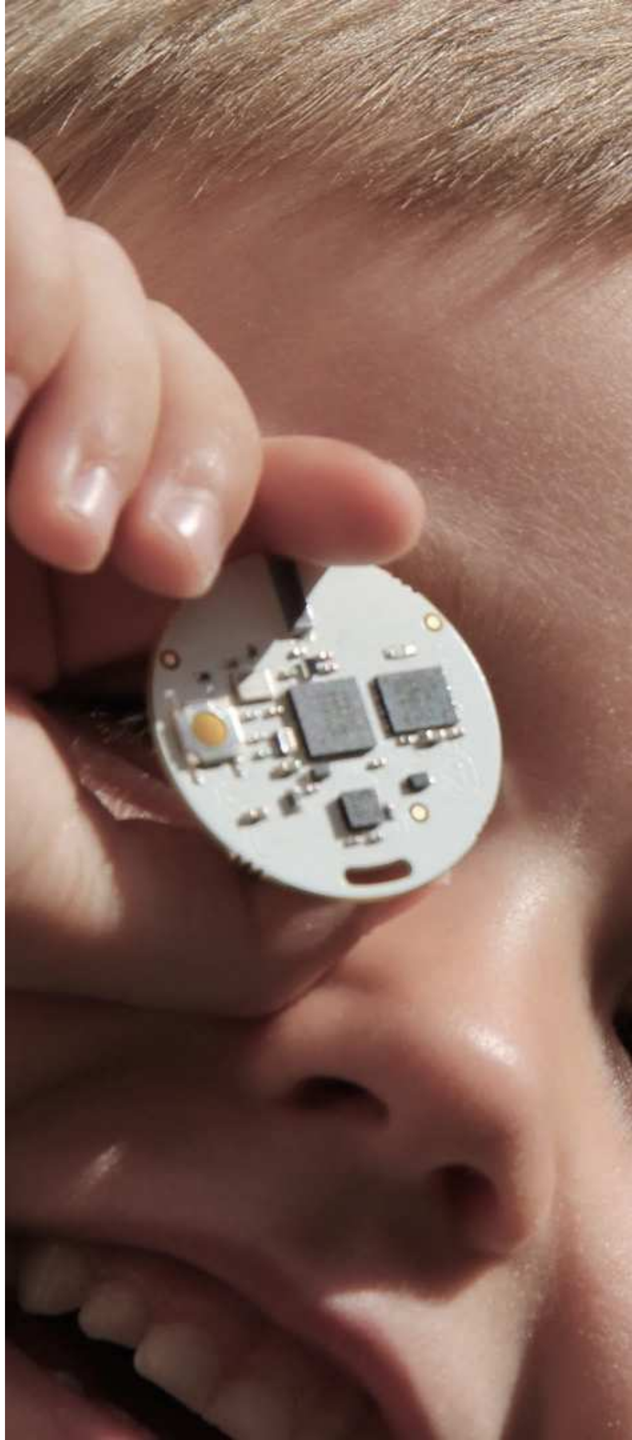
Public-key crypto is absent from many low end devices due to memory and speed constraints.

SHORTCUTS

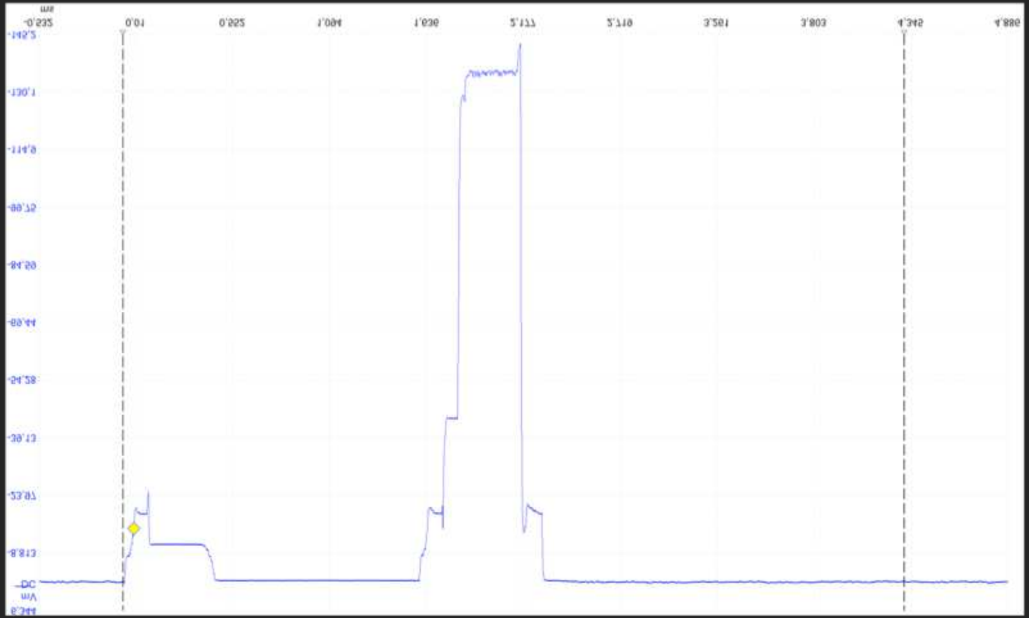
Vendors take compromises on security to reduce footprint. Think for example of commonplace class key usage versus supporting key provisioning and per-device secrets..

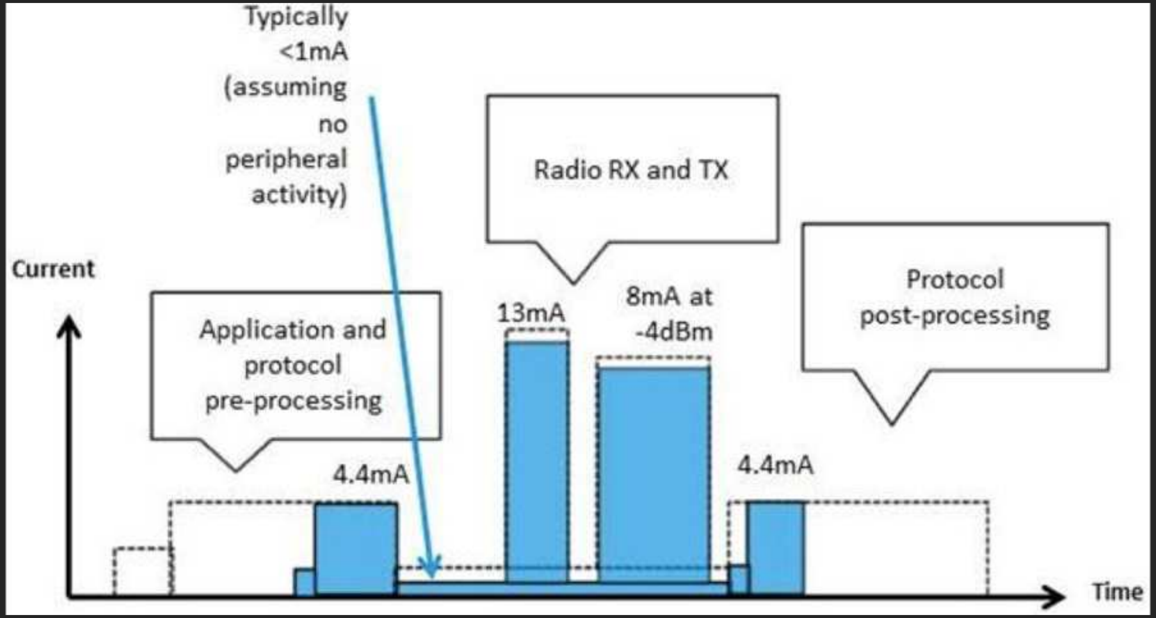
COMMUNICATION

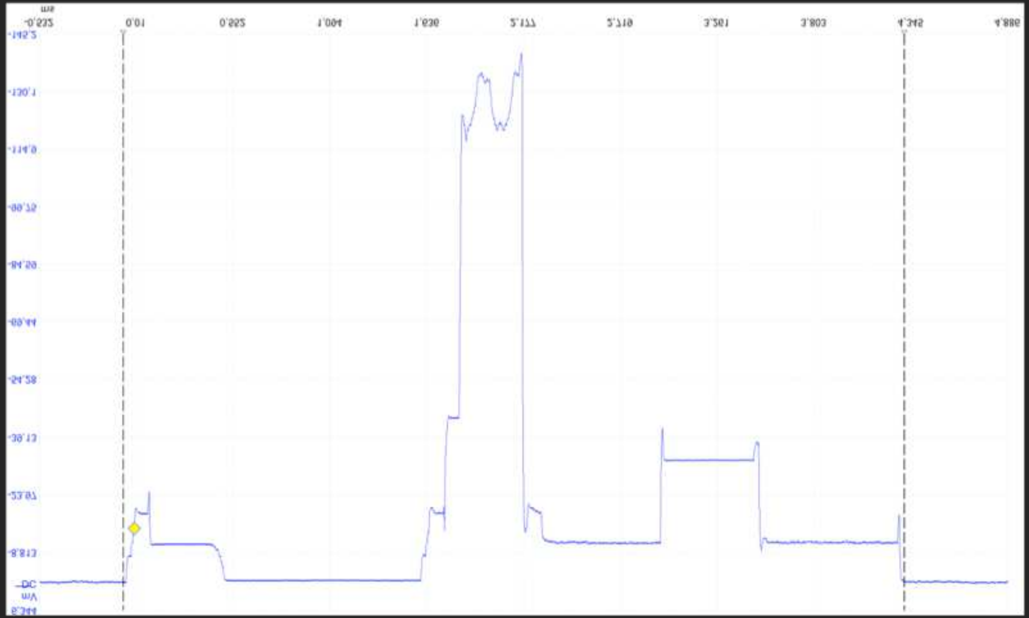
Less abstraction requires more communication on limitations. This usually results in higher level components having wrong assumptions about low level components. Think of flash config writing in the presence of a local attacker. Be scared.



Device power consumption: The perfect tool for understanding device operation







17

RANDOM NUMBER GENERATION

Random, or not?

APPLICATION DEVELOPERS AND SECURITY

Developers have trouble understanding the requirements for randomness - resulting in bad design choices.
Local attackers can often control all external entropy sources or storage and reset the device at will.

TIME IS NOT RANDOM

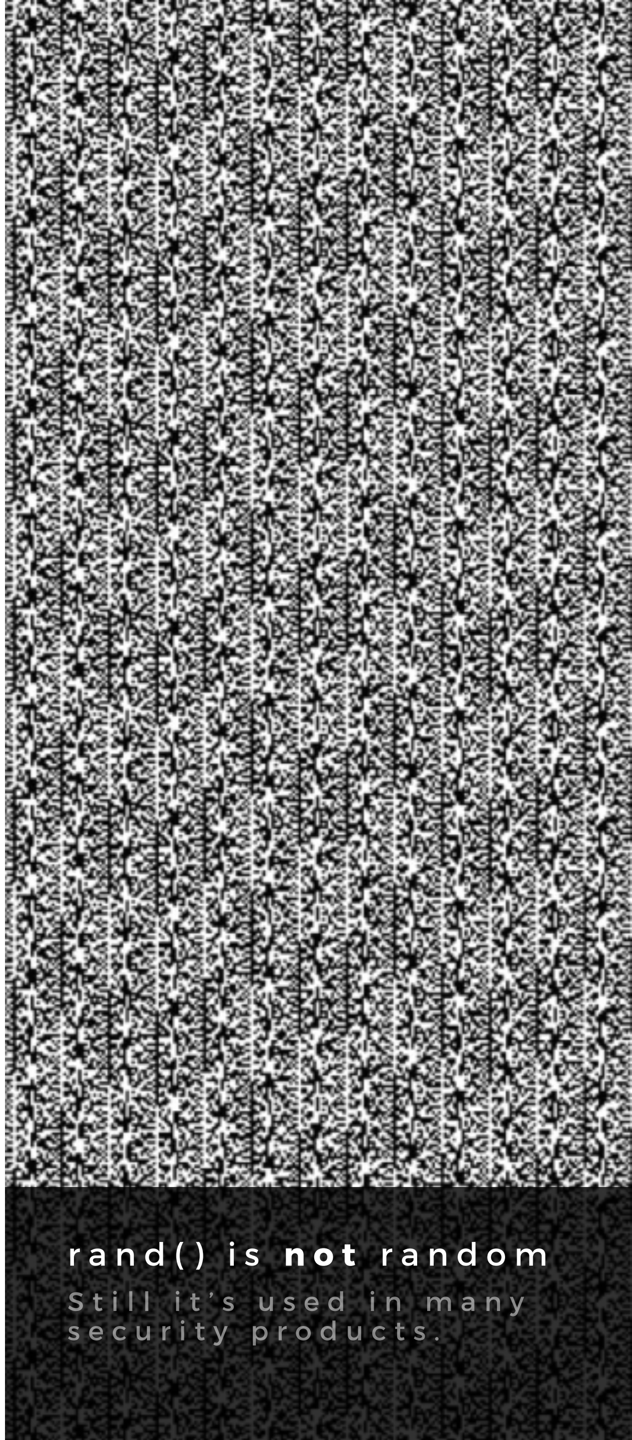
Assume that the attacker can issue requests at precise times after booting your microcontroller and fully control your clock sources..

PRNG vs. TRNG

Lack of a true random number generators microcontrollers is often used as an excuse for security flaws. Secure pseudorandom numbers are seldom implemented correctly.

PRNG REQUIREMENTS

PRNGs require both a device-specific-random seed during manufacturing and a non-volatile pool that can be updated regularly and not reset by local attackers.



```
uint8_t key[104]; // pointless length
srand(time(NULL));
for (int i = 0; i < 104; i++) {
        key[i] = rand() & 0xff;
}
```

CODE FROM A DATABASE APPLICATION USED
BY THE GERMAN GOVERNMENT
FOR SECURITY AUDIT MANAGEMENT

Courtesy [30C3 Talk] "Reverse engineering of CHIASMUS from GSTOOL"

CODE & DATA STORAGE

Storage, seriously?

YOUR SECURITY ASSUMPTIONS ARE PROBABLY BROKEN

Most high-end application/mobile CPUs have follow standardized security models for securing code and data. Microcontroller storage security is still highly diverse and usually started off as code-read protection. Yet, your keys and firmware updates need to be secured against tampering and rollback. Malware should not be allowed to become persistent by installing itself into flash.

OUT OF MEMORY

As functionality and wireless stacks grow, microcontrollers run out code memory – thus firmware updates must to be buffered externally to avoid bricking,

These mechanisms are commonly flawed in the presence of local attackers controlling power or data.

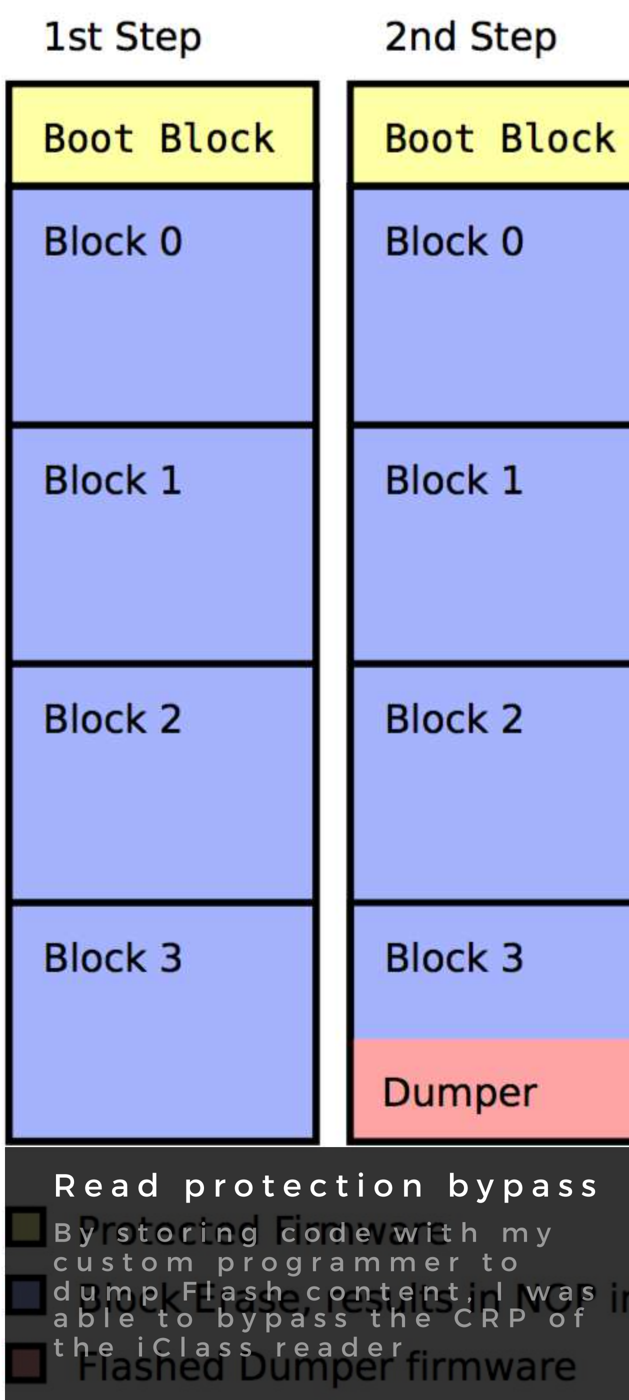
DATA SECURE, TOO?

How confident are you, that your SRAM is erased when resetting the copy protection fuses on your microcontrollers?

Also data is often [extracted indirectly]() using CPU core registers and [stepping through existing code]().

SIDE CHANNELS

Microcontrollers often vulnerable to extraction via side channels like power.

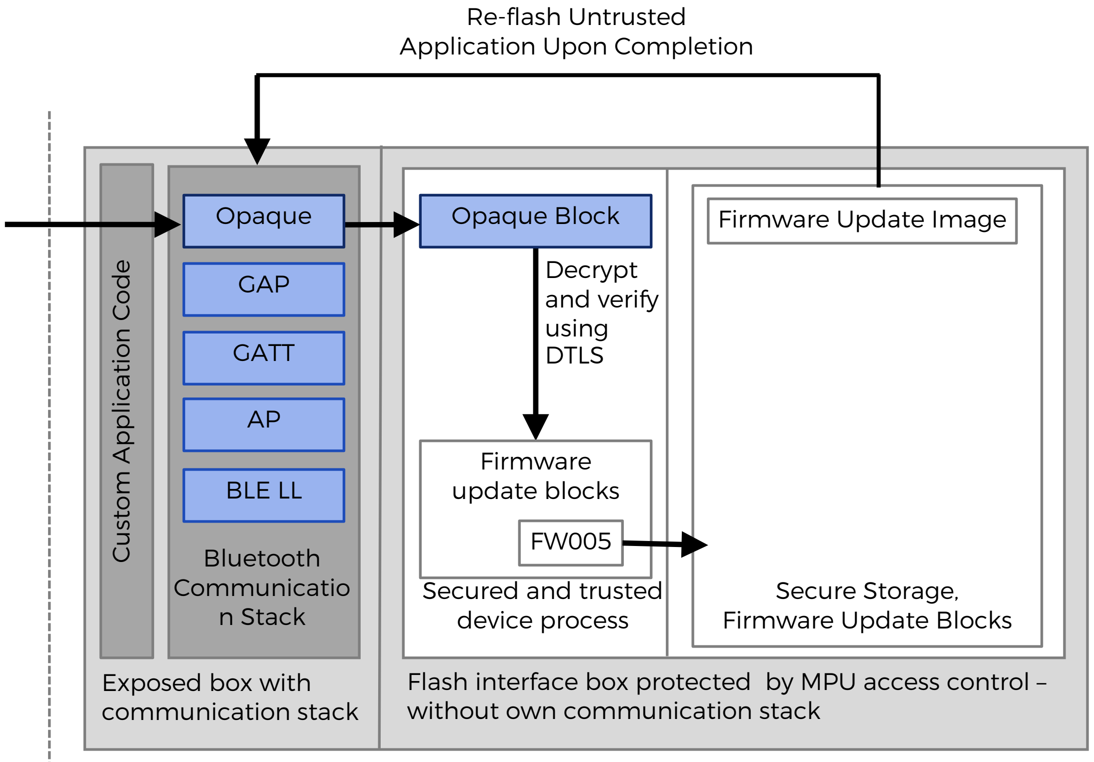


Read protection bypass

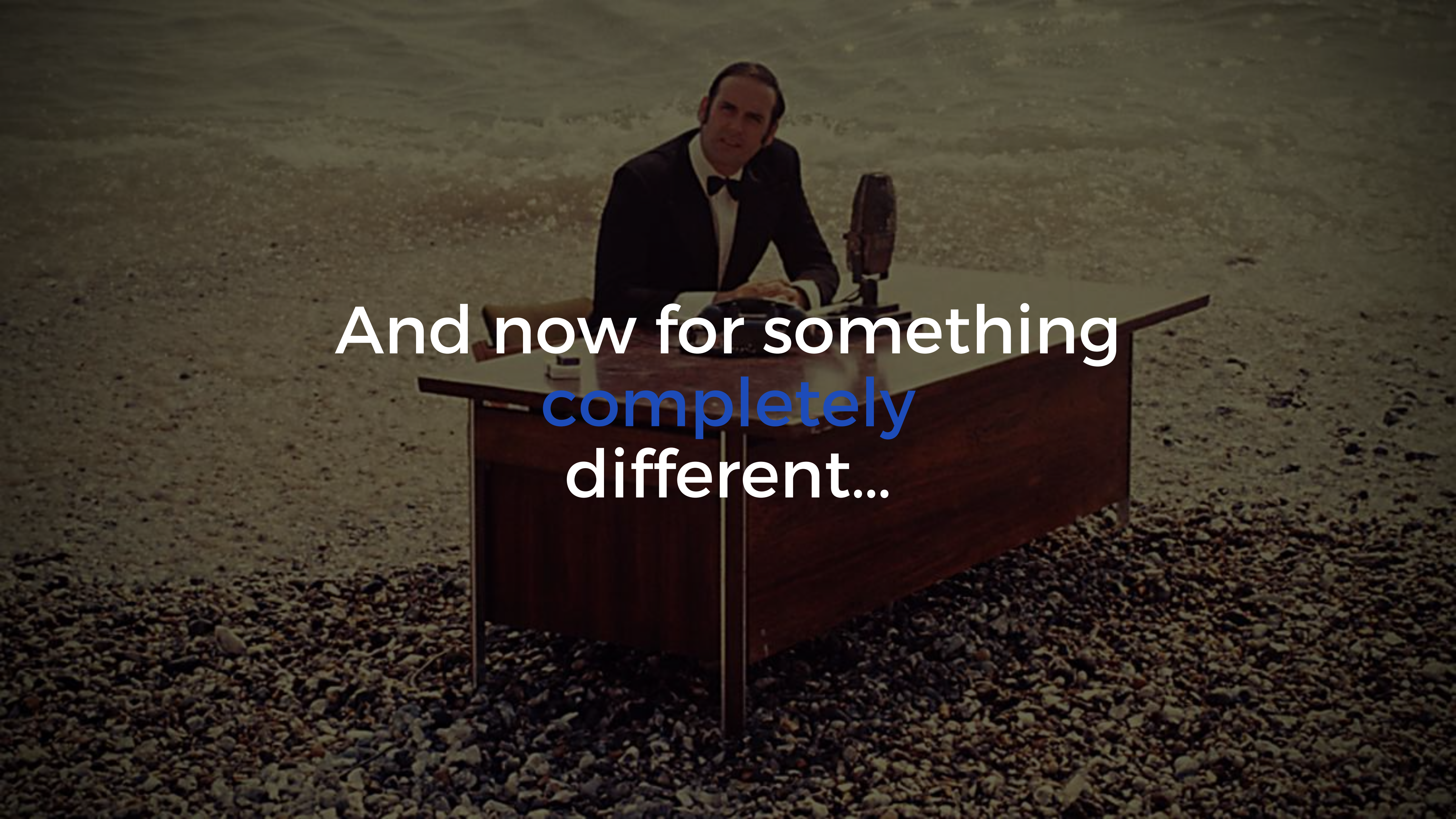
By storing code with my custom programmer to dump Flash content, I was able to bypass the CRP of the iClass reader

Case Study: Secure Firmware Update

- Flash access is exclusive to the firmware update core service.
- Using the MPU for blocking access to the flash controller to everybody but the firmware update service.
- Malware is forced to use APIs to attempt writing to flash
  - Public Key signatures of the device owner or manufacturer are required for API to accept an update.
  - Firmware is downloaded piece by piece into secure storage. The system reboots after initial verification into a boot loader for copying the new firmware into its actual position in internal flash.
  - The internal firmware is activated after final verification.
- Crypto watchdog box enforces remote updates even for infected devices as only the server can re-trigger the watchdog with its cryptographic secret.

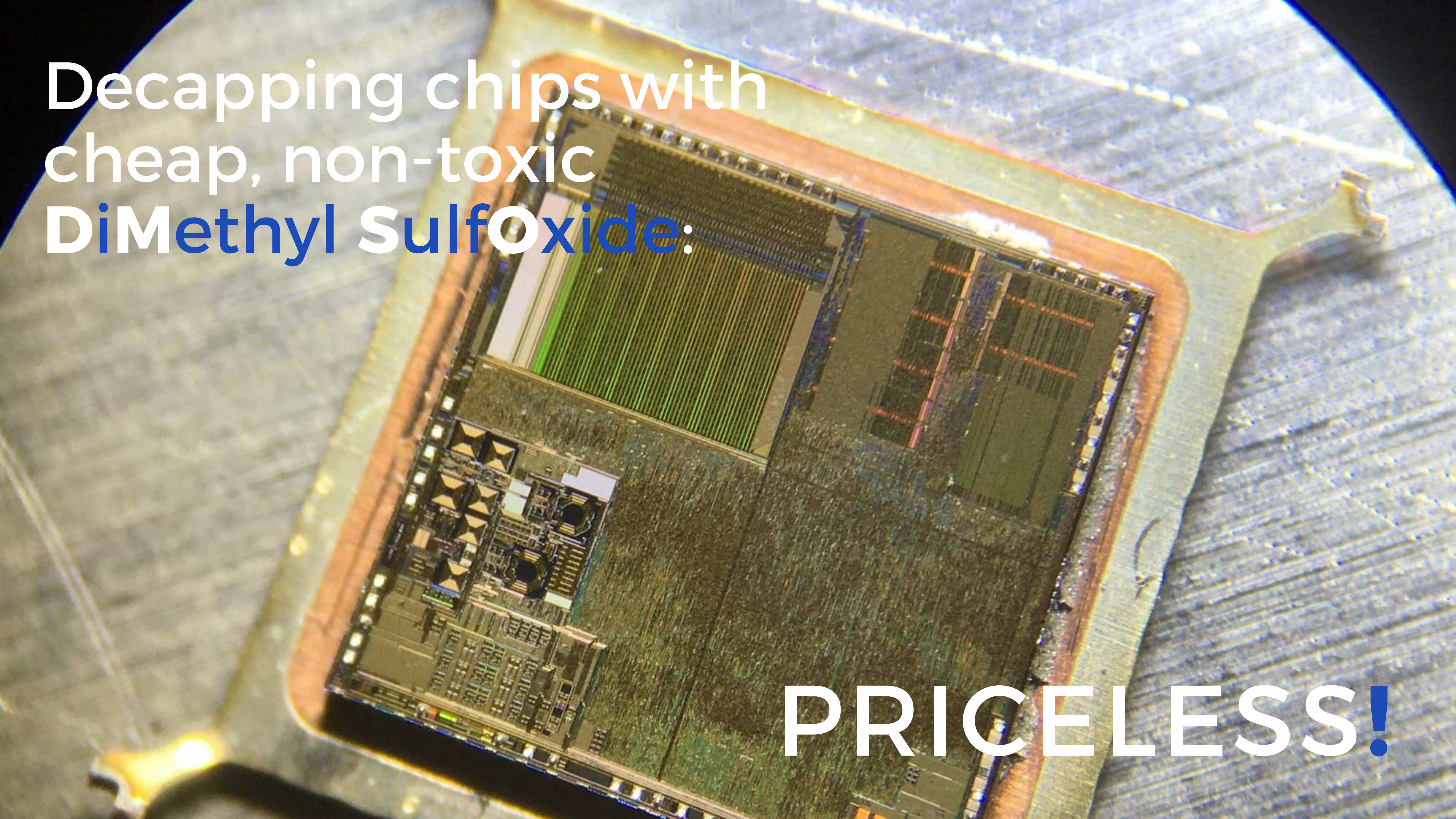


, Messages delivered independently of communication stacks


And now for something completely different...

An 180˚C PTC heater from AliExpress: $4


… taped to a ceramic plate
with Kapton tape …
… and a superglued
screw-cap: $5


Decapping chips with
cheap, non-toxic
DiMethyl SulfOxide:
PRICELESS!




Keep on trollin'
Keep on breakin'
One fine day you'll gonna be the one
To make us understand
Oh yeah

SONG BY THE SPENCER DAVIS GROUP

KIND-OF,
SO LONG AND THANKS FOR ALL THE FISH!

THANKS!