EURECOM
Sophia Antipolis

Embedded Systems Security

Aurélien Francillon

Secappdev
Feb 2015

Aurélien Francillon

@aurelsec

Previously

- PhD from INRIA (Grenoble)
- *Postdoc* ETH Zürich

Assistant professor at EURECOM

- Since 2011

Eurecom

- Petite école d'ingénieurs a Sophia Antipolis
- Membre de l'institut mines-télécom

2/3 d'étudiants étrangers

A un nouveau diplôme cette année

- Diplôme d'ingénieur de spécialisation (CTI)

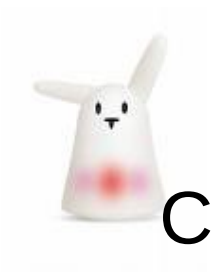
Embedded Systems Security

This is a very wide topic, what we will see today

- Basics in embedded systems security, hardware attacks
- Example of an embedded system software compromise
- How to analyze such systems
  - Example of static and dynamic analysis

Embedded devices diversity

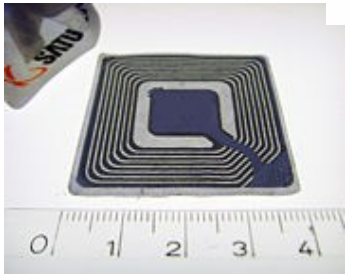
SmartCards



Connected devices

Sensors

RFID



Industrial systems

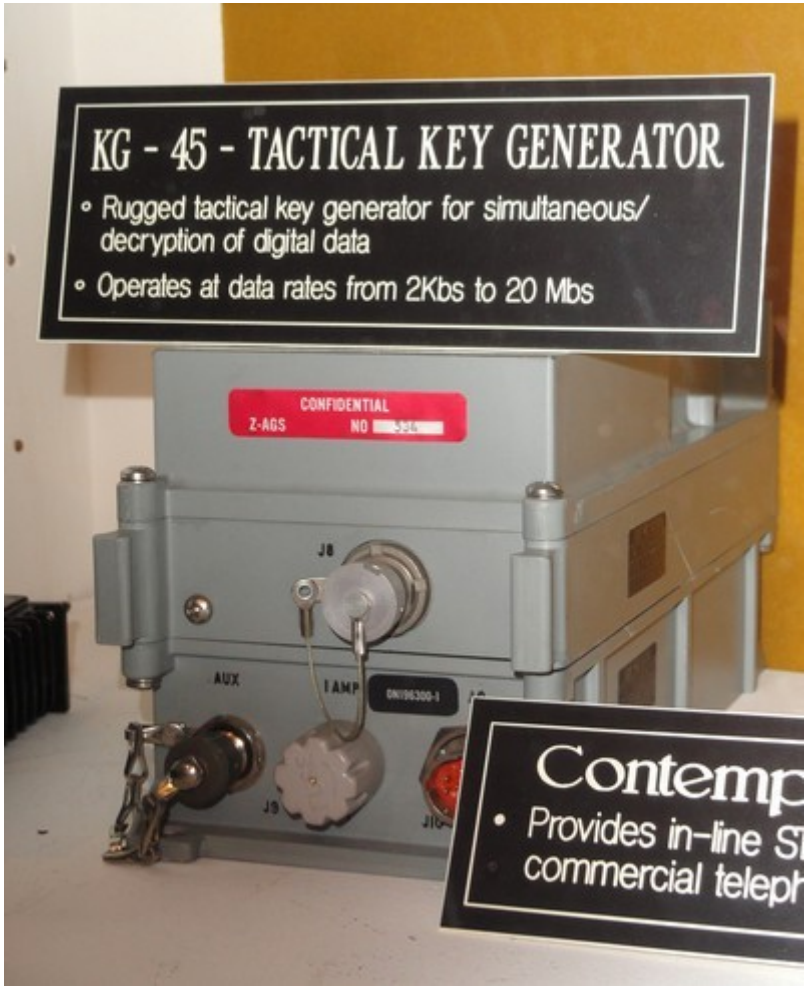
There are *security* devices

Devices that are designed for a security application

- Well defined purpose, often custom
- Limited attack surface
- High level of security achievable
- Not (too) cost sensitive, volume production

There are *security* devices

- We know how to do security for a while…



There are 'just' devices

Those embedded devices are everywhere

- Not security devices, security not their main purpose
- Often COTS devices

Sometimes availability and/or safety is important

- But often no real need for security
- Lack of a clearly identified adversary
- Resistant to random faults
- Attacker will trigger improbable conditions

Your computer is made of many “computers”

Keyboard controller

Network card

CPU microcode

Hard-drive

Northbridge chipset

GPU

All of them are considered to be in our TCB.
What could happen when one is compromised?

Internet of what ?

MIT Technology Review
BUSINESS REPORT

The Internet of Things

Billions of computers that can sense and communicate from anywhere are coming online. What will it mean for business?

CONTENTS

A single match for “security”

George Arnold

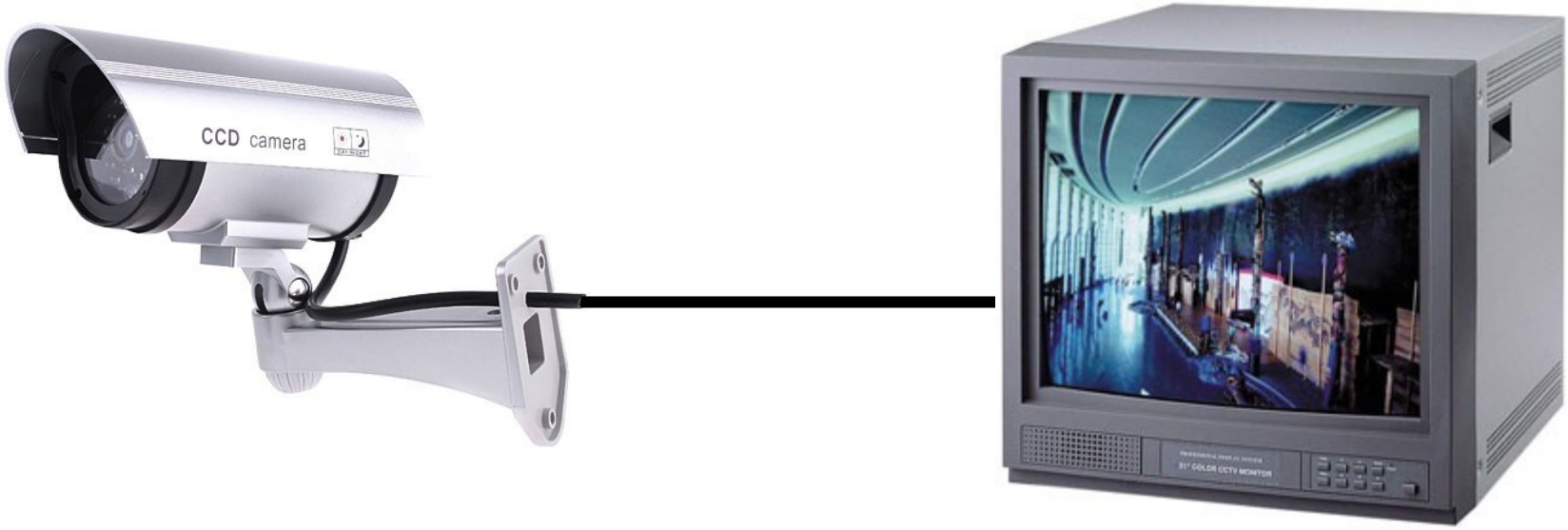
Director, Standards Coordination Office, and national coordinator, smart grid interoperability, National Institute of Standards and Technology
Gaithersburg, Maryland

On the Internet of things, standards and protocols will determine winners and losers. That makes George Arnold a quietly important player. As a smart-grid czar at the federal National Institute of Standards and Technology, Arnold is involved in setting standards for how digital technologies will affect the politically and strategically important electricity grid and other “cyber physical” systems. As part of an informal industry group known as the Kitchen Cabinet, Arnold was influential in establishing the commercial Industrial Internet Consortium this year. NIST’s 2014 budget request included $18.8 million to study cyber-physical systems and their security. Arnold is a veteran of Bell Labs and the coauthor of one of the earliest examples of chess-playing software.

Is there really a problem with security in IoT?

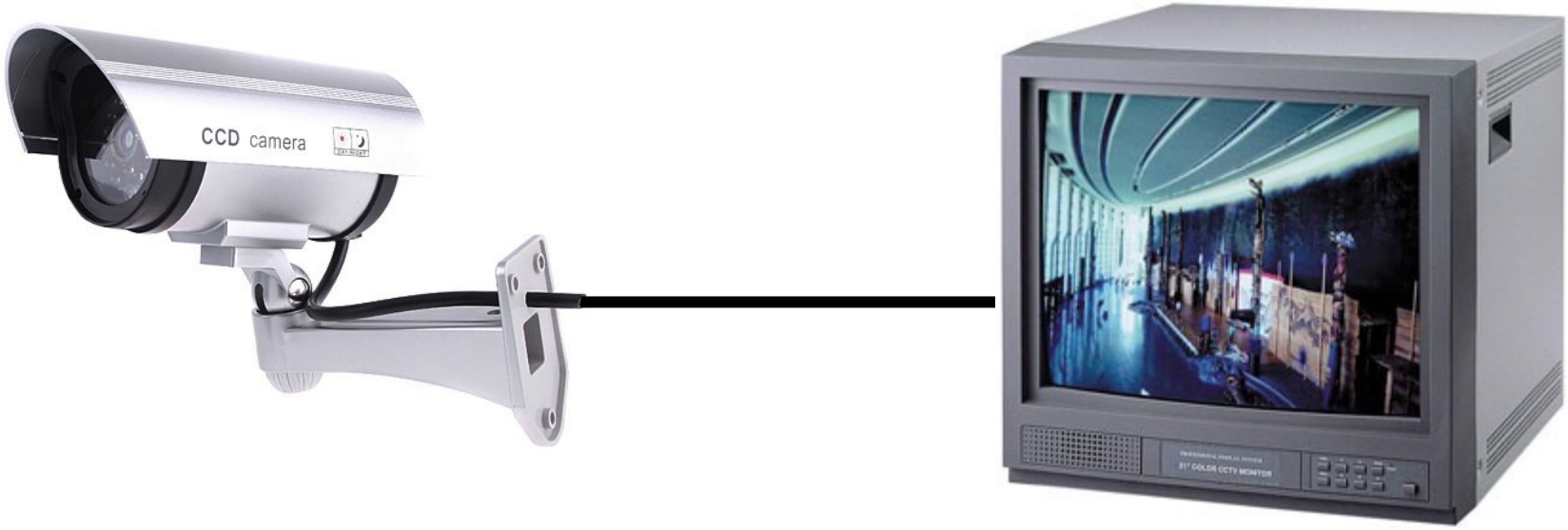
Before IoT

- Video Protection



Before IoT

- Video ~~Protection~~ Surveillance



IoT: composition

Composition Kills

Slide Courtesy of T. Goodspeed

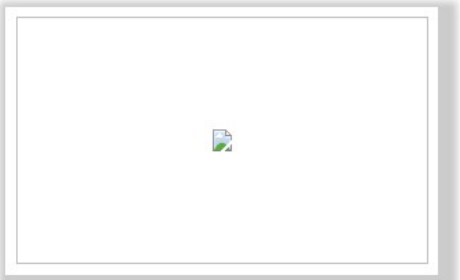
[redacted]

World CCTV cameras online | AvTech DVRs | Foscam cameras | Hikvision DVRs | Panasonic cameras | Linksys cameras | IPCamera can

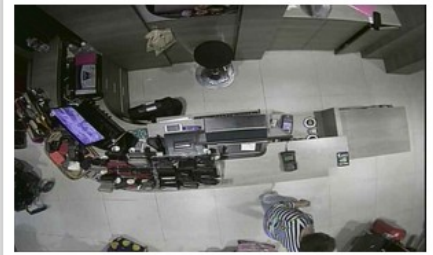
- United States (9480)
- Korea, Republic Of (4730)
- China (2543)
- France (2440)
- Netherlands (2186)
- Italy (2159)
- Mexico (1893)
- United Kingdom (1768)
- Colombia (1755)
- India (1391)
- Indonesia (1190)
- Turkey (1044)
- Argentina (1007)
- Hong Kong (961)
- Japan (940)
- Canada (861)
- Brazil (841)
- Romania (759)
- Poland (757)
- Spain (755)
- Australia (726)

IP cameras: France

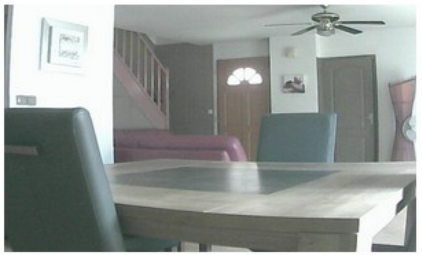
page 222 of 407.



FR, NICE
2 ch.



FR, NICE
8 ch.



FR, NICE
1 ch.

FR, NICE
1 ch.

FR, NICE
1 ch.

We just “I”-ified a camera

What happened?

- Attack surface increased
- Devices not designed for security now online
- Threat model changed
- Value increased just by being online
- Deploy and forget

Challenges

Challenges

Embedded devices not immune anymore to compromise

- Security devices are rather secure
- But we rely on *non security devices* for our daily security and privacy

But embedded devices are obscure, hidden

- You know the OS you are running on your server/laptop and if it is up to date
- But when was the last update of your “fridge’s” OS?

Challenges

Networking/wireless interfaces are increasing the attack surface

- IoT=> IoIT? Is it the *Internet of the Insecure Things ?*
- Remote, software attacks become possible

In general difficult to know the runtime status of a device

Sophisticated attacks previously demonstrated

- Hundreds of thousands of devices compromised

So what can we do about this ?

Good design
- Security Development Lifecycle (SDL)
- Hardening devices

Raise public awareness
- Evaluate the impact of compromises

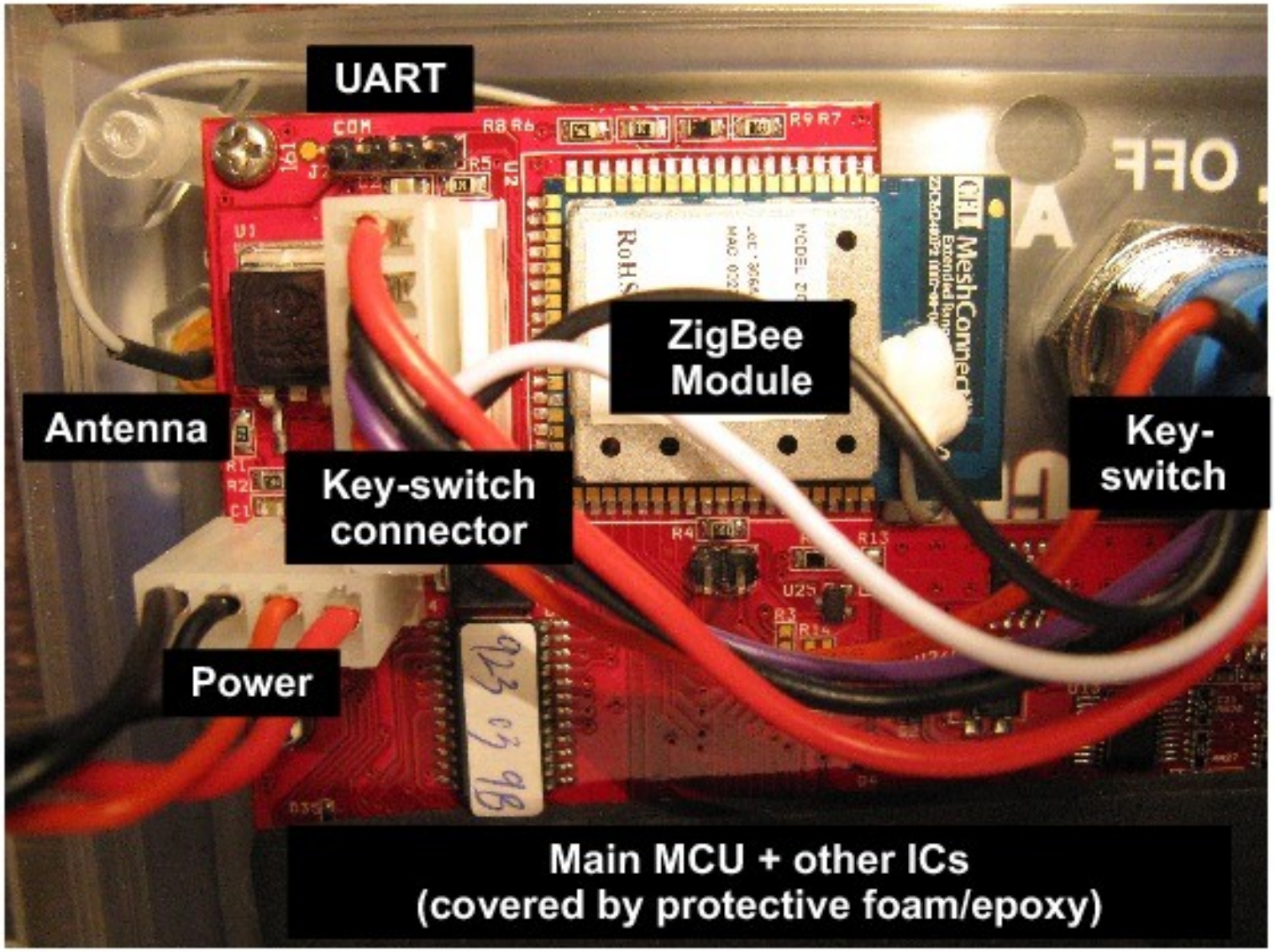
Develop tools and techniques to improve security evaluation at low cost

Making devices more robust
Hardware hardening

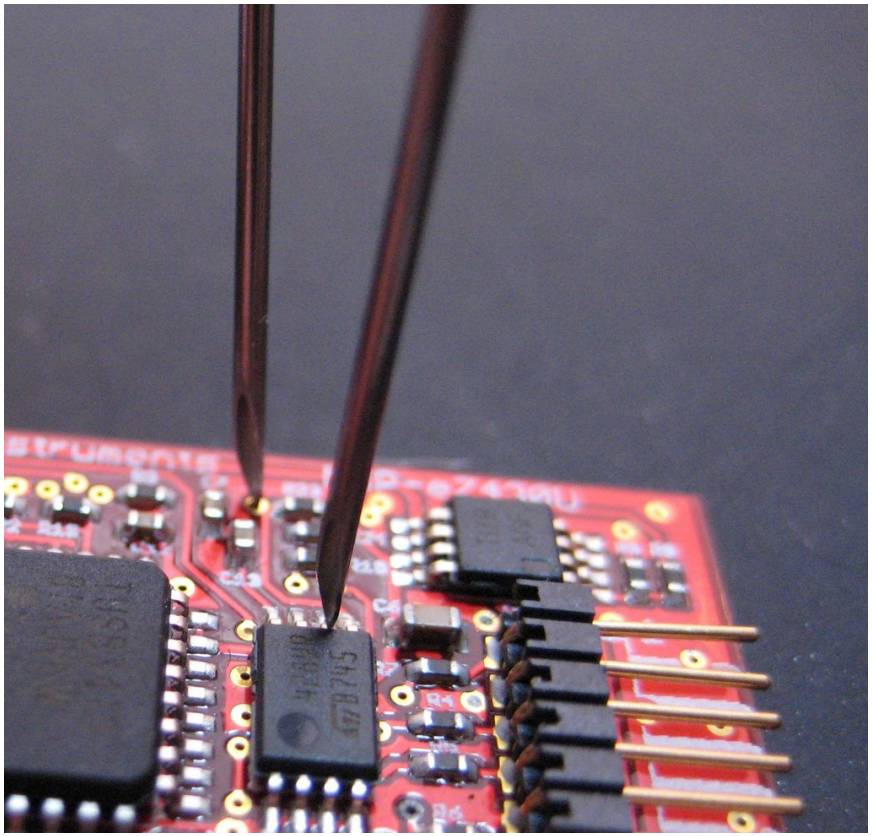
Different possible attacks

Simple hardware attacks are very simple!

- Identify the chips on the device
- Search for the datasheets of the chips
- Trace the PCB, connections between the devices
- Identify the purpose of each chip on the device
- Find jtag, serial, i2c …
- Dump memories, (serial flash !)
- Try to obtain the software, access internal secrets, execute code...


UART
ZigBee Module
Antenna
Key-switch connector
Key-switch
Power
Main MCU + other ICs
(covered by protective foam/epoxy)

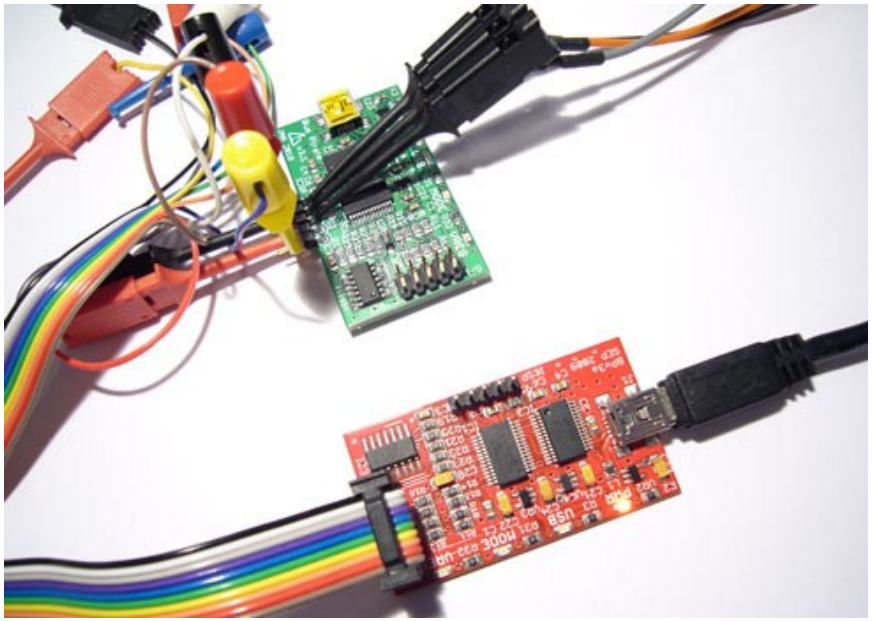
Probing bus, following vias



- Needles and a multimeter to follow the vias
- Alternative, desolder all components and take picture
- For more see Joe Grand woot'14 paper

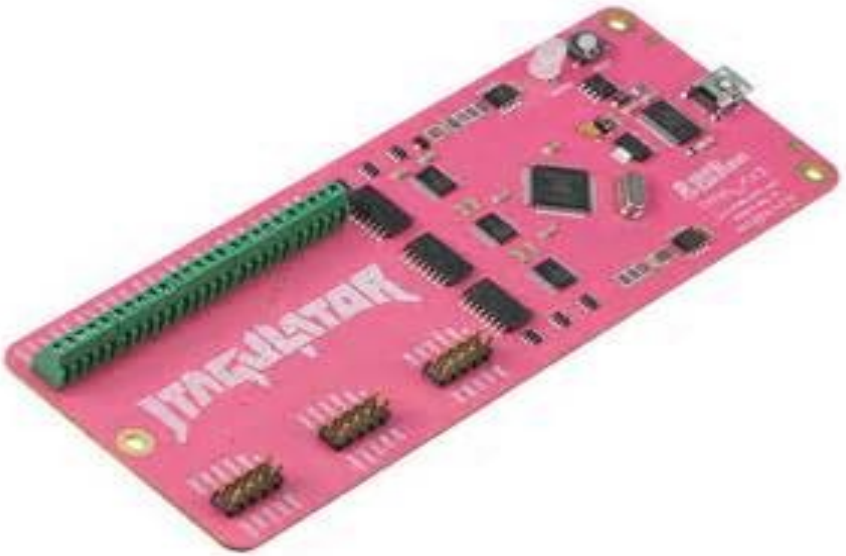
Picture T. Goodspeed

Bus pirate



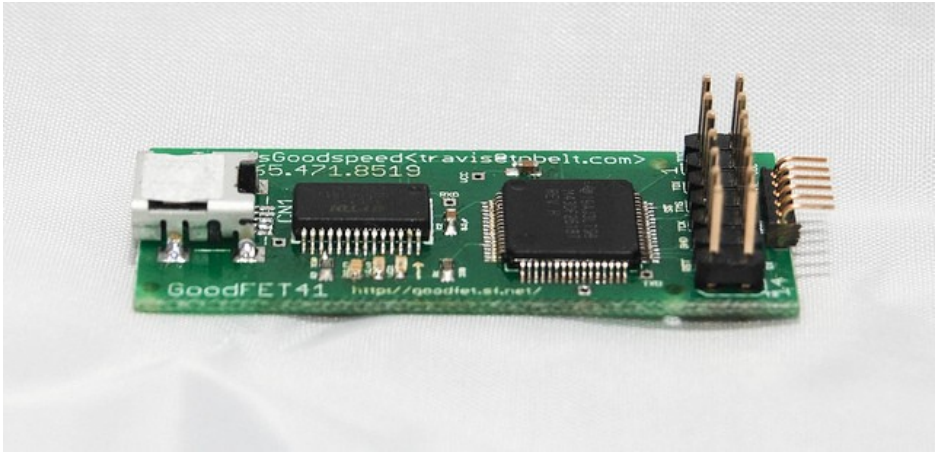
- Once a port is found we need to read it
- An universal device for talking many simple protocols

Jtagulator



- Automate detection of JTAG or serial ports

Goodfet

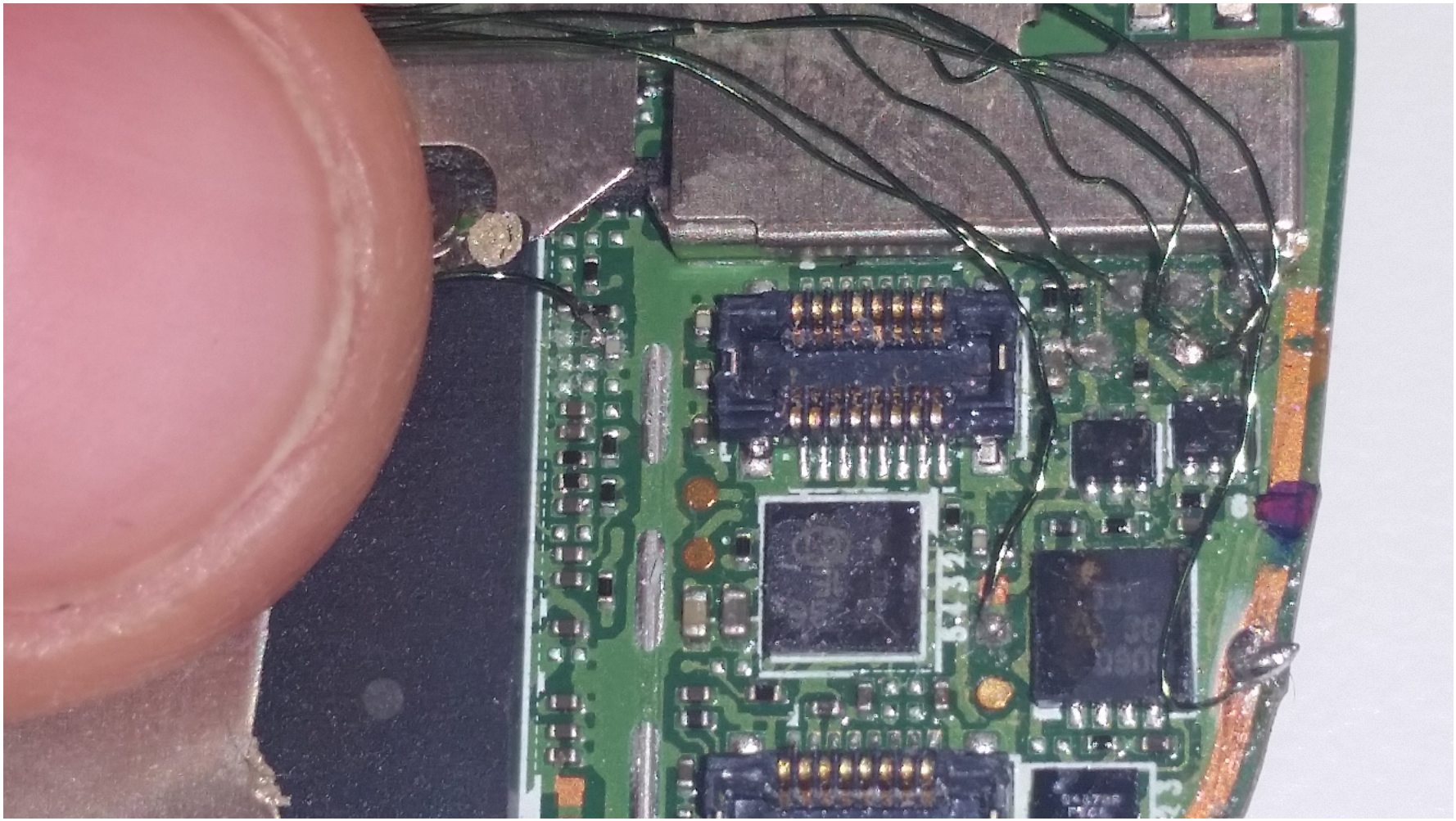


- Similar to buspirate
- Initially designed to be a cheap JTAG
- Supports many protocols, flash memories…
- Easily extensible firmware
- Cheap PCB available!
- Good to learn

Many other more advanced devices

- Good oscilloscope
- Logic analyzers
- ...

Soldering skills help



Simple countermeasures

Hardware countermeasures

- Obfuscate labels
- Bury important traces into deep layers
- Use BGA (harder to tap and desolder)
- Epoxy on top of chips
- Don't connect JTAG, serial or debug ports
- Use chips with internal memories
- Encrypt memory bus
- ...

Attacking the hardware

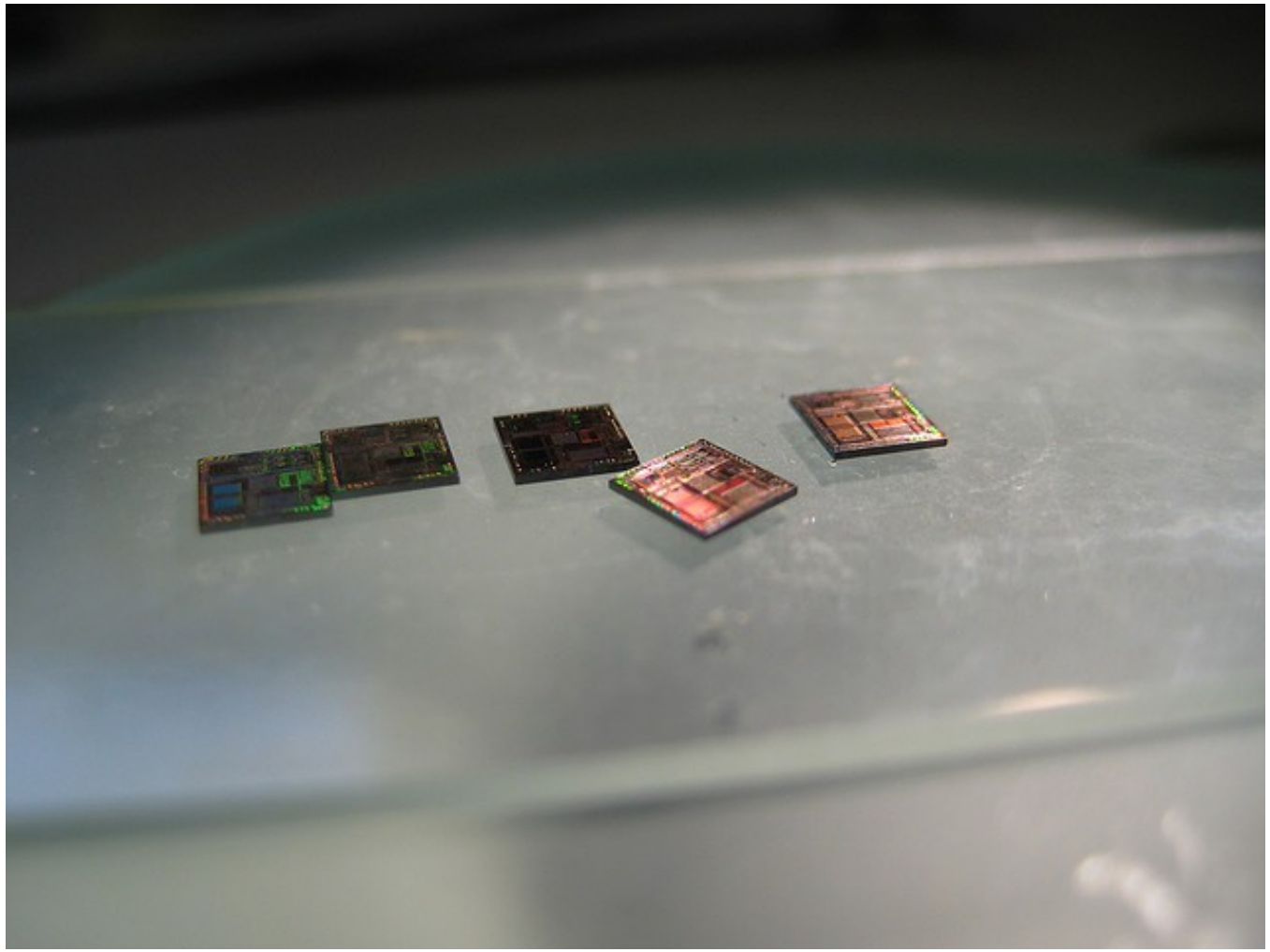
When an attacker can access the hardware a lot of things can be done

- Digging secrets in hardware
- Tampering internal signals and memories

Attacking the hardware directly

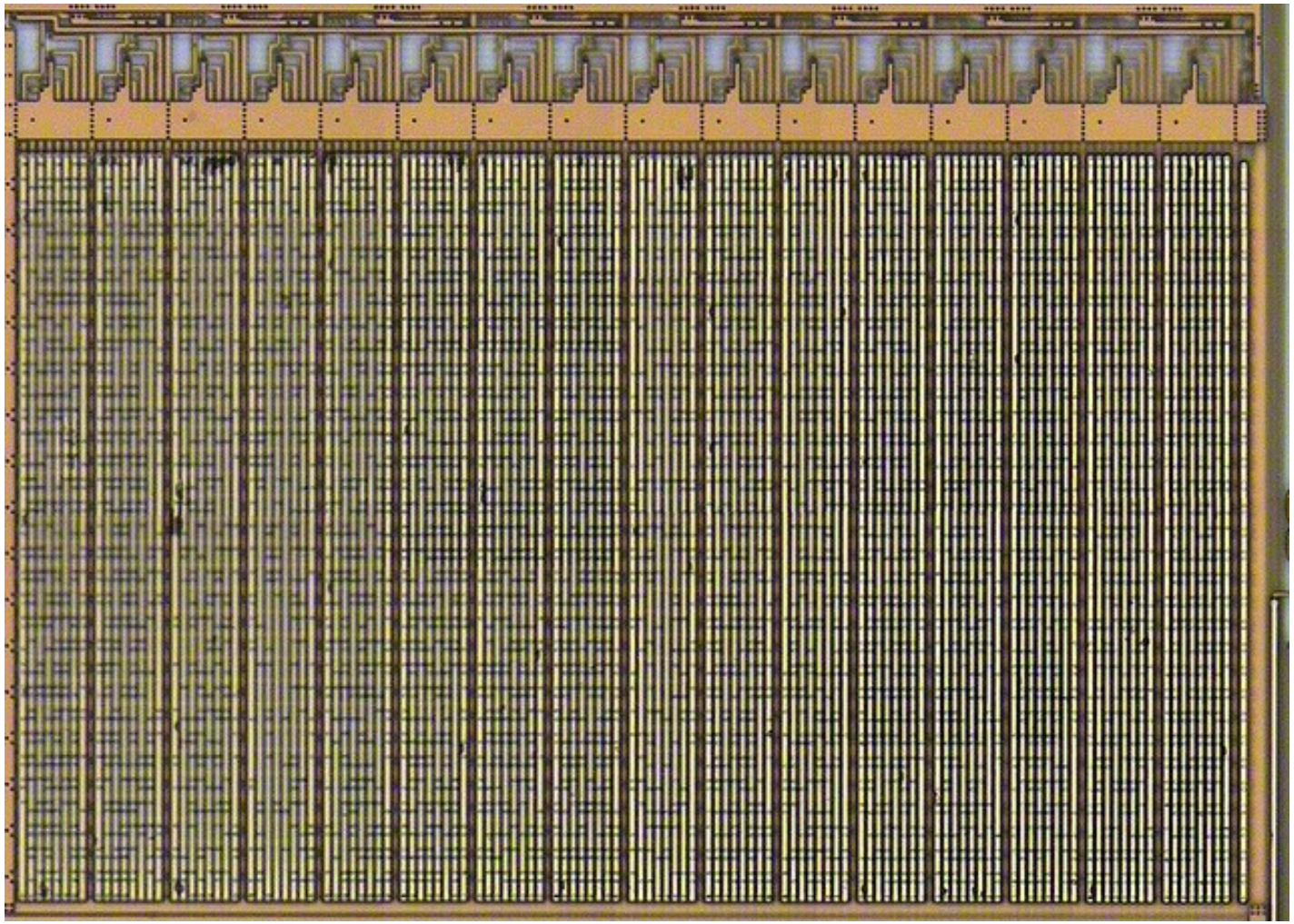
Picture T. Goodspeed

Decaping chips



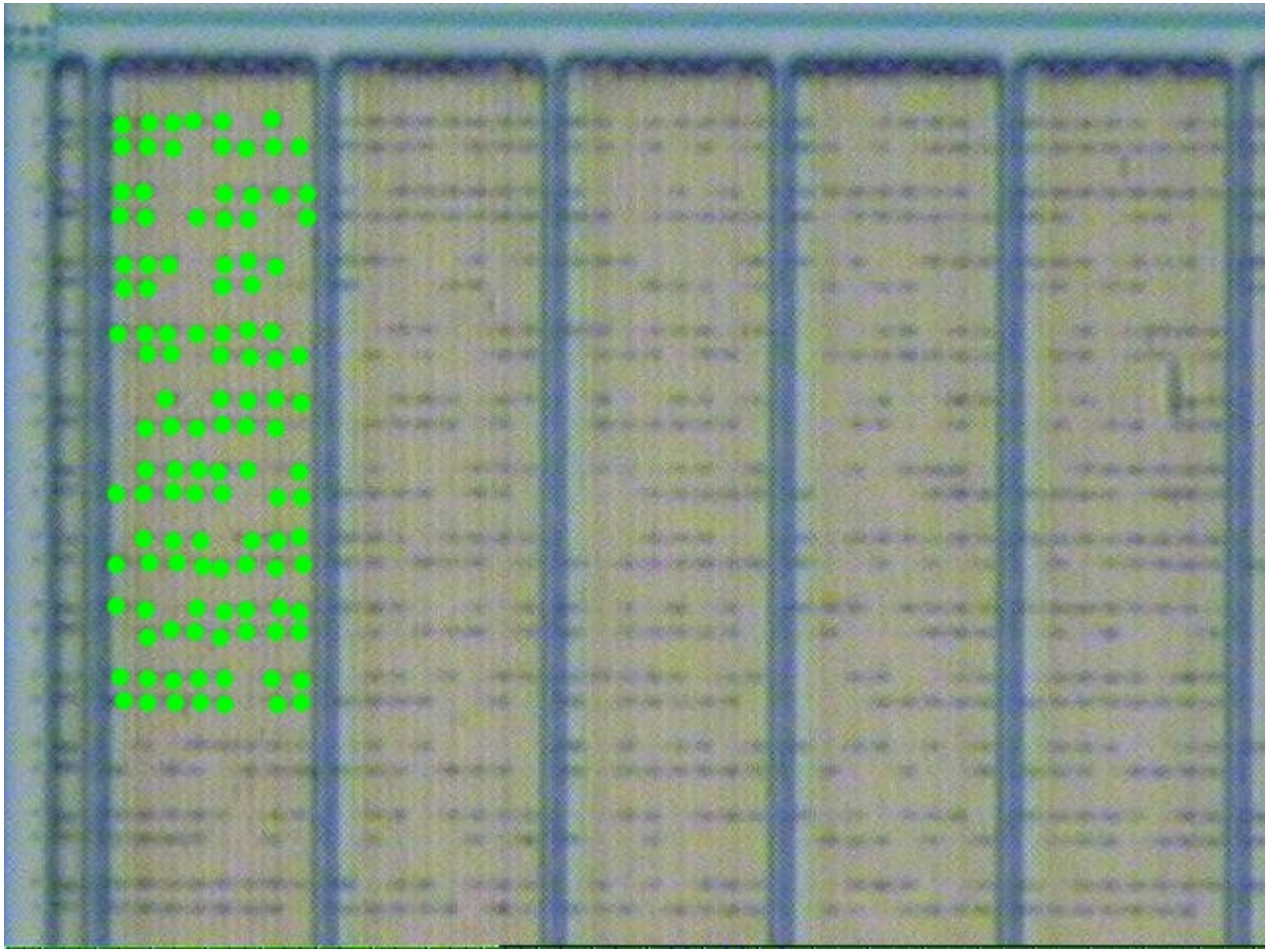
Picture T. Goodspeed

ROM memory form an MSP430



Picture T. Goodspeed

ROM memory form an MSP430



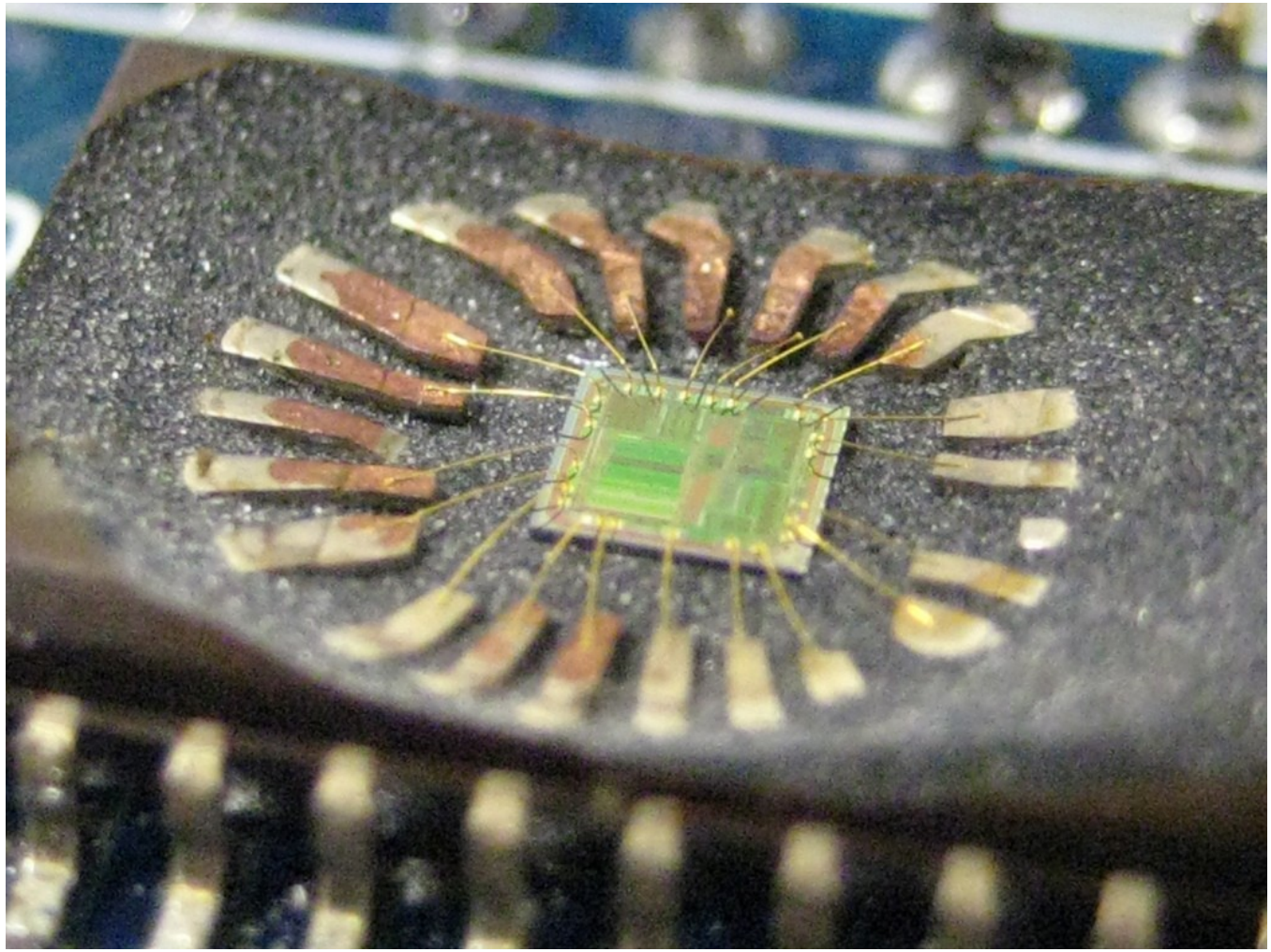
Picture T. Goodspeed

Other attacks

- Fault attacks
  - Make some logic fail, e.g., interpret incorrectly CPU instructions
  - Very efficient attacks

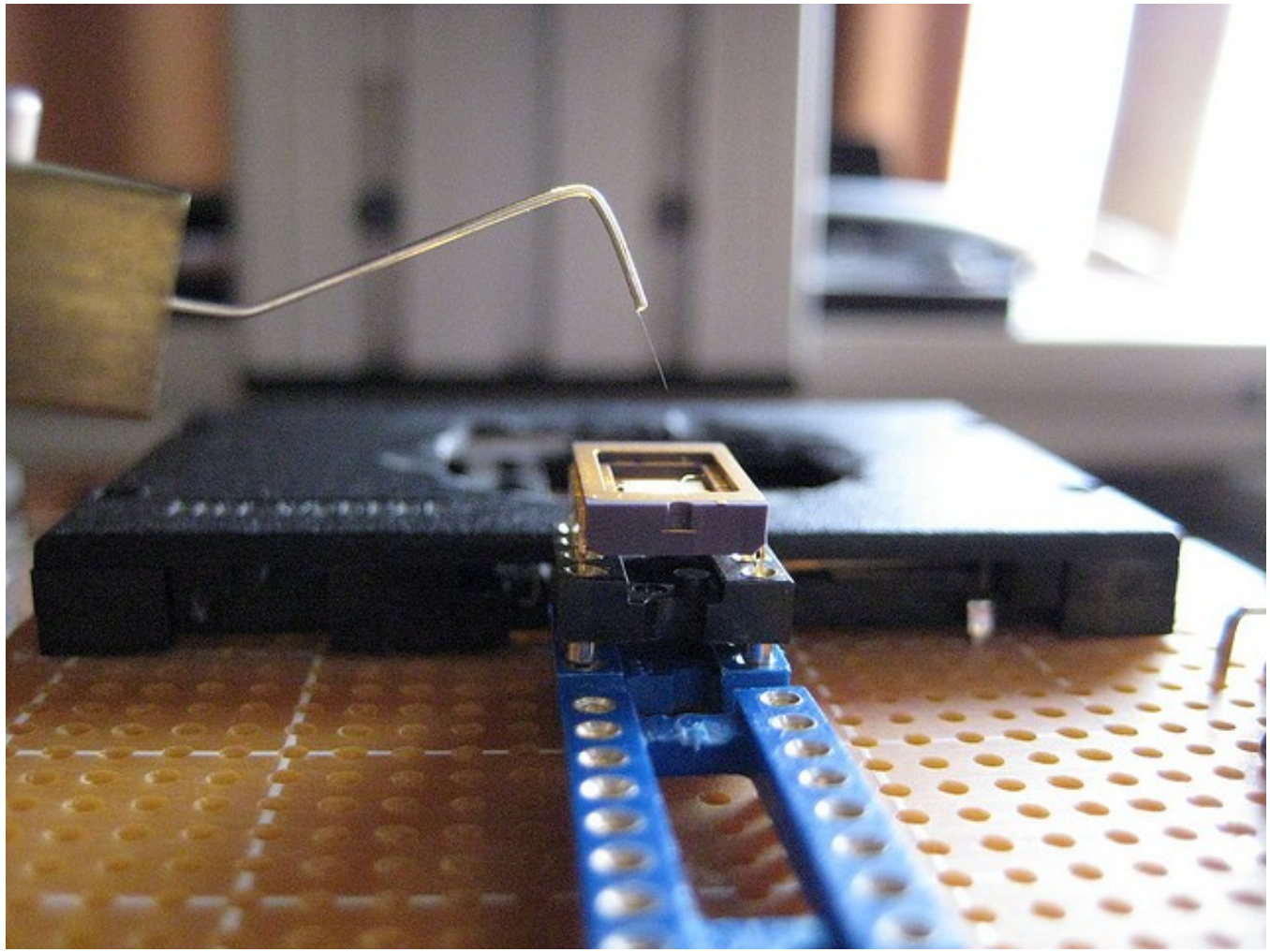
- Most common are
  - Clock, voltage Glitching
  - Laser fault

Functionnal chip



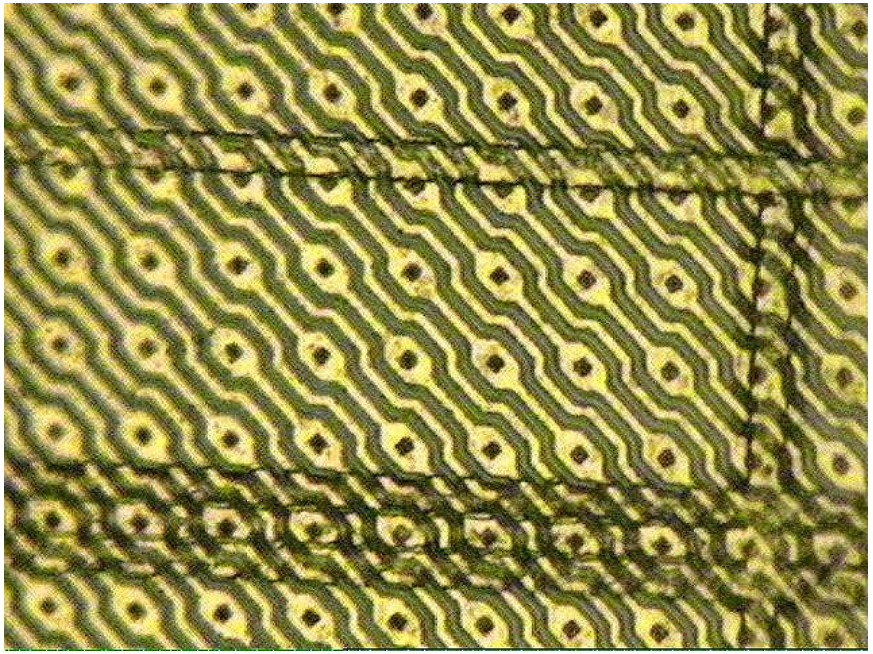
Picture T. Goodspeed

Probing



Picture T. Goodspeed

Hardware Countermeasures



- Mesh of wires above the logic
  - Tamper detection
- Light sensors
- Internal clock
- Glitch detection
- Etc…

Picture T. Goodspeed

Hardware attacks cont. (a.k.a. bazooka)

- A Focused Ion Beam can be used to modify a chip
- This is very powerful
- Quite expensive equipment
- What is the right level of security for the device ?

Hardware attacks countermeasures

- Additional metal layers
- Side channel/fault resistance

- This is smartcard level security
- Very few embedded devices have this level of protection
  - Has a cost

So what can we do about this ?

Good design
- Security Development Lifecycle (SDL)
- Hardening devices

Raise public awareness
- Evaluate the impact of compromises

Develop tools and techniques to improve security evaluation at low cost

Example of a software attack

Implementation and Implications of a Stealth Hard-Drive Backdoor

ACSAC 2013 (Best Student Paper Award)

EURECOM
Sophia Antipolis

Jonas Zaddach
Davide Balzarotti
Aurélien Francillon

Erik-Oliver Blass

Travis Goodspeed

IBM Research

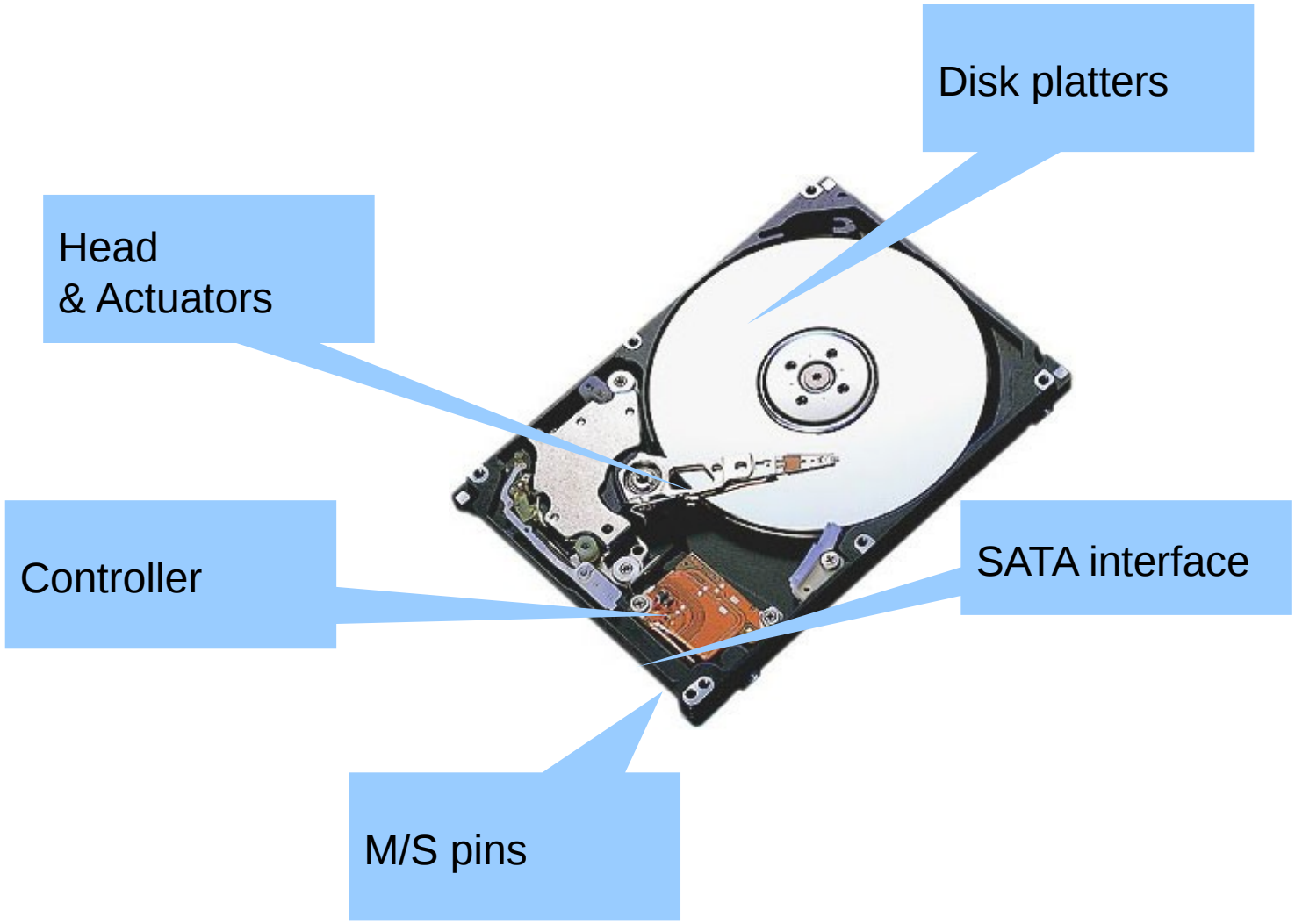
Anil Kurmus
Ioannis Koltsidas

Moitrayee Gupta

Goals

- **It's about threat models !**
  - Do we care about “hardware” compromises ?
  - Is it practical, feasible ?

- An example attack would be to
  - Understand, then backdoor the firmware
  - Malware compromises OS
  - Updates HDD with malicious firmware update
  - Disk is formatted, OS “re-installed”
  - But malicious HDD firmware remains!
  - OS compromised again from on the next boot


Disk platters
Head
& Actuators
Controller
SATA interface
M/S pins

Reverse engineering approach

Study PCB
- Chips : DRAM, Serial flash, JTAG (deactivated?), motor controller
- Google model ! Data recovery services.

Firmware updates available but format obscure

A serial diagnostic menu is available from M/S pins
- Gives PEEK/POKE primitives
- Allows to dump memory/load code

A serial Flash is on PCB
- Contains 2nd bootloader
- Could be desoldered/dumped/changed

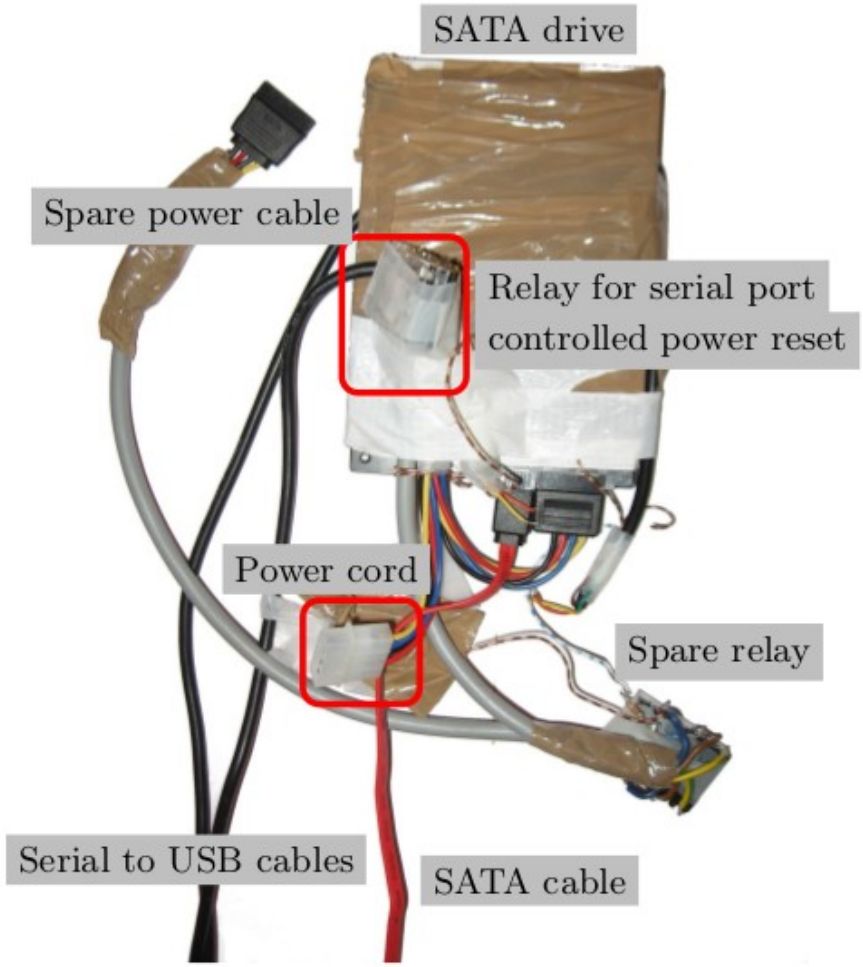
49

Reverse engineering approach tools

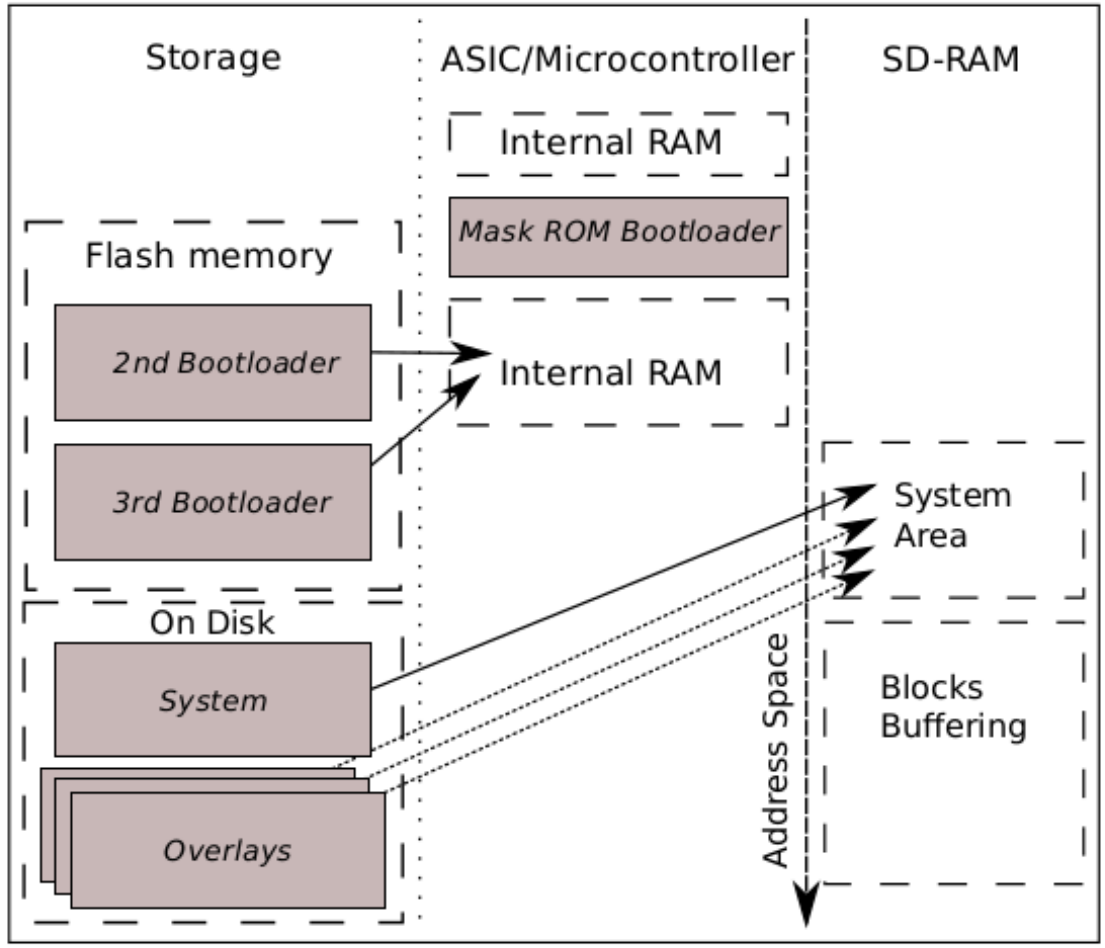
- Main task is to understand the firmware
  - But it's very large and obscure...
  - We need a way to debug the running firmware
  - To hook the backdoor in the original code

- Ida pro, a lot of patience,
  - Custom tools, dynamic analysis

Device instrumentation



Architecture



Backdoor Implementation

Many technical difficulties...

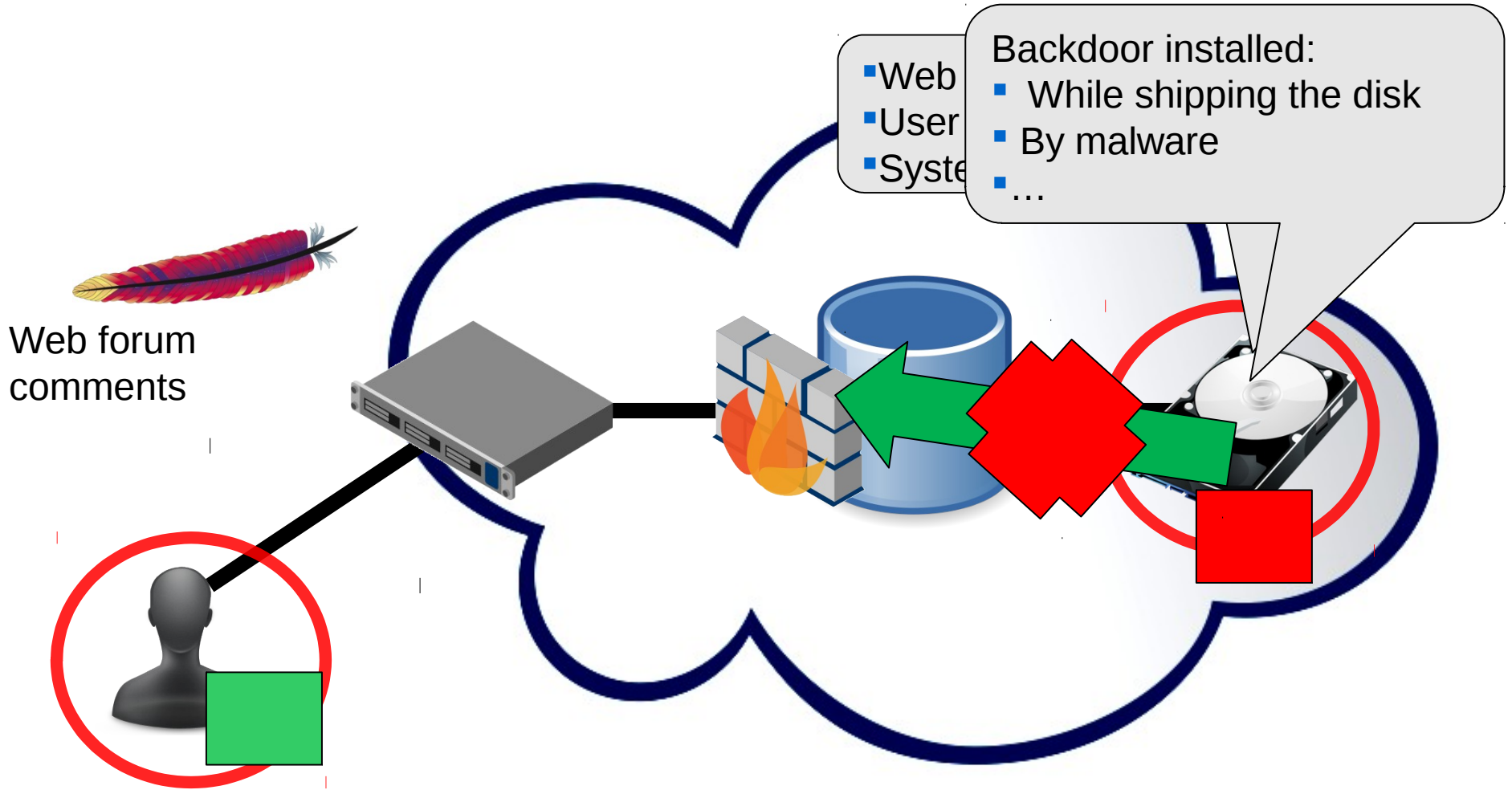
- Custom, event based OS
- Large statically linked code, no symbols

Results:

- Backdoor inserted in a firmware update
- Intercepts disk writes
- Can read blocks from disk (unstable*)
- No significant overhead (1%)

53

Exfiltration exemple: an online forum



54

Example: Exfiltrating a sensitive file

Use HDD as remote block device

- We can request any block
- So we can “mount” partitions

Exfiltrate /etc/shadow in nine “queries”:

- First retrieve partition table in MBR
- Then superblock of ext3 partition
- …

Total time: < 1 minute

55

In summary

We reverse engineered and backdoored a COTS drive

- 10 person-month effort
- Without any privileged information
- No significant performance overhead

Data-exfiltration backdoor

- No cooperation from host
- Stealthy

So is this a realistic threat model after all ?

- IRATEMONK

56

IRATEMONK (12/2013) ?

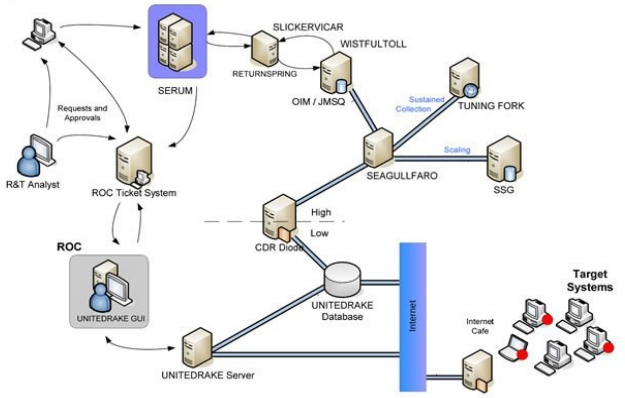
TOP SECRET//COMINT//REL TO USA, FVEY

IRATEMONK

ANT Product Data

(TS//SI//REL) IRATEMONK provides software application persistence on desktop and laptop computers by implanting the hard drive firmware to gain execution through Master Boot Record (MBR) substitution.

06/20/08



(TS//SI//REL) IRATEMONK Extended Concept of Operations

(TS//SI//REL) This technique supports systems without RAID hardware that boot from a variety of Western Digital, Seagate, Maxtor, and Samsung hard drives. The supported file systems are: FAT, NTFS, EXT3 and UFS.

(TS//SI//REL) Through remote access or interdiction, UNITEDRAKE, or STRAITBAZZARE are used in conjunction with SLICKERVICAR to upload the hard drive firmware onto the target machine to implant IRATEMONK and its payload (the implant installer). Once implanted, IRATEMONK's frequency of execution (dropping the payload) is configurable and will occur when the target machine powers on.

Status: Released / Deployed. Ready for Immediate Delivery

Unit Cost: $0

POC: [redacted], S32221, [redacted], [redacted]@nsa.ic.gov

Derived From: NSA/CSSM 1-52
Dated: 20070108
Declassify On: 20320108

TOP SECRET//COMINT//REL TO USA, FVEY

Internship? (1/2015)

(TS//SI//REL) Integrate SSD research into IRATEMONK products. This will involve 4 different parts:

- (TS//SI//REL) Leveraging research to create ARM-based SSD implant. This works involves reverse engineering SSD firmware and creating C and ARM assembly code to place inside of a firmware image to implement the IRATEMONK algorithm.
- (TS//SI//REL) Create version of the IMBIOS code that supports the SSD implant. This code runs on the x86 host and involves writing both C and

3 of 1

nternProjects - WikiInfo

x86 assembly. This work will involve interacting with the firmware implant as well as the code that IMBIOS bootstraps (SIERRAMIST).

- (TS//SI//REL) Add support for the SSD to WICKEDVICAR. WICKEDVICAR is the remote tool used to perform remote survey and installation. This code is C++ and will involve interacting with the firmware implant from a Windows OS.
- (TS//SI//REL) Add the SSD vendor support to the IRATEMONK firmware

First malware sample found by Kaspersky (2/2015)

10. What is the most sophisticated thing about the EQUATION group?

Although the implementation of their malware systems is incredibly complex, surpassing even Regin in sophistication, there is one aspect of the EQUATION group's attack technologies that exceeds anything we have ever seen before. **This is the ability to infect the hard drive firmware.**

We were able to recover two HDD firmware reprogramming modules from the EQUATIONDRUG and GRAYFISH platforms. The EQUATIONDRUG HDD firmware reprogramming module has version 3.0.1 while the GRAYFISH reprogramming module has version 4.2.0. These were compiled in 2010 and 2013, respectively, if we are to trust the PE timestamps.

NSA approach to backdooring Disks

From public documents we know that:

- Compromise a computer
- A DLL loads a modified firmware image
- The firmware replaces the MBR by a modified MBR
- Most likely then use this to infect the boot chain
  - Not very stealth !
  - Could be detected by a TPM ?

Snowden documents on “interdiction”

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A “load station” implants a beacon

(TS//SI//NF) In one recent case, after several months a beacon implanted through supply-chain interdiction called back to the NSA covert infrastructure. This call back provided us access to further exploit the device and survey the network.

What can we do about this ?

So what can we do about this ?

Good design
- Security Development Lifecycle (SDL)
- Hardening devices

Raise public awareness
- Evaluate the impact of compromises

Develop tools and techniques to improve security evaluation at low cost

Performing analysis of embedded systems

2 main ways to analyze embedded systems:

- Static analysis
- Dynamic Analysis

We consider only black (or grey) box analysis

- Also realistic for manufacturers, audits,...

AVATAR: A Framework for Dynamic Security Analysis of Embedded Systems' Firmwares

Presented at NDSS '14

Jonas Zaddach,

Luca Bruno, Aurélien Francillon, Davide Balzarotti

EURECOM
Sophia Antipolis

Dynamic analysis

Problem:

- Unknown peripherals
- Emulating CPU only not sufficient
- Limited visibility with execution on hardware
  - With a gdb stub, JTAG...

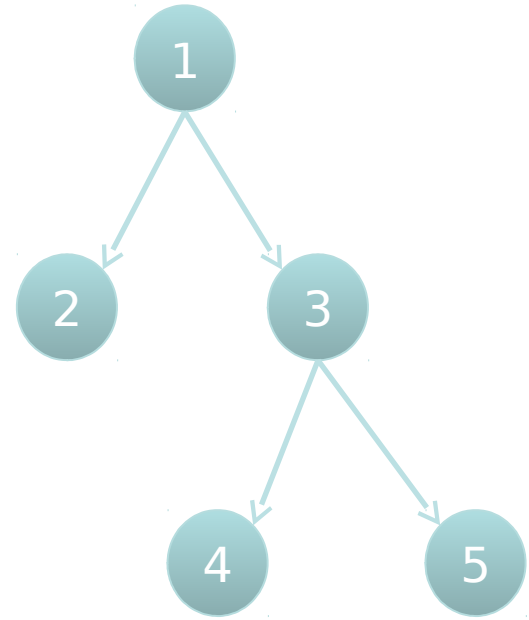
Advanced analysis impossible

- Tracing, Tainting, Symbolic execution

Tools for security evaluation

Techniques that are typically used on a PC

- Advanced debugging techniques
  - Tracing
  - Fuzzing
  - Tainting
  - Symbolic Execution



Tools for security evaluation

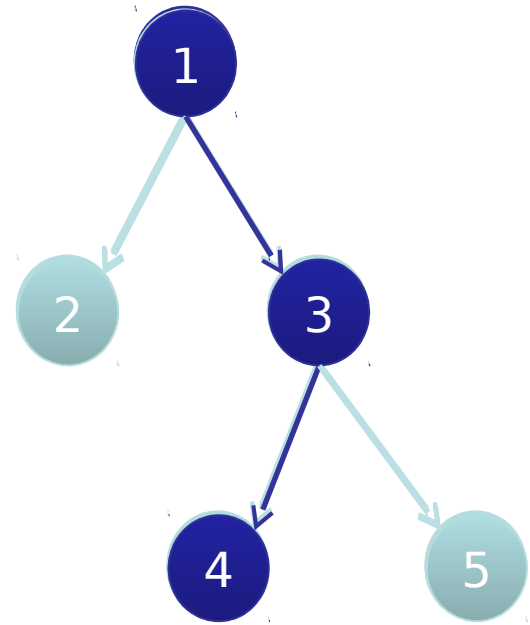
Techniques that are typically used on a PC

- Advanced debugging techniques
  - Tracing
  - Fuzzing
  - Tainting
  - Symbolic Execution

Collecting an execution trace

Trace

1: x = 4
3: x = 4
4: x = 6



Tools for security evaluation

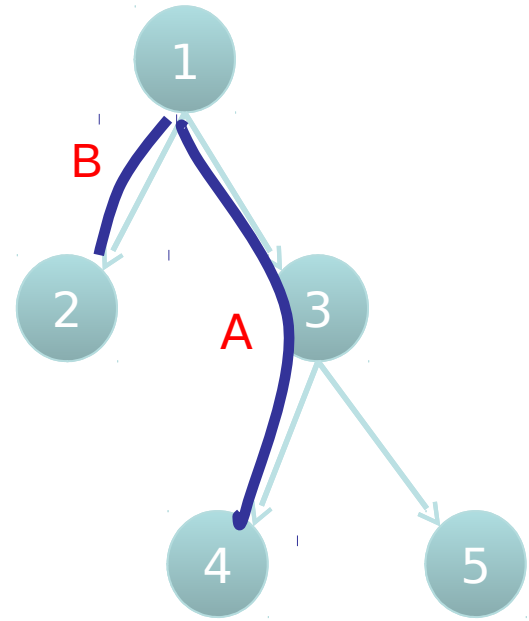
Techniques that are typically used on a PC

- Advanced debugging techniques
  - Tracing
  - Fuzzing
  - Tainting
  - Symbolic Execution

Testing with random input

Testcase A
x = 4

Testcase B
x = 2

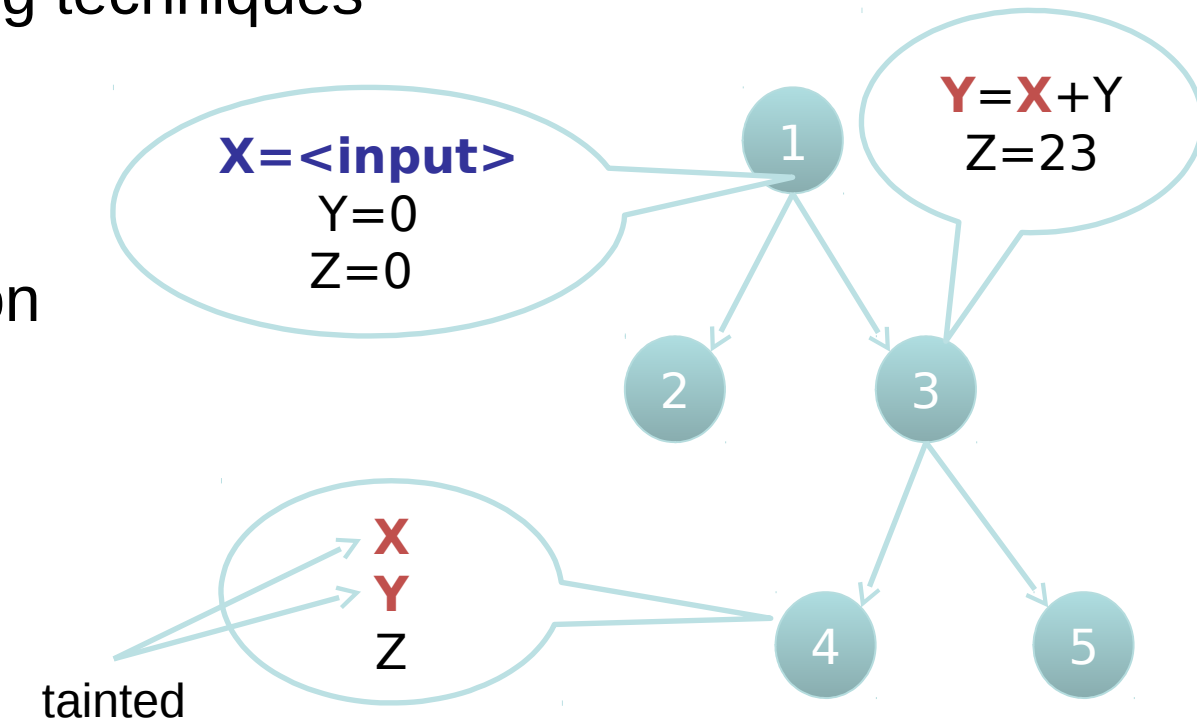


Tools for security evaluation

Techniques that are typically used on a PC

- Advanced debugging techniques
  - Tracing
  - Fuzzing
  - Tainting
  - Symbolic Execution

Data flow tracking

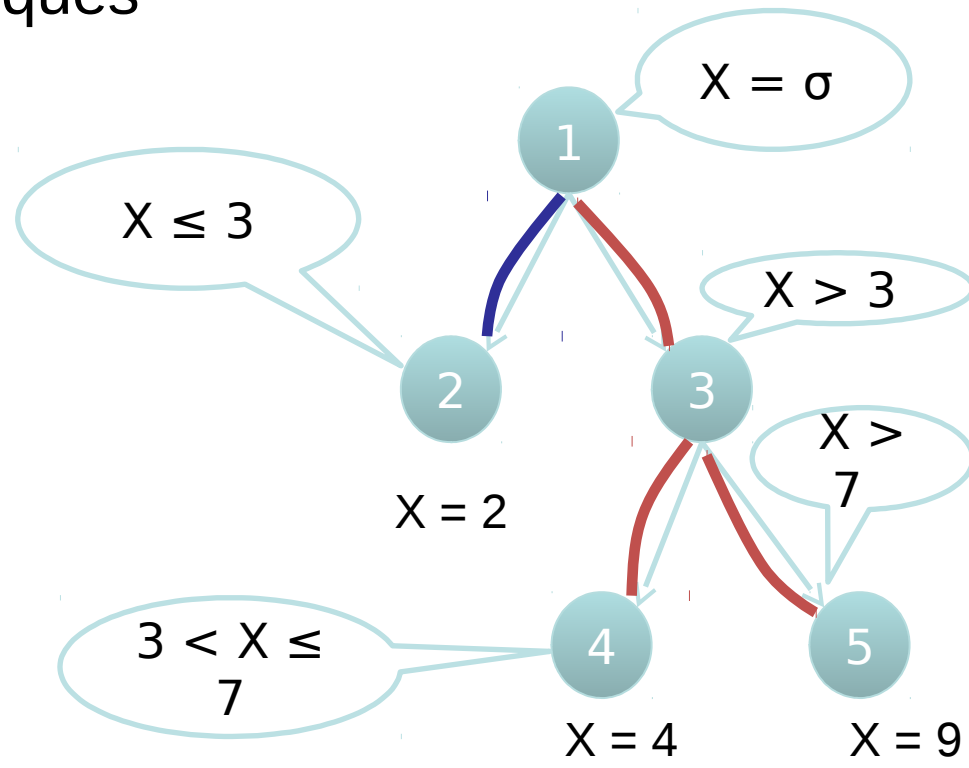


Tools for security evaluation

Techniques that are typically used on a PC

- Advanced debugging techniques
  - Tracing
  - Fuzzing
  - Tainting
  - Symbolic Execution

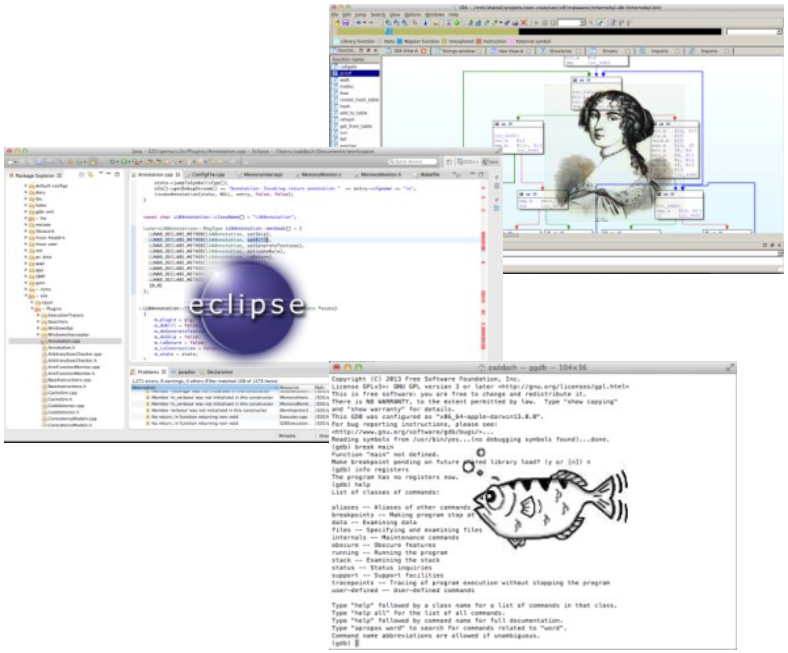
Multipath exploration



Tools for security evaluation

Techniques that are typically used on a PC

- Advanced debugging techniques
  - Tracing
  - Fuzzing
  - Tainting
  - Symbolic Execution
- Integrated tools
  - IDA Pro
  - GDB
  - Eclipse



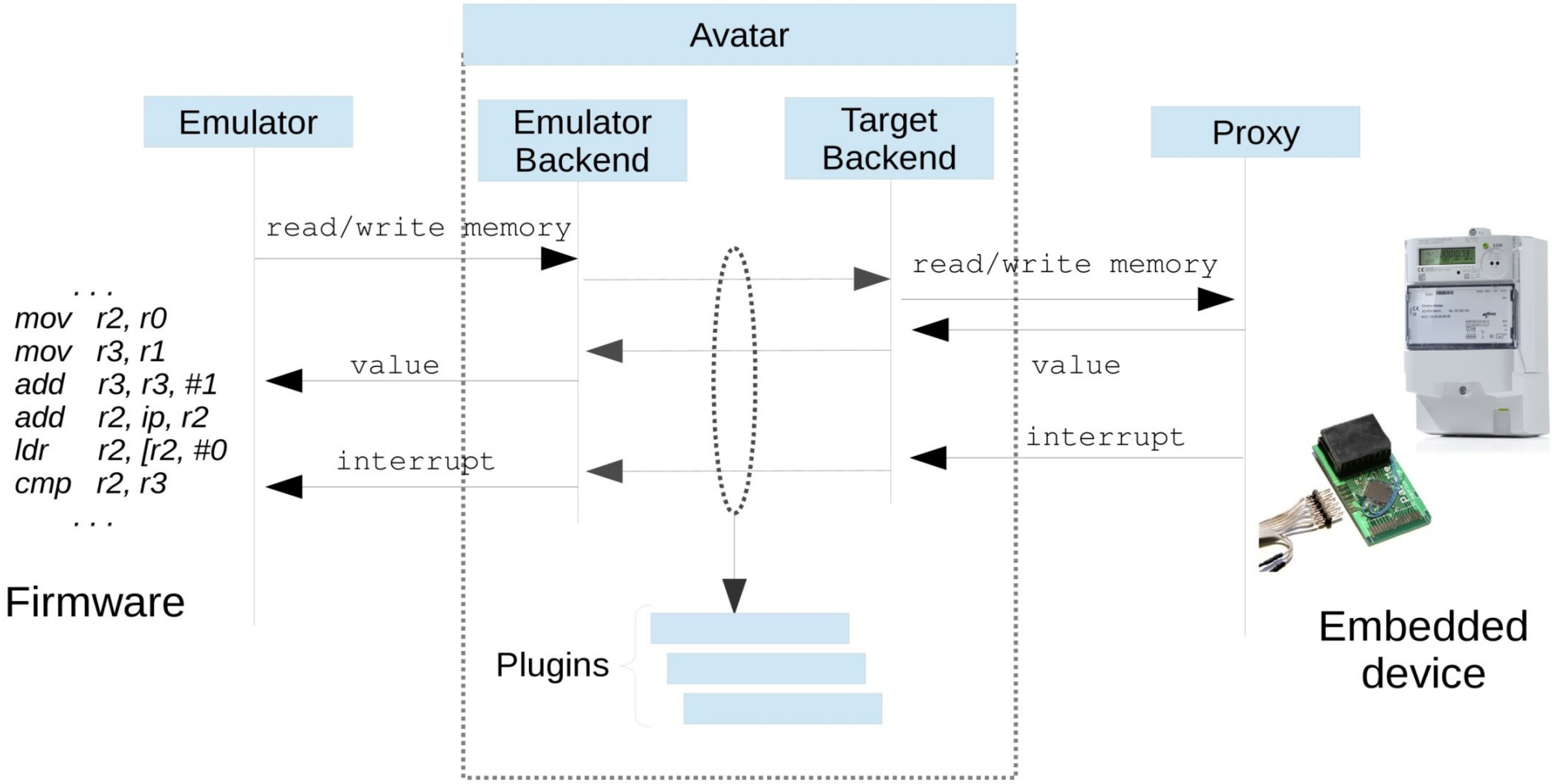
Avatar

Avatar idea :

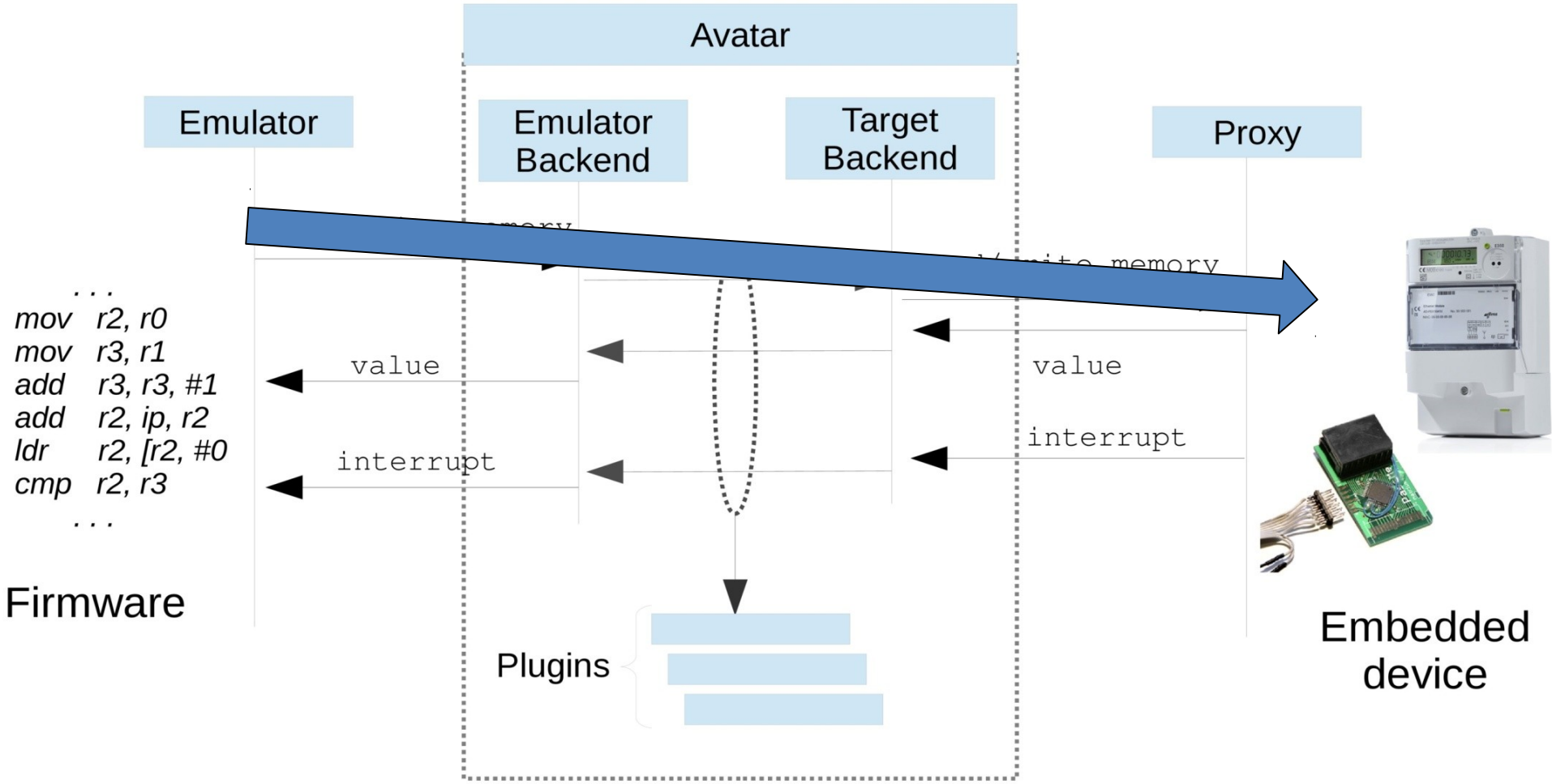
- Arbitration framework
- Emulate the firmware on the emulator
  - Forward IO to deivce
- Python scripting of all tools (qemu, jtag, symbolic execution…)
- A process, helps for reverse engineering at the same time as automating many tasks and testing

- Open source, with examples: http://s3.eurecom.fr/tools/avatar/

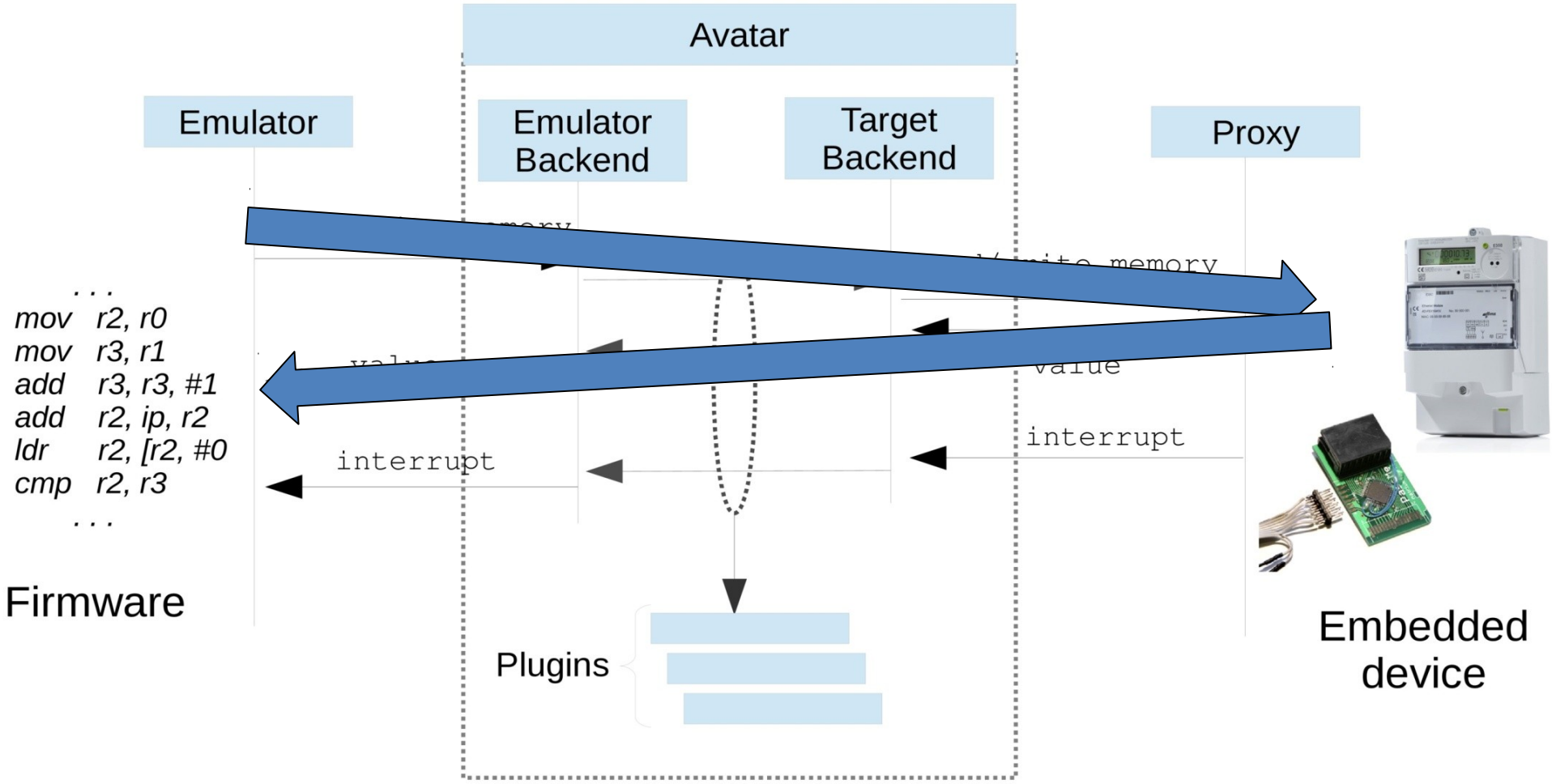
Avatar simplified principle



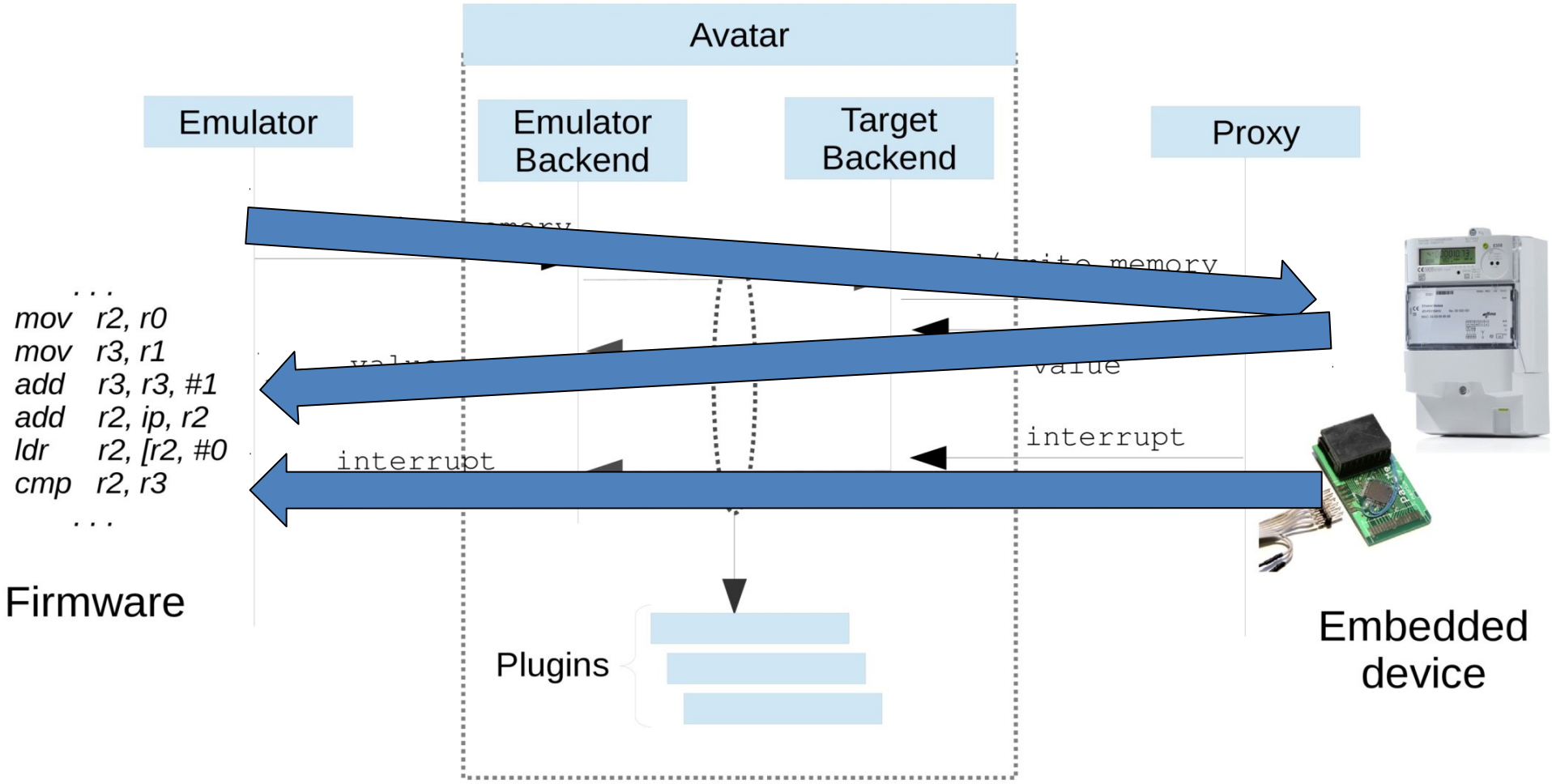
Avatar simplified principle



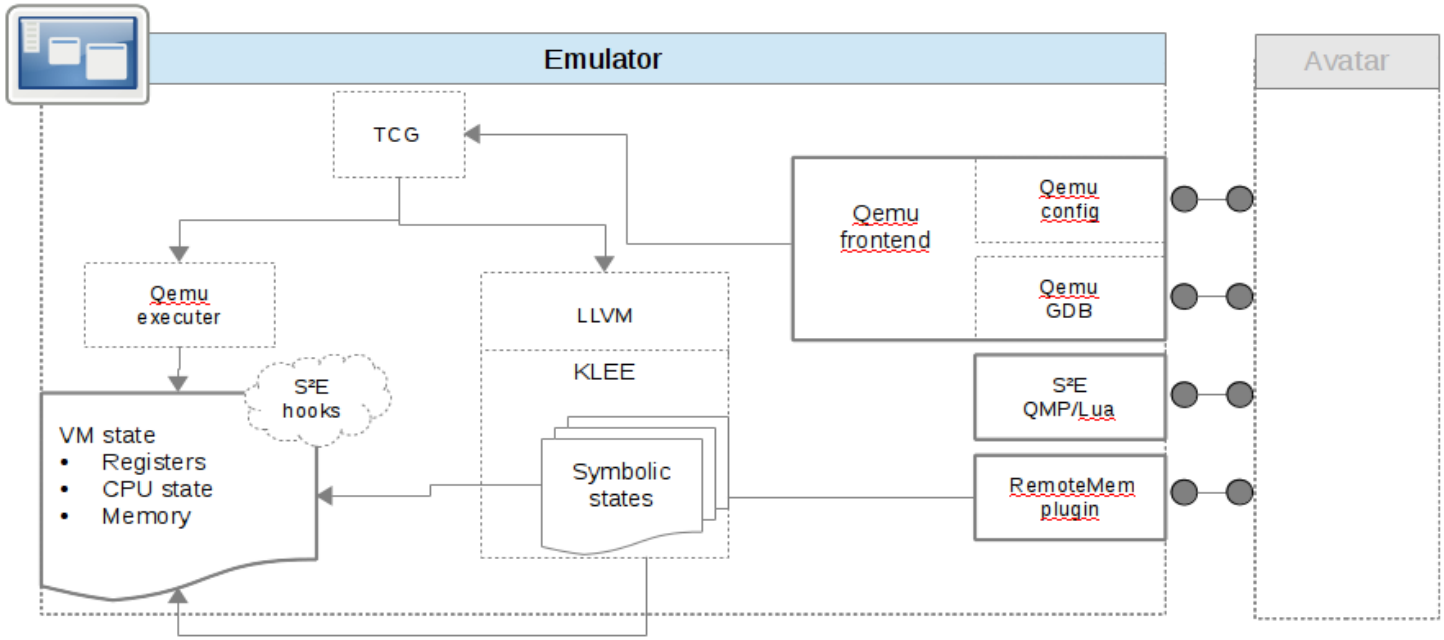
Avatar simplified principle



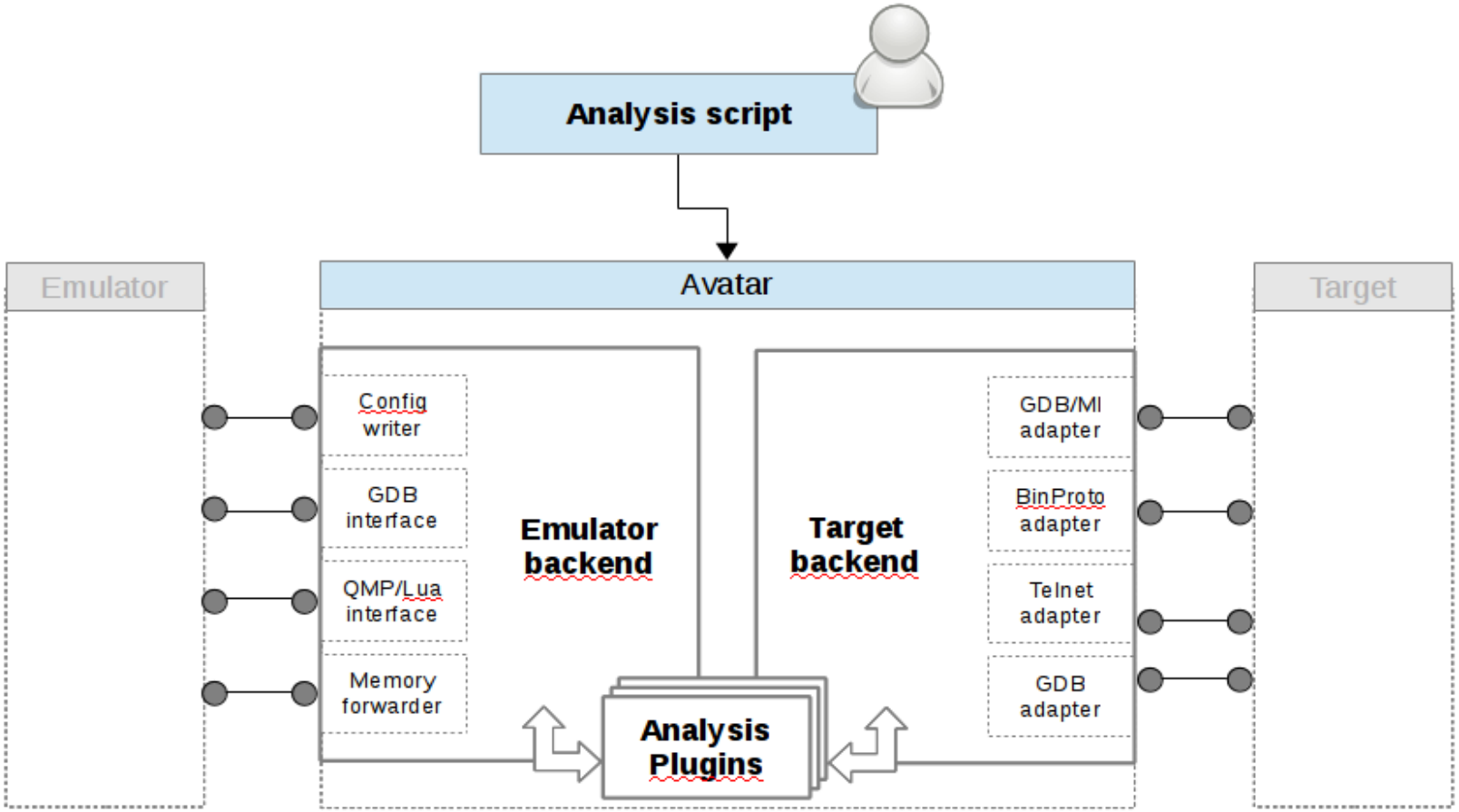
Avatar simplified principle



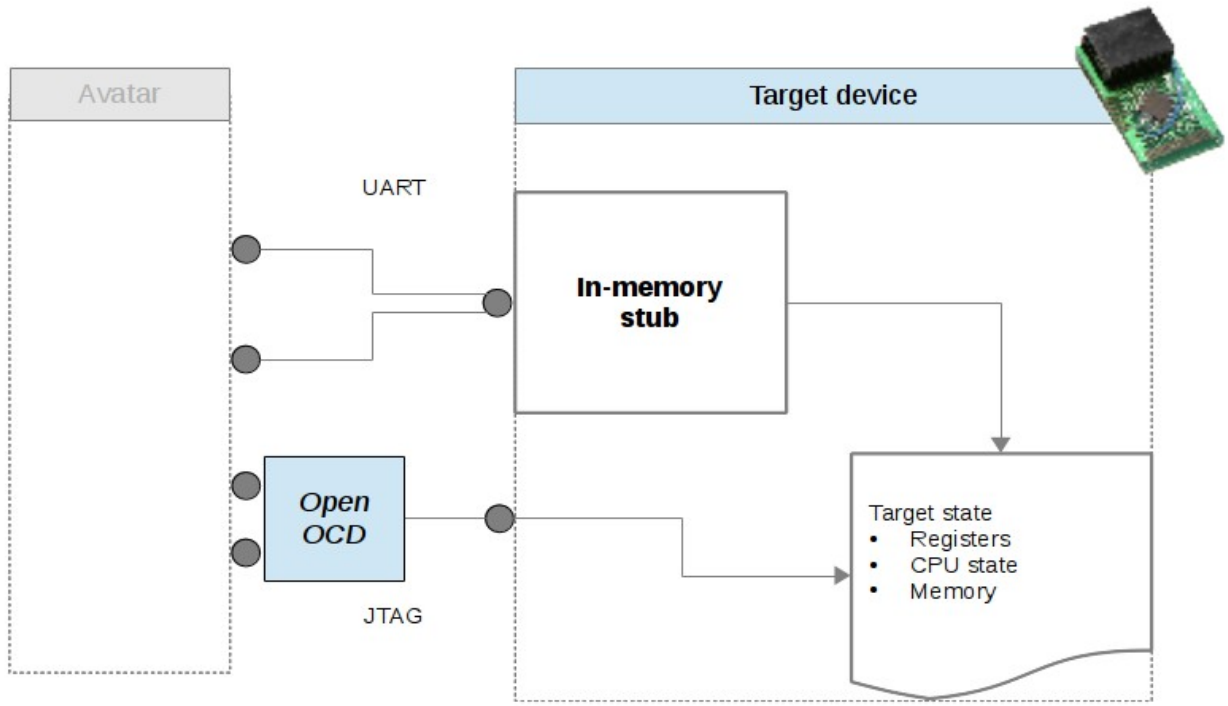
Avatar simplified principle



Avatar simplified principle



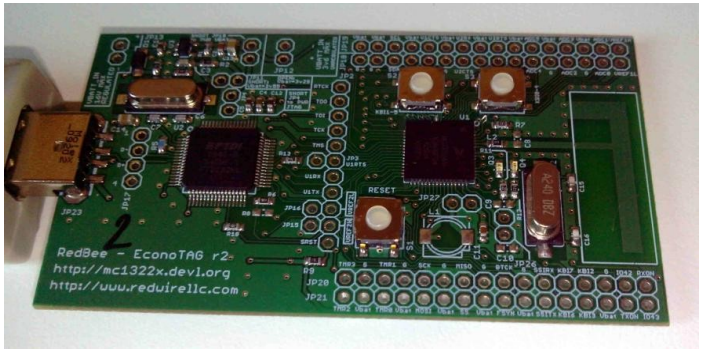
Avatar simplified principle



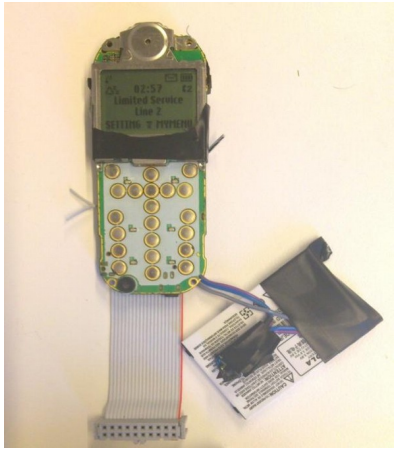
Use cases

- **Analyzing the ROM bootloader of an HDD**

- **Finding bugs in a Zigbee wireless sensor device**



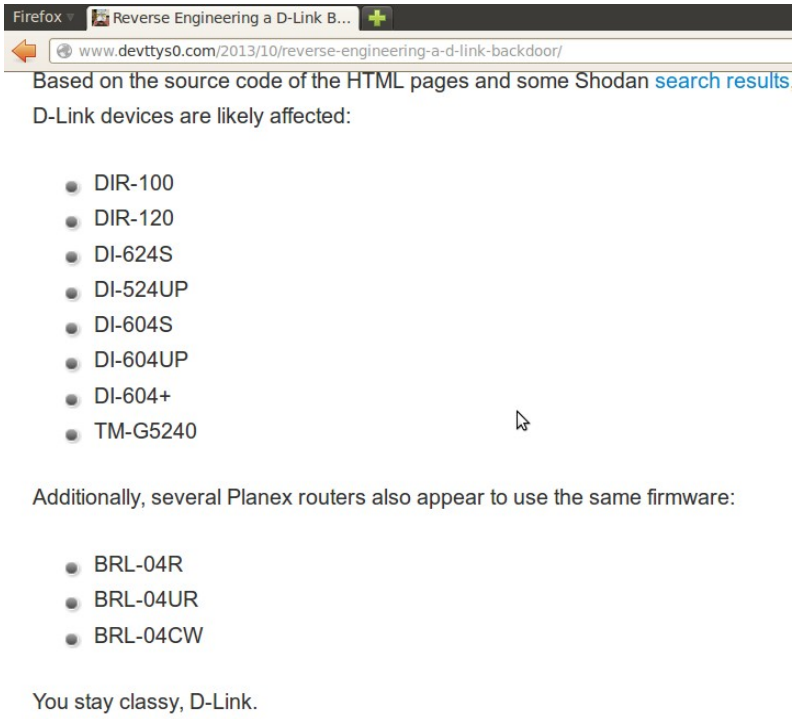
- **Analyzing the baseband code of a GSM feature phone**



Dynamic analysis is great, but does not scale.
How can we analyze such systems at a large scale ?

Many Examples of Insecure Embedded Systems

- Routers
- Printers
- VoIP
- Cars
- Drones



Many Examples of Insecure Embedded Systems

- Routers
- Printers
- VoIP
- Cars
- Drones

Firefox ▾ Reverse Engineering a D-Link B...

www.devtt

Based on the

D-Link devic

- DIR-1(
- DIR-1
- DI-624
- DI-524
- DI-604
- DI-604
- DI-604
- TM-G

Additionally,

- BRL-0
- BRL-0
- BRL-0

You stay classy, D-Link.

Networked Printers at Risk

By Jimmy Shah on Dec 30, 2011

Like 0 Share 0 +1 0 Tweet 0

Multifunction printers (MFPs) have been common in offices for years. They let employees print, scan, and copy documents. Two separate talks at the 28th Chaos Communications Congress (28c3) show how attackers can infect these trusted office devices.

Hacking MFPs

In Andrei Costin's presentation "Hacking MFPs," he covered the history of printer and copier hacks from the 1960s to today. The meat of the talk concerned executing remote code on an MFP using crafted PostScript. Just printing a particular document can get code to run on the machine. Previous research proof of concepts have done exactly that, once with a specially designed Word document and once with a Java applet.

Many Examples of Insecure Embedded Systems

- Routers
- Printers
- VoIP
- Cars
- Drones

Firefox ▾ Reverse Engineering a D-Link R…

www.devtt

Based on the

By Jim

D-Link devic

- DIR-1(
- DIR-1
- DI-624
- DI-524
- DI-604
- DI-604
- DI-604
- TM-G

Additionally,

- BRL-0
- BRL-0
- BRL-0

You stay classy, D

Multi print, Cong

Hack In Ar hack

MFP mach desig

FBI: Criminals Auto-dialing With Hacked VoIP Systems

By Robert McMillan, IDG News Service — Dec 5, 2008 5:50 PM

Criminals are taking advantage of a bug in the Asterisk Internet telephony system that lets them pump out thousands of scam phone calls in an hour, the U.S. Federal Bureau of Investigation warned Friday.

The FBI didn't say which versions of Asterisk were vulnerable to the bug, but it advised users to upgrade to the latest version of the software. Asterisk is an open-source product that lets users turn a Linux computer into a VoIP (Voice over Internet Protocol) telephone exchange.

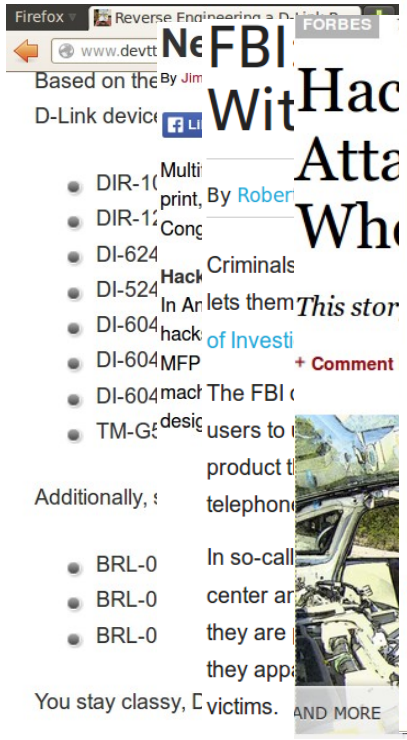
In so-called vishing attacks, scammers usually use a VoIP system to set up a phony call center and then use phishing e-mails to trick victims into calling the center. Once there, they are prompted to give private information. But in the scam described by the FBI, they apparently are taking over legitimate Asterisk systems in order to directly dial victims.

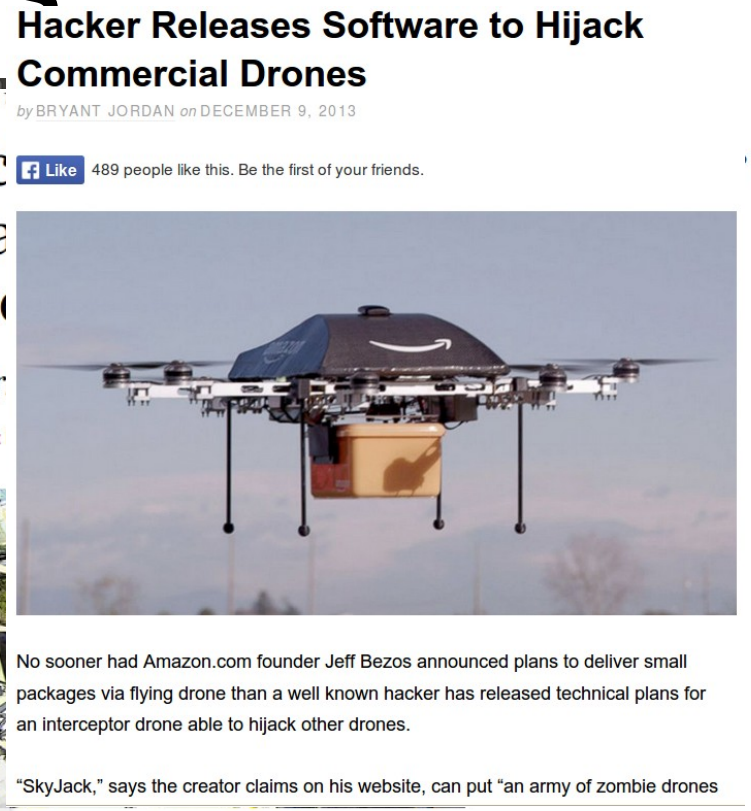
Many Examples of Insecure Embedded Systems

- Routers
- Printers
- VoIP
- Cars
- Drones

Many Examples of Insecure Embedded Systems

- Routers
- Printers
- VoIP
- Cars
- Drones





Many Examples of Insecure Embedded Systems

- Routers
- Printers
- VoIP
- Cars
- Drones

Problem:

- Those are individual, manual, tedious efforts
- How to do this at large scale?

The problem with large scale analysis

- Heterogeneity of
  - Hardware, architectures, OSes, users, requirements, security goals
- Manual analysis does not scale, it requires
  - Finding and downloading the firmwares
  - Unpacking and performing initial analysis
  - (Re-)discovering the same or similar bugs in other firmwares

Our approach

1. Collect a large number of firmware images
2. Perform broad but simple static analysis
3. Correlate across firmwares

Many advantages:
- No intrusive online testing, no devices involved
- Scalable

But also many challenges

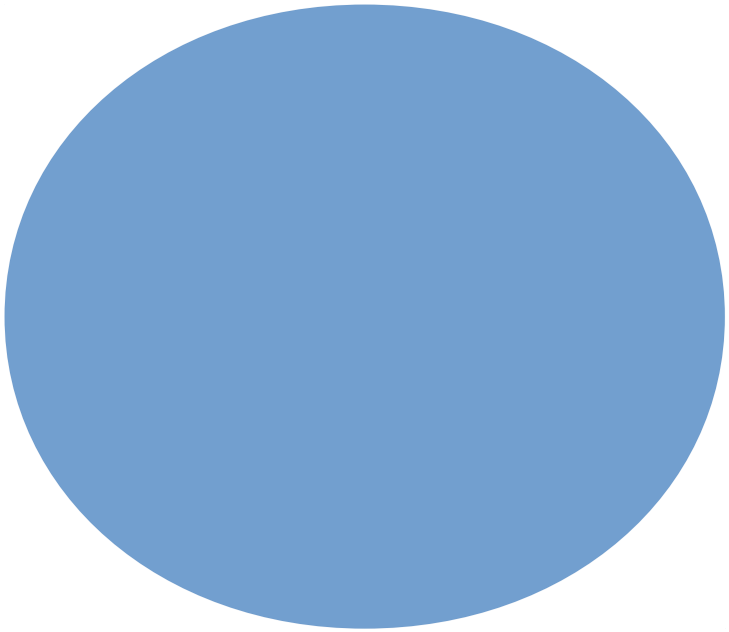
« A Large Scale Analysis of the Security of Embedded Firmwares »
Andrei Costin, Jonas Zaddach, Aurélien Francillon, Davide Balzarotti
USENIX Security 2015

Challenges

- Firmware identification (.exe/.ps/...)
- Firmware Unpacking
- Representative dataset
- Scalability, computational limits
- Results confirmation

Challenge: Firmware Identification

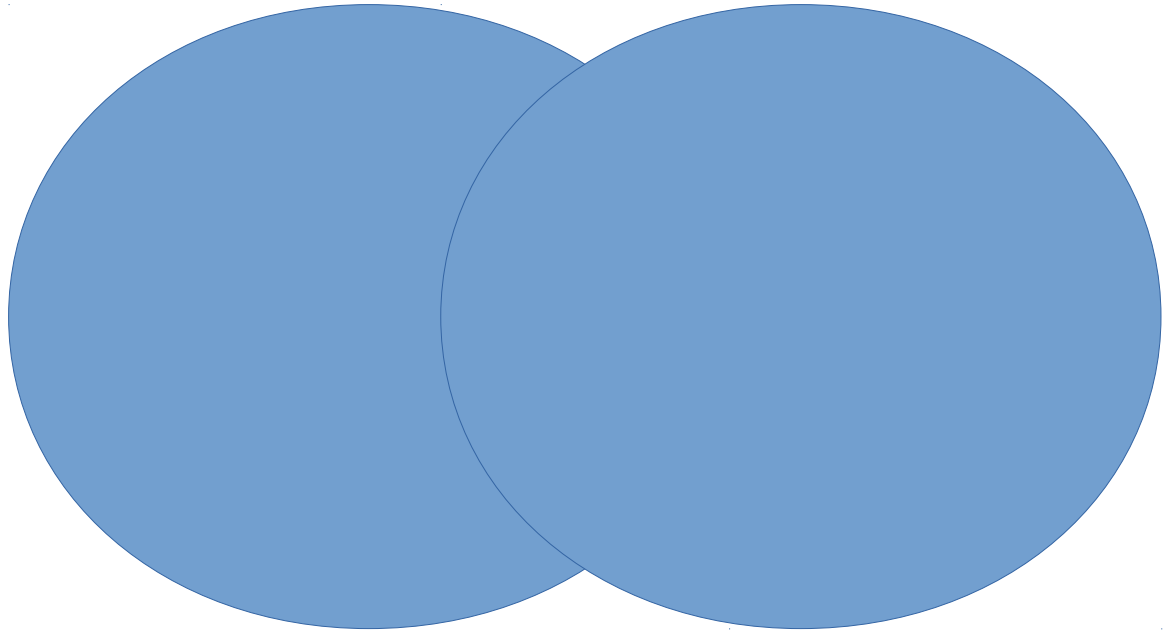
Clearly a Firmware



Challenge:
Firmware Identification

Clearly a Firmware

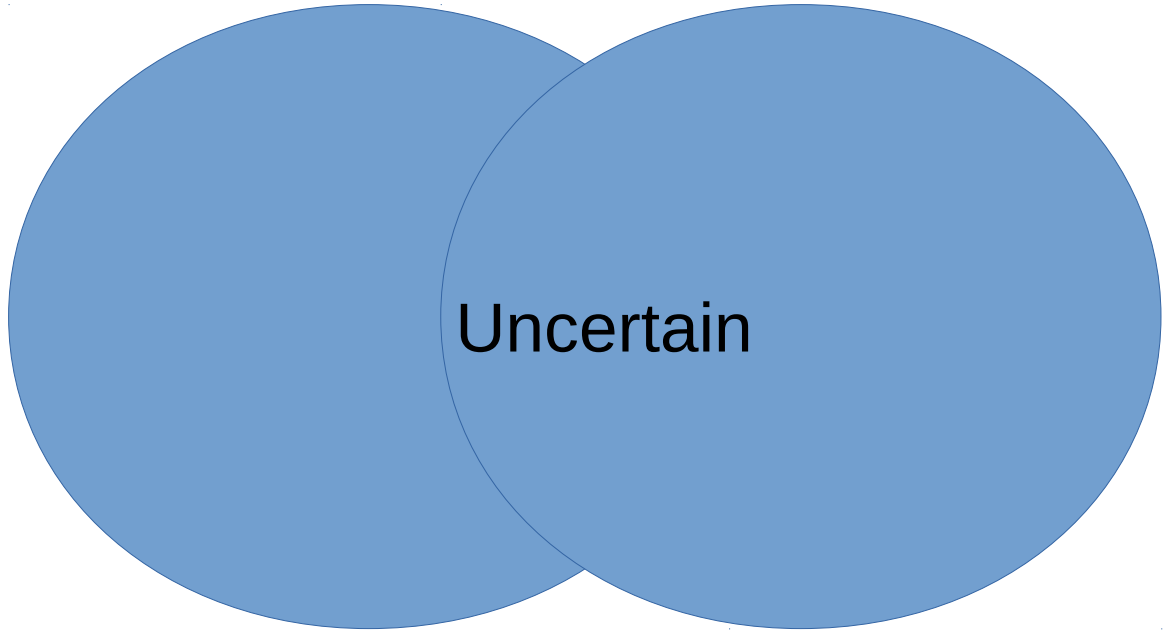
Clearly not a Firmware



Challenge: Firmware Identification

Clearly a Firmware

Clearly not a Firmware



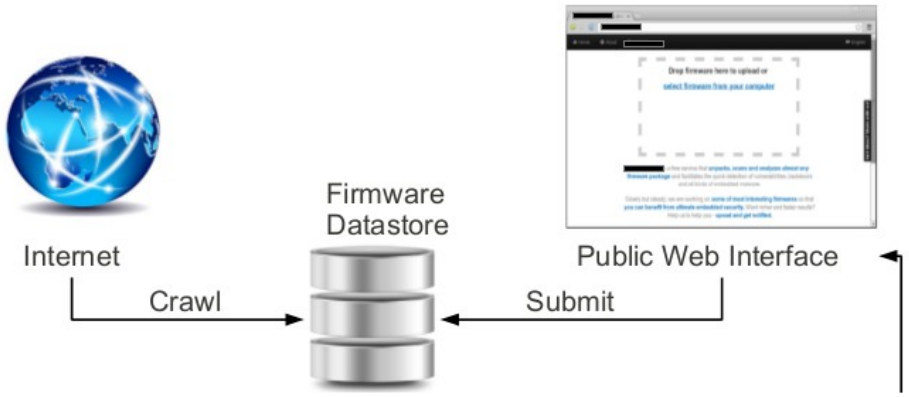
Challenge: Firmware Identification

- E.g., upgrade by printing a PS document

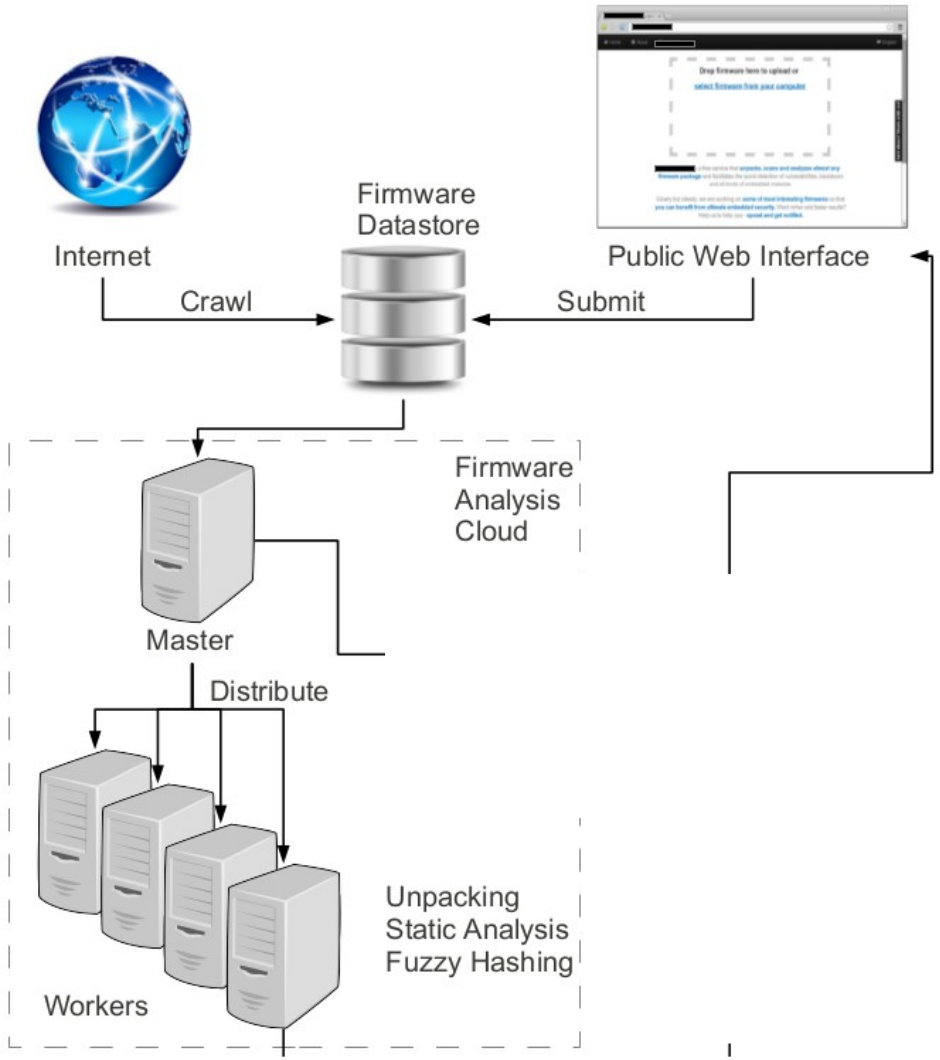
Challenge: Unpacking & Custom Formats

- How to reliably unpack and learn formats?
- E.g., vendor provides a .ZIP 'firmware package'
  - .ZIP → .EXE+.PS
    - .EXE → self-extracting archive
      - Extract more or not?
      - Turns out to contain a printer driver inside
    - .PS → ASCII85 stream → ELF file that could be:
      - A complete embedded system software
      - An executable performing the firmware upgrade
      - A firmware patch
- Often, a firmware image → just 'data' binary blob

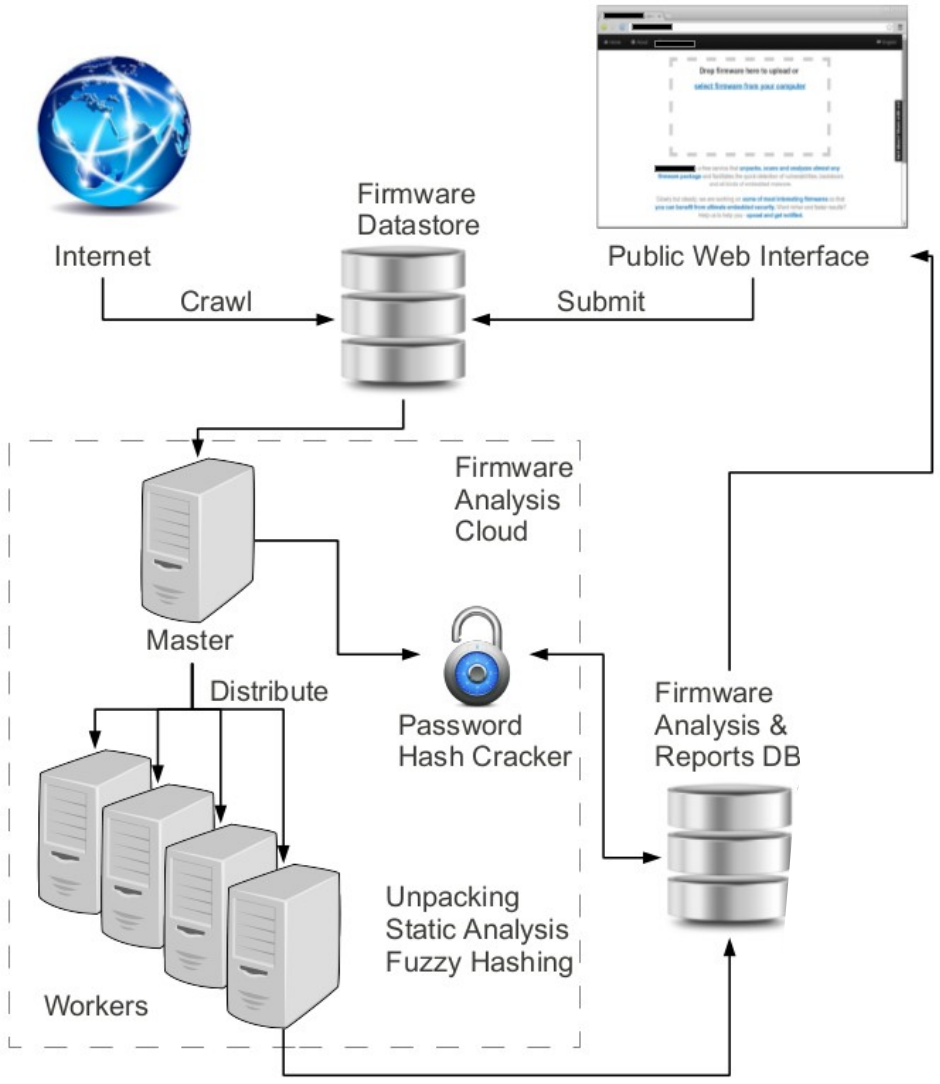
Architecture



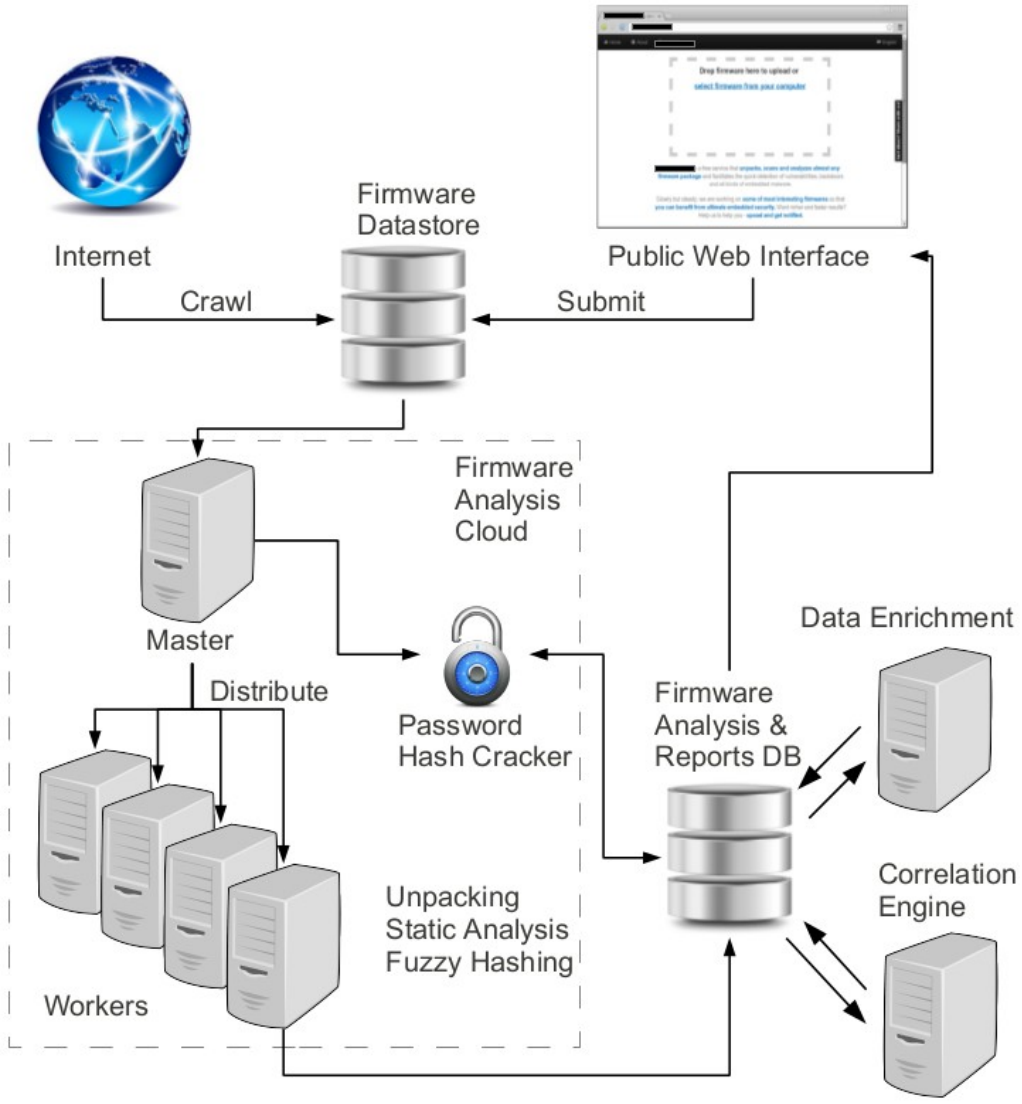
Architecture



Architecture



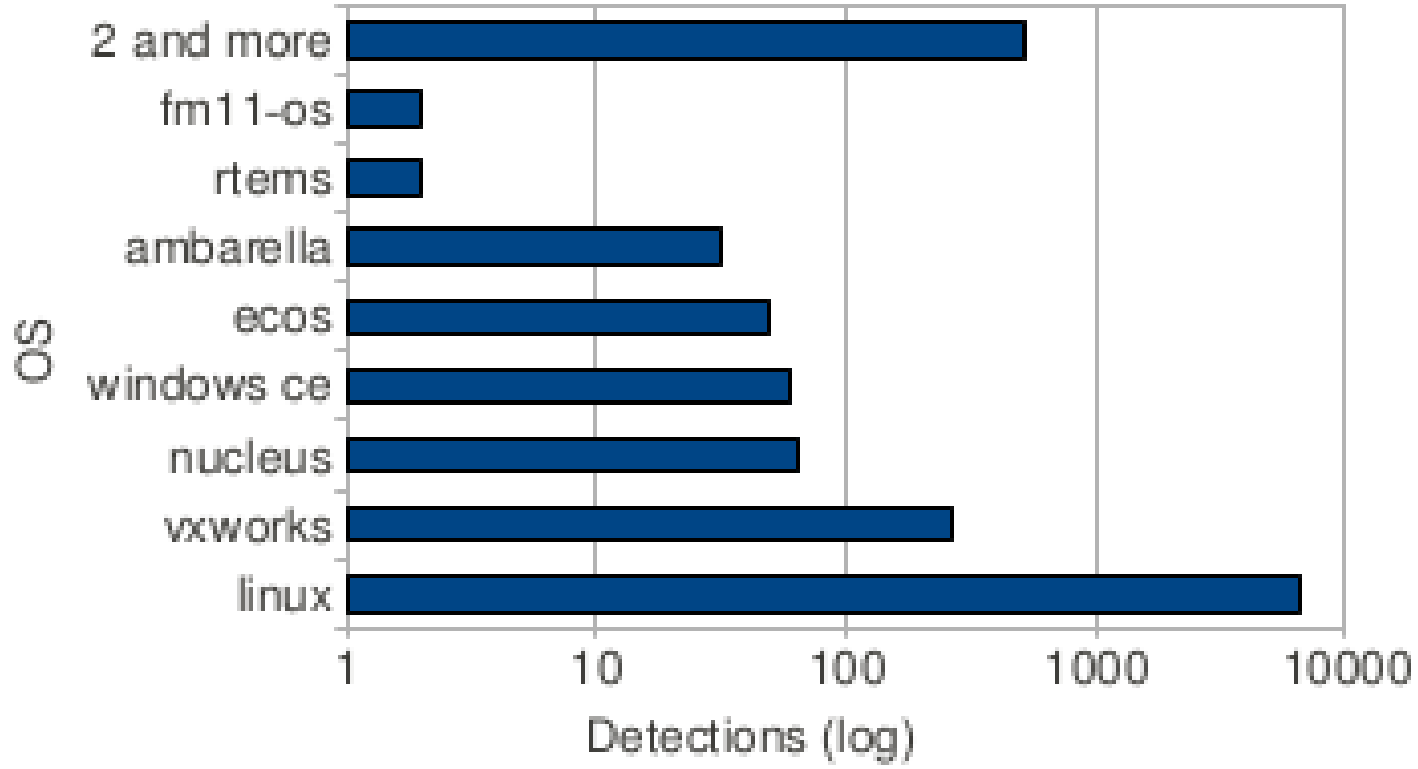
Architecture



Unpacking

- 759 K total files collected
  - ↓ Filter non firmware
- 172 K filtered files (firmware candidates)
  - ↓ Random selection
- 32 K firmwares analyzed
  - ↓ Unpack attempt
- 26 K firmwares unpacked (fully or partially)
  - ↓ Files extraction
- 1.7 M files after unpacking

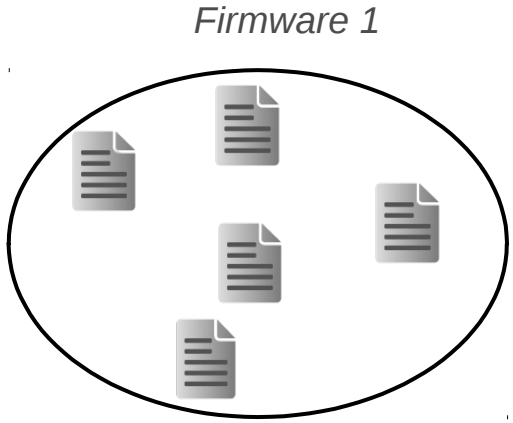
OS in our dataset



63 %ARM, 7 % Mips,
86 % Linux 7 % VxWorks/Nucleus RTOS/Windows CE

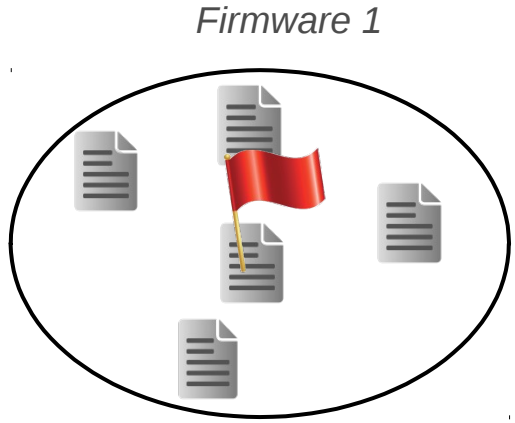
Example: Correlation

- Correlation via fuzzy-hashes (ssdeep, sdhash)
  - E.g., Vulnerability Propagation



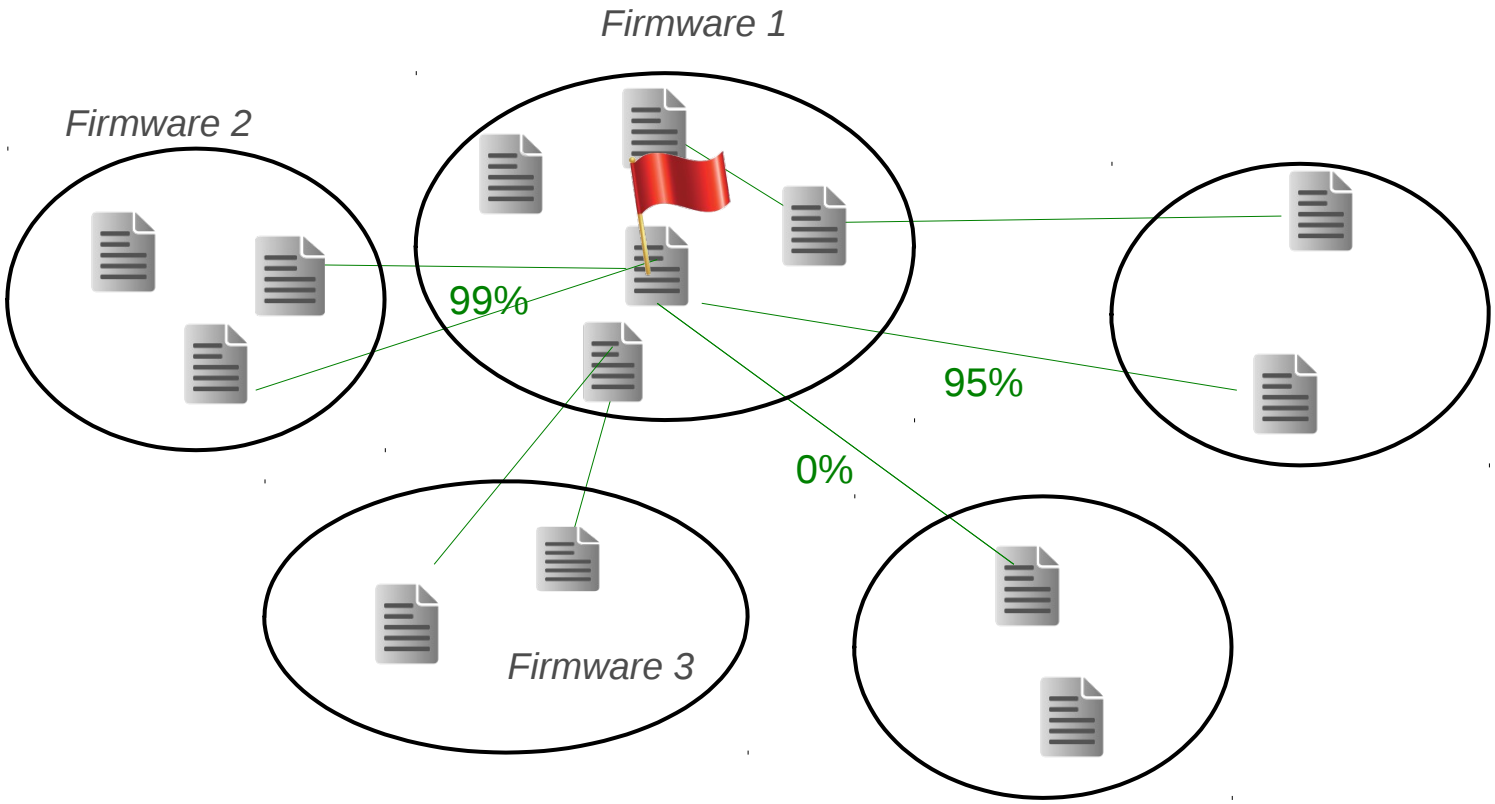
Example: Correlation

- Correlation via fuzzy-hashes (ssdeep, sdhash)
  - E.g., Vulnerability Propagation



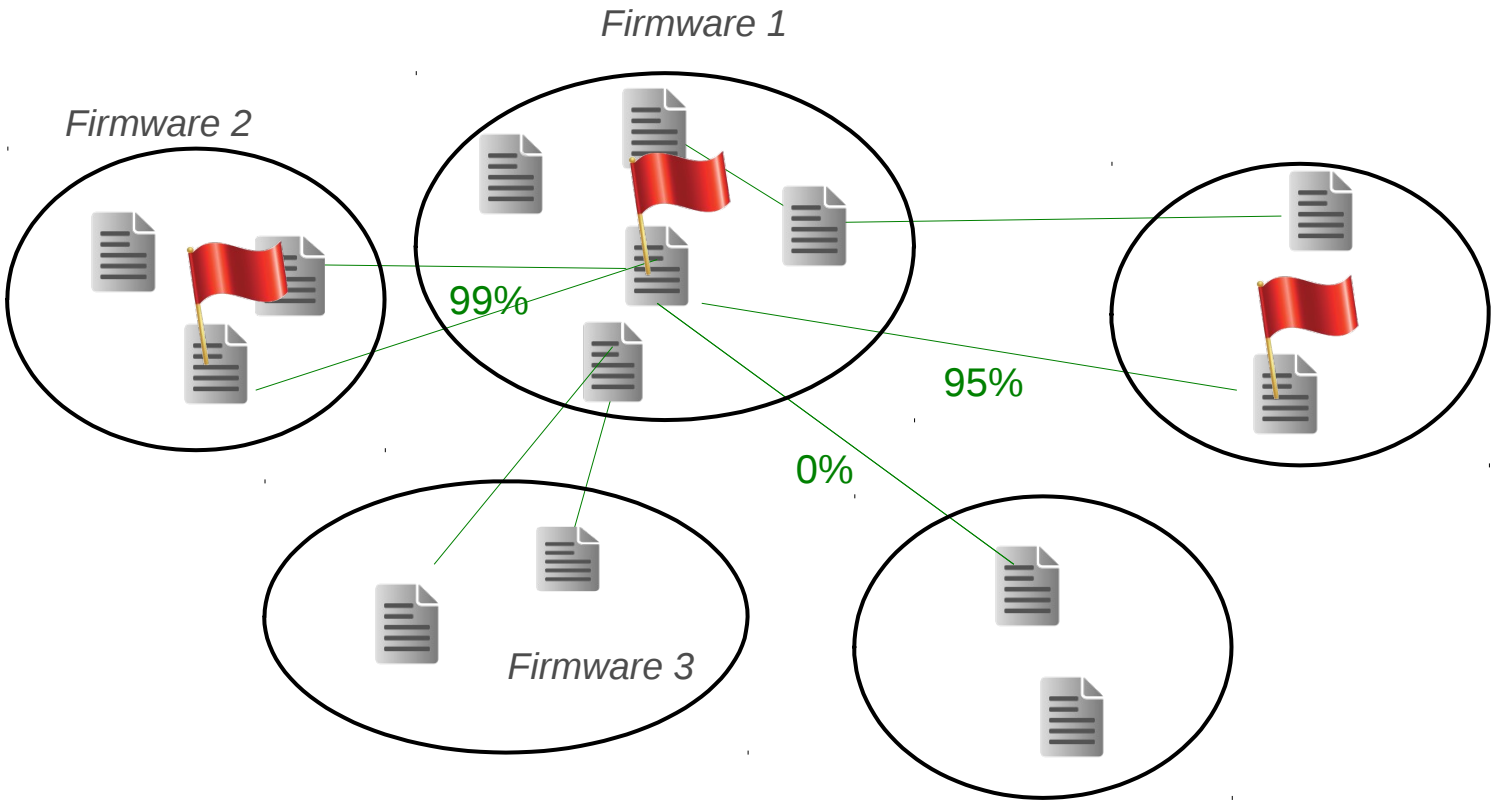
Example: Correlation

- Correlation via fuzzy-hashes (ssdeep, sdhash)
  - E.g., Vulnerability Propagation



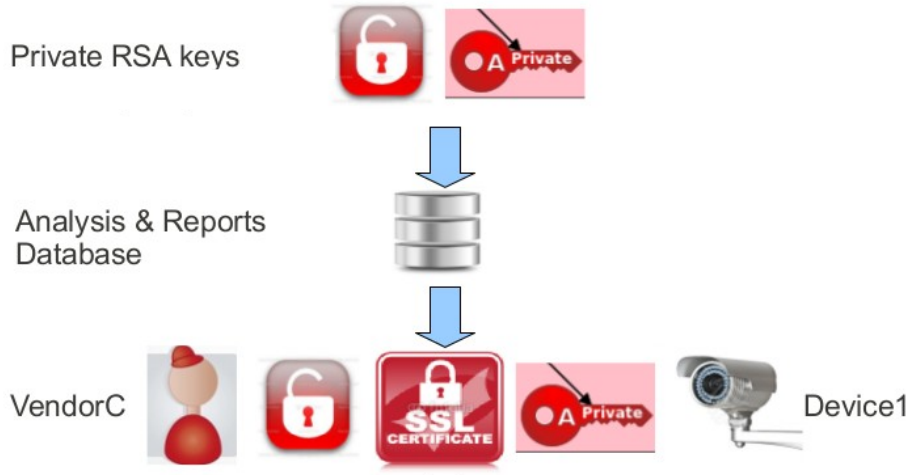
Example: Correlation

- Correlation via fuzzy-hashes (ssdeep, sdhash)
  - E.g., Vulnerability Propagation



RSA Keys

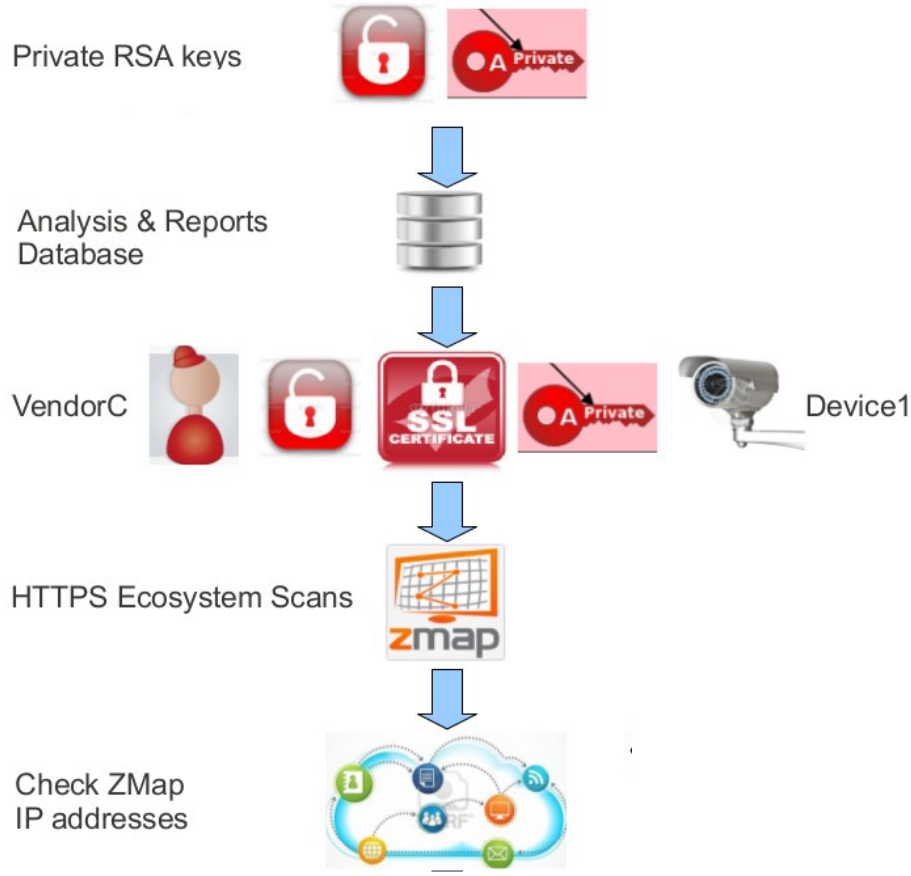
- SSL keys correlation vulnerability propagation



RSA Keys

- SSL keys correlation vulnerability propagation

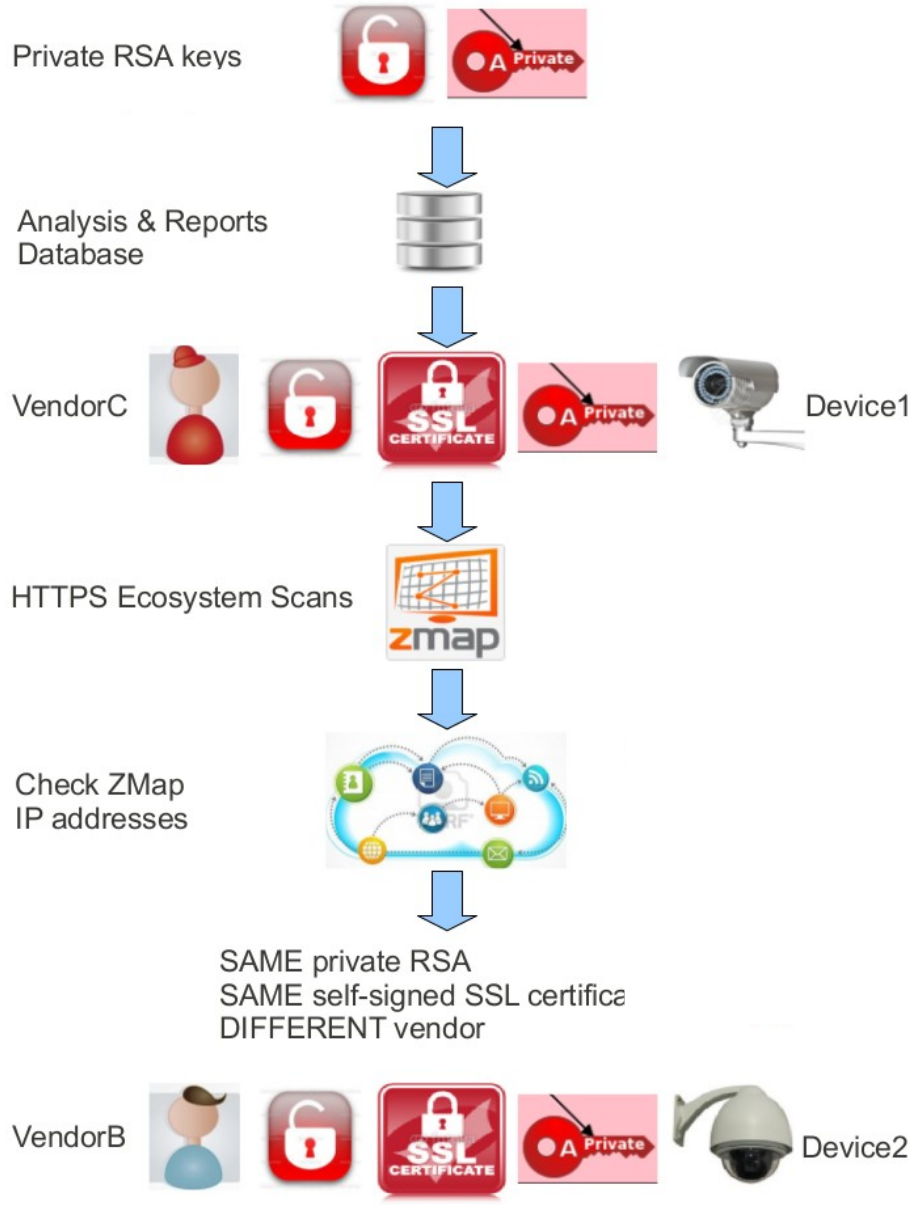
- 1 RSA private key: 30,000 vulnerable devices online



RSA Keys

- SSL keys correlation vulnerability propagation

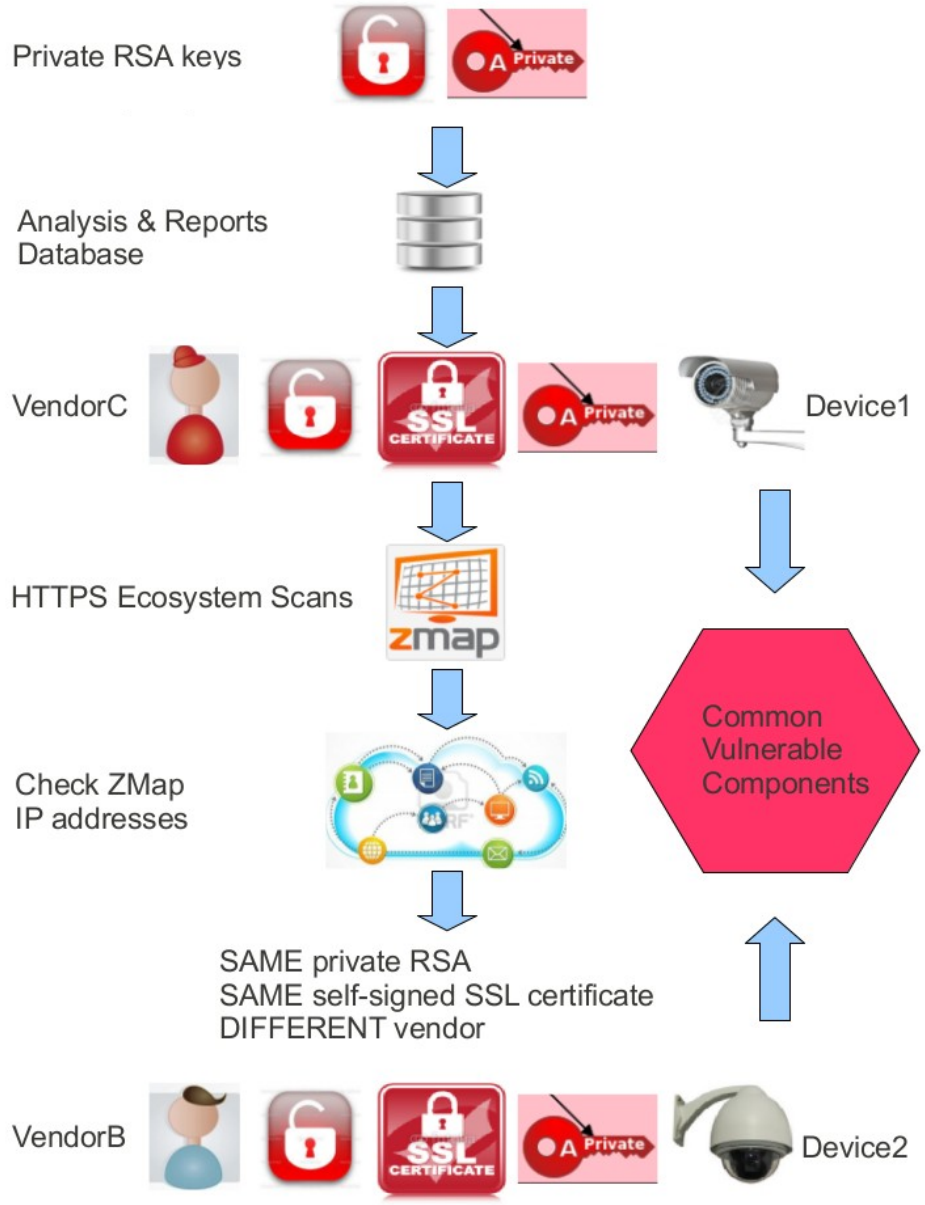
- 1 RSA private key: 30,000 vulnerable devices online
- Not all the same brand



RSA Keys

- SSL keys correlation vulnerability propagation

- 1 RSA private key: 30,000 vulnerable devices online
- Not all the same brand



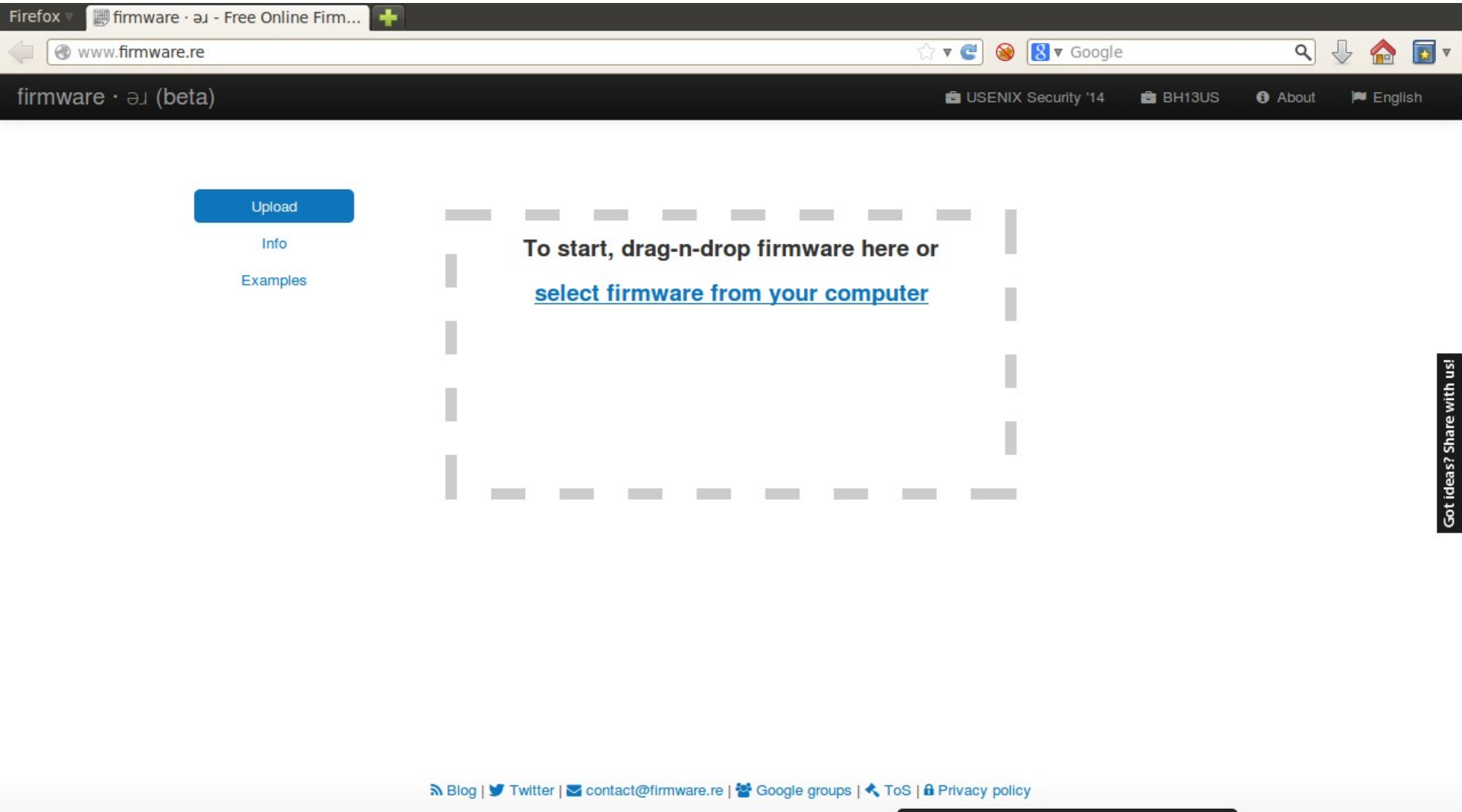
Another example of composition failure: Fireworks!

- Replacing wires by wireless in a system
- Lack of security
- Anyone can control the fireworks

- Fortunately firmware updates possible and now deployed

Costin et al. ACM Wisec 2014

www.firmware.RE (beta)



Results: Summary

- 38 new vulnerabilities (CVE)
- Correlated them to 140 K vulnerable online devices
- Affected 693 firmware files by at least one vulnerability

See our Usenix Security 2014 paper

Next steps

- Rather simple analysis so far

- We are now working on
  - Doing smarter analysis on this dataset
  - Improving the dataset

Take away message

Saltzer and Schroeder (1975)

Few of the principles from Saltzer and Schroeder (a.k.a. the basics):

- Economy of Mechanism (“KISS”)
- Fail-safe defaults
- Open design
- Separation of privilege
- Least privilege
- Psychological acceptability

Wrong Assumptions

- Forget things like “It will never be attacked because it is:
  - Stripped, binary only
  - Firmware is not on the Internet
  - Hardware is not documented
  - We disabled JTAG
  - …

Frequently found problems in firmware

- Updates with old software (release/compilation date)
- Default passwords
- SSL private keys
- SSH keys (authorized_keys)
- Debug access a.k.a. backdoors...
- Web vulnerabilities
- Building Images as root
- Packaging Outdated and Vulnerable Software

Defending / obfuscation (by far not an exhaustive list!)

- Plan for updates automated and secure
- Clean default passwords, keys
- Implement countermeasures (NX/Canaries/ASLR...)
- Secure boot, signed updates
- (contradictory) Please don't lock the user out!

- Obfuscating firmware
  - Remove strings, strip symbols
  - Make firmware hard to obtain (encrypted)
  - Careful key management
  - Make it hard to analyze hardware

Security Trade-off

Security is hard

- Costs money, Time, manpower

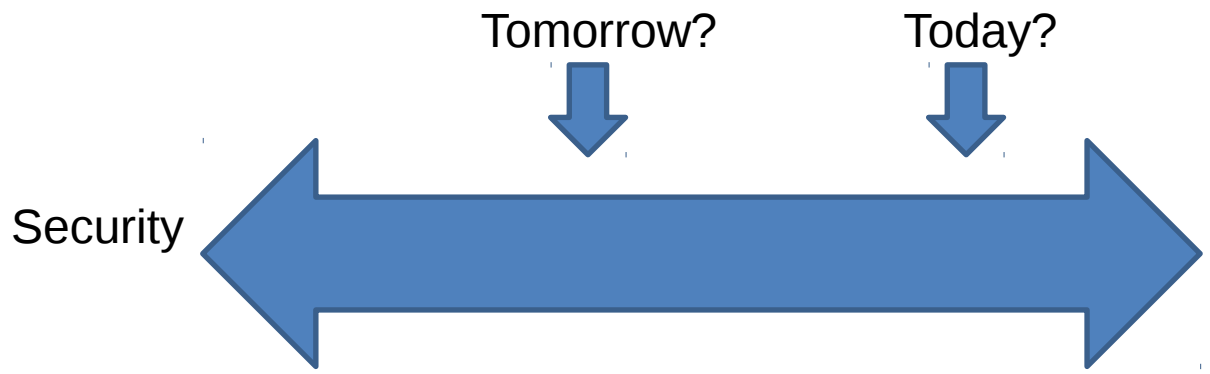
Features are selling points

- More Features, more attack surface and less security

Less features, higher cost, latter

- market failure ?

Long term v.s. short term thinking



Conclusion

A lot of poorly secured devices produced

- But bad cases more visible

Security is hard and expensive, we need

- Public/customer awareness
- Security Standards?
- Independent security audit
- Automated Firmware updates!

Diverse and powerful adversaries

No such thing as total security

- But there should be a minimal level of security

References

- Bunnie Huang blog: hardware attack
  - Hacking the PIC 18F1320 http://www.bunniestudios.com/blog/?page_id=40
  - Hacking the Xbox An introduction to Reverse engineering (free ebook!)
- http://travisgoodspeed.blogspot.com
- http://siliconpr0n.org/
- http://zeptobars.ru/
- Sergei Skorobogatov work http://www.cl.cam.ac.uk/~sps32/

References

- Oliver Kömmerling, Markus G. Kuhn: Design Principles for Tamper-Resistant Smartcard Processors, Proceedings of the USENIX Workshop on Smartcard Technology, 1999
- Printed Circuit Board Deconstruction Techniques, Joe Grand, WOOT '14
- Firmware unpacking tools BAT and Binwalk
- Talk about hardware attacks on secuer chips “Hardware reverse engineering tools” Olivier Thomas, Recon 2013 https://www.youtube.com/watch?v=o77GTR8RovM

Some of our projects

Avatar Project :

- http://s3.eurecom.fr/tools/avatar/

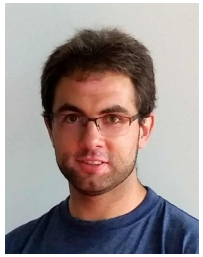
Firmware.re:

- http://www.firmware.re/

Our publications can be found here :
http://www.s3.eurecom.fr/publications.html

Questions ?

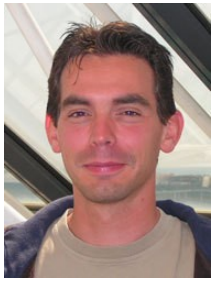
Some of the people working on this:



Jonas Zadach

Luca Bruno

Andrei Costin



Davide Balzarotti