Our mission

Enable the Internet of Tomorrow = Internet of Things + Security

Without security:

• Impossible to deploy a network of connected devices
• Impossible to scale the Internet of Things
• Impossible to trust a system to keep data private & confidential

- eHealth
- Smart Grid
- Smart Grid
- Smart Home
- Smart City
- Industry 4.0
- Connected Cars
July 2015 Miller & Valasek’s Attack

- Malicious connection to infotainment through Uconnect™.
- Malicious firmware update.
- Sending of fake / impersonating commands (commands for the conditioning, for the engine, ...)

⇒ Combination of logical problems on open architecture

Wired magazine 7/21/2015
Hackers can use communications with the external world to exploit errors in:

- Applicative layers,
- Protocols,
- Configuration,
- Personalization,
- Firmware update,
- Secure Boot,
- OS/Kernel,
- ...

Errors in security rationale
The Security Challenge

- Security chain:
  - Cryptographic algorithms
  - Cryptographic protocols
  - Physical attacks resistant subsystems (e.g. secure elements)
  - Robustness of the Trusted Computing Base (TCB) to logical attacks

- Issues with errors and vulnerabilities, particularly in operating systems:
  - An already alarming situation which is still degrading (e.g. the NIST database statistics).
The main challenge is to secure the software

• Hackers will exploit bugs, weaknesses and errors that exist in thousands in the software of embedded systems
• It is not possible to directly protect against attacks OSes such as iOS, Android, Linux, large RTOS ... There are issues with:
  • Size of the software stack to secure
  • “Trusted Computing Base” (TCB) includes kernel whose size and complexity are too big to build trust (and correctness of security properties)
• ➔ Issues & vulnerabilities, particularly in operating systems.
Security need evolution (1/2)

- Small TCB with few peripherals and small attack surface
  - Secure element is usually the right solution
  - Resistance to physical attack is the biggest challenge
- More peripherals and thus larger TCB and larger attack surface (typically mobile security)
  - Use a small secure OS/kernel (TEE),
  - Resistance to physical attack can be addressed with secure elements or similar embedded IP,
  - Resistance to logical attack becomes the biggest challenge
Security need evolution (2/2)

• IOT case: Still more peripherals, better business model for hackers, larger damages at stake, with large TCB and large attack surface, in many cases remote device is unattended, etc.
  • *Logical and Physical TCB are to be distinguished*
  • *Resistance to physical attack can still be addressed with secure elements or similar embedded IP*
  • *The secure OS/kernel (such as the TEE), and all other complex part of the TCB need to be formally verified*
  • *Resistance to logical attack is achieved using a trusted and reliable security rationale (Attacks exploit error in the security rationale)*
Prove & Run answer’s to the challenge

- Two critical secure COTS (ready for integration) that are needed to host “security sensitive” applications and build layered security perimeters:
  - **ProvenCore**: microkernel proven for security to secure gateways and connected devices (Industrial Things), smartphone, tablets, etc.
    - Execution of security critical applications
    - Secure protection of the “Smart and Safe world” (Existing OS)
    - Provided together with its **secure boot**
  - **ProvenVisor**: proven secure hypervisor for mobile devices and IoT virtualization solutions
    - Secure isolation of existing OS and legacy SW stack
  - **Built with ProvenTools**: a patented software development tool that makes it possible to formally prove the correctness of the software
    - Be as close as possible to “zero-bug”
Quality of Security Rationale is Essential

• The rationale of why security is achieved should be provided in an explicit and auditable format
  • Risk analysis,
  • Product security requirements,
Quality of Security Rationale is Essential

• The rationale of why security is achieved needs to be provided in an auditable format
  • Risk analysis,
  • Product security requirements,

Confidence in rationale is key
Quality of Security Rationale is Essential

• The rationale of why security is achieved needs to be provided in an auditable format
  • Risk analysis,
  • Product security requirements,
  • Identifying the TCB,
Quality of Security Rationale is Essential

- The rationale of why security is achieved needs to be provided in an auditable format
  - Risk analysis,
  - Product security requirements,
  - Identifying the TCB,
Quality of Security Rationale is Essential

- The rationale of why security is achieved needs to be provided in an auditable format
  - Risk analysis,
  - Product security requirements,
  - Identifying the TCB,

**TCB should be small enough to be trustable**
Quality of Security Rationale is Essential

• The rationale of why security is achieved needs to be provided in an auditable format
  • Risk analysis,
  • Product security requirements,
  • Identifying the TCB,

Large Oses such as Linux, Or Android when used should not be part of the TCB
Quality of Security Rationale is Essential

• The rationale of why security is achieved needs to be provided in an auditable format
  • Risk analysis,
  • Product security requirements,
  • Identifying the TCB,

Layered architectures highly recommended
Quality of Security Rationale is Essential

Rich OS based system (Linux, QNX, Android, ...)

Processor
Processor
Processor
Processor

IOC

Trusted Computing Base
Quality of Security Rationale is Essential

Rich OS based system (Linux, QNX, Android, ...)

- Auth
- ID
- IP
- FW1
- FW2
- SB
- FU

Trusted Computing Base
Quality of Security Rationale is Essential

Rich OS based system (Linux, QNX, Android, ...)

Trusted Computing Base

Auth ID IP FW1 FW2

SB FU

IOC ID IP

SB FU F1 F2

Processor

Processor

Processor

Processor
Quality of Security Rationale is Essential

Rich OS based system (Linux, QNX, Android, ...)

Trusted Computing Base
Rich OS based system (Linux, QNX, Android, ...)

- Auth
- ID
- IP
- FW1
- FW2
- SB
- FU

Trusted Computing Base
Remote attacks exploit entry points
Remote attacks exploit entry points

SB: Secure Boot
FU: Firmware Update
FW: Firewall
SS: Secure Storage
CL: Crypto Library
AUT: Authentication
Remote attacks exploit entry points
Introduction to TrustZone
TrustZone ARM Cortex A – High Level Principles

Normal World  Secure World

I/O devices can be configured to be controlled by Secure World
TrustZone ARM Cortex A – High Level Principles

User Mode Mode

Kernel Mode

Hypervisor Mode

Monitor Mode

TrustZone™ Monitor

I/O devices can be configured to be controlled by Secure World
TrustZone ARM Cortex A – High Level Principles

User Applications

Rich OS (Linux, Windows ..)

Hypervisor Mode

User Mode

Kernel Mode

Monitor Mode

I/O devices can be configured to be controlled by Secure World
TrustZone ARM Cortex A – High Level Principles

User Applications

Rich OS (Linux, Windows ..)

Hypervisor Mode

Security Applications

ProvenCore

Monitor Code

TrustZone™ Monitor

I/O devices can be configured to be controlled by Secure World
TrustZone ARM Cortex A – High Level Principles

- User Applications
- Rich OS (Linux, Windows..)
- Hypervisor Mode
- Security Applications
- ProvenCore

Monitor Code

TrustZone™ Monitor

I/O devices can be configured to be controlled by Secure World
ARM Cortex M – v8 (Next Generation)

Normal World

Secure World

I/O devices can be configured to be controlled by Secure World
I/O devices can be configured to be controlled by Secure World.
I/O devices can be configured to be controlled by Secure World.
I/O devices can be configured to be controlled by Secure World
Rich OS based system (Linux, QNX, Android, ...)
Quality of Security Rationale is Essential

- The rationale of why security is achieved needs to be provided in an auditable format
  - Risk analysis,
  - Product security requirements,
  - Identifying the TCB,
- Confidence in the rationale becomes the key to security
  - Some parts of the rationale can be informal and be tested and/or evaluated using traditional approaches,
  - For some others trust and confidence in the absence of errors can only be achieved using formal proof.
  - This is the case of kernels that are part of the TCB
Quality of Security Rationale is Essential

Rich OS based system (Linux, QNX, Android, ...)

TrustZone™ Secure World

Cortex A

ProvenCore

FU ID IP FW1

Security Rational

Formal proof needed
Looking more closely to the Secure Remote Firmware Update

Firmware Update Preparation
Firmware Update / Boot

Normal OS

ProvenCore

TrustZone™ Secure World

Cortex A

Security Rational

Formal proof needed
Looking more closely to the Secure Remote Firmware Update

Firmware Update Preparation

Normal OS

ProvenCore

Cortex A

TrustZone™ Secure World

Security Rational

Formal proof needed
Looking more closely to the TCP/IP FW

TrustedZone™ Secure
World

ProvenCore

Linux

VPN/NTCB
TLS/NTCB
TCP/IP Stack

Cortex A

Formal proof needed

Security Rational
Using a Hypervisor
Wrong expectations about Hypervisors
Prove & Run answer’s to the challenge

- An hypervisor is used to virtualize hardware
  - Either because you want to replace two or more processors by a single one
  - Or because you want to have more virtual chips to isolate software stacks.
- It is thus important to do it securely and this is why we need a really secure hypervisor such as ProvenVisor

But an hypervisor is just not enough
A Formally Proven OS/Kernel is required

Any unit with external communication

- Wifi
- BLE
- 4G
A Formally Proven OS/Kernel is required

Hypervisor might be useful
A Formally Proven OS/Kernel is required

But is not sufficient
A Formally Proven OS/Kernel is required

- A kernel such as the one of Linux, QNX, Android, etc. is too large to be practically proven or shown to be secure,
  - Hackers will always find weaknesses to exploit.
- So such a kernel cannot be put directly in contact with the external world,
- A (secure) hypervisor is not the solution for that problem either
  - As you have to let the same communication media open
Some sub-systems remain in contact
A Formally Proven OS/Kernel is required

- Secure applications are needed to implement
  - firewalling (low level and high level),
  - Secure application management,
  - Secure (OTA) firmware update,
  - Secure authentication,
  - Etc.
A Formally Proven OS/Kernel is required

- Such applications cannot be implemented on bare metal
  - Otherwise too complex and error prone => would need to be formally proven by themselves,
- They cannot be implemented on a non proven kernel (otherwise same problem again).
- They can only be implemented on top of a formally proven kernel that is close to zero bug.
Implementation (Main Use Case)

Android with its applications

ProvenCore

Formally Proven
Implementation (Other Possibility)

Android with its applications

ProvenCore

Secure Hypervisor
VM

Formally Proven
Implementation  (To further isolate applications)

Android with its applications

Android with its applications

ProvenCore

Secure Hypervisor

TrustZoneTM

Potentially Formally Proven

Formally Proven
A secure OS/Kernel is required

• You need to have security applications to do various tasks:
  • Filtering on various communications channels, using and managing keys, administrating configurations and security, logging events, possibly performing various analysis and attack responses, ...

• You need to place such secure applications on a trusted and robust ground:
  • Not on large untrusted OS, Not on Linux (even sitting on a hypervisor, as it will have to communicate and interact with the peripherals and is this thus vulnerable)
  • Not on hardware,
  • Not on an hypervisor (which would provide by definition a similar hardware abstraction)
Prove & Run answer’s to the challenge

• Two critical secure COTS (ready for integration) that are needed to host “security sensitive” applications and build security perimeters:
  • *ProvenCore*: microkernel proven for security to secure gateways and connected devices (Industrial Things), smartphone, tablets, etc.
    • Execution of security critical applications
    • Secure protection of the “Smart and Safe world” (Existing OS)
    • Provided together with its *secure boot*
  • *ProvenVisor*: proven secure hypervisor for mobile devices and IoT virtualization solutions
    • Secure isolation of existing OS and legacy SW stack
  • *Built with ProvenTools*: a patented software development tool that makes it possible to formally prove the correctness of the software
    • Be as close as possible to “zero-bug”
With ProvenCore and ProvenVisor, Secure a Smart and Safe Embedded World

Formally Proven Security and CC EAL7 ready

The 2 missing bricks needed to create the Internet of Tomorrow
Conclusion

• With a **secure boot** and one or two COTS you can secure virtually any architecture:

  • *ProvenCore* : a microkernel proven for security.
    • Execution of security critical applications (firewalling, FOTA, etc.)
    • Secure protection of the “Smart and Safe world” (Existing OS)

  • *ProvenVisor* (optional) : a proven secure
    • Secure isolation of existing OS and legacy SW stack

  • **Built with ProvenTools:**
    • To be as close as possible to “zero-bug”