Proven Security for the Internet of Things (IoT)

77, avenue Niel, 75017 Paris, France

contact@provenrun.com
Prove & Run’s 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
July 2015 Miller & Valasek’s Attack

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

⇒ 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
Security is as strong as its weakest link

- **Security chain:**
  - Cryptographic algorithms
  - Cryptographic protocols
  - Technology and know-how to resist physical attacks
    - Ex: Smartcards
  - Technology and know-how to resist **logical attacks**
    - Hackers will exploit bugs, weaknesses and errors that exist in thousands in the software of embedded systems, in particular Operating Systems.
    - Existing OSs such as Android, Linux and large RTOSs cannot be technically secured and used as such:
      - 1000’s of bugs officially reported / year
Security is changing …

• Traditional: 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: the IoT disruption

• 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 parts 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)
Addressing the New Challenge

• Use of a state-of-the-art security methodology to clearly identify the security issues of the targeted system
  • For example the Common Criteria methodology
  • The rationale of why security is achieved needs to be provided in an auditable format:
    • Perform a Risk Analysis
    • Confidence in rationale is key
    • Identify the “Trusted Computing Base” (TCB)
    • TCB should be small enough to be trustable
    • Large OSs such as Linux or Android when used should not be part of the TCB
  
• For the OS and kernels that are included in the TCB;
  • Apply formal methods to the complex part of the TCB (this includes kernels)
  • Ability to get as close as possible to “Zero-Bug”
  • Ability to demonstrate security (proof and certification)

• Reach the highest levels of security at cost/skills requirements compatible with value chain constraints
  • Reuse COTS to control the cost of developing a secure product
Prove & Run answer’s to the challenge

• Two critical secure COTS (ready for integration) that are needed to host “security sensitive” applications and to build layered security perimeters:
  • *ProvenCore*: Microkernel proven for security to secure gateways and connected devices (Industrial Things), smartphones, 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 OSs 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”
Remote attacks exploit entry points

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

- SB: Secure Boot
- FU: Firmware Update
- FW: Firewall
- SS: Secure Storage
- CL: Crypto Library
- AUT: Authentication

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

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

I/O devices can be configured to be controlled by Secure World

Formal proof needed
Securing an Entry Point on ARM Cortex-A

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

Cortex-A with TrustZone™

ProvenCore

TrustZone™ Secure World

SS	FU	FW	AUT

Secure Boot

Security rationale

Formal proof needed

Prove & Run
Looking more closely to the Secure Remote Firmware Update

- Rich OS based system (Linux, QNX, Android, ...)
- ProvenCore
- TrustZone™ Secure World
- Formal proof needed
- Security Rational

Cortex A

Firmware Update Preparation
Firmware Update / Boot

Secure Boot
Looking more closely to the Secure Remote Firmware Update

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

Firmware Update Preparation

ProvenCore

Secure Boot

TrustZone™ Secure World

Firmware Update/Boot

Security Rational

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

Looking more closely to the TCP/IP Firewall

- Formal proof needed
- Security Rational
- TrustZone™ Secure World

Linux

- VPN/NTCB
- TLS/NTCB
- TCP/IP Stack
- DXP

ProvenCore

- TLS/TCB
- FW1
- DV
- VPN/TCB

Cortex A

Secure Boot
ARM next-generation microcontrollers (Cortex-M v8)

Microcontroller OS

TrustZone™ Secure World

ProvenCore

Secure Boot

Cortex-M – v8

FU FW AM AUT

Security rationale

Trusted Computing Base

Prove & Run
Using a Hypervisor

- A1, A2, A3
- A4, A5, A6
- ProvenCore
- Cortex-A
- ProvenVisor
- Trusted Computing Base

Linux

 FW
 FU
 AM
 Auth
Using an Hypervisor

• An hypervisor may be used to virtualize hardware or create virtual hardware isolation
  • 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
But a secure OS kernel is required

- You need to have security applications to do various tasks:
  - Filtering various communications channels, Firmware Update (FOTA), Using and managing keys, Administrating configurations and security, Logging events, possibly Performing various analysis and attack responses, etc.

- You need to place such secure applications on a trusted and robust ground:
  - Not on a large untrusted OS such as Linux (even sitting on a hypervisor, as it will have to communicate and interact with the peripherals and is thus vulnerable)
  - Not on hardware,
  - Not on a hypervisor (which would provide by definition a similar hardware abstraction)
Conclusion

• With a **secure boot** and ProvenCore you can cope with a very large set of security issues:
  
  - **ProvenCore**: A microkernel proven for security
    - Execution of security critical applications (firewalling, FOTA, etc.)
    - Secure protection of the “Smart and Safe World” (Existing OS)
  
  • For more sophisticated cases, you may need to have a secure hypervisor
    
    - **ProvenVisor**: A proven secure hypervisor
      - Secure isolation of existing OS and legacy SW stack
    
    • **ProvenCore and ProvenVisor are built with ProvenTools**: To be as close as possible to “zero-bug”