RSAConference2016

Abu Dhabi | 15–16 November | Emirates Palace

Connect to Protect

SESSION ID: CCS-W04

Live Demo: A New Hardware-Based Approach to Secure the Internet of Things

Cesare Garlati

Chief Security Strategist
prpl Foundation
@CesareGarlati

#RSAC

Securing the Internet of (broken) Things

CHARLIE & CHRIS

1.4M

FIAT CHRYSLER RECALLS 1.4 MILLION VEHICLES AUGUST 2015

HOSPIRA DRUG PUMP

FDA

STRONGLY ENCOURAGE TO DISCONTINUE USE OF THESE PUMPS - MAY 2015

BOEING 737/800

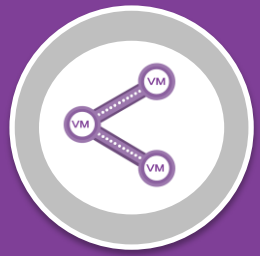
FBI

1. Reverse engineer proprietary software to expose vulnerabilities [Uconnect 8.4AN/RA4]
2. Exploit weak implementations of network protocols [D-BUS service port 6667]
3. Modify firmware and re-flash image to execute arbitrary code [TI OMAP-DM3730]
4. Laterally move from the compromised head unit to the target CAN system [CAN mcu Renesas v850]

OPEN SOURCE | INTEROPERABILITY | ROOT OF TRUST | VIRTUALIZATION

A New Hardware-based Approach

Micro Hypervisor
HW Virtualization



Secure Inter-VM
Communications

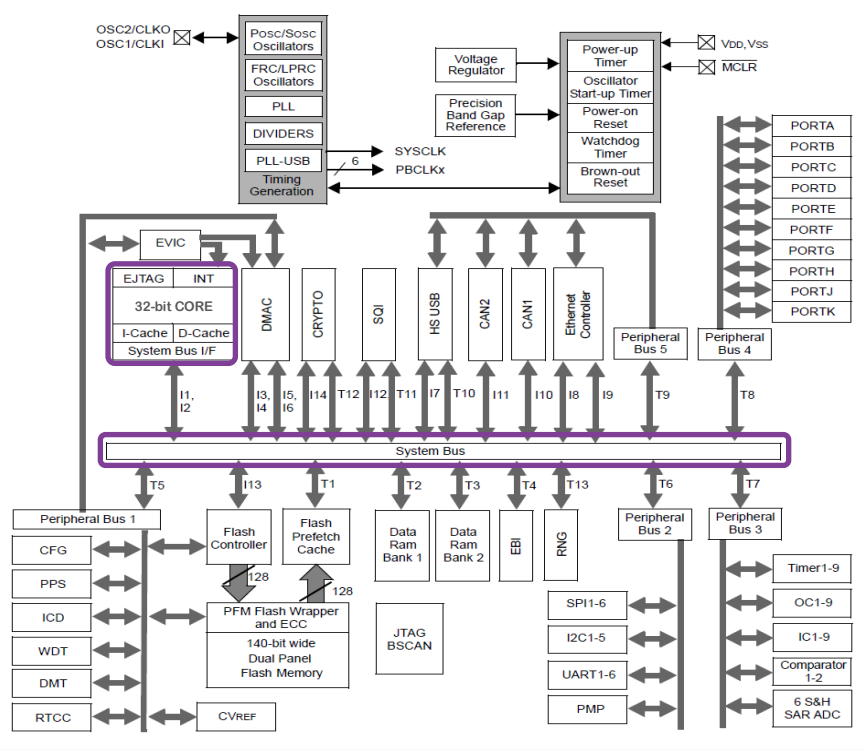
PUF Physically
Unclonable Funcs

Root of Trust
Secure Boot

Secure JTAG
In-circuit Debug

Multidomain security framework across hardware and software components

What is SoC Hardware Virtualization?



Virtualized SoC Example – IoT controller

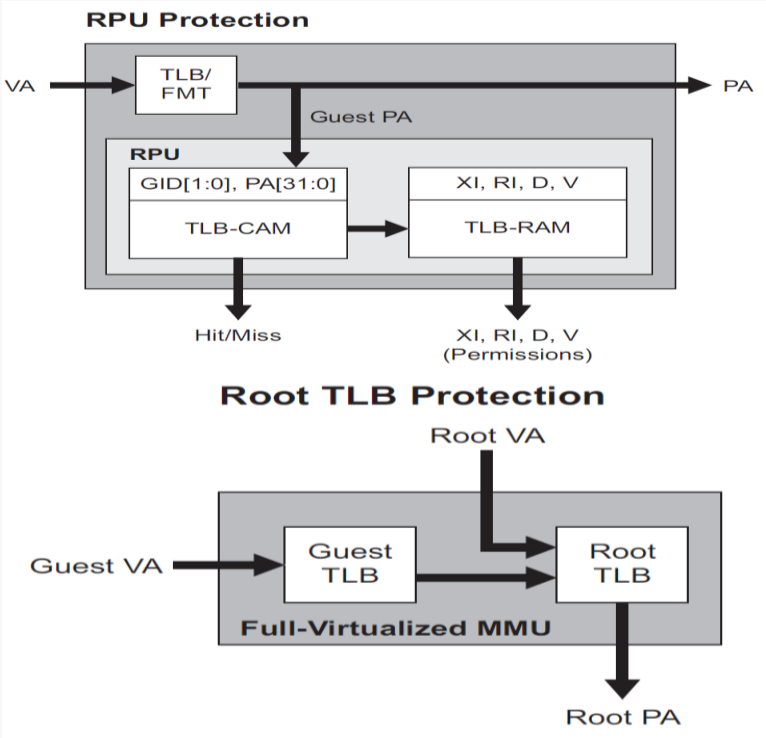
- ✓ CPU (shadow registers)
- ✓ Memory (MMU + RPU)
- ✓ System Bus Interconnect (Fabric + Guest ID lines))
- ✓ I/O (I/O MMU)
- ✓ DMA
- ✓ Micro kernel / hypervisor / root monitor

Two independent contexts <u>physically isolated</u>:

- **Guest** (OS) abstracts apps <> hardware
- **Root** (hypervisor) abstracts OS <> hardware

prpl

What is SoC Hardware Virtualization?



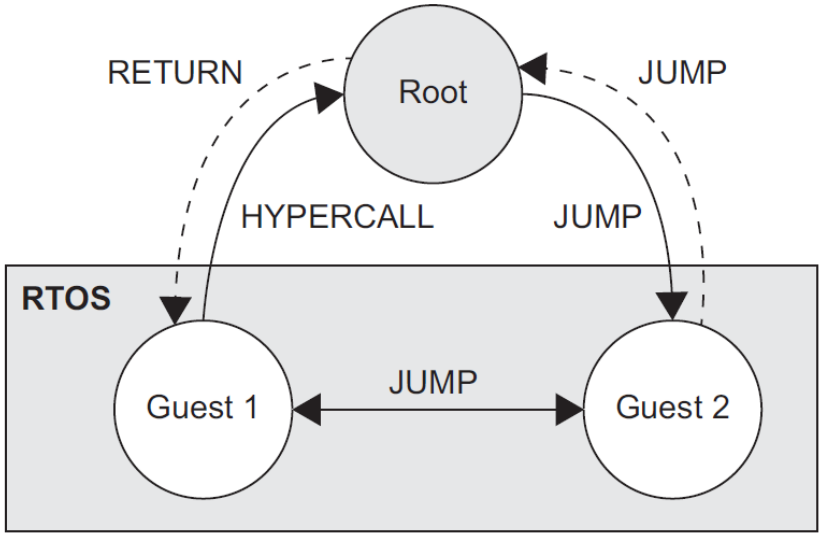
Controlling access to memory and MMIO

Guest virtual addresses VA are typically mapped to PA in a two-step process through one of the following methods:

1. Guest Fixed Mapping Table (FMT) and Root RPU
2. Guest TLB and Root RPU
3. Guest TLB and Root TLB

- ✓ The RPU technique provides guest-specific protection: read-inhibit (RI), write (D), execute inhibit (XI), and miss
- ✓ The RPU only supplies these page-level protection bits and does not provide address re-mapping.
- ✓ In contrast, the Root TLB supplies both address re-mapping and page-level protection bits

What is SoC Hardware Virtualization?

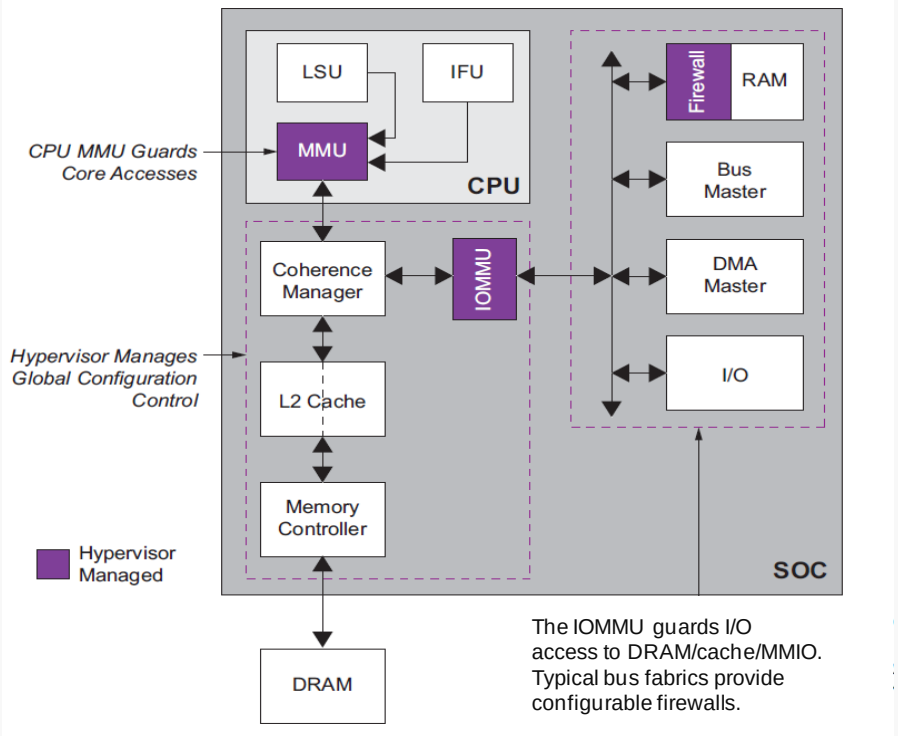


The RPU protects all transitions between guests within the same operating context (RTOS) or transitions from guest to root.

RPU Guarding Guest Address Space

- ✓ Access from any code segment to all other segments is guarded through a page-based memory protection scheme
- ✓ RPU setting determines whether one guest can access the address space of another
- ✓ RPU used for this purpose is programmed during the secure boot sequence
- ✓ The hypervisor is the only entity that can program the RPU, thus the RPU remains secure throughout runtime
- ✓ Guest-to-guest and guest-to-root calls are possible where permitted by RPU policy configuration
- ✓ One guest's code can jump to another guest's code for a function call only if the RPU allows it
- ✓ Otherwise, a guest can call root software to obtain permission to access another region in real time

What is SoC Hardware Virtualization?



Hypervisor-Managed MMU / IOMMU / RAM

SoC resources outside the core must be able to distinguish accesses by different guests and root – or to emulate

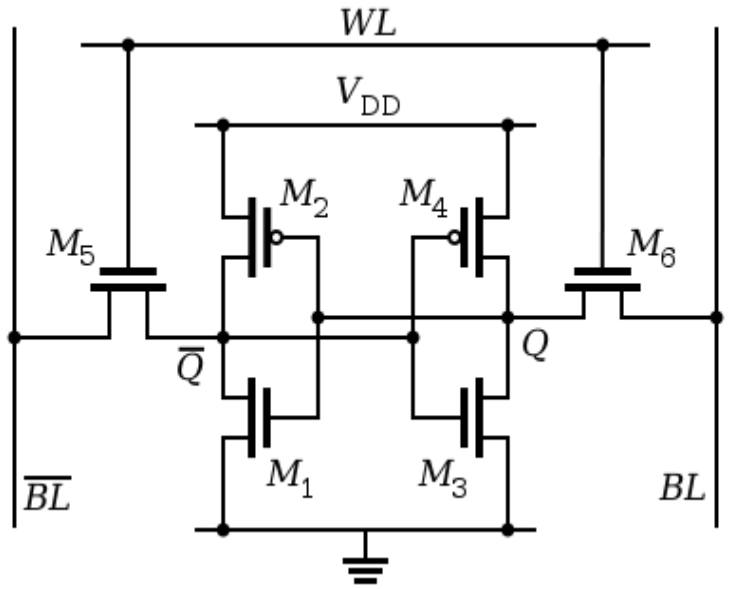
I/O device access:

- ✓ Static – guests have predefined access to a set of devices that is fixed at boot (pass-through)
- ✓ Root only – hypervisor has direct access to IO. Guests obtain access via hypercalls
- ✓ Dynamic - similar to static. Guests access is grant on demand. This method is considered a fully-virtualized model.

I/O MMU (firewall + address translation):

- ✓ legacy devices programmed with physical addresses
- ✓ DMA access on demand such as for PCIe

What is PUF - Physical Unclonable Functions?



Physical Unclonable Functions

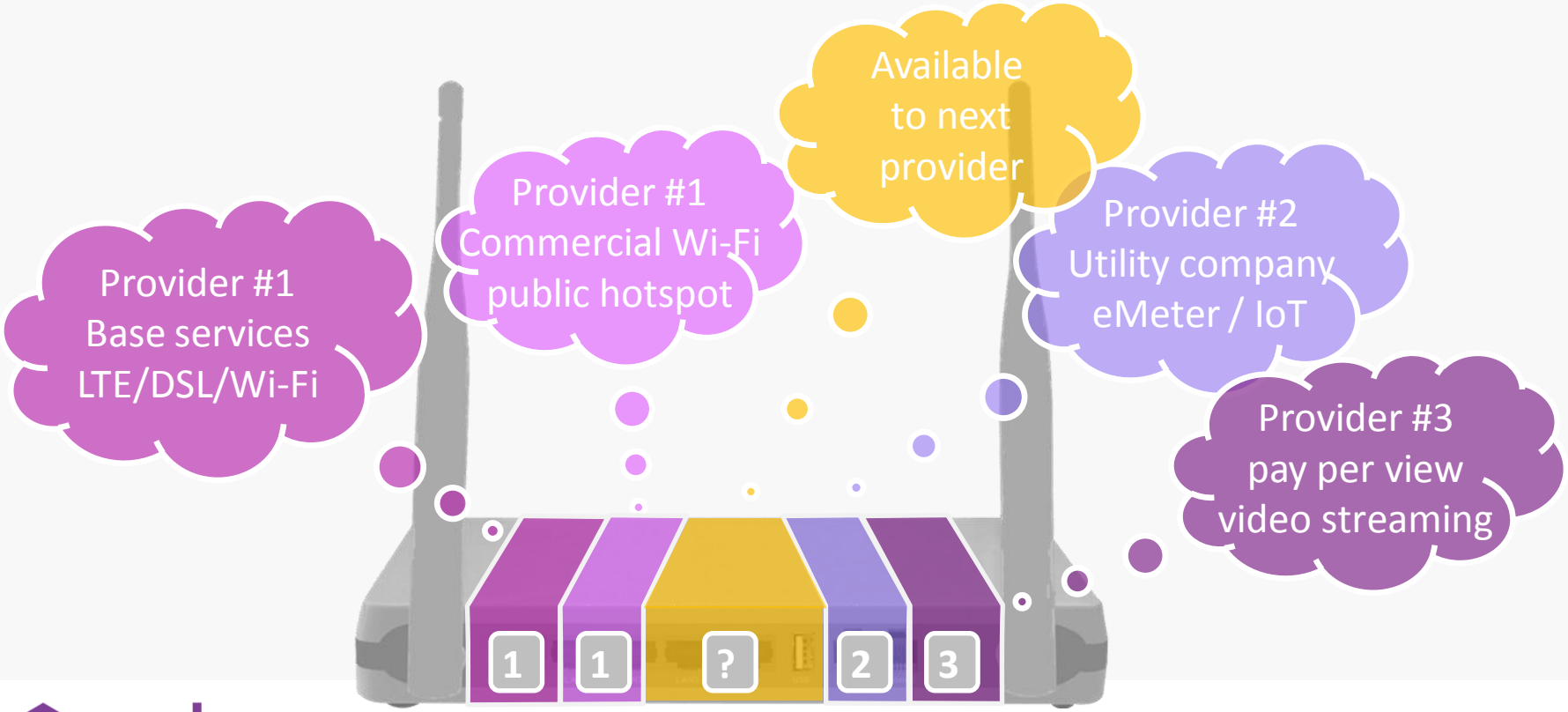
- ✓ A way to extract a unique repeatable identifier from every individual piece of silicon – die / chip fingerprint
- ✓ The uniqueness of the ID depends on random physical factors introduced during the manufacturing process
- ✓ These factors are unpredictable and uncontrollable which makes it impossible to duplicate or clone the ID
- ✓ When a physical stimulus (challenge) is applied the device reacts in an unpredictable (but repeatable) way (response)
- ✓ A fuzzy extractor or algorithm extract a unique strong cryptographic key from the physical microstructure.
- ✓ The same unique key is reconstructed every time the PUF is evaluated.
- ✓ No secret is ever stored in non volatile memory

What is PUF - Physical Unclonable Functions?

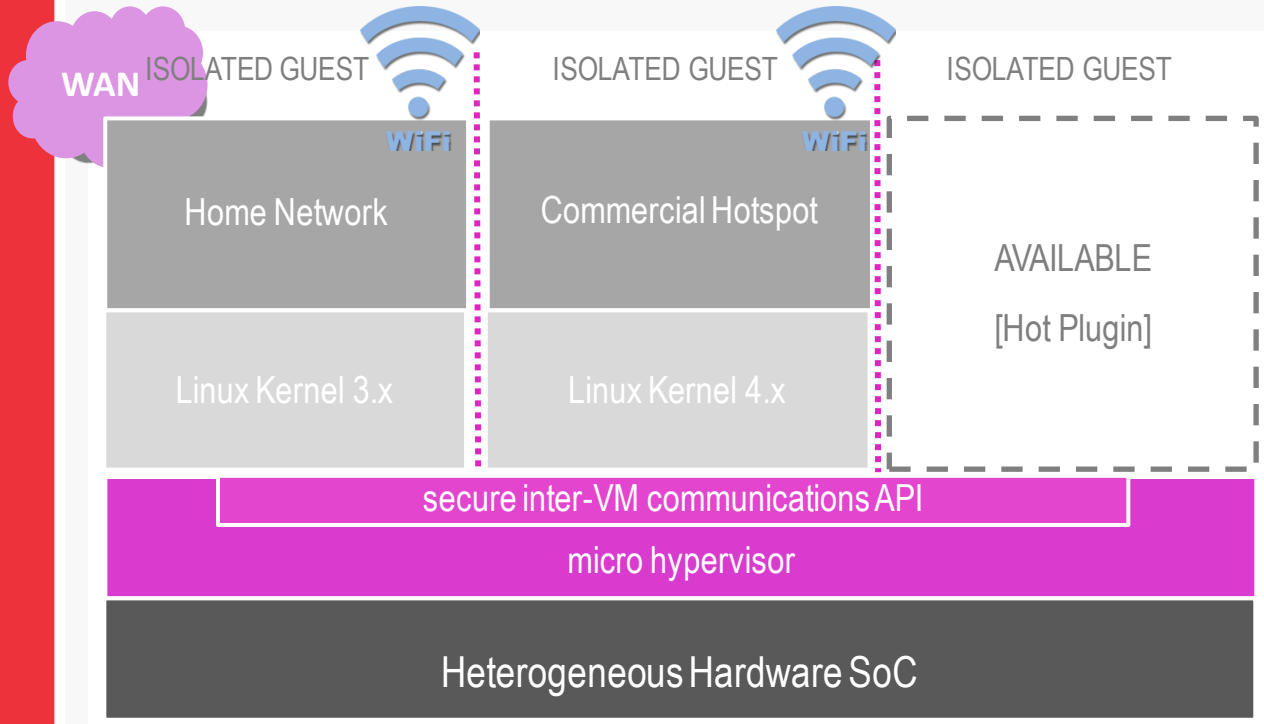
Static random-access memory PUF

- ✓ The initial state of an SRAM cell is a function of the process variation due to the silicon manufacturing process
- ✓ Each memory cell has a preference to start-up as either zero or one due to tiny mismatch in the cross-coupled inverters
- ✓ It is impossible to predict what cell has what preferred start-up state and some will always change – noise
- ✓ Keys derived from SRAM PUF are not stored 'on the chip' but extracted 'from the chip' and only when needed
- ✓ Once the initial state is read the SRAM can be used normally by the system
- ✓ This is a pure software approach that doesn't require modification to the manufacturing process
- ✓ The residual noise (approx 8%) can be used in true random generators or to add entropy to pseudo random generators

IoT Multitenant Use Case (Linux)

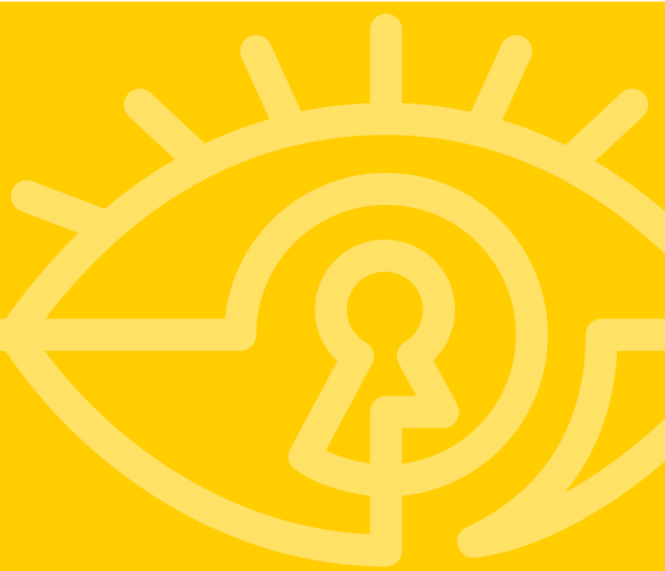


IoT Multitenant Use Case (Linux)



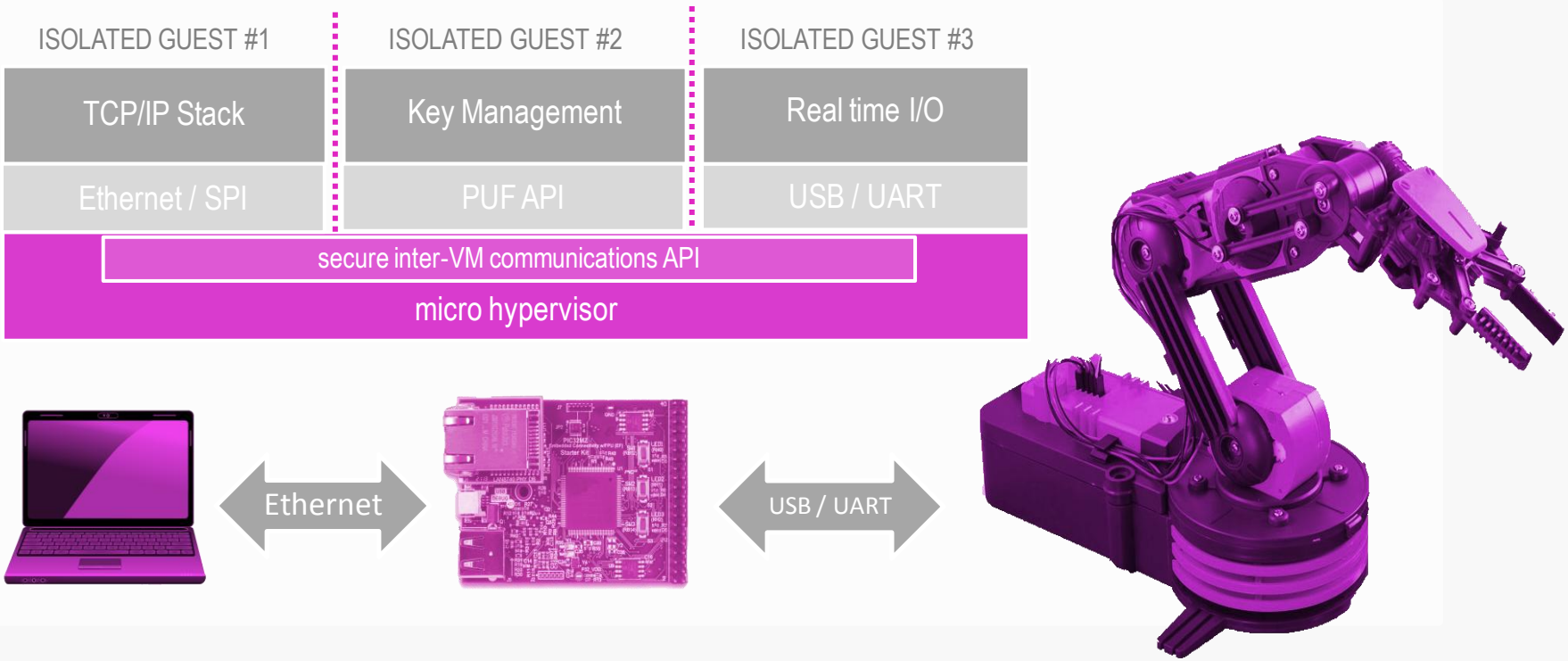
Multidomain Security

- ✓ New multitenant use cases – not just trusted/not-trusted islands
- ✓ Strong security model perfectly fits new multicore scenarios
- ✓ Hypervisor based – does not require rich OS modifications
- ✓ Open source framework and APIs – no royalties / control
- ✓ Reference framework open to ecosystem development



Live Demo

Live Demo – Application Concept



Live Demo – Building the firmware

1	*Clone GitHub repo*	2	*Make firmware*	3	*Flash firmware*

File Edit View Search Terminal Help

```
~$: git clone https://github.com/prplfoundation/prpl-hypervisor
~$: cd prpl-hypervisor
~/prplHypervisor$: git checkout demo-july-2016
```

Live Demo – Building the firmware

1 *Clone GitHub repo*

2 *Make firmware*

3 *Flash firmware*

```
File  Edit  View  Search  Terminal  Help

~$: git clone https://github.com/prplfoundation/prpl-hypervisor
~$: cd prpl-hypervisor
~/prplHypervisor$: git checkout demo-july-2016
~/prplHypervisor$: make
```

Live Demo – Building the firmware

1 *Clone GitHub repo*

2 *Make firmware*

3 *Flash firmware*

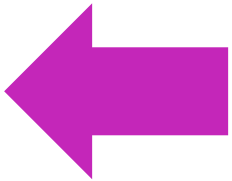
File Edit View Search Terminal Help

```
~$: git clone https://github.com/prplfoundation/prpl-hypervisor
~$: cd prpl-hypervisor
~/prplHypervisor$: git checkout demo-july-2016
~/prplHypervisor$: make
~/prplHypervisor$: make load
```

Live Demo – Running the demo

1	2	3
Ping 192.168.0.2	*Telnet 192.168.0.2*	*Send commands*

```
File  Edit  View  Search  Terminal  Help
cesare@cesare-pc:~$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=6.60 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.761 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=4.20 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=10.1 ms
64 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=0.745 ms
^C
--- 192.168.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.745/4.496/10.166/3.598 ms
```



Live Demo – Running the demo

1 *Ping 192.168.0.2*

2 *Telnet 192.168.0.2*

3 *Send commands*

```
File  Edit  View  Search  Terminal  Help
telnet 192.168.0.2 80
Trying 192.168.0.2...
Connected to 192.168.0.2.
Escape character is '^]'.
95a84a049651e231f6d358d0e6cb3af2010000000000000000000000000008051f26287be978cf8
399628ce365e9e8fe9a4328a95514c27
```

Live Demo – Running the demo

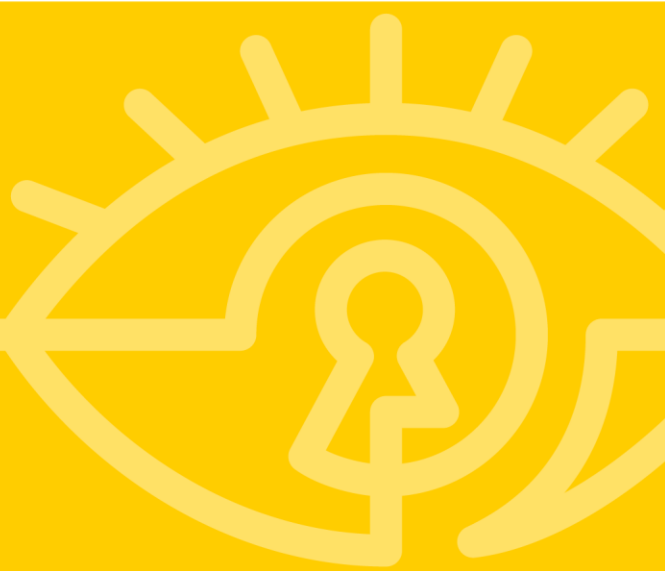
1 *Ping 192.168.0.2*

2 *Telnet 192.168.0.2*

3 *Send commands*

```
File  Edit  View  Search  Terminal  Help
telnet 192.168.0.2 80
Trying 192.168.0.2...
Connected to 192.168.0.2.
Escape character is '^]'.
95a84a049651e231f6d358d0e6cb3af2010000000000000000000000000008051f26287be978cf8
399628ce365e9e8fe9a4328a95514c27
95a84a049651e231f6d358d0e6cb3af2010000000000000000000000000008051f26287be978cf8
399628ce365e9e8fe9a4328a95514c271
95a84a049651e231f6d358d0e6cb3af2010000000000000000000000000008051f26287be978cf8
399628ce365e9e8fe9a4328a95514c272
```

prpl



Live Demo - End

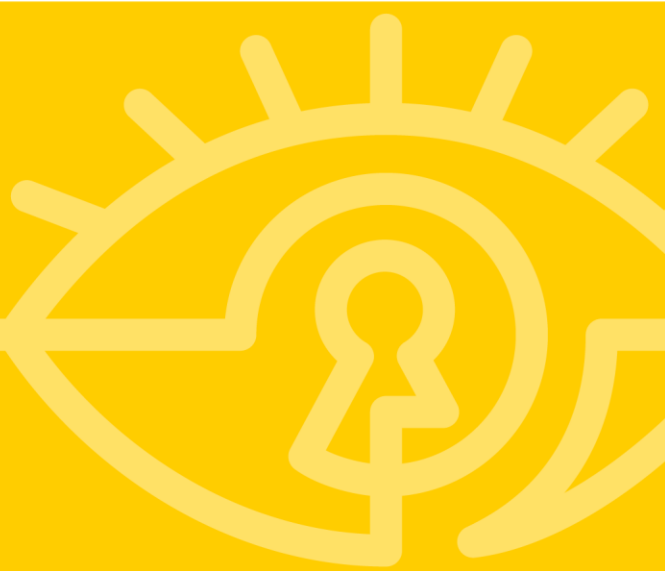
Takeaways

- IoT is already here but its security is fundamentally broken - and could soon result in human fatalities
- IoT security challenges include proprietary software, connectivity, firmware updates, lack of separation
- A new hardware security approach: open source APIs, interoperable protocols, virtualization, PUF, secure boot

1. *Security, more than anything else, will drive the next wave of IoT adoption*
2. *IoT security requires a multilayer approach that must include hardware-level security*
3. *Hardware security technology will become mainstream only if based on cross-vendor open standards*

Apply What You Have Learned Today

- Next week you should:
  - Identify critical IoT devices within your organization
- In the first three months following this presentation you should:
  - Assess the hardware security characteristics of these devices
  - Define appropriate patching and pen-testing programs
- Within six months you should:
  - Select IoT product and vendors compliant to multidomain security
  - Participate to cross-industry associations to drive IoT security standards



Thanks!

cesare@prplFoundation.org