
# In Silicon we Trust
# How to Fix the Internet of Broken Things

## IoT Security Foundation Congress 2016

Cesare Garlati, Chief Security Strategist, prpl Foundation

# 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**

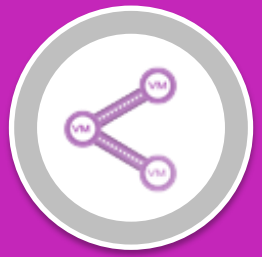
**FBI**


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

# prplSecurity™ Framework

prplHypervisor™
HW Virtualization



prplSecureVM™
Communications

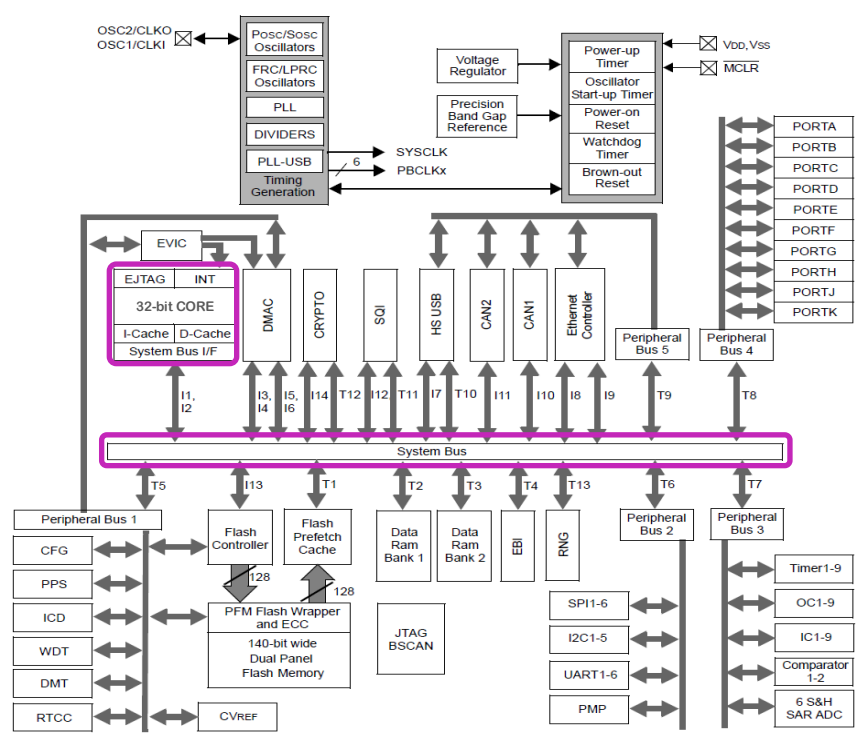
prplPUF™ API
Physically
Unclonable Funcs

prplSecureBoot™
Root of Trust

prplSecureJTAG™
In-circuit Debug

Multidomain security 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 physically isolated:

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

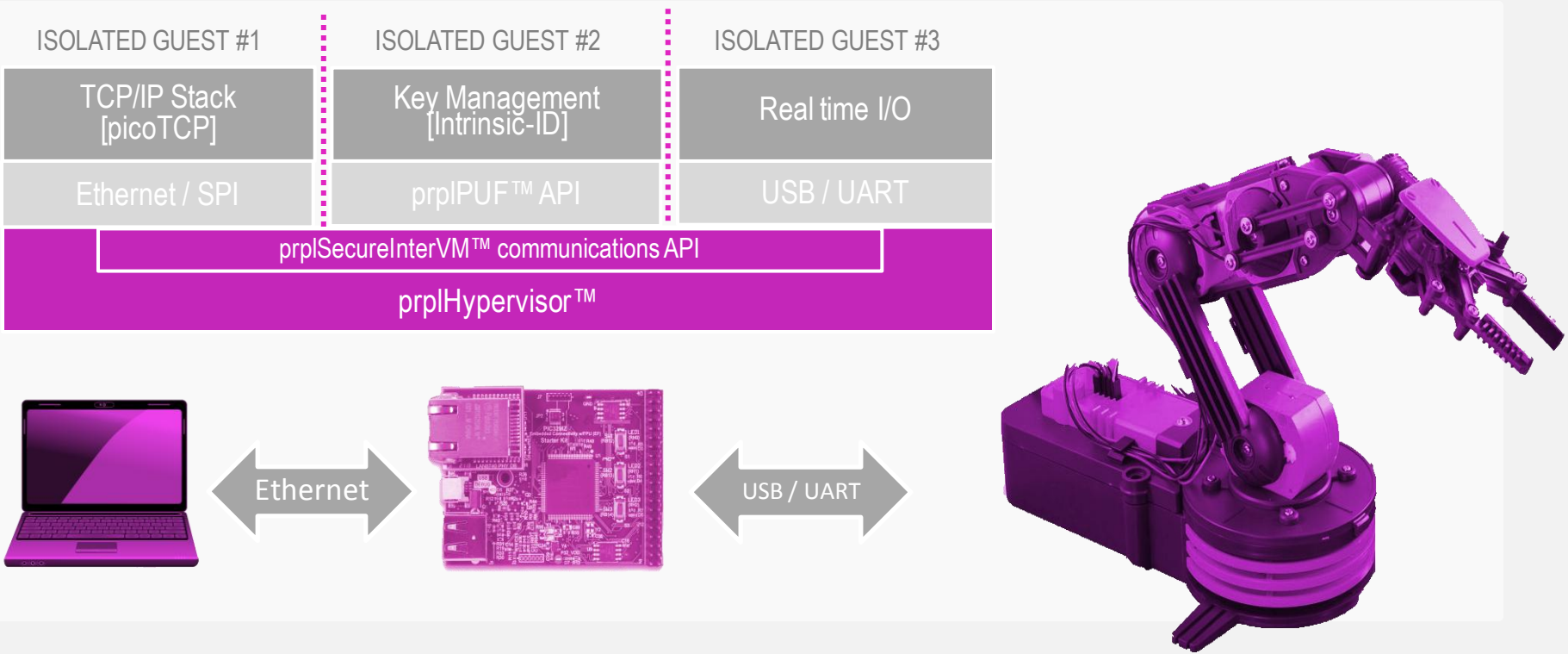
# What is PUF - Physical Unclonable Functions?

## Physical Unclonable Functions [SRAM]

- ✓ 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
- ✓ 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

# prplSecurity™ Live Demo

# prplSecurity™ Demo – Application Concept



# prplSecurity™ Demo– Building the firmware

| 1 | 2 | 3 |
| --- | --- | --- |
| *Clone GitHub repo* | *Make firmware* | *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
```

# prplSecurity™ Demo– Building the firmware

| 1 | 2 | 3 |
|---|---|---|
| *Clone GitHub repo* | *Make firmware* | *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
```

# prplSecurity™ Demo – Building the firmware

1 *Clone GitHub repo*

2 *Make firmware*

3 *Flash firmware*

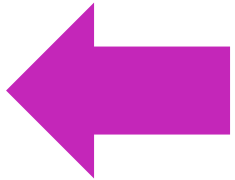
File Edit View Search Terminal Help

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

# prplSecurity™ 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
```



# prplSecurity™ 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

```
telnet 192.168.0.2 80
Trying 192.168.0.2...
Connected to 192.168.0.2.
Escape character is '^]'.
95a84a049651e231f6d358d0e6cb3af2010000000000000000000000000008051f26287be978cf8
399628ce365e9e8fe9a4328a95514c27
```

# prplSecurity™ 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
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
```


cesare@prplFoundation.org

http://prpl.works

# prpl Foundation Reference Publications