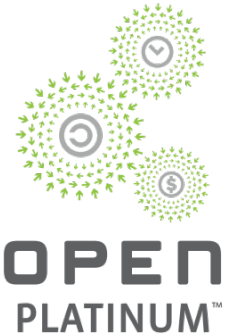

Open. Together.
OCP
SUMMIT

Ownership and Control of Firmware in Open Compute Project (OCP) Devices

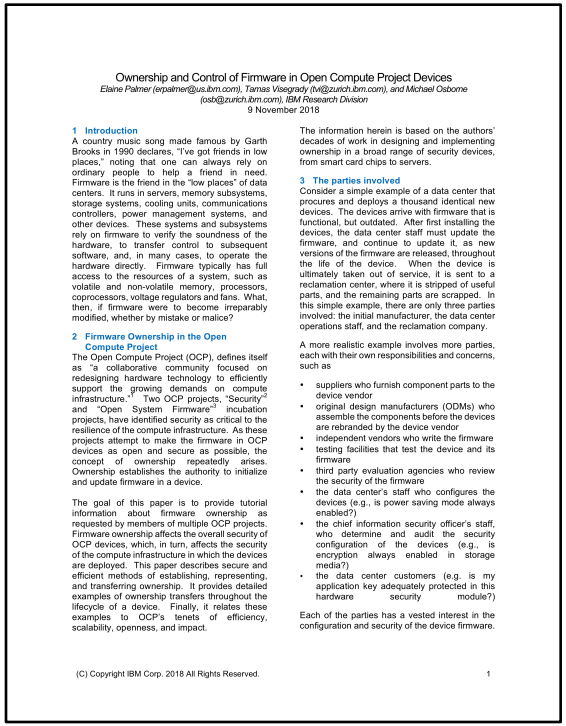
Elaine Palmer, Senior Technical Staff Member, IBM Thomas J. Watson Research

Tamas Visegrady, Research Staff Member, IBM Research - Zurich

Michael Osborne, Research Staff Member, IBM Research - Zurich



Ownership and Control of Firmware in Open Compute Project (OCP) Devices



Ownership and Control of Firmware in Open Compute Project Devices

Elaine Palmer (erpalmer@us.ibm.com), Tamas Visegrady (tvi@zurich.ibm.com), and Michael Osborne (osb@zurich.ibm.com), IBM Research Division

9 November 2018

1 Introduction

A country music song made famous by Garth Brooks in 1990 declares, "I've got friends in low places," noting that one can always rely on ordinary people to help a friend in need. Firmware is the friend in the "low places" of data centers. It runs in servers, memory subsystems, storage systems, cooling units, communications controllers, power management systems, and other devices. These systems and subsystems rely on firmware to verify the soundness of the hardware, to transfer control to subsequent software, and, in many cases, to operate the hardware directly. Firmware typically has full access to the resources of a system, such as volatile and non-volatile memory, processors, coprocessors, voltage regulators and fans. What, then, if firmware were to become irreparably modified, whether by mistake or malice?

2 Firmware Ownership in the Open Compute Project

The Open Compute Project (OCP), defines itself as "a collaborative community focused on redesigning hardware technology to efficiently support the growing demands on compute infrastructure."[1] Two OCP projects, "Security"[2] and "Open System Firmware"[3] incubation projects, have identified security as critical to the resilience of the compute infrastructure. As these projects attempt to make the firmware in OCP devices as open and secure as possible, the concept of ownership repeatedly arises. Ownership establishes the authority to initialize and update firmware in a device.

The goal of this paper is to provide tutorial information about firmware ownership as requested by members of multiple OCP projects. Firmware ownership affects the overall security of OCP devices, which, in turn, affects the security of the compute infrastructure in which the devices are deployed. This paper describes secure and efficient methods of establishing, representing, and transferring ownership. It provides detailed examples of ownership transfers throughout the lifecycle of a device. Finally, it relates these examples to OCP's tenets of efficiency, scalability, openness, and impact.

The information herein is based on the authors' decades of work in designing and implementing ownership in a broad range of security devices, from smart card chips to servers.

3 The parties involved

Consider a simple example of a data center that procures and deploys a thousand identical new devices. The devices arrive with firmware that is functional, but outdated. After first installing the devices, the data center staff must update the firmware, and continue to update it, as new versions of the firmware are released, throughout the life of the device. When the device is ultimately taken out of service, it is sent to a reclamation center, where it is stripped of useful parts, and the remaining parts are scrapped. In this simple example, there are only three parties involved: the initial manufacturer, the data center operations staff, and the reclamation company.

A more realistic example involves more parties, each with their own responsibilities and concerns, such as

- suppliers who furnish component parts to the device vendor
- original design manufacturers (ODMs) who assemble the components before the devices are rebranded by the device vendor
- independent vendors who write the firmware
- testing facilities that test the device and its firmware
- third party evaluation agencies who review the security of the firmware
- the data center's staff who configures the devices (e.g., is power saving mode always enabled?)
- the chief information security officer's staff, who determine and audit the security configuration of the devices (e.g., is encryption always enabled in storage media?)
- the data center customers (e.g. is my application key adequately protected in this hardware security module?)

Each of the parties has a vested interest in the configuration and security of the device firmware.

White Papers

SECURITY

OPEN SYSTEMS FIRMWARE

MANAGEMENT

https://www.opencompute.org/documents/ibm-white-paper-ownership-and-control-of-firmware-in-open-compute-project-devices

Contents

“It’s déjà vu all over again.”

Yogi Berra

Smart cards, hardware security modules, server motherboards, Trusted Platform Modules, adapters, . . .

We thought everyone knew this stuff (but realized they didn't)

It’s easy to do it right.

It affects multiple OCP groups

Security

Open System Firmware

Hardware Management (Open BMC)

It is an IBM Contribution to OCP.

What problem are we trying to solve?

"If you don't own your firmware, your firmware owns you."

Ron Minnich, Google Software Engineer,

Linux Security Summit, Vancouver, 2018

How do we prevent an attacker from establishing ownership of a device?

How do we initialize a device with information about its owner?

How do we transfer ownership securely?

How do we transfer ownership when we don't know who the next owner is?

Who owns field replacement parts?

An owner controls what firmware is allowed to run on a device.

Owner

Authority

Officer

Administrator

An owner is not necessarily. . .

the one who purchased the device

in physical control of the device

the one who holds intellectual property rights over it

One device may have multiple owners (all at once or across its lifetime)

Power On Self Test

Secure boot

Authenticated update

Hypervisor

Operating System

Application

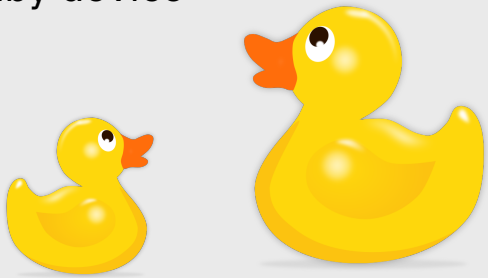
Manufacturing tests

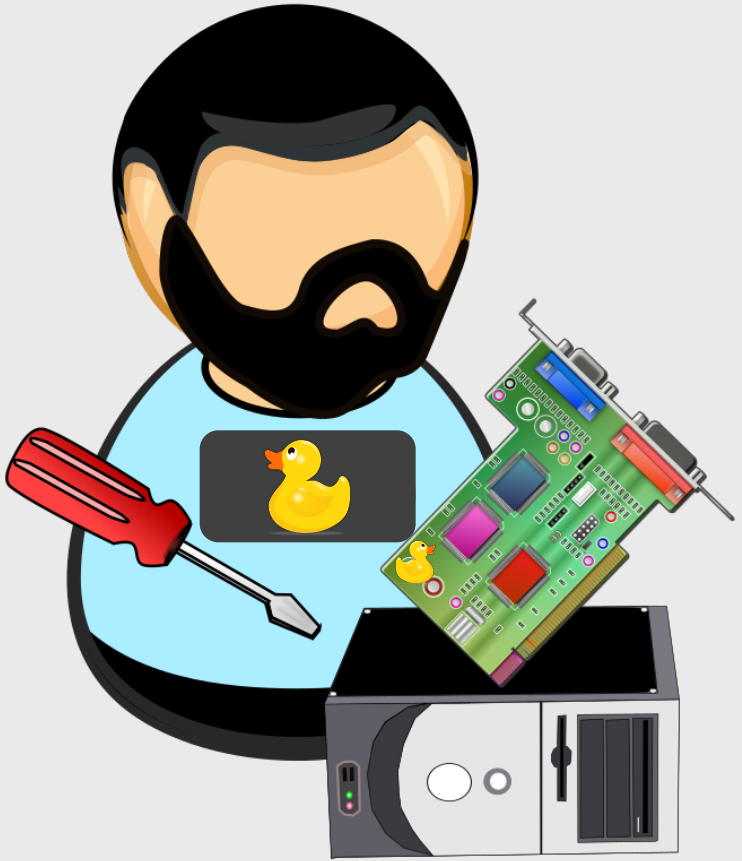
Imprinting in the field

You must be my ~~mother~~ owner

First come first served

You must be my ~~baby~~ device





Some attacks on imprinting

Good credentials / bad device or bad credentials / good device

Software clone

Tricks the owner into thinking it's a real device.

Owner issues legitimate credentials to malicious software.

Duck call fools mama duck into thinking it's real.

Hardware clone

Tricks the owner into thinking it's the right kind of device.

Owner issues legitimate credentials to malicious hardware.

Decoy fools mama duck into thinking it's real.

Get there first

Attacker initializes with its own keys, then tricks the owner into thinking it's a pristine device.

Owner issues legitimate credentials to authentic, but compromised hardware.

Duckling cyborg fools mama duck into thinking it's an ordinary duckling.

False credentials

Attacker tricks device to connect to false credentialing authority.

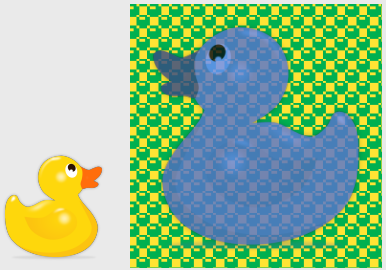
Attacker issues false credentials to legitimate device.

Buzzard fools duckling into thinking it's its mother.

"Man-in-the-middle" (MITM) attacks are easy during imprinting, so use MITM-resistant protocols!

Temporary Transport Keys (aka Service Keys)

Use this as your temporary ~~mother~~ owner



Works well when

- we know how many to produce with that key
- both parties can be trusted to guard shared secrets

Problems with temporary transport keys

How many to produce?

The manufacturer must initialize the right number of devices with the transport key.

What if the next owner is unknown? How many should have that key?

Can we re-initialize unsold devices with different keys for a different buyer?

Is the transport key really unique to these parties?

Did the seller put the same key in others' devices too? What if other buyers use that key to attack my devices?

Can I trust the other party not to leak a shared secret?

Did the seller leave our shared secret on a disk in an open manufacturing system?

How to initialize field stock?

What keys do we put in spare parts sitting in the warehouse? They don't belong to anyone yet.

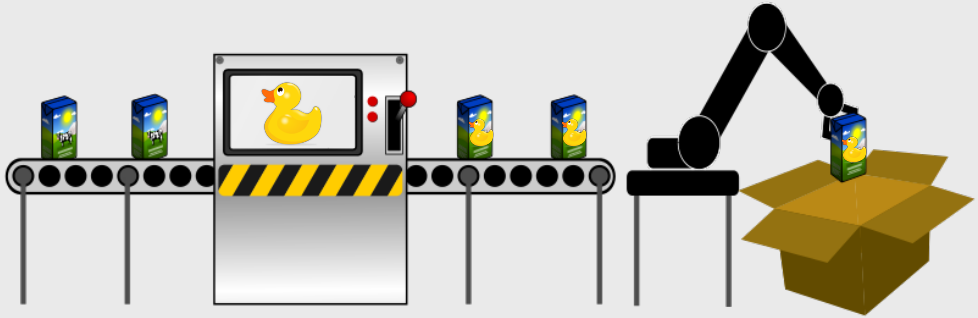
Permanent Keys at Manufacture

The device knows its identity and gets its credentials before it ever leaves the factory.

The IBM 4767-002

Works well when

- the device knows at least one owner before it leaves the factory
- the owner's public key can be protected by hardware



Problems with permanent keys at manufacturing

Do I know at least one owner?

A slight variation of “How many to produce?”

How does the device protect the owner’s key that it holds?

Does the device have tamper-protected / tamper-responding storage?

What if the buyer wants to sell it to someone else?

Does the owner have to authorize it?

Will the device allow itself to be owned by someone else?

Of course, there are hybrid schemes.

Minimal requirements for maintaining and using ownership

Remember and write protect the owner's public key (or a list of keys)

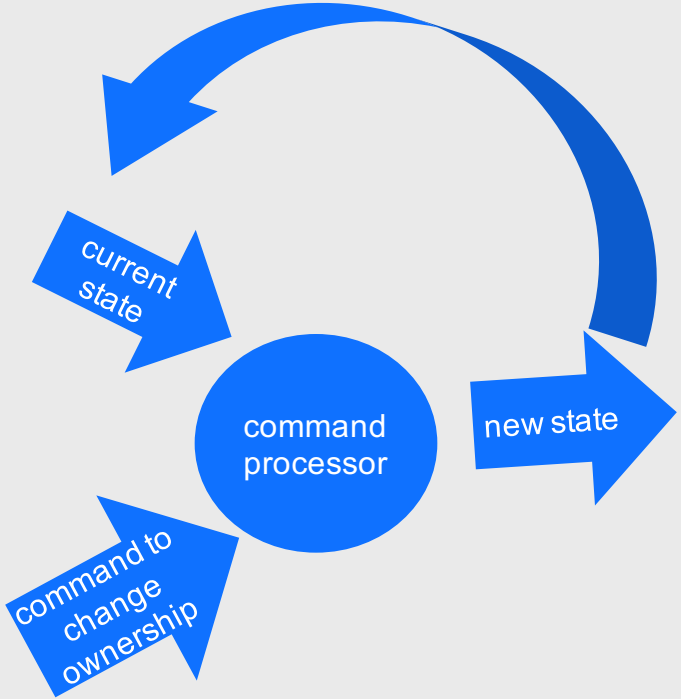
Verify the digital signature of firmware that was signed (elsewhere) using the owner's private key

Representing ownership in (a small amount of) persistent memory

Current Owner	Previous Owner(s)	Designated successor	Reversability
(one X.509 certificate)	(one X.509 certificate)	(one X.509 certificate)	One bit

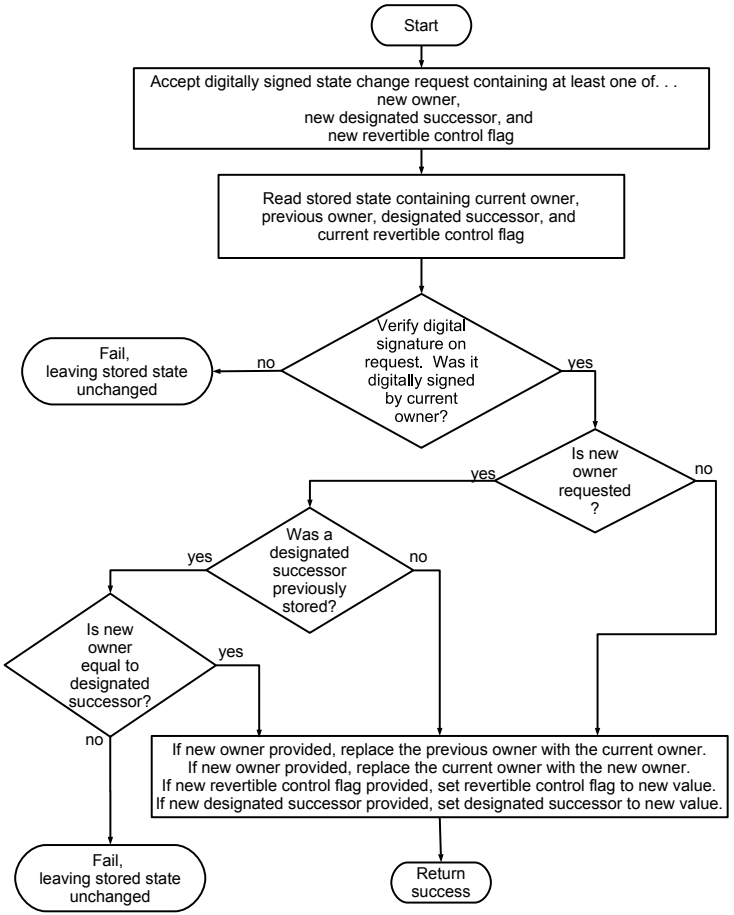
State machine

state:
- current owner
- previous owner
- designated successor
- reversibility



command (digitally signed by current owner) with
- new owner, new designated successor, new reversibility

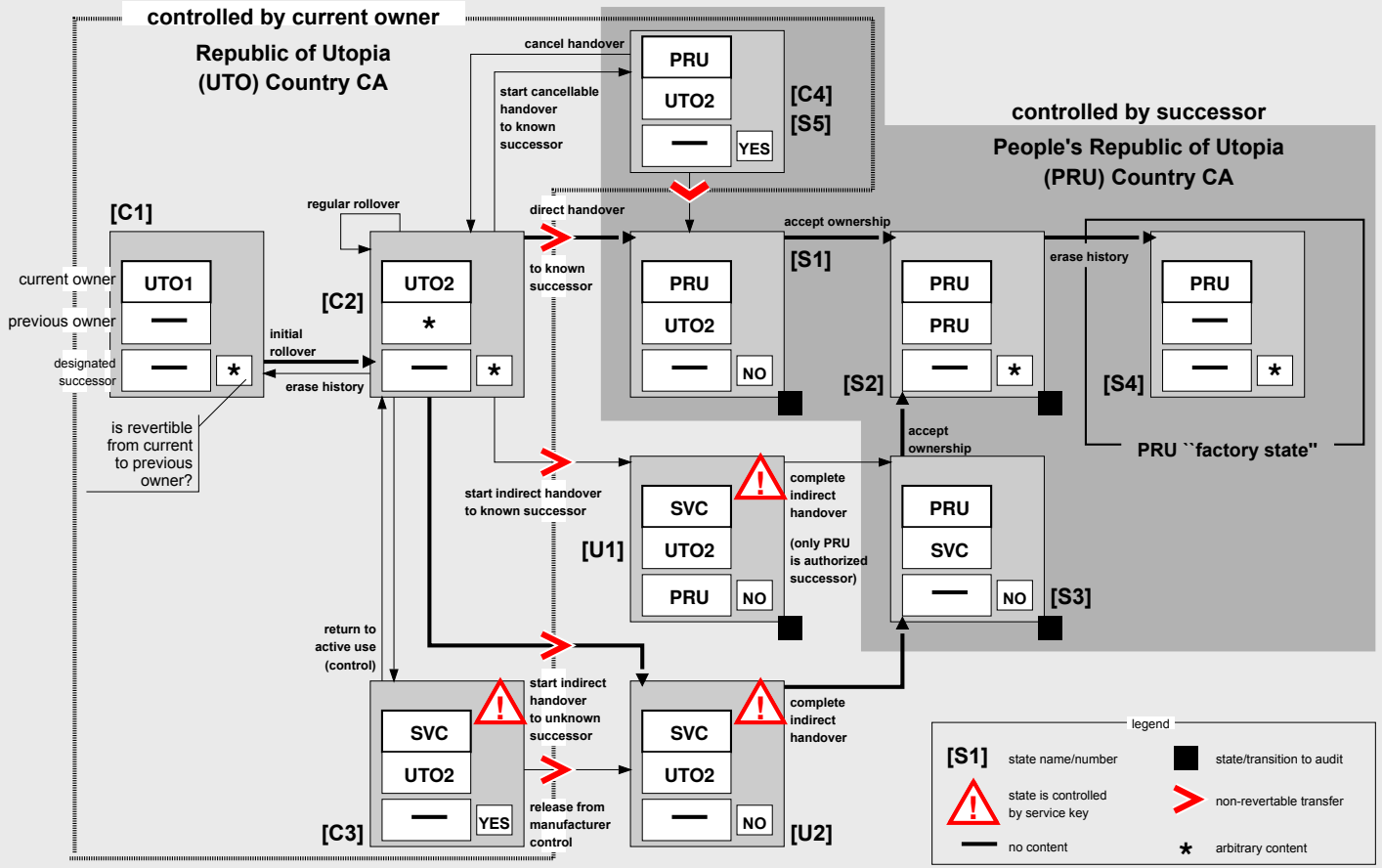
Command processing for an ordinary transfer



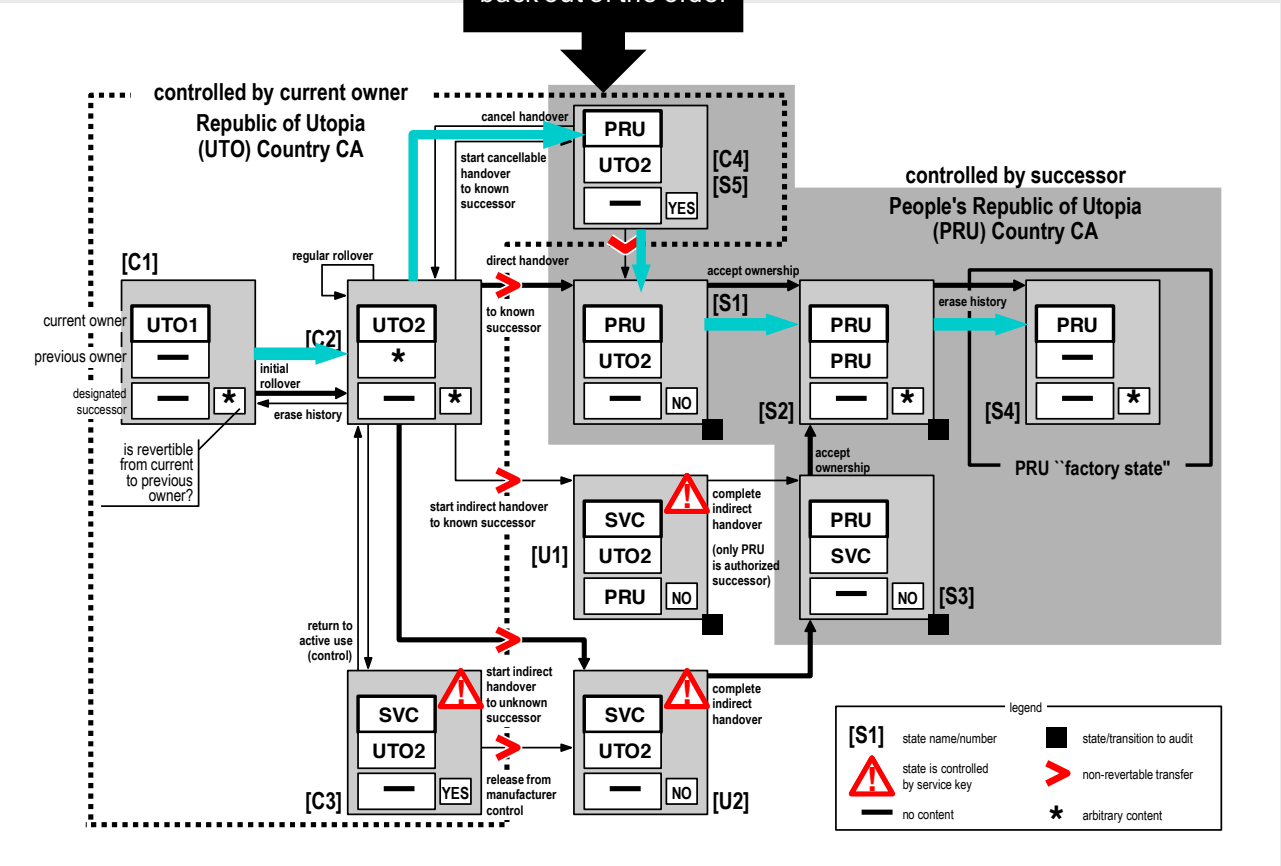
Some of the handovers supported

- Direct, cancellable
- Direct, not cancellable
- Return to inventory
- Erase history
- Cancel handover
- Indirect, unknown successor
- Indirect, known successor

Ownership transfer



Direct, cancellable handover



OCP Tenets

Efficiency

storage < 6K bytes

signature verification code (likely there anyway for firmware)

command processing and state machine

Scalability

small enough for adapters

component of many devices

remote administration

longer histories

Openness

white paper makes the technology and techniques known to others

Impact

improves the security of the supply chain

devices can manage their own update processes

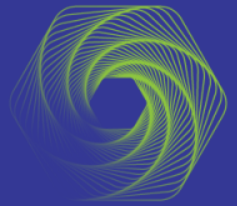
OCP Security Project – how to get involved

Check it out (overview)
https://www.opencompute.org/projects/security

Dig deeper (project wiki)
https://www.opencompute.org/wiki/Security

Follow us (mailing list)
https://ocp-all.groups.io/g/OCP-Security

Participate
weekly calls Tuesdays @ 8:30 a.m. PT



Open. Together.

OCP Global Summit | March 14–15, 2019