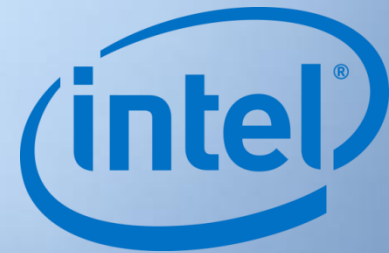




# Open Source Test Tools for UEFI

Ricardo Neri, Software Engineer  
Vincent Zimmer, Sr. Principal Engineer

*presented by*



# Agenda



Linux UEFI Validation Operating System

CHIPSEC - Platform Security Assessment Framework

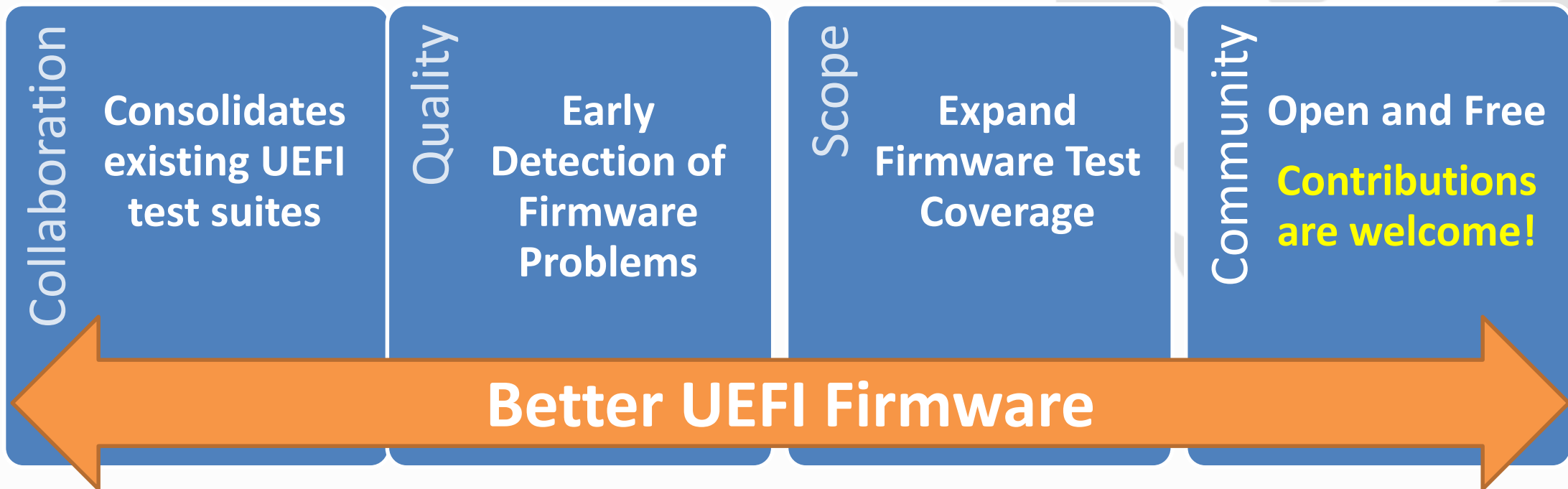
What Now?



# Linux UEFI Validation Operating System



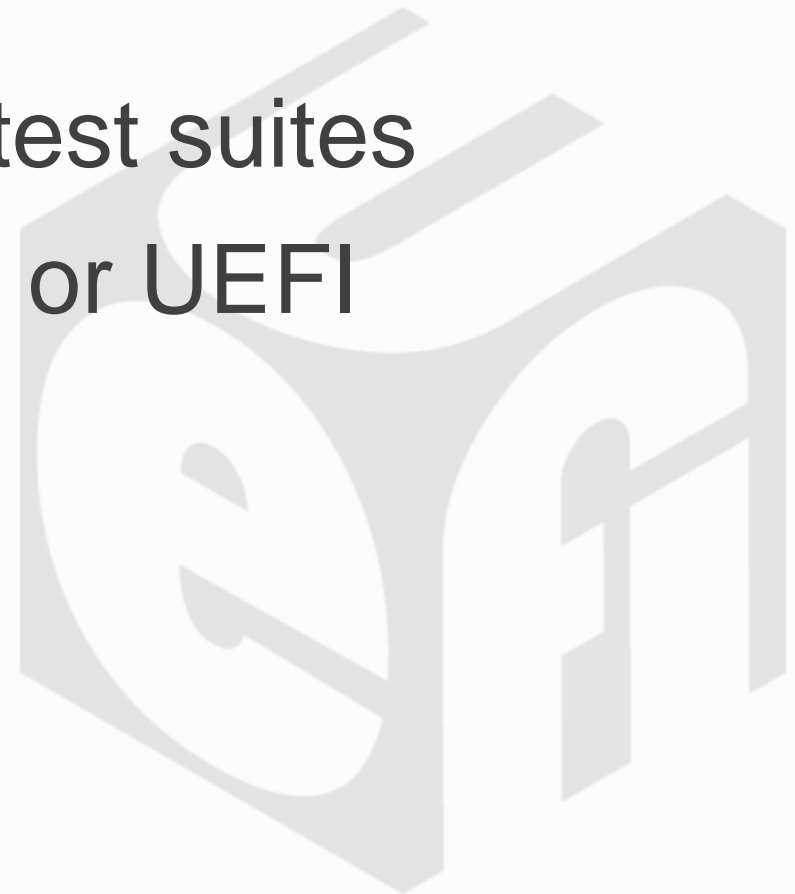
Objective: Create a Linux-readiness validation operating system which helps firmware engineers and Linux contributors reduce development and enabling time for Linux on an UEFI system.



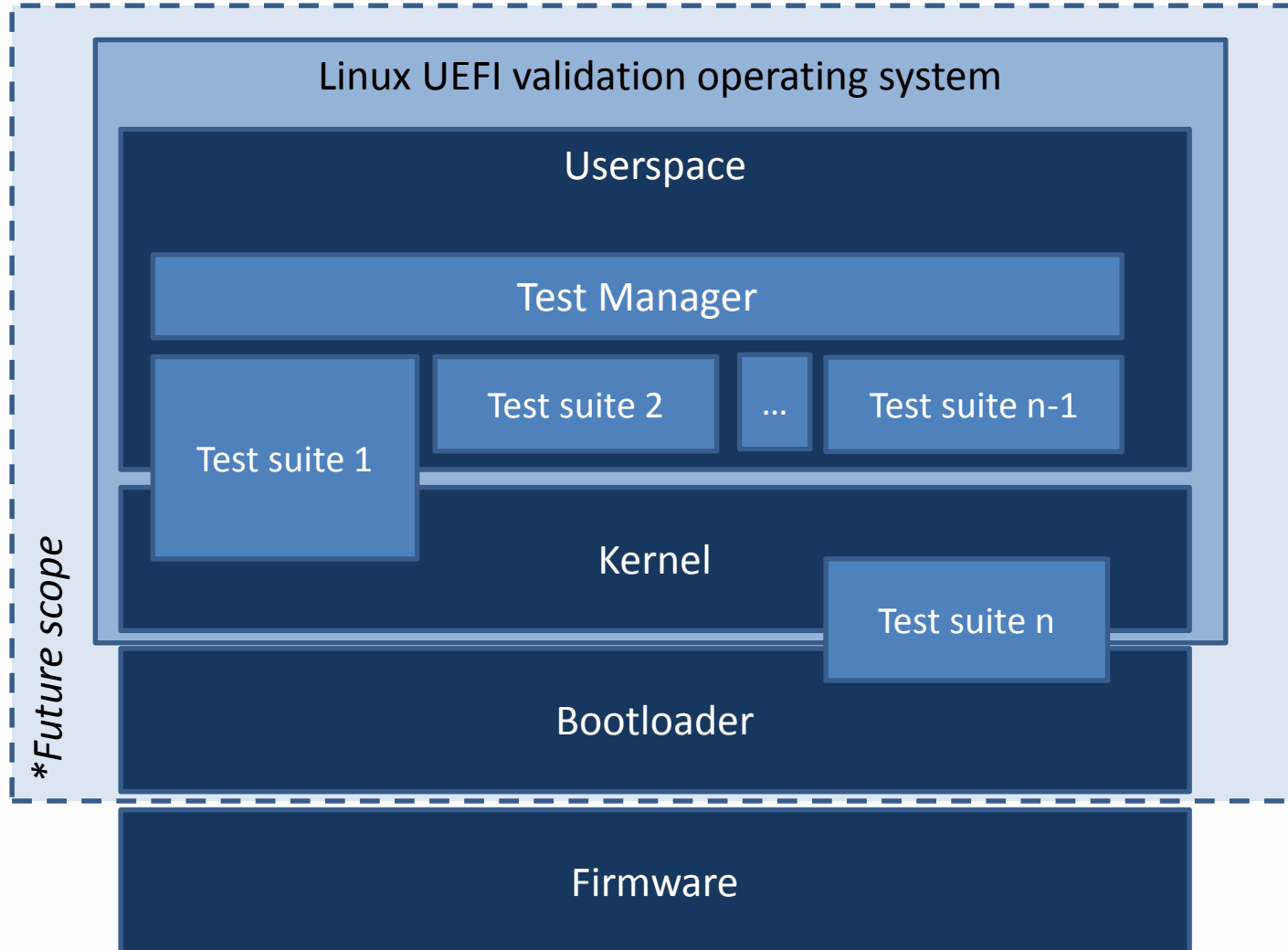
# What is it not?



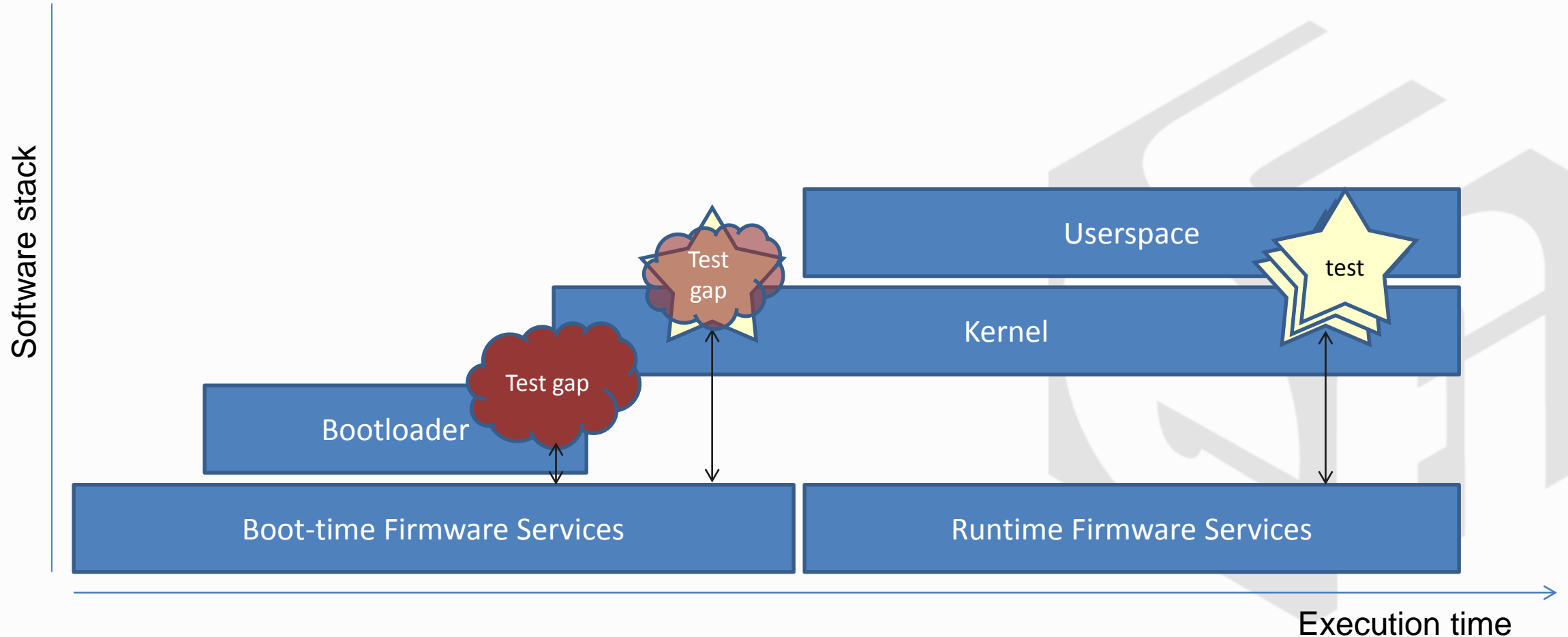
- ❌ A new test suite
- ❌ A replacement for existing test suites
- ❌ A certification tool for Linux or UEFI compliance
- ❌ A proprietary Intel tool



# The Details



# Covers Entire Execution Cycle



# Future Development



Tests for bootloaders

Automation

UEFI Capsule tests

UEFI Network stack tests

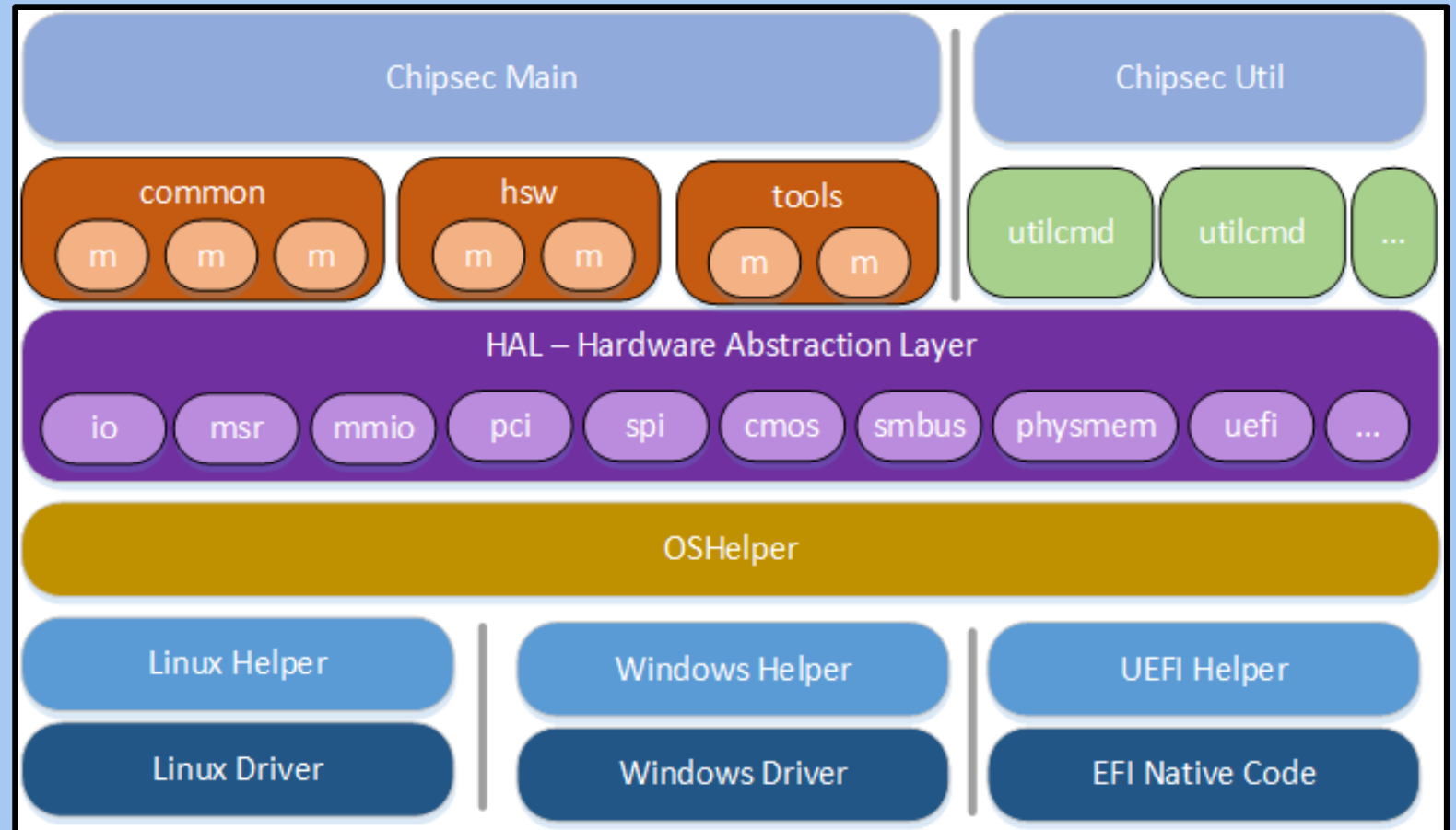
UEFI Secure Boot tests



# CHIPSEC - Platform Security Assessment Framework

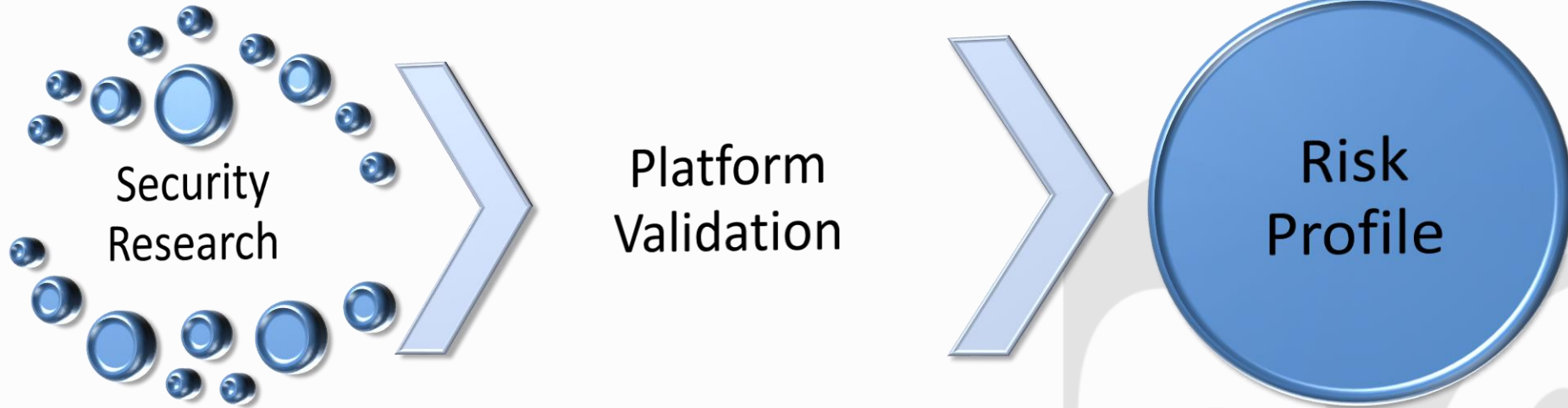


A single test designed to run in multiple environments





# How do we raise the bar?



New Attacks

Test Modules  
for OEMs/IBVs

End-user Risk

## Empowering End-Users to Make a Risk Decision

# Known Threats and CHIPSEC modules



Issue	CHIPSEC Module	Public Details
SMRAM Locking	common.smm	<a href="#">CanSecWest 2006</a>
BIOS Keyboard Buffer Sanitization	common.bios_kbrd_buffer	<a href="#">DEFCON 16 2008</a>
SMRR Configuration	common.smrr	<a href="#">ITL 2009</a> <a href="#">CanSecWest 2009</a>
BIOS Protection	common.bios_wp	<a href="#">BlackHat USA 2009</a> <a href="#">CanSecWest 2013</a> <a href="#">Black Hat 2013</a> <a href="#">NoSuchCon 2013</a> <a href="#">Flashrom</a>
SPI Controller Locking	common.spi_lock	<a href="#">Flashrom</a> <a href="#">Copernicus</a>
BIOS Interface Locking	common.bios_ts	<a href="#">PoC 2007</a>
Access Control for Secure Boot Keys	common.secureboot.keys	<a href="#">UEFI 2.4 Spec</a>
Access Control for Secure Boot Variables	common.secureboot.variables	<a href="#">UEFI 2.4 Spec</a>

# Example: BIOS Write Protection



Black Hat USA 2013 "[BIOS Security](#)" – MITRE (Kovah, Butterworth, Kallenberg)

NoSuchCon 2013 "[BIOS Chronomancy: Fixing the Static Root of Trust for Measurement](#)" – MITRE (Kovah, Butterworth, Kallenberg)

Is BIOS correctly protected?

▶ common.bios\_wp

```
[+] imported chipsec.modules.common.bios_wp
```

```
[x] [ =====  
[x] [ Module: BIOS Region Write Protection  
[x] [ =====
```

```
BIOS Control (BDF 0:31:0 + 0xDC) = 0x2A
```

```
[05] SMM_BWP = 1 (SMM BIOS Write Protection)  
[04] TSS_ = 0 (Top Swap Status)  
[01] BLE = 1 (BIOS Lock Enable)  
[00] BIOSWE = 0 (BIOS Write Enable)
```

```
[+] BIOS region write protection is enabled (writes restricted to SMM)
```

```
[*] BIOS Region: Base = 0x00500000, Limit = 0x00FFFFFF
```

```
SPI Protected Ranges
```

PRx (offset)	Value	Base	Limit	WP?	RP?
PR0 (74)	00000000	00000000	00000000	0	0
PR1 (78)	8FFF0F40	00F40000	00FFF000	1	0
PR2 (7C)	8EDF0EB1	00EB1000	00EDF000	1	0
PR3 (80)	8EB00EB0	00EB0000	00EB0000	1	0
PR4 (84)	8EAF0C00	00C00000	00EAF000	1	0

```
[!] SPI protected ranges write-protect parts of BIOS region (other parts of BIOS can be modified)
```

```
[+] PASSED: BIOS is write protected
```

# Direct HW Access for Manual Testing



## Examples:

```
chipsec_util msr 0x200
chipsec_util mem 0x0 0x41E 0x20
chipsec_util pci enumerate
chipsec_util pci 0x0 0x1F 0x0 0xDC byte
chipsec_util io 0x61 byte
chipsec_util mmcfg 0 0x1F 0 0xDC 1 0x1
chipsec_util cmos dump
chipsec_util ucode id
chipsec_util smi 0x01 0xFF
chipsec_util idt 0
chipsec_util cpuid 1
chipsec_util spi read 0x700000 0x100000 bios.bin
chipsec_util decode spi.bin
chipsec_util uefi var-list
..
```

# Forensics



## Live system firmware analysis

```
chipsec_util spi info
chipsec_util spi dump rom.bin
chipsec_util spi read 0x700000 0x100000 bios.bin
chipsec_util uefi var-list
chipsec_util uefi var-read db D719B2CB-3D3A-4596-
A3BC-DAD00E67656F db.bin
```

## Offline system firmware analysis

```
chipsec_util uefi keys PK.bin
chipsec_util uefi nvram vss bios.bin
chipsec_util uefi decode rom.bin
chipsec_util decode rom.bin
```

# Moving Forward



Test tools complement the SCT, but **the community can do more!**

Changing our development philosophy?

- [“Testing shows the presence, not the absence of bugs”](#) (*Dijkstra, 1970*)
- [Better Living Through Tools?](#) (*Zimmer, 2013*)

Getting code coverage closer to 100%?

- Internal Intel effort using [DDT](#) with EDK II
- Moving to [KLEE](#) (open source)

“[Infrastructure for automatic code checking](#)” (coreboot)

- Automated system including KLEE, Splint, Frama-C

# What Now?



## *Linux UEFI Validation Operating System*

Download the code...

<https://github.com/01org/luv-yocto>

Download test images ...

<https://01.org/linux-uefi-validation>

Submit your contributions

Questions?

- [luv@lists.01.org](mailto:luv@lists.01.org)
- [ricardo.neri@intel.com](mailto:ricardo.neri@intel.com)
- [matt.fleming@intel.com](mailto:matt.fleming@intel.com)

## **CHIPSEC**

Get the tool

<https://github.com/chipsec/chipsec>

Submit your contributions

Questions?

- [chipsec@intel.com](mailto:chipsec@intel.com)
- [vincent.zimmer@intel.com](mailto:vincent.zimmer@intel.com)

For more information on the  
Unified EFI Forum and UEFI  
Specifications, visit  
<http://www.uefi.org>



*presented by*

