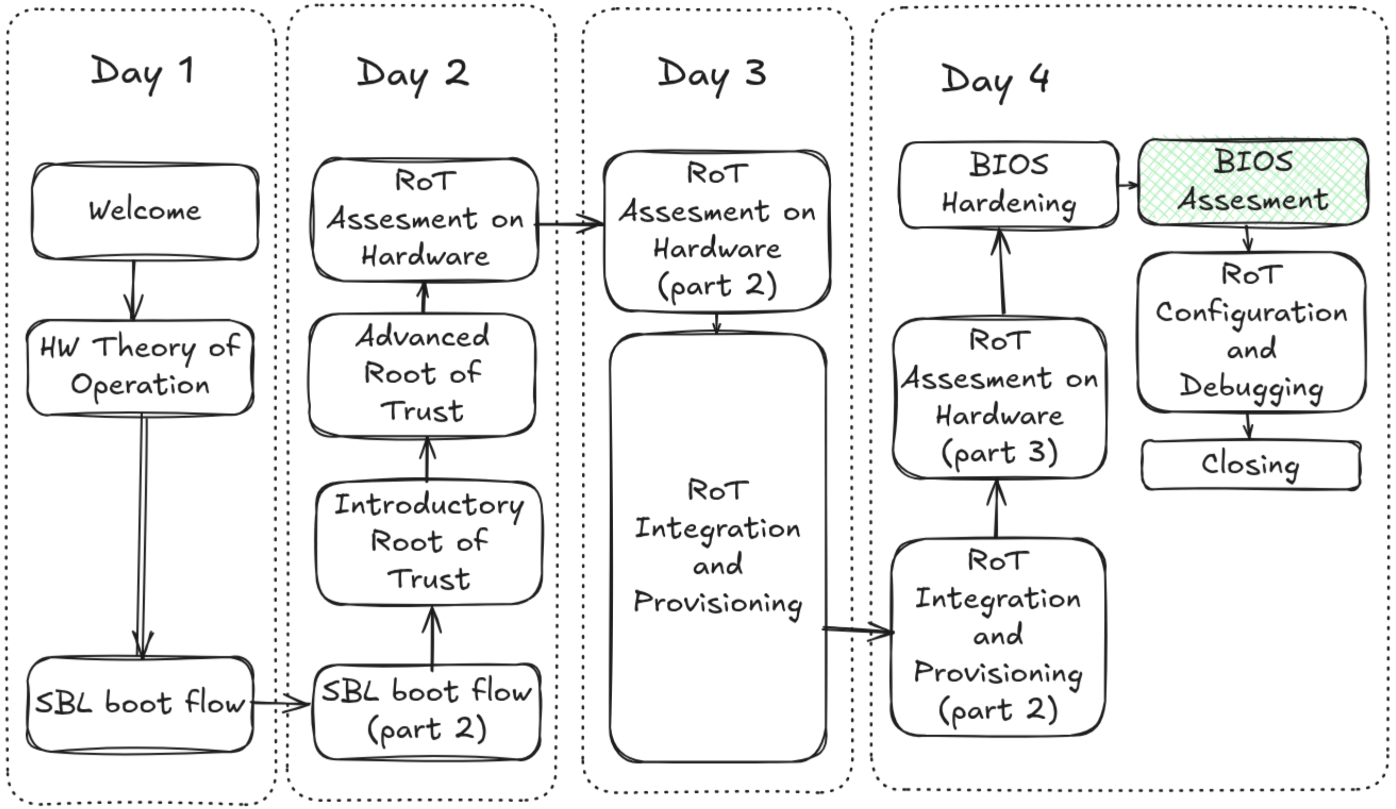
DS09SBL: Dasharo TrustRoot Training

BIOS Assessment

Dasharo TrustRoot Training

Where we are in the course


Day 1
Welcome
HW Theory of Operation
SBL boot flow
Day 2
SBL boot flow (part 2)
Introductory Root of Trust
Advanced Root of Trust
RoT Assesment on Hardware
Day 3
RoT Assesment on Hardware (part 2)
RoT Integration and Provisioning
Day 4
RoT Integration and Provisioning (part 2)
RoT Assesment on Hardware (part 3)
BIOS Hardening
BIOS Assesment
RoT Configuration and Debugging
Closing

Goals of the Presentation

In this presentation, we aim to:

- Introduce to CHIPSEC
- Explain how CHIPSEC can be used to assess Slim Bootloader hardening
- Introduce to fwupd and HSI (Host Security Index)
- Explain how fwupd HSI can be used to assess Slim Bootloader hardening

Introductory CHIPSEC

Goals of the Presentation

In this presentation, we aim to:

- Learn what is CHIPSEC and how to use it.
- See basic usage of CHIPSEC.
- Understand CHIPSEC reports.
- Understand CHIPSEC internals and how add custom tests.

https://github.com/chipsec

Recommended Materials

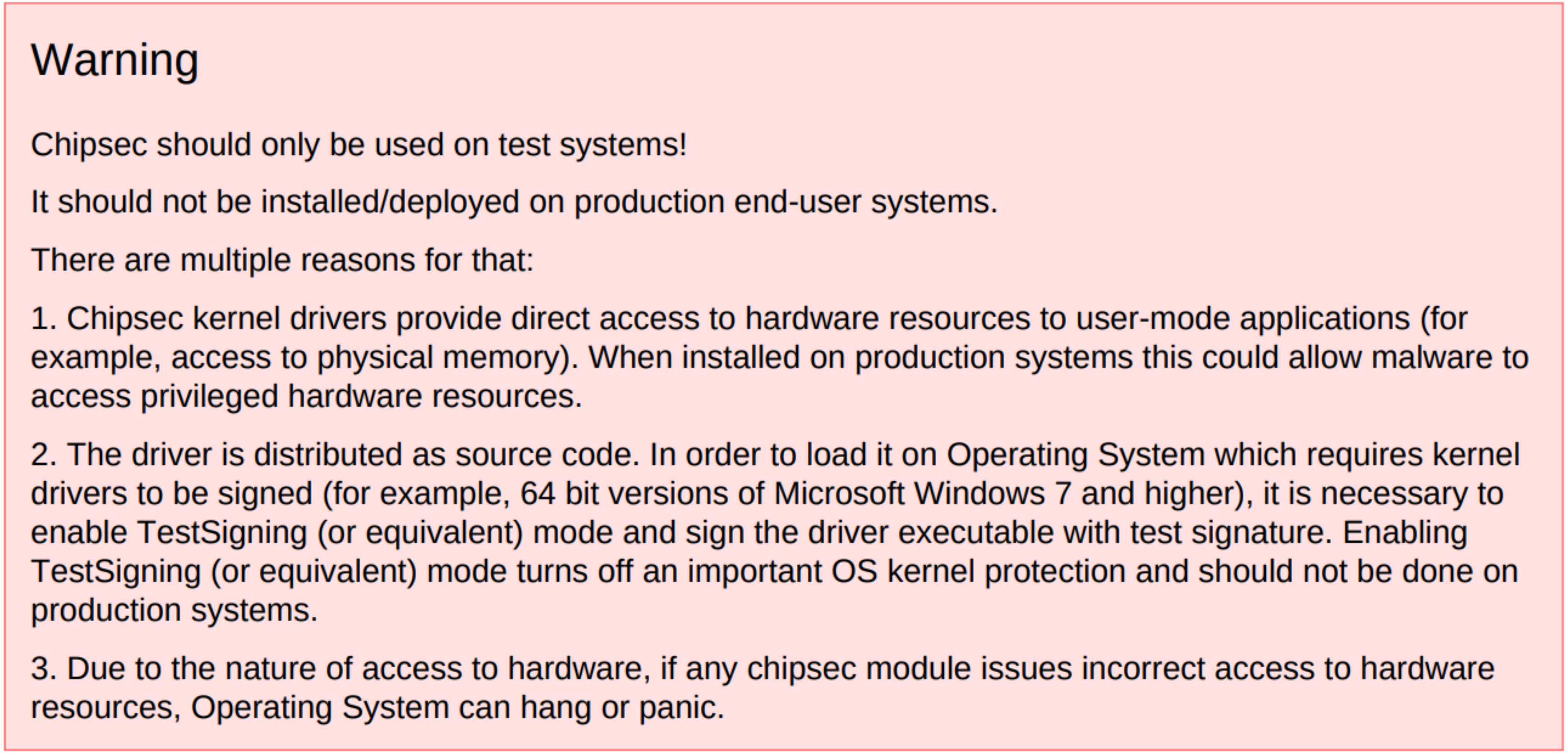
- CHIPSEC Documentation

CHIPSEC overview

- Framework for platform security assessment
- Originally developed by Intel as an internal tool
  - v1.1.0 was publicly released on GitHub on May 30, 2014
  - Active development
  - Most recent version v1.13.15 was released Aug 01, 2025
- Written in Python (89%) and C (8.8%)
- GPLv2 license
- Framework is designed to support public and private ecosystem.
- Significant community around
  - quarterly community meetings: https://github.com/chipsec/chipsec/wiki/Community-Meetings
  - mailing list:

CHIPSEC overview

- Can analyze the security of the whole platform
    - hardware configuration
    - system firmware (BIOS/UEFI/SBL)
    - other platform components
- Supports multiple environments
    - Linux
    - Windows
    - MacOS (beta support)
    - UEFI shell
- Modular and easy to extend
- Two modes of operation
    - standalone command line tool
    - programming library



Warning

Chipsec should only be used on test systems!

It should not be installed/deployed on production end-user systems.

There are multiple reasons for that:

1. Chipsec kernel drivers provide direct access to hardware resources to user-mode applications (for example, access to physical memory). When installed on production systems this could allow malware to access privileged hardware resources.

2. The driver is distributed as source code. In order to load it on Operating System which requires kernel drivers to be signed (for example, 64 bit versions of Microsoft Windows 7 and higher), it is necessary to enable TestSigning (or equivalent) mode and sign the driver executable with test signature. Enabling TestSigning (or equivalent) mode turns off an important OS kernel protection and should not be done on production systems.

3. Due to the nature of access to hardware, if any chipsec module issues incorrect access to hardware resources, Operating System can hang or panic.

https://github.com/chipsec/chipsec/blob/main/chipsec-manual.pdf

Installation - common

- CHIPSEC requires a custom kernel driver
- On Linux/MacOS custom-compiled driver is needed
- On Windows it used to use RWeverything
  - support was dropped due to PCI configuration space access issues
  - it needs a custom-compiled driver now as well
- Below steps are common for all OS
- Install `git`
- Install `Python 3.7` or higher
  - Python2 support in CHIPSEC is deprecated since June 2020
- Clone code repository

```
$ git clone https://github.com/chipsec/chipsec.git
```

Installation on Windows

- Install Python dependencies

```
$ pip install -r windows_requirements.txt
```

- Install Visual Studio and WDK (Windows Driver Kit)
  - at least WDK 8 and VS 2012
  - https://docs.microsoft.com/en-us/windows-hardware/drivers/other-wdk-downloads
- Turn off kernel driver signature checks
  - https://chipsec.github.io/installation/Install in Windows.html#turn-off-kernel-driver-signature-checks
- Compile

```
$ python setup.py build_ext -i
```

Installation on Linux

- Install build dependencies

```
$ apt-get install build-essential python3-dev python3 gcc linux-headers-$(uname -r) nasm
```

- Install `setuptools` package

```
$ pip install setuptools
```

- Compile

```
$ python3 setup.py build_ext -i
```

Other ways of running CHIPSEC

- Regular installation was presented on the previous slides
- Run CHIPSEC directly from UEFI shell
    - needs preparation of bootable USB drive
    - place Python for UEFI and CHIPSEC on this drive
    - https://chipsec.github.io/installation/USBwithUEFIShell.html
- Create bootable Linux Live USB with CHIPSEC
    - https://chipsec.github.io/installation/InstallLinux.html
- Linux UEFI Validation Distribution (LUV) - no longer maintained
    - bootable Linux image with firmware validation tools (including CHIPSEC)
    - no longer maintained - should not be used anymore

Practice #312 Exercise #1

- Install CHIPSEC on Debian 13

```
git clone https://github.com/chipsec/chipsec.git
cd chipsec/
sudo apt install virtualenv
virtualenv venv
source venv/bin/activate
pip install -r linux_requirements.txt
sudo apt-get install build-essential python3-dev python3 gcc linux-headers-$(uname -r) nasm
python setup.py build_ext -i
sudo insmod chipsec/helper/linux/chipsec.ko
sudo python3 chipsec_main.py -m common.bios_wp
```

Components and structure

- `chipsec_main.py` - main application logic and automation functions
- `chipsec_util.py` - utility functions (access to various hardware resources)
- `chipsec/chipset.py` - chipset detection
- `chipsec/command.py` - base class for util commands
- `chipsec/defines.py` - common defines
- `chipsec/file.py` - reading from/writing to files
- `chipsec/logger.py` - logging functions
- `chipsec/module.py` - generic functions to import and load modules
- `chipsec/module_common.py` - base class for modules
- `chipsec/result_deltas.py` - supports checking result deltas between test runs
- `chipsec/testcase.py` - support for XML and JSON log file output
- `chipsec/helper/helpers.py` - registry of supported OS helpers
- `chipsec/helper/oshelper.py` - OS helper: wrapper around platform specific code that invokes kernel driver

```
(venv) debian@debian:~/chipsec$ sudo python3 chipsec_main.py -m common.bios_wp
[sudo] password for debian:

################################################################
##                                                            ##
##  CHIPSEC: Platform Hardware Security Assessment Framework  ##
##                                                            ##
################################################################
[CHIPSEC] Version  : 1.13.15
[CHIPSEC] Arguments: -m common.bios_wp

ERROR: Unknown Platform: VID = 0x8086, DID = 0x4678, RID = 0x00, CPUID = 0xB06E0
ERROR: Platform is not supported (Unknown Platform: VID = 0x8086, DID = 0x4678, RID = 0x00, CPUID = 0xB06E0).
WARNING: Platform dependent functionality is likely to be incorrect
( ... )
```

Practice #312 Exercise #2

- Run CHIPSEC against AMI BIOS on ODROID-H4

```
################################################################
##                                                            ##
##  CHIPSEC: Platform Hardware Security Assessment Framework  ##
##                                                            ##
################################################################
[CHIPSEC] Version  : 1.13.15
[CHIPSEC] Arguments: -p ADL -m common.bios_wp

WARNING: Enumerated Platform PCI DID not found in XML Configs. System info may not be 100% accurate.

[CHIPSEC] OS      : Linux 6.12.38+deb13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.12.38-1 (2025-07-16) x86_64
[CHIPSEC] Python  : 3.13.5 (64-bit) - Enabled GIL
[CHIPSEC] Helper  : LinuxHelper (/home/debian/chipsec/chipsec/helper/linux/chipsec.ko)
[CHIPSEC] Platform: ADL-P/U15 2+8
[CHIPSEC]    CPUID: B06E0
[CHIPSEC]      VID: 8086
[CHIPSEC]      DID: 4678
[CHIPSEC]      RID: FF
[CHIPSEC] PCH     : Intel ADL-M (600) PCH
[CHIPSEC]      VID: 8086
[CHIPSEC]      DID: 5481
[CHIPSEC]      RID: 00
( ... )
```

```
[+] loaded chipsec.modules.common.bios_wp
[*] running loaded modules ..

[*] Running module: chipsec.modules.common.bios_wp
[*] Metadata tags: ['OPENSOURCE', 'IA', 'COMMON', 'BIOS_WP']
[x][ =======================================================================
[x][ Module: BIOS Region Write Protection
[x][ =======================================================================
[*] BC = 0x10000888 << BIOS Control (b:d.f 00:31.5 + 0xDC)
    [00] BIOSWE           = 0 << BIOS Write Enable
    [01] BLE              = 0 << BIOS Lock Enable
    [02] SRC              = 2 << SPI Read Configuration
    [04] TSS              = 0 << Top Swap Status
    [05] SMM_BWP          = 0 << SMM BIOS Write Protection
    [06] BBS              = 0 << Boot BIOS Strap
    [07] BILD             = 1 << BIOS Interface Lock Down
    [11] ASE_BWP          = 1 << Async SMI Enable for BIOS Write Protection
[-] BIOS region write protection is disabled!
```

```
[*] BIOS Region: Base = 0×00600000, Limit = 0×00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (84)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (88)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR2 (8C)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR3 (90)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR4 (94)     | 00000000 | 00000000 | 00000000 | 0   | 0
```

```
[!] None of the SPI protected ranges write-protect BIOS region

[!] BIOS should enable all available SMM based write protection mechanisms.
[!] Or configure SPI protected ranges to protect the entire BIOS region.
[-] FAILED: BIOS is NOT protected completely

[CHIPSEC] ***************************  SUMMARY  ***************************
[CHIPSEC] Time elapsed            0.003
[CHIPSEC] Modules failed to run   0:
[CHIPSEC] Modules passed          0:
[CHIPSEC] Modules information     0:
[CHIPSEC] Modules failed          1:
[-] FAILED: chipsec.modules.common.bios_wp
[CHIPSEC] Modules with warnings   0:
[CHIPSEC] Modules not applicable  0:
[CHIPSEC] Modules total           1
[CHIPSEC] *****************************************************************
```

Result	Meaning
PASSED	A **mitigation** to a known vulnerability has been detected
FAILED	A known **vulnerability** has been detected
WARNING	We have detected something that could be a vulnerability but **manual analysis is required** to confirm (inconclusive)
NOT_APPLICABLE	The issue checked by this module is not applicable to this platform. This result can be ignored
INFORMATION	This module does not check for a vulnerability. It just prints information about the system
ERROR	Something went wrong in the execution of CHIPSEC

https://github.com/chipsec/chipsec/blob/main/chipsec-manual.pdf

Practice #312 Exercise #2

- Run CHIPSEC main suite against:
    - AMI BIOS 01/18/2024
    - Dasharo (Slim Bootloader + UEFI) DEBUG
    - Dasharo (Slim Bootloader + UEFI) DEBUG with Intel Boot Guard

```
[CHIPSEC] ***************************  SUMMARY  ***************************
[CHIPSEC] Time elapsed            0.067
[CHIPSEC] Modules failed to run   0:
[CHIPSEC] Modules passed          17:
[+] PASSED: chipsec.modules.common.bios_kbrd_buffer
[+] PASSED: chipsec.modules.common.bios_smi
[+] PASSED: chipsec.modules.common.bios_ts
[+] PASSED: chipsec.modules.common.cpu.ia_untrusted
[+] PASSED: chipsec.modules.common.cpu.spectre_v2
[+] PASSED: chipsec.modules.common.debugenabled
[+] PASSED: chipsec.modules.common.ia32cfg
[+] PASSED: chipsec.modules.common.memconfig
[+] PASSED: chipsec.modules.common.remap
[+] PASSED: chipsec.modules.common.secureboot.variables
[+] PASSED: chipsec.modules.common.smm_code_chk
[+] PASSED: chipsec.modules.common.smm_dma
[+] PASSED: chipsec.modules.common.smrr
[+] PASSED: chipsec.modules.common.spd_wd
[+] PASSED: chipsec.modules.common.spi_fdopss
[+] PASSED: chipsec.modules.common.spi_lock
[+] PASSED: chipsec.modules.common.uefi.access_uefispec
( ... )
```

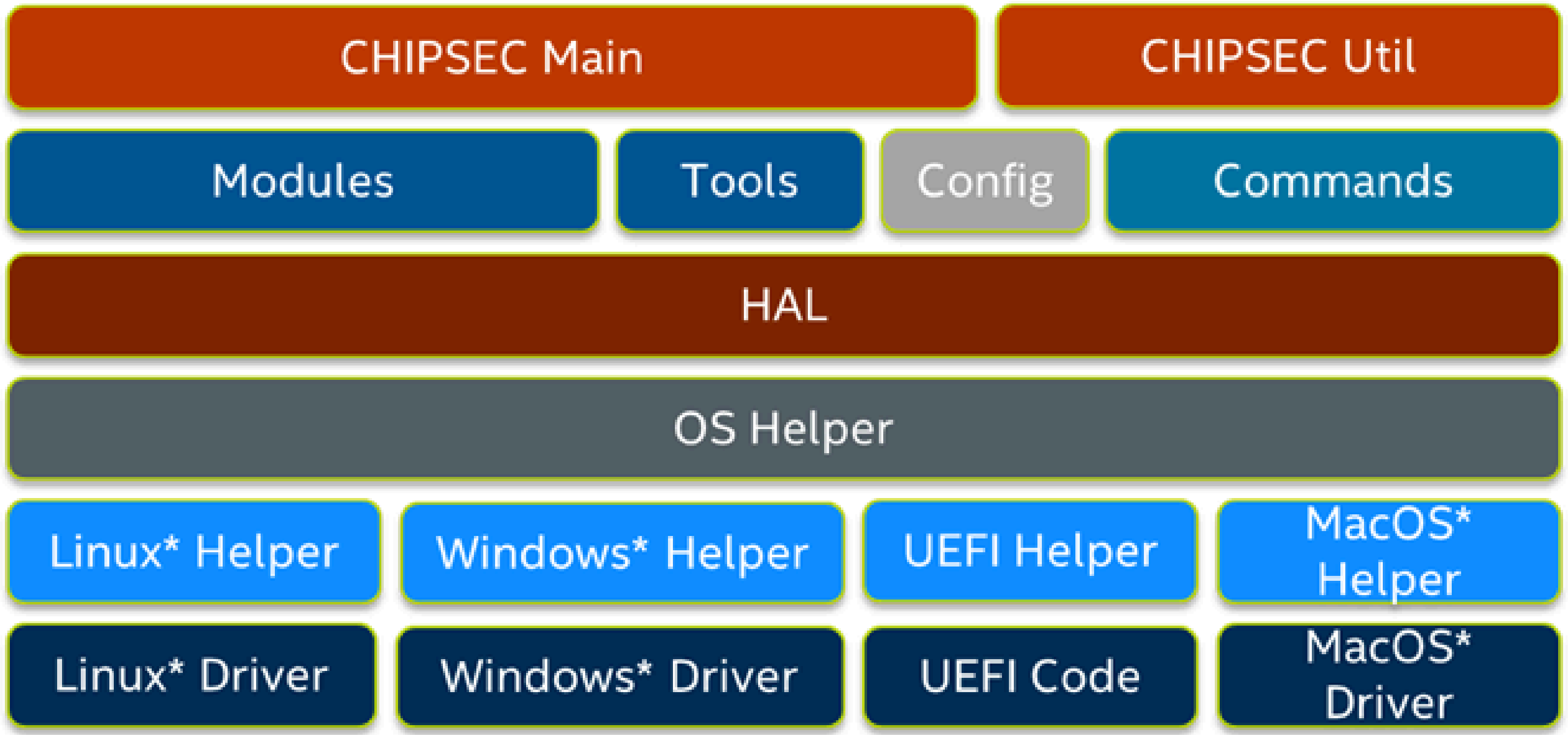
```
[CHIPSEC] Modules information     2:
[#] INFORMATION: chipsec.modules.common.cet
[#] INFORMATION: chipsec.modules.common.cpu.cpu_info
[CHIPSEC] Modules failed          4:
[-] FAILED: chipsec.modules.common.bios_wp
[-] FAILED: chipsec.modules.common.me_mfg_mode
[-] FAILED: chipsec.modules.common.spi_access
[-] FAILED: chipsec.modules.common.spi_desc
[CHIPSEC] Modules with warnings   1:
WARNING: chipsec.modules.common.rtclock
[CHIPSEC] Modules not applicable  7:
NOT APPLICABLE: chipsec.modules.common.memlock
NOT APPLICABLE: chipsec.modules.common.rom_armor
NOT APPLICABLE: chipsec.modules.common.sgx_check
NOT APPLICABLE: chipsec.modules.common.smm
NOT APPLICABLE: chipsec.modules.common.smm_addr
NOT APPLICABLE: chipsec.modules.common.smm_close
NOT APPLICABLE: chipsec.modules.common.smm_lock
[CHIPSEC] Modules total           32
[CHIPSEC] *****************************************************************
```

```
[CHIPSEC] ***************************  SUMMARY  ***************************
[CHIPSEC] Time elapsed            0.076
[CHIPSEC] Modules failed to run   0:
[CHIPSEC] Modules passed          13:
[+] PASSED: chipsec.modules.common.bios_kbrd_buffer
[+] PASSED: chipsec.modules.common.bios_ts
[+] PASSED: chipsec.modules.common.bios_wp
[+] PASSED: chipsec.modules.common.cpu.ia_untrusted
[+] PASSED: chipsec.modules.common.cpu.spectre_v2
[+] PASSED: chipsec.modules.common.debugenabled
[+] PASSED: chipsec.modules.common.ia32cfg
[+] PASSED: chipsec.modules.common.memconfig
[+] PASSED: chipsec.modules.common.remap
[+] PASSED: chipsec.modules.common.spd_wd
[+] PASSED: chipsec.modules.common.spi_fdopss
[+] PASSED: chipsec.modules.common.spi_lock
[+] PASSED: chipsec.modules.common.uefi.access_uefispec
```

```
[CHIPSEC] Modules information     2:
[#] INFORMATION: chipsec.modules.common.cet
[#] INFORMATION: chipsec.modules.common.cpu.cpu_info
[CHIPSEC] Modules failed          6:
[-] FAILED: chipsec.modules.common.bios_smi
[-] FAILED: chipsec.modules.common.me_mfg_mode
[-] FAILED: chipsec.modules.common.smm_dma
[-] FAILED: chipsec.modules.common.smrr
[-] FAILED: chipsec.modules.common.spi_access
[-] FAILED: chipsec.modules.common.spi_desc
[CHIPSEC] Modules with warnings   3:
WARNING: chipsec.modules.common.rtclock
WARNING: chipsec.modules.common.secureboot.variables
WARNING: chipsec.modules.common.smm_code_chk
[CHIPSEC] Modules not applicable  7:
NOT APPLICABLE: chipsec.modules.common.memlock
NOT APPLICABLE: chipsec.modules.common.rom_armor
NOT APPLICABLE: chipsec.modules.common.sgx_check
NOT APPLICABLE: chipsec.modules.common.smm
NOT APPLICABLE: chipsec.modules.common.smm_addr
NOT APPLICABLE: chipsec.modules.common.smm_close
NOT APPLICABLE: chipsec.modules.common.smm_lock
[CHIPSEC] Modules total           32
[CHIPSEC] *****************************************************************
```

```
[CHIPSEC] ***************************  SUMMARY  ***************************
[CHIPSEC] Time elapsed            0.085
[CHIPSEC] Modules failed to run   0:
[CHIPSEC] Modules passed          12:
[+] PASSED: chipsec.modules.common.bios_kbrd_buffer
[+] PASSED: chipsec.modules.common.bios_ts
[+] PASSED: chipsec.modules.common.bios_wp
[+] PASSED: chipsec.modules.common.cpu.ia_untrusted
[+] PASSED: chipsec.modules.common.cpu.spectre_v2
[+] PASSED: chipsec.modules.common.ia32cfg
[+] PASSED: chipsec.modules.common.memconfig
[+] PASSED: chipsec.modules.common.remap
[+] PASSED: chipsec.modules.common.spd_wd
[+] PASSED: chipsec.modules.common.spi_fdopss
[+] PASSED: chipsec.modules.common.spi_lock
[+] PASSED: chipsec.modules.common.uefi.access_uefispec
```

```
[CHIPSEC] Modules information     2:
[#] INFORMATION: chipsec.modules.common.cet
[#] INFORMATION: chipsec.modules.common.cpu.cpu_info
[CHIPSEC] Modules failed          7:
[-] FAILED: chipsec.modules.common.bios_smi
[-] FAILED: chipsec.modules.common.debugenabled
[-] FAILED: chipsec.modules.common.me_mfg_mode
[-] FAILED: chipsec.modules.common.smm_dma
[-] FAILED: chipsec.modules.common.smrr
[-] FAILED: chipsec.modules.common.spi_access
[-] FAILED: chipsec.modules.common.spi_desc
[CHIPSEC] Modules with warnings   3:
WARNING: chipsec.modules.common.rtclock
WARNING: chipsec.modules.common.secureboot.variables
WARNING: chipsec.modules.common.smm_code_chk
[CHIPSEC] Modules not applicable  7:
NOT APPLICABLE: chipsec.modules.common.memlock
NOT APPLICABLE: chipsec.modules.common.rom_armor
NOT APPLICABLE: chipsec.modules.common.sgx_check
NOT APPLICABLE: chipsec.modules.common.smm
NOT APPLICABLE: chipsec.modules.common.smm_addr
NOT APPLICABLE: chipsec.modules.common.smm_close
NOT APPLICABLE: chipsec.modules.common.smm_lock
[CHIPSEC] Modules total           32
[CHIPSEC] *****************************************************************
```



https://chipsec.github.io/development/Architecture-Overview.html

Configuration

- There are a lot of supported platforms
- There is also configuration specific to virtualization features like VT-x and VT-d
- This configuration contains platform or feature specific registers
- All register definitions are provided based on officially supported documentation
- There is a template for extending support for new platforms, it contains sections as:
  - Integrated devices
  - Memory Mapped I/O spaces (MMIO BARs)
  - I/O spaces (I/O BARs)
  - Memory ranges
  - Configuration registers
  - Controls (e.g. SMI global lock, BIOS interface lock)
- `chipsec/cfg/chipsec_cfg.xsd` contains schema for configuration XMLs

Quiz

Quiz

What is the main purpose of the CHIPSEC framework?

Quiz

What is the main purpose of the CHIPSEC framework?

- Platform security assessment

Quiz

What is the main purpose of the CHIPSEC framework?

- Platform security assessment

Which language is CHIPSEC written in?

Quiz

What is the main purpose of the CHIPSEC framework?

- Platform security assessment

Which language is CHIPSEC written in?

- Python and C

Quiz

What is the main purpose of the CHIPSEC framework?

- Platform security assessment

Which language is CHIPSEC written in?

- Python and C

Why it is not safe to run CHIPSEC in production?

Quiz

What is the main purpose of the CHIPSEC framework?

- Platform security assessment

Which language is CHIPSEC written in?

- Python and C

Why it is not safe to run CHIPSEC in production?

- CHIPSEC kernel drivers provide direct access to hardware what can undermine platform security posture of the system.
- Safety checks are disabled in OS to be able to load unsigned drivers.

Quiz

What is the main purpose of the CHIPSEC framework?

- Platform security assessment

Which language is CHIPSEC written in?

- Python and C

Why it is not safe to run CHIPSEC in production?

- CHIPSEC kernel drivers provide direct access to hardware what can undermine platform security posture of the system.
- Safety checks are disabled in OS to be able to load unsigned drivers.

Which operating systems/environments does CHIPSEC support?

Quiz

What is the main purpose of the CHIPSEC framework?

- Platform security assessment

Which language is CHIPSEC written in?

- Python and C

Why it is not safe to run CHIPSEC in production?

- CHIPSEC kernel drivers provide direct access to hardware what can undermine platform security posture of the system.
- Safety checks are disabled in OS to be able to load unsigned drivers.

Which operating systems/environments does CHIPSEC support?

- Windows, Linux, Mac OS X and UEFI

Production approach

- Run the CHIPSEC scan tool to check if all lockable registers are indeed locked, especially the flash lock and management.
- Run the CHIPSEC scan tool in a special boot mode, such as S3 resume, S4 resume, recovery mode, or even manufacturing mode, to check if the end user can trigger the manufacturing mode restart.

Conclusion

- CHIPSEC is very capable platform security assessment framework which can be adjusted to required threat model, but to leverage it one must by very familiar with test it implements and follow its development.

Q&A

Introductory fwupd

Goals of the Presentation

In this presentation, we aim to:

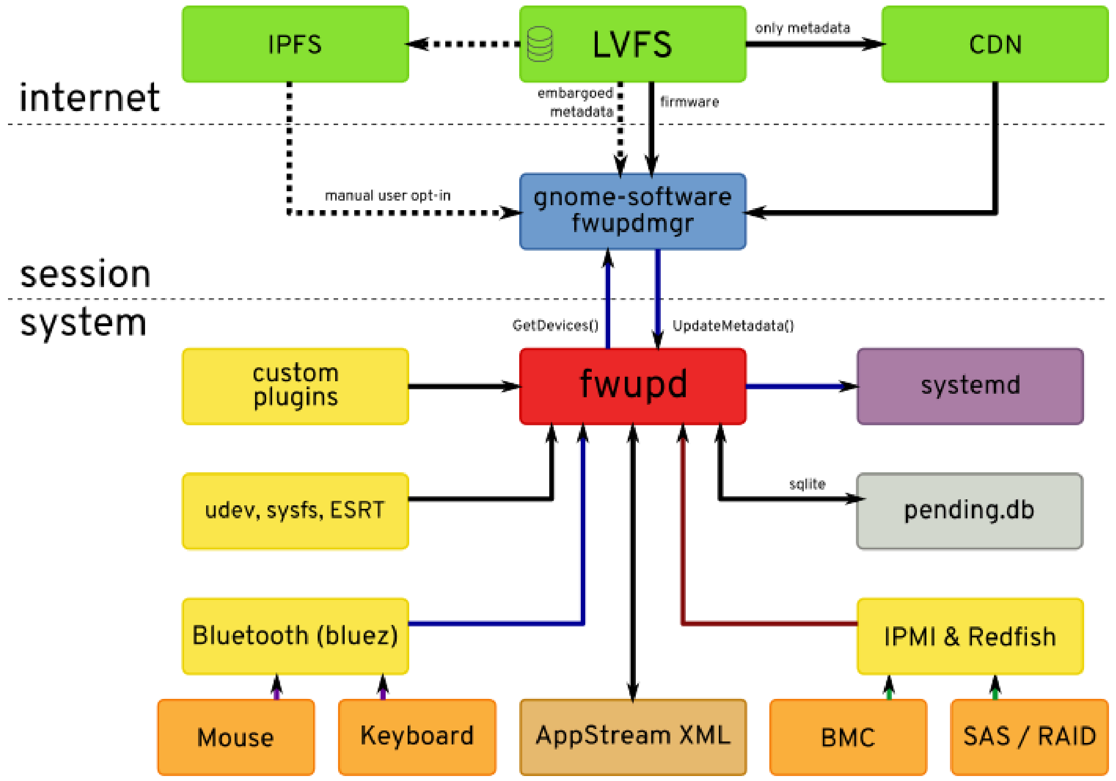
- Learn what is fwupd Host Security ID (HSI) and how to use it.
- Understand HSI reports.
- Understand fwupd HSI internals and how add custom tests.

https://fwupd.org/

fwupd: An overview

- Outdated firmware makes devices vulnerable to the different attacks
- fwupd project can query supported hardware for the current firmware versions and also deploy new firmware versions to devices
  - Latest version: 2.0.13 (Jul 22 2025)
  - License: LGPLv2.1
  - Written in C
  - Used by users, typically with a GUI
- LVFS (Linux Vendor Firmware Service) is a web service that provides information about available firmware updates. It can be used by the OEMs to upload firmware archives downloaded by the users
  - License: GPLv2
  - Used by vendors: OEMs and ODMs
- Both projects were created by Richard Hughes of Red Hat and are now Linux Foundation projects

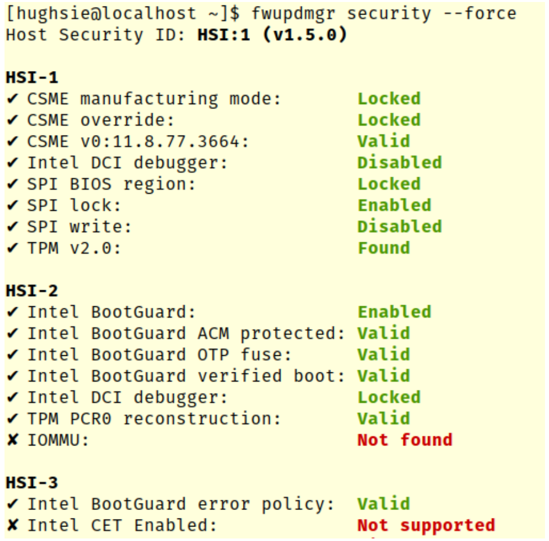
fwupd/LVFS architecture



https://lvfs.readthedocs.io/en/latest/intro.html

Problem statement

- There is no easy way to determine level of platform security
- What we can do today
  - obtain access to NDA documentation
  - obtain access to source code or skill to perform RE tasks
  - get deep technical knowledge of hardware, firmware, bootloader, hypervisor, OS and system software
  - explore infinite space of new frameworks
- CHIPSEC was one of the first and most successful frameworks to detect most common platform security issues
  - of course not without own issues: UEFI Secure Boot had to be disabled, some tests required unsigned Linux kernel modules
  - Intel-centric
  - not all modules are OSS
- Now we are in era of fwupd/LVFS



```
[hughsie@localhost ~]$ fwupdmgr security --force
Host Security ID: HSI:1 (v1.5.0)

HSI-1
✔ CSME manufacturing mode:       Locked
✔ CSME override:                 Locked
✔ CSME v0:11.8.77.3664:          Valid
✔ Intel DCI debugger:            Disabled
✔ SPI BIOS region:               Locked
✔ SPI lock:                      Enabled
✔ SPI write:                     Disabled
✔ TPM v2.0:                      Found

HSI-2
✔ Intel BootGuard:               Enabled
✔ Intel BootGuard ACM protected: Valid
✔ Intel BootGuard OTP fuse:      Valid
✔ Intel BootGuard verified boot: Valid
✔ Intel DCI debugger:            Locked
✔ TPM PCR0 reconstruction:       Valid
✘ IOMMU:                         Not found

HSI-3
✔ Intel BootGuard error policy:  Valid
✘ Intel CET Enabled:             Not supported
```

https://blogs.gnome.org/hughsie/2020/10/26/new-fwupd-1-5-0-release/

- Since fwupd already possess vast information about firmware security features
  - Intel Boot Guard entablement status
  - SPI flash protection
  - TPM 2.0 presence
- In version v1.5.0 fwupd introduced **HSI** (Host Security ID), which gives ability to easily determine platform security.

HSI Specification

- HSI was turned into official specification authored by:
  - Richard Hughes
  - Mario Limonciello
  - Alex Bazhaniuk
  - Alex Matrosov
- With essential goal of providing easy-to-understand information to people buying hardware
- https://github.com/fwupd/fwupd/blob/main/docs/hsi.md.in
- https://fwupd.github.io/libfwupdplugin/hsi.html

HSI levels

- `HSI:0` - **Insecure:** None or few firmware protections
  - The lowest security level with little or no detected firmware protections. This is the default security level if no tests can be run or some tests in the next security level have failed.
- `HSI:1` - **Critical:** Most basic protections, may or may not be sufficient to protect against remote attacks
  - This security level corresponds to the most basic of security protections considered essential by security professionals. Any failures at this level would have critical security impact and could likely be used to compromise the system firmware without physical access.
- `HSI:2` - **Risky:** Protects against more difficult attacks
  - This security level corresponds to firmware security issues that pose a theoretical concern or where any exploit would be difficult or impractical to use. At this level various technologies may be employed to protect the boot process from modification by an attacker with local access to the machine.

HSI levels

- `HSI:3` - **Protected:** Few issues, high security
  - This security level corresponds to out-of-band protection of the system firmware perhaps including recovery.
- `HSI:4` - **Secure:** No issues, several layers of protection for firmware
  - The system is corresponding several kind of encryption and execution protection for the system firmware.
- `HSI:5` - **Secure Proven:** Out-of-band attestation of firmware, no tests implemented at this moment.
  - This security level corresponds to out-of-band attestation of the system firmware. There are currently no tests implemented for HSI:5 and so this security level cannot yet be obtained.

Example HSI ID

- LVFS does not have to contain firmware update to be able to obtain HSI ID
- The HSI can be determined using `fwupdmgr`:

https://asciinema.org/a/6925gtnsspoEEoOWElHHvKpzd?ssize=big

fwupd HSI security report

Practice #308 Exercise #1

- Run `fwupdmgr security --force` in Linux for:
  - AMI BIOS 01/18/2024
  - Dasharo (Slim Bootloader + UEFI) DEBUG with Intel Boot Guard
  - Dasharo (Slim Bootloader + UEFI) DEBUG

```
Host Security ID: HSI:0! (v2.0.8)

HSI-1
✔ BIOS firmware updates:          Enabled
✔ csme override:                  Locked
✔ csme v0:16.50.5.1303:           Valid
✔ Platform debugging:             Not supported
✔ SPI write:                      Disabled
✔ Supported CPU:                  Valid
✔ TPM empty PCRs:                 Valid
✔ TPM v2.0:                       Found
✔ UEFI bootservice variables:     Locked
✘ csme manufacturing mode:        Unlocked
✘ SPI lock:                       Disabled
✘ SPI BIOS region:                Unlocked
✘ UEFI platform key:              Invalid

HSI-2
✔ Intel BootGuard:                Enabled
✔ IOMMU:                          Enabled
✔ Platform debugging:             Not supported
✔ TPM PCR0 reconstruction:        Valid
✘ Intel BootGuard ACM protected: Invalid
✘ Intel BootGuard OTP fuse:      Invalid
✘ Intel BootGuard verified boot: Invalid
```

```
HSI-3
✔ CET Platform:                 Supported
✔ Pre-boot DMA protection:      Enabled
✘ Intel BootGuard error policy: Invalid
✘ Suspend-to-idle:              Disabled
✘ Suspend-to-ram:               Enabled

HSI-4
✔ SMAP:                         Enabled
✘ Encrypted RAM:                Not supported

Runtime Suffix -!
✔ fwupd plugins:                Untainted
✔ Linux kernel:                 Untainted
✘ CET OS Support:               Not supported
✘ Linux kernel lockdown:        Disabled
✘ Linux swap:                   Unencrypted
✘ UEFI db:                      Invalid
✘ UEFI secure boot:             Disabled
```

```
WARNING: UEFI capsule updates not available or enabled in firmware setup
See https://github.com/fwupd/fwupd/wiki/PluginFlag:capsules-unsupported for more information.
Host Security ID: HSI:INVALID:chassis[unset] (v2.0.8)

HSI-1
✔ csme override:                 Locked
✔ csme v0:16.50.10.1351:         Valid
✔ SPI write:                     Disabled
✔ SPI lock:                      Enabled
✔ SPI BIOS region:               Locked
✔ Supported CPU:                 Valid
✔ TPM empty PCRs:                Valid
✔ TPM v2.0:                      Found
✔ UEFI bootservice variables:    Locked
✘ BIOS firmware updates:         Disabled
✘ csme manufacturing mode:       Unlocked
✘ Platform debugging:            Enabled

HSI-2
✔ Intel BootGuard ACM protected: Valid
✔ Intel BootGuard:               Enabled
✔ Intel BootGuard verified boot: Valid
✔ IOMMU:                         Enabled
✔ Platform debugging:            Locked
✔ TPM PCR0 reconstruction:       Valid
✘ Intel BootGuard OTP fuse:      Invalid
```

```
HSI-3
✔ CET Platform:                    Supported
✔ Intel BootGuard error policy:    Valid
✔ Pre-boot DMA protection:         Enabled
✘ Suspend-to-idle:                 Disabled
✘ Suspend-to-ram:                  Enabled

HSI-4
✔ SMAP:                            Enabled
✘ Encrypted RAM:                   Not supported

Runtime Suffix -!
✔ fwupd plugins:                   Untainted
✔ Linux kernel:                    Untainted
✘ CET OS Support:                  Not supported
✘ Linux kernel lockdown:           Disabled
✘ Linux swap:                      Unencrypted
✘ UEFI secure boot:                Disabled
```

Host Security Events

```
Host Security Events
  2025-08-24 23:00:54:  ✔ SPI lock changed: Disabled → Enabled
  2025-08-24 23:00:54:  ✔ SPI BIOS region changed: Unlocked → Locked
  2025-08-24 23:00:54:  ✘ BIOS firmware updates changed: Enabled → Disabled
  2025-08-24 23:00:54:  ✘ Platform debugging changed: Not supported → Enabled
  2025-08-24 23:00:54:  ✔ Intel BootGuard ACM protected changed: Invalid → Valid
  2025-08-24 23:00:54:  ✔ Intel BootGuard verified boot changed: Invalid → Valid
  2025-08-24 23:00:54:  ✔ Platform debugging changed: Not supported → Locked
  2025-08-24 23:00:54:  ✔ Intel BootGuard error policy changed: Invalid → Valid
```

Conclusion

- We discussed CHIPSEC and fwupd as tools for BIOS security assessment.
- Project documentation and associated specification clearly define security criteria that should be met.
- Based on threat model tools can be adjusted to specific use cases.

Q&A