Automating Low-Level Firmware Validation with Robot Framework

FOSDEM 2025

Testing and Continuous Delivery

Maciej Pijanowski

3MDEB

whoami

Maciej Pijanowski
Engineering Manager

- @macpijan
- maciej.pijanowski@3mdeb.com
- linkedin.com/in/maciej-pijanowski-9868ab120

- Over 8 years in 3mdeb
- Open-source contributor
- Interested in:
  - build systems (e.g., Yocto)
  - embedded, OSS, OSF
  - firmware/OS security

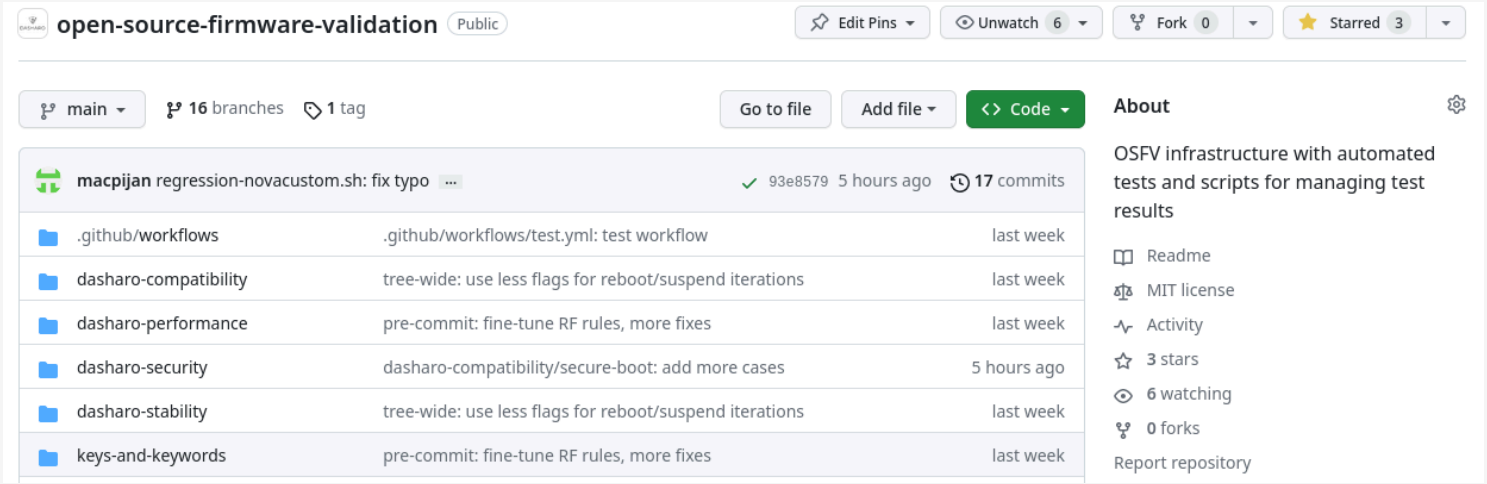
Who we are ?

- coreboot licensed service providers since 2016 and leadership participants
- UEFI Adopters since 2018
- Yocto Participants and Embedded Linux experts since 2019
- Official consultants for Linux Foundation fwupd/LVFS project since 2020
- IBM OpenPOWER Foundation members since 2020

Agenda

- History
- Dasharo OSFV and RF
- Hardware support
- Interfacing with hardware
- Example scenarios
- Q&A

History



- We've been using OSFV at least since 2018
  - when validating PC Engines coreboot releases on a monthly basis
  - executed over **50k** tests
  - publicly releasing **150+** binaries of open-source firmware
- Initially, it was an internal project

https://github.com/Dasharo/open-source-firmware-validation

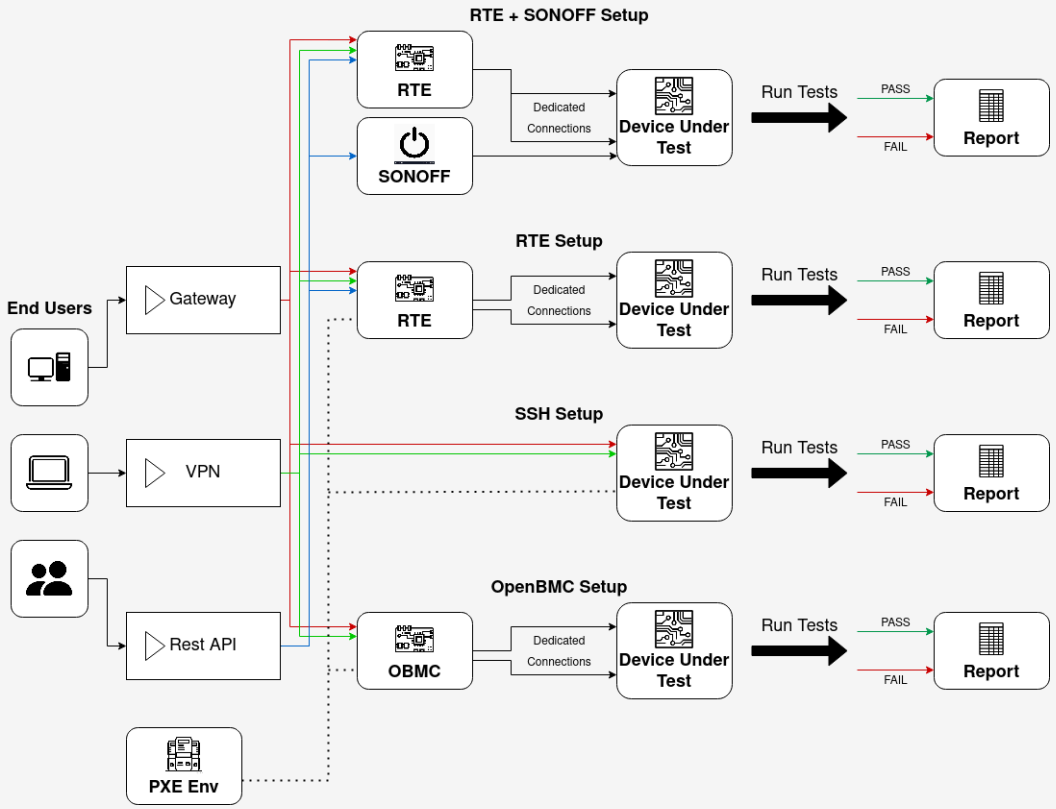
- Published as open-source project Sep 2023 (small part of it earlier)
- Use cases of Dasharo OSFV
  - validation of (open-source) firmware
    - can be used for any firmware, really
  - testing Dasharo firmware releases
  - test-driven bug fixing (and adding new features)
  - regression testing
    - after introducing new features
    - after major changes (update base from upstream project)
  - mainly Dasharo with UEFI payload right now
- Scripts written in:
  - mostly Robot Framework (keywords, test suites)
  - some Python (sometimes more suitable than RF)
  - shell scripts (mostly some wrappers for test execution)

Robot Framework

- Robot Framework is a generic open source automation framework
- It can be used for test automation and Robotic Process Automation (RPA)
- Used widely for web apps testing (with Selenium), and many more
- Used by OpenBMC (firmware, embedded Linux) for test automation
  - https://github.com/openbmc/openbmc-test-automation
- Active community, quality documentation
  - https://robotframework.org/#community
  - https://docs.robotframework.org/docs
- Robot Framework Conference
  - https://robocon.io/

Supported Platforms

Regression Testing Architecture



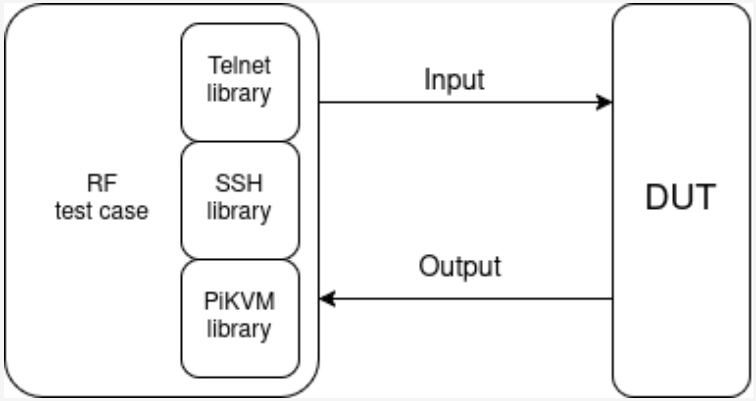
Power control

- WiFi plug with custom firmware
- https://tasmota.github.io/docs/devices/Sonoff-S26-Smart-Socket/

- DC relay
- OSHW control board
- HTTP API to control gpio/power
- https://github.com/3mdeb/rte-schematics
- https://github.com/3mdeb/RteCtrl

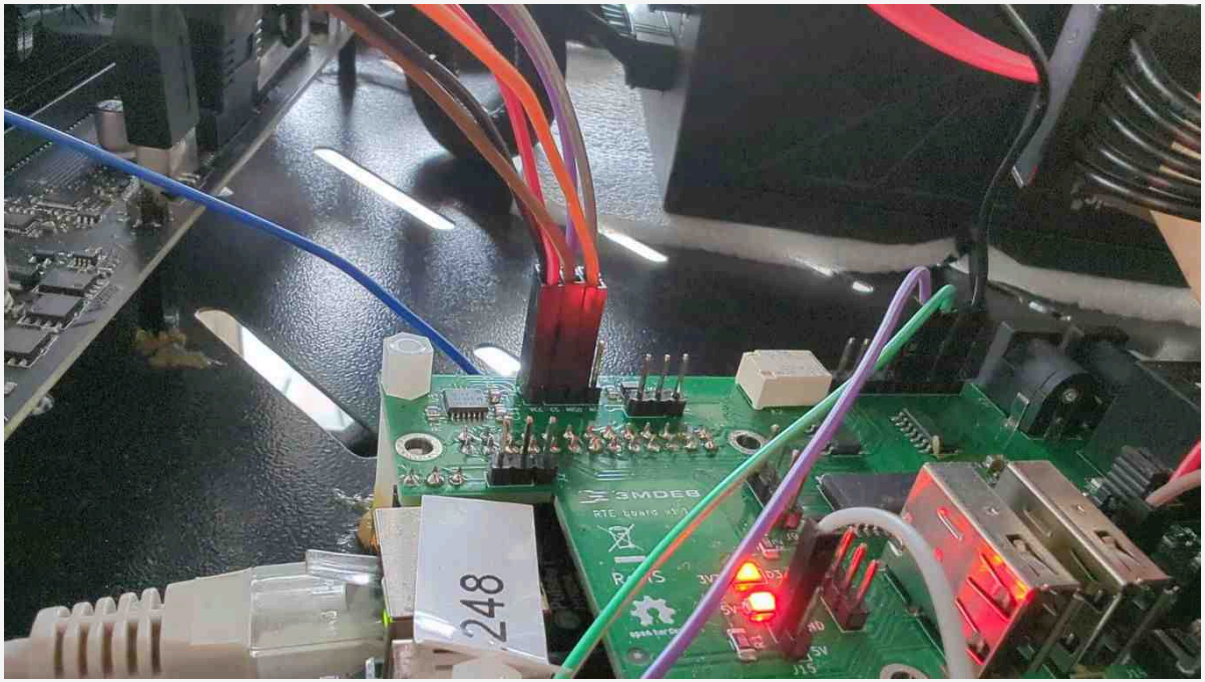
Test execution interface

- Output
  - gather logs via serial port
  - SSH
- Input
  - serial port
  - USB keyboard emulation (via PiKVM)
  - SSH



GPIO control

- Power / reset button
- Power LED status
- Custom GPIOs



Firmware flashing

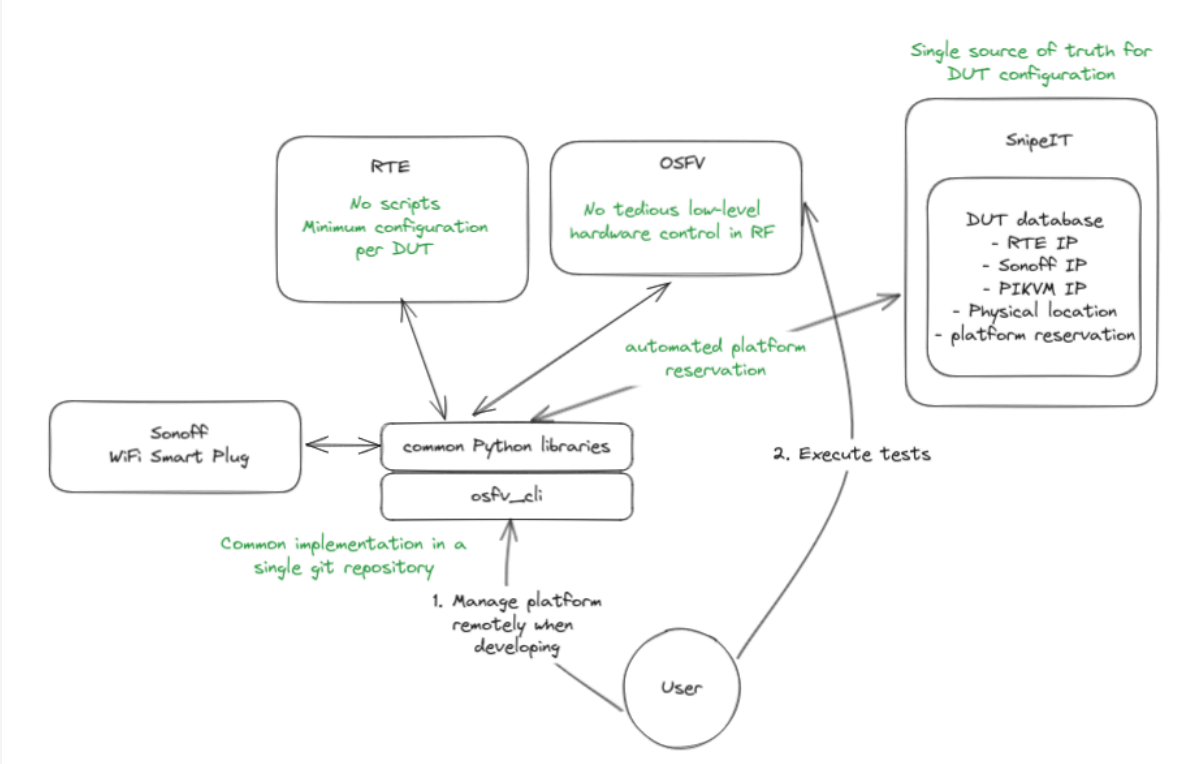
osfv_cli - developer usage scenario

https://github.com/Dasharo/osfv-scripts/tree/main/osfv_cli

```
# Reserve platform
osfv_cli snipeit check_out --rte_ip $RTE_IP
# Read backup firmware
osfv_cli rte --rte_ip $RTE_IP flash read --rom backup.rom
# Flash new firmware
osfv_cli rte --rte_ip $RTE_IP flash read --rom backup.rom
# Enable power supply
osfv_cli rte --rte_ip $RTE_IP pwr psu on
# Get logs from serial
osfv_cli rte --rte_ip $RTE_IP serial
# Reset platform
osfv_cli rte --rte_ip $RTE_IP pwr reset
```

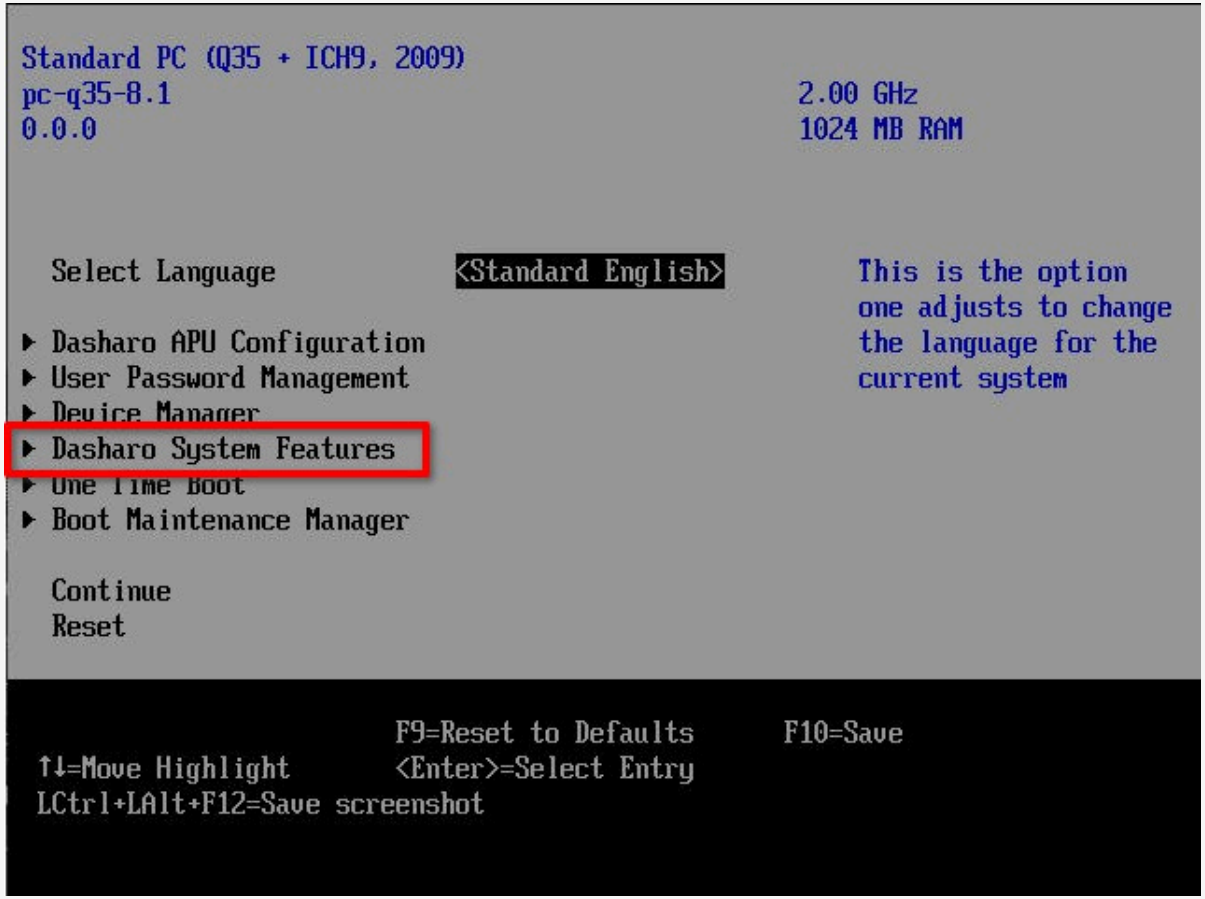
osfv_cli integration

- Integrate low-level hardware operations into Python libraries
- Reuse the same libraries by test framework and CLI tool

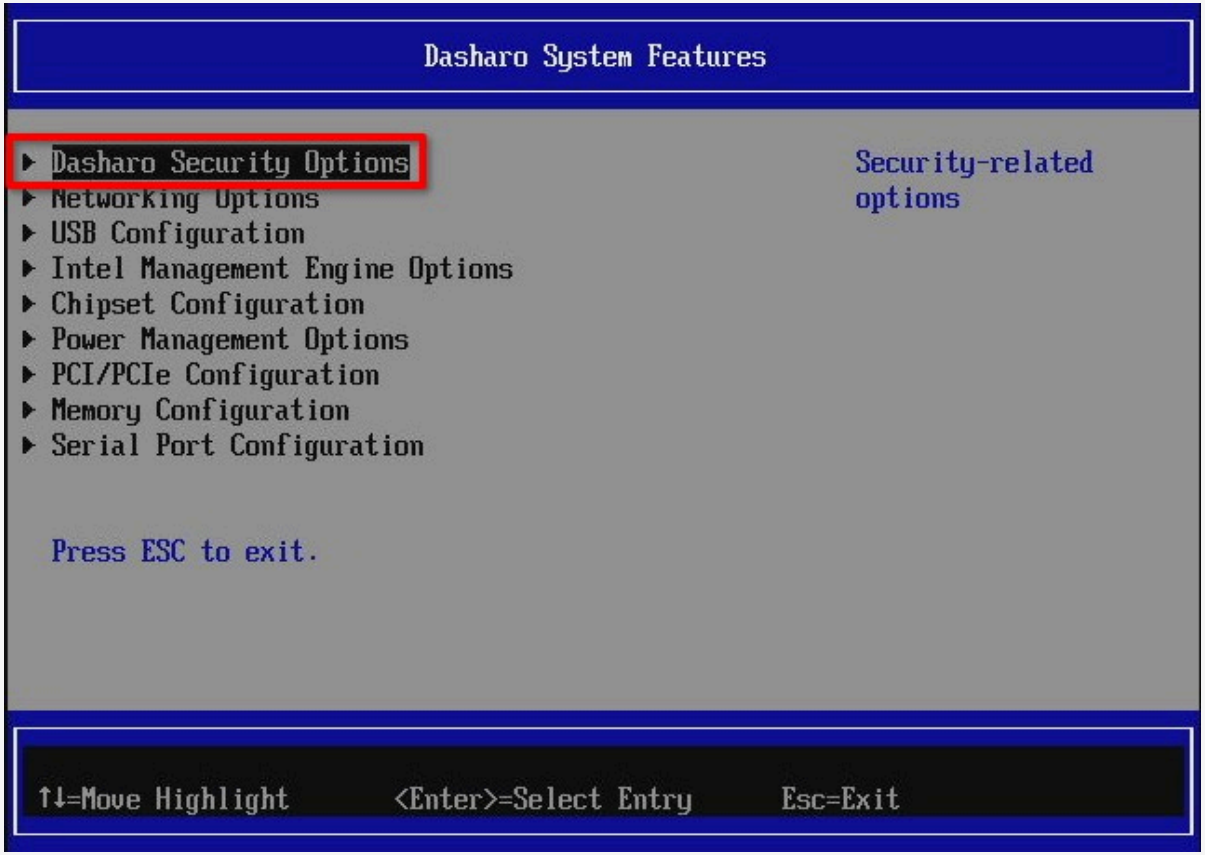


Example test scenario

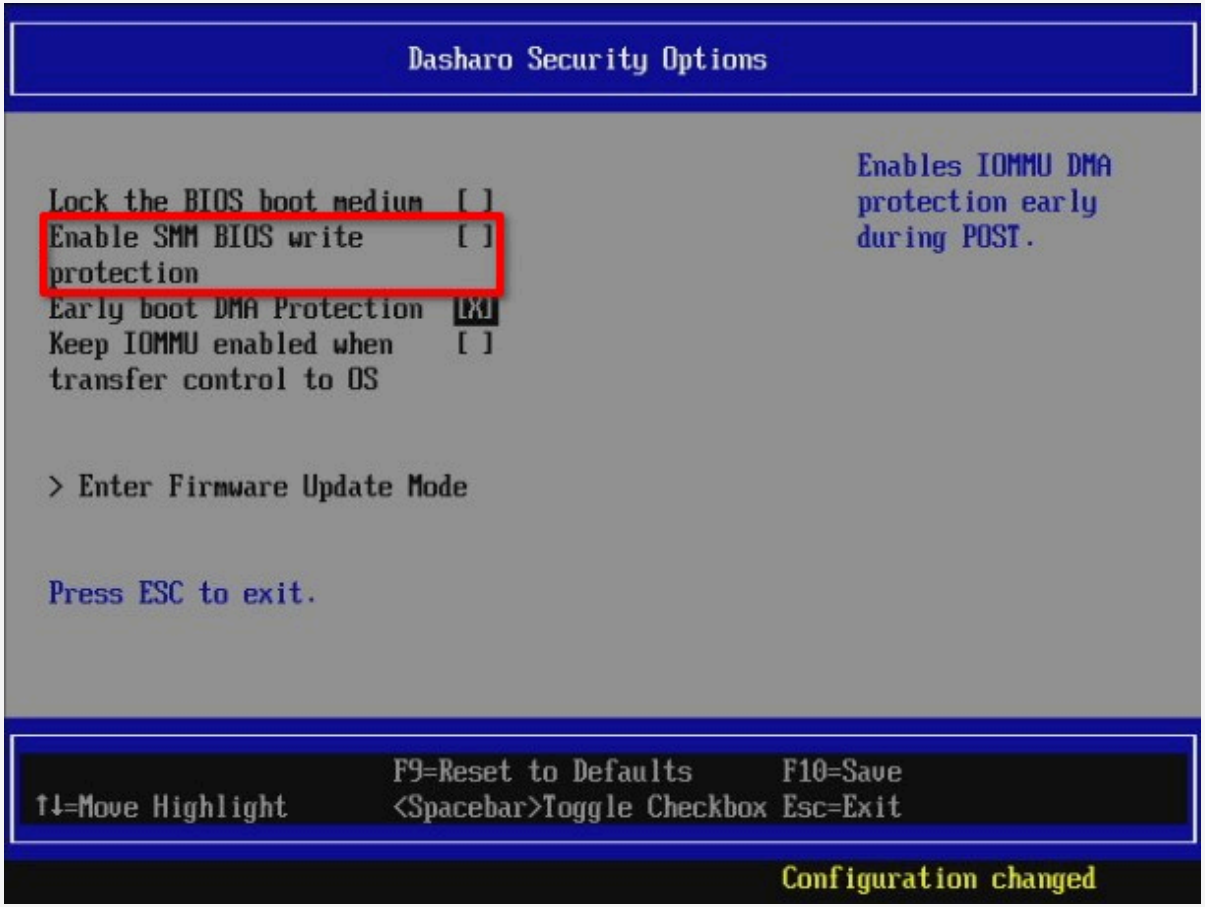
Example test scenario



Example test scenario



Example test scenario



Example test scenario

- Code

```
SMM001.001 SMM BIOS write protection enabling (Ubuntu)
    Power On
    ${setup_menu}=    Enter Setup Menu Tianocore And Return Construction
    ${dasharo_menu}=    Enter Dasharo System Features    ${setup_menu}
    ${network_menu}=    Enter Dasharo Submenu    ${dasharo_menu}    Dasharo Security Options
    Set Option State    ${network_menu}    Enable SMM BIOS write    ${TRUE}
    Save Changes And Reset
    Boot System Or From Connected Disk    ubuntu
    Login To Linux
    Switch To Root User
    ${out_flashrom}=    Execute Command In Terminal    flashrom -p internal
    Should Contain    ${out_flashrom}    SMM protection is enabled
```

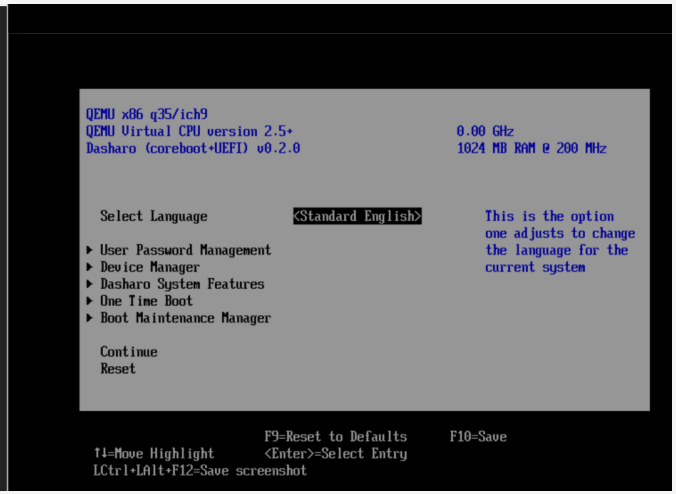
- Run command

```
CONFIG=msi-pro-z690-a-ddr5 \
RTE_IP=AAA.BBB.CCC.DDD \
DEVICE_IP=EEE.FFF.GGG.HHH \
FW_FILE=msi_ms7d25_v1.1.4_ddr5.rom \
./scripts/run.sh dasharo-security/smm-bios-write-protection.robot
```

Running in QEMU

- Spin up QEMU with Dasharo firmware
  - `./scripts/ci/qemu-run.sh graphic firmware`
- Run test
- Observe the robot execution in console
- Observe how the machine is being controlled in the QEMU window

```
(venv) macpijan in ~/projects/github/dasharo/open-source-firmware-validation on stop-cloud-d
l ● ● λ SNIPEIT_NO=true RTE_IP=127.0.0.1 CONFIG=qemu  DEVICE_IP=127.0.0.1 ./scripts/run.sh d
asharo-compatibility/custom-boot-menu-key.robot
Logs will be saved at logs/qemu/2025_02_01_00_27_55
Watch "logs/qemu/2025_02_01_00_27_55/dasharo-compatibility/custom-boot-menu-key.robot_debug.
log" to monitor the progress of the test
==============================================================================
Custom-Boot-Menu-Key
==============================================================================
CBK001.${VAR_ID} Custom boot menu key :: Check whether the DUT is ... | PASS |
------------------------------------------------------------------------------
CBK002.001 Custom setup menu key :: Check whether the DUT is confi... | PASS |
------------------------------------------------------------------------------
Custom-Boot-Menu-Key                                                  | PASS |
2 tests, 2 passed, 0 failed
==============================================================================
Debug:   /home/macpijan/projects/github/dasharo/open-source-firmware-validation/logs/qemu/20
25_02_01_00_27_55/dasharo-compatibility/custom-boot-menu-key.robot_debug.log
Output:  /home/macpijan/projects/github/dasharo/open-source-firmware-validation/logs/qemu/20
25_02_01_00_27_55/dasharo-compatibility/custom-boot-menu-key.robot_out.xml
Log:     /home/macpijan/projects/github/dasharo/open-source-firmware-validation/logs/qemu/20
25_02_01_00_27_55/dasharo-compatibility/custom-boot-menu-key.robot_log.html
Report:  /home/macpijan/projects/github/dasharo/open-source-firmware-validation/logs/qemu/20
25_02_01_00_27_55/dasharo-compatibility/custom-boot-menu-key.robot_report.html
(venv) macpijan in ~/projects/github/dasharo/open-source-firmware-validation on stop-cloud-d
l ● ● λ
```



Community

- Standard GH issues and PR flow for contributors
  - https://github.com/Dasharo/open-source-firmware-validation
- Join Dasharo Matrix Space
  - https://matrix.to/#/#dasharo:matrix.org
- Join Dasharo OSFV Matrix room
  - https://matrix.to/#/#osfv:matrix.3mdeb.com

Contact us

We are open to cooperate and discuss

- contact@3mdeb.com
- facebook.com/3mdeb
- @3mdeb_com
- linkedin.com/company/3mdeb
- https://3mdeb.com
- Book a call
- Sign up for the newsletter

Feel free to contact us if you believe we can help you in any way. We are always open to cooperate and discuss.

Q&A

Extras

Getting started with OSFV

- What do I need to learn first?
- How do I run existing test?
- How do I write a new test?
- How do I add support for a new platform?

What do I need to learn first?

- Some basic RF knowledge
  - go through basics in User Guide:
  - go through basic RF libraries:
  - `BuildIn`, `Collections`, `Strings`, `Telnet`
  - add them to your bookmarks, you will need them often
  - check out SSHLibrary
- Some basic Python knowlednge
  - there are plenty of learning materials, pick your favourite one

How do I run existing test?

- Consult README for:
  - supported platforms
  - getting started
  - running single tests
- Look through existing tests in:
  - `dasharo-compatibility`
  - `dasharo-security`
  - `dasharo-performance`
  - `dasharo-stability`

How do I run existing test?

- Start with QEMU to learn how it works
- Spin up QEMU with Dasharo firmware
  - `./scripts/ci/qemu-run.sh graphic firmware`
- Run test
- Observe the robot execution in console
- Observe how the machine is being controlled in the QEMU window

How do I run existing test?

- Check if selected test is supported by the given platform
- In `platform-configs/qemu.robot`:

```
${CUSTOM_BOOT_MENU_KEY_SUPPORT}=    ${TRUE}
```

- In `dasharo-compatibility/custom-boot-menu-key.robot`

```
Skip If    not ${CUSTOM_BOOT_MENU_KEY_SUPPORT}    CBK001.001 not supported
```

- Example on hardware:

```
$ robot  -L TRACE -v config:protectli-vp4630 -v rte_ip:192.168.10.244 dasharo-compatibility/custom-boot-menu-key.robot

==============================================================================
Custom-Boot-Menu-Key
==============================================================================
CBK001.001 Custom boot menu key :: Check whether the DUT is config... | PASS |
------------------------------------------------------------------------------
CBK002.001 Custom setup menu key :: Check whether the DUT is confi... | PASS |
------------------------------------------------------------------------------
Custom-Boot-Menu-Key                                                  | PASS |
2 tests, 2 passed, 0 failed
==============================================================================
Output:  /home/macpijan/projects/github/dasharo/open-source-firmware-validation/output.xml
Log:     /home/macpijan/projects/github/dasharo/open-source-firmware-validation/log.html
Report:  /home/macpijan/projects/github/dasharo/open-source-firmware-validation/report.html
```

How do I write a new test?

- Refer to the existing tests
  - `self-tests` are good examples
  - other commonly used tests
    - `dasharo-security/bios-lock.robot`
    - `dasharo-security/me-neuter.robot`
    - `dasharo-security/smm-bios-write-protection.robot`
    - `dasharo-compatibility/network-boot.robot`
    - `dasharo-compatibility/network-boot-utilities.robot`
  - some tests may currently not work or be of a lower quality

How do I add support for a new platform?

- Pick a similar board from `platform-config` and adjust it
  - config documentation: `docs/adding-new-platforms.md`
- Power control
  - RTE, Sonoff WiFi Smart Plug, or both (or something else, which is not supported yet)
- Flashing
  - preferably external flashing is supported - like SOIC clip connected to RTE all the time
- Serial connection
  - ser2net service to expose serial via telnet
- Hardware setup may be complex
  - https://docs.dasharo.com/variants/asus_kgpe_d16/setup/