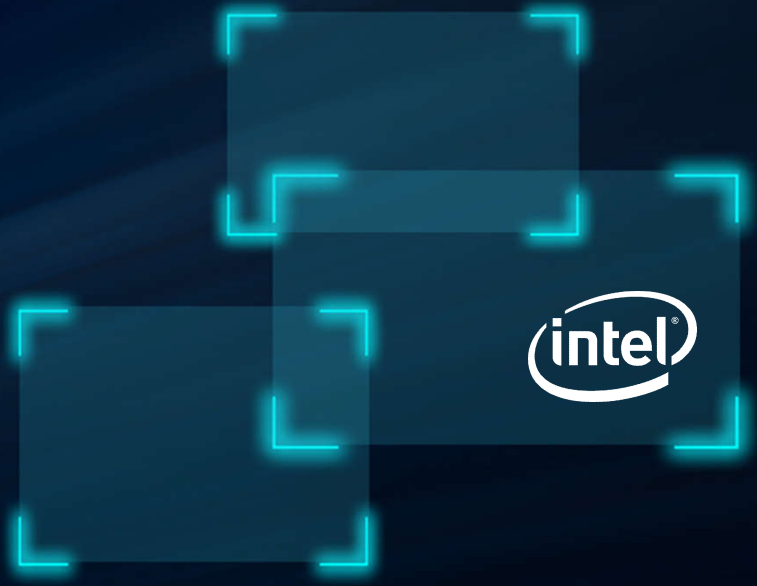


Impact of platform firmware on Linux kernel

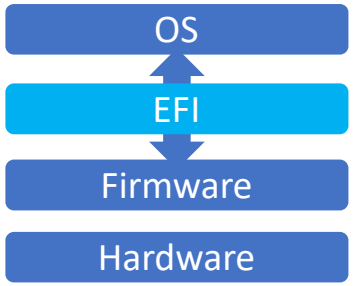
Megha Dey, Sai Praneeth Prakhya
Intel Open Source Technology Center

AGENDA

- Introduction to UEFI
- UEFI implementation bug crashing Linux
- Linux EFI subsystem bug affecting system stability
- Introduction to Linux UEFI Validation (LUV) project
- Make your UEFI system Linux ready with LUV

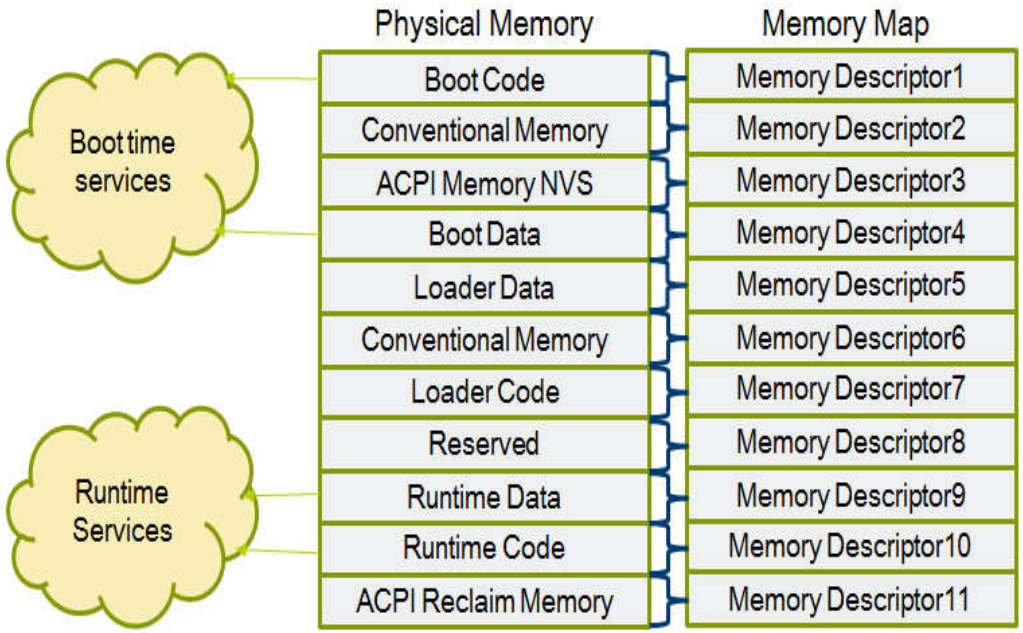
Introduction to UEFI

- UEFI forum members – Many biggies like AMD, Apple, Dell, IBM, Microsoft etc.. and of course Intel.
- UEFI firmware, not only in PC and servers but also in embedded space.
- UEFI (Unified Extensible Firmware Interface) is a specification that defines a software interface between an OS and platform firmware.



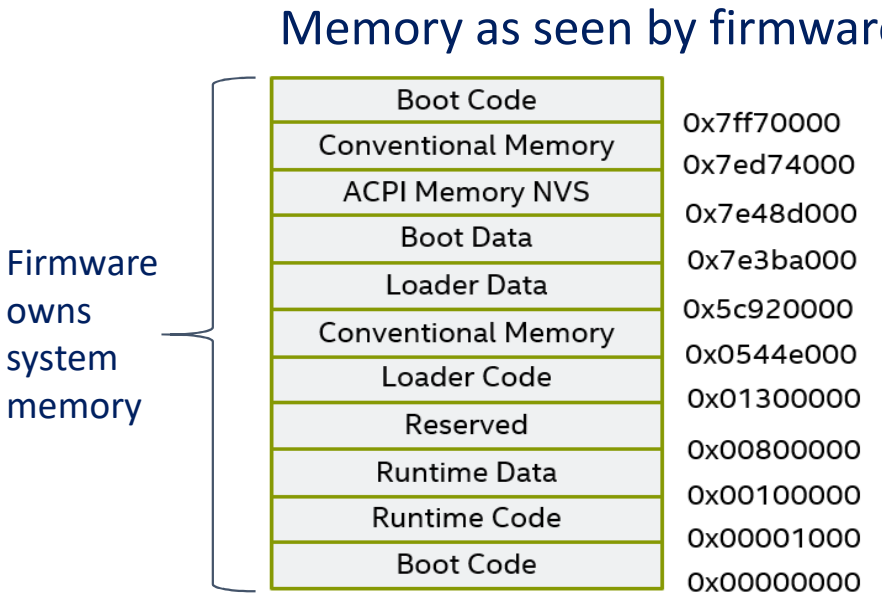
UEFI Services

- Implemented as API's – has code and data regions in memory.
- UEFI offers two types of services:
  1. Boot time services: API's available to OS loader before ExitBootServices()
  2. Runtime services: API's available to OS loader **before and after** ExitBootServices()

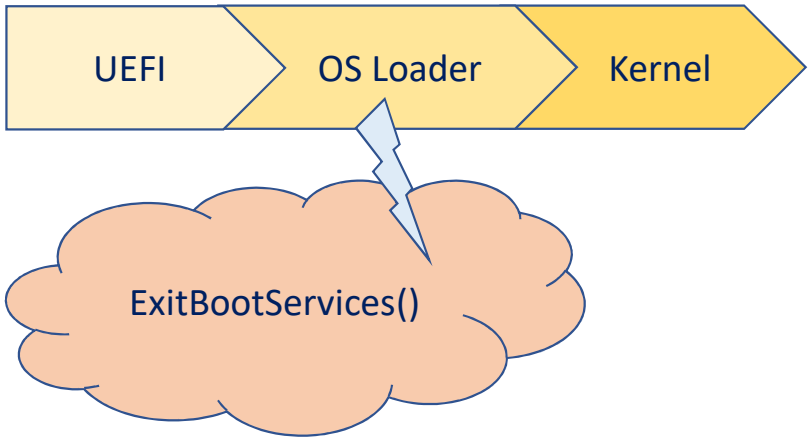


Boot Process – Before ExitBootServices

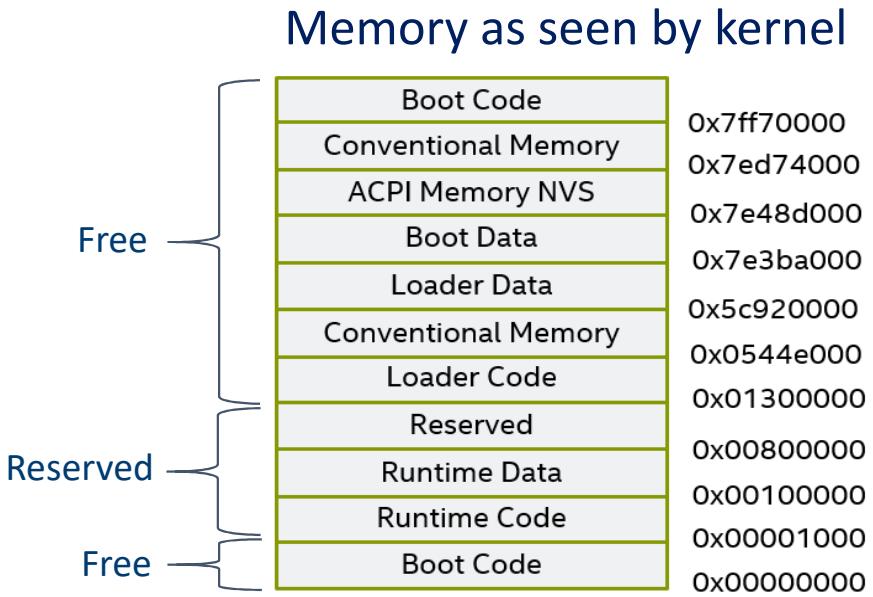
- Firmware controls system memory
- Memory accesses in physical mode



Boot Process – After ExitBootServices

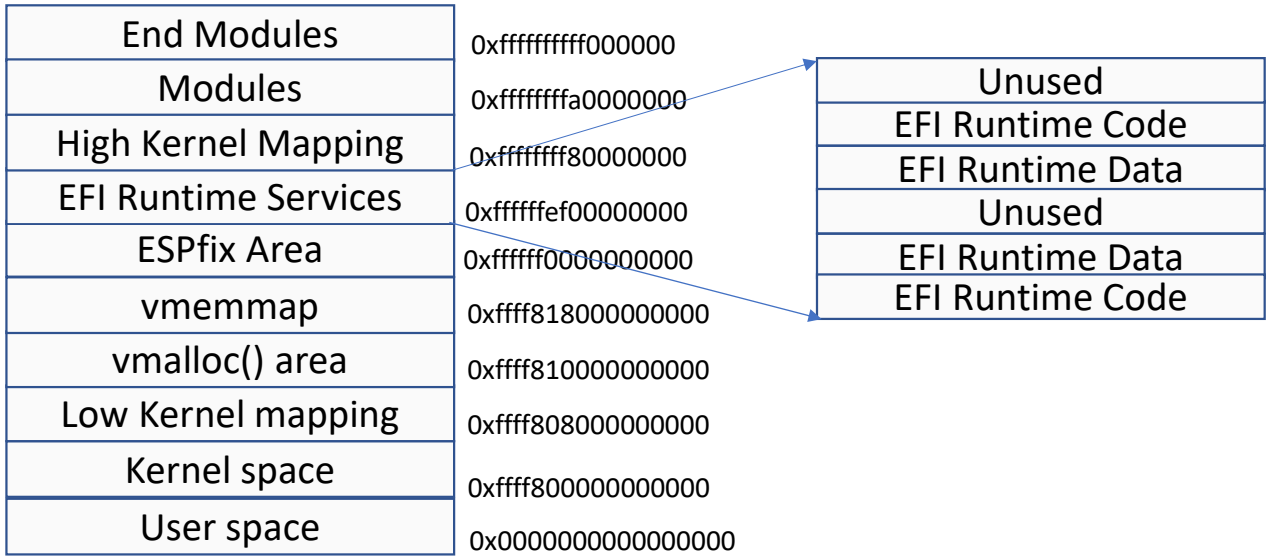


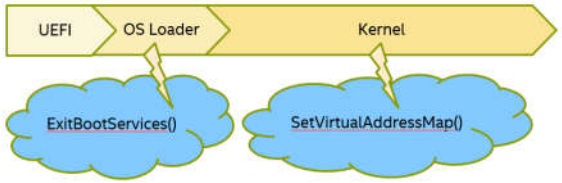
- Kernel controls system memory
- Physical addressing mode
- Before calling SetVirtualAddressMap()



Boot Process – Virtual Address space

Kernel Virtual Address space



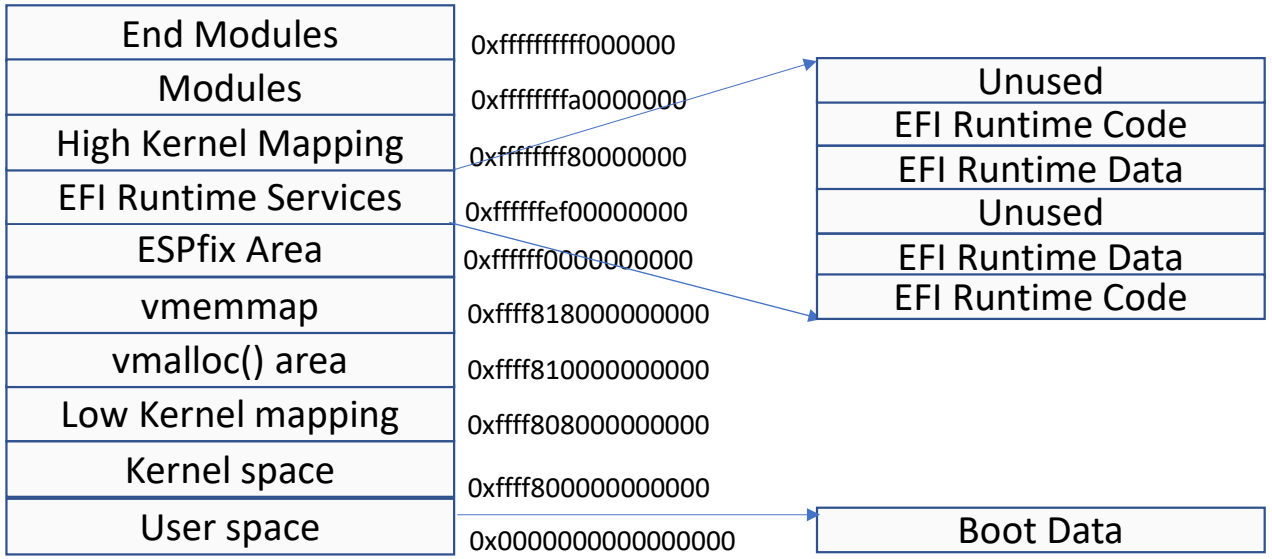


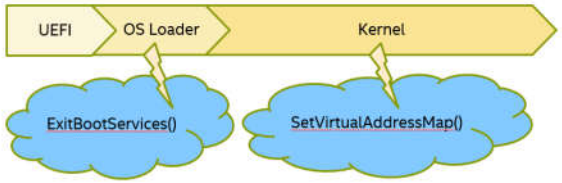
Memory as seen by firmware

Boot Code	0x7ff70000
Conventional Memory	0x7ed74000
ACPI Memory NVS	0x7e48d000
Boot Data	0x7e3ba000
Loader Data	0x5c920000
Conventional Memory	0x0544e000
Loader Code	0x01300000
Reserved	0x00800000
Runtime Data	0x00100000
Runtime Code	0x00001000
Boot Code	0x00000000

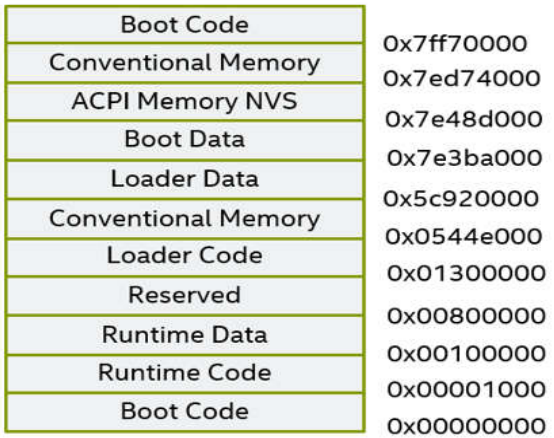
Illegal accesses by firmware

Kernel Virtual Address space





Memory as seen by firmware



Illegal access by EFI Runtime Service

```
[  244.699880] Call Trace:
[  244.702975]  [<ffffffff810abf01>] ? __lock_acquire+0x121/0x1480
[  244.709553]  [<ffffffff810ac57f>] ? __lock_acquire+0x79f/0x1480
[  244.716104]  [<ffffffff810599a8>] ? efi_call+0x58/0x90
[  244.721894]  [<ffffffff81748e4b>] ? virt_efi_set_variable+0x7b/0x190
[  244.728905]  [<ffffffffa0065e63>] ? efi_runtime_ioctl+0xe43/0x108c [efi_runtime]
[  244.736964]  [<ffffffff810893b4>] ?  might_sleep+0x44/0x80
```

EFI Runtime Service that caused illegal access

```
244.524426] BUG: unable to handle kernel paging request at 0000000018847980
244.531896] IP: [<ffffffffef6981ec3>] 0xffffffffef6981ec3
244.537625] PGD 4324d3063 PUD 4324e4063 PMD 0
244.542634] Oops: 0000 [#1] SMP
244.546267] Modules linked in: efi_runtime(O) chipsec(O) igb e1000e [last unloaded: efi_runtime]
244.555695] CPU: 5 PID: 2308 Comm: fwts Tainted: G        W  O    4.7.0-yocto-standard #1
244.564379] Hardware name: Gigabyte Technology Co., Ltd. Z87X-UD5H/Z87X-UD5H-CF, BIOS F9 03/18/2014
```

Illegally accessed address

```
[    0.038627] ---[ User Space ]---
[    0.041860] 0x0000000000000000-0x0000000000008000          32K     RW                GLB x  pte
[    0.050849] 0x0000000000008000-0x0000000000060000         352K                              pte
[    0.059697] 0x0000000000060000-0x00000000000a0000         256K     RW                GLB x  pte
[    0.068685] 0x00000000000a0000-0x0000000000200000        1408K                              pte
[    0.077535] 0x0000000000200000-0x0000000040000000        1022M                              pmd
[    0.086388] 0x0000000040000000-0x0000000080000000           1G                              pud
[    0.095246] 0x0000000080000000-0x00000000a6600000         614M                              pmd
```

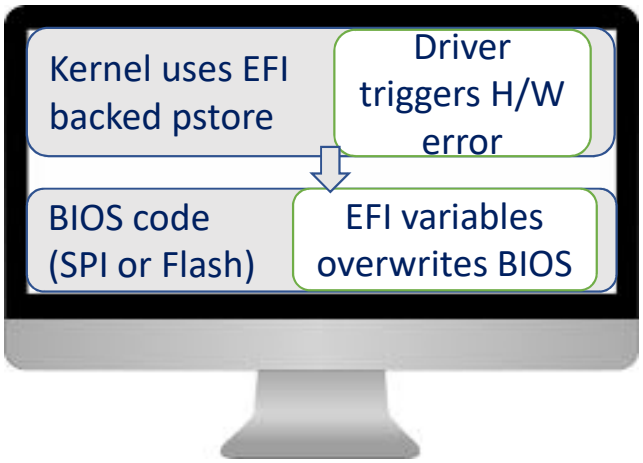
Unmapped region

```
[    0.000000] efi: mem07: [Loader Data        |   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x0000000000100000-0x0000000002115fff] (32MB)
[    0.000000] efi: mem08: [Conventional Memory|   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x0000000002116000-0x00000000021fffff] (0MB)
[    0.000000] efi: mem09: [Loader Data        |   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x0000000002200000-0x0000000004215fff] (32MB)
[    0.000000] efi: mem10: [Conventional Memory|   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x0000000004216000-0x00000000331b5fff] (751MB)
[    0.000000] efi: mem11: [Loader Data        |   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x00000000331b6000-0x00000000358d2fff] (39MB)
[    0.000000] efi: mem12: [Conventional Memory|   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x00000000358d3000-0x00000000784d7fff] (1068MB)
[    0.000000] efi: mem13: [Loader Data        |   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x00000000784d8000-0x00000000a2dcbfff] (680MB)
[    0.000000] efi: mem14: [Conventional Memory|   |  |  |  |  |  |  |   |WB|WT|WC|UC] range=[0x00000000a2dcc000-0x00000000a538bfff] (37MB)
```

EFI Runtime Service referencing EFI Conventional Memory

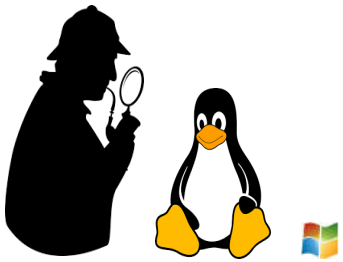
Samsung laptops bricked by UEFI

PC enthusiast reports “Booting Linux bricks Samsung laptops”



Mathew Garett founds astonishing facts

Kernel hacker starts investigation



Buggy firmware

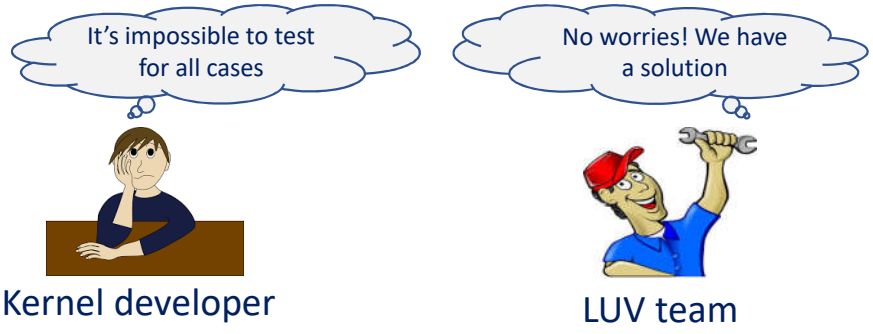
Kernel developer

Reference: https://mjg59.dreamwidth.org/22855.html

Kernel bugs causing panic

Kernel panics on BGRT enabled platform

```
[     0.251599] Last level dTLB entries: 4KB 64, 2MB 0, 4MB 0, 1GB 4
[     0.259126] Freeing SMP alternatives memory: 32K (ffffffff8230e000 - ffffffff8231
[     0.271803] BUG: unable to handle kernel paging request at fffffffefce35002
[     0.279740] IP: [<ffffffff821bca49>] efi_bgrt_init+0x144/0x1fd
[     0.286383] PGD 300f067 PUD 0
[     0.289879] Oops: 0000 [#1] SMP
[     0.293566] Modules linked in:
[     0.417252] DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
[     0.425350] Stack:
[     0.427638]  ffffffffffffffff ffffffff82256900 ffff88016dbaea40 ffffffff82003f40
[     0.436086]  ffffffff821bbce0 ffffffff82003f88 ffffffff8219c0c2 0000000000000000
[     0.444533]  ffffffff8219ba4a ffffffff822622c0 0000000000083000 00000000ffffffff
[     0.452978] Call Trace:
[     0.455763]  [<ffffffff821bbce0>] efi_late_init+0x9/0xb
[     0.461697]  [<ffffffff8219c0c2>] start_kernel+0x463/0x47f
[     0.467928]  [<ffffffff8219ba4a>] ? set_init_arg+0x55/0x55
[     0.474159]  [<ffffffff8219b120>] ? early_idt_handler_array+0x120/0x120
[     0.481669]  [<ffffffff8219b5ee>] x86_64_start_reservations+0x2a/0x2c
[     0.488982]  [<ffffffff8219b72d>] x86_64_start_kernel+0x13d/0x14c
[     0.495897] Code: 00 41 b4 01 48 8b 78 28 e8 09 36 01 00 48 85 c0 48 89 c3 75 13
[     0.518151] RIP  [<ffffffff821bca49>] efi_bgrt_init+0x144/0x1fd
[     0.524888]  RSP <ffffffff82003f18>
[     0.528851] CR2: fffffffefce35002
[     0.532615] ---[ end trace 7b06521e6ebf2aea ]---
[     0.537852] Kernel panic - not syncing: Attempted to kill the idle task!
```



Kernel developer

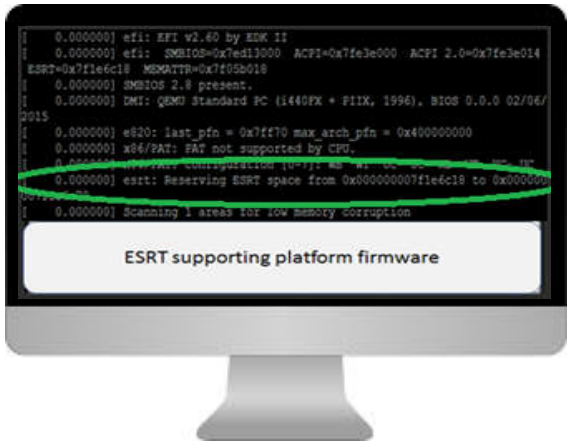
LUV team

Enable all possible tables/features in OVMF and use Qemu + OVMF to find bugs

Reference: https://lkml.org/lkml/2015/12/10/599
BGRT - Boot Graphics Resource Table
OVMF - Opensource Virtual Machine Firmware

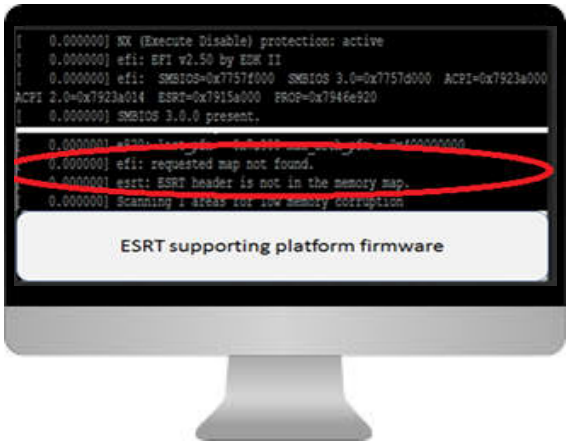
Buggy firmware causes issues

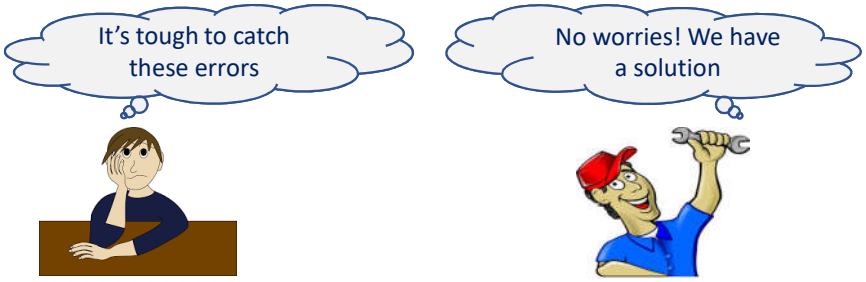
ESRT parsed successfully



Same kernel booted on two different machines

Failed to parse ESRT





QA Engineer

LUV team

Enable all possible tables/features in OVMF and use Qemu + OVMF to find bugs

ESRT - EFI System Resource Table

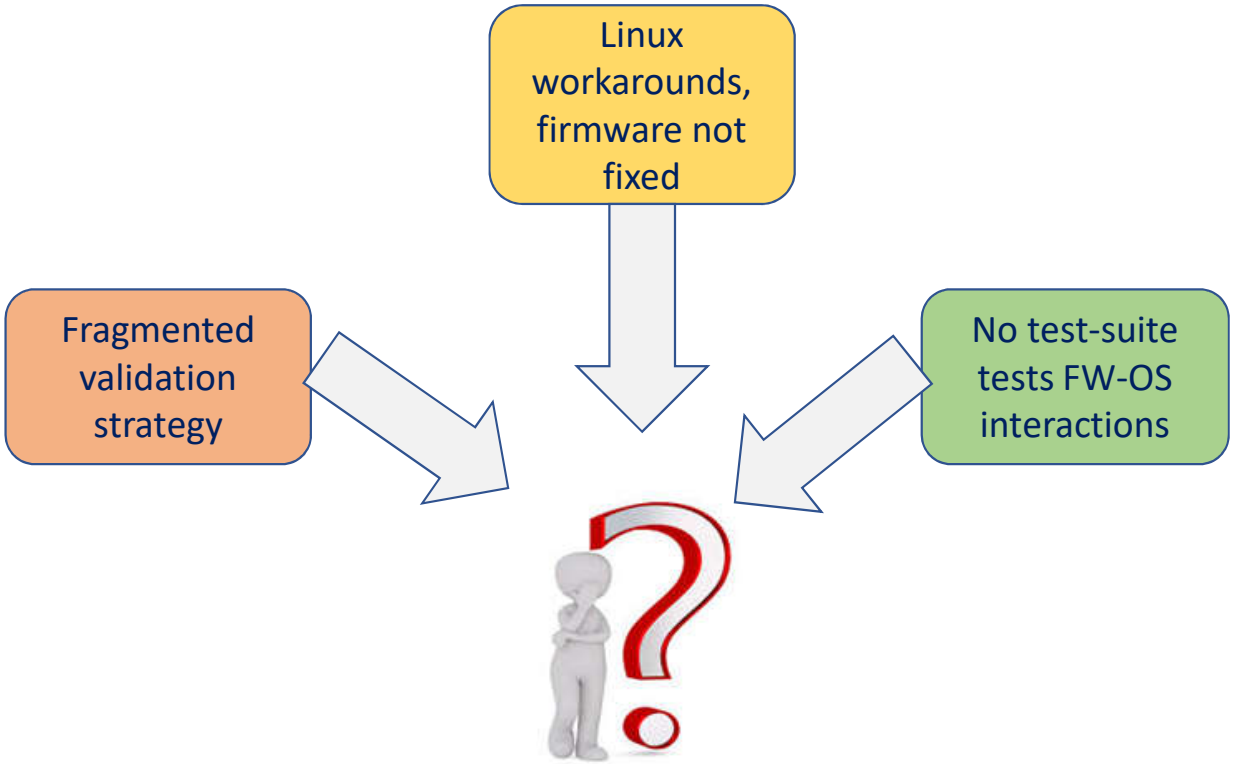
UEFI implementation is critical

Things can go wrong during interactions between firmware and Linux kernel

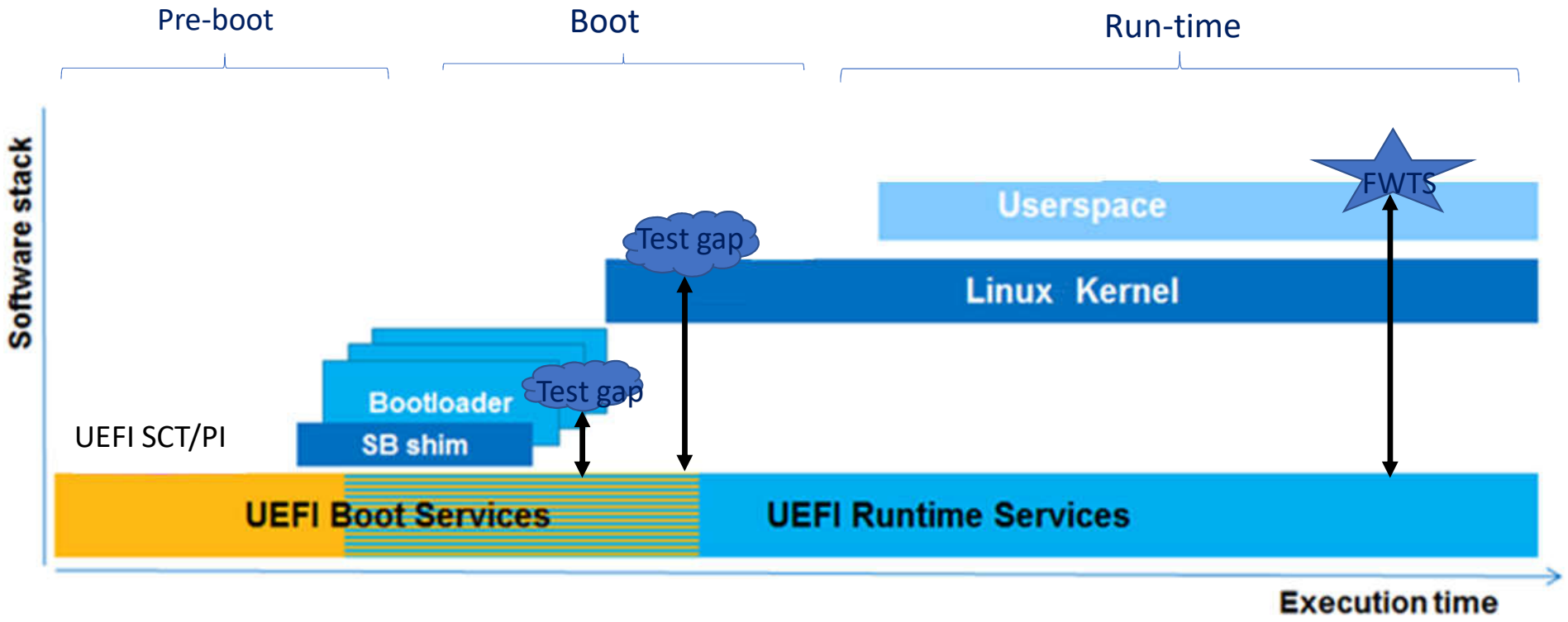
Faulty implementation of generic UEFI hurts Linux

Linux UEFI Validation (LUV) project was started

Why LUV?


Linux workarounds, firmware not fixed
Fragmented validation strategy
No test-suite tests FW-OS interactions

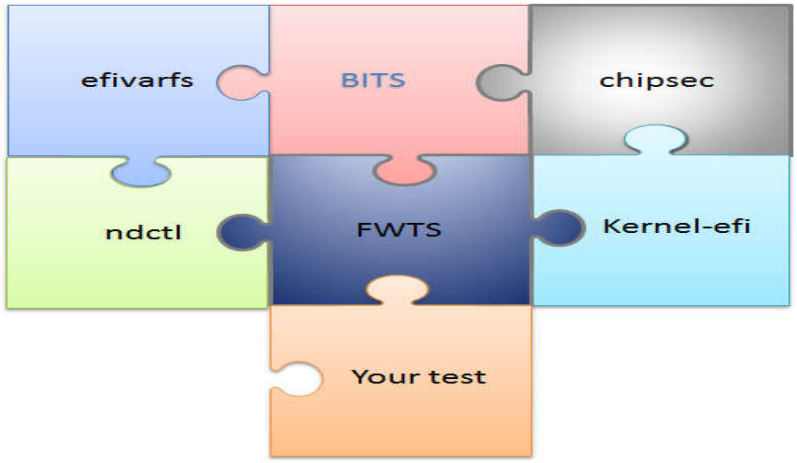
Current Validation Strategy



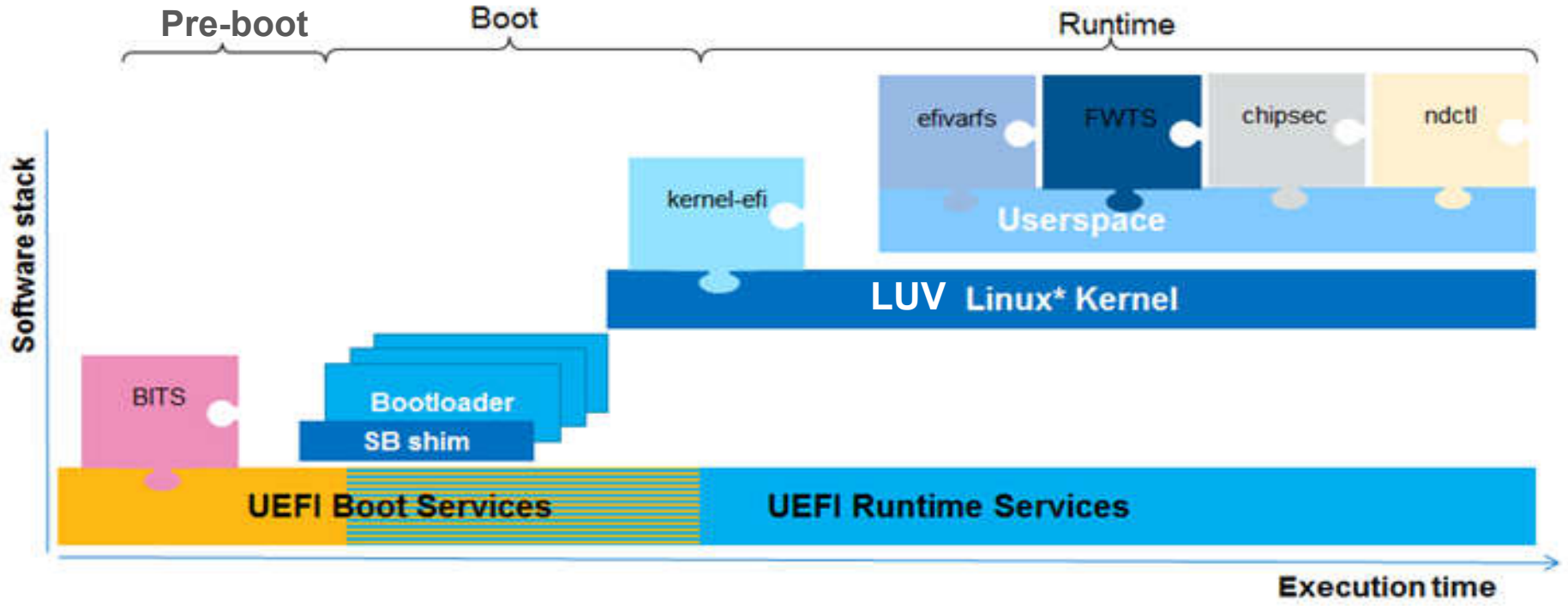
LUV: A unified framework

LUV is a complete Linux distro

- Continuous Integration framework of upstream open source projects:
  - Linux kernel
  - Yocto Project
  - Open source test-suites (eg. FWTS)



LUV: Covering the entire spectrum



LUV: Detect bugs early

At boot time:

```
Linux UEFI Validation Distribution 2.1-rc2

Date and time of the system : 2017-03-28--13-15-48

System: Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz x 8

Test results summary:

[+]bits
[+]chipsec
[+]efivarfs
[+]fwts
[-]kernel-efi-warnings
        [-]EFI_illegal_accesses_test... 1 failures!
                Test: EFI_illegal_accesses_test
                Description: Check for illegal accesses to EFI memory regions.
                FAIL: Found 1 instances of "efi: \[Firmware Bug\]: Fixing illegal access to EFI region at PA: ".
                Related kernel information
                ---------------------------------------------------
                [ 0.012085] pid_max: default: 32768 minimum: 301
                [ 0.017353] ACPI: Core revision 20160930
                [ 0.060109] ACPI: 17 ACPI AML tables successfully acquired and loaded
                [ 0.067511] efi: [Firmware Bug]: Fixing illegal access to EFI region at PA: 0xb82ffc98
                [ 0.076656] ---[ User Space ]---
                [ 0.080243] 0x0000000000000000-0x0000000000001000 4K RW GLB x pte
                [ 0.090241] 0x0000000000001000-0x0000000000200000 2044K pte
                ---------------------------------------------------
        [+]Kernel_FW_BUG... 1 failures!
        [+]Kernel_FW_WARN... passed
        [+]Kernel_FW_INFO... passed
[+]ndctl

Ran 6 testsuites and 134 unittests, 72 passes, 268 fails, 44 skipped, 6 warnings.
```

Kernel fixing illegal access by firmware during Set Virtual Address Map

At run-time:

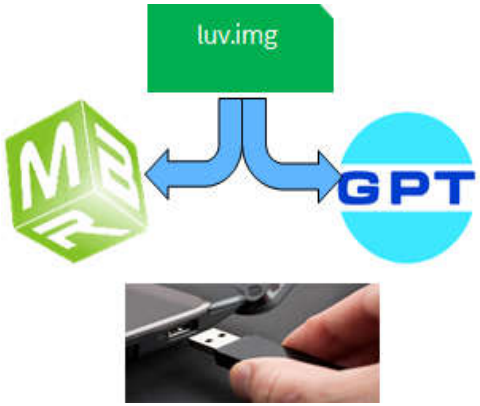
```
[  238.856558] [-] fwts
[  238.859017]   [+] version... skipped
[  238.863030]   [+] bios_info... passed
[  238.867136]   [+] prd_info... skipped
[  238.871245]   [+] oops... passed
[  238.874859]   [+] olog... skipped
[  238.974803]   [+] klog... 1 failures!
[  238.978925]   [+] bmc_info... skipped
[  239.031610]   [+] mtrr... 3 failures!
[  239.036934]   [+] acpiinfo... passed
[  239.044789] efi: [Firmware Bug]: Fixing illegal access to EFI region at PA: 0xaccc2520
[  239.053614] efi: [Firmware Bug]: Fixing illegal access to EFI region at PA: 0xad7a5cd6
```

Kernel fixing illegal access by firmware when EFI Runtime Service is invoked

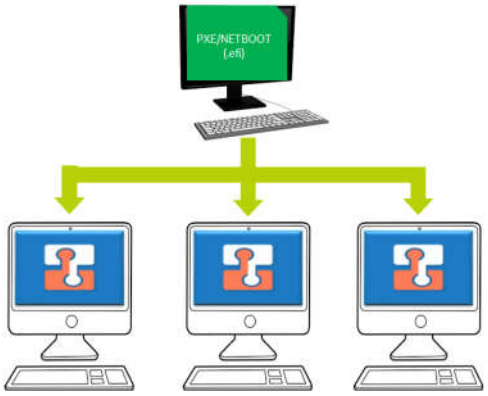
LUV uses hardened kernel to work around these issues (during boot time and run time)

LUV: Easy to use

Plug and play



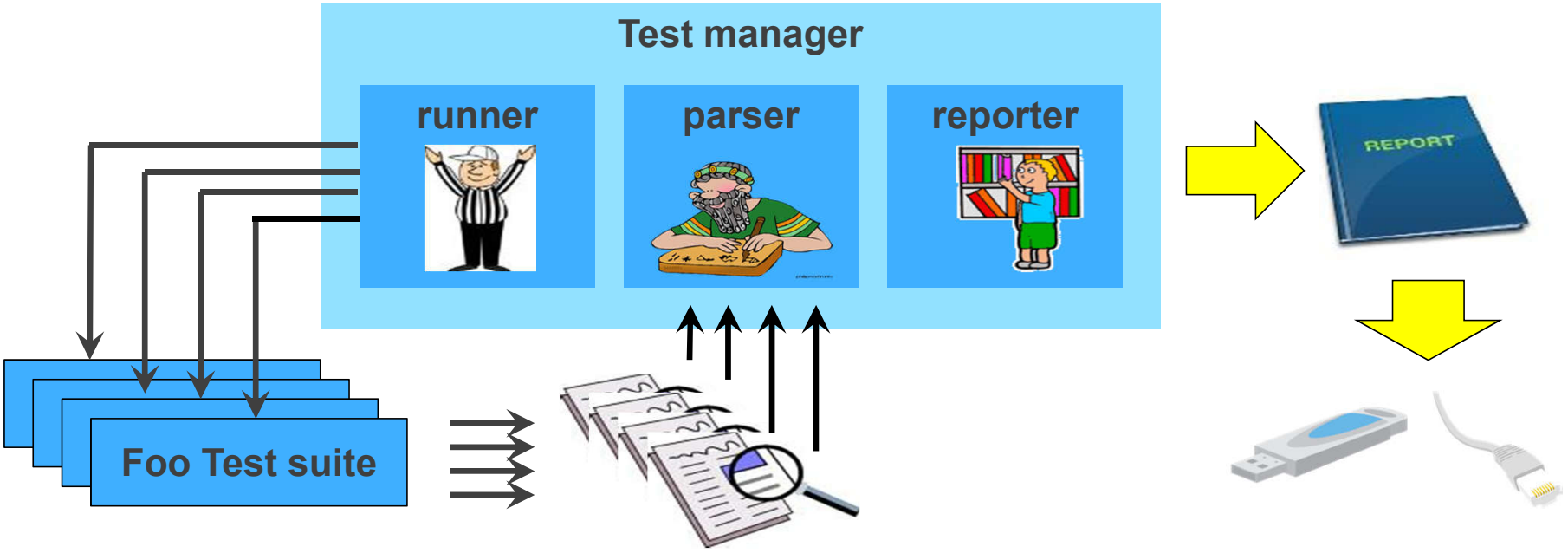
Diskless boot



Boot on Virtual Machine

LUV: Easy to contribute

LUV is open source!



LUV: Available features

- Results are saved for viewing
  1. If (USB Stick) - Saved in LUV_RESULTS folder
  2. If (Net Boot) - Saved to HTTP server
  3. If (VM) - Saved in the results partition of luv.img
- Debug via network using Netconsole.
- Telemetrics mechanism to send error report to the LUV server incase of crash.
- Lists all available DSMs on a platform.
- *Future feature: Add support for more bootloaders.

```
Linux UEFI Validation Distribution 2.2-dev Date and time of the system : 2018-02-15--21-39-35
System: Intel(R) Core(TM) i7-4770HQ CPU @ 2.20GHz x 8 Test results summary:
[-]chipsec
        [+]chipsec.modules.common.bios_ts... passed
        [+]chipsec.modules.common.spi_access... 2 warnings
        [+]chipsec.modules.common.ia32cfg... passed
        [+]chipsec.modules.common.spi_lock... passed
        [+]chipsec.modules.common.smm... passed
        [+]chipsec.modules.common.bios_kbrd_buffer... passed
        [+]chipsec.modules.common.bios_smi... passed
        [+]chipsec.modules.common.smrr... passed
        [+]chipsec.modules.common.spi_fdopss... passed
        [+]chipsec.modules.common.spi_desc... passed
        [+]chipsec.modules.common.rtclock... 1 warnings
        [+]❗❗❗chipsec.modules.common.bios_wp... 1 failures!
        [+]chipsec.modules.common.secureboot.variables... 1 warnings
        [+]❗❗❗chipsec.modules.common.uefi.s3bootscript... 1 failures!
        [+]chipsec.modules.common.uefi.access_uefispec... passed
        [+]chipsec.modules.memconfig... passed
        [+]chipsec.modules.remap... passed
        [+]chipsec.modules.smm_dma... passed
[+]efivarfs
[+]fwts
[+]kernel-efi-warnings
[+]ndctl
[+]pstore-tests

Ran 6 testsuites and 159 unittests, 80 passes, 117 fails, 59 skipped, 4 warnings.
LEGEND:
Severity of failures: Low: ❗ Medium: ❗❗ High: ❗❗❗ Unknown: ❓
```

LUV: Path to better firmware

Collaborate

Consolidates various open source test-suites

Comprehensive

Test the entire software stack

Quality

Detect firmware/kernel bugs early

Community

Easy to use

Easy to contribute

What do you need?

Tell us your needs to better validate your firmware!

Join US

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

Credits

Neri Ricardo
Matt Fleming
Gayatri Kammela
Naresh Bhat
And many more contributors…

Thank you.

Any Questions?