

BITS and CHIPSEC as coreboot payloads

OSFC 2018

Piotr Król and Michał Żygowski







- Introduction
- Motivation
- BITS features
- CHIPSEC features
- Enabling BITS as payload
- Enabling CHIPSEC as payload
- BITS and CHIPSEC in action (demo)
- Summary



Piotr Król



*Founder & Embedded Systems
Consultant*

-  @pietrushnic
-  piotr.krol@3mdeb.com
-  [linkedin.com/in/krolpiotr](https://www.linkedin.com/in/krolpiotr)
-  [facebook.com/piotr.krol.756859](https://www.facebook.com/piotr.krol.756859)



Michał Żygowski

Firmware Engineer

-  michal.zygowski@3mdeb.com
-  [linkedin.com/in/michał-żygowski-88954416b](https://www.linkedin.com/in/michał-żygowski-88954416b)

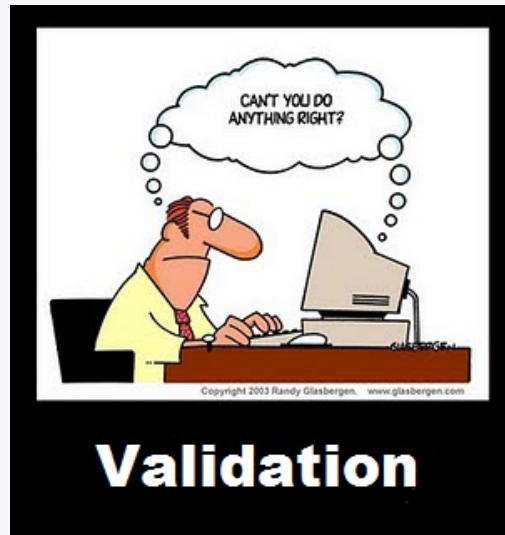
How we validate quality of our firmware?

- Not enough validation in open source firmware
- Firmware security is about validation and formal development process
- BITS and CHIPSEC are recognized frameworks for quality checks
- Linux UEFI Validation (LUV), what about coreboot?
- Certification issues

Why we should avoid running tests in OS?

- OS is external firmware customer
- firmware is treated as part of hardware and should work out of the box
- OS may introduce another point of failures

In following presentation we would like to present our achievements while using BITS and CHIPSEC as validation payloads for MinnowBoard Turbot.

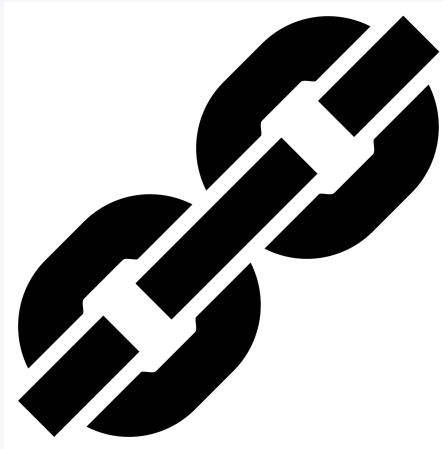


<http://blog.gatestlab.com/wp-content/uploads/2011/02/validation.jpg>

- system tables
 - ACPI
 - SMBIOS
 - MultiProcessor tables
 - \$PIR (PCI Interrupt Routing) table
 - Runtime and Boot Services (Tianocore payload)
 - any other structure that firmware present to OS
- hardware configuration
 - SPI protection
 - SMRAM protection
- other
 - spectre mitigation presence
 - vendor specific features (USB DCI, ME, PSP)

- BITS (BIOS Implementation Test Suite) consist of a GRUB2 bootloader extended with runtime Python support.
- Consist of Python scripts that validate:
 - ACPI
 - SMRR configuration
 - SMI latency
 - MP Table
 - MSRs
- Typically it is run using bootable USB created using BITS ISO image
- Can be run through GRUB menu entries or using batch mode
- Results can be read directly on screen or saved to filesystem
- Extensibility: Python interpreter in GRUB2

- BITS is quite big in size 45MB - not suitable for SPI flash
 - was solved by utilizing only core (GRUB2+Python)
- Environment is not user friendly - each modification requires SPI reflashing
 - network boot or using USB for development can solve that
 - live Python usage
- Build system has real problems since it depends on obsolete libraries
 - we used Docker container for compilation



<http://clipground.com/images/constraint-clipart-1.jpg>

- checks SPI and firmware protection
- verifies integrity of flash descriptor
- tests SMM, SMIs and SMM cache poisoning
- checks MSRs, SMRRs and memory configuration
- other various tools:
 - NMI sending
 - IOMMU check
 - TPM, EC utility
 - CMOS, PCI, SPD utility
 - etc.



- utilize GRUB from coreboot - already suitable for SPI flash
- re-add couple functions required by BITS
 - grub_strcat
 - disable support for software floating point arithmetics using compiler flags
 - small fixes to printf arguments parsing
- port Python support for GRUB
- adjust build system
- hack BITS to correctly handle paths in SPI flash
 - isdir hack
- enable serial output in toplevel config
- LZMA compression

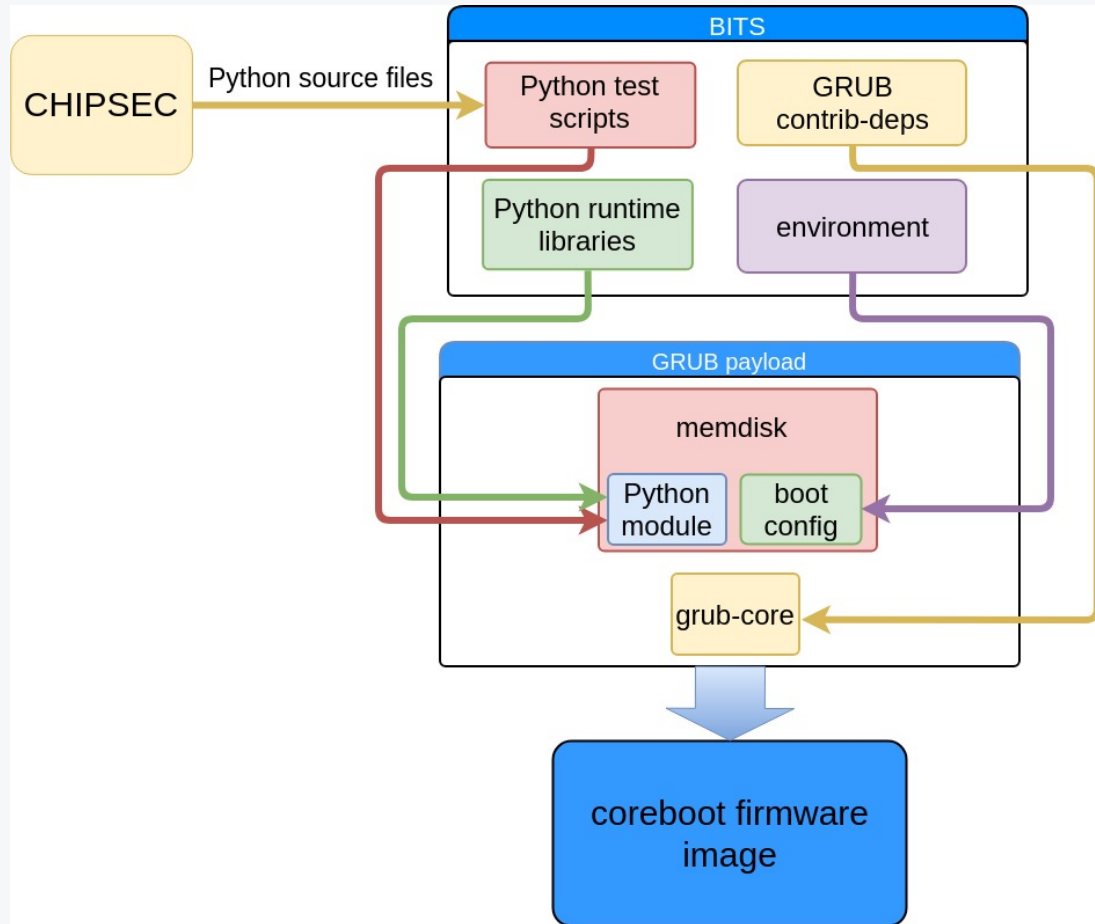
- add Python source code to BITS
- resolve dependency from Python standard library
 - BITS has different needs than CHIPSEC
 - add missing Python standard libraries (xml, JSON, subprocess, distutils...)
- Implement BITS OS helper
- Use some methods from BITS as backend for CHIPSEC calls



- Initial size of USB image - **45MB**
- Cutting off not needed pieces: **12.6MB**
 - UEFI support
- LZMA compression: **3.6MB**

FMAP REGION: COREBOOT

Name	Offset	Type	Size	Comp	
cbfs master header	0x0	cbfs header	32	none	
fallback/romstage	0x80	stage	31372	none	
cpu_microcode_blob.bin	0x7b80	microcode	104448	none	
fallback/ramstage	0x21400	stage	61533	none	
config	0x304c0	raw	669	none	
revision	0x307c0	raw	582	none	
cmos_layout.bin	0x30a40	cmos_layout	1208	none	
fallback/dsdt.aml	0x30f40	raw	12528	none	
fallback/payload	0x34080	simple elf	3643338	none	<--- HERE
(empty)	0x3ad8c0	null	74968	none	
fsp.bin	0x3bfdc0	fsp	229376	none	
(empty)	0x3f7e00	null	30936	none	
bootblock	0x3ff700	bootblock	1720	none	



BITS and CHIPSEC demo

```
[CHIPSEC] ***** SUMMARY *****
[CHIPSEC] Time elapsed          2.762
[CHIPSEC] Modules total        17
[CHIPSEC] Modules failed to run 0:
[CHIPSEC] Modules passed       4:
[+] PASSED: chipsec.modules.common.spi_fdopss
[+] PASSED: chipsec.modules.common.bios_ts
[+] PASSED: chipsec.modules.common.bios_kbrd_buffer
[+] PASSED: chipsec.modules.common.smrr
[CHIPSEC] Modules information  0:
[CHIPSEC] Modules failed      5:
[-] FAILED: chipsec.modules.common.memlock
[-] FAILED: chipsec.modules.common.bios_wp
[-] FAILED: chipsec.modules.common.spi_access
[-] FAILED: chipsec.modules.common.spi_desc
[-] FAILED: chipsec.modules.common.spi_lock
[CHIPSEC] Modules with warnings 0:
[CHIPSEC] Modules not implemented 8:
[*] NOT IMPLEMENTED: chipsec.modules.common.ia32cfg
[*] NOT IMPLEMENTED: chipsec.modules.common.bios_smi
[*] NOT IMPLEMENTED: chipsec.modules.common.smm
[*] NOT IMPLEMENTED: chipsec.modules.common.rtclock
[*] NOT IMPLEMENTED: chipsec.modules.memconfig
[*] NOT IMPLEMENTED: chipsec.modules.remap
[*] NOT IMPLEMENTED: chipsec.modules.smm_dma
[*] NOT IMPLEMENTED: chipsec.modules.debugenabled
[CHIPSEC] Modules not applicable 0:
[CHIPSEC] *****
```


- Bare metal (Micro)Python support is important for firmware validation
- There is not enough validation in firmware
- We should utilize existing tools that were proved in industry

Further steps

- Mainlining process
- Consider BITS and CHIPSEC port to MicroPython
- Fix platform bugs and misconfiguration

Q&A