Monthly Research 2016.7
About security assessment framework “CHIPSEC”

FFRI,Inc.
http://www.ffri.jp

E-Mail: research-feedback[at]ffri.jp
Twitter: @FFRI_Research

Outline

- About CHIPSEC
- Inspection menu
- How to install
- Usage
- Check of inspection result
- Data analysis
- Conclusion
- References

About CHIPSEC

- A hardware security assessment tool developed by Intel
  - It inspects BIOS/UEFI configurations and data read/write
  - The inspection result is “PASSED” or “FAILED”

  - It includes some utility scripts
    - Dump/Restore CMOS memory
    - Dump PCI interface information

  - Execution environments are Windows, Linux and UEFI Shell
  - It is written in Python and it has been developed on GitHub
  - License is GPL v2

Inspection menu

- SMRAM Locking/SPI Controller Locking/BIOS Interface Locking
  - Checking lock of controller settings
  - There are risks of brick or persistent malware if unlocked setting was modified
- BIOS Keyboard Buffer Sanitization
  - Checking keyboard buffer
  - There is a risk of password leak if data remain on keyboard buffer
- SMRR Configuration
  - Checking protection for the SMRR(System Management Range Register)
  - There is a risk of rootkit infection if it has problem with this configuration

Inspection menu

- BIOS Protection
  - Checking BIOS settings
  - There is a risk of brick if the settings are rewritten by malware

- Access Control for Secure Boot Keys/Variables
  - Checking Secure Boot settings
  - There is a risk of secure boot bypass if this settings have problems

How to install

1. Install Python
2. Install of python modules
   - pwin32
   - Wconio
   - py2exe
3. Disable Windows driver signing check
   - bcdedit /set TESTSIGNING ON
   - reboot
4. Install Driver
   - sc create chipsec binpath= <PATH_TO_CHIPSEC_SYS> type= kernel DisplayName= "Chipsec driver
   - sc start chipsec

For more information refer to the manual of CHIPSEC

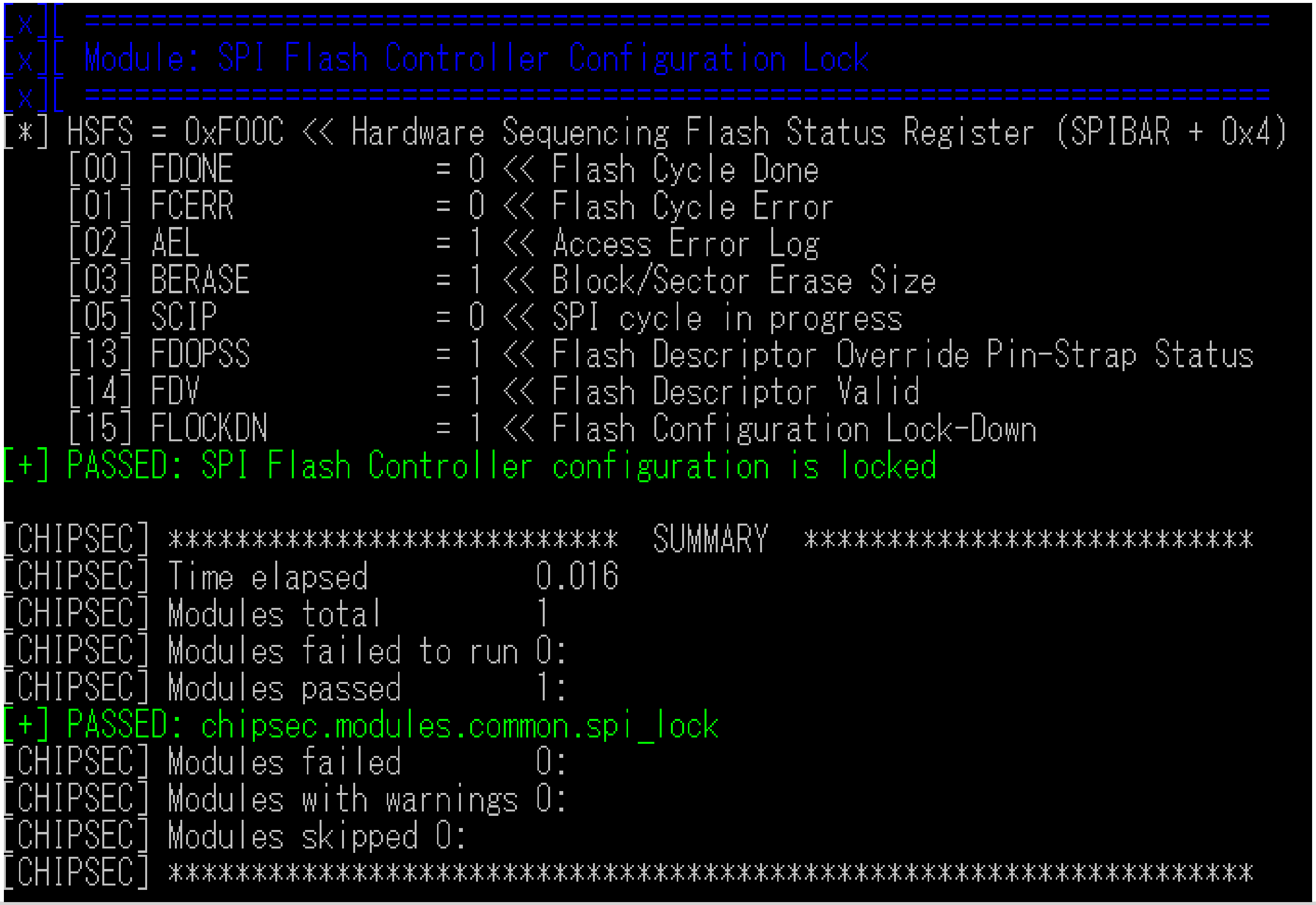
Usage

- Inspection (chipsec_main.py)
  - BIOS lock check
    - python chipsec_main.py -m common.bios_wp
  - SPI Memory lock check
    - python chipsec_main.py –m common.spi_lock etc...
  - Summary is displayed when the check is completed
    - Result is “PASSED” or “FAILED”

- Utility (chipsec_util.py)
  - SPI Memory Dump
    - python chipsec_util.py spi dump
  - PCI ROM Dump
    - python chipsec_util.py pci dump

Inspection result

- An example of the results is shown below

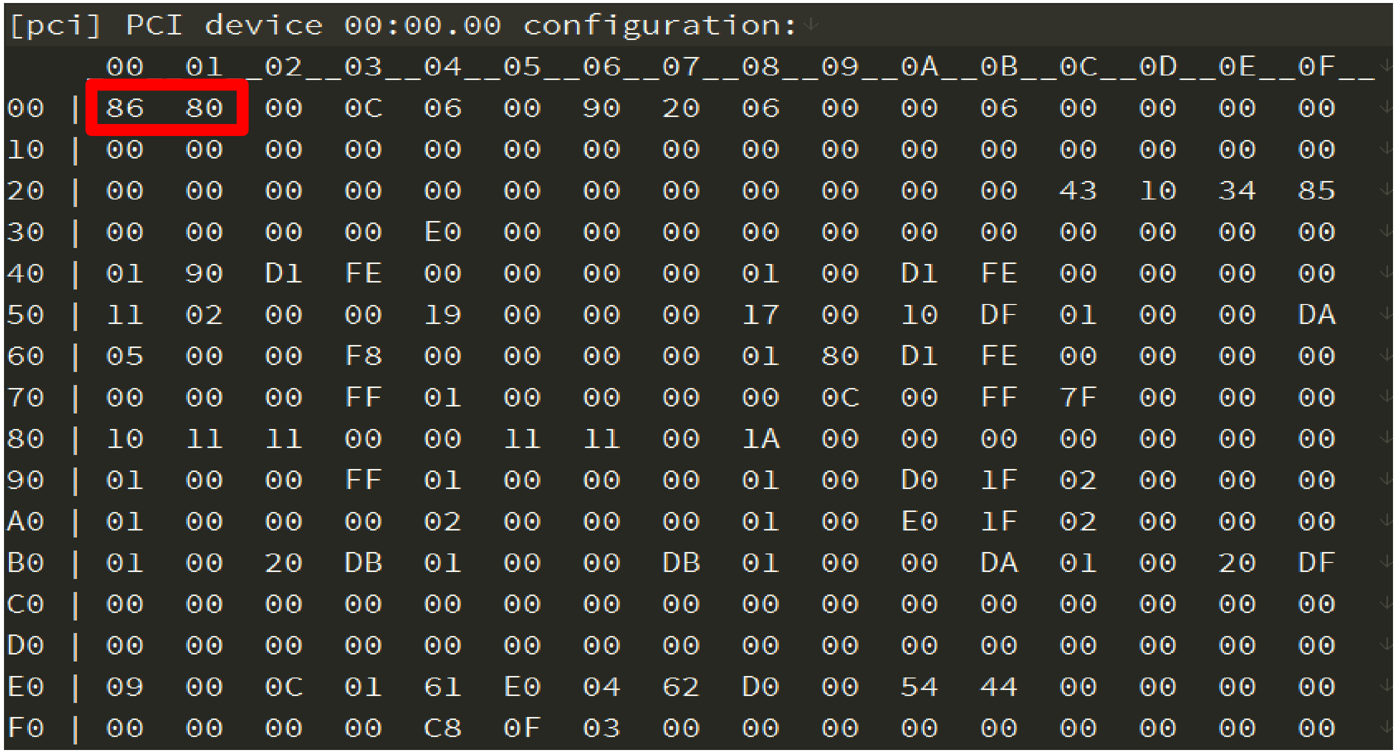


```
[x][ =======================================================================
[x][ Module: SPI Flash Controller Configuration Lock
[x][ =======================================================================
[*] HSFS = 0xF00C << Hardware Sequencing Flash Status Register (SPIBAR + 0x4)
    [00] FDONE            = 0 << Flash Cycle Done
    [01] FCERR            = 0 << Flash Cycle Error
    [02] AEL              = 1 << Access Error Log
    [03] BERASE           = 1 << Block/Sector Erase Size
    [05] SCIP             = 0 << SPI cycle in progress
    [13] FDOPSS           = 1 << Flash Descriptor Override Pin-Strap Status
    [14] FDV              = 1 << Flash Descriptor Valid
    [15] FLOCKDN          = 1 << Flash Configuration Lock-Down
[+] PASSED: SPI Flash Controller configuration is locked

[CHIPSEC] ***************************  SUMMARY  ***************************
[CHIPSEC] Time elapsed          0.016
[CHIPSEC] Modules total         1
[CHIPSEC] Modules failed to run 0:
[CHIPSEC] Modules passed        1:
[+] PASSED: chipsec.modules.common.spi_lock
[CHIPSEC] Modules failed        0:
[CHIPSEC] Modules with warnings 0:
[CHIPSEC] Modules skipped 0:
[CHIPSEC] *****************************************************************
```

Data analysis (PCI ROM)

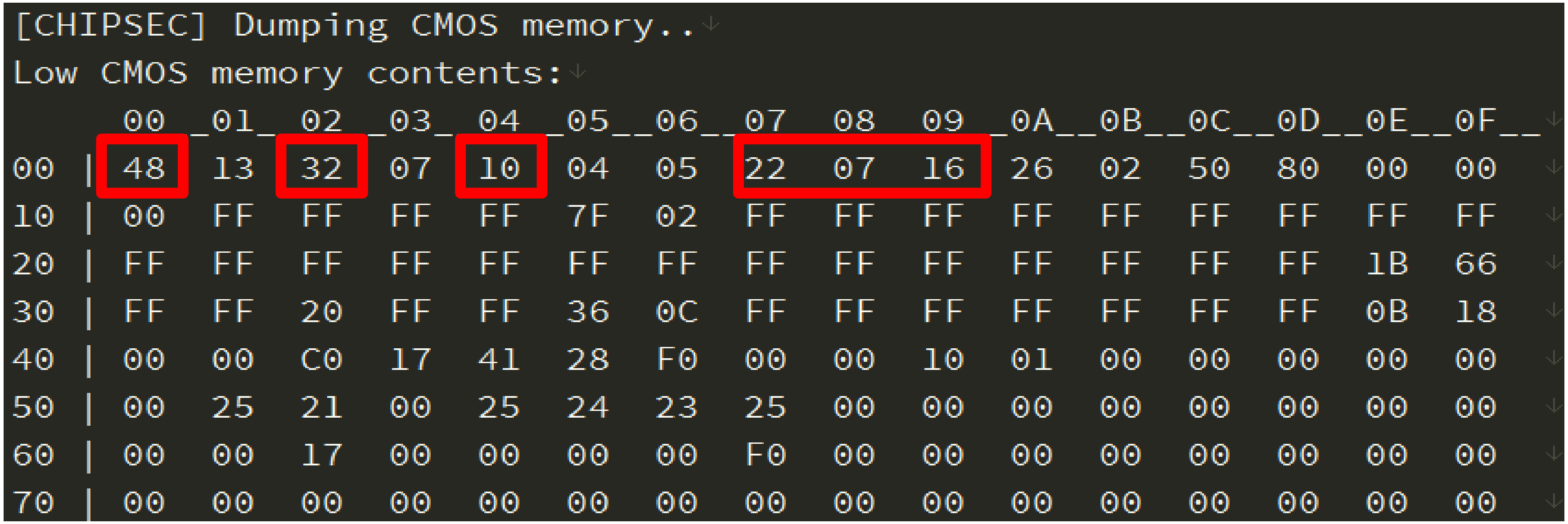
- PCI ROM dump by chipsec_util.py
  - Obtaining information of each PCI devices which are connected
  - e.g. 2byte from the top vendor ID(Little endian) 8086 is Intel



```
[pci] PCI device 00:00.00 configuration:
      00   01  _02__03__04__05__06__07__08__09__0A__0B__0C__0D__0E__0F__
00 |  86   80   00   0C   06   00   90   20   06   00   00   06   00   00   00   00
10 |  00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
20 |  00   00   00   00   00   00   00   00   00   00   00   00   43   10   34   85
30 |  00   00   00   00   E0   00   00   00   00   00   00   00   00   00   00   00
40 |  01   90   D1   FE   00   00   00   00   01   00   D1   FE   00   00   00   00
50 |  11   02   00   00   19   00   00   00   17   00   10   DF   01   00   00   DA
60 |  05   00   00   F8   00   00   00   00   01   80   D1   FE   00   00   00   00
70 |  00   00   00   FF   01   00   00   00   00   0C   00   FF   7F   00   00   00
80 |  10   11   11   00   00   11   11   00   1A   00   00   00   00   00   00   00
90 |  01   00   00   FF   01   00   00   00   01   00   D0   1F   02   00   00   00
A0 |  01   00   00   00   02   00   00   00   01   00   E0   1F   02   00   00   00
B0 |  01   00   20   DB   01   00   00   DB   01   00   00   DA   01   00   20   DF
C0 |  00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
D0 |  00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
E0 |  09   00   0C   01   61   E0   04   62   D0   00   54   44   00   00   00   00
F0 |  00   00   00   00   C8   0F   03   00   00   00   00   00   00   00   00   00
```

Data analysis (CMOS Memory)

- CMOS memory contains the BIOS settings
  - Data sequence is defined in Memory map
  - Red frame represents the date and time(2016/07/22 10:32:48)



```
[CHIPSEC] Dumping CMOS memory..
Low CMOS memory contents:
     00 _01_ 02 _03_ 04 _05__06__07  08  09 _0A__0B__0C__0D__0E__0F__
00 | 48  13  32  07  10  04  05  22  07  16  26  02  50  80  00  00
10 | 00  FF  FF  FF  FF  7F  02  FF  FF  FF  FF  FF  FF  FF  FF  FF
20 | FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  1B  66
30 | FF  FF  20  FF  FF  36  0C  FF  FF  FF  FF  FF  FF  FF  0B  18
40 | 00  00  C0  17  41  28  F0  00  00  10  01  00  00  00  00  00
50 | 00  25  21  00  25  24  23  25  00  00  00  00  00  00  00  00
60 | 00  00  17  00  00  00  00  F0  00  00  00  00  00  00  00  00
70 | 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
```

Conclusion

- Vulnerable BIOS/UEFI configuration can become target of cyber attack
  - The following threats are concerned
    - Brick
    - Persistent malware/rootkit infection
    - Leak of password from BIOS keyboard buffer
    - Bypass of Secure boot

- CHIPSEC is a useful tool for BIOS/UEFI security checking
  - Various inspection modules and simple command
  - Possible to add original inspection modules
  - Possible to integrate to the other tool
  - Possible to dump various data with utility scripts

References

- CHPSEC's GitHub page
  - https://github.com/chipsec/chipsec
- CMOS Memory Map - BIOS Central
  - http://www.bioscentral.com/misc/cmosmap.htm
- CHIPSEC Platform Security Assessment Framework
  - BlackHat2014
  - https://www.blackhat.com/docs/us-14/materials/arsenal/us-14-Bulygin-CHIPSEC-Slides.pdf
- A Tour of Intel CHIPSEC
  - http://www.basicinputoutput.com/2016/05/a-tour-of-intel-chipsec.html
- Malicious Code Execution in PCI Expansion ROM
  - http://resources.infosecinstitute.com/pci-expansion-rom/