Monthly Research 2016.7
About security assessment framework “CHIPSEC”
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

```
Module: SPI Flash Controller Configuration Lock

[*] HSFS = 0xFOO0 << 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
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:

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>0A</th>
<th>0B</th>
<th>0C</th>
<th>0D</th>
<th>0E</th>
<th>0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>48</td>
<td>13</td>
<td>32</td>
<td>07</td>
<td>10</td>
<td>04</td>
<td>05</td>
<td>22</td>
<td>07</td>
<td>16</td>
<td>26</td>
<td>02</td>
<td>50</td>
<td>80</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>7F</td>
<td>02</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>20</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>30</td>
<td>FF</td>
<td>FF</td>
<td>20</td>
<td>FF</td>
<td>FF</td>
<td>36</td>
<td>0C</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>0B</td>
</tr>
<tr>
<td>40</td>
<td>00</td>
<td>00</td>
<td>C0</td>
<td>17</td>
<td>41</td>
<td>28</td>
<td>F0</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>50</td>
<td>00</td>
<td>25</td>
<td>21</td>
<td>00</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>25</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>60</td>
<td>00</td>
<td>00</td>
<td>17</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>F0</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>70</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>
```
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

• A Tour of Intel CHIPSEC

• Malicious Code Execution in PCI Expansion ROM
  – http://resources.infosecinstitute.com/pci-expansion-rom/