Using Python 3 in the UEFI Shell for Platform Security Analysis

UEFI 2022 Virtual Summit
August 16, 2022
Aaron Frinzell, Nathaniel Mitchell, Sara Batllori, Dan Scott
Meet the Presenters

Aaron Frinzell  
security research

Nathaniel Mitchell  
security research

Sara Batllori  
security research

Dan Scott  
security research
More Questions?

Following today’s webinar, join the live, interactive WebEx Q&A for the opportunity to chat with the presenters

- Meeting number: 2558 523 4281
  Password: uefiforum (83343678 from phones and video systems)
Agenda

• EFI Shell Python 3.6.8 Implementation
• CHIPSEC in the EFI Shell
CHIPSEC

Python based framework for analyzing the security of PC platforms including hardware, system firmware (BIOS/UEFI), and platform components. It includes a security test suite, tools for accessing various low-level interfaces, and forensic capabilities.

Source: https://github.com/chipsec/chipsec
Our First Try...

• Python 2.7
• Based on the EDK2 implementation
• Support two Python versions
• EOL’d January 1, 2020 (PEP 373)

Python 2.7 EDK2:  https://github.com/tianocore/edk2-archive/blob/master/AppPkg/Applications/Python/PythonReadMe.txt

Python 2 EOL:  https://www.python.org/doc/sunset-python-2/
EDK2-LIBC Python Implementation

• Python 3.6.8 support added Sept 2, 2021
• https://github.com/tianocore/edk2-libc
• /AppPkg/Applications/Python/Python-3.6.8/

Source: https://github.com/tianocore/edk2-libc/commit/7769ee0af3429ca650b651a7894308ae09906302
• Decoupled text and binary data
  – str, bytes, & Unicode
  – Py2 could use str type for both text and binary data
  – Py3 str now always Unicode
• Class and Syntax changes
• cpython changes
  – Parsing arguments and format specifier deprecation
    • u# - old style Py_UNICODE API
  – Method Calling convention:
    • METH_OLDARGS deprecated
    • METH_VARARGS, function type PyCFunction

Source: https://docs.python.org/3/howto/pyporting.html
CHIPSEC edk2module.c Changes

Source: [https://github.com/chipsec/chipsec/tree/main/chipsec_tools/edk2/PythonEFI](https://github.com/chipsec/chipsec/tree/main/chipsec_tools/edk2/PythonEFI)
Build Prep

• Latest EDK2
  – [https://github.com/tianocore/edk2](https://github.com/tianocore/edk2)
    • git clone git@github.com:tianocore/edk2.git
  – Update submodules
    • cd edk2
    • git submodule update --init

• Latest EDK2-LIBC
  – [https://github.com/tianocore/edk2-libc](https://github.com/tianocore/edk2-libc)
    • git clone git@github.com:tianocore/edk2-libc.git

Source: [https://github.com/tianocore/edk2/blob/master/ReadMe.rst](https://github.com/tianocore/edk2/blob/master/ReadMe.rst)
CHIPSEC Source Code

- Files to copy/overwrite:
  - edk2module.c - all CHIPSEC related functions (#ifdef CHIPSEC)
  - cpu.asm – asm functions
Python Updates

- AppPkg/Applications/Python/Python-3.6.8
  - Execute srcprep.py
  - Overwrite:
    - /PyMod-3.6.8/Modules/edk2module.c
  - Add:
    - /PyMod-3.6.8/Modules/cpu.asm
  - Update:
    - /Python368.inf
    - Add ‘PyMod-$(PYTHON_VERSION)/Modules/cpu.asm’ under [Sources.X64] section

Source: https://chipsec.github.io/installation/USB%20with%20UEFI%20Shell.html
Build python368.efi

- Build efi
  - build -a X64 -p AppPkg\AppPkg.dsc
- Build Python package
  - cd AppPkg\Applications\Python\Python-3.6.8
  - create_python368_pkg.bat (.sh)
    - Switches: <tool_chain> <target> <arch> <out_folder>
    - Example:
      
      create_python368_pkg.bat VS2019 RELEASE X64 d:\

Source: https://chipsec.github.io/installation/USB%20with%20UEFI%20Shell.html
It Works, But…

• Most platforms ‘worked’ but a handful exhibited GP
• Enter uuid.py library
  – Uses environmental specific calls to generate random UUID
  – Defaulted to Linux routines for reading Unix timestamp and MAC address

Source: https://github.com/tianocore/edk2-libc/commit/c3222fed9927420fc46da503dea1ebb874698b6
 EFI Shell Python Setup

1. https://github.com/chipsec/chipsec
2. /__install__/UEFI/chipsec_py368_uefi_x64.zip
3. FAT32 media
4. Extract to /EFI
5. Boot to Shell

Source: https://github.com/chipsec/chipsec/blob/main/__install__/UEFI/chipsec_py368_uefi_x64.zip
Using EFI Shell Python

• fs0:> python368.efi
• >>> import edk2

```
fs0:/> python368
Python 3.6.8 (default, Feb 23 2022, 17:43:18) [C] on uefi
Type "help", "copyright", "credits" or "license" for more information.
>>> import edk2
>>> hex(edk2.rdmsr(0x1f2)[0])
'0x88400006'
```
CHIPSEC EDK2 Library

```c
edk2.readmem(phys_address_lo, phys_address_hi, length)
edk2.readmem_dword(phys_address_lo, phys_address_hi)
edk2.writemem(phys_address_lo, phys_address_hi, buffer, length)
edk2.writemem_dword(phys_address_lo, phys_address_hi, value)
edk2.allocphysmem(length, max_pa)[0]
edk2.readio(io_port, size)
edk2.writeio(io_port, size, value)
edk2.readpci(bus, device, function, address, size)
edk2.writepci(bus, device, function, address, value, size)
edk2.swsmi(SMI_code_data, _rax, _rbx, _rcx, _rdx, _rsi, _rdi)
edk2.rdmsr(msr_addr)
edk2.wrmsr(msr_addr, eax, edx)
edk2.cpuid(eax, ecx)
edk2.GetVariable(name, guidstr, size)
edk2.GetNextVariableName(size, name, guid)
edk2.SetVariable(name, guidstr, int(attrs), data, datasize)
```
CHIPSEC Release

• Python 2 is EOL’d in CHIPSEC
• CHIPSEC v1.8.0 introduced Python 3.6.8 support, released Dec 2021

Source: https://github.com/chipsec/chipsec/releases/tag/1.8.0
Running CHIPSEC

• Shell> fs0:
• FS0:> cd chipsec
• FS0:\chipsec\> python368 chipsec_main.py
Or
• FS0:\chipsec\> python368 chipsec_util.py
Demo
CHIPSEC: Platform Hardware Security Assessment Framework

CHIPSEC] Version 1.8.7
CHIPSEC] Arguments: -l cmd_nuc.txt

CHIPSEC] API mode: using CHIPSEC kernel module API
CHIPSEC] OS : uefi
CHIPSEC] Python : 3.6.8 (64-bit)
CHIPSEC] Helper : EfiHelper (None)
CHIPSEC] Platform: CometLake U
CHIPSEC] VID: 8086
CHIPSEC] DID: 9B61
CHIPSEC] RID: 0C
CHIPSEC] PCH : Intel 400 series PCH-LP Prem-U
CHIPSEC] VID: 8086
CHIPSEC] DID: 0284
CHIPSEC] RID: 00
Log Module

137 [*] Running module: chipsec.modules.common.cpu.spectre_v2
138 [x] Module: Checks for Branch Target Injection / Spectre v2 (CVE-2017-5715)
141 [*] CPUID.7H:EDX[26] = 1 Indirect Branch Restricted Speculation (IBRS) & Predictor Barrier (IBPB)
142 [*] CPUID.7H:EDX[27] = 1 Single Thread Indirect Branch Predictors (STIBP)
143 [*] CPUID.7H:EDX[29] = 1 IA32_ARCH_CAPABILITIES
144 [+] CPU supports IBRS and IBPB
145 [+] CPU supports STIBP
146 [*] Checking enhanced IBRS support in IA32_ARCH_CAPABILITIES...
147 [*] cpu0: IBRS_ALL = 1
148 [+] CPU supports enhanced IBRS (on all logical CPU)
149 [*] Checking if OS is using Enhanced IBRS...
150 [*] cpu0: IA32_SPEC_CTRL[IBRS] = 0
151 [*] cpu0: IA32_SPEC_CTRL[STIBP] = 0
152 [-] OS doesn't seem to use Enhanced IBRS
153 [#] INFORMATION: Unable to determine if the OS uses STIBP
154 [!] WARNING: CPU supports mitigation (enhanced IBRS) but OS is not using it
155 [!] OS may be using software based mitigation (e.g. retpoline)
156 [!] WARNING: Retpoline check not implemented in current environment
Log Summary

<table>
<thead>
<tr>
<th>CHIPSEC</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time elapsed</td>
<td>1,000</td>
</tr>
<tr>
<td>Modules total</td>
<td>27</td>
</tr>
<tr>
<td>Modules failed to run</td>
<td>0</td>
</tr>
<tr>
<td>Modules passed</td>
<td>19</td>
</tr>
</tbody>
</table>

[+] PASSED: chipsec.modules.common.bios_kbrd_buffer
[+] PASSED: chipsec.modules.common.bios_smi
[+] PASSED: chipsec.modules.common.bios_ts
[+] PASSED: chipsec.modules.common.biws_wp
[+] PASSED: chipsec.modules.common.debugenabled
[+] PASSED: chipsec.modules_common.ia32cfg
[+] PASSED: chipsec.modules_common.me_mfg_mode
[+] PASSED: chipsec.modules_common.memlock
[+] PASSED: chipsec.modules_common.remap
[+] PASSED: chipsec.modules_common.secureboot_variables
[+] PASSED: chipsec.modules_common.smm
[+] PASSED: chipsec.modules_common.smm_codechk
[+] PASSED: chipsec.modules_common.smm_dma
[+] PASSED: chipsec.modules_common.smrr
[+] PASSED: chipsec.modules_common.spd_wd
[+] PASSED: chipsec.modules_common.spi_desc
[+] PASSED: chipsec.modules_common.spi_fdpss
[+] PASSED: chipsec.modules_common.spi_lock
[+] PASSED: chipsec.modules_common.uefi.access_uefi_spec

[CHIPSEC] Modules information 1:

[#] INFORMATION: chipsec.modules.common.cpu.cpu_info

[CHIPSEC] Modules failed 0:

[CHIPSEC] Modules with warnings 4:

[!] WARNING: chipsec.modules.common.cpu.spectre_v2
[!] WARNING: chipsec.modules_common.rtclock
[!] WARNING: chipsec.modules_common.spi_access

[CHIPSEC] Modules skipped 0:

[CHIPSEC] Modules not applicable 3:

[*] NOT APPLICABLE: chipsec.modules_common.cpu.untrusted
[*] NOT APPLICABLE: chipsec.modules_common.memconfig
[*] NOT APPLICABLE: chipsec.modules_common.sgx_check

[CHIPSEC] ********************************************************************************

www.uefi.org
What’s Next?

• See for yourself
  • Run Python 3 in the shell
    • Use CHIPSEC's pre-compiled version [github.com]
    • Build it yourself from EDK2 [github.com]
  • Join the CHIPSEC community
    • CHIPSEC's latest release [github.com]
    • Join the Discord Channel [discord.gg]
    • Quarterly Community Meetings [github.com]
• If you are an Intel customer with an CNDA in place, contact chipsec@intel.com for additional tests and platform support.

CHIPSEC’s Discord
Questions?
More Questions?

Following today’s webinar, join the live, interactive WebEx Q&A for the opportunity to chat with the presenters

• **Visit this link to attend:** [https://bit.ly/3OnTLCr](https://bit.ly/3OnTLCr)
• **Meeting number:** 2558 523 4281  
  **Password:** uefiforum (83343678 from phones and video systems)
Thanks for attending the UEFI 2022 Virtual Summit

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org

Notices & Disclaimers:
Intel technologies may require enabled hardware, software or service activation. No product or component can be absolutely secure. Your costs and results may vary.
© Intel Corporation.
Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.