Less-Insecure Network Edge Virtualization with Low Size, Weight and Power

Platform Security Summit 2019 (10/02/2019)

Piotr Król
Piotr Król
Founder & Embedded Systems Consultant

- open-source firmware
- platform security
- trusted computing
- @pietrushnic
- piotr.krol@3mdeb.com
- linkedin.com/in/krolpiotr
- facebook.com/piotr.krol.756859
• Problem statement, goals and motivation
• TPM 2.0 in open-source
• PC Engines hardware and firmware
• SRTM: coreboot and GRUB2
• DRTM: TrenchBoot
• PCRMs and measurements
• OPNsense and NDVM performance
• Demo
• Where to go from here?
Edge/Fog Computing and IoT Gateways hype will increase the amount of network appliance devices.
Vulnerability in network appliance software may lead to malicious firmware modification, which can be undetected over the whole life cycle of the device.
Toolbox for reestablishing trust in firmware is almost empty and not easy to use.
How we can build less-insecure system?

- Use virtualization to isolate system components
  - strong, hardware-supported isolation
- Implement narrow interfaces
  - devices disaggregation
- Open-source software stack
  - on every stage of platform operation
  - in each device controller
- Reproducible builds
- Modern hardware features
  - firmware storage security
  - secure launch (aka late launch or secure startup)
  - IOMMU
- Trusted Platform Module
Goal and Motivation

How many solutions are shipped with those features working out-of-the-box?

How open those solutions are?

Goal

Create less-insecure virtual network appliance

Motivation

- Need for secure network appliance which measurements can be attested
- Creation TrenchBoot implementation for AMD platforms
- Build solid ecosystem for Edge Computing "era"
- Building foundation for secure remote updates
• Trusted Platform Modules are plug-and-play modules
• TPM enables SRTM and DRTM fundamental for platform integrity
• TPM 1.2 are common and well supported, but new designs should use 2.0
• Recent advancement in open-source TPM 2.0 support
  ○ tpm2-software (TSS, PKCS#11, OpenSSL Engine)
    https://github.com/tpm2-software
  ○ TPM Genie mitigation in Linux kernel
    https://github.com/nccgroup/TPMGenie
  ○ FreeBSD Secure Boot support
    https://papers.freebsd.org/2019/bsdcan/stanek-improving_security_of_the_freebsd_boot_process/
• TPM 2.0 is mandatory for Edge Computing and IoT Gateways (AWS, Azure)
- Made by open-minded Swiss company
- Frequently OEMed used as base for other products by other vendors
- AMD Embedded G series GX-412TC, 1 GHz quad Jaguar core with 64 bit
- AES-NI
- SKINIT for DRTM
- 4GB DDR3-1333 DRAM with ECC
- SD, USB, mSATA, SATA
- 2x mPICe (one with SIM socket)
- 3 Gigabit Ethernet channels using Intel i210AT / i211AT NICs depending on the model
- 6-10W@12V DC
- Lot of buses (RS232, LPC, GPIO, I2C)
PC Engines open-source firmware

- There are already thousands of devices in the field
- Each can be leveraged as a less-insecure network appliance
- Reasonable community that cares about open-source firmware
- Regular monthly firmware releases (for last 27 months)
- Reproducible builds
- Signed hashes provided
  - QubesOS-like key chain
  - 3mdeb master->3mdeb open-source firmware->PC Engines release key
• SPI image is build using the coreboot build system
• SRTM (or rather S-CRTM) implemented based on vboot library
• coreboot measures all boot stages and payload (GRUB2)
• SPI image can be build using the following repository
  ○ [https://github.com/pcengines/coreboot/tree/pcengines_trenchboot](https://github.com/pcengines/coreboot/tree/pcengines_trenchboot)
  ○ Use config.pcengines_apu2.tb
• Features
  ○ measured boot
  ○ verified boot (booting unsigned code result triggers recovery mode)
  ○ firmware updates
  ○ signed SPI images
Required for SRTM implementation
Default Winbond chip for PC Engines apu2 doesn't have that feature
Adesto AT25SF641 which provide features w/o "special order"
Flashrom - A utility for identifying, reading, writing, verifying and erasing flash chips
PC Engines fork of flashrom was extended to support Adesto chip and OTP feature

https://github.com/pcengines/flashrom

---

Read-only SPI region

<table>
<thead>
<tr>
<th></th>
<th>Lock-Down</th>
<th>until the next power-down, power-up cycle.(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

One Time Program\(^{(2)}\) | Status Register is permanently protected and can not be written to.

Note:
1. When SRP1, SRP0 = (1, 0), a power-down, power-up cycle will change SRP1, SRP0 to (0, 0) state. 2. This feature is available upon special order. Please contact Winbond for details.
- Open-source ecosystem-wide framework for launch integrity
- Main goal is to enable out-of-the-box support for TCG D-RTM in open-source ecosystem
- Currently targeting Intel and AMD implementation design is prepared to handle other solutions
- Note that the AMD TrenchBoot implementation is completely open-source. This is not possible with Intel.
coreboot, TrenchBoot, and hypervisor

**SPI flash**
- coreboot (SRTM) → Run payload → GRUB2
- SKINIT (DRTM) → TrenchBoot Landing Zone → jmp → Linux+initramfs

**Storage**
- create VM → Xen
- create VM → dom0
- NDVM
- create VM → OPNsense
Hardware platform

- Trusted Platform Module (TPM) 2.0
- SSD 16 GB mSata
- 3 x Gigabit Ethernet
- SPI
Visible during coreboot boot

| PCR-2 | a16bc92eb28ae11c13ad6d9c2ad0632dc9909983f4b17663dbf388eb756ddf9c | SHA256   | [FMAP: COREBOOT CBFS: bootblock] |
| PCR-2 | 8035cc56c197087c7950498bd75c5841a8c7886eb8236ee14d662f191d95d8dad | SHA256   | [FMAP: COREBOOT CBFS: fallback/romstage] |
| PCR-0 | 62571891215b4eefc1ceab744ce59dd866ea6f73 | SHA256   | [VBOOT: boot mode] |
| PCR-1 | a66c8c2cda246d322d0c2025b6266e1e23c89410051002f46bfad1c9265f43d0 | SHA256   | [VBOOT: GBB HWID] |
| PCR-2 | 957757f20415edc1c6f7e5acbaaf9c9ac4c858ec2df1913958be0655daf45a | SHA256   | [FMAP: COREBOOT CBFS: fallback/ramstage] |
| PCR-2 | 6c1d20616d91442b61de89de6bf81f00e8e929919c9284061e00d004de893994 | SHA256   | [FMAP: COREBOOT CBFS: spd.bin] |
| PCR-3 | 585721d9e083591b901df05178b7124a73602d88c4ab8258793d8658ab5061 | SHA256   | [PSPDIR] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 1f58561c980ddd7c6d3c3cb6a845894cf674dc754b9215b54b85057b25ed3c1ea | SHA256   | [FMAP: COREBOOT CBFS: AGESA] |
| PCR-2 | 5b6d45566a1b3157a6f437c320be95fe85eb371eeab63efcd4520be763572f4683 | SHA256   | [FMAP: COREBOOT CBFS: fallback/dsdt.aml] |
| PCR-2 | 6c1d20616d91442b61de89de6bf81f00e8e929919c9284061e00d004de893994 | SHA256   | [FMAP: COREBOOT CBFS: spd.bin] |
| PCR-2 | 19d059948d4188fca951d7297defe02d68a221dd78a9188s15c69994c82c8fd | SHA256   | [FMAP: COREBOOT CBFS: fallback/payload] |
• Obtained in Xen dom0

<table>
<thead>
<tr>
<th></th>
<th>sha1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1e745033ad915853b44c9439116f311dd85011c8</td>
</tr>
<tr>
<td>1</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>4</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>5</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>6</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>7</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>8</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>9</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>10</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>11</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>12</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>13</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>14</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>15</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>16</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>17</td>
<td>169a4fe87b0df9470678339ba78ba05e6fb9489</td>
</tr>
<tr>
<td>18</td>
<td>3d9f3514b22efe39c15094110f109e9ab06d5daf</td>
</tr>
<tr>
<td>19</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>20</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>21</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>22</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>23</td>
<td>0000000000000000000000000000000000000000</td>
</tr>
</tbody>
</table>
Obtained in Xen dom0

sha256:
0: 0000000000000000000000000000000000000000000000000000000000000000
1: d965b906c85450d5aad254368b53f043480e811b590ce37a524331d2b9135368
2: 6e6e3d56aa878cd7f8ae745104d589f299938cd4009c3c79ea9cad2701690b12
3: f80ac628144ea6ac2743d9a48715e8f1ecb7458925d1d2a1b9e4bf5011ae1f5b
4: 0000000000000000000000000000000000000000000000000000000000000000
5: 0000000000000000000000000000000000000000000000000000000000000000
6: 0000000000000000000000000000000000000000000000000000000000000000
7: 0000000000000000000000000000000000000000000000000000000000000000
8: 0000000000000000000000000000000000000000000000000000000000000000
9: 0000000000000000000000000000000000000000000000000000000000000000
10: 0000000000000000000000000000000000000000000000000000000000000000
11: 0000000000000000000000000000000000000000000000000000000000000000
12: 0000000000000000000000000000000000000000000000000000000000000000
13: 0000000000000000000000000000000000000000000000000000000000000000
14: 0000000000000000000000000000000000000000000000000000000000000000
15: 0000000000000000000000000000000000000000000000000000000000000000
16: 0000000000000000000000000000000000000000000000000000000000000000
17: 0d7bc289be9cdfec68e69665de67bdbd92aaa860a8bf4db49803991690db3a6
18: 0000000000000000000000000000000000000000000000000000000000000000
19: 0000000000000000000000000000000000000000000000000000000000000000
20: 0000000000000000000000000000000000000000000000000000000000000000
21: 0000000000000000000000000000000000000000000000000000000000000000
22: 0000000000000000000000000000000000000000000000000000000000000000
23: 0000000000000000000000000000000000000000000000000000000000000000
SHA-1

- **PCR0** - extended by vboot logic with boot mode (normal, dev, recovery, keyblock)

SHA-256

- **PCR1** - extended by vboot logic and represent hardware ID
- **PCR2** - used by coreboot to extend measurements of all loaded and/or executed components from CBFS'es, either read-only or read-write parts
- **PCR3** - used by coreboot to extend measurements of variable and runtime data that might change across boots (MRC cache, CMOS configuration, etc.)
- **PCR4-16** - not used
SHA-1

- **PCR17** - extended by TrenchBoot Landing Zone code and measures LZ, Linux+u-root
- **PCR18** - command line used for booting Linux+u-root

SHA-256

- **PCR17** - extended natively by AMD SKINIT instruction
• pfSense fork since 2015
• BSD 2-Clause “Simplified” license
• different UI
• IPS/IDS based on Suricata included
• weekly security updates
Performance tests setup

VM1 (iperf client)  VM2 (iperf server)  VM3 (iperf client)

Xen dom0

NIC1  NIC2  NIC3

Ethernet

PCI pass-through

PC (iperf server)
• VM1 client (VM to VM test)

<table>
<thead>
<tr>
<th>[ID]</th>
<th>Interval</th>
<th>Transfer</th>
<th>Bitrate</th>
<th>Retr</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5]</td>
<td>0.00-120.00 sec</td>
<td>13.1 GBytes</td>
<td>940 Mbits/sec</td>
<td>575</td>
<td>sender</td>
</tr>
<tr>
<td>[5]</td>
<td>0.00-120.04 sec</td>
<td>13.1 GBytes</td>
<td>939 Mbits/sec</td>
<td></td>
<td>receiver</td>
</tr>
</tbody>
</table>

• VM2 server (VM to VM test)

<table>
<thead>
<tr>
<th>[ID]</th>
<th>Interval</th>
<th>Transfer</th>
<th>Bitrate</th>
<th>Retr</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5]</td>
<td>0.00-120.00 sec</td>
<td>13.1 GBytes</td>
<td>940 Mbits/sec</td>
<td>575</td>
<td>sender</td>
</tr>
<tr>
<td>[5]</td>
<td>0.00-120.04 sec</td>
<td>13.1 GBytes</td>
<td>939 Mbits/sec</td>
<td></td>
<td>receiver</td>
</tr>
</tbody>
</table>

• VM3 client (VM to PC test)

<table>
<thead>
<tr>
<th>[ID]</th>
<th>Interval</th>
<th>Transfer</th>
<th>Bitrate</th>
<th>Retr</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5]</td>
<td>0.00-120.00 sec</td>
<td>13.1 GBytes</td>
<td>940 Mbits/sec</td>
<td>575</td>
<td>sender</td>
</tr>
<tr>
<td>[5]</td>
<td>0.00-120.04 sec</td>
<td>13.1 GBytes</td>
<td>939 Mbits/sec</td>
<td></td>
<td>receiver</td>
</tr>
</tbody>
</table>
Demo time
What we can do with that?

- Security
  - reproducible builds of whole software stack
  - only firmware signed by trusted party can be used
  - possibility of reestablishing trust without reboot
  - remote attestation
- Multifunction network appliance with clean isolation
- Simplified configuration reproducibility and migration
- Multiple levels of management
  - direct connection with firewall VM for network management
  - orchestration through hypervisor
- With additional VMs sky is the limit (or rather hardware performance)
- All of that without losing trust in your network appliance
- And even if that will happen you detect and can reestablish it
Where to go from here?

- [https://trenchboot.github.io](https://trenchboot.github.io)
- [https://github.com/TrenchBoot](https://github.com/TrenchBoot)
- [https://groups.google.com/forum/#!forum/trenchboot-devel](https://groups.google.com/forum/#!forum/trenchboot-devel)
3mdeb is based in Gdańsk, Poland
Over 4 years we worked with 50 customers from 21 countries
Example engagements

- Open-source firmware implementation and maintenance

- IoT Gateways and Edge/Fog Computing devices
• 3mdeb inside

• community projects manufactured by 3mdeb
• Examples engagements
  ◦ From hardware bring-up to complete Board Support Package (BSP)
  ◦ x86 firmware (BIOS/UEFI/coreboot) development, debugging and optimization
  ◦ AWS and Microsoft IoT Cloud integration
  ◦ firmware and embedded systems maintenance
• Desired partnership
  ◦ hardware makers and OEMs
  ◦ open-source firmware promoters
• Hiring
  ◦ we always look for people motivated to open firmware ecosystem and promote open solutions on the edge
  ◦ Interested? Please send CV
• If you looking for commercial support feel free to visit our website or contact us:
  ◦ https://3mdeb.com
  ◦ contact@3mdeb.com
Q&A
Hardware platforms limitations

- TPM2.0 over LPC
  - TPM Genie and sniffing issues
  - Patches to Linux kernel under review
- SPI header
  - anyone with recovery dongle and working firmware
• C environment bootblock for apu2
  ◦ we cannot use vboot before romstage
  ◦ everything up to romstage has to be locked by RO flag
• Recovery mode
  ◦ currently, there is not enough space
  ◦ we deployed commercial solutions with SRTM, firmware A/B, and recovery mode
• TPM2.0 logs are stored in TCPA (TPM1.2 log)
  ◦ TPM2 logs are different than TCPA logs, implement correct TPM2 log format and differentiate from TCPA log API
• minimize the read-only locked code by moving raminit code to read-write partitions
• save TPM2 measurements in TPM2 log area in a format compliant with TCG (coreboot)
• implement TrenchBoot support for the direct secure launch of Xen
• use SHA256 sums for all PCRs in TrenchBoot and coreboot

Start trusting Your BIOS SRTM with vboot, TPM and permanent flash protection, Michał Żygowski, OSFC 2019