DECAF: Automatic, Adaptive De-bloating and Hardening of COTS Firmware

Supported by the Office of Naval Research

Today:
Jake Christensen
Radu Sion
Introduction

- Despite its privileged position, firmware is almost entirely opaque to the end-user
- The delivered blob is the result of a long chain (e.g. EDK II, American Megatrends, Dell)
- Code is of questionable quality
- Lots of code reuse leads to easily replicable attacks
  - Kovah & Kallenberg 2015
- Many (up to 69%) modules are unnecessary
SerialNumStrLen = StrLen(SerialNumberPtr);
if (SerialNumStrLen > SMBIOS_STRING_MAX_LENGTH)
{
    return EFI_UNSUPPORTED;
}
....
SKUNumStrLen = StrLen(SKUNumberPtr);
if (SerialNumStrLen > SMBIOS_STRING_MAX_LENGTH)
{
    return EFI_UNSUPPORTED;
}
....
FamilyStrLen = StrLen(FamilyPtr);
if (SerialNumStrLen > SMBIOS_STRING_MAX_LENGTH)
{
    return EFI_UNSUPPORTED;
}
Introducing DECAF

- DECAF is an extensible platform for debloating commercial UEFI firmware
- Automatically prune up to 70% of an image!
- No source code needed
- Customizable functionality
- DECAFed firmware running in production data centers since mid-2017
Benefits of pruning

- Remove potentially unknown vulnerabilities
- Removed code is NOT unused/unreachable
- Pruned firmware boots faster, and contains less potentially vulnerable code
- Features can be removed on demand, while retaining other functionality

“Remove all other stuff you don’t want or need, if the firmware can still boot your OS - it’s fine to have that components removed”
Background: UEFI Firmware

- Splits platform initialization into four phases
  - Security (SEC)
  - Pre-EFI Initialization (PEI)
  - Driver Execution Environment (DXE)
  - Boot Device Selection (BDS)
- Basic building unit is a module (generally containing a PE32 executable)
- Modules communicate via EFI protocols
Dynamic firmware surgery

- Setup Control Environment Toolkit
- Module Management Tool
- UEFI ROM Dissect Tool
- BIOS Configuration Toolkit
- Setup Data Extraction
- Key Management
- Signing Server
- BIOS Morphing Engine
- IPMI Controller
- DHCP
- TFTP
- App Test Harness Image

COTS Hardware

Firmware

IPMI

DECAF Platform
Morphing Harness Modules

• **Gordon**
  – Motherboard Control
  – Flashing Mechanisms

• **Aura**
  – Firmware Binary Parser
  – Firmware Editor

• **Zarkov**
  – Runtime Validation Layer

• **Luigi**
  – Workflow Engine

• **Vultan**
  – ...
DECAF Pruning Overview

- Luigi workflow engine used for scheduling tasks ([https://github.com/spotify/luigi](https://github.com/spotify/luigi))
- Python layer based on UEFITool used for modifying images ([https://github.com/LongSoft/UEFITool](https://github.com/LongSoft/UEFITool))
- Python tools used to manage IPMI operations and collect info
- Docker images loaded onto booted images to validate the flashed firmware
- Custom dependency discovery modules written in C
Pruning Tasks and Phases

- Process can be parallelized on multiple boards
- Pruning happens in two phases: merge and hill climbing
  - Modules tried individually
  - Successfully removed groups are merged
  - Modules are then randomly selected and added to candidate solution
Dependency Discovery

- UEFI modules communicate with each other (using EFI protocols), creating dependencies
- Dependencies vary at runtime
- Module removal order becomes important!
- Solution: hijack the EFI protocol API and log active modules
Validation

DECAF employs several utilities to validate the pruned images:

- dmidecode
- lspci
- /proc/acpi
- CHIPSEC

CHIPSEC scans for known firmware vulnerabilities
- DECAF did not fix any CHIPSEC vulnerabilities
Results I

- Boot time reduction up to 24%
  - 55 to 44 seconds for SuperMicro
  - 34 to 27 seconds for Tyan
- DECAF can also selectively remove features
  - USB, network, VGA, etc
- Many common attacks on USB, network stack
  - BadUSB, Karsten Nohl and Jakob Lell, BlackHat 2014
- Example: 6/244 modules removed to disable USB on SuperMicro board
Results II

- DECAF can also selectively remove features
  - USB, network, VGA, etc
- Many common attacks on USB, network stack
  - BadUSB, Karsten Nohl and Jakob Lell, BlackHat 2014
- Example: 6/244 modules removed to disable USB on SuperMicro board
Initial Results: 50% reduction

DECAF Runtime - SuperMicro A2SDi

*Removed 154/312 modules
** ~50% of modules
** ~50% of binary
Initial Results: 40-70% reduction

DECAF Runtime - Tyan S5533

*Removed 134/194 modules
** ~70% of modules
** ~40% of binary
Initial Results: 62-70% reduction

DECAF Runtime - SuperMicro A1Sri

*Removed 152/244 modules
** ~62% of modules
** ~70% of binary
## Results II

<table>
<thead>
<tr>
<th>Motherboard</th>
<th>Original modules</th>
<th>Remaining modules</th>
<th>Reduction</th>
<th>Original Gadgets</th>
<th>Remaining Gadgets</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM A1SAi-2550F (V519)</td>
<td>244</td>
<td>90</td>
<td>63.11%</td>
<td>37846</td>
<td>14240</td>
<td>62.37%</td>
</tr>
<tr>
<td>Tyan 5533V101</td>
<td>194</td>
<td>60</td>
<td>69.07%</td>
<td>38776</td>
<td>20317</td>
<td>47.60%</td>
</tr>
<tr>
<td>HP DL380 Gen10</td>
<td>643</td>
<td>323</td>
<td>49.77%</td>
<td>183677</td>
<td>105116</td>
<td>42.77%</td>
</tr>
<tr>
<td>SM A1SAi-2550F (V827)</td>
<td>241</td>
<td>124</td>
<td>48.55%</td>
<td>37735</td>
<td>23055</td>
<td>38.90%</td>
</tr>
<tr>
<td>SM A2SDi-12C-HLN4F</td>
<td>313</td>
<td>194</td>
<td>38.02%</td>
<td>43593</td>
<td>31003</td>
<td>28.88%</td>
</tr>
<tr>
<td>SM A2SDi-H-TP4F</td>
<td>313</td>
<td>206</td>
<td>34.19%</td>
<td>44121</td>
<td>31024</td>
<td>29.68%</td>
</tr>
<tr>
<td>SM X10SDV-8C-TLN4F</td>
<td>316</td>
<td>286</td>
<td>9.49%</td>
<td>51534</td>
<td>45724</td>
<td>11.27%</td>
</tr>
</tbody>
</table>

*SM is short for SuperMicro*
Thank you for your attention!

Jake: jake@privatemachines.com
Radu: radu@privatemachines.com
Jan: jan@privatemachines.com
Sumeet: sumeet@privatemachines.com
Rob: rob@privatemachines.com
Mike: mike@privatemachines.com