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

Supported by the Office of Naval Research

Today:
Jake Christensen
Radu Sion

Private Machines
High Performance Zero Trust Infrastructure Tech

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

Code Sample: Intel Galileo firmware

```
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;
}
```

Analysis courtesy Nikolaj Schlej
(https://www.viva64.com/en/b/0326/)

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

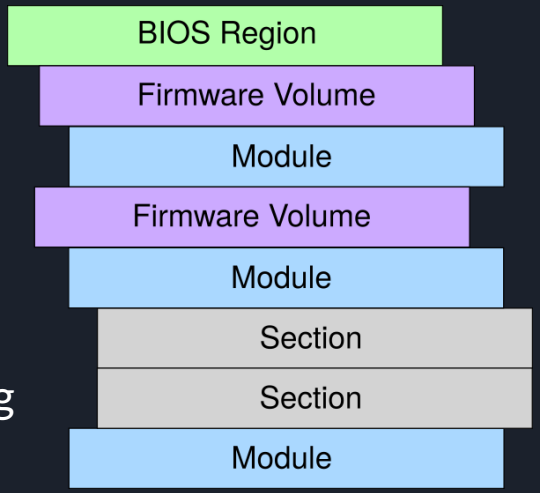
Benefits of pruning

- 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

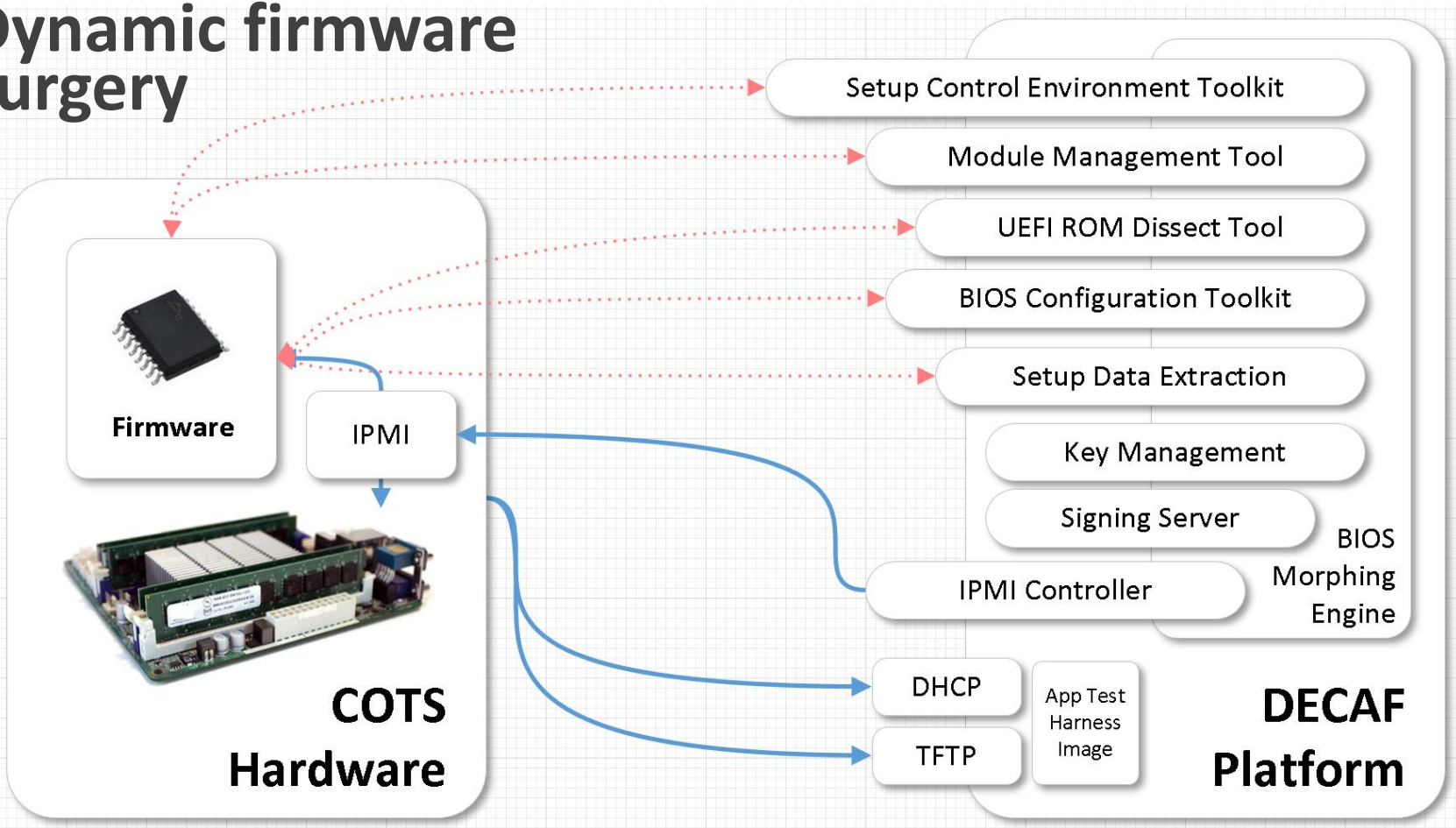
"Remove all other stuff you don't want or need, if the firmware can still boot your OS - it's fine to have that components removed"
- Nikolaj Schlej, Zero nights, 2015.

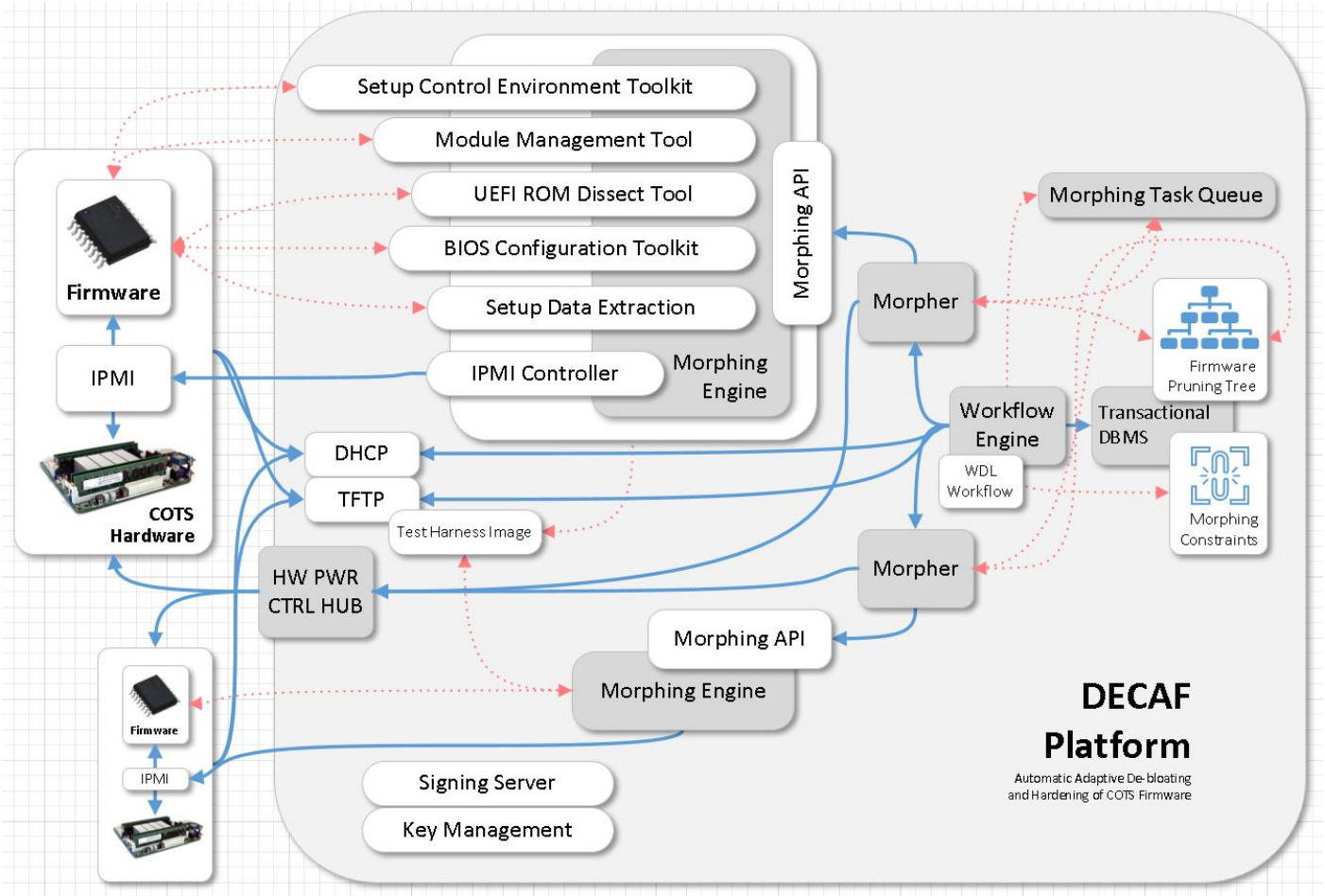
Background: UEFI Firmware

- 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



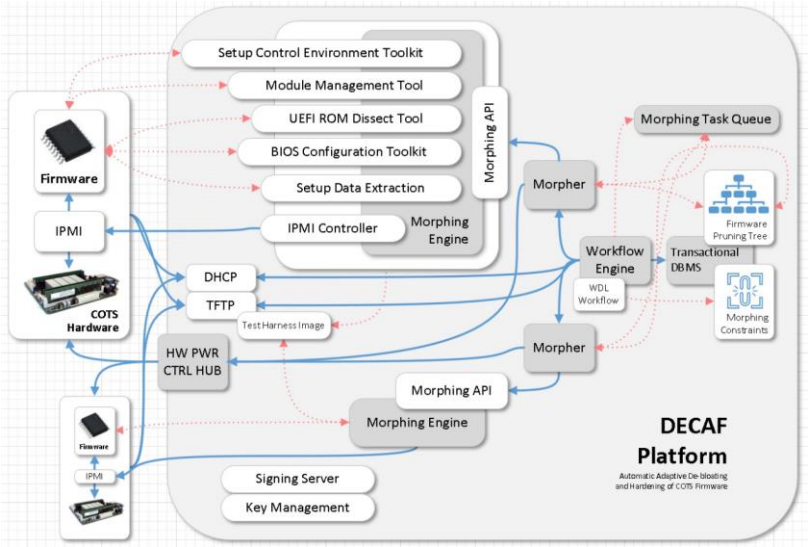
Dynamic firmware surgery



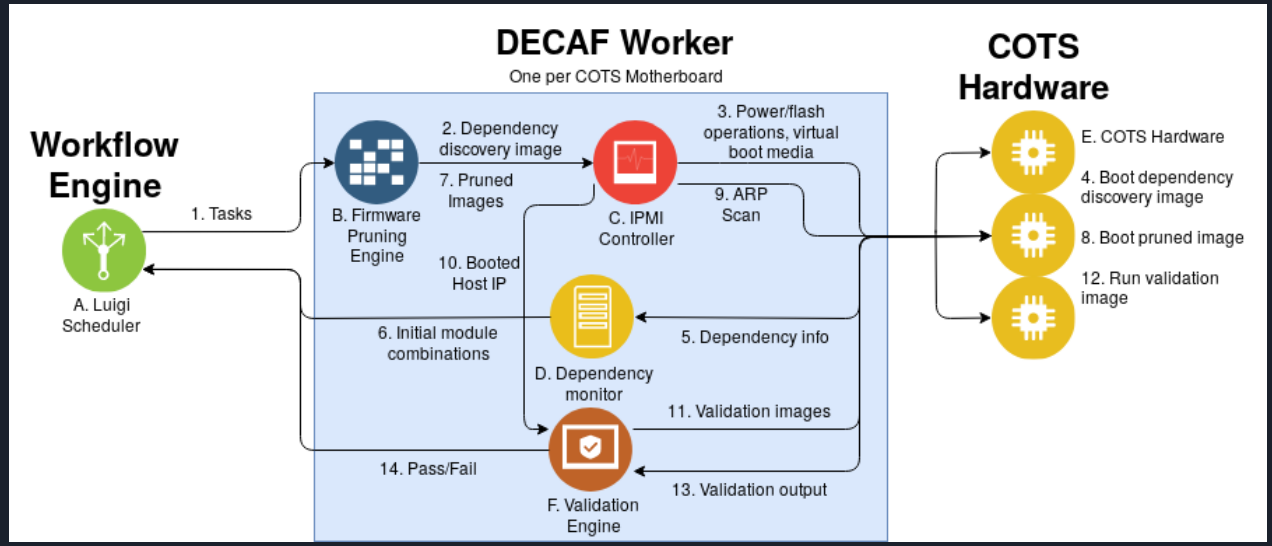

Setup Control Environment Toolkit
Module Management Tool
UEFI ROM Dissect Tool
BIOS Configuration Toolkit
Setup Data Extraction
IPMI Controller
Morphing Engine
Morphing API
Morphing Task Queue
Morpher
Firmware
IPMI
COTS Hardware
DHCP
TFTP
Test Harness Image
HW PWR CTRL HUB
Workflow Engine
WDL Workflow
Transactional DBMS
Firmware Pruning Tree
Morphing Constraints
Morpher
Morphing API
Morphing Engine
Firmware
IPMI
Signing Server
Key Management
DECAF Platform
Automatic Adaptive De-bloating and Hardening of COTS Firmware

Morphing Harness Modules

- Gordon
  - Motherboard Control
  - Flashing Mechanisms
- Aura
  - Firmware Binary Parser
  - Firmware Editor
- Zarkov
  - Runtime Validation Layer
- Luigi
  - Workflow Engine
- Vultan
  - …



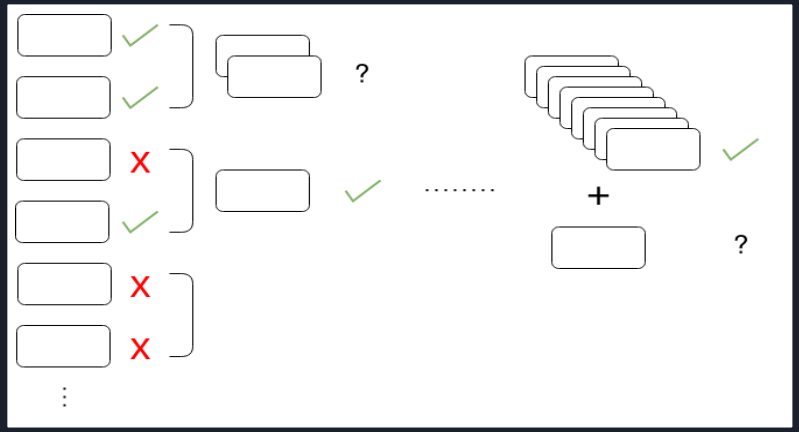
DECAF Pruning Overview



- Luigi workflow engine used for scheduling tasks (https://github.com/spotify/luigi)
- Python layer based on UEFITool used for modifying images (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

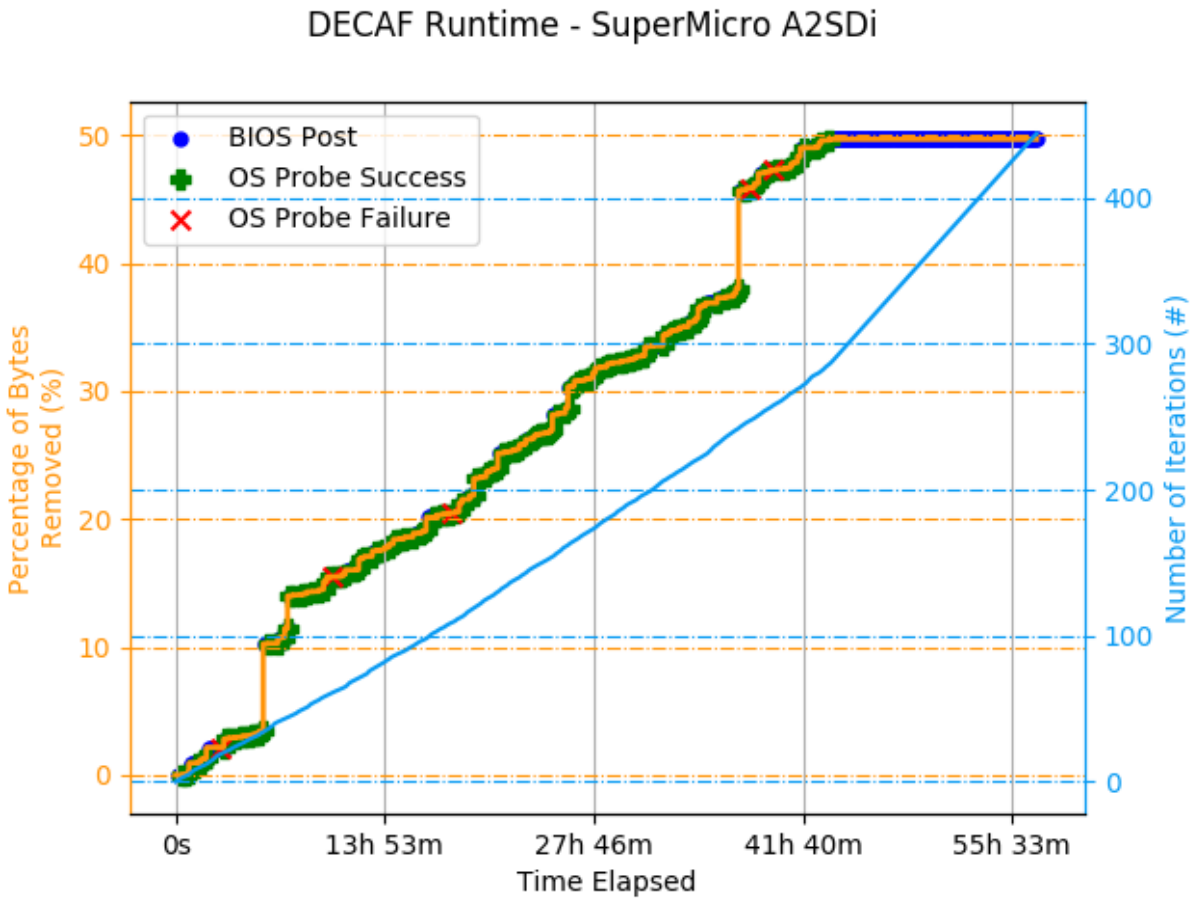
Results I

- 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

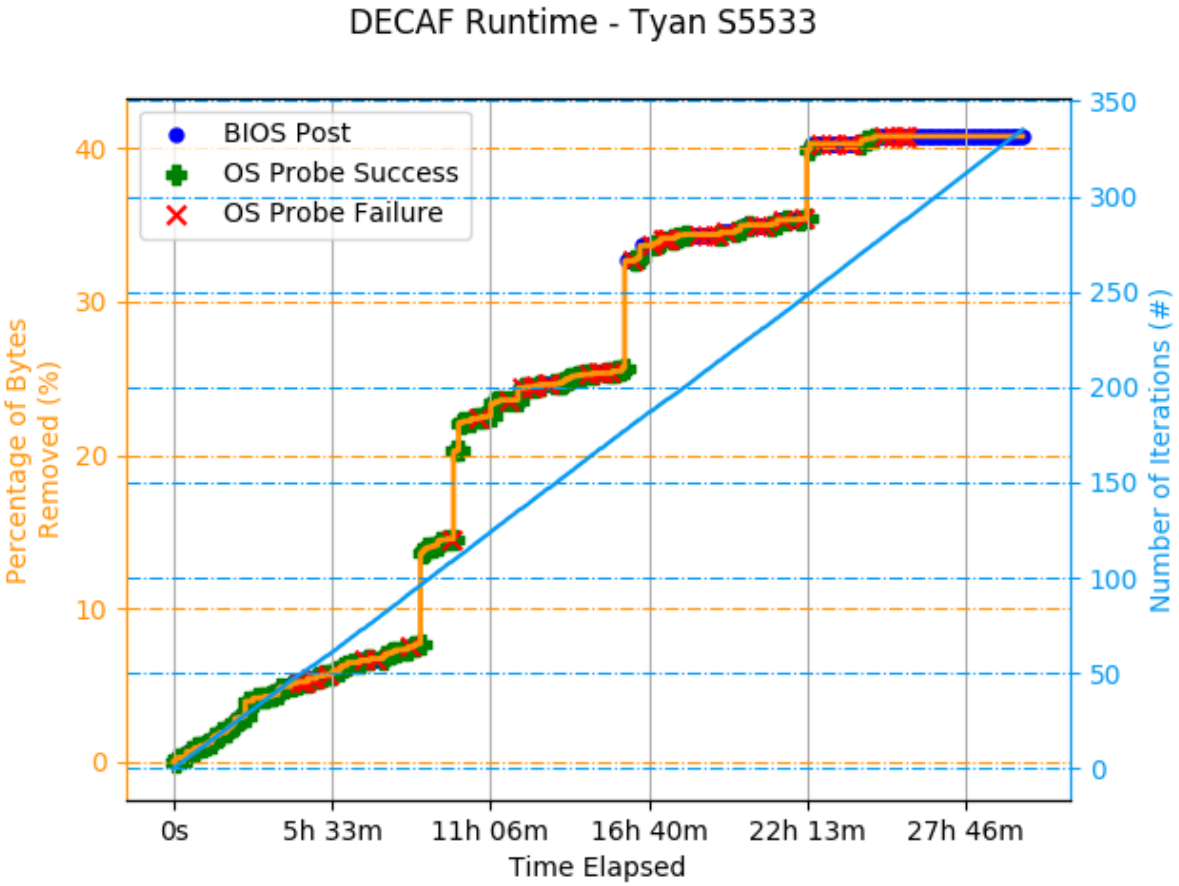


*Removed 154/312 modules

** ~50% of modules

** ~50% of binary

Initial Results: 40-70% reduction

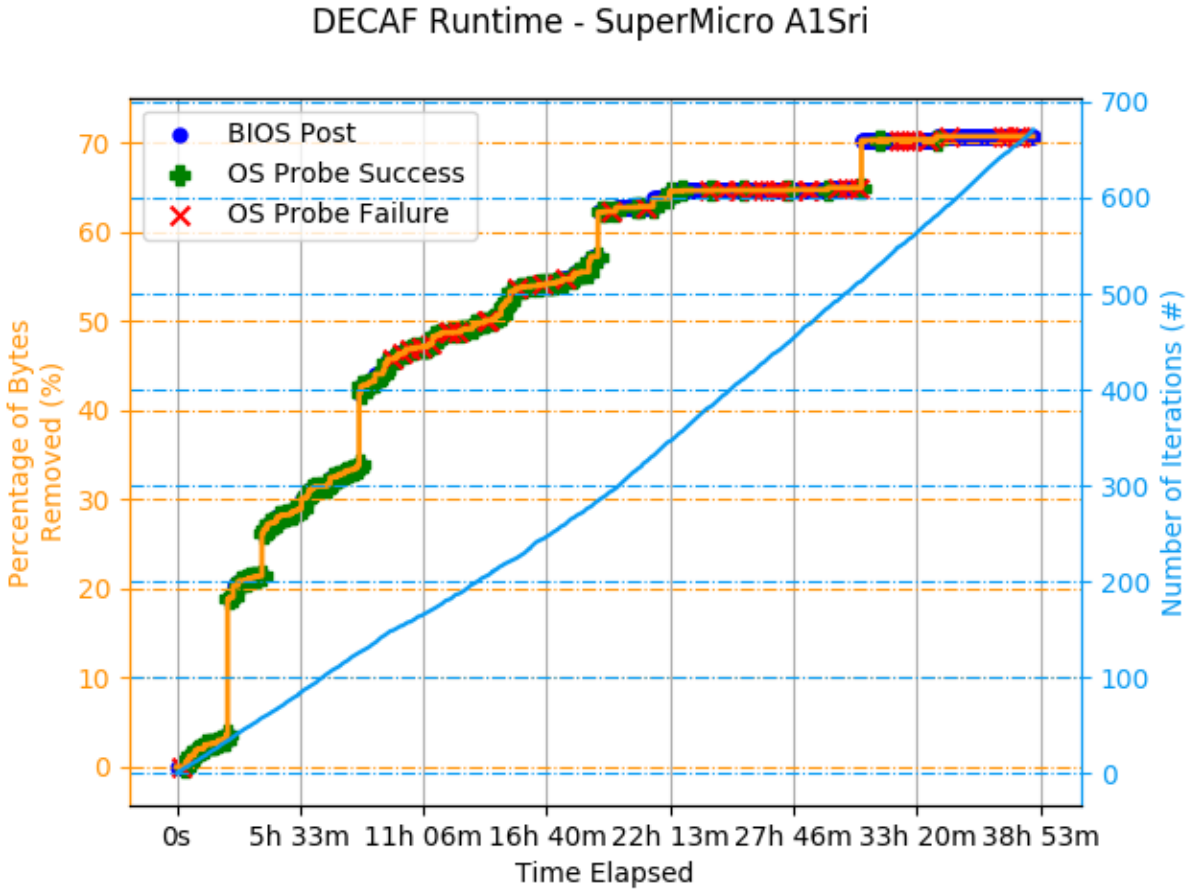


*Removed 134/194 modules

** ~70% of modules

** ~40% of binary

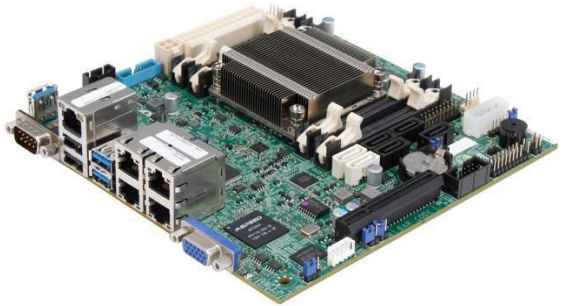
Initial Results: 62-70% reduction



*Removed 152/244 modules

** ~62% of modules

** ~70% of binary



Results II

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

*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