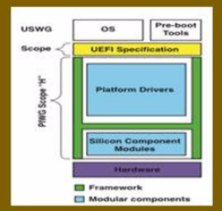
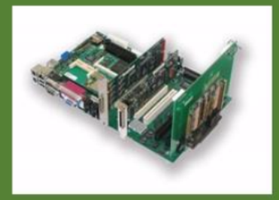
Why UEFI?

Mostly written in C.
High code re-use.

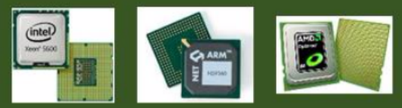


Emphasis on
Specifications.
Standards compliance.

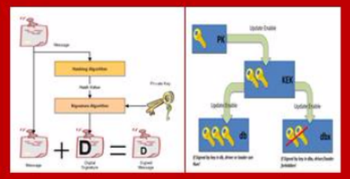


Better platform
scaling. For e.g. removes
shadow ROM limits.

Storage.
GPT removes 2.2 TB
MBR restriction.

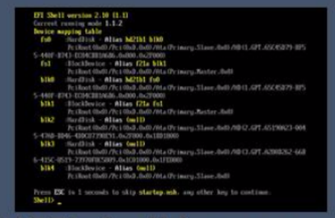


CPU Architecture
independent. Platform
design flexibility.



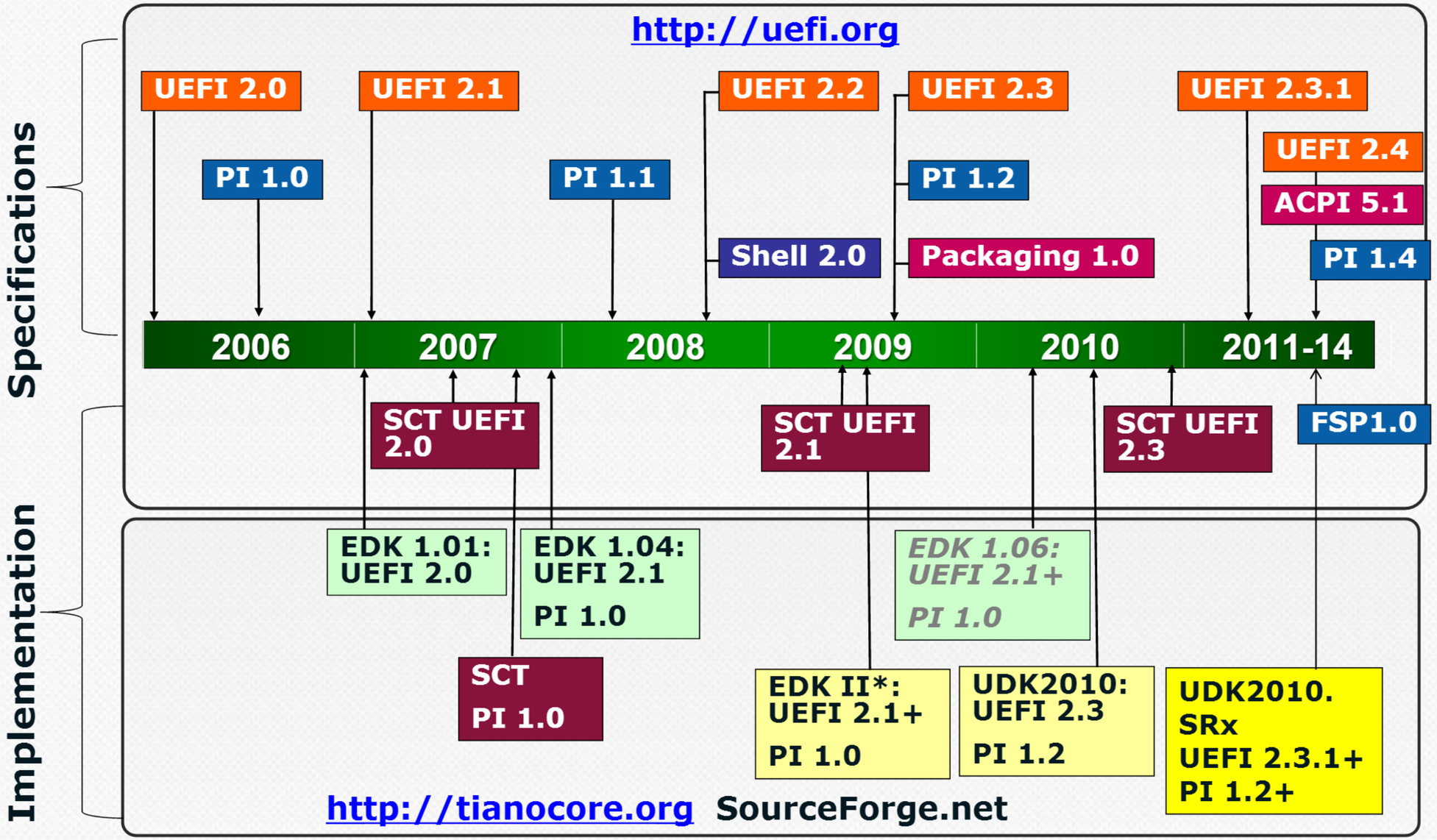
Secure boot solves
“trust” related system
integration challenges.

Pre-boot Networking.
Ipv4, Ipv6, PXE,
VLAN, iSCSI etc.



UEFI shell improves
pre-boot testing &
diagnostics experience.

Specification & Tianocore.org Timeline



All products, dates, and programs are based on current expectations and subject to change without notice.

Building UEFI

Industry Standards Compliance
- UEFI 2.0, UEFI 2.1, UEFI 2.2, UEFI 2.3, UEFI2.4; PI 1.0, PI 1.1, PI 1.2, PI1.3; ACPI 5.1

Extensible Foundation for Advanced Capabilities
- Pre-OS Security
- Rich Networking
- Manageability

Support for UEFI Packages
- Import/export modules source/binaries to many build systems

Maximize Re-use of Source Code**
- Platform Configuration Database (PCD) provides "knobs" for binaries
- ECP provides for reuse of EDK1117 (EDK I) modules
- Improved modularity, library classes and instances
- Optimize for size or speed

Multiple Development Environments and Tool Chains**
- Windows, Linux, OSX
- VS2003, VS2005, WinDDK, Intel, GCC

Fast and Flexible Build Infrastructure**
- 4X+ Build Performance Improvement (vs EDKI)
- Targeted Module Build Flexibility

*** benefit of EDK II codebase*

Maximize the open source at www.tianocore.org

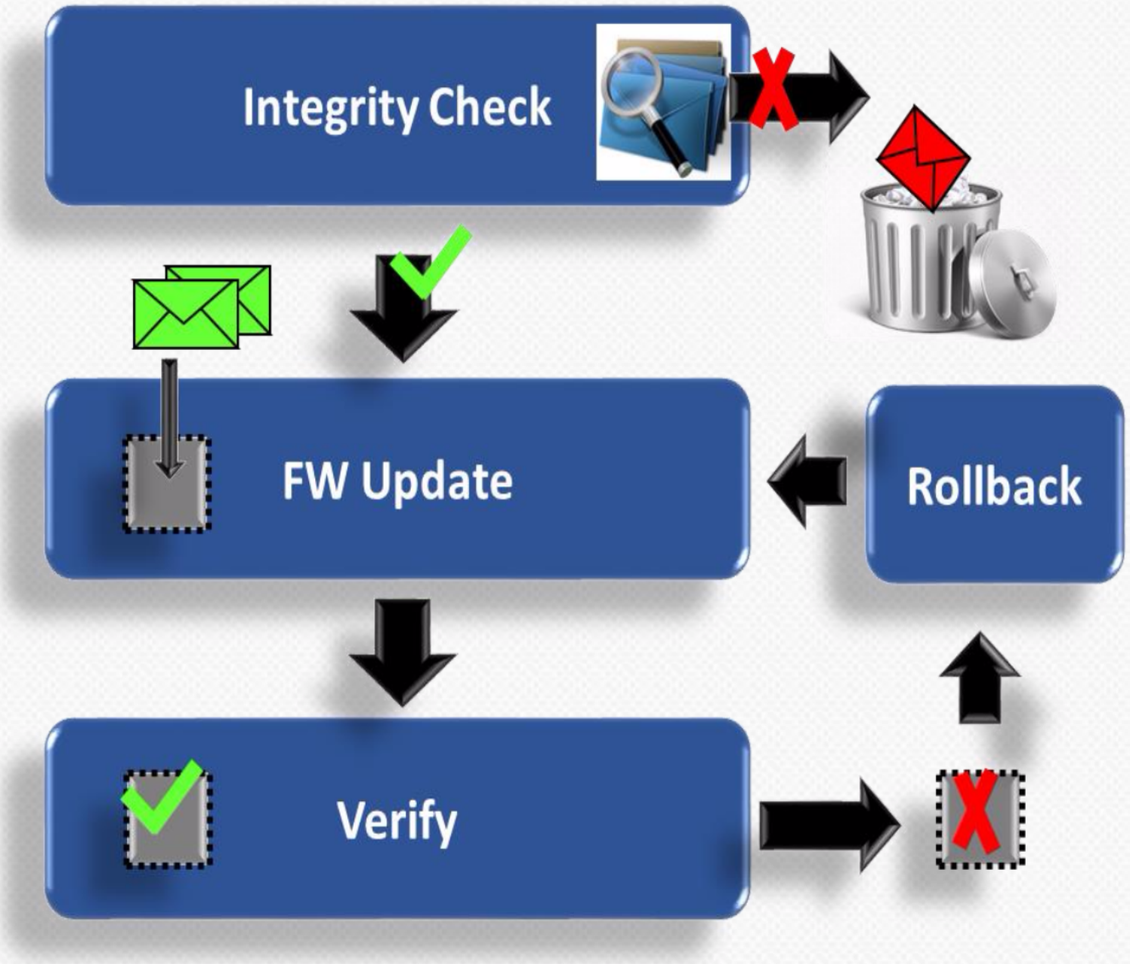
Firmware Updates

Firmware Update Challenges

- Components from multiples vendors
- Delivering firmware
- Different types of devices
- Recovery from failures
- Node equivalence across datacenter
- Security, security, security……
- **Note: BMC based FW updates not covered here**

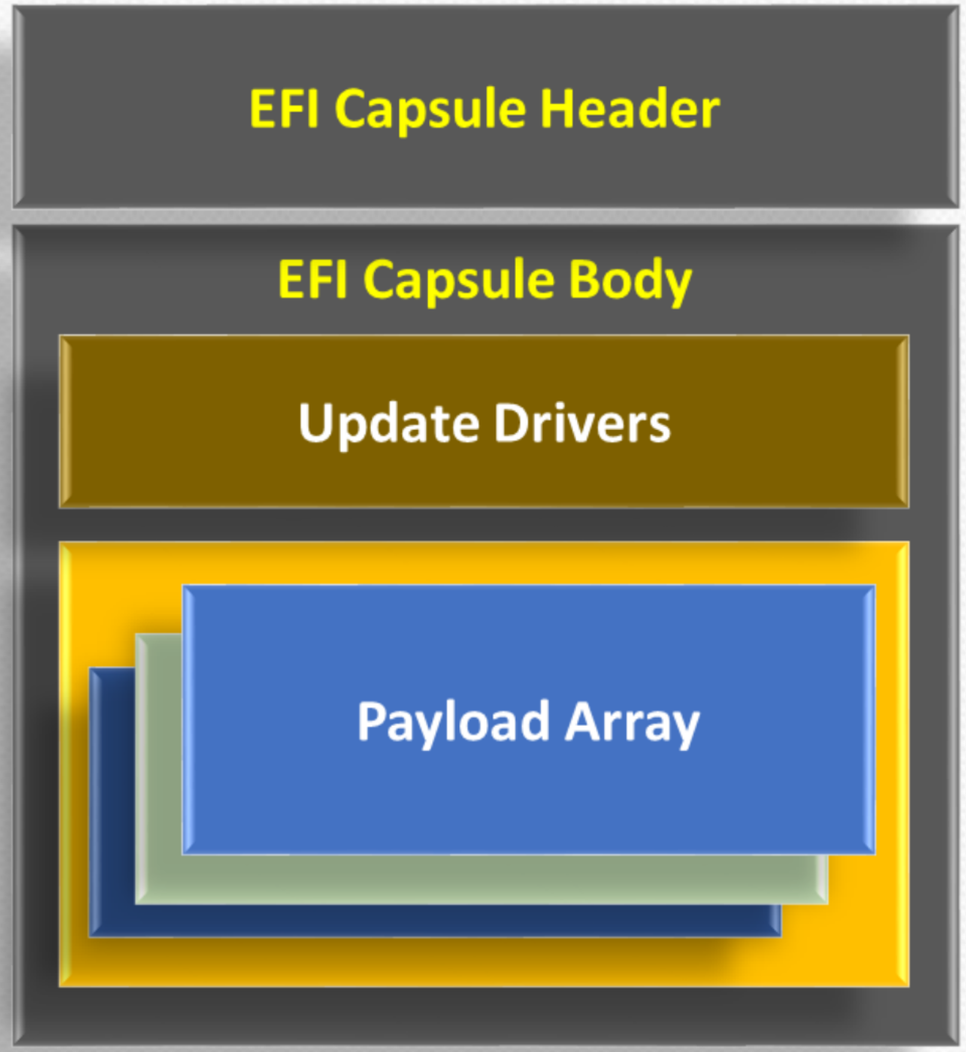
Solving Firmware Update

- Reliable update story
  - Fault tolerant
  - Scalable & repeatable
- How can UEFI Help?
  - Capsule model for binary delivery
  - Bus / Device Enumeration
  - Managing updates via
    - EFI System Resource Table
    - Firmware Management Protocol
    - Capsule Signing



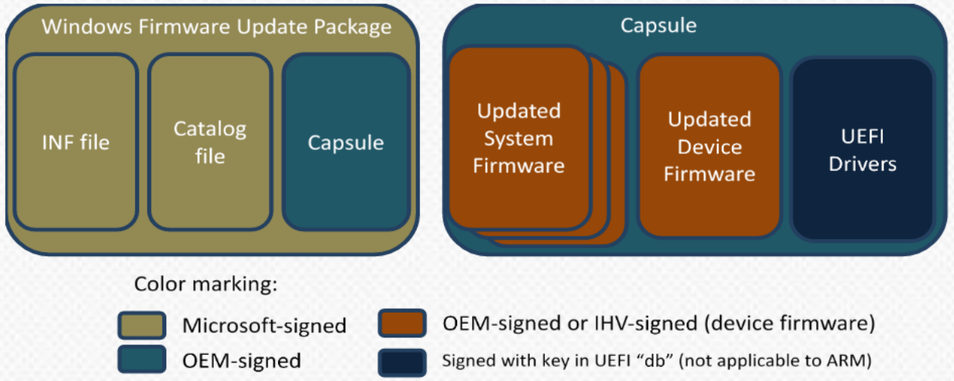
Delivering Firmware Binaries

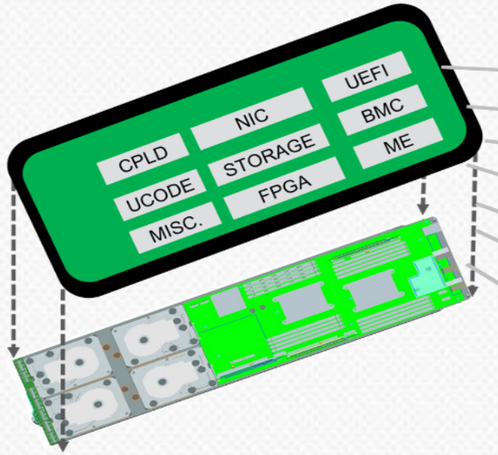
- UEFI supports Capsule format
  - Tools for capsule generation
  - Core logic for capsule handling
- Extensible Capsule format
  - Self-contained
  - Discrete updates
  - Composite updates
- Firmware Management Protocol allows
  - Reading / updating firmware
  - Integrity checks



EFI System Resource Table

- Update types
  - Largely OS assisted
  - Largely BIOS assisted
- FW updateability rules can be encoded into the capsule
  - Least version
  - Signing
- Describe various updateable components on the platform





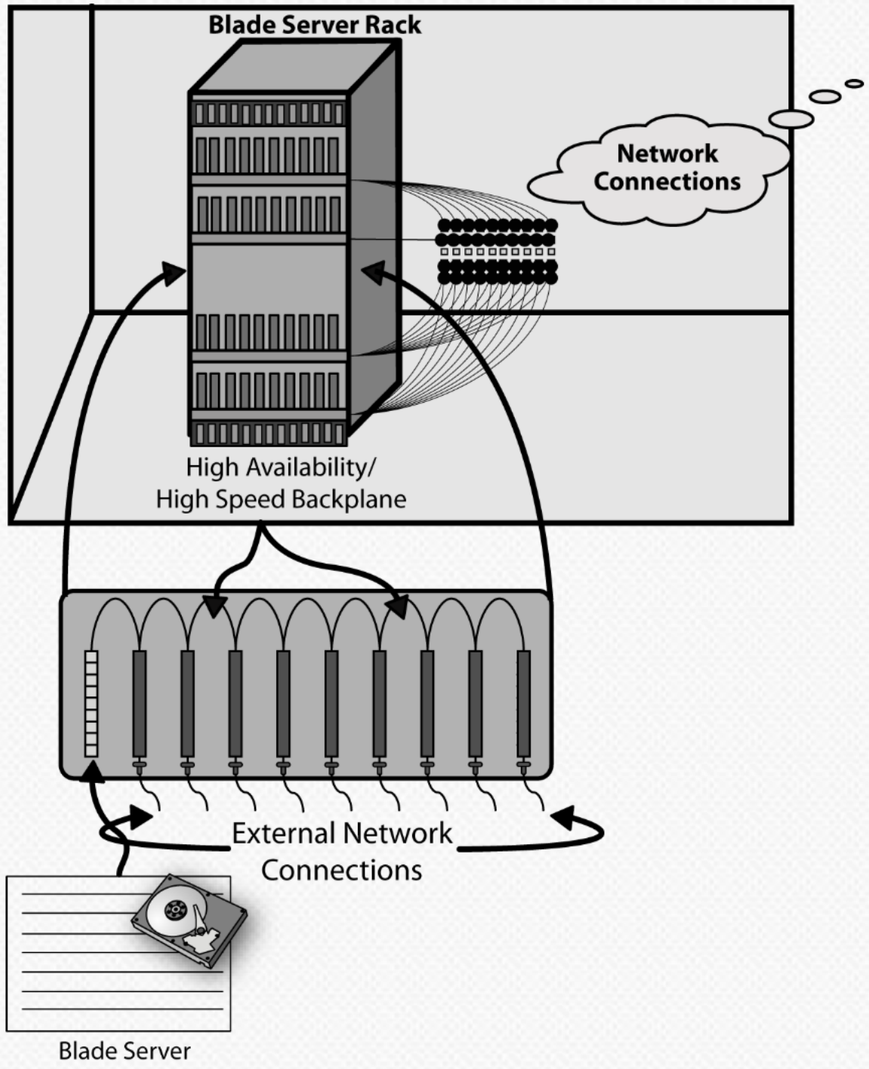
FW	GUID
BIOS	{ $GUID_1$}
BMC	{ $GUID_2$}
ME	{ $GUID_3$}
NIC	{ $GUID_4$}
Storage FW	{ $GUID_5$}
UCODE	{ $GUID_6$}
CPLD	{ $GUID_7$}
Etc..	{ GUID..}

Example ESRT

Bare Metal Provisioning

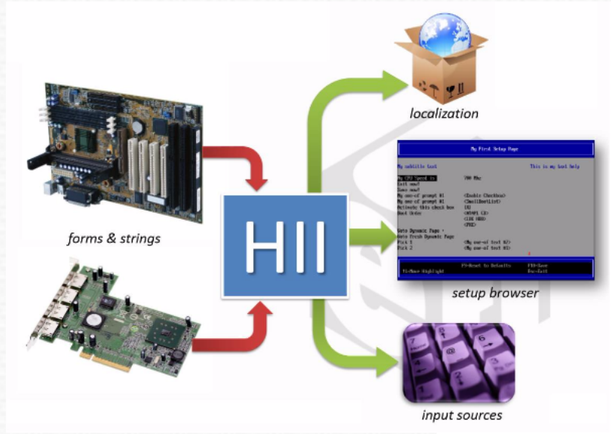
Bare Metal Provisioning Challenges

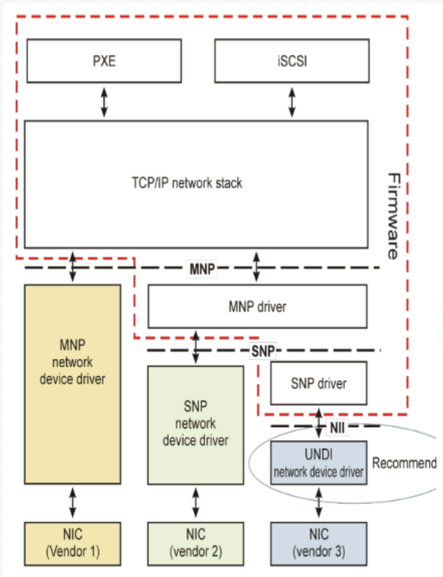
- Hardware Detection
- Installation
  - Local / Remote
- Configuration
  - Local / Remote / Scriptable
- Cloning
  - Automated
- Backup / Recovery
  - Local / Remote / Automated

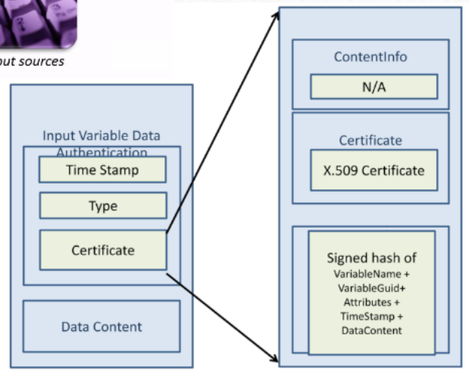


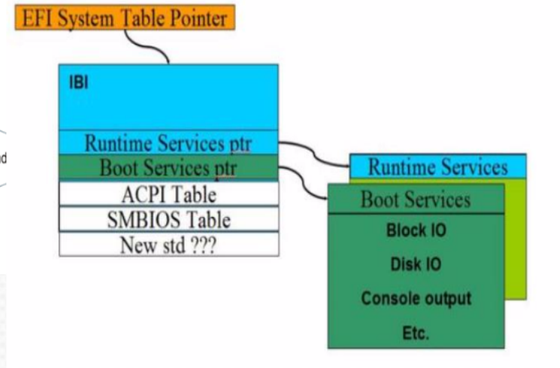
Bare Metal Provisioning Solutions

- Need a ‘no-touch’, automated installation mechanism
  - Repurpose / Configure / Recover
- HII and IFR for consistent & scriptable configuration
- Non-blocking local disk and networking services for high throughput image delivery and recovery
- UEFI Variables for booting and Authenticated Variables for safe storage of settings, like UEFI secure boot database



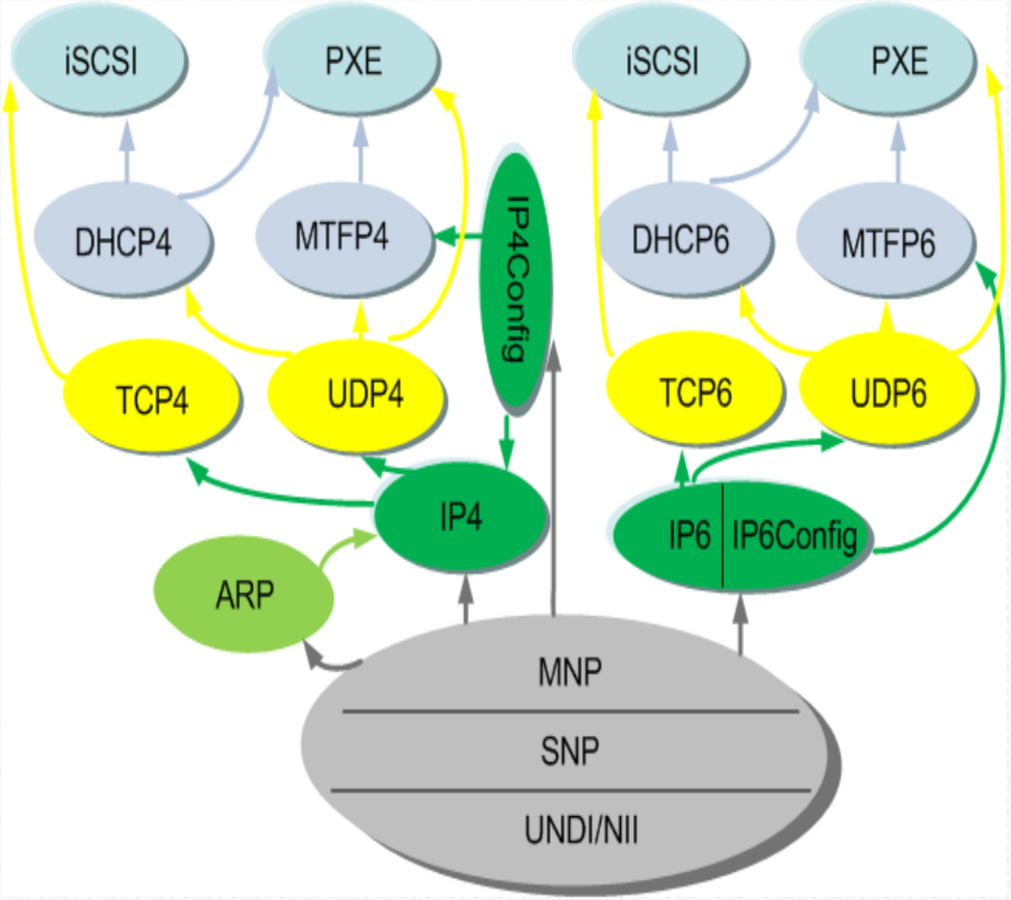






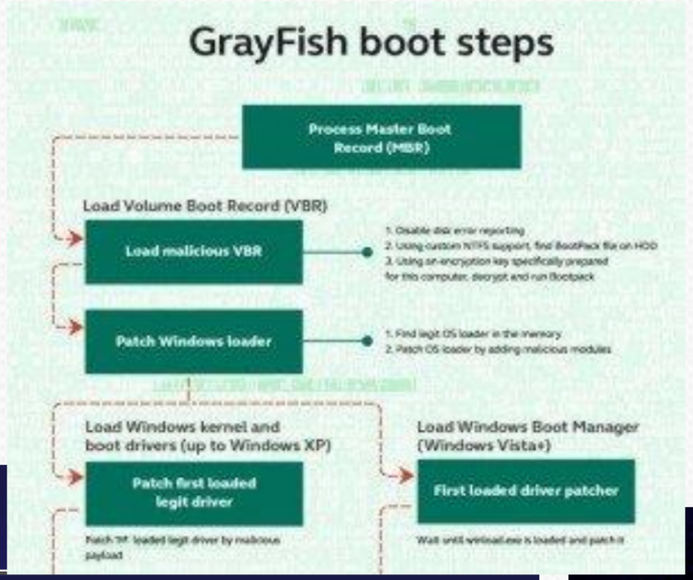
Networking in UEFI

- UEFI offers rich set of Networking Features during pre-boot
  - PXE boot support for network boot, OS installations, provisioning etc.
  - Native support for IPv4 as well as IPv6
  - Network file system support
  - Virtual LAN support, iSCSI
  - IpSec for supporting secure communication
- Evolution of networking –
  - RFC 5970 allows for ‘boot from URI’
    - HTTP, NFS,…



Security

Attacks on Firmware



DE MYSTERIIS DOM JOBSIVS: MAC EFI ROOTKITS

SNARE
@ SYSCAN SINGAPORE
APRIL 2012

assurance

IN CONCLUSION... I HAD FUN.

- So basically we're all screwed
  - What should you do?
    - Glue all your ports shut
    - Use an EFI password to prevent basic local attacks
    - Stop using computers, go back to the abacus
  - What should Apple do?
    - Implement UEFI Secure Boot (actually use the TPM)
    - Use the write-enable pin on the firmware data flash properly
      - NB: They may do this on newer machines, just not my test one
    - Audit the damn EFI code (see Heasman/ITL)
    - Sacrifice more virgins

De Mysteriis Dom Jobsivs - SyScan April, 2012

Hacking the Extensible Firmware Interface

John Heasman, Director of Research

Code Injection Attacks

- Important when firmware verifies digital signatures
  - Depends on implementation flaw in driver
  - e.g. stack overflow, heap overflow
  - or incorrect signature verification
- Plenty of targets:
  - File system drivers (e.g. FAT32, HFS+)
  - PE parsing code
  - Crypto code (Data in certs, ASN.1 decoding)
  - Network interaction (PXE)

Network Nightmare

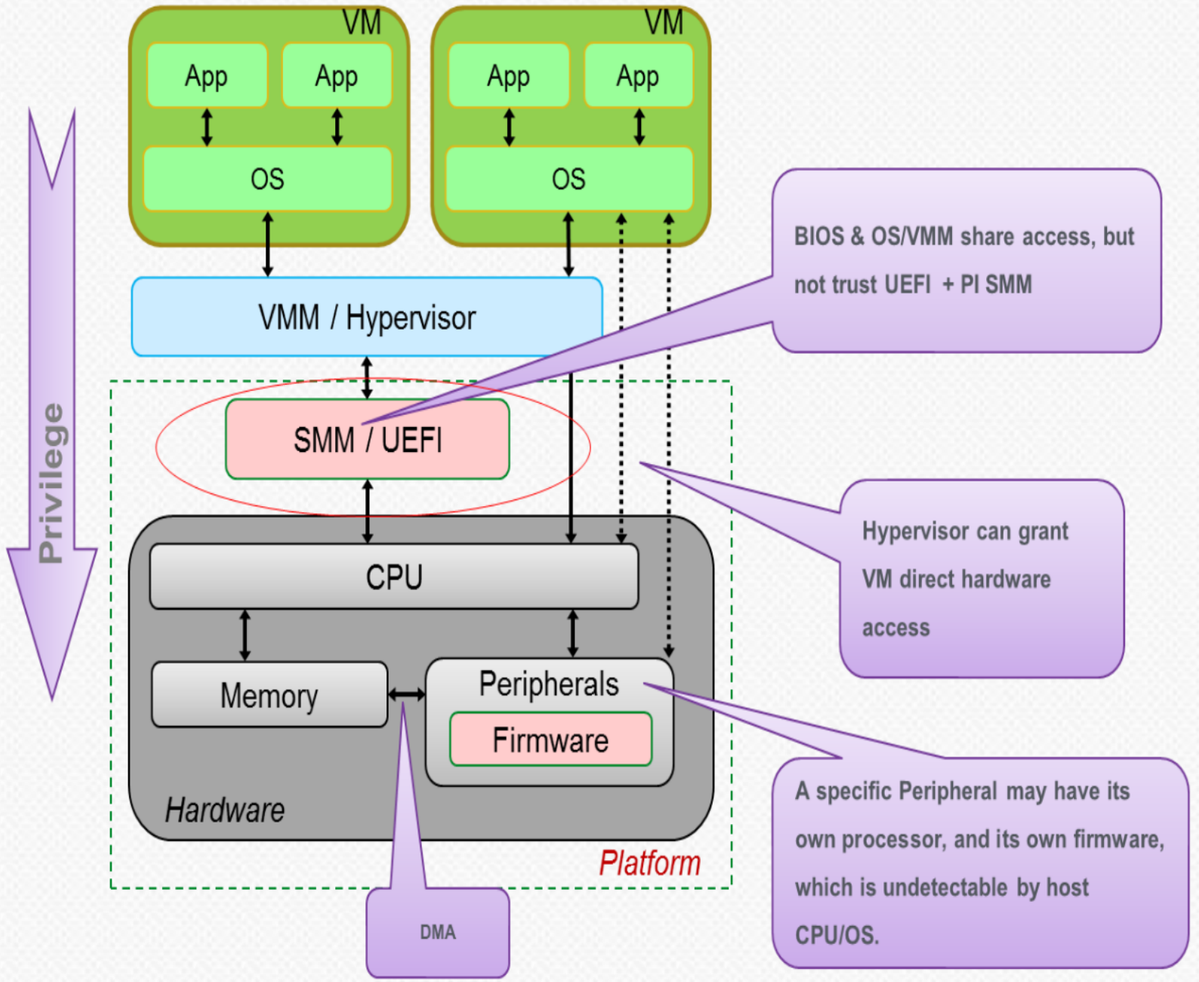
Ruling the nightlife between shutdown and boot with pxesploit

Bootkits

Stephen Cobb, senior security researcher at ESET North America, says that hacking firmware can be particularly effective because it is so hard to eliminate.
It's also particularly challenging to do, says Jean Taggart, security researcher at Malwarebytes.
"Doing this on just one brand of hard drive would be an almost Herculean task," he says. "You have to understand the hardware as well–if not more–than the original manufacturer."
– Stan Alcorn, Marketplace. Feb 17, 2015

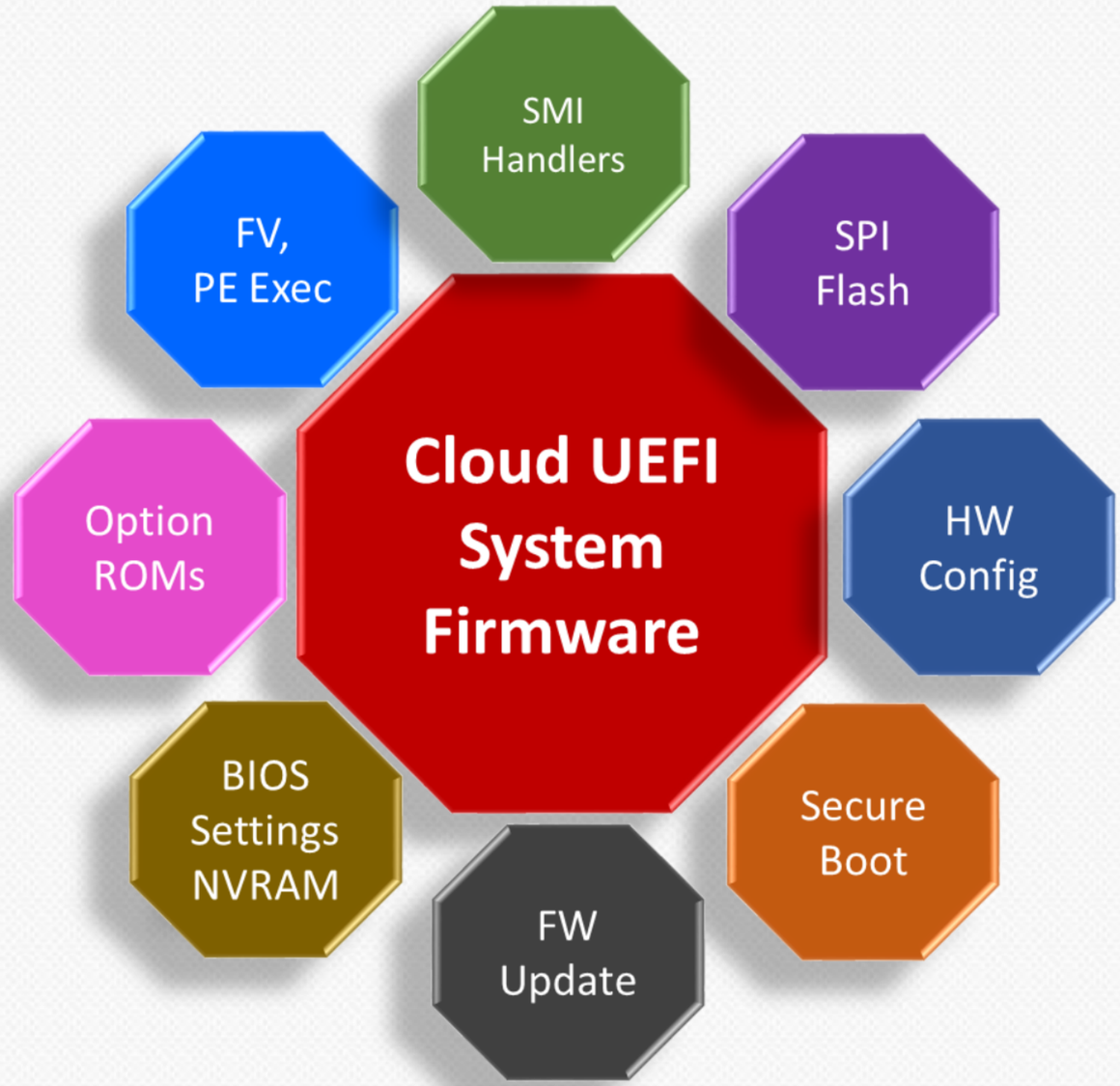
Security Challenges

- Different elements in platform from many vendors
- How to establish trust anchor in the hardware
- How to protect elements
- How to protect the platform
- How to allow platform scaling



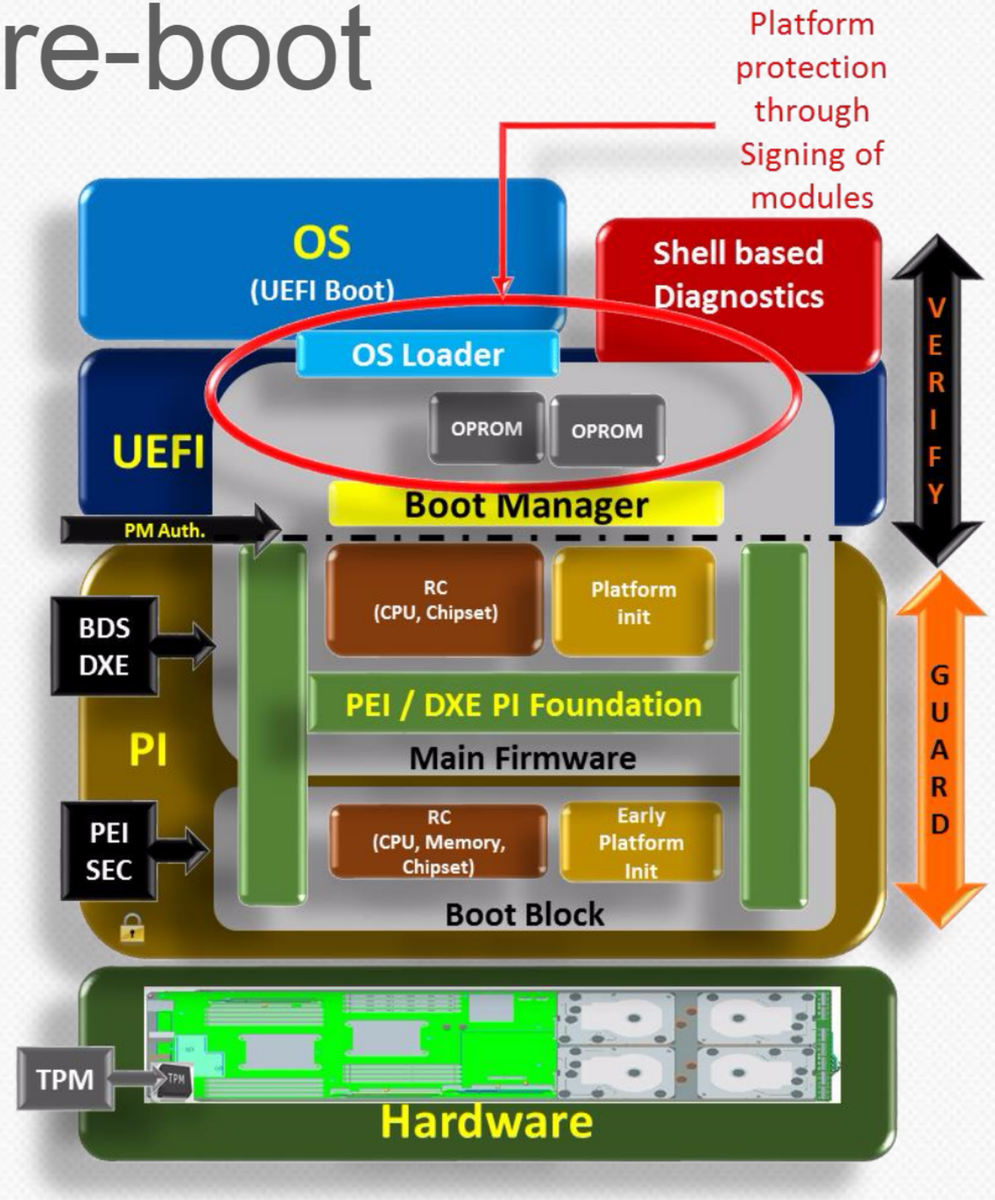
Security Solutions

- Signed capsule updates
- UEFI Secure boot
  - local / network
- TPM on the platform
  - Measured boot
  - Root of Trust for Reporting
  - Storage
- Protect machine configuration & UEFI Secure boot trust anchors
- In-band and out-of-band network security

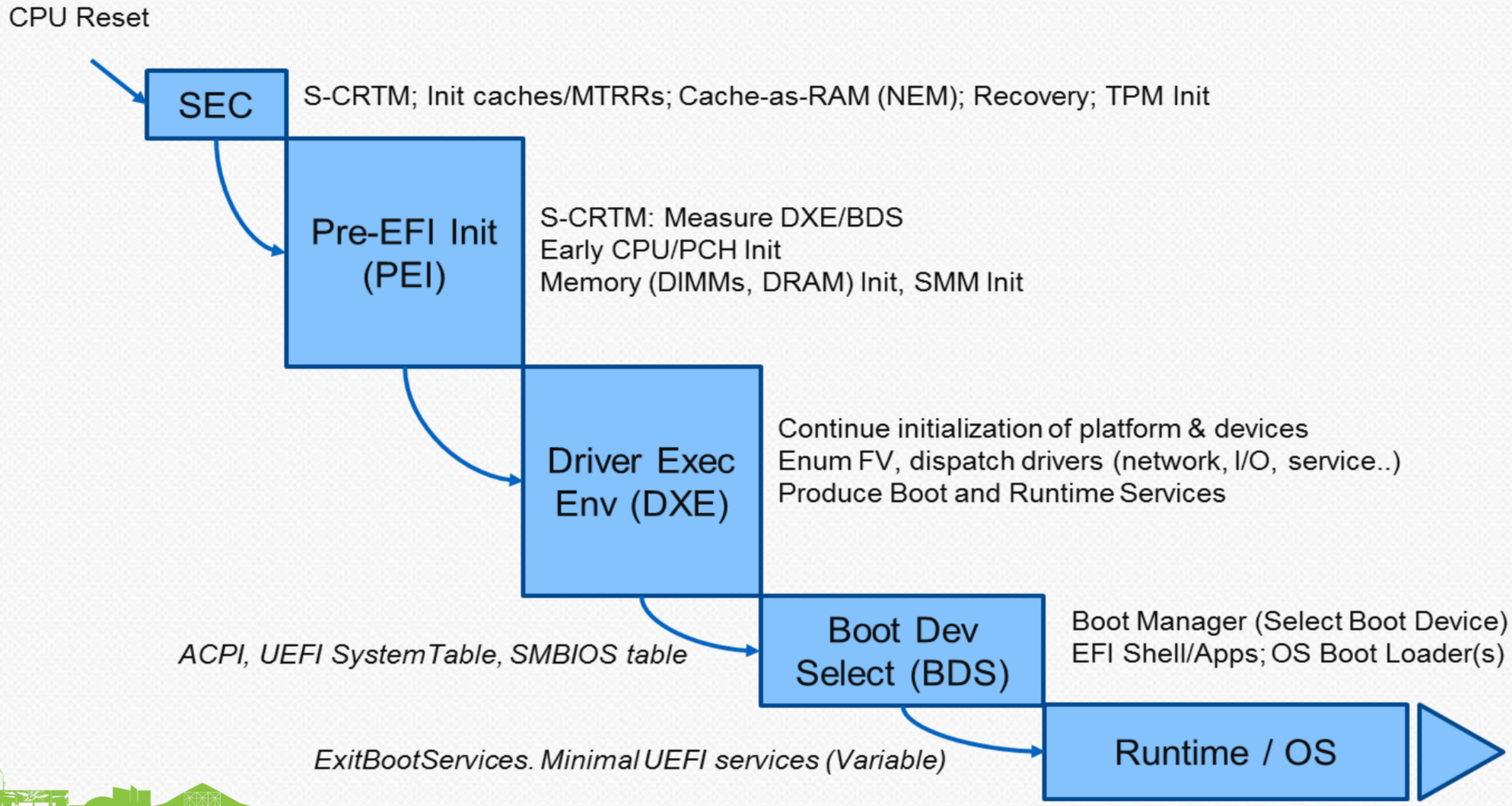


Guarding & Verifying in Pre-boot

- PI & UEFI complement each other to impart **platform security** through guarding and verification during pre-boot.
- PI facilitates **platform hardening** by guarding internal firmware ingredients that consume reset vector, initialization of CPU, Memory, Chipset etc.
- UEFI signing allows **robust platform scaling** through verified inclusion of external firmware ingredients such as OPROMS into the trust chain



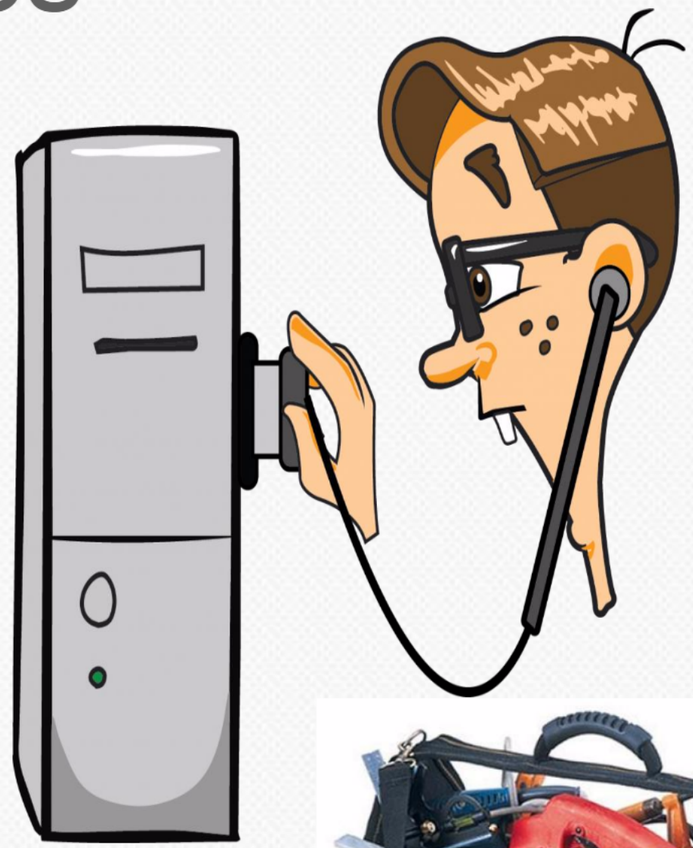
Recommended UEFI Boot Flow



Tools & Diagnostics

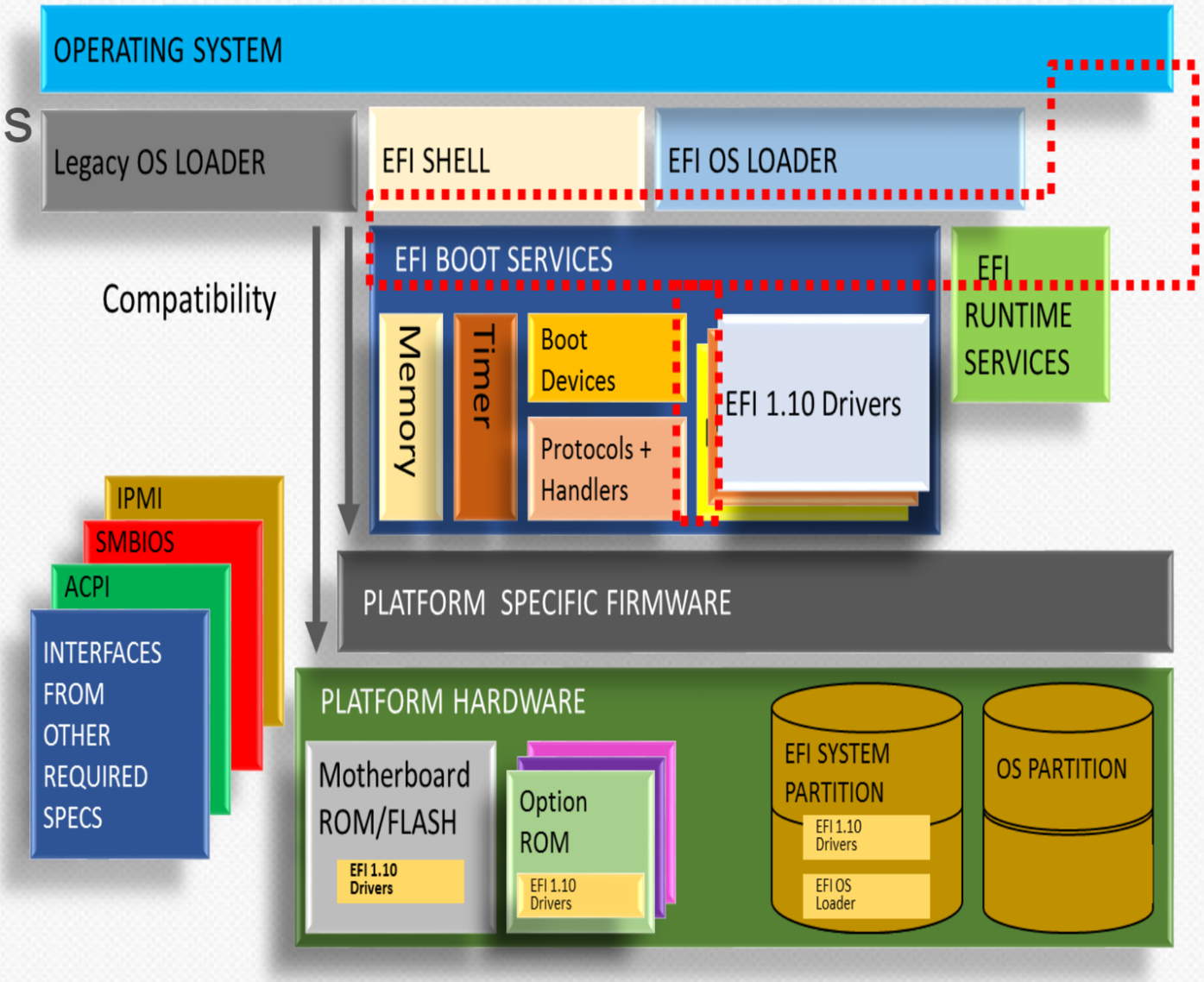
Tools & Diagnostic Challenges

- Platform ingredients from many vendors
- How to assess health, security, compliance of the elements
- Consistent environment to run diagnostics
  - Log / Report / Journal results
- Recovery agent considerations
  - Local / Remote / In-band / Out-of-band



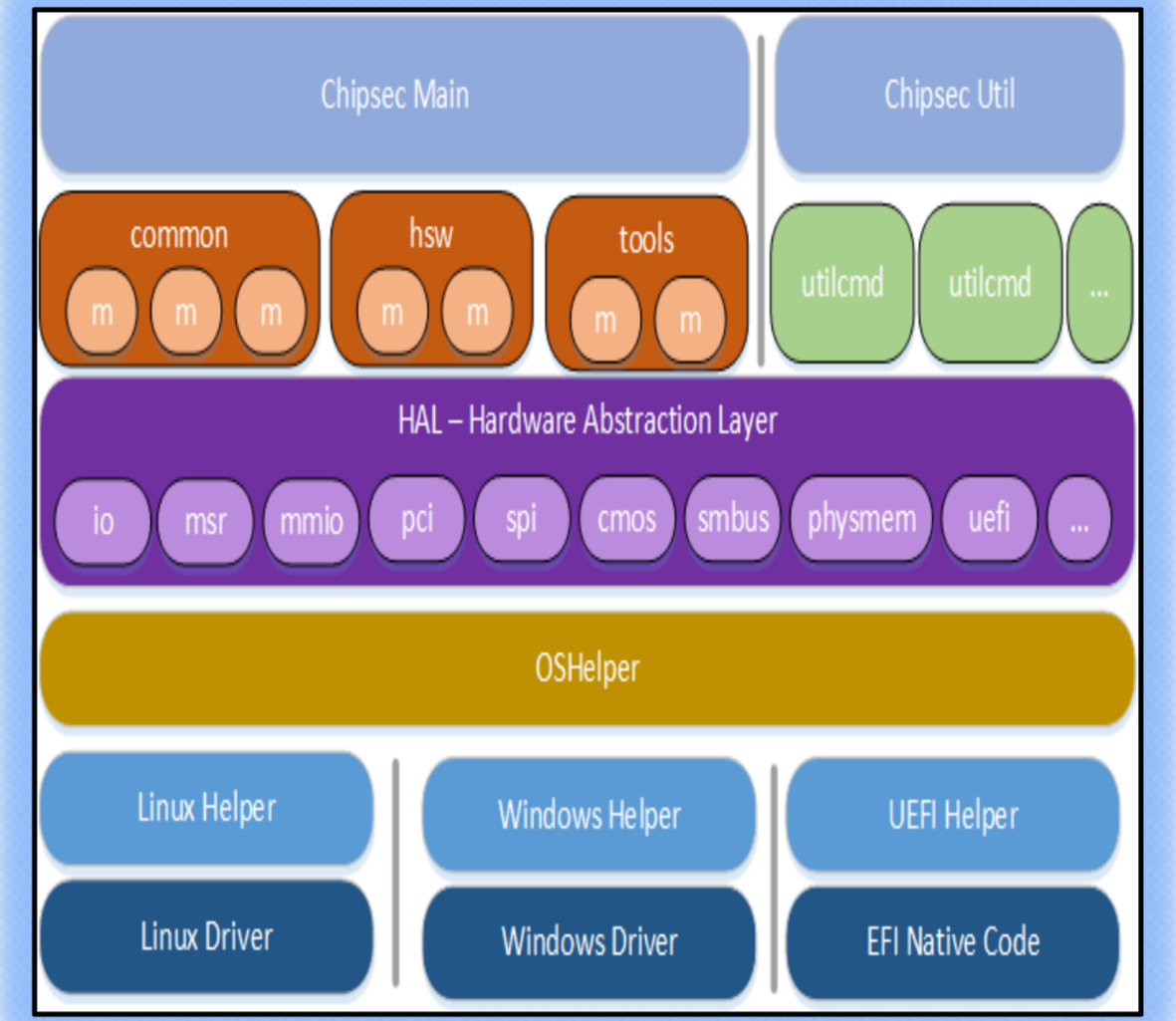
Tools Solutions

- Environment for hosting tools
  - UEFI Shell
  - Linux UEFI Validation project
- Tools for deployment
  - UEFI SCT
  - PI SCT
  - ACPI Compliance
  - SMBIOS Compliance
  - Security
    - Chipsec
    - Copernicus
  - Selftest



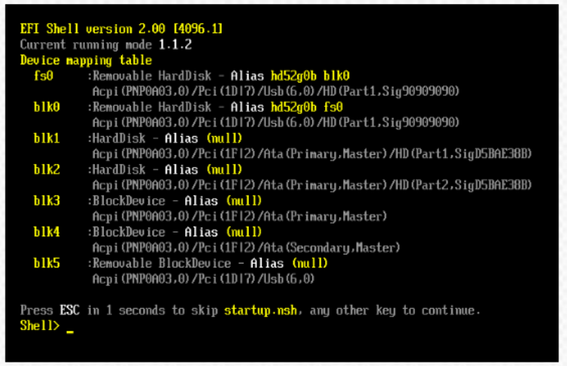
chipsec Tool

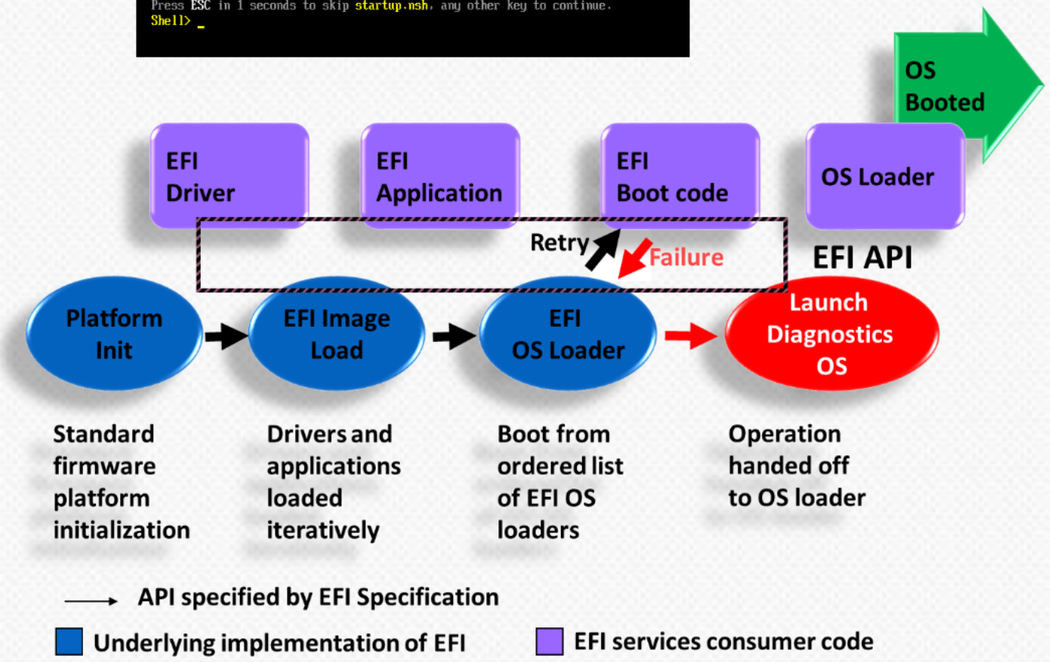
- Essentially a platform security assessment framework for risk assessment
- Can be extended to meet specific platform security concerns
- Open sourced
  - https://github.com/chipsec/chipsec
- Supported Environments
  - Windows
  - Linux
  - UEFI (over Python)



Diagnostics Solutions

- Once in UEFI, how to assess, probe, and prod the system
  - Type15 SMBIOS Records
  - Dmpstore for UEFI variables, incl WHEA variable
  - ACPI CA for executing/dumping/viewing namespace
  - UEFI shell to run above, redirect output to file or ‘virtual file’ (e.g., volatile variable)
  - PCI command to read/write/assess hardware state. Scriptable too
  - Results can be installed in UEFI system table like other hand-off info, or variable, or file on ESP, or sent across the network using UEFI network stack





Conclusions & Next Steps

Call to actions

Get involved in the standards

If you’re an IHV, implement FMP, Capsules, Security reviews of code

If you’re an OSV, build UEFI loader and boot agent

If you’re middle ware, leverage HII and shell scripting/config

Conclusion

Cloud has challenges for platform

UEFI for interop

Evolve updates

Provisioning

Diagnostics

Security

More information

www.opencompute.org – OCP specs

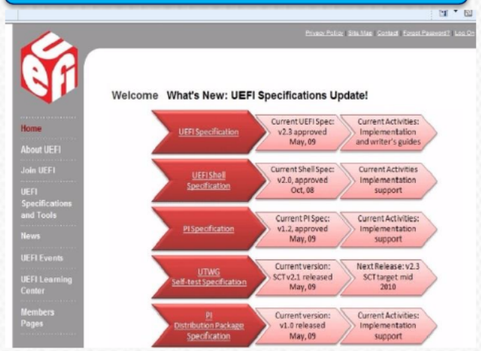
www.uefi.org – UEFI, ACPI, Shell, PI Specifications

www.Tianocore.org – open source UEFI

http://firmware.intel.com – white papers, training

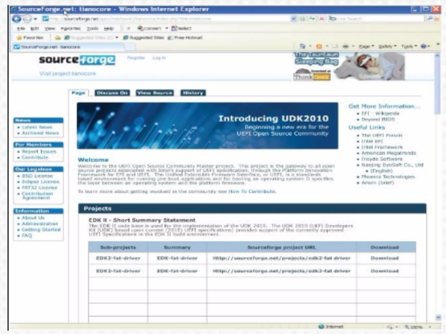
UEFI Industry Resources

UEFI Forum



www.uefi.org

UEFI Open Source



www.tianocore.org

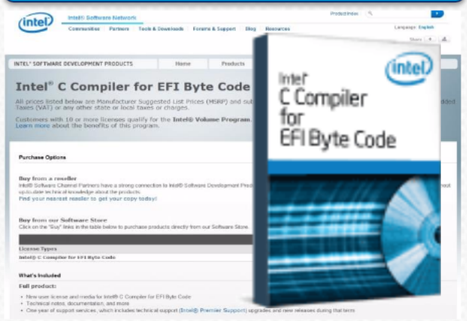
Intel UEFI Resources

Firmware.intel.com

Open Compute Resources

www.opencompute.org

Intel EBC Compiler



http://software.intel.com/en-us/articles/intel-c-compiler-for-efi-byte-code-purchase/

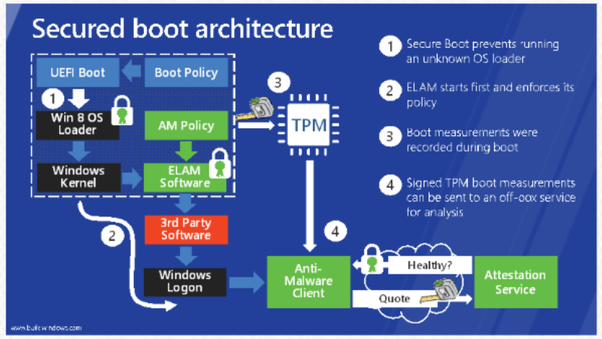
UEFI Books/ Collateral

www.intel.com/intelpress

http://www.intel.com/content/www/us/en/research/intel-technology-journal/2011-volume-15-issue-01-intel-technology-journal.html

http://www.apress.com/apressopen

Microsoft UEFI Resources

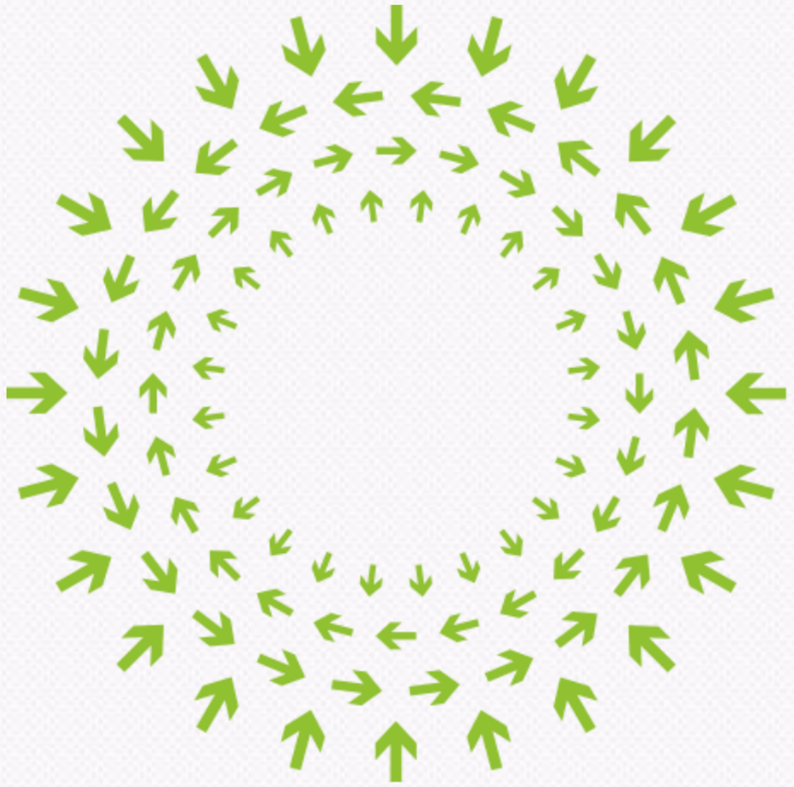


http://channel9.msdn.com/Events/BUILD/BUILD2011/HW-462T

https://msdn.microsoft.com/en-us/library/windows/hardware/hh973604.aspx

https://technet.microsoft.com/en-us/library/hh824898.aspx

Windows



OPEN

Compute Engineering Workshop

March 9, 2015

San Jose