Developing Best-In-Class Security Principles with Open Source Firmware

Vincent Zimmer
Senior Principal Engineer Intel Corporation

STTS003

Agenda

- Problem Statement
- Ingredients
- System Management Mode (SMM)
- Open Platforms

Agenda

- Problem Statement
- Ingredients
- System Management Mode (SMM)
- Open Platforms

Platform Threats

Platform Threats

Bootkits

Platform Threats

Bootkits

Evil Maid

Platform Threats

BIOS Malware

Bootkits

Evil Maid

Platform Threats

BIOS Malware

Bootkits

SMM Rootkits

Evil Maid

Platform Threats

BIOS Malware

UEFI Rootkits

Bootkits

SMM Rootkits

Evil Maid

Platform Threats

BIOS Malware

UEFI Rootkits

Bootkits

SMM Rootkits

Evil Maid

HVM Rootkits (Blue Pill)

Platform Threats

BIOS Malware

UEFI Rootkits

Bootkits

SMM Rootkits

Device FW Malware

ACPI Rootkits

Option ROM Malware

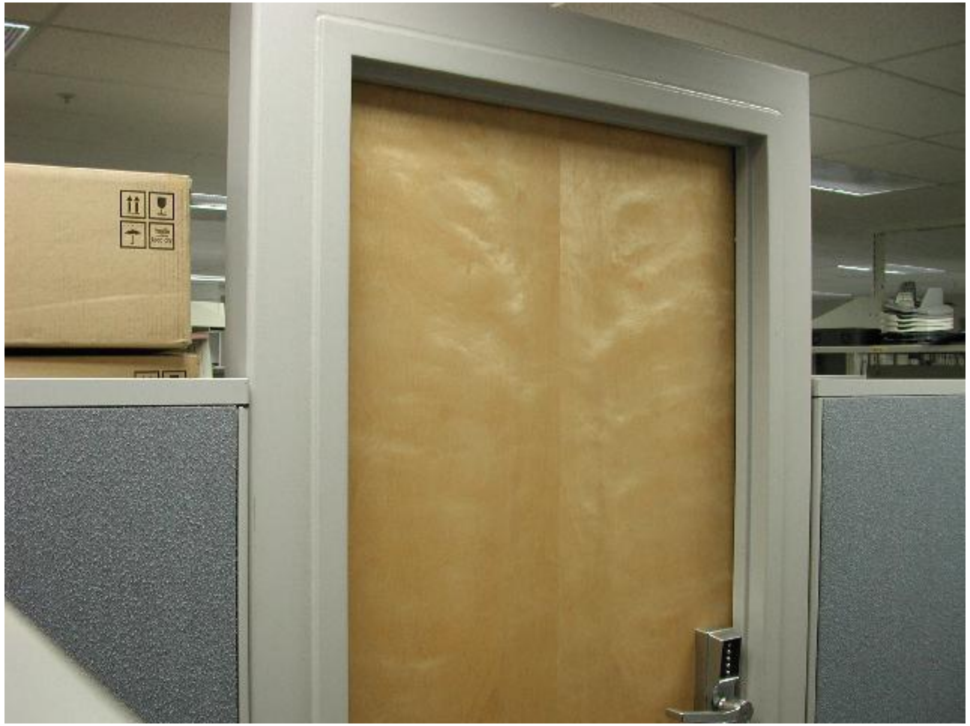
Evil Maid

HVM Rootkits (Blue Pill)

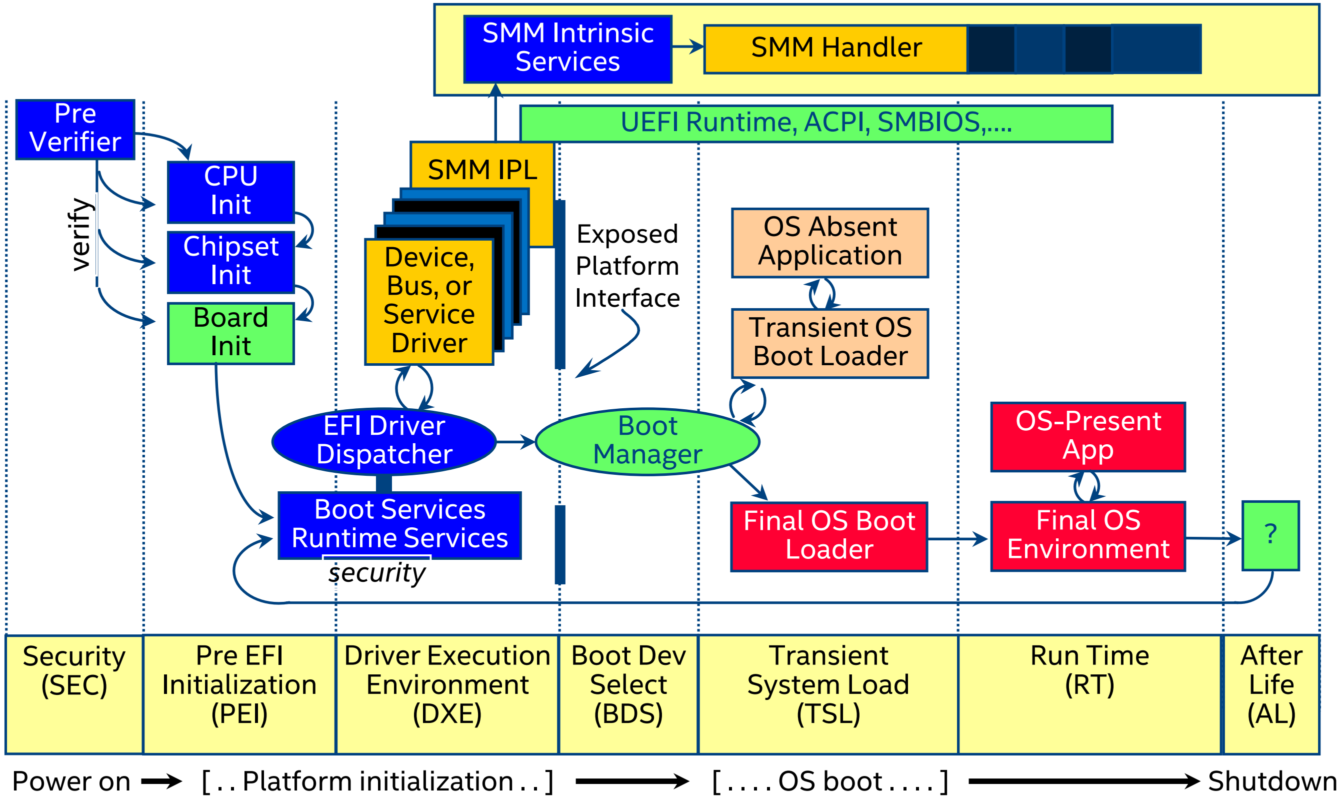
HW Trojans

Security Fundamentals

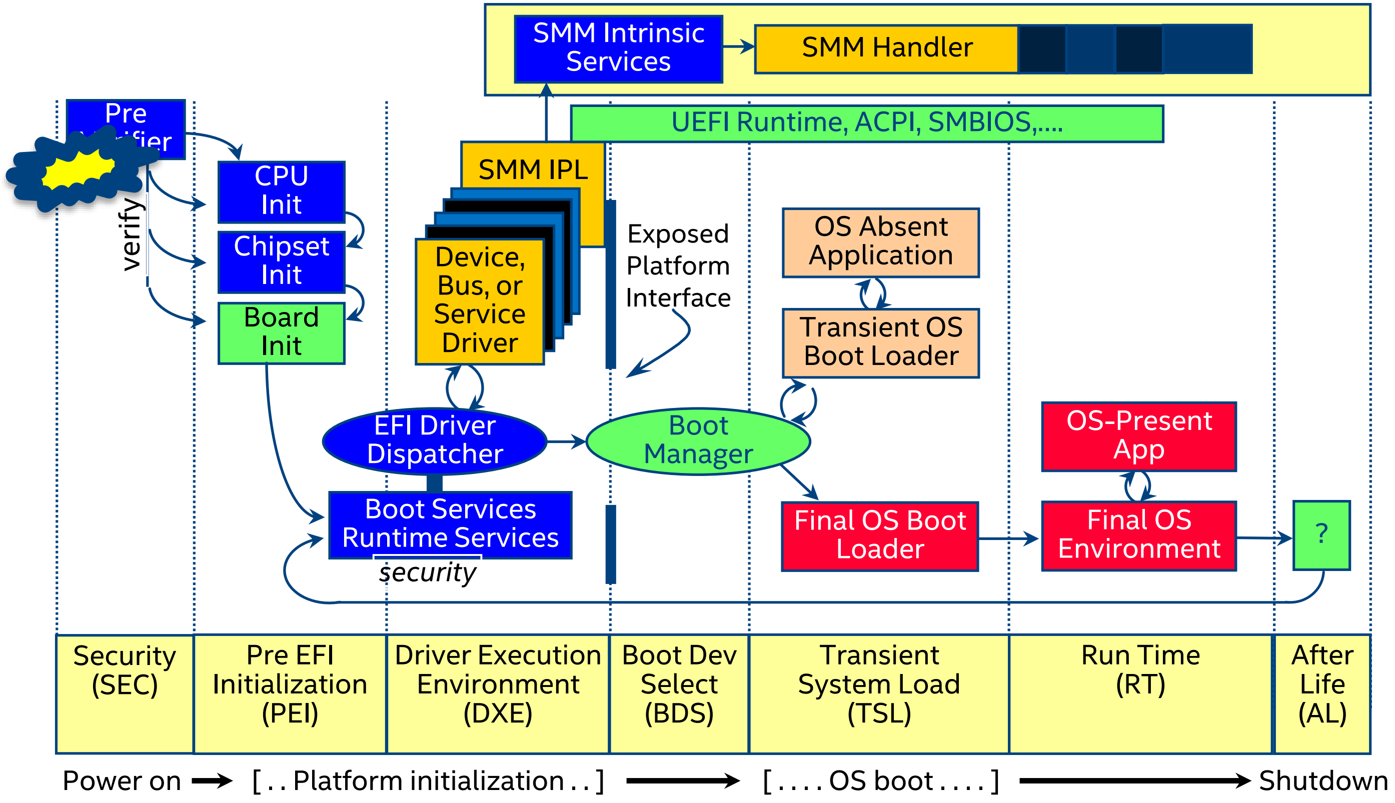
Security Fundamentals



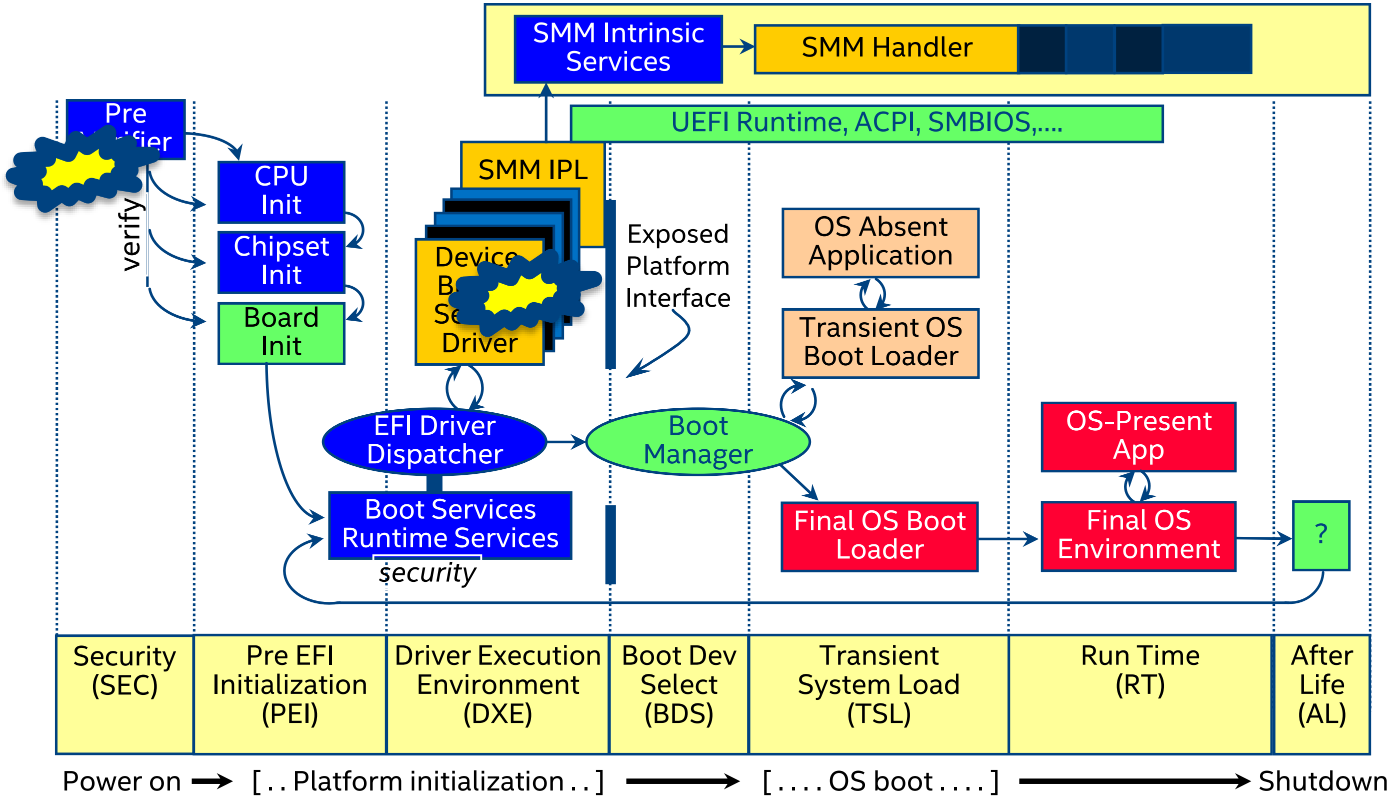
What Could Possibly Go Wrong???



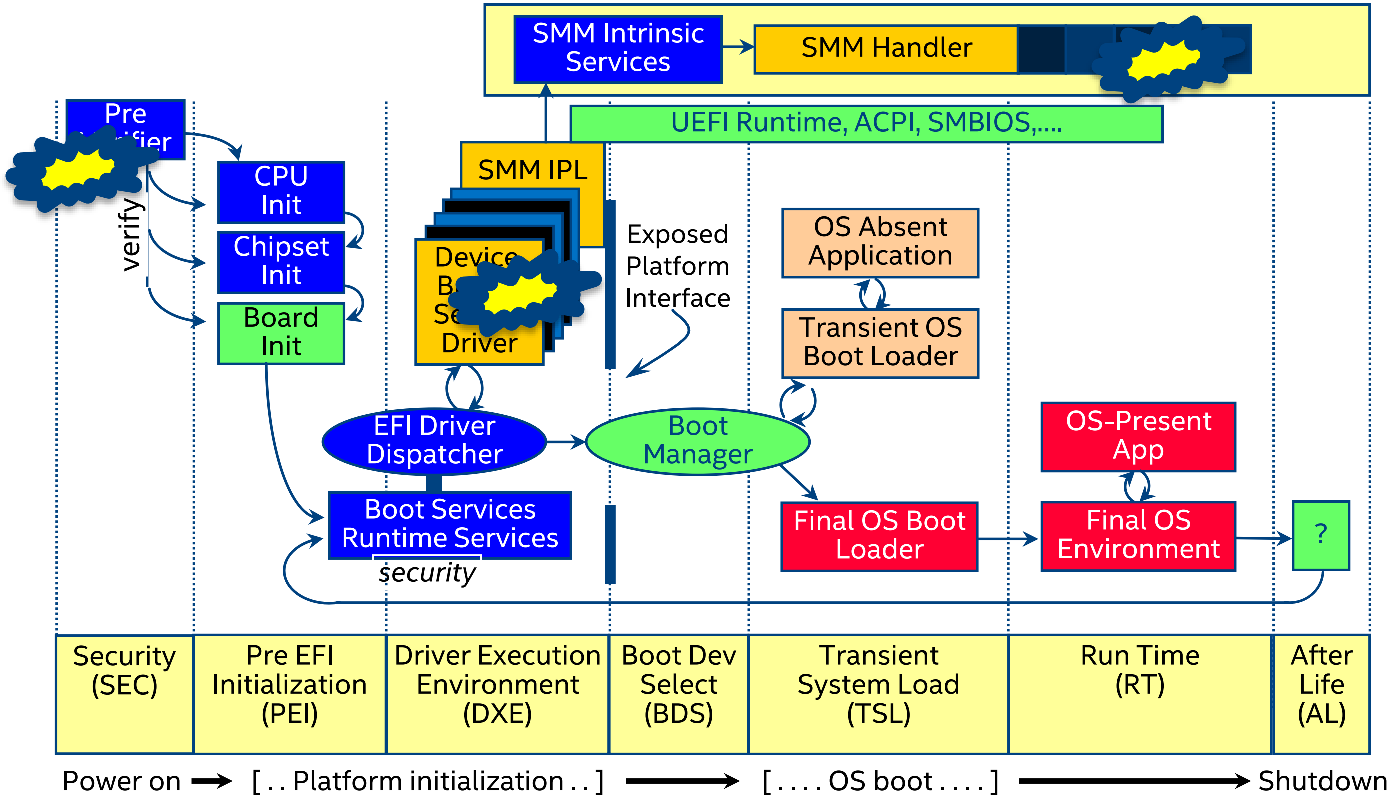
What Could Possibly Go Wrong???



What Could Possibly Go Wrong???

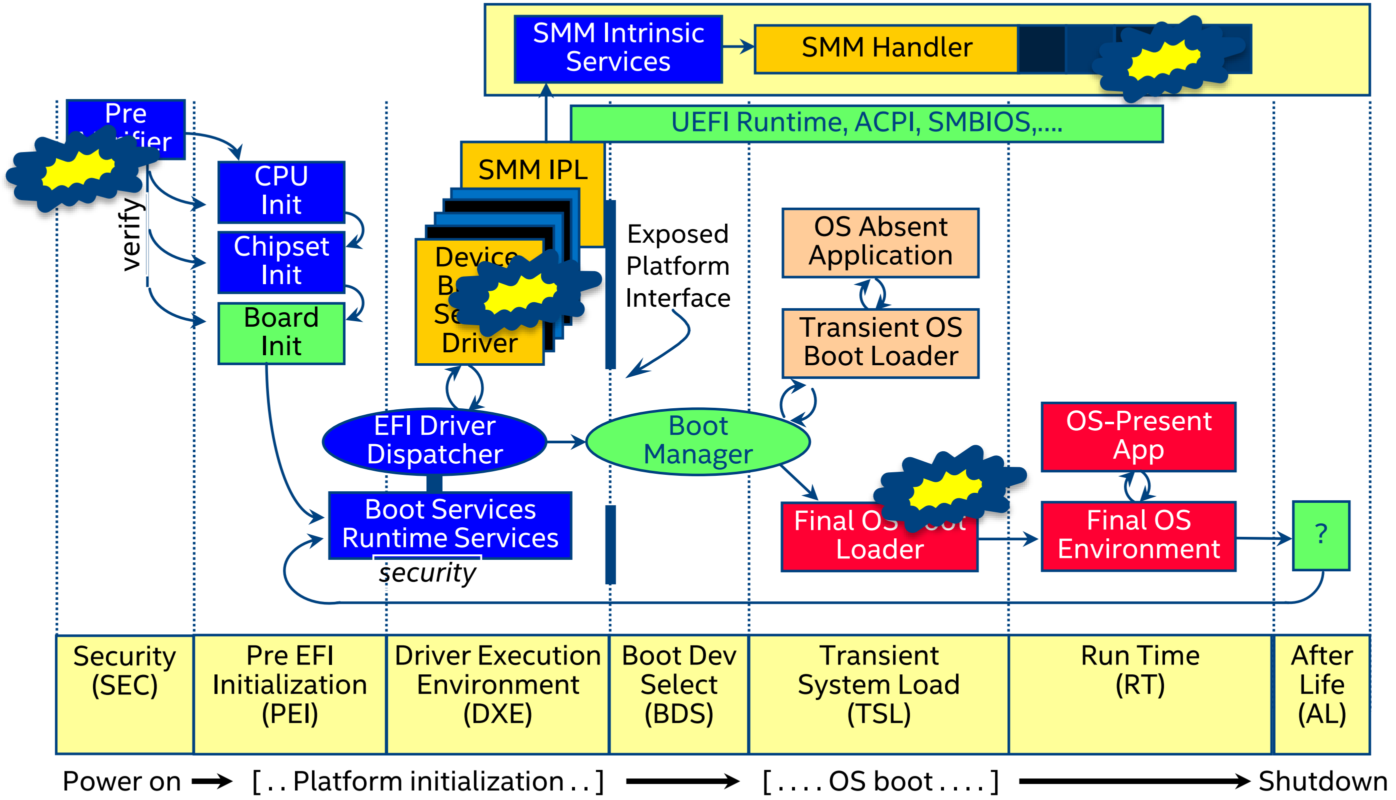


What Could Possibly Go Wrong???



IDF15 INTEL DEVELOPER FORUM

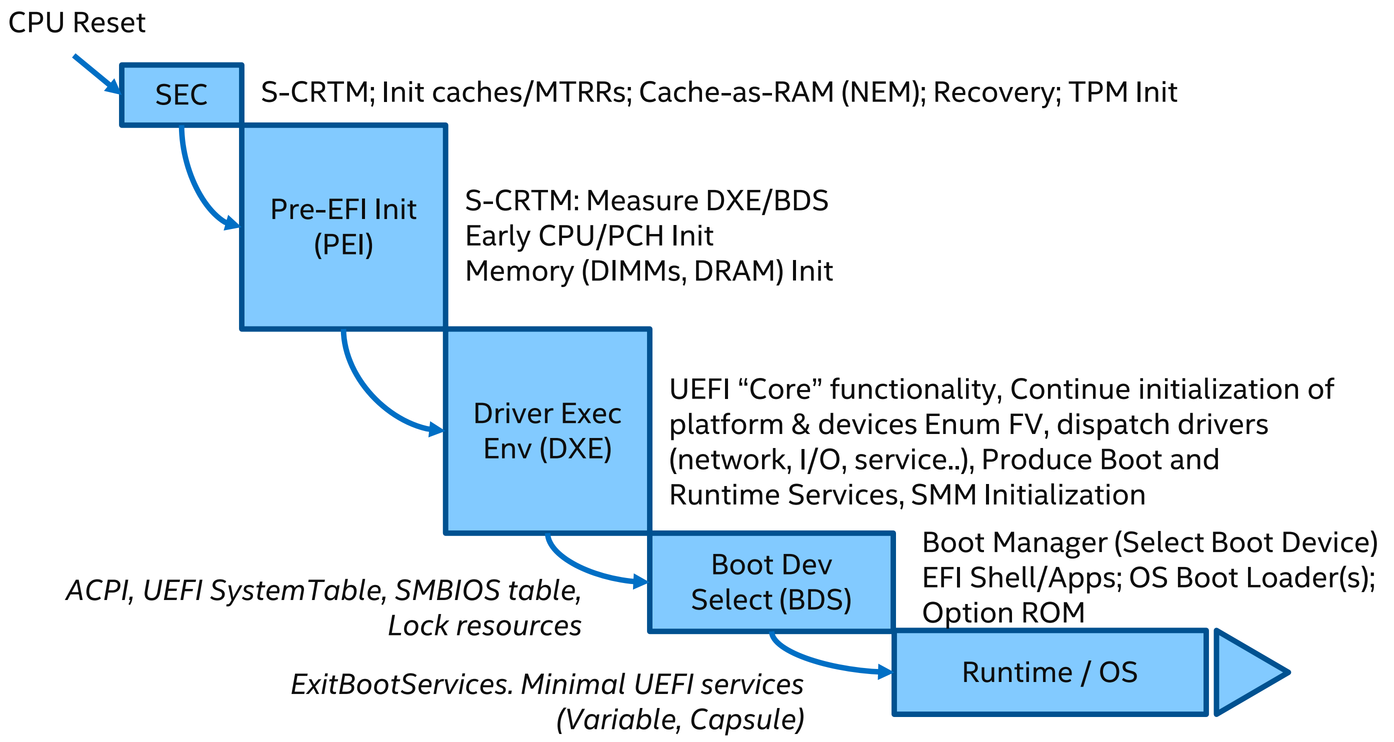
What Could Possibly Go Wrong???

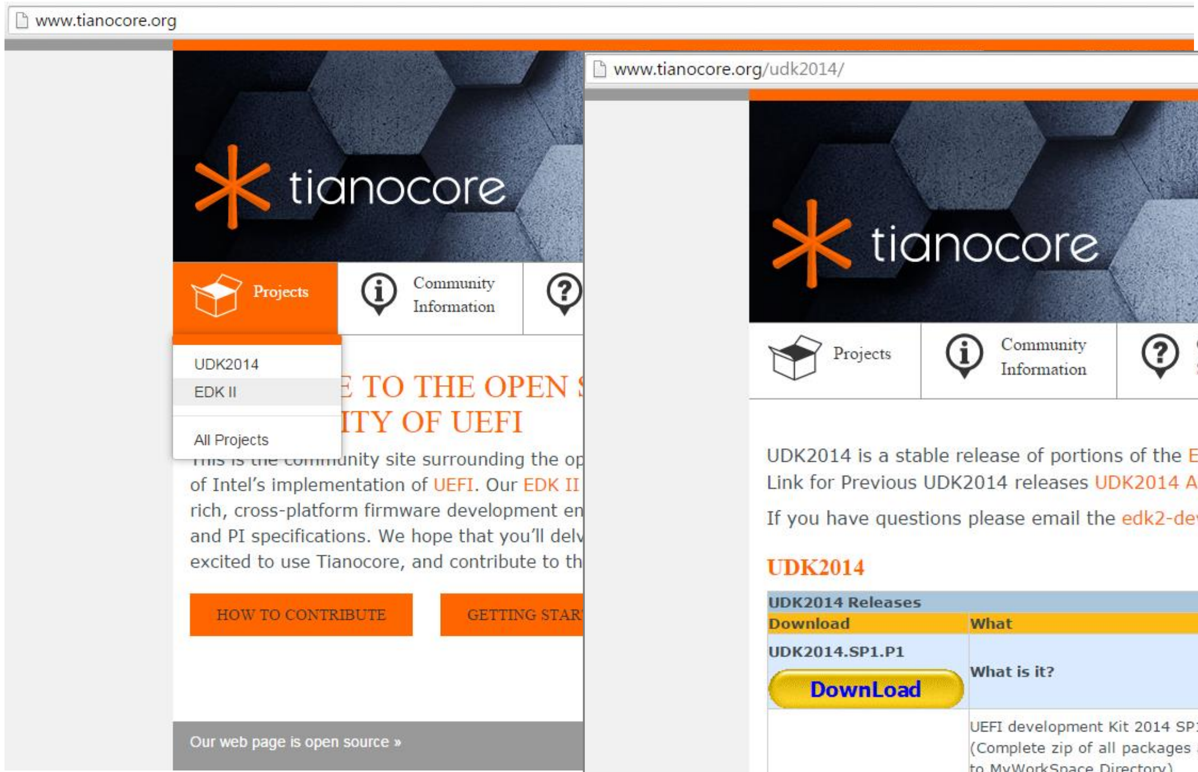


Agenda

- Problem Statement
- Ingredients
- System Management Mode (SMM)
- Open Platforms

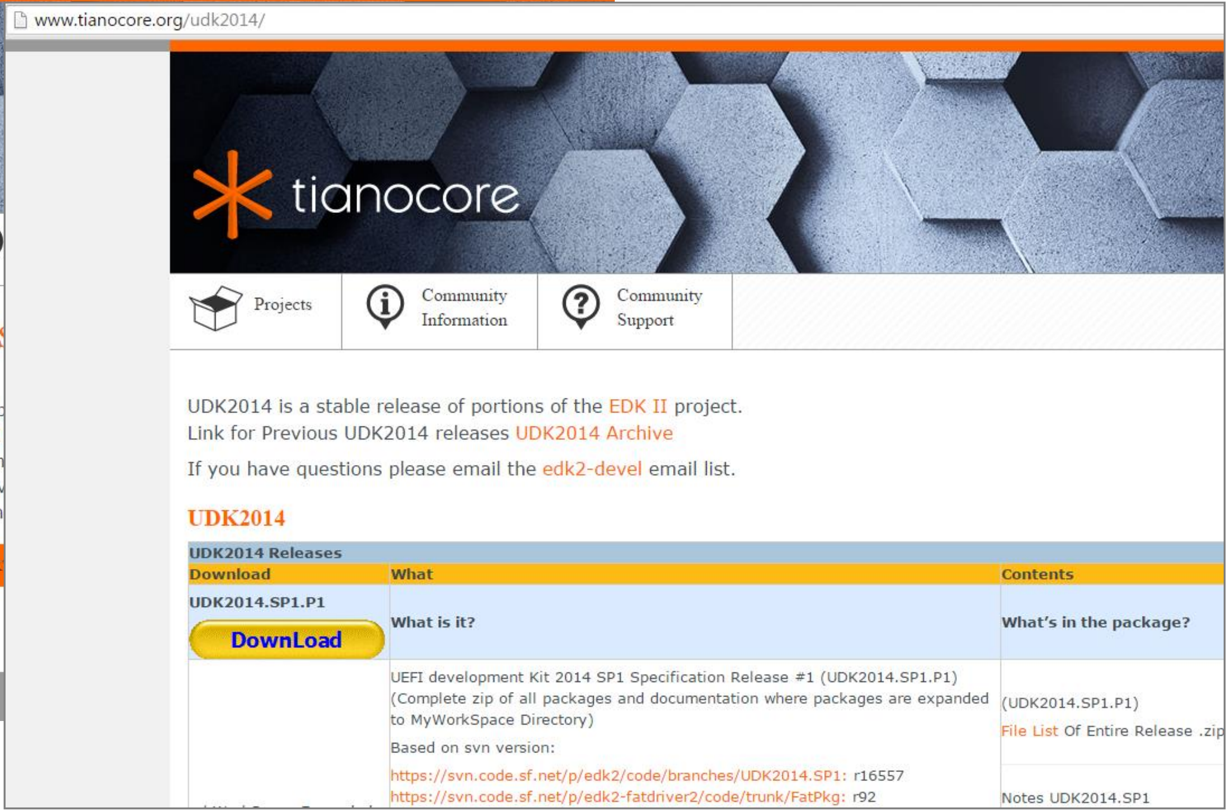
UEFI PI [Compliant] Firmware


CPU Reset
SEC
S-CRTM; Init caches/MTRRs; Cache-as-RAM (NEM); Recovery; TPM Init
Pre-EFI Init (PEI)
S-CRTM: Measure DXE/BDS
Early CPU/PCH Init
Memory (DIMMs, DRAM) Init
Driver Exec Env (DXE)
UEFI "Core" functionality, Continue initialization of platform & devices Enum FV, dispatch drivers (network, I/O, service..), Produce Boot and Runtime Services, SMM Initialization
Boot Dev Select (BDS)
Boot Manager (Select Boot Device)
EFI Shell/Apps; OS Boot Loader(s); Option ROM
ACPI, UEFI SystemTable, SMBIOS table, Lock resources
Runtime / OS
ExitBootServices. Minimal UEFI services (Variable, Capsule)



E TO THE OPEN
ITY OF UEFI

This is the community site surrounding the op
of Intel's implementation of UEFI. Our EDK II
rich, cross-platform firmware development en
and PI specifications. We hope that you'll delv
excited to use Tianocore, and contribute to th



UDK2014 is a stable release of portions of the EDK II project.
Link for Previous UDK2014 releases UDK2014 Archive

If you have questions please email the edk2-devel email list.

UDK2014

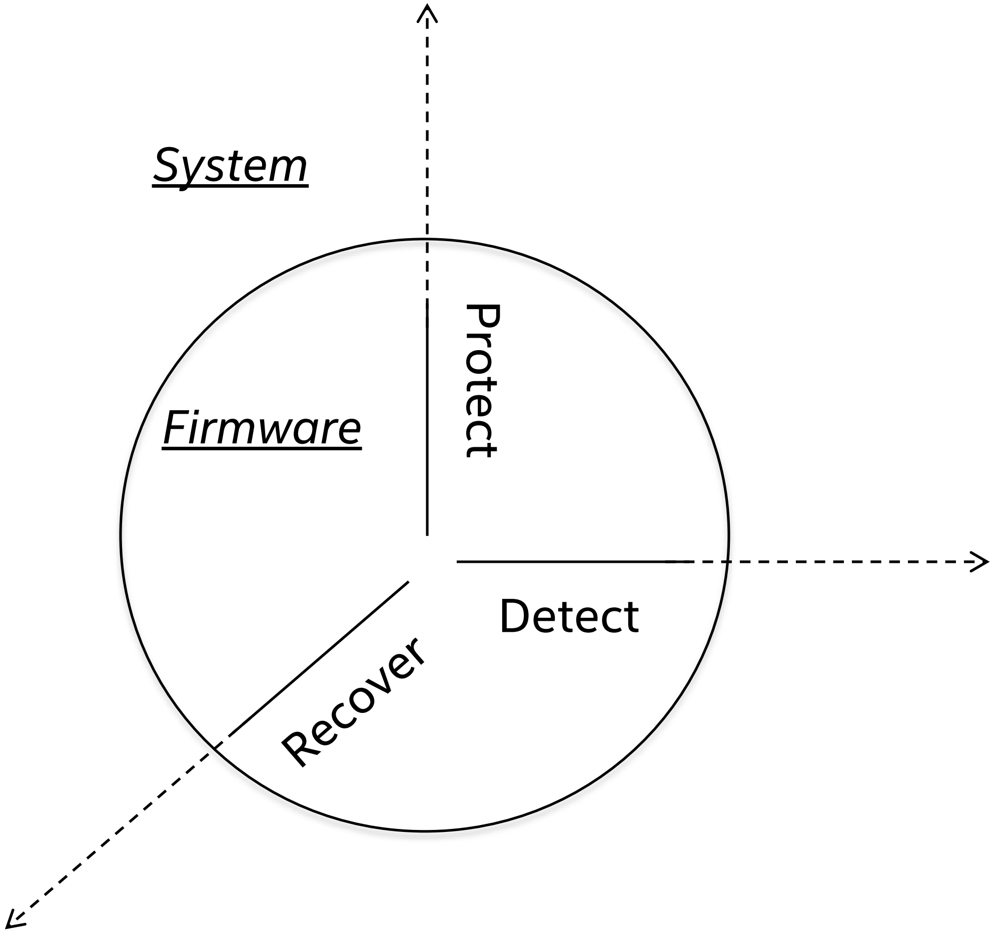
UDK2014 Releases

Download	What	Contents
UDK2014.SP1.P1 DownLoad	What is it?	What's in the package?
	UEFI development Kit 2014 SP1 Specification Release #1 (UDK2014.SP1.P1) (Complete zip of all packages and documentation where packages are expanded to MyWorkSpace Directory) Based on svn version: https://svn.code.sf.net/p/edk2/code/branches/UDK2014.SP1: r16557 https://svn.code.sf.net/p/edk2-fatdriver2/code/trunk/FatPkg: r92	(UDK2014.SP1.P1) File List Of Entire Release .zip Notes UDK2014.SP1

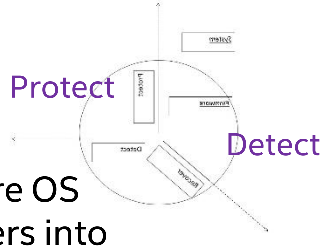
UDK2014 Available on Tianocore.org

UDK2015 Coming Soon

Usage of the EDK II Security Ingredients

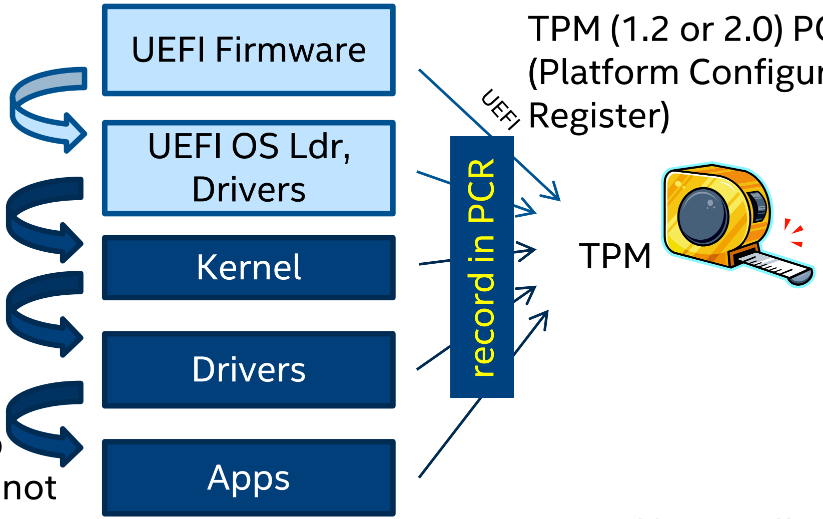


UEFI Secure Boot vs. TCG Trusted Boot



UEFI authenticate OS loader
(pub key and policy)

Check signature of before loading

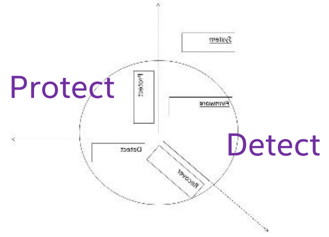


UEFI PI will measure OS loader & UEFI drivers into TPM (1.2 or 2.0) PCR (Platform Configuration Register)

- UEFI Secure boot will stop platform boot if signature not valid (OEM to provide remediation capability)
- UEFI will require remediation mechanisms if boot fails

- TCG Trusted boot will never fail
- Incumbent upon other software to make security decision using attestation

UEFI Development Kit 2014 SecurityPkg



RandomNumberGenerator

- UEFI driver implementing the EFI_RNG_PROTOCOL from the UEFI2.4 specification

Trusted Computing Group (TCG)

- PEI Modules & DXE drivers implementing Trusted Computing Group measured boot
- EFI_TCG_PROTOCOL and EFI_TREE_PROTOCOL from the TCG and Microsoft* MSDN websites, respectively

UserIdentification

- DXE drivers that support multi-factor user authentication
- Chapter 31 of the UEFI 2.4 specification

Library

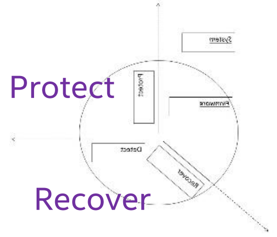
- DxeVerificationLib for “UEFI Secure Boot”, chapter 27.2 of the UEFI 2.4 specification + other support libs

VariableAuthenticated

- SMM and runtime DXE authenticated variable driver, chapter 7 of the UEFI2.4 specification

https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg

Additional Capabilities in Open Source



Variable Lock Protocol

Make variables read-only

https://github.com/tianocore/edk2/blob/master/MdeModulePkg/Include/Protocol/VariableLock.h

Lock Box

Protect content across re-starts

https://github.com/tianocore/edk2-MdeModulePkg/blob/master/Include/Protocol/LockBox.h

Capsule Update

Generic capsule update driver support

http://comments.gmane.org/gmane.comp.bios.tianocore.devel/8402

Recovery

Device support for recovery from PEI

https://svn.code.sf.net/p/edk2/code/trunk/edk2/MdeModulePkg/Include/Guid/RecoveryDevice.h

https://svn.code.sf.net/p/edk2/code/trunk/edk2/

Code Management

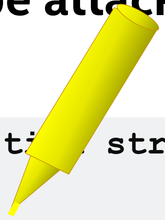
Analyze and Mark external Interfaces where input can be attacker controlled data, comment headers

```
/**  Install child handles if the Handle supports GPT partition structure.

  Caution: This function may receive untrusted input.
  The GPT partition table is external input, so this routine will do basic validation
  for GPT partition table before install child handle for each GPT partition.

  @param[in]  This        Calling context.
  @param[in]  Handle      Parent Handle.
  @param[in]  DevicePath  Parent Device Path.

 **/
 EFI_STATUS
 PartitionInstallGptChildHandle
```

UEFI Development Kit 2010 example:
http://edk2.svn.sourceforge.net/svnroot/edk2/trunk/edk2/MdeModulePkg/Universal/Disk/PartitionDxe/Gpt.c

Code Management

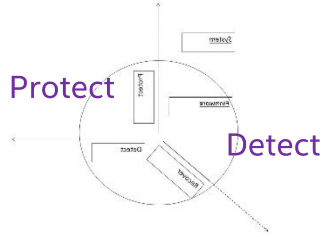
Analyze and Mark external Interfaces where input can be attacker controlled data, comment headers

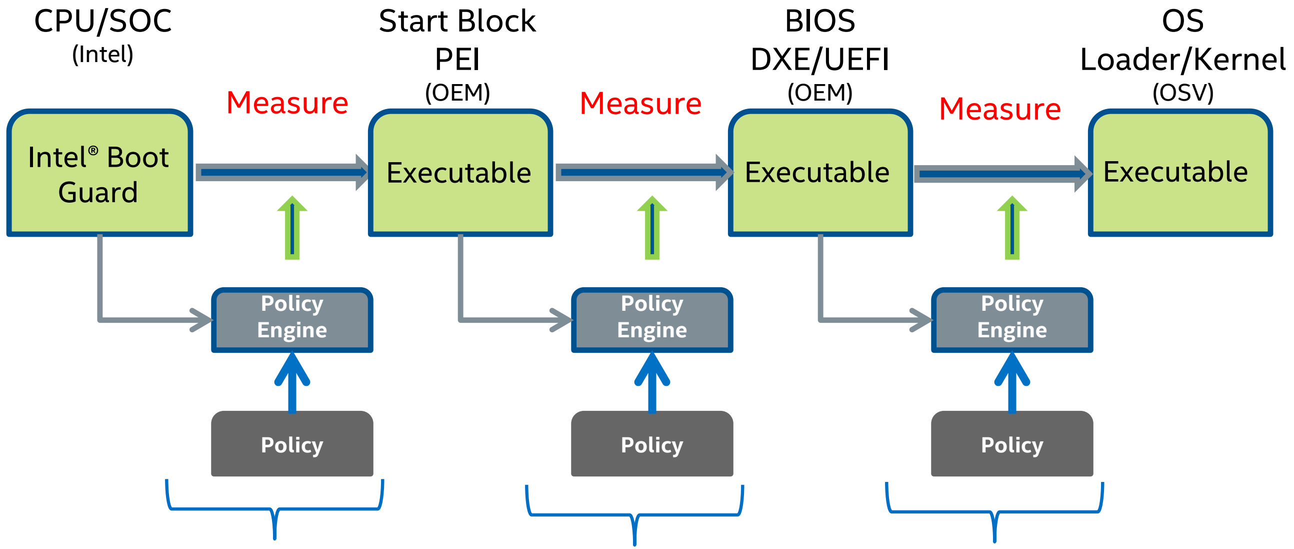


```
/**  Install child handles if the Handle supports GPT partiti[illegible] structure.

   Caution: This function may receive untrusted input.
   The GPT partition table is external input, so this routine will do basic validation
   for GPT partition table before install child handle for each GPT partition.

   @param[in]  This       Calling context.
   @param[in]  Handle     Parent Handle.
   @param[in]  DevicePath Parent Device Path.

**/
EFI_STATUS
PartitionInstallGptChildHandle
```

UEFI Development Kit 2010 example:
http://edk2.svn.sourceforge.net/svnroot/edk2/trunk/edk2/MdeModulePkg/Universal/Disk/PartitionDxe/Gpt.c

Full Verified Boot Sequence





Intel® Device Protection Technology with Boot Guard

http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf

OEM PI Verification Using PI Signed Firmware Volumes

Vol 3, section 3.2.1.1
of PI 1.3 Specification

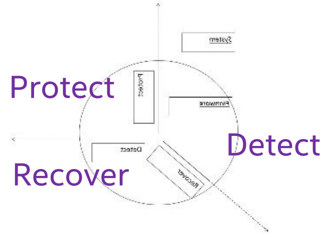
OEM UEFI 2.4 Secure Boot

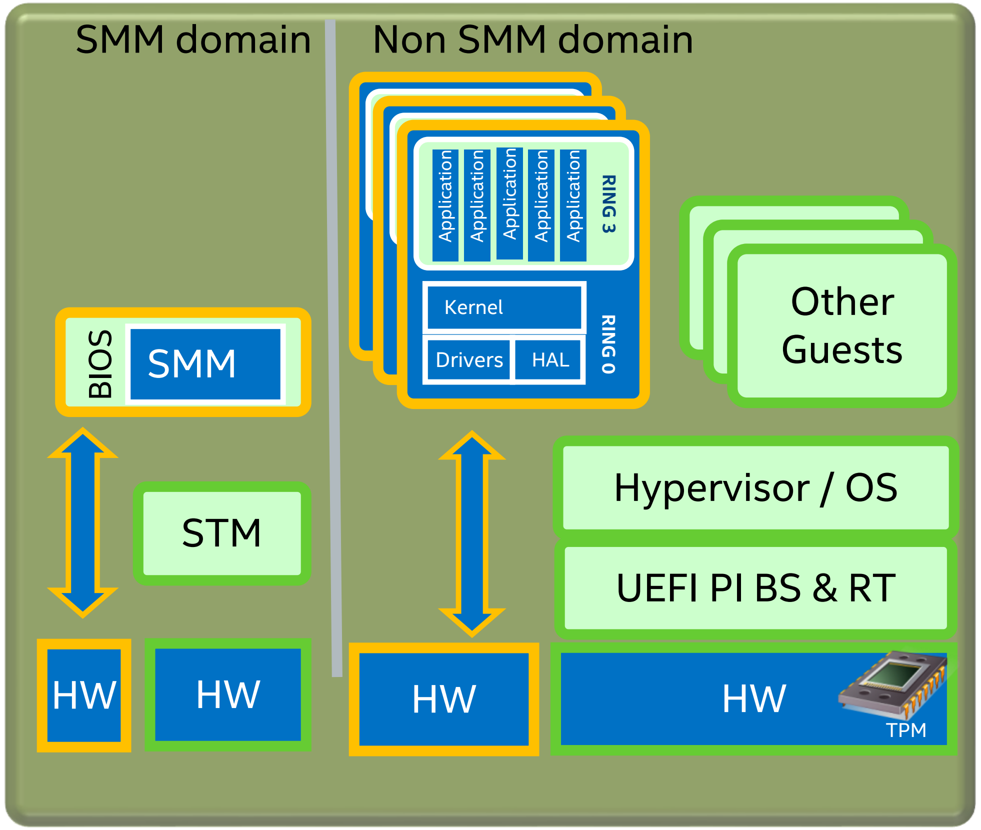
Chapter 27.2 of
The UEFI 2.4
Specification

Agenda

- Problem Statement
- Ingredients
- System Management Mode (SMM)
- Open Platforms

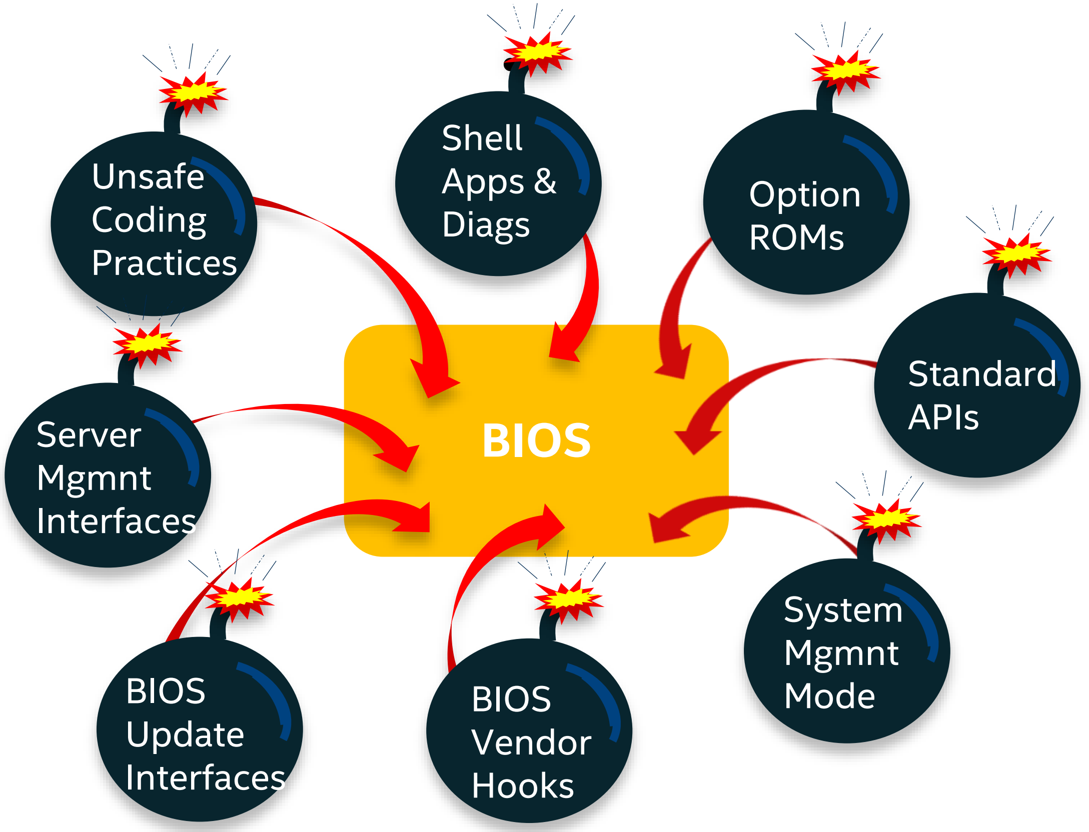
Full System Picture – UEFI PI Boot and Runtime





- **Protect & Recover the UEFI PI implementation**
  - **UEFI Capsule Update**
  - **Hardware Secure Boot using Boot Guard on non-open platforms**
- **Detect if the Hypervisor and OS is expected one**
  - **UEFI Secure Boot (and TXT+LCP on non-open platforms)**
  - **EFI TCG Measured boot**
- **Protect at runtime**
  - **SMM Transfer Monitor (STM) to protect platform, hypervisor, and operating system (OS) from the BIOS SMM**

BIOS Attack Surfaces

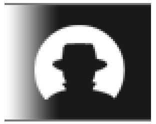


BIOS Attack Surfaces

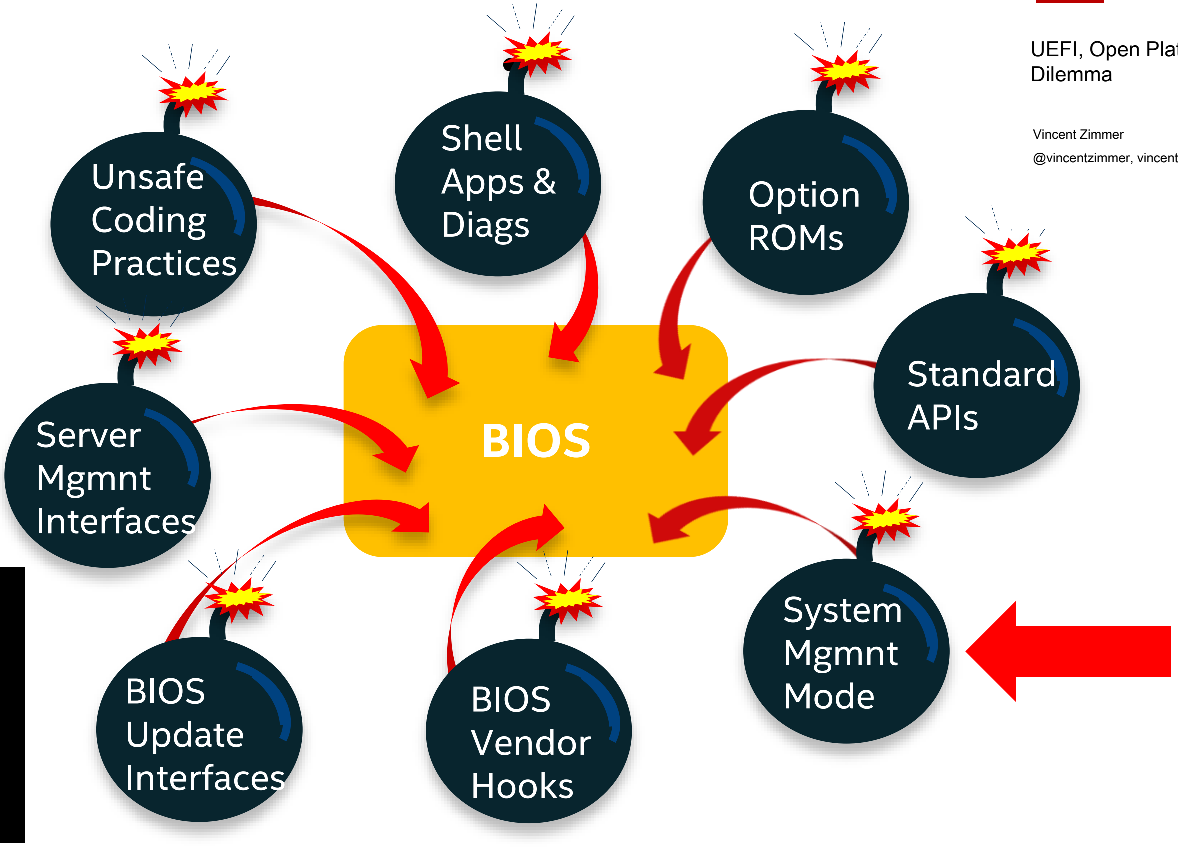
CanSecWest 2015 Vancouver, Canada

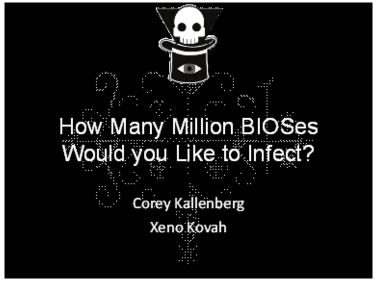
UEFI, Open Platforms, and the Defender's Dilemma

Vincent Zimmer
@vincentzimmer, vincent.zimmer @intel.com | @gmail.com



Attacking Hypervisors Using Firmware and Hardware
Bulygin, Matrosov, Gorobets, & Bazhaniuk

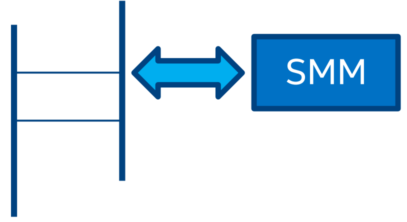




How Many Million BIOSes Would you Like to Infect?

Corey Kallenberg
Xeno Kovah

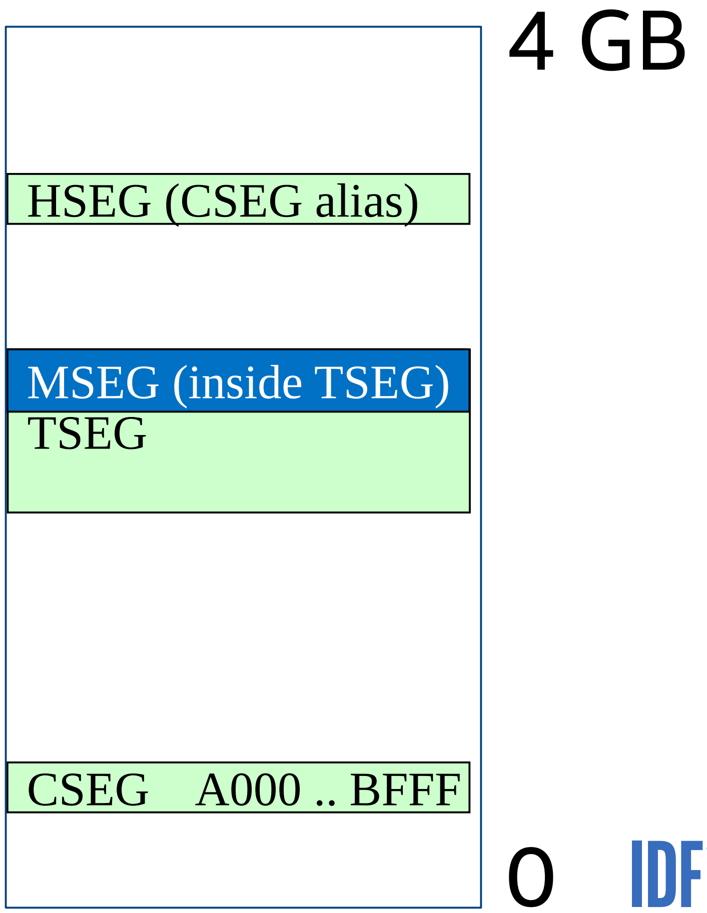
System Management Mode (SMM)



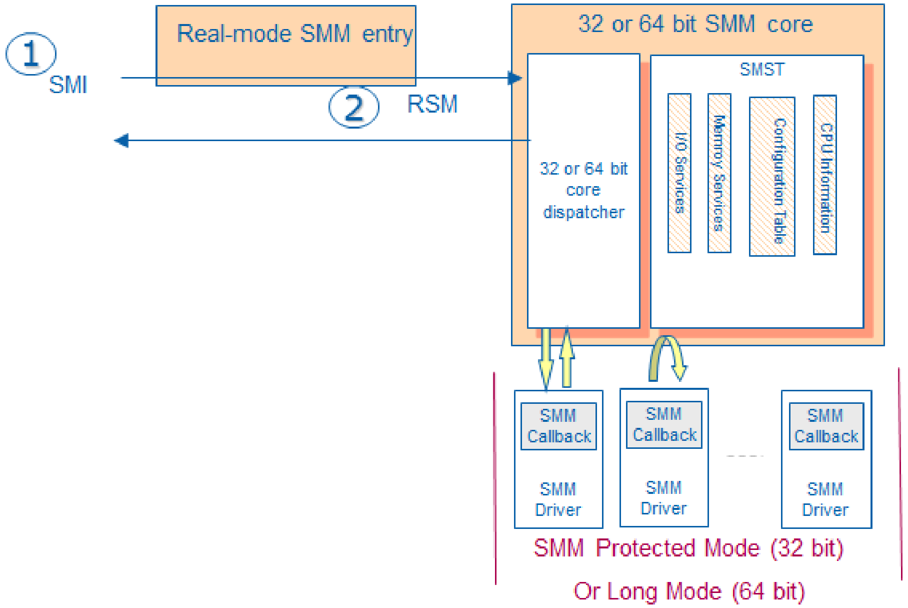
- SMM is the most privileged software in the system
- It has access to all host accessible resources
  - Memory, TPM, chipset registers, device registers
  - It can be used to protect flash – *which contains UEFI code and variables*
- It is not affected by typical OS/VMM level software controls
  - Protection rings, paging, VMX…
  - System Management Interrupt (SMI) can't be masked
  - SMRAM can't be inspected & is transparent to typical system software
- It is commonly critical for proper system operation
- Mitigations
  - code review, validate internal/external input, no call outs

System Management Mode RAM (SMRAM)

- Three SMRAM regions today: CSEG, HSEG (CSEG alias), and TSEG
  - CPU core's view of SMRAM based on internal register, SMBASE
    - SMM state save area
    - SMI entry point
- MSEG cleaved from top of TSEG on a 4K boundary
  - Related registers (programmed by BIOS):
    - IA32_SMM_MONITOR_CTL.MSEG_BASE

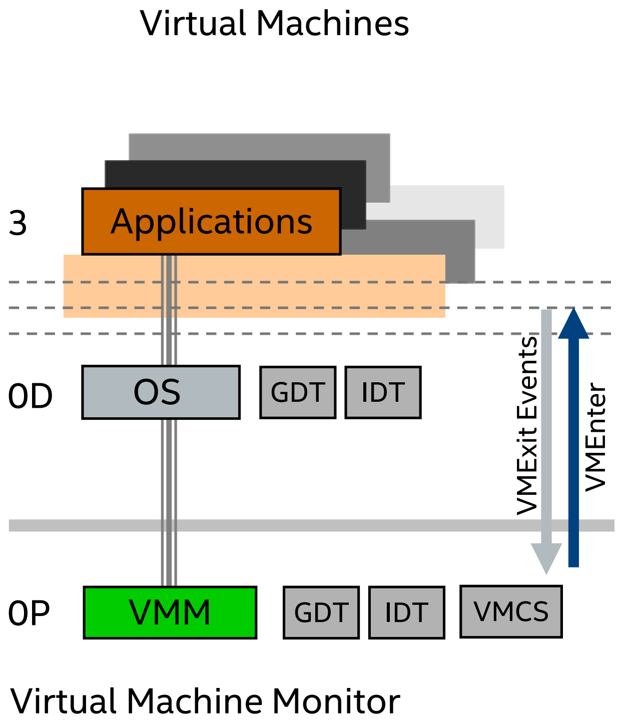


System Management Mode with UEFI PI



- Orange regions are SMRAM
- Software model defined in PI 1.4 specification, volume 4
- Implementation at edk2\MdeModulePkg\Core\PiSmmCore

Intel® Virtualization Technology (Intel® VT)



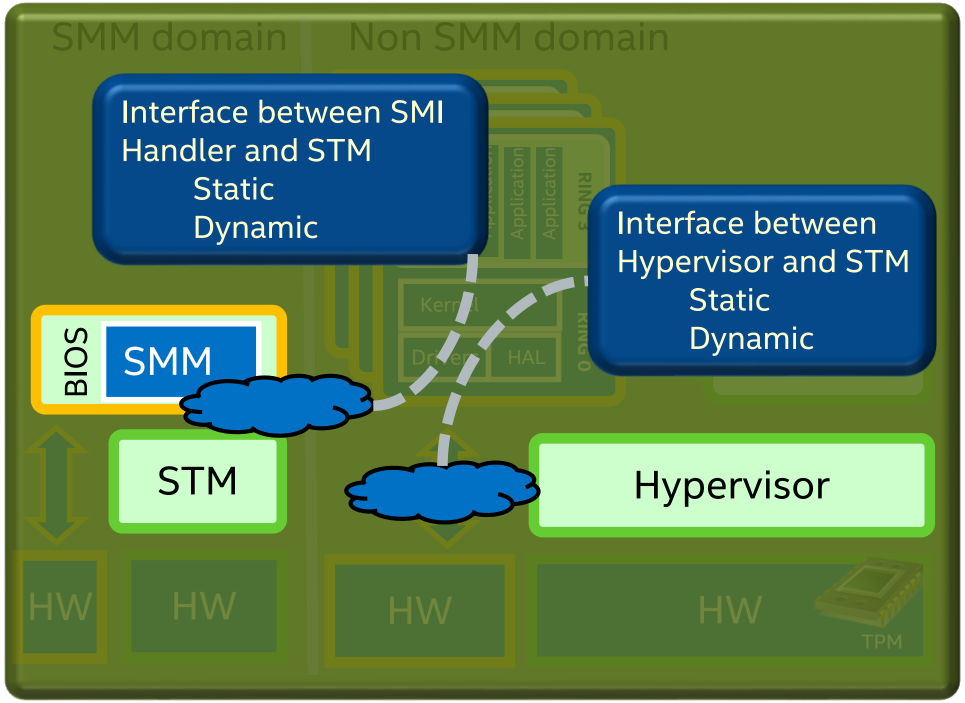
<u>VMEXIT Conditions</u>

- CR0, CR4 accesses (basic CPU operations)
- CR3 writes (address space changes)
- CR3 reads, INVLPG (paging)
- MSR & debug register accesses
- I/O instructions (per-port bitmap)
- CPUID, INVD
- Exceptions

<u>VM Control Structure (VMCS)</u>

- Which operations cause VMEXITs
- Which states change on VMEXITs and VMENTER
- VMM state area (state loaded on VMEXITs)
- Guest state area (saved on VMEXIT, restored on VMENTER)

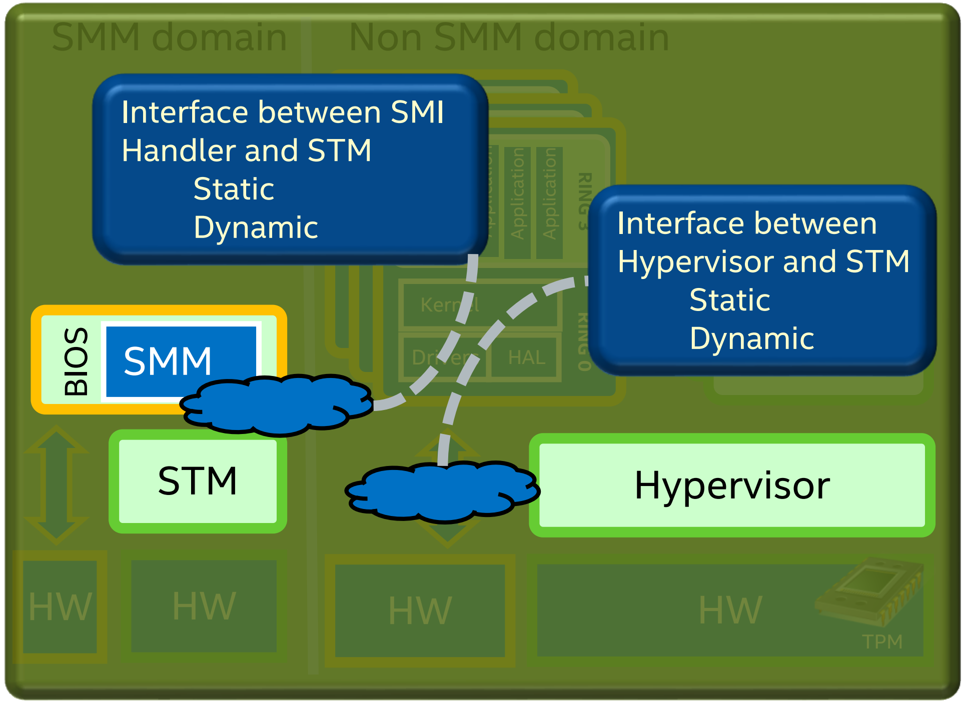
SMI Transfer Monitor (STM)



- **STM user guide defines software interfaces to manage:**
  - Setup
  - Teardown
  - Steady state (runtime)
  - … it also defines some optional ACPI and SMI based interfaces
- **STM user guide does NOT define:**
  - Protection policy
  - How protections are achieved - This is done using normal Intel® Architecture mechanisms (Intel VT, paging, etc.)

IDF15
INTEL DEVELOPER FORUM

SMI Transfer Monitor (STM)



- **STM user guide defines software interfaces to manage:**
  - Setup
  - Teardown
  - Steady state (runtime)
  - … it also defines some optional ACPI and SMI based interfaces
- **STM user guide does NOT define:**
  - Protection policy
  - How protections are achieved - This is done using normal Intel® Architecture mechanisms (Intel VT, paging, etc.)

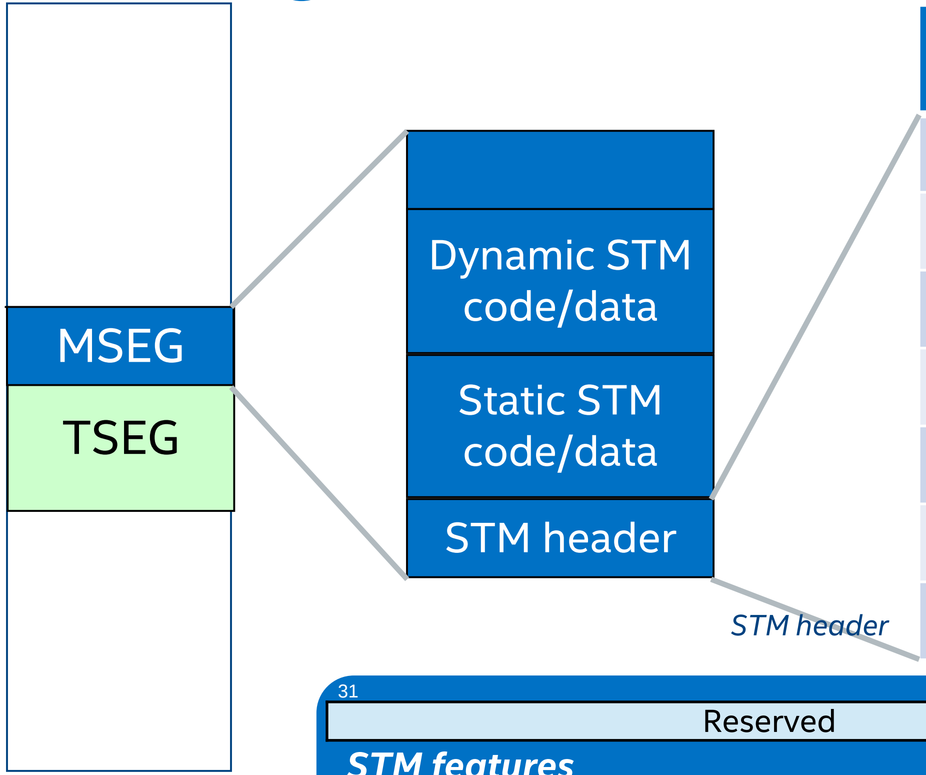
The STM provides isolation from the SMI handler

IDF15 INTEL DEVELOPER FORUM

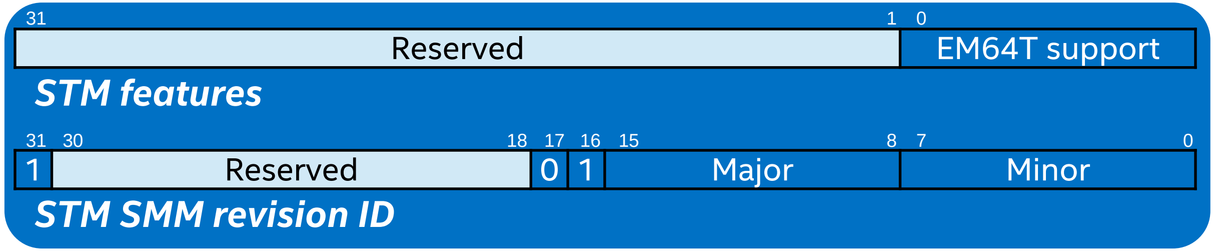
BIOS STM Opt-in

- BIOS should vigorously defend SMRAM
  - …because of its power, and critical importance to platform function
  - Therefore, BIOS must not enable an arbitrary or unknown STM
- BIOS populates MSEG with an STM image
  - BIOS should enforce it's own policy regarding what is an "acceptable" STM
    - Likely policy:
      - ❖STM image supplied as part of BIOS flash image
      - ❖BIOS flash image has controlled updates (e.g. signed)
      - ❖Therefore: "I found it in my flash, so it's acceptable"
    - Many other BIOS policy options are possible
- IA32_SMM_MONITOR_CTL.[0]   /* Valid bit */
  - BIOS sets to 1 if STM is present, BIOS clears to 0 (default) if no STM is present
  - Must be programmed identically across all CPU threads, Register only writable from SMM
- STM is idle and quiescent until it is "configured"
  - SMI is handled via legacy SMI mechanism

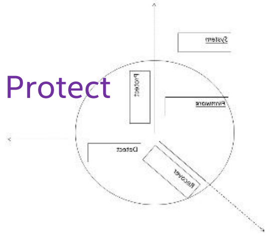
STM Image Format

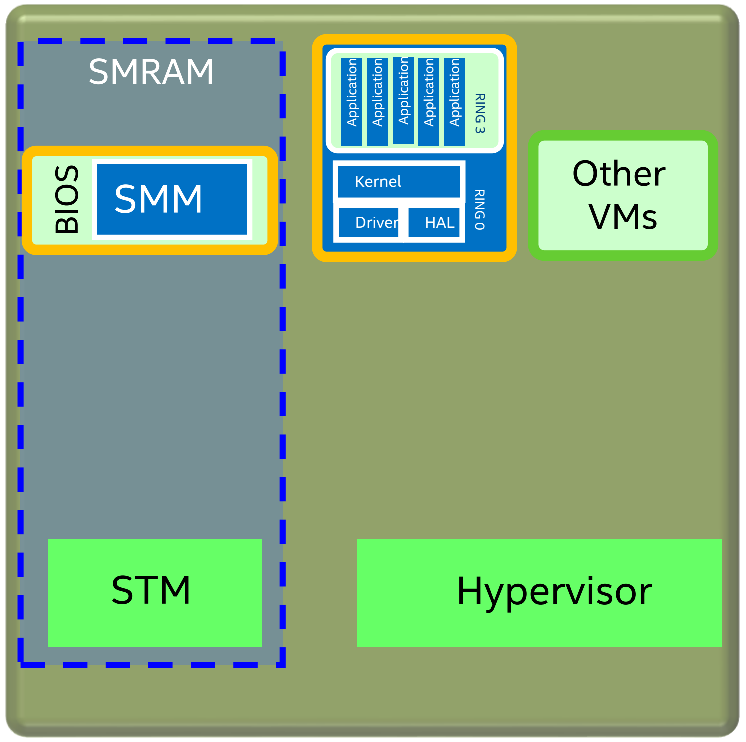


Byte offset	Field
0	MSEG-header
2K	Static image size
2K + 4	Per processor dynamic size
2K + 8	Additional dynamic size
2K + 12	STM features
2K + 16	NumSmmRevIds
2K + 20	SmmRevId[NumSmmRevIds]

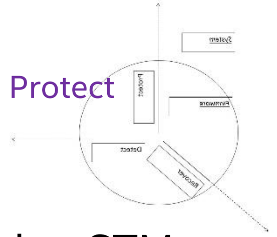


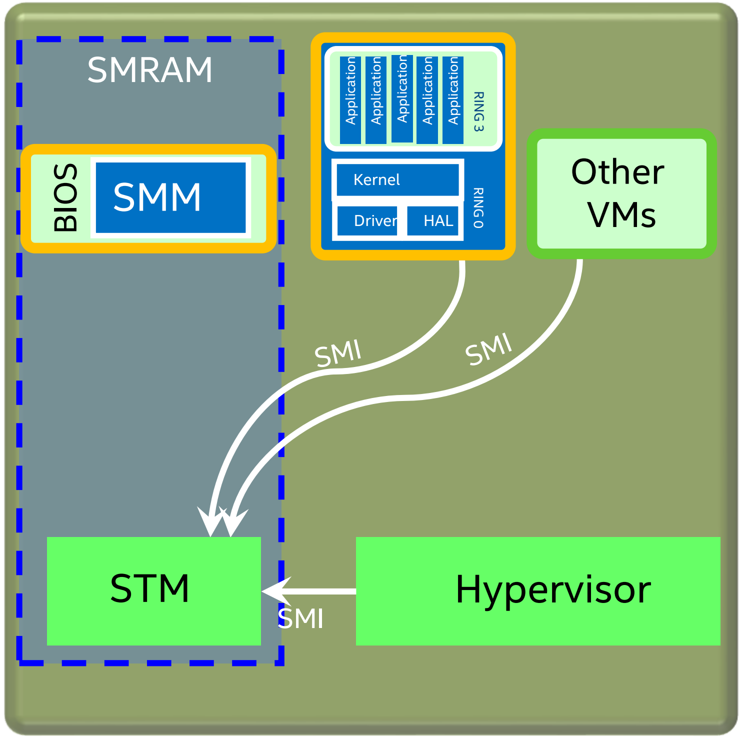
Virtualization of BIOS SMI Handler





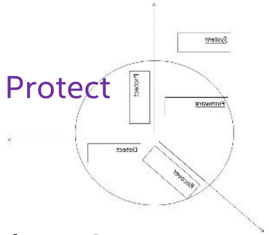
Virtualization of BIOS SMI Handler

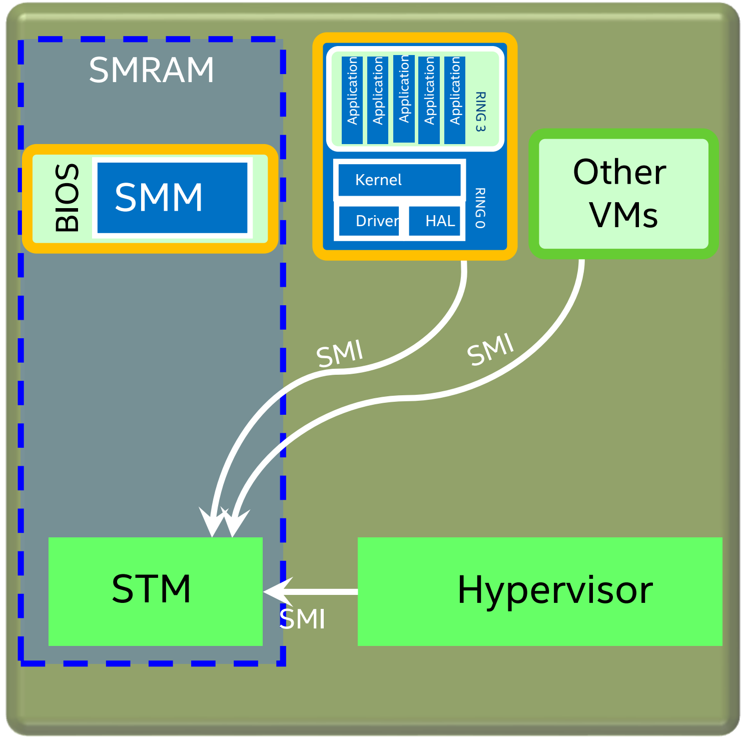




- SMI occurs - control is transferred to STM (VMEXIT)

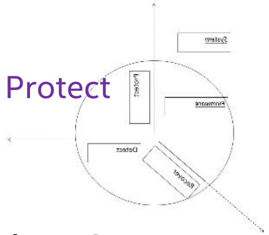
Virtualization of BIOS SMI Handler

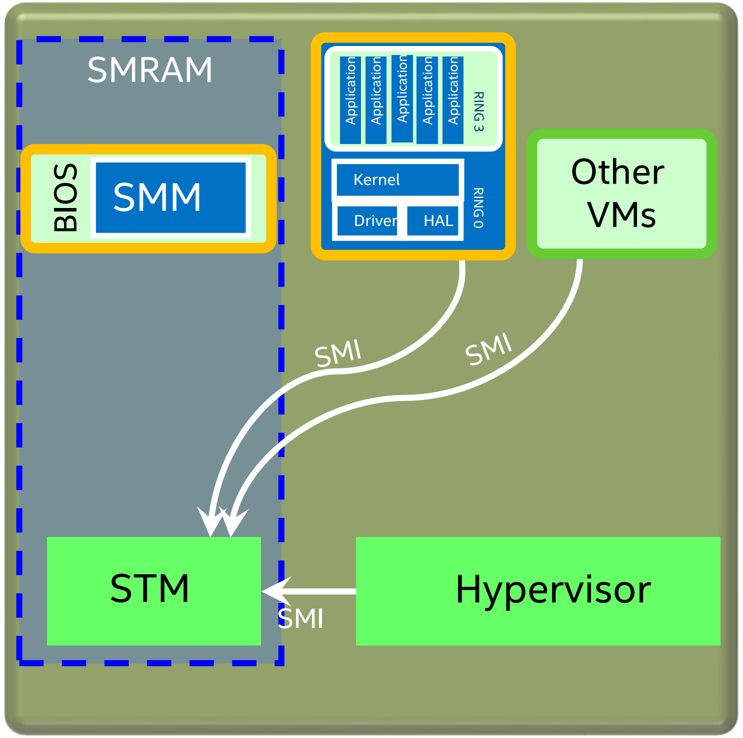




- SMI occurs - control is transferred to STM (VMEXIT)
- STM creates SMM state save area for BIOS

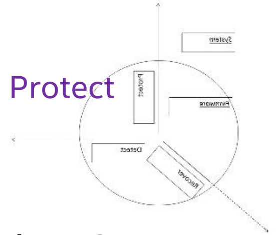
Virtualization of BIOS SMI Handler

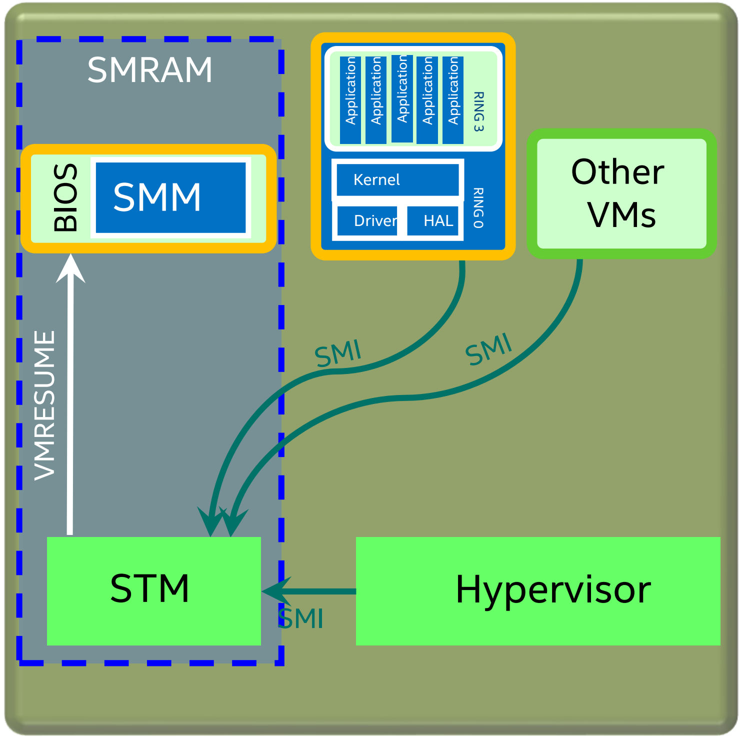




- SMI occurs - control is transferred to STM (VMEXIT)
- STM creates SMM state save area for BIOS
  - Scrubs register state if protected code has been interrupted by SMI

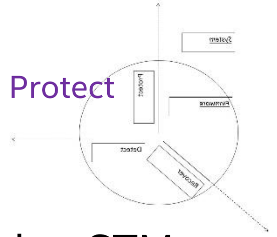
Virtualization of BIOS SMI Handler

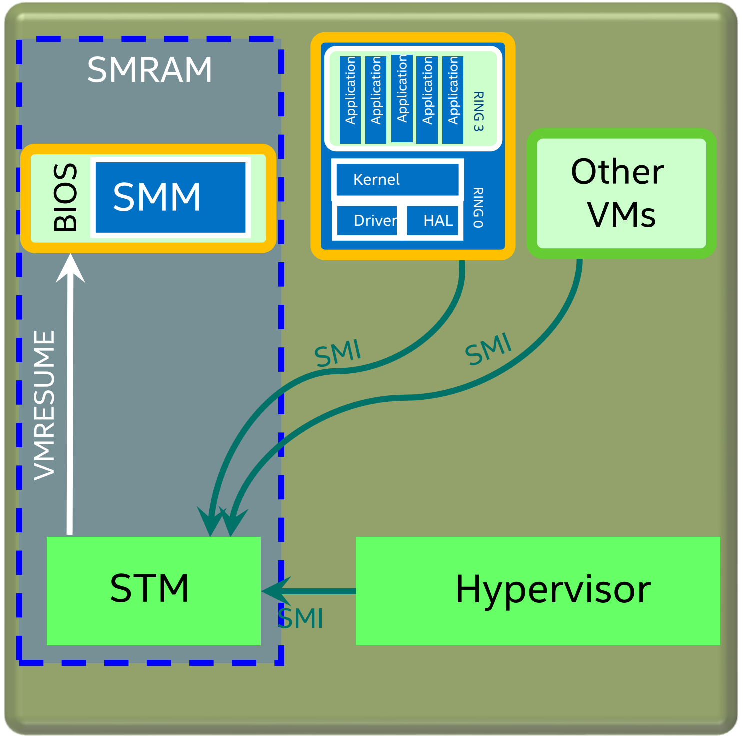




- SMI occurs - control is transferred to STM (VMEXIT)
- STM creates SMM state save area for BIOS
  - Scrubs register state if protected code has been interrupted by SMI
- STM resumes BIOS SMI handler in guest VM

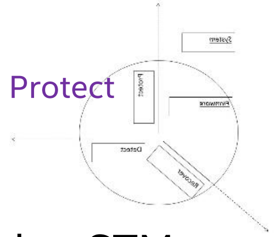
Virtualization of BIOS SMI Handler

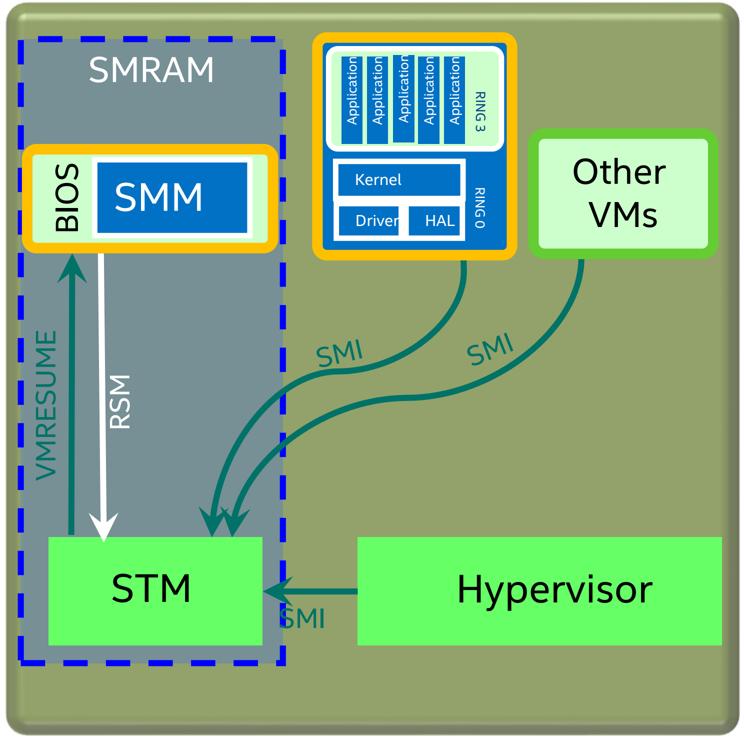




- SMI occurs - control is transferred to STM (VMEXIT)
- STM creates SMM state save area for BIOS
  - Scrubs register state if protected code has been interrupted by SMI
- STM resumes BIOS SMI handler in guest VM
- BIOS SMM code handles SMI
- STM traps on protected hardware accesses
  - Based on negotiated protection profile

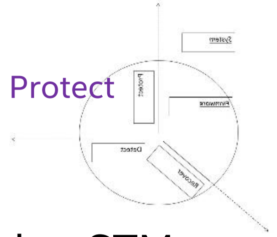
Virtualization of BIOS SMI Handler

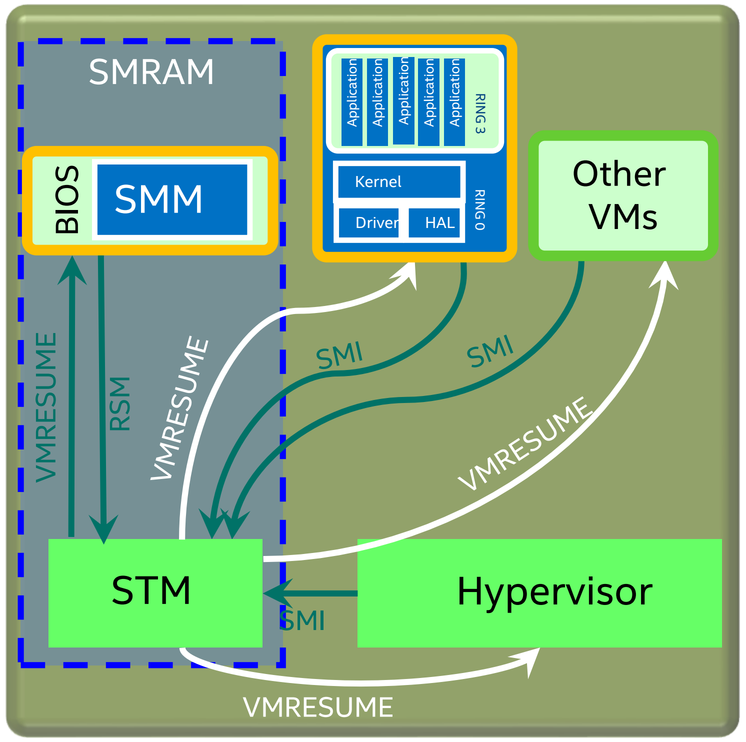




- SMI occurs - control is transferred to STM (VMEXIT)
- STM creates SMM state save area for BIOS
  - Scrubs register state if protected code has been interrupted by SMI
- STM resumes BIOS SMI handler in guest VM
- BIOS SMM code handles SMI
- STM traps on protected hardware accesses
  - Based on negotiated protection profile
- BIOS SMM executes RSM (VMEXIT) to STM

Virtualization of BIOS SMI Handler





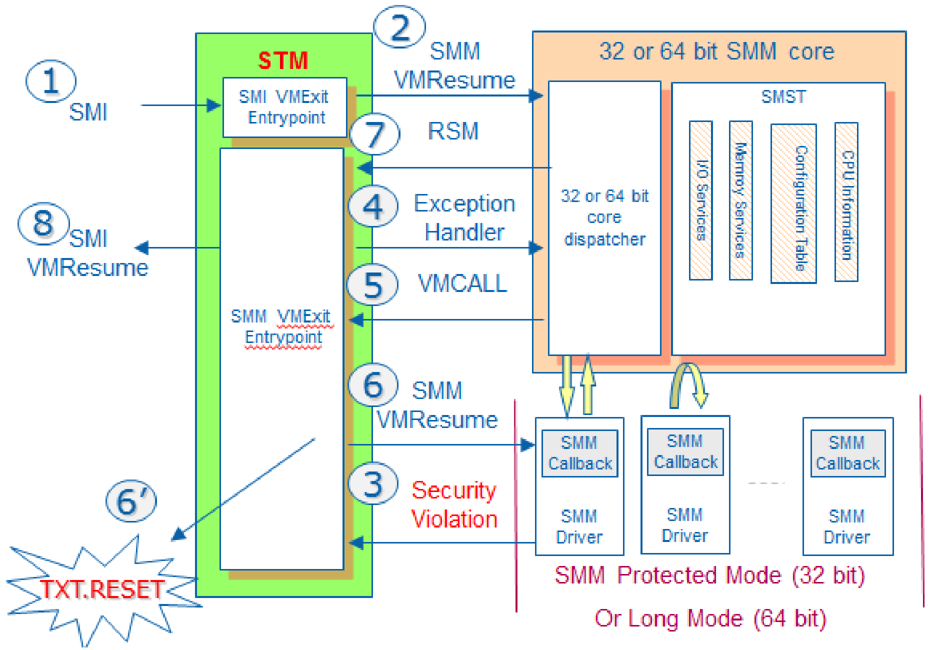
- SMI occurs - control is transferred to STM (VMEXIT)
- STM creates SMM state save area for BIOS
  - Scrubs register state if protected code has been interrupted by SMI
- STM resumes BIOS SMI handler in guest VM
- BIOS SMM code handles SMI
- STM traps on protected hardware accesses
  - Based on negotiated protection profile
- BIOS SMM executes RSM (VMEXIT) to STM
- STM restores interrupted VMs and resumes them

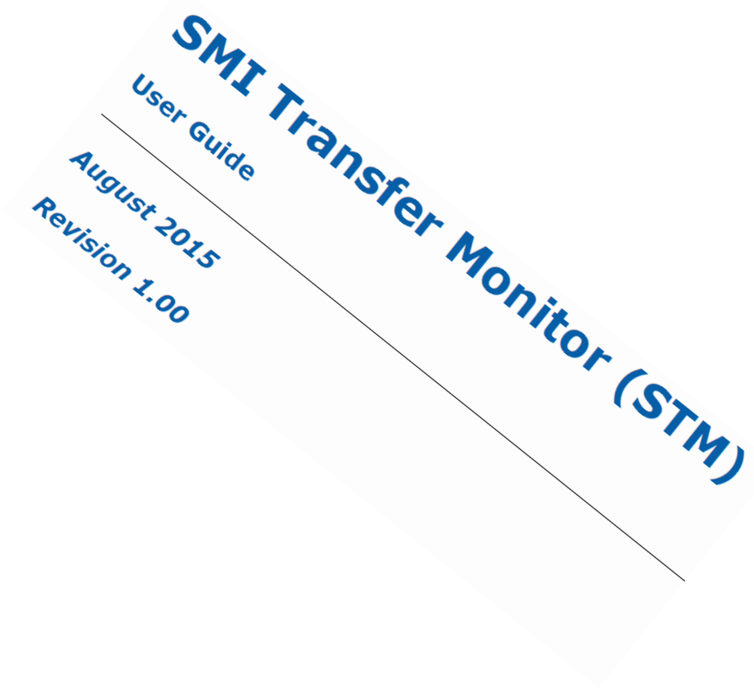
How to Declare Resources Allowed for SMM

- STM allocates hardware resources to BIOS SMI and MLE on a first-come-first-served basis. BIOS always has first opportunity to make a request
  - This is done statically via the BiosHwResourceRequirementsPtr
  - 64 bit physical pointer to a STM_RESOURCE_LIST

```
<STM_RESOURCE_LIST> ::= { <STM_RSC> } <STM_RSC_END>

<STM_RSC> ::=    <STM_RSC_MEM_DESC>
      |    <STM_RSC_IO_DESC>
      |    <STM_RSC_PCI_CFG_DESC>
      |    <STM_RSC_MSR_DESC>

<END> ::= <STM_RSC_END>
```

SMM Flow with STM



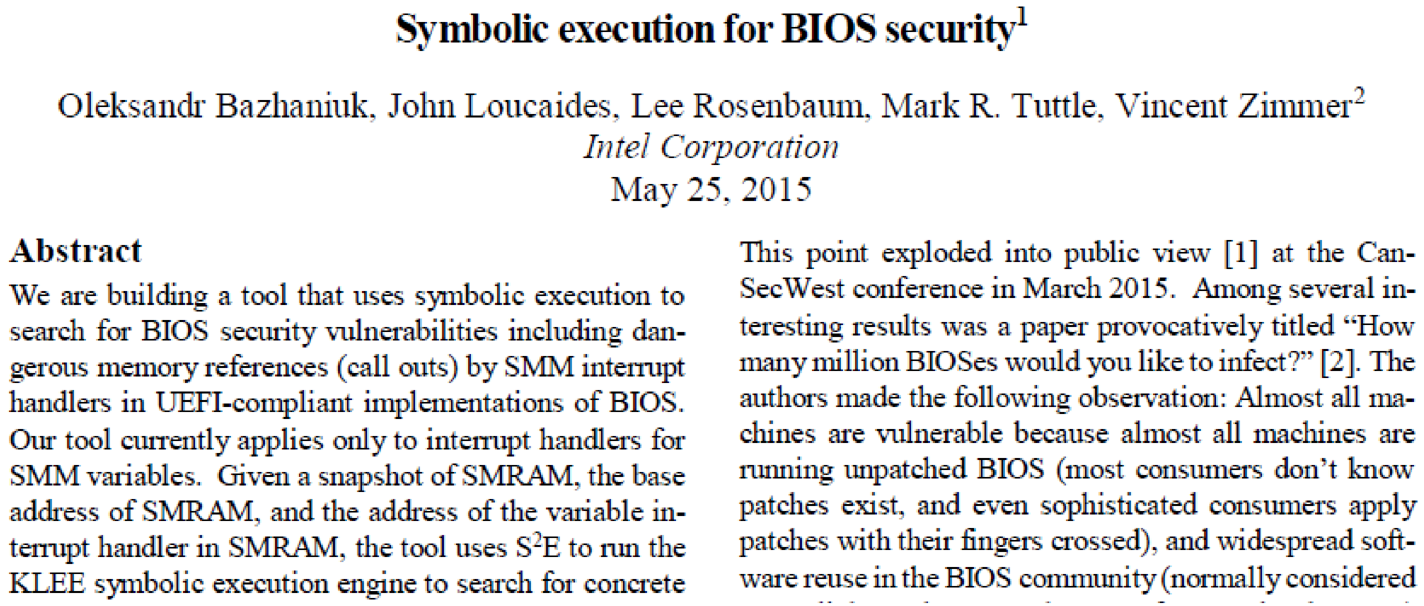


- firmware.intel.com to find STM user guide
- STM Reference implementation built on EDKII infrastructure
  - Build system, Mde Libraries, test driver and MinnowMax integration

Heading to TianoCore.org

Beyond STM Isolation, Moving to Testing

Usenix* WOOT 2015: KLEE → S2E →....



Symbolic execution for BIOS security[1]

Oleksandr Bazhaniuk, John Loucaides, Lee Rosenbaum, Mark R. Tuttle, Vincent Zimmer[2]
Intel Corporation
May 25, 2015

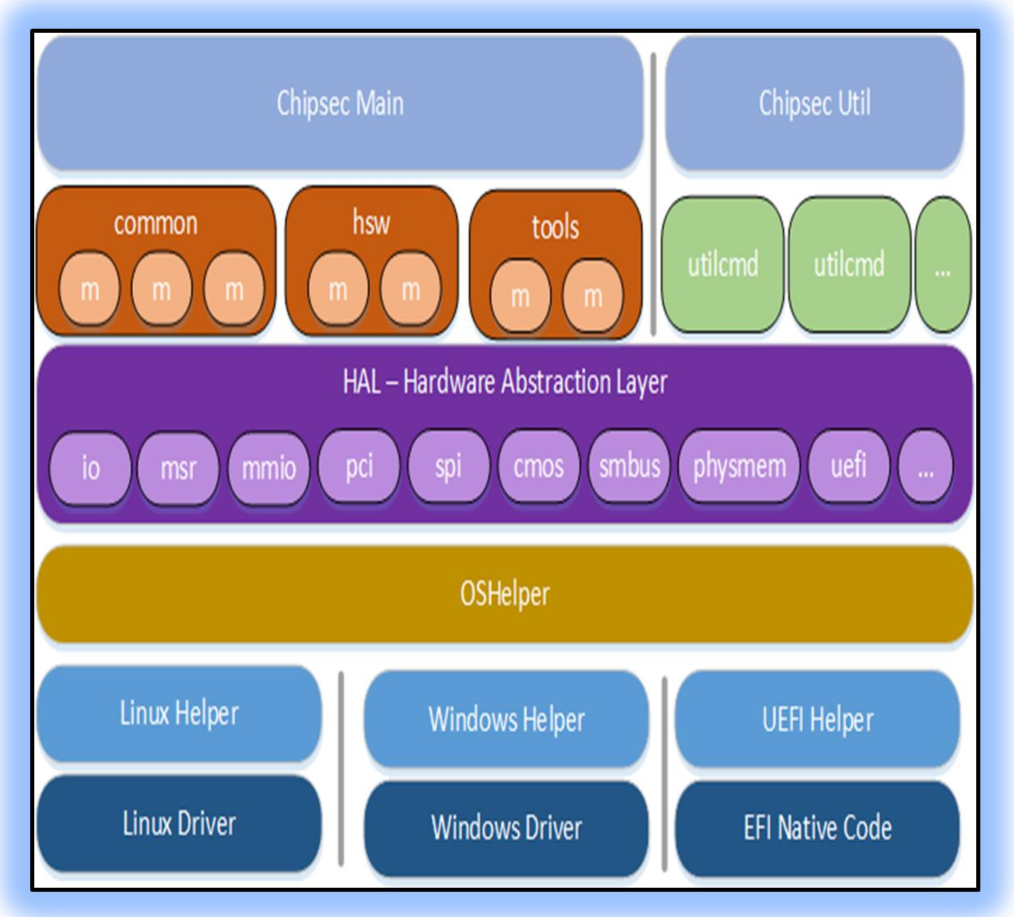
Abstract
We are building a tool that uses symbolic execution to search for BIOS security vulnerabilities including dangerous memory references (call outs) by SMM interrupt handlers in UEFI-compliant implementations of BIOS. Our tool currently applies only to interrupt handlers for SMM variables. Given a snapshot of SMRAM, the base address of SMRAM, and the address of the variable interrupt handler in SMRAM, the tool uses S^2E to run the KLEE symbolic execution engine to search for concrete

This point exploded into public view [1] at the CanSecWest conference in March 2015. Among several interesting results was a paper provocatively titled "How many million BIOSes would you like to infect?" [2]. The authors made the following observation: Almost all machines are vulnerable because almost all machines are running unpatched BIOS (most consumers don't know patches exist, and even sophisticated consumers apply patches with their fingers crossed), and widespread software reuse in the BIOS community (normally considered

WOOT 2015 Paper

chipsec

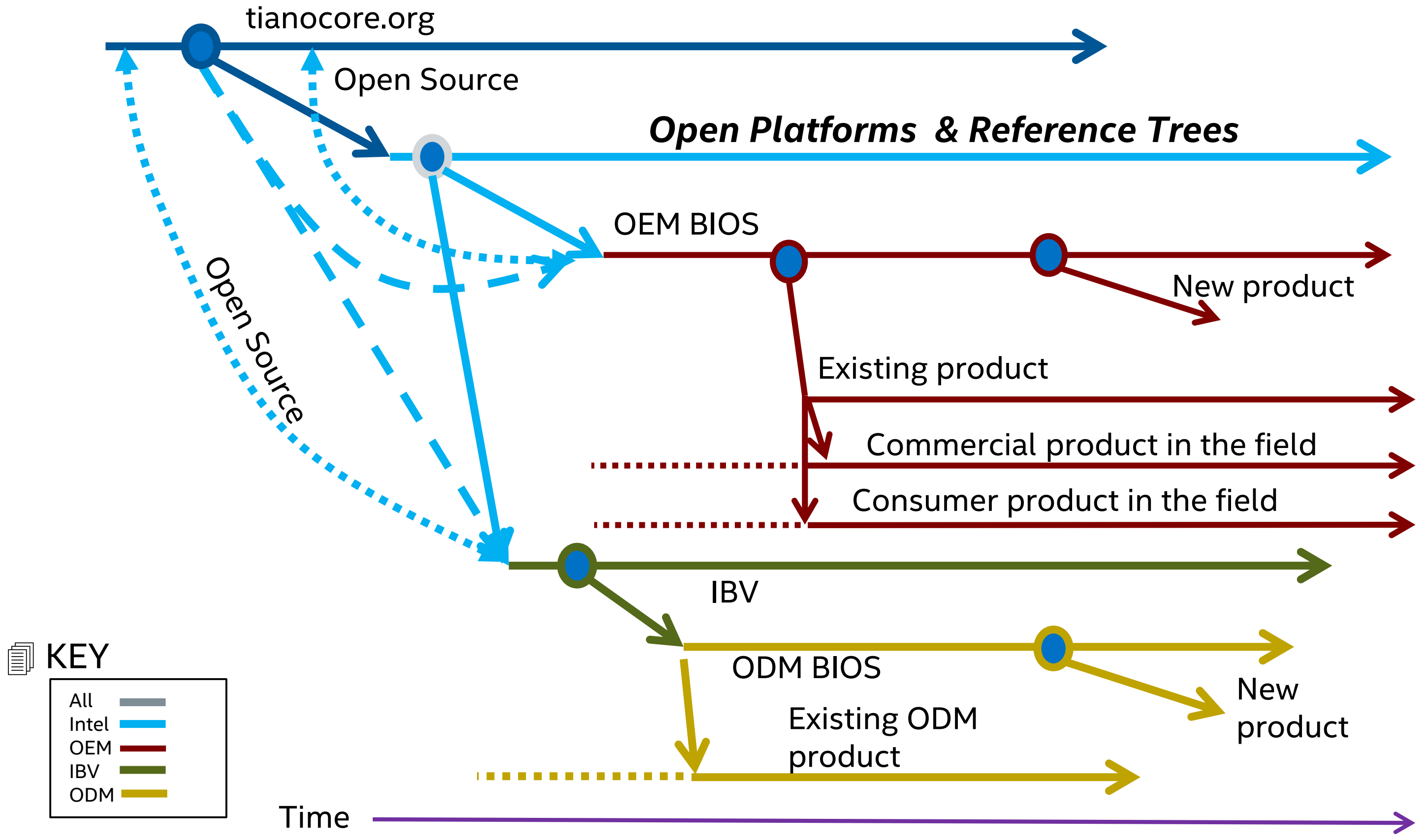
- A platform security assessment framework for risk assessment
- Can be extended to meet specific platform security concerns
- Open sourced https://github.com/chipsec/chipsec



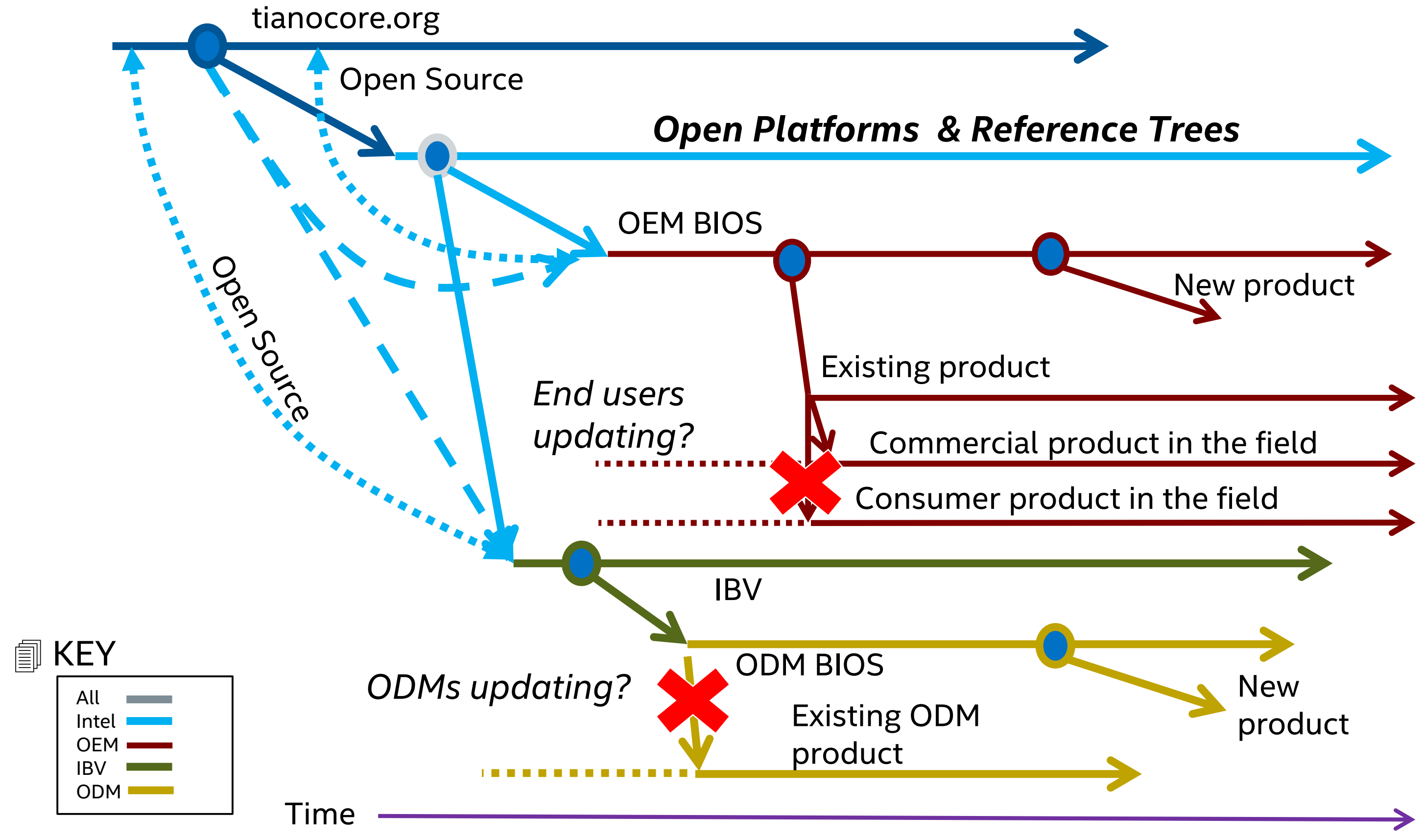
Agenda

- Problem Statement
- Ingredients
- System Management Mode (SMM)
- Open Platforms

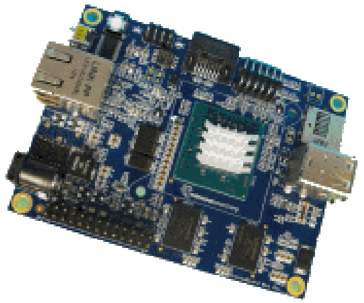
The Road from Core to Platform



The Road from Core to Platform



MinnowBoard Max



- Open hardware platform
- Intel® Atom™ SoC E38xx Series SoC single or dual core
- From http://firmware.intel.com/projects
- This project focuses in on the firmware source code (and binary modules) required to create the boot firmware image for the MinnowBoard MAX. The UEFI Open Source (EDKII project) packages for MinnowBoard MAX are available at http://tianocore.sourceforge.net/wiki/EDK2. To learn more about getting involved in the UEFI EDKII project visit the How to Contribute page.
- The source code builds using Microsoft Visual Studios* and GNU* C Compiler (for both 32 and 64 bit images) - production and debug execution environments. The source code builds the same UEFI firmware image shipping on MinnowBoard MAX.
- See more at: http://firmware.intel.com/projects#sthash.1oOc8srY.dpuf

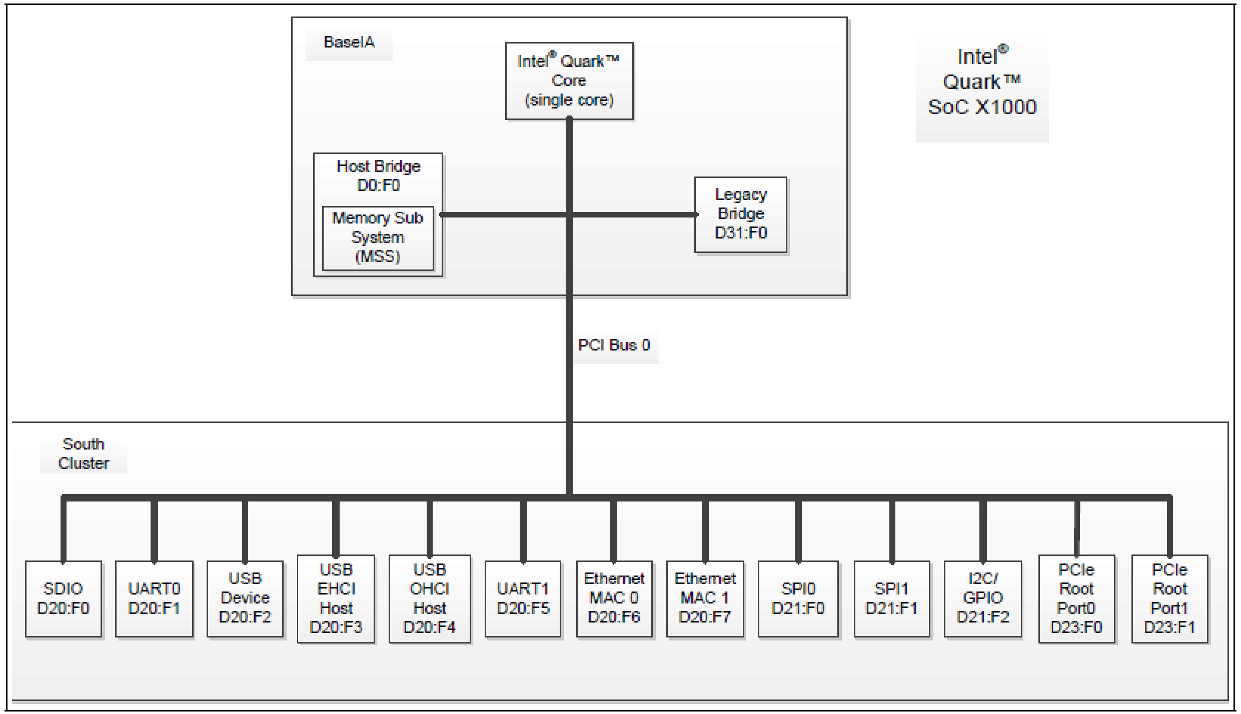
MinnowBoard Max

- Focused on the maker community, but….

- 64-bit Intel® Atom™ SoC E38xx Series
- Has UEFI Secure Boot
- Built off of live tree
- Supports the SMM Transfer Monitor (STM) without Intel® Trusted Execution Technology (Intel® TXT)
- Ability to update with latest capabilities on http://www.tianocore.org

Intel® Quark™ SoC – Hardware Overview

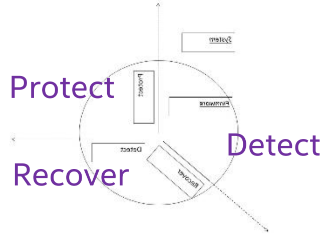
- ISA-class 32 bit Intel® Pentium® processor
- PCI
- USB
- I2C
- Single core



UEFI for Intel® Quark™ SoC

- First fully open source Intel® Galileo based platform
- Builds on Intel® UDK2014 packages like MdePkg, MdeModulePkg w/ a 32-bit build, adding
  - IA32FamilyCpuBasePkg
  - QuarkPlatformPkg
  - QuarkSocPkg
- Standard build is 1 Mbyte image w/full features
  - Capsule update, SMM, S3, PCI, recovery, full UEFI OS support, FAT OS support, UEFI variables

Intel® Quark™ SoC and Security



- Support for I2C-attached TPM
- Hardware Secure Boot option
- UEFI Secure Boot implementation
- UEFI Capsule update support w/ hardware verification assist
- Demonstrates one way to build out UEFI Security Features with a full open source platform tree, with the following summary

	Capsule update	UEFI Secure Boot	TCG Measured Boot	STM	chipsec
MinnowBoard Max	Yes – with open source Capsule driver	Yes	Yes – Integrated TPM	Yes – VT w/o TXT	Yes
Intel® Quark™ Intel® Galileo	Yes – with BootROM support	Yes	Yes – I2C TPM	No	Yes

Summary and Next Steps

- Many security problems, including SMM escalation
- Open source ingredients
- New approach to handle SMM and Testing
- Use open platforms to demonstrate ingredients

Additional Sources of Information

- A PDF of this presentation is available from our Technical Session Catalog: www.intel.com/idfsessionsSF. This URL is also printed on the top of Session Agenda Pages in the Pocket Guide.
- Booth info: #511

More information on security

- UEFI and PI specification – http://www.uefi.org
- EDK II Implementation – http://www.tianocore.org
- Platform Security information: https://firmware.intel.com/blog/
- EDK II Security Fixes: http://www.tianocore.or/security
- SMM attacks from https://cansecwest.com/agenda.html **"Corey Kallenberg & Xeno Kovah, LegbaCore - How many million BIOSes would you like to infect?", "Rafal Wojtczuk & Corey Kallenberg - Attacks on UEFI Security", "John Loucaides & Andrew Furtak, Intel - A new class of vulnerability in SMI Handlers of BIOS/UEFI Firmware"**
- STM Specification and code: https://firmware.intel.com/content/smi-transfer-monitor-stm
- CHIPSEC: https://github.com/chipsec/chipsec
- Intel® Quark™ Soc X1000 Version 1.1.0 BIOS https://downloadcenter.intel.com/download/23197/Intel-Quark-BSP
- MinnowMax http://www.minnowboard.org/meet-minnowboard-max/

Other Technical Sessions

	Session ID	Title	Day	Time	Room
✓	STTS001	Firmware in the Data Center: Building a Modern Deployment Framework Using UEFI and Redfish REST APIs	Tue	11:00	2002
✓	STTS002	Building a Firmware Component Ecosystem with the Intel® Firmware Engine	Tue	1:15	2002
✓	STTS003	Developing Best-in-Class Security Principles with Open Source Firmware	Tue	2:30	2002
	STTS004	Planning and Predicting Big Data Clusters for Spark*, NoSQL and SQL-on-Hadoop* Deployments	Tue	4:00	2002
	STTS005	Accelerating Real-time Analytics Insights with Open Source Software from Intel	Wed	9:30	2002

✓ = DONE

Legal Notices and Disclaimers

Risk Factors

Rev. 4/14/15