presented by

American Megatrends

Secure Firmware Update

UEFI Winter Plugfest – February 20-23, 2012
Presented by Zachary Bobroff(AMI)

Agenda

- **Background Information**
- Methodology
- Implementation
- Demonstration
- Call to Action

Why Secure Flash Update?

- Platform security is a broad topic…
  - Many overlapping technologies (TPM, secure boot, secure flash update, etc)
  - System complexity is increasing with new technologies (Execute Disable, virtualization, etc)
  - No one specification ties all security technologies together
- Firmware modification/tinkering by the hobbyist is becoming more commonplace
- The UEFI specification completely documents all interfaces
  - Malicious software can attack the firmware

Connection with Secure Boot

- Secure boot dictates that all external images must be authenticated prior to execution
- Secure boot ensures the system booted in a trusted state
- Secure boot prevents attacks targeting the firmware to OS handoff
- Secure boot does not prevent any direct attacks on the firmware itself, and the UEFI specification has no provisioning for firmware protection

NIST Involvement

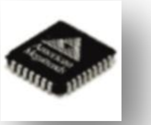
- NIST has developed firmware protection guidelines (NIST publication 800-147)
- This publication requires:
  - The BIOS must be protected
  - BIOS updates must be signed
  - BIOS protection cannot be bypassed
  - A user must be present for all BIOS updates
  - There must be anti-rollback protection

Agenda

- Background Information
- **Methodology**
- Implementation
- Demonstration
- Call to Action

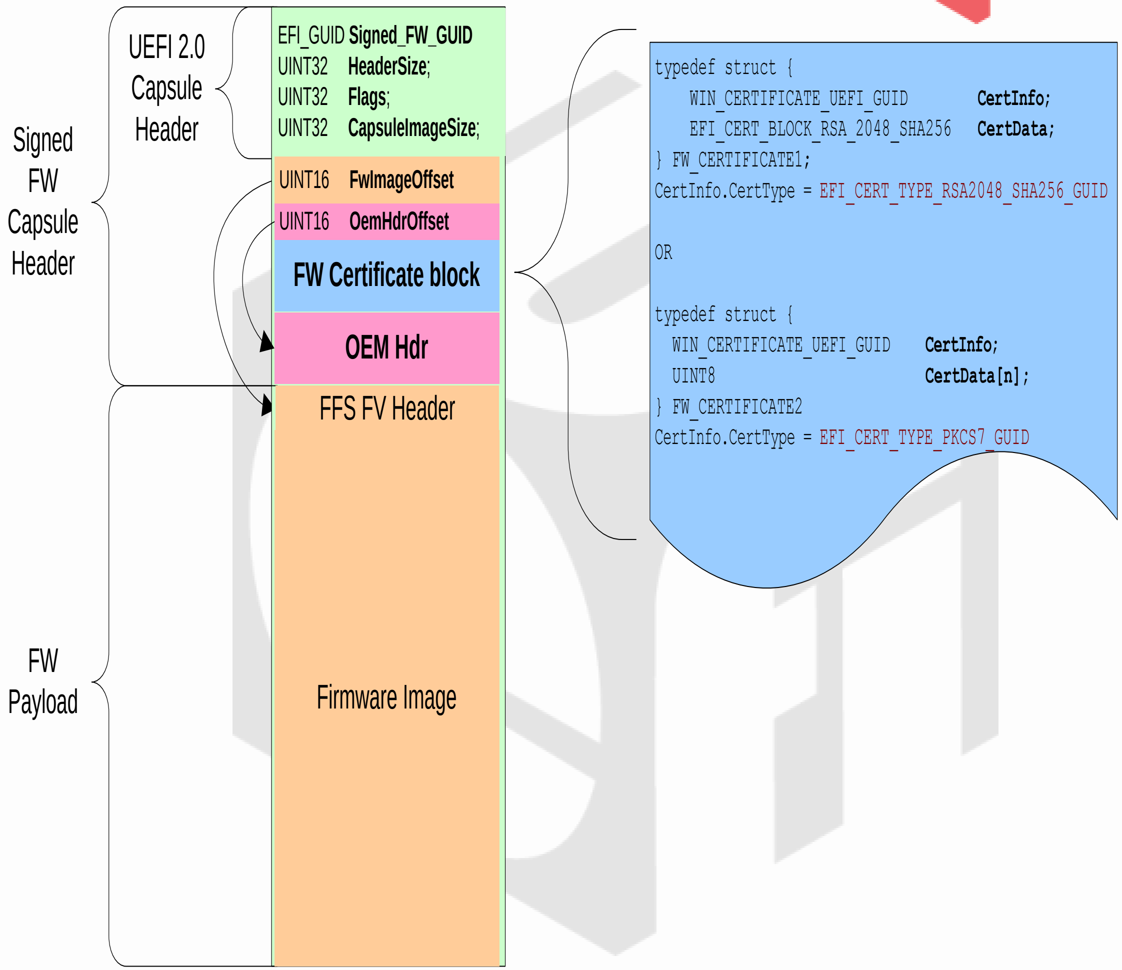
Methodology

- Use digital signatures to authenticate the BIOS image similar to secure boot in UEFI 2.3.1
  - Industry approved digital signature protocols
  - EMSA PKCS v1.15, RSA PSS signature schemas
  - 2048 bit RSA Key, SHA256 hash (NIST requirement)
- Use the UEFI Firmware Capsule as preferred delivery mechanism
- Use silicon features to prevent unauthorized updates to the flash part
  - Consult your silicon documentation for proper support information



Signed FW Capsule

- Image is a combination of the firmware payload with the firmware certificate
- Includes OEM Header and UEFI-defined Capsule Structure
- OEM Header can contain information to pass to the BIOS update process



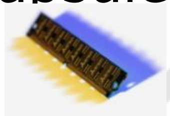
```
typedef struct {
    WIN_CERTIFICATE_UEFI_GUID        CertInfo;
    EFI_CERT_BLOCK_RSA_2048_SHA256   CertData;
} FW_CERTIFICATE1;
CertInfo.CertType = EFI_CERT_TYPE_RSA2048_SHA256_GUID

OR

typedef struct {
  WIN_CERTIFICATE_UEFI_GUID    CertInfo;
  UINT8                        CertData[n];
} FW_CERTIFICATE2
CertInfo.CertType = EFI_CERT_TYPE_PKCS7_GUID
```

Agenda

- Background Information
- Methodology
- **Implementation**
- Demonstration
- Call to Action

Implementation

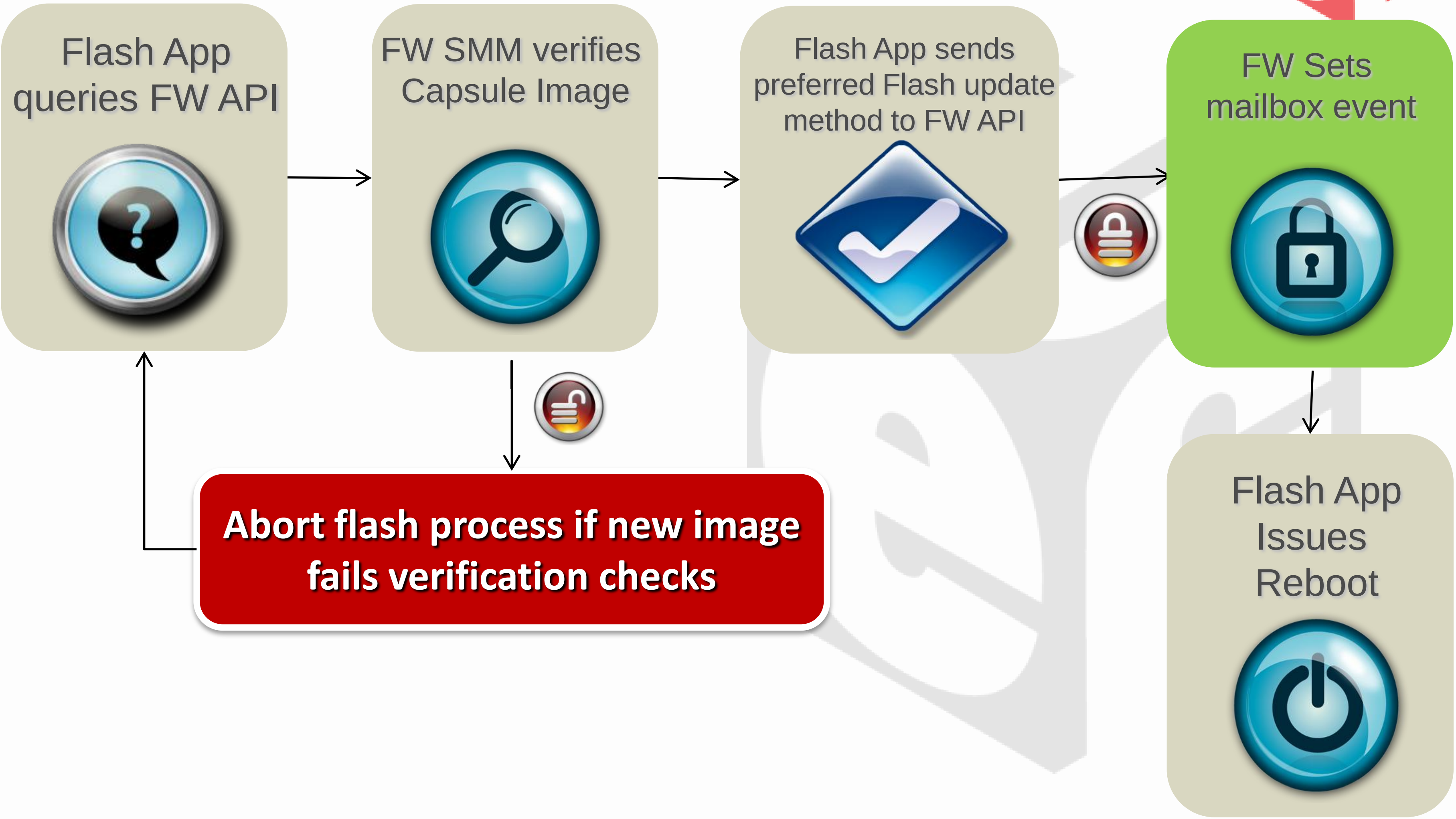
- All methods implemented using capsules defined by UEFI
- *Capsule ("Capsule-in-Memory")*



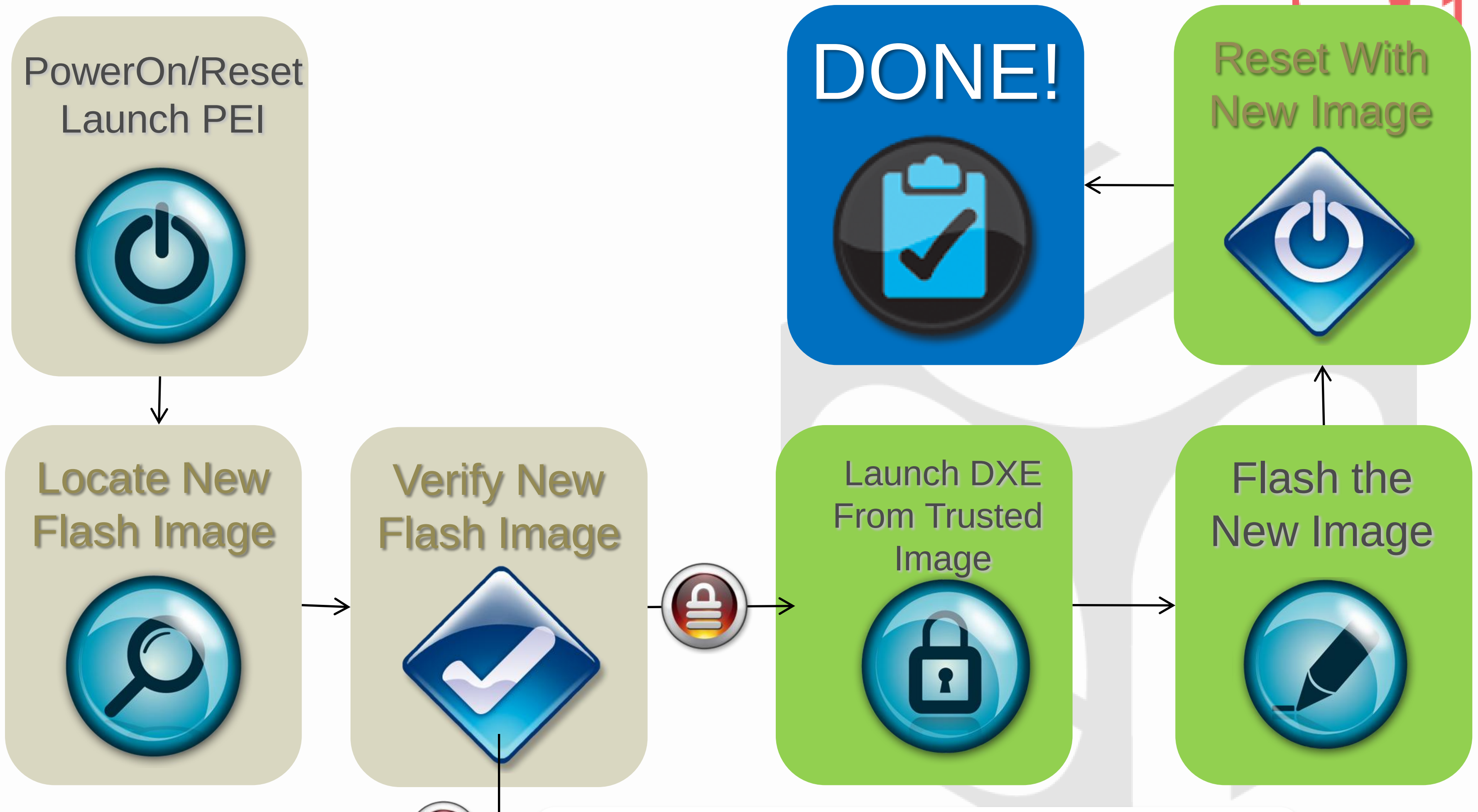
  - A capsule is put in memory by an application in the OS
  - Mailbox event is set to inform BIOS of pending update
  - System reboots, verifies the capsule image and update is performed by the BIOS
- *Recovery ("Capsule-on-Disk")*

  - Capsule is stored on a predefined disk in the OS
  - Mailbox event is set to inform BIOS of pending update
  - System reboots, loads the image from the disk, verifies the image and an update is performed by the BIOS

Secure Flash Update Process


Flash App queries FW API
FW SMM verifies Capsule Image
Flash App sends preferred Flash update method to FW API
FW Sets mailbox event
Flash App Issues Reboot
Abort flash process if new image fails verification checks

Secure Flash Update Process



PowerOn/Reset Launch PEI

Locate New Flash Image

Verify New Flash Image

Launch DXE From Trusted Image

Flash the New Image

Reset With New Image

DONE!

Abort flash process if image fails authentication

Agenda

- Background Information
- Methodology
- Implementation
- **Demonstration**
- Call to Action

Secure Flash Demonstration

- The following will be demonstrated:
  - The capsule update method using AMI ASFU (AMI Secure Flash Update) Utility
  - Anti-Rollback will be tested by trying to flash original image
  - A modified binary will be used to simulate a malicious BIOS update
    - A binary modified after signing will have an invalid signature

Agenda

- Background Information
- Methodology
- Implementation
- Demonstration
- **Call to Action**

Call to Action

- Review chapter 27 of the UEFI specification (Security – Secure Boot, Driving Signing and Hash)
  - Concentrate on the interfaces concerned with image authentication
- Review the BIOS Protection Guidelines by NIST
  - NIST special publication 800-147 (BIOS Protection Guidelines)
- Ensure all system firmware meets requirements of both specifications

Thanks for attending the UEFI Fall Plugfest 2012

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by

American Megatrends