UEFI Security Enhancements

UEFI Fall Plugfest – October 24-27, 2011
Presented by Kevin Davis (Insyde Software)
Agenda

- Introduction
- Authenticated Variables
- Driver Signing
- System Defined Variables
- Secure Boot value
- Demo
- Questions
Introduction: Why UEFI Secure Boot?

- Current OSs have improved virus resistance
- Microsoft Windows 8 improves even more
- Legacy BIOS has become the latest malware target

Called Mebromi, the malware is reminiscent of the IceLord proof of concept BIOS rootkit in 2007, was a late 1990s virus that was able to erase the motherboard software. This new rootkit is a different caliber as it appears to be one of the most persistent malware programs we have heard so far. – Tom’s Hardware (Sept 15, 2011)

Are BIOS rootkits a real threat? Yes, we can consider Mebromi the first real BIOS rootkit incident discovered in the wild – let’s consider IceLord BIOS rootkit more a proof of concept. -- webroot threat blog (Sept. 13, 2011)

- UEFI 2.3.1 Secure Boot
  - Software identity checking at every step of boot – Platform Firmware, Option Cards, and OS Bootloader

- Secure Boot is a Windows 8 requirement!
UEFI 2.3.1 Specification Update

- Security
  - Authenticated Variable Update Changes
  - Key Management Service (KMS)
  - Storage Security Command Protocol for encrypted HDD

- Network
  - Netboot6 client use DUID-UUID to report platform identifier
  - New FC and SAS Device Path

- Interoperability
  - FAT32 data region alignment
  - HII clarification & update
  - HII Modal Form

- Performance
  - Non-blocking interface for BLOCK oriented devices

- Technology
  - USB 3.0

- Maintenance
  - User Identifier, etc.

UEFI 2.3.1 Enabling More Security Support
## Secure Boot compared to Measured Boot

<table>
<thead>
<tr>
<th>Security Function</th>
<th>Secure Boot</th>
<th>Measured Boot (TCG - TPM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help BIOS verify OS is OK</td>
<td>Help OS verify BIOS and OS Boot paths are unchanged</td>
<td></td>
</tr>
<tr>
<td>Scan boot path and hash all BIOS code</td>
<td>No</td>
<td>Yes reduce BIOS attack surface</td>
</tr>
<tr>
<td>Check OS boot loader for unauthorized replacement or modification?</td>
<td>Yes reduce OS boot attack surface</td>
<td>Yes</td>
</tr>
<tr>
<td>Easy for End User to update system</td>
<td>Yes New version must be signed by someone in KEK or db</td>
<td>No Measured boot must be manually turned off to update</td>
</tr>
<tr>
<td>TPM Required?</td>
<td>No</td>
<td>Yes store measurements in TPM PCRs</td>
</tr>
</tbody>
</table>

- **Secure Boot**: Helps the BIOS verify that the OS is OK.
- **Measured Boot (TCG - TPM)**: Helps the OS verify that the BIOS and OS Boot paths are unchanged.
UEFI Secure Boot Overview

- System Firmware Store is the ‘root of trust’
  - Firmware is hardware-protected
  - All Firmware Updates must be a secure process

- UEFI 2.3.1 provides Building Blocks

- BIOS implements Secure Boot using the Building Blocks in the UEFI Specification
Building Blocks

1. Authenticated Variables

2. Driver Signing

3. System Defined Variables
UEFI Authenticated Variables

- Uses standard UEFI Variable Functions
- Available Pre-boot and also Runtime
- Typically stored in Flash
- Variable Creator signs Variable Hash with Private Key (PKCS-7 format)
- Signature & Variable passed together for Create, Replace, Extend, or Delete
- Several System defined variables for Secure Boot

Extensible Integrity Architecture
Updating Authenticated Variables

• Support for Append added (UEFI 2.3.1)
• Counter-based authenticated variables (UEFI 2.3)
  – Uses monotonic count to protect against suspicious replay attack
  – Hashing algorithm – SHA256
  – Signature algorithm – RSA-2048
• Time-based authenticated variable (UEFI 2.3.1)
  – Uses timestamp as rollback protection mechanism
  – Hashing algorithm – SHA256
  – Signature algorithm – X.509 certificate chains
    • Complete X.509 certificate chain
    • Intermediate certificate support (non-root certificate as trusted certificate)
Building Blocks

1. Authenticated Variables

2. Driver Signing

3. System Defined Variables
UEFI Driver Signing

• UEFI Driver Signing utilizes Microsoft Authenticode Technology to sign UEFI executable

• Secure Boot should check these signatures ...
  – UEFI Drivers loaded from PCI-Express cards
  – Drivers loaded from mass storage and USB
  – UEFI Shell apps (example: BIOS update utilities)
  – UEFI OS Boot loaders

• UEFI Signing is not required for ...
  – Drivers in the factory BIOS
  – Legacy components used only during legacy boots
**UEFI Driver Signing Process**

**Signing – by the creator:**

1. **Input**: Driver or Program
2. **Hash Function (SHA256)**: Hash
3. **Encrypt Hash Using Signer’s Private Key**: Signature
4. **Attach to Program**: Digitally Signed Program

**Verification – In the PC:**

1. **Hash Function**: Hash
2. **Decrypt Hash with Signer’s Public Key**: Hash
3. **Check local databases for certificate. If certificate found and not revoked, run UEFI Executable.**
1. Authenticated Variables

2. Driver Signing

3. System defined Variables
## Secure Boot Authenticated Variables

**Notes:**
- Owner of certificate in KEK can update db, dbx
- Owner of certificate in PK can update KEK

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td><strong>Platform Key</strong> – Root key set to enable Secure Boot</td>
</tr>
</tbody>
</table>
| KEK      | **Key Exchange Key**  
List of Cert. Owners with db, dbx update privilege |
| db       | List of Allowed Driver or App. Signers (or hashes) |
| dbx      | List of Revoked Signers (or hashes) |
| SetupMode | 1 = in Setup Mode, 0 = PK is Set (User Mode) |
| SecureBoot | 1 = Secure Boot in force |
Building Blocks

1. Authenticated Variables

2. Driver Signing

3. System Defined Variables

= InsydeH2O Secure Boot
Thanks for attending the UEFI Fall Plugfest 2011

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

**Welcoming Remarks** — Aven Chuang, Insyde Software

**UEFI Forum Updates** — Dong Wei, VP of the UEFI Forum

**Tips for UEFI Driver Compatibility** — Stefano Righi, American Megatrends, Inc.

**Understanding Platform Requirements for UEFI HII** — Brian Richardson, Intel Corporation

**UEFI Security Enhancements** — Kevin Davis, Insyde Software

**How to Protect the Pre-OS Environment with UEFI** — Tony Mangefeste, Microsoft

**Pre-OS Display Switching using GOP** — James Huang, AMD

**Debug Methodology Under UEFI** — Jack Wang, Phoenix Technologies

Download presentations after the plugfest at [www.uefi.org](http://www.uefi.org)
Backup Materials
Authenticode Format

Typical Windows PE File Format

<table>
<thead>
<tr>
<th>MS-DOS 2.0 Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE File Header</td>
</tr>
<tr>
<td>Optional Header</td>
</tr>
<tr>
<td>Windows-Specific Fields</td>
</tr>
<tr>
<td>Checksum</td>
</tr>
<tr>
<td>Data Directories</td>
</tr>
<tr>
<td>Certificate Table</td>
</tr>
</tbody>
</table>

Section Table (Headers)

| Section 1 |
| Section 2 |
| ...      |
| Section N |

Attribute Certificate Table

bCertificate binary array (contains Authenticode Signature)

Remaining Content

Blue: Objects describe the location of the Authenticode-related data

Objects omitted from the Authenticode hash value

Authenticode Signature Format

PKCS#7

contentInfo

Set to SPCIndirectDataContent, and contains:
PE File hash value
Legacy structures

Certificates
Includes:
• X.509 certificates for software publisher’s signature
• X.509 certificates for timestamp signature (optional)

SignerInfos

SignerInfo

Includes:
Signed hash of contentInfo
Publisher description and URL (optional)
Timestamp (optional)

Timestamp (optional)

A PKCS#9 counter-signature, stored as an unauthenticated attribute, which includes:
Hash value of the SignerInfos signature
UTC timestamp creation time
Timestamping authority signature
Secure Boot begins at the Factory

Pre-production

Certificate Generating Station @ OEM

1. OEM collects certificates provided by OSVs, Partners, and OEM’s own keys.
   “DB Generator” creates the Initial Security Load for new computers.

Production

Initial Security Load

2. Initial Security Load is installed onto each computer at the factory, enabling Secure Boot.
   1) Initial db and dbx
   2) KEK with allowed updaters
   3) Platform Key (PK)

User

OEM is responsible for Initializing Secure Boot
Secure Boot protects the End User

User attempts to boot a compromised system

OS Boot-loader image checked against pre-loaded database

Root-kit fails checks, user protected by Secure Boot

Secure Boot Tests Signatures to Reject Potential Threats