

Brendan Kerrigan  
Tamas K Lengyel

# Anti-Evil Maid with UEFI and Xen

Platform Security Summit, 2018



# Overview

1. Motivation
2. UEFI background
3. Anti-Evil Maid
4. Extending the SRTM chain
5. EFI\_LOAD\_OPTION
6. Launching DRTM after SRTM

# Motivation

## OpenXT

- Security hardened distribution
- Based on Xen & Linux
- Anti-Evil Maid using Intel TXT
- No boot-time security for non-Intel hw
- No UEFI / SecureBoot support

# UEFI/SecureBoot

## UEFI (2007)

- Successor of Intel's EFI specification
- Heavily influenced by Microsoft
- Reused PE binary format (ie. exe)

## SecureBoot (2011)

- Public-key crypto using signatures
- Heavily influenced by Microsoft

# UEFI/SecureBoot

1. Install public keys in firmware NVRAM
2. Sign binaries with private key
3. Firmware verifies signatures as software is loaded
4. Firmware only executes binaries with valid signatures

Doesn't use the TPM!

# UEFI/SecureBoot

Systems ship with OEM and Microsoft keys pre-loaded

What about Linux / FreeBSD / etc?

1. Require users to install custom SecureBoot keys
2. Get Microsoft to sign GRUB and/or every kernel?

Neither of these options is “easy”

# 3rd option: the shim

Small UEFI application with a new SecureBoot key embedded

Capable of checking signatures with embedded key

Can be signed by Microsoft

Shim key-owner can sign its own kernel updates independently

# Xen and the shim

Shim registers a service with the firmware

Shim loads & launches Xen

1. SecureBoot verification
2. Measures it into the TPM

Xen locates shim protocol and verifies dom0 kernel



# Authenticate..

1. ..the user to the computer
  - “Please enter your password”
2. ..the software to the computer
  - SecureBoot
3. ..the computer to the user
  - Anti-Evil Maid (AEM)

# AEM Requirements

Have to ensure that the system:

1. Says what it does
2. Does what it says
3. Can prove it

# AEM Proof options

## Using local encryption

1. Record state of a “good” system into the TPM
2. Encrypt (“seal”) <something> with the TPM
3. TPM only decrypts (“unseal”) <something> when state matches

## Using remote attestation

1. Record state of the system into the TPM
2. Send quote to remote machine to verify

# Measurements needed

## When to measure?

1. From the start
  - Static-root-of-trust (SRTM)
  - Usually considered too complex to be practical
2. After we are done booting
  - Dynamic-root-of-trust (DRTM)
  - Intel Trusted Execution Technology (TXT)

# DRTM

To be fully effective requires either:

1. Secure Transfer Monitor (STM)
  - Without it firmware is in the TCB
  - No OEM supports it
2. Combined with SRTM
  - Firmware is in the TCB

# SRTM

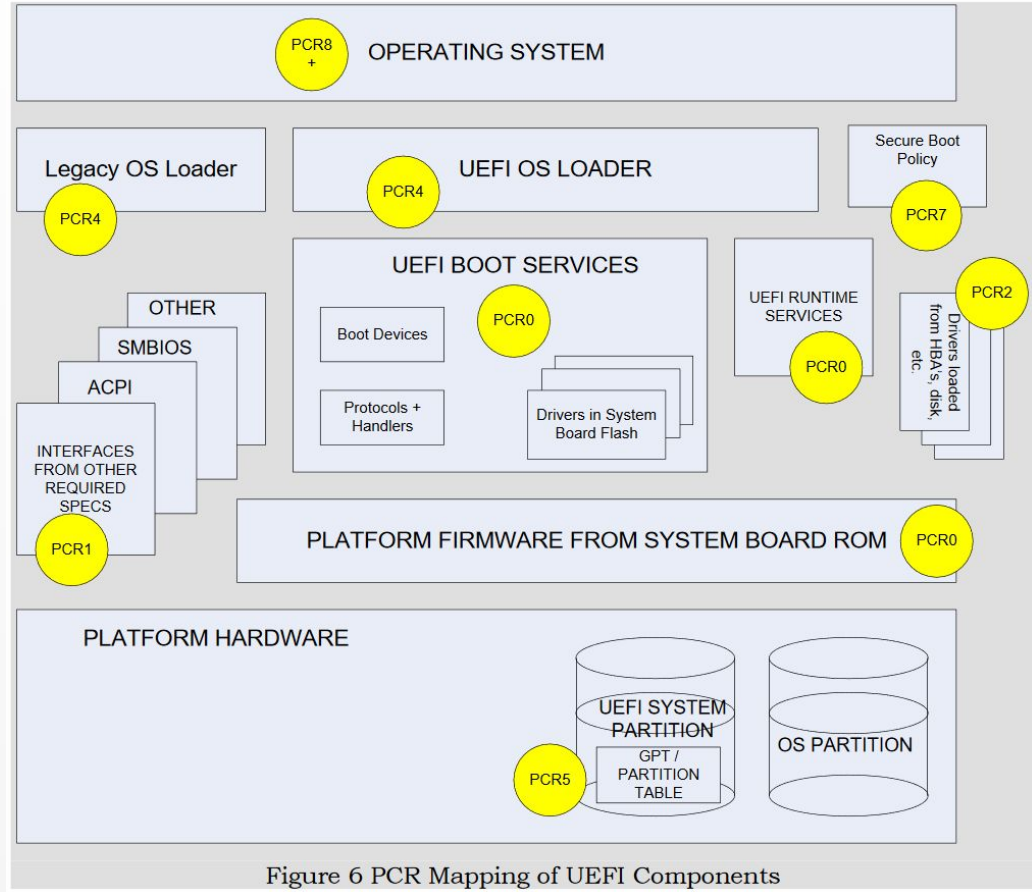


Figure 6 PCR Mapping of UEFI Components

# Extending the SRTM chain

## Need to measure

- Xen
- Xen command line
- XSM policy
- Dom0 kernel
- Dom0 initrd
- Dom0 command line
- Dom0 rootfs



Compile everything into static images



# Static compile-in options

## CONFIG\_CMDLINE\_OVERRIDE: Built-in command line overrides boot loader arguments

### General informations

The Linux kernel configuration item CONFIG\_CMDLINE\_OVERRIDE has multiple definitions:

### Built-in command line overrides boot loader arguments found in arch/tile/Kconfig

The configuration item CONFIG\_CMDLINE\_OVERRIDE:

- prompt: Built-in command line overrides boot loader arguments
- type: bool
- depends on: [CONFIG\\_CMDLINE\\_BOOT](#)
- defined in [arch/tile/Kconfig](#)
- found in Linux kernels: 2.6.36-2.6.39, 3.0-3.19, 4.0-4.15, 4.16-rc-HEAD

```
config XSM_POLICY
bool "Compile Xen with a built-in security policy"
default y if HAS_CHECKPOLICY = "y"
depends on XSM
```

---help---

This includes a default XSM policy in the hypervisor so that the bootloader does not need to load a policy to get sane behavior from an XSM-enabled hypervisor. If this is disabled, a policy must be provided by the bootloader or by Domain 0. Even if this is enabled, a policy provided by the bootloader will override it.

This requires that the SELinux policy compiler (checkpolicy) be available when compiling the hypervisor.

If unsure, say Y.

## CONFIG\_INTRAMFS\_SOURCE: Source directory of cpio\_list

### General informations

The Linux kernel configuration item CONFIG\_INTRAMFS\_SOURCE has multiple definitions:

### Initramfs source file(s) found in drivers/block/Kconfig

The configuration item CONFIG\_INTRAMFS\_SOURCE:

- prompt: Initramfs source file(s)
- type: string
- depends on: (none)
- defined in [drivers/block/Kconfig](#)
- found in Linux kernels: 2.6.11-2.6.13

### Help text

This may be either a single cpio archive with a .cpio suffix or a space-separated list of directories and files for building the initramfs image. A cpio archive should contain a filesystem archive to be used as an initramfs image. Directories should contain a filesystem layout to be included in the initramfs image. Files should contain entries according to the format described by the "tar\_get\_init\_cpio" program in the kernel tree.

If no multiple directories and files are specified then the initramfs image will be the aggregate of all of them.

See [file Documentation/initramfs-tools/README](#) for more details.

If you are not sure, leave it blank.

### config CMDLINE

```
string "Built-in hypervisor command string" if EXPERT = "y"
default ""
```

---help---

Enter arguments here that should be compiled into the hypervisor image and used at boot time. When the system boots, this string will be parsed prior to the bootloader command line. So if a non-cumulative option is set both in this string and in the bootloader command line, only the latter one will take effect.

# Extending the SRTM chain

1. UEFI loads, measures & verifies the shim
2. The shim loads, measures & verifies Xen
3. Xen loads dom0
4. Xen uses the shim to measure & verify dom0

This should “just work”

# Almost...

- Xen fails to launch when loaded by the shim
- Shim protocol only available with SecureBoot on
- The shim only measures with SecureBoot on
- The shim SecureBoot verifies dom0 but doesn't measure it
- The shim silently hides TPM failures

*Fixes upstreamed/under review*

# TPM2 failure on Dell

All measurements of PE images fail with PE\_COFF\_IMAGE flag

rhboot / shim

<> Code ⓘ Issues 17 🔄 Pull requests 8 📁 Projects 0 📖 Wiki 📊 Insign

## Fall-back TPM2 measurement if it fails with PE\_COFF\_IMAGE flag

Signed-off-by: Tamas K Lengyel <lengyel@ainfosec.com>

🔗 master 🏠 latest-release 15

👤 Tamas K Lengyel authored and vathpela committed on Nov 8, 2017 1 parent ba06a43

Family "2.0"  
Level 00 Revision 00.09

TCG Public Review  
Copyright © TCG 2015

Page 33  
July 17, 2015

### 6.6.5 Description

The EFI\_TCG2\_PROTOCOL Hash Log Extend Event function call calculates the measurement of a data buffer (possibly containing a PE/COFF binary image) and causes the EFI TCG2 protocol driver to extend the measurement. In addition, the service optionally creates an event log entry and appends it to the event log for each event log format supported by the service. The service allows a caller to make use of the TPM without knowing anything about specific TPM commands.

The use of this function to measure PE/COFF images must be done before relocations have been applied to the image. Note: Use caution using this method to measure PE/COFF images. Generally implementations that load PE/COFF images strip important data during the load process from the image and may change the image section alignment in memory. The net result is calculating the hash of an in-memory image does not match the actual measurement for the image as properly calculated when it is loaded from storage media.

# TPM2 failure with Dell

Measurement type will differ on good vs bad firmware

- PE\_COFF\_IMAGE=Authenticode; raw=sha256
- Pre-calculating expected PCR values not possible
- Custom patch to shim required to standardize on sha256
- Until all firmwares work with PE\_COFF\_IMAGE flag

```
commit 3ad44002b7387cbc4ffe0549f37b09d3b77b4333
Author: Tamas K Lengyel <tamas@tklengyel.com>
Date: Fri Mar 2 23:31:27 2018 -0700

    Don't measure with Authenticode

    On TPM2 devices calculating the Authenticode hash needs to be performed
    by the firmware. However, requesting measurements with PE_COFF_IMAGE flag
    fails on certain firmwares (Dell). For consistency we fall back to measuring
    the images as a whole without parsing the headers.
```

# Problems with built-in parts

Still possible to override from bootloader Xen's..

- ..built-in XSM policy
- ..built-in command line

Xen under UEFI can load the initrd for dom0

- Even if the dom0 kernel has one built-in

```
[ubuntu]
options=console=vga,com1 com1=115200,8n1 iommu=verbose ucode=scan flask=disabled conring_size=2097152 loglvl=all
kernel=linux-4.8.0-generic root=UUID=3f1e35fb-9907-48d1-b621-42369d5ad88f ro quiet vt.handoff=7 console=hvc0
ramdisk=initrd.img-4.8.0-41-generic
```

# Problems with built-in parts

External initramfs images:  
-----

If the kernel has `initrd` support enabled, an external `cpio.gz` archive can also be passed into a 2.6 kernel in place of an `initrd`. In this case, the kernel will autodetect the type (`initramfs`, not `initrd`) and extract the external `cpio` archive into `rootfs` before trying to run `/init`.

This has the memory efficiency advantages of `initramfs` (no `ramdisk` block device) but the separate packaging of `initrd` (which is nice if you have non-GPL code you'd like to run from `initramfs`, without conflating it with the GPL licensed Linux kernel binary).

It can also be used to supplement the kernel's built-in `initramfs` image. The files in the external archive will overwrite any conflicting files in the built-in `initramfs` archive. Some distributors also prefer to customize a single kernel image with task-specific `initramfs` images, without recompiling.

<https://www.kernel.org/doc/Documentation/filesystems/ramfs-rootfs-initramfs.txt>



If we have to start patching Xen and Linux to make it work,  
we might as well reconsider our approach



Measure everything as we boot  
(dealing with static command lines is a pain anyway)

# Shim Measure

We already use the shim to measure & verify dom0

- The shim already implements TPM extend functions

Expose TPM functions through the shim protocol

- Re-use well tested wrappers

# Shim Measure

```
1842 +EFI_STATUS shim_measure (void *buffer, UINT32 size, UINT8 pcr)
1843 +{
1844 +   EFI_STATUS status;
1845 +   PE_COFF_LOADER_IMAGE_CONTEXT context;
1846 +
1847 +   in_protocol = 1;
1848 +
1849 +   status = read_header(buffer, size, &context);
1850 +   if (status == EFI_SUCCESS) {
1851 +       status = tpm_log_pe((EFI_PHYSICAL_ADDRESS)(UINTN)buffer, size, NULL, pcr);
1852 +   } else {
1853 +       status = tpm_log_event((EFI_PHYSICAL_ADDRESS)(UINTN)buffer, (UINTN)size, pcr, (CHAR8 *)"shim_measure");
1854 +   }
1855 +
1856 +   in_protocol = 0;
1857 +
1858 +   return status;
1859 +}
```

Log as PE but don't measure  
with Authenticode

Log & measure as generic  
shim\_measure event

# Shim Measure








Xen can now measure non-PE components

- Config file (which includes Xen & dom0 command line)
- XSM policy
- dom0 initrd

```
1237 +
1238 +     efi_bs->LocateProtocol(&shim_lock_guid, NULL, (void **)&shim_lock);
1239 +
1240 +     if ( shim_lock &&
1241 +         (status = shim_lock->Measure(cfg.ptr, cfg.size, 4)) != EFI_SUCCESS )
1242 +         PrintErrMsg(L"Configuration file could not be measured", status);
1243 +
```

# Extending the SRTM chain

## Measurements now include

- Xen 
- Xen command line 
- XSM policy 
- Dom0 kernel 
- Dom0 initrd 
- Dom0 command line 
- Dom0 rootfs 

# Linux IMA

Integrity Measurement Architecture (IMA)

Linux measure all files touched (r/w/x) by root into the TPM

Nondeterministic when multiple cores are used to boot

- The order in which files get measured change the hash
- Not good for sealing/remote attestation
- Perhaps TPM2 non-brittle PCRs could make it work?
- Not ideal if TPM1.2 devices still need to be supported

# Read-only rootfs necessary

Create separate /home, /var, /srv

- <https://wiki.debian.org/ReadOnlyRoot>

Alternatively, apply AUFS based temp layer on top of /

- <https://help.ubuntu.com/community/aufsRootFileSystemOnUsbFlash>

Measure static root into the TPM from initrd!

SRTM chain is now complete!



# Anti-Evil Maid proof

Use TPM measurements to prove system authenticity

Use locally sealed secret

- Needs to be unique and hard to spoof
- Spoofing a green check-mark is pretty easy (OpenXT)

Remote attestation

- Requires remote system to verify TPM quote
- Can be via network or with a QR code (TPM-TOTP)

# No GRUB!

How do we set boot options?

- Necessary for having debug options for dom0 and/or Xen
- Or to deploy a single Xen efi config file with many targets

# EFI\_LOAD\_OPTION

UEFI has built-in support

- Record boot option in NVRAM

```
# efibootmgr -c -d /dev/sda -p 1 -w -L "Xen (normal)" -l \EFI\xen\shim.efi -u "normal"
```

```
# efibootmgr -c -d /dev/sda -p 1 -w -L "Xen (debug)" -l \EFI\xen\shim.efi -u "debug"
```

UEFI measures the whole struct into the TPM

- Including the OptionalData

```
EFI_LOAD_OPTION
{
  UINT32 Attributes;
  UINT16 FilePathListLength;
  CHAR16 Description[];
  EFI_DEVICE_PATH_PROTOCOL FilePathList[];
  UINT8 OptionalData[];
}
```

# EFI\_LOAD\_OPTION

OptionalData selects section from config file

```
[global]  
default=normal
```

```
[normal]
```

```
options=console=vga  
kernel=vmlinuz-4.8.0-41-generic-signed root=/dev/sda2 ro quiet console=hvc0  
ramdisk=initrd.img-4.8.0-41-generic
```

```
[debug]
```

```
options=console=vga,com1 com1=115200,8n1,pci iommu=verbose loglvl=all guest_loglvl=all  
kernel=vmlinuz-4.8.0-41-generic-signed root=/dev/sda2 ro quiet console=hvc0  
ramdisk=initrd.img-4.8.0-41-generic
```

# EFI\_LOAD\_OPTION

## [Xen-devel] [PATCHv3] xen: Add EFI\_LOAD\_OPTION support

- **To:** [xen-devel@xxxxxxxxxxxxxxxxxxx](mailto:xen-devel@xxxxxxxxxxxxxxxxxxx)
- **From:** Tamas K Lengyel <[tamas@xxxxxxxxxxxxxxxx](mailto:tamas@xxxxxxxxxxxxxxxx)>
- **Date:** Mon, 22 Jan 2018 17:21:04 -0700
- **Cc:** [openxt@xxxxxxxxxxxxxxxx](mailto:openxt@xxxxxxxxxxxxxxxx), Tamas K Lengyel <[tamas@xxxxxxxxxxxxxxxx](mailto:tamas@xxxxxxxxxxxxxxxx)>, Jan Beulich <[jbeulich@xxxxxxxx](mailto:jbeulich@xxxxxxxx)>, Tamas K Lengyel <[lengyel@xxxxxxxxxxxxxxxx](mailto:lengyel@xxxxxxxxxxxxxxxx)>
- **Delivery-date:** Tue, 23 Jan 2018 00:22:52 +0000
- **List-id:** Xen developer discussion <xen-devel.lists.xenproject.org>

When booting Xen via UEFI the Xen config file can contain multiple sections each describing different boot options. It is currently only possible to choose which section to boot with if the buffer contains a string. UEFI provides a different standard to pass optional arguments to an application, and in this patch we make Xen properly parse this buffer, thus making it possible to have separate EFI boot options present for the different config sections.

Signed-off-by: Tamas K Lengyel <[lengyel@xxxxxxxxxxxxxxxx](mailto:lengyel@xxxxxxxxxxxxxxxx)>

---

Cc: Jan Beulich <[jbeulich@xxxxxxxx](mailto:jbeulich@xxxxxxxx)>

Cc: [openxt@xxxxxxxxxxxxxxxx](mailto:openxt@xxxxxxxxxxxxxxxx)

## Ack pending..

<https://lists.xenproject.org/archives/html/xen-devel/2018-01/msg01955.html>

Now that the SRTM chain is complete,  
can we do a DRTM?

Yes!

# Launching tboot\*

1. Load tboot from xen.efi & measure with shim
2. Load second copy of Xen & measure+verify with shim
3. Build multiboot struct in xen.efi pointing to second copy of Xen
4. Launch tboot using multiboot struct

No more “gap”!

\*Joint work with Daniel De Graaf

# Status

SRTM patches merged to OpenXT, DRTM patches posted

Shim patches upstreamed/under review

Xen patch posted, Ack pending

General instructions, patches and scripts posted at

- <https://github.com/tklengyel/xen-uefi>



# Thanks!



Tamas K Lengyel  
lengyelt@ainfosec.com  
@tklengyel

Brendan Kerrigan  
kerriganb@ainfosec.com