(Ab)using Linux as a Trusted Bootloader

Eric Richter
erichte@linux.vnet.ibm.com
Overview

1 Background Info
   • TPM
   • Trusted Boot

2 Petitboot

3 “Trusted” Petitboot

4 Kernel Changes

5 Questions
Background Information
Trusted Platform Module (TPM)

• Small, inexpensive security cryptoprocessor
• Contains a bank of “append only” data registers called PCRs

TPM 1.2 Figure by Guillaume Piolle
Trusted Boot

Start (RTM)

A
Trusted Boot

1 Measure next component

Start (RTM)

A → B

1
Trusted Boot

1. Measure next component
2. Log the Event

Start (RTM)

A → B → Log

(Ab)using Linux as a Trusted Bootloader
Trusted Boot

1. Measure next component
2. Log the Event
3. Extend PCR
Trusted Boot

1. Measure next component
2. Log the Event
3. Extend PCR
4. Transfer Execution

Start (RTM)

TPM

A

4

B

Log

(Ab)using Linux as a Trusted Bootloader
Trusted Boot

1. Measure next component
2. Log the Event
3. Extend PCR
4. Transfer Execution
5. Repeat

(Ab)using Linux as a Trusted Bootloader
Trusted Boot on OpenPOWER

Ab)using Linux as a Trusted Bootloader

- Hostboot
- Skiboot
- Log
- Petitboot
- TPM
- Target OS
What is Petitboot?
UEFI + GRUB

GRUB bootloader

- UDP/IP

UEFI firmware

- SCSI layer
- disk driver
- FAT32 fs
- TCP/IP
- net driver
- DHCP client
- USB driver
- gpu driver
- PCI layer

Disk

Net
Reinvented?

Linux
kernel

SCSI
layer
disk
driver
FAT32
fs
TCP
/IP
net
driver
DHCP
client
USB
driver
gpu
driver
PCI
layer

(Ab)using Linux as a Trusted Bootloader
Petitboot

Linux Kernel based bootloader via kexec
- Reuse kernel drivers
- Provides userspace, simplifying development
- Platform independence
- Well tested ;)

(Ab)using Linux as a Trusted Bootloader
kexec_file_load(...)
Kexec

reboot(LINUX_REBOOT_CMD_KEXEC)

Kernel A

execute

Kernel B
Kexec

Kernel B
“Trusted” Petitboot?
Petitboot is a userspace application, take the measurements in there!

Except...

- Need some form of TCG Software Stack (TSS)
- Need write access to the event log
- Exit to shell → manual kexec
Measure in kexec-{tools,lite}...

Okay, so modify the kexec binary to take the measurement.

Well, that fixes the manual kexec issue, but...

- Still needs some kind of TSS
- Still need to append the log
"Trusted" Petitboot?

Trusted measurements provide a new set of challenges in a kernel environment.

Would need...

- ...a TPM driver
- ...a way to interface with the TPM
- ...a way to measure the next component
- ...a way to log the measurement
- ...for this to be upstreamable

If only we had an integrity subsystem...
"Trusted" Petitboot.

Oh wait, there’s IMA.

IMA can handle...

• ...talking to the TPM via the device driver
• ...extending the data
• ...measuring the next component
• ...logging what was measured
• ...some small modifications
Measures and appraises files based on set policy rules. Example rule to measure files opened by root:

```
measure func=FILE_CHECK mask=MAY_READ uid=0
```

Logs them like so:

```
10 1d8d532d463c9f8c205d0df7787569a85f93e25a  lma-ng  sha1:00000000000000000000000000000000 boot_aggregate
10 ef411bae164fd624ea94fc9ef02f092c82d78dcd  lma-ng  sha1:bbe98e29b850f3907611fb963545e607a9179f4  /init
10 5a793eedd4512ea7cb589adccae4d3d894eb648a  lma-ng  sha1:cbad4f29908120251837d0b173878b71dfc53fba5  /bln/busybox
10 52c1930e65d9e96cc58b5cfab6fc85175c66dc4e5  lma-ng  sha1:8fb2fcb2e0cc752cf15c9712ba039e97793703cd1  /lib64/ld64.so.2
10 97a68f05ce0c81aa32e9be0543f6d6a0245d88d4  lma-ng  sha1:1e763bf47b038f72c289c019aa5e92c721a1285a  /etc/ld.so.cache
```
Well, it’s not a perfect solution.

Still need...

1. IMA Hook for measuring kernel, initramfs
2. Support for extending alternate PCRs
3. Preservation of the IMA log across kexec
4. Device tree passing and measurement?
1. Kernel & Initramfs Hooks

**Problem:** Need to measure kernel and initramfs

**Solution:** As of 4.6, these are already included!

Example policy:

```bash
measure func=KEXEC_KERNEL_CHECK
measure func=KEXEC_INITRAMFS_CHECK
```
2. Extending Other PCRs

**Problem:** IMA measures into a PCR as defined by Kconfig.

**Solution:** Add flexibility into the policy

```
measure func=FILE_CHECK mask=MAY_READ uid=0 pcr=11
```

**Included in 4.8**
3. Preserving the IMA Log

**Problem:** How can measurements from a previous kernel be validated?

**Solution:**

1. Serialize the IMA log
2. Store in the kexec image
3. (reboot)
4. IMA restores log from physical memory
Implementation for POWER

1. kexec_file_load:
   - Allocate buffer (kexec segment) for log serialization
   - Create device tree entry for the start and ending address
   - Set reboot hook

2. Reboot hook:
   - Serialize the log
   - Store into kexec image

3. Immediately Before Purgatory:
   - Move log from kexec image to physical memory address

4. New kernel boot:
   - IMA checks for device tree entry
   - If exists, restores log
4. Device Tree

**Problem:** Need to measure and pass the device tree through `kexec_file_load`

**Solution:** 

**Current questions:**
- What should be measured? (or can be?)
- What should the userspace be allowed to supply?
Questions?