Boot strapping slightly more secure systems with coreboot and Heads

Trammell Hudson @qrs

A6C7 4E34 1054 A169 CE52
BE5F B65B FE54 0DEF 86C0
Building, not breaking: Magic Lantern

Press 'r' for recovery shell: r

!!!!!! User requested recovery shell

!!!!!! Starting recovery shell

New value of PCR[4]: 8a6a96fde1a8dd96271479dc40742b36aba3c2b3

/bin/ash: can't access tty; job control turned off

# qubes-install
Why firmware security is important
Firmware can subvert all OS protections (Intel ATR)
UEFI Vendors have shipped malware used shady 'rootkit' tactic to quietly reinstall unwanted software

Even when users reinstalled a clean version of Windows on some devices, the software would still reappear.

Hacking Team's malware uses a UEFI rootkit to survive operating system reinstalls

The feature allows the company's software to persist even if the hard disk drive if replaced.

By Lucian Constantin | Follow
Romania Correspondent, IDG News Service
Jul 14, 2015 6:56 AM PT
DE MYSTERIIS DOM JOBSIVS MAC EFI ROOTKITS

SNARE
@ BLACK HAT USA
JULY 2012
assurance

(U)Sonic Screwdriver v1.0
User’s Guide
November 29, 2012
BIOS Software Supply Chain Breakdown

Definition: IBV

- Independent BIOS Vendors are 3rd-party UEFI developers that sell value-added UEFI, toolkits, and custom development services.
The problem with EFI is that it actually superficially looks much better than the BIOS, but in practice it ends up being one of those things where it has few real advantages, and often just a lot of extra complexity because of the "new and improved" interfaces that were largely defined by a committee.

Not that I'd ever claim that the BIOS is wonderful either.

Linus
Open and Flexible

Simple and Understandable

Reproducible and Measured
Heads is:

coreboot + Linux Kernel in ROM

(plus Security research and tools)
Computer Scientists Just Super

from PAGE B1

Computing Systems group.

“We build the biggest machines and we’ve always built the biggest machines,” said University of New Mexico computer science professor Barney MacCabe, a consultant to the lab and long-time member of the team.

Since 1993, Sandia has repeatedly come to the team with the same problem: Take a pile of blazing fast computer hardware and write the software plumbing to turn it into a well-oiled machine.

Three times in the last decade the group has succeeded doing what few others can claim, turning those piles of hardware into the fastest computer in the world.

SOFTWARE TRIO: Ron Brightwell, left, Rolf Riesen and Tramm Hudson worked the software magic needed to get the new Cplant supercomputer working at Sandia National Laboratories.
coreboot Stages

- TCB
  - Bootblock
    - Prepare Cache-as-RAM and Flash access
  - ROM Stage
    - Memory and early chipset init
  - RAM Stage
    - Device enumeration and resource assignment
    - ACPI Table creation
    - SMM Handler
  - Payload

(Also the TPM)

TCB:
- 1.5k
- 70k
- 80k
- 4MB
Run './start-xen' to load the hypervisor
Run 'kexec -e' to boot it

TPM TOTP:
[ 1.664441] random: unsealfile urandom read with 8 bits of entropy available
2016-11-03 11:45:29: 438116

/bin/ash: can't access tty; job control turned off
/ # [ 2.5205251] clocksource: Switched to clocksource tsc
uname -a
Linux (none) 4.7.0-heads #17 SMP Fri Oct 28 10:27:26 EDT 2016 x86_64 GNU/Linux
Linux: More eyes

EDK II http://www.tianocore.org/edk2/

- 20,378 commits
- 8 branches
- 0 releases
- 110 contributors

Linux kernel source tree

- 634,133 commits
- 1 branch
- 485 releases
- Contributors
- GPL-2.0
Chipset

Arrow keys navigate the menu. <Enter> selects submenus --> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

*** SoC ***
*** CPU ***
[*] Enable VMX for virtualization
   Include CPU microcode in CBFS (Generate from tree) -->
   Microcode binary path and filename (NEW)
   Northbridge ***
   -*- Use native raminit
   Southbridge ***

[ ] Flash ROM locking on S3 resume (Lock all flash ROM sections on S3 resume) -->
   Super I/O ***
   Embedded Controllers ***
   Intel Firmware ***
[ ] Add Intel descriptor.bin file (NEW)
[*] Build with a fake IFD (NEW)
   BIOS Region Starting:Ending addresses within the ROM (NEW)
   ME/TXE Region Starting:Ending addresses within the ROM (NEW)
   GBE Region Starting:Ending addresses within the ROM (NEW)
   Platform Region Starting:Ending addresses within the Rom (NEW)
   Vboot non-volatile storage in CMOS. (NEW)
   v(+)

<Select> < Exit > < Help > < Save > < Load >
Reducing System Reboot Time With kexec

Andy Pfiffer
Open Source Development Labs, Inc.
15275 SW Koll Parkway - Suite H
Beaverton, OR 97006
503-626-2455x30
http://www.osdl.org/
andyp@osdl.org

Kexec was developed by Eric Biederman for use on diskless nodes with a fast-reboot requirement. Rather than restricting the feature to Linux-only bootable images, kexec was developed to support the rebooting of well-formed ELF binary images. In principle, a non-Linux kernel image could be restarted on the platform without using the platform firmware. On supported platforms, kexec can boot unmodified Linux kernel images.
#!/bin/sh

# Load the kernel, initrd and Xen hypervisor into RAM
kexec \
  -l \
  --module "${KERNEL} ${KERNEL_CMDLINE}" \
  --module "${INITRD}" \
  --command-line "${XEN_CMDLINE}" \
  "${XEN}" \
|| die "Unable to load kernel/xen"

# Shutdown Heads kernel and "reboot" into Xen
kexec -e

# If we make it here...
echo >&2 "Something is very wrong..."
exec /bin/sh
Protecting the boot process
adversary: nation state (NSA)

them:
- $$$$$$$$
- power of the law
- power of the beyond
- the law
- rational & amoral

you:
- all the encryption
- all the Tor
- become famous enough you can’t be secretly murdered

steph
@corcra

your threat model is not my threat model but your threat model is okay

RETWEETS 10  LIKES 33

7:52 AM - 1 Jun 2015
we'll be thinking: can we assure having good code and how the peripherals interfere here again: we will look at this SPI flash loaded on the platform
“The root cause is that firmware assumes hardware is trusted”

http://www.intelsecurity.com/advanced-threat-research/content/data/REConBrussels2017_BARing_the_system.pdf
Hardware-Hacker

Laptop zerlegt, Angriff abgewehrt

Mikrofon herausreißen, Schnittstellen kappen, Bios überschreiben: Um Computer vor Angreifern zu schützen, muss man brutal vorgehen. Wir haben den Eingriff gewagt - und einen Laptop "gehärtet".

Von Ole Reißmann
White Paper

A Tour beyond BIOS Using Intel® VT-d for DMA Protection in UEFI BIOS

(Yao and Zimmer, 2015)
Support for Intel IOMMU using DMA Remapping Devices

CONFIG_INTEL_IOMMU:

DMA remapping (DMAR) devices support enables independent address translations for Direct Memory Access (DMA) from devices. These DMA remapping devices are reported via ACPI tables and include PCI device scope covered by these DMA remapping devices.

Symbol: INTEL_IOMMU [=y]
Type: boolean
Prompt: Support for Intel IOMMU using DMA Remapping Devices

Location:
- Device Drivers
- IOMMU Hardware Support (IOMMU_SUPPORT [=y])

Defined at drivers/iommu/Kconfig:140
Depends on: IOMMU_SUPPORT [=y] && PCI_MSI [=y] && ACPI [=y] && (\x86 [=y] || IA64_GENERIC)
Selects: IOMMU_API [=y] && IOMMU_IOVA [=y] && DMAR_TABLE [=y]
Loading kernel cache file 'System\Library\Caches\kernelcache'...

root device uuid is '7A18BC97-4624-3FE9-A158-41D2

++++ ExitBootServices ++++

***** Password: '2pwtwo!\x000D'

Starting OS... 10 0F 0E 0D 0C 0B 0A 09 08 07 06 05

Thunderbolt device with OptionROM exploit

File Vault password

The problem was further demonstrated by @qrs's Thunderstrike (1), Thundergate.io, and our joint work with Trammell on Thunderstrike 2 [3/x]

Thunderbolt OROMs have been a constant source of problems, since 2012 when @snare (now also at Apple :) used them to bootkit macs [2/x]
CONFIG_ON_DEVICE_ROM_LOAD:

Load Option ROMs stored on PCI/PCIe/AGP devices in coreboot.

If disabled, only Option ROMs stored in CBFS will be executed by coreboot. **If you are concerned about security, you might want to disable this option, but it might leave your system in a state of degraded functionality.**

When using a SeaBIOS payload it runs all option ROMs with much more complete BIOS interrupt services available than coreboot, which some option ROMs require in order to function correctly.

If unsure, say N when using SeaBIOS as payload, Y otherwise.

**Symbol:** ON_DEVICE_ROM_LOAD [=n]
**Type:** boolean
**Prompt:** Load Option ROMs on PCI devices
**Location:**
# Table 5. Protection Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Status register condition</th>
<th>WP# and SRWD bit status</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software protection mode (SPM)</td>
<td>Status register can be written in (WEL bit is set to “1”) and the SRWD, BP3-BP0 bits can be changed</td>
<td>WP#=1 and SRWD bit=0, or WP#=0 and SRWD bit=0, or WP#=1 and SRWD bit=1</td>
<td>The protected area cannot be program or erase.</td>
</tr>
<tr>
<td>Hardware protection mode (HPM)</td>
<td>The SRWD, BP3-BP0 of status register bits cannot be changed</td>
<td>WP#=0, SRWD bit=1</td>
<td>The protected area cannot be program or erase.</td>
</tr>
</tbody>
</table>
Where is firmware found?

- In peripherals
  - GPU, wireless, network, storage, USB etc...
- In system controllers & integrated processors on x86
  - Intel
    - Management Engine/AMT etc... (ME, ARC, SPARC)
  - AMD

ME: High-level overview

- Management Engine (or Manageability Engine) is a dedicated microcontroller on all recent Intel platforms
- In first versions it was included in the network card, later moved into the chipset (GMCH, then PCH, then MCH)
- Shares flash with the BIOS but is completely independent from the main CPU
- Can be active even when the system is hibernating or turned off (but connected to mains)
- Has a dedicated connection to the network interface; can intercept or send any data without main CPU's knowledge
python me_cleaner.py -r -d -o clean_flash.bin flash.bin

Output:

Full image detected
The ME/ TXE region goes from 0x3000 to 0x4f000
Found FPT header at 0x3010
Found 23 partition(s)
Found FTPR header: FTPR partition spans from 0x183000 to 0x183f000
ME/ TXE firmware version 8.130.1350
Removing extra partitions...
Removing extra partition entries in FPT...
Removing EFFS presence flag...
Removing ME/ TXE R/W access to the other flash regions...
Correcting checksum (0x7d)...
Reading FTPR modules list...

UPDATE (LZMA , 0x1cf4f2 - 0x1cf6b0): removed
ROMP (Huffman, fragmented data ): NOT removed
BUP (Huffman, fragmented data ): NOT removed
KERNEL (Huffman, fragmented data ): removed
POLICY (Huffman, fragmented data ): removed
HOSTCOMM (LZMA , 0x1cf6b0 - 0x1d648b): removed
RSA (LZMA , 0x1d648b - 0x1d66c0): removed
Protecting our secrets
% tpm read_pcr 9
0000000000000000000000000000000000000000
  Initial value

% echo -n hello | tpm extend 9
00629997206c7d587b4ed79aabc3db58c32e1492
  sha1(cat(0x0000... , sha1("hello")))

% echo -n world | tpm extend 9
112a61c687541ce9ff07cf6fc770cce7ec9df041
  sha1(cat(0x0062... , sha1("world")))
nv_readvalue
  -in "$TPM_INDEX"
  -sz "$TPM_SIZE"
  -of /tmp/sealed
 || die "Unable to read key from TPM NVRAM"

unsealfile
  -if /tmp/sealed
  -of "$key_file"
  -pwdd "$tpm_password"
  -hk 4000000000
 || die "Unable to unseal disk encryption key"

cryptsetup --key-file "$key_file" open "$dev"
 || die "$dev: Unable to decrypt with TPM"
#!/bin/bash
#
# Measure the cryptographic headers on the disk devices
#
for dev in "@$"; do
  if ! cryptsetup isLuks "${dev}"; then
    continue;
  fi

  cryptsetup luksDump "${dev}" \
    | extend -ix "${tpm_index}" \
    || die "${dev}: Unable to measure cryptographic header"
done
nv_readvalue
    -in "\$TPM_INDEX" \\
    -sz "\$TPM_SIZE" \\
    -of /tmp/sealed \\
|| die "Unable to read key from TPM NVRAM"

unsealfile
    -if /tmp/sealed \\
    -of "\$key_file" \\
    -pwdd "\$tpm_password" \\
    -hk 400000000 \\
|| die "Unable to unseal disk encryption key"

cryptsetup --key-file "\${key_file}" open "\${dev}" \\
|| die "\${dev}: Unable to decrypt with TPM"
Can the CPU executing the firmware that launched the bootloader that loaded the kernel running the software asking for your password be trusted?
Run './start-xen' to load the hypervisor
Run 'kexec -e' to boot it

Sun Jul 31 09:25:05 EDT 2016

Verify TPM PCR: 356705

/bin/ash: can't access tty; job
/ # [ 2.451809] clocksource
#!/bin/sh
# Retrieve the sealed file from the NVRAM, unseal it with the PCRs 
# and compute the TOTP value for the current time

nv_readvalue \
    -in ${tpm_index} \
    -sz 312 \
    -of /tmp/sealed \
|| die "Unable to retrieve sealed file from TPM NV"

unsealfile \
    -hk 400000000 \
    -if /tmp/sealed \
    -of /tmp/secret \
|| die "Unable to unseal totp secret"

echo -n "Verify TPM TOTP code: 

totp < /tmp/secret \
|| die "Unable to compute TOTP hash"
# ./ssss-combine -t 3 -n 5 -x
Enter 3 shares separated by newlines:
Share [1/3]: 4-98AF4F6627C90A8A21AFDE59314C2C1B
Share [2/3]: 5-AA92A78D51607E3A6B098BE243527269
Share [3/3]: 2-8DBA2716EF48139D5E2242D9D80C47CA
Resulting secret: C73A4AA4B1DC96AF5235C464BDFA86C7
Commits on Nov 3, 2016

*check PGP signatures on xen, kernel and initrd (partial fix for #43)*

osresearch committed 6 days ago

Commits on Sep 28, 2016

*ignore vdsos fake library on fedora*

osresearch committed on Sep 28

Commits on Sep 27, 2016

*some frequently asked questions, with draft answers*

osresearch committed on Sep 27
#!/bin/sh

XEN=/boot/xen-4.6.3.gz
INITRD=/boot/initramfs-4.4.14-11.pvops.qubes.x86_64.img
KERNEL=/boot/vmlinux-4.4.14-11.pvops.qubes.x86_64

# Check signature files
for file in XEN INITRD KERNEL; do
gpgv "${file}.asc" "${file}" || die "${file} signature failed"
done

tpm read_counter -ix 2 | gpgv /boot/gen.asc - \\
|| die "Rollback detected: "

ekexec -l \\
--module "${KERNEL} rd.luks.keyfile=/secret.key" \\
--module "${INITRD}" \\
--command-line "no-real-mode reboot=no console=vga" \\
${XEN} \\
|| die "Kernel load failed"

ekexec -e \\
|| die "Kernel exec failed"
#!/bin/sh

XEN=/boot/xen-4.6.3.gz
INITRD=/boot/initramfs-4.4.14-11.pvops.qubes.x86_64.img
KERNEL=/boot/vmlinux-4.4.14-11.pvops.qubes.x86_64

gpgv "$XEN.asc" "$XEN" || die "Xen signature failed"
gpgv "$INITRD.asc" "$INITRD" || die "Initrd signature failed"
gpgv "$KERNEL.asc" "$KERNEL" || die "Kernel signature failed"

tmp read_counter -ix 2 | gpgv /boot/gen.asc - \
|| die "Rollback detected:"

kexec -l \ 
    --module "$KERNEL" rd.luks.keyfile=/secret.key" \ 
    --module "$INITRD" \ 
    --command-line "no-real-mode reboot=no console=vga" \ 
    "$XEN" \ 
|| die "Kernel load failed"

kexec -e \ 
|| die "Kernel exec failed"
**Verified Boot**

Android 4.4 and later supports verified boot through the optional device-mapper-verity (dm-verity) kernel feature, which provides transparent integrity checking of block devices. dm-verity helps prevent persistent rootkits that can hold onto root privileges and compromise devices. This feature helps Android users be sure when booting a device it is in the same state as when it was last used.

A public key is included on the boot partition, which must be verified externally by the OEM. That key is used to verify the signature for that hash and confirm the device's system partition is protected and unchanged.
Qubes-certified hardware should run only open-source boot firmware (aka “the BIOS”), such as coreboot.

allow reproducible builds of xen.gz

The mkelf32 executable was using an uninitialized stack buffer for
padding after the enhdr and phdr are written to the xen file, which
leads to non-deterministic bytes in the binary and prevented Xen
hypervisors from being reproducibly built.

Additionally, the file was then compressed with gzip -9 without the
-n | --no-name flag, which lead to the xen.gz file having
non-deterministic bytes (the timestamp) in the compressed file.

Signed-off-by: Trammell Hudson <trammell.hudson@twosigma.com>
Reviewed-by: Jan Beulich <jbeulich@suse.com>
What's next?
<table>
<thead>
<tr>
<th>OEM Public Key Hash</th>
<th>9B 40 6E 27 DD 0E 4B 0C BD...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Manifest ID</td>
<td>0x1</td>
</tr>
<tr>
<td>Boot Guard Profile Config</td>
<td>Boot Guard Profile 4 - FVE</td>
</tr>
<tr>
<td>Force Boot Guard ACM Enabled</td>
<td>true</td>
</tr>
<tr>
<td>Verified Boot Enabled</td>
<td>true</td>
</tr>
<tr>
<td>Measured Boot Enabled</td>
<td>false</td>
</tr>
<tr>
<td>Protect BIOS Environment Enabled</td>
<td>true</td>
</tr>
<tr>
<td>Error Enforcement Policy</td>
<td>3</td>
</tr>
</tbody>
</table>

Enter raw hash string or certificate file. This is a 256-bit field representing the SHA-256 hash of another public key, used by the ACM to verify the Boot Policy manifest.

Choose the Boot Guard Profile:
- false = Disables the Boot Guard ACM to launch during platform boot; true = Force the Boot Guard profile.
- false = Platform does not perform verified boot; true = Platform performs verified boot.
- false = Platform does not perform measured boot; true = Platform performs measured boot.
- false = Take no actions to control the environment during execution of BIOS components; true = Boot Guard Error enforcement policy.
Securing Bare Metal Clouds

Jason Hennessey, Nabil Shear, Trammell Hudson, Orran Krieger, Gerardo Ravago, Kyle Hogan, Ravi S. Gudimetla, Larry Rudolph, Mayank Varia, Peter Desoyers, and Manuel Egele

Bare Metal Clouds & Trusted Hardware
Bare metal nodes provide performance and privacy advantages over virtual machines, but the direct hardware access they give opens up new attack vectors that must be addressed.

Threat Model
- Bare metal clouds expose the tenant to new attack vectors:
  - Embedded firmwares run early in the boot sequence (BIOS), are rarely updated, difficult to inspect, and persist between tenants
  - Tenants could flash malware into the firmware of a machine before returning it to the pool, affecting all future users of the machine
  - Any persistent state left on a machine by a tenant can be exploited by future attackers seeking to scrape secrets from disk or memory

A tenant is using the node and writes some data to the RAM and local storage. When they give up the node, they don’t wipe this data.

The next tenant to get the node is malicious. They can read the data left behind by the previous user. Before giving up the node they also flash the NIC’s firmware

This firmware remains when the node is given to a new tenant. It can compromise their possibly sensitive tasks.

Virtualized Clouds vs. Bare Metal Clouds
As a mitigation, we demonstrate a complete chain of measurements rooted in a hardware TPM. A user is then able to attest to the boot time integrity of their node.

This removes the need for much of the trust that traditional clouds require tenants to place in the provider and their fellow clients.

Current Directions
- Using the Hardware isolation layer in conjunction with Bare Metal Imaging and Keylime’s attestation and key management
- Allowing tenants to have a measurable trust in their infrastructure through hardware-based attestations

Future Directions
- Adapt current Keylime architecture to be bootable OS
- Increase the amount of firmware that is measurable
- Open source, minimal firmware
- Even measured firmware can contain bugs/backdoors
- Proprietary firmware stops receiving updates before product end of life
- Open source firmware is auditable & can be updated by community effort

30 Open  32 Closed

1. **S3 resume script in coreboot?** [question] [security]
   #69 opened 3 days ago by osresearch

2. **Build of x230 coreboot with 4.5 fails on first build** [bug] [buildsystem]
   #68 opened 8 days ago by osresearch

3. **initrd/bin has tracked files** [buildsystem] [enhancement] [initrd]
   #63 opened 9 days ago by osresearch

4. **Building the ROM takes two passes?** [bug] [buildsystem]
   #62 opened 9 days ago by osresearch

5. **GPG is too large and overkill**
   #57 opened 21 days ago by zaolin

6. **Chell needs DMAR ACPI table for VT-d** [bug]
   #47 opened on Nov 1 by osresearch

7. **cryptsetup should be in initrd** [enhancement] [initrd]
   #46 opened on Nov 1 by osresearch
https://github.com/osresearch/heads
http://osresearch.net/