Heads: Status update
Heads at 33c3 (2016!!)

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
/ # [ 2.520525] 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
/ #
Plan for today

- **Who am I?**
- What is Heads
- Why Heads
- What’s new?
- What’s next?
- **Insurgo Open Technology** founder and CEO.
- Former Security Analyst/Psychology Bachelor/Security Researcher and Developer.
- Now freedom defender as a open source firmware researcher/developer/integrator.

- Past collaborator to Libreboot, QubesOS contributor and Heads collaborator/reviewer.
- **Currently main Heads maintainer.**

- Started Insurgo Open Technologies in 2017.
- Made the PrivacyBeast X230 certified by QubesOS in July 2019.
- Nlnet grantee for the **Accessible Security** project in April 2019.
- Nlnet grantee once again for Authenticated Heads (**Heads-OpenPGP**) project.

*Insurgo's mission* is to **facilitate accessibility to security and confidentiality** to the masses.
What will we be talking about

Plan for today

- Who am I?
- **What is Heads**
- Why Heads
- What’s new?
- What’s next?
What is Heads?

- Heads as a runtime environment
- Heads as a build system
Heads as runtime environment

Heads is:

coreboot + Linux Kernel in ROM

(plus Security research and tools)
Heads as a runtime environment

- coreboot
  - Hardware initialization
    - Heads as linux’s “bootloader” (kexec)
  - Heads: linux as a coreboot payload
    - linux Kernel
    - linux initrd (initramfs)
      - Contain standard linux tools
      - Enforced security policies (shell scripts)
Heads – coreboot native hardware init

- Openness/ownership/auditability of a Heads’ coreboot supported platform depends on its coreboot’s blobs requirements
- Native hardware initialization depends on arch + chipset
  - On x86: Intel - Ivy bridge/ Sandy bridge : all native (no blobs) (watch for Haswell: native ram init coming)
  - On x86: AMD – Fam15h: all native (no blobs)
    - KGPE-D16 (coreboot 4.11 last official supported version)
    - Dasharo/coreboot (based on coreboot 4.16)
  - Power9 : Talos II: all native (no blobs)
- More info: https://github.com/osresearch/heads/issues/692
Closed source firmware / BIOS Supply chain

**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

**Diagram:**
- CPU Mfg + TianoCore
- IBV
- ODM

**Typical OEMs:**
- Lenovo
- NEC
- Acer
- ASUS
- DELL
- HP
- Microsoft
- Apple

**Notes:**
- OEM’s typically generate < 10% of BIOS Code

**Copyright © 2016**
Heads OSFW

- Coreboot source code (and faith in its platform’s blobs dependencies)
- Linux kernel (version + config)
- Linux tools (modules dependencies version+config) of a board configuration

Let’s note that the linux kernel has:
- extremely limited built in modules/dependencies (linux config for a board)
  - drivers built as modules are loaded on demand
    - USB drivers needed for HOTP USB security dongle
      - USB keyboard not present unless no PS2
      - USB HID not present unless no PS2
  - Drivers compiled as modules are measured prior of being loaded (TPM : PCR5)
    - consequently, a default boot
      - won’t release TPM NV sealed disk unlock key if cannot be unsealed
      - **PCRs will mismatch from sealed and will not unseal**
Heads as a runtime environment

- coreboot
  - Hardware initialization
  - **Heads as linux’s “bootloader”** (kexec)
- Heads: linux as a coreboot payload
  - linux Kernel
  - linux initrd (initramfs)
    - Contain standard linux tools
    - Enforced security policies (shell scripts)
A bootloader is
- Drivers to be able to deal with I/O
- Parser for OS configs
- Pass control to the OS

Bootloader sits between BIOS and OS to be booted

Linux and scripts can do exactly the same without duplicating hardware init nor extending trusted code base...
- Busybox (shell)
- Cryptsetup, lvm, tpm toolstack...
Heads as a Linux bootloader

Select your boot option

Choose the boot option [1-3, a to abort]:

- Debian_GNU/Linux
- Debian_GNU/Linux_with_Linux_5.10.0-20-amd64
- Debian_GNU/Linux_with_Linux_5.10.0-20-amd64_(recovery_mode)

Confirm boot details

Confirm the boot details for Debian GNU/Linux:
kernel /vmlinuz-5.10.0-20-amd64

Make default

Boot one time
Heads as Linux’s “bootloader” (kexec)

In action: gui-init boot policy:
- detach signature against public key + hash validation. TPM NV auth + unsealing LUKS
- cpio constructed with parsed OS’s crypttab + TPM disk unlock key inserted at kexec (cpio)
Heads as linux’s “bootloader” (kexec)

Sda1 is flat ext4 partition with ISO+detach signature (iso.asc) put there. "media-scan" merged recently
Heads as Linux’s “bootloader” (kexec)
Heads as Linux’s “bootloader” (kexec)

Detached signature validation against ROM’s OS’s distro signing public key
Then grub entree selection parsing + kexec
Heads as Linux’s “bootloader” (kexec)
Heads as a runtime environment

- **coreboot**
  - Hardware initialization
  - Heads as Linux’s “bootloader” (kexec)

- **Heads: Linux as a coreboot payload**
  - Linux Kernel
  - Linux initrd (initramfs)
    - Contain standard Linux tools
    - Enforced security policies (shell scripts)

(gui-init through fbwhiptail above)
What is Heads?

- Heads as a runtime environment
- Heads as a build system
Heads as a build system

• Heads is basically a ‘Make’ project
  • Global Makefile [https://github.com/osresearch/heads/blob/master/Makefile](https://github.com/osresearch/heads/blob/master/Makefile)
    • make BOARD=board_name module_name.statement options
    • Boards [https://github.com/osresearch/heads/tree/master/boards](https://github.com/osresearch/heads/tree/master/boards)
    • Existing modules (compilable software): [https://github.com/osresearch/heads/tree/master/modules](https://github.com/osresearch/heads/tree/master/modules)
    • Patches to be applied after module verification + extraction: [https://github.com/osresearch/heads/tree/master/patches](https://github.com/osresearch/heads/tree/master/patches)
  • Produces
    – Artifacts : creboot rom(s) images stitching the following (but produced independently)
    – BzImage (compiled kernel + in-kernel modules)
    – Initrd.cpio.xz
      • tools.cpio (compiled modules stripped binaries)
      • modules.cpio (compiled as modules kernel drivers to be loaded on demand)
      • heads.cpio (scripts and config files generated at build time linked to board config and [https://github.com/osresearch/heads/tree/master/initrd](https://github.com/osresearch/heads/tree/master/initrd) content
    – Hashes.txt file containing individual packed files, cpios, initrd.cpio.xz and coreboot roms
Plan for today

- Who am I?
- What is Heads
- **Why Heads**
- What’s new?
- What’s next?
Why Heads

- Outside of coreboot’s “minimalist” mandate:
  - Linux as its bootloader
    - Linux kernel enumerates devices
    - Linux supports of peripherals/buses
    - Linux support of filesystems
    - Linux kernel setups of IOMMU
    - Linux permits Kexec’ing into final OS
  - Standard linux tools in initrd (**add your own module if missing**)
    - TPM toolstacks (sealing/unsealing of secrets)
  - Bash scripts defines the boot policies launched by init
    - **gui-init**: validates/creates detached signatures, hashes, LUKS headers
  - **Maximized roms** takes back all ME neutered freed space, unlocks firmware descriptor, permitting internal full firmware upgrades!

---

BIOS Software Supply Chain Breakdown

- Independent BIOS Vendors are 3rd-party UEFI developers that sell value-added UEFI, toolkits, and custom development services
- CPU Mfg + TianoCore
  - IBV
  - ODM
  - OEM
- Lenovo
  - NEC
  - ASUS
  - DELL
  - HP
  - Microsoft
  - Typical
- OEM's typically generate <10% of BIOS Code

Copyright © 2016
Why Heads

- Extensive TPM usage
  - Coreboot measured boot mode extends TPM PCR2 (no DRTM as of now on supported platforms. Might change with T440p and other newer platforms)
  - Heads extends others
    - PCR4: Boot mode (0 during /init, then recovery or normal-boot)
    - PCR5: Heads Linux kernel modules
    - PCR6: Drive LUKS headers
    - PCR7: Heads user-specific files stored in CBFS (config.user, GPG keyring, etc).
- Why important?
  - Sealing secrets in TPM NV memory
    - TOTP/HOTP sealed secret (based on PCRs 0-4) : Firmware integrity attestation
    - TPM Disk Unlock Key (based on PCRs 0-7) sealed secret with custom passphrase: releases key to OS without you having to type passphrase to that OS
gui-init policy + TPM released Disk unlock key
What will we be talking about

Plan for today

- Who am I?
- What is Heads
- Why Heads
- **What’s new?**
- What’s next?
What’s new

- Maximized boards vs Legacy boards, or how to dodge blob redistribution legal limitations
- Whiptail/FBWhiptail: one graphical interface (GUI) to rule them all
- OEM Factory reset/Re-Ownership wizard upstreamed
- QEMU/KVM board configurations with swtpm and USB Security dongle support to ease development/testing
OEM/Legacy BIOS

Intel Firmware Descriptor (IFD): Locked. Describes GBE ME BIOS sizes here!
IFD, Intel ME and BIOS regions : cannot be modified/unlocked but externally
x230-flash/x230-hotp-verification (legacy example)

```
export CONFIG_FLASHROM_OPTIONS="--force --noverify-all -p internal --ifd --image bios"

# This board has two SPI flash chips, an 8 MB that holds the IFD,
# the ME image and part of the coreboot image, and a 4 MB one that
# has the rest of the coreboot and the reset vector.
#
# Only flashing to thebios region is safe to do. The easiest is to
# flash internally when the IFD is unlocked for writing, and x230-flash
# is installed first.
```
Maximized boards

- Contains unlocked + modified IFD

```
user@heads-tests:~:/heads/blobs/xx30$ ~/heads/build/x86/coreboot-4.13/util/ifdtool/ifdtool -f layout.txt ifd.bin
File ifd.bin is 4096 bytes
Wrote layout to layout.txt
user@heads-tests:~:/heads/blobs/xx30$ cat layout.txt
00000000:00000000 ff
00010000:00bfffff fd
0001b000:00bfffff bios
00003000:0001afff me
00001000:00002fff gbe
```

- \texttt{00bfffff - 0001b000 = BE4FFF}
  - Can be maximized even more since should match coreboot’s \texttt{CBFS\_SIZE}!

- Contains Neutered ME

- Contains generated GBE (per specs)
Maximized boards

- Blobs redistribution legal issue dodging
  - Github contains download+extraction code
  - CircleCI downloads and stitches ROMs
  - No blobs are hosted
Maximized boards

```bash
#!/bin/bash

function printusage {
  echo "Usage: $0 -m <m_cleaners(optional)>"
}

$LIBDIR="/" $dir="/" $(dirname "$($BASH_SOURCE[0])") | &

if [ "$#" = "0" ]; then printusage; fi
while getopt "n:m:" opt do
  case opt in
    m)
      if [ -x "$OPTARG" ]; then
        MECLEAN="$OPTARG"
        fi;
        esac
      esac
    esac
  esac
if [ -z "$MECLEAN" ]; then
  MECLEAN="command -v $LIBDIR/.../build/coreboot */util/mc_cleaner/mc_cleaner.py 2> /dev/null"
  if [ -z "$MECLEAN" ]; then
    echo "m_cleaner.py required but not found or specified with -m. Aborting.";
    exit 1;
  fi
  fi
  echo "### (creating temp dir)"
  extractdir=$((mktemp -d CD "$extractdir"))
  echo "### Downloading http://download.lenovo.com/pcsbbs/mobiles/glrg24aw.exe..."
  wget https://download.lenovo.com/pcsbbs/mobiles/glrg24aw.exe " echo "Failed" sha256sum verification on downloaded binary..." & exit 1;
  echo "### Verifying expected hash of glrg24aw.exe"
  echo ""JME_EXE SHA256SUM"" | sha256sum -c - echo "Failed sha256sum verification on extracted binary..." & exit 1;
  echo "### Extracting glrg24aw.exe..."
  innoextract ./glrg24aw.exe " echo "Failed calling innoextract. Tool installed on host?" & exit 1;
  echo "### Verifying expected hash of app/MB_SM Production.bin"
  echo "JME_PRODRODUCTION SHA256SUM" | sha256sum -c - echo "Failed sha256sum verification on extracted binary..." & exit 1;
  echo "### Applying m_cleaner to neutral-deactivate-maximize reduction of ME on bioscopy, outputting minimized ME under $LIBDIR/mc_cleaner..."
  MECLEAN -r $0 "$LIBDIR/mc_cleaner" app/MB_SM Production.bin
  if [ -e "$MECLEAN" ]; then
    echo "### Verifying expected hash of mc_cleaner.bin"
    echo ""FINAL_BIN SHA256SUM"" | sha256sum -c - echo "Failed sha256sum verification on final binary..." & exit 1;
  fi
  echo "### Cleaning up..."
  cd - & rm -r "$extractdir""
```
Maximized boards

CircleCI downloads/clean/put blobs in place
Maximized boards

CircleCI stitches the ROM together
Maximized boards

CircleCI stitches the ROM together
Maximized boards

CircleCI keeps artifacts for each built commit (30 days)
What’s new

- Maximized boards vs Legacy boards, or how to dodge blob redistribution legal limitations
- Whiptail/FBWhiptail: one graphical interface (GUI) to rule them all
- OEM Factory reset/Re-Ownership wizard upstreamed
- QEMU/KVM board configurations with swtpm and USB Security dongle support to ease development/testing
Whiptail: console (server/BMC)
What’s new

• Maximized boards vs Legacy boards, or how to dodge blob redistribution legal limitations
• Whiptail/FBWhiptail: one graphical interface (GUI) to rule them all
• **OEM Factory reset/Re-Ownership wizard upstreamed**
• QEMU/KVM board configurations with swtpm and USB Security dongle support to ease development/testing
This operation will automatically:

* ERASE the TPM and own it with a password
* ERASE any keys or passwords on the GPG smart card, reset it to a factory state, generate new keys
  and optionally set custom PIN(s)
* Add the new GPG key to the firmware and reflash it
* Sign all of the files in /boot with the new GPG key

It requires that you already have an OS installed on a dedicated /boot partition. Do you wish to continue?
Would you like to change the current LUKS Disk Recovery Key passphrase? (Highly recommended if you didn’t install the Operating System yourself, so that past provisioned passphrase would not permit to access content. Note that without re-encrypting disk, a backedup header could be restored to access encrypted content with old passphrase) (y/n): y

Would you like to re-encrypt LUKS encrypted container and generate new Disk Recovery key? (Highly recommended if you didn’t install the operating system yourself; this would prevent any LUKS backedup header to be restored to access encrypted data) (y/n): n

The following security components will be provisioned with default or chosen PINs/passwords:

- LUKS Disk Recovery Key passphrase
- TPM Ownership password
- GPU Admin PIN
- GPU User PIN

Would you like to set a single custom password that will be provisioned to previously stated security components? (y/n): n
Would you like to set distinct PINs/passwords to be provisioned to previously stated security components? (y/n): n

Enter desired replacement for current Disk Recovery Key passphrase (at least 8 characters long):

Insurgo Open Technologies

Enter current Disk Recovery Key passphrase (Provisioned at OS installation or by OEM):

Insurgo Open Technologies

Would you like to set custom user information for the GnuPG key? (y/n): n
Would you like to export your public key to an USB drive? (y/n): n

Checking for USB Security Dongle...

Detecting and setting boot device...

Boot device set to /dev/sda1

Reencrypting /dev/sda1 LUKS encrypted drive content with current Recovery Disk Key passphrase...
TPM Owner Password: 12345678
GPG Admin PIN: 12345678
GPG User PIN: 123456
OEM Factory Reset / Re-Ownership has completed successfully

After rebooting, you will need to generate new TOTP/HOTP secrets when prompted in order to complete the setup process.

Press Enter to reboot.
What’s new

- Maximized boards vs Legacy boards, or how to dodge blob redistribution legal limitations
- Whiptail/FBWhiptail: one graphical interface (GUI) to rule them all
- OEM Factory reset/Re-Ownership wizard upstreamed
- QEMU/KVM board configurations with swtpm and USB Security dongle support to ease development/testing
QEMU/KVM boards

- Permit easy testing and debugging (debian-12 recommended)
  - After having followed instructions:
    
    https://osresearch.net/Emulating-Heads/#comprehensive-test

    ```
    make BOARD=qemu-coreboot-fbwhiptail-tpm2
    PUBKEY_ASC=~/QubesIncoming/Insurgo/Insurgo_2023_pub.asc
    USB_TOKEN=NitrokeyStorage ROOT_DISK_IMG=~/QubesIncoming/heads-tests/root.qcow2
    QEMU_MEMORY_SIZE=1G inject_gpg
    ```

    ```
    make BOARD=qemu-coreboot-fbwhiptail-tpm2
    PUBKEY_ASC=~/QubesIncoming/Insurgo/Insurgo_2023_pub.asc
    USB_TOKEN=NitrokeyStorage ROOT_DISK_IMG=~/QubesIncoming/heads-tests/root.qcow2
    QEMU_MEMORY_SIZE=1G run
    ```
QEMU/KVM boards

Could not access KVM kernel module: No such file or directory.
qemu-system-x86_64: failed to initialize kvm: No such file or directory
What’s Next

- TPM2 support on QEMU/KVM and SWTPM (skeleton: https://github.com/osresearch/heads/pull/1292)
- A better build system to guarantee reproducible builds based on NixOS if everything goes well (PoCs started)
- Clean room, in ram GPG key generation with backup/restore/USB thumb drive emergency usage capabilities (No more USB Security dongle strong requirement to use Heads while still highly recommended).
  - Authenticated Heads recovery shell, USB boot and more! (under design!)
- Finally: flash write protection options!
  - Platform chipset locking (only Heads can flash firmware) (PoC: https://github.com/osresearch/heads/pull/326#issuecomment-1019684512)
  - SPI Write protection, permitting to write protect coreboot’s bootblock region (requires external flashing when coreboot version bumps happen under Heads. For the most paranoid only!)
    - 3mdeb dasharo/flashrom merge soon: https://github.com/osresearch/heads/pull/1251
- International keyboard support (PoC started: https://github.com/osresearch/heads/issues/555)
- On demand MAC randomization inside of Heads, overwriting GBE region inside of firmware. Persistence across firmware upgrades. (PoC started: https://github.com/osresearch/heads/pull/1195)
- Even more space for Maximized roms! (additional ~0.3-0.6mb): (PoC: https://github.com/osresearch/heads/pull/1298)
References:

Differences between linuxboot, Heads NERF
Heads conference (Hudson, 33c3, 2016)
Linuxboot conference (Hudson, 34c3, 2017)
Heads: a call for collaboration (Laurion, FOSDEM, 2020)
Coreboot measured boot, SRTM mode (coreboot doc)
Heads current measured boot scheme (Heads doc)

Project homes

Heads searchable documentation
Heads project's home (GitHub code/features/issues)
Heads documentation's home (GitHub documentation/issues)
Heads community direct link
Questions/Comments?

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

CPU Mfg + TianoCore

AMD

intel

* tianocore

IBV

American Megatrends

insyde

phoenix

ODM

COMPAQ

FOXCONN

PEGATRON

WISTRON

flex

inventec

OEM's typically generate < 10% of BIOS Code

OEM

Lenovo

NEC

ACER

ASUS

dell

HP

Microsoft

www.Intel.com

Copyright © 2016

Lenovo