LinuxBoot

Let Linux do it

Daniel Maslowski

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Agenda

- Motivation
- LinuxBoot Concept
- UEFI Integration
- Implementations
- Future Work

Motivation

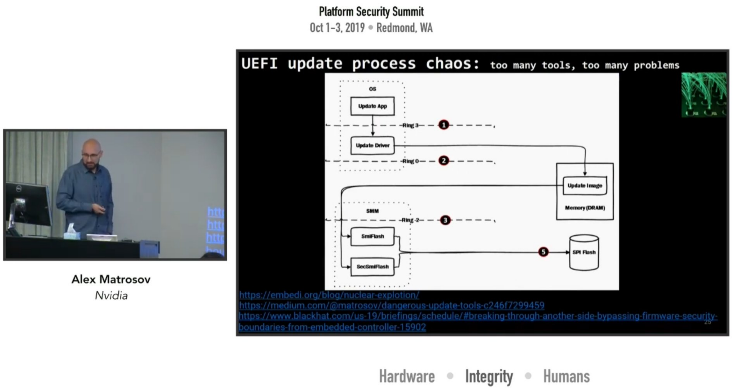
Firmware now vs back then

- 1999: birth of coreboot as LinuxBIOS
  - open source x86 firmware \o/
- 2004: Tiano initial release by Intel
  - now EDK I/II, maintained by UEFI community
- 2014: Intel Haswell release
  - requires proprietary MRC (Memory Reference Code) binary
  - later on: FSP (Firmware Support Package)
- 2014: AMD Generic Encapsulated Software Architecture (AGESA) lockdown
  - binary only since then
  - was initially open sourced for coreboot in early 2011
  - an open laptop would have been nice
- 2019: UDF (UEFI Dumpster Fire™)
  - criticized by many people
  - for many years

neglected: Intel ME, AMD PSP, ARM and other SoCs

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

State of security



Platform Security Summit
Oct 1–3, 2019 • Redmond, WA

Alex Matrosov
Nvidia

UEFI update process chaos: too many tools, too many problems

https://embedi.org/blog/nuclear-explotion/
https://medium.com/@matrosov/dangerous-update-tools-c246f7299459
https://www.blackhat.com/us-19/briefings/schedule/#breaking-through-another-side-bypassing-firmware-security-boundaries-from-embedded-controller-15902

Hardware • Integrity • Humans

- update processes are often insecure
- vendors and firmware projects take no responsibility
- great summary by Alex Matrosov

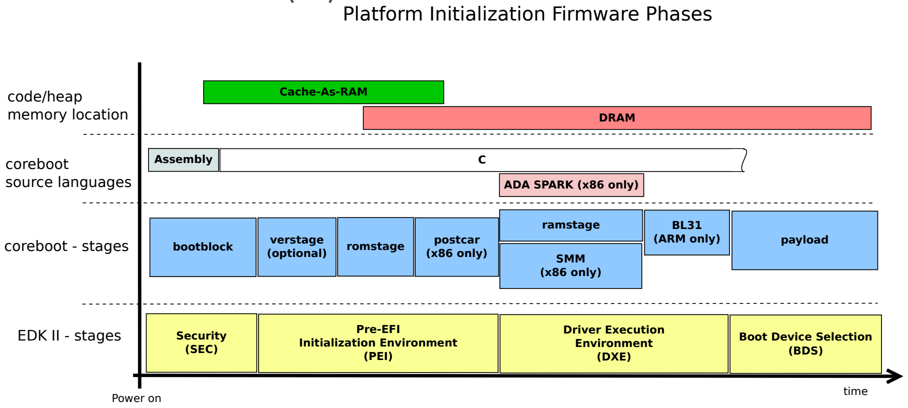
ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Right to repair bill

- vendors still propose security by obscurity
  - although known to be pointless against sophisticated attackers
- repair technicians suffer from proprietiary information
  - consumers and researchers alike

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Platform Initialization (PI)



basic platform initialization: CPU, chipset, RAM (PEI / romstage)

- has to be rerun similarly for S3 resume

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

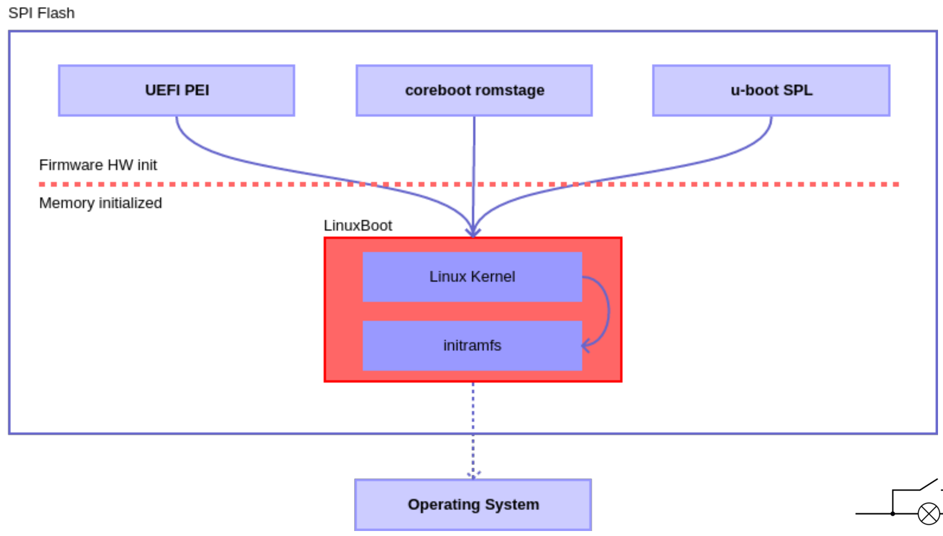
LinuxBoot Concept

LinuxBoot

- Linux kernel + initramfs in SPI flash
- can run on top of
  - coreboot: as payload
  - U-Boot
  - vendor UEFI firmware: remove DXEs, build Linux with EFI support

=> approach rather than implementation

Integrations



ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Constraints

- only few megabytes of space (8 to 16 common)
- build minimum kernel
  - disk drivers
  - filesystems
  - possibly networking
- build basic initramfs
  - core utilities like `ls`, `cat`, etc
  - bootloader(s) - need to boot an OS ;)

=> very similar to OpenWrt, except for bootloader instead of routing tools

UEFI Integration

UEFI binary format

PE32 / PE32+ format, without symbol tables

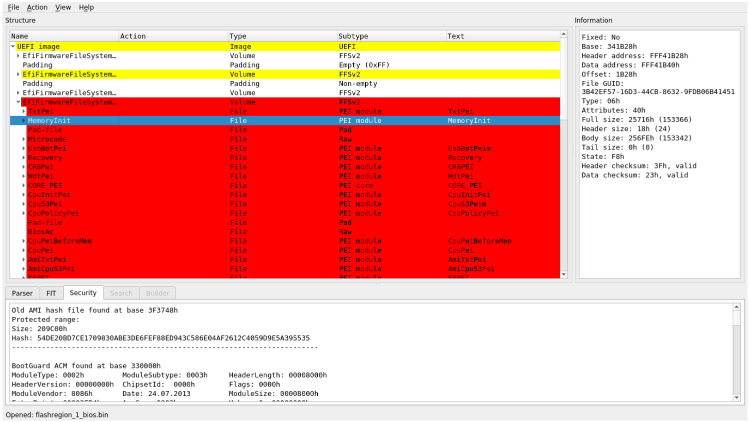
Three types:

- applications
  - OS loaders
  - utilities
- boot service drivers
  - disk drivers
  - network drivers
- runtime drivers
  - may remain loaded while OS is running

$=>$ replace applications and boot service drivers with LinuxBoot

Tools

- Fiano
  - utk with DXE cleaner
- UEFITool



Name	Action	Type	Subtype	Text
UEFI image		Image	UEFI	
EfiFirmwareFileSystem…		Volume	FFSv2	
Padding		Padding	Empty (0xFF)	
EfiFirmwareFileSystem…		Volume	FFSv2	
Padding		Padding	Non-empty	
EfiFirmwareFileSystem…		Volume	FFSv2	
EfiFirmwareFileSystem…		Volume	FFSv2	
TxtPei		File	PEI module	TxtPei
MemoryInit		File	PEI module	MemoryInit
Pad-file		File	Pad	
Microcode		File	Raw	
UsbBotPei		File	PEI module	UsbBotPeim
Recovery		File	PEI module	Recovery
CRBPei		File	PEI module	CRBPEI
WdtPei		File	PEI module	WdtPei
CORE_PEI		File	PEI core	CORE_PEI
CpuInitPei		File	PEI module	CpuInitPei
CpuS3Pei		File	PEI module	CpuS3Peim
CpuPolicyPei		File	PEI module	CpuPolicyPei
Pad-file		File	Pad	
BiosAc		File	Raw	
CpuPeiBeforeMem		File	PEI module	CpuPeiBeforeMem
CpuPei		File	PEI module	CpuPei
AmiTxtPei		File	PEI module	AmiTxtPei
AmiCpuS3Pei		File	PEI module	AmiCpuS3Pei

```
Fixed: No
Base: 341B28h
Header address: FFF41B28h
Data address: FFF41B40h
Offset: 1B28h
File GUID:
3B42EF57-16D3-44CB-8632-9FDB06B41451
Type: 06h
Attributes: 40h
Full size: 25716h (153366)
Header size: 18h (24)
Body size: 256FEh (153342)
Tail size: 0h (0)
State: F8h
Header checksum: 3Fh, valid
Data checksum: 23h, valid
```

```
Old AMI hash file found at base 3F3748h
Protected range:
Size: 209C00h
Hash: 54DE20BD7CE1709830ABE3DE6FEF88ED943C586E04AF2612C4059D9E5A395535
------------------------------------------------------------------------

BootGuard ACM found at base 330000h
ModuleType: 0002h         ModuleSubtype: 0003h     HeaderLength: 00008000h
HeaderVersion: 00000000h  ChipsetId:  0000h        Flags: 0000h
ModuleVendor: 8086h       Date: 24.07.2013         ModuleSize: 00008000h
```

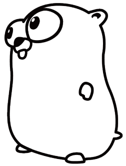
Opened: flashregion_1_bios.bin

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Implementations

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

u-root

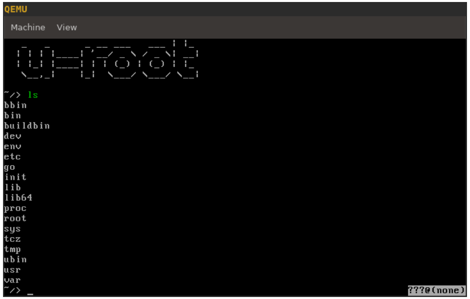


- initramfs tool written in Go
- utilities like busybox (`ls`, `cat`, ...)
- offers bootloaders (SystemBoot)

Try out u-root in QEMU

```
go get github.com/u-root/u-root
# build an initramfs
GOOS=linux \
  ~/go/bin/u-root -build=bb -o /tmp/initramfs.linux_amd64.cpio
# get a kernel
MIRROR="http://mirror.rackspace.com" REL="2019.10.01" \
  wget "$MIRROR/archlinux/iso/$REL/arch/boot/x86_64/vmlinuz"
# run it :)
qemu-system-x86_64 -kernel vmlinuz \
  -initrd /tmp/initramfs.linux_amd64.cpio
```

u-root demo



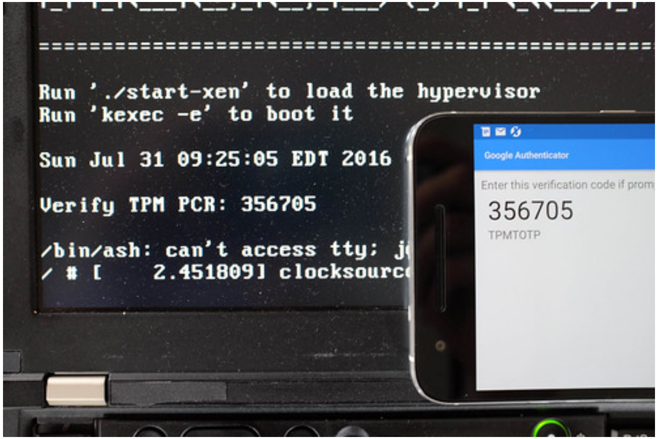
```
                              | |
  _   _      _ __ ___   ___ | |_
 | | | |____| '__/ _ \ / _ \| __|
 | |_| |____| | | (_) | (_) | |_
  \__,_|    |_|  \___/ \___/ \__|

~/> ls
bbin
bin
buildbin
dev
env
etc
go
init
lib
lib64
proc
root
sys
tcz
tmp
ubin
usr
var
~/> _                                                  ???@(none)
```

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Heads

- authenticated / measured boot



```
======================================================

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; j
/ # [    2.451809] clocksourc
```

u-bmc

- u-root for BMCs
- alternative to OpenBMC

Project	OpenBMC	u-bmc
Languages	C++, Python	Go
Tooling	Yocto, OpenEmbedded	u-root
Kernel	OpenBMC Linux fork	OpenBMC Linux fork
Init	systemd	
IPC	D-Bus	
RPC	IPMI, REST	gRPC
Metrics		OpenMetrics

Future Work

ACH … ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

CHIPSEC blacklist in MFT

- UEFI Forum openly discussed security measures for firmware development and answered questions from participants

  Q: Can consumers audit the firmware? If so, how?

  A: There are a variety of tools that can allow a consumer to inspect firmware images. CHIPSEC and UEFI Tool are two tools that can analyze a firmware image and allow a consumer to inspect its contents. CHIPSEC has a blacklist of UEFI modules which include a tool that will check a ROM image for blacklisted modules.

- Mimoja released the MimojaFirmwareToolkit (MFT) https://firmware.doctor
  - integrate CHIPSEC blacklist in analysis?
  - contributions are welcome ;)

ACME for firmware update PKI

Since firmware updates are such an issue:

- we had a very similar issue on the web with secure communication
- leverage the ACME protocol (Let’s Encrypt) also for firmware?
- create issues on TianoCore GitHub org for discussion

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Questions?

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX

Thanks! :)

ACH ... ICH LASS DAS JETZT SO!
LABORTAGE MMXIX