Try Harder 2 Hardening the COREs

Shawn C

#whois

* Shawn C[a.k.a "citypw"]
* Day job at TYA infotech
    * Open source security consulting
* GNU/Linux security engineer
* Free/libre SW/FW/HW enthusaists
* Member of EFF/FSF/FSFE/SFC
* Patient Zer0 at Hardenedlinux community(https://hardenedlinux.github.io/)

#cat /proc/agenda

* History: Ring 3
* Attacking the core
* Under the water
* Invincible devil
* Hope or delusion?

#Ring 3

* MAC/DAC/Sandboxing(seccomp)
* Baseline checks(STIG-4-Debian)
* Firewall/IDS/SOC
* etc

#We had a history with Ring 3

* Once upon a time, the stack was so "pure"...

* Stack/Heap layout
* Stack grows down (x86, MIPS)
* ESP points to the current top of the stack
* EBP points to the current function frame

```
High  +---------------------+
      |    Stack            |
      |(grow downward)|
      |---------------------|
      |                     |
      |---------------------|
      |  (grow upward)   |
      |     Heap            |
 Low  +---------------------+
```

#War at Ring 3

Stack overflow

Heap overflow

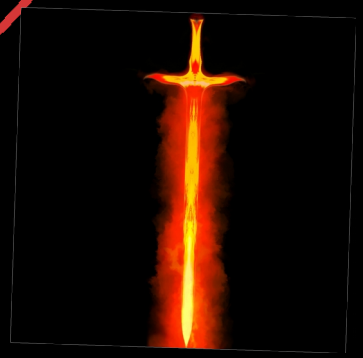
Integer overflow

Race condition

String formt

ret2libc/ROP

Overwrite ret/GOT/dtors



PaX ASLR/Vanilla
ASLR + PIE

Stack cookie

PAGEEXEC/SEGEXEC/NX bit

FORTIFY SOURCE

RELRO

PAX_MPROTECT

Forward-edged CFI

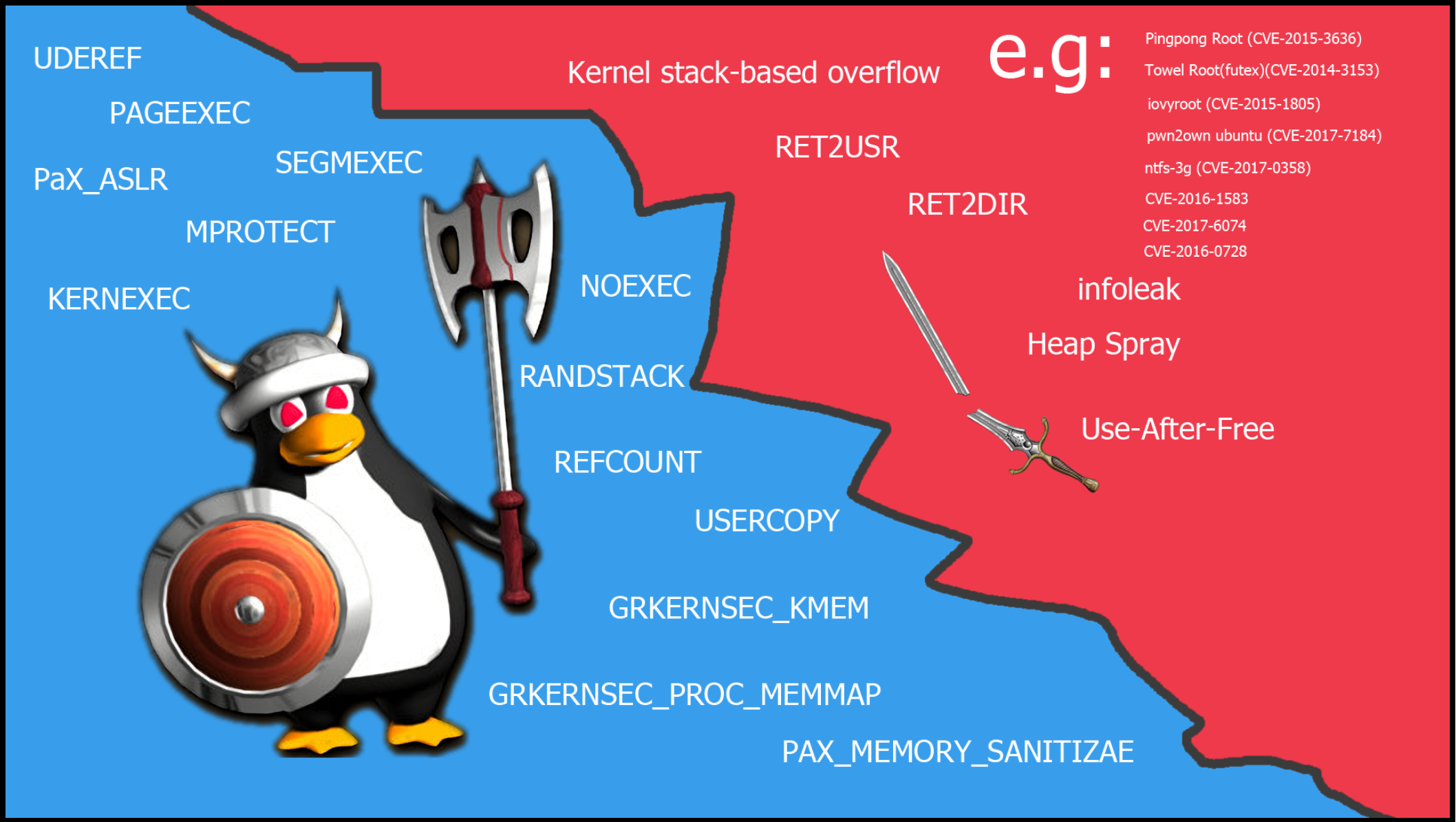
#End of story?

We got everything we need! Ok, this is it. Thanks for coming. Bye, cruel world!

#Why attack the core?

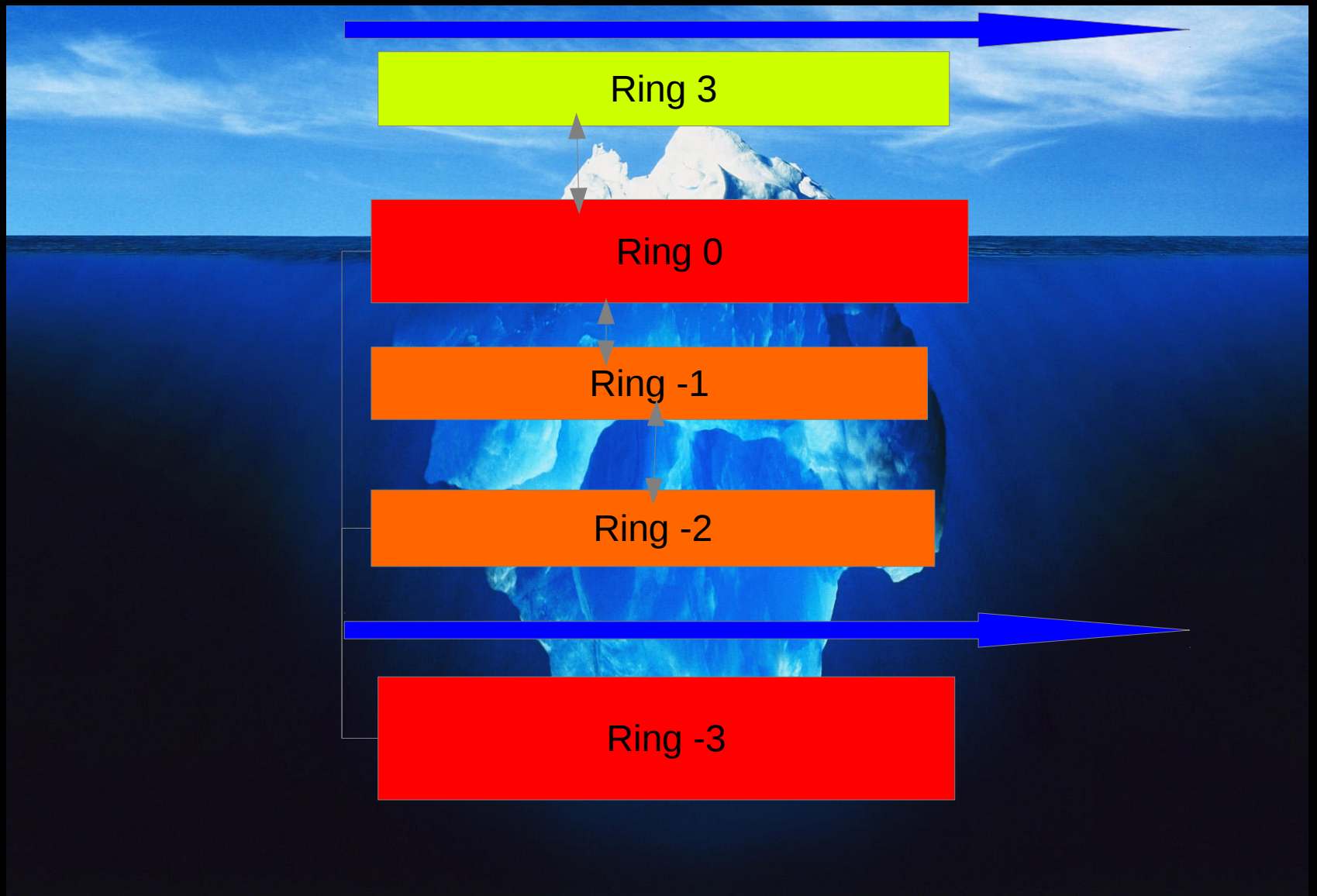
* Linux kernel sucks in 2000s
  * One null-ptr deref can rule them all
* Still a cargo-cult security in 2017?
* Harder to exploit userspace programs

#Defend the core



#Wait, which CORE exactly?

* RING 0 is **not** the CORE anymore



#Under the water

* VM guest escape
* Did all device drivers follow the best practice(let's say IOMMU)?
* Old/new good/bad attacks on SMM
* Persistent attack chain
  * -1: Hijacking the kernel(perf impact?)
  * -2: Memory tricks

See the reference...

#Demo

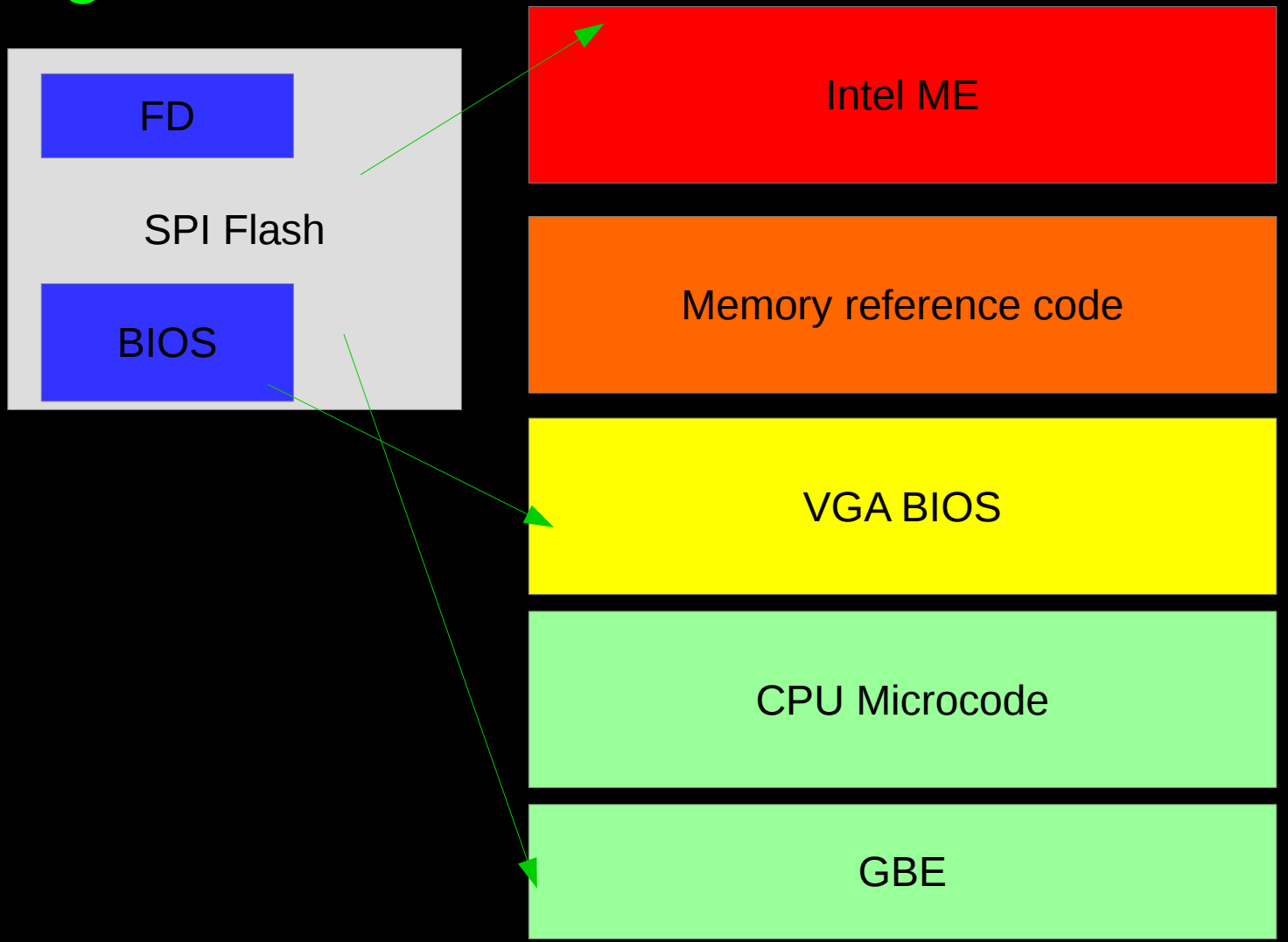
What could possiblely go wrong with your “trusted” watcher? Don’t freaking out...

#Invincible devil

* Intel ME
* A lot of ME "apps" based on it
* Changed a lot since v11
* Can’t be disabled

#From a libre FW's perspective

* Not good



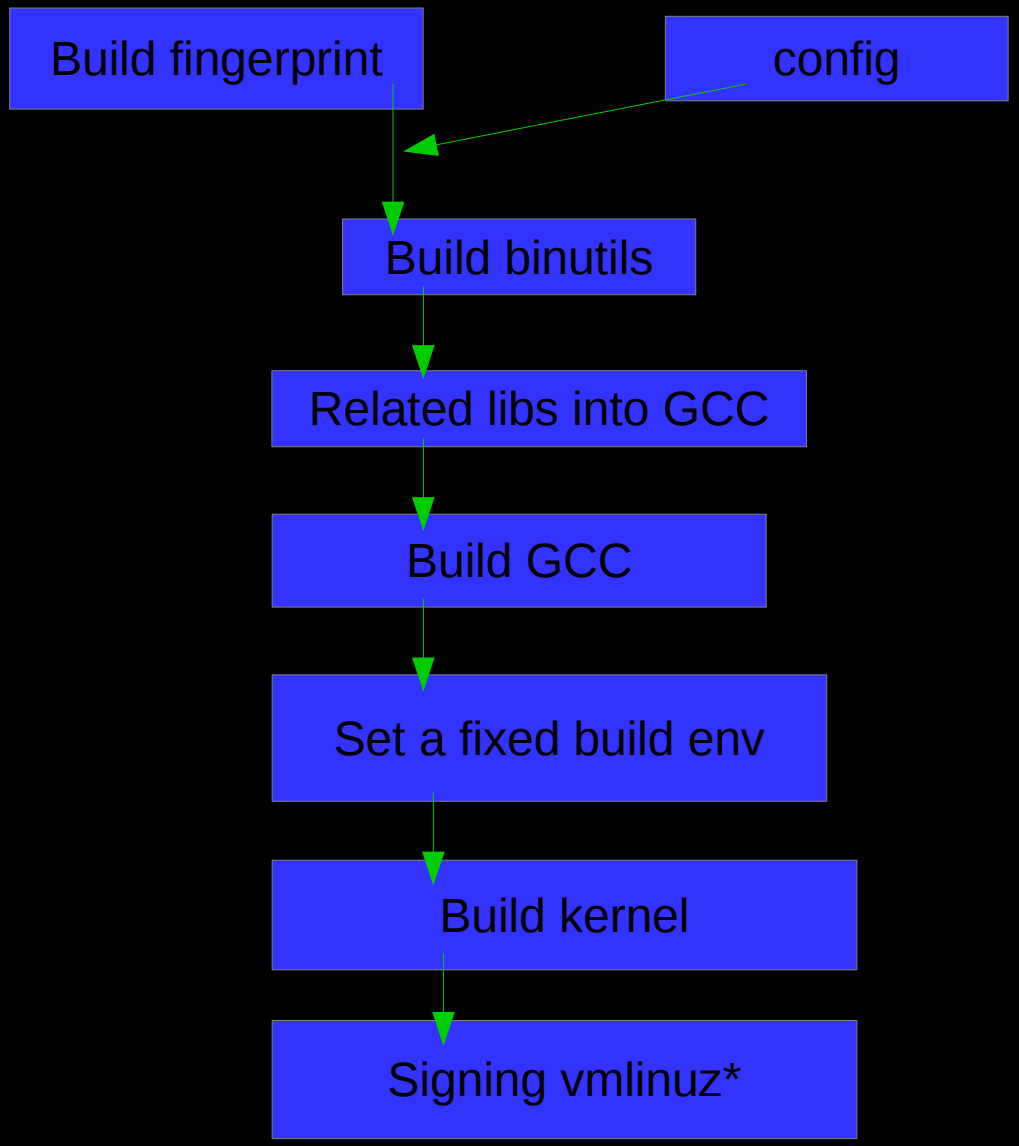
#Hope or delusion?

* Some trade-off must be made
* Reproducible builds for PaX/Grsecurity
* Hardenedboot: Measured boot & verified boot
* Restricted ME via minimizing its fuctions

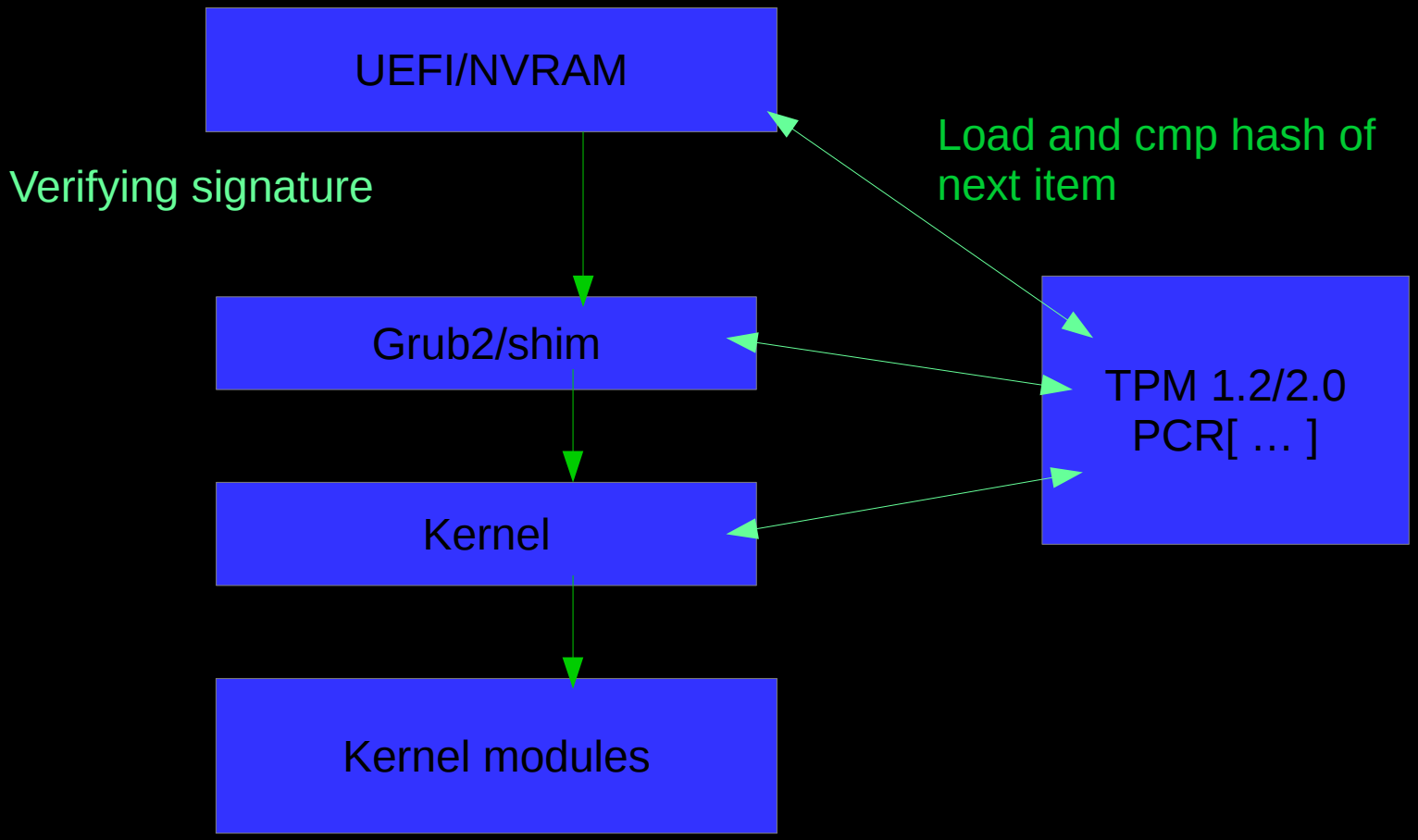
#Why PaX/Grsecurity matters

* Kill bug classes
* Kill exploit vectors
* Assumption: Let data center takes physical security into account
    * Kernel is still the path to the under water
    * Hardened guards kills the attack surfaces

#Reproducible builds for PaX/Grsecurity



#Hardened Boot



#Free/libre solution?

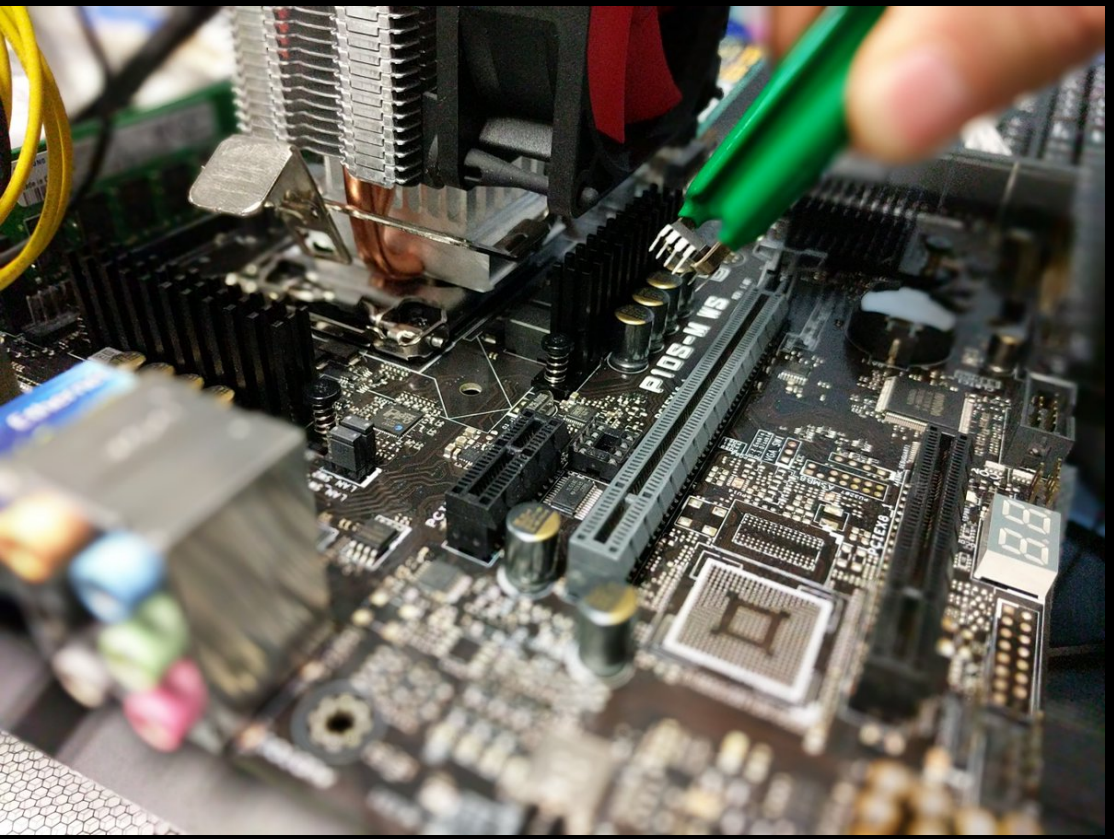
* Coreboot/FSP
* Libreboot
* Situational hardening
  * Old good machines for critical assets
  * Newer machines for generic purposes
* Reference model for custom firmware
* Long-term plan: coreboot for RISC-V?

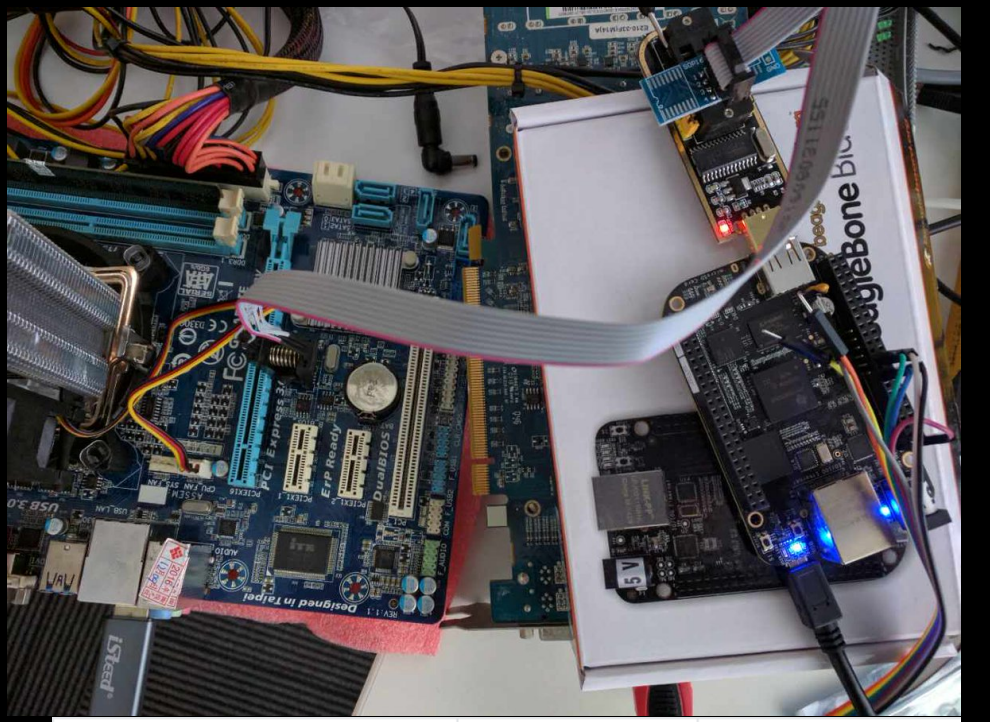
#Open == auditable

* open != secure, a-random-coreboot-machine:

```
[CHIPSEC] ***************************  SUMMARY  ***************************
[CHIPSEC] Time elapsed          0.014
[CHIPSEC] Modules total         17
[CHIPSEC] Modules failed to run 0:
[CHIPSEC] Modules passed        4:
[+] PASSED: chipsec.modules.common.smm
[+] PASSED: chipsec.modules.common.ia32cfg
[+] PASSED: chipsec.modules.common.smrr
[+] PASSED: chipsec.modules.common.spi_fdopss
[CHIPSEC] Modules failed        8:
[-] FAILED: chipsec.modules.common.bios_wp
[-] FAILED: chipsec.modules.common.bios_ts
[-] FAILED: chipsec.modules.common.spi_lock
[-] FAILED: chipsec.modules.common.spi_desc
[-] FAILED: chipsec.modules.common.bios_smi
[-] FAILED: chipsec.modules.memconfig
[-] FAILED: chipsec.modules.smm_dma
[-] FAILED: chipsec.modules.remap
[CHIPSEC] Modules with warnings 2:
[!] WARNING: chipsec.modules.common.bios_kbrd_buffer
[!] WARNING: chipsec.modules.common.rtclock
[CHIPSEC] Modules skipped 3:
[*] SKIPPED: chipsec.modules.common.uefi.s3bootscript
[*] SKIPPED: chipsec.modules.common.uefi.access_uefispec
[*] SKIPPED: chipsec.modules.common.secureboot.variables
[CHIPSEC] ***************************************************************
```

#Neutralizing ME





Mainboard	CPU	Tested BIOS
GA-B75M-D3H	SandyBridge	OEM/Coreboot
GA-B75M-D3V	IvyBridge	OEM/Coreboot
Lenovo T420	IvyBridge	OEM/Coreboot
Lenovo X220/X220i	SandyBridge	OEM/Coreboot
Lenovo X230	IvyBridge	OEM/Coreboot
Chromebook XE550C22	IvyBridge	OEM/Coreboot
ASUS P10S-M WS	Skylake	OEM

#Extra notes;-)

* Neutralized ME affected remote attestation via SGX?
* Think harder about your situational hardening solution before ask OEM write your pk
* You don’t need Intel Bootguard if the data center could take care of physical security
* Big clouds(AWS/Google/Facebook/BAT3H/etc) should do the situational hardening for their core infrastructure. Otherwise it might become someone else’s computer;-)
* Masters of Pwn killed by PaX/Grsecurity

#Are we done?

* Or just another starting point?
* Know your enemy(from Ring 3/0/-1/-2/-3)
* Risk assessment for important production/assets
    * Known vulnerability/exploit vectors


HardenedLinux's roadmap
Crypto Engineering
Firmware
Compiler
Security operations
Situational Hardening
KERNEL
Free/Libre & Open Source Software's eco-system:
+ GPL-compliance
+ Legislation
+ Education
Adversary
The West Of Middle Earth At The End Of The Third Age
FORODWAITH
Angmar
Mount Gunduband
Ered Mithrin
Iron Hills
ARNOR
HITHAEGLIR
ERED LUIN
Forlindon
Ettenmoors
Weather Hills
Rhudaur
Erebor (The Lonely Mountain)
Erareth on the Long Lake
ARTHEDAIN
Old Forest
The Shire
Bree
Rivendell
Mirkwood
East Bight
RHÛN
Sea of Rhun
Blue Mountains
Harlindon
ERIADOR
Eregion
MINHIRIATH
Dunland
Lorien
Fangorn
Isengard
The Brown Lands
ENEDWAITH
Helms
Ered Lithui (Ash Mountians)
N. Ithilien
Plateau of Gorgoroth
MORDOR
NURN
Anfalas
Lebennin
S. Ithilien

Reference

* PaX/Grsecurity:

https://grsecurity.net/

* Coreboot:

http://coreboot.org/

* Linux kernel mitigation checklist:

https://hardenedlinux.github.io/system-security/2016/12/13/kernel_mitigation_checklist.html

* Ring 0: Linux kernel vulnerablity & exploitation & silent fixes

https://github.com/hardenedlinux/grsecurity-101-tutorials/blob/master/kernel_vuln_exp.md

* Virtualization security:

https://github.com/hardenedlinux/grsecurity-101-tutorials/blob/master/virt_security.md

Reference

* Firmware security:

https://github.com/hardenedlinux/firmware-anatomy/blob/master/hack_ME/firmware_security.md

* Reproducible builds for PaX/Grsecurity

https://github.com/hardenedlinux/grsecurity-reproducible-build

* Hardened Boot:

https://github.com/hardenedlinux/Debian-GNU-Linux-Profiles/tree/master/docs/hardened_boot

* Neutralized ME with coreboot stuff

https://github.com/hardenedlinux/hardenedlinux_profiles/tree/master/coreboot

* Intel ME's info:

https://github.com/hardenedlinux/firmware-anatomy/blob/master/hack_ME/me_info.md

QA

Thanks