

信誉危机

广受认可的硬件和软件遭遇信任危机

Seunghun Han


国家安全研究所高级安全研究员

29 May 2019

我是谁?



- NSR (韩国国家安全研究所) 高级安全研究员
- Black Hat Asia 2019影响力成员
- KIMCHICON 审查委员会成员
- 作为演讲嘉宾出席以下会议：
 - USENIX Security 2018
 - Black Hat Asia 2017 - 2019
 - HITBSecConf 2016 - 2017
 - BeVX and KIMCHICON 2018

- “64-bit multi-core OS principles and structure, Vol. 1 and Vol. 2)的作者
- a.k.a kkamagui  @kkamagui1

本演讲的目标

- 介绍一个关于信誉的刻板印象
 - 信誉并不代表值得信赖!
 - 不幸的是，由于信誉，我们很容易相信某些东西!
- 列举信誉厂商令人失望的案例
 - BIOS / UEFI固件和可信平台模块 (TPM) 由值得信赖的公司制作!
 - 但是，我发现了两个漏洞，CVE-2017-16837和CVE-2018-6622，可以破坏TPM
- 提出了对策以及我们所需要做的
 - 不要基于信誉而轻信，眼见为实，自己动手检查!

以前的工作


black hat[®]
ASIA 2018

MARCH 20-23, 2018
MARINA BAY SANDS / SINGAPORE

I Don't Want to Sleep Tonight: Subverting Intel TXT with S3 Sleep

Seunghun Han, Jun-Hyeok Park
(hanseunghun || parkparkqw)

Wook Shin, Junghwan Kang, HyungChun Kim
(wshin || ultract || khche)

#BHASIA / @BLACKHATASIA


black hat[®]
ASIA 2019

MARCH 26-29, 2019
MARINA BAY SANDS / SINGAPORE

Finally, I Can Sleep Tonight: Catching Sleep Mode Vulnerabilities of the TPM with Napper

Seunghun Han, Jun-Hyeok Park
(hanseunghun || parkparkqw)@nsr.re.kr

Wook Shin, Junghwan Kang, HyungChun Kim
(wshin || ultract || khche)@nsr.re.kr

 **usenix**[®]
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping

Seunghun Han, Wook Shin, Jun-Hyeok Park, and HyungChun Kim,
National Security Research Institute

<https://www.usenix.org/conference/usenixsecurity18/presentation/han>

Proceedings of the
USENIX Security Symposium.

Baltimore, MD, USA

18-46-1

Open access to the Proceedings of the
18th USENIX Security Symposium
is sponsored by USENIX.

信誉

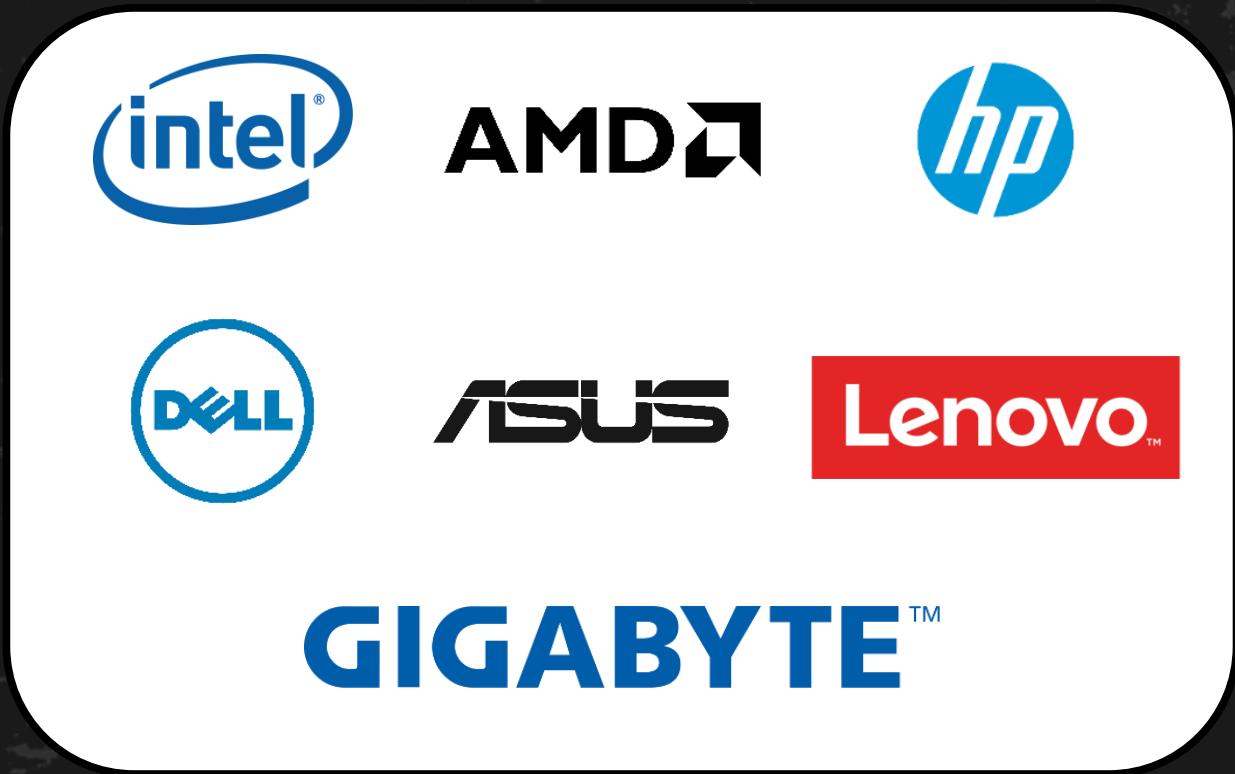
基于

信任!

我们只**相信**
值得信赖
的公司出产的
产品

信誉良好的公司
(高价)

其他公司
(低价)



您的

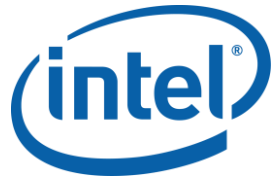
仅供演示!

信誉良好的公司

其他公司

I KNOW WHAT YOU DID

(值)



GI



DOWN

IDS

FOR THE PRESENTS!

又供演示!

Trusted Platform Module Library

Part 1: Architecture

Family "2.0"

Level 00 Revision 01.38

September 29, 2016

衡量信任的根源

核心 RTM

值得信赖的要素

9.2.5 Trust Authority

When the **RTM** begins to execute the **CRTM**, the entity that may vouch for the correctness of the **TBB** is the entity that created the TBB. For typical systems, this is the platform manufacturer. In other words, the manufacturer is the authority on what constitutes a valid TBB, and its reputation is what allows someone to trust a given TBB.

TCG

TCG Published

Copyright © TCG 2006-2016

信誉良好的产品

真的

值得信任吗？

信譽

≠

可信!

每个人都有一个计划，
直到他们脸上挨了一拳。

- 迈克 泰森

每个人都有一个计划，
直到他们脸上挨了一拳。

- 迈克 泰森

每个研究者 都有一个计划，
直到遇到他们的经理。

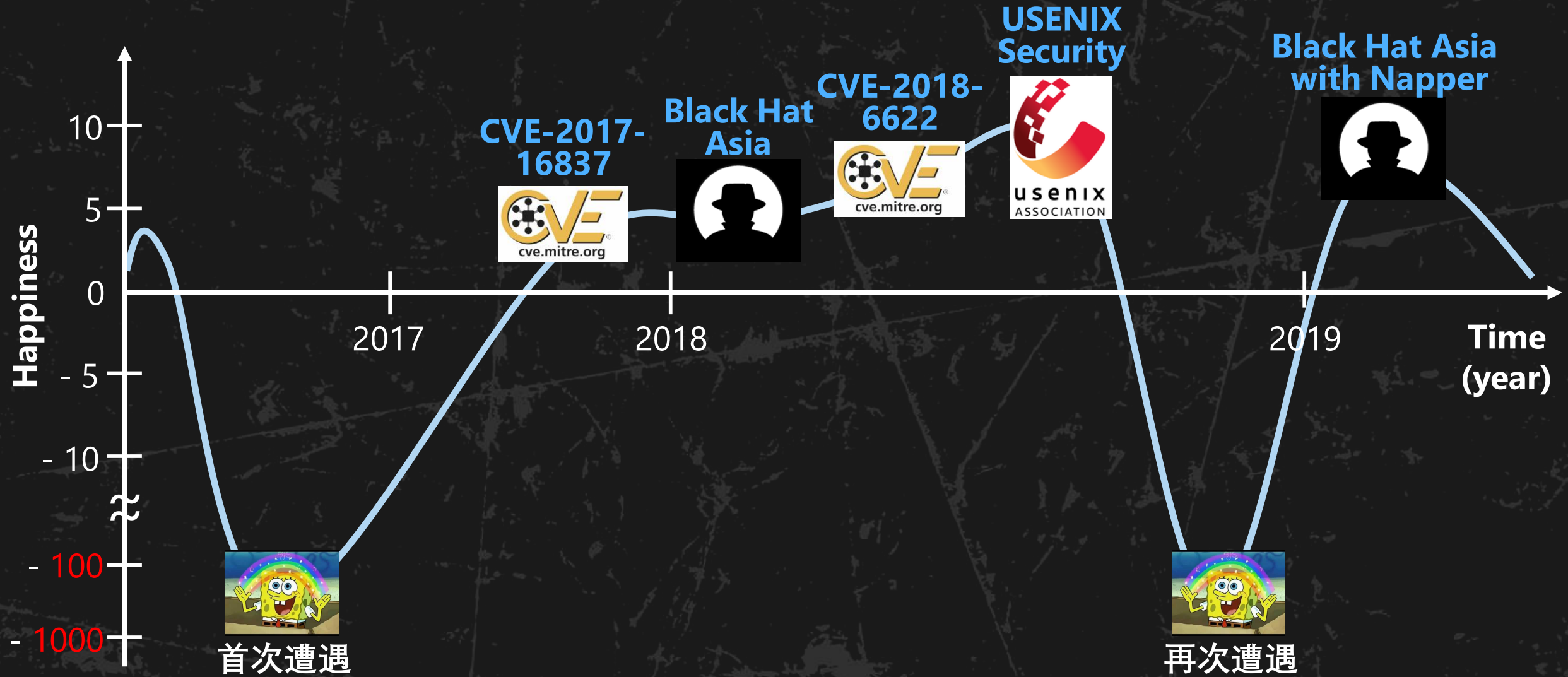
- 佚名



每个研究者 都有一个计划，
直到遇到他们的**经理**。

- 佚名

时间轴



内容 - 背景



可信计算组织 (TCG)

- 定义全球行业规范和标准

- 英特尔, AMD, IBM, 惠普, 戴尔, 联想, 微软, 思科, 瞻博网络和英飞凌等知名公司均为其成员

- 支持硬件信任根

- 可信平台模块 (TPM) 是核心技术

- TCG技术已应用于统一可扩展固件接口 (UEFI)



TCG的可信计算基 (TCB)

- 是主机平台上的软件和硬件集合
- 管理和执行系统的安全策略
- 能够防止自己受到入侵
 - 可信平台模块 (TPM) 有助于确保TCB正确实例化并值得
信赖

可信平台模块 (TPM) (1)

- 是防篡改设备
- 拥有自己的处理器，RAM，ROM和非易失性RAM
 - 它有自己的与系统分开的独立状态
- 提供加密和累积测量功能
 - 测量值累积到平台配置寄存器 (PCR # 0 ~ # 23)



可信平台模块 (TPM) (2)

- 用于通过调查存储在PCR中的值来确定系统的可信度
 - 可以使用本地验证或远程证明
- 用于根据特定的PCR值限制对秘密数据的访问
 - “密封”操作利用TPM的PCR加密秘密数据
 - 只有当PCR值与特定值匹配时，“Unseal”操作才能解密密封数据

测量信任的根源 (RTM)

- 向TPM发送与完整性相关的信息 (测量)
 - TPM将测量结果累积到具有PCR中先前存储的值的PCR

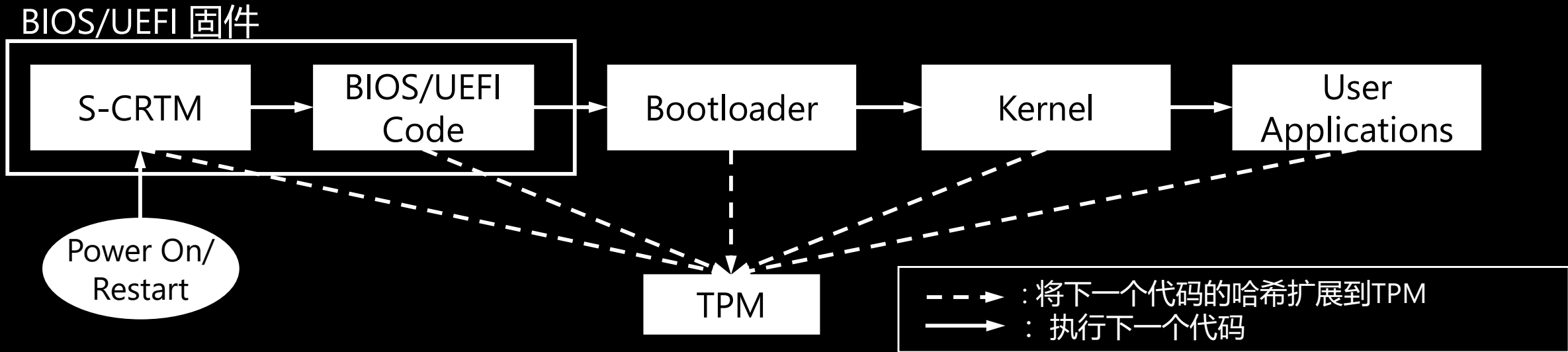
Extend: $PCR_{new} = \text{Hash}(PCR_{old} || \text{Measurement}_{new})$

- CPU是否由Core RTM (CRTM) 控制
 - 当建立新的信任链时, CRTM是第一组指令

静态和动态RTM (SRTM和DRTM)

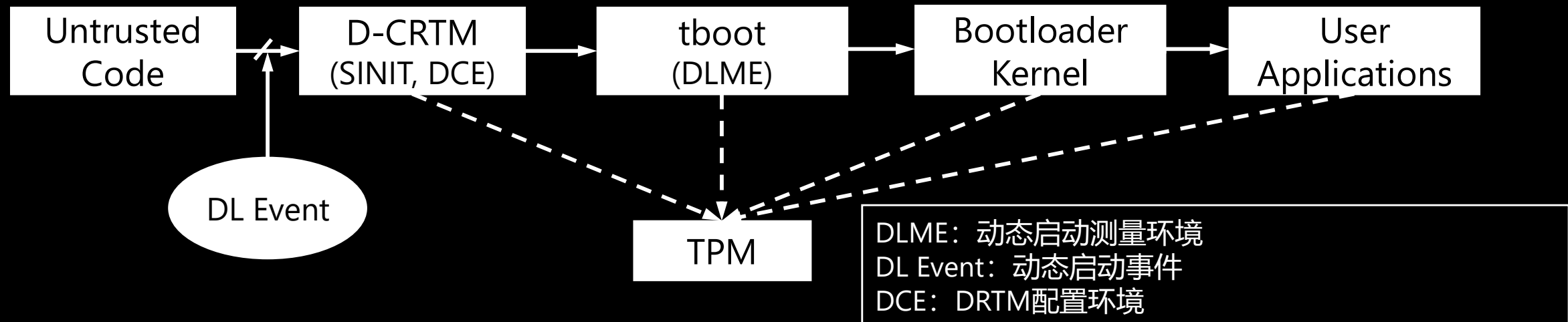
- 当主机平台在**POWER-ON**或**RESTART**启动时，**SRTM**由静态**CRTM (S-CRTM)** 启动
- **DRTM**由动态**CRTM (D-CRTM)** 在运行时启动，**无需平台重置**
- 在将控制传递给它们**前**，它们将组件的测量值（散列）扩展到**PCR**

静态测量信任根 (SRTM)



动态测量信任根 (DRTM)

(英特尔可信执行技术)



Bank/Algorithm: TPM ALG SHA1(0x0004)

PCR_00:	3d	ca	ea	25	dc	86	55	4d	94	b9	4a	a5	bc	8f	73	5a	49	21	2a	f8
PCR_01:	b2	a8	3b	0e	bf	2f	83	74	29	9a	5b	2b	df	c3	1e	a9	55	ad	72	36
PCR_02:	b2	a8	3b	0e	bf	2f	83	74	29	9a	5b	2b	df	c3	1e	a9	55	ad	72	36
PCR_03:	b2	a8	3b	0e	bf	2f	83	74	29	9a	5b	2b	df	c3	1e	a9	55	ad	72	36
PCR_04:	df	5a	d0	48	a8	b1	09	2c	79	b8	69	e6	7d	f6	d7	45	a3	a7	7e	5f
PCR_05:	cd	ca	c6	1f	16	b2	22	b8	00	79	62	23	8a	f4	b1	73	5c	28	c5	d8
PCR_06:	b2	a8	3b	0e	bf	2f	83	74	29	9a	5b	2b	df	c3	1e	a9	55	ad	72	36
PCR_07:	40	37	33	6f	a7	bc	0e	ab	e3	77	8f	cf	ff	5f	cd	0e	e6	ad	cd	e3
PCR_08:	4e	d8	ea	d3	c3	04	1f	26	13	63	3f	f8	11	15	c9	ce	69	c7	a8	ad
PCR_09:	a6	2d	c8	08	06	d3	b0	ce	45	90	31	ec	0b	3c	5a	4a	ec	00	79	9a
PCR_10:	8e	06	97	8b	9c	73	3f	fa	b2	df	9d	c9	d9	12	c3	1a	b0	6a	b6	d0
PCR_11:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_12:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_13:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_14:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_15:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_16:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_17:	fc	8a	d7	96	cf	4d	02	18	0f	15	6c	1c	a3	45	1b	bd	30	8a	09	71
PCR_18:	7f	a7	c1	56	a5	ad	09	da	8c	0f	0e	5e	f7	25	da	22	41	fc	6c	e0
PCR_19:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_20:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_21:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_22:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
PCR_23:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

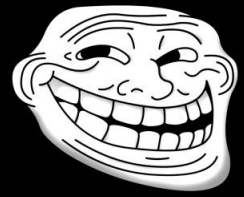
SRTM

DRTM

PCR 保护

- 即使攻击者获得root权限，也不得通过不允许的操作重置它们！
 - 只有主机复位时，才能复位静态PCR (PCR # 0~ # 15)
 - 仅当主机初始化DRTM时，才能重置动态PCR (PCR # 17~ # 22)
- 如果攻击者重置PCR，他们可以通过重放哈希值来重现特定的PCR值
 - 他们可以窃取秘密并欺骗本地和远程验证

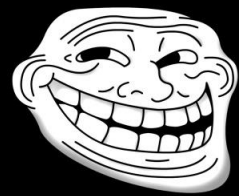
我们相信所有这些机制
因为信誉!



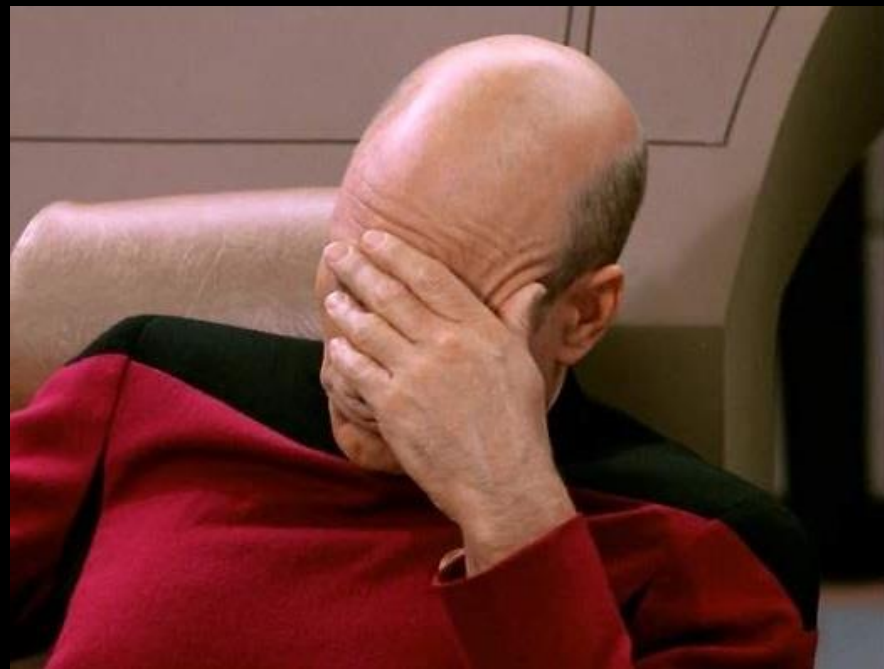
幸运的是，他们起作用了！

我们

机制



幸运的

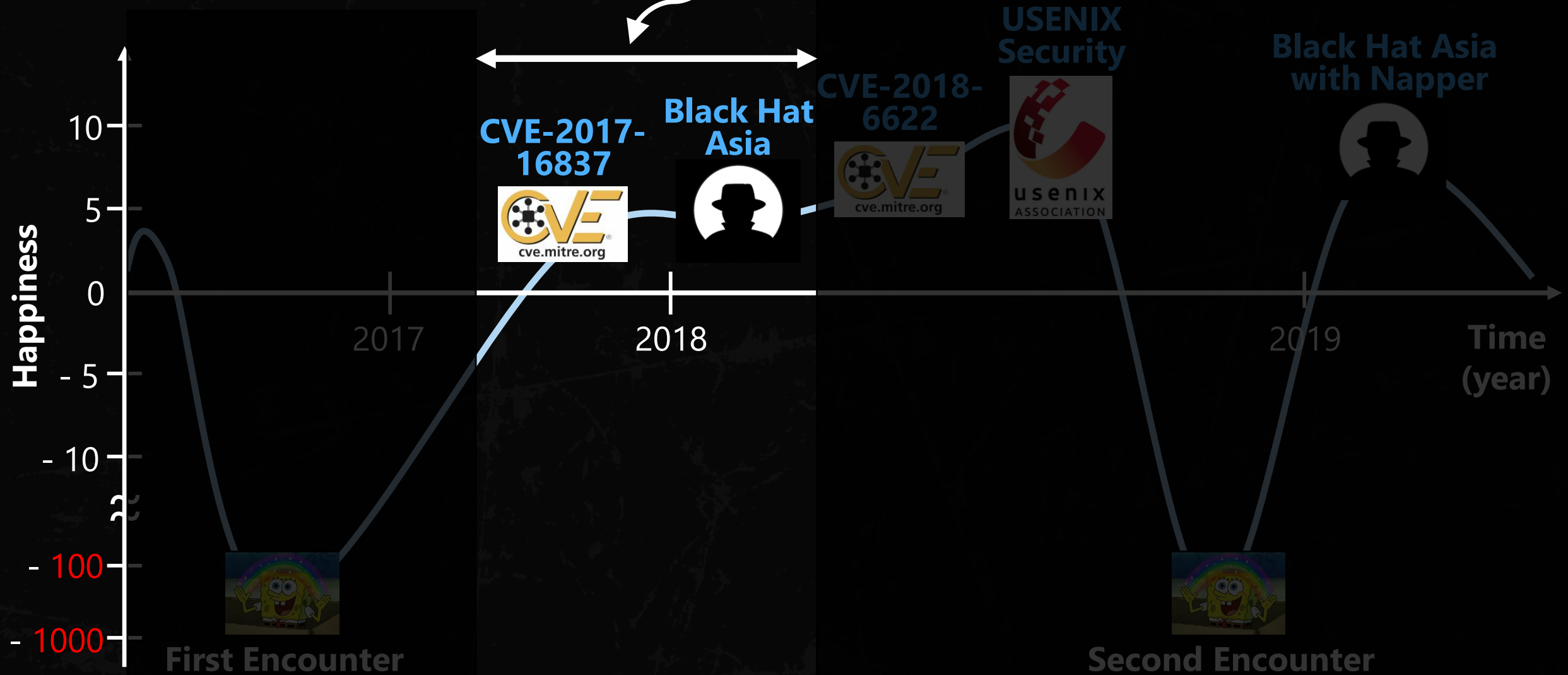


你背叛我!

作用了!

直到我发现了漏洞!

内容 - CVE-2017-16837



英特尔可信执行环境 (TXT)

- 是TCG规范的**DRTM**技术

- 英特尔只使用自己的术语

- ex) DCE =安全初始化认证代码模块 (SINIT ACM)

- DLME =测量启动环境 (MLE)

- 有一个特殊命令 (**SEENTER**和**SEXIT**) 进入可信任状态并从中退出

- SEENTER检查SINIT ACM是否有有效签名

- 英特尔在网站上发布SINIT ACM

可信启动 (tBoot)

- 是英特尔TXT的**参考实现**

- 这是一个开源项目 (<https://sourceforge.net/projects/tboot/>)

- 它包括许多Linux发行版, 如RedHat, SUSE和Ubuntu的

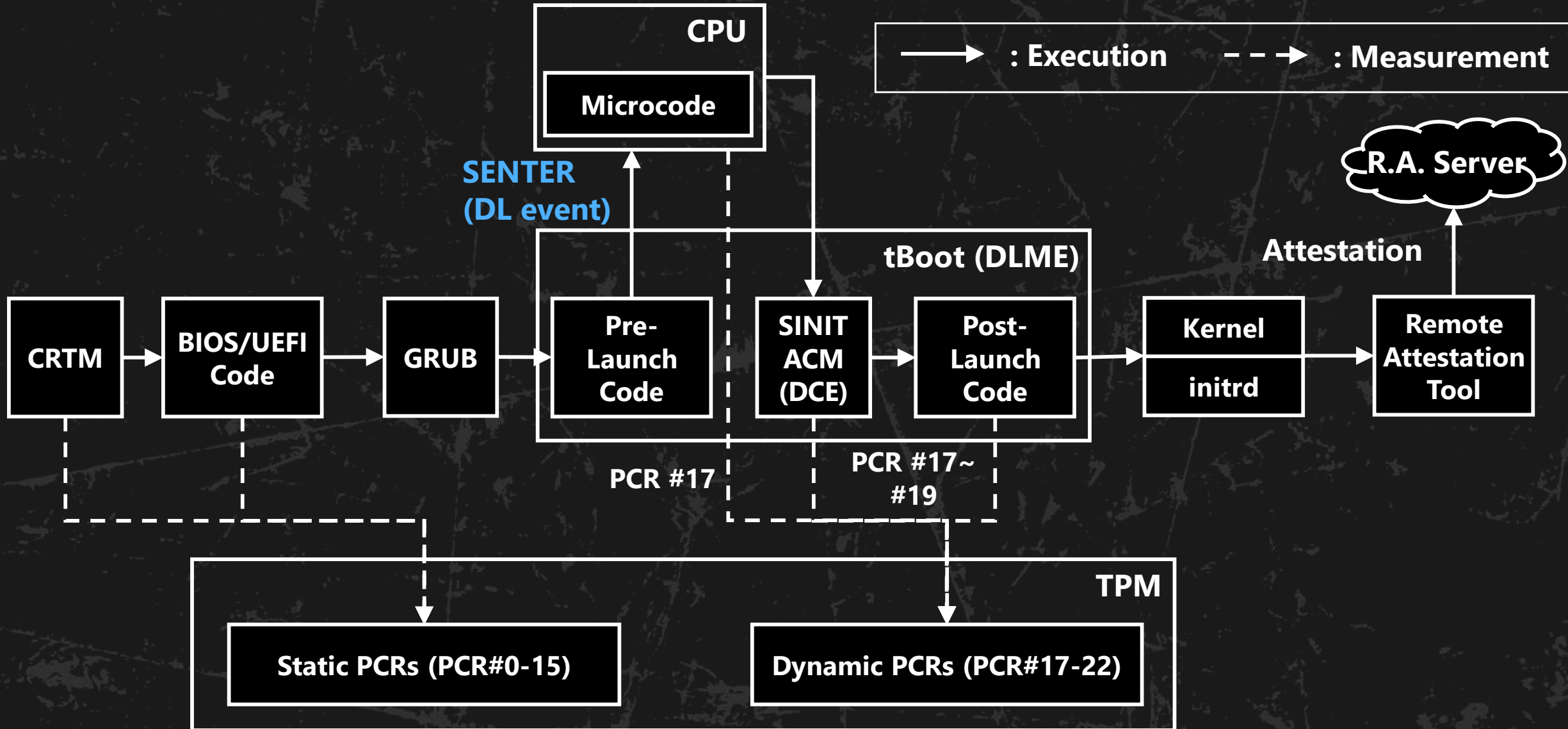
- 可以验证操作系统和虚拟机监视器 (VMM)

- 它测量OS组件并将散列存储到TPM

- TPM的PCR中的测量结果可以由远程认证服务器 (例如Intel Open CIT) 验证

- 它通常用于服务器环境

tBoot的启动过程



启动过程很完美!

(也许)

休眠过程怎么样??

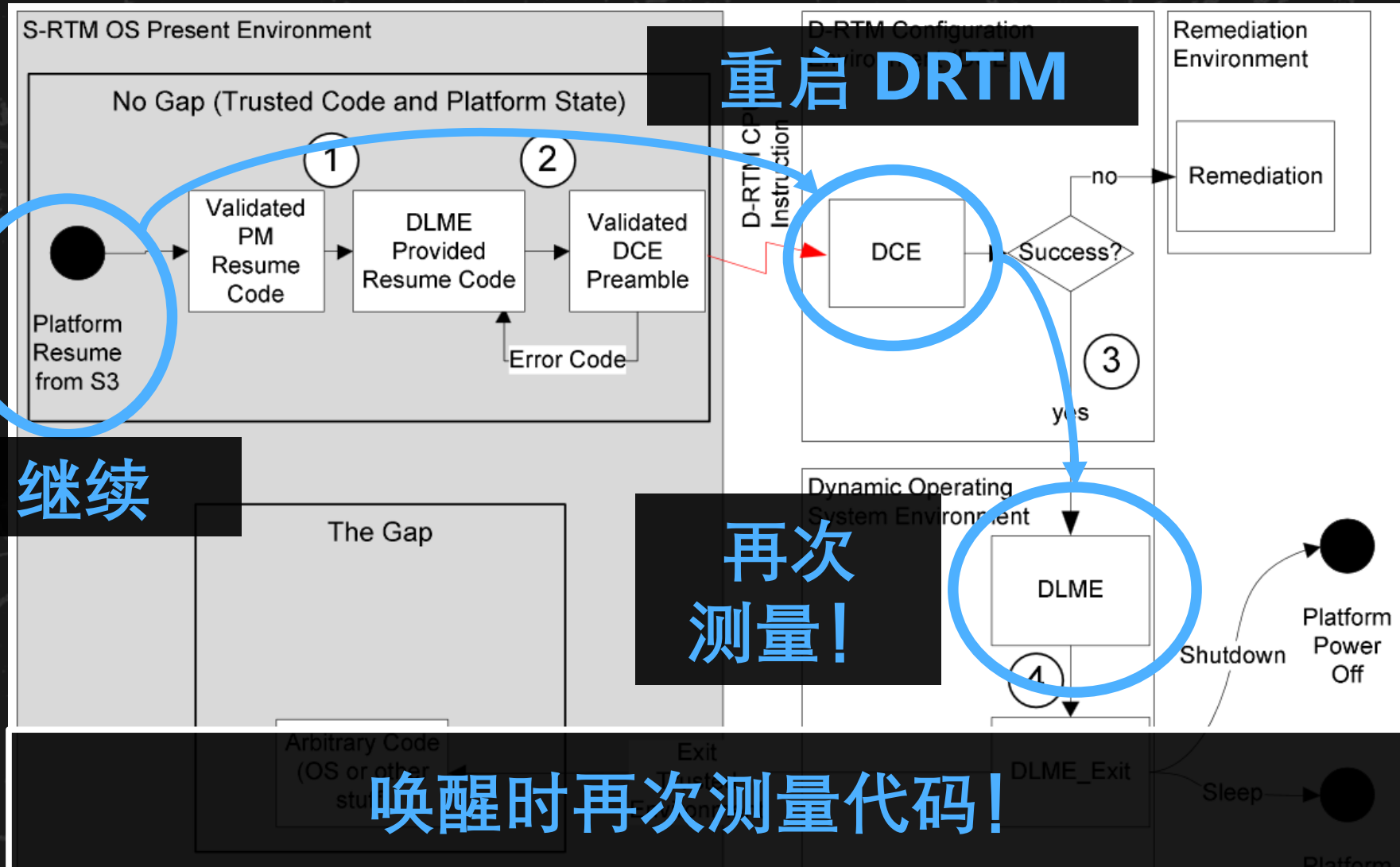
高级配置和电源接口 (ACPI) 和休眠状态

- 关闭电源之后

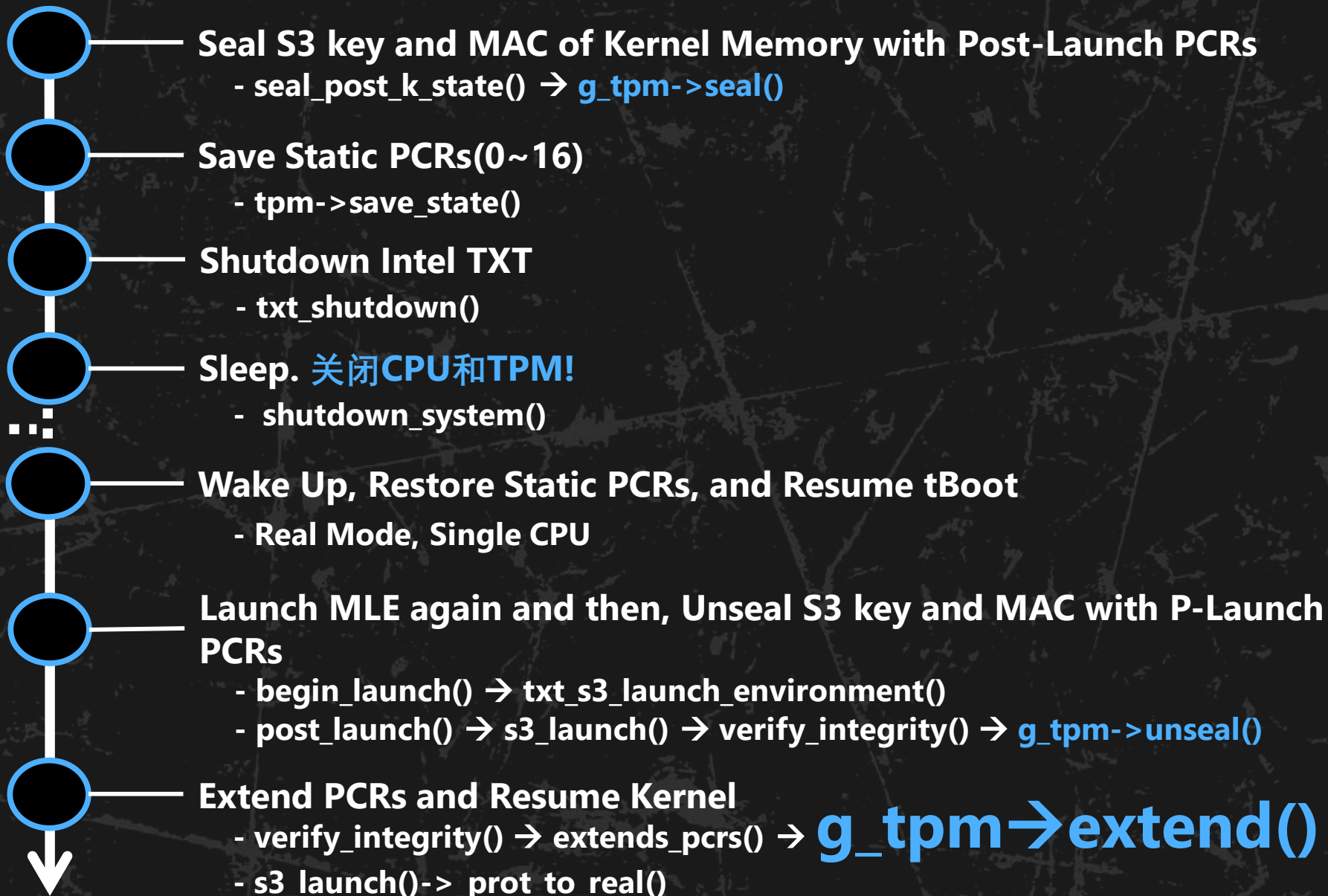
- S0: 正常, 没有上下文丢失
- S1: 待机, CPU缓存丢失
- S2: 待机, CPU处于断电状态
- S3: 暂停, CPU和设备已关闭电源
- S4: 休眠, CPU, 设备和RAM均已关闭
- S5: 软关闭, 所有部件均已关闭电源

TPM也已关机!

唤醒DRTM的过程

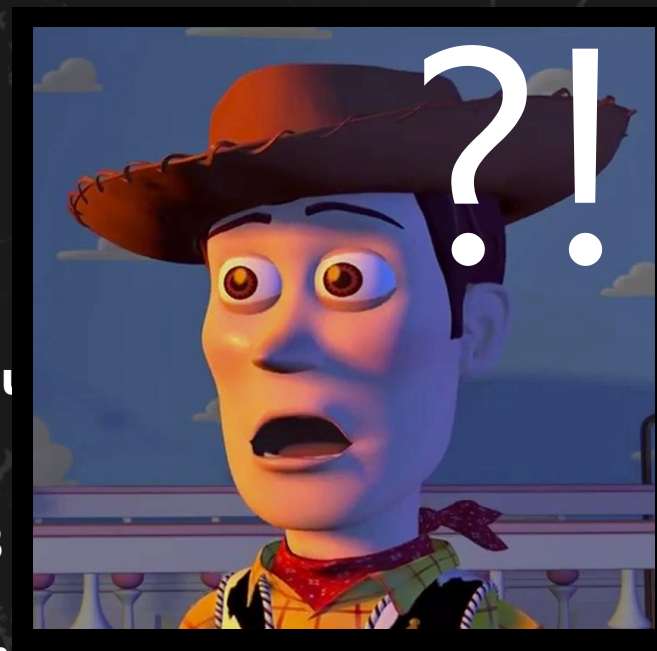


使用tBoot进行休眠的过程

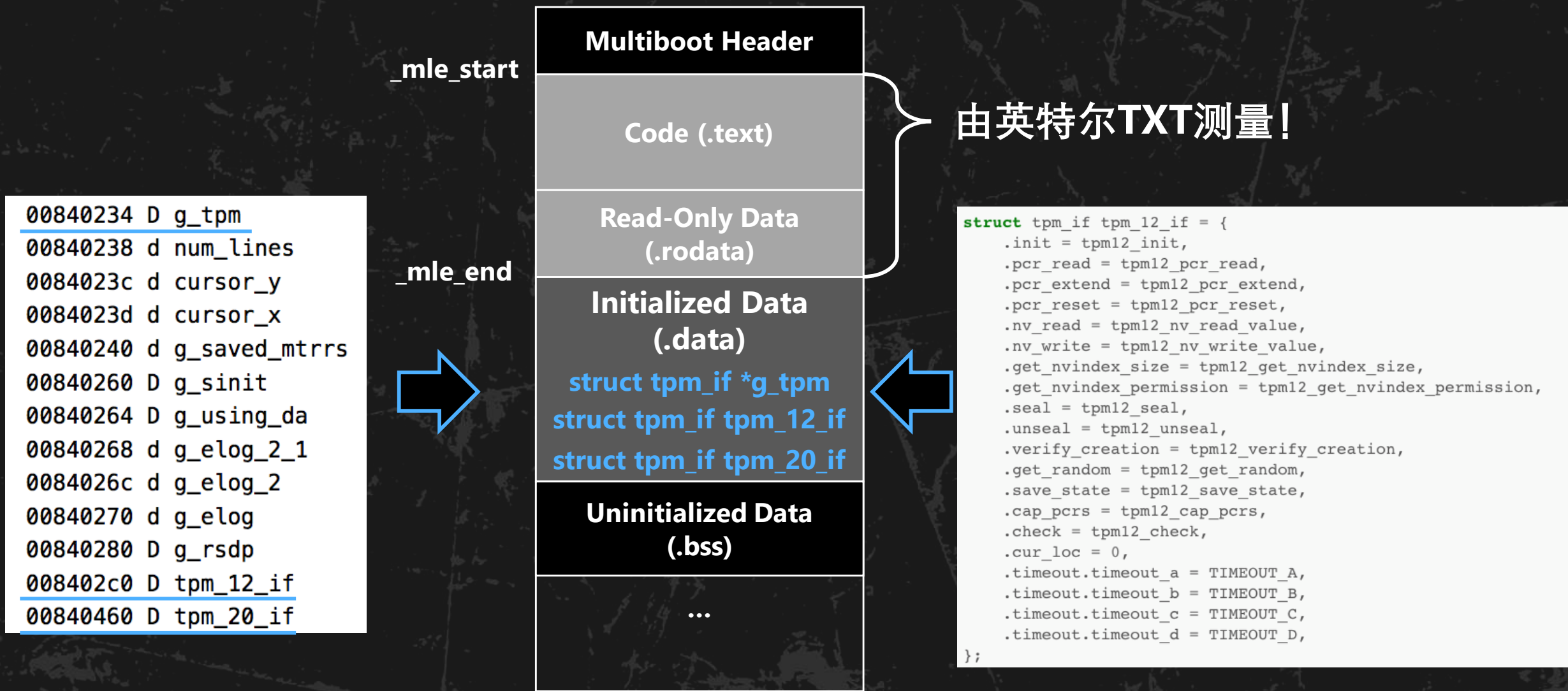


使用tBoot进行休眠的过程

- Seal S3 key and MAC of Kernel Memory with Post-Launch PCRs
 - seal_post_k_state() → `g_tpm->seal()`
- Save Static PCRs(0~16)
 - tpm->save_state()
- Shutdown Intel TXT
 - txt_shutdown()
- Sleep. 关闭CPU和TPM!
 - shutdown_system()
- Wake Up, Restore Static PCRs, and Rest
 - Real Mode, Single CPU
- Launch MLE again and then, Unseal S3 PCRs
 - begin_launch() → txt_s3_launch_environment()
 - post_launch() → s3_launch() → verify_integrity() → `g_tpm->unseal()`
- Extend PCRs and Resume Kernel
 - verify_integrity() → extends_pcrs() → `g_tpm->extend()`
 - s3 launch()-> prot to real()



“丢失的指针”漏洞 (CVE-2017-16837)



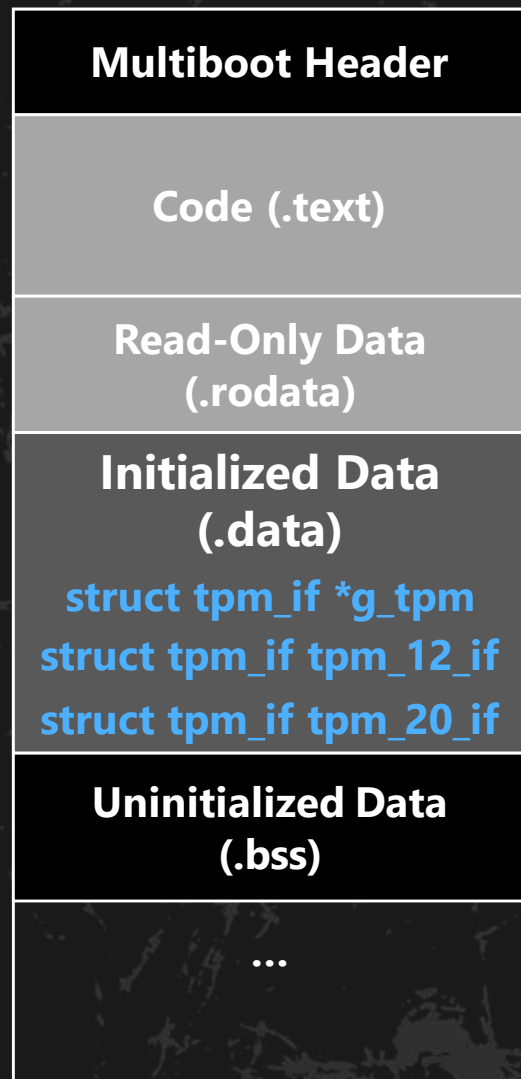
Memory Layout of tBoot

“丢失的指针”漏洞 (CVE-2017-16837)

```
00840234 D g_tpm  
00840238 d num_lines  
0084023c d cursor_y  
0084023d d cursor_x  
00840240 d g_saved_mtrrs  
00840260 D g_sinit  
00840264 D g_using_da  
00840268 d g_eelog_2_1  
0084026c d g_eelog_2  
00840270 d g_eelog  
00840280 D g_rsdp  
008402c0 D tpm_12_if  
00840460 D tpm_20_if
```

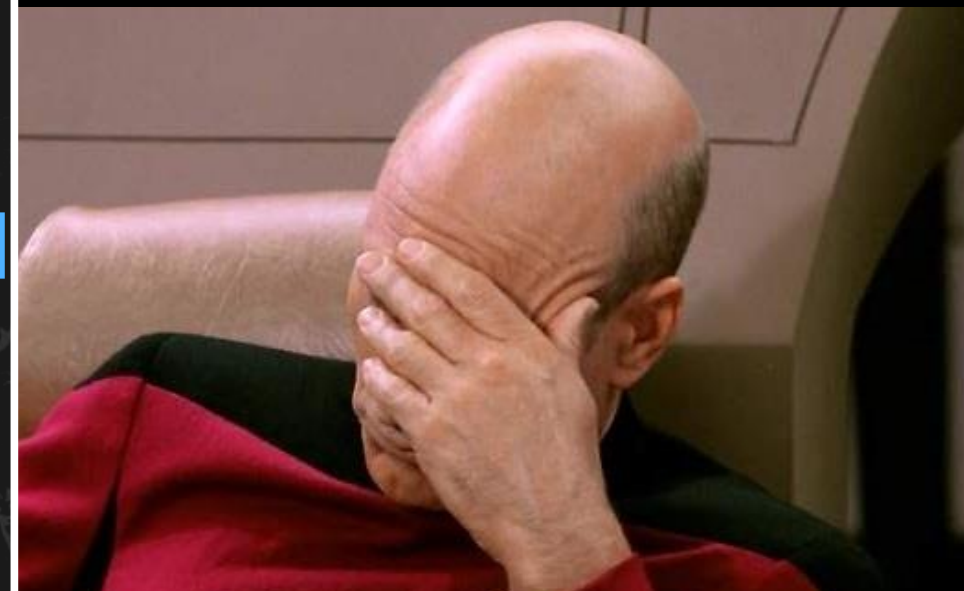
_mle_start

_mle_end



由英特尔TXT测量!

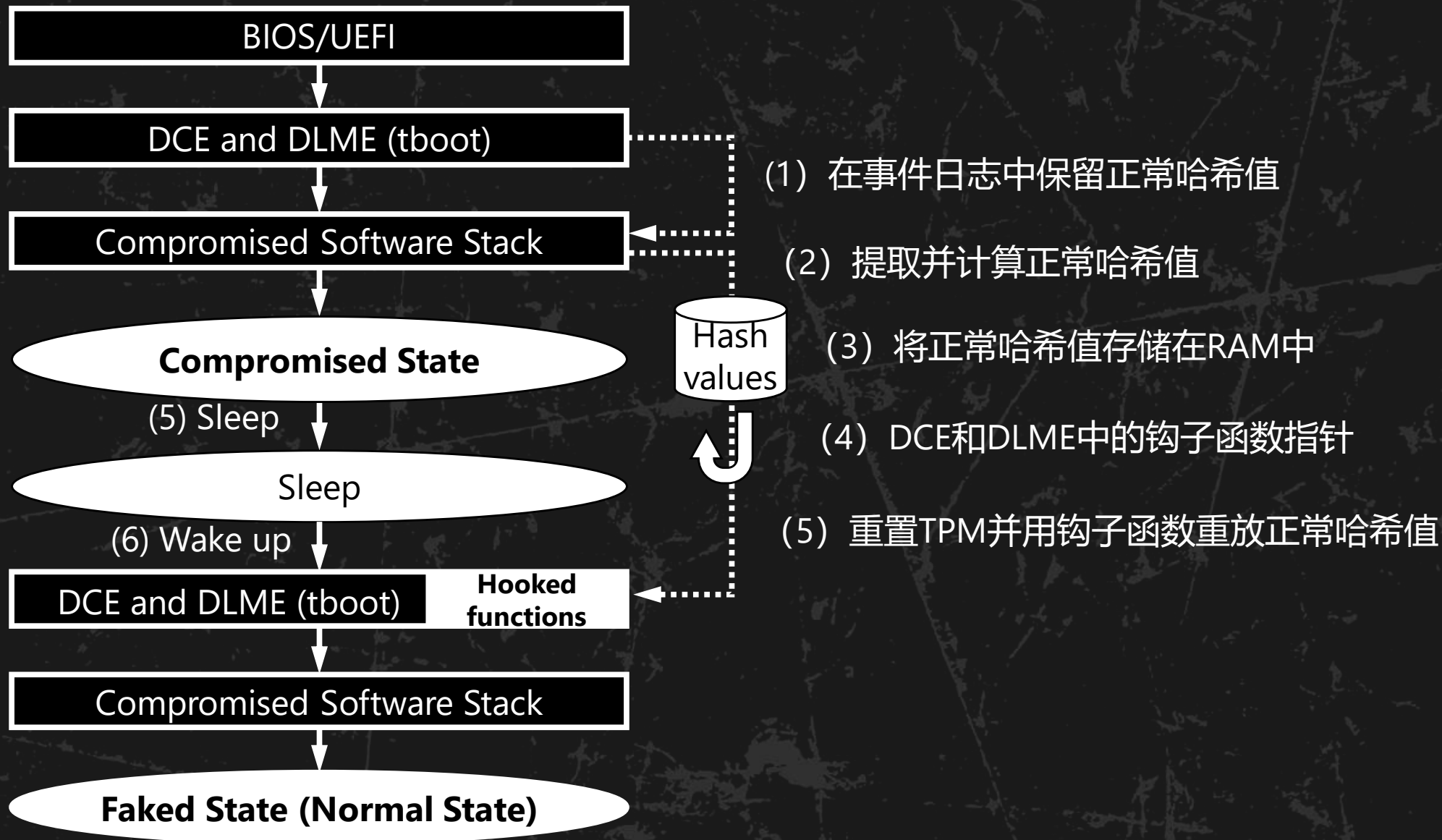
不可测量!



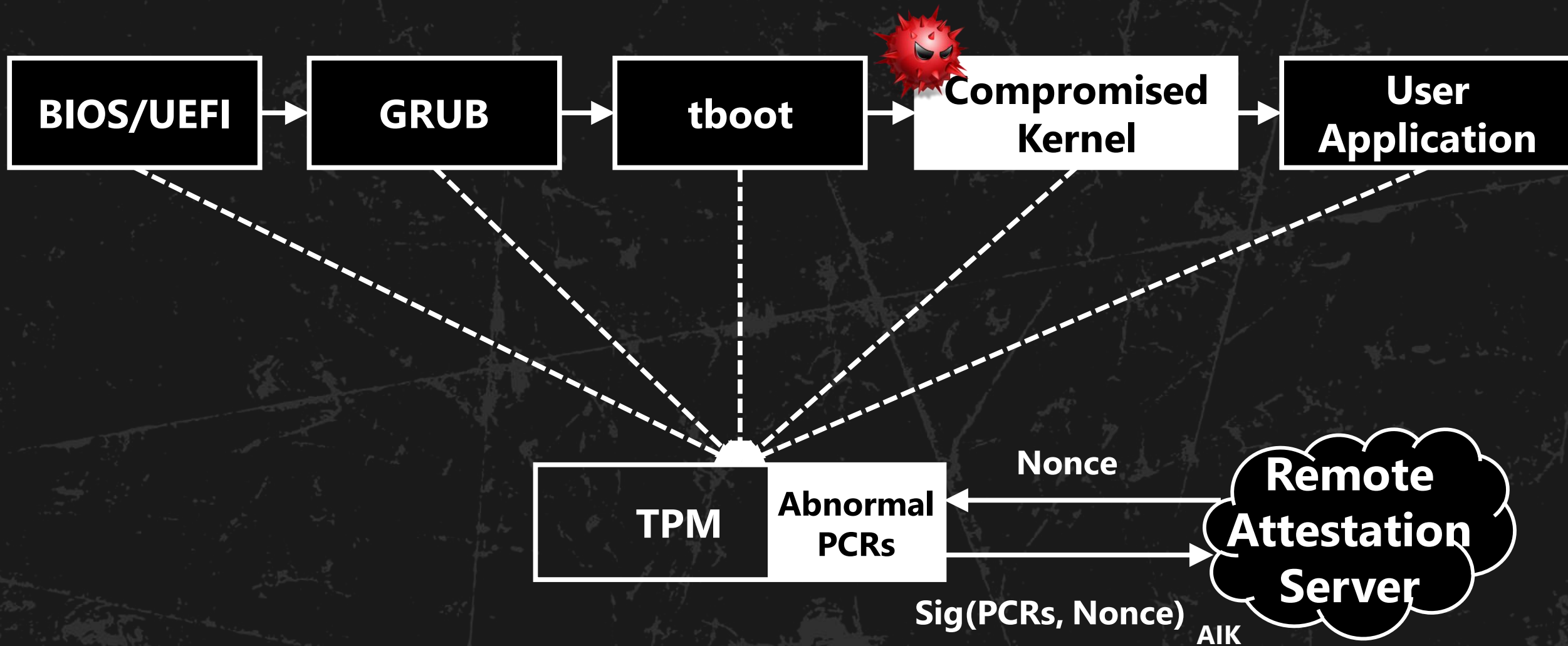
你背叛我!

Memory Layout of tBoot

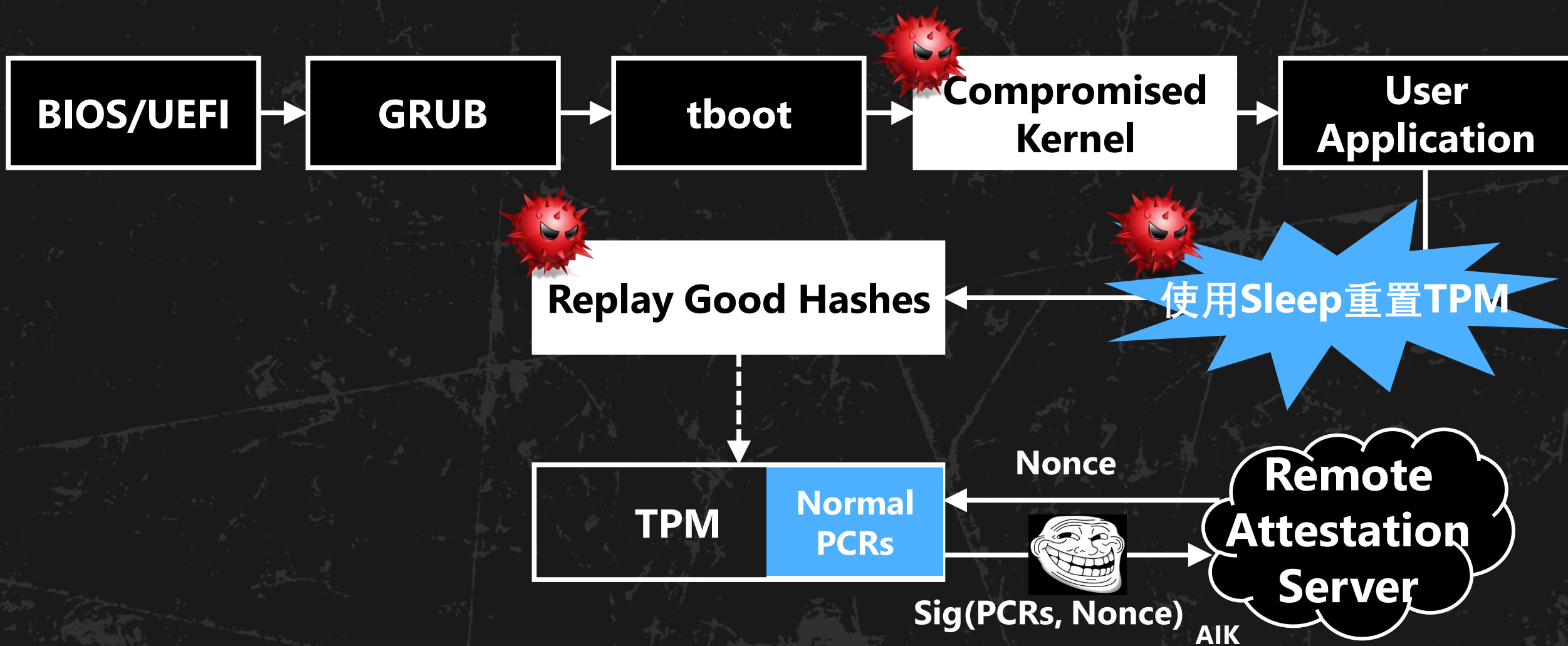
利用CVE-2017-16837的情景(1)



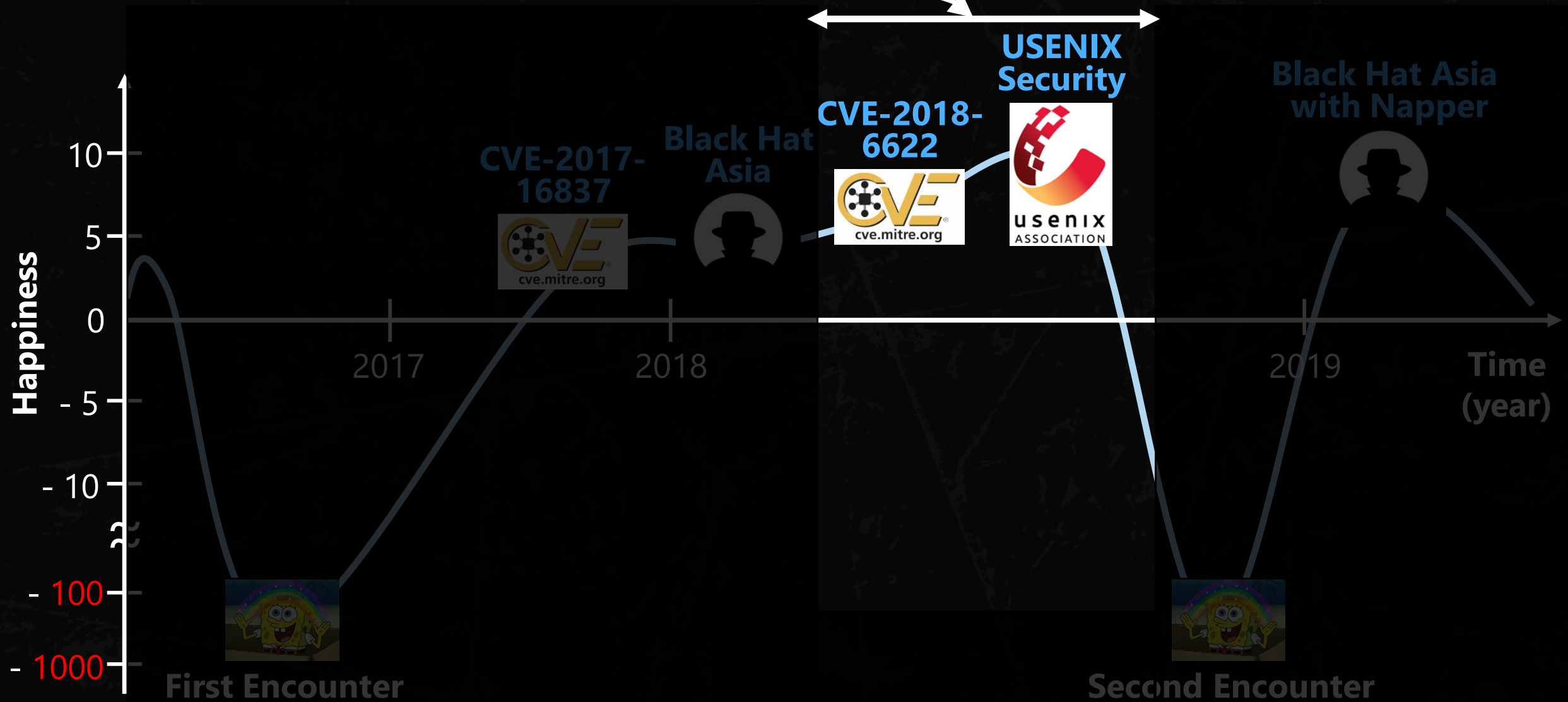
利用CVE-2017-16837的情景(2)



利用CVE-2017-16837的情景(3)



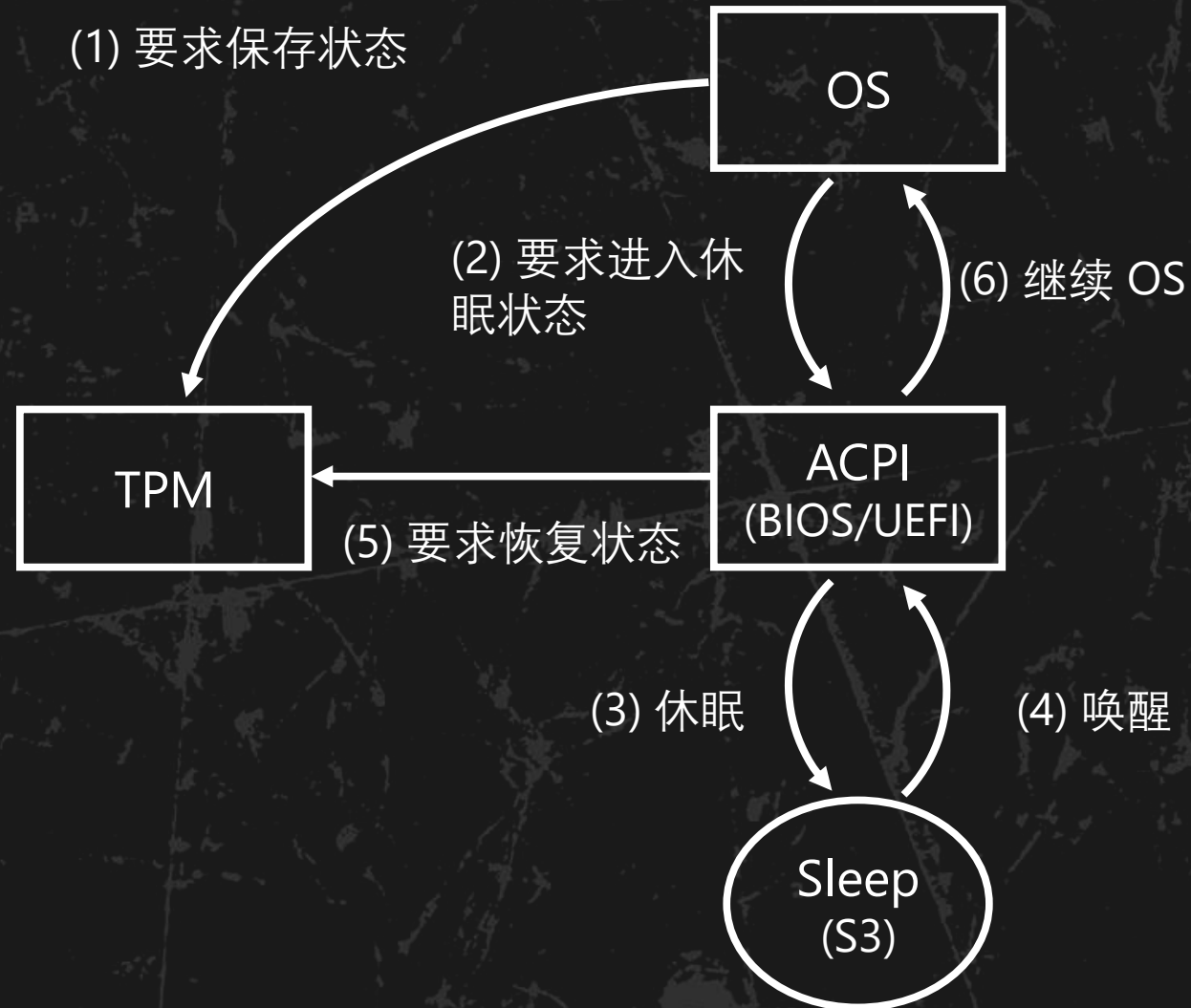
内容- CVE-2018-6622



DRTM在唤醒时测量代码!

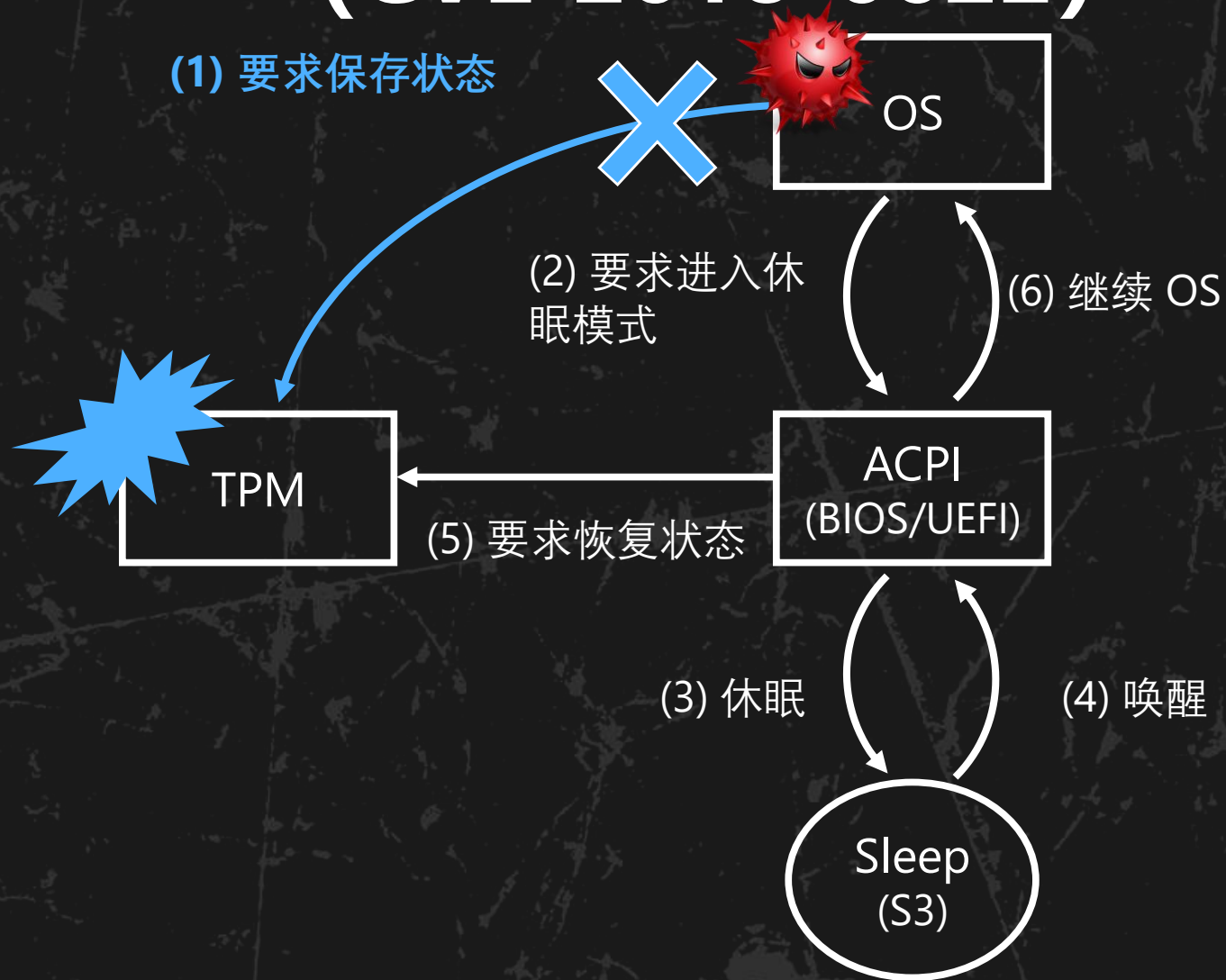
SRTM怎么样??

唤醒SRTM的过程



“灰色地带”漏洞 (1)

(CVE-2018-6622)



“灰色地带”漏洞 (2) (CVE-2018-6622)

TPM 2.0

什么是“纠正措施”?

If the TPM receives Startup(STATE) that was not preceded by Shutdown(STATE), then there is no state to restore and the TPM will return TPM_RC_VALUE. The CRTM is expected to take corrective action to prevent malicious software from manipulating the PCR values such that they would misrepresent the state of the platform. The CRTM would abort the Startup(State) and restart with Startup(CLEAR).

这意味着“重置TPM”

TPM 1.2

The startup behavior defined by this specification is different than TPM 1.2 with respect to Startup(STATE). A TPM 1.2 device will enter Failure Mode if no state is available when the TPM receives Startup(STATE). This is not the case in this specification. It is up to the CRTM to take corrective action if it the TPM returns TPM_RC_VALUE in response to Startup(STATE).

< 可信平台模块库第1部分：体系结构规范 >

我不知道“纠正措施”
我什么都做不了!

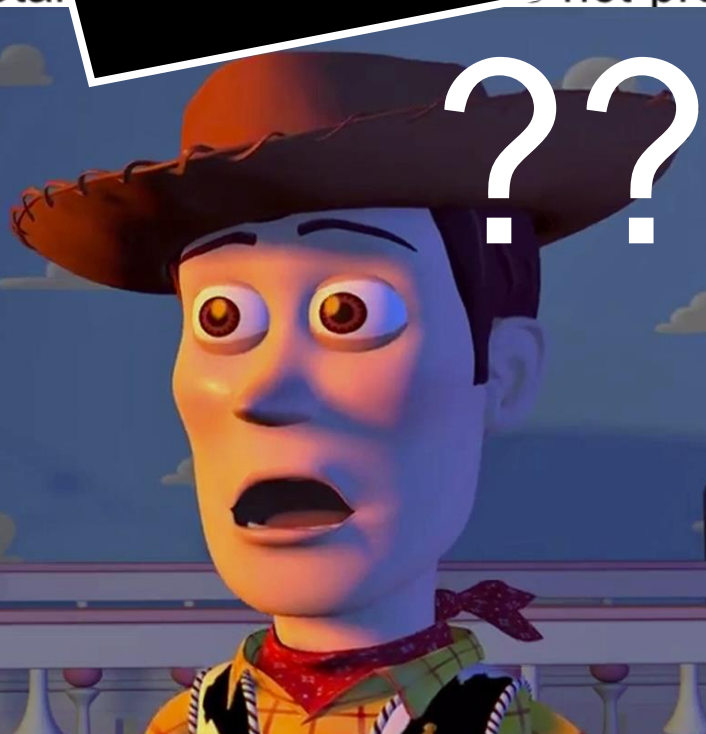
TPM 2.0

If the TPM receives a command to restore and the TPM cannot prevent malicious state of the platform

This means

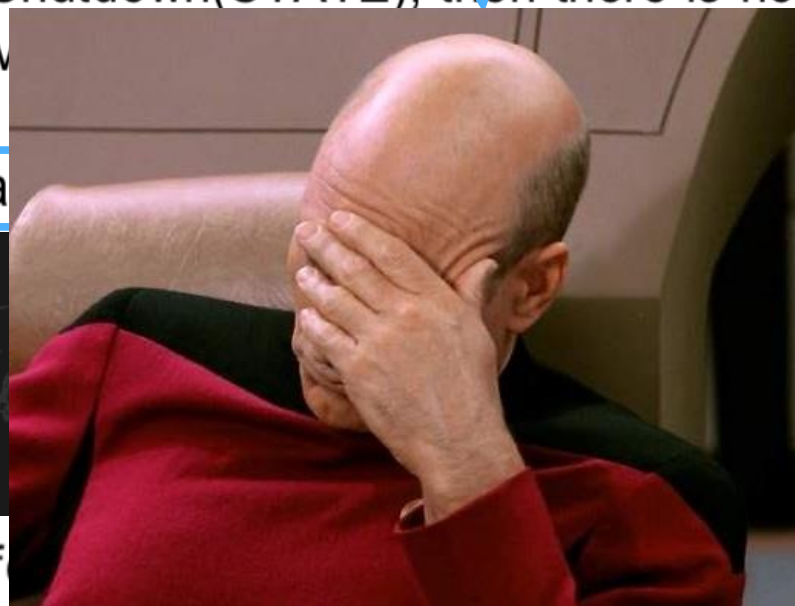
TPM 1.2

The startup behavior of the TPM is Startup(STATE). A TPM receives Startup(STATE) and takes corrective action if it the TPM returns TPM_RC_VALUE in response to Startup(STATE)

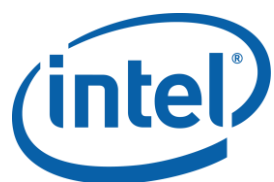


The CRTM CR values are set up(State) and

on is different from Secure Mode this specific



你背叛我!



Lenovo

ASUS

GIGABYTE™

“灰色地带”漏洞 (2) (CVE-2018-6622)

TPM 2.0

什么是“纠正措施”?

If the TPM receives Startup(STATE) that was not preceded by Shutdown(STATE), then there is no state to restore and the TPM will return TPM_RC_VALUE. The CRTM is expected to take **corrective action** to prevent malicious software from manipulating the PCR values such that they would misrepresent the state of the platform. **The CRTM would abort the Startup(State) and restart with Startup(CLEAR).**

这意味着“重置TPM”

TPM 1.2

The startup behavior defined by this specification is different than TPM 1.2 with respect to Startup(STATE). A TPM 1.2 device will enter Failure Mode if no state is available when the TPM receives Startup(STATE). This is not the case in this specification. It is up to the CRTM to take corrective action if it the TPM returns TPM_RC_VALUE in response to Startup(STATE).

<可信平台模块库第1部分：体系结构规范>

Bank/Algorithm: TPM_ALG_SHA1(0x0004)

```

PCR_00: 3d ca ea 25 dc 86 55 4d 94 b
PCR_01: b2 a8 3b 0e bf 2f 83 74 29
PCR_02: b2 a8 3b 0e bf 2f 83 74 29
PCR_03: b2 a8 3b 0e bf 2f 83 74 29
PCR_04: 1c 25 49 f2 27 42 98 48 bd e
PCR_05: cd ca c6 1f 16 b2 22 b8 00 7
PCR_06: b2 a8 3b 0e bf 2f 83 74 29
PCR_07: 40 37 33 6f a7 bc 0e ab e3 7
PCR_08: 6b 0f 47 1f 31 a7 0f e0 ec 1
PCR_09: 77 67 e9 eb 68 d7 bc e7 7a c
PCR_10: 3c 72 6c db 57 ba a5 08 02 8
PCR_11: 00 00 00 00 00 00 00 00 00
PCR_12: 00 00 00 00 00 00 00 00 00
PCR_13: 00 00 00 00 00 00 00 00 00
PCR_14: 00 00 00 00 00 00 00 00 00
PCR_15: 00 00 00 00 00 00 00 00 00
PCR_16: 00 00 00 00 00 00 00 00 00
PCR_17: ff ff ff ff ff ff ff ff ff
PCR_18: ff ff ff ff ff ff ff ff ff
PCR_19: ff ff ff ff ff ff ff ff ff
PCR_20: ff ff ff ff ff ff ff ff ff
PCR_21: ff ff ff ff ff ff ff ff ff
PCR_22: ff ff ff ff ff ff ff ff ff
PCR_23: 00 00 00 00 00 00 00 00 00

```

Bank/Algorithm: TPM ALG SHA1(0x0004)

```

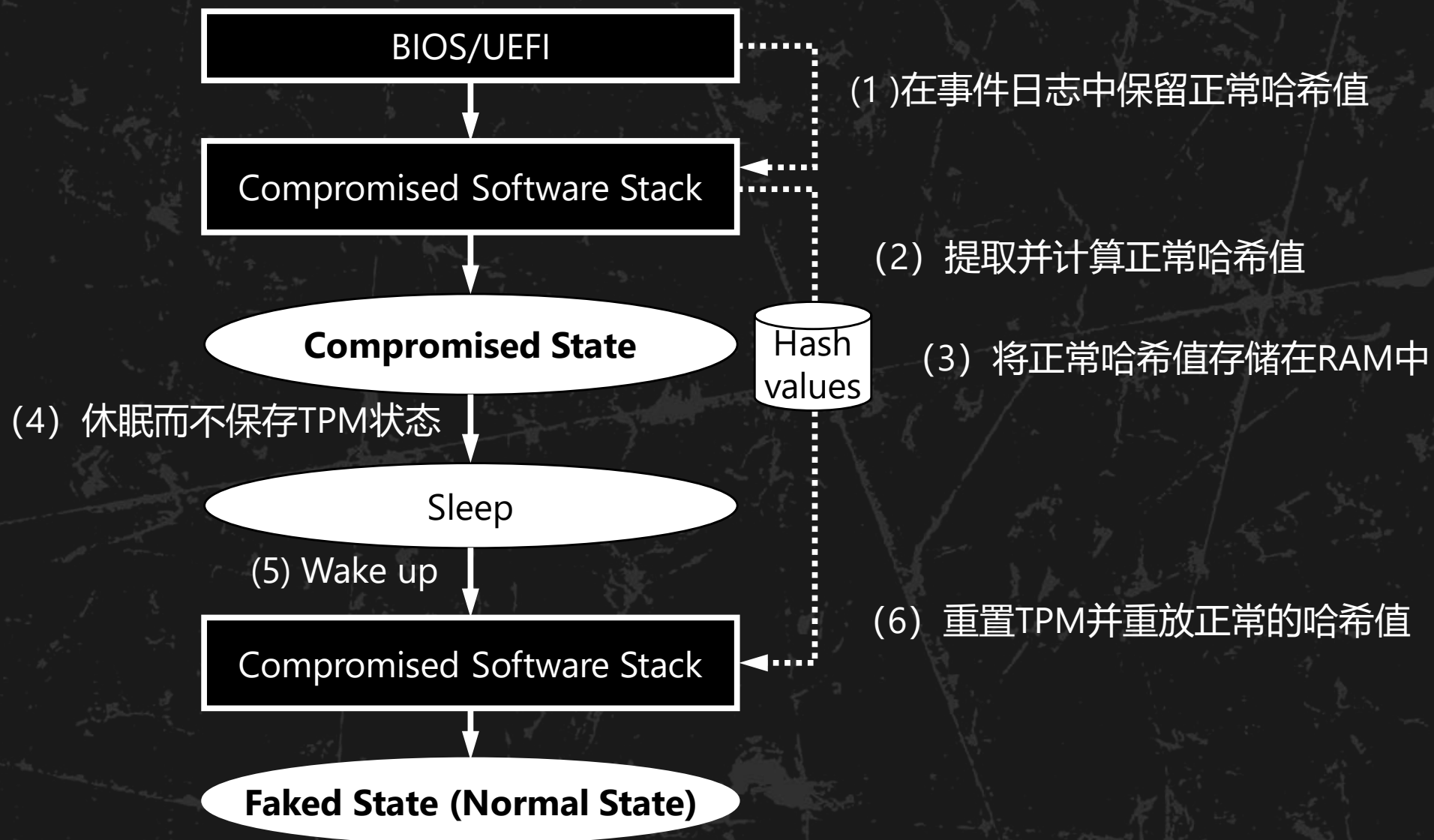
PCR_00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_01: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_02: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_03: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_04: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_05: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_06: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_07: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_08: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_09: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_11: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_12: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_13: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_14: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_15: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_16: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR_17: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
PCR_18: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
PCR_19: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
PCR_20: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
PCR_21: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
PCR_22: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
PCR_23: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

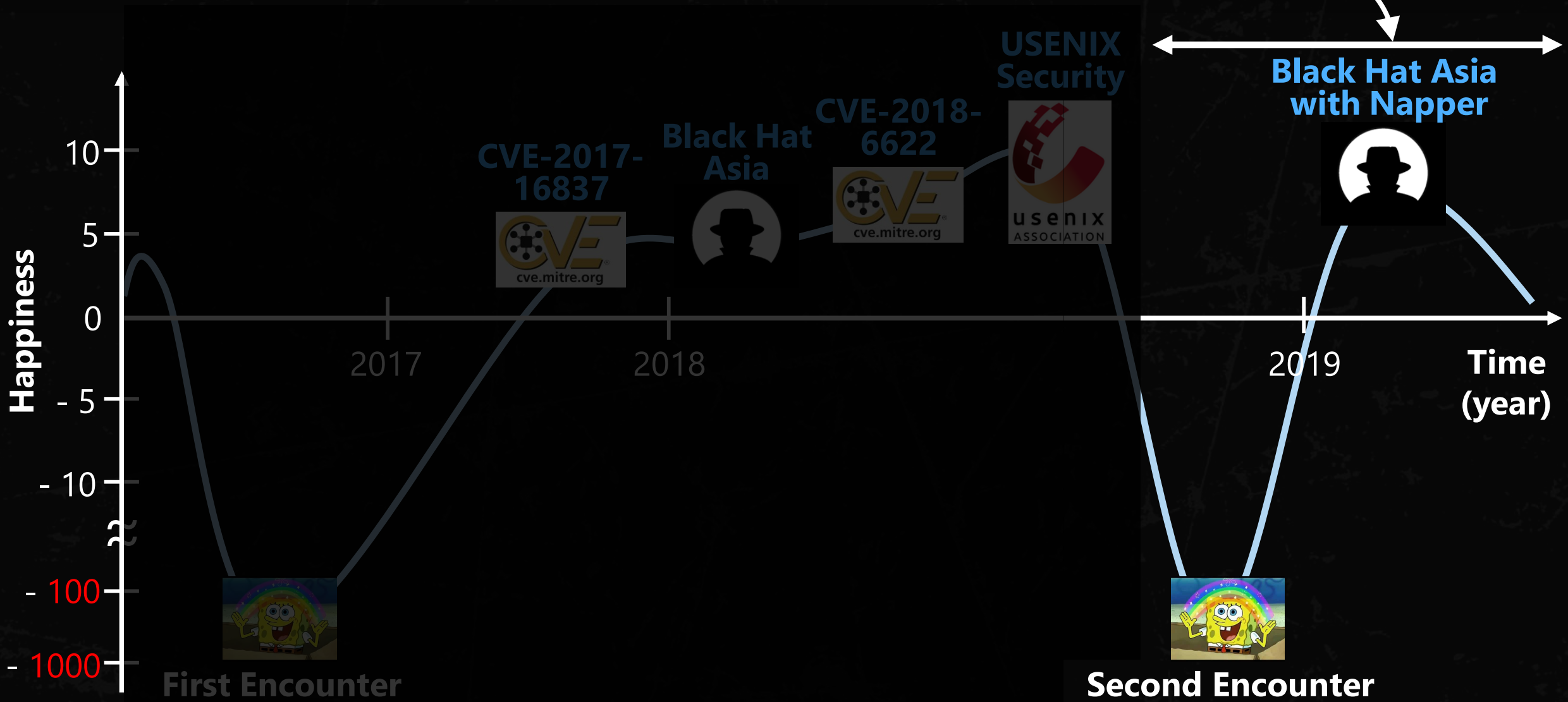
睡眠

安全!

利用CVE-2018-6622的方案



内容 - "Napper"





又是你!

经理



愿景风暴!

二次遭遇!!!

“Napper”?

- 是否可以检查TPM中的ACPI S3睡眠模式漏洞的工具
 - 它是一个基于Ubuntu 18.04的可启动USB设备
 - 它有一个内核模式和用户态应用程序



- 使系统“小睡”并检查漏洞
 - 内核模块通过修补内核代码在休眠时利用灰色区域漏洞 (CVE-2018-6622)
 - 用户级应用程序检查TPM状态并显示报告

“Napper”?

- 是否可以检查TPM中的ACPI S3睡眠模式漏洞的工具
 - 它是一个基于Ubuntu 18.04的可启动USB设备
 - 它有一个内核模式和用户态应用程序
- 使系统“小睡”并检查漏洞




CVE-2017-16837是一个软件漏洞!
如果版本低于**v1.9.7**, 请升级**tBoot**

Napper内核模块(1)

- 在TPM驱动程序中修补 `tpm_pm_suspend ()` 函数
 - S3休眠序列时内核调用该函数
 - 内核模块将函数更改为“`return 0;`”

```
1 int tpm_pm_suspend(struct device *dev)
2 {
3     struct tpm_chip *chip = dev_get_drvdata(dev);
4     struct tpm_cmd_t cmd;
5     int rc, try;
6
7     u8 dummy_hash[TPM_DIGEST_SIZE] = { 0 };
8
9     if (chip == NULL)
10        return -ENODEV;
11
12    if (chip->flags & TPM_CHIP_FLAG_ALWAYS_SUSPEND)
13        return 0;
14
15    if (chip->flags & TPM_CHIP_FLAG_TPM2) {
16        tpm2_shutdown(chip, TPM2_SU_STATE);
17        return 0;
18    }
```



```
1 int tpm_pm_suspend(struct device *dev)
2 {
3     // Do nothing!
4     return 0;
5 }
```

Napper内核模块(2)

```
1 static int __init napper_init(void)
2 {
3     TEXT_POKE fn_text_poke;
4     unsigned long tpm_suspend_addr;
5
6     // Byte code of "XOR RAX, RAX; RET;"
7     unsigned char ret_op_code[] = {0x48, 0x31, 0xC0, 0xC3};
8     unsigned char org_op_code[sizeof(ret_op_code)];
9
10    // Find needed functions
11    fn_text_poke = (TEXT_POKE) kallsyms_lookup_name("text_poke");
12    tpm_suspend_addr = kallsyms_lookup_name("tpm_pm_suspend");
13
14    // Backup code and patch it
15    memcpy(org_op_code, (unsigned char*) tpm_suspend_addr, sizeof(org_op_code));
16    fn_text_poke((void*) tpm_suspend_addr, ret_op_code, sizeof(ret_op_code));
17
18    return 0;
19 }
```


Napper用户态应用程序

- 由TPM相关软件和启动器软件组成
 - 我在tpm2_tools中添加了一个命令行工具“tpm2_extendpcrs”
 - 我还制作了一个易于使用的启动器软件
- 加载内核模块并检查TPM漏洞
 - 发射器加载napper的内核模块并小睡一会儿
 - 检查TPM的PCR是否都是ZEROS并扩展PCR
 - 使用tpm2_getinfo, dmidecode和journalctl工具收集和报告TPM和系统信息

Napper Live-CD和USB可启动设备



Ubuntu 18.04

+ Kernel 4.18.0-15

+ TPM-related software

+ User-level Applications

+ Pinguybuilder_5.1-7

Napper Live-CD.iso

Napper Live-CD和USB可启动设备

Ubuntu 18.04

+ Kernel 4.18.0-15



Project page:

<https://github.com/kkamagui/napper-for-tpm>



+ User level Applications

Pinguybuilder_5.1-7

Napper Live-CD.iso

Model	Status	BIOS			TPM	
		Vendor	Version	Release Date	Manufacturer	Vendor String
ASUS Q170M-C	Vulnerable	American Megatrends Inc.	4001	11/09/2018	Infineon (IFX)	SLB9665
Dell Optiplex 7040	Vulnerable	Dell	1.11.1	10/10/2018	NTC	rls NPCT
Dell Optiplex 7050	Vulnerable	Dell	1.11.0	11/01/2018	NTC	rls NPCT
GIGABYTE H170-D3HP	Vulnerable	American Megatrends Inc.	F20g	03/09/2018	Infineon (IFX)	SLB9665
GIGABYTE Q170M-MK	Vulnerable	American Megatrends Inc.	F23	04/12/2018	Infineon (IFX)	SLB9665
HP Spectre x360	Vulnerable	American Megatrends Inc.	F.24	01/07/2019	Infineon (IFX)	SLB9665
Intel NUC5i5MYHE	Vulnerable	Intel	MYBDWi5v.86A. 0049.2018. 1107.1046	11/07/2018	Infineon (IFX)	SLB9665
Lenovo T480 (20L5A00TKR)	Safe	Lenovo	N24ET44W (1.19)	11/07/2018	Infineon (IFX)	SLB9670
Lenovo T580	Safe	Lenovo	N27ET20W (1.06)	01/22/2018	ST- Microelectronics	
Microsoft Surface Pro 4	Safe	Microsoft Corporation	108.2439.769	12/07/2018	Infineon (IFX)	SLB9665

对策 - CVE-2018-6622

(灰色区域漏洞)

- 1) 在BIOS菜单中禁用ACPI S3睡眠功能
 - 残酷，但简单而有效
- 2) 修改TPM 2.0规范以详细定义“纠正措施”并修补BIOS / UEFI固件
 - 很长一段时间修改并应用于TPM或BIOS / UEFI固件
 - 但是，根本的解决方案！

检查并更新BIOS/UEFI固件！

对策 - CVE-2017-16837 (丢失的指针漏洞)

1) 将我的补丁应用于**tBoot** -

<https://sourceforge.net/p/tboot/code/ci/521c58e51eb5be105a29983742850e72c44ed80e/>

2) 将**tBoot**更新到最新版本

结论

- 到目前为止，我们已经信任不可信赖的硬件和软件！
 - “信誉”不是“可信度”
 - 不仅要信任信誉，还要为自己检查一切
- Napper可帮助您检查TPM漏洞
 - 使用Napper检查您的系统或访问项目站点以获取结果
- 使用最新版本更新BIOS / UEFI固件
 - 如果还没有修补固件，请立即禁用BIOS菜单中的ACPI S3睡眠功能！

BLUEHAT

SHANGHAI 2019

背叛信誉

用信誉为赌注信任不可信的硬件和软件

Seunghun Han

hanseunghun@nsr.re.kr

Twitter:  @kkamagui1

Project: <https://github.com/kkamagui/napper-for-tpm>

参考资料

- Seunghun, H., Wook, S., Jun-Hyeok, P., and HyoungChun K. *Finally, I Can Sleep Tonight: Catching Sleep Mode Vulnerabilities of the TPM with the Napper*. Black Hat Asia. 2019.
- Seunghun, H., Wook, S., Jun-Hyeok, P., and HyoungChun K. *A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping*. USENIX Security. 2018.
- Seunghun, H., Jun-Hyeok, P., Wook, S., Junghwan, K., and HyoungChun K. *I Don't Want to sleep Tonight: Subverting Intel TXT with S3 Sleep*. Black Hat Asia. 2018.
- Trusted Computing Group. *TCG D-RTM Architecture*. 2013.
- Trusted Computing Group. *TCG PC Client Specific Implementation Specification for Conventional BIOS*. 2012.
- Intel. *Intel Trusted Execution Technology (Intel TXT)*. 2017.
- Butterworth, J., Kallenberg, C., Kovah, X., and Herzog, A. *Problems with the static root of trust for measurement*. Black Hat USA. 2013.
- Wojtczuk, R., and Rutkowska, J. *Attacking intel trusted execution technology*. Black Hat DC. 2009.
- Wojtczuk, R., Rutkowska, J., and Tereshkin. A. *Another way to circumvent Intel trusted execution technology*. Invisible Things Lab. 2009.
- Wojtczuk, R., and Rutkowska, J. *Attacking Intel TXT via SINIT code execution hijacking*. Invisible Things Lab. 2011.
- Sharkey, J. *Breaking hardware-enforced security with hypervisors*. Black Hat USA. 2016.