Fault Attacks against your Zen

Jean-Pierre Seifert
Einstein Professor
TU Berlin, Berlin (Germany)
jeanpierreseifert@gmail.com

Based on:

One Glitch to Rule Them All: Fault Injection Attacks Against AMD's Secure Encrypted Virtualization, R. Buhren, H. N. Jacob, T. Krachenfels, J.-P. Seifert, ACM CCS 2021.

Insecure Until Proven Updated: Analyzing AMD SEV's Remote Attestation, R. Buhren, J.-P. Seifert, Christian Werling, ACM CCS 2019.

On authenticated computing and RSA-based authentication, J.-P. Seifert, ACM CCS 2005.

17th September 2021 @ FDTC 2021

Agenda

- Amusing history
- What is Zen/AMD SEV?
- Research question
- Glitching the AMD SP
- Conclusions

Agenda

Amusing history

What is Zen/AMD SEV?

Research question

Glitching the AMD SP

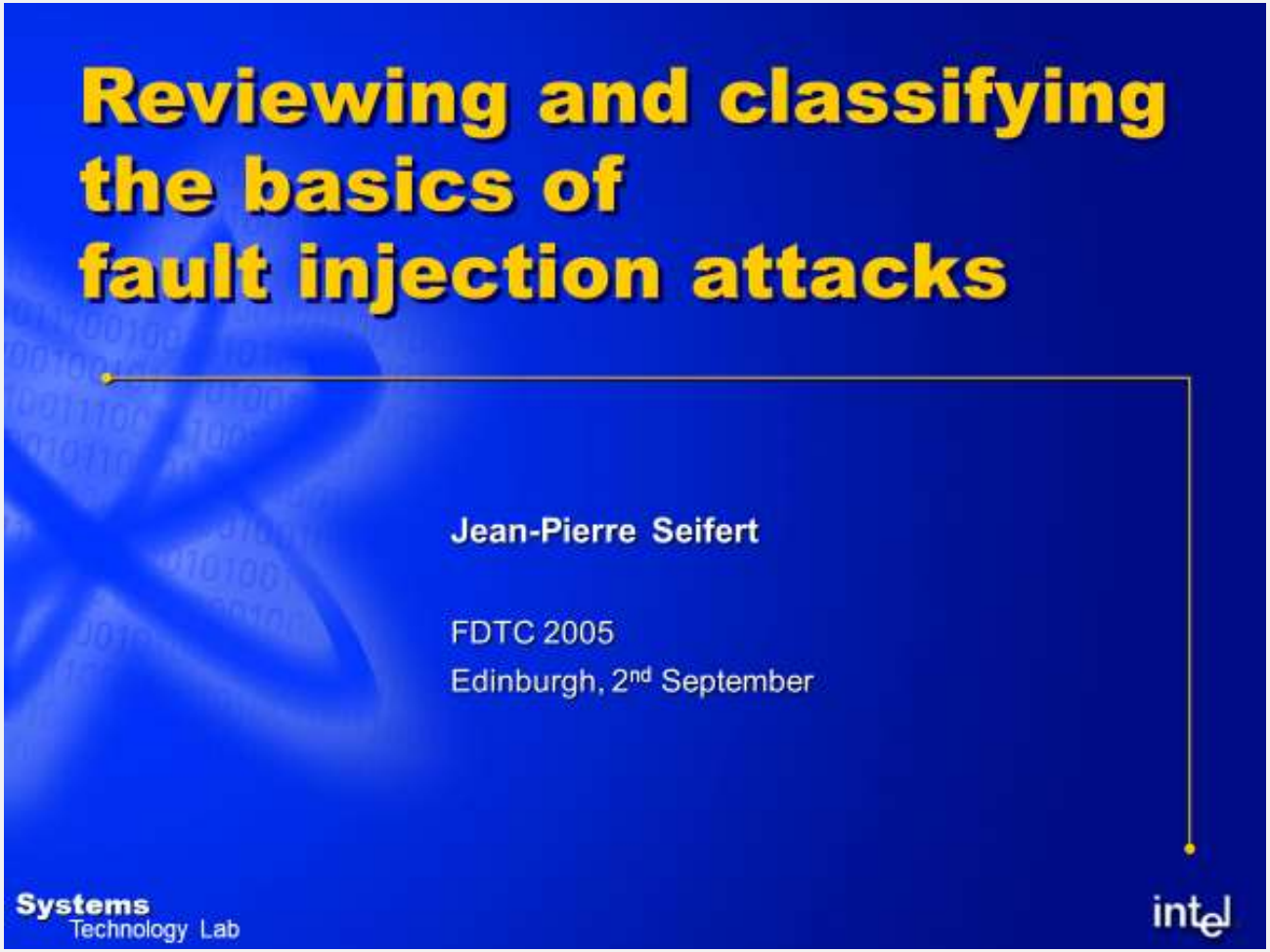
Conclusions

FDTC 2005

Do you remember the invited talk?

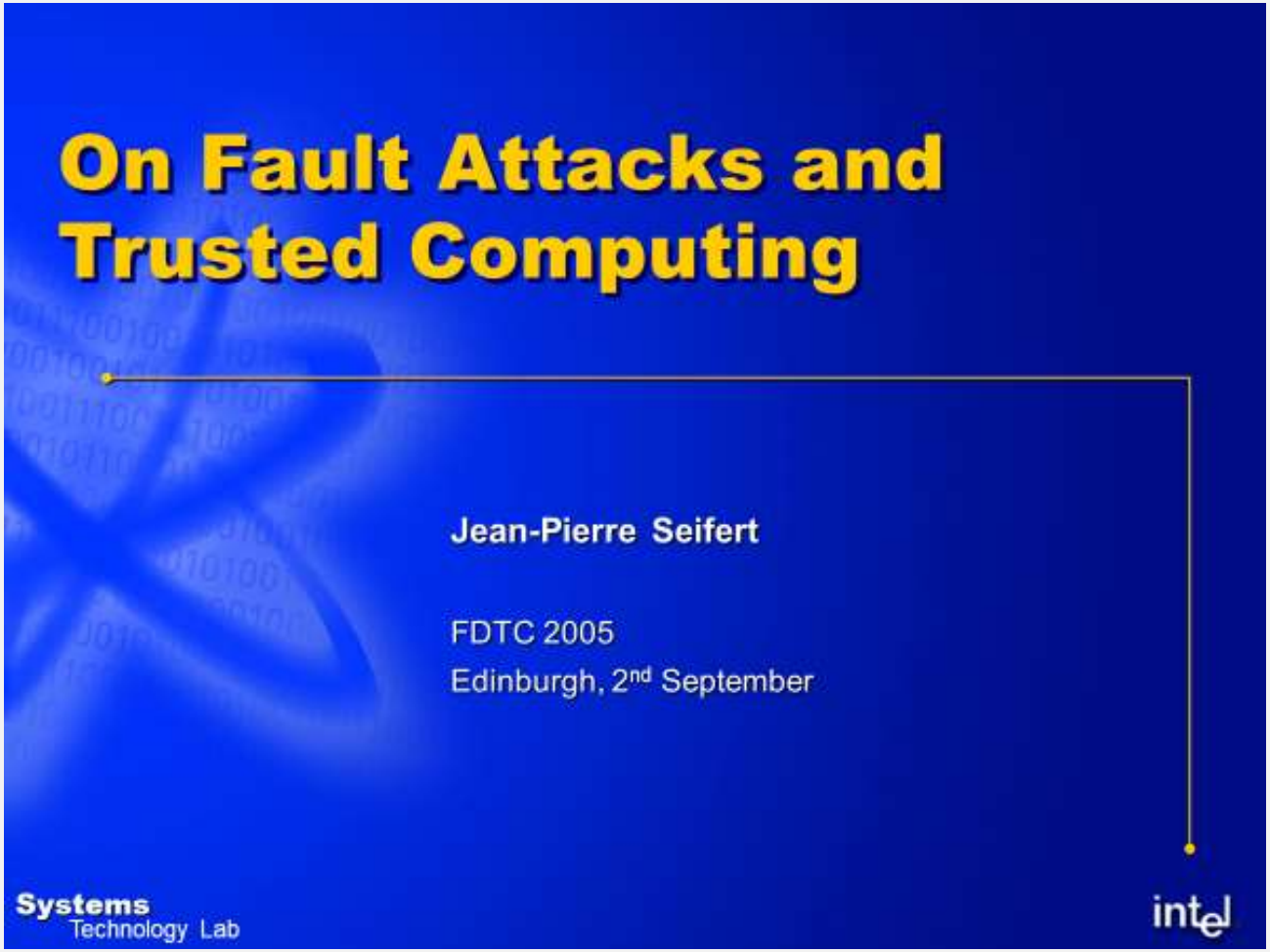
Do you remember the invited talk?

FDTC 2005



FDTC 2005

Do you remember the invited talk?

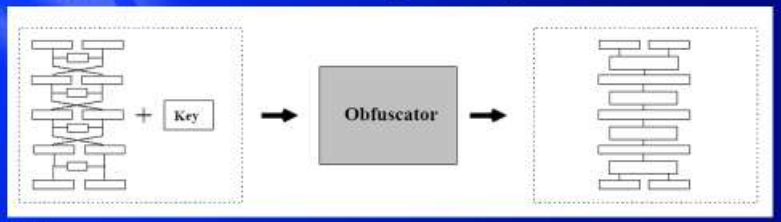


VMs for Fault Injection?

VMM = software that manages the machine's real resource among VMs.

As the VMM has the full control over a VM it is very simple to inject via a VMM from time to time faults into the execution of a cipher.

Application: Very easy way to circumvent sw tamper-sistance enhancing methods like obfuscation - you don´t ve to do the hard reverse-engineering task:



gy Lab 46

Conclusion

It is of great wisdom to have a trustworthy VMM vendor.

Systems Technology Lab 47

Agenda

Amusing history

What is Zen/AMD SEV?

Research question

Glitching the AMD SP

Conclusions

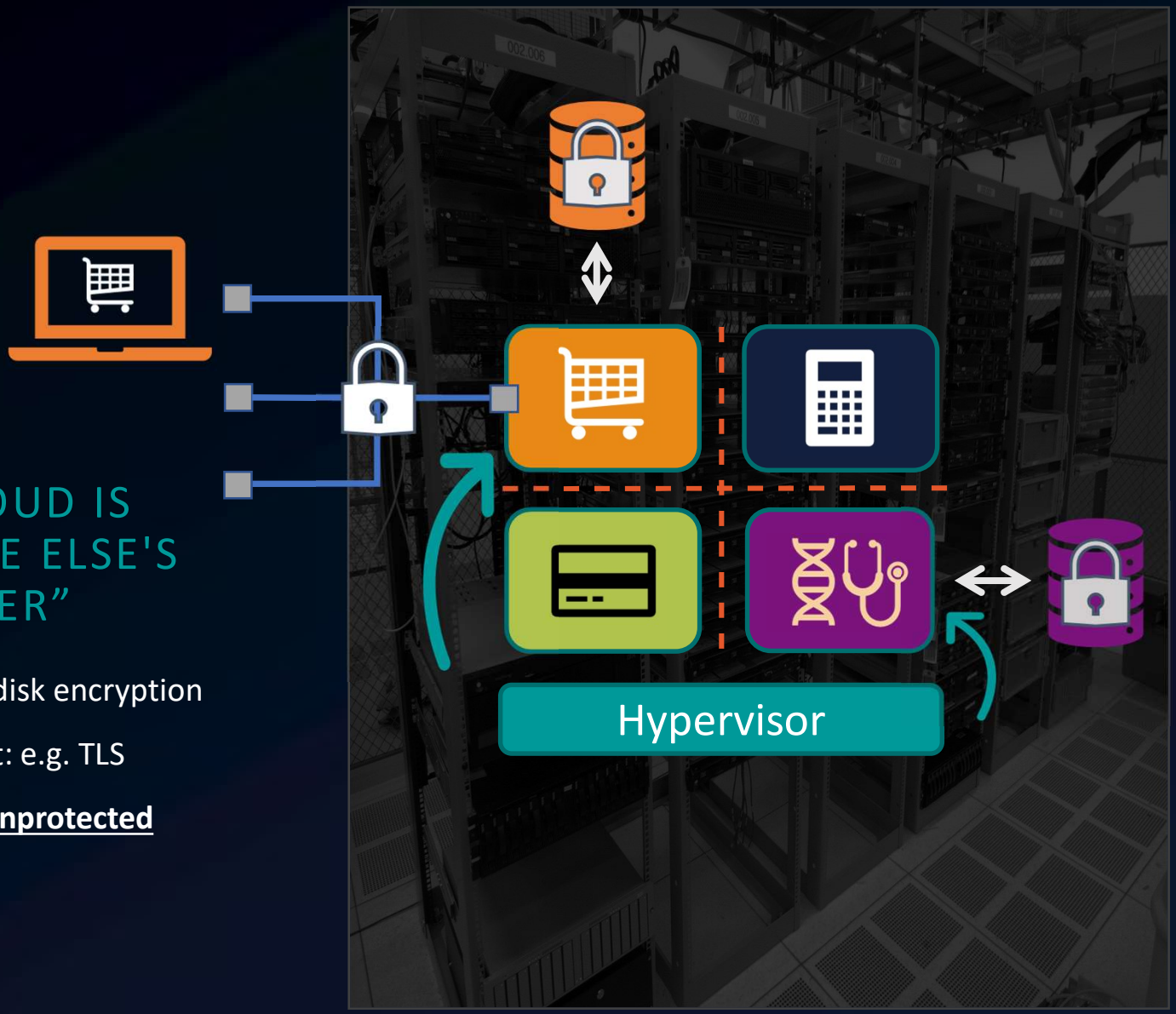
“THE CLOUD IS SOMEONE ELSE'S COMPUTER”

"THE CLOUD IS SOMEONE ELSE'S COMPUTER"

Data-At-Rest: disk encryption

Data-In-Transit: e.g. TLS

Data-In-Use: **unprotected**



SECURE ENCRYPTED VIRTUALIZATION

"... SEV protects data-in-use enabling customer workloads to be protected cryptographically from each other as well as protected from the hosting software.

Even an administrator with malicious intentions at a cloud data center would not be able to access the data in a hosted VM."

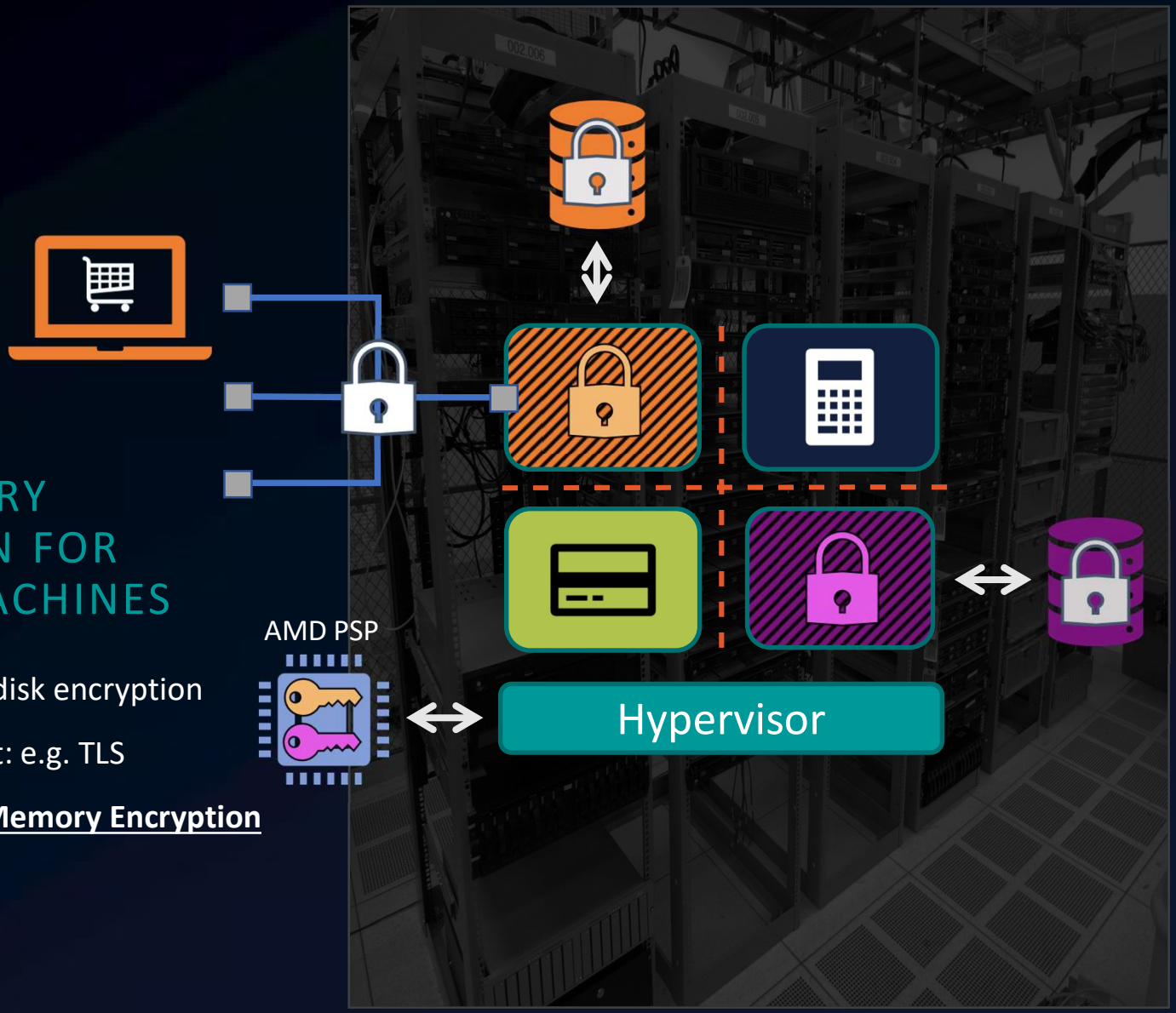
https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf

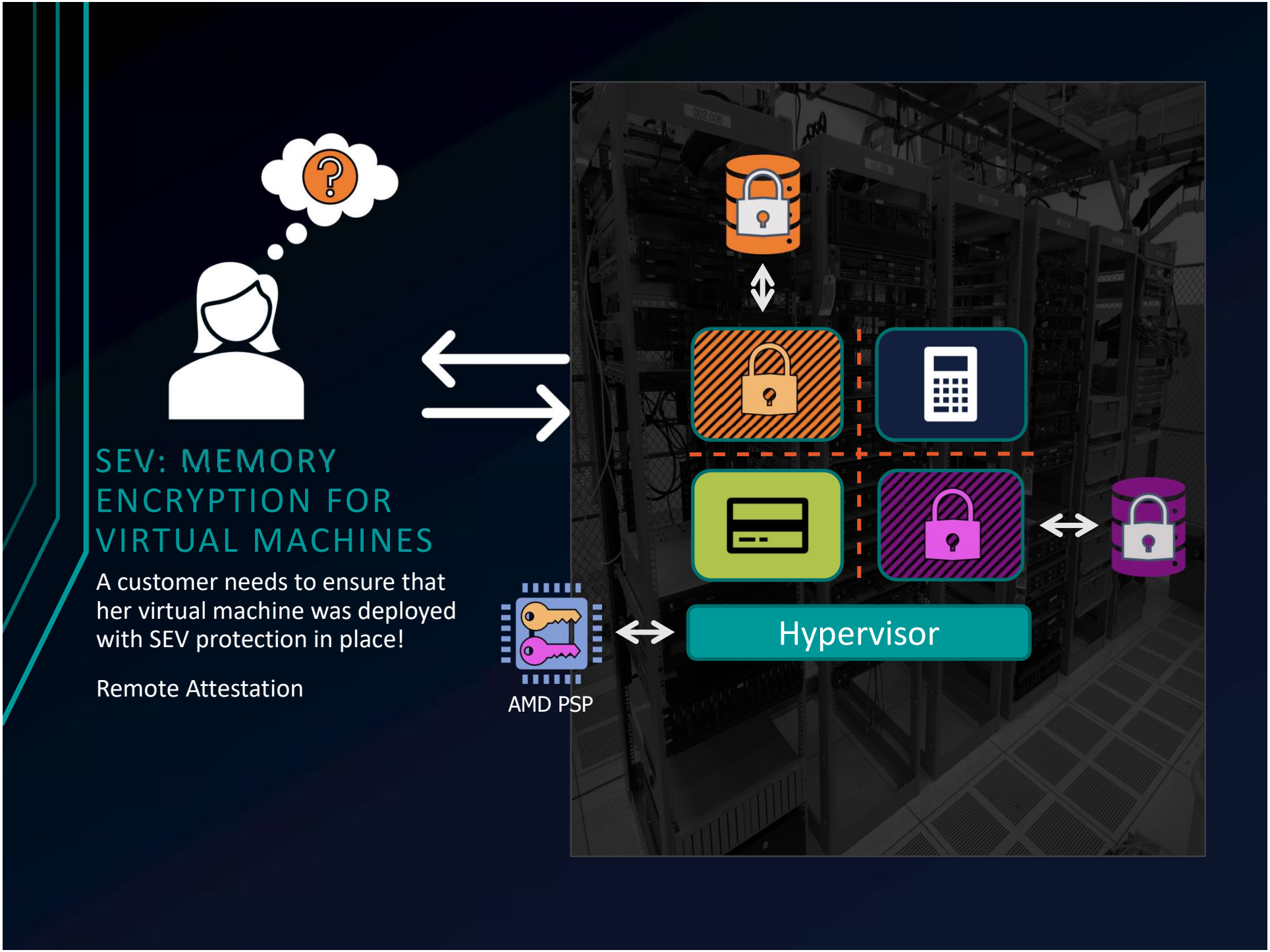
SEV: MEMORY ENCRYPTION FOR VIRTUAL MACHINES

Data-At-Rest: disk encryption

Data-In-Transit: e.g. TLS

Data-In-Use: **Memory Encryption** (AES-128)




SEV: MEMORY ENCRYPTION FOR VIRTUAL MACHINES
A customer needs to ensure that her virtual machine was deployed with SEV protection in place!
Remote Attestation
Hypervisor
AMD PSP

SEV Extensions

SEV

- VM memory encryption
- Guest **registers** are NOT encrypted

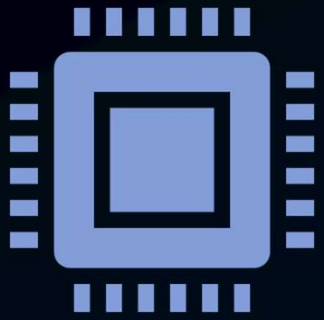
SEV ENCRYPTED STATE

- **register** encryption

SEV SECURE NESTED PAGING

- **software-based** memory integrity protection[1]
- Versioned Chip Endorsement Key (VCEK)
- New VM migration mechanism
- VM privilege levels (VMPLs)
- Trusted platform information (CPUID)

[1] *Fault Attacks on Encrypted General Purpose Compute Platforms,* R. Buhren, S. Gueron, J. Nordholz, J.-P. Seifert, J. Vetter, CODASPY 2017: 197-204



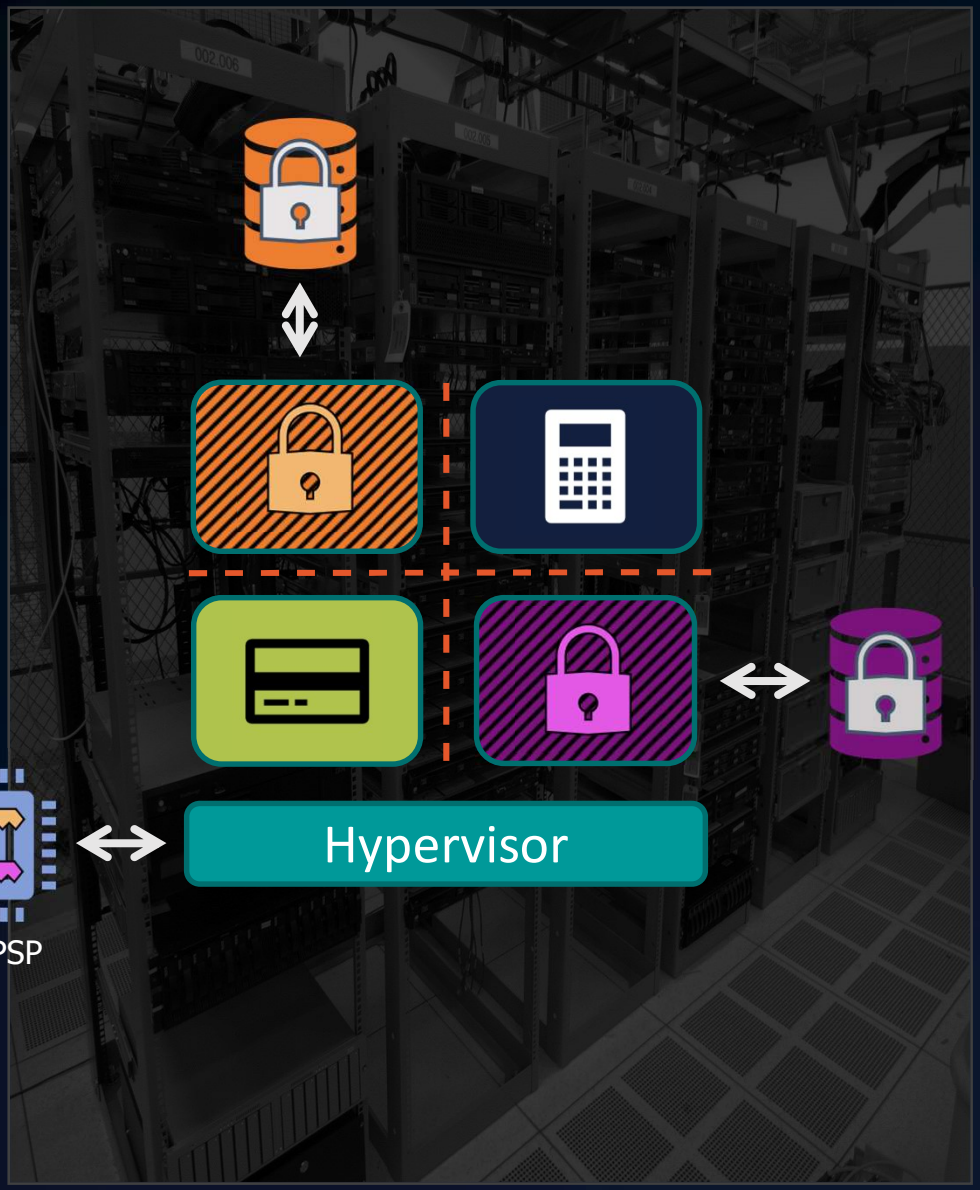
SEV: AMD-SP

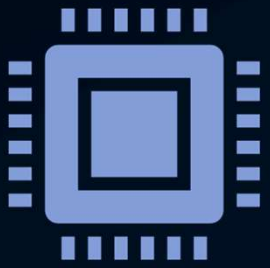
Hosts the SEV firmware that implements the SEV API

Memory encryption keys

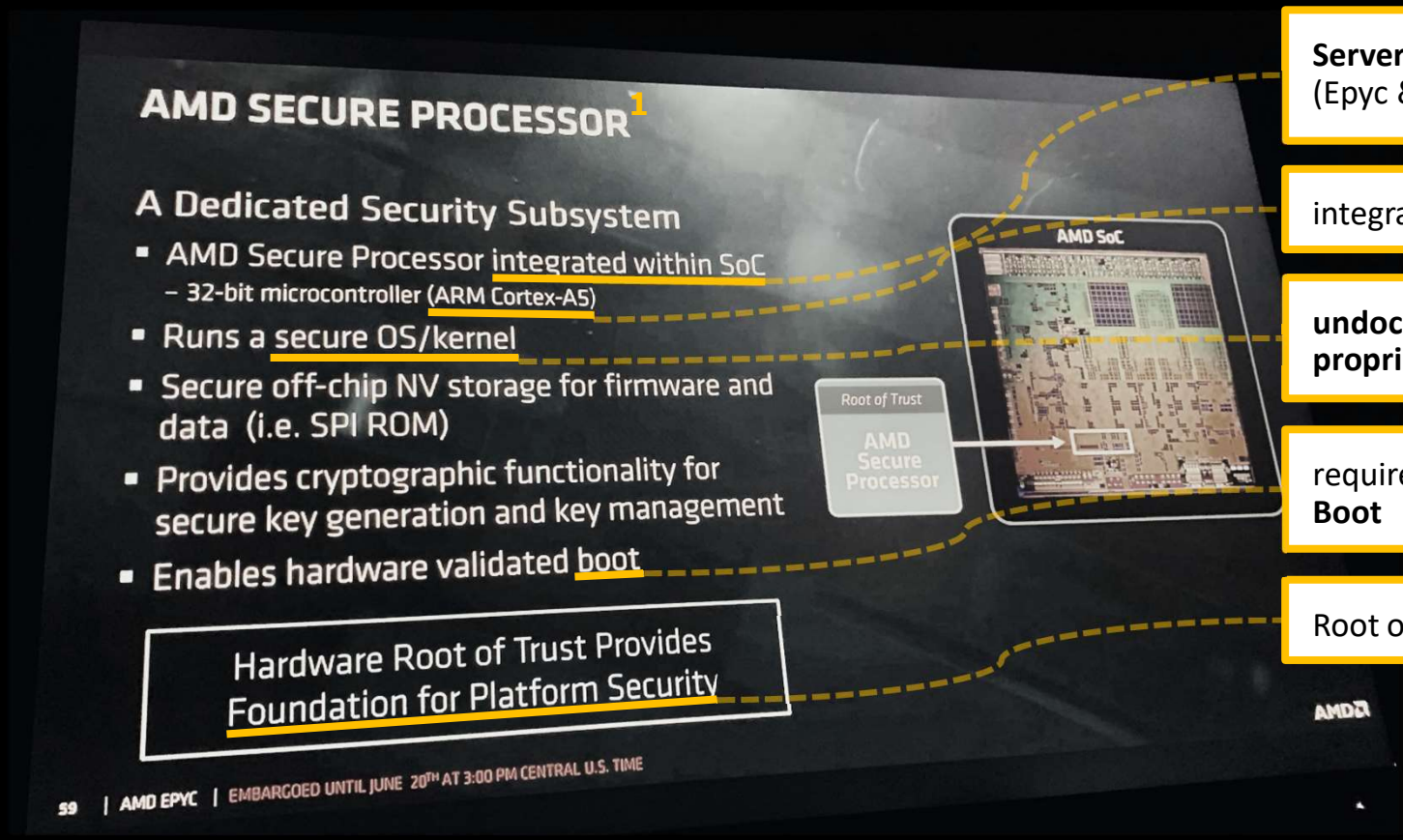
Endorsement keys (CEK / VCEK)

AMD PSP





The AMD Secure Processor



Server & Desktops (Epyc & Ryzen)

integrated **since 2013**

undocumented, **proprietary** firmware

required for **Secure Boot**

Root of trust

[1] Formerly known as *Platform Security Processor (i.e. **PSP**)*

Applications

SECURE ENCRYPTED VIRTUALIZATION (EPYC)

- **SEV p**rotects virtual machines in **untrusted** environments by encrypting VM memory
- SP manages encryption keys
- SP provides remote attestation

SECURE OS (RYZEN)

- Firmware TPM

TRUSTED EXECUTION ENVIRONMENT (RYZEN / EPYC?)

- Trusted Execution Environment
- Linux to support **SP TEE API**[1]

[1] https://www.phoronix.com/scan.php?page=news_item&px=AMD-TEE-Driver-For-Linux-5.6

Agenda

Amusing history

What is Zen/AMD SEV?

Research question

Glitching the AMD SP

Conclusions

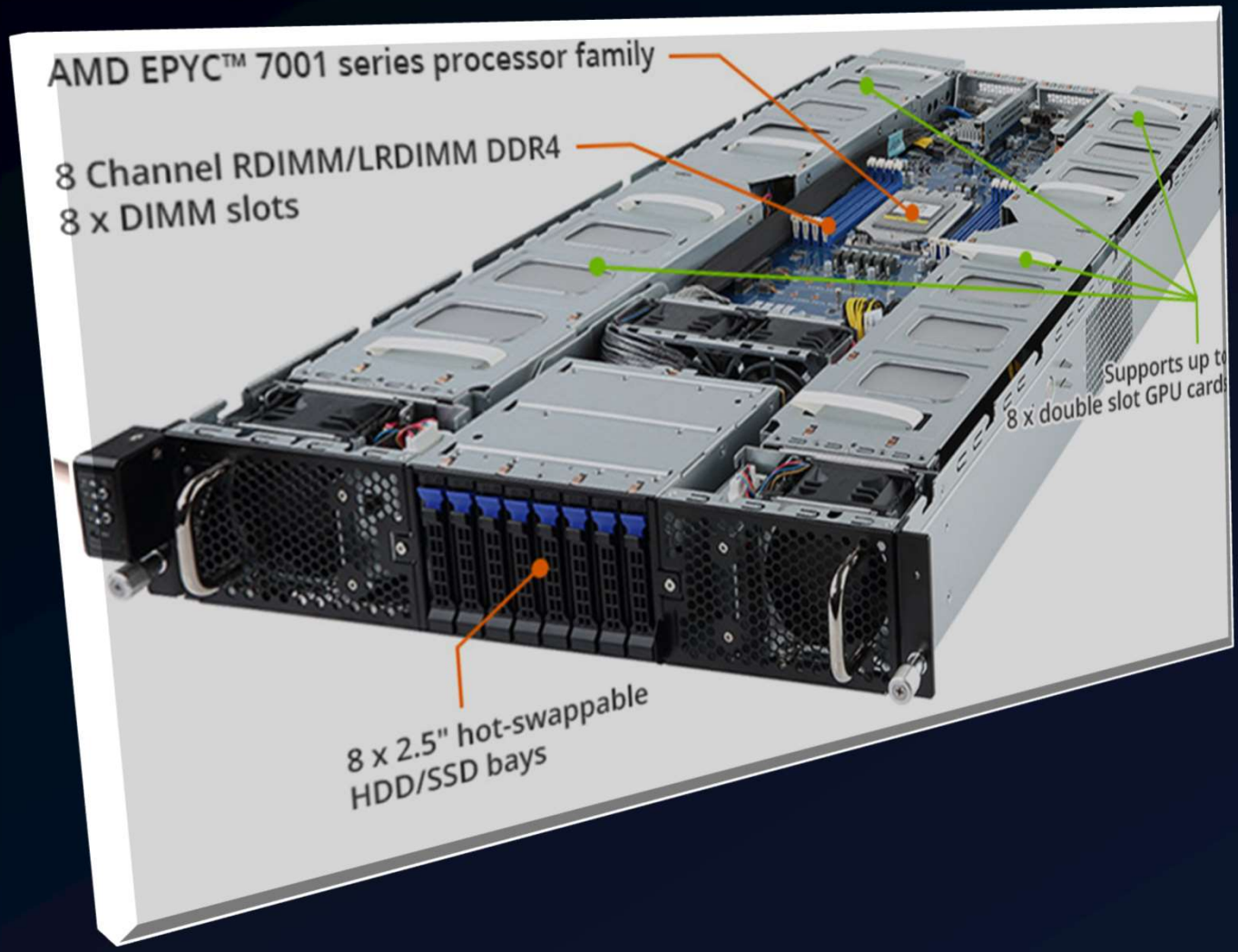
Research Question

Due to its crucial role in the SEV technology, targeting the AMD-SP instead of the protected VMs potentially allows an attacker to circumvent any protection guarantees of SEV, independent from the targeted VM!

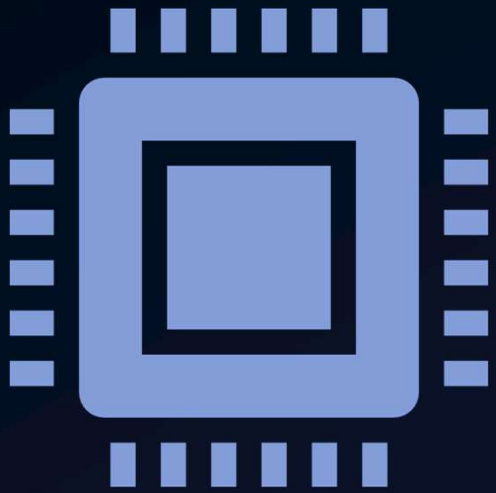
Consequently, in this work, we answer the following question:

What are the implications of fault injection attacks against the AMD-SP for the SEV technology?

THE TARGET MACHINE



FIRMWARE ANALYSIS



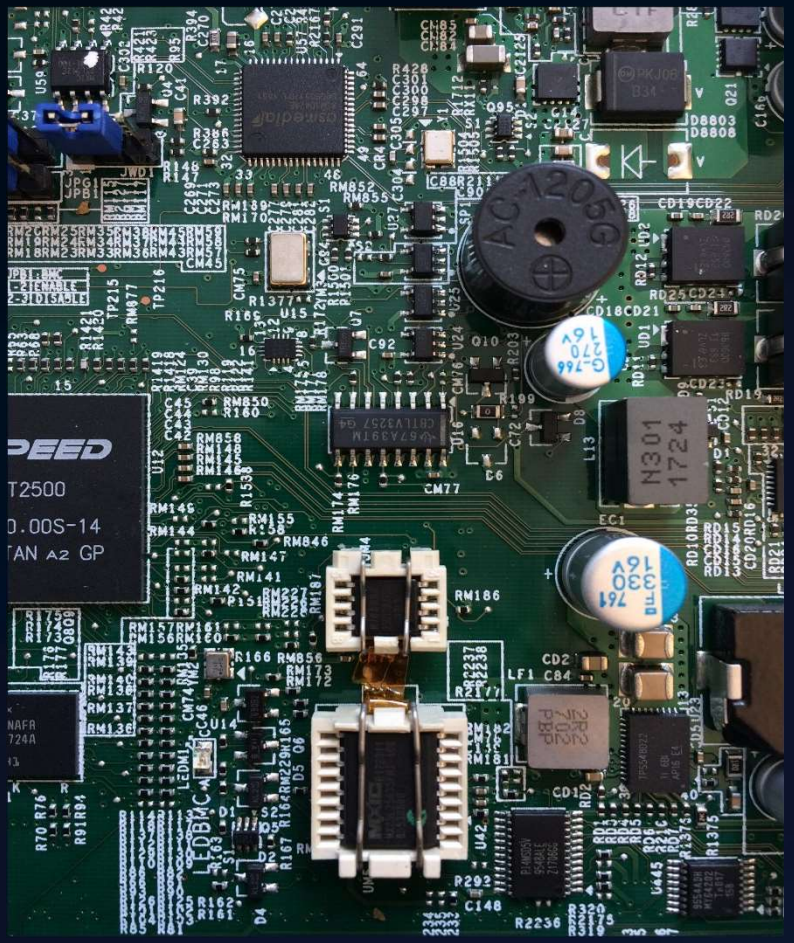
FIRMWARE ANALYSIS

Secure Processor is part of x86 die.

- ARM Cortex A5

Firmware is stored along UEFI FW!

Updatable through UEFI update.



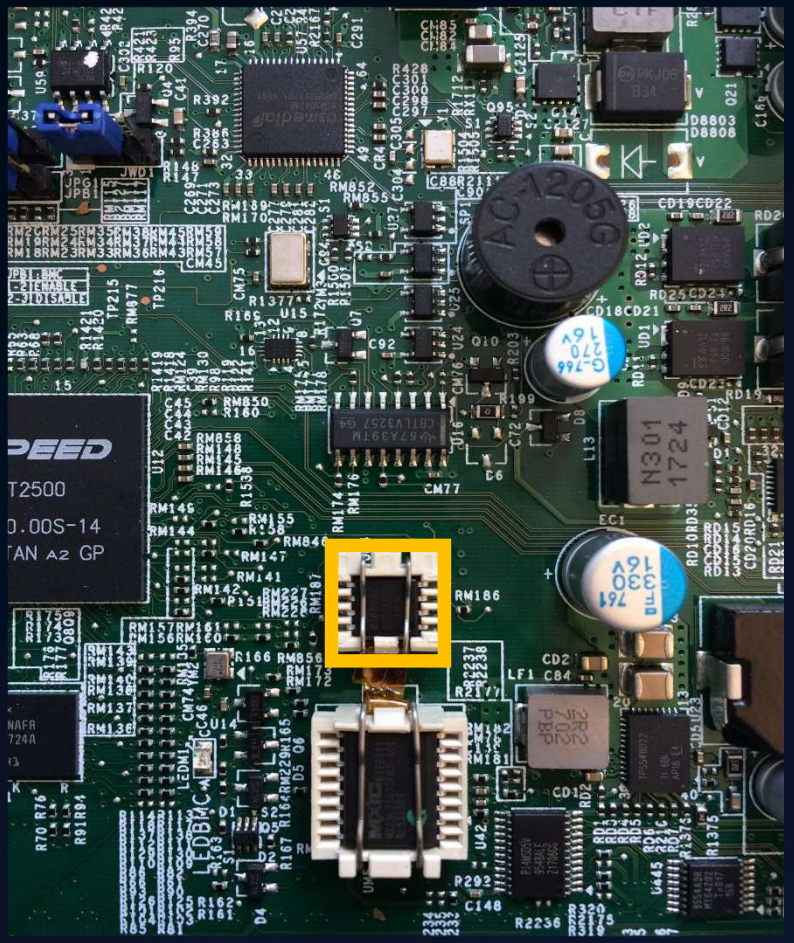
FIRMWARE ANALYSIS

Secure Processor is part of x86 die.

- ARM Cortex A5

Firmware is stored along UEFI FW!

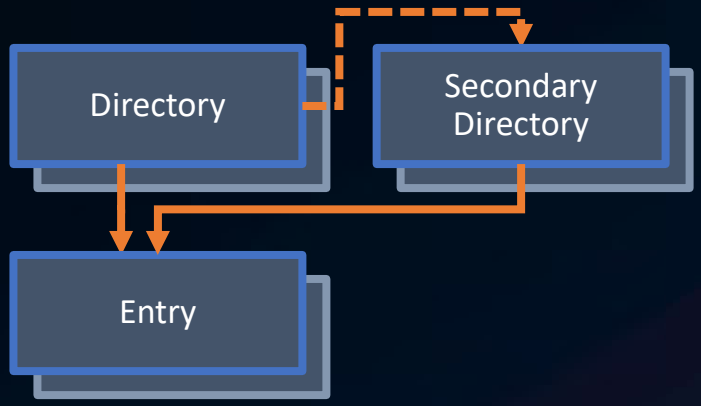
Updatable through UEFI update.

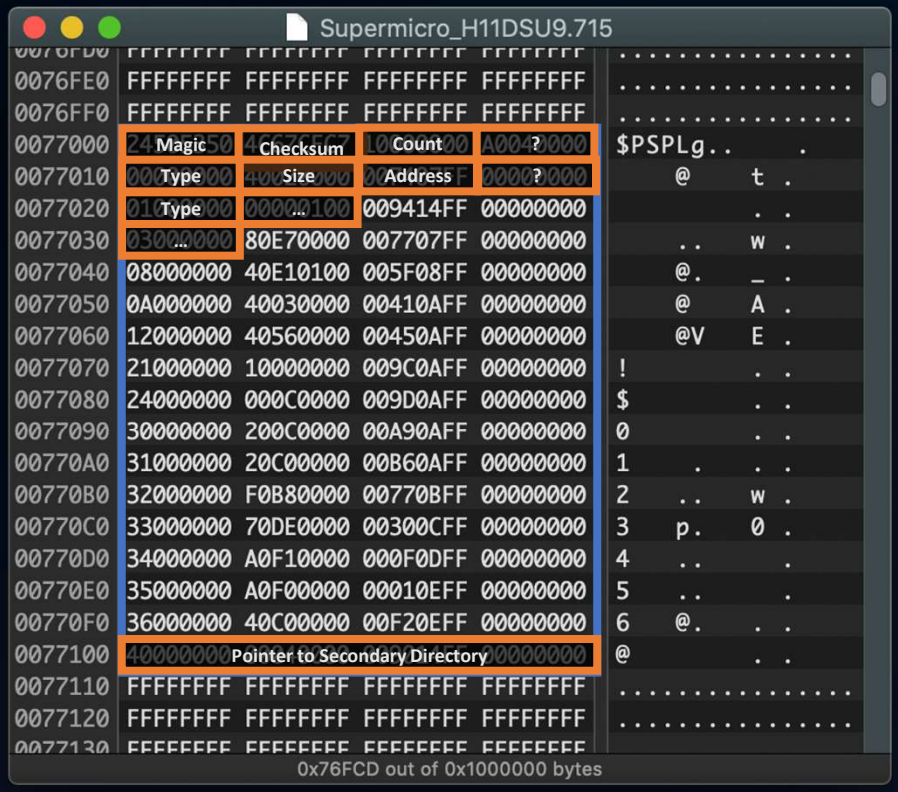


AMD SP Firmware

FIRMWARE FILE SYSTEM

- Contained within UEFI padding





https://github.com/ridiculousfish/HexFiend

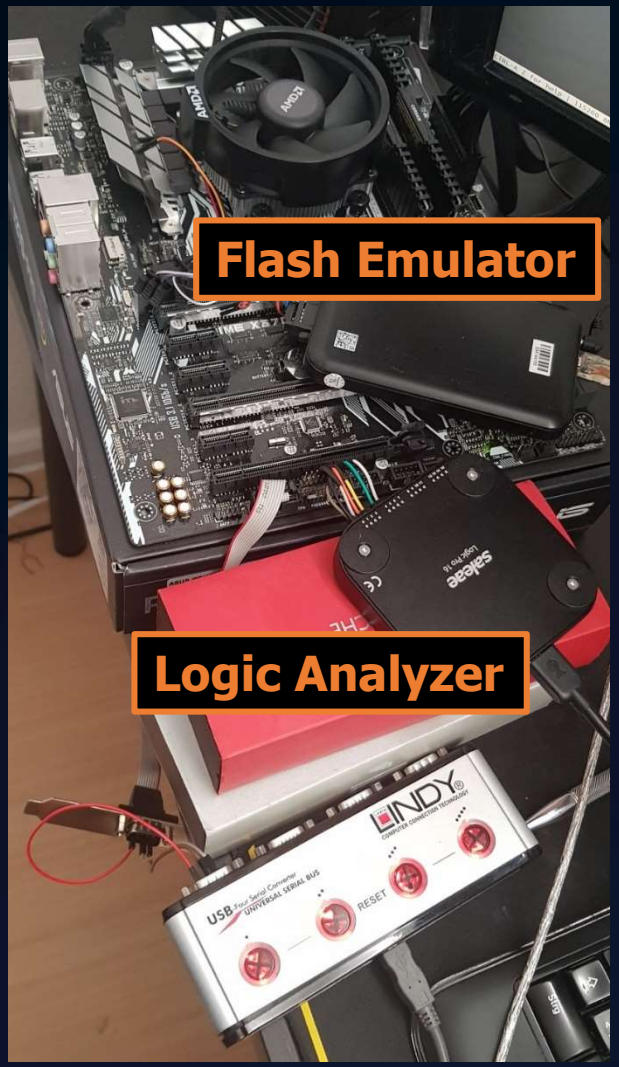
```
$ psptool uefi_image.bin
```

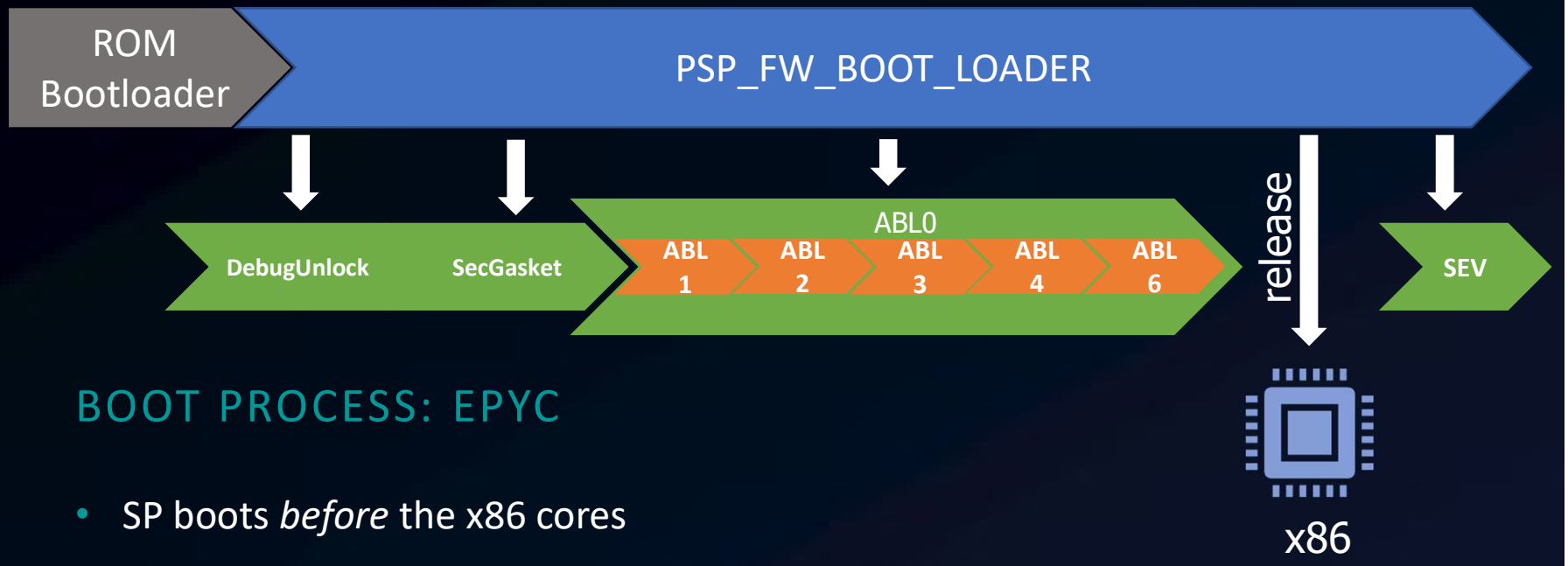
	Entry	Address	Size	Type	Type Name	Version	Signed by
	0	0xc2000	0x240	0x0	AMD_PUBLIC_KEY		
	1	0x281000	0x8000	0x1	PSP_FW_BOOT_LOADER	0.5.0.3B	AMD_PUBLIC_KEY
	2	0x289000	0x14000	0x8	SMU_OFFCHIP_FW	0.0.0.0	None
	3	0xc3000	0x6000	0x3	PSP_FW_RECOVERY_BOOT_LOADER	0.5.0.17	AMD_PUBLIC_KEY
	4	0xc9000	0x340	0x5	BIOS_PUBLIC_KEY		
	5	0xfff000	0x1000	0x6	BIOS_RTM_FIRMWARE		
	6	0x29d000	0x1e000	0x2	PSP_FW_TRUSTED_OS	0.5.0.3B	AMD_PUBLIC_KEY
	7	0xa0000	0x10000	0x4	PSP_NV_DATA		
	8	0x2bb000	0x14000	0x108	PSP_SMU_FN_FIRMWARE	0.0.0.0	None
	9	0xca000	0x340	0x9	AMD_SEC_DBG_PUBLIC_KEY		
	10	0x1	0xffffffff	0xb	AMD_SOFT_FUSE_CHAIN_01	E9.0.0.0	None
	11	0xcb000	0x340	0xd	PSP_BOOT_TIME_TRUSTLETS_KEY		

psptool: https://github.com/PSPReverse/psptool

psptrace: https://github.com/PSPReverse/PSPTool

No.	Address	Size	Type	Info
0	0xfa0000	0x18	!UEFI-IMAGE	[c]
3	0xc20000	0x10	Unknown area	[c]
5	0x020000	0x32	Firmware Entry Table	
6	0x0c0000	0x6a	Unknown area	CCP [c]
8	0x2a9000	0xfc	Directory: $PSP	CCP [c]
12	0x2a9400	0x440	AMD_PUBLIC_KEY	CCP [c]
			~ 60 µs delay ~	
29	0x798400	0xd0c0	PSP_FW_BOOT_LOADER	CCP [c]
			~ 2025 µs delay ~	
864	0x798000	0x100	!PL2_SECONDARY_DIRECTORY	CCP [c]





BOOT PROCESS: EPYC

- SP boots *before* the x86 cores
- **On**-Chip Bootloader loads **Off**-Chip bootloader from flash
- **Off**-Chip Bootloader loads and executes apps in specific order
- System is initialized by different **ABL stages**
- Load UEFI image and release x86 cores from reset
- **SEV app** is loaded during runtime upon the **request of the OS**

Agenda

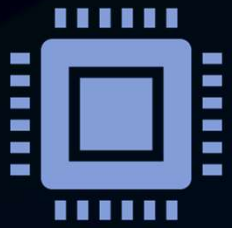
Amusing history

What is Zen/AMD SEV?

Research question

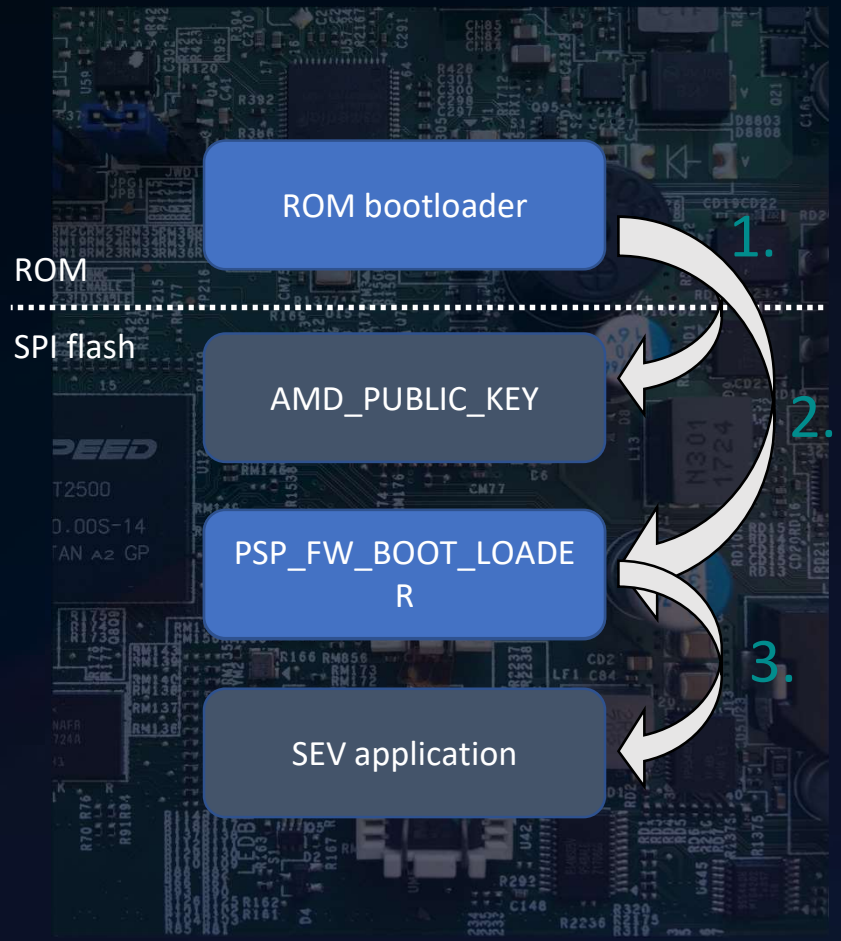
Glitching the AMD SP

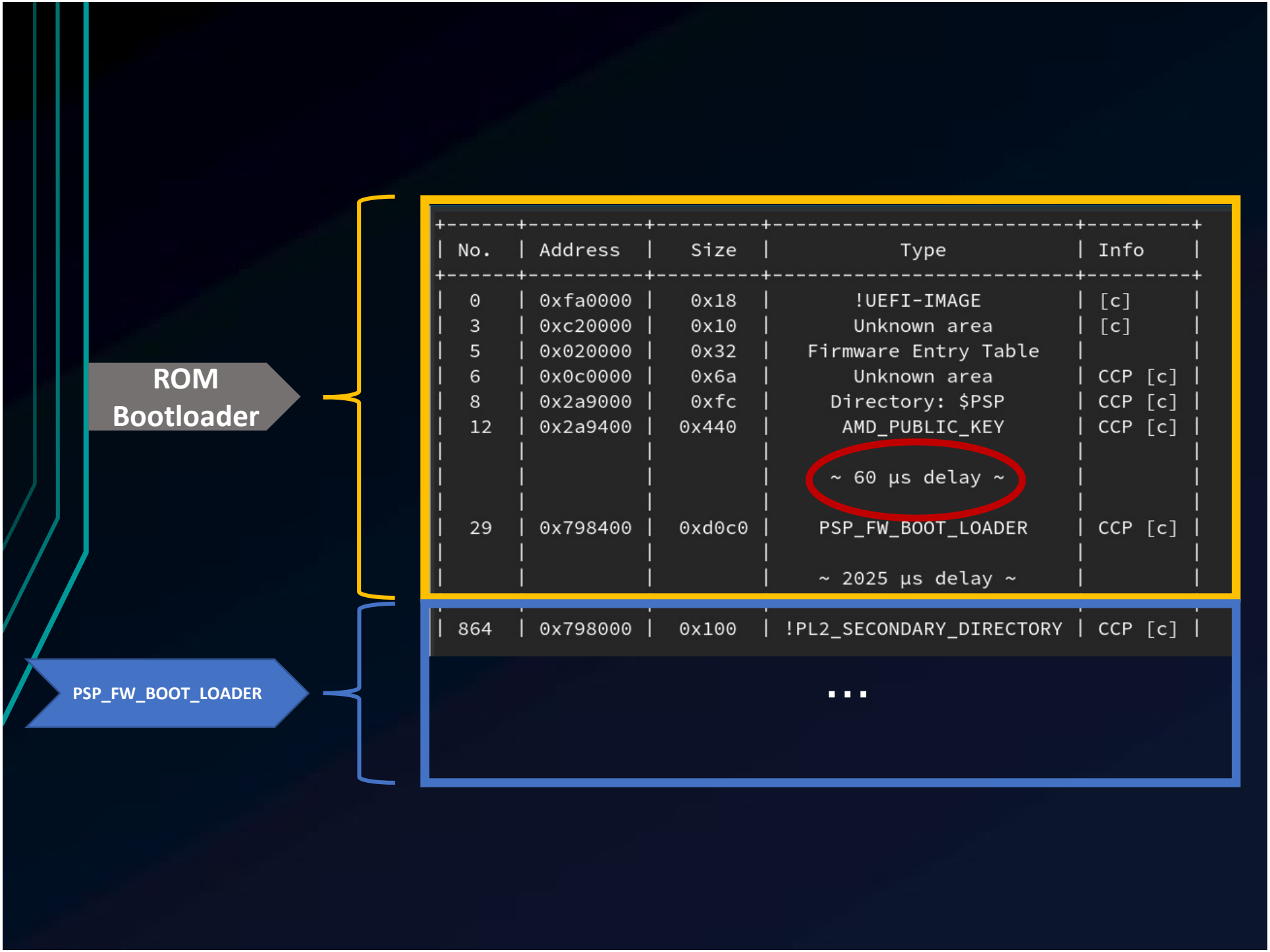
Conclusions



FIRMWARE ANALYSIS

1. Load & verify AMD_PUBLIC_KEY
   - verify using hash
2. Load & verify PSP_FW_BOOT_LOADER
   - verify using public key
3. Load & verify SEV application
   - verify using public key




ROM Bootloader
PSP_FW_BOOT_LOADER
No.	Address	Size	Type	Info
0	0xfa0000	0x18	!UEFI-IMAGE	[c]
3	0xc20000	0x10	Unknown area	[c]
5	0x020000	0x32	Firmware Entry Table	
6	0x0c0000	0x6a	Unknown area	CCP [c]
8	0x2a9000	0xfc	Directory: $PSP	CCP [c]
12	0x2a9400	0x440	AMD_PUBLIC_KEY	CCP [c]
~ 60 µs delay ~
| 29 | 0x798400 | 0xd0c0 | PSP_FW_BOOT_LOADER | CCP [c] |
~ 2025 µs delay ~
| 864 | 0x798000 | 0x100 | !PL2_SECONDARY_DIRECTORY | CCP [c] |
...

EXPERIMENTING A BIT …

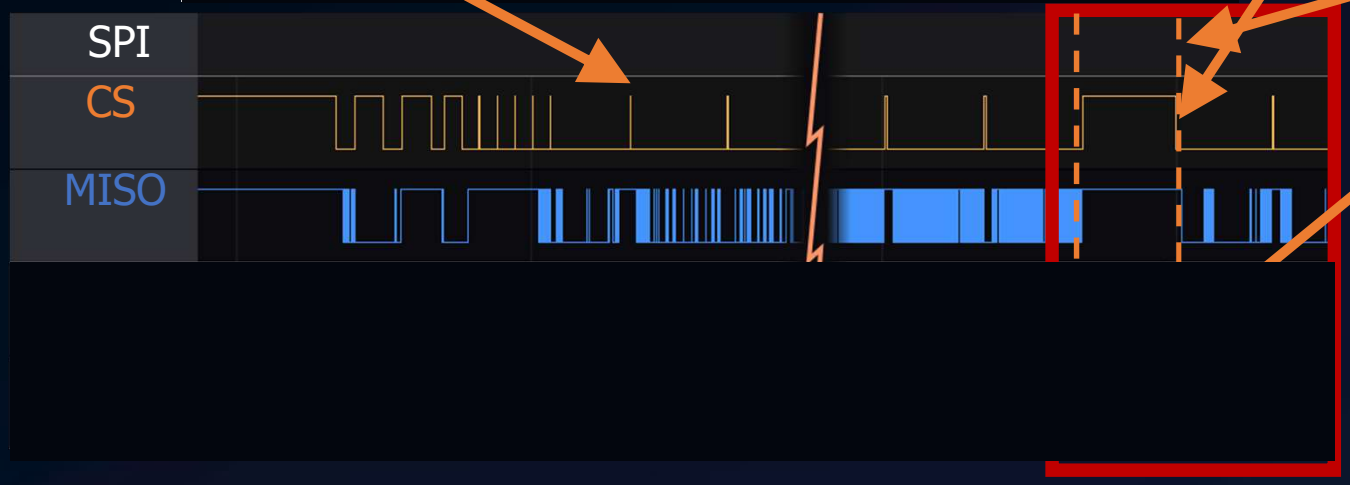
No.	Address	Size	Type	Info
0	0xfa0000	0x18	!UEFI-IMAGE	[c]
3	0xc20000	0x10	Unknown area	[c]
5	0x020000	0x32	Firmware Entry Table	
6	0x0c0000	0x6a	Unknown area	CCP [c]
8	0x2a9000	0xfc	Directory: $PSP	CCP [c]
12	0x2a9400	0x440	AMD_PUBLIC_KEY	CCP [c]

Constant # of level changes

Continued SPI activity

Key verification!

-> glitch "window"



Original pubkey

No SPI activity

Modified pubkey

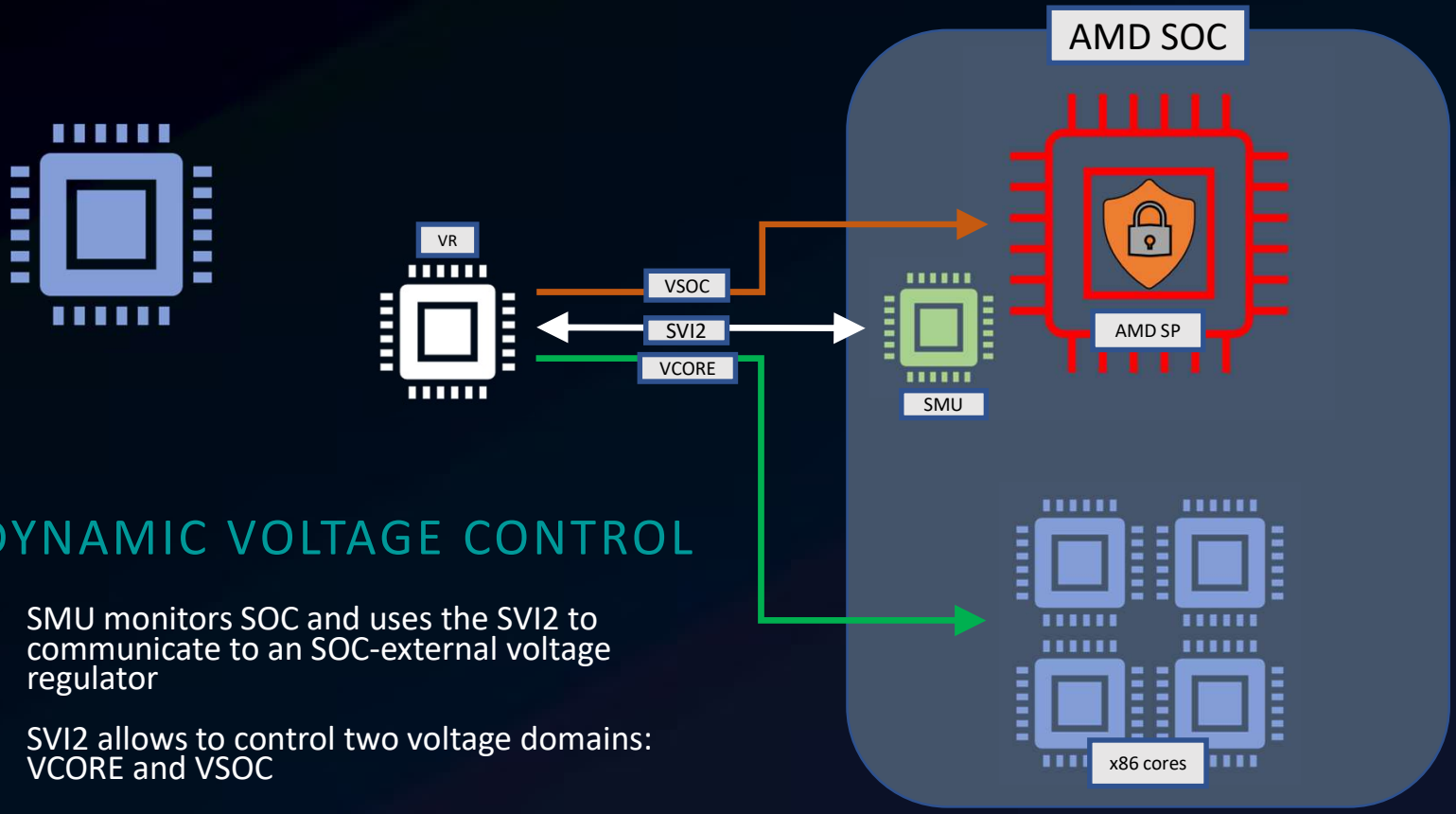
ATTACK OVERVIEW

Our goal is to execute our own payload right after the ROM bootloader.

1. Create custom public key
2. Replace AMD_PUBLIC_KEY in UEFI image
3. Replace PSP_FW_BOOT_LOADER component with payload
4. Sign payload with custom key
5. Glitch pubkey verification

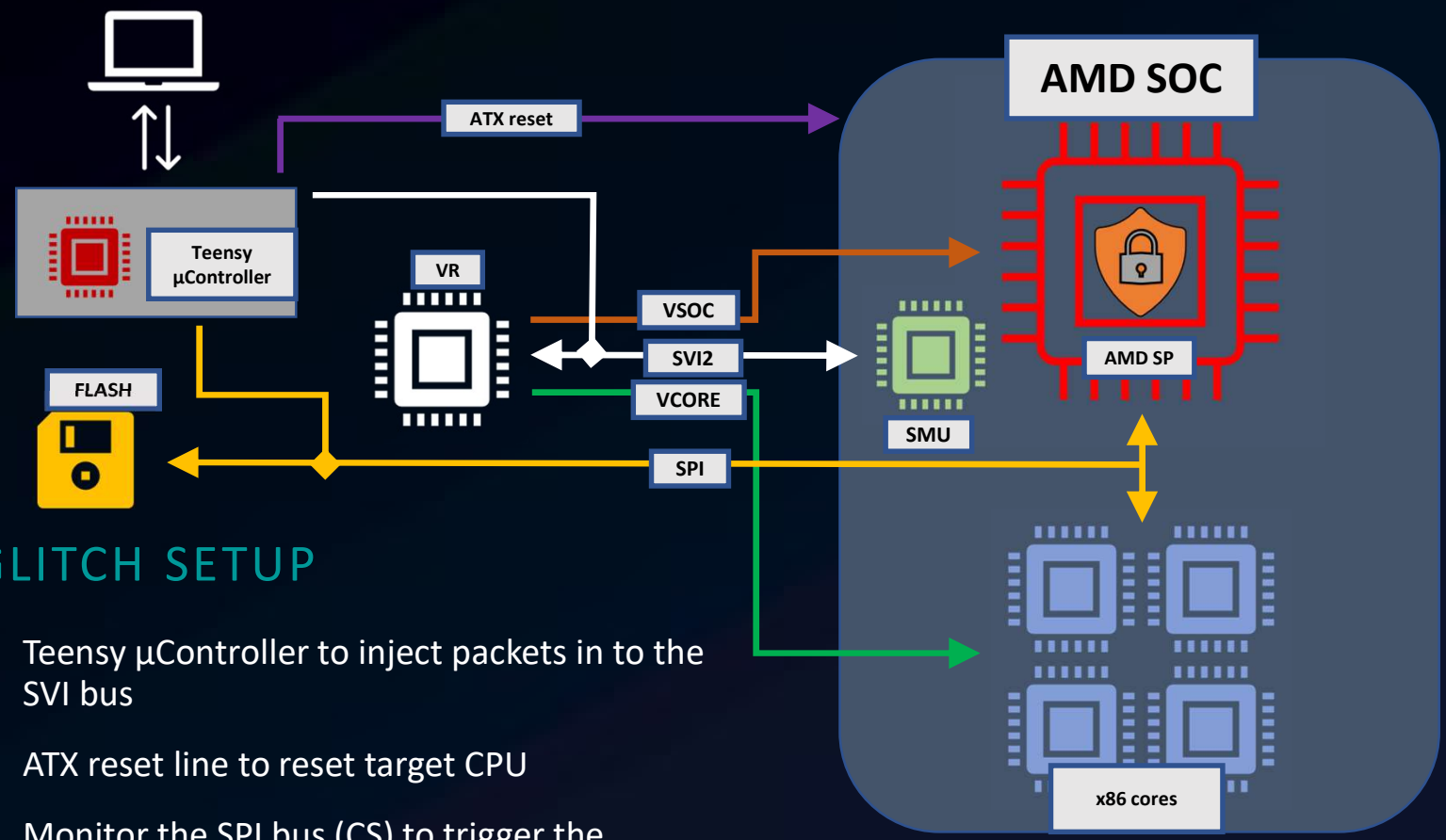
No.	Address	Size	Type	Info
0	0xfa0000	0x18	!UEFI-IMAGE	[c]
3	0xc20000	0x10	Unknown area	[c]
5	0x020000	0x32	Firmware Entry Table	
6	0x0c0000	0x6a	Unknown area	CCP [c]
8	0x2a9000	0xfc	Directory: $PSP	CCP [c]
12	0x2a9400	0x440	CUSTOM_KEY	CCP [c]
			~ 60 µs delay ~	
29	0x798400	0xd0c0	PAYLOAD	CCP [c]
			~ 2025 µs delay ~	
864	0x798000	0x100	!PL2_SECONDARY_DIRECTORY	CCP [c]

On authenticated computing and RSA-based authentication,
J.-P. Seifert, ACM CCS 2005.



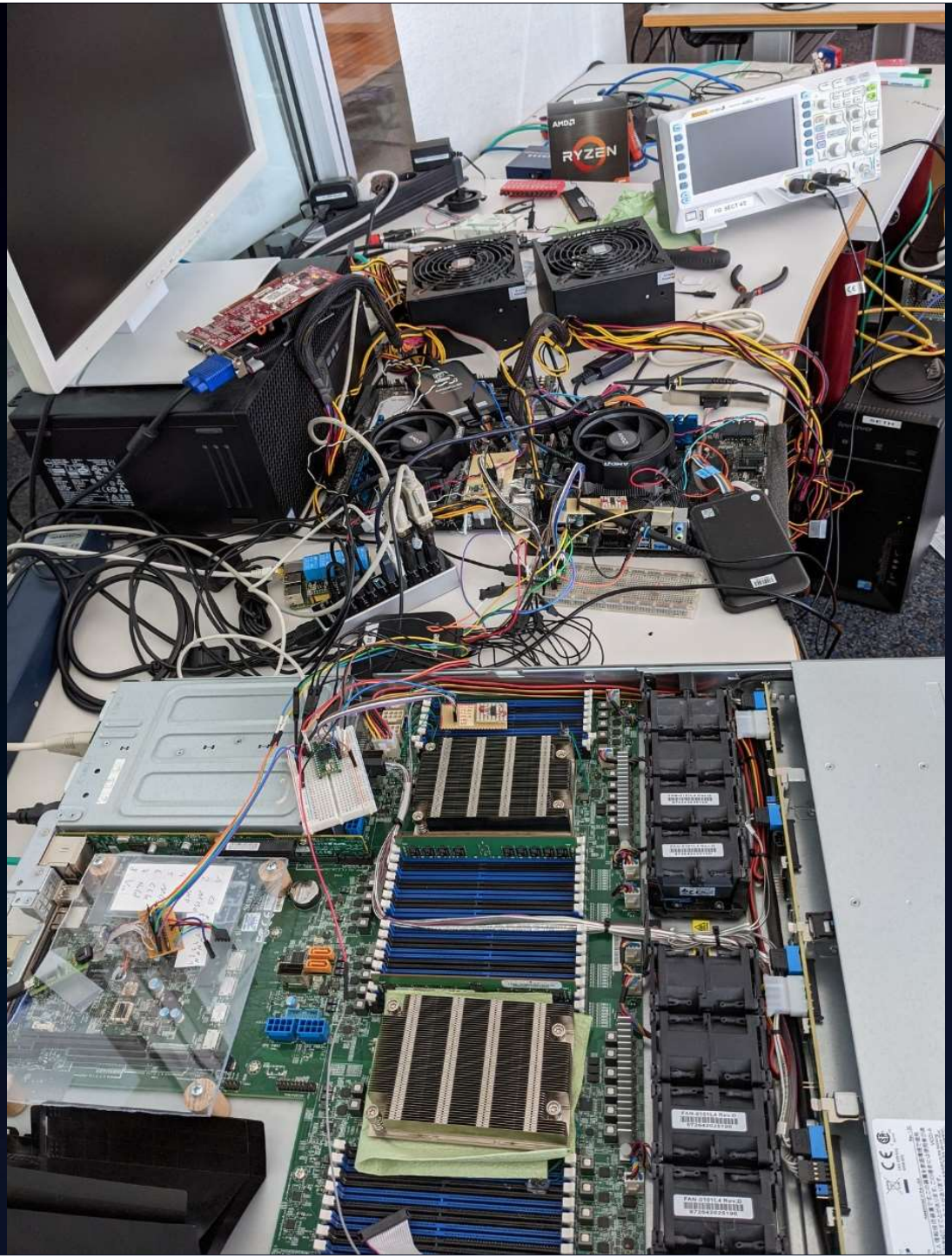
DYNAMIC VOLTAGE CONTROL

- SMU monitors SOC and uses the SVI2 to communicate to an SOC-external voltage regulator
- SVI2 allows to control two voltage domains: VCORE and VSOC
- Ryzen: AMD-SP uses VSOC
- Epyc: AMD-SP uses VCORE of a dedicated VR (two VRs per CPU)



GLITCH SETUP

- Teensy µController to inject packets in to the SVI bus
- ATX reset line to reset target CPU
- Monitor the SPI bus (CS) to trigger the voltage glitch
- Control glitch parameters via external PC

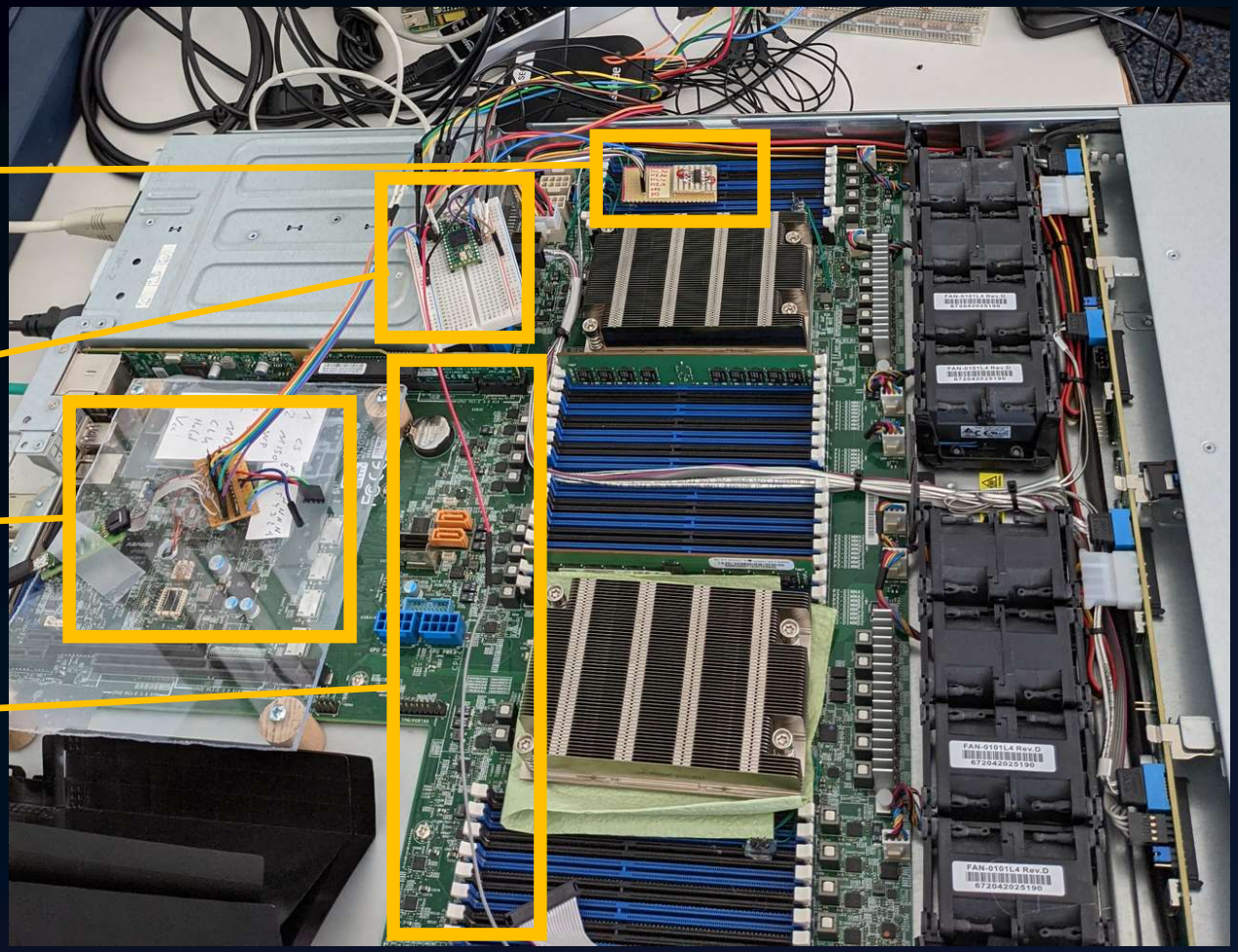

AMD
RYZEN

SVI2 access

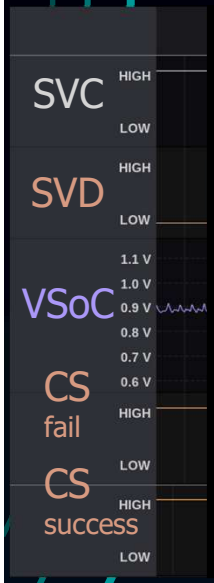
Teensy
µController

SPI access

ATX reset

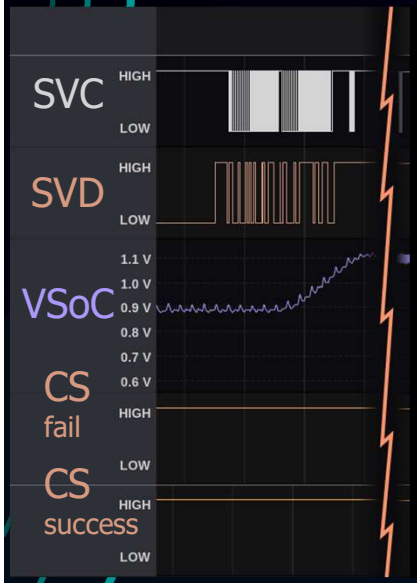


Glitch steps



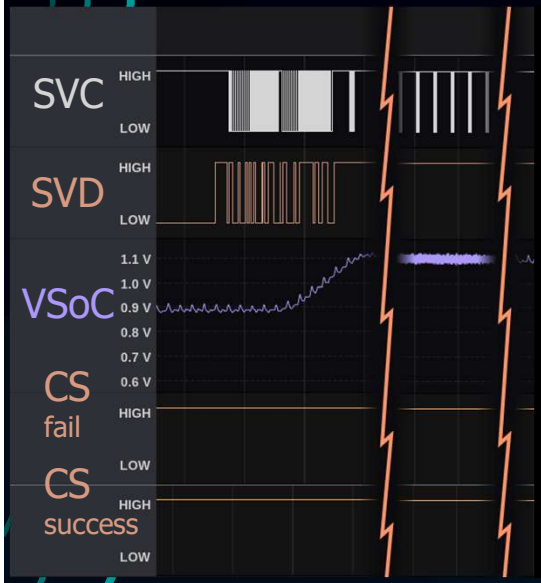
- SVI2 SVC – clock, CPU/VR (shared)
- SVI2 SVD – data from CPU, pulled low when inactive
- VSoC – target input voltage
- SPI CS – SPI's chip-select signal (successful/failed pubkey verification)

Glitch steps



- SVI2 SVD: becomes high -> start attack logic
- CPU initially configures voltage

Glitch steps

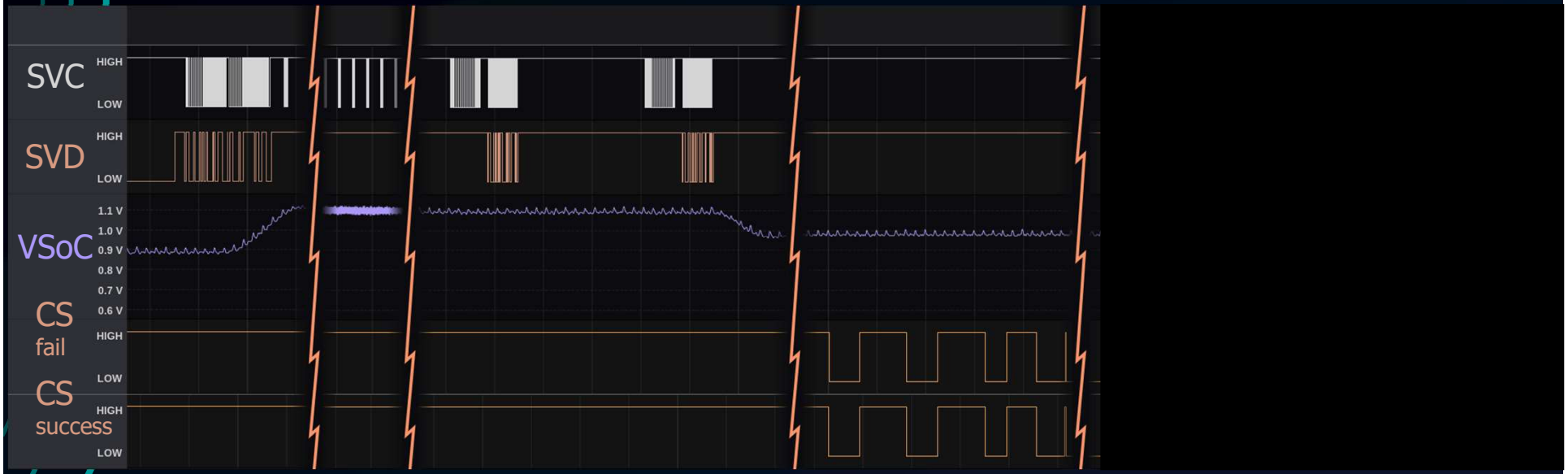


- SVI2 SVD: becomes high -> start attack logic
- CPU initially configures voltage
- VR constantly sends telemetry data to CPU

Glitch steps

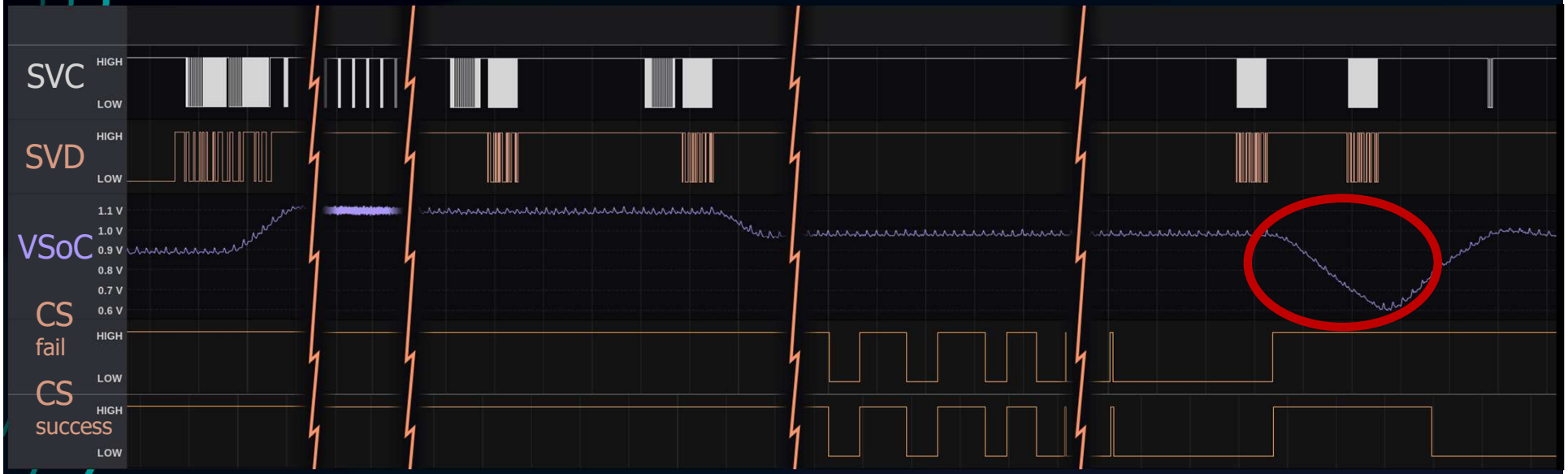
- SVI2 SVD: becomes high -> start attack logic
- CPU initially configures voltage
- VR constantly sends telemetry data to CPU
- Inject packets to disable telemetry -> avoids packet collision

Glitch steps



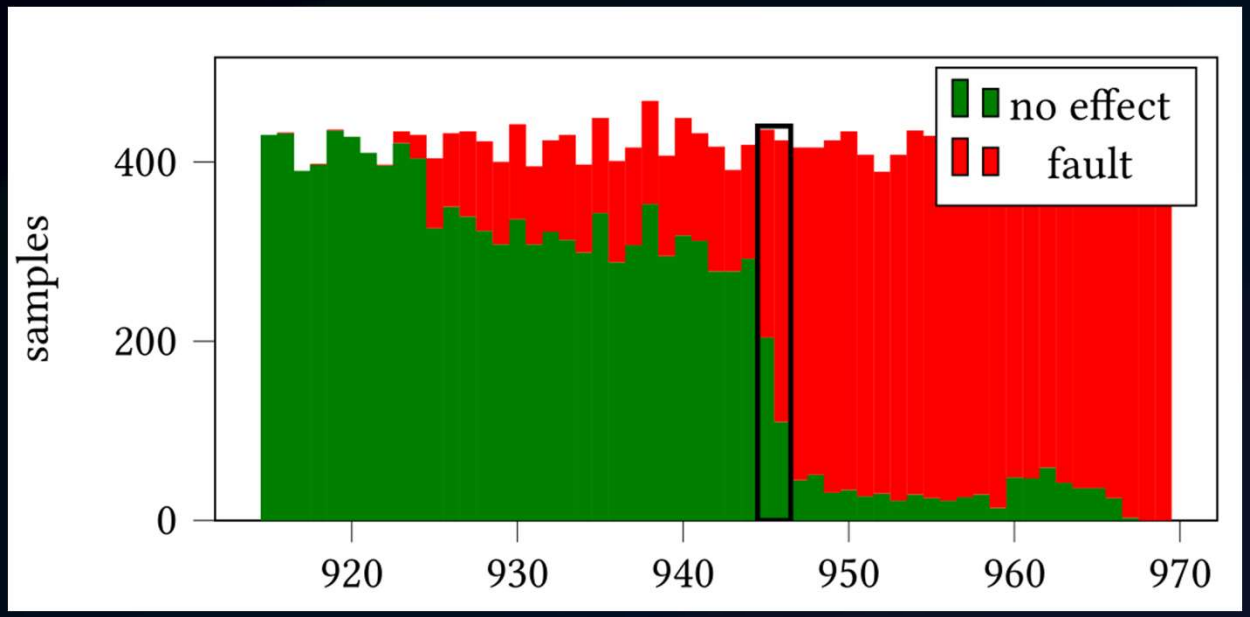
- Wait until SPI CS becomes active

Glitch steps

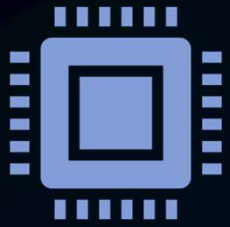


- Wait until SPI CS becomes active
- Count # of CS level changes to time glitch
- Inject packet to drop voltage and to revert to the original voltage level
- Verify success by observing CS again -> reset if CS not “low” after timeout
- Glitch duration window size

Duration window



- SPI image with original AMD public key
- Glitch introduced during the “glitch window”
- no effect: AMD-SP continued to boot -> CS is low after the “glitch window”
- fault: AMD-SP failed to boot -> CS is high after the “glitch window”

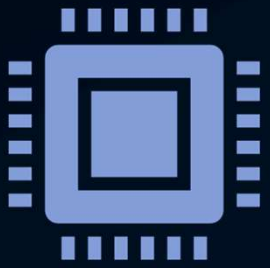


RESULTS

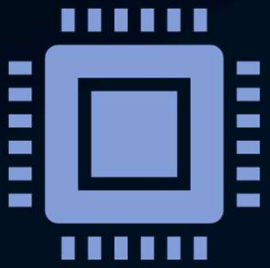
- Epyc and Ryzen CPUs are affected
- Successful glitch between every ~13min (Zen 1) and every ~46min (Zen 3)

Payloads:

- SPI “Hello World”
- Decrypt firmware (Zen 3)
- Dump ROM bootloader to SPI bus
- SEV-policy override (Zen 2):
  - Boot system with patched SEV firmware: Enables the “DBG_DECRPYT” SEV API command regardless of a guest’s SEV policy
- Dump (V)CEK secrets to the SPI bus



Reverse-engineering SEV's (V)CEK key derivation



Attacks

Attacker types

Malicious cloud administrator:

- Full hypervisor access
  - Send commands to the AMD-SP: SEV API
- Cloud management access
  - Able to install new systems in the datacenter
  - Able to migrate a VM to a different system

Malicious tenant + VM escape:

- Full hypervisor access
  - Send commands to the AMD-SP: SEV API

Agenda

Amusing history

What is Zen/AMD SEV?

Research question

Glitching the AMD SP

Conclusions

RESOURCES

https://github.com/RobertBuhren/amd-sev-migration-attack

- Proof-of-concept implementation of the migration attack.

https://github.com/RobertBuhren/Insecure-Until-Proven-Updated-Analyzing-AMD-SEV-s-Remote-Attestation

- Proof-of-concept signature created with an extracted CEK.

https://github.com/PSPReverse/PSPTool

- psptool & psptrace

https://lsseu2019.sched.com/event/TynP/upcoming-x86-technologies-for-malicious-hypervisor-protection-david-kaplan-amd

- AMD SEV-SNP Talk at the Linux Security Summit 2019.

One Glitch to Rule Them All: Fault Injection Attacks Against AMD's Secure Encrypted Virtualization,
R. Buhren, H. N. Jacob, T. Krachenfels, J.-P. Seifert, ACM CCS 2021.

Insecure Until Proven Updated: Analyzing AMD SEV's Remote Attestation,
R. Buhren, J.-P. Seifert, Christian Werling, ACM CCS 2019.

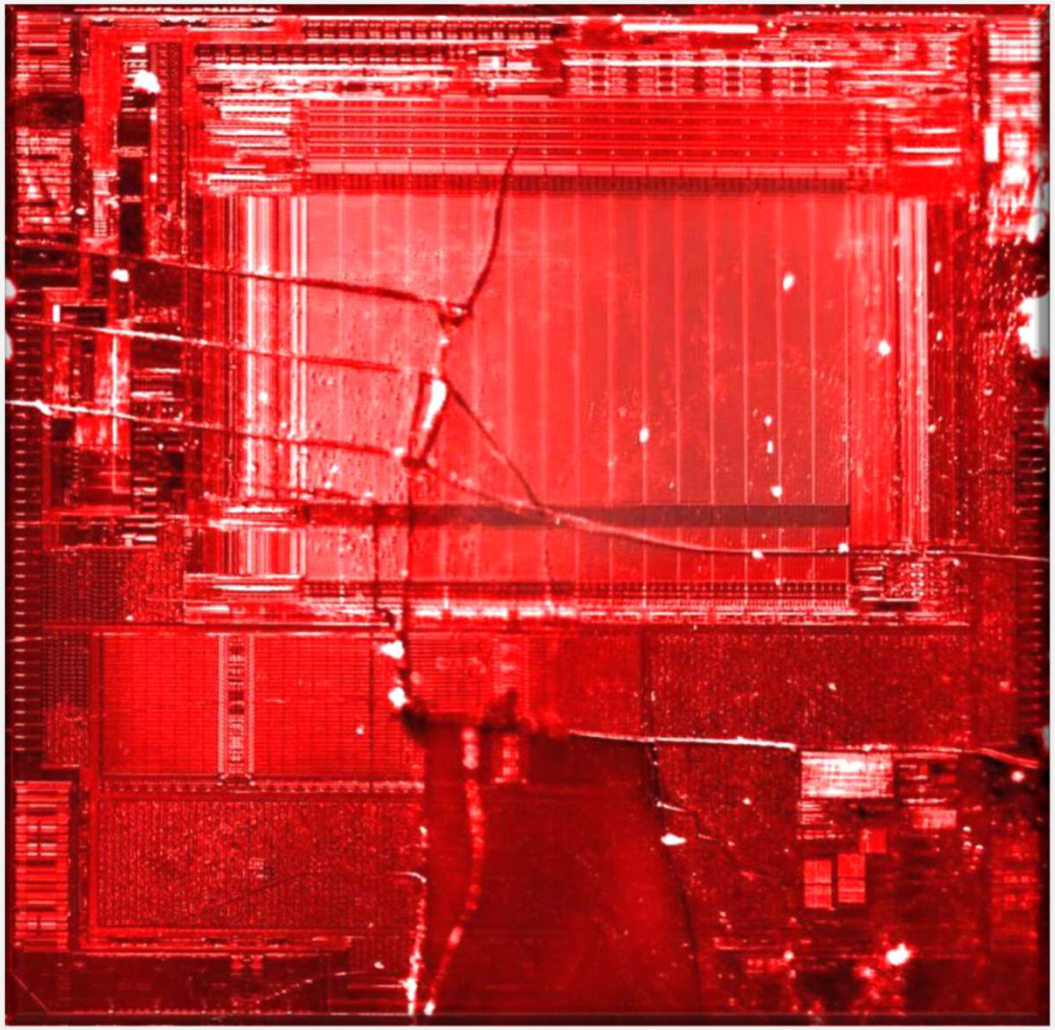
Thank you for your attention!

This work has been supported by:

DFG Deutsche Forschungsgemeinschaft

EINSTEIN Foundation.de

Questions?



EINSTEIN Foundation.de

SECURITY IN TELECOMMUNICATIONS
TECHNISCHE UNIVERSITÄT BERLIN