# blackhat ASIA 2024

# APRIL 18-19, 2024

BRIEFINGS

# Faults In Our Bus: Novel Bus Fault Attacks to Break ARM TrustZone



Nimish Mishra, Anirban Chakraborty, Debdeep Mukhopadhyay Indian Institute of Technology Kharagpur, India





# Who are we?



Nimish Mishra



Anirban Chakraborty



Indian Institute of Technology Kharagpur India



### Debdeep Mukhopadhyay





- 1. What are Faults?
- 2. Traditional Fault Points on Embedded Systems and SoCs
- 3. A (new) Fault Point on SoCs
- 4. OP-TEE?
- 5. End-to-end Attack
  - Load (adversarial) Trusted Application through Faults
  - Redirect communication for other Trusted Applications
  - Decrypt (redirected) communication
- 6. Impact





# What are Faults?





Actively perturb data or control-flow of a system and gain ۲ information about the secret through faulty system response



Correct Output



- Fault causes error and error can be exploited to leak secret information
- Fault attack sometimes combined with side channel can lead to stronger attacks







Fault Injection

Side Channel Observation









# **Fault Attack Vectors**



Fig: Electromagnetic Fault Injection (EMFI) Probe

**WHAT:** Strategically modify execution environment of a system





Top view of electromagnetic fault injection loop



HOW: Through changes in external operational conditions



Side-view of electromagnetic fault injection loop



### Electric current / Electric field

### Magnetic field perpendicular to the direction of electric field

### Fig: Working principle of EMFI Probe



# **FI Attack Vectors**

• WHAT: Strategically modify execution environment of a system

HOW: Through changes in external operational conditions

**WHY:** Bias software execution to adversarial advantage







## Granularity

- 1. Single bit
- 2. Multiple bits
- 3. Byte or Word





## Granularity

- 1. Single bit
- 2. Multiple bits
- 3. Byte or Word

# Fault-type

- 1. Stuck-at (zero or one)
- 2. Bit flip
- 3. Random





## Granularity

- 1. Single bit
- 2. Multiple bits
- 3. Byte or Word

## **Attacker Control**

- 1. Precise
- 2. Loose
- 3. None

# Fault-type

- 1. Stuck-at (zero or one)
- 2. Bit flip
- 3. Random





## Granularity

- 1. Single bit
- Multiple bits 2.
- Byte or Word 3.

### **Attacker Control**

- 1. Precise
- 2. Loose
- 3. None

## Fault-type

- 1. Stuck-at (zero or one)
- 2. Bit flip
- 3. Random

## Duration of the fault

- 1. Transient
- 2. Permanent
- 3. Persistent





# **Traditional Fault Points**





# 













External interface (voltage/clock glitch)



Dynamic Frequency and Voltage Scaling (DVFS)











External interface (voltage/clock glitch)



Dynamic Frequency and Voltage Scaling (DVFS)

### Rowhammer



# Laser/EM Fault injection









No external interface (in SoCs; ex RPi)



Privileged



### Rowhammer



# Laser/EM Fault injection









No external interface (in SoCs; ex RPi)



Privileged



### ECC checks



# Casings (requires invasive depackaging)



# Are there other architectural aspects which can be used for faults,

for which no known defences are deployed yet?





# A (new) Fault Point on SoCs









ECC checks



# Casings (requires invasive depackaging)







# (requires invasive



- Uncased and exposed
- Involved mainly with **load/store** instructions
- Prior works
  - Simulation of bus faults
  - External voltage glitches on PlayStation consoles to skip memory cycles



Fig: Exposed bus connections in RPi3





Zoomed in view. The exposed system bus between the processor and memory



### **load** dest\_reg, [mem\_addr]



Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3







Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3







Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3







Fig: Electromagnetic Fault Injection probe positioned over the exposed system bus on a RPi3





















# Address Bus Faults

Success rate breakdown



# **Implication**: Register sweeping to mount an end-to-end attack on Open Portable Trusted Execution Environment (OP-TEE)











# **"Trusted" Execution Environment**

- WHAT: An attempt to **disentangle** critical applications from generic software (including kernel)
- HOW: (Hardware backed) isolation of system resources
- OP-TEE: Implementation of GlobalPlatformAPI specification for ARM TZ

 Maintained by the Trusted Firmware, with members like Google, ARM, Linaro, NXP, STMicroelectronics

 $\odot \mbox{Deployed}$  in commercial platforms like Apertis, iWave, and so on





# **"Trusted" Execution Environment**

• Two main divisions

### 1. TEE or Trusted Execution Environment

Execution context where all the security critical operations reside. TEE has its own

- a) secure/encrypted memory storage,
- b) secure I/O peripherals,
- c) secure context switching




## **"Trusted" Execution Environment**

Two main divisions

### 1. TEE or Trusted Execution Environment

Execution context where all the security critical operations reside. TEE has its own

- a) secure/encrypted memory storage,
- b) secure I/O peripherals,
- c) secure context switching

### 2. REE or Rich Execution Environment

Execution context where rest of the things run. REE invokes the services of TEE when required.

Secure world (Trusted Execut Environment or	i tion FEE)
TEE Userspace EL 0	
1	
i	
· :=1/-	
Trusted applicatio	~
i irusted applicatio	n
i /	
· 11	
TEE	
الألي	
• []	
!	
TEE ADI	
I IEE API	
1	
1	
1	





### **"Trusted" Execution Environment**

- Two main divisions
  - 1. TEE or Trusted Execution Environment
  - 2. REE or Rich Execution Environment

**Note:** <u>All Trusted Applications (TAs) running in the TEE are</u> <u>checked for integrity, implying no adversary having</u> <u>complete control over REE can execute arbitrary TEE code</u>.





### **"Trusted" Execution Environment**

- Two main divisions
  - 1. TEE or Trusted Execution Environment
  - 2. REE or Rich Execution Environment

### ADVERSARIAL GOAL!

**Note:** <u>All Trusted Applications (TAs) running in the TEE are</u> <u>checked for integrity, implying no adversary having</u> <u>complete control over REE can execute arbitrary TEE code</u>.

Secure world (Trusted Execution Environment or TEE)
TEE Userspace EL 0
1
1
1
1
· IIIK
· · · · · ·
Trusted application
: 42
i
ITEE
Userspace $\checkmark$
TEE API
1
1
1





• Goal 1 : Entire attack must be online (without taking the device offline)





- Goal 1 : Entire attack must be online (without taking the device offline)
  - Challenge 1 : Secure Boot cannot be attacked (requires taking the device offline)

(Our) Solution: Attack the loading of Trusted Applications in the TEE





- Goal 1 : Entire attack must be online (without taking the device offline)
  - Challenge 1 : Secure Boot cannot be attacked (requires taking the device offline)

(Our) **Solution**: Attack the loading of Trusted Applications in the TEE

• Challenge 2: Cannot use code-based triggers (requires code modifications to the OP-TEE kernel) (Our) **Solution**: Construct a combined adversary (side-channel analysis + fault injection)



Goal 2 : The attack must be non-invasive





Goal 2: The attack must be non-invasive

• Challenge 3 : Cannot inject processor faults (requires depackaging). Trivial attacks like instruction skips cannot work

(Our) Solution: Work with a new fault model (register sweeping) on the system-bus (requires no invasive alterations to the target device)





### **Fault Attack Target**









(Exception Layer 2)







External glitch DVFS

Rowhammer

Stealing signing key







Not Available Not Available

Protected TA

Signing key not stored on device









faults







**Register Sweeping**: Fault the load to 0x0 through data bus faults





### **Fault Attack Results**

• No Effect (denoted by a "dot"): No effect of the injected fault

 Partial Success : Injected fault changes the value of the load, but Or causes SEGFAULT

• Success : Faults value of the load to 0x0.

	[PARTIAL	SUCCESS1 F	Registe	er value co	orru	pted to 0x10	73c
	Segmentat	ion fault	(core	dumped)			No s receiper - Chill
	Illeg	al instruc	tion (	(core dumpe	ed)		
	Illegal i	nstruction	(core	e dumped)			
					· · · ·	Illegal in	structio
	I	llegal ins	tructi	ion (core d	dump	ed)	
	Illega	l instruct	ion (c	core dumped	( t		
		[Sl	ICCESS	Register	val	ue corrupted	to 0x0
	[SUCCESS]	Register	value	corrupted	to	0×0	1
	[SUCCESS]	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0×0	-
	[SUCCESS]	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0×0	1
	[SUCCESS]	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0×0	1
	[SUCCESS]	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0.0	
	[SUCCESS]	Register	value	corrupted	to	0X0	
	[SUCCESS]	Register	value	corrupted	10	0x0	
		Register	value	corrupted	10	0x0	
F		Register	value	corrupted	10	0.00	
	[SUCCESS]	Pegister	value	corrupted	10		
	[SUCCESS]	Register	value	corrupted	10	0×0	
	[SIICCESS]	Register	value	corrupted	to	0x0	
	[SIICCESS]	Register	value	corrupted	to	0x0	
	[SUCCESS]	Register	value	corrupted	to	0×0	
	SUCCESSI	Register	value	corrupted	to	0×0	
	SUCCESSI	Register	value	corrupted	to	0×0	
	ISUCCESSI	Register	value	corrupted	to	0×0	
	SUCCESSI	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0×0	
	[SUCCESS]	Register	value	corrupted	to	0×0	





# **End to End Attack**

Load (adversarial) Trusted Applications through Faults

**Redirect communication for other Trusted Applications** 

**Decrypt (redirected) communication** 





# End to End Attack

Load (adversarial) Trusted Applications through Faults







Power side-channel to inform fault injection in a **non-invasive** way (no recompilation of OP-TEE necessary)







Power side-channel to inform fault injection in a **non-invasive** way (no recompilation of OP-TEE necessary)



Actual Fault Injection on signature verification

		25	OM	S/S	÷	<u> </u>		4	5		-	8	2 V
na dana katangan kana kana kana kana kana kana kana				e-1100.12				13					
cted onto the ta	arge	et o	dev	vice.		25	15	25		ALMA	10		
ator gets trigge	red		F 13					1			1		- 
riggered ection trigger		a a	4 (4			ся	19	19		1		10	a a a
		9 B	1 12	æ		12	58	35				3	
pens here		8 B	8 93	94	-	ΞĨ.	24	331	1			56	983















Fallout: Register sweeping fault attack loads a self-signed, adversarial controlled Trusted Application in the secure world of OP-TEE





# Fallout: Register sweeping fault attack le Escalation 1 Application in The Escalation of OP-TEE





# End to End Attack Redirect communication for other Trusted Applications









Insecure World



Secure World



Universally Unique **ID**entifier (UUID) comparison

...





### Secure Trusted Application execution



**Our Findings:** GlobalPlatform API specification (upon which OP-TEE is constructed) **offloads** the responsibility of choosing UUID to **Original Equipment Manufacturer**. It is the responsibility of the OEM to ensure **no two Trusted Applications (TA) share same UUID**.





**Our Findings:** GlobalPlatform API specification (upon which OP-TEE is constructed) **offloads** the responsibility of choosing UUID to **Original Equipment Manufacturer**. It is the responsibility of the OEM to ensure **no two Trusted Applications (TA) share same UUID**.

**UUID confusion:** Behaviour of the system when **UUID are non-unique is undefined**. Our empirical conclusion is that, when UUIDs are shared, a **non-persistent TA is preferred over persistent TA**.











**Insecure World** 

Secure World



╳

Universally Unique **ID**entifier (UUID) comparison (with self-signed TA loaded after register sweeping attack)







Secure Trusted Application execution (persistent TA)

### Self-signed Trusted Application execution (**non-persistent TA** with UUID confusion)



# End to End Attack Decrypt (redirected) communication





### **Decrypt (redirected) communication**

### Third Party extension: SeCReT

- Symmetric key management
- Blocks SIGTRAP
- Blocks unauthorized read to sensitive data pages







## **Decrypt (redirected) communication**



### Third Party extension: SeCReT

- Symmetric key management
- Blocks SIGTRAP
- Blocks unauthorized read to sensitive data pages







# **Decrypt (redirected) communication**



### Third Party extension: SeCReT

- Symmetric key management
- Blocks SIGTRAP
- Blocks unauthorized read to sensitive data pages
- Does not block SIGSEGV. Leaks key through coredumps







(adh) ht

# **Decrypt (redirected) communication**

#4 0x00010708 in main () (odb)	Leaked secret key
2?1\216\330\$+\241\204dV\211\321\f\266\340\004Z\304\035F	{\226\371D?;\030\266hT\331A2\237\211\267v\020\231Q\033\067N\233\002\022") at
sk=0x107d6b8 "\a2TL\254\330,\354\245\177v\233\351C\	266\b\b\274=\261\177\003?\231mt2_^E\025?\262\250\032\344\377{n_m\274\021\320
#3 0x00010904 in POCLEAN DILITHIUM2 CLEAN crypto sign	(sm=0x107e790 "", smlen=0xbefbd420, m=0x18950 "This is a very random message
2\277`\272?1\216\330\$+\241 Target function	26\371D?;\030\266hT\331A2\237\211\267v\020\2310\033\067N\233\002
sk=sk@entry=0x107d6b8 Target function	b\b\274=\261\177\003?\231mb2_^E\025?\262\250\032\344\377{nmm\27
<pre>m=m@entry=0x107f104 "This is a very random message"</pre>	, mlen=mlen@entry=30.
#2 0x00010afc in POCLEAN DILITHIUM2 CLEAN crypto sign	signature (sig=sig@entry=0x107e790 "" siglen=0x0 siglen@entry=0xbefbd420
te-tegentry-experiorzo (SSIA2(25)(211(20)V(025)	1202(230(032(344(37))1)))) (274(021(3200(274(327(374(0(324(334(032(277((272(1(2ofh13c0 c)-0)))))))))))))))))))))))))))))))))
kou-kou@ent.ru-@vhofh@f20 "T\ 33142\ 237\ 211\ 267u\ 0252	262\ 250\ 622\ 344\ 277 [p_m) 274\ 621\ 2261\ 274\ 227\ 274\ v\ 224\ 254\ 622\ 277`\ 27221\ 2
1=L1@entry=0x0e100100 m62+ C+\241\2040V\211\321\1	\200\340\0042\304\033F{\220\371Df;\030\20001\331A2\237\211\207V\025f\202\230
#1 0X00011520 IN PULLEAN_DILITHIUM2_LLEAN_UNPACK_SK (F	no=rno@entry=0xberb0ee0 ,
\030\314\0/11\303\250\031\023Y\334\300\000\264\305(]\34	5\215\350\0/1\a\377\000?\370\a\235(\D11Q\004\264") at poly.c:694
\227\320\331\t2\241J\236\215B\265\\\t\254L\020\305\335\	344J\223\350\310\n\0010\023\272G\237\035\223\230\t(4z5\226\225\344\265*\326(
\234\340N\240\250\313`\036\2010!\307_\340\347\322\376(\	241u\361e\037\071\277-}\031\240\177.\242]v\177n\267!oN\025\062\261\370F\353\
a@entry=0x107da18 "\250\322b\017\241\377/\366\201\0	25\273M\373\265J2`\022\342\002\r\246\376\351Q\235\002\257\305\001\n\237\242g
<pre>#0 PQCLEAN_DILITHIUM2_CLEAN_polyt0_unpack (r=r@entry=0</pre>	xbefb43c8, a=0xbffffbd8 <error: 0xbffffbd8="" access="" address="" at="" cannot="" memory="">,</error:>

4\021\320U\274\327\374\v\324\354\03 \022") at sign.c:107 ", mlen=30, U\274\327\374\v\324\354\032\277`\27 sign.c:227

16\330\$", f38000 "D/\003") at packing.c:155

\032\344\377{n\_m\274\021\320U\274\3

Memory access violation by faulting address bus J\202\bb\024\254-\253\262\346cH\031 \352\060ŭ\326\070A\332\340\200\267\\ (\030m\342\2201\033\221\261q\256\360



# End to End Attack Bird's eye-view





### **End to End Attack**



![](_page_70_Picture_3.jpeg)

![](_page_71_Picture_0.jpeg)

![](_page_71_Picture_1.jpeg)

![](_page_71_Picture_2.jpeg)


# **Responsible Disclosure**

• CVE 2022-47549

• Worked together with Linaro to deploy countermeasure in OP-TEE kernel

Website: <a href="https://nimishmishra.wixsite.com/disarmament">https://nimishmishra.wixsite.com/disarmament</a>





## Countermeasure

-	res = crypto_acipher_rsassa_verify(shdr->algo, &key, shdr->hash
-	SHDR_GET_HASH(shdr), shdr->h
-	SHDR_GET_SIG(shdr), shdr->si
+	<pre>FTMN_CALL_FUNC(res, &amp;ftmn, FTMN_INCR0,</pre>
+	<pre>crypto_acipher_rsassa_verify, shdr-&gt;algo, &amp;key,</pre>
+	<pre>shdr-&gt;hash_size, SHDR_GET_HASH(shdr), shdr-&gt;hash</pre>
+	<pre>SHDR_GET_SIG(shdr), shdr-&gt;sig_size);</pre>
+	if (!res) {
+	<pre>ftmn_checkpoint(&amp;ftmn, FTMN_INCR0);</pre>
+	goto out;
+	}
+	err_incr = 1;
+ err:	
+	res = TEE_ERROR_SECURITY;
+	<pre>FTMN_SET_CHECK_RES_NOT_ZER0(&amp;ftmn, err_incr * FTMN_INCR0, res);</pre>



### n\_size, nash\_size, ig\_size);

#### \_\_\_\_\_\_\_\_\_,

**# BHASIA** @BlackHatEvents



# **Other Implications**

 Re-enable Differential Fault Attack (DFA) on T-table implementation of AES (on SoCs)

Address Bus Faults to leak all shares of Masked PQC implementations (like Kyber)

**Observation:** All shares encapsulated within a **single** memory structure





# **Takeaways!**

• System + Execution Environment, not *just* the System

- Register sweeping fault model on a (new) architectural aspect System Bus
  - Implications for other systems?
- Rethinking protocol specifications for embedded systems in light of SCA+FI adversaries





## **Research @ Secured Embedded Architecture Laboratory, IIT Kgp**

### (Some) Research Directions

- Power/EM Side-channel evaluation of FPGAs/micro-controllers/SoCs
- Fault Attacks, Fault Analysis, and design of countermeasures
- Evaluation of **Microarchitectural attack** s cenarios on workstations as well as embedded systems
- Others directions...



**# BHASIA** @BlackHatEvents

# black hat ASIA 2024

# APRIL 18-19, 2024

BRIEFINGS

# Thank You!



## WEBSITE

**#BHASIA** @BlackHatEvents