Intellectual Property Exposure

Subverting and Securing Intellectual Property Encapsulation in Texas Instruments Microcontrollers

Marton Bognar, Cas Magnus, Frank Piessens, Jo Van Bulck

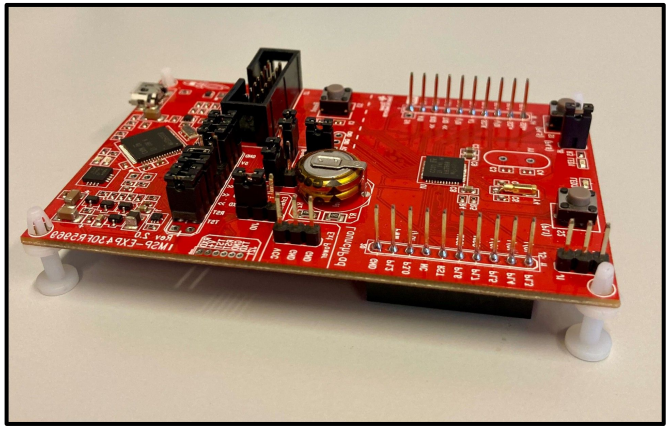
DistriNet, KU Leuven, Belgium

DistriNet

KU LEUVEN

Texas Instruments

Texas Instruments MSP430

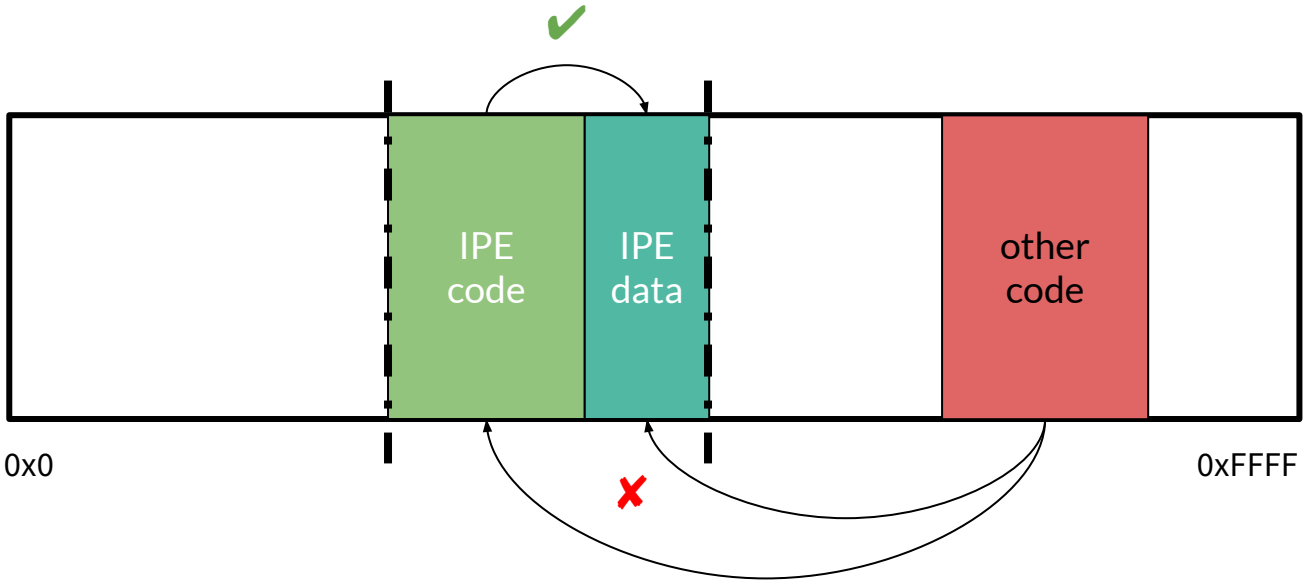


- Low-power microcontrollers
- FRAM edition (2014) with security features
  - Physical tamper protection
  - Memory protection unit (MPU)
  - **Intellectual Property Encapsulation (IPE)**

Intellectual Property Encapsulation

bottom:
0x0400

top:
0x0600



+ protection from JTAG debug port, direct memory access (DMA)
→ Looks like a trusted execution environment (TEE)!

Attack primitives

	Attack primitive	C ✗	I ✗	Section
Architectural	Controlled `call` corruption *(new)*	◐	●	§3.1
	Code gadget reuse [35]	◐	◐	§3.2
	Interrupt register state [73]	●	●	§3.3
	Interface sanitization [69]	◐	◐	§6.1
Side channels	Cache timing side channel [23, 39]	◐	○	§3.4.1
	Interrupt latency side channel [71]	◐	○	§3.4.2
	Controlled channel [25, 77]	◐	○	§3.4.3
	Voltage fault injection [31, 40]	○	○	§A.1
	DMA contention side channel [7, 8]	○	○	§A.2

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●)
Tested on multiple different MSP430 CPUs

Attack primitives

Software-based

	Attack primitive	C ✗	I ✗	Section
Architectural	Controlled `call` corruption *(new)*	◐	●	§3.1
	Code gadget reuse [35]	◐	◐	§3.2
	Interrupt register state [73]	●	●	§3.3
	Interface sanitization [69]	◐	◐	§6.1
Side channels	Cache timing side channel [23, 39]	◐	○	§3.4.1
	Interrupt latency side channel [71]	◐	○	§3.4.2
	Controlled channel [25, 77]	◐	○	§3.4.3
	Voltage fault injection [31, 40]	○	○	§A.1
	DMA contention side channel [7, 8]	○	○	§A.2

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●)
Tested on multiple different MSP430 CPUs

Attack primitives

Sah et al. “RIPencapsulation: Defeating IP Encapsulation on TI MSP Devices”. WOOT ‘24

(concurrent work with a focus on these primitives)

	Attack primitive	C ✗	I ✗	Section
Architectural	Controlled `call` corruption *(new)*	◐	●	§3.1
	Code gadget reuse [35]	◐	◐	§3.2
	Interrupt register state [73]	●	●	§3.3
	Interface sanitization [69]	◐	◐	§6.1
Side channels	Cache timing side channel [23, 39]	◐	○	§3.4.1
	Interrupt latency side channel [71]	◐	○	§3.4.2
	Controlled channel [25, 77]	◐	○	§3.4.3
	Voltage fault injection [31, 40]	○	○	§A.1
	DMA contention side channel [7, 8]	○	○	§A.2

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●)
Tested on multiple different MSP430 CPUs

Attack primitives

	Attack primitive	C ✗	I ✗	Section
Architectural	Controlled `call` corruption *(new)*	◐	●	§3.1
	Code gadget reuse [35]	◐	◐	§3.2
	Interrupt register state [73]	●	●	§3.3
	Interface sanitization [69]	◐	◐	§6.1
Side channels	Cache timing side channel [23, 39]	◐	○	§3.4.1
	Interrupt latency side channel [71]	◐	○	§3.4.2
	Controlled channel [25, 77]	◐	○	§3.4.3
	Voltage fault injection [31, 40]	○	○	§A.1
	DMA contention side channel [7, 8]	○	○	§A.2

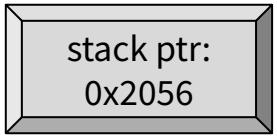
Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●)
Tested on multiple different MSP430 CPUs

Controlled *call* corruption

```
int factorial(int n) {
  int sub = n - 1;
  return (n * factorial(sub));
}

int main() {
  int result = factorial(5);
  result += 4;
}
```

Controlled *call* corruption

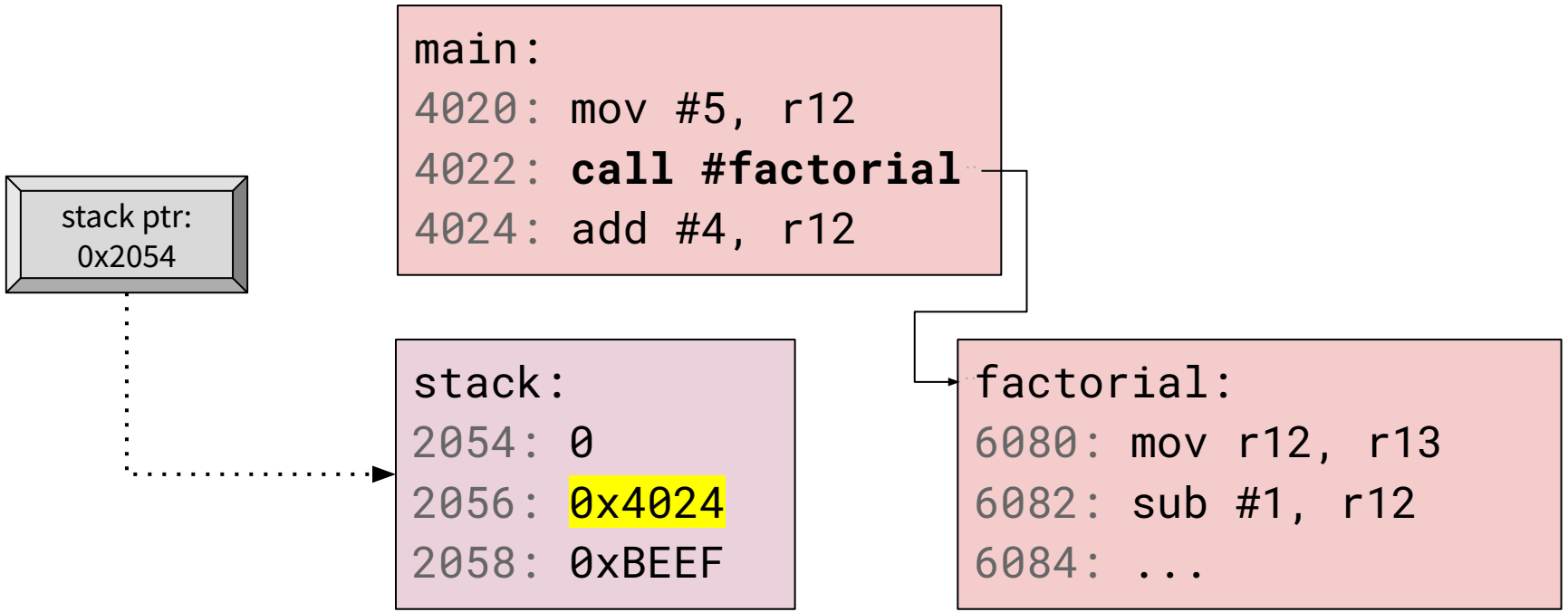



```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
stack:
2054: 0
2056: 0
2058: 0xBEEF
```

```
factorial:
6080: mov r12, r13
6082: sub #1, r12
6084: ...
```

Controlled *call* corruption



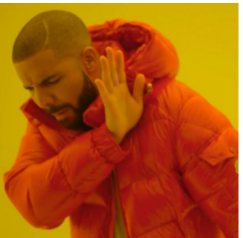
Controlled *call* corruption

stack ptr:
0x5050

```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: AES_KEY
5052: 0
5054: 0x42
```

```
factorial:
6080: mov r12, r13
6082: sub #1, r12
6084: ...
```



Controlled *call* corruption

stack ptr:
0x5050

```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: 0
5052: 0
5054: 0x42
```

```
factorial_ipe:
5000: mov r12, r13
5002: sub #1, r12
5004: ...
```

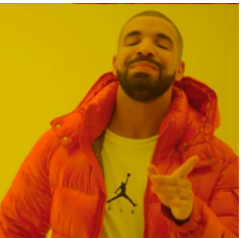
Controlled *call* corruption

stack ptr:
0x504E

```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: 0x4024
5052: 0
5054: 0x42
```

```
factorial_ipe:
5000: mov r12, r13
5002: sub #1, r12
5004: ...
```



Controlled *call* corruption

Corrupt code in IPE to crash the application

Overwrite secret data with known values

Insert a universal read gadget

Controlled *call* corruption

Table 3: A subset of instruction encodings on MSP430, falling in the RAM (*italic*) or FRAM ranges (cf. Table 2).

Encoding	Inst	Encoding	Inst
0xFxxx	and	0x8xxx	sub
0xExxx	xor	0x7xxx	subc
0xDxxx	bis	0x6xxx	addc
0xCxxx	bic	0x5xxx	add
0xBxxx	bit	0x4xxx	mov
0xAxxx	dadd	*0x3xxx*	*jmp, jl, jge, jn*
0x9xxx	cmp	*0x2xxx*	*jc, jnc, jz, jnz*

Controlled *call* corruption

Corrupt code in IPE to crash the application

Overwrite secret data with known values

Insert a universal read gadget

Overwrite the stored IPE configuration to remove the protection

PSIRT Notification

MSP430FR5xxx and MSP430FR6xxx IP Encapsulation Write Vulnerability

Texas Instruments

Summary

The IP Encapsulation feature of the Memory Protection Unit may not properly prevent writes to an IPE protected region under certain conditions. This vulnerability assumes an attacker has control of the device outside of the IPE protected region (access to non-protect memory, RAM, and CPU registers).

Vulnerability

TI PSIRT ID

TI-PSIRT-2023-040180

CVE ID

Not applicable.

CVSS Base Score

7.1

CVSS Vector

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Affected Products

- MSP430FR58xx family devices
- MSP430FR59xx family devices
- MSP430FR6xxx family devices

Mitigations?

Easy fixes require hardware changes…

Software mitigation framework

Components:

- Source-to-source translator
- Assembly stubs
- C helper functions

Responsibilities:

- Repurpose the MPU
- Protect against CCC and other architectural attack primitives

Limitations:

- Weaker attacker model
- Initiates a reset when switching to untrusted code

IPE Exposure

- Security analysis of Texas Instruments IPE
  - Novel vulnerability: *controlled call corruption*
  - Reproduction of other known primitives, including side channels
  - Complete leakage of protected code and data
- Software mitigation framework
  - Restoring security guarantees
  - Performance evaluation

https://github.com/martonbognar/ipe-exposure