



**XIII
JORNADAS
STIC
CCN-CERT**

ccn-cert
centro criptológico nacional



**Los secretos de tu
chip IoT**

**COMUNIDAD Y CONFIANZA,
BASES DE NUESTRA CIBERSEGURIDAD**

#XIIIJORNADASCNCERT



Javier Tallón – Director Técnico

 jtallon@jtsec.es

 [@javiertallon](https://twitter.com/javiertallon)

Más de 12 años en Seguridad IT

Full-stack hacker wannabe



 [@jtsecES](https://twitter.com/jtsecES)

Certificación de la ciberseguridad

Laboratorio acreditado por CCN

Evaluaciones LINCE



Sedes en Granada y Madrid

Índice

1. Introducción
2. Técnicas de ataque a la protección contra lectura
 1. *ARM Cortex-M0 Register Readout*
 2. *Heart of Darkness*
 3. *UV-C Security Fuse Erase*
 4. *Cold Boot Stepping*
 5. *Flash Lock Race*
 6. Inyección de Faltas
 7. *Side Channel Attack*
3. Conclusiones

Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. *ARM Cortex-M0 Register
Readout*

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

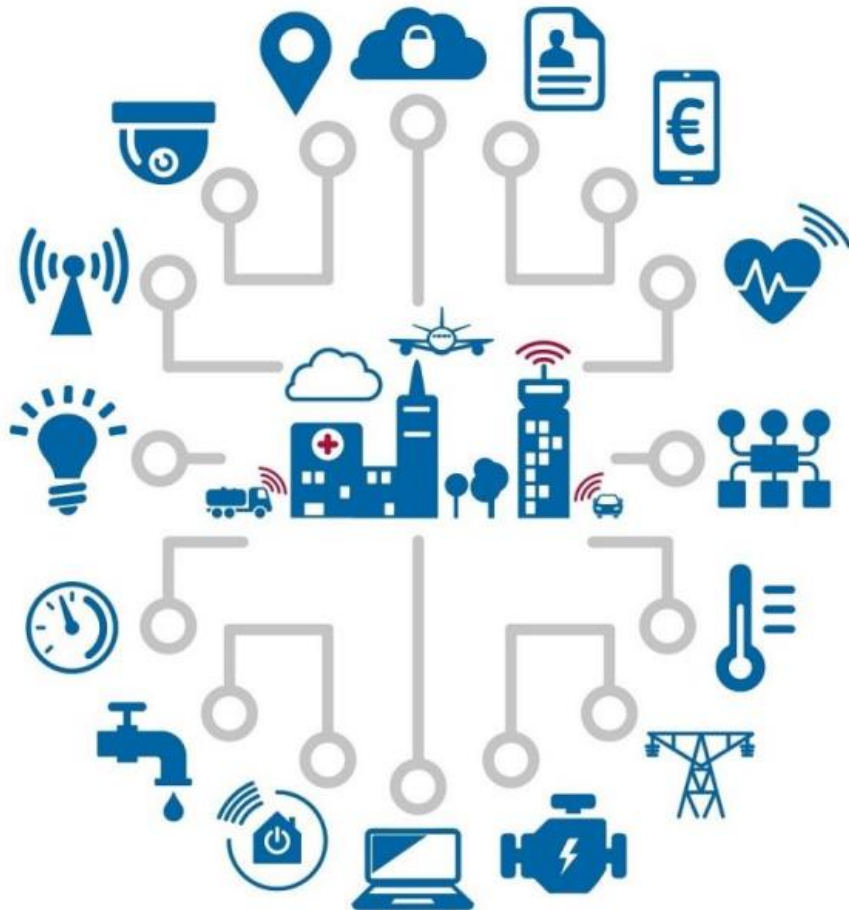
6. *Inyección de Faltas*

7. *Side Channel Attack*

3. Conclusiones

1. Introducción

¿Qué es IoT?



ENISA define el IoT como **“un ecosistema ciberfísico de sensores y actuadores interconectados que permiten toma de decisiones inteligente”**.

Fuente: <https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot>

1. Introducción

Predicción del Crecimiento IoT

Connected devices (billions)



	2016	2022	CAGR
Wide-area IoT	0.4	2.1	30%
Short-range IoT	5.2	16	20%
PC/laptop/tablet	1.6	1.7	0%
Mobile phones	7.3	8.6	3%
Fixed phones	1.4	1.3	0%
	16 billion	29 billion	10%

Fuente: <https://www.ericsson.com/en/mobility-report/internet-of-things-forecast>

Internet of Things

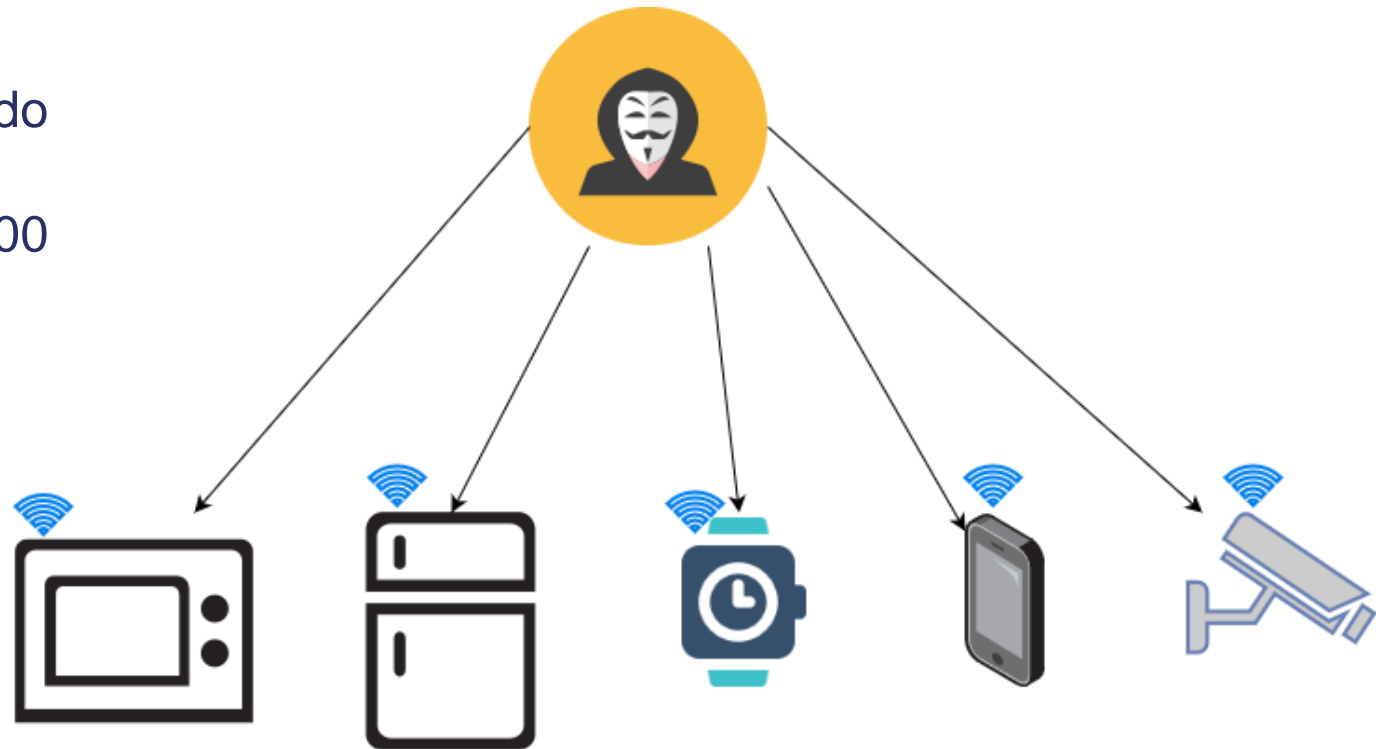
“The S in IoT stands for Security”

1. Introducción

IoT en el Punto de Mira de los Hackers

- IoT *botnets*
- Denegación de Servicio Distribuido (**DDOS**)
- Mirai (2016): Más de 600,000 dispositivos infectados
- Okiru y Masuta: Más de 700,000

Fuente: <https://krebsonsecurity.com/2019/09/satori-iot-botnet-operator-pleads-guilty/>




1. Introducción

Vulnerabilidades IoT. *eHealth*

Security

Hacking these medical pumps is as easy as copying a booby-trapped file over the network

Uncle Sam sounds alarm after Windows CE SMB left wide open on hospital equipment

By Thomas Claburn in San Francisco 13 Jun 2019 at 19:22 34  SHARE ▼



Fuente:
https://www.theregister.co.uk/2019/06/13/medical_workstation_vulnerabilities/



ÚLTIMA HORA 19/11/2019 16:47

Publicado un informe de código dañino sobre Raccoon Stealer

[Inicio](#) > [Seguridad al día](#) > [Noticias de actualidad](#) > [Cómo implementar servicios de autenticación de doble factor](#)

Las bombas de insulina de Medtronic anteriores a 2013 podrían sufrir ciberataques

Fuente: <https://www.ccn-cert.cni.es/seguridad-al-dia/noticias-seguridad/8342-las-bombas-de-insulina-de-medtronic-antiores-a-2013-podrian-sufrir-ciberataques.html>



CYBER
MDX

CyberMDX Research

Team Discovers

Medical Device

Vulnerability in GE

Anesthesia and

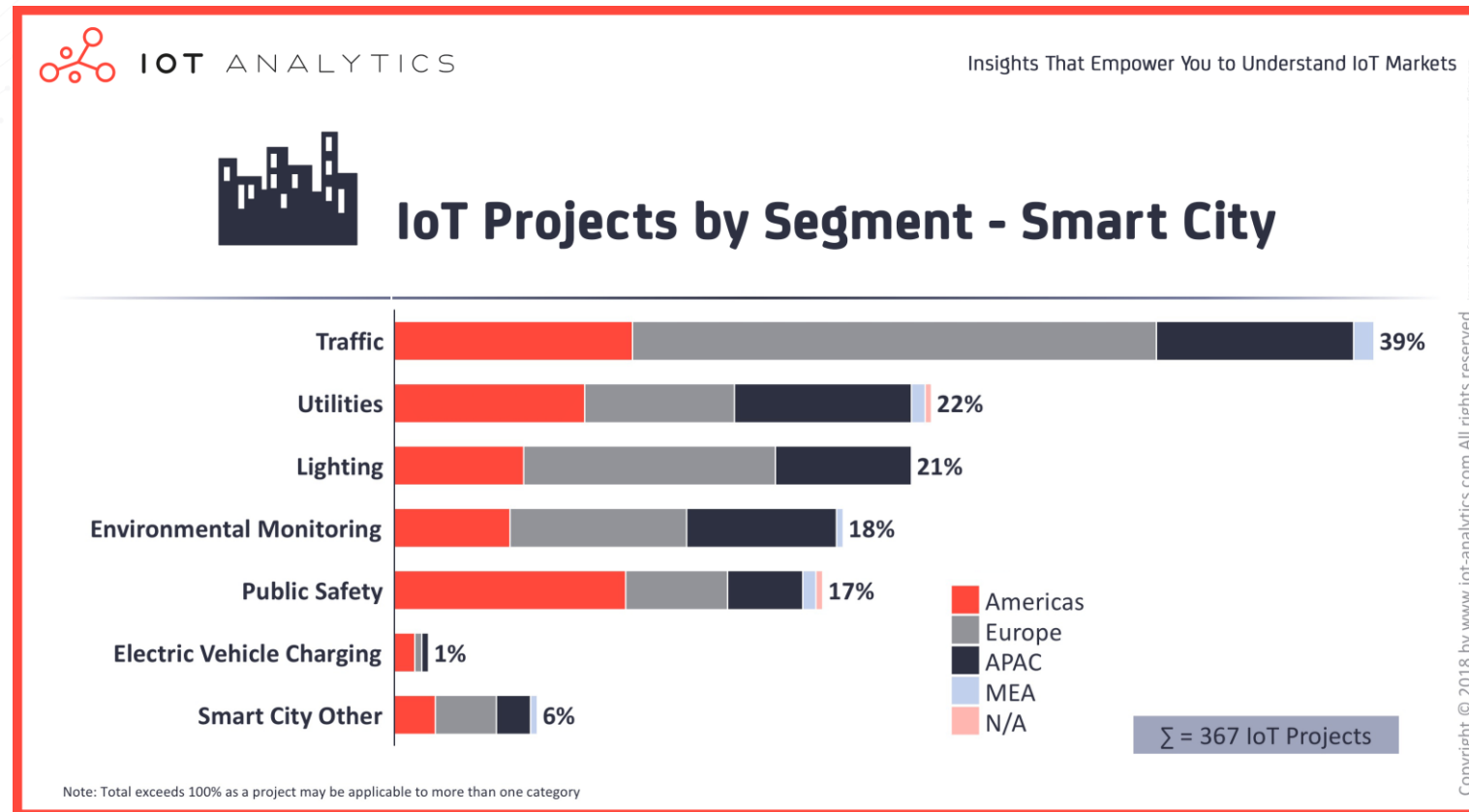
Respiratory Devices

ICS-CERT Advisory
(ICSMA-19-190-01)

Fuente:
<https://www.cybermdx.com/vulnerability-research-disclosures/vulnerability-in-ge-anesthesia-aesthesia-aespire>

1. Introducción

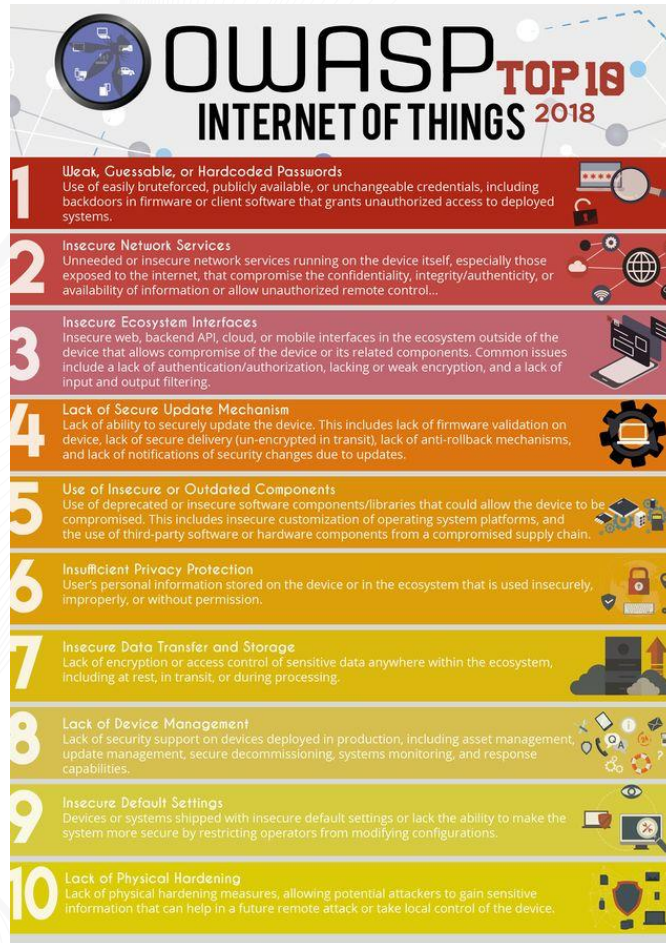
Vulnerabilidades IoT. *Smart Cities*



Fuente: <https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/>

1. Introducción

Vulnerabilidades IoT

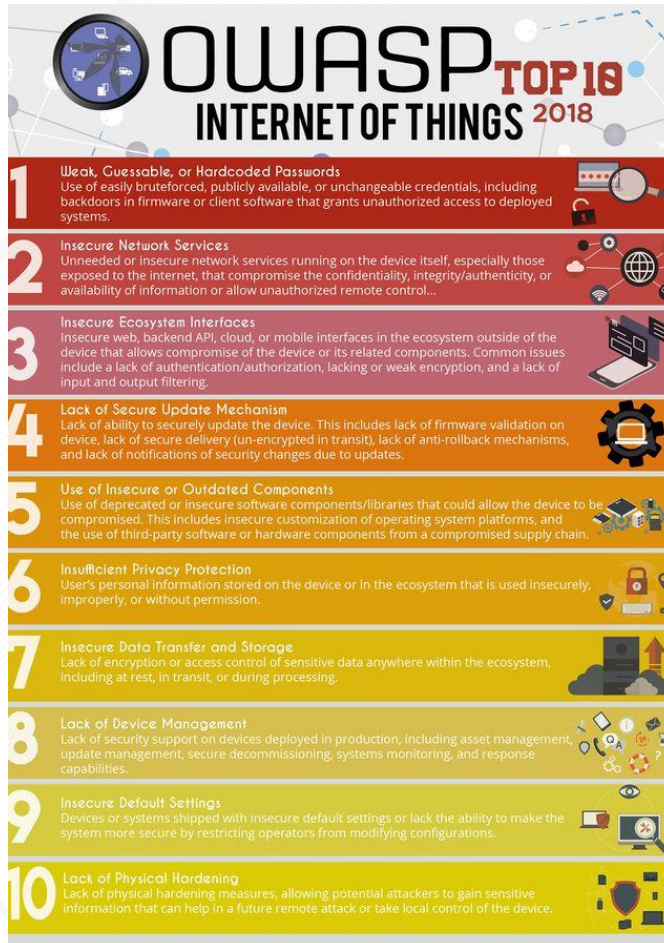


1. Contraseñas débiles, inseguras o por defecto.
2. Servicios expuestos a Internet inseguros
3. Ecosistema inseguro (cifrado débil, falta de autenticación, integridad, etc.)
4. Falta de mecanismos de actualización seguros
5. Uso de componentes desactualizados
6. Protección de privacidad insuficiente
7. Transferencia y almacenamiento de datos de forma insegura
8. Monitorización insuficiente
9. Configuración por defecto insegura
10. Falta de seguridad física

Fuente: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

1. Introducción

Vulnerabilidades IoT



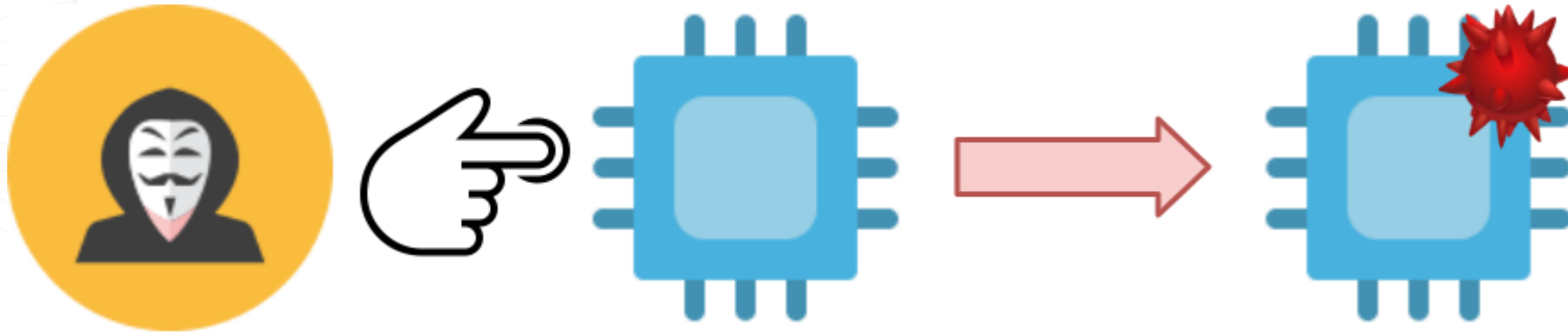
1. Contraseñas débiles, inseguras o por defecto.
2. Servicios expuestos a Internet inseguros
3. Ecosistema inseguro (cifrado débil, falta de autenticación, integridad, etc.)
4. Falta de mecanismos de actualización seguros
5. Uso de componentes desactualizados
6. Protección de privacidad insuficiente
7. Transferencia y almacenamiento de datos de forma insegura
8. Monitorización insuficiente
9. Configuración por defecto insegura
10. Falta de seguridad física

Fuente: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

1. Introducción

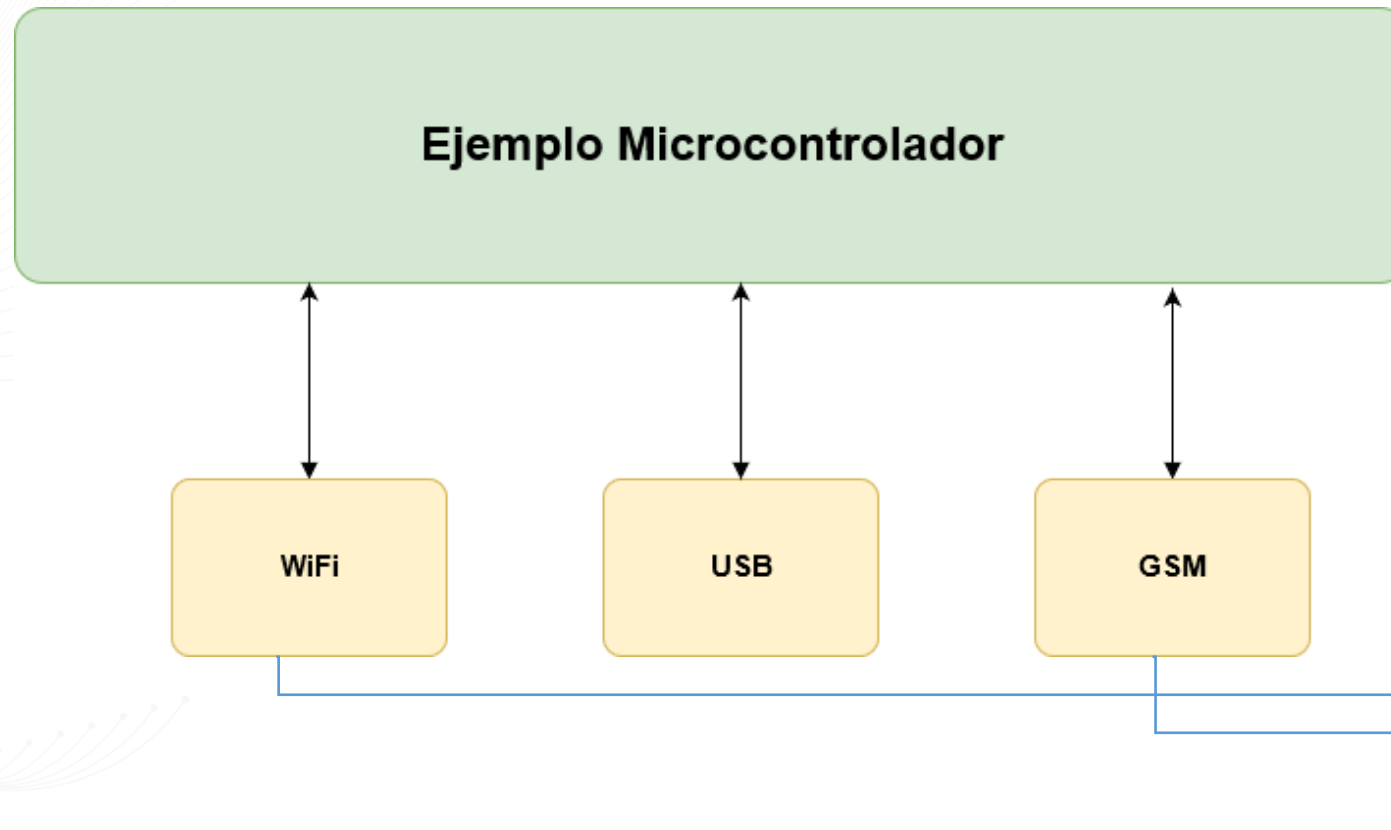
Acceso Físico a un Dispositivo IoT

Game Over?



1. Introducción

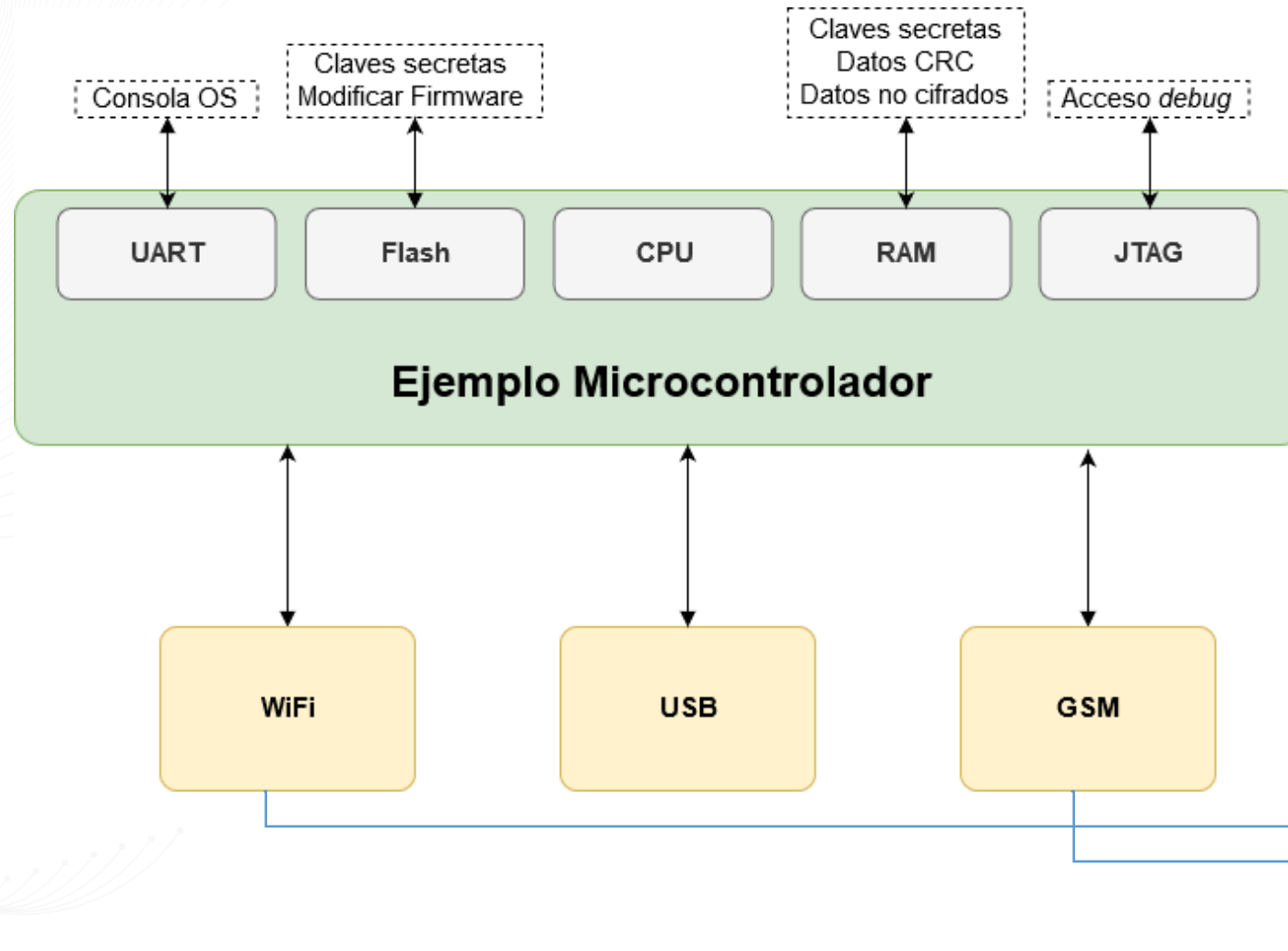
De Caja Negra a Caja Gris



Back-end

1. Introducción

De Caja Negra a Caja Gris



Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. *ARM Cortex-M0 Register
Readout*

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

6. *Inyección de Faltas*

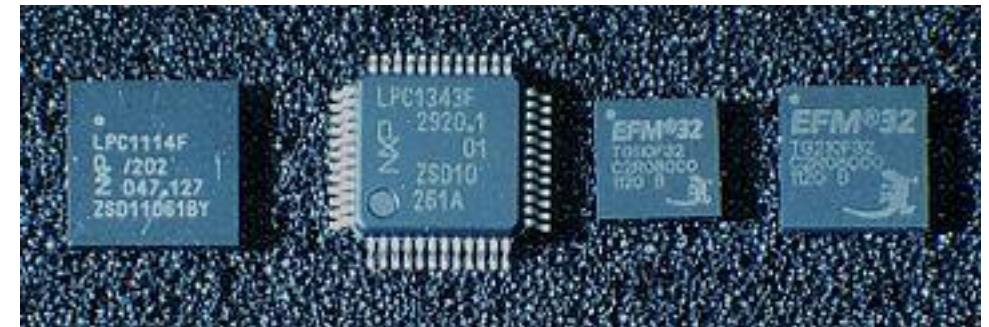
7. *Side Channel Attack*

3. Conclusiones

2. Técnicas de Ataque a la Protección Contra Lectura

ARM Cortex-M0 *Register Readout*

- ABOV Semiconductor AC30M1x64
- Cypress PSoC 4000, 4100, 4100M, 4200, 4200DS, 4200L, 4200M
- Infineon XMC1100, XMC1200, XMC1300, XMC1400, TLE984x
- Dialog Semiconductor DA1458x, DA1468x
- **Nordic nRF51**
- NXP LPC1100, LPC1200
- nuvoTon NuMicro M0 Family
- Sonix SN32F700
- **ST STM32 F0**
- Toshiba TX00
- ...



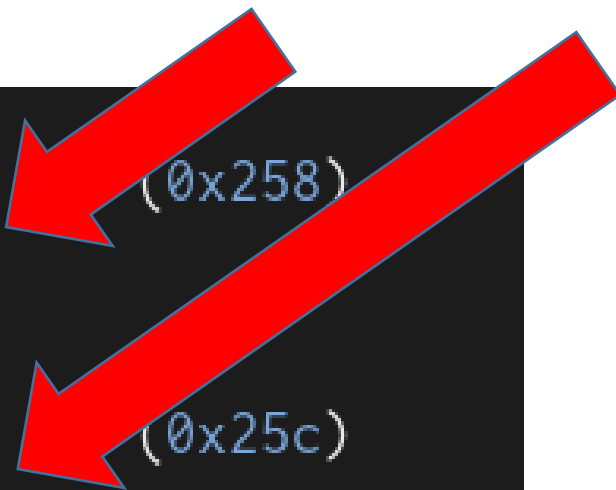
2. Técnicas de Ataque a la Protección Contra Lectura

ARM Cortex-M0 *Register Readout*

- Memoria *flash* protegida contra lectura
- Tenemos acceso a los registros de la CPU
- Capturar memoria flash cuando esta es cargada en algún registro

**Instrucciones
Vulnerables**

```
(gdb) x/7i 0x22e
0x22e:    ldr    r0, [pc, #40]
0x230:    ldr    r2, [r0, #0]
0x232:    orrs   r2, r1
0x234:    str    r2, [r0, #0]
0x236:    ldr    r0, [pc, #36]
0x238:    ldr    r2, [r0, #0]
0x23a:    orrs   r2, r1
```



Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. ARM Cortex-M0 *Register Readout*

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

6. Inyección de Faltas

7. *Side Channel Attack*

3. Conclusiones

2. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness

- PIC18FXX2/XX8
- Dividido en bloques
- Configuración **individual** para cada bloque

MEMORY SIZE / DEVICE			Block Code Protection Controlled By:
16 Kbytes (PIC18FX42)	32 Kbytes (PIC18FX52)	Address Range	
Boot Block	Boot Block	000000h 0001FFh	CPB, WRTB, EBTRB
Block 0	Block 0	000200h 001FFFh	CP0, WRT0, EBTR0
Block 1	Block 1	002000h 003FFFh	CP1, WRT1, EBTR1
Unimplemented Read '0's	Block 2	004000h 005FFFh	CP2, WRT2, EBTR2
Unimplemented Read '0's	Block 3	006000h 007FFFh	CP3, WRT3, EBTR3
Unimplemented Read '0's	Unimplemented Read '0's	008000h 1FFFFFFh	(Unimplemented Memory Space)

Fuente: <http://ww1.microchip.com/downloads/en/DeviceDoc/39576b.pdf>

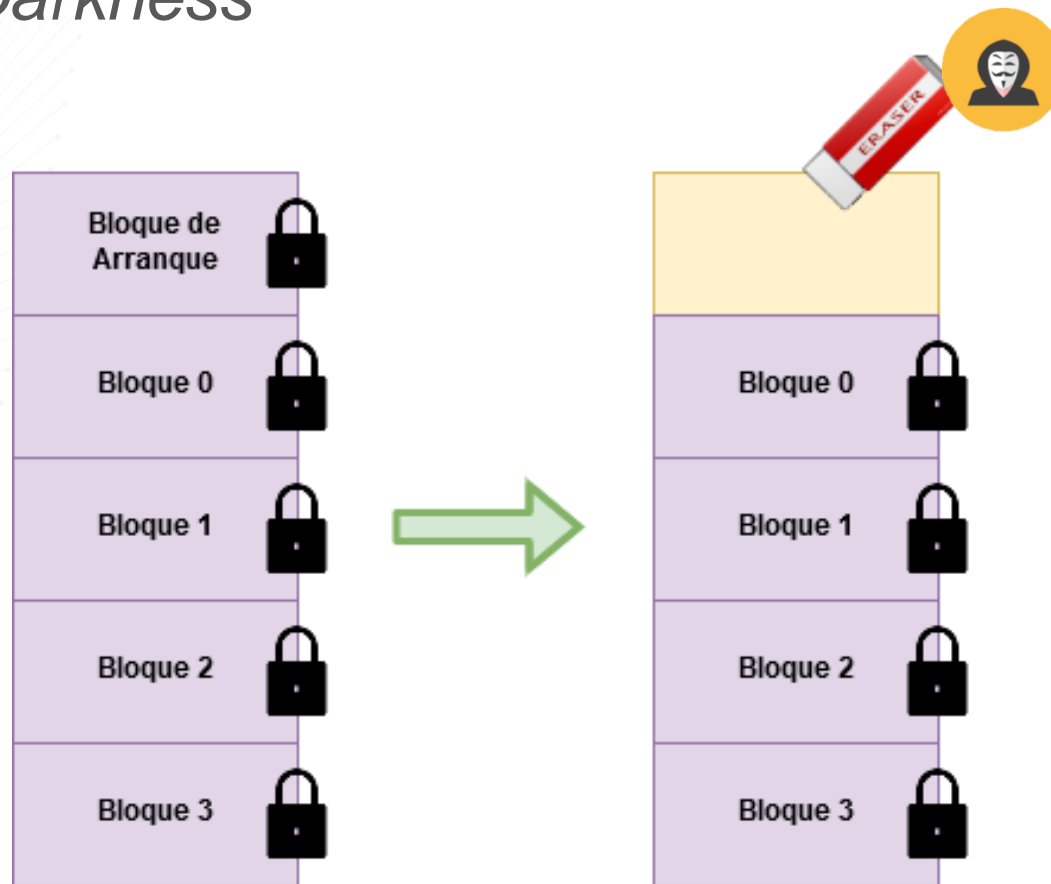
2. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



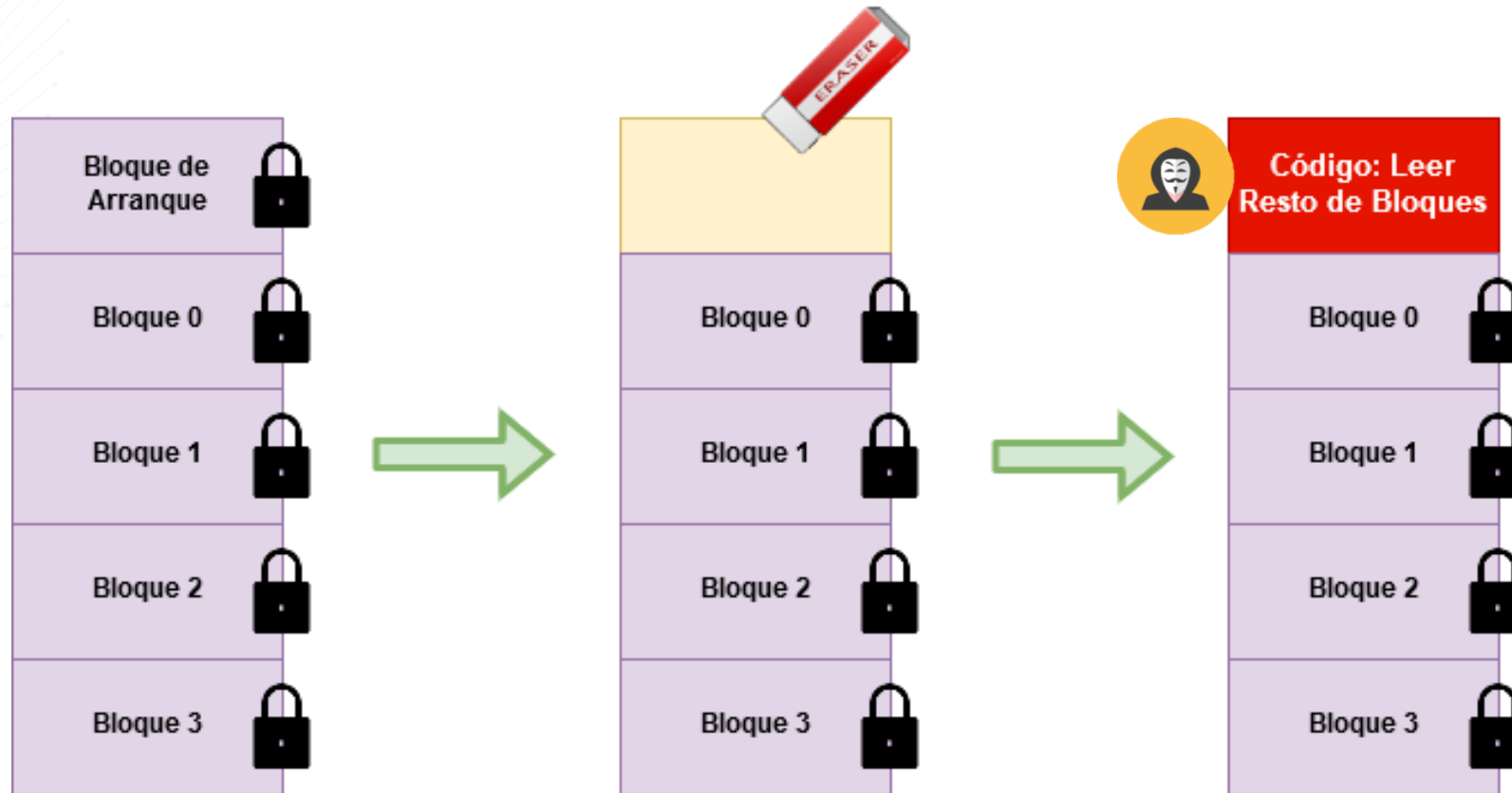
2. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



2. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



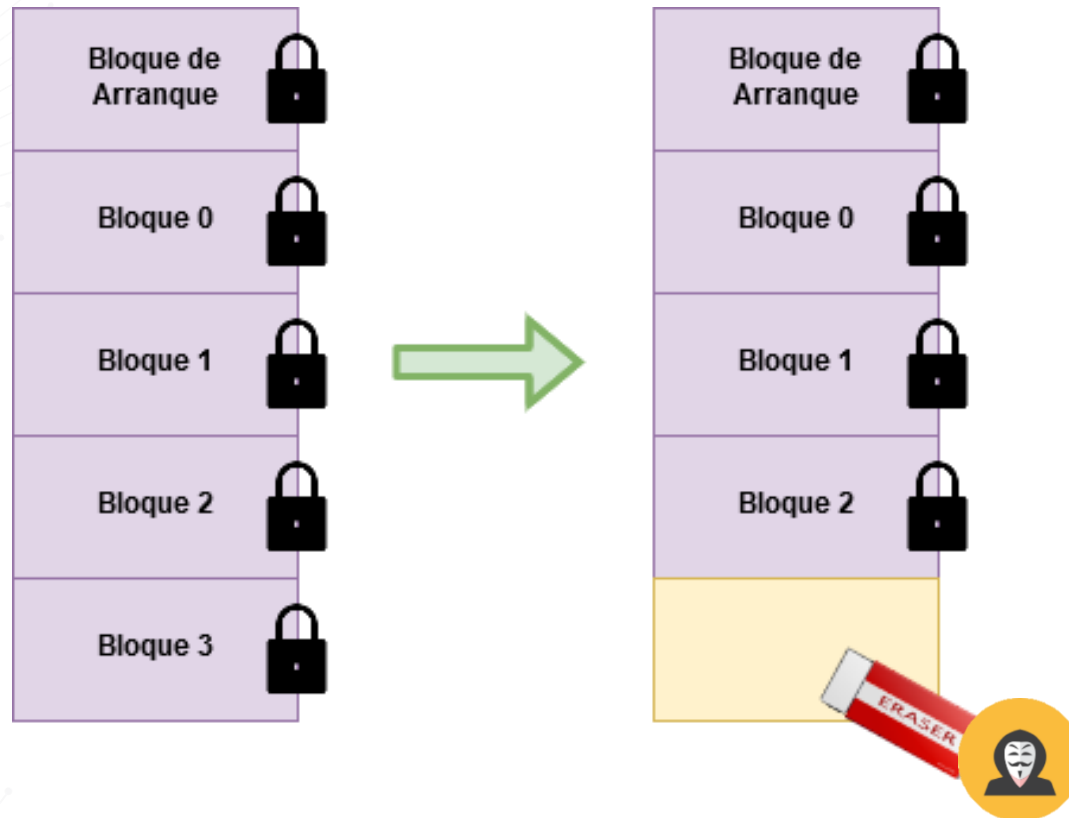
2. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



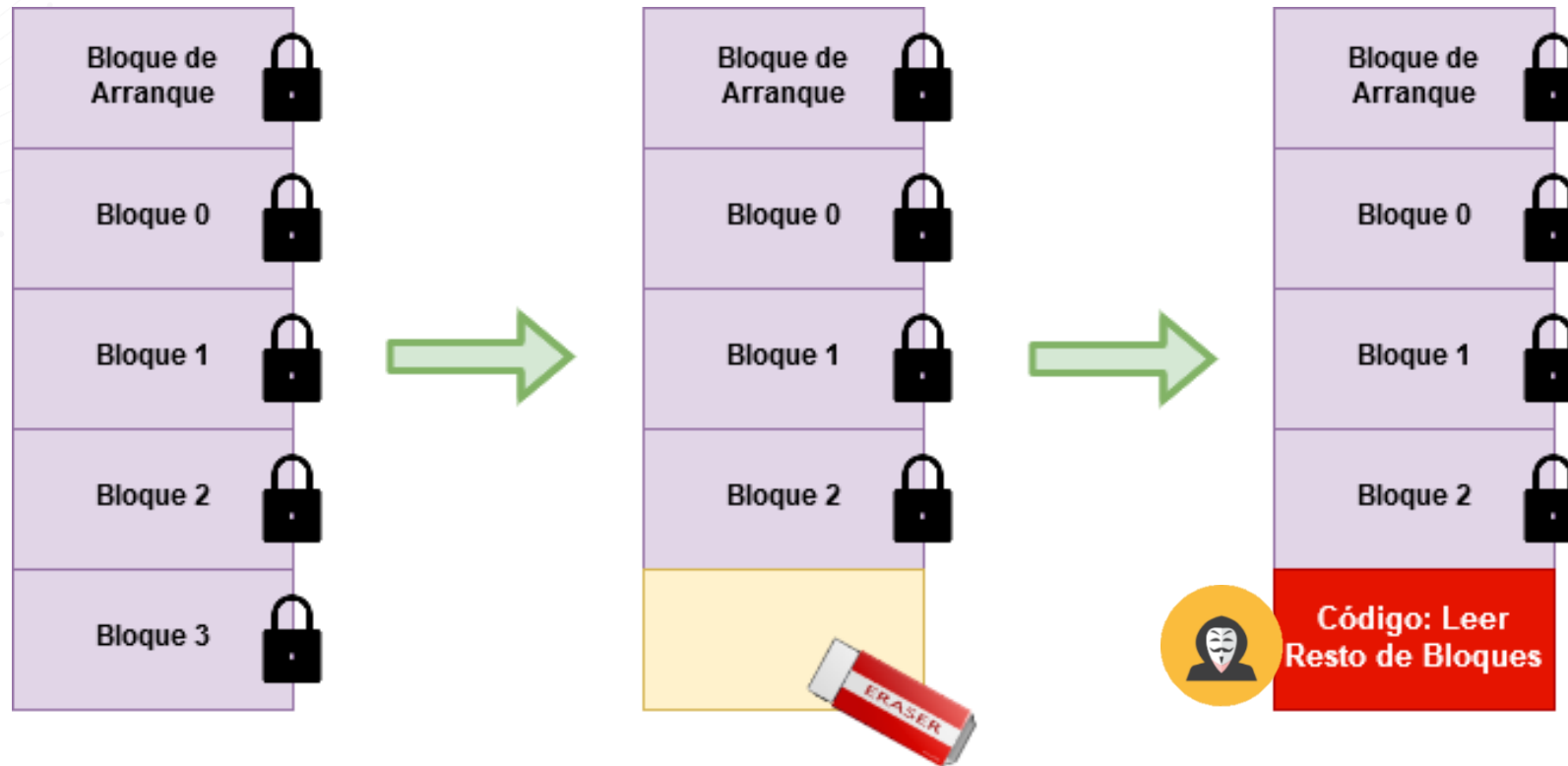
2. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



2. Técnicas de Ataque a la Protección Contra Lectura

Heart of Darkness



Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. ARM Cortex-M0 *Register
Readout*

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

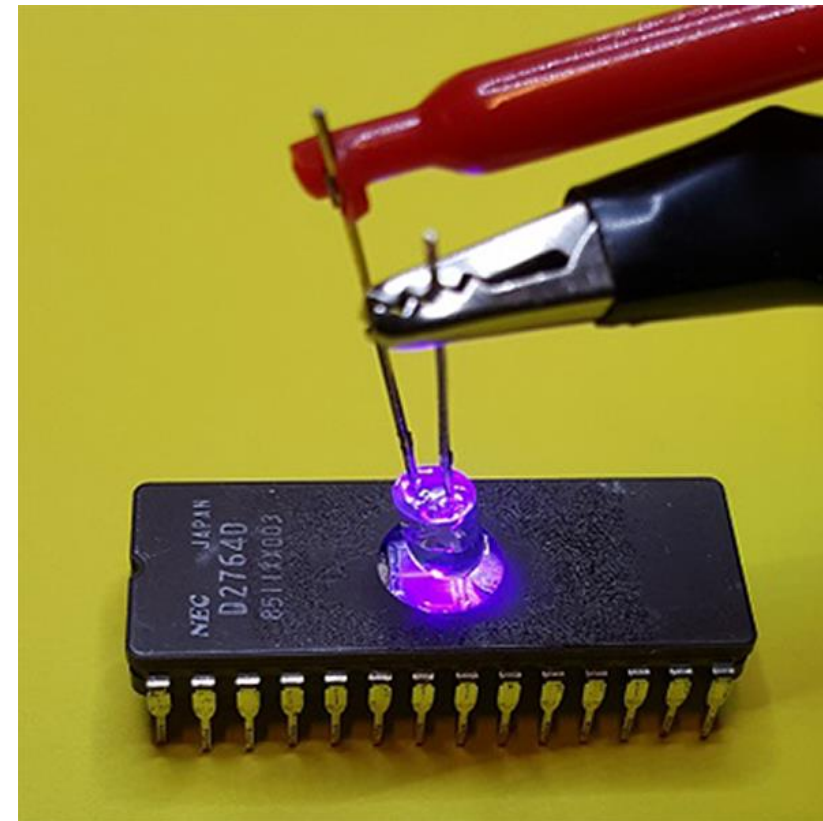
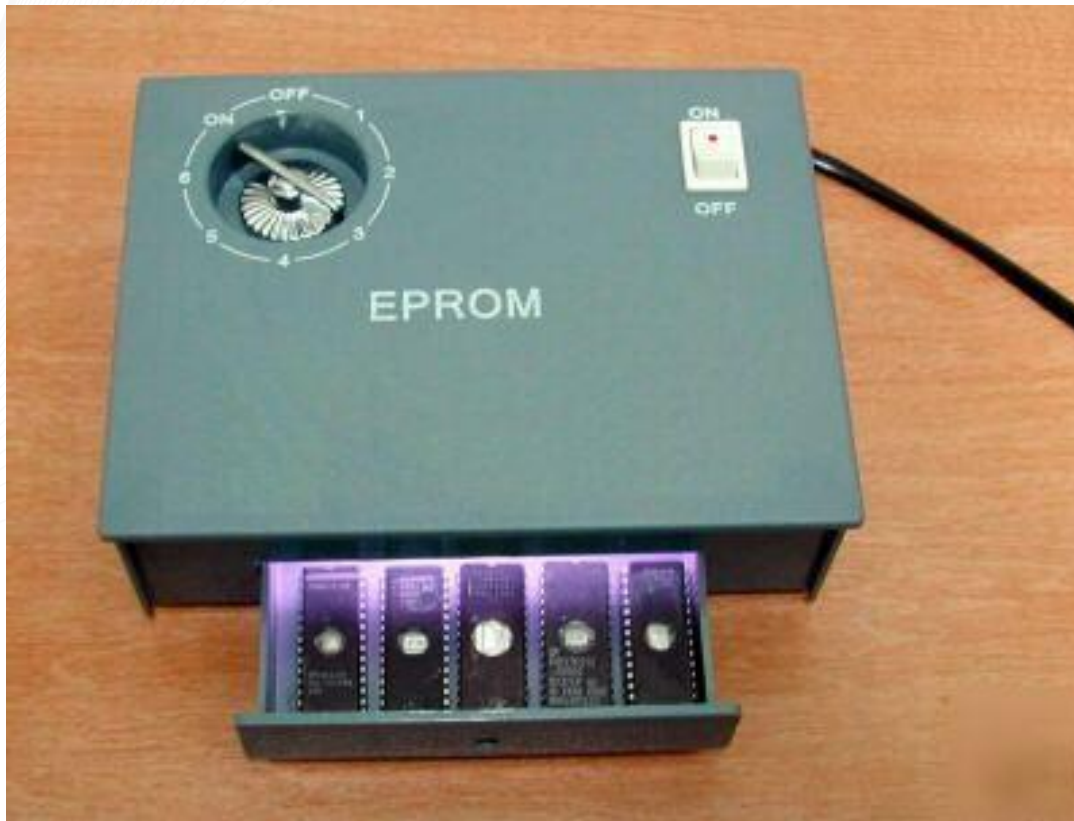
6. Inyección de Faltas

7. *Side Channel Attack*

3. Conclusiones

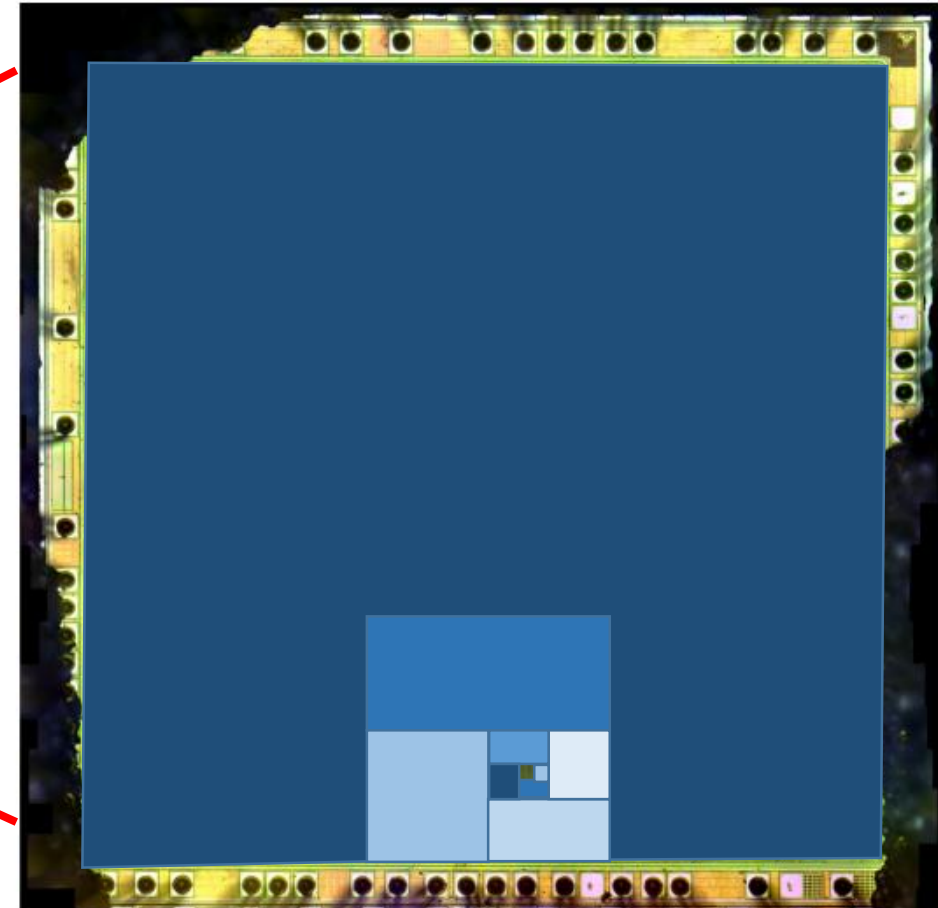
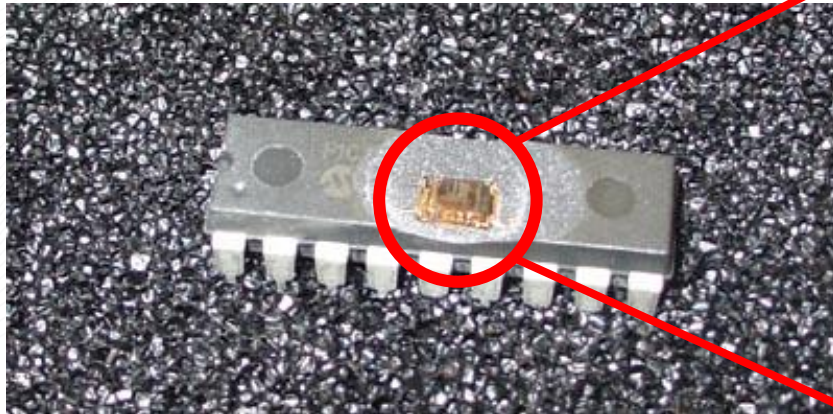
2. Técnicas de Ataque a la Protección Contra Lectura

UV-C Security Fuse Erase



2. Técnicas de Ataque a la Protección Contra Lectura

UV-C Security Fuse Erase



Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. ARM Cortex-M0 *Register*

Readout

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

3. Conclusiones

4. *Cold Boot Stepping*

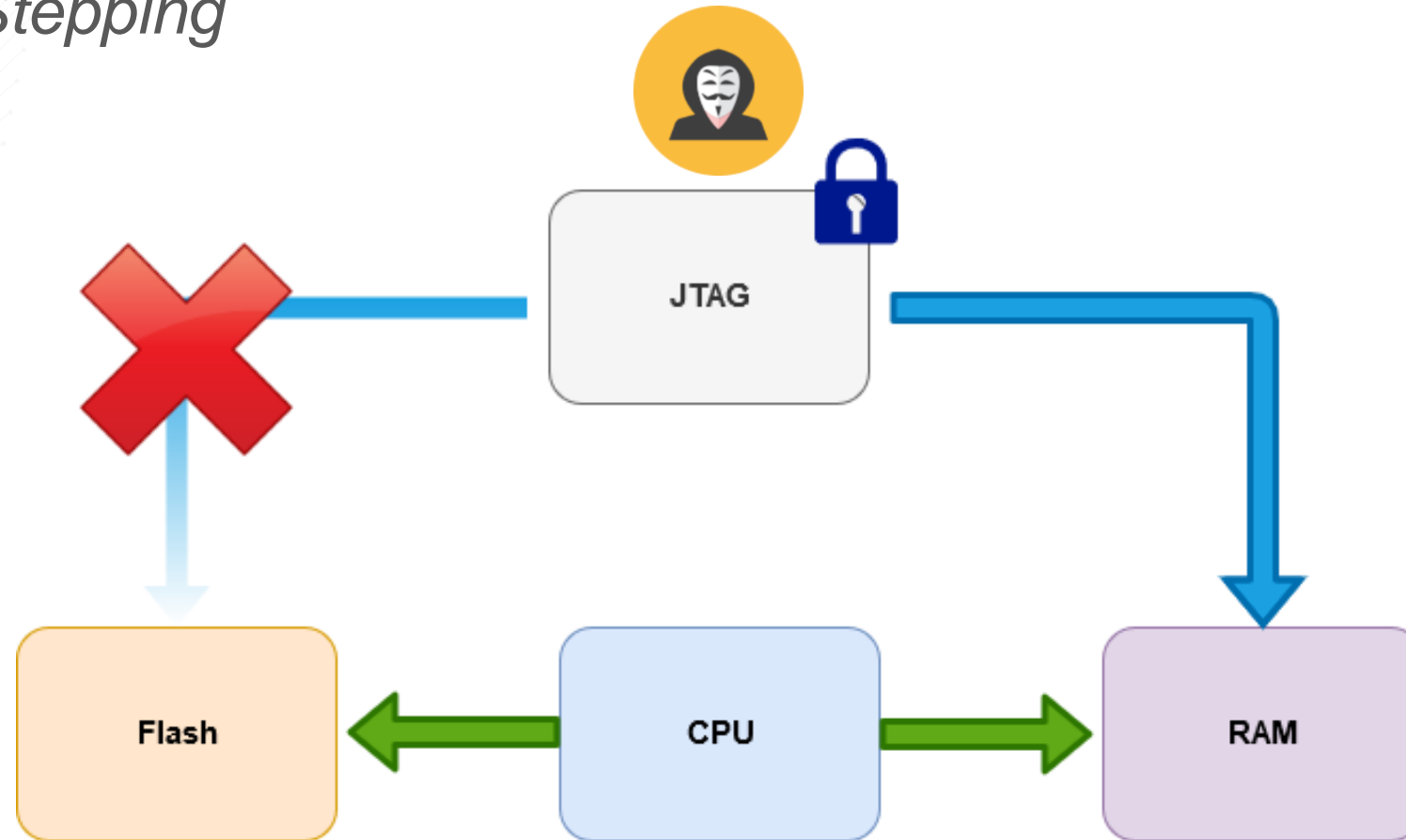
5. *Flash Lock Race*

6. Inyección de Faltas

7. *Side Channel Attack*

2. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping



2. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping

- *Bootloader* comprueba la integridad de la memoria *flash* usando **CRC** o **hash**

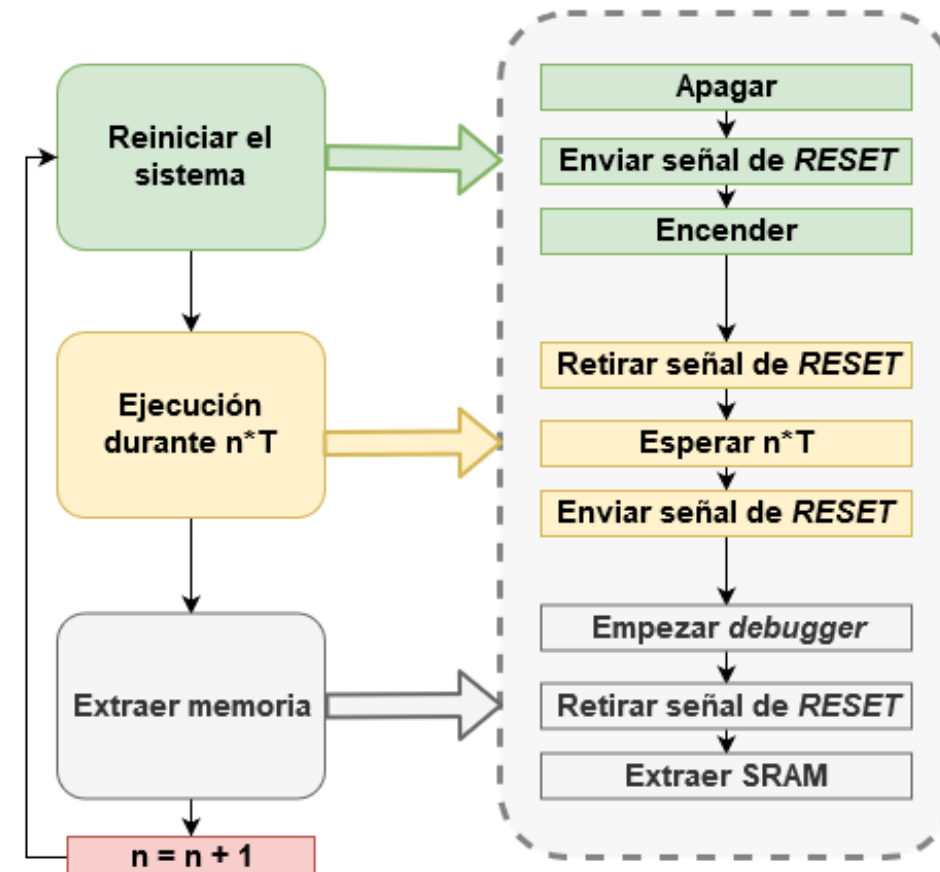
```
257 // calc crc
258 u16_t crc = 0xffff;
259 const u32_t start = SHMEM->user[BOOTLOADER_SHMEM_SPIF_FW_ADDR_UIX] + sizeof(fw_upgrade_info);
260 for (addr = start; addr < start + fui.len; addr++) {
261     u8_t c;
262     BL_IMG_READ(addr, (u8_t*)&c, 1);
263     crc = _bootloader_crc_ccitt_16(crc, c);
264 }
265
266 if (crc != fui.crc) {
267     b_putstr(" fw file bad crc:");
268     b_puthex16(crc);
269     b_put('/');
270     b_puthex16(fui.crc);
271     b_put('\n');
272     return FALSE;
273 }
```

Loops over flash calculating CRC

2. Técnicas de Ataque a la Protección Contra Lectura

Cold Boot Stepping

- Cada iteración CRC tarda T milisegundos
- Parar el sistema n veces en el **momento exacto** en que cada uno de los bytes de la flash pasa por la memoria RAM
- Podría ser de utilidad un **reloj externo**



Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. ARM Cortex-M0 *Register Readout*

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

6. Inyección de Faltas

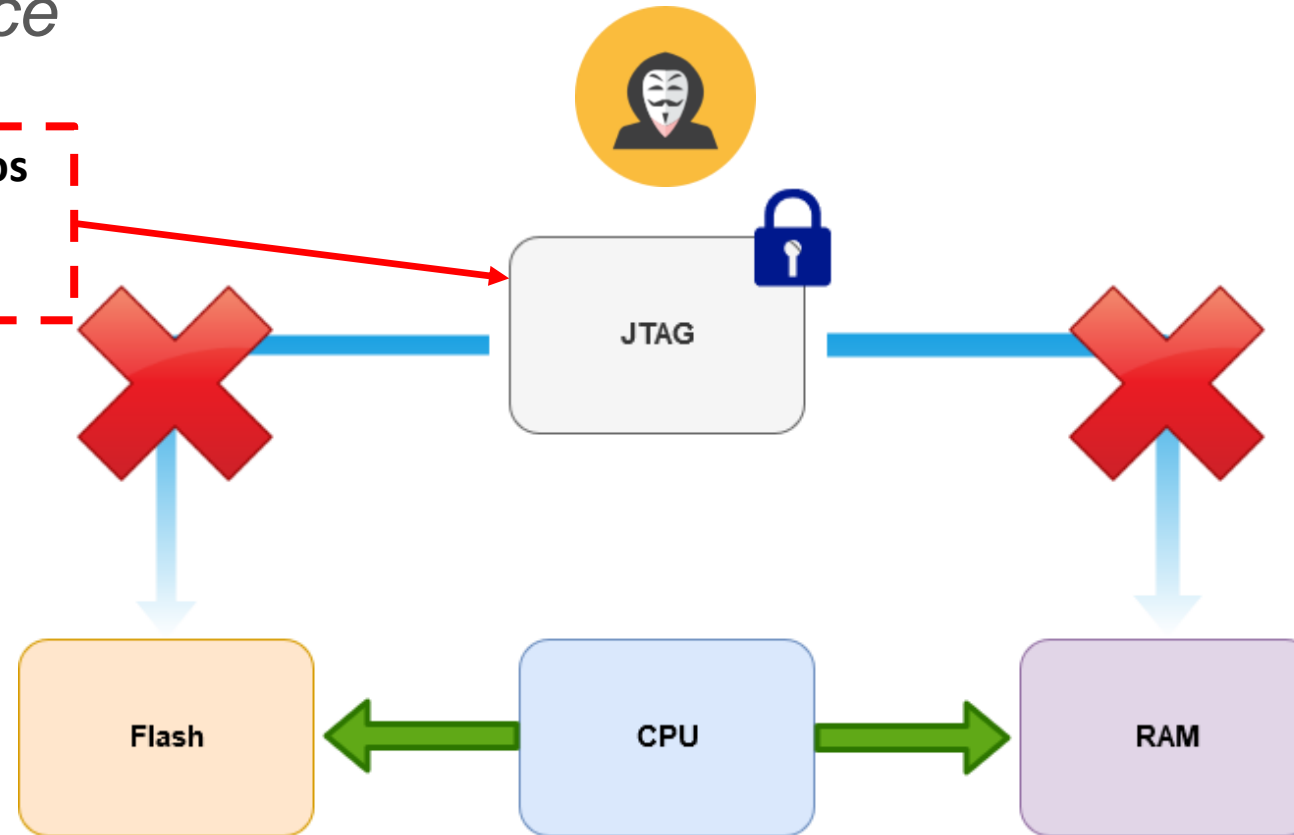
7. *Side Channel Attack*

3. Conclusiones

2. Técnicas de Ataque a la Protección Contra Lectura

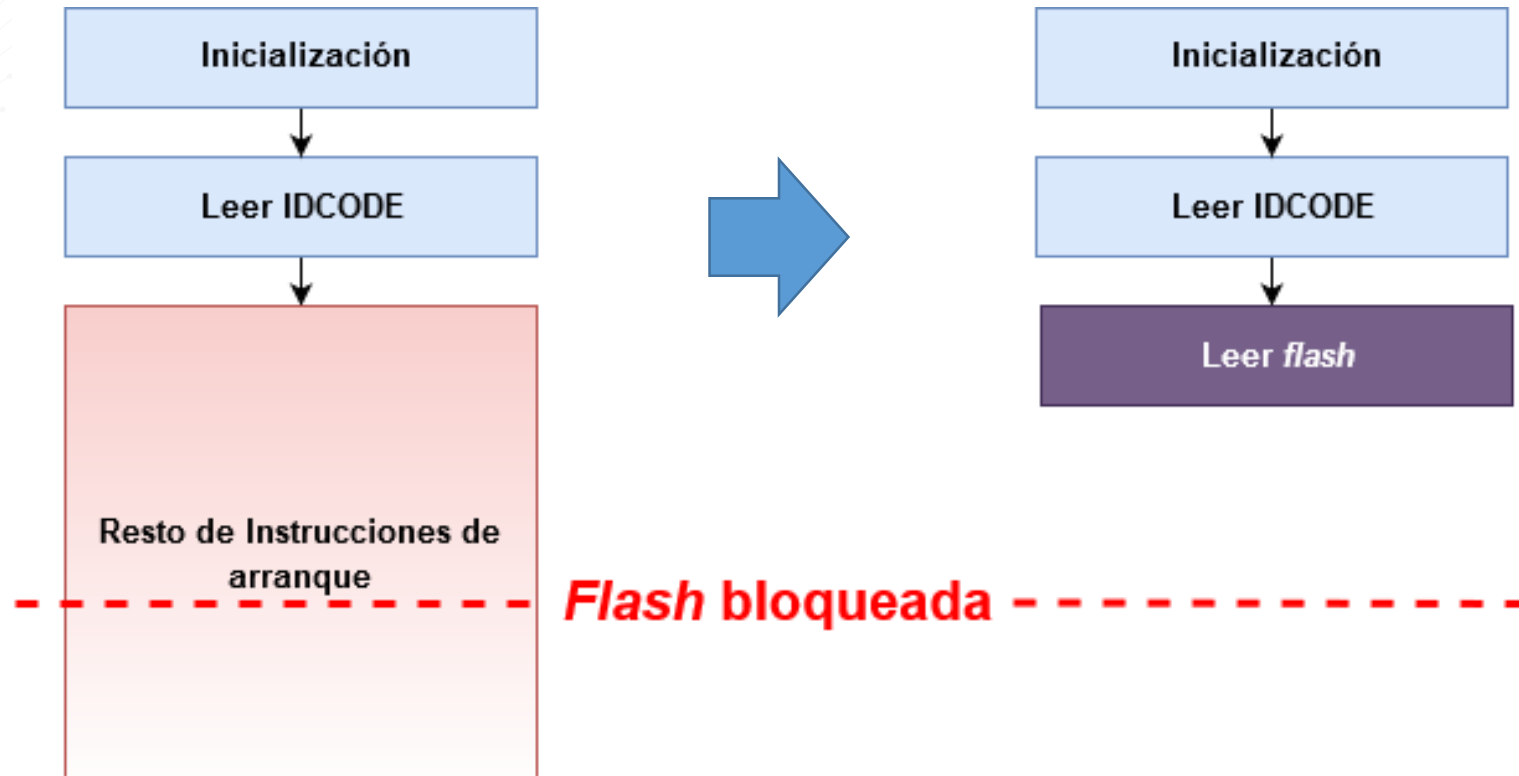
Flash Lock Race

No conocemos los
detalles de la
implementación



2. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race



2. Técnicas de Ataque a la Protección Contra Lectura

Flash Lock Race

- Implementación mínima de protocolo de debug (JTAG/SWD)
- Solicitar acceso a la memoria *flash*
- Condición de carrera, debemos leer la memoria antes de que esta sea bloqueada
 1. Reiniciamos el sistema
 2. Inicializamos la interfaz de *debug*
 3. Establecemos la dirección de memoria que queremos leer
 4. Leemos la memoria
 1. Si tenemos éxito, repetimos el proceso para subsecuentes secciones de memoria
 2. Si no, volver a intentar



Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. ARM Cortex-M0 *Register Readout*

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

6. Inyección de Faltas

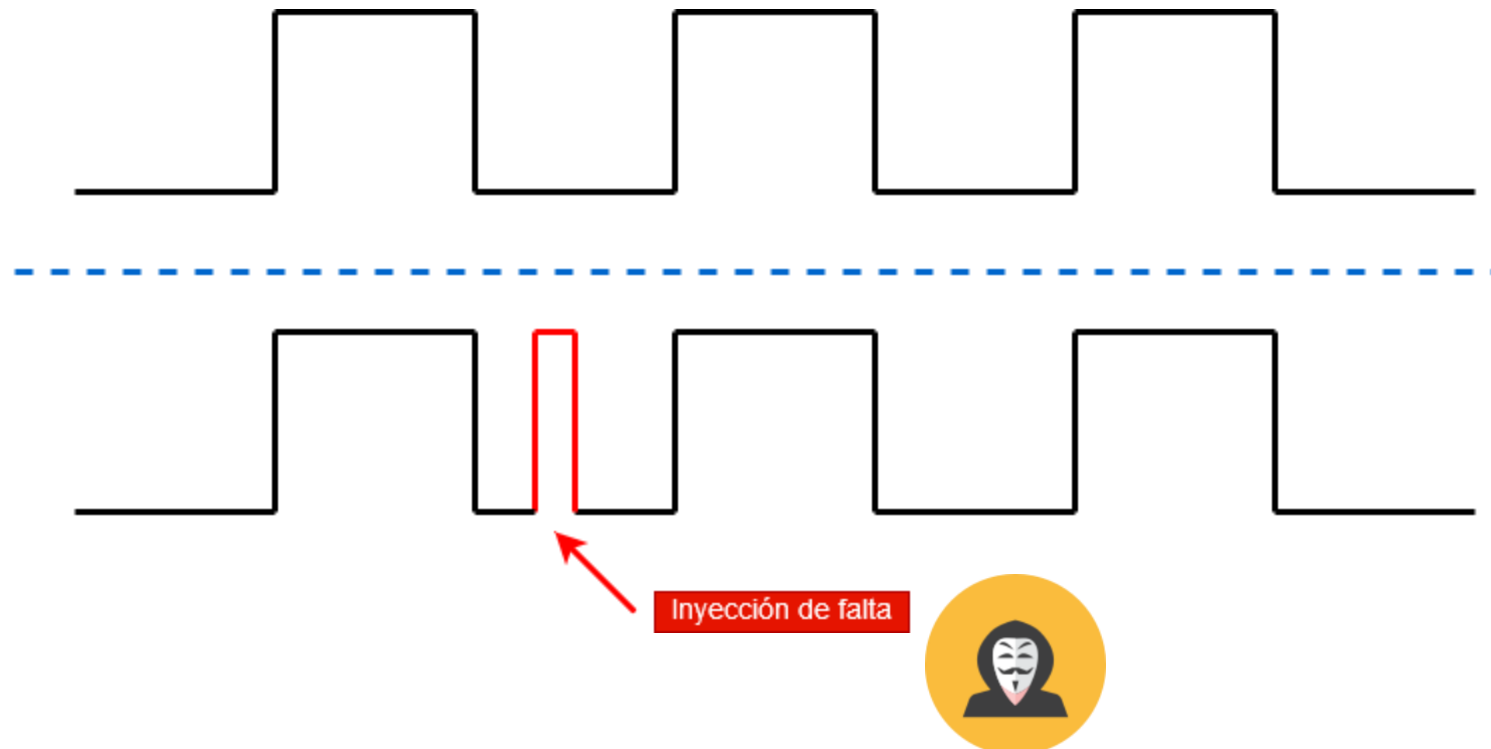
7. *Side Channel Attack*

3. Conclusiones

2. Técnicas de Ataque a la Protección Contra Lectura

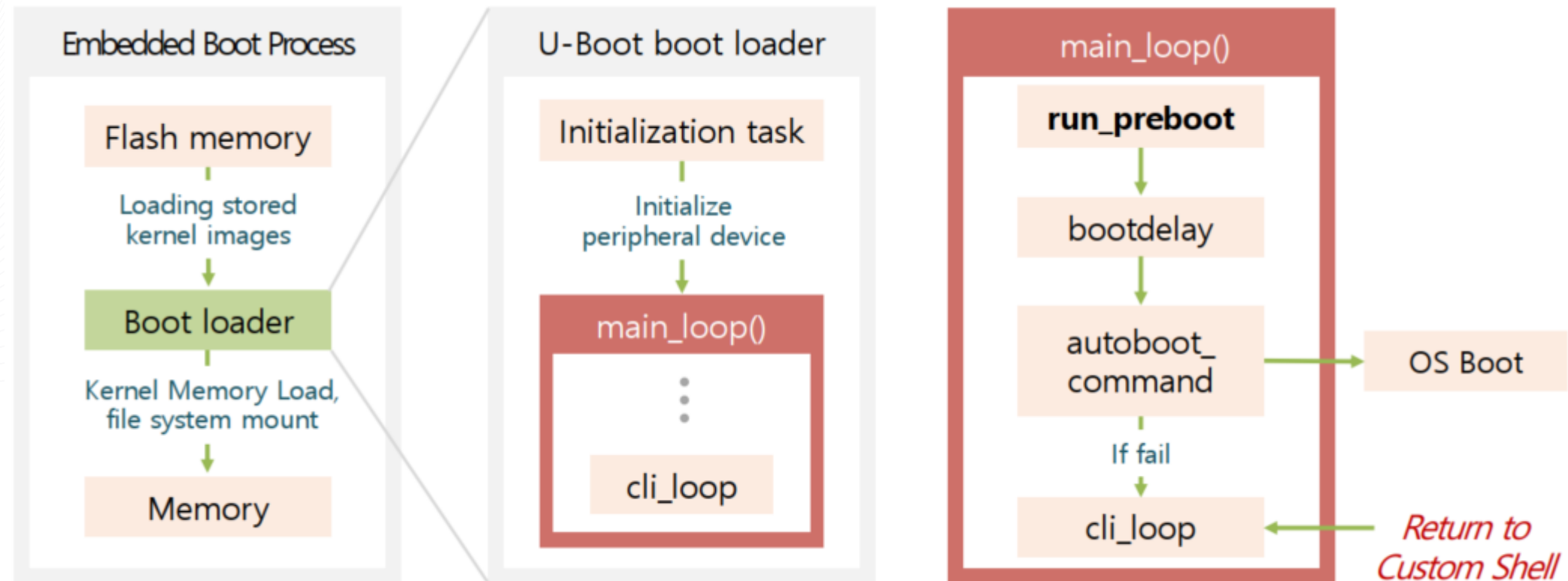
Inyección de Faltas

- El objetivo es enviar un pulso de reloj en un intervalo de tiempo no esperado, causando que se salten instrucciones
- El éxito y el tiempo que se necesita para llevar a cabo este ataque son poco predecibles



2. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas



2. Técnicas de Ataque a la Protección Contra Lectura

Inyección de Faltas

- Si la función `autoboot_command` falla, obtenemos un **Shell**
- Hacemos que falle mediante **inyección de faltas**

Fuente: <https://gitlab.denx.de/u-boot/u-boot/blob/master/common/main.c>

```
39  /* We come here after U-Boot is initialised and ready to process commands */
40  void main_loop(void)
41  {
42      const char *s;
43
44      bootstage_mark_name(BOOTSTAGE_ID_MAIN_LOOP, "main_loop");
45
46      if (IS_ENABLED(CONFIG_VERSION_VARIABLE))
47          env_set("ver", version_string); /* set version variable */
48
49      cli_init();
50
51      if (IS_ENABLED(CONFIG_USE_PREBOOT))
52          run_preboot_environment_command();
53
54      if (IS_ENABLED(CONFIG_UPDATE_TFTP))
55          update_tftp(0UL, NULL, NULL);
56
57      s = bootdelay_process();
58      if (cli_process_fdt(&s))
59          cli_secure_boot_cmd(s);
60
61      autoboot_command(s);
62
63      cli_loop();
64      panic("No CLI available");
65  }
```

Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. ARM Cortex-M0 *Register*

Readout

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

6. Inyección de Faltas

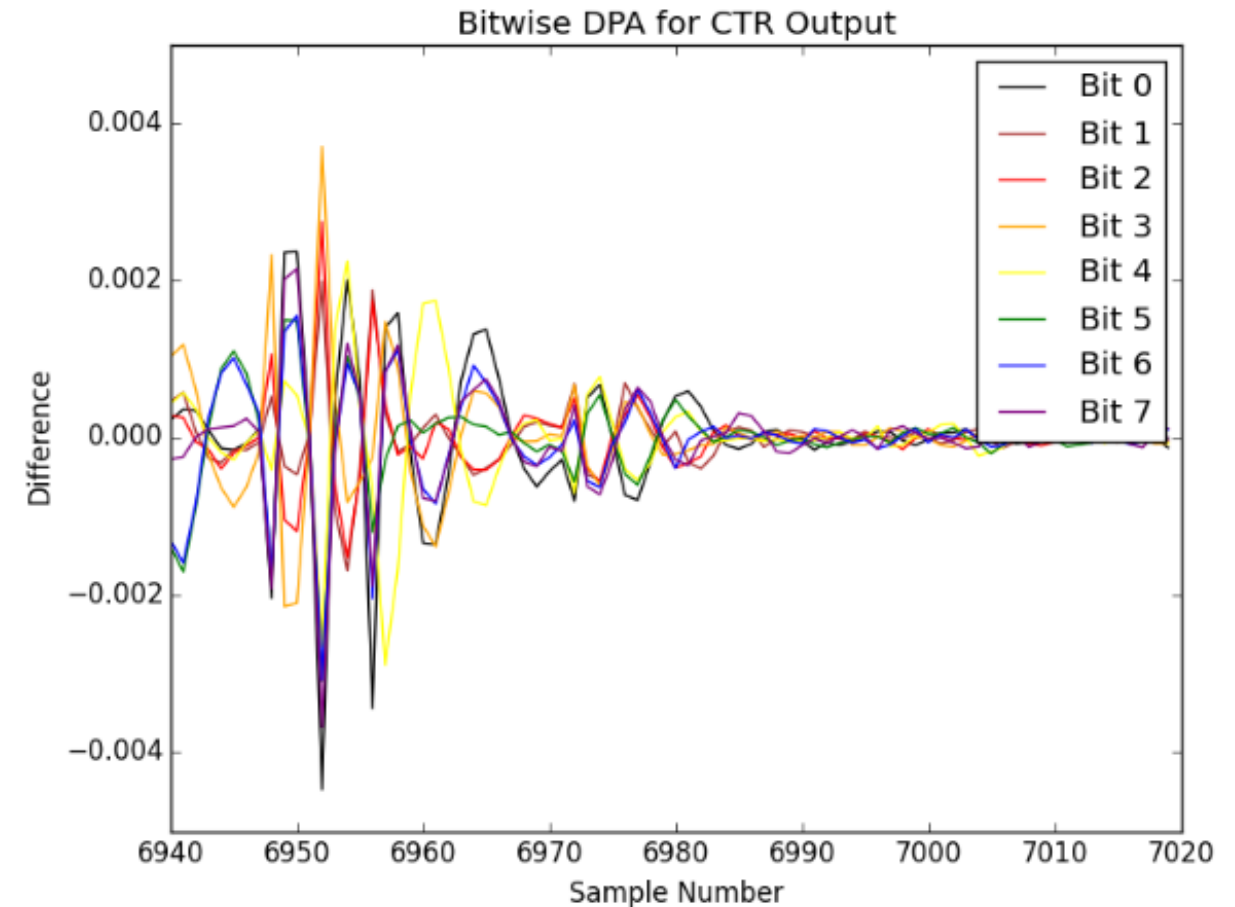
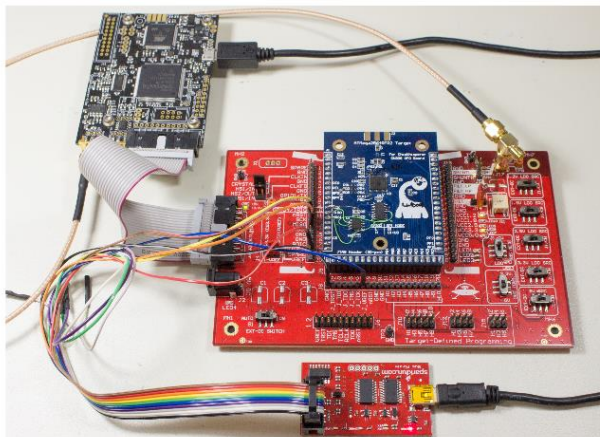
7. *Side Channel Attack*

3. Conclusiones

2. Técnicas de Ataque a la Protección Contra Lectura

Side Channel Attack

- Permite inferir el valor de uno o varios bits
- Realizable con herramientas como osciloscopio o **ChipWhisperer**



Fuente: <https://eprint.iacr.org/2016/1047.pdf>

2. Técnicas de Ataque a la Protección Contra Lectura

Side Channel Attack

- Bombillas con Micro Atmel ATmega2564RFR2
- Se extraen las **claves de cifrado** del *firmware* (AES CCM)
- Podemos inyectar ***firmware modificado***, firmado con las claves extraídas
- Usamos el *firmware* modificado **capaz de propagarse** a otros dispositivos
- Requiere **proximidad física** (protocolo *ZigBee* de corto alcance)
- **Primer gusano puro IoT**



Fuente: <https://eprint.iacr.org/2016/1047.pdf>

Índice

1. Introducción

2. Técnicas de ataque a la protección contra lectura

1. ARM Cortex-M0 *Register*

Readout

2. *Heart of Darkness*

3. *UV-C Security Fuse Erase*

4. *Cold Boot Stepping*

5. *Flash Lock Race*

6. Inyección de Faltas

7. *Side Channel Attack*

3. Conclusiones

3. Conclusiones

- Normalmente no tenemos **acceso físico** a los dispositivos
- Este paradigma cambia con la introducción de las **Smart Cities e IoT**
- Con acceso físico la **superficie de ataque** incrementa considerablemente
- Evitar la **seguridad por oscuridad**
- Diseñar el firmware asumiendo que la protección de lectura **fallará**
- Si aun así hay que guardar **parámetros de seguridad críticos**, utilizar tecnología que sea **capaz de protegerlos**
- **Esos secretos podrían permitir acceder al backend**
- **O propagar el ataque a más dispositivos de la red**



3. Conclusiones

- **OWASP IoT Top 10**
 - ***Lack of physical hardening measures**, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.*
- **ENISA Baseline Security Recommendations for IoT Nov. 20, 2017**
 - *GP-TM-01: Employ a hardware-based immutable **root of trust**.*
 - *GP-TM-02: Use hardware that incorporates **security features** to strengthen the protection and integrity of the device*
- **ENISA Good Practices for Security of IoT - Secure Software Development Lifecycle Nov. 19, 2019**
 - *However, securing IoT, and especially IoT edge-devices, can prove a difficult task for software developers if **hardware comes without basic security capabilities**.*
- **ETSI EN 303 645 Cyber Security for Consumer Internet of Things → ISO → ¿CSA?**
 - *Provision 4.4-1 Devices shall **store sensitive security parameters securely**:*
 - *Secure, trusted storage mechanisms can be used to secure sensitive security parameters, such as those provided by a **Trusted Execution Environment (TEE)** and associated trusted secure storage, **secure element (SE)** and processing capabilities of software.*

3. Conclusiones

- En corto plazo serán **exigidos** por el marco regulatorio europeo: **Cyber Security Act**
- Los **propietarios de los riesgos** deberían plantearse hacer cumplir los estándares
- **Mínimo self-assessment** del estándar, en caso necesario certificación
- **CCN** actúa como tercero de confianza proporcionando servicios de **certificación a través de sus laboratorios** mediante las normas **LINCE** o Common Criteria



**COMUNIDAD Y CONFIANZA,
BASES DE NUESTRA CIBERSEGURIDAD**

#XIIIJORNADASCNCERT

OC.CCN.CNI.ES

WWW.CCN.CNI.ES

WWW.CCN-CERT.CNI.ES

**XIII
JORNADAS
STIC
CCN-CERT**

CCN-cert
centro criptológico nacional