SIGPwny

Embedded FA2025 • 2025-10-13

Embedded PWN

Adarsh and Swetha

Meeting content can be found at sigpwny.com/meetings.

SIGPwny

What is PWN (generally)?

- More descriptive term: **binary exploitation**

- Exploits that abuse the mechanisms behind how compiled code is executed
  - Dealing with what the CPU actually sees and executes on or near the hardware level

- Most modern weaponized/valuable exploits fall under this category

PWN in an Embedded Context

- Dealing with lightweight OS (RTOS, Zephyr, etc.) rather than full-featured OS (Linux, Windows, etc.)

- Similar attacks to “regular” PWN, but dealing with different memory layout

- Utilizing communication protocols (e.g. UART, SPI, I2C, etc.) to get firmware dumps

Memory in Embedded Systems

- Diagram not applicable to all MCUs or embedded systems
- Just for highlighting main sections of embedded system memory
- Main differences from computer memory: flash memory, EEPROM, SRAM

Memory in Embedded Systems

Attribute	Computer Memory	Embedded System Memory
Volatile Memory	DRAM	SRAM
Non-volatile/persistent storage	SSD/HDD	Flash memory & EEPROM
Bootloader	BIOS/UEFI	ROM/Flash

- EEPROM = electrically erasable programmable read-only memory

JTAG and ICSP

- JTAG - low-level communication with MCU for debugging or programing
  - TDI/TDO (Test Data In/Out) - send data in/out of a chip
  - TCK - clock for data
  - TMS (Test Mode Select) - directs state of chip
- ICSP - in-circuit serial programming
- Total and partial JTAG locks
  - Limit debugging and memory access of JTAG interface
  - Depending on system, possible to attack bootloader, use gadget to read from flash memory, extract individual words from flash, read SRAM, etc
  - Even on embedded systems, CPU complexity makes exploitable corner cases more likely

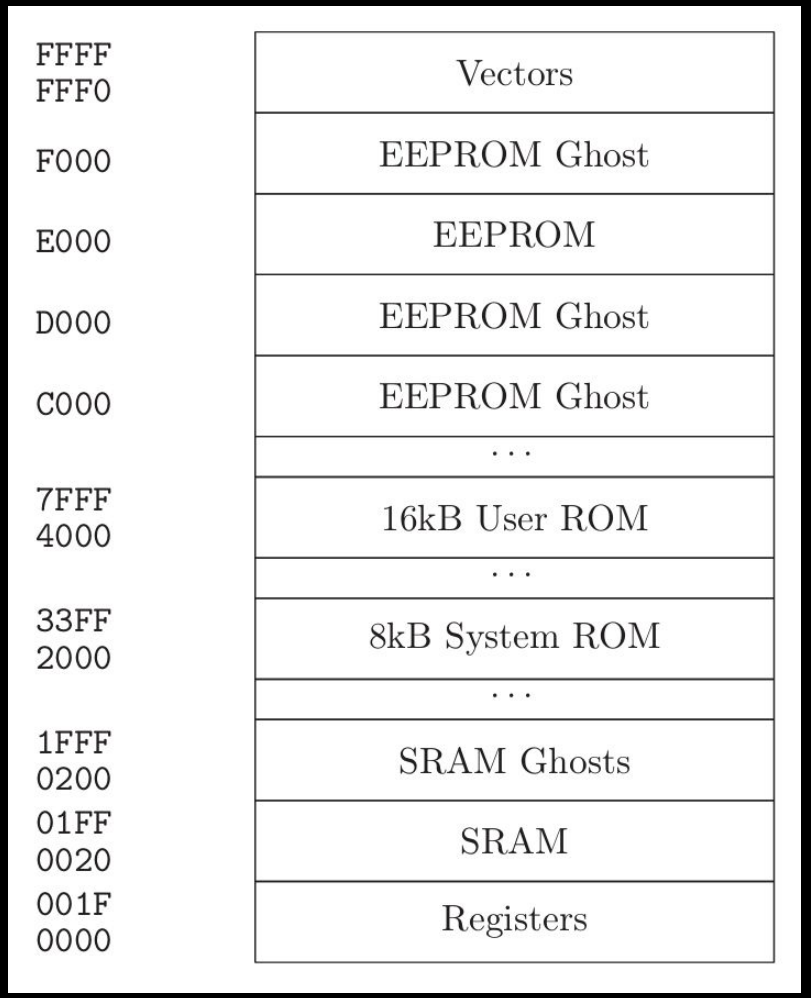
Attacks: Case Studies

Examples taken from *Microcontroller Exploits* by Travis Goodspeed

- Buffer overflow - Dish network smart card (Ch 6)
- SPI bus sniffing - STM32F217 DFU (Ch 2)
- UART access - Card reader (Ch 12)

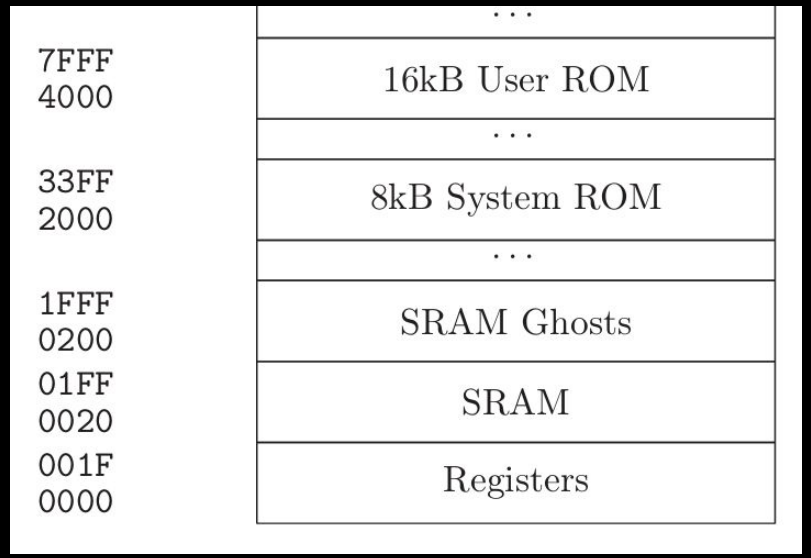
Buffer Overflow: Dish Network Smart Card

- ST16CF54 chip used
- Took advantage of SRAM ghosting property
- Ghosting - copying writes to duplicated chunks of memory
- We're focusing on the SRAM ghosting



Buffer Overflow: Dish Network Smart Card

- Write to 0x0220 is the same as a write to 0x0020 or a write to 0x0420
- Effectively overflow the buffer at 0x019C to write into SRAM Ghost 1
- Ghosting property lets us corrupt actual SRAM



0x0400	SRAM Ghost 2
0x0200	SRAM Ghost 1
0x019C	Target Buffer (in SRAM)
0x0020	SRAM

SPI Bus Sniffing: STM32F217 DFU

- JTAG lets us write an application into unused SRAM
  - Write a program that transmits packets of flash memory via SPI bus

- Executing our program
  - DFU bootloader executes from ROM, and we can set the address pointer
  - We set the address pointer to our application we wrote (using JTAG)
  - Upon exiting the bootloader, execution jumps to the application we wrote

- Reading the data
  - Can read from the SPI bus using a logic analyzer
  - Appending the packets we've read gives us the firmware image

UART Access: HID RW400 Card Reader

- Each memory page has **CP** (Code Protection), **WRT** (Write Protection), **EBT** (Table Read Protection) bits
  - Bits are cleared to enable protections
  - To set bits, you have to erase page
- Meriac (2010) notices that CP and WRT bits are cleared but EBT bits aren't
  - Why aren't all bits cleared?
- EBT bit allows entire firmware to be dumped by code running on a page
- Exploit: erase page, disable all protections, write shellcode to dump firmware through UART
  - Shellcode written by bitbanging ICSP through FTDI GPIO pins

Defenses & Protections in an Embedded Context

- Privilege levels
  - Most MCUs have at least 2 privilege modes - along the lines of “privileged” and “unprivileged” mode
  - ARM - “handler” mode and changeable privileged or unprivileged “thread”mode

- MPU - memory protection unit
  - Enforces privilege levels on regions of memory
  - Can raise memory management faults when violations occur

- Bus-level protection
  - Privilege levels for master and slave signals
  - ARM TrustZone-M → only “secure” masters can access secure regions of memory

Discussion: Potential Exploits on our MCU

M0 Cortex (board being used for eCTF 2026!)

Datasheet: MSPM0L222x, MSPM0L122x Mixed-Signal Microcontrollers datasheet (Rev. A)

What info can you get from the datasheet that would be helpful for exploitation?

Next Meetings

2025-10-18 • **Next Monday**

- Attacking “Secure” Protocols