Reading the STM32 firmware

Almost every microcontroller with integrated flash memory has protection against reading the firmware. This is done to protect intellectual property, cryptographic keys and algorithms from intruders. Microcontrollers of the STM32 series, which have become widespread in recent years, are especially often attacked, however, there is no practical experience or information regarding the protection of STM32 from such attacks is publicly available. In this article, we will consider the firmware protection systems using the STM32f0 series as an example.

Protection concept

Flash Readout Protection (RDP) is a key protection component included in all microcontroller lines. It protects the system firmware stored in the internal flash memory from being read out. Depending on the lineup, additional mechanisms such as Memory Protection Unit (MPU) and privileged / non-privileged execution modes may be included. Together, these systems are designed to enhance security.

RDP has 3 levels of protection, RDP level 0, 1, 2. The security increases with the number.

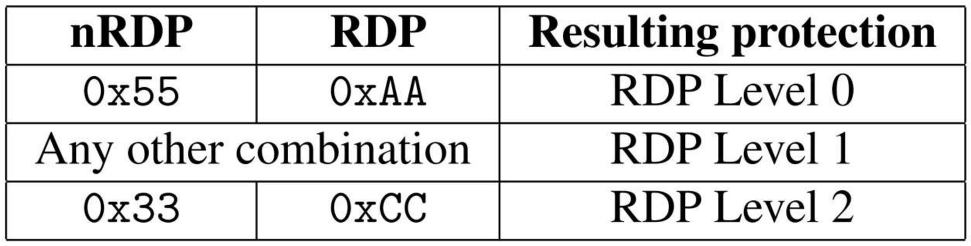
RDP level 0: installed by default and does not offer protection. Using the debug interface, you can get full access to the device.

PRD level 1: The debug interface remains active, but access to the flash is limited. As soon as the debug interface is connected, the flash memory is locked. It cannot be read directly, through DMA, or by executing instructions from it. The protection level can be either raised to 2 or downgraded to 0, with the loss of the contents of the entire flash memory.

PRD level 2: limits as much as possible and provides the maximum level of protection. Debug interface disabled. The level cannot be downgraded. However, despite the highest level of protection, level 1 is widely used. Many companies choose not to block devices completely, assuming the ability to fix bugs and malfunctions, since debugging is not possible at level 2. In addition, the STM32f1 series does not support RDP level 2.

RDP Security Appliance

The RDP level is a part of the microcontroller system configuration, stored in a dedicated option bytes section as 16 bits of non-volatile memory in the form of two registers, RDP and nRDP. nRDP is complementary bitwise to RDP. Redundancy is necessary to protect against level changes by swapping one bit.



nRDP	RDP	Resulting protection
0x55	0xAA	RDP Level 0
Any other combination		RDP Level 1
0x33	0xCC	RDP Level 2

RDP configuration registers

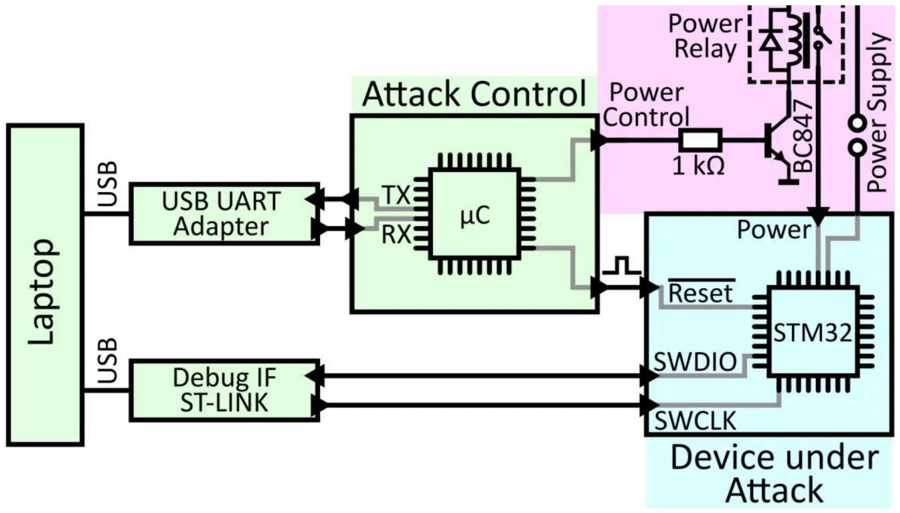
RDP logic

According to the datasheet, there are two execution modes at RDP level 1. User mode and debug mode. As soon as the mk goes into debug mode, access to the flash is blocked. Reading from flash, according to the manufacturer, should cause a bus error, then a Hard Fault interrupt.

Cold-Boot Stepping Attack

In the RDP level 1 mode, when the debugger is connected, access to the FLASH memory is limited only, while the SRAM remains available. We can try to subtract the data at the moment when it is loaded into RAM. Developers of cryptographic libraries are struggling with this vulnerability. Encryption keys are only stored in SRAM during use, which is a few milliseconds, which makes such an attack almost impossible to do, even without the fact that we do not know about the memory organization.

To overcome this limitation, the authors of the article developed Cold-Boot Stepping (CBS), a method by which you can take accurate snapshots of RAM. The idea of the method is to accurately count the time from an event, for example, RESET, and cyclically with a step of several clock cycles to make a snapshot of the SRAM content. The attack consists of the following steps:



Installation diagram for a CBS attack

1. Setting the system back to its original state

1. Power off. It is necessary for the mk to be able to read from flash memory again.
2. Setting RESET before power up. Allows you to start the system without starting code execution
3. Power supply under the set RESET

2. Launching the system in N number of steps.

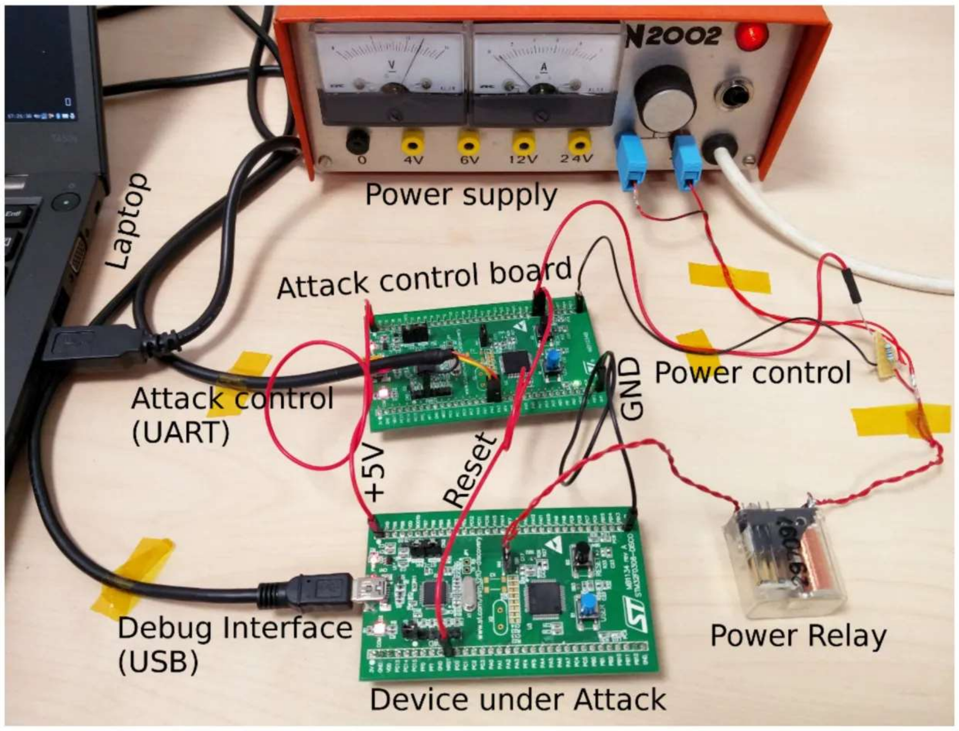
1. Starts the execution of the code by removing RESET
2. Waiting until the execution of the firmware reaches the set place
3. Installing Reset. Stops execution, but data in SRAM remains intact.

3. Reading SRAM content into file

1. Connecting the debugger to the mk
2. Removal of the reset signal. MK does not start code execution, because it is in the halt state set by the debugger.
3. Subtraction of SRAM

By repeating this algorithm as many times as we need, we can get information about the context of program execution. To implement this attack, it is necessary to precisely control the time, which is possible only when using an additional micron.

In many products, manufacturers use a bootloader, the firmware verification algorithm of which is based on calculating the check amount, for example, CRC32, implemented in some MK lines. Using CBS (Cold-Boot stepping) at the stage of the bootloader operation, you can completely restore the firmware by analyzing the registers of the hardware check of the amount or its software implementation, because at a certain step it stores the bytes of the part of the firmware we are interested in.



Installation for a CBS attack

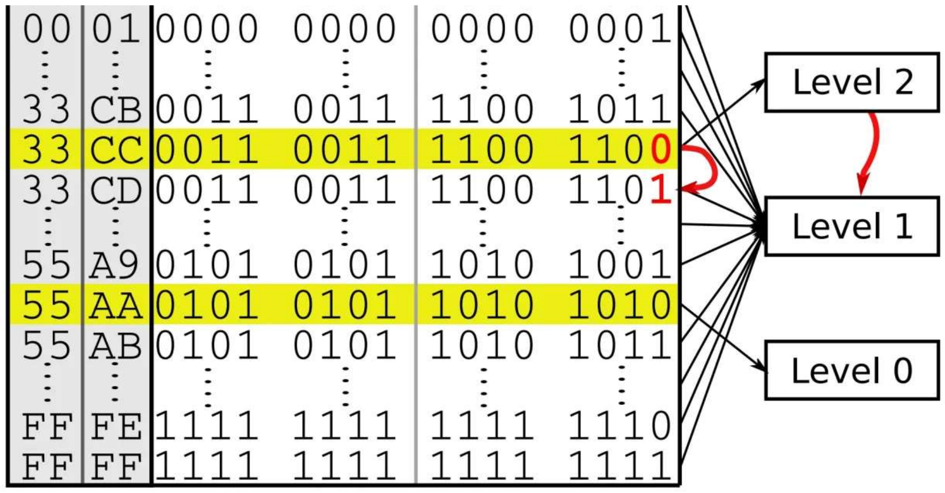
The photo shows a standalone installation for extracting firmware. The laptop dynamically adjusts the step based on the success of the previous step. For mk with a small amount of memory, for example STM32F051R8T6 at 64kb, the extraction will take several days.

It turns out that even though RDP level 1 provides SRAM read protection, it can be compromised.

Using RDP level 2 allows you to protect the device from such attacks, one often manufacturers use RDP level 1. For example, in the popular debugging software, OpenOCD, only provides the "Lock" command to protect the flash memory of the device. However, the command only supports RDP level 1.

Downgrading the level of protection

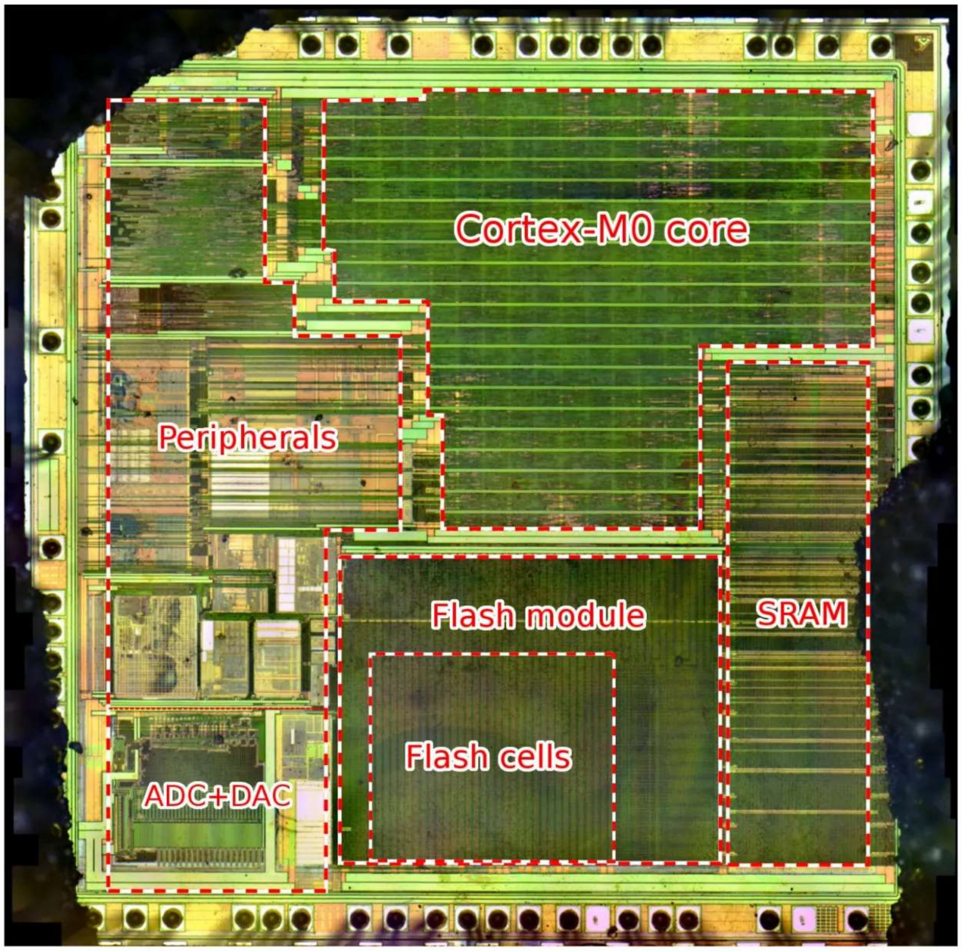
Let's now look at methods for downgrading RDP. The manufacturer claims that setting RDP level 2 is irreversible. Ideally, we need to lower level 2 to 0, but the redundancy of RDP registers requires replacing 8 bits. To downgrade 2-> 1 requires changing only 1 bit.



00	01	0000	0000	0000	0001
⋮	⋮	⋮	⋮	⋮	⋮
33	CB	0011	0011	1100	1011
33	CC	0011	0011	1100	1100
33	CD	0011	0011	1100	1101
⋮	⋮	⋮	⋮	⋮	⋮
55	A9	0101	0101	1010	1001
55	AA	0101	0101	1010	1010
55	AB	0101	0101	1010	1011
⋮	⋮	⋮	⋮	⋮	⋮
FF	FE	1111	1111	1111	1110
FF	FF	1111	1111	1111	1111

RDP State Mapping Table

By UV-C optical exposure, the bit can be changed from "0" to "1". When 254nm radiation hits the gate, electrons are introduced and the state of the logic cell goes from 0 (charged) to 1 (uncharged). You must first clean the crystal using chemical etching.



However, it is required to localize the region of the crystal where the RDP bytes are located. The manufacturer does not document the internal structure of the crystal. Let's write a program that will able to lower the RDP security level without additional mistakenly changed bits.

Protection against under-level protection

There is no protection against downgrading RDP, but you can write your program so that during the initialization phase it checks the value of the RDP and FLASH_OBR bits, which stores the current protection level, as early as possible, and stops execution, making the CBS extraction method useless.

Hacking the debug interface

Microcontrollers from the ST manufacturer assume debugging via the SWD interface [2]… When the debugger connects to the MCU with RDP level 1, the flash protection restricts access. The poorly documented debugging mechanism raises many questions and encourages learning.

The authors of the article have created their own implementation of the SWD interface to study how protection works. It turns out that protection is activated only if the debugger interacts with the AHB-Lite bus [1]… Accessing only the SWD registers, the protection is not activated, but as soon as access to the peripherals is requested, the SRAM or Flash mk goes into debug mode and the flash memory is blocked.

To determine the logic of the protection operation, the authors reduced the number of SWD requests to the required minimum for successful initialization. During initialization, protection is not triggered.

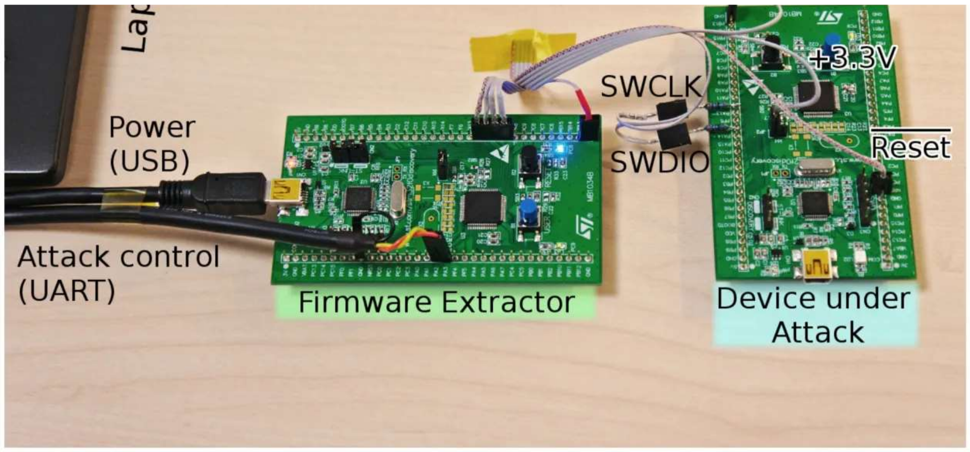
According to the Cortex-M0 documentation [3] processor instructions take precedence over the debug interface. It turns out that the debugger needs to wait for a free cycle on the bus to execute its request. If the debugger gains access to the bus before flash memory protection, it will be able to read data from non-locked memory.
The authors of the article have studied the work of protection with firmware that emulates the load on the bus, which consists of intensive reading and NOP operations. If there are no such operations in the firmware, then the memory read by the debugger takes 2 cycles: address resolution and read directly. If you add one NOP operation, then one of the three read requests will fail. The dependence of the probability of successful reading on the number of NOP operations can be expressed as a formula

$$P_s = 1 - \frac{w}{2+w} = \frac{2}{2+2}$$

Using STR operations as a load will show that flash memory itself controls access. The firmware also blinks very quickly with the LED, and the moment when it stops blinking indicates that the memory is locked and the execution of the code has stopped.

One of the explanations for this vulnerability may be the incorrect implementation of the coordination of the clock source and the rest of the logic.

The authors presented a working implementation of code extraction using two STM32F0 Discovery. The data is sent to the PC via the UART interface, and the SWD is implemented on one of the STMs.



The attack consists of the following steps:

1. Reboot the system by disconnecting and applying power to reset the flash protection.
2. Debug interface initialization.
3. Set the debug word length to 32 bits
4. Setting the read address from flash
5. Attempt to read from memory
6. Reading read data via SWD
7. Repeat until we have read the entire memory by incrementing the address

The average read speed is about 45 bytes per second, which makes it possible to read the most capacious "stone" of 256kb in 2 hours. However, the experiments were carried out only on the STM32F0 series, and it is assumed that due to the similar internal state, all mic lines are susceptible to similar attacks. Other episodes may not be affected.
This attack can be avoided by using the second level of RDP, but as shown earlier, the level of protection can be changed. While the CBS method requires operability of the program code, the vulnerability in the debugger can work in the case of firmware damaged when the RDP level is lowered.

conclusions

The MK STM32F0 series contains a number of vulnerabilities that allow a laboratory with basic equipment to create an installation for reading firmware. The methods can be combined to achieve the best result, or they can be run at RDP level 2.

All materials required, source code and examples are provided by the authors of the article publicly under the MIT license https://science.obermaier-johannes.de/...

Original article: Shedding too much Light on a Microcontroller's Firmware Protection | USENIX

[1] ARM LIMITED. AMBA 3 AHB-Lite Protocol Specification v1.0, 2006.

[2] ARM LIMITED. CoreSight Components Technical Reference Manual, 2009.

[3] ARM LIMITED. Cortex-M0 Technical Reference Manual, 2009.

Shrink RS-485 designs

Ad Texas Instruments

Uy Tín Phục Vụ Tận Tâm

Ghế Massage, Xe Đạp Tập, Máy Mát Xa Chân, Hôm Nay Để Nhận Ưu Đãi Tốt Nhất

⟵ **PREVIOUS**

The Dendy ad we deserve

Texas Instruments
ONE RS-485 TRANSCEIVER FOR ANY NETWORK
Design a single PCB for multiple configurations with the industry's first truly flexible RS-485 transceiver
Learn more

NEXT ⟶

New muses. DIY tools

Similar Posts

then decided to approach from the point of view of neurophysiology and learned to be a coach

acquaintance

Okachi.vn

Uy Tín Phục Vụ Tận Tâm

Leave a Reply

Enter your comment here...

[illegible]