racerxdl / **stm32f0-pico-dump** Public

STM32F0x Protected Firmware Dumper with Raspberry Pi Pico

MIT license

63 stars 6 forks

Star Notifications

Code Issues 1 Pull requests Actions Projects Security Insights

main Go to file

**racerxdl** First commit. Everything works :) ... on Sep 9, 2022 2

View code

README.md

# STM32F0x Protected Firmware Reader with Pi Pico

This is a proof of concept protected firmware extractor which uses a bus race-condition in SWD readouts. For exploiting it you need to avoid "talking too much" with the MCU (like the SWD Probes do) and just get direct to the point. Then it's possible to extract a single 32 bit DWORD from the firmware before the protection mechanism triggers. The proof of concept needs to control both reset and power for the target device, since the read out protection on SWD requires a power-cycle to be reset.

There are several PoC over the internet for this debug mode exploit, but most of them use STM32 with the MBED SDK which makes necessary a ton of stuff to compile. I like simple stuff so I choose platform.io which makes everything to you. I also ported over Raspberry Pi Pico (RP2040) which is more accessible nowadays.

This will **only** work for Level 1 Readout Protection, since it requires SWD active and Level 2 Readout Protection totally blocks it. It might work with other STM32 variants but haven't been tested (if you test a different series than STM32F0x let me know!)

# Building it

For building it you need platform.io installed:

```
pip install platformio
```

Or check it out specifics for different platforms at https://platformio.org/

To build you just run:

```
pio run
```

The firmware will be at `.pio/pico/firmware.uf2`.

You can also use platform.io vscode plugin which allows you do to everything in a GUI environment.

# Usage

The pinouts are defined at include/main.h by default as:

```
#define TARGET_RESET_Pin 27
#define TARGET_PWR_Pin 26
#define SWDIO_Pin 14
#define SWCLK_Pin 15
```

The pico should be able to completely turn off the device by using the `TARGET_PWR_Pin`, if your target board is just the STM32 or it has few stuff on it, you can just use the `TARGET_PWR_Pin` directly as 3.3V power supply for the board. Keep in mind that the current sink of the pico is very low, but should be enough for the attack. If you're unsure, you can use a relay/mosfet to power on/off the target power supply.

If your STM32 target has different than 32KB flash memory, you should also edit in main.cpp the parameter `size`.

After powering everything up, the pico will repeat the message `Send anything to start...` on the serial port, that means it is ready. Press up any key and it will start dumping the content:

```
Send anything to start...
Starting
0x08000000: deadbeef
0x08000004: deadbeef
0x08000008: deadbeef
0x0800000C: deadbeef
0x08000010: deadbeef
0x08000014: deadbeef
0x08000018: deadbeef
```

You can also use the `dump.py` script to dump to a file.

# Disclaimer

This work is heavily based on Johanes Obermaier paper Shedding too much Light on a Microcontroller's Firmware Protection. I also used portions of its Proof of Concept to migrate this to a Raspberry Pi Pico and Platform.IO so I released in the same license as the original PoC: MIT.

## Languages