

 tay đũa xdl / stm32f0-pico-dump Cộng cộng

STM32F0x Trình quét chương trình cơ sở được bảo vệ với Raspberry Pi Pico


 giấy phép MIT

☆ 63 sao 6 đĩa 🍷

☆ Ngôi sao

 Notifications

<> Mã số ⌚ Vấn đề 1 🔗 Yêu cầu kéo ▶ hành động 📁 dự án ⚠ Bảo vệ 📄 Thông tin

 chủ yếu ▾

Đi nộp



racerxdl Cam kết đầu tiên. Làm tất cả mọi việc :) ...

vào ngày 9 tháng 9 năm 2022

 2[View code](#)

☰ README.md

Trình đọc chương trình cơ sở được bảo vệ STM32F0x với Pi Pico

Đây là bằng chứng về khái niệm trích xuất chương trình cơ sở được bảo vệ sử dụng điều kiện đua xe buýt trong phần đọc SWD. Để khai thác nó, bạn cần tránh "nói quá nhiều" với MCU (như SWD Probe đã làm) và chỉ cần đi thẳng vào vấn đề. Sau đó, có thể trích xuất một DWORD 32 bit duy nhất từ phần sụn trước khi kích hoạt cơ chế bảo vệ. Bằng chứng về khái niệm cần kiểm soát cả đặt lại và cấp nguồn cho thiết bị đích, vì bảo vệ đọc ra trên SWD yêu cầu đặt lại chu kỳ nguồn.

Có một số PoC trên internet để khai thác chế độ gỡ lỗi này, nhưng hầu hết trong số chúng sử dụng STM32 với MBED SDK, điều này tạo ra rất nhiều thứ cần thiết để biên dịch. Tôi thích những thứ đơn giản nên tôi chọn platform.io, nền tảng tạo ra mọi thứ cho bạn. Tôi cũng đã chuyển qua Raspberry Pi Pico (RP2040) ngày nay để truy cập hơn.

Điều này sẽ **chỉ** hoạt động đối với Bảo vệ đầu ra cấp 1, vì nó yêu cầu SWD hoạt động và Bảo vệ đầu đọc cấp 2 chặn hoàn toàn nó. Nó có thể hoạt động với các biến thể STM32 khác nhưng chưa được thử nghiệm (nếu bạn thử nghiệm một dòng khác với STM32F0x, hãy cho tôi biết!)

xây dựng nó

Để xây dựng nó, bạn cần cài đặt platform.io:

```
pip install platformio
```

Hoặc xem chi tiết cụ thể cho các nền tảng khác nhau tại <https://platformio.org/>

Để xây dựng, bạn chỉ cần chạy:

```
pio run
```

Phần sụn sẽ ở mức `.pio/pico/firmware.uf2`.

Bạn cũng có thể sử dụng plugin platform.io vscode cho phép bạn thực hiện mọi thứ trong môi trường GUI.

Cách sử dụng

Sơ đồ chân được xác định tại [include/main.h](#) theo mặc định là:

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

Pico có thể tắt hoàn toàn thiết bị bằng cách sử dụng `TARGET_PWR_Pin`, nếu bo mạch mục tiêu của bạn chỉ là STM32 hoặc nó có ít thứ trên đó, bạn chỉ có thể sử dụng

`TARGET_PWR_Pin` trực tiếp làm nguồn điện 3,3V cho bo mạch. Hãy nhớ rằng mức chìm hiện tại của pico là rất thấp, nhưng phải đủ cho cuộc tấn công. Nếu không chắc chắn, bạn có thể sử dụng role/mosfet để bật/tắt nguồn điện mục tiêu.

Nếu mục tiêu STM32 của bạn có bộ nhớ flash khác với 32KB, bạn cũng nên chỉnh sửa tham số trong [main.cpp](#) `size`.

Sau khi bật mọi thứ lên, pico sẽ lặp lại thông báo `Send anything to start...` trên cổng nối tiếp, điều đó có nghĩa là nó đã sẵn sàng. Nhấn bất kỳ phím nào và nó sẽ bắt đầu kết xuất nội dung:

Send anything to start...

Starting

0x08000000: deadbeef

0x08000004: deadbeef

0x08000008: deadbeef

0x0800000C: deadbeef

0x08000010: deadbeef

0x08000014: deadbeef

0x08000018: deadbeef

Bạn cũng có thể sử dụng `dump.py` tập lệnh để kết xuất thành tệp.

từ chối trách nhiệm

Công việc này chủ yếu dựa trên bài viết của Johanes Obermaier [Làm sáng tỏ quá nhiều về Bảo vệ phần sụn của vi điều khiển](#). Tôi cũng đã sử dụng các phần của Proof of Concept để di chuyển phần này sang Raspberry Pi Pico và Platform.IO, vì vậy tôi đã phát hành theo cùng giấy phép với PoC gốc: MIT.

ngôn ngữ

● C 82,9% ● con trăn 8,8% ● C++ 8,3%