SPI CAPTURE

Author: certSIGN

[Publish Date]
European Cyber Security Challenge

1.Initial Write-Up

Description

We managed to capture the SPI communication between a Winbond flash memory and microcontroller with a DSLogic logic analyzer. Unfortunately, there are some glitches on one of the SPI lines because of a loose connection.

Can you deal with those glitches and extract the flag from the attached capture?

2. Challenge specifications

- Category: IoT/HW communication/Binary Encoding
- Difficulty: Hard
- Estimated time: 4h – 6h

3.Questions and answers

What is the flag?

CTF{771A2ACD754746C547520431995EE148906E740B5150F9189EC718C05737784A}

What is the string for SHA256SUM?

SHA256SUM of random data

4.Artefact hashes

FILES	MD5	SHA256
spi_flash_capture_v3.sr	DE54918C395FED855398EF81A1284835	ACA53DF50E75B55B1CAB41FB62B99C761F1BBBE1BF50F5A841B9F033CFE19018
w25q32bv.jpg	2B1BC647D7A1848629328F0939EDB781	06B2CA66BABE6526C93719D515F1D6D32ECBA687D8955E1D54090B33D5F50F91

5.Tools needed

- PulseView
- Python

6.Walkthrough (writeup)

This capture was made with PulseView (also known as Sigrok) and can't be opened by DSLogic analyzer producer software.

The file is an archive and can be extracted. The metadata file contains the name and the version of the software that was used for capture (sigrok version=0.5.2).

Open .sr file in PulseView;

Add SPI flash decoder.

Channel 0 is data (CS - chip select);

Channel 1 is clock (CLK);

Channel 2 is MOSI;

Channel 3 is MISO;

We could also add a protocol decoder for SPI Flash, and we can select chip type Winbond W25Q32BV. We can download the chip datasheet to learn about the communication protocol and command structure.

Most of the protocol is now decoded correctly except some data where there are some glitches on CS channel at the start of the communication and after communication ends. This data can be decoded manually by ignoring the glitches.

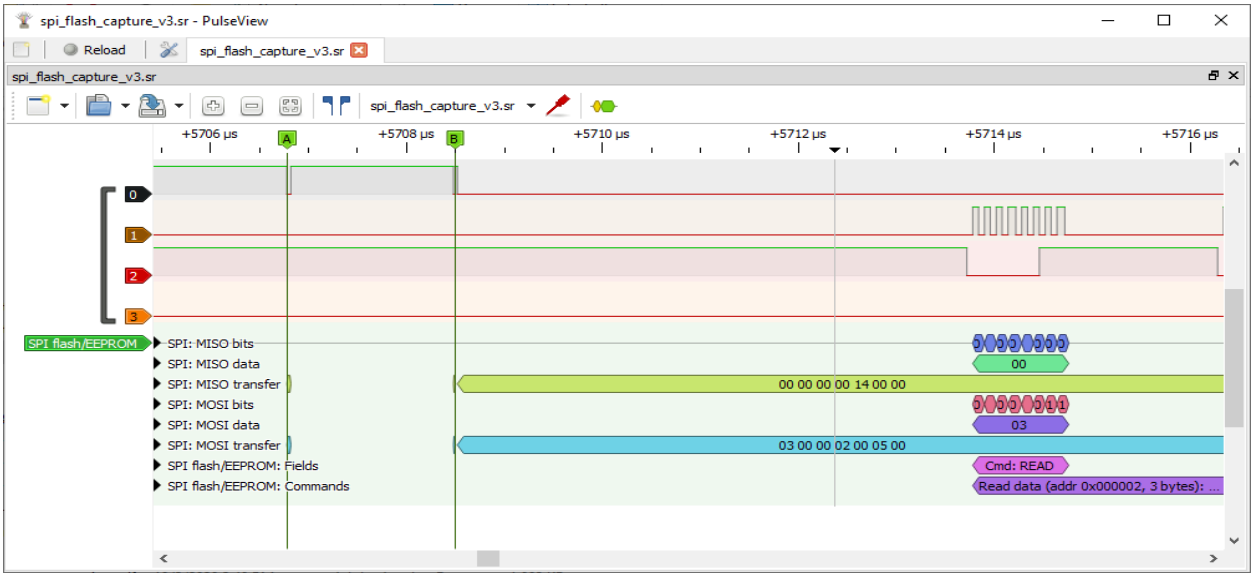
We could also remove the CS Channel from the decoding. In this case we will lose protocol decoding but each byte from the communication will be decoded correctly. In this case we should read the chip datasheet.

Now we can export all annotation for command row in a text file.

We are interested only for read data lines:

279035-279817 SPI flash/EEPROM: Commands: Read data (addr 0x000000, 1 bytes): 44
282166-283340 SPI flash/EEPROM: Commands: Read data (addr 0x000001, 1 bytes): 54
285689-288537 SPI flash/EEPROM: Commands: Read data (addr 0x000002, 3 bytes): 14 00 00
289211-289994 SPI flash/EEPROM: Commands: Read data (addr 0x000003, 1 bytes): 44
292762-297770 SPI flash/EEPROM: Commands: Read data (addr 0x000004, 8 bytes): 24 00 00 00 00 00 00 54
299025-301464 SPI flash/EEPROM: Commands: Read data (addr 0x000006, 3 bytes): 54 00 00
302157-306274 SPI flash/EEPROM: Commands: Read data (addr 0x000007, 8 bytes): 64 00 00 00 00 00 00 24
308470-309389 SPI flash/EEPROM: Commands: Read data (addr 0x000029, 1 bytes): 13

Looking at the data we notice that on the 3rd line are read 3 bytes starting from address 0x03 and the result is 0x14, 0x00, 0x00. On the 4th line is read one byte form address 0x04 and the result is 0x44 but if we consider previous read the byte should be 0x00. Looking at PulseView we notice that this is when the first 2 glitches occur (markers A and B).



Looking at PulseView we can assume that all 1-byte reads are decoded correctly and the rest of them are not. We can write a Python script that extracts the address-data pairs.

We can now extract all the data that was read from the flash. We will notice now that the data is encoded somehow.

Looking at the data we will easily see lots of 3 and 4 on the second byte character in hex and only two that ends with 7 which are 0xB7 and 0xC7. Knowing the flag format, we should have "{" and "}" which are 0x7B and 0x7C. Looks that if swap the hex characters for each byte will we obtain the flag. (Swap first 4 bits with the last 4 bits - little/big endian but on 4 bits).

Starting from the eighth byte we notice that the read is made in a pseudorandom order.

Updating the Python script with this information and sorting by addresses we can put one "*" on the addresses we don't have (glitched reads) and we will obtain a partial flag:

DE*D****CTF{771A2ACD754746C547520431995EE148906E74**5150F9189EC7*8C05737784A}DEADBEEF

We can remove glitched channel CS and manually decode the protocol for the first part of the capture by using the datasheet or the correctly decoded addresses.

We will obtain the full flag:

CTF{771A2ACD754746C547520431995EE148906E740B5150F9189EC718C05737784A}

7. References

https://sigrok.org/wiki/PulseView

https://sigrok.org/wiki/Protocol_decoder:Spiflash

https://www.winbond.com/resource-files/w25q32bv_revi_100413_wo_automotive.pdf