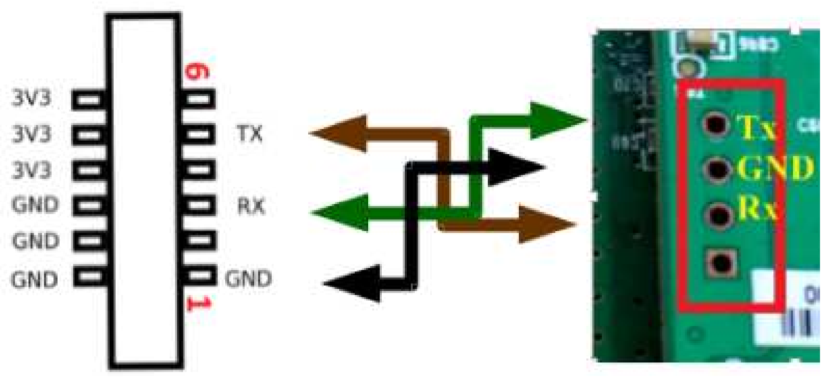
IoT Security - Part 17 (101 - Hardware Attack Surface: UART)

Asmita-Jha

27-September-2020



This blog is part of the IoT Security series, where we discuss the basic concepts about the IoT/IIoT eco-system and its security. If you have not gone through the previous blogs in the series, I will urge you to go through those first. In case you are only interested in hardware UART, feel free to continue.

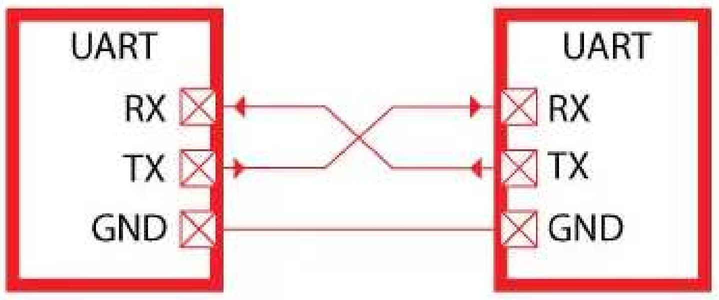
IoT Security - Part 1 (101 - IoT Introduction And Architecture)

IoT Security - Part 16 (101 - Hardware Attack Surface: I2C)

If you are a beginner at hardware hacking and looking for some points to start with, this blog series can help you. As discussed, UART as a hardware attack surface in the IoT attack surface blog here. In this blog, we will discuss it in detail. **Universal Asynchronous Receiver-Transmitter (UART)** is used in most of the embedded devices. While attacking any device, if you can find a UART port on the hardware, hurray!! You got a way to break into the device. In this blog, we will cover the introduction to UART, why is it interesting from a hardware attack perspective, how to interact with it after finding a UART port on the device, and the possible attack scenarios.

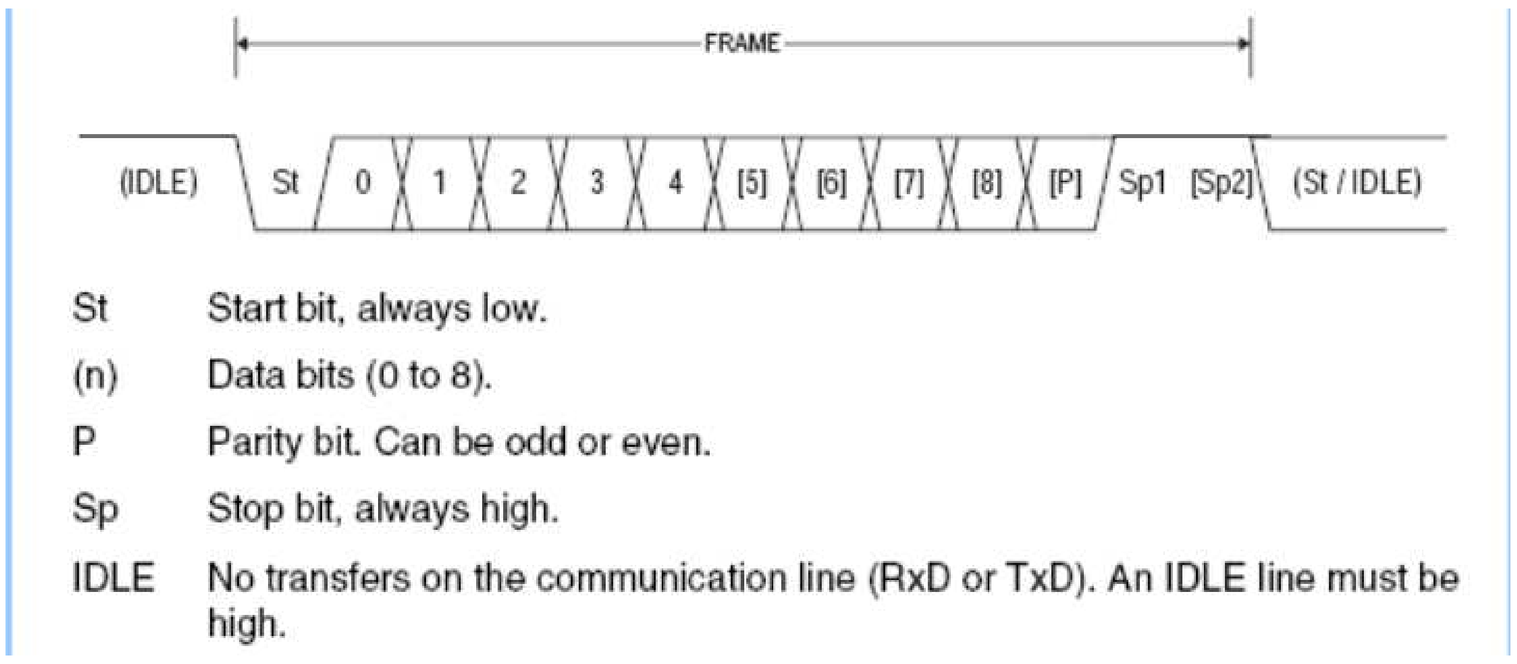
Introduction

UART interface is a hardware device (physical circuit in the controller or a standalone IC) used for asynchronous serial communication. It enables the translation of data between the serial and parallel interfaces using a shift register. It is the most commonly used in embedded devices. The communication directly takes place between two UARTs. The UART interface on the transmitting side takes in the parallel data and coverts it in serial form. It then transmits the data serially to the UART interface on the receiving side. The receiving UART interface takes in the serial data and converts back to the parallel to give it to the receiving side's controlling device. It communicates using two signal lines RX (receiver) and TX (transmitter). The data from the TX pin on the transmitting UART interface is received at the RX pin on the receiving UART interface. The image is shown below (Image Source - here)



During transmission of data packets, START (always logical LOW) and STOP (always logical HIGH) bits are added that defines the start and the end of the data packets. UART communication has one start bit, 5 to 8-bit data (or even 9 bit, if no parity bit used), and 1 stop bit, means 8-bit data transfer once the signal is high to low.

The image for the data frame is shown below (Image Source - here)



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxD or TxD). An IDLE line must be high.

Each data byte to be transferred has a logic low START bit, data bits, optional parity bit, one or more stop bits. In most cases, the least significant (LSB) of the data bit is sent first. The start bit indicates the receiver regarding the arrival of a new data byte, followed by 5 to 9-bit data. An optional parity bit would be present after all data bits, followed by one or more stop bits that are always logic HIGH. STOP bit indicates the receiver regarding the completion of the data byte sent by the transmitter. More detail can be read from here. The rate at which the data bits are transferred i.e., bits per second, is called baudrate. The transmitting and the receiving UART interface should have the same baudrate, data length, parity bit, and stop bit for proper operation.

UART interface is used in microcontrollers to communicate with the peripheral devices. It is used in various communication devices like modems, wifi chips, Bluetooth modules, GPS modules, routers, cameras, and many other applications. UART is also used as an alternative debug interface for the embedded devices that do not have interfaces like a keyboard or monitor to debug. In further sections, we look into possible attack scenarios. We will brief you about the tools and a few methods that you can use to attack the device via UART.

Possible Attack Scenarios

One of the important aspects of why UART port is so interesting from the hardware security perspective is how it is used to connect to the debug shell or console of the embedded operating system of the device. If that is not secured properly, it gives the attacker the path to break into the device by getting access to the root shell.

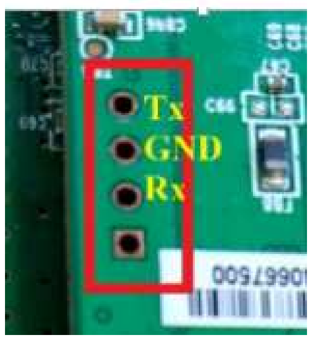
Getting access to the device root shell enables the attacker to (Source - here):

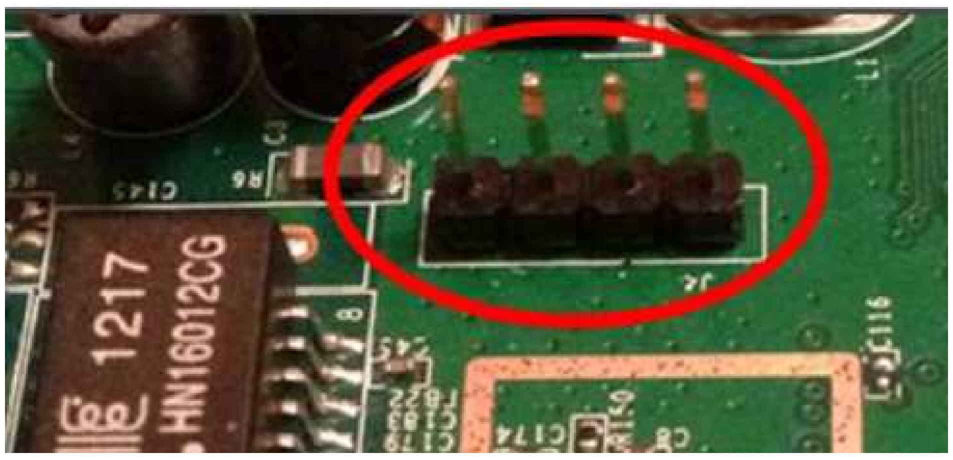
- Hunt around the file system and look for some vulnerable binaries or services running that could be easy to fuzz for an attacker. Sometimes, access to debug or network-related binaries can enable remote access to compromise the device.
- Perform modifications on the running system
- Possibility to flash the patched firmware on the device.
- Explore the file system to find some sensitive and hardcoded values such as encryption keys, configurations, credentials, etc. Getting access to this sensitive info and make it much easier for an attacker to perform further attacks.

Apart from getting access to the root shell, the adversary can sniff the communication between the peripheral device and the controller that is happening via UART serial communication. Also, if device console output is connected to UART, it gives much debugging and system log related info on the console. It can be very useful to the adversary for reverse engineering and knowing about the system's behavior.

Attacking hardware via UART interface

To attack the device via UART port, we first need to identify the UART pins RX, TX on the hardware. Below are the few images showing how most UART port test points/ pinouts on the hardware look.





Source - here

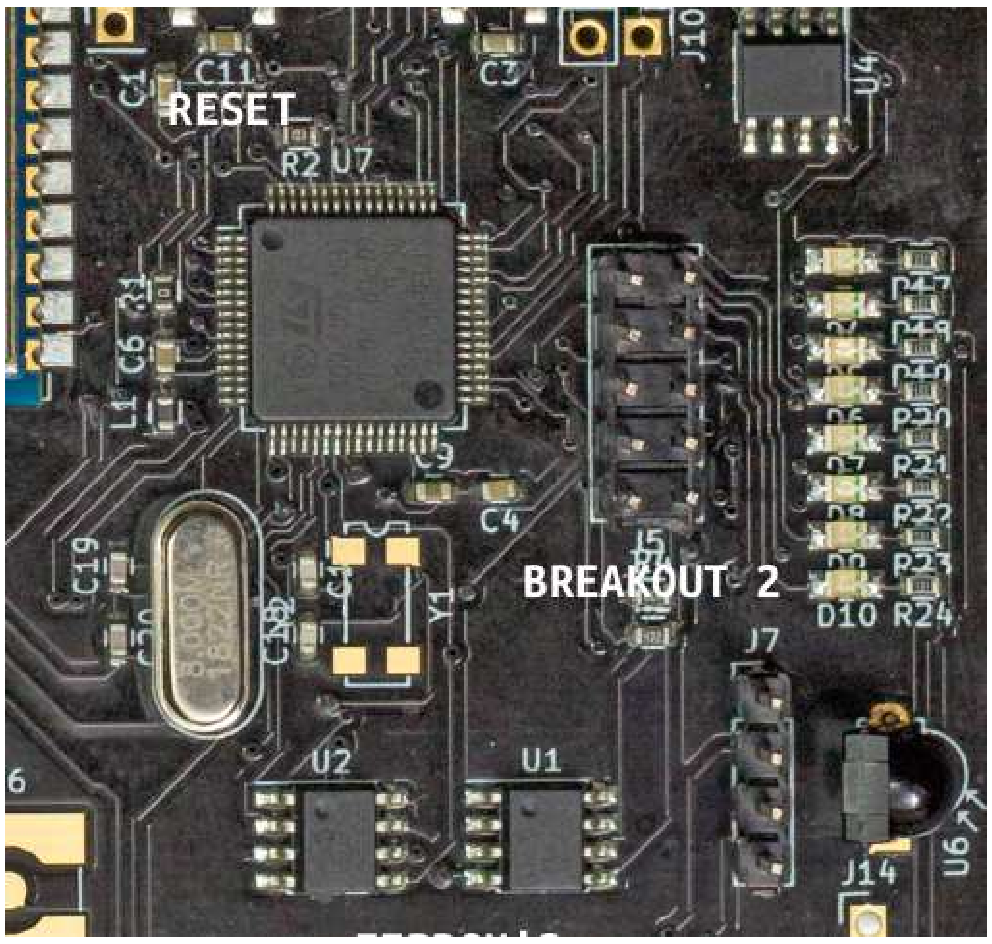
Recon

Case 1: You do not have the actual hardware but you know the FCCID number of the device. Go to the FCCID website, search for the FCCID number of the device. If correct, you will find all the internal images and detailed internal, external information about the device. Seeing the internal image, you may get a hint for devices with UART interfaces on the hardware. Yayy!! Once you know that you have the attack surface, you can get/purchase that device and perform further required steps to attack the device.

Case 2: You got access to the hardware. Once the hardware of a device is found, the first step should be to perform reconnaissance. Inspect each test point and chips present on the printed circuit board (PCB) to look if you can get a UART interface. You can read more about hardware recon here.

As shown in the images above, UART port usually has 4 pins RX, TX, Vcc, and GND. If we find a set of 4 pins together, we hope it is a UART port. Few ways in which you can manually identify the UART port pins on the device are mentioned below :

1. Digital Multimeter conductivity test - Identify the microcontroller used in the device, take out its datasheet, and identify the microcontroller's UART pins. For example, we will take the example of **EXPLIoT DIVA board**.



The microcontroller used in the diva board marked as U7 on the PCB is **STM32F411x**, we can find in its datasheet about the pins having UART interface connection. The image below shows that section of the datasheet.

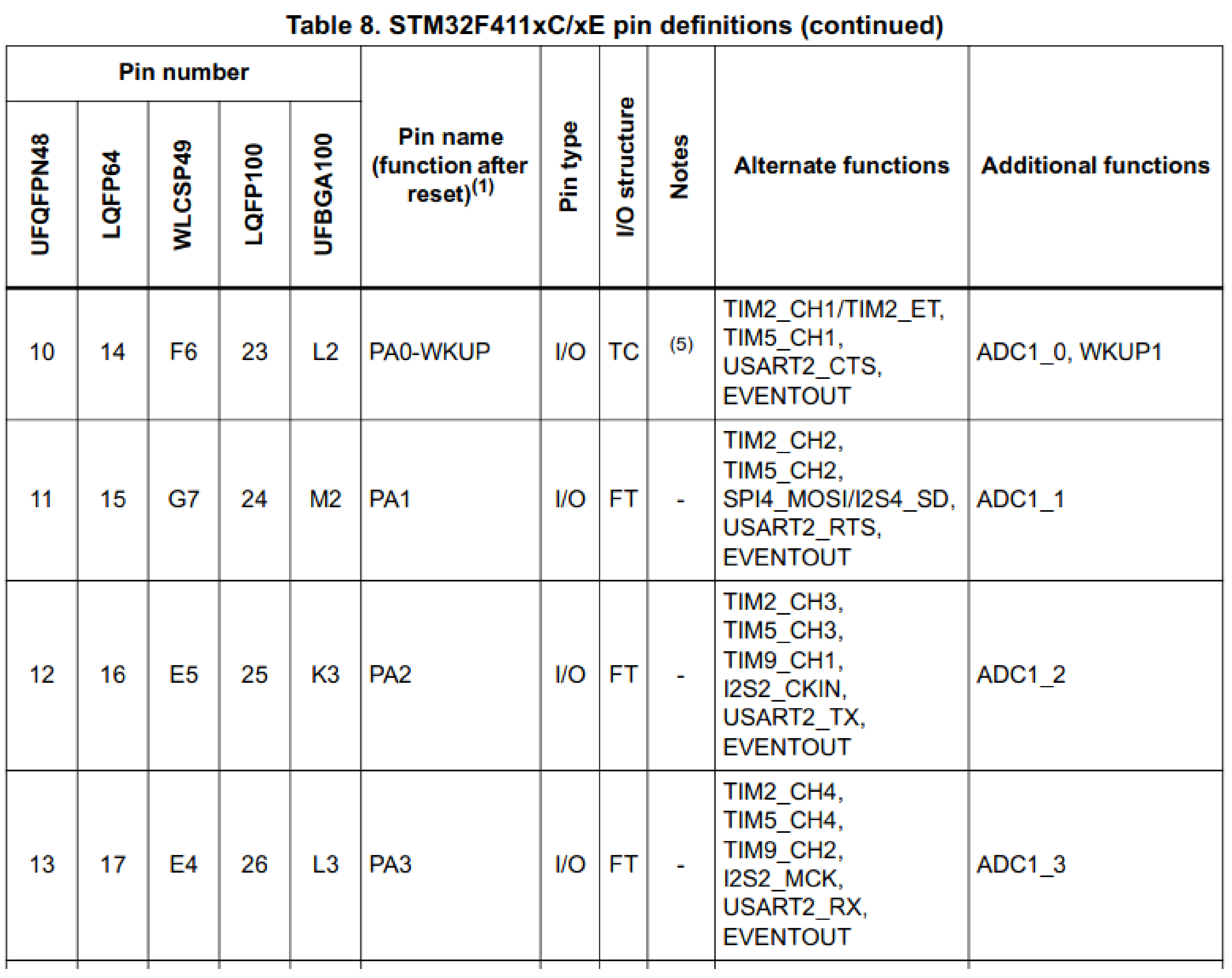
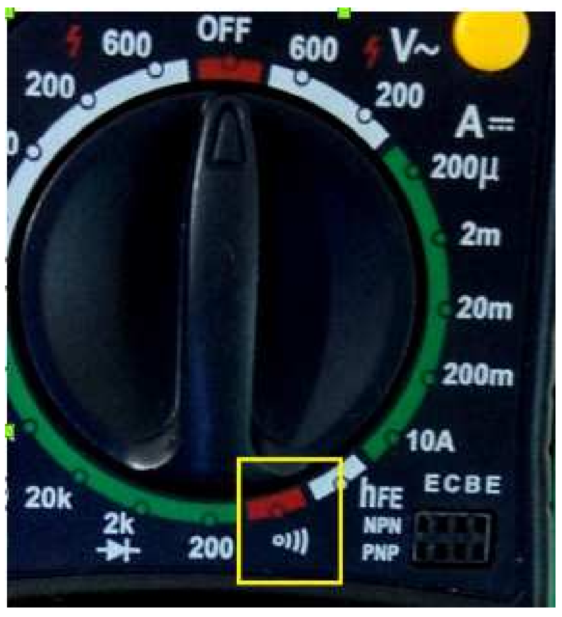


Table 8. STM32F411xC/xE pin definitions (continued)

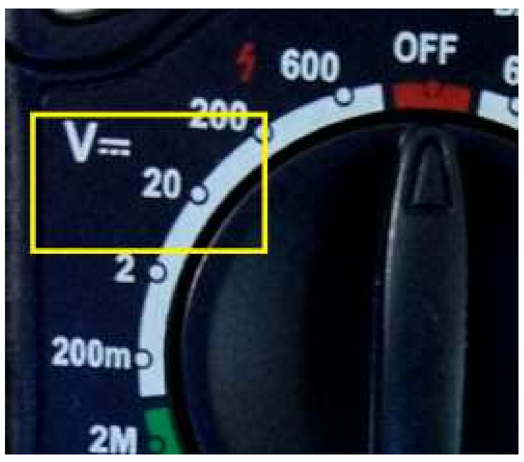
Pin number					Pin name (function after reset)[1]	Pin type	I/O structure	Notes	Alternate functions	Additional functions
UFQFPN48	LQFP64	WLCSP49	LQFP100	UFBGA100						
10	14	F6	23	L2	PA0-WKUP	I/O	TC	(5)	TIM2_CH1/TIM2_ET, TIM5_CH1, USART2_CTS, EVENTOUT	ADC1_0, WKUP1
11	15	G7	24	M2	PA1	I/O	FT	-	TIM2_CH2, TIM5_CH2, SPI4_MOSI/I2S4_SD, USART2_RTS, EVENTOUT	ADC1_1
12	16	E5	25	K3	PA2	I/O	FT	-	TIM2_CH3, TIM5_CH3, TIM9_CH1, I2S2_CKIN, USART2_TX, EVENTOUT	ADC1_2
13	17	E4	26	L3	PA3	I/O	FT	-	TIM2_CH4, TIM5_CH4, TIM9_CH2, I2S2_MCK, USART2_RX, EVENTOUT	ADC1_3

Depending on the IC package that is used in the device (here its LQFP64), we identify the UART port pins on the controller. Then, we set the multimeter in continuity mode as in the image below :



Then, we put one probe on the UART port pins on the controller and the other probe on the test points/ headers pins on the PCB that are suspected to be UART port pins. This test is repeated until pins are identified. Vcc and Ground pins can also be identified similarly.

1. Identify using voltage measurements - In this case, even if we do not have the datasheet of the controller used in the device, we can identify the UART port pins by measuring the voltage on the suspected pins/test points.
    - After opening the device, we look for possible UART port pin/test points groups. Generally, they are four or more pins in the group, as shown in the images above. Mark all such suspected UART port pin groups to test all of them.
    - Starting with identifying the GND pin, set the multimeter in continuity mode. Put the red probe of the multimeter on the suspected ground pin and the other probe on the ground pin of the device's input supply or any metallic shield present on the device. If the multimeter beeps, i.e., if there is continuity between the suspected ground pin and the actual ground pin on the device, it means that is the ground pin that can be used for UART port connections. If not, then repeat with other pins in the suspected UART port pins group.
    - Vcc is not generally used for connecting to the UART interface but identifying it narrows down our search to find the other two pins Rx and Tx.
        - Power on the device
        - Set the multimeter rotary to DC voltage like V (20) (considering device under 20 v range) as in the image below :



    - Put the black probe on the device's ground pin and red probe on the suspected Vcc pin. If the voltage reading on the multimeter shows a constant 3.3v or 5v (depending on the device voltage), that's the Vcc pin. If not, then repeat with other pins in the suspected UART port pins group.

- To identify the Tx pin, power on the device, set the multimeter to dc voltage range as mentioned in the previous point. Put the black probe on the device's ground pin and the red probe on the suspected pin. In the reading on the multimeter shows varying voltage, it means that it is the TX pin. It is because the device keeps transmitting on the TX pins. When the device boots up, it sends the log messages, so the voltage varies when data gets transferred. So, perform this test immediately after powering up the device.
- Generally, if we identify the three pins Tx, Gnd, Vcc successfully, it becomes easier to identify the fourth pin Rx. While measuring voltage as we did in the previous point, it sometimes shows different results in different cases. Sometimes it shows constant low or high voltage or even varying voltage in some cases.

1. Automated Scanning - There are tools to perform automated scanning, like **Bus Auditor** to scan and identifying debugging and communication interfaces exposed on any hardware board. Its demo example in case of UART interface can be read in the **blog here**. **Jtagulator**, **Arduino based UARTFuzz** can also be used to scan for UART port pins. Along with scanning, these tools also detect the baud rate.

UART port interfacing, communication sniffing & device accessing

After identifying the UART port pins on the hardware, now its time to attack.

To sniff the communication

Tools like Logic Analyzer can be used to sniff the communication between the device having the UART interface and the controller.

The logic analyzer shows the signals on the RX and TX lines of UART port. These signals are as per the UART communication protocol structure, as discussed above. Softwares like Saleae Logic Analyzer, PulseView have the feature to detect these UART communication signals that directly shows you the decoded data w.r.t the signals.

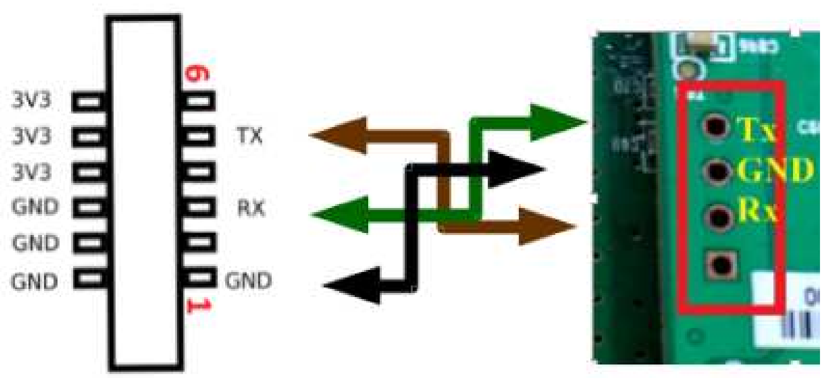
Accessing the device console over UART interface

As discussed above, in most embedded devices, UART port is used as a debug interface. It is connected to the device's serial console that can give access to a root shell, log messages. To access the device's console over UART port, we need an adapter that supports the UART interface. Before using any external tool with the device, we need to match their voltages. Also, while connecting the external tool and the target device, make their grounds common. Few tools that can be used as USB to UART converters for UART interfacing are :

- **Expliot nano**
- **Bus Pirate**

- **Shikra**
- **CH341A**
- **CP2102 USB to TTL covertors**

After the tools are selected, do the proper connections as per the datasheet. For example, here we would show using Expliot nano. We will do the connections as per its datasheet/pinout manual. The connection diagram between the device's UART port pins and the Expliot nano is shown below.



The RX of the device is connected to TX of the adapter and the TX of the device is connected to the RX of the adapter. The ground of both devices is connected. Now, the device's serial console can be accessed using serial terminals like minicom, picocom, putty, and teraterm. Before connecting, we need to know the baudrate. If the UART port scanning is done using the automated tools like bus auditor and jtagulator, we automatically get the baudrate. However, if the UART scanning is done manually using the multimeter or the voltage measurement, we need to find the baudrate. For example, with Expliot nano, we have EXPLIoT Framework that can be used to find UART baudrate using the plugin `run uart.generic.baudscan -p <device_port> -v`. In manual cases, we need to do hit and trials for finding the baudrate.

Once the baudrate is detected, the converter is connected to the host machine via the serial terminals to access the device's serial console. If authentication is not implemented securely in the device, the adversary can directly get into the device after getting access to the root shell. It can be used to attack the device in several ways, as discussed above in the "Possible Attack Scenarios" section. Stay tuned for our future blogs, where we will demonstrate attacks using the above methods in some IoT devices.

Conclusion

In this blog, we learned about the UART interface, its application, the possibilities of attack scenarios, and the methods to attack. WeI hope you enjoyed and got some valuable information out of it. These tools and attack methods would give you a basic understanding of what to do when you find a UART interface on the hardware and play around with it to attack some devices.

Continue to the next part - IoT Security - Part 18 (101 - Hardware Attack Surface: JTAG, SWD)

References

- **https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter**
- **https://cs140e.sergio.bz/notes/lec4/uart-basics.pdf**
- **https://media.blackhat.com/us-13/US-13-Kohlenberg-UART-Thou-Mad-WP.pdf**
- **https://jcjc-dev.com/2016/04/08/reversing-huawei-router-1-find-uart/**