**Deliverable #3: Hardware and Protocol Investigation**

**Project: IoT Vulnerability Research Project**

**Participants:**

* Vikram Kher
* Varin Jaggi
* Tarik Wilkinson
* Patrick Kakooza
* Liza Moody
* Haoda Wang

**Introduction:**

For deliverable 3, we continued working on the hardware analysis and began looking into the firmware of the La View camera to understand how data is transmitted to and from the device. In this phase, we identified the protocols used for the hardware and software components of the device. We pulled information from the datasheets we found in the last phase, which not only discussed the hardware side of the camera, but also provided information on how the camera operated on the software side. The camera uses *UART* protocol which is used for communication, *I2C* protocol which asks/requests information, and an ethernet module for communication with the SoC. The *I2S* interface is also used in the communication and processing of the digital audio in the camera, which can be leveraged to gain access to all digital audio from the camera. This phase of the project was heavily focused on understanding how the LaView camera works from a software standpoint and communicating with Professor Allen to gain access to a lab on campus.

Once we get access to the lab, we will work on utilizing the lab's equipment to test which of the copper pads on the device are active and how SOC is communicating with other components of the camera. We will also use the lab equipment to attempt to uncover what different commands the CPU is sending to other components of the device. Hopefully, these efforts will help us better understand the vulnerabilities of the LaView camera system.

**UART Protocol:**

UART is an acronym for Universal Asynchronous Receiver / Transmitter. The UART is a physical hardware device that serializes data from other components, such as the CPU, for communication purposes. The data is sent asynchronously between two UART devices which means that information does not flow according to a clock cycle. Instead of using a clock cycle, UART devices send “start” and “stop” bits with data to denote their boundary. In addition, UART also sends a parity bit, which can be used to tell if data has been corrupted during its transmission.

UART is a relatively simple hardware device. It contains only one input line (Rx pin) and one output line (Tx pin). Once we get access to a lab, we would like to use a scope device to be able to attempt to read the data sent between the two UART devices. This can be accomplished by looking at the voltages between sent between the two devices and comparing that to the ASCII table. Hopefully, by understanding the commands that are being send between the two devices, we will be able to insert our own commands into the system to control different devices.

**SPI Protocol:**

SPI is an acronym for Serial Peripheral Interface. The biggest benefit of the SPI protocol is that data can be sent in a continuous stream. This is in contrast to UART which sends data in blocks (separated by start and stop bits). SPI also utilizes a clock signal to synchronize data sent across its platform. SPI usually consists of a leader device and a follower device. The leader device, like a CPU, is responsible for controlling the follower device, like a sensor. The leader communicates via the MOSI line, while the follower receives its orders from the MOSI pin.

**I2C Protocol:**

I2C is an acronym short for Inter-Integrated Circuit. Two wires protocol: SDA (Serial Data) and SCL (serial clock). I2C combines the best features of UART and SPI. Is a serialization protocol done on a single SDA line, sending the data synchronously. Generally used for asking and requesting information. Has at least ONE master (leader) that can connect to one or MULTIPLE slave (follower) devices. The leader can send and request to followers. The clock signal is controlled by the leader. Generally less power than other communication protocols. All information is sent through messages using addresses to tell follower data is being sent to it. This message consists consist of:

* Start condition: SDA line switches from high to low voltage BEFORE SCL line switch
* Stop condition: SDA switch low to high voltage AFTER SCL line switch
* Address Frame: 7 to 10 bit sequence unique to each follower identifies which follower leader is speaking to
* Read/Write bit: single bit specifying whether leader is sending data to follower or requesting it
* ACK/NCK bit: if the address frame was successfully received, an ACK returned to the sender.
* Data Frame: the actual information being sent between devices

Each data frame is limited to 8 bits.

If multiple leaders are connected to the same bus, the leader must read SDA and ONLY send if high. (No leader is using it, free).

**NOTE**: can have a rather slow transfer rate than SPI, making it easier to intercept and interpret the data being sent.

**I2S Interface:**

This is the interface used to handle all the audio for the camera. The inter-IC sound (I2S) bus is a serial link developed especially for digital audio. The bus handles only audio data through a simple, 3-line serial bus consisting of:

* Continuous Serial Clock (SCK)
* Word Select (WS)
* Serial Data (SD)

I2S is a 3-wire protocol and we can use it in our project to retrieve audio data from the camera and potentially livestream the audio from the microphone of the camera. The interface protocol is used for the transmission of 2 channel (stereo) Pulse Code Modulation digital data, where each audio sample is sent MSB first. The working modes on our camera for I2S are:

* I2S transmitter slave mode:
* In this mode, the AK3918 accepts the BCLK and LRCLK signals and outputs the data from DAC controller to the external DAC via the I2S interface. The word length ranges from 4 bits to 24 bits, which is configured by DAC controller. If the word length of external DAC is more than 24 bits, 0s are added to the end of LSB.
* I2S receiver slave mode:
  + In this mode, the AK3918 accepts the BCLK signal, LRCLK signal and the data from external ADC via the I2S interface. The word length ranges from 3 bits to 16 bits, which is configured by ADC controller. If the word length of the external ADC is more than 16 bits, the LSBs are ignored.

**Ethernet Module:**

The ATBM603X module communicates with the SoC with a USB2.0 host interface. It utilizes 802.11i security and its operation frequency is 2.412 to 2.462GHz. The 802.11i security protocol is robust and utilizes a 4-way handshake and group key handshake protocol. These protocols help establish the authenticity of cryptographic keys.

**Linux drivers for SOC:**

While searching for the SDK, we found a fork of Linux v. 3.4.35, which included the drivers for the SoC and the camera parts. This was found as part of a different open-source camera project, however from the documentation within the code it seems like it originates from the SoC vendor. As this version of Linux was released in Mar 2013, and at least ~1000 CVEs listed for the kernel since then, the camera is most likely vulnerable to at least one of these exploits.

The vendor wrote custom non-in-tree drivers for the camera and CPU architecture, which are potentially vulnerable since they have not passed the checks typically done before a merge into the mainline Linux kernel. As there are no merge requests to the mainline kernel for these drivers, we also cannot speculate as to what vulnerabilities these drivers might hold. Future work may involve using static analysis techniques to search for vulnerabilities within this kernel that may be exploited.