SWIFT Wireless Fire Alarm System Analysis

Drew Petry (Advisor)
Research Engineer II
Georgia Tech Research Institute
Atlanta, Georgia, United States
drew.petry@gtri.gatech.edu

Aniyah Bussey
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, United States
abussey6@gatech.edu

Manuel Fabregas
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, United States
mfabregas3@gatech.edu

Garrett Brown (Advisor)
Research Scientist I
Georgia Tech Research Institute
Atlanta, Georgia, United States
garrett.brown@gtri.gatech.edu

Daniel Chou
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, United States
dchou33@gatech.edu

Sidney Wright
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, United States
swright@gatech.edu

***Abstract*—Given the prevalence of large commercial buildings and offices, building management systems have scaled to meet the perpetually increasing requirements. One such example is the fire alarm system. Although traditionally wired, newer developments are migrating to wireless infrastructure to support convenient expansion of sensors and pull stations. However, a byproduct of wireless advancement is the addition of new attack vectors to exploit fire systems. An experienced malicious actor could gain control of a building's fire alarm through wireless means, and they could remotely spoof and suppress alarms at will, leaving those in the building in a dangerous situation. This ongoing study analyzes the vulnerabilities of Honeywell's Smart Wireless Integrated Fire Technology (SWIFT) system, which integrates wired and wireless communication by using a gateway communicating with smoke detectors, pull stations, and other addressable fire devices.**

I. System Introduction

The FACP, or Fire Alarm Control Panel, is a wired component of any fire alarm system, and used to control the functions of other systems in the fire alarm system, as well as receive information from all wired devices in the system.

The SWIFT system is a commercial wireless fire detection system that uses a robust mesh network to integrate the SWIFT gateway and the wireless devices into an existing wired system [1]. The wireless gateway is the SWIFT device that bridges the gap between wired and wireless devices. Thus, it is the main target for this project, as it controls the mesh network, which consists of wireless devices, by managing its formation and configuration while also interfacing the wireless mesh network with the wired network. In traditional systems, all components communicate over a wire which is daisy-chained through all devices. This wire is known as the Signaling Line Circuit, or the SLC. The FACP the team analyzes is made by Honeywell and communication is provided to the control panel by its SWIFT devices.

The gateway has two processors which communicate with each other through a universal asynchronous receiver-transmitter, or UART channel; these are the SLC and RF (radio frequency) processors, whose main functions are to interface with the wired devices, and handle wireless communication and initialize the bootup process, respectively. The gateway has three firmware update files; they contain the bootloader firmware, the RF firmware, and the SLC firmware [2]. The team uses Ghidra and other software to reverse engineer the previously mentioned firmware files to work towards the primary goal of identifying methods of gaining control of the system.

The SLC chip is one of the primary components of the gateway that the team is analyzing, and is outlined in blue in figure 1. This processor's main responsibility is interfacing with the physical SLC line, whose primary purposes are to carry signals between components and provide power for the addressable device modules in the fire alarm system. The signals sent on the SLC line both report device status and give instructions to the devices on the panel. For the purposes of this project, the team is trying to reverse-engineer the gateway's SLC chip firmware. Doing so can reveal details about the specific instructions and commands the SLC chip sends to the fire alarm system so that the team can learn from it and eventually use that information to control the system.

The RF chip is outlined in orange as seen in figure 1, and it is responsible for managing the mesh network that contains the wireless devices, relaying wireless device information to the SLC chip for communication with the FACP, and updating both itself and the SLC processor when the gateway's firmware is updated.

II. Previous Research

Earlier research led the team to discover an authentication bypass that could allow unauthorized parties to gain access to the system. By default, once the gateway is locked, a new password must be created to unlock the gateway. Additionally, a Hall effect sensor is used to check the presence of a physical user. Sniffing the connection between the SWIFT tools and the gateway using Serial Port Monitor led the team to discover the

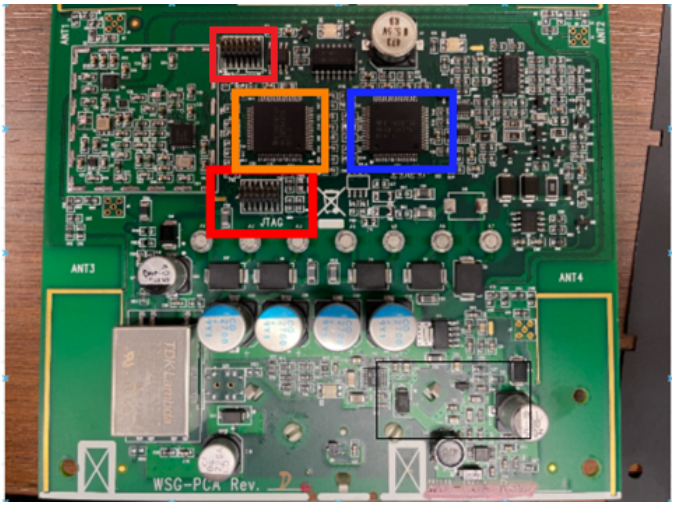

Fig. 1. The wireless gateway board, with various components outlined, such as the RF (orange) and SLC (blue) processors, as well as the two JTAG headers (red).

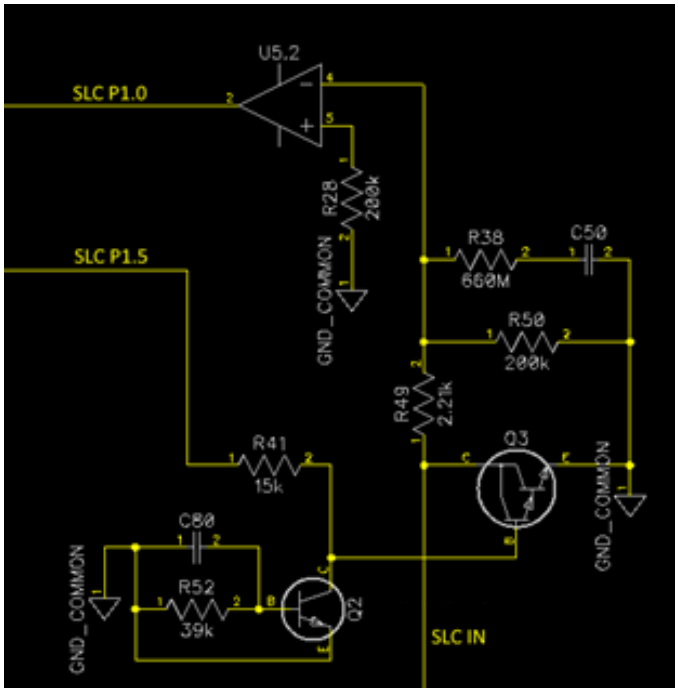

Fig. 2. Schematic showing how SLC chip pins P1.0 and P1.5 are connected to the SLC line

general authentication flow, which revealed that it is possible to unlock the gateway without the use of a password by simply using a part of the authentication process. However, this attack still requires the physical presence check to succeed [3]. In turn, this requirement can also be circumvented; by placing the gateway into bootloader mode and issuing a reboot command packet, the gateway can be unlocked. Entering bootloader mode does not require physical presence authentication, and once completed, an attacker or other unauthorized party has access to upgrade or downgrade the firmware [3].

The team further verified that it is possible to upload custom firmware on SWIFT's gateway. The SWIFT system performs firmware integrity checks using a cyclic redundancy check (CRC) algorithm when a new firmware is uploaded. Once the firmware is uploaded, it compares its CRC with a hard-coded CRC value within the firmware image to verify its integrity. The team created a tool that mimics the CRC calculation for any arbitrary firmware, the result of which can be inserted into the firmware itself, guaranteeing that the integrity check passes when uploaded to the gateway. Ultimately, this allows any unauthorized party to replace the gateway's firmware with any arbitrary image [4].

The team had attempted to reverse-engineer the SLC processor's firmware with two goals in mind: to understand the SLC wire protocol, and to send arbitrary SLC messages using the previous vulnerability that allows attackers to upload modified firmware into the gateway. Through continuity testing, the team identified pins 1 and 3 on the upper JTAG header as being linked to the universal serial communication interface (USCI) pins of the SLC processor, specifically, those for USCI port A1, which are used to expose the SLC chip's debugging output [5]. The team identified all necessary pins for debugging the SLC processor, permitting dynamic analysis of the SLC chip and improving the team's capability to decode the SLC protocol by examining the SLC messages as they are created and sent. Bit 5 of the SLC chip's byte-wide general-purpose IO (GPIO) port 1 (pin 1.5) was found to be the likely method used by the gateway to send SLC messages. The SLC port 1.5 pin is configured during the SLC chip's "initialization" function as output and is connected to the base of a Darlington transistor, whose collector is connected to the shared SLC_OUT wire, and whose emitter is connected to the SLC ground. When the pin asserts a logic high, it will short the SLC line to ground. Since SLC messages consist of logic lows, this constitutes the SLC message-sending capabilities to the gateway [5].

III. Gateway Interaction with SLC

A. Communication Pins

As the team had established previously [5], the SLC chip is able to send messages on the SLC line through bit 5 of port 1 (P1.5). However, the question of how it could receive messages remained open. Continuity tests done by the team revealed that the SLC line going into the gateway is also taken as input to a high-voltage comparator, whose output leads directly to bit 0 of the SLC chip's port 1 (P1.0), as seen in figure 2. Logic captures taken of that pin, as seen in figure 3, confirm that the comparator effectively acts as a step-down transformer for the SLC line, in addition to inverting the signal, as SLC messages are transmitted by shorting the loop [6].

B. Circuit Analysis: Comparator Inputs

Previously, the team focused on the U5 comparator chip, which is used to draw 60 milliamperes to drain current over a wide range of voltages [7]. The comparator is outlined in figure 5 in a red box. During this previous research, the team

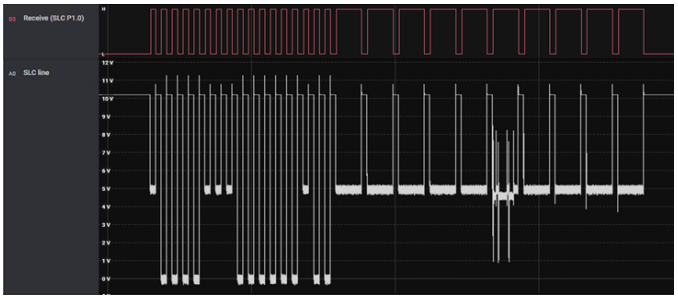

Fig. 3. Logic capture of SLC chip pin P1.0 and SLC line, confirming use of P1.0 as SLC input for SLC processor

was able to find out that the comparator's output pin connects to the SLC P1.7 pin. Because of this, the team established that the comparator chip's input pins' paths must be found. To do this, continuity tests using a multimeter were conducted. Initial tests revealed that the top input pin, labeled with 1IN- as seen in figure 5, was connected to the R38 resistor, while the bottom input pin, labeled with 1IN+ in figure 5, is connected to the R31 resistor. Upon further testing, the team discovered that the comparator's 1IN- input connects to the SLC input via a previously discovered path, and this path is outlined in lime green from the input pin to the R38 resistor, and hot pink from the R38 resistor to the SLC input wire, both of which are outlined in figure 5. After conducting more continuity tests on the 1IN+ pin, the team discovered that the path, which is outlined in maroon in figure 5, leads to ground. Therefore, the SLC P1.7 path seems to perform the same comparison between the SLC and ground as the SLC chip's port 1 (P1.0), so the two pins appear to serve the same purpose. To further corroborate the team's findings from both the current and last semester, logic captures were conducted on the SLC chip's pins and U5 comparator chip's pins that connected to paths that the team identified. More specifically, Logic Analyzer was used to confirm the cyan/blue path identified and was done by conducting a logic capture on the SLC chip's P1.0 pin and the U5 comparator chip's 2OUT pin. As seen in figure 6, because the captures are the same, the identified path is correct. This process was repeated when the team confirmed the lime green/pink path by using Logic Analyzer

C. Logic Analysis

To further corroborate the team's findings from both the current and last semester, logic captures were conducted on the SLC chip's pins and U5 comparator chip's pins that connected to paths that the team identified. More specifically, Logic Analyzer was used to confirm the cyan/blue path identified and was done by conducting a logic capture on the SLC chip's P1.0 pin and the U5 comparator chip's 2OUT pin. As seen in figure 6, because the captures are the same, the identified path is correct. This process was repeated when the team confirmed the lime green/pink path by using Logic Analyzer. Since the captures between the U5 comparator chip's 1IN- and 2IN- pins are the same as seen in the figure 7, the team was able to confirm the identified path.

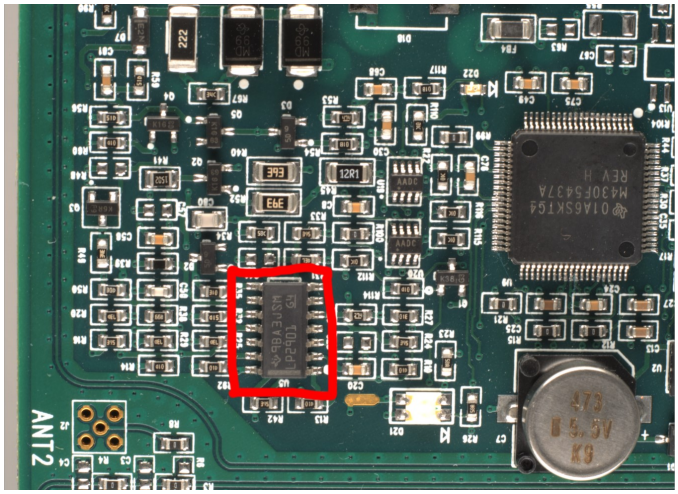

Fig. 4. Zoomed in capture of the gateway, with the U5 comparator chip outlined in red

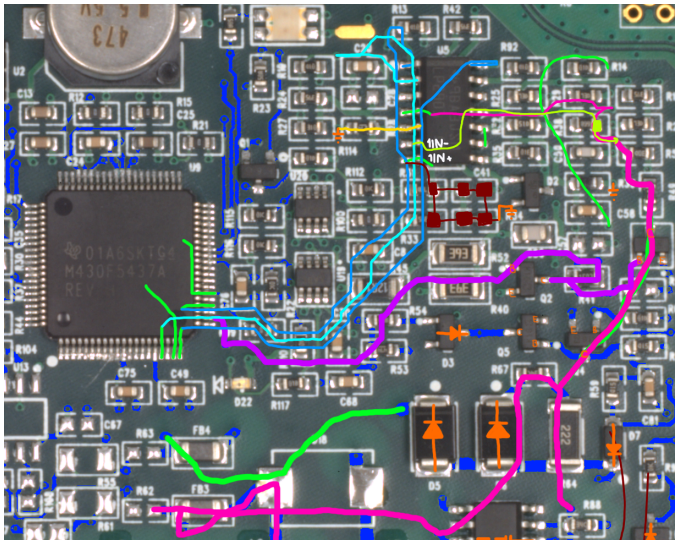

Fig. 5. The U5 comparator chip's input paths are outlined in the figure above. The 1IN- input is outlined in lime green, and it connects to the path of the SLC input, which is outlined in hot pink. The 1IN+ input path is outlined in maroon and connects to ground.)

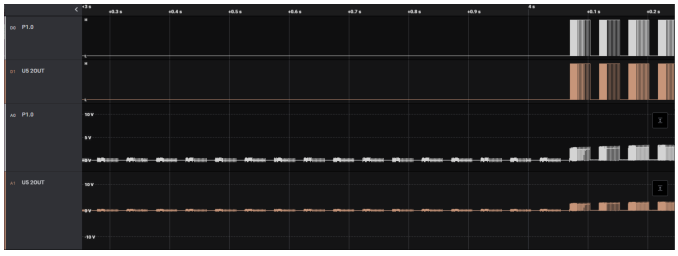

Fig. 6. The path between the U5 comparator chip's 2OUT pin and the SLC chip's P1.0 pin is confirmed with the above logic capture. Because the digital and analog logic readings are the same, we know the blue and cyan path the team identified is correct.

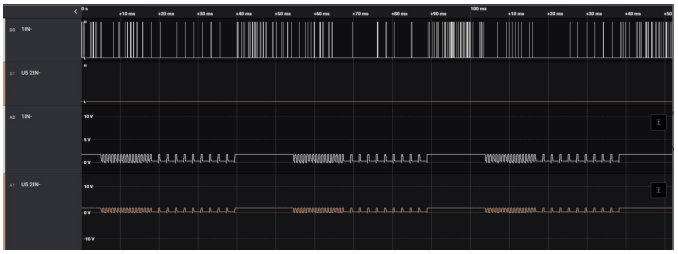

Fig. 7. The path between the U5 comparator chip's 1In- and 2In- pins is confirmed with the above logic capture. Because the digital and analog logic readings are the same, we know the lime green and path the team identified is correct.

D. SLC Message Processing

With methods for both SLC input and output known, the team was able to start focusing effort on reverse engineering the structure of SLC messages. First, the team utilized Ghidra's cross-reference feature to find references in the SLC firmware binary to the two pins of note. While there are several writes to P1.5, there is only one read from P1.0, and during normal operation, all accesses to those pins ultimately derive from two functions: one at address 0x5cd2, and the other at 0x5d62. Though these are not called directly, their addresses are located in a large jump table that is itself referenced in the chip's interrupt vector table as documented in the processor datasheet [8], as the handlers for interrupts from port 1 and timer A0, respectively.

To further the goal of understanding the SLC message structure, the team prioritized reverse engineering the message reception functions over those for transmission. The one reference to P1.0 is in a function at address 0x2b106, which reads the value at that pin multiple times, waiting for two possible time intervals between each read, and ultimately returns a boolean value. The values returned are both saved directly into memory and also used to determine different control flows in the rest of the handler, so the team's current hypothesis is that multiple shorts of the SLC line constitute one logical bit of SLC message, which is level of abstraction that the gateway deals with. This would be able to explain previous confusion about logic captures of the SLC during both idle and alarm states appearing to be identical; if multiple transitions correspond to one logical bit, then the identical capture patterns are simply the same bit, instead of the same message, as previously assumed.

Returning to the interrupt handler functionality, the port 1 handler seems to implement a finite state machine (FSM), with each logical bit of SLC message being its own state. On every port 1 interrupt, which happens whenever those interrupts are enabled and any of the 8 pins which make up port 1 is driven high, the handler will increment a global counter representing the bit's number before getting the logical SLC bit, then doing some work that depends on the bit number and the previous bits of the message. This setup logically follows from the serialized nature of the SLC messages.

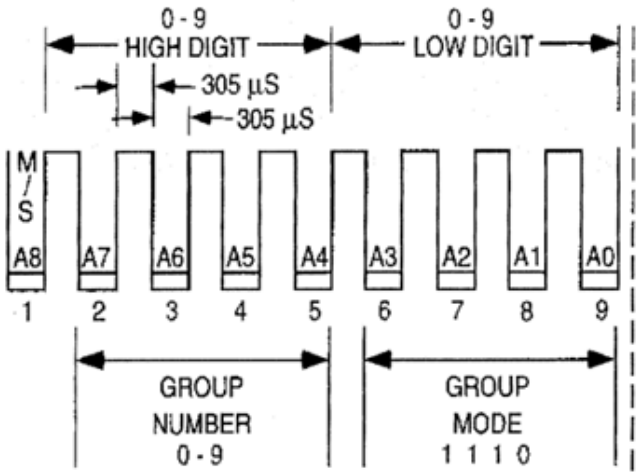

Fig. 8. Representation of the FlashScan common message header, as presented in the FlashScan patent [9].

D/M	HIGH ADDR.			LOW ADDR.			OUTPUT BITS					
A8	A7	A6	A5	A4	A3	A2	A1	A0	B2	B1	B0	P
1	1	0	0	1	1	0	0	1	0	0	0	1

Fig. 9. Example of a single device polling message. This particular example polls control unit 99.

E. SLC Message Structure

The team had previously examined a patent for the FlashScan protocol filed by Honeywell in 1996 [9]. FlashScan is Honeywell's proprietary SLC protocol extension and implementation which provides lower device polling latency, higher polling bandwidth, and is capable of addressing more devices than the standard CLIP protocol. Though the patent details different message types, structures, and purposes, the team was unsure to what degree those semantics had been retained in the nearly 30 years since the patent was filed.

As described in the patent, FlashScan has two types of messages sent by the FACP: single device polls, and device group polls. Devices are grouped based on the combination of their SLC address's tens and hundreds digits; for example, a device with address 115 would belong to group 11. Both message types share a common 9 bit header. The first bit determines whether the message is intended for a controller module, such as a beacon or siren, or a detector module, such as a smoke or heat detector. The next four bits represent the SLC group number. The last four bits represent the ones digit of a single device's address if below 10, or otherwise represents an "escape code" for a device group poll. The structure of the common header can be seen in figure 8.

For single device polling messages, the common header is followed by three bits representing the "command" requested by the FACP, then ended with a single parity bit, which mitigates the potential of message corruption. An example can be seen in figure 9.

D/C	HIGH ADDR.			LOW ADDR.			GROUP MEMBER SELECTION BITS								CONTROL				
A8	A7	A6	A5	A4	A3	A2	A1	A0	G9	G8	G7	G6	G5	G4	G3	G2	G1	G0	B2	B1	B0	P
0	0	0	0	0	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	1	1	0

Fig. 10. Example device polling message with an explicit group member mask. This particular example selects detector units 5 and 8.

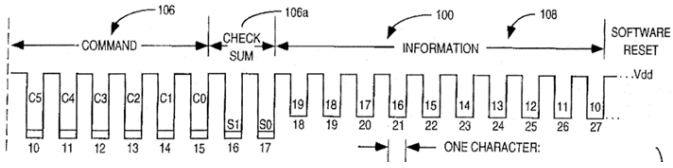

Fig. 11. Example of the body of a device group polling message with no explicit group member mask. This would be preceded by a common header similar to the one shown in figure 8.

The patent presents two different types of device group polls. The first type is exemplified in figure 10. In this case, the common header is followed by 10 bits acting as a mask of which devices within the group are to respond to the message. Similar to the single device poll, this is then followed by three command bits and one parity bit. The patent allows for devices to respond in either digital or analog form after receiving the message. In contrast, the patent then presents the second group polling message structure, as seen in figure 11. This represents the command field using six bits, uses two checksum bits instead of a single parity bit, allows for bidirectional communication during the “information” period after the checksum, and contains no mask of selected devices. Though the example illustration from the patent shows 10 subdivisions of the information interval, it clarifies that devices may be allocated multiple subdivisions, and that the total number need not necessarily be fixed.

Returning to the FSM in the SLC chip’s firmware responsible for processing messages received from the SLC line, it seems that the semantics of the first message bit as a flag for distinguishing between seems to have been preserved. As seen in figure 12, one of the uses of bit 1 is to decide which of two arrays to read from,. These two arrays are referenced elsewhere alongside logging output mentioning ”MOD” and ”DET”, which would be short for ”module” and ”detector”, respectively, with a value of “0” seeming to still represent detectors, while “1” signifies a controller, as in the patent. This

```
    if (BOOL_00003039) {
      if ((&modules[0].type)[DAT_00003042 + DAT_00003042 * 2] != 0) {
        return true;
      }
    }
    else if ((&detectors[0].type)[DAT_00003042 + DAT_00003042 * 2] != 0) {
      bVar4 = true;
    }
  }
  return bVar4;
}
```

Fig. 12. Example of message bit 1, stored in `BOOL_00003039`, used to distinguish access to different arrays

```
if ((&modules[0].type)[n + n * 2] == 0) {
  bVar4 = false;
  send_str_to_USCI_A1(s_MOD:_00016da4);
  send_3_digit_num_to_USCI_A1(n);
  send_str_to_USCI_A1(s_REM_not_found_00016da9);
}
else if ((&detectors[0].type)[n + n * 2] == 0) {
  bVar4 = false;
  send_str_to_USCI_A1(s_DET:_00016dba);
  send_3_digit_num_to_USCI_A1(n);
  send_str_to_USCI_A1(s_REM_not_found_00016dbf);
}
```

Fig. 13. Decompiled output from Ghidra showing correlation of strings ”MOD” and ”DET” with certain arrays

usage is shown in figure 13. In fact, the entire common header structure has been implemented as described in the patent, with the four bit high address signifying device group, and the low address maintaining its role as differentiating single device from device group polls, as well as providing the ones place of the SLC address for a single device poll.

The single device polling structure has also seemingly been kept with no major modification. Though the patent is vague on the exact mechanism for devices to return information to the control panel after the polling message, the FSM contains five states after the parity bit which do not read from P1.0, the SLC input pin. This, coupled with the gateway’s use of a comparator to receive SLC signals, seems to suggest that the actual implementation of FlashScan only allows for digital encoding of information, dropping support for analog signals.

In contrast, the gateway seems to only support device group polls without a member mask, though the structure, as in figure 10, seems to again have been retained. The information section comprises 10 states, with each being dedicated to one member of the device group. Interestingly, the gateway seems to permute the order of the members’ reply slots. Instead of a standard 0-9 ordering, member 9 gets the first slot, followed by 4, 3, 2, 6, 1, 9, 7, 5, and finally 0. The team does not currently have a hypothesis as to why this would be.

As the gateway is intended to integrate into an existing wired fire protection system, it will check the device(s) addressed in a polling message by indexing into various arrays stored in the SLC chip’s memory before responding. This allows the gateway to only respond on behalf of the wireless devices that are otherwise unable to communicate with the FACP. However, it also indicates that, contrary to previous assumptions, the gateway does not synchronously relay messages between the wired SLC loop and devices in the wireless mesh network. Instead, it responds on behalf of wireless devices using information stored in its memory, which is presumably updated by the RF processor using the USCI channel between the two, which was previously identified by the team [5].

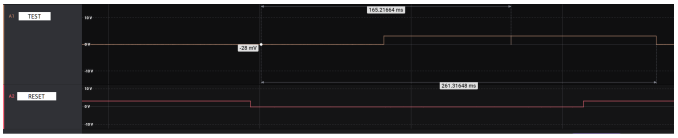

Fig. 14. Saleae Logic Analyzer capture of TEST and RST pins of MSP-FET during invocation of BSL.

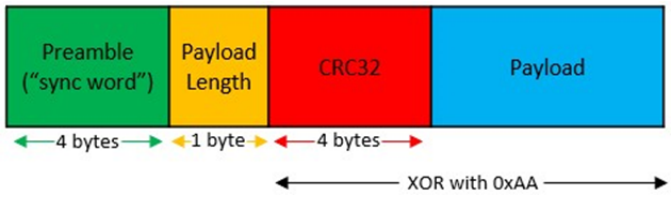

Fig. 15. Layout of bytes for W-USB Over-the-Air protocol

IV. BSL Scripter

A. BSL Invocation Sequence

Though the team was unable to interface with the RF chip through BSL due to unknown errors, further analysis was conducted to ensure proper function of the BSL setup. [5] The first contact that the program makes with the chip is an invocation sequence whose existence was known to the team but not explored. It takes place on the TEST and RST pins between the MSP Flash Emulation Tool (MSP-FET) and the MSP430 microcontroller. [10] In order to activate BSL on the MSP430, the TEST signal must have two rising edges and be held high while the RST signal transitions from low to high. Once the TEST pin is pulled down after this sequence, the program counter of the chip jumps to the BSL flash memory where instructions for execution lie. Based on readings from the respective pins during BSL Scripter startup in 14, the invocation sequence is working correctly, meaning the errors for BSL originate from another source.

V. RF Backdoor Attack

A. Background

One possible avenue of attack is a backdoor into the gateway. This would allow access to the gateway while circumventing authentication methods, and, depending on the implementation, it could give the attacker a direct method through which to manipulate the fire system. The team began work on a backdoor attack against the RF chip to demonstrate control over the system by executing an arbitrary function triggered by a message to the gateway. Since previous research has investigated the RF chip's workings, the backdoor could take advantage of this knowledge in its implementation. For example, the team's planned approach utilizes the firmware upload script to add custom firmware containing the backdoor to the RF chip [4]. Additionally, since the gateway frequently communicates with the W-USB, analysis of the USB's wireless protocol in 15 provides a foundation to understand the gateway message handling to be exploited in the backdoor [3].

B. RF Message Reception

The firmware of the RF chip must be configured to recognize a message that triggers the backdoor activation. As a result, the team investigated the RF chip's message reception mechanisms. The chip is connected to an SX1231 RF transceiver via an SPI channel linked to the RF chip's USCI B0 port. The channel enables the RF chip to manipulate the transceiver's registers, which include a byte-wide buffer, flags for the buffer state, and sync word configuration [11]. The register is addressed using bits zero through six while bit 7 acts as a flag for a write command. If bit 7 is zero, a read of the register indicated by the rest of the byte is immediately transmitted to the gateway. If bit 7 is one, the next byte sent by the gateway will be stored in the indicated register. For the backdoor, an understanding of RF message reception would allow the team to edit the range of messages that the gateway can parse, making the references to the USCI B0 and the SPI connection a point of interest in firmware analysis.

C. Firmware Analysis

The team reverse-engineered the logic in the RF firmware that relates to message reception, but message parsing remains undiscovered. The chip uses helper functions `read_from_transceiver_at_reg` and `Writes_data_out_SPIB0_to_RF_Chip_at_Reg` to interact with the transceiver registers. An important register for reception is 0x28, which holds flags indicating the status of the payload buffer. Specifically, bit 6 is set when the buffer is not empty, and bit 2 is set when the whole payload of a message is ready to be read [11]. The RF chip frequently reads from register 0x28 and checks bits to decide when it is receiving a message from a device or the W-USB.

Through recognition of these patterns, the team has determined the partial flow of receiving messages from the transceiver and storing them in memory. First, `do_stuff_and_wait_payload_ready` loops while checking if a payload is ready, and once a payload can be read, it calls `put_fifo_at_461d_and_verify_crc` to store and verify the payload of the message. Stepping into the function, the team found that a helper function stores bytes read from the transceiver buffer at 0x461d in memory with the length of the message stored at 0x471c. The function moves on to verification of the message. It extracts the CRC value in figure 15 and proceeds to calculate the expected CRC of the payload. The SX1231 transceiver performs data whitening by XORing data with 0xAA, so an XOR is performed to undo the whitening before calculating the expected CRC [11]. If CRC values match and the payload is uncorrupted, `put_fifo_at_461d_and_verify_crc` returns a 0, otherwise a non-zero value whose significance has not been determined.

The original function `do_stuff_and_wait_payload_ready` checks the return value after CRC verification, breaking from the payload-waiting loop if it is valid. If not, the function restarts the transceiver to improve its received signal strength indicator (RSSI). Despite understanding the storage at 0x461d

and verification of a payload, the team has not determined how the gateway parses the payload for message type and commands.

VI. Conclusion

A. Next Steps

Continuing the team's efforts in creating an RF backdoor attack is an important goal for the future, and it will involve further analysis of the RF firmware. With payload storage and verification explored, the team will need to deduce what is done with the data after the aforementioned steps. Although efforts to reverse engineer them have not been successful, functions that perform payload parsing are likely tightly coupled with functions that have already been analyzed. Given the team is able to fully reverse-engineer the parsing logic, it will be possible to edit the firmware in preparation for the backdoor, and the W-USB or a software-defined radio will be able to trigger it.

A natural extension of the team's analysis of the SLC message structure is to reverse engineer the semantics of the command and response fields. The team's attempts to analyze the functions in the SLC firmware responsible for interpreting those fields were largely unsuccessful, due to the frequent references to device information stored in the SLC chip's memory. The team is currently almost entirely unaware of the layout and information contained within those sections of memory, and so began turning its attention to the RF firmware, from which those locations are set. The team had previously decoded the wireless over-the-air protocol between wireless devices and the gateway; however, the message format between the two gateway processors is entirely different. Therefore, the next step towards fully decoding the wired SLC protocol would be to examine the RF firmware to determine the interprocessor message structure, then identify what information from those messages are stored and where, then correlate that knowledge with how the command bits of an SLC poll are interpreted, as well as how the response is constructed.

A goal that the team will begin to prioritize is spoofing a fire alarm on the wired SLC network. One of the approaches the team will be taking to accomplish this goal is to create a surrogate device that can send spoofed SLC messages. The surrogate device would most likely be made using a Raspberry Pi and the components that will connect the device to the wired network, like the materials depicted in the figure 16. Currently, the team has decided on using two 5V relay boards to control the voltage supply. Also, the team decided on a Raspberry Pi, rather than an Arduino, so that the breadboard that is connected to the Pi will allow for more components to be used in the surrogate device, and so components can be removed and replaced easily. In order to achieve this goal, the team concluded that a device needs to be taken out of the SLC line, which is daisy chained through all of the devices, and the surrogate device needs to be inserted in its place and do what the replaced device did so that the SLC line does not realize a device is missing. Along with replicating the replaced device's functionality, the surrogate device will also need to insert the spoofed message the team makes in order to successfully spoof a fire alarm. Moving forward, the team will be working on assembling the various components of the surrogate device, making the spoofed message, and conducting a logic capture of the wired SLC network in order to determine which device the surrogate device will replace and when said device is polled by the FACP.

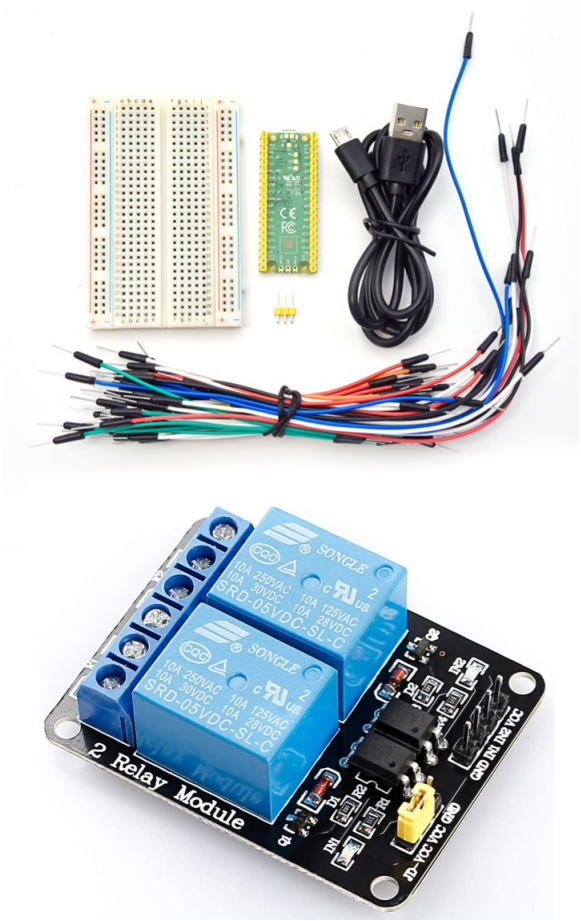

Fig. 16. This image depicts some potential components of the surrogate device: a breadboard (top left, beige), a Raspberry Pi (top middle, green), wires (middle, multi-colored), and a 5V relay board (bottom, blue and black).

References

[1] Detectors. [Online]. Available: https://www.securityandfire.honeywell.com/notifier/en-us/browseallcategories/wireless/swift/detectors

[2] Texas Instruments. SWIFT™ Smart Wireless Integrated Fire Technology Manual. [Online]. Available: https://prod-edam.honeywell.com/content/dam/honeywell-edam/hbt/en-us/documents/manuals-and-guides/user-manuals/LS10036-000FL.pdf?download=false

[3] D. Lawrence, G. Kokinda, G. B. A. Lukman, Y. Kim, J. Smalligan, and C. Roberts, "Swift wireless fire alarm pull station analysis," Nov. 2021.

[4] D. Lawrence, G. Kokinda, G. Brown, D. Chou, Y. Kim, S. Wright, and C. Roberts, "Swift wireless fire alarm system analysis," May 2022.

[5] A. Bussey, D. Chou, J. Y. Kim, S. Suman, S. Wright, and D. Keskin, "Swift wireless fire alarm system analysis," Nov. 2022.

[6] D. Krantz, *Make It Work - Addressable Signaling Line Circuits*. Douglas Krantz, 2021.

[7] Texas Instruments. Low-power quad differential comparators. [Online]. Available: https://www.ti.com/lit/ds/symlink/lp339.pdf

[8] ——. MSP430F543xA, MSP430F541xA Mixed-Signal Microcontrollers. [Online]. Available: https://www.ti.com/lit/ds/symlink/msp430f5419a.pdf?ts=1645687841030
[9] E. Bystrak and A. Berezowski, “Enhanced group addressing system,” U.S. Patent 5 539 389, Jul. 23, 1996.
[10] Texas Instruments. MSP430™ Flash Devices Bootloader (BSL). [Online]. Available: https://www.ti.com/lit/ug/slau319af/slau319af.pdf
[11] Semtech. Sx1231 datasheet. [Online]. Available: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/44000000MDkO/lWPNMeJClEs8Zvyu7AlDlKSyZqhYdVpQzFLVfUp.EXs