CSAW Embedded Security Challenge Fall 2023 Final Paper

Levi Doyle
Georgia Tech
Atlanta, U.S.
ldoyle9@gatech.edu

Adith Devakonda
Georgia Tech
Atlanta, U.S.
adevakonda3@gatech.edu

Madelyn Novelli
Georgia Tech
Atlanta, U.S.
mnovelli3@gatech.edu

John Zhang
Georgia Tech
Atlanta, U.S.
jzhang3213@gatech.edu

***Abstract*—The 2023 CSAW Embedded Security Challenge focuses on side channel attacks (SCAs) on cyber-physical systems (CPS). The competition phase put side channel knowledge into practice as teams attempted to retrieve flags from an Arduino based system running various programs.**

I. Introduction

The competition phase of the 2023 CSAW Embedded Security Challenge spanned three weeks, during which two challenges were released each week. The challenges were to be run on an Arduino Uno R3 board with provided peripherals, including a relay, keypad, speaker, microphone, vibration motor, and seven-segment display. During the three weeks, the team were only able to receive the flag for the Vender Bender challenge. Guides to the challenge solutions were released by CSAW after the competition period, which the team used to successfully receive the flags for all of the challenges except All White Party.

II. Setup

The team primarily used the Arduino IDE and its built-in serial monitor to analyze and complete the given challenges. A phone is used to record the Arduino for the Bluebox, SPItFire, and czNxdTNuYzM challenges. For the Bluebox challenge, the team also utilizes a guitar tuning app to record the frequencies of sounds output by the Arduino.

III. All White Party

A. Overview

This challenge simulates a username and password interface. The user is prompted for a username over serial port. If the username is incorrect, the Arduino vibrates and then sends an 'Incorrect Username' message. If the username is correct, the user is then prompted for a 10 digit 2FA pin on the keypad upon receiving the correct username. Upon entering the correct pin, the user is granted access to the system. An incorrect PIN causes to Arduino to quickly vibrate and the login process is restarted.

B. Challenge Period Attempt

The challenge description gave the hint of "time" for this challenge. This led the team to search for a solution using a timing attack, which involves analyzing the running time of a system. [4] After experimenting with continuously inputting usernames in the system at a rapid pace, we found that while the Arduino is vibrating, it will not process the next username input until it finishes its vibration and sends its 'Incorrect Username' message. This allowed the team to have consistent timing with our inputs, and using the timestamps provided by the serial monitor in the Arduino IDE allowed us to measure inputs. The team started with single character inputs with every possible input, and saw a consistent delay of around 1.7-2.2 seconds before the system responds. The only significant outlier was 'B', which caused a 2.5 second delay. Seeing that the username was likely a normal word, the team then tested with variations of 'B' followed by a vowel, and found that 'Ba' was also a significant outlier at 2.7 seconds from the 2.4-2.5 second range of the other variations. The search for the next character found early that 'Bar' created a 3 second delay while other inputs averaged 2.79-2.82 seconds. Enough information was gathered to correctly guess that the username was 'Barry' and move on to the password portion of the challenge. However, the passcode was not able to be determined with the same method. All 10 digits of the passcode needed to be inputted before the system

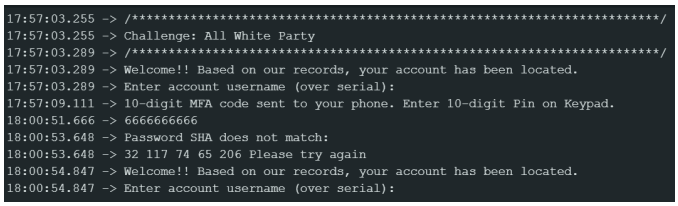

Fig. 1. All White Party running in the Arduino IDE serial monitor

Start Time:	97.679	
Z	99.879	2.2
X	102.068	2.189
C	104.272	2.204
V	106.482	2.21
B	108.996	2.514
N	111.192	2.196
M	113.397	2.205
<	115.575	2.178
>	117.796	2.221
?	119.99	2.194
SPACE	122.177	2.187

Fig. 2. Input Delay Differences

would process it, and the Arduino IDE timestamp only counted when the first digit of the passcode was entered. The Arduino also did not follow the same vibration and delay process as the username, so we were unsure how much our previous method would apply. The 'Incorrect Password' message also included a set of numbers, which the team found would change upon inputting 'Barry' again into the serial monitor before fully entering the passcode, and then making further inputs. However, progress on the challenge halted after this point. To mitigate the possibility of a timing attack on the username entry, an artificial delay can be added that would increase the amount of characters needed to have a measurable time difference and therefore greatly reduce the practicality of a timing attack. Finding a way for the Arduino's vibration to be asynchronous from processing an input could also present another possible solution.

C. Post-Challenge

Due to time constraints, the team was unable to obtain the flag for All White Party. However, the challenge solutions show a method that can be used to obtain the flag. While 'Barry' was a valid username, the system only checks the first five bytes of input, so the system would accept any username as valid if it started with 'Barry', such as 'Barryyyyy', 'BarryWorlow', or 'Barry3d4j37l36'. Teams in the challenge were intended to use this information to find a combination of a username beginning with 'Barry' and a 10-digit password that created a SHA1 hash collision, meaning they share the same hash. The matching password would be the correct password to enter the system. According to the solution description, a program comparing two sorted dictionaries of eligible usernames and passwords would be able to quickly find a hash collision. [1]

IV. Bluebox

A. Overview

The premise of this challenge is to recreate different combinations of audio tones using keypad inputs. Each key on the numpad correlated with a specific frequency, likely mimicking a dual-tone multi-frequency (DTMF) telephone pad. Once the first four frequencies are recreated, an eight-note tone is played and must also recreated in order to successfully complete the challenge.

B. Challenge Period Attempt

The nature of the challenge seemed to indicate that acoustic cryptanalysis, an emissions-based side-channel attack in which audio data is gathered with the intention of gaining critical information about the device, was the correct solution. [2]

C. Post-Challenge

A frequency analyzer app was used to record the tones played by each key. However, there were inconsistencies in the app, so a instrument tuning app was used instead to record the frequency of each key. Then the tuning app's results were recorded when a new sequence was played so the tones could

1	A(5)# 953Hz
2	B(5) 1011Hz
3	C(6) 1068Hz
4	D(6) 1190Hz
5	D(6)# 1249Hz
6	E(6) 1308Hz
7	F(6) 1428Hz
8	F(6)# 1487Hz
9	G(6) 1542Hz
0	A(6) 1723Hz
A	C#(6) 1130Hz
B	F(6) 1365Hz
C	G(6) 1602Hz
D	A#(6) 1837Hz
#	G#(6)1666Hz

Fig. 3. Frequency Table

```
18:10:13.185 -> /******************  YOU BEAT THE CHALLENGE!!!   ******************/
18:10:13.185 -> Congrats! Please add the flag to the report...
18:10:13.185 -> Flag: B339B009
18:10:13.185 -> /******************         Congrats!!!        ******************/
18:10:13.185 ->
18:10:13.185 ->
```

Fig. 4. Success Message for Bluebox

be replayed and identified. Once the correct 4 tone sequence was entered, a new sequence was played with 5 distinct changes in tone, but with 2 tones held. Incorrectly entering it would return to the previous stage, but correctly entering it would yield success chimes and a message on serial stating that the flag was the key sequence used to replay the tone: "B339B009".

V. Operation SPItFire

A. Overview

This challenge's premise is accessing a security camera. The serial monitor states that the Arduino is receiving the message 'HELLO' while the relay on the Arduino board clicks repeatedly in a pattern, turning on and off a light attached to the relay. The serial monitor then prompts the user to enter the message 'FLAG' in hex using the correct formatting in order to establish communication with the camera.

B. Challenge Period Attempt

At first, we believed that the relay's clicks resembled Morse Code. However to translate it as Morse code yielded inconsistent results proved difficult and we would end up with messages such as "IISEESTETEM". The team was unable to make much more progress past this point.

C. Post-Challenge

The solutions to the challenges revealed that the clicking of the Arduino during the transmission of the "HELLO" message was binary, with the relay light being on representing 1, and the light being off representing 0. [1] Recording the message with a phone and analysing the outputs allowed us to translate

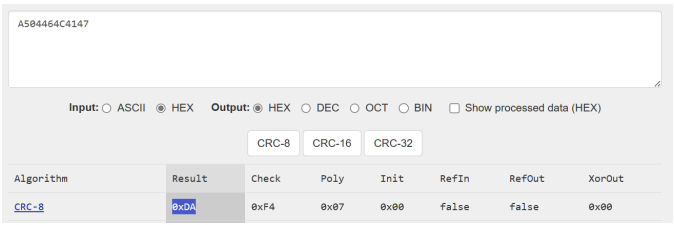

Fig. 5. CRC found at https://crccalc.com/

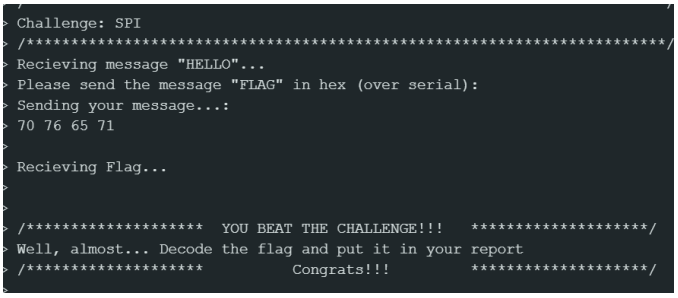

Fig. 6. Success message for SPItFire displayed as the Arduino plays the flag relay

the message into hexadecimal (A50548454C4C4F39), which helped the team to find the correct format to send ”FLAG” in. Sending ”FLAG” in hex (464C4147) produced an ”Incorrect Header” message. Sending the same message the first byte of the header message of the ”HELLO” message (A5464C4147) passed the header check and gave us an ”Incorrect Length” error. Analyzing the ”HELLO” message once more shows that after the header, the next byte is 05 followed by the bytes for ”HELLO” in hex. 5 is the length of ”HELLO”, so the team added the length of ”FLAG” to our message (A504464C4147), but still received an ”Incorrect Length” message. After noticing that the ”HELLO” message had an extra byte after the hexadecimal for ”HELLO”, the team added an arbitrary extra byte to the end our message. This passed the length check and gave a new ”Bad CRC” message. CRC, which stands for Cyclic Redundancy Check, appears to serve as an extra byte on the data that allows for checking for corruption, meaning ’A504464C4147’ has a specific CRC byte that goes with it. This was easily found with an online tool (https://crccalc.com/). The completed ”FLAG” message (A504464C4147DA) is accepted by the system and causes the Arduino the transmit a new message through the relay, and prompts the user to translate the message and enter it into our paper as the flag. The message was much longer than the hello message, making translation more difficult. After analysis of a video of the relay sequence, the flag seems to be ”Spy Burned” if not slightly altered.

VI. czNxdTNuYzM

A. Overview

This challenge sends its own name over serial port followed by the string:

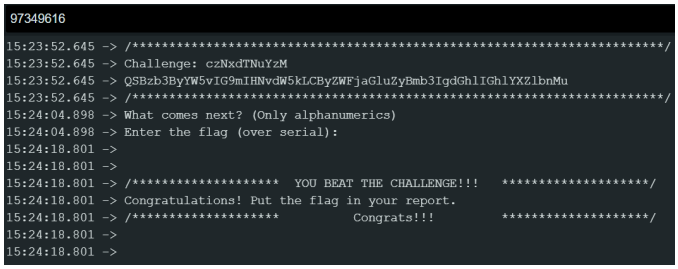

Fig. 7. Success message for czNxdTNuYzM

”QSBzb3ByYW5vIG9mIHNvdW5kLCByZWFjaGluZyBmb3IgdGhlIGhlYXZlbnMu”

It asks for the next in the sequence as alphanumeric characters after the board finishes presenting a sequence of numbers and underscores on the seven segment display. Observing a recording of the seven-segment display on the Arduino board yields the sequence: ”1.2.6.2_4.12_0.2_0.1_4_0.1_1_2_0.1_0_0_8_0.1_0_0_8.1_8.9_2_4.1_20_2.85_8.1_2_8_7_0.2_0_5_9_2_0.3_5_0_0_6_4_0.1_94_8.3_6_9_5_1_2_0.1_8_4_7_5_6.3_8_7_9_8_7_6.2”

B. Challenge Period Attempt

The team found that no features of the numerical sequence appear to align with the string. The sequence features only four numbers with 2 digits and has varied lengths for sequences of underscores. The team was unable to come up with a potential solution for this challenge in time.

C. Post-Challenge

Rewriting the sequence as numbers shows a possible pattern since many of the first few inputs are multiples of each other if the underscores are to be understood as connecting the digits of numbers. The length of time periods were held for implies they likely separate the numbers. Rewriting the numbers gives: 1, 2, 6, 24, 120, 20, 140, 1120, 10080, 1008, 18, 924, 1202, 858, 12870, 205920, 3500640, 1948, 3695120, 184756, 3879876, 2 There appears to be ascending sequences but they are followed by varying amounts of decreasing numbers or smaller numbers. Taking the ratio between numbers gives: 2, 3, 4, 5, 1/6, 7, 8, 9, 1/10, 1/56, 154/3, 601/462, 429/601, 15, 16, 17, 487/875160, 923780/487, 1/20, 21, 1/1939938. A hint demonstrates that the sequence increases by the increasing ratio unless the number is divisible by the number and that some of the numbers were observed incorrectly. By calculating the latter half of the sequence and several additional numbers in the sequence, it can be checked against the sequence shown on the board to identify the next number. By observing the display, the final number ends with ”234”. Comparing it with numbers in the sequence, it aligns with the 23rd number in the sequence. The 24th number is ”97,349,616” and entering it yields that it is the flag for the challenge.

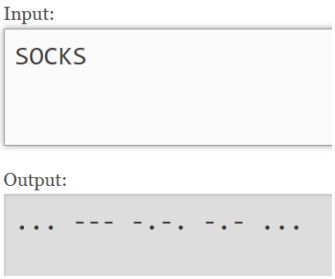

Fig. 8. ”SOCKS” translated into Morse Code. The Arduino beeping correlates with the dots. Website used: https://morsecode.world/international/translator.html

VII. SockAndRoll

A. Overview

This challenge presents the premise of escaping a locked room in a sock factory using a communication device in the room. Over serial port, the device sends a message in French that translates to ”It’s out. Help us. Help us.” The speaker on the Arduino board outputs a beep several seconds long, then shortly beeps three times in succession each, followed by another long beep before pausing then repeating, while the speaker plays background whirring noises. The microphone on the Arduino records the beeps, and the program interprets them. Between repetitions, the serial port sends the message ”Glad to know things are okay. Enjoy your time at the factory!”

B. Challenge Period Attempt

The team found that the beeping does not align with the convention for an S.O.S., and was not able to make any further progress in time.

C. Post-Challenge

The challenge solutions revealed that the beeping and whirring output by the Arduino Speaker is a Morse Code transmission of ”SOCKS.” Teams were intended to disrupt transmission during ”CK” in order for the message to read ”SOS.” [1] Upon analysis of the beeping message with Morse Code in mind, the team discovered that the long beeps corresponded with ”S” that appears at the beginning and end of ”SOCKS” and ”SOS.” The three shorter beeps in between are used for the transmission of ”CK.” The peripherals on the Arduino are easily detachable, so the microphone is easily able to be detached and re-attached to manipulate the message that is recorded by the Arduino. By letting the microphone read the long beep to transmit ”S,” waiting a short period for ”O” to be transmitted, then detaching the microphone before the three short beeps and re-attaching it when the second long beep plays to transmit the second ”S,” the message ”SOS” is received by the microphone and the challenge is completed. The program gives the flag ’f0otNOt3.’

Fig. 9. Success message for SockAndRoll

Fig. 10. Success message for Vender Bender

VIII. Vender Bender

A. Overview

This challenge prompts over serial port to send the string ”ERR” over serial when a token is received to attempt to jam a motor. The relay module clicks back in forth at regular intervals, and a successful motor movement message is sent over serial port followed by a repeat of the other message.

B. Challenge Period Attempt

By observing the timing of the messages and relay, we found that the second click following the instruction to send ”ERR” is timed closely with the successful motor movement message. Sending the message at the same time as the second click causes a new message to show up on serial port and a new set of two clicks of the relay at a different timing. The new message states that there was slight motor movement and gives a count that it is one of five. By continuing to send ”ERR” during every second click, successive sets of relay clicks and slight motor movement messages occur. The 5th consecutive motor movement message becomes a success, which grants the success message over serial port with sounds of fanfare on the board, revealing the flag ’mMmCaNdY’.

Treating the situation as if we are actually jamming the motors of a vending machine, we are performing a fault injection attack. We introduce ’faults’ (jam the motors) and cause sensitive data (food in the vending machine) to become visible when it should not be (the food was not paid for), which fits the definition of a fault injection attack. [3] Possible mitigations for this side channel attack include preventing the vending machine’s motors to be locked on command when in a vulnerable state, or simply removing the ability to manually lock the motors.

References

[1] Folkerts, Lars. (2023, November 15) *Solutions*. GitHub. https://github.com/TrustworthyComputing/csaw$_esc$2023/$blob/main/Solutions.md$

[2] Genkin, D., Shamir, A. and Tromer, E. Acoustic Cryptanalysis. J Cryptol 30, 392–443 (2017). https://doi.org/10.1007/s00145-015-9224-2
[3] Giraud, C. (2005). DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds) Advanced Encryption Standard – AES. AES 2004. Lecture Notes in Computer Science, vol 3373. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11506447_4
[4] Karthik, S. (2020, December 23). *Introduction to Timing Attacks!*. Medium. https://medium.com/spidernitt/introduction-to-timing-attacks-4e1e8c84b32b