CSAW ESC Final – Report

Hackcess, IUT ROANNE, France

Mathieu DUMAS
Hypno
Mathieu.bovy@hackcess.org

Gabin LEVEQUE
Alvi
gabin.leveque@orange.fr

Guy MBENGUE
Gmx.0x1
Mbengue.guy@hackcess.org

Elisa MILLOT
Tensheli
Elisa.millot@hackcess.org

I. INTRODUCTION

A. Context

This report is about our team, Hackcess, qualifying for the CSAW ECS 2024 finals. We were excited to be named finalists, and we received the Arduino board and other components, officially starting the competition.

B. Objectives

In this report, we will share our achievements in the various challenges we faced over four (4) weeks, which were very various.

C. Hardware and tools

Here is the list of equipment we had:

- Arduino Uno R3 board
- AC/DC Power supply
- Motor shield L293D 4
- Fan motor
- Servo motor
- Stepper motor

The Arduino Uno R3 board was faulty, so we had to replace it with another one, which caused a slight delay.

II. CHALLENGES WEEK 1

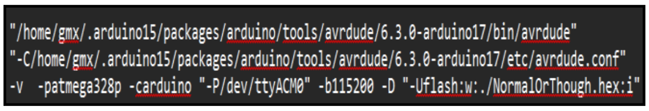
A. Normal or Though

The first challenge was called "Normal or Though" and its description was:

"You decided to go to a pub in New York named "Normal Or Though" in the upper west side, a quality restaurant. You hope to have great food based on a quirky recommendation. You were told to order a peculiar food that rarely anyone orders, but the waiter brings you the food after only one minute, and you are surprised by the quick response. As soon as you finish your food, the waiter gives you an arduino, for you to pay and give a quality review. The waiter leaves and you notice that you are the only one left in a quiet room. Then you have a realization... "

After completing the "hardware test," we received the command line to upload a ".hex" (fig1) file to the board. When we uploaded the hex files of the challenge, we noticed that only the fan motor was functioning.

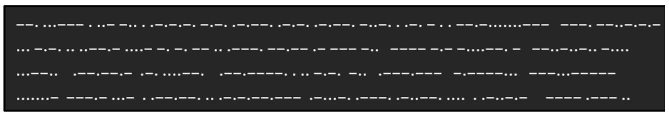
We also noticed that the electric pulsations of the fan motor sounded like Morse code.



```
"/home/gmx/.arduino15/packages/arduino/tools/avrdude/6.3.0-arduino17/bin/avrdude"
"-C/home/gmx/.arduino15/packages/arduino/tools/avrdude/6.3.0-arduino17/etc/avrdude.conf"
-v  -patmega328p -carduino "-P/dev/ttyACM0" -b115200 -D "-Uflash:w:./NormalOrThough.hex:i"
```

fig 1 : command to upload hex code.

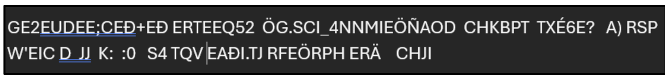
First, we transcribed the pulsations of the fan motor into Morse code (fig2)



```
--. ..---- . ..- -.. . .-.-.-. -.-. .-.--. .-.-. .-.--. -..-. . .-. - . . --.-......---  ---. --..-.-.-
... -.-. .. ..--.- ....- -. -. -- .. .---. --.-- .- --- -..  ---- -.- -....--. -  --.-..-.. -....
...--..   .--.--.- .-. ....--.   .--.----. . .. -.-. -..  .---.---  -.----...  ---..-----
.......- ---.- ...- . .--.--. .. .-.-.--.--- .-...-. .---. .-..--. .... . .-..-.-   ---- .--- ..
```

fig 2 : Binary serial

then, we transcribed morse code into alphabet, it seems this message. (fig3)



```
GE2EUDEE;CEĐ+EĐ ERTEEQ52 ÖG.SCI_4NNMIEÖÑAOD CHKBPT TXÉ6E? A) RSP
W'EIC D JJ K: :0 S4 TQV EAĐI.TJ RFEÖRPH ERÄ CHJI
```

fig 3 : transcription

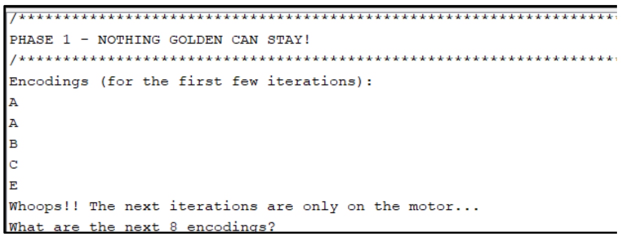
We were stuck here for the first challenge because we did not find any other clues.

B. Friendly Disposition

The second challenge of the first week was called: "Friendly Disposition" and its description was:

"In the heart of a bustling metropolis, a renowned cybersecurity firm, The Sequence Syndicate, has been tasked with securing the city's critical infrastructure against a series of sophisticated cyber-attacks. The attackers, known only as Malicious Disposition, have left behind a trail of encrypted messages that seem to follow mysterious patterns. Decode the sequences and figure out what message the attackers are trying to communicate. "

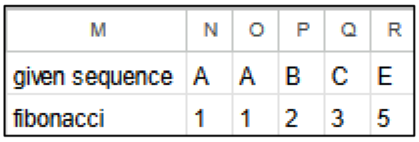
1. First phase: "Nothing golden can stay"



```
/*************************************************************************
PHASE 1 - NOTHING GOLDEN CAN STAY!
/*************************************************************************
Encodings (for the first few iterations):
A
A
B
C
E
Whoops!! The next iterations are only on the motor...
What are the next 8 encodings?
```

fig 4 : phase 1

After looking for the eight encodings with angles, we noticed that the first five letters of the given sequence resembled a Fibonacci sequence.

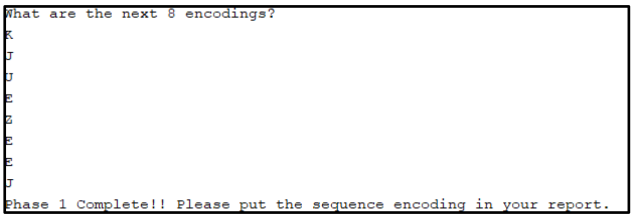


M	N	O	P	Q	R
given sequence	A	A	B	C	E
fibonacci	1	1	2	3	5

Fig 5 : letters and fibonnacci

The value of the Fibonacci sequence corresponds to a letter of the alphabet referring by its place. When we exceed 26, we subtract 26 from that value as many times as necessary to get back under 26.

The servo motor completed 16 iterations, and we used our modified Fibonacci algorithm to determine the height next iterations.

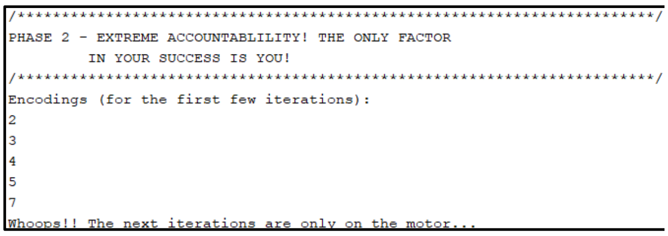


```
What are the next 8 encodings?
K
J
U
E
Z
E
E
J
Phase 1 Complete!! Please put the sequence encoding in your report.
```

Fig 6: Phase 1 completed

sequence : KJUEZEEJ

2. Second phase : "EXTREME ACCOUNTABLILITY"



```
/************************************************************************/
PHASE 2 - EXTREME ACCOUNTABLILITY! THE ONLY FACTOR
          IN YOUR SUCCESS IS YOU!
/************************************************************************/
Encodings (for the first few iterations):
2
3
4
5
7
Whoops!! The next iterations are only on the motor...
```

fig 7: phase 2

in the next phase, another sequence was given. This sequence was composed with 5 digits.

We tried the Fibonacci algorithm in this second phase, but it did not match the "logic" of the sequence. Then, we attempted other known sequences such as squares, prime numbers, arithmetic, and perfect numbers. Unfortunately, we were unable to match or understand the second phase. We are only certain that the 16 encodings consist of 0 to 9 digits.

III. CHALLENGES WEEK 2

A. KeyRing 1

In this challenge, we had to classify samples from a microphone and a vibration sensor from a key manufacturing site. We also had a bunch of "labeled" files which were already classified to help us. Each key type is labeled A,B,C or D, our first task was to discover the key type of every unclassified samples by using the mp3 and CSV files produced by the sensors.

This could have been easily done using a program allowing the files to be sorted by comparing them to those already classified. Which we did. However an error prevented the program from working on our computers, we did not get any results.

B. KeyRing 2

For the Keyring2, we had to find a key that does not match any of the samples we had in the previous challenge. By analyzing the mp3, csv and stl files, we could have found matches between vibrations, sounds and the shape of the key given in the stl file. A program would have been able to recreate a 3D model approaching what could have been produced at this moment. But we only have this theory.

IV. CHALLENGES WEEK 3

A. Fast&Max

the resumed statement of this challenge:

In a bustling downtown, a bank conceals a treasure in a secret chamber protected by two sophisticated safes. A group of 17 thieves, Fast&Max, aims to break in. To unlock the first safe, they need a 16-digit employee ID encrypted with RSA-like encryption. The second safe requires a cleverly crafted alphabetic PIN to access the hidden treasures.

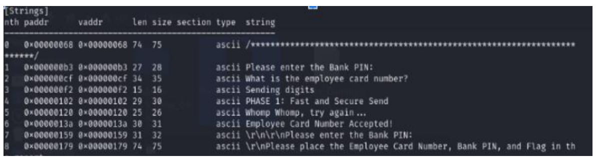
We have noticed that reverse engineering was allowed for this challenge.

We looked at what the servo motor and the serial monitor were doing. We saw that it sent the ID numbers of an employee that were encrypted with an RSA-like key, and that the serial monitor sent 4 digits at a time that we needed to retrieve.

we disassembled the code using "Radare2,

```
srec_cat FastMax_Dummy.hex -Intel -o FastMax_Dummy.bin -Binary
```

then we opened the FastMax_Dummy.bin file in Radare2



```
[Strings]
nth paddr      vaddr      len size section type  string
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
0   0x00000068 0x00000068 74  75           ascii /******************************************************************
******/
1   0x000000b3 0x000000b3 27  28           ascii Please enter the Bank PIN:
2   0x000000cf 0x000000cf 34  35           ascii What is the employee card number?
3   0x000000f2 0x000000f2 15  16           ascii Sending digits
4   0x00000102 0x00000102 29  30           ascii PHASE 1: Fast and Secure Send
5   0x00000120 0x00000120 25  26           ascii Whomp Whomp, try again...
6   0x0000013a 0x0000013a 30  31           ascii Employee Card Number Accepted!
7   0x00000159 0x00000159 31  32           ascii \r\n\r\nPlease enter the Bank PIN:
8   0x00000179 0x00000179 74  75           ascii \r\nPlease place the Employee Card Number, Bank PIN, and Flag in th
```

fig 8 : Radare2 interface

```
r2 -A FastMax_Dummy.bin
```

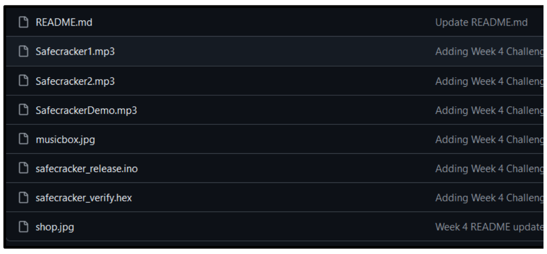
Once in Radare2, this command lists all functions found in the binary file, allowing us to see entry points and potential code sections for analysis.

We observed the servo motor's angles while it displayed ID numbers: 1-4 between 75° and 150°, 5-8 between 0° and 75°, 9-12 again between 75° and 150, and 13-16 between 0° and 75°. The angles varied significantly in each range. Despite attempts to identify repeating sequences or transformations, we found no conclusive results. We hypothesize that the servo motor may communicate in Morse code.

V. CHALLENGES WEEK 4

A. Safe cracker 1

For the safe cracker, I had an ino file allowing me to understand how the challenge worked. I understood thanks to it that the first code would necessarily use the single mode. Then, I looked at the demo audio file. I understood that it reproduced a sequence of a value of 100 2 times in each of the modes following the following model: single, single, double, double, interleaved, interleaved, microstep, microstep.



README.md	Update README.md
Safecracker1.mp3	Adding Week 4 Challeng
Safecracker2.mp3	Adding Week 4 Challeng
SafecrackerDemo.mp3	Adding Week 4 Challeng
musicbox.jpg	Adding Week 4 Challeng
safecracker_release.ino	Adding Week 4 Challeng
safecracker_verify.hex	Adding Week 4 Challeng
shop.jpg	Week 4 README update

fig 9 : tools and files for safe cracker

Thanks to this file, I was therefore interested in the value of 100 in single mode and I discovered that it lasted about 3.15s.

Then, I was able to look at the safecracker audio file in order to carry out the analysis of the audio file and to make the cross product so that I could know what sequence I had to put in the serial monitor. screen result python program, python program also and serial monitor result.

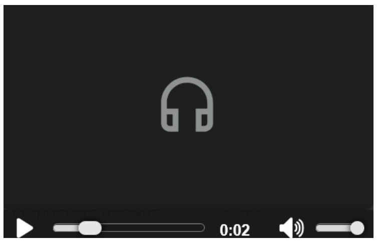
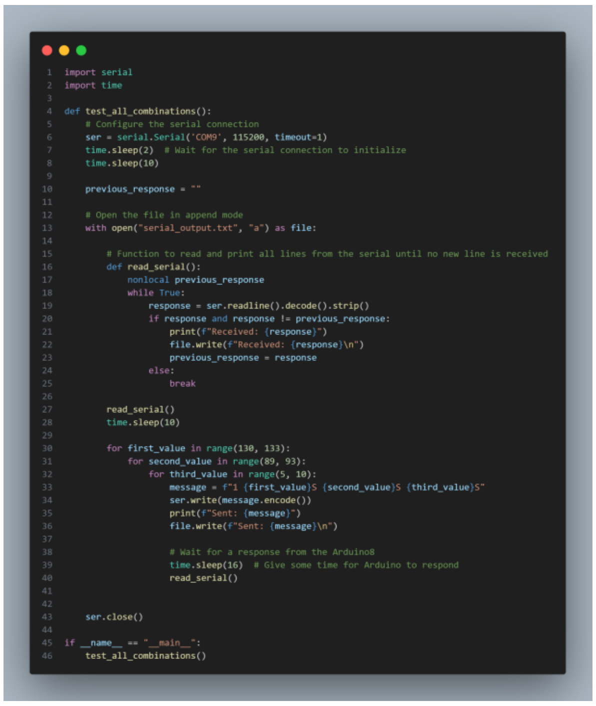


fig 10 : Audio file

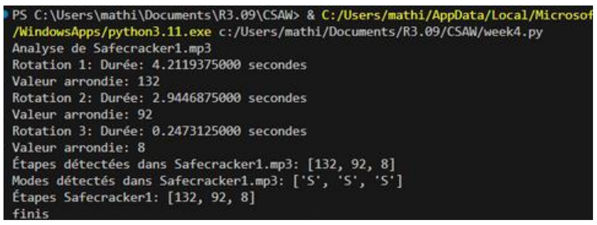
Then, when I tried to put them, it told me that it was not the right one. However, being relatively confident about the code produced, I created a program trying each combination at -2, +2 around the value that my first program gave by sending them to the serial monitor and getting back what was written in it.



```python
import serial
import time

def test_all_combinations():
    # Configure the serial connection
    ser = serial.Serial('COM9', 115200, timeout=1)
    time.sleep(2)  # Wait for the serial connection to initialize
    time.sleep(10)

    previous_response = ""

    # Open the file in append mode
    with open("serial_output.txt", "a") as file:

        # Function to read and print all lines from the serial until no new line is received
        def read_serial():
            nonlocal previous_response
            while True:
                response = ser.readline().decode().strip()
                if response and response != previous_response:
                    print(f"Received: {response}")
                    file.write(f"Received: {response}\n")
                    previous_response = response
                else:
                    break

        read_serial()
        time.sleep(10)

        for first_value in range(130, 133):
            for second_value in range(89, 93):
                for third_value in range(5, 10):
                    message = f"1 {first_value}S {second_value}S {third_value}S"
                    ser.write(message.encode())
                    print(f"Sent: {message}")
                    file.write(f"Sent: {message}\n")

                    # Wait for a response from the Arduino8
                    time.sleep(16)  # Give some time for Arduino to respond
                    read_serial()


    ser.close()

if __name__ == "__main__":
    test_all_combinations()
```

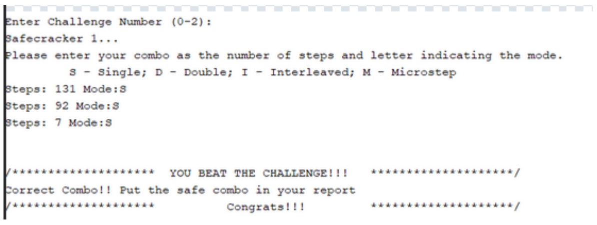
fig 11 : brute force python code

After 4 hours of testing, the program gave me a text file with the corresponding sequence. I was able to discover that the program had found the right sequence, and I restarted a serial monitor to check it.



```
PS C:\Users\mathi\Documents\R3.09\CSAW> & C:/Users/mathi/AppData/Local/Microsof
/WindowsApps/python3.11.exe c:/Users/mathi/Documents/R3.09/CSAW/week4.py
Analyse de Safecracker1.mp3
Rotation 1: Durée: 4.2119375000 secondes
Valeur arrondie: 132
Rotation 2: Durée: 2.9446875000 secondes
Valeur arrondie: 92
Rotation 3: Durée: 0.2473125000 secondes
Valeur arrondie: 8
Étapes détectées dans Safecracker1.mp3: [132, 92, 8]
Modes détectés dans Safecracker1.mp3: ['S', 'S', 'S']
Étapes Safecracker1: [132, 92, 8]
finis
```

fig 12 : running brute force python file



```
Enter Challenge Number (0-2):
Safecracker 1...
Please enter your combo as the number of steps and letter indicating the mode.
         S - Single; D - Double; I - Interleaved; M - Microstep
Steps: 131 Mode:S
Steps: 92 Mode:S
Steps: 7 Mode:S


/********************  YOU BEAT THE CHALLENGE!!!   ********************/
Correct Combo!! Put the safe combo in your report
/********************          Congrats!!!         ********************/
```

fig 13 : safe cracker 1 completed

correct combo: 131 92 7

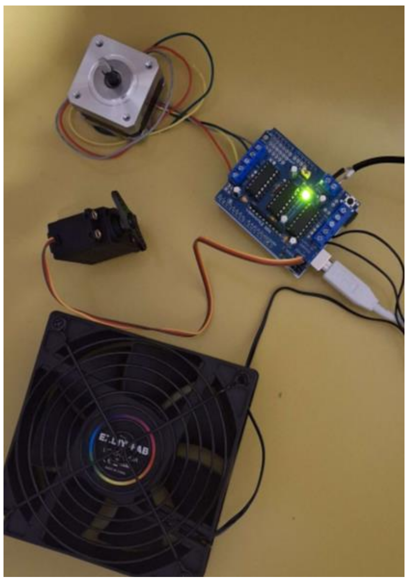


fig 14 : Our setup

VI. CONCLUSION

In conclusion, throughout this month of challenges, we have significantly enhanced our skills in cybersecurity and logical thinking. This experience has been a valuable opportunity for growth and learning.