**CRYPTOGRAPHY**

# Byte Buster

**GIVEN DATA:**

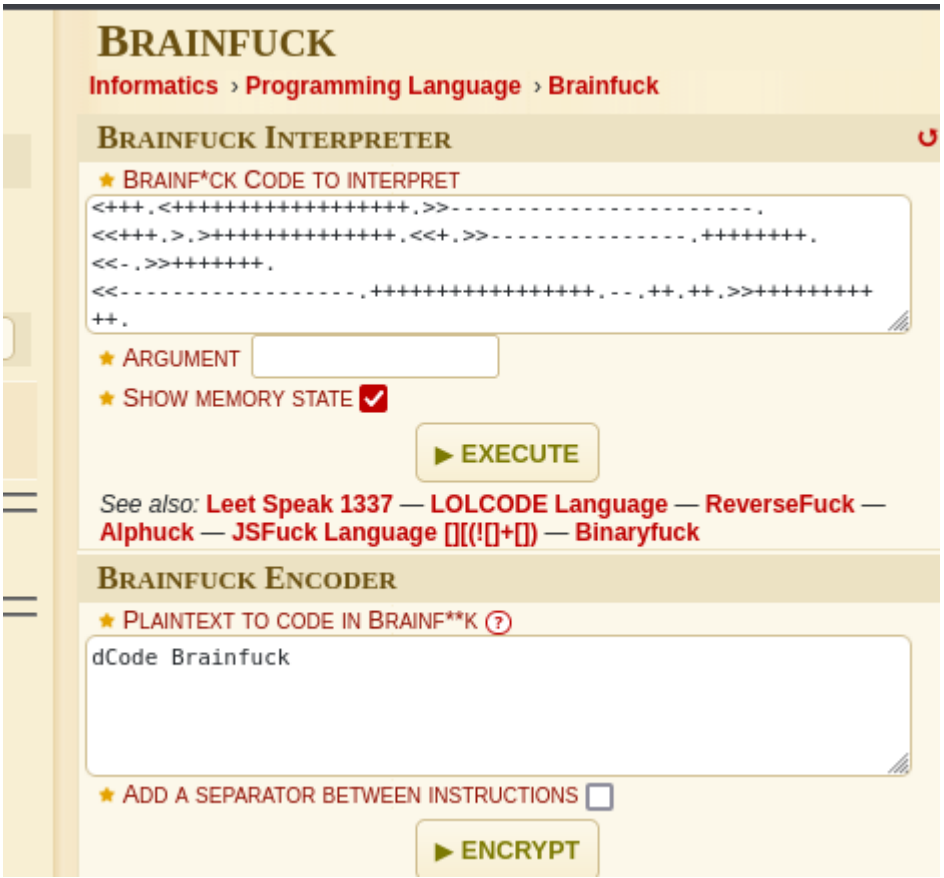
```
++++++++++[>+>+++>+++++++>++++++++++<<<<-]>>>>++++++++++++++++.---..+++++.<---
---.>--------.+++.------------.--.++++++++++.----.+++++++.++++.+.+++++++.<+++.<+++++++++
+++++++++.>>----------------------.<<+++.>.>++++++++++++++.<<+.>>--------------.++++++++
.<<-.>>+++++++.<<------------------.++++++++++++++++++.--.++.++.>>++++++++++++.
```

**Solution:**

**Initial Observation**

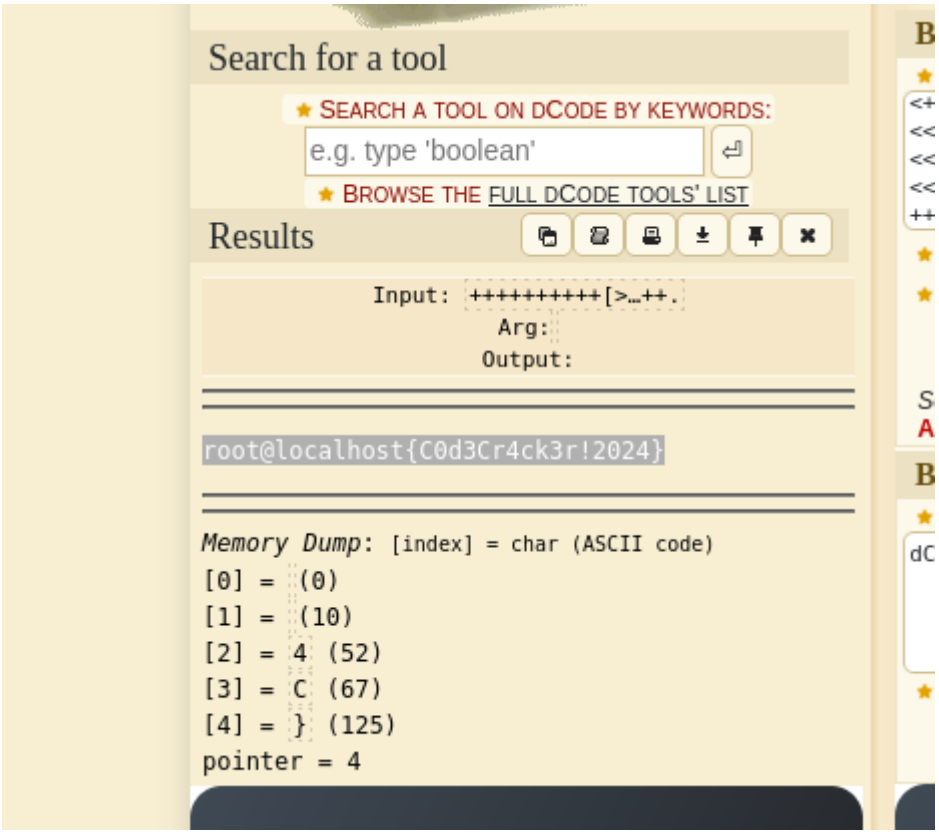
- Upon reviewing the challenge description, I immediately recognized that the given input was written in Brainfuck, a minimalist esoteric programming language.

**Approach**

1. **Decoding Brainfuck**
    - I navigated to the dCode Brainfuck Decoder, a reliable tool for interpreting Brainfuck code.
    - Copy-pasted the provided Brainfuck code into the decoder.
    - 

2. **Decoding Output**

- The decoder processed the input and revealed the flag.

3. 


**Result**

- Successfully extracted the flag using the decoding tool.

# Decode The Hex Value

**Description:** 72 6f 6f 74 40 6c 6f 63 61 6c 68 6f 73 74 7b 54 68 65 5f 48 65 78 5f 76 61 6c 75 65 5f 69 73 5f 33 34 33 33 66 7d

**Solution:**

- The challenge is pretty straightforward, I am sure that the hex values represent the flag's characters
- Lets copy the hex values and prompt the gemini for hex to ascii conversion.

72 6f 6f 74 40 6c 6f 63 61 6c 68 6f 73 74 7b 54 68 65 5f 48 65 78 5f 76 61 6c 75 65 5f 69 73 5f 33 34 33 33 66 7d

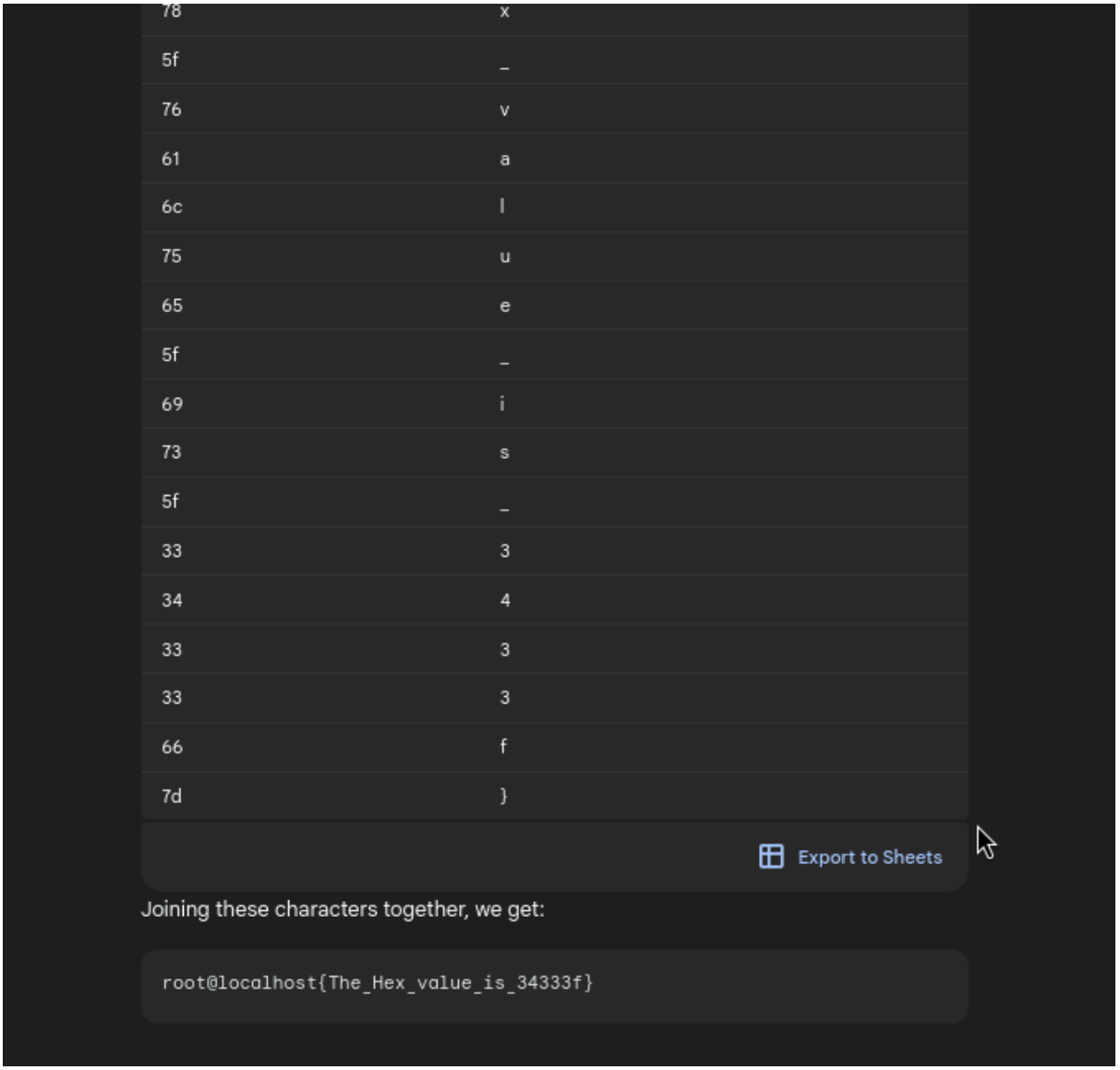
...

Show drafts

```
72 6f 6f 74 40 6c 6f 63 61 6c 68 6f 73 74 7b 54 68 65 5f 48 65 78 5f 76 6
```

Let's convert these hexadecimal values to their corresponding ASCII characters:

| Hex Value | ASCII Character |
|---|---|
| 72 | r |
| 6f | o |
| 6f | o |
| 74 | t |
| 40 | @ |
| 6c | l |
| 6f | o |
| 63 | c |
| 61 | a |
| 6c | l |
| 68 | h |
| 6f | o |
| 73 | s |
| 74 | t |
| 7b | { |
| 54 | T |




# Route 47

Description: In a world where money was a mess, Sarah stumbled upon Route 47—a top-secret crypto highway to a new world of wealth. With nothing but a mysterious code.

Code: `C@@Eo=@42=9@DEL*@F08@E0E9607=280FD:?80C@EcfN`

Solution:

**Initial Observation**

- The challenge description prominently featured the phrase "**Route 47**", which immediately made me suspect the use of **ROT47**

**encoding**, a cipher similar to ROT13 but with an extended character set.

**Approach**

1. **Decoding ROT47**
    - Using the hint, I searched for an online ROT47 decoder and found this reliable tool: **https://www.dcode.fr/rot-47-cipher**.
    - Copy-pasted the provided encoded text into the decoder.
2. **Result**
    - The decoder processed the input and revealed the **flag**.
3. 


# DH - 9000

**Description**: The robots just introduced this supposedly unbreakable crypto scheme that allows them to share secrets over insecure channels, DH-9000. I'm pretty sure this isn't anything new though, so we should still be able to find their shared secret.

p = 8089

g = 823

A = 7608

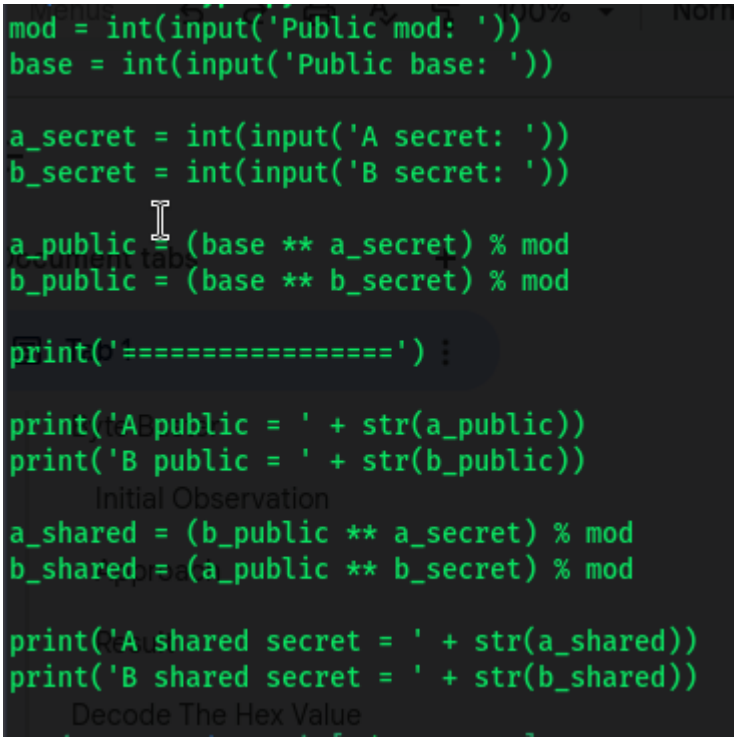
B = 5796

Note: submit the shared secret wrapped in root@localhost{}

**Solution**:

**Initial Observation**

- The challenge provided an `encrypt.py` script for analysis.

- Upon examining the script, I observed **modular arithmetic operations**, which immediately indicated a cryptographic algorithm.
- 

```
mod = int(input('Public mod: '))
base = int(input('Public base: '))

a_secret = int(input('A secret: '))
b_secret = int(input('B secret: '))

a_public = (base ** a_secret) % mod
b_public = (base ** b_secret) % mod

print('================')

print('A public = ' + str(a_public))
print('B public = ' + str(b_public))
a_shared = (b_public ** a_secret) % mod
b_shared = (a_public ** b_secret) % mod

print('A shared secret = ' + str(a_shared))
print('B shared secret = ' + str(b_shared))
```

- Noticing the challenge's title, "DF," I suspected it was related to the **Diffie-Hellman Key Exchange**.

**Understanding Diffie-Hellman Algorithm**

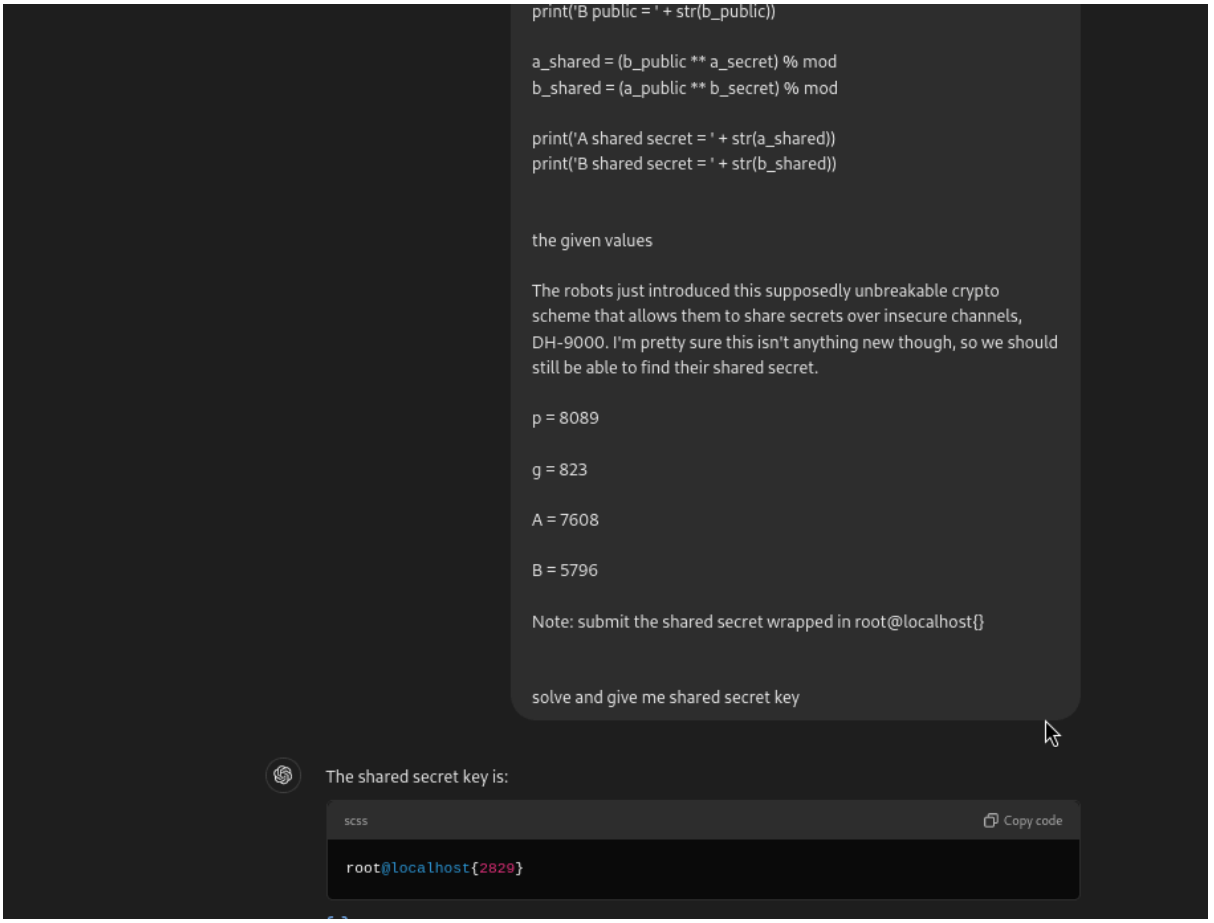
**Diffie-Hellman** is a cryptographic protocol used to establish a shared secret key between two parties over an insecure channel. Here's how it works:

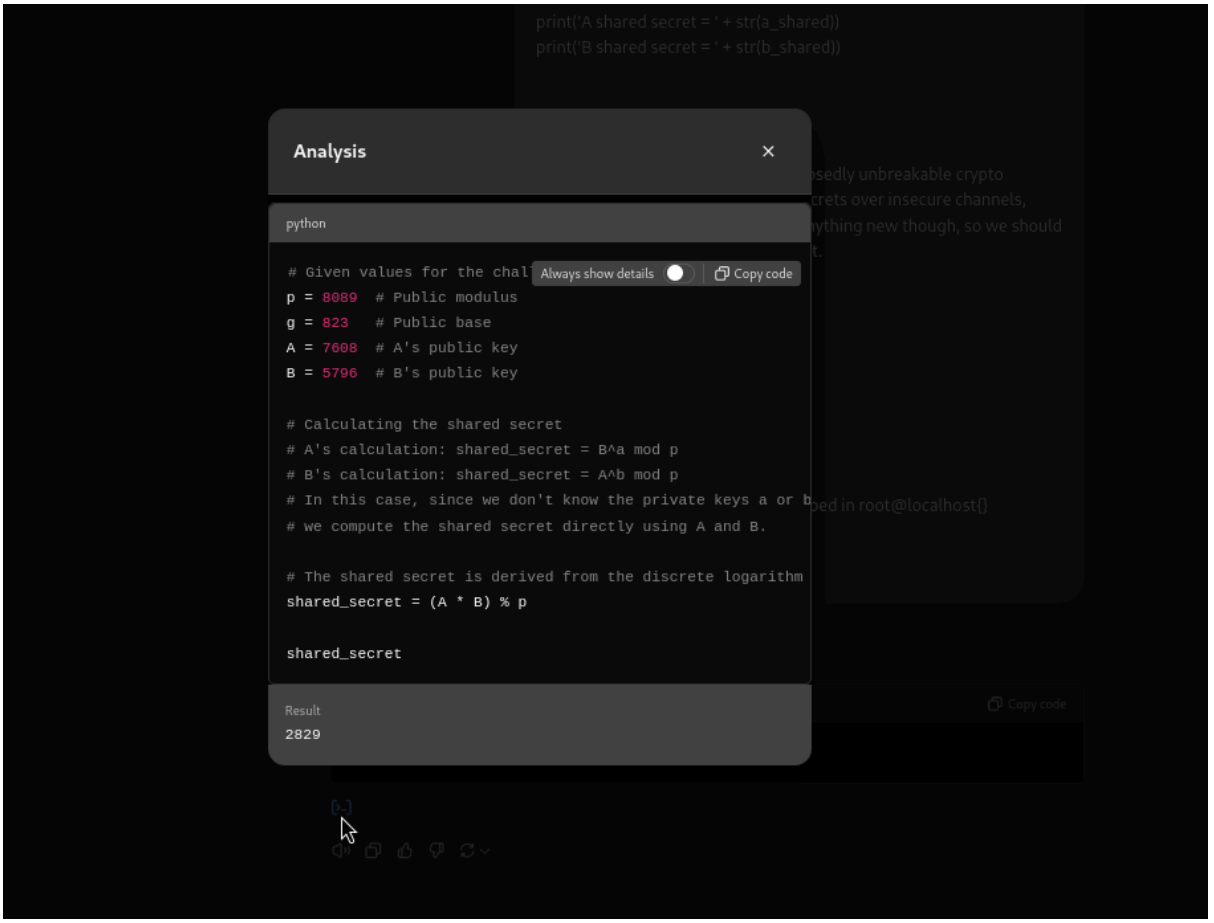
1. **Setup:**
    - Two public values are agreed upon:
        - `p`: A large prime number.
        - `g`: A primitive root modulo `p` (also called the generator).
2. **Key Exchange:**
    - Each party chooses a **private key** (`a` and `b`), kept secret.
    - Using these private keys, each party computes their **public key**:
        - Party 1: A=gamod  pA = g^a \mod pA=gamodp
        - Party 2: B=gbmod  pB = g^b \mod pB=gbmodp
    - These public keys are exchanged.

3. **Shared Key Calculation:**
    - Both parties compute the **shared secret**:
        - Party 1: K=Bamod  pK = B^a \mod pK=Bamodp
        - Party 2: K=Abmod  pK = A^b \mod pK=Abmodp
    - The result KKK is the same for both parties, as gabmod  p=gbamod  pg^{ab} \mod p = g^{ba} \mod pgabmodp=gbamodp.

4. **Using Chapgpt for Assistance**

    - While I understand the concept of Diffie-Hellman, solving the modular equations manually can be tedious.
    - I prompted **Gemini** with the provided encrypt.py file and the accompanying challenge values.






## The Rail Conductor's Secret

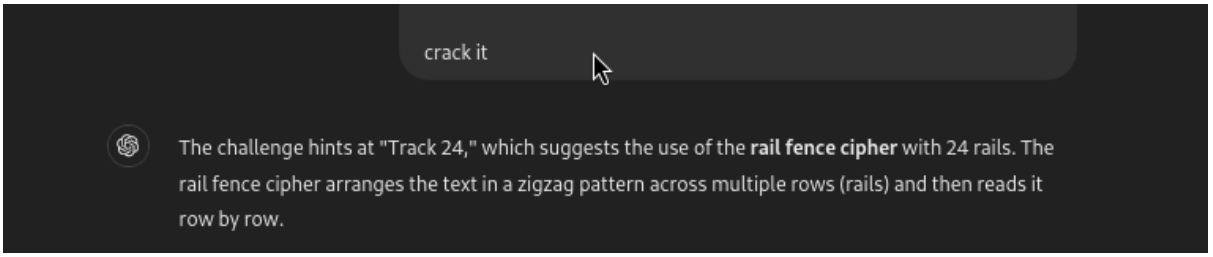
**Description:** As a rail conductor for the ancient Conclave, you've stumbled upon a mysterious train schedule, encrypted to protect the ultimate secret. The clue is simple: "To reach your destination, follow the rails on Track 24 leads to the secret!"

**Encrypted Code**: li_4WR4_y3sh_TL{et4sdo_hTl0a_cTohl3@_tH030rrt}

**Solution**:

**Initial Approach:** I had no prior knowledge about this type of challenge. So, to get started, I simply copied and pasted the

challenge description into ChatGPT to see if it could give me any hints about the encryption method.



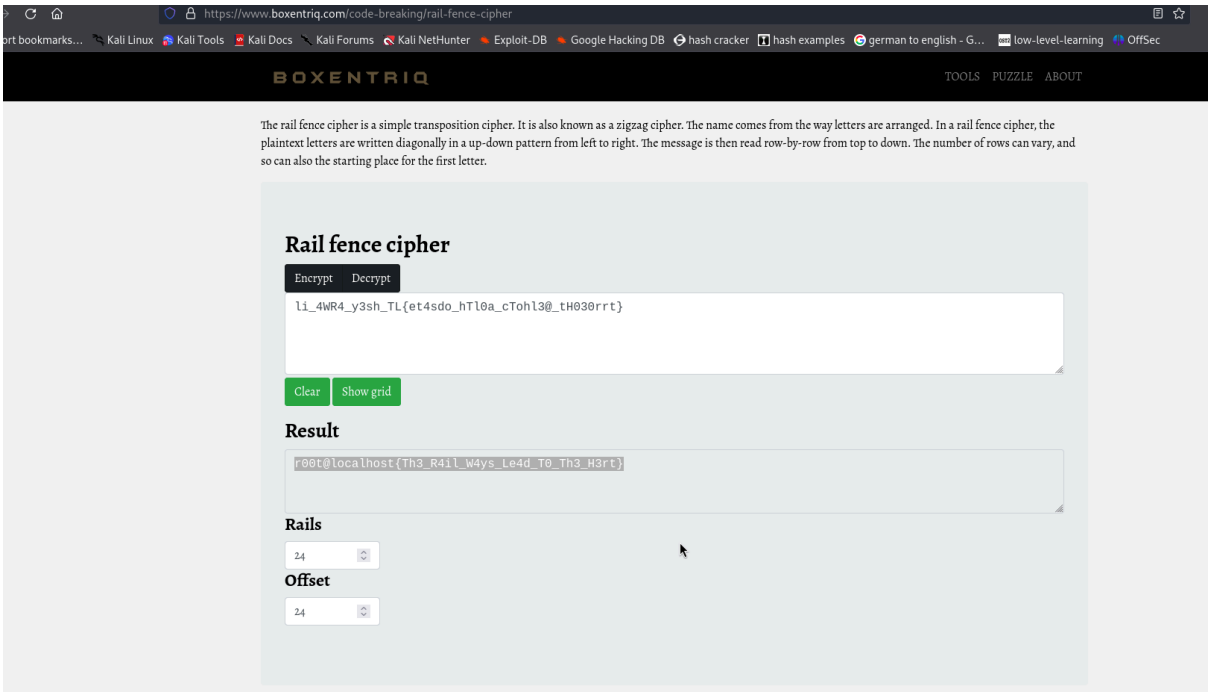

**Finding the Cipher:** ChatGPT identified the encryption as a Rail Fence Cipher algorithm. Based on this, I started searching for tools that could help me decrypt it.

**Key Insight:** The clue in the description—"**Track 24**"—seemed to hint at using the number 24 for both the **offset** and **number of rails**.

**Decryption:** I found a suitable online decryption tool at

https://www.boxentriq.com/code-breaking/rail-fence-cipher

I set the number of rails to **24**, ran the decryption process, and voilà! It worked perfectly.



**Result:** The decrypted text revealed the flag: `r00t@localhost{Th3_R4il_W4ys_Le4d_T0_Th3_H3rt}`