Looking through main, I tried to reverse engineer the encryption process, but I failed. As a result, I decided to see what happens to message.txt, the input.

First (according to IDA), she message is read into v26, then saved into v16

Afterwards, any x's are transformed into _

Afterwards, if the text length is odd, it adds an x which I assume is also turned into _

Afterwards, from my understanding, it takes every 2 characters, encrypts them using some sort of built table, and converts them.

After a bit of testing, it turns out my assumptions were correct.

This means one simple thing: the program encrypts characters 2 by 2. We can bruteforce each 2 possible chars until we recover the full plaintext.

```
import subprocess

encoded_message =
"460047464095481418042432979042431251934048439466974607954443349"
plaintext = ""
while True:
    for i in range(26):
        for j in range(26):
            if i == j:
                continue
            two_char_combo = chr(i + 97) + chr(j + 97)
            partial_plaintext = plaintext + two_char_combo
            with open("message.txt", "w") as file:
                file.write(partial_plaintext)
            result = subprocess.run(["./rev_secret_secret.o"],
capture_output=True, text=True)
            hex_string = None
            for line in result.stdout.splitlines():
                if line.startswith("Encoded:"):
                    hex_string = line.split("Encoded:")[1].strip()
                    break
            if hex_string and encoded_message.startswith(hex_string):
                plaintext += two_char_combo
                plaintext = plaintext.replace('x', '_')
                print(f"Matching prefix found! Current plaintext: {plaintext}")
                if hex_string == encoded_message:
                    print(f"Full match found! Final plaintext: {plaintext}")
                    exit(0)
```

Additional note: I skipped the cases when i==j because I noticed the final string isn't formatted properly. After a test, turns out yy encodes the same as yx, so there is no reason to have 2 identical chars. I probably missed this when studying the source code.

Made with love by: AndreiCat