Challenge: Vault breaker

# Challenge Description :

**Money maker, Big Boy Bonnie has a crew of his own to do his dirty jobs. In a tiny little planet a few lightyears away, a custom-made vault has been found by his crew. Something is hidden inside it, can you find out the way it works and bring it to Bonnie?**

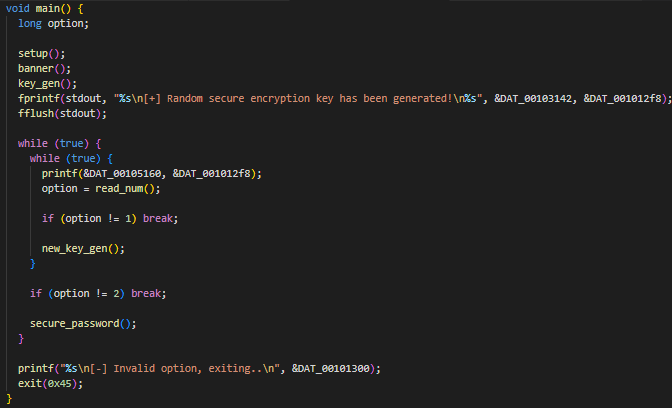
# Context :

## You are given a compiled binary file that we need to decompile, during decompiling the file you need to analyze the binary code and find an exploit towards this file, so you can exploit the remote version.

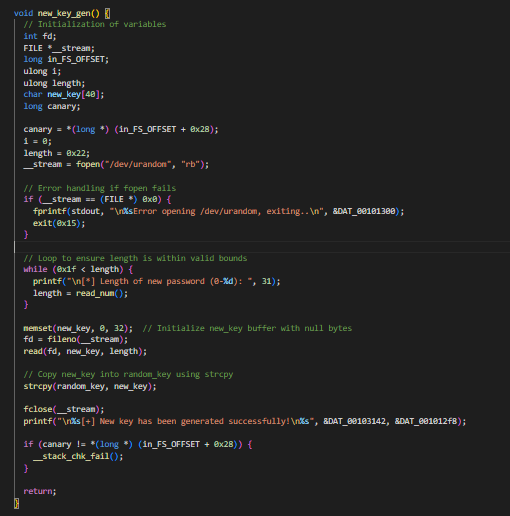
## To solve the challenge, you need to exploit the fact that DataStore can be interpreted both as a string and as an integer, and they share the same memory..

# Flag :

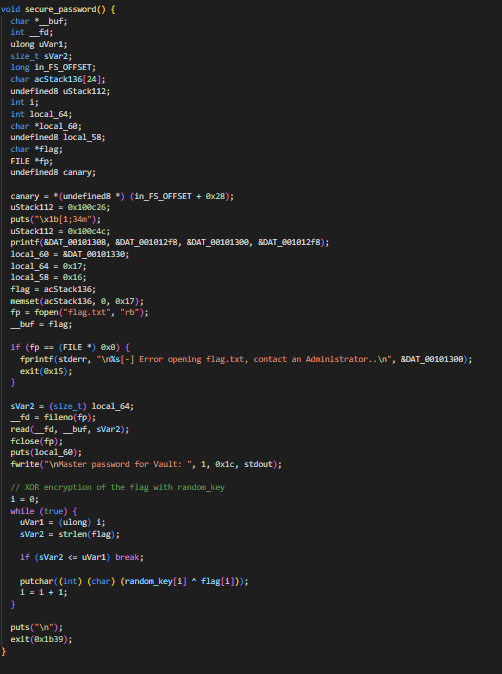
* **When we run the program, it gives us two options. The first option lets us create a new encryption key using random data from /dev/urandom.**

****

* **However, there’s a big mistake in how the new key is handled. The program uses strcpy to copy the new key into random\_key, but strcpy treats the key as a string that ends with a null byte.**

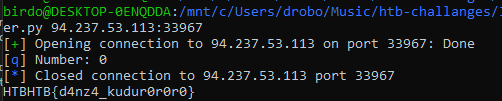
****

* **If the new key has any null bytes, it can cut off the rest of the key. This mistake can lead to random\_key being filled with null bytes if the new key is shorter than the old one.**
* **If we choose the second option, "Secure the Vault," the program reads a password from flag.txt and encrypts it using a simple XOR with random\_key.**

****

* **Because of the bug in new\_key\_gen, an attacker can change random\_key to be all null bytes, making the encryption useless.**
* **This allows the attacker to get the plain text password from flag.txt without needing to decrypt it using the flaw in how strcpy works and the XOR method.**
* **Basically the new\_key\_gen uses strcpy to copy new\_key to random\_key.If new\_key contains null bytes, strcpy will stop copying prematurely, potentially leaving random\_key filled with null bytes. Providing a length to a new\_key\_gen that starts at 31 characters and decreases.**
* **Eventually, when new\_key has a length of 0, random\_key will be completely nullified due to strcpy stopping at the first null byte encountered.**
* **The python script to complete this will be linked, as it's on my github, you can either run it locally side by side with the compiled Binary file as well as sending the payload to the hosted instance of the file.**

**[** [**Link**](https://github.com/Birdo1221/HTB-writeup/blob/main/Vault%20breaker.py) **]**

****

* **The Flag given is : HTB{d4nz4\_kudur0r0r0}**