Assignment 6 -

Confundo

Names : Shree Vishnu, Vikram

Roll Numbers: CS19B045, CS19B021

Course: CS6570 - Secure Systems Engineering

Submission Date: 12/04/2022

Approach:-

- The C code is in **function.c** and is well-documented.
- We use 100 random ASCII strings generated from the script in the code **generation** folder to make our password. The script also generates a key(mapping) on how to get the password string using these 100 strings. The script also prints the correct password in the end.
- We use the codes and mapping from the above point in **function.c.** It also has some unnecessary headers and irrelevant junk and especially implements 2 other functions from the function list. A string relevant to **“Sum of sums function”** will be seen if we use the strings utility on our executable. We also have functions that do quicksort also!
- Now, all our obfuscation becomes useless if we let the **abstraction of a function be there finally.** An attacker can simply insert a **breakpoint at main()** and if he somehow

gets to the **mapping (the crux of our security check),** then we are doomed.

- Also, if we let the compiler do **optimisations like dead code removal, liveness analysis and other aggressive loop optimisations, then the attacker's job becomes easy.**
- So, we have our **really long and manually extracted** compilation command for **function.c** in **comp.cmd.** This removes the user-defined function's names and produces a **long chunk of code under the .text section!** There are **no symbol tables for debugging and static analysis will be a pain** because of the sheer length and absence of optimisations and functional abstraction. We have compiled our code in our **Ubuntu 20.04 system but the executable works on the lab VM as well.**
- Dynamic analysis is also difficult because there are no functions as such and hence, the attacker must manually put breakpoints at instruction addresses and this is really difficult.

- The length of the password string(=100) makes it difficult to do automation as it will take exponential time to check all the guesses.
- Finally, you may use **tester.py** with the correct password as in **code generation/out.out** to test our executable.