‘Love Potion’ Reversing Walkthrough

Tools i will be using :

- GDB
- Decompiler (IDA, Ghidra ...)
- Text Editor for note taking

First reflexes :

-File :

```
salma@babylove:~/myChallenges/Love_Potion$ file love_potion.exe
love_potion.exe: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamicall
y linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=e86910e699365e8e49d
5bbed30b75d926d837603, for GNU/Linux 3.2.0, not stripped
```

-Strings :

After executing *strings love_potion.exe*, we will start noticing somethings :

```
35T74
h1AorC
0XhYc
```

There are suspicious string.

-Executing :

We proceed by giving the right to execute by running *chmod +x cupid.exe*
Then run the process using *./cupid.exe*

The process needs an argument passed with the execution.
To get that information we are going to need another tool. Let’s try decompiling the exe file using IDA PRO decompiler.

Decompiling + Static Analysis :

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  if ( argc == 1 )
  {
    printf("Help Sabrina find her post-it.");
  }
  else if ( (unsigned int)func_369((char *)argv[1]) )
  {
    printf("Well Done !");
  }
  else
  {
    printf("That's not it !");
  }
  return 0;
}
```

This is the main function decompiled. We notice that our input is given as an argument to the function *func_369*
Then we continue to get into the functions that takes the input as args.

```c
_int64 __fastcall func_234(char *a1)

  int i; // [rsp+1Ch] [rbp-34h]
  int j; // [rsp+20h] [rbp-30h]
  int k; // [rsp+24h] [rbp-2Ch]

  if ( strlen(a1) <= 8 )
    return 0LL;
  for ( i = 0; i < strlen("35T74"); i += 2 )
  {
    if ( a1[strlen(a1) - i / 2 - 1] != a35t74[i] )
      return 0LL;
  }
  for ( j = 1; j < strlen("h1AorC"); j += 2 )
  {
    if ( a1[strlen(a1) - j / 2 - 4] != aH1aorc[j] )
      return 0LL;
  }
  for ( k = 0; k < strlen("0XhYc"); k += 2 )
  {
    if ( a1[strlen(a1) - k / 2 - 7] != a0xhyc[k] )
      return 0LL;
  }
  return 1LL;
```

We can now say that we have found our box : *func_234* !
This function is generating dynamically generating the flag from our strings and comparing character by character.

The *func_234* function checks if an input string conforms to specific patterns. It first verifies that the input has a length of at least 9 characters. Then, it compares the input against our three strings by examining characters at specific positions. If any character doesn't match its corresponding pattern, the function returns 0. Otherwise, if the input matches all patterns, it returns 1. This function plays a crucial role in validating input strings against expected patterns for further processing.

So we take it block by block and see what it is that we are comparing.

```
for ( i = 0; i < strlen("35T74"); i += 2 )
{
  if ( a1[strlen(a1) - i / 2 - 1] != a35t74[i] )
    return 0LL;
}
```

We are comparing the last 3 characters of the flag with "3T4" .
in the second block :

```
for ( j = 1; j < strlen("h1AorC"); j += 2 )
{
  if ( a1[strlen(a1) - j / 2 - 4] != aH1aorc[j] )
    return 0LL;
}
```

We are comparing the 3 middle characters with "1oC" .
in the last one :

```
for ( k = 0; k < strlen("0XhYc"); k += 2 )
{
  if ( a1[strlen(a1) - k / 2 - 7] != a0xhyc[k] )
    return 0LL;
}
```

we are comparing the first 3 characters with "ohc"
we can now put them together and have the flag.

Thank you <3