Questions

1. What is the command to compile the files with extra symbols that are useful for GDB?

   Ans: gcc -g GDBassign.c blowfish.c

2. What's the address of stuff?

   Ans: gdb a.out
   (gdb) break main
   (gdb) run
   (gdb) print &stuff
   0x7fffffffde50: 0x72657645

3. What's the address of stuff[0]?

   Ans:- x &stuff[0]

   0x7fffffffde50: 0x72657645

4. Do we expect these to be the same? Why? Explain what the [ ] operator does in C.

   Ans: We do since they are the same because in C, the name of an array is a pointer to its first element. Thus, stuff and &stuff[0] represent the same address. The [] operator is short for *(ptr + i). Where i would be the index.

5. In Blowfish_Init( ), what is the value of key?

   Ans:- key = 0x400be0 "LAME_KEY"

6. What command(s) did you type in order to learn this?

   Ans: (gdb) break Blowfish_Init
   D key

7. In Blowfish_Init( ), what are the values of i and j after the nested for loops have finished? i.e., after: for (i = 0; i < 4; i++) { for (j = 0; j < 256; j++) ctx->S[i][j] = ORIG_S[i][j]; }

   Ans: I = 4 j = 256

8. What command(s) did you type in order to learn this?

   Ans: Break blowfish_int

   s //until reach blowfish function

   s //until in for loop

   u // to finish for loop

   p i

   p j

   j

9. Before the Blowfish_Encrypt function is called, what is the value of stuff[3] (for each, print the value, and the command used to obtain the value): o in hex? o in binary? o as a float? o as 4 chars?

   Ans:
   break main
   N //until printf("encrypting buffer..\n");
   In hex:
   P /x stuff[3]
   0x772612073
   In Binary:
   P /x stuff[3]
   11101110110000100100000001110011
   As a float:
   P /t stuff[3]
   4.56611305e+33
   As 4 chars:
   X /4c &stuff[3]
   115 's', 32 ' '. 97 'a', 119 'w'

10. Before the Blowfish_Encrypt function is called, what is the value of stuff if we treat it as a string? (You don't have to write the whole string. Just describe what's there.) What was the command typed in order to obtain this value?

    Ans: "Everything is awesome\n Everything is cool when you're part of a team\n Everything is awesome\n When we're living our dream\n \n Everything is better when we stick together\n Side by side\n You and I\n Gonna "

11. What is the value of x the first time that the function F() in Blowfish.c is called?

    Ans: Breakpoint 2, F (ctx=0x7fffffffc7d0, x=1753098189) at blowfish.c:550

550 d = x & 0x00FF;
(gdb) p x
$10 = 1753098189

12. What is the output if we run GDB's backtrace (abbreviated "bt") command inside the function F() in Blowfish.c the first time F() is called? Briefly explain the output of the command in your own words.

    (gdb) bt

    #0 F (ctx=0x7fffffffc7d0, x=1753098189) at blowfish.c:550

    #1 0x0000000000400749 in Blowfish_Encrypt (ctx=0x7fffffffc7d0, xl=0x7fffffffc7a4,

    xr=0x7fffffffc7a0) at blowfish.c:602

    #2 0x0000000000400974 in Blowfish_Init (ctx=0x7fffffffc7d0, key=0x400be0

    "LAME_KEY", keyLen=8) at blowfish.c:754

    #3 0x0000000000400581 in main () at GDBassign.c:228

    Explanation: The backtrace command in GDB traces the sequence of function calls from main() up to the point of execution inside F(). It outlines how the program starts with Blowfish_Init(), progresses to Blowfish_Encrypt(), and finally reaches F(). Each entry in the call stack details function arguments and their current values, providing insight into data flow and interactions within the program. This trace is invaluable for debugging, allowing developers to pinpoint where and how errors or unexpected behaviors arise by following the call path backward.