Challenge:Questionnaire

Challenge Description :

It's time to learn some things about binaries and basic c. Connect to a remote server and answer some questions to get the flag.

Context :

- This is a questionnaire about the C programming language, you get given a compiled program and you need to answer the question according to the questionnaire via a server that you connect to via netcat.
- The questions will also give little hints and what tools you can use to find the answer but isnt always direct with it.
- A similar challenge like this is called Lesson.

Notes :

- Tools:
    - Cli Terminal - A Terminal to Netcat into the server to answer the questions you just need to type.

Flag :

- Start the instance, and then Netcat into the server with the IP:PORT. First look at the terminal I got when I first connected seems to be a questionnaire.
- I don't think this challenge is really important as there are better challenges to learn from.
- The answers to the Questions goes as:

```
gef➤  checksec
Canary                        : ✘
NX                            : ✓
PIE                           : ✘
Fortify                       : ✘
RelRO                         : Partial

[*] Question number 0x1:

Is this a '32-bit' or '64-bit' ELF? (e.g. 1337-bit)

>> 64-bit

      Correct

[*] Question number 0x2:

What's the linking of the binary? (e.g. static, dyna

>> dynamic

      Correct

[*] Question number 0x3:

Is the binary 'stripped' or 'not stripped'?

>> not stripped

      Correct

[*] Question number 0x4:

Which protections are enabled (Canary, NX, PIE, Fort

>> NX

      Correct
```

```
void vuln(){
    char buffer[0x20] = {0};
    fprintf(stdout, "\nEnter payload here: ");
    fgets(buffer, 0x100, stdin);
}

The first line declares a 0x20-byte buffer of characters
The second line calls 'fprintf()' to print a message to s
Finally, the third line calls 'fgets()' to read 0x100 byt
aforementioned buffer.

Then, there is a custom 'gg()' function which calls the s
flag. This function is never called by default.

void gg(){
    system("cat flag.txt");
}

Run the 'man <function_name>' command to see the manual p

[*] Question number 0x5:

What is the name of the custom function the gets called insi

>> vuln()

      Correct

[*] Question number 0x6:

What is the size of the 'buffer' (in hex or decimal)?

>> 02x20

      Wrong

[*] Question number 0x6:

What is the size of the 'buffer' (in hex or decimal)?
>> 0x20

      Correct

[*] Question number 0x7:

Which custom function is never called? (e.g. vuln())

>> gg()

      Correct
```

```
[*] Question number 0x8:

What is the name of the standard function that could trigger a Buffer Overflow? (e.g. fprintf())

>> fgets()

* * * * * * * * * * *
*                   *
*      Correct      *
*                   *
* * * * * * * * * * *

[*] Question number 0x9:

Insert 30, then 39, then 40 'A's in the program and see the output.

After how many bytes a Segmentation Fault occurs (in hex or decimal)?

>> 40

* * * * * * * * * * *
*                   *
*      Correct      *
*                   *
* * * * * * * * * * *

[*] Question number 0xa:

What is the address of 'gg()' in hex? (e.g. 0x401337)

>> 0x401176

* * * * * * * * * * *
*                   *
*      Correct      *
*                   *
* * * * * * * * * * *

Great job! It's high time you solved your first challenge! Here is the flag!

HTB{l34rn_th3_b451c5_b3f0r4_u_5t4rt}
```

- At the end we get the flag : HTB{l34rn_th3_b451c5_b3f0r4_u_5t4rt}