






bbbloat


bbbloat is a reverse engineering challenge where we are given a binary file, which also has an obfuscation vector in it.

Bbbloat 

 | 300 points 

Tags: picoCTF 2022 Reverse Engineering binary obfuscation

AUTHOR: LT 'SYREAL' JONES

Hints 

Description

(None)

Can you get the flag?

Reverse engineer this [binary](#).

Let's wget the binary file and download it into our directory

```

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ wget https://artifacts.picoctf.net/c/47/bbbloat
--2024-02-21 20:15:01-- https://artifacts.picoctf.net/c/47/bbbloat
Resolving artifacts.picoctf.net (artifacts.picoctf.net) ... 18.154.144.85, 18.154.144.107, 18.154.144.103,
...
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|18.154.144.85|:443 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 14472 (14K) [application/octet-stream]
Saving to: 'bbbloat'

bbbloat          100%[=====>] 14.13K  --.-KB/s   in 0s
2024-02-21 20:15:02 (225 MB/s) - 'bbbloat' saved [14472/14472]

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$

```

Let's now get more information about the file using the file command

```

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ file bbbloat
bbbloat: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64
/ld-linux-x86-64.so.2, BuildID[sha1]=1989eb2c7cb4aad23df277d8aed3c97482740d7a, for GNU/Linux 3.2.0, strip
ped

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$

```

We can see that it's an ELF 64-bit LSB PIE (or Position Independent Executable)

Let's mark it as an executable using:

```
chmod +x bbbloat
```

```

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ chmod +x bbbloat

```

Let's now try to run this executable

Note that this file only runs on the x86 architecture. Since I am running Kali on an arch64 machine, I had to move to another machine to run this executable.

```
(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ chmod +x bbbloat

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ ls
bbbloat

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ ./bbbloat
What's my favorite number? 21
Sorry, that's not it!
```

It gives me the following input. That looks interesting. Maybe we can try to reverse engineer this binary and try to find any hardcoded information.

It probably runs some loop to validate this information and might have some hardcoded information about a true condition.

Let's run Ghidra and try finding the code.

We'll have to install JDK 11 for it and set the path.

```

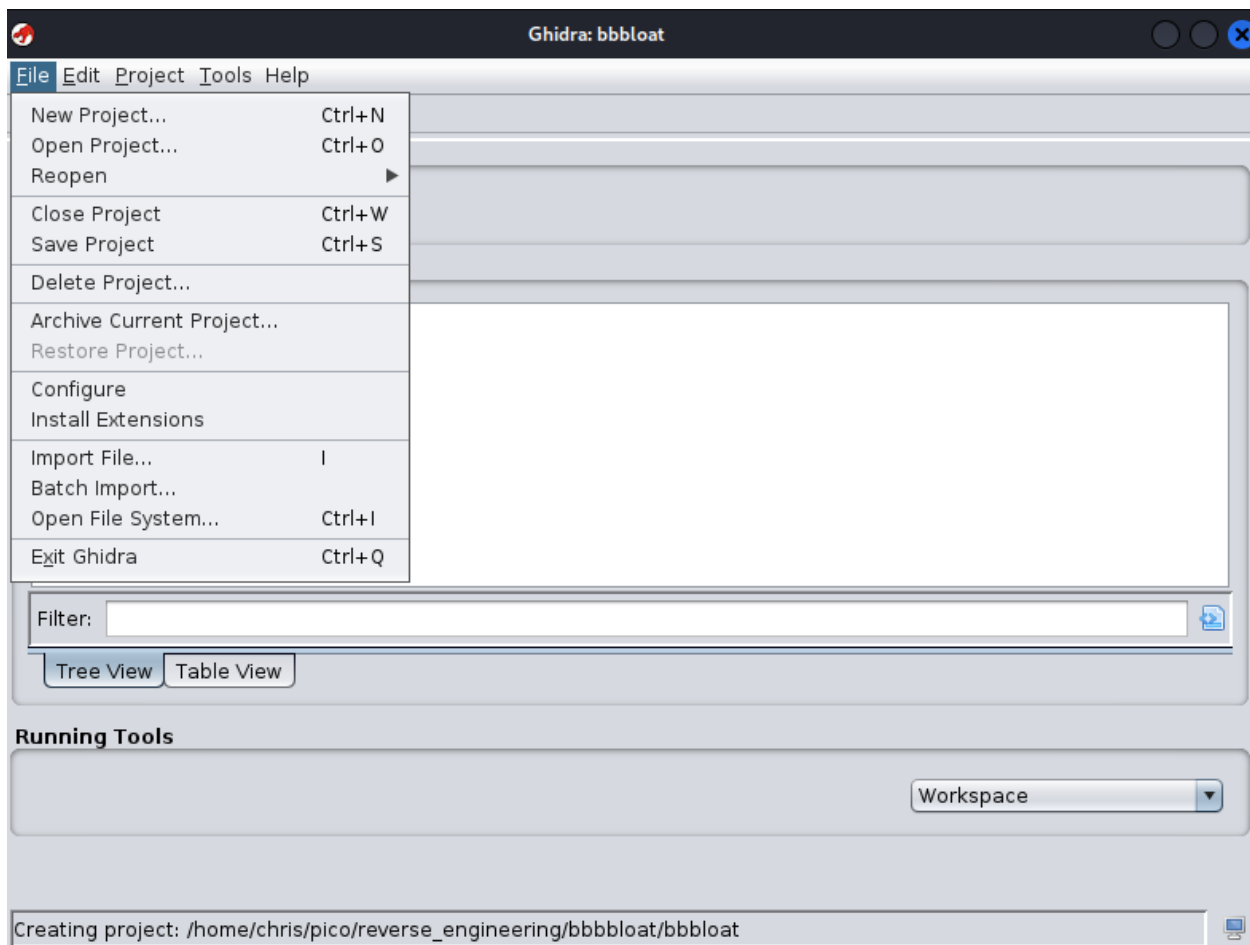
(chris@kali)-[~]
$ apt-cache search java11
default-jdk - Standard Java or Java compatible Development Kit
default-jdk-headless - Standard Java or Java compatible Development Kit (headless)
default-jre - Standard Java or Java compatible Runtime
default-jre-headless - Standard Java or Java compatible Runtime (headless)
openjdk-11-jdk - OpenJDK Development Kit (JDK)
openjdk-11-jdk-headless - OpenJDK Development Kit (JDK) (headless)
openjdk-11-jre - OpenJDK Java runtime, using Hotspot JIT
openjdk-11-jre-headless - OpenJDK Java runtime, using Hotspot JIT (headless)
openjdk-17-jdk - OpenJDK Development Kit (JDK)
openjdk-17-jdk-headless - OpenJDK Development Kit (JDK) (headless)
openjdk-17-jre - OpenJDK Java runtime, using Hotspot JIT
openjdk-17-jre-headless - OpenJDK Java runtime, using Hotspot JIT (headless)
openjdk-21-jdk - OpenJDK Development Kit (JDK)
openjdk-21-jdk-headless - OpenJDK Development Kit (JDK) (headless)
openjdk-21-jre - OpenJDK Java runtime, using Hotspot JIT
openjdk-21-jre-headless - OpenJDK Java runtime, using Hotspot JIT (headless)
openjdk-22-jdk - OpenJDK Development Kit (JDK)
openjdk-22-jdk-headless - OpenJDK Development Kit (JDK) (headless)
openjdk-22-jre - OpenJDK Java runtime, using Hotspot JIT
openjdk-22-jre-headless - OpenJDK Java runtime, using Hotspot JIT (headless)

(chris@kali)-[~]
$ sudo apt install openjdk-11-jdk
[sudo] password for chris:
Reading package lists... Done
Building dependency tree ... Done
Reading state information... Done
The following additional packages will be installed:
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless

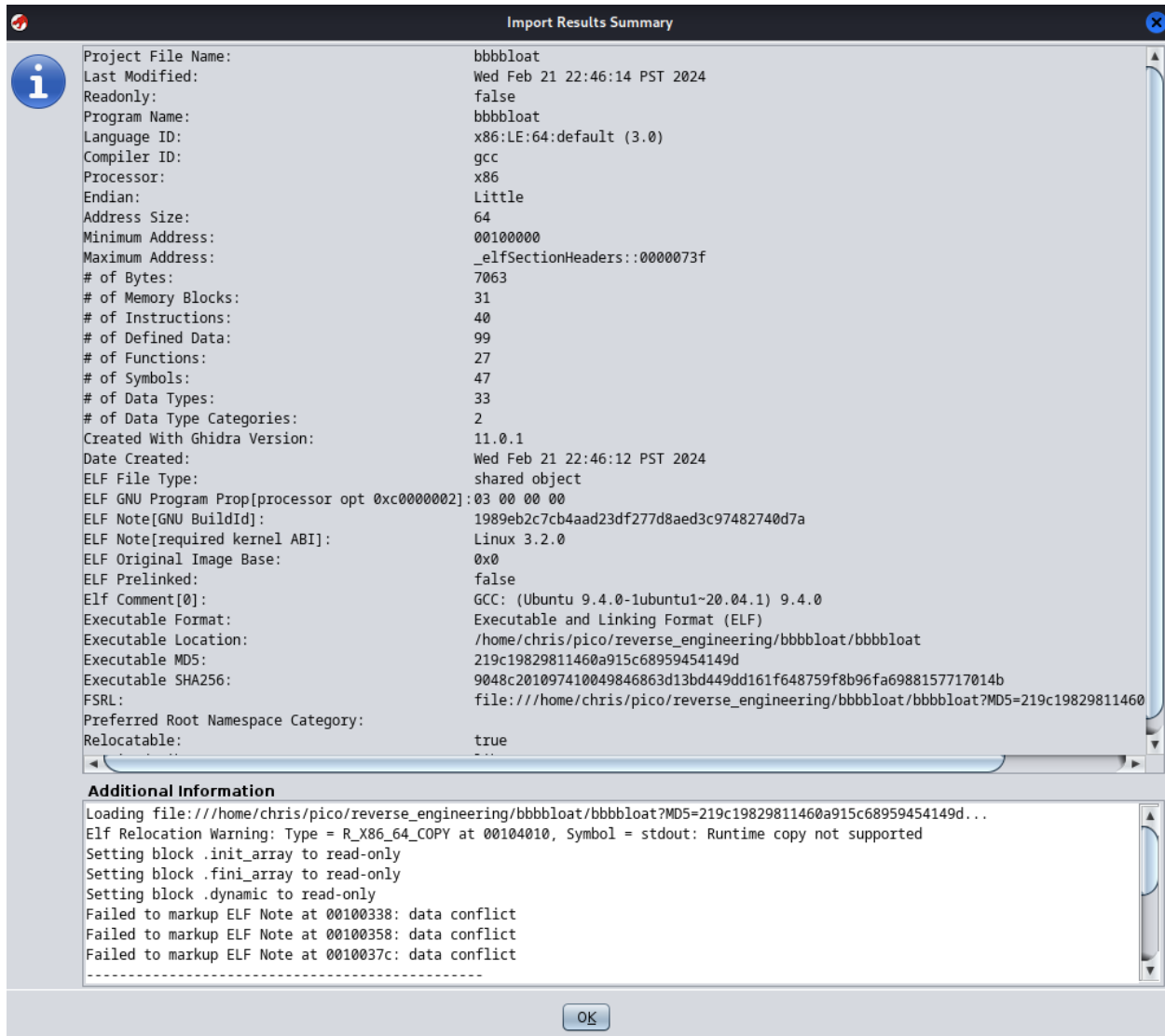
```

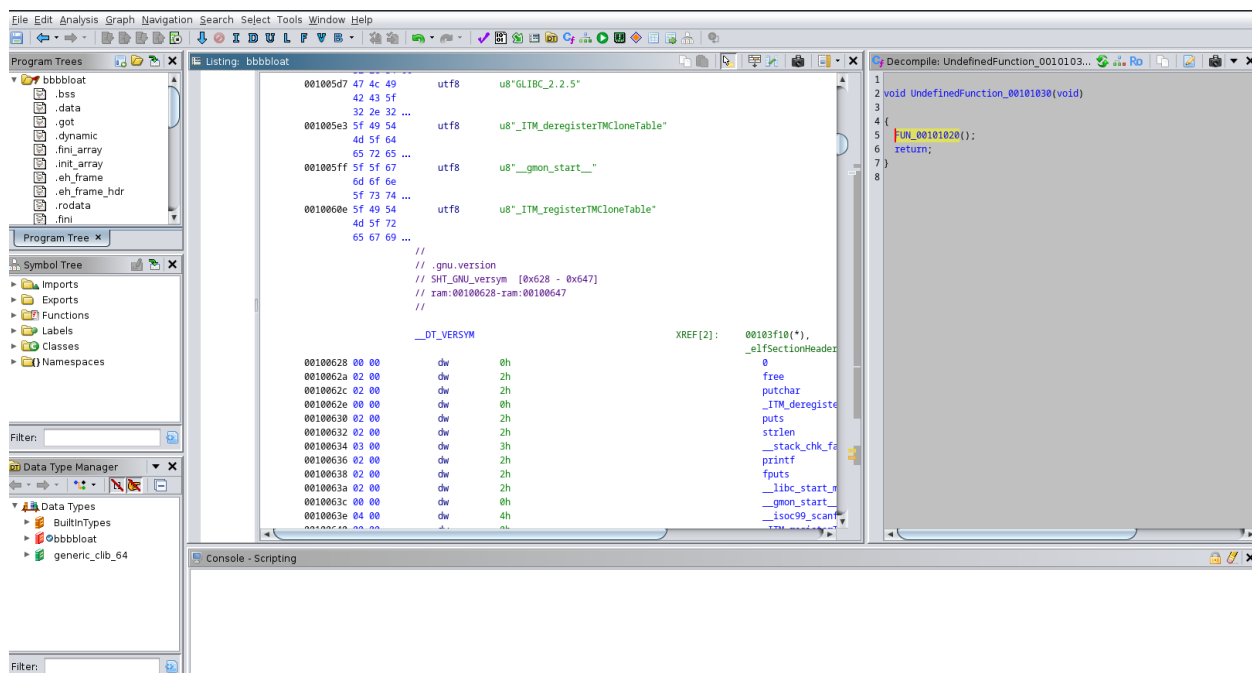
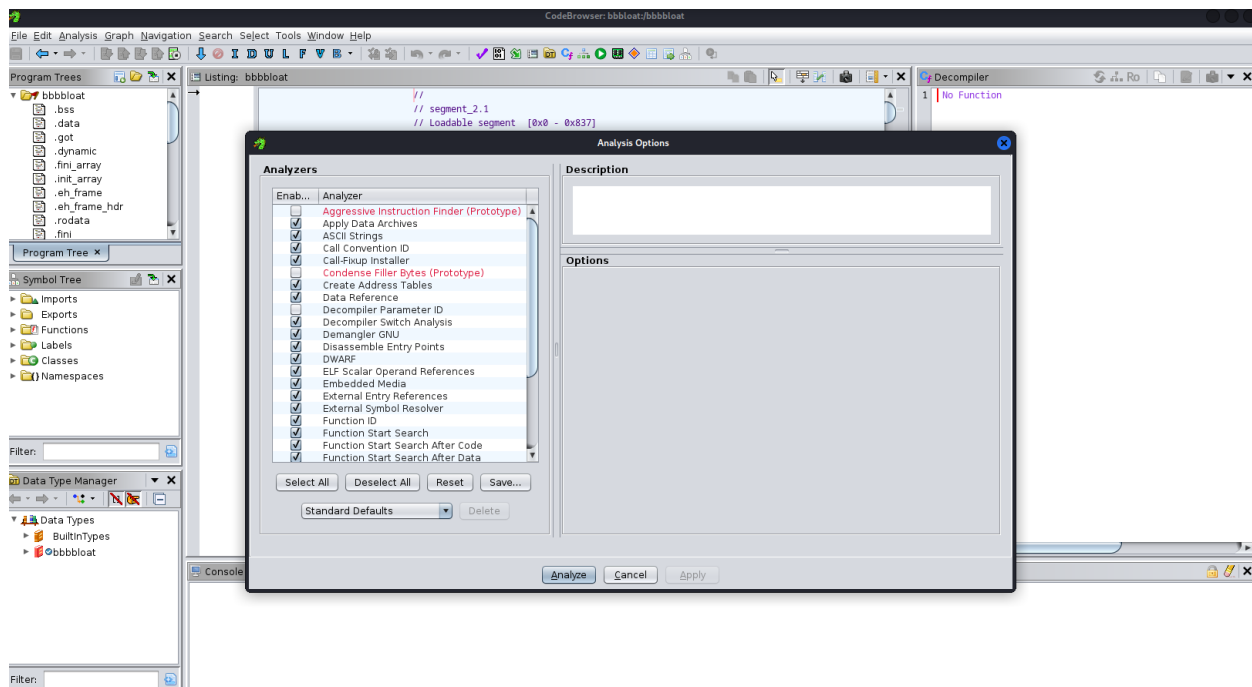
Let's run, Ghidra

```
./ghidraRun
```

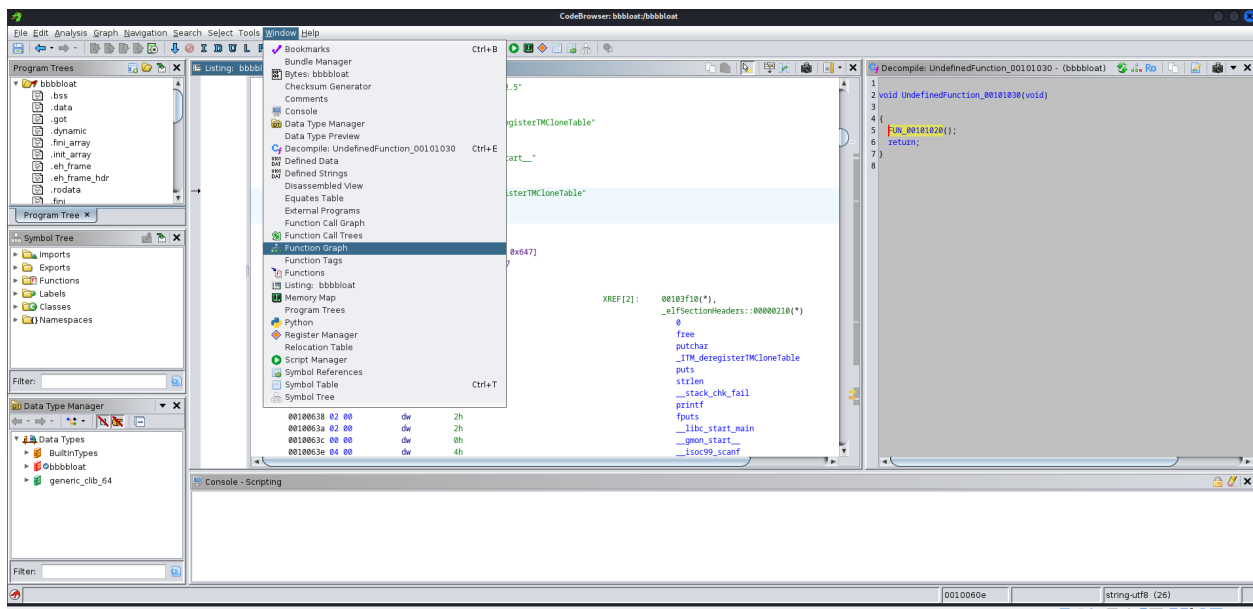


Let's now import our binary into Ghidra using the import file in the files tab



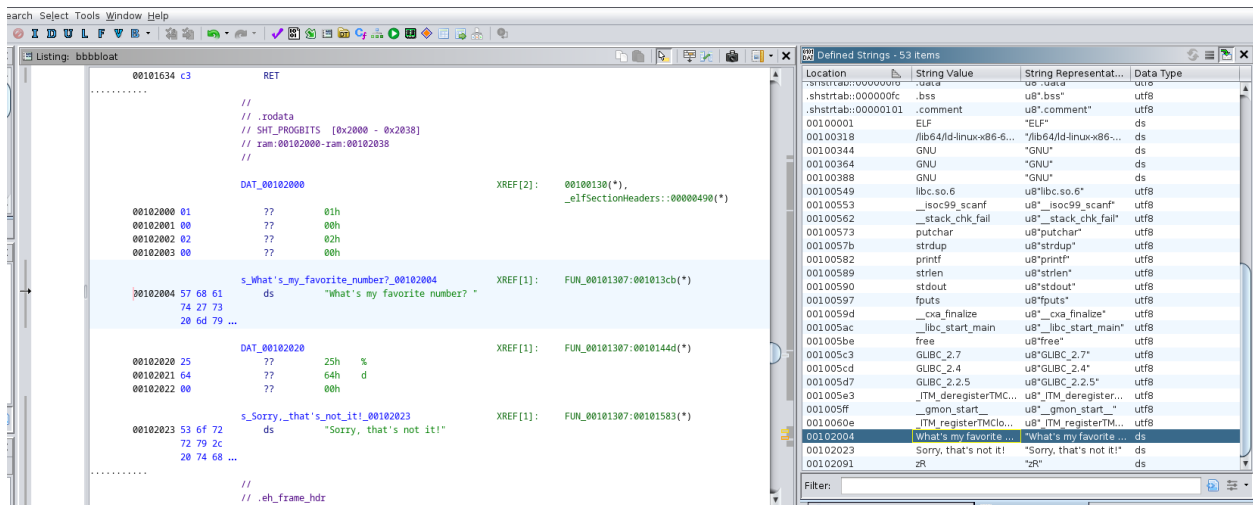


The next step is to find the "What is your favorite number" string.

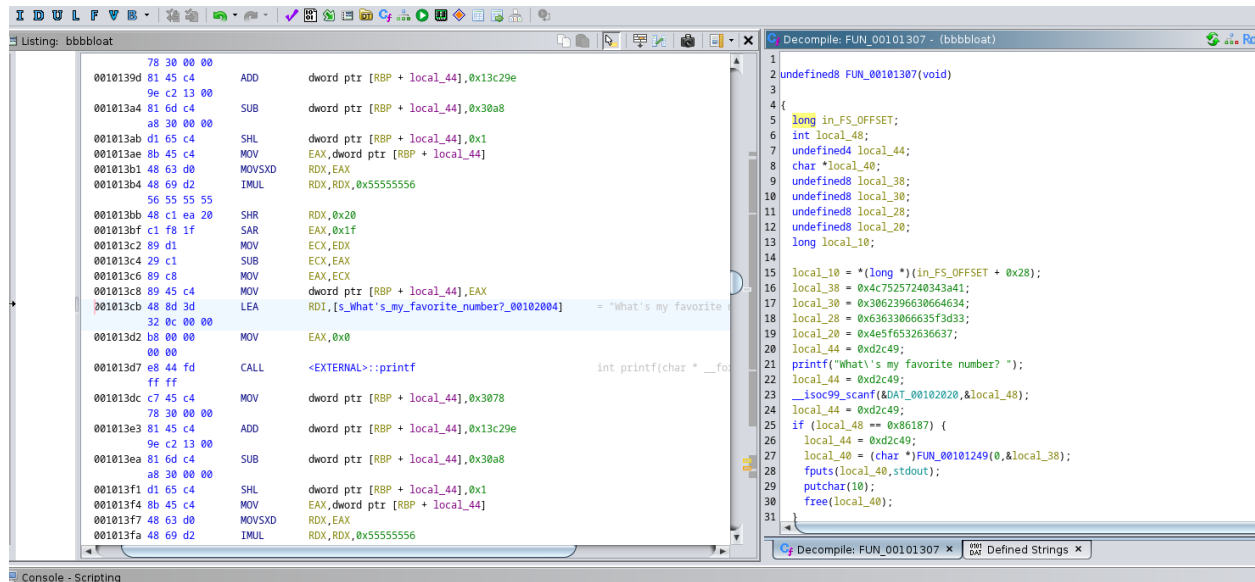


Click on Window > Defined strings

Navigate the Defined Strings menu and find the What's your favorite number string



It can be seen that there is a function present in the string. When I double-click the function and open the Decompile tab, some interesting observation is made.



It is observed that there's an if loop and the input is being validated by the local48 value which is 0x86187

```
printf("What's my favorite number? ");
local_44 = 0xd2c49;
__isoc99_scanf(&DAT_00102020,&local_48);
local_44 = 0xd2c49;
if (local_48 == 0x86187) {
    local_44 = 0xd2c49;
    local_40 = (char *)FUN_00101249(0,&local_38);
    fputs(local_40,stdout);
    putchar(10);
}
```

It's a hexadecimal number. Let's try to convert the hexadecimal number to a number

We can use python3 and paste the hex value and it will convert the number and give us the number.

```
(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ python3
Python 3.11.4 (main, Jun 17 2023, 10:13:09) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 0x86187
549255
>>> █
```

This might be the favorite number. Let's run our binary and input this number when prompted.

```
(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ ls
bbbloat

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ ./bbbloat
What's my favorite number? 21
Sorry, that's not it!

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ ./bbbloat
What's my favorite number? 549255
picoCTF{cu7_7h3_bl047_44f74a60}

(chris@kali)-[~/pico/reverse_engineering/bbbloat]
$ █
```

And we get the flag which is: picoCTF{cu7_7h3_bl047_44f74a60}