medium.com

picoCTF: Bbbbloat - S. H. - Medium

S. H.

1–2 minutes

The Challenge

Bbbbloat | 30(

Tags: picoCTF 2022 | Reverse Engineering | binary | obfuscation

AUTHOR: LT 'SYREAL' JONES

Description

Can you get the flag?

Reverse engineer this binary.

Hints

(None)

Checking out the file

We've got an executable that asks us to guess its favorite number

```
kali@kali:~/Documents/pico$ file bbbbloat
bbbbloat: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=99c5c1ce06be2403
22c15bcabc3cd90318eb2003, for GNU/Linux 3.2.0, stripped
kali@kali:~/Documents/pico$ ./bbbbloat
What's my favorite number? 42
Sorry, that's not it!
```

`strings` doesn't yield any plaintext of the flag or any easily decoded version of it

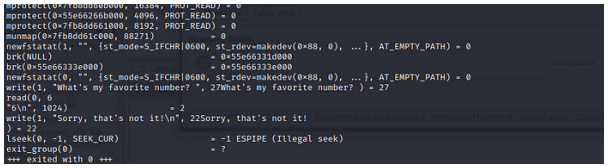
```
kali@kali:~/Documents/pico$ strings bbbbloat
/lib64/ld-linux-x86-64.so.2
libc.so.6
__isoc99_scanf
__stack_chk_fail
putchar
strdup
printf
strlen
stdout
fputs
__cxa_finalize
__libc_start_main
free
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u+UH
< ~XH
A:4@r%uLH
4Ff0f9b0H
3=_cf0beH
55b^e2N
VUUUH
VUUUH
VUUUH
VUUUH
VUUUH
VUUUH
```

`ltrace` doesn't show us any extra info

```
kali@kali:~/Documents/pico$ ltrace ./bbbbloat
What's my favorite number? 6
Sorry, that's not it!
+++ exited (status 0) +++
```

`strace` don't yield anything that looks immediately useful

```
kali@kali:~/Documents/pico$ strace ./bbbbloat
execve("./bbbbloat", ["./bbbbloat"], 0x7ffc0173d590 /* 54 vars */) = 0
brk(NULL)                               = 0x55e66331d000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=88271, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 88271, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fb8dd61c000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 9\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\200\0\300\4\0\0\0\1\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\27\n\3\204\346\353\257Q\2\31\305\316\347\315F\250"..., 68, 880) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1900960, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fb8dd61a000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 1934632, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fb8dd441000
mmap(0x7fb8dd463000, 1409024, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7fb8dd463000
mmap(0x7fb8dd5b0000, 327680, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17a000) = 0x7fb8dd5b0000
mmap(0x7fb8dd60b000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c9000) = 0x7fb8dd60b000
mmap(0x7fb8dd611000, 34088, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fb8dd611000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fb8dd43f000
arch_prctl(ARCH_SET_FS, 0x7fb8dd61b5c0) = 0
```



There are some long strands of bytes but they don't contain the flag.

Analysis with Ghidra

So let's take a look in Ghidra. After we open the file in Ghidra and analyze it, we can look at the decompilation of the entry point.

```
void entry(undefined8 param_1,undefined8 param_2,undefined8 param_3)

{
  undefined8 in_stack_00000000;
  undefined auStack8 [8];

  __libc_start_main(FUN_00101307,in_stack_00000000,&stack0x00000008,FUN_001015b0,FUN_00101620,
                    param_3,auStack8);
  do {
                    /* WARNING: Do nothing block with infinite loop */
  } while( true );
}
```

It looks like we're using some functions here, so let's start with investigating the one Ghidra labels as `FUN_00101307`

```
printf("What\'s my favorite number? ");
__isoc99_scanf();
if (local_48 == 0x86187) {
  __s = (char *)FUN_00101249(0,&local_38);
  fputs(__s,stdout);
  putchar(10);
  free(__s);
}
else {
  puts("Sorry, that\'s not it!");
}
```

We can see the if/else that checks if we have the right favorite number. So let's see what `local_48` is compared to. Hovering over the hex, we see `0x86187` is `549255` in decimal.

```
= 0x86187) {
*)FUN_00101249(0,&local_38);
dout);

 that\'s n

= *(long *)(in_FS_OFFSET + 0x28)) {
    /* WARNING: Subroutine does not return */
fail();
```

	Hex	Decimal
dword	86187h	549255
sdword	86187h	549255
wchar16[]		u"\b懇"

This also looks like the right place to find the favorite number we need because the function it jumps into in the if block, `FUN_00101249`, does a lot of string manipulation:

Decompile: FUN_00101249 - (bbbbloat)

```
char * FUN_00101249(undefined8 param_1,char *param_2)

{
  char cVar1;
  char *__s;
  size_t sVar2;
  ulong local_20;

  __s = strdup(param_2);
  sVar2 = strlen(__s);
  for (local_20 = 0; local_20 < sVar2; local_20 = local_20 + 1) {
    if ((' ' < __s[local_20]) && (__s[local_20] != '\x7f')) {
      cVar1 = (char)(__s[local_20] + 0x2f);
      if (__s[local_20] + 0x2f < 0x7f) {
        __s[local_20] = cVar1;
      }
      else {
        __s[local_20] = cVar1 + -0x5e;
      }
    }
  }
  return __s;
}
```

Running the program with `549255` we get the flag

```
kali@kali:~/Documents/pico$ ./bbbbloat
What's my favorite number? 549255
picoCTF{cu7_7h3_bl047_36dd316a}
```