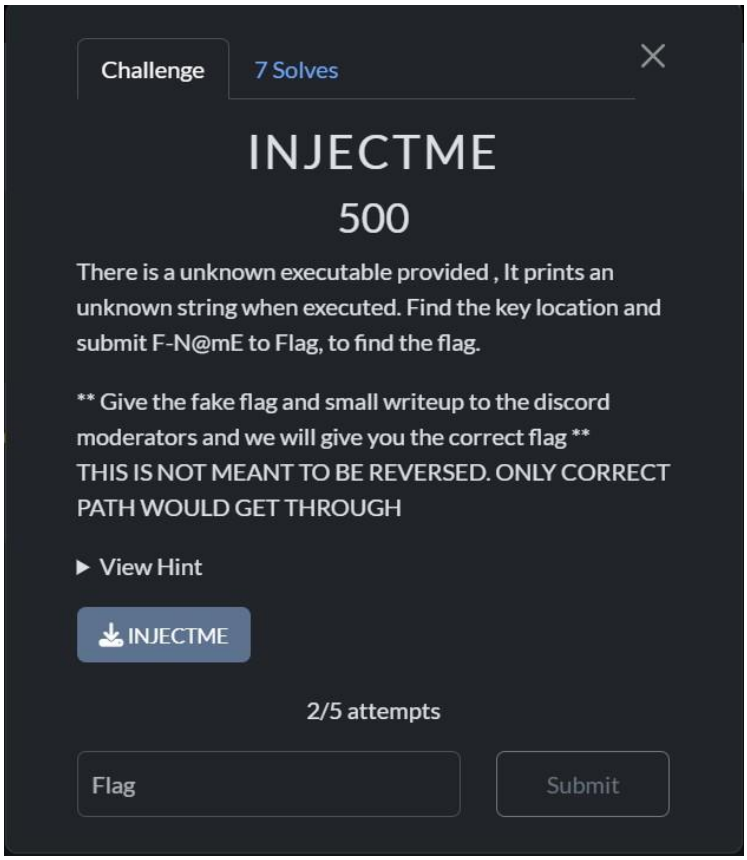
Challenge Name: INJECTME (500 Points) (PWN)



INJECTME

500

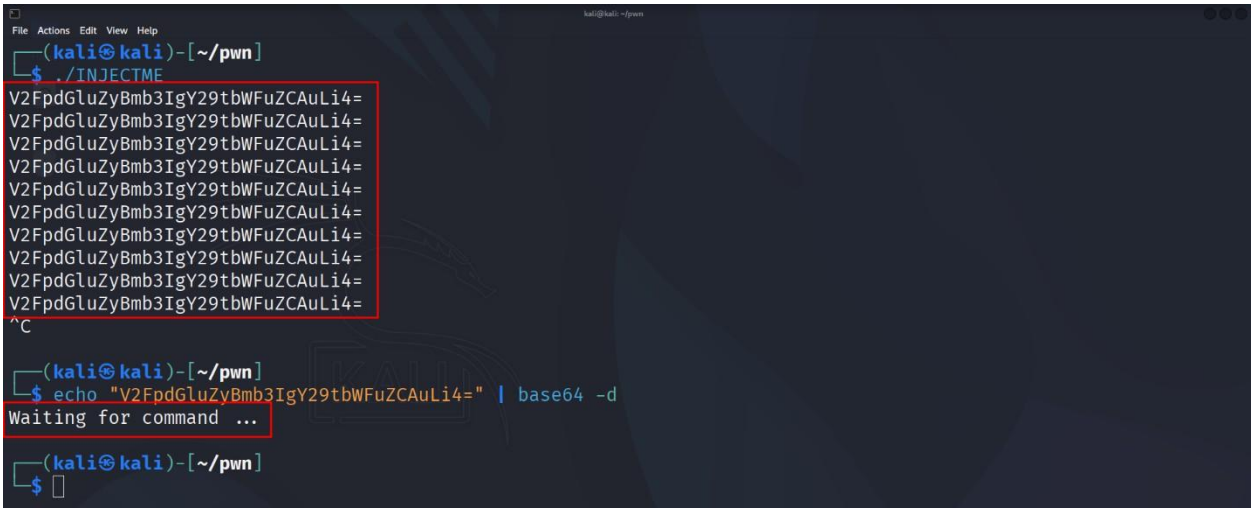
There is a unknown executable provided , It prints an unknown string when executed. Find the key location and submit F-N@mE to Flag, to find the flag.

** Give the fake flag and small writeup to the discord moderators and we will give you the correct flag **
THIS IS NOT MEANT TO BE REVERSED. ONLY CORRECT PATH WOULD GET THROUGH

2/5 attempts

Now this is a PWN challenge.

I start by executing the binary, where I got a base64 string, which gave me this output after decoding it



```
┌──(kali㉿kali)-[~/pwn]
└─$ ./INJECTME
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=
^C

┌──(kali㉿kali)-[~/pwn]
└─$ echo "V2FpdGluZyBmb3IgY29tbWFuZCAuLi4=" | base64 -d
Waiting for command ...

┌──(kali㉿kali)-[~/pwn]
└─$ 
```

Then I ran the **file** command.

```
┌──(kali㉿kali)-[~/pwn]
└─$ file INJECTME
INJECTME: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), BuildID[sha1]=fb77b0efbdd2746b1817803d1695cde29ce67953, for GNU/Linux 3.2.0, statically linked, no section header

┌──(kali㉿kali)-[~/pwn]
└─$
```

After seeing this, I run the **strings** command, which gave me an interesting output

```
┌──(kali㉿kali)-[~/pwn]
└─$ strings INJECTME | tail
com@v,
2X$/
1?||
2dCE?
69d?
6y("
@@]d
K?xx
UPX!
UPX!

┌──(kali㉿kali)-[~/pwn]
└─$
```

It was packed by **UPX packer**, so I unpacked it using UPX Packer

```
┌──(kali㉿kali)-[~/pwn]
└─$ upx -d INJECTME -o injectme
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.4       Markus Oberhumer, Laszlo Molnar & John Reiser    May 9th 2024

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     24167 <-      6468   26.76%   linux/amd64   injectme

Unpacked 1 file.

┌──(kali㉿kali)-[~/pwn]
└─$
```

After executing the file command again, I came to know that it was a stripped binary.

```
┌──(kali㉿kali)-[~/pwn]
└─$ file injectme
injectme: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-
x86-64.so.2, BuildID[sha1]=fb77b0efbdd2746b1817803d1695cde29ce67953, for GNU/Linux 3.2.0, stripped

┌──(kali㉿kali)-[~/pwn]
└─$
```

Because it is a stripped binary, we cannot see the main functions in the gdb debugger

So, we apply the breakpoint at the entry point of the binary

```
┌──(kali㉿kali)-[~/pwn]
└─$ pwndbg ./injectme
Reading symbols from ./injectme...
(No debugging symbols found in ./injectme)
pwndbg: loaded 190 pwndbg commands. Type pwndbg [filter] for a list.
pwndbg: created 13 GDB functions (can be used with print/break). Type help function to see them.
──────── tip of the day (disable with set show-tips off) ────────
Pwndbg context displays where the program branches to thanks to emulating few instructions into the future. You
 can disable this with set emulate off which may also speed up debugging
pwndbg> info files
Symbols from "/home/kali/pwn/injectme".
Local exec file:
        `/home/kali/pwn/injectme', file type elf64-x86-64.
        Entry point: 0x4010b0
        0x0000000000400318 - 0x0000000000400358 is .note.gnu.property
        0x0000000000400358 - 0x000000000040037c is .note.gnu.build-id
        0x000000000040037c - 0x000000000040039c is .note.ABI-tag
        0x0000000000401000 - 0x000000000040101b is .init
        0x0000000000401020 - 0x00000000004010b0 is .plt
        0x00000000004010b0 - 0x0000000000401295 is .text
        0x0000000000401298 - 0x00000000004012a5 is .fini
        0x0000000000402000 - 0x000000000040201c is .interp
        0x0000000000402020 - 0x0000000000402044 is .gnu.hash
        0x0000000000402048 - 0x0000000000402168 is .dynsym
        0x0000000000402168 - 0x00000000004021e0 is .dynstr
```

After the executing the binary, I stopped at the breakpoint

```
 ► 0x4010b0    endbr64
   0x4010b4    xor    ebp, ebp                    EBP ⇒ 0
   0x4010b6    mov    r9, rdx                     R9 ⇒ 0x7ffff7fcbc20 ◂— push rbp
   0x4010b9    pop    rsi                         RSI ⇒ 1
   0x4010ba    mov    rdx, rsp                    RDX ⇒ 0x7fffffffdd78 —▸ 0x7fffffffe110 ◂— '/home/kali/pwn/in
jectme'
   0x4010bd    and    rsp, 0xfffffffffffffff0     RSP ⇒ 0x7fffffffdd70 (0x7fffffffdd78 & -0x10)
   0x4010c1    push   rax
   0x4010c2    push   rsp
   0x4010c3    xor    r8d, r8d                    R8D ⇒ 0
   0x4010c6    xor    ecx, ecx                    ECX ⇒ 0
   0x4010c8    mov    rdi, 0x40122c               RDI ⇒ 0x40122c ◂— push rbp
──────────────────────────────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────────────────────────────────
00:0000│ r13 rsp 0x7fffffffdd70 ◂— 1
01:0008│         0x7fffffffdd78 —▸ 0x7fffffffe110 ◂— '/home/kali/pwn/injectme'
02:0010│         0x7fffffffdd80 ◂— 0
03:0018│ rcx     0x7fffffffdd88 —▸ 0x7fffffffe128 ◂— 0x5245545f5353454c ('LESS_TER')
04:0020│         0x7fffffffdd90 —▸ 0x7fffffffe13d ◂— 'POWERSHELL_TELEMETRY_OPTOUT=1'
05:0028│         0x7fffffffdd98 —▸ 0x7fffffffe15b ◂— 'LANGUAGE='
06:0030│         0x7fffffffdda0 —▸ 0x7fffffffe165 ◂— 'USER=kali'
07:0038│         0x7fffffffdda8 —▸ 0x7fffffffe16f ◂— 0x5245545f5353454c ('LESS_TER')
──────────────────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────────────────────────────────
 ► 0         0x4010b0 None
   1               0x1 None
   2   0x7fffffffe110 None
   3               0x0 None
──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> b *0x40122c
Breakpoint 2 at 0x40122c
pwndbg>
```

The first highlighted text in the image shows the main function address, So I added it on to breakpoint

In the main function, I saw a memory address being called, so I jumped right onto that function and started analyzing it

```
   0x401270    mov    rax, qword ptr [rip + 0x2df9]     RAX, [0x404070] ⇒ 0x402340 ◂— 'V2FpdGluZyBmb3IgY29tbWFuZCAuLi4='
   0x401277    mov    rdi, rax                          RDI ⇒ 0x402340 ◂— 'V2FpdGluZyBmb3IgY29tbWFuZCAuLi4='
   0x40127a    call   puts@plt                     <puts@plt>

 ► 0x40127f    mov    eax, 0                            EAX ⇒ 0
   0x401284    call   0x401196                     <0x401196>

   0x401289    mov    edi, 1                            EDI ⇒ 1
   0x40128e    call   sleep@plt                    <sleep@plt>

   0x401293    jmp    0x401270                     <0x401270>
    ↓
   0x401270    mov    rax, qword ptr [rip + 0x2df9]     RAX, [0x404070]
──────────────────────────────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────────────────────────────────
00:0000│ rbp rsp 0x7fffffffdc60 ◂— 1
01:0008│ +008    0x7fffffffdc68 —▸ 0x7ffff7dd9ca8 ◂— mov edi, eax
02:0010│ +010    0x7fffffffdc70 —▸ 0x7fffffffdd60 —▸ 0x7fffffffdd68 ◂— 0x38 /* '8' */
03:0018│ +018    0x7fffffffdc78 —▸ 0x40122c ◂— push rbp
04:0020│ +020    0x7fffffffdc80 ◂— 0x100400040 /* '@' */
05:0028│ +028    0x7fffffffdc88 —▸ 0x7fffffffdd78 —▸ 0x7fffffffe110 ◂— '/home/kali/pwn/injectme'
06:0030│ +030    0x7fffffffdc90 —▸ 0x7fffffffdd78 —▸ 0x7fffffffe110 ◂— '/home/kali/pwn/injectme'
07:0038│ +038    0x7fffffffdc98 ◂— 0xcbed7bcb12574dbc
──────────────────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────────────────────────────────
 ► 0         0x40127f None
   1   0x7ffff7dd9ca8 None
   2   0x7ffff7dd9d65 __libc_start_main+133
   3         0x4010d5 None
──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg>
```

```
   0x40119a    sub    rsp, 0x10                          RSP => 0x7fffffffdc40 (0x7fffffffdc50 - 0x10)
   0x40119e    mov    rax, qword ptr [rip + 0x2ee3]      RAX, [0x404088] => 0x4052a0 <- 0x47414c46 /* 'FLAG' */
   0x4011a5    test   rax, rax                           0x4052a0 & 0x4052a0     EFLAGS => 0x10206 [ cf PF af zf sf IF df of ac
 ]
   0x4011a8  x je     0x401229                     <0x401229>

 ▶ 0x4011aa    mov    rax, qword ptr [rip + 0x2ed7]      RAX, [0x404088] => 0x4052a0 <- 0x47414c46 /* 'FLAG' */
   0x4011b1    mov    esi, 0x402361                      ESI => 0x402361 <- 'INJECTME'
   0x4011b6    mov    rdi, rax                           RDI => 0x4052a0 <- 0x47414c46 /* 'FLAG' */
   0x4011b9    call   strcmp@plt                   <strcmp@plt>

   0x4011be    test   eax, eax
   0x4011c0    jne    0x401229                     <0x401229>
──────────────────────────────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────────────────────────────────
00:0000│  rsp 0x7fffffffdc40 <- 0
01:0008│ -008 0x7fffffffdc48 -> 0x7fffffffdd88 -> 0x7fffffffe128 <- 0x5245545f5353454c ('LESS_TER')
02:0010│  rbp 0x7fffffffdc50 -> 0x7fffffffdc60 <- 1
03:0018│ +008 0x7fffffffdc58 -> 0x401289 <- mov edi, 1
04:0020│ +010 0x7fffffffdc60 <- 1
05:0028│ +018 0x7fffffffdc68 -> 0x7ffff7dd9ca8 <- mov edi, eax
06:0030│ +020 0x7fffffffdc70 -> 0x7fffffffdd60 -> 0x7fffffffdd68 <- 0x38 /* '8' */
07:0038│ +028 0x7fffffffdc78 -> 0x40122c <- push rbp
──────────────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────────────────────────────
 ▶ 0         0x4011aa None
   1         0x401289 None
   2   0x7ffff7dd9ca8 None
   3   0x7ffff7dd9d65 __libc_start_main+133
   4         0x4010d5 None
──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg>
```

Here I saw a strcmp instruction which was being done on the **INJECTME** and **FLAG**

So, I changed the value of the rax register to the memory address of **INJECTME** string

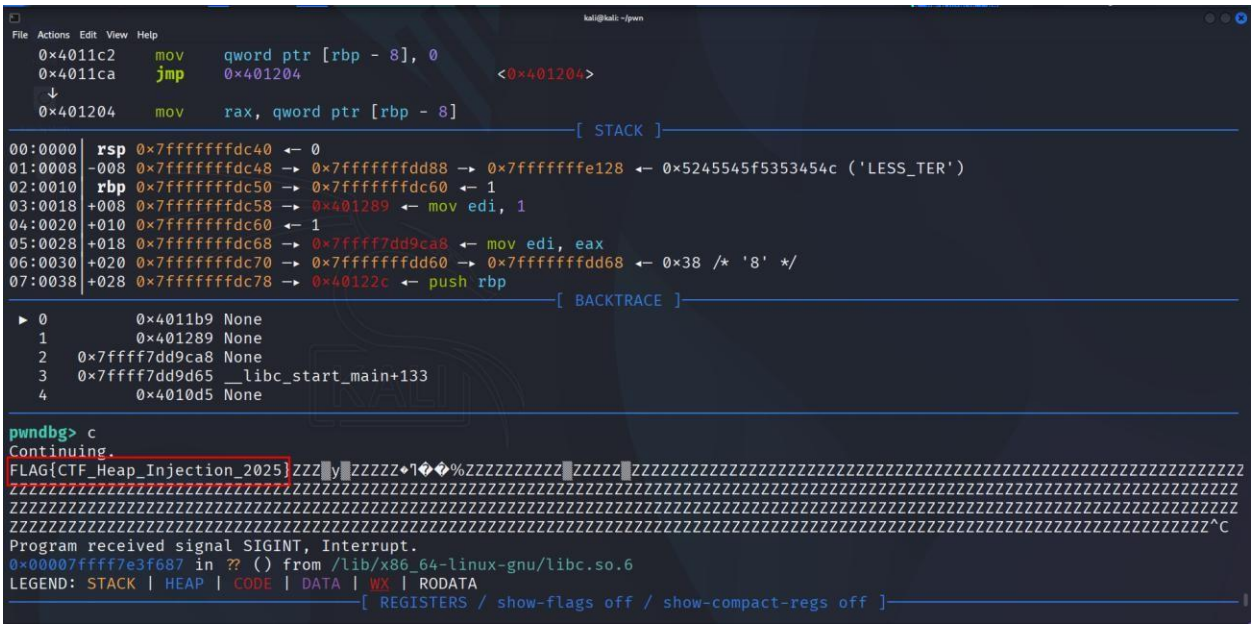
```
File  Actions  Edit  View  Help
   0x4011a8  x je     0x401229                     <0x401229>

   0x4011aa    mov    rax, qword ptr [rip + 0x2ed7]      RAX, [0x404088] => 0x4052a0 <- 0x47414c46 /* 'FLAG' */
   0x4011b1    mov    esi, 0x402361                      ESI => 0x402361 <- 'INJECTME'
 ▶ 0x4011b6    mov    rdi, rax                           RDI => 0x4052a0 <- 0x47414c46 /* 'FLAG' */
   0x4011b9    call   strcmp@plt                   <strcmp@plt>

   0x4011be    test   eax, eax
   0x4011c0    jne    0x401229                     <0x401229>

   0x4011c2    mov    qword ptr [rbp - 8], 0
   0x4011ca    jmp    0x401204                     <0x401204>
──────────────────────────────────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────────────────────────────────
00:0000│  rsp 0x7fffffffdc40 <- 0
01:0008│ -008 0x7fffffffdc48 -> 0x7fffffffdd88 -> 0x7fffffffe128 <- 0x5245545f5353454c ('LESS_TER')
02:0010│  rbp 0x7fffffffdc50 -> 0x7fffffffdc60 <- 1
03:0018│ +008 0x7fffffffdc58 -> 0x401289 <- mov edi, 1
04:0020│ +010 0x7fffffffdc60 <- 1
05:0028│ +018 0x7fffffffdc68 -> 0x7ffff7dd9ca8 <- mov edi, eax
06:0030│ +020 0x7fffffffdc70 -> 0x7fffffffdd60 -> 0x7fffffffdd68 <- 0x38 /* '8' */
07:0038│ +028 0x7fffffffdc78 -> 0x40122c <- push rbp
──────────────────────────────────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────────────────────────────────
 ▶ 0         0x4011b6 None
   1         0x401289 None
   2   0x7ffff7dd9ca8 None
   3   0x7ffff7dd9d65 __libc_start_main+133
   4         0x4010d5 None
──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> set $rax=0x402361
pwndbg>
```

After this I simply pressed '**c**' which executed the further instructions and gave me the flag.



And here we got the flag.

FLAG{CTF_Heap_Injection_2025}

According to the challenge description, participants are required to create a small writeup with the flag and submit it to a CTF Discord moderator. Upon verification and a subsequent questioning phase, the moderator will release the true flag, which is:

FLAG{CTF_Heap_Memory_Injection_2025}

~ By Team justahacker