pwn_myself

使用ghidra反編譯，首先看到main function:

```
1
2  undefined8 main(void)
3
4  {
5    __uid_t uid;
6    undefined8 exit_code;
7    long in_FS_OFFSET;
8    undefined buf [24];
9    long canary;
10
11   canary = *(long *)(in_FS_OFFSET + 0x28);
12   uid = geteuid();
13   if (uid == 0) {
14     pwn_myself(buf);
15     exit_code = 0;
16   }
17   else {
18     exit_code = 0xffffffff;
19   }
20   if (canary != *(long *)(in_FS_OFFSET + 0x28)) {
21                     // WARNING: Subroutine does not return
22     __stack_chk_fail();
23   }
24   return exit_code;
25 }
26
```

首先分析一下他做了什麼事情:
在第12行的地方呼叫了getuid()，也就是取得使用者id，並且做判斷，如果不等於0則直接離開，因此要讓程式
= 0)

繼續往下看，進入(uid == 0)成立時會執行的function:

```
2  void pwn_myself(char *buf_0x18)
3
4  {
5    char *payload;
6    char *arr;
7    int i;
8    int j;
9    undefined8 ptr;
10   long ptr2;
11
12   ptr = *(undefined8 *)(buf_0x18 + 0x18);
13   ptr2 = *(long *)(buf_0x18 + -8);
14   payload = (char *)malloc(0x38);
15   for (i = 0; i < 0x18; i = i + 1) {
16     payload[i] = 'A';
17   }
18   *(undefined8 *)(payload + 0x18) = ptr;
19   *(undefined8 *)(payload + 0x20) = 0;
20   *(long *)(payload + 0x28) = ptr2 + 0x1a;
21                       // overwrite stack return address
22   *(long *)(payload + 0x30) = ptr2 + -0x292;
23   arr = buf_0x18;
24   for (j = 0; j < 0x38; j = j + 1) {
25     *arr = payload[j];
26     arr = arr + 1;
27   }
28   return;
29 }
30
```

這邊是在做pwn，藉由複寫stack上的return address，來跳躍到指定的地方去。
這裡是原本的return address (0x166b51)

```
00166b49 48 89 c7      MOV       RDI,exit_code
00166b4c e8 e6 fe      CALL      pwn_myself
         ff ff
00166b51 b8 00 00      MOV       exit_code,0x0
         00 00

                     LAB_00166b56
```

然後他把這個位置改成 0x166b51 - 0x292 = 0x1668bf
0x1668bf是一個function:

```
void jump_here(void)

{
  ssize_t len;
  long in_FS_OFFSET;
  uint cmd;
  uint local_34;
  int local_30;
  int fd;
  undefined buf [16];
  short local_18;
  ushort local_16;
  int local_14;
  long canary;

  canary = *(long *)(in_FS_OFFSET + 0x28);
  local_34 = 0;
  local_30 = 0;
                    // maybe: /dev/input/event? (keyboard)
  fd = open_device_input_unknown();
  if (fd != 0) {
    cmd = 0;
    ioctl(fd,0x80044519,&cmd);
    local_34 = cmd >> 1 & 1;
    while (len = read(fd,buf,24), 0 < len) {
      if (local_18 == 1) {
        if (local_14 == 1) {
          if (local_16 == 42) {
            local_30 = 2;
          }
        }
        else if (local_14 == 0) {
          if (local_16 == 42) {
            local_30 = 0;
          }
          else if (local_16 == 58) {
            local_34 = local_34 ^ 1;
          }
          else if (*(long *)((&PTR_DAT_0049b960)[(int)(local_34 + local_30)] + (ulong)local_16 * 8) != 0) {
            got_it();
          }
        }
      }
    }
    close(fd);
  }
  if (canary == *(long *)(in_FS_OFFSET + 0x28)) {
    return;
  }
                    // WARNING: Subroutine does not return
  __stack_chk_fail();
}
```

input

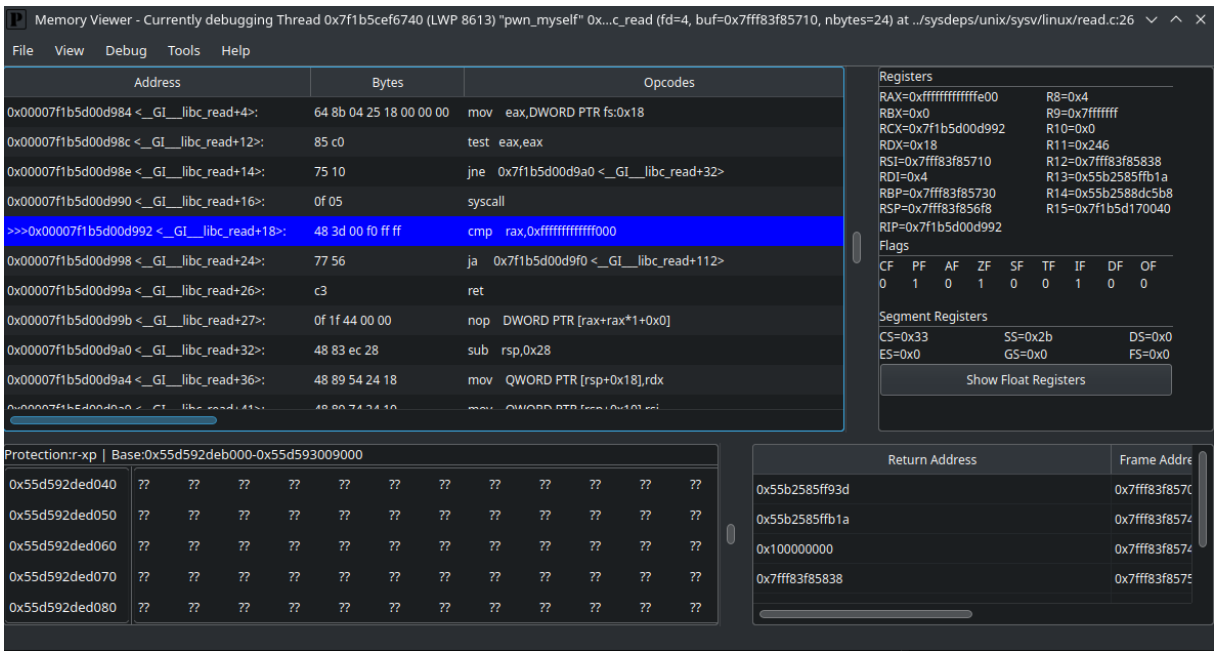
仔細往下追，先看open_device_input_unknown這個function:

```
int open_device_input_unknown(void)

{
  int iVar1;
  undefined8 input_dev_char;
  long in_FS_OFFSET;
  int fd;
  uint i;
  ulong local_178 [12];
  char device_path [264];
  long canary;

  canary = *(long *)(in_FS_OFFSET + 0x28);
  (*(code *)PTR_open_dir_copy_str_004970b8)(&dir,"/dev/input");
  do {
    while( true ) {
      do {
        iVar1 = (*(code *)PTR_find_device_or_event_004970c0)(&dir);
                    // no device(0x2) found
        if (iVar1 == 0) {
          fd = 0;
          goto LAB_001668a9;
        }
        input_dev_char = (*(code *)PTR_get_dir_info_004970c8)(&dir);
        memset(device_path,0,0x100);
        sprintf(device_path,"/dev/input/%s",input_dev_char);
        fd = open(device_path,0);
      } while (fd < 0);
      memset(local_178,0,0x60);
      ioctl(fd,0b10000000000111110100010100100000,local_178);
      if ((local_178[0] & 2) != 0) break;
      close(fd);
    }
    memset(local_178,0,0x60);
    ioctl(fd,0x82ff4521,local_178);
    for (i = 0; i < 32; i = i + 1) {
      if ((local_178[(ulong)(long)*(int *)(&DAT_004970e0 + (long)(int)i * 4) >> 6] >> ((byte)*(undefined4 *)(&DAT_004970e0 + (long)(int)i * 4) & 0x3f) & 1) == 0) {
        close(fd);
        fd = 0;
        break;
      }
    }
  } while (fd == 0);
LAB_001668a9:
  if (canary == *(long *)(in_FS_OFFSET + 0x28)) {
    return fd;
  }
                    // WARNING: Subroutine does not return
  __stack_chk_fail();
}
```

這邊會去開一個檔案，路徑是/dev/input/eventX，這個路徑在Linux上是用來表達各種輸入的檔案(linux上萬
我不太確定這邊到底開的是哪一個，但用動態分析看的時候應該是截到keyboard的那個event。

Dynamic analysis

這邊使用PINCE做動態分析，看看程式執行的樣子。
執行之後，他被read()這個function block住了，在等待輸入。



這邊不太確定他要什麼輸入什麼東西，先往後面的function看

Correct ?

往下追幾個function:

```
1
2  void got_it(char *param_1)
3
4  {
5    char *data;
6
7    data = param_1;
8    while (*data != '\0') {
9      (&heap_0049ddc0)[i] = *data;
10     i = i + 1;
11     data = data + 1;
12     if (i == 44) {
13       solved();
14       i = 0;
15     }
16   }
17   return;
18 }
19
```

```
1
2  void solved(void)
3
4  {
5    long in_FS_OFFSET;
6    undefined4 len;
7    uint i;
8    undefined4 length;
9    undefined buf [88];
10   long canary;
11
12   canary = *(long *)(in_FS_OFFSET + 0x28);
13   length = maybe_decrypt();
14   for (i = 0; (i < 48 && ((&DAT_0049de00)[(int)i] == (&DAT_00497040)[(int)i])); i = i + 1) {
15   }
16   if (i != 48) {
17     send_data_17209(&DAT_0049de00,length);
18     if (canary == *(long *)(in_FS_OFFSET + 0x28)) {
19       return;
20     }
21                     // WARNING: Subroutine does not return
22     __stack_chk_fail();
23   }
24   FUN_0016622d(&DAT_00497020,0x20,buf,&len);
25   send_data_17209(buf,len);
26                     // WARNING: Subroutine does not return
27   exit(1);
28 }
29
```

我認爲這邊應該是當吃到正確的input時，會來執行的地方，所以應該很接近flag了。

OpenSSL

繼續往下追:

```
int maybe_decrypt(void)

{
  int iVar1;
  undefined8 uVar2;
  long in_FS_OFFSET;
  int local_20;
  int local_1c;
  long local_18;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_18 = FUN_0016a860();
  if (local_18 == 0) {
                    // WARNING: Subroutine does not return
    exit(1);
  }
  uVar2 = FUN_0016a1c0();
  iVar1 = FUN_0016d8a0(local_18,uVar2,0,&DAT_00497070,&DAT_00497080);
  if (iVar1 != 1) {
                    // WARNING: Subroutine does not return
    exit(1);
  }
  iVar1 = FUN_0016abc0(local_18,&DAT_0049de00,&local_20,&heap_0049ddc0,0x2c);
  if (iVar1 != 1) {
                    // WARNING: Subroutine does not return
    exit(1);
  }
  local_1c = local_20;
  iVar1 = FUN_0016adc0(local_18,&DAT_0049de00 + local_20,&local_20);
  if (iVar1 != 1) {
                    // WARNING: Subroutine does not return
    exit(1);
  }
  local_1c = local_1c + local_20;
  FUN_0016cb40(local_18);
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    // WARNING: Subroutine does not return
    __stack_chk_fail();
  }
  return local_1c;
}

```

這邊我認爲是在做解密的動作，因爲這幾個function往下點可以看到很多openssl相關的東西，例如:

```
int FUN_0016adc0(long *param_1,undefined8 param_2,uint *param_3)

{
  uint uVar1;
  long lVar2;
  int iVar3;
  uint uVar4;
  undefined8 uVar5;
  long in_FS_OFFSET;
  ulong local_48;
  long local_40;

  local_40 = *(long *)(in_FS_OFFSET + 0x28);
  if (param_3 == (uint *)0x0) {
    FUN_0026fff0();
    iVar3 = 0;
    FUN_00270110("../crypto/evp/evp_enc.c",0x29a,"EVP_EncryptFinal_ex");
    FUN_002704c0(6,0xc0102,0);
    goto LAB_0016ae64;
  }
  *param_3 = 0;
  iVar3 = *(int *)(param_1 + 2);
  if (iVar3 == 0) {
    FUN_0026fff0();
    FUN_00270110("../crypto/evp/evp_enc.c",0x2a0,"EVP_EncryptFinal_ex");
    FUN_002704c0(6,0x94,0);
    goto LAB_0016ae64;
  }
  lVar2 = *param_1;
  if (lVar2 == 0) {
    FUN_0026fff0();
    iVar3 = 0;
    FUN_00270110("../crypto/evp/evp_enc.c",0x2a5,"EVP_EncryptFinal_ex");
    FUN_002704c0(6,0x83,0);
    goto LAB_0016ae64;
  }
  if (*(long *)(lVar2 + 0x78) == 0) {
    if ((*(byte *)(lVar2 + 0x12) & 0x10) == 0) {
      uVar4 = *(uint *)(lVar2 + 4);
      if (0x20 < uVar4) {
                    // WARNING: Subroutine does not return
        FUN_00178960("assertion failed: b <= sizeof(ctx->buf)","../crypto/evp/evp_enc.c",0x2cc);
      }
      iVar3 = 1;
      if (uVar4 != 1) {
        uVar1 = *(uint *)((long)param_1 + 0x14);
        if ((*(byte *)((long)param_1 + 0x71) & 1) == 0) {
          if (uVar1 < uVar4) {
            memset((void *)((long)param_1 + (ulong)uVar1 + 0x38),uVar4 - uVar1 & 0xff,(ulong)((uVar4 - uVar1) - 1) + 1);
          }
          iVar3 = (**(code **)(lVar2 + 0x28))(param_1,param_2,param_1 + 7,uVar4);
          if (iVar3 != 0) {
```

很多字串應該是在解密失敗或者是一些錯誤的時候會列印出的東西，像是exception之類的(我的猜測)。

Socket

先假設已經解密完成，回到剛剛的地方(solved這個function)，往下可以看到一些送資料的程式碼:

```
void send_data_17209(void *buf,int len)

{
  ssize_t sVar1;
  long in_FS_OFFSET;
  undefined4 optval;
  int sockfd;
  int err;
  sockaddr socket_address;
  long canary;

  canary = *(long *)(in_FS_OFFSET + 0x28);
  optval = 1;
  sockfd = socket(2,2,0);
  if (sockfd < 0) {
                    // WARNING: Subroutine does not return
    exit(1);
  }
  err = setsockopt(sockfd,1,6,&optval,4);
  if (err < 0) {
                    // WARNING: Subroutine does not return
    exit(1);
  }
  memset(&socket_address,0,0x10);
                    // AF_INET
  socket_address.sa_family = 2;
  socket_address.sa_data._2_4_ = htonl(0xffffffff);
  socket_address.sa_data._0_2_ = htons(17209);
  sVar1 = sendto(sockfd,buf,(long)len,0,&socket_address,0x10);
  err = (int)sVar1;
  if (canary != *(long *)(in_FS_OFFSET + 0x28)) {
                    // WARNING: Subroutine does not return
    __stack_chk_fail();
  }
  return;
}

```

這邊我猜就是已經把flag或者什麼資料解密完之後，用socket把資料送出去了，看起來是送到17209的port，然

試著寫一點程式碼，看能不能收到資料:

```
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("localhost", 17209))
conn.listen(100)

while True:
    try:
        client, _ = conn.accept()
        while True:
            data = client.recv(100)
            print(data)
    except Exception as err:
        print(err)
```

結果不論按什麼鍵，都沒能收到資料

What do you want?

即便看了很多個function，最終還是不知道read那邊他究竟想要什麼東西，也許是要按照一定的順序按鍵盤?(
雖然很多想法，但一些地方無法正確理解，解題失敗。