RpcCalc Writeup

—— SJTU Ph0t1n1a ——

Basic Functions

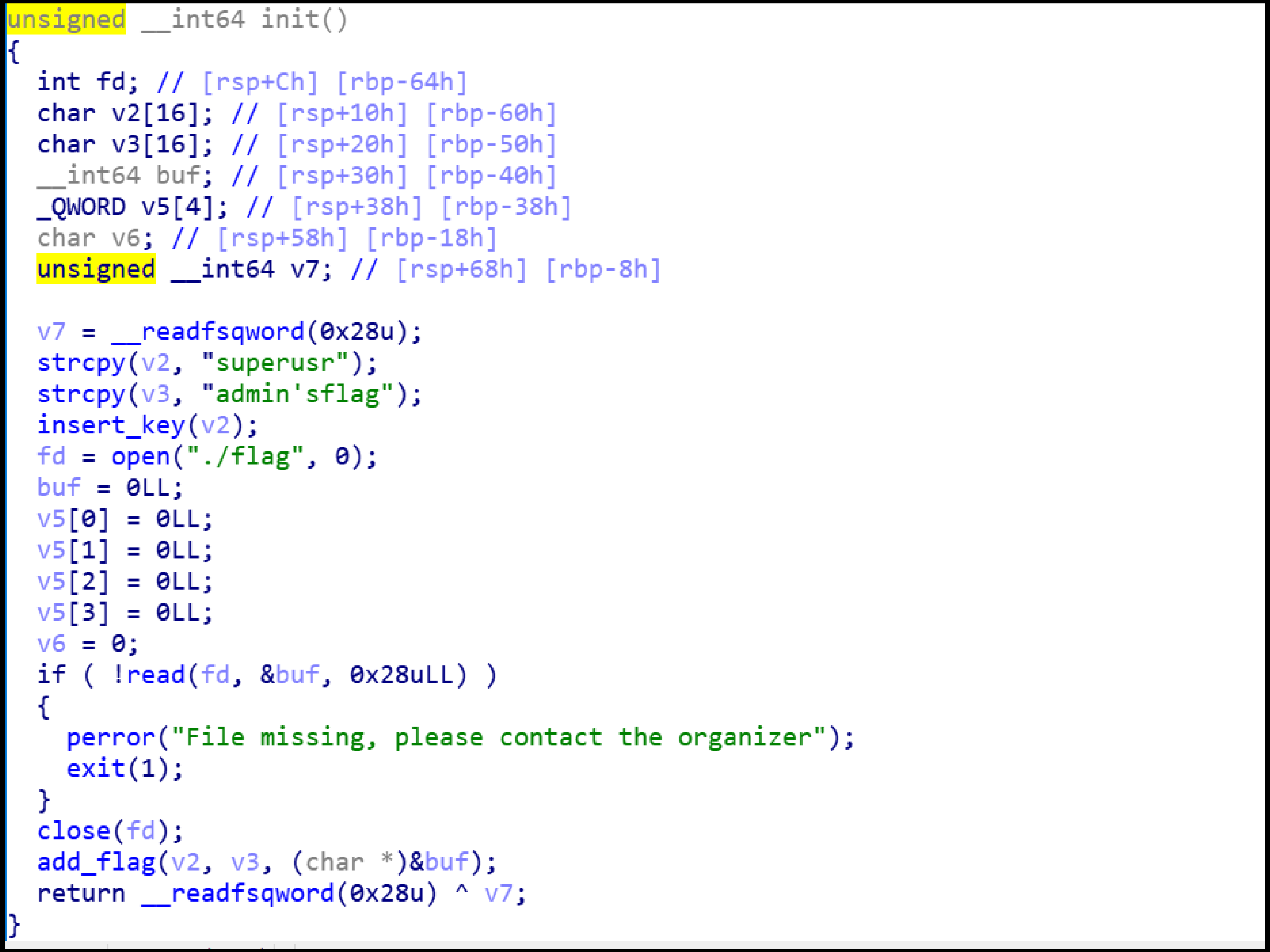
- 0-4： Mentioned in template

- 5： if_you_forget_your_expr_id()
  - It can achieve same function as get_result() (case 2)
  - However it can return the result of given user without providing the expr_id directly

- 6： status_extant_for_checker()
  - Used for checker to check whether the flag exists in program memory

Init()

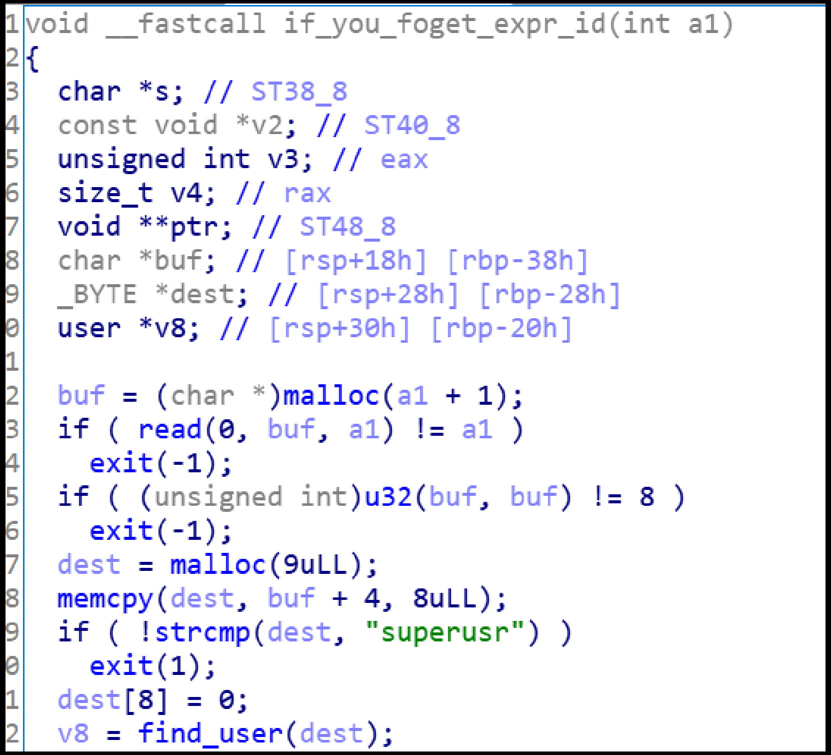
- Read the flag to the buffer
- New user “superusr”
- New corrid “admin’sflag”
- Corr result = flag’s address



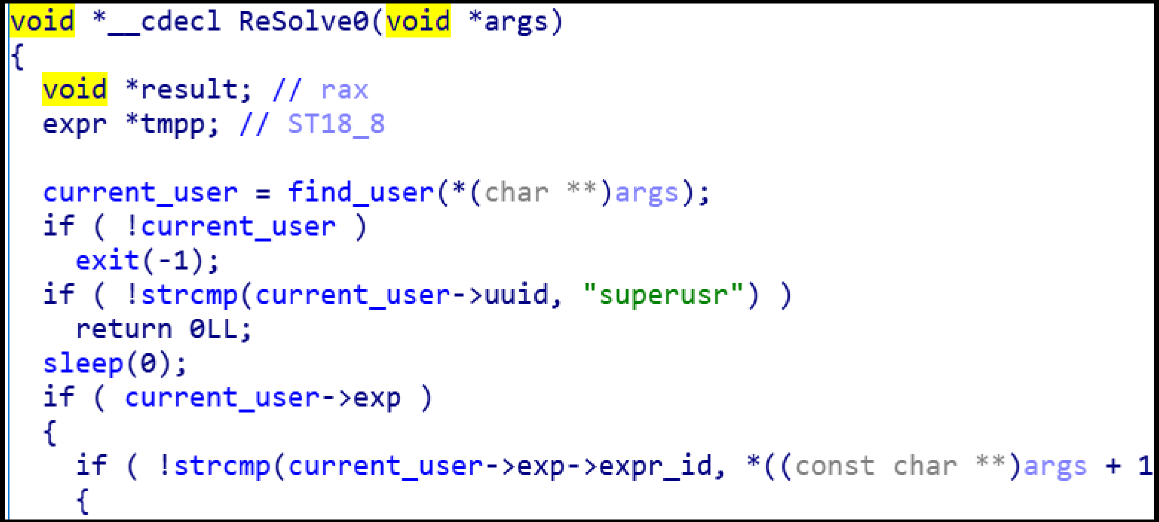
```
unsigned __int64 init()
{
  int fd; // [rsp+Ch] [rbp-64h]
  char v2[16]; // [rsp+10h] [rbp-60h]
  char v3[16]; // [rsp+20h] [rbp-50h]
  __int64 buf; // [rsp+30h] [rbp-40h]
  _QWORD v5[4]; // [rsp+38h] [rbp-38h]
  char v6; // [rsp+58h] [rbp-18h]
  unsigned __int64 v7; // [rsp+68h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  strcpy(v2, "superusr");
  strcpy(v3, "admin'sflag");
  insert_key(v2);
  fd = open("./flag", 0);
  buf = 0LL;
  v5[0] = 0LL;
  v5[1] = 0LL;
  v5[2] = 0LL;
  v5[3] = 0LL;
  v6 = 0;
  if ( !read(fd, &buf, 0x28uLL) )
  {
    perror("File missing, please contact the organizer");
    exit(1);
  }
  close(fd);
  add_flag(v2, v3, (char *)&buf);
  return __readfsqword(0x28u) ^ v7;
}
```

Problem is …

- However you cannot directly read the flag / flag’s address



```
void __fastcall if_you_foget_expr_id(int a1)
{
  char *s; // ST38_8
  const void *v2; // ST40_8
  unsigned int v3; // eax
  size_t v4; // rax
  void **ptr; // ST48_8
  char *buf; // [rsp+18h] [rbp-38h]
  _BYTE *dest; // [rsp+28h] [rbp-28h]
  user *v8; // [rsp+30h] [rbp-20h]

  buf = (char *)malloc(a1 + 1);
  if ( read(0, buf, a1) != a1 )
    exit(-1);
  if ( (unsigned int)u32(buf, buf) != 8 )
    exit(-1);
  dest = malloc(9uLL);
  memcpy(dest, buf + 4, 8uLL);
  if ( !strcmp(dest, "superusr") )
    exit(1);
  dest[8] = 0;
  v8 = find_user(dest);
```



```
void *__cdecl ReSolve0(void *args)
{
  void *result; // rax
  expr *tmpp; // ST18_8

  current_user = find_user(*(char **)args);
  if ( !current_user )
    exit(-1);
  if ( !strcmp(current_user->uuid, "superusr") )
    return 0LL;
  sleep(0);
  if ( current_user->exp )
  {
    if ( !strcmp(current_user->exp->expr_id, *((const char **)args + 1
    {
```

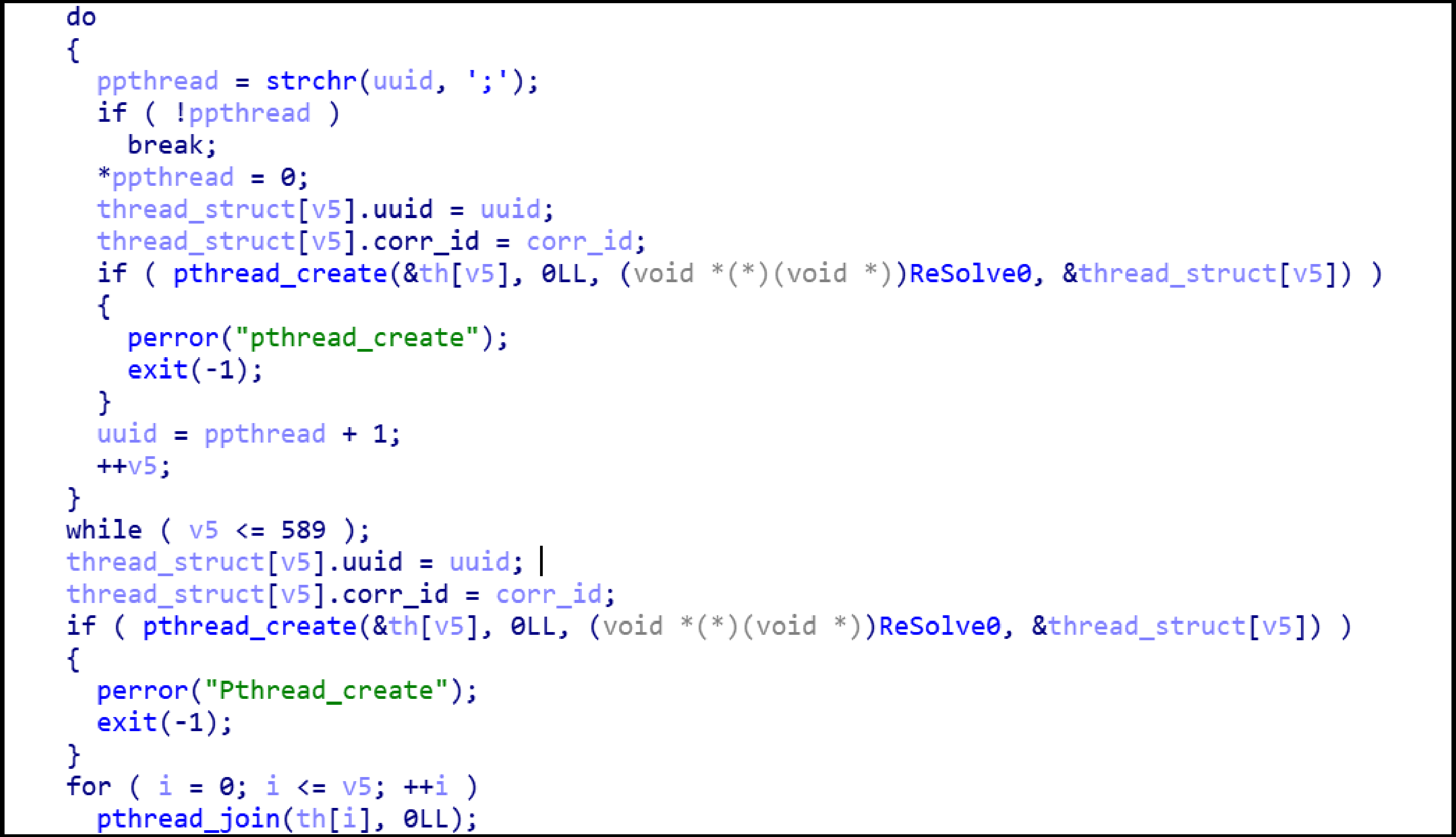
Vulnerabilities

- 1. Double Fetch
  - Leak the flag address

- 2. Uninitialized variables
  - Read 8 bytes from any given address

Double Fetch

- Get_result()
- uuid can be divided by ‘;’ to multiuser
- Server will create multi-subthread to fetch these results

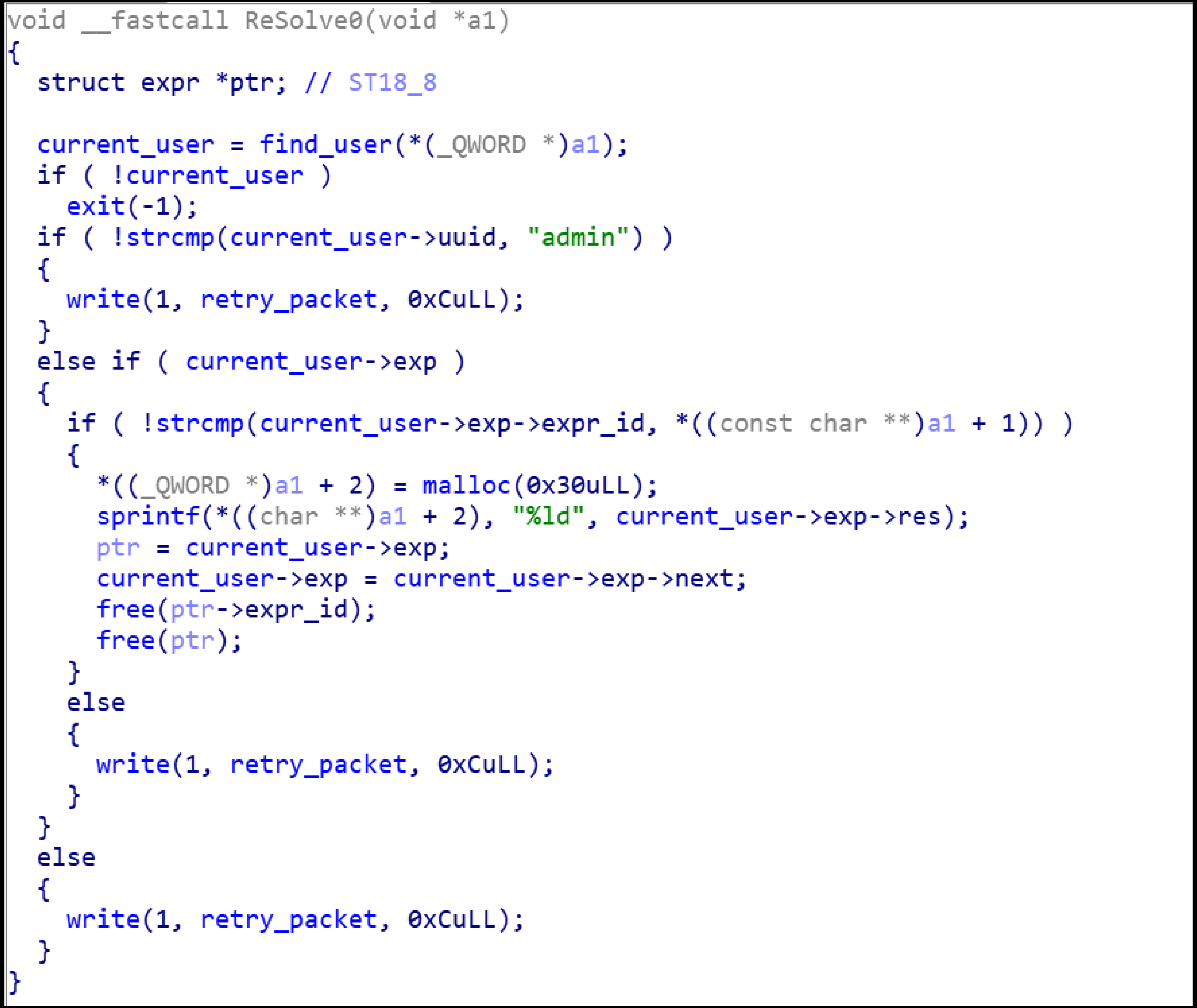


```
do
{
  ppthread = strchr(uuid, ';');
  if ( !ppthread )
    break;
  *ppthread = 0;
  thread_struct[v5].uuid = uuid;
  thread_struct[v5].corr_id = corr_id;
  if ( pthread_create(&th[v5], 0LL, (void *(*)(void *))ReSolve0, &thread_struct[v5]) )
  {
    perror("pthread_create");
    exit(-1);
  }
  uuid = ppthread + 1;
  ++v5;
}
while ( v5 <= 589 );
thread_struct[v5].uuid = uuid;
thread_struct[v5].corr_id = corr_id;
if ( pthread_create(&th[v5], 0LL, (void *(*)(void *))ReSolve0, &thread_struct[v5]) )
{
  perror("Pthread_create");
  exit(-1);
}
for ( i = 0; i <= v5; ++i )
  pthread_join(th[i], 0LL);
```

Double Fetch

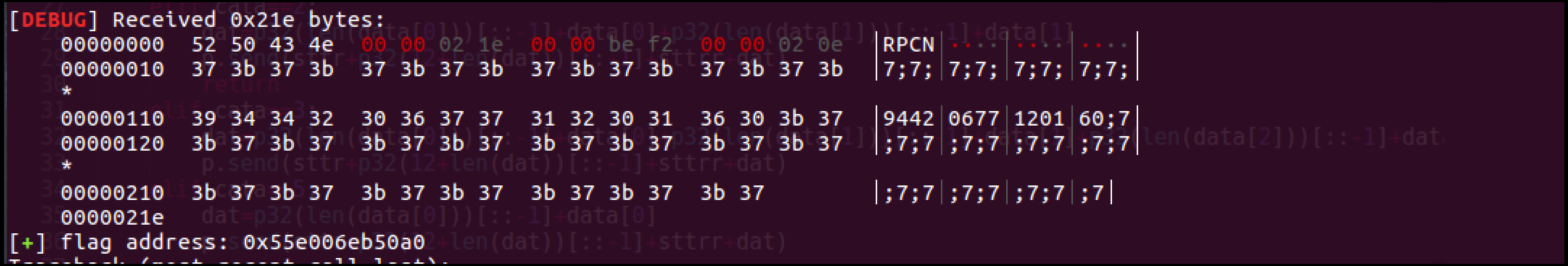
- ReSolve0()

- When multi-thread is executing, current_user (global var in .bss) can be tampered after check



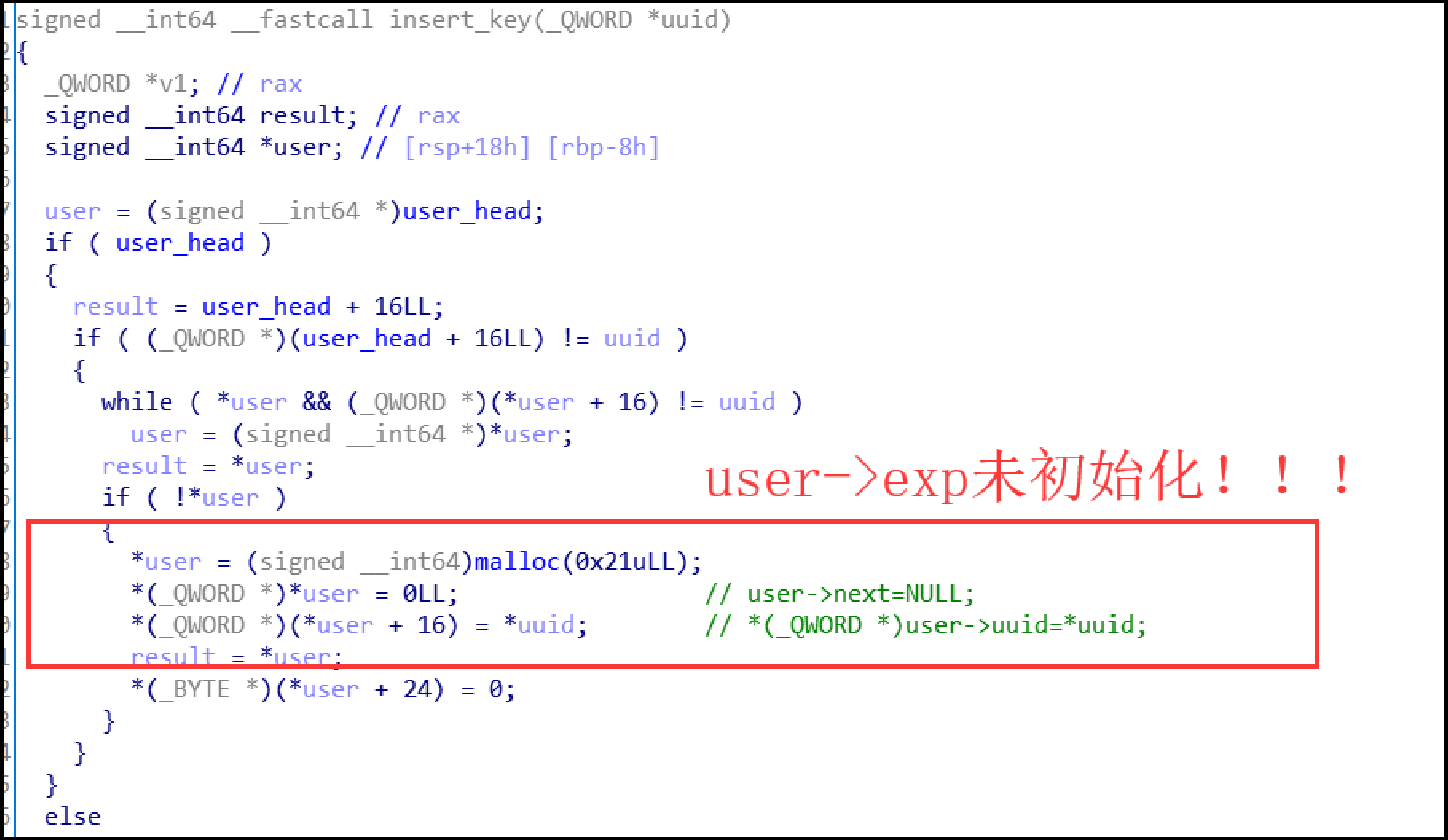
```
void __fastcall ReSolve0(void *a1)
{
  struct expr *ptr; // ST18_8

  current_user = find_user(*(_QWORD *)a1);
  if ( !current_user )
    exit(-1);
  if ( !strcmp(current_user->uuid, "admin") )
  {
    write(1, retry_packet, 0xCuLL);
  }
  else if ( current_user->exp )
  {
    if ( !strcmp(current_user->exp->expr_id, *((const char **)a1 + 1)) )
    {
      *((_QWORD *)a1 + 2) = malloc(0x30uLL);
      sprintf(*((char **)a1 + 2), "%ld", current_user->exp->res);
      ptr = current_user->exp;
      current_user->exp = current_user->exp->next;
      free(ptr->expr_id);
      free(ptr);
    }
    else
    {
      write(1, retry_packet, 0xCuLL);
    }
  }
  else
  {
    write(1, retry_packet, 0xCuLL);
  }
}
```

Get the flag's address



```
[DEBUG] Received 0x21e bytes:
    00000000  52 50 43 4e  00 00 02 1e  00 00 be f2  00 00 02 0e  │RPCN│····│····│····│
    00000010  37 3b 37 3b  37 3b 37 3b  37 3b 37 3b  37 3b 37 3b  │7;7;│7;7;│7;7;│7;7;│
    *
    00000110  39 34 34 32  30 36 37 37  31 32 30 31  36 30 3b 37  │9442│0677│1201│60;7│
    00000120  3b 37 3b 37  3b 37 3b 37  3b 37 3b 37  3b 37 3b 37  │;7;7│;7;7│;7;7│;7;7│
    *
    00000210  3b 37 3b 37  3b 37 3b 37  3b 37 3b 37  3b 37        │;7;7│;7;7│;7;7│;7│
    0000021e
[+] flag address: 0x55e006eb50a0
```

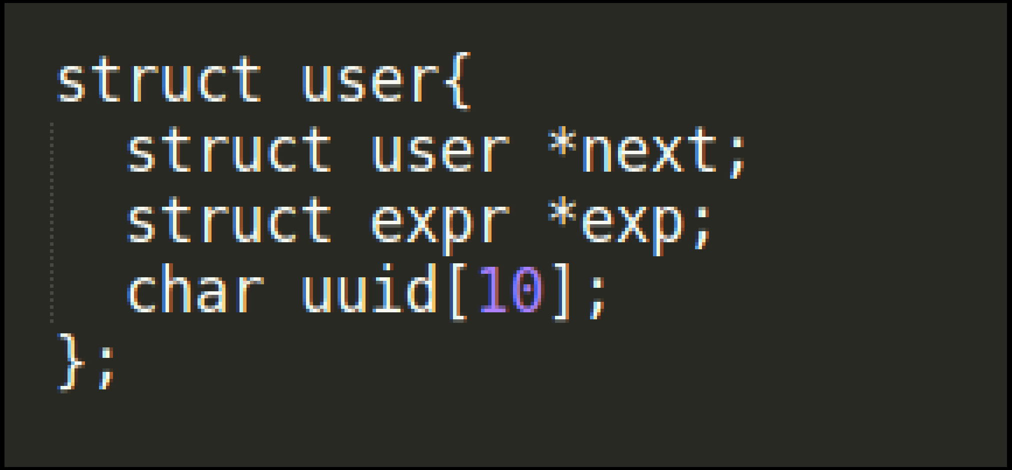
Uninitialized Data



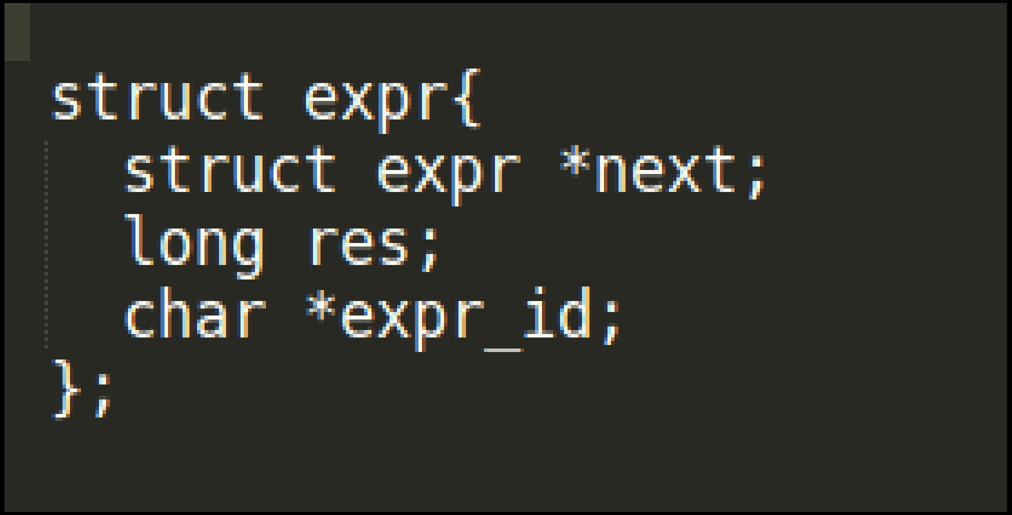
```
signed __int64 __fastcall insert_key(_QWORD *uuid)
{
  _QWORD *v1; // rax
  signed __int64 result; // rax
  signed __int64 *user; // [rsp+18h] [rbp-8h]

  user = (signed __int64 *)user_head;
  if ( user_head )
  {
    result = user_head + 16LL;
    if ( (_QWORD *)(user_head + 16LL) != uuid )
    {
      while ( *user && (_QWORD *)(*user + 16) != uuid )
        user = (signed __int64 *)*user;
      result = *user;
      if ( !*user )
      {
        *user = (signed __int64)malloc(0x21uLL);
        *(_QWORD *)*user = 0LL;                 // user->next=NULL;
        *(_QWORD *)(*user + 16) = *uuid;        // *(_QWORD *)user->uuid=*uuid;
        result = *user;
        *(_BYTE *)(*user + 24) = 0;
      }
    }
  }
  else
```

user->exp未初始化！！！

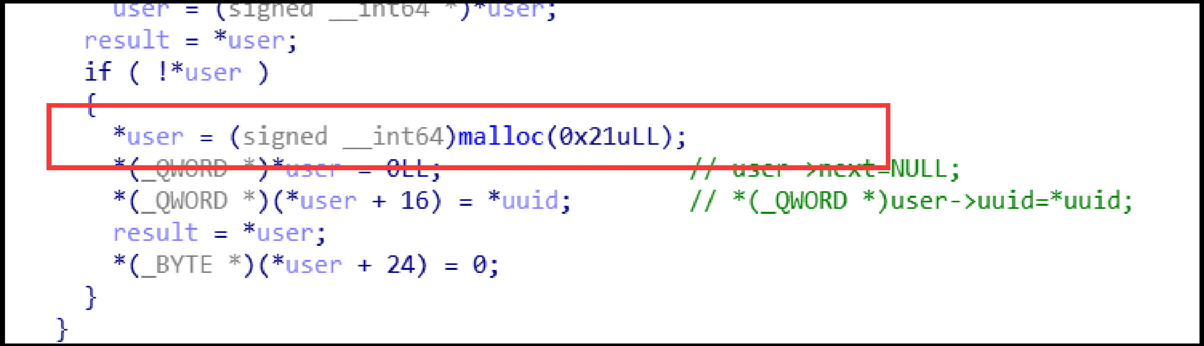
Forge the struct



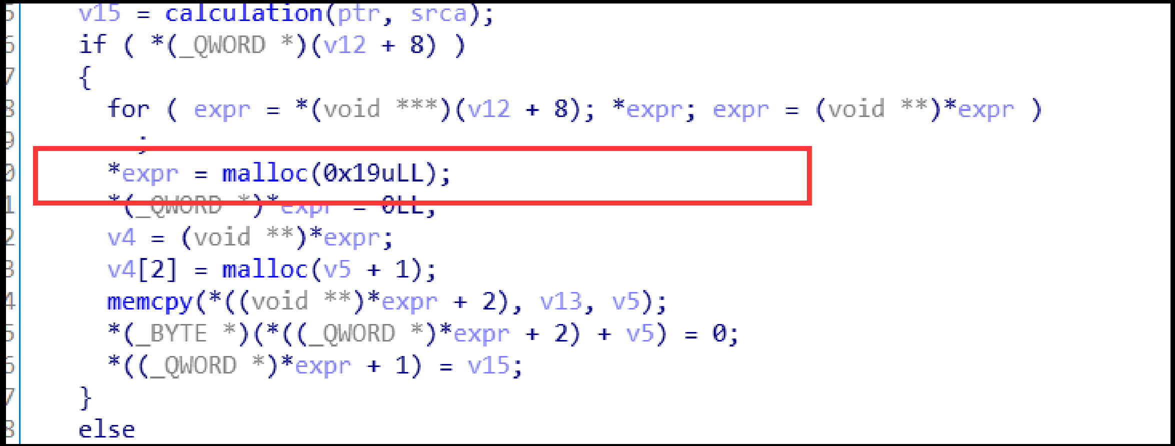
```
struct user{
  struct user *next;
  struct expr *exp;
  char uuid[10];
};
```



```
struct expr{
  struct expr *next;
  long res;
  char *expr_id;
};
```

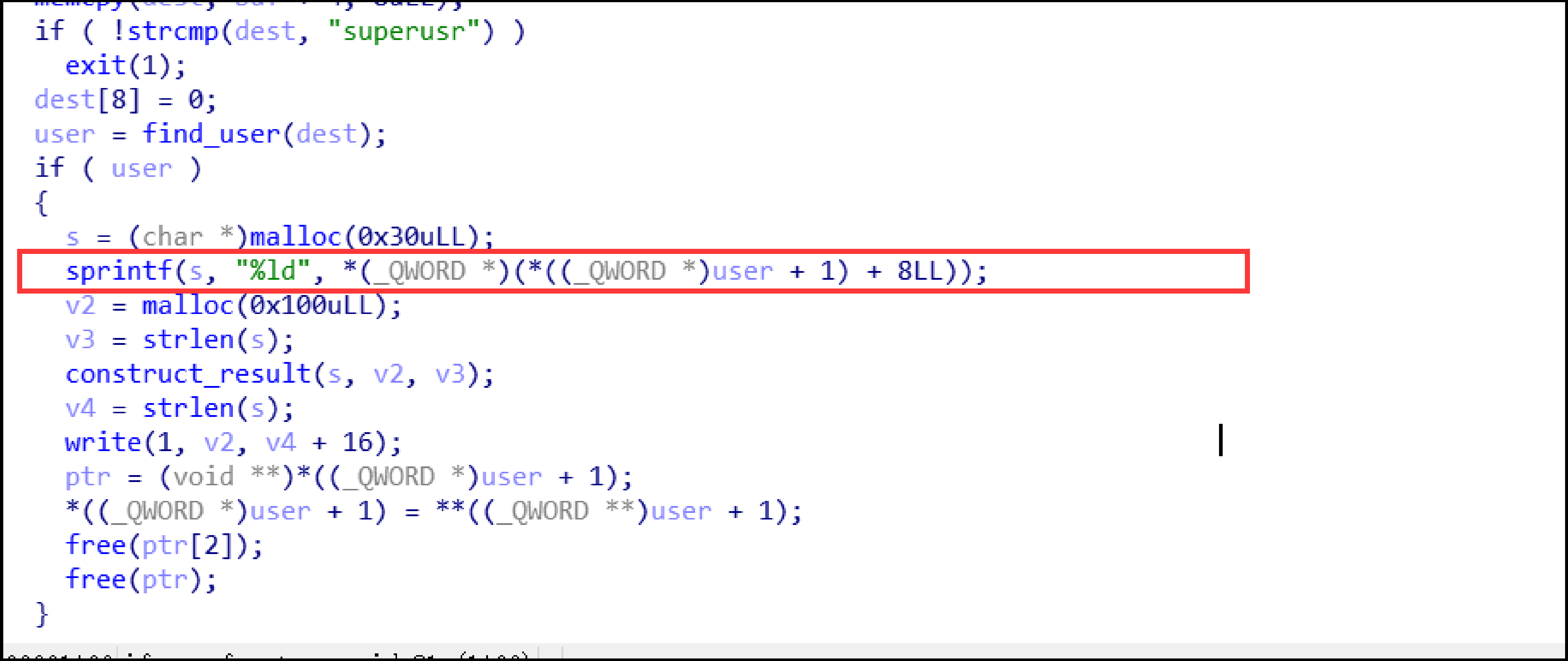


```
    user = (signed __int64 *)*user;
  result = *user;
  if ( !*user )
  {
    *user = (signed __int64)malloc(0x21uLL);
    *(_QWORD *)*user = 0LL;                     // user->next=NULL;
    *(_QWORD *)(*user + 16) = *uuid;            // *(_QWORD *)user->uuid=*uuid;
    result = *user;
    *(_BYTE *)(*user + 24) = 0;
  }
}
```



```
  v15 = calculation(ptr, srca);
  if ( *(_QWORD *)(v12 + 8) )
  {
    for ( expr = *(void ***)(v12 + 8); *expr; expr = (void **)*expr )
      ;
    *expr = malloc(0x19uLL);
    *(_QWORD *)*expr = 0LL;
    v4 = (void **)*expr;
    v4[2] = malloc(v5 + 1);
    memcpy(*((void **)*expr + 2), v13, v5);
    *(_BYTE *)(*((_QWORD *)*expr + 2) + v5) = 0;
    *((_QWORD *)*expr + 1) = v15;
  }
  else
```

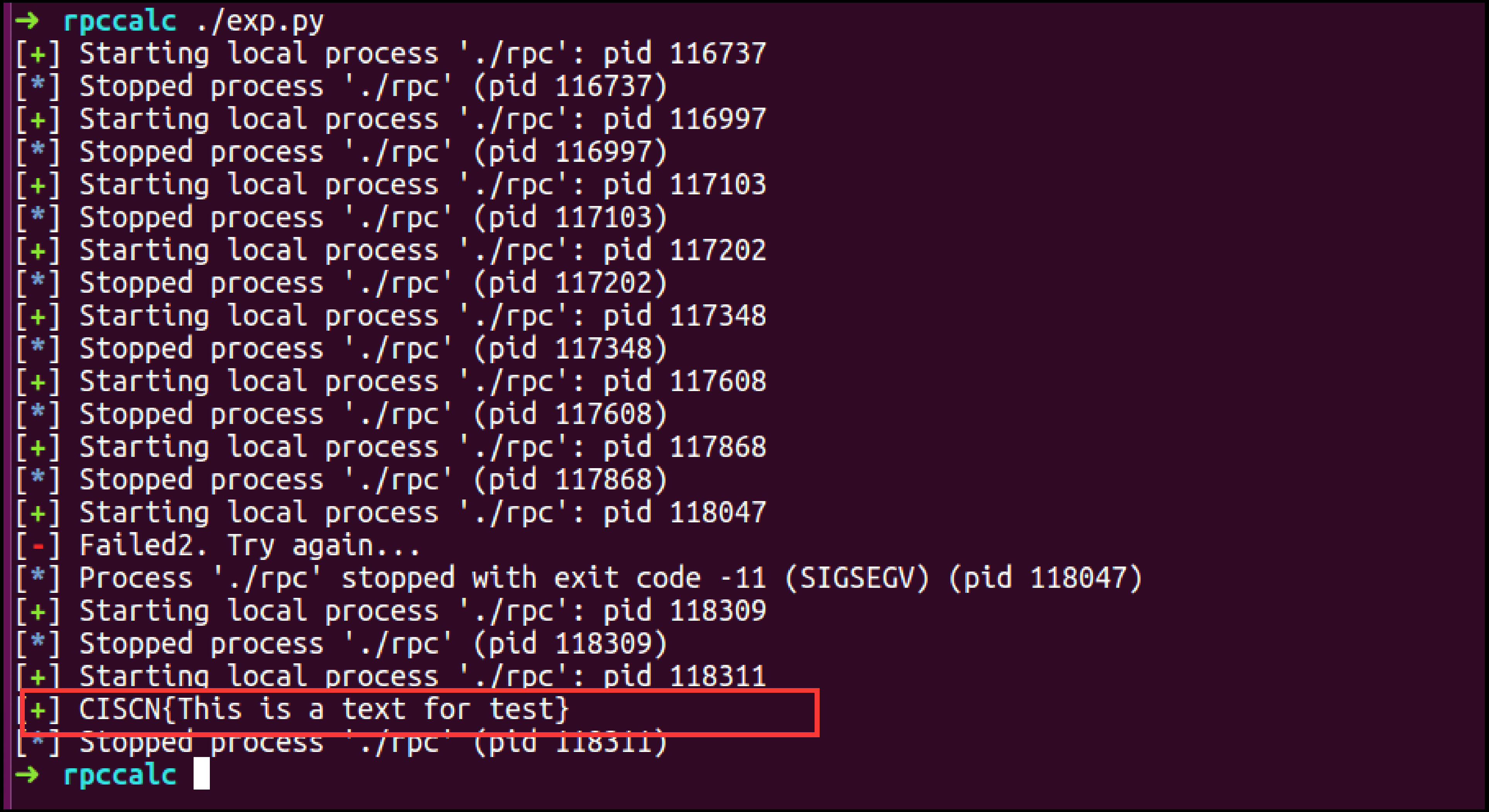
Arbitrary read



```
if ( !strcmp(dest, "superusr") )
  exit(1);
dest[8] = 0;
user = find_user(dest);
if ( user )
{
  s = (char *)malloc(0x30uLL);
  sprintf(s, "%ld", *(_QWORD *)(*((_QWORD *)user + 1) + 8LL));
  v2 = malloc(0x100uLL);
  v3 = strlen(s);
  construct_result(s, v2, v3);
  v4 = strlen(s);
  write(1, v2, v4 + 16);
  ptr = (void **)*((_QWORD *)user + 1);
  *((_QWORD *)user + 1) = **((_QWORD **)user + 1);
  free(ptr[2]);
  free(ptr);
}
```

GET FLAG!

It can really take much time



```
➜  rpccalc ./exp.py
[+] Starting local process './rpc': pid 116737
[*] Stopped process './rpc' (pid 116737)
[+] Starting local process './rpc': pid 116997
[*] Stopped process './rpc' (pid 116997)
[+] Starting local process './rpc': pid 117103
[*] Stopped process './rpc' (pid 117103)
[+] Starting local process './rpc': pid 117202
[*] Stopped process './rpc' (pid 117202)
[+] Starting local process './rpc': pid 117348
[*] Stopped process './rpc' (pid 117348)
[+] Starting local process './rpc': pid 117608
[*] Stopped process './rpc' (pid 117608)
[+] Starting local process './rpc': pid 117868
[*] Stopped process './rpc' (pid 117868)
[+] Starting local process './rpc': pid 118047
[-] Failed2. Try again...
[*] Process './rpc' stopped with exit code -11 (SIGSEGV) (pid 118047)
[+] Starting local process './rpc': pid 118309
[*] Stopped process './rpc' (pid 118309)
[+] Starting local process './rpc': pid 118311
[+] CISCN{This is a text for test}
[*] Stopped process './rpc' (pid 118311)
➜  rpccalc
```

Thank you

Q&A?

—— SJTU Ph0t1n1a ——