REVERSE ENGINEERING

AGENDA

- Lab 0 : Reversing an ARM binary
  - Find the patch logic using GDB
  - Using Ghidra and Hopper to understand ARM binaries.
- Lab 1: Reversing an ARM binary
  - Find the patch logic using GDB.
  - Create a patch and run in GDB.

REVERSING BINARY : LAB1

REVERSE ENGINEERING : GDB

- Reverse engineer binary ‘rev1’ to print “Yes, xx is correct” by giving any value in input.

```
maverick@maverick-workforce:~/Documents/teaching/Brno_23/trg/week3/seminar_raspi3$ cat rev1.c
#include <stdio.h>
#include <string.h>

int main(int argc, char** argv) {

    if (argc != 2) {
        printf("Need exactly one argument.\n");
        return -1;
    }

    char* correct = "xxxxxxxxxxxxxxxx";

    if (strncmp(argv[1], correct, strlen(correct))) {
        printf("No, %s is not correct.\n", argv[1]);
        return 1;
    } else {
        printf("Yes, %s is correct!\n", argv[1]);
        return 0;
    }

}
```

gcc -o rev1 rev1.c

REVERSE ENGINEERING : GDB

- Step 1 : Learn about the binary type.

```
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $ objdump -f rev1

rev1:     file format elf32-littlearm
architecture: arm, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x0001038c

pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
```

REVERSE ENGINEERING : GDB

- Step 2 : Disassemble the binary

```
000104b4 <main>:
   104b4:       e92d4810        push    {r4, fp, lr}
   104b8:       e28db008        add     fp, sp, #8
   104bc:       e24dd014        sub     sp, sp, #20
   104c0:       e50b0018        str     r0, [fp, #-24]  ; 0xfffffff8
   104c4:       e50b101c        str     r1, [fp, #-28]  ; 0xfffffff4
   104c8:       e51b3018        ldr     r3, [fp, #-24]  ; 0xfffffff8
   104cc:       e3530002        cmp     r3, #2
   104d0:       0a000003        beq     104e4 <main+0x30>
   104d4:       e59f008c        ldr     r0, [pc, #140]  ; 10568 <main+0xb4>
   104d8:       ebffff99        bl      10344 <puts@plt>
   104dc:       e3e03000        mvn     r3, #0
   104e0:       ea00001d        b       1055c <main+0xa8>
   104e4:       e59f3080        ldr     r3, [pc, #128]  ; 1056c <main+0xb8>
   104e8:       e50b3010        str     r3, [fp, #-16]
   104ec:       e51b301c        ldr     r3, [fp, #-28]  ; 0xfffffff4
   104f0:       e2833004        add     r3, r3, #4
   104f4:       e5934000        ldr     r4, [r3]
   104f8:       e51b0010        ldr     r0, [fp, #-16]
   104fc:       ebffff99        bl      10368 <strlen@plt>
   10500:       e1a03000        mov     r3, r0
   10504:       e1a00004        mov     r0, r4
   10508:       e51b1010        ldr     r1, [fp, #-16]
   1050c:       e1a02003        mov     r2, r3
   10510:       ebffff97        bl      10374 <strncmp@plt>
   10514:       e1a03000        mov     r3, r0
   10518:       e3530000        cmp     r3, #0
   1051c:       0a000007        beq     10540 <main+0x8c>
   10520:       e51b301c        ldr     r3, [fp, #-28]  ; 0xfffffff4
   10524:       e2833004        add     r3, r3, #4
   10528:       e5933000        ldr     r3, [r3]
   1052c:       e59f003c        ldr     r0, [pc, #60]   ; 10570 <main+0xbc>
   10530:       e1a01003        mov     r1, r3
   10534:       ebffff7f        bl      10338 <printf@plt>
   10538:       e3a03001        mov     r3, #1
   1053c:       ea000006        b       1055c <main+0xa8>
   10540:       e51b301c        ldr     r3, [fp, #-28]  ; 0xfffffff4
   10544:       e2833004        add     r3, r3, #4
   10548:       e5933000        ldr     r3, [r3]
   1054c:       e59f0020        ldr     r0, [pc, #32]   ; 10574 <main+0xc0>
   10550:       e1a01003        mov     r1, r3
   10554:       ebffff77        bl      10338 <printf@plt>
   10558:       e3a03000        mov     r3, #0
   1055c:       e1a00003        mov     r0, r3
   10560:       e24bd008        sub     sp, fp, #8
   10564:       e8bd8810        pop     {r4, fp, pc}
   10568:       000105ec        .word   0x000105ec
   1056c:       00010608        .word   0x00010608
   10570:       00010610        .word   0x00010610
   10574:       00010628        .word   0x00010628
```

REVERSE ENGINEERING : GDB

- Step 3 : Run the binary

```
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $ ./rev1
Need exactly one argument.
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $ ./rev1 aaaa
No, aaaa is not correct.
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $ ./rev1 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
No, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa is not correct.
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $ ./rev1 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
No, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa is not correct.
pi@raspberrypi:~/Desktop/binary_exploitation/codes/seminar_3 $
```

REVERSE ENGINEERING : GDB

- Step 4 : Inspect assembly in GDB/Ghidra/Hopper/BinaryNinja
- Step 5 : Find the patch logic
- Step 6 : Create a patched file

REVERSING BINARY : LAB2

REVERSE ENGINEERING : GHIDRA

- Reverse engineer binary ‘challenge1’ and understand the program logic.
- Find the patch logic to print “You won !!” by giving any arbitrary value as input.
- Create a patched binary using Ghidra/Hopper/BinaryNinja.

Note:

Ghidra Commands
https://ghidra-sre.org/CheatSheet.html
Patching Ghidra
https://materials.rangeforce.com/tutorial/2020/04/12/Patching-Binaries/

HOMEWORK 3

[EASY] (3 marks)

Crack the binary **rev2** and generate a patched version as **rev2_patched**. Explain the logic of the password by creating the code **rev2.c**.

[NOT EASY] (2 marks).

Crack the binary **challenge2** by reverse engineering and explain a way to print "Access granted enjoy".

[HARD](bonus 1 mark)

Crack the binary **challenge2** and generate a patched version as **challenge2_patched** to print "Access granted enjoy" on giving any arbitrary inputs.

Questions

Reverse Engineering