Linux
2021 03 13

https://github.com/MyDearGreatTeacher/2021_1_courses/tree/main/CTF%20%E6%90%B6%E6%97%97%E5%A4%A7%E8%B3%BD%E5%AF%A6%E5%8B%99%E6%8A%80%E8%A1%93/WEEK2

A_0_組合語言程式語法與解讀_起手式.md

```
產生組合語言
(1)使用gcc/g++產生組合語言    ==> gcc/g++
(2)使用逆向程式工具把執行檔逆回組合語言 ==> objdump
(3)直接使用nasm/masm/fasm/gas/....撰寫組合語言程式 ==> GAS nasm
```

```
// helloCTFer.c
#include <stdio.h>

int main()
{
   printf("Hello CTFer\n ”);
   return 0;
}
```

(1)使用gcc/g++產生組合語言

```
[32bit與64bit組合語言不一樣!!]

產生預設的AT&T 語法的組合語言
    gcc -S func1.c -o func1.s

產生intel 語法的組合語言
    gcc -S -masm=intel func1.c -o func1_intel.s

gcc -S -masm=intel func1.c -o func1_intel.s -fno-asynchronous-unwind-tables
```

```
// helloCTFer.c
#include <stdio.h>

int main()
{
   printf("Hello CTFer\n ”);
   return 0;
}
```

(2)使用逆向程式工具把執行檔逆回組合語言 ==> objdump

```
gcc helloCTFer.c -o helloCTFer -g
    原始C程式        執行檔       重要編譯參數
```

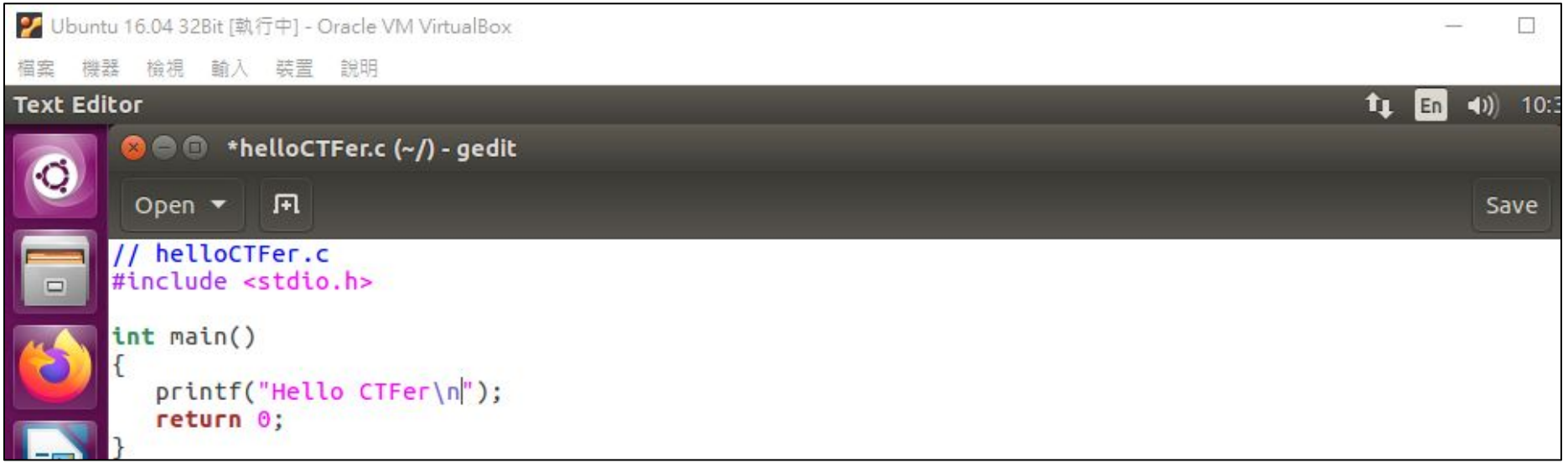
```
objdump -S helloCTFer  ==> AT&T

objdump -S -M intel helloCTFer

objdump -S -j .text -M intel helloCTFer

objdump -S -j .text -M intel helloCTFer --no-show-raw-insn
```

```
ksu@KSU-Ubuntu-1604-32:~$ gedit helloCTFer.c
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.c -o helloCTFer -g
```



原始C程式 ==> 1.c

```
// helloCTFer.c  1.c
#include <stdio.h>

int main()
{
   printf("Hello CTFer\n ”);
   return 0;
}
```

連結靜態庫（該庫只有hello函數）生成的可執行文件反編譯結果：

```
#分步驟編譯：
#1) 預處理
gcc -E test.c -o test.i
#在當前目錄下會多出一個預處理結果文件 test.i，打開 test.i 可以看到，在test.c 的基礎上把stdio.h和stdlib.h的內容插進去了。

#2)編譯為彙編代碼
gcc -S test.i -o test.s
#其中-S參數是在編譯完成後退出，-o為指定文件名。

#3) 彙編為目標文件
gcc -c test.s -o test.o
#.o就是目標文件。目標文件與可執行文件類似，都是機器能夠識別的可執行代碼，但是由於還沒有連結，結構會稍有不同。

#3) 連結並生成可執行文件
gcc test.o -o test

#4)在linux下對利用反彙編器對.o文件進行反彙編。
objdump -d test.o
```


Ubuntu 16.04 32Bit [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
File Edit View Search Tools Documents Help
En
10:59
Open
Save
// helloCTFer.c 1.c
#include <stdio.h>
int main()
{
printf("Hello CTFer\n ");
return 0;
}

```
ksu@KSU-Ubuntu-1604-32:~$ gedit 1.c
ksu@KSU-Ubuntu-1604-32:~$ gcc -S 1.c -o 1.s
ksu@KSU-Ubuntu-1604-32:~$ gcc -c 1.s -o 1.o
ksu@KSU-Ubuntu-1604-32:~$ gcc 1.o -o 1
ksu@KSU-Ubuntu-1604-32:~$ objdump -d 1.o

1.o:     file format elf32-i386


Disassembly of section .text:

00000000 <main>:
   0:   8d 4c 24 04             lea    0x4(%esp),%ecx
   4:   83 e4 f0                and    $0xfffffff0,%esp
   7:   ff 71 fc                pushl  -0x4(%ecx)
   a:   55                      push   %ebp
   b:   89 e5                   mov    %esp,%ebp
   d:   51                      push   %ecx
   e:   83 ec 04                sub    $0x4,%esp
  11:   83 ec 0c                sub    $0xc,%esp
  14:   68 00 00 00 00          push   $0x0
  19:   e8 fc ff ff ff          call   1a <main+0x1a>
  1e:   83 c4 10                add    $0x10,%esp
  21:   b8 00 00 00 00          mov    $0x0,%eax
  26:   8b 4d fc                mov    -0x4(%ebp),%ecx
  29:   c9                      leave
  2a:   8d 61 fc                lea    -0x4(%ecx),%esp
  2d:   c3                      ret
```