

CTF 搶旗大賽實務技術

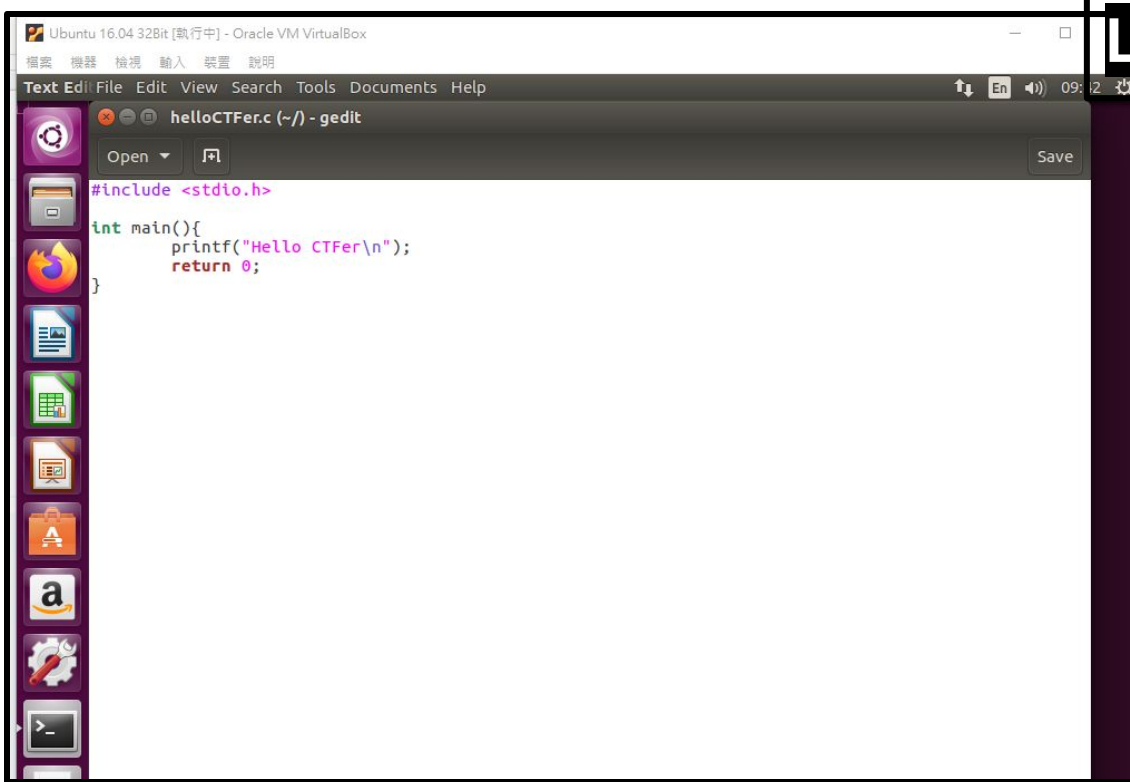
平時報告 1

實戰報告綜合

學生:王昱凱

指導教授: 偉大的恩師 龍大大

2021/3/13



Linux C 程式的編譯與運行:

開記事本編輯程式碼儲存

Linux C 程式的編譯與運行:

```
ksu@KSU-Ubuntu-1604-32:~$ gedit helloCTFer.c
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.c
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.c -o helloCTFer
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.c -o helloCTFer.exe
ksu@KSU-Ubuntu-1604-32:~$ ./a.out
Hello CTFer
ksu@KSU-Ubuntu-1604-32:~$ ./helloCTFer
Hello CTFer
ksu@KSU-Ubuntu-1604-32:~$ ./helloCTFer.exe
Hello CTFer
ksu@KSU-Ubuntu-1604-32:~$ file ./a.out
./a.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripped
ksu@KSU-Ubuntu-1604-32:~$ file ./helloCTFer
./helloCTFer: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripped
ksu@KSU-Ubuntu-1604-32:~$ file ./helloCTFer.exe
./helloCTFer.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripped
```

(1) 編譯

```
==> gcc helloCTFer.c ==> 產生a.out執行檔
==> gcc helloCTFer.c -o helloCTFer
==> gcc helloCTFer.c -o helloCTFer.exe
```

(2) 執行

```
==> ./a.out
==> ./helloCTFer
==> ./helloCTFer.exe
```

(3) 檢查執行檔檔案格式

```
==> file ./a.out
==> file ./helloCTFer
==> file ./helloCTFer.exe
```

Linux C 程式的編譯與運行:

```
ksu@KSU-Ubuntu-1604-32:~$ file helloCTFer.exe
helloCTFer.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripped
```

```
ksu@KSU-Ubuntu-1604-32:~$ file helloCTFer.exe
helloCTFer.exe: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=13d33ba42c32245aa46436a2f22dfd4875c2fb45, not stripped
```

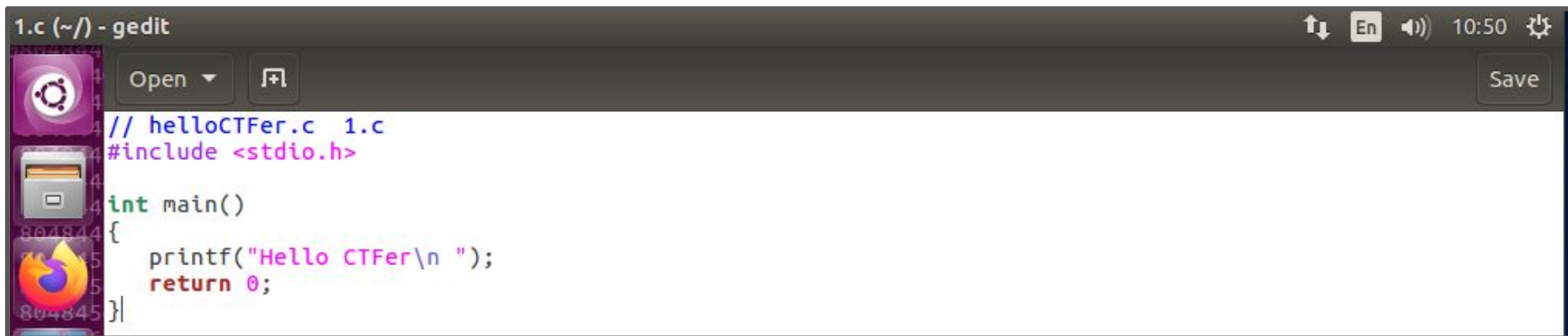
[查看該檔案執行位元...等資訊](#)

objdump

objdump是在類Unix作業系統上顯示關於目的檔的各種資訊的命令行程式。

例如，它可用作反組譯器來以組譯代碼形式檢視可執行檔。

它是GNU Binutils的一部分，用於在可執行檔和其他二進位資料上進行精細粒度控制。objdump使用BFD庫來讀取目的檔的內容。



```
1.c (~/) - gedit
// helloCTFer.c 1.c
#include <stdio.h>

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

開記事本編輯程式碼儲

存

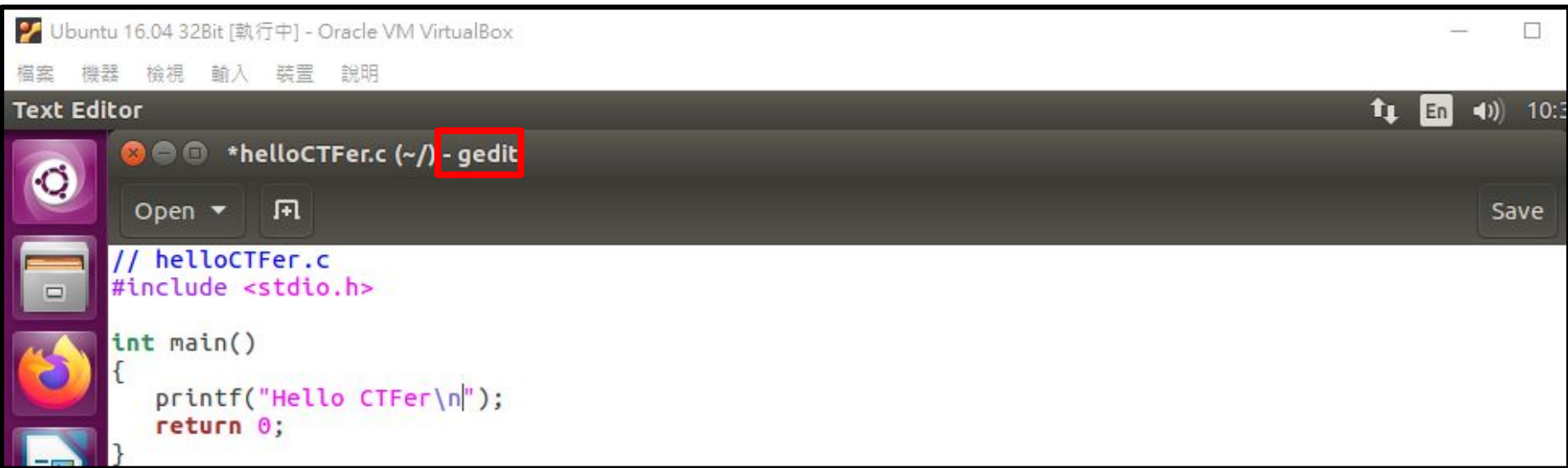
objdump

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

原始c程式

執行檔

重要編譯參數



gedit 檔案+副檔名 開記事本編輯程式碼儲存

objdump

```
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    $0xffffffff0,%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  %ecx
2a: 8d 61 fc        lea    -0x4(%ecx),%esp
2d: c3              ret
```

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
```


Linux C 程式的編譯與運行:

```
ls -al helloCTFer.*
```

```
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*  
-rw-rw-r-- 1 ksu ksu 73  6 12:23 helloCTFer.c  
-rwxrwxr-x 1 ksu ksu 7356 6 12:24 helloCTFer.exe
```

Linux C 程式的編譯與運行:

```
hexdump -C helloCTFer.exe
```

```
00000000  7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00 |.ELF.....|
00000010  02 00 03 00 01 00 00 00 10 83 04 08 34 00 00 00 |.....4...|
00000020  e0 17 00 00 00 00 00 00 34 00 20 00 09 00 28 00 |.....4. ...(.|
00000030  1f 00 1c 00 06 00 00 00 34 00 00 00 34 80 04 08 |.....4...4...|
00000040  34 80 04 08 20 01 00 00 20 01 00 00 05 00 00 00 |4... ... ..|
00000050  04 00 00 00 03 00 00 00 54 01 00 00 54 81 04 08 |.....T...T...|
00000060  54 81 04 08 13 00 00 00 13 00 00 00 04 00 00 00 |T.....|
00000070  01 00 00 00 01 00 00 00 00 00 00 00 00 80 04 08 |.....|
00000080  00 80 04 08 c4 05 00 00 c4 05 00 00 05 00 00 00 |.....|
```

.....

```
ksu@KSU-Ubuntu-1604-32:~$ hexdump -C helloCTFer.exe
00000000  7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00 |.ELF.....|
00000010  02 00 03 00 01 00 00 00 10 83 04 08 34 00 00 00 |.....4...|
00000020  e4 17 00 00 00 00 00 00 34 00 20 00 09 00 28 00 |.....4. ...(.|
00000030  1f 00 1c 00 06 00 00 00 34 00 00 00 34 80 04 08 |.....4...4...|
00000040  34 80 04 08 20 01 00 00 20 01 00 00 05 00 00 00 |4... ... ..|
00000050  04 00 00 00 03 00 00 00 54 01 00 00 54 81 04 08 |.....T...T...|
00000060  54 81 04 08 13 00 00 00 13 00 00 00 04 00 00 00 |T.....|
00000070  01 00 00 00 01 00 00 00 00 00 00 00 00 80 04 08 |.....|
00000080  00 80 04 08 c8 05 00 00 c8 05 00 00 05 00 00 00 |.....|
00000090  00 10 00 00 01 00 00 00 08 0f 00 00 08 9f 04 08 |.....|
000000a0  08 9f 04 08 14 01 00 00 18 01 00 00 06 00 00 00 |.....|
000000b0  00 10 00 00 02 00 00 00 14 0f 00 00 14 9f 04 08 |.....|
000000c0  14 9f 04 08 e8 00 00 00 e8 00 00 00 06 00 00 00 |.....|
000000d0  04 00 00 00 04 00 00 00 68 01 00 00 68 81 04 08 |.....h...h...|
000000e0  68 81 04 08 44 00 00 00 44 00 00 00 04 00 00 00 |h...D...D.....|
000000f0  04 00 00 00 50 e5 74 64 d0 04 00 00 d0 84 04 08 |....P.td.....|
00000100  d0 84 04 08 2c 00 00 00 2c 00 00 00 04 00 00 00 |.....|
00000110  04 00 00 00 51 e5 74 64 00 00 00 00 00 00 00 00 |....Q.td.....|
00000120  00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 00 |.....|
00000130  10 00 00 00 52 e5 74 64 08 0f 00 00 08 9f 04 08 |....R.td.....|
00000140  08 9f 04 08 f8 00 00 00 f8 00 00 00 04 00 00 00 |.....|
00000150  01 00 00 00 2f 6c 69 62 2f 6c 64 2d 6c 69 6e 75 |.../lib/ld-linu|
00000160  78 2e 73 6f 2e 32 00 00 04 00 00 00 10 00 00 00 |x.so.2.....|
```

編譯的各階段

原始程式碼 ===> helloCTFer.c

預處理 ===> gcc -E helloCTFer.c -o helloCTFer.i ===> 查看.i的架構

編譯 ===> gcc -S helloCTFer.i -o helloCTFer.s ===> 查看.s的架構

彙編 ===> gcc -c helloCTFer.s -o helloCTFer.o ===> 查看.o的架構

連結 ===> gcc helloCTFer.o -o helloCTFer ===> 查看helloCTFer的架構

Linux C 程式的編譯與運行:

預處理

```
gcc -E helloCTFer.c -o helloCTFer.i
```

```
ls -al helloCTFer.*
```

```
ksu@KSU-Ubuntu-1604-32:~$ gcc -E helloCTFer.c -o helloCTFer.i
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*
-rw-rw-r-- 1 ksu ksu 73 6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu 7356 6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538 6 13:48 helloCTFer.i
```

Linux C 程式的編譯與運行:

```
cat helloCTFer.i
```

```
ksu@KSU-Ubuntu-1604-32:~$ cat helloCTFer.i
# 1 "helloCTFer.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 1 "<command-line>" 2
# 1 "helloCTFer.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 367 "/usr/include/features.h" 3 4
# 1 "/usr/include/i386-linux-gnu/sys/cdefs.h" 1 3 4
# 410 "/usr/include/i386-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/i386-linux-gnu/bits/wordsize.h" 1 3 4
# 411 "/usr/include/i386-linux-gnu/sys/cdefs.h" 2 3 4
# 368 "/usr/include/features.h" 2 3 4
# 391 "/usr/include/features.h" 3 4
# 1 "/usr/include/i386-linux-gnu/gnu/stubs.h" 1 3 4

# 1 "/usr/include/i386-linux-gnu/gnu/stubs-32.h" 1 3 4
# 8 "/usr/include/i386-linux-gnu/gnu/stubs.h" 2 3 4
# 392 "/usr/include/features.h" 2 3 4
# 28 "/usr/include/stdio.h" 2 3 4

# 1 "/usr/lib/gcc/i686-linux-gnu/5/include/stddef.h" 1 3 4
# 216 "/usr/lib/gcc/i686-linux-gnu/5/include/stddef.h" 3 4

# 216 "/usr/lib/gcc/i686-linux-gnu/5/include/stddef.h" 3 4
typedef unsigned int size_t;
# 34 "/usr/include/stdio.h" 2 3 4
```

Linux C 程式的編譯與運行:

編譯

```
gcc -S helloCTFer.i -o helloCTFer.s
```

```
ls -al helloCTFer.*
```

```
ksu@KSU-Ubuntu-1604-32:~$ gcc -S helloCTFer.i -o helloCTFer.s
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*
-rw-rw-r-- 1 ksu ksu 73 6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu 7356 6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538 6 13:48 helloCTFer.i
-rw-rw-r-- 1 ksu ksu 664 6 13:53 helloCTFer.s
```


AT&T

cat helloCTFer.s ==>

Linux C 程式的編譯與運行:

```
ksu@KSU-Ubuntu-1604-32:~$ cat helloCTFer.s
.file "helloCTFer.c"
.section .rodata
.LC0:
.string "Hello CTFer\n "
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
leal 4(%esp), %ecx
.cfi_def_cfa 1, 0
andl $-16, %esp
pushl -4(%ecx)
pushl %ebp
.cfi_escape 0x10,0x5,0x2,0x75,0
movl %esp, %ebp
pushl %ecx
.cfi_escape 0xf,0x3,0x75,0x7c,0x6
subl $4, %esp
subl $12, %esp
pushl $.LC0
call printf
addl $16, %esp
movl $0, %eax
movl -4(%ebp), %ecx
.cfi_def_cfa 1, 0
leave
.cfi_restore 5
leal -4(%ecx), %esp
.cfi_def_cfa 4, 4
ret
.cfi_endproc
.LFE0:
.size main, .-main
.ident "GCC: (Ubuntu 5.4.0-6ubuntu1-16.04.12) 5.4.0 20160609"
.section .note.GNU-stack,"",@progbits
```


Linux C 程式的編譯與運行:

```
ksu@KSU-Ubuntu-1604-32:~$ gcc -c helloCTFer.s -o helloCTFer.o
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer.*
-rw-rw-r-- 1 ksu ksu 73      6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu 7356    6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538   6 13:48 helloCTFer.i
-rw-rw-r-- 1 ksu ksu 1080    6 13:59 helloCTFer.o
-rw-rw-r-- 1 ksu ksu 664     6 13:53 helloCTFer.s
```

彙編

```
gcc -c helloCTFer.s -o helloCTFer.o
```

```
ls -al helloCTFer.*
```

Linux C 程式的編譯與運行:

查看helloCTFer.o

```
hexdump -C helloCTFer.o
```

```
ksu@KSU-Ubuntu-1604-32:~$ hexdump -C helloCTFer.o
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00 |.ELF.....|
00000010  01 00 03 00 01 00 00 00  00 00 00 00 00 00 00 00 |.....|
00000020  30 02 00 00 00 00 00 00  34 00 00 00 00 00 28 00 |0.....4....(|
00000030  0d 00 0a 00 8d 4c 24 04  83 e4 f0 ff 71 fc 55 89 |.....L$....q.U.|
00000040  e5 51 83 ec 04 83 ec 0c  68 00 00 00 00 e8 fc ff |.Q.....h.....|
00000050  ff ff 83 c4 10 b8 00 00  00 00 8b 4d fc c9 8d 61 |.....M...a|
00000060  fc c3 48 65 6c 6c 6f 20  43 54 46 65 72 0a 20 00 |..Hello CTFer. .|
00000070  00 47 43 43 3a 20 28 55  62 75 6e 74 75 20 35 2e |.GCC: (Ubuntu 5.|
00000080  34 2e 30 2d 36 75 62 75  6e 74 75 31 7e 31 36 2e |4.0-6ubuntu1~16.|
00000090  30 34 2e 31 32 29 20 35  2e 34 2e 30 20 32 30 31 |04.12) 5.4.0 201|
000000a0  36 30 36 30 39 00 00 00  14 00 00 00 00 00 00 00 |60609.....|
000000b0  01 7a 52 00 01 7c 08 01  1b 0c 04 04 88 01 00 00 |.zR..|.....|
000000c0  28 00 00 00 1c 00 00 00  00 00 00 00 2e 00 00 00 |(.|.....|
000000d0  00 44 0c 01 00 47 10 05  02 75 00 43 0f 03 75 7c |.D...G...u.C..u|
000000e0  06 5b 0c 01 00 41 c5 43  0c 04 04 00 00 00 00 00 |.[...A.C.....|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 01 00 00 00 |.....|
00000100  00 00 00 00 00 00 00 00  04 00 f1 ff 00 00 00 00 |.....|
00000110  00 00 00 00 00 00 00 00  03 00 01 00 00 00 00 00 |.....|
00000120  00 00 00 00 00 00 00 00  03 00 03 00 00 00 00 00 |.....|
00000130  00 00 00 00 00 00 00 00  03 00 04 00 00 00 00 00 |.....|
00000140  00 00 00 00 00 00 00 00  03 00 05 00 00 00 00 00 |.....|
00000150  00 00 00 00 00 00 00 00  03 00 07 00 00 00 00 00 |.....|
00000160  00 00 00 00 00 00 00 00  03 00 08 00 00 00 00 00 |.....|
00000170  00 00 00 00 00 00 00 00  03 00 06 00 0e 00 00 00 |.....|
00000180  00 00 00 00 2e 00 00 00  12 00 01 00 13 00 00 00 |.....|
00000190  00 00 00 00 00 00 00 00  10 00 00 00 00 68 65 6c |.....hel|
000001a0  6c 6f 43 54 46 65 72 2e  63 00 6d 61 69 6e 00 70 |loCTFer.c.main.p|
000001b0  72 69 6e 74 66 00 00 00  15 00 00 00 01 05 00 00 |rintf.....|
000001c0  1a 00 00 00 02 0a 00 00  20 00 00 00 02 02 00 00 |.....|
000001d0  00 2e 73 79 6d 74 61 62  00 2e 73 74 72 74 61 62 |..symtab..strtab|
000001e0  00 2e 73 68 73 74 72 74  61 62 00 2e 72 65 6c 2e |..shstrtab..rel.|
000001f0  74 65 78 74 00 2e 64 61  74 61 00 2e 62 73 73 00 |text..data..bss.|
00000200  2e 72 6f 64 61 74 61 00  2e 63 6f 6d 6d 65 6e 74 |.rodata..comment|
00000210  00 2e 6e 6f 74 65 2e 47  4e 55 2d 73 74 61 63 6b |..note.GNU-stack|
00000220  00 2e 72 65 6c 2e 65 68  5f 66 72 61 6d 65 00 00 |..rel.eh_frame..|
00000230  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|
*
00000250  00 00 00 00 00 00 00 00  1f 00 00 00 01 00 00 00 |.....|
```

Linux C 程式的編譯與運行:

```
file helloCTFer.o
```

```
ksu@KSU-Ubuntu-1604-32:~$ file helloCTFer.o  
helloCTFer.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not stripped
```

[查看helloCTFer.c執行位元...等資訊](#)

Linux C 程式的編譯與運行:

連結

```
gcc helloCTFer.o -o helloCTFer
```

```
./helloCTFer
```

```
ksu@KSU-Ubuntu-1604-32:~$ gcc helloCTFer.o -o helloCTFer
ksu@KSU-Ubuntu-1604-32:~$ ./helloCTFer
Hello CTFer
```

```
ls -al helloCTFer*
```

```
ksu@KSU-Ubuntu-1604-32:~$ ls -al helloCTFer*
-rwxrwxr-x 1 ksu ksu 7356  6 14:12 helloCTFer
-rw-rw-r-- 1 ksu ksu  73  6 12:23 helloCTFer.c
-rwxrwxr-x 1 ksu ksu 7356  6 12:24 helloCTFer.exe
-rw-rw-r-- 1 ksu ksu 17538 6 13:48 helloCTFer.i
-rw-rw-r-- 1 ksu ksu 1080  6 13:59 helloCTFer.o
-rw-rw-r-- 1 ksu ksu  664  6 13:53 helloCTFer.s
```


查看linux版本c函式庫版本

```
ksu@KSU-Ubuntu-1604-32:~$ cd /lib/i386-linux-gnu
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ls -al libc*
-rwxr-xr-x 1 root root 1790580 六 6 2020 libc-2.23.so
lrwxrwxrwx 1 root root 14 三 4 17:55 libcap.so.2 -> libcap.so.2.24
-rw-r--r-- 1 root root 18112 十 24 2015 libcap.so.2.24
lrwxrwxrwx 1 root root 21 三 4 17:55 libcgmanager.so.0 -> libcgmanager.so.0.0.0
-rw-r--r-- 1 root root 181564 一 19 2016 libcgmanager.so.0.0.0
-rw-r--r-- 1 root root 185932 六 6 2020 libcidn-2.23.so
lrwxrwxrwx 1 root root 15 六 6 2020 libcidn.so.1 -> libcidn-2.23.so
lrwxrwxrwx 1 root root 17 一 22 2020 libcom_err.so.2 -> libcom_err.so.2.1
-rw-r--r-- 1 root root 13816 二 22 2020 libcom_err.so.2.1
-rw-r--r-- 1 root root 38428 一 6 2020 libcrypto-2.23.so
-rw-r--r-- 1 root root 2005636 17 23:17 libcrypto.so.1.0.0
lrwxrwxrwx 1 root root 22 三 4 17:55 libcryptsetup.so.4 -> libcryptsetup.so.4.6.0
-rw-r--r-- 1 root root 185172 九 6 2017 libcryptsetup.so.4.6.0
lrwxrwxrwx 1 root root 16 六 6 2020 libcrypt.so.1 -> libcrypt-2.23.so
lrwxrwxrwx 1 root root 12 六 6 2020 libc.so.6 -> libc-2.23.so
```

查看linux版本c函式庫版本

```
ls -al libc.*
```

```
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ls -al libc.*  
lrwxrwxrwx 1 root root 12 六 6 2020 libc.so.6 -> libc-2.23.so
```

```
ldd --version
```

```
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ldd --version  
ldd (Ubuntu GLIBC 2.23-0ubuntu11.2) 2.23  
Copyright (C) 2016 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
Written by Roland McGrath and Ulrich Drepper.
```

```
./libc-2.23.so
```

```
ksu@KSU-Ubuntu-1604-32:/lib/i386-linux-gnu$ ./libc-2.23.so  
GNU C Library (Ubuntu GLIBC 2.23-0ubuntu11.2) stable release version 2.23, by Roland McGrath et al.  
Copyright (C) 2016 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions.  
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A  
PARTICULAR PURPOSE.  
Compiled by GNU CC version 5.4.0 20160609.  
Available extensions:  
  crypt add-on version 2.1 by Michael Glad and others  
  GNU Libidn by Simon Josefsson  
  Native POSIX Threads Library by Ulrich Drepper et al  
  BIND-8.2.3-T5B  
libc ABIs: UNIQUE IFUNC  
For bug reporting instructions, please see:  
<https://bugs.launchpad.net/ubuntu/+source/glibc/+bugs>.
```


CTF解題

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 History Bookmarks Tools Help

Reverse-CTF 2021_1_courses/CTF 搶

120.114.62.214/challenges#Reverse

Reverse-CTF 參加隊伍 討論板 競賽題目

林思

Challenge 14 Solves

Reverse

50

程式開發是許多資訊人員必備的技術，但開發安全的程式更是極度欠缺高階程式師。一般程式可能被逆向工程技術破解，請利用你所知道的逆向工程技術破解以下程式。

提示1:此題有多種方法破解，你可以嘗試看看提示2:可利用一般debugger工具輕鬆找出flag

reverse

Flag Submit

Level 1

Level 0

解題過程

```
Ubuntu 16.04 32Bit [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
ksu@KSU-Ubuntu-1604-32:~$ cd Downloads
ksu@KSU-Ubuntu-1604-32:~/Downloads$ ls
adder          ezreverse     hexedit       LuckyGuess    strace
Coffee.class   hexable       liar          reverse
ksu@KSU-Ubuntu-1604-32:~/Downloads$ file reverse
reverse: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=0eae3e9fd6d9a63980d737561cba79a68e655c9f, not stripped
ksu@KSU-Ubuntu-1604-32:~/Downloads$ ls -al reverse
-rw-rw-r-- 1 ksu ksu 7380  13 11:32 reverse
ksu@KSU-Ubuntu-1604-32:~/Downloads$ chmod +x reverse
ksu@KSU-Ubuntu-1604-32:~/Downloads$ ls -al reverse
-rwxrwxr-x 1 ksu ksu 7380  13 11:32 reverse
ksu@KSU-Ubuntu-1604-32:~/Downloads$ strings reverse
/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
gets
puts
__libc_start_main
__gnon_start__
GLIBC_2.0
PTRh
QVh;
UWVS
t$.U
[" ]
BreakALLCTF{4U49uY70JCrJL0vtbXjd}
rev again?
; *2$(
GCC: (Ubuntu 5.4.0-6ubuntu1-16.04.4) 5.4.0 20160609
crtstuff.c
_JCR_LIST__
deregister_tm_clones
__do_global_dtors_aux
completed.7200
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
re-1.c
__FRAME_END__
__JCR_END__
__init_array_end
DYNAMIC
__init_array_start
__GNU_EH_FRAME_HDR
__GLOBAL_OFFSET_TABLE__
__libc_csu_fini
__ITM_deregisterTMCloneTable
__x86.get_pc_thunk.bx
gets@@GLIBC_2.0
edata
data start
puts@@GLIBC_2.0
__gnon_start__
__dso_handle
__IO_stdin_used
```

解題成功

Level 1

Reverse



50

```
__libc_start_main@@GLIBC_2.0
__libc_csu_init
__fp_hw
__bss_start
main
_Jv_RegisterClasses
__TMC_END__
_ITM_registerTMCloneTable
.symtab
.strtab
.shstrtab
.interp
.note.ABI-tag
.note.gnu.build-id
.gnu.hash
.dynsym
.dynstr
.gnu.version
.gnu.version_r
.rel.dyn
.rel.plt
.init
.plt.got
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
.fini_array
.jcr
.dynamic
.got.plt
.data
.bss
.comment
ksu@KSU-Ubuntu-1604-32:~/Downloads$
```

CTF解題

Reverse-CTF x +

120.114.62.214/challenges#EasyCTF_adder

參加隊伍 排行榜 競賽題目 活動公告 解題狀況 隊伍資料

Challenge 22 Solves x

EasyCTF_adder

25

flag 格式為 easycctf{...}

add

Flag Submit

F_hexedit 25

EasyCTF_Hexable 25

CSIE_strace 25

EasyCTF_adder 25

林思辰_read-asm 50

解題過程

File Edit View Search Terminal Help

ksu@KSU-Ubuntu-1804-64:~\$ cd Downloads

ksu@KSU-Ubuntu-1804-64:~/Downloads\$ ls

adder

ksu@KSU-Ubuntu-1804-64:~/Downloads\$ file adder

adder: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=5247f49239e8007694bfa30ed2f96bb961f04c5c, not stripped

ksu@KSU-Ubuntu-1804-64:~/Downloads\$./adder

bash: ./adder: Permission denied

ksu@KSU-Ubuntu-1804-64:~/Downloads\$ ls -al adder

-rw-rw-r-- 1 ksu ksu 13440 13 13:55 adder

ksu@KSU-Ubuntu-1804-64:~/Downloads\$ chmod +x adder

ksu@KSU-Ubuntu-1804-64:~/Downloads\$ ls -al adder

-rwxrwxr-x 1 ksu ksu 13440 13 13:55 adder

ksu@KSU-Ubuntu-1804-64:~/Downloads\$ strings adder

/lib64/ld-linux-x86-64.so.2

libstdc++.so.6

__gmon_start__

_Jv_RegisterClasses

_ITM_deregisterTMCloneTable

_ITM_registerTMCloneTable

_ZNSt8ios_base4InitD1Ev

_ZSt3cin

_ZNSt8ios_base4InitD1Ev

_ZSt4cout

_ZStlsISt11char_traitsIceERSt13basic_ostreamIcT_ES5_PKc

_ZNSt8ios_base4InitC1Ev

libm.so.6

libgcc_s.so.1

libc.so.6

puts

putchar

__cxa_atexit

malloc

__libc_start_main

free

GLIBC_2.2.5

GLIBCXX_3.4

UH-x`

UH-x`

[JA\A]A^A_

Enter three numbers!

easyctf{

nope.

;*3\$"

GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-16)

.syntab

.strtab

.shstrtab

.interp

note_ABT_tag

解題過程

File Edit View Search Terminal Help

ksu@KSU-Ubuntu-1804-64:~/Downloads\$ objdump -S adder

adder: file format elf64-x86-64

Disassembly of section .init:

```
000000000400778 <.init>:
400778: 48 83 ec 08      sub    $0x8,%rsp
40077c: 48 8b 05 75 18 20 mov    0x201875(%rip),%rax      # 601ff8 <__gmon_start__>
400783: 48 85 c0         test   %rax,%rax
400786: 74 05          je     40078d <.init+0x15>
400788: e8 23 00 00 00   callq 4007b0 <__gmon_start__@plt>
40078d: 48 83 c4 08      add    $0x8,%rsp
400791: c3             retq
```

Disassembly of section .plt:

```
0000000004007a0 <_.plt>:
4007a0: ff 35 62 18 20 00 pushq  0x201862(%rip)      # 602008 <_GLOBAL_OFFSET_TABLE_+0x8>
4007a6: ff 25 64 18 20 00 jmpq   *0x201864(%rip)      # 602010 <_GLOBAL_OFFSET_TABLE_+0x10>
4007ac: 0f 1f 40 00      nopl   0x0(%rax)
```

```
>_ 0000000004007b0 <__gmon_start__@plt>:
4007b0: ff 25 62 18 20 00 jmpq   *0x201862(%rip)      # 602018 <__gmon_start__>
4007b6: 68 00 00 00 00   pushq  $0x0
4007bb: e9 e0 ff ff ff   jmpq   4007a0 <_.plt>
```

```
0000000004007c0 <puts@plt>:
4007c0: ff 25 5a 18 20 00 jmpq   *0x20185a(%rip)      # 602020 <puts@GLIBC_2.2.5>
4007c6: 68 01 00 00 00   pushq  $0x1
4007cb: e9 d0 ff ff ff   jmpq   4007a0 <_.plt>
```

```
0000000004007d0 <putchar@plt>:
4007d0: ff 25 52 18 20 00 jmpq   *0x201852(%rip)      # 602028 <putchar@GLIBC_2.2.5>
4007d6: 68 02 00 00 00   pushq  $0x2
4007db: e9 c0 ff ff ff   jmpq   4007a0 <_.plt>
```

```
0000000004007e0 <_ZNSt8ios_base4InitC1Ev@plt>:
4007e0: ff 25 4a 18 20 00 jmpq   *0x20184a(%rip)      # 602030 <_ZNSt8ios_base4InitC1Ev@GLIBCXX_3.4>
4007e6: 68 03 00 00 00   pushq  $0x3
4007eb: e9 b0 ff ff ff   jmpq   4007a0 <_.plt>
```

```
0000000004007f0 <malloc@plt>:
4007f0: ff 25 42 18 20 00 jmpq   *0x201842(%rip)      # 602038 <malloc@GLIBC_2.2.5>
4007f6: 68 04 00 00 00   pushq  $0x4
4007fb: e9 a0 ff ff ff   jmpq   4007a0 <_.plt>
```

```
000000000400800 <__libc_start_main@plt>:
400800: ff 25 3a 18 20 00 jmpq   *0x20183a(%rip)      # 602040 <__libc_start_main@GLIBC_2.2.5>
```

File Edit View Search Terminal Help

000000000400b1e <main>:

```

400b1e: 55          push    %rbp
400b1f: 48 89 e5    mov     %rsp,%rbp
400b22: 48 83 ec 20 sub     $0x20,%rsp
400b26: c7 45 f4 00 00 00 00 movl    $0x0,-0xc(%rbp)
400b2d: c7 45 f0 00 00 00 00 movl    $0x0,-0x10(%rbp)
400b34: c7 45 ec 00 00 00 00 movl    $0x0,-0x14(%rbp)
400b3b: be d0 0c 40 00 mov     $0x400cd0,%esi
400b40: bf a0 21 60 00 mov     $0x6021a0,%edi
400b45: e8 e6 fc ff ff callq   400830 <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@plt>
400b4a: 48 8d 45 f4 lea     -0xc(%rbp),%rax
400b4e: 48 89 c6    mov     %rax,%rsi
400b51: bf 80 20 60 00 mov     $0x602080,%edi
400b56: e8 f5 fc ff ff callq   400850 <_ZNSirsERI@plt>
400b5b: 48 8d 55 f0 lea     -0x10(%rbp),%rdx
400b5f: 48 89 d6    mov     %rdx,%rsi
400b62: 48 89 c7    mov     %rax,%rdi
400b65: e8 e6 fc ff ff callq   400850 <_ZNSirsERI@plt>
400b6a: 48 8d 55 ec lea     -0x14(%rbp),%rdx
400b6e: 48 89 d6    mov     %rdx,%rsi
400b71: 48 89 c7    mov     %rax,%rdi
400b74: e8 d7 fc ff ff callq   400850 <_ZNSirsERI@plt>
400b79: 8b 55 f4    mov     -0xc(%rbp),%edx
400b7c: 8b 45 f0    mov     -0x10(%rbp),%eax
400b7f: 01 c2      add     %eax,%edx
400b81: 8b 45 ec    mov     -0x14(%rbp),%eax
400b84: 01 d0      add     %edx,%eax
400b86: 89 c7      mov     %eax,%edi
400b88: e8 c0 fd ff ff callq   40094d <_Z3geni>
400b8d: 48 89 45 f8 mov     %rax,-0x8(%rbp)
400b91: 8b 55 f4    mov     -0xc(%rbp),%edx
400b94: 8b 45 f0    mov     -0x10(%rbp),%eax
400b97: 01 c2      add     %eax,%edx
400b99: 8b 45 ec    mov     -0x14(%rbp),%eax
400b9c: 01 d0      add     %edx,%eax
400b9e: 3d 39 05 00 00 cmp     $0x539,%eax
400ba3: 75 27      jne     400bcc <main+0xae>
400ba5: be e6 0c 40 00 mov     $0x400ce6,%esi
400baa: bf a0 21 60 00 mov     $0x6021a0,%edi
400baf: e8 7c fc ff ff callq   400830 <_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@plt>
400bb4: 48 8b 45 f8 mov     -0x8(%rbp),%rax
400bb8: 48 89 c7    mov     %rax,%rdi

```

解題過程

I have the following src:

```
1 #include<stdio.h>
2
3 int main(void) {
4     int i= 1337; // breakpoint after this value is assigned
5     return 0;
6 }
```

In the asm from `gdb` I get:

```
ksu@KSU-Ubuntu-1804-64:~/Downloads$ ./adder
Enter three numbers!
1337
0
0
easyc tf{y0u_added_thr33_nums!}
ksu@KSU-Ubuntu-1804-64:~/Downloads$
```

```
!0x0000000004004f1 main+4 movl $0x539,-0x4(%rbp)
```

And I verified that `$0x539 = 1337` How can I see the memory address where the value `1337` is stored? The value of the `rbp` memory address shows:

```
rbp 0x00007fffffffeb20
```

解題成功

EasyCTF_adder ✓

25

Ghidra

Ghidra實戰主題

該軟件被寫入 [的Java](#) 使用 [搖擺框架](#) 的 [GUI](#)。這 [反編譯器](#) 組件是用 [C ++](#) 編寫的。Ghidra 插件可以用 Java 或 Python (通過 [Jython](#) 提供) 開發

支持的架構

- [x86 16, 32 and 64 bit](#)
- [ARM and AARCH64](#)
- [PowerPC 32/64 and VLE](#)
- [MIPS 16/32/64](#)
- [MicroMIPS](#)
- [68xxx](#)
- [Java and DEX bytecode](#)
- [PA-RISC](#)
- [PIC 12/16/17/18/24](#)
- [SPARC 32/64](#)
- [CR16C](#)
- [Z80](#)
- [6502](#)
- [8051](#)
- [MSP430](#)
- [AVR8, AVR32](#)
- [SuperH](#)

Ghidra實戰主題

```
ksu@KSU-Ubuntu-1604-32: ~  
ksu@KSU-Ubuntu-1604-32: ~$ gedit run_ghidra.sh
```

```
*run_ghidra.sh (~/) - gedit  
Open ▾ [icon]  
Save
```

```
mkdir Ghidra  
cd Ghidra  
wget https://ghidra-sre.org/ghidra_9.0.1_PUBLIC_20190325.zip  
unzip ghidra_9.0.1_PUBLIC_20190325.zip  
cd ghidra_9.0.1  
sudo add-apt-repository ppa:openjdk-r/ppa  
sudo apt update  
sudo apt install openjdk-11-jdk  
sudo apt install openjdk-11-jre-headless  
chmod +x ghidraRun  
./ghidraRun|
```

Ghidra實戰主題

```
ksu@KSU-Ubuntu-1604-32 ~$ chmod +x run_ghidra.sh
ksu@KSU-Ubuntu-1604-32 ~$ ./run_ghidra.sh
--2021-03-13 14:56:09-- https://ghidra-sre.org/ghidra_9.0.1_PUBLIC_20190325.zip
Resolving ghidra-sre.org (ghidra-sre.org)... 13.227.73.28, 13.227.73.84, 13.227.73.9, ...
Connecting to ghidra-sre.org (ghidra-sre.org)|13.227.73.28|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 296076039 (282M) [application/zip]
Saving to: 'ghidra_9.0.1_PUBLIC_20190325.zip'

ghidra_9.0.1_PUBLIC 100%[=====>] 282.36M  2.73MB/s   in 64s

2021-03-13 14:57:14 (4.40 MB/s) - 'ghidra_9.0.1_PUBLIC_20190325.zip' saved [296076039/296076039]

Archive:  ghidra_9.0.1_PUBLIC_20190325.zip
  creating: ghidra_9.0.1/
  inflating: ghidra_9.0.1/ghidraRun.bat
  inflating: ghidra_9.0.1/ghidraRun
   creating: ghidra_9.0.1/licenses/
  inflating: ghidra_9.0.1/licenses/Modified_Nuvola_Icons_-_LGPL_2.1.txt
  inflating: ghidra_9.0.1/licenses/Tango_Icons_-_Public_Domain.txt
  inflating: ghidra_9.0.1/licenses/Nuvola_Icons_-_LGPL_2.1.txt
  inflating: ghidra_9.0.1/licenses/Oxygen_Icons_-_LGPL_3.0.txt
  inflating: ghidra_9.0.1/licenses/LGPL_2.1.txt
  inflating: ghidra_9.0.1/licenses/GPL_2_With_Classpath_Exception.txt
  inflating: ghidra_9.0.1/licenses/Public_Domain.txt
  inflating: ghidra_9.0.1/licenses/Jython_License.txt
  inflating: ghidra_9.0.1/licenses/JDOM_License.txt
  inflating: ghidra_9.0.1/licenses/Apache_License_2.0.txt
  inflating: ghidra_9.0.1/licenses/FAMFAMFAM_MINI_ICONS_-_Public_Domain.txt
  inflating: ghidra_9.0.1/licenses/MIT.txt
  inflating: ghidra_9.0.1/licenses/BSD.txt
  inflating: ghidra_9.0.1/licenses/Christian_Plattner.txt
  inflating: ghidra_9.0.1/licenses/Crystal_Clear_Icons_-_LGPL_2.1.txt
  inflating: ghidra_9.0.1/licenses/Creative_Commons_Attribution_2.5.html
  inflating: ghidra_9.0.1/licenses/FAMFAMFAM_Icons_-_CC_2.5.txt
  inflating: ghidra_9.0.1/licenses/LGPL_3.0.html
```

Ghidra實戰主題

Ghidra User Agreement

Licensed under the Apache License, Version 2.0 (the "License"); Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

As a software reverse engineering (SRE) framework, Ghidra is designed solely to facilitate lawful SRE activities. You should always ensure that any SRE activities in which you engage are permissible as computer software may be protected under governing law (e.g., copyright) or under an applicable licensing agreement. In making Ghidra available for public use, the National Security Agency does not condone or encourage any improper usage of Ghidra. Consistent with the Apache 2.0 license under which Ghidra has been made available, you are solely responsible for determining the appropriateness of using or redistributing Ghidra.

I Agree

I Don't Agree

```
Building dependency tree  
Reading state information... Done  
openjdk-11-jre-headless is already the newest version (11.0.11+4-0ubuntu3~16.04~  
1).  
openjdk-11-jre-headless set to manually installed.  
The following package was automatically installed and is no longer required:  
  snapd-login-service  
Use 'sudo apt autoremove' to remove it.  
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.  
ksu@KSU-Ubuntu-1604-32:~$
```

Welcome To Ghidra



GHIDRA

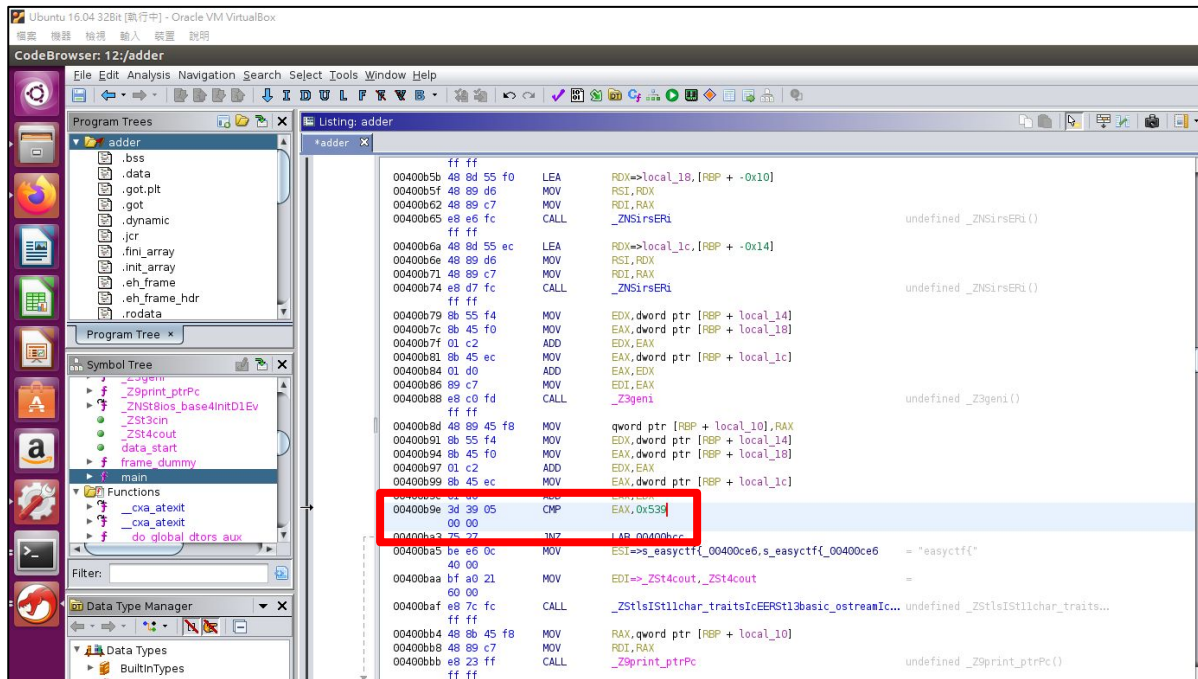
Version 9.0.1
Build PUBLIC
2019-Mar-25 1752 EDT
Java Version [11.0.11-ea](#)

Licensed under the Apache License, Version 2.0 (the "License"); Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This program also includes third party components which have licenses other than Apache 2.0. See the LICENSE.txt file for details.

Scanning jar: DATA.jar

Ghidra實戰主題



Ghidra實戰主題