Reverse engineering cyberchallenge.it 2025

What's reverse engineering?

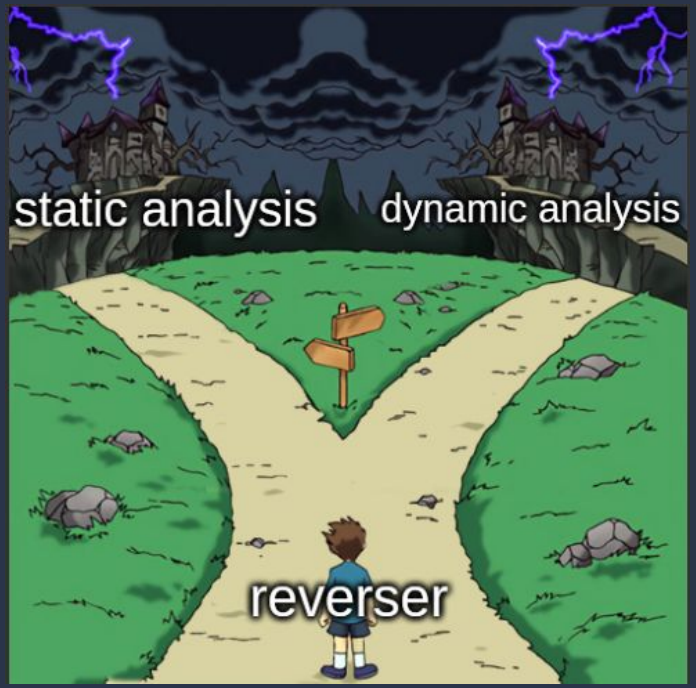
Most of the codes out there are closed source.

The aim of reversing is to understand what is happening under the hood of a software as deep as possible.

Why should I reverse?

- Malware analysis
- Exploit analysis
- Game Cheating & mod dev
- Find vulns
- …
- Ctf points

2 ways of reversing:



Static analysis

Pros:

- safety
- no specific hardware/emulator required
- cover all execution path
- bypass anti debug tricks

Cons:

- no clue what data the program is using
- code can be heavily obfuscated

Dynamic analysis

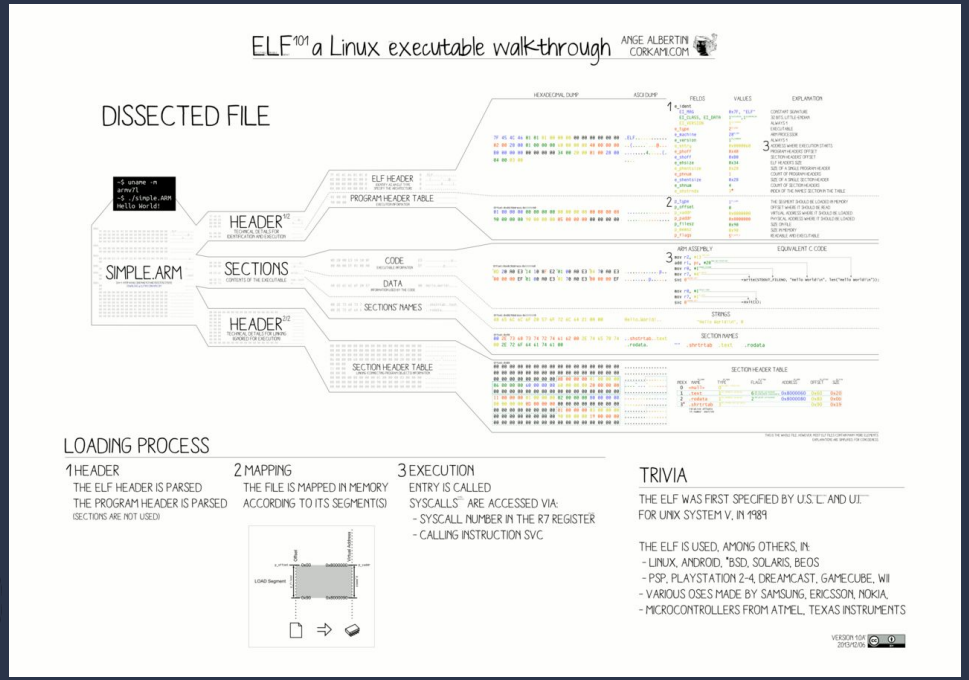
Pros:

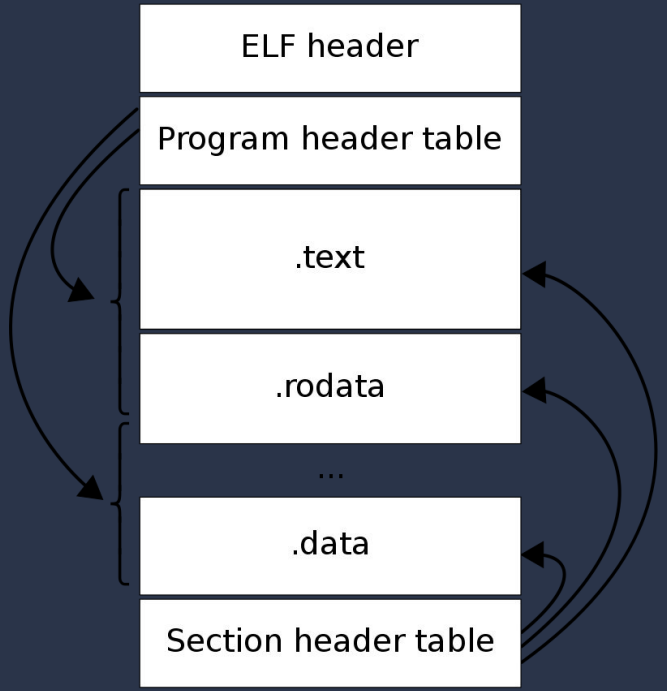
- you actually see which data the program is using
- could be faster

Cons:

- anti debug techniques exists
- you could need libraries or emulators to run the program

ELF: executable and linkable format





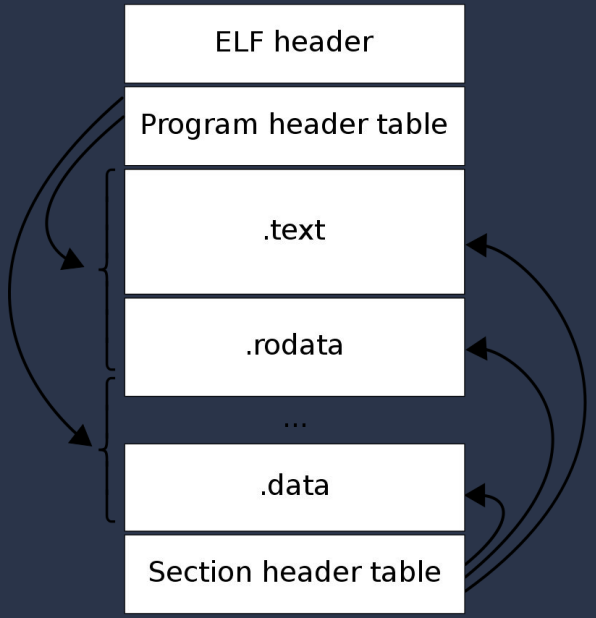
ELF - sections and symbols

- .text : contains executable code and is generally read-only and fixed size
- .data : global variables and local static variables which have a defined value and can be modified
- .bss : global variables and local static variables that are initialized to zero or do not have explicit initialization
- .rodata : used for global read-only data
- ...

How do I get specific informations?

$ readelf -header <file>

$ readelf -s <file>



Readelf

```c
#include <stdio.h>
int globale = 152;
void do_none(){
    return;
}
int main(){
    int a = 0;
    return 1;
}
```

```
❯ readelf -header test
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x1040
  Start of program headers:          64 (bytes into file)
  Start of section headers:          14840 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         13
  Size of section headers:           64 (bytes)
  Number of section headers:         35
  Section header string table index: 34
```

```
Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .interp           PROGBITS         0000000000000318  00000318
       000000000000001c  0000000000000000   A       0     0     1
  [ 2] .note.gnu.pr[...] NOTE             0000000000000338  00000338
       0000000000000030  0000000000000000   A       0     0     8
  [ 3] .note.gnu.bu[...] NOTE             0000000000000368  00000368
       0000000000000024  0000000000000000   A       0     0     4
  [ 4] .note.ABI-tag     NOTE             000000000000038c  0000038c
       0000000000000020  0000000000000000   A       0     0     4
  [ 5] .gnu.hash         GNU_HASH         00000000000003b0  000003b0
       0000000000000024  0000000000000000   A       6     0     8
  [ 6] .dynsym           DYNSYM           00000000000003d8  000003d8
       0000000000000090  0000000000000018   A       7     1     8
  [ 7] .dynstr           STRTAB           0000000000000468  00000468
       0000000000000088  0000000000000000   A       0     0     1
  [ 8] .gnu.version      VERSYM           00000000000004f0  000004f0
       000000000000000c  0000000000000002   A       6     0     2
  [ 9] .gnu.version_r    VERNEED          0000000000000500  00000500
       0000000000000030  0000000000000000   A       7     1     8
  [10] .rela.dyn         RELA             0000000000000530  00000530
       00000000000000c0  0000000000000018   A       6     0     8
  [11] .init             PROGBITS         0000000000001000  00001000
       000000000000001b  0000000000000000  AX       0     0     4
  [12] .plt              PROGBITS         0000000000001020  00001020
       0000000000000010  0000000000000010  AX       0     0     16
  [13] .plt.got          PROGBITS         0000000000001030  00001030
       0000000000000010  0000000000000010  AX       0     0     16
  [14] .text             PROGBITS         0000000000001040  00001040
       0000000000000117  0000000000000000  AX       0     0     16
  [15] .fini             PROGBITS         0000000000001158  00001158
       000000000000000d  0000000000000000  AX       0     0     4
  [16] .rodata           PROGBITS         0000000000002000  00002000
       0000000000000004  0000000000000004  AM       0     0     4
  [17] .eh_frame_hdr     PROGBITS         0000000000002004  00002004
       0000000000000034  0000000000000000   A       0     0     4
  [18] .eh_frame         PROGBITS         0000000000002038  00002038
       00000000000000b4  0000000000000000   A       0     0     8
  [19] .init_array       INIT_ARRAY       0000000000003df0  00002df0
       0000000000000008  0000000000000008  WA       0     0     8
  [20] .fini_array       FINI_ARRAY       0000000000003df8  00002df8
       0000000000000008  0000000000000008  WA       0     0     8
  [21] .dynamic          DYNAMIC          0000000000003e00  00002e00
       00000000000001c0  0000000000000010  WA       7     0     8
  [22] .got              PROGBITS         0000000000003fc0  00002fc0
       0000000000000040  0000000000000008  WA       0     0     8
  [23] .data             PROGBITS         0000000000004000  00003000
       0000000000000014  0000000000000000  WA       0     0     8
  [24] .bss              NOBITS           0000000000004014  00003014
       0000000000000004  0000000000000000  WA       0     0     1
  [25] .comment          PROGBITS         0000000000000000  00003014
       000000000000002b  0000000000000001  MS       0     0     1
  [26] .debug_aranges    PROGBITS         0000000000000000  0000303f
       0000000000000030  0000000000000000           0     0     1
  [27] .debug_info       PROGBITS         0000000000000000  0000306f
       00000000000000d7  0000000000000000           0     0     1
  [28] .debug_abbrev     PROGBITS         0000000000000000  00003146
       0000000000000082  0000000000000000           0     0     1
  [29] .debug_line       PROGBITS         0000000000000000  000031c8
       000000000000005f  0000000000000000           0     0     1
  [30] .debug_str        PROGBITS         0000000000000000  00003227
       00000000000000e9  0000000000000001  MS       0     0     1
  [31] .debug_line_str   PROGBITS         0000000000000000  00003310
       0000000000000037  0000000000000001  MS       0     0     1
  [32] .symtab           SYMTAB           0000000000000000  00003348
       0000000000000378  0000000000000018          33    18     8
  [33] .strtab           STRTAB           0000000000000000  000036c0
```

Strip binaries

$ strip <file>

This removes .symtab and debug sections entirely.

Stripping binaries makes them much lighter


```
❯ readelf -s test

Symbol table '.dynsym' contains 6 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND _[...]@GLIBC_2.34 (2)
     2: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterT[...]
     3: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     4: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_registerTMC[...]
     5: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND [...]@GLIBC_2.2.5 (3)

Symbol table '.symtab' contains 37 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS Scrt1.o
     2: 000000000000038c    32 OBJECT  LOCAL  DEFAULT    4 __abi_tag
     3: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
     4: 0000000000001070     0 FUNC    LOCAL  DEFAULT   14 deregister_tm_clones
     5: 00000000000010a0     0 FUNC    LOCAL  DEFAULT   14 register_tm_clones
     6: 00000000000010e0     0 FUNC    LOCAL  DEFAULT   14 __do_global_dtors_aux
     7: 0000000000004014     1 OBJECT  LOCAL  DEFAULT   24 completed.0
     8: 0000000000003df8     0 OBJECT  LOCAL  DEFAULT   20 __do_global_dtor[...]
     9: 0000000000001120     0 FUNC    LOCAL  DEFAULT   14 frame_dummy
    10: 0000000000003df0     0 OBJECT  LOCAL  DEFAULT   19 __frame_dummy_in[...]
    11: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS test.c
    12: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
    13: 00000000000020e8     0 OBJECT  LOCAL  DEFAULT   18 __FRAME_END__
    14: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS
    15: 0000000000003e00     0 OBJECT  LOCAL  DEFAULT   21 _DYNAMIC
    16: 0000000000002004     0 NOTYPE  LOCAL  DEFAULT   17 __GNU_EH_FRAME_HDR
    17: 0000000000003fc0     0 OBJECT  LOCAL  DEFAULT   22 _GLOBAL_OFFSET_TABLE_
    18: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_mai[...]
    19: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterT[...]
    20: 0000000000004000     0 NOTYPE  WEAK   DEFAULT   23 data_start
    21: 0000000000004014     0 NOTYPE  GLOBAL DEFAULT   23 _edata
    22: 0000000000001158     0 FUNC    GLOBAL HIDDEN    15 _fini
    23: 0000000000004000     0 NOTYPE  GLOBAL DEFAULT   23 __data_start
    24: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
    25: 0000000000004008     0 OBJECT  GLOBAL HIDDEN    23 __dso_handle
    26: 0000000000004010     4 OBJECT  GLOBAL DEFAULT   23 globale
    27: 0000000000002000     4 OBJECT  GLOBAL DEFAULT   16 _IO_stdin_used
    28: 0000000000004018     0 NOTYPE  GLOBAL DEFAULT   24 _end
    29: 0000000000001040    38 FUNC    GLOBAL DEFAULT   14 _start
    30: 0000000000004014     0 NOTYPE  GLOBAL DEFAULT   24 __bss_start
    31: 0000000000001134    35 FUNC    GLOBAL DEFAULT   14 main
    32: 0000000000001129    11 FUNC    GLOBAL DEFAULT   14 do_none
    33: 0000000000004018     0 OBJECT  GLOBAL HIDDEN    23 __TMC_END__
    34: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_registerTMC[...]
    35: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND __cxa_finalize@G[...]
    36: 0000000000001000     0 FUNC    GLOBAL HIDDEN    11 _init
```

Strings

test_2.c

```c
#include <stdio.h>
void main(){
  puts("fibonhack");
  return;
}
```

```
❯ strings test_2
/lib64/ld-linux-x86-64.so.2
__cxa_finalize
__libc_start_main
puts
libc.so.6
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
fibonhack
:*3$"
GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Scrt1.o
__abi_tag
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
test_2.c
__FRAME_END__
_DYNAMIC
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
__libc_start_main@GLIBC_2.34
_ITM_deregisterTMCloneTable
puts@GLIBC_2.2.5
_edata
_fini
__data_start
```

File

```
❯ file test_2.c
test_2.c: C source, ASCII text
❯ file test
test: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /
lib64/ld-linux-x86-64.so.2, BuildID[sha1]=d055a0c54bb646ff5397985f36d9dc8362dd4e84, for GNU/Linu
x 3.2.0, with debug_info, not stripped
```

Static analysis: tools

Preliminary analysis:

- strings
- nm
- readelf
- ldd
- file

Deep analysis:

- Got money?
  - IDA pro
  - Binary ninja
- Only cli?
  - radare2
- Open source?
  - Ghidra

ltrace & strace

ltrace: library call tracer

strace: trace system call and signal


```
❯ ltrace ./test_2
puts("fibonhack"fibonhack
)                                 = 10
+++ exited (status 10) +++
❯ strace ./test_2
execve("./test_2", ["./test_2"], 0x7ffe4da97110 /* 67 vars */) = 0
brk(NULL)                         = 0x55b2ab4ce000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd72294700) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7feb9b8d8000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=109679, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 109679, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7feb9b8bd000
close(3)                          = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7feb9b600000
mprotect(0x7feb9b628000, 2023424, PROT_NONE) = 0
mmap(0x7feb9b628000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
mmap(0x7feb9b7bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7feb9b7b
mmap(0x7feb9b816000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) =
mmap(0x7feb9b81c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7feb
close(3)                          = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7feb9b8ba000
arch_prctl(ARCH_SET_FS, 0x7feb9b8ba740) = 0
set_tid_address(0x7feb9b8baa10)         = 8763
set_robust_list(0x7feb9b8baa20, 24)     = 0
rseq(0x7feb9b8bb0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7feb9b816000, 16384, PROT_READ) = 0
mprotect(0x55b2a94dd000, 4096, PROT_READ) = 0
mprotect(0x7feb9b912000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7feb9b8bd000, 109679)          = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}, AT_EMPTY_PATH) = 0
getrandom("\x70\x2c\x3d\x6e\x9b\x63\xbf\x49", 8, GRND_NONBLOCK) = 8
brk(NULL)                         = 0x55b2ab4ce000
brk(0x55b2ab4ef000)               = 0x55b2ab4ef000
write(1, "fibonhack\n", 10fibonhack
)            = 10
exit_group(10)                    = ?
+++ exited with 10 +++
```

Dynamic analysis: tools

Preliminary analysis:

- ltrace
- strace

Deep analysis:

- gdb (gef or pwndbg)
- rr
- frida

Tips & tricks

- play around with the executable to get a grasp on what it's doing
- Use both static and dynamic analysis. e.g.: while doing static analysis make some guesses and try to verify/disproof that hypothesis
- When reversing a huge codebase, don't waste time on trying to understand everything. Focus on the things you think are important. Most of the time you can ignore lots of stuff and still get what's happening.
- *Fai roba a caso* - NickOve

Time for a live demo:

What are you gonna learn?

- basic use of ghidra
- install third parties extension/script
- add debug sections or symbols
- patch binaries
- basic use of gdb

keygenme

125 500

binary

@ danielepusceddu

Status: up, last checked: *5 minutes ago*

Registrare il nostro prodotto è facilissimo, basta inserire ID e chiave!

nc keygenme.challs.olicyber.it 10017

Hints

Hint 1 -0pt

Attachments

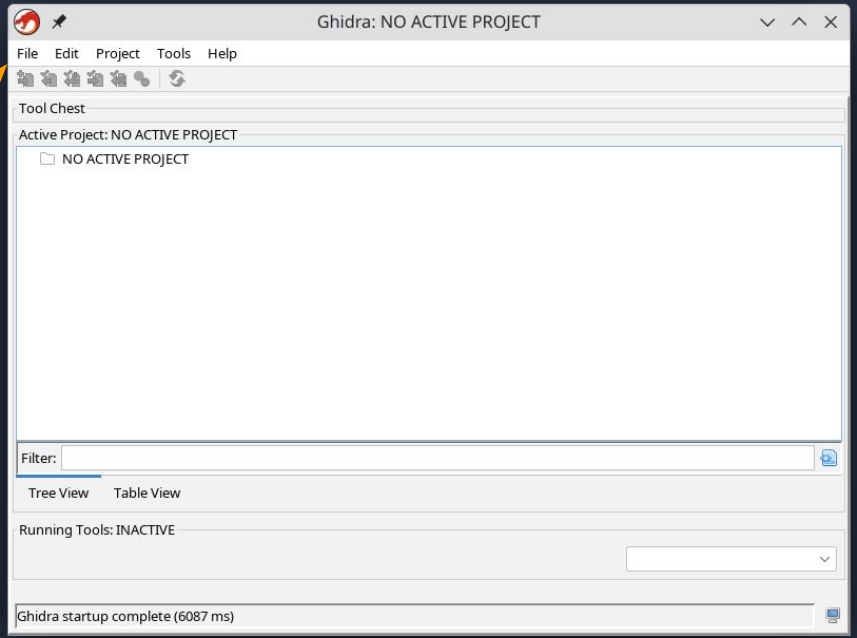
keygenme

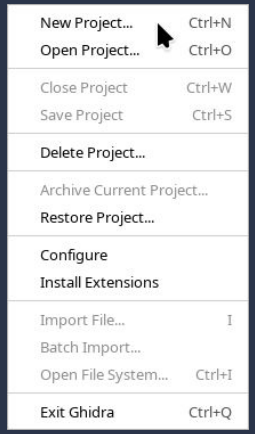
flag{...} Submit

Ghidra

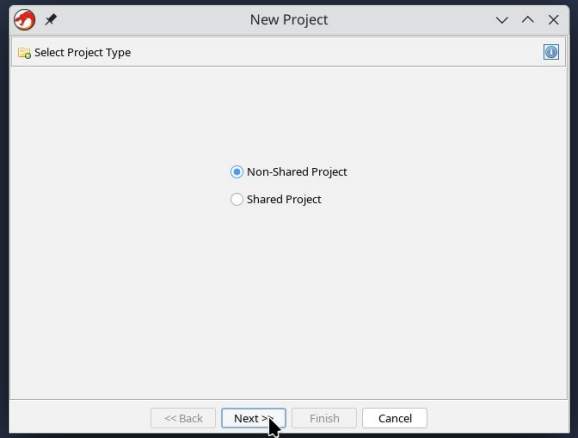
- install Java [>21]
- download Ghidra
- unzip and execute ghidraRun

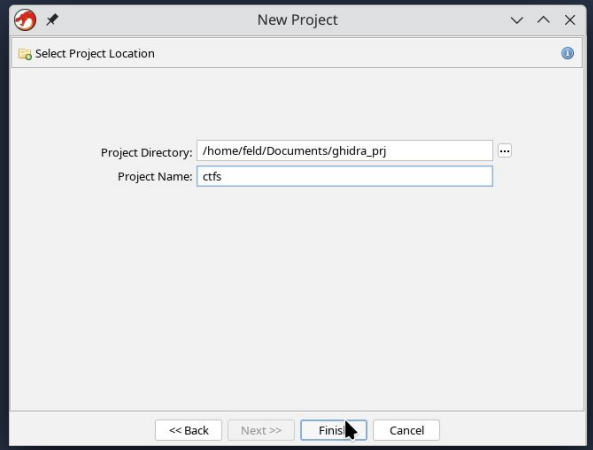
Create Project





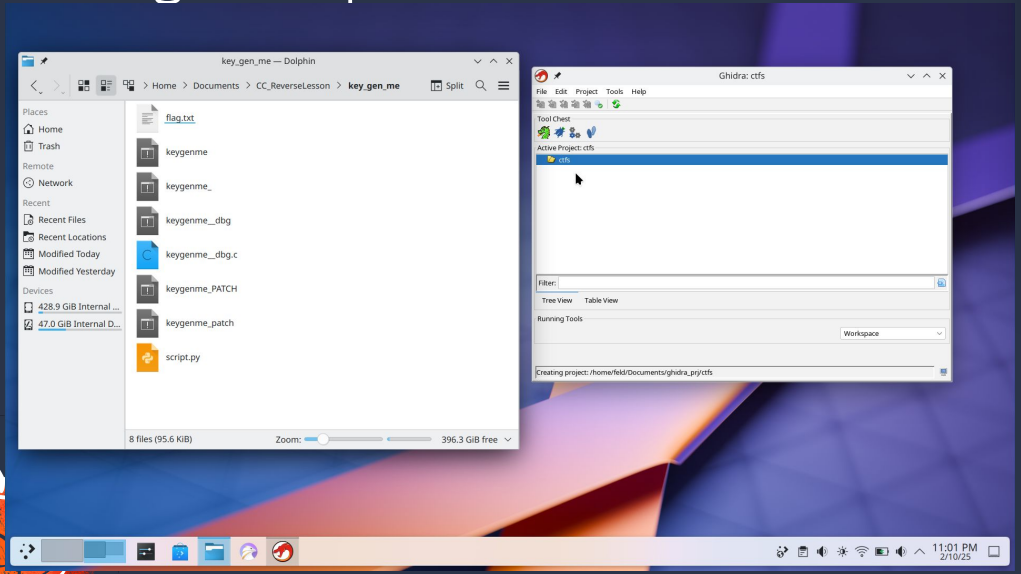
Create Project

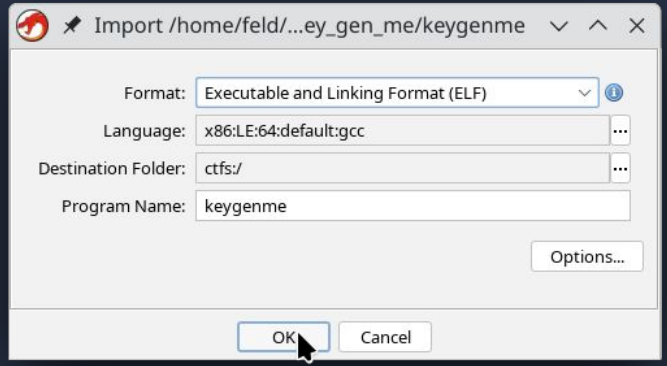


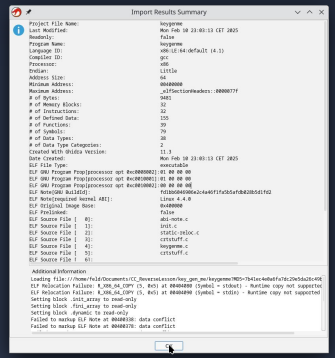


Add executable to project

drag & drop

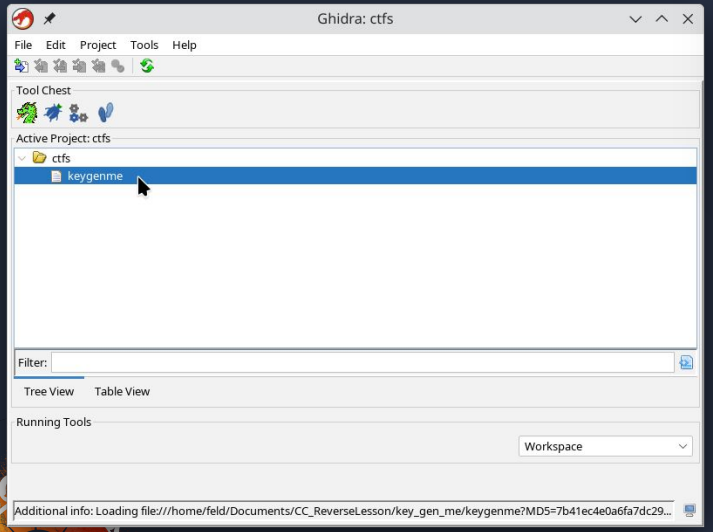


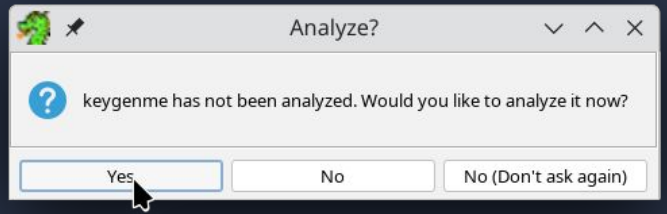


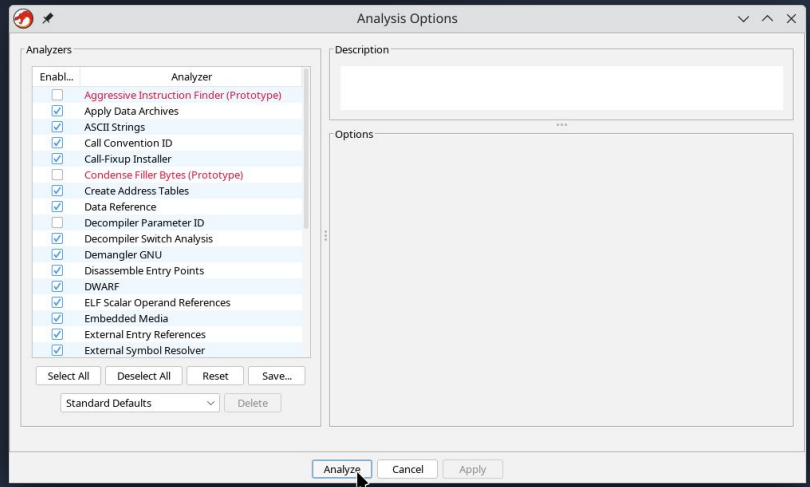


Analyze file

double click

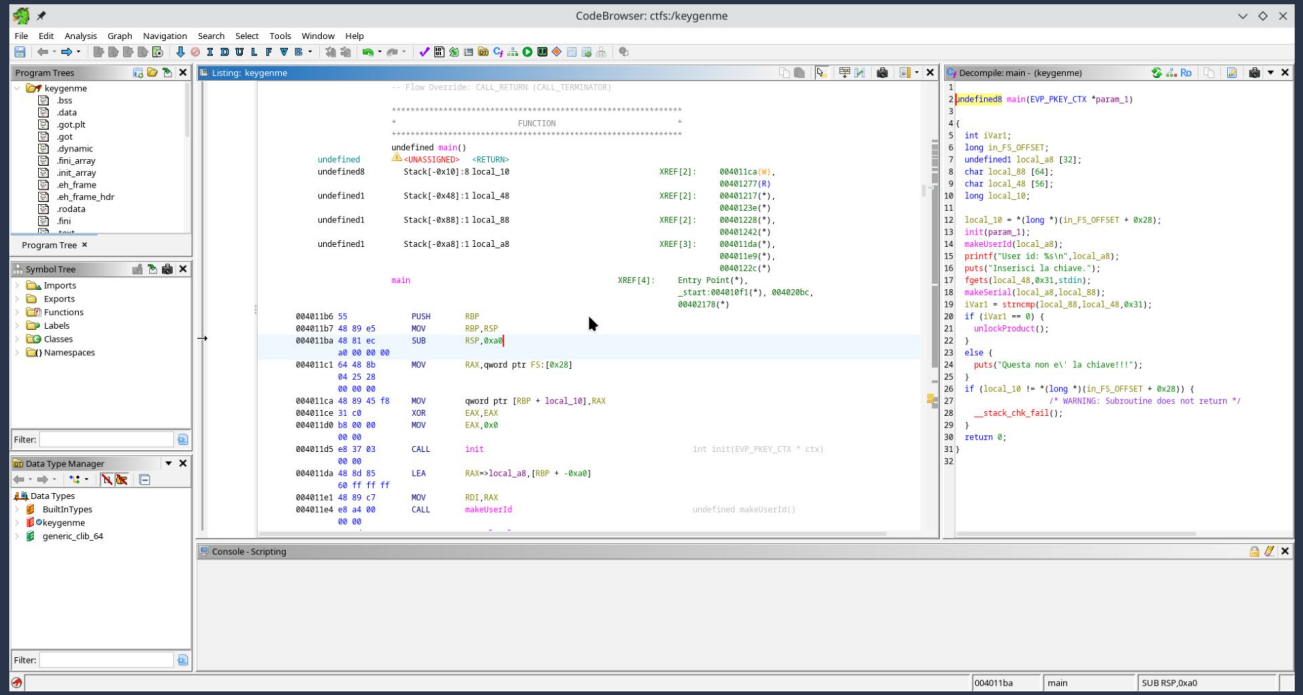




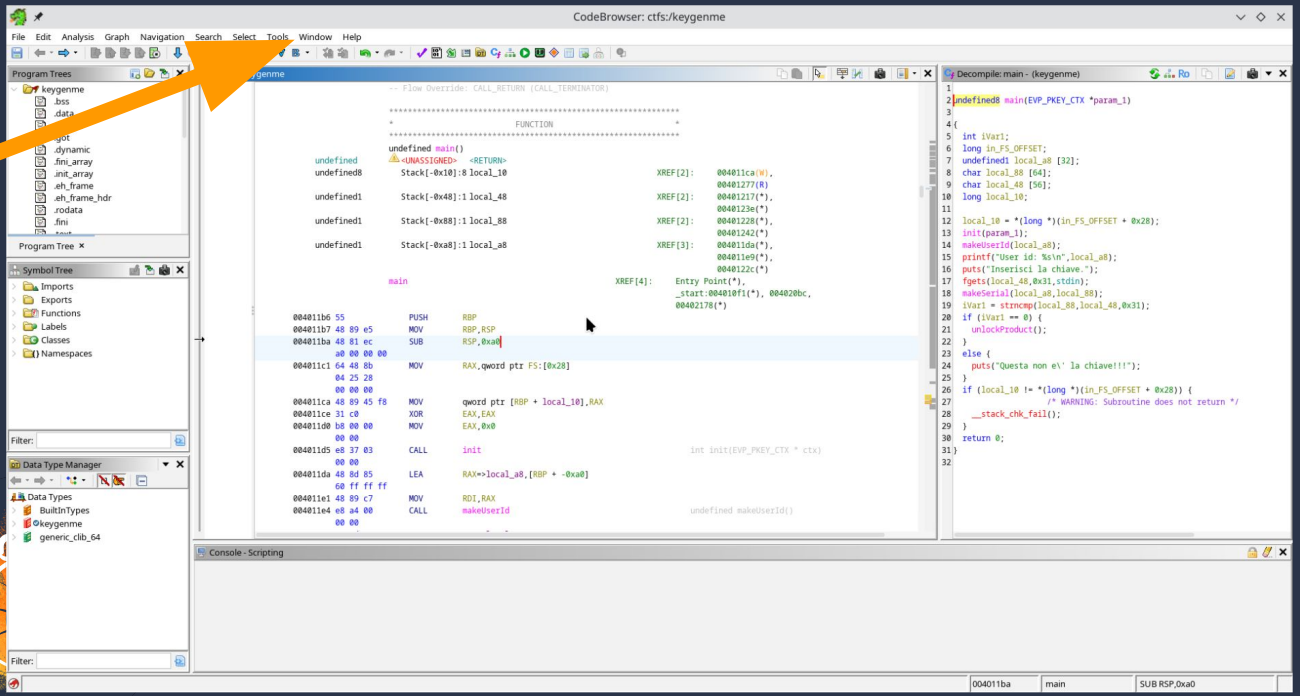


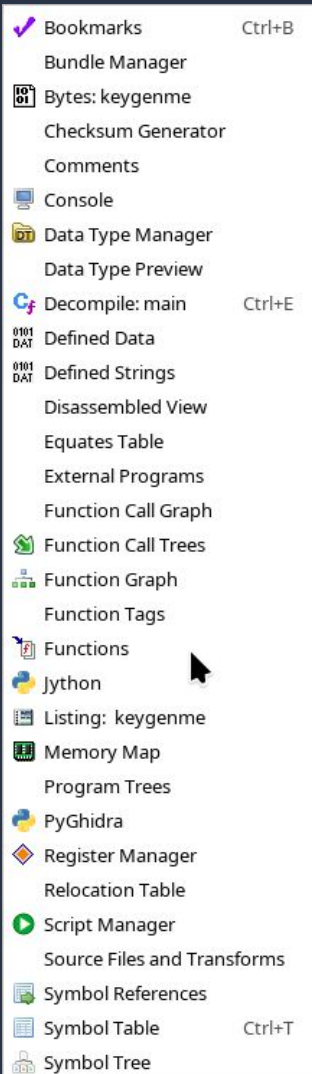
GUI

drag & drop windows in the interface



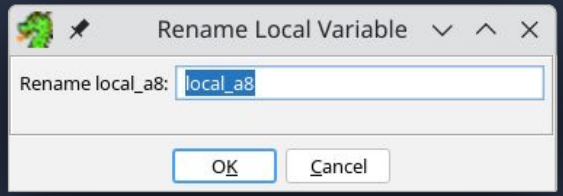
Add windows


CodeBrowser: ctfs:/keygenme
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
Symbol Tree
Data Type Manager
Console - Scripting
Decompile: main - (keygenme)


Bookmarks Ctrl+B
Bundle Manager
Bytes: keygenme
Checksum Generator
Comments
Console
Data Type Manager
Data Type Preview
Decompile: main Ctrl+E
Defined Data
Defined Strings
Disassembled View
Equates Table
External Programs
Function Call Graph
Function Call Trees
Function Graph
Function Tags
Functions
Jython
Listing: keygenme
Memory Map
Program Trees
PyGhidra
Register Manager
Relocation Table
Script Manager
Source Files and Transforms
Symbol References
Symbol Table Ctrl+T
Symbol Tree

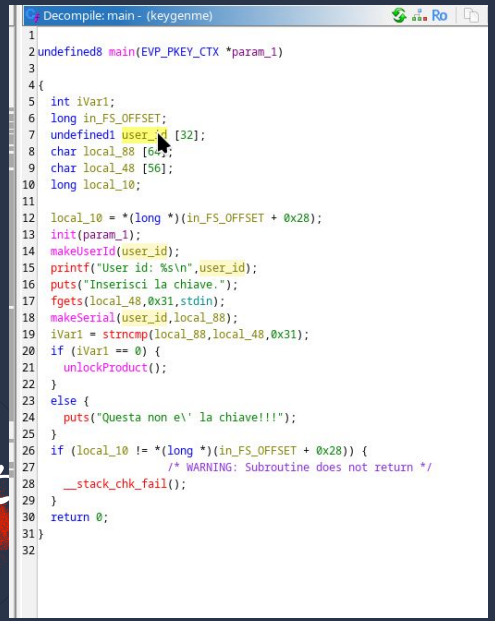
rename variable or functions

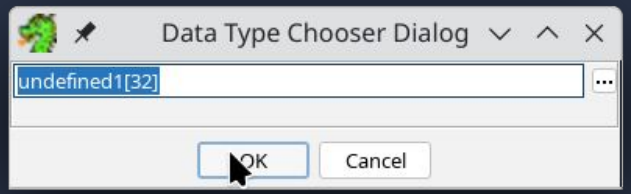
press L



change variable type

press CTRL+L





color legend

Decompile: main - (keygenme)

```
1
2 undefined8 main(EVP_PKEY_CTX *param_1)
3
4 {
5   int iVar1;
6   long in_FS_OFFSET;
7   undefined1 user_id [32];
8   char local_88 [64];
9   char local_48 [56];
10  long local_10;
11
12  local_10 = *(long *)(in_FS_OFFSET + 0x28);
13  init(param_1);
14  makeUserId(user_id);
15  printf("User id: %s\n",user_id);
16  puts("Inserisci la chiave.");
17  fgets(local_48,0x31,stdin);
18  makeSerial(user_id,local_88);
19  iVar1 = strncmp(local_88,local_48,0x31);
20  if (iVar1 == 0) {
21    unlockProduct();
22  }
23  else {
24    puts("Questa non e\' la chiave!!!");
25  }
26  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
27                    /* WARNING: Subroutine does not return */
28    __stack_chk_fail();
29  }
30  return 0;
31 }
32
```

patch instructions

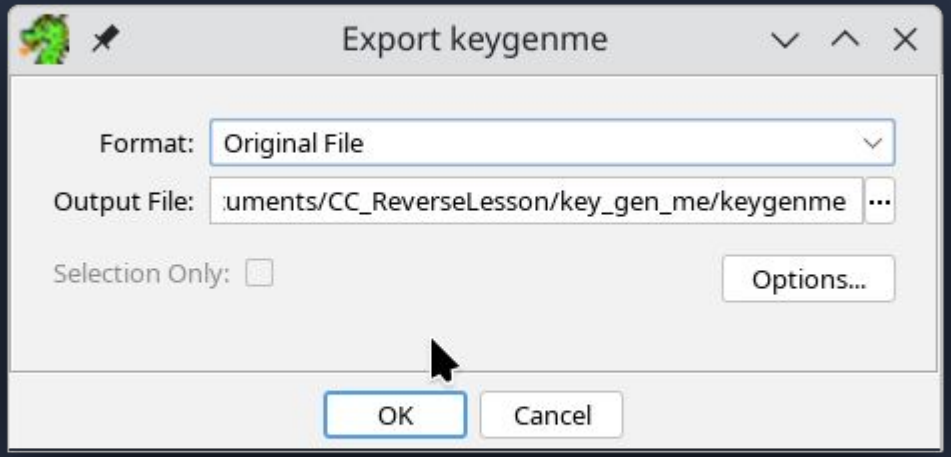
```
00401233 48 89 d6        MOV        RSI,RDX
00401236 48 89 c7        MOV        RDI,RAX
00401239 e8 d9 00        CALL       makeSerial
         00 00
0040123e 48 8d 4d c0     LEA        RCX=>local_48,[RBP + -0x40]
00401242 48 8d 45 80     LEA        RAX=>local_88,[RBP + -0x80]
00401246 ba 31 00        MOV        EDX,0x31
         00 00
0040124b 48 89 ce        MOV        RSI,RCX
0040124e 48 89 c7        MOV        RDI,RAX
00401251 e8 da fd        CALL       <EXTERNAL>::strncmp
         ff ff
00401256 85 c0           TEST       EAX,EAX
00401258 75 0c           JNZ        LAB_00401266
0040125a b8 00 00        MOV        EAX,0x0
         00 00
0040125f e8 21 02        CALL       unlockProduct
         00 00
00401264 eb 0c           JMP        LAB_00401272

                     LAB_00401266                                    XREF[1]:
00401266 48 8d 3d        LEA        RDI,[s_Questa_non_e'_la_chiave!!!_0040202a]
         bd 0d 00 00
0040126d e8 ce fd        CALL       <EXTERNAL>::puts
         ff ff

                     LAB_00401272                                    XREF[1]:
00401272 b8 00 00        MOV        EAX,0x0
         00 00
00401277 48 8b 4d f8     MOV        RCX,qword ptr [RBP + local_10]
0040127b 64 48 2b        SUB        RCX,qword ptr FS:[0x28]
```


```
00401233 48 89 d6        MOV        RSI,RDX
00401236 48 89 c7        MOV        RDI,RAX
00401239 e8 d9 00        CALL       makeSerial
         00 00
0040123e 48 8d 4d c0     LEA        RCX=>local_48,[RBP + -0x40]
00401242 48 8d 45 80     LEA        RAX=>local_88,[RBP + -0x80]
00401246 ba 31 00        MOV        EDX,0x31
         00 00
0040124b 48 89 ce        MOV        RSI,RCX
0040124e 48 89 c7        MOV        RDI,RAX
00401251 e8 da fd        CALL       <EXTERNAL>::strncmp
         ff ff
00401256 85 c0           TEST       EAX,EAX
00401258 75 0c           JNZ        0x00401266
0040125a b8 00 00        MOV        EAX,0x0
         00 00
0040125f e8 21 02        CALL       unlockProduct
         00 00
00401264 eb 0c           JMP        LAB_00401272

                     LAB_00401266                                    XREF[1]:
00401266 48 8d 3d        LEA        RDI,[s_Questa_non_e'_la_chiave!!!_0040202a]
         bd 0d 00 00
0040126d e8 ce fd        CALL       <EXTERNAL>::puts
         ff ff

                     LAB_00401272                                    XREF[1]:
00401272 b8 00 00        MOV        EAX,0x0
         00 00
00401277 48 8b 4d f8     MOV        RCX,qword ptr [RBP + local_10]
0040127b 64 48 2b        SUB        RCX,qword ptr FS:[0x28]
```

patch instructions

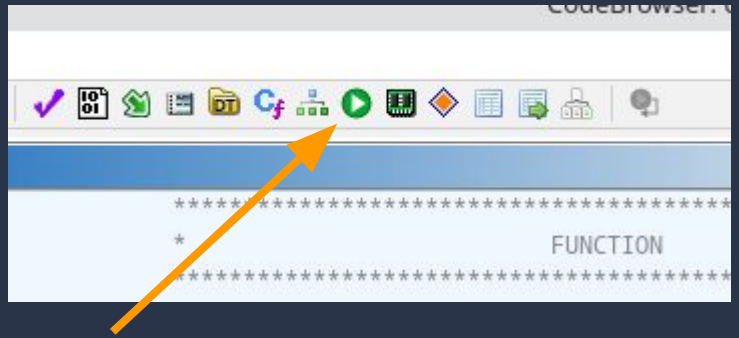
file -> export program

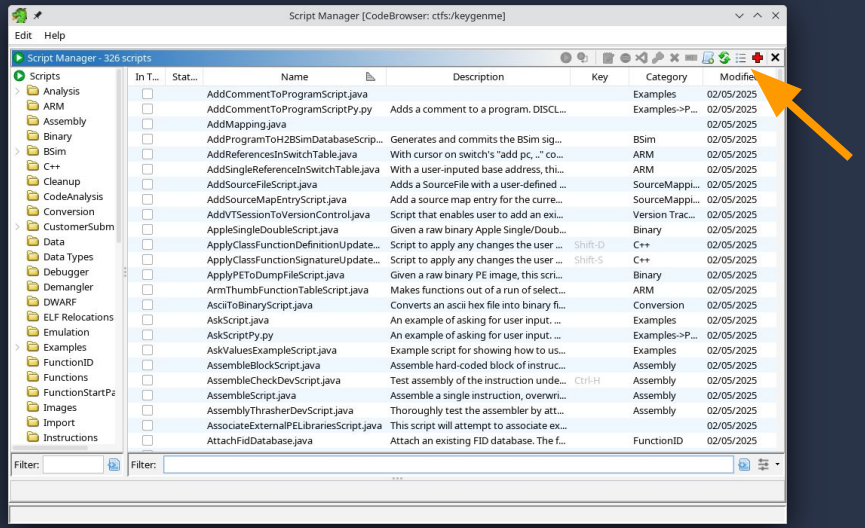
select format “Original File” and output file name



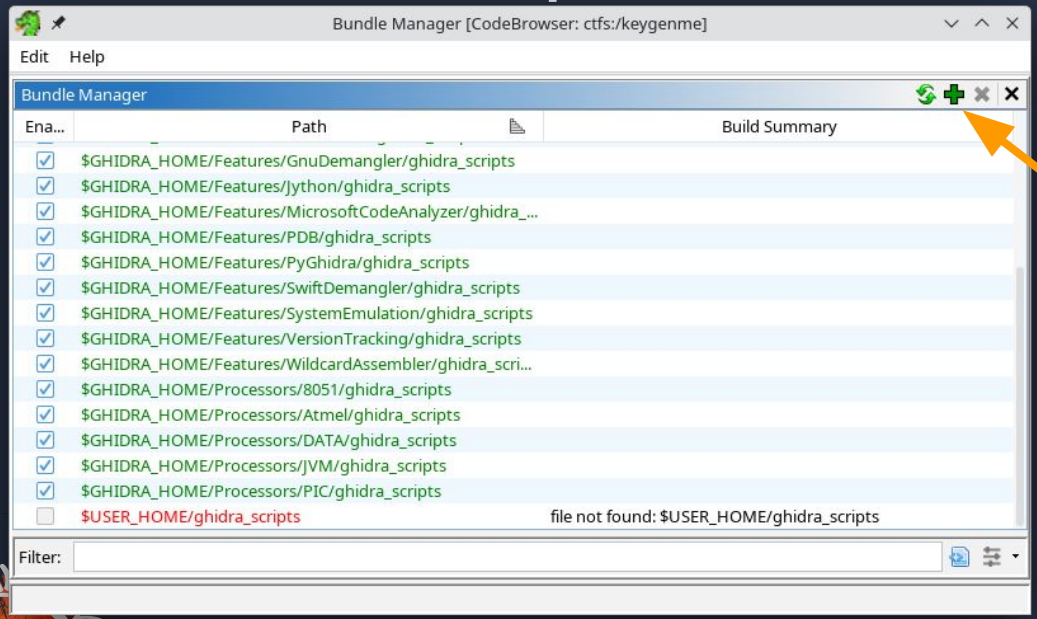
External Script

- download [syms2elf](#) or [ghidra2dwarf](#)



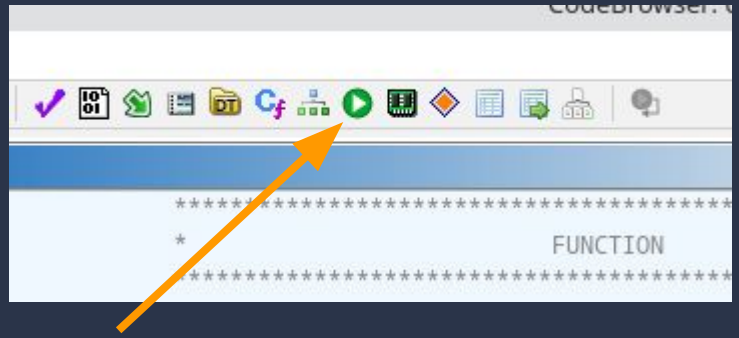


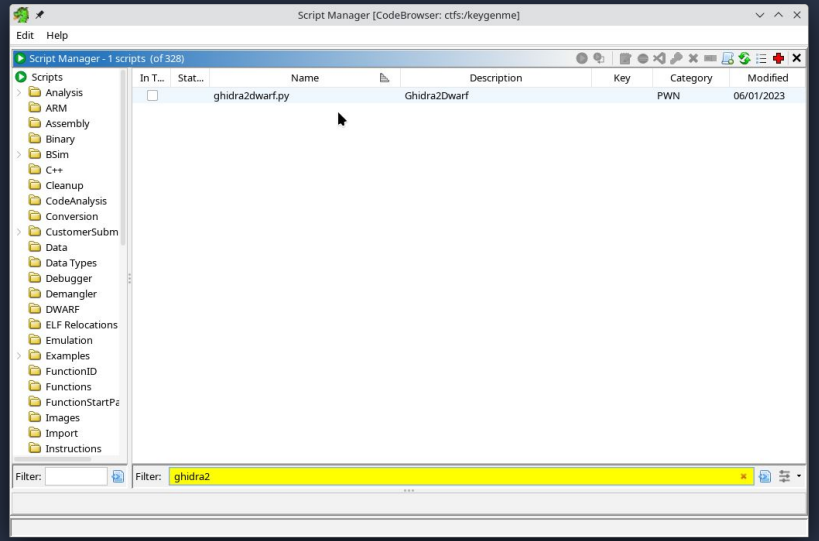
External Script



then choose the directory

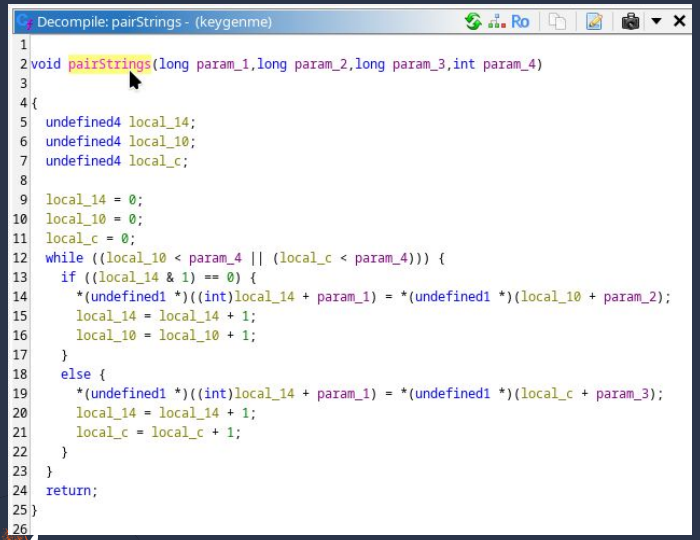
Run script

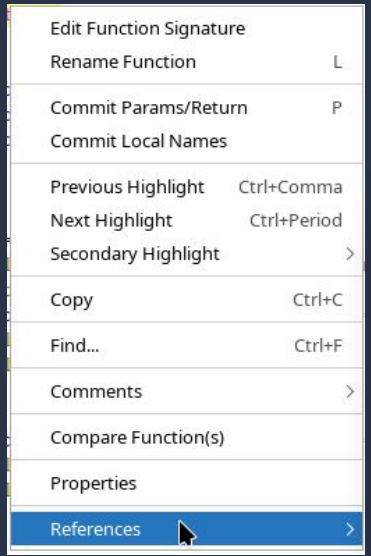


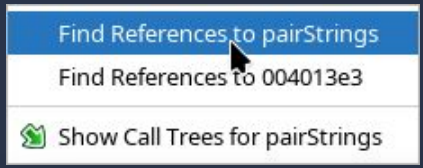


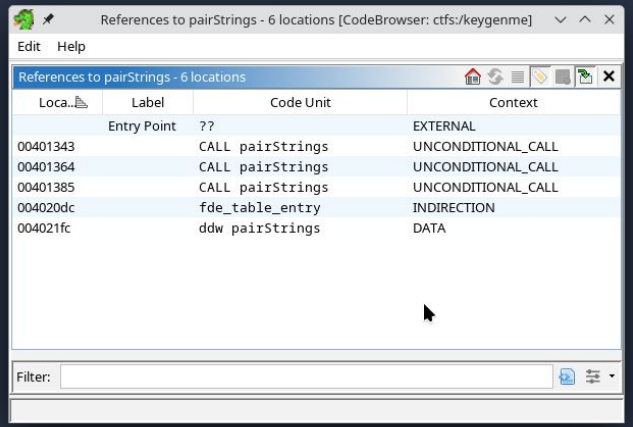
double click on script

Find reference to

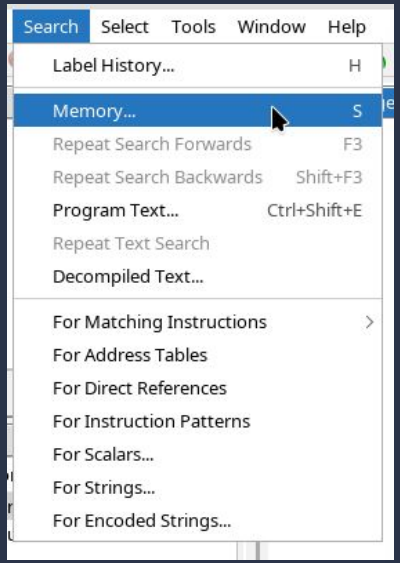


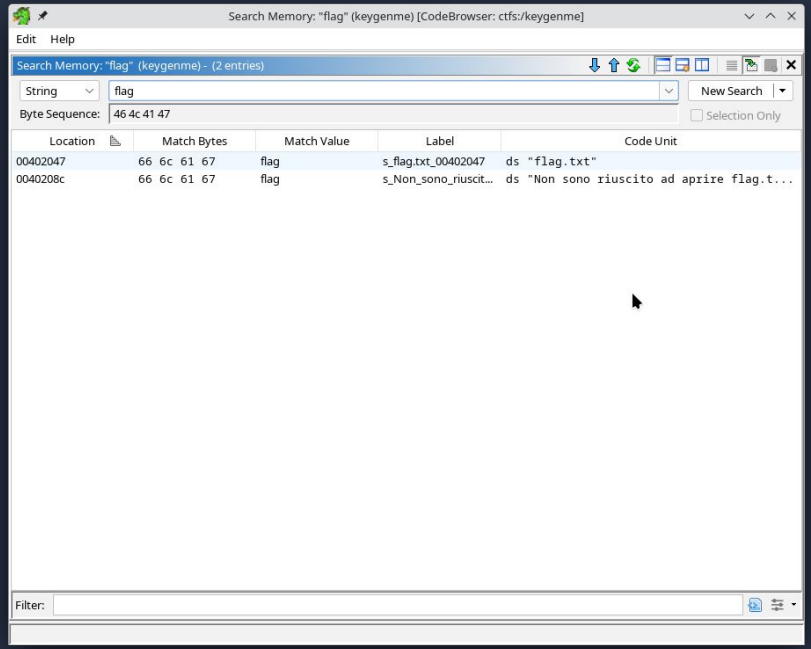




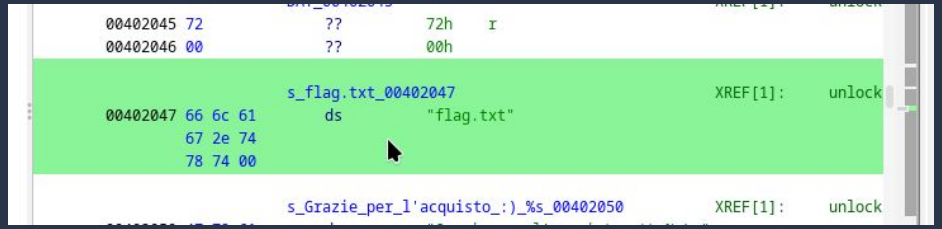


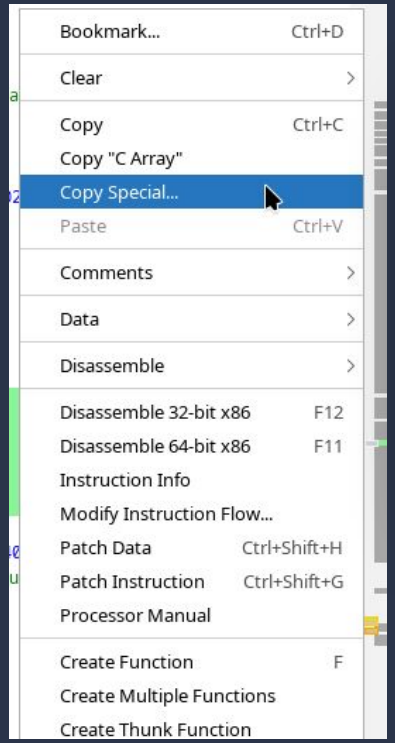
Search Memory

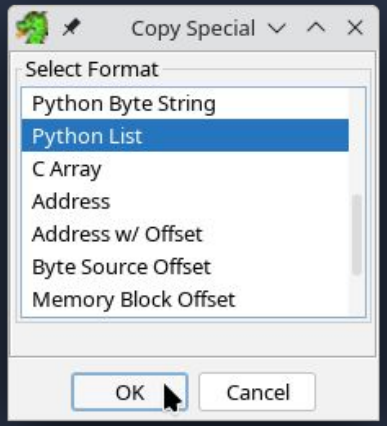




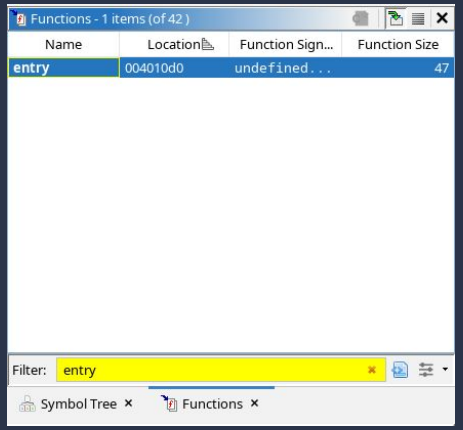
Copy Special

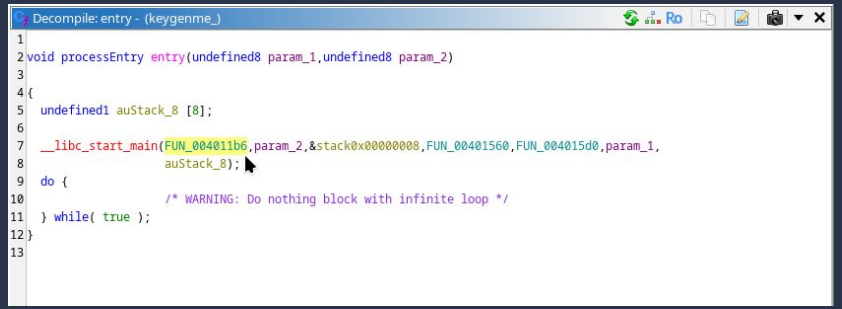






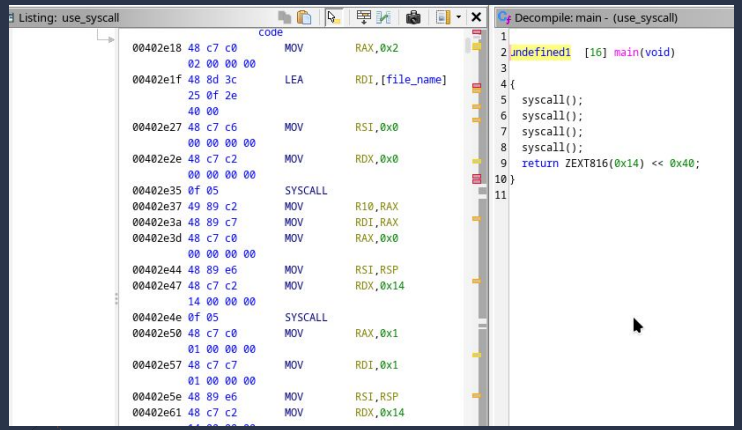
find Main in stripped elf

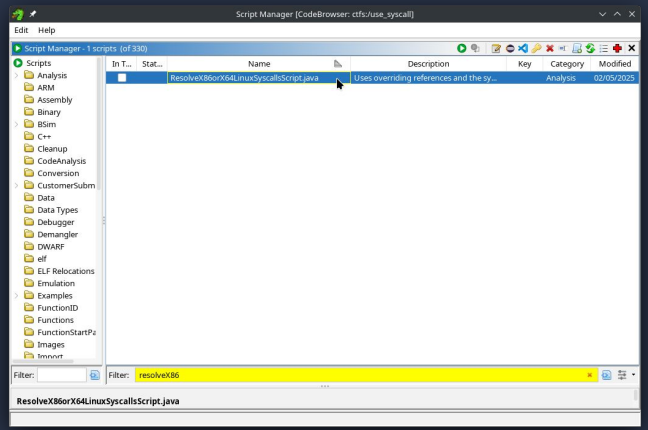


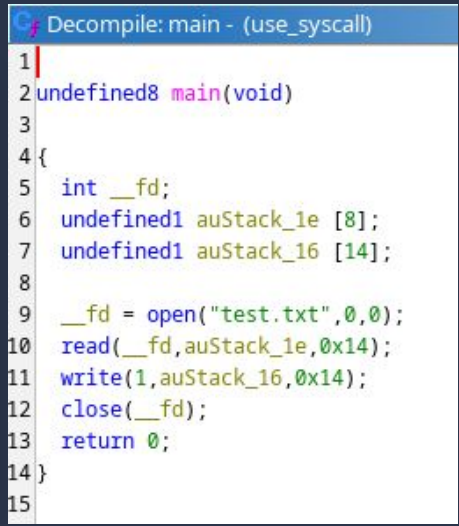


Resolve Linux Syscall

ResolveX86orX64LinuxSyscallsScript.java



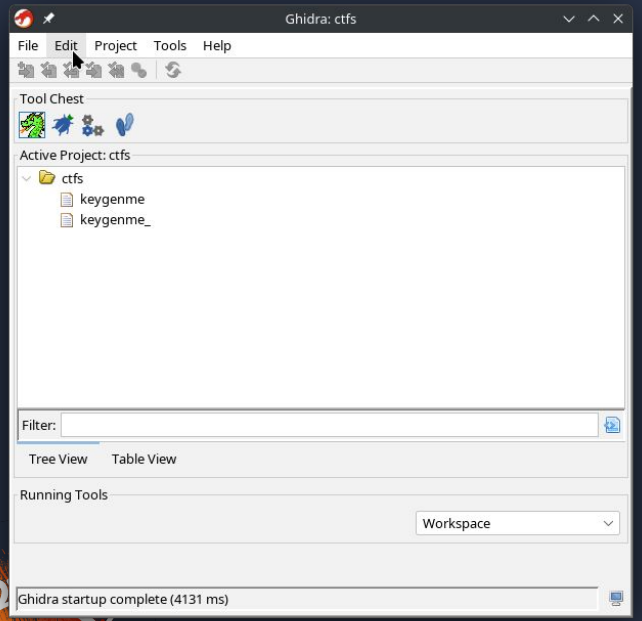


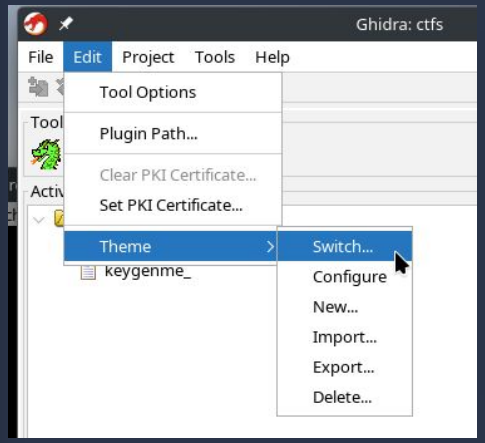


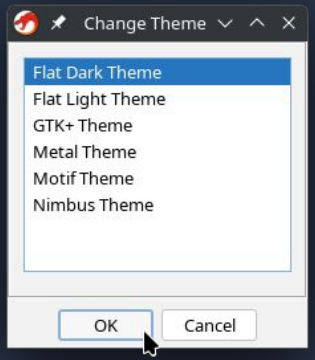
Simple stack strings

- code
- tutorial

Dark Theme


Ghidra: ctfs
File
Edit
Project
Tools
Help
Tool Chest
Active Project: ctfs
ctfs
keygenme
keygenme_
Filter:
Tree View
Table View
Running Tools
Workspace
Ghidra startup complete (4131 ms)


Ghidra: ctfs
File
Edit
Project
Tools
Help
Tool Options
Plugin Path...
Clear PKI Certificate...
Set PKI Certificate...
Theme
Switch...
Configure
New...
Import...
Export...
Delete...
keygenme_


Change Theme
Flat Dark Theme
Flat Light Theme
GTK+ Theme
Metal Theme
Motif Theme
Nimbus Theme
OK
Cancel

Additional Material for Ghidra

- What are you telling me, Ghidra?
- Mio padre StackSmashing

gdb [gef]

- r - run
- r < file.txt - run with input from file
- r args - run with arguments
- ni - next instructions
- si - step instruction
- br <addr> - break points
- c - continue executing
- fin - finish executing current function
- disass <function> - disassemble function
- x/4g <addr> - examine 4 long of memory
- x/s <addr> - examine as string
- x/i - examine as instructions

- got - show global offset table
- telescope - show stack
- canary - print canary
- vmmap - show memory regions

NOT intended to be complete list, refer to nicer cheatsheet or official documentation.

Java Decompiler

jd-gui

Z3 theorem prover

Crackme & keygenme

Spesso le challenge di reverse engineering prevedono l'analisi di software simili ai checker delle licenze.

Questo genere di challenge prende il nome di *crackme* ed emulano il processo di cracking dei programmi proprietari protetti da licenza.

Il tipico flusso di un software del genere prevede la lettura dell'input utente, una sua manipolazione ed il successivo controllo di validità, attraverso una stringa nota o altre procedure.

z3-solver

z3-solver è una libreria che ci permette di risolvere problemi di tipo SAT, cioè permette di trovare i valori che rendono vere delle equazioni che gli diamo in pasto.

Una volta ricavate quindi le operazioni che il software calcola sui dati in ingresso possiamo lasciare a z3 il compito di trovare i valori che rendono vero il sistema.

Installazione ed import

E’ una libreria python, possiamo quindi installarla attraverso pip:

```
pip install z3-solver
```

Per comodità importiamo tutto:

```
from z3 import *
```

Variabili

Dato che dobbiamo scrivere delle equazioni, necessitiamo di variabili da usare. Ne abbiamo di diversi tipi, i principali sono:

- intere: x = Int('x')
- booleane: x = Bool('x')
- bitvector: x = BitVec('x', <dimensione>)

Variabili (cont.)

Per rappresentare al meglio i valori dei programmi compilati conviene usare i BitVec in quanto permettono l' overflow e le operazioni bit a bit.

Le variabili intere invece non hanno una dimensione massima e sono più simili agli interi di python.

In una sola chiamata possiamo dichiarare più variabili:

```
x, y, z = BitVecs('x y z', 32)
```

Singole equazioni

Per risolvere delle singole equazioni possiamo usare la funzione *solve*:

```
solve(x * 2 == 4)
[x = 2]
```

Solver

Per costruire equazioni più complesse ed in maniera più automatizzata possiamo ricorrere all'oggetto Solver che ci permette di aggiungere equazioni e costruire un sistema completo da fargli risolvere.

Una volta costruito si può controllare se sia risolvibile o meno ed in caso positivo estrarre dei valori come soluzione.

Solver (cont.)

Per creare un oggetto solver:

```
s = Solver()
```

Per aggiungere una equazione:

```
s.add(x*2 == 4)
```

Per controllare la soddisfacibiltà:

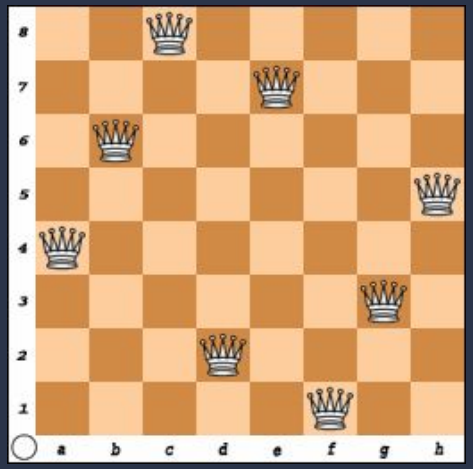
```
s.check()
```

Per estrarre la soluzione, se il sistema è **sat**:

```
s.model()
```

8 Queen Problem

```python
from z3 import *

s = Solver()
# definisco le 16 variabili come intere
chessboard = [ [ Int("x_%s_%s" % (i+1, j+1)) for j in range(8) ] for i in range(8) ]

# Impongo che possano essere solo 0 (cella vuota) o 1 (cella con regina)
for i in range(8):
    for j in range(8):
        s.add(  Or(chessboard[i][j] == 0, chessboard[i][j] == 1) )

# Impongo che in ogni riga ci sia esattamente una regina
for raw in chessboard:
    s.add( Sum(raw) == 1 )

# Impongo che in ogni colonna ci sia esattamente una regina
transposed = list(map(list, zip(*chessboard)))
for column in transposed:
    s.add( Sum(column) == 1 )


# Impongo che in ogni diagonale crescente ci sia al più una regina
for diagonal in range(8):
    diag1 = [ chessboard[diagonal-j][j] for j in range(diagonal+1)]
    diag2 = [ chessboard[7 - diagonal + j][7-j] for j in range(diagonal+1)]
    s.add( Sum(diag1) <= 1 )
    s.add( Sum(diag2) <= 1 )

# Impongo che in ogni diagonale decrescente ci sia al più una regina
reflected = [ raw[::-1] for raw in chessboard ]
for diagonal in range(8):
    diag1 = [ reflected[diagonal-j][j] for j in range(diagonal+1)]
    diag2 = [ reflected[7 - diagonal + j][7-j] for j in range(diagonal+1)]
    s.add( Sum(diag1) <= 1 )
    s.add( Sum(diag2) <= 1 )

if s.check() == sat:
    for raw in chessboard:
        print([s.model()[r] for r in raw])
```




```python
# We know each queen must be in a different row.
# So, we represent each queen by a single integer: the column position
Q = [ Int('Q_%i' % (i + 1)) for i in range(8) ]

# Each queen is in a column {1, ... 8 }
val_c = [ And(1 <= Q[i], Q[i] <= 8) for i in range(8) ]

# At most one queen per column
col_c = [ Distinct(Q) ]

# Diagonal constraint
diag_c = [ If(i == j,
              True,
              And(Q[i] - Q[j] != i - j, Q[i] - Q[j] != j - i))
           for i in range(8) for j in range(i) ]

solve(val_c + col_c + diag_c)
```

Demo

Risolviamo assieme la challenge coffee_hash.

Coffee Hash

```java
package defpackage;
public class coffee_hash {
    static String hash =
"630:624:622:612:609:624:623:610:624:624:567:631:638:639:658:593:546:605:607:585:648:636:635:704:7

    public static void main(String... strArr) {
        if (strArr.length != 1) {
            System.out.println("Usage: java Challenge <password>");
            System.exit(1);
        }
        if (checkPassword(strArr[0])) {
            System.out.printf("flag{%s}\n", strArr[0]);
        } else {
            System.out.println("Incorrect password!");
        }
    }

    public static boolean checkPassword(String str) {
        String str2 = "";
        for (int i = 0; i < str.length(); i++) {
            char c = 0;
            for (int i2 = 0; i2 < 7; i2++) {
                c += str.charAt((i + i2) % str.length());
            }
            str2 = str2 + (str2.length() == 0 ? Integer.valueOf(c) : ":" + c);
        }
        return hash.equals(str2);
    }
}
```

```python
from z3 import *
s = Solver()

hash_string =
"630:624:622:612:609:624:623:610:624:624:567:631:638:639:658:593:546:605:607:585:648:636:635:704:7
hash = [int(n) for n in hash_string.split(":")]

password_len = len(hash)

password = [BitVec('password_%d' % i,8) for i in range(password_len)]

# Printable Constraints
for i in range(password_len):
    s.add(And(password[i] >= 32, password[i] <= 126))

# Challenge Constraints
for i in range(password_len):
    cumulatore = []
    for j in range(7):
        cumulatore.append(password[ (i+j) % password_len])
    s.add(sum(cumulatore) == hash[i])

print(s.check())
if s.check() == sat:
    m = s.model()
    print("".join([chr(m[password[i]].as_long()) for i in range(password_len)]))
```

Risorse

- Documentazione completa di z3: link
- Programming z3

- Introduction to angr
- Raccolta di challenge per imparare angr
- Klee: un altro symbolic execution engine

LD_PRELOAD trick

Now we'd like to achieve whats called Function Hooking.

We can hook and substitute every function in a shared object (like libc)

ld.so & ldd

The programs ld.so and ld-linux.so find and load the shared objects (shared libraries) needed by a program, prepare the program to run, and then run it.

```
❯ ldd chall
        linux-vdso.so.1 (0x00007fff957d6000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007eff7e000000)
        /lib64/ld-linux-x86-64.so.2 (0x00007eff7e32b000)
```

LD_PRELOAD

LD_PRELOAD is and environment variable used to specify shared object to be loaded before all the others.

This feature can be used to selectively override functions in other shared objects.

```
❯ LD_PRELOAD=./hook.so ldd chall
        linux-vdso.so.1 (0x00007ffc9efa6000)
        ./hook.so (0x00007fcc56d0b000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fcc56a00000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fcc56d17000)
```

How to?

dlsym: obtain address of a symbol in a shared object or executable

```c
// gcc -o hook.so -shared hook.c -fPIC
#define _GNU_SOURCE
#include <stdio.h>
#include <dlfcn.h>

typedef FILE *(*fopen_t)(const char *pathname, const char *mode);
fopen_t real_fopen;

FILE *fopen(const char *pathname, const char *mode) {
  fprintf(stderr, "called fopen(%s, %s)\n", pathname, mode);
  if (!real_fopen) {
    real_fopen = dlsym(RTLD_NEXT, "fopen");
  }

  return real_fopen(pathname, mode);
}

__attribute__((constructor)) static void setup(void) {
  fprintf(stderr, "called setup()\n");
}
```