Reverse Engineering with Ghidra

Paolo Falcarin

Ca' Foscari University of Venice

Department of Enviromental Sciences, Informatics and Statistics

paolo.falcarin@unive.it

CM0626 – Software Security

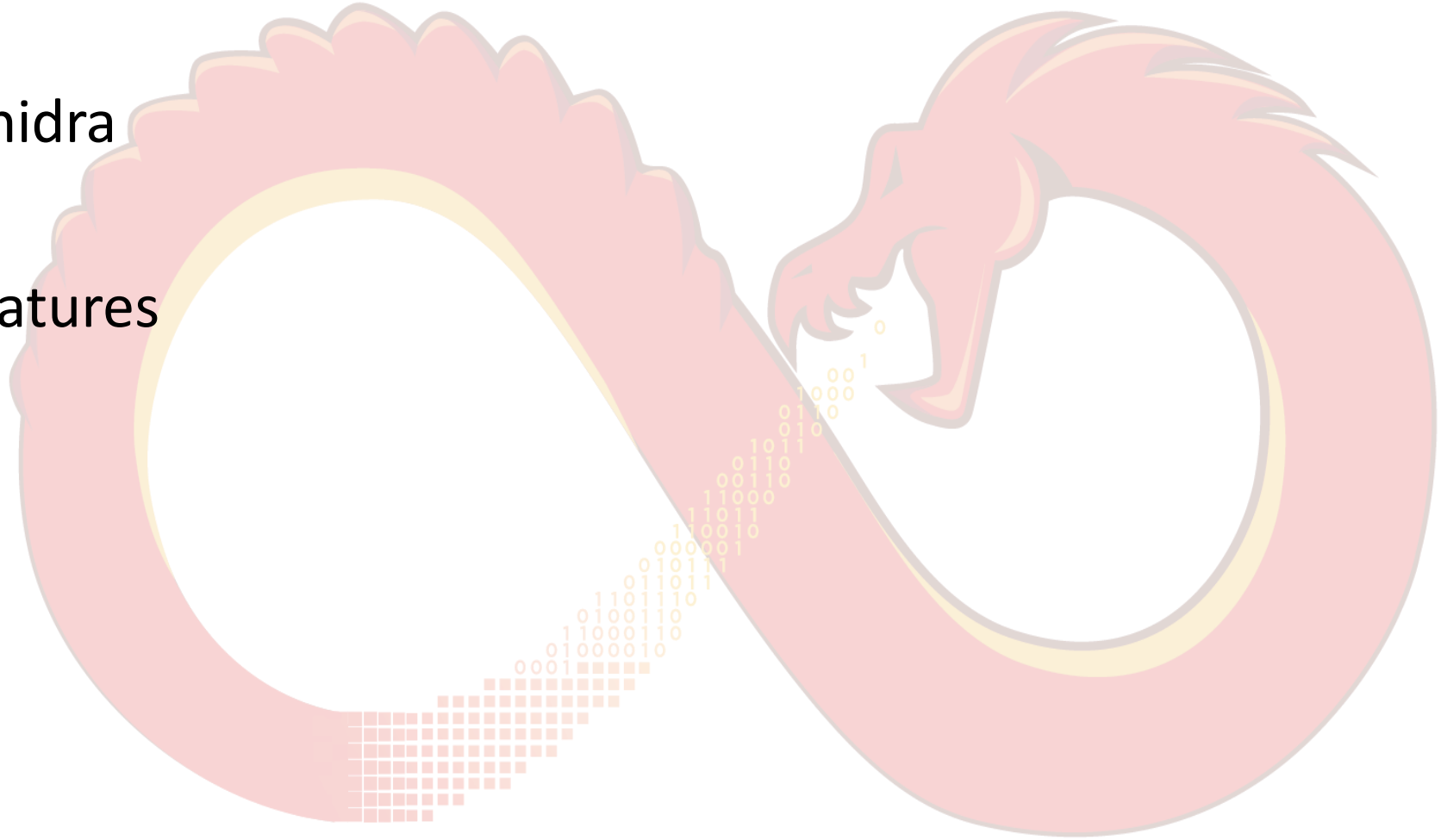
CM0631-2 – Software Security

25 February 2025

Outline

Ca' Foscari University of Venice

- Ghidra Overview
- Static Attack with Ghidra
- Ghidra Scripting
- Ghidra Advanced Features



Reverse Engineering Tools

Ca' Foscari University of Venice

- IDA Pro from Hex Rays
  - Historical commercial leader
  - https://hex-rays.com/ida-pro/
- BinaryNinja
  - Emerging commercial leader
  - https://binary.ninja/
- Radare2
  - Kali-Linux Open-Source Tool
  - https://rada.re/n/radare2.html
- Ghidra
  - Emerging Multi-platform Open-Source Tool
  - Published in early 2019

Other Reverse Engineering Tools

Ca' Foscari University of Venice

- ODA online disassembler
  - Many Hardware architectures supported
  - https://syscall7.com/oda/
- Hopper
  - For Mac and Linux
  - https://www.hopperapp.com
- Angr
  - open-source binary analysis platform for Python.
  - It combines both static and dynamic symbolic ("concolic") analysis, providing tools to solve a variety of tasks.
  - https://angr.io/

Ghidra

Ca' Foscari University of Venice

- Java-based interactive reverse engineering tool developed by US National Security Agency - similar in functionality to IDA Pro, Binary Ninja, etc…
- Static analysis only currently, debugger support promised to be coming soon
- Runs on Mac, Linux, and Windows
- All credit for creating Ghidra goes to the developers at NSA
- Released open source at RSA in March 2019
- 1.2M+ lines of code
- NSA has not discussed the history of the tool, but comments in source files go as far back as February 1999

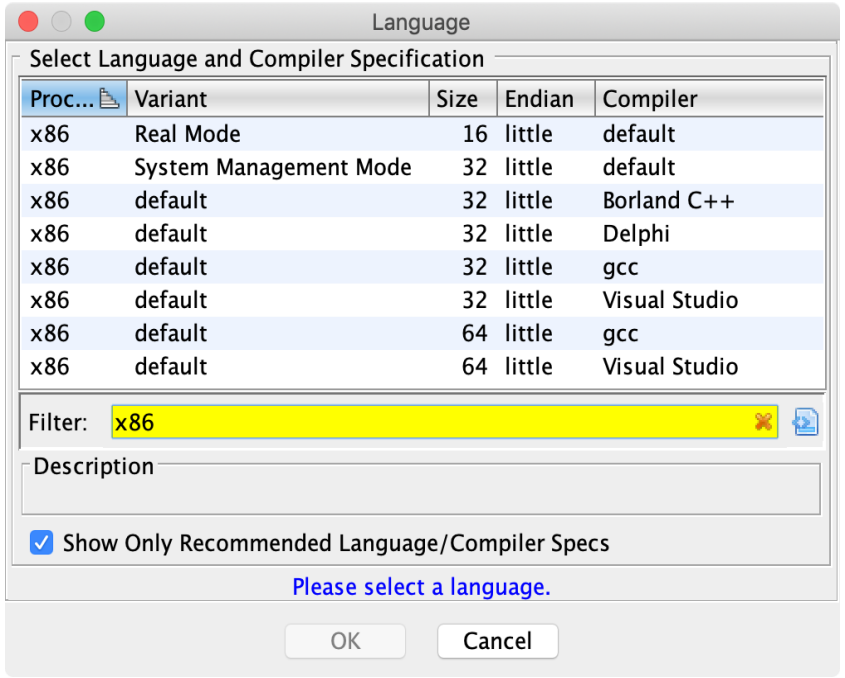
```
$ grep -r "1999" * --include *.java
src/Generic-src/ghidra/util/exception/NotYetImplementedException.java: * @version 1999/02/05
src/SoftwareModeling-src/ghidra/program/model/address/AddressOutOfBoundsException.java: * @version 1999-03-31
src/SoftwareModeling-src/ghidra/program/model/scalar/ScalarOverflowException.java: * @version 1999-03-31
src/SoftwareModeling-src/ghidra/program/model/scalar/ScalarFormat.java: * @version 1999/02/04
```

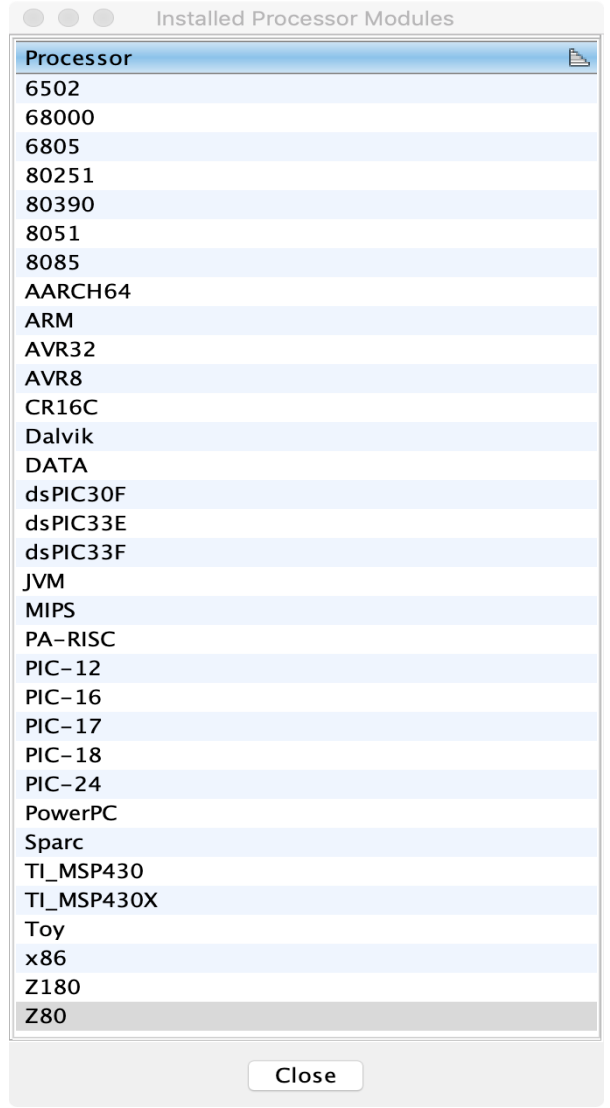
Architecture Support

Ca' Foscari University of Venice

- Supports a variety of common desktop, embedded, and VM architectures
- Can handle unique compiler idioms and CPU modes
- Users can extend Ghidra with their own custom processor modules
- Ghidra can decompile

anything it can disassemble

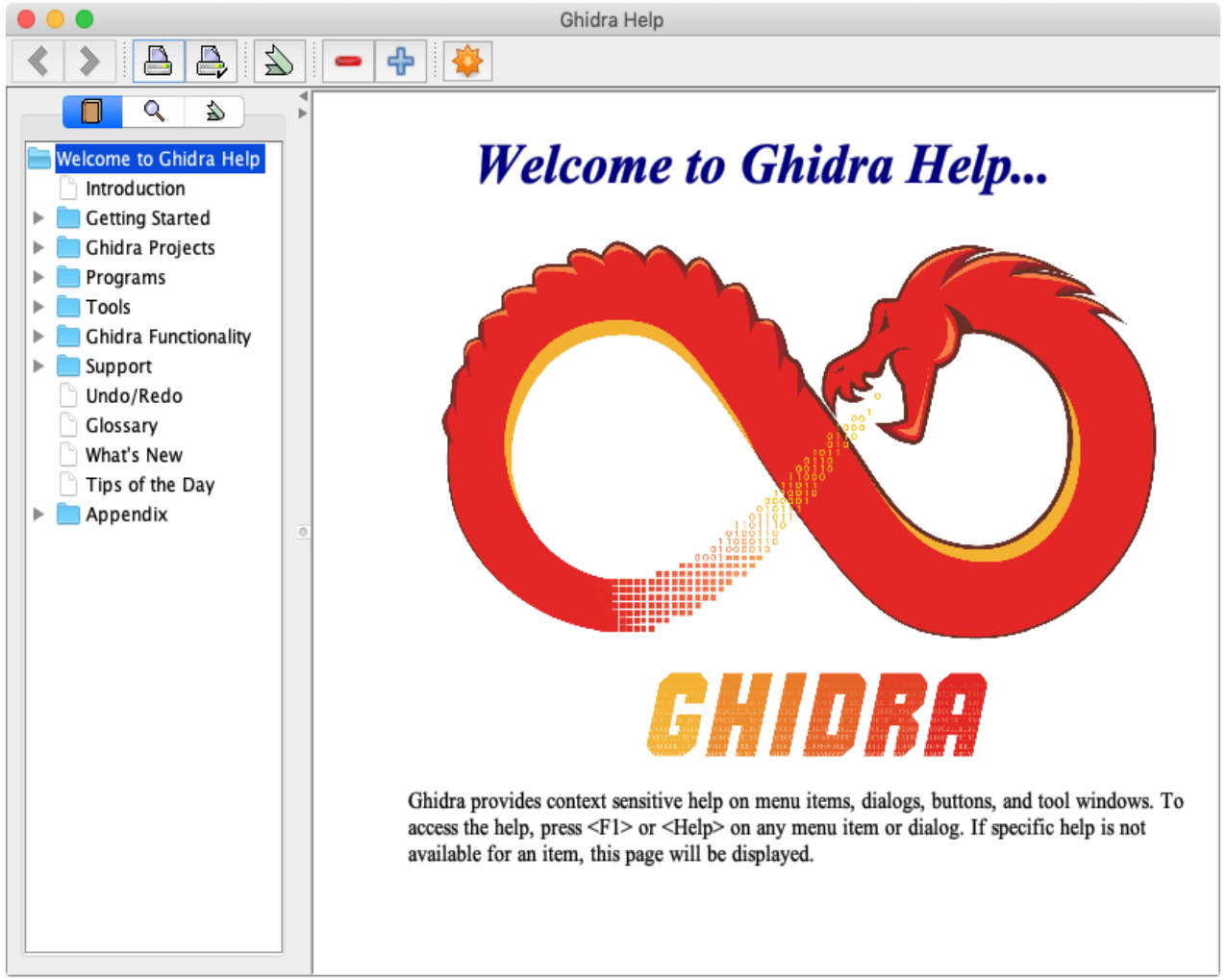




Documentation

Ca' Foscari University of Venice

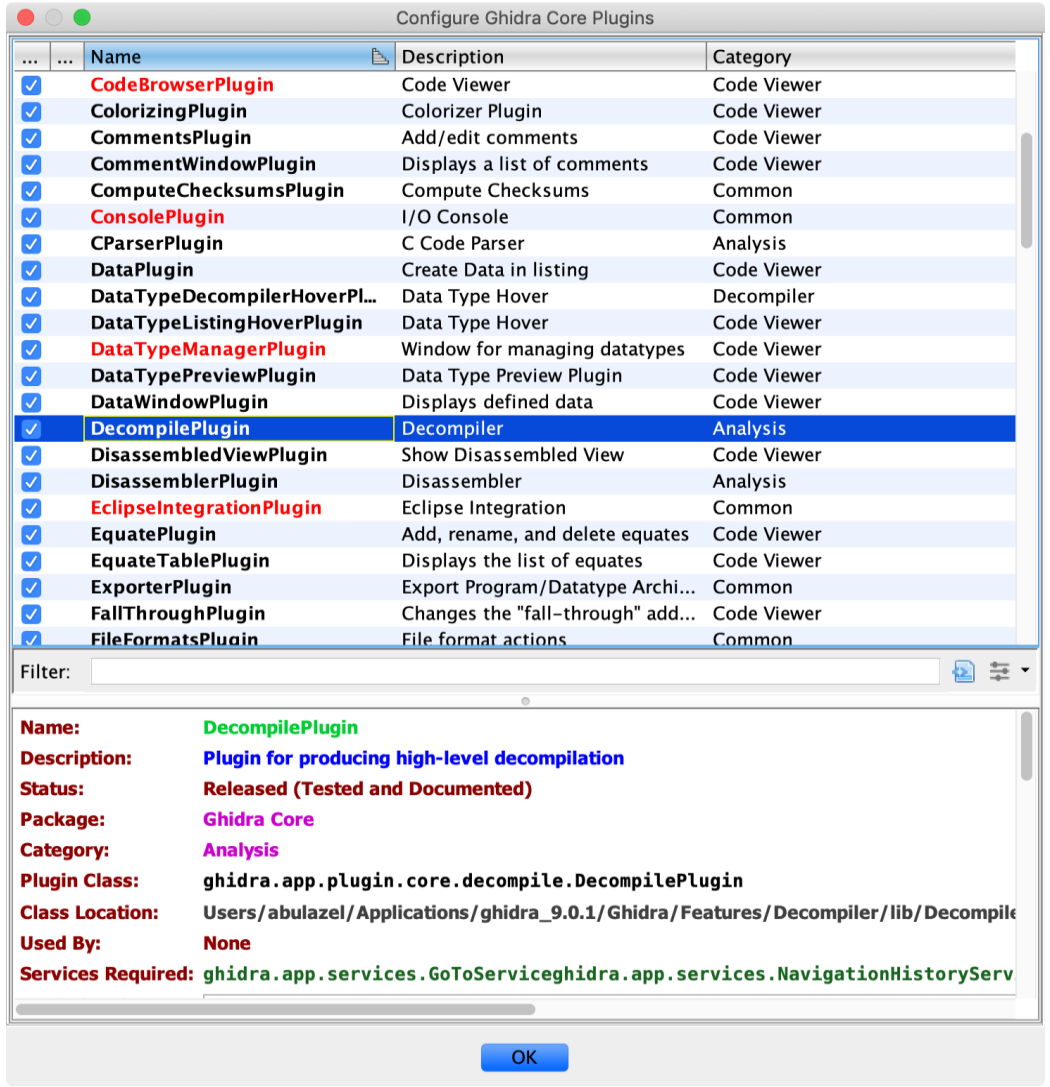
- Help > Contents
- F1 or Help key while pointing at any field or menu option
- docs directory
- JavaDoc
- Several classes
- P-code



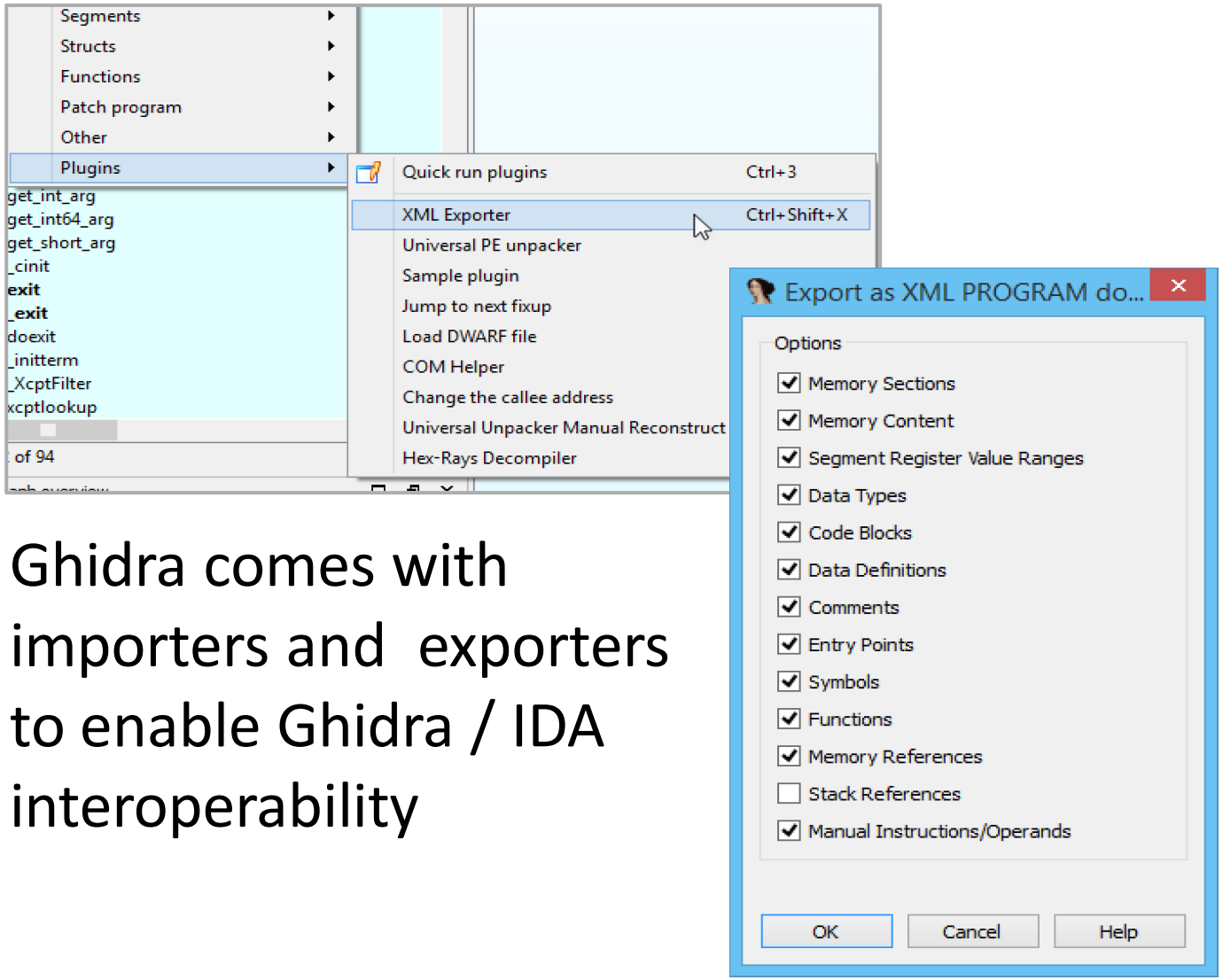
Tools

Ca' Foscari University of Venice

- Ghidra tools are assemblies of plugins
- Ghidra comes with two tools: CodeBrowser and VersionTracker
- Tools can be configured and customized to suit unique RE needs



IDA Interoperability



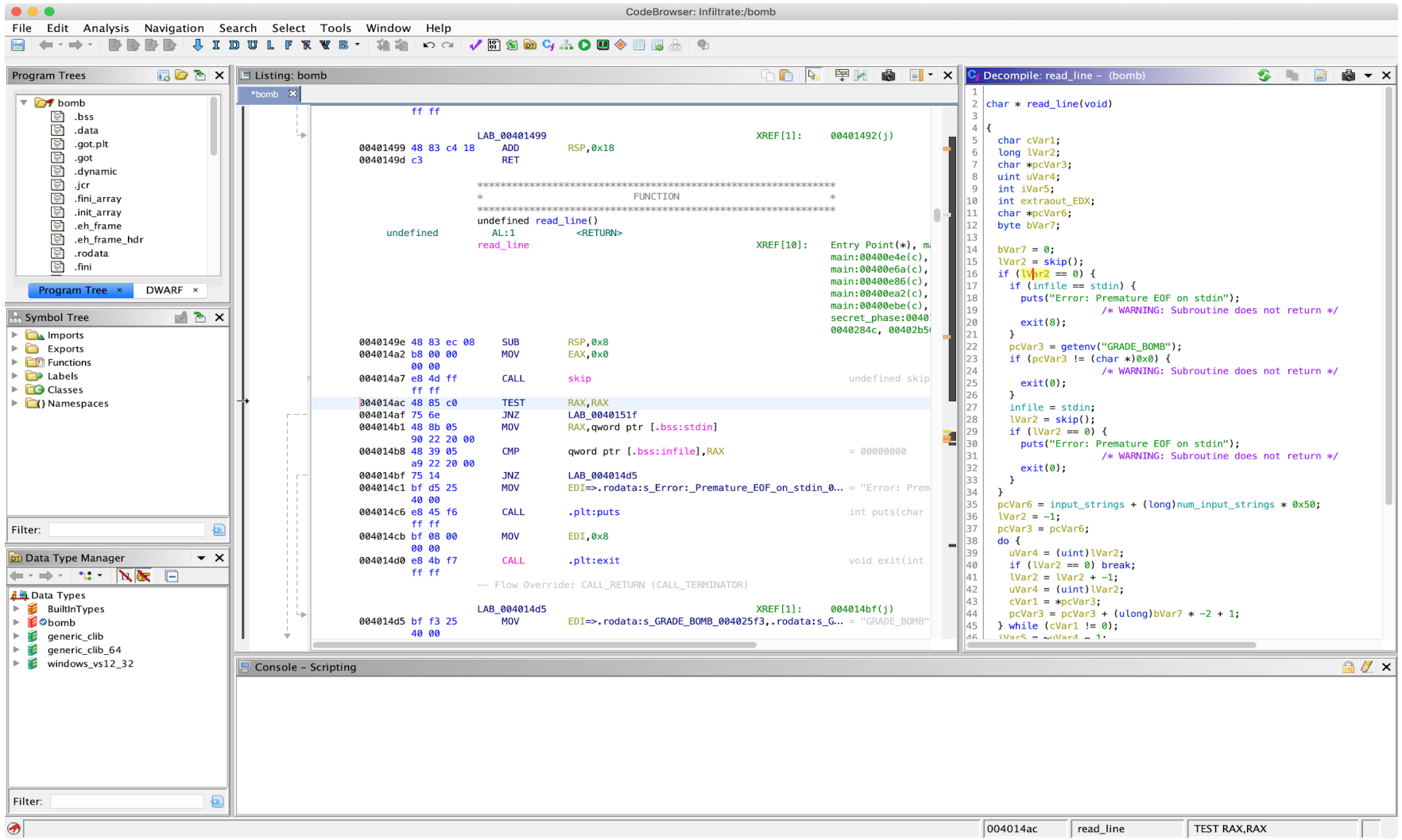
Ghidra comes with importers and exporters to enable Ghidra / IDA interoperability

Decompile: FUN_00401070 - (13-2)

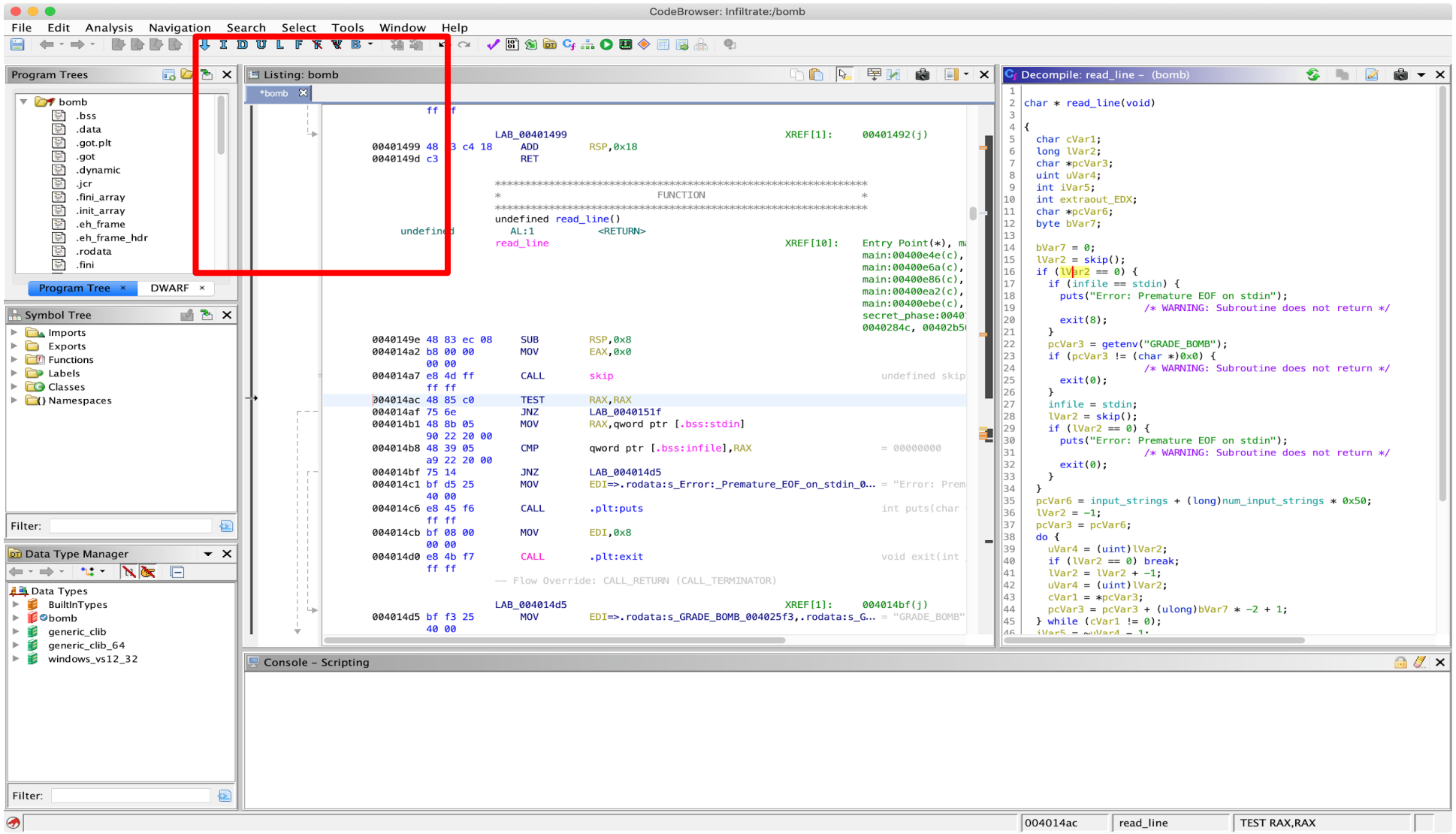
```
22  undefined *h;
23  word local_18;
24  dword local_16;
25  dword local_e;
26  dword cy;
27
28  uVar1 = (*(code *)(undefined *)0xffffffff)(0);
29  uVar2 = (*(code *)(undefined *)0xffffffff)(1);
30  hWnd = (*(code *)(undefined *)0xffffffff)();
31  hDC = (*(code *)(undefined *)0xffffffff)(hWnd);
32  uVar3 = (*(code *)(undefined *)0xffffffff)(hDC);
33  uVar4 = (*(code *)(undefined *)0xffffffff)(hDC,uVar1,uVar2);
34  (*(code *)(undefined *)0xffffffff)(uVar3,uVar4);
35  (*(code *)(undefined *)0xffffffff)(uVar3,0,0,uVar1,uVar2,hDC,0,0,0xcc0020);
36  (*(code *)(undefined *)0xffffffff)(uVar4,0x18,&pv);
37  bmi.bmiHeader.biSize = 0x28;
38  bmi.bmiHeader.biPlanes = 1;
39  bmi.bmiHeader.biBitCount = 0x20;
40  bmi.bmiHeader.biCompression._0_2_ = 0;
41  bmi.bmiHeader.biCompression._2_2_ = 0;
42  bmi.bmiHeader.biSizeImage = 0;
43  bmi.bmiHeader.biXPelsPerMeter = 0;
44  bmi.bmiHeader.biYPelsPerMeter = 0;
45  bmi.bmiHeader.biClrUsed = 0;
46  bmi.bmiHeader.biClrImportant = 0;
47  iVar6 = local_6c * 0x20 + 0x1f;
48  iVar6 = ((int)(iVar6 + (iVar6 >> 0x1f & 0x1fU)) >> 5) * 4 * cLines;
49  uVar1 = (*(code *)(undefined *)0xffffffff)(0x42,iVar6);
50  bmi.bmiColors = (*(code *)(undefined *)0xffffffff)(uVar1);
51  (*(code *)(undefined *)0xffffffff)(hDC,uVar4,0,cLines,bmi.bmiColors,&bmi,0);
52  local_18 = 0x4d42;
53  uVar2 = (*(code *)(undefined *)0xffffffff)(0x42,iVar6 + 0x36);
54  iVar5 = (*(code *)(undefined *)0xffffffff)(uVar2);
55  _memcpy(iVar5,&local_18,0xe);
56  _memcpy(iVar5 + 0xe,&bmi,0x28);
57  _memcpy(iVar5 + 0x36,bmi.bmiColors,iVar6);
58  (*(code *)(undefined *)0xffffffff)(uVar1);
59  (*(code *)(undefined *)0xffffffff)(uVar1);
60  (*(code *)(undefined *)0xffffffff)(hWnd,hDC);
61  (*(code *)(undefined *)0xffffffff)(uVar3);
62  (*(code *)(undefined *)0xffffffff)(uVar4);
63  *(int *)param_1 = iVar5;
64  *(int *)param_2 = iVar6 + 0x36;
65  return;
```

Default UI - CodeBrowser

Ca' Foscari University of Venice

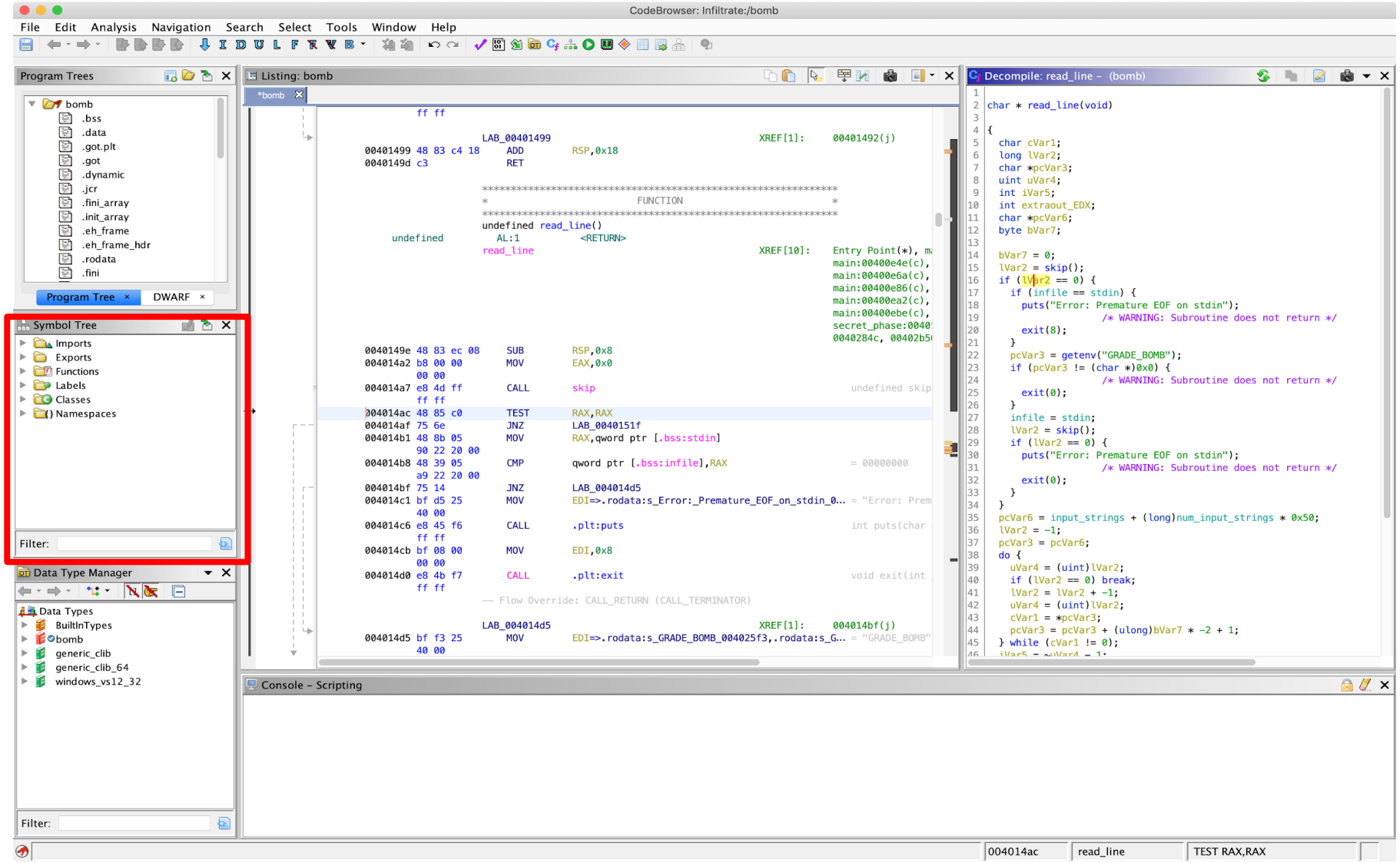


Default UI - Program Trees

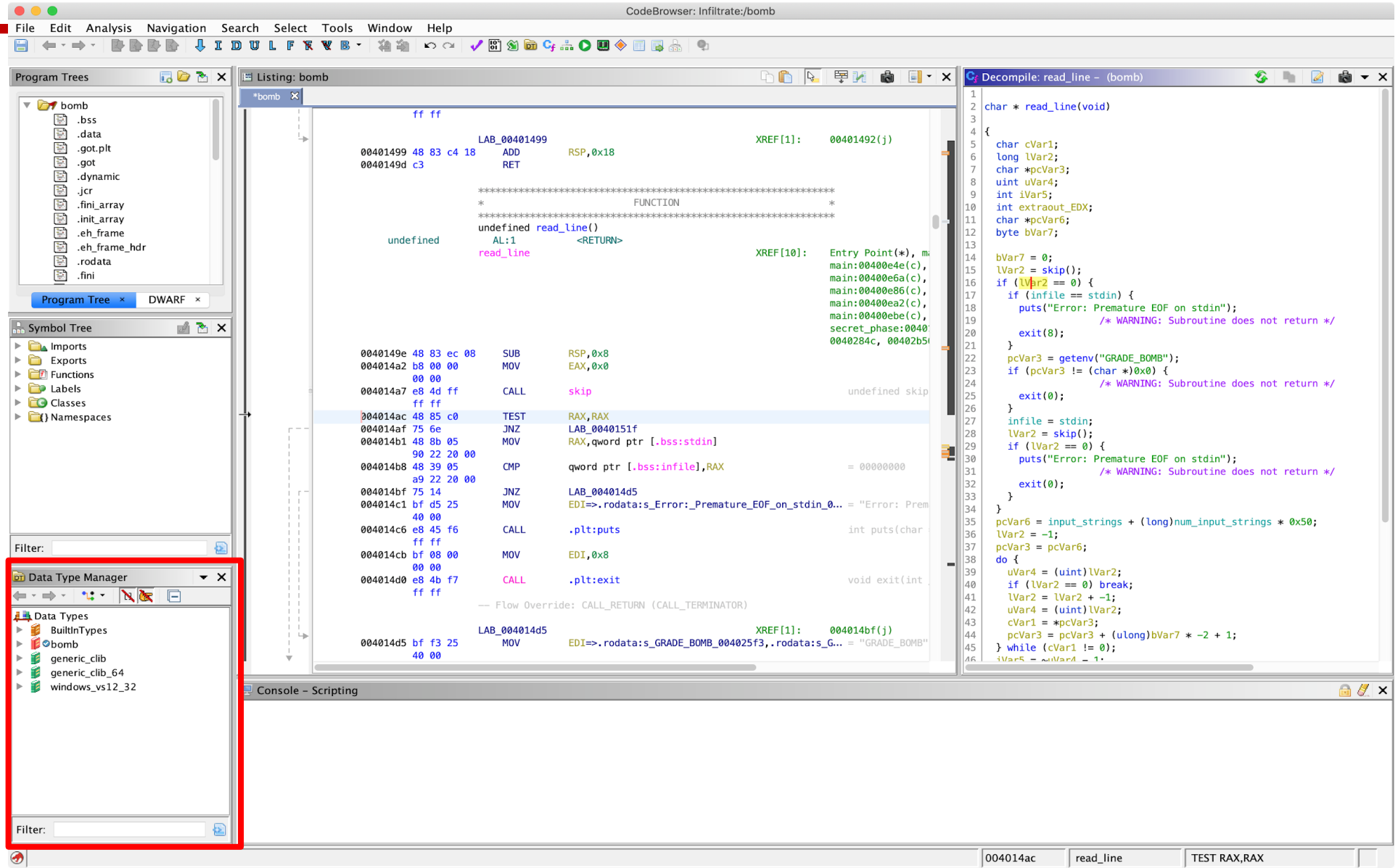


Default UI - Symbol Tree

Ca' Foscari University of Venice

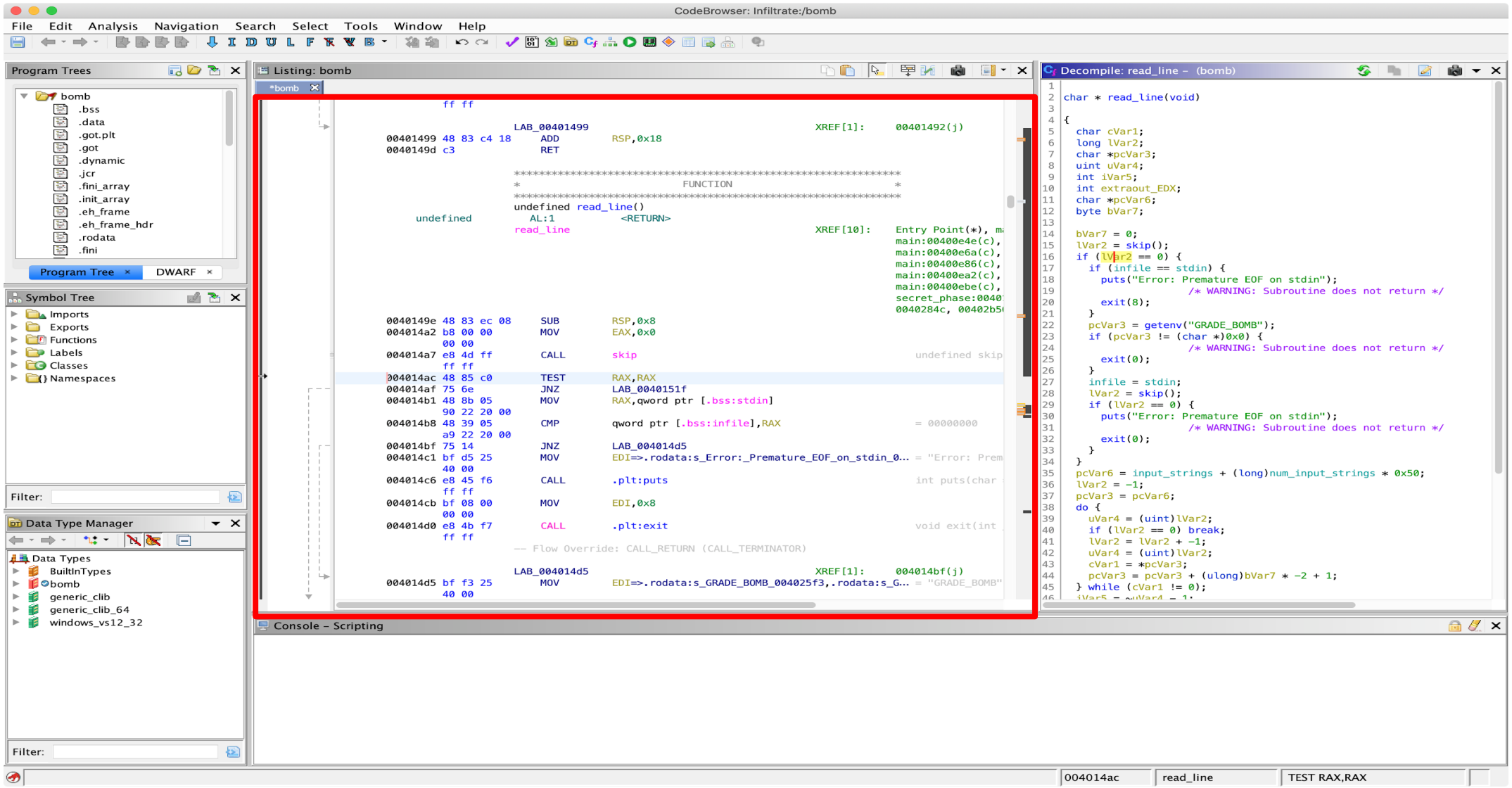


Default UI - Data Type Manager

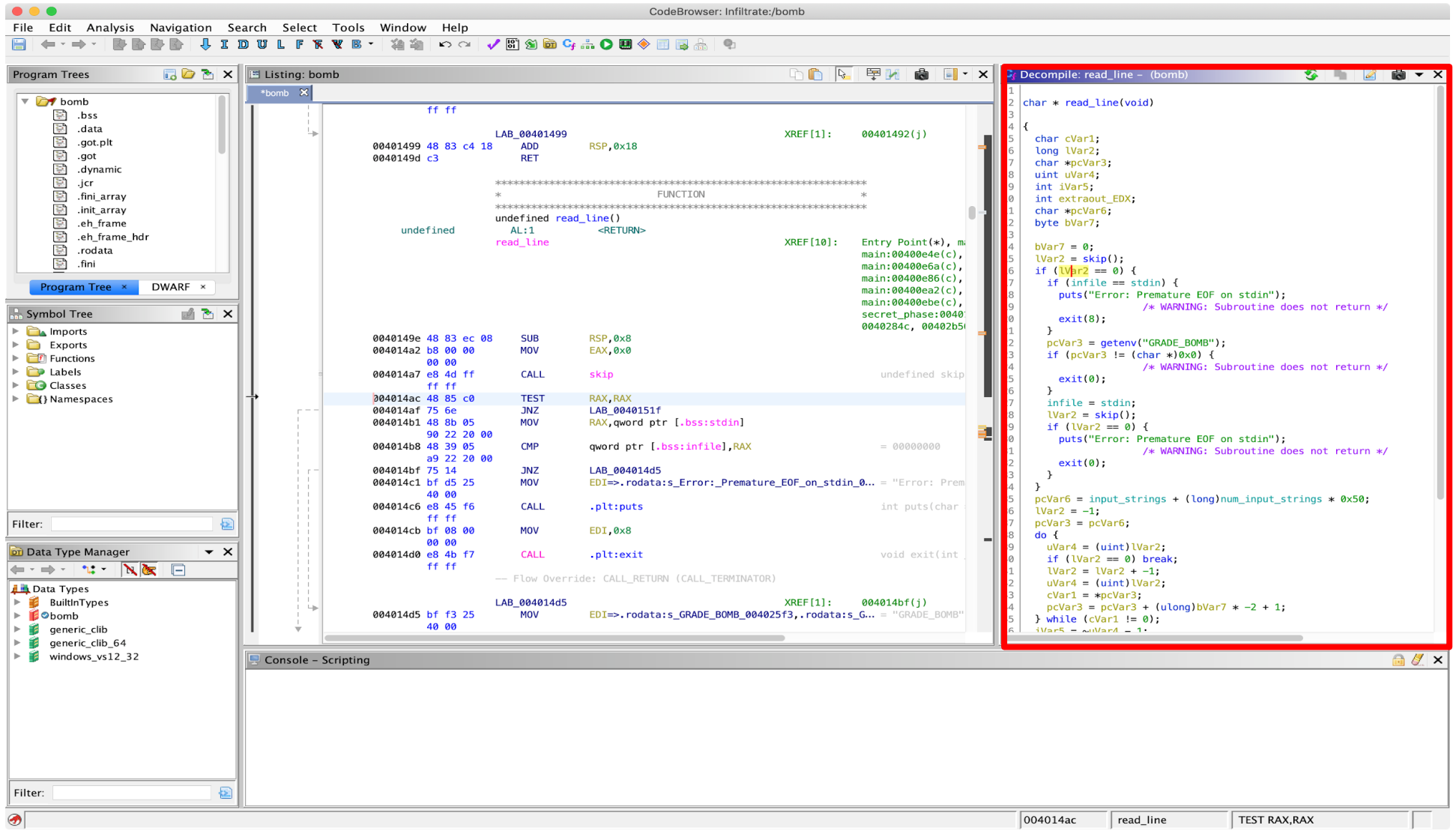


Default UI - Listing (Disassembly)

Ca' Foscari University of Venice

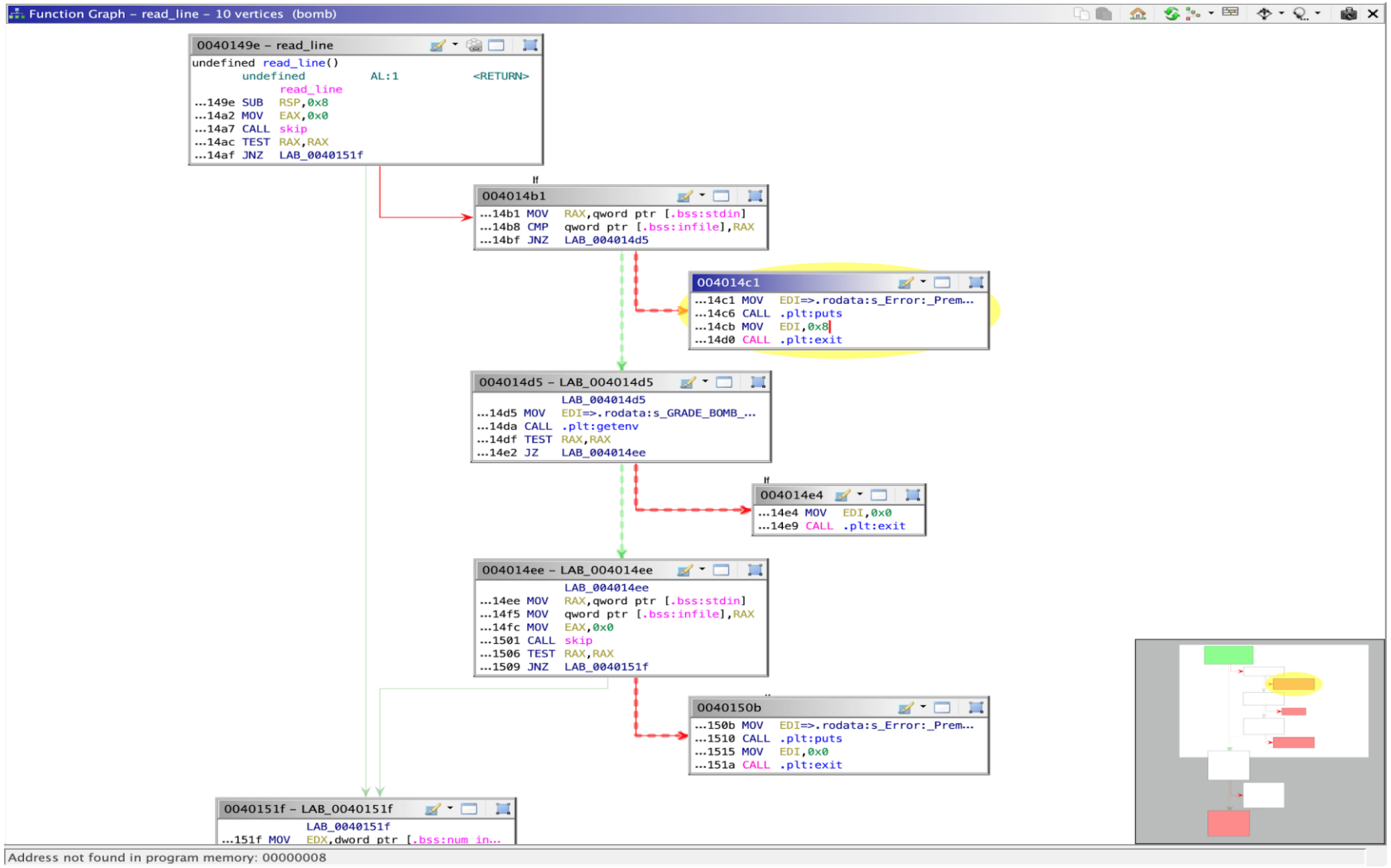


Default UI - Decompiler



Control Flow Graph

Ca' Foscari University of Venice



Disassembly with P-Code

Ca' Foscari University of Venice

Listing: bomb

*bomb

```
004014cb bf 08 00 00 00     MOV         EDI,0x8
                                        (register, 0x38, 8) = COPY (const, 0x8, 8)
004014d0 e8 4b f7 ff ff     CALL        .plt:exit                                   void exit(int __status)
                                        (register, 0x20, 8) = INT_SUB (register, 0x20, 8), (const, 0x8, 8)
                                        STORE (const, 0x131, 8), (register, 0x20, 8), (const, 0x4014d5, 8)
                                        CALL (ram, 0x400c20, 8)
                                        RETURN (const, 0x0, 8)
                    -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                    LAB_004014d5                                        XREF[1]:     004014bf(j)
004014d5 bf f3 25 40 00     MOV         EDI=>.rodata:s_GRADE_BOMB_004025f3,.rodata:s_G... = "GRADE_BOMB"
                                        (register, 0x38, 8) = COPY (const, 0x4025f3, 8)
004014da e8 01 f6 ff ff     CALL        .plt:getenv                                 char * getenv(char * __name)
                                        (register, 0x20, 8) = INT_SUB (register, 0x20, 8), (const, 0x8, 8)
                                        STORE (const, 0x131, 8), (register, 0x20, 8), (const, 0x4014df, 8)
                                        CALL (ram, 0x400ae0, 8)
004014df 48 85 c0           TEST        RAX,RAX
                                        (register, 0x200, 1) = COPY (const, 0x0, 1)
                                        (register, 0x20b, 1) = COPY (const, 0x0, 1)
                                        (unique, 0xbb10, 8) = INT_AND (register, 0x0, 8), (register, 0x0, 8)
                                        (register, 0x207, 1) = INT_SLESS (unique, 0xbb10, 8), (const, 0x0, 8)
                                        (register, 0x206, 1) = INT_EQUAL (unique, 0xbb10, 8), (const, 0x0, 8)
004014e2 74 0a              JZ          LAB_004014ee
                                        CBRANCH (ram, 0x4014ee, 8), (register, 0x206, 1)
004014e4 bf 00 00 00 00     MOV         EDI,0x0
                                        (register, 0x38, 8) = COPY (const, 0x0, 8)
004014e9 e8 32 f7 ff ff     CALL        .plt:exit                                   void exit(int __status)
                                        (register, 0x20, 8) = INT_SUB (register, 0x20, 8), (const, 0x8, 8)
                                        STORE (const, 0x131, 8), (register, 0x20, 8), (const, 0x4014ee, 8)
                                        CALL (ram, 0x400c20, 8)
                                        RETURN (const, 0x0, 8)
                    -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                    LAB_004014ee                                        XREF[1]:     004014e2(j)
004014ee 48 8b 05 53 22 20 00   MOV     RAX,qword ptr [.bss:stdin]
                                        (register, 0x0, 8) = COPY (ram, 0x603748, 8)
004014f5 48 89 05 6c 22 20 00   MOV     qword ptr [.bss:infile],RAX                 = 00000000
                                        (ram, 0x603768, 8) = COPY (register, 0x0, 8)
004014fc b8 00 00 00 00     MOV         EAX,0x0
```

Decompilation across Architectures - x64

Ca' Foscari University of Venice

Listing: elf-Linux-x64-bash

*elf-Linux-x64-bash

```
00493a9e 75 c0           JNZ         LAB_00493a60

                     LAB_00493aa0                                    XREF[2]:
00493aa0 e8 0b af        CALL        rl_end_undo_group
         00 00
00493aa5 48 8b 7c        MOV         RDI,qword ptr [RSP + local_50]
         24 08
00493aaa e9 01 ff        JMP         LAB_004939b0
         ff ff
00493aaf 90              ??          90h

                     LAB_00493ab0                                    XREF[1]:
00493ab0 48 8b 7c        MOV         RDI=>local_50,qword ptr [RSP + 0x8]
         24 08

                     LAB_00493ab5                                    XREF[2]:
00493ab5 e8 86 ee        CALL        FUN_00492940
         ff ff
00493aba 48 8b 7c        MOV         RDI,qword ptr [RSP + local_50]
         24 08
00493abf e9 ec fe        JMP         LAB_004939b0
         ff ff
00493ac4 0f              ??          0Fh
00493ac5 1f              ??          1Fh
00493ac6 40              ??          40h    @
00493ac7 00              ??          00h

                     LAB_00493ac8                                    XREF[1]:
00493ac8 48 8b 0d        MOV         RCX,qword ptr [.bss:rl_line_buffer]
         d9 16 25 00
00493acf 48 63 d5        MOVSXD      RDX,EBP
00493ad2 3a 44 11 ff     CMP         AL,byte ptr [RCX + RDX*0x1 + -0x1]
00493ad6 0f 94 c0        SETZ        AL
00493ad9 0f b6 c0        MOVZX       EAX,AL
00493adc 29 c5           SUB         EBP,EAX
00493ade e9 48 ff        JMP         LAB_00493a2b
         ff ff
00493ae3 0f              ??          0Fh
00493ae4 1f              ??          1Fh
00493ae5 44              ??          44h    D
00493ae6 00              ??          00h
00493ae7 00              ??          00h
```

Decompile: rl_complete_internal - (elf-Linux-x64-bash)

```
41  pcVar9 = rl_completion_entry_function;
42  if (rl_completion_entry_function == (code *)0x0) {
43    pcVar9 = rl_filename_completion_function;
44  }
45  uVar5 = 0;
46  if (rl_point != 0) {
47    local_39[0] = _rl_find_completion_word(&local_44,&local_40);
48    uVar5 = rl_point;
49  }
50  rl_point = uVar1;
51  __src = (char *)rl_copy_text((ulong)uVar5,(ulong)uVar1);
52  uVar7 = 0;
53  local_50 = (char **)FUN_00492490(__src,(ulong)uVar5,(ulong)uVar1,pcVar9,(ulong)loc
54                                   (ulong)(uint)(int)local_39[0]);
55  if (local_50 != (char **)0x0) {
56    iVar3 = strcmp(__src,*local_50);
57    uVar7 = (ulong)(iVar3 != 0);
58  }
59  free(__src);
60  if ((local_50 == (char **)0x0) ||
61     (iVar3 = FUN_00490cb0(&local_50,(ulong)rl_filename_completion_desired), ppcVar8
62     iVar3 == 0)) {
63    rl_ding();
64    if (__dest != (char *)0x0) {
65      free(__dest);
66    }
67    rl_readline_state = rl_readline_state & 0xffffbfff;
68    DAT_006e5570 = 0;
69    rl_completion_found_quote = 0;
70    rl_completion_quote_character = 0;
71    return 0;
72  }
73  if (uParm1 == 0x2a) {
74    rl_begin_undo_group();
75    if ((uVar5 != 0) && (local_39[0] != 0)) {
76      uVar5 = uVar5 - (uint)(local_39[0] == rl_line_buffer[(long)(int)uVar5 + -1]);
77    }
78    rl_delete_text((ulong)uVar5,(ulong)rl_point);
79    __src = ppcVar8[1];
80    if (__src == (char *)0x0) {
81      rl_point = uVar5;
82      __src = (char *)FUN_004916e0(*ppcVar8,1,local_39);
83      rl_insert_text(__src);
84      rl_insert_text(&DAT_004ad25a);
```

Decompilation across Architectures - SPARC

Ca' Foscari University of Venice

Listing: elf-Linux-SparcV8-bash

*elf-Linux-SparcV8-bash

```
                        undefined type_builtin()
      undefined          o0:1          <RETURN>
                        type_builtin                                XREF[1]:
0008e320 9d e3 bf a0     save        sp,-0x60,sp
0008e324 80 a6 20 00     cmp         i0,0x0
0008e328 02 40 00 49     bpe,pn      %icc,LAB_0008e44c
0008e32c 82 10 20 00     _mov        0x0,g1
0008e330 c2 06 20 04     lduw        [i0+0x4],g1
0008e334 c2 00 40 00     lduw        [g1+g0],g1
0008e338 c4 48 40 00     ldsb        [g1+g0],g2
0008e33c 80 a0 a0 2d     cmp         g2,0x2d
0008e340 12 40 00 30     bpne,pn     %icc,LAB_0008e400
0008e344 25 00 03 5d     _sethi      %hi(0xd7400),l2
0008e348 2d 00 03 5d     sethi       %hi(0xd7400),l6
0008e34c 2b 00 03 5d     sethi       %hi(0xd7400),l5
0008e350 3b 00 03 5d     sethi       %hi(0xd7400),i5
0008e354 37 00 03 5d     sethi       %hi(0xd7400),i3
0008e358 27 00 03 21     sethi       %hi(0xc8400),l3
0008e35c a4 14 a3 28     or          l2,0x328,l2
0008e360 ac 15 a3 18     or          l6,0x318,l6
0008e364 aa 15 63 10     or          l5,0x310,l5
0008e368 ba 17 63 20     or          i5,0x320,i5
0008e36c b6 16 e3 30     or          i3,0x330,i3
0008e370 a6 14 e0 20     or          l3,0x20,l3
0008e374 a0 10 00 18     mov         i0,l0
0008e378 a8 10 20 74     mov         0x74,l4
0008e37c b8 10 20 61     mov         0x61,i4
0008e380 ae 10 20 70     mov         0x70,l7
0008e384 c4 48 60 01     ldsb        [g1+0x1],g2

                        LAB_0008e388                                XREF[1]:
0008e388 a2 00 60 01     add         g1,0x1,l1
0008e38c 80 a0 a0 74     cmp         g2,0x74
0008e390 02 40 00 31     bpe,pn      %icc,LAB_0008e454
0008e394 c2 08 60 01     _ldub       [g1+0x1],g1
0008e398 80 a0 a0 2d     cmp         g2,0x2d
0008e39c 02 40 00 39     bpe,pn      %icc,LAB_0008e480
0008e3a0 80 a0 a0 70     _cmp        g2,0x70
```

Decompile: type_builtin - (elf-Linux-SparcV8-bash)

```
15 type_builtin(undefined4 *puParm1,undefined4 uParm2,undefined4 uParm3,undefined *puParr
16             undefined4 uParm5,char *pcParm6)
17
18 {
19   char cVar1;
20   char *pcVar2;
21   int *piVar3;
22   int iVar4;
23   int iVar5;
24   undefined4 uVar6;
25   undefined4 *puVar7;
26   uint uVar8;
27   char *__s1;
28   int **ppiVar9;
29   uint uVar10;
30
31   if (puParm1 == (undefined4 *)0x0) {
32     return 0;
33   }
34   pcVar2 = *(char **)puParm1[1];
35   if (*pcVar2 == '-') {
36     pcParm6 = "-path";
37     puParm4 = &DAT_000d7730;
38     uParm5 = 0x61;
39     cVar1 = pcVar2[1];
40     puVar7 = puParm1;
41     do {
42       __s1 = pcVar2 + 1;
43       if (cVar1 == 't') {
44         iVar4 = strcmp(__s1,"type");
45         if (iVar4 == 0) {
46           *__s1 = 't';
47 LAB_0008e470:
48           *(undefined *)(*(int *)puVar7[1] + 2) = 0;
49           goto LAB_0008e3d8;
50         }
51         puVar7 = (undefined4 *)*puVar7;
52       }
53       else {
```

Decompilation Across Architectures - PowerPC

Ca' Foscari University of Venice

Listing: MachO-OSX-ppc-openssl-1.0.1h

*elf-Linux-SparcV8-bash | *MachO-OSX-ppc-openssl-1.0.1h

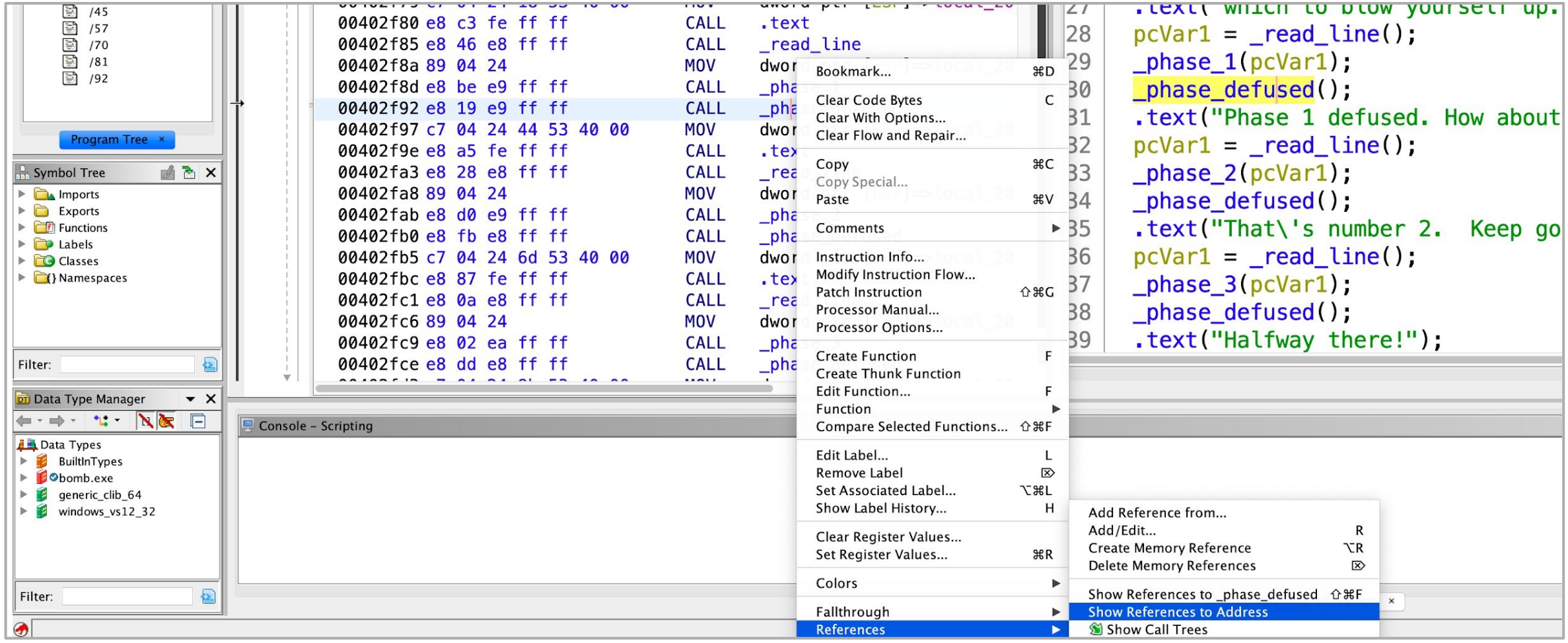
```
                        LAB_000212c8                                    XR
000212c8 40 9e 00 10     bne         cr7,LAB_000212d8
000212cc 48 00 00 18     b           LAB_000212e4

                        LAB_000212d0                                    XR
000212d0 2f 82 00 00     cmpwi       cr7,r2,0x0
000212d4 40 9e 00 10     bne         cr7,LAB_000212e4

                        LAB_000212d8                                    XR
000212d8 68 02 00 2f     xori        r2,r0,0x2f
000212dc 7c 42 00 34     cntlzw      r2,r2
000212e0 54 42 d9 7e     rlwinm      r2,r2,0x1b,0x5,0x1f

                        LAB_000212e4                                    XR
000212e4 39 29 00 01     addi        r9,r9,0x1

                        LAB_000212e8                                    XR
000212e8 88 09 00 00     lbz         r0,0x0(r9)
000212ec 7c 00 07 75     extsb.      r0,r0
000212f0 40 82 ff 98     bne         LAB_00021288

                        LAB_000212f4                                    XR
000212f4 2f 82 00 03     cmpwi       cr7,r2,0x3
000212f8 38 00 00 01     li          r0,0x1
000212fc 41 9e 00 10     beq         cr7,LAB_0002130c
00021300 38 02 00 01     addi        r0,r2,0x1
00021304 7c 00 00 34     cntlzw      r0,r0
00021308 54 00 d9 7e     rlwinm      r0,r0,0x1b,0x5,0x1f

                        LAB_0002130c                                    XR
0002130c 40 82 00 24     bne         LAB_00021330
00021310 3c 9f 00 04     addis       r4,r31,0x4
00021314 7f 43 d3 78     or          r3,r26,r26
00021318 80 84 fe 18     lwz         r4=>__cstring:s_HTTP/1.0_200_ok_Cor

0002131c 48 02 39 51     bl          __symbol_stub1:__symbol_stub1::_BIO
00021320 3c 9f 00 03     addis       r4,r31,0x3
00021324 7f 85 e3 78     or          r5,r28,r28
00021328 38 84 30 50     addi        r4=>__cstring:s_'%s'_is_an_invalid_
0002132c 48 00 00 30     b           LAB_0002135c
```

Decompile: UndefinedFunction_00020aec - (MachO-OSX-ppc-openssl-1.0.1h)

```
108   if ((lVar11 == 0) || (lVar5 = _SSL_new((ulonglong)_ctx), lVar5 == 0)) goto LAB_00021588;
109   if (_s_tlsextdebug != 0) {
110     _SSL_callback_ctrl(lVar5,0x38,ZEXT48(PTR__tlsext_cb_000603ac));
111     _SSL_ctrl(lVar5,0x39,0,ZEXT48(_bio_s_out));
112   }
113   if ((char *)uParm3 != (char *)0x0) {
114     sVar12 = _strlen((char *)uParm3);
115     _SSL_set_session_id_context(lVar5,uParm3,sVar12);
116   }
117   append = _BIO_new_socket(iParm2,0);
118   if (_s_nbio_test != 0) {
119     type = _BIO_f_nbio_test();
120     b = _BIO_new(type);
121     append = _BIO_push(b,append);
122   }
123   _SSL_set_bio(lVar5,append,append);
124   _SSL_set_accept_state(lVar5);
125   _BIO_ctrl(bp_00,0x6d,1,(void *)lVar5);
126   _BIO_push(bp,bp_00);
127   if (_s_debug != 0) {
128     _SSL_set_debug(lVar5,1);
129     bp_00 = (BIO *)_SSL_get_rbio(lVar5);
130     _BIO_set_callback(bp_00,(long (*)(bio_st *,int,char *,int,long,long))
131                             PTR__bio_dump_callback_00060398);
132     bp_00 = (BIO *)_SSL_get_rbio(lVar5);
133     _BIO_set_callback_arg(bp_00,(char *)_bio_s_out);
134   }
135   if (_s_msg != 0) {
136     _SSL_set_msg_callback(lVar5,ZEXT48(PTR__msg_cb_000603a8));
137     _SSL_ctrl(lVar5,0x10,0,ZEXT48(_bio_s_out));
138   }
139 LAB_00020d80:
140   do {
141     if (_hack != 0) {
142       while ((uVar6 = _SSL_accept(lVar5), (int)uVar6 < 1 &&
143               (iVar10 = _SSL_get_error(lVar5,uVar6), iVar10 == 4))) {
144         _BIO_printf(_bio_s_out,"LOOKUP during accept %s\n",(ulonglong)_srp_callback_parm);
145         DAT_000618e0 = _SRP_VBASE_get_by_user((ulonglong)DAT_000618dc,(ulonglong)_srp_callback_parm)
146         ;
147         if (DAT_000618e0 == 0) {
148           _BIO_printf(_bio_s_out,"LOOKUP not successful\n");
149         }
150         else {
```

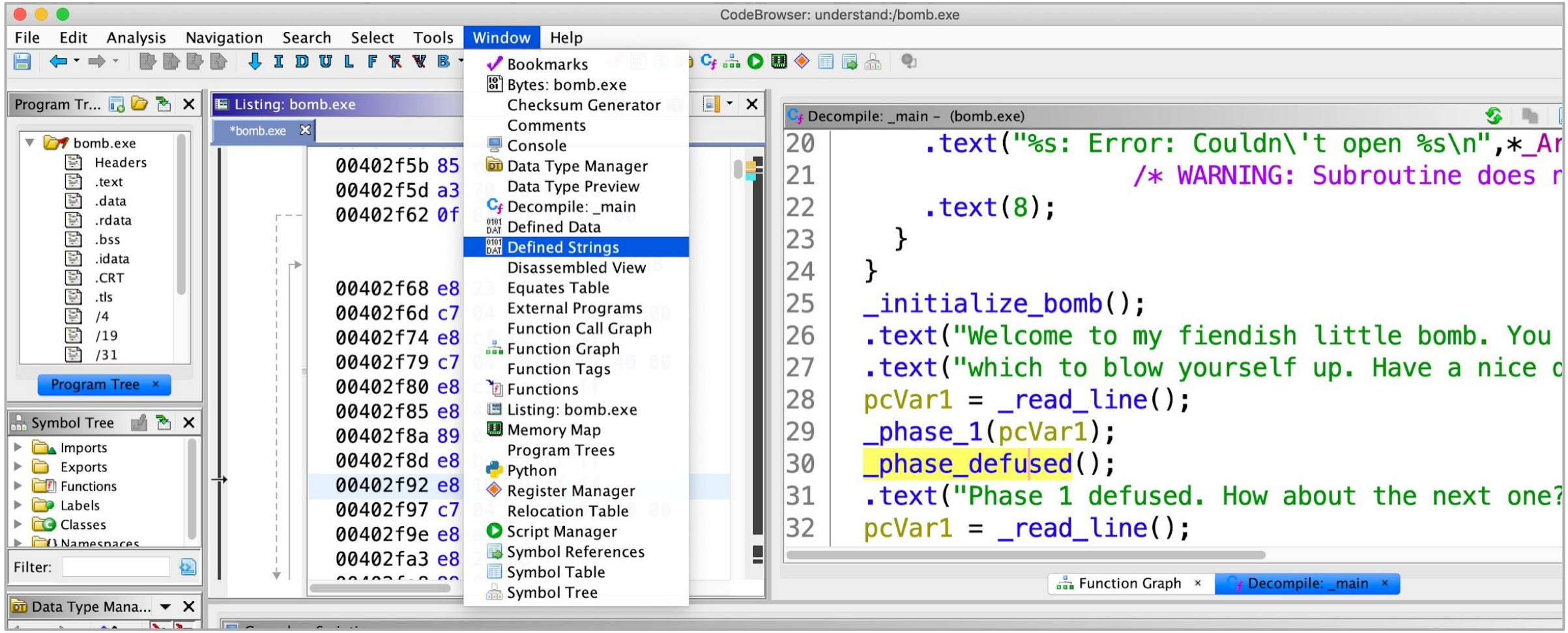
Cross-References

Ca' Foscari University of Venice



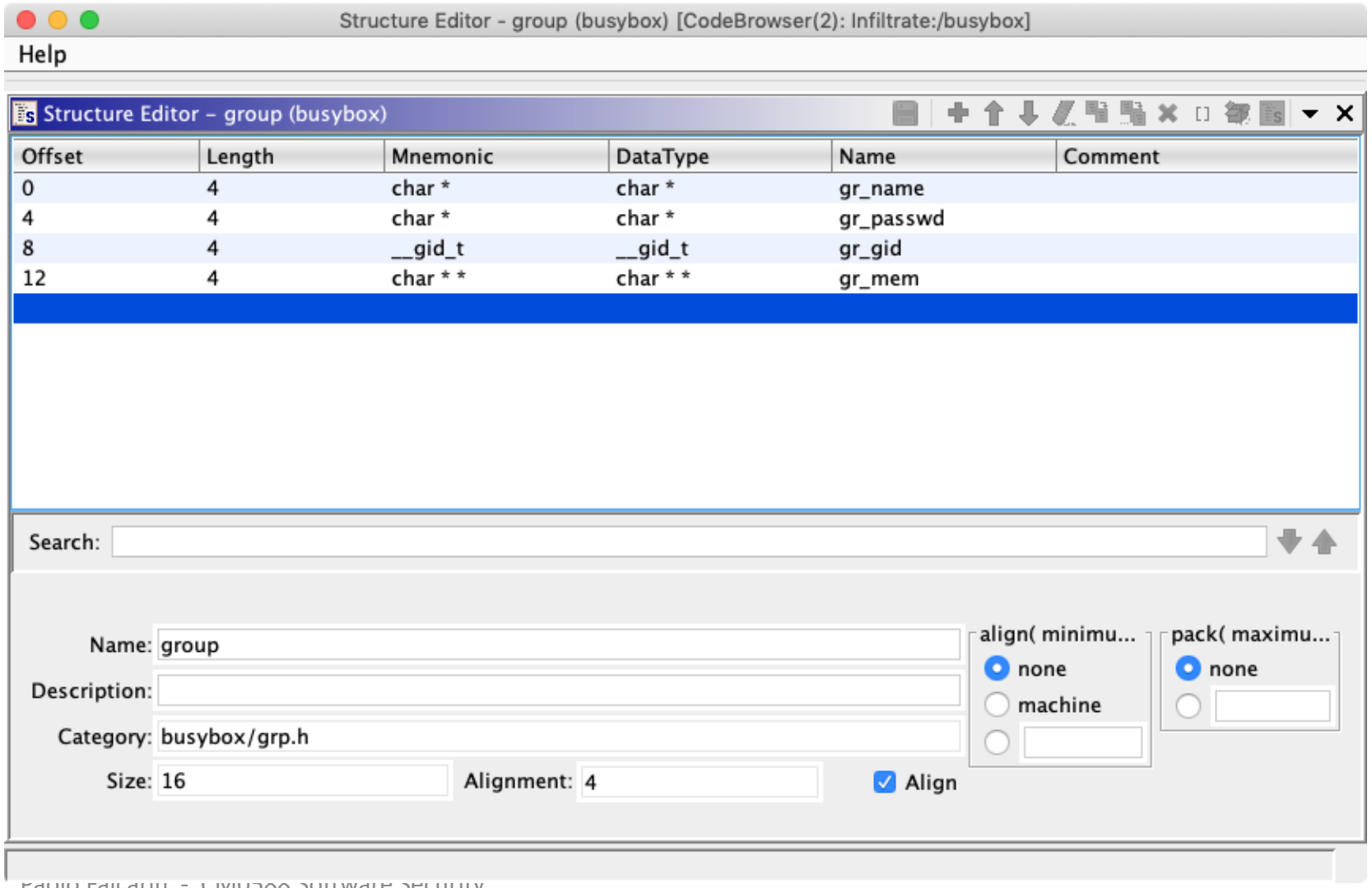
Strings

Ca' Foscari University of Venice



Type System / Structure Recovery

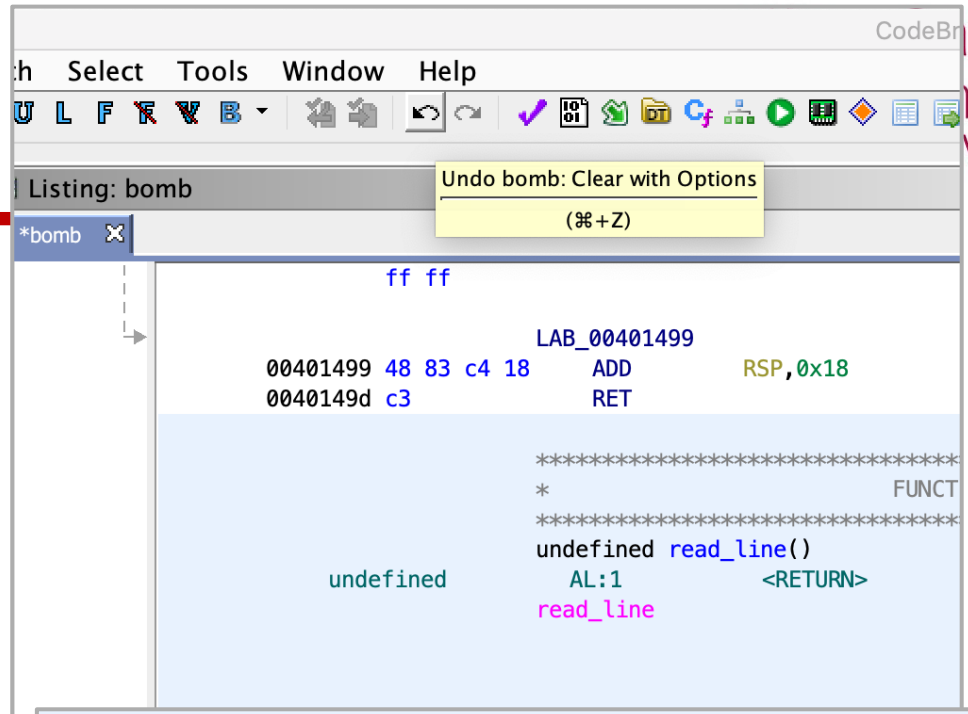
Ca' Foscari University of Venice



The Undo Button!

```
0040149e 48 83 ec 08      SUB         RSP,0x8
004014a2 b8 00 00         MOV         EAX,0x0
         00 00
004014a7 e8 4d ff         CALL        skip
         ff ff
004014ac 48 85 c0         TEST        RAX,RAX
004014af 75 6e            JNZ         LAB_0040151f
004014b1 48 8b 05         MOV         RAX,qword ptr [.bss:stdin]
         90 22 20 00
004014b8 48 39 05         CMP         qword ptr [.bss:infile],RAX
         a9 22 20 00
004014bf 75 14            JNZ         LAB_004014d5
```

```
0040149e 48               ??          48h    H
0040149f 83               ??          83h
004014a0 ec               ??          ECh
004014a1 08               ??          08h
004014a2 b8 00 00         MOV         EAX,0x0
         00 00
004014a7 e8 4d ff         CALL        skip
         ff ff
004014ac 48 85 c0         TEST        RAX,RAX
004014af 75 6e            JNZ         LAB_0040151f
004014b1 48 8b 05         MOV         RAX,qword ptr [.bss:stdin]
         90 22 20 00
004014b8 48 39 05         CMP         qword ptr [.bss:infile],RAX
         a9 22 20 00
004014bf 75 14            JNZ         LAB_004014d5
004014c1 bf d5 25         MOV         EDI=>.rodata:s_Error:_Premat
         40 00
004014c6 e8 45 f6         CALL        .plt:puts
```




```
0040149e 48 83 ec 08      SUB         RSP,0x8
004014a2 b8 00 00         MOV         EAX,0x0
         00 00
004014a7 e8 4d ff         CALL        skip
         ff ff
004014ac 48 85 c0         TEST        RAX,RAX
004014af 75 6e            JNZ         LAB_0040151f
004014b1 48 8b 05         MOV         RAX,qword ptr [.bss:stdin]
         90 22 20 00
004014b8 48 39 05         CMP         qword ptr [.bss:infile],RAX
         a9 22 20 00
004014bf 75 14            JNZ         LAB_004014d5
004014c1 bf d5 25         MOV         EDI=>.rodata:s_Error:_Premat
         40 00
004014c6 e8 45 f6         CALL        .plt:puts
```

Static Attack with Ghidra

Ghidra set-up

Ca' Foscari University of Venice

1. Install JDK (version >= 11), add its `bin` directory to your `PATH`

   `jdk.java.net/11`

2. Download Ghidra

   `ghidra-sre.org`

   `github.com/NationalSecurityAgency/ghidra/releases`

3. The installation instructions are here:

   `https://ghidra-sre.org/InstallationGuide.html#Platforms`

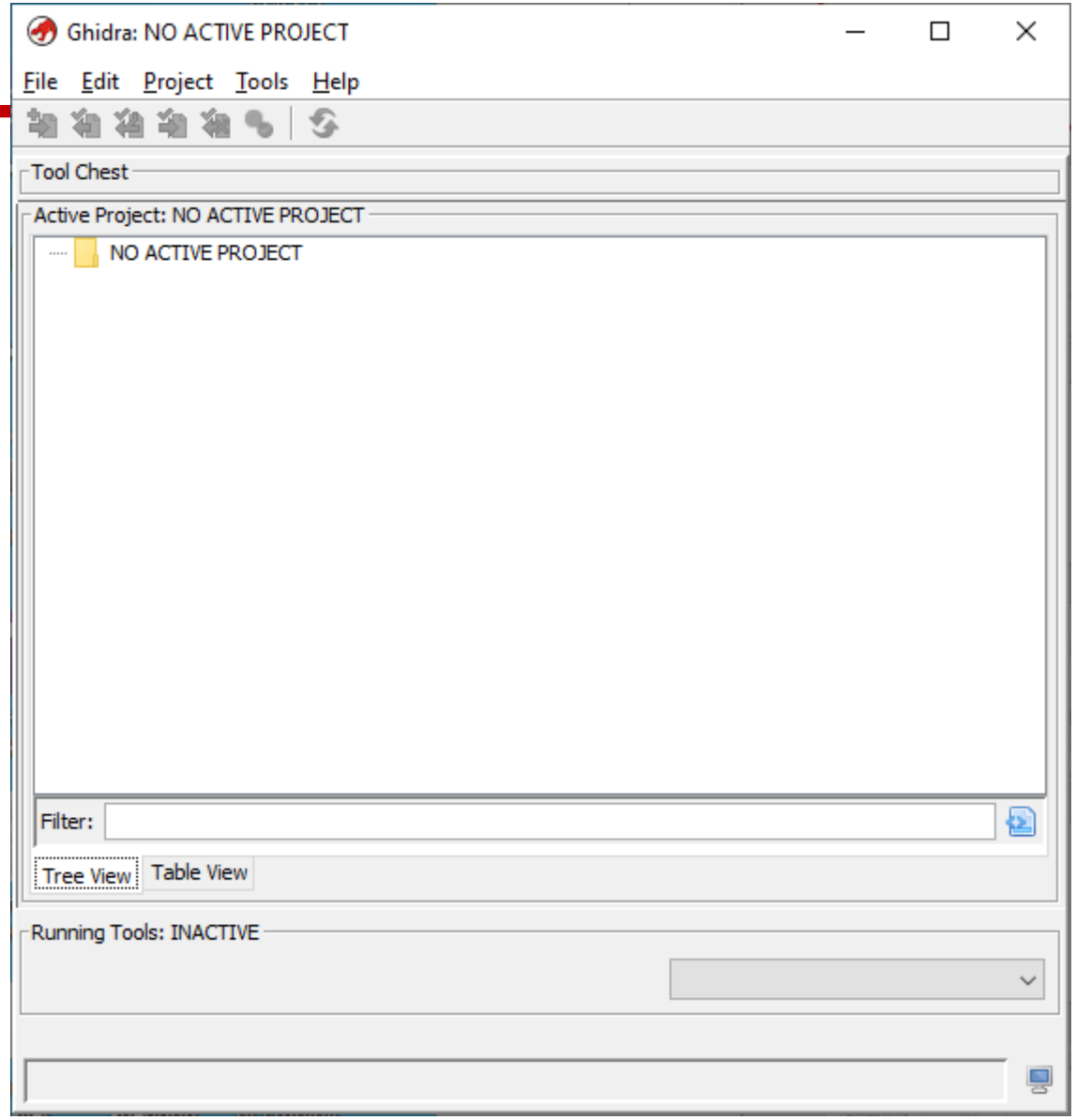
4. Run Ghidra

   `ghidraRun.bat ( or ghidra.sh)`

MacOS: Make sure you start ghidra from bash shell, not zsh shell (which is now default on MacOS)

Ghidra

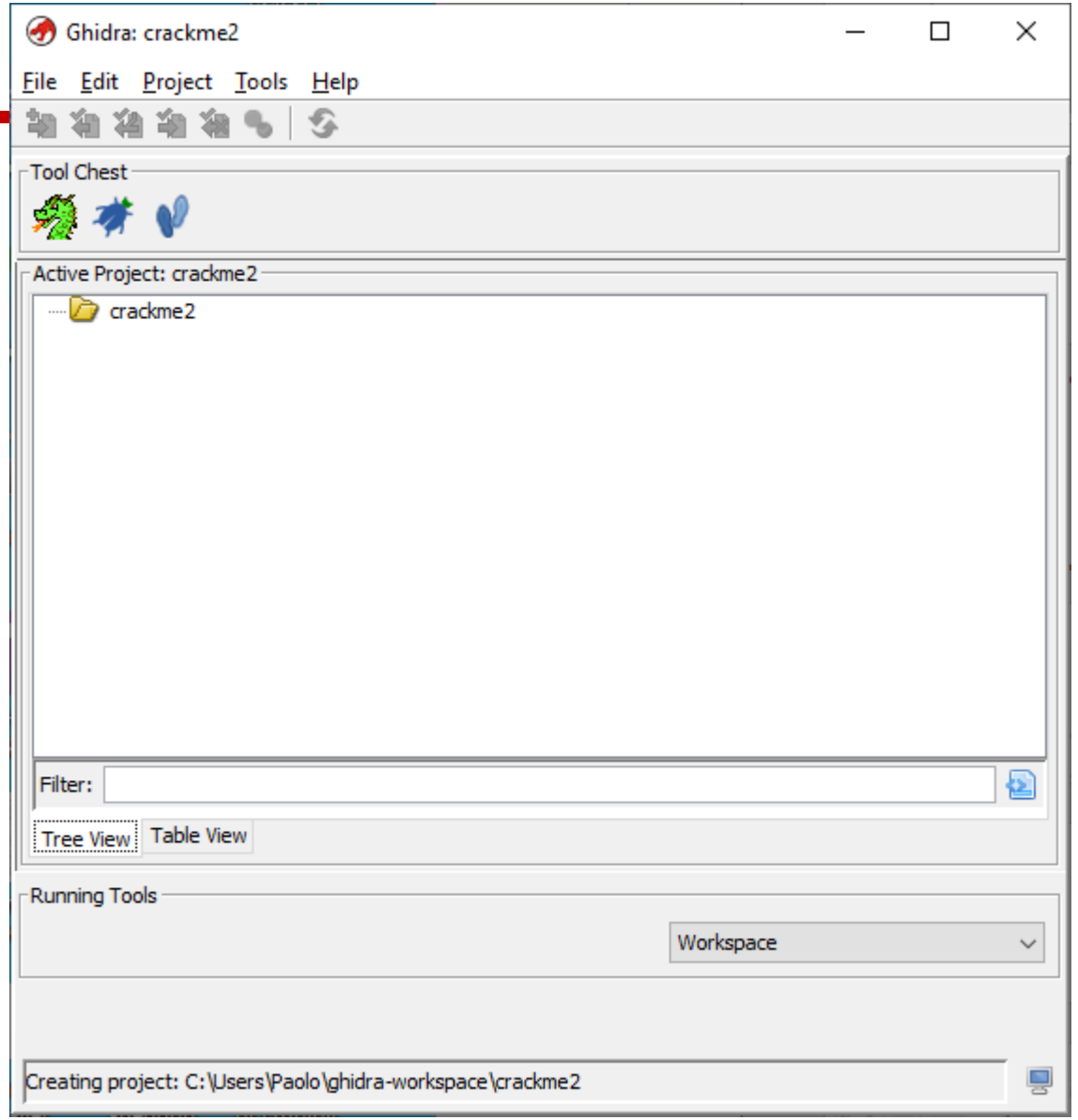
- Open Ghidra



Ca' Foscari University of Venice

New Project

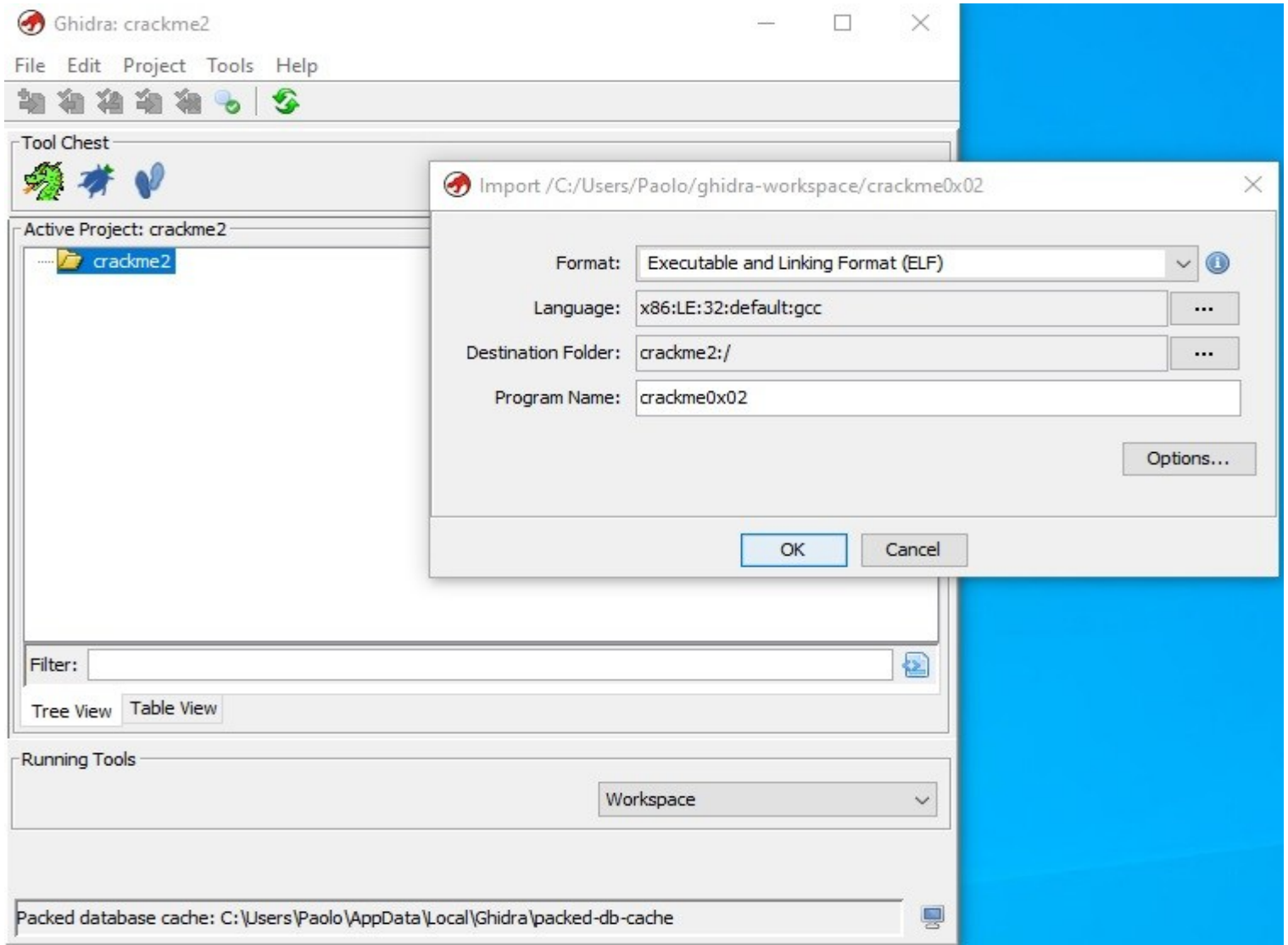
- Not-shared
- Name it “Crackme2”



Ca’ Foscari University of Venice

Import Program in project

- Menu File
- Import File
- Browse and select Crackme0x02

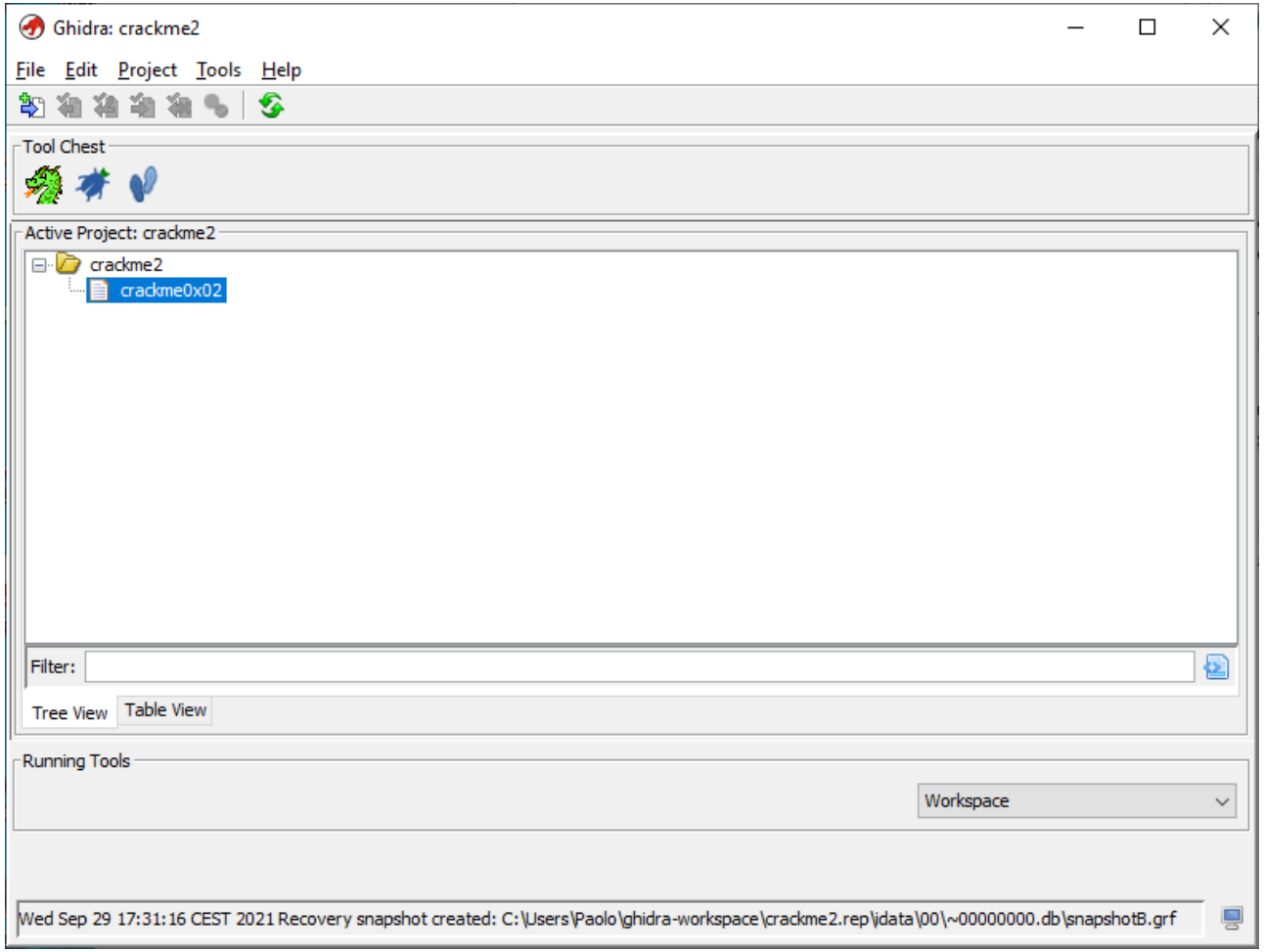


Launch CodeBrowser

Ca' Foscari University of Venice

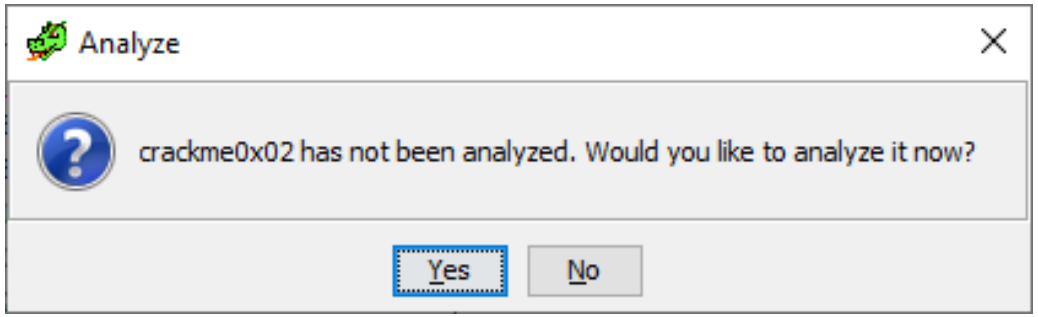
- Click Ghidra

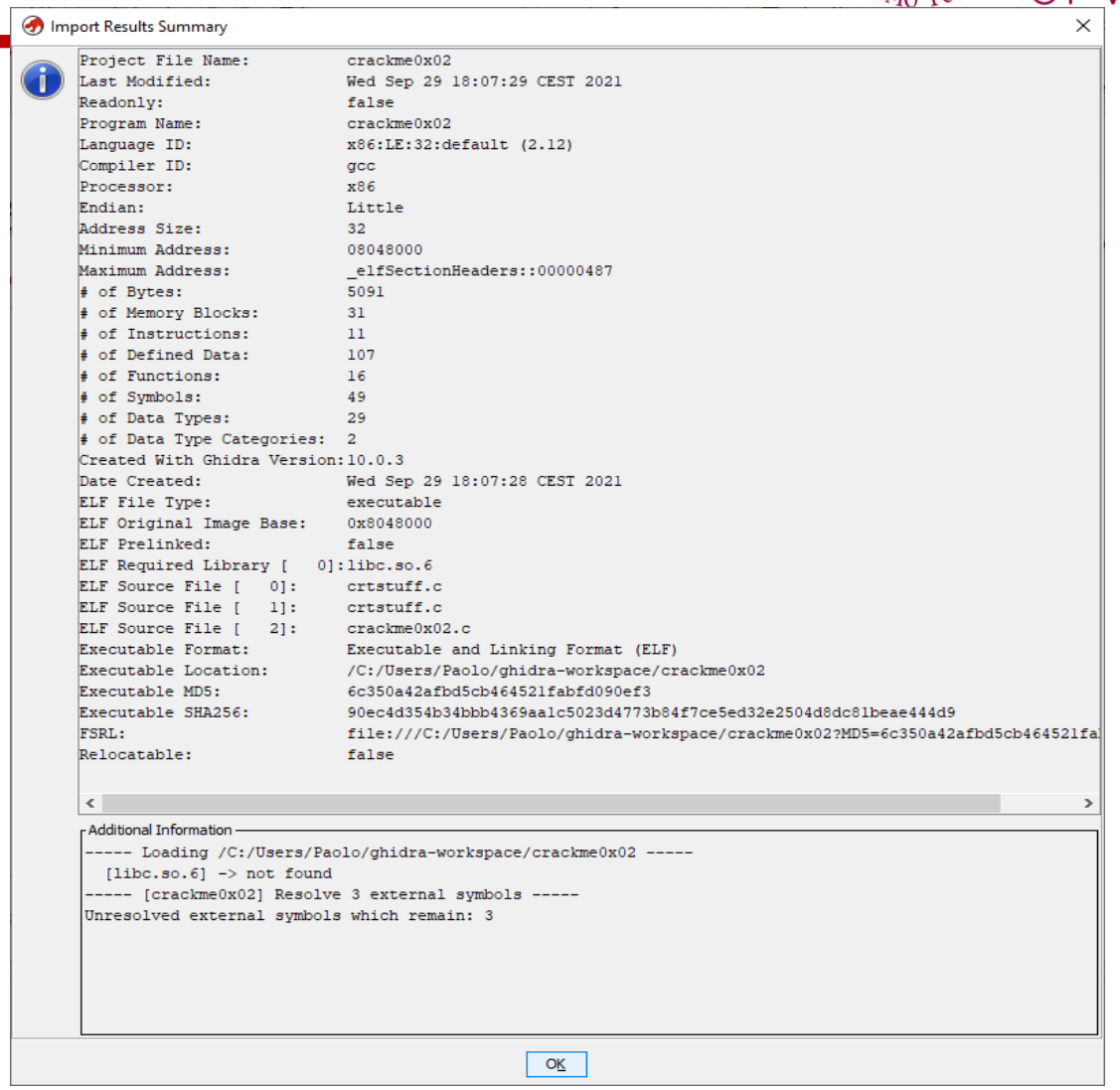
(the green dragon icon)
to open the CodeBrowser



About Program

- Info on the file
- Then Prompt to analyse it

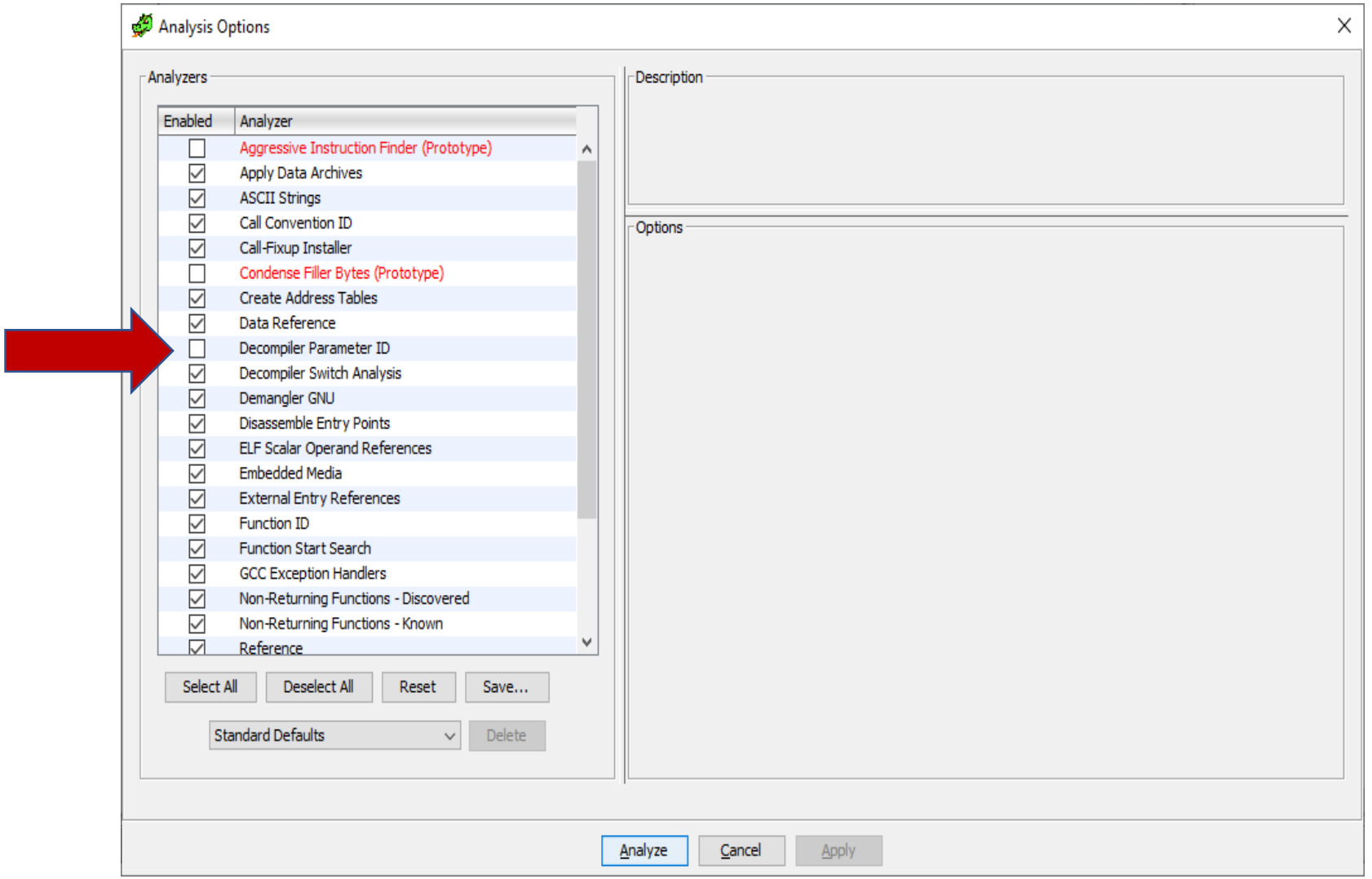




Analysis Options: Decompiler

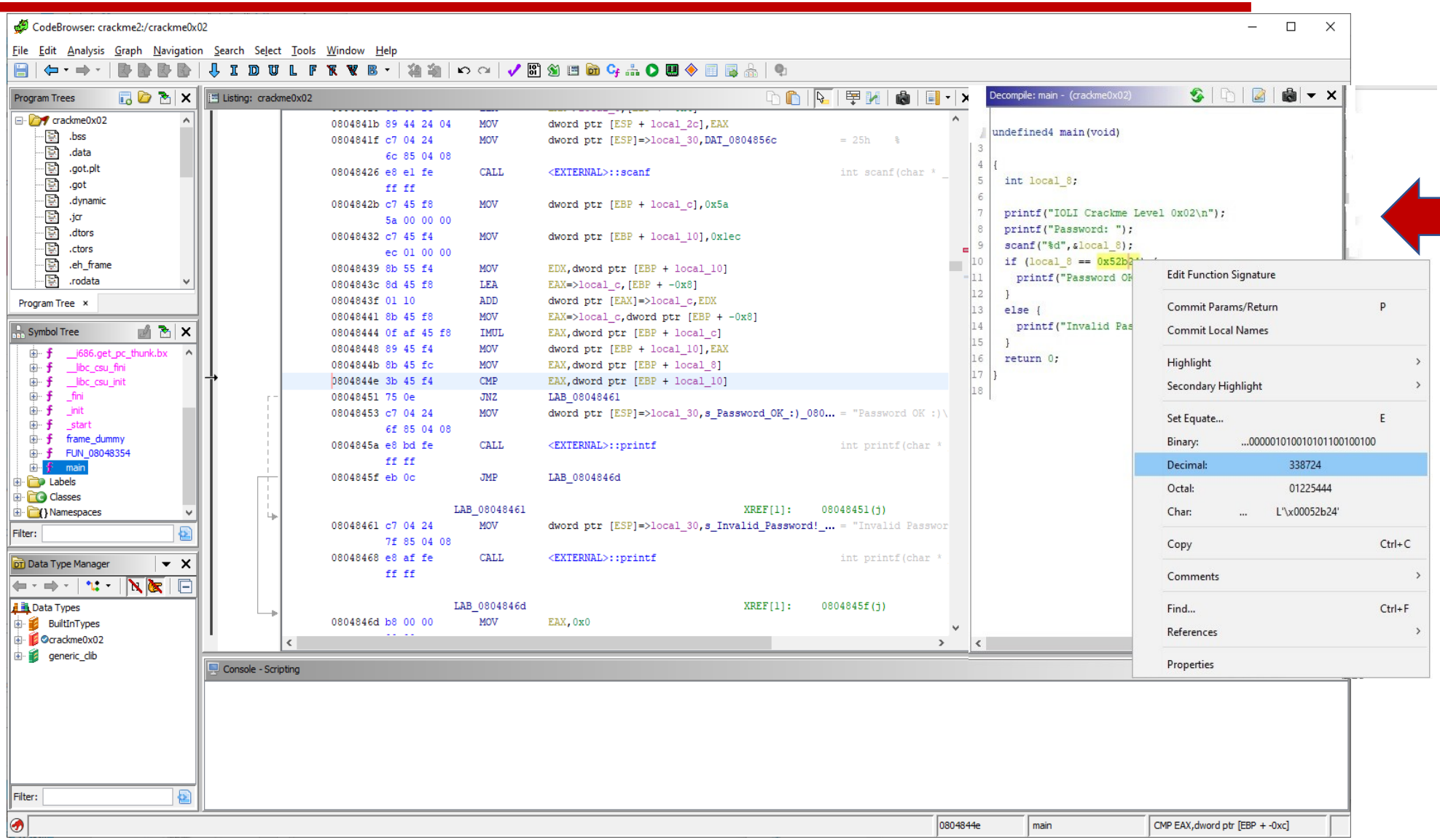
Ca' Foscari University of Venice

- Select Decompiler Parameter ID
- Then in the Functions View, select the "main" function



CodeBrowser: Crackme0x02

Ca' Foscari University of Venice

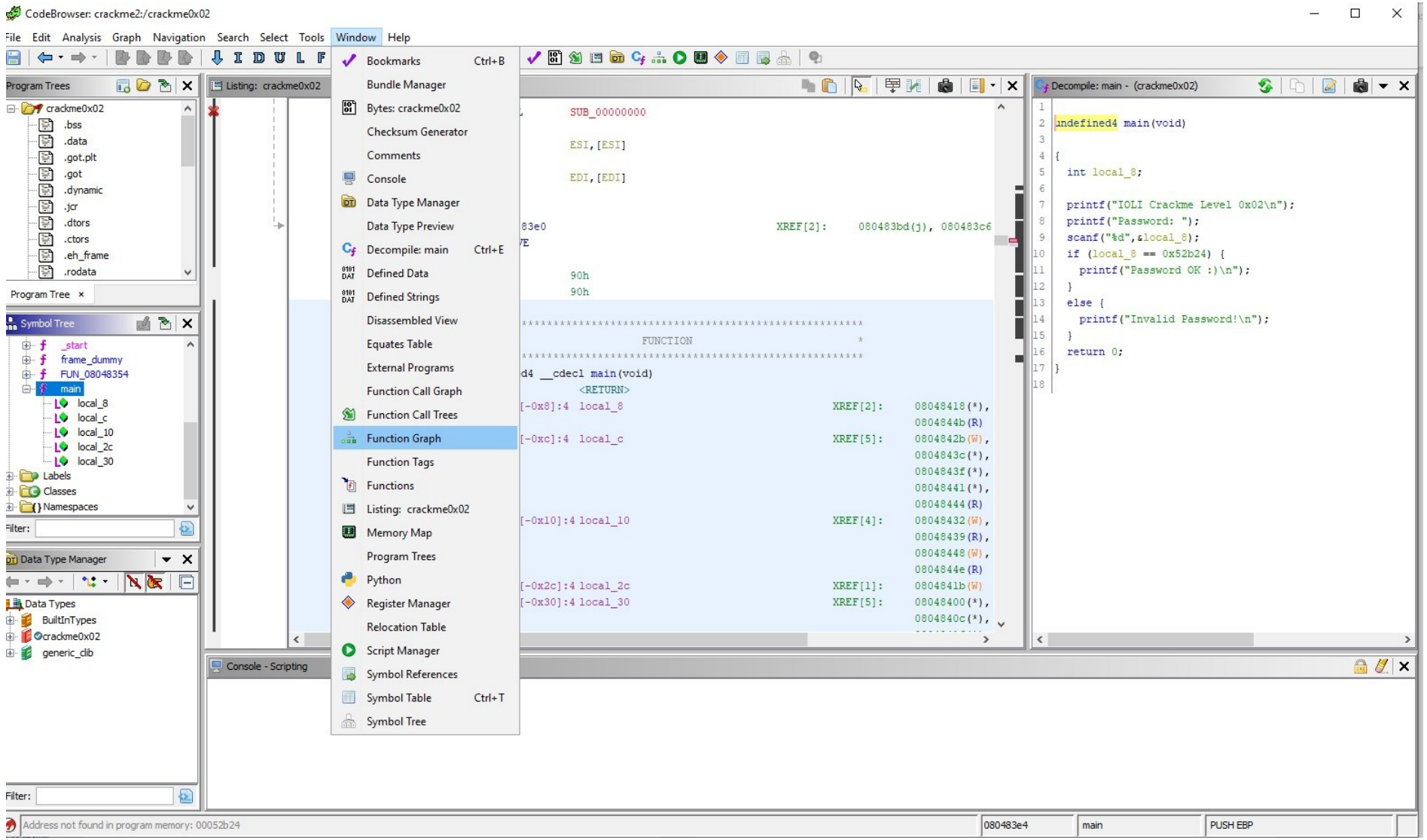


Decompiled C code here

Right-Click on constants to see the values in various bases or char encoding

Select a function and create its CFG

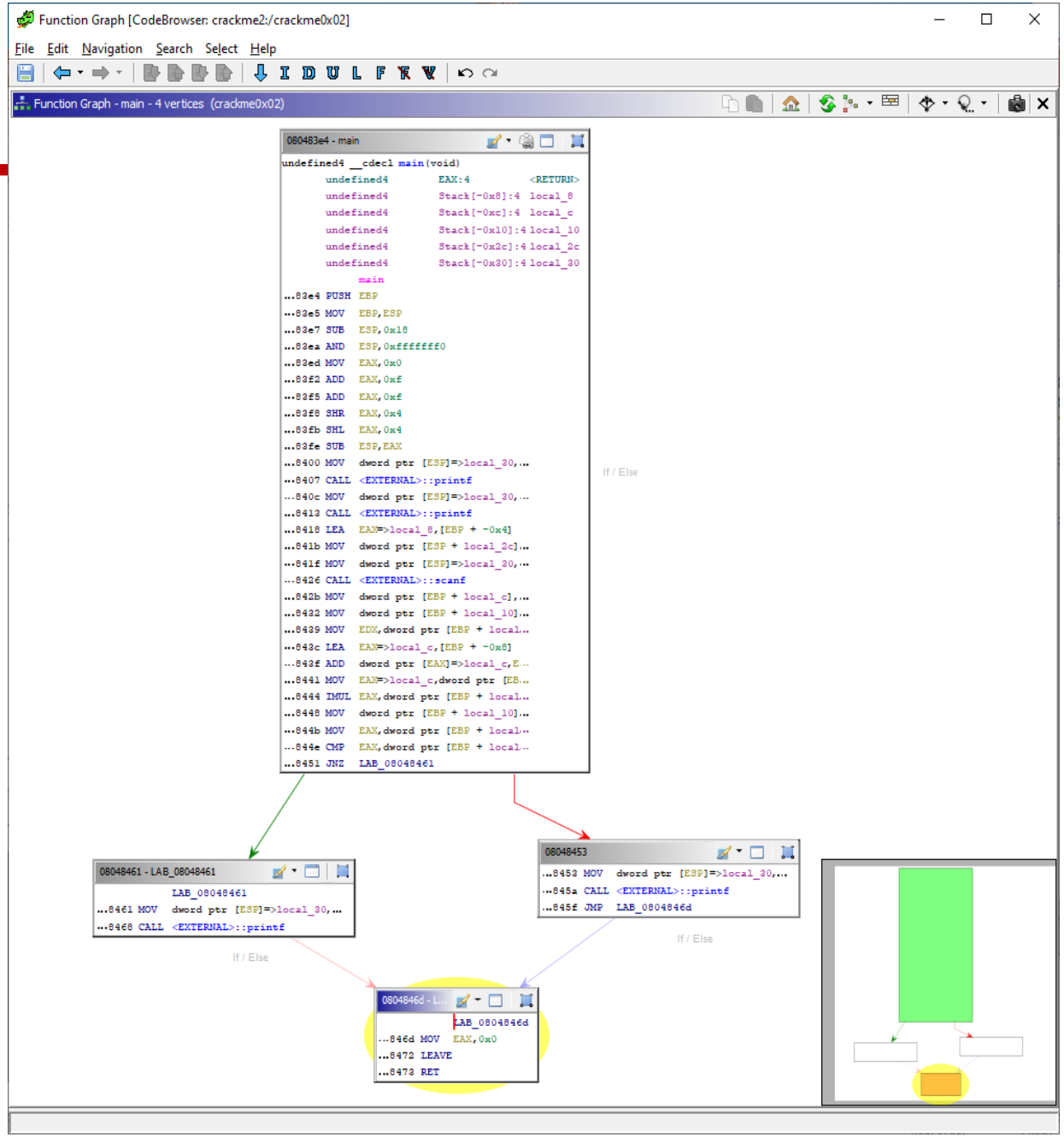
Ca' Foscari University of Venice



1. Select function
2. Menu Window
3. Function Graph

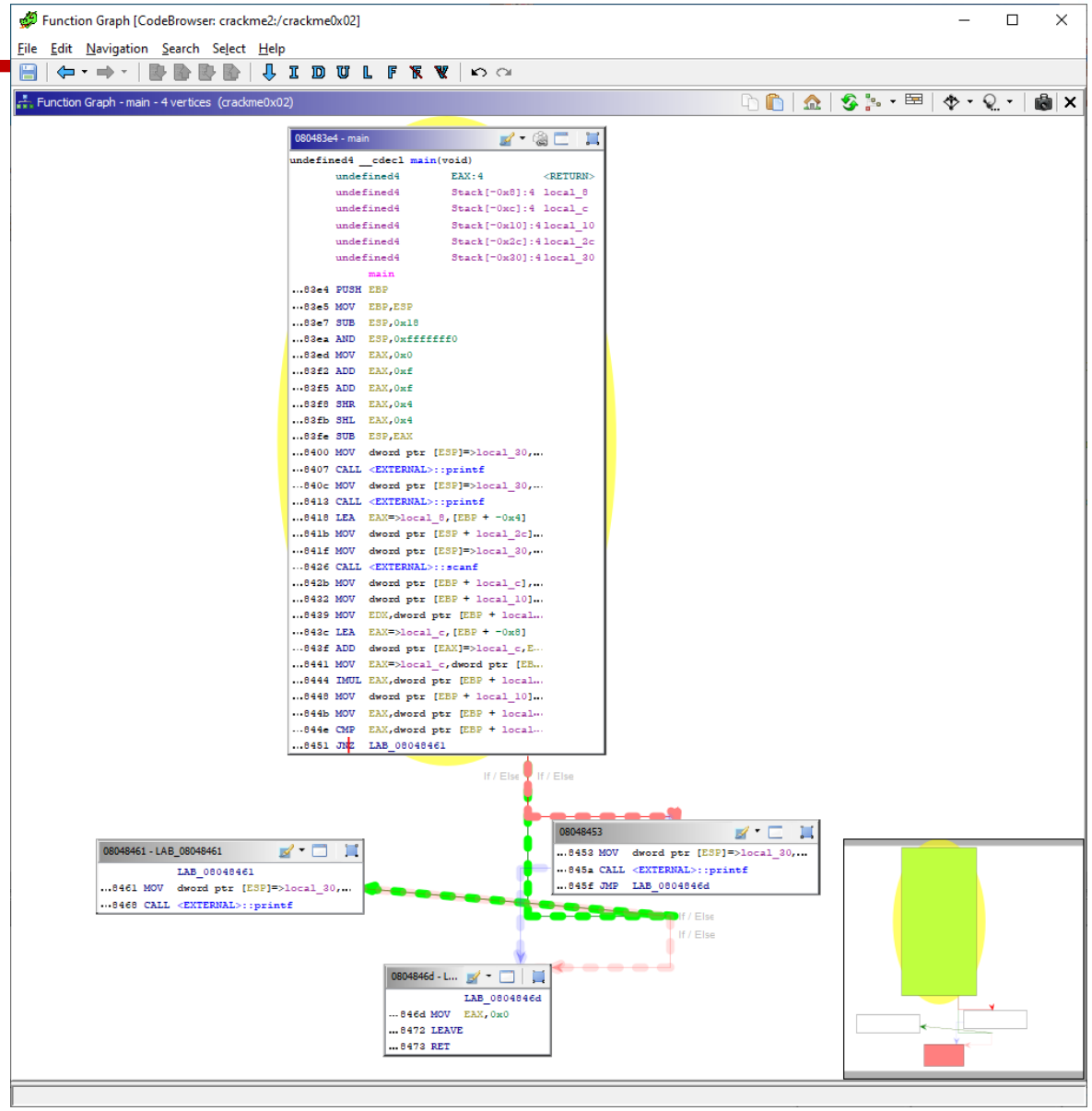
CFG

- If (green) else (red)
- View Outline
- Resizable Layout



Highlight Paths

- When selecting a jump instruction



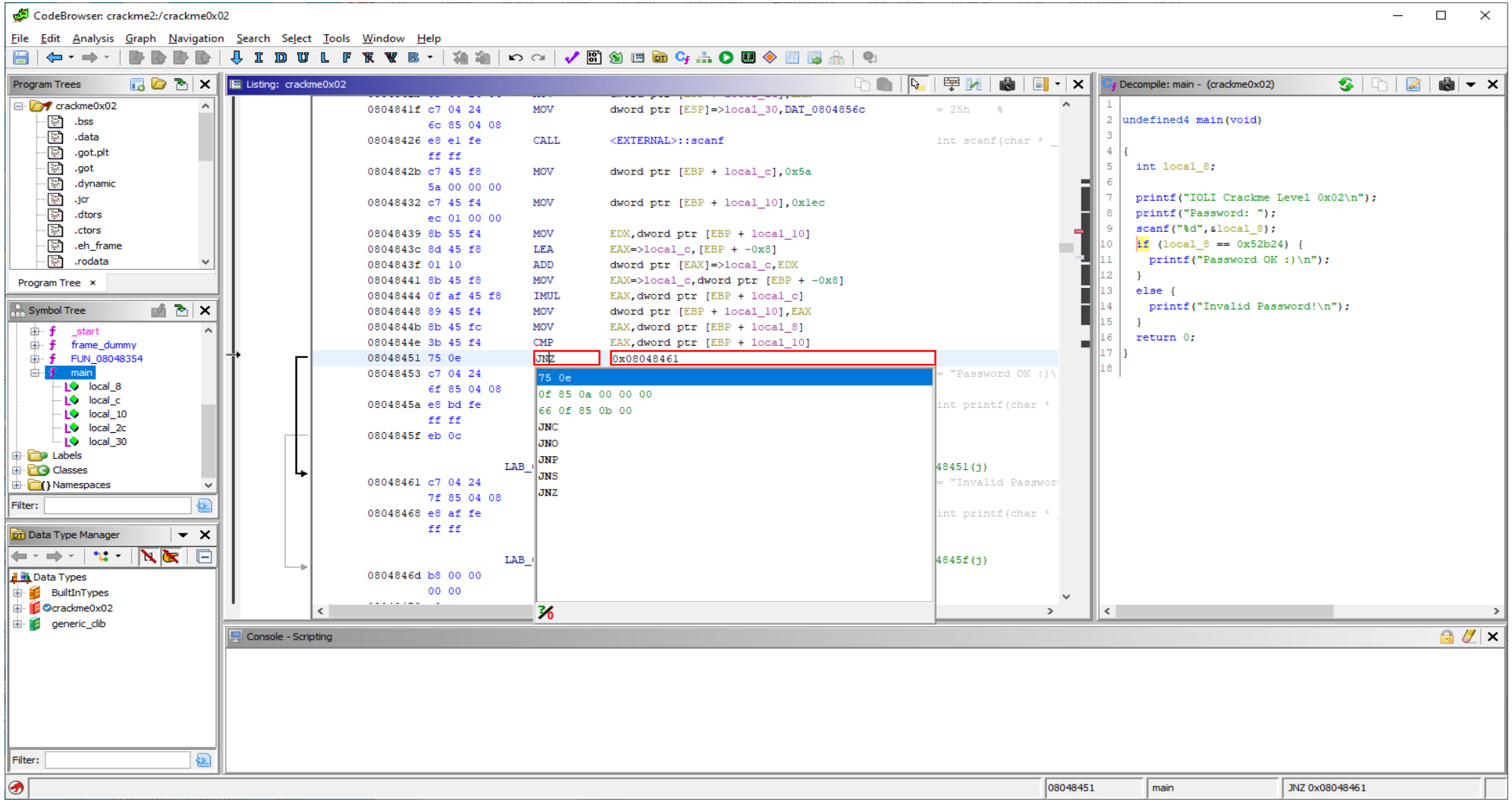
Jumps Recap

Ca' Foscari University of Venice

CCCC	Name	Means
0000	O	overflow
0001	NO	Not overflow
0010	C/B/NAE	Carry, below, not above nor equal
0011	NC/AE/NB	Not carry, above or equal, not below
0100	E/Z	Equal, zero
0101	NE/NZ	Not equal, not zero
0110	BE/NA	Below or equal, not above
0111	A/NBE	Above, not below nor equal
1000	S	Sign (negative)
1001	NS	Not sign
1010	P/PE	Parity, parity even
1011	NP/PO	Not parity, parity odd
1100	L/NGE	Less, not greater nor equal
1101	GE/NL	Greater or equal, not less
1110	LE/NG	Less or equal, not greater
1111	G/NLE	Greater, not less nor equal

Patch Instruction

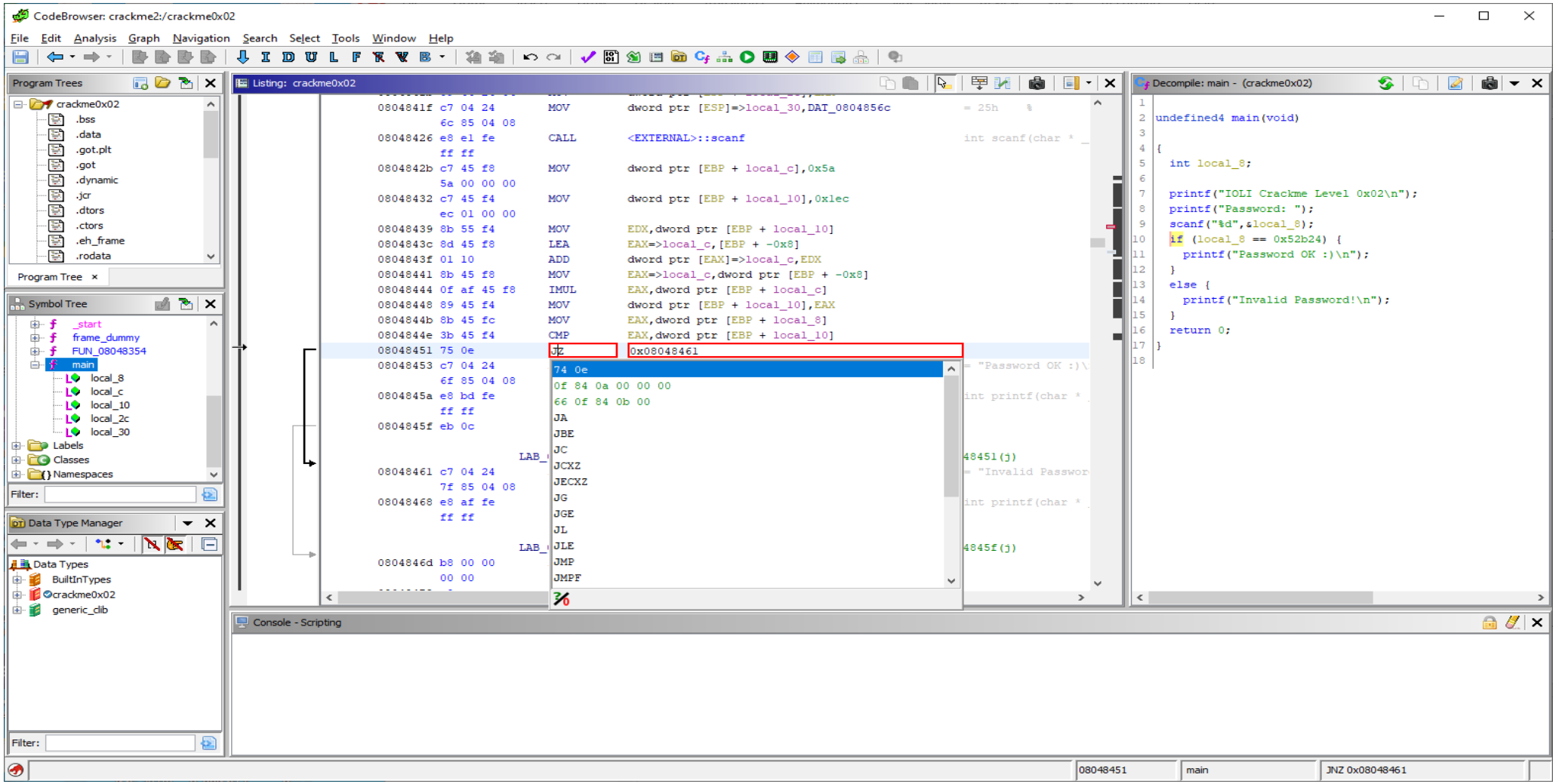
Ca' Foscari University of Venice



Edit Assembly Code

Ca' Foscari University of Venice

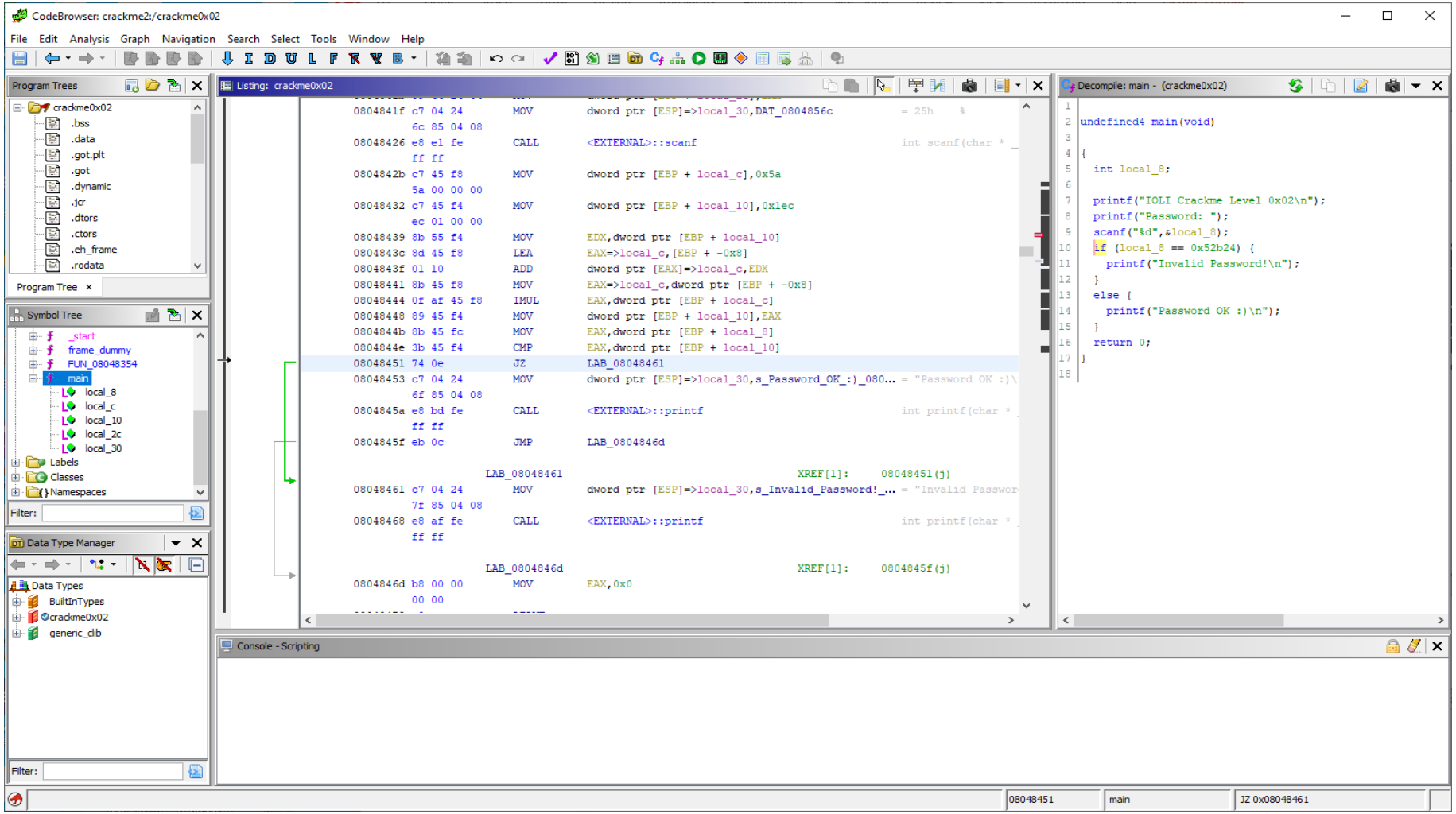
- Hit Enter to save the change



If else condition changed!

Ca' Foscari University of Venice

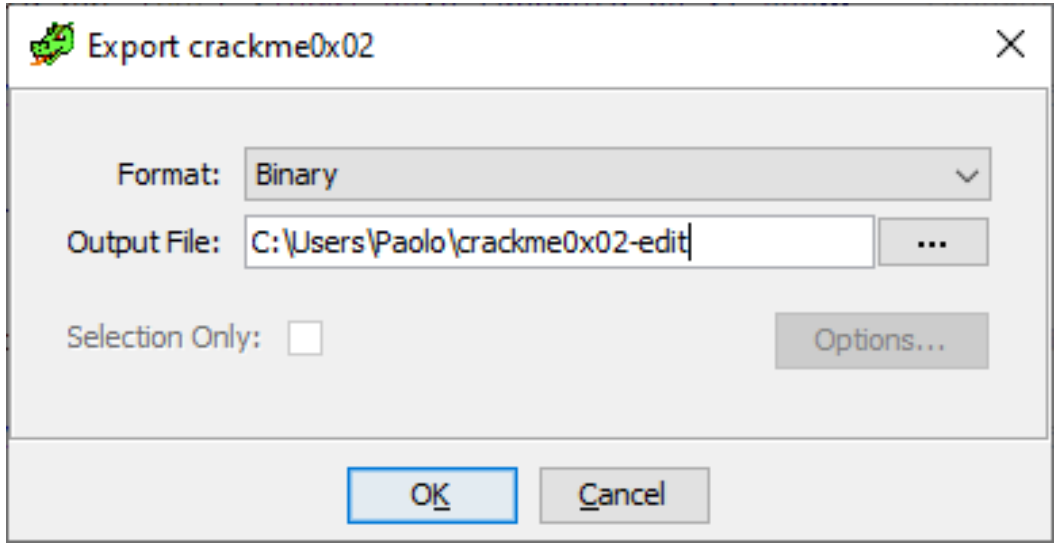
- Our first static attack on Ghidra!
- Opcode 75 (JNZ) is now 74 (JZ)

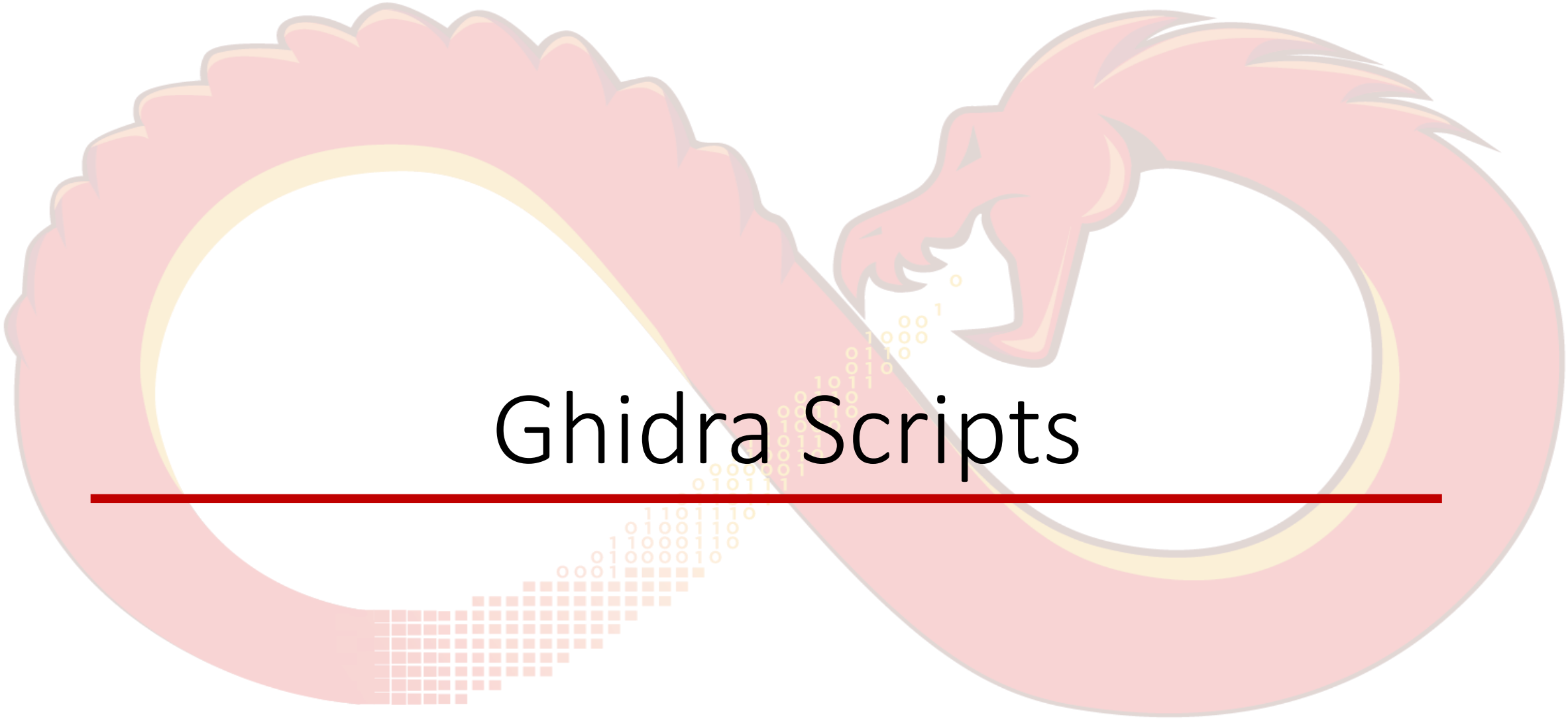


Export the edited binary to a new file

Ca' Foscari University of Venice

- Choose the Binary format
- In Linux make it executable
  - chmod a+rx crackme0x02-edit.bin
- Then run it and see the different behaviour!





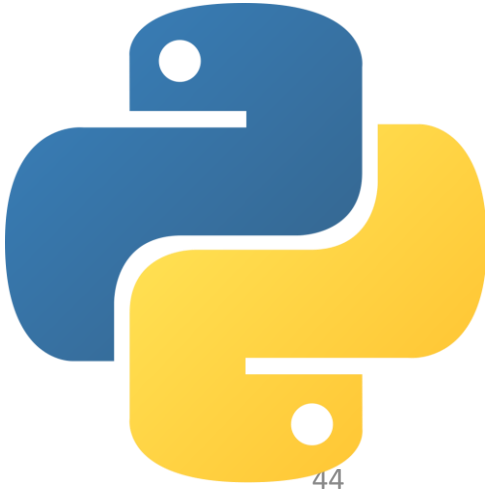
Ghidra Scripts

Script vs. Plugin vs. Extension

- Scripts: do a single thing with defined start and end point
- Plugins: base components for everything you interact with in Ghidra, such as UI panes
- Extensions: sets of plugins for extended functionality, e.g., custom binary format loaders, interfaces to external tools, libraries for use by other scripts

Scripting With Ghidra

- Available in Java (natively) and Python (via Jython)
- Can be run with interactive GUI or in headless mode
- Ghidra comes with 230+ scripts pre-installed
  - Educational examples
  - Code patching
  - Import / export
- Scripts for Analysis enhancements
  - Windows, Mac, Linux, VXWorks
  - PE, ELF, Mach-O, COFF
  - x86, MIPS, ARM/THUMB, 8051, etc…

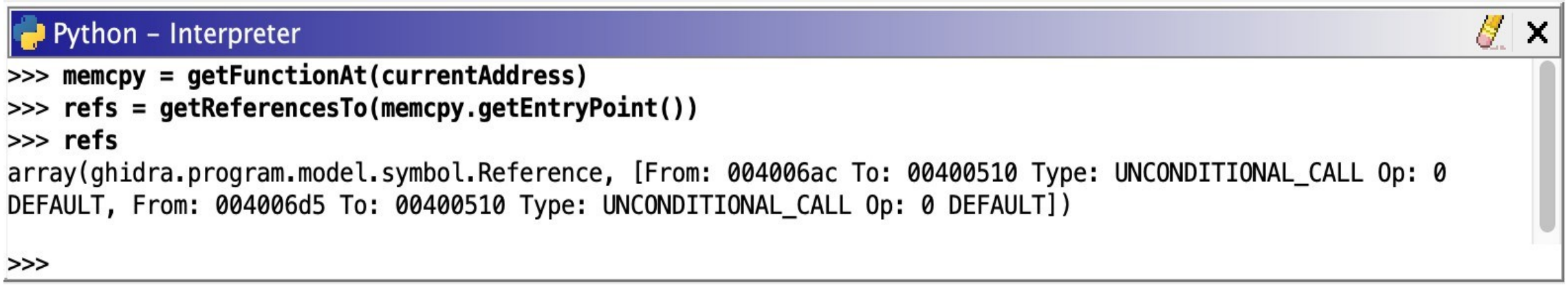


Python Interpreter Window

Ca' Foscari University of Venice

- Unlike Java, which must be compiled in order to run, Python can be run inside Ghidra in an interactive REPL shell

- The shell can be helpful for exploring unfamiliar objects - Ghidra has great Python object str implementations



Ghidra APIs

Ca' Foscari University of Venice

- FlatProgramAPI
  - Simple "flattened" API for Ghidra scripting
  - Programmatic access to common tasks
    - query / modify / iterate / create / delete - functions / data / instructions / comments
  - Mostly doesn't require the use of Java objects
  - Stable

Ghidra Program API

- More complex rich API for deeper scripting
- Object-oriented (Program, Memory, Function, Instruction, etc…)
- Utility functions help with common scripting tasks
- UI scripting / interactivity
- Prone to change between versions

API Highlights

Ca' Foscari University of Venice

Rich Scripting Interface

- Programmatic access to binary file formats
- P-code interaction
- Decompiler API
- C header parsing
- Interface for graphing (implementation not included)
- Cyclomatic complexity

Common Utilities Included

- UI windows
- Assembly
- Data serialization
- String manipulation
- Hashing
- Search / byte matching
- XML utilities

Eclipse Integration

Ca' Foscari University of Venice

- Ghidra has built-in Eclipse integration, via its "GhidraDev" Eclipse plugin
- NOTE: We'll be using Ghidra's built-in basic editor – no need for it now

Scripting - Java vs Python

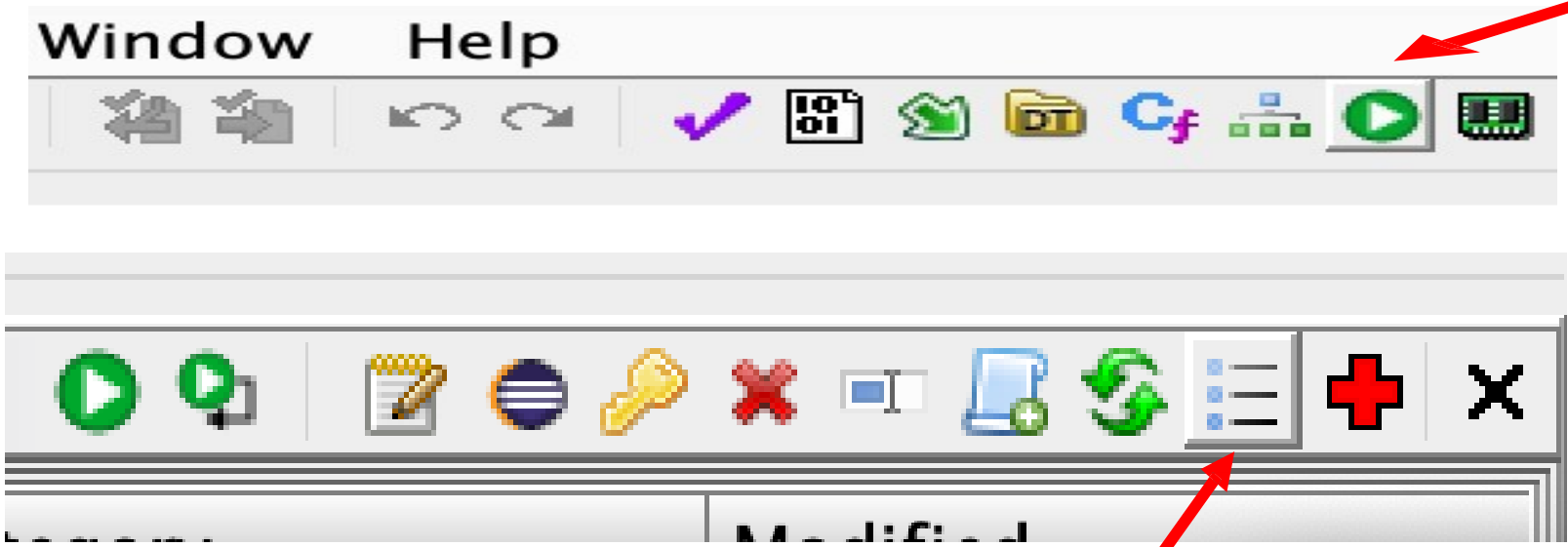
Ca' Foscari University of Venice

- Java will catch errors at compile time, Ghidra's API is highly object-oriented and benefits from this
- Complex Python scripts feel like binding together Java API calls with Python control flow and syntax
- Recommended workflow: prototype and experiment with APIs / objects in the Python interpreter, write final code in Java

Jython

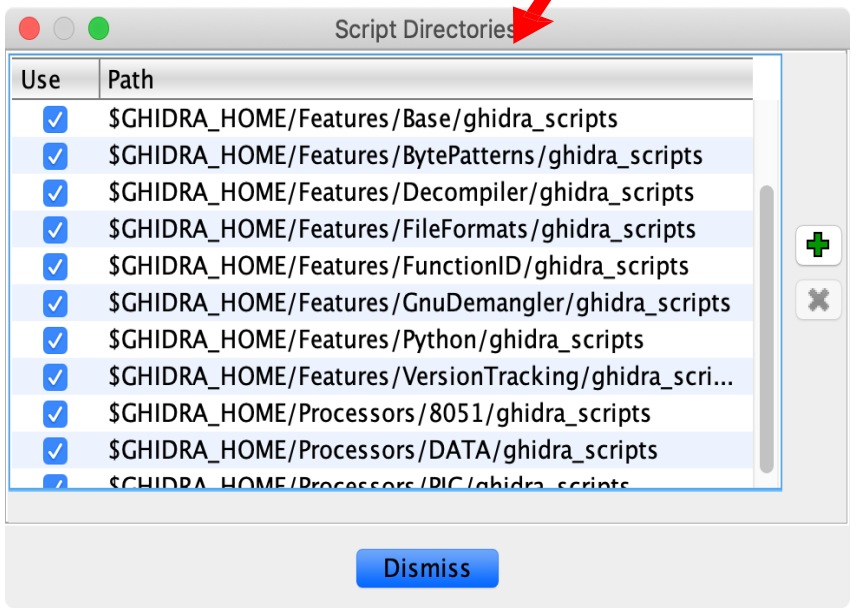
Importing Demo Scripts

Ca' Foscari University of Venice



Click the "Display Script Manager" button to open the Script Manager Window

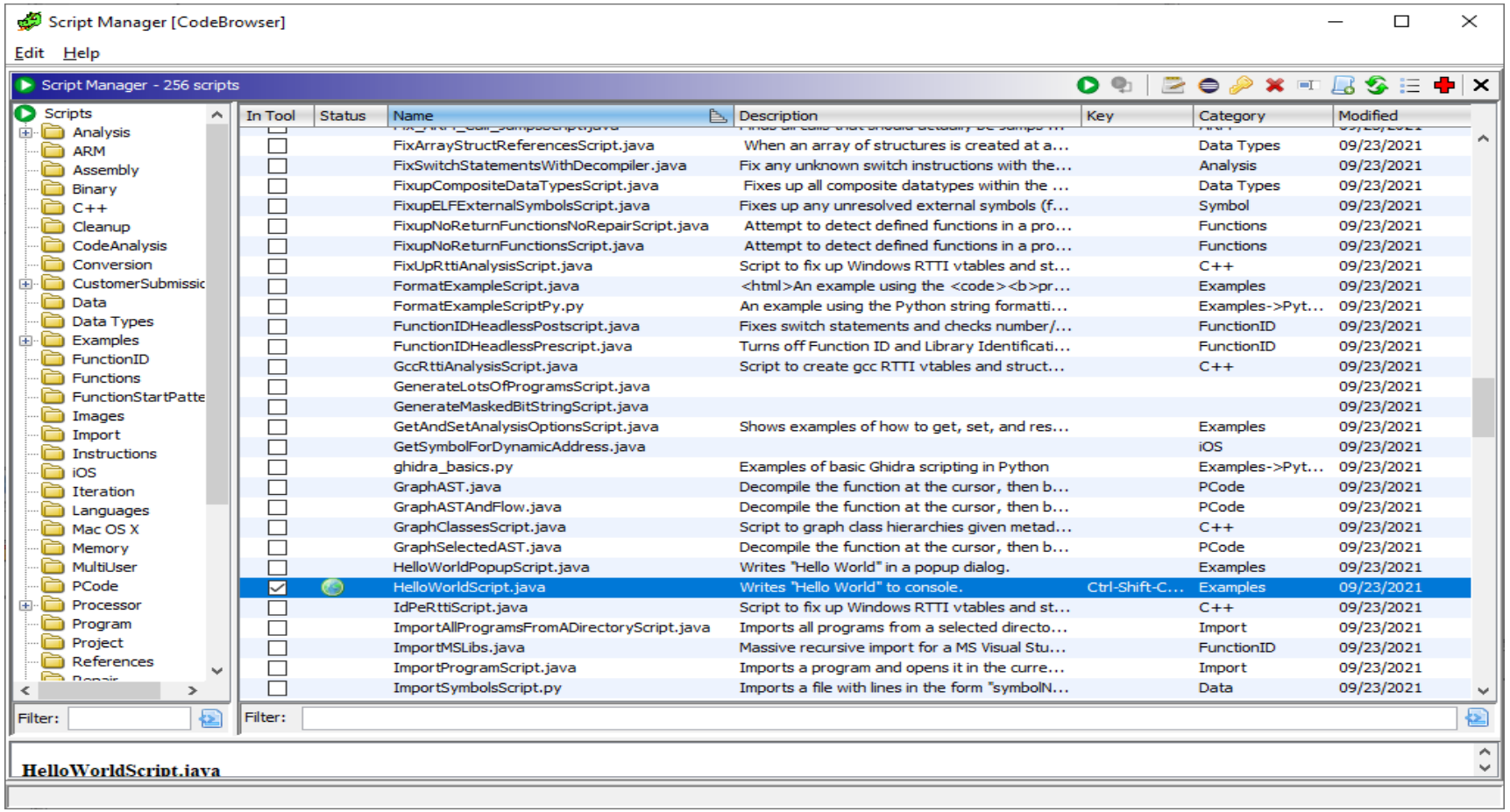
Click the green plus to open the file chooser, choose the script directory



Click the "Script Directories" button on the Script Click the "Display Script Manager" button to open the Script Manager Window

HelloWorld Script

Ca' Foscari University of Venice



Console – Scripting Output

HelloWorldScript.java> Running...

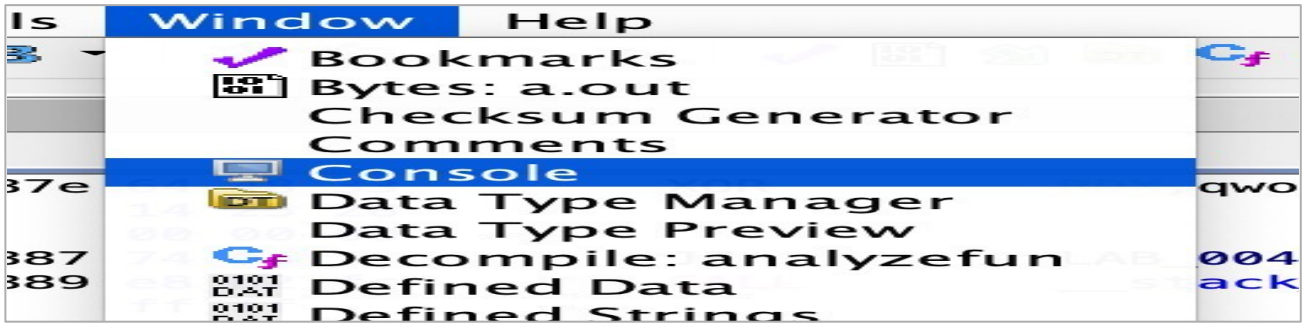
HelloWorldScript.java> Hello World

HelloWorldScript.java> Finished!

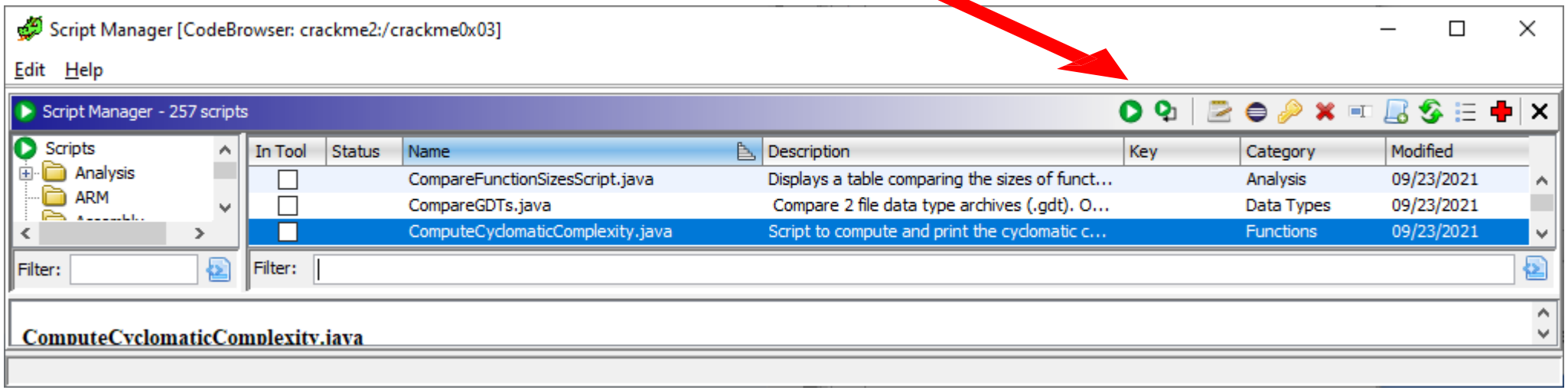
Running Script Demos

Ca' Foscari University of Venice

Make sure the "Console" window is open if you want to see output



Choose a script ComputeCyclomaticComplexity and click "Run Script" to run



Calculating Cyclomatic Complexity

Ca' Foscari University of Venice

- Cyclomatic complexity is a measure of the number of unique paths which may be taken through a given function.
- The cyclomatic complexity M is then defined as

$$M = E - N + 2P$$

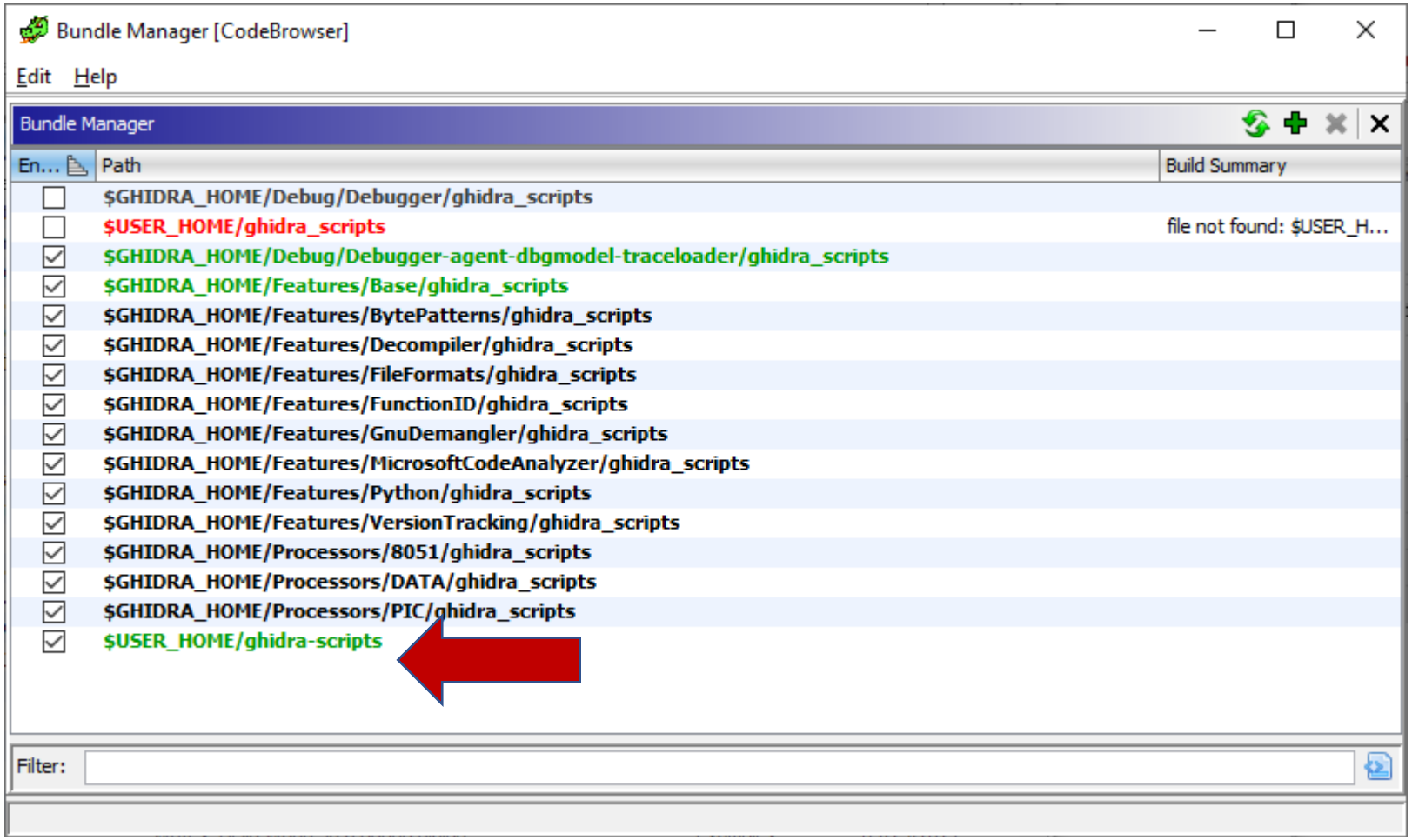
where

- E = the number of edges of the graph.
- N = the number of nodes of the graph.
- P = the number of connected components in the Control Flow Graph

- Leverage Ghidra's API for calculating cyclomatic complexity to easily analyze a whole program
- It can be helpful in finding complex functions likely to have vulnerabilities, e.g. complex parsing routines or state machines.

Add Your Script Folder

Ca' Foscari University of Venice

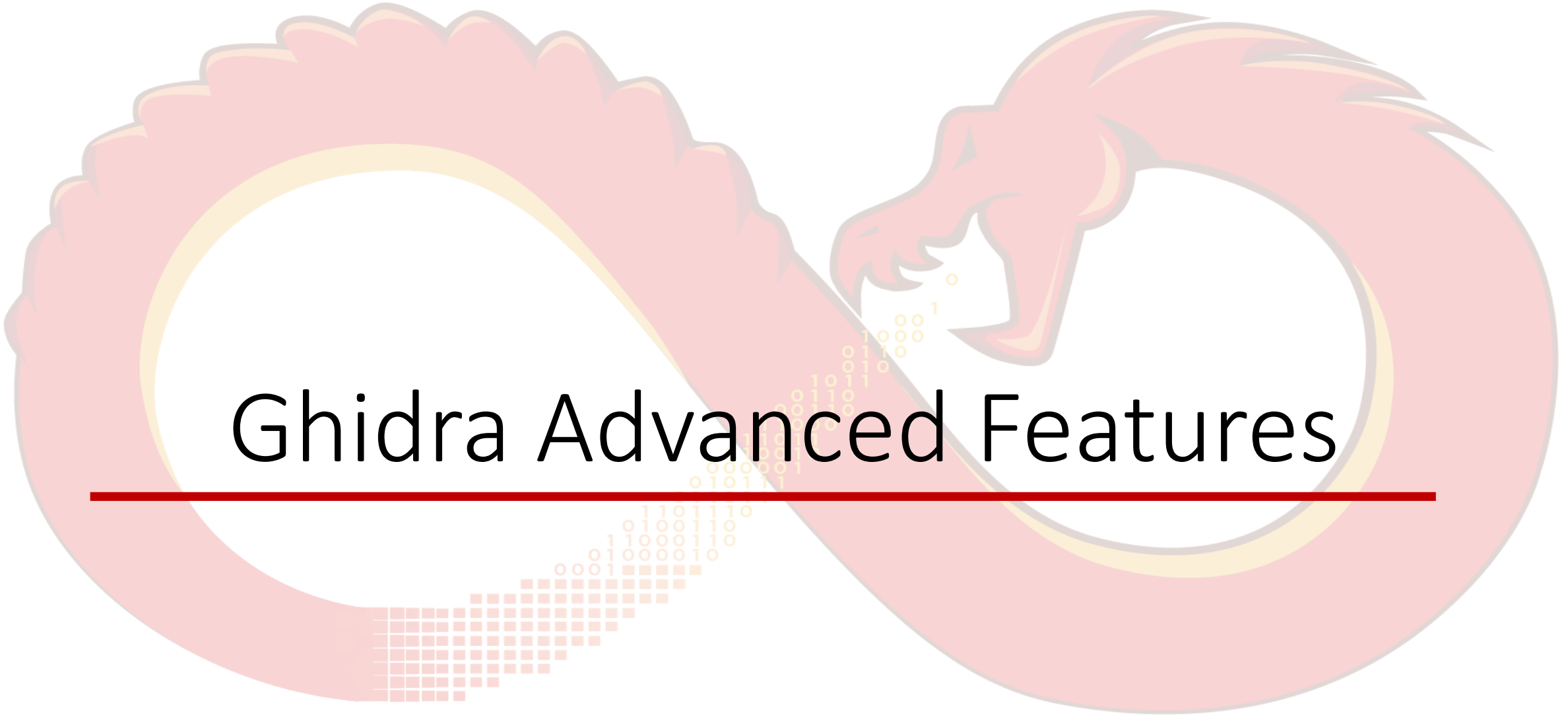


Browse to your script folder to add it to the other bundles

Task

Ca' Foscari University of Venice

- Now look at the other available scripts and try them on the crackme files or any other files you want.
- Explain your findings to the class!



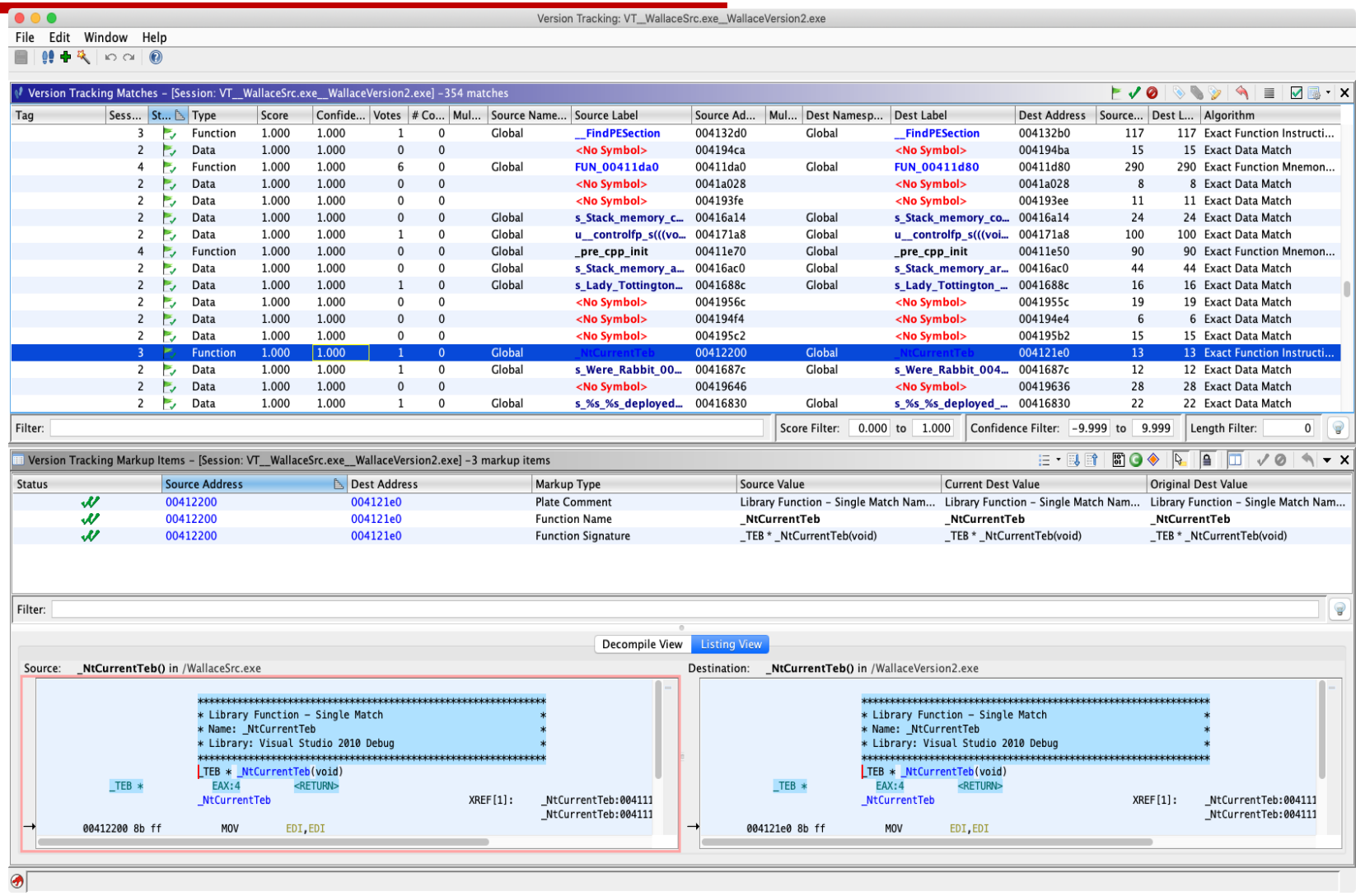
Ghidra Advanced Features

Credits: Alexei Bulazel & Jeremy Blackthorne

Version Tracker - Overview

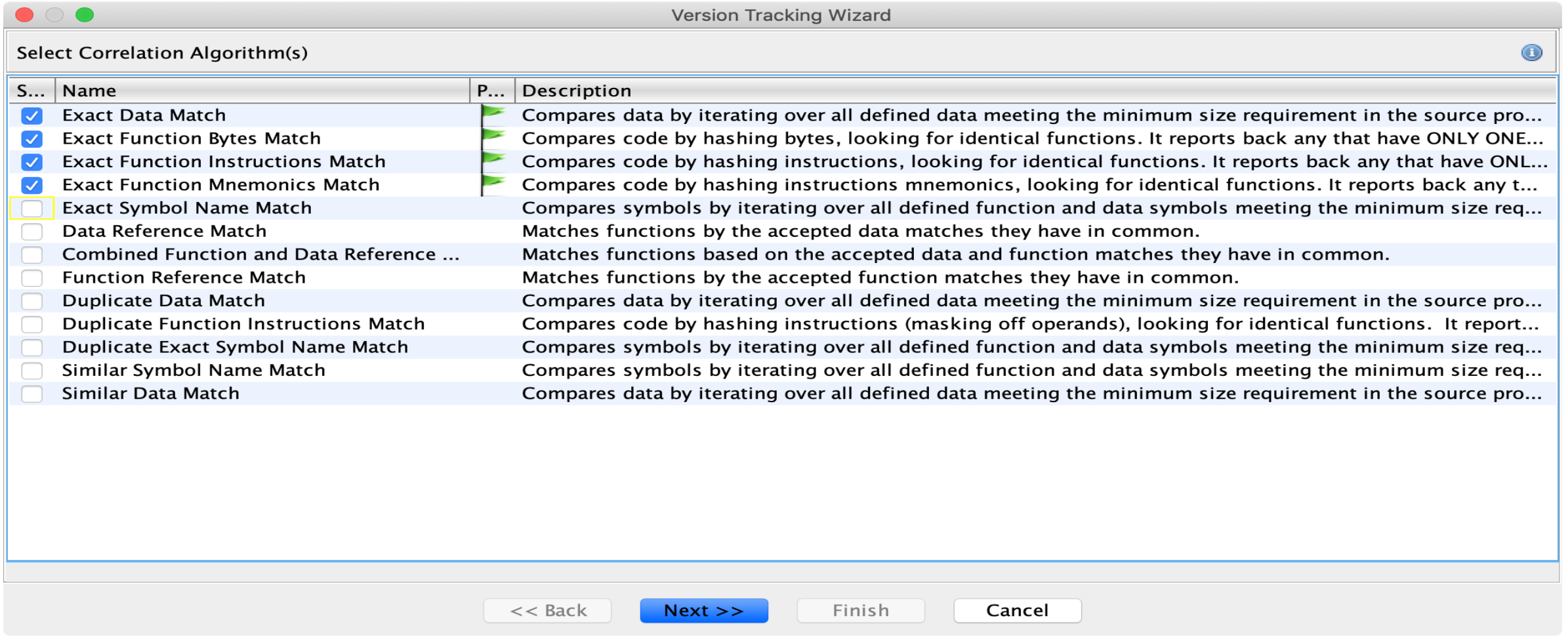
Ca' Foscari University of Venice

- Feature for porting RE symbols, annotations, etc. between incrementally updated versions of the same program
- Not well suited for quick 1-day discovery in patch analysis
  - Use Diaphora or BinDiff for this purpose



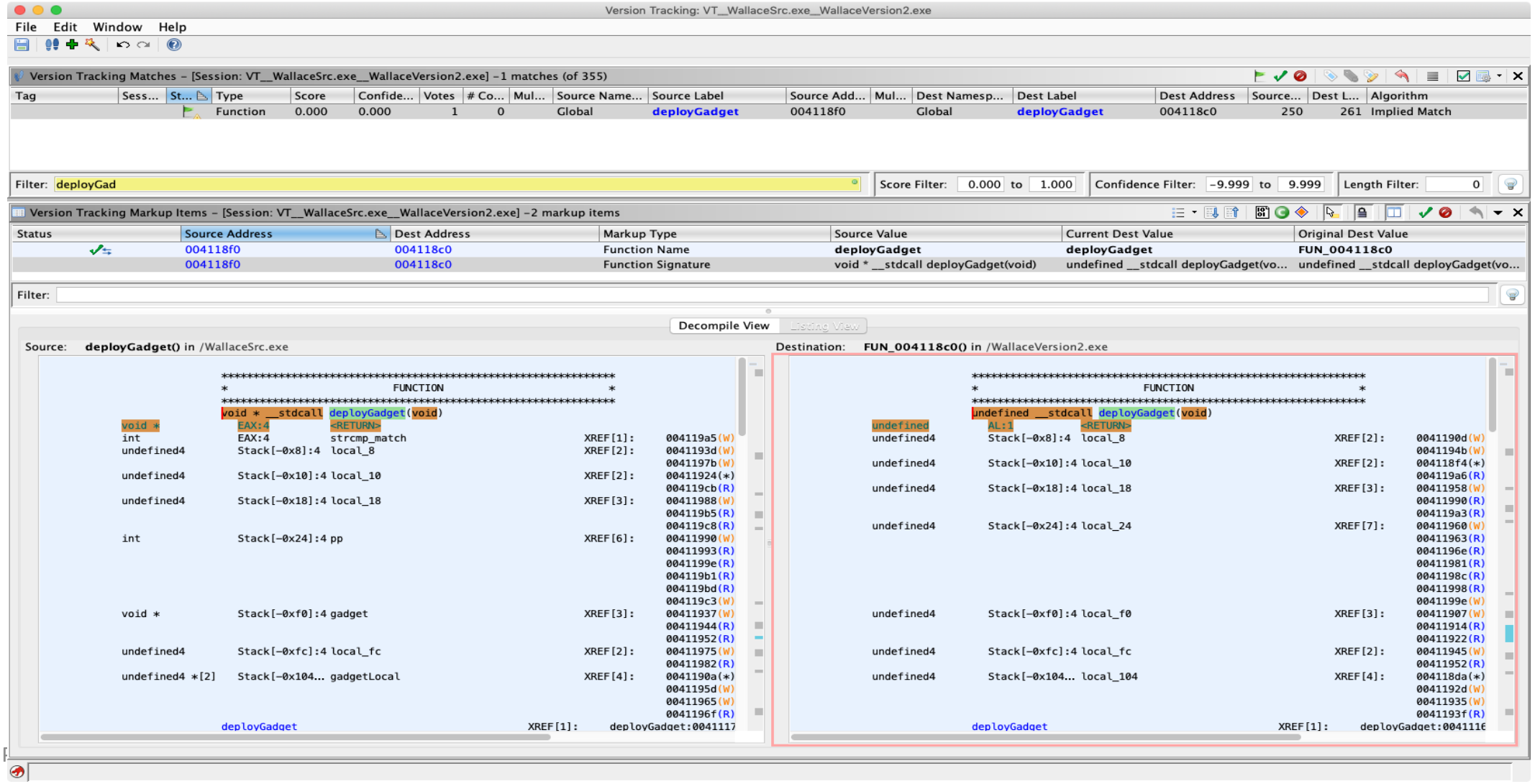
Version Tracker - Selecting Correlation Algorithms

Ca' Foscari University of Venice



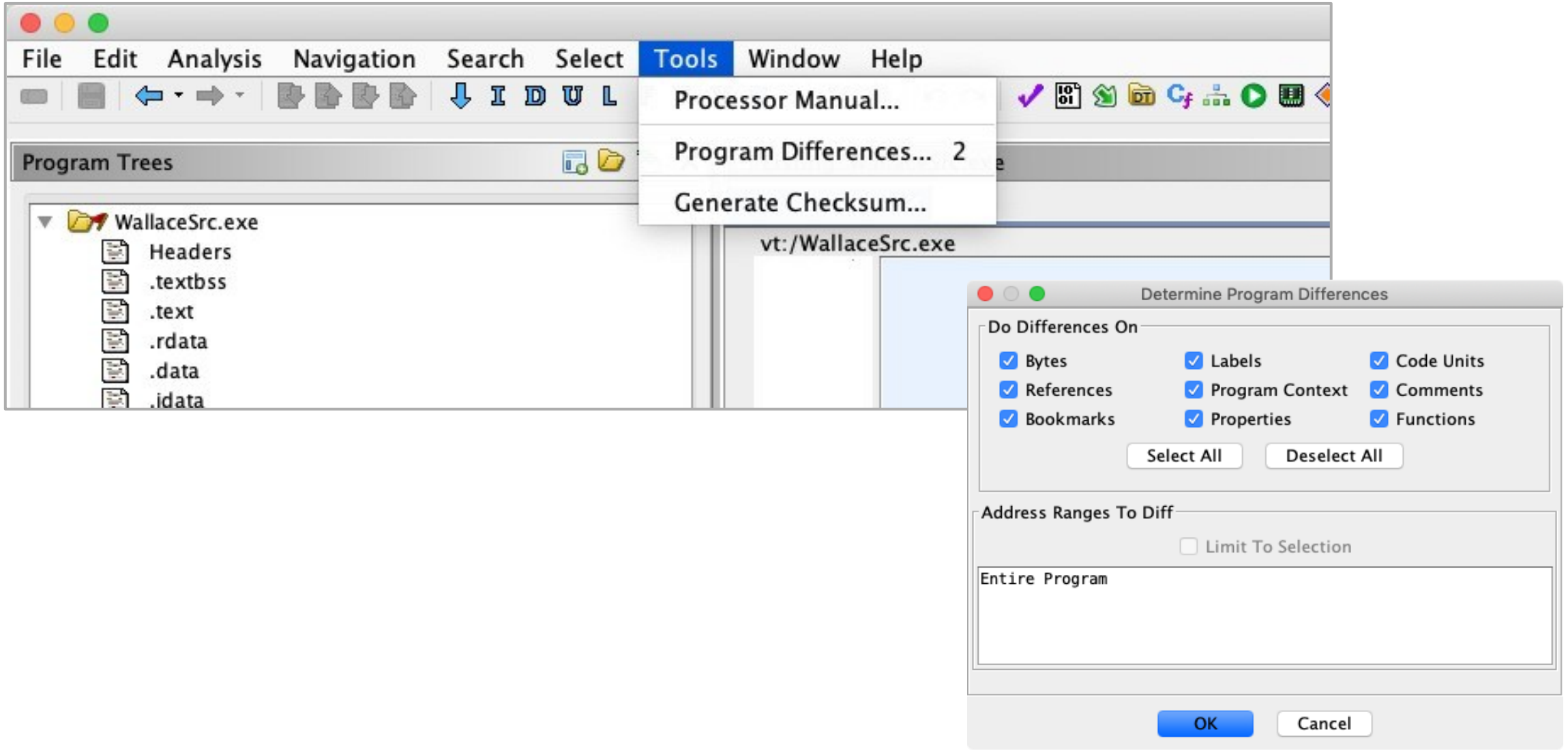
Version Tracker - Function Name Ported

Ca' Foscari University of Venice



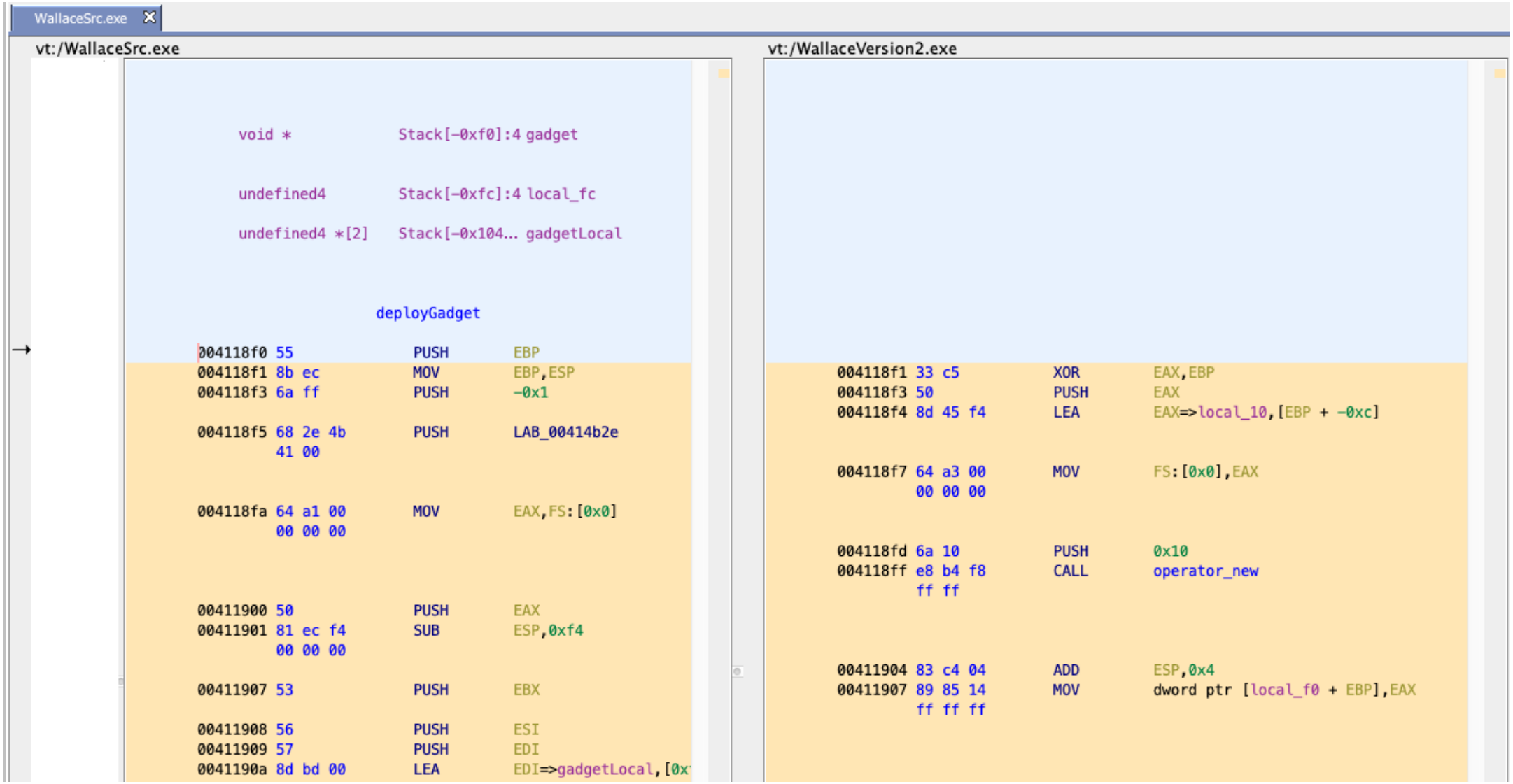
Program Differences

Ca' Foscari University of Venice



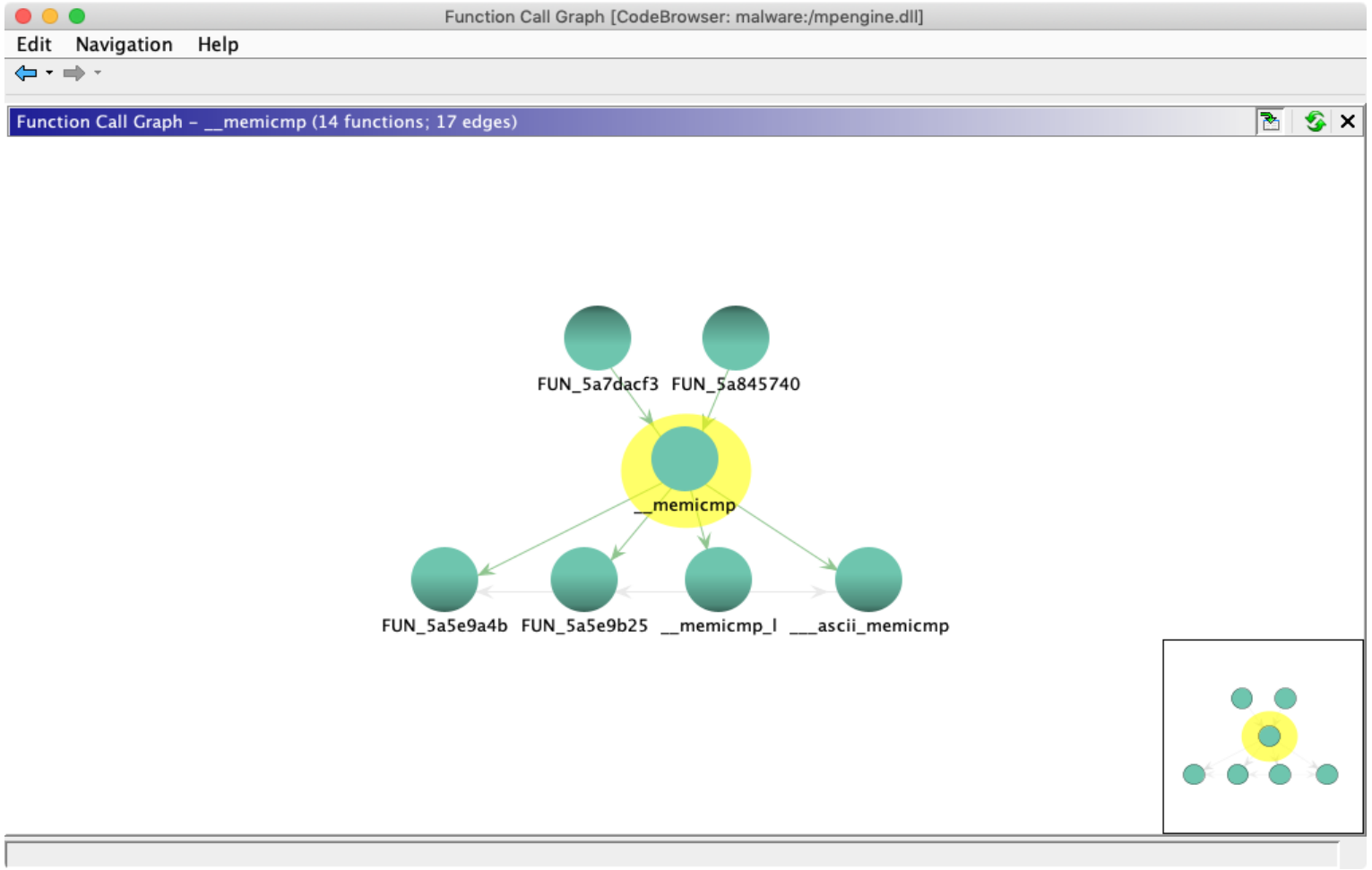
Program Differences

Ca' Foscari University of Venice



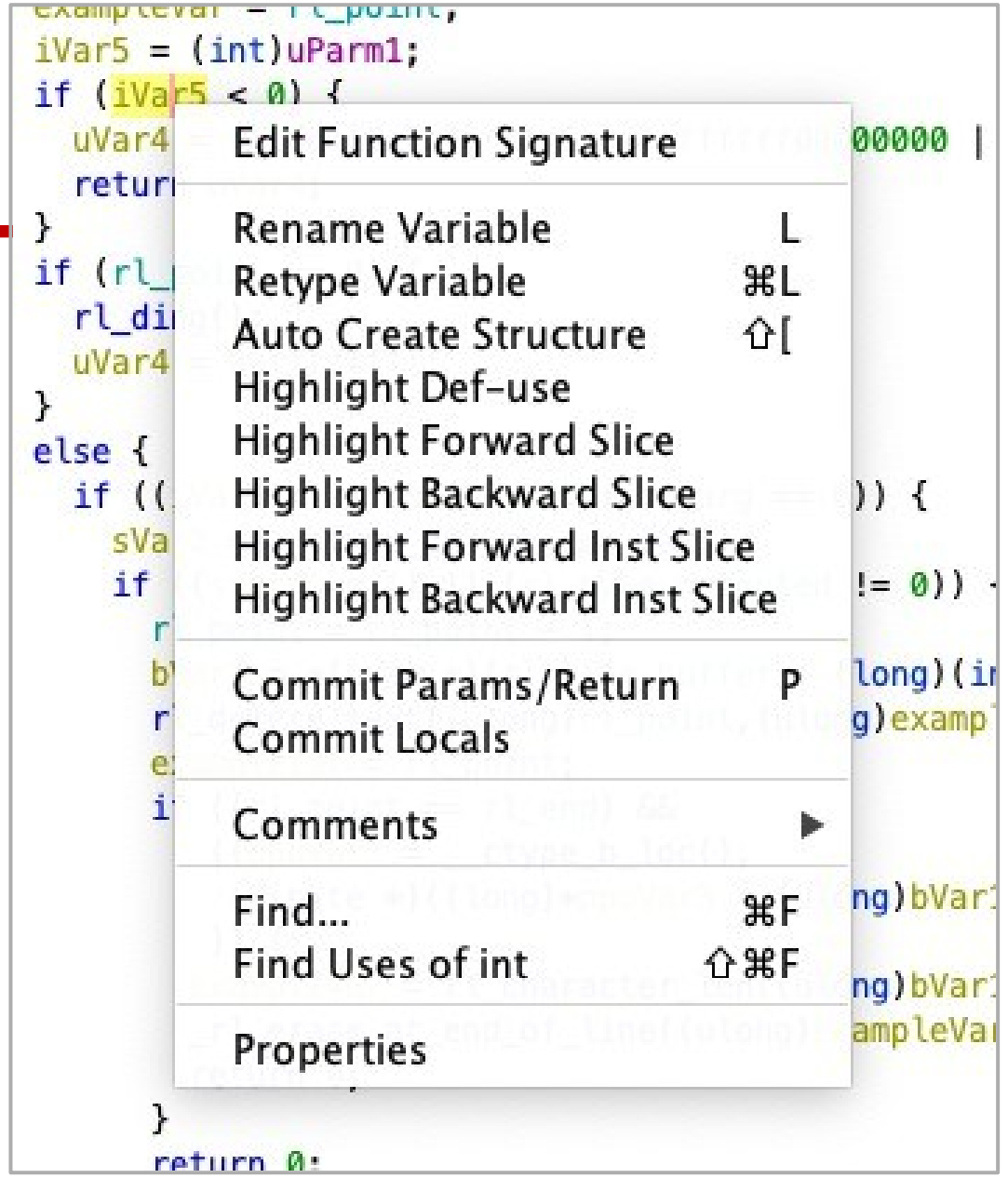
Function Call Graph

Ca' Foscari University of Venice



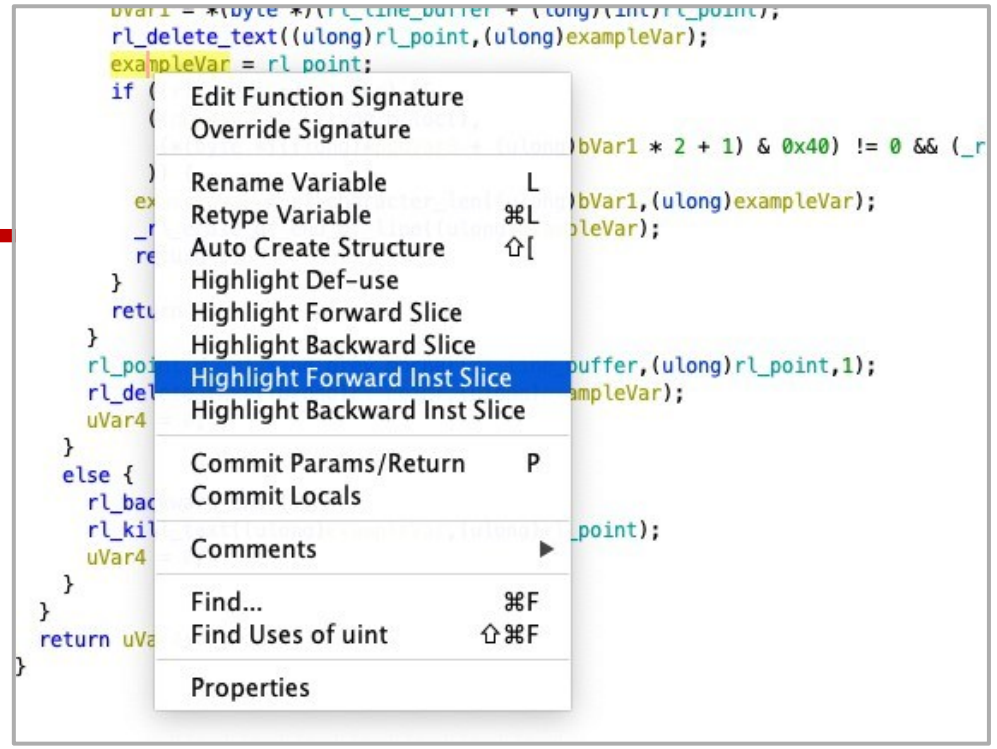
Decompiler Slicing

- Forward Slice consists of all statements in program P that may be affected by the value of variable v in a statement S
- Backward Slice consists of all statements in program P that may affect the value of variable v in a statement S



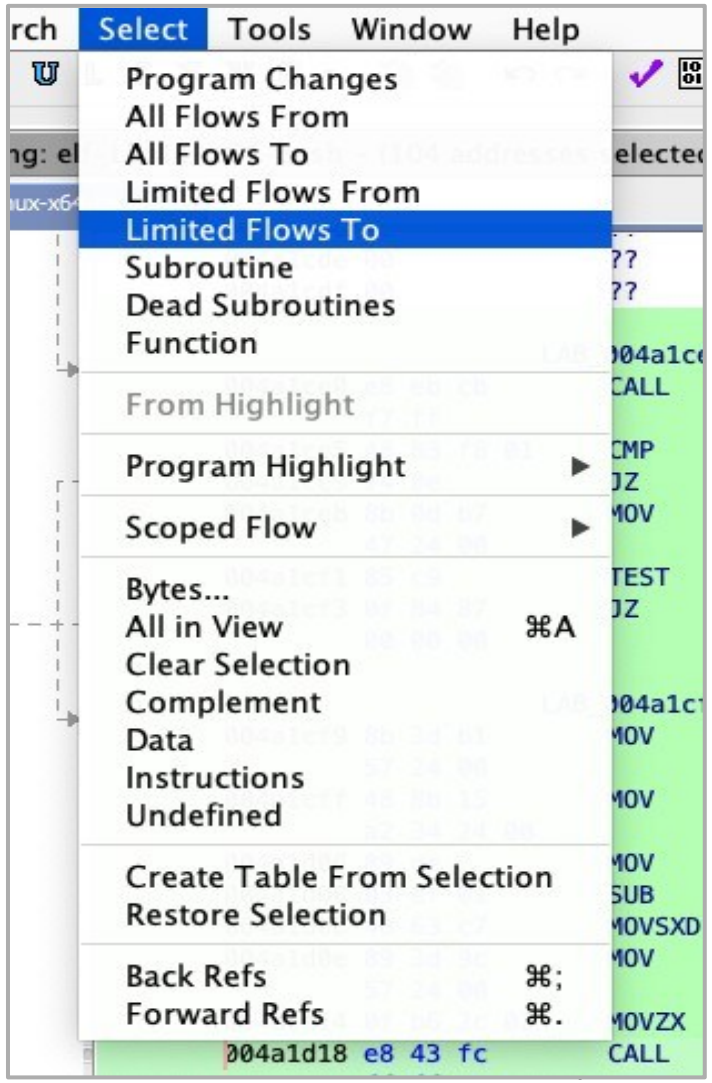
Slicing

```
undefined8 _rl_rubout_char(ulong uParm1)

{
  byte bVar1;
  size_t sVar2;
  ushort **ppuVar3;
  undefined8 uVar4;
  int iVar5;
  uint exampleVar;

  exampleVar = rl_point;
  iVar5 = (int)uParm1;
  if (iVar5 < 0) {
    uVar4 = rl_delete(uParm1 & 0xffffffff00000000 | (ulong)(uint)-iVar5);
    return uVar4;
  }
  if (rl_point == 0) {
    rl_ding();
    uVar4 = 0xffffffff;
  }
  else {
    if ((iVar5 < 2) && (rl_explicit_arg == 0)) {
      sVar2 = __ctype_get_mb_cur_max();
      if ((sVar2 == 1) || (rl_byte_oriented != 0)) {
        rl_point = rl_point - 1;
        bVar1 = *(byte *)(rl_line_buffer + (long)(int)rl_point);
        rl_delete_text((ulong)rl_point,(ulong)exampleVar);
        exampleVar = rl_point;
        if ((rl_point == rl_end) &&
           ((ppuVar3 = __ctype_b_loc(),
            (*(byte *)((long)*ppuVar3 + (ulong)bVar1 * 2 + 1) & 0x40) != 0 &&
            )) {
          exampleVar = rl_character_len((ulong)bVar1,(ulong)exampleVar);
          _rl_erase_at_end_of_line((ulong)exampleVar);
          return 0;
        }
        return 0;
      }
      rl_point = _rl_find_prev_mbchar(rl_line_buffer,(ulong)rl_point,1);
      rl_delete_text((ulong)rl_point,(ulong)exampleVar);
      uVar4 = 0;
    }
    else {
      rl_backward_char();
      rl_kill_text((ulong)exampleVar,(ulong)rl_point);
      uVar4 = 0;
    }
  }
  return uVar4;
}
```



Ca' Foscari University of Venice

Decompiler Slicing




```
undefined8 _rl_rubout_char(ulong uParm1)

{
  byte bVar1;
  size_t sVar2;
  ushort **ppuVar3;
  undefined8 uVar4;
  int iVar5;
  uint exampleVar;

  exampleVar = rl_point;
  iVar5 = (int)uParm1;
  if (iVar5 < 0) {
    uVar4 = rl_delete(uParm1 & 0xffffffff00000000 | (ulong)(uint)-iVar5);
    return uVar4;
  }
  if (rl_point == 0) {
    rl_ding();
    uVar4 = 0xffffffff;
  }
  else {
    if ((iVar5 < 2) && (rl_explicit_arg == 0)) {
      sVar2 = __ctype_get_mb_cur_max();
      if ((sVar2 == 1) || (rl_byte_oriented != 0)) {
        rl_point = rl_point - 1;
        bVar1 = *(byte *)(rl_line_buffer + (long)(int)rl_point);
        rl_delete_text((ulong)rl_point,(ulong)exampleVar);
        exampleVar = rl_point;
        if ((rl_point == rl_end) &&
           ((ppuVar3 = __ctype_b_loc(),
            (*(byte *)((long)*ppuVar3 + (ulong)bVar1 * 2 + 1) & 0x40) != 0 && (_rl_last_c_pos != 0))
           )) {
          exampleVar = rl_character_len((ulong)bVar1,(ulong)exampleVar);
          _rl_erase_at_end_of_line((ulong)exampleVar);
          return 0;
        }
        return 0;
      }
      rl_point = _rl_find_prev_mbchar(rl_line_buffer,(ulong)rl_point,1);
      rl_delete_text((ulong)rl_point,(ulong)exampleVar);
      uVar4 = 0;
    }
    else {
      rl_backward_char();
      rl_kill_text((ulong)exampleVar,(ulong)rl_point);
      uVar4 = 0;
    }
  }
  return uVar4;
}
```

P-Code

- Ghidra's intermediate language
- Dates back to at least 2005 according to documentation
- Code for different processors can be lifted into p-code, data-flow analysis and decompilation can then run over the p-code
- Pseudo-assembly, represents lifted instructions as small atomic operations without side-effects
- Built-in floating point support

```
MOVSXD      RAX,EDX
                    (register, 0x0, 8) = INT_SEXT (register, 0x10, 4)
LEA         RAX,[RAX + RAX*0x4]
                    (unique, 0x660, 8) = INT_MULT (register, 0x0, 8), (const, 0x4, 8)
                    (unique, 0x680, 8) = INT_ADD (register, 0x0, 8), (unique, 0x660, 8)
                    (register, 0x0, 8) = COPY (unique, 0x680, 8)
```

P-Code Design

Ca' Foscari University of Venice

- The language is machine independent.
- The language is designed to model general purpose processors.
- Instructions operate on user defined registers and address spaces.
- All data is manipulated explicitly. Instructions have no indirect effects.
- Individual p-code operations mirror typical processor tasks and concepts.

Processor to p-code modeling:

- RAM → address space
- Register → varnode
- Instruction → operation

```
SCASB.REPNE   RDI
                        $U22d0:1 = INT_EQUAL RCX, 0:8
                        CBRANCH *[ram]0x401548:8, $U22d0
                        RCX = INT_SUB RCX, 1:8
                        $U1d90:8 = COPY RDI
                        $U1da0:8 = INT_ADD RDI, 1:8
                        $U1db0:8 = INT_ZEXT DF
                        $U1dc0:8 = INT_MULT 2:8, $U1db0
                        RDI = INT_SUB $U1da0, $U1dc0
                        $U1de0:1 = LOAD ram($U1d90)
                        CF = INT_LESS AL, $U1de0
                        $U1de0:1 = LOAD ram($U1d90)
                        OF = INT_SBORROW AL, $U1de0
                        $U1de0:1 = LOAD ram($U1d90)
                        $Uac60:1 = INT_SUB AL, $U1de0
                        SF = INT_SLESS $Uac60, 0:1
                        ZF = INT_EQUAL $Uac60, 0:1
                        $U22f0:1 = BOOL_NEGATE ZF
                        CBRANCH *[ram]0x401546:8, $U22f0
NOT           RCX
                        RCX = INT_NEGATE RCX
```

Category	P-Code Operations
Data Moving	COPY, LOAD, STORE
Arithmetic	INT_ADD, INT_SUB, INT_CARRY, INT_SCARRY, INT_SBORROW, INT_2COMP, INT_MULT, INT_DIV, INT_SDIV, INT_REM, INT_SREM
Logical	INT_NEGATE, INT_XOR, INT_AND, INT_OR, INT_LEFT, INT_RIGHT, INT_SRIGHT
Int Comparison	INT_EQUAL, INT_NOTEQUAL, INT_SLESS, INT_SLESSEQUAL, INT_LESS, INT_LESSEQUAL
Boolean	BOOL_NEGATE, BOOL_XOR, BOOL_AND, BOOL_OR
Floating Point	FLOAT_ADD, FLOAT_SUB, FLOAT_MULT, FLOAT_DIV, FLOAT_NEG, FLOAT_ABS, FLOAT_SQRT, FLOAT_NAN
FP Compare	FLOAT_EQUAL, FLOAT_NOTEQUAL, FLOAT_LESS, FLOAT_LESSEQUAL
FP Conversion	INT2FLOAT, FLOAT2FLOAT, TRUNC, CEIL, FLOOR, ROUND
Branching	BRANCH, CBRANCH, BRANCHIND, CALL, CALLIND, RETURN
Extension / Truncation	INT_ZEXT, INT_SEXT, PIECE, SUBPIECE

IDA vs Binary Ninja vs Ghidra

Ca' Foscari University of Venice

IDA

- Maturity
- Windows support
- Decompiler
- Existing corpus of powerful plugins
- Debugger
- Support for paid customers
- Well tested
- Industry standard

Binary Ninja

- Innovation and modern design
- Program analysis features (SSA)
- Multi-level IL
- Rich API
- Embeddable
- Python-native scripting
- Clean modern UI
- Community

Ghidra

- Maturity
- Embedded support
- Decompiler
- Massive API
- Documentation
- Breath of features
- Collaboration
- Version tracking
- Price and open source extensibility

Decompiler - IDA Hex-Rays vs Ghidra

Ca' Foscari University of Venice

IDA Hex-Rays

- Optional add-on for IDA for IDA
- Microcode-based
- Supports limited architectures
- Better built-in support for Windows
- Variables, data, and functions can be xrefed from decompiler
- Variables can be mapped
- Variable representation can be changed in the decompiler (decimal, hex, char immediate, etc)
- Click to highlight

Ghidra Decompiler

- Deeply integrated with Ghidra
- P-code based
- Supports all architectures
- No way to xref from decompiler
- Produces fewer `goto` statements and seemingly more idiomatic C
- Built in program analysis features, e.g., slicing and data flow
- Variables cannot be mapped
- Variable representation cannot be changed in the decompiler
- *Middle click* to highlight

Binary Ninja vs Ghidra

Ca' Foscari University of Venice

Binary Ninja

- Multi-level: LLIL, MLIL, forthcoming HLIL
- Machine consumable and human readable
- SSA form
- Designed in light of years of program analysis research
- Feels nicer to work with
- Deferred flag calculations

Ghidra

- Single level p-code, but can be enhanced by decompiler analysis
- Designed for machine consumption first, not human readability
- Uses SSA during decompilation, but raw p-code is not SSA
- Design origins based off of program analysis research from 20+ years ago

ILs - Binary Ninja vs Ghidra

Ca' Foscari University of Venice

```
phase_4:
rsp = rsp - 0x18
rcx = rsp + 0xc {var_c}
rdx = rsp + 8 {var_10}
esi = 0x4025cf  {"%d %d"}
eax = 0
call(__isoc99_sscanf)
if (eax != 2) then 7 @ 0x401035 else 9 @ 0x401033
```

LLIL

```
phase_4:
int32_t* rcx = &var_c
int32_t* rdx = &var_10
rax = __isoc99_sscanf(arg1, 0x4025cf, rdx, rcx)  {"%d %d"}
if (rax.eax != 2) then 4 @ 0x401035 else 6 @ 0x401033
```

MLIL

```
0040100c - phase_4
undefined phase_4()
        undefined          AL:1           <RETURN>
        undefined4         Stack[-0xc]:4  local_c
        undefined4         Stack[-0x10]:4 local_10
        CF = INT_LESS RSP, 24:8
        OF = INT_SBORROW RSP, 24:8
        RSP = INT_SUB RSP, 24:8
        SF = INT_SLESS RSP, 0:8
        ZF = INT_EQUAL RSP, 0:8
        $U770:8 = INT_ADD 12:8, RSP
        RCX = COPY $U770
        $U770:8 = INT_ADD 8:8, RSP
        RDX = COPY $U770
        RSI = COPY 0x4025cf:8
        RAX = COPY 0:8
        RSP = INT_SUB RSP, 8:8
        STORE ram(RSP), 0x401029:8
        CALL *[ram]0x400bf0:8
        CF = INT_LESS EAX, 2:4
        OF = INT_SBORROW EAX, 2:4
        $U58c0:4 = INT_SUB EAX, 2:4
        SF = INT_SLESS $U58c0, 0:4
        ZF = INT_EQUAL $U58c0, 0:4
        $U2080:1 = BOOL_NEGATE ZF
        CBRANCH *[ram]0x401035:8, $U2080
```

Ghidra is good for…

Ca' Foscari University of Venice

- Scripting reverse engineering
- Firmware / embedded systems analysis
- Analysis of software that Hex-Rays can't decompile
- Collaborative long-term professional RE
- Unique windows
  - Checksum Generator
  - Disassembled View
  - Data Type Preview
  - Function Tags
  - Symbol tree

Ghidra Pros

- Built for multi-monitor use
- "Moving ants" highlight on control flow graphs
- Configurable "tool" views
- Hotkeys mappable to actions and scripts
- Right click > "extract and import"
- Processor manual integration
- Undo button
- Import directly from zip file
- Snapshot views
- Configurable listings
- Version tracker

Ghidra Pros

Ca' Foscari University of Venice

- Project-based multi-binary RE
- `F1` to open help on whatever
  - the mouse is pointing at
- File System browser
- Highly configurable assembly code listing
- Data flow analysis built into UI
- Embedded image detection
- Search for matching instructions

Contributing to Ghidra

Ca' Foscari University of Venice

- Ghidra code is available on Github
  - Apache License 2.0
- NSA has been responsive to community questions and bug reports posted on Github

- Official site: ghidra-sre.org
- Open source: github.com/NationalSecurityAgency/ghidra
  github.com/NationalSecurityAgency/ghidra-data

Conclusions

Ca' Foscari University of Venice

- Ghidra is a powerful binary reverse engineering tool built by the US National Security Agency
- For reverse engineers, by reverse engineers
- Interactive and headless scripting
- Built for program analysis