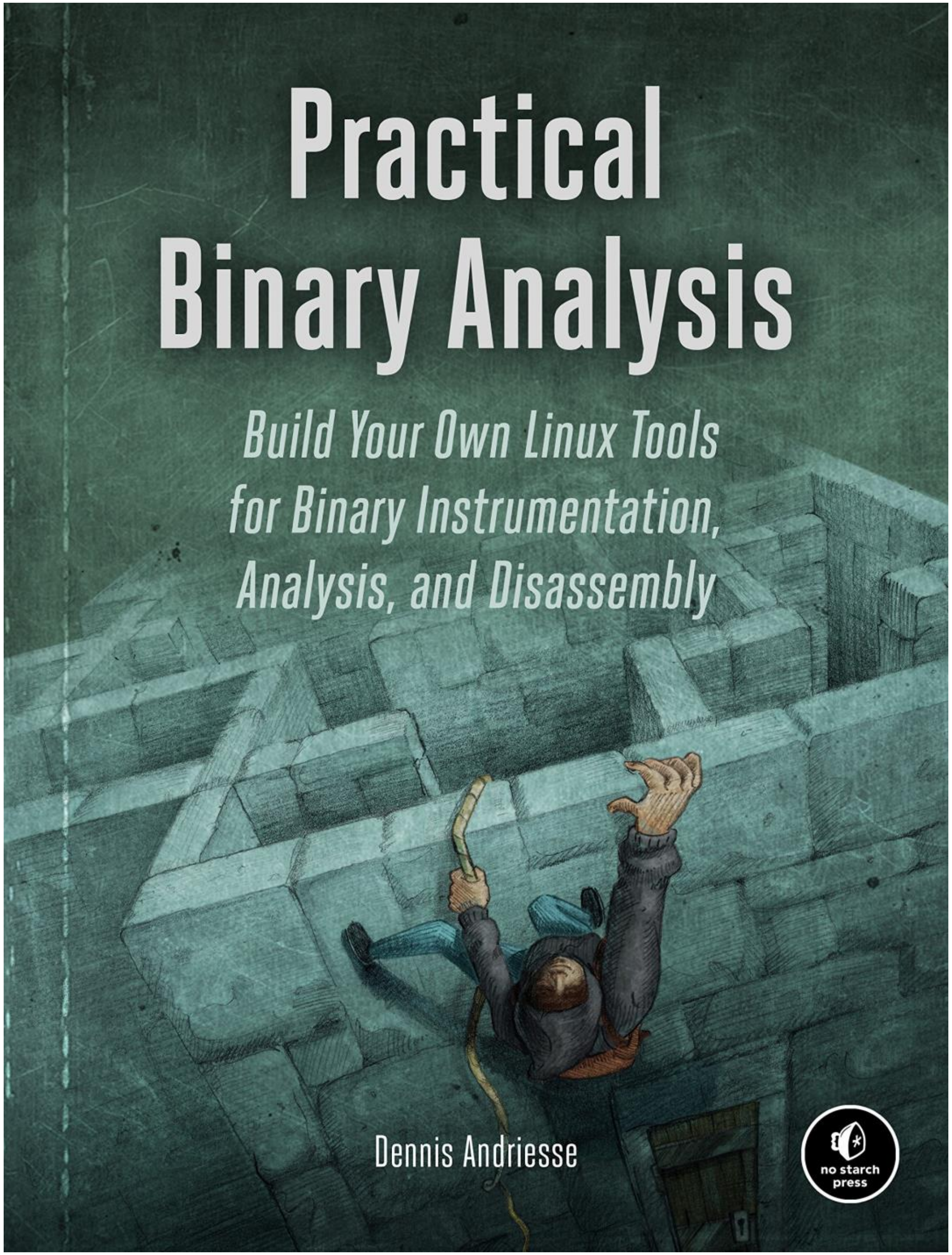
Binary Analysis

Dennis Andriesse

Finse Winter School 2018

Who am I?

- Researcher at Vrije Universiteit Amsterdam
  - Reverse engineering
  - Hardening programs/anti-exploitation
  - Malware analysis
  - …
- Attack developer in GameOver Zeus takedown
- Past year: writing a book on *binary analysis*
  - Topic of these lectures



Practical Binary Analysis

Build Your Own Linux Tools for Binary Instrumentation, Analysis, and Disassembly

Dennis Andriesse

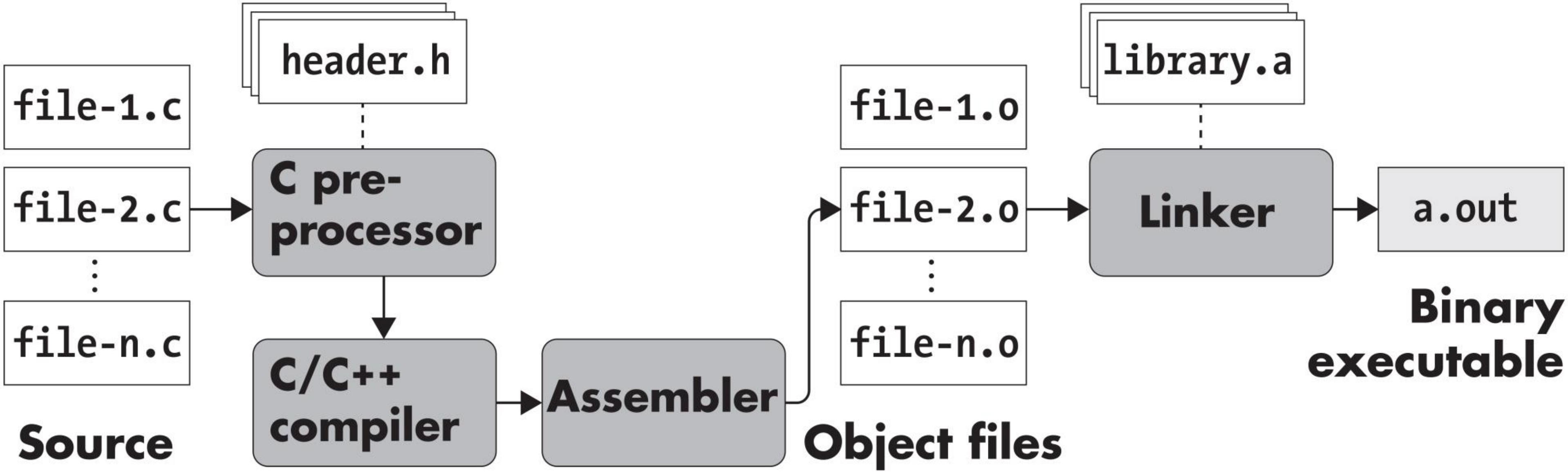
no starch press

What is Binary Analysis?

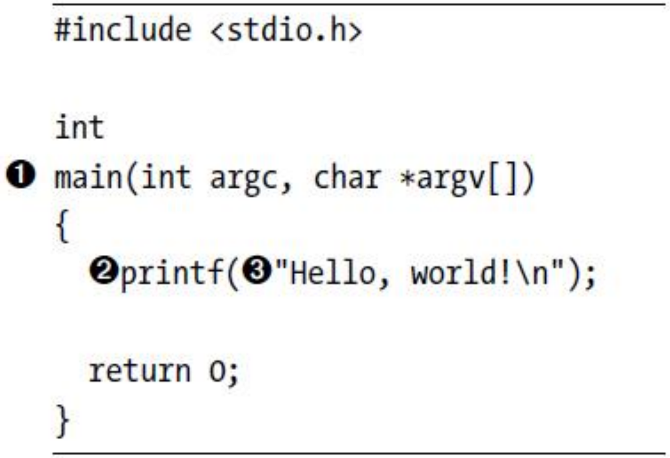
- Analyzing and/or modifying programs at the *binary* (e.g. *machine code*) level
- As opposed to source-level analysis (C/Java/Python/. . .)
- Simple example: disassembling a program with `objdump`
- Here: focus on x86 ELF binaries

Producing a Binary

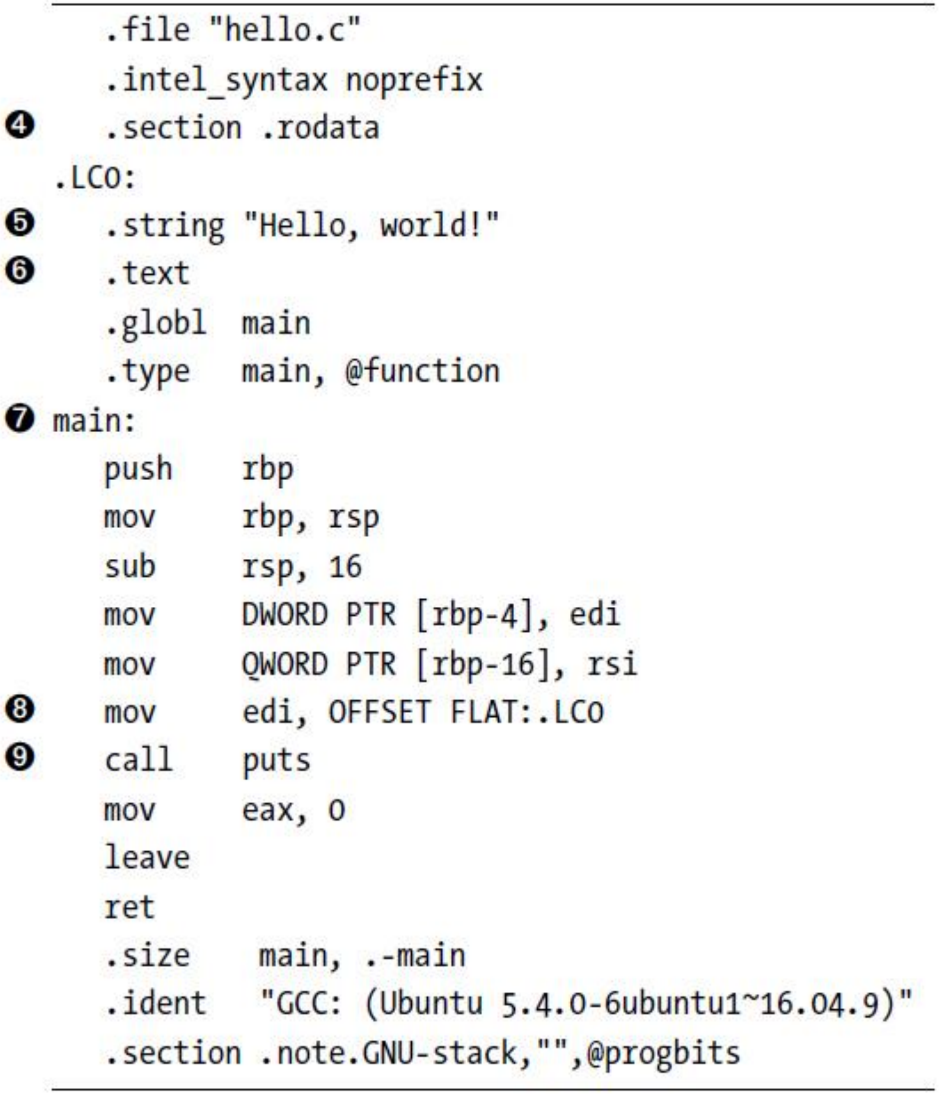
- High-level C programs compile into binaries
- Intermediate step: assembly language



Example of source vs assembly

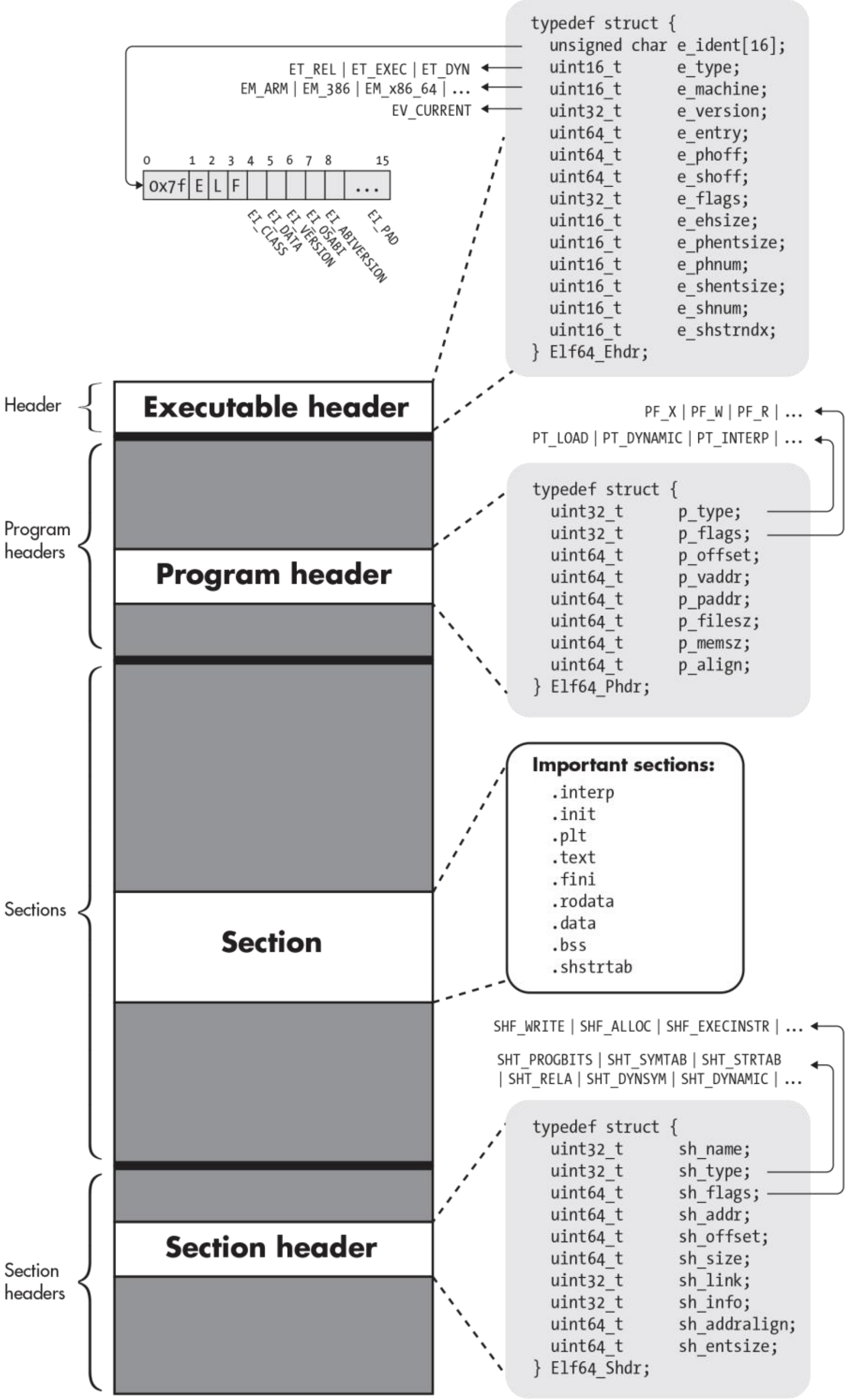


```
  #include <stdio.h>

  int
❶ main(int argc, char *argv[])
  {
    ❷printf(❸"Hello, world!\n");

    return 0;
  }
```



```
       .file "hello.c"
       .intel_syntax noprefix
❹      .section .rodata
    .LC0:
❺      .string "Hello, world!"
❻      .text
       .globl  main
       .type   main, @function
❼  main:
       push    rbp
       mov     rbp, rsp
       sub     rsp, 16
       mov     DWORD PTR [rbp-4], edi
       mov     QWORD PTR [rbp-16], rsi
❽      mov     edi, OFFSET FLAT:.LC0
❾      call    puts
       mov     eax, 0
       leave
       ret
       .size    main, .-main
       .ident   "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.9)"
       .section .note.GNU-stack,"",@progbits
```

The ELF binary format



Header
Executable header
Program headers
Program header
Sections
Section
Section headers
Section header
ET_REL | ET_EXEC | ET_DYN
EM_ARM | EM_386 | EM_x86_64 | ...
EV_CURRENT
0 1 2 3 4 5 6 7 8 15
0x7f E L F ...
EI_CLASS
EI_DATA
EI_VERSION
EI_OSABI
EI_ABIVERSION
EI_PAD
PF_X | PF_W | PF_R | ...
PT_LOAD | PT_DYNAMIC | PT_INTERP | ...
SHF_WRITE | SHF_ALLOC | SHF_EXECINSTR | ...
SHT_PROGBITS | SHT_SYMTAB | SHT_STRTAB
| SHT_RELA | SHT_DYNSYM | SHT_DYNAMIC | ...

```
typedef struct {
  unsigned char e_ident[16];
  uint16_t      e_type;
  uint16_t      e_machine;
  uint32_t      e_version;
  uint64_t      e_entry;
  uint64_t      e_phoff;
  uint64_t      e_shoff;
  uint32_t      e_flags;
  uint16_t      e_ehsize;
  uint16_t      e_phentsize;
  uint16_t      e_phnum;
  uint16_t      e_shentsize;
  uint16_t      e_shnum;
  uint16_t      e_shstrndx;
} Elf64_Ehdr;
```

```
typedef struct {
  uint32_t      p_type;
  uint32_t      p_flags;
  uint64_t      p_offset;
  uint64_t      p_vaddr;
  uint64_t      p_paddr;
  uint64_t      p_filesz;
  uint64_t      p_memsz;
  uint64_t      p_align;
} Elf64_Phdr;
```

Important sections:

```
.interp
.init
.plt
.text
.fini
.rodata
.data
.bss
.shstrtab
```

```
typedef struct {
  uint32_t      sh_name;
  uint32_t      sh_type;
  uint64_t      sh_flags;
  uint64_t      sh_addr;
  uint64_t      sh_offset;
  uint64_t      sh_size;
  uint32_t      sh_link;
  uint32_t      sh_info;
  uint64_t      sh_addralign;
  uint64_t      sh_entsize;
} Elf64_Shdr;
```

Disassembly

- Tools like objdump disassemble binaries into approximation of the original assembly code
- Binary analysis uses disassembly or code recovered at runtime

Disassemble binary “foobar”

```
$ objdump -d ~/foobar
/home/dnx/foobar: file format elf64-x86-64

Disassembly of section .text:
...
4005ae:  55                     push %rbp
4005af:  48 89 e5               mov %rsp,%rbp
4005b2:  48 83 ec 20            sub $0x20,%rsp
4005b6:  89 7d ec               mov %edi,-0x14(%rbp)
4005b9:  48 89 75 e0            mov %rsi,-0x20(%rbp)
4005bd:  c7 45 fc 2a 00 00 00   movl $0x2a,-0x4(%rbp)
4005c4:  bf be 06 40 00         mov $0x4006be,%edi
4005c9:  e8 62 fe ff ff         callq 400430 <puts@plt>
4005ce:  8b 45 fc               mov -0x4(%rbp),%eax
4005d1:  89 c7                  mov %eax,%edi
...
```

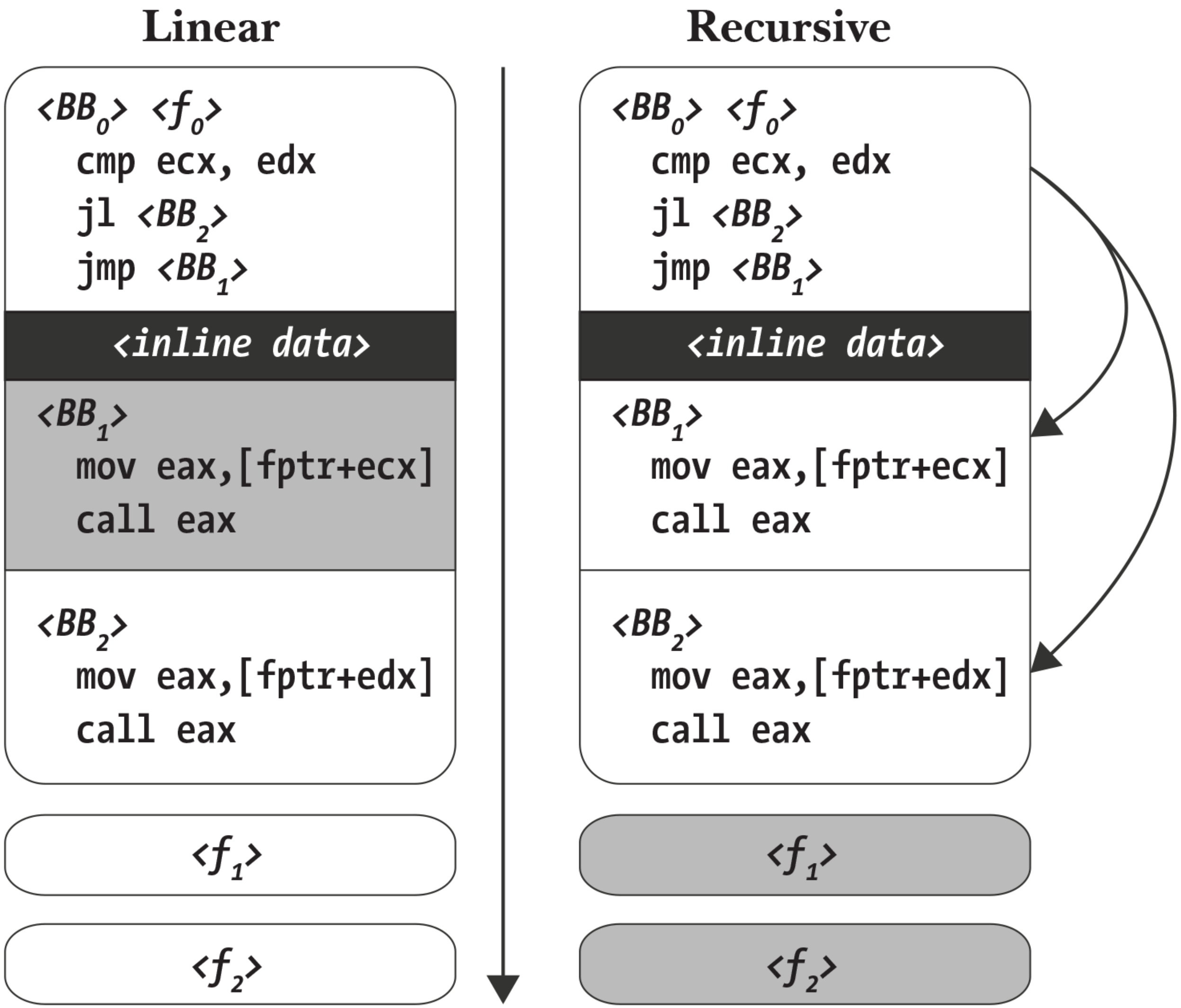
Read from register %rbp

Read from memory

Call function

Opcodes (machine level)

Linear vs Recursive Disassembly



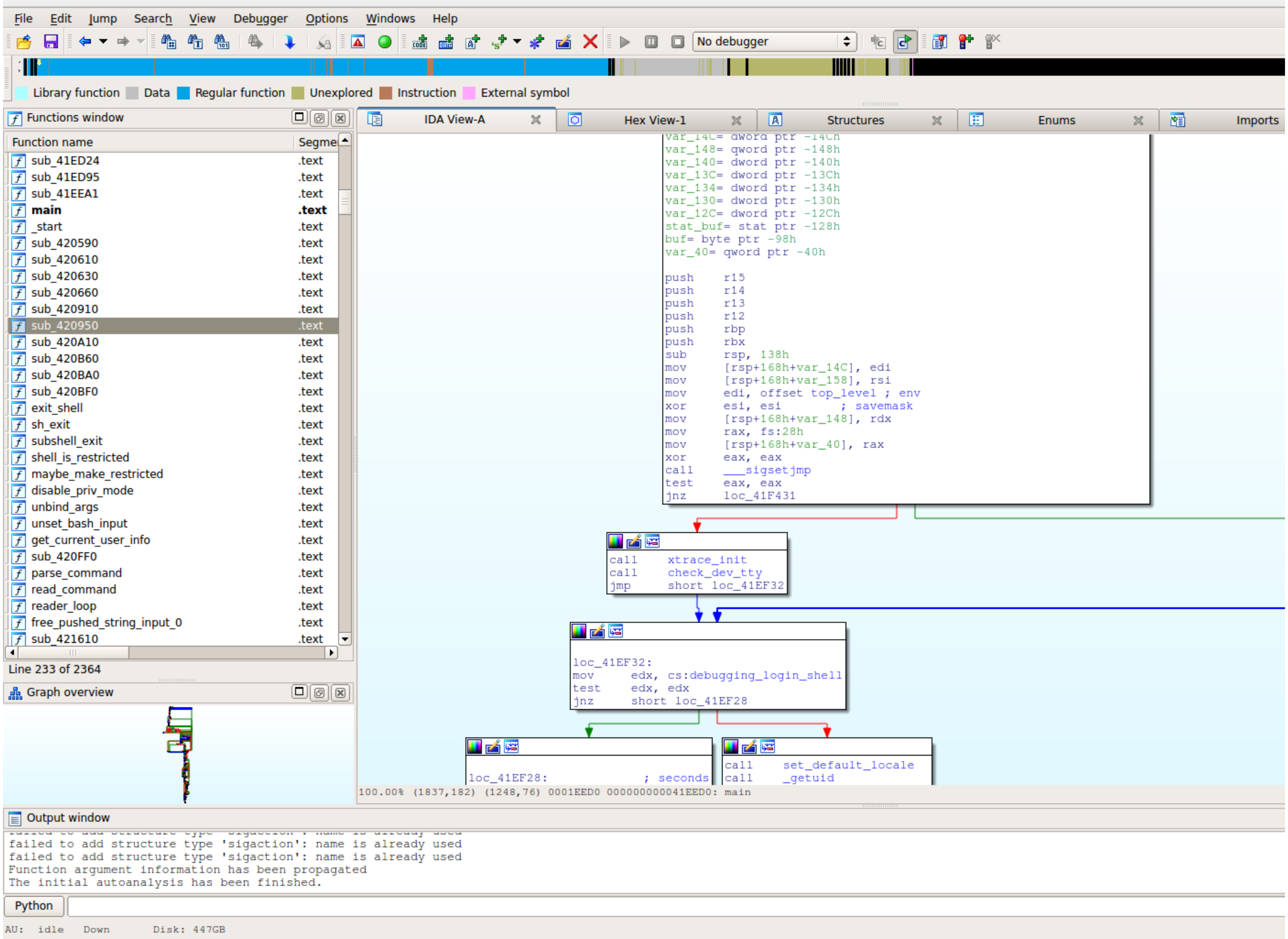
Linear

```
<BB_0> <f_0>
  cmp ecx, edx
  jl <BB_2>
  jmp <BB_1>
<inline data>
<BB_1>
  mov eax,[fptr+ecx]
  call eax
<BB_2>
  mov eax,[fptr+edx]
  call eax
<f_1>
<f_2>
```

Recursive

```
<BB_0> <f_0>
  cmp ecx, edx
  jl <BB_2>
  jmp <BB_1>
<inline data>
<BB_1>
  mov eax,[fptr+ecx]
  call eax
<BB_2>
  mov eax,[fptr+edx]
  call eax
<f_1>
<f_2>
```

Disassembly with IDA Pro



Function name	Segme
sub_41ED24	.text
sub_41ED95	.text
sub_41EEA1	.text
main	**.text**
_start	.text
sub_420590	.text
sub_420610	.text
sub_420630	.text
sub_420660	.text
sub_420910	.text
sub_420950	.text
sub_420A10	.text
sub_420B60	.text
sub_420BA0	.text
sub_420BF0	.text
exit_shell	.text
sh_exit	.text
subshell_exit	.text
shell_is_restricted	.text
maybe_make_restricted	.text
disable_priv_mode	.text
unbind_args	.text
unset_bash_input	.text
get_current_user_info	.text
sub_420FF0	.text
parse_command	.text
read_command	.text
reader_loop	.text
free_pushed_string_input_0	.text
sub_421610	.text

Line 233 of 2364

```
var_14C= dword ptr -14Ch
var_148= qword ptr -148h
var_140= dword ptr -140h
var_13C= dword ptr -13Ch
var_134= dword ptr -134h
var_130= dword ptr -130h
var_12C= dword ptr -12Ch
stat_buf= stat ptr -128h
buf= byte ptr -98h
var_40= qword ptr -40h

push    r15
push    r14
push    r13
push    r12
push    rbp
push    rbx
sub     rsp, 138h
mov     [rsp+168h+var_14C], edi
mov     [rsp+168h+var_158], rsi
mov     edi, offset top_level ; env
xor     esi, esi        ; savemask
mov     [rsp+168h+var_148], rdx
mov     rax, fs:28h
mov     [rsp+168h+var_40], rax
xor     eax, eax
call    ___sigsetjmp
test    eax, eax
jnz     loc_41F431
```

```
call    xtrace_init
call    check_dev_tty
jmp     short loc_41EF32
```

```
loc_41EF32:
mov     edx, cs:debugging_login_shell
test    edx, edx
jnz     short loc_41EF28
```

```
loc_41EF28:                 ; seconds
```

```
call    set_default_locale
call    _getuid
```

```
100.00% (1837,182) (1248,76) 0001EED0 000000000041EED0: main
```

```
failed to add structure type 'sigaction': name is already used
failed to add structure type 'sigaction': name is already used
failed to add structure type 'sigaction': name is already used
Function argument information has been propagated
The initial autoanalysis has been finished.
```

```
AU:  idle    Down       Disk: 447GB
```

Binary Analysis is Hard!

- No symbolic names for variables/functions
- No info on function/class layout
- No type information
- No clear distinction between code/data
- Inserting new code/data can break things

- *Loads of undecidable problems to deal with!*

So why do it?

- Only way to really know what a program does
- Only way to analyze malware
- Discover low-level vulnerabilities/backdoors
- Only way to change/fix binary programs
  - Source may be lost/proprietary
  - Example: Microsoft’s recent Equation Editor patch
  - Lots of vulnerable legacy programs!

BA is a large and active field

- Lots of different topics:
  - Disassembly/Reverse engineering/Malware analysis
  - Binary instrumentation/binary hardening
  - Taint analysis
  - Symbolic execution
  - . . .
- Static and dynamic (runtime) analysis

BA is a large and active field

Here we’ll focus on basic binary analysis in Linux

Many more advanced and automated analysis and binary modification tools available!

Demo: Basic Binary Analysis in Linux