A Practical Introduction to Reverse Engineering

Workshop

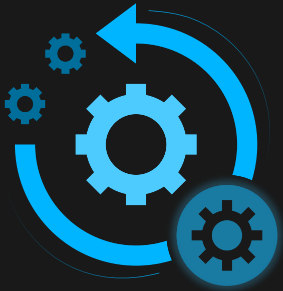
–

Romain Thomas

Introduction

Workshop Presentation

- Introduction to x86-64 Linux reverse engineering.
- 4-hours workshop + 1 hour for 1x1 questions.
- Driven by hands-on.
- A 30 minutes evaluation at the end of the session.

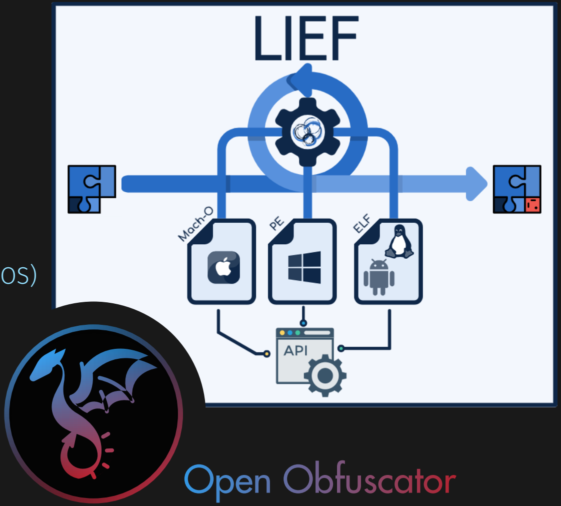


VM

- Ubuntu 22.04
- Login: `re` / Password: `re`
- Ghidra 10.2

About

- Security Engineer
- Enjoy reverse engineering and development
- Mostly doing reverse on mobile (Android & iOS)



Practical Reverse Engineering

Reverse Engineering

The purpose of reverse engineering is to highlight a *functionality* or an *asset* without having access to the original information (e.g. the source code).

Reverse Engineering

Functionalities:

- An algorithm.

Reverse Engineering

Functionalities:

- An algorithm.
- A check.

Reverse Engineering

Functionalities:

- An algorithm.
- A check.
- A structure.

Reverse Engineering

Assets:

- Password.

Reverse Engineering

Assets:

- Password.
- An API Key.

Reverse Engineering

Original information:

- Without the source code.

Reverse Engineering

Original information:

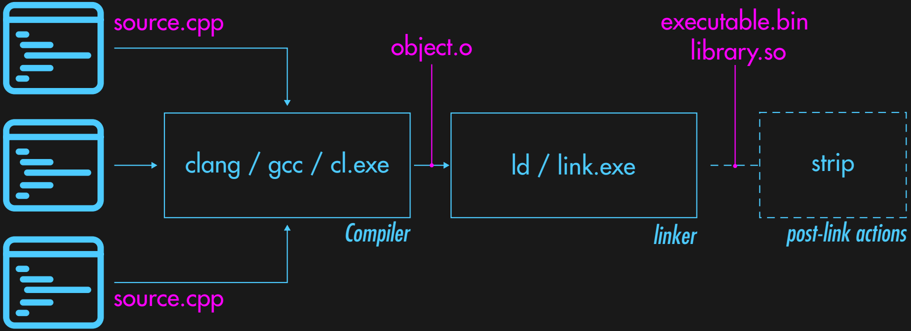
- Without the source code.
- Without the symbols.

Reverse Engineering

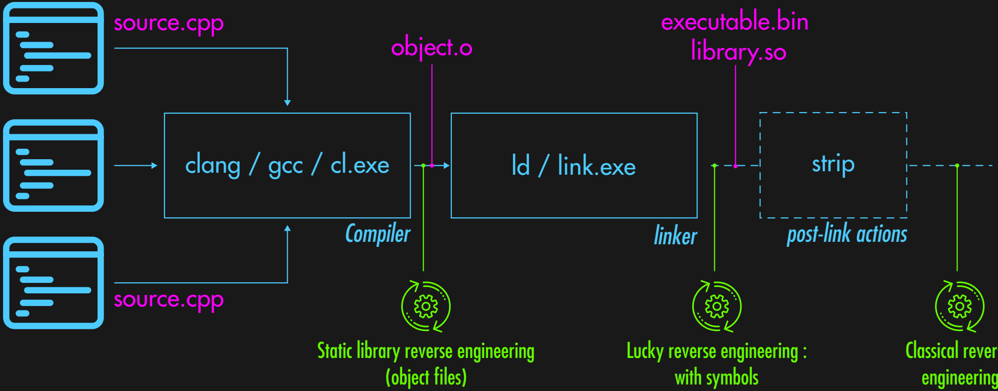
Original information:

- Without the source code.
- Without the symbols.
- With obfuscation.

Reverse Engineering



Reverse Engineering



Linux x86-64 Reverse Engineering

Linux x86-64 Reverse Engineering

Learning reverse engineering is somehow similar to learning a new language:

- Vocabulary

Linux x86-64 Reverse Engineering

Learning reverse engineering is somehow similar to learning a new language:

- Vocabulary

- Instructions

Linux x86-64 Reverse Engineering

Learning reverse engineering is somehow similar to learning a new language:

- Vocabulary
- Grammar

- Instructions

Linux x86-64 Reverse Engineering

Learning reverse engineering is somehow similar to learning a new language:

- Vocabulary
- Grammar

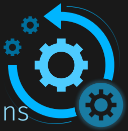
- Instructions
- Addressing Modes/ABI Conventions

Linux x86-64 Reverse Engineering

Learning reverse engineering is somehow similar to learning a new language:

- Vocabulary
- Grammar
- Idioms/Expressions

- Instructions
- Addressing Modes/ABI Conventions



Linux x86-64 Reverse Engineering

Learning reverse engineering is somehow similar to learning a new language:

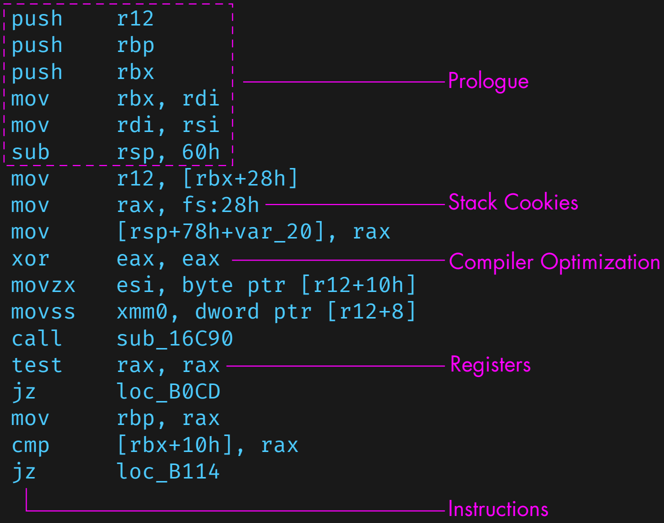
- Vocabulary
- Grammar
- Idioms/Expressions

- Instructions
- Addressing Modes/ABI Conventions
- Compiler Patterns/Optimizations

Linux x86-64 Reverse Engineering

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```

Linux x86-64 Reverse Engineering



```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```

Prologue

Stack Cookies

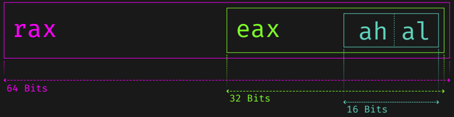
Compiler Optimization

Registers

Instructions

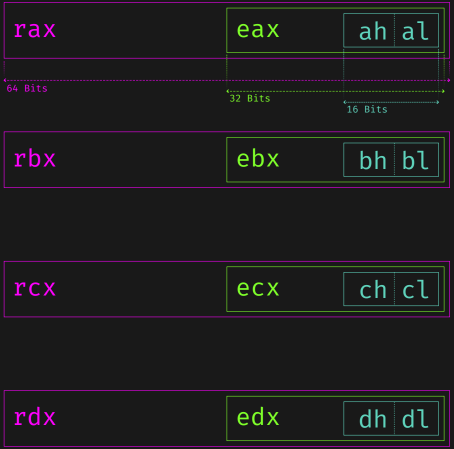
x86-64: Registers

```
push     r12
push     rbp
push     rbx
mov      rbx, rdi
mov      rdi, rsi
sub      rsp, 60h
mov      r12, [rbx+28h]
mov      rax, fs:28h
mov      [rsp+78h+var_20], rax
xor      eax, eax
movzx    esi, byte ptr [r12+10h]
movss    xmm0, dword ptr [r12+8]
call     sub_16C90
test     rax, rax
jz       loc_B0CD
mov      rbp, rax
cmp      [rbx+10h], rax
jz       loc_B114
```



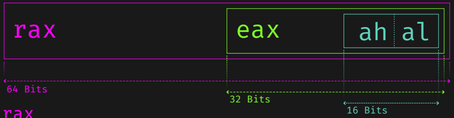
x86-64: Registers

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```



x86-64: Registers

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```

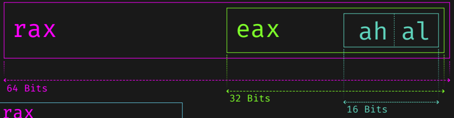


rax
rbx
rcx
rdx
rsi
rdi
rbp
rsp
r8
r9
r10
r11
r12
r13
r14
r15

16 General-Purpose Registers

x86-64: Registers

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```



rax
rbx
rcx
rdx
rsi
rdi

Calling convention & scratch registers

rbp
rsp

Stack registers

r8
r9

r10
r11
r12
r13
r14
r15

Scratch registers but callee-saved.

16 General-Purpose Registers

x86-64: Instructions

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```

```
mov DST, SRC

mov rax,            rdi
mov ecd,            dl
mov rdi,            qword ptr [rsi + 0x8]
mov byte ptr [rsp], cl
mov rax,            0x123
```

x86-64: Instructions

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```

```
push rax
pop  rbb
```

x86-64: Instructions

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```

```
add rax, rbx
sub rdx, rcx
xor eax, eax
or  rax, rax
( … )
```

x86-64: Instructions

```
push    r12
push    rbp
push    rbx
mov     rbx, rdi
mov     rdi, rsi
sub     rsp, 60h
mov     r12, [rbx+28h]
mov     rax, fs:28h
mov     [rsp+78h+var_20], rax
xor     eax, eax
movzx   esi, byte ptr [r12+10h]
movss   xmm0, dword ptr [r12+8]
call    sub_16C90
test    rax, rax
jz      loc_B0CD
mov     rbp, rax
cmp     [rbx+10h], rax
jz      loc_B114
```

```
jmp loc_2345
jmp rax
jnz rcx

call FUNC_123
call rax
```

Compiler Optimizations

x86-64: Instructions

```
rax = rbx + 2
```

```
mov     rax, rbx
add     rax, 0x2
```

x86-64: Compiler Optimizations

```
rax = rbx + 2
```

```
lea     rax, [rbx + 0x2]
```

x86-64: Compiler Optimizations

```
rax := 0            mov    rax, 0x0  48 c7 c0 00 00 00 00
```

x86-64: Compiler Optimizations

```
rax := 0            xor   rax, rax  48 31 c0
```

x86-64: Compiler Optimizations

```
X % 8
```

```
mov     rax, <X>
mov     rcx, 0x8
idiv    rcx
mov     rax, rdx
```

x86-64: Compiler Optimizations

```
X % 8
```

```
mov rax, <X>
and rax, 7
```

x86-64: Compiler Optimizations

```
X % 26
```

```
mov     rax, <X>     48 8b 45 f8
mov     rcx, 26      b9 1a 00 00 00
div     rcx          48 f7 f1
mov     rax, rdx     48 89 d0
```

x86-64: Compiler Optimizations

```
X % 26
```

```
mov     rax, <X>      48 89 f8
push    26            6a 1a
pop     rcx           59
div     rcx           48 f7 f1
mov     rax, rdx      48 89 d0
```

x86-64: Compiler Optimizations

```
X % 26
```

```
movabs rcx, 0x4ec4ec4ec4ec4ec5
mov    rax, <X>
mul    rcx
shr    rdx, 0x3
lea    rax, [rdx+rdx*4]
lea    rax, [rax+rax*4]
add    rax, rdx
sub    <X>, rax
```

x86-64: Compiler Optimizations

$$X \equiv r \mod 26$$

$$X = 26q + r$$

$$X - 26q = r$$

$$X - 26\left\lfloor \frac{X}{26} \right\rfloor = r$$

```
movabs rcx, 0x4ec4ec4ec4ec4ec5
mov    rax, <X>
mul    rcx
shr    rdx, 0x3
lea    rax, [rdx+rdx*4]
lea    rax, [rax+rax*4]
add    rax, rdx
sub    <X>, rax
```

x86-64: Compiler Optimizations

$$X - 26\left\lfloor \frac{X}{26} \right\rfloor = r$$

```
movabs rcx, 0x4ec4ec4ec4ec4ec5
mov    rax, <X>
mul    rcx
shr    rdx, 0x3
lea    rax, [rdx+rdx*4]
lea    rax, [rax+rax*4]
add    rax, rdx
sub    <X>, rax
```

Calling Convention

x86-64: Calling Convention

A calling convention defines how registers should be used when calling a function.

This convention depends on:

1. The architecture
2. The operating system

It also defines which registers must be preserved when calling functions.

x86-64: Calling Convention

A calling convention defines how registers should be used when calling a function.

```
int x = 1;
int y = 2;
int result = compute(x, y);
```

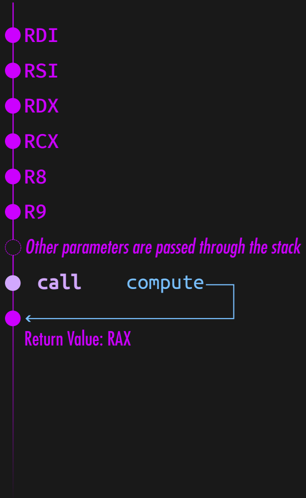
This convention depends on:

1. The architecture
2. The operating system

```
mov     dword ptr [rbp-0Ch], 1
mov     dword ptr [rbp-8],   2
mov     edx, [rbp-8]
mov     eax, [rbp-0Ch]
mov     esi, edx
mov     edi, eax
call    compute
mov     [rbp-4], eax
```

It also defines which registers must be preserved when calling functions.

x86-64: Calling Convention



RDI

RSI

RDX

RCX

R8

R9

Other parameters are passed through the stack

```
call    compute
```

Return Value: RAX

```
int x = 1;
int y = 2;
int result = compute(x, y);
```

```
mov     dword ptr [rbp-0Ch], 1
mov     dword ptr [rbp-8],   2
mov     edx, [rbp-8]
mov     eax, [rbp-0Ch]
mov     esi, edx
mov     edi, eax
call    compute
mov     [rbp-4], eax
```

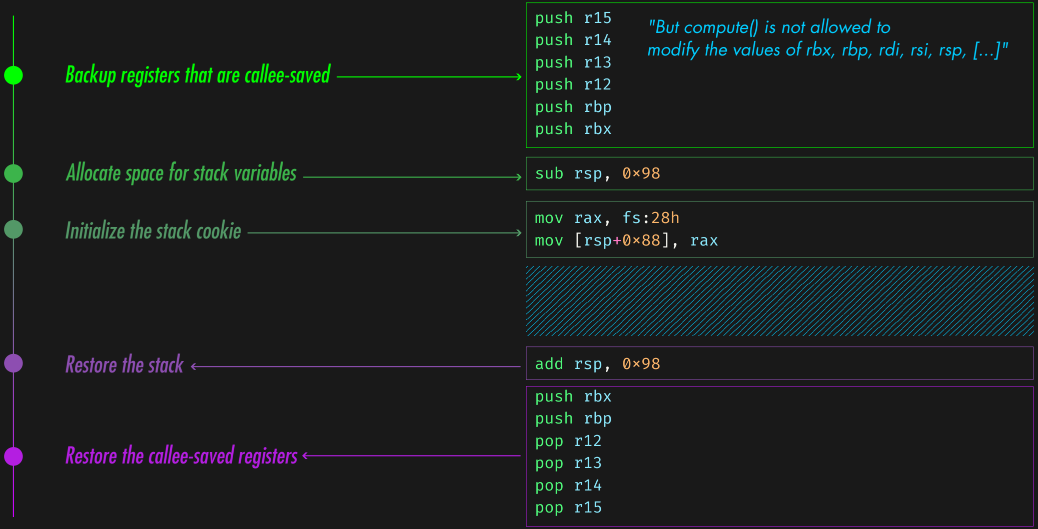
x86-64: Prologue / Epilogue

When calling compute(), the calling convention allows computes() to modify rax, rcx, rdx, r8, r9, r10, r11.

But compute() is not allowed to
modify the values of rbx, rbp, rdi, rsi, rsp, r12, r13, r14, r15.

```
int x = 1;
int y = 2;
int result = compute(x, y);
```

x86-64: Prologue / Epilogue



Backup registers that are callee-saved

```
push r15
push r14
push r13
push r12
push rbp
push rbx
```

"But compute() is not allowed to modify the values of rbx, rbp, rdi, rsi, rsp, [...]"

Allocate space for stack variables

```
sub rsp, 0×98
```

Initialize the stack cookie

```
mov rax, fs:28h
mov [rsp+0×88], rax
```

Restore the stack

```
add rsp, 0×98
```

Restore the callee-saved registers

```
push rbx
push rbp
pop r12
pop r13
pop r14
pop r15
```

x86-64: Endianness

x86-64 is a little-endian architecture which means that the least significant byte is stored at the "highest" memory address.

Basically, the memory representation is reversed compared to the in-register representation.

```c
uintptr_t* memory = ...;
*memory = 0x11223344;
```

44	33	22	11	00	00	00	00

12	45	32	AE	E3	00	32	11

```
mov     rax, [rbp+var_8]
```

RAX = 0x113200E3AE324512

Linux Execution Bootstrapping

Linux Execution Bootstrapping

All executables must define a `main()` function which is the *first* function being executed when the executable starts.

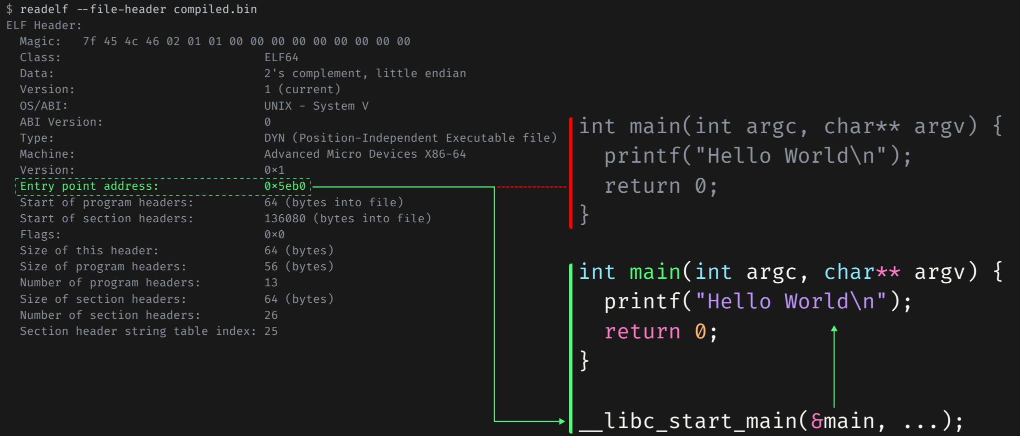
```c
int main(int argc, char** argv) {
  printf("Hello World\n");
  return 0;
}
```

Linux Execution Bootstrapping

```
$ readelf --file-header compiled.bin
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:               0x5eb0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          136080 (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:         26
  Section header string table index: 25
```

```c
int main(int argc, char** argv) {
  printf("Hello World\n");
  return 0;
}
```

Linux Execution Bootstrapping

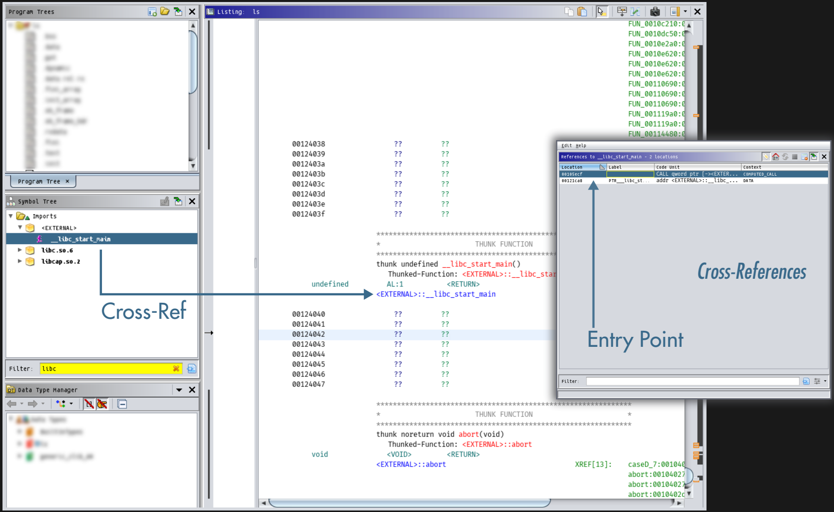


```
$ readelf --file-header compiled.bin
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Position-Independent Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x5eb0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          136080 (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:         26
  Section header string table index: 25
```

```c
int main(int argc, char** argv) {
  printf("Hello World\n");
  return 0;
}
```

```c
int main(int argc, char** argv) {
  printf("Hello World\n");
  return 0;
}

__libc_start_main(&main, ...);
```

Linux Execution Bootstrapping



Cross-Ref

Cross-References

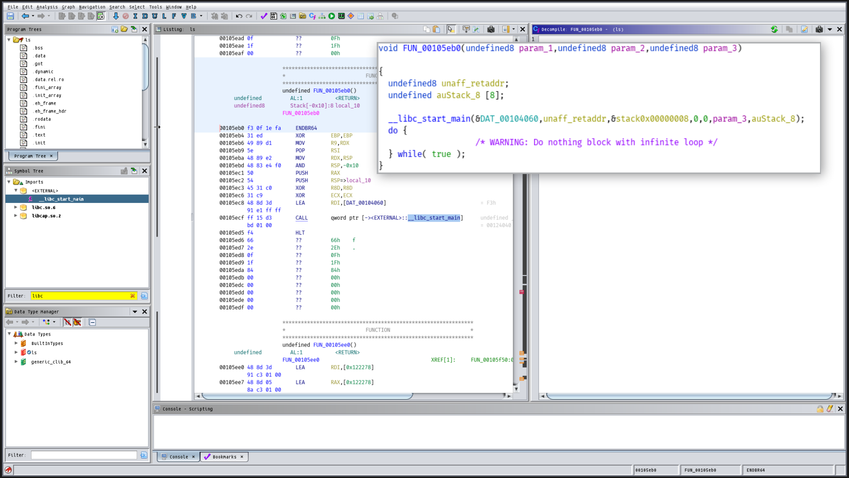
Entry Point

```
                                                          FUN_0010c210:0
                                                          FUN_0010dc50:0
                                                          FUN_0010e2a0:0
                                                          FUN_0010e620:0
                                                          FUN_0010e620:0
                                                          FUN_0010e620:0
                                                          FUN_00110690:0
                                                          FUN_00110690:0
                                                          FUN_00110690:0
                                                          FUN_001119a0:0
                                                          FUN_001119a0:0
                                                          FUN_00114480:0

00124038            ??          ??
00124039            ??          ??
0012403a            ??          ??
0012403b            ??          ??
0012403c            ??          ??
0012403d            ??          ??
0012403e            ??          ??
0012403f            ??          ??

                 *****************************************************
                 *                    THUNK FUNCTION
                 *****************************************************
                 thunk undefined __libc_start_main()
                   Thunked-Function: <EXTERNAL>::__libc_star
       undefined   AL:1         <RETURN>
                 <EXTERNAL>::__libc_start_main

00124040            ??          ??
00124041            ??          ??
00124042            ??          ??
00124043            ??          ??
00124044            ??          ??
00124045            ??          ??
00124046            ??          ??
00124047            ??          ??

                 **************************************************************
                 *                    THUNK FUNCTION                          *
                 **************************************************************
                 thunk noreturn void abort(void)
                   Thunked-Function: <EXTERNAL>::abort
       void        <VOID>       <RETURN>
                 <EXTERNAL>::abort                    XREF[13]:   caseD_7:001040
                                                                  abort:0010402
                                                                  abort:0010402
                                                                  abort:0010402
```

References to __libc_start_main - 2 locations

Location	Label	Code Unit	Context
00105ecf		CALL qword ptr [-><EXTER...	COMPUTED_CALL
00121ca8	PTR___libc_st...	addr <EXTERNAL>::__libc_...	DATA

Linux Execution Bootstrapping



```
void FUN_00105eb0(undefined8 param_1,undefined8 param_2,undefined8 param_3)

{
  undefined8 unaff_retaddr;
  undefined auStack_8 [8];

  __libc_start_main(&DAT_00104060,unaff_retaddr,&stack0x00000008,0,0,param_3,auStack_8);
  do {
                    /* WARNING: Do nothing block with infinite loop */
  } while( true );
}
```

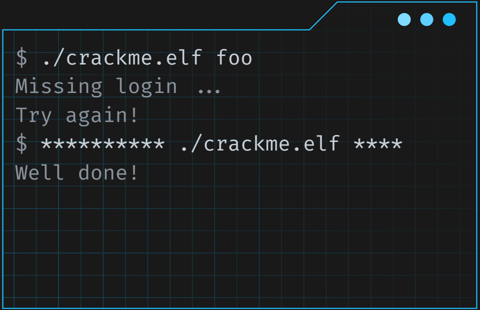
```
        00105ead 0f            ??          0Fh
        00105eae 1f            ??          1Fh
        00105eaf 00            ??          00h

                         ***************************************
                         *                                   FUNC
                         ***************************************
                         undefined FUN_00105eb0()
             undefined     AL:1          <RETURN>
             undefined8    Stack[-0x10]:8 local_10
                         FUN_00105eb0

        00105eb0 f3 0f 1e fa   ENDBR64
        00105eb4 31 ed         XOR         EBP,EBP
        00105eb6 49 89 d1      MOV         R9,RDX
        00105eb9 5e            POP         RSI
        00105eba 48 89 e2      MOV         RDX,RSP
        00105ebd 48 83 e4 f0   AND         RSP,-0x10
        00105ec1 50            PUSH        RAX
        00105ec2 54            PUSH        RSP=>local_10
        00105ec3 45 31 c0      XOR         R8D,R8D
        00105ec6 31 c9         XOR         ECX,ECX
        00105ec8 48 8d 3d      LEA         RDI,[DAT_00104060]                        = F3h
                 91 e1 ff ff
        00105ecf ff 15 d3      CALL        qword ptr [-><EXTERNAL>::__libc_start_main]    undefined
                 bd 01 00                                                                = 00124040
        00105ed5 f4            HLT
        00105ed6 66            ??          66h    f
        00105ed7 2e            ??          2Eh    .
        00105ed8 0f            ??          0Fh
        00105ed9 1f            ??          1Fh
        00105eda 84            ??          84h
        00105edb 00            ??          00h
        00105edc 00            ??          00h
        00105edd 00            ??          00h
        00105ede 00            ??          00h
        00105edf 00            ??          00h

                         **************************************************************
                         *                          FUNCTION                          *
                         **************************************************************
                         undefined FUN_00105ee0()
             undefined     AL:1          <RETURN>
                         FUN_00105ee0                              XREF[1]:     FUN_00105f50:0
        00105ee0 48 8d 3d      LEA         RDI,[0x122278]
                 91 c3 01 00
        00105ee7 48 8d 05      LEA         RAX,[0x122278]
                 8a c3 01 00
```

Demo

1

Hands-on #1: Simple Crackme

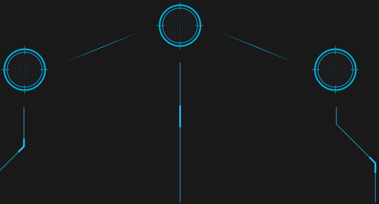
1



```
$ ./crackme.elf foo
Missing login ...
Try again!
$ ********** ./crackme.elf ****
Well done!
```

Level: Easy

Objectives: Getting started with reverse engineering and compiler optimizations.



ELF x86-64

Involves Basic Mathematics

Not Stripped

Hints

- http://flaviojslab.blogspot.com/2008/02/integer-division.html
- The charset of the password is `abcdefghijklmnopqrstuvwxyz` (lower case)
- What is the priority between xor and add ?

Reverse Engineering Structures

Reverse Engineering Structures

Reverse engineering is not always about understanding a function or an algorithm.

It might also involve understanding complex data types like structures or C++ classes.

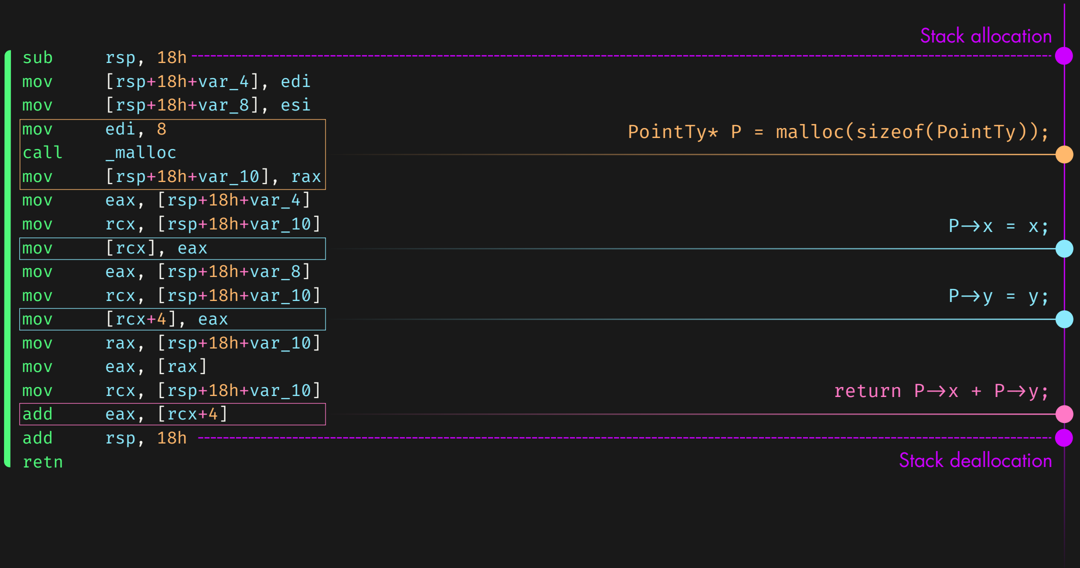
```
struct PointTy{
  int x;
  int y;
};

int compute(int x, int y) {
  PointTy* P = malloc(sizeof(PointTy));
  P->x = x;
  P->y = y;
  return P->x + P->y;
}
```

Reverse Engineering Structures

```
sub     rsp, 18h
mov     [rsp+18h+var_4], edi
mov     [rsp+18h+var_8], esi
mov     edi, 8
call    _malloc
mov     [rsp+18h+var_10], rax
mov     eax, [rsp+18h+var_4]
mov     rcx, [rsp+18h+var_10]
mov     [rcx], eax
mov     eax, [rsp+18h+var_8]
mov     rcx, [rsp+18h+var_10]
mov     [rcx+4], eax
mov     rax, [rsp+18h+var_10]
mov     eax, [rax]
mov     rcx, [rsp+18h+var_10]
add     eax, [rcx+4]
add     rsp, 18h
retn
```

```
struct PointTy {
  int x;
  int y;
};

int compute(int x, int y) {
  PointTy* P = malloc(sizeof(PointTy));
  P->x = x;
  P->y = y;
  return P->x + P->y;
}
```

Reverse Engineering Structures



```
sub     rsp, 18h
mov     [rsp+18h+var_4], edi
mov     [rsp+18h+var_8], esi
mov     edi, 8
call    _malloc
mov     [rsp+18h+var_10], rax
mov     eax, [rsp+18h+var_4]
mov     rcx, [rsp+18h+var_10]
mov     [rcx], eax
mov     eax, [rsp+18h+var_8]
mov     rcx, [rsp+18h+var_10]
mov     [rcx+4], eax
mov     rax, [rsp+18h+var_10]
mov     eax, [rax]
mov     rcx, [rsp+18h+var_10]
add     eax, [rcx+4]
add     rsp, 18h
retn
```

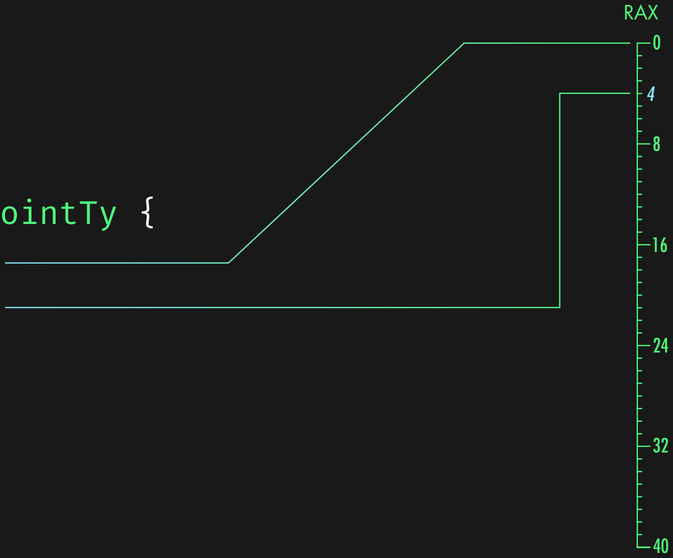
Stack allocation

```
PointTy* P = malloc(sizeof(PointTy));
P->x = x;
P->y = y;
return P->x + P->y;
```

Stack deallocation

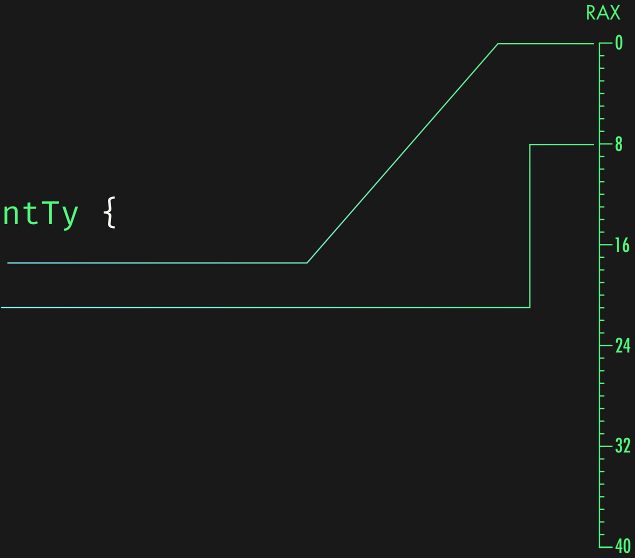
Reverse Engineering Structures

```
struct PointTy {
  int x;
  int y;
};
```



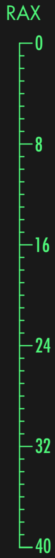
Reverse Engineering Structures

```
struct PointTy {
  void* x;
  void* y;
};
```



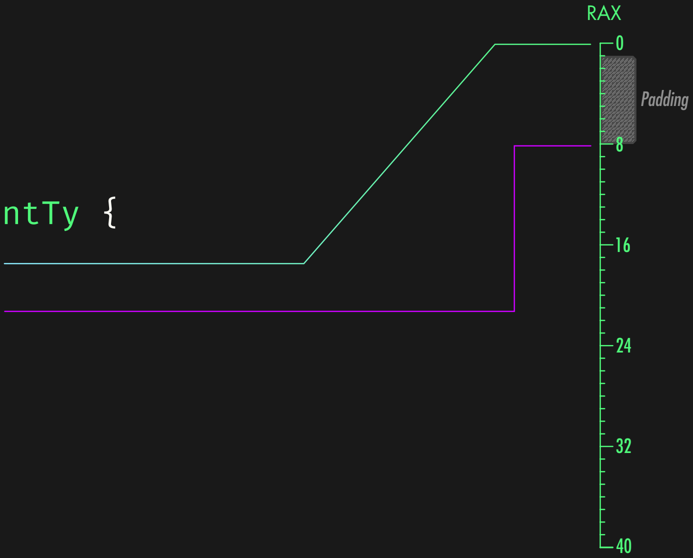
Reverse Engineering Structures

```
struct PointTy {
  char  x;
  void* y;
};
```



Reverse Engineering Structures

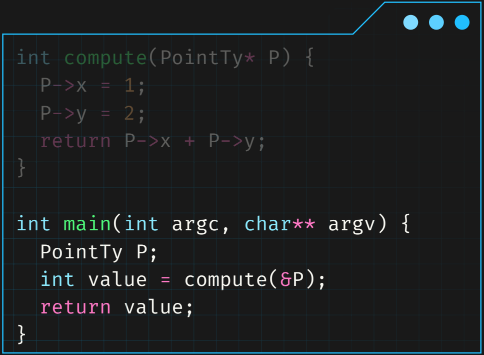
```
struct PointTy {
  char  x;
  void* y;
};
```



Reverse Engineering Structures

```c
int compute(PointTy* P) {
  P->x = 1;
  P->y = 2;
  return P->x + P->y;
}

int main(int argc, char** argv) {
  PointTy P;
  int value = compute(&P);
  return value;
}
```

Reverse Engineering Structures

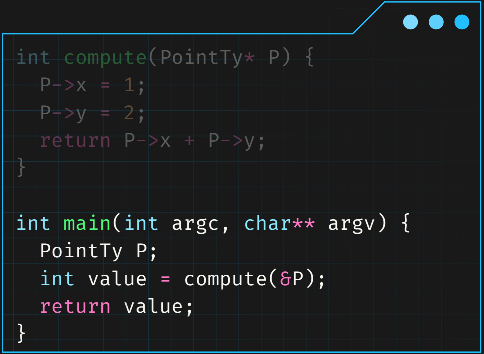


```
int compute(PointTy* P) {
  P->x = 1;
  P->y = 2;
  return P->x + P->y;
}

int main(int argc, char** argv) {
  PointTy P;
  int value = compute(&P);
  return value;
}
```

```
$ clang++ -O0 [ ... ]
```

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0x20
MOV         dword ptr [RBP + local_c],0x0
MOV         dword ptr [RBP + local_10],EDI
MOV         qword ptr [RBP + local_18],RSI
LEA         RDI=>local_20,[RBP + -0x18]
CALL        FUN_00101120
MOV         dword ptr [RBP + local_24],EAX
MOV         EAX,dword ptr [RBP + local_24]
ADD         RSP,0x20
POP         RBP
RET
```

Reverse Engineering Structures

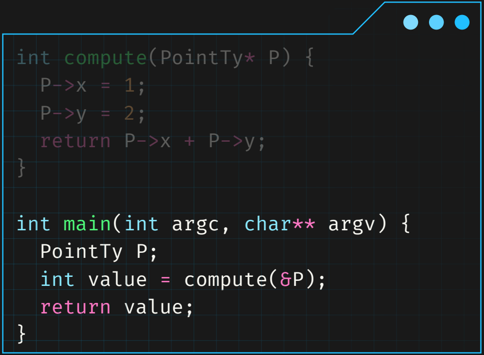


```
int compute(PointTy* P) {
  P->x = 1;
  P->y = 2;
  return P->x + P->y;
}

int main(int argc, char** argv) {
  PointTy P;
  int value = compute(&P);
  return value;
}
```

```
$ clang++ -O0 [ ... ]
```

```
// main
undefined4 FUN_00101150(undefined4 param_1, ... ) {
  undefined4 uVar1;
  undefined  local_20 [8];
  undefined8 local_18;
  undefined4 local_10;
  undefined4 local_c;

  local_c = 0;
  local_18 = param_2;
  local_10 = param_1;
  uVar1 = FUN_00101120(local_20);
  return uVar1;
}
```

Reverse Engineering Structures

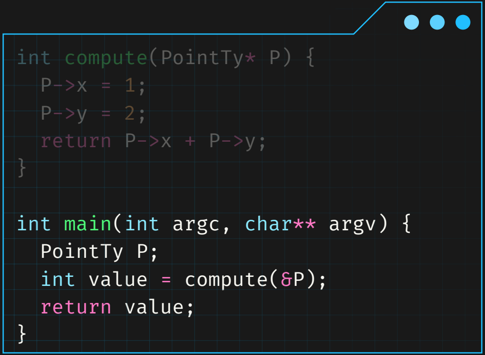


```cpp
int compute(PointTy* P) {
  P->x = 1;
  P->y = 2;
  return P->x + P->y;
}

int main(int argc, char** argv) {
  PointTy P;
  int value = compute(&P);
  return value;
}
```

```
$ clang++ -O1 [ ... ]
```

```
PUSH       RAX
MOV        RDI,RSP
CALL       FUN_00101120
POP        RCX
RET
```

Reverse Engineering Structures

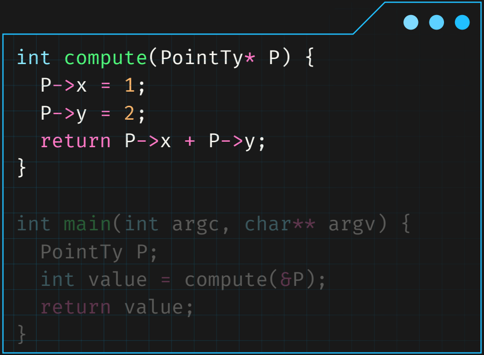


```
int compute(PointTy* P) {
  P->x = 1;
  P->y = 2;
  return P->x + P->y;
}

int main(int argc, char** argv) {
  PointTy P;
  int value = compute(&P);
  return value;
}
```

```
$ clang++ -O1 [ … ]
```

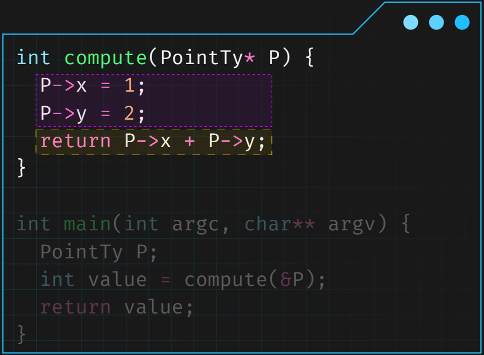
```
// main
void FUN_00101150(void) {
  undefined auStack_8[8];

  FUN_00101120(auStack_8);
  return;
}
```

Reverse Engineering Structures



```
int compute(PointTy* P) {
  P->x = 1;
  P->y = 2;
  return P->x + P->y;
}

int main(int argc, char** argv) {
  PointTy P;
  int value = compute(&P);
  return value;
}
```

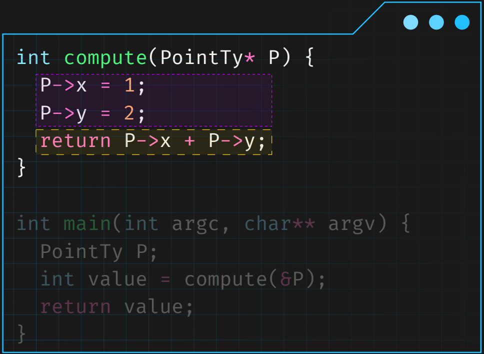
Reverse Engineering Structures



```c
int compute(PointTy* P) {
   P->x = 1;
   P->y = 2;
   return P->x + P->y;
}

int main(int argc, char** argv) {
   PointTy P;
   int value = compute(&P);
   return value;
}
```

```
MOV        qword ptr [RSP + local_8],RDI
MOV        RAX,qword ptr [RSP + local_8]
MOV        dword ptr [RAX],0x1
MOV        RAX,qword ptr [RSP + local_8]
MOV        dword ptr [RAX + 0x4],0x2
MOV        RAX,qword ptr [RSP + local_8]
MOV        EAX,dword ptr [RAX]
MOV        RCX,qword ptr [RSP + local_8]
ADD        EAX,dword ptr [RCX + 0x4]
RET
```

Reverse Engineering Structures



```
int compute(PointTy* P) {
  P->x = 1;
  P->y = 2;
  return P->x + P->y;
}

int main(int argc, char** argv) {
  PointTy P;
  int value = compute(&P);
  return value;
}
```

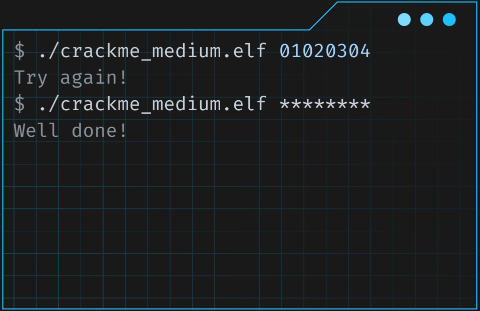
```
// compute
int FUN_00101120(int *param_1) {
  *param_1 = 1;
  param_1[1] = 2;

  return *param_1 + param_1[1];
}
```

Demo

2

Hands-on #2: Structures

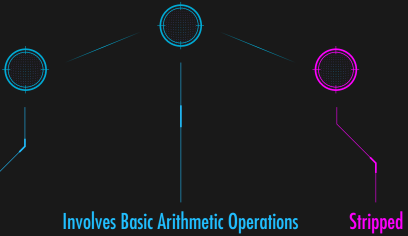
2



```
$ ./crackme_medium.elf 01020304
Try again!
$ ./crackme_medium.elf ********
Well done!
```

Level: Medium

Objectives: Identify and reverse structures



ELF x86-64

Involves Basic Arithmetic Operations

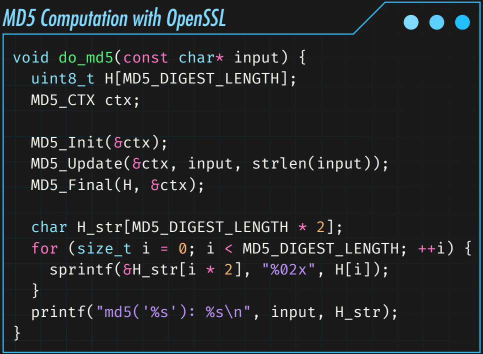
Stripped

Reverse Engineering Large Binaries

Reverse Engineering Large Binaries

Most of the programs rely on third-party libraries that can be dynamically or statically linked.

Reverse Engineering Large Binaries



MD5 Computation with OpenSSL

```
void do_md5(const char* input) {
  uint8_t H[MD5_DIGEST_LENGTH];
  MD5_CTX ctx;

  MD5_Init(&ctx);
  MD5_Update(&ctx, input, strlen(input));
  MD5_Final(H, &ctx);

  char H_str[MD5_DIGEST_LENGTH * 2];
  for (size_t i = 0; i < MD5_DIGEST_LENGTH; ++i) {
    sprintf(&H_str[i * 2], "%02x", H[i]);
  }
  printf("md5('%s'): %s\n", input, H_str);
}
```

OpenSSL
Cryptography and SSL/TLS Toolkit

Reverse Engineering Large Binaries

MD5 Computation with OpenSSL

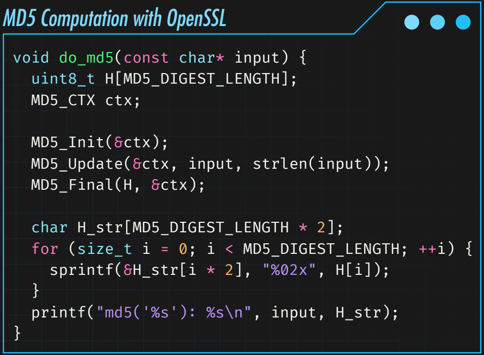
```
void do_md5(const char* input) {
  uint8_t H[MD5_DIGEST_LENGTH];
  MD5_CTX ctx;

  MD5_Init(&ctx);
  MD5_Update(&ctx, input, strlen(input));
  MD5_Final(H, &ctx);

  char H_str[MD5_DIGEST_LENGTH * 2];
  for (size_t i = 0; i < MD5_DIGEST_LENGTH; ++i) {
    sprintf(&H_str[i * 2], "%02x", H[i]);
  }
  printf("md5('%s'): %s\n", input, H_str);
}
```

```
$ clang -O0 main.cpp -o main -lcrypto
```

↑

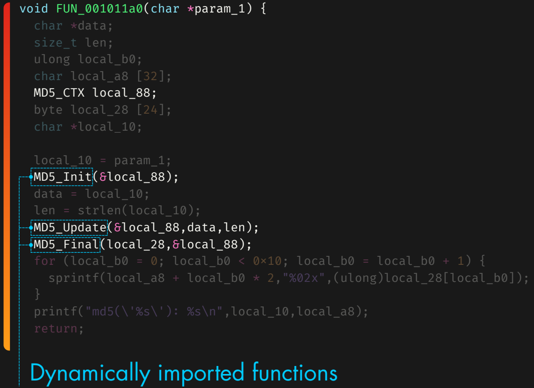
Dynamic link with OpenSSL

Reverse Engineering Large Binaries



MD5 Computation with OpenSSL

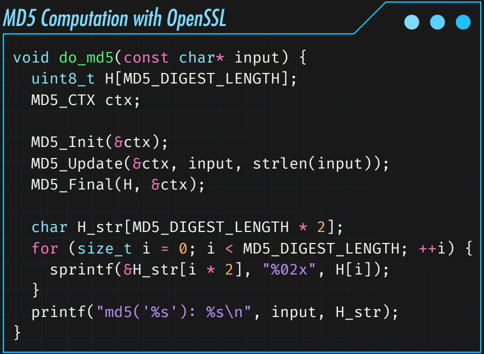
```
void do_md5(const char* input) {
  uint8_t H[MD5_DIGEST_LENGTH];
  MD5_CTX ctx;

  MD5_Init(&ctx);
  MD5_Update(&ctx, input, strlen(input));
  MD5_Final(H, &ctx);

  char H_str[MD5_DIGEST_LENGTH * 2];
  for (size_t i = 0; i < MD5_DIGEST_LENGTH; ++i) {
    sprintf(&H_str[i * 2], "%02x", H[i]);
  }
  printf("md5('%s'): %s\n", input, H_str);
}
```



```
void FUN_001011a0(char *param_1) {
  char *data;
  size_t len;
  ulong local_b0;
  char local_a8 [32];
  MD5_CTX local_88;
  byte local_28 [24];
  char *local_10;

  local_10 = param_1;
  MD5_Init(&local_88);
  data = local_10;
  len = strlen(local_10);
  MD5_Update(&local_88,data,len);
  MD5_Final(local_28,&local_88);
  for (local_b0 = 0; local_b0 < 0x10; local_b0 = local_b0 + 1) {
    sprintf(local_a8 + local_b0 * 2,"%02x",(ulong)local_28[local_b0]);
  }
  printf("md5(\'%s\'): %s\n",local_10,local_a8);
  return;
```

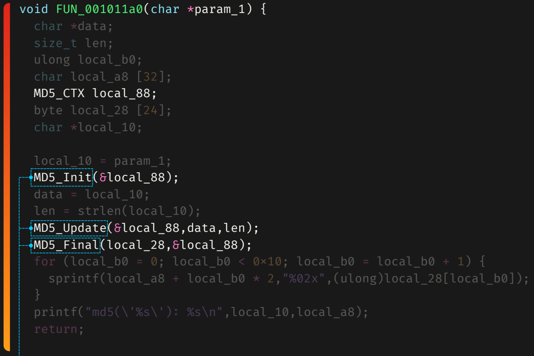
Dynamically imported functions

Reverse Engineering Large Binaries



MD5 Computation with OpenSSL

```c
void do_md5(const char* input) {
  uint8_t H[MD5_DIGEST_LENGTH];
  MD5_CTX ctx;

  MD5_Init(&ctx);
  MD5_Update(&ctx, input, strlen(input));
  MD5_Final(H, &ctx);

  char H_str[MD5_DIGEST_LENGTH * 2];
  for (size_t i = 0; i < MD5_DIGEST_LENGTH; ++i) {
    sprintf(&H_str[i * 2], "%02x", H[i]);
  }
  printf("md5('%s'): %s\n", input, H_str);
}
```

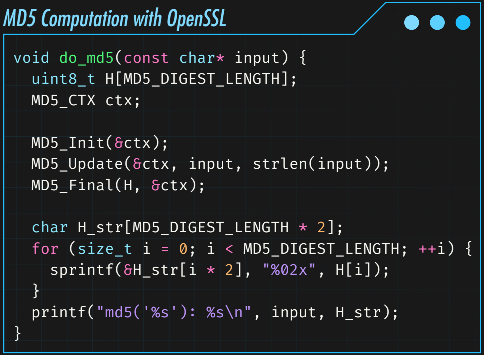


```c
void FUN_001011a0(char *param_1) {
  char *data;
  size_t len;
  ulong local_b0;
  char local_a8 [32];
  MD5_CTX local_88;
  byte local_28 [24];
  char *local_10;

  local_10 = param_1;
  MD5_Init(&local_88);
  data = local_10;
  len = strlen(local_10);
  MD5_Update(&local_88,data,len);
  MD5_Final(local_28,&local_88);
  for (local_b0 = 0; local_b0 < 0x10; local_b0 = local_b0 + 1) {
    sprintf(local_a8 + local_b0 * 2,"%02x",(ulong)local_28[local_b0]);
  }
  printf("md5(\'%s\'): %s\n",local_10,local_a8);
  return;
```

Dynamically imported functions

Can't be stripped

Reverse Engineering Large Binaries



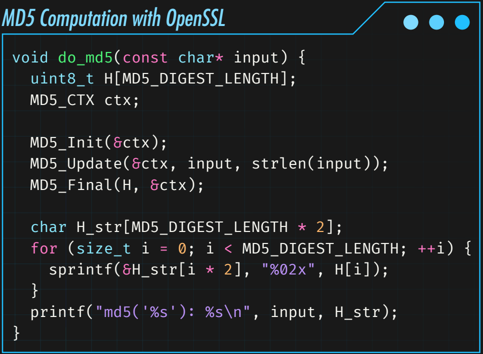
MD5 Computation with OpenSSL

```
void do_md5(const char* input) {
  uint8_t H[MD5_DIGEST_LENGTH];
  MD5_CTX ctx;

  MD5_Init(&ctx);
  MD5_Update(&ctx, input, strlen(input));
  MD5_Final(H, &ctx);

  char H_str[MD5_DIGEST_LENGTH * 2];
  for (size_t i = 0; i < MD5_DIGEST_LENGTH; ++i) {
    sprintf(&H_str[i * 2], "%02x", H[i]);
  }
  printf("md5('%s'): %s\n", input, H_str);
}
```

```
$ clang -O0 main.cpp -o main libcrypto.a
```

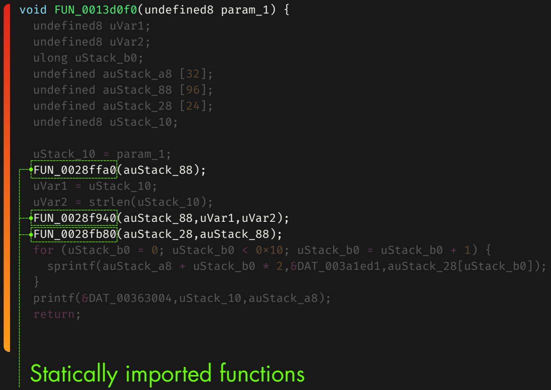
Static link with OpenSSL

Reverse Engineering Large Binaries



MD5 Computation with OpenSSL

```c
void do_md5(const char* input) {
  uint8_t H[MD5_DIGEST_LENGTH];
  MD5_CTX ctx;

  MD5_Init(&ctx);
  MD5_Update(&ctx, input, strlen(input));
  MD5_Final(H, &ctx);

  char H_str[MD5_DIGEST_LENGTH * 2];
  for (size_t i = 0; i < MD5_DIGEST_LENGTH; ++i) {
    sprintf(&H_str[i * 2], "%02x", H[i]);
  }
  printf("md5('%s'): %s\n", input, H_str);
}
```



```c
void FUN_0013d0f0(undefined8 param_1) {
  undefined8 uVar1;
  undefined8 uVar2;
  ulong uStack_b0;
  undefined auStack_a8 [32];
  undefined auStack_88 [96];
  undefined auStack_28 [24];
  undefined8 uStack_10;

  uStack_10 = param_1;
  FUN_0028ffa0(auStack_88);
  uVar1 = uStack_10;
  uVar2 = strlen(uStack_10);
  FUN_0028f940(auStack_88,uVar1,uVar2);
  FUN_0028fb80(auStack_28,auStack_88);
  for (uStack_b0 = 0; uStack_b0 < 0x10; uStack_b0 = uStack_b0 + 1) {
    sprintf(auStack_a8 + uStack_b0 * 2,&DAT_003a1ed1,auStack_28[uStack_b0]);
  }
  printf(&DAT_00363004,uStack_10,auStack_a8);
  return;
```

Statically imported functions

Reverse Engineering Large Binaries

Dynamic Link

```c
void FUN_001011a0(char *param_1) {
  char *data;
  size_t len;
  ulong local_b0;
  char local_a8 [32];
  MD5_CTX local_88;
  byte local_28 [24];
  char *local_10;

  local_10 = param_1;
  MD5_Init(&local_88);
  data = local_10;
  len = strlen(local_10);
  MD5_Update(&local_88,data,len);
  MD5_Final(local_28,&local_88);
  for (local_b0 = 0; local_b0 < 0x10; local_b0 = local_b0 + 1) {
    sprintf(local_a8 + local_b0 * 2,"%02x",(ulong)local_28[local_b0]);
  }
  printf("md5(\'%s\'): %s\n",local_10,local_a8);
  return;
}
```

Static Link (and stripped)

```c
void FUN_0013d0f0(undefined8 param_1) {
  undefined8 uVar1;
  undefined8 uVar2;
  ulong uStack_b0;
  undefined auStack_a8 [32];
  undefined auStack_88 [96];
  undefined auStack_28 [24];
  undefined8 uStack_10;

  uStack_10 = param_1;
  FUN_0028ffa0(auStack_88);
  uVar1 = uStack_10;
  uVar2 = strlen(uStack_10);
  FUN_0028f940(auStack_88,uVar1,uVar2);
  FUN_0028fb80(auStack_28,auStack_88);
  for (uStack_b0 = 0; uStack_b0 < 0x10; uStack_b0 = uStack_b0 + 1) {
    sprintf(auStack_a8 + uStack_b0 * 2,&DAT_003a1ed1,auStack_28[uStack_b0]);
  }
  printf(&DAT_00363004,uStack_10,auStack_a8);
  return;
}
```

Reverse Engineering Large Binaries

When a library is statically linked and the program correctly stripped[1], the reverse engineering of the whole binary can be challenging.

[1] this step is error prone

Reverse Engineering Large Binaries

When a library is statically linked and the program correctly stripped[1], the reverse engineering of the whole binary can be challenging.

⇒ We can quickly get lost while trying to analyze the binary.

[1] this step is error prone

Reverse Engineering Large Binaries

Dynamic Link

- Smaller binary size
- Require that the user has the library with the correct version
- Easier to reverse

Static Link (and stripped)

```
void FUN_0013d0f0(undefined8 param_1) {
  undefined8 uVar1;
  undefined8 uVar2;
  ulong uStack_b0;
  undefined auStack_a8 [32];
  undefined auStack_88 [96];
  undefined auStack_28 [24];
  undefined8 uStack_10;

  uStack_10 = param_1;
  FUN_0028ffa0(auStack_88);
  uVar1 = uStack_10;
  uVar2 = strlen(uStack_10);
  FUN_0028f940(auStack_88,uVar1,uVar2);
  FUN_0028fb80(auStack_28,auStack_88);
  for (uStack_b0 = 0; uStack_b0 < 0x10; uStack_b0 = uStack_b0 + 1) {
    sprintf(auStack_a8 + uStack_b0 * 2,&DAT_003a1ed1,auStack_28[uStack_b0]);
  }
  printf(&DAT_00363004,uStack_10,auStack_a8);
  return;
}
```

Reverse Engineering Large Binaries

Dynamic Link

- Smaller binary size
- Require that the user has the library with the correct version
- Easier to reverse

Static Link (and stripped)

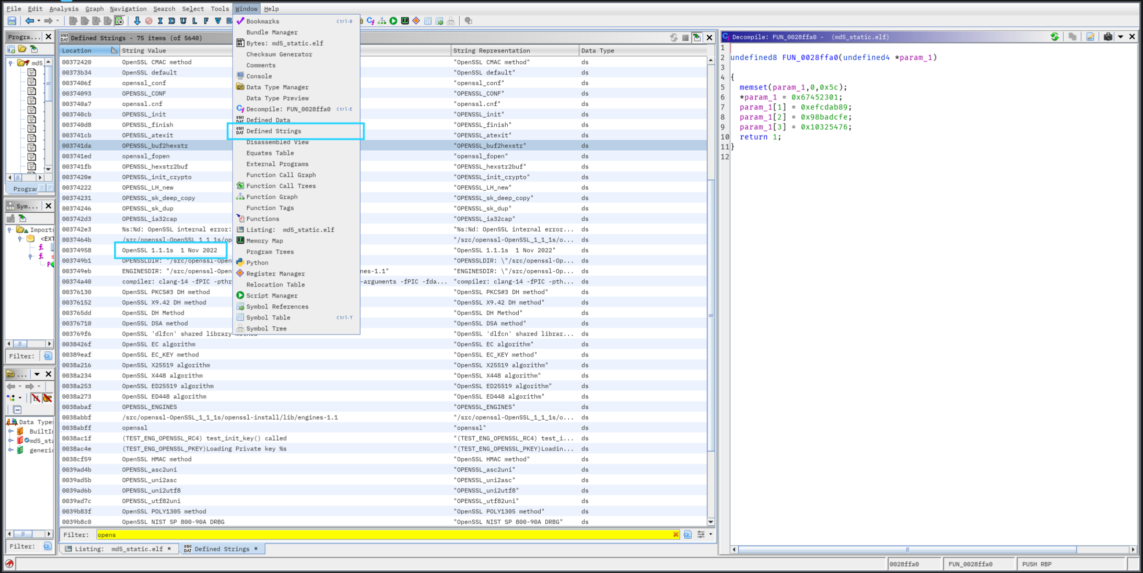
- Larger binary size
- The library is embedded in the binary
- Can be challenging to reverse

Reverse Engineering Large Binaries

- Strings, logs
- Constants
- Functions relative position

1. Strings

Strings



Defined Strings - 75 items (of 5640)

Location	String Value	String Representation	Data Type
00372420	OpenSSL CMAC method	"OpenSSL CMAC method"	ds
00373b34	OpenSSL default	"OpenSSL default"	ds
0037406f	openssl_conf	"openssl_conf"	ds
00374093	OPENSSL_CONF	"OPENSSL_CONF"	ds
003740a7	openssl.cnf	"openssl.cnf"	ds
003740cb	OPENSSL_init	"OPENSSL_init"	ds
003740d8	OPENSSL_finish	"OPENSSL_finish"	ds
003741cb	OPENSSL_atexit	"OPENSSL_atexit"	ds
003741da	OPENSSL_buf2hexstr	"OPENSSL_buf2hexstr"	ds
003741ed	openssl_fopen	"openssl_fopen"	ds
003741fb	OPENSSL_hexstr2buf	"OPENSSL_hexstr2buf"	ds
0037420e	OPENSSL_init_crypto	"OPENSSL_init_crypto"	ds
00374222	OPENSSL_LH_new	"OPENSSL_LH_new"	ds
00374231	OPENSSL_sk_deep_copy	"OPENSSL_sk_deep_copy"	ds
00374246	OPENSSL_sk_dup	"OPENSSL_sk_dup"	ds
003742d3	OPENSSL_ia32cap	"OPENSSL_ia32cap"	ds
003742e3	%s:%d: OpenSSL internal error:	"%s:%d: OpenSSL internal error...	ds
0037464b	/src/openssl-OpenSSL_1_1_1s/op	"/src/openssl-OpenSSL_1_1_1s/o...	ds
00374958	OpenSSL 1.1.1s 1 Nov 2022	"OpenSSL 1.1.1s 1 Nov 2022"	ds
003749b1	OPENSSLDIR: "/src/openssl-Open	"OPENSSLDIR: \"/src/openssl-Op...	ds
003749eb	ENGINESDIR: "/src/openssl-Open nes-1.1"	"ENGINESDIR: \"/src/openssl-Op...	ds
00374a40	compiler: clang-14 -fPIC -pthr -arguments -fPIC -fda...	"compiler: clang-14 -fPIC -pth...	ds
00376130	OpenSSL PKCS#3 DH method	"OpenSSL PKCS#3 DH method"	ds
00376152	OpenSSL X9.42 DH method	"OpenSSL X9.42 DH method"	ds
003765dd	OpenSSL DH Method	"OpenSSL DH Method"	ds
00376710	OpenSSL DSA method	"OpenSSL DSA method"	ds
003769f6	OpenSSL 'dlfcn' shared library method	"OpenSSL 'dlfcn' shared librar...	ds
0038426f	OpenSSL EC algorithm	"OpenSSL EC algorithm"	ds
00389eaf	OpenSSL EC_KEY method	"OpenSSL EC_KEY method"	ds
0038a216	OpenSSL X25519 algorithm	"OpenSSL X25519 algorithm"	ds
0038a234	OpenSSL X448 algorithm	"OpenSSL X448 algorithm"	ds
0038a253	OpenSSL ED25519 algorithm	"OpenSSL ED25519 algorithm"	ds
0038a273	OpenSSL ED448 algorithm	"OpenSSL ED448 algorithm"	ds
0038abaf	OPENSSL_ENGINES	"OPENSSL_ENGINES"	ds
0038abbf	/src/openssl-OpenSSL_1_1_1s/openssl-install/lib/engines-1.1	"/src/openssl-OpenSSL_1_1_1s/o...	ds
0038abff	openssl	"openssl"	ds
0038ac1f	(TEST_ENG_OPENSSL_RC4) test_init_key() called	"(TEST_ENG_OPENSSL_RC4) test_i...	ds
0038ac4e	(TEST_ENG_OPENSSL_PKEY)Loading Private key %s	"(TEST_ENG_OPENSSL_PKEY)Loadin...	ds
0038cf59	OpenSSL HMAC method	"OpenSSL HMAC method"	ds
0039ad4b	OPENSSL_asc2uni	"OPENSSL_asc2uni"	ds
0039ad5b	OPENSSL_uni2asc	"OPENSSL_uni2asc"	ds
0039ad6b	OPENSSL_uni2utf8	"OPENSSL_uni2utf8"	ds
0039ad7c	OPENSSL_utf82uni	"OPENSSL_utf82uni"	ds
0039b83f	OpenSSL POLY1305 method	"OpenSSL POLY1305 method"	ds
0039b8c0	OpenSSL NIST SP 800-90A DRBG	"OpenSSL NIST SP 800-90A DRBG"	ds

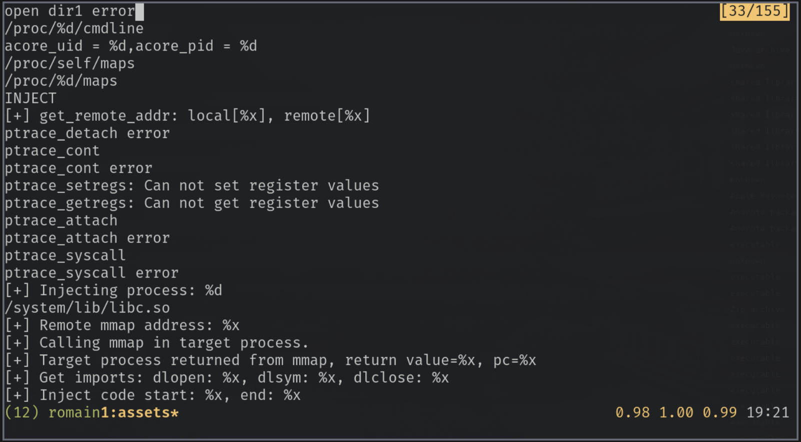
Filter: opens

Decompile: FUN_0028ffa0 - (md5_static.elf)

```

undefined8 FUN_0028ffa0(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

0028ffa0 | FUN_0028ffa0 | PUSH RBP

Strings

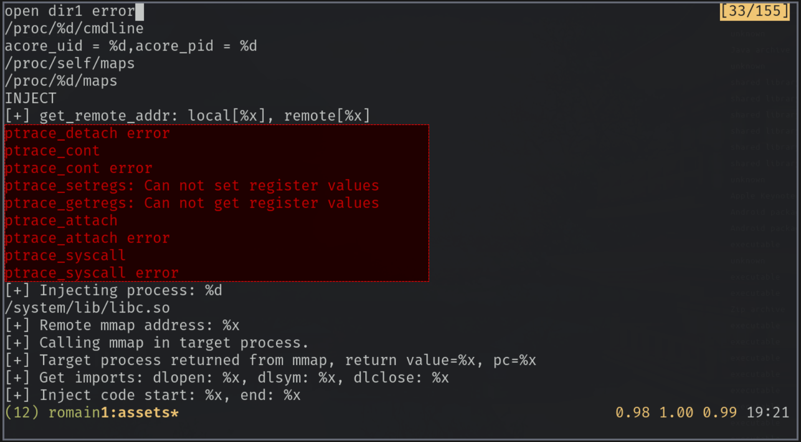


```
open dir1 error                                                                        [33/155]
/proc/%d/cmdline
acore_uid = %d,acore_pid = %d
/proc/self/maps
/proc/%d/maps
INJECT
[+] get_remote_addr: local[%x], remote[%x]
ptrace_detach error
ptrace_cont
ptrace_cont error
ptrace_setregs: Can not set register values
ptrace_getregs: Can not get register values
ptrace_attach
ptrace_attach error
ptrace_syscall
ptrace_syscall error
[+] Injecting process: %d
/system/lib/libc.so
[+] Remote mmap address: %x
[+] Calling mmap in target process.
[+] Target process returned from mmap, return value=%x, pc=%x
[+] Get imports: dlopen: %x, dlsym: %x, dlclose: %x
[+] Inject code start: %x, end: %x
(12) romain1:assets*                                                        0.98 1.00 0.99 19:21
```

assets/injectso_arm

List of strings in one of the libraries used in the Android Pegasus spyware.

Strings



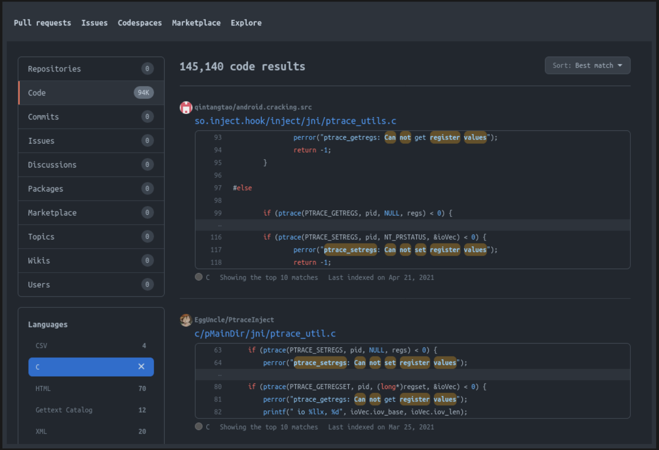
```
open dir1 error                                                        [33/155]
/proc/%d/cmdline
acore_uid = %d,acore_pid = %d
/proc/self/maps
/proc/%d/maps
INJECT
[+] get_remote_addr: local[%x], remote[%x]
ptrace_detach error
ptrace_cont
ptrace_cont error
ptrace_setregs: Can not set register values
ptrace_getregs: Can not get register values
ptrace_attach
ptrace_attach error
ptrace_syscall
ptrace_syscall error
[+] Injecting process: %d
/system/lib/libc.so
[+] Remote mmap address: %x
[+] Calling mmap in target process.
[+] Target process returned from mmap, return value=%x, pc=%x
[+] Get imports: dlopen: %x, dlsym: %x, dlclose: %x
[+] Inject code start: %x, end: %x
(12) romain1:assets*                                       0.98 1.00 0.99 19:21
```

List of strings in one of the libraries used in the Android Pegasus spyware.

assets/injectso_arm

Strings

```
ptrace_detach error
ptrace_cont
ptrace_cont error
ptrace_setregs: Can not set register values
ptrace_getregs: Can not get register values
ptrace_attach
ptrace_attach error
ptrace_syscall
ptrace_syscall error
```



145,140 code results

qintangtao/android.cracking.src

so.inject.hook/inject/jni/ptrace_utils.c

```
                perror("ptrace_getregs: Can not get register values");
                return -1;
        }

#else

        if (ptrace(PTRACE_GETREGS, pid, NULL, regs) < 0) {
…
        if (ptrace(PTRACE_SETREGS, pid, NT_PRSTATUS, &ioVec) < 0) {
                perror("ptrace_setregs: Can not set register values");
                return -1;
```

C Showing the top 10 matches Last indexed on Apr 21, 2021

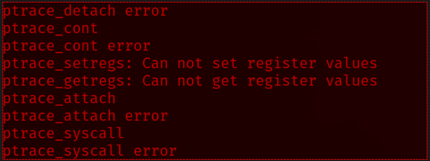
EggUncle/PtraceInject

c/pMainDir/jni/ptrace_util.c

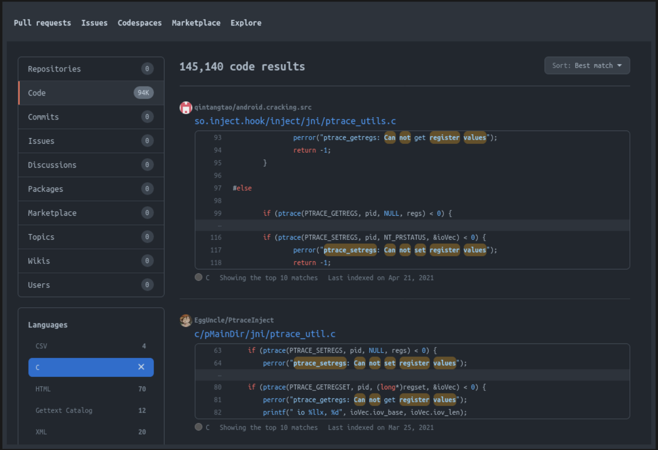
```
    if (ptrace(PTRACE_SETREGS, pid, NULL, regs) < 0) {
        perror("ptrace_setregs: Can not set register values");
…
    if (ptrace(PTRACE_GETREGSET, pid, (long*)regset, &ioVec) < 0) {
        perror("ptrace_getregs: Can not get register values");
        printf(" io %llx, %d", ioVec.iov_base, ioVec.iov_len);
```

C Showing the top 10 matches Last indexed on Mar 25, 2021

Strings



```
ptrace_detach error
ptrace_cont
ptrace_cont error
ptrace_setregs: Can not set register values
ptrace_getregs: Can not get register values
ptrace_attach
ptrace_attach error
ptrace_syscall
ptrace_syscall error
```



145,140 code results

qintangtao/android.cracking.src

so.inject.hook/inject/jni/ptrace_utils.c

```
                        perror("ptrace_getregs: Can not get register values");
                        return -1;
                }

#else

                if (ptrace(PTRACE_GETREGS, pid, NULL, regs) < 0) {
…
                if (ptrace(PTRACE_SETREGS, pid, NT_PRSTATUS, &ioVec) < 0) {
                        perror("ptrace_setregs: Can not set register values");
                        return -1;
```

C Showing the top 10 matches Last indexed on Apr 21, 2021

EggUncle/PtraceInject

c/pMainDir/jni/ptrace_util.c

```
        if (ptrace(PTRACE_SETREGS, pid, NULL, regs) < 0) {
            perror("ptrace_setregs: Can not set register values");
…
        if (ptrace(PTRACE_GETREGSET, pid, (long*)regset, &ioVec) < 0) {
            perror("ptrace_getregs: Can not get register values");
            printf(" io %llx, %d", ioVec.iov_base, ioVec.iov_len);
```

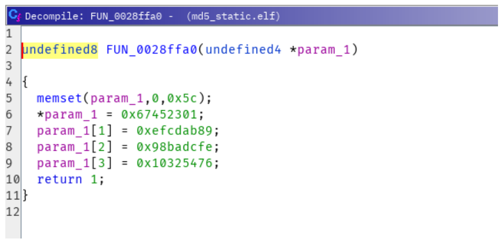
C Showing the top 10 matches Last indexed on Mar 25, 2021

Don't spend time on reversing open-source code!

2. Constants

Constants

```
void FUN_0013d0f0(undefined8 param_1) {
  undefined8 uVar1;
  undefined8 uVar2;
  ulong uStack_b0;
  undefined auStack_a8 [32];
  undefined auStack_88 [96];
  undefined auStack_28 [24];
  undefined8 uStack_10;

  uStack_10 = param_1;
  FUN_0028ffa0(auStack_88);
  uVar1 = uStack_10;
  uVar2 = strlen(uStack_10);
  FUN_0028f940(auStack_88,uVar1,uVar2);
  FUN_0028fb80(auStack_28,auStack_88);
  for (uStack_b0 = 0; uStack_b0 < 0x10; uStack_b0 = uStack_b0 + 1) {
    sprintf(auStack_a8 + uStack_b0 * 2,&DAT_003a1ed1,auStack_28[uStack_b0]);
  }
  printf(&DAT_00363004,uStack_10,auStack_a8);
  return;
```

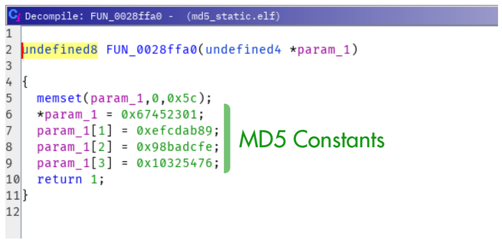


Decompile: FUN_0028ffa0 - (md5_static.elf)

```

undefined8 FUN_0028ffa0(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

Constants

```
void FUN_0013d0f0(undefined8 param_1) {
  undefined8 uVar1;
  undefined8 uVar2;
  ulong uStack_b0;
  undefined auStack_a8 [32];
  undefined auStack_88 [96];
  undefined auStack_28 [24];
  undefined8 uStack_10;

  uStack_10 = param_1;
  FUN_0028ffa0(auStack_88);
  uVar1 = uStack_10;
  uVar2 = strlen(uStack_10);
  FUN_0028f940(auStack_88,uVar1,uVar2);
  FUN_0028fb80(auStack_28,auStack_88);
  for (uStack_b0 = 0; uStack_b0 < 0x10; uStack_b0 = uStack_b0 + 1) {
    sprintf(auStack_a8 + uStack_b0 * 2,&DAT_003a1ed1,auStack_28[uStack_b0]);
  }
  printf(&DAT_00363004,uStack_10,auStack_a8);
  return;
```

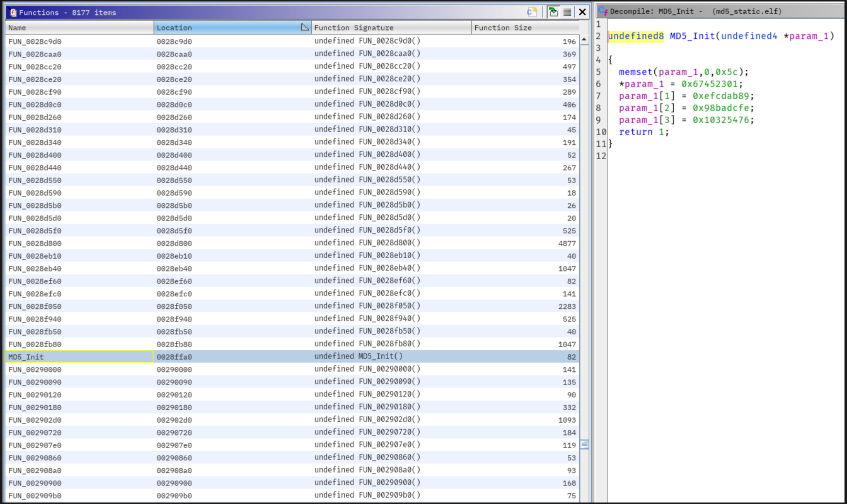


Decompile: FUN_0028ffa0 - (md5_static.elf)

```

undefined8 FUN_0028ffa0(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

MD5 Constants

Constants



Functions - 8177 items

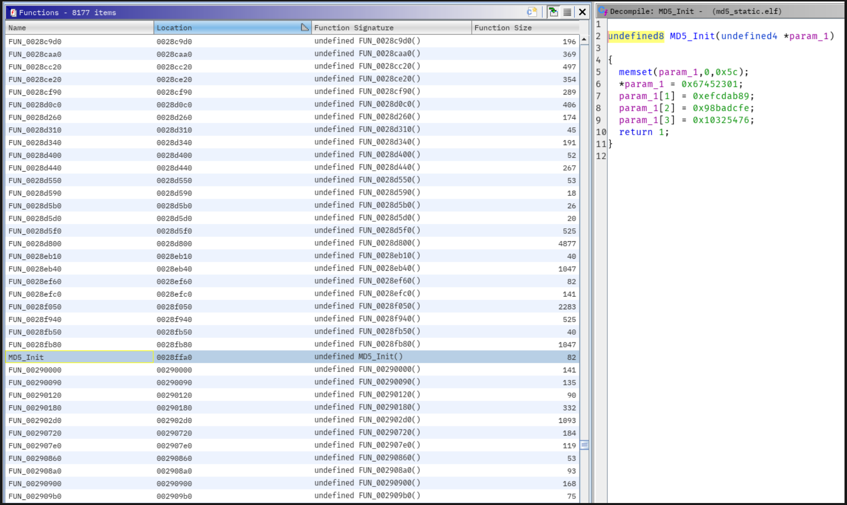
Name	Location	Function Signature	Function Size
FUN_0028c9d0	0028c9d0	undefined FUN_0028c9d0()	196
FUN_0028caa0	0028caa0	undefined FUN_0028caa0()	369
FUN_0028cc20	0028cc20	undefined FUN_0028cc20()	497
FUN_0028ce20	0028ce20	undefined FUN_0028ce20()	354
FUN_0028cf90	0028cf90	undefined FUN_0028cf90()	289
FUN_0028d0c0	0028d0c0	undefined FUN_0028d0c0()	406
FUN_0028d260	0028d260	undefined FUN_0028d260()	174
FUN_0028d310	0028d310	undefined FUN_0028d310()	45
FUN_0028d340	0028d340	undefined FUN_0028d340()	191
FUN_0028d400	0028d400	undefined FUN_0028d400()	52
FUN_0028d440	0028d440	undefined FUN_0028d440()	267
FUN_0028d550	0028d550	undefined FUN_0028d550()	53
FUN_0028d590	0028d590	undefined FUN_0028d590()	18
FUN_0028d5b0	0028d5b0	undefined FUN_0028d5b0()	26
FUN_0028d5d0	0028d5d0	undefined FUN_0028d5d0()	20
FUN_0028d5f0	0028d5f0	undefined FUN_0028d5f0()	525
FUN_0028d800	0028d800	undefined FUN_0028d800()	4877
FUN_0028eb10	0028eb10	undefined FUN_0028eb10()	40
FUN_0028eb40	0028eb40	undefined FUN_0028eb40()	1047
FUN_0028ef60	0028ef60	undefined FUN_0028ef60()	82
FUN_0028efc0	0028efc0	undefined FUN_0028efc0()	141
FUN_0028f050	0028f050	undefined FUN_0028f050()	2283
FUN_0028f940	0028f940	undefined FUN_0028f940()	525
FUN_0028fb50	0028fb50	undefined FUN_0028fb50()	40
FUN_0028fb80	0028fb80	undefined FUN_0028fb80()	1047
MD5_Init	0028ffa0	undefined MD5_Init()	82
FUN_00290000	00290000	undefined FUN_00290000()	141
FUN_00290090	00290090	undefined FUN_00290090()	135
FUN_00290120	00290120	undefined FUN_00290120()	90
FUN_00290180	00290180	undefined FUN_00290180()	332
FUN_002902d0	002902d0	undefined FUN_002902d0()	1093
FUN_00290720	00290720	undefined FUN_00290720()	184
FUN_002907e0	002907e0	undefined FUN_002907e0()	119
FUN_00290860	00290860	undefined FUN_00290860()	53
FUN_002908a0	002908a0	undefined FUN_002908a0()	93
FUN_00290900	00290900	undefined FUN_00290900()	168
FUN_002909b0	002909b0	undefined FUN_002909b0()	75

Decompile: MD5_Init - (md5_static.elf)

```

undefined8 MD5_Init(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

3. Relative Positioning

Constants



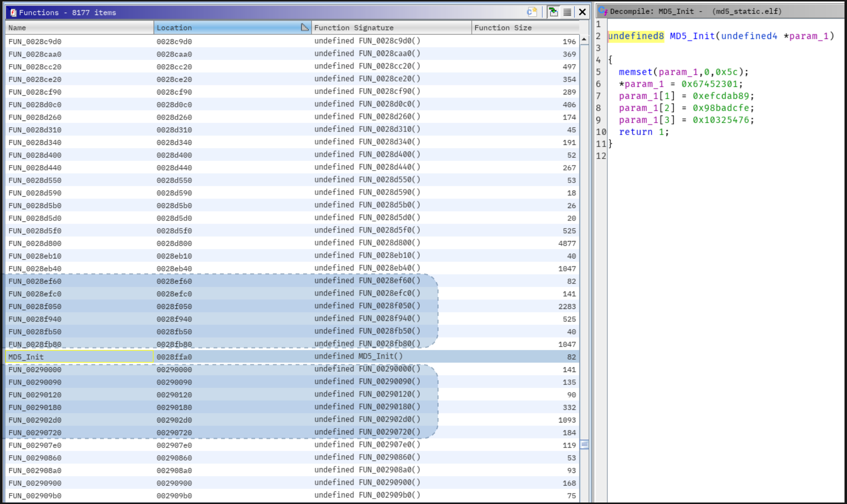
Functions - 8177 items

Name	Location	Function Signature	Function Size
FUN_0028c9d0	0028c9d0	undefined FUN_0028c9d0()	196
FUN_0028caa0	0028caa0	undefined FUN_0028caa0()	369
FUN_0028cc20	0028cc20	undefined FUN_0028cc20()	497
FUN_0028ce20	0028ce20	undefined FUN_0028ce20()	354
FUN_0028cf90	0028cf90	undefined FUN_0028cf90()	289
FUN_0028d0c0	0028d0c0	undefined FUN_0028d0c0()	406
FUN_0028d260	0028d260	undefined FUN_0028d260()	174
FUN_0028d310	0028d310	undefined FUN_0028d310()	45
FUN_0028d340	0028d340	undefined FUN_0028d340()	191
FUN_0028d400	0028d400	undefined FUN_0028d400()	52
FUN_0028d440	0028d440	undefined FUN_0028d440()	267
FUN_0028d550	0028d550	undefined FUN_0028d550()	53
FUN_0028d590	0028d590	undefined FUN_0028d590()	18
FUN_0028d5b0	0028d5b0	undefined FUN_0028d5b0()	26
FUN_0028d5d0	0028d5d0	undefined FUN_0028d5d0()	20
FUN_0028d5f0	0028d5f0	undefined FUN_0028d5f0()	525
FUN_0028d800	0028d800	undefined FUN_0028d800()	4877
FUN_0028eb10	0028eb10	undefined FUN_0028eb10()	40
FUN_0028eb40	0028eb40	undefined FUN_0028eb40()	1047
FUN_0028ef60	0028ef60	undefined FUN_0028ef60()	82
FUN_0028efc0	0028efc0	undefined FUN_0028efc0()	141
FUN_0028f050	0028f050	undefined FUN_0028f050()	2283
FUN_0028f940	0028f940	undefined FUN_0028f940()	525
FUN_0028fb50	0028fb50	undefined FUN_0028fb50()	40
FUN_0028fb80	0028fb80	undefined FUN_0028fb80()	1047
MD5_Init	0028ffa0	undefined MD5_Init()	82
FUN_00290000	00290000	undefined FUN_00290000()	141
FUN_00290090	00290090	undefined FUN_00290090()	135
FUN_00290120	00290120	undefined FUN_00290120()	90
FUN_00290180	00290180	undefined FUN_00290180()	332
FUN_002902d0	002902d0	undefined FUN_002902d0()	1093
FUN_00290720	00290720	undefined FUN_00290720()	184
FUN_002907e0	002907e0	undefined FUN_002907e0()	119
FUN_00290860	00290860	undefined FUN_00290860()	53
FUN_002908a0	002908a0	undefined FUN_002908a0()	93
FUN_00290900	00290900	undefined FUN_00290900()	168
FUN_002909b0	002909b0	undefined FUN_002909b0()	75

Decompile: MD5_Init - (md5_static.elf)

```

undefined8 MD5_Init(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

Relative Positioning



Name	Location	Function Signature	Function Size
FUN_0028c9d0	0028c9d0	undefined FUN_0028c9d0()	196
FUN_0028caa0	0028caa0	undefined FUN_0028caa0()	369
FUN_0028cc20	0028cc20	undefined FUN_0028cc20()	497
FUN_0028ce20	0028ce20	undefined FUN_0028ce20()	354
FUN_0028cf90	0028cf90	undefined FUN_0028cf90()	289
FUN_0028d0c0	0028d0c0	undefined FUN_0028d0c0()	406
FUN_0028d260	0028d260	undefined FUN_0028d260()	174
FUN_0028d310	0028d310	undefined FUN_0028d310()	45
FUN_0028d340	0028d340	undefined FUN_0028d340()	191
FUN_0028d400	0028d400	undefined FUN_0028d400()	52
FUN_0028d440	0028d440	undefined FUN_0028d440()	267
FUN_0028d550	0028d550	undefined FUN_0028d550()	53
FUN_0028d590	0028d590	undefined FUN_0028d590()	18
FUN_0028d5b0	0028d5b0	undefined FUN_0028d5b0()	26
FUN_0028d5d0	0028d5d0	undefined FUN_0028d5d0()	20
FUN_0028d5f0	0028d5f0	undefined FUN_0028d5f0()	525
FUN_0028d800	0028d800	undefined FUN_0028d800()	4877
FUN_0028eb10	0028eb10	undefined FUN_0028eb10()	40
FUN_0028eb40	0028eb40	undefined FUN_0028eb40()	1047
FUN_0028ef60	0028ef60	undefined FUN_0028ef60()	82
FUN_0028efc0	0028efc0	undefined FUN_0028efc0()	141
FUN_0028f050	0028f050	undefined FUN_0028f050()	2283
FUN_0028f940	0028f940	undefined FUN_0028f940()	525
FUN_0028fb50	0028fb50	undefined FUN_0028fb50()	40
FUN_0028fb80	0028fb80	undefined FUN_0028fb80()	1047
MD5_Init	0028ffa0	undefined MD5_Init()	82
FUN_00290000	00290000	undefined FUN_00290000()	141
FUN_00290090	00290090	undefined FUN_00290090()	135
FUN_00290120	00290120	undefined FUN_00290120()	90
FUN_00290180	00290180	undefined FUN_00290180()	332
FUN_002902d0	002902d0	undefined FUN_002902d0()	1093
FUN_00290720	00290720	undefined FUN_00290720()	184
FUN_002907e0	002907e0	undefined FUN_002907e0()	119
FUN_00290860	00290860	undefined FUN_00290860()	53
FUN_002908a0	002908a0	undefined FUN_002908a0()	93
FUN_00290900	00290900	undefined FUN_00290900()	168
FUN_002909b0	002909b0	undefined FUN_002909b0()	75

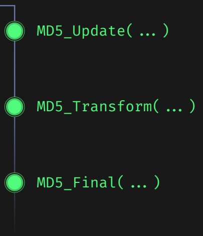
```

undefined8 MD5_Init(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

Likely MD5-related functions

Likely MD5-related functions

Relative Positioning

```c
#include <stdio.h>
#include "md5_local.h"
#include <openssl/opensslv.h>

/*
 * Implemented from RFC1321 The MD5 Message-Digest Algorithm
 */

#define INIT_DATA_A (unsigned long)0x67452301L
#define INIT_DATA_B (unsigned long)0xefcdab89L
#define INIT_DATA_C (unsigned long)0x98badcfeL
#define INIT_DATA_D (unsigned long)0x10325476L

int MD5_Init(MD5_CTX *c)
{
    memset(c, 0, sizeof(*c));
    c->A = INIT_DATA_A;
    c->B = INIT_DATA_B;
    c->C = INIT_DATA_C;
    c->D = INIT_DATA_D;
    return 1;
}
```



MD5_Update(...)

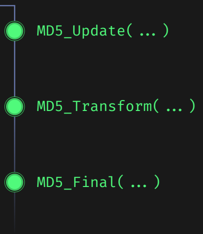
MD5_Transform(...)

MD5_Final(...)

OpenSSL_1_1_1s/crypto/md5/md5_dgst.c

Relative Positioning

```c
#include <stdio.h>
#include "md5_local.h"
#include <openssl/opensslv.h>

/*
 * Implemented from RFC1321 The MD5 Message-Digest Algorithm
 */

#define INIT_DATA_A (unsigned long)0x67452301L
#define INIT_DATA_B (unsigned long)0xefcdab89L
#define INIT_DATA_C (unsigned long)0x98badcfeL
#define INIT_DATA_D (unsigned long)0x10325476L

int MD5_Init(MD5_CTX *c)
{
    memset(c, 0, sizeof(*c));
    c->A = INIT_DATA_A;
    c->B = INIT_DATA_B;
    c->C = INIT_DATA_C;
    c->D = INIT_DATA_D;
    return 1;
}
```



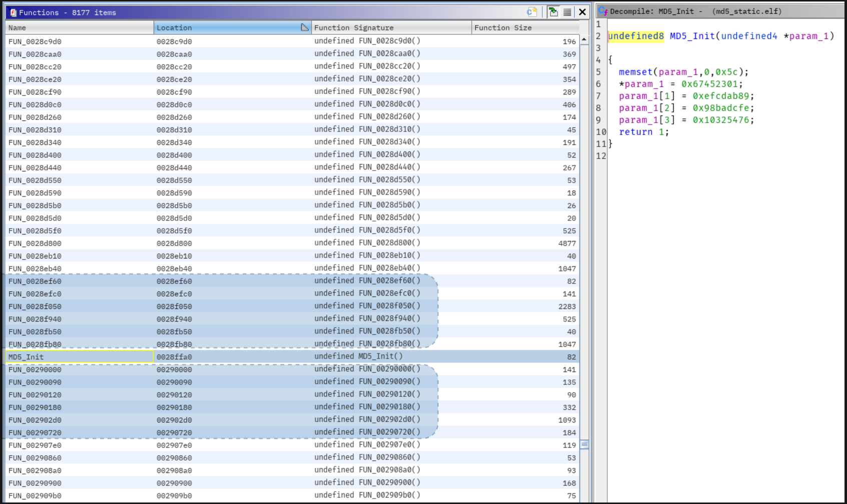
```
$ readelf -SW ./md5_dgst.o
Section Headers:
  [Nr] Name                Type       Off    Size   ES Flg Lk Inf Al
  [ 3] .text.MD5_Update    PROGBITS   000040 00020d 00  AX  0   0 16
  [ 5] .text.MD5_Transform PROGBITS   000250 000028 00  AX  0   0 16
  [ 7] .text.MD5_Final     PROGBITS   000280 000417 00  AX  0   0 16
  [ 9] .text.MD5_Init      PROGBITS   0006a0 000052 00  AX  0   0 16
```

OpenSSL_1_1_1s/crypto/md5/md5_dgst.c

Relative Positioning

Likely MD5-related functions

Likely MD5-related functions



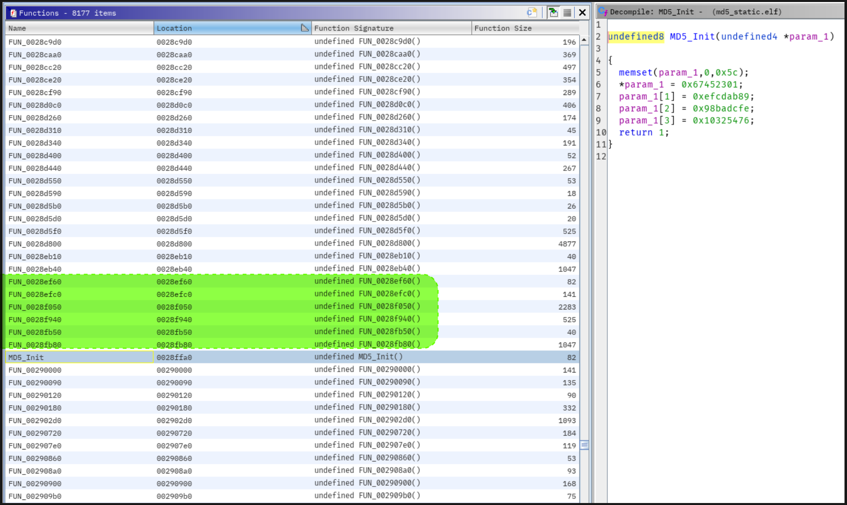
Functions - 8177 items

Name	Location	Function Signature	Function Size
FUN_0028c9d0	0028c9d0	undefined FUN_0028c9d0()	196
FUN_0028caa0	0028caa0	undefined FUN_0028caa0()	369
FUN_0028cc20	0028cc20	undefined FUN_0028cc20()	497
FUN_0028ce20	0028ce20	undefined FUN_0028ce20()	354
FUN_0028cf90	0028cf90	undefined FUN_0028cf90()	289
FUN_0028d0c0	0028d0c0	undefined FUN_0028d0c0()	406
FUN_0028d260	0028d260	undefined FUN_0028d260()	174
FUN_0028d310	0028d310	undefined FUN_0028d310()	45
FUN_0028d340	0028d340	undefined FUN_0028d340()	191
FUN_0028d400	0028d400	undefined FUN_0028d400()	52
FUN_0028d440	0028d440	undefined FUN_0028d440()	267
FUN_0028d550	0028d550	undefined FUN_0028d550()	53
FUN_0028d590	0028d590	undefined FUN_0028d590()	18
FUN_0028d5b0	0028d5b0	undefined FUN_0028d5b0()	26
FUN_0028d5d0	0028d5d0	undefined FUN_0028d5d0()	20
FUN_0028d5f0	0028d5f0	undefined FUN_0028d5f0()	525
FUN_0028d800	0028d800	undefined FUN_0028d800()	4877
FUN_0028eb10	0028eb10	undefined FUN_0028eb10()	40
FUN_0028eb40	0028eb40	undefined FUN_0028eb40()	1047
FUN_0028ef60	0028ef60	undefined FUN_0028ef60()	82
FUN_0028efc0	0028efc0	undefined FUN_0028efc0()	141
FUN_0028f050	0028f050	undefined FUN_0028f050()	2283
FUN_0028f940	0028f940	undefined FUN_0028f940()	525
FUN_0028fb50	0028fb50	undefined FUN_0028fb50()	40
FUN_0028fb80	0028fb80	undefined FUN_0028fb80()	1047
MD5_Init	0028ffa0	undefined MD5_Init()	82
FUN_00290000	00290000	undefined FUN_00290000()	141
FUN_00290090	00290090	undefined FUN_00290090()	135
FUN_00290120	00290120	undefined FUN_00290120()	90
FUN_00290180	00290180	undefined FUN_00290180()	332
FUN_002902d0	002902d0	undefined FUN_002902d0()	1093
FUN_00290720	00290720	undefined FUN_00290720()	184
FUN_002907e0	002907e0	undefined FUN_002907e0()	119
FUN_00290860	00290860	undefined FUN_00290860()	53
FUN_002908a0	002908a0	undefined FUN_002908a0()	93
FUN_00290900	00290900	undefined FUN_00290900()	168
FUN_002909b0	002909b0	undefined FUN_002909b0()	75

Decompile: MD5_Init - (md5_static.elf)

```

undefined8 MD5_Init(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

Relative Positioning



Functions - 8177 items

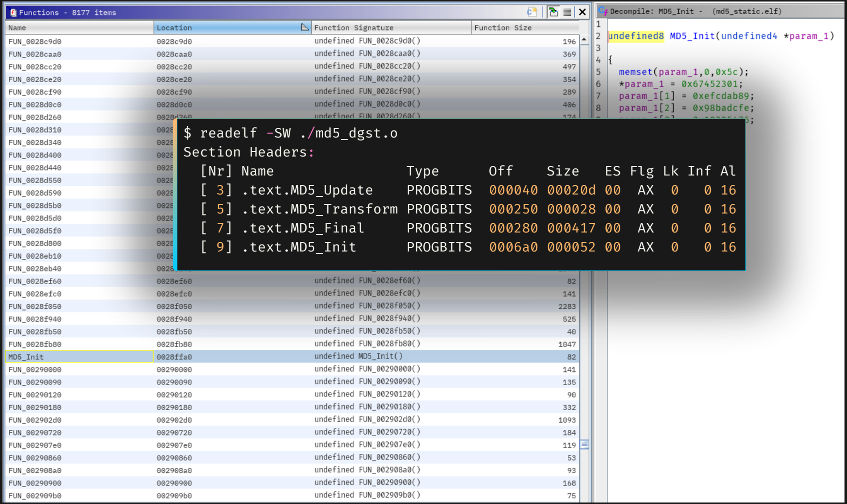
Name	Location	Function Signature	Function Size
FUN_0028c9d0	0028c9d0	undefined FUN_0028c9d0()	196
FUN_0028caa0	0028caa0	undefined FUN_0028caa0()	369
FUN_0028cc20	0028cc20	undefined FUN_0028cc20()	497
FUN_0028ce20	0028ce20	undefined FUN_0028ce20()	354
FUN_0028cf90	0028cf90	undefined FUN_0028cf90()	289
FUN_0028d0c0	0028d0c0	undefined FUN_0028d0c0()	406
FUN_0028d260	0028d260	undefined FUN_0028d260()	174
FUN_0028d310	0028d310	undefined FUN_0028d310()	45
FUN_0028d340	0028d340	undefined FUN_0028d340()	191
FUN_0028d400	0028d400	undefined FUN_0028d400()	52
FUN_0028d440	0028d440	undefined FUN_0028d440()	267
FUN_0028d550	0028d550	undefined FUN_0028d550()	53
FUN_0028d590	0028d590	undefined FUN_0028d590()	18
FUN_0028d5b0	0028d5b0	undefined FUN_0028d5b0()	26
FUN_0028d5d0	0028d5d0	undefined FUN_0028d5d0()	20
FUN_0028d5f0	0028d5f0	undefined FUN_0028d5f0()	525
FUN_0028d800	0028d800	undefined FUN_0028d800()	4877
FUN_0028eb10	0028eb10	undefined FUN_0028eb10()	40
FUN_0028eb40	0028eb40	undefined FUN_0028eb40()	1047
FUN_0028ef60	0028ef60	undefined FUN_0028ef60()	82
FUN_0028efc0	0028efc0	undefined FUN_0028efc0()	141
FUN_0028f050	0028f050	undefined FUN_0028f050()	2283
FUN_0028f940	0028f940	undefined FUN_0028f940()	525
FUN_0028fb50	0028fb50	undefined FUN_0028fb50()	40
FUN_0028fb80	0028fb80	undefined FUN_0028fb80()	1047
MD5_Init	0028ffa0	undefined MD5_Init()	82
FUN_00290000	00290000	undefined FUN_00290000()	141
FUN_00290090	00290090	undefined FUN_00290090()	135
FUN_00290120	00290120	undefined FUN_00290120()	90
FUN_00290180	00290180	undefined FUN_00290180()	332
FUN_002902d0	002902d0	undefined FUN_002902d0()	1093
FUN_00290720	00290720	undefined FUN_00290720()	184
FUN_002907e0	002907e0	undefined FUN_002907e0()	119
FUN_00290860	00290860	undefined FUN_00290860()	53
FUN_002908a0	002908a0	undefined FUN_002908a0()	93
FUN_00290900	00290900	undefined FUN_00290900()	168
FUN_002909b0	002909b0	undefined FUN_002909b0()	75

Likely MD5-related functions

Decompile: MD5_Init - (md5_static.elf)

```

undefined8 MD5_Init(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
  return 1;
}

```

Relative Positioning



Functions - 8177 items

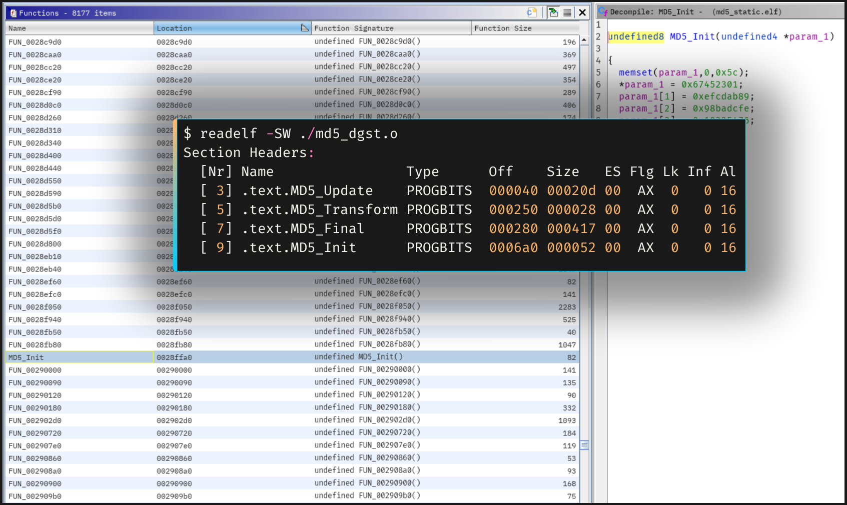
Name	Location	Function Signature	Function Size
FUN_0028c9d0	0028c9d0	undefined FUN_0028c9d0()	196
FUN_0028caa0	0028caa0	undefined FUN_0028caa0()	369
FUN_0028cc20	0028cc20	undefined FUN_0028cc20()	497
FUN_0028ce20	0028ce20	undefined FUN_0028ce20()	354
FUN_0028cf90	0028cf90	undefined FUN_0028cf90()	289
FUN_0028d0c0	0028d0c0	undefined FUN_0028d0c0()	406
FUN_0028d260	0028d260	undefined FUN_0028d260()	174
FUN_0028d310	0028		
FUN_0028d340	0028		
FUN_0028d400	0028		
FUN_0028d440	0028		
FUN_0028d550	0028		
FUN_0028d590	0028		
FUN_0028d5b0	0028		
FUN_0028d5d0	0028		
FUN_0028d5f0	0028		
FUN_0028d800	0028		
FUN_0028eb10	0028		
FUN_0028eb40	0028		
FUN_0028ef60	0028ef60	undefined FUN_0028ef60()	82
FUN_0028efc0	0028efc0	undefined FUN_0028efc0()	141
FUN_0028f050	0028f050	undefined FUN_0028f050()	2283
FUN_0028f940	0028f940	undefined FUN_0028f940()	525
FUN_0028fb50	0028fb50	undefined FUN_0028fb50()	40
FUN_0028fb80	0028fb80	undefined FUN_0028fb80()	1047
MD5_Init	0028ffa0	undefined MD5_Init()	82
FUN_00290000	00290000	undefined FUN_00290000()	141
FUN_00290090	00290090	undefined FUN_00290090()	135
FUN_00290120	00290120	undefined FUN_00290120()	90
FUN_00290180	00290180	undefined FUN_00290180()	332
FUN_002902d0	002902d0	undefined FUN_002902d0()	1093
FUN_00290720	00290720	undefined FUN_00290720()	184
FUN_002907e0	002907e0	undefined FUN_002907e0()	119
FUN_00290860	00290860	undefined FUN_00290860()	53
FUN_002908a0	002908a0	undefined FUN_002908a0()	93
FUN_00290900	00290900	undefined FUN_00290900()	168
FUN_002909b0	002909b0	undefined FUN_002909b0()	75

Decompile: MD5_Init - (md5_static.elf)

```

undefined8 MD5_Init(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
  param_1[3] = 0x10325476;
```

```
$ readelf -SW ./md5_dgst.o
Section Headers:
  [Nr] Name               Type      Off    Size   ES Flg Lk Inf Al
  [ 3] .text.MD5_Update   PROGBITS  000040 00020d 00  AX  0   0 16
  [ 5] .text.MD5_Transform PROGBITS 000250 000028 00  AX  0   0 16
  [ 7] .text.MD5_Final    PROGBITS  000280 000417 00  AX  0   0 16
  [ 9] .text.MD5_Init     PROGBITS  0006a0 000052 00  AX  0   0 16
```

Relative Positioning



Functions - 8177 items

Name	Location	Function Signature	Function Size
FUN_0028c9d0	0028c9d0	undefined FUN_0028c9d0()	196
FUN_0028caa0	0028caa0	undefined FUN_0028caa0()	369
FUN_0028cc20	0028cc20	undefined FUN_0028cc20()	497
FUN_0028ce20	0028ce20	undefined FUN_0028ce20()	354
FUN_0028cf90	0028cf90	undefined FUN_0028cf90()	289
FUN_0028d0c0	0028d0c0	undefined FUN_0028d0c0()	406
FUN_0028d260	0028d260	undefined FUN_0028d260()	174
FUN_0028d310	0028		
FUN_0028d340	0028		
FUN_0028d400	0028		
FUN_0028d440	0028		
FUN_0028d550	0028		
FUN_0028d590	0028		
FUN_0028d5b0	0028		
FUN_0028d5d0	0028		
FUN_0028d5f0	0028		
FUN_0028d800	0028		
FUN_0028eb10	0028		
FUN_0028eb40	0028		
FUN_0028ef60	0028ef60	undefined FUN_0028ef60()	82
FUN_0028efc0	0028efc0	undefined FUN_0028efc0()	141
FUN_0028f050	0028f050	undefined FUN_0028f050()	2283
FUN_0028f940	0028f940	undefined FUN_0028f940()	525
FUN_0028fb50	0028fb50	undefined FUN_0028fb50()	40
FUN_0028fb80	0028fb80	undefined FUN_0028fb80()	1047
MD5_Init	0028ffa0	undefined MD5_Init()	82
FUN_00290000	00290000	undefined FUN_00290000()	141
FUN_00290090	00290090	undefined FUN_00290090()	135
FUN_00290120	00290120	undefined FUN_00290120()	90
FUN_00290180	00290180	undefined FUN_00290180()	332
FUN_002902d0	002902d0	undefined FUN_002902d0()	1093
FUN_00290720	00290720	undefined FUN_00290720()	184
FUN_002907e0	002907e0	undefined FUN_002907e0()	119
FUN_00290860	00290860	undefined FUN_00290860()	53
FUN_002908a0	002908a0	undefined FUN_002908a0()	93
FUN_00290900	00290900	undefined FUN_00290900()	168
FUN_002909b0	002909b0	undefined FUN_002909b0()	75

Decompile: MD5_Init - (md5_static.elf)

```
undefined8 MD5_Init(undefined4 *param_1)

{
  memset(param_1,0,0x5c);
  *param_1 = 0x67452301;
  param_1[1] = 0xefcdab89;
  param_1[2] = 0x98badcfe;
```

```
$ readelf -SW ./md5_dgst.o
Section Headers:
  [Nr] Name               Type       Off    Size   ES Flg Lk Inf Al
  [ 3] .text.MD5_Update    PROGBITS  000040 00020d 00  AX  0   0 16
  [ 5] .text.MD5_Transform PROGBITS  000250 000028 00  AX  0   0 16
  [ 7] .text.MD5_Final     PROGBITS  000280 000417 00  AX  0   0 16
  [ 9] .text.MD5_Init      PROGBITS  0006a0 000052 00  AX  0   0 16
```

MD5_Update
MD5_Transform
MD5_Final

Relative Positioning

As a rule of thumb:
Functions that are close each other in the source file (or compilation unit), are likely close in the compiled binary.

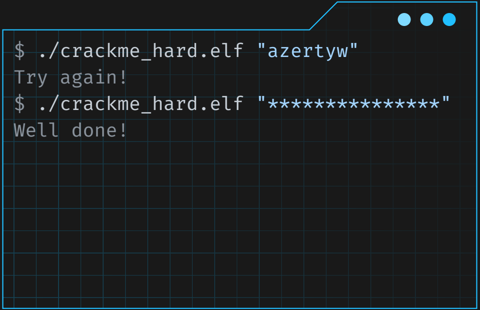
In particular, the relative order in the source code is preserved in the final binary[1].

[1] This can be mitigated: *https://github.com/open-obfuscator/o-mvll/blob/main/src/core/utils.cpp#L231-L257*

3

Hands-on #3: Embedded Library

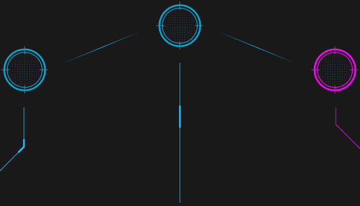
3



```
$ ./crackme_hard.elf "azertyw"
Try again!
$ ./crackme_hard.elf "***************"
Well done!
```

Level: Hard

Objectives: Reverse engineering a large binary with cryptography functions.



ELF x86-64

Involves known cryptography algorithms

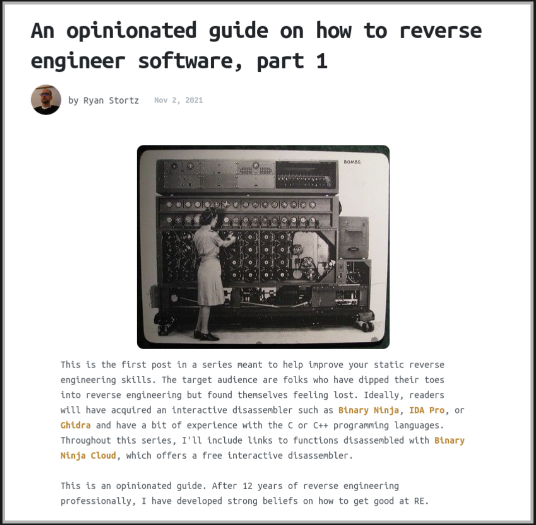
Stripped

Guidelines

- Statically linked against an open-source cryptography library
- The API for cryptographic functions follows this sequence:
    1. `init()`
    2. `update()`
    3. `finalize()`

Closing Remarks

Closing Remarks



An opinionated guide on how to reverse engineer software, part 1

by Ryan Stortz Nov 2, 2021

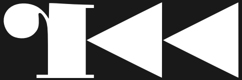
This is the first post in a series meant to help improve your static reverse engineering skills. The target audience are folks who have dipped their toes into reverse engineering but found themselves feeling lost. Ideally, readers will have acquired an interactive disassembler such as Binary Ninja, IDA Pro, or Ghidra and have a bit of experience with the C or C++ programming languages. Throughout this series, I'll include links to functions disassembled with Binary Ninja Cloud, which offers a free interactive disassembler.

This is an opinionated guide. After 12 years of reverse engineering professionally, I have developed strong beliefs on how to get good at RE.

A highly recommended reading:

1. https://margin.re/2021/11/an-opinionated-guide-on-how-to-reverse-engineer-software-part-1/
2. https://margin.re/2021/11/an-opinionated-guide-on-how-to-reverse-engineer-software-part-2/

Closing Remarks



Closing Remarks

🔗 https://www.root-me.org

Thank You!