MEMBONGKAR KODE PROGRAM DENGAN REVERSE ENGINEERING

Yohanes Nugroho

PERKENALAN

Yohanes Nugroho

https://tinyhack.com | https://cintaprogramming.com | https://yohan.es

Programming, reverse engineering, hobi elektronik

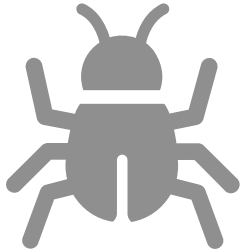
Materi hari ini: reverse engineering untuk pemula

REVERSE ENGINEERING

- Memahami apapun dengan “membongkar” hal tersebut:
  - Reverse engineering resep masakan
  - Reverse engineering design mesin
  - Reverse engineering layout PCB
- Presentasi ini hanya membahas membongkar kode

KODE PROGRAM

Software desktop, mobile



Malware: virus, trojan, ransomware

IOT

UNTUK APA?

- Mencari bug security (segala macam bug level kernel perlu pemahaman reverse engineering)
- Memahami algoritma tertentu
  - Memahami protokol tertentu
- Modifikasi program/game (cheat)
  - Ekstraksi asset game
  - Atau sekedar mencari tahu fitur berikutnya

How I reversed a NodeJS malware and found the author

The Devops Guy Follow

Jan 30 · 4 min read ★

To give a bit of context, I am a Discord admin on a small server about development, and we recently got a report from one of our users that someone was trying to get him to download an EXE file.

CIA DAN REVERSING SPACECRAFT

HOW THE CIA HIJACKED A SOVIET SPACECRAFT IN 1959

Alex Hollings | **January 27, 2022**

With the Cold War raging and the Soviets securing victory after victory in the Space Race, America's CIA wasn't sitting on the sidelines. The Soviet Union's space technology was beating America's in just about every appreciable way, and America's intelligence agencies were working overtime to monitor and decipher data spilling out of Soviet rockets as they poured into the sky. It was a time of uncertainty–and perhaps even a bit of desperation–for the burgeoning superpower that was America in the 1950s. So, when a Soviet Lunar satellite was sent out on a global tour to parade their successes before the world, it offered a unique opportunity for the CIA to *hijack* the satellite for a bit of research while it was still firmly planted on the ground.

https://www.sandboxx.us/blog/the-cia-hijacked-a-soviet-spacecraft-in-1959/

PEKERJAAN RE

- Saat ini sulit mencari pekerjaan yang hanya berfokus pada RE di Indonesia
- Sudah banyak dicari di luar negeri (paling dekat: Singapura)
- Reversing malware
- Mencari bug di software/hardware IOT

RE DAN HAL ILLEGAL

- Banyak hasil Reverse Engineering yang ditemui sehari-hari yang sifatnya illegal
  - Konten WebRIP hasil dari reverse engineering Widevine DRM
  - Game dan software bajaka
  - Game mod dan cheat

DASAR TEORI

- Kita perlu memiliki dasar teori dalam hal apapun yang kita reverse
- Jika diberi roket untuk direverse engineer, bisakah kita:
  - Membongkarnya? Butuh tool apa?
  - Memahami apa yang dilihat? (ini komponen apa?)
  - Mengukur komponen-komponennya (dimensinya, berat, dan aneka informasi lain untuk benda elektroniknya)
  - Memahami fungsinya

TOOL

- Tool bergantung pada: apa yang ingin direverse engineer
  - Aplikasi mobile: apktool, jadx
  - Aplikasi desktop: IDA, Binary Ninja, Ghidra
  - Device IOT: IDA, binary ninja, ghidra
- Bergantung juga pada proteksi yang ada
  - Memory dumper
  - Manual unpack
  - Koding sendiri unpacker

BLACK BOX RE

- Mudah dilakukan: sekedar mengamati program ketika dijalankan
- Mengubah input dan melihat output
  - Kadang sudah cukup untuk reverse engineering format file
- Kelemahan: Sulit menemukan hal tersembunyi, misalnya jika program menghapus harddisk para Jumat Kliwon tanggal 13, tidak akan terdeteksi dengan hanya mencoba-coba

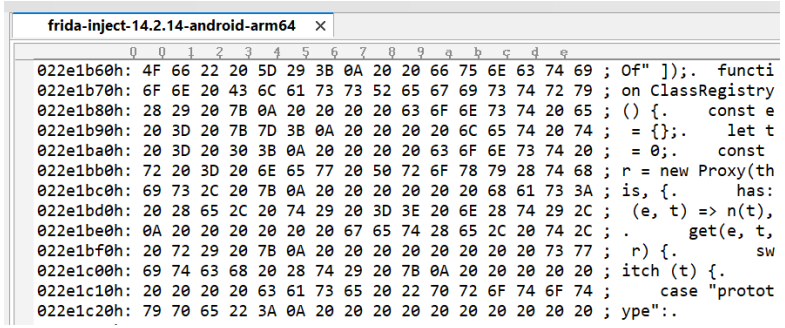
WHITE BOX RE

- Memahami program dengan membaca kodenya
- Kadang source code tersedia dan bisa dibaca (contoh: kode Python, PHP)
- Sering kali source code sudah diterjemahkan menjadi bahasa mesin

SOURCE CODE

- Mendapatkan source code hanya satu langkah saja, **memahami kode** adalah langkah yang lebih sulit
- Sudah banyak software di github, tapi apakah bisa dipahami?
- Kernel Linux juga sudah terbuka source codenya bahkan ada banyak buku yang menjelaskan tiap komponennya, apakah bisa mudah dimengerti kebanyakan programmer?

HEX EDITOR



Untuk Aplikasi yang sangat sederhana, sudah cukup untuk membongkar program

Kita melihat representasi heksadesimal dan string-string yang terbaca

Apa yang dicari: string yang tidak biasa (kemungkinan password), URL, email

DISASSEMBLER

Membuat kode biner menjadi kode assembly

Kode sederhana dan kecil bisa dibaca, tapi kode kompleks akan sulit dibaca

```
.text:00000001400029B0
.text:00000001400029B0 ; _QWORD *__fastcall sub_1400029B0(_QWORD *, unsigned __int64)
.text:00000001400029B0 sub_1400029B0   proc near               ; CODE XREF: AfxMergeMenus(HMENU__ *,HMENU__ *,long *,int,int)+146↓p
.text:00000001400029B0                                         ; _AfxCopyStgMedium(ushort,tagSTGMEDIUM *,tagSTGMEDIUM *)+2D1↓p ...
.text:00000001400029B0
.text:00000001400029B0 arg_0           = qword ptr  8
.text:00000001400029B0 arg_8           = qword ptr  10h
.text:00000001400029B0 arg_10          = qword ptr  18h
.text:00000001400029B0
.text:00000001400029B0                 mov     [rsp+arg_8], rbx
.text:00000001400029B5                 mov     [rsp+arg_10], rsi
.text:00000001400029BA                 mov     [rsp+arg_0], rcx
.text:00000001400029BF                 push    rdi
.text:00000001400029C0                 sub     rsp, 20h
.text:00000001400029C4                 mov     rdi, rdx
.text:00000001400029C7                 mov     rbx, rcx
.text:00000001400029CA                 call    sub_140006894
.text:00000001400029CF                 mov     rcx, rax
.text:00000001400029D2                 xor     esi, esi
.text:00000001400029D4                 test    rax, rax
.text:00000001400029D7                 jz      short loc_140002A46
.text:00000001400029D9                 mov     rax, [rax]
.text:00000001400029DC                 call    qword ptr [rax+18h]
.text:00000001400029DF                 add     rax, 18h
.text:00000001400029E3                 mov     [rbx], rax
.text:00000001400029E6                 test    rdi, rdi
.text:00000001400029E9                 jz      short loc_140002A22
.text:00000001400029EB                 cmp     rdi, 10000h
.text:00000001400029F2                 jnb     short loc_140002A13
```

ANTI-DISASSEMBLER

Di arsitektur tertentu (misalnya intel), kita bisa membuat program yang membingungkan disassembler dengan melompat ke tengah instruksi

Kita bisa mencampur kode dan data

DECOMPILER

Mengembalikan kode sebisa mungkin ke high-level language

Perlu campur tangan manual agar benar-benar terbaca

```c
static void bubble_sort(int *numbers, int count)
{
        printf("Bubble sort method\n");
        int i,j;
        for (i = 0; i < count; i++) {
                for (j=i+1; j < count; j++) {
                        if (numbers[i] > numbers[j]) {
                                int tmp = numbers[i];
                                numbers[i] = numbers[j];
                                numbers[j] = tmp;
                        }
                }
        }
}
```

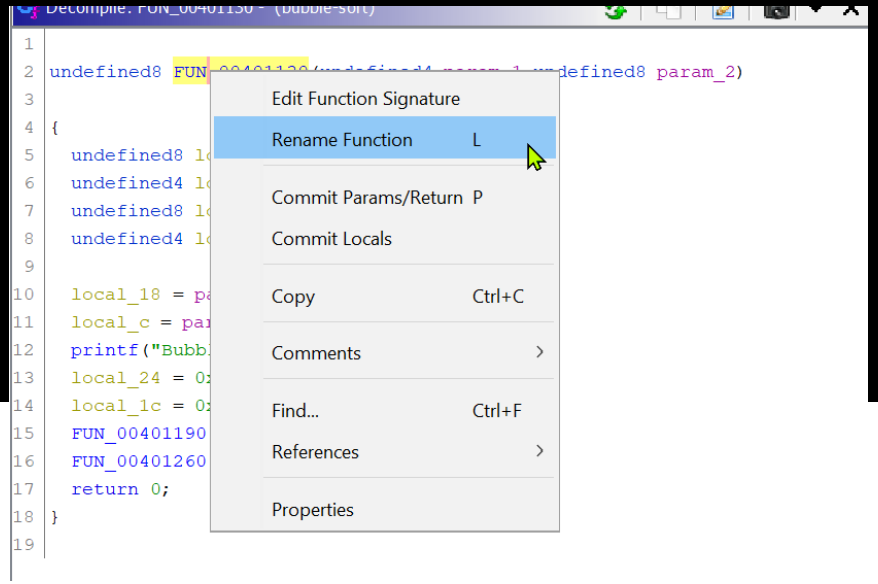
KODE ASLI

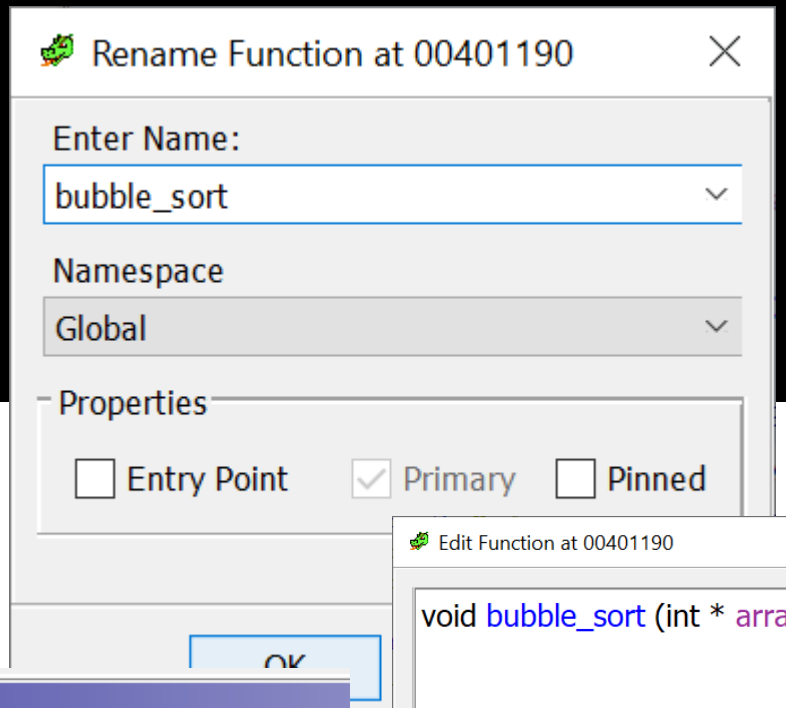
Decompile: FUN_00401190 - (bubble-sort)

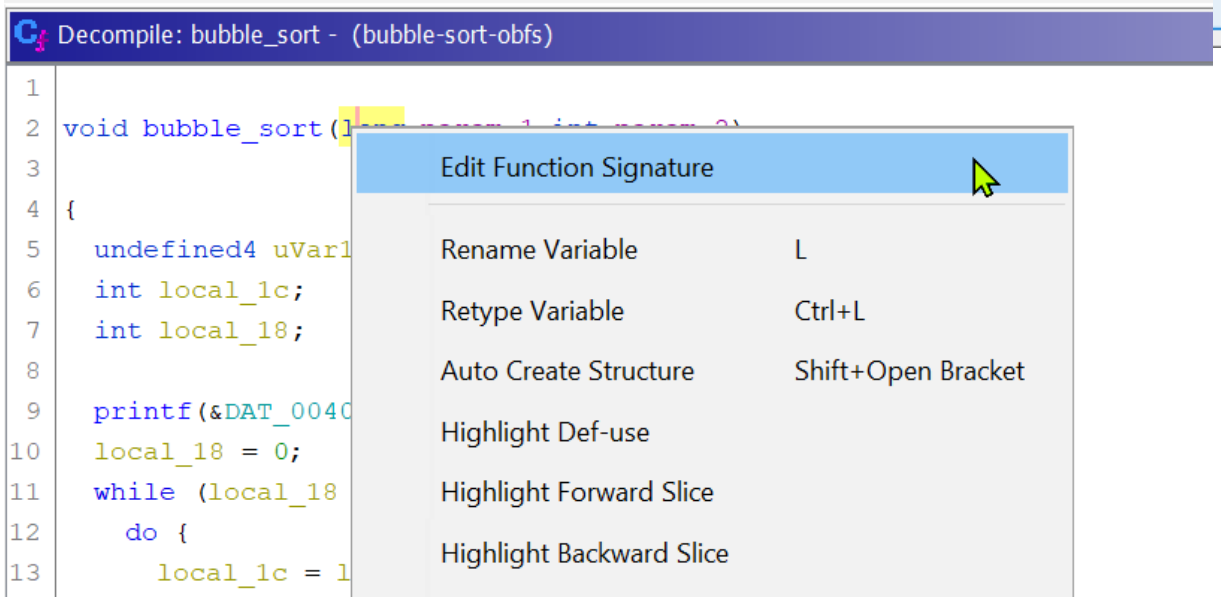
```
1
2  void FUN_00401190(long param_1,int param_2)
3
4  {
5    undefined4 uVar1;
6    int local_1c;
7    int local_18;
8
9    printf("Bubble sort method\n");
10   local_18 = 0;
11   while (local_1c = local_18, local_18 < param_2) {
12     while (local_1c = local_1c + 1, local_1c < param_2) {
13       if (*(int *)(param_1 + (long)local_1c * 4) < *(int *)(param_1 + (long)local_18 * 4)) {
14         uVar1 = *(undefined4 *)(param_1 + (long)local_18 * 4);
15         *(undefined4 *)(param_1 + (long)local_18 * 4) =
16              *(undefined4 *)(param_1 + (long)local_1c * 4);
17         *(undefined4 *)(param_1 + (long)local_1c * 4) = uVar1;
18       }
19     }
20     local_18 = local_18 + 1;
21   }
22   return;
23 }
24
```

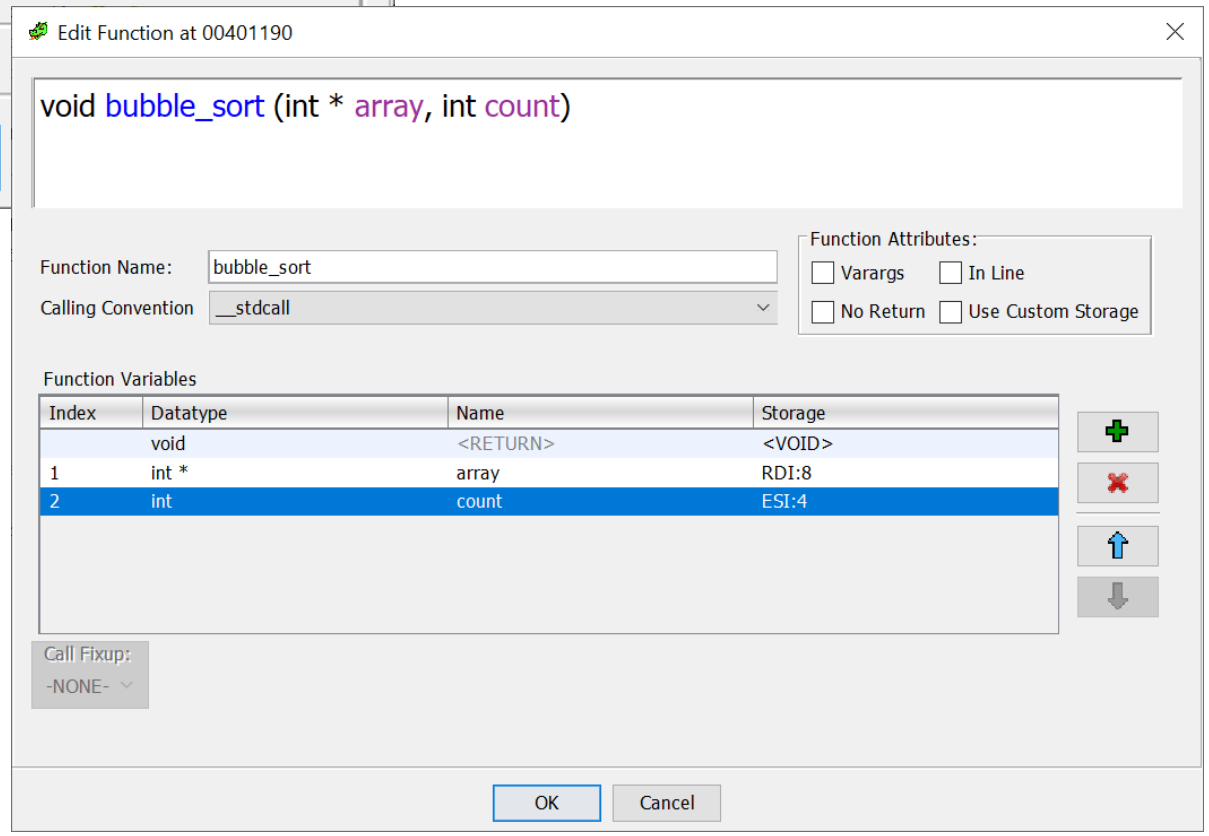
Hasil Dekompilasi

Manual Tuning









Decompile: bubble_sort - (bubble-sort)

```
1
2  void bubble_sort(int *array,int count)
3
4  {
5    int tmp;
6    int i;
7    int j;
8
9    printf("Bubble sort method\n");
10   j = 0;
11   while (i = j, j < count) {
12     while (i = i + 1, i < count) {
13       if (array[i] < array[j]) {
14         tmp = array[j];
15         array[j] = array[i];
16         array[i] = tmp;
17       }
18     }
19     j = j + 1;
20   }
21   return;
22 }
23
```

OBFUSCATION

- Meskipun kita mendapatkan source code, kode tersebut bisa obfuscated (tersamarkan)
- Tidak ada komentar, tidak ada dokumentasi
- Nama fungsi dan variabel sengaja diubah
- Kode bisa dibuat sangat membingungkan, sehingga meskipun sudah dalam bahasa tingkat tinggi, tetap tidak bisa dimengerti

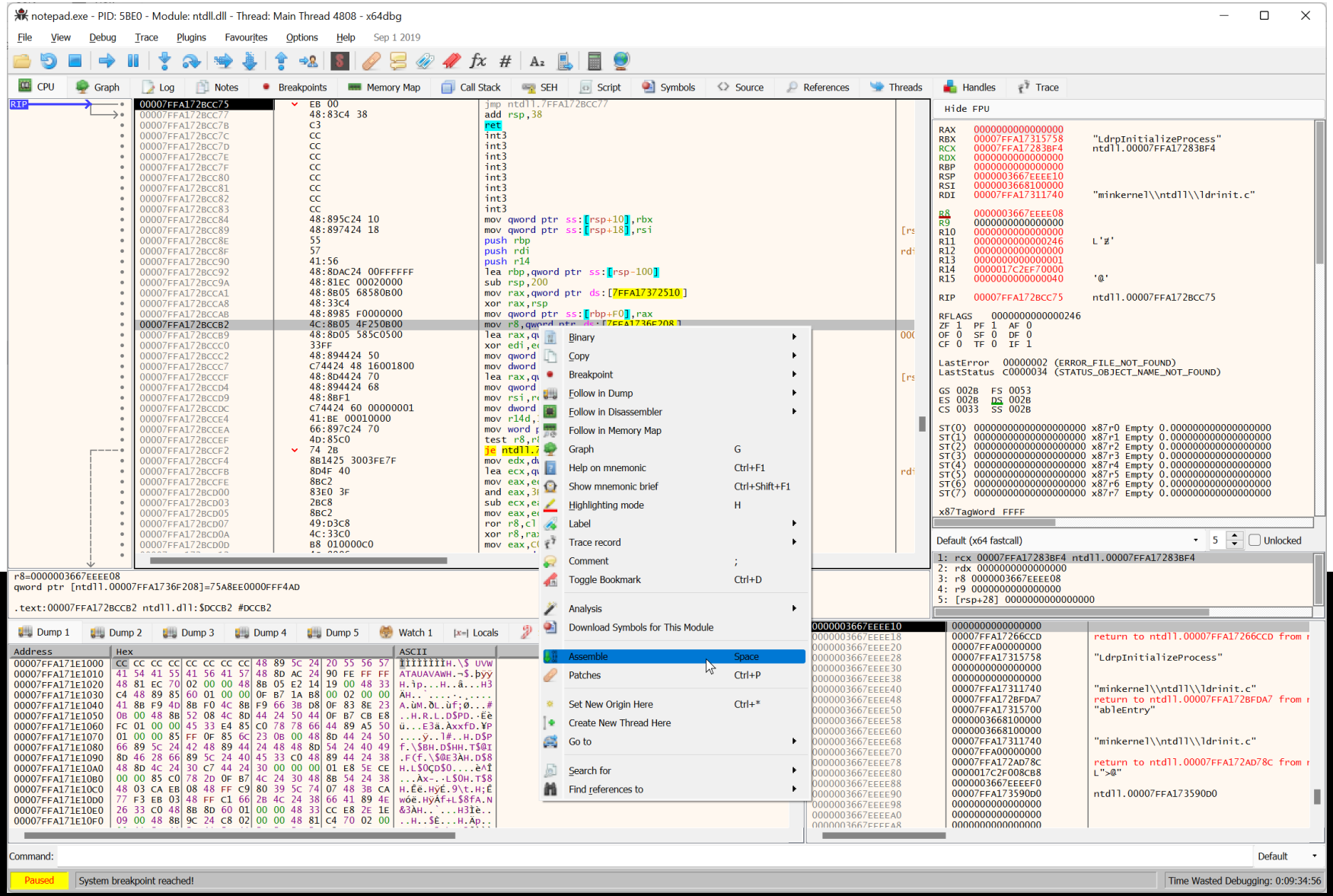
```
do {
  while( true ) {
    while( true ) {
      while( true ) {
        while( true ) {
          while( true ) {
            while( true ) {
              while( true ) {
                while( true ) {
                  while( true ) {
                    while( true ) {
                      while( true ) {
                        while( true ) {
                          while( true ) {
                            while( true ) {
                              while( true ) {
                                while( true ) {
                                  while( true ) {
                                    while (local_2c == -0x4d98a826) {
                                      local_24 = local_20 + 1;
                                      local_2c = 0x629e5c5d;
                                      if ((DAT_0040407c * (DAT_0040407c + -1) & 1U) == 0 ||
                                          DAT_00404088 < 10) {
                                        local_2c = 0xd4536596;
                                      }
                                    }
                                    if (local_2c != -0x4589fc89) break;
                                    local_2c = 0x5b785d81;
                                    if ((local_9 & 1U) != 0) {
                                      local_2c = 0xdcc1b8f9;
                                    }
                                  }
                                  if (local_2c != -0x42710278) break;
                                  local_2c = 0x7aaab1cb;
                                  if (local_20 < param_2) {
```

Obfuscated

DEBUGGER

Melihat bagaimana program berjalan

Bisa melihat memori dan variable/register, bisa melompat ke alamat tertentu


notepad.exe - PID: 5BE0 - Module: ntdll.dll - Thread: Main Thread 4808 - x64dbg
File View Debug Trace Plugins Favourites Options Help Sep 1 2019
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
Hide FPU
RAX 0000000000000000
RBX 00007FFA17315758 "LdrpInitializeProcess"
RCX 00007FFA17283BF4 ntdll.00007FFA17283BF4
RDX 0000000000000000
RBP 0000000000000000
RSP 000000367EEEE10
RSI 0000003668100000
RDI 00007FFA17311740 "minkernel\\ntdll\\ldrinit.c"
R8 000000367EEEE08
R9 0000000000000000
R10 0000000000000000
R11 0000000000000246 L'ǵ'
R12 0000000000000000
R13 0000000000000001
R14 0000017C2EF70000
R15 0000000000000040 '@'
RIP 00007FFA172BCC75 ntdll.00007FFA172BCC75
RFLAGS 0000000000000246
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1
LastError 00000002 (ERROR_FILE_NOT_FOUND)
LastStatus C0000034 (STATUS_OBJECT_NAME_NOT_FOUND)
GS 002B FS 0053
ES 002B DS 002B
CS 0033 SS 002B
ST(0) 00000000000000000000 x87r0 Empty 0.00000000000000000000
ST(1) 00000000000000000000 x87r1 Empty 0.00000000000000000000
ST(2) 00000000000000000000 x87r2 Empty 0.00000000000000000000
ST(3) 00000000000000000000 x87r3 Empty 0.00000000000000000000
ST(4) 00000000000000000000 x87r4 Empty 0.00000000000000000000
ST(5) 00000000000000000000 x87r5 Empty 0.00000000000000000000
ST(6) 00000000000000000000 x87r6 Empty 0.00000000000000000000
ST(7) 00000000000000000000 x87r7 Empty 0.00000000000000000000
x87TagWord FFFF
Default (x64 fastcall)
1: rcx 00007FFA17283BF4 ntdll.00007FFA17283BF4
2: rdx 0000000000000000
3: r8 000000367EEEE08
4: r9 0000000000000000
5: [rsp+28] 0000000000000000
r8=000000367EEEE08
qword ptr [ntdll.00007FFA1736F208]=75A8EE0000FFF4AD
.text:00007FFA172BCCB2 ntdll.dll:$DCCB2 #DCCB2
Binary
Copy
Breakpoint
Follow in Dump
Follow in Disassembler
Follow in Memory Map
Graph G
Help on mnemonic Ctrl+F1
Show mnemonic brief Ctrl+Shift+F1
Highlighting mode H
Label
Trace record
Comment ;
Toggle Bookmark Ctrl+D
Analysis
Download Symbols for This Module
Assemble Space
Patches Ctrl+P
Set New Origin Here Ctrl+*
Create New Thread Here
Go to
Search for
Find references to
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals
Address Hex ASCII
Command:
Default
Paused
System breakpoint reached!
Time Wasted Debugging: 0:09:34:56

ANTI DEBUG

Sebuah program bisa mendeteksi apakah sedang dijalankan dalam debugger

Jika mendeteksi debugger, bisa:

- exit
- crash
- melakukan hal lain
- random

BINARY INSTRUMENTATION

Selain debugger, ada juga tool lain yang bisa dipakai untuk menelusuri program ketika berjalan

Intel PIN

Frida

https://blog.compactbyte.com/2019/08/11/mengenal-frida-untuk-reverse-engineering/

ENKRIPSI DAN KOMPRESI

Jika string dienkripsi, maka tidak akan muncul seperti apa adanya. Kode juga bisa dienkrip dan didekrip pada runtime

Kompresi juga bisa digunakan untuk menyembunyikan string. Kode juga bisa dikompres, misalnya dengan UPX.

TEKNIK ANTI-RE TINGKAT LANJUT

Virtual Machine

Membuat “mesin” virtual sendiri, dengan instruksi sendiri

KISAH RE

Contoh Aplikasi: pembaca buku/majalah Indonesia

Banyak orang tidak mau membeli

Banyak yang ingin membajak/mengcopy PDF majalahnya

Banyak juga yang sharing account

LEVEL 0

Dulu ketika dilaunch, aplikasi memakai IAP tapi caranya tidak aman. Di device yang dijailbreak, bisa mendownload semua majalah gratis

Tidak butuh RE

LEVEL 1

Pengecekan IAP di server

PDF didownload aplikasi dengan password statik, sama untuk semua majalah/ebook

Hex Editor cukup untuk membukanya

LEVEL 2

Password berbeda untuk tiap majalah/buku

Password disertakan dalam request REST/JSON untuk konten

Cukup memakai intercepting proxy

LEVEL 3

Dalam JSON disertakan password terenkripsi, perlu didekrip oleh aplikasi

Sekedar intercept tidak bisa

Harus melakukan RE untuk mencari algoritmanya

LEVEL 4

Kode password diobfuscate

Sulit mencari algoritmanya

Bisa diintercept dengan Frida dan atau Tweak/Module XPosed

MAU MEMULAI?

- Belajar dasar programming
- Bahasa/teknologi apa yang perlu dipelajari?
  - Tergantung pada apa yang mau direverse engineer
  - Sebagai reverse engineer, tidak bisa memilih (misalnya: tidak bisa meminta malware developer untuk tidak memakai Go)

TOOL MUDAH DIPELAJARI

- Sama seperti mengajari mengetik dengan Word bisa dilakukan dalam hitungan jam
- Tapi mengajari orang jadi penulis buku/novel butuh waktu bertahun-tahun

BUKU KHUSUS RE

Kebanyakan buku cepat ketinggalan jaman

- Practical Reverse Engineering: X86, X64, ARM, Windows Kernel, Reversing Tools, and Obfuscation
- Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software
- The Ghidra Book: The Definitive Guide / The IDA Pro Book, 2nd Edition

ONLINE COURSE

https://www.udemy.com/topic/reverse-engineering/

https://www.sans.org/cyber-security-courses/reverse-engineering-malware-malware-analysis-tools-techniques/

BERLATIH

Kunci utama

CTF

CTF merupakan ajang latihan yang bagus

Beberapa soal CTF sangat sederhana, dan beberapa sangat sulit

Tiap tahun ada Flare-On, waktunya cukup lama

TIPS UNTUK PEMULA

- Pelajari teknologi yang ingin dibongkar
  - Belajar memprogram dulu
- Sampai level tertentu, banyak yang bisa dibongkar dengan mudah
  - Tapi jika ingin naik level, banyak yang perlu dipelajari
- Bertanya yang jelas:
  - Bagimana cara mendapatkan key enkripsi aplikasi ini?
  - Bagaimana caranya mendapatkan kunci rumah orang itu?