Muhammad Izham Bin Norhamadi
B032020039

LAB TEST 2 MALWARE ANALYSIS AND DIGITAL INVESTIGATION

1) The type of file of the sample

- Executable and Linking Format (ELF)

2) The tool use to analyze the sample

- Ghidra
- edb debugger
- IDA pro

3)The behavior of the sample do

1. Immediately after executing, the sample tries to resolve the connection to ftmk.utem.edu.my to get system.file

```
┌──(kali㉿kali)-[~/Desktop/Lab Test 2]
└─$ ./LabTestSem12122
--2022-01-19 01:37:03--  https://ftmk.utem.edu.my/dfir4/system.file
Resolving ftmk.utem.edu.my (ftmk.utem.edu.my)... 
```

2. Then, the sample clears the terminal and the message “Your machine has been infected ;P” was shown

```
kali@kali: ~/Desktop/Lab Test 2
File  Actions  Edit  View  Help
Your machine has been infected ;P
```

4) The traces of the activity the sample do

- The sample will attempt to resolve connection to ftmk.utem.edu.my, when the machine does not have connection it will still print “Your machine has been infected ;P” after some time.

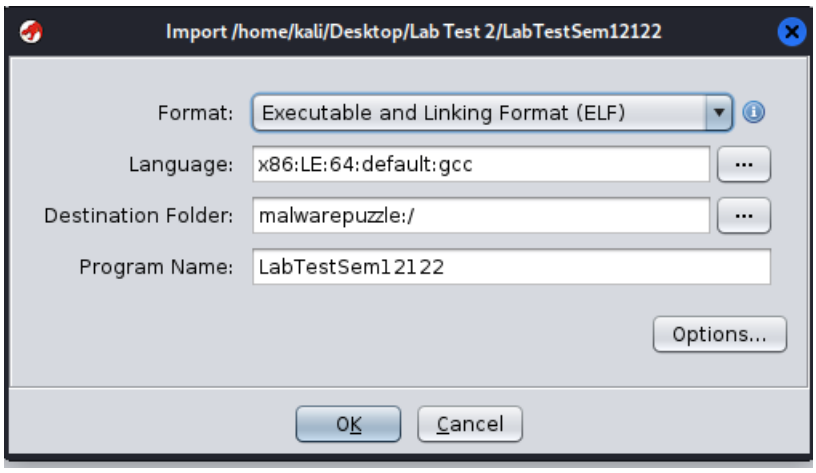
-The sample has filestream calls which can output characters into a file

5) The Two Flag you can get from the sample file

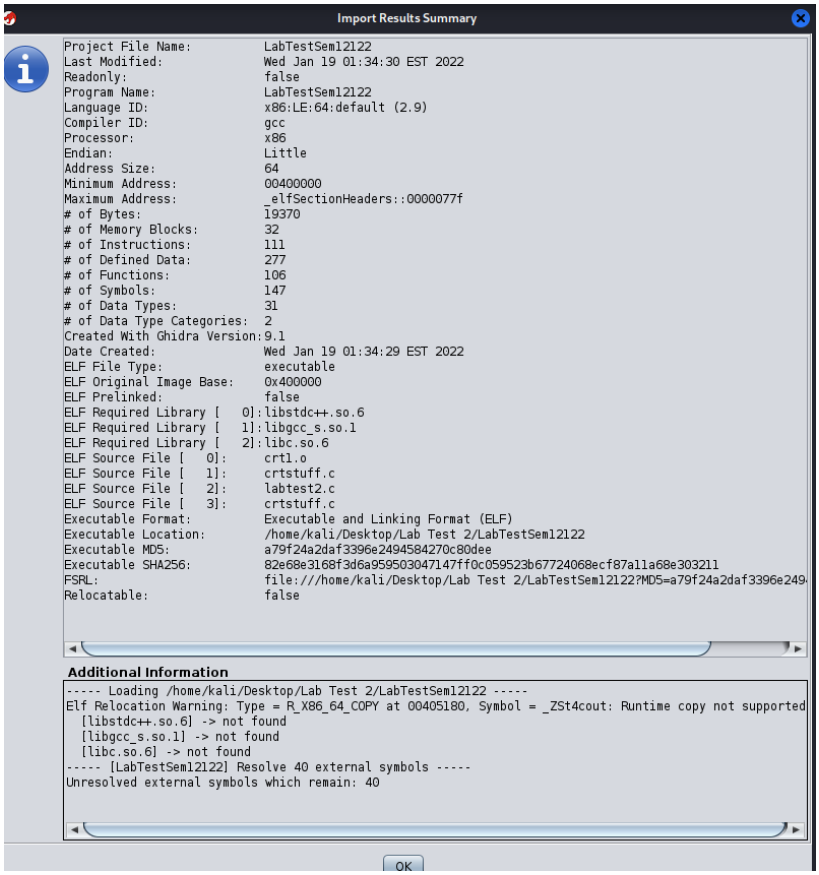
BITS3453<labtest2_flag_one>

Static Analysis

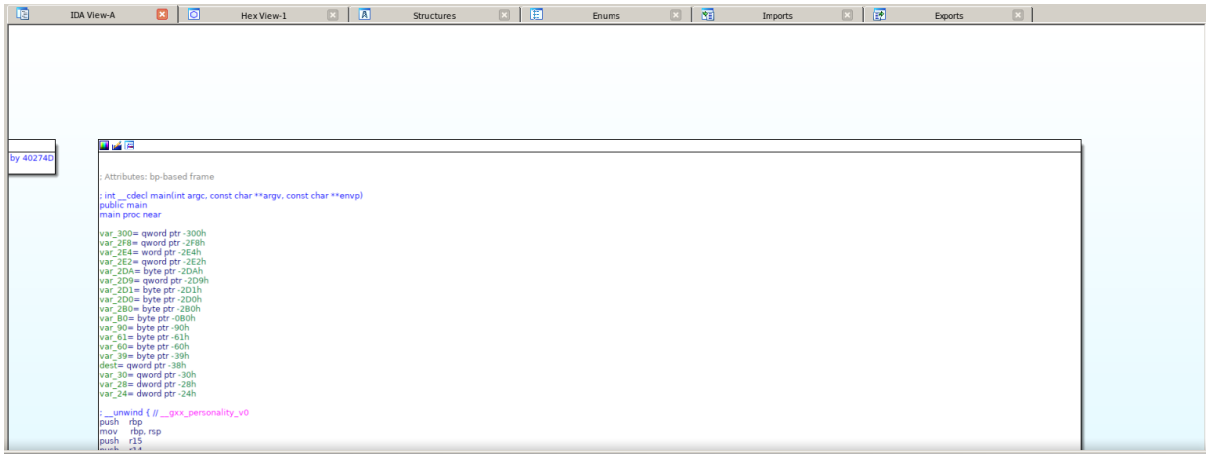
1. Import file to Ghidra and IDA pro



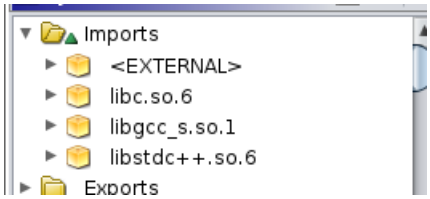
2. Check import result summary for details of the ELF



3. IDA pro has IDA View to view the structure of the ELF



4. The imports has libgcc which means the ELF was written in C++ language



5. Go to the main function to analyse the process of the ELF with the decompiled to C language window on Ghidra

Decompile: main - (LabTestSem12122)

```
1
2  undefined8 main(void)
3
4  {
5    undefined *__dest;
6    byte bVar1;
7    int iVar2;
8    undefined4 uVar3;
9    ulong uVar4;
10   undefined8 __src;
11   long alStack784 [4];
12   ushort local_2ec;
13   undefined8 local_2ea;
14   undefined local_2e2;
15   undefined8 local_2e1;
16   undefined local_2d9;
17   undefined local_2d8 [32];
18   undefined local_2b8 [512];
19   undefined local_b8 [32];
20   basic_string<char,std--char_traits<char>,std--allocator<char>> local_98 [47];
21   allocator<char> local_69;
22   undefined local_68 [39];
23   undefined local_41;
24   undefined *local_40;
25   long local_38;
26   int local_30;
27   int local_2c;
28
29   alStack784[0] = 0x402735;
30   allocator();
31                    /* try { // try from 0040274d to 00402751 has its CatchHandler @ 004029c0 */
32   alStack784[0] = 0x402752;
33   basic_string<std--allocator<char>>
```

6. Then, the main function calls the Decode function with integer parameter

Decompile: Decode - (LabTestSem12122)

```
/* Decode(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>) */

basic_string<char,std--char_traits<char>,std--allocator<char>> * Decode(basic_string param_1)

{
  char cVar1;
  char cVar2;
  char cVar3;
  int iVar4;
  undefined uVar5;
  int iVar6;
  int iVar7;
  char *pcVar8;
  void *pvVar9;
  void *pvVar10;
  basic_string<char,std--char_traits<char>,std--allocator<char>> *in_RSI;
  undefined4 in_register_0000003c;
  int local_2c;
  int local_28;
  int local_24;
  int local_20;
  int local_1c;

  local_1c = 0;
  iVar6 = length();
  iVar7 = iVar6;
  if (iVar6 < 0) {
    iVar7 = iVar6 + 3;
  }
  iVar4 = (iVar7 >> 2) * 3;
  local_20 = 0;
  while (local_20 < 2) {
```

7. Then the function calls CharToSixBit function in loops

```
  local_24 = 0;
  while (local_24 < iVar6) {
    pcVar8 = (char *)operator[](in_RSI,(long)local_24);
    uVar5 = CharToSixBit(*pcVar8);
    *(undefined *)((long)local_24 + (long)pvVar9) = uVar5;
    local_24 = local_24 + 1;
  }
```

8. CharToSixBit function encodes word lengths on multiple of 6

Decompile: CharToSixBit - (LabTestSem12122)

```
/* CharToSixBit(char) */

ulong CharToSixBit(char param_1)

{
  undefined8 local_58;
  undefined8 local_50;
  undefined8 local_48;
  undefined8 local_40;
  undefined8 local_38;
  undefined8 local_30;
  undefined8 local_28;
  undefined8 local_20;
  uint local_c;

  local_58 = 0x4847464544434241;
  local_50 = 0x504f4e4d4c4b4a49;
  local_48 = 0x5857565554535251;
  local_40 = 0x6665646362615a59;
  local_38 = 0x6e6d6c6b6a696867;
  local_30 = 0x767574737271706f;
  local_28 = 0x333231307a797877;
  local_20 = 0x2f2b393837363534;
  if (param_1 != '=') {
    local_c = 0;
    while ((int)local_c < 0x40) {
      if (param_1 == *(char *)((long)&local_58 + (long)(int)local_c)) {
        return (ulong)local_c;
      }
      local_c = local_c + 1;
    }
  }
  return 0;
```

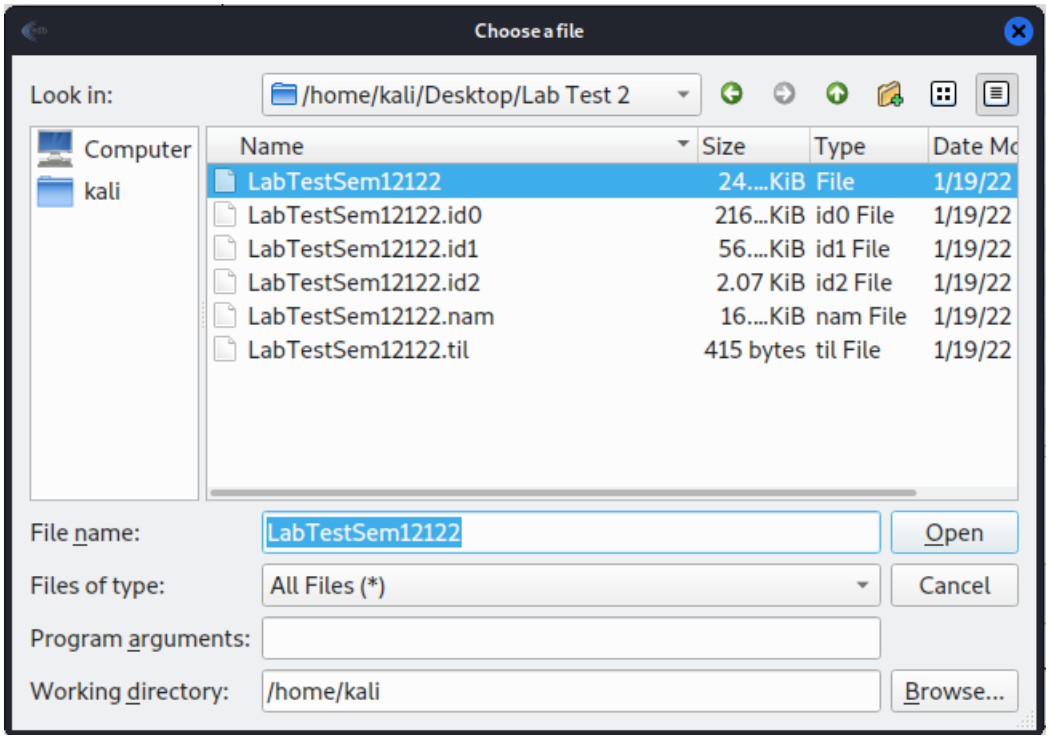
9. In main function, the executable loops rbp+var_24 (with value 0) with 8 and adding its value by one, calling outstream function then prints out “Your machine has been infected :P”, and the program ends.

```
loc_402927:
cmp     [rbp+var_24], 8
jle     short loc_4028E6
```

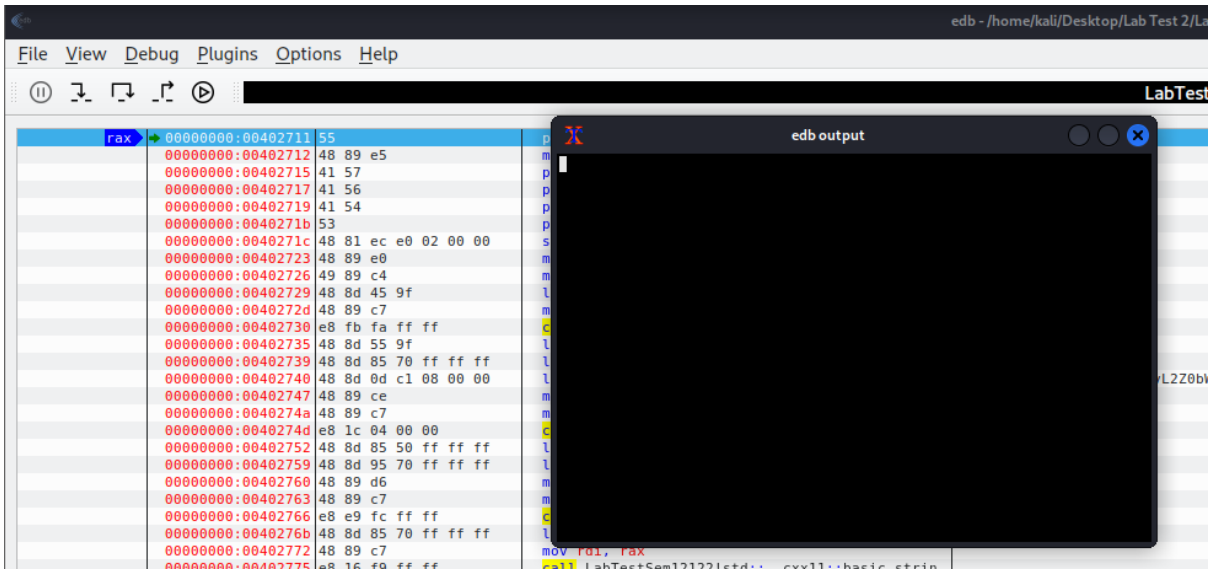
```
lea     rax, [rbp+var_2B0]
mov     rdi, rax
call    __ZNSt14basic_ofstreamIcSt11char_traitsIcEE5closeEv ; std::ofstream::close(void)
mov     rax, [rbp+dest]
mov     rdi, rax        ; command
call    _system
lea     rax, command    ; "clear"
mov     rdi, rax        ; command
call    _system
lea     rax, aYourMachineHas ; "Your machine has been infected :P\n"
mov     rsi, rax
lea     rax, _ZSt4cout@GLIBCXX_3_4
mov     rdi, rax
call    __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc ; std::operator<<<std::char_traits<char>>(std::ostream &,char const*)
;   } // starts at 40289F
mov     ebx, 0
lea     rax, [rbp+var_2D0]
mov     rdi, rax
call    __ZNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEED1Ev ; std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string()
```

Dynamic Analysis

1. Open the ELF file in edb debugger



2. Press play once and analyse any notable functions of the program



3. Toggle breakpoints on notable function such as Decoder

```
File  View  Debug  Plugins  Options  Help

00000000:0040281f 48 89 c7          mov rdi, rax
00000000:00402822 e8 e9 f9 ff ff    call LabTestSem12122!std::basic_ofstream<char,…
00000000:00402827 48 8d 45 c7       lea rax, [rbp-0x39]
00000000:0040282b 48 89 c7          mov rdi, rax
00000000:0040282e e8 fd f9 ff ff    call LabTestSem12122!std::allocator<char>::all…
00000000:00402833 48 8d 55 c7       lea rdx, [rbp-0x39]
00000000:00402837 48 8d 45 a0       lea rax, [rbp-0x60]
00000000:0040283b 48 8d 0d 23 08 00 00  lea rcx, [rel 0x403065]
00000000:00402842 48 89 ce          mov rsi, rcx
00000000:00402845 48 89 c7          mov rdi, rax
00000000:00402848 e8 21 03 00 00    call LabTestSem12122!std::__cxx11::basic_strin…
00000000:0040284d 48 8d 85 30 fd ff ff  lea rax, [rbp-0x2d0]
00000000:00402854 48 8d 55 a0       lea rdx, [rbp-0x60]
00000000:00402858 48 89 d6          mov rsi, rdx
00000000:0040285b 48 89 c7          mov rdi, rax
00000000:0040285e e8 f1 fb ff ff    call LabTestSem12122!Decode(std::__cxx11::basi…
00000000:00402863 48 8d 45 a0       lea rax, [rbp-0x60]
00000000:00402867 48 89 c7          mov rdi, rax
00000000:0040286a e8 21 f8 ff ff    call LabTestSem12122!std::__cxx11::basic_strin…
00000000:0040286f 48 8d 45 c7       lea rax, [rbp-0x39]
00000000:00402873 48 89 c7          mov rdi, rax
00000000:00402876 e8 c5 f8 ff ff    call LabTestSem12122!std::allocator<char>::~al…
00000000:0040287b 48 8d 85 30 fd ff ff  lea rax, [rbp-0x2d0]
00000000:00402882 48 89 c7          mov rdi, rax
00000000:00402885 e8 e6 f7 ff ff    call LabTestSem12122!std::__cxx11::basic_strin…
00000000:0040288a 48 89 c1          mov rcx, rax
```

4. Step into Decoder function

```
00000000:0040244d b8 00 00 00 00    mov eax, 0
00000000:00402452 5d                pop rbp
00000000:00402453 c3                ret
00000000:00402454 55                push rbp
00000000:00402455 48 89 e5          mov rbp, rsp
00000000:00402458 53                push rbx
00000000:00402459 48 83 ec 68       sub rsp, 0x68
00000000:0040245d 48 89 7d 98       mov [rbp-0x68], rdi
00000000:00402461 48 89 75 90       mov [rbp-0x70], rsi
00000000:00402465 c7 45 ec 00 00 00 00  mov dword [rbp-0x14], 0
00000000:0040246c 48 8b 45 90       mov rax, [rbp-0x70]
00000000:00402470 48 89 c7          mov rdi, rax
00000000:00402473 e8 68 fd ff ff    call LabTestSem12122!std::__cxx11::basic_strin…
00000000:00402478 89 45 d8          mov [rbp-0x28], eax
00000000:0040247b 8b 45 d8          mov eax, [rbp-0x28]
00000000:0040247e 89 45 d4          mov [rbp-0x2c], eax
00000000:00402481 8b 45 d4          mov eax, [rbp-0x2c]
00000000:00402484 8d 50 03          lea edx, [rax+3]
00000000:00402487 85 c0             test eax, eax
00000000:00402489 0f 48 c2          cmovs eax, edx
00000000:0040248c c1 f8 02          sar eax, 2
00000000:0040248f 89 45 d0          mov [rbp-0x30], eax
00000000:00402492 8b 55 d0          mov edx, [rbp-0x30]
00000000:00402495 89 d0             mov eax, edx
00000000:00402497 01 c0             add eax, eax
00000000:00402499 01 d0             add eax, edx
```

5. Toggle breakpoint on CharToSixBit function and step into function

```
File View Debug Plugins Options Help

00000000:0040251e 00 0f                    add [rdi], cl
00000000:00402520 be c0 8b 55 e4           mov esi, 0xe4558bc0
00000000:00402525 48 63 ca                 movsxd rcx, edx
00000000:00402528 48 8b 55 c0              mov rdx, [rbp-0x40]
00000000:0040252c 48 8d 1c 11              lea rbx, [rcx+rdx]
00000000:00402530 89 c7                    mov edi, eax
00000000:00402532 e8 69 fe ff ff           call LabTestSem12122!CharToSixBit(char)
00000000:00402537 88 03                    mov [rbx], al
00000000:00402539 83 45 e4 01              add dword [rbp-0x1c], 1
00000000:0040253d 8b 45 e4                 mov eax, [rbp-0x1c]
00000000:00402540 3b 45 d4                 cmp eax, [rbp-0x2c]
00000000:00402543 7c c2                    jl 0x402507
00000000:00402545 c7 45 e0 00 00 00 00     mov dword [rbp-0x20], 0
00000000:0040254c e9 1c 01 00 00           jmp 0x40266d
00000000:00402551 8b 45 e0                 mov eax, [rbp-0x20]
00000000:00402554 c1 e0 02                 shl eax, 2
00000000:00402557 48 63 d0                 movsxd rdx, eax
00000000:0040255a 48 8b 45 c0              mov rax, [rbp-0x40]
00000000:0040255e 48 01 d0                 add rax, rdx
00000000:00402561 0f b6 00                 movzx eax, byte [rax]
00000000:00402564 88 45 b3                 mov [rbp-0x4d], al
00000000:00402567 8b 45 e0                 mov eax, [rbp-0x20]
00000000:0040256a c1 e0 02                 shl eax, 2
00000000:0040256d 48 98                    cdqe
00000000:0040256f 48 8d 50 01              lea rdx, [rax+1]
00000000:00402573 48 8b 45 c0              mov rax, [rbp-0x40]
00000000:00402577 48 01 d0                 add rax, rdx
```

6. Back in main function, the program loops output stream until rbp-0x24 value is more than 8

```
00000000:004028e1 00 00                    add [rax], al
00000000:004028e3 00 eb                    add bl, ch
00000000:004028e5 41 8b 45 dc              mov eax, [r13-0x24]
00000000:004028e9 48 98                    cdqe
00000000:004028eb 0f b6 94 05 1e fd ff ff  movzx edx, byte [rbp+rax-0x2e2]
00000000:004028f3 8b 45 dc                 mov eax, [rbp-0x24]
00000000:004028f6 48 98                    cdqe
00000000:004028f8 0f b6 84 05 27 fd ff ff  movzx eax, byte [rbp+rax-0x2d9]
00000000:00402900 31 d0                    xor eax, edx
00000000:00402902 88 85 1c fd ff ff        mov [rbp-0x2e4], al
00000000:00402908 0f b6 85 1c fd ff ff     movzx eax, byte [rbp-0x2e4]
00000000:0040290f 0f be d0                 movsx edx, al
00000000:00402912 48 8d 85 50 fd ff ff     lea rax, [rbp-0x2b0]
00000000:00402919 89 d6                    mov esi, edx
00000000:0040291b 48 89 c7                 mov rdi, rax
00000000:0040291e e8 4d f8 ff ff           call LabTestSem12122!std::basic_ostream<char, …
00000000:00402923 83 45 dc 01              add dword [rbp-0x24], 1
00000000:00402927 83 7d dc 08              cmp dword [rbp-0x24], 8
00000000:0040292b 7e b9                    jle 0x4028e6
00000000:0040292d 48 8d 85 50 fd ff ff     lea rax, [rbp-0x2b0]
00000000:00402934 48 89 c7                 mov rdi, rax
00000000:00402937 e8 74 f8 ff ff           call LabTestSem12122!std::basic_ofstream<char,…
```

7. Then the program prints out “Your machine has been infected ;P”

```
00000000:00402937 e8 74 f8 ff ff           call LabTestSem12122!std::basic_ofstream<char,…
00000000:0040293c 48 8b 45 c8              mov rax, [rbp-0x38]
00000000:00402940 48 89 c7                 mov rdi, rax
00000000:00402943 e8 a8 f7 ff ff           call LabTestSem12122!system@plt
00000000:00402948 48 8d 05 27 07 00 00     lea rax, [rel 0x403076]          ASCII "clear"
00000000:0040294f 48 89 c7                 mov rdi, rax
00000000:00402952 e8 99 f7 ff ff           call LabTestSem12122!system@plt
00000000:00402957 48 8d 05 22 07 00 00     lea rax, [rel 0x403080]          ASCII "Your machine has been infected ;P\n"
00000000:0040295e 48 89 c6                 mov rsi, rax
00000000:00402961 48 8d 05 18 28 00 00     lea rax, [rel 0x405180]
00000000:00402968 48 89 c7                 mov rdi, rax
```