2026 상반기 DFRC 신입생 세미나

Reversing Assignment

2026-01-20 (Tue)

조민혁 (Jo Min Hyuk)

cgumgek8@gmail.com

고려대학교 정보보호대학원
Korea University
School of Cybersecurity

DFRC Korea University
Digital Forensic Research Center

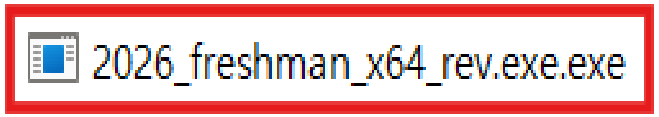
Contents

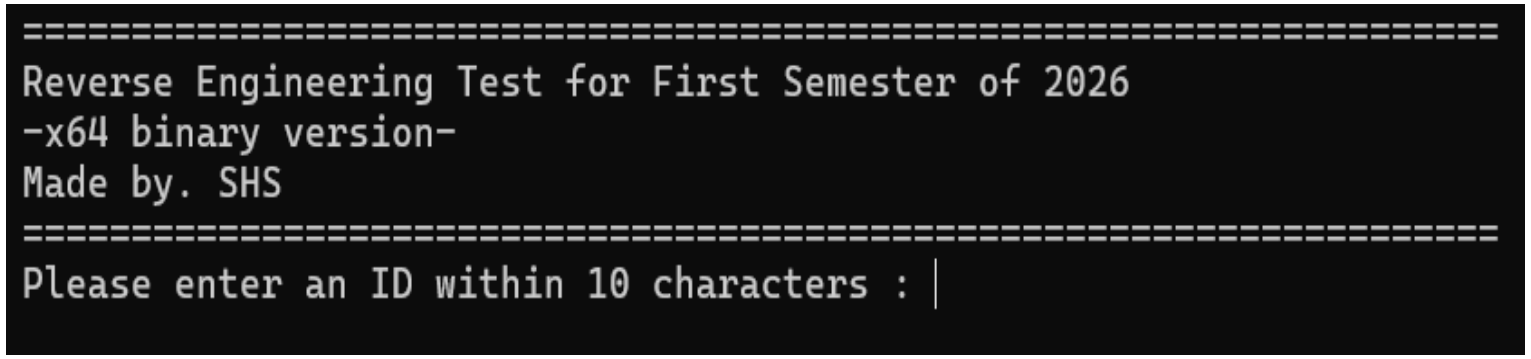
- Introduction to Assignment
- Analysis Environment
- Problem Analysis and Solution
- Conclusion

Introduction to Assignment

Introduction to Assignment

Initial Screen





Introduction to Assignment

Notice

- **No security vulnerabilities involved**
  - 1. Solve the problem using only **static and dynamic analysis.**
  - 2. Solving the problem using methods other than reverse engineering that exploit vulnerabilities will **not receive any additional credit.**

- **Submit answers including both the solving process and the final answer**
  - 1. Be sure to include the **core parts** in your answer.
  - 2. For problems where screen capture is possible, **include screen captures** in your answer.

Analysis Environment

Analysis Environment

Environment and Tools

- **Analysis Environment**
  - Windows 11, x64 Arch

OS 이름	Microsoft Windows 11 Home
버전	10.0.26100 빌드 26100
기타 OS 설명	사용할 수 없음
OS 제조업체	Microsoft Corporation
시스템 이름	MINHYUK
시스템 제조업체	SAMSUNG ELECTRONICS CO., LTD.
시스템 모델	950QCG
시스템 종류	x64 기반 PC
시스템 SKU	SCAI-A5A5-A5A5-A5A5-PAJC
프로세서	Intel(R) Core(TM) i5-1035G4 CPU @ 1.10GHz, 1498Mhz, 4 코어, 8 논리 프...

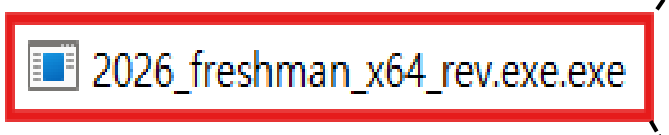
- **Static Analysis Tool**
  - Ghidra 11.4.2 Public

- **Dynamic Analysis Tool**
  - x64dbg

Problem Analysis and Solution

Problem Analysis and Solution

Launch Program




```
==================================================================
Reverse Engineering Test for First Semester of 2026
-x64 binary version-
Made by. SHS
==================================================================
Please enter an ID within 10 characters :
```

Input ID

```
==================================================================
Reverse Engineering Test for First Semester of 2026
-x64 binary version-
Made by. SHS
==================================================================
Please enter an ID within 10 characters : minhyuk
Please enter a flag :
```

Input flag

```
==================================================================
Reverse Engineering Test for First Semester of 2026
-x64 binary version-
Made by. SHS
==================================================================
Please enter an ID within 10 characters : minhyuk
Please enter a flag : min
[System] Welcome to DFRC!
[System] Incorrect! Please try again...:(
```

Problem Analysis and Solution

Basic Problems

- **[문제 1] 동적 분석과 정적 분석에 대해 간단히 설명하고, 리버싱에서 두 분석이 모두 필요한 이유에 대해 간단히 서술하시오. (1점)**
  - **Dynamic Analysis?**

➔A method of analysis that observes a program's behavior while it is actually running. It involves using a debugger to analyze runtime behavior by examining **registers, memory changes, API calls, branches, and executed code.**

  - **Static Analysis?**

➔A method of analysis performed without executing the program. It analyzes the overall structure of the program by examining binaries or source code, **including assembly code, control flow, strings, and function structures.**

Problem Analysis and Solution

Basic Problems

- **[문제 1] 동적 분석과 정적 분석에 대해 간단히 설명하고, 리버싱에서 두 분석이 모두 필요한 이유에 대해 간단히 서술하시오. (1점)**
  - **Why are both analyses necessary in reverse engineering?**

➔ Static analysis alone makes it difficult to identify values determined at **runtime, conditional behavior, obfuscated code, or packed code.**

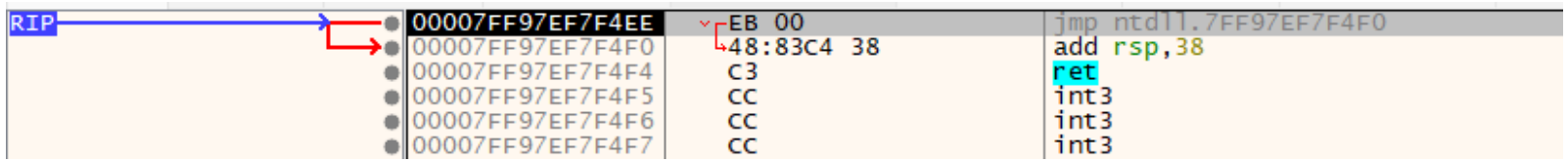
➔ Dynamic analysis alone makes it hard to systematically understand **the entire structure of the binary or all possible execution paths.**

➔ Therefore, **static analysis** is used to understand **the overall structure and determine an analysis strategy,** while **dynamic analysis** is used to analyze **actual behavior and hidden logic**, making reverse engineering more effective.

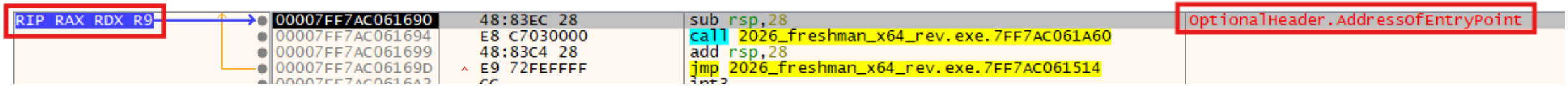
Problem Analysis and Solution

Basic Problems

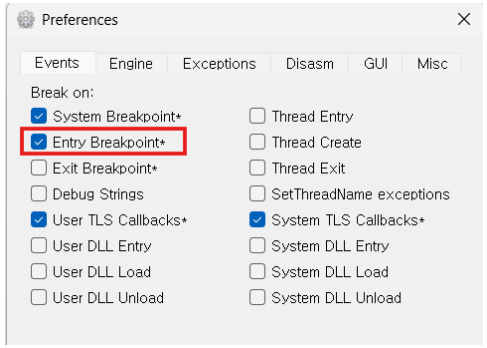
- [문제 2] 동적 분석 도구를 통해 문제 파일을 실행한 후, 해당 파일의 EP(Entry Point)를 식별하시오. (2점)



F9 Click



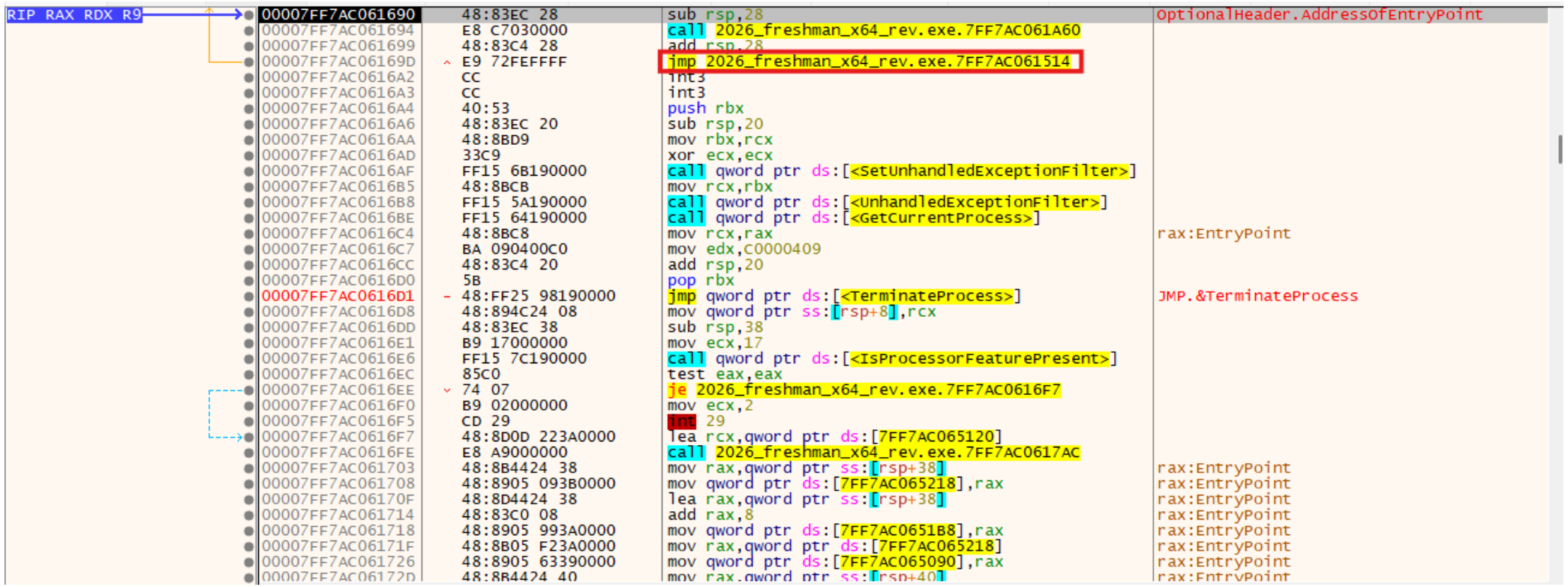
- **Answer:** Address of Entry Point (EP) is **0x7FF7AC061690**



Problem Analysis and Solution

Core Problems

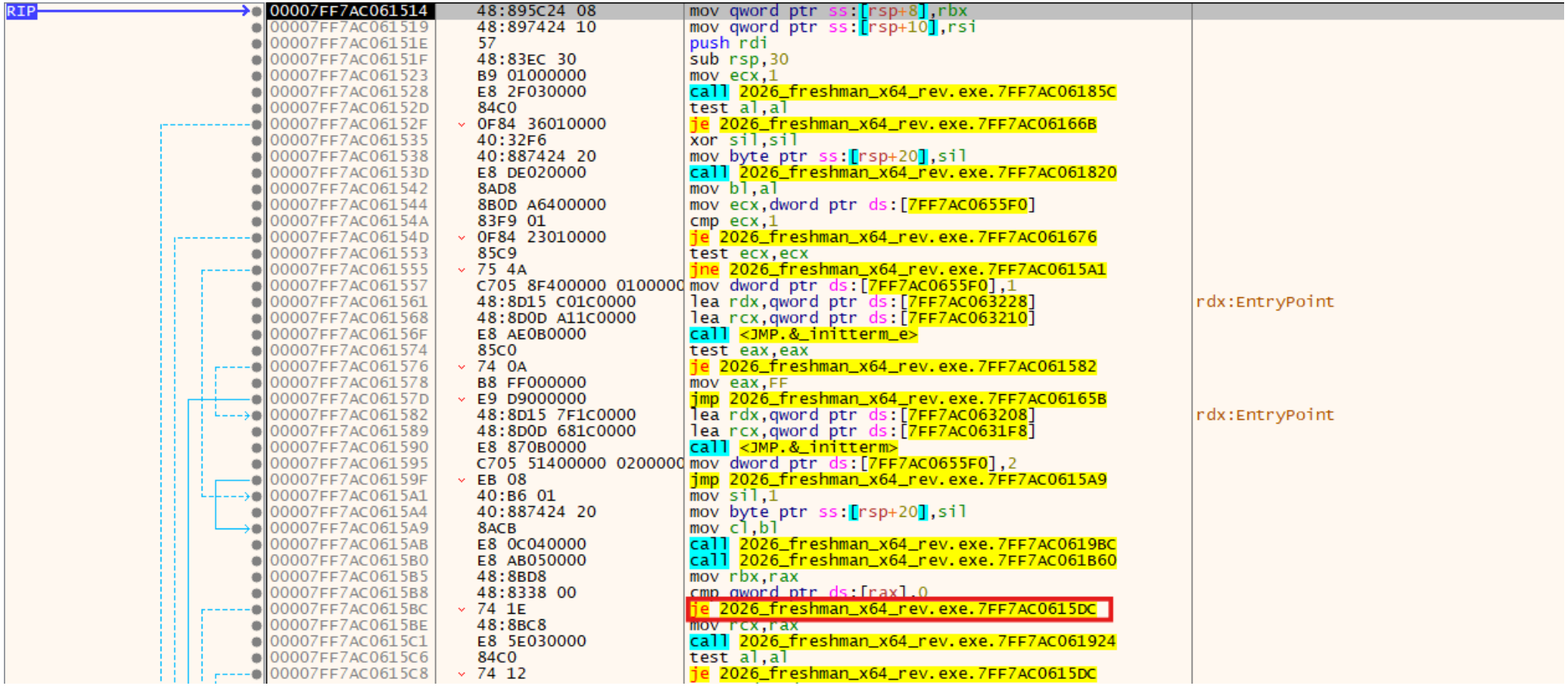
- **[문제 3] 동적 분석을 통해 EP로부터 main함수를 찾아가는 과정을 서술하시오. (2점)**
  - **STEP 1-1)** 'F8' Click, then **jmp** **occurs.**



Problem Analysis and Solution

Core Problems

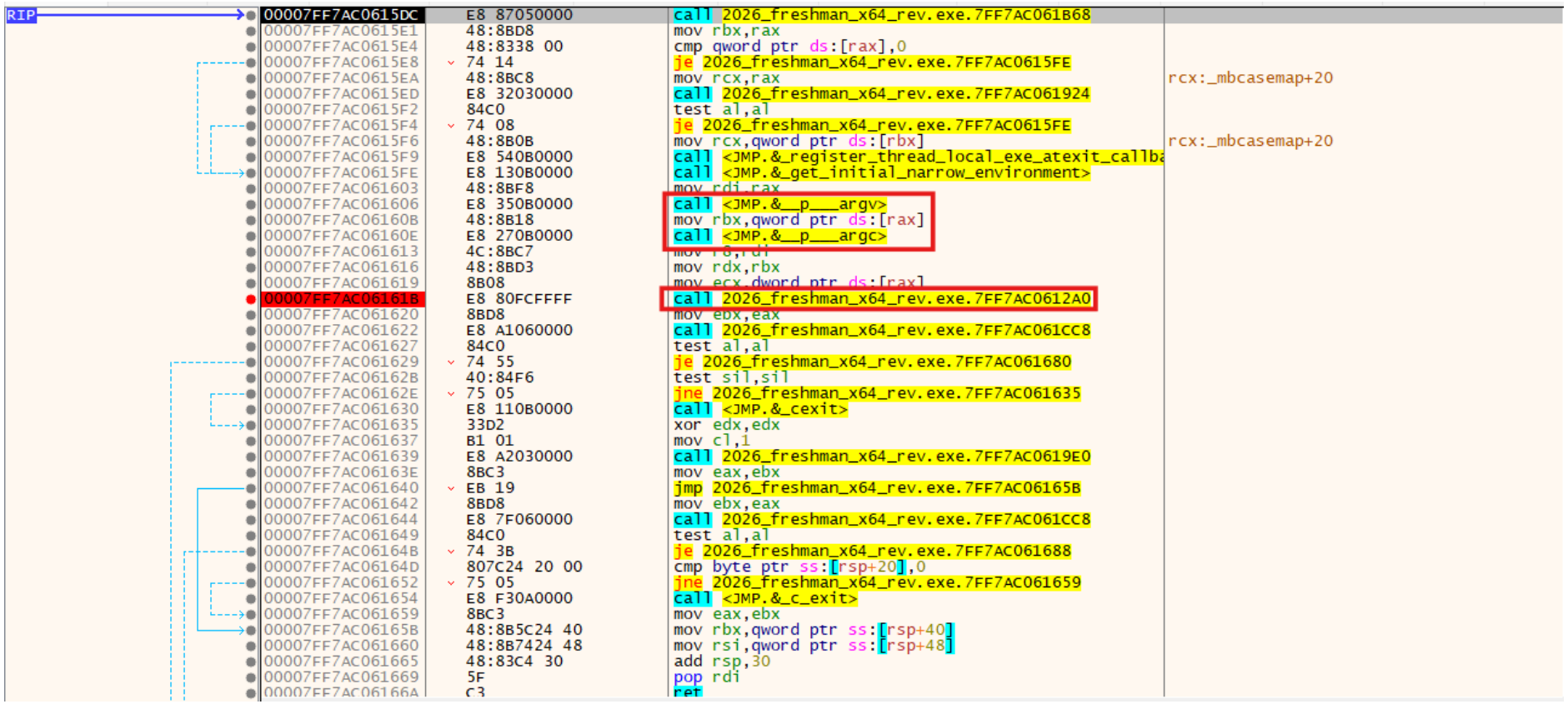
- **[문제 3] 동적 분석을 통해 EP로부터 main함수를 찾아가는 과정을 서술하시오. (2점)**
  - **STEP 1-2)** 'F8' Click, then **je occurs.**



Problem Analysis and Solution

Core Problems

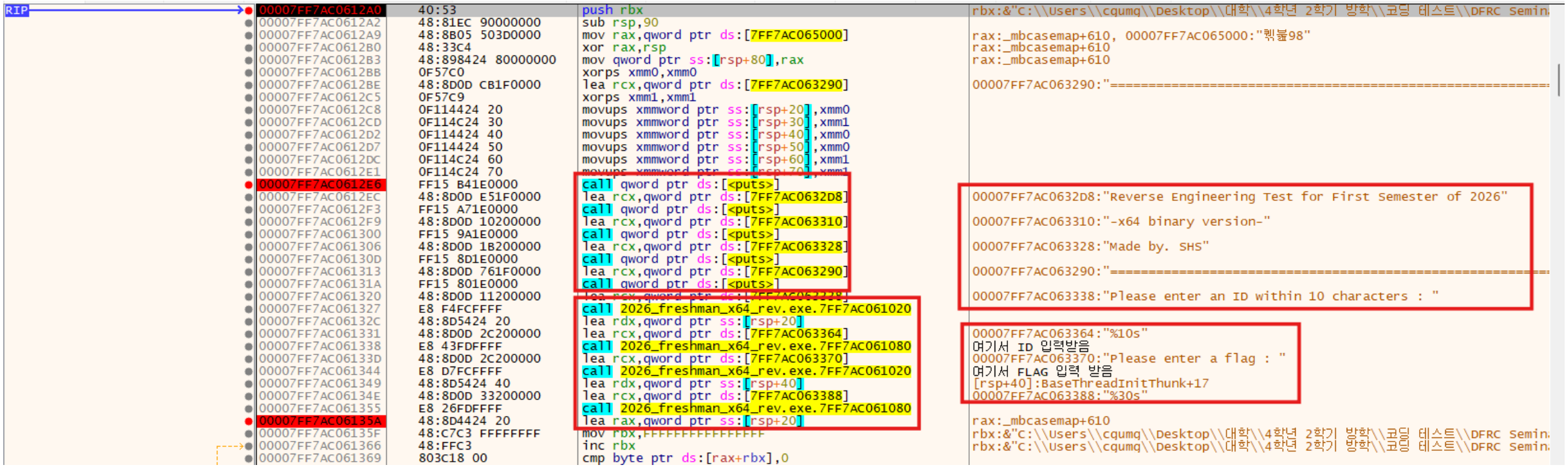
- **[문제 3] 동적 분석을 통해 EP로부터 main함수를 찾아가는 과정을 서술하시오. (2점)**
  - **STEP 2)** Then, we can find **'argv', 'argc'**, and **call occurs. Let's keep clicking F8.**



Problem Analysis and Solution

Core Problems

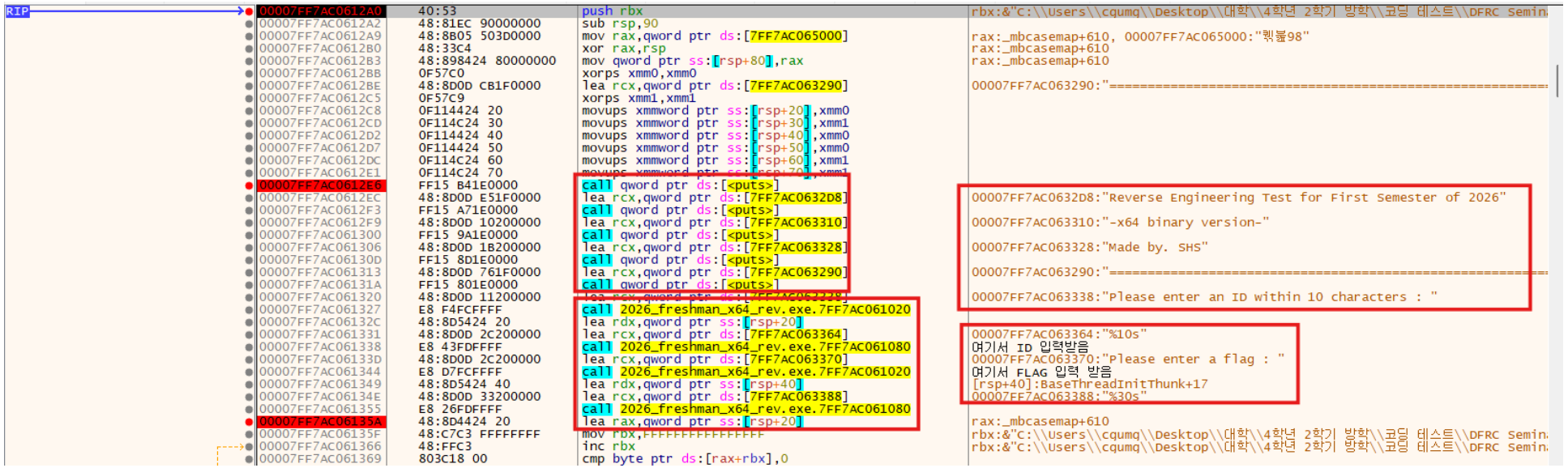
- **[문제 3] 동적 분석을 통해 EP로부터 main함수를 찾아가는 과정을 서술하시오. (2점)**
  - **STEP 3)** Then we can find **console message.** Therefore, this part is **main function.**



Problem Analysis and Solution

Core Problems

- **[문제 4] main함수의 전체 코드 흐름에 대해 본인이 파악한 대로 추론하여 서술하시오. (동적 및 정적 분석 중 본인이 편한 방법을 사용하되, 두 방법 모두 사용해도 무관함) (3점)**
  - **STEP 1)** Through **dynamic analysis**, it can be observed that the program **prints a message** using the puts function and **receives input** from the user. However, the subsequent execution flow was **too complex** to fully trace using dynamic analysis alone.



Problem Analysis and Solution

Core Problems

- **[문제 4] main함수의 전체 코드 흐름에 대해 본인이 파악한 대로 추론하여 서술하시오. (동적 및 정적 분석 중 본인이 편한 방법을 사용하되, 두 방법 모두 사용해도 무관함) (3점)**
  - **STEP 2)** Through static analysis, it can be identified that in **FUN_1400012a0**, messages are **printed via puts**, and **user input is received** through **FUN_140001020** and **FUN_140001080**.

```
void FUN_1400012a0(undefined8 param_1,undefined8 param_2,undefined8 param_3,undefined8 param_4)

puts("================================================================");
puts("Reverse Engineering Test for First Semester of 2026");
puts("-x64 binary version-");
puts("Made by. SHS");
puts("================================================================");
FUN_140001020("Please enter an ID within 10 characters : ",param_2,param_3,param_4);
pcVar3 = local_78;
FUN_140001080(&DAT_140003364,pcVar3,param_3,param_4);
FUN_140001020("Please enter a flag : ",pcVar3,param_3,param_4);
FUN_140001080(&DAT_140003388,local_58,param_3,param_4);
```

```
local_res10 = param_2;
local_res18 = param_3;
local_res20 = param_4;
uVar1 = __acrt_iob_func(1);
puVar2 = (undefined8 *)FUN_140001000();
__stdio_common_vfprintf(*puVar2,uVar1,param_1,0,&local_res10);
return;
```

```
local_res10 = param_2;
local_res18 = param_3;
local_res20 = param_4;
uVar1 = __acrt_iob_func(0);
puVar2 = (undefined8 *)FUN_140001010();
__stdio_common_vfscanf(*puVar2,uVar1,param_1,0,&local_res10);
return;
```

Problem Analysis and Solution

Core Problems

- **[문제 4] main함수의 전체 코드 흐름에 대해 본인이 파악한 대로 추론하여 서술하시오. (동적 및 정적 분석 중 본인이 편한 방법을 사용하되, 두 방법 모두 사용해도 무관함) (3점)**
  - **STEP 3)** After that, **FUN_140001110** is called, indicating that it performs a certain function.
    - Since it appears to **modify param_1**, it is presumed to **manipulate the input value**.

```
uVar6 = 0xffffffffffffffff;
do {
  uVar5 = uVar6 + 1;
  lVar1 = uVar6 + 1;
  uVar6 = uVar5;
} while (local_78[lVar1] != '\0');
lVar1 = 0;
do {
  pcVar3 = local_78 + lVar1;
  *(char *)((longlong)&local_68 + lVar1) = *pcVar3;
  lVar1 = lVar1 + 1;
} while (*pcVar3 != '\0');
iVar4 = (int)uVar5;
FUN_140001110((undefined1 *)&local_68,iVar4);
```

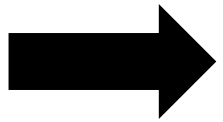
```
void FUN_140001110(undefined1 *param_1,int param_2)

{
  undefined1 uVar1;
  undefined1 *puVar2;
  longlong lVar3;

  puts("[System] Welcome to DFRC!");
  lVar3 = (longlong)(param_2 / 2);
  if (0 < param_2 / 2) {
    puVar2 = param_1 + (longlong)param_2 + -1;
    do {
      uVar1 = *param_1;
      *param_1 = *puVar2;
      param_1 = param_1 + 1;
      *puVar2 = uVar1;
      lVar3 = lVar3 + -1;
      puVar2 = puVar2 + -1;
    } while (lVar3 != 0);
  }
  return;
}
```

Problem Analysis and Solution

Core Problems

- **[문제 4] main함수의 전체 코드 흐름에 대해 본인이 파악한 대로 추론하여 서술하시오. (동적 및 정적 분석 중 본인이 편한 방법을 사용하되, 두 방법 모두 사용해도 무관함) (3점)**
  - **STEP 4)** After that, **FUN_140001180** is called, indicating that it performs a certain function.
    - Since it appears to **XOR arbitrary bytes**, it is inferred that this function is related to **key generation**.

```
FUN_140001180(&local_68,iVar4,&local_38);
```



```
void FUN_140001180(void *param_1,int param_2,void *param_3)

{
  byte bVar1;
  size_t _Size;

  bVar1 = FUN_1400010e0();
  _Size = (size_t)param_2;
  bVar1 = bVar1 ^ (byte)param_2;
  memcpy(param_3,param_1,_Size);
  *(byte *)(_Size + 2 + (longlong)param_3) = bVar1 ^ 0x78;
  *(byte *)(_Size + (longlong)param_3) = bVar1 ^ 0x74;
  *(byte *)(_Size + 3 + (longlong)param_3) = bVar1 ^ 0x67;
  *(byte *)(_Size + 1 + (longlong)param_3) = bVar1 ^ 0x35;
  *(byte *)(_Size + 4 + (longlong)param_3) = bVar1 ^ 0x54;
  *(byte *)(_Size + 5 + (longlong)param_3) = bVar1 ^ 0x75;
  *(byte *)(_Size + 6 + (longlong)param_3) = bVar1 ^ 0x4b;
  *(byte *)(_Size + 7 + (longlong)param_3) = bVar1 ^ 0x70;
  *(byte *)(_Size + 8 + (longlong)param_3) = bVar1 ^ 0x69;
  return;
}
```

Problem Analysis and Solution

Core Problems

- **[문제 4] main함수의 전체 코드 흐름에 대해 본인이 파악한 대로 추론하여 서술하시오. (동적 및 정적 분석 중 본인이 편한 방법을 사용하되, 두 방법 모두 사용해도 무관함) (3점)**
  - **STEP 5)** After that, **FUN_140001220** is called, indicating that it performs a certain function.
    - Since **strncmp** is used, the input value is compared with the correct **answer**.
    - Afterward, the function returns whether the input is correct through **additional validation.**

```
uVar2 = FUN_140001220(local_58,(char *)&local_38,iVar4);
```

```
undefined8 FUN_140001220(char *param_1,char *param_2,int param_3)

{
  byte bVar1;
  int iVar2;
  undefined8 uVar3;
  char *_Str1;
  char *_Str2;

  _Str1 = param_1;
  _Str2 = param_2;
  bVar1 = FUN_1400010e0();
  iVar2 = strncmp(_Str1,_Str2,(longlong)param_3);
  if (iVar2 == 0) {
    iVar2 = 0;
    do {
      if ((byte)(param_1[iVar2 + param_3] + 2U ^ bVar1 ^ (byte)param_3) != param_2[iVar2 + param_3
      ])
        goto LAB_140001282;
      iVar2 = iVar2 + 1;
    } while (iVar2 < 9);
    uVar3 = 1;
  }
  else {
LAB_140001282:
    uVar3 = 0;
  }
  return uVar3;
```

Problem Analysis and Solution

Core Problems

- **[문제 4] main함수의 전체 코드 흐름에 대해 본인이 파악한 대로 추론하여 서술하시오. (동적 및 정적 분석 중 본인이 편한 방법을 사용하되, 두 방법 모두 사용해도 무관함) (3점)**
  - **STEP 6)** After that, depending on **whether the input is correct**, **a success or failure message is displayed**, and the program then **terminates**

```
pcVar3 = "[System] Congratulation! You have answered correctly!";
if ((int)uVar2 == 0) {
  pcVar3 = "[System] Incorrect! Please try again...:(";
}
puts(pcVar3);
FUN_140001080(&DAT_1400033f4,local_58,uVar5,param_4);
FUN_140001410(local_18 ^ (ulonglong)auStack_98);
return;
```

Problem Analysis and Solution

Core Problems

- **[문제 4] main함수의 전체 코드 흐름에 대해 본인이 파악한 대로 추론하여 서술하시오. (동적 및 정적 분석 중 본인이 편한 방법을 사용하되, 두 방법 모두 사용해도 무관함) (3점)**
  - Summary of the **overall code flow** of the **main function**

STEP 2) Calls **FUN_140001020** and **FUN_140001080** to execute **puts, printf, and scanf**

STEP 3) Calls **FUN_140001110** to **manipulate the input value**

STEP 4) Calls **FUN_140001180** to perform **key generation**

STEP 5) Calls **FUN_140001220** to **compare** the input value with the correct answer

STEP 6) **Outputs a success or failure message** depending on whether the answer is correct and **terminates** the program

Problem Analysis and Solution

Core Problems

- **[문제 5] 아래의 문자열을 찾고, 해당 문자열이 위치한 함수의 동작 원리 및 역할에 대해 자세히 서술하시오. (4점)**
  - Welcome to DFRC!
  - Based on Ghidra, it can be found in **FUN_140001110**

```
void FUN_140001110(undefined1 *param_1,int param_2)

{
  undefined1 uVar1;
  undefined1 *puVar2;
  longlong lVar3;

  puts "[System] Welcome to DFRC!");
  lVar3 = (longlong)(param_2 / 2);
  if (0 < param_2 / 2) {
    puVar2 = param_1 + (longlong)param_2 + -1;
    do {
      uVar1 = *param_1;
      *param_1 = *puVar2;
      param_1 = param_1 + 1;
      *puVar2 = uVar1;
      lVar3 = lVar3 + -1;
      puVar2 = puVar2 + -1;
    } while (lVar3 != 0);
  }
  return;
}
```

Problem Analysis and Solution

Core Problems

- **Operation mechanism and role**
  - **local_68** receives the value after the ID is stored in **local_78** and passed through **pcVar3.**

    That is, **local_68** represents the **ID** and corresponds to **param_1**.
  - **iVar4** receives the value after **uVar5** stores the length of the ID.

    That is, **iVar4** represents the ID size and corresponds to **param_2**.

```
FUN_140001020("Please enter an ID within 10 characters : ",param_2,param_3,param_4);
pcVar3 = local_78;
FUN_140001080(&DAT_140003364,pcVar3,param_3,param_4);
FUN_140001020("Please enter a flag : ",pcVar3,param_3,param_4);
FUN_140001080(&DAT_140003388,local_58,param_3,param_4);
uVar6 = 0xffffffffffffffff;
do {
  uVar5 = uVar6 + 1;
  lVar1 = uVar6 + 1;
  uVar6 = uVar5;
} while (local_78[lVar1] != '\0');
lVar1 = 0;
do {
  pcVar3 = local_78 + lVar1;
  *(char *)((longlong)&local_68 + lVar1) = *pcVar3;
  lVar1 = lVar1 + 1;
} while (*pcVar3 != '\0');
iVar4 = (int)uVar5;
FUN_140001110(&local_68,(undefined1 *)&local_68,iVar4);
```

Problem Analysis and Solution

Core Problems

- **Operation mechanism and role**
  - **puVar2** points to the end of the ID array.
  - **uVar1** stores a backup of **param_1**.
  - **param_1** is assigned the value of **puVar2.**

➔ That is, by assigning **param_1** to the end of the ID array, the value of **param_1** is being reversed.

➔ Therefore, this function can be identified as an **id_reverse function.**

```
void FUN_140001110(undefined1 *param_1,int param_2)

{
  undefined1 uVar1;
  undefined1 *puVar2;
  longlong lVar3;

  puts("[System] Welcome to DFRC!");
  lVar3 = (longlong)(param_2 / 2);
  if (0 < param_2 / 2) {
    puVar2 = param_1 + (longlong)param_2 + -1;
    do {
      uVar1 = *param_1;
      *param_1 = *puVar2;
      param_1 = param_1 + 1;
      *puVar2 = uVar1;
      lVar3 = lVar3 + -1;
      puVar2 = puVar2 + -1;
    } while (lVar3 != 0);
  }
  return;
}
```

Problem Analysis and Solution

Core Problems

- [문제 6] 프로그램의 Key 생성 방식에 대해 서술하시오. (4점)

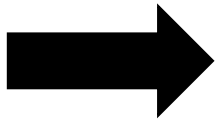
```
void FUN_140001180(void *param_1,int param_2,void *param_3)

{
  byte bVar1;
  size_t _Size;

  bVar1 = FUN_1400010e0();
  _Size = (size_t)param_2;
  bVar1 = bVar1 ^ (byte)param_2;
  memcpy(param_3,param_1,_Size);
  *(byte *)(_Size + 2 + (longlong)param_3) = bVar1 ^ 0x78;
  *(byte *)(_Size + (longlong)param_3) = bVar1 ^ 0x74;
  *(byte *)(_Size + 3 + (longlong)param_3) = bVar1 ^ 0x67;
  *(byte *)(_Size + 1 + (longlong)param_3) = bVar1 ^ 0x35;
  *(byte *)(_Size + 4 + (longlong)param_3) = bVar1 ^ 0x54;
  *(byte *)(_Size + 5 + (longlong)param_3) = bVar1 ^ 0x75;
  *(byte *)(_Size + 6 + (longlong)param_3) = bVar1 ^ 0x4b;
  *(byte *)(_Size + 7 + (longlong)param_3) = bVar1 ^ 0x70;
  *(byte *)(_Size + 8 + (longlong)param_3) = bVar1 ^ 0x69;
  return;
}
```

Problem Analysis and Solution

Core Problems

- **[문제 6] 프로그램의 Key 생성 방식에 대해 서술하시오. (4점)**
  - **local_68** holds the ID in its **reversed state.** **(param_1)**
  - **iVar4** represents the **length of the ID.** **(param_2)**
  - **local_38** is a buffer used to store the **result of the function execution.** **(param_3)**

```
FUN_140001180(&local_68,iVar4,&local_38);
```



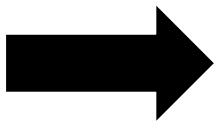
```
void FUN_140001180(void *param_1,int param_2,void *param_3)

{
  byte bVar1;
  size_t _Size;

  bVar1 = FUN_1400010e0();
  _Size = (size_t)param_2;
  bVar1 = bVar1 ^ (byte)param_2;
  memcpy(param_3,param_1,_Size);
  *(byte *)(_Size + 2 + (longlong)param_3) = bVar1 ^ 0x78;
  *(byte *)(_Size + (longlong)param_3) = bVar1 ^ 0x74;
  *(byte *)(_Size + 3 + (longlong)param_3) = bVar1 ^ 0x67;
  *(byte *)(_Size + 1 + (longlong)param_3) = bVar1 ^ 0x35;
  *(byte *)(_Size + 4 + (longlong)param_3) = bVar1 ^ 0x54;
  *(byte *)(_Size + 5 + (longlong)param_3) = bVar1 ^ 0x75;
  *(byte *)(_Size + 6 + (longlong)param_3) = bVar1 ^ 0x4b;
  *(byte *)(_Size + 7 + (longlong)param_3) = bVar1 ^ 0x70;
  *(byte *)(_Size + 8 + (longlong)param_3) = bVar1 ^ 0x69;
  return;
}
```

Problem Analysis and Solution

Core Problems

- **[문제 6] 프로그램의 Key 생성 방식에 대해 서술하시오. (4점)**
  - First, the function **FUN_1400010e0** is called and its return value is stored in **bVar1**.
  - The function **FUN_1400010e0** always **returns 0xF9.**

```
void FUN_140001180(void *param_1,int param_2,void *param_3)

{
  byte bVar1;
  size_t _Size;

  bVar1 = FUN_1400010e0();
  _Size = (size_t)param_2;
  bVar1 = bVar1 ^ (byte)param_2;
  memcpy(param_3,param_1,_Size);
  *(byte *)(_Size + 2 + (longlong)param_3) = bVar1 ^ 0x78;
  *(byte *)(_Size + (longlong)param_3) = bVar1 ^ 0x74;
  *(byte *)(_Size + 3 + (longlong)param_3) = bVar1 ^ 0x67;
  *(byte *)(_Size + 1 + (longlong)param_3) = bVar1 ^ 0x35;
  *(byte *)(_Size + 4 + (longlong)param_3) = bVar1 ^ 0x54;
  *(byte *)(_Size + 5 + (longlong)param_3) = bVar1 ^ 0x75;
  *(byte *)(_Size + 6 + (longlong)param_3) = bVar1 ^ 0x4b;
  *(byte *)(_Size + 7 + (longlong)param_3) = bVar1 ^ 0x70;
  *(byte *)(_Size + 8 + (longlong)param_3) = bVar1 ^ 0x69;
  return;
}
```



```
undefined1 FUN_1400010e0(void)

{
  if (DAT_14000565c == 0) {
    DAT_140005668 = 0xf9;
    DAT_14000565c = 1;
    return 0xf9;
  }
  return DAT_140005668;
}
```

Problem Analysis and Solution

Core Problems

- **[문제 6] 프로그램의 Key 생성 방식에 대해 서술하시오. (4점)**
  - **_Size** represents the **length of the ID**, and **bVar1** is obtained by **XOR-ing 0xF9 with the ID length.**
  - After that, the **reversed ID** is copied into **param_3**.
  - Then, **bVar1** is **XOR-ed with a specific fixed byte value** and stored in **param_3**.
  - That is, **param_3[0 ~ _Size-1]** contains the **reversed ID value.**
  - **param_3[_Size ~ _Size+8]** contains the result of **0xF9 ^ ID length ^ {0x74, 0x35, 0x78, 0x67, 0x54, 0x75, 0x4B, 0x70, 0x69}.**
  - Ultimately, **local_38 (param_3)** consists of **reversed_ID | (0xF9 ^ ID length ^ fixed bytes)**.

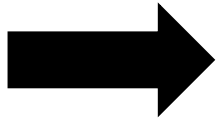
```
void FUN_140001180(void *param_1,int param_2,void *param_3)

{
  byte bVar1;
  size_t _Size;

  bVar1 = FUN_1400010e0();
  _Size = (size_t)param_2;
  bVar1 = bVar1 ^ (byte)param_2;
  memcpy(param_3,param_1,_Size);
  *(byte *)(_Size + 2 + (longlong)param_3) = bVar1 ^ 0x78;
  *(byte *)(_Size + (longlong)param_3) = bVar1 ^ 0x74;
  *(byte *)(_Size + 3 + (longlong)param_3) = bVar1 ^ 0x67;
  *(byte *)(_Size + 1 + (longlong)param_3) = bVar1 ^ 0x35;
  *(byte *)(_Size + 4 + (longlong)param_3) = bVar1 ^ 0x54;
  *(byte *)(_Size + 5 + (longlong)param_3) = bVar1 ^ 0x75;
  *(byte *)(_Size + 6 + (longlong)param_3) = bVar1 ^ 0x4b;
  *(byte *)(_Size + 7 + (longlong)param_3) = bVar1 ^ 0x70;
  *(byte *)(_Size + 8 + (longlong)param_3) = bVar1 ^ 0x69;
  return;
}
```

Problem Analysis and Solution

Advanced Problems

- [문제 7] 현재까지 해결했던 문제들을 바탕으로, 제공된 프로그램의 정답을 출력하시오. (프로그램의 ID, Flag, 성공 메시지 모두 나오도록 **캡쳐**하여 답안을 작성) (4점)

```
uVar2 = FUN_140001220(local_58,(char *)&local_38,iVar4);
```



```
undefined8 FUN_140001220(char *param_1,char *param_2,int param_3)

{
  byte bVar1;
  int iVar2;
  undefined8 uVar3;
  char *_Str1;
  char *_Str2;

  _Str1 = param_1;
  _Str2 = param_2;
  bVar1 = FUN_1400010e0();
  iVar2 = strncmp(_Str1,_Str2,(longlong)param_3);
  if (iVar2 == 0) {
    iVar2 = 0;
    do {
      if ((byte)(param_1[iVar2 + param_3] + 2U ^ bVar1 ^ (byte)param_3) != param_2[iVar2 + param_3
      ])
      goto LAB_140001282;
      iVar2 = iVar2 + 1;
    } while (iVar2 < 9);
    uVar3 = 1;
  }
  else {
LAB_140001282:
    uVar3 = 0;
  }
  return uVar3;
```

Problem Analysis and Solution

Advanced Problems

- **[문제 7] 현재까지 해결했던 문제들을 바탕으로, 제공된 프로그램의 정답을 출력하시오. (프로그램의 ID, Flag, 성공 메시지 모두 나오도록 캡쳐하여 답안을 작성) (4점)**
  - **local_58** is input flag. **(param_1)**
  - **local_38**은 (**reverse_ID | 0xF9 ^ ID_length ^ fixed_bytes**) **(param_2)**
  - **iVar4**는 **ID length (param_3)**

```
FUN_140001080(&DAT_140003364,pcVar3,param_3,param_4);
FUN_140001020("Please enter a flag : ",pcVar3,param_3,param_4);
FUN_140001080(&DAT_140003388,local_58,param_3,param_4);
uVar6 = 0xffffffffffffffff;
do {
  uVar5 = uVar6 + 1;
  lVar1 = uVar6 + 1;
  uVar6 = uVar5;
} while (local_78[lVar1] != '\0');
lVar1 = 0;
do {
  pcVar3 = local_78 + lVar1;
  *(char *)((longlong)&local_68 + lVar1) = *pcVar3;
  lVar1 = lVar1 + 1;
} while (*pcVar3 != '\0');
iVar4 = (int)uVar5;
FUN_140001110((undefined1 *)&local_68,iVar4);
FUN_140001180(&local_68,iVar4,&local_38);
uVar5 = uVar5 & 0xffffffff;
uVar2 = FUN_140001220(local_58,(char *)&local_38,iVar4);
```

Problem Analysis and Solution

Advanced Problems

- **[문제 7] 현재까지 해결했던 문제들을 바탕으로, 제공된 프로그램의 정답을 출력하시오. (프로그램의 ID, Flag, 성공 메시지 모두 나오도록 캡쳐하여 답안을 작성) (4점)**
  - **_Str1** contains **local_58 (input flag)**.
  - **_Str2** contains **local_38 (reversed_ID | 0xF9 ^ ID_length ^ fixed_bytes)**.
  - **bVar1** stores the value **0xF9**.
  - Using **strncmp,** the flag is compared with reversed_ID for the length of the ID.
  - Therefore, the **prefix of the flag must match the reversed ID**.

```
undefined8 FUN_140001220(char *param_1,char *param_2,int param_3)

{
  byte bVar1;
  int iVar2;
  undefined8 uVar3;
  char *_Str1;
  char *_Str2;

  _Str1 = param_1;
  _Str2 = param_2;
  bVar1 = FUN_1400010e0();
  iVar2 = strncmp(_Str1,_Str2,(longlong)param_3);
  if (iVar2 == 0) {
    iVar2 = 0;
    do {
      if ((byte)(param_1[iVar2 + param_3] + 2U ^ bVar1 ^ (byte)param_3) != param_2[iVar2 + param_3
      ])
        goto LAB_140001282;
      iVar2 = iVar2 + 1;
    } while (iVar2 < 9);
    uVar3 = 1;
  }
  else {
LAB_140001282:
    uVar3 = 0;
  }
  return uVar3;
```

Problem Analysis and Solution

Advanced Problems

- **[문제 7] 현재까지 해결했던 문제들을 바탕으로, 제공된 프로그램의 정답을 출력하시오. (프로그램의 ID, Flag, 성공 메시지 모두 나오도록 캡쳐하여 답안을 작성) (4점)**
  - After that, **comparison of the remaining part** begins using **param_1[iVar2 + param_3].**
  - The condition can be interpreted as follows:
    - **Left-hand side**: (FLAG[iVar2 + param_3] + 2) ^ 0xF9 ^ param_3
    - **Right-hand side**: 0xF9 ^ param_3 ^ fixed bytes
  - Thus, **FLAG[iVar2 + param_3] = fixed_byte − 2.**
  - **(0x74, 0x35, 0x78, 0x67, 0x54, 0x75, 0x4B, 0x70, 0x69) − 2**
    = t5xgTuKpi − 2
    = r3veRsIng

  Therefore, **the suffix of the flag** is **r3veRsIng.**

```
undefined8 FUN_140001220(char *param_1,char *param_2,int param_3)

{
  byte bVar1;
  int iVar2;
  undefined8 uVar3;
  char *_Str1;
  char *_Str2;

  _Str1 = param_1;
  _Str2 = param_2;
  bVar1 = FUN_1400010e0();
  iVar2 = strncmp(_Str1,_Str2,(longlong)param_3);
  if (iVar2 == 0) {
    iVar2 = 0;
    do {
      if ((byte)(param_1[iVar2 + param_3] + 2U ^ bVar1 ^ (byte)param_3) != param_2[iVar2 + param_3
      ])
        goto LAB_140001282;
      iVar2 = iVar2 + 1;
    } while (iVar2 < 9);
    uVar3 = 1;
  }
  else {
LAB_140001282:
    uVar3 = 0;
  }
  return uVar3;
```

Problem Analysis and Solution

Advanced Problems

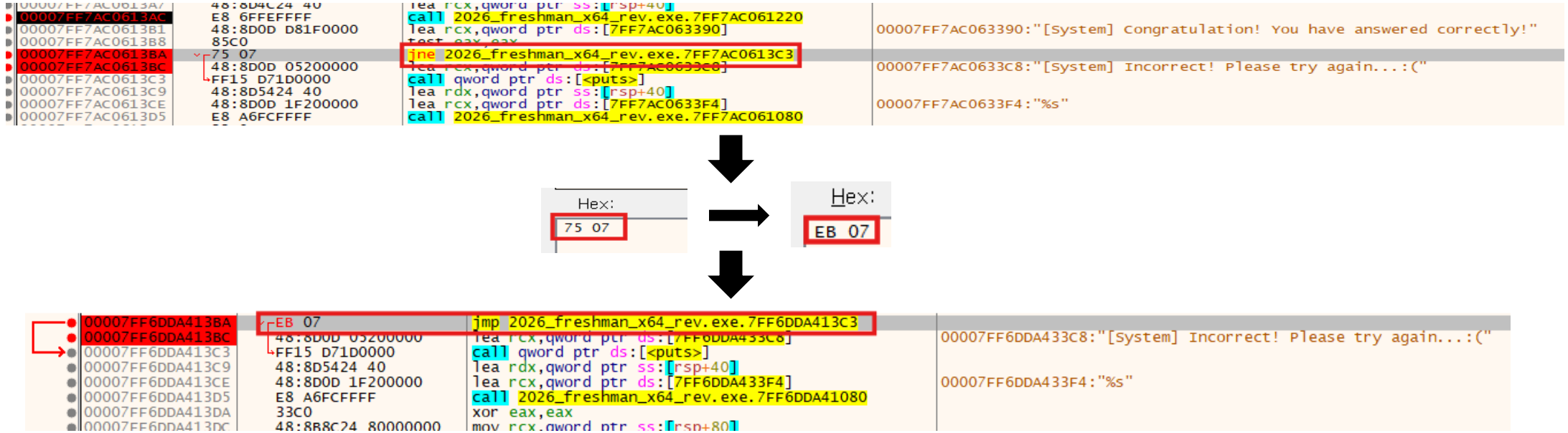
- **[문제 7] 현재까지 해결했던 문제들을 바탕으로, 제공된 프로그램의 정답을 출력하시오. (프로그램의 ID, Flag, 성공 메시지 모두 나오도록 캡쳐하여 답안을 작성) (4점)**
  - Therefore, composition of the flag is **'reverse_ID | r3veRsIng'**

```
==========================================================
Reverse Engineering Test for First Semester of 2026
-x64 binary version-
Made by. SHS
==========================================================
Please enter an ID within 10 characters : minhyuk
Please enter a flag : kuyhnimr3veRsIng
[System] Welcome to DFRC!
[System] Congratulation! You have answered correctly!
```

Problem Analysis and Solution

Advanced Problems

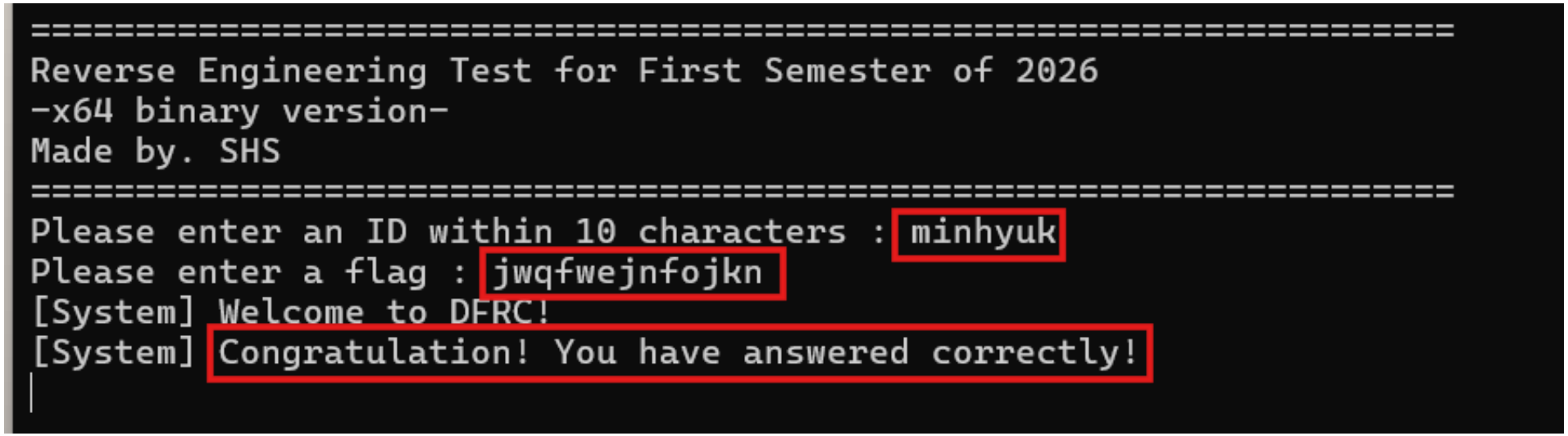
- **[문제 8] 인라인 패치를 통해 입력 값과 관계 없이 항상 정답 메시지가 출력되도록 프로그램을 수정하시오. (패치한 위치, 성공 메시지 모두 나오도록 캡쳐하여 답안을 작성) (5점)**
  - During dynamic analysis, the execution path changes depending on the **jne** instruction. Therefore, an inline patch was applied by changing **jne** to **jmp (Unconditional jump).**



Problem Analysis and Solution

Advanced Problems

- [문제 8] 인라인 패치를 통해 입력 값과 관계 없이 항상 정답 메시지가 출력되도록 프로그램을 수정하시오. (패치한 위치, 성공 메시지 모두 나오도록 캡쳐하여 답안을 작성) (5점)



Problem Analysis and Solution

Additional Problems

- **[문제 9] 리버싱과 관련된 경험 (연구, 개인 공부, 대회, 워게임 등) 이 있다면 적어주세요. (1점)**
  - **[Ransomware Reversing]**
    - I have experience analyzing the **Conti ransomware** as part of a previous undergraduate assignment.
    - At the time, I identified **symmetric and asymmetric cryptographic functions** and their **invocation points** through **static analysis** (using objdump) and **dynamic analysis** (using gdb).
    - I then neutralized the symmetric encryption via **ELF injection**–based binary injection, and blocked the asymmetric encryption by overriding dynamic libraries using **LD_PRELOAD.**

  - **[Copy-and-Patch JIT Compiler Reversing]**
    - Conducted reverse engineering on an open-source **Copy-and-Patch JIT Compiler for research purposes.**
    - Identified that **system calls are used to change memory permissions** to executable when frequently used code regions are copy-and-patched.
    - Exploited this behavior to **obtain a shell**, and the research findings were **published as a paper**.

  - **Reference-1:** Conti-Ransomeware Reversing
  - **Reference-2:** Copy-and-Patch JIT Compiler Reversing Paper

Problem Analysis and Solution

Additional Problems

- **[문제 10] 본인이 연구실에서 리버싱을 통해 진행하고 싶은 연구 주제를 자유롭게 적어주세요. (1점)**
  - Research on Incident Response and Digital Forensics through Reverse Engineering Analysis of RaaS-based Ransomware **(Draft)**

    - Research Motivation

    : The widespread adoption of Ransomware-as-a-Service (RaaS) has increased the complexity of analyzing ransomware-related security incidents.

    - **Research Objective**

    : This study aims to analyze the internal behavior and configuration of ransomware through reverse engineering and to apply the findings to incident response and digital forensics.

    - **Expected Outcomes**

    : The analysis is expected to support accurate damage assessment, reconstruction of attack timelines, and identification of attacks conducted by the same threat actor.

Conclusion

Conclusion

- This problem effectively demonstrated the efficiency of both dynamic and static analysis.
- First, the entry point (EP) was identified through dynamic analysis, and the overall behavior of the main function was inferred.
- Subsequently, the behavior and roles of each function called within main were analyzed.
- By understanding the behavior and roles of these functions, the structure of the FLAG could be derived.
- Additionally, a success message could be obtained simply by modifying the machine code through an inline patch.

- From a personal perspective, this assignment helped me regain interest in reverse engineering after having stepped away from it for some time, and it clearly illustrated why both static and dynamic analysis need to be used together.

Thank You