Assignment3 Team6 Report

Haohan Fu, Jiayang Xu, Xibin Yu, Xi Wang, Zhengyang Cheng and Haoyu Ju

26, March, 2025

1. 2. / 3. / / 4. /

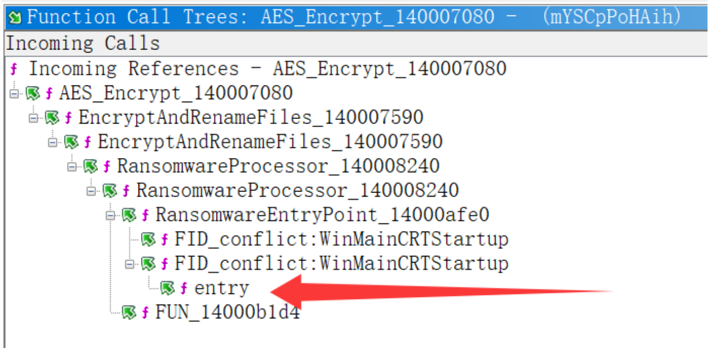
1 Analyze the malware's code

1.1 Start

We used Ghidra to analyze the malware code. We found it difficult to find the code that implements the encryption function directly from the entry point, so we started at defined strings in the program. Then we found the AES encryption function AES_Encrypt_140007080, whose function call tree is shown in Figure 1:

1.2 The AES encryption function

AES_Encrypt_140007080 is the AES encryption function.



```
Function Call Trees: AES_Encrypt_140007080 -  (mYSCpPoHAih)
Incoming Calls
f Incoming References - AES_Encrypt_140007080
⊟ f AES_Encrypt_140007080
  ⊟ f EncryptAndRenameFiles_140007590
    ⊟ f EncryptAndRenameFiles_140007590
      ⊟ f RansomwareProcessor_140008240
        ⊟ f RansomwareProcessor_140008240
          ⊟ f RansomwareEntryPoint_14000afe0
              f FID_conflict:WinMainCRTStartup
            ⊟ f FID_conflict:WinMainCRTStartup
                f entry
            f FUN_14000b1d4
```

Figure 1: Function call trees

AES-CBC-128

2 Determine what files are targeted

EncryptAndRenameFiles_140007590

```
    do
    {
      /* Exclude Directory */
      if ((local_10e8.dwFileAttributes & 0x10) == 0)
      {
        CopyWPath_140007bc0(local_a68, 0x104, dir);
        input_addr = local_a68;
        ConcatWPath_140007b20(input_addr, 0x104, local_10e8.cFileName);
        GetModuleFileNameW((HMODULE)0x0, local_858, 0x104);
        thunk_FUN_14000c700((undefined8 *)local_e98,
                            (undefined8 *)local_10e8.cFileName, 6);
        /* Exclude "~en" */
        iVar2 = wcscmp(local_e98, L"~en");
        if (iVar2 != 0)
        {
          _Str2 = PathFindFileNameW(local_858);
          /* Exclude Malware Itelf */
          iVar2 = wcscmp(local_10e8.cFileName, _Str2);
          if (iVar2 != 0)
          {
            CopyWPath_140007bc0(local_648, 0x104, dir);
            output_addr = local_648;
            ConcatWPath_140007b20(output_addr, 0x104, (short *)&DAT_140070fd8);
            ConcatWPath_140007b20(output_addr, 0x104, local_10e8.cFileName);
            AES_Encrypt_140007080(input_addr, output_addr);
            DeleteFileW(input_addr);
          }
        }
      }
      BVar3 = FindNextFileW(local_1110, &local_10e8);
    } while (BVar3 != 0);
```

Listing 1: EncryptAndRenameFiles_140007590

RansomwareProcessor_140008240

```
void RansomwareProcessor_140008240(void)
{
    /*...*/
    WCHAR dir[264];
    /*...*/
    printf((char *)L"Getting current directory. ");
    GetCurrentDirectoryW(0x104, dir);
    EncryptAndRenameFiles_140007590(dir);
    Sleep(10000);
    /*...*/
}
```

Listing 2: RansomwareProcessor_140008240

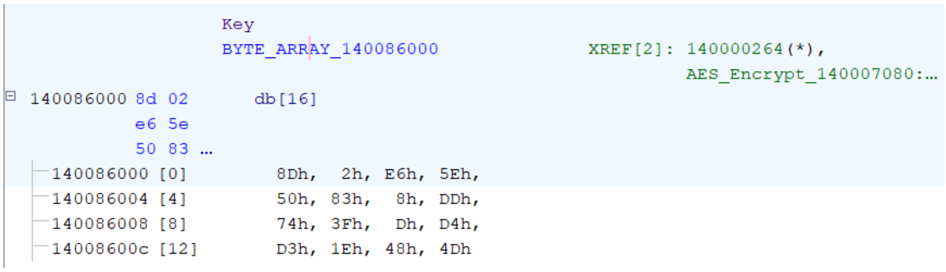
3 Recover the AES key

As noted above, the memory address of the AES key is the second parameter of the InitEncryption_140008790 function:

```
                                            /*Address of AES key*/
InitEncryption_140008790((longlong)context_array,0x140086000,(undefined8 *)
    IV_140086010);
```

Listing 3: call of InitEncryption_140008790

Then we took a screenshot of the key in Ghidra, as shown in Figure 2.
The AES key is ’8d02e65e508308dd743f0dd4d31e484d’.



```
                    Key
                    BYTE_ARRAY_140086000              XREF[2]: 140000264(*),
                                                               AES_Encrypt_140007080:…
⊟ 140086000 8d 02      db[16]
            e6 5e
            50 83 …
   ├─140086000 [0]        8Dh,  2h, E6h, 5Eh,
   ├─140086004 [4]        50h, 83h,  8h, DDh,
   ├─140086008 [8]        74h, 3Fh,  Dh, D4h,
   └─14008600c [12]       D3h, 1Eh, 48h, 4Dh
```

Figure 2: the AES key in Ghidra

4 Decrypt Hank’s files.

The tool to decrypt Hank’s files is ’assinment3-team6-data/AES_decrypt.py’.
There are two important functions in the program.

4.1 Decrypt a block

Only keep the actual plaintext length portion, the rest is meaningless padding used during encryption.

```python
def decrypt_block(ciphertext_block, key, iv, actual_plaintext_len):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_block = cipher.decrypt(ciphertext_block)
    return decrypted_block[:actual_plaintext_len]
```

Listing 4: decrypt_block

4.2 Decrypt the file

```python
def decrypt_file(input_path, output_path, key_hex):
    #...
    with open(input_path, 'rb') as f_in, open(output_path, 'wb') as f_out:
        while True:
            # Read the 16-byte IV
            iv = f_in.read(16)  #0a0b0c0d0e0fa0b0c0d0e0f0aabbccdd
            #...
            # Read the 4-byte actual plaintext length
            block_len_bytes = f_in.read(4)
            #...
            actual_plaintext_len = struct.unpack('<I', block_len_bytes)[0]
            # Read the encrypted 1008-byte block
            ciphertext_block = f_in.read(BLOCK_SIZE)
            #...
            # AES Decrypt
            plaintext_block = decrypt_block(ciphertext_block, key, iv, actual_plaintext_len)
            f_out.write(plaintext_block)
            #...
```

Listing 5: decrypt_file

To use this python script, please install pycryptodome firstly.

```
pip3 install pycryptodome
```

Then replace the following line with YOUR directory of the files to be decrypted, and DO NOT add a '/' to the end of your directory.

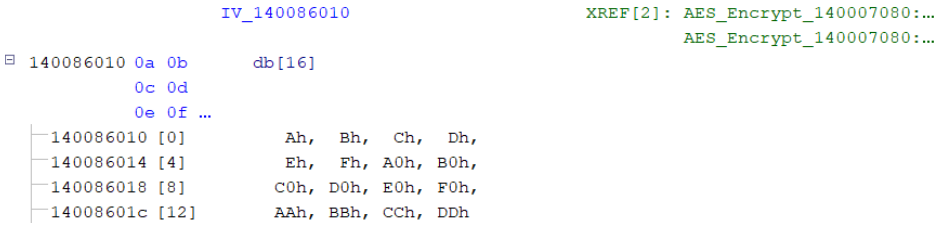
```
FILE_DIRECTORY = "HanksBackup"
```

Academic Conduct & Plagiarism:

We take plagiarism seriously. By submitting your solution, you agree that:
1. The submission is your group's own work and that you have not worked with others in preparing this assignment.
2. Your submitted solutions and report were written by you and **in your own words**, except for any materials from published or other sources which are clearly indicated and acknowledged as such by appropriate referencing.
3. The work is not copied from any other person's work (published or unpublished), web site, book or other source, and has not previously been submitted for assessment either at the University of Birmingham or elsewhere.
4. You have not asked, or paid, others to prepare any part of this work.

A Ghidra screenshots

Screenshot of IV in Ghidra is shown in Figure 3. IV is '0a0b0c0d0e0fa0b0c0d0e0f0aabbccdd'.



```
                     IV_140086010                      XREF[2]: AES_Encrypt_140007080:…
                                                                AES_Encrypt_140007080:…
⊟ 140086010 0a 0b      db[16]
            0c 0d
            0e 0f …
   140086010 [0]          Ah,  Bh,  Ch,  Dh,
   140086014 [4]          Eh,  Fh, A0h, B0h,
   140086018 [8]         C0h, D0h, E0h, F0h,
   14008601c [12]        AAh, BBh, CCh, DDh
```

Figure 3: IV in Ghidra

B C-style decompiled codes

All the C-style decompiled codes mentioned above can be found in the directory 'assinment3-team6-data/C-style decompiled code'.

In addition, there are some functions not mentioned above, but which are also valuable (because they are part of the function call tree), listed below:

1. entry.c

```c
void entry(void)
{
    __security_init_cookie();
    RansomwareEntryPoint_14000afe0();
    return;
}
```

Listing 6: entry.c

2. RansomwareEntryPoint.c

```c
ulonglong RansomwareEntryPoint_14000afe0(void)
{
    /*...*/
            __scrt_get_show_window_mode();
            _get_wide_winmain_command_line();
            /*Ransomware Processor here*/
            uVar3 = RansomwareProcessor();
            uVar7 = __scrt_is_managed_app();
    /*...*/
}
```

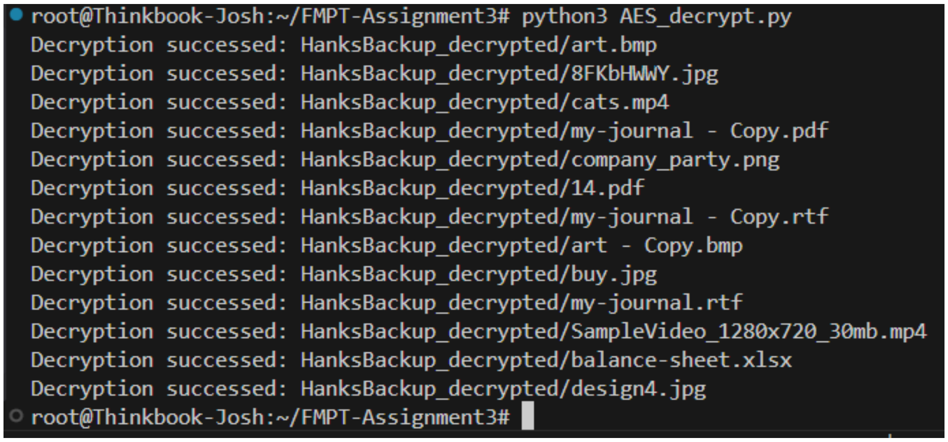
Listing 7: RansomwareEntryPoint_14000afe0.c

C Decrypted files

Decrypted files can be found at:
`https://github.com/Superior-Josh/FMPT-Assignment3/tree/main/HanksBackup_decrypted`

C.1 Screenshot of the output

The successful output of the decryption tool is shown in Figure 4.



```
root@Thinkbook-Josh:~/FMPT-Assignment3# python3 AES_decrypt.py
Decryption successed: HanksBackup_decrypted/art.bmp
Decryption successed: HanksBackup_decrypted/8FKbHWWY.jpg
Decryption successed: HanksBackup_decrypted/cats.mp4
Decryption successed: HanksBackup_decrypted/my-journal - Copy.pdf
Decryption successed: HanksBackup_decrypted/company_party.png
Decryption successed: HanksBackup_decrypted/14.pdf
Decryption successed: HanksBackup_decrypted/my-journal - Copy.rtf
Decryption successed: HanksBackup_decrypted/art - Copy.bmp
Decryption successed: HanksBackup_decrypted/buy.jpg
Decryption successed: HanksBackup_decrypted/my-journal.rtf
Decryption successed: HanksBackup_decrypted/SampleVideo_1280x720_30mb.mp4
Decryption successed: HanksBackup_decrypted/balance-sheet.xlsx
Decryption successed: HanksBackup_decrypted/design4.jpg
root@Thinkbook-Josh:~/FMPT-Assignment3#
```

Figure 4: Decryption tool output

A decrypted example (SampleVideo_1280×720_30mb.mp4) is shown in Figure 5.

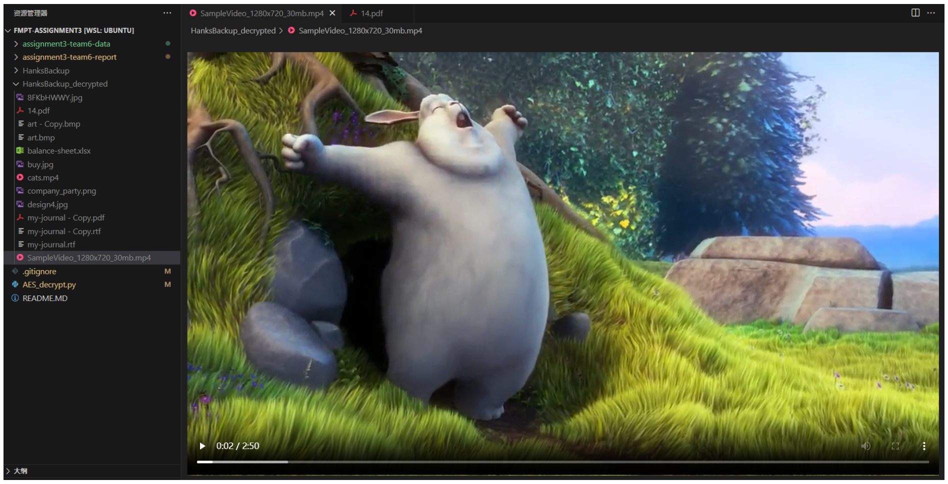


Figure 5: Decrypted file example