CMSC 491/691 Malware Analysis HW 4

Name: Daniel Roh
Assigned: 3/2/2020
Due: 3/11/2020 by 5:00pm
How to turn this in for grading: You can edit your answers right into this file. Submit the homework a .docx or .pdf to RJ via email.

Download and extract hw4.7z onto your Flare VM. Use Ghidra to answer the following questions:

1) **How many DLL files does the hw4 import functions from? (5 pts)**
   HW4 imports 3 dll files

2) **What is the address of the string kbdax2.dll? (5 pts)**
   Address: 1002e344

3) **What is the address of the ServiceMain export? (5 pts)**
   Address: 100018b0

4) **How many parameters does the function FUN_10003060 take? How many local variables does it have? (8 pts)**
   The function FUN_10003060 takes 2 variables as input (char * and char).
   The function FUN_10003060 has 1 local variable (char).

5) **How many times is WriteFile called? (5 pts)**
   WriteFile is called 3 times
   __Note to TA:__ if the answer to this is 4 then I discounted the KERNEL32.dll call as a countable call due to it being called in a library rather than in the malware.

6) **In what function is GetEnvironmentVariableA called? (5 pts)**
   GetEnvironmentVariableA is called in the function FUN_10000a350

7) **In a few sentences, describe what FUN_100081e4 does. What is the name of the technique that this function performs? (12 pts)**
   - Function FUN_100081e4 attempts to get a handle to the KERNEL32.dll then runs a check to see if the handle to KERNEL32.dll is working.
   - This function is doing run time linking technique

8) **Describe in detail what occurs from 0x100018DC to 0x10001937. (15 pts)**
   From 0x100018DC to 0x10001937, the program is loading a string a character onto the stack one at a time to avoid detection from people trying to see what that program is trying to do. Basically, the program is making a stack string.

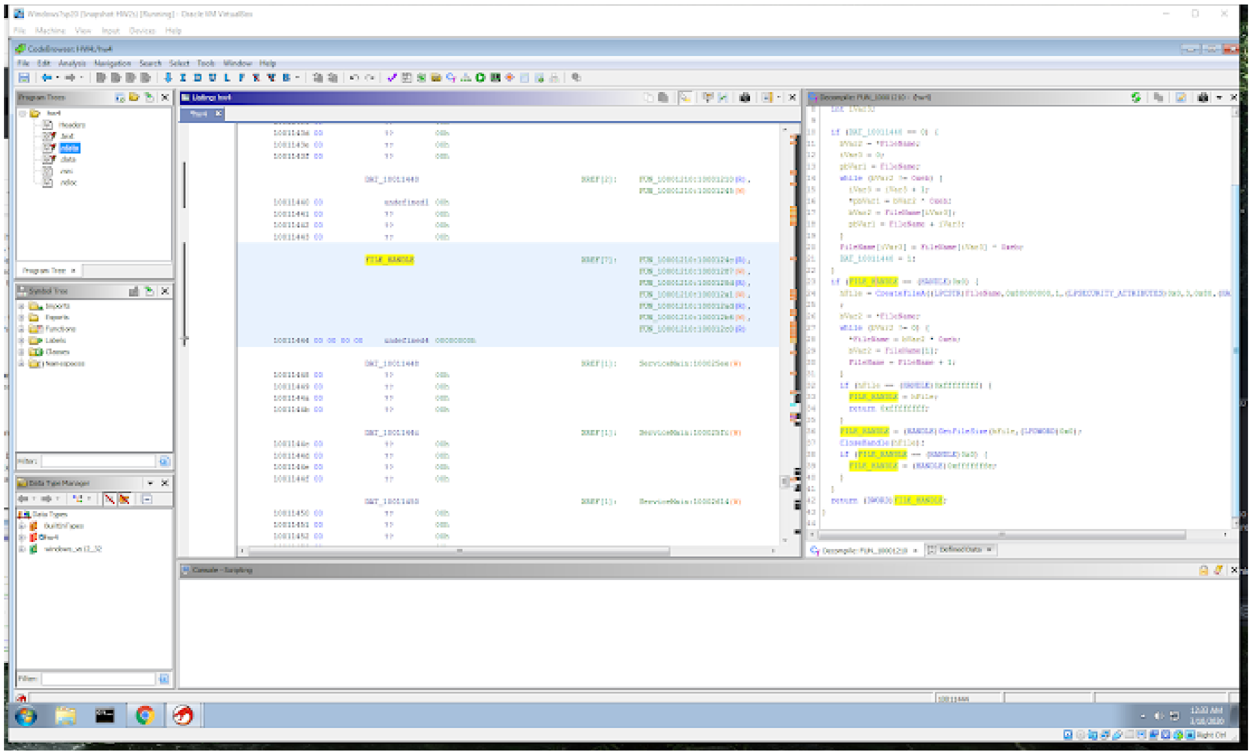
9) **Describe in detail what the code between 0x10001210 and 0x1000123B does. (15 pts)**
   The code between 0x10001210 - 0x1000123B is a function that takes a encoded file name and a

hex value as input. First, the functions start by checking if a file has been made since the program has been run. In the case the file is not made, the code takes the file name and decodes it with the hex value 0x4b. Then the code checks to see if the runtime linking handle was successful. In the case it was not successful, it creates a handle to CreateFileA with the decoded file name. After creating the file handle, it continues by taking the decoded file name and re-encodes it by multiplying it by 0xeb. Then the code checks to see if the handle is invalid and if so returns the handle. If not then the size of the newly created file is checked to see if it's null. In the case its null, it returns a handle, else it returns a handle to the size of the new file.

Basically, this code is checking to see if a file has been created and in the case the file was not created, create a file and then checking to see if the file is properly made. If the file is already made, then the code returns the file handler.

10) **The function FUN_10001580 calls FUN_10001210 at 0x100015DA. Using your analysis from question 9, what will be the name of the file created when execution reaches the call to CreateFileA at 0x10001268? (15 pts)**
File name: PlayToManager.dll

11) **Rename DAT_10011444 to something more descriptive. Justify the variable name you chose and provide a screenshot (10 pts).**
The reason I renamed DAT_1001144 to FILE_HANDLE is due to FILE_HANDLE being used as a global handle for file operations. In the case of this code, once the file is created. The file is referenced by this file's handler which is DAT_1001144. So it in my mind, it makes sense to call it a FILE_HANDLE



```
  int iVar3;

  if (DAT_10011448 == 0) {
    bVar2 = *FileName;
    iVar3 = 0;
    pbVar1 = FileName;
    while (bVar2 != 0xeb) {
      iVar3 = iVar3 + 1;
      *pbVar1 = bVar2 ^ 0xeb;
      bVar2 = FileName[iVar3];
      pbVar1 = FileName + iVar3;
    }
    FileName[iVar3] = FileName[iVar3] ^ 0xeb;
    DAT_10011448 = 1;
  }
  if (FILE_HANDLE == (HANDLE)0x0) {
    hFile = CreateFileA((LPCSTR)FileName,0x80000000,1,(LPSECURITY_ATTRIBUTES)0x0,3,0x80,0x0
    ;
    bVar2 = *FileName;
    while (bVar2 != 0) {
      *FileName = bVar2 ^ 0xeb;
      bVar2 = FileName[1];
      FileName = FileName + 1;
    }
    if (hFile == (HANDLE)0xffffffff) {
      FILE_HANDLE = hFile;
      return 0xffffffff;
    }
    FILE_HANDLE = (HANDLE)GetFileSize(hFile,(LPDWORD)0x0);
    CloseHandle(hFile);
    if (FILE_HANDLE == (HANDLE)0x0) {
      FILE_HANDLE = (HANDLE)0xffffffff;
    }
  }
  return (DWORD)FILE_HANDLE;
}
```