[5] Trace function arguments of the following Windows APIs.

- API로 **CreateFlieW, WriteFile, CreateProcessA**가 주어졌다.
- 해당 API들의 arguments들을 모두 출력해야 한다.

1. Frida 설치

- 한 두시간 넘게 걸렸다. 결국 Flare-vm이 아니라 본체에 설치했다.

2. ku-re.py

```
import frida
import sys

def main(target_process, script_text):
    print("[*] Inspecting file operations")
    session = frida.attach(target_process)

    script = session.create_script(script_text)
    script.load()
    print("[*] Wating...")
    sys.stdin.read()

if __name__ == '__main__':
    try:
        target_process = int(sys.argv[1])
    except ValueError:
        target_process = sys.argv[1]
    script_text = ''
    with open(sys.argv[2], 'r') as fd:
        script_text += fd.read() + '\n'
    main(target_process, script_text)
```

3. 후킹 시도

- 강의자료에 있는 코드들을 적극적으로 활용했다.
- KU-RE{FinaL}.exe의 PID를 알아내려고 frida-ps를 쳤는데 나오지가 않았다.
- 다시 알아보니 프로그램을 실행시키고 있어야지만 찾아지는 거였다.

```
> frida-ps
  PID  Name
-----  --------------------------

10816  KU-RE{FinaL}.exe
```

- 처음에 일부분만 가지고 테스트를 했을 때, ke-re.py 부분에서 IndentationError: unindent does not match any outer indentation level 에러가 떴었는데, 이 오류는 탭,띄어쓰기 등으로 인한 오

류였다. vscode로 작성하던 거를 노션으로 옮겨보니까 탭이 잘못 되어있는 것을 알게 됐고, 올바르게 하니까 실행이 됐다.

- 하지만 또, ppt에 나온 그대로 ku-re.py를 사용하니, API 실행 과정은 나오지도 않고 종료가 됐다.
  - 그래서 찾아보니까 ku-re.py에 아래 코드를 추가하면 되는 거였다.

```
print("[*] Wating...")
sys.stdin.read()
```

4. inspect-file.js 생성

- 각각의 argument들의 형식을 고려해서 출력하게 만든다.
- 문자는 문자로, 숫자는 숫자로, 포인터는 가리키는 버퍼를 출력하게 만든다(만약 포인터가 NULL이면 가리키는 곳이 없으므로 NULL을 출력하게 한다.). Hex dump 출력하는 게 정말 유용한 것 같다!

4.1. inspect-file.js : CreateFileW

```
var createFile = Module.findExportByName("kernel32.dll", "CreateFileW");

Interceptor.attach(createFile, {
    onEnter: function(args)
    {
    console.log("");
    console.log("==== CreateFile's lpFileName ====");
    console.log(Memory.readUtf16String(args[0]));

    console.log("==== CreateFile's dwDesiredAccess ====");
    console.log(args[1]);

    console.log("==== CreateFile's dwShareMode ====");
    console.log(args[2]);

    console.log("==== CreateFile's lpSecurityDisposition ====");
    if (args[3] == 0)
    { console.log("NULL"); }
    else
    { console.log("Pointer: " + args[3]);
      console.log(hexdump(args[3], { length: 128, ansi: true})); }

    console.log("==== CreateFile's dwCreationDisposition ====");
    console.log(args[4]);

    console.log("==== CreateFile's dwFlagAndAttribution ====");
    console.log(args[5]);

    console.log("==== CreateFile's hTemplateFile ====");
    console.log(args[6]);

    console.log("");
    }
});
```

- `lpFileName`: 이름을 readUtf16String으로 출력한다.
- `dwDesiredAccess`, `dwShareMode`, `dwCreationDisposition`, `dwFlagAndAttribution`, `hTemplateFile`: 해당 값들은 그냥 출력한다.
- `lpSecurityDisposition`: 포인터가 가리키는 곳의 Hexdump를 출력한다. NULL도 가능하기 때문에 NULL도 출력 가능하게 한다.

4.2. inspect-file.js : WriteFile

```
var writeFile = Module.findExportByName("kernel32.dll", "WriteFile");

Interceptor.attach(writeFile, {
    onEnter:function(args){
        console.log("")

        console.log("==== WriteFile's hFile ====");
        console.log(args[0]);

        console.log("==== WriteFile's lpBuffer ====");
        console.log("Pointer: " + args[1]);
        console.log(hexdump(args[1], { length: 128, ansi: true}));

        console.log("==== WriteFile's nNumberOfBytesToWrite ====");
        console.log(args[2]);

        console.log("==== WriteFile's lpNumberOfBytesWritten ====");
        if(args[3] == 0)
        { console.log("NULL"); }
        else
        { console.log("Pointer: " + args[3]);
          console.log(hexdump(args[3], { length: 128, ansi: true})); }

        console.log("==== WriteFile's lpOverlapped ====");
        if(args[4] == 0)
        { console.log("NULL") }
        else
        { console.log("Pointer: " + args[4]);
          console.log(hexdump(args[4], { length: 128, ansi: true})); }

        console.log("");
    }
})
```

- `hFile`, `nNumberOfBytesToWrite`, `lpNumberOfBytesWritten`: 해당 값들은 그냥 출력한다.
- `lpBuffer`, `lpOverlapped`: 포인터가 가리키는 Hexdump를 출력한다. lpOverlapped는 NULL도 가능하기 때문에 NULL도 출력 가능하게 한다.

4.3. inspect-file.js : CreateProcessA

```
var createProcess = Module.findExportByName("kernel32.dll", "CreateProcessA");
Interceptor.attach(createProcess, {
    onEnter: function(args){
        console.log("");
```

```
        console.log("==== CreateProcess's lpApplicationName ====");
        console.log(Memory.readUtf16String(args[0]));

        console.log("==== CreateProcess's lpCommandLine ====");
        console.log(Memory.readCString(args[1]));

        console.log("==== CreateProcess's lpProcessAttributes ====");
        if (args[2] == 0)
        { console.log("NULL"); }
        else
        { console.log("Pointer: " + args[2]);
          console.log(hexdump(args[2], { length: 128, ansi: true}));}

        console.log("==== CreateProcess's lpThreadAttributes ====");
        if (args[3] == 0)
        { console.log("NULL"); }
        else
        { console.log("Pointer: " + args[3]);
          console.log(hexdump(args[3], { length: 128, ansi: true})); }

        console.log("==== CreateProcess's bInheritHandles ====");
        if (args[4])
        { console.log("TRUE"); }
        else
        { console.log("FALSE"); }

        console.log("==== CreateProcess's dwCreationFlags ====");
        console.log(args[5]);

        console.log("==== CreateProcess's lpEnvironment ====");
        if (args[6] == 0)
        { console.log("NULL"); }
        else
        { console.log("Pointer: " + args[6]);
          console.log(hexdump(args[6], { length: 128, ansi: true})); }

        console.log("==== CreateProcess's lpCurrentDirectory ====");
        console.log(Memory.readUtf16String(args[7]));

        console.log("==== CreateProcess's lpStartupInfo ====");
        console.log("Pointer: " + args[8]);
        console.log(hexdump(args[8], { length: 128, ansi: true}));

        console.log("==== CreateProcess's lpProcessInformation ====");
        console.log("Pointer: " + args[9]);
        console.log(hexdump(args[9], { length: 128, ansi: true}));

        console.log("");
        }
})
```

- `lpApplicationName`, `lpCurrentDirectory`: readUtf16String으로 출력한다.
- `lpCommandLine`: readCString으로 출력한다.
- `lpProcessAttributes`, `lpThreadAttributes`, `lpEnvironment`, `lpProcessInformation`: 포인터가 가리키는 Hexdump를 출력한다.
- `bInheritHandles`: True/False를 출력한다.
- `dwCreationFlags`: 그냥 값을 출력한다.

결과

- 파일을 켜놓은 다음에 frida-ps로 PID를 알아낸다.

```
> frida-ps
  PID  Name
-----  ------------------------------
17840  KU-RE{FinaL}.exe
```

- 만들어놓은 ku-re.py, inspect-file.js를 이용해서 inspecting한다.
  - 명령어를 쳐놓고, 이름값을 복사하고, 복사한 값을 이름에 넣고, 학번과 플래그를 입력한 다음에 버튼을 누르면 해당 API들의 활동내역들이 inspecting된다.
- cmd창에 출력된 순서대로 argument 정보를 해석해보겠다.

1. **CreateFile**

```
> python ku-re.py 17840 inspect-file.js
[*] Inspecting file operations
[*] Wating...

==== CreateFile's lpFileName ====
C:\Users\kangj\바탕 화면\KU-RE{FinaL}.exe
==== CreateFile's dwDesiredAccess ====
0x80000000
==== CreateFile's dwShareMode ====
0x1
==== CreateFile's lpSecurityDisposition ====
NULL
==== CreateFile's dwCreationDisposition ====
0x3
==== CreateFile's dwFlagAndAttribution ====
0x80
==== CreateFile's hTemplateFile ====
0x0
```

- **lpFileName**: `C:\Users\kangj\바탕 화면\KU-RE{FinaL}.exe`
- **dwDesiredAccess**: GENERIC_WRITE(`0x80000000`). 파일에 쓰기 권한을 가진다.
- **dwShareMode**: FILE_SHARE_READ(`0x1`). 다른 프로세스의 열기 권한을 허가한다.
- **lpSecurityDisposition**: SECURITY_ATTRIBUTES를 사용하지 않는다.
- **dwCreationDisposition**: OPEN_EXISTING(`0x3`). 파일이 존재할 경우에만 파일을 연다.
- **dwFlagAndAttribution**: FILE_ATTRIBUTE_NORMAL(`0x80`). 파일에 다른 특성 집합이 없다.
- **hTemplateFile**: GENERIC_READ(`0x0`). 액세스 권한이 있는 템플릿 파일에 대한 유효한 핸들이 없다.

2. **CreateProcess**

```
==== CreateProcess's lpApplicationName ====
null
==== CreateProcess's lpCommandLine ====
PowerShell.exe -noexit -enc TgBlAHcALQBJAHQAZQBtACAALQBGAG8AcgBjAGUAIAAnAEMAOgBc
AFUAcwBlAHIAcwBcAGsAYQBuAGcAagBcAEEAcABwAEQAYQB0AGEAXABMAG8AYwBhAGwAXABUAGUAbQBw
AFwASwBVAC0AUgBFAHsAawAyAHkAfQAnADsAIABTAGUAdAAtAEMAbwBuAHQAZQBuAHQAIAAnAEMAOgBc
AFUAcwBlAHIAcwBcAGsAYQBuAGcAagBcAEEAcABwAEQAYQB0AGEAXABMAG8AYwBhAGwAXABUAGUAbQBw
AFwASwBVAC0AUgBFAHsAawAyAHkAfQAnACAAJwA0AEQANQBBADkAMAAwADAAMAAzADAAMAAwADAAMAAw
ADAANAAwADAAMAAwADAAMABGAEYARgBGADAAMAAwADAAQgA4ADAAMAAwADAAMAAwADAAMAAwADAAMAAw
ADAAMAA0ADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAw
ADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAw
ADAAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAxADAAMAAwADAAMABFADEARgBCAEEAMABF
ADAAMABCADQAMAA5AEMARAAyADEAQgA4ADAAMQA0AEMAQwBEADIAMQA1ADQANgA4ADYAOQA3ADMAMgAw
ADcAMAA3ADIANgBGADYANwA3ADIANgAxADYARAAyADAANgAzADYAMQA2AEUANgBFADYARgA3ADQAMgAw
ADYAMgA2ADUAMgAwADcAMgA3ADUANgBFADIAMAA2ADkANgBFADIAMAA0ADQANABGADUAMwAyADAANgBE
ADYARgA2ADQANgA1ADIARQAwAEQAMABEADAAQQAyADQAMAAwADAAMAAwADAAMAAwADAAMAAwADAAMAAw
ACcA
==== CreateProcess's lpProcessAttributes ====
NULL
==== CreateProcess's lpThreadAttributes ====
NULL
==== CreateProcess's bInheritHandles ====
TRUE
==== CreateProcess's dwCreationFlags ====
0x8000000
==== CreateProcess's lpEnvironment ====
NULL
==== CreateProcess's lpCurrentDirectory ====
null
==== CreateProcess's lpStartupInfo ====
Pointer: 0xd1477febe0
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
d1477febe0  68 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  h...............
d1477febf0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec50  50 6f 77 65 72 53 68 65 6c 6c 2e 65 78 65 20 2d  PowerShell.exe -
==== CreateProcess's lpProcessInformation ====
Pointer: 0xd1477febc0
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
d1477febc0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477febd0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477febe0  68 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  h...............
d1477febf0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477fec30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

- **lpApplicationName**: NULL
- **lpCommandLine**: powershell.exe에서 실행할 명령줄이다. 암호화를 하려는것 같다.
- **lpProcessAttributes**: 새 프로세스 개체에 반환된 핸들을 자식 프로세스에서 상속하지 않는다.
- **lpThreadAttributes**: 새 스레드 개체에 반환된 핸들을 자식 프로세스에서 상속하지 않는다.
- **bInheritHandles**: 호출 프로세스의 상속 가능한 각 핸들이 새 프로세스에서 상속된다.

- **dwCreationFlags**: CREATE_NO_WINDOW(`0x80000000`). 이 프로세스는 콘솔 창 없이 실행되는 콘솔 애플리케이션이다.
- **lpEnvironment**: 새 프로세스는 호출 프로세스의 환경을 사용한다.
- **lpCurrentDirectory**: 새 프로세스는 호출 프로세스와 동일한 현재 드라이브 및 디렉터리를 갖는다.
- **lpStartupInfo**: STARTRUPINFO or STARTUPINFOEX에 대한 구조체에 대한 포인터이다. 해당 구조체를 출력하고 있다.
- **lpProcessInformation**: 새 프로세스에 대한 식별 정보를 수신하는 PROCESS_INFORMATION 구조체에 대한 포인터이다. 해당 구조체를 출력하고 있다.

3. **CreateFile**

```
==== CreateFile's lpFileName ====
C:\Users\kangj\AppData\Local\Temp\KU-RE{k2y}
==== CreateFile's dwDesiredAccess ====
0x80000000
==== CreateFile's dwShareMode ====
0x1
==== CreateFile's lpSecurityDisposition ====
NULL
==== CreateFile's dwCreationDisposition ====
0x4149444100000003
==== CreateFile's dwFlagAndAttribution ====
0x4151444100000080
==== CreateFile's hTemplateFile ====
0x0
```

- **lpFileName**: `C:\Users\kangj\AppData\Local\Temp\KU-RE{k2y}`
- **dwDesiredAccess**: GENERIC_WRITE(`0x80000000`). 파일에 쓰기 권한을 가진다.
- **dwShareMode**: FILE_SHARE_READ(`0x1`). 다른 프로세스의 열기 권한을 허가한다.
- **lpSecurityDisposition**: SECURITY_ATTRIBUTES를 사용하지 않는다.
- **dwCreationDisposition**: OPEN_EXISTING(`0x3`). 파일 또는 디바이스가 있는 경우에만 연다. 지정된 파일 또는 디바이스가 없으면 함수가 실패하고 마지막 오류 코드가 ERROR_FILE_NOT_FOUND(2)로 설정된다.
- **dwFlagAndAttribution**: FILE_ATTRIBUTE_NORMAL(`0x80`). 파일에 다른 특성 집합이 없다.
- **hTemplateFile**: GENERIC_READ(`0x0`). 액세스 권한이 있는 템플릿 파일에 대한 유효한 핸들이 없다.

4. **CreateFile**

```
==== CreateFile's lpFileName ====
\\?\C:\Users\kangj\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-4272752103-685001032
-1152607559-1001\053d0b1b4ff31678498743d14020f5c6_a3f6ed4c-2c03-45ee-96e4-40d4e3bb8392
==== CreateFile's dwDesiredAccess ====
0x80000000
```

```
==== CreateFile's dwShareMode ====
0x1
==== CreateFile's lpSecurityDisposition ====
NULL
==== CreateFile's dwCreationDisposition ====
0x3
==== CreateFile's dwFlagAndAttribution ====
0x27808000000
==== CreateFile's hTemplateFile ====
0x0
```

- **lpFileName**: `\\?\C:\Users\kangj\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-4272752103-685001032-1152607559-1001\053d0b1b4ff31678498743d14020f5c6_a3f6ed4c-2c03-45ee-96e4-40d4e3bb8392`
- **dwDesiredAccess**: GENERIC_WRITE(`0x80000000`). 파일에 쓰기 권한을 가진다.
- **dwShareMode**: FILE_SHARE_READ(`0x1`). 다른 프로세스의 열기 권한을 허가한다.
- **lpSecurityDisposition**: SECURITY_ATTRIBUTES를 사용하지 않는다.
- **dwCreationDisposition**: OPEN_EXISTING(`0x3`). 파일이 존재할 경우에만 파일을 연다.
- **dwFlagAndAttribution**: FILE_FLAG_SEQUENTIAL_SCAN(`0x8000000`). 액세스는 처음부터 끝까지 순차적으로 사용할 수 있다.
- **hTemplateFile**: GENERIC_READ(`0x0`). 액세스 권한이 있는 템플릿 파일에 대한 유효한 핸들이 없다.

5. **CreateFile**

```
==== CreateFile's lpFileName ====
KU-DFRC-ReverseMe2-Happy-New-Year.hdn
==== CreateFile's dwDesiredAccess ====
0x40000000
==== CreateFile's dwShareMode ====
0x0
==== CreateFile's lpSecurityDisposition ====
NULL
==== CreateFile's dwCreationDisposition ====
0x52002d00000002
==== CreateFile's dwFlagAndAttribution ====
0xc775c72000000002
==== CreateFile's hTemplateFile ====
0x0
```

- **lpFileName**: `KU-DFRC-ReverseMe2-Happy-New-Year.hdn`
- **dwDesiredAccess**: GENERIC_READ(`0x40000000`). 파일에 읽기 권한을 가진다.
- **dwShareMode**: `0x0`. 모든 프로세스의 접근을 차단한다. 다른 프로세스가 접근을 시도할 경우 엑세스가 거부된다.
- **lpSecurityDisposition**: SECURITY_ATTRIBUTES를 사용하지 않는다.
- **dwCreationDisposition**: CREATE_ALWAYS(`0x2`). 항상 새 파일을 만든다.

- **dwFlagAndAttribution**: FILE_ATTRIBUTE_HIDDEN(`0x2`). 파일이 숨겨져 있다.
- **hTemplateFile**: GENERIC_READ(`0x0`). 액세스 권한이 있는 템플릿 파일에 대한 유효한 핸들이 없다.

6. **WriteFile**

```
==== WriteFile's hFile ====
0x50c
==== WriteFile's lpBuffer ====
Pointer: 0xd1477ff7e0
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
d1477ff7e0  31 00 30 00 2e 00 30 00 2e 00 31 00 39 00 30 00  1.0...0...1.9.0.
d1477ff7f0  34 00 33 00 0a 00 44 00 45 00 53 00 4b 00 54 00  4.3...D.E.S.K.T.
d1477ff800  4f 00 50 00 2d 00 41 00 39 00 48 00 52 00 4f 00  O.P.-.A.9.H.R.O.
d1477ff810  41 00 39 00 00 00 00 00 00 00 00 00 00 00 00 00  A.9.............
d1477ff820  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477ff830  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477ff840  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d1477ff850  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
==== WriteFile's nNumberOfBytesToWrite ====
0x34
==== WriteFile's lpNumberOfBytesWritten ====
Pointer: 0xd1477ff708
             0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
d1477ff708  00 00 00 00 00 00 00 00 4b 00 55 00 2d 00 44 00  ........K.U.-.D.
d1477ff718  46 00 52 00 43 00 2d 00 52 00 65 00 76 00 65 00  F.R.C.-.R.e.v.e.
d1477ff728  72 00 73 00 65 00 4d 00 65 00 32 00 2d 00 48 00  r.s.e.M.e.2.-.H.
d1477ff738  61 00 70 00 70 00 79 00 2d 00 4e 00 65 00 77 00  a.p.p.y.-.N.e.w.
d1477ff748  2d 00 59 00 65 00 61 00 72 00 2e 00 68 00 64 00  -.Y.e.a.r...h.d.
d1477ff758  6e 00 00 00 30 30 30 30 44 00 45 00 53 00 4b 00  n...0000D.E.S.K.
d1477ff768  54 00 4f 00 50 00 2d 00 41 00 39 00 48 00 52 00  T.O.P.-.A.9.H.R.
d1477ff778  4f 00 41 00 39 00 00 00 00 00 00 00 00 00 00 00  O.A.9...........
==== WriteFile's lpOverlapped ====
NULL
```

- **hFile**: 파일 또는 I/O 디바이스에 대한 핸들(`0x50c`)
- **lpBuffer**: 파일에 쓸 데이터를 가지는 버퍼에 대한 포인터. 해당 버퍼를 출력하고 있다.
- **nNumberOfBytesToWrite**: 파일 또는 디바이스에 쓸 바이트 수(`0x34`)
- **lpNumberOfBytesWritten**: 파일에 실제 저장된 데이터 크기를 얻기 위해 변수의 주소를 지정한다. 해당 포인터가 가리키는 곳을 출력했다.
- **lpOverlapped**: OVERLAPPED 구조체에 대한 포인터가 없다.

이렇게 6가지의 후킹 결과를 해석해봤다.