Information leakage by USB

Track	Digital Forensic
Category	Tech_05
Nick Name	L3ad0xFF

목 차

1. 개발 목적 및 목표

1.1 개발 목적

- USB를 이용하여 정보유출이 되었다고 판단 될 경우 유출된 내역을 분석하기 위함이다.
- Script를 이용하여 활성시스템에서 정보가 유출되었다고 판단할 근거를 찾는다.
- 분석 대상 PC에 연결된 USB와 연관 있는 정보들을 확인한다.
- Setupapi.dev.log 파일을 이용하여 분석 PC에서 확인한 USB들의 최초 연결 기록을 확인한다.
- Script를 이용한 분석 결과를 csv 파일 형식으로 기록하여, 분석자에게 제공한다.

1.2 개발 조건 및 목표.

- USB 형식의 저장소가 연결되어 사용 되었을 경우 분석 대상 PC의 레지스트리에 접근하여 정보를 확인한다.
- 분석 대상 PC의 EventLog를 분석하여 USB 연결 및 해제 흔적을 확인한다.
- USB 저장소에서 파일을 실행한 흔적 분석을 위해 바로가기 파일(Linkfile)을 분석한다.
- USB 저장소에서 파일을 실행한 흔적 분석을 위해 레지스트리(Shellbag)을 분석한다.
- 모든 분석 결과를 CSV 파일 형식으로 저장하여 분석자에게 제공한다.

2. 개발 진행 시 특이점

2.1 Winreg Module 사용

- Window Registry에 접근하여 특정 레지스트리의 경로를 입력하여 접근 시 사용한다.
- 지정 경로의 하위 키, 지정한 키의 값들을 조회하여 변수에 저장이 가능하다.

2.2 Platform Module 사용

- 분석 대상 PC의 운영체제 정보를 확인 할 때 사용한다.

2.3 pyevtx Module 사용

- 분석 대상 PC의 EventLog evtx 파일 접근하여 출력이 가능하도록 한다.
- evtx 파일을 xml 형식의 데이터 스트림으로 변환 저장이 가능하다.

3. 개발 Script_1 (USB_infoleak.py)

- 분석 대상 PC의 대상 레지스트리에 접근하여 USB 관련 정보를 추출한다
- 외부 저장기기 연결 및 해제 시 기록하는 Window EventLog를 분석하여, USB 형식의 이동식 저장소의 연결 및 해제 기록을 추출한다.
- Setupapi.dev.log 파일을 이용하여 분석 PC에 연결 되었던 외부 저장소의 최초 연결 기록을 확인한다.
- 레지스트리 분석과 EventLog 분석, Setupali.dev.log 결과를 USB 종류별로 분류하여 하나의 CSV 파일로 생성한다.
- 분석 PC내에서 USB 연결 후 USB에 있는 파일을 실행 하였을 경우 생성되는 바로가기(Linkfile)을 분석하여 실행 된 파일의 USB와 종류와 드라이브 볼륨 명을 확인한다.

3.1 Main function - Initiate

```
if __name__ == '__main__' :
    global os_release, controlSetinfo
    print ("OS System Info : ", end=" "); print (platform.platform())
    os_release = platform.release()
    controlSetinfo = list()
    D_classes()
    for i in range (0, len(d_class_id)) :
        controlSetinfo.append(d_class_id[i].split(" : ")[0])
        controlSetinfo = list(set(controlSetinfo))
    D_classes_1()
    V_id_P_id()
    mountDevice()
    Volume_Deivce()
    VolumeName()
    setupdevlog()
    regi2dic()
    Eventlogload()
    linkfile()
```

- Platform Module을 이용하여, 분석 대상 PC의 운영체제 버전을 확인한다.
  - 분석 대상의 운영체제의 버전에 따라 분석할 EventLog의 구조가 다르기 때문이다.
- 사용 함수들을 분석 대상 PC에 저장된 모든 USB의 물리 장치 이름, USB 시리얼 넘버, 분석 대상 PC에 저장된 USB 장치 명, setupapi.dev.log를 이용한 최초 접속 이력, 각 USB 연결 시 등록된 볼륨 드라이브 명, 제조사 명칭, 설정된 볼륨 명, 연결 관련 이벤트로그 분석, 레지스트리 분석과 이벤트로그 분석 결과를 분류한 CSV 파일 생성, 링크파일 분석 후 CSV 파일 생성 순으로 순차적인 함수 사용이다
- 추가적으로 ControlSet00#이 여려 개일 경우 각 번호로 분류가 가능하도록 설정하였다..

3.2 Definition Function D_classes

```
def D_classes() :
    global d_class_id, d_unique_instance_id, search4log
    varSubkey = "SYSTEM" # 서브레지스트리 목록 지정
    varReg = ConnectRegistry(None, HKEY_LOCAL_MACHINE) # 루트 레지스트리 핸들 객체 얻기
    varinitKey = OpenKey(varReg, varSubkey) # 레지스트리 핸들 객체 얻기

    deviceclass = ['{10497b1b-ba51-44e5-8318-a65c837b6661}', '{53f56307-b6bf-11d0-94f2-00a0c91efb8b}',
    '{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}', '{6ac27878-a6fa-4155-ba85-f98f491d4f33}']
    d_class_id = list()
    d_unique_instance_id = list()
    search4log = list()
```

- 최초 분석 대상 PC의 ContorlSet의 경우를 확인하기 위해 HKLM/SYSTEM 레지스트리 키에 접근한다.
- USB의 경우 저장되는 하위 키의 이름이 지정되어 있기 때문에, 해당 리스트를 생성한다.

```
for z in range(1024) :
    try :
        searchKey = OpenKey(varReg,varSubkey)
        searchKeyname = EnumKey(varinitKey, z)
        for num in range(1000) :
            if len(str(num)) == 1 :
                number = "00"+str(num)
            elif len(str(num)) == 2 :
                number = "0"+str(num)
            else :
                number = str(num)

            if searchKeyname in ("ControlSet"+number) :
                firstsubKey = "%s\\%s" % (varSubkey, searchKeyname+"\control\DeviceClasses")
                varKey = OpenKey(varReg, firstsubKey)
                for i in range(1024):
                    try:
                        keyname = EnumKey(varKey, i)
                        if keyname in deviceclass :
                            # print(keyname)
                            varSubkey2 = "%s\\%s" % (firstsubKey, keyname)
                            varKey2 = OpenKey(varReg, varSubkey2)
                            for j in range(1024):
                                try:
                                    keyname2 = EnumKey(varKey2, j)
                                    # print(keyname2)
                                    if "#USBSTOR#" in keyname2 :
                                        varSubkey3 = "%s\\%s" % (varSubkey2, keyname2)
                                        varKey3 = OpenKey(varReg, varSubkey3)
                                        for k in range(1024) :
                                            _name, _value, _type = EnumValue(varKey3, k)
                                            # print (_value)
                                            search4log.append(searchKeyname + " : " + _value)
                                            d_class_id.append(searchKeyname + " : " + _value.split("\\")[1])
                                            d_unique_instance_id.append(searchKeyname + " : " + _value.split("\\")[2].split("&")[1])
                                except :
                                    errorMsg = "Exception Inner:", sys.exc_info()[0]
                            CloseKey(varKey3)
                            CloseKey(varKey2)
                    except :
                        errorMsg = "Exception Inner:", sys.exc_info()[0]
                CloseKey(varKey)
        CloseKey(searchKey)
```

- SYSTEM 하위 키 중 ControlSet 마다 모두 같은 하위 키가 존재하기 때문에, 가장 먼저 ControlSet00#을 탐색한다.
- 존재하는 ControlSet 만큼 다음 하위 경로를 탐색하며, USB 형식의 저장소일 경우 USBSTOR의 문자열을 포함하기 때문에 USBSTOR의 문자열의 값을 각각 Parsing하여 원하는 변수에 저장하도록 한다.

3.3 Definition Function D_classes_1

```
def D_classes_1() :
    global stor_d_name
    stor_d_name = list()
    varSubkey = "SYSTEM"
    varReg = ConnectRegistry(None, HKEY_LOCAL_MACHINE)
    varinitKey = OpenKey(varReg, varSubkey)
    for z in range(1024) :
        try :
            searchKey = OpenKey(varReg,varSubkey)
            searchKeyname = EnumKey(varinitKey, z)
            for num in range(1000) :
                if len(str(num)) == 1 :
                    number = "00"+str(num)
                elif len(str(num)) == 2 :
                    number = "0"+str(num)
                else :
                    number = str(num)

                if searchKeyname in ("ControlSet"+number) :
                    firstsubKey = "%s\\%s" % (varSubkey, searchKeyname+"\Enum\USBSTOR")
                    varKey = OpenKey(varReg, firstsubKey)
                    for i in range(1024) :
                        try :
                            keyname = EnumKey(varKey, i)
                            stor_d_name.append(searchKeyname + " : " + keyname)
                        except:
                            errorMsg = "Exception Inner:", sys.exc_info()[0]
                            break
```

- 최종 탐색 레지스트리 경로 : "HKLM/SYSTEM/ContorlSet00#/Enum/USBSTOR"
- 해당 레지스트리 경로는 USB 형식의 저장소가 분석 대상 PC에 연결이 되었을 경우, USB의 물리적인 장치 명이 하위키 형태로 존재한다.
- 해당 키로 앞서 분석한 USB 기본정보와 대조하여 추출한 정보를 검증한다.
- 해당 레지스트리 경로에도 ContorlSet의 키가 중간에 여러 개 존재할 가능성이 있다. 그렇기 때문에 SYSTEM Path 이후 ContorlSet의 개수를 파악하여 존재하는 모든 ControlSet의 하위 키를 탐색하여 정보를 확인한다.

3.4 Definition Function V_id_P_id

```
def V_id_P_id() :
    global v_id_p_id
    v_id_p_id = list()
    varSubkey = "SYSTEM"
    varReg = ConnectRegistry(None, HKEY_LOCAL_MACHINE)
    varinitKey = OpenKey(varReg, varSubkey)
    for z in range(1024) :
        try :
            searchKey = OpenKey(varReg,varSubkey)
            searchKeyname = EnumKey(varinitKey, z)
            for num in range(1000) :
                if len(str(num)) == 1 :
                    number = "00"+str(num)
                elif len(str(num)) == 2 :
                    number = "0"+str(num)
                else :
                    number = str(num)

                if searchKeyname in ("ControlSet"+number) :
                    firstsubKey = "%s\\%s" % (varSubkey, searchKeyname+"\Enum\\USB")
                    for i in range(1024) :
                        try :
                            varKey = OpenKey(varReg, firstsubKey)
                            keyname = EnumKey(varKey, i)
                            varSubkey2 = "%s\\%s" % (firstsubKey, keyname)
                            varKey2 = OpenKey(varReg, varSubkey2)
                            try :
                                for j in range(1024) :
                                    keyname2 = EnumKey(varKey2, j)
                                    varSubkey3 = "%s\\%s" % (varSubkey2, keyname2)
                                    # print (varSubkey3)
                                    varKey3 = OpenKey(varReg, varSubkey3)
                                    for k in range(1024) :
                                        n, v, t = EnumValue(varKey3, k)
                                        if ("ParentIdPrefix" in n) :
                                            for serial_loc in range (0, len(d_unique_instance_id)) :
                                                if d_unique_instance_id[serial_loc].split(" : ")[0] == searchKeyname and d_unique_instance_id[serial_loc].split(" : ")[1] in v :
                                                    v_id_p_id.append(searchKeyname + " : " + keyname)
                            except :
                                errorMsg = "Exception Inner:", sys.exc_info()[0]
                            CloseKey(varKey2)
                            CloseKey(varKeyK)
```

- 최종 레지스트리 경로 : "HKLM/SYSTEM/ControlSet00#/Enum/USB"
- 각 USB에는 제조사와 제조사에서 생성 시 고유한 물리 장치 명을 부여한다.
- 파일의 물리 장치 명 중 일부 문자열이며, 해당 부분으로 특정 제조사 및 USB 종류를 분류 시, 도움이 된다.
- 앞선 레지스트리 경로 분석과 동일하게 ControlSet이 여러 개 존재한다.

3.5 Definition Function V_id_P_id

```
def mountDevice() :
    global logic_drive_name
    logic_drive_name = list()
    varSubkey = "SYSTEM\MountedDevices"
    varReg = ConnectRegistry(None, HKEY_LOCAL_MACHINE)
    varKey = OpenKey(varReg, varSubkey)

    for i in range(1024) :
        try :
            _name, _value, _type = EnumValue(varKey, i)
            if "DosDevice" in _name :
                for serial_loc in range (0, len(d_unique_instance_id)) :
                    if d_unique_instance_id[serial_loc].split(" : ")[1] in str(_value).replace("\\x00","") :
                        logic_drive_name.append(d_unique_instance_id[serial_loc].split(" : ")[0] + " : " + _name[-2:])
        except :
            errorMsg = "Exception Outter:", sys.exc_info()[0]
            break
    CloseKey(varKey)
    CloseKey(varReg)
```

- 최종 레지스트리 경로 : "HKLM/SYSTEM/MountedDevices"
- USB 저장소를 연결 하였을 경우 PC에서 할당하는 드라이브 문자열이 부여되어, 같은 USB 일 경우에도 드라이브 문자열이 다르게 할당 되기 때문에 특정 드라이브로 명으로 파일의 이동 시, USB 특정에 사용할 수 있는 정보가 된다.

➢ 여러 번 연결 및 해제하면서 같은 드라이브 명이 다른 저장소에 할당될 경우도 있지만 해당 경우는 추후 분석 할 내용과 결합하여 분석을 진행하고, 동시에 USB가 다수 연결 되어 있을 경우, 드라이브 문자열이 모두 다르게 지정된다는 점을 이용한 분석에 사용된다.

3.6 Definition Function VolumeName

```
def VolumeName() :
    global win7_Volname, win10_Volname
    if os_release == '7' :
        win7_Volname = list()
        varSubkey = "SYSTEM\CurrentControlSet\Enum\WpdBusEnumRoot\\UMB"
        varReg = ConnectRegistry(None, HKEY_LOCAL_MACHINE)
        varKey = OpenKey(varReg, varSubkey)
        for j in range (0, len(d_class_id)) :
            for i in range (1024) :
                try :
                    keyname = EnumKey(varKey, i)
                    if d_class_id[j].split(" : ")[1].upper() in keyname :
                        varSubkey2 = "%s\\%s" % (varSubkey, keyname)
                        varKey2 = OpenKey(varReg, varSubkey2)
                        for k in range(1024) :
                            _name, _value, _type = EnumValue(varKey2, k)
                            if "FriendlyName" in _name :
                                win7_Volname.append(d_class_id[j].split(" : ")[0] + " : " + _value)
                        CloseKey(varKey2)
                except :
                    errorMsg = "Exception Outter:", sys.exc_info()[0]
                    break
        CloseKey(varKey)
        CloseKey(varReg)
```

➢ 최종 레지스트리 경로(Windows 7) :
"HKLM/SYSTEM/CurrentControlSet/Enum/WpdBusEnumRoot/UMB"

➢ Script 시작 시, 확인한 운영체제의 정보에 맞추어 레지스트리 키를 탐색한다.

➢ 최종 경로에 존재하는 키에 있는 값들 중 'FrendlyName'의 value는 USB연결 시, 드라이브 명과 함께 나타내는 USB의 볼륨 이름이다. USB 사용자가 임의로 변경이 가능한 부분으로서 해당 부분이 사용자의 특징이 담겨있을 가능성이 존재한다.

➢ 하위 키 안의 값들을 확인 할 경우 EnumValue와 연결된 키를 입력하고, 각각 존재하는 이름, 값, 종류에 맞게 하나씩 매칭한다.

3.7 Definition Function setupdevlog

```
# ==> * Find Install time from setupapi.dev.log Area *
def setupdevlog() :
    global initcontime, recentcontime
    initcontime = {}
    recentcontime = list()
    filepath = "C:\Windows\INF\setupapi.dev.log"
    file = open(filepath, 'r')
    contents = file.readlines()
    for i in range (0, len(search4log)) :
        for j in range (0, len(contents)) :
            if (search4log[i].split(" : ")[1] in contents[j] and "Install" in contents[j] and "start" in contents[j+1]) :
                initcontime = {search4log[i].split("\\")[1] : contents[j+1].split("start ")[1].replace("\n","").replace("/", "-")}
    file.close()
```

➢ 분석 대상 PC에 연결된 모든 외부 기기 들 중 앞서 레지스트리 분석을 이용하여 확인한 장치명에 맞는 최초 연결 시간을 분석한다.

➢ USB를 최초 연결 시, 분석 대상 PC에는 알맞은 Driver 설치를 진행하며, 설치한 기록이 setupapi.dev.log 파일에 기록되고 기록에는 설치 시작시간이 명시되어 있어, 해당 시간으로 최초 연결 시간을 파악할 수 있다.

3.8 Definition Function regi2dic

```
def regi2dic() :
    global usb_info

    usb_info = {}
    for j in range (0, len(controlSetinfo)) :
        device_list = list()
        stor_device_list = list()
        serial_list = list()
        v_p_list = list()
        logic_drive_list = list()
        volumename_list = list()
        for i in range (0, len(d_class_id)) :
            if controlSetinfo[j] == d_class_id[i].split(" : ")[0] :
                device_list.append(d_class_id[i].split(" : ")[1])
                stor_device_list.append(stor_d_name[i].split(" : ")[1])
                serial_list.append(d_unique_instance_id[i].split(" : ")[1])
                v_p_list.append(v_id_p_id[i].split(" : ")[1])
                logic_drive_list.append(logic_drive_name[i].split(" : ")[1])
                volumename_list.append(win7_Volname[i].split(" : ")[1])

                device_list = list(set(device_list))
                stor_device_list = list(set(stor_device_list))
                serial_list = list(set(serial_list))
                v_p_list = list(set(v_p_list))
                logic_drive_list = list(set(logic_drive_list))
                volumename_list = list(set(volumename_list))

                usb_info[controlSetinfo[j]] = {"Device_Name" : device_list  , "USBSTOR_Device_Name" : stor_device_list,
                "Serial_Number" : serial_list, "Vander & Product" :v_p_list,
                "Logical_Drive" : logic_drive_list, "Mounted_Volume_Name" : volumename_list}

    for j in range (0, len(controlSetinfo)) :
        initcon_list = list()
        for i in range (0, len(usb_info[controlSetinfo[j]]["USBSTOR_Device_Name"])) :
            if usb_info[controlSetinfo[j]]["USBSTOR_Device_Name"][i] in initcontime.keys() :
                initcon_list.append(initcontime[usb_info[controlSetinfo[j]]["USBSTOR_Device_Name"][i]])
                initcon_list = list(set(initcon_list))
                usb_info[controlSetinfo[j]]["Init Connect Time"] = initcon_list
```

- 레지스트리에서 추출한 값들을 모두 각각의 변수에 list형식으로 저장하였다.
- List에 저장된 값들을 분류하여 dictionary의 {key:value} 형식으로 변환하는 함수 부분이다.
- ControlSet 들을 넓은 범위의 key로 지정하고 각 ControlSet에서 "Device Name", "USBSTRO_Device_Name", "SeriaL_Number", "Vander & Product", "Logical_Drive", "Mounted_Volume_Name"의 key값을 지정하여 dictionary를 생성한다.
- 장치명을 이용하여 setupapi.dev.log에서 추출한 최초 시간이 ContorlSet과 "Device_Name"에 일치할 경우, "Init Connect Time"이라는 키를 생성하고 dictionary에 추가한다.

3.9 Definition Function Eventlogload

```
def Eventlogload() : # https://github.com/libyal/libevtx/wiki/Development
    global file, csvWriter
    file_object = open("C:\Windows\System32\winevt\Logs\Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx", "rb")
    evtx_file = pyevtx.file()
    evtx_file.open_file_object(file_object)
    file = open("./regi_evtx.csv", 'w', newline = "\n")
    csvWriter = csv.writer(file)
    csvWriter.writerow(['ControlSet', 'Device_Name', 'Serial_Number', 'Mounted_Volume_Name', 'Logical_Drive', 'Init Connected Time', 'Connected Time',
        'Disconnected Time', 'LifeTime'])

    # help(pyevtx.file) # How to use pyevtx
    # print ("\n================== * EventLog Contents * ==================")
    for event_cnt in range (0, evtx_file.number_of_records) :
        record = evtx_file.get_record(event_cnt)
        usb_event_log = xml_parser(record.xml_string)
        if usb_event_log != 0 :
            # print (usb_event_log, end=" "); print("\n")
            r_s_e_csv(usb_event_log)
        else :
            pass
    file.close()
    evtx_file.close()
```

- USB등 외부 기기를 PC에 연결 및 해제할 경우, 기록되는 Window EventLog를 분석한다.

- 분석 대상 EventLog : "C:₩Windows₩System32₩winevt₩Logs₩Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx"
- 해당 함수에서는 pyevtx를 이용하여 설정한 eventlog에 접근하여 데이터를 읽고, xml 형식으로 모든 event 기록 확인이 가능하도록 설정한다.
- 동시에 레지스트리에서 추출하여 dictionary 생성한 데이터와 eventlog를 분석한 데이터를 장치 명과 serial Number를 기준으로 분류하여 결합하고 csv로 저장하는 r_s_e_csv 함수를 호출한다.
- 이벤트로그에는 lifetime 값이 연결 및 해제에 기록되며 쌍으로 존재한다. 연결 시 liftetime과 해제 시 liftetime이 같으면 하나의 연결장치의 연결 및 해제를 나타낸다.

3.10 Definition Function xml_parser

```
# ==> * EventLog Microsoft-Windows-DriverFrameworks-UserMode%4Operational.evtx Parsing Area *
def xml_parser(xml_data) :
    if (xml_data[xml_data.find('<EventID>'):xml_data.find('</EventID>')][-4:] == '2003') :
        event_val = {}
        evtx_System = xml_data[xml_data.find('<System>') : xml_data.find('</System>')] # event_id, time, userid
        evtx_UserData = xml_data[xml_data.find('<UserData>') : xml_data.find('</UserData>')] # device name, lifetime

        e_id = xml_data[xml_data.find('<EventID>'):xml_data.find('</EventID>')][-4:]
        e_logtime = evtx_System[evtx_System.find('<TimeCreated SystemTime='):evtx_System.find('Z"/>')].split('="')[1].replace("T"," ")
        e_uid = evtx_System[evtx_System.find('<Security UserID='):].split('"')[1]
        e_device_name = evtx_UserData[evtx_UserData.find('instance='):].split("USBSTOR")[1].split("#")[1]
        e_serial = evtx_UserData[evtx_UserData.find('instance='):].split("USBSTOR")[1].split("#")[2].split("&")[1]
        e_liftime = evtx_UserData[evtx_UserData.find('lifetime='):].split('"')[1]

        event_val = {"Event_ID" : e_id, "Log_Time" : e_logtime, "User_ID" : e_uid, "Device_Name" : e_device_name,
        "Serial_Number" : e_serial, "Lifetime" : e_liftime}
        # print (event_val, end=" "); print ("\n")
        return event_val

    elif (xml_data[xml_data.find('<EventID>'):xml_data.find('</EventID>')][-4:] == '2100') :
        event_val = {}
        evtx_System = xml_data[xml_data.find('<System>') : xml_data.find('</System>')] # event_id, time, userid
        evtx_UserData = xml_data[xml_data.find('<UserData>') : xml_data.find('</UserData>')] # device name, lifetime

        e_id = xml_data[xml_data.find('<EventID>'):xml_data.find('</EventID>')][-4:]
        e_logtime = evtx_System[evtx_System.find('<TimeCreated SystemTime='):evtx_System.find('Z"/>')].split('="')[1].replace("T"," ")
        e_uid = evtx_System[evtx_System.find('<Security UserID='):].split('"')[1]
        e_device_name = evtx_UserData[evtx_UserData.find('instance='):].split("USBSTOR")[1].split("#")[1]
        e_serial = evtx_UserData[evtx_UserData.find('instance='):].split("USBSTOR")[1].split("#")[2].split("&")[1]
        e_liftime = evtx_UserData[evtx_UserData.find('lifetime='):].split('"')[1]

        event_val = {"Event_ID" : e_id, "Log_Time" : e_logtime, "User_ID" : e_uid, "Device_Name" : e_device_name,
        "Serial_Number" : e_serial, "Lifetime" : e_liftime}
        return event_val
        # print (event_val, end =" "); print("\n")
    else :
        return 0
```

- 분석 이벤트로그 파일에 존재하는 모든 이벤트 기록이 xml 형태로 불러오며, 해당 xml에서 Event ID, 로그 기록시간, User_ID, Device_Name, Serial_Number, Lifetime에 해당하는 값을 dictionary 형태로 저장한다.
- Xml을 parsing 할 경우 특정 연결 event id는 2003, 해제 event id는 2100이며, 해당 event id에 속하는 이벤트로그만 parsing하여 dictionary로 저장한다.

3.11 Definition Function r_s_e_csv

```
def r_s_e_csv(events) :
    # linkfile()
    # print (events, end=" "); print("\n")
    if events["Event_ID"] == '2003' :
        for i in range (0, len(controlSetinfo)) :
            for j in range (0, len(usb_info[controlSetinfo[i]]["USBSTOR_Device_Name"])) :
                if ((usb_info[controlSetinfo[i]]["USBSTOR_Device_Name"][j].upper() == events["Device_Name"]) and
                    (usb_info[controlSetinfo[i]]["Serial_Number"][j].upper() == events["Serial_Number"])) :
                    csvWriter.writerow([list(usb_info.keys())[i], usb_info[controlSetinfo[i]]["Device_Name"][j], usb_info[controlSetinfo[i]]["Serial_Number"][j],
                    usb_info[controlSetinfo[i]]["Mounted_Volume_Name"][j], usb_info[controlSetinfo[i]]["Logical_Drive"][j],
                    usb_info[controlSetinfo[i]]["Init Connect Time"][j], events["Log_Time"], "", events["Lifetime"]])
                else :
                    pass
    elif events["Event_ID"] == '2100' :
        for i in range (0, len(controlSetinfo)) :
            for j in range (0, len(usb_info[controlSetinfo[i]]["USBSTOR_Device_Name"])) :
                if ((usb_info[controlSetinfo[i]]["USBSTOR_Device_Name"][j].upper() == events["Device_Name"]) and
                 (usb_info[controlSetinfo[i]]["Serial_Number"][j].upper() == events["Serial_Number"])) :
                    csvWriter.writerow([list(usb_info.keys())[i], usb_info[controlSetinfo[i]]["Device_Name"][j], usb_info[controlSetinfo[i]]["Serial_Number"][j],
                    usb_info[controlSetinfo[i]]["Mounted_Volume_Name"][j], usb_info[controlSetinfo[i]]["Logical_Drive"][j],
                    usb_info[controlSetinfo[i]]["Init Connect Time"][j], "", events["Log_Time"], events["Lifetime"]])
                else :
                    pass
    else :
        pass
```

- 레지스트리에서 확인한 USB 물리 장치 명 및 Serial Number와 EventLog에서 확인한 물리 장치 명과 Serial Number를 이용하여 두 기준 값이 모두 일치할 경우 csv 1행에 ControlSet00#, Device Name, Serial Number, Mounted Volume Name, Logical Drive, Init Connect Time, Log Time, Liftime 순으로 기록한다.
- EventLog에서 확보한 Log Time의 경우, Event ID가 2003이면 연결 시간, 2100이면 해제 시간으로 분리하여 CSV에 저장한다.

3.12 Definition Function linkfile

```
def linkfile() :
    # "C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Start Menu" # 시작메뉴
    # "C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent" # 최근문서
    # "C:\Users\<username>\AppData\Roaming\Microsoft\Internet Explorer\Quick Launch" # 빠른실행
    # "C:\Users\<username>\Desktop" #바탕화면

    lnk_info ={}
    file1 = open("./LNK_Parse.csv", 'w', newline = "\n")
    csvWriter1 = csv.writer(file1)
    csvWriter1.writerow(['Filename', 'Lnk Modified Time', 'Target Create Time', 'Target Accessed Time', 'Target Modified Time',
        'Volume Serial', 'Volume ID', 'Local Based Path'])
    for (path, dirs, files) in os.walk("C:\\"):
        for filename in files:
            ext = os.path.splitext(filename)[-1]
            if (ext == '.lnk') or (ext == '.LNK'):
                lnkfile = "%s/%s" % (path, filename)
                lnk_info = parse_lnk(lnkfile)
                if len(lnk_info) == 8 :
                    csvWriter1.writerow([lnk_info["Filename"], lnk_info["Lnk_Modified_Time"], lnk_info["Create_Time"], lnk_info["Access_Time"], lnk_info["Modified_Time"],
                    lnk_info["Volume Serial"], lnk_info["Volume ID"], lnk_info["Local_Based_Path"]])
                else :
                    csvWriter1.writerow([lnk_info["Filename"], "", lnk_info["Create_Time"], lnk_info["Access_Time"], lnk_info["Modified_Time"], "",
                    lnk_info["Volume ID"], lnk_info["Local_Based_Path"]])
    file1.close()
```

- 분석 대상 PC의 Root Directory (“C:₩”)를 기준으로 하위에 있는 모든 링크파일을 확인한다.
- 확인된 링크파일의 정보를 각각 조사하여 dictionary 형태로 저장하도록 한다.
- Dictionary로 저장한 링크파일들의 정보를 csv 파일로 저장하는 함수이다.
- 링크파일의 파일 명, 링크파일 자체의 수정시간, 대상파일 생성시간, 대상파일 접근시간, 대상파일 수정시간, 링크파일을 실행한 저장장치의 Serial Number, 실행한 Volume의 상태, 링크파일로 실행한 원본 위치를 분석한다.
- 해당 함수의 경우 LNK_Parser.py를 import하여 사용한다.

4. 개발 Script_2 (LNK_Parser.py)

➢ LNK_Parser.py는 외부 https://github.com/gold1029/pylnker/blob/master/pylnker.py를 참조하였다.

4.1 링크파일 여부 검증

```
def parse_lnk(filename):
    #read the file in binary module
    lnk_info = {}
    f = open(filename, 'rb')
    flags = reverse_hex(read_unpack(f,20,4))

    try:
        assert_lnk_signature(f)
    except Exception as e:
        return "[!] Exception: "+ str(e)
```

```
def assert_lnk_signature(f):
    f.seek(0)
    sig = f.read(4)
    # print (sig)
    guid = f.read(16)
    if sig != b'L\x00\x00\x00':
        raise Exception("This is not a .lnk file.")
    if guid != b'\x01\x14\x02\x00\x00\x00\x00\x00\xc0\x00\x00\x00\x00\x00\x00F':
        raise Exception("Cannot read this kind of .lnk file.")
```

➢ 최초 lnk 또는 LNK 확장자 파일의 헤더를 검증하여 링크파일의 여부를 확인한다.

4.2 연결된 Target 파일의 MAC Time 확인

```
create_time = reverse_hex(read_unpack(f,28,8))
c_time = ms_time_to_unix_str(int(create_time, 16)).split(".")[0]

# Access time 8 bytes@ 0x24 = 36D
access_time = reverse_hex(read_unpack(f,36,8))
a_time = ms_time_to_unix_str(int(access_time, 16)).split(".")[0]

# Modified Time8b @ 0x2C = 44D
modified_time = reverse_hex(read_unpack(f,44,8))
m_time = ms_time_to_unix_str(int(modified_time, 16)).split(".")[0]
```

➢ 링크파일로 실행한 Target의 MAC Time을 확인한다.

➢ Ms_time_to_unix_str의 경우, 링크파일 시간 확인을 위해 접근한 offset의 data를 변환하면 unixtimestamp 형태로 출력되며, 해당 timestamp를 yyyy-mm-dd hh:mm:ss 형태의 시간으로 변환한다. (사용자가 생성한 함수)

```
def ms_time_to_unix_str(windows_time):
    time_str = ''
    try:
        unix_time = windows_time / 10000000.0 - 11644473600
        time_str = str(datetime.datetime.fromtimestamp(unix_time))
    except:
        pass
    return time_str
```

4.3 링크파일 생성되기 위해 실행한 파일의 저장소 확인(Link Flag 확인)

- 실행한 Volume의 상태, 링크파일로 실행한 원본 위치 확인의 경우 LinkFlag를 확인하여, HasLinkInfo가 존재하는 flag를 파일이 가지면 확인이 가능하며, 자세한 내용은 MSDN의 Linkfile Format 분석을 참고하였다. (참조문서 MSDN 기재)

5. 추가 내용

- 링크파일 분석

  → 링크파일을 분석하였을 경우, 링크파일 자체의 수정시간을 os module을 이용하여 filesystem 접근하여 확인하였다. 그러나 출력되지 않는 예외의 파일들이 존재하였으며, 이 경우 실행한 볼륨 정보도 확인이 불가능하는 불안전하였다.

- Shellbag

  → Shellbag의 경우 외부 github를 사용하려고 하였으나 개발 환경과 상이한 pyrhon version 지원 및 해당 code도 shellbag 파일 구조 및 저장 레지스트리 분석이 추가로 필요함을 느꼈다.

  → 추가 구현이 필요한 상황이며, 현재 parser 구현을 하지 못하였다.

- CSV 파일

  → CSV 형태로 저장 시, 레지스트리와 이벤트로그의 경우 기준 값을 선정하여 결합 후 저장하였으나 추가적으로 링크파일과도 연계해야 한다.

6. 참고문헌

- Python evtx : https://github.com/libyal/libevtx/wiki/Development
- Python evtx module pyevtx 설치 url : https://github.com/libyal/libevtx/releases/download/20170122/libevtx-alpha-20170122.tar.gz
- Linkfile MSDN : https://msdn.microsoft.com/en-us/library/dd871305.aspx
- Linkfile MSDN pdf : [MS-SHLLINK].pdf (위 MSDN 경로에서 Donwload)
- 레지스트리 분석 참조 1 : http://forensic-proof.com/archives/3632
- 레지스트리 분석 참조 2 : [BOB] 윈도우 침해사고 분석.pptx – 박상호
- Linkfile 분석 : FP-바로가기링크 파일-포렌식.pdf – forensic-Proof / J K Kim
- Shellbag : http://www.williballenthin.com/forensics/shellbags/#shellbags.py
- Shellbag code : https://github.com/williballenthin/python-registry