디바이스 드라이버 후킹

- 박근영 / mash9(devpia)

후킹은 프로그램 기술중에서 가장 멋진기술입니다.

하지만 일반적인 후킹(SetWindowsHookEx)은 윈도우가 제공하는 범위내에서 가능한것들이고 때때로 그것보다 아래 단계에서 후킹을 하고싶을때가 있는데 특히 Filemon이나 Packet sniffer 같은 멋진 프로그램들은 SetWindowsHookEx 가지고는 구현 불가능합니다. 여러분이 후킹에 목말라 있다면 좀 더 이 글을 읽어서 자신이 원하는 해답을 얻기를 바랍니다.

필요한 것

Visual Studio
Driver Development Kit

디바이스장치에 접근하는 방법은 몇가지가 있습니다. 필터 드라이버를 작성하는것과 드라이버를 후킹하는것인데 여기에서는 필터드라이버에 관한 내용은 다루지 않습니다. 필터드라이버는 현재 WDM책마다 최소한 한번 이상은 다루기 때문에 필터드라이버를 작성하시려면 책을 추천해 드립니다.

저는 윈도우 API중에서 SCM(Service control manager)을 이용해서 드라이버를 후킹하고 지우는 방법을 골랐는데 이유는 간단합니다. 재부팅이 무척 싫고 작업시간도 배로 늘어나기 때문에, 필터드라이버는 설치하면 재부팅을 해야합니다.

아래는 SCM을 이용해 드라이버를 등록하고 제거 하는방법이다.

```
BOOL CDeviceControlDlg::CreatDevice(char* sc_name, char* sc_path)

{
	SC_HANDLE hSCMan = NULL;
	SC_HANDLE hService = NULL;

	if((hSCMan = OpenSCManager(0, 0, SC_MANAGER_ALL_ACCESS)) == 0)
	{
		TRACE("서비스를 열수 없습니다.");
		return FALSE;
	}

	if((hService = CreateServiceA(hSCMan, sc_name, sc_name,
			SC_MANAGER_ALL_ACCESS, SERVICE_KERNEL_DRIVER,
```

```
                SERVICE_DEMAND_START, SERVICE_ERROR_NORMAL, sc_path,
                0, 0, 0, 0, 0)) == 0)
        {
                if(GetLastError() == ERROR_SERVICE_EXISTS)
                        TRACE("이미 설치되어있습니다.₩n");
                else if(GetLastError() == ERROR_SERVICE_MARKED_FOR_DELETE)
                        TRACE("서비스가 제거되도록 설정되어 있어 생성불가.₩n");
                CloseServiceHandle(hSCMan);
                return FALSE;
        }

        StartService(hService, 0, 0);
        TRACE("설치완료₩n");
        CloseServiceHandle(hService);
        CloseServiceHandle(hSCMan);
        return TRUE;
}

BOOL CDeviceControlDlg::UnloadDevice(char* sc_name)
{
        DWORD dwError = ERROR_SUCCESS;
        SC_HANDLE hSCMan = NULL;
        SC_HANDLE hService = NULL;
        SERVICE_STATUS serviceStatus;

        if((hSCMan = OpenSCManager(0, 0, SC_MANAGER_ALL_ACCESS)) == 0)
        {
                TRACE("서비스를 열수 없습니다.");
                return FALSE;
        }

        if((hService = OpenServiceA(hSCMan, sc_name, SERVICE_ALL_ACCESS)) == 0)
        {
                TRACE("서비스가 설치되지 않았습니다.₩n");
                CloseServiceHandle(hSCMan);
                return FALSE;
        }

        ControlService(hService,SERVICE_CONTROL_STOP,&serviceStatus);
```

```
        if(DeleteService(hService) == 0)
        {
                if(GetLastError() == ERROR_SERVICE_MARKED_FOR_DELETE)
                        TRACE("서비스가 이미 제거 설정되었습니다.₩n");
        }
        else
            TRACE("서비스가 제거되었습니다. 프로그램을 종료해야합니다.₩n");

        CloseServiceHandle(hSCMan);
        return TRUE;
}
```

SCM을 이용해서 디바이스를 등록하는 과정은 약간의 제약이 있는데 한개의 프로세스 스레드에서 서비스를 제거시 바로 제거되는것이 아니라 서비스 제거 마킹만 할뿐 실질적으로 프로세스를 종료해야 서비스가 제대로 제거됩니다. 제거 마킹상태에서는 다시 서비스가 생성되지 않습니다.

또 주목해야할 부분은

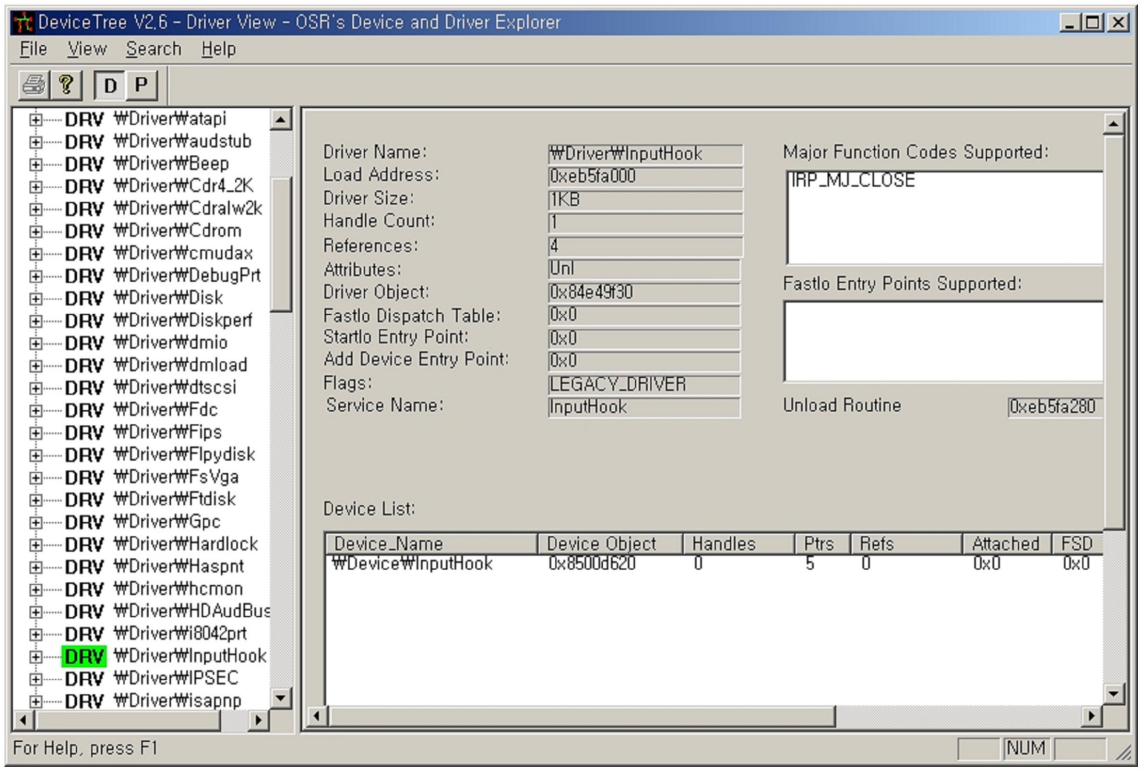
CreateServiceA(hSCMan, sc_name, sc_name, SC_MANAGER_ALL_ACCESS,
SERVICE_KERNEL_DRIVER, SERVICE_DEMAND_START, SERVICE_ERROR_NORMAL,
sc_path, 0, 0, 0, 0, 0))

이 부분에서 5 번째와 6번째 파라메터 부분

5번째는 커널드라이버이기 때문에 당연히 SERVICE_KERNEL_DRIVER
라고 넣었겠지만 6번째는 어째서 SERVICE_AUTO_START를 사용하지 않고

SERVICE_DEMAND_START 넣은후에 StartService(hService, 0, 0); 서비스 시작을 해야만 하였는가...

그 부분에 대해서는 아직 내공이 부족해서 답을 얻지 못했는데
다만 SERVICE_AUTO_START 를 했을경우 드라이버 엔트리포인트에 진입하지 못한다는것을 확인 했기 때문에 이 글을 읽는 분들이라면 이 같은 실수는 하지 않길 바랍니다.



이제 윈도우 DDK에서 제공되는 Device Tree라는 툴로 후킹 드라이버가 성공적으로 올라간 것을 확인할수 있습니다.

이번에는 직접 후킹드라이버의 기본 프레임을 작성해 보겠습니다.

여러분이 WDM에 어느정도 기본 지식이 있다는 가정하에 진행되겠습니다.

아래는 일반적인 드라이버의 엔트리 부분입니다.

```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING RegistryPath)
{
    UINT i;
    UNREFERENCED_PARAMETER (RegistryPath);

    for (i = 0 ; i <= IRP_MJ_MAXIMUM_FUNCTION ; i++)
        DriverObject->MajorFunction[i] = OnStubDispatch;
```

```
    DriverObject->MajorFunction [IRP_MJ_CREATE] = CreateHandler;
    DriverObject->MajorFunction [IRP_MJ_CLOSE] = CloseHandler;
    DriverObject->MajorFunction [IRP_MJ_PNP] = PNPHandler
    DriverObject->MajorFunction [IRP_MJ_POWER] = PowerHandler;
    DriverObject->MajorFunction [IRP_MJ_INTERNAL_DEVICE_CONTROL] = ControlHandler;

    DriverObject->DriverUnload = OnUnload;
    DriverObject->DriverExtension->AddDevice = AddDevice;

    return STATUS_SUCCESS;
}
```

하지만 우리가 시도하려는 후킹드라이버는 드라이버 엔트리를 진입하기는 하지만 정상적인 시작이 아니기 때문에 대부분의 함수 포인터를 채울 필요가 없습니다.

```
DriverObject->MajorFunction[IRP_MJ_CLOSE] = DrvClose;
DriverObject->DriverUnload = DrvUnload;
```

를 남기고 모두 지워 버리세요. SCM을 이용해서 드라이버를 생성시키고 시작하는 과정에는 IRP_MJ_CREATE, DriverObject->DriverExtension->AddDevice 함수포인터에 접근하지 않습니다.

그리고 드라이버엔트리에 직접 IoCreateDevice 으로 디바이스를 생성시킵니다.

한가지 재미있는 점은 드라이버엔트리는 DSP = 0 , Ring 0 의 권한을 가지고 있습니다.
때에 따라서 악명높은 CIH바이러스와 같이 바이오스를 지워버릴수도 직접 제어 할 수도 있습니다.

이야기를 다시 돌려서 생성시킨 장치는 IoCreateSymbolicLink로 등록 시킵니다.

이제 드라이버를 다시 작성하면

```
const WCHAR devicename[]=L"₩₩Device₩₩HookDriver";
const WCHAR devicelink[]=L"₩₩DosDevices₩₩HOOKDRIVER";

NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING
RegistryPath)
{
    PDEVICE_OBJECT devobject = 0;
    UNICODE_STRING devlink,devname;
    RtlInitUnicodeString(&devname,devicename);
```

```
    RtlInitUnicodeString(&devlink,devicelink);

    IoCreateDevice(DriverObject, 256, &devname, FILE_DEVICE_UNKNOWN, 0, FALSE,
&devobject);
    IoCreateSymbolicLink(&devlink,&devname);

    DriverObject->MajorFunction[IRP_MJ_CLOSE] = DrvClose;
    DriverObject->DriverUnload = DrvUnload;

    return STATUS_SUCCESS;
}

void DrvUnload(IN PDRIVER_OBJECT driver)
{
    UNICODE_STRING devlink;
    RtlInitUnicodeString(&devlink,devicelink);

    IoDeleteSymbolicLink(&devlink);
    IoDeleteDevice(driver->DeviceObject);
}

NTSTATUS DrvClose(IN PDEVICE_OBJECT device,IN PIRP Irp)
{
    Irp->IoStatus.Information=0;
    Irp->IoStatus.Status=0;
    IoCompleteRequest(Irp,IO_NO_INCREMENT);

    return 0;
}
```

과 같이 작성이 되겠습니다. 이제 후킹드라이버의 기본 프레임이 완성되었습니다.
드라이버 작성내용은 되도록 DDK관련 문서를 참고하셔서 보시는 편이 도움이 되십겁니다.

이번에는 실질적으로 드라이버에 훅을 걸어보겠습니다.

훅의 원리는 의외로 간단한데 훅을 걸려고 하는 드라이버 포인터를 얻어와서 훅 드라이버에 연결시키기만 하면 됩니다.

일단 훅을 걸려고하는 드라이버 포인터를 얻어오겠습니다.

```
NTSTATUS ObOpenObjectByName(IN POBJECT_ATTRIBUTES ObjectAttributes,
IN ULONG  OPTIONAL, IN ULONG AccessMode, IN ULONG OPTIONAL2,
IN ULONG DesiredAccess OPTIONAL, IN OUT PVOID ParseContext OPTIONAL,
OUT PHANDLE Handle);

PDRIVER_OBJECT SearchDriverObject(PUNICODE_STRING pUni)
{
    NTSTATUS st;
    HANDLE Handle;
    UNICODE_STRING Uni;
    OBJECT_ATTRIBUTES ObjectAttributes;
    PDRIVER_OBJECT Object;

    InitializeObjectAttributes( &ObjectAttributes, pUni, OBJ_CASE_INSENSITIVE, NULL, NULL
);

    st = ObOpenObjectByName( &ObjectAttributes, 0L, 0L, 0L, 0L, 0L, &Handle );
    if( st != STATUS_SUCCESS )
        return (PDRIVER_OBJECT)0;

    st = ObReferenceObjectByHandle( Handle, 0x80000000, NULL, 0, &Object, NULL );
    if( st != STATUS_SUCCESS )
    {
        ZwClose( Handle );
        return (PDRIVER_OBJECT)0;
    }

    ZwClose( Handle );
    ObDereferenceObject( Object );
    return Object;
}
```

위의 SearchDriverObject 함수는 '디바이스 구조와 원리'에서 발췌했습니다.

```
PDRIVER_OBJECT pDriver;

RtlInitUnicodeString(&Uni, L"₩₩Driver₩₩Serial");
pDriver = SearchDriverObject(&Uni);
```

이렇게 입력하면 시리얼 드라이버 포인터를 얻어오게 됩니다.

이제 원래 드라이버 포인터를 전역변수로 담아 놓습니다. 드라이버 포인터를 담아 놓는 이유는 후킹드라이버를 종료시켰을때를 위해서 입니다.

```
BackupReadHandler = pDriver->MajorFunction[IRP_MJ_READ];
BackupWriteHandler = pDriver->MajorFunction[IRP_MJ_WRITE];
```

그리고 새로운 함수포인터를 집어 넣습니다.

```
pDriver->MajorFunction[IRP_MJ_READ]  = ReadHandler;
pDriver->MajorFunction[IRP_MJ_WRITE]  = WriteHandler;
```

이 새로운 함수의 작성은 해당 드라이버의 DDK의 샘플을 참고로 작성하시면 됩니다. 물론 그대로 가져와 쓰셔도 됩니다.

```
PDRIVER_OBJECT pDriver;
PDEVICE_OBJECT BackupReadHandler;
PDEVICE_OBJECT BackupWriteHandler;

NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING RegistryPath)
{
    PDEVICE_OBJECT devobject = 0;
    UNICODE_STRING devlink,devname;

    DriverSearch();

    DriverObject->MajorFunction[IRP_MJ_READ]  = ReadHandler;
    DriverObject->MajorFunction[IRP_MJ_CLOSE]=DrvClose;
    DriverObject->DriverUnload = DrvUnload;

    RtlInitUnicodeString(&devname,devicename);
    RtlInitUnicodeString(&devlink,devicelink);

    IoCreateDevice(DriverObject, sizeof(DEVICE_EXTENSION), &devname, FILE_DEVICE_UNKNOWN, 0, FALSE, &devobject);
    IoCreateSymbolicLink(&devlink,&devname);

    return 0;
}

void DriverSearch()
```

```
{
    RtlInitUnicodeString(&Uni, L"₩₩Driver₩₩Serial");
    pDriver = SearchDriverObject(&Uni);

    BackupReadHandler = pDriver->MajorFunction[IRP_MJ_READ];
    BackupWriteHandler = pDriver->MajorFunction[IRP_MJ_WRITE];
}

void DrvUnload(IN PDRIVER_OBJECT driver)
{
    UNICODE_STRING devlink;

    RtlInitUnicodeString(&devlink,devicelink);

    pDriver->MajorFunction[IRP_MJ_READ] = BackupReadHandler
    pDriver->MajorFunction[IRP_MJ_WRITE] = BackupWriteHandler

    IoDeleteSymbolicLink(&devlink);
    IoDeleteDevice(driver->DeviceObject);
}
```

이렇게 3부로 나뉘어진 디바이스 드라이버 훅에 모든 설명이 끝났습니다.
좀 더 많은 부분을 설명하고 싶었는데 아쉽네요. 내공이 부족한 부분도 있고... 자료도 그렇고

특히 키보드나 마우스같은 입력장치 드라이버의 경우 IRP_MJ_WRITE 가 존재하지 않기때문에 버퍼에 직접 접근해서 값을 넣어줘야 합니다.

ntdd8042.h

IOCTL_INTERNAL_I8042_HOOK_KEYBOARD
IOCTL_INTERNAL_I8042_KEYBOARD_WRITE_BUFFER

사실 처음부터 그런것을 만들려고 했으면 필터드라이버 쪽이 더 편하지 않았나 싶네요.
WDM쪽은 너무 생소해서 해당분야에 종사하시는 분이 아니면 공부하는데도 한계가 있구요.
저도 시간 날때마다 조금씩 공부하는데 상당히 난해합니다....

이번 강좌를 통해서 WDM이 좀 더 쉽게 다가왔으면 하는 바램입니다.

참고 서적

EXPLOITING SOFTWARE : How to break code
MICROSOFT WINDOWS DRIVER MODEL
API로 배우는 Windows 구조와 원리
윈도우즈 드라이버 모델 WDM
디바이스 드라이버 구조와 원리 그리고 제작 노하우