Kernel Shim Engine for fun and not so much (but still a little?) profit

Or how to write a super long title for nothing :)

Who am i?

- Gaby - @pwissenlit
- RE engineer at Quarkslab (fr)
- Play with Windows Internals
- Attending BlackHoodie for the third time
  - And will probably do it again and again and again

Shim Engine

What is a Shim?

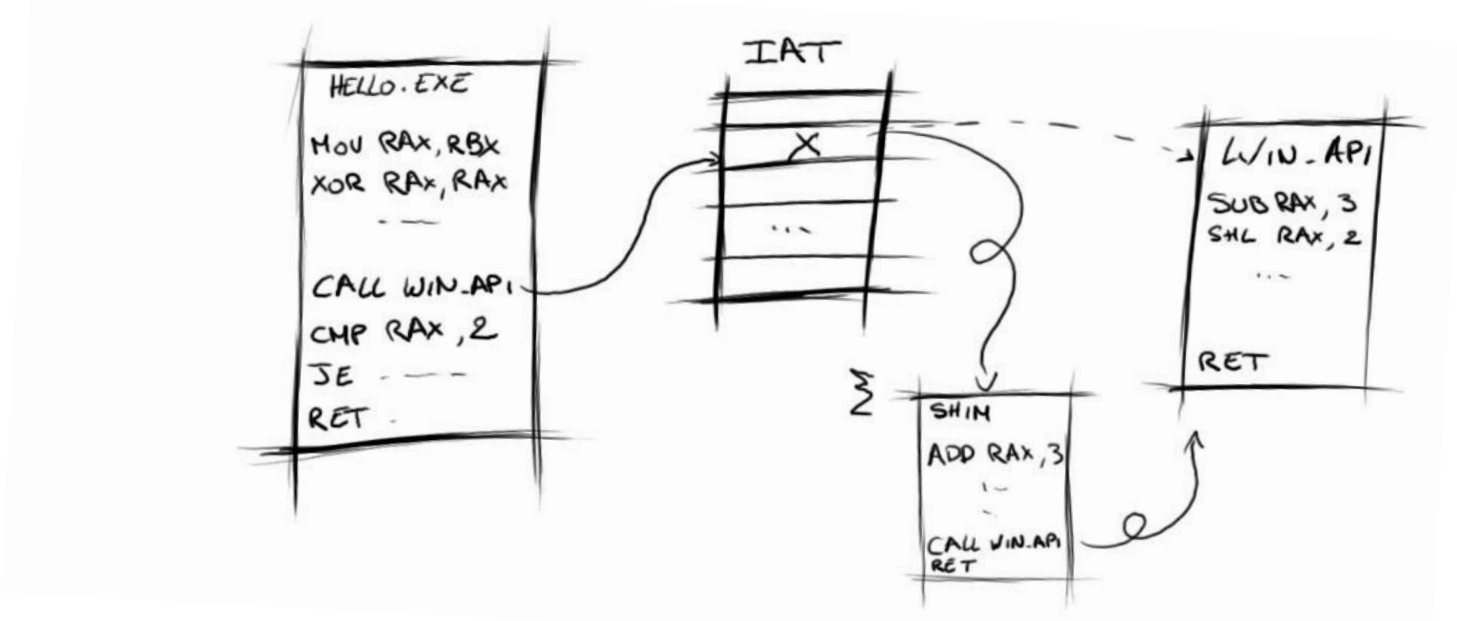
A shim is one of the very few four-letter words in use by Microsoft that isn't an acronym of some sort. It's a metaphor based on the English language word shim, which is an engineering term used to describe a piece of wood or metal that is inserted between two objects to make them fit together better. In computer programming, a shim is a small library which transparently intercepts an API, changes the parameters passed, handles the operation itself, or redirects the operation elsewhere. Shims can also be used for running programs on different software platforms than they were developed for.

©blogs.technet.microsoft.com

- A.k.a Windows Application Compatibility
- Mechanisms to ensure retro-compatibility for 3rd party apps
- Exist since Windows XP

Shim Engine

- Hot patch import address table (IAT) when the app is loaded
- Redirect exec flow before calling the external function



Useful if the API behaviour changed from what you were expecting

Kernel Shim Engine

- Since Windows 8.1
- Same idea – but in kernel!
- Not really known for some reasons
  - Only two (badass) guys talked about it (AFAIK)…
    - Alex Ionescu - Recon 2016
    - Geoff Chappell - (awesome) blog on windows internals

What does it do?

- Can be applied on drivers and devices
- Can hook:
  - Import address table (IAT)
  - Driver callbacks
    - DRIVER_OBJECT's DriverUnload, DriverStartIo, etc.
    - DRIVER_EXTENSION's AddDevice, etc.
  - I/O request packet (IRP)
- Applied when the driver is loaded

  -> Great way to ensure persistence :DDD

- Cf. Ionescu's slides at recon 2k16

What do we do?

Keylogger (speedrun)

- Need to hook the keyboard driver
  - i8042ptr.sys driver
- Class Service Callback routine
  - Retrieve the keystrokes
- IRP_MJ_DEVICE_CONTROL callback
  - Set the class service callback routine during the driver init

-> Let’s shim that callback <-

BTW how do we write a shim?

Recipe for an easy keylogger at home

Ingredients

A.k.a kernel shim components… :-^

Ingredients

Functions in ntoskrnl.exe

Start most of the time with Kse*:

- KsepEngineInitialize
- KseRegisterShim
- KseShimDatabaseOpen
- KsepResolveShimHooks
- KsepPoolAllocatePaged
- KsepGetShimsForDriver
- KsepApplyShimsToDriver
- etc.

No documentation but some symbols :)

Ingredients

Functions in ntoskrnl.exe

KSE engine in memory

Stores shim engine information like:

- Current engine status
- Shimmed drivers list
- Shim providers list
- Etc.

Ingredients

Functions in ntoskrnl.exe

KSE engine in memory

A bunch of shim providers

- Drivers storing the functions where the execution flow will be redirected to

Ingredients

Functions in ntoskrnl.exe

Database (SDB) on the file system

- Stores registered shims on the OS
  -> C:\Windows\apppatch\drvmain.sdb
- Binary file
- Same format as in userland
  (but with new tags…)
- The SDB is not signed!

KSE engine in memory

A bunch of shim providers

Ingredients

Functions in ntoskrnl.exe

Database (SDB) on the file system

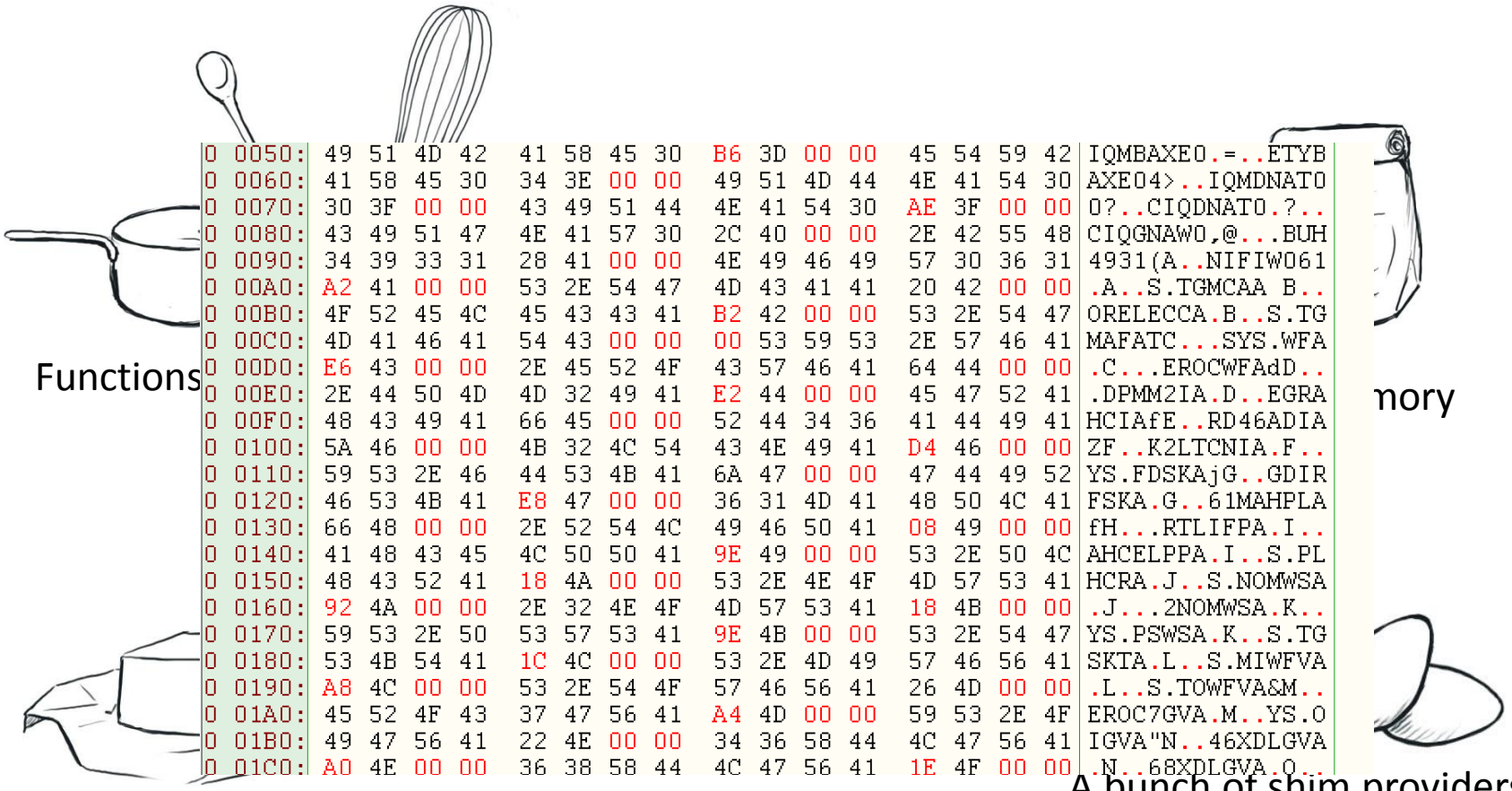
KSE engine in memory

Registry

- Not mandatory
- Override the SDB

A bunch of shim providers

Ingredients



Ingredients

But parsers exist ffs!

- Same format as in userland
- Some tools available:
  - Sdb2xml.exe
  - Sdb-explorer.exe
  - Etc.

But parsers exist ffs!

- Decompiled SDB:

```
<?xml version="1.0" encoding="UTF-16"?>
<DATABASE NAME="Microsoft Driver Compatibility Database" ID="{F9AB2228-3312-4A73-B6F9-936D70E112EF}">
[…]
<DRIVER NAME="WSRRCI" VENDOR="Wisair">
    <KDRIVER NAME="wsr_rci.sys" ID="{1E61CDCD-D929-4094-B3BD-1772F7459CBE}" RUNTIME_PLATFORM="X86">
      <KSHIM NAME="usbshim" COMMAND_LINE="null" />
    </KDRIVER>
  </DRIVER>
[…]
<LIBRARY>
    <KSHIM NAME="autofail" ID="{407D63CE-419D-4550-B54A-4F1C1B5BDD9F}" ONDEMAND="YES" FILE="autofail" />
[…]
    <KSHIM NAME="usbshim" ID="{FD8FD62E-4D94-4FC7-8A68-BFF7865A706B}" FILE="usbd" />
  </LIBRARY>
```

But not that much for editing…

- Geoff Chappell's article in PoC||GTFO 13:9
- Or… We can write our own!
  - Kaitai struct to the rescue \o/

```
meta:
  id: sdb
  file-extension: sdb
  endian: le
seq:
  - id: magic
    type: header
  - id: index_section
    type: section
  - id: database_section
    type: section
  - id: string_table
    type: section
types:
  header:
    seq:
      - id: majorversion
        size: 4
      - id: minorversion
        size: 4
      - id: magic
        contents: 'sdbf'
  section:
    seq:
      - id: dbheader
        type: header_tag
      - id: dbcontent
```

hex viewer

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	03	00	00	00	00	00	00	00	73	64	62	66	02	78	0e	3b	........sdbf.x.;
00000010	00	00	03	78	00	0e	00	00	02	38	07	70	03	38	01	60	...x.....8.p.8.`
00000020	16	40	01	00	00	00	01	98	ec	0d	00	00	2e	54	41	44	.@.......ì....TAD
00000030	4d	4d	34	30	58	3b	00	00	43	49	51	48	43	52	41	30	MM40X;..CIQHCRA0
00000040	d6	3b	00	00	45	50	41	54	54	4c	44	30	54	3c	00	00	Ö;..EPATTLD0T<..
00000050	49	51	4d	42	41	58	45	30	d2	3c	00	00	45	54	59	42	IQMBAXE0Ò<..ETYB
00000060	41	58	45	30	50	3d	00	00	49	51	4d	44	4e	41	54	30	AXE0P=..IQMDNAT0
00000070	4c	3e	00	00	43	49	51	44	4e	41	54	30	ca	3e	00	00	L>..CIQDNAT0Ê>..
00000080	43	49	51	47	4e	41	57	30	48	3f	00	00	2e	42	55	48	CIQGNAW0H?...BUH
00000090	34	39	33	31	44	40	00	00	4e	49	46	49	57	30	36	31	4931D@..NIFIW061
000000a0	be	40	00	00	53	2e	54	47	4d	43	41	41	3c	41	00	00	¾@..S.TGMCAA<A..
000000b0	4f	52	45	4c	45	43	43	41	ce	41	00	00	53	2e	54	47	ORELECCAÎA..S.TG
000000c0	4d	41	46	41	7c	42	00	00	00	53	59	53	2e	57	46	41	MAFA|B...SYS.WFA
000000d0	0e	43	00	00	2e	45	52	4f	43	57	46	41	8c	43	00	00	.C...EROCWFA.C..
000000e0	2e	44	50	4d	4d	32	49	41	0a	44	00	00	45	47	52	41	.DPMM2IA.D..EGRA
000000f0	48	43	49	41	8e	44	00	00	52	44	34	36	41	44	49	41	HCIA.D..RD46ADIA
00000100	82	45	00	00	4b	32	4c	54	43	4e	49	41	fc	45	00	00	.E..K2LTCNIAüE..

Need just a bit of work to have the builder

Recipe

How to write a shim in few^W^W^W a lot of slides…

1- Create a provider

- Create a driver & implement the hook functions
- Define the shim and hooks
  - Register them in the KSE engine with:

```
NTSTATUS KseRegisterShimEx(
                    KSE_SHIM *pShim,
                    PVOID ignored,
                    ULONG flags,
                    DRIVER_OBJECT *pDrv_Obj);
```

-> exported by ntoskrnl.exe but not declared in any headers!

1- Create a provider

- Shim object

```
typedef struct _KSE_SHIM {
        _In_ SIZE_T          Size;
        _In_ PGUID           ShimGuid;
        _In_ PWCHAR          ShimName;
        _Out_ PVOID          KseCallbackRoutines;
        _In_ PVOID           ShimmedDriverTargetedNotification;
        _In_ PVOID           ShimmedDriverUntargetedNotification;
        _In_ PVOID           HookCollectionsArray; // array of _KSE_HOOK_COLLECTION
} KSE_SHIM, *PKSE_SHIM;
```

1- Create a provider

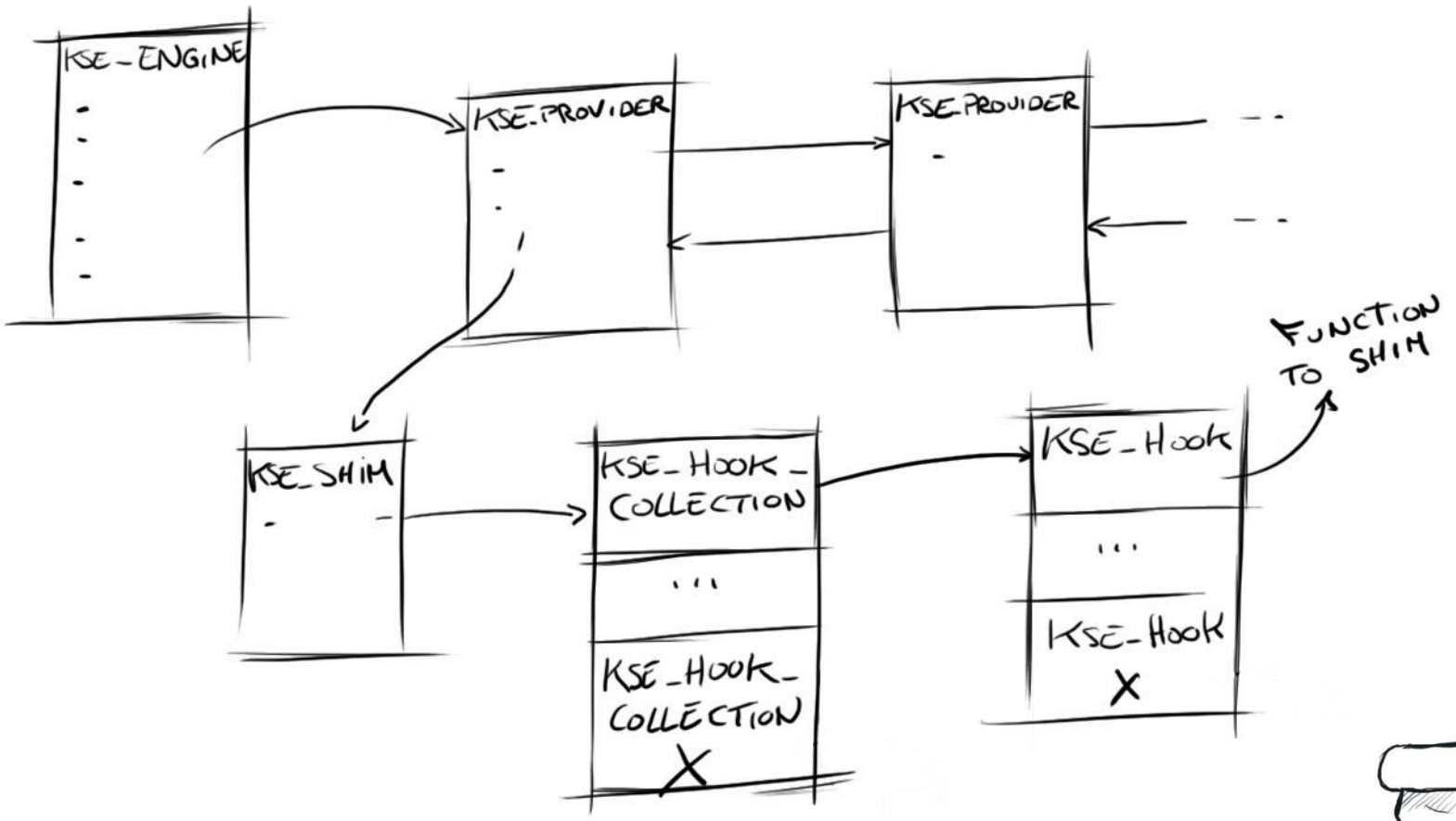
- Collection of similar hooks

```
typedef struct _KSE_HOOK_COLLECTION {
        ULONG64  Type;        // 0: NT Export, 1: HAL Export, 2: Driver Export, 3: Callback, 4: Last
        PWCHAR   ExportDriverName;    // If Type == 2
        PVOID    HookArray;           // array of _KSE_HOOK
} KSE_HOOK_COLLECTION, *PKSE_HOOK_COLLECTION;


KSE_HOOK_COLLECTION pCollecArray[2];
pCollecArray[0].Type = 3;                       // Driver callback
pCollecArray[0].ExportDriverName = NULL;
pCollecArray[0].HookArray = pHookArray;

pCollecArray[1].Type = 4;                       // Last entry in array
pCollecArray[1].ExportDriverName = NULL;
pCollecArray[1].HookArray = NULL;
```

1- Create a provider

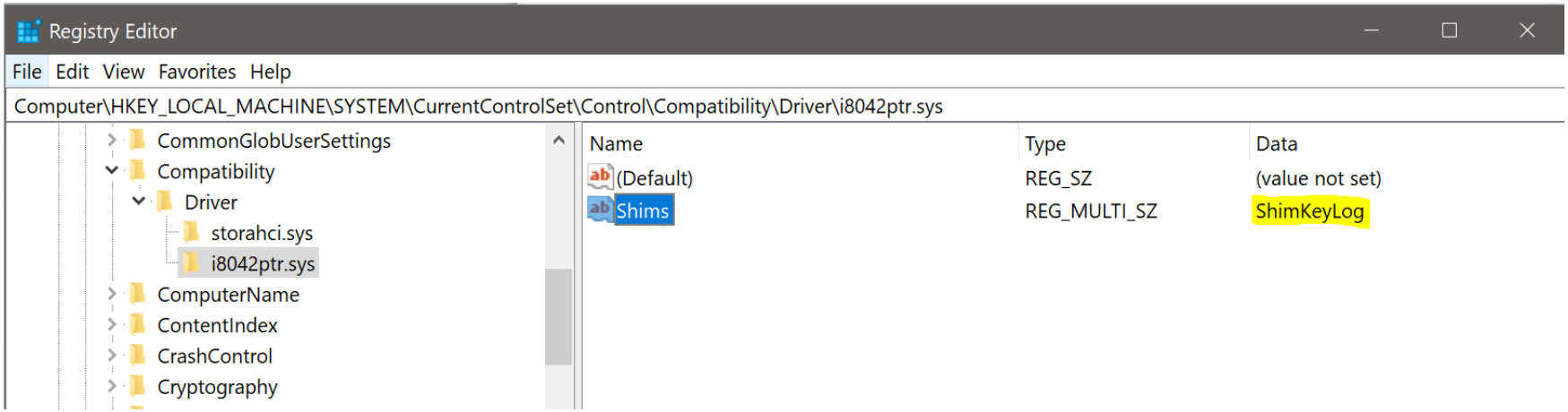
```
typedef struct _KSE_HOOK {
        _In_ ULONG64 Type;                      // 0: Function, 1: IRP Callback, 2: Last
        union {
                _In_ PCHAR FunctionName;        // If Type == 0
                _In_ ULONG64 CallbackId;        // If Type == 1
        };
        _In_ PVOID HookFunction;
        _Out_ PVOID OriginalFunction;
} KSE_HOOK, *PKSE_HOOK;

KSE_HOOK pHookArray[2];
pHookArray[0].Type = 1;                         // IRP Callback
pHookArray[0].CallbackId = 115;                 // IRP_MJ_DEVICE_CONTROL
pHookArray[0].HookFunction = (PVOID)ShimCallbackAddr;
pHookArray[0].OriginalFunction = NULL;

pHookArray[0].Type = 2;                         // Last entry in array
pHookArray[0].FunctionName = NULL;
pHookArray[0].HookFunction = NULL;
pHookArray[0].OriginalFunction = NULL;
```

In the KSE engine in memory…



2- Associate the shim with a driver

- With the Registry
  - Easy peasy…



- With the SDB
  - C:\Windows\apppatch\drvmain.sdb
  - A bit harder to modify…

3- Associate the shim with the provider

- Hijack a shim already defined in the SDB
  - Register the provider (step 1) with the same name as in the SDB
    - autofail.sys… ;))
- Add a new entry in the SDB

3- Associate the shim with the provider

```
Strnametag = BinaryTag()
Strnametag.tag = 0x8801
Strnametag.data = unicodeStr('ShimKeyLog')
Strnametag.buffer_size = len(strnametag.data)

offset = sdb_str_section.getsize()
sdb_str_section.append(strnametag)

Kshim_name = ParentBlock()
Kshim_name.tag = 0x6001
Kshim_name.reste =  offset

[…]

Kshim_tag = ParentBlock()
Kshim_tag.tag = 0x7025
Kshim_tag.reste = ListTag()
Kshim_tag.reste.addList([Kshim_name, Kshim_guid, Kshim_flag, Kshim_module])

sdb_db_section.append(Kshim_tag)
```

To sum up

1. Create a shim provider (driver)
   - Define the hooks and the shims structure
   - Register the shim provider in the KSE Engine
2. Define the modules that should use the shim
   - Either in the registry or in the shim database
3. Add the correlation between shim and shim provider in the sdb
   - Or hijack one already defined

For fun!

Demo time \o/

Plz, demo god!

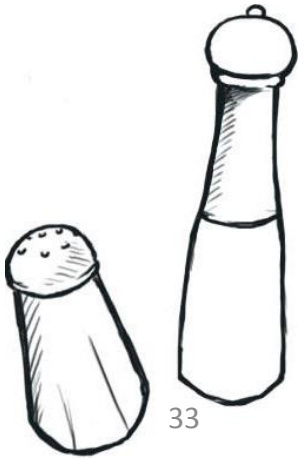
…

No actually I don’t trust you…

Not reaaally for profit

- Good points
  - Hard to detect if you don't know where to look
  - Kind of legit actually
    - Not that much ugly hooking required
- Bad points
  - Need to sign the provider
    - Expensive… >.>'
  - Hard to load through a vulnerable driver
    - at boot time or by reloading the shimmed driver
  - Hard to use for really early started drivers

Question?



Just in case…

Keylogger

- Key pressed
- IRQ sent to CPU -> interrupt!
- Call the interrupt handler (ISR)
- Cannot do the job -> dispatch DPC
  - To execute a routine later
- Call the Deferred Routine from the keyboard driver
- Call a Class Service Callback routine
- --> retrieve the data from the hardware

MSDN with love

Remarks

Keyboard Class Service Callback

Here is the definition of the keyboard class service callback routine.

Kbdclass uses an **IOCTL_INTERNAL_KEYBOARD_CONNECT** request to connect its class service callback to a keyboard device. In this call, the driver sets its implementation in a **CONNECT_DATA** structure.

- Device Input and Output Control
  - Control code used to communicate with the driver
  - Callback #15 called on driver side
    - IRP_MJ_DEVICE_CONTROL for the ones who wonder…
    - Driver performs the job assigned to the IOCTL