CALLBACK OBJECTS

Everything you didn’t know you wanted to hook in the kernel

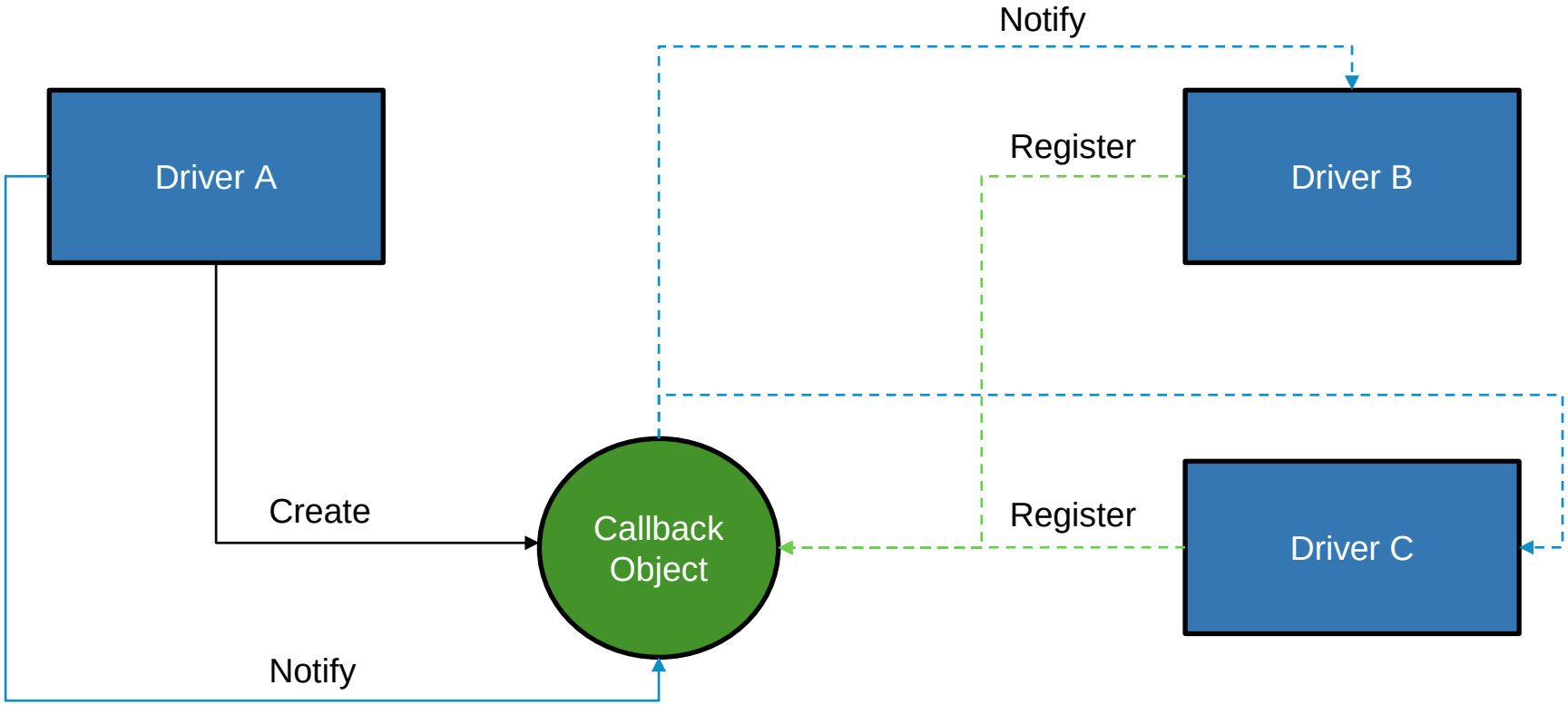
Yarden Shafir

ABOUT ME

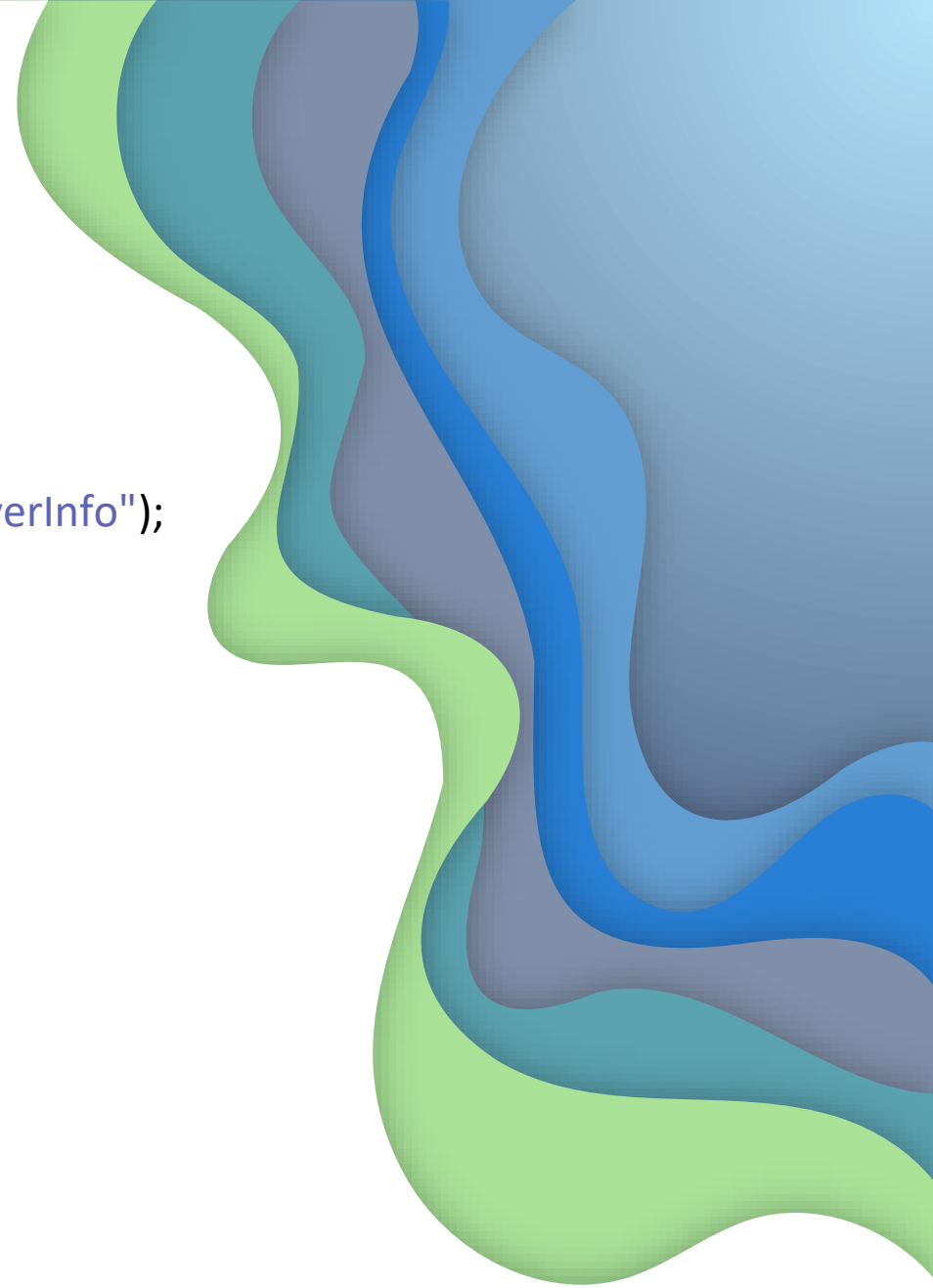
- Circus artist and aerial instructor
- Software engineer @Crowdstrike
- Windows Internals instructor @Winsider
- Former pastry chef
- Taught mom how to unmute zoom
- Almost succeeded at training a cat
- Usually found upside down

ANOTHER KERNEL COMMUNICATION MECHANISM

- Named Objects
- Can be created by any driver with ExCreateCallback
- Other drivers can register with ExRegisterCallback
  - Get a handle with ExCreateCallback
  - Can be configured on only allow one registered function
- Any driver can notify with ExNotifyCallback
  - All registered drivers need to know what types to expect in both arguments


Notify
Driver A
Register
Driver B
Create
Callback Object
Register
Driver C
Notify

```
PCALLBACK_OBJECT CallbackObject;
PCALLBACK_REGISTRATION CallbackHandle;
OBJECT_ATTRIBUTES ObjectAttributes;
UNICODE_STRING CallbackName;
RtlInitUnicodeString(&CallbackName, L"\\Callback\\SeImageVerificationDriverInfo");
InitializeObjectAttributes(
        &ObjectAttributes,
        &CallbackName,
        OBJ_KERNEL_HANDLE | OBJ_CASE_INSENSITIVE | OBJ_PERMANENT,
        nullptr,
        nullptr);
status = ExCreateCallback(&CallbackObject, &ObjectAttributes, 0, 1);
CallbackHandle = (PCALLBACK_REGISTRATION)ExRegisterCallback(
        CallbackObject,
        (PCALLBACK_FUNCTION)ImageVerificationCallbackFunction,
        nullptr);
```



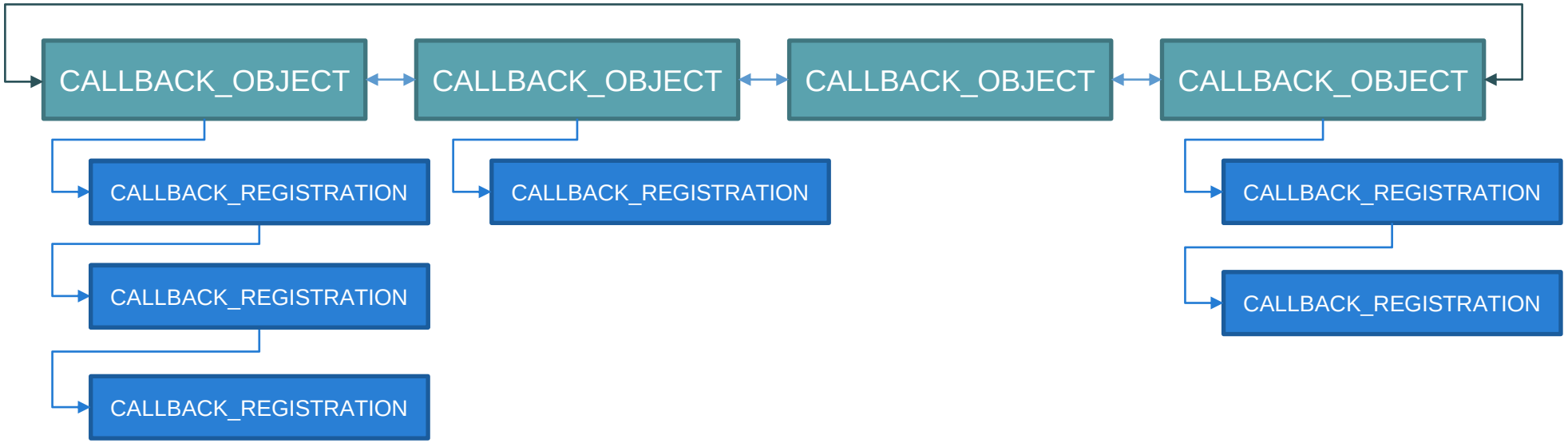
STRUCTURES AND FIELDS

```
typedef struct _CALLBACK_REGISTRATION
{
    LIST_ENTRY Link;
    PCALLBACK_OBJECT CallbackObject;
    PCALLBACK_FUNCTION CallbackFunction;
    PVOID CallbackContext;
    ULONG Busy;
    BOOLEAN UnregisterWaiting;
} CALLBACK_REGISTRATION, *PCALLBACK_REGISTRATION;
```

```
typedef struct _CALLBACK_OBJECT
{
    ULONG Signature;
    KSPIN_LOCK Lock;
    LIST_ENTRY RegisteredCallbacks;
    BOOLEAN AllowMultipleCallbacks;
    UCHAR reserved[3];
    LIST_ENTRY CallbackList;
} CALLBACK_OBJECT, *PCALLBACK_OBJECT;
```

CALLBACKS ARE LINKED IN A LIST

- So are the registered functions
- Get a pointer to one, you get them all

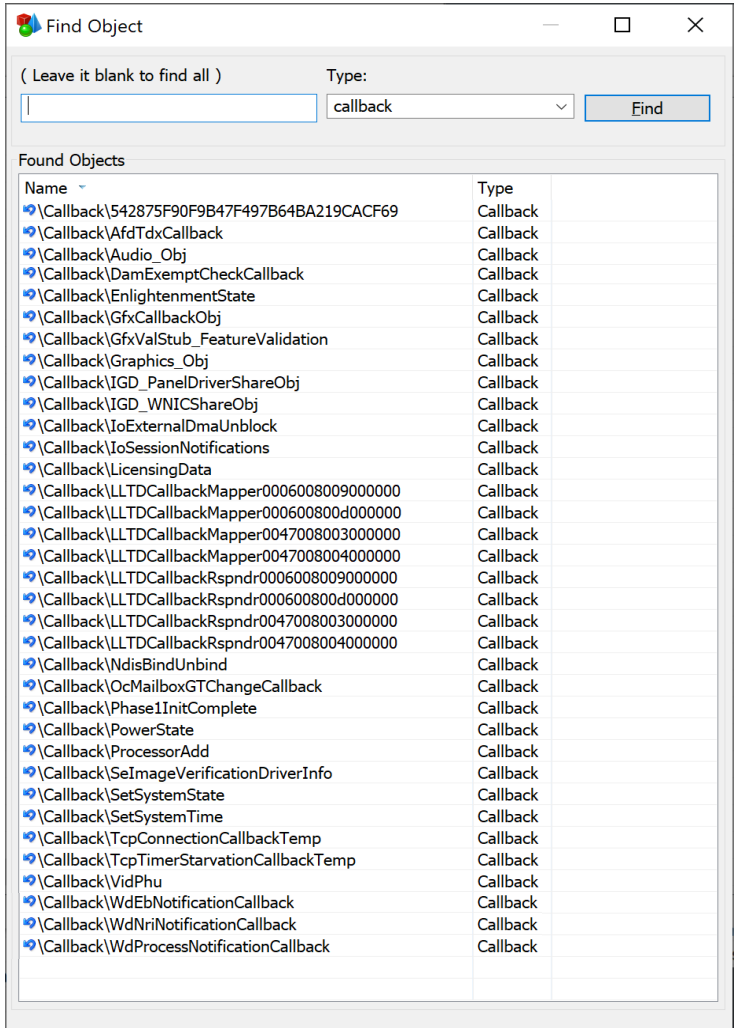


WHO USES CALLBACKS?

- The kernel notifies drivers about OS events with callbacks
- Win32k uses anonymous callbacks
- Security products use them for internal communication
- PatchGuard has a callback used to trigger checks

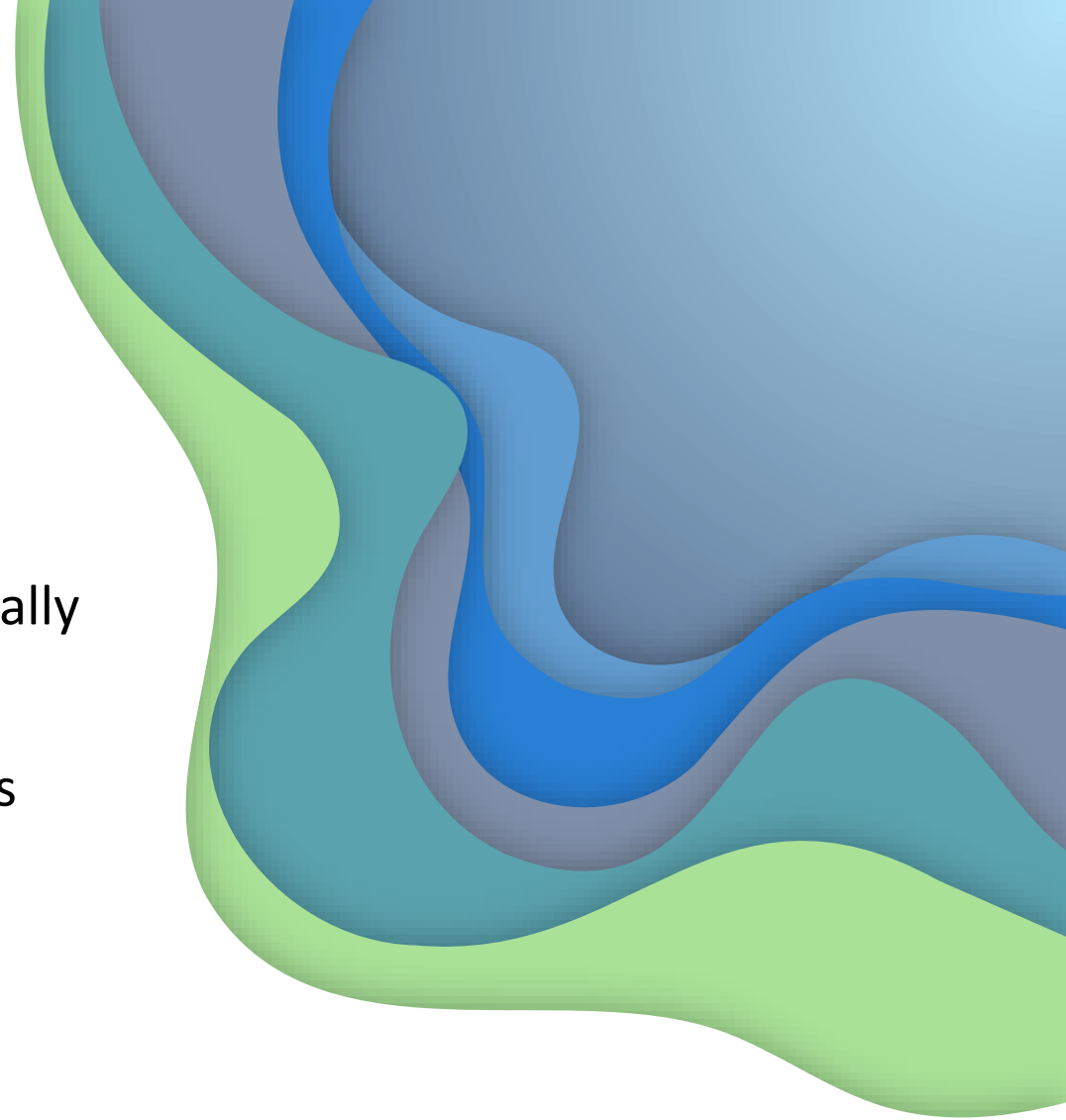
A FEW EXAMPLES…

- Graphics callbacks
- Power State
- SeImageVerificationDriverInfo
- Windows Defender Callbacks
- There are also unnamed callbacks that don’t appear here
  - Those are harder to find and use because you can’t open them by name
  - Can still find them by iterating over the callbacks list



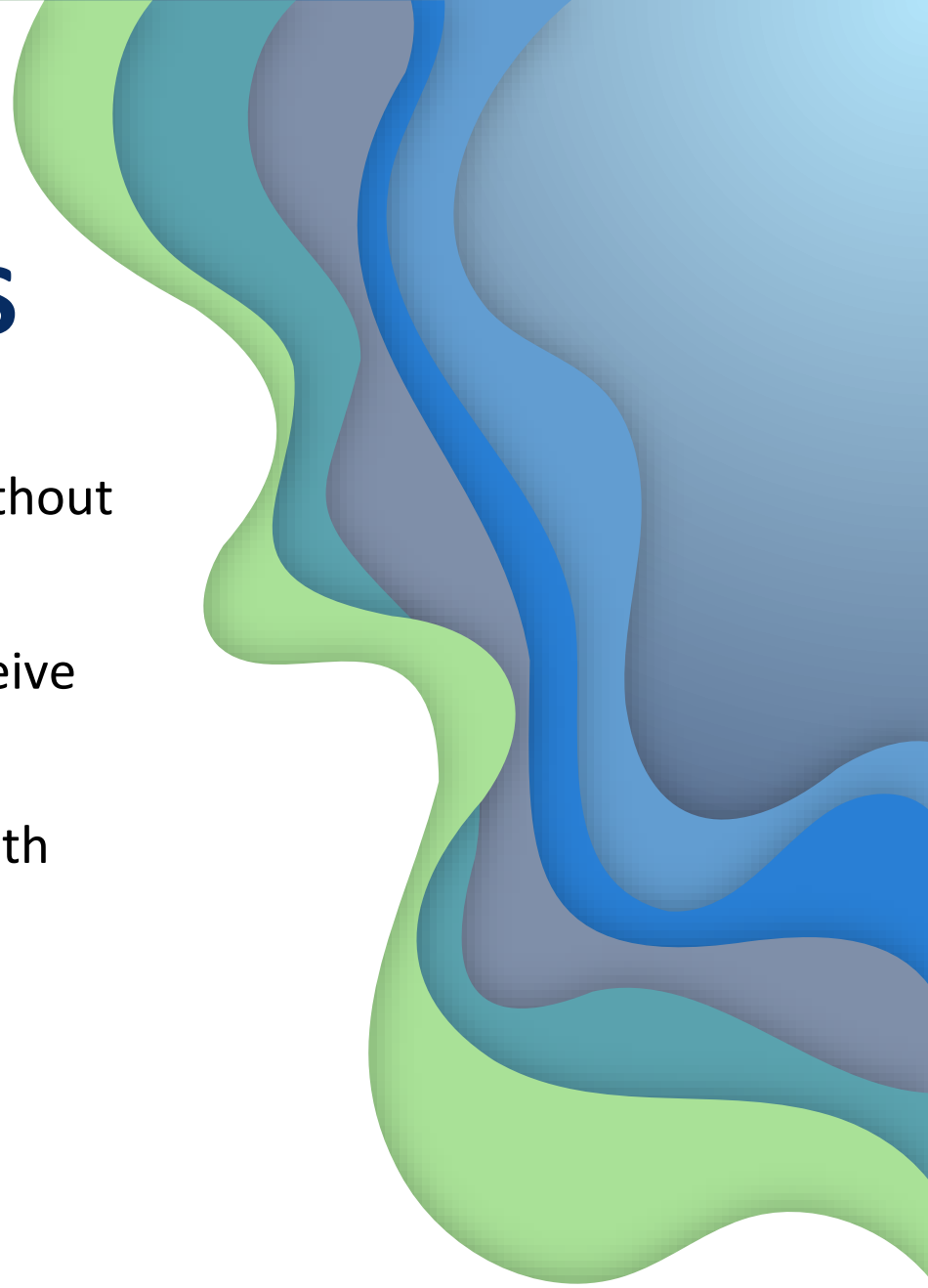
HOW CAN WE ABUSE THIS?

- Listen to private driver communication
- Hook callbacks to get interesting data without officially “registering”
- Notify callbacks with false data and “lie” to products
- Find interesting pointers and structures in callback contexts



WINDOWS DEFENDER CALLBACKS

- Process notification – A way to get process information without using the well-known callbacks
- Nri service – called when the network service starts to receive process information
- Boot driver callback (ELAM) – Called by the ELAM driver with information about early loaded drivers



PATCH GUARD CALLBACK

- Callback Name: 542875F90F9B47F497B64BA219CACF69
- MsSecFlt.sys registers function SecKernelIntegrityCallback
- Notified once when PG is initialized
  - Sets a pointer to a function that will be called for periodic PG checks
- Nothing checks if the pointer is already set
  - Pointer can be replaced after being set the first time