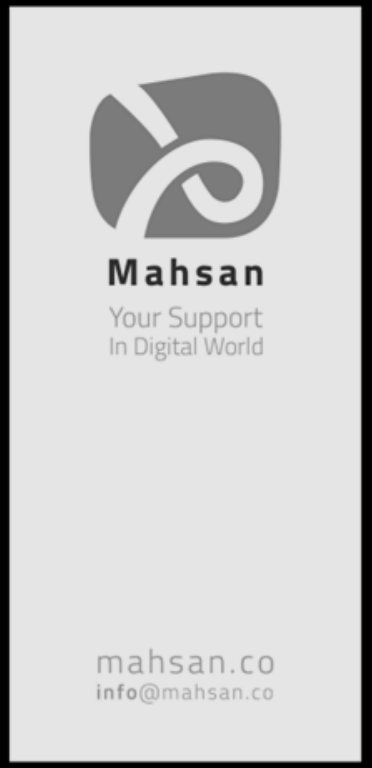
INTERNALS SESSIONS 04:
Windows Kernel Modules 101

Abolfazl Kazemi



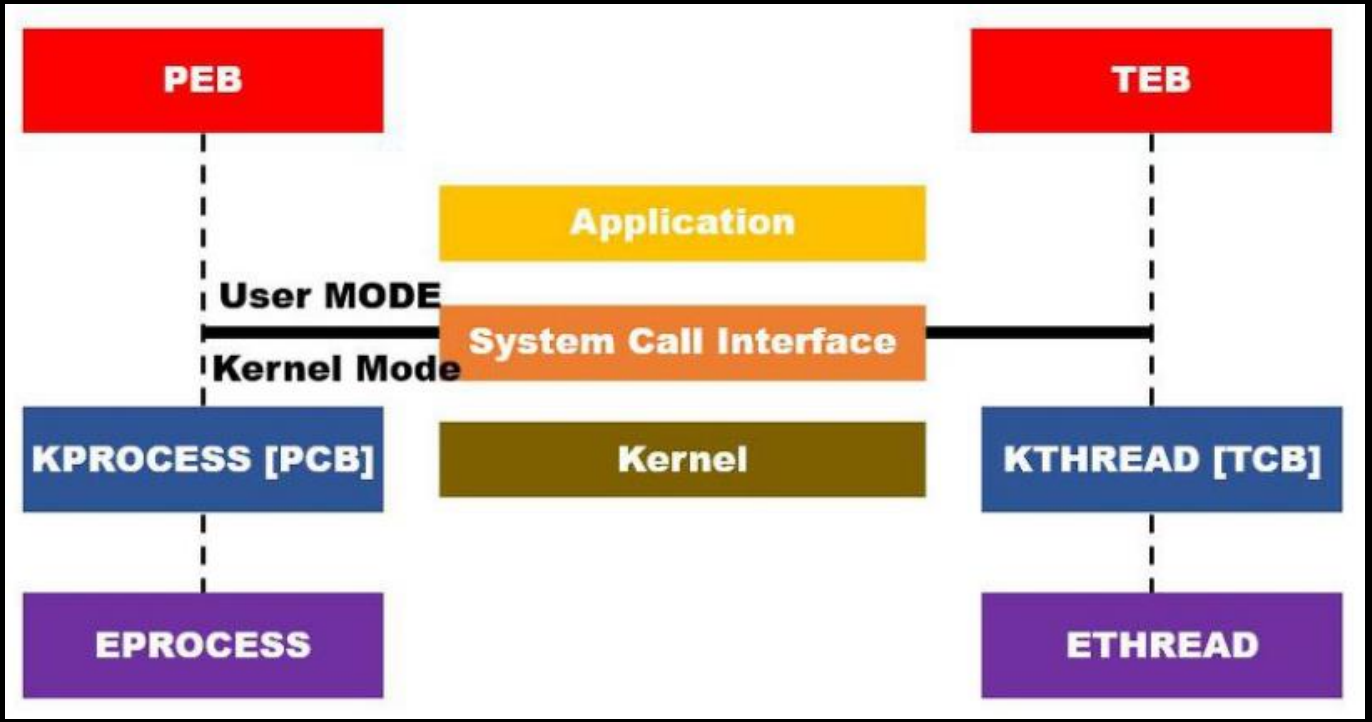
Agenda

- Kernel Modules Introduction
- Windows Driver Models
- Driver Signing and Development
- Kernel LinkedLists and Tokens
- Anatomy of a Driver and IRPs
- Kernel Rootkits

مهسان
تکیه گاه شما
در دنیای هوشمند

User Mode vs. Kernel Mode

- Thread access modes
- User mode
  - Allows access to non-operating system code & data only
  - No access to the hardware
  - Protects user applications from crashing the system
- Kernel mode
  - Privileged mode for use by the kernel and device drivers only
  - Allows access to all system resources
  - Can potentially crash the system



Device Drivers

- Device drivers are loadable kernel modules
  - Technically the only officially supported way to get 3rd party code into the kernel
- Classic device drivers provide the “glue” between hardware devices and the operating system
- Several ways to segregate device driver into categories
- User mode device drivers
  - Printer drivers
  - Drivers based on the User Mode Driver Framework (UMDF since Vista)
- Kernel mode drivers
  - File system drivers
  - Plug & Play drivers
  - Software drivers (non-Plug & Play drivers)

مهسان
تکیه گاه شما
در دنیای هوشمند

Kernel Device Drivers

- Always execute in kernel mode
  - Use the kernel mode stack of a thread
  - Image part of system space
  - Unhandled exceptions will crash the system
- Typically has a SYS file extension
- Usually invoked by client code
  - Usually `ReadFile`, `WriteFile`, `DeviceIoControl`
- Exports entry points for various functions
  - Called by system code when appropriate
- System handles all device independent aspects of I/O
  - No need for hardware specific code or assembly

مهسان
تکیه گاه شما
در دنیای هوشمند

The Windows Driver Model (WDM)

- Drivers for Windows 95 and NT 4 were completely separate
- WDM is a model for writing device drivers
  - Mostly source compatible between Windows 98/ME and Windows 2000/XP
  - Supports a wide range of buses (PCI, USB, IEEE1394 and more)
  - Extensible to support future buses
  - Supports a wide range of device classes (HID, Scanners, Cameras, etc.)
  - Can still be used today
- Not included in WDM
  - File system drivers
  - Video drivers

مهسان
تکیه گاه شما
در دنیای هوشمند

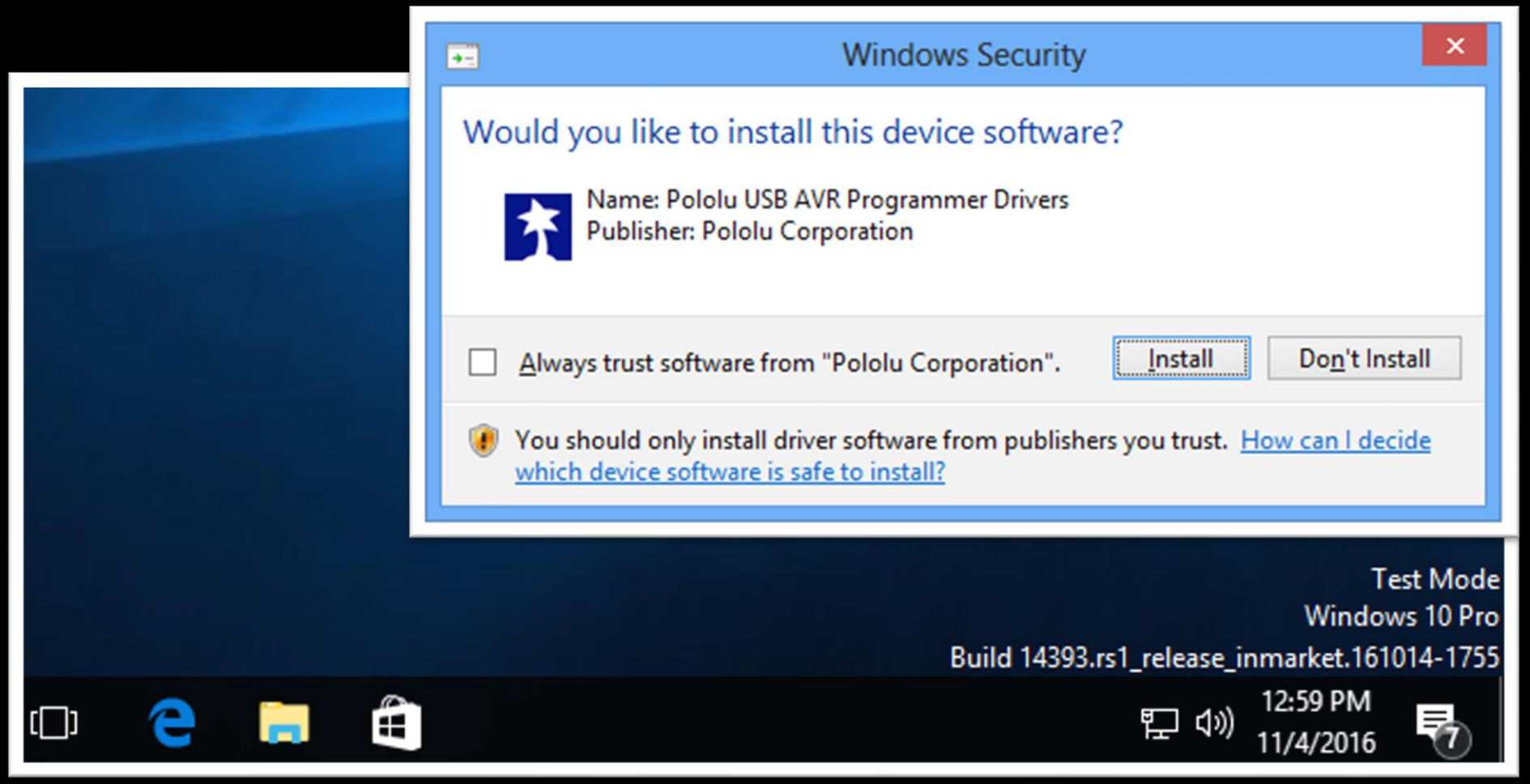
The Windows Driver Frameworks (WDF)

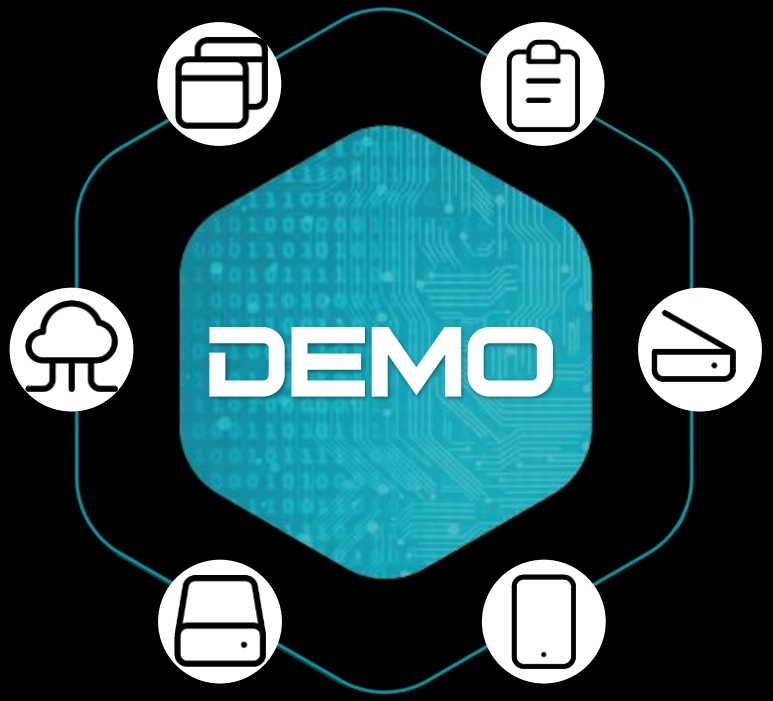
- A newer driver model, introduced in Windows Vista
  - Formerly called “Windows Driver Foundation”
  - Open source on Github
- WDF has two parts
  - KMDF – Kernel Mode Driver Framework
  - UMDF – User Mode Driver Framework
- KMDF is a replacement for WDM
  - Back-ported up to Windows 2000
  - Consistent object-based model
    - Properties, methods and events
  - Boilerplate Plug & Play and Power code implemented by the framework
    - Driver just needs to register for “interesting” events
  - Object lifetime management
  - Versioning with side by side support

مهسان
تکیه گاه شما
در دنیای هوشمند

Test Signing Mode

```
Bcdedit.exe -set TESTSIGNING ON
```





Testing a Simple Driver

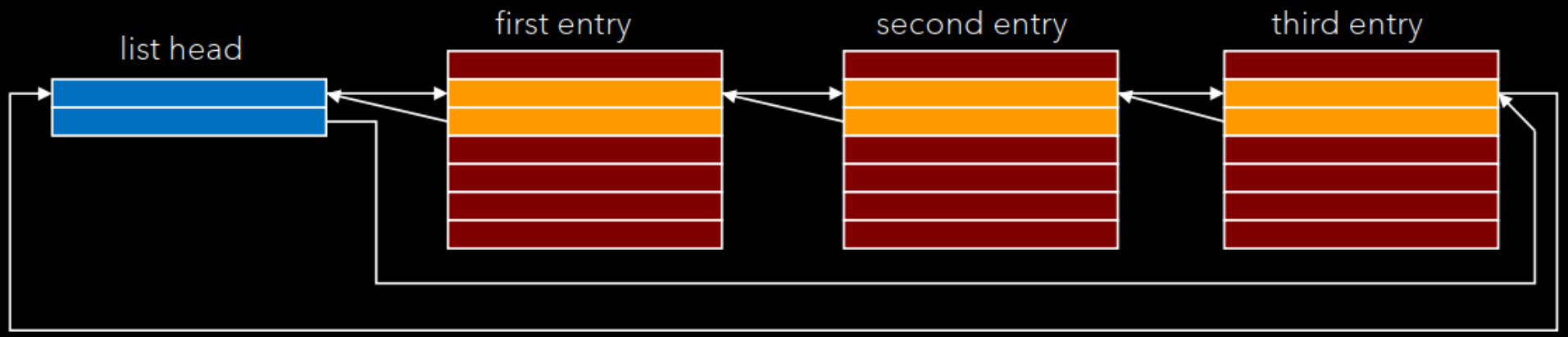
Some Concepts Before We Continue!

مهسان
تکیه گاه شما
در دنیای هوشمند

Kernel Doubly-Linked Lists

- Example list with three entries

```
typedef struct _LIST_ENTRY {
    struct _LIST_ENTRY *Flink;
    struct _LIST_ENTRY *Blink;
} LIST_ENTRY, *PLIST_ENTRY;
```



- To get to the larger structure, use the `CONTAINING_RECORD` macro

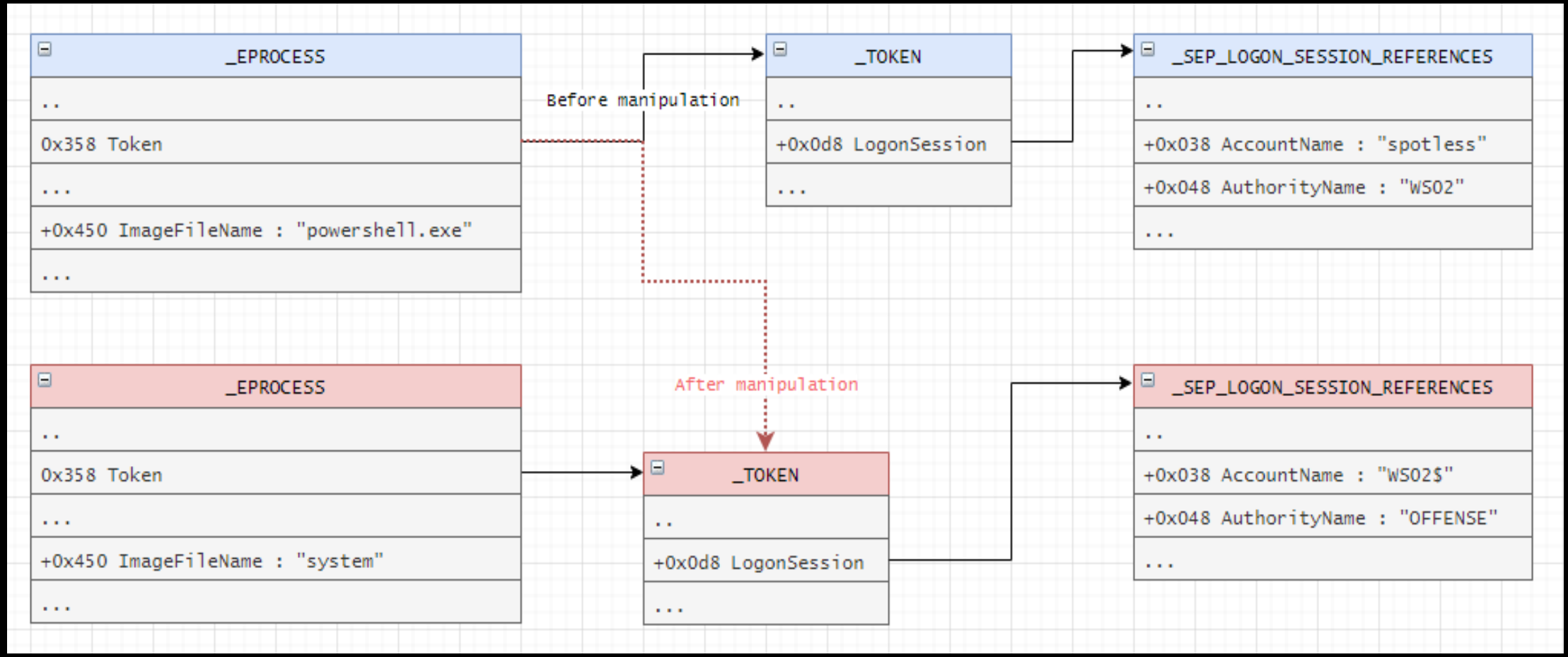
```
CONTAINING_RECORD(address, type, field)
```

Process Related Data Structures

- Kernel mode
  - `EPROCESS` is the executive process object
  - `KPROCESS` is the kernel process object
    - First member of `EPROCESS` named `Pcb` (Process Control Block)
  - All processes linked as a doubly linked-list
    - `LIST_ENTRY` member is `ActiveProcessLinks`
    - Root is `PsActiveProcessHead` kernel variable
- User mode
  - `PEB` (Process Environment Block)

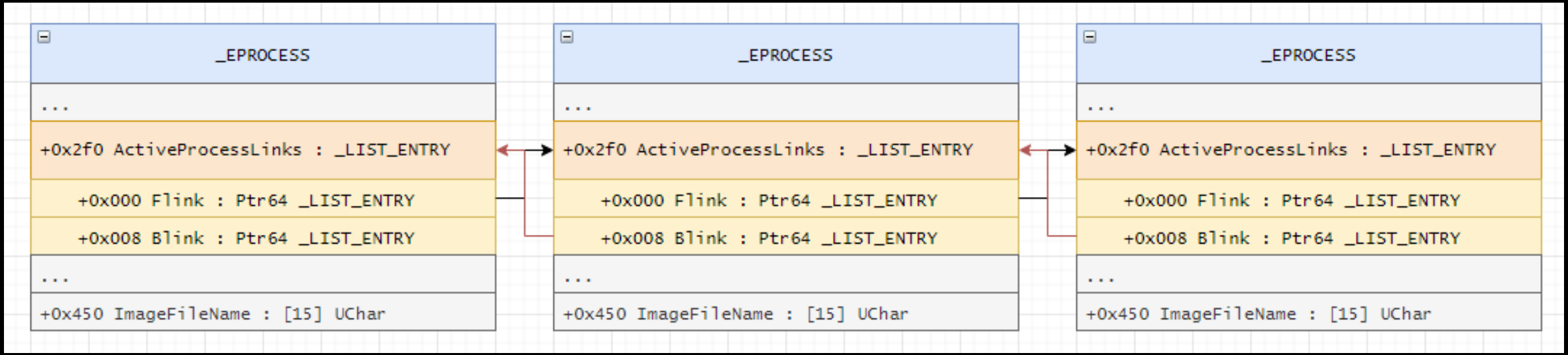
مهسان
تکیه گاه شما
در دنیای هوشمند

Token Stealing (WinDbg)



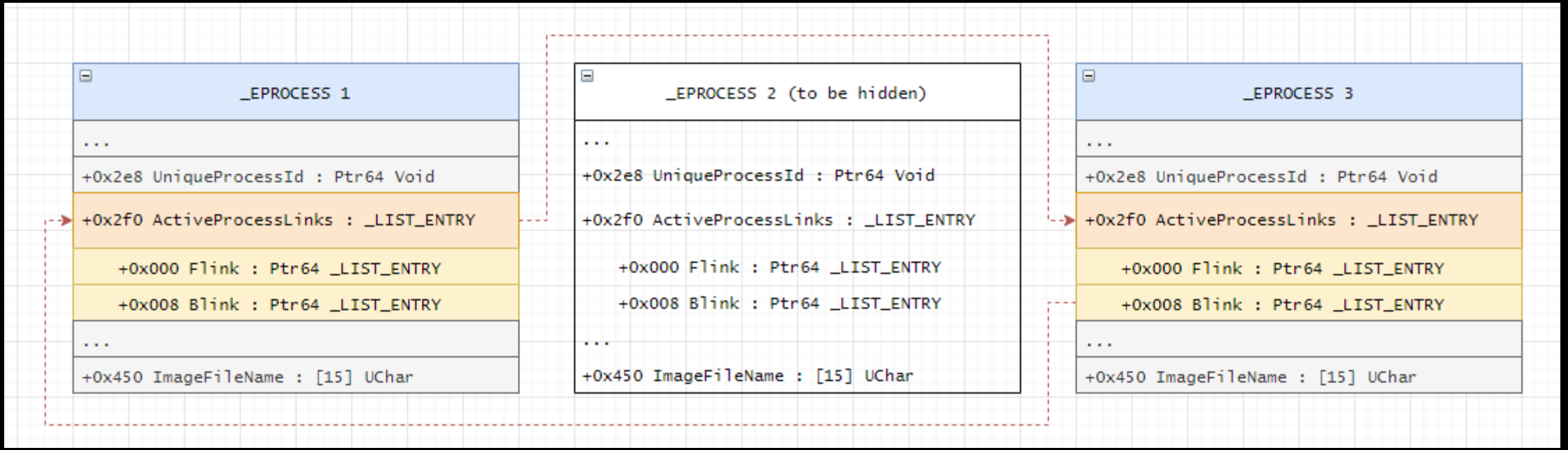
Source

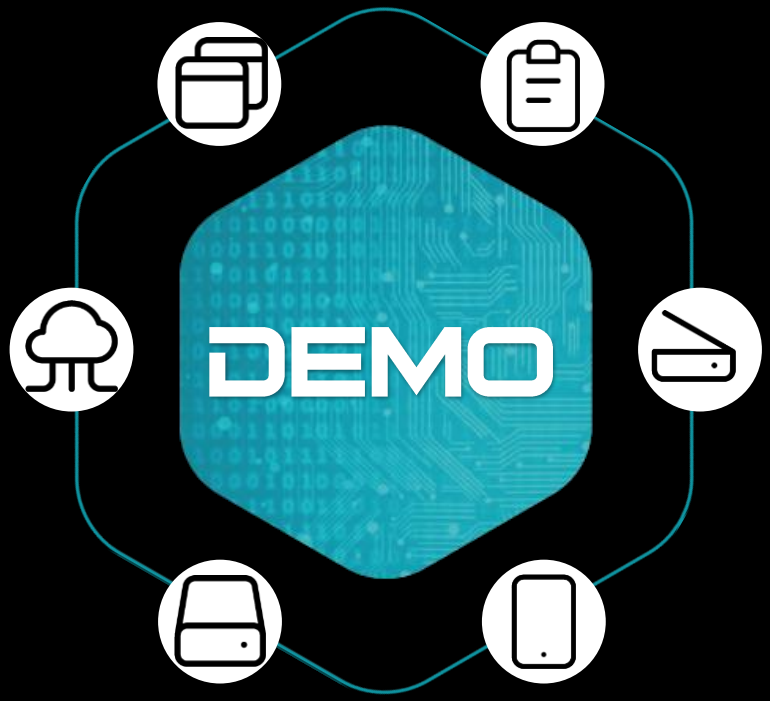
Manipulating ActiveProcessLinks (Before Change)



Source

Manipulating ActiveProcessLinks (AFTER Change)

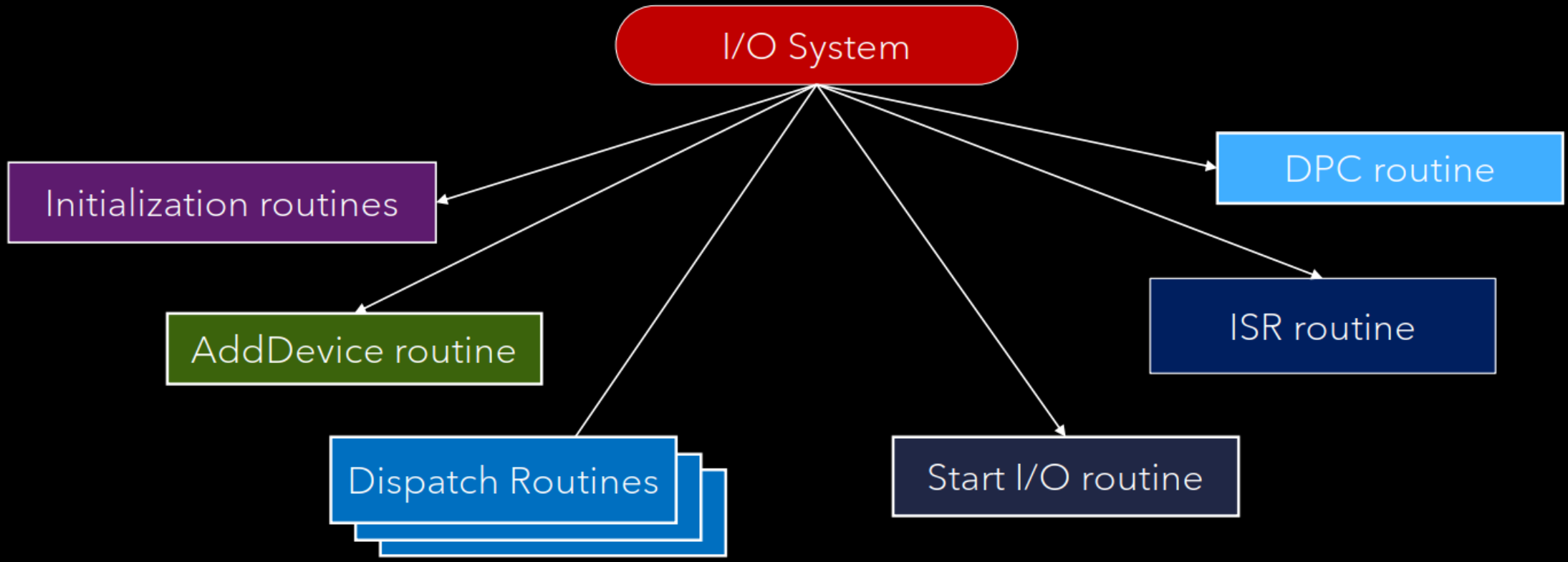




Examining Kernel LinkedLists & Process Information

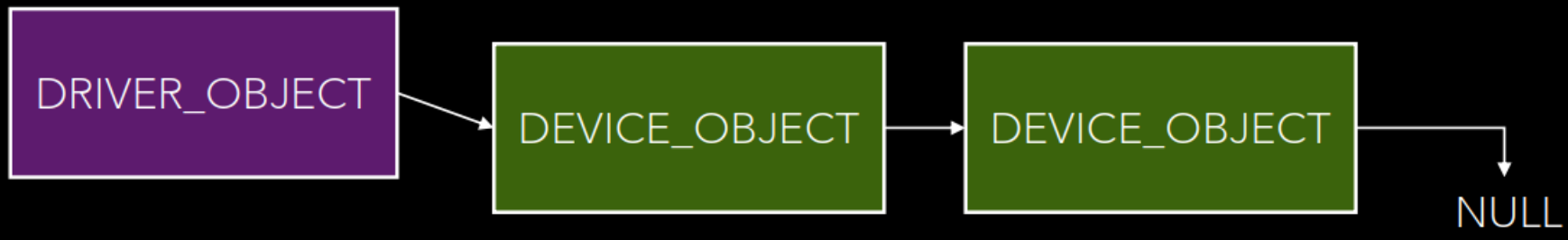
Anatomy of a Driver

- A driver exports functionality, callable by the I/O system



Driver and Device Objects

- Drivers are represented in memory using a **`DRIVER_OBJECT`** structure
  - Created by the I/O system
  - Provided to the driver in the **`DriverEntry`** function
  - Holds all exported functions
- Device objects are created by the driver on a per-device basis
  - Represented by the **`DEVICE_OBJECT`** structure
  - Typically created in the Driver's AddDevice routine
  - Several can be associated with a single driver object
- I/O system is device-centric, not driver-centric



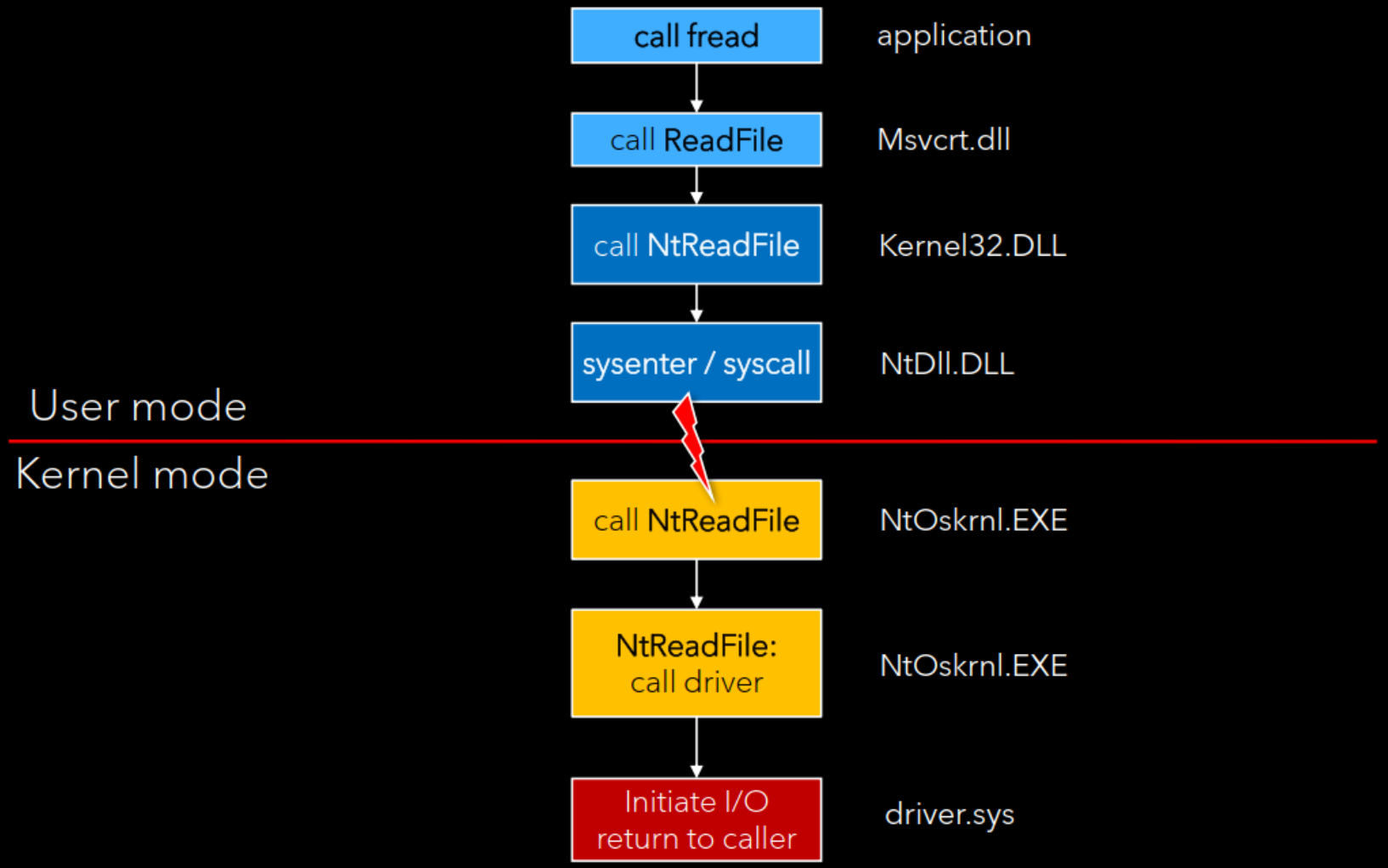
مهسان
تکیه گاه شما
در دنیای هوشمند

Accessing Devices

- A client that wants to communicate with a device, must open a handle to the device
  - `CreateFile` or `CreateFile2` from user mode
  - `ZwCreateFile` from kernel mode
- `CreateFile` accepts a "filename" which is actually a device symbolic link
  - "file" being just one specific case
  - For devices, the name should have the format \\.\name
    - Cannot access non-local device
    - Must use double backslashes "\\\\.\\name" in C/C++

مهسان
تکیه گاه شما
در دنیای هوشمند

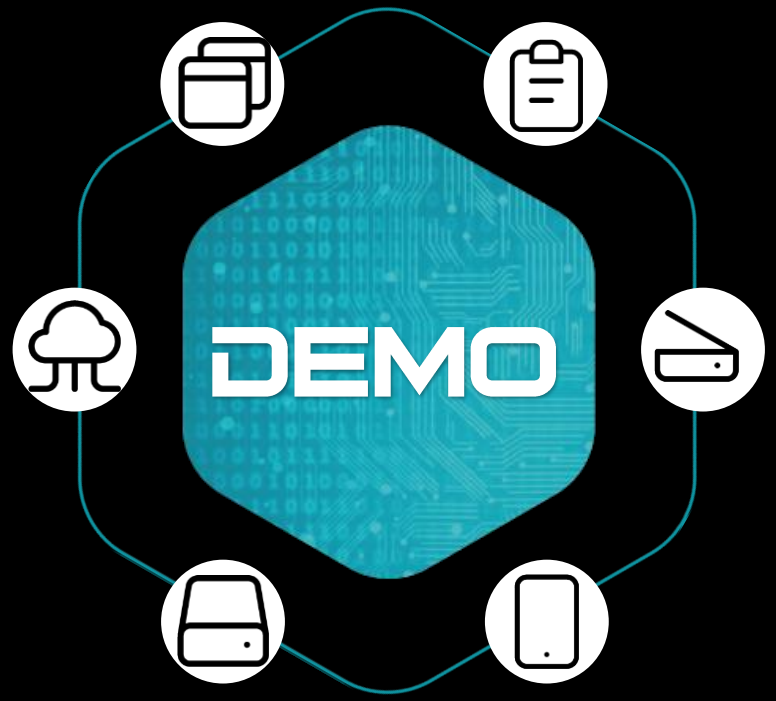
Invoking a Driver



I/O Request Packet (IRP)

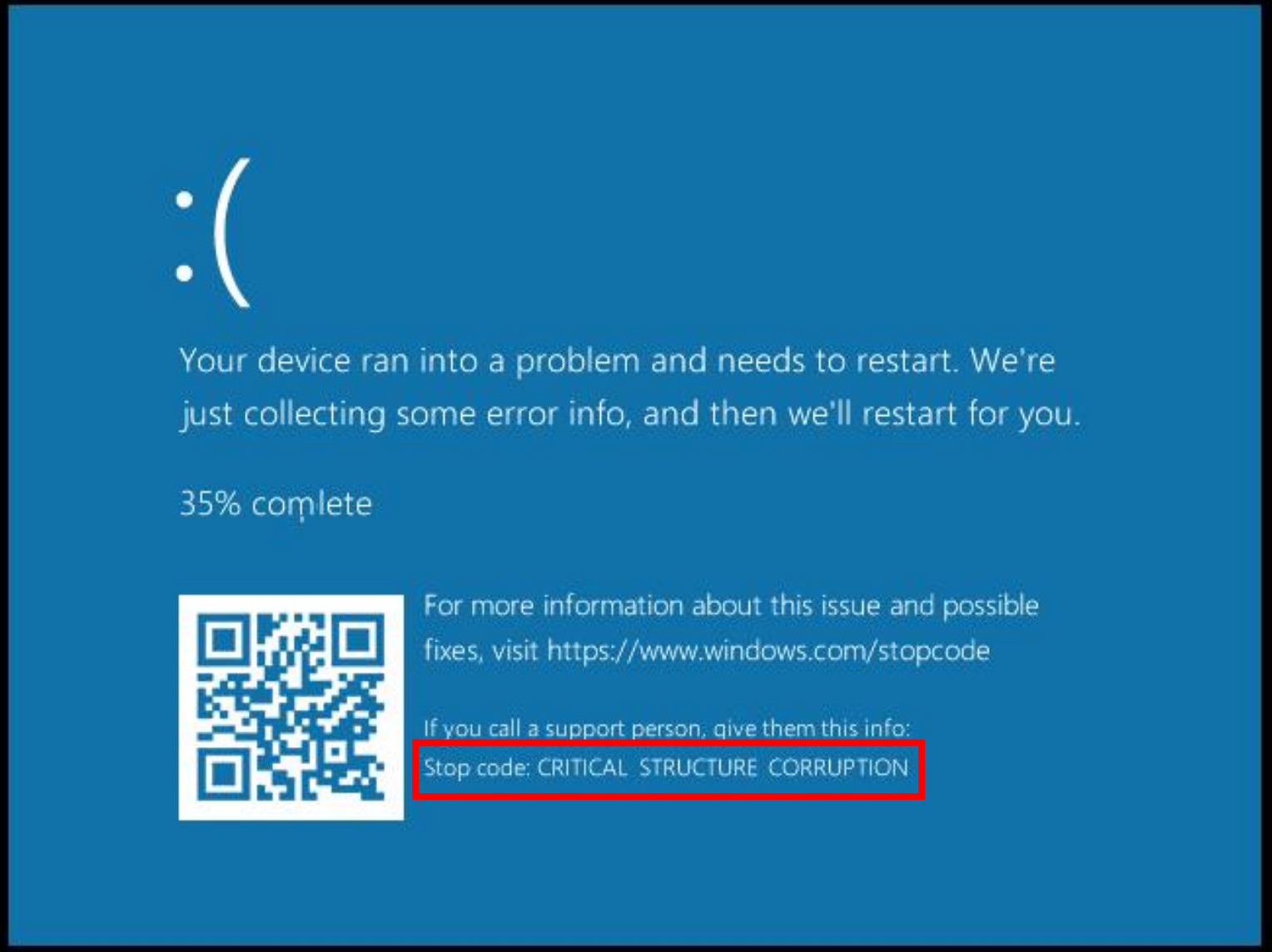
- A structure representing some request
  - Represented by the IRP structure
  - Contains all details needed to handle the request (codes, buffers, sizes, etc.)
- Always allocated from non-paged pool
- Accompanied by a set of structures of type `IO_STACK_LOCATION`
  - Number of structures is the number of the devices in this DevNode
  - Complements the data in the IRP
- IRPs are typically created by the I/O Manager, P&P Manager or the Power Manager
  - Can be explicitly created by drivers as well

مهسان
تکیه گاه شما
در دنیای هوشمند

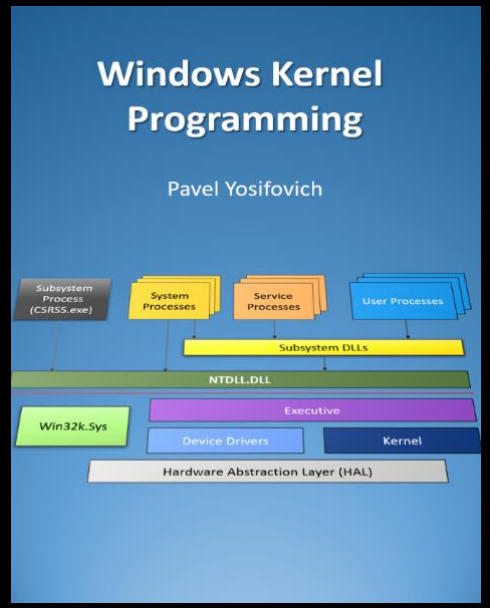


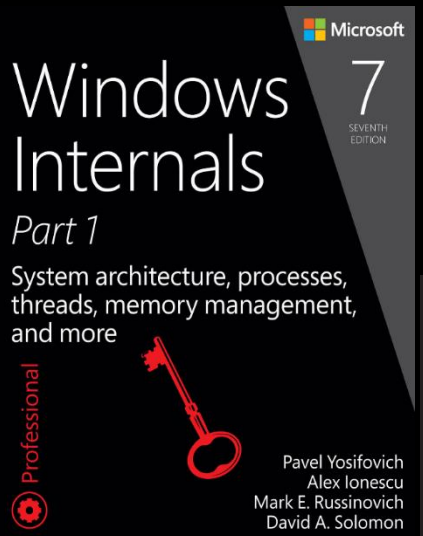
A Simple Kernel Rootkit!!!

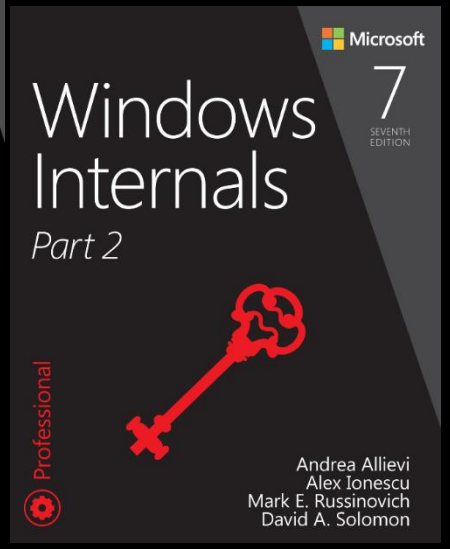
What is Patch Guard?

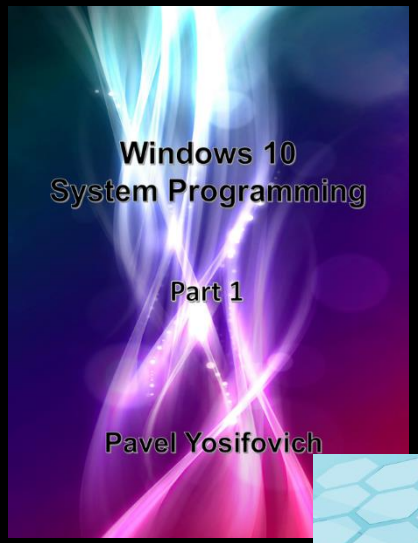


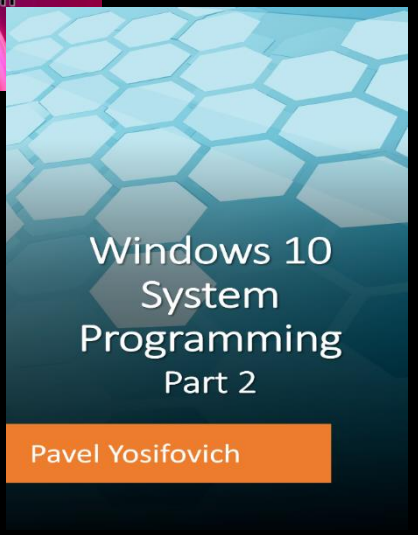
Further Readings











BTW, Most of the slides are from Pavel Yosifovich


THANK
YOU