RSAConference2022
San Francisco & Digital | June 6 – 9

TRANSFORM

SESSION ID: **HT-T01**

Bypassing Windows Hello for Business and Pleasure

Omer Tsarfati

Security Researcher
CyberArk Labs
@OmerTsarfati

WHOAMI

CYBERARK®

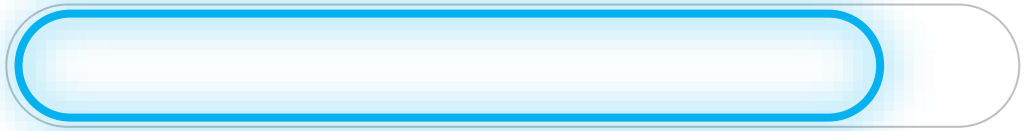
- Omer Tsarfati
- Security Researcher @ CyberArk Labs
- Twitter - @OmerTsarfati
- 24 years old
- 7 years in vulnerability research
- Call of Duty fan

AGENDA


1st
2nd
3rd
4th
Passwordless
Windows Hello
Exploiting
What's Next

BY THE END OF THIS TALK

WINDOWS HELLO AND ITS FEATURS



85%

USB & UVC

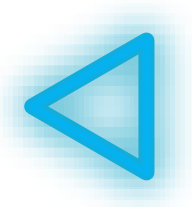
50%

CONFUSION

10%

FUN

MAX_INT%



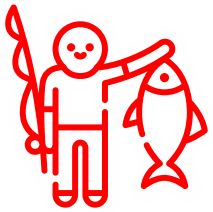
Part 1: Passwordless

and friends

WHY PASSWORDLESS

WEAK PASSWORDS

DATA BREACHES



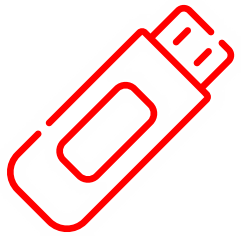
PHISHING ATTACKS

REUSE

HOW PASSWORDLESS

FINGERPRINT

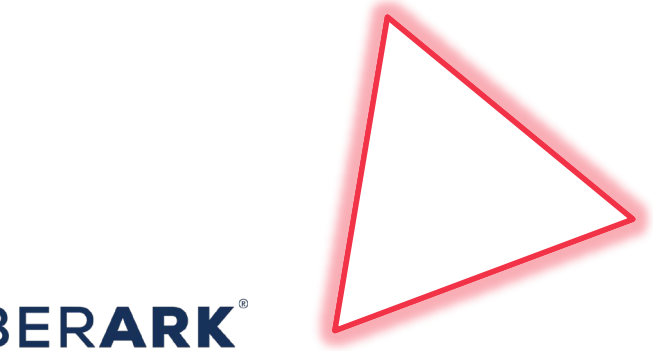
FACE RECGONITION



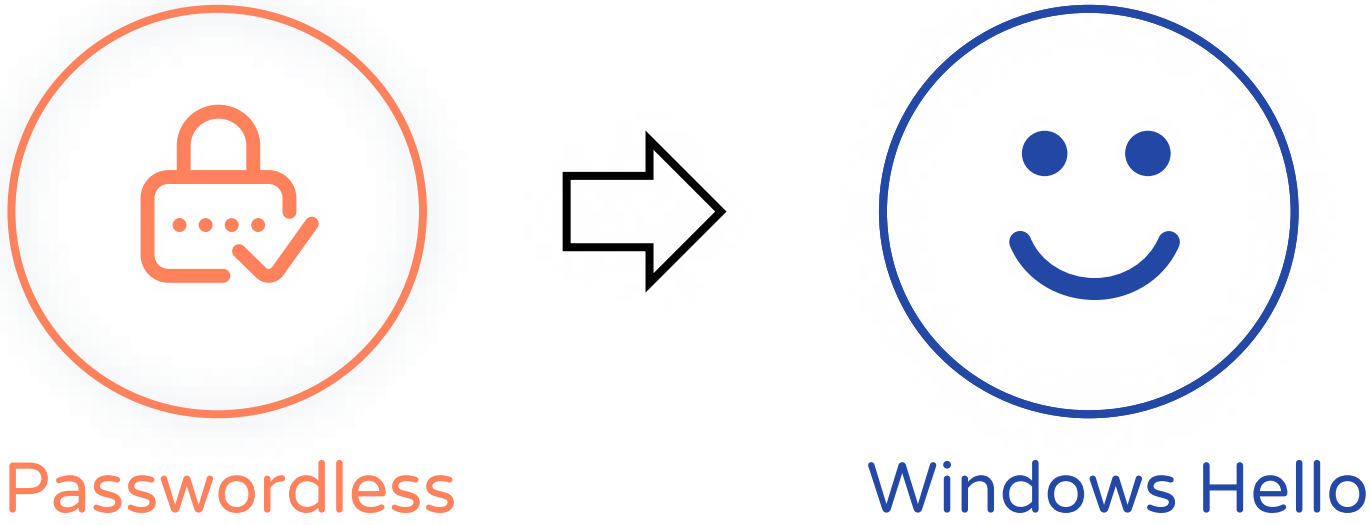
SECURITY KEY

OTP

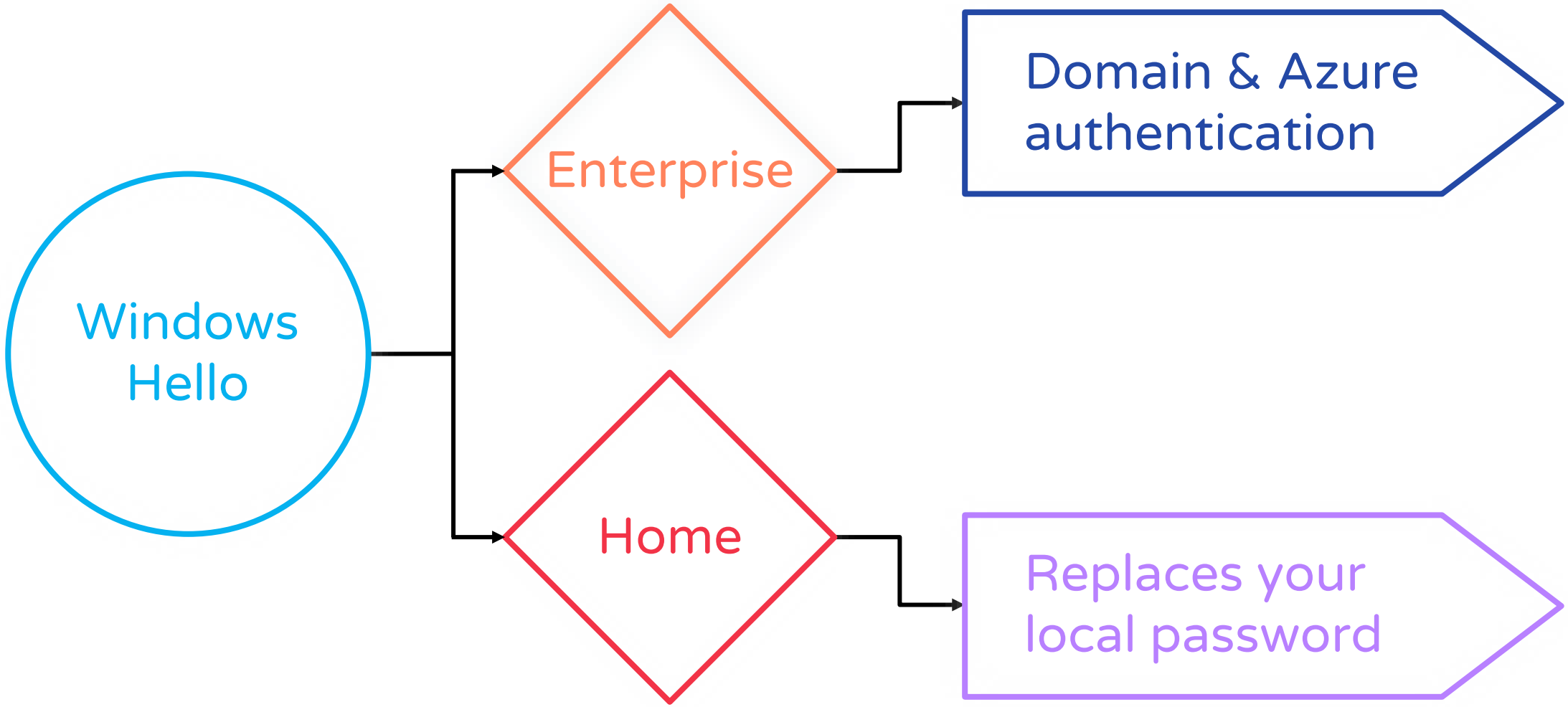
Part 2: Windows Hello



SAY _____ TO WINDOWS _____



EVERYONE CAN USE IT



FEATURES

FACE RECOGNITION

IR sensor and advanced image analysis

PIN CODE

And it is not the same as password

FINGERPRINT

Same good old fingerprint login

FINGERRINT

DATA

Challanging to capture and modify

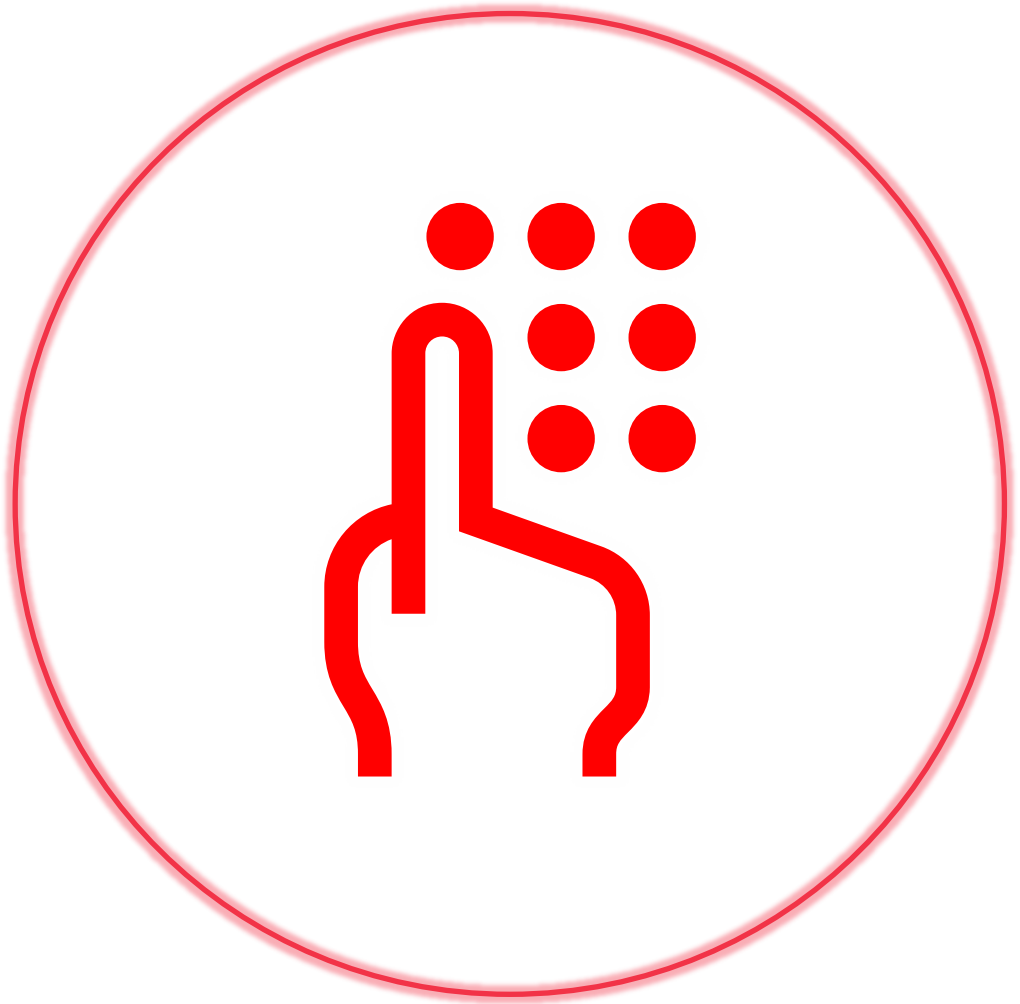
PREVIOUS ART

Many public reseraches

PROTOCOL

Complicated protocol and lack of public implmentation

PIN CODE



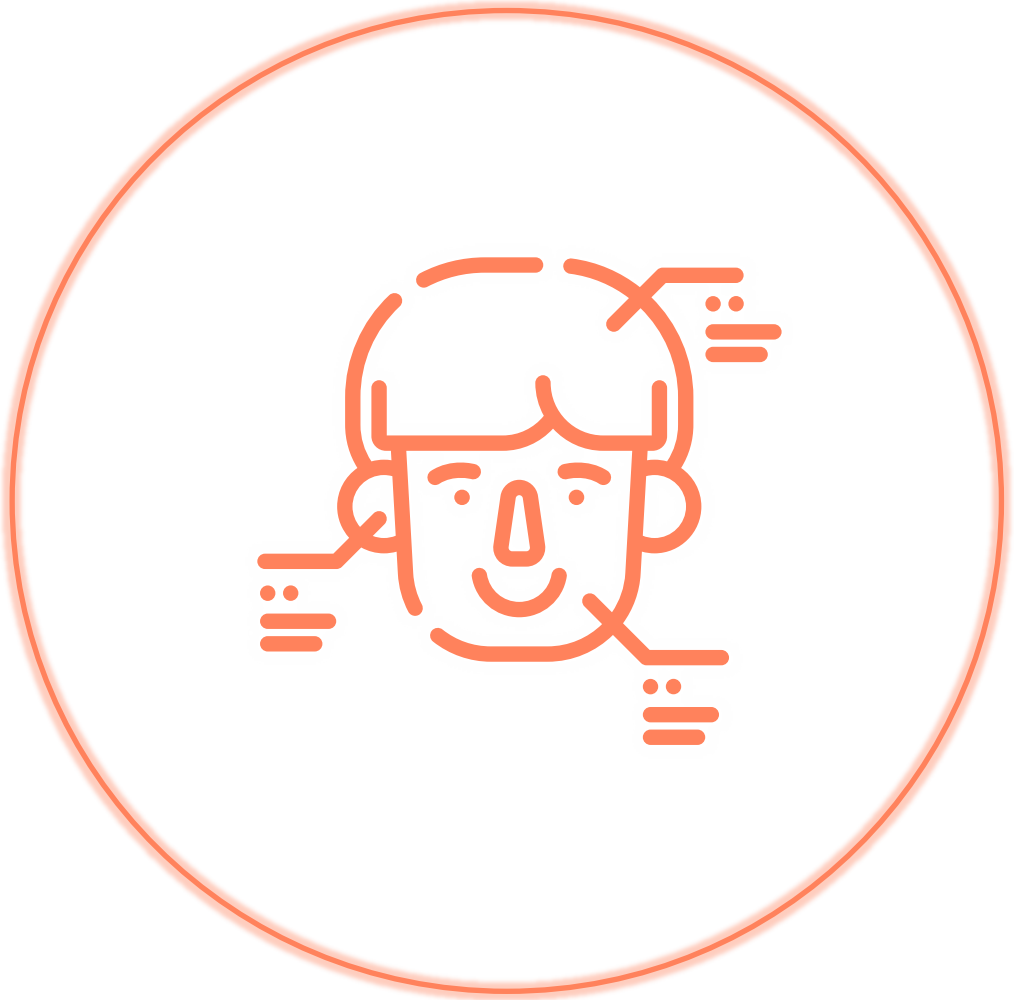
ANTI-BRUTEFORCE

Anti-bruteforce mechanism

SAME CHALLANGE

Same challenge as cracking a password

FACE RECOGNITION



ANTI-BRUTEFORCE

Anti-bruteforce mechanism

EASY START

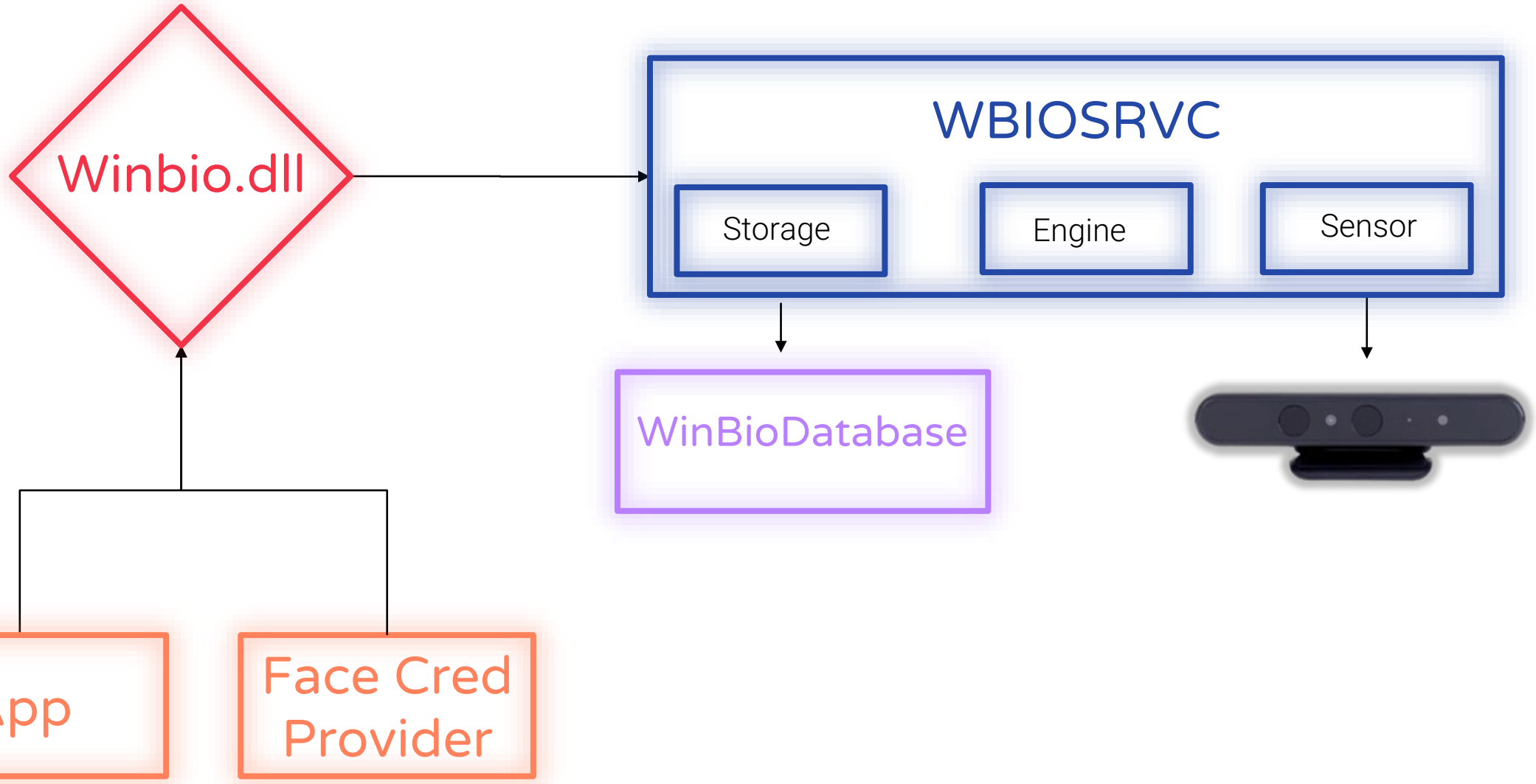
NXP implemented a basic
USB video camera

PUBLIC DATA

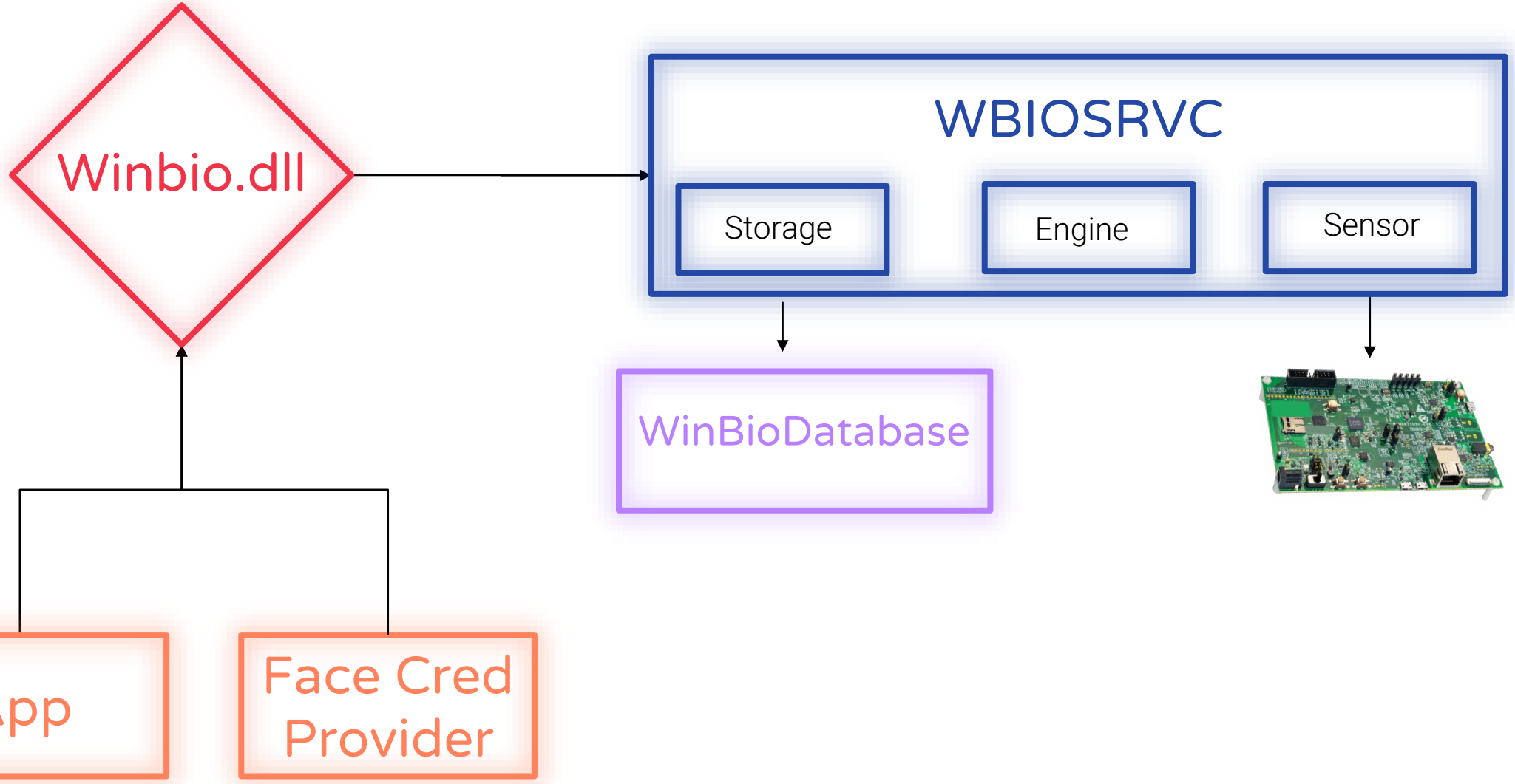
Face images can be captured
easily

FACE RECOGNITION

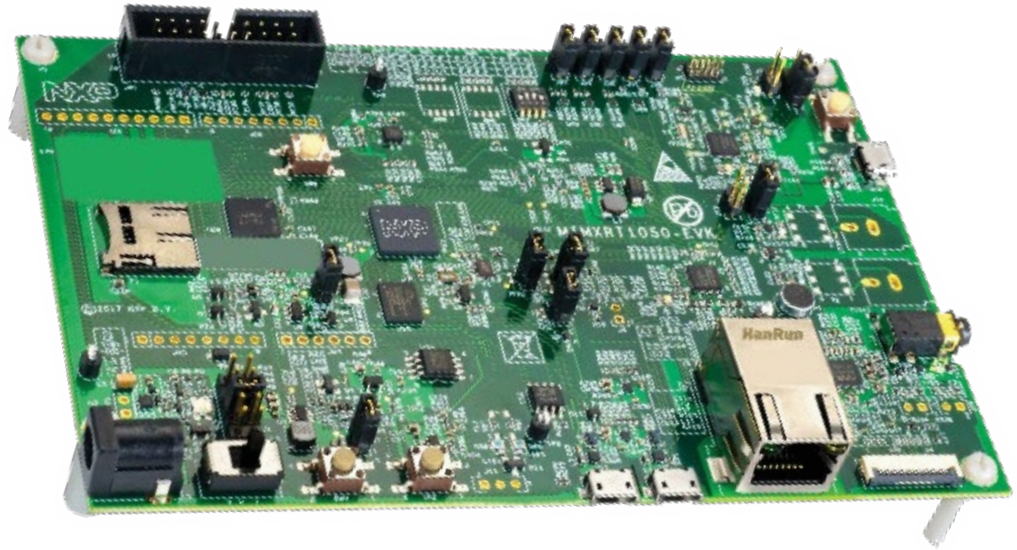
BIOMETRIC AUTH & WINDOWS HELLO



BIOMETRIC AUTH & WINDOWS HELLO



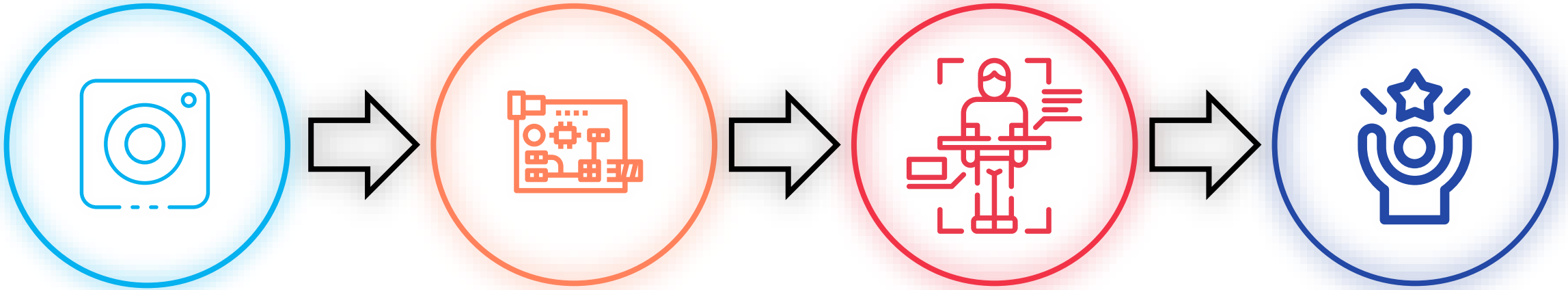
NXP – EVALUATION BOARD



Part 3: Exploiting

show me the money $$$

RESEARCH VECTOR



ANALYZE

USB & UVC

MIMIC

Mimics `hello` camera

CAPTURE

Capture the target person

EXPLOIT

Bypass Windows Hello

ANALYZE

ANALYZE – USB SPECS

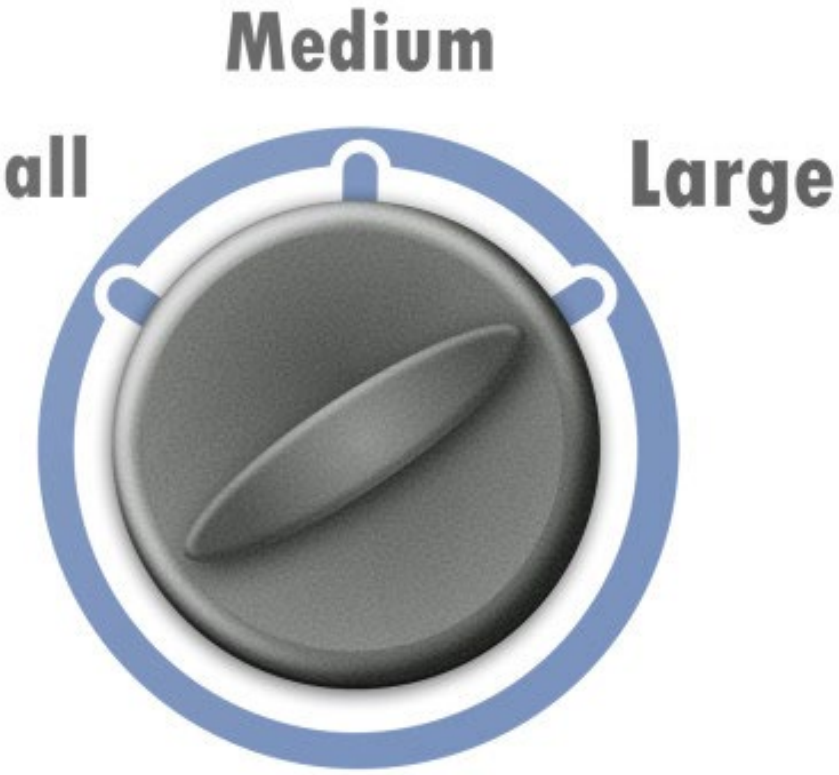
A USB logical device appears to the
endpoint sets that implement an int
manages the device using the Defa
bundles (associated with an endp
between a buffer on the host and
depending on transfer direction
coc when bus access i

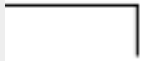
5.3.2.1 Stream P

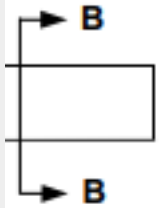
Stream pipes delive
the data content. D
pipes are always un

Data flowing throug
The USB System S
using the same strea
first-in, first-out.

A stream pipe to a
corresponding to an
opposite direction c



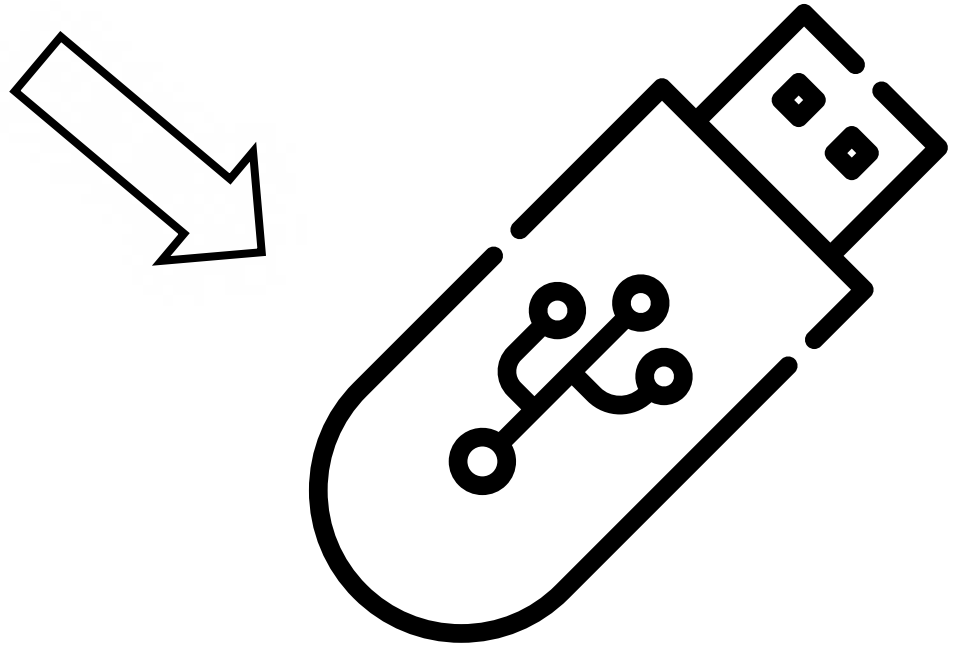




Cut End
B Device.)

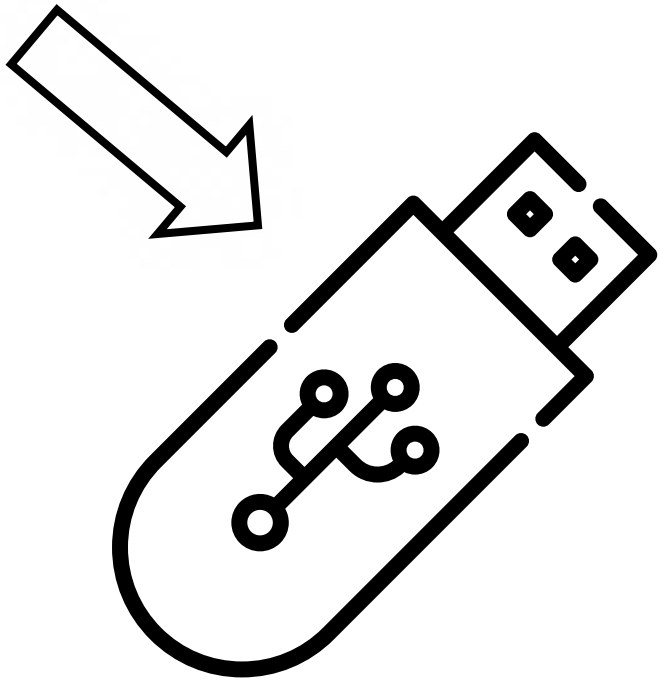
ANALYZE – USB STRUCTURE

NICE GUY



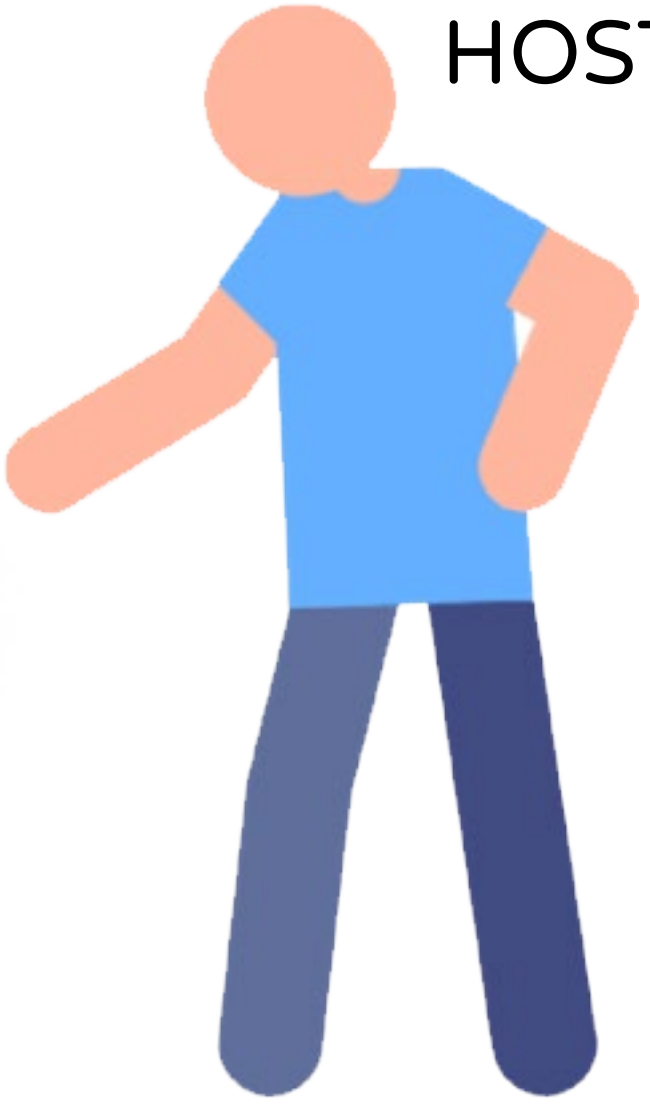
ANALYZE – USB STRUCTURE

NICE GUY

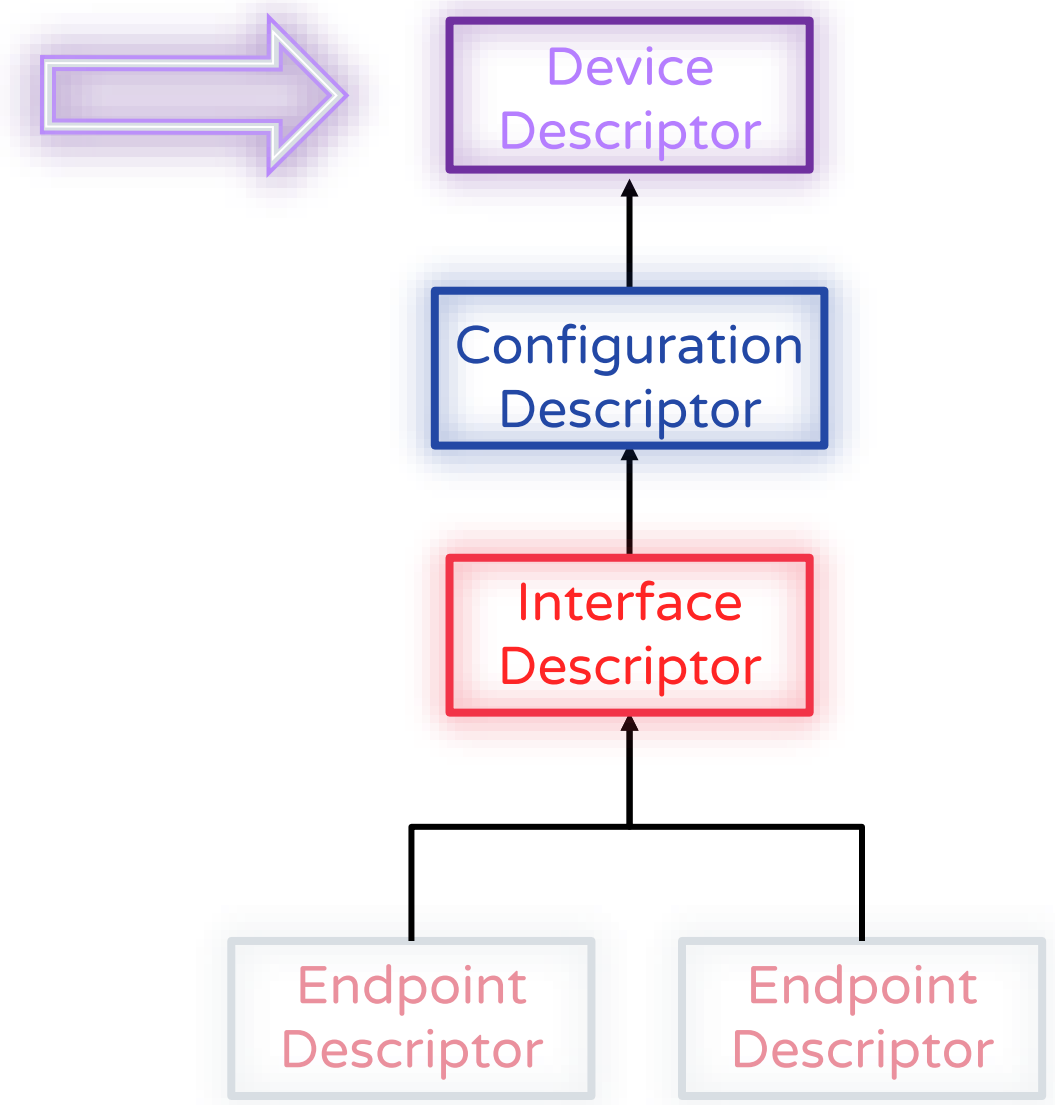


Bully

HOSTin



ANALYZE – USB STRUCTURE



Device Descriptor

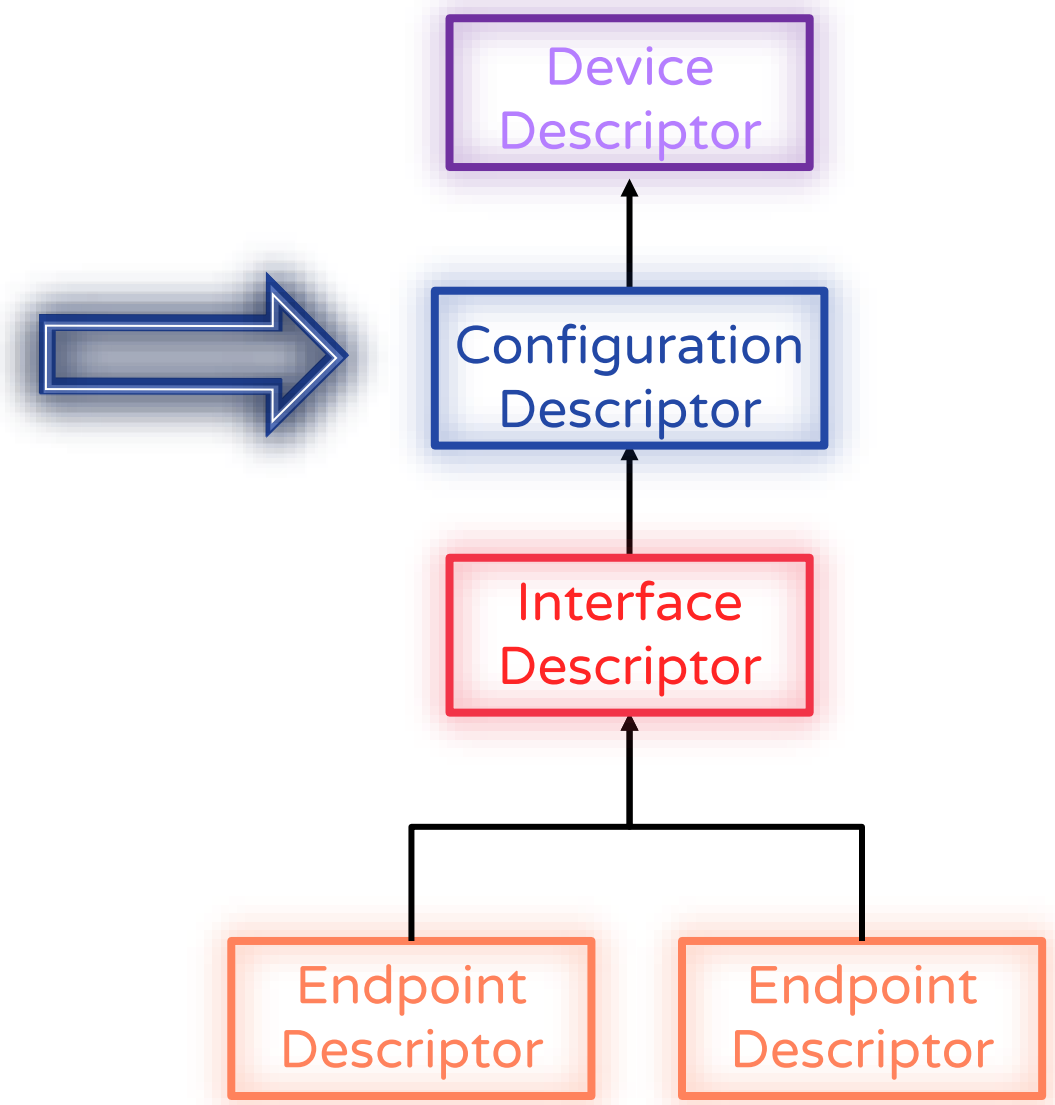
```
> Frame 1749: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface wireshark_extcap1796, id 0
> USB URB
v DEVICE DESCRIPTOR
    bLength: 18
    bDescriptorType: 0x01 (DEVICE)
    bcdUSB: 0x0201
    bDeviceClass: Miscellaneous (0xef)
    bDeviceSubClass: 2
    bDeviceProtocol: 1 (Interface Association Descriptor)
    bMaxPacketSize0: 64
    idVendor: Quanta Computer, Inc. (0x0408)
    idProduct: Unknown (0x7090)
    bcdDevice: 0x0011
    iManufacturer: 3
    iProduct: 1
    iSerialNumber: 2
    bNumConfigurations: 1
```

Device Descriptor

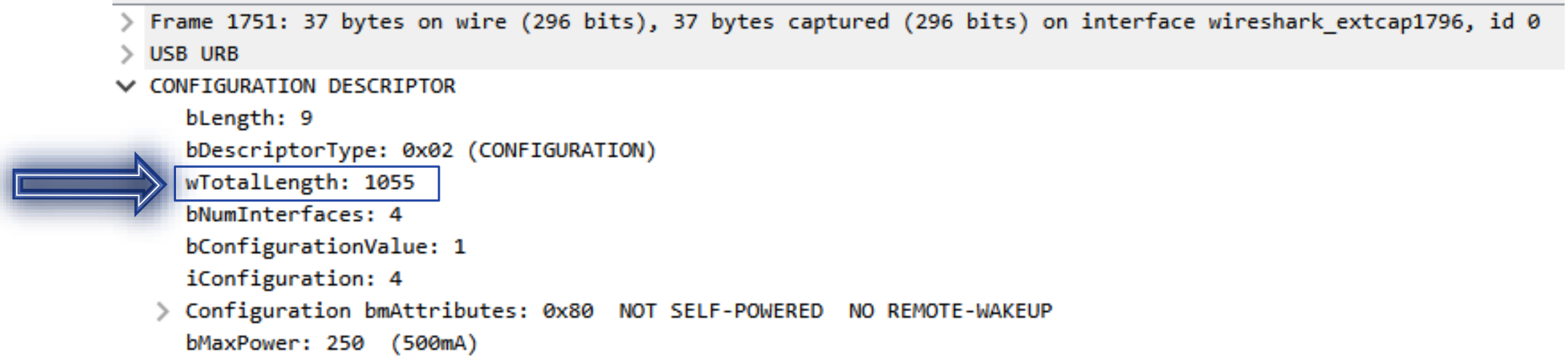
```
> Frame 1749: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface wireshark_extcap1796, id 0
> USB URB
v DEVICE DESCRIPTOR
    bLength: 18
    bDescriptorType: 0x01 (DEVICE)
    bcdUSB: 0x0201
    bDeviceClass: Miscellaneous (0xef)
    bDeviceSubClass: 2
    bDeviceProtocol: 1 (Interface Association Descriptor)
    bMaxPacketSize0: 64
    idVendor: Quanta Computer, Inc. (0x0408)
    idProduct: Unknown (0x7090)
    bcdDevice: 0x0011
    iManufacturer: 3
    iProduct: 1
    iSerialNumber: 2
    bNumConfigurations: 1
```

Device Descriptor

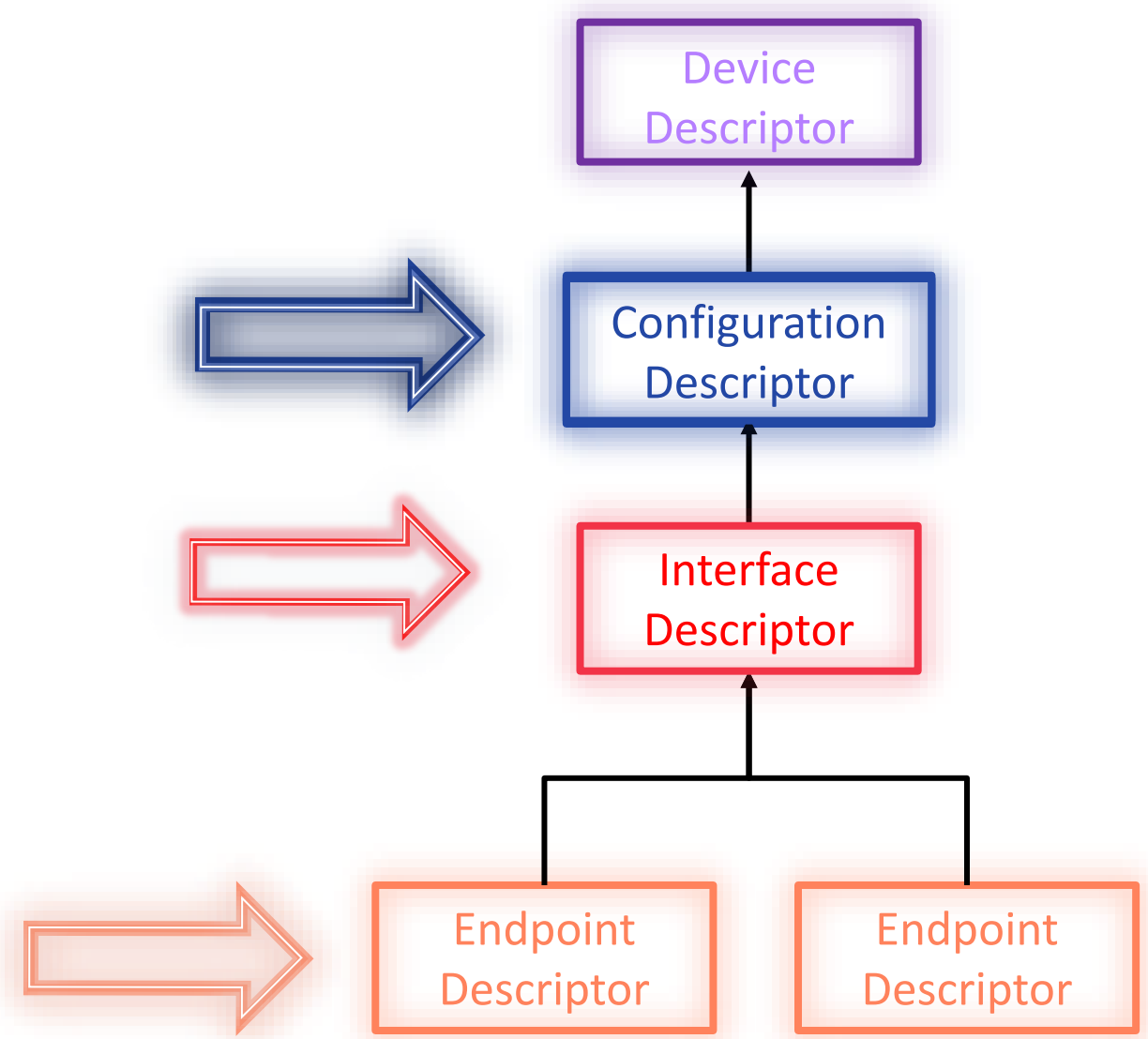
```
> Frame 1749: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface wireshark_extcap1796, id 0
> USB URB
v DEVICE DESCRIPTOR
    bLength: 18
    bDescriptorType: 0x01 (DEVICE)
    bcdUSB: 0x0201
    bDeviceClass: Miscellaneous (0xef)
    bDeviceSubClass: 2
    bDeviceProtocol: 1 (Interface Association Descriptor)
    bMaxPacketSize0: 64
    idVendor: Quanta Computer, Inc. (0x0408)
    idProduct: Unknown (0x7090)
    bcdDevice: 0x0011
    iManufacturer: 3
    iProduct: 1
    iSerialNumber: 2
    bNumConfigurations: 1
```


Device Descriptor
Configuration Descriptor
Interface Descriptor
Endpoint Descriptor
Endpoint Descriptor

Configuration Descriptor



ANALYZE – USB STRUCTURE



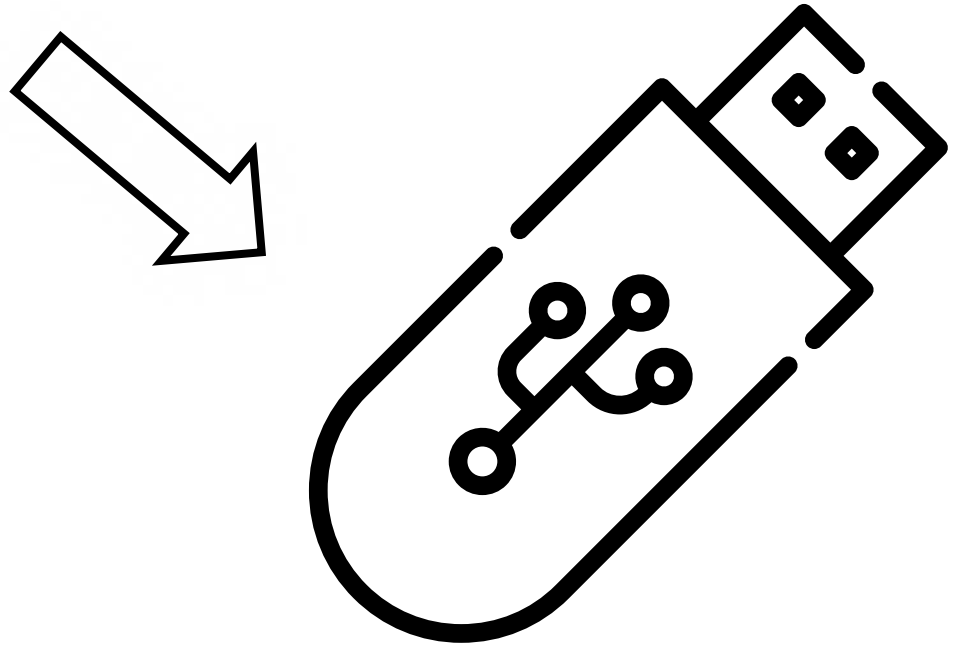
ANALYZE – USB STRUCTURE

Configuration Descriptor

```
> Frame 1753: 1083 bytes on wire (8664 bits), 1083 bytes captured (8664 bits) on interface wireshark_extcap1796, id 0
> USB URB
> CONFIGURATION DESCRIPTOR
> INTERFACE ASSOCIATION DESCRIPTOR
> INTERFACE DESCRIPTOR (0.0): class Video
> VIDEO CONTROL INTERFACE DESCRIPTOR [Header]
> VIDEO CONTROL INTERFACE DESCRIPTOR [Input Terminal] (Entity 1)
> VIDEO CONTROL INTERFACE DESCRIPTOR [Processing Unit] (Entity 2)
> VIDEO CONTROL INTERFACE DESCRIPTOR [Output Terminal] (Entity 3)
> VIDEO CONTROL INTERFACE DESCRIPTOR [Extension Unit] (Entity 4)
> VIDEO CONTROL INTERFACE DESCRIPTOR [Extension Unit] (Entity 6)
> ENDPOINT DESCRIPTOR
> VIDEO CONTROL ENDPOINT DESCRIPTOR [Interrupt]
> INTERFACE DESCRIPTOR (1.0): class Video
> VIDEO STREAMING INTERFACE DESCRIPTOR [Input Header]
> VIDEO STREAMING INTERFACE DESCRIPTOR [Format MJPEG]  (Format 1)
> VIDEO STREAMING INTERFACE DESCRIPTOR [Frame MJPEG]   (Index  1):  640 x  360
> VIDEO STREAMING INTERFACE DESCRIPTOR [Frame MJPEG]   (Index  2):  640 x  480
> VIDEO STREAMING INTERFACE DESCRIPTOR [Frame MJPEG]   (Index  3):  848 x  480
> VIDEO STREAMING INTERFACE DESCRIPTOR [Frame MJPEG]   (Index  4):  960 x  540
> VIDEO STREAMING INTERFACE DESCRIPTOR [Frame MJPEG]   (Index  5): 1280 x  720
> VIDEO STREAMING INTERFACE DESCRIPTOR [Frame MJPEG]   (Index  6):  640 x  360
> VIDEO STREAMING INTERFACE DESCRIPTOR [Colorformat]
```

ANALYZE – USB STRUCTURE

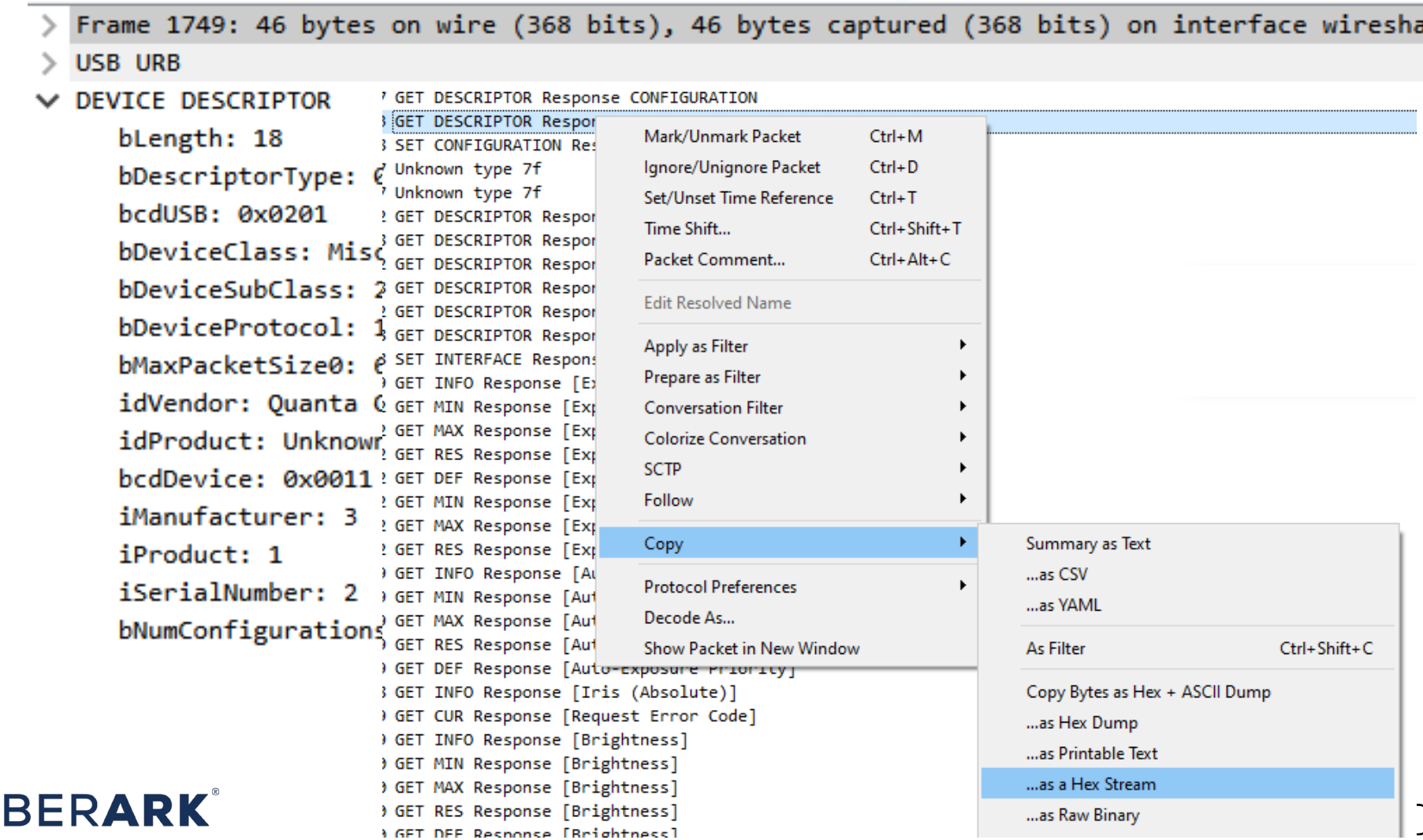
NICE GUY



Wireshark to the rescue!

```
> Frame 1749: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface wireshark_extcap1796, id 0
> USB URB
v DEVICE DESCRIPTOR
    bLength: 18
    bDescriptorType: 0x01 (DEVICE)
    bcdUSB: 0x0201
    bDeviceClass: Miscellaneous (0xef)
    bDeviceSubClass: 2
    bDeviceProtocol: 1 (Interface Association Descriptor)
    bMaxPacketSize0: 64
    idVendor: Quanta Computer, Inc. (0x0408)
    idProduct: Unknown (0x7090)
    bcdDevice: 0x0011
    iManufacturer: 3
    iProduct: 1
    iSerialNumber: 2
    bNumConfigurations: 1
```

Wireshark to the rescue!

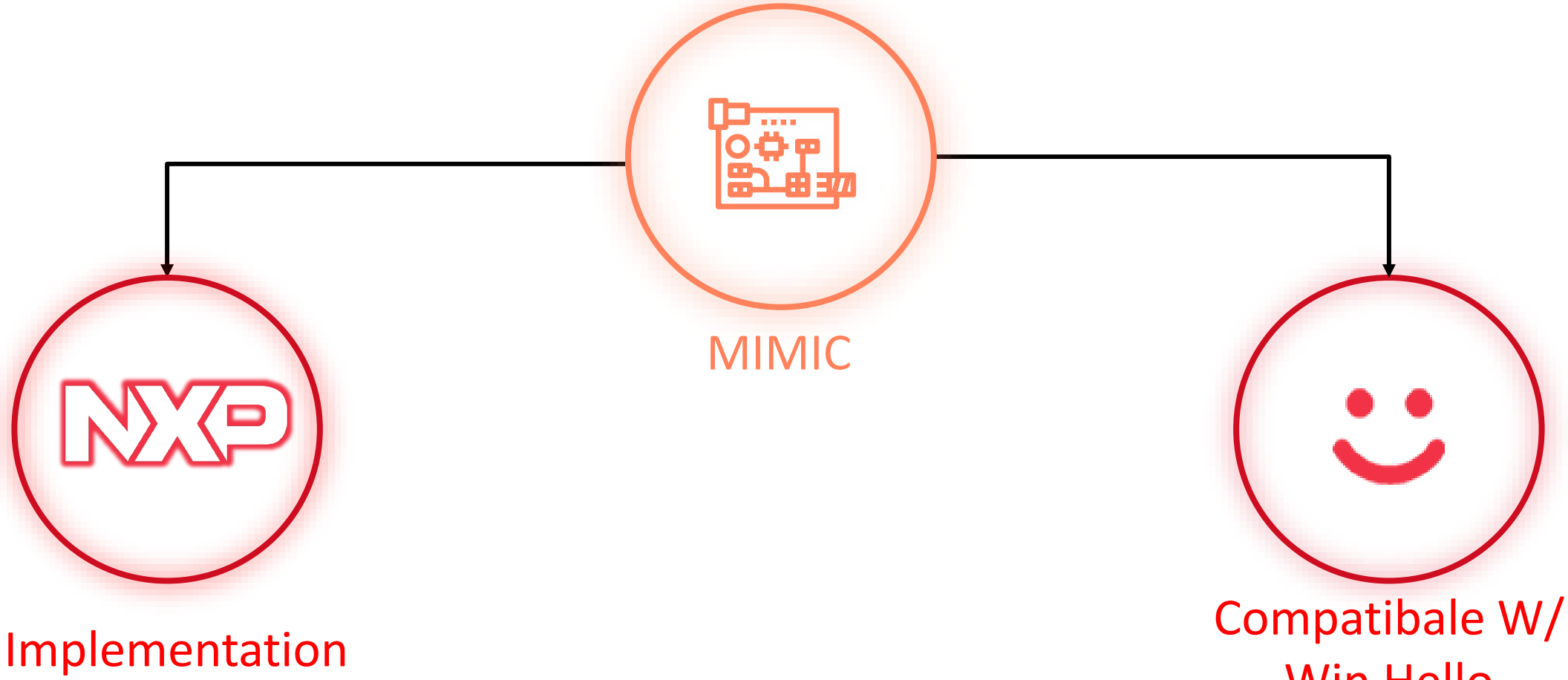


ANALYZE

USB Specs

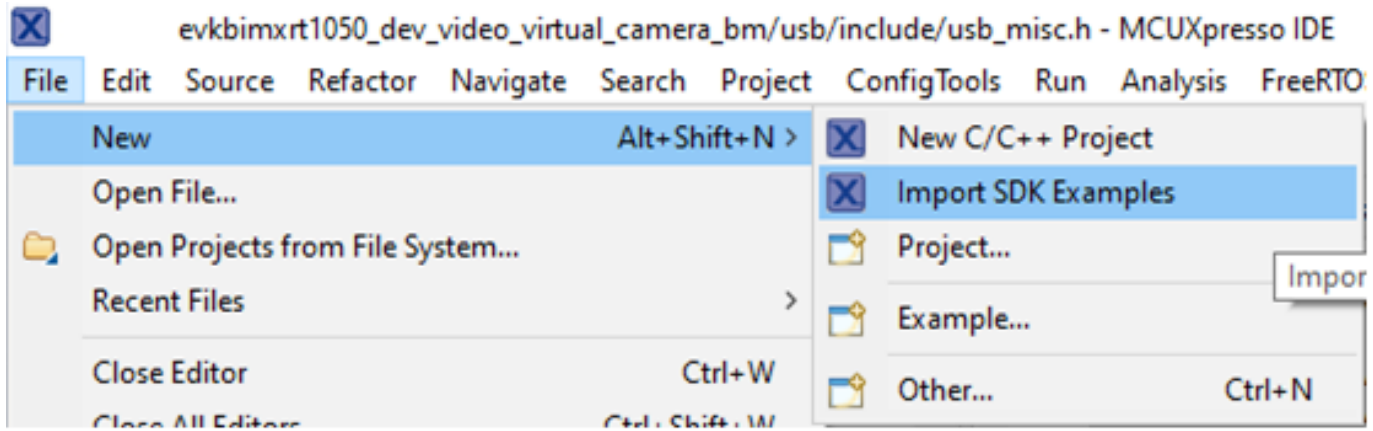
- USB 2.0 Specs
  - https://www.usb.org/document-library/usb-20-specification
- USB in a Nutshell
  - A summary of USB Chapter 9.5
  - https://www.beyondlogic.org/usbnutshell/usb5.shtml

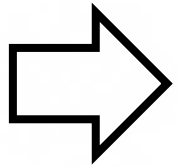
MIMIC

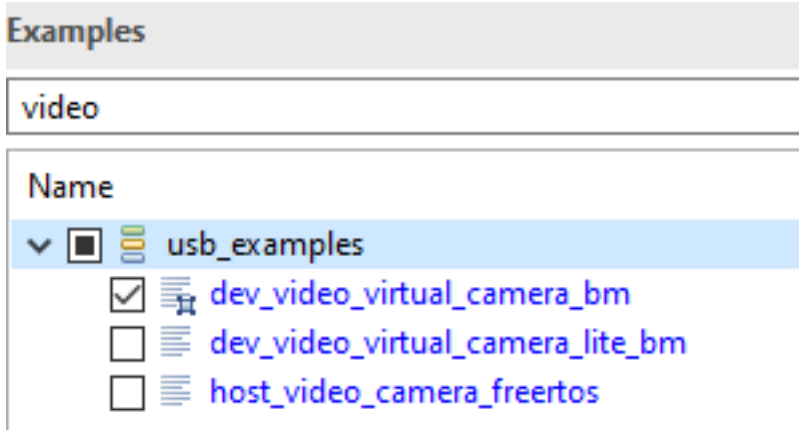


MIMIC

- Download NXP IDE
  - MCUXpresso
- Import an SDK example project
  - usb_examples -> dev_video_virtual_camera_bm



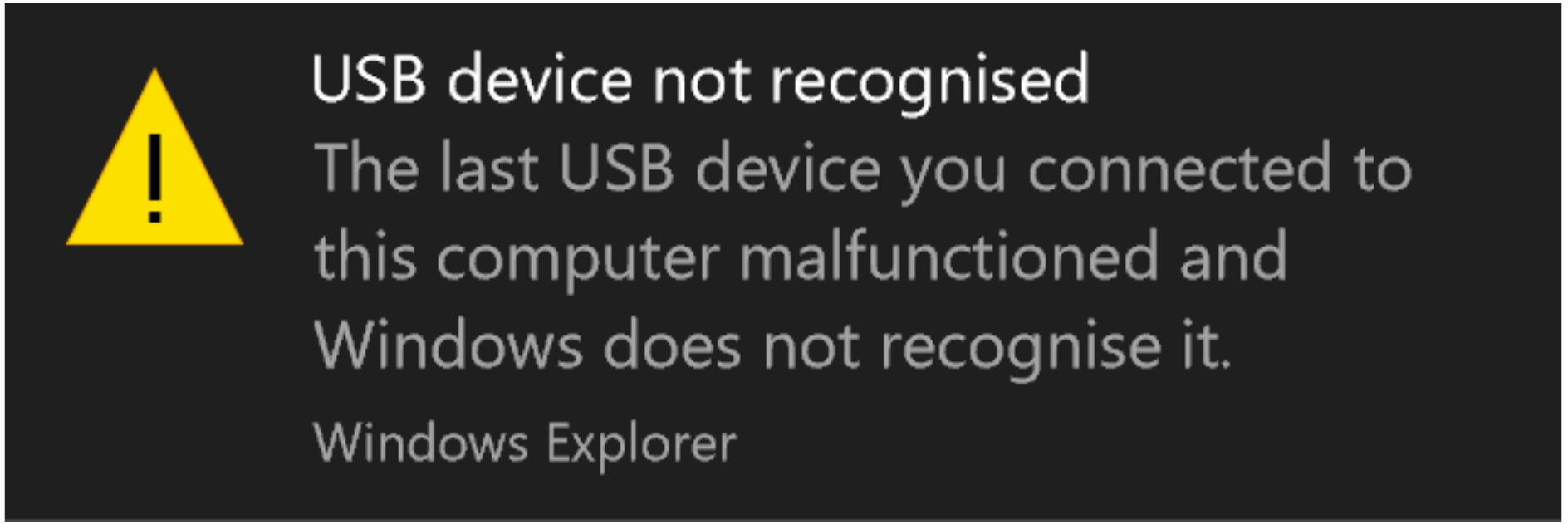




MIMIC

- Configure the example project with the extracted descriptors
- Added support for two cameras
- We connected the USB device to the computer AND….

MIMIC


USB device not recognised
The last USB device you connected to this computer malfunctioned and Windows does not recognise it.
Windows Explorer

MIMIC

BOS Descriptor Query and Validation

The USB 3.0 and USB 2.0 LPM specifications define a new USB descriptor called the Binary Device Object Store (BOS). The BOS descriptor returns a set of device-level capability descriptors for the USB device. For a USB device, which reports a **bcdUSB** value greater than 0x0200 in their device descriptor, in Windows 8, the USB driver stack requests for the BOS descriptor header immediately after the Language ID query request. If that request fails, the driver stack proceeds to the Product ID String query request.

If the request for the BOS descriptor header succeeds, the driver stack requests the entire BOS descriptor set by using the value returned in the BOS descriptor's **wTotalLength** field as the request length. If that request fails, the driver stack will fail enumeration of the device.

Note: To make sure your device enumerates on Windows 8 and future versions of Windows, do not rely on the sequence in which the USB driver stack queries for the BOS descriptor. Instead, make sure that the device reports correct values to pass validation checks as described in these sections.

MIMIC

BOS Descriptor Validation

The USB driver stack validates the retrieved BOS descriptor. Make sure that in your device:

- The number of bytes greater than or equal to the size of the BOS descriptor.
- **bDescriptorType** indicates the BOS Descriptor type (0x0F).
- **bLength** is the correct size of the BOS descriptor.
- **wTotalLength** is greater than or equal to the size of the BOS descriptor.
- **wTotalLength** is large enough to contain the number of capability descriptors reported in **bNumDeviceCaps** . The device computes the value by assuming a minimal 2-byte capability descriptor containing **bLength** and **bDescriptorType** fields, multiplied by the **bNumDeviceCaps** value, and added to the size of the BOS descriptor.
- **bNumDeviceCaps** is not zero.

If any of those validation checks fail, the driver stack will fail enumeration of the device. Otherwise the stack validates each Device Capability Descriptor.

MIMIC

BOS D

The US

- T
- b
- b
- w
- w
- b

d in

ability

eviceCaps

If any c

erwise the

stack va

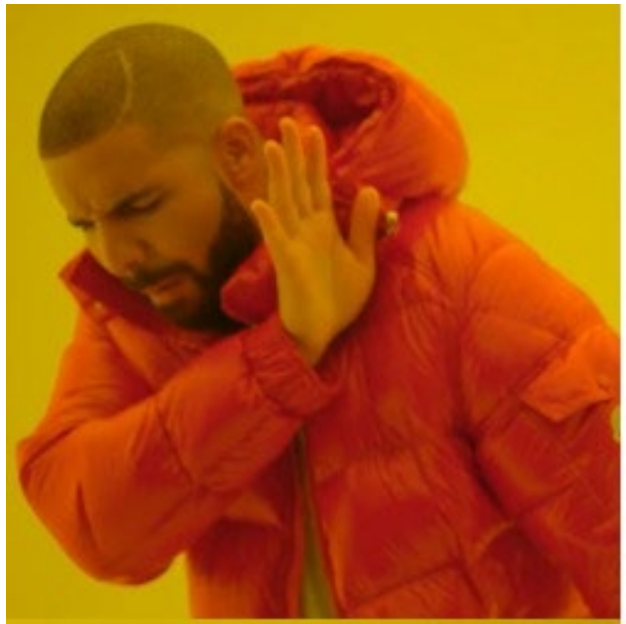
```
uint8_t USBVideoMSOS20DescriptorSet[USBVideoMSOS20DescriptorSetLength] =
{
    /* Microsoft OS 2.0 Descriptor Set Header */
    0x0A, 0x00,             // wLength of MSOS20_SET_HEADER_DESCRIPTOR
    0x00, 0x00,             // wDescriptorType == MSOS20_SET_HEADER_DESCRIPTOR
    0x00, 0x00, 0x00, 0x0A, // dwWindowsVersion - 0x10000000 for Windows 10
    USB_SHORT_GET_LOW(USBVideoMSOS20DescriptorSetLength),
    USB_SHORT_GET_HIGH(USBVideoMSOS20DescriptorSetLength),// wTotalLength - Total length 0x2C8 (712)

    /* Microsoft OS 2.0 Configuration Subset Header */
    0x08, 0x00,             // wLength of MSOS20_SUBSET_HEADER_CONFIGURATION
    0x01, 0x00,             // wDescriptorType == MSOS20_SUBSET_HEADER_CONFIGURATION
    0x00,                   // bConfigurationValue set to the first configuration
    0x00,                   // bReserved set to 0.
    //0xBE, 0x02,             // wTotalLength - Total length 0x2BE (702)
    USB_SHORT_GET_LOW(MicrosoftOS20ConfigurationSubsetHeaderLength) ,
    USB_SHORT_GET_HIGH(MicrosoftOS20ConfigurationSubsetHeaderLength),

    /***************Color Camera Function*****************/

    /* Microsoft OS 2.0 Function Subset Header */
    0x08, 0x00,             // wLength of MSOS20_SUBSET_HEADER_FUNCTION
    0x02, 0x00,             // wDescriptorType == MSOS20_SUBSET_HEADER_FUNCTION
    0x00,                   // bFirstInterface field of the first IAD
    0x00,                   // bReserved set to 0.
    //0x6E, 0x01,             // wSubsetLength - Length 0x16E (366)
    USB_SHORT_GET_LOW(MicrosoftOS20FunctionSubsetColorCameraLength),
    USB_SHORT_GET_HIGH(MicrosoftOS20FunctionSubsetColorCameraLength),
    /***************Register the Color Camera in a sensor group*****************/

    /* Microsoft OS 2.0 Registry Value Feature Descriptor */
    USB_SHORT_GET_LOW(MicrosoftOS20RegistryValueFeatureDescriptorColorCameraFSSensorGroupId),
    USB_SHORT_GET_HIGH(MicrosoftOS20RegistryValueFeatureDescriptorColorCameraFSSensorGroupId),
    //0x80, 0x00,             // wLength 0x80 (128) in bytes of this descriptor
    0x04, 0x00,             // wDescriptorType - MSOS20_FEATURE_REG_PROPERTY
    0x01, 0x00,             // wPropertyDataType - REG_SZ
    0x28, 0x00,             // wPropertyNameLength - 0x28 (40) bytes
    'U', 0x00, 'V', 0x00,   // Property Name - "UVC-FSSensorGroupID"
    'C', 0x00, '-', 0x00,
    'F', 0x00, 'S', 0x00,
    'S', 0x00, 'e', 0x00,
    'n', 0x00, 's', 0x00,
    'o', 0x00, 'r', 0x00,
    'G', 0x00, 'r', 0x00,
    'o', 0x00, 'u', 0x00,
    'p', 0x00, 'I', 0x00,
    'D', 0x00, 0x00, 0x00,
    0x4E, 0x00,             // wPropertyDataLength - 0x4E (78) bytes
                            // FSSensorGroupID GUID in string format:
                            // "{20C94C5C-F402-4F1F-B324-0C1CF0257870}"
    '{', 0x00, '6', 0x00,
```

MIMIC

No.	Time	Source	Destination	Protocol	Lengtl	Info
1748	42.579512	host	1.16.0	USB	36	GET DESCRIPTOR Request DEVICE
1749	42.579548	1.16.0	host	USB	46	GET DESCRIPTOR Response DEVICE
1750	42.579556	host	1.16.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
1751	42.579557	1.16.0	host	USB	37	GET DESCRIPTOR Response CONFIGURATION
1752	42.579559	host	1.16.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
1753	42.579560	1.16.0	host	USB	1083	GET DESCRIPTOR Response CONFIGURATION
1754	42.581664	host	1.16.0	USB	36	SET CONFIGURATION Request
1755	42.583735	1.16.0	host	USB	28	SET CONFIGURATION Response
1756	42.583830	host	1.16.0	USB	27	Unknown type 7f
1757	42.583833	1.16.0	host	USB	27	Unknown type 7f
1758	42.583843	host	1.16.0	USB	27	Unknown type 7f
1759	42.583845	1.16.0	host	USB	27	Unknown type 7f
1760	42.584862	host	1.16.0	USB	36	GET DESCRIPTOR Request STRING
1761	42.585424	1.16.0	host	USB	32	GET DESCRIPTOR Response STRING
1762	42.585453	host	1.16.0	USB	36	GET DESCRIPTOR Request STRING
1763	42.585989	1.16.0	host	USB	48	GET DESCRIPTOR Response STRING
1764	42.595665	host	1.16.0	USB	36	GET DESCRIPTOR Request STRING
1765	42.596145	1.16.0	host	USB	32	GET DESCRIPTOR Response STRING
1766	42.596199	host	1.16.0	USB	36	GET DESCRIPTOR Request STRING
1767	42.596697	1.16.0	host	USB	48	GET DESCRIPTOR Response STRING
1772	44.195114	host	1.16.0	USB	36	GET DESCRIPTOR Request STRING
1773	44.195695	1.16.0	host	USB	32	GET DESCRIPTOR Response STRING
1774	44.195729	host	1.16.0	USB	36	GET DESCRIPTOR Request STRING
1775	44.196281	1.16.0	host	USB	48	GET DESCRIPTOR Response STRING
1776	44.206714	host	1.16.3	USB	27	URB_INTERRUPT in
1777	44.206901	host	1.16.0	USB	36	SET INTERFACE Request

```
> Frame 1748: 36 bytes on wire (288 bits), 36 bytes captured (288 bits) on interface wireshark_extcap1796, id 0
> USB URB
v Setup Data
  > bmRequestType: 0x80
    bRequest: GET DESCRIPTOR (6)
    Descriptor Index: 0x00
    bDescriptorType: DEVICE (0x01)
    Language Id: no language specified (0x0000)
    wLength: 18
```

MIMIC

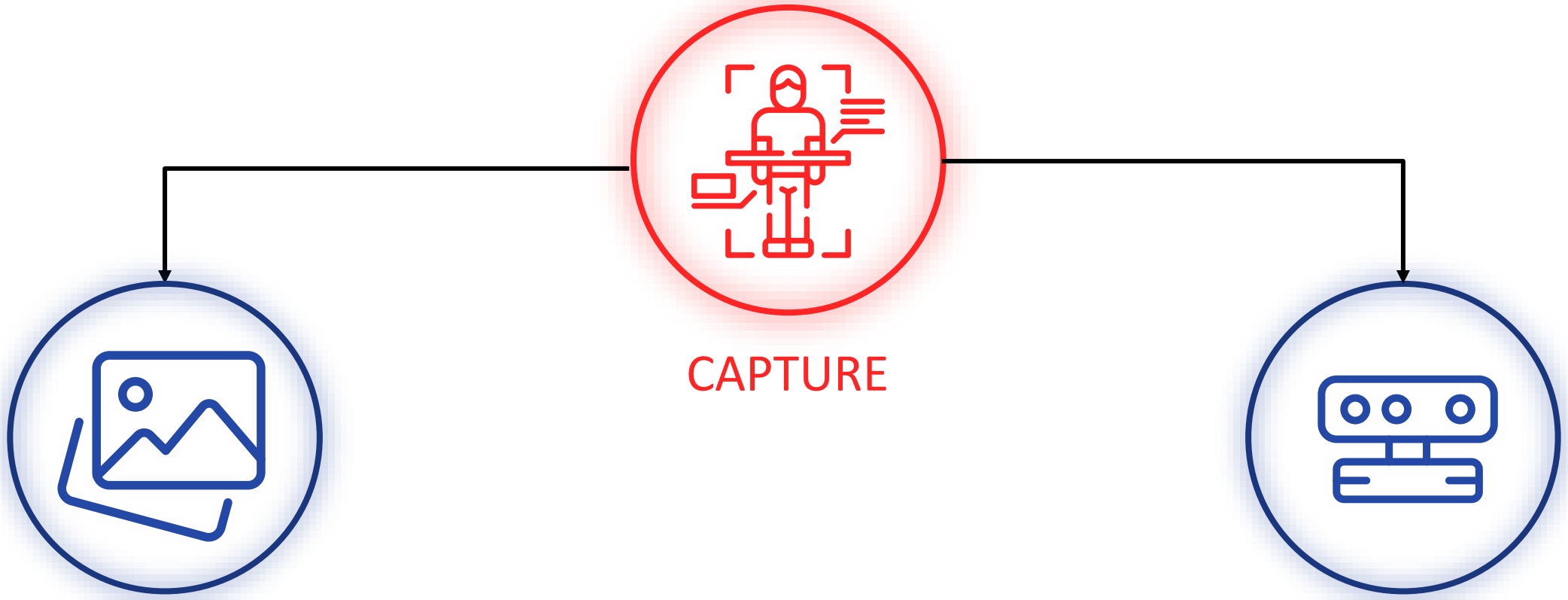



Sign-in options
Manage how you sign in to your device
Select a sign-in option to add, change, or remove it.
Windows Hello Face
This option is currently unavailable—click to learn more
We couldn't find a camera compatible with Windows Hello Face
Learn more
Automatically dismiss the lock screen if Windows recognizes your face
On
Improve recognition
Remove


imgflip.com


Sign-in options
Manage how you sign in to your device
Select a sign-in option to add, change, or remove it.
Windows Hello Face
Sign in with your camera (Recommended)
Windows Hello Fingerprint
This option is currently unavailable—click to learn mor
Windows Hello PIN
Sign in with a PIN (Recommended)
Security Key
Sign in with a physical security key

CAPTURE



CAPTURE

Data Transfer

```
Frame 7804: 75535 bytes on wire (604280 bits), 65535 bytes captured (524280 bits) on interface wireshark_extcap1796, id 0
USB URB
USB isochronous packet
    ISO Data offset: 0x00000000
    ISO Data length: 0x00000164 (relevant)
    ISO USBD status: USBD_STATUS_SUCCESS (0x00000000) (relevant)
    ISO Data: 0c8cc0693d0045134000e100ffffffff8787878787858484858481828181808180818071f…
USB isochronous packet
    ISO Data offset: 0x00000164
    ISO Data length: 0x000002b4 (relevant)
    ISO USBD status: USBD_STATUS_SUCCESS (0x00000000) (relevant)
    ISO Data: 0c8cc0693d00981a4000e1008886868787848684848281828280808283807f7d7f7e7c7d…
USB isochronous packet
    ISO Data offset: 0x00000418
    ISO Data length: 0x000002b4 (relevant)
    ISO USBD status: USBD_STATUS_SUCCESS (0x00000000) (relevant)
    ISO Data: 0c8cc0693d00ea214000e10089868787868483858484838283818171f7f807e7b7c7c7c7a…
```

```
0620  02 00 00 00 00 00 00 0c  8c c0 69 3d 00 45 13 40   ········ ··i=·E·@
0630  00 e1 00 ff ff ff ff 87  87 87 87 87 85 84 84 85   ···· ···· ····
0640  84 81 82 81 81 80 81 80  81 80 7f 7e 7f 7d 7c 7c   ···· ···· ···~·}||
0650  7a 79 79 7a 78 75 76 74  75 73 73 73 72 73 72 73   zyyzxuvt usssrsrs
0660  73 70 6f 70 6f 6e 6c 6b  6c 6b 6b 6b 6a 68 68 67   spoponlk lkkkjhhg
```

CAPTURE

```
STRUCT_PACKED
struct _usb_device_video_mjpeg_payload_header_struct
{
    uint8_t bHeaderLength; /*!< The payload header length. */
    union
    {
        uint8_t bmheaderInfo; /*!< The payload header bitmap field. */
        struct
        {
            uint8_t frameIdentifier : 1U; /*!< Frame Identifier. This bit toggles at each frame start boundary and stays
                                               constant for the rest of the frame.*/
            uint8_t endOfFrame : 1U; /*!< End of Frame. This bit indicates the end of a video frame and is set in the
                                          last video sample that belongs to a frame.*/
            uint8_t presentationTimeStamp : 1U; /*!< Presentation Time Stamp. This bit, when set, indicates the presence
                                                     of a PTS field.*/
            uint8_t sourceClockReference : 1U;  /*!< Source Clock Reference. This bit, when set, indicates the presence
                                                     of a SCR field.*/
            uint8_t reserved : 1U;              /*!< Reserved. Set to 0. */
            uint8_t stillImage : 1U;  /*!< Still Image. This bit, when set, identifies a video sample that belongs to a
                                           still image.*/
            uint8_t errorBit : 1U;    /*!< Error Bit. This bit, when set, indicates an error in the device streaming.*/
            uint8_t endOfHeader : 1U; /*!< End of Header. This bit, when set, indicates the end of the BFH fields.*/
        } headerInfoBits;
    } headerInfoUnion;
    uint32_t dwPresentationTime;      /*!< Presentation time stamp (PTS) field.*/
    uint8_t bSourceClockReference[6]; /*!< Source clock reference (SCR) field.*/
} STRUCT_UNPACKED;
typedef struct _usb_device_video_mjpeg_payload_header_struct usb_device_video_mjpeg_payload_header_struct_t;
```

CAPTURE

Data Transfer

```
    ===>Video Streaming Uncompressed Format Type Descriptor<===
bLength:                            0x1B
bDescriptorType:                    0x24
bDescriptorSubtype:                 0x04
bFormatIndex:                       0x01
bNumFrameDescriptors:               0x01
guidFormat:                         {00000032-0002-0010-8000-00AA00389B71}
bBitsPerPixel:                      0x08
bDefaultFrameIndex:                 0x01
```

KSDATAFORMAT_SUBTYPE_L8_IR

```
    ===>Video Streaming MJPEG Frame Type Descriptor<===
    --->This is the Default (optimum) Frame index
bLength:                            0x1E
bDescriptorType:                    0x24
bDescriptorSubtype:                 0x07
bFrameIndex:                        0x01
bmCapabilities:                     0x01
wWidth:                           0x0280 = 640
wHeight:                          0x0168 = 360
dwMinBitRate:                  0x06978000
dwMaxBitRate:                  0x06978000
dwMaxVideoFrameBufferSize:     0x00070800
dwDefaultFrameInterval:        0x00051615 = 33.333300 mSec (30.00 Hz)
bFrameIntervalType:                 0x01
===>Additional Discrete Frame TypeData
dwFrameInterval[1]:            0x00051615 = 33.333300 mSec (30.00 Hz)
```

CAPTURE

Data Transfer

```
===>Video Streaming Uncompressed Format Type Descriptor<===
bLength:                              0x1B
bDescriptorType:
bDescriptorSubty
bFormatIndex:
bNumFrameDescrip
guidFormat:
bBitsPerPixel:
bDefaultFrameInd
```

```
===>Video Streaming MJPEG Frame Type Descriptor<===
                                                    ex
                                                    ec (30.00 Hz)
                                                    ec (30.00 Hz)
```

IR stream

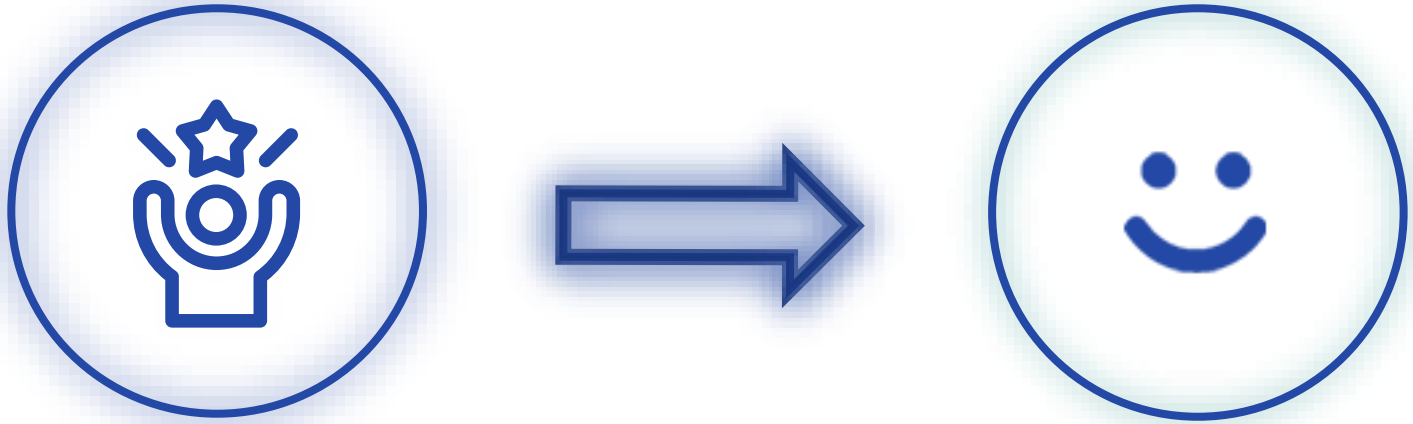
The Windows inbox USB video class (UVC) driver supports cameras that capture the scene in YUV format and transmit the pixel data over USB as uncompressed YUV or as compressed MJPEG frames.

The following format type GUIDs should be specified in the stream video format descriptor, as defined in the WDK ksmedia.h header file:

Type	Description
KSDATAFORMAT_SUBTYPE_L8_IR	Uncompressed 8 bit luma plane. This type maps to MFVideoFormat_L8.
KSDATAFORMAT_SUBTYPE_L16_IR	Uncompressed 16 bit luma plane. This type maps to MFVideoFormat_L16.
KSDATAFORMAT_SUBTYPE_MJPG_IR	Compressed MJPEG frames. Media Foundation converts this into NV12 uncompressed frames and uses only the luma plane.

https://docs.microsoft.com/en-us/windows-hardware/drivers/stream/uvc-camera-implementation-guide#ir-stream

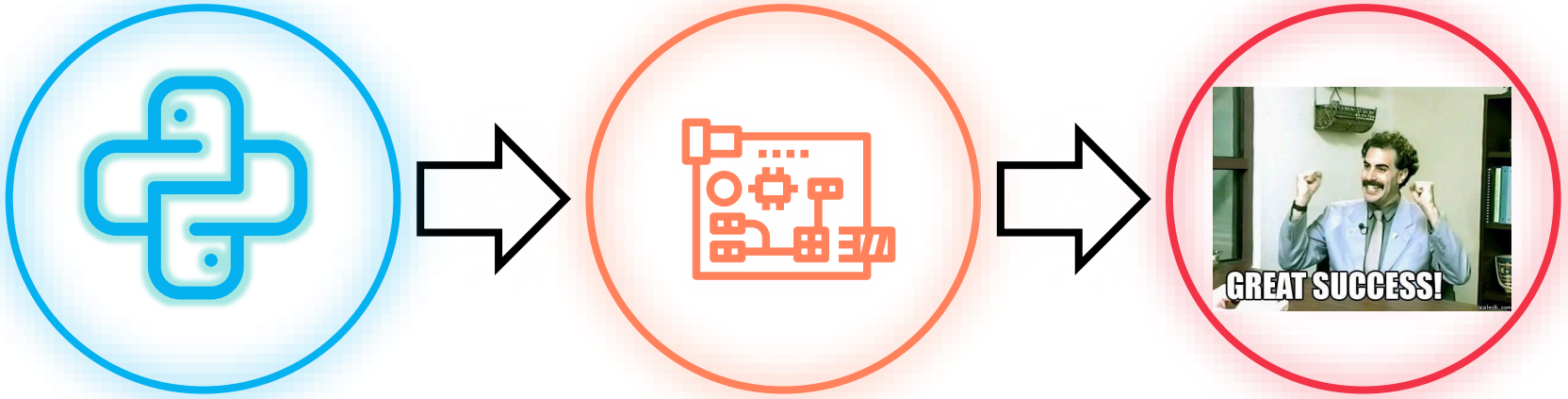
EXPLOIT



EXPLOIT

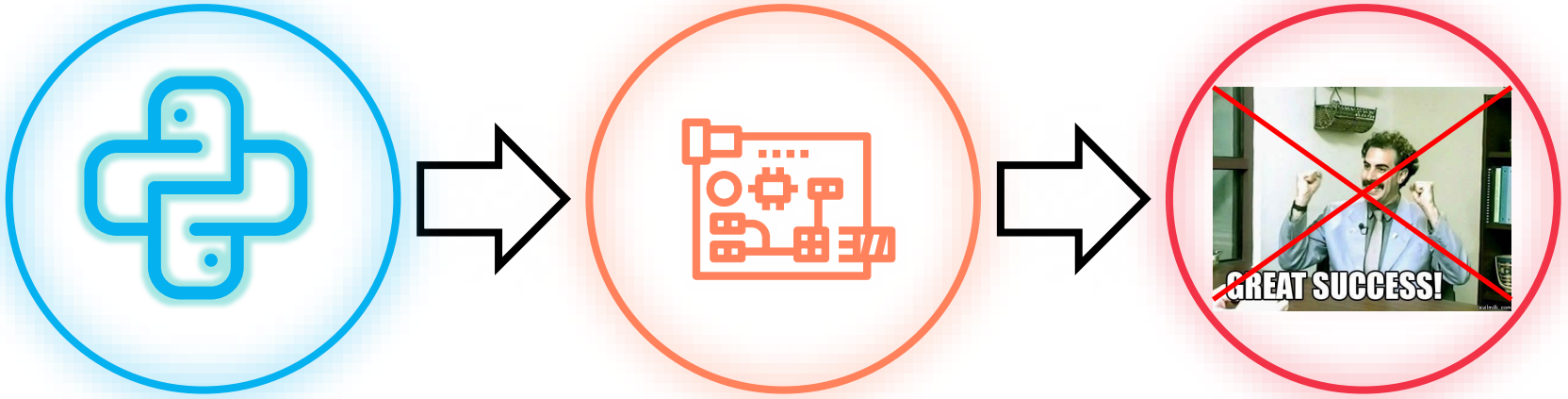
BYPASS

EXPLOIT



PyShark → Save → Profit

EXPLOIT

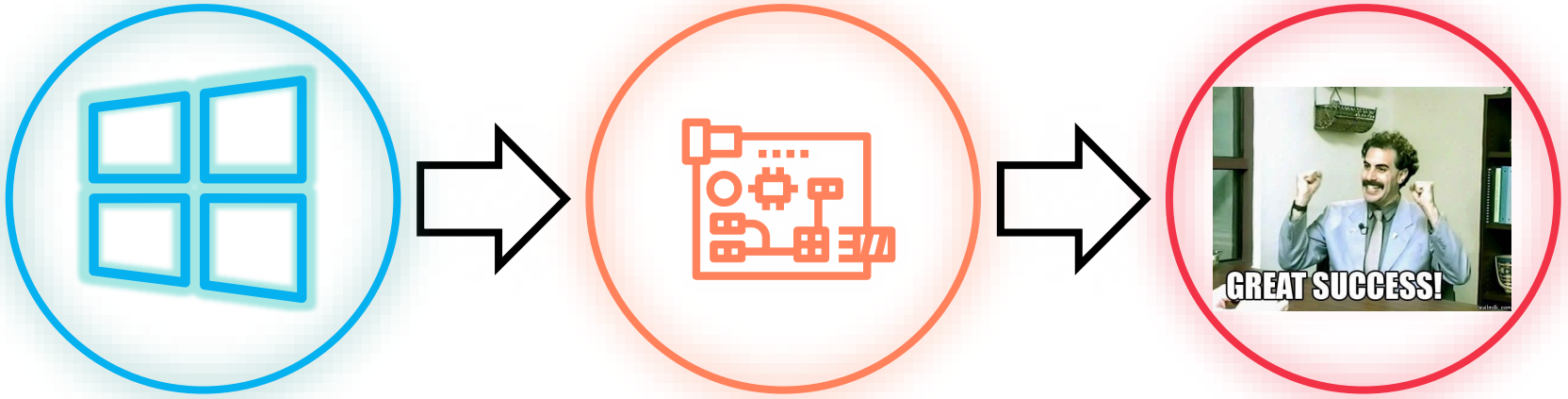


PyShark

Save

Profit

EXPLOIT



MS Media Framework

Save

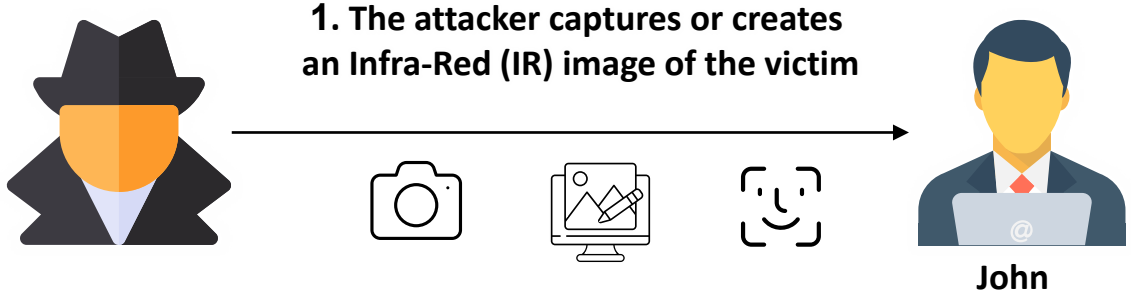
Profit

EXPLOIT

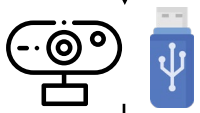
IT’S DEMO TIME!

EXPLOIT

1. The attacker captures or creates an Infra-Red (IR) image of the victim

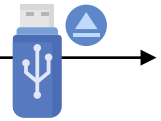


2. The attacker creates a custom USB device that impersonates a USB camera device



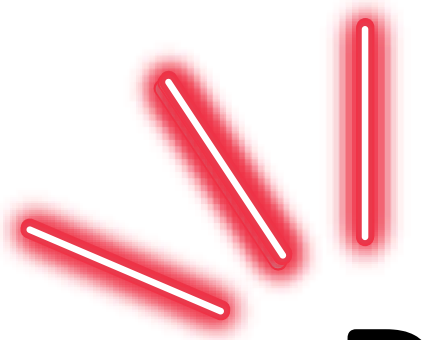
3. The attacker saves the captured data on a custom USB device

4. The Attacker connects the custom USB device to John's computer. The USB device transmits the captured IR image



5. The biometrical system accepts the biometrical credentials and logs the user in

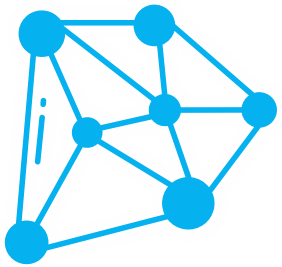
6. The attacker gets access to sensitive information and files of the victim



Part 4: What's next

You say goodbye and I say…

What's next

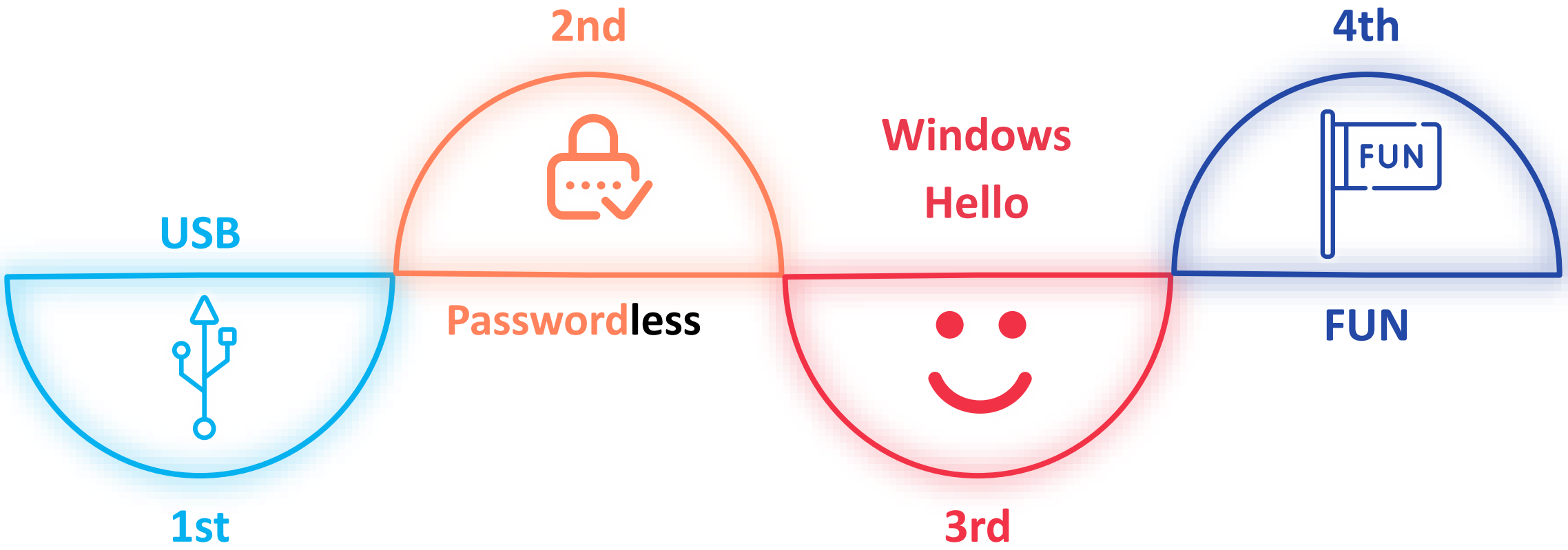


Converting Color Image to IR

Manipulating Bio Database

Fuzzing

Closing Remarks



Apply - Customers

- Make sure your computer OS is updated
- Don’t base your authentication on public factors
- Make sure you combine biometric authentication with other authentication factors
- Microsoft published mitigations for the vulnerability
  - Some of them are optional - enable them too

Apply – Reseachers / Developers

- Biometric systems can be compromised
  - Dev - Create ‘trust’ between the sensor and your system
  - Research – Attack the sensors
- Creating trust with external sensors is challenging
  - Dev – When possible -> disable external sensors
  - Research – A weak node in the chain

Mitigations + Blog post

- CVE-2021-344
  - https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2021-34466 CVE-2021-34466
- KB5005478
  - https://support.microsoft.com/en-us/topic/kb5005478-windows-hello-cve-2021-34466-6ef266bb-c68a-4083-aed6-31d7d9ec390e

Thank you!

@OmerTsarfati