

# Bypassing Windows Hello for Business and Pleasure



Hello, Omer!

# WHOAMI



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

# AGENDA



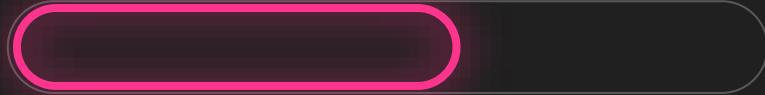
# BY THE END OF THIS TALK

WINDOWS HELLO  
AND ITS FEATURS



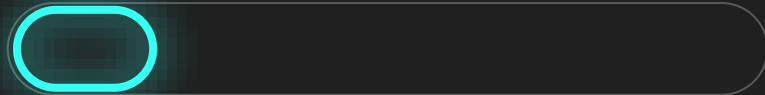
85%

USB & UVC



50%

CONFUSION



10%

FUN



MAX\_INT%



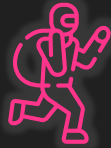
# Part 1: Passwordless

and friends

# WHY PASSWORDLESS



WEAK  
PASSWORDS



DATA  
BREACHES



PHISHING  
ATTACKS



REUSE

# HOW PASSWORDLESS



FINGERPRINT



FACE  
RECGONITION

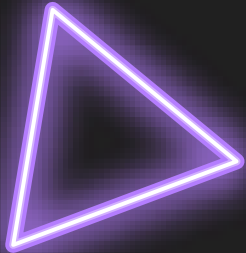


SECURITY  
KEY



OTP

# Part 2: Windows Hello





SAY \_\_\_\_\_ TO WINDOWS \_\_\_\_\_

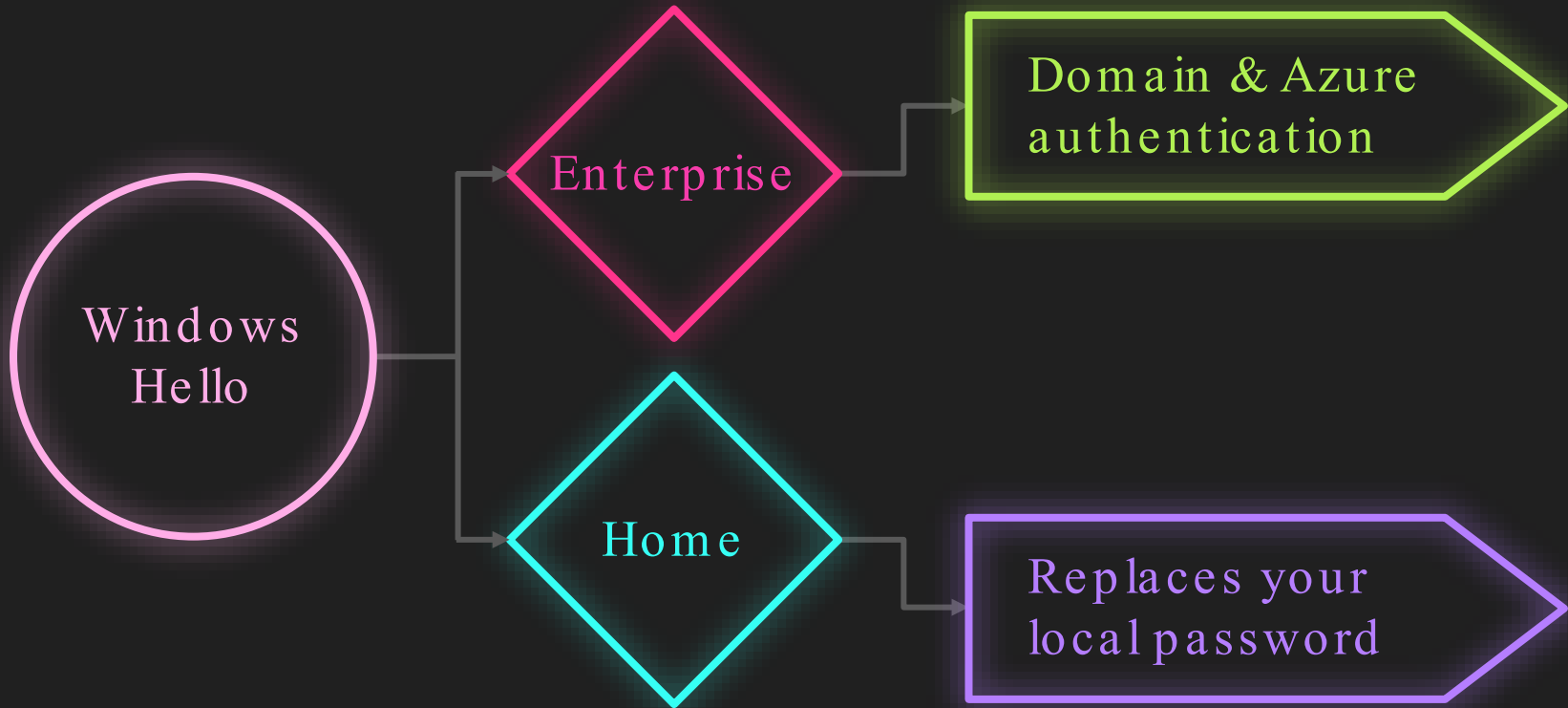


Passwordless

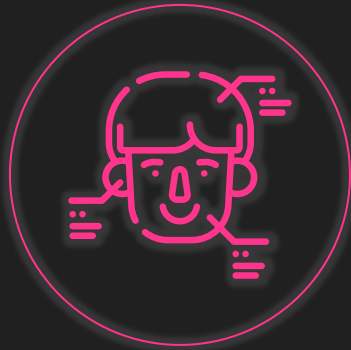


Windows Hello

# 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

Challenging to capture and modify

## PREVIOUS ART

Many public reseraches

## PROTOCOL

Complicated protocol and lack of public implmentation

# PIN CODE



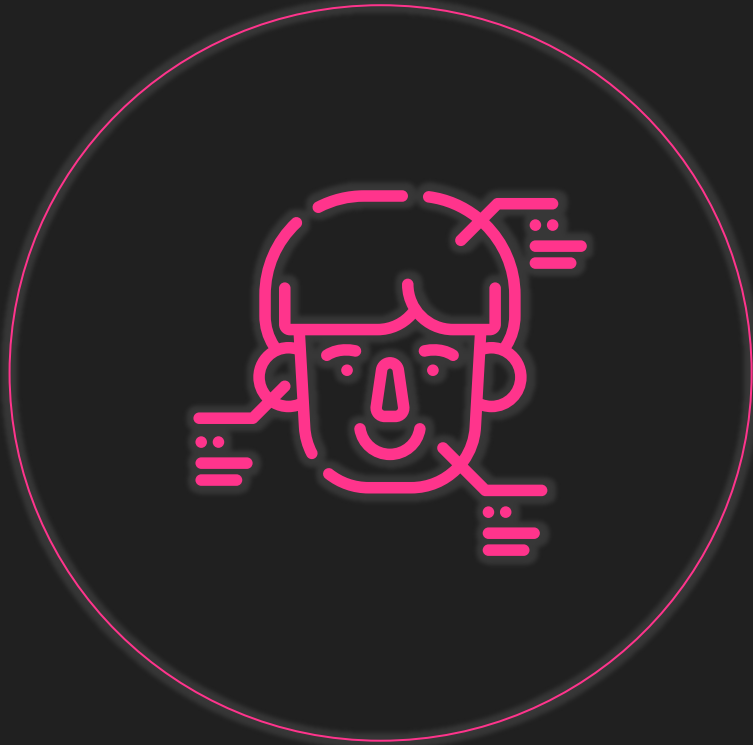
## ANTI-BRUTEFORCE

Anti-bruteforce mechanism

## SAME CHALLENGE

Same challenge as cracking a password

# FACE RECOGNITION



## ANTI-BRUTEFORCE

Anti-bruteforce mechanism

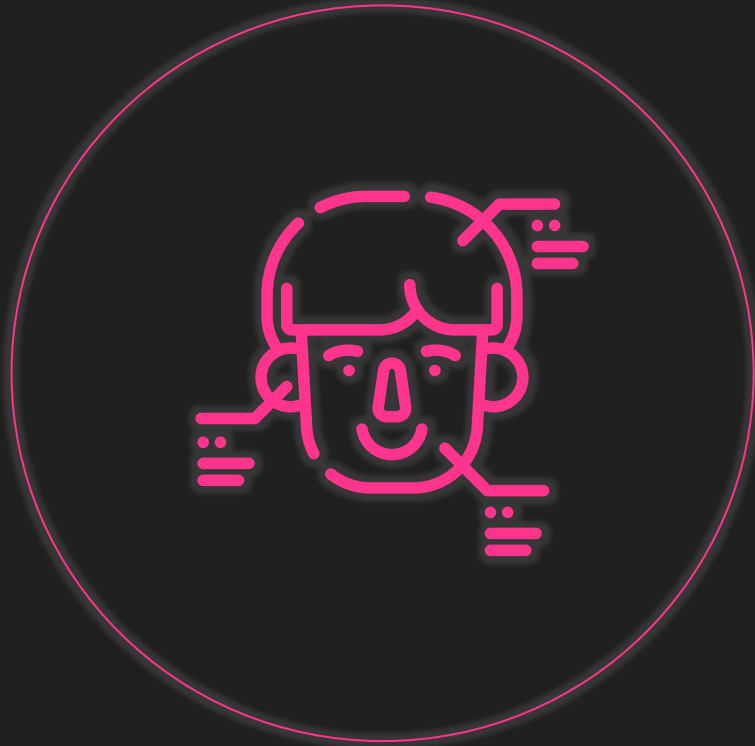
## EASY START

NXP implemented a basic  
USB video camera

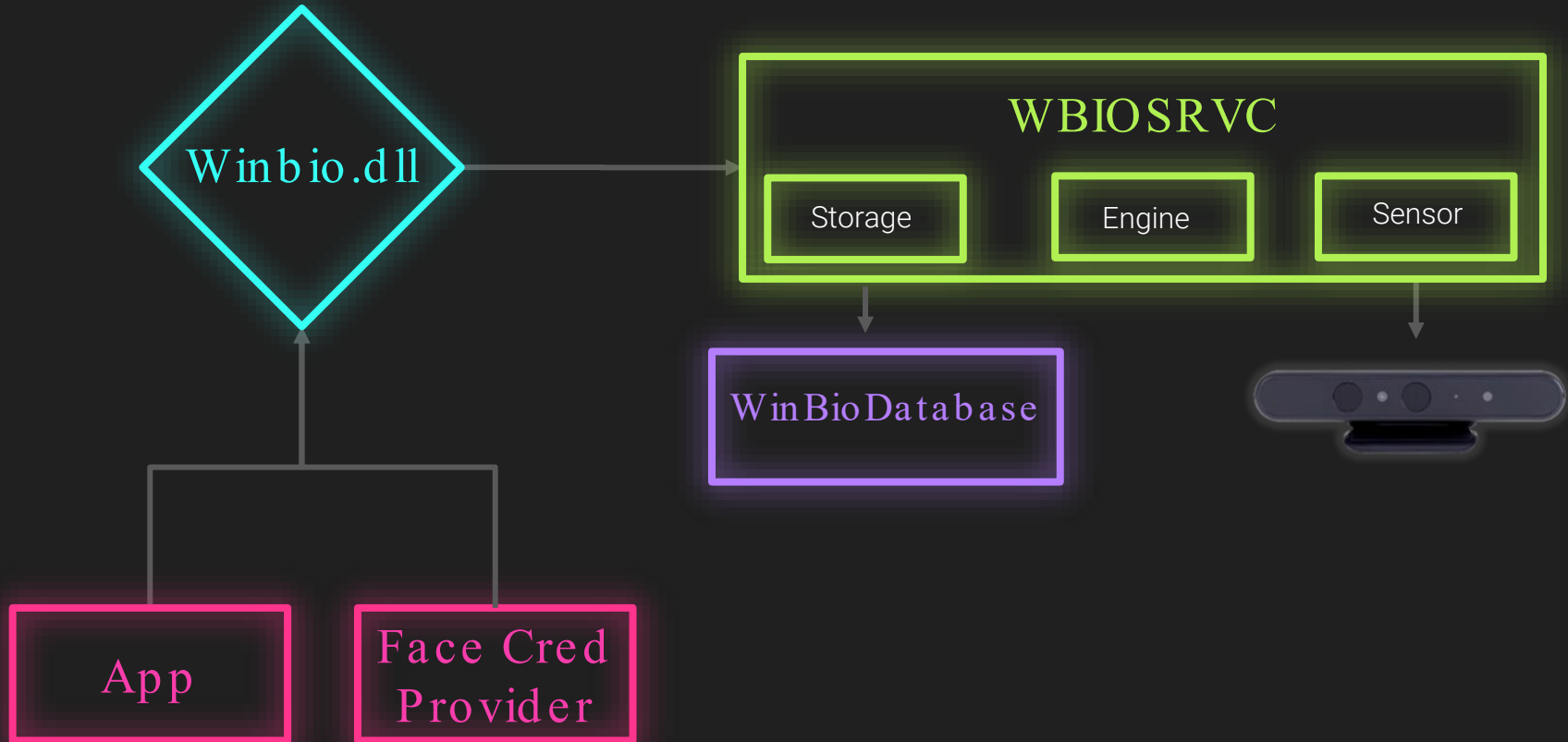
## PUBLIC DATA

Face images can be captured  
easily

# FACE RECOGNITION

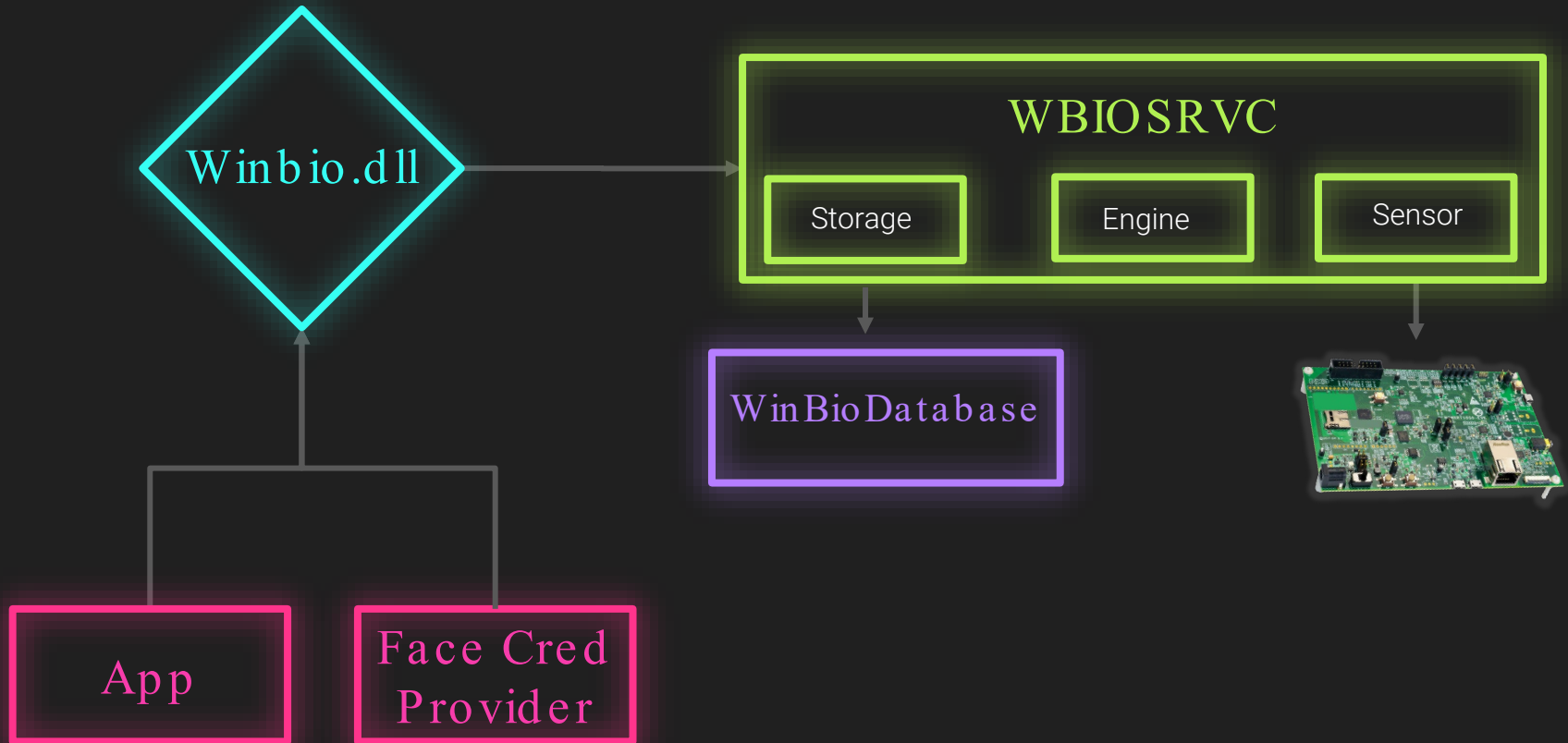


# BIOMETRIC AUTH & WINDOWS HELLO

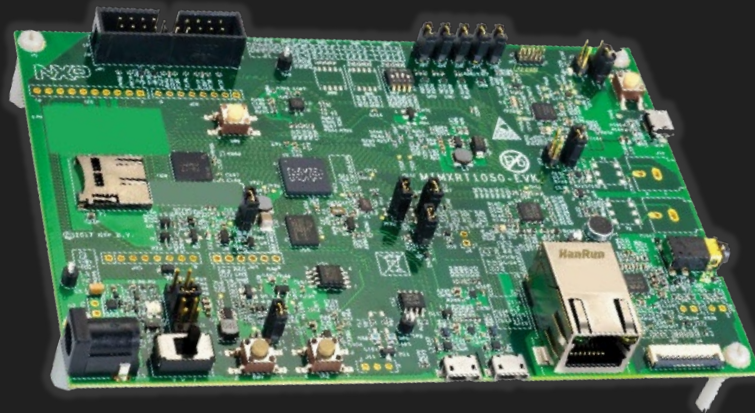


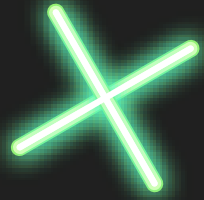


# BIOMETRIC AUTH & WINDOWS HELLO



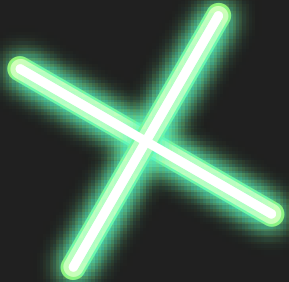
# NXP – EVALUATION BOARD





# Part 3: Exploiting

show me the **money** \$\$\$

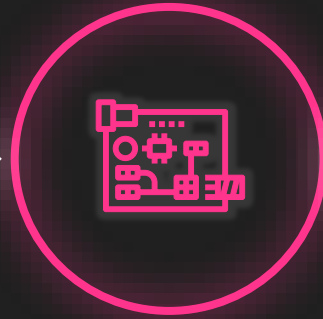


# RESEARCH VECTOR



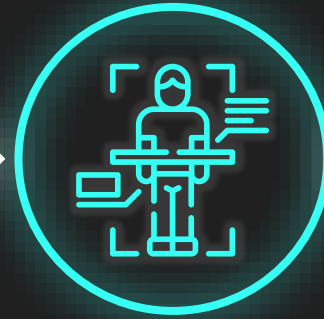
**ANALYZE**

USB & UVC



**MIMIC**

Mimics `hello` camera



**CAPTURE**

Capture the target person



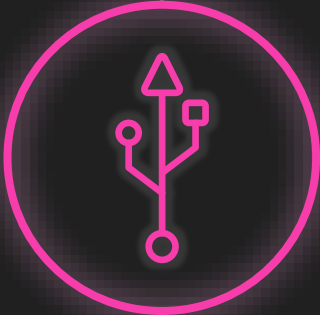
**EXPLOIT**

Bypass Windows Hello

# ANALYZE



ANALYZE



USB Specs



Capture Traffic

# 5.3.2 ANALYZE – USB SPECS

A USB logical device appears to the endpoint sets that implement an interface manages the device using the Default bundles (associated with an endpoint between a buffer on the host and depending on transfer direction when bus access is

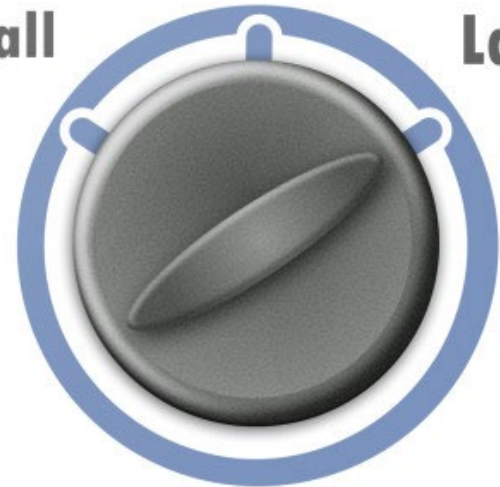
## 5.3.2.1 Stream P

Stream pipes deliver the data content. Data pipes are always unidirectional.

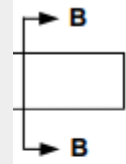
Data flowing through the USB System Software using the same stream pipe is first-in, first-out.

A stream pipe to a device corresponds to an endpoint in the opposite direction of

Small Medium Large



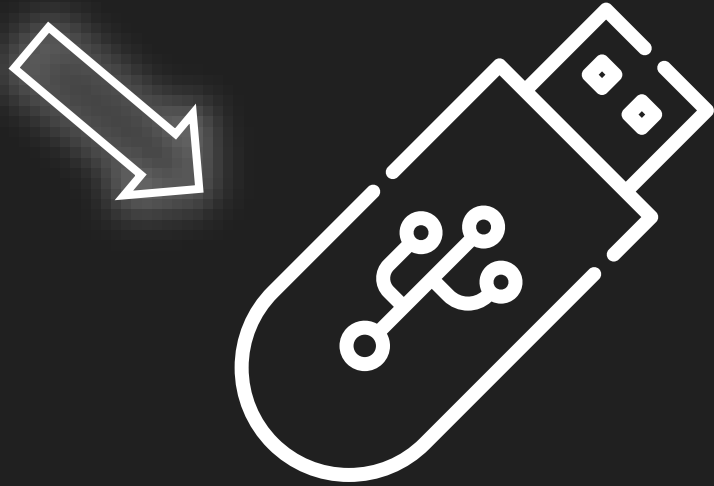
# OOF SIZE



Cut End  
B Device.)

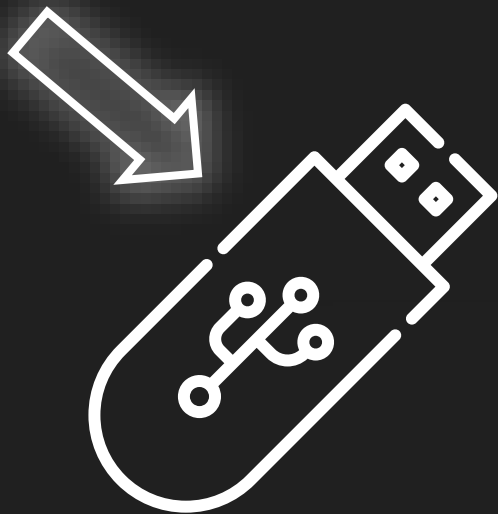
# ANALYZE – USB STRUCTURE

NICE GUY

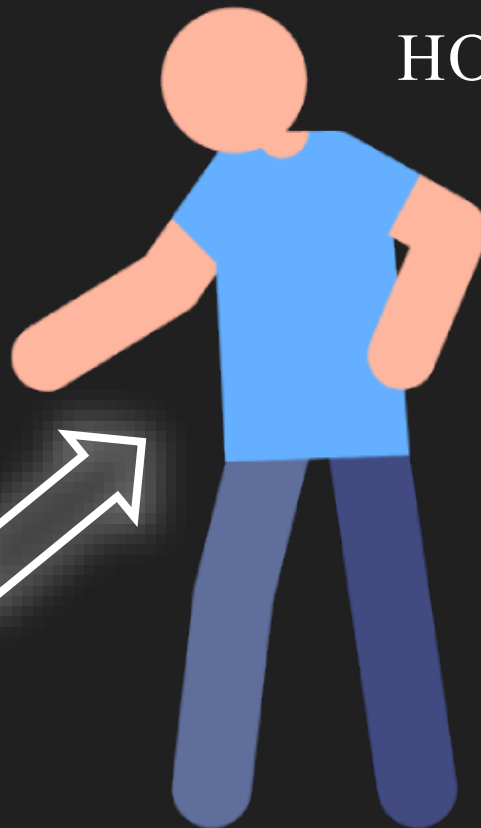


# ANALYZE – USB STRUCTURE

NICE GUY



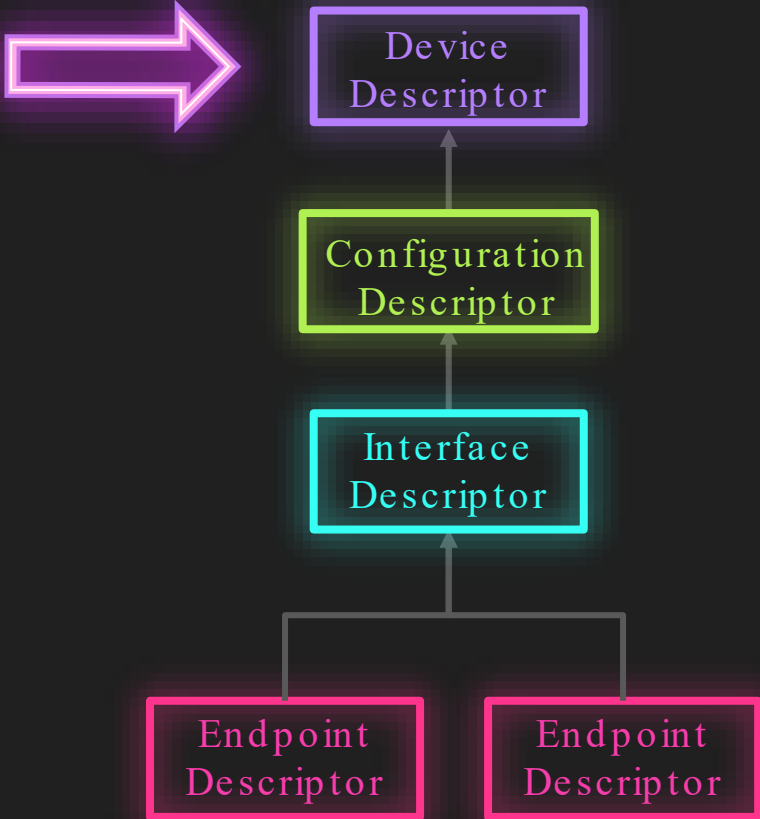
HOSTin



Bully



# ANALYZE – USB STRUCTURE



# 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
  ✓ 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
```

# ANALYZE – USB STRUCTURE

## Device Descriptor

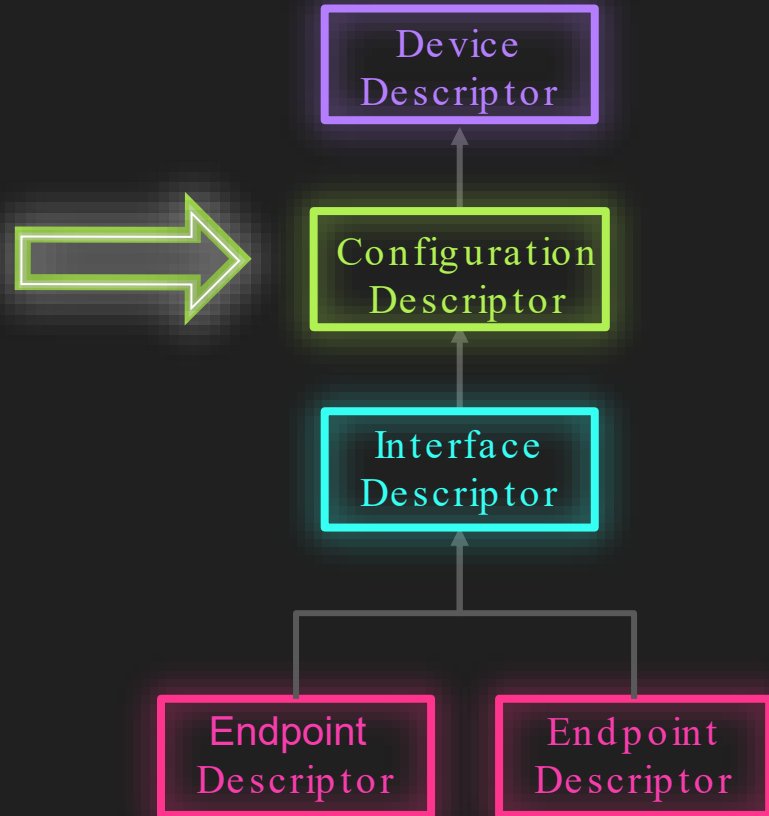
```
> Frame 1749: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface wireshark_extcap1796, id 0
> USB URB
  ✓ DEVICE DESCRIPTOR
    bLength: 18
    bDescriptorType: 0x01 (DEVICE)
    bcdUSB: 0x0201
    bDeviceClass: Miscellaneous (0xef)
    bDeviceSubClass: 2
    bDeviceProtocol: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

# 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
  ✓ 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
```


# ANALYZE – USB STRUCTURE



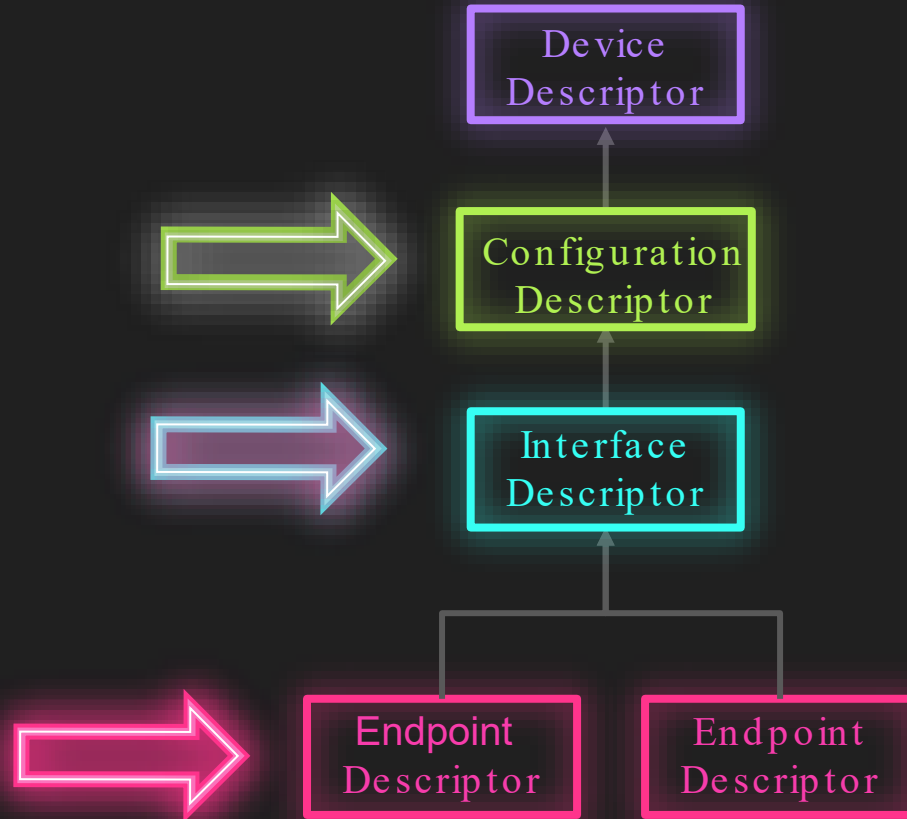
# ANALYZE – USB STRUCTURE

## Configuration Descriptor

```
> Frame 1751: 37 bytes on wire (296 bits), 37 bytes captured (296 bits) on interface wireshark_extcap1796, id 0
> USB URB
√ CONFIGURATION DESCRIPTOR
  bLength: 9
  bDescriptorType: 0x02 (CONFIGURATION)
  wTotalLength: 1055
  bNumInterfaces: 4
  bConfigurationValue: 1
  iConfiguration: 4
  > Configuration bmAttributes: 0x80 NOT SELF-POWERED NO REMOTE-WAKEUP
  bMaxPower: 250 (500mA)
```



# ANALYZE – USB STRUCTURE



# ANALYZE – USB STRUCTURE

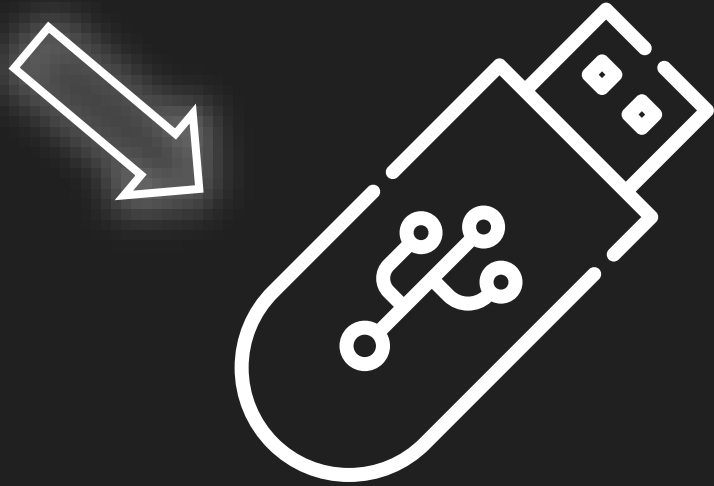
## Configuration Descriptor

```
> 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
▼ 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!



> Frame 1749: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface wireshark\_extcap1796, id 0  
> USB URB  
▼ DEVICE DESCRIPTOR GET\_DESCRIPTOR Response CONFIGURATION

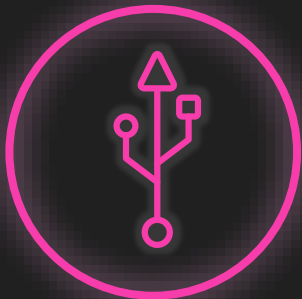
bLength: 18	GET_DESCRIPTOR Response
bDescriptorType: 0	SET CONFIGURATION Response
bcdUSB: 0x0201	Unknown type 7f
bDeviceClass: Misc	Unknown type 7f
bDeviceSubClass: 2	GET_DESCRIPTOR Response
bDeviceProtocol: 3	GET_DESCRIPTOR Response
bMaxPacketSize0: 64	GET_DESCRIPTOR Response
idVendor: Quanta	SET INTERFACE Response
idProduct: Unknown	GET INFO Response [Export]
bcdDevice: 0x0011	GET MIN Response [Export]
iManufacturer: 3	GET MAX Response [Export]
iProduct: 1	GET RES Response [Export]
iSerialNumber: 2	GET DEF Response [Export]
bNumConfigurations: 1	GET MIN Response [Export]

Context menu options:

- Mark/Unmark Packet (Ctrl+M)
- Ignore/Unignore Packet (Ctrl+D)
- Set/Unset Time Reference (Ctrl+T)
- Time Shift... (Ctrl+Shift+T)
- Packet Comment... (Ctrl+Alt+C)
- Edit Resolved Name
- Apply as Filter
- Prepare as Filter
- Conversation Filter
- Colorize Conversation
- SCTP
- Follow
- Copy
- Protocol Preferences
- Decode As...
- Show Packet in New Window

Copy submenu options:

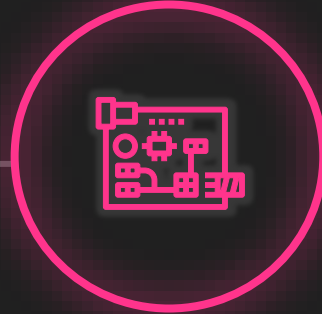
- Summary as Text
- ...as CSV
- ...as YAML
- As Filter (Ctrl+Shift+C)
- Copy Bytes as Hex + ASCII Dump
- ...as Hex Dump
- ...as Printable Text
- ...as a Hex Stream
- ...as Raw Binary



## 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



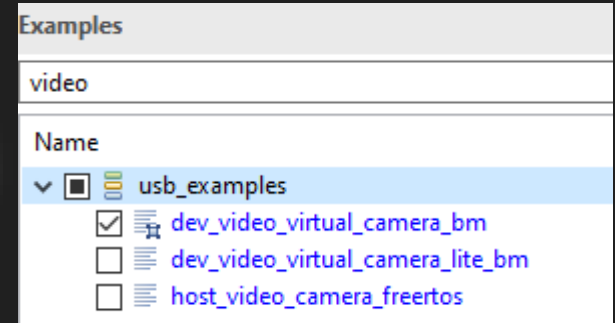
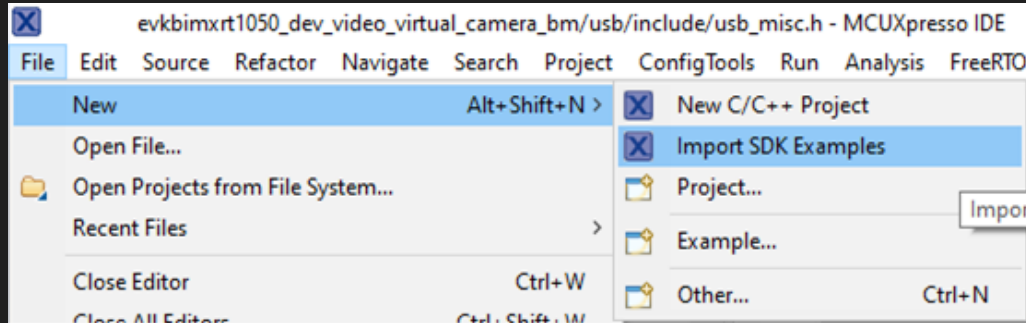
Implementation



Compatible W/  
Win Hello

# 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

```
uint8_t USBVideoMSOS20DescriptorSet[USBVideoMSOS20DescriptorSetLength] =
BOS D
The US
• T
• b
• b
• w
• w
• b
d
v
• b
If any
stack v

/* 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.
//0x0E, 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, 'G', 0x00,
```

in  
ability  
deviceCaps

otherwise the

# MIMIC

No.	Time	Source	Destination	Protocol	Length	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

Setup Data

> bmRequestType: 0x80

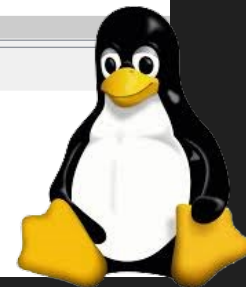
bRequest: GET\_DESCRIPTOR (6)

Descriptor Index: 0x00



bDescriptorType: DEVICE (0x01)

Language Id: no language specified (0x0000)

wLength: 18



# MIMIC



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**  
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](#)

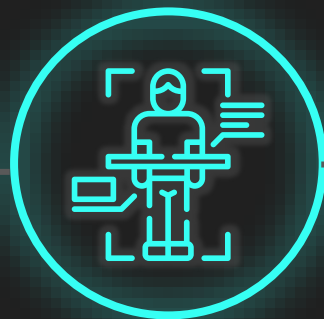
## Sign-in options

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 more
-  **Windows Hello PIN**  
Sign in with a PIN (Recommended)
-  **Security Key**  
Sign in with a physical security key

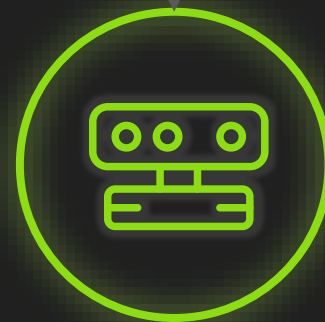
# CAPTURE



CAPTURE



Capture color  
frames



Capture IR frames



# 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

## 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.

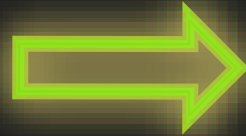
The following format type GUIDs should be specified in the stream video format descriptor, as defined in the WDK ksmedia.h header file:

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

# EXPLOIT

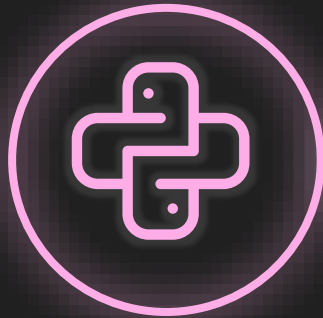


EXPLOIT

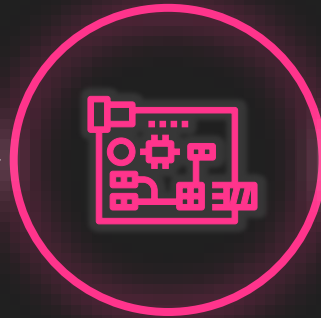


BYPASS

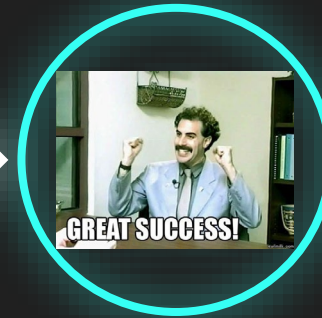
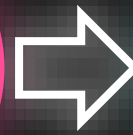
# EXPLOIT



PyShark

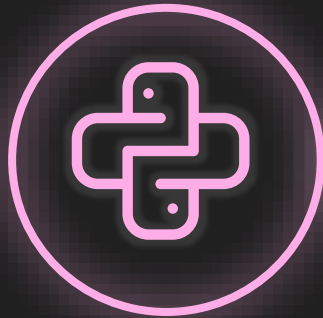


Save

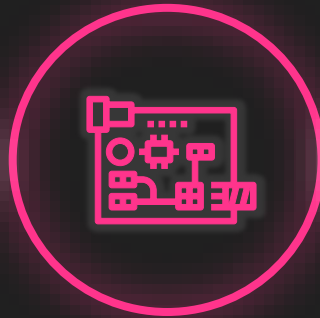


Profit

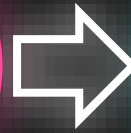
# EXPLOIT



PyShark

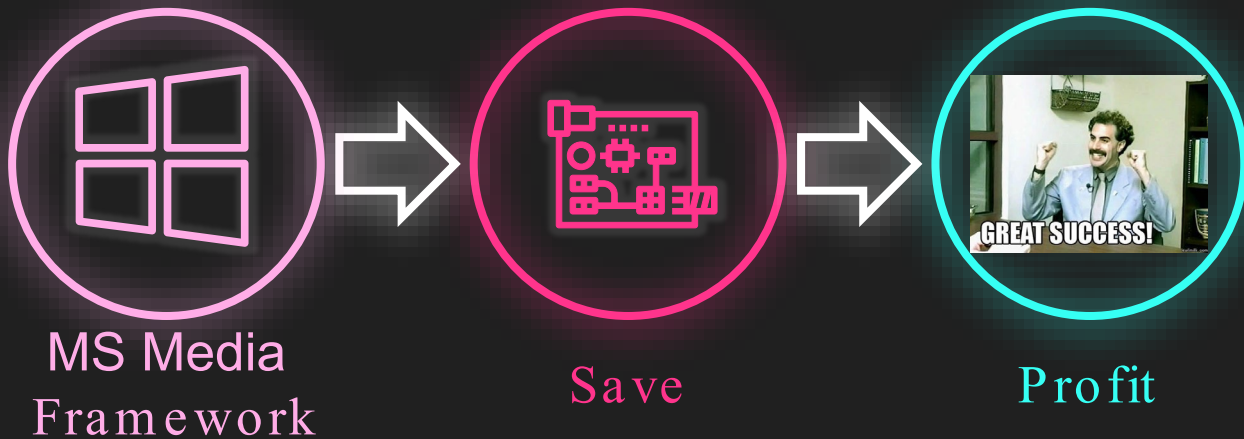


Save



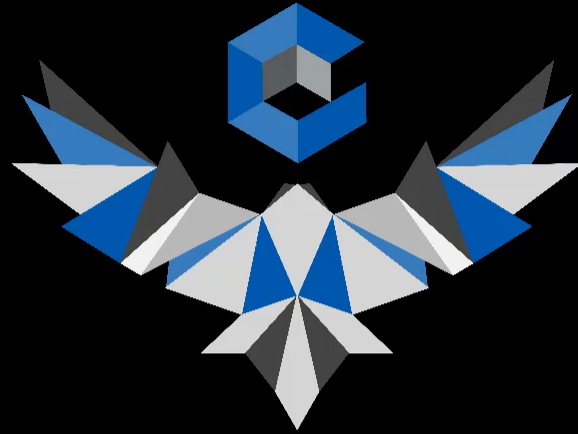
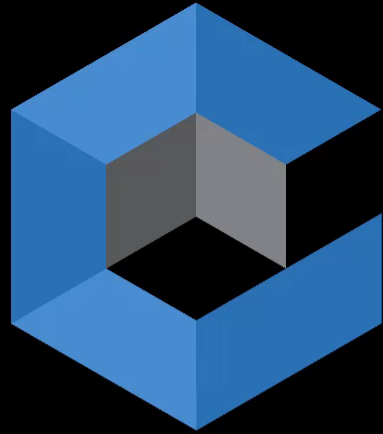
Profit

# EXPLOIT



EXPLOIT

IT'S DEMO TIME!

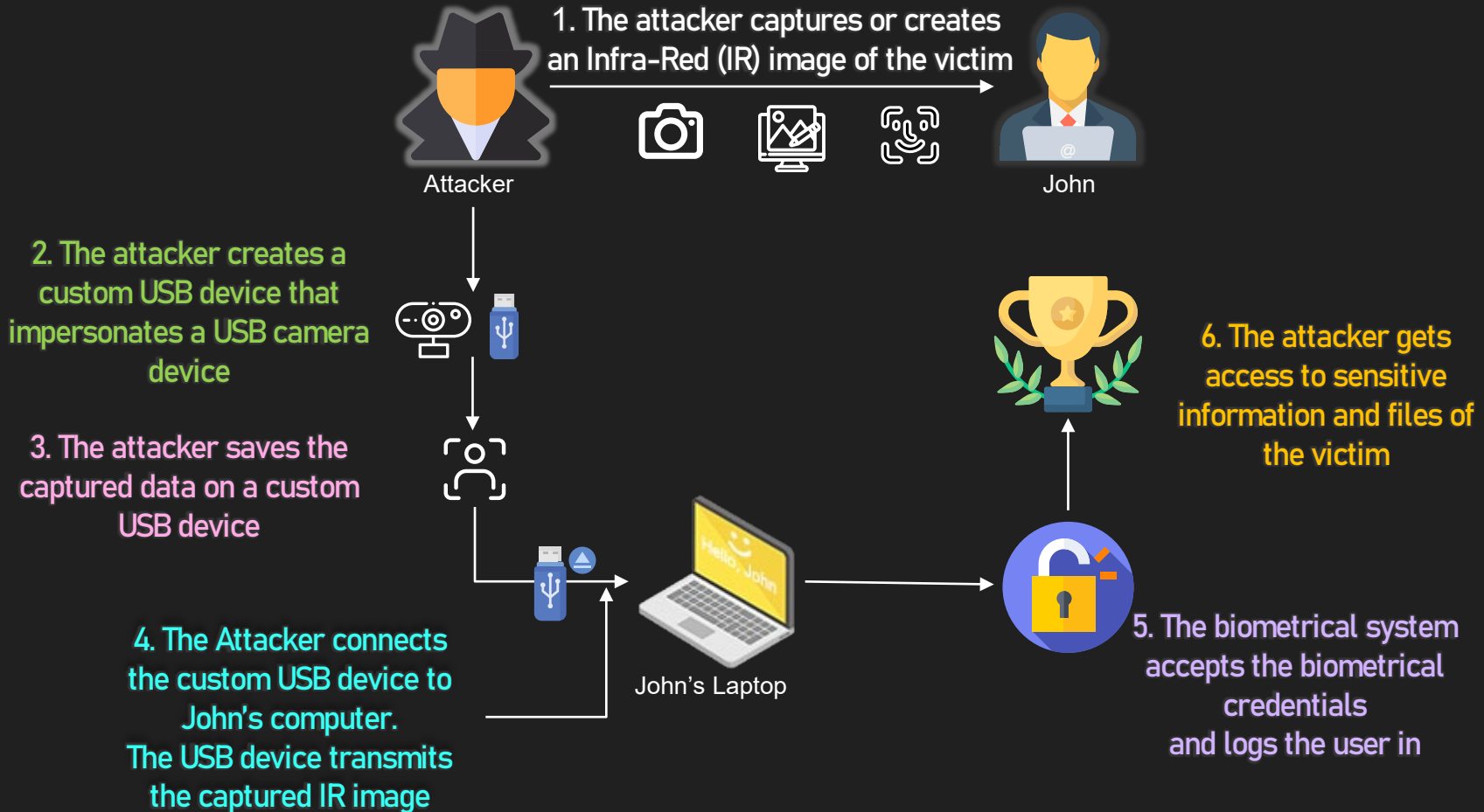


# Bypassing Windows Hello via a Custom USB Device

CyberArk Labs



# EXPLOIT

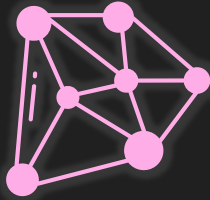




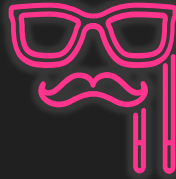
# Part 4: What's next

You say goodbye and I say...

# What 's next



Converting  
Color Image  
to IR

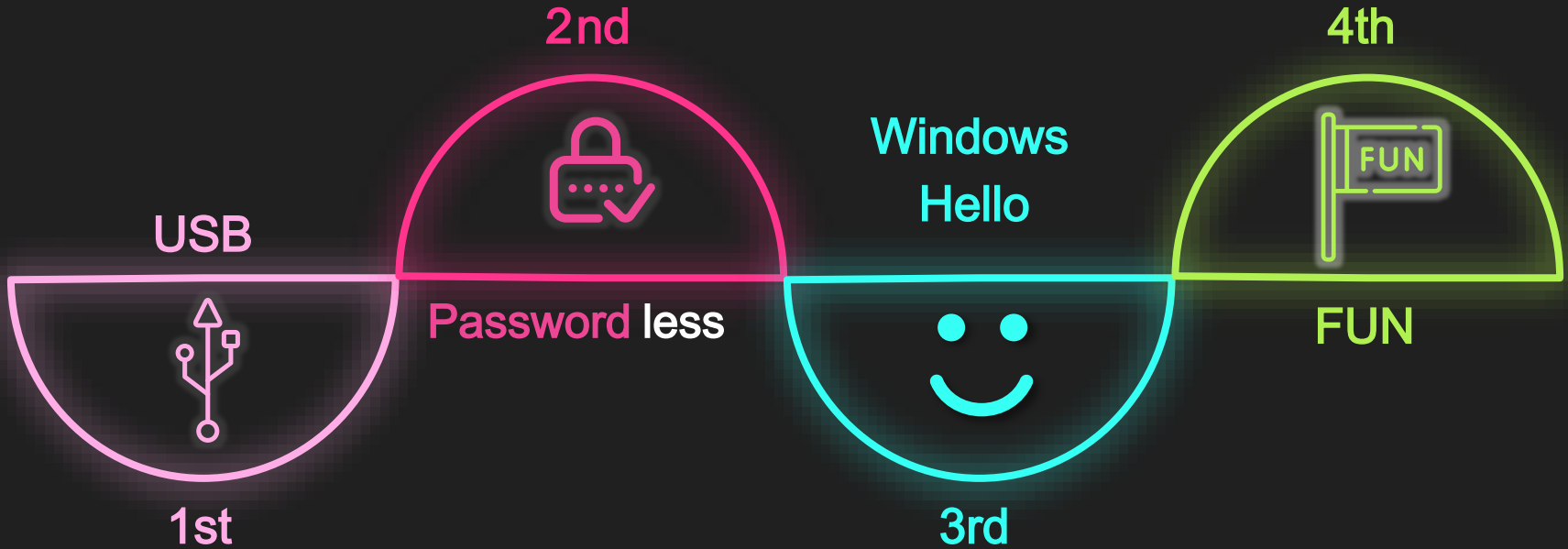


Manipulating  
Bio Database



Fuzzing

# Closing Remarks



## Closing Remarks

Don't base your authentication on a public factor

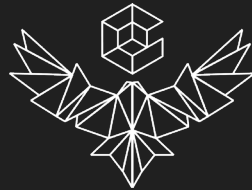
# Closing Remarks

- Blog post
  - <https://www.cyberark.com/resources/threat-research-blog/bypassing-windows-hello-without-masks-or-plastic-surgery>
- CVE-2021-34466
  - <https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2021-34466>

# Thank you!



@OmerTsarfati



**CYBERARK®**