Usb Video Class Frame Detector

Version 1.0.0

December 16, 2024

UVCFD (UVC Frame Detector)

This programme uses a pipeline with tshark to parse data, recombine URB into payloads and frames. Detects invalid data and payload headers, frames. Can use a live stream using a camera application or can use a pcapng file.

How to build

Build with cmake

For the release mode:

Mkdir images
Mkdir build
Cd build
Cmake -dcmake_build_type = Release ..
Cmake –build . –config Release

For the debug mode:

Mkdir images
Mkdir build
Cd build
Cmake -dcmake_build_type = Debug ..
Cmake –build . –config Debug

Compiler information
-- Building for: Visual Studio 17 2022
-- Selecting Windows SDK version 10.0.26100.0 to target Windows 10.0.19045.
-- The CXX compiler identification is MSVC 19.41.34120.0
-- The C compiler identification is MSVC 19.41.34120.0

How to run

Use the shellscript given. Make sure the library packages are installed. The Wireshark program needs to be installed at the Program Files directory with USBPcapCMD. If not, change the shellscript path. But DO NOT Edit the Fields Data or its Sequence. Make sure the uvcfd.exe correctly path, or edit the shellscript.

.\run_uvcfd.ps1

This will display the list of currently connected USB devices. Find the designated device and note down the correct device number and endpoint address (usually 1 for UVC video). Open any camera application that starts streaming.

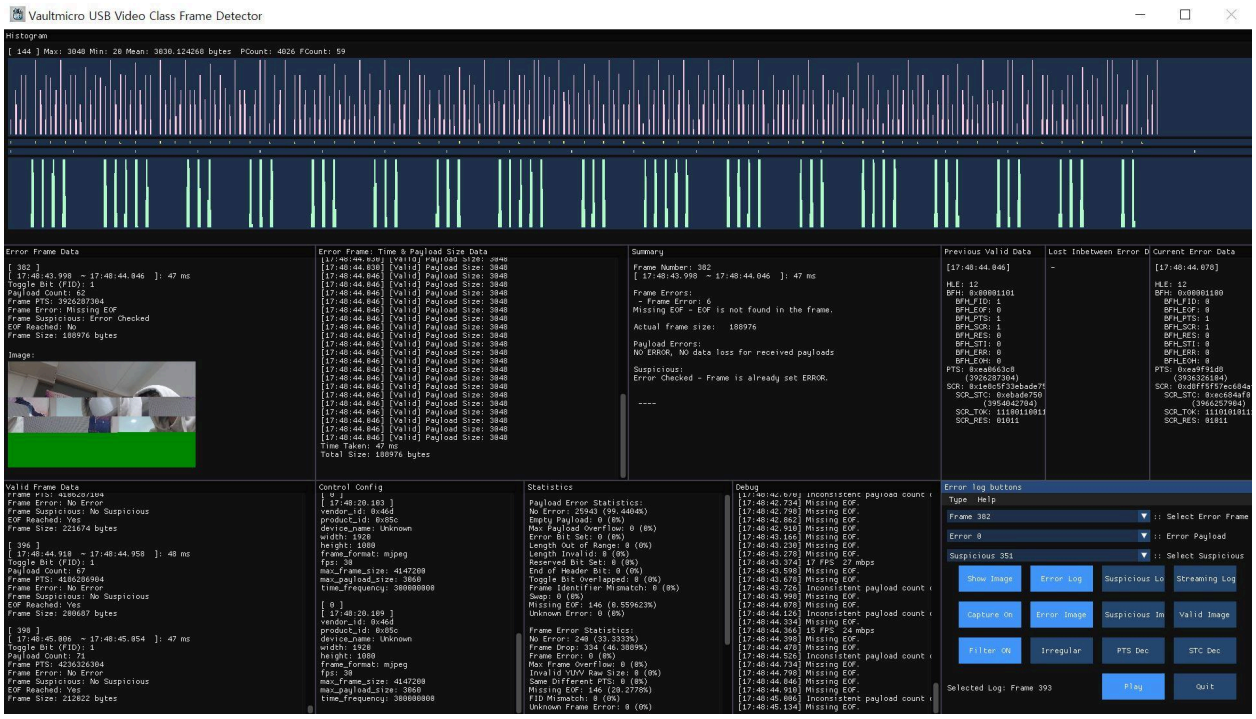
To use .pcapng file that has been recorded previously,

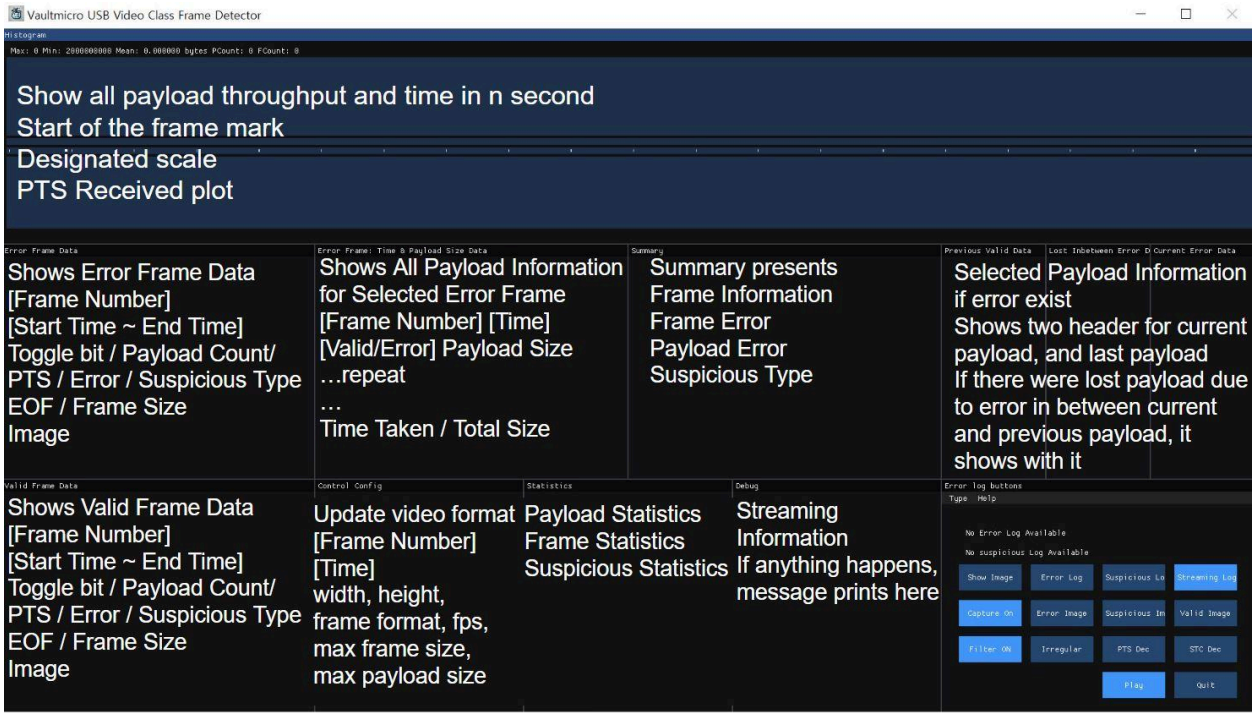
.\run_uvcfdp.ps1

It will receive the .pcapng file path and show the received control data. Same work here, note down the correct device number and endpoint address.

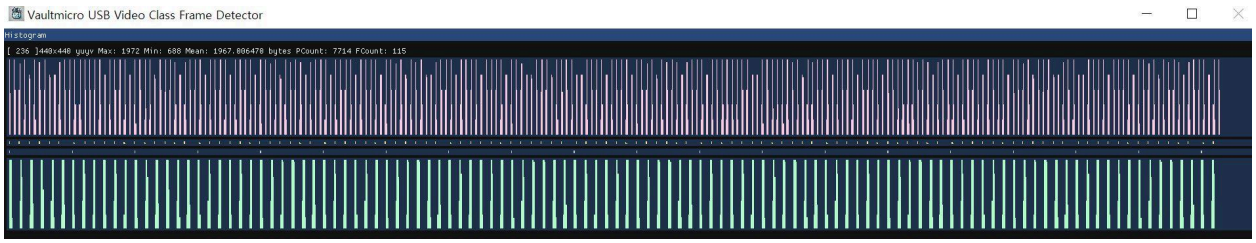
Screen Layout

Overall, the upper three windows display the image, graph, and control panel. The middle section shows error log frame information and summarizes the error types. The bottom section displays the latest streaming data information with updating statistics.





Graphs

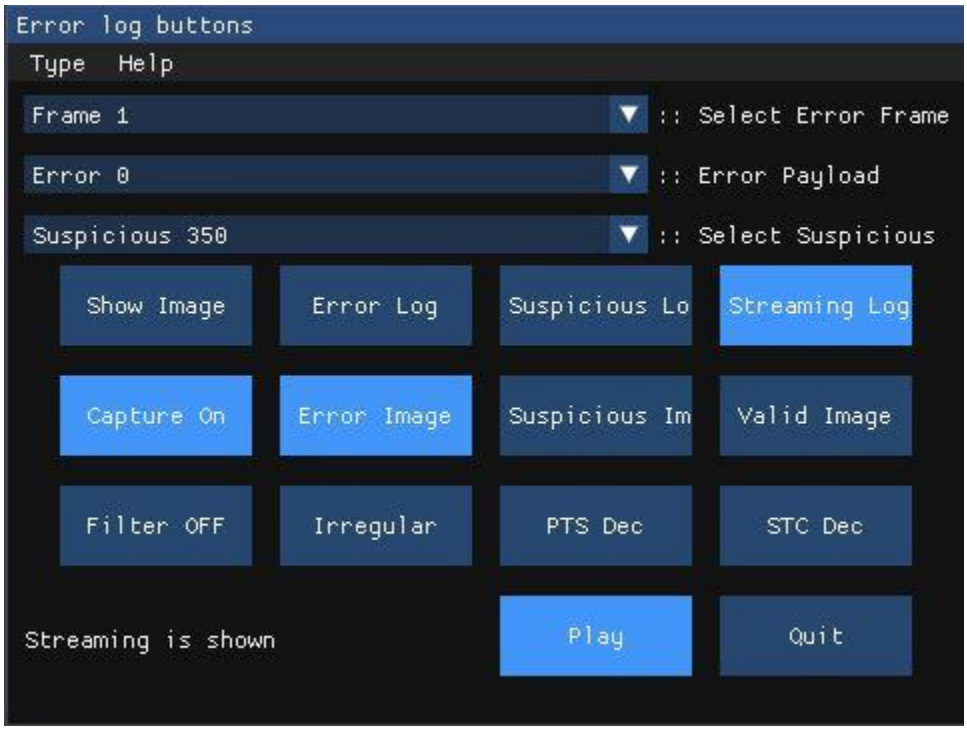


Coral coloured graph shows the actual received time of the URB block in the computer system. Mint coloured graph shows the PTS time that the camera gave. Yellow dots represent the start of every frame. White dots represent the time scale of every designated millisecond. When the same value is received in the programme, it draws right next to the previous plot, which means the most left plot of the block represents actual interpreted time. By comparing SOF yellow dot and PTS block's left edge, shows the delay of receiving streaming data.

PTS data is calculated in a different way due to the overflow actions.

To change the graph scale, change the **#define** value in **window_manager.hpp.**

Buttons



Type > Payload Error
Type > Frame Error
Type > Frame Suspicious
Gives the explanation of error types.

Help > Usage
Gives each buttons' usage and methods.
Help > Application Info
Gives information about used libraries and versions.
Including this application version.

Error Frame Drop Down Combo Box:
When Error Frame is created, a drop down combo box for frame and payload is generated. Users can select a specific Error Frame to view its data in the following windows: Error Frame Data / Error Frame: Time & Payload Size Data / Summary / Payload Header Info.

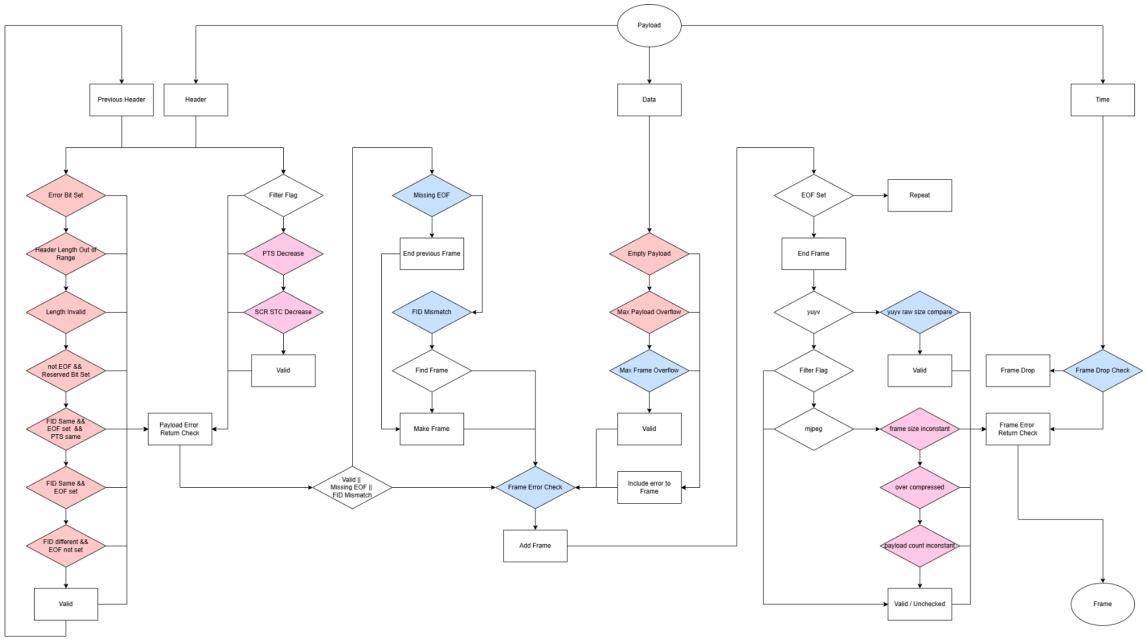
Error Payload Drop Down Combo Box:
It shows payload header information when errors occur. It shows payload errors for selected frames.

Suspicious Frame Drop Down Combo Box:
Suspicious Frame drop-down combo box is provided,

	enabling users to select a specific Frame to view related data in the same windows.
Show Image:	When Error Log or Suspicious Log button is pressed and Frame is selected, it shows the saved image.
Error Log:	To view saved log data, press Error Log button to switch screens.
Suspicious Log:	To view saved log data, press Suspicious Log button to switch screens.
Streaming Log:	To view currently streaming data information, press Streaming Log button to switch screens.
Capture On/Off:	Switch Off all three save image buttons or recall previous options.
Error Image:	Capture Error Frames.
Suspicious Image:	Capture Suspicious Frames.
Valid Image:	Capture Valid Frames.
Filter On/Off:	Switch Off all three capture filter buttons or recall previous options.
Irregular:	Detect Frame size drop below 10% for last three frames, or over 95% compressed frame, or payload count is smaller than the last three frames average.
PTS Decrease:	Detecting PTS field decrease, excluding overflow.
SCR STC Decrease:	Detecting SCR STC fields decreases, excluding overflow.
Play/Pause:	Play button to start receiving data, Pause to discard streaming data.
Quit:	Press to exit the application.

To change the default setting, modify static values in **uvcpheader_checker.cpp.**

How it works



When the UVCPHeaderChecker class receives payload data, it parses the header and validates it sequentially.

Red symbols:
These symbols are responsible for verifying the validity of the payload. Ensuring that the payload data conforms to expected standards and specifications before further processing. If an invalid payload is detected during frame processing,except for eof error and fid error, the erroneous payload data is discarded to prevent corruption or further issues in the overall data flow.

Blue symbols:
These symbols focus on validating the integrity of the entire frame, identifying any anomalies or inconsistencies.

Pink symbols:
These symbols are used to identify and handle suspicious data, but only when the suspicious flag is activated. Aggregating all suspicious data found within the frames, allowing for detailed analysis and corrective action as needed.

Payload Error Information:

ERR_NO_ERROR = 0	No error detected in the payload, valid.
ERR_EMPTY_PAYLOAD = 1	UVC Payload is empty, including the header.
ERR_MAX_PAYLOAD _OVERFLOW = 2	Payload size exceeds maximum transfer size.
ERR_ERR_BIT_SET = 3	BFH Error bit is set. Payload is invalid.
ERR_LENGTH_OUT_OF _RANGE = 4	HLE (Header Length Extension) is out of the valid range (2 to 12).
ERR_LENGTH_INVALID = 5	HLE length does not match with PTS (Presentation Time Stamp) or SCR (Source Clock Reference) settings: PTS = 0, SCR = 0, HLE = 2 PTS = 1, SCR = 0, HLE = 6 PTS = 0, SCR = 1, HLE = 8 PTS = 1, SCR = 1, HLE = 12
ERR_RESERVED_BIT _SET = 6	Reserved bit (RES) set to 1 when EOF (End of Frame) is 0. Some of the cameras set RES when EOF is set.
ERR_EOH_BIT = 7	– deleted logic
ERR_TOGGLE_BIT _OVERLAPPED = 8	– deleted logic
ERR_FID_MISMATCH = 9	FID (Frame Identifier) matches the previous frame while EOF is incorrectly set.
ERR_SWAP = 10	PTS matches the previous payload with the same FID, indicating a toggle bit error.
ERR_MISSING_EOF = 11	Missing EOF in the previous frame's payload.
ERR_UNKNOWN = 99	Unknown error.

Frame Error Information:

ERR_FRAME_NO _ERROR = 0	No frame error detected.

ERR_FRAME_DROP = 1	If FPS measurement indicates that the actual frames per second are lower than expected, the missing frames are classified as dropped frames.
ERR_FRAME_ERROR = 2	Occurs due to missing EOF or payload validation error, typically resulting in an incomplete frame.
ERR_FRAME_MAX _FRAME_OVERFLOW = 3	Occurs when the frame size exceeds the maximum defined size in the interface descriptor, often caused by dummy or erroneous payload data.
ERR_FRAME_INVALID _YUYV_RAW_SIZE = 4	Occurs when the YUYV frame size is not as expected (width * height * 2), which indicates a frame size mismatch.
ERR_FRAME_SAME _DIFFERENT_PTS = 5	– deleted logic
ERR_FRAME _MISSING_EOF = 6	Missing EOF in the frame, resulting in incomplete or invalid frame data.
ERR_FRAME _FID_MISMATCH = 7	FID (Frame Identifier) matches the previous frame while EOF is set.
ERR_ FRAME _UNKNOWN = 99	Unknown error.

Frame Suspicious Information:

SUSPICIOUS_NO _SUSPICIOUS = 0	No suspicious behavior detected.
SUSPICIOUS_PAYLOAD _TIME_INCONSISTENT = 1	Payload timestamps are inconsistent.
SUSPICIOUS_FRAME _SIZE_INCONSISTENT = 2	In MJPEG format frame sizes are inconsistent . Comparison is made with the last three frames, when frame data size is smaller than 90% of the average. Needs to be used when receiving static, unchanged image data
SUSPICIOUS_PAYLOAD _COUNT_INCONSISTENT	Number of payloads received is inconsistent with expectations. Comparison is made with the last three

= 3	frames. Needs to be used when receiving static, unchanged image data.
SUSPICIOUS _PTS_DECREASE = 4	PTS decreased unexpectedly, exclude overflow by ensuring the difference is smaller than half the range of the possible counter value (0x80000000).
SUSPICIOUS_SCR _STC_DECREASE = 5	SCR STC decreased unexpectedly, exclude overflow by ensuring the difference is smaller than half the range of the possible counter value (0x80000000).
SUSPICIOUS _OVERCOMPRESSED = 6	When the MJPEG frame image is smaller than 5% of raw data. (width * height *2 * 0.05)
SUSPICIOUS_ERROR _CHECKED = 97	It is already defined as an error.
SUSPICIOUS _UNKNOWN = 98	Unknown suspicious.
SUSPICIOUS _UNCHECKED = 99	Suspicion is not checked.

About the code

In this project, the file structure is organized as follows:

moncapwer.cpp is the main file. In this code there is different handling for windows urb and linux urb. Check out the difference if needed.

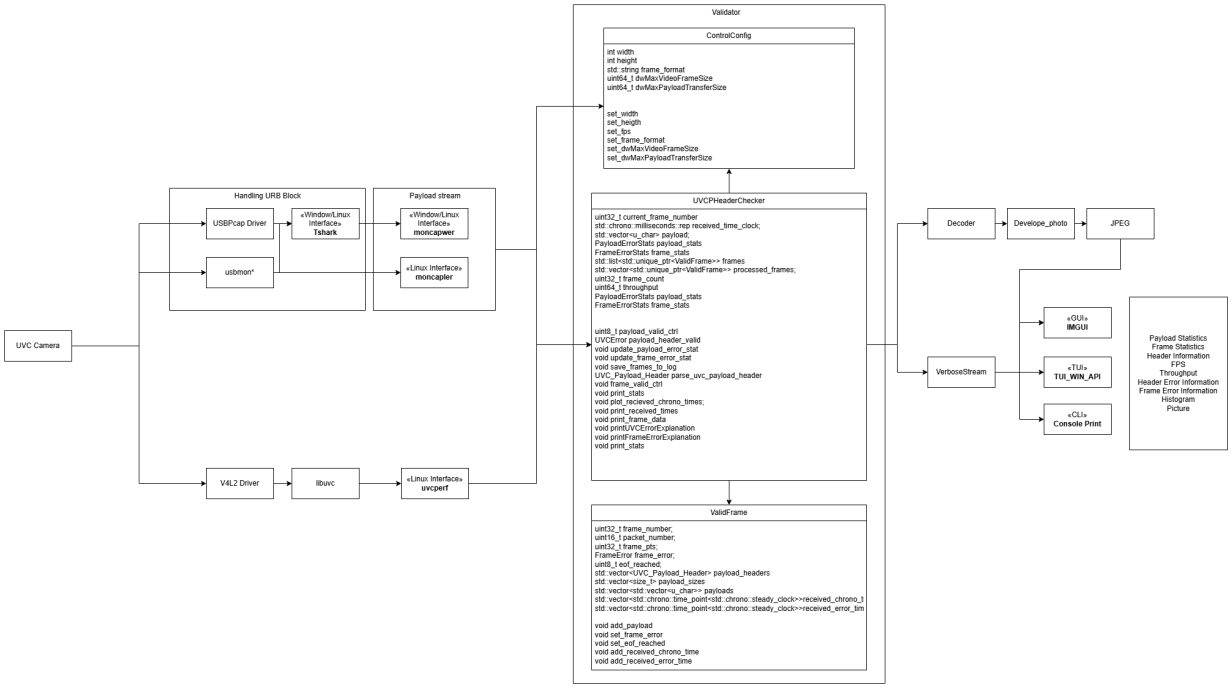
source/validuvc contains validators and controllers for description and SetCur values.

source/utils contains verbose controllers and streaming standard output, which provide the direction of the window number.

source/image_develope contains functionality for validating image data and saving it as a JPEG file.

source/libuvc & source/validuvc/linux contain the Linux interface, which uses usbmon to capture raw data and provide information to the validator. However, this requires control configuration data input, as the Linux URB does not provide such information. Which are not used in the Windows environment.

Diagram



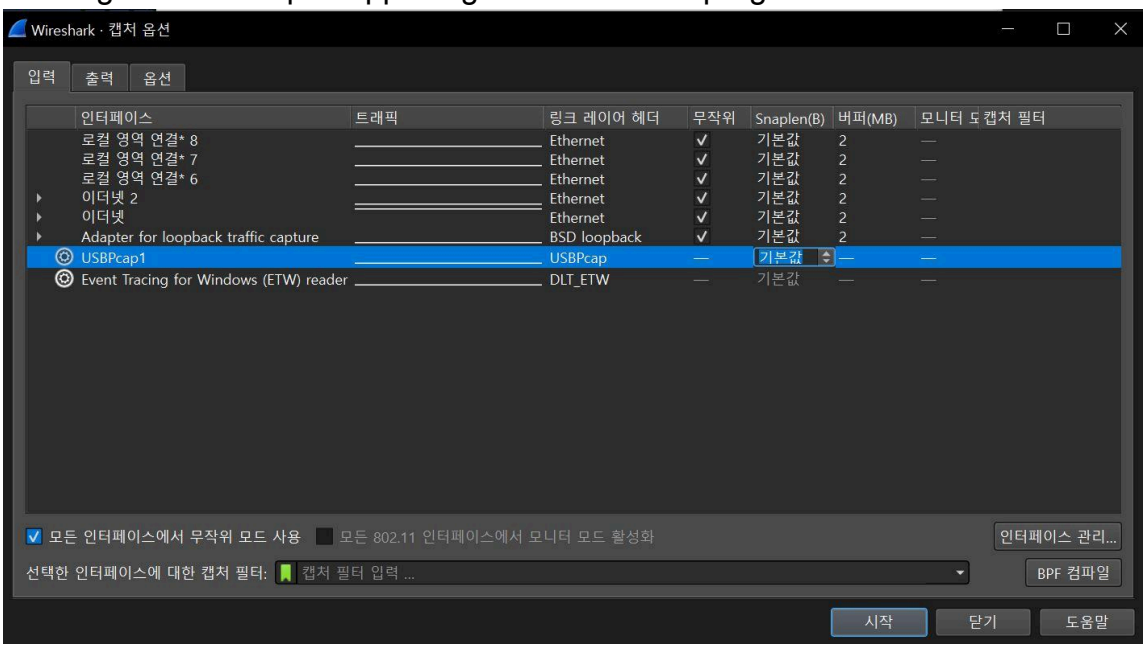
ICON

FAQ

Once the program starts, it may fail to receive streaming data, resulting in blank windows. Reconnecting the device could resolve the issue. Ensure you find the new device address after the device is plugged in again.

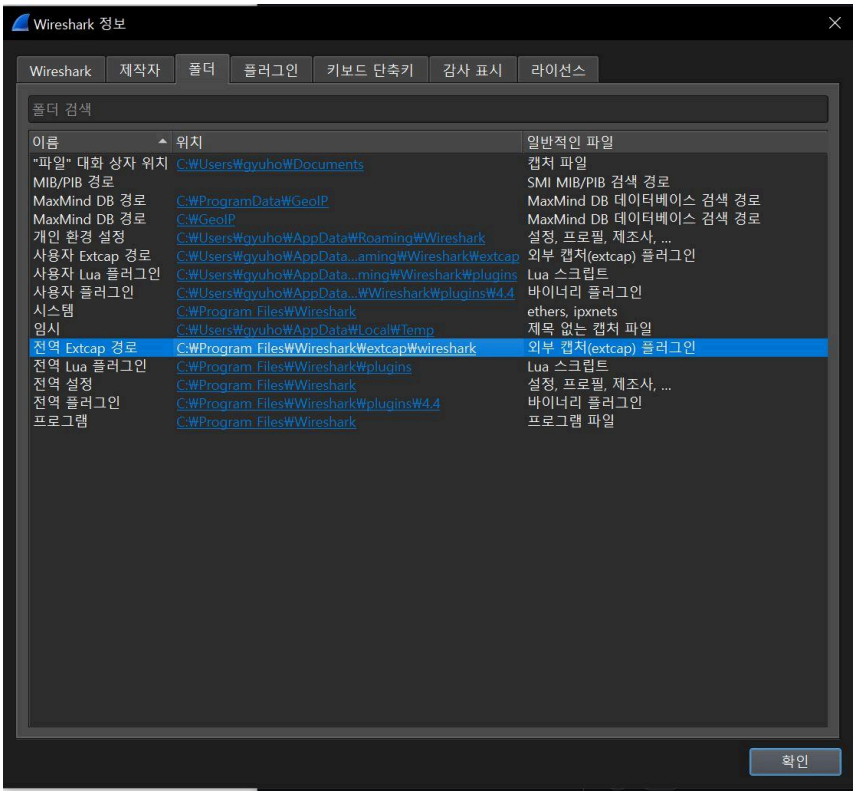
When images are not shown properly:
‘Invalid Image / Failed to load images. Image could be zero size or not found.‘
When the image is not shown on the screen, there could be three possible situations.

1. ‘images’ directory is not made Make sure there is images directory in the place you are running the programme. Or images will not be saved.
2. File format is not supported. Zero length data or wrong header format is being received. Therefore the image file is made but can not open since there is no data. You can check raw data files by xxd or hxd.
3. image is not created Image could not be made for some reason. Currently h.264 format is not provided. Or some error may be caused from developing the frame image. If this keeps happening this could be a programme error.



When using a saved pcapng format file to run the program, make sure to set snaplen(B) value as big as all of the data can be received, without drop. Can check or edit in

(*Wireshark* > *Capture* > *Option* > *Snaplen*). For the streaming data, make sure '-s 0' field is valid. If not, for the high quality video format, every frame may have EOF missing error.



In Windows OS, when using wireshark, please check USBPcapCMD is in the correct location.The path C:\ProgramFilesWireshark\extcap\wireshark might not be one of the default settings. Can check extcap settings in (*Wireshark* > *Help* > *Wireshark Info* > *Folders* > *Extcap*) path.

License

MIT License

Copyright (c) 2024 Vault Micro, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.