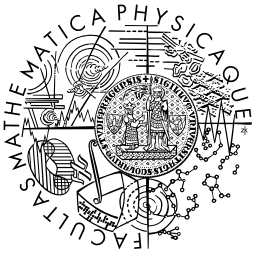


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Peter Lakatoš

Analyzátor USB paketů

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek, Ph.D.
Studijní program: Informatika
Studijní obor: Programování a softwarové systémy

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Poděkování.

Název práce: Analyzátor USB paketů

Autor: Peter Lakatoš

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Pavel Ježek, Ph.D., Katedra distribuovaných a spolehlivých systémů

Abstrakt: USB zbernica je dnes jedným z najrozšírenejších spôsobov pripojenia periférií k počítaču. Cieľom práce bolo vytvoriť software, ktorý analyzuje zachytenú komunikáciu medzi zariadnením pripojeným na danú zbernicu a počítačom.

Aplikácia následne rozumným spôsobom vizuálne zobrazuje zanalyzované dáta. Počiatočná verzia sa špecificky zameriava na HID triedu zariadení a ponúka aj sémantický význam jej úzkej podmnožiny do ktorej patria myš, klávesnica a joystick. Pri vizuálnej reprezentácii dát sa práca sa inšpiruje rôznymi dostupnými softwarmi, pričom rozlične kombinuje resp. dopĺňa ich vlastnosti a implementuje z nich tie, ktoré vníma ako najlepšie riešenie v danej situácii.

Ďalšie vlastnosti aplikácie sú napríklad parsovanie HID Report Descriptoru vďaka ktorému je jednoduchšie pridať sémantickú analýzu rôznym ďalším HID zariadeniam. Celkový návrh aplikácie by mal ponúknuť možnosť budúcej implementácie ďalších USB tried pre prípadné rozšírenie.

Klíčová slova: USB HID

Title: USB Packet Analyzer

Author: Peter Lakatoš

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Pavel Ježek, Ph.D., Department of Distributed and Dependable Systems

Abstract: Abstract.

Keywords: key words

Obsah

1. Úvod

Možnosti pripojenia rôznych periférií k zariadeniu sú v dnešnej dobe rozsiahle. Aj napriek tomu, že technológia každým dňom napreduje a svet sa uberá viac bezdrôtovým smerom, je USB stále najrozšírenejší sériový spôsob prenášania dát. Už z názvu „Universal Serial Bus“ je jasné, že rozpätie zariadení, ktoré možno k tejto zbernici pripojiť je obrovské. Práve preto je USB protokol jeden z najkomplexnejších protokolov ktoré sa využívajú na komunikáciu.

V tejto práci sa pozrieme na užšiu podmnožinu USB protokolu a na komunikáciu s dopredu vybratými perifériami, konkrétne myšou, klávesnicou a joystickom.

1.1 USB

vysvetlenie zakladnych pojmov spojenych USB: historia, usb port/conector, plug and play(https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/introductio to-plug-and-play), host-master, low/full/high speed zariadenia

1.2 Existujúce aplikácie

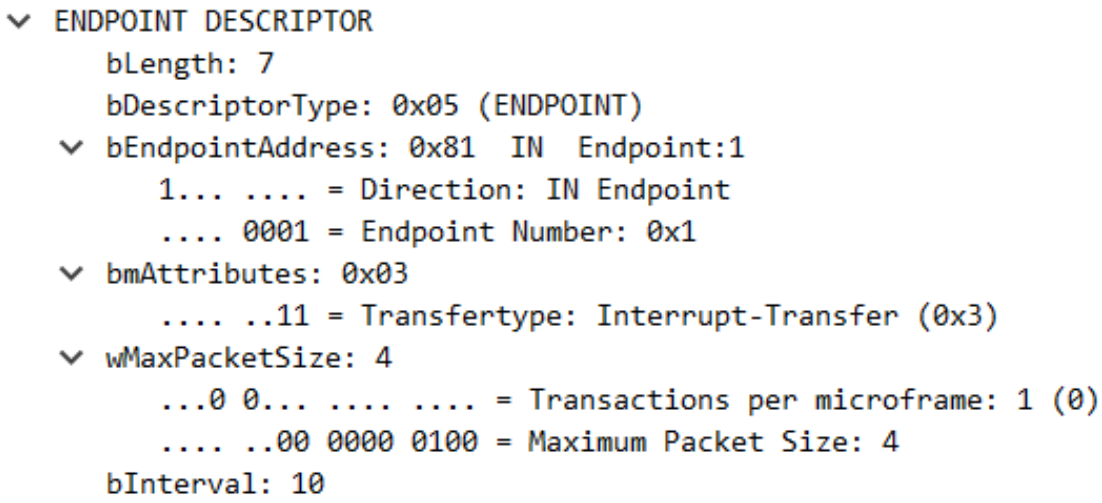
Momentálne existuje niekoľko známych aplikácií ktoré slúžia na analýzu USB paketov. V tejto kapitole si ich zopár ukážeme, pričom mnohé z nich nám poslúžili ako inšpirácia pri ppísaní našej práce a riešení konkrétnych problémov na ktoré sa pozrieme v nasledujúcich kapitolách.

Je nutné upozorniť, že väčšina dnešných analyzátorov sú platené aplikácie, prípadne majú odomknuté len základné vlastnosti s možnosťou dokúpenia si plnej verzie. Práve preto pri ich prípadnom porovnávaní budeme brať do úvahy len funkcionalitu, ktorá je dostupná zadarmo.

Wireshark

Pravdepodobne najznámejšia third-party aplikácia na analýzu paketov. Jeho funkcionalita je veľmi rozsiahla, a vzhľadom na to, že sa jedná o open-source projekt, neustále rastie. Zameriava sa hlavne na analýzu sieťových paketov. Napriek tomu podporuje spoluprácu s rôznymi inými sniffermi. Jeden z takýchto snifferov je USBPcap ktorý zachytáva USB komunikáciu, a tak nie je prekvapivé, že Wireshark podporuje analýzu paketov aj nad touto zbernicou.

Vzhľadom na obľúbenosť a rozsiahlosť programu, nám Wireshark poslúžil ako referenčná aplikácia, z ktorej sme čerpali celkovú inšpiráciu na funkcie, ktoré by mal bežný analyzátor paketov spĺňať. Medzi tie úplne základné patrí napríklad hexdump dát nad ktorými prebieha analýza, ale napríklad aj spôsob vyobrazenia rôznych deskriptorov, ktoré je riešené cez stromovú štruktúru. Medzi viac špecifické funkcie ktoré sme neskôr implementovali aj v našom programe patrí napríklad detailnejšie vyobrazenie jednotlivých bytov a ich význam, ako je možné vidieť nižšie na obrázku 1.1. Túto vlastnosť aj napriek jej využitiu mnohé konkurenčné aplikácie postrádajú.



Obrázek 1.1: Ukážka vyobrazenia jednotlivých bytov.

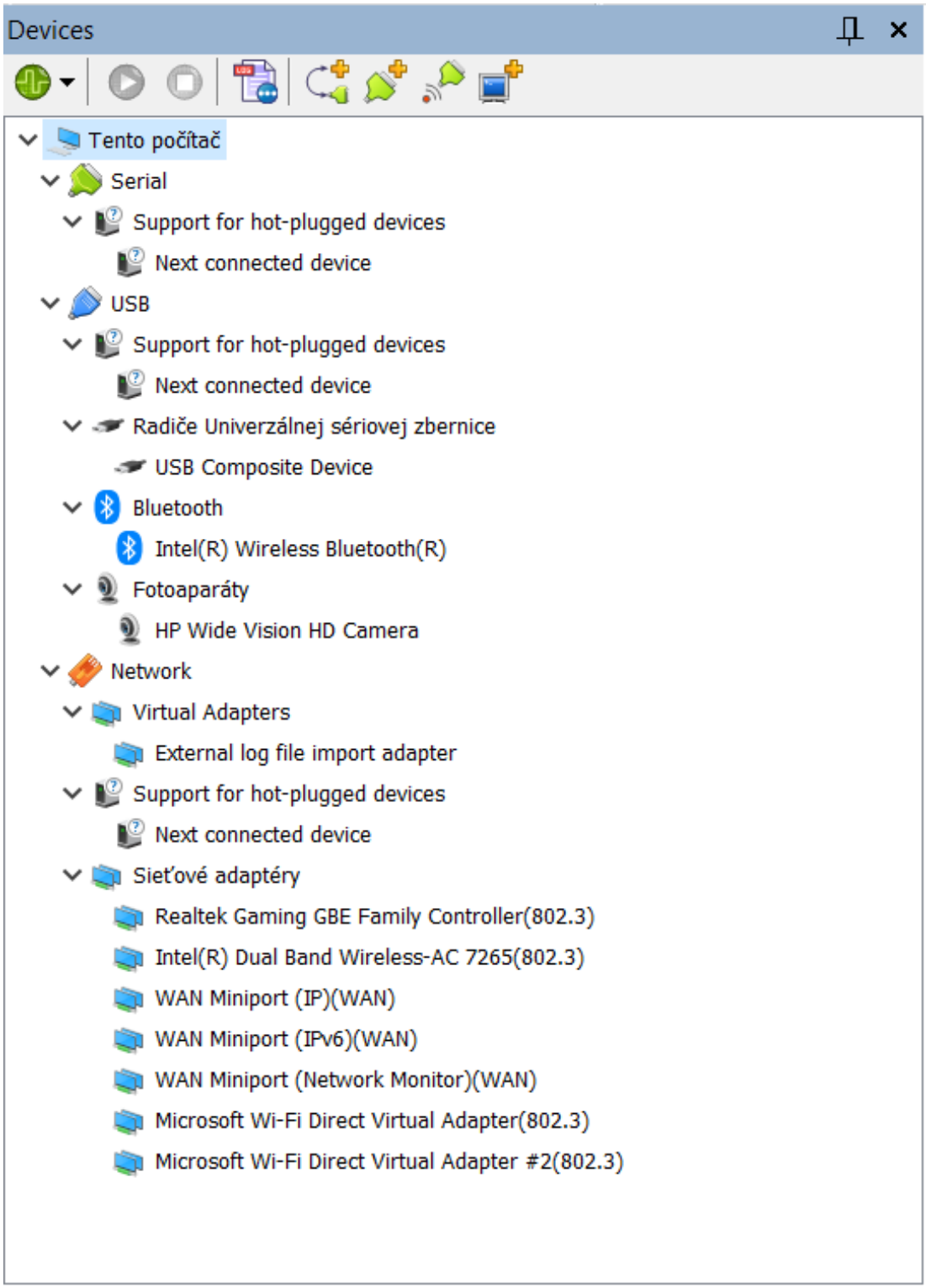
Jeho výhoda je hlavne v tom, že podporuje širokú škálu deskriptorov a plná verzia programu je dostupná úplne zadarmo. Z pohľadu užívateľa je až prekvapivé, že aj napriek rozsiahlosti programu je aplikácia veľmi user-friendly orientovaná a dopĺňa ju veľmi intuitívne užívateľské rozhranie.

Device Monitoring Studio

Plná verzia aplikácie je platená. Verzia zadarmo ponúka analýzu sieťových a USB paketov, tak ako aj analýzu komunikácie prebiehajúcej cez sériový port.

Ako prvé na aplikácii zaujme spôsob zvolenia si zariadenia s ktorým bude sledovaná komunikácia. Je implementovaný štýlom stromovej štruktúry ako je ukázané na obrázku 1.2 nižšie. Už základná verzia programu poskytuje veľmi rosiahlu funkcionalitu. Používateľovi je umožnené posielať zariadeniu požiadavky definované v USB špecifikácii(ODKAZ). Aplikácia taktiež zobrazuje sémantickú analýzu niektorých HID zariadení, ale nie v práve najlepšej podobe.

Na základe výstupu sa dá povedať, že sémantická analýza je naimplementovaná skôr obecne a pri niektorých položkách je namiesto ich významu napísané „Unknown“. Taktiež nie je veľmi jasné odkiaľ sa dané hodnoty berú, keďže k nim chýba ich dátová reprezentácia. Medzi zachytenými paketami sa prvotne nezobrazujú tie, ktoré označujú nakonfigurovanie daného zariadenia (je nutné ho odpojiť a znova napojiť počas monitorovania). Užívateľské rozhranie je veľmi chaotické a chvíľu trvá, kým človek nájde čo i len základné informácie ako napríklad hlavičky ku jednotlivým paketom. Nepoteší ani fakt, že verzia zadarmo nedovoľuje monitorovanie dlhšie ako 10 minút a maximálny počet monitorovaní za jeden deň je taktiež 10.



Obrázek 1.2: Ukážka stromovej štruktúry na zvolenie si zariadenia s ktorým bude zachytávaná komunikácia.

1.3 Požadované funkcie

Na základe minulých príkladov už existujúcich aplikácií sme si mohli všimnúť, že všetky obsahujú niekoľko základných funkcií, ktoré by mal obsahovať každý analyzátor. V tejto sekcii zhrnieme funkcie, ktoré sme si zvolili pre našu aplikáciu. Zo základných funkcií sme si vybrali tie, ktoré považujeme za absolútne nevyhnutné pre každý analyzátor. Následne sme sa niekoľko z nich pokúsili akýmsi spôsobom vylepšiť tak, aby maximalizovali ich využiteľnosť a zároveň čo najlepšie zlepšovali prácu s aplikáciou jej užívateľovi. Zároveň ich dopĺňame o pokročilejšie funkcie z ktorých sú niektoré viac špecificky zamerané.

Základné

Výsledná aplikácia by teda mala splňovať nasledujúce zákadné požiadavky:

P1 Mala by byť schopná analyzovať USB pakety zachytené v **pcap** formáte pomocou **USBPcap** snifferu.

P2 Mala by podporovať sémantickú analýzu pre všetky základné USB descriptory spomenuté v USB 2.0 dokumentácii(ODKAZ kapitola 9.6) (ako napríklad *Device descriptor*, *Interface descriptor*, *Endpoint descriptor*,...)

P3 Mala by vedieť rozumne zobraziť dáta ktoré **USBPcap** zachytí a uloží. Pod pojmom rozumne myslíme spôsob zobrazenia pomocou hexdumpu.

P4 Mala by na prvý pohľad jasne zobraziť základné informácie o každom analyzovanom pakete (ako napr. dĺžka paketu, typ prenosu, ...) a pri bližšom skúmaní jednotlivých paketov detailnejšie zobraziť celú jeho hlavičku.

P5 Detailnejšie informácie o pakete budú zobrazované na základe interakcie užívateľa s aplikáciou.

Pokročilejšie

Zároveň sme aplikáciu doplnili o niekoľko pokročilejších a špecificky zameraných požiadaviek:

P6 Mala by určitým spôsobom uľahčiť používateľovi orientáciu v hexdumpe.

P7 Mala by byť schopná rozparsovať **HID Report Descriptor** takým štýlom, aby bolo neskôr možné sématnicky reprezentovať input určitých HID zariadení

P8 Mala by byť schopná vhodným spôsobom vizuálne zobraziť sémantický význam dát posielaných danou podmnožinou HID zariadení do ktorej patrí myš, klávesnica a joystick.

P9 V miestach kde to dáva zmysel, by aplikácia mala byť schopná zobrazovať význam dát až na úrovni jednotlivých bitov.

1.4 Ciele práce

vytvorit aplikaciu navrh musi byt dostatocne obecny aby sa dal rozsirit o dalsie USB protokoly dostatocne obecny navrh pre lahke pridavanie novych HID zariadeni prehladny interface

2. USB a Windows

2.1 USB zbernica

Plug and Play device tree(sposob akym si windows udrziava strom zariadeni na zbernici)(https://docs.microsoft.com/sk-sk/windows-hardware/drivers/gettingstarted/devic nodes-and-device-stacks)

2.2 Device object a device stack

PDO,FDO, Device object(https://docs.microsoft.com/en-us/windows-hardware/drivers/ker to-device-objects) https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/creatin a-device-object

2.2.1 Drivery

windows driver model(WDM) : https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/types-of-wdm-drivers bus driver(https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/bus-drivers), function driver(https://docs.microsoft.com/ us/windows-hardware/drivers/kernel/function-drivers) a filter driver(https://docs.microsoft.co us/windows-hardware/drivers/kernel/filter-drivers)

2.3 Komunikacia s USB zariadenim

sposob komunikacie operacneho systemu so zariadenim pripojenym na USB zbernicu : IRP(https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/i-o-request-packets) , URB (https://docs.microsoft.com/en-us/windows-hardware/drivers/usbco with-a-usb-device) a pod. https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/ irps

2.4 USB descriptory

opis zakladnych USB descriporov, hlavne tych ktore neskor aj vyuzivam v program(Device, Interface, Endpoint, Configuration, String, Setup) : https://docs.microsoft.com/e us/windows-hardware/drivers/usbcon/usb-descriptors https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/usb-control-transfer

2.4.1 Rozlozenie USB zariadenia z hladiska descriptorov

https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/usb-device-layout

2.5 HID zariadenia

hid zariadenie obecne, priklady https://docs.microsoft.com/en-us/windows-hardware/drivers/hid/

2.5.1 Reporty

Input/Output/Feature reporty.

2.5.2 Report Descriptor

Opis report descriptoru, k comu sluzi, pripadne ako z neho vycitat zaujimave data (neskor vyuzite v programe pri parsovani HID Report Descriptoru na naslednu semanticku analyzu dat ktore posiela zariadenie)

3. Analýza

3.1 Získanie USB packetov

3.1.1 Windows exclusive mód

opisat co to je, a dolezite je spomenut, ze windows otvara v exclusive mode zakladne HID zariadenia ako mys a klavesnica

3.1.2 Známe knižnice

opisat zakladne kniznice na sledovanie USB zbernice a preco som ich nemohol pouzit : libUSB, hidAPI, moufiltr, SetupAPI, WinUSB

3.1.3 Third-party aplikácie

opisat odkial nakoniec ziskavam packety - USBPcap a Wireshark

3.2 Spracovávanie pcap súborov

moznosti ako citat pcap subory : bud pouzit uz existujucu kniznicu : na linuxe Libpcap, windows NPcap(deprecated WinPcap), alebo citat subory manualne : std::istream alebo QFile

3.3 Sémantická reprezentácia dát

ako si z dat vytiahnut udaje ktore su potom pouzite na semanticku analyzu implementovanych HID zariadeni : HID Report parser, InputValues a EndpointDevice struct. Nasledne sparovanie - ako vybrat spravny report pre konkretny input

3.4 Voľba frameworku

obecne co by som od toho GUI priblizne chcel, potom opisat preco som si vybral prave Qt a v nasledujucich kapitolach opisat rozhodnutia uz v Qt dovod preco som si zvolil qt namiesto inych c++ GUI frameworkov(napriklad sfml)

3.5 Zobrazenie základných informácií

ako zobrazovat zakladne info o packete : pouzit QListWidget alebo QTableWidget (pripadne nieco ine ako nejaky abstract viewmodel), narok na zakladne funkcionality : lahka rozsiritenlnost o dalsie ”stlpceky” , moznost jednoduchej interakcie(doubleClick na polozku). Mat vsetky info na jednom okne / mat pop-up okna.

3.6 Zobrazenie sémantického významu dát

ako vyzobrazit semanticky vyznam roznych dat - descriptory, usb header, vyznam input dat roznych HID zariadeni

3.7 Hexdump

ako v qt urobit hexdump - do coho zobrazovat data(vytvorit si vlastny viewer dedeny od QAbstractScrollArea, pripadne niecoho ineho) vs najst nieco co uz v qt je a upravit to aby to sedelo poziadavkam. Vziat do uvahy bezne funkcie hexdumpu : selection mody(oznacit naraz hexa a im odpovedajuce printable), logicke oddelenie dat(napriklad farbami)

4. Vývojová dokumentácia

4.1 Architekrúra aplikácie

4.2 Jadro aplikácie

4.2.1 USB_Packet_Analyzer

riadi celkovy beh programu, reaguje na input od uzivatela

4.2.2 Item Manager

spracovanie samostatneho packetu a ulozenie dat o nom

4.2.3 DataViewer

trieda ktora ma na starosti vyskakovacie okno po dvojkliku a item a nasledne reaguje na input od uzivatela v okne

4.2.4 TreeItem

reprezentuje jednotlive nody v stromovej strukture ktora sa potom vyuziva na zobrazenie dat v QTreeView

4.3 Modely

4.3.1 AdditionaldataModel

model na spravovanie zvysnych dat(data ktore nie su sucastou hlavicky packetu)

4.3.2 ColorMapModel

vyobrazenie pomocnej mapy na lepsie sa zorientovanie v zvyraznemom hexdumpe

4.3.3 DataViewerModel

model na hexdump - prenasa hex/printable a zaroven o co vlastne ide(konkretny descriptor, interrupt data, ...)

4.3.4 TreeItemBaseModel

model na QTreeView ktorz vyuziva TreeItem

4.3.5 USBPcapHeaderModel

model na QTreeView ale specialne pre USBPcap hlavicku packetu

4.4 Interpretery

4.4.1 BaseInterpreter

abstractna trieda od ktorej dedia vsetkz interpretery

4.4.2 Interpreter factory

facory trieda na pridelenie konkretneho interpreteru za runtimu kvoli jednoduchosti na lepsie rozsirenie programu do buducnosti

4.4.3 Interpretery descriptorov

Config,Device,Setup,String,...

4.4.4 Interrupt transfer interpretery

obecne interrupt transfer interpreter - sluzi skor ako factory na rozne doteraz implementovane HID zariadenia

Joystick interpreter

Mouse interpreter

Keyboard interpreter

4.5 Delegáti

DataViewerDelegate

Qt delegat - stara sa o highlight hexdumpu

4.6 HID

4.6.1 HIDDevices

staticka trieda, drzi vsetky rozpoznane HID zariadenia a obsahuje funkcie specificke nich - parsovanie HID Report descriptoru

4.7 Práca so súbormi

4.7.1 FileReader

praca zo suborom a predavanie precitanych dat, offline/online capture, QFile vs std::istream

4.8 Globálne dáta

4.8.1 ConstDataHolder

staticka trieda na drzanie si konstant ktore su potrebne napriec celym programom. Mapovanie z enumu do jeho stringovej reprezentacie

4.8.2 PacketExternStructs

obsahuje definiciu vsetkych dolezitych USBPcap structov, pcap structov, enumov a vsetkych structov ktore pouzivam v aplikacii

5. Možnosti rozšírenia

Rozobrať čo všetko sa dá urobiť s tými dátami, ktoré už mám uložené v pamati, ale momentálne sa s nimi nič nedeje

5.1 Ukladanie výstupu do súboru

výstup analýzy do súboru(textového)

5.2 Iná vizuálna reprezentácia dát

Momentálne vyzobrazujem dáta prevažne v QTreeView alebo QTableView, ale vdaka tomu ako ich mám uložené + to že nad nimi operuje nejaký model ktorý vie vrátiť dáta na základe indexu, by nemuselo byť taká zložité pridať inú vizualizáciu dát(napríklad obrázkovú ako tu : https://www.usbmadesimple.co.uk/ums_5.htm)

5.3 Pridávanie nových interpreterov pre descriptory

pridanie nových druhov descriptorov - pridať nový interpreter do factory

5.4 Pridanie intepreteru na interrupt tranfser

pridanie analyzy interrupt transferu aj pre ine ako hid zariadenia

5.4.1 Pridanie nových HID zariední

nove HID zariadenie - pridanie do interrupt "factory"

5.5 Pridanie analýzy pre isochronous a bulk transfer

semanticka analyza aj inych ako interrupt alebo control transferov - momentalne su rozpoznavane len v hexdumpe

5.6 ?Možnosť rozšírenia na iné platformy?

uprava aplikacie aby bola prenositelna aj na ine platformy, co vsetko by tam bolo treba upravit(pravdepodobne nie vela, kedze qt je prenosne, a prakticky jedine co pouzivam spojene s windowsom su jeho structy na rozne descriptory)

6. Užívateľská dokumentácia

6.1 Inštalácia

nastavenie celkovej aplikácie, ale aj nainstalovanie USBPcap + wireshark a ich kombinácia pre live capture

6.2 Orientácia v GUI aplikácie

popis k jednotlivým tlačidlám gui

6.3 Používanie aplikácie

ako spustit live/offline capture, a celkovo ako pracovať s aplikáciou(popis funkcií - doubleClick na item => zobrazi sa pop-up okno s bližšou analýzou)

7. Záver

7.1 Zhrnutie

celkove zhrnutie prace, ?praca s Qt?

7.2 Budúce plány

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

Přílohy

.1 První příloha