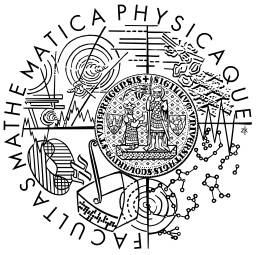


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

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 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 **3**
- 1.1 Základné pojmy . . . 3
- 1.2 Existujúce aplikácie . . . 3
- 1.3 Požadované funkcie . . . 8
- 1.4 Ciele práce . . . 9

2 USB a Windows **10**
- 2.1 USB zbernica . . . 10
- 2.2 Device object a device stack . . . 10
  - 2.2.1 Drivery . . . 10
- 2.3 Komunikacia s USB zariadenim . . . 10
- 2.4 USB descriptory . . . 10
  - 2.4.1 Rozlozenie USB zariadenia z hladiska descriptorov . . . 10
- 2.5 HID zariadenia . . . 11
  - 2.5.1 Reporty . . . 11
  - 2.5.2 Report Descriptor . . . 11

3 Analýza **12**
- 3.1 Získanie USB packetov . . . 12
  - 3.1.1 Windows exclusive mód . . . 12
  - 3.1.2 Známe knižnice . . . 12
  - 3.1.3 Third-party aplikácie . . . 12
- 3.2 Spracovávanie pcap súborov . . . 12
- 3.3 Sémantická reprezentácia dát . . . 12
- 3.4 Voľba frameworku . . . 12
- 3.5 Zobrazenie základných informácií . . . 12
- 3.6 Zobrazenie sémantického významu dát . . . 13
- 3.7 Hexdump . . . 13

4 Vývojová dokumentácia **14**
- 4.1 Architekrúra aplikácie . . . 14
- 4.2 Jadro aplikácie . . . 14
  - 4.2.1 USB_Packet_Analyzer . . . 14
  - 4.2.2 Item Manager . . . 14
  - 4.2.3 DataViewer . . . 14
  - 4.2.4 TreeItem . . . 14
- 4.3 Modely . . . 14
  - 4.3.1 AdditionaldataModel . . . 14
  - 4.3.2 ColorMapModel . . . 14
  - 4.3.3 DataViewerModel . . . 14
  - 4.3.4 TreeItemBaseModel . . . 14
  - 4.3.5 USBPcapHeaderModel . . . 14
- 4.4 Interpretery . . . 15
  - 4.4.1 BaseInterpreter . . . 15
  - 4.4.2 Interpreter factory . . . 15

4.4.3 Interpretery descriptorov . . . 15
4.4.4 Interrupt transfer interpretery . . . 15
4.5 Delegáti . . . 15
4.6 HID . . . 15
4.6.1 HIDDevices . . . 15
4.7 Práca so súbormi . . . 15
4.7.1 FileReader . . . 15
4.8 Globálne dáta . . . 16
4.8.1 ConstDataHolder . . . 16
4.8.2 PacketExternStructs . . . 16

5 Možnosti rozšírenia **17**
5.1 Ukladanie výstupu do súboru . . . 17
5.2 Iná vizuálna reprezentácia dát . . . 17
5.3 Pridávanie nových interpreterov pre descriptory . . . 17
5.4 Pridanie intepreteru na interrupt tranfser . . . 17
5.4.1 Pridanie nových HID zariední . . . 17
5.5 Pridanie analýzy pre isochronous a bulk transfer . . . 17
5.6 ?Možnosť rozšírenia na iné platformy? . . . 17

6 Užívateľská dokumentácia **18**
6.1 Inštalácia . . . 18
6.2 Orientácia v GUI aplikácie . . . 18
6.3 Používanie aplikácie . . . 18

7 Záver **19**
7.1 Zhrnutie . . . 19
7.2 Budúce plány . . . 19

Seznam obrázků **20**

Seznam tabulek **21**

Seznam použitých zkratek **22**

Přílohy **23**
.1 První příloha . . . 24

1. Úvod

USB je najrozšírenejší sériový spôsob prenášania dát. Vznikol v druhej polovici 90. rokov 20. storočia kedy sa na pripojenie periférií k počítaču používalo viacero rozličných portov (sériový a paralelný port, PS/2, ...). Na pripojenie základných zariadení ako myš alebo klávesnica slúžil napríklad port PS/2. Jeden z najznámejších sériových portov RS-232 zase mohol slúžiť na pripojenie tlačiarne. USB vzniklo za účelom nahradiť a zjednotiť tieto spôsoby pripojenia bežných periférií k počítaču. Zároveň ale ponúklo aj možnosť prenosu dát zo zariadení ako externé HDD. Z toho vylýva, že USB protokol je jeden z najkomplexnejších protokolov na komunikáciu aký existuje. Pre účel lepšieho pochopenia daného protokolu alebo v prípade implementácie vlastného zariadenia nám môžu pomôcť tvz. USB paket analyzátory. Ich zmysel je určitým spôsobom priblížiť a vyobraziť komunikáciu na danej zbernici. Cieľom tejto práce bude práve taktýto analyzátor zostrojiť.

1.1 Základné pojmy

V tejto sekcii si vysvetlíme niektoré základné pojmy ktoré budeme neskôr v texte používať.

Master/slave

Paket

URB

Analyzátor vs sniffer

USB zariadenie

+HID

Typy prenosov

control,interrupt,isochornous,bulk (str. 20,21)

USB deskriptor

1.2 Existujúce aplikácie

Našu aplikáciu chceme hlavne zamerať na analýzu základných USB deskriptorov a komunikáciu so základnými perifériami. Konkrétne na užšiu podmnožinu HID zariadení ako sú myš, klávesnica a joystick.

Momentálne existuje niekoľko známych aplikácií ktoré slúžia na analýzu USB paketov. V tejto kapitole si ich zopár ukážeme a rozoberieme si niektoré ich funkcie a spôsob, akým analyzujú dané pakety.

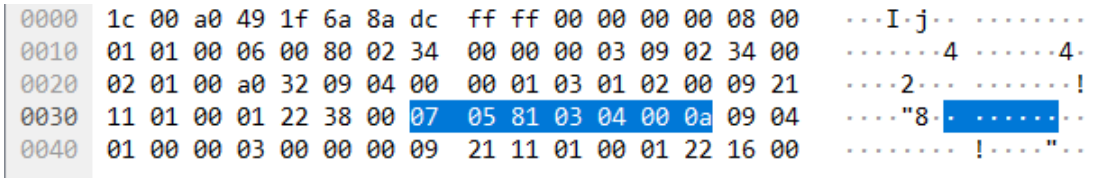
Je nutné upozorniť, že väčšina dnešných analyzátorov sú platené aplikácie, prípadne majú odomknuté len základné vlastnosti s možnosťou dokúpenia si pl-

nej verzie. Práve preto sme nemali možnosť si pri všetkých vyskúšať ich celú funkcionalitu a na niektoré platené funkcie máme len veľmi high-level pohľad.

Wireshark

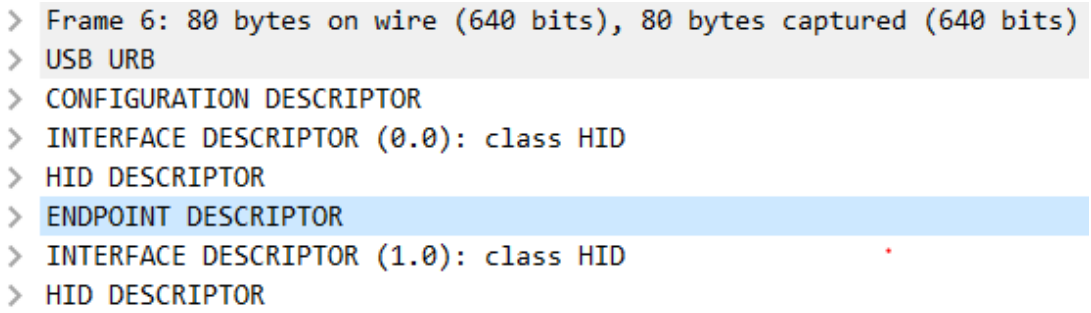
Pravdepodobne najznámejšia third-party aplikácia na analýzu sieťových paketov. Jeho funkcionalita je veľmi rozsiahla, a vzhľadom na to, že sa jedná o open-source projekt, neustále rastie. Napriek tomu, že sa zameriava hlavne na sieťové pakety, podporuje spoluprácu s rôznymi inými sniffermi. Jeden z takýchto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

Teraz si postupne ukážeme niektoré funkcie Wiresharku na konkrétnom príklade (presnejšie na zachytenej komunikácii s myšou). Medzi tie úplne základné určite patrí hexdump dát nad ktorými prebieha analýza, ktorý je vyobrazený na obrázku 1.1.



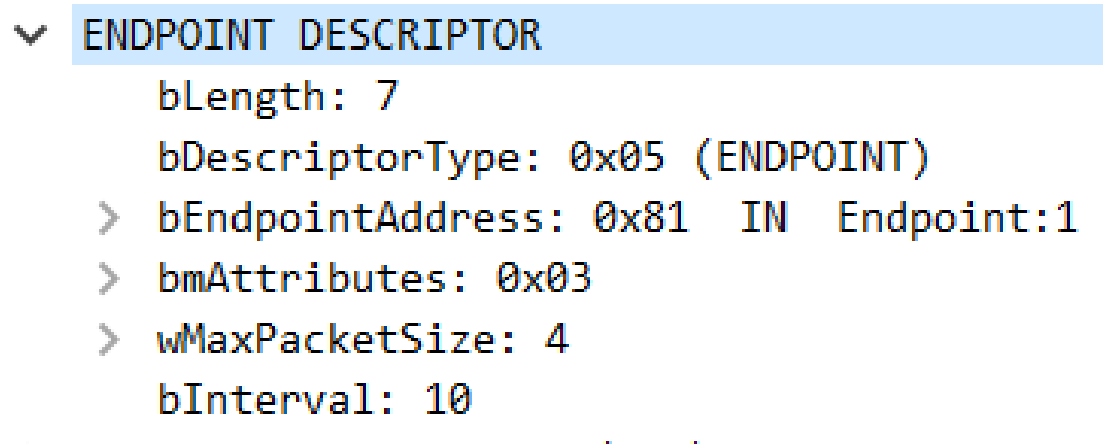
Obrázek 1.1: Ukážka hexdumpu so zvýrazneným endpoint deskriptorom.

Tento hexdump je tvorený dátami z jedného control prenosu, kde zariadenie posiela informácie o sebe v podobe rôznych deskriptorov. Pri pohybe myšou nad daným hexdumpom ponúka Wireshark interaktívnu odozvu, pričom farebne oddeľuje jednotlivé byty podľa ich významu. Zvýraznené byty na predchádzajúcom obrázku reprezentujú jeden endpoint deskriptor. Ďalšia užitočná vlastnosť je označenie nie len číselnej reprezentácie dát, ale takisto im odpovedajúcim tlačiteľným znakom. Tie isté dáta sa ale dajú reprezentovať viacerými spôsobmi. Môžu byť vyobrazené pomocou stromovej štruktúry, ktorá už jednotlivým bytom pridáva ich sémantický význam v slovnom tvare ako je ukázané na obrázku 1.2 nižšie.



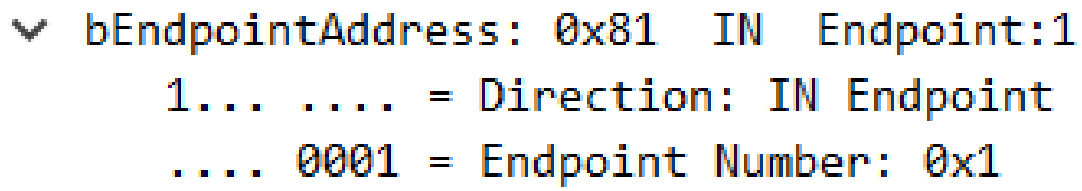
Obrázek 1.2: Ukážka reprezentácie dát pomocou stromovej štruktúry.

Jednotlivé položky si môžeme bližšie rozbaliť. Napríklad, vyššie zvýraznených 7 bytov reprezentujú endpoint deskriptor s konkrétnymi hodnotami čo ukazuje obrázok 1.3.



Obrázek 1.3: Endpoint deskriptor reprezentovaný dátami zvýraznenými na obrázku vyššie.

Medzi viac špecifické funkcie patrí detailnejšie vyobrazenie jednotlivých bytov a ich význam, ako je možné vidieť nižšie na obrázku 1.4. Túto vlastnosť aj napriek jej využitiu mnohé konkurenčné aplikácie postrádajú.



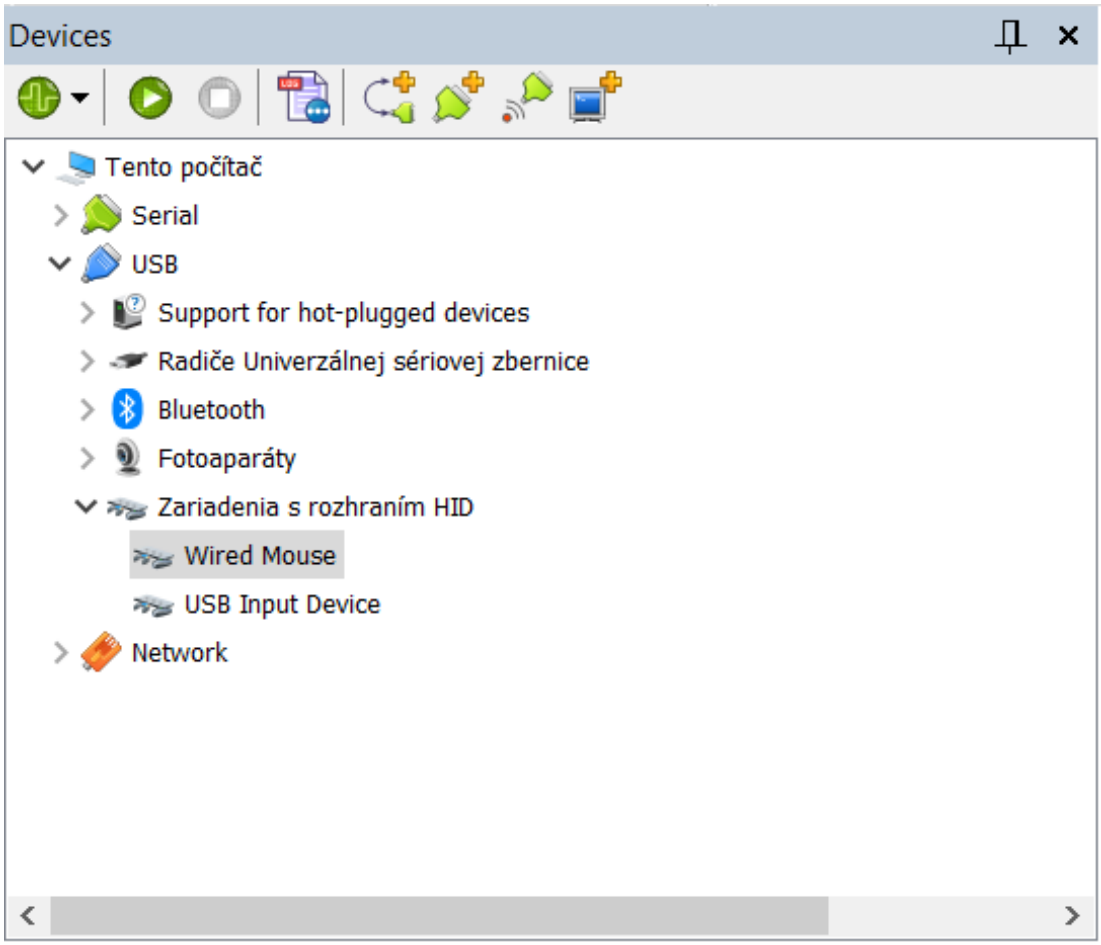
Obrázek 1.4: Ukážka vyobrazenia jednotlivých bytov.

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

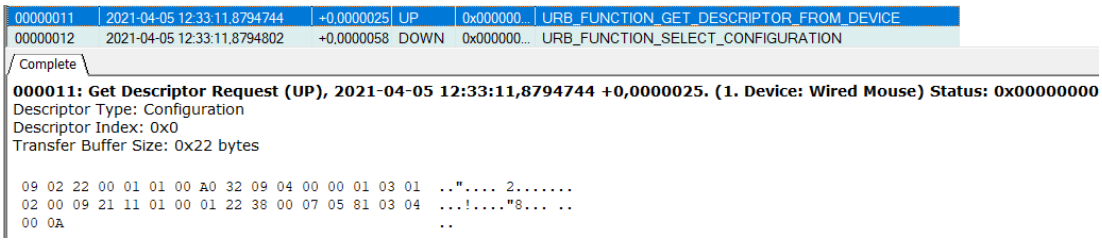
Aplikácia ponúka analýzu sieťových a USB paketov, tak ako aj analýzu komunikácie prebiehajúcej cez sériový port.

Ako prvé na aplikácii zaujme spôsob zvolenia si zariadenia s ktorým bude sledovaná komunikácia. Je implementovaný štýlom stromovej štruktúry ako je ukázané na obrázku 1.5 nižšie, kde máme konkrétne označenú rovnakú myš s ktorou komunikáciu sme sledovali predchádzajúcim programom.



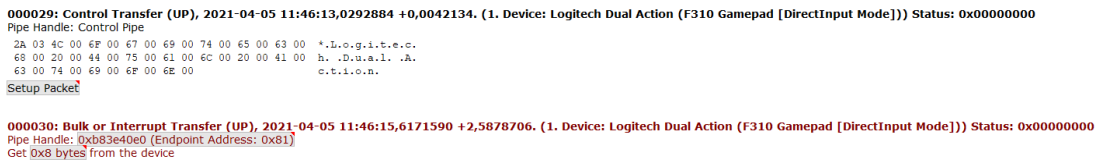
Obrázek 1.5: Ukážka stromovej štruktúry na zvolenie si zariadenia s ktorým bude zachytávaná komunikácia.

Základná verzia programu ponúka vizuálne zobrazenie *URB*, tak ako aj analýzu jednotlivých paketov. Pod analýzou si tu môžeme predstaviť ale len obyčajný hexdump ktorý neposkytuje žiadne významové oddelenie dát, tým pádom je obtiažnejšie sa v ňom zorientovať. Takisto tu nemáme žiadne sémantické vysvetlenie čo znamenajú poslané dáta (napríklad pomocou stromovej štruktúry ako to rieši konkurencia). Príklad takejto analýzy môžeme vidieť na obrázku 1.6 nižšie.



Obrázek 1.6: Príklad analýzy paketov.

Vyobrazenie URB (obrázok 1.7) tak ponúka trochu obecnejší prehľad o tom, čo sa na danej zbernici deje.

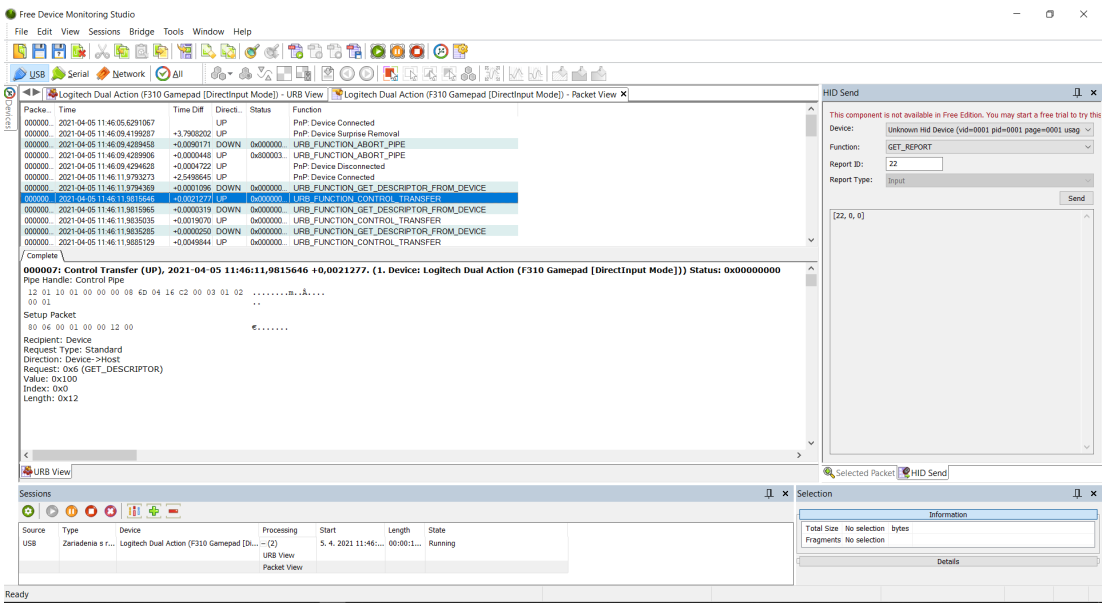


Obrázek 1.7: Ukážka vyobrazenia URB.

Pričom pri dvojkliku na šedé časti textu (napríklad *Setup Packet* alebo *Endpoint Address*) sa užívateľovi rozbalí okno s detailnejším popisom.

Zaujímavá funkcionalita ktorú ale program ponúka len v platenej verzii, je umožnenie užívateľovi priamo komunikovať so zvoleným zariadením. Môžeme mu tak posielať rôzne požiadavky (niektoré z nich sú spomenuté v USB dokumentácii v kapitole 9.4 (ODKAZ)) ako napríklad *GET_REPORT* kde špecifikujeme *Report ID* a prípadné ďalšie parametre, a zariadenie nám patrične odpovie.

Medzi zachytenými paketami sa prvotne nezobrazujú tie, ktoré označujú nakonfigurovanie daného zariadenia (je nutné ho odpojiť a znova napojiť počas monitorovania). Užívateľské rozhranie vyobrazené nižšie pomocou obrázku 1.8 ,pozostáva z pomerne veľa ikoniek a celkovo sa javí ako trochu neprehľadné. Pri prvotnej interakcii s programom 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 1.8: Užívateľské rozhranie Device Monitoring Studio

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 zlepšili prácu s aplikáciou jej užívateľovi. Zároveň ich dopĺňame o pokročilejšie funkcie z ktorých sú niektoré viac špecificky zamerané pre účely ktoré by sme chceli dosiahnuť v našej aplikácii.

Ako prvé by sme si mali zadefinovať platformu na ktorú budeme cieliť s našou aplikáciou:

P1 Cieľová platforma našej aplikácie by mala byť Windows.

Našu aplikáciu by sme chceli zamerať už na jednotlivú analýzu paketov a nie ich zachytávaniu. Z toho vyplývajú nasledujúce požiadavky:

P2 Mala by byť schopná analyzovať USB pakety zachytené do súboru v rozumnom formáte pomocou predom definovaného snifferu.

P3 Mala by byť schopná analýzy paketov v reálnom čase. To znamená, že bude podporovať čítanie súboru súvisle s tým ako do neho bude zapisovať iný software (za predpokladu, že to daný software povoľuje).

Vzhľadiska zamerania našej aplikácie na základnú podmnožinu USB deskriptorov si zadefinujeme nasledujúcu požiadavku:

P4 Mala by podporovať sémantickú analýzu (vyobrazenie pomocou stromovej štruktúry) pre všetky základné USB deskriptory spomenuté v USB 2.0 dokumentácii(ODKAZ kapitola 9.6) (ako napríklad *Device descriptor*, *Interface descriptor*, *Endpoint descriptor*,...).

Požiadavky na samotnú analýzu sú:

P5 Mala by vedieť rozumne zobraziť dáta ktoré daný sniffer zachytí a uloží. Pod pojmom rozumne myslíme spôsob zobrazenia pomocou hexdumpu.

P6 Mala by uľahčiť používateľovi orientáciu v hexdumpe. Jednotlivé znaky budú farebne označené na základe ich významu (hlavička paketu, rôzne typy deskriptorov, ...).

P7 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.

P8 Detailnejšie informácie o pakete budú zobrazované na základe interakcie užívateľa s aplikáciou.

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

Vzhľadom na zamerianie sa na užšiu podmnožinu HID zariadení doplníme nasledujúce požiadavky:

P10 Mala by byť schopná rozparsovať *HID Report Descriptor* takým štýlom, aby bolo neskôr možné sématnicky reprezentovať input určitých HID zariadení.

P11 Mala by byť schopná vhodným spôsobom vizuálne zobraziť sémantický význam dát posielaných danou podmnožinou HID zariadení, do ktorej patrí myš, klávesnica a joystick.

1.4 Ciele práce

Celkové ciele tejto práce sú následovné :

C1 Vytvoriť funkčný analyzátor ktorý spĺňa všetky požadované funkcie **??-P9**

C2 Návrh programu musí byť dostatočne obecný aby splňoval nasledujúce:

- Jednoduché rozšírenie o analýzu ďalších typov USB prenosov.
- Jednoduché pridanie sémantickej analýzy pre ďalšie HID zariadenia.

2. USB a Windows

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), low/full/high speed zariadenia

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

Opisat ako teda bezne analyzatory/sniffery funguju 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/en-us/windows-hardware/drivers/kernel/f drivers) a filter driver(https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/filte 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 hex-dumpe

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

1.1 Ukážka hexdumpu so zvýrazneným endpoint deskriptorom. 4
1.2 Ukážka reprezentácie dát pomocou stromovej štruktúry. 4
1.3 Endpoint deskriptor reprezentovaný dátami zvýraznenými na obrázku vyššie. 5
1.4 Ukážka vyobrazenia jednotlivých bytov. 5
1.5 Ukážka stromovej štruktúry na zvolenie si zariadenia s ktorým bude zachytávaná komunikácia. 6
1.6 Príklad analýzy paketov. 6
1.7 Ukážka vyobrazenia URB. 7
1.8 Užívateľské rozhranie Device Monitoring Studio 7

Seznam tabulek

Seznam použitých zkratek

Přílohy

.1 První příloha