VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ŘÍZENÍ PŘÍSTUPU K USB SBĚRNICI

ACCESS CONTROLL FOR USB BUS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN KRAJČI**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN OČENÁŠ,**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Krajči Martin**
Program: Informační technologie
Název: **Řízení přístupu k USB sběrnici**
Access Controll for USB Bus
Kategorie: Bezpečnost

Zadání:

1. Nastudujte princip fungování USB sběrnice.
2. Navrhněte způsob softwarového omezení přístupu určitých USB zařízení k počítači.
3. Implementujte nástroj realizující omezení přístupu navržené v bodě 2.
4. Ověřte funkčnost nástroje na vybraných počítačích a USB zařízeních.
5. Porovnejte vámi vytvořený nástroj s existujícími nástroji.
6. Dosažené řešení zhodnoťte a navrhněte další vylepšení.

Literatura:

- Jan Axelson: USB Complete: The Developer's Guide, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Očenáš Martin, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 31. října 2019

Abstrakt

Táto bakalárska práca sa zaoberá kontrolovaním USB zariadení v operačnom systéme Linux. Hlavným cieľom je vytvorenie aplikácie, ktorá by USB zariadenia analyzovala a na základe ich atribútov rozhodovala o tom, či môžu byť povolené alebo je nutné ich zablokovať, pričom kontrolu je potrebné vykonať ešte pred tým, než zariadenie dostane šancu páchať potencionálne škodlivú činnosť. Práca taktiež pojednáva o tom, prečo je z hľadiska bezpečnosti vhodné niektoré zariadenia nepovoliť. Výsledná aplikácia bola implementovaná vo forme dvoch nezávislých modulov. Prvý modul predstavuje rozhranie na prácu s pravidlami, podľa ktorých sa bude rozhodovať o bezpečnosti zariadenia. Druhý modul využíva vytvorené pravidlá na kontrolu USB zariadení, o ktorých sa dozvedá pomocou správ z jadra, určených pre podsystém UDEV. Na základe výsledku kontroly rozhoduje, či má byť zariadenie autorizované alebo neautorizované, o čom patrične oboznámi používateľa pomocou terminálu.

Abstract

This bachelor thesis addresses the matter of controlling USB devices in operating system Linux. The main objective consists of making an application, which analyzes USB devices based on their attributes and decides, whether they should be allowed or blocked, while it is necessary to finish the inspection before a device is given a chance to perform potentially harmful activities. The thesis also discuses the need of blocking certain devices in terms of security. The final application was implemented as two independent modules. The first module represents an interface for defining specific rules to be used in order to determine the harmlessness of the device. Second module uses created rules to control USB devices discovered by kernel messages designed for subsystem UDEV. Based on the result of the control, the application decides whether the device should be authorized or unauthorized and informs the user about the performed action via terminal.

Kľúčové slová

USB, C/C++, Linux, UDEV, bezpečnosť, blokovanie USB zariadení

Keywords

USB, C/C++, Linux, UDEV, security, USB device blocking

Citácia

KRAJČI, Martin. *Řízení přístupu k USB sběrnici.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Očenáš,

Řízení přístupu k USB sběrnici

Prehlásenie

Prehlasujem, že túto bakalársku prácu som vypracoval samostatne pod vedením pána Ing. Martina Očenáša. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Martin Krajči
4. júna 2020

Poďakovanie

Chcel by som poďakovať môjmu vedúcemu, pánovi Ing. Martinovi Očenášovi za jeho odborné rady a pripomienky počas celej doby vypracovávania bakalárskej práce.

Obsah

Kapitola 1

Úvod

Táto bakalárska práca sa zaoberá detekciou a rozpoznávaním USB zariadení v operačnom systéme linux. Keďže v dnešnej dobe osobné počítače vo väčšine prípadov neposkytujú všetku potrebnú funkcionalitu, je potrebné ju doplniť ďalšími zariadeniami, ako sú napríklad web kamery či herné ovládače. Na takéto pripojenie sa využíva najčastejšie protokol USB, ktorý sa stal veľmi populárnym.

Popularita tomuto protokolu priniesla aj mnoho problémov a niekoľko z nich sa radí do kategórie bezpečnosti. Cenová dostupnosť zariadení využívajúcich tento protokol spôsobila, že vznikli rôzne varianty útokov [11]. Zariadenie sa môže na prvý pohľad javiť ako bezpečné a vykonávajúce správnu funkciu, no zároveň v sebe ukrývať ďalšie skryté rozhrania schopné v systéme napáchať škodu. Typickým predstaviteľom tohto útoku je upravený USB pamäťový kľúč. Okrem toho, že poskytuje úložisko súborov, môže obsahovať naimplementované rozhranie, ktoré sa tvári ako klávesnica alebo eternertový adaptér. Útočník potom nepotrebuje nič viac než zapojiť škodlivé zariadenia do osobného počítača a chvíľku nepozornosti obete. Časť zariadenia, ktorá sa tvári ako klávesnica môže pomocou skratiek spustiť terminál, nakaziť operačný systém nakopírovaním škodlivého softvéru, vytvoriť a spustiť nebezpečný skript a podobne [7]. Eternetový adaptér zase dokáže presmerovať internetovú komunikáciu na útočníka, ktorý by mu bol schopný podvrhnúť falošné stránky, prípadne by útočníkovi poskytol priamy prístup do počítača.

Spôsobov, ako sa proti týmto útokom brániť nie je mnoho. Pomocou internetu sa dá dohľadať niekoľko zdĺhavých a ťažko pochopiteľných návodov, ako zablokovať všetky USB zariadenia, vrátane USB pamäťových kľúčov, prípadne len jedno konkrétne zariadenie. Takéto riešenie je pomerne nepraktické z dôvodu, že v prípade zmeny blokovaného zariadenia je potrebné všetky prvotné úpravy vrátiť do pôvodného stavu, a všetky kroky zopakovať pre nové zariadenie. Okrem iného, prístup, kedy sú potenciálne škodlivé zariadenia blokované a všetky ostatné povoľované, nie je správny. Vhodné je stanoviť, ktoré zariadenia sú povolené na akých USB portoch. Potenciálne riešenie tohto problému je nástroj s názvom USBGuard[1]. Problém je ten, že sa jedná o veľmi komplikovaný nástroj a človek, ktorý by chcel chrániť svoj osobný počítač pred nebezpečnými USB zariadeniami, ale nepohybuje sa v obore IT, by mohol mať vážny problém nástroj používať.

Preto si táto bakalárska práca kladie za cieľ vytvoriť nástroj, ktorý by bol schopný blokovať potenciálne škodlivé zariadenia a zariadenia, ktoré si zákazník v systéme neželá. Zároveň má byť jeho používanie čo najjednoduchšie a zrozumiteľné tak, aby s tým nemal problém ani neznalý používateľ.

[1]Dokumentácia k nástroju USBGuard sa nachádza na tomto odkaze https://usbguard.github.io/.

V kapitole 2 sa nachádza súhrn teórie, nevyhnutnej pre pochopenie návrhu a riešenia výslednej aplikácie. Prvá sekcia tejto kapitoly sa zaoberá protokolom USB, a to najmä zariadeniami, ktoré tento protokol využívajú. Hlavná časť je venovaná atribútom USB zariadení, ktoré tieto zariadenia popisujú. Druhá sekcia popisuje podsystém UDEV, so zameraním najmä na detekciu USB zariadení a popisom umiestnenia ich atribútov. Kapitola 3 pojednáva o existujúcich riešeniach, pomocou ktorých je možné obmedzovať prístup USB zariadení, pričom sú rozoberané aj ich výhody a nevýhody. V kapitole 4 sa nachádza súhrn vlastností, ktoré by mala výsledná aplikácia zahŕňať. Zároveň je tam možné nájsť popísaný návrh aplikácie, rozdelený na dve časti – návrh hlavného programu a programu na prácu s pravidlami. Kapitola 5 popisuje implementáciu, ktorá bola vyhotovené na základe návrhu. Okrem implementácie dvoch modulov sú zobrazené použité technológie vrátane odôvodnenia, prečo boli v práci použité. Overenie funkčnosti výslednej aplikácie je popísané v kapitole 6, pričom boli použité štyri testy, a každý z testov sa zaoberal inou vlastnosťou alebo funkciou. Na konci kapitoly sú taktiež zhrnuté možné vylepšenia, ktoré by do budúcnosti mohli byť do aplikácie doplnené a ako posledné je zobrazené porovnanie s existujúcimi riešeniami.

Kapitola 2

Teória USB zariadení a ich detekcie na systéme Linux

V druhej kapitole sa nachádza úplný prehľad teórie, potrebnej k pochopeniu návrhu a implementácie výslednej aplikácie. Keďže je nevyhnutné mať poznatky z dvoch, na prvý pohľad nesúvisiacich oblastí, je rozdelená na dve sekcie. Prvá sa venuje zariadeniam využívajúcim protokol USB a druhá popisuje podsystém UDEV.

2.1 Zariadenia využívajúce USB štandard

Táto sekcia popisuje základné informácie o USB štandarde, možnosti zapojenia a spôsob detekcie nových zariadení [1] [2]. Nasleduje štruktúra, pričom je braný ohľad najmä na detailný popis deskriptorov [13] [2]. Posledná podsekcia obsahuje informácie o zariadeniach, ktoré často využíva na ovládanie človek – HID [2].

2.1.1 Štandard USB

USB, v preklade univerzálna sériová zbernica, je štandard, ktorý definuje konfiguráciu zariadení, konektory a spôsob komunikácie medzi komunikujúcimi stranami. Využíva sa na prepojenie zariadení rôznych typov, no najčastejšie sa jedná o periférie rozširujúce funkcionalitu osobného počítača. Komunikácie sa zúčastňujú zariadenia a vždy práve jeden hostiteľ. Jediná požiadavka na dáta, ktoré sú pomocou protokolu USB prijímané a odosielané je ten, že musia byť v digitálnej forme a zabalené do paketov. Charakteristickým prvkom je využívanie systému správ, pri ktorom je akákoľvek výmena dát vyvolaná hostiteľom, čím sa zamedzí konfliktom na zbernici.

Využívanie tohoto štandardu prináša niekoľko výhod. K hostiteľovi je možné pripojiť až 127 zariadení, pričom tieto zariadenia môžu byť napájané priamo z USB konektoru, o čom rozhoduje výška vstupného napätia a potrebný príkon. Typy konektorov sú štandardizované a väčšinou je možné pomocou jedného káblu prepojiť veľké množstvo zariadení. Ďalšou veľkou výhodou je to, že periférie je možné pripojiť za behu osobného počítača a nie je pri tom nutné reštartovať operačný systém. V absolútnej väčšine prípadov nie je potrebná ani manuálna konfigurácia a zariadenia sú ihneď pripravené vykonávať požadovanú funkciu.

2.1.2 Topológia a možnosti zapojenia zariadení

Základom každého zapojenia je hostiteľ, ktorý sa skladá z radiča a koreňového rozbočovača[1]. Hostiteľský radič zabezpečuje odosielanie dát k zariadeniam a zároveň spracovávanie prijatých dát tak, aby im rozumel operačný systém. Koreňový rozbočovač zabezpečuje detekciu pripájania a odpájania zariadení a je stredom celého zapojenia, nazývaného hviezda[2]. Do tohto rozbočovača sa pripájajú zariadenia, medzi ktoré môžu patriť aj ďalšie rozbočovače. Tie môžeme zapájať za sebou, teda do série, pričom každý bude tvoriť zapojenie do hviezdy. Celkovo môžeme za seba zapojiť sedem úrovní rozbočovačov.

2.1.3 Detekcia nového zariadenia

Každý rozbočovač, vrátane toho koreňového, má koncový bod prerušení, ktorý zabezpečuje nahlasovanie novo pripojených zariadení hostiteľovi. Po zistení zmeny na rozbočovači hľadá hostiteľ presný port, na ktorom sa zmena nachádza, a vytvorí spojenie so zariadením na koncovom bode nula, popísanom tu 2.1.4. Aby mohlo byť zariadenie správne zaregistrované, musí odpovedať na prijímané požiadavky, odosielať žiadané informácie a vykonávať potrebné úkony. Po získaní potrebných dát sa vyberie vhodná konfigurácia a šírka pásma vyžadovaná pre správnu funkciu.

2.1.4 Štruktúra zariadení

Každé zariadenie môže vykonávať viacero funkcií, čomu odpovedá aj jeho štruktúra. Skladá sa vždy zo štyroch úrovní, pri čom na každej úrovni sa musí nachádzať aspoň jeden prvok[3]. Na samotnom vrchole sa nachádza úroveň s názvom zariadenie, ďalej je to konfigurácia, rozhranie a koncový bod. Každý prvok nachádzajúci sa v jednej zo spomínaných úrovní obsahuje deskriptor, popisovač, ktorý má významnú úlohu a poskytuje nám dôležité informácie, potrebné pre správne zaobchádzanie operačného systému so zariadením. Každé zariadenie ho musí obsahovať a byť schopné ho na požiadanie poskytnúť. Ukážka štruktúry usporiadania deskriptorov sa nachádza na obrázku 2.1.

Na ukážke sa nachádza aj HID deskriptor, vysvetlený v podsekcii 2.1.5.

Trieda a podtrieda

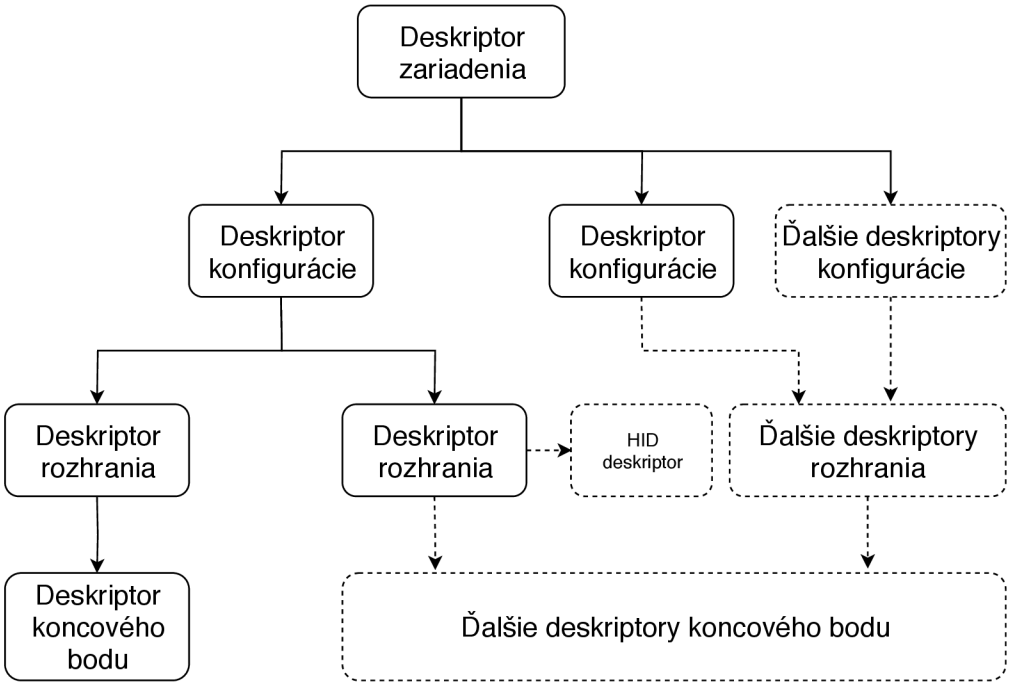
V oboch prípadoch sa jedná o dvojciferné hexadecimálne číslo, ktoré môžeme nájsť v deskriptoroch popísaných nižšie. Zatiaľ čo trieda definuje všeobecnú funkcionalitu, podtrieda ju upresňuje a zužuje tak skupinu zariadení, ktoré tomuto popisu zodpovedajú. Kombinácia týchto dvoch hodnôt definuje, akú konkrétnu funkciu zariadenie vykonáva, prípadne do akej kategórie patrí. Oba atribúty sa nachádzajú v deskriptore zariadenia aj rozhrania a je to z toho dôvodu, že zariadenie môže vykonávať viacero odlišných funkcií. V takom prípade by mala trieda aj podtrieda na úrovni zariadenia hodnotu `00` a oba atribúty by boli definované pre každé rozhranie zvlášť. Príklad takého prípadu sa môže skladať z dvoch rozhraní. Trieda a podtrieda by mali v deskriptore zariadenia hodnoty `00`, v jednom deskriptore rozhrania `03` a `01`, a v druhom `08` a `06`, čo by definovalo, že jedno zariadenie je schopné vykonávať funkciu HID[4] a zároveň USB pamäťového kľúča.

[1] Rozbočovač je zariadenie, slúžiace na zvýšenie počtu portov v systéme [3].

[2] Topológia USB zariadení vzhľadom na jeden rozbočovač predstavuje hviezdu, no zapojenie viacerých rozbočovačov spolu so zariadeniami vytvorí strom.

[3] V prípade prvej úrovne sa v absolútnej väčšine zariadení vyskytuje maximálne jeden prvok.

[4] Napríklad myš, klávesnica, herný ovládač a iné.



Obr. 2.1: Znázornenie štruktúry USB zariadenia

Deskriptor zariadenia

Prvý deskriptor, o ktorý žiada hostiteľ je na úrovni zariadenia. Je potrebný na získanie ďalších podstatných informácií ako napríklad počet konfigurácií. Okrem iného obsahuje aj ID výrobcu a produktu. Tieto identifikátory presne definujú typ zariadenia a môžu slúžiť na načítanie správneho ovládača, ak ho nie je možné načítať na základe triedy a podtriedy, ako je tomu napríklad v prípade zariadení so špecifickou, výrobcom definovanou funkciou.[5]. Identifikátor výrobcu je možné získať po podaní žiadosti a zaplatení administratívneho poplatku. Pre túto prácu je taktiež okrem spomínaných atribútov dôležitá aj trieda a podtrieda.

Deskriptor konfigurácie

Aktívna môže byť vždy len jedna konfigurácia, no celkovo môže zariadenie disponovať viacerými. Tento deskriptor popisuje okrem iného počet rozhraní, maximálny odoberaný prúd, alebo možnosti napájania. Niektoré zariadenia môžu oplývať vlastným zdrojom energie a zároveň byť schopné fungovať na napájaní z USB portu. V takom prípade by v zariadení vznikli dve konfigurácie a hostiteľ by si mohol vybrať tú, ktorá by mu v aktuálnej situácii vyhovovala viac. Pokiaľ zariadenie nemá vlastné napájanie a hostiteľ potrebný prúd nemôže dodať, môže zariadenie odmietnuť.

Deskriptor rozhrania

Pod každou konfiguráciou môže byť niekoľko rozhraní, pričom viac ako jedno môže byť aktívne. V tomto deskriptore sa nachádza taktiež viacero atribútov, no pre túto bakalársku

[5]List identifikátorov výrobcov a produktov sa nachádza na http://www.linux-usb.org/usb.ids. Vzhľadom na to, že sa nejedná o oficiálnu databázu, obsah nemusí byť v danom momente aktuálny.

prácu sú opäť významné trieda a podtrieda. Okrem toho tu môžeme nájsť napríklad aj protokol zariadenia. Všetky tri spomínané atribúty sú priraďované spoločnosťou USB Org a môžu slúžiť na priradenie správneho ovládača zariadeniu.

Deskriptor koncového bodu

Obsahuje informácie ako napríklad adresu alebo maximálnu veľkosť paketu. Výnimkou je takzvaný *koncový bod nula*, ktorý deskriptor neobsahuje. Zariadenie ho musí mať vždy a slúži na prvotnú identifikáciu zariadenia a získanie deskriptorov.

2.1.5 HID - Human Interface Device

HID, v preklade zariadenie s ľudským rozhraním, je štandard používaný nie len v rámci USB. Tak, ako vyplýva z názvu, je určený pre zariadenia, ktoré pre človeka predstavujú spôsob, ako ovládať systém, s ktorým chce pracovať. Keďže štandard HID predstavuje množstvo výhod, nemusí tento popis zariadenia sedieť vždy a môžeme ho tak nájsť aj v netradičných USB zariadeniach, spomenutých nižšie. Dôležité je, aby správy, ktoré odosielajú, spĺňali formu zadefinovanú v HID deskriptore. Najčastejšie sa jedná o bežne používané zariadenia, ako napríklad myš, klávesnica alebo herné zariadenie, ale môžeme sa stretnúť aj s batériovými systémami alebo simulátormi. Vďaka štandardizácii typov zariadení a dát, ktoré môžu odosielať, nie je potrebný ovládač pre každé zariadenie zvlášť, čo podstatným spôsobom uľahčuje pripájanie nových zariadení.

HID deskriptor správ

Hostiteľovi poskytuje informácie, ktoré potrebuje vedieť, aby dokázal komunikovať s HID zariadením. Ide najmä o správny formát správ a priraďovanie významu prijatých dát. Hostiteľ získa HID deskriptor pri žiadosti o konfiguráciu, pod ktorou sa nachádza rozhranie s triedou a podtriedou definujúcou, že sa jedná o HID. Konkrétne by trieda obsahovala hodnotu `03` a podtrieda `01` alebo `00`. Prvý prípad sa používa u zariadení, ktoré sú schopné fungovať ešte pred načítaním operačného systému. Môže sa jednať o klávesnicu alebo myš s podporou zjednodušeného protokolu, ktorému BIOS rozumie aj bez načítania deskriptoru správ. Táto funkcia sa hodí napríklad v menu nastavení, ktoré je dostupné pri zavádzaní operačného systému.

Príklad deskriptoru správ

Ukážka 2.2 predstavuje časť reálneho deskriptoru zariadenia, použitého v jednom z dvoch rozhraní USB HID myši. Jedná sa o výpis hexidecimálnych čísel s komentárom, ktorý bol vytvorený pomocou online nástroja[6]. Každá z týchto hodnôt má špeciálny význam a spolu vytvárajú definíciu toho, aké správy bude zariadenie odosielať a ako ich má príjemca interpretovať.

Začiatok deskriptora tvorí definícia stránky použitia (Usage Page), v rámci ktorej sa udáva konkrétne použitie (Usage). Kolekcie (Collection) slúžia na shlukovanie dát patriacich do rovnakej kategórie. Každý deskriptor musí obsahovať kolekciu aplikácie (Collection (Application)), ktorej prvky vytvárajú jednu funkciu. Logické minimum a maximum (Logical Minimum, Logical Maximum) definujú hodnoty, ktoré môže zariadenie v správe odoslať.

[6]Online nástroj pre popisovanie deskriptorov `https://eleccelerator.com/usbdescreqparser/`.

```
0x05, 0x01, // Usage Page (Generic Desktop Ctrls)
0x09, 0x02, // Usage (Mouse)
0xA1, 0x01, // Collection (Application)
0x09, 0x01, //  Usage (Pointer)
0xA1, 0x00, //  Collection (Physical)
0x05, 0x09, //   Usage Page (Button)
0x19, 0x01, //   Usage Minimum (0x01)
0x29, 0x10, //   Usage Maximum (0x10)
0x15, 0x00, //   Logical Minimum (0)
0x25, 0x01, //   Logical Maximum (1)
0x75, 0x01, //   Report Size (1)
0x95, 0x10, //   Report Count (16)
0x81, 0x02, //   Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null Position)
0x05, 0x01, //   Usage Page (Generic Desktop Ctrls)
0x09, 0x30, //   Usage (X)
0x09, 0x31, //   Usage (Y)
0x16, 0x01, 0x80, //   Logical Minimum (-32767)
0x26, 0xFF, 0x7F, //   Logical Maximum (32767)
0x75, 0x10, //   Report Size (16)
0x95, 0x02, //   Report Count (2)
0x81, 0x06, //   Input (Data,Var,Rel,No Wrap,Linear,Preferred State,No Null Position)
⋮
0xC0, //  End Collection
⋮
0xC0, // End Collection
```

Obr. 2.2: Príklad USB HID deskriptoru

Kombináciou stránky použitia s použitím dostaneme konkrétne funkcie, ktoré môže zariadenie vykonávať. Okrem toho obsahuje počet správ (Report Count), ktorý udáva množstvo údajov, ktoré môžeme vyčítať z jednej správy a veľkosť správy (Report Size).

2.2 UDEV

Táto časť popisuje spôsob pridávania a obsluhy novo pripojených zariadení pomocou podsystému UDEV. V prvej časti sa nachádzajú základné informácie o jeho chovaní [5][10]. Nasleduje krátky popis soketov, a to konkrétne z rodiny Netlink [6][8]. Tie môžu prijímať správy, ktoré sa nazývajú Uevent a sú podrobnejšie popísané v ďalšej podsekcii [5]. Posledná podsekcia obsahuje popis súborového systému `sysfs` z pohľadu podstatného pre túto prácu [12][9].

2.2.1 Základné informácie a chovanie

V systéme Linux sa nachádza počínajúc verziou jadra 2.5, kedy vznikol z dôvodu náhrady pôvodného nepraktického riešenia. Pred verziou 2.5 sa v systéme nachádzal skript s názvom MAKEDEV, ktorý všetky súbory teoreticky pripojiteľných zariadení nahral do priečinka

`/dev`. Toto riešenie nebolo optimálne, a preto vznikol UDEV, ktorý mal vyššie popísaný problém odstrániť.

Nové riešenie je navrhnuté tak, aby súbory zariadení, vďaka ktorým môžu aplikácie pristupovať k hardvéru, nahrávalo do priečinka `/dev` dynamicky, a to len vtedy, keď sú reálne pripojené do systému. Túto činnosť vykonáva v používateľskom priestore na základe správ nazývaných `uevent`, ktoré prijíma od jadra systému pomocou Netlink soketov. Uevent aj Netlink sokety sú vysvetlené v nasledujúcich dvoch podsekciách.

2.2.2 Uevent

Po pripojení nového zariadenia do systému ho obslúži samotné jadro. Získa od neho potrebné deskriptory a informácie z nich rozloží na jednotlivé položky, nazývajúce sa atribúty, ktoré sú uložené v súboroch. Jednotlivé súbory môžu obsahovať položky s hodnotami popisovanými v podsekcii 2.1.4. Cesta k týmto súborom začína v priečinku `/sys` a zvyšok je predmetom už spomínanej `uevent` správy.

```
KERNEL[50947.544356] add /devices/pci0000:00/0000:00:14.0/usb1/1-1 (usb)
KERNEL[50947.546032] add /devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1:1.0 (usb)
KERNEL[50947.547793] add
/devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1:1.0/0003:09DA:054F.0003 (hid)
```

Obr. 2.3: Príklad `uevent` správ získaných pomocou nástroja `udevadm monitor`

Na príklade 2.3 môžeme vidieť časť výstupu z nástroja `udevadm monitor` po zapojení USB HID myši do osobného počítača. Na začiatku správy figuruje odosielateľ správy, teda jadro (KERNEL). Druhá časť správy hovorí podsystému UDEV, čo sa má so zariadením stať. V tomto konkrétnom prípade sa má pridať (add). Posledná časť správy je cesta v rámci priečinka `/sys`. Prvá správa hovorí o pridaní zariadenia a druhá o pridaní rozhrania v rámci zariadenia definovaného v prvej správe. Keďže bolo v rozhraní uvedené, že sa jedná o HID zariadenie, tretia správa obsahuje cestu k priečinku, ktorý obsahuje okrem iného aj HID deskriptor.

Pri použití príkazu `udevadm monitor --environment` môžeme vidieť ďalšie informácie o správe, prípadne atribúty z priečinka uvedeného v správe. Príklad výstupu tohto príkazu môžeme vidieť na 2.4.

```
KERNEL[53380.564193] add /devices/pci0000:00/0000:00:14.0/usb1/1-1 (usb)
ACTION=add
DEVPATH=/devices/pci0000:00/0000:00:14.0/usb1/1-1
SUBSYSTEM=usb
DEVNAME=/dev/bus/usb/001/007
DEVTYPE=usb_device
PRODUCT=9da/54f/274
TYPE=0/0/0
BUSNUM=001
DEVNUM=007
SEQNUM=4352
MAJOR=189
MINOR=6
```

Obr. 2.4: Detailný príklad `uevent` správy získanej pomocou nástroja `udevadm monitor`

`Uevent` správy sú prijímané aj odosielané pomocou Netlink soketov a majú mierne odlišnú podobu od podoby prezentovanej nástrojom `udevadm monitor`. Bez spracovania by správy z prvého príkladu vyzerali tak, ako ich je možné vidieť na ukážke 2.5.

```
add@/devices/pci0000:00/0000:00:14.0/usb1/1-1
add@/devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1:1.0
add@/devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1:1.0/0003:09DA:054F.0003
```

Obr. 2.5: Príklad `uevent` správ získaných pomocou Netlink soketu

2.2.3 Netlink

V operačnom systéme Linux je bežné, že v jadre bežia len kritické aplikácie, najčastejšie s plným prístupom k hardvéru. Tie, ktoré nepotrebujú žiadne špeciálne požiadavky, ako napríklad webový prehliadač, sú umiestnené v používateľskej vrstve. Funkcionalitu týchto dvoch vrstiev je často potrebné spojiť, čo sa nezaobíde bez ich vzájomnej komunikácie. Z toho dôvodu vznikla doména soketov Netlink.

Netlink soket umožňuje medziprocesovú obojsmernú komunikáciu medzi jadernými modulmi a aplikáciami na používateľskej vrstve. Podporuje multicast, čo znamená, že jeden proces môže odoslať správu pre celú skupinu Netlink adries, vďaka čomu predstavuje vhodné prostredie pre jadro na oznamovanie udalostí. Je možné definovať až 32 multicastových skupín, čo je užitočné z dôvodu, že tento systém komunikácie môže využívať viacero skupín aplikácií, bez toho aby sa navzájom ovplyvňovali.

Daný typ komunikácie je taktiež vhodný pre túto prácu, pretože je potrebné, aby správy, ktoré jadro odosiela boli doručené podsystému UDEV a zároveň aplikácii, ktorá má slúžiť ako ochrana proti nebezpečným zariadeniam.

2.2.4 Sysfs

Je to súborový systém založený na `ramfs`, ktorý sprostredkúva spôsob poskytovania dát z jadra pre užívateľskú vrstvu. Dáta predstavujú štruktúry a ich atribúty, ktoré sa po exportovaní uložia v podobe súboru, a sú zobraziteľné pre používateľa tak, ako tomu je aj v prípade USB zariadení.

Súborový systém `sysfs` sa v absolútnej väčšine prípadov nachádza v priečinku `/sys`, v ktorom je možné nájsť ďalšiu hierarchiu priečinkov. Priamo v `/sys` je možné nájsť niekoľko podpriečinkov, pričom má každý svoj špecifický význam. S ohľadom na USB zariadenia sú zaujímavé najmä `/sys/devices` a `/sys/bus`. V oboch by sme mohli nájsť priečinky zodpovedajúce USB zariadeniam a je to možné z toho dôvodu, že spadajú do dvoch rôznych kategórií. Do `/sys/devices` spadajú, pretože je tento podpriečinok určený na uloženie stromu zariadení v jadre, a do `/sys/bus` preto, lebo sa jedná o zariadenia, komunikujúce cez zbernicu pomocou USB protokolu, a tento podpriečinok slúži na ukladanie dát súvisiacich so zbernicami. Z dôvodu šetrenia pamäte sa reálne nenachádzajú v oboch priečinkoch, no v jednom z nich sú umiestnené len symbolické odkazy.

Hierarchické rozloženie priečinkov sa týka aj USB zariadení, kde každé zariadenie predstavuje priečinok, v ktorom sa nachádzajú atribúty zariadenia. Okrem toho sa v ňom nachádza jeden alebo viac priečinkov, ktoré zodpovedajú rozhraniam, a taktiež obsahujú zodpovedajúce atribúty. Podobne to platí aj pre koncové body, pričom hierarchia priečinkov zodpovedá hierarchii deskriptorov.

Kapitola 3

Existujúce spôsoby blokovania USB zariadení

Existujúcich riešení, ako efektívne blokovať USB zaradenia nie je mnoho. Veľa z nich je nepoužiteľných z viacerých dôvodov. Niektoré návody popisujú blokovanie všetkých USB zariadení, prípadne len USB pamäťových kľúčov a diskov. Toto riešenie je veľmi striktné a často reálne nepoužiteľné. Okrem toho, že tieto návody vo väčšine prípadov vyžadujú zadanie niekoľkých príkazov do terminálu a predpokladajú pokročilejšie znalosti s operačným systémom Linux, môžu používateľovi spôsobiť vážne problémy. Vymazanie alebo presun súboru dôležitého pre beh systému môže viesť k nepredpokladanému a často nechcenému chovaniu. Ďalším problémom je, že používateľ sa časom môže rozhodnúť, že USB zariadenia už blokovať nechce, a teda je nútený vrátiť systém pomocou terminálu do pôvodného stavu. Existujú aj riešenia poskytujúce omnoho väčšiu flexibilitu, no pre neskúseného sú značne náročné a svojou zložitosťou môžu odradiť svojich potenciálnych používateľov.

3.1 Práca s terminálom

Pri vyhľadávaní možností, ako blokovať USB zariadenia na systéme Linux, sa v absolútnej väčšine prípadov zobrazia odkazy na fóra a články, ktoré popisujú niekoľko príkazov. Tieto príkazy je potrebné zadať do terminálu a po ich vykonaní má byť určitá skupina USB zariadení zablokovaná. Najčastejšie sa jedná o vpísanie textového reťazca „blacklist usb-storage“, do súboru `/etc/modprobe.d/blacklist.conf`, čo spôsobí nenačítanie ovládača pre USB pamäťové zariadenia pri ďalšom spustení systému, a teda aj jeho nefunkčnosť. Okrem spomínaných problémov je pri tejto variante nevhodné to, že akýkoľvek privilegovaný používateľ môže ovládače načítať manuálne spustením jedného príkazu, a tým obísť celé opatrenie na blokovanie. To rieši ďalší spôsob blokovania, pri ktorom je nutné vymazať alebo presunúť ovládače vybraných zariadení z priečinka, kde operačný systém očakáva, že sa budú nachádzať [14]. To ale spôsobuje ďalší problém, pretože ak by používateľ po vymazaní ovládača chcel tieto USB zariadenia opäť povoliť, musel by odinštalovať a znova nainštalovať celý operačný systém. V prípade, že by súbor len presunul na súkromné úložisko, vždy sa môže stať, že o súbor príde, čo by viedlo k rovnakým následkom.

3.2 UDEV pravidlá

Jednou z efektívnych možností, ako blokovať potenciálne škodlivé USB zariadenia sú UDEV pravidlá, úzko súvisiace s podsystémom UDEV, popísaným v kapitole 2.2. Pri prijatí `uevent` správy získa UDEV prístup do priečinka, kde sa nachádzajú atribúty zariadenia, ktoré bolo pripojené do systému a treba ho obslúžiť. To, akým spôsobom má byť obslúžené definujú práve UDEV pravidlá. Nejedná sa teda iba o spôsob, ktorý môže používateľ využívať na blokovanie USB zariadení, ale o zaužívaný systém, ktorý môže podľa potreby doplniť tak, aby spĺňal jeho požiadavky.

Pravidlá sa môžu nachádzať v dvoch rôznych priečinkoch, pričom priečinok `/usr/lib/udev/rules.d/` obsahuje pravidlá pridané systémom. Po vytvorení vlastného pravidla je nutné ho uložiť do priečinka `/etc/udev/rules.d/`. Pravidlá sú súbory, ktorých názov sa začína číslom a končí príponou „.rules“, pričom číslo určuje poradie, v akom sa budú pravidlá kontrolovať. Okrem tvorby vlastných pravidiel, je možné meniť už existujúce, no za týmto účelom je potrebné uložiť ho do priečinka s používateľsky definovanými pravidlami pod rovnakým menom. Editácia súborov v druhom priečinku by nezanechala perzistentné zmeny a po reštarte systému by boli súbory opäť nedotknuté. Obsah súborov sa dá rozdeliť na základné časti. V prvej je nutné definovať, či sa má pravidlo aplikovať pri pridaní, odstránení alebo zmene zariadenia. V druhej časti je potrebné zadať atribúty, pomocou ktorých je možné vyfiltrovať zariadenia, pre ktoré pravidlo nie je určené. V poslednej časti sa nachádza definícia toho, čo sa má vykonať, ak pripojené zariadenie pravidlu vyhovuje. Môže sa jednať o nastavenie systémovej premennej, spustenie vlastného skriptu, alebo načítanie ovládača [4]. Posledná možnosť je častá a môžeme ju vidieť aj na ukážke 3.1.

```
ACTION!="add"
# Load he thunderbolt-net driver if we a device of type thunderbolt_xdomain
# is added.
SUBSYSTEM=="thunderbolt ", ENVDEVTYPE=="thunderbolt_xdomain",
RUNbuiltin+="kmod load thunderbolt-net"
```

Obr. 3.1: Ukážka časti pravidla zo súboru „90-nm-thunderbolt.rules“

Jedná sa o zjednodušenú časť pravidla zo súboru s názvom „90-nm-thunderbolt.rules“ v priečinku `/etc/udev/rules.d/`. Toto pravidlo určuje, že pokiaľ bolo do systému zapojené zariadenie z podsystému „thunderbolt“ a typ zariadenia je zhodný s „thunderbolt_xdomain“, má UDEV spustiť zavádzanie jaderného modulu (ovládača) „thunderbolt-net“.

Okrem spomínaných úkonov je možné využiť UDEV pravidlá aj na blokovanie USB zariadení. Na zistenie atribútov, ktoré môžeme v UDEV pravidlách použiť nám opäť poslúži aplikácia `udevadm`, konkrétne `udevadm info`. Pre ilustráciu môžeme opäť zvoliť USB HID myš a použiť cestu k jej atribútom získanú z `ueventu`. Výsledok získaný spustením aplikácie je zobrazený na ukážke 3.2.

Na základe týchto údajov je možné vytvoriť pravidlo blokujúce presne určený typ zariadenia. Príklad takého pravidla je možné vidieť na ukážke 3.3.
V prvom prípade je pri detekcii vyššie definovanej USB myši spustený skript, ktorý má zariadenie obslúžiť, prípadne ho zablokovať. Na to, aby takúto funkciu mohol spĺňať, je nevyhnutné mať hlbšie poznatky z oblasti fungovania USB na systéme Linux. V druhom prípade je priamo implementovaný spôsob, ako spomínané USB zariadenie zablokovať.

```
udevadm info -a /sys/devices/pci0000:00/0000:00:14.0/usb1/1-1

looking at device '/devices/pci0000:00/0000:00:14.0/usb1/1-1':
KERNEL=="1-1"
SUBSYSTEM=="usb"
DRIVER=="usb"
⋮
ATTR{idProduct}=="054f"
ATTR{idVendor}=="09da"
```

Obr. 3.2: Časť výpisu aplikácie `udevadm info` pre USB HID myš

```
ACTION=="add", SUBSYSTEM=="usb", ATTRidVendor=="054f",
ATTRidProduct=="09da", RUN="~/block-usb.sh"
```

prípadne

```
ACTION=="add", SUBSYSTEM=="usb", ATTRidVendor=="054f",
ATTRidProduct=="09da", RUN="/bin/sh -c 'echo 0 >/sys/$devpath/authorized'"
```

Obr. 3.3: Príklad pravidla blokujúceho konkrétny typ USB zariadenia

V oboch prípadoch sa síce jedná o efektívnu a flexibilnú metódu, ako USB zariadenia blokovať, no nie je možné ju využívať bez poznatkov z viacerých oblastí, a to najmä USB na systéme Linux a tvorba UDEV pravidiel. Využitie tejto možnosti neskúsenými používateľmi je teda nepravdepodobné.

3.3 USBGuard

Jedná sa o softvér, slúžiaci na zvýšenie bezpečnosti na systéme Linux. Jeho primárne využitie je blokovanie novo pripojených USB zariadení, ktoré by mohli byť potenciálne škodlivé, prípadne si ich používateľ v systéme nežiada. Okrem toho poskytuje ďalšie funkcie, súvisiace s USB zariadeniami, ako napríklad zobrazenie všetkých rozpoznaných zariadení alebo príkazy na prácu s pravidlami.

Na týchto pravidlách je funkcia programu založená a pred spusteným aplikácie je potrebné vytvoriť minimálne jedno. Pri tvorbe môžeme využiť viacero atribútov, ako napríklad ID portu, triedu a podtriedu rozhrania alebo meno zariadenia. Taktiež je nutné definovať, čo sa má so zariadením stať v prípade, že jeho atribúty sa budú zhodovať s jednotlivými položkami pravidla. Je možné použiť povolenie, blokovanie alebo odmietnutie zariadenia. Pravidlo pre USBGuard, obdobné UDEV pravidlu na blokovanie USB HID myši z predošlej sekcie, je zobrazené na ukážke 3.4

```
block 054f:09da
```

Obr. 3.4: Pravidlo vytvorené pre aplikáciu USBGuard

Okrem toho je v pravidlách možné použiť podmienky. Pravidlo môže byť aplikované pokiaľ boli všetky, jedna alebo žiadna podmienka vyhodnotená ako pravdivá. V tele podmienky je možné použiť napríklad časový rozsah, kedy má byť pravidlo použité. Taktiež sa môžeme

rozhodovať na základe toho, či pravidlo už bolo aplikované, kontrolované, prípadne podľa náhodne vygenerovanej hodnoty.

Na uľahčenie tvorby pravidiel je možné použiť takzvanú počiatočnú politiku. Pri budovaní počiatočnej politiky sú vytvárané pravidlá na základe zariadení, ktoré sú práve pripojené v systéme. Používateľ ich môže ďalej upravovať a dopĺňať podľa potreby.

Aplikácia USBGuard síce poskytuje viacero funkcií a možností ako ovplyvňovať overovanie USB zariadení, no sú pomerne zložité a návod, ako softvér používať, je pre neskúseného používateľa nedostačujúci. Okrem toho, výber samotných atribútov, ktoré tvoria jadro pravidla je oproti UDEV pravidlám malý.

3.4 Zhrnutie

Keďže je blokovanie USB zariadení pomocou terminálu nešikovné a často nemysliteľné, jediné použiteľné riešenia predstavujú UDEV pravidlá a aplikácia USBGuard. Prvý prípad prináša používateľovi veľkú flexibilitu. Okrem toho, že pri tvorbe pravidiel môže použiť všetky atribúty, ktorými zariadenia oplývajú, výber úkonu, ktorý sa má pri aplikovaní pravidla vykonať nie je nijak obmedzený. Na druhú stranu, na to, aby bol používateľ schopný tvoriť pokročilejšie pravidlá a zabezpečiť tak systém proti škodlivým USB zariadeniam bez toho, aby obmedzil neškodné zariadenia, potrebuje znalosti z oblasti tvorenia UDEV pravidiel a spôsobu fungovania USB zariadení na systéme Linux. Používanie aplikácie USBGuard je o niečo jednoduchšie, no návod určený na tvorbu pravidiel neposkytuje všetky potrebné informácie. Okrem toho, niektorým používateľom môžu chýbať atribúty, ako napríklad trieda a podtrieda zariadenia alebo počet rozhraní.

Kapitola 4

Návrh aplikácie

Nasledujúca kapitola sa zameriava popisom vlastností, ktoré má výsledná aplikácia spĺňať. Taktiež priblíži výslednú funkcionalitu, pričom sa bude odvolávať na teoretické poznatky spomenuté v kapitole 2.

4.1 Potrebné vlastnosti

Dôležitým krokom pri návrhu a tvorbe úspešnej aplikácie je určenie hlavných vlastností, ktoré má spĺňať. Jedná sa najmä o charakteristické črty výsledného produktu, ktoré priamo nevyplývajú z funkcionality a sú na tvorcovi. Viaceré nasledujúce body popisujú vlastnosti, vychádzajúce z nedostatkov aktuálnych možných riešení blokovania USB zariadení, popísaných v kapitole 3.

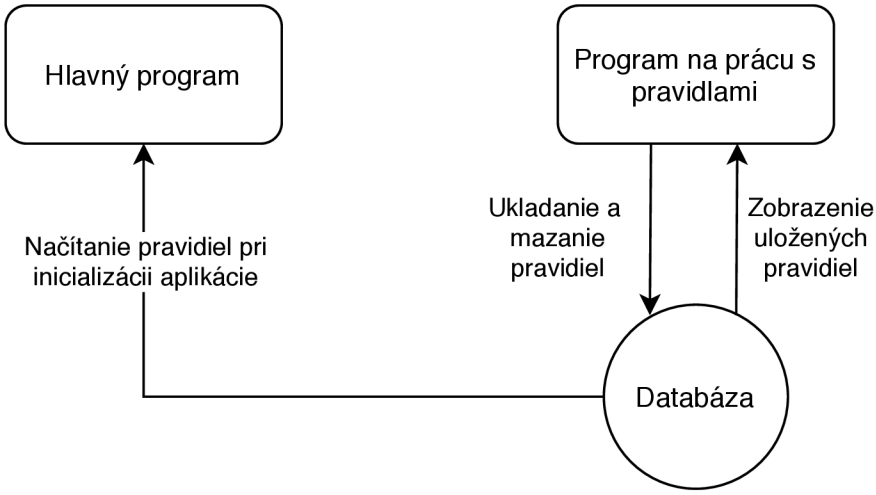
- Poskytnúť užívateľovi čo najjednoduchšie rozhranie na ovládanie blokovania.
- Blokovať USB zariadenia na základe pravidiel.
- Celé ovládanie aplikácie a narábanie s pravidlami sa bude vykonávať pomocou terminálu.
- K aplikácii bude poskytnutý kvalitný manuál uvádzajúci všetky nevyhnutné znalosti.
- Množstvo atribútov, z ktorých sa budú môcť tvoriť pravidlá bude také, aby nemiatlo používateľa, no zároveň poskytovalo dostatočnú flexibilitu.
- Nepovoliť funkciu zariadenia, kým nedôjde k jeho kontrole a tým maximalizovať bezpečnosť.

4.2 Základná charakteristika a architektúra aplikácie

Keďže je nemysliteľné aby aplikácia blokovala nechcené zariadenia na základe umelej inteligencie, pretože o väčšine zariadení nedokážeme s istotou povedať, že sú nebezpečné, je potrebné dodať používateľovi spôsob, ako definovať, ktoré zariadenia si v systéme neželá. Vhodným spôsobom na riešenie tohto problému je použitie systému pravidiel. Na vytváranie, zobrazovanie a mazanie pravidiel je potrebné vytvoriť používateľsky prívetivé prostredie. Pretože pri práci s pravidlami nie je dôvod na dlhodobý beh aplikácie, môže sa po patričnom úkone ukončiť.

Pre porovnanie, druhá časť aplikácie musí tieto pravidlá prečítať z pevného disku a uložiť ich do operačnej pamäte, kde budú uložené po dobu behu aplikácie. Po pripojení nového USB zariadenia do systému je potrebné na túto zmenu reagovať tak, aby zariadenie bolo overené čo najskôr a používateľ nepostrehol oneskorené započatie vykonávania jeho funkcie. Predpokladá sa, že aplikácia pobeží ako služba po celú dobu behu systému a bude kontrolovať všetky pripojené zariadenia. Preto je nevyhnutné, aby sa aplikácia neukončila sama, s výnimkou toho, že došlo k závažnej chybe a nie je možné v kontrole pokračovať.

Z toho dôvodu je vhodné rozdeliť aplikáciu na dva nezávisle funkčné moduly. Jeden modul bude spravovať pravidlá, zatiaľ čo druhý modul ich prečíta a použije na kontrolu, či je kontrolované zariadenia povolené.



Obr. 4.1: Základná štrukúra aplikácie

Na obrázku 4.1 môžeme vidieť oba vyššie spomínané moduly aplikácie, aj s naznačením toku dát. Tento tok dát prebieha medzi oboma modulmi a úložiskom na pevnom disku, konkrétne databázou.

Umiestnenie aplikácie

Väčšina aplikácií potrebuje na svoj beh určitú časť operačnej pamäte, do ktorej môže ukladať dáta potrebné k ich behu. V závislosti na tom, akú činnosť vykonávajú, môžu byť tieto dáta citlivé a poškoditeľné škodlivým softvérom, čo by mohlo spôsobiť pád celého systému. Preto býva operačná pamäť v systéme Linux rozdelená na dve časti. Jedna z nich je chránená časť, s ktorou môžu pracovať aplikácie so špeciálnym prístupom, prípadne aj aplikácie bez tohoto prístupu, no nepriamo a výhradne cez systémové volania. Tento priestor v pamäti sa nazýva priestor jadra a patrí sem napríklad jadro systému alebo ovládače hardvérových zariadení. Okrem toho, tieto aplikácie, ktoré bežia nad operačnou pamäťou z priestoru jadra, majú povolený prístup k určitej bežne neprístupnej časti systému, ako napríklad priama interakcia s hardvérom. Druhá časť sa nazýva používateľský priestor a patria sem všetky ostatné aplikácie, ako napríklad prehliadač alebo textový editor.

Keďže jeden z dvoch modulov aplikácie slúži ako rozhranie na prácu s databázou pravidiel, nevyžaduje žiadne špeciálne požiadavky a teda spadá do užívateľského priestoru. Druhý modul, teda hlavná časť aplikácie, úzko súvisí s priestorom jadra. Po pripojení USB zariadenia do systému, túto zmenu detekuje ako prvé jadro a vďaka Netlink soke-

tom o tom oboznámi aj aplikácie bežiace v používateľskom priestore. Z toho dôvodu ani jeden z modulov aplikácie nevyžaduje žiadne špeciálne požiadavky a je vhodné ich umiestniť do používateľského priestoru.

4.3 Program na prácu s pravidlami

Keďže pravidlá sú pre túto aplikáciu dôležitou súčasťou, ktorej sa používateľ nedokáže vyhnúť, pretože pre chod programu je nutné vytvoriť minimálne jedno pravidlo, je potrebné navrhnúť a implementovať užívateľsky prívetivé a intuitívne prostredie. Z hľadiska zachovania jednoduchosti je interakcia s pravidlami obmedzená len na terminál. Pre vykonanie jednotlivých úkonov bude nutné aplikáciu spustiť s parametrami a ich prípadnými atribútmi, pričom tento parameter bude presne definovať čo sa má vykonať.

Tvorba pravidiel

Pretože aplikácia potrebuje na svoj beh údaje, na základe ktorých bude zariadenia povoľovať, je pridávanie nových pravidiel nevyhnutný a najdôležitejší úkon, ktorý sa dá nad databázou pravidiel vykonať. Pravidlá sa budú skladať z informácií, ktoré popisujú funkcionalitu a vlastnosti USB zariadení. Tieto informácie, nazývajúce sa atribúty, je možné získať z deskriptorov zariadenia, popísaných v podsekcii 2.1.4. Teoreticky použiteľné atribúty sú na viacerých úrovniach a je ich niekoľko, no niektoré sa nemusia vyskytovať vždy a mnoho z nich poskytuje nízku mieru popisu zariadenia, ako napríklad veľkosť odosielaného paketu, ktorá je potrebná na správnu komunikáciu medzi jadrom a zariadením[1]. Preto je potrebné pre udržanie prehľadnosti programu vybrať len tie atribúty, ktoré zariadenie čo najlepšie definujú, a prvotný výber prípadne meniť na základe spätnej väzby od používateľov.

Využiteľné atribúty na tvorbu pravidiel

Prvotná skupina sa skladá z triedy a podtriedy zariadenia i rozhrania, ID výrobcu a produktu, celkového počtu rozhraní, ktoré obsahuje práve používaná konfigurácia a čísla portu v systéme, v ktorom sú zariadenia zapojené.

Trieda a podtrieda určujú, aký typ funkcie bude zariadenie vykonávať, teda pomocou páru týchto informácií je možné napísať pravidlo, ktoré dokáže povoliť napríklad USB huby, tlačiarne, HID zariadenia, zariadenia slúžiace ako úložisko a podobne, čo predstavuje jasný a zrozumiteľný spôsob, ako povoliť určitú skupinu zariadení s rovnakou alebo podobnou funkcionalitou, a teda aj vhodné atribúty pre aplikáciu.

Číslo výrobcu a číslo produktu presne definujú určitý typ výrobku, ktorý využíva protokol USB. To znamená, že pomocou tejto dvojice atribútov môžeme povoliť napríklad USB myš a zároveň všetky USB myši, ktoré sú od toho istého výrobcu a z rovnakej série. Okrem toho je možné použiť iba jeden atribút, a tak napríklad povoliť všetky produkty od výrobcu, ktorého ID sa bude nachádzať v pravidle.

Číslo portu predstavuje identifikátor, pod ktorým je tento port označený na systéme, kde aplikácia beží. Jedná sa o veľmi užitočný atribút, pretože vďaka nemu môžeme zadefinovať pravidlá, ktoré priradia každému portu svoju funkciu. Je tak možné vylúčiť, aby sa do systému pripojilo zariadenie, ktorého funkcionalitu používateľ v systéme nepovolil. Okrem toho je možné zamedziť tomu, aby sa v systéme nachádzali dva USB pamäťové kľúče naraz.

[1] List všetkých atribútov vo všetkých deskriptoroch sa nachádza tu https://www.beyondlogic.org/usbnutshell/usb5.shtml.

Počet rozhraní hrá dôležitú úlohu z toho dôvodu, že zariadenie, ktoré spĺňa určitú funkcionalitu povolenú na systéme, môže zároveň vykonávať aj funkciu úplne odlišnú, ktorá môže byť potenciálne škodlivá. Príklad reprezentujúci tento problém môže byť USB pamäťový kľúč, ktorý okrem toho, že vykonáva očakávanú činnosť, má implementované aj rozhranie klávesnice. Takéto zariadenie by mohlo na systéme napáchať značné škody, preto je vhodné obmedziť USB pamäťové kľúče na iba jedno rozhranie.

Použitie pravidiel v aplikácii tohto typu je výhodné, pretože poskytuje flexibilný spôsob, ako určiť, čo má byť zablokované alebo povolené. Používateľ dostane na výber z vyššie spomínaných atribútov, pričom nie je potrebné využiť všetky. Pokiaľ niektorému atribútu hodnotu nepriradí, program bude pre daný atribút akceptovať akúkoľvek hodnotu.

Problém ale nastáva pri zariadeniach, ktoré majú viac ako jedno rozhranie. Keďže môže byť v každom pravidle trieda a podtrieda rozhrania iba raz, používateľ nemá možnosť obmedziť zariadenia s viacerými rozhraniami dostatočne presne. Preto je vhodné v aplikácii použiť skupiny pravidiel. Každá skupina bude identifikovateľná jedinečným číselným označením a bude obsahovať atribúty zariadenia. Do tejto skupiny bude možné vkladať ďalšie pravidlá, ktoré budú mať definované len atribúty rozhrania, pričom počet pravidiel nebude obmedzený. Príklad použitia skupín pravidiel je možné opäť ukázať na USB HID myši. Keďže väčšina USB myší má viacero bočných tlačidiel, musí mať okrem rozhrania definujúceho myš aj rozhranie pre klávesnicu. Pokiaľ by používateľ chcel napísať obyčajné pravidlo, ktoré by odpovedalo tejto myši, mohol by zadefinovať jedno rozhranie, teda napríklad myš, a obmedziť počet rozhraní na dve. To by ale nebránilo potenciálne škodlivým USB myšiam, aby druhé rozhranie implementovalo sieťovú kartu. Vďaka skupinám pravidiel je možné definovať triedu aj podtriedu oboch rozhraní, a tým odstrániť bezpečnostné riziko.

Zobrazenie pravidiel

Množstvo pravidiel v databáze nie je obmedzené a od používateľa sa neočakáva, že si ich bude pamätať. Preto je nutné, aby bolo možné všetky pravidlá zobraziť. Ak si používateľ zobrazenie vyžiada, na štandardný výstup terminálu sa vypíšu všetky definovateľné atribúty, pričom tie, u ktorých nebola definovaná žiadna hodnota budú zobrazovať hviezdičku.

Odstránenie pravidiel

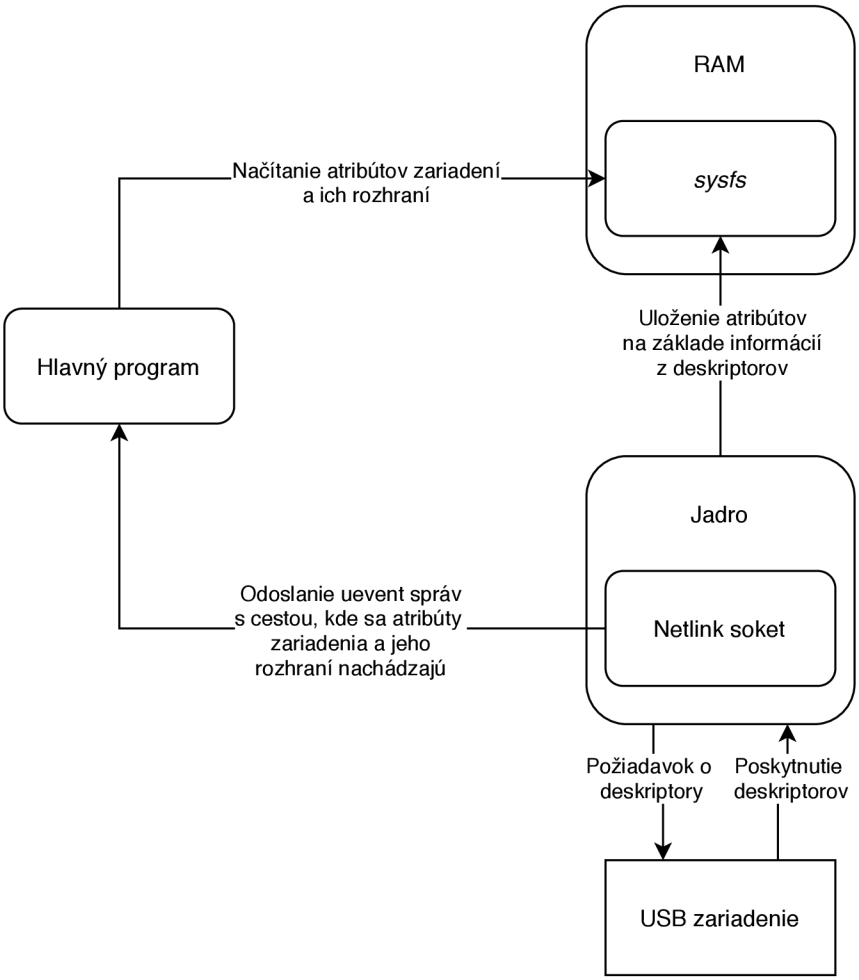
Keďže sa požiadavky na zariadenia, ktoré v systéme smú byť povolené, môžu meniť, je nevyhnutné, aby sa pravidlá dali jednoduchým spôsobom mazať. Pri ich vytváraní je každému pravidlu pridelený jedinečný identifikátor, ktorý je zobraziteľný spolu s ostatnými atribútmi. Toto číselné označenie je potom možné využiť k odstráneniu konkrétneho pravidla. Pre urýchlenie odstraňovania pravidiel je týchto označení možné použiť aj viac naraz.

4.4 Hlavný program

Po zadefinovaní pravidiel je potrebné pre ochranu systému pred nechcenými USB zariadeniami spustiť druhú časť aplikácie. Tá hneď po spustený prečíta všetky pravidlá z databázy a uloží ich do operačnej pamäte, aby s nimi bolo možné pracovať. Pri pripojení nového zariadenia aplikácia prečíta jeho atribúty a bude ich porovnávať voči pravidlám, kým nenájde zhodu. V prípade, že zhoda nastala, zariadenie dostane oprávnenie komunikovať s aplikáciami bežiacimi na systéme a používateľ bude oboznámený správou v termináli. Ak zhoda nenastala, aplikácia zariadenie zablokuje a vykonaný úkon taktiež oznámi pomocou správy.

Detekcia novo pripojeného USB zariadenia

Pre kontrolu USB zariadenia je potrebná jeho čo najrýchlejšia detekcia. Keďže aplikácia je umiestnená v používateľskej vrstve, nemá priamy prístup k hardvéru a detekciu musí ako prvé vykonať jadro systému. To so zariadením komunikuje za účelom získania potrebných informácií a ich uloženia v súborovom systéme `sysfs`. Následne po uložení všetkých dostupných atribútov do hierarchie priečinkov, odpovedajúcej hierarchii zariadenia, vytvorí jadro `uevent` správu, ktorú odošle pomocou Netlink soketu. Táto správa je určená pre podsystém UDEV, no vzhľadom na to, že správy sú odosielané pomocou spôsobu komunikácie multicast, dokáže ich zachytávať aj aplikácia na blokovanie USB zariadení. Táto správa obsahuje cestu ku koreňovému priečinku hierarchie priečinkov, teda získaním tejto správy aplikácia deteguje nové zariadenie a zároveň získava prístup k dôležitým informáciám, ako sú napríklad typ zariadenia alebo jeho výrobca. Podobným spôsobom sa aplikácia dostane k priečinkom rozhraní tohto zariadenia a následne aj k ich atribútom.



Obr. 4.2: Detekcia USB zariadenia

Činnosť jadra s ohľadom na detekciu USB zariadení a činnosť aplikácie pri detekcii je zobrazená na obrázku 4.2.

Spôsob blokovania a autorizácie zariadení

Operačný systém Linux za normálnych okolností (tzn. bez úprav nastavení týkajúcich sa USB) automaticky povoľuje všetky pripojené USB zariadenia. V prípade, že sa používateľ snaží zvýšiť bezpečnosť systému blokovaním určitých USB zariadení, to predstavuje problém. Kontrola každého novo pripojeného zariadenia podľa jeho atribútov a jeho následné zablokovanie v prípade, že je vyhodnotené ako potenciálne škodlivé, by stále predstavovalo bezpečnostné riziko. Počas kontroly je zariadenie pripojené a pokiaľ je jeho ovládač dostupný, operačný systém ho načíta do operačnej pamäte. To znamená, že môže vykonávať funkciu, na ktorú bolo určené, čo môže predstavovať aj škodlivú činnosť. Predpokladá sa, že kontrola bude vykonaná čo najrýchlejšie a zariadenia nedostane šancu v systéme napáchať škodu, no pokiaľ bude zariadenie pripojené i veľmi krátky čas, nie je možné to vylúčiť. Preto je potrebné zmeniť prístup systému, ktorý automaticky autorizuje všetky USB zariadenia.

Túto zmenu je možné vykonať vďaka možnostiam nastavení v súborovom systéme `sysfs`. Tieto možnosti majú opäť podobu súboru a nachádzajú sa v každom priečinku odpovedajúcom USB zbernici. Je možné ich nájsť na ceste `/sys/bus/usb/devices/usb*/interface_authorized_default`, kde je namiesto hviezdičky potrebné doplniť číslo zbernice, ktoré jej priradil systém [2]. Štandardná hodnota nachádzajúca sa v tomto súbore je 1, čo znamená, že USB zariadenia, respektívne všetky ich rozhrania, sú povoľované automaticky hneď po pripojení. Táto hodnota je tam zapísaná operačným systémom vždy pri jeho štarte, preto je nutné, aby bola prepísaná pri každom spustení aplikácie za predpokladu, že bude bežať počas celého behu systému. Namiesto pôvodnej hodnoty je potrebné do súboru zapísať 0, čo spôsobí, že rozhrania USB zariadení nebudú autorizované automaticky.

Je teda možné s istotou povedať, že zariadenie od momentu pripojenia do momentu skončenia jeho kontroly v systéme nemôže vykonať žiadnu škodlivú akciu. V prípade, že zariadenie je na základe kontroly autorizované, je potrebné všetky jeho rozhrania povoliť. Na to slúži opäť súbor v súborovom systéme `sysfs`, konkrétne v priečinku zodpovedajúceho rozhrania. Tento súbor má názov `authorized`, a po zmene hodnoty v súbore `interface_authorized_default` na hodnotu 0, bude takisto obsahovať hodnotu 0. Pre samotné povolenie rozhrania je potrebné do tohto súboru zapísať hodnotu 1.

Zariadenie však ani po povolení jeho rozhraní nebude v systéme schopné vykonávať svoju funkciu, a to z toho dôvodu, že ovládače pre jeho jednotlivé rozhrania nie sú načítané do operačnej pamäte. Ak by boli rozhrania autorizované automaticky, ovládač sám načíta operačný systém, avšak v prípade tejto aplikácie, kedy automaticky autorizované nie sú, je potrebné spustiť nahrávanie ovládačov manuálne. To je možné zaistiť vpísaním textového reťazca, popisujúceho konkrétne rozhranie, do súboru `/sys/bus/usb/drivers_probe`. Tento textový reťazec slúži ako jeho názov, pričom o danom rozhraní poskytuje informácie, ako napríklad na ktorej zbernici a v ktorom porte je pripojený. Príklady takýchto textových reťazcov pochádzajúcich z USB myši sú `1-3:1.0` a `1-3:1.1`, kde prvé číslo predstavuje číslo zbernice, druhé port, kde je zapojená, tretie aktuálne vybranú konfiguráciu zariadenia a posledné je číslo rozhrania. Po vpísaní týchto reťazcov do spomínaného súboru sú všetky potrebné ovládače v prípade ich dostupnosti nahraté a zariadenie sa tak stáva plne funkčným.

Ak majú USB zariadenia po kontrole a prípadnej autorizácii fungovať správne, je potrebné, aby výsledná aplikácia vykonala všetky vyššie spomínané úkony.

[2] Čísla všetkých USB zberníc v systéme je možné zistiť zadaním príkazu `lsusb` do terminálu.

Whitelisting vs blacklisting

Podstatou tejto časti aplikácie je blokovanie USB zariadení, ktoré, ako bolo spomenuté vyššie, je vykonávané na základe pravidiel. Spôsobov, ako sa aplikácia k týmto pravidlám môže postaviť je viacero, no najčastejšie používané sú blacklisting a whitelisting.

Blacklisting je metóda blokovania, ktorá spočíva v zakázaní potenciálne škodlivých prvkov. Pred použitím tejto metódy je potrebné vytvoriť zoznam zakázaných aplikácií, zariadení, prípadne web stránok, podľa toho, na aký spôsob zaistenia bezpečnosti sa použije. V prípade tejto aplikácie by bolo potrebné vytvoriť list zakázaných USB zariadení, podľa ktorého by sa novo pripojené zariadenia kontrolovali. Pokiaľ by pri kontrole nastala zhoda s niektorým z pravidiel, bolo by zariadenie považované za nechcené, a následne by bolo zablokované. Zvyšná USB funkcionalita by nebola nijak ovplyvnená.

Opačný princíp používa metóda whitelisting. Rovnako ako v predošlom prípade, opäť je potrebné vytvoriť list zariadení, no tentokrát sa tam budú nachádzať len tie, ktoré sú v systéme povolené. Ak sa skontrolované zariadenie na liste nenachádza, bude zablokované. To znamená, že je potrebné definovať množinu zariadení, ktoré môžu byť potencionálne pripojené a zároveň sú povolené.

Obe metódy majú svoje výhody aj nevýhody a výber jednej z nich je podmienený viacerými faktormi. Blacklisting je spôsob, ktorý býva v aplikáciách používaný častejšie a je to najmä z toho dôvodu, že vytvoriť list s potenciálne škodlivými zariadeniami je často jednoduchšie, ako ho vytvárať zo zariadení, ktoré chceme používať a v systéme nám neprekážajú. To ale prináša bezpečnostné riziko, pretože vytvoriť zoznam, ktorý by naozaj zahŕňal všetky potenciálne škodlivé zariadenia je veľmi obtiažne, prípadne nemožné, pretože vždy sa môže objaviť nový bezpečnostný problém, ktorý vo svete ešte nie je známy. Naopak whitelisting je metóda, pri ktorej je obtiažnejšie vytvoriť list povolených zariadení, no po jeho vytvorení je systém omnoho bezpečnejší než v predchádzajúcom prípade a s ohľadom na bezpečnosť vždy predstavuje lepšiu voľbu.

Po zvážení vyššie popísaných faktorov som sa rozhodol pre aplikáciu, ktorou sa zaoberá táto bakalárska práca, použiť metódu whitelisting. Je to najmä z toho dôvodu, že poskytuje väčšiu bezpečnosť, no taktiež kvôli tomu, že typov zariadení, ktoré bude bežný používateľ na systéme využívať vo väčšine prípadov nie je veľa a dajú sa odhadnúť dopredu. Nepredstavuje to taký problém, aký by to spôsobilo pri použití tejto metódy v antivírusovom programe, kde by bolo potrebné každú novo nainštalovanú aplikáciu pridať na whitelist. Šanca, že bude do systému pripojené zariadenie, ktoré je povolené, no používateľ naň pri vytváraní listu nemyslel je podstatne menšia. V prípade, že by k tomuto problému došlo, stále je možné list zariadení aktualizovať.

Kapitola 5

Implementácia

V kapitole s názvom implementácia sú popisované konkrétne postupy pri tvorbe aplikácie. V prvej časti sú vymenované použité technológie, vrátane odôvodnenia ich výberu, ďalšia časť popisuje spôsob implementácie hlavného programu a posledná sekcia sa venuje implementačným detailom programu na tvorbu pravidiel.

5.1 Použité technológie

V tejto sekcii sa nachádza súhrn použitých technológií spolu s odôvodnením, prečo boli zvolené. Konkrétne sa jedná o výber programovacieho jazyka, v ktorom má byť aplikácia implementovaná a knižnice pre prácu s databázami.

Výber programovacieho jazyka

Ako hlavný jazyk implementácie bol zvolený C++. Jeden z hlavných dôvodov tohto výberu je ten, že sa jedná o kompilovaný a nie interpretovaný jazyk, čo znamená, že sa vyznačuje rýchlosťou vykonávania programu aj napriek tomu, že C++ patrí medzi vysokoúrovňové jazyky. Je potrebné brať ohľad na rýchlosť výsledného programu, a to najmä z dôvodu, že aplikácia na blokovanie USB zariadení má predstavovať len medzičlánok vložený do už implementovaného systému a používateľ by nemal spozorovať oneskorenie vykonávania pôvodnej funkcie. Takto je možné dosiahnuť čo najkratšiu dobu kontroly novo pripojeného USB zariadenia a vykonávania patričných úkonov pri jeho zablokovaní alebo autorizácii.

Medzi ďalšie výhody patrí fakt, že sa jedná o objektovo orientovaný jazyk. Poskytuje tak prezentáciu dát v podobe objektov, čím sa problém riešený v programe približuje problému z reálneho sveta. Okrem toho je v C++ programoch možné využívať aj jazyk C a teda aj všetky jeho výhody, medzi ktoré, okrem iného, patrí nízkoúrovňový prístup k pamäti.

Jazyk na prácu s databázou

Pre tvorbu a prácu s databázou bola použitá knižnica SQLite, poskytujúca množstvo výhod. Bola implementovaná v jazyku C, čo umožňuje rýchle čítanie aj zápis dát. Vďaka tejto knižnici je možné dosiahnuť až 35% zrýchlenie oproti súborovému systému[1].

Najväčšou výhodou použitia tejto technológie je vysokoúrovňové narábanie s dátami, vďaka čomu je možné jednoduchým spôsobom pridávať, odoberať, prípadne filtrovať pra-

[1] Viac o podporujúcich štúdiách a spôsoboch merania je možné dozvedieť sa tu https://www.sqlite.org/fasterthanfs.html.

vidlá umiestnené v databáze, pričom nie je potrebné komunikovať s databázou ako so serverom a je možné zapisovať priamo na disk.

Používanie SQLite nezávisí na platforme, čo umožňuje používať rovnakú databázu na 32 bitových aj 64 bitových systémoch a problém nespôsobí ani little-endian alebo big-endian. Celá databáza sa nachádza v rámci jediného súboru, čo pri spomínanej nezávislosti od platformy používateľovi umožňuje bezproblémový prenos dát, pokiaľ sú tieto dáta plne kompatibilné s novým systémom. Môže sa napríklad jednať o prenos pravidiel v databáze zo starého počítača na nový s rovnakou konfiguráciou.

Ďalšou výhodou je jej veľkosť. Hranica, koľko môže databáza minimálne zaberať je 512 bajtov. Spolu to tak aj s vyššie popísanými výhodami vytvára nástroj vhodný na správu pravidiel pre aplikáciu na kontrolu a blokovanie USB zariadení.

5.2 Implementácia hlavného programu

Táto sekcia popisuje jednotlivé časti hlavného programu a spôsob ich implementácie, pričom celkovú funkciu rozdeľuje na inicializáciu prostredia, detekciu zariadení prebiehajúcu v nekonečnej slučke a kontrolu samotného zariadenia po jeho detekcii.

5.2.1 Inicializácia hlavného programu

Na začiatku programu je pre jeho funkciu nevyhnutné načítať z databázy používateľom vytvorené pravidlá. Najskôr je nad databázou potrebné vykonať niekoľko kontrol. Medzi ne patrí kontrola, či sa v databáze nachádza tabuľka pravidiel, samotné pravidlá a skupinové pravidlá. Následne sú pravidlá rozdelené na skupiny pravidiel, pravidlá patriace do skupín a samostatné pravidlá, pričom sú z nich vytvárané objekty patriace do odpovedajúcich tried (`RulesGroup, InterfaceRule, Rule`), ukladané do operačnej pamäte, kde budú uložené počas celého behu programu.

Okrem toho je potrebné vykonať inicializáciu prostredia tým, že sa zastaví automatická autorizácia USB zariadení zapísaním hodnoty 0 do súboru `/sys/bus/usb/devices/usb*/interface_authorized_default`. Túto akciu je nutné vykonať pre všetky USB zbernice a čísla konkrétnych zberníc doplniť do cesty na miesto hviezdičky. Aktuálne prítomné zbernice v systéme aplikácia získa iteráciou v priečinku `/sys/bus/usb/devices` pomocou knižnice `filesystem` a vyfiltrovaním len podstatných priečinkov cez regulárne výrazy.

5.2.2 Detekcia novo pripojených USB zariadení

Program využíva na objavovanie novo pripojených zariadení správy zachytené pomocou soketov z rodiny Netlink. Predmetom tejto správy je cesta k pripojenému zariadeniu. Program na základe mena posledného priečinka cesty za využitia regulárnych výrazov zhodnotí, či sa jedná o priečinok zariadenia alebo rozhrania.

V prípade, že sa jedná o priečinok zariadenia, program vytvorí objekt tomuto zariadeniu odpovedajúci (`Device`), a zo spomínaného priečinka nahrá do objektu všetky atribúty spomínané v sekcii návrhu programu na prácu s pravidlami 4.3. Ak sa jedná o priečinok rozhrania, opäť je vytvorený objekt odpovedajúci rozhraniu (`Interface`) a sú do neho nahrané potrebné atribúty, podobne ako tomu je v prípade zariadenia. Keďže každé rozhranie musí patriť pod zariadenie, je toto zariadenie potrebné nájsť a rozhranie mu priradiť. Na nájdenie zhody medzi zariadením a rozhraním program iteruje medzi všetkými uloženými zariadeniami a pokúša sa nájsť také, ktorého cesta predstavuje podreťazec cesty rozhrania.

Pri nájdení zhody je taktiež kontrolovaný počet už uložených rozhraní zariadenia voči celkovému počtu. Pokiaľ sa tieto dve hodnoty rovnajú, všetky rozhrania boli nájdené a je možné vykonať kontrolu zariadenia a jeho rozhraní za pomoci pravidiel.

5.2.3 Kontrola novo pripojeného zariadenia

Samotná kontrola je vykonávaná ihneď po inicializácii, a to len v prípade, že bola prijatá správa o pripojení zariadenia do systému. Ak boli v systéme prítomné USB zariadenia pred spustením programu, je ich funkcia naďalej umožnená a nie je potrebné pre ne vytvárať pravidlá do momentu, kým nenastane ich odpojenie a opätovné pripojenie.

Samotnú kontrolu je možné rozdeliť na dve časti. V prvej prebieha kontrola voči samostatným pravidlám. Každý jeden atribút zariadenia je porovnávaný voči atribútu pravidla, pričom sú vytvárané všetky možné kombinácie pravidiel a rozhraní zariadenia, až kým nedôjde k zhode. Pokiaľ sa tak nestane ani v jednom prípade, zariadenie v kontrole voči samostatným pravidlám neprešlo. Stále je možné nájsť zhodu so skupinou pravidiel, pokiaľ aspoň jedna existuje, v čom spočíva druhá časť kontroly.

Pri tejto kontrole sú najskôr skontrolované atribúty zariadenia voči atribútom skupiny pravidiel. Pokiaľ sa nájde zhoda s určitou skupinou, kontrola pokračuje vytváraním kombinácií z atribútov rozhraní a samotných pravidiel patriacich do skupiny. Pri tejto kontrole je možné využiť každé pravidlo, patriace do skupiny v rámci jedného USB zariadenia iba raz. Ak bola nájdená zhoda medzi atribútmi zariadenia a atribútmi skupiny a zároveň zhoda medzi všetkými rozhraniami a pravidlami patriacimi do tejto skupiny, kontrola končí úspešne, čo znamená že zariadenie bude autorizované.

Pre autorizáciu zariadenia je potrebné, aby aspoň jedna z častí kontroly našla zhodu medzi pravidlom a zariadením. V takom prípade sú povolené rozhrania zariadenia a je spustené načítanie ich ovládačov. Pokiaľ zhoda s pravidlom nájdená nebola, zariadenie je deautorizované. V oboch prípadoch je o tom používateľ oboznámený pomocou terminálu, kde je vypísaná informačná hláška o tom, či zariadenie bolo autorizované/deautorizované. Zároveň obsahuje aj detaily o tomto zariadení, vďaka ktorým môže používateľ zistiť o aký typ zariadenia sa jednalo a taktiež obsahuje informáciu o tom, s akým pravidlom bola zhoda nájdená.

5.2.4 Ukončenie programu a ošetrenie chýb

Funkcia hlavného programu je implementovaná pomocou nekonečnej slučky, čo znamená, že program sa neukončí sám, pokiaľ nedôjde k chybe. Pre ukončenie jeho funkcie je potrebné zaslať pomocou terminálu signál prerušenia, a to stlačením klávesovej skratky `CTRL + C`. Po ukončení programu je zavolaná funkcia, ktorá vráti prostredie do pôvodného stavu. Tým je myslené opätovné povolenie automatickej autorizácie USB zariadení, zapísaním hodnoty `1` do súboru `interface_authorized_default` pre každú zbernicu.

Pri behu programu sa môžu vyskytnúť chyby, ktoré spôsobia, že program nebude ďalej schopný pokračovať. Ošetrené sú pomocou vyhodenia výnimiek, ktoré sú vždy zachytávané vo funkcii `main`. Po zachytení takejto výnimky je taktiež zavolaná funkcia, ktorá vráti prostredie do pôvodného stavu, do terminálu sa vypíše chybová hláška a program sa ukončí.

5.3 Implementácia programu na tvorbu pravidiel

V tejto sekcii sa nachádza popis implementácie jednotlivých častí programu na prácu s pravidlami. Opísané je spracovanie parametrov zadaných pri spustení aplikácie a následné akcie, ktoré tieto parametre vyvolajú.

5.3.1 Spracovanie vstupných parametrov

Celý program na prácu s pravidlami je implementovaný v triede `Database`, ktorá obsahuje premenné na ukladanie parametrov zadaných pri spustení programu pomocou terminálu. Samotné spracovanie parametrov a ich argumentov je podstatne uľahčené vďaka knižnici `getopt`. Pri spracovaní sú nastavené indikátory určujúce aký úkon má aplikácia s pravidlami vykonať a všetky poskytnuté atribúty sú uložené.

5.3.2 Tvorba nového pravidla

Jeden zo spomínaných indikátorov môže predstavovať pridanie nového pravidla. Pred tým, než môže k samotnému pridaniu dôjsť, je potrebné vykonať niekoľko kontrol. Vytvorenie pravidla nie je možné vykonať bez toho, aby používateľ zadal aspoň jeden atribút, v čom spočíva prvá kontrola. Ďalšia overuje, či používateľ žiada o pridanie pravidla do skupiny pravidiel. V takom prípade je potrebné skontrolovať, či pravidlo neobsahuje atribúty zariadenia, prípadne či skupina pravidiel so zadaným ID existuje.

Následne je zo zadaných atribútov vytvorený textový reťazec, ktorý predstavuje SQL príkaz na pridanie nového záznamu do databázy. Pre každý atribút je potrebné počítať s možnosťou, že bude v SQL príkaze ako prvý, čo znamená, že je nutné implementovať dva rozdielne spôsoby pridávania atribútu do reťazca.

Každý z atribútov je možné zadávať v takej forme, ako sú zobrazované vo výstupoch príkazov `lsusb -v` a `lsusb -t`, čo je zhodné s tým, ako sú prezentované v súborovom systéme `sysfs`. Okrem toho je možné pre atribút USB port zadať aj inú formu, ktorú je vhodné použiť, ak používateľ počítať s pripojením externého USB rozbočovača. Napríklad je možné pre port zadať hodnotu „3.“, čo určuje, že ak bude v porte 3 zapojený USB rozbočovač, je možné pripojiť zariadenie, definované daným pravidlom do ktoréhokoľvek portu v rámci tohto rozbočovača.

5.3.3 Zobrazenie stávajúcich pravidiel

Pomocou parametrov je možné zvoliť zobrazenie všetkých pravidiel, čo je implementované pomocou SQL príkazu na zobrazenie celého obsahu tabuľky, kde sú uložené pravidlá. Výsledok je spracovaný a zobrazený tak, že v každom pravidle sú zobrazené všetky atribúty s priradenými hodnotami. Pokiaľ niektorý z atribútov hodnotu nemá, je na miesto nej zobrazená hviezdička.

5.3.4 Odstránenie vybraných pravidiel

Pre odstránenie pravidiel je potrebné zadať patričný parameter spolu s jedným alebo viacerými identifikátormi pravidiel. Samotné vymazanie pravidla je opäť realizované pomocou SQL príkazu a je rozdelené na dve časti. Prvá časť je určená pre odstraňovanie skupín pravidiel. V takom prípade je okrem samotnej skupiny potrebné vymazať aj pravidlá nachádzajúce sa v nej. Pokiaľ sa nejedná o skupinu, sú vymazané len tie pravidlá, ktoré zadal používateľ v podobe argumentu.

5.3.5 Vytvorenie predvolených pravidiel

Vo viacerých systémoch sa vyskytujú zariadenia, ktoré sa javia ako USB zariadenia, no sú zabudované priamo v počítači. Najčastejšie sa jedná o notebooky a príklad takého zariadenia môže byť web kamera. Takýchto zariadení môže byť niekoľko, čo za predpokladu, že sú v systéme povolené, predlžuje celý proces tvorby pravidiel. Z toho dôvodu aplikácia obsahuje možnosť tvorby pravidiel na základe zariadení pripojených v systéme, pomocou zadania jediného parametra pri spustení aplikácie. Okrem toho získa používateľ šancu pripojiť do systému všetky USB zariadenia, ktoré by mohol v budúcnosti potrebovať a vytvoriť pre ne pravidlá bez nutnosti zadávať každé jedno manuálne.

Vytváranie predvolených pravidiel je implementované tak, že pomocou knižnice `filesystem` sú objavené všetky položky v priečinku `/sys/bus/usb/devices/`, ktoré predstavujú zariadenie, rozhranie alebo USB zbernicu. Následne sú pomocou regulárnych výrazov vybrané iba zariadenia a USB zbernice, pre ktoré sú vytvorené skupiny pravidiel. V rámci týchto zariadení alebo zberníc sú vyhľadané všetky dostupné rozhrania, ku ktorým sú taktiež vytvorené pravidlá, vložené do odpovedajúcich skupín. Týmto spôsobom je možné dosiahnuť čo najpresnejší popis zariadení, ktoré sa práve nachádzajú zapojené v systéme.

Kapitola 6

Testovanie a súhrn výsledkov

V tejto kapitole sa nachádza experimentálne overenie funkčnosti výslednej aplikácie pomocou testovania pre rôzne vstupy a taktiež súhrn výsledkov. Na začiatku sú popísané podmienky, za akých testovanie prebiehalo. Následne sú podrobne zobrazené testované zariadenia a samotné testy, pričom sa bude meniť databáza pravidiel. Na konci kapitoly sú zhrnuté možné vylepšenia do budúcnosti a aplikácia je porovnaná s existujúcimi riešeniami.

6.1 Informácie o prostredí

Testovanie prebiehalo na dvoch zariadeniach s operačným systémom Linux. Konkrétne sa jedná o notebook a tablet. Operačný systém, verzia jadra a architektúra týchto zariadení sú popísané v tabuľke 6.1.

Zariadenie	Názov operačného systému	Verzia jadra	Architektúra
Notebook	Fedora 31	5.4.8	x86-64
Tablet	Kali GNU/Linux Rolling	4.17.0	x86-64

Tabuľka 6.1: Súhrn informácií o prostredí

Kým jeden z operačných systémov bol vytvorený na systéme Debian a pochádza od firmy Offensive Security, druhý pochádza od firmy Red Hat, pričom spoločne pokrývajú veľkú časť dnes používaných distribúcií. Rovnako tak boli použité rôzne verzie jadra, aby bola preukázaná funkčnosť aplikácie na rozdielnych systémoch. Obidve zariadenia disponujú dvoma portmi USB typu B a jedným typu C.

6.2 Testované USB zariadenia

Pre správne otestovanie aplikácie boli vybrané USB zariadenia s rôznymi vlastnosťami. Ide najmä o rôzne triedy a podtriedy, ale aj rozdielny počet rozhraní. Testovanie bolo vykonané na USB myši, hernom ovládači, USB pamäťovom kľúči, externom disku, smartfóne, USB – TTL prevodníku a USB rozbočovači. Informácie o zariadeniach v nasledujúcich testoch boli získané pomocou príkazu `lsusb -v`. Všetky testy boli vykonané na notebooku aj tablete, no výsledné chovanie a celkový výstup bol zhodný.

6.3 Priebeh testovania

Jednotlivé testy boli vykonávané na jednom, ale aj na niekoľkých zariadeniach naraz, pričom boli pripojené do systému vždy až po štarte aplikácie. Pri každom teste sú zobrazené pravidlá, získané pomocou aplikácie na prácu s pravidlami a spracované do tabuľky kvôli kompaktnosti. Taktiež sú uvedené informácie o zariadení a výstup z hlavnej aplikácie.

Test na konkrétnom porte

Prvý test spočíval v otestovaní jednoduchého pravidla po zapojení USB herného ovládača na konkrétnom porte, vopred definovanom v pravidle. Atribúty pravidla a zariadenia sú spracované v tabuľkách 6.2 a 6.3. Výstup hlavnej aplikácie je možné vidieť na 6.1

Typ	Trieda	Podtrieda	ID výrobcu	ID produktu	Počet rozhraní
Zariadenie	00	00	0458	1004	1
Rozhranie	03	00			

Tabuľka 6.2: Atribúty herného ovládača

ID	Trieda zariadenia	Podtrieda zariadenia	Výrobca	Produkt
1	*	*	*	*
Trieda rozhrania	**Podtrieda rozhrania**	**Počet rozhraní**	**Port**	**ID skupiny**
03	00	*	1	*

Tabuľka 6.3: Výpis informácií z pravidla

```
Disconnecting potentially dangerous device with 0458:1004 vendor:product ID and class:
00, subclass: 00
with interface(s):
(1) class: 03, sublclass: 00
on port 3

Device with 0458:1004 vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: 03, sublclass: 00
authorized on port 1
Authorization based on rule 1
```

Obr. 6.1: Výstup hlavnej aplikácie

Po pripojení USB herného ovládača do portu číslo 3, bolo zariadenie odmietnuté a odpojené. Aplikácia zariadenie autorizovala až po zapojení do portu s číslom 1, pretože jedine v tomto prípade sa zhodovali trieda a podtrieda rozhrania ako aj číslo portu. Zvyšné atribúty v pravidle definované neboli a aplikácia v takom prípade povolí akúkoľvek hodnotu.

Test pre konkrétne zariadenie

Podstata druhého testu bola vytvoriť pravidlo, ktoré by povolilo iba jeden presne určený typ zariadenia. Atribúty v pravidle sú zhodné s atribútmi zariadenia okrem čísla portu, čo znamená, že v systéme je povolené iba jedno konkrétne USB zariadenie a nezáleží na tom,

v akom porte. Atribúty pravidla a zariadenia sú spracované v tabuľkách 6.4 a 6.5. Výstup hlavnej aplikácie je možné vidieť na 6.2.

Typ	Trieda	Podtrieda	ID výrobcu	ID produktu	Počet rozhraní
Zariadenie	00	00	05e3	0736	1
Rozhranie	08	06			

Tabuľka 6.4: Atribúty USB pamäťového kľúča

ID	Trieda zariadenia	Podtrieda zariadenia	Výrobca	Produkt
1	00	00	05e3	0736
Trieda rozhrania	**Podtrieda rozhrania**	**Počet rozhraní**	**Port**	**ID skupiny**
08	06	1	*	*

Tabuľka 6.5: Atribúty použitého pravidla

Device with 05e3:0736 vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: 08, sublclass: 06
authorized on port 3
Authorization based on rule 1

Disconnecting potentially dangerous device with 09da:054f vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: 03, sublclass: 01
(2) class: 03, sublclass: 01
on port 3

Obr. 6.2: Výstup hlavnej aplikácie

Vyššie spomínané zariadenie, predstavujúce USB pamäťový kľúč, bolo po pripojení autorizované. Po zapojení akéhokoľvek iného zariadenia, čo v tomto teste reprezentuje USB HID myš, k povoleniu nedošlo.

Test zariadenia s viacerými rozhraniami

Podstata tretieho testu má ukázať správnosť funkcie skupín pravidiel, ktoré boli vytvorené najmä pre zariadenia s viacerými rozhraniami. Aby bolo možné také zariadenie čo najpresnejšie opísať a vyhnúť sa tak situácii, pri ktorej by zariadenie poskytovalo okrem povolenej funkcionality aj škodlivú, je potrebné vytvoriť skupiny pravidiel a niekoľko pravidiel do nich vložiť. Ako testovaný subjekt poslúži USB HID myš, popísaná v tabuľke 6.6, ktorá má dve rozhrania, kvôli čomu je potrebné do skupiny vložiť dve pravidlá. Skupina a pravidlá sú popísané v tabuľkách 6.7, 6.8 a 6.9. Výstup hlavnej aplikácie je možné vidieť na 6.3.

Po pripojení spomínanej myši pri zobrazených pravidlách došlo k autorizácii. Pravidlá vložené do skupiny sú zhodné, čo je potrebné z toho dôvodu, že zariadenie má dve rovnaké rozhrania. Pokiaľ by sa v skupine nachádzalo iba jedno pravidlo, druhé rozhranie by nenašlo zhodu so žiadnym pravidlom a k autorizácii by nedošlo.

Typ	Trieda	Podtrieda	ID výrobcu	ID produkut	Počet rozhraní
Zariadenie	00	00	09da	054f	2
Rozhranie	03	01			
Rozhranie	03	01			

Tabuľka 6.6: Atribúty USB myši

ID	Trieda zariadenia	Podtrieda zariadenia	Výrobca	Produkt
1	00	00	09da	054f
Trieda rozhrania	**Podtrieda rozhrania**	**Počet rozhraní**	**Port**	**ID skupiny**
*	*	*	1	1

Tabuľka 6.7: Atribúty skupiny

Test na USB rozbočovač

Posledný test mal dokázať schopnosť aplikácie vysporiadať sa s USB rozbočovačmi. Do každého USB portu bolo zapojené zariadenie a na základe týchto zariadení boli pomocou aplikácie na tvorbu pravidiel vytvorené tzv. predvolené pravidlá. Spolu tak vzniklo dvadsaťtri pravidiel, ktoré sa viažu na konkrétne zariadenie a konkrétny port. Pravidlá ani atribúty zariadení s ohľadom na rozsiahlosť nie sú zobrazené. Výstup hlavnej aplikácie je možné vidieť na 6.4.

Po zapojení USB rozbočovača na porte 3 došlo k jeho autorizácii, k čomu sa viažu dve prvé správy z výpisu hlavnej aplikácie. Testovaný USB rozbočovač, rovnako ako aj koreňový rozbočovač, podporuje USB zariadenia verzie 3.0 ale aj 2.0 a nižšej. Z toho dôvodu je rozdelený na dve zariadenia, vyžaduje pre správnu funkciu dve pravidlá a aplikácia ho musí overiť dvakrát. Aplikácia overila všetky zvyšné zariadenia zapojené v samotnom rozbočovači a na základe pravidiel ich povolila.

6.3.1 Záver testovania

Každý zo štyroch testov bol zameraný na inú funkciu alebo schopnosť aplikácie. Zariadenia s uvedenými atribútmi boli testované voči sade vypísaných pravidiel. Na základe týchto vstupov aplikácia vytvorila adekvátny výstup. Tento výstup sa skladal z blokovania alebo povolenia zariadenia a oznámení o patričnom úkone používateľovi pomocou terminálu.

Výstup pri každom teste splnil očakávania a aplikáciu je možné považovať za plne funkčnú, pričom je vhodná jednak pre domáce používanie nenáročným používateľom, ako aj pre firmy a ich zamestnancov, ktoré chcú chrániť svoje systémy pred potenciálnym nebezpečím.

6.4 Návrh ďalších vylepšení

Testovanie dokázalo, že aplikácia je schopná obmedzovať prístup USB zariadení na základe ich vlastností a spĺňa tak stanovené požiadavky. Aj napriek tomu, že aplikácia je funkčná ako celok, je možné navrhnúť vylepšenia, ktoré by mohli byť v budúcnosti implementované.

Medzi najpodstatnejšie vylepšenie patrí podrobnejšie rozpoznávanie HID zariadení. Tak, ako bolo v podsekcii 2.1.5 spomenuté, USB zariadenia využívajúce protokol HID tvoria podskupinu všetkých USB zariadení, ktoré majú spoločné vlastnosti. Okrem iného, ich funkcionalitu je možné ohraničiť a popísať pomocou preddefinovaných skupín, nazývajúcich

ID	Trieda zariadenia	Podtrieda zariadenia	Výrobca	Produkt
2	*	*	*	*
Trieda rozhrania	**Podtrieda rozhrania**	**Počet rozhraní**	**Port**	**ID skupiny**
03	01	*	*	1

Tabuľka 6.8: Atribúty prvého pravidla zo skupiny

ID	Trieda zariadenia	Podtrieda zariadenia	Výrobca	Produkt
3	*	*	*	*
Trieda rozhrania	**Podtrieda rozhrania**	**Počet rozhraní**	**Port**	**ID skupiny**
03	01	*	*	1

Tabuľka 6.9: Atribúty druhého pravidla zo skupiny

sa stránky použitia (Usage page) a použitie (Usage). Pomocou nich dokážeme zadefinovať, že zariadenie má slúžiť napríklad ako klávesnica a taktiež, aké konkrétne klávesy na nej môžu byť stláčané. Predstavuje to tak spôsob, ako popisovať USB HID zariadenia (tie, ktoré majú v rozhraní definovanú triedu `03`) ešte detailnejším spôsobom, čo by mohlo byť využité pri tvorbe pravidiel.

Najpodstatnejším dôvodom, prečo aplikácia nerozpoznáva HID zariadenia detailnejšie je to, že jadro neposkytuje možnosť nastavenia, či má byť novo pripojené HID zariadenie automaticky povolené tak, ako je to v prípade zariadení a rozhraní. K tomu, aby aplikácia získala k HID deskriptoru prístup pomocou súborového systému `sysfs` je najskôr potrebné dané rozhranie povoliť a načítať HID ovládač, čo vedie k možnosti tohto zariadenia okamžite vykonávať svoju funkciu ešte pred jeho kontrolou. Preto je nutné pred implementáciou tohto vylepšia nájsť spôsob, ako deskriptor získať bez toho, aby bolo zariadenie uvedené do funkcie načítaním jeho ovládača.

Okrem toho je medzi ďalšie vylepšenia možné zaradiť vytvorenie grafického rozhrania, čo by síce viedlo k strate jednoduchosti aplikácie, no mohlo pôsobiť používateľsky prívetivejšie. Taktiež správy, ktoré aplikácia zobrazuje v termináli by mohli byť zobrazované pomocou vyskakujúcich notifikácií.

6.5 Porovnanie s existujúcimi nástrojmi

Výsledná aplikácia poskytuje podstatne vyššiu flexibilitu než poskytoval prvý prípad existujúcich riešení, popísaný tu 3.1. Možností, na základe ktorých je možné USB zariadenia blokovať je viac a pravidlá je možno tvoriť aj mazať jednoduchším spôsobom. Vo výslednej flexibilite detekcie a popisu USB zariadení jednoznačne vyhrávajú UDEV pravidlá, no tento spôsob je značne komplikovaný a tvorba pravidiel je pre bežného používateľa nezrozumiteľná. Zároveň si samotné blokovanie a automatickú autorizáciu musí používateľ vyriešiť sám. V oboch prípadoch došlo po implementácii aplikácie k zlepšeniu – pridávanie pravidiel je podstatne jednoduchšie a blokovanie alebo povolenie zariadenia automaticky zabezpečené.

Najväčšia podobnosť je medzi aplikáciou USBGuard a aplikáciou, ktorá je výsledkom tejto bakalárskej práce. Flexibilita tvorby pravidiel je pri aplikácii USBGuard väčšia. Väčšia flexibilita predstavuje väčšie množstvo možností, ktoré je možné pri tvorbe pravidiel využiť, čo zároveň prináša zťaženie samotnej tvorby a zvyšuje požiadavky na znalosť používateľa. Výsledok tejto práce mala byť aplikácia, ktorá bude poskytovať pohodlné prostredie na blokovanie USB zariadení a zároveň bude zachovávať maximálnu jednoduchosť a zrozumi-

Device with 09da:054f vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: 03, sublclass: 01
(2) class: 03, sublclass: 01
authorized on port 1
Authorization based on group 1

Obr. 6.3: Výstup hlavnej aplikácie

teľnosť, čo bolo splnené. Manuál poskytnutý na zrozumiteľné používanie aplikácie a tvorbu pravidiel je dostatočne podrobný a poskytuje všetky nevyhnutné teoretické poznatky, čo predstavuje ďalšiu výhodu oproti aplikácii USBGuard. Okrem toho bol pridaný atribút počet rozhraní, určený na jednoduchý no účinný spôsob blokovania zariadení s viacerými rozhraniami, ktoré by mohli okrem požadovanej funkcie vykonávať aj škodlivú činnosť. Taktiež je možné definovať triedu a podtriedu nie len rozhrania, ale aj zariadenia, čo umožní blokovať napríklad USB rozbočovače.

Device with 05e3:0610 vendor:product ID and class: 09, subclass: 00
with interface(s):
(1) class: 09, sublclass: 00
authorized on port 3
Authorization based on group 6

Device with 05e3:0626 vendor:product ID and class: 09, subclass: 00
with interface(s):
(1) class: 09, sublclass: 00
authorized on port 3
Authorization based on group 1

Device with 05e3:0736 vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: 08, sublclass: 06
authorized on port 3.1
Authorization based on group 11

Device with 067b:2303 vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: ff, sublclass: 00
authorized on port 3.2
Authorization based on group 3

Device with 2717:ff40 vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: ff, sublclass: ff
authorized on port 3.3
Authorization based on group 8

Device with 174c:1053 vendor:product ID and class: 00, subclass: 00
with interface(s):
(1) class: 08, sublclass: 06
authorized on port 3.4
Authorization based on group 5

Obr. 6.4: Výstup hlavnej aplikácie

Kapitola 7

Záver

Cieľom tejto práce bolo vytvorenie aplikácie, ktorá by bola schopná obmedzovať prístup určitých USB zariadení. Pri návrhu a tvorbe sa kládol dôraz najmä na zrozumiteľnosť a jednoduchosť, aby mohla byť aplikácia používaná aj menej skúsenými používateľmi. Zároveň bolo potrebné poskytnúť dostatočnú flexibilitu pri popisovaní zariadení, ktoré majú byť povolené, prípadne zablokované.

Na základe teoretických poznatkov bol vypracovaný návrh, kde bola aplikácia rozdelená na dve časti. Prvá časť predstavuje nástroj určený na popis USB zariadení pomocou atribútov. Na základe tohto popisu, nazývaného pravidlá, rozhoduje druhá časť aplikácie o tom, či má byť novo pripojené zariadenie v systéme povolené. Navrhnutá bola tak, aby jej na funkciu postačoval používateľský priestor, pričom využíva správy, ktoré jadro odosiela pri detekcii nového hardvéru.

Po dokončení návrhu bola aplikácia úspešne implementovaná. Bol vytvorený nástroj, kde je možné pomocou terminálu tvoriť nové pravidlá, ktoré sú ukladané do databázy. Taktiež bola implementovaná hlavná aplikácia, ktorá pravidlá z databázy načíta do operačnej pamäte a využíva ich na kontrolu detegovaných zariadení. Výsledkom kontroly je to, či má byť dané zariadenie pripojené a uvedené do funkcie.

Výsledná aplikácia bola otestovaná pomocou štyroch testov, pričom každý z nich sa zameriaval na inú funkcionalitu alebo vlastnosť implementovanej aplikácie. Na začiatku boli presne stanovené podmienky, za akých bude testovanie prebiehať a následne boli zhodnocované výstupy aplikácie. Výsledky testovania potvrdili správnu funkčnosť finálneho produktu a je možné ho v praxi použiť na zlepšovanie bezpečnosti systému.

Voči existujúcim riešeniam bolo vykonaných niekoľko zmien. Najpodstatnejšia z nich predstavuje zjednodušenie obmedzovania USB zariadení oproti aktuálne dostupným možnostiam. Taktiež bol vytvorený zrozumiteľný návod, poskytujúci všetky potrebné informácie a poznatky. Tieto vlastnosti spôsobili, že aplikácia je vhodná aj pre používateľov menej znalých v oblasti USB na operačnom systéme Linux. Okrem toho bol vytvorený zoznam atribútov, určených na popis zariadení tak, aby poskytoval dostatočnú flexibilitu, no zároveň si zachoval zrozumiteľnosť.

Aplikácia je plne funkčná ako celok, no je možné navrhnúť zopár vylepšení. Medzi ne patrí najmä podrobnejší spôsob popisu USB HID zariadení, prípadne vytvorenie grafického rozhrania na tvorbu pravidiel a zobrazovanie notifikácií o zariadení, ktoré sa do systému práve pripojilo.

Literatúra

[1] *Charakteristika sběrnice USB*. 2019. Dostupné z: https://www.fit.vutbr.cz/study/courses/IPZ/public/texty/usb/usb2019.pdf.

[2] Axelson, J. *USB Complete: The Developer's Guide*. 4. vyd. Lakeview Research, 2009. ISBN 1931448086.

[3] Brain, M. *How USB Ports Work*. 2000. Dostupné z: https://computer.howstuffworks.com/usb2.htm.

[4] Docile, E. *Tutorial on how to write basic udev rules in Linux*. 2018. Dostupné z: https://linuxconfig.org/tutorial-on-how-to-write-basic-udev-rules-in-linux.

[5] Hallinan, C. *Embedded Linux Primer: A Practical Real-World Approach*. 2. vyd. USA: Prentice Hall Press, 2010. ISBN 0137017839.

[6] He, K. K. *Kernel Korner - Why and How to Use Netlink Socket*. 2005. Dostupné z: https://www.linuxjournal.com/article/7356.

[7] Hoffman, C. *Don't Panic, But All USB Devices Have a Massive Security Problem*. 2017. Dostupné z: https://www.howtogeek.com/203061/don%E2%80%99t-panic-but-all-usb-devices-have-a-massive-security-problem/.

[8] Horman, N. *Understanding And Programming WithNetlink Sockets*. 2004. Dostupné z: https://people.redhat.com/nhorman/papers/netlink.pdf.

[9] Kerrisk, M. *Sysfs(5) - Linux manual page*. 2017. Dostupné z: http://man7.org/linux/man-pages/man5/sysfs.5.html.

[10] Kroah Hartman, G. *Proceedings of the Linux Symposium*. 2003.

[11] McLarnon, M. *The Human Interface Device (HID) Attack, aka USB Drive-By*. 2016. Dostupné z: https://www.cyberpointllc.com/posts/cp-human-interface-device-attack.html.

[12] Mochel, P. a Murphy, M. *Sysfs - _The_ filesystem for exporting kernel objects*. 2003. Dostupné z: https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt.

[13] Peacock, C. *USB in a NutShell - Chapter 5 - USB Descriptors*. 2018. Dostupné z: https://www.beyondlogic.org/usbnutshell/usb5.shtml.

[14] Shusain. *How to disable USB storage on Linux*. 2017. Dostupné z: https://linuxtechlab.com/disable-usb-storage-linux/.

Príloha A

Obsah priloženého pamäťového média

Zložky:

- **bin** – V tejto zložke sa po kompilácii projektu budú nachádzať binárne súbory, pomocou ktorých je výslednú aplikáciu možné spustiť.
- **build** – Po kompilácii sa tu budú nachádzať objektové súbory, ktoré ešte neboli zlinkované.
- **database** – Zložka bude po spustení aplikácie obsahovať súbor, určený na uloženie databázy pravidiel.
- **src** – Zdrojové súbory aplikácie implementované v jazyku C++.
- **text** – Zdrojové súbory, pomocou ktorých je možné vygenerovať tento text vo formáte pdf.

Súbory:

- **Makefile** – Súbor, slúžiaci na zjednodušenie prekladu aplikácie.
- **xkrajc21_projekt.pdf** – Tento text vo formáte PDF.
- **README.md** – Návod, obsahujúci prerekvizity k prekladu aplikácie a popis ovládania obi dvoch častí aplikácie, vrátane nevyhnutných poznatkov a ukážkových príkladov.