Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Zabezpečenie pracovnej stanice s OS Linux

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Jaroslav Janáček, PhD.

Bratislava, 2019
Bc. Peter Vašut

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Peter Vašut
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Zabezpečenie pracovnej stanice s OS Linux
Securing Linux Workstation

Anotácia: Práca sa zaoberá možnosťami zvýšenia bezpečnosti pracovnej stanice s OS Linux využitím pokročilej bezpečnostnej technológie AppArmor tak, aby používateľ mohol bezpečne pracovať s citlivými údajmi a súčasne vykonávať bežné činnosti spojené so zvýšeným bezpečnostným rizikom (napr. prístup k webu, spracovanie elektronickej pošty, spracovanie potenciálne škodlivých dokumentov, spúšťanie potenciálne škodlivých aplikácií).

Cieľ: Identifikovať bežné scenáre použitia pracovnej stanice, navrhnúť kategorizáciu aplikácií, analyzovať možnosti súčasných distribúcií OS Linux, navrhnúť a implementovať ľahko použiteľné riešenie umožňujúce bezpečné súčasné používanie aplikácií rôznych kategórií.

Vedúci: RNDr. Jaroslav Janáček, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 21.11.2018

Dátum schválenia: 21.11.2018 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem školiteľovi RNDr. Jaroslavovi Janáčkovi, PhD. za nadštandardné množstvo času a úsilia, ktoré mi pri písaní tejto práce venoval.

Abstrakt

V práci sa zaoberáme zvýšením bezpečnosti počítača s OS Linux. V rámci riešenia navrhujeme grafový model, ktorého vrcholy popisujú kategórie aplikácií, požiadavky aplikácií, možné rozhodnutia používateľa a vzťah k implementácii prostredníctvom technológie AppArmor. Tento graf je následne možné použiť pri uvažovaní o bezpečnosti aplikácie, ako aj pri používaní navrhnutého konfiguračného programu.

Kľúčové slová: Linux, bezpečnosť, AppArmor

Abstract

This work is focused on improving security of computer with OS Linux. We have developed graph model, in which vertices represent categories of applications, application requirements, user decisions and implementation using AppArmor. This graph can be helpful during evaluation of security of application, or it can be used by configurator developed for this work.

Keywords: Linux, security, AppArmor

Obsah

Zoznam obrázkov

Úvod

Prístup k informačným technológiám sa v priebehu niekoľkých desaťročí rozšíril z úzkych kruhov vedcov a ďalších odborníkov na široké masy ľudí. Zároveň s rozšírením týchto technológií rastie aj množstvo dát súkromných osôb aj firiem. Tento rast prináša nové výzvy v oblasti zabezpečenia týchto dát. Vývojári a administrátori systémov sa zvyknú venovať zabezpečeniu serverov, avšak používateľský počítač v domácnosti alebo v malej firme zostane často bez väčšej pozornosti. Je pochopiteľné, že domáci používateľ nemá také požiadavky na bezpečnosť ako firma, ktorej existencia závisí od ochrany svojich informačných systémov. Avšak veríme, že aj skúsenejší domáci používateľ by mal mať aspoň základný prehľad o tom, aké operácie vykonávajú programy v jeho počítači. Administrátor vo firme, alebo šikovný známy v domácnosti by mal byť schopný dopomôcť technicky menej zdatnému používateľovi obmedziť, čo robia programy na jeho počítači. Veríme, že operačný systém Linux je dôležitou súčasťou prítomnosti aj budúcnosti používateľských počítačov, preto sa v tejto práci venujeme práve zabezpečeniu používateľského počítača s Linuxom.

Poznámka: Paralelne s touto prácou prebieha ďalšia práca [19] zameraná na *SELinux*. Táto práca sa bude zameriavať na *AppArmor* a na formálne definovanie požiadaviek podľa ktorého bude možné vytvoriť *AppArmor* profil.

Kapitola 1

Popis AppArmoru

V tejto kapitole uvedieme základný popis *AppArmoru*, ktorý poslúži ako dostatočný úvod pre potreby tejto práce. Nejde o kompletnú dokumentáciu.

AppArmor je bezpečnostný systém pre OS Linux určený na obmedzovanie jednotlivých procesov. Základné informácie o projekte sú získané z Wiki stránky projektu [3]. Zámer projektu a jeho popis možno nájsť na stránke o projekte (*About*) [1]. Technické podrobnosti pochádzajú z Dokumentácie projektu [2].

1.1 Základná architektúra

AppArmor zavádza *povinné riadenie prístupu* (*mandatory access control*, *MAC*). Administrátor systému definuje *profily* pre jednotlivé aplikácie. Nie je nutné definovať profil pre všetky aplikácie (*selective confinement*). Prístupy procesu, ktorý nie je obmedzený žiadnym profilom sa riadia bežnými Linuxovými prístupovými oprávneniami (*discretionary access control*, *DAC*) zvolenými majiteľom súboru. Tieto oprávnenia sa vyhodnocujú aj v prípade, že na pristupujúcu aplikáciu sa vzťahuje obmedzujúci profil. *AppArmor* profil môže prístup povolený *DAC* zakázať, ale nemôže zakázaný prístup povoliť.

Ako už bolo naznačené, jednou zo základných úloh *AppArmoru* je obmedziť prístup k súborom. Na toto nie je potrebné pridávať žiadne značky do súborového systému. *AppArmor* vyhodnocuje prístupy na základe ciest uvedených v profile procesu.

1.2 Výhody a nevýhody

Výhody: Vytváranie profilov je relatívne jednoduché (napríklad v porovnaní so *SELinux*om[1]). Pre jednoduché profily stačí vytvoriť jeden súbor podľa prehľadnej syntaxe. Zložitejšie profily je možné zostaviť importovaním súborov, alebo už hotových abstrakcií. Nie je nutné vytvárať profil pre každú aplikáciu. Pre používanie *AppArmor*u nie je vyžadovaná podpora od súborového systému[2].

Nevýhody: V prípade, že pre nejakú aplikáciu nevytvoríme profil, zostane táto aplikácia neovplyvnená *AppArmor*om. To môže mať negatívny dopad na bezpečnosť systému, aj keď stále platí, že bežne používané linuxové prístupové oprávnenia ostávajú v platnosti. Absencia možnosti použitia značkovania objektov v súborovom systéme spôsobuje, že chýba priamočiary spôsob určovania bezpečnostných požiadaviek na daný objekt.

AppArmor je pomerne nová technológia, takže rôzne rady a návody chýbajú. Oficiálna dokumentácia pôsobí nedokončene.

1.3 Konfiguračné súbory

Konfigurácia *AppArmor*u je daná textovými súbormi, štandardne umiestnenými v priečinku `/etc/apparmor.d/`. Textové súbory sú, pre administrátora transparentne, prevedené do binárnej podoby, ktorá je následne načítaná do kernelu.

1.3.1 Syntax

Profily sa začínajú názvom profilu. Názov môže byť cesta k spustiteľnému súboru, na ktorý sa má profil aplikovať, v absolútnom tvare. Táto cesta môže obsahovať aj špeciálne znaky vyhovujúce viacerým súborom (*wildcard*). Tieto znaky sú nasledovné:

- „*”: Zodpovedá reťazcu znakov na jednej úrovni cesty (teda medzi dvoma „/”, alebo medzi „/” a koncom reťazca cesty). Môže zodpovedať aj prázdnemu reťazcu, avšak iba v prípade, ak tým v ceste nevzniknú dva oddeľovače „/” za sebou. Tomuto znaku vyhovujú aj súbory, ktorých meno začína bodkou (okrem špeciálnych súborov „.” a „..”).

[1]Pre viac informácií o *SELinuxe* odporúčame navštíviť zdroje uvádzané na oficiálnej stránke projektu [17].

[2]Napríklad *SELinux používa značky uložené v súborovom systéme.*

- „`**`": Podobne ako pre „`*`", avšak môže zodpovedať aj viacerým úrovniam priečinkov.
- „`?`": Zodpovedá jednému znaku rôznemu od „`/`".
- množinové zátvorky: Obsahujú zoznam možností oddelených čiarkou.
- hranaté zátvorky: Popisujú triedu znakov. Znak „`^`" umožňuje definovať zakázané znaky.

V prípade, že požadovaná cesta k súboru má obsahovať niektorý zo špeciálnych znakov, je nutné pred tento dať znak „\". Znaky je možné vyjadrovať aj v tvare *číslo*, kde *číslo* popisuje kód daného znaku.

Telo profilu je ohraničené množinovými zátvorkami. Konfiguračný súbor môže obsahovať ignorované riadky - komentáre, začínajúce sa znakom „`#`". (Znak „`#`" má pri zahŕňaní súborov pomocou reťazca „`#include`" špeciálny význam. V tomto prípade sa nejedná o komentár.)

1.3.2 Pravidlá pre súbory

Pravidlá pre súbory umožňujú procesu povoliť rôzne typy prístupu k súborom (pričom prístup, ktorý nie je povolený je implicitne zakázaný). Pravidlo pozostáva z úplnej cesty (začínajúcej znakom „`/`"), povolení pre súbory a čiarky („`,`") indikujúcej koniec pravidla. Možné je použiť nasledovné povolenia:

- „**r**": povolenie čítať súbor
- „**w**": povolenie na zápis
- „**x**": povolenie spustiť súbor (pred znakom `x` musí byť jedno z dostupných upresnení)
  - **ix**: nový proces bude bežať pod aktuálnym profilom
  - **px**: nový proces bude bežať pod profilom podľa cesty spustiteľného súboru
  - **ux**: nový proces nebude obmedzený *AppArmorom*
  - **cx**: nový proces bude bežať pod vnoreným profilom (*child profile*) podľa cesty spustiteľného súboru
  - *ďalšie možnosti*: Medzi upresnenie a znak `x` je možné vložiť záložné upresnenie (*fallback*). Napríklad z `px` je možné vytvoriť `pix`. V prípade,

že by neexistoval profil pre spúšťaný program, `px` by spustenie zablokovalo, zatiaľ čo `pix` nový program spustí ako pri možnosti `ix`. Upresnenia (okrem záložných) je možné napísať veľkým písmenom, vtedy sa zahodí prostredie (odporúča sa). Rozpísanie všetkých kombinácií možností je možné nájsť v návode [16].

- „**m**": povolenie mapovať súbor do pamäte
- „**k**": povolenie na uzamknutie súboru (musí byť v kombinácií s `r` alebo `w`)
- „**l**": povolenie vytvárať linku na súbor (*hardlink* s danou cestou); nie je takto možné získať prístup navyše - povolenia udelené na vytvorenú linku (ignorujúc `l`) musia byť podmnožinou povolení cieľa

1.3.3 Pravidlá pre prístup k sieti

Pravidlá pre prístup k sieti môžu prístupy k sieti buď povoľovať alebo zakazovať. Pravidlá zväčša začínajú slovom `network` a končia čiarkou. Pred pravidlom môže byť modifikátor (*rule qualifier*), ktorý môže pravidlo upresniť. Napríklad modifikátor `deny` zakazuje vykonanie popísanej sieťovej operácie. Pravidlo, ktoré nemá konkretizovaný niektorý parameter sa vzťahuje na všetky možnosti daného parametra. Je na autorovi profilu, aby použil konkrétne povoľovacie pravidlo, alebo aby následne použil modifikátor `deny` na obmedzenie pravidla. Ak napríklad celé pravidlo pozostáva iba z reťazca `network,` bez akejkoľvek ďalšej špecifikácie, toto pravidlo procesu povolí akúkoľvek sieťovú komunikáciu. Následne môžeme napríklad pridaním pravidla `deny` network tcp, zakázať komunikáciu prostredníctvom protokolu *TCP* (viď. upresnenia popísané nižšie.) Za slovom `network` môžu nasledovať upresnenia nasledovných typov:

- **permissions**
  - `create`: vytvorenie socketu
  - `shutdown`: ukončenie socketu
  - `listen`: počúvanie na sockete
  - `bind`: vykonanie operácie *bind* na danú zdrojovú adresu (v prípade neuvedenia, na akúkoľvek adresu)
  - `connect`: pripojenie sa na uvedenú cieľovú adresu (v prípade neuvedenia, na akúkoľvek adresu)
  - `accept`: prijatie spojenia zo zadanej adresy (v prípade neuvedenia, z akejkoľvek adresy)

  - `read/receive`: prijatie paketu z cieľovej adresy na zdrojovej adrese
  - `write/send`: odoslanie paketu zo zdrojovej adresy na cieľovú
  - `getname`: získanie mena alebo adresy socketu na lokálnej strane
  - `getpeer`: získanie mena alebo adresy socketu na vzdialenej strane
  - `setopt`: nastavenie parametrov socketu

- **domain**: typ domény, napríklad `inet`, `bluetooth`, ...
  - Typ domény môže byť v niektorých prípadoch vynechaný. Napríklad, v prípade že chceme povoliť *TCP spojenia*, je samozrejme nutné mať príslušné povolenie na prístup k sieti (napríklad `network inet stream,`), avšak toto povolenie nie je nutné písať explicitne v prípade, že už máme použité pravidlo `network tcp,`.
- **type**: dodatočný typ spojenia (napríklad pre doménu `inet` existujú známe `stream`, `dgram`, ...) (V prípade neuvedenia typu sa pravidlo bude vzťahovať na všetky typy implikované protokolom a spôsobom adresovania.)
- **protocol**: komunikačný protokol, napríklad `tcp`, `udp`, `icmp`
  - Niektoré protokoly môžu priamo implikovať doménu alebo typ. V takom prípade nie je nutné písať pravidlo pre danú doménu alebo typ.
  - Za uvedeným protokolom je možné špecifikovať zdrojovú alebo cieľovú adresu pre protokol.

1.3.4 Capabilities

AppArmor umožňuje použitie bežného Linuxového systému *Capabilities*. (Podobne, ako pri ostatných pravidlách, *AppArmor* limituje *capabilities*, ktoré nie sú v profile spomenuté, ale neprideľuje tie ktoré sú povolené.) Jednotlivé *capabilities* sa udeľujú v tvare `capability názov,`, kde za slovom `capability` nasleduje názov príslušného povolenia. Informácie o jednotlivých povoleniach sa nachádzajú v manuálových stránkach systému *Capabilities* [5].

1.3.5 Pravidlá pre pripájanie súborových systémov

Pravidlá pre pripájanie súborových systémov umožňujú procesu pripojiť alebo odpojiť súborový systém. Začínajú jedným zo slov `mount`, `umount` a `remount`.

Najjednoduchší spôsob udelenia povolení pre pripojenie súborových systémov je uviesť pravidlo v podobe `mount ,`. Pravidlo, ktoré nemá uvedené žiadne ďalšie možnosti, sa vzťahuje na všetky typy súborových systémov a na ľubovoľné ďalšie nastavenia.

Zložitejšie pravidlo môže upresniť povolené *možnosti* pre pripojenie a *cestu* kam sa súborový systém môže pripojiť:

```
mount options=nastavenia /dev/foo -> /cesta/,
```

Povolené nastavenia sú v rovnakom tvare, ako v prípade parametra `-o` nástroja `mount`. Viacero nastavení je možné zadať v zátvorkách, oddelené čiarkami.

Ak nevyžadujeme, aby pripájanie súborového systému obsahovalo všetky uvedené nastavenia, namiesto `options=nastavenia` je potrebné uviesť `options in (nastavenie 1,nastavenie 2,...)`. V takomto prípade sa povolenie udelí, ak žiadané nastavenia sú podmnožinou nastavení uvedených v zátvorke.

V prípade uvedenia viacerých výrazov `options` pravidlo udelí povolenie v prípade, že požadované nastavenia vyhovujú aspoň jednému z týchto výrazov.

1.3.6 Pravidlá pre systémové volanie ptrace

Tieto pravidlá umožňujú procesu používať systémové volanie `ptrace` určené na ladenie programov. Bežné aplikácie toto volanie väčšinou nepotrebujú.

1.3.7 Pravidlá pre posielanie signálov

Tieto pravidlá umožňujú procesu posielať alebo prijímať signály. Manuálová stránka pre konfiguračný súbor *AppArmor*u [4] obsahuje nasledujúce príklady:

```
signal, # Povolí prístup ku všetkým signálom.

deny signal (send) set=(hup, int),
# Explicitne zakáže posielanie signálov HUP[3] a INT[4].

signal (receive) peer=unconfined,
# Povolí procesom, ktoré nie sú obmedzené AppArmorom (unconfined) aby ···
  posielali signály tejto aplikácií.

signal (send) peer=/usr/bin/foo,
# Povolí posielanie signálov procesu bežiacemu pod profilom /usr/bin/···
  foo
```

[3] Signal hang up: indikuje procesu odpojenie terminálu.
[4] Signal interrupt: používateľ žiada prerušenie procesu

```
signal (receive, send) set=("exists"),
# Povolí kontrolovanie či existuje PID procesu.

signal peer=@{profile_name}, # Povolí posielanie signálov sebe.
```

1.3.8 Pravidlá pre riadenie prístupu k mechanizmu DBus

Pravidlá umožňujúce komunikáciu medzi procesmi pomocou mechanizmu *DBus*. Podrobný popis pravidiel je možné nájsť v manuálových stránkach [4].

1.3.9 Premenné

V konfiguračnom súbore je možné používať premenné. Premennej je možné priradiť hodnotu v tvare `@{NÁZOV}=hodnota`. Priradenia je nutné vykonať na začiatku súboru, ešte pred začiatkom profilu. Jednej premennej je možné priradiť aj viacero hodnôt (oddelených medzerami). Následne je možné premennú používať, ako keby obsahovala len jednu hodnotu. Prekladač hodnoty automaticky rozbalí do viacerých pravidiel.

1.3.10 Aliasy

Aliasy slúžia na nahradenie časti cesty. Nahradenie sa vykonáva až po dosadení hodnoty premenných. Globálne *aliasy* sa nachádzajú v súbore `/etc/apparmor.d/tunables/alias`, ktorý je štandardne zahrnutý súborom `/etc/apparmor.d/tunables/global`, ktorý býva zvyčajne (podľa manuálových stránok [4]) zahrnutý na začiatku *AppArmor* profilu.

Príklad:

```
alias /usr/ -> /mnt/usr/,
```

Štandardne sú niektoré dáta na čítanie uložené v priečinku `/usr/`. Ak máme tieto dáta výnimočne pripojené do priečinku `/mnt/usr/`, bolo by nepraktické prepisovať všetky profily. Alias z príkladu spôsobí, že pri vyhodnocovaní sa cesta upraví automaticky. (Pôvodné pravidlá zostanú zachované.)

1.3.11 Kvalifikátory pravidiel (Rule Qualifiers)

- `audit`: Požiadavky na povolenia ktoré prislúchajú danému pravidlu budú zaznamenané do logu.

- `deny`: Požiadavky na povolenia ktoré prislúchajú danému pravidlu budú odmietnuté. Tento kvalifikátor je možné použiť spolu s kvalifikátorom `audit`.
- `owner`: Proces musí mať rovnaké *euid/fsuid* ako objekt, na ktorý sa vzťahuje kontrola oprávnení.

1.3.12 Zahŕňanie súborov (include)

Do súboru s profilom je možné zahrnúť už existujúce súbory. Slúži na to kľúčové slovo `include`. (Možné je písať aj `#include`, avšak podľa dokumentácie [16] by to v nových profiloch nemalo byť používané.) Do úvodzoviek je možné dať relatívnu cestu k súboru (relatívne voči aktuálnemu súboru), medzi symboly `<` a `>` cestu vzhľadom na definovaný priečinok s konfiguračnými súbormi (`/etc/apparmor.d`).

Príklad:

```
include <abstractions/dconf>
include "mojeprofily/subor"
```

1.3.13 Vnorené profily (*child profiles*)

Vnútri profilu je možné definovať vnorený profil (podľa rovnakej syntaxe ako hlavný profil - názov profilu a telo profilu v množinových zátvorkách). Následne, programy spustené vďaka povoleniu `cx` (alebo obdobnému) budú spustené pod príslušným vnoreným profilom (podľa cesty k príslušnému binárnemu súboru). Ako názov profilu je možné namiesto cesty súboru uviesť meno profilu. Následne je možné explicitne definovať, že daný program má byť spustený pod daným profilom.

Príklad:

```
/cesta/k/programom/* cx -> menoprofilu,
/bin/* cux,


/bin/bash {
        /** rwixkm,
        # ...
}

menoprofilu {
        # ...
}
```

1.4 Rozšírenosť a existujúce nasadenia

V Linuxovej distribúcii *Ubuntu* (a odvodených distribúciách ako *Linux Mint*) je *AppArmor* štandardne obsiahnutý a zapnutý. Niektoré konkrétne aplikácie majú vytvorený profil. Po otestovaní existujúceho profilu pre prehliadač *Firefox* (`/etc/apparmor.d/usr.bin.firefox`) sa ukázalo, že pri zapnutí daného profilu niektoré funkcie nefungujú. Napríklad dialóg na otvorenie stiahnutého súboru neponúkal na výber aplikácie. Dostupných je viacero abstrakcií (niektoré sú použité aj v kapitole 3) a rôzne prednastavené premenné (priečinok `/etc/apparmor.d/tunables/`). Za povšimnutie stojí profil `sanitized_helper` v súbore `/etc/apparmor.d/abstractions/ubuntu-helpers`, ktorý slúži ako náhrada použitia neohraničeného (*unconfined*) spustenia. Ošetruje prostredie. Profily, ktorých názvy začínajú na „`ubuntu-`" následne aplikujú tento profil na niektoré aplikácie, ktoré nemajú vytvorený cielený profil.

AppArmor je obsiahnutý v Linuxovom kerneli od verzie *2.6.36*[9]. Spomedzi Linuxových distribúcií je ďalej *AppArmor* štandardne podporovaný napríklad v distribúciách *Debian* [11], *SUSE Linux Enterprise server* a *openSUSE* [12].

Kapitola 2

Rámec na zvýšenie bezpečnosti počítača

V tejto kapitole sa budeme venovať konkrétnemu návrhu rámca na zabezpečenie OS Linux.

2.1 Ciele

Cieľom je zjednodušiť dostatočne skúsenému používateľovi uvažovanie o jednotlivých aplikáciách a ich bezpečnostných dopadoch, a umožniť aplikovať technológiu *AppArmor* v praxi. Poznatky z tejto práce by mali byť aplikovateľné aj všeobecne, na širšie spektrum používateľských aplikácií, nie len na konkrétne príklady. Práca by mala do istej miery sformalizovať spôsob uvažovania o vytváraní profilov na zvýšenie bezpečnosti počítača. Keďže v súčastnosti majú aplikácie špecifické potreby, ktoré nie sú dostatočne štandardizované, nie je možné vyhnúť sa skúmaniu konkrétnych reprezentantov kategórie aplikácií.

Na základe budúcich potrieb by malo byť možné vytvorený model rozšíriť. Táto práca následne môže poskytnúť základ pre ďalšiu štandardizáciu, ktorá by umožnila vývojárom aplikácií špecifikovať požiadavky ich aplikácie v prehľadnej, používateľsky pochopiteľnej, forme.

2.2 Grafový model

Našim cieľom bude popísať, čo aplikácia potrebuje a zakázať všetko ostatné. Pre popis požiadaviek aplikácie pri očakávateľných činnostiach a následné zjednodušenie uvažovania o bezpečnostných dopadoch a implementácií riešenia prostredníctvom technológie *AppArmor*, navrhujeme použitie orientovaného grafu. Tento graf

môže obsahovať štyri rôzne typy vrcholov.

2.2.1 Typy vrcholov

- **požiadavka** (P-vrchol): Vrchol popisuje abstraktnú alebo konkrétnu požiadavku aplikácie pri bežnej činnosti. Pojem *požiadavka* a príslušný typ vrcholu budeme (okrem bežného významu) používať aj na označenie definovanej časti funkcionality aplikácie (napríklad tlač, čítanie dokumentov, významovo odlíšená skupina viacerých operácií, ...). Keď to bude vhodné, budeme používať pojem *povolenie*. (Aplikácia potrebuje povolenia na vykonávanie rôznych akcií práve na to, aby mohla poskytovať funkcionalitu. Povolenie *napĺňa* požiadavku.) V špecifických prípadoch (keď do vrcholu vedie jediná hrana a to z *D-vrcholu*) je možné *P-vrchol* použiť aj na reprezentáciu špecifického komponentu (napríklad modul v systéme, používateľské prostredie, metóda vstupu, ...).

  Abstraktné požiadavky majú význam pre človeka, nie sú priamo závislé na konkrétnom operačnom systéme a jeho súčastiach. (Príkladom takejto požiadavky môže byť tlač dokumentov. Aj pre bežného používateľa je význam zrejmý, avšak takáto požiadavka nič nehovorí o tom ako je tlač implementovaná v konkrétnom systéme.) *Konkrétne požiadavky* špecifikujú potrebu prístupu ku konkrétnym službám poskytovaným systémom (napríklad prístup k súboru, komunikácia prostredníctvom sieťového protokolu a podobne). Z hľadiska typov vrcholov ich nebudeme odlišovať.

  Požiadavky by nemali byť závislé od konkrétneho bezpečnostného modulu (AppArmor). Tento typ vrcholov do istej miery reprezentuje očakávané správanie aplikácie. Pre každý takýto vrchol by mala byť možná diskusia o potrebe priradenia daného vrcholu danej aplikácii a o prípadných spôsobených bezpečnostných rizikách.

- **kategória aplikácie** (K-vrchol): Tento druh vrcholu popisuje typ aplikácie. Ide o intuitívne rozdelenie pochopiteľné aj bežným používateľom. Rozdelenie aplikácií do kategórií nie je objektívne dané, slúži hlavne praktickým účelom. Kategória by mala byť dostatočne všeobecná, aby zahŕňala čo najväčšie množstvo aplikácií. V ideálnom stave by bolo možné každú používateľskú aplikáciu zaradiť do niektorej kategórie. Zároveň by kategória mala byť dostatočne konkrétna, aby bolo možné určiť dostatočne malú skupinu činností (požiadaviek), ktoré sa od aplikácie v danej kategórií dajú očakávať. Tieto

požiadavky aj samé o sebe do istej miery kategóriu definujú, avšak pre vysvetlenie sémantického významu používateľovi (prípadne osobe rozširujúcej graf) je dôležitá príslušná dokumentácia.

- **implementácia pomocou *AppArmor*u** (A-vrchol): Vrchol popisujúci časť konfiguračného súboru pre profil aplikácie. Napĺňajú sa ním požiadavky aplikácie. Súčasťou vrcholu môže byť konkrétny reťazec určený na vloženie do konfiguračného súboru, alebo skrátený zápis zahrnutia súboru v preddefinovanom umiestnení v tvare `<abstractions/názov>`, kde *názov* popisuje konkrétny súbor. Takto zadefinovaná abstrakcia sa vo finálnom konfiguračnom súbore profilu preloží do tvaru `include <abstractions/názov>`.

- **možnosti** (D-vrchol): Vrchol reprezentujúci očakávanie vstupu od používateľa. Cieľový profil môže závisieť od konkrétnych vlastností systému (nainštalované aplikácie a moduly, konfigurácia systému) a od požiadaviek používateľa pri konkrétnom scenári použitia. Vrchol obsahuje človekom pochopiteľný názov alebo stručný popis rozhodnutia čo sa má vykonať.

2.2.2 Hrany

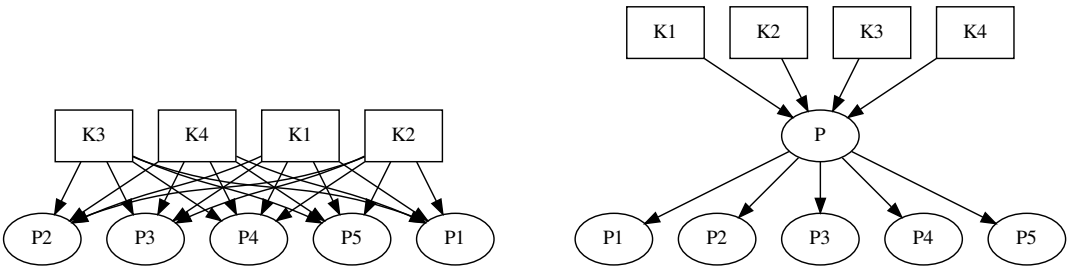
Principiálne je možné ktorékoľvek dva vrcholy prepojiť orientovanou hranou. Osoba vytvárajúca graf je zodpovedná za to, aby hrany dávali zmysel. Nižšie uvádzame, na čo slúžia hrany medzi danými typmi vrcholov. (Uvedené sú len zmysluplné typy hrán.)

- K-vrchol → K-vrchol : Hrana reprezentujúca príslušnosť kategórie aplikácie do nad-kategórie. Kategória môže mať viacero podkategórií. Pre prehľadnosť by mali byť jednotlivé kategórie aplikácií definované tak, aby nemusela nejaká kategória patriť pod viacero nad-kategórií (avšak ak také rozdelenie nie je možné, nič nebráni priradeniu konkrétnej kategórie pod viaceré nad-kategórie).

- K-vrchol → P-vrchol : Hrana priraďuje požiadavku danej kategórii aplikácií. Rôzne kategórie môžu zdieľať požiadavku, čo uľahčuje pridávanie ďalších kategórií[1].

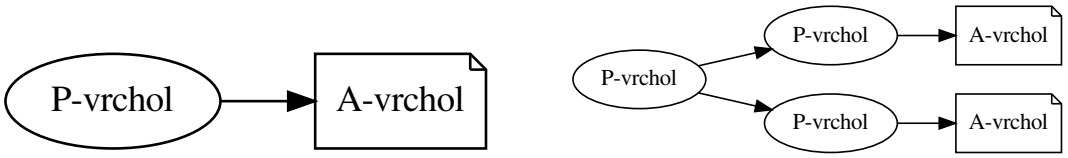
- P-vrchol → P-vrchol : Požiadavka sa môže skladať z viacerých menších požiadaviek. Pre rozumne definovateľnú, často používanú skupinu požiadaviek by mal existovať vrchol reprezentujúci túto skupinu, z ktorého hrany vedú do jednotlivých menších požiadaviek (obrázok 2.1). Ak do takto

[1]Nie je nutné vytvárať pre každú kategóriu *P-vrcholy* a všetky ďalšie dosiahnuteľné vrcholy zvlášť.



(a) Pred (b) Po

Obr. 2.1: Spojenie P-vrcholov



(a) Pred (b) Po

Obr. 2.2: Rozdelenie A-vrcholu

novo-vytvoreného vrcholu vedie i hrán, pričom z neho vedie o hrán, takouto operáciou sa ušetrilo $i \cdot o - i - o$ hrán, čo zvýši prehľadnosť grafu.

- (P-vrchol) → [A-vrchol] : Hrana popisuje časť implementácie požiadavky prostredníctvom *AppArmor*u. Z jedného *P-vrcholu* môžu vychádzať hrany do viacerých *A-vrcholov*. Bežne by z jedného *P-vrcholu* mala vychádzať nanajvýš jedna hrana do *A-vrcholu* ktorý nepopisuje abstrakciu. Ak nie je potrebné zdieľať časť riadkov viacerými *P-vrcholmi*, tak je možné text z viacerých *A-vrcholov* spojiť do jedného. Ak je potrebné zdieľať len niektoré riadky *A-vrcholu*, je vhodné rozdeliť *P-vrchol* aj *A-vrchol* na dva tak, ako je znázornené na obrázku 2.2. Novo vytvorené *P-vrcholy* budú zastrešovať sémantický význam textu v *A-vrcholoch*, a je ich teda možné neskôr zdieľať. Zároveň bude naďalej platiť, že do *A-vrcholu* bude viesť len jedna hrana[2].

- [K-vrchol] / (P-vrchol) → ◁D-vrchol▷ : Hrana pre konkrétnu kategóriu alebo požiadavku popisuje nutnosť opýtať sa používateľa alebo administrátora na požadovaný výber z možností. Z jedného *K/P-vrcholu* môže viesť viacero hrán do *D-vrcholov*. Rozdelenie možností do viacerých *D-vrcholov* má najmä sémantický význam - používateľ alebo administrátor vykonávajúci konfiguráciu aj tak vyberie podmnožinu možností zo zjednotenia možností pod oboma *D-vrcholmi*. Do jedného *D-vrcholu* by mala vstupovať len jedna hrana. Jediná hrana z *K-vrcholu* popisuje situáciu, keď je pri

[2] *K-vrchol* má len jeden význam, ktorý musí popísať nejaký *P-vrchol*. Nedáva zmysel, aby dva rôzne *P-vrcholy* popisovali ten istý význam

konfigurácii vyžadovaný výber požiadaviek pre kategóriu. Hrana z *P-vrcholu* spája požiadavku a rôzne možnosti pre naplnenie danej požiadavky. Možnosti pre naplnenie požiadavky dávajú zmysel iba v prípade, ak existuje práve jedna príslušná požiadavka (ako rodič *D-vrcholu*).

- 

: Hrana priraďuje konkrétnej „otázke"[3] (*D-vrchol*) jednotlivé možnosti odpovede (*P-vrcholy*). Odpovede je stále možné chápať v súlade s definíciou *P-vrcholu* v časti 2.2.1. V prípade, že z *D-vrcholu* vedie len jedna hrana, ide o výber typu *áno/nie*. Z *D-vrcholu* môže viesť viacero hrán, v takomto prípade ide o *výber možností* (osoba vykonávajúca konfiguráciu vyberá podmnožinu z vrcholov do ktorých vedie hrana z *D-vrcholu*).

2.3 Formát súboru na popis grafu

V tejto časti zadefinujeme jazyk na popis konkrétneho modelu. Jazyk je navrhnutý tak, aby bolo možné graf podľa časti 2.2 zakomponovať do zdrojového kódu vo formáte `.dot` pre nástroj na kreslenie grafov *Graphviz* [8]. V praxi bude teda možné vytvoriť súbor, z ktorého je možné prostredníctvom nástroja *Graphviz* vygenerovať obrazovú reprezentáciu grafu a prostredníctvom konfigurátora popísaného v kapitole 4 vytvoriť profil pre vybranú aplikáciu.

Súbor popisujúci graf pozostáva z riadkov. Riadok môže, ale nemusí, obsahovať špeciálne reťazce s významom v rámci modelovania grafu. (Text, ktorý nemá priradený žiaden význam, môže byť pre účely reprezentácie grafu definovanej v kapitole 2.2 považovaný za komentár. Tento text však môže mať význam napríklad pre vytvorenie obrazovej reprezentácie grafu pomocou nástroja *Graphviz*.)

Reťazce s definovaným významom

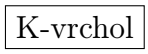
Reťazce v tejto kapitole môžu, ale nemusia obsahovať medzery ohraničujúce oddeľovače „`=`", „`->`", „`,`".

- ***názov*** `[label = "`***popis***`"]`: Definícia vrcholu s názvom „*názov*" a obsahom vrcholu „*popis*". Špeciálne reťazce „\l" a „\n" umožňujú vložiť koniec riadku. (Reťazec „**\l**" slúži na zarovnanie riadku doľava v grafickej reprezentácii grafu.)

[3] Slovo *otázka* je v tomto prípade v úvodzovkách, keďže vrchol nemusí nutne obsahovať opytovaciu vetu.

- *`zdrojový vrchol -> cieľové vrcholy`*: Definícia hrany. V prípade, že zo zdrojového vrcholu vedie hrana do viacerých cieľových vrcholov je možné uviesť viacero cieľových vrcholov oddelených čiarkami.

- `shape = typ vrcholu`: Reťazec popisuje typ vrcholu na danom riadku a na všetkých nasledujúcich riadkoch (až po ďalší príkaz `shape`, alebo po koniec bloku). *Typ vrcholu*:

  - `rectangle`: 

  - `ellipse`: 

  - `diamond`: 

  - `note`: 

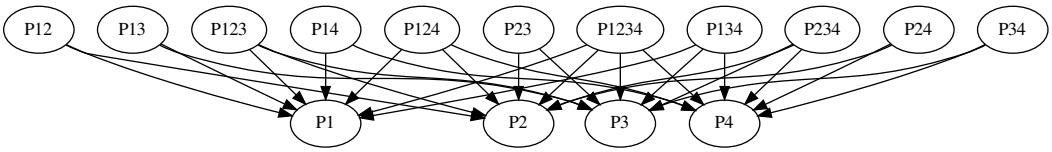
- Ak riadok obsahuje znak „{" (a zároveň sa pred ním nenachádza nepárny počet úvodzoviek „`"`"), daný riadok označuje začiatok bloku. Podobne, riadok obsahujúci znak „}" označuje koniec bloku. Ak riadok popisuje začiatok alebo koniec bloku, celý zvyšný obsah riadku sa ignoruje. Na začiatku bloku sa na zásobník uloží typ vrcholu nastavený príkazom `shape` alebo vybratím zo spomenutého zásobníku. Na konci bloku sa vyberie typ vrcholu zo zásobníka, a začne sa aplikovať na nasledujúce vrcholy (ako keby bol nastavený príkazom `shape`). Blok teda slúži na priradenie jedného typu viacerým vrcholom bez nutnosti opakovane písať `shape`.

2.4 Výhody a problémy

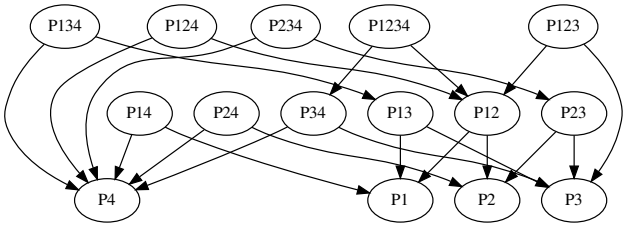
Graf umožňuje v prehľadnej forme reprezentovať znalosti o aplikačných programoch. Formát súboru napĺňa dva hlavné účely: umožňuje vytvorenie obrazovej reprezentácie pomocou nástroja *Graphviz* a umožňuje vytvorenie profilu pre aplikáciu pomocou konfigurátora z tejto práce. Obrazová reprezentácia grafu umožňuje používateľovi a osobe vykonávajúcej konfiguráciu získať prehľad o požiadavkách a ich vzájomných prepojeniach. Prítomnosť *A-vrcholov* zas umožňuje konfigurátoru vytvoriť profil pre aplikáciu na základe toho istého modelu, o ktorom majú ľudia prehľad, a je možné o ňom viesť diskusiu a analyzovať ho.

Takáto reprezentácia môže byť mimoriadne efektívna (v zmysle, že počet vrcholov môže byť rovnaký ako počet abstraktných konceptov ktoré chceme popísať a počet hrán môže byť nižší ako počet prepojení medzi nimi[4]). V praxi sa pri spre-

[4]V prípade, že použijeme spájanie vrcholov aké bolo spomenuté pri popise $P \rightarrow A$ hrán (viď. obrázok 2.2).



(a) Triviálne prepojenie nových vrcholov s pôvodnými



(b) Jedno z možných minimálnych priradení hrán (vzhľadom na počet hrán). Každý vrchol používa práve dva vrcholy reprezentujúce menší počet pôvodných vrcholov.

Obr. 2.3: Spájanie P-vrcholov prostredníctvom nových vrcholov

hľadnení grafu a spájaní vrcholov môžu pridať vrcholy nad rámec nevyhnutného počtu.

Problém pri spájaní skupín vrcholov prostredníctvom novovytvorených vrcholov je, že v najhoršom prípade je možné takto nad množinou n vrcholov vytvoriť až $2^n - n - 1$ nových vrcholov (veľkosť potenčnej množiny bez prázdnych a jednoprvkových množín). Z každého nového vrcholu vedú najmenej dve hrany. Na obrázku 2.3 je znázornené takéto spájanie vrcholov pre štyri vrcholy. Pôvodné vrcholy obsahujú práve jednu číslicu, číslice v nových vrcholoch zodpovedajú číslam pôvodných vrcholov ktoré reprezentuje daný vrchol). Riešením tohto problému je nevytvárať nové vrcholy pre všetky možné kombinácie vrcholov, ale len také, do ktorých povedie nejaká hrana. V prípade, že by takýchto vrcholov bolo priveľa (s ohľadom na prehľadnosť grafu), je nutné nájsť kompromis medzi počtom hrán a počtom vrcholov. V praxi pridanie nového vrcholu ovplyvní aj to, či daný vrchol bude reprezentovať rozumné zoskupenie požiadaviek podľa reálneho sveta.

Kapitola 3

Konkrétny graf a definície vrcholov

V tejto kapitole spomenieme konkrétnu inštanciu grafu podľa časti 2.2. Prezentovaná inštancia grafu bola navrhnutá pre operačný systém *Linux Mint 19*. Poznatky z tejto práce a konkrétny graf by sa mali dať bez väčších problémov preniesť na ďalšie distribúcie Linuxu, avšak nie je to možné zaručiť.

Jednotlivé odseky popisujú význam vrcholov grafu. Hlavička odseku obsahuje *označenie vrcholu*, *meno vrcholu*, *vstupné hrany* a *výstupné hrany*[1]. To, aké vrcholy sú priradené pod daný vrchol môže byť odôvodnené nasledovnými spôsobmi:

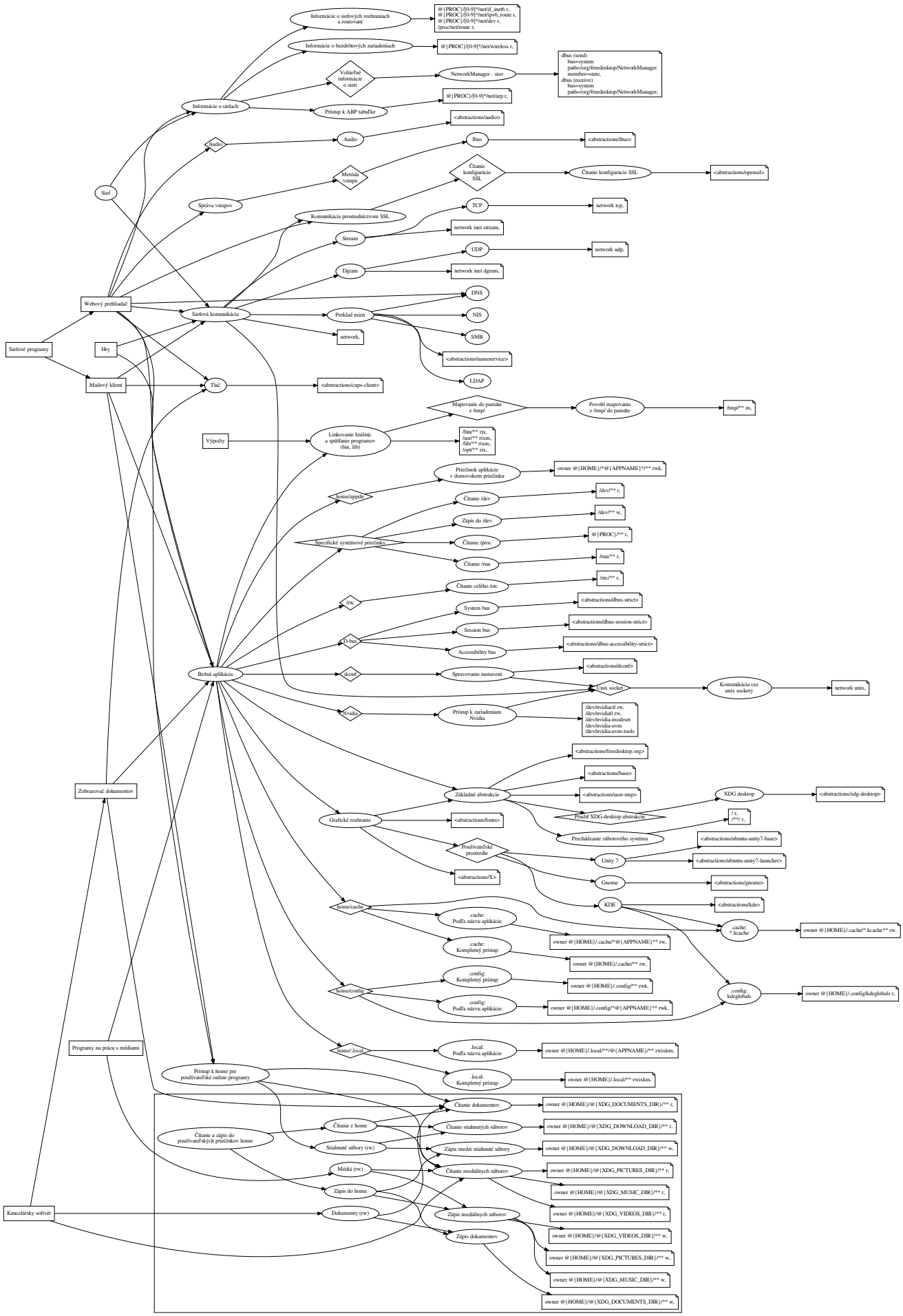
- Vyplýva to priamo z popisu vrcholu.
- Pri danom vrchole je priamo uvedené vysvetlenie, prečo z neho vedú konkrétne hrany.
- Potomok vrcholu obsahuje odôvodnenie prečo je zaradený pod daný vrchol, prípadne to vyplýva z popisu potomka.
- Ide o $P \rightarrow A$ hranu, alebo o hranu vychádzajúcu z *D-vrcholu*. Pri týchto hranách je ich význam triviálne zrejmý.

Obrazovú reprezentáciu grafu je možné nájsť na obrázku 3.1.

3.1 P-vrcholy

System bus (`dbus_strict`)**:** **in**: `opt_dbus`; **out**: `aa_dbus_strict`;

[1]Hrany sú označené menom vrcholu na druhom konci hrany. V elektronickej verzii textu obsahujú odkaz na daný vrchol



Obr. 3.1: Obrazová reprezentácia grafu

Obmedzené povolenie prístupu k *system bus* systému *Dbus*. Tento *bus* slúži na komunikáciu medzi procesmi v rámci systému. Môže slúžiť aj na komunikáciu medzi procesmi rôznych používateľov. Povolenie sa vzťahuje len na prístup k samotnému *busu* na to určeným spôsobom.

Accessibility bus (`dbus_accessibility_strict`)**: in**: `opt_dbus`; **out**: `aa_dbus_accessibility_strict`;

Obmedzené povolenie prístupu k *accessibility bus* systému *Dbus*. Povolenie sa vzťahuje len na prístup k samotnému *busu* na to určeným spôsobom. Samotná abstrakcia ktorá implementuje tento vrchol je podľa obsahu súboru vytvorená firmou *Canonical*, avšak podrobnejšia dokumentácia k danému *busu* nie je v systéme obsiahnutá.

Session bus (`dbus_session_strict`)**: in**: `opt_dbus`; **out**: `aa_dbus_session_strict`;

Obmedzené povolenie prístupu k *session bus* systému *Dbus*. Tento *bus* slúži na komunikáciu medzi procesmi v rámci jedného sedenia používateľa. Povolenie sa vzťahuje hlavne na prístup k samotnému *busu*.

Sieť (`network`)**: in**: –; **out**: `network_com`; `info_network`;

Zastrešenie všetkých, v grafe obsiahnutých, požiadaviek týkajúcich sa sietí. Nejde len o komunikáciu prostredníctvom sietí, ale aj o prípadné požiadavky nastavovať a zisťovať informácie o sieťach. Z dôvodu príliš veľkej všeobecnosti by tento vrchol nemal byť pri konfigurácií povolený. V nami popísanom grafe tento vrchol nie je dosiahnuteľný zo žiadneho *K-vrcholu* (čo znamená, že by osoba vykonávajúca konfiguráciu musela povoliť vrchol ručne). V prípade rozširovania grafu je odporúčané túto vlastnosť zachovať.

Sieťová komunikácia (`network_com`)**: in**: `game`; `web`; `mail`; `network`; **out**: `ssl`; `nameservice`; `stream`; `dgram`; `aa_network`; `opt_net_unix`;

Zastrešenie požiadaviek obsiahnutých v grafe týkajúcich sa komunikácie prostredníctvom sietí. Pri povolení sieťovej komunikácie je potrebné myslieť na to, že aplikácia môže prichádzať do kontaktu s dátami z nedôveryhodnej strany. Napríklad chyba pri implementácii použitého sieťového protokolu môže narušiť správanie aplikácie v prospech útočníka. Ďalej môže byť sieťová komunikácia zneužitá na vynášanie dôverných informácií.

V prípade, že existujú dôvody na presnejšie špecifikovanie povolenej komunikácie, je potrebné použiť niektorý z pod-vrcholov, nie tento vrchol. V prípade,

že požadovaný pod-vrchol neexistuje je možné pridať ho na základe špecifických potrieb.

Prístup k ARP tabuľke (`arp_table`): in: `info_network`; **out**: `aa_arp`;

Povolenie čítať systémovú *ARP* tabuľku. Umožňuje získať unikátne, výrobcom pridelené, hardvérové adresy. Tieto informácie môžu byť následne zneužité napríklad na identifikáciu konkrétneho zariadenia v ďalšej sieti. (Napríklad útočník pomocou tohto povolenia získa MAC adresu obete, a následne zistí, ktorý z používateľov pripojených na verejnú sieť, kontrolovanú útočníkom, je táto obeť.) Informácia o zariadeniach v sieti, s ktorými používateľský počítač komunikuje, môže byť pre útočníka zaujímavá aj z ďalších dôvodov.

Komunikácia prostredníctvom SSL (`ssl`): in: `web`; `network_com`; **out**: `opt_ssl_conf`;

Požiadavka na zabezpečenú komunikáciu prostredníctvom *SSL*. Táto komunikácia sa z pohľadu *AppArmoru* principiálne nelíši od inej aplikačnej komunikácie, avšak niektoré aplikácie môžu vyžadovať špeciálne povolenia, napríklad čítanie konfigurácie *SSL*.

Čítanie konfigurácie SSL (`read_openssl_conf`): in: `opt_ssl_conf`; **out**: `ab_openssl`;

Umožňuje čítať konfiguráciu *OpenSSL* - otvorenej implementácie *SSL* protokolu. Je možné, že daná konfigurácia sa na konkrétnom systéme vôbec nenachádza. Odporúčame toto povolenie neudeľovať v prípade, že to nie je explicitne požadované.

Preklad mien (`nameservice`): in: `network_com`; **out**: `dns`; `ldap`; `nis`; `smb`; `ab_nameservice`;

Povolenie prístupu k súborom pre vyhľadávanie pomocou DNS, LDAP, NIS, SMB, a vyhľadávania v databázach používateľov, skupín a hesiel, a vyhľadávanie protokolov a služieb. (Vrchol je vytvorený podľa abstrakcie *nameservice*.)

DNS (`dns`): in: `web`; `nameservice`; **out**: `-`;

Dotazy na systém *DNS* (*Domain Name System*). Vrchol slúži na zapuzdrenie časti požiadaviek rodičovského vrcholu.

NIS (`nis`): in: `nameservice`; **out**: `-`;

Dotazy na systém *NIS* (*Network Information Service*). Vrchol slúži na zapuzdrenie časti požiadaviek rodičovského vrcholu.

SMB (`smb`): **in**: `nameservice`; **out**: `-`;

Dotazy na systém *SMB* (*Server Message Block*). Vrchol slúži na zapuzdrenie časti požiadaviek rodičovského vrcholu.

LDAP (`ldap`): **in**: `nameservice`; **out**: `-`;

Dotazy na systém *LDAP* (*Lightweight Directory Access Protocol*). Vrchol slúži na zapuzdrenie časti požiadaviek rodičovského vrcholu.

Stream (`stream`): **in**: `network_com`; **out**: `tcp`; `aa_stream`;

Povolenie ľubovoľnej prúdovej internetovej komunikácie. Máme na mysli najmä protokol *TCP*. Ďalšie je možné dopĺňať podľa potreby.

Dgram (`dgram`): **in**: `network_com`; **out**: `udp`; `aa_dgram`;

Povolenie ľubovoľnej internetovej komunikácie prostredníctvom datagramov. Máme na mysli najmä protokol *UDP*. Ďalšie je možné dopĺňať podľa potreby.

TCP (`tcp`): **in**: `stream`; **out**: `aa_tcp`;

Povolenie komunikácie prostredníctvom protokolu *TCP*.

UDP (`udp`): **in**: `dgram`; **out**: `aa_udp`;

Povolenie komunikácie prostredníctvom protokolu *UDP*.

Informácie o sieťach (`info_network`): **in**: `web`; `network`; **out**: `info_inet_if`; `info_wireless`; `opt_info_network`; `arp_table`;

Povolenie prístupu k základným informáciám o sieťach. Niektoré aplikácie môžu potrebovať poznať stav pripojení a ďalšie informácie o sieťových rozhraniach a sieťach. Zložitejšia aplikácia napríklad môže meniť svoje správanie na základe toho, či komunikuje cez dátovo neobmedzené pripojenie, alebo cez spoplatnené pripojenie. Dostupnosť konkrétnych sieťových pripojení môže naznačovať približnú polohu zariadenia. Používateľské aplikácie na zobrazovanie stavu sietí taktiež môžu požadovať toto oprávnenie. Podrobnejšie informácie je možné nájsť v pod-vrcholoch.

Informácie o sieťových rozhraniach a routovaní (`info_inet_if`): **in**: `info_network`; **out**: `aa_info_net_if`;

Povolenie čítať informácie o sieťových rozhraniach. Pridelenie tohto povolenia aplikácií umožní zistiť ich názvy a sumárne štatistiky o prenesených dátach a špecifické informácie o smerovaní v IPv6 sieťach.

Informácie o bezdrôtových zariadeniach (`info_wireless`): in: `info_network`; **out**: `aa_proc_net_wireless`;

Povolenie čítať informácie špecifické pre bezdrôtové zariadenia, napríklad intenzitu signálu. Intenzita signálu môže, okrem iného, čiastočne odhaliť polohu zariadenia.

NetworkManager - stav (`nm_state`): in: `opt_info_network`; **out**: `aa_dbus_nm`;

Povolenie zistiť stav systému *NetworkManager*. Jedná sa o stav pripojenia k sieti (napríklad *odpojené*, *pripájanie*, *pripojené*...). Podrobnosti je možné nájsť na stránke projektu *Gnome* [10] v časti *enum NMState*.

Komunikácia cez unix sockety (`net_unix`): in: `opt_net_unix`; **out**: `aa_net_unix`;

Univerzálne povolenie pristupovať k *unixovým socketom*. V prípade, že nechceme alebo nevieme špecifikovať povolenie konkrétnejšie, môžeme použiť tento pred-pripravený vrchol.

Audio (`audio`): in: `opt_audio`; **out**: `ab_audio`;

Povolenie čítať a zapisovať do zvukových zariadení. Neželané zvuky môžu byť pre používateľa nepríjemné, prípadne až zmätočné (za predpokladu, že nie je jasné že daná aplikácia je zdrojom zvuku). Aplikácia napríklad môže napodobniť zvuk vydávaný používateľským prostredím, a prinútiť používateľa vykonať nejakú akciu, alebo predpokladať že sa vykonala akcia ktorá sa nevykonala. Nahrávanie zo zvukových zariadení môže byť zneužité na odpočúvanie.

Keďže možných typov zariadení je pomerne veľa, tento vrchol už ďalej nerozdeľujeme. Zjednoduší to implementáciu vrcholu prostredníctvom použitej abstrakcie v *A-vrchole*.

Správa vstupov (`input`): in: `web`; **out**: `opt_input`;

Povolenie spravovať metódy vstupu. Metódy vstupu riešia napríklad jazyk a správanie klávesnice. Jedná sa o pokročilú funkcionalitu, bežná aplikácia by takýto prístup nemala potrebovať. Konkrétne možnosti správy vstupov závisia od použitej metódy vstupu.

Ibus (`ibus`): in: `opt_input`; **out**: `ab_ibus`;

Povolenie čítať konfiguráciu metódy vstupu *Ibus* a komunikácie s týmto systémom.

Spravovanie nastavení (`settings`): **in**: `opt_dconf`; **out**: `ab_dconf`; `opt_net_unix`;

Povolenie čítať nastavenia uložené v systéme na globálnu správu a ukladanie nastavení-*dconf*. Ide o pomerne všeobecnú požiadavku. Povolenie na túto požiadavku je vhodné udeliť len v prípade, že daná aplikácia potrebuje čítať nastavenia väčšieho množstva používateľských alebo systémových aplikácií, prípadne ak ručné prideľovanie konkrétnejších oprávnení je nepraktické. V prípade udelenia povolenia sa treba ešte rozhodnúť, či povolíme komunikáciu pomocou *Unixových socketov* pomocou všeobecného vrcholu `net_unix`, prípadne ručne pridáme špecifické povolenie pre komunikáciu (napríklad podľa zakázaných prístupov v logu).

Bežná aplikácia (`basic_app`): **in**: `media`; `game`; `web`; `mail`; `reader`; **out**: `basic_abstractions`; `binlib`; `gui`; `opt_etc`; `opt_dbus`; `opt_config`; `opt_cache`; `opt_home_appdir`; `opt_dconf`; `opt_sysdir`; `opt_nvidia`; `opt_local`;

Jedná sa o jeden z najdôležitejších vrcholov. Sumarizuje často sa opakujúce požiadavky bežnej používateľskej aplikácie, od ktorých zároveň nie je očakávané, že by používateľom boli považované za špeciálnu vlastnosť aplikácie. Patria sem nasledovné požiadavky:

- Grafické rozhranie: Používateľské aplikácie často potrebujú vedieť vykresľovať okná, a pracovať s ďalšími grafickými prvkami.
- Spúšťanie programov a linkovanie knižníc: Aplikácia môže pozostávať s viacerých programov, napríklad spúšťač a samotná aplikácia. Pri Linuxových aplikáciach je bežné, že potrebujú dynamicky linkovať knižnice. Programy na prácu s médiami napríklad môžu potrebovať dodatočné doplnky na vytvorenie efektu, alebo spracovanie formátu súboru.
- Ďalšie požiadavky: Odôvodnenie sa nachádza v príslušných vrcholoch.

Čítanie celého /etc (`etc_all`): **in**: `opt_etc`; **out**: `aa_etc_all`;

Povolí aplikácií čítať obsah priečinku `/etc`. Problém je, že aplikácie často potrebujú prístup k rôznym konfiguračným súborom. Niektoré majú vytvorené vlastné konfiguračné súbory. Osoba vykonávajúca konfiguráciu sa môže rozhodnúť ručne pridať všetky potrebné súbory, avšak to môže byť v rozpore s cieľom zabezpečiť jednoduchú konfiguráciu. Preto existuje možnosť povoliť prístup na čítanie celého `/etc`. Pripomíname, že stále v platnosti ostávajú všetky obmedzenia ktoré v systéme boli pred vytvorením profilu (viď. časť 1.1), teda používateľ stále nemôže čítať čokoľvek. Dôsledkom pridania tohto vrcholu je teda iba to, že neobmedzíme prístup v prípade chybnej konfigurácie systému, prípadne neobmedzíme niektoré prístupy pre aplikáciu zbytočné.

Grafické rozhranie (`gui`)**: in**: `basic_app`; **out**: `opt_de`; `ab_fonts`; `ab_X`; `basic_abstractions`;

Požiadavka zobrazovať grafický obsah alebo pracovať s používateľským prostredím (*desktop environment*).

Prístup k zariadeniam Nvidia (`dev_nvidia`)**: in**: `opt_nvidia`; **out**: `opt_net_unix`; `aa_nvidia`;

Povoľuje prístup na čítanie a zápis k zariadeniam `/dev/nvidiactl` a `/dev/nvidia0`. Toto povolenie je potrebné udeliť iba v prípade, že je na počítači nainštalovaný proprietárny ovládač grafickej karty *Nvidia*. Pridaná je možnosť povoliť *unix sockety*, keďže sa ukázalo, že niektoré programy pri využívaní grafickej karty toto povolenie potrebujú.

Základné abstrakcie (`basic_abstractions`)**: in**: `gui`; `basic_app`; **out**: `ab_freedesktop`; `opt_xdgdesktop`; `ab_base`; `ab_tmp`; `ls_fs`;

Zoskupenie preddefinovaných abstrakcií a základných operácií. Zahŕňa prácu s priečinkom
`tmp` a prístup k nevyhnutným systémovým súborom a priečinkom. Použitie preddefinovaných abstrakcií zníži pravdepodobnosť, že nejaké potrebné povolenie zabudneme uviesť.

Prechádzanie súborového systému (`ls_fs`)**: in**: `basic_abstractions`; **out**: `aa_ls_fs`;

Povolenie čítať obsah a vlastnosti priečinkov. Nevzťahuje sa na čítanie obsahu súborov. Toto povolenie je vhodné prideliť vždy, jednotlivé ďalšie vrcholy následne nemusia popisovať právo na čítanie všetkých priečinkov v ceste. (Ak napríklad aplikácia pracujúca s dokumentmi dostane povolenie na čítanie obsahu priečinku `/home/peter/Documents`, nebude potrebné prácne priraďovať povolenie čítať `/home/peter`, `/home` a `/`. Užívateľ sa teda napríklad môže cez dialógové okno aplikácie určené na otvorenie súboru dostať až k súboru na ktorý má povolenie.)

Útočník sa za pomoci škodlivej aplikácie vie dostať k názvom súborov a priečinkov v systéme. Podstatná časť súborovej štruktúry je však verejne známa. Útočník môže zistiť, aký softvér (a konkrétne verzie) sú na danom systéme nainštalované. Názvy súborov a aplikácií môžu prezradiť čiastočnú informáciu o ich obsahu[2].

[2]Napríklad názov súboru `nedokoncena_diplomka.pdf` môže naznačovať, že používateľ ešte píše diplomovú prácu.

Linkovanie knižníc a spúšťanie programov (bin, lib) (`binlib`): in: `turing`; `basic_app`; **out**: `aa_bin_lib`; `opt_tmpm`;

Umožňuje prístup ku štandardným umiestneniam programov a knižníc, ich spúšťanie a linkovanie. Niektoré programy vyžadujú mapovanie dočasných súborov do pamäte. V prípade, že sa v systéme nachádzajú binárne súbory aj v nejakých neštandardných umiestneniach je možné pod tento vrchol pridať ďalšie *A-vrcholy*, ktoré pridajú jednotlivé povolenia.

Povoliť mapovanie z /tmp/ do pamäte (`tmpm`): in: `opt_tmpm`; **out**: `aa_tmp_m`;

Vrchol umožňuje mapovať súbory z priečinka `/tmp/` do pamäte. Tieto prístupy sú využívané aj používateľskými aplikáciami.

XDG desktop (`xdgdesktop`): in: `opt_xdgdesktop`; **out**: `ab_xdgdesktop`;

Vrchol umožňuje prístup na čítanie a zápis do špecifických priečinkov v používateľovom domovskom adresári určených na ukladanie aplikačných dát. (Tento prístup sa nevzťahuje na podpriečinky a súbory.) Štandardne sem patria priečinky `.config` (konfiguračné súbory) a `.cache` (dočasné údaje aplikácie) (viď. `opt_config` a `opt_cache`). Ďalej sem patrí priečinok `.local/share/`, ktorý by mal podľa dokumentácie [18] obsahovať dôležité dáta aplikácií. Môžu sem patriť alternatívy `/usr/share/` a `/usr/local/share/`. Toto povolenie sa nevzťahuje na súbory a podpriečinky.

.cache: Kompletný prístup (`cache_all`): in: `opt_cache`; **out**: `aa_cache_all`;

Povolenie prístupu k `@{HOME}/.cache/` na čítanie a zápis. Ak je to možné, odporúčame použiť konkrétnejší vrchol.

.cache: Podľa názvu aplikácie (`cache_appname`): in: `opt_cache`; **out**: `aa_cache_appname`;

Povolenie prístupu k `@{HOME}/.cache/` na čítanie a zápis, avšak len v prípade, že cesta obsahuje (používateľom definovaný) názov aplikácie. Nepovoľujeme teda prístup do priečinkov iných aplikácií, teda zvyšujeme mieru izolácie jednotlivých aplikácií.

.cache: *.kcache (`cache_kcache`): in: `gui_kde`; `opt_cache`; **out**: `aa_cache_kcache`;

Prístup na čítanie a zápis k súborom s príponou `.kcache` v priečinku `.cache` v domovskom adresári. Týka sa systémov s používateľským prostredím *KDE*.

.config: Kompletný prístup (`config_all`): in: `opt_config`; **out**: `aa_config_all`;

Povolenie prístupu k `@{HOME}/.config/` na čítanie a zápis. Ak je to možné, odporúčame použiť konkrétnejší vrchol. Povoľujeme aj uzamknutie súboru (**k**), keďže niektoré aplikácie používajú práve priečinok `config` na zablokovanie viacerých spustení.

.config: Podľa názvu aplikácie (`config_appname`): in: `opt_config`; **out**: `aa_config_appname`;

Povolenie prístupu k `@{HOME}/.config/` na čítanie a zápis, avšak len v prípade, že cesta obsahuje (používateľom definovaný) názov aplikácie. Nepovoľujeme teda prístup do priečinkov iných aplikácií, teda zvyšujeme mieru izolácie jednotlivých aplikácií. Povoľujeme aj uzamknutie súboru (**k**), keďže niektoré aplikácie používajú práve priečinok `config` na zablokovanie viacerých spustení.

.config: kdeglobals (`config_kdeglobals`): in: `gui_kde`; `opt_config`; **out**: `aa_config_kdeglobals`;

Prístup na čítanie k súboru `kdeglobals` v priečinku `.cache` v domovskom adresári. Týka sa systémov s používateľským prostredím *KDE*.

Priečinok aplikácie v domovskom priečinku (`home_appdir`): in: `opt_home_appdir`; **out**: `aa_home_appdir`;

Niektoré aplikácie (napríklad *Firefox*) sa nepridržiavajú špecifikácie [18], ale ukladajú svoje dáta a nastavenia priamo v priečinku nachádzajúcom sa v domovskom adresári. (Zvyčajne je tento priečinok nazvaný `.@{`*`APPNAME`*`}`, kde *@{APPNAME}* je názov aplikácie.) Toto povolenie teda prideľuje prístup na čítanie, zápis a zamykanie pre podpriečinky domovského adresára, ktoré v názve obsahujú názov aplikácie.

.local: Kompletný prístup (`local_all`): in: `opt_local`; **out**: `aa_local_all`;

Povolenie čítať, zapisovať a spúšťať súbory z priečinku `/.local/`. Odporúčame toto povolenie neprideľovať, ale (ak je to možné) použiť konkrétnejší vrchol `local_appname`. To tohto priečinku si ukladajú svoje dáta viaceré aplikácie, každá do svojho priečinku. Podľa špecifikácie *XDG Base Directory Specification* [18] spomína akurát, že ide o priečinok, ktorý obsahuje používateľsky špecifické dáta.

.local: Podľa názvu aplikácie (`local_appname`): in: `opt_local`; **out**: `aa_local_appname`;

Povolenie čítať, zapisovať a spúšťať súbory z priečinku `/.local/`, avšak iba v prípade, že sa v ceste nachádza priečinok s názvom aplikácie. Očakávame, že priečinky aplikácií budú v tvare `@{HOME}/.local/share/@{APPNAME}`, avšak môže sa stať, že niektorá aplikácia nepoužije pod-priečinok `share`[3].

Tlač (`print`): **in**: `web`; `mail`; `reader`; **out**: `ab_cups_client`;

Povolenie tlačiť na papier. Jedná sa o bežnú požiadavku kancelárskeho softvéru, avšak povolenie na túto požiadavku môže byť zneužité na spôsobenie materiálnych a ďalších škôd (minutie papiera a atramentu, zablokovanie služieb spoločnej tlačiarne vo firme, ...).

Domovský priečinok Vrcholy reprezentujúce požiadavku čítať a zapisovať do používateľského domovského priečinku (`/home`). Prideľovať kompletný prístup nie je vhodné, keďže tento priečinok môže obsahovať citlivé údaje. Prístup je možné rozdeliť podľa typu priečinku (stiahnuté súbory, médiá, dokumenty), alebo podľa typu prístupu (čítanie, zápis). Keďže sú zmysluplné oba spôsoby rozdelenia, spôsobuje to zvýšenie počtu vrcholov (dva vrcholy pre každý typ priečinku - čítanie a zápis, súhrnné vrcholy pre čítanie a zápis (pre každý typ), súhrnné vrcholy pre každý typ priečinku (čítanie aj zápis)).

Riešenie tohto problému by mohlo spočívať vo vytvorení špeciálneho typu vrcholu, ktorý by ponúkal na výber možnosti nastavenia, pričom hrana by vyberala ktoré nastavenia sa požadujú. Išlo by však len o zapuzdrenie problému, keďže v konečnom dôsledku aj tak každá možnosť musí byť implementovaná *A-vrcholom*. Komplexnosť grafového modelu by sa zbytočne zvýšila. Preto je tento ojedinelý problém vyriešený tak, že v obrazovej reprezentácií grafu sú tieto vrcholy orámované a zoskupené k sebe.

Ďalší problém je, že pri *AppArmor*e (na rozdiel od *SELinux*u) nevieme používať značkovanie súborov v súborovom systéme. Nie je možné teda jednoznačne rozlíšiť, pri ktorých súboroch by mal byť kladený vyšší dôraz na dôvernosť. Pre uloženie citlivých súborov navrhujeme vytvoriť samostatný priečinok s unikátnym názvom, ku ktorému sa v grafe nenachádza žiadne povolenie na prístup. Následne je pre špecifické aplikácie možné ručne pridať oprávnenia na prístup. Používateľ by mal byť oboznámený s významom jednotlivých priečinkov.

[3]Napríklad program *Mathics* nepoužíva priečinok `share` ale používa priečinok `var`.

Vymenované vrcholy týkajúce sa domovských priečinkov[a]:

Stiahnuté súbory (rw) (home_downloads_rw): in: home_online; **out**: home_downloads_r; home_downloads_w;
Dokumenty (rw) (home_documents_rw): in: office; **out**: home_documents_r; home_documents_w;
Čítanie dokumentov (home_documents_r): in: reader; home_online; home_all_r; home_documents_rw; **out**: aa_xdg_documents_r;
Čítanie z home (home_all_r): in: home_all_rw; **out**: home_documents_r; home_media_r; home_downloads_r;
Čítanie stiahnutých súborov (home_downloads_r): in: home_all_r; home_downloads_rw; **out**: aa_xdg_downloads_r;
Zápis mediálnych súborov (home_media_w): in: home_all_w; home_media_rw; **out**: aa_xdg_pictures_w; aa_xdg_music_w; aa_xdg_videos_w;
Zápis medzi stiahnuté súbory (home_downloads_w): in: home_all_w; home_downloads_rw; **out**: aa_xdg_downloads_w;
Čítanie mediálnych súborov (home_media_r): in: office; home_online; home_all_r; home_media_rw; **out**: aa_xdg_pictures_r; aa_xdg_music_r; aa_xdg_videos_r;
Zápis do home (home_all_w): in: home_all_rw; **out**: home_documents_w; home_media_w; home_downloads_w;
Médiá (rw) (home_media_rw): in: media; **out**: home_media_r; home_media_w;
Zápis dokumentov (home_documents_w): in: home_all_w; home_documents_rw; **out**: aa_xdg_documents_w;
Čítanie a zápis do používateľských priečinkov home (home_all_rw): in: –; **out**: home_all_r; home_all_w;

[a]Neuvádzame popis pre každý vrchol zvlášť, z názvu je ich význam zrejmý.

Prístup k home pre používateľské online programy (home_online): in: web; mail; **out**: home_downloads_rw; home_documents_r; home_media_r;

Vrchol vznikol operáciou spojenia *P-vrcholov* (viď. časť 2.2.2 a obrázok 2.1).

Zápis je povolený len do priečinku Downloads, ktorý je priamo určený na ukladanie stiahnutých súborov. Používateľ teda vie (alebo môže byť zaškolený), že súbory v tomto priečinku môžu pochádzať z nedôveryhodného zdroja. Následne je možné sústrediť pozornosť na tento priečinok, napríklad nastaviť antivírusovú kontrolu na každý nový súbor v tomto priečinku (napríklad pomocou softvéru *ClamAV*, ktorý je určený aj na priebežnú kontrolu príloh e-mailov). Ak sa podarí útočníkovi získať kontrolu nad prehliadačom, nie je mu umožnený zápis do ostatných používateľských priečinkov. Integrita unikátnych dát používateľa (fotografie, dokumenty, diplomová práca, ...) zostane teda zachovaná. (Útočník teda napríklad nemôže zašifrovať všetky súbory a žiadať výkupné.)

Povolenie čítať súbory môže narušiť dôvernosť uložených údajov. Preto navrhujeme nepoužívať štandardné umiestnenia na ukladanie citlivých informácií. (Viac informácií v časti o domovskom priečinku.)

Čítanie /dev (`dev_r`): **in**: `opt_sysdir`; **out**: `aa_dev_r`;

Povolenie čítať všetko v priečinku `/dev`. Niektoré aplikácie môžu potrebovať priamy prístup k zariadeniam. Väčším problémom je, že tento priečinok môže obsahovať aj rôzne virtuálne zariadenia, prípadne zariadenia pridané ovládačmi tretích strán. Tieto zariadenia je ťažšie predpovedať.

V prípade, že aplikácia požaduje čítanie z nejakého konkrétneho zariadenia, odporúčame toto konkrétne zariadenie ručne špecifikovať v profile, a tento vrchol nepoužívať. V krajnom prípade je možné tento vrchol povoliť, s tým, že v takom prípade je zabezpečenie systému závislé na správnych prístupových oprávneniach. (V prípade štandardného priečinku `dev` by mali byť oprávnenia korektne nastavené systémom, napríklad priamy prístup na disk je umožnený len administrátorovi.)

Zápis do /dev (`dev_w`): **in**: `opt_sysdir`; **out**: `aa_dev_w`;

Povolenie zapisovať do priečinku `/dev`. Niektoré aplikácie môžu potrebovať priamy prístup k zariadeniam. V prípade, že má aplikácia dostatočné systémové oprávnenia, toto povolenie môže spôsobiť, že aplikácia ľubovoľným spôsobom upraví systém (pomocou zápisu na disk). Na testovacom počítači niektoré aplikácie požadujú prístup k súboru `/dev/nvidiactl` (počítač má grafickú kartu výrobcu *Nvidia*), pričom bez tohto prístupu nefungujú. V prípade, že je to možné, odporúčame podobné nečakané prístupy povoľovať ručne (prípadne prispôsobiť graf špecifickým požiadavkám systému). Tento vrchol je určený pre krajný prípad, že takéto prideľovanie nie je praktické. Povolením tohto vrcholu je nutné spoľahnúť sa na korektne nastavené prístupové práva.

Čítanie /proc (`proc_r`): **in**: `opt_sysdir`; **out**: `aa_proc_r`;

Povolenie čítať priečinok `/proc` obsahujúci podrobné informácie o procesoch. Odporúčame povoľovať len pre špecifické aplikácie ktoré majú dôvod toto povolenie vyžadovať. Ukazuje sa, že aj bežné aplikácie (napríklad *Thunderbird*) sa pokúšajú pristupovať k informáciám o svojom procese, avšak často aplikácia funguje aj bez tohto povolenia. (Je problém vopred určiť PID procesu, bolo by teda nutné udeliť úplný prístup.)

Čítanie /run (`run_r`): **in**: `opt_sysdir`; **out**: `aa_run_r`;

Povolenie čítať priečinok `/run` obsahujúci premenlivé dáta vytvorené od posledného spustenia systému. Väčšinou ide o údaje systémových démonov, avšak priečinok môže obsahovať aj súbory ďalšieho softvéru. Obsah tohto priečinku na konkrétnom počítači teda nie je možné v práci vopred predpovedať. Preto odporúčame toto povolenie neprideľovať, pokiaľ ho daná aplikácia nepotrebuje. Povolenia

na špecifické priečinky požadované aplikáciou je možné pridať do grafu, alebo pridať do profilu ručne.

Vrcholy reprezentujúce používateľské prostredia:

Unity 7 (`unity7`)**:** **in**: `opt_de`; **out**: `ab_unity7_base`; `ab_unity7_launcher`;

Používateľské prostredie Unity je štandardne dodávané so systémom Ubuntu.

KDE (`gui_kde`)**:** **in**: `opt_de`; **out**: `ab_kde`; `config_kdeglobals`; `cache_kcache`;

Používateľské prostredie *KDE*.

Gnome (`gui_gnome`)**:** **in**: `opt_de`; **out**: `ab_gnome`;

Tento vrchol je možné použiť nie len pre používateľské prostredie *Gnome*, ale aj v prípade použitia prostredia *Cinnamon*. Použitá abstrakcia pridá niektoré oprávnenia, ktoré by bolo nutné pridať ručne (napríklad príslušné súbory v domovskom priečinku potrebné pre fungovanie grafických tém *GTK*).

3.2 D-vrcholy

Voliteľné informácie o sieti (`opt_info_network`)**:** **in**: `info_network`; **out**: `nm_state`;

Možnosť prideliť aplikácií povolenia pre získavanie informácií o sieti, ktoré nie sú nutné pre samotné vykonávanie komunikácie.

Unix socket (`opt_net_unix`)**:** **in**: `settings`; `network_com`; `dev_nvidia`; **out**: `net_unix`;

Umožňuje používateľovi povoliť komunikáciu medzi procesmi pomocou *unixových socketov*. Keďže sa nejedná o typickú komunikáciu medzi počítačmi, pod vrchol `network_com` je zavesený práve tento vrchol a nie `net_unix`.

Metóda vstupu (`opt_input`)**:** **in**: `input`; **out**: `ibus`;

Umožňuje vybrať povolenia na čítanie konfigurácie metód vstupu a na vykonávanie ďalších operácií (v závislosti od zvolenej metódy).

Používateľské prostredie (`opt_de`)**:** **in**: `gui`; **out**: `gui_kde`; `unity7`; `gui_gnome`;

Umožňuje používateľovi vybrať si používateľské prostredie. Na základe výberu budú aplikácií priradené potrebné oprávnenia (ideálne už preddefinované abstrakcie v systéme). Vzhľadom na veľké množstvo dostupných používateľských prostredí môže byť nutné pod tento vrchol pripojiť ďalšie.

home/cache (`opt_cache`): **in**: `basic_app`; **out**: `cache_appname`; `cache_kcache`; `cache_all`;

Možnosť upresniť prístup k priečinku `.cache` v domovskom priečinku. Podľa *XDG* špecifikácie [18] ide o štandardnú pozíciu kde by mali byť uložené používateľské dáta s nízkou dôležitosťou.

home/config (`opt_config`): **in**: `basic_app`; **out**: `config_all`; `config_appname`; `config_kdeglobals`;

Možnosť upresniť prístup k priečinku `.config` v domovskom priečinku. Podľa *XDG* špecifikácie [18] ide o štandardnú pozíciu kde by mali byť uložené používateľské konfiguračné súbory.

Špecifické systémové priečinky (`opt_sysdir`): **in**: `basic_app`; **out**: `run_r`; `proc_r`; `dev_r`; `dev_w`;

Všeobecné prístupy k systémovým priečinkom. Na rozdiel od používateľských priečinkov majú tieto priečinky systémom definované oprávnenia.

3.3 K-vrcholy

Sieťové programy (`net`): **in**: –; **out**: `web`; `mail`;

Programy, ktoré na napĺňanie svojho primárneho účelu potrebujú prístup k sieťovej komunikácií. Ide o zoskupenie kategórií aplikácií. V prípade, že je to možné odporúčame použiť konkrétnu kategóriu.

Webový prehliadač (`web`): **in**: `net`; **out**: `basic_app`; `opt_audio`; `print`; `input`; `dns`; `ssl`; `info_network`; `home_online`; `network_com`;

V súčastnosti webové prehliadače napĺňajú široké spektrum úloh. Ich hlavným cieľom je prostredníctvom sieťovej komunikácie stiahnuť webový dokument a zobraziť ho používateľovi, preto je táto kategória typickým predstaviteľom sieťových programov. Webové dokumenty sú často tvorené nedôveryhodnou treťou stranou, čiže sa môžu snažiť zneužiť chyby v prehliadači. Na druhú stranu, veľké množstvo požiadaviek pri legitímnej činnosti spôsobuje, že je nutné hľadať kompromis medzi bezpečnosťou a funkčnosťou prehliadača. Ako v celej práci, zameriavame sa najmä na používateľa v domácnosti alebo v kancelárií.

Cieľom útočníka môže byť získavanie údajov, prípadne poškodenie dostupnosti systému. Nie je možné vopred presne určiť, aká zraniteľnosť sa v prehliadači bude nachádzať, avšak môžeme predpokladať, že útočník získa možnosť vykonávať činnosti s privilégiami prehliadača. (Predpokladáme, že chyba nie je v samotnom operačnom systéme. Napríklad predpokladáme, že implementácia *AppArmoru* je korektná.) Jednoduchšou úlohou je ochrániť integritu systému, keďže prehliadače potrebujú upravovať obmedzenú skupinu súborov. Môžeme teda zabrániť úprave používateľových konfiguračných súborov, ktoré by napríklad útočníkovi umožnili vytvoriť si perzistentný prístup do systému. Zachovať dôvernosť súborov je náročnejšie, keďže *AppArmor* neponúka mechanizmus na značkovanie súborov podľa dôvernosti, avšak prehliadače môžu byť použité na odovzdávanie lokálnych súborov. Detailnejšie odôvodnenie jednotlivých požiadaviek uvádzame nižšie.

- **Bežná aplikácia:** Podstatná časť vyžadovaných oprávnení je schovaná pod toto povolenie. Najmä v prípade webových prehliadačov ide o vhodné zjednodušenie. (Nie je možné klásť veľký dôraz na konkrétne požiadavky, prehliadače ich majú priveľa a nie všetky sú bežne očakávateľné. V tomto prípade je teda istá miera abstrakcia mimoriadne vítaná).

- **Audio:** Súčasťou webových stránok je aj multimediálny obsah, prípadne možnosť vykonávať hlasovú komunikáciu. Pre domáceho používateľa je teda toto povolenie prakticky nutnosť. Napriek tomu toto povolenie neprideľujeme automaticky, ale dávame možnosť výberu, keďže prehliadač môže byť používaný napríklad ako tenký klient k administratívnemu softvéru vo firme. V takom prípade zvuk nie je potrebný. Mediálny obsah môže znižovať produktivitu zamestnancov, zatiaľ čo nahrávanie zvuku môže byť (úmyselne alebo omylom) zneužité na priemyselnú špionáž.

- **Tlač:** Tlač webových dokumentov je bežná, štandardne očakávateľná, používateľská činnosť. Riziko spôsobené udelením povolenia na tlač je malé.

- **Spravovanie nastavení (dconf):** Ukazuje sa, že niektoré webové prehliadače (napríklad *Firefox*) toto povolenie vyžadujú. Keďže ide len o čítanie nastavení, prehliadač nemôže týmto zasahovať do ostatných aplikácií.

- **Správa vstupov:** Pravdepodobne pre zjednodušenie života používateľovi, niektoré prehliadače vedia užšie spolupracovať s metódami vstupu. (Napríklad *Firefox* štandardne v profile dodávanom so systémom *Linux Mint* očakáva `<abstractions/ibus>`.) Preto umožňujeme osobe vykonávajúcej konfiguráciu vybrať, ktoré metódy vstupu sú v systéme používané.

- **Rôznorodé informácie týkajúce sa sietí a komunikácie:** Webové prehliadače na svoju činnosť často potrebujú nadštandardné informácie. Môžu nahrádzať niektoré úlohy operačného systému napríklad zobrazovať informácie o chybách a o sieti. V prípade pripojenia bezdrôtovou sieťou môže byť informácia o sile signálu použitá na lokalizovanie zariadenia. Podľa toho, aké sieťové rozhranie je použité môže prehliadač rozhodnúť, ako sa bude správať k obsahu (šetrenie dát a podobne). Keďže prehliadače sú často tvorené pre viacero platforiem, môže to spôsobiť, že implementujú časti ktoré vie poskytnúť systém.
- **Prístup k používateľským priečinkom v domovskom priečinku:** Podrobnosti v popise vrcholu `home_online`.

  Čítanie je povolené z používateľských priečinkov obsahujúcich dokumenty a mediálne súbory, nakoľko používateľ môže vyžadovať odoslanie súborov na vzdialený server. (Potenciálny problém s narušením dôvernosti dát je spomenutý v časti o domovskom priečinku). Zápis je povolený do priečinku určeného pre ukladanie stiahnutých súborov (`Downloads`). K tomuto priečinku je v prípade potreby možné pristupovať opatrnejšie, napríklad nastaviť kontrolu každého zmenených súborov pomocou antivírusového softvéru, alebo zaviesť organizačné opatrenia v prípade systému ktorý používa viacero používateľov. Očakávame, že používateľ ručne presunie súbory, ktoré chce ponechať, na príslušné miesto. V prípade napadnutia prehliadača ostane integrita súborov v ostatných priečinkoch zachovaná.

Zobrazovač dokumentov (`reader`): **in**: `office`; **out**: `basic_app`; `print`; `home_documents_r`;

Kategória reprezentuje programy na zobrazovanie dokumentov. Nepatrí sem softvér na úpravu dokumentov (kancelársky softvér, textové editory, vývojové prostredia, ...). Čo sa týka prístupu k používateľským priečinkom, tejto kategórií programov stačí čítať priečinok obsahujúci dokumenty. Tieto programy ďalej môžu slúžiť na tlač dokumentov.

Na rozdiel od kancelárskeho softvéru (`office`) tieto programy nepotrebujú ukladať dokumenty na vopred neznáme miesta, čo teda značne uľahčuje ochranu integrity údajov v systéme. Preto v prípadoch keď je to možné odporúčame uprednostniť túto kategóriu. Ak útočník potrebuje zapísať súbor (napríklad zmeniť konfiguračný súbor alebo vytvoriť spustiteľný program alebo skript), nepodarí sa mu to.

Mailový klient (`mail`): in: `net`; **out**: `print`; `basic_app`; `home_online`; `network_com`;

Do tejto kategórie patria programy na posielanie a prijímanie mailov. Zameriavame sa najmä na moderné programy, preto očakávame že sa jedná o bežnú aplikáciu poskytujúcu grafické rozhranie. Z hľadiska bezpečnosti sú podobné webovým prehliadačom (`web`), nakoľko pracujú s cudzím aj lokálnym obsahom.

Výpočty (`turing`): in: -; **out**: `binlib`;

Kategória aplikácií, ktoré nemajú takmer žiadne požiadavky na systém. Aplikácia len dostáva vstup, vykonáva nejaké výpočty, a vypíše výstup. Nepožadujeme žiadne špeciálne vedľajšie efekty. Praktické využitie môže nájsť táto kategória pri hodnotení algoritmických úloh študentov, v prípade, že vyučujúci chce aplikáciu spustiť, ale nechce, aby napríklad vynášala informácie po sieti alebo modifikovala súbory. Podobné využitie existuje pri rôznych algoritmických súťažiach, kde treba otestovať, či daná aplikácia počíta to čo má na testovacích vstupoch. V oboch prípadoch treba požadovať vstup a výstup len cez štandardný vstup a výstup (nie zo súborov). Praktickejším využitím môžu byť rôzne špecializované „kalkulačky", napríklad pre počítanie fyzikálneho výpočtu, skóre v hrách a podobne.

Táto kategória nemusí nutne obsahovať aplikácie úplne nezávislé od systému. Keďže v Linuxových systémoch je (najmä zásluhou *open-source* filozofie) zvykom čo najviac používať dynamické linkovanie, povoľujeme spúšťanie a linkovanie súborov. Spustenie iného súboru alebo prilinkovanie knižnice umožní zdieľať strojový kód, avšak obmedzenia *AppArmor*om zostanú v platnosti. V prípade, že chceme použiť statické linkovanie, nie je nutné udeliť žiadne povolenie. Tým dosiahneme ideál zabezpečenia, aspoň v rámci možností *AppArmor*u. (V takomto prípade stačí nepoužiť žiaden vrchol. Nevytvárali sme špeciálny *K-vrchol*, z ktorého by následne neviedli žiadne hrany.) V prípade statického linkovania veľkosť binárneho súboru narastie[4], avšak pri niektorých príkladoch ktoré sme uviedli to nemusí byť problém.

Kancelársky softvér (`office`): in: -; **out**: `reader`; `home_documents_rw`; `home_media_r`;

Kategória reprezentujúca aplikácie, ktoré sú súčasťou kancelárskych balíkov. Na rozdiel od kategórie *zobrazovač dokumentov* (`reader`) je možné dokumenty aj zapisovať. Pri vytváraní dokumentov môže byť potrebné čítať mediálne súbory (pridávanie obrázkov, zvuku a videí do prezentácií a podobne). Problém môžu spôsobiť špeciálne upravené súbory vytvorené útočníkom, ktoré zneužívajú prípadný

[4]Pre ilustráciu, C++ program ktorý pomocou knižnice *iostream* načíta dve čísla a vypíše ich súčet má pri dynamickom linkovaní $20kB$, zatiaľ čo pri statickom linkovaní $2.2MB$.

zabudovaný makro systém, alebo inú zraniteľnosť za účelom spustenia vlastného kódu. Zápis je obmedzený na priečinok s dokumentmi, čo slúži na ochranu integrity konfiguračných a ďalších súborov.

Hry (`game`): in: `-`; **out**: `basic_app`; `network_com`;

Kategória reprezentujúca hry. Spolu s kategóriou *webový prehliadač* (`web`) ide o jednu z najproblematickejších kategórií. Veríme, že kategóriu *bežná aplikácia*(`basic_app`) sme navrhli dostatočne robustne, aby obsiahla požiadavky aj takto náročnej kategórie (z hľadiska potrebných požiadaviek pre funkčnosť). Je možné, že osoba vykonávajúca konfiguráciu bude musieť povoliť väčší počet voliteľných požiadaviek. Typická hra obsahuje aj režim pre viacerých hráčov (prípadne umožňuje sťahovať doplnky, alebo tabuľku najlepších hráčov), preto povoľujeme aj sieťovú komunikáciu.

Čo sa týka bezpečnosti, malou netechnickou výhodou je, že hry často pochádzajú od veľkých vývojárskych štúdií alebo zo známejších open-source projektov, teda motivácia poškodiť používateľa je menšia. Avšak nemusí to platiť pre všetky hry. Navyše, komerčné produkty v záujme zvyšovania zisku tiež môžu narušiť súkromie používateľa, prípadne modifikovať systém nevhodným spôsobom.

Programy na prácu s médiami (`media`): in: `-`; **out**: `basic_app`; `home_media_rw`;

Programy, ktoré dokážu otvoriť mediálny súbor, pracovať s ním, a v prípade potreby uložiť zmeny. Medzi mediálne súbory radíme obrázky, videá a audio súbory. Patria sem napríklad editory fotiek, programy na strih videa, 3D modelovací softvér a ďalšie.

3.4 Nepopísané D-vrcholy

Nie všetky vrcholy sú slovne popísané v tejto kapitole. Ak napríklad z *D-vrcholu* vedú hrany do *P-vrcholov*, pričom všetky tieto *P-vrcholy* majú uvedený slovný popis, uvádzanie popisu *D-vrcholu* by bolo redundantné. Pre úplnosť uvádzame ich zoznam bez podrobných komentárov.

- `opt_xdgdesktop` - Použiť XDG-desktop abstrakciu
- `opt_etc` - /etc
- `opt_dbus` - D-bus
- `opt_audio` - Audio
- `opt_home_appdir` - home/appdir
- `opt_dconf` - dconf

- opt_ssl_conf - Čítanie konfigurácie SSL
- opt_nvidia
- opt_tmpm
- opt_local

3.5 A-vrcholy

Poznámka: Keďže *A-vrcholy* iba implementujú požiadavky, popis ich významu je možné nájsť v rodičovskom *P-vrchole*. Na tomto mieste sú pre úplnosť vymenované.

- ab_kde
- ab_cups_client
- aa_xdg_videos_r
- ab_X
- aa_xdg_music_r
- aa_xdg_downloads_r
- aa_xdg_videos_w
- ab_gnome
- aa_xdg_documents_r
- aa_cache_all
- aa_xdg_pictures_r
- ab_xdgdesktop
- aa_arp
- aa_proc_net_wireless
- aa_cache_kcache
- ab_unity7_base
- aa_bin_lib
- ab_nameservice
- ab_freedesktop
- ab_dconf
- ab_audio
- ab_openssl
- ab_ibus
- aa_xdg_downloads_w
- aa_cache_appname
- ab_base
- aa_config_kdeglobals
- aa_dbus_nm
- ab_tmp
- aa_dbus_strict
- aa_dbus_session_strict
- aa_ls_fs
- ab_fonts
- aa_etc_all
- ab_unity7_launcher
- aa_dbus_accessibility_strict
- aa_xdg_music_w
- aa_config_all
- aa_xdg_pictures_w
- aa_config_appname
- aa_xdg_documents_w
- aa_network
- aa_info_net_if
- aa_run_r
- aa_nvidia
- aa_stream
- aa_dgram
- aa_tcp
- aa_udp
- aa_home_appdir
- aa_dev_r
- aa_dev_w
- aa_proc_r
- aa_local_all
- aa_local_appname
- aa_net_unix
- aa_tmp_m

Kapitola 4

Konfiguračný nástroj pre tvorbu profilu

V kapitolách 2 a 3 sme zadefinovali grafový model pre formálnu reprezentáciu požiadaviek aplikácie a spôsob implementácie daných požiadaviek. Pohľad na grafickú reprezentáciu grafu a čítanie dokumentácie dá používateľovi prehľad, avšak nie je praktické, aby následne používateľ (alebo správca systému) vykonával celú konfiguráciu ručne pre každú používanú aplikáciu. Preto odporúčame použiť priložený konfiguračný nástroj. V tejto kapitole popíšeme tento nástroj a spôsob akým ho používať.

4.1 Vytvorenie profilu pre aplikáciu

4.1.1 Práca s dokumentáciou

Pred začiatkom akejkoľvek konfigurácie je dôležité prečítať si túto prácu. Odporúčame si aspoň raz prečítať kapitoly 1, 2, 3 a 4. Pre ďalšiu činnosť následne môže stačiť vyhľadávanie potrebných informácií v kapitole 3, a v prípade potreby systematické postupovanie podľa časti 4.1.

Počas vytvárania profilu je dôležité študovať popis konkrétneho vrcholu, ale aj popis vrcholov spojených hranami s daným vrcholom. (V elektronickej verzii textu je možné kliknúť na odkaz, v papierovej verzii pomôže register na konci. Prípadne je možné zobrazovať dokumentáciu pomocou konfigurátora.) K *D-vrcholu* môže byť priradený popis, ktorý všeobecne opisuje dané rozhodnutie, avšak podstatná informácia sa môže nachádzať aj v popise *P-vrcholov* pod daným *D-vrcholom*. *P-vrchol* v takomto prípade bude obsahovať už konkrétny popis samotnej možnosti. Ak má *D-vrchol* priradenú len jednu možnosť, písanie jeho popisu je vo väčšine prípadov redundantné, ide len o rozhodnutie typu „*áno/nie*" pre jedinú

požiadavku. *K-vrchol* môže popísať dôvody pridelenia jednotlivých požiadaviek, príslušné *P-vrcholy* môžu popísať požiadavku a situácie pre jej udelenie z pohľadu nezávislého od konkrétnej kategórie aplikácií.

V prípade, že je z názvu vrcholu zrejmé, čo daný vrchol popisuje, popis nie je uvedený[1].

4.1.2 Objasnenie pojmov

V tejto kapitole sú spomenuté rôzne typy zoznamov a ďalšie pojmy, ktoré majú špecifický význam. Používame viac-menej intuitívne označenia. Keďže pochopenie ich významov je dôležité pre porozumenie jednotlivým príkazom konfigurátora, uvádzame na tomto mieste popis jednotlivých zoznamov.

- *zoznam zahrnutých vrcholov*: Zoznam vrcholov, ktoré sú v danom momente zahrnuté do profilu. Inými slovami, budú použité pri vytváraní textu profilu príkazmi `export` alebo `print`.
- *zoznam ručne vybratých vrcholov*: Vrcholy, ktoré používateľ ručne označil na zahrnutie do profilu príkazom `keep`. Neobsahuje vrcholy, ktoré sa pridajú do profilu automaticky pri rekurzívnom prehľadávaní.
- *fronta na prehľadanie*: Zoznam vrcholov, do ktorého je možné na jeho koniec pridať vrchol príkazom `search`. Okrem toho vrcholy pridáva konfigurátor automaticky v prípade, že pri rekurzívnom prehľadávaní narazí na *D-vrchol*. Príkaz `next` vyberie prvý prvok ktorý bude považovaný za *aktuálny prvok*. Zoznam slúži na ukladanie prvkov, na ktoré by sa mal používateľ konfigurátora ešte pozrieť a uľahčuje celý proces vytvárania profilu.
- *aktuálny prvok*: Aktuálne spracovávaný prvok. Používateľ by sa mal rozhodnúť, či chce daný vrchol alebo jeho potomkov zahrnúť príkazom `keep`, prípadne či chce použiť príkaz `search` na potomkov vrcholu.

4.1.3 Použitie konfigurátora

Pre spustenie je potrebné mať nainštalované potrebné knižnice, ktorých zoznam je možné nájsť v súbore `requirements.txt`.

[1] Zväčša ide o situácie, keď daný vrchol popisuje konkrétny objekt alebo koncept.

Príklad príkazov pre prvé spustenie:

```
virtualenv -p python3 venv
source venv/bin/activate
pip install -r requirements.txt
./configurator dag.dot /usr/bin/firefox
```

Pri ďalších spusteniach stačí pre dané sedenie raz aktivovať virtuálne prostredie, a následne je možné používať konfigurátor. Prípadne je možné nepoužiť virtuálne prostredie, a nainštalovať potrebné systémové balíčky. Prvý argument je cesta k súboru s grafom, druhý argument je cesta k binárnemu súboru pre ktorý vytvárame profil.

Konfigurátor počas interaktívneho sedenia postupne prechádza vrcholy na prehľadanie v poradí *FIFO*. Na začiatku sa nachádza v tejto fronte iba pomyselný koreň označený „`ROOT`". Osoba vykonávajúca konfiguráciu môže zadávať rôzne príkazy:

- `keep` ***názvy vrcholov oddelené medzerami***: Zahrnutie vrcholov, ktoré chceme ponechať v profile. Program rekurzívne prechádza v smere hrán na ďalšie vrcholy, pričom *A-vrcholy* rovno priraďuje do *zoznamu zahrnutých vrcholov*, *D-vrcholy* pridáva do fronty *na prehľadanie*. Očakávané použitie je, že si používateľ vyberie kategóriu aplikácie spomedzi ponúkaných *K-vrcholov*, alebo v prípade *D-vrcholu* podmnožinu jeho synov. Príkaz je však možné použiť na ľubovoľné vrcholy (fronta vrcholov *na prehľadanie* a príkazy `next` a `current` slúžia osobe vykonávajúcej konfiguráciu len ako pomôcka).

- `search` ***názvy vrcholov oddelené medzerami***: Oznámenie konfigurátoru, že nechceme daný vrchol ihneď rekurzívne zahrnúť do profilu, ako napríklad v prípade `keep`, avšak očakávame, že budeme chcieť možno použiť niektorých synov (či už pomocou `keep` alebo `search`). Konfigurátor uvedené vrcholy zaradí do fronty *na prehľadanie*, a neskôr ich ponúkne na zváženie.

- `next` alebo *prázdny riadok*: Konfigurátor vyberie a zobrazí ďalší vrchol z fronty *na prehľadanie*. V prípade, že už vo fronte nie je žiaden vrchol, príkaz iba vypíše chybové hlásenie. (Možnosť použiť prázdny riadok namiesto príkazu `next` je praktická napríklad po použití príkazov `keep` a `search`. Zadanie príkazov ukončujeme stlačením klávesy `Enter`, čiže opakovaným stlačením sa rovno presunieme na ďalší vrchol.)

- `current`: Vypíše aktuálny vrchol a zoznam pod-vrcholov. Pri zahrnutých

vrcholoch sa zobrazuje zelená farba, pri navštívených[2], ale nevybratých, žltá farba.

- `list`: Program vypíše zoznam vrcholov, ktoré boli ručne vybraté príkazom `keep` (*zoznam ručne vybratých vrcholov*). Červenou farbou pozadia sú vyznačené vrcholy, ktoré sa nenachádzajú v grafe. Sivou farbou sú vyznačené pozmenené vrcholy. Tieto špeciálne prípady môžu nastať iba v prípade, že načítaný súbor bol vytvorený podľa iného grafu.

- `remove` ***`názvy vrcholov oddelené medzerami`***: Program odstráni zadané vrcholy zo *zoznamu ručne vybratých vrcholov*.

- `apply`: Všetky vrcholy zo *zoznamu ručne vybratých vrcholov* sa automaticky vyberú do *zoznamu zahrnutých vrcholov*. Výber sa vykonáva rekurzívne, podobne ako v prípade použitia príkazu `keep`.

- `save` ***`cesta k súboru`***: Aktuálny stav programu sa uloží do súboru.

- `load` ***`cesta k súboru`***: Načíta uložený stav programu. Obnoví sa informácia o aktuálne prehľadávanom vrchole, *Zoznam zahrnutých vrcholov* a zoznam vrcholov *na prehľadanie*. Informácie o navštívených a zahrnutých vrcholoch sa aplikujú na aktuálny graf (ktorý teoreticky môže byť odlišný od uloženého grafu). *zoznam ručne vybratých vrcholov* sa rozšíri o prvky zo súboru.

  Je nutné poznamenať, že vrcholy sa obnovujú celé, aj s obsahom a pod-vrcholmi, nie len podľa názvu. Môže to pomôcť pri použití rôznych grafov v čase ukladania a v čase obnovy. (Prípadne pri strate súboru s grafom.)

- `reset`: Zmaže všetko, okrem *zoznamu ručne pridaných vrcholov*.

- `help`: Zobrazí zoznam dostupných príkazov. V prípade uvedenia konkrétneho názvu príkazu vypíše stručnú nápovedu k danému príkazu.

- `man` ***`parameter`***: Spustí príkaz `documentation.sh` s uvedeným parametrom. Slúži na otvorenie strany dokumentu, na ktorej sa nachádza popis daného vrcholu. Do istej miery to nahrádza nutnosť používať papierovú dokumentáciu. Samotný skript `documentation.sh` je nutné prispôsobiť konkrétnemu systému a dostupným programom.

[2]Pod pojmom „*navštívených*" myslíme vrcholy navštívené používateľom, teda že používateľ mal možnosť urobiť rozhodnutie o danom vrchole a pod-vrcholoch. Nerátajú sa sem vrcholy, ktoré konfigurátor prehľadal automaticky v rámci rekurzie.

Odstraňovanie vybratých vrcholov: Keďže pri zahrnutí vrcholu príkazom `keep` sa môžu automaticky zahrnúť ďalšie vrcholy, odstránenie vrcholu, pridaného príkazom `keep`, z profilu nie je triviálne. Po dokončení konfigurácie je možné príkazom `remove` odstrániť vrchol zo *zoznamu ručne vybratých vrcholov*, príkazom `reset` premazať všetko okrem tohto zoznamu, a nakoniec aplikovať takto upravený zoznam príkazom `apply`. Tento postup môže byť užitočný napríklad pri testovaní: v prípade, že je žiadané z profilu niečo odstrániť je možné načítať uložený stav a použiť spomenutú techniku.

Automatické dopĺňanie textu: Počas používania konfigurátora je možné používať klávesu `Tab` pre automatické dopĺňanie textu. Funguje pre názvy príkazov, názvy vrcholov (z grafu alebo *zoznamu ručne vybratých vrcholov* - podľa kontextu) a cesty súborov.

Zmena obsahu grafu: Ako už bolo spomenuté, rátame aj s možnosťou zmeny grafu. Môže sa stať, že v priebehu úprav potrebujeme do grafu pridať nejaké vrcholy, prípadne potrebujeme pridávať vrcholy z viacerých grafov. V týchto situáciách je možné uložiť stav programu a načítať ho v novom behu programu, ktorý už bude používať nový graf. Pokiaľ je to možné, odporúčame vyhýbať sa načítavaniu uloženého súboru na odlišnom grafe. V prípade použitia odlišného grafu očakávané použitie vyzerá nasledovne:

- kontrola *zoznamu ručne pridaných vrcholov*: Treba overiť, či tento zoznam neobsahuje zmazané alebo pozmenené vrcholy, a uvážiť, či ich chceme ponechať alebo odstrániť a nahradiť novými. Tento zoznam je možné zobraziť príkazom `list`.
- použitie príkazov `keep` a `next`: Pri pridávaní vrcholov je potrebné brať do úvahy, že po načítaní zoznam vrcholov *na spracovanie* obsahuje vrcholy z pôvodného grafu. Tieto vrcholy nemusia v novom grafe existovať, alebo môžu byť pozmenené, nemusia byť teda žiadúce. Pred prvým použitím príkazu `keep` (po načítaní súboru) odporúčame teda použiť príkaz `reset`.

4.1.4 Testovanie

Po vykonaní ľubovoľných zmien v konfigurácii je dôležité otestovať, či všetka používaná funkcionalita aplikácie funguje. Používame nasledovný postup (ktorý je možné prispôsobovať svojim potrebám):

- Zavedieme profil do *enforcing* režimu (viď. časť 4.1.6).

- V prvom terminálovom okne spustíme vypisovanie systémového logu príkazom `dmesg -w`. Parameter `-w` spôsobí, že program `dmesg` bude čakať na nové správy.
- Pomocou ďalšieho terminálového okna spustíme požadovanú aplikáciu.
- Vykonávame činnosti, pre ktoré by mala aplikácia fungovať a monitorujeme výpisy v terminálových oknách. V prípade, že *AppArmor* zablokuje nejaký prístup, zapíše o tom informáciu do systémového logu. Samotná aplikácia môže na štandardný výstup, alebo štandardný chybový výstup vypisovať ladiace informácie, ktoré tiež môžu byť užitočné. Na rozdiel od výpisov *AppArmor*u, výpisy aplikácie môžu obsahovať nadhľad o tom, čo sa aplikácia snažila vykonať.

V prípade, že by mal *AppArmor* pri vypisovaní do logu použiť špeciálne znaky, zakóduje text ako reťazec hexadecimálnych čísel. Takto zakódovaný reťazec je možné spätne dekódovať príkazom `aa-decode 4F4B4E4F`, kde *4F4B4E4F* treba nahradiť príslušným reťazcom.

4.1.5 Ručné pridávanie

V prípade, že po vykonaní konfigurácie za pomoci konfigurátora nie je dostupná všetka funkcionalita, je potrebné aplikácii pridať ďalšie povolenia. Po vykonaní testovania (časť 4.1.4) by malo byť zrejmé, aké povolenie aplikácii chýba. Najprv je potrebné overiť, či sa potrebné povolenie naozaj nenachádza vo vhodnom vrchole v grafe. V prípade, že nie, je možné vytvoriť v grafe nové vrcholy. V prípade, že je povolenie špecifické pre danú aplikáciu, a nie je možné ho zovšeobecniť, je nutné pridať príslušné povolenie do profilu ručne podľa znalostí z kapitoly 1.

4.1.6 Zavedenie a zablokovanie profilu

Konfiguračný súbor s profilom je nutné umiestniť v priečinku `/etc/apparmor.d/`. Následne je možné zapnúť vynucovanie profilu pomocou nasledovného príkazu: `aa-enforce <cesta k súboru profilu>`. Profil je možné v prípade potreby zablokovať príkazom `aa-disable <cesta k súboru profilu>`.

4.1.7 Možné problémy

Symbolický odkaz: Ak pre zisťovanie cesty k binárnemu súboru aplikácie používame príkaz `which`, môže nastať problém, že cesta, ktorú takto získame, je iba symbolický odkaz. Ak použijeme túto cestu, profil sa vôbec nebude

aplikovať. (Napríklad, `/usr/bin/thunderbird` na testovanom systéme odkazuje na súbor `/usr/lib/thunderbird/thunderbird.sh`. V tomto prípade overiť to, že ide o odkaz je možné príkazom `ls -l /usr/bin/thunderbird` alebo `stat /usr/bin/thunderbird`.)

Viacero súborov popisuje profil pre ten istý program: V priečinku `/etc/apparmor.d` sa nemôže nachádzať viacero profilov popisujúcich rovnaký program. Ak je potrebné odložiť si neaktívne verzie profilu, treba nakoniec ich názvov pridať znak `~`. V opačnom prípade pri ďalšom vykonaní `aa-enforce` nastane chyba.

4.2 Implementácia konfigurátora

Konfigurátor je program napísaný v jazyku *python3*. (Nachádza sa v elektronickej prílohe pod názvom `configurator.py`). Pre rýchly prístup do tejto dokumentácie slúži *Bash*-skript `documentation.sh`. Súbor `fix_links.py` obsahuje ukážku programu na úpravu zdrojového súboru dokumentácie vo formáte pre systém LaTeX, nebudeme sa mu podrobnejšie venovať. Nasledujúce časti popíšu funkcionalitu a implementačné detaily.

4.2.1 Spracovanie vstupného súboru

Ku spracovaniu textu sa dá pristupovať rôznymi spôsobmi. Existujú rôzne knižnice na parsovanie textu, napríklad všeobecne použiteľná knižnica *Pyleri* [15]. Využitie všeobecných knižníc má dve hlavné nevýhody. Rozhranie na prácu s knižnicou je všeobecné, a teda často zbytočne náročné na použitie pre konkrétnu aplikáciu. Zároveň je nutné, aby používateľ mal danú knižnicu nainštalovanú, čo nielen, že zaberá zbytočne veľa diskového priestoru (vzhľadom na prínos pre konkrétnu aplikáciu), ale môže mierne skomplikovať nasadenie riešenia.

Vhodnejšími kandidátmi sú knižnice určené špecificky na prácu so súbormi typu `dot`. Príkladmi sú knižnice *Pydot* [13] a *Pydot plus* [14]. Tieto knižnice však majú svoje nedostatky. Po sparsovaní súboru by aj tak bolo nutné implementovať napríklad priradenie typu vrcholu. Knižnica *Pydot plus* neumožňuje praktickú prácu s pod-grafmi.

Finálne riešenie teda nepoužíva žiadnu knižnicu na spracovanie vstupného súboru. Definícia formátu súboru na uloženie grafu umožňuje jednoduché spracovanie

v jednom prechode po riadkoch. Pre každý riadok sa overujú podmienky v nasledovnom poradí:

- V prípade, že riadok obsahuje začiatok bloku ({) je aktuálny typ vrcholu vložený na zásobník a spracovanie pokračuje ďalším riadkom.
- V prípade, že riadok obsahuje koniec bloku (}) sa aktuálny typ vrcholu nastaví na hodnotu, ktorá je vybratá so zásobníka. Spracovanie pokračuje ďalším riadkom.
- Ak sa na riadku nachádza definícia typu vrcholu, posledná definícia typu vrcholu v riadku určí *aktuálny* typ vrcholu.
- Ak sa v riadku nachádzajú definície vrcholov, tak sa príslušné vrcholy pridajú do grafu.
- Ak sa v riadku nachádzajú definície hrán, príslušné hrany sa pridajú do grafu.

Pre vyhľadávanie významných častí riadku sú použité regulárne výrazy. Regulárne výrazy pre overenie začiatku a konca bloku overujú, či sa príslušná zátvorka nenachádza v reťazci oddelenom úvodzovkami (pred zátvorkou môže byť len párny počet úvodzoviek).

Poznámka: Na parsovanie `.dot` súborov existujú knižnice *pydot* a *pydotplus*. Chýba im však dostatočná podpora pod-grafov (*subgraph*{}). Prispôsobená implementácia čítania súborov má výhodu, že je možné definovať ďalšie rozšírenia jazyka na popis grafov pre konfigurátor (napríklad umiestnením príkazov do komentárov jazyka `.dot`).

4.2.2 Vstup a používateľské rozhranie

Do úvahy prichádzajú dva hlavné druhy používateľského rozhrania: grafické rozhranie, alebo príkazový riadok. Ďalej existuje možnosť vykresľovať grafické rozhranie v termináli pomocou knižnice *curses* (knižnica pre *python 3* ktorá používa verziu knižnice pre *C*) alebo *urwid*. Druhá menovaná poskytuje prostriedky na vykresľovanie zložitejších prvkov rozhrania, napríklad tlačidlá a dialógy.

Z viacerých dôvodov používame príkazové rozhranie:

- Cieľom konfigurátora je previesť osobu vykonávajúcu konfiguráciu všetkými rozhodnutiami. Dotyčná osoba nemusí ručne prehľadávať graf, jednotlivé vrcholy sú zobrazované postupne za sebou. Príkazový riadok (ktorý zo svojej

podstaty zobrazuje aj históriu príkazov a odpovedí) je na takýto jednosmerný postup prirodzene vhodný.

- Program je možné spustiť aj bez podpory grafického rozhrania. Očakávame, že na bežnom používateľskom počítači je grafické rozhranie dostupné. Avšak v prípade, že by mal používateľovi niekto (napríklad administrátor vo firme) vykonať konfiguráciu na diaľku, je praktickejšie použiť čisto textový vzdialený prístup[3].

- Je jednoduchšie na diaľku (napríklad cez telefón) radiť osobe vykonávajúcej konfiguráciu. Nie je nutné popisovať polohu grafických prvkov, stačí spomenúť text príkazu.

- Predpokladáme, že bežný používateľ Linuxového systému by mal byť schopný spustiť program v príkazovom riadku a nasledovať inštrukcie.

Pre spracovanie vstupu je použitá (štandardne pribalená) knižnica *cmd*. Táto knižnica zabezpečuje automatické dopĺňanie názvov príkazov. Pre dopĺňanie argumentov je možné definovať funkcie, ktoré dostanú ako argumenty argument príkazu, celý riadok príkazu a pozíciu argumentu v riadku. Výstupom funkcie je zoznam možných doplnených verzií argumentu. Problém akurát nastane pri príkazoch, ktoré očakávajú cestu k súboru. Ak totiž špecifikujeme medzeru ako oddeľovač príkazu (`readline.set_completer_delims()`), cesty obsahujúce medzeru budú chápané ako viacero argumentov. Preto pri spracovaní tohto druhu príkazu program používa celý riadok. Všetko za prvou medzerou je považované za cestu súboru. Jednotlivé možné doplnenia cesty k súboru sú následne skrátené tak, aby začínali poslednou, zatiaľ zadanou časťou cesty za medzerou. (Menší problém je, že keď osoba vykonávajúca konfiguráciu dva krát stlačí klávesu `Tab`, zobrazia sa orezané verzie cesty ako možnosti argumentu. Avšak všetko potrebné pre rozhodnutie sa o ďalšej časti cesty je zobrazené.)

Pre zobrazovanie farieb je použitá knižnica *colorama* [6]. Pre prehľadnosť považujeme zobrazovanie farieb za dôležité. Popis vrcholu je jasne rozoznateľný od názvu vrcholu. Pozmenené alebo chýbajúce vrcholy sú zvýraznené farebným pozadím, medzera na riadku ktorý popisuje navštívený alebo vybratý vrchol má taktiež priradenú farbu pozadia (viď časť 4.1.3).

[3]Napríklad textová konzola má pri nedostatočnom pripojení menšiu odozvu

4.2.3 Pomocné funkcie na aktualizáciu odkazov v dokumentácii

Keďže je v konfigurátore implementované načítanie grafového modelu, môžeme túto funkcionalitu použiť na ďalšie účely. Načítaný graf môžeme napríklad použiť na opravu odkazov v dokumentácii napísanej v systéme LaTeX. Dosiahneme, že príslušný súbor popisujúci grafový model umožňuje vytvorenie obrazovej reprezentácie grafu, textovej dokumentácie a profilov pre aplikácie na základe tých istých dát. Vyhneme sa tak nezhodám medzi rôznymi zdrojmi informácie spôsobenými ľudskou chybou.

Príklad programu na spracovanie textu je v súbore `fix_links.py`. Funkcia `process_latex_file` vyhľadá príslušné vrcholy podľa názvu v súbore, a upraví parametre popisujúce zoznamy hrán. Ďalšie funkcie sú pomocné.

4.2.4 Príklad skriptu na zobrazovanie dokumentácie

(Skirpt sa v elektronickej prílohe nachádza pod názvom `documentation.sh`.)

```
#!/bin/bash

node=${1//_/ }
echo $node
var=‘pdfgrep -oP "(\ \ |\n)$node, ([0-9]+)" *.pdf | grep -oP [0-9]+‘
echo $var
xreader -p $var *.pdf
```

Skript vyhľadá pomocou príkazu `pdfgrep` príslušný vrchol v registri na konci *PDF*. Následne pomocou programu *xreader* otvorí príslušnú stranu dokumentácie. Skript je možné prispôsobiť, možné je použiť zvlášť vytvorený systém dokumentácie.

Kapitola 5

Príklady použitia a vyhodnotenie

V tejto kapitole pre každú z kategórií vyberieme bežne známu a dostupnú aplikáciu. Na vybraných príkladoch demonštrujeme použitie konfigurátora a popíšeme, ktoré vrcholy sú nevyhnutné a ktoré nie. Prvá časť (5.1) je rozpísaná podrobnejšie, môže poslúžiť ako príklad na objasnenie toho, ako konfigurátor prakticky používať. Ďalšie časti obsahujú nevyhnutné minimum informácií pre praktickú demonštráciu použiteľnosti vytvorených profilov, ako aj ukážky možných problémov. Nakoľko v súčasnosti správanie aplikácií nie je dostatočne štandardizované, pri reálnom nasadení môže byť nutné ručne prispôsobiť profil alebo použitý graf konkrétnym potrebám.

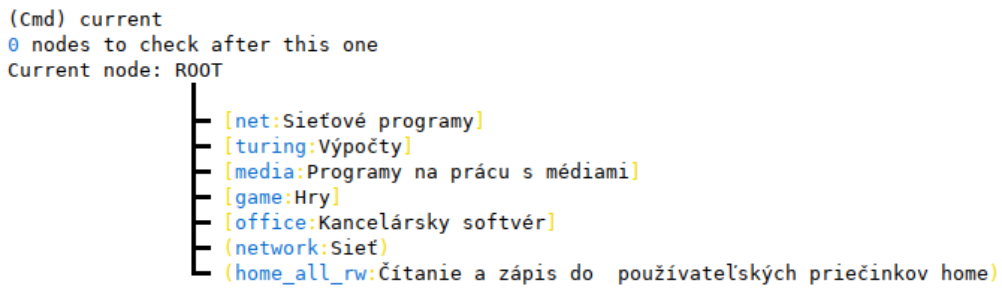
5.1 Firefox (webový prehliadač)

Ako reprezentanta webových prehliadačov použijeme prehliadač *Firefox* [7] vo verzii 68. Ide o robustný prehliadač určený aj pre bežného používateľa, poskytuje teda vhodný príklad pre účely tejto práce. Na testovacom systéme je tento prehliadač štandardne nainštalovaný. Príkazom `firefox` spúšťame program z umiestnenia `/usr/bin/firefox`. Je dôležité si všimnúť, že na tomto umiestnení sa nachádza iba symbolický odkaz na spustiteľný skript `/usr/lib/firefox/firefox.sh`. Profil je potrebné vytvárať pre tento spustiteľný skript, nie pre odkaz (viď. časť 4.1.7).

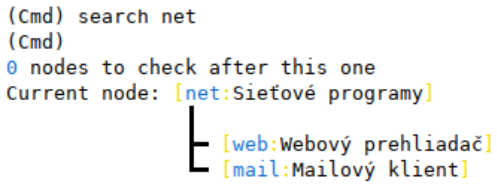
Konfigurátor spustíme nasledovným príkazom:

```
./configurator.py dag.dot /usr/lib/firefox/firefox.sh
```

Používateľa privíta príkazový riadok konfigurátora. Použitím príkazu `current` je možné zobraziť aktuálne prehľadávaný vrchol. Keďže sme ešte nepoužili žiaden z príkazov `search` a `keep`, zobrazí sa pomyselný koreň a jeho pod-vrcholy reprezentujúce kategórie aplikácií ktoré sú najvyššie v hierarchii (obrázok 5.1).



Obr. 5.1: Konfigurátor - zobrazenie koreňa a pod-vrcholov



Obr. 5.2: Konfigurátor - ukážka prehľadávania vrcholov

Intuitívne sa ponúka možnosť vybrať možnosť `net`, ktorá podľa uvedeného popisu reprezentuje sieťové programy. Záujem ponechať v profile požiadavky nachádzajúce sa pod touto kategóriou by sme mohli vyjadriť príkazom `keep net`. Avšak „sieťové programy" znie príliš všeobecne, nie je isté či chceme použiť takto všeobecný vrchol, preto použijeme príkaz `search net`. Týmto príkazom označíme záujem bližšie sa pozrieť na daný vrchol bez akejkoľvek zmeny profilu. (Príkaz `keep net` v prípade potreby môžeme použiť kedykoľvek neskôr, napríklad ak by sa ukázalo, že pod daným vrcholom sa nenachádzajú vhodnejšie pod-kategórie.)

Príkazom `next` pokračujeme v prehľadávaní. (Na obrázku 5.2 sme namiesto príkazu `next` použili prázdny riadok, čo je praktická skratka za tento príkaz. Tento príkaz je totiž používaný pomerne často.) Keďže sme zatiaľ na prehľadávanie označili iba vrchol `net`, *aktuálnym vrcholom* sa stal práve tento vrchol. *Aktuálny vrchol* je možné zobraziť príkazom `current`, avšak ako je vidno z obrázku 5.2, po použití príkazu `next` sa ďalší vrchol zobrazí automaticky. Dozvieme sa, že pod vrcholom `net` sa nachádzajú *K-vrcholy* `web` a `mail`.

Ukazuje sa, že namiesto `keep net` je vhodnejšie použiť konkrétnejší vrchol príkazom `keep web`. Po vykonaní tohto príkazu konfigurátor rekurzívne prehľadá potomkov a popridáva prípadné nájdené *A-vrcholy* do profilu. V prípade *D-vrcholov* musí rozhodnúť používateľ. Používaním príkazov `next`, `search` a `keep` prehľadávame graf za pomoci konfigurátora. Na obrázku 5.3 je možné vidieť okrem ďalšieho prehľadávaného vrcholu po príkaze `keep web` aj počet ďalších vrcholov, ktoré sa zatiaľ nachádzajú v zozname *na prehľadanie*.

```
(Cmd) keep web
(Cmd)
13 nodes to check after this one
Current node: <opt_xdgdesktop:Použiť XDG-desktop abstrakciu>
              └ (xdgdesktop:XDG desktop)
```

Obr. 5.3: Konfigurátor - pridávanie a prehľadávanie vrcholov

```
(Cmd)
Nothing left in the queue.
(Cmd) save /tmp/firefox_test.sav
(Cmd) export /tmp/firefox_profile
Enter possible names of app separated by space:
firefox .firefox mozilla .mozilla
```

Obr. 5.4: Konfigurátor - záverečná fáza vytvárania profilu

Po ukončení prehľadávania program používateľa upozorní že už vo fronte *na prehľadanie* nie sú žiadne vrcholy. Všetko je pripravené na vygenerovanie profilu. Na uloženie profilu do súboru slúži príkaz `export`, ktorý očakáva ako argument cestu súboru. Pred ukončením konfigurátora odporúčame použiť príkaz `save`, ktorý uloží aktuálny stav programu do uvedeného súboru a umožňuje neskôr vykonávať zmeny ak to bude v budúcnosti potrebné. Obrázok 5.4 zobrazuje tieto operácie. Ukazuje aj otázku na možné mená aplikácie a odpovede. Okrem variantov názvu programu je vhodné pridať aj názov organizácie, keďže niektoré programy (vrátane *Firefoxu*) používajú tento názov pre niektoré svoje priečinky. Konfigurátor sa na možné názvy aplikácie spýta len v prípade, že je to potrebné.

Použité vrcholy: Pri testovaní sme pridali nasledovné vrcholy: `web`, `xdgdesktop`, `etc_all`, `dbus_strict`, `dbus_accesibility_strict`, `dbus_session_strict`, `config_appname`, `config_kdeglobals`, `cache_appname`, `cache_kcache`, `home_appdir`, `settings`, `run_r`, `proc_r`, `dev_nvidia`, `audio`, `ibus`, `read_openssl_conf`, `nm_state`, `arp_table`, `tmpm`, `dev_r`, `dev_w`.

Prvotné testovanie bolo zamerané na funkčnosť programu, pričom niektoré vrcholy môžu byť pridané navyše. Vrcholy boli pridávané intuitívne, snahou bolo pokryť čo najviac funkcionality, avšak aj tak bolo nutné pridať nejaké vrcholy na základe testovania podľa časti 4.1.4. Pracujeme teda s nasledovným zoznamom vrcholov: `web`, `xdgdesktop`, `etc_all`, `dbus_strict`, `dbus_accesibility_strict`, `dbus_session_strict`, `config_appname`, `config_kdeglobals`, `cache_appname`, `cache_kcache`, `home_appdir`, `settings`, `run_r`, `proc_r`, `dev_nvidia`, `audio`, `ibus`, `read_openssl_conf`, `nm_state`, `arp_table`, `tmpm`, `dev_r`, `dev_w`.

Ak teda na všetky tieto vrcholy použijeme príkaz `keep`, dostaneme viac-menej použiteľný profil pre *Firefox*. Testujeme základnú funkcionalitu prehliadača, teda

otváranie používateľských webových stránok. Pri tejto konfigurácii je možné prehliadač otvoriť a bez problémov zobrazovať webové stránky. Ako test sme otvorili aj web-stránku *YouTube* a otestovali prehrávanie videa, ktoré fungovalo bez problémov (vrátane zvuku). Jediný problém je, že prehliadač nemá povolený prístup k súboru `/.config/gtk-3.0/settings.ini`, nezobrazuje sa teda správna *GTK* téma. Tento problém je možné vyriešiť ručným pridaním povolenia čítať daný súbor, alebo použitím vrcholu `gui_gnome`. (Taktiež by bolo možné použiť vrchol `config_all`, avšak táto požiadavka toho povoľuje zbytočne veľa.)

Potrebné/Nepotrebné vrcholy: Takýmto obšírnym pridávaním vrcholov dosiahneme, že program bude fungovať, avšak niektoré pridelené požiadavky môžu byť pre beh programu zbytočné, avšak potenciálne dopady na bezpečnosť zostanú zachované. Preto po jednom skúšame odstraňovať jednotlivé požiadavky, a skúšame požadovanú funkcionalitu (otvorenie web-stránky a prehranie videa).

- `web`: *K-vrchol*, ktorý je potrebný pre automatické pridanie potrebných vrcholov do profilu.

- `cache_appname`: Vrchol je potrebný pre prístup k súborom v priečinku `/.cache/mozilla`. Názov *mozilla* je potrebné pridať do zoznamu možných názvov aplikácie, ktorý si konfigurátor v požadovanom momente vypýta. Bez tohto vrcholu prehliadač vôbec nenačíta webové stránky.[1]

- `home_appdir`: V priečinku `/.mozilla/firefox/` sa nachádzajú používateľské profily programu *Firefox*. Bez prístupu k týmto profilom sa prehliadač nespustí, ale vypíše chybovú hlášku „Your Firefox profile cannot be loaded. It may be missing or inaccessible." Pripomíname, že používateľ má v tomto domovskom priečinku uložené len svoje profily, toto povolenie teda neumožňuje prístupy k profilom cudzích používateľov (ktoré sú obmedzené bežnými prístupovými oprávneniami). [2]

- `audio`: Bez pridania tohto vrcholu prehliadač funguje, avšak pri pokuse prehrať video sa neprehrá zvuk[3]. Pre domáceho používateľa je nutné tento vrchol pridať. Okrem prehrávania zvuku vrchol povoľuje aj nahrávanie zvuku. Pre použitie vo firme, kde sa prehliadač nepoužíva na mediálne účely tento vrchol nie je potrebný. Navyše, jeho zakázanie môže zabrániť potenciálnemu útočníkovi nahrávať zvuk a špehovať používateľa a jeho okolie.

[1] Nie je totiž štandardizované, či aplikácia používa svoj názov alebo názov spoločnosti ako názov priečinku.

[2] *Profily* spomenuté v tejto časti si netreba zameniť s *AppArmor* profilmi ktoré sa spomínajú v zvyšku práce.

[3] Ide o očakávané správanie.

- `dev_r`: Prehliadač sa bez daného povolenia nespustí, iba zobrazí chybovú hlášku (obrázok 5.5). Potrebný je prístup k zariadeniam v priečinku `/dev/shm/`, ktoré existujú až počas behu programu. V prípade používania proprietárnych ovládačov pre grafické karty *Nvidia* je vyžadovaný prístup na čítanie príslušných zariadení v priečinku `/dev/`.

- `dev_w`: Po pridaní práv na zápis do zariadení v priečinku */dev/shm/* prehliadač funguje. V prípade, že nevyžadujeme ručnú konkretizáciu je možné použiť priamo tento vrchol.

- `xdgdesktop`: V prípade neudelenia povolenia sú zamietnuté nesúvisiace prístupy. Samotný prehliadač pri teste funguje aj bez povolenia tohto prístupu. Samotné povolenie prístupu je však viac-menej neškodné, keďže prístup sa vzťahuje len na priečinky, ide teda skôr o technický detail.

- `etc_all`: Pri neudelení tohto povolenia sú zaznamenané a zakázané nasledovné prístupy:

  - `/etc/firefox/syspref.js r,`
  - `/etc/gtk-3.0/settings.ini,`
  - `/etc/lsb-release r,`
  - `/etc/dpkg/origins/ubuntu,`
  - `/etc/debian_version r,`
  - spustenie príkazu `apt-cache`

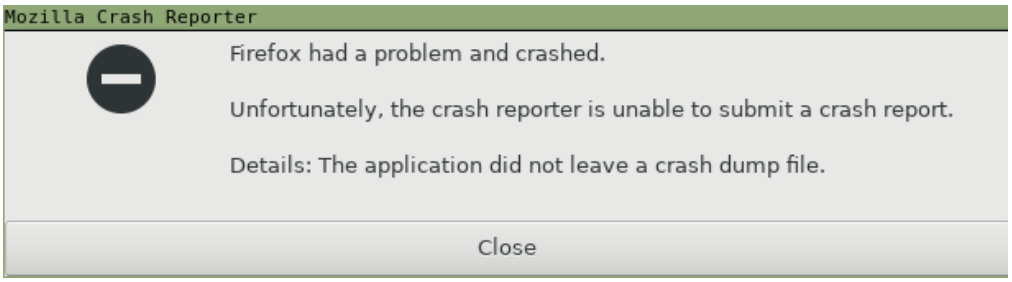
  Samotný prehliadač pri testovanom scenári funguje. Niektoré súbory môžu byť nevyužité (napríklad `syspref.js` na testovanom systéme neobsahuje nič okrem komentárov), prípadne využité obmedzene pri konkrétnych nekritických činnostiach.

- `dbus_strict`, `dbus_session_strict`, `dbus_accesibility_strict`: Program pri testovanom scenári funguje bez problémov aj pri odstránení.

- `config_appname`: Povolenie nie je nutné pridávať, nakoľko *Firefox* ukladá svoju konfiguráciu priamo do domovského priečinku.

- `config_kdeglobals`, `cache_kcache`: Program nebol testovaný na operačnom systéme s nainštalovaným prostredím *KDE*, ani nebol napísaný špeciálne pre toto prostredie.

- `ibus`, `read_openssl_conf`, `nm_state`, `arp_table`: Testovaný scenár funguje bez akýchkoľvek problémov, dokonca v logu nie sú žiadne súvisiace zakázané



Obr. 5.5: Ukážka zlyhania prehliadača Firefox

prístupy. `tmpm`: *Firefox* sa pokúša o daný prístup, avšak testovaný scenár funguje bez problémov. Napriek tomu, odporúčame toto povolenie udeliť, keďže testovaný scenár je obmedzený na nutné minimum.

- `settings`: V prípade neudelenia povolenia sú zakázané prístupy do súborov `/home/peter/.config/dconf/user` a `/run/user/1000/dconf/user`, avšak samotný prehliadač pri testovanom scenári funguje. Pridanie tohto vrcholu druhý menovaný prístup nepovolí. Pokiaľ to nie je potrebné, odporúčame toto povolenie neprideľovať.
- `run_r`, `dev_nvidia`: Prehliadač napriek niektorým zablokovaným prístupom funguje bez problémov. Nie sú principiálne nevyhnutné dôvody prideľovať tieto povolenia, preto odporúčame prideliť ich iba v špeciálnych prípadoch, ak sa to ukáže nevyhnutné.
- `proc_r`: *Firefox* sa snaží čítať viacero informácií o procesoch, avšak funguje aj bez tohto povolenia. (Napríklad, snaží sa čítať súbor `mountinfo`, problémy má príkaz `nvidia-modprobe` ktorý zdedí profil, . . .)

Overenie blokovaného prístupu: Za otestovanie stojí aj pokus stiahnuť súbor do priečinku, kam sťahovanie nie je povolené. Užívateľ by mohol napríklad omylom prepísať konfiguračné súbory iných programov. Ak by používateľ stiahol útočníkov súbor s názvom `.bashrc`, a bez rozmýšľania ho uložil do domovského priečinku, útočník takto môže získať prístup do systému pri ďalšom spustení terminálu. Pri pokuse stiahnuť súbor na nepovolené miesto nevyskočí žiadna chybová hláška, avšak ikonka sťahovania obsahuje varovný kruh a sťahovanie je v zozname uvedené ako zlyhané. *Firefox* funguje bez zlyhania ďalej, situácia je v programe korektným (a používateľsky prívetivým) spôsobom ošetrená.

5.2 SuperTuxKart (hra)

SuperTuxKart je závodná hra s 3D grafikou s otvoreným zdrojovým kódom. Táto hra teda využíva všetky základné prvky hier (audio, renderovanie 3D grafiky pomocou grafickej karty, čítanie vstupu, ...).

Test 1: Pri prvotnom teste boli povolené nasledovné vrcholy: `game`, `xdgdesktop`, `gui_gnome`, `config_appname` a `cache_appname`. Po spustení hry[4] niekoľko krát zhasla obrazovka a zmenilo sa rozlíšenie obrazovky, avšak hra sa nakoniec nespustila. Z logov vyplýva, že boli zakázané viaceré prístupy, napríklad zariadenia v `/sys/devices/`, čítanie súboru `/proc/modules` (zoznam modulov zavedených do kernelu), rôzne súbory v priečinku `/etc/modprobe.d` čítané programom `modprobe` a priečinok `/.local/share/supertuxkart`.

Test 2: Pri prvom teste sa ukázalo ako potrebné pridať ďalšie vrcholy. Pridané boli nasledovné povolenia: `etc_all`, `run_r`, `proc_r`, `dev_r` a `dev_w`. Okrem týchto vrcholov sa ukázalo ako potrebné pridať do grafu vrchol `opt_local` a príslušné pod-vrcholy. Pre účely tohto testu sme pridali do profilu vrchol `local_appname`. Hru je následne možné spustiť a hrať.

5.3 Výpočty

Na test používame jednoduchý program napísaný v jazyku *C++*, ktorý spočíta dve čísla. Stačí povoliť vrchol `turing`, a program bez problémov funguje (vrátane použitia dynamicky linkovanej knižnice).

5.4 Thunderbird (mailový klient)

Thunderbird je e-mailový klient s grafickým rozhraním od spoločnosti *Mozilla*. Na test sme pridelili nasledovné vrcholy: `mail`, `xdgdesktop`, `gui_gnome`, `etc_all`, `config_appname`, `cache_appname`, `home_appdir`, `proc_r`, `run_r`, `dev_r` a `dev_w`. Ako názov aplikácie je potrebné uviesť `thunderbird` aj `.thunderbird`, nakoľko v domovskom priečinku a v priečinku `cache` program používa rôzne názvy priečinkov.

[4] skript `run_game.sh`

5.5 LibreOffice Writer (kancelársky softvér, zobrazovač dokumentov)

Keďže kancelársky softvér zahŕňa všetky požiadavky zobrazovača dokumentov, uvedieme len jeden príklad pre obe tieto kategórie. (Ako príklad pre zobrazovač dokumentov ktorý nepatrí do kategórie kancelárskeho softvéru by sme mohli použiť napríklad program *Okular*, ktorý dobre poslúžil pri malých pokusoch s *AppArmorom*.) *LibreOffice* je voľne dostupný balík kancelárskych programov. *Writer* slúži na úpravu formátovaných textových dokumentov. Pridáme skupinu vrcholov: `office`, `xdgdesktop`, `gui_gnome`, `etc_all`, `config_appname`, `settings`. Názov aplikácie: `libreoffice`[5]. Editor sa podarí spustiť a použiť na vytvorenie súboru v priečinku Dokumenty. Do ostatných priečinkov sa nedá zapisovať. Pri pokuse zapísať do priečinku kde program nemá povolenie vyskočí chybová hláška, avšak program naďalej funguje. Z neznámeho dôvodu chýba horné menu programu. Pri pokuse odstrániť tento problém ručným pridávaním riadkov do profilu sa podarilo docieliť stav, že *AppArmor* do logu už nevypisuje žiadne zakázané prístupy, avšak problém s nezobrazeným menu pretrváva.[6] Odhalenie konkrétnej príčiny by si vyžadovalo podrobnejšie skúmanie, prípadne podrobnú znalosť zdrojového kódu *LibreOffice*. V prípade, že program nie je spustený príkazom `lowriter`, ale príkazom `libreoffice` a vytvorením nového dokumentu kliknutím na ikonku v menu, tak sa problém nevyskytuje, beží totiž pod profilom `libreoffice-soffice` nachádzajúcim sa v súbore `/etc/apparmor.d/usr.lib.libreoffice.program.soffice.bin`, ktorý je do systému nainštalovaný z balíčka `libreoffice-common`. Znamená to, že sa dá vytvoriť prispôsobený profil ktorý funguje.

Poznámka: V tomto prípade ide len o testovací príklad, za bežných okolností je vhodné (ak je to vyhovujúce) použiť existujúci profil.

5.6 Blender (program na prácu s médiami)

Testujeme program vo verzii *2.79*, keďže podstatne zmenená verzia *2.80* vyšla vo finálnej podobe až tesne pred odovzdaním práce. Program je primárne určený na tvorbu 2D/3D grafického obsahu, strih videa a vytváranie vizuálnych efektov. Ide o rozsiahly program. Okrem iného, má zabudovaný interpreter jazyka *Python 3*, čo môže do značnej miery narušiť bezpečnosť. (Avšak štandardne je pre neznáme súbory spúšťanie skriptov zakázané.)

[5] priečinok `.config/libreoffice`

[6] Samozrejme, keď je profil zablokovaný, menu sa zobrazuje korektne. V opačnom prípade by bol pôvod problému zrejmý.

Štandardne, najprv pridáme vrcholy, ktoré sú očividné alebo relatívne bezpečné: `media`, `xdgdesktop`, `gui_gnome`, `audio` a `config_appname`. V prípade, že *Blender* nemá všetky potrebné povolenia, vypíše do konzoly stručnú chybovú hlášku a skončí. Podľa logov *AppArmoru* pridáme ďalšie vrcholy `etc_all`, `proc_r` a `dev_nvidia`. Za povšimnutie stojí vrchol `dev_nvidia`, ktorý tu nájde reálne uplatnenie. Aby fungovalo renderovanie na grafickej karte podporujúcej *Cuda*, musíme povoliť komunikáciu prostredníctvom *Unixových socketov* (`net_unix`). Nevieme povoliť konkrétnu adresu druhej strany, keďže tá obsahuje neznámu numerickú hodnotu, preto používame všeobecný vrchol.

Kapitola 6

Záver

V práci sme navrhli grafový model pre formálny popis požiadaviek kategórií aplikácií. Uviedli sme aj konkrétnu inštanciu grafu, ktorá môže poslúžiť ako základ pre ďalšie rozširovanie a použitie. V súčastnosti praktickému použitiu bráni nedostatočná štandardizácia vývoja aplikácií. Napriek tomu sa nám podarilo vytvoriť inštanciu grafu, ktorá je aspoň čiastočne použiteľná aj pre reálne používateľské aplikácie. Okrem popisu návrhu modelu a konkrétneho grafu sme vytvorili aj program, ktorý zjednoduší vytváranie profilu pre aplikácie.

Budúce práce sa môžu zamerať na zlepšenie používateľského zážitku, napríklad vytvorením grafickej verzie konfigurátora, ktorá by umožňovala jednoduchšie čítanie dokumentácie, rekurzívne porovnávanie vrcholov z grafu a uloženého stavu predošlého sedenia, určovanie dôležitosti vrcholu pomocou strojového učenia na základe predošlých profilov atď. Voľná definícia jazyka na popis grafov umožňuje do príslušného súboru pridať ďalší obsah, ktorý môže grafický konfigurátor používať na zlepšenie používateľského zážitku. Takisto je tu priestor pre ďalšie rozširovanie referenčného grafu alebo vytvorenie prispôsobených grafov konkrétnym scenárom použitia, ktoré by vyžadovali menej otázok na používateľa. Poslednou dôležitou oblasťou je tvorba štandardov pre písanie programov, ktoré by následne podstatne zjednodušili vytváranie všeobecne použiteľných grafov.

Neprinášame univerzálne riešenie ktoré vyrieši všetky problémy (také ani neexistuje), ale veríme, že táto práca a elektronické prílohy nájdu svoje uplatnenie pri zabezpečení používateľských počítačov, a svojim dielom pomôžu minimalizovať bezpečnostné riziká s ktorými sa denno-denne stretávame.

Register

Literatúra

[1] About. [Online], [cit. 2018-11-25].
URL https://gitlab.com/apparmor/apparmor/wikis/About

[2] AppArmor Documentation. [Online], [cit. 2018-11-25].
URL https://gitlab.com/apparmor/apparmor/wikis/Documentation

[3] Apparmor Wiki. [Online], [cit. 2018-11-25].
URL https://gitlab.com/apparmor/apparmor/wikis/home

[4] *APPARMOR.D(5) AppArmor.*

[5] *CAPABILITIES(7) Linux Programmer's Manual.*

[6] Colorama. [Online], [cit. 2019-07-31].
URL https://pypi.org/project/colorama/

[7] Firefox - stránka sťahovania. [Online], [cit. 2019-07-15].
URL https://www.mozilla.org/sk/firefox/new/

[8] Graphviz - Graph Visualization Software. [Online], [cit. 2019-05-04].
URL https://graphviz.org/

[9] Linux 2.6.36. [Online], [cit. 2019-05-04].
URL https://kernelnewbies.org/Linux_2_6_36

[10] NetworkManager D-Bus API Types. [Online], [cit. 2019-04-22].
URL https://developer.gnome.org/NetworkManager/stable/nm-dbus-types.html

[11] New In Buster. [Online], [cit. 2019-05-04].
URL https://wiki.debian.org/NewInBuster

[12] openSUSE Support Database: AppArmor. [Online], [cit. 2019-05-04].
URL https://en.opensuse.org/SDB:AppArmor

[13] Pydot. [Online], [cit. 2019-07-20].
URL https://pypi.org/project/pydot/

[14] Pydotplus. [Online], [cit. 2019-07-20].
URL https://pypi.org/project/pydotplus/

[15] Python Left-Right Parser. [Online], [cit. 2019-07-20].
URL https://github.com/transceptor-technology/pyleri

[16] A quick guide to AppArmor profile Language. [Online], [cit. 2019-04-29].
URL https://gitlab.com/apparmor/apparmor/wikis/QuickProfileLanguage

[17] SELinux Project Wiki. [Online], [cit. 2019-05-04].
URL http://selinuxproject.org/page/Main_Page

[18] XDG Base Directory Specification. [Online], [cit. 2019-04-26].
URL https://specifications.freedesktop.org/basedir-spec/basedir-spec-latest.html

[19] HROMCOVÁ, Z.: *A Secure Linux Desktop Environment*. Diplomová práca, FMFI UK, 2019.