**Vysoká škola ekonomická v Praze**
**Fakulta informatiky a statistiky**

Katedra informačního a znalostního inženýrství
Obor znalostní technologie

**Diplomová práce**

# Bezpečnostní politiky SELinuxu pro vybrané aplikace prostředí KDE

**Vypracoval: Bc. Ondřej Vadinský**
**Vedoucí práce: RNDr. Radomír Palovský, CSc.**

**Praha, květen 2011**

Vysázeno systémem XeTeX

## Prohlášení

Prohlašuji, že jsem samostatně vypracoval diplomovou práci na téma *Bezpečnostní politiky SELinuxu pro vybrané aplikace prostředí KDE*. Použitou literaturu a další podkladové materiály uvádím v přiloženém seznamu literatury.

V Praze dne 2. května 2011 . . . . . .
podpis

# Poděkování

Na tomto místě bych rád poděkoval RNDr. Radomíru Palovskému, CSc. za vedení diplomové práce a za jeho rady a připomínky, které mi byly přínosem při jejím zpracování.

Dále bych také rád poděkoval lidem z poštovní konference *Referenční politiky*, zejména Dominicku Griftovi, za poskytnuté konzultace ohledně vývoje *bezpečnostních politik SELinuxu*.

Nesmím opomenout poděkovat ani všem, kteří mi pomáhali s korekturou textu této práce, především Mgr. Olze Bičišťové za velkou trpělivost s mou nedokonalou češtinou.

Konečně také děkuji vývojářům svobodných nástrojů [1], které jsem použil k vytvoření práce samotné a vývoji vzorové *bezpečnostní politiky*.

[1] Při zpracování této diplomové práce jsem mimo jiné použil následující svobodný software: typografický systém *XeLaTeX*, textový editor *VIM*, vektorový editor *Inkscape* a systém správy verzí *Git* v distribucích *Arch Linux* a *Fedora* s pracovním prostředím *KDE*. Pro obsažená schémata jsem využil svobodné schéma barev *Solarized* (Domovská stránka: `http://ethanschoonover.com/solarized`).

## Abstrakt

Tato práce zkoumá *technologie tvorby bezpečnostních politik SELinuxu*. Dále práce analyzuje *uživatelský prostor* systému *GNU/Linux* se zvláštním zaměřením na *pracovní prostředí KDE*. Na základě této analýzy pak práce navrhuje metodiku tvorby *bezpečnostní politiky* „zdola nahoru“. Nabyté znalosti se pak v práci uplatňují při realizaci jejího hlavního cíle, totiž tvorbě *bezpečnostní politiky* pro vybrané aplikace *prostředí KDE*.

Při popisu technologie tvorby *bezpečnostních politik* práce čerpá informace z dostupných zdrojů. Vstupem do analýzy *uživatelského prostoru* jsou poté jednak dostupné elektronické zdroje a také vlastní zkoumání konkrétních aplikací. To spolu s obecnými filosofickými principy pak slouží k vytvoření metodiky pro tvorbu politiky „zdola nahoru“. Následný vývoj *bezpečnostní politiky* vychází ze stanovených *bezpečnostních cílů*, nabytých poznatků, vytvořené metodiky a definovaných případů užití.

Přínosem práce je v teoretické rovině navrhovaná metodika tvorby *bezpečnostních politik* pro *uživatelský prostor*. V praktické rovině pak práce přináší vzorovou *bezpečnostní politiku SELinuxu* pro vybrané aplikace.

Struktura práce odpovídá jednotlivým cílům. Pro ty jsou vyčleněny tři části odlišující rešerši dostupných zdrojů, vlastní teoretické úvahy a vlastní praktický výstup práce. Jednotlivé části práce se dále člení podle potřeb řešeného tématu.

## Klíčová slova

*SELinux*, *povinné řízení přístupu*, *bezpečnostní politika*, *uživatelský prostor*, *pracovní prostředí KDE*.

## Abstract

This thesis deals with technologies of *SELinux security policy* writing. Furthermore the thesis analyzes *userspace* of *GNU/Linux* operating system with special focus on *KDE desktop environment*. On the basis of this analysis a bottom-up methodics to create a *security policy* is devised. Acquired knowledge is then used in practice when realizing the main goal of the thesis, which is to create example *security policies* for chosen *KDE applications*.

When describing technologies of *security policy* writing the thesis draws information from available sources of information. Input for *userspace* analysis are available electronic sources of information and author's own experience with analyzed applications. This is used with common philosophic principles to devise bottom-up methodics of policy writting. Following act of policy building draws from defined *security goals*, acquired knowledge, created methodics and defined usecases.

Theoretical contribution of the thesis is devised methodics of *userspace* policy building. Main practical contribution are then created example *SELinux policies* for chosen *KDE applications*.

The structure of the thesis follows its goals. For them three parts are created: background research of available resources, author's own theoretical contemplations and practical output of the thesis. Those parts are then devided into sections according to needs of each topic.

## Keywords

*SELinux*, *Mandatory Access Control*, *Security Policy*, *Userspace*, *KDE Desktop Environment*.

# Obsah

x

# Úvod

Tématem této diplomové práce je *tvorba bezpečnostních politik SELinuxu pro vybrané aplikace pracovního prostředí KDE*. *Security Enhanced Linux*, neboli *SELinux*, rozšiřuje operační systém *GNU/Linux* o podporu *povinného řízení přístupu*, avšak většina jeho *bezpečnostních politik* se zaměřuje na serverové nasazení a služby systému. Jako dlouhodobý uživatel *GNU/Linuxu* jsem se rozhodl směřovat své *bezpečnostní politiky* na uzavření *uživatelského prostoru*, protože zde spatřuji největší nedostatky současného nasazení *SELinuxu*. Jak roste zastoupení *GNU/Linuxu* na pracovních stanicích, stává se zajištění bezpečnosti *uživatelských dat* stále zásadnějším problémem. *SELinux* vidím jako jednu z možností, jak vyšší bezpečnosti dosáhnout.

V návaznosti na předchozí bakalářskou práci směřuje tato diplomová práce k uzavření aplikací *uživatelského prostoru*, a sice *pracovního prostředí KDE*. Důvody, proč se o něco takového snažit, vidím především v zajištění integrity *uživatelských dat*. Tato mohou být ohrožena zejména chybami v aplikacích, se kterými uživatel pracuje. Za nejvíce rizikové považuji aplikace přistupující k síti, protože ta dále rozšiřuje možnosti zneužití chyb v aplikacích. Toto považuji za velmi dobrý dlouhodobý cíl svého snažení, k jehož řešení by měla přispět i tato diplomová práce. V rámci ní nejprve prodiskutuji možné přístupy k vývoji *bezpečnostní politiky* pro aplikace *uživatelského prostoru*. Přitom vysvětlím, proč jsem se rozhodl vyvíjet tuto politiku od koncových aplikací směrem k pracovnímu prostředí a *uživatelskému prostoru*. Toto ovlivnilo i volby konkrétních cílů práce.

Mezi cíle této diplomové práce patří: prozkoumání technologie tvorby *bezpečnostních politik SELinuxu* se zaměřením na *uživatelský prostor*, analýza vybrané části *uživatelského prostoru* a navržení metody pro jeho uzavření a konečně vytvoření, otestování a zdokumentování *bezpečnostních politik* pro typové situace. Jako hlavní cíl své práce vidím vytvoření *bezpečnostních politik* pro vybrané *KDE aplikace* přistupující k síti. Konkrétně jde o poštovní klient *KMail*, s ním úzce spolupracující správce adresáře *KAddressBook* a webový prohlížeč *Konqueror* [2]. Tím chci ochránit *uživatelská data* před následky chyb v těchto aplikacích a jejich vzdáleným zneužitím. Uzavření zvolených aplikací vyžaduje také vytvoření *bezpečnostních politik* pro *služby KDE*, které aplikace využívají. Především jde o správu zdrojů *PIM* dat *Akonadi* a úschovnu hesel *KWallet*. V neposlední řadě je také třeba dále rozvinout základ politiky pro samotné *KDE* [3] a případně upravit některé moduly *Referenční politiky*.

Při zkoumání technologií tvorby *bezpečnostních politik* čerpám především ze studia literatury a na internetu dostupných zdrojů o této problematice. K analýze *uživatelského prostoru* využiji především dokumentaci *prostředí KDE*. Tuto analýzu

[2] Základ *bezpečnostní politiky* pro *Konqueror* jsem vyvinul v rámci své bakalářské práce, soustředím se tedy na změny provedené od té doby.

[3] Základ *bezpečnostní politiky* pro *KDE* jsem vyvinul v rámci své bakalářské práce, i zde se tedy soustředím pouze na změny provedené od té doby.

spolu s obecnějšími poznatky pak uplatním při tvorbě vhodné strategie k uzavření vybraných aplikací *uživatelského prostoru*, a sice směrem od koncových aplikací pracovního prostředí přes jeho služby k obecným službám *uživatelského prostoru*. Tyto teoretické poznatky a návrhy pak uplatním při vlastním vývoji *bezpečnostních politik* pro vybrané *aplikace KDE*. K tomu využiji známou metodu spočívající ve vytvoření základní kostry *bezpečnostní politiky*, jejím zavedení do systému a procesu zkoušení funkčnosti aplikace provázaném s rozšiřováním vyvíjené politiky. Tuto metodu dále rozšířím o testování spolupráce s dalšími uzavíranými aplikacemi, čímž bych rád dosáhl stavu kompletnosti a funkčnosti odpovídajícího daným časovým omezením této práce.

V této práci předpokládám u čtenářů základní znalosti systému *GNU/Linux* a *počítačové bezpečnosti*. Dále pak očekávám podrobnější znalost *SELinuxu* a principů jeho fungování odpovídající úrovni přiblížené mou bakalářskou prací. Tato práce tak není manuálem *SELinuxu*, ale spíše dokumentací mé snahy o vytvoření *bezpečnostních politik* pro vybrané aplikace. Ač práce nezachází do detailů *SELinuxu*, lze tyto dohledat v uváděných zdrojích. Stejně tak práce uvádí domovské stránky zmiňovaných projektů, které se doporučují navštívit pro získání dalších informací. Tato omezení plynou jednak z doporučeného rozsahu práce a jednak z náročnosti vlastního tématu práce.

Struktura práce odpovídá výše stanoveným cílům. První část práce pojednává o technologiích tvorby *bezpečnostních politik SELinuxu*. Toto dílčí téma je nejprve zasazuje do širšího kontextu bezpečnosti počítačových systémů, dále stručně shrnuje vybrané principy a termíny *SELinuxu* a také se zaměřuje na vlastní technologie tvorby *bezpečnostních politik SELinuxu*. Druhá část práce zkoumá vybranou část *uživatelského prostoru*, tedy *aplikace KDE* a způsoby, jakými tyto spolupracují. Dále se v této části čtenáři podávají jednotlivé přístupy k uzavírání *uživatelského prostoru*, tedy především vývoj politiky „shora dolů“ a „zdola nahoru“. Zde uvádím také důvody, proč jsem se rozhodl k uzavírání *uživatelského prostoru* metodou „zdola nahoru“, tedy od koncových aplikací. Ve třetí části práce dokumentuji důležité prvky *bezpečnostních politik*, které jsem vytvořil. Jejich zdrojové kódy uvádím v přílohách na konci práce a na CD, kde se také nachází elektronická verze této práce.

Pro lepší srozumitelnost práce používám následující jednotný způsob vyznačování. Názvy programů (*KMail*) a projektů (*KDE*) vyznačuji kurzívou. Tu také používám pro termíny (*uživatelský prostor*) a názvy článků či knih (*Oranžová kniha*). Jména jejich autorů či jiných osob (Grift) zvýrazňuji kapitálkami. Názvy příkazů (`ls`), SELinuxových typů či modulů (`kde_home_t`, `unconfined`) a linuxové cesty (`/usr/bin/kmail`) sázím písmem s pevnou šířkou. Obdobnou úpravu mají URL adresy (`http://selinuxproject.org/`), které se navíc zalamují podle specifických pravidel a pro elektronickou verzi práce jsou spolu s odkazy uvnitř práce vyznačeny modrou barvou. V práci použité zkratky vyznačuji kurzívou (*TE*) a dále je uvádím a vysvětluji v samostatném seznamu zkratek na straně 74.

Část I

# Technologie tvorby bezpečnostních politik

Tato část práce zasazuje tvorbu *bezpečnostních politik* do širšího kontextu *počítačové bezpečnosti*. Jedním ze způsobů realizace *počítačové bezpečnosti* v linuxových systémech je pak *SELinux*. Tato část práce tedy připomíná důležité termíny a principy *SELinuxu*. Celý *SELinux* však do značné míry stojí na *bezpečnostních politikách* pro jednotlivé aplikace a části systému. Právě na technologii jejich tvorby klade tato část největší důraz.

## 1 Potřeba počítačové bezpečnosti

Máme-li hovořit o potřebě *počítačové bezpečnosti*, musíme si nejprve říci, co to *bezpečnost počítačových systémů* vůbec je. Dále je třeba pohovořit o tom, proč se něčím takovým zabývat, a konečně pak shrnout možnosti zabezpečení počítačových systémů. Vzhledem k pojetí této práce se zde zaměřím na linuxové systémy.

### 1.1 Bezpečnost počítačových systémů

Pojem *počítačová bezpečnost* byl definován v mnoha uznávaných standardech. Příkladem může být tzv. *Oranžová kniha* amerického ministerstva obrany. [4] *Bezpečný systém* podle ní pak musí řídit přístup k informacím tak, aby pouze řádně autorizovaní jedinci či procesy běžící v jejich zastoupení měli možnost tyto informace číst, zapisovat, vytvářet a mazat. Viz [1].

Na *bezpečný systém* pak *Oranžová kniha* klade šest požadavků: existenci explicitní *bezpečnostní politiky*, přiřazení přístupových práv ke všem objektům v systému, identifikaci jednotlivých subjektů, možnost auditu pro vysledování porušení bezpečnosti, zajištění nezávislého vyhodnocení úrovně bezpečnosti a stálou ochranu před nepovolenou změnou systému. Viz [1].

Na základě míry splnění výše uvedených požadavků pak *Oranžová kniha* definuje čtyři úrovně bezpečnosti:

[4] Zde je nutno poznamenat, že jde o dokument z 80. let 20. století, který samozřejmě reflektuje svou dobu. Je tedy zaměřen na nesíťové značně monolytické systémy, takže je částečně nerelevantní pro dnešní stav. Definuje však některé základní pojmy oblasti *počítačové bezpečnosti* a na něm založené třídění bezpečných systémů se v upravené podobě používá dodnes.

D – minimální bezpečnost,

C – oddělovaná bezpečnost,

B – vynucovaná bezpečnost,

A – dokázaná bezpečnost.

Tyto kategorie jsou rozděleny do dílčích podkategorií – je tak celkem sedm úrovní bezpečnosti. Mezi kategoriemi se pak výrazně mění míra důvěry, kterou je možno takovému systému přisoudit. Viz [1].

Nyní platná mezinárodní norma *Common Criteria* definuje sedm úrovní ověření bezpečnosti *EAL1 – EAL7*, které přibližně odpovídají úrovním a jejich podkategoriím definovaným v *Oranžové knize*. Tyto úrovně vychází z hodnocení bezpečnostních požadavků a implementačně nezávislých profilů ochrany. Viz [2].

Jiným příkladem může být definice počítačové bezpečnosti iniciativy *OSA* [5]: Bezpečnost poskytovanou informačními systémy lze definovat jako jeho schopnost ochránit důvěrnost a integritu zpracovávaných dat, poskytnout dostupnost systému i dat, zajistit odpovědnost za prováděné operace a zaručit fungování systému odpovídající jeho návrhu. Viz [3].

## 1.2 Potřeba počítačové bezpečnosti

V posledních desetiletích vstoupily počítačové systémy výraznou měrou do našich životů. Veřejné instituce, soukromé firmy i jednotlivci jim svěřují svá data, která jsou v mnoha případech citlivé povahy. Zároveň je trendem značná propojenost těchto systémů. Toto se děje jak uvnitř, tak i vně institucí. Díky internetu jsou pak systémy propojovány celosvětově.

Jak roste naše závislost na propojených počítačových systémech, roste i potenciální ztráta, které čelíme, pokud by byly tyto systémy kompromitovány. Na to, že je potřeba data a počítačové systémy důkladně chránit, upozorňuje již v roce 1998 článek *Nevyhnutelnost chyby* [4]. Podle článku nelze předpokládat, že bezpečnost mohou přiměřeně zajistit samotné aplikace, aniž by se jim poskytly odpovídající funkce operačních systémů. Východiskem pak má být zavedení *vynucované bezpečnosti* do operačních systémů. Viz [4].

Potřebu zajímat se a řešit *počítačovou bezpečnost* podtrhují i statistiky zpracovávající *hlášení CVE* [6]. Do roku 2009 bylo publikováno téměř 40 000 bezpečnostních chyb v různém softwaru. Trendy zachycené statistikami ukazují na setrvalý pozvolný pokles výskytu známých chyb jako přetečení zásobníku či navýšení práv.

[5] *Open Security Architecture* – Otevřená bezpečnostní architektura.

[6] *CVE* neboli „Common Vulnerabilities and Exposures“ je rejstřík identifikátorů veřejně známých bezpečnostních chyb softwaru. Domovská stránka projektu *CVE*: `http://cve.mitre.org/index.html`.

Oproti tomu se ukazuje nárůst chyb souvisejících s webovými aplikacemi a s tím související vysoké zastoupení chyb týkajících se *PHP*, skriptování a *SQL*. Postupně se také zvyšuje diverzita zveřejňovaných chyb a množství chyb souvisejících s aplikacemi a aplikačními servery. Viz [5, 6].

Pro řešení *počítačové bezpečnosti* je také důležitý trend zvyšující se modularity operačních systémů i samotných aplikací, který diverzifikuje jednotlivé počítačové systémy. Celý ekosystém se tak stává nepřehlednější, což mimo jiné komplikuje jeho ochranu ale také testování vyvíjených aplikací a operačních systémů.

## 1.3 Bezpečnost a linuxové systémy

Bezpečnost je na úrovni operačního systému zajišťována takzvaným *řízením přístupu*. V případě *GNU/Linuxu* vychází *řízení přístupu* z návrhu *diskrétního řízení přístupu* označovaného také jako *DAC* [7]. Linuxový systém rozlišuje tři typy uživatelů: vlastníka objektu, skupinu vlastníka objektu a ostatní. Pro tyto tři subjekty pak lze nastavit přístupová práva k objektu a sice pro čtení, zápis a spuštění – v případě adresářů jde o výpis obsahu, změnu obsahu a vstup do adresáře. Pomocí dalších mechanizmů je pak možné ještě dosáhnout spuštění souboru jako vlastník, odstranění jen jako vlastník a vytvoření s jinou skupinou. Uživatel `root` pak může libovolně měnit jakákoliv práva. Viz [8].

*Diskrétní řízení přístupu* v *GNU/Linuxu* je přehledné a snadno použitelné, ale neumožňuje dostatečně jemné nastavení práv a ani není ze své podstaty vynucováním *bezpečnostní politiky*. V průběhu doby vzniklo několik způsobů, jak tyto nedostatky odstranit. Jedním z nich, který adresuje nemožnost jemného nastavení práv, je *ACL* [8]. Viditelným výsledkem použití *ACL* je možnost nastavit práva čtení, zápisu a spuštění zvlášť pro jednotlivé uživatele. Viz [7]. Mechanizmus, jakým se toto provádí, podrobně popisuje zdroj [7].

Pro uložení *ACL* informací se v *GNU/Linuxu* používá mechanizmus *rozšířených atributů* [9]. *Rozšířené atributy* jsou páry identifikátoru a hodnoty trvale přiřazené objektu v souborovém systému. Jako takové pak musí být *rozšířené atributy* podporované na úrovni souborového systému. Integrace *rozšířených atributů* do systému *GNU/Linux* umožnila jeho rozšíření o další způsoby *řízení přístupu*, především pak *povinné řízení přístupu*. Viz [7].

Úplné vyřešení nedostatků základních linuxových práv však nabízí až některý ze *systémů povinného řízení přístupu*, označovaného jako *MAC* [10]. Mezi systémy,

[7] *Diskrétní řízení přístupu (DAC)* určuje přístup k objektům na základě identity subjektu případně příslušnosti subjektu ke skupině. Diskrétnost zde vyplývá z faktu, že subjekt může svá přístupová práva přenést na jiný subjekt. Viz [1].

[8] *ACL*, neboli *Access Control Lists*, jsou seznamy pro *řízení přístupu* umožňující jemnější nastavení přístupových práv v *GNU/Linuxu*.

[9] V originále: „Extended Attributes“.

[10] *Povinné řízení přístupu (MAC)* povoluje přístup subjektů k objektům na základě závazné *bezpečnostní politiky*, kterou určuje bezpečnostní administrátor. Uživatel tedy nemůže předávat dále svá

které rozšiřují *GNU/Linux* o *povinné řízení přístupu*, patří například:

- *SELinux* [11] – Security Enhanced Linux,
- *AppArmor* [12],
- *LIDS* [13] – Linux Intrusion Detection System,
- *RSBAC* [14] – Rule Set Based Access Control,
- *grsecurity* [15],
- *TOMOYO Linux* [16].

Jednotlivé *systémy povinného řízení přístupu* mohou být do linuxového jádra začleněny prostřednictvím architektury bezpečnostních modulů *LSM*. Architektura *LSM* umožňuje vložit do jádra bezpečnostní modul, který upřesní rozhodnutí výchozího *diskrétního řízení přístupu*. Jedná se o obecnou architekturu, kterou může implementovat více různých modulů. Z výše uváděných jde o *SELinux*, *AppArmor*, *LIDS* a *TOMOYO Linux*. Bezpečnostní modul je pak jádrem konzultován v případě, že výchozí *DAC* umožní přístup k objektu. Podle toho, jakou odpověď bezpečnostní modul poskytne, jádro přístup k objektu povolí, či zakáže. Viz [9].

Vývoj architektury *LSM* byl značně provázaný s vývojem *SELinuxu*, který poskytuje komplexní bezpečnostní řešení. Nepočítalo se tedy s provozem více bezpečnostních modulů naráz, což způsobilo, že některá specializovanější řešení tuto architekturu dodnes nevyužívají. Z výše zmíněných jde o *RSBAC* a *grsecurity*.

## 2 SELinux jako způsob řešení bezpečnosti v GNU/Linuxu

*Security Enhanced Linux* je technologie pro zabezpečení *GNU/Linuxu* podložená dlouhodobým výzkumem v oblasti bezpečných systémů. Jde o implementaci mechanizmu *povinného řízení přístupu* v podobě *vynucování typů (TE)*, které lze dále doplnit o *řízení přístupu založené na rolích (RBAC)*, *víceúrovňovou bezpečnost (MLS)* a *bezpečnost založenou na kategoriích (MCS)*. Viz [10].

Aby bylo možné výše uvedené principy realizovat, musí *SELinux* přiřadit objektům i subjektům takzvaný *bezpečnostní kontext*. Vlastní zabezpečení systému se pak realizuje prostřednictvím pravidel *SELinuxu*. Tato pravidla se obvykle sdružují

práva. Viz [1].

[11] Domovská stránka projektu *SELinux*: http://selinuxproject.org/page/Main_Page.

[12] Domovskou stránku projektu *AppArmor* lze nalézt na: https://apparmor.wiki.kernel.org/index.php/Main_Page.

[13] Domovská stránka projektu *LIDS*: http://www.lids.org/.

[14] Domovská stránka projektu *RSBAC*: http://grsecurity.net/.

[15] Domovská stránka projektu *grsecurity*: http://grsecurity.net/.

[16] Domovská stránka projektu *TOMOYO Linux*: http://tomoyo.sourceforge.jp/.

do *bezpečnostní politiky*. Takto lze pomocí *SELinuxu* chránit systém i *uživatelská data* před chybami v aplikacích i před síťovými útoky z vnějšku. Viz [10].

O výše uvedených principech a prvcích *SELinuxu* stručně pojednávají následující sekce.

## 2.1 Vynucování typů (TE)

*Vynucování typů* [17] je flexibilní, avšak robustní způsob realizace *povinného řízení přístupu*, a tím pádem i *vynucované bezpečnosti*. Lze jej přizpůsobit širokému spektru *bezpečnostních cílů* a zároveň umožňuje sestoupit při *řízení přístupu* až na úroveň jednotlivých programů. Viz [10].

Realizace *vynucování typů* v *SELinuxu* úzce souvisí s *principem minimálních privilegií* [18]. Tento povoluje jednotlivým subjektům přístup pouze k takovým objektům, které subjekt nutně potřebuje pro naplnění svého legitimního účelu [19]. Toto snižuje riziko poškození objektů kompromitovaným subjektem, i samotné riziko kompromitace subjektu. Viz [10].

*Vynucování typů* přiřazuje subjektům i objektům atribut nazývaný *typ*. V souladu s výchozí politikou *řízení přístupu* povoluje *vynucování typů* přístup jen tehdy, když je *typ* subjektu autorizován pro *typ* objektu. Toto rozhodnutí je dále ovlivněno konkrétní *třídou subjektu* i *objektu* a s touto třídou asociovanými jemně odstupňovanými právy. Viz [10].

Na *vynucování typů* je pak vystavěn celý *SELinux*. Provádí totiž rozdělení celého systému tzv. „pískovišť“ [20], na kterých si jednotlivé procesy „hrají na vlastním písečku“, aniž by mohly závažně ovlivnit ostatní. Samozřejmě v systému nastávají často situace, kdy jeden proces spouští jiný, takže jednotlivá „pískoviště“ nejsou oddělená zcela neprodyšně a lze mezi nimi – v povolených případech – přecházet.

## 2.2 Řízení přístupu založené na rolích (RBAC)

*Řízení přístupu založené na rolích* neoperuje s jednotlivými uživateli, ale s *rolemi*, které jim byly přiřazeny. Toto umožňuje značně zjednodušit správu oprávnění uživatelských účtů vložením abstrakce. Uživateli je přiřazena *role*. *Roli* jsou přiřazena jednotlivá práva. Jedná se tedy o mechanizmus vhodný především pro velké organizace s jasnou organizační strukturou. Viz [11].

V *SELinuxu* lze uživatelům přiřadit *role*. Mezi nimi je pak možné přecházet, pokud to *bezpečnostní politika* povolí. Jednotlivým *rolím* se pak přiřazují *typy*. Pro-

[17] Anglická podoba termínu *vynucování typů* je *„type enforcement“*. V českých textech se používají obě varianty. Tato práce se přiklání k českému *vynucování typů*, případně, jak je zvykem u jiných druhů *řízení přístupu*, používá zkratku *TE*.

[18] V originále: „Principle of least privilege“.

[19] V originále: „Legitimate purpose“.

[20] V anglické terminologii: „Sandbox“.

ces spuštěný uživatelem s určitou *rolí* tak může přistupovat pouze k *typům*, které jsou povoleny této *roli*. *RBAC* je v *SELinuxu* volitelným rozšířením a není příliš využíváno. Viz [10].

## 2.3 Víceúrovňová bezpečnost (MLS)

Koncept *víceúrovňové bezpečnosti* zavádí rozdílné *hierarchické úrovně* klasifikace informací a prověření uživatelů. Prověření uživatelů pak musí být aspoň takové, jako je klasifikace informace. *Bell-La Padulův model* pak zavádí principy „no read-up“ a „no write-down“. Nelze tedy číst citlivější informace, než jaké je prověření, a nelze předat citlivé informace nedostatečně prověřeným uživatelům. Viz [12].

Implementace *MLS* v *SELinuxu* zavádí obvykle šestnáct obecných *úrovní citlivosti*. Tyto se chápou jako prověření, pokud jsou použity u subjektu, či jako úroveň utajení, pokud jsou použity u objektu. Opět jde o volitelné rozšíření výchozího *vynucování typů*, které se příliš nepoužívá. Viz [10].

## 2.4 Multi-Category Security (MCS)

*Multi-Category Security* je obecně chápána jako část *víceúrovňové bezpečnosti*. Na rozdíl od ní však definuje pro subjekty a objekty *nehierarchické kategorie*, takže nevyužívá *Bell-La Padulův model*. Díky tomu odpadají problémy se sdílením informací mezi různými úrovněmi utajení. Viz [10].

V *SELinuxu* dochází při použití *MCS* ke skrytí nevyužitých částí *MLS* a nastavení stejné *úrovně citlivosti* pro všechny objekty a subjekty v systému. Přístup subjektů k objektům je pak povolen jen tehdy, když mají oba stejnou *bezpečnostní kategorii*. I *MCS* je implementována jako volitelné rozšíření, které se vyhodnocuje až po výchozím *vynucování typů*. Viz [10].

## 2.5 Bezpečnostní kontext

*Bezpečnostní kontext* v terminologii *SELinuxu* odpovídá přístupovým atributům objektů a subjektů v obecném *řízení přístupu*. Na rozdíl od linuxového *DAC* je *bezpečnostní kontext SELinuxu* chápán jako jeden atribut složený z několika částí. Tyto části obsahují informace důležité pro jednotlivé prvky *SELinuxu*, tedy např. pro *RBAC*, *vynucování typů* či *MLS*. Viz [10].

Příkladem *bezpečnostního kontextu* může být:

```
user_u:object_r:user_home_t:s0:c24.c25
```

Jak je vidět z příkladu, má tento *bezpečnostní kontext* pět částí. Řetězec `user_u` definuje neprivilegovaného *SELinux uživatele*, označení `object_r` určuje *roli* (zde konkrétně objekt, tedy nějaký soubor), část `user_home_t` popisuje *typ* (v tomto

případě jde o *typ* přiřazený uživatelskému domovskému adresáři), `s0` udává *MLS úroveň citlivosti* a konečně `c24.c25` vymezuje přiřazené *MCS kategorie*.

Protože *MCS* i *MLS* jsou volitelné, nemusí být v politice povoleny, a tedy se ani nemusí vyskytovat v *bezpečnostním kontextu*. Jeho minimální podoba pak je následující trojice:

```
user_u:object_r:user_home_t
```

Z názvů identifikátorů částí *kontextu* je vidět, že dodržují jisté konvence. *SELinux uživatel* je tak vždy `_u`, *role* `_r`, *typ* `_t`, *MLS úroveň citlivosti* `s` a *MCS kategorie* `c`. Viz [10].

Je také potřeba zdůraznit, že všechny části *bezpečnostního kontextu* jsou vnitřní záležitostí *SELinuxu*. *SELinux uživatel* tak vůbec nemusí korespondovat s linuxovým *DAC* uživatelem. Význam *MCS kategorií* a *MLS úrovní citlivosti* musí do lidsky srozumitelné podoby přeložit také bezpečnostní administrátor. Viz [10].

Z celého *bezpečnostního kontextu* je pak pro *SELinux* vzhledem k jeho zaměření na *vynucování typů* nejdůležitější *typ*. Viz [10].

## 2.6 Pravidla SELinuxu

*Řízení přístupu* v *SELinuxu* je řešeno pomocí *vynucování typů*. To k tomuto účelu používá několik druhů pravidel:

- *Přístupová pravidla* – ve výchozím stavu nepovoluje *SELinux* vůbec žádný přístup, *přístupová pravidla* umožňují v určitých případech přístup povolit.
- *Přechodová pravidla* – umožňují vstoupit procesu do určitého *typu*, tedy povolují přechod mezi *typy*.
- *Pravidla rolí* – přiřazují jednotlivým *rolím* povolené *typy*. Viz [10].

V následujících sekcích se budeme věnovat stručnému popisu jednotlivých druhů pravidel.

### 2.6.1 Přístupová pravidla

*Přístupová pravidla* povolují subjektu určitý druh přístupu k objektu. Pro tento účel tedy operují s *typem* subjektu (obvykle jde o *typ* procesu), s *typem* objektu (obvykle nějakého druhu linuxového souboru), s druhem objektu (neboli jeho *třídou*) a s oprávněními (povolenými operacemi). Viz [10].

Takové *přístupové pravidlo* může vypadat následovně:

```
allow kmail_t user_kmail_home_conf_t : file \
      {read getattr write create rename unlink lock};
```

Uvedené pravidlo povoluje subjektu – procesu – s *typem* `kmail_t` operace `read`, `getattr`, `write`, `create`, `rename`, `unlink` a `lock` pro všechny objekty *třídy* `file` – tedy obyčejné soubory – s *typem* `user_kmail_home_conf_t`.

Kromě pravidel s *klíčovým slovem* `allow`, která povolují přístup, existují ještě pravidla s `auditallow`, která přístup povolí a zaznamenají, a `dontaudit`, která nezaznamenají nepovolený přístup. Pokud neexistuje `dontaudit` pravidlo, zaznamenává se každý pokus o nepovolený přístup do logovacího souboru. Viz [10].

Zde se dále hodí připomenout, že *SELinux* je nadstavbou linuxového *DAC*. Aby tedy měla *přístupová pravidla SELinuxu* vůbec nějaký efekt, musí mít subjekt povolený přístup k objektu i oním linuxovým *DAC*. Viz [10].

### 2.6.2 Přechodová pravidla

*Přechodová pravidla* slouží k tomu, aby bylo možné zajistit bezpečný přechod mezi *typy* v okamžiku spuštění aplikace. Jde tedy o něco podobného `suid` bitu v linuxovém *DAC*. V případě *přechodových pravidel* je však vše chráněno *vynucovanou bezpečností*. K tomuto účelu dochází k zavedení nového *typu* pro spustitelný soubor aplikace, který je pak prohlášen za tzv. *vstupní bod*. Program na něj tedy získá oprávnění `entrypoint`. Kromě něj se používá ještě *typ* uživatelského *shellu*, který program spouští, a *typ*, do kterého má spouštěný program přejít. Viz [10].

Toho všeho je dosaženo kombinací tří pravidel, která mohou vypadat následovně:

```
allow kmail_t kmail_exec_t : file entrypoint;
allow user_t kmail_exec_t : file {getattr execute};
allow user_t kmail_t : process transition;
```

V ukázce chceme umožnit přechod z *typu* uživatelského *shellu*, označeného `user_t`, do *typu* aplikace *KMail*, tedy `kmail_t`. Přiřadíme tedy subjektu s *typem* `kmail_t` právo `entrypoint` na objekt *třídy* `file` s *typem* `kmail_exec_t`. Pro proces aplikace *KMail* je tak jeho „binárka“ *vstupním bodem* do *typu* `kmail_t`. Dále povolíme *shellu* běžného uživatele s *typem* `user_t` provádět operace `getattr` a `execute` na souboru s *typem* `kmail_exec_t`. Uživatelský *shell* tedy může „binárku“ *KMailu* vypsat a spustit. Nyní konečně povolíme procesu přechod z *typu* uživatelského *shellu* `user_t` do *typu* `kmail_t`.

Tímto ale došlo pouze k povolení možnosti přechodu mezi *typy*. *SELinux* nebude tyto přechody provádět, pokud k nim nebude explicitně vyzván. Pro vynucení přechodu mezi *typy* je nutné přidat vlastní *přechodové pravidlo*. Viz [10].

*Přechodové pravidlo* pro náš příklad může vypadat takto:

```
type_transition user_t kmail_exec_t : process kmail_t;
```

Toto pravidlo zajišťuje, že pokud *typ* `user_t` přistoupí k *typu* `kmail_exec_t`, dojde v případě spuštění procesu k přechodu do *typu* `kmail_t`.

### 2.6.3 Pravidla rolí

*Pravidlo rolí* povoluje *roli* určitý *typ*. Specifikuje tedy tuto *roli* a jí povolovaný *typ*. Viz [10].

Příklad *pravidla rolí* může být takovýto:

```
role user_r type kmail_t;
```

*Roli* běžného *SELinux uživatele* `user_r` zde povolujeme pracovat s *typem* aplikace *KMail*, tedy `kmail_t`.

## 2.7 Bezpečnostní politika SELinuxu

Samotný *SELinux* nedefinuje žádná pravidla a při *řízení přístupu* postupuje podle principu, co není povoleno, je zakázáno. Je to právě *bezpečnostní politika*, která určuje, jaké přístupy mají být povoleny. *Bezpečnostní politika* je tedy souhrnem pravidel *SELinuxu* a přináší tak komplexní řešení *řízení přístupu* v systému. Viz [10].

*Bezpečnostní politika* je kompilovaná ze zdrojových souborů. Může mít podobu jednoho binárního souboru, nebo základního *modulu* a skupiny rozšiřujících *modulů*. Modulární pojetí *bezpečnostní politiky* je dnes nejběžnější, umožňuje snazší správu a také nasazení pouze těch částí, které jsou opravdu potřeba. Viz [10].

Vzhledem ke složitosti moderních distribucí *GNU/Linuxu* jsou i rozsah a složitost typické *bezpečnostní politiky* značné. Ač se jednotlivé distribuce *GNU/Linuxu* od sebe do určité míry odlišují, větší část kódu *bezpečnostní politiky* zůstává napříč distribucemi stejná. Z těchto důvodů vznikaly vzorové politiky, které si pak jednotlivé distribuce už jen drobně upravovaly. Vyvrcholením těchto snah byl vznik *Referenční politiky* [21], kterou dnes ve více či méně upravené podobě používá většina distribucí nabízejících *SELinux*. Viz [10, 13].

# 3 Technologie tvorby bezpečnostních politik SELinuxu

Abychom mohli hovořit o vlastní technologii tvorby *bezpečnostních politik SELinuxu*, musíme se nejprve blíže podívat na vnitřní strukturu typické *bezpečnostní politiky*. Protože distribuce používají *bezpečnostní politiky* vycházející z *Referenční politiky*, je třeba se dále seznámit i s jejími charakteristikami.

Vlastní technologie tvorby *bezpečnostních politik SELinuxu* spočívá ve zvládnutí syntaxe jazyka pravidel *SELinuxu* rozšířeného *makry jazyka M4*, v orientaci v *rozhraních Referenční politiky* a konečně i v uplatnění obecné metody tvorby *bezpečnostní politiky SELinuxu*. Samotná syntaxe pravidel *SELinuxu* byla již zmíněna

[21] Domovskou stránku projektu *Referenční politiky* lze nalézt na: http://oss.tresys.com/projects/refpolicy/.

v předchozí sekci, hlavní těžiště této sekce tedy bude spočívat v *makrech jazyka M4*, v popisu důležitých *rozhraní* a ve vysvětlení obecné metody tvorby *bezpečnostních politik*.

## 3.1 Vnitřní struktura bezpečnostní politiky

Na nejnižší úrovni je *bezpečnostní politika* tvořena jedním zdrojovým souborem `policy.conf` v jazyce kompilátoru politiky `checkpolicy`. Tento soubor obsahuje:

- Definice *tříd objektů* a *subjektů* a jim přiřazená oprávnění.
- Množinu všech pravidel *SELinuxu* a definice *typů* a *rolí*.
- Určení *SELinux* uživatelů.
- Vnitřní omezení politiky, mimo jiné i implementaci *MLS* a *MCS*.
- Specifikace přiřazující *bezpečnostní kontext* objektům. Viz [10].

Vývojář *bezpečnostních politik* obvykle nemění definice *tříd objektů* a oprávnění ani vnitřní omezení politiky. Naopak pracuje s částí politiky, kde jsou specifikována pravidla *SELinuxu* a definovány *typy* a *role*. Viz [10].

## 3.2 Definované třídy objektů a oprávnění v *bezpečnostní politice*

*Třídy objektů* reprezentují kategorie zdrojů systému. V termínech počítačové bezpečnosti jde jak o objekty, tak i o subjekty. *Třída objektů* je tedy množina všech objektů či subjektů daného druhu. Například tedy může jít o všechny běžné soubory (třída `file`). Viz [10].

Oprávnění představuje množinu operací, které lze se *třídou objektů* provádět. Množina možných oprávnění závisí na konkrétní *třídě objektů*. Procesům lze tedy povolit zasílání signálů (např. oprávnění `signull`), běžným souborům zase čtení (oprávnění `read`). Jednotlivá oprávnění jsou potřeba, aby mohla být provedena určitá systémová volání. Množina všech možných oprávnění pro danou *třídu objektů* se nazývá *přístupový vektor*. Viz [10].

Dostupné *třídy objektů* procházely během vývoje *SELinuxu* různými změnami. Jejich úplný výčet tedy záleží na konkrétní verzi jádra a *Referenční politiky*. Obecně je lze rozdělit na několik skupin:

- *Souborové třídy objektů* – jsou *třídy objektů* na trvalých úložištích, např. adresáře `dir`, běžné soubory `file`, připojené systémy souborů `filesystem`, symbolické odkazy `lnk_file` či Unixové sockety `sock_file`.

- *Síťové třídy objektů* – reprezentují síťové zdroje jako síťová rozhraní `netif`, vzdálený stroj reprezentovaný IP adresou `node`, TCP socket `tcp_socket` či UDP socket `udp_socket`.
- *System V IPC třídy objektů* – reprezentují zdroje *IPC*, jako je fronta zpráv `msgq` či segment sdílené paměti `shm`.
- *Ostatní třídy objektů* – do této skupiny patří například proces `process`. Viz [10].

Většině *souborových tříd objektů* jsou pak společná např. práva: získání atributů `getattr`, čtení `read`, připsání dat `append`, zápisu `write`, vytvoření `create`, vytvoření pevného odkazu `link`, smazání `unlink`, vstupního bodu `entrypoint` a změny *bezpečnostního kontextu* `relabelfrom` a `relabelto`. Je patrné, že jde o značně jemné rozlišení oprávnění. Obdobně důkladně jsou definována práva i pro další *třídy objektů*. Viz [10].

## 3.3 Syntaxe jazyka TE a RBAC pravidel

Syntaxe jazyka pravidel *vynucování typů* již byla částečně zmíněna v podsekci 2.6 na straně 9. Nyní se na ni podíváme detailněji.

Ač je v typické politice *TE pravidel* velké množství, jsou jednotlivá pravidla relativně jednoduchá a existuje pouze několik druhů pravidel. V zásadě lze *TE pravidla* rozdělit na *přístupová pravidla* a *pravidla typů*. *Přístupová pravidla* se týkají přístupu mezi dvěma *typy*, *pravidla typů* řídí přidělování *bezpečnostních kontextů* měněným či vytvářeným objektům. Kromě toho jazyk *TE pravidel* obsahuje výroky umožňující definovat své dílčí komponenty. Viz [10].

*RBAC pravidla* zahrnují jak *role*, tak *SELinux uživatele*. Tyto je třeba nejprve deklarovat. Poté lze pomocí *pravidel povolujících změnu role*, *přechodových pravidel rolí* a *pravidel dominance rolí* pracovat s *rolemi*. *SELinux uživatelé* se mapují k linuxovým uživatelům při přihlášení na základě voleb v příslušném konfiguračním souboru. Viz [10].

### 3.3.1 Deklarace typů, atributů a aliasů

*Typy* jsou základem pro *bezpečnostní politiky SELinuxu*. *Atributy* a *aliasy* pak slouží ke zjednodušení práce se samotnými *typy* v rámci *bezpečnostní politiky*. *Atributy* umožňují odkazovat na určitou skupinu *typů*. *Aliasy* pak určují zástupná jména pro vybraný *typ*. Z hlediska politiky jsou *aliasy* i *atributy* chápány jako *typy*. Viz [10].

Všechny používané *typy* musí být nejprve explicitně deklarovány. Při deklaraci *typů* lze také deklarovat jim přidružené *atributy* a *aliasy*. Viz [10]. Příkladem deklarace *typu* může být:

```
type user_kmail_home_conf_t alias { staff_kmail_home_conf_t \
     unconfined_home_conf_t } , file_type;
```

Zde tedy definujeme *typ* `user_kmail_home_conf_t`. Tomu pak přiřazujeme *aliasy* `staff_kmail_home_conf_t` a `unconfined_kmail_home_conf_t` a *atribut* `file_type`. *Atribut* musí však být nejprve sám deklarován:

```
attribute file_type;
```

Přiřadíme-li *atribut* `file_type` všem *typům* asociovaným se soubory, můžeme pak snadno například zálohovací aplikaci povolit přístup ke všem souborům v systému. Viz [10].

### 3.3.2 Přístupová pravidla

*Přístupová pravidla* se vztahují k oprávněním pro *třídy objektů*. Existují čtyři druhy *přístupových pravidel*. Rozlišuje je *klíčové slovo* uváděné na počátku pravidla:

- `allow` povoluje přístup mezi dvěma *typy*,
- `dontaudit` zakazuje zaznamenávání *AVC zpráv* [22] při zamítnutí přístupu,
- `auditallow` aktivuje zaznamenávání *AVC zpráv* při povoleném přístupu, k žádnému udělení přístupu však tímto pravidlem nedochází,
- `neverallow` určuje, která práva nemají být nikdy povolena jiným pravidlem, a při porušení generuje chybu při překladu. Viz [10].

Obecná syntaxe *přístupových pravidel* vypadá následovně:

- *klíčové slovo* pravidla,
- *zdrojový typ*,
- *cílový typ*,
- *třída objektu cílového typu*,
- *oprávnění pro tuto třídu*.

Kromě *klíčového slova* je možné všechny dílčí části pravidla podle potřeby zmnožit. V takovém případě se daná skupina uzavírá do závorek `{ }` a jednotlivé zmnožené členy se oddělují mezerou. Viz [10]. Příkladem může být již uváděné pravidlo:

[22] Ve výchozím nastavení *SELinux* zaznamenává všechny zamítnuté pokusy o přístup. Žádné povolené přístupy pak nezaznamenává. *AVC zprávy* se ukládají do *logů* v adresáři `/var/log/`. Konkrétní soubor *logu* závisí na dané distribuci. Zakázané přístupy jsou zaznamenávány, jen pokud byl přístup zamítnut *SELinuxem*, tzn. po povolení linuxovým *DAC*. Viz [10].

```
allow kmail_t user_kmail_home_conf_t : file \
      {read getattr write create rename unlink lock};
```

Pokudi bychom v daném příkladě chtěli zaznamenávat udělené přístupy, změníme *klíčové slovo* `allow` za `auditallow`. Pokud bychom naopak nechtěli zaznamenávat neudělení daných přístupových práv, zaměníme `allow` za `dontaudit`. Syntaxe zbytku pravidla zůstává stejná. Viz [10].

*Přístupová pravidla* jsou aditivní. Je-li jedním pravidlem povolena operace čtení `read` a jiným pravidlem operace zápisu `write`, politika jako celek povoluje obě tyto operace `read` i `write`. Viz [10].

Namísto konkrétního *typu* lze na pozici *zdrojového* i *cílového typu* použít *atribut*. Snadno tak lze jedním pravidlem zapsat význam mnoha pravidel. Při překladu politiky pak dojde k nahrazení *atributu* konkrétními *typy*. Pravidlo tak expanduje do mnoha konkrétních pravidel. Stejně jako *typy* lze i *atributy* v pravidle zmnožit. Viz [10].

Dalším prostředkem zjednodušujícím zápis *přístupových pravidel* je *typový výraz* `self`. Tento výraz se používá na místě *cílového typu*. Při překladu je nahrazen *zdrojovým typem*. Pokud by byl *zdrojový typ* zmnožený, vznikne pro každý jednotlivý *typ* jedno pravidlo. Viz [10]. Příkladem může být následující pravidlo, které povoluje procesu aplikace *KMail* zasílání signálů sobě samému:

```
allow kmail_t self : process signal;
```

Ve spojení s *atributem* je možné použít *operátor negace* – vynechávající některé *typy*, jež tento *atribut zahrnuje*. Viz [10]. Na místě *zdrojového* či *cílového typu* tak lze vytvořit výraz vybírající všechny *typy atributu* `file_type` až na `kmail_home_conf_t`:

```
{ file_type -kmail_home_conf_t }
```

Pro určení udělených *oprávnění* lze použít *operátor „divoké karty“* `*`, který vybírá všechna dostupná *oprávnění* pro danou *třídu objektu*. Kromě něj je přípustný ještě *operátor doplňku* `~`, který vybírá zbývající *oprávnění* nevyjmenovaná ve skupině. Viz [10].

Z povahy *modulární politiky* vyplývá, že některé *typy* či další výrazy nemusí být deklarovány ve zdrojových kódech *modulu*. Existují dva výrazové prostředky, jak s takovou situací naložit: *povinný blok* `require` a *nepovinný blok* `optional`. Takto lze určit povinné a nepovinné závislosti *modulu*. Tyto bloky jde pak podle potřeby kombinovat. Viz [10]. Příkladem může být následující výňatek z politiky, který povolí aplikačnímu *typu* `kmail_t` číst a zapisovat do souborů s *typem* `akonadi_home_conf_t` v případě, že je tento *typ* dostupný, tj. pokud je zaveden i *modul* `akonadi`:

```
optional {
     require {
          type akonadi_home_conf_t;
     }
allow kmail_t akonadi_home_conf_t : file { read write };
}
```

### 3.3.3 Pravidla typů

*Pravidla typů* se dělí na dvě skupiny. *Přechodová pravidla* určují výchozí *typ* při přechodu mezi *typy* nebo při vytvoření nového objektu. *Pravidla změny typu* určují *typy* pro přeznačkování objektů prováděné aplikacemi. Viz [10].

Obecná syntaxe *pravidel typů* je podobná *přístupovým pravidlům*:

- *klíčové slovo* pravidla, tedy `type_transition`, nebo `type_change`,
- *zdrojový typ*,
- *cílový typ*,
- *třída vytvářeného objektu*,
- *výchozí typ*. Viz [10].

Kromě *výchozího typu* je opět možné ostatní části pravidla zmnožit. *Výchozí typ* musí být jedinečný pro kombinace *zdrojového* a *cílového typu* a *tříd vytvářených objektů*. Viz [10].

*Přechodová pravidla* se používají ve dvou situacích. V první jde o již zmíněný přechod mezi *typy* v okamžiku spuštění aplikace, kdy objektem i subjektem je *třída* `process`. Druhou situací je změna *typu* nově vytvářených objektů, především těch ze *tříd objektů* systému souborů, tedy např. `file`, `dir` a podobně. První případ se často také označuje jako *doménový přechod* [23], druhý pak jako *objektový přechod*. Není-li zadáno žádné *přechodové pravidlo*, dojde při vytváření nového objektu k ponechání *typu* subjektu. *Přechodová pravidla* nepovolují přístup, a proto musí být doplněna příslušnými *přístupovými pravidly*, aby mohla úspěšně fungovat. Viz [10].

Příkladem pravidla pro *doménový přechod* je zde již uváděné pravidlo:

```
type_transition user_t kmail_exec_t : process kmail_t;
```

Příkladem pravidla pro *objektový přechod* může být například:

[23] Pojem *doména* se zejména ve starší terminologii používá pro *třídu objektu* `process` i v jiných souvislostech než je *doménový přechod*.

```
type_transition kmail_t kde_home_conf_t : \
     file kmail_home_conf_t;
```

Vytvoří-li proces s *typem* `kmail_t` soubor v adresáři s *typem* `kde_home_conf_t`, bude tento soubor označen *typem* `kmail_home_conf_t` namísto výchozího *typu* daného *typem* adresáře `kde_home_conf_t`.

*Pravidla změny typu* lze například využít k přeznačkování zařízení terminálu při přihlášení uživatele. Jejich využití však je z hlediska tvorby politiky pro aplikace zanedbatelné. Viz [10].

### 3.3.4 Deklarace rolí a SELinux uživatelů

Společně s deklarací *role* probíhá i asociace *typů* s *rolí*. Pokud již byla *role* dříve deklarována, má tento výraz pouze význam asociace *typu* s *rolí*. Viz [10].

Následujícím pravidlem tak *roli* `user_r` – která se v politikách asociuje s běžným uživatelským účtem – přiřazujeme *typy* `kmail_t` a `konqueror_t`. Má-li uživatel přiřazenou tuto *roli*, může spustit aplikace, které přecházejí do uvedených *typů*:

```
role user_r types { kmail_t konqueror_t };
```

Podobně vypadá i deklarace *SELinux uživatele*. Opět se v jednom kroku provádí s asociací s *rolí*. Příkladem může být deklarace *obecného SELinux uživatele* `user_u`. Viz [10]:

```
user user_u roles user_r;
```

### 3.3.5 Ostatní RBAC pravidla

Mezi ostatní *RBAC pravidla* patří *pravidla povolující změnu role* s *klíčovým slovem* `allow`, která povolí změnu *role* za běhu programu. Dále pak jde o *přechodová pravidla rolí* s *klíčovým slovem* `role_transition`, která způsobí automatické provedení změny *role*. *Pravidla dominance rolí* s *klíčovým slovem* `dominance` vytvářejí hierarchickou strukturu *rolí*. Viz [10].

## 3.4 Vzory pro výchozí značkování objektů v systému souborů

Kromě objektů, které vzniknou za běhu systému a mají tedy *bezpečnostní kontext* přiřazený *TE pravidlem*, či zděděný po svém tvůrci, existují i objekty na trvalých úložištích, které *bezpečnostní kontext* přidělený nemají. Kromě vyměnitelných médií, kterými se v této práci nebudu zabývat, se jedná i o objekty v linuxových systémech souborů. Tato situace nastává obzvláště při tvorbě či zavádění nové části

*bezpečnostní politiky. SELinux* toto řeší prostřednictvím *vzorů pro výchozí značkování objektů.* Viz [10].

Tyto vzory jsou uložené ve zvláštním souboru odděleném od vlastní politiky. K identifikaci objektu používají linuxové cesty, samotný *kontext* je však po provedení označkování systému souborů spjat s konkrétním objektem a nikoliv jeho cestou. Cesta k objektu je zpravidla zapsána pomocí regulárních výrazů, které umožňují širší pokrytí objektů. Vyskytovat se mohou také určité proměnné. Kromě toho vzory volitelně určují *třídu objektů* a její výchozí *kontext.* Viz [10].

Příkladem mohou být následující vzory:

```
/usr/bin/kaddressbook.*                    -- \
     system_u:object_r:kaddressbook_exec_t,s0
HOME_DIR/\.kde/share/apps/kabc(/.*)?         \
     system_u:object_r:kaddressbook_home_data_t,s0
```

První vzor vybírá soubory v adresáři `/usr/bin/`, jejichž název začíná řetězcem `kaddressbook` a pokračuje libovolně četným (tedy i nulovým) opakováním libovolného znaku. Tento regulární výraz tedy vybere jak soubor `kaddressbook`, tak i `kaddressbookmigrator`. Těmto souborům pak přiřadí daný *bezpečnostní kontext.*

Následující vzor vybírá veškeré objekty, které jsou umístěny v podadresáři `.kde/share/apps/kabc` domovských adresářů uživatelů. Mezi takové objekty patří jak samotný adresář `kabc`, tak veškerý jeho obsah. To je způsobeno regulárním výrazem `(/.*)?`, který říká, že se v cestě za řetězcem `kabc` může či nemusí vyskytovat ještě lomítko a libovolně četné opakování libovolného znaku. Protože se znak tečky (`.kde`) vyskytuje ve vlastní cestě, musí být předcházen zpětným lomítkem, aby nedošlo k jeho interpretaci jakožto regulárního výrazu. Objektům je pak přiřazen daný *bezpečnostní kontext.*

## 3.5 Charakteristika Referenční politiky

*Referenční politika* si klade několik cílů, a to jak v rovině bezpečnostní, tak v rovině funkcionality. Z bezpečnostního hlediska má jít především o poskytnutí pevného základu zajišťujícího, že *bezpečnostní cíle* systémů s *SELinuxem* jsou dosažitelné a ověřitelné. Z hlediska funkčnosti jde především o usnadnění správy politiky, jejího rozšiřování i tvorby nástrojů pro práci s ní. Viz [13].

Pro zajištění vhodné úrovně bezpečnosti si *Referenční politika* klade následující čtyři cíle:

- Ochránit systém, samotnou politiku a jednotlivé aplikace před sebou samými i před sebou navzájem;

- zajistit důvěru ve správnost a úplnost politiky během celého vývojového cyklu;
- umožnit bezpečné rozšíření politiky, které neohrozí stávající ochranu aplikací;
- a vylepšit oddělení *rolí* tak, aby byly jemně rozčleněny. Viz [13, 14].

Aby dosáhla deklarované funkčnosti, zaměřuje se *Referenční politika* na následující cíle:

- Dosáhnout zvládnutelné složitosti pro vývojáře politiky;
- podporovat modulární i monolitickou podobu politiky;
- vhodně strukturovat politiku, aby byla snáze čitelná pro různé nástroje;
- umožnit třetím stranám vlastní rozšíření politiky prostřednictvím stabilních určených *rozhraní*;
- vylepšit srozumitelnost důkladnou dokumentací;
- a podporovat v jednom stromě všechny dostupné druhy politiky, které zpřístupní vhodné volby při překladu do binární podoby. Viz [13, 14].

K dosažení výše uvedených cílů zavádí *Referenční politika* několik návrhových konceptů a pravidla pro jejich dodržování. Za účelem usnadnění porozumění politice a zpřehlednění organizace jejích *modulů*, používá *Referenční politika* slabý princip *vrstev*. *Moduly* politiky jsou seskupovány do *vrstev* podle své funkce v systému. U nižších *vrstev* se očekává, že budou zahrnuty do většiny politik, vyšší *vrstvy* často zahrnovány být nemusí. Mezi nyní používané *vrstvy* patří:

- *kernel* – obsahuje *moduly* politiky pro linuxové jádro, jednotlivá zařízení, systémy souborů, či síťování,
- *roles* – obsahuje *moduly* politiky pro uživatelské *role*,
- *system* – obsahuje *moduly* politiky pro sdílené knihovny, init systém, správu přihlašování či správu sítě,
- *services* – obsahuje *moduly* politiky pro služby a démony, které nejsou součástí *vrstvy system*,
- *admin* – obsahuje *moduly* politiky pro správcovské nástroje,
- *apps* – obsahuje *moduly* politiky pro všechny ostatní programy. Viz [10, 13, 15].

Velmi důležitým konceptem pro *Referenční politiku* je *silná modularita*. Tato se projevuje nejen v samotných binárních *modulech* politiky, ale i v tom, že každý *modul* má oddělené zdrojové kódy. Každý *modul* obsahuje deklarace prvků politiky, se kterými pracuje, vlastní pravidla, vzory pro výchozí značkování objektů a rozhraní pro komunikaci s ostatními *moduly*. K dosažení této *striktní modularity* využívá *Referenční politika* koncepty *zapouzdření* a *abstrakce*. Princip *zapouzdření* vyžaduje, aby všechny implementační detaily *modulu* pro něj byly soukromé. Takto je lze měnit bez vlivu na ostatní *moduly*. V *Referenční politice* tak neexistují veřejné *typy* ani *atributy* – ostatní *moduly* k nim přistupují pouze přes *rozhraní*. Pomocí *abstrakce* určuje *Referenční politika* vysokoúrovňové koncepty přístupu. *Abstrakce* je dosahováno *M4 makry*, o kterých bude řeč v sekci 3.7 na straně 23. Viz [10, 13].

Výraznou změnou oproti předchozím vzorovým politikám je koncept *rozhraní*, který *Referenční politika* zavedla. Bližší popis některých rozhraní bude následovat v sekci 3.8 na straně 25. Nyní se zmíním jen o několika důležitých obecnostech. Účelem *rozhraní* je zpřístupnit soukromé prostředky daného *modulu*. Všechny *moduly*, které potřebují konkrétní přístup k onomu *modulu*, tak budou používat stejné *rozhraní*. Případné změny v implementaci konkrétního přístupu se tedy odehrávají na jednom místě. V *Referenční politice* existují dva druhy *rozhraní*:

- *přístupová rozhraní* – udělují nějaký druh přístupu,
- *šablony* – vytvářejí *odvozené typy*.

Všechna *rozhraní* používají stanovený způsob pojmenování, který zlepšuje srozumitelnost. Jeho obecná forma je `modul_účel`, podrobněji se mu budeme věnovat v sekci 3.6 na straně 20. Viz [10, 13].

## 3.6 Struktura modulu Referenční politiky

Každý *modul Referenční politiky* je ve své zdrojové podobě tvořen třemi textovými soubory bez ohledu na to, zda výsledná binární podoba politiky je *modulární*, nebo *monolitická*. Jedná se o *soubor s pravidly* (`.te`), *soubor s rozhraními* (`.if`) a *soubor s kontexty* (`.fc`). *Soubor s pravidly* obsahuje soukromé části politiky pro daný *modul*. *Soubor s rozhraními* poskytuje veřejná abstraktní *rozhraní* pro přístup k těmto soukromým částem politiky. V *souborech s kontexty* jsou pak uložené *vzory pro výchozí přeznačkování objektů*. Viz [10, 13, 16].

### 3.6.1 Soubor s pravidly (.te)

Základní členění *souboru s pravidly* je následující. Nejprve se uvádí hlavička, která obsahuje název *modulu* a jeho verzi. Viz [16]. Tedy například:

```
policy_module(kmail,0.2.1)
```

Poté následuje část s deklaracemi pro celý *modul*. Nejprve se uvádí deklarace *přepínačů* [24], následují deklarace výchozích *voleb* [25] pro volitelné bloky politiky, poté se deklarují *atributy* a nakonec *typy*. Každá dílčí skupina deklarací se řadí abecedně. S deklaracemi se pojí také volání přidružených *rozhraní* podrobněji specifikujících *typ* či *atribut*. Viz [17].

V další části souboru se uvádějí pravidla soukromé politiky *modulu*, a to postupně pro každou jeho *doménu*. Nejprve se vypíší pravidla s výrazem `self`. Po nich následují pravidla týkající se objektů vlastněným daným *modulem* podle *třídy objektu*. Další na řadě jsou volání *rozhraní* z *vrstvy kernel* řazená abecedně. Následují volání *rozhraní* z *vrstvy system* řazené abecedně. Poté se píší volání všech *rozhraní* z ostatních *modulů*. Následují zápisy jednotlivých bloků politiky: nejprve *bloky distribucí* ovlivnitelné nastavením při překladu, jako je `distro_redhat`, pak *podmíněné bloky* ovlivnitelné přepínači `conditional_policy`, poté *volitelné bloky* `tunable_policy` a konečně *nepovinné bloky* `optional_policy`. Viz [17].

Na konec *souboru s pravidly* se píše vše týkající se *domény* `unconfined` podle stejných pravidel, jaká se vztahují na soukromou politiku *modulu*. Viz [17].

### 3.6.2 Soubor rozhraní (.if)

*Soubor s rozhraními* obsahuje jak samotná *rozhraní*, tak i jejich dokumentaci. Krom toho tento soubor obsahuje hlavičku s dokumentací celého *modulu*. Dokumentace využívá *XML* formátu a jsou v ní povoleny i některé *HTML* značky. Dokumentace musí být předcházena dvojím znakem pro komentář ##. Většinou se v hlavičce uvádí pouze krátký popis aplikace. Viz [15, 16]. Příkladem může být:

```
## <summary>Kmail KDE e-mail client</summary>
```

Zbytek souboru obsahuje jednotlivá *rozhraní* předcházená jejich dokumentací. Dokumentace stručně popisuje, jaký přístup dané *rozhraní* uděluje a jaké jsou jeho parametry. Uveďme příklad dokumentace k *rozhraní* `kmail_role()`:

```
########################################
## <summary>
##      Role access for kmail
## </summary>
## <param name="role">
##      <summary>
##      Role allowed access
##      </summary>
## </param>
## <param name="domain">
```

[24] V originále „booleans“.
[25] V originále „tunables“.

```
##      <summary>
##      User domain for the role
##      </summary>
## </param>
#
```

Podle dokumentace tedy toto *rozhraní* přidává dané *roli* přístup k *doméně* aplikace *KMail*.

Jednotlivá *rozhraní* musí být řazena podle přesně uvedené struktury. Jako první se uvádějí *šablony* daného *modulu*. Následují *přidružená rozhraní typů*, která určují význam daného *typu*. Viz [17].

Následují *přístupová rozhraní* řazené podle *třídy objektu*, dále podle rostoucích oprávnění a konečně podle *klíčového slova* pravidla. Poté se zapisují *administrační rozhraní* nejprve pro jednotlivé komponenty *modulu* a nakonec pro celý *modul*. Jako poslední se uvádějí *rozhraní* týkající se *domén* `unconfined`. Viz [17].

Jednotlivá *rozhraní* a *šablony* musí dodržovat jmenné konvence. Název *rozhraní* se skládá ze čtyř částí, místo mezer se používají podtržítka:

- *Název modulu*, který se případně zkracuje pro moduly s dlouhým názvem.
- *Modifikátor* je volitelný a určuje variace daného rozhraní, např. `dontaudit`.
- *Sloveso* popisuje akci či přístup, které dané rozhraní povoluje.
- *Predikát* se skládá ze jména s proměnným počtem přívlastků a určuje, jakého objektu se *rozhraní* týká. Predikát může být vynechán, pokud daný objekt vyplývá jasně z kontextu. Viz [18].

Vyčerpávající přehled příkladů *sloves* a *predikátů* přináší zdroj [18], uvedu tedy jen jednoduchý příklad pojmenování *rozhraní* z politiky poštovního klienta *KMail*: `kmail_read_data_home_files`. *Název modulu* je tedy `kmail`. *Modifikátor* není uveden, jde tak o *rozhraní* poskytující přístup `allow`. *Sloveso* vyjadřující akci je `read`, což znamená povolení čtení objektu. *Predikát* popisující *třídu objektu* je `data_home_file`. Jde o běžný soubor v adresáři s daty aplikací v domovském adresáři uživatele, tedy `~/.local`.

Obdobně vypadají i jmenné konvence pro *šablony*. Jako první se uvádí název *modulu*, následuje část určující účel *rozhraní* a konečně se udává slovo `template` pro odlišení od běžných *rozhraní*. Viz [19].

### 3.6.3 Soubor s kontexty (.fc)

*Soubor s kontexty* obsahuje jednotlivé *vzory pro výchozí přeznačkování objektů* s případnými komentáři či *bloky distribucí*. Jednotlivé *vzory* se řadí abecedně podle cesty, poté se vzrůstající hloubkou a konečně se *vzory* s metavýrazy upřednostňují před *vzory* s přesnou shodou. Viz [17].

## 3.7 Syntaxe maker jazyka M4

*Makra jazyka M4* v *Referenční politice* slouží k vytváření abstrakce. Vztahují se na ně však mnohá omezující pravidla: každé *makro* se smí týkat pouze jednoho konceptu a není možné *makra* přidávat zcela libovolně. *Referenční politika* rozlišuje tři typy *maker*: *podpůrná makra*, *makra rozhraní* a *makra šablon*. Hlavní rozdíl mezi *podpůrnými makry* na jedné straně a *makry rozhraní a šablon* na druhé straně spočívá pro tvůrce politiky v předpokládaném pasivním využití *podpůrných maker*. *Makra rozhraní a šablon* obvykle tvůrce politiky – alespoň pro svůj *modul* – sám vytváří. Viz [13].

### 3.7.1 Podpůrná makra

*Podpůrná makra* jsou distribuována se zdrojovými kódy *Referenční politiky* v souborech adresáře `policy/support`:

- `loadable_module.spt` – podpora pro *moduly* politiky,
- `misc_macros.spt` – *makra* generující *SELinux uživatele*, *přepínače* a *bezpečnostní kontexty*,
- `mls_mcs_macros.spt` – podpora pro *MLS* a *MCS*,
- `file_patterns.spt` – seskupení pravidel povolujících přístup k souborům a adresářům,
- `ipc_patterns.spt` – seskupení pravidel povolujících přístup k *IPC* zdrojům,
- `misc_patterns.spt` – podpora *doménových přechodů*,
- `obj_perm_sets.spt` – seskupení oprávnění pro *třídy objektů*. Viz [19].

*Podpůrná makra* zjednodušují programování *bezpečnostní politiky*. Při překladu nástrojem `make` [26] expandují do odpovídajících pravidel *SELinuxu*. Obecná syntaxe pro volání *makra* je: `identifikátor_makra(parametr1, parametr2)`. V případě, že parametrem *makra* jsou celé výrazy – jako u *nepovinných bloků* a podobně – uvádí se zpětný apostrof ‘ na začátek a běžný apostrof ’ na konec bloku. Viz [19].

Mezi *makra* ze souboru `loadable_module.spt` patří již uváděné výrazy: `policy_module()`, `optional_policy()`, `tunable_policy()`. Dále například *makro* pro *povinný blok*, které má v *Referenční politice* podobu `gen_require()`, nebo *makro* generující *přepínače politiky* `gen_tunable()`. Viz [19].

[26] Nástroj `make` využívá, jak specifikuje `Makefile` dané politiky, pro vlastní kompilaci skupinu nástrojů. Těmito nástroji jsou například: `checkpolicy`, `checkmodule`, `semodule`, `xmllint`, `m4` či `python`.

Příkladem *maker* definovaných v souboru `misc_macros.spt` pak je především *makro* `gen_context(context,mls,mcs)`. Parametr `context` udává základní *SELinux kontext*, volitelné parametry `mlc` a `mcs` pak udávají případné části *kontextu* pro *MLS* a *MCS* politiky. Jiným příkladem je *makro* `gen_user()`. Viz [19].

V souboru `obj_perm_sets.spt` se nacházejí *makra*, která udělují odstupňované skupiny práv pro jednotlivé *třídy objektů*. *Makra* tohoto druhu jsou bezparametrická a lze je použít na místě oprávnění v *přístupových pravidlech*. Příkladem mohou být *makra*: `read_file_perm`, `write_file_perm`, `create_file_perm`, `delete_file_perm` a `manage_file_perm`, která vždy seskupují všechna potřebná práva pro danou operaci.

Konečně *makra* v souboru `file_patters` generují *přístupová pravidla* k souborovým *třídám objektů* obdobně odstupňovaná dle vzrůstajících oprávnění. Tato *makra* mají obvykle tři parametry: *doménu* přistupujícího procesu, *typ* nadřazeného adresáře a *typ* vytvářeného objektu. Mohou tak sloužit i jako *přechodová pravidla objektů*. Na příklad se jedná o *makra*: `read_sock_files_pattern()`, `write_sock_files_pattern()`, `manage_sock_files_pattern()` a další obdobně pojmenovaná.

### 3.7.2 Makra rozhraní a šablon

*Makra rozhraní* smí být definována pouze v *souborech s rozhraními*. Volána mohou být i ze *souboru z pravidly*. Dokumentace *makra* je povinná, její absence vyvolá chybu při překladu. Obecná syntaxe pro definici *makra rozhraní* vypadá následovně: `interface('identifikátor_rozhraní','pravidla')`. Viz [19].

*Identifikátor rozhraní* musí odpovídat jmenným konvencím pro *rozhraní* tak, jak byly popsány v sekci 3.6.2 na straně 21. Mezi *pravidla* patří jednotlivá *pravidla SELinuxu*, či volání jiných *rozhraní*, volitelně další vyjadřovací prostředky politiky a povinně blok `gen_require('závislosti')`. Mezi *závislosti* v něm uváděné patří všechny *typy* či *role*, se kterými *makro* přímo operuje. Předávání parametrů *makra* probíhá proměnnými `$1`, `$2`, atd., kdy čísla odpovídají pořadí definice parametru v dokumentaci, respektive pořadí zadávání parametrů při volání definovaného *makra*. Viz [19]. Úplný příklad volání *makra rozhraní* následuje:

```
########################################
## <summary>
##      Execute a domain transition to run kmail.
## </summary>
## <param name="domain">
## <summary>
##      Domain allowed to transition.
## </summary>
## </param>
```

```
#
interface(‘kmail_domtrans’,‘
        gen_require(‘
                type kmail_t;
                type kmail_exec_t;
        ’)

        domtrans_pattern($1,kmail_exec_t,kmail_t)
’)
```

Definované *rozhraní* nese název `kmail_domtrans`, ke své činnosti potřebuje *typy* `kmail_t` a `kmail_exec_t` z *modulu* `kmail` a svou činnost naplňuje voláním *podpůrného makra* `domtrans_pattern()`, kterému předává jediný proměnný parametr a dva *typy*, na kterých závisí. Z hlediska uživatele tohoto *makra* poskytuje *makro* službu *doménového přechodu* zadané *domény* do *domény* aplikace *KMail*.

*Makra šablon* slouží k vytvoření *domény* a *odvozených typů* aplikace pro daného *SELinux uživatele* a *roli*. V takto vytvořené *doméně* pak spouští aplikaci, čímž umožňují od sebe oddělit různé instance téže aplikace. Viz [19].

*Makra šablon* obvykle mají tři parametry: *SELinux uživatele*, který slouží jako *prefix*, *typ domény* daného uživatele a *roli* daného uživatele. Komentáře, ze kterých se generuje dokumentace, jsou opět povinné. Obecný předpis pro definici *šablony* je následující: `template(‘identifikátor_šablony’,‘pravidla’)`. Identifikátor *šablony* musí odpovídat jmenným konvencím uváděným v sekci 3.6.2 na straně 21. Mezi pravidla patří různé výrazy politiky, vlastní *pravidla SELinuxu*, volaná *rozhraní*, či volání *podpůrných maker*. I v tomto případě je povinností uvést blok `gen_require()` definující závislosti na *typech* či *rolích*. Viz [19].

## 3.8 Vybraná rozhraní Referenční politiky

Úplný výčet *rozhraní* a *šablon Referenční politiky* generovaný z dokumentace se nachází na stránkách *API Referenční politiky* [15]. V této sekci uvedu některá *rozhraní*, která byla využita při tvorbě politik pro aplikace *prostředí KDE*.

Při deklaraci *typů bezpečnostní politiky* jsou často používána následující *rozhraní*, která zpřesňují význam deklarovaných *typů*:

- `files_tmp_file(typ)`: označuje dočasné soubory v `/tmp` a `/var/tmp`.
- `ubac_constrained(typ)`: určuje *typy* – objektu i subjektu – omezené pomocí *UBAC* [27].

[27] *UBAC* je *řízení přístupu Referenční politiky* založené na *SELinux uživatelích*. Umožňuje tedy oddělit soubory a procesy jednotlivých *SELinux uživatelů*. Viz [15].

- `userdom_user_home_content(typ)`: nastaví daný *typ* tak, aby byl použitelný v rámci domovských adresářů uživatelů. Viz [15].

Řada dalších *rozhraní* se používá pro udělení přístupu k *typům*, které aplikace potřebuje. Zmíním několik z nich:

- `corecmd_exec_bin(doména)`: umožní programu běžícímu v *doméně* spustit programy v `/bin`, `/usr/bin`, ..., které nemají vlastní *typ*, aniž by došlo k *doménovému přechodu*.
- `dev_read_urand(doména)`: povoluje programu číst z `/dev/urandom`.
- `dbus_connect_session_bus(doména)`: program může požádat uživatelský *D-Bus* o zaregistrování služby.
- `dbus_session_bus_client(doména)`: *šablona* povolující přístupy k uživatelskému *D-Busu*.
- `files_read_etc_files(doména)`: umožní programu číst konfigurační soubory v `/etc`, které nemají vlastní *typ*.
- `kernel_read_system_state(doména)`: povolí programu získávat informace o stavu systému z `/proc`.
- `userdom_use_user_terminals(doména)`: program lze spustit v terminálu běžného uživatele.
- `xserver_user_x_domain_template(prefix, doména, typ)`: poskytuje základní klientský přístup k *X serveru*. Kromě obvyklé *domény* potřebuje tato *šablona* znát i *prefix* aplikace a *typ* jejích dočasných souborů. Tato *šablona* je nutností pro každou grafickou aplikaci. Viz [15].

Další *rozhraní* řeší specifičtější přístupy, takže se již značně liší podle konkrétní aplikace, pro níž je politika určena. Většina uživatelských aplikací však potřebuje volat výše uvedená *rozhraní Referenční politiky*.

## 3.9 Obecná metoda tvorby modulu bezpečnostní politiky

Obecně použitelnou metodu vývoje *modulu bezpečnostní politiky SELinuxu* uvádí Dan Walsh ve svém článku *Průvodce tvorbou nového modulu politiky SELinuxu* [20]. Tato metoda spočívá ve třech dílčích krocích a není závislá na konkrétních nástrojích, ač autor uvádí některé, které mohou s jednotlivými kroky pomoci. Před vlastním nasazením metody je pak nezbytné poznat danou aplikaci a stanovit si *bezpečnostní cíle* pro vytvářenou politiku. Viz [20].

Prvním krokem metody je vytvoření základní kostry politiky pro danou aplikaci. Především jde o deklaraci *typů* a s tím spojená volání *rozhraní* upřesňujících

jejich význam. Též je potřeba napsat pravidlo *doménového přechodu* do *domény* uzavírané aplikace. Aplikaci se musí povolit přístup k deklarovaným *typům* a je vhodné vytvořit *rozhraní* pro práci s těmito *typy*, která pak mohou využít i jiné aplikace. S tím souvisí i konstrukce *vzorů pro výchozí značkování objektů* k deklarovaným *typům*. Protože linuxové aplikace často přistupují k některým službám systému či potřebují číst některá zařízení nebo konfigurační soubory, lze v tuto chvíli vygenerovat i *přístupová pravidla* či volat příslušná *rozhraní* pro tyto situace. Dan Walsh doporučuje ke generování základní kostry použít nástroj *sepolgen*. Viz [20].

Druhý krok začíná překladem dosud vytvořené politiky příkazem `make`, kterému se jako parametr předává `Makefile` používané *bezpečnostní politiky*, již rozšiřujeme. V tuto chvíli může být nutné opravit nějaké chyby v syntaxi pravidel. Poté lze nový *modul* zavést do systému příkazem `semodule`. Po přeznačkování systému souborů příkazem `restorecon` je čas přejít ke třetímu kroku. Viz [20].

Ve třetím kroku se začíná testováním, zda lze aplikaci spustit, případně v ní vykonat požadovanou činnost. V tuto chvíli nejspíše dojde k zamítnutí velkého množství přístupů a vygenerování jim odpovídajících *AVC zpráv*. K jejich prohlížení lze využít nástroj *SETroubleShoot*. Nyní je třeba posoudit, které přístupy jsou pro daný *bezpečností cíl* potřeba, a které nikoliv. Podle toho se pak rozšiřuje stávající politika o další pravidla či volání *rozhraní*. Viz [20].

Druhý a třetí krok metody se opakuje tak dlouho, dokud spouštění a používání aplikace generuje nějaké *AVC zprávy*. V okamžiku, kdy už se další *AVC zprávy* neobjevují, lze prohlásit vývojovou fázi za ukončenou a vytvořený *modul* poskytnout k testování v běžném provozu. Pak už zbývá pouze *modul* udržovat, aby reflektoval vývoj aplikace a případné další požadavky od uživatelů *modulu*. Viz [20].

Část II

# Bezpečnostní politiky pro uživatelský prostor

Druhá část této práce se zabývá vztahem *uživatelského prostoru* a *bezpečnostní politiky*. Nejprve tedy samotný pojem definuje a charakterizuje a to i ve vztahu k *SELinuxu*. Poté se zaměřuje na analýzu *uživatelského prostoru*, kde především akcentuje *prostředí KDE*, pro jehož vybrané aplikace vznikla v rámci této práce *bezpečnostní politika*. Nakonec se tato část vyjadřuje k možným přístupům k uzavírání *uživatelského prostoru* a argumentuje, proč byl vybrán přístup vytvářející politiku od koncových aplikací směrem ke službám *prostředí KDE* a zbytku *uživatelského prostoru*.

## 4 Specifika uživatelského prostoru pro zabezpečení

Nejprve je třeba termín *uživatelský prostor* definovat. Jeho definice umožní přejít k podrobnější charakteristice *uživatelského prostoru* systému *GNU/Linux*. Poté je možné podívat se na vztah *uživatelského prostoru* a *SELinuxu* a s tím související úroveň uzavření *uživatelského prostoru* v *Referenční politice*. Pak lze stanovit *bezpečnostní cíle* pro politiku *SELinuxu* uzavírající *uživatelský prostor*.

### 4.1 Charakteristika uživatelského prostoru GNU/Linuxu

Termín *uživatelský prostor* se tak, jak jej používá tato práce, vymezuje vůči termínu *systém*. *Systémem* se myslí nejen samotné jádro *Linux* [28], ale také skupina nástrojů *GNU* [29], různé démony a serverová část grafického systému *X11* [30]. Na *uživatelský prostor* tedy zbývají různé druhy *aplikačního softwaru*. V zásadě jde o některé *CLI aplikace* [31] a snad naprostou většinu *GUI aplikací*. Je-li řeč o grafických aplikacích, je třeba zvláště připomenout, že se jedná i o *pracovní prostředí* a jejich služby a nikoliv pouze o koncové aplikace, se kterými uživatel pracuje.

[28] Domovská stránka projektu *Linux*: `http://kernel.org/`.

[29] Domovská stránka projektu *GNU*: `http://www.gnu.org/software/`.

[30] Domovská stránka projektu *X.org*: `http://www.x.org/wiki/`.

[31] Příkladem *CLI aplikací uživatelského prostoru* může být webový prohlížeč `links` (domovská stránka: `http://www.jikos.cz/~mikulas/links/`), *IM* klient `irssi` (domovská stránka: `http://irssi.org/`) či poštovní klient `mutt` (domovská stránka: `http://www.mutt.org/`).

*Uživatelský prostor* však není pouze o softwaru, jeho velmi důležitou součástí jsou *uživatelská data*. Tato data bývají umístěna v domovském adresáři uživatele. Jde o vlastní obsah vytvořený uživatelem, tak i o nastavení či logy jeho programů. Mimo domovský adresář pak uživatelská data pronikají do adresáře s dočasnými daty `/tmp` či `/var/tmp`. Značné množství uživatelských dat se může nacházet na výměnných úložištích.

V neposlední řadě pojem *uživatelský prostor* zahrnuje jednotlivé uživatele. To znamená jak uživatelské účty, tak jim přidělené skupiny, jimiž se de facto v rámci linuxového *DAC* vyjadřují jejich *role*. Toto vše tedy zahrnuje termín *uživatelský prostor*.

Když jsme si určili, co to *uživatelský prostor* je, podívejme se nyní podrobněji na jeho vlastnosti. Obecně lze říci, že *uživatelský prostor* je značně různorodý, a to jak z hlediska softwaru, tak i dat a uživatelských účtů. Také je jasně vidět, že *uživatelský prostor* je velmi početný – opět existuje mnoho aplikací, bývá zvykem mít mnoho dat a v případě organizací i mnoho uživatelů. *Uživatelský prostor* na jedné straně využívá různé *systémové zdroje*, na druhé straně je určitým způsobem používán uživatelem.

Pro software, který je součástí *uživatelského prostoru*, pak navíc často platí značná komplexita a propojenost. Zde je nutno přiznat, že se to týká především velkých *pracovních prostředí* jako *KDE* či *GNOME* a aplikací psaných pro ně. Naproti nim pak existuje určitá skupina programů, jejichž cílem není úzká integrace. Výše zmíněné je v podstatě dvojí stránkou jedné věci – propojenost aplikací zvyšuje jejich hodnotu pro uživatele, ovšem zároveň zvyšuje složitost systému jako celku. [32]

Co se týče dat v *uživatelském prostoru*, podstatnou vlastností je jejich důležitost pro uživatele – leč tato bývá i samotnými uživateli často značně podceňována. Z tohoto důvodu je tedy uživatelská data potřeba chránit. S tím souvisí i citlivost uživatelských dat. Ta se samozřejmě liší pro konkrétní data, avšak lze říci, že je v zásadě vyšší než citlivost dat samotného *systému*. Zmíněná důvěrnost dat bývá opět často opomíjena mimo jiné proto, že je pro nepropojená data značně nižší. V *uživatelském prostoru* se však nachází mnoho různých dat, jejichž propojením a uchopením ve správném kontextu lze získat velmi citlivé informace. Tato hrozba je asi nejdůležitějším důvodem pro ochranu uživatelských dat.

Samotné uživatelské účty, respektive jednotliví uživatelé, často vystupují ve více *rolích*. Zde je pak možný jak přístup, kdy se jeden fyzický uživatel přepíná do jiné *role*, tak i přístup, kdy jsou jeho práva dána sjednocením práv všech přiřazených *rolí*. Co vyplývá ze samé podstaty *rolí*, jsou jejich odlišná práva, což může mít důsledky na citlivost vytvářených dat a závažnost kompromitace aplikace s právy této *role*.

[32] Naopak často klesá složitost samotné aplikace, protože propojenost aplikací umožňuje značné sdílení kódu. Vytvořit novou aplikaci je tak snazší.

## 4.2 Uživatelský prostor a SELinux

Ptáme-li se po vztahu *uživatelského prostoru* a *SELinuxu*, jde především o zabezpečení *uživatelského prostoru bezpečnostní politikou*. Zde však nastává jistý problém: oficiální vývoj politiky probíhá v projektu *Referenční politiky*, o které již byla řeč.

Ta je pak upravována a nasazována jednotlivými distribucemi, existuje tedy řada *distribučních politik*. Za distribuci s asi nejlepší podporou *SELinuxu* je považována *Fedora*[33]. *Bezpečnostní politika Fedory*[34] obsahuje mnoho změn a je tak de facto druhým místem oficiálního vývoje.

Kromě toho je třeba vzít v potaz decentralizovaný model vývoje svobodného softwaru. Mimo oficiální vydání *Referenční* či *distribuční politiky* provedli sami vývojáři či nadšenci různé pokusy směřující k uzavření *uživatelského prostoru*. To jsou tedy tři oblasti, ve kterých budeme zkoumat vztah *uživatelského prostoru* a *SELinuxu*.

### 4.2.1 Stav v Referenční politice

Nahlédneme-li do zdrojových kódů *Referenční politiky*, zjistíme, že se soustředí především na zabezpečení *systému* a *uživatelským prostorem* se zabývá jen okrajově. V podstatě jde pouze o `neohraničenou zónu`, *role* a výchozí *SELinux uživatele* a politiky pro několik aplikací.

*Neohraničená zóna* vychází z předpokladu, že nelze napsat *bezpečnostní politiku* pro každou aplikaci. *SELinux* by pak spuštění většiny aplikace zakázal, což není pro běžné nasazení žádoucí. `Neohraničená zóna` tedy v rámci *SELinuxu* simuluje chování linuxového *DAC*. Fungují zde však přechodová pravidla a základní kontroly spustitelné paměti. *Uživatelského prostoru* se především týká *neohraničená zóna* `unconfined_t` Viz [21].

*Referenční politika* definuje několik výchozích *SELinux uživatelů*. Každý z nich pak má přiřazenou *roli* a několik jí odpovídajících *typů* pro *uživatelský prostor*. Takto je vymezen základní rámec *uživatelského prostoru*. Vlastní politika se nachází v *modulech vrstvy* `roles` a dále pak v *modulu* `userdomain` a `unconfined`.

Jako příklad uveďme neprivilegovaného *SELinux uživatele* `user_u`. Jeho *rolí* je `user_r`. Prostřednictvím *šablon modulu* `userdomain` je pak *roli* přiřazen *typ uživatelského shellu* `user_t`, *typ domovského adresáře* `user_home_t` a *typ dočasného adresáře* `user_tmp_t`. Obdobně je řešen neomezený uživatel `unconfined_u`, privilegovaný uživatel `staff_u` či systémový administrátor `sysadmin_u`.

*Referenční politika* uzavírá několik aplikací. Většina z nich jsou nástroje příkazové řádky jako `cdrecord`, `gpg` či `irc`. Mezi *GUI* aplikace pak patří různé konfigurační nástroje jako `sambagui`. Koncových aplikací je jen několik a většinou jde

[33] Domovské stránky projektu *Fedora*: http://fedoraproject.org/.

[34] Zdrojové kódy *Bezpečnostní politiky Fedory* jsou dostupné v repositáři systému správy verzí git: http://git.fedorahosted.org/git/?p=selinux-policy.git;a=summary.

o *GNOME* či *GTK* programy, zmiňme například *moduly*: `evolution`, `mozilla` či `thunderbird`. Politiky pro tyto programy pak stojí za zavedením některých *typů* do uživatelských domovských adresářů jako je `mozilla_home_t`.

Kromě základního rámce a několika aplikací však *Referenční politika* příliš *uživatelský prostor* neuzavírá.

### 4.2.2 Stav v politice Fedory

*Bezpečnostní politika Fedory* se více zaměřuje na uživatelskou přívětivost nasazeného *SELinuxu*. Díky tomu může po instalaci distribuce *SELinux* automaticky běžet. Toto je mimo jiné umožněno příklonem k *neohraničeným zónám*.

*Neohraničená zóna* `unconfined` byla v *bezpečnostní politice Fedory* rozdělena do dvou zón. Vznikla tak *neohraničená zóna pro uživatele* realizovaná v novém *modulu* `unconfineduser` a *neohraničená zóna pro programy*. Viz [22].

Ve Fedoře lze rozlišit čtyři třídy *SELinux uživatelů*: neomezené, běžné, omezené a základní. *Neomezení uživatelé* `unconfined_u` mají skoro vše povoleno, až na `execmod`, `execheap`, `execstack` a `execmem`. Jejich procesy zůstávají v *doméně* `unconfined_t`, pokud je to možné. *Běžní uživatelé* jsou omezeni v tom smyslu, že jejich procesy přecházejí do *omezených domén*, pokud je to možné. Jinak je jejich omezení minimální. V případě `user_u` nemají povoleno spouštění `suid` aplikací. *SELinux uživatel* `staff_u` toto omezení nemá. *SELinux uživatel* `sysadmin_u` pak může většinu věcí jako `root` s tím, že dochází k *doménovým přechodům*, kdykoliv je to možné. Na *omezené uživatele* se vztahuje princip nejmenších možných oprávnění. Příkladem je `guest_u`, který slouží uživatelům připojeným vzdáleně přes *SSH* a `xguest_u`, který smí přistupovat pouze přes *XDM*. V obou případech je zakázaná práce se sítí a spouštění `suid` aplikací. *Základní uživatelé* pak mají ještě menší práva, nemohou ani přistupovat k uživatelským datům. Určeni jsou pro specifické účely, příkladem může být `webadm_u`. Politiky pro omezené a základní *SELinux uživatele* vyžadují před použitím některé úpravy podle svého konkrétního nasazení. Viz [23].

Celkově jsou však *SELinux uživatelé* a *role* podobní těm v *Referenční politice*. Liší se však některými povolenými přístupy a mají asociované *domény* aplikací, které tato politika uzavírá navíc oproti *Referenční politice*.

Mezi tyto nově uzavřené aplikace patří například webový prohlížeč *Chrome*, kancelářský balík *Open Office* či *IM* framework *Telepaty*. Kromě koncových aplikací bylo také uzavřeno několik nástrojů typických pro *Fedoru*.

Speciálním nástrojem používaným ve *Fedoře* ve vztahu k *uživatelskému prostoru* je `sandbox`. Jeho původní myšlenkou je umožnit uživatelům bezpečně zpracovat nedůvěryhodný obsah z příkazové řádky. Jde o *bezpečnostní politiku* doplněnou o jednoduchý *CLI* nástroj. Politika umožňuje `sandboxu` spouštět systémové „binárky“ v *doméně* `sandbox_t`, zakazuje skoro veškeré vytváření a čtení sou-

borů v systému, zakazuje přístup k síti, povoluje zápis na uživatelské terminály a v případě běhu více instancí se od sebe odděluje pomocí *MCS*. Viz [24].

Tento nástroj byl později rozšířen o volbu `-X`, která mu umožňuje spouštět v *doméně* `sandbox_x_client_t` různé *GUI* aplikace. Za tímto účelem vytváří nový domovský adresář i nový dočasný adresář, do kterých se přepne pomocí nástroje `seunshare`. K těmto novým adresářům s *typem* `sandbox_x_file_t` má přístup pro čtení i zápis. Poté pomocí aplikace *Xephyr* spouští nový *X server* s okenním správcem *Matchbox*. Stále funguje oddělování více instancí pomocí *MCS*. Přístup k síti je zakázán, pokud se nepoužije volba `-t sandbox_web_t` pro webový prohlížeč, nebo `-t sandbox_net_t` pro plný přístup k síti. Viz [25].

*Fedora* tedy při uzavírání *uživatelského prostoru* nabrala poněkud jiný směr, spočívající v použití *neohraničené zóny* s možností uzavřít vybranou aplikaci pomocí nástroje `sandbox`.

### 4.2.3 Shrnutí ostatního vývoje

Problémy objevující se při uzavírání *uživatelského prostoru* popisuje Dan Walsh následovně: „Striktní politika a uzavření uživatelů vyžaduje správně nastavené *bezpečnostní kontexty* v uživatelských domovských adresářích. Ukázalo se však, že udržet správné označkování souborů v domovských adresářích je velmi složité. Konečně využití *SELinuxu* k uzavření něčeho vyžaduje ‚*bezpečnostní politiku*'. Tím myslím, že je třeba určit, jaký přístup by měla aplikace či uživatel mít." [26].

Určení žádoucího přístupu však není snadné. Zdá se, že se nikdo neshodne ani na tom, zda povolit prohlížeči pouze přístup k webu, nebo i spouštění pomocných aplikací. Je-li pak touto pomocnou aplikací kancelářský balík, není jasné, zda má přejít do vlastní *domény*, nebo zůstat v *doméně* prohlížeče. Co když nyní v takto spuštěné aplikaci chce uživatel otevřít nějaký soubor. Další problémy jsou s tím, zda povolit webovému prohlížeči nahrávat soubory na web a odkud a zda povolit ukládání souborů z webu a kam. A pak zůstává otázka, jak mají aplikace spolupracovat. Hlavním problémem však je, že se na řešení výše uvedených problémů nikdo neshodne. Viz [26].

Přitom je-li cílem zvýšit bezpečnost při nasazení systému na pracovní stanici, je problém webového prohlížeče přistupujícího k nedůvěryhodnému obsahu na webu zásadní. V moderním *pracovním prostředí* však existuje mnoho komunikačních kanálů, které webový prohlížeč – či jiná aplikace – využívá. Zabránit odhodlanému útočníkovi ve využití chyby webového prohlížeče je tak obtížné. To vše je komplikováno snahou o co „nejpříjemnější" zážitek pro uživatele. Problémy tedy nastávají zejména ve třech oblastech: nahrávání souborů na web a jejich stahování z webu, interakce se spouštěnými pomocnými programy a jejich práva, spouštění aplikací stažených z webu. Viz [27].

Jednou z možností, jak toto řešit, je zaměřit se na zásuvné moduly prohlížeče. Toto je možné, pokud zásuvné moduly běží v jiném procesu než webový prohlížeč,

například s použitím nástroje *nspluginwrapper*. Toto se nakonec díky uživatelské přívětivosti dostalo do *Fedory*. Viz [27].

Jiným příkladem snah o uzavření *uživatelského prostoru* je politika Dominicka Grifta. Ta je založena na kombinaci *Referenční politiky* a *politiky Fedory*, které dále rozšiřuje především směrem k výchozímu uzavření *uživatelského prostoru*. Za tímto účelem důsledně využívá *UBAC*, a to včetně uživatele `root`, a s tím související *prefixy uživatelských typů*. Dále politika rozšiřuje míru uzavření uživatelských aplikací a přidává *moduly* pro další. Viz [28].

Poslední změny v politice Dominicka Grifta zahrnují rozdělení politiky do dílčích politik, z nichž je pro uzavření *uživatelského prostoru* důležitá politika pro základní pracovní stanici s *GNOME*. Jednotlivé *GNOME* aplikace nemohou narušit *SELinux uživatele* ani spouštět obecné aplikace. Funguje tak uzavření mnoha koncových aplikací jako je: *Totem*, *Rhythmbox*, *Evince*, *Compiz*, *Gedit*, *Eclipse* či *Nautilus* a další. Viz [29].

## 4.3 Bezpečnostní cíle politiky pro uživatelský prostor

Jak již bylo zmíněno v sekci 4.1 na straně 28, *uživatelský prostor* obnáší samotné uživatele, *uživatelská data* a programy spouštěné uživateli. Tyto tři oblasti mohou být předmětem útoku či chyby s následky a před obojím má smysl je chránit.

### 4.3.1 Zajištění integrity uživatelských dat

Z myslitelných ohrožení jednotlivých skupin jinými či sebou, pak jsou taková, jejichž závažnost je vyšší. Většina z nich pak souvisí s oblastí dat, protože to jsou právě ona, která bývají cílem útoků, či jim hrozí poškození v důsledku chyby, a jejichž ztráta či vyzrazení by mohly být pro uživatele skutečně závažným problémem. Tak závažné by rozhodně nebylo samo nechtěné ukončení uživatelského procesu, jako nezvratné poškození dat, která zpracovával. Proto se jako nejdůležitější cíl *bezpečnostní politiky* pro *uživatelský prostor* jeví *zajištění integrity uživatelských dat*.

Tento cíl jakožto nejvyšší však vyvolává řadů cílů dílčích, bez kterých jeho dosažení není možné, či které jsou alespoň v některých situacích značně důležité. Zaměříme-li se na to, odkud mohou být *uživatelská data* ohrožena, pak nám v rámci *uživatelského prostoru* [35] vychází, že akutní nebezpečí hrozí především od programů. Nelze však zapomenout ani na sféru uživatelů, kteří také mohou data

[35] Bezpečnost je samozřejmě komplexní problém, tedy *uživatelským datům* hrozí nebezpečí i od *systému* či vnějšího okolí. Obojí je však již nad rámec zaměření této práce. Zmíním alespoň, že *SELinux* zabezpečení *systému* obstojně řeší a že i jiným oblastem ochrany dat se již na mnoha místech patřičná pozornost věnovala. Samozřejmě zabezpečení *uživatelského prostoru* na ostatním zabezpečení závisí, a tedy by mělo následovat ve chvíli, kdy alespoň *systém* je zabezpečen a jsou řešeny i další aspekty bezpečnosti alespoň v rozsahu, který přináleží danému nasazení.

ohrozit. Pro úplnost přidejme ještě možnost ohrožení dat daty. Rozeberme tedy nyní tyto možnosti a vyvoďme dílčí cíle pro ochranu *uživatelských dat*.

### 4.3.2 Ochrana uživatelských aplikací zvláště síťových

Co se týče programů, jsou časté dva scénáře. Prvním z nich je chyba v aplikaci, druhým pak je cílený útok na tuto aplikaci. Dnešní aplikace jsou natolik složité a spolupracují se zbytkem systému i uživatelem tolika způsoby, že je prakticky nemožné, aby byly všechny chyby odstraněny před vlastním nasazením aplikace do provozu. Následně i při seberychlejším opravování známých chyb nelze zabránit tomu, aby uživatel nějakou dobu používal aplikaci obsahující chybu. Přidáme-li k tomu, že jsou chyby, které ještě nebyly nalezeny, [36] lze zodpovědně vyvodit, že v každé aplikaci nejspíše nějaká chyba je. Toto „nejspíše“ by již mělo být dostatečným důvodem pro to, abychom se zabývali ochranou aplikace v módu minimalizujícím dopad chyby na *uživatelská data*, či ostatní běžící aplikace.

Co se útoků týče, i zde často jde o zneužití nějaké chyby v aplikaci. Útoky mohou být lokálního či vzdáleného původu. Ač bývá nad lokálním prostředí často menší moc, než jak se předpokládá, vzdálené prostředí je v tomto ohledu zcela nekontrolovatelné. Vyššímu riziku cíleného útoku budou tedy zřejmě vystaveny aplikace, které přistupují ke vzdálenému obsahu, či umožňují, aby k nim bylo vzdáleně přistupováno. I zde existuje opodstatněná obava, že se všem útokům nedá zabránit, zvláště když využijí neodhalené chyby v aplikaci a ne chyby v jejím nastavení. Tedy i zde existuje dostatečný důvod pro budování takové bezpečnosti, která minimalizuje následky případného průniku. Přibývá tedy první dílčí cíl ochrany *uživatelských dat*, kterým je *uzavření jednotlivých uživatelských aplikací se zvláštním zřetelem na aplikace pracující se sítí*.

### 4.3.3 Využití RBAC k oddělení uživatelů a rolí

Z hlediska hrozeb ze strany uživatelů se nabízí chyba uživatele, který data vlastní, či snaha jiných uživatelů o neoprávněný přístup k těmto datům. Ochránit data před jejich vlastníkem dokáže snad akorát pravidelné zálohování, toto je tedy z hlediska *bezpečnostní politiky SELinuxu* neřešitelné. Základní ochranu před jinými uživateli pak poskytuje linuxové *DAC*. Toto nemusí být nutné vždy řešit, je-li uživatelem počítače jen jedna osoba, i když i v takovém případě může být užitečné posílit *DAC* o prvky *RBAC* alespoň na úrovni `root` versus onen uživatel. Obzvláště v korporátní sféře opravdu má smysl se tímto zabývat. Přibývá tedy druhý dílčí cíl, kterým je *důkladné oddělení uživatelů, rolí a jimi spouštěných procesů pomocí důsledné aplikace RBAC*.

Ohrožení *uživatelských dat* jinými daty v podstatě spočívá v tom, že tato data budou obsahovat nějaký škodlivý kód, který zneužije chybu aplikace, jež data

[36] Nalezeny ve smyslu, že o nich vědí vývojáři aplikace.

zpracovává. Takto se tento problém redukuje na zabezpečení uživatelských aplikací. Druhou možností je snaha spustit program získaný nedůvěryhodnou cestou, což dokáže *SELinux* řešit. V praxi to však vyžaduje uplatnit na uživatele přístupy striktní politiky, což se kvůli uživatelské přívětivosti málokdy dělá.

### 4.3.4 Důraz na ochranu citlivých dat a programů s nimi pracujících

Nyní je třeba se ještě vrátit k hlavnímu cíli, tedy *zajištění integrity uživatelských dat* a rozebrat si ho podrobněji. Oblast *uživatelských dat*, jak už bylo ostatně řečeno v sekci 4.1 na straně 28, není nijak jednolitá. Po stránce původu tu jsou tedy data pocházející přímo od uživatele a data pocházející od programu, jako jsou logy jeho činnosti či uložené nastavení. Po stránce důležitosti pak mohou některá data být pro uživatele zásadnější než jiná, což je však často značně individuální. Na druhou stranu to, že soukromé klíče uživatele či jeho hesla k webovým účtům jsou z bezpečnostního hlediska zásadní, je snad jasné. Z toho tedy plyne třetí dílčí cíl, kterým je *důsledné využití TE k rozdělení uživatelských dat na co nejvíce typů.* S tím souvisí konečně i čtvrtý dílčí cíl, kterým je *důraz na zabezpečení aplikací, které pracují s citlivějšími typy dat.*

### 4.3.5 Cíle zvolené pro vytvářenou politiku

Po předchozí úvaze a vzhledem k časovému rámci této práce jsem pro svou politiku zvolil následující *bezpečnostní cíle*:

- *Zajištění integrity uživatelských dat* je stále prioritním cílem.
- *Důsledné využití TE k rozdělení uživatelských dat na co nejvíce typů* má výše uvedený hlavní cíl podpořit.
- *Uzavření uživatelských aplikací se zvláštním zřetelem na aplikace pracující se sítí* považuji za nezbytné k dosažení hlavního cíle.
- *Důraz na zabezpečení aplikací pracujících s citlivými daty* hodlám dodržet zejména u služeb využívaných koncovými aplikacemi.

Pro tuto svou práci jsem nezvolil cíl *využití RBAC k oddělení uživatelů a rolí* zejména proto, že se soustředím na koncovou uživatelskou pracovní stanici. Tam je přínos *RBAC* příliš malý na to, aby ve vymezeném čase vyvážil obtíže spojené s důslednou implementací *RBAC prefixů typů* do jednotlivých *modulů* politiky. Považuji však tento cíl za zajímavý námět na rozšíření pro někoho více korporátně orientovaného.

# 5 Analýza uživatelského prostoru

Máme-li vytvářet *bezpečnostní politiku* pro vybrané aplikace v *uživatelském prostoru*, musíme tento nejprve analyzovat. Tato analýza se pak zvláště soustředí na část *uživatelského prostoru*, kterou je *pracovní prostředí KDE*. Nejprve však je namístě stručná charakteristika *prostředí KDE* a jeho částí. Poté přejdu k samotné analýze tohoto prostředí, jeho jednotlivých aplikací a služeb a konečně i vztahů uvnitř prostředí a vztahů prostředí jako celku ke zbytku *uživatelského prostoru*. Účelem této analýzy je porozumění oblasti, pro kterou vytvářím *bezpečnostní politiku*, a převedení tohoto porozumění do návrhu vyvíjené politiky.

## 5.1 Charakteristika prostředí KDE

Protože se má *bezpečnostní politika* zaměřuje na *prostředí KDE*, je namístě ho charakterizovat. V následujících sekcích se tedy budu věnovat jak popisu prostředí jako celku, tak i popisu některých jeho důležitých částí. Zde je třeba zmínit, že tento popis nemá být vyčerpávajícím popisem *prostředí KDE* ani jeho aplikací, ale má přinést představu o tom, co dané prostředí je, k čemu se jeho aplikace používají, jaké služby v něm běží a podobně. Toto seznámení pak koncipuji tak, že dávám větší prostor aplikacím a službám, na které se při vytváření *bezpečnostní politiky* zaměřuji. Ostatní aplikace, které s danými nějak souvisí pak popisuji stručněji, a ty zbývající jen zmiňuji pokud jsou nějak zásadní, nebo zcela vynechávám.

### 5.1.1 Prostředí KDE

To, co je v běžné mluvě a této práci označováno jako *prostředí KDE*, je tvořeno třemi částmi: *vývojovou platformou KDE*, *grafickými prostředími KDE* a *aplikacemi KDE*. Toto vše vyvíjí komunita označkovaná jako *KDE*. Zároveň se pro toto vše používá, nutno podotknout že ne zcela správně, označení *KDE*. Vybraná část vývojové platformy, grafických prostředí a aplikací je pak v půlročním cyklu vydávána jako *„KDE Software Compilation“*, zkráceně *KDE SC*. Viz [30, 31].

*Vývojovou platformu KDE* tvoří jednotlivé *knihovny KDE* a řada služeb jako, je *PIM systém Akonadi*, *sémantický desktop Nepomuk*, nejen síťový vstupně-výstupní *framework KIO*, *vykreslovací jádro KHTML* či hardwarová vrstva *Solid*. *Vývojová platforma KDE* tak slouží ke sdílení funkcionality mezi *KDE aplikacemi* a zprostředkování technologií, na kterých jsou pak tyto aplikace vystavěny. Tím usnadňuje a zefektivňuje vývoj jednotlivých aplikací a spojuje je do jednotného prostředí. Samotná platforma staví na *Qt toolkitu* a lze ji provozovat nejen na *GNU/Linuxu*, ale na jiných operačních systémech, a to nejen pro PC, ale i pro mobilní zařízení. Viz [30, 32].

*Grafická prostředí KDE* zahrnují okenního správce *KWin*, správce relace, menu, spouštěč aplikací *KRunner*, panel či samotnou aplikaci plochy *Plasma*. Umožňují

spouštění libovolných aplikací a práci s nimi. Kvůli přizpůsobení jednotlivým koncovým zařízením a způsobům práce vzniklo několik *grafických prostředí KDE*. Pro stolní počítače a notebooky je tak určeno prostředí *Plasma Desktop*, pro menší zařízení, jako netbooky či tablety, zase *Plasma Nebook*. Viz [30, 33].

*Aplikace KDE* pak tvoří různorodou skupinu programů využívajících *platformu KDE* a určených pro všechny představitelné účely. Tyto aplikace mají často i vlastní vývojové cykly a mnoho z nich ani netvoří pravidelné vydání *KDE SC*. Rozděleny jsou do několika skupin, jako je vývoj, výuka, hry, grafika, internet, multimédia, kancelář a další. Některé pak tvoří dílčí celky jako *KDE PIM*, *KDE Network* a jiné. Viz [30, 31, 34].

### 5.1.2 PIM systém Akonadi

*Akonadi* je framework sloužící jako lokální vyrovnávací paměť pro různé zdroje *PIM* dat. Poskytuje tak jednotné *API* pro práci se vzdálenými poštovními servery, groupwarem, různými online službami, ale i místními soubory kalendáře či adresáře. *Akonadi* tedy samo žádná *uživatelská data* neukládá, pouze k datům již někde uloženým poskytuje jednotný přístup. Efekt fungování je podobný jako u vyrovnávací paměti. Tím zrychluje práci s těmito daty, a také zpřístupňuje offline data uložená vzdáleně. *Akonadi* zároveň poskytuje o zpřístupněných datech různá metadata. Díky *Akonadi* lze k *PIM* datům přistupovat i z jiných než původních aplikací, a to včetně aplikací mimo svět *KDE*. Metadata patřící *Akonadi* jsou pak uložena v databázi a k hledání v nich se používá *Nepomuk*. Dalšími funkcemi *Akonadi* je přístup k datům na pozadí nezávisle na uživatelském rozhraní klienta a víceprocesový návrh. Viz [35, 36, 37].

V současné době [37] je *Akonadi* plně využíváno pouze adresářem *KAddressBook*. Ostatní aplikace *KDE PIM* ho využívají jen v menší míře. Práce na integraci do dalších aplikací, zejména poštovního klienta *KMail*, je však již ve finální fázi.

### 5.1.3 Sémantický desktop Nepomuk

*Nepomuk* je služba *prostředí KDE* zpřístupňující *sémantická data*. Tato data jsou uložena ve formě *RDF* v backendu *Soprano* a pro jejich definici jsou používány standardizované *ontologie*. *Nepomuk* získává *sémantická data* dvěma způsoby: ručním značkováním uživatelem a automatickým indexováním pomocí *Strigi*. Cílem *Nepomuku* je zpřístupnit *sémantická data* různým aplikacím a uživatelům, vyhledávat v nich a sdílet je. Jedním ze zdrojů *sémantických dat* pro *Nepomuk* je i *PIM* systém *Akonadi*. Viz [37].

[37] Práci na *bezpečnostní politice* pro *KDE aplikace* jsem začal s *KDE SC 4.4.x* a následně rozvíjel s *KDE SC 4.5.x*. V době vydání *KDE SC 4.6.1* byla situace ohledně integrace *Akonadi* do *KDE PIM* stále stejná.

### 5.1.4 Klíčenka KWallet

*KWallet* je služba úschovny hesel pro *pracovní prostředí KDE* spravovaná pomocí nástroje *KWalletManager*. Slouží k bezpečnému uložení hesel do šifrovaných souborů. Uložená hesla lze použít k lokální i vzdálené autentizaci a ostatním aplikacím jsou dostupná až poté, co uživatel zadá heslo klíčenky, a tak ji odemkne. S *KWallet* je integrována řada *KDE aplikací* mimo jiné prohlížeč *Konqueror* či poštovní klient *KMail*. Viz [38].

### 5.1.5 Správce osobních informací Kontact

*Kontact* pomocí technologie *KParts* integruje jednotlivé aplikace *KDE PIM* v podobě komponent. Komponentami *Kontactu* jsou *RSS* čtečka *Akregator*, adresář *KAddressBook*, záznamník nápadů *KJots*, poštovní klient *KMail*, čtečka novinek *KNode*, poznámky na ploše *KNotes*, kalendář a plánovač *KOrganizer* a sledovač využití času *Ktimetracker*. K nim přidává ještě sumární pohled na důležité dění v jednotlivých komponentách. *Kontact* tedy spojuje aplikace pro jednotlivé účely do rozsáhlého groupwarového nástroje. Viz [39].

### 5.1.6 Poštovní klient KMail

*KMail* je výchozí poštovní klient *prostředí KDE*. Může být použit jako samostatná aplikace, nebo jako komponenta pro *Kontact*. Podporuje standardní poštovní protokoly *IMAP*, *POP3* a *SMTP*, textovou i zabezpečenou autentizaci, šifrování pomocí *PGP/MIME*, *S/MIME* a *OpenPGP*. Krom toho integruje kontrolu pravopisu, filtrování spamu nebo uchování hesel v klíčence *KWallet*. Dále je vybaven i groupware funkcionalitou. Viz [40].

### 5.1.7 Adresář KAddressBook

*KAddressBook* je správce kontaktů pro *KDE*. Od verze z *KDE SC 4.4.0* přistupuje k datům adresáře prostřednictvím *Akonadi*. Díky tomu podporuje ukládání kontaktů do různých zdrojů od lokálních souborů po vzdálené servery. Stejně jako *KMail* může být používán samostatně nebo v rámci *Kontactu*. Kromě toho se i těsněji integruje s poštovním klientem *KMail*. Viz [41].

### 5.1.8 Webový prohlížeč Konqueror

*Konqueror* je víceúčelová aplikace sloužící k prohlížení webu s jádrem *KHTML* či *Webkit*, prohlížení různých souborů díky *KParts* a *KIO* nebo správě souborů. K jeho obecným funkcím patří záložky, podpora více karet a rámců a správa sezení. Samozřejmostí je integrace s dalšími *KDE aplikacemi* a *službami*, ať už jde o *KWallet*, *Akregator* nebo správce stahování *KGet*. Od uvedení *KDE 4* je akcentována především jeho funkce webového prohlížeče. Viz [42].

## 5.2 Analýza aplikací a služeb prostředí KDE

Po stručném seznámení s jednotlivými aplikacemi můžeme přistoupit k vlastní analýze. Pro vytváření *bezpečnostní politiky* je třeba znát, s jakými objekty která aplikace pracuje, které služby prostředí a systému využívá, a se kterými dalšími aplikacemi a jak spolupracuje. Po pochopení vztahů uvnitř prostředí zbývá ještě zaměřit se na vztahy *prostředí KDE* jako takového ke zbytku *uživatelského prostoru*. Při této analýze se plně soustředím na aplikace, které jsem pro tvorbu politiky vybral.

### 5.2.1 Poštovní klient KMail

Spustitelné soubory *KMailu* jsou umístěny v adresáři `/usr/bin/`. Kromě vlastní „binárky“ `kmail` jde o pomocné „binárky“ `kmailcvt` a `kmail-migrator` a několik skriptů `kmail_*.sh`. Poštovní klient *KMail* ukládá uživatelská nastavení do řady konfiguračních souborů: `kmailrc`, `mailtransports`, `emaildefaults` a `emailidentities`. Ty se nacházejí v adresáři `~/.kde/share/config/`. Uživatelské e-maily pak *KMail* ukládá do adresáře `~/.kde/share/apps/kmail/`.

*KMail* komunikuje prostřednictvím sítě s poštovními servery prostřednictvím standardních protokolů a jim odpovídajících portů. Pro komunikaci s ostatními aplikacemi využívá *D-Bus*. Protože jde o GUI aplikaci, využívá také *X server*.

Hesla k účtům může *KMail* ukládat do *KWallet*, v omezené míře pak využívá službu *Akonadi* a poměrně těsně se integruje s adresářem *KAddressBook*. *KMail* může být spuštěn v rámci aplikace *Kontact* a dále komunikuje i s ostatními *KDE PIM* aplikacemi.

### 5.2.2 Adresář KAddressBook

„Binárkou“ aplikace *KAddressBook* je soubor `/usr/bin/kaddressbook`. Program obstarávající migraci kontaktů má tamtéž „binárku“ `kaddressbookmigrator`. Soubory s nastavením vlastního programu `kaddressbookrc` a jeho pomocníka `kaddressbookmigratorrc` se nacházejí v adresáři `~/.kde/share/config/`. Výchozím adresářem pro ukládání lokálních souborů adresáře, tedy vlastních *uživatelských dat*, je `~/.kde/share/apps/kabc/`.

`KAdressBook` je grafická aplikace, využívá tedy *X server*, jako ostatní *KDE aplikace* pak umožňuje komunikaci prostřednictvím rozhraní *D-Bus*.

Kromě úzké integrace s *KMailem* se *KAddressBook* integruje do *Kontactu* a spolupracuje s dalšími *KDE PIM* aplikacemi. Mimo to značně využívá *PIM* systém *Akonadi*.

### 5.2.3 Webový prohlížeč Konqueror

Spustitelný soubor *Konqueroru* je `/usr/bin/konqueror`. Konfigurační soubory programu se nacházejí v adresáři `~/.kde/share/config` a jde o `konquerorrc`, `konq_history` a `konqsidebartng.rc`. Další data jako záložky, uzavřené karty či sezení si *Konqueror* ukládá do adresáře `~/.kde/share/apps/konqueror`.

Jako webový prohlížeč přistupuje *Konqueror* ke vzdálenému obsahu prostřednictvím standardních protokolů. Díky technologii *KIO* pak nemusí přistupovat pouze k samotnému webu, ale může využít i další síťové služby. I *Konqueror* využívá *D-Bus* a *X server*.

*Konqueror* umožňuje ukládání hesel do klíčenky *KWallet*, komunikuje s ní prostřednictvím rozhraní *D-Bus*. Integruje se také s dalšími *KDE aplikacemi*: například po kliknutí na emailovou adresu spustí *KMail* pro odeslání zprávy.

### 5.2.4 Klíčenka KWallet

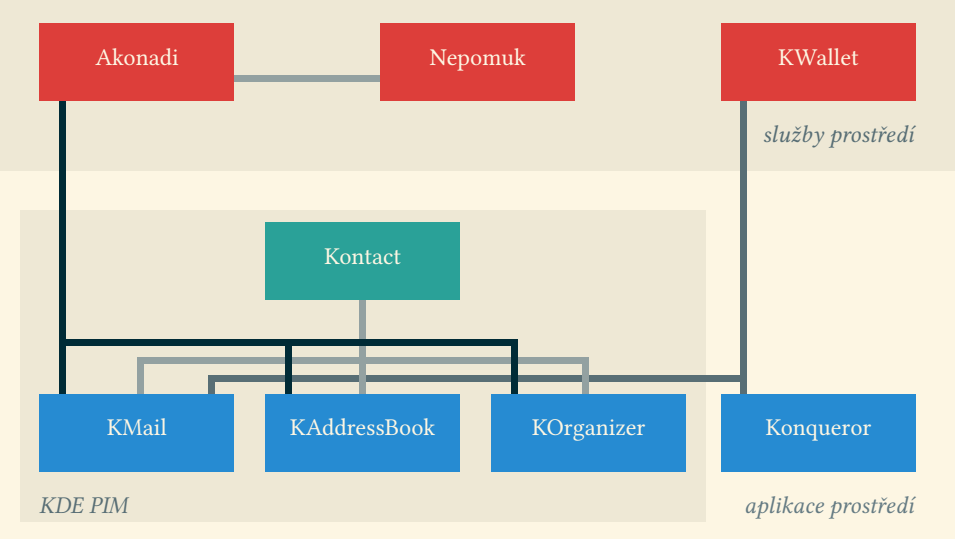
Klíčenka *KWallet* má dvě „binárky“ v adresáři `/usr/bin`, a to „binárku“ samotné služby `kwalletd` a „binárku“ grafického rozhraní k ní `kwalletmanager`. Konfigurační soubory klíčenky `kwalletrc` a jejího správce `kwalletmanagerrc` jsou v adresáři `~/.kde/share/config`. V adresáři `~/.kde/share/apps/kwallet` se nalézají zašifrované soubory obsahující uložená hesla.

Samotná služba `kwalletd` je *CLI* aplikace, nástroj *KWalletManager* je pak grafický a tedy využívá *X server*. Aplikace, které umožňují ukládání hesel do klíčenky, s ní komunikují přes *D-Bus*.

### 5.2.5 PIM systém Akonadi

Spustitelné soubory *Akonadi* se nacházejí v adresáři `/usr/bin`. Jde o samotnou „binárku“ serveru `akonadiserver` a nástroje pro práci s ní či její ladění jako `akonadictl` či `akonadiconsole`. Pak je tu skupina „binárek“ *agentů Akonadi* a *zdrojů Akonadi*, které se starají o komunikaci se vzdálenými a lokálními úložišti *PIM* dat. K monitorování stavu *Akonadi* uživatelem z *GUI* slouží nástroj spouštěný v informační části panelu `akonaditray`. *Akonadi* má většinu konfiguračních souborů v adresáři `~/.config/akonadi`, několik konfiguračních souborů pojmenovaných `akonadi*rc` se také nachází v adresáři `~/.kde/share/config`. Tyto obsahují nastavení *zdrojů Akonadi* a také průvodce prvním spuštěním. Konfigurační soubory v `~/.config/akonadi` se týkají přímo úložiště vyrovnávací paměti a *agentů Akonadi* pro jednotlivé zdroje. Vlastní data, tedy obsah vyrovnávací paměti a metadata, ukládá *Akonadi* do databáze, která standardně sídlí v adresáři `~/.local/akonadi`.

Jako úložiště vyrovnávací paměti a metadat slouží standardně *MySQL* databáze. Většina nástrojů pro práci s *Akonadi* jsou *CLI* aplikace. Výjimku tvoří *GUI*




Obrázek 1: Vztahy mezi vybranými *KDE aplikacemi* a *službami*

konfigurační a monitorovací nástroje, které tedy potřebují přístup k *X serveru*. Některé ze *zdrojů Akonadi* přistupují přes síť ke vzdálenému obsahu. *Akonadi* také využívá *D-Bus*.

*Akonadi* slouží jednotlivým aplikacím z *KDE PIM*. V současné době je plně využíváno pouze adresářem *KAddressBook*. Ostatní *KDE PIM* aplikace využívají jen část funkcionality *Akonadi*. Z důvodu potřeby vyhledávání a kvůli možnostem správy metadat se *Akonadi* dále integruje se sémantickým desktopem *Nepomuk* prostřednictvím několika *zdrojů Akonadi*.

### 5.2.6 Vztahy aplikací v rámci prostředí KDE

Jak je patrné z předchozích rozborů, *prostředí KDE* dodržuje jistá pravidla pro ukládání konfiguračních souborů a dat aplikací. Až na výjimky se většina souborů, se kterými *KDE aplikace* interně pracují, nachází v adresáři `~/.kde`. V adresáři `~/.kde/share/config` jsou uloženy konfigurační soubory, soubory s daty aplikací se nacházejí v adresáři `~/.kde/share/apps`. Služby prostředí pak mají konfigurační soubory v adresáři `.config` a data v adresáři `.local`.

Interakci vybraných aplikací a služeb zachycuje obrázek 1 na straně 41. Jednotlivé aplikace *KDE PIM* spolupracují mezi sebou a zároveň mohou být integrovány do správce *PIM* informací *Kontact*. *KDE PIM* aplikace také v různé míře využívají *PIM* systém *Akonadi*. Poštovní klient *KMail* a webový prohlížeč *Konqueror* používají úschovnu hesel *KWallet*. Služba *Akonadi* spolupracuje se sémantickým desktopem *Nepomuk*.

### 5.2.7 Vztah prostředí KDE k uživatelskému prostoru

Vývoj *prostředí KDE* probíhá v souladu se sjednocujícími iniciativami projektu *freedesktop.org*. Ten se snaží o zvýšení interoperability mezi jednotlivými *pracovními prostředími* běžícími v *X Window systému*. Pro tuto analýzu je důležité používání sběrnice *D-Bus* a dodržování specifikace *základních adresářů XDG*. S tímto ještě souvisí specifikace *uživatelských adresářů XDG*. Její využívání je sice v kompetenci distribuce, ale pokud ta ji používá, *KDE* s těmito adresáři také pracuje.

*D-Bus* je sběrnice pro zasílání zpráv meziprocesové komunikace. Kromě toho umožňuje snadnou správu spouštění a ukončování aplikací, takže se hodí pro tvorbu démonů a spouštění aplikací na vyžádání. *D-Bus* poskytuje jak systémovou sběrnici pro komunikaci událostí směrem od systému k aplikacím, tak sběrnice pro jednotlivá sezení, které slouží pro komunikaci mezi uživatelskými aplikacemi. *D-Bus* lze použít jak k lokální komunikaci, tak ke vzdálené komunikaci přes nezabezpečené *TCP/IP* spojení. Viz [43].

Specifikace *základních XDG adresářů* stojí na několika konceptech. Definuje jeden adresář určený pro ukládání uživatelských dat. Ve výchozím nastavení jde o `~/.local/share`. Dále definuje jeden adresář určený pro ukládání uživatelského nastavení. Ve výchozím stavu jde o `~/.config`. Třetí definovaný adresář slouží k ukládání nedůležitých uživatelských dat a funguje jako vyrovnávací paměť. Ve výchozím nastavení jde o `~/.cache`. Pro soubory vytvářené uživatelskými aplikacemi za běhu pak slouží další adresář. Kromě těchto jedinečných adresářů umístěných v uživatelském domovském adresáři specifikace určuje ještě jejich systémové obdoby. Cesty, na které jsou *základní XDG adresáře* nastavené, jsou uložené v globálních proměnných a mohou být při splnění určitých podmínek měněny. Viz [44].

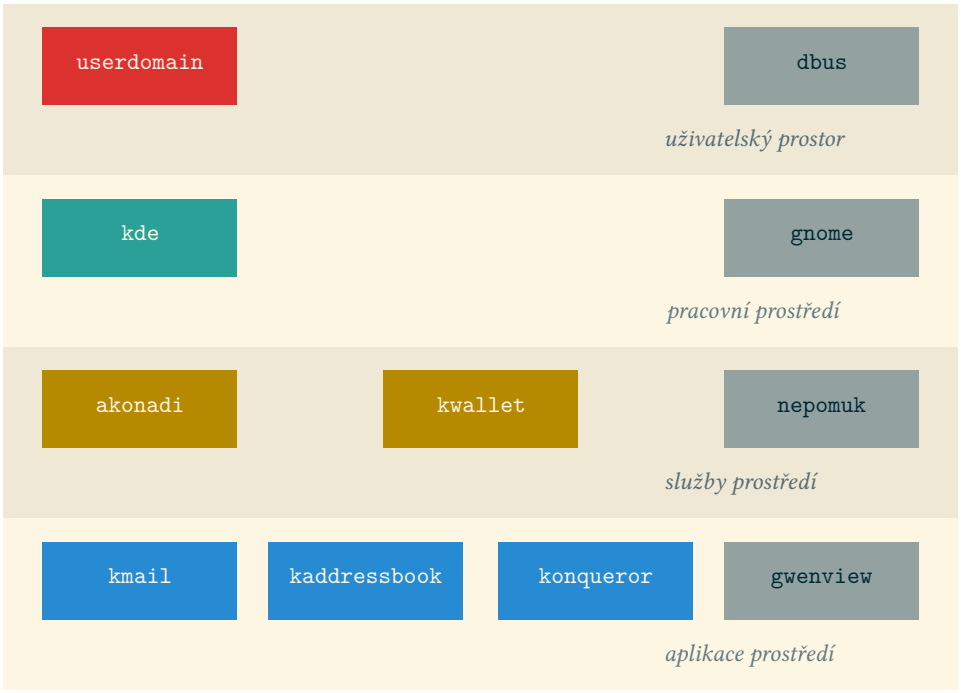
Další obecně známé adresáře pro uživatelská data se nastavují pomocí nástroje `xdg-user-dirs`. Tento je spouštěn časně při přihlášení uživatele a umožňuje vytvořit *uživatelské XDG adresáře*, pokud ještě neexistují. Adresáře jako je adresář plochy, adresář hudby či adresář dokumentů se vytvářejí v aktuální jazykové mutaci a opět jim odpovídají globální proměnné. Viz [45].

## 5.3 Převedení výsledků analýzy do návrhu politiky pro prostředí KDE

V této části využiji poznatky z předchozí analýzy k vytvoření návrhu *bezpečnostní politiky*. Půjde o architekturu mnou vytvářené politiky. Dále stanovím obecný návrh *modulu* této politiky. A konečně uvedu, které *moduly* budu vytvářet.

### 5.3.1 Celkový návrh bezpečnostní politiky

Pro svou politiku jsem zvolil čtyřvrstvou architekturu tak, jak ji ukazuje obrázek 2 na straně 43. Nutno podotknout, že jde o logickou architekturu. Umístění jednotlivých *modulů* do *vrstev Referenční politiky* odpovídá jejím zvyklostem – většina




Obrázek 2: Architektura politiky pro *prostředí KDE*

*modulů* je tedy ve *vrstvě* `apps`. Ve schématu barevně vyznačené *moduly* jsem vyvíjel či upravoval v rámci této práce, *moduly* vyznačené šedě jsem do schématu zahrnul pro zachování přiměřeně širokého kontextu.

Jak ukazuje obrázek 2, jednotlivé logické *vrstvy* jsou následující:

- *aplikace prostředí*,
- *služby prostředí*,
- *pracovní prostředí* a
- *uživatelský prostor*.

*Vrstva aplikací prostředí*, jak napovídá její název, zahrnuje všechny uživatelské aplikace daného *pracovního prostředí*. Pro *prostředí KDE* jde například o webový prohlížeč *Konqueror*, poštovní klient *KMail* a další *KDE PIM aplikace* či prohlížeč obrázků *Gwenview*. Jednotlivé *moduly* této *vrstvy* mohou spolupracovat mezi sebou podle požadavků daných aplikací. Dále se očekává, že budou využívat služby poskytované *moduly* z vyšších *vrstev*.

*Vrstva služby prostředí* sdružuje jednotlivé služby, které *pracovní prostředí* poskytuje. V *prostředí KDE* je to zejména *PIM* systém *Akonadi*, sémantický desktop *Nepomuk*, klíčenka *KWallet* či hardwarová vrstva *Solid*. *Moduly* na této *vrstvě* také

spolupracují podle provázanosti konkrétních služeb. Tyto *moduly* využívají služeb poskytovaných *moduly* vyšších *vrstev*, ale také nabízejí své služby *vrstvě aplikací prostředí*.

*Vrstva pracovní prostředí* obsahuje zastřešující *modul* `kde`. Analogicky by do této vrstvy spadal zastřešující *modul* `gnome`, který však není součástí vyvíjené politiky. Účelem této *vrstvy* je poskytovat pro prostředí specifické služby *modulům* nižších vrstev. Zejména jde o *typy* oddělující prostředí od zbytku *uživatelského prostoru*.

*Vrstva uživatelský prostor* poskytuje funkcionalitu společnou všem prostředím. Jde tedy o již existující *modul Referenční politiky* `userdomain` nebo o uživatelskou část *modulu* `dbus` a případné další potřebné *moduly*. V této *vrstvě* se očekávají především změny a rozšiřování stávající infrastruktury podle požadavků jednotlivých prostředí.

Tato logická architektura je využitelná nejen pro vytváření politiky *prostředí KDE*, ale i pro ostatní *pracovní prostředí* v *GNU/Linuxu*.

### 5.3.2 Obecný návrh modulu politiky

Obecný návrh *modulu* vytvářené politiky respektuje konvence dodržované *Referenční politikou*. Pro každý *modul* se tak vytvářejí tři soubory se zdrojovými kódy: `modul.te`, `modul.if` a `modul.fc` s odpovídající strukturou a řádným pojmenováním *typů* a *rozhraní*.

Definované *typy* pak odpovídají zjištěním z prováděné analýzy. Důsledné *vynucování typů* je realizováno prostřednictvím oddělení *typů* jednotlivých *modulů*, jenž jsou dále odděleny podle toho, jaký druh dat tyto *typy* obsahují. To, jak ukázala analýza, závisí na jejich umístění.

Konfigurační soubory aplikací mají *typ* podle vzoru `modul_config_home_t`, pokud se nacházejí v adresáři `~/.config`, nebo `modul_home_conf_t`, pokud se nacházejí v adresáři `~/.kde/share/config`. Vlastní data aplikací pak používají buď *typ* `modul_data_home_t`, pokud jsou uložena v adresáři `~/.local`, nebo `modul_home_data_t`, pokud se nacházejí v adresáři `~/.kde/share/apps`.

### 5.3.3 Vytvářené moduly politiky

Mezi vytvářené *moduly* ve *vrstvě aplikací prostředí* patří *modul* `kmail` pro poštovní klient *KMail* a `kaddressbook` pro adresář *KAddressBook*. Dále to této *vrstvy* patří upravovaný *modul* `konqueror` pro webový prohlížeč *Konqueror*.

Ve *vrstvě služeb prostředí* jsem pak vytvářel *moduly* `akonadi` pro *PIM* systém *Akonadi* a `kwallet` pro klíčenku *KWallet*.

Jediným *modulem* ve *vrstvě pracovního prostředí* je *modul* `kde`, který zastřešuje celé *prostředí KDE*. Tento *modul* byl upravován oproti stavu v mé bakalářské práci.

V nejvyšší *vrstvě uživatelského prostoru* jsem pracoval na změnách v *modulu* `userdomain`. Vzhledem k odlišnostem mezi *Referenční politikou* a *politikou Fedory* si toto vyžádalo vytvoření *modulu* `fedora-compat` a napsání patche pro *modul userdomain Referenční politiky*.

# 6 Přístupy k uzavírání uživatelského prostoru

Tato sekce se bude zabývat strategiemi vývoje *bezpečnostních politik* směřujících k uzavření *uživatelského prostoru*. Nejprve popíši přístup uzavírající *uživatelský prostor* „shora dolů“, tj. od služeb *uživatelského prostoru* směrem ke koncovým aplikacím. Poté se zaměřím na přístup uzavírající *uživatelský prostor* „zdola nahoru“, to jest směrem od koncových aplikací ke službám *uživatelského prostoru*. Nakonec budu argumentovat, proč jsem k vytvoření *bezpečnostní politiky* pro vybrané služby a aplikace *prostředí KDE* vybral přístup postupující „zdola nahoru“.

## 6.1 Tvorba politiky „shora dolů“

V této sekci se blíže podívám na přístup tvořící *bezpečnostní politiku SELinuxu* „shora dolů“. Nejprve popíši, co tímto přístupem míním, a poté se zaměřím na jeho vlastnosti, možné přínosy a nedostatky.

### 6.1.1 Vymezení přístupu tvorby politiky „shora dolů“

Podíváme-li se znovu na obrázek 2 na straně 43, uděláme si představu o vývoji politiky přístupem „shora dolů“ poměrně snadno. Tento přístup tedy nejprve uzavírá společné *moduly* služeb *uživatelského prostoru*, poté pokračuje sjednocujícím *modulem* pro vybrané *pracovní prostředí*, následuje uzavřením služeb, které toto *pracovní prostředí* poskytuje, a končí tvorbou politiky pro jednotlivé uživatelské aplikace. Jde tedy o přístup postupující od obecného ke konkrétnímu.

### 6.1.2 Vlastnosti, možné přínosy a nedostatky

Tento přístup plynule navazuje na již vytvořené části *Referenční politiky* či *politiky Fedory* a respektuje také obecně používaný princip vývoje *pracovního prostředí*, který začíná nejprve vytvořením knihoven a pak teprve nad nimi vytváří vlastní aplikace. V průběhu vývoje by tak nemělo docházet k zásadním změnám v kódu politiky, respektive by tyto změny neměly vyvolat rozsáhlou druhotnou potřebu na změnu závislého kódu.

Jak ale ukázal například vývoj *pracovního prostředí KDE 4*, jde o metodu, které dlouho trvá, než začne dávat nějaké pro koncové uživatele viditelné a jimi žádané výsledky. Vztáhne-li se navíc tato metoda na tvorbu *bezpečnostní politiky SELinuxu*, narazí zpočátku na značné problémy s obtížností reálného testování. Dá se

totiž očekávat, že rané fáze politiky budou obsahovat řadu chyb, a bude-li první politikou politika pro samotné prostředí, bude jejich dopad zásadně ovlivňovat funkčnost, a tedy i vůli uživatelů testovat.

Z hlediska cílů politiky pro *uživatelský prostor* definovaných v sekci 4.3 na straně 33 je třeba říci, že je tento přístup zpočátku vůbec nenaplňuje. S *uživatelskými daty* pracují koncové aplikace. Aby bylo možné zajistit jejich integritu, musí tento přístup dojít do své poslední fáze, tedy musí prakticky vytvořit politiku pro celý *uživatelský prostor*. Stejně tak, aby politika efektivně chránila koncové aplikace, musí přístup dojít do své poslední fáze. Naopak pro důsledné využití *RBAC* je tento přístup vhodný, protože to může být implementováno ihned od nejvyšší vrstvy. Dosažení důkladné ochrany citlivých dat a programů s nimi pracujících ale opět znamená dospět v aplikaci přístupu „shora dolů“ až do jeho poslední fáze.

## 6.2 Tvorba politiky „zdola nahoru“

V této sekci se blíže podívám na vývoj politiky přístupem „zdola nahoru“. Nejprve opět přiblížím, co tímto označením myslím. Poté se zaměřím na charakteristiku tohoto přístupu z hlediska možných problémů a přínosů.

### 6.2.1 Vymezení přístupu tvorby politiky „zdola nahoru“

Nahlédnutí na obrázek 2 na straně 43 opět značně pomůže v pochopení přístupu budujícího politiku „zdola nahoru“. Jako první se nyní vytvářejí *moduly* pro koncové aplikace. Při jejich tvorbě vznikají požadavky na uzavření logicky vyšších celků, jako jsou služby prostředí, samo prostředí či se objevuje nutnost provést změny až na úrovni *uživatelského prostoru*. Po jejich bezprostředním vyřešení se vývoj vrací na úroveň aplikací, kde opět generuje další požadavky na uzavření vyšších úrovní. Takto se tedy postupně uzavírá *uživatelský prostor* odspodu. Jde tedy o přístup postupující od konkrétního k obecnému po vzoru Baconových postupných indukcí.

### 6.2.2 Vlastnosti, možné přínosy a nedostatky

Tento přístup vychází z předpokladu existence nějakého výchozího neznámého stavu – již vytvořené funkční *pracovní prostředí*. Naším cílem je tento stav prozkoumat, pochopit a popsat – vytvořit *bezpečnostní politiku*. Začíná tedy pracovat s jednotlivostmi, u kterých se dá případná chyba snadno odhalit – koncovými aplikacemi. Takováto metoda však sama o sobě počítá se situací, kdy se dosud přijímané závěry ukáží jako neoprávněné, a bude tedy nutné je opravit či přijmout nové – pozměnit již vytvořenou politiku.

Rizikem této metody se tak jeví možnost nutnosti provést rozsáhlou revizi kódu politiky, pokud požadavky na vyšší logické úrovně ukáží, že se některý pro-

blém dosud řešil špatně. Toto si může vyžádat zvláště rozsáhlé změny kódu v případě, že k takovému odhalení dojde až v pozdní fázi vývoje. Metoda postupných indukcí však byla navrhována s ohledem na tuto možnost, takže užívá-li se jí správně – tedy jsou-li postupné kroky malé a nevyvozují-li se hned dalekosáhlé závěry – povede se tyto případy odhalit včas a nutnost změny se tak co do rozsahu minimalizuje.

Z hlediska reálného testování nese však tato metoda výhodu v tom, že brzy produkuje použitelné výsledky na úrovni politiky pro koncové aplikace. Tyto politiky tak mohou být včas podrobeny důkladnému testování ve skutečném nasazení. To dále posílí princip postupných indukcí generováním příslušných požadavků na funkcionalitu poskytovanou vyššími úrovněmi.

Problematickým však může být fakt, že lze jen těžko vyvíjenou politiku označit za zcela hotovou a její části za neměnné. Dokud neproběhne celý vývoj dokonce, je třeba mít na paměti, že stále mohou přijít požadavky vedoucí k velkým změnám.

Co se týče cílů politiky pro *uživatelský prostor* definovaných v sekci 4.3 na straně 33, umožňuje tento přístup jejich naplňování již od samého začátku vývoje. Protože tento přístup začíná uzavřením koncových aplikací, lze již poměrně záhy *zajistit integritu uživatelských dat*, se kterými tyto aplikace pracují. Obdobně bude politika již od začátku chránit samotné koncové aplikace. Naopak důsledné použití *RBAC* může být s tímto přístupem komplikovanější, protože okamžitě vyvolá požadavky na úpravy funkčnosti na všech vyšších úrovních. Ochránit citlivá data a aplikace, které s nimi pracují by, však mělo opět jít zavčas.

## 6.3 Argumenty pro zvolený přístup postupující „zdola nahoru“

Metodu tvorby politiky „zdola nahoru“ jsem vybral po pečlivém uvážení z několika důvodů. Rozhodujícím faktem bylo, že se tento přístup již od počátku své aplikace zaměřuje na naplnění cílů vytvářené politiky stanovených v sekci 4.3.5 na straně 35. Toto považuji za důležitou výhodu pro svou práci, neboť ta je omezena časovými, rozsahovými i kapacitními limity. Mám-li pak předkládat výsledky, musím metodu své práce volit tak, aby bylo jejich dosažení reálné, což metoda vývoje politiky přístupem „zdola nahoru“ umožňuje. Naopak kdybych vytvářel politiku přístupem „shora dolů“, nemohl bych zodpovědně tvrdit, že s danými omezeními dosáhnu viditelného a použitelného výsledku.

Druhou podstatnou výhodu spatřuji v tom, že přístup budující politiku „zdola nahoru“ usnadňuje testování funkčnosti vytvořených *modulů* politiky. Takto se mohu při vývoji politiky zaměřit v danou chvíli na jednu aplikaci, kterou mohu otestovat v jejím reálném nasazení, tedy ve spuštěném *pracovním prostředí* se všemi běžícími službami a dalšími aplikacemi. Při uzavírání *uživatelského prostoru* směrem „shora dolů“ bude úvodní testování *modulů* politiky komplikované.

Mé rozhodování také ovlivnil fakt, že bych uvítal, pokud by vytvořená politika někde byla reálně využita, či někým dále rozšířena. Aby toto bylo možné,

musí být výsledná politika opravdu nasaditelná, tedy musí svému uživateli poskytnout hmatatelný bezpečnostní přínos. Ten se odvíjí od míry splnění cílů a je to právě přístup tvořící politiku „zdola nahoru“, který mi dává reálnou šanci tyto cíle s danými rozsahovými a časovými omezeními splnit.

Konečně jsem také při výběru přístupu k vývoji politiky vycházel z filosofického hlediska. Filosofická metodologie zná metody dedukce a indukce a každou z nich považuje za vhodnou pro jinou situaci. Adekvátní nasazení deduktivní metody nezbytně vyžaduje vycházet z obecného. V případě tvorby *bezpečnostní politiky* pro již existující aplikace však žádné obecno, na které bych mohl vztáhnout nějaký deduktivní přístup, není. Logicky se tedy nabízí přístup induktivní, který se snaží z jednotlivin vyvozovat obecnosti. O vhodném užití induktivní metody hovoří již Bacon, jehož metodu postupných indukcí si beru za vzor při návrhu aplikace přístupu tvořícího politiky „zdola nahoru“ a na níž zakládám oprávněnost jeho použití. Jde o metodu přírodovědeckou, která se uplatňuje v popisu existujících dějů ve světě, a takto pak i analogicky vhodnou pro tvorbu *bezpečnostní politiky*, která má popsat bezpečné a nebezpečné chování reálně existujících aplikací. To, co je bezpečné a nebezpečné, již samozřejmě musí být dedukováno z cílů *bezpečnostní politiky*.

Část III

# Bezpečnostní politiky vybraných KDE aplikací a služeb

Tato část práce pojednává o vytvořené *bezpečnostní politice* pro vybrané aplikace a služby *prostředí KDE*. Podrobnější výpis zdrojových kódů politiky [38] je uveden v přílohách práce od strany 80. Důležité části zdrojových kódů vytvořené politiky komentuji v této části práce, která je v souladu s použitým přístupem vývoje politiky směrem „zdola nahoru“ strukturována následovně: V první sekci se zaměřuji na popis politiky pro vybrané *KDE aplikace*, tedy na nejnižší *vrstvu* schématu na obrázku 2 na straně 43. Ve druhé sekci pak popisuji *moduly* pro *služby prostředí KDE*, které jsou potřeba pro provoz vybraných aplikací. Konečně ve třetí sekci se věnuji vytvořené politice nejvyšších dvou *vrstev* ve zmiňovaném schématu. Jde tedy o popis *modulu* zastřešujícího celé *prostředí KDE* a navrhované úpravy *Referenční politiky*.

Vývoj politiky probíhal v distribuci *Fedora* ve verzi 13 s průběžnou aktualizací nainstalovaného softwaru. *Prostředí KDE* tak dosáhlo čísel řad 4.4 a 4.5. Pro tvorbu zdrojových kódů politiky jsem použil textový editor *VIM*. Překlad zdrojových kódů do binární podoby probíhal příkazem:

```
make -f /usr/share/selinux/devel/Makefile
```

*AVC zprávy* o zamítnutí přístupu jsem sledoval pomocí dodávaného nástroje *SELinux Troubleshooter*, případně pomocí nástrojů `cat` a `grep` volaných na soubory s logy z příkazové řádky. Na sledování verzí vytvořené politiky jsem použil nástroj `git` a za základ jeho úložiště jsem vzal zdrojové kódy *Referenční politiky* uložené ve veřejném repositáři [39].

[38] Úplné zdrojové kódy všech vytvořených *modulů* přesahují svým rozsahem 10 000 řádek kódu. Z toho přes 9 000 řádek tvoří soubory s *rozhraními*, které jsou z poloviny dále tvořeny povinnými komentáři pro generování dokumentace politiky. Tato *rozhraní* jsou si velmi podobná jak v rámci jednotlivých *modulů*, tak i mezi nimi. V přílohách je tedy uveden úplný výpis pouze u *souborů s kontexty* a *pravidly*, u *souborů s rozhraními* jsou plně vypsaná jen *rozhraní přístupu rolí* a hlavičky. U ostatních *rozhraní* uvádím pouze názvy. Úplné zdrojové kódy jsou součástí elektronických příloh k práci.

[39] Repositář zdrojových kódů projektu *Referenční politiky*: *http://oss.tresys.com/git/refpolicy.git*.

# 7 Politiky pro vybrané KDE aplikace

V této sekci popíši vytvořené *moduly* politiky pro vybrané *KDE aplikace*. Konkrétně jde o *moduly* pro poštovního klienta *KMail*, adresář *KAddressBook* a webový prohlížeč *Konqueror*. V případě *Konqueroru* se zaměřím pouze na změny provedené od verze, kterou jsem vytvořil spolu s mou bakalářskou prací. Tyto *moduly* používají služby poskytované *moduly* vyšších *vrstev* vytvářené politiky, jak je vidět na obrázku 2 na straně 43. Jejich implementace byla vyvolána potřebou jednotlivých *modulů* koncových aplikací v souladu s využitým přístupem budujícím politiku „zdola nahoru“. Protože jsou zde volána jejich *rozhraní*, budou tato stručně popsána, ale úplného popisu se dočkají až v sekcích 8 a 9 na stranách 60 a 64. Případné obtíže nastalé při tvorbě politiky budou popsány na místech, kde k nim došlo, avšak konkrétní řešení bude důkladně popisováno tam, kam ono samo spadá, a zde bude jen zmíněno.

## 7.1 Modul pro poštovního klienta KMail

V této sekci přiblížím *modul* politiky pro poštovního klienta *KMail*. Jde o *modul* pro koncovou aplikaci, takže je v souladu s použitou metodikou vývoje politiky připraven k uživatelskému využití shodujícímu se se stanovenými případy užití. Nejprve se budu věnovat definovaným *typům*, poté popíši vybraná *rozhraní*, na to naváži popisem důležitých částí politiky *modulu* ve formě *SELinux pravidel* a volání *rozhraní* a charakteristiku *modulu* ukončím výčtem případů užití, které tento *modul* podporuje.

### 7.1.1 Definované typy

Pro binární soubory *KMailu* jsem definoval *typ* `kmail_exec_t`. Tyto soubory se nacházejí v `/usr/bin/` a příslušné části politiky umožňují jejich prostřednictvím přejít do *domény* `kmail_t`.

Konfigurační soubory *KMailu* jsem označil *typem* `kmail_home_conf_t`. Jde o soubory umístěné v adresáři `~/.kde/share/config/`, které mohou obsahovat některé údaje o poštovních účtech uživatele. Může se však stát, že k nim některá z jiných *KDE aplikací* bude chtít přistupovat. V takovém případě by povolení mělo nastat, jen pokud je skutečně opodstatněné.

Vlastní data *KMailu*, tedy hlavně poštovní zprávy uživatele, jsou označena *typem kmail_home_data_t* a nacházejí se v adresáři `~/.kde/share/apps/kmail/` a `~/.kde/share/apps/emailidentities/`. Tato data jsou vysoce citlivá a tudíž je vhodné nepovolovat jiným aplikacím přístup k těmto *typům*. Výjimkou mohou být služby, na kterých *KMail* závisí, jako je *Akonadi* či aplikace, které ze své podstaty potřebují přistupovat k veškerému obsahu v *uživatelském prostoru* jako je správce souborů.

Data *KMailu* dočasně pronikají do adresářů `~/.local/share/local-mail/` a `~/.local/share/.local-mail/`. Tam umístěná data jsem pak označil *typem* `kmail_data_home_t`. Opět jde o uživatelskou poštu, v tomto případě o odesílané zprávy.

Konečně také v politice existuje *typ* pro dočasné soubory *KMailu* vyžadovaný *rozhraním* pro přístup k *X serveru*. Tento *typ* jinak není využíván, protože *KMail* spolu s dalšími uzavřenými *KDE aplikacemi* používá pro své dočasné soubory sdílený *typ* `kde_tmp_t` [40].

### 7.1.2 Vybraná rozhraní

Interakci s *modulem* politiky `kmail` zajišťuje několik definovaných *rozhraní*. *Rozhraní* `kmail_role(role, typ)` zajišťuje přidělení *typu* `kmail_t` *roli*. Zároveň také povoluje *typu* této *role* různé způsoby komunikace s *modulem* `kmail`. Jde například o zasílání různých druhů signálů, komunikaci přes „unix stream socket“ či sběrnici *D-Bus*. Z tohoto *rozhraní* se také volá *rozhraní* `kmail_domtrans(typ)`, které umožní zadanému *typu role* přejít do *domény* `kmail_t`. Toho všeho se dosahuje následujícím kódem:

```
interface(‘kmail_role’,‘
        gen_require(‘
                type kmail_t;
        ’)

        role $1 types kmail_t;

        # Allow the user domain to signal/ps.
        ps_process_pattern($2, kmail_t)
        allow $2 kmail_t:process signal_perms;
        allow kmail_t $2:process signal_perms;

        kmail_domtrans($2)

        kmail_stream_connect($2)

        # Allow user domain to dbus-chat with kmail
        optional_policy(‘
                kmail_dbus_chat($2)
        ’)
’)
```

Další *rozhraní* se starají o obousměrnou komunikaci mezi *doménou* `kmail_t` a zadanou *doménou*. Jde o výše volaná *rozhraní* `kmail_stream_connect(doména)`

[40] Podrobněji o *typu* `kde_tmp_t` a důvodech pro jeho současné použití v sekci 9.1 na straně 65.

a `kmail_dbus_chat(doména)`. Tato *rozhraní* jsou obdobná pro ostatní *moduly* politiky, proto uvedu jejich kód pouze na tomto místě:

```
interface(‘kmail_stream_connect’,‘
        gen_require(‘
                type kmail_t;
        ’)

        allow kmail_t $1:unix_stream_socket connectto;
        allow $1 kmail_t:unix_stream_socket connectto;
’)

interface(‘kmail_dbus_chat’,‘
        gen_require(‘
                type kmail_t;
                class dbus send_msg;
        ’)
        allow $1 kmail_t:dbus send_msg;
        allow kmail_t $1:dbus send_msg;
’)
```

Zbývající *rozhraní modulu* `kmail` umožňují různou míru přístupu k jeho *typům*. Pojmenování vychází ze jmenných konvencí *Referenční politiky*. Namátkou jde o *rozhraní* jako:

- `kmail_read_data_home_files(doména)`,
- `kmail_search_home_data(doména)`
- nebo `kmail_manage_home_data_symlinks(doména)`.

### 7.1.3 Důležitá pravidla

Na počátku *souboru s pravidly modulu* `kmail` jsou definovány *přepínače*, které umožňují zapínání či vypínání určených bloků politiky za běhu. Konkrétně jde o části politiky umožňující *doméně* `kmail_t` čtení či zápis *uživatelských dat* uložených v *uživatelských XDG adresářích*. Výchozí nastavení toto ve většině případů zakazuje, výjimkou je složka se staženými soubory, kam má *KMail* povolený plný přístup, a složka s veřejnými soubory, které *KMail* může číst.

Poté následují deklarace *typů* a pravidla a volání *rozhraní* povolující základní interakce *KMailu* se sebou samým. Prvním zajímavějším pravidlem je až:

```
corecmd_exec_bin(kmail_t)
```

Toto pravidlo povoluje *KMailu* spouštět obecné spustitelné aplikace. Je to potřeba, protože ne všechny nástroje, které *KMail* v některých situacích spouští, jsou v současné době uzavřeny. Konkrétně jde o nástroj pro hlášení pádů programu `drkonqi` či o nástroj na podepisování zpráv `gpg`.

Dalšími poněkud obtížně odhalitelnými přístupy, které *KMail* ke své činnosti potřebuje, je připojení ke sběrnici *D-Bus*, a to jak klientské, tak i systémové. Mimo to se *KMail* dožaduje například systémového času, informací o procesoru, stavu podpory *rozšířených atributů* na kořenovém systému souborů či čtení některých obecných konfiguračních souborů v adresáři */etc/*.

Spolupráci *KMailu* s *modulem uživatelského prostoru* `userdomain` řeší volání několika *rozhraní*:

```
userdom_use_user_terminals(kmail_t) #run from terminal
userdom_signull_unpriv_users(kmail_t) #signull to userdomain
userdom_list_config_home_dirs(kmail_t)
#read, write .config/enchant spellchecking
userdom_read_config_home_files(kmail_t)
userdom_write_config_home_files(kmail_t)
userdom_search_data_home_dirs(kmail_t)
#read mime/genericicons
userdom_read_data_home_files(kmail_t)
```

Z původních *rozhraní* zmíněného *modulu* jde o povolení spouštění v uživatelském terminálu, které může být užitečné pro odhalení chyb, a o zasílání signálu `signull`. Ostatní volaná *rozhraní* byla do *modulu* přidána v rámci této práce, budou tedy důkladněji popsána v sekcích 9.2 a 9.3 na stranách 67 a 68, jejich význam je však celkem jasný z použitých identifikátorů a komentářů ve zdrojovém kódu.

Přístup k *X serveru* zajišťuje *šablona*:

```
xserver_user_x_domain_template(kmail, kmail_t, kmail_tmp_t)
```

Následují vlastní definice již zmíněných *přepínatelných politik*. Ty také používají přidaná *rozhraní modulu* `userdomain`. Poté se zapisují *nepovinné bloky* politiky. V souladu s konvencemi *Referenční politiky* jsem zde uvedl volání *rozhraní* ostatních *modulů* politiky pro *prostředí KDE*, ač si lze ve většině případů představit jejich nepovinnost jen stěží. Zejména v případě integrace s *Akonadi* či *modulem* `kde` je nemyslitelné, aby tyto *moduly* nebyly načteny současně s politikou pro *KMail*. *Referenční politika* to však pro *moduly vrstvy* `apps` vyžaduje a zařazení zmíněných *modulů* do jiných *vrstev* se zase nezdá vhodné, protože nejde o *systémové služby* ani *systém* samotný.

Způsoby integrace *KMailu* s *Akonadi* se též ukázaly jako nesnadno odhalitelné. Zejména bylo obtížné zjistit, které konkrétní další přístupy *KMail* k *Akonadi* potřebuje, protože o sobě nedaly vědět prostřednictvím *AVC zpráv* o zamítnutí přístupu.

Musel jsem tak zkoumat výpisy spuštěného *KMailu* a *Akonadi* na konzoli a z tam uváděných hlášek, z nichž většina byla nerelevantních, odhadovat, co ještě je třeba aplikaci povolit. Výsledkem je poměrně široká množina volaných *rozhraní modulu* `akonadi`, která budou blíže vysvětlena v sekci 8.2 na straně 62:

```
# KMail needs akonadi
optional_policy(‘
        akonadi_list_config_home_dirs(kmail_t)
        # akonadi socket configuration
        akonadi_read_config_home_files(kmail_t)

        # akonadi error logs, if akonadi fails
        akonadi_read_data_home_files(kmail_t)

        akonadi_list_data_home_dirs(kmail_t)
        akonadi_write_data_home_sockets(kmail_t)

        # akonadi-firstrun, kresources
        akonadi_manage_home_conf_files(kmail_t)
        akonadi_rw_home_conf_dirs(kmail_t) #kresources

        akonadi_dbus_chat(kmail_t)
        akonadi_signull(kmail_t)
        akonadi_stream_connect(kmail_t)

        akonadi_domtrans(kmail_t)
’)
```

*KMail* jednak potřebuje přecházet do *domény* `akonadi` v případě, že se pokouší *Akonadi* spouštět sám. Dále s *Akonadi* potřebuje komunikovat prostřednictvím sběrnice *D-Bus* a „Unix stream socketu“. Celkem očekávaně pak potřebuje číst konfiguraci *Akonadi* v adresáři `~/.config`, poněkud překvapivě však vyžaduje úplný přístup ke konfiguračním souborům přidružených nástrojů *Akonadi* a adresáři `~/.kde/share/config`.

Integrace s klíčenkou *KWallet* byla oproti tomu snazší. Povolit stačilo komunikaci přes *D-Bus* a čtení i teď již nepřekvapující zápis do jejích konfiguračních souborů. Integrace s adresářem *KAddressBook* pak spočívá v povolení *doménového přechodu* do *domény* `kaddressbook_t`.

Od zastřešujícího *modulu* pro *prostředí KDE* si *KMail* také vyžádal poměrně rozsáhlou funkcionalitu. Část z ní byla zapříčiněna zvoleným přístupem k řešení ochrany dočasných souborů, což bude rozebráno, jak již bylo uvedeno, v sekci věnované tomuto zastřešujícímu *modulu*. Další povolené přístupy se buď týkají nějaké formy integrace s dosud neuzavřenými aplikacemi či službami jako nástrojem na hlášení chyb, programem na správu certifikátů či sémantickým desktopem

*Nepomuk*, nebo ošetřují vytváření konfiguračních souborů a adresářů s daty *KMail* v situaci, kdy je spuštěn poprvé:

```
# Temp acces for kmail
# Access to kde_home_t, should be reduced in future
# Transition so that kmail_home_files in kde_home_t dir
# wouldn't switch to parent directory type
optional_policy(‘
        kde_manage_tmp_files(kmail_t)
        kde_manage_tmp_sockets(kmail_t)
        kde_manage_tmp_symlinks(kmail_t)
        kde_manage_tmp_dirs(kmail_t)

        kde_read_config_home_files(kmail_t) #trolltech.conf
        kde_write_config_home_files(kmail_t)

        kde_read_home_symlinks(kmail_t) #tmp
        kde_rw_home_dirs(kmail_t)

        kde_read_home_conf_files(kmail_t) # kdeglobals
        # kdebugrc, kleopatra
        kde_write_home_conf_files(kmail_t)
        kde_home_conf_filetrans(kmail_t, \
             kmail_home_conf_t, { file })

        kde_read_home_conf_kio_files(kmail_t) #kio http
        kde_write_home_conf_kio_files(kmail_t)

        kde_read_home_data_files(kmail_t)
        # kdebugrc, kleopatra
        kde_write_home_data_files(kmail_t)
        kde_write_home_data_sockets(kmail_t) # nepomuk
        kde_home_data_filetrans(kmail_t, \
             kmail_home_data_t, { dir })
’)
```

### 7.1.4 Případy užití

Politiku pro poštovního klienta *KMail* jsem vytvářel a testoval pro následující případy užití:

- spuštění aplikace při běžícím *Akonadi serveru* z konzole, menu či pomocí aplikace *KRunner*,

- vytvoření uživatelských identit a přidání poštovních účtů,
- prohlížení pošty přes protokol *IMAP*,
- stažení pošty s využitím možností „odpojeného *IMAPu*“,
- stažení pošty přes protokol *POP3*,
- odeslání pošty přes protokol *SMTP*,
- uložení přílohy do vyčleněného adresáře se staženými soubory,
- otevření přílohy v pomocné aplikaci bez *doménového přechodu*,
- přidání přílohy z vyčleněného adresáře s veřejnými soubory,
- volitelné přidání přílohy z ostatních známých uživatelských adresářů,
- uložení e-mailové adresy do adresáře *KAddressBook*,
- odeslání pošty na adresu z adresáře *KAddressBook*,
- uchování přístupových hesel k poštovním účtům v klíčence *KWallet*,
- ukončení aplikace.

## 7.2 Modul pro adresář KAddressBook

V této sekci se budu věnovat *modulu* pro adresář *KAddressBook*. Vzhledem k tomu, že jde o koncovou aplikaci a vzhledem k použité metodice vývoje, je tento *modul* připraven pro uživatelské využití v souladu se stanovenými případy užití. Při popisu *modulu* nejprve popíši definované *typy*, poté přiblížím některá vybraná *rozhraní*, okomentuji důležitá pravidla a nakonec uvedu případy užití, pro které byl tento *modul* vyvíjen. Vzhledem k tomu, že *KDE aplikace* používají stejnou infrastrukturu, jsou si svými požadavky velmi podobné, takže se i *bezpečnostní politiky* pro ně podobají. Proto si mohu dovolit být v této sekci stručnější.

### 7.2.1 Definované typy

Binární soubory *KAddressBooku* jsem označil *typem* `kaddressbook_exec_t`. Příslušná pravidla umožňují přechod do *domény* `kaddressbook_t`. Pro konfigurační soubory *KAddressBooku* jsem použil *typ* `kaddressbook_home_conf_t`. Oproti *KMailu* tyto soubory neobsahují uživatelsky citlivá data. Soubory s uživatelskými daty, tedy knihy adres, jsou uložené v adresáři `~/.kde/share/apps/kabc/`. Těm jsem přiřadil *typ* `kaddressbook_home_data_t`.

### 7.2.2 Vybraná rozhraní

*Rozhraní modulu* `kaddressbook` zahrnují obvyklou sadu přístupového *rozhraní* pro *roli*, přechodového *rozhraní* do *domény* `kaddressbook_t`, *rozhraní* pro obousměrnou komunikaci přes sběrnici *D-Bus* a „unix stream socket".

*Rozhraní* pro *přístup role* `kaddressbook_role(role, typ)` umožňuje *typu role* komunikovat s *doménou* `kaddressbook_t` prostřednictvím signálů, *D-Busu* i „unix stream socket". Dále pak volá rozhraní *doménového přechodu*.

Zbývající *rozhraní* pak různým způsobem zpřístupňují objekty s *typy KAddressBooku* v uživatelském adresáři. Příkladem budiž *rozhraní*, které umožňuje dané *doméně* číst konfigurační soubory *KAddressBooku*:

```
kaddressbook_read_home_conf_files(doména)
```

### 7.2.3 Důležitá pravidla

Také *modul* `kaddressbook` využívá *přepínatelných politik* pro povolení přístupu k uživatelskému obsahu. Výchozí nastavení povoluje pouze čtení obsahu adresáře s veřejnými dokumenty.

*KAddressBook* též potřebuje přístup k systémovému i uživatelskému *D-Busu* a jeho integrace s *Akonadi* si jako vedlejší produkt vyžádala využití přidaných *rozhraní modulu* `userdomain`, které povolují prohledávání *základních XDG adresářů* `~/.config` a `~/.local`.

Další přístupy vyžádané integrací s *Akonadi* jsou velmi podobné situaci s *modulem* `kmail`. *KAddressBook* potřebuje číst soubory patřící *Akonadi* v adresářích `~/.config` a `~/.local` a požaduje také plný přístup ke konfiguračním souborům *Akonadi* umístěným v adresáři `~/.kde/share/config`. Komunikace mezi *KAddressBookem* a *Akonadi* pak probíhá přes sběrnici *D-Bus* a „unix stream socket".

Integrace s poštovním klientem *KMail* spočívá v nutnosti povolit *doméně KAddressBooku* čtení a zápis do konfiguračních souborů *KMailu*.

Integrace se zastřešujícím *modulem* `kde` pak opět zahrnuje přístupy nutné vzhledem ke zvolenému řešení sdíleného přístupu do dočasného adresáře a ošetření *objektových přechodů* při prvním spuštění programu *KAddressBook*. Další přístupy jsou nutné vzhledem k úzké integraci s dosud neuzavřenou aplikací kalendáře *KOrganizer*. Ta si vyžádala udělení plného přístupu na soubory s daty dosud neuzavřených *KDE aplikací*, což vyvolává poměrně silnou potřebu uzavřít zbývající *KDE PIM aplikace*.

### 7.2.4 Případy užití

Politiku pro adresář *KAddressBook* jsem vytvářel a testoval pro následující případy užití:

- spuštění aplikace s běžícím *Akonadi serverem* z konzole, menu či pomocí aplikace *KRunner*,
- přidání nové knihy adres ve formátu *VCard*,
- přidání, odstranění a úprava kontaktu na osobu,
- načtení obrázku osoby ze složky s veřejnými dokumenty,
- volitelné načtení obrázku osoby z ostatních *uživatelských XDG složek*,
- zaslání zprávy na e-mailovou adresu prostřednictvím *KMailu*,
- ukončení aplikace.

## 7.3 Modul pro webový prohlížeč Konqueror

V této sekci popíši změny, které se udály v *modulu* politiky pro webový prohlížeč *Konqueror*. Základ této politiky jsem vytvořil v rámci své bakalářské práce, ale ač měla již tehdy většinu své současné funkcionality, ukázalo se v diskuzích s vývojáři *Referenční politiky*, že některé věci je lepší udělat jinak. Zejména šlo o vyšší míru využití *rozhraní Referenční politiky* a o některé jiné, dříve neviditelné problémy. Další změny jsem pak v politice *Konqueroru* musel provést v souvislosti s vývojem politiky pro další *KDE aplikace*, jak již ostatně předpokládá zvolená metodika.

*Konqueror* je koncovou aplikací, tedy je jeho politika vhodná pro uživatelské použití v souladu s uvažovanými případy užití. Toto je dále posíleno faktem, že se jedná o nejstarší ze zde prezentovaných *modulů*, který tak byl podroben nejdlouhodobějšímu testování. Ve zbytku této sekce tedy popíši změny v definovaných *typech*, okomentuji vybraná *rozhraní* a důležitá pravidla a nakonec také uvedu případy užití.

### 7.3.1 Definované typy

Podstatnou změnou v definovaných *typech Konqueroru* je jejich rozdělení na samostatný *typ* pro konfigurační soubory `konqueror_home_conf_t` a *typ* pro vlastní data `konqueror_home_conf_t`. Tento krok souvisí se stanoveným cílem pro *bezpečnostní politiku* vyvíjenou v rámci této práce, totiž *důsledným využitím TE k rozdělení uživatelských dat na co nejvíce typů*. Díky tomu lze bezpečněji implementovat interakci mezi jednotlivými *KDE aplikacemi*, které, jak jsme viděli v předchozích sekcích, mají často ve zvyku zapisovat do cizích konfiguračních souborů.

Další viditelnou změnou je upuštění od používá *typu* pro dočasné soubory v `/tmp` `konqueror_tmp_t`. Jak již bylo uvedeno dříve, důvody důkladně proberu v sekci 9.1 na straně 65.

### 7.3.2 Vybraná rozhraní

Z důvodu sladění s konvencemi *Referenční politiky* došlo k několika změnám v definovaných *rozhraních*. Zejména bylo zrušeno *rozhraní* `konqueror_run(doména, role, terminál)`, které bylo nahrazeno v *Referenční politice* vyžadovaným *rozhraním* pro *přístup k rolím* `konqueror_role(role, typ)`.

Dalších změn pak doznalo především pojmenování *přístupových rozhraní* tak, aby odpovídalo jmenným konvencím. Vzhledem k nasazení důkladnějšího *TE* také přibyla některá *rozhraní* pro manipulaci s rozdělenými *typy* pro konfigurační soubory a data aplikace.

Poslední změnou bylo přidání *rozhraní* umožňujícího komunikaci přes „unix stream socket“ `konqueror_stream_connect(doména)`.

### 7.3.3 Důležitá pravidla

Značných změn doznal *soubor s pravidly modulu* `konqueror`. Především byly odstraněny bloky `require` a `gen_require(‘’)`, které nejsou v *Referenční politice* v tomto typu souborů povoleny. Logika zajišťující přidání *typu* `konqueror_t` mezi povolené *typy role* tak byla přesunuta do *rozhraní* `konqueror_role(role, typ)`, které se volá z příslušných *modulů* jednotlivých *rolí*.

Pro komunikaci s ostatními *moduly* politiky pro *KDE aplikace* jsem pak v souladu s konvencemi *Referenční politiky* i zde použil příslušná *rozhraní* volaná z bloků `optional_policy(‘’)`.

Nová funkcionalita implementovaná do *modulu* `konqueror` zahrnuje integraci s klíčenkou *KWallet*. I webový prohlížeč *Konqueror* si žádá práva čtení a zápisu do jejích konfiguračních souborů. Vlastní komunikace pak probíhá přes sběrnici *D-Bus*.

Dále jsem pro *modul* `konqueror` vytvořil *přepínatelné politiky*, které umožňují, či zakazují přístup do známých *uživatelských XDG adresářů*. Výchozí nastavení povoluje *Konqueroru* plný přístup do adresáře se staženými soubory a umožňuje mu číst veřejné dokumenty.

Integraci s *modulem* zastřešujícím *prostředí KDE* jsem upravil tak, aby reflektovala důkladnější *TE*, což podstatně snížilo rozsah práv, která bylo nutné *Konqueroru* přidělit oproti předchozímu stavu. Dále si *Konqueror* na *modulu* `kde` vyžádal vytvoření zvláštních *rozhraní* pro manipulaci s *typy* konfiguračních souborů technologie *KIO slaves* a dat zásuvných modulů *nsplugins*.

Konečně *Konqueror* využívá i některé funkcionality poskytované přidanými *rozhraními modulu* `userdomain`. Zejména jde o integraci se systémy automatické kontroly pravopisu.

### 7.3.4 Případy užití

*Modul* pro webový prohlížeč *Konqueror* jsem vyvíjel a testoval pro následující případy užití:

- spuštění aplikace z konzole, menu či aplikací *KRunner*,
- prohlížení webu pomocí protokolu *HTTP*,
- prohlížení webu pomocí protokolu *HTTPS*,
- stažení souboru z webu a jeho uložení do adresáře se staženými soubory,
- stažení souboru z webu a jeho otevření v pomocné aplikaci bez *doménového přechodu*,
- nahrání souboru na web z adresáře se staženými soubory nebo z adresáře s veřejnými dokumenty,
- volitelná nahrání souboru na web z ostatních *uživatelských XDG adresářů*,
- uchování přístupových údajů k webům v klíčence *KWallet*,
- ukončení programu.

# 8 Politiky pro vybrané KDE služby

V této sekci popíši implementaci *modulů* uzavírajících vybrané služby tak, jak to požadovaly *moduly* pro koncové aplikace. Jde o *modul* pro klíčenku *KWallet* a *PIM* systém *Akonadi*. Ve schématu na obrázku 2 na straně 43 jde o druhou spodní *vrstvu*. Tyto *moduly* tedy musí poskytnout požadované služby koncovým aplikacím. V souladu s použitou metodikou jde jen o služby jimi požadované – tyto *moduly* si tedy nekladou za cíl zpřístupnit veškerou funkcionalitu uzavíraných služeb. Dále se *moduly* uzavírající *KDE služby* integrují se zastřešujícím *modulem prostředí KDE* a s *moduly uživatelského prostoru*, na jejichž funkcionalitu také kladou jisté požadavky.

## 8.1 Modul pro klíčenku KWallet

V této sekci popíši *modul* pro klíčenku *KWallet*. Jde o službu uchovávající uživatelská hesla, což jsou velmi citlivá data. Vzhledem k použité metodice tvorby politiky je tento *modul* vhodný pro zkušební nasazení odpovídající vymezeným případům užití v kombinaci s *moduly* aplikací, pro které byl vyvíjen a testován. Nyní se zaměřím na popis definovaných *typů*, vybraných *rozhraní* a důležitých pravidel. Nakonec doplním uvažované případy užití.

### 8.1.1 Definované typy

Spustitelné soubory služby *KWallet* i jejího konfiguračního nástroje jsem označil *typem* `kwallet_exec_t`. Příslušná pravidla pak umožňují přechod do *domény* `kwallet_t`.

Konfigurační soubory služby i konfiguračního nástroje pak mají přiřazen *typ* `kwallet_home_conf_t`. Tyto soubory neobsahují vyloženě citlivá data, jak již bylo uvedeno v předchozích sekcích, a aplikace integrující se s klíčenkou k nim vyžadují právo čtení i zápisu. Přesto bych doporučoval omezit množství *modulů* s povoleným přístupem na minimum.

Hesla, která *KWallet* uchovává, jsou uložena v zašifrovaných souborech, jimž jsem přidělil *typ* `kwallet_home_data_t`. K nim by měla přistupovat pouze samotná služba *KWallet*.

### 8.1.2 Vybraná rozhraní

Pro využití politiky klíčenky *KWallet* je důležité *rozhraní* `kwallet_role(role, typ)`, které povoluje *typu* příslušné *roli* komunikaci pomocí signálů, *D-Busu* i „unix stream socketu“. Dále také volá *rozhraní* způsobující *doménový přechod*.

*Moduly*, které využívají služeb úschovny *KWallet*, k ní potřebují přistupovat přes sběrnici *D-Bus*. O to se stará *rozhraní* `kwallet_dbus_chat(doména)`, které povoluje *doméně* dané aplikace obousměrnou komunikaci přes *D-bus* s *KWallet*.

Ostatní *rozhraní* povolují různé formy přístupu k různým *třídám objektů* vlastněným *modulem* `kwallet`. Pro *moduly* využívající služby *KWallet* jsou důležitá *rozhraní* udělující právo čtení a zápisu do konfiguračních souborů *KWallet* tedy:

```
kwallet_read_home_conf_files(doména)
kwallet_write_home_conf_files(doména)
```

### 8.1.3 Důležitá pravidla

Konfigurační *GUI* nástroj klíčenky spouští nástroj umožňující hlásit chyby v aplikaci. Tento však zatím není uzavřený, proto jsem povolil spouštění obecných programů. Démon zajišťující samotnou službu pak spouští `shell`, což jsem mu umožnil voláním *rozhraní*:

```
corecmd_exec_shell(kwallet_t)
```

Nejspíše kvůli svým kryptografickým funkcím požaduje démon povolit čtení ze zařízení `/dev/random`. K tomu slouží *rozhraní*:

```
dev_read_rand(kwallet_t)
```

Mimo to potřebuje *KWallet* také komunikovat se systémovou i uživatelskou sběrnicí *D-Bus*. Integrace se zastřešujícím *modulem prostředí KDE* spočívá hlavně v přístupu ke sdílenému *typu* pro dočasné soubory, zajištění správného *objektového přechodu* při prvním spuštění a čtení některých základních konfiguračních souborů.

### 8.1.4 Případy užití

Politiku *modulu* pro klíčenku *KWallet* jsem vyvíjel a testoval pro následující případy užití:

- vyvolání startu služby přes *D-Bus* aplikacemi *Konqueror* a *KMail*,
- ruční spuštění démona služby z konzole,
- spuštění konfiguračního nástroje z konzole, menu nebo aplikací *KRunner*,
- přidání, odstranění a upravení klíčenky s využitím konfiguračního nástroje *KWalletManager*,
- uložení hesla do klíčenky z aplikace *KMail* nebo *Konqueror*,
- vyvolání hesla z klíčenky aplikacemi *KMail* a *Konqueror*,
- ukončení konfiguračního nástroje *KWalletManager*.

## 8.2 Modul pro PIM systém Akonadi

Na tomto místě popíši důležité momenty politiky *modulu* pro *PIM* systém *Akonadi*. Tento *modul* vznikal na základě požadavků, které se vyskytly při tvorbě *modulů* pro koncové aplikace závisející na službě *Akonadi* – tedy *KMailu* a *KAddressBooku*. Jde o modul druhé spodní *vrstvy* schématu na obrázku 2 na straně 43, takže jeho účelem je poskytovat funkcionalitu koncovým aplikacím a dále požadovat určitou funkcionalitu od *modulů* vyšších *vrstev*. Vzhledem k použité metodice tvorby politiky je tento *modul* vhodný pro zkušební použití respektující stanovené případy užití za předpokladu interakce s programy, pro které byl vytvářen. Nyní popíši definované *typy*, vybraná *rozhraní* a důležitá pravidla a nakonec uvedu zamýšlené případy užití, pro které jsem politiku testoval.

### 8.2.1 Definované typy

„Binárky“ související s *Akonadi* jsem označil *typem* `akonadi_exec_t`, příslušná pravidla pak při jejich spuštění umožňují přechod do *domény* `akonadi_t`.

Konfigurační soubory *Akonadi* se vyskytují v adresáři `~/.config`, tam nesou označení `akonadi_config_home_t` a v adresáři `~/.kde/share/config`,

kde nesou označení `akonadi_home_conf_t`. V prvním případě jde především o konfigurační soubory *Akonadi serveru*, v druhém pak o konfigurační soubory pomocných programů.

Citlivější informace *Akonadi* ukládá do adresáře `~/.local`, kde jsem použil *typ* `akonadi_data_home_t`. Jde o metadata o *uživatelských datech*, která *Akonadi* zpřístupňuje.

*Akonadi* jako jediný *modul* nezbytně vyžadoval vlastní *typ* `akonadi_tmp_t` pro dočasné soubory. Nemohl fungovat jen se sdíleným *typem* v rámci celého *prostředí KDE*.

### 8.2.2 Vybraná rozhraní

O spuštění jednotlivých procesů *Akonadi* ve správném *kontextu* pro danou *roli* se stará *rozhraní* `akonadi_role(role, typ)`. To také zajišťují volaná *rozhraní*, která se starají o *doménový přechod* a komunikaci prostřednictvím sběrnice *D-Bus* a „unix stream socketu“.

Další *rozhraní* už pouze umožňují různé druhy přístupu k různým *třídám objektů* jednotlivých *typů* náležejících k *Akonadi*, případně zajišťují *objektový přechod* při prvním spuštění aplikace. Oproti dosud komentovaným *modulům* je jich poněkud více, protože *Akonadi* jako služba vytváří vlastní socket, který slouží jako komunikační kanál.

### 8.2.3 Důležitá pravidla

*Akonadi* ukládá metadata o *uživatelských datech* do *MySQL* databáze. Z hlediska zabezpečení s tím souvisí volání několika *rozhraní*, která jsou někdy poněkud překvapivá. Celý proces uzavírání navíc komplikovalo to, že se ne všechny zamítnuté přístupy projevovaly patřičnými *AVC zprávami*. Navíc se zdá, že si *Akonadi* žádá značně rozsáhlá práva k mnoha *typům*, bez kterých se ani nespustí:

```
corecmd_exec_bin(akonadi_t)
mysql_read_config(akonadi_t)
mysql_can_exec(akonadi_t)
userdom_manage_user_tmp_files(akonadi_t)
```

První zmíněné *rozhraní* povoluje spouštění obecných „binárek“. Jedna z nich slouží k vytvoření *MySQL* databáze. Dále je potřeba číst konfigurační soubory *MySQL*, pak je nutné spustit vlastní databázi bez *doménového přechodu*. Některé dočasné soubory se však vytvářejí s uživatelským *typem*, proto je nutné k němu povolit přístup posledním *rozhraním*.

Současná implementace integrace s *Nepomukem* spočívá v jeho spouštění bez *doménového přechodu*. Nezbytností je také zápis do jeho socketu povolený voláním *rozhraní*:

```
kde_write_home_data_sockets(akonadi_t)
```

Jako ostatní *KDE aplikace a služby* vyžaduje i *Akonadi* přístup k systémové i uživatelské sběrnici *D-Bus*. Pomocné *GUI* programy *Akonadi* si pak žádají přístup k *X serveru*.

Mezi volaná *rozhraní modulu* `kde` pak patří i ta, která zajišťují přístup *Akonadi* k datům dosud neuzavřených *KDE PIM* aplikací. Konkrétně jde o *rozhraní*:

```
kde_read_home_data_files(akonadi_t)
kde_write_home_data_files(akonadi_t)
```

Vzhledem k tomu, že *Akonadi* slouží jako vyrovnávací paměť pro data adresáře *KAddressBook*, je třeba povolit k nim plný přístup:

```
kaddressbook_manage_home_data_files(akonadi_t)
kaddressbook_manage_home_data_dirs(akonadi_t)
```

Jak je vidět jen z požadavků vyvolaných uzavřením jedné *KDE PIM* aplikace, *Akonadi* bude pracovat s mnoha citlivými daty. Při dalším rozvíjení politiky by se politice pro *Akonadi* měla věnovat zvýšená pozornost a snaha tyto přístupy pokud možno minimalizovat, například rozčleněním do více *aplikačních domén* podle konkrétního *zdroje Akonadi*.

### 8.2.4 Případy užití

Politiku pro *PIM* systém *Akonadi* jsem vyvíjel a testoval pro následující případy užití:

- ruční spouštění služby z příkazové řádky nebo *GUI* nástroje,
- automatické spouštění služby při startu *prostředí KDE*,
- využití služby aplikací *KMail* do verze distribuované s *KDE SC 4.5*,
- využití služby aplikací *KAddressBook*,
- ruční a automatické ukončení služby.

## 9 Politika pro KDE a návrhy úprav Referenční politiky

Nyní již zbývá popsat *moduly* nejvyšších dvou *vrstev* schématu na obrázku 2 na straně 43. Jednak jde o vrstvu *pracovních prostředí*, kde jsem upravoval *modul* pro *prostředí KDE* tak, aby vyhovoval nově objeveným požadavkům *modulů* aplikací a

služeb. Dále sem patří změny, které jsem prováděl v *modulech vrstvy uživatelského prostoru*. Tam šlo o *moduly* `userdomain`, `mysql` a *moduly rolí*. Protože jsem politiku vyvíjel na *Fedoře* a zároveň ji ladil i pro *Referenční politiku*, a protože se od sebe tyto politiky v některých aspektech liší, byl jsem nucen vytvořit odlišné sady změn *modulu* `userdomain` pro *politiku Fedory* a *Referenční politiku*. V případě *politiky Fedory* jsem zvolil přístup samostatného *modulu* dodávajícího požadovanou funkčnost a využívajícího již existující *moduly*. V případě *Referenční politiky* jsem prováděl změny přímo v kódu *modulu* `userdomain` formou patche.

## 9.1 Modul pro prostředí KDE

*Modul* `kde` zastřešuje požadavky jednotlivých aplikací a služeb *prostředí KDE*. Jde především o požadavky na *typy* pro objekty, které jsou využívány několika *KDE apliakcemi* nebo *službami* a *typy* dosud neuzavřených aplikací či služeb. Základ tohoto *modulu* jsem vytvářel v rámci své bakalářské práce. Provedené změny souvisejí především s důslednějším uplatněním *TE* dle stanoveného *bezpečnostního cíle*. Dále si je vyžádalo rozhodnutí realizovat v tomto *modulu* sdílený *typ* pro dočasné soubory. Nyní popíši změny učiněné v definovaných *typech*, budu argumentovat pro své rozhodnutí vytvořit sdílený *typ* pro dočasné soubory *KDE aplikací* a nakonec uvedu příklady *rozhraní*, které tento *modul* poskytuje.

### 9.1.1 Definované typy

V rámci důkladnější aplikace *TE* jsem dříve jednotný *typ* `kde_shared_home_t` rozdělil na samostatný *typ* `kde_config_home_t` pro konfigurační soubory v adresáři `~/.config/`, `kde_home_conf_t` pro konfigurační soubory *KDE aplikací* a *služeb* v adresáři `~/.kde/share/config` a `kde_home_data_t` pro data dosud neuzavřených aplikací a služeb. Pro samotný adresář *~/.kde* a jeho nejbližší obsah jsem zachoval původní *typ* v přejmenované podobě `kde_home_t`. Výsledkem je navýšení bezpečnosti, protože ne všechny aplikace potřebují přístup k datům ostatních aplikací, zato však často přistupují k jejich konfiguračním souborům.

Výhodným se jeví i vytvoření *typu* `kde_home_data_nsplugin_t` pro data uložená zásuvnými moduly webových prohlížečů a `kde_home_conf_kio_t` pro konfigurační soubory technologie *KIO*. Data odpovídající těmto *typům* jsou často požadována aplikacemi přistupujícími k síti. Jejich existence tak umožňuje snížit množství a závažnost přístupů požadovaných k obecným *KDE typům*.

### 9.1.2 Uzavření dočasného adresáře

Při testování politiky pro *KMail* v součinnosti s politikou pro *Konqueror* se ukázalo, že tyto aplikace potřebují přistupovat k několika stejným souborům. Bližší inspekce odhalila, že mezi dočasnými soubory *KDE aplikací* existují takové, ke

kterým potřebuje přistupovat každá aplikace, a pak ty, které jsou pro některou aplikaci výlučné.

Pokud by se nejednalo o soubory umístěné v dočasných adresářích, šlo by tento problém řešit vytvořením sdíleného *typu* vlastněného *modulem* `kde` pouze pro dané soubory, což by bylo optimální. Protože je však dočasný adresář při každém startu počítače mazán, toto řešení založené na přeznačkování souborů nástrojem `restorecon` není použitelné.

V tom případě je nutné pátrat po procesu, který dané soubory vytváří. Ten je však bytostně svázán se startem celého prostředí, takže jeho uzavření vzhledem k použité metodice v této fázi vývoje politiky není možné – nejsou uzavřeny služby a aplikace spouštěné při startu prostředí.

Situace si tedy žádá schůdné provizorní řešení. Možnost spočívající v zachování samostatných *typů* pro jednotlivé aplikace a vzájemném povolení manipulace s těmito *typy* jsem zavrhl, protože by brzy vedla k přílišné složitosti. Rozhodl jsem se tedy vytvořit *typ* `kde_tmp_t` a ten u všech uzavřených aplikací použít pro jejich dočasné soubory.

Jednotlivé *KDE aplikace* a *služby* si tak vidí do svých dočasných souborů, což není ideální, zabraňuje to však předčasnému pokusu o uzavření procesu startujícího *pracovní prostředí*. Zároveň toto řešení zajišťuje, že data již uzavřených aplikací nedostanou zcela obecný uživatelský *typ* `user_tmp_t`.

Tento problém hezky ukazuje úskalí zvolené metodiky tvorby politiky. Nyní vytvořené *moduly* používají kód, který bude odstraněn, až se podaří prostředí rozumně uzavřít. Na druhou stranu právě díky použité metodice jsou nyní k dispozici *moduly* pro koncové aplikace, které by se při přístupu „shora dolů“ s danými omezeními práce vytvořit nepodařilo.

### 9.1.3 Poskytovaná rozhraní

*Rozhraní* poskytovaná *modulem* `kde` pouze udělují různé úrovně přístupu k různým *třídám objektů* označeným jednotlivými *typy*. Snažil jsem se vytvářet *rozhraní* s co možná nejodstupňovanějšími právy, aby bylo možné udělit pouze opravdu nutné přístupy. Vlastní obsah těchto *rozhraní* je analogický *rozhraním* ostatních *modulů* a jejich pojmenování odpovídá konvencím *Referenční politiky*.

Zmíním tedy pouze ta *rozhraní*, která udělují přístup ke sdílenému *typu* pro dočasné soubory. Jde o čtyři *rozhraní* umožňující plný přístup k souborům, odkazům, socketům a adresářům s *typem* `kde_tmp_t`. Příkladem může být rozhraní pro přístup k dočasným souborům:

```
interface(‘kde_manage_tmp_files’,‘
        gen_require(‘
                type kde_tmp_t;
```

```
        ')

        manage_files_pattern($1,kde_tmp_t,kde_tmp_t);
        # As these could be inside of user_tmp_t
        # access to it and transition is needed.
        userdom_read_user_tmp_files($1)
        userdom_write_user_tmp_files($1)
        userdom_user_tmp_filetrans($1, kde_tmp_t, file)
')
```

Jak je vidět ze zdrojového kódu, toto *rozhraní* zajišťuje, že *doménou* jakkoliv měněný soubor zůstane v *typu* `kde_tmp_t`. U nově vytvářených souborů to zajistí volané *rozhraní přechodu typů* `userdom_user_tmp_filetrans()`. Problém způsobený neuzavřením startovacího procesu *prostředí KDE* však způsobuje, že dočasné soubory patřící *prostředí KDE* mohly již být vytvořeny s *typem* `user_tmp_t`. Nezbývá tedy nic jiného, než tyto soubory povolit číst a také do nich umožnit zápis, při kterém zafunguje *přechod typů*.

Zbývající pravidla zajišťující přístup k ostatním *třídám objektů* sdíleného *typu* `kde_tmp_t` jsou tomu výše uvedénému již velmi podobná. Odlišnosti plynou jen z rozdílů v možných právech mezi jednotlivými *třídami objektů*.

## 9.2 Modul kompatibility s politikou Fedory

*Modul kompatibility s politikou Fedory* doplňuje *modul* `userdomain`. Při tvorbě politiky pro vybrané *KDE aplikace* se vyskytly požadavky na přístup k funkcionalitě nejvyšší *vrstvy* schématu na obrázku 2. Tyto přístupy se týkaly *základních* a *uživatelských XDG adresářů*, ve kterých mají některé *KDE aplikace* či *služby* svůj obsah, respektive kde se nachází *uživatelská data*, ke kterým může být potřeba přistupovat.

*Bezpečnostní politika Fedory* však oproti *Referenční politice* některým z těchto adresářů již přiřadila jisté *typy*. V případě *základních XDG adresářů* jde o *typy* `config_home_t`, `data_home_t` a `gconf_home_t` deklarované v *modulu* `gnome`. V případě *uživatelských XDG adresářů* pak jde o *typ* `audio_home_t` deklarovaný v *modulu* `userdomain`. S druhým případem lze souhlasit, ale nezdá se mi rozumné, aby *typy* pro *základní XDG adresáře* byly deklarovány v *modulu* `gnome` a měly tomu odpovídající jmenné konvence, když se netýkají pouze *GNOME aplikací*, ale i *KDE aplikací* a jiných aplikací nezačleněných k žádnému *pracovnímu prostředí*.

Toto byly motivace, které mě vedly k vytvoření *modulu kompatibility s politikou Fedory*, jehož hlavním cílem je zpřístupnit *typy* pro *uživatelské* a *základní XDG adresáře* skrze *rozhraní modulu* `userdomain`. Umístění těchto *typů* do tohoto *modulu* je totiž z hlediska návrhu politiky mnohem logičtější. Vytvořený *modul kompatibility* pak překrývá současnou *politiku Fedory*

### 9.2.1 Definované typy

Pojmenování jednotlivých *typů základních XDG adresářů* v *politice Fedory* dobře vyjadřuje, jakých dat se tyto *typy* týkají. Proto jsem v jeho duchu pouze přidal nový *typ* `cache_home_t`.

Pro *uživatelské XDG adresáře* jsem pak zavedl *typy* odpovídající jejich anglickému názvu: `documents_home_t`, `downloads_home_t`, `pictures_home_t`, `public_home_t`, `templates_home_t` a `videos_home_t`. V tomto ohledu nemohu být zcela spokojen s pojmenováním `audio_home_t` pro adresář `~/Music`, které je v *politice Fedory* používáno.

### 9.2.2 Poskytovaná rozhraní

*Rozhraní* poskytovaná *modulem kompatibility s politikou Fedory* zpřístupňují nově definované i původní *typy* a zároveň zakrývají nevhodné umístění původních *typů*. Jejich pojmenování odpovídá konvencím *Refereční politiky* pro jejich logické umístění. Prefixem *modulu* je tedy řetězec `userdom`.

Jednotlivá *rozhraní* jsem se snažil vytvářet tak, aby jimi poskytovaná práva k *třídám objektů* byla dostatečně jemně odstupňována. Příkladem mohou být *rozhraní* pro přístup k souborům *typu* `videos_home_t`:

- `userdom_read_videos_home_files()`
- `userdom_write_videos_home_files()`
- `userdom_create_videos_home_files()`
- `userdom_manage_videos_home_files()`

## 9.3 Úpravy modulu userdomain Referenční politiky

Úpravy *modulu* `userdomain` *Referenční politiky* jsou jen přenesením zdrojového kódu *modulu kompatibility s politikou Fedory* popsaného v předešlé sekci do správného *modulu*, kterým je *modul* `userdomain`. Oproti *politice Fedory* nemá *Referenční politika* žádné zvláštní *typy* pro *uživatelské* ani *základní XDG soubory*, je tedy nutné vytvořit více *typů*. Jejich pojmenování jsem zachoval, protože vhodně vystihuje jejich účel, a tedy koresponduje se jmennými konvencemi *Referenční politiky*.

### 9.3.1 Definované typy

Pro *základní XDG adresáře* jsem tak použil *typy*: `cache_home_t` pro adresář s vyrovnávací pamětí aplikací, `config_home_t` pro adresář s konfiguračními soubory

aplikací a `data_home_t` pro adresář s daty těchto aplikací. *Uživatelským XDG adresářům* jsem přiřadil *typy*: `documents_home_t` adresáři s uživatelskými dokumenty, `downloads_home_t` adresáři se staženými soubory, `music_home_t` adresáři s hudbou, `pictures_home_t` adresáři s obrázky, `public_home_t` adresáři s veřejným obsahem, `templates_home_t` adresáři se šablonami a adresáři s videi *typ* `videos_home_t`. Všechny tyto *typy* jsou definované v *modulu* `userdomain`, kam podle mé úvahy patří, protože se týkají adresářů *uživatelského prostoru* využívaných různými aplikacemi i samotným uživatelem. Jsou tak značně podobné *typu* pro domovský adresář uživatele.

Nutno však podotknout, že se nejedná o optimální řešení. Tyto adresáře mohou být vytvořeny až za běhu systému nějakou pomocnou aplikací, jako například `xdg-user-dirs`, která by tak měla být sama uzavřena. Dále pak mnou navrhovaný způsob implementace neřeší, že *uživatelské XDG adresáře* jsou lokalizované do národních jazyků. K tomuto je však potřeba přistupovat v rámci tvorby politiky k příslušným konfiguračním souborům či systémovým proměnným, což nyní není podporováno. Mé řešení tak pouze poskytuje příslušné *typy* a přístup k nim a dořešení celého problému nechává v souladu s použitou metodikou vývoje na později, až se z nižších vrstev politiky dostatečně ukáže, co přesně je potřeba.

### 9.3.2 Poskytovaná rozhraní

Konečně jsou součástí úprav *modulu* `userdomain` i přidaná *rozhraní* pro práci s novými *typy*. Tato *rozhraní* jsou pojmenována stejně jako v případě *politiky Fedory*, i jejich obsah je ve většině případů shodný. Pouze uváděné *typy* jsou všechny deklarovány v *modulu userdomain*. Příkladem takového *rozhraní* může být to, které povoluje vytvořit symbolický odkaz v adresáři s daty aplikací:

```
########################################
## <summary>
##      Create user .local directory symlinks.
## </summary>
## <param name="domain">
##      <summary>
##      Domain allowed access.
##      </summary>
## </param>
#
interface(‘userdom_create_data_home_symlinks’,‘
        gen_require(‘
                type data_home_t;
                type gconf_home_t;
        ’)
```

```
        create_lnk_files_pattern($1,data_home_t,data_home_t)
        userdom_search_user_home_dirs($1)
')
```

Kromě *rozhraní* pro přístup k novým *typům* jsem vytvořil i *rozhraní* umožňující *objektový přechod* zvolené *třídy objektů* vytvářených v těchto adresářích do požadovaného *typu*.

## 9.4 Úpravy modulu mysql Referenční politiky

Jak již bylo uvedeno v sekci 8.2 na straně 62, požaduje *PIM* systém *Akonadi* spouštění *MySQL* databáze bez *doménového přechodu*. Takovou službu ale současná politika pro *MySQL* neposkytuje. Napsal jsem tedy jednochý patch přidávající do *modulu* `mysql` následující *rozhraní*, jež spuštění bez *doménového přechodu* umožňuje:

```
########################################
## <summary>
##      Execute notrans MySQL
## </summary>
## <param name="domain">
##      <summary>
##      Domain allowed access
##      </summary>
## </param>
#
interface(‘mysql_can_exec’,‘
        gen_require(‘
                type mysqld_exec_t;
        ’)
        can_exec(akonadi_t, mysqld_exec_t)
’)
```

## 9.5 Úpravy modulů rolí Referenční politiky

Aby bylo možné vytvořenou politiku používat, je potřeba zvoleným *rolím* přidat *typy* uzavřených aplikací a služeb a *typu* dané *role* povolit určité přístupy k *doméně* aplikace či služby. Toto řeší *rozhraní přístupu rolí* jednotlivých *modulů*, která jsou pojmenovaná jako `modul_role()`.

Poslední úpravou *Referenční politiky* tak je přidání volání těchto *rozhraní* jednotlivým politikám *rolí*. Protože se vytvářená politika týká uživatelských aplikací a služeb, zvolil jsem *moduly rolí* `unconfined`, `unprivuser` a `staff`. Do souborů

s pravidly jsem tedy přidal *nepovinné bloky* obsahující volání *rozhraní přístupu rolí*. Následuje příklad pro *modul* `unconfined`:

```
optional_politicy(‘
        kde_role(unconfined_r, unconfined_t)
’)
optional_politicy(‘
        akonadi_role(unconfined_r, unconfined_t)
’)
optional_politicy(‘
        konqueror_role(unconfined_r, unconfined_t)
’)
optional_politicy(‘
        kmail_role(unconfined_r, unconfined_t)
’)
optional_politicy(‘
        kaddressbook_role(unconfined_r, unconfined_t)
’)
optional_politicy(‘
        kwallet_role(unconfined_r, unconfined_t)
’)
```

# Závěr

Mým cílem v této diplomové práci bylo vytvořit vzorové *bezpečnostní politiky SELinuxu* pro vybrané aplikace *uživatelského prostoru* reprezentovaného *pracovním prostředím KDE*. Zodpovědně jsem tedy prozkoumal teorii spojenou s technologickými postupy vývoje *bezpečnostních politik SELinuxu*. Poté jsem analyzoval *uživatelský prostor*, kde jsem se důkladně věnoval *pracovnímu prostředí KDE* a především pak aplikacím, pro které jsem zamýšlel vytvořit *bezpečnostní politiky*. Následně jsem uvažoval nad tím, jakým způsobem přistupovat k procesu uzavírání *uživatelského prostoru* tak, aby to bylo v souladu s vhodně stanovenými *bezpečnostními cíli*, ale i s rozsahovými a zejména časovými omezeními této práce. Nabyté teoretické znalosti doplněné o mé zkoumání *uživatelského prostoru* jsem pak uplatnil při vytváření vzorových politik pro vybrané aplikace a služby *prostředí KDE*.

V teoretické rovině považuji za nejdůležitější přínos této práce vytvoření metodiky pro přístup k tvorbě *bezpečnostních politik uživatelského prostoru* „zdola nahoru“. Zde jsem vztáhl princip postupných indukcí na tvorbu *bezpečnostní politiky*. Vytvářejí se tak nejprve *moduly* koncových aplikací, při jejichž vývoji vyvstávají požadavky na uzavření služeb *pracovního prostředí*. Ty se zhmotňují do *modulů* pro tyto služby, přičemž zpětně ovlivňují politiku již vytvořených aplikačních *modulů*, ale také samy vznášejí své požadavky k vyšším vrstvám *bezpečnostní politiky*. Takto dochází k přenášení požadavků výše spolu s jejich reflexí v nižších vrstvách, čímž se vytváří *bezpečnostní politika* směrem od aplikací, se kterými uživatel pracuje, přes služby, které tyto aplikace využívají, až k zastřešujícím *modulům* pracovního prostředí a *uživatelského prostoru*.

Vlastním přínosem v praktické rovině je vytvořená politika pro vybrané aplikace a služby *prostředí KDE*. Tato politika se integruje s *Referenční politikou SELinuxu* i *politikou Fedory*, na které jsem ji vyvíjel a testoval. Politika uzavírá poštovní klient *KMail*, adresář *KAddressBook* a rozšiřuje politiku pro webový prohlížeč *Konqueror*. Vzhledem k použité metodice tvorby politiky pro *uživatelský prostor* zapříčinilo uzavírání těchto aplikací i vytvoření politik pro služby *prostředí KDE*: *PIM* systém *Akonadi* a klíčenku *KWallet*. Jejich bezpečnostní požadavky se pak promítly do rozšíření zastřešujícího *modulu prostředí KDE*, do vytvoření *modulu kompatibility s politikou Fedory* a do úprav *modulů Refereční politiky*.

Metodika tvořící *bezpečnostní politiku* „zdola nahoru“ je obecným principem, který lze využít nejen k rozšiřování mnou vytvořené politiky pro *prostředí KDE*, ale také k vytvoření politiky pro *prostředí GNOME* či jiné části *uživatelského prostoru*. Nejvíce vhodná je tam, kde vyniknou její výhody spočívající především v poměrně rychlém vytvoření relativně kvalitní politiky pro koncové aplikace, což například umožní brzké zkušební nasazení.

Úplnost vytvořených *bezpečnostních politik* vychází z použité metodiky jejich vývoje. Politiky pro aplikace *KMail*, *KAddressBook* a *Konqueror* jsou použitelné v souladu se zde definovými případy užití těchto aplikací. Tyto politiky jsou tak

poměrně komplexní a vzhledem k časovému rámci práce i důkladně otestované. Politiky pro služby *Akonadi* a *KWallet* jsem vyvíjel pro spolupráci s výše uvedenými aplikacemi. Z jejich hlediska jsou tedy také poměrně robustní a testované. Tyto služby však jsou využívány i jinými aplikacemi, se kterými jsem funkčnost politik nezkoumal a pro které bude nejspíše – v duchu nasazené metodiky – potřeba *bezpečnostní politiky* dále rozšířit. Ještě silněji to pak platí pro politiku zastřešující *prostředí KDE* a změny provedené v politikách *uživatelského prostoru*. Ty jsem testoval a vyvíjel na základě požadavků zmíněných koncových aplikací, což je však samozřejmě jen malá část z toho, co musí obsloužit. Celkově pak vidím vytvořenou politiku jako kompletní v tom smyslu, že splňuje svůj hlavní cíl, tedy uzavřít vybrané aplikace. Zároveň je však potřeba si uvědomit, že z pohledu celého *uživatelského prostoru* či *prostředí KDE* je vytvořená politika pouze kapkou v moři a nelze tak od ní očekávat, že bude sloužit k něčemu, pro co nebyla vyvíjena. Vytvořená politika tak může být vhodným ukazatelem jednoho z přístupů k uzavírání *uživatelského prostoru*. Může také jasně poukázat na komplexnost řešeného problému a odhalit obtíže, které na vývojáře *bezpečnostních politik* pro *uživatelský prostor* čekají.

Jak jsem již výše naznačil, jakkoliv je vytvořená politika co do počtu řádek kódu rozsáhlá, týká se jen tří aplikací, dvou služeb a provádí pár změn v *modulech* vyšších úrovní. Zbývá zde tak mnoho prostoru k jejímu dalšímu rozšíření. Nejlogičtějším návazným krokem by bylo uzavření zbývajících aplikací *KDE PIM*. Ze střednědobého hlediska jsou nejpravděpodobnějším cílem aplikace přistupující k síti a dlouhodobě se lze snažit o uzavření celého *prostředí KDE*. Ve více obecné rovině se pak nabízí k důkladnějšímu prozkoumání obrácený postup, totiž tvorba politiky *uživatelského prostoru* „shora dolů“.

Jak ukázaly předchozí stránky, uzavírání *uživatelského prostoru bezpečnostními politikami SELinuxu* není snadnou záležitostí. Jde o komplexní a dlouhodobý proces, jehož výsledky v podobě značně vyšší bezpečnosti nemusí každý umět adekvátně ocenit. Důležitost *počítačové bezpečnosti* si totiž mnozí začínají uvědomovat až v okamžiku, kdy čelí následkům jejího selhání. Zde se tedy ukazuje prostor pro širší společenskou debatu, která by pozvedla povědomí o *počítačové bezpečnosti* a jejím významu v dnešním počítači řízeném světě.

## Seznam zkratek

**ACL** *Access Control Lists* – seznamy *řízení přístupu* umožňující jemnější nastavení přístupových práv v *GNU/Linuxu*

**AVC** *Access Vector Cache* – vyrovnávací paměť *přístupových vektorů* udržující předešlá rozhodnutí o udělení či zamítnutí přístupu a tím optimalizující vyhodnocování *pravidel SELinuxu*

**CLI** *Command Line Interface* – rozhraní příkazové řádky

**CVE** *Common Vulnerabilities and Exposures* – rejstřík identifikátorů veřejně známých bezpečnostních chyb softwaru

**DAC** *Discretionary Access Control – diskrétní řízení přístupu*

**GNU** *GNU's Not Unix* – projekt zaštiťující základní nástroje kolem linuxového jádra a spolu s ním vytvářející operační systém *GNU/Linux*

**GPL** *General Public License* – licence pro *svobodný software* vyžadující zachování změn v něm jako svobodných zveřejněním jejich zdrojového kódu

**GUI** *Graphical User Interface* – grafické uživatelské rozhraní

**IM** *Instant Messaging* – internetová služba zasílání okamžitých zpráv mezi uživateli

**IPC** *Inter-Process Communication* – meziprocesová komunikace

**KDE** *K Desktop Environment* – komunita [41] vyvíjející jedno z hlavních *pracovních prostředí* pro operační systém *GNU/Linux*

**LSM** *Linux Security Modules* – bezpečnostní moduly pro linuxové jádro umožňující rozšíření *Linuxu* o různé modely *řízení přístupu*

**MAC** *Mandatory Access Control – povinné řízení přístupu*

**MCS** *Multi Category Security – řízení přístupu* rozlišující nehierarchické *kategorie* objektů a subjektů

**MLS** *Multi Level Security – řízení přístupu* pracující s hierarchickými *úrovněmi*

**NSA** *National Security Agency* – Národní bezpečnostní agentura vlády Spojených států amerických

**PIM** *Personal Information Manager* – aplikační software pro správu osobních informací (kontaktů, úkolů, pošty, ...)

[41] Původní význam zkratky by již měl být zapomenut.

**RBAC** *Role-Based Access Control – řízení přístupu založené na rolích*

**RDF** *Resource Description Framework* – standardizovaný formát pro uchování sémantických dat

**SUID** *Set User ID* – příznak u souboru umožňující spuštění procesu s právy jeho vlastníka namísto s právy uživatele, který jej spouští

**TE** *Type Enforcement* – způsob *řízení přístupu* spočívající v povolování přístupu na základě *typů* subjektů a objektů

**UBAC** *User-Based Access Control – řízení přístupu* založené na *SELinux uživatelích* používané v *Referenční politice*

**XDG** *X Desktop Group* – dřívější název projektu *freedesktop.org*

# Reference


[1] Department of Defence: *Trusted Computer System Evaluation Criteria* [online]. 1985. Dostupný také z WWW: <http://www.fas.org/irp/nsa/rainbow/std001.htm>.

[2] ISO/IEC 15408: *Common Criteria for Information Technology Security Evaluation* [online]. [cit. 2011-03-02]. Dostupný z WWW: <http://www.commoncriteriaportal.org/cc/>.

[3] Open Security Architecture: *IT Security* [online]. [cit. 2011-03-03]. Dostupný z WWW: <http://www.opensecurityarchitecture.org/cms/en/definitions/it-security>.

[4] Loscocco, P. A. – Smalley, S. D. – Muckelbauer, P. A. – Taylor, R. C. – Turner, S. J. – Farrell, J. F.: *The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments* [online]. 1998. Dostupný z WWW: <http://www.nsa.gov/research/_files/selinux/papers/inevit-abs.shtml>.

[5] Christey, S. – Martin, R. A.: *Vulnerability Type Distributions in CVE (2001 – 2006)* [online]. 22. 5. 2007 [cit. 2011-04-17]. Dostupný z WWW: <http://cve.mitre.org/docs/vuln-trends/vuln-trends.pdf>.

[6] Neuhaus, S. – Zimmermann, T.: *Security Trend Analysis with CVE Topic Models* v Proceedings of the 21$^{st}$ IEEE International Symposium on Software Reliability Engineering (ISSRE 2010), San Jose, Kalifornie, USA, Listopad 2010. Dostupný také z WWW: <http://thomas-zimmermann.com/publications/files/neuhaus-issre-2010.pdf>.

[7] Grünbacher, A.: *POSIX Access Control Lists on Linux* [online]. 2003. Dostupný z WWW: <http://www.suse.de/~agruen/acl/linux-acls/online/>.

[8] ABCLinuxu.cz *Učebnice GNU/Linuxu: Přístupová práva* [online]. Poslední změna 20090-03-04 [cit. 2011-03-10]. ISSN 1214-1267. Dostupný z WWW: <http://www.abclinuxu.cz/ucebnice/zaklady/principy-prace-se-systemem/pristupova-prava>.

[9] Wright, C. – Cowan, C. – Smalley, S. D. – Moris J. – Kroah-Hartman, G.: *Linux Security Module Framework* [online]. 2002. Dostupný z WWW: <http://www.kroah.com/linux/talks/ols_2002_lsm_paper/lsm.pdf>.

[10] Mayer, F. – MacMillan, L. – Caplan, K.: *SELinux by Example: Using Security Enhanced Linux.* 1$^{st}$ Ed. Upper Saddle River, Prentice Hall 2006. 456 s. ISBN-10: 0-13-196369-4.

[11] Ferraiolo, D. F. – Kuhn, D. R.: *Role-Based Access Control* [online]. 1992. 15$^{th}$ National Computer Security Conference. str. 554 – 563. Dostupný z WWW: <http://csrc.nist.gov/groups/SNS/rbac/documents/ferraiolo-kuhn-92.pdf>.

[12] Bell, D. E.: *Looking Back at the Bell-La Padula Model* [online]. 2005. $21^{st}$ Annual Computer Security Applications Conference. Tucson, Arizona, USA. str. 337 – 351. Dostupný z WWW: <http://www.acsac.org/2005/papers/Bell.pdf>.

[13] PeBenito, Ch. J. – Mayer, F. – MacMillan, K.: *Reference Policy for Security Enhanced Linux* [online]. 2006. Dostupný z WWW: <http://www.tresys.com/pdf/Reference-Policy-for-SELinux.pdf>.

[14] Tresys Technology: *SELinux Reference Policy* [online] [cit. 2011-03-23]. Dostupný z WWW: <http://oss.tresys.com/projects/refpolicy/>.

[15] Tresys Technology: *SELinux Reference Policy API* [online] [cit. 2011-03-23]. Dostupný z WWW: <http://oss.tresys.com/docs/refpolicy/api/>.

[16] Tresys Technology: *Getting Started with Reference Policy* [online] [cit. 2011-03-23]. Dostupný z WWW: <http://oss.tresys.com/projects/refpolicy/wiki/GettingStarted>.

[17] Tresys Technology: *Reference Policy Style Guide* [online] [cit. 2011-03-23]. Dostupný z WWW: <http://oss.tresys.com/projects/refpolicy/wiki/StyleGuide>.

[18] Tresys Technology: *Reference Policy Interface and Template Naming Conventions* [online] [cit. 2011-03-23]. Dostupný z WWW: <http://oss.tresys.com/projects/refpolicy/wiki/InterfaceNaming>.

[19] Heines, R.: *The SELinux Notebook: Volume 1 The Foundations.* [online] $2^{nd}$ Ed. 2010. 314 s. Dostupný z WWW: <http://www.freetechbooks.com/efiles/selinuxnotebook/The_SELinux_Notebook_Volume_1_The_Foundations.pdf>.

[20] Walsh, D.: *A step-by-step Guide to Building a New SELinux Policy Module* [online]. Red Hat Magazine, ISSN: 1933-7973. Poslední změna 2007-08-21 [citováno 2011-03-26]. Dostupný z WWW: <http://magazine.redhat.com/2007/08/21/a-step-by-step-\guide-to-building-a-new-selinux-policy-module/>.

[21] Walsh, D.: *Unconfined Domain* [online]. 6. 12. 2006 [citováno 2011-03-28]. Dostupný z WWW: <http://danwalsh.livejournal.com/9031.html>.

[22] Grift, D.: *SELinux Lockdown Part Eight: Unconfined* [online]. 26. 6. 2009 [citováno 2011-03-28]. Dostupný z WWW: <http://selinux-mac.blogspot.com/2009/06/selinux-lockdown-part-eight-unconfined.html>.

[23] Grift, D.: *My thoughts on the user domain* [online]. 17. 7. 2010 [citováno 2011-03-28]. Dostupný z WWW: <http://selinux-mac.blogspot.com/2010/07/my-thoughts-on-user-domain.html>.

[24] Walsh, D.: *Introducing the SELinux Sandbox* [online]. 26. 5. 2009 [citováno 2011-03-28]. Dostupný z WWW: <http://danwalsh.livejournal.com/28545.html>.

[25] Walsh, D.: *Cool things with SELinux... Introducing sandbox -X* [online]. 16. 9. 2009 [citováno 2011-03-28]. Dostupný z WWW: <http://danwalsh.livejournal.com/31146.html>.

[26] Walsh, D.: *Confining the User with SELinux* [online]. 27. 5. 2007 [citováno 2011-03-30]. Dostupný z WWW: <http://danwalsh.livejournal.com/10461.html>.

[27] Walsh, D.: *Confining User Space* [online]. 4. 2. 2008 [citováno 2011-03-28]. Dostupný z WWW: <http://danwalsh.livejournal.com/15700.html>.

[28] Grift, D.: *Targeted configured semi-strict with UBAC for Fedora/Redhat distros* [online]. 12. 7. 2010 [citováno 2011-03-28]. Dostupný z WWW: <http://selinux-mac.blogspot.com/2010/07/targeted-configured-semi-strict-with.html>.

[29] Grift, D.: *My Fedora 14 policy for the adventurous* [online]. 29. 11. 2010 [citováno 2011-03-28]. Dostupný z WWW: <http://selinux-mac.blogspot.com/2010/11/my-fedora-14-policy-for-adventurous.html>.

[30] Luňák, L.: *Co je KDE: KDE re-branding* [online]. 18. 10. 2010 [cit. 2011-04-02]. Dostupný z WWW: <http://www.abclinuxu.cz/blog/Seli/2010/10/co-je-kde-kde-re-branding>.

[31] KDE.org: *KDE Software Compilation* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://kde.org/community/whatiskde/softwarecompilation.php>.

[32] KDE.org: *The KDE Development Platform* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://kde.org/developerplatform/>.

[33] KDE.org: *The KDE Workspaces* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://kde.org/workspaces/>.

[34] KDE.org: *The KDE Applications* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://kde.org/applications/>.

[35] Kügler, S.: *Demystifying Akonadi* [online]. 26. 10. 2010 [cit. 2011-04-02]. Dostupný z WWW: <http://vizzzion.org/blog/2010/08/demystifying-akonadi/>.

[36] KDE Community Wiki: *KDE PIM Akonadi* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://community.kde.org/KDE_PIM/Akonadi>.

[37] McGuire, T.: *Akonadi, Nepomuk and Strigi Explained* [online]. 3. 10. 2009 [cit. 2011-04-02]. Dostupný z WWW: <http://thomasmcguire.wordpress.com/2009/10/03/akonadi-nepomuk-and-strigi-explained/>.

[38] Staikos, G. – Watts, L.: *KWallet Handbook: Introduction* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://docs.kde.org/development/en/kdeutils/kwallet/introduction.html>.

[39] KDE User Base: *Kontact* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://userbase.kde.org/Kontact>.

[40] KDE User Base: *Kmail* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://userbase.kde.org/Kmail>.

[41] KDE User Base: *KAddressBook 4.4* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://userbase.kde.org/KAddressBook_4.4>.

[42] KDE User Base: *Konqueror* [online]. [cit. 2011-04-02]. Dostupný z WWW: <http://userbase.kde.org/Konqueror>.

[43] Freedesktop.org: *Software: D-Bus* [online]. [cit. 2011-04-03]. Dostupný z WWW: <http://www.freedesktop.org/wiki/Software/dbus>.

[44] Freedesktop.org: *XDG Base Directory Specification* [online]. [cit. 2011-04-03]. Dostupný z WWW: <http://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html>.

[45] Freedesktop.org: *Software: XDG-user-dirs* [online]. [cit. 2011-04-03]. Dostupný z WWW: <http://freedesktop.org/wiki/Software/xdg-user-dirs>.

# Přílohy

## I Modul kmail.pp

### I Soubor s kontexty (kmail.fc)

```
# KMail binaries
/usr/bin/kmail.* -- \
     gen_context(system_u:object_r:kmail_exec_t,s0)
# KMail configuration files
     HOME_DIR/\.kde/share/config/kmail(.*)? -- \
gen_context(system_u:object_r:kmail_home_conf_t,s0)
     HOME_DIR/\.kde/share/config/emaildefaults -- \
gen_context(system_u:object_r:kmail_home_conf_t,s0)
     HOME_DIR/\.kde/share/config/emailidentities -- \
gen_context(system_u:object_r:kmail_home_conf_t,s0)
     HOME_DIR/\.kde/share/config/kpgprc -- \
gen_context(system_u:object_r:kmail_home_conf_t,s0)
     HOME_DIR/\.kde/share/config/mailtransports -- \
gen_context(system_u:object_r:kmail_home_conf_t,s0)

# KMail data files in home
HOME_DIR/\.kde/share/apps/emailidentities(/.*)? \
     gen_context(system_u:object_r:kmail_home_data_t,s0)
HOME_DIR/\.kde/share/apps/kmail(/.*)? \
     gen_context(system_u:object_r:kmail_home_data_t,s0)

# KMail data files in .local due to Akonadi
HOME_DIR/\.local/share/\.local-mail\.directory(/.*)? \
     gen_context(system_u:object_r:kmail_data_home_t,s0)
HOME_DIR/\.local/share/local-mail(/.*)? \
     gen_context(system_u:object_r:kmail_data_home_t,s0)
```

### II Soubor s pravidly (kmail.te)

```
policy_module(kmail,0.3.0)

########################################
```

```
#
# KMail personal declarations
#

## <desc>
## <p>
## Allow KMail read documents_home_t to load user's documents
## </p>
## </desc>

gen_tunable(kmail_read_documents_home_t, false)

## <desc>
## <p>
## Allow KMail manage downloads_home_t to save mail attachements
## </p>
## </desc>

gen_tunable(kmail_manage_downloads_home_t, true)

## <desc>
## <p>
## Allow KMail read audio_home_t to load user's music
## </p>
## </desc>

gen_tunable(kmail_read_music_home_t, false)

## <desc>
## <p>
## Allow KMail read pictures_home_t to load user's pictures
## </p>
## </desc>

gen_tunable(kmail_read_pictures_home_t, false)

## <desc>
## <p>
## Allow KMail read public_home_t to load user's public files
## </p>
## </desc>

gen_tunable(kmail_read_public_home_t, true)
```

```
## <desc>
## <p>
## Allow KMail read videos_home_t to load user's videos
## </p>
## </desc>

gen_tunable(kmail_read_videos_home_t, false)

type kmail_t;
type kmail_exec_t;
application_domain(kmail_t, kmail_exec_t)
ubac_constrained(kmail_t)

type kmail_data_home_t;
userdom_user_home_content(kmail_data_home_t)

type kmail_home_conf_t;
userdom_user_home_content(kmail_home_conf_t)

type kmail_home_data_t;
userdom_user_home_content(kmail_home_data_t)

type kmail_tmp_t;
files_tmp_file(kmail_tmp_t)
ubac_constrained(kmail_tmp_t)

########################################
#
# KMail local policy
#

allow kmail_t self:fifo_file rw_file_perms;
allow kmail_t self:process signal_perms;

# Full access to kmail home
kmail_manage_data_home_files(kmail_t)
kmail_manage_data_home_dirs(kmail_t)
kmail_manage_home_conf_files(kmail_t)
kmail_manage_home_data_files(kmail_t)
kmail_manage_home_data_dirs(kmail_t)

# allow to run bin_t (drkonqi, gpg)
corecmd_exec_bin(kmail_t)
dev_read_urand(kmail_t) #/dev/urandom
```

```
# Access to dbus
dbus_connect_session_bus(kmail_t)
dbus_session_bus_client(kmail_t)

files_read_etc_files(kmail_t)
files_read_usr_files(kmail_t)
# extended atributes support, aka get_attr for /
fs_getattr_xattr_fs(kmail_t)
gpg_domtrans(kmail_t) # exec /usr/bin/gpg2
kernel_read_system_state(kmail_t) #/proc (cpuinfo)
miscfiles_read_localization(kmail_t) #/etc/localtime

# KMail interacts with Akonadi, which uses MySQL
# that is, if Akonadi is not already running,
# KMail tries to run it.
mysql_domtrans_mysql_safe(kmail_t)

sysnet_dns_name_resolve(kmail_t) #dns, resolve.conf?

userdom_use_user_terminals(kmail_t) #run from terminal
#signull to userdomain
userdom_signull_unpriv_users(kmail_t)
userdom_list_config_home_dirs(kmail_t)
#read, write .config/enchant spellchecking
userdom_read_config_home_files(kmail_t)
userdom_write_config_home_files(kmail_t)
userdom_search_data_home_dirs(kmail_t)
#read mime/genericicons
userdom_read_data_home_files(kmail_t)

# X access
xserver_user_x_domain_template(kmail, kmail_t, kmail_tmp_t)

#
# Tunable policies
#

tunable_policy(‘kmail_read_documents_home_t’,‘
     userdom_list_documents_home_dirs(kmail_t)
     userdom_read_documents_home_files(kmail_t)
     userdom_read_config_home_files(kmail_t)
’)
```

```
tunable_policy(‘kmail_manage_downloads_home_t’,‘
     userdom_manage_downloads_home_dirs(kmail_t)
     userdom_manage_downloads_home_files(kmail_t)
     userdom_read_config_home_files(kmail_t)
’)

tunable_policy(‘kmail_read_music_home_t’,‘
     userdom_manage_music_home_dirs(kmail_t)
     userdom_manage_music_home_files(kmail_t)
     userdom_read_config_home_files(kmail_t)
’)

tunable_policy(‘kmail_read_pictures_home_t’,‘
     userdom_list_pictures_home_dirs(kmail_t)
     userdom_read_pictures_home_files(kmail_t)
     userdom_read_config_home_files(kmail_t)
’)

tunable_policy(‘kmail_read_public_home_t’,‘
     userdom_list_public_home_dirs(kmail_t)
     userdom_read_public_home_files(kmail_t)
     userdom_read_config_home_files(kmail_t)
’)

tunable_policy(‘kmail_read_videos_home_t’,‘
     userdom_list_videos_home_dirs(kmail_t)
     userdom_read_videos_home_files(kmail_t)
     userdom_read_config_home_files(kmail_t)
’)

#
# Optional policies
#

# KMail needs akonadi
optional_policy(‘
     #akonadi socket configuration
     akonadi_list_config_home_dirs(kmail_t)
     akonadi_read_config_home_files(kmail_t)
     #akonadi error logs, if ako fails
     akonadi_read_data_home_files(kmail_t)
     akonadi_list_data_home_dirs(kmail_t)
     akonadi_write_data_home_sockets(kmail_t)
     #akonadi-firstrun, kresources
```

```
     akonadi_manage_home_conf_files(kmail_t)
     akonadi_rw_home_conf_dirs(kmail_t) #kresources
     akonadi_dbus_chat(kmail_t)
     akonadi_signull(kmail_t)
     akonadi_stream_connect(kmail_t)
     akonadi_domtrans(kmail_t)
')

# KMail uses KWallet to store passwords
optional_policy(‘
     kwallet_read_home_conf_files(kmail_t)
     kwallet_write_home_conf_files(kmail_t)
     kwallet_dbus_chat(kmail_t)
')

# KMail uses KAddressBook
optional_policy(‘
     kaddressbook_domtrans(kmail_t)
')

# Temp acces for kmail
# Access to kde_home_t, should be reduced in future
# Transition so that kmail_home_files in kde_home_t dir
# wouldn’t switch to parent directory type
optional_policy(‘
     kde_manage_tmp_files(kmail_t)
     kde_manage_tmp_sockets(kmail_t)
     kde_manage_tmp_symlinks(kmail_t)
     kde_manage_tmp_dirs(kmail_t)

     kde_read_config_home_files(kmail_t) #trolltech.conf
     kde_write_config_home_files(kmail_t)
     kde_read_home_symlinks(kmail_t) #tmp
     kde_rw_home_dirs(kmail_t)

     kde_read_home_conf_files(kmail_t) # kdeglobals
     kde_write_home_conf_files(kmail_t) # kdebugrc, kleopatra
     kde_home_conf_filetrans(kmail_t, \
          kmail_home_conf_t, { file })

     kde_read_home_conf_kio_files(kmail_t) #kio http
     kde_write_home_conf_kio_files(kmail_t)
     kde_read_home_data_files(kmail_t)
     kde_write_home_data_files(kmail_t) # kdebugrc, kleopatra
```

```
     kde_write_home_data_sockets(kmail_t) # nepomuk socket
     kde_home_data_filetrans(kmail_t, \
          kmail_home_data_t, { dir })
')
```

## III Soubor s rozhraními (kmail.if)

```
## <summary>Kmail KDE e-mail client</summary>

########################################
## <summary>
##     Role access for kmail
## </summary>
## <param name="role">
##     <summary>
##          Role allowed access
##     </summary>
## </param>
## <param name="domain">
##     <summary>
##          User domain for the role
##     </summary>
## </param>
#
interface(‘kmail_role’,‘
     gen_require(‘
          type kmail_t;
     ’)

     role $1 types kmail_t;

     # Allow the user domain to signal/ps.
     ps_process_pattern($2, kmail_t)
     allow $2 kmail_t:process signal_perms;
     allow kmail_t $2:process signal_perms;

     kmail_domtrans($2)
     kmail_stream_connect($2)
     # Allow user domain to dbus-chat with kmail
     optional_policy(‘
          kmail_dbus_chat($2)
     ’)
’)
```

```
interface(‘kmail_domtrans’,‘’)

interface(‘kmail_read_data_home_files’,‘’)
interface(‘kmail_manage_data_home_files’,‘’)
interface(‘kmail_manage_data_home_dirs’,‘’)

interface(‘kmail_read_home_conf_files’,‘’)
interface(‘kmail_write_home_conf_files’,‘’)
interface(‘kmail_manage_home_conf_files’,‘’)

interface(‘kmail_search_home_data’,‘’)
interface(‘kmail_read_home_data_files’,‘’)
interface(‘kmail_manage_home_data_files’,‘’)
interface(‘kmail_manage_home_data_symlinks’,‘’)
interface(‘kmail_manage_home_data_dirs’,‘’)

interface(‘kmail_signull’,‘’)
interface(‘kmail_stream_connect’,‘’)
interface(‘kmail_dbus_chat’,‘’)
```

# II Modul kaddressbook.pp

## I Soubor s kontexty (kaddressbook.fc)

```
# KAddressBook binries
/usr/bin/kaddressbook.* -- \
     gen_context(system_u:object_r:kaddressbook_exec_t,s0)

# KAddressBook configuration files in user home
HOME_DIR/\.kde/share/config/kaddressbook(.*)? -- \
    gen_context(system_u:object_r:kaddressbook_home_conf_t,s0)

# KAddressBook data files in user home
HOME_DIR/\.kde/share/apps/kabc(/.*)? \
    gen_context(system_u:object_r:kaddressbook_home_data_t,s0)
```

## II Soubor s pravidly (kaddressbook.te)

```
policy_module(kaddressbook,0.3.0)
```

```
########################################
#
# KAddressBook personal declarations
#

## <desc>
## <p>
## Allow KAddressBook manage downloads_home_t to save files
## </p>
## </desc>

gen_tunable(kaddressbook_manage_downloads_home_t, false)

## <desc>
## <p>
## Allow KAddressBook read pictures_home_t
## to load user's pictures
## </p>
## </desc>

gen_tunable(kaddressbook_read_pictures_home_t, false)

## <desc>
## <p>
## Allow KAddressBook read public_home_t
## to load user's public files
## </p>
## </desc>

gen_tunable(kaddressbook_read_public_home_t, true)

type kaddressbook_t;
type kaddressbook_exec_t;
application_domain(kaddressbook_t, kaddressbook_exec_t)
ubac_constrained(kaddressbook_t)

type kaddressbook_home_conf_t;
userdom_user_home_content(kaddressbook_home_conf_t)

type kaddressbook_home_data_t;
userdom_user_home_content(kaddressbook_home_data_t)

type kaddressbook_tmp_t;
files_tmp_file(kaddressbook_tmp_t)
```

```
ubac_constrained(kaddressbook_tmp_t)

########################################
#
# KAddressBook local policy
#

allow kaddressbook_t self:fifo_file rw_file_perms;
allow kaddressbook_t self:process signal_perms;

# Full access to kaddressbook home
kaddressbook_manage_home_conf_files(kaddressbook_t)
kaddressbook_manage_home_data_files(kaddressbook_t)
kaddressbook_manage_home_data_dirs(kaddressbook_t)

# allow to run bin_t (drkonqi, gpg)
corecmd_exec_bin(kaddressbook_t)
dev_read_urand(kaddressbook_t) #/dev/urandom

# Access to dbus
dbus_connect_session_bus(kaddressbook_t)
dbus_session_bus_client(kaddressbook_t)

files_read_etc_files(kaddressbook_t)
files_read_usr_files(kaddressbook_t)
# extended atributes support, aka get_attr for /
fs_getattr_xattr_fs(kaddressbook_t)

gpg_domtrans(kaddressbook_t) # exec /usr/bin/gpg2
kernel_read_system_state(kaddressbook_t) #/proc (meminfo)
miscfiles_read_localization(kaddressbook_t) #/etc/localtime

userdom_use_user_terminals(kaddressbook_t) #run from terminal
userdom_list_config_home_dirs(kaddressbook_t)
userdom_search_data_home_dirs(kaddressbook_t)

# X access
xserver_user_x_domain_template(kaddressbook, kaddressbook_t, \
     kaddressbook_tmp_t)

#
# Tunable policies
#
```

```
tunable_policy(‘kaddressbook_manage_downloads_home_t’,‘
     userdom_manage_downloads_home_dirs(kaddressbook_t)
     userdom_manage_downloads_home_files(kaddressbook_t)
     userdom_manage_config_home_files(kaddressbook_t)
’)

tunable_policy(‘kaddressbook_read_pictures_home_t’,‘
     userdom_list_pictures_home_dirs(kaddressbook_t)
     userdom_read_pictures_home_files(kaddressbook_t)
     userdom_manage_config_home_files(kaddressbook_t)
’)

tunable_policy(‘kaddressbook_read_public_home_t’,‘
     userdom_list_public_home_dirs(kaddressbook_t)
     userdom_read_public_home_files(kaddressbook_t)
     userdom_manage_config_home_files(kaddressbook_t)
’)

#
# Optional policies
#

# KAddressBook needs akonadi
optional_policy(‘
     akonadi_list_config_home_dirs(kaddressbook_t)
     #akonadi socket configuration
     akonadi_read_config_home_files(kaddressbook_t)
     #akonadi error logs, if ako fails
     akonadi_read_data_home_files(kaddressbook_t)
     akonadi_list_data_home_dirs(kaddressbook_t)
     akonadi_write_data_home_sockets(kaddressbook_t)
     akonadi_read_home_conf_files(kaddressbook_t)
     #akonadi-firstrun, kresources
     akonadi_manage_home_conf_files(kaddressbook_t)
     akonadi_manage_home_conf_dirs(kaddressbook_t) #kresources
     akonadi_dbus_chat(kaddressbook_t)
     akonadi_signull(kaddressbook_t)
     akonadi_stream_connect(kaddressbook_t)
     akonadi_domtrans(kaddressbook_t)
’)

# KMail integration
optional_policy(‘
     kmail_read_home_conf_files(kaddressbook_t)
```

```
     kmail_write_home_conf_files(kaddressbook_t)
')

# Temp acces for kaddressbook
# Access to kde_home_t, should be reduced in future
# Transition so that kaddressbook_home_files in kde_home_t dir
# wouldn't switch to parent directory type
optional_policy(‘
     kde_manage_tmp_files(kaddressbook_t)
     kde_manage_tmp_sockets(kaddressbook_t)
     kde_manage_tmp_symlinks(kaddressbook_t)
     kde_manage_tmp_dirs(kaddressbook_t)

     #.config/Trolltech.conf
     kde_read_config_home_files(kaddressbook_t)
     kde_write_config_home_files(kaddressbook_t)
     kde_read_home_symlinks(kaddressbook_t) #tmp

     kde_read_home_conf_files(kaddressbook_t) # kdeglobals
     kde_write_home_conf_files(kaddressbook_t) # kdebugrc
     kde_home_conf_filetrans(kaddressbook_t, \
          kaddressbook_home_conf_t, { file })

     #kioslaves integration
     kde_read_home_conf_kio_files(kaddressbook_t)
     kde_write_home_conf_kio_files(kaddressbook_t)

     # korganizer interaktion
     kde_manage_home_data_files(kaddressbook_t)
     kde_home_data_filetrans(kaddressbook_t, \
          kaddressbook_home_data_t, { dir })
')
```

## III Soubor s rozhraními (kaddressbook.if)

```
## <summary>KAddressBook KDE contacts manager</summary>

########################################
## <summary>
##     Role access for kaddressbook
## </summary>
## <param name="role">
##     <summary>
```

```
##          Role allowed access
##     </summary>
## </param>
## <param name="domain">
##     <summary>
##          User domain for the role
##     </summary>
## </param>
#
interface(`kaddressbook_role',`
     gen_require(`
          type kaddressbook_t;
     ')

     role $1 types kaddressbook_t;

     # Allow the user domain to signal/ps.
     ps_process_pattern($2, kaddressbook_t)
     allow $2 kaddressbook_t:process signal_perms;
     allow kaddressbook_t $2:process signal_perms;

     kaddressbook_domtrans($2)
     kaddressbook_stream_connect($2)
     # Allow user domain to dbus-chat with kaddressbook
     optional_policy(`
          kaddressbook_dbus_chat($2)
     ')
')

interface(`kaddressbook_domtrans',`')

interface(`kaddressbook_read_home_conf_files',`')
interface(`kaddressbook_write_home_conf_files',`')
interface(`kaddressbook_manage_home_conf_files',`')

interface(`kaddressbook_search_home_data',`')
interface(`kaddressbook_read_home_data_files',`')
interface(`kaddressbook_write_home_data_files',`')
interface(`kaddressbook_manage_home_data_files',`')
interface(`kaddressbook_manage_home_data_symlinks',`')
interface(`kaddressbook_rw_home_data_dirs',`')
interface(`kaddressbook_manage_home_data_dirs',`')

interface(`kaddressbook_signull',`')
```

```
interface(‘kaddressbook_stream_connect’,‘’)
interface(‘kaddressbook_dbus_chat’,‘’)
```

# III Modul konqueror.pp

## I Soubor s kontexty (konqueror.fc)

```
# Konqueror binary
     /usr/bin/konqueror -- \
gen_context(system_u:object_r:konqueror_exec_t,s0)

# Konqueror configuration inside kde_home_conf directory
HOME_DIR/\.kde/share/config/konq_history -- \
     gen_context(system_u:object_r:konqueror_home_conf_t,s0)
HOME_DIR/\.kde/share/config/konquerorrc -- \
     gen_context(system_u:object_r:konqueror_home_conf_t,s0)
HOME_DIR/\.kde/share/config/konqsidebartng.rc -- \
     gen_context(system_u:object_r:konqueror_home_conf_t,s0)
HOME_DIR/\.kde/share/config/kuriikwsfilterrc -- \
     gen_context(system_u:object_r:konqueror_home_conf_t,s0)

# Konqueror user data inside kde_home_data directory
HOME_DIR/\.kde/share/apps/konqueror(/.*)? \
     gen_context(system_u:object_r:konqueror_home_data_t,s0)
HOME_DIR/\.kde/share/apps/khtml(/.*)? \
     gen_context(system_u:object_r:konqueror_home_data_t,s0)
```

## II Soubor s pravidly (konqueror.te)

```
policy_module(konqueror,0.7.0)

########################################
#
# Konqueror personal declarations
#

## <desc>
## <p>
## Allow Konqueror read documents_home_t
## to load user’s documents
## </p>
```

```
## </desc>

gen_tunable(konqueror_read_documents_home_t, false)

## <desc>
## <p>
## Allow Konqueror manage downloads_home_t
## to save dowloaded files
## </p>
## </desc>

gen_tunable(konqueror_manage_downloads_home_t, true)

## <desc>
## <p>
## Allow Konqueror read audio_home_t to load user's music
## </p>
## </desc>

gen_tunable(konqueror_read_music_home_t, false)

## <desc>
## <p>
## Allow Konqueror read pictures_home_t
## to load user's pictures
## </p>
## </desc>

gen_tunable(konqueror_read_pictures_home_t, false)

## <desc>
## <p>
## Allow Konqueror read public_home_t
## to load user's public files
## </p>
## </desc>

gen_tunable(konqueror_read_public_home_t, true)

## <desc>
## <p>
## Allow Konqueror read videos_home_t to load user's videos
## </p>
## </desc>
```

```
gen_tunable(konqueror_read_videos_home_t, false)

type konqueror_t;
type konqueror_exec_t;
application_domain(konqueror_t, konqueror_exec_t)
ubac_constrained(konqueror_t)

type konqueror_home_conf_t;
userdom_user_home_content(konqueror_home_conf_t)

type konqueror_home_data_t;
userdom_user_home_content(konqueror_home_data_t)

type konqueror_tmp_t;
files_tmp_file(konqueror_tmp_t)
ubac_constrained(konqueror_tmp_t)

########################################
#
# Konqueror local policy
#

# Internal communication using fifo
allow konqueror_t self:fifo_file rw_file_perms;
# get self process priority, signull when runed program fails
allow konqueror_t self:process { getsched signull };
allow konqueror_t self:tcp_socket create_stream_socket_perms;
konqueror_dbus_chat(konqueror_t) # internal dbus comunication

# Full access to konqueror home configuration and data
konqueror_manage_home_conf_files(konqueror_t)
konqueror_manage_home_data_files(konqueror_t)
konqueror_manage_home_data_symlinks(konqueror_t)
konqueror_manage_home_data_dirs(konqueror_t)

# Allow exec bin_t (drkonqi, helper programs)
corecmd_exec_bin(konqueror_t)

# Access to ports
corenet_all_recvfrom_unlabeled(konqueror_t)
corenet_tcp_sendrecv_generic_if(konqueror_t)
corenet_tcp_sendrecv_generic_node(konqueror_t)
corenet_tcp_sendrecv_generic_port(konqueror_t)
```

```
corenet_tcp_connect_ftp_data_port(konqueror_t)
corenet_tcp_connect_ftp_port(konqueror_t)
corenet_tcp_connect_http_port(konqueror_t)
corenet_tcp_connect_http_cache_port(konqueror_t)

# Acces to dbus, needed for Konqueror to start
dbus_connect_session_bus(konqueror_t)
dbus_session_bus_client(konqueror_t)
dev_read_urand(konqueror_t) #/dev/urandom

files_read_etc_files(konqueror_t)
files_read_usr_files(konqueror_t) #/usr
fs_getattr_xattr_fs(konqueror_t) # extended atributes support
kernel_read_system_state(konqueror_t) #/proc

# Read localization and fonts
miscfiles_read_fonts(konqueror_t)
miscfiles_read_localization(konqueror_t)
sysnet_dns_name_resolve(konqueror_t)

userdom_use_user_terminals(konqueror_t) #run from terminal
userdom_signal_unpriv_users(konqueror_t)
userdom_signull_unpriv_users(konqueror_t)
userdom_read_config_home_files(konqueror_t) #spellchecking
userdom_write_config_home_files(konqueror_t)

# X access
xserver_user_x_domain_template(konqueror, \
     konqueror_t, konqueror_tmp_t)

#
# Tunable policies
#

tunable_policy(‘konqueror_read_documents_home_t’,‘
     userdom_list_documents_home_dirs(konqueror_t)
     userdom_read_documents_home_files(konqueror_t)
     userdom_read_config_home_files(konqueror_t)
’)

tunable_policy(‘konqueror_manage_downloads_home_t’,‘
     userdom_manage_downloads_home_dirs(konqueror_t)
     userdom_manage_downloads_home_files(konqueror_t)
     userdom_read_config_home_files(konqueror_t)
```

```
')

tunable_policy(‘konqueror_read_music_home_t’,‘
     userdom_list_music_home_dirs(konqueror_t)
     userdom_read_music_home_files(konqueror_t)
     userdom_read_config_home_files(konqueror_t)
')

tunable_policy(‘konqueror_read_pictures_home_t’,‘
     userdom_list_pictures_home_dirs(konqueror_t)
     userdom_read_pictures_home_files(konqueror_t)
     userdom_read_config_home_files(konqueror_t)
')

tunable_policy(‘konqueror_read_public_home_t’,‘
     userdom_list_public_home_dirs(konqueror_t)
     userdom_read_public_home_files(konqueror_t)
     userdom_read_config_home_files(konqueror_t)
')

tunable_policy(‘konqueror_read_videos_home_t’,‘
     userdom_list_videos_home_dirs(konqueror_t)
     userdom_read_videos_home_files(konqueror_t)
     userdom_read_config_home_files(konqueror_t)
')

#
# Optional policies
#

# Temp acces for konqueror
# Access to kde_home_t, should be reduced in future
# Transition so that konqueror_home_conf files in
# kde_home_conf_t dir wouldn’t switch to parent dir type
optional_policy(‘
     kde_manage_tmp_files(konqueror_t)
     kde_manage_tmp_sockets(konqueror_t)
     kde_manage_tmp_symlinks(konqueror_t)
     kde_manage_tmp_dirs(konqueror_t)

     kde_read_config_home_files(konqueror_t) #troltech.conf
     kde_write_config_home_files(konqueror_t)

     kde_read_home_symlinks(konqueror_t) # links to temp
```

```
     kde_read_home_conf_files(konqueror_t) # kdeglobals
     kde_write_home_conf_files(konqueror_t)
     kde_home_conf_filetrans(konqueror_t, \
          konqueror_home_conf_t, file)
     kde_read_home_conf_kio_files(konqueror_t)
     # kdebug_rc until drkonqi is confined
     kde_write_home_conf_kio_files(konqueror_t)

     kde_home_data_filetrans(konqueror_t, \
          konqueror_home_data_t, dir)
     # access to nsplugins
     kde_read_home_data_nsplugin_files(konqueror_t)
     kde_write_home_data_nsplugin_files(konqueror_t)
')

# Konqueror uses KWallet to store passwords
optional_policy(‘
     kwallet_read_home_conf_files(konqueror_t)
     kwallet_write_home_conf_files(konqueror_t)
     kwallet_dbus_chat(konqueror_t)
’)
```

### III Soubor s rozhraními (konqueror.if)

```
## <summary>Konqueror KDE web browser</summary>
########################################
## <summary>
##     Role access for konqueror
## </summary>
## <param name="role">
##     <summary>
##          Role allowed access
##     </summary>
## </param>
## <param name="domain">
##     <summary>
##          User domain for the role
##     </summary>
## </param>
#
interface(‘konqueror_role’,‘
     gen_require(‘
```

```
          type konqueror_t;
     ')

     role $1 types konqueror_t;

     # Allow the user domain to signal/ps.
     ps_process_pattern($2, konqueror_t)
     allow $2 konqueror_t:process signal_perms;

     konqueror_domtrans($2)
     konqueror_stream_connect($2)
     # Allow user domain to dbus-chat with konqueror
     optional_policy(`
          konqueror_dbus_chat($2)
     ')
')

interface(`konqueror_domtrans',`')

interface(`konqueror_read_home_conf_files',`')
interface(`konqueror_manage_home_conf_files',`')

interface(`konqueror_search_home_data',`')
interface(`konqueror_read_home_data_files',`')
interface(`konqueror_manage_home_data_files',`')
interface(`konqueror_manage_home_data_symlinks',`')
interface(`konqueror_manage_home_data_dirs',`')

interface(`konqueror_stream_connect',`')
interface(`konqueror_dbus_chat',`')
```

# IV Modul kwallet.pp

## I Soubor s kontexty (kwallet.fc)

```
# KWallet binaries
/usr/bin/kwalletd -- \
     gen_context(system_u:object_r:kwallet_exec_t,s0)
/usr/bin/kwalletmanager -- \
     gen_context(system_u:object_r:kwallet_exec_t,s0)

# KWallet konfiguration files
```

```
HOME_DIR/\.kde/share/config/kwalletrc -- \
     gen_context(system_u:object_r:kwallet_home_conf_t,s0)
HOME_DIR/\.kde/share/config/kwalletmanagerrc -- \
     gen_context(system_u:object_r:kwallet_home_conf_t,s0)

# KWallet data files
HOME_DIR/\.kde/share/apps/kwallet(/.*)? \
     gen_context(system_u:object_r:kwallet_home_data_t,s0)
```

### II Soubor s pravidly (kwallet.te)

```
policy_module(kwallet,0.2.0)

########################################
#
# KWallet personal declarations
#

type kwallet_t;
type kwallet_exec_t;
application_domain(kwallet_t, kwallet_exec_t)
ubac_constrained(kwallet_t)

type kwallet_home_conf_t;
userdom_user_home_content(kwallet_home_conf_t)

type kwallet_home_data_t;
userdom_user_home_content(kwallet_home_data_t)

type kwallet_tmp_t;
files_tmp_file(kwallet_tmp_t)
ubac_constrained(kwallet_tmp_t)

########################################
#
# KWallet local policy
#

allow kwallet_t self:fifo_file rw_file_perms;
allow kwallet_t self:process signal_perms;
kwallet_stream_connect(kwallet_t)

# Full access to kwallet home
```

```
kwallet_manage_home_conf_files(kwallet_t)
kwallet_manage_home_data_files(kwallet_t)
kwallet_manage_home_data_dirs(kwallet_t)

corecmd_exec_bin(kwallet_t) # allow to run bint (drkonqi)
corecmd_exec_shell(kwallet_t) # allow to run shell (bash)
dev_read_urand(kwallet_t) #/dev/urandom
dev_read_rand(kwallet_t) #/dev/random

# Access to dbus
dbus_connect_session_bus(kwallet_t)
dbus_session_bus_client(kwallet_t)
files_read_etc_files(kwallet_t)
files_read_usr_files(kwallet_t)
# extended atributes support, aka get_attr for /
fs_getattr_xattr_fs(kwallet_t)
kernel_read_system_state(kwallet_t) #/proc (cpuinfo)
miscfiles_read_localization(kwallet_t) #/etc/localtime

userdom_use_user_terminals(kwallet_t) #run from terminal
userdom_read_data_home_files(kwallet_t) #read generic icons

# X access
xserver_user_x_domain_template(kwallet, \
     kwallet_t, kwallet_tmp_t)

#
# Optional policies
#

# Temp acces for kwallet
# Access to kde_home_t, should be reduced in future
# Transition so that kwallet_home_files in kde_home_t dir
# wouldn't switch to parent directory type
optional_policy(‘
     kde_manage_tmp_files(kwallet_t)
     kde_manage_tmp_sockets(kwallet_t)
     kde_manage_tmp_symlinks(kwallet_t)
     kde_manage_tmp_dirs(kwallet_t)

     kde_read_config_home_files(kwallet_t) #trolltech.conf
     kde_read_home_symlinks(kwallet_t) #tmp
     kde_rw_home_dirs(kwallet_t)
     kde_read_home_conf_files(kwallet_t) # kdeglobals
```

```
     kde_write_home_conf_files(kwallet_t) # kdebugrc
     kde_home_conf_filetrans(kwallet_t, \
          kwallet_home_conf_t, { file })
     kde_read_home_data_files(kwallet_t)
     kde_home_data_filetrans(kwallet_t, \
          kwallet_home_data_t, { dir })
')
```

### III Soubor s rozhraními (kwallet.if)

```
## <summary>KWallet KDE keyring</summary>

########################################
## <summary>
##     Role access for kwallet
## </summary>
## <param name="role">
##     <summary>
##          Role allowed access
##     </summary>
## </param>
## <param name="domain">
##     <summary>
##         User domain for the role
##     </summary>
## </param>
#
interface(‘kwallet_role’,‘
     gen_require(‘
          type kwallet_t;
     ’)

     role $1 types kwallet_t;

     # Allow the user domain to signal/ps.
     ps_process_pattern($2, kwallet_t)
     allow $2 kwallet_t:process signal_perms;
     allow kwallet_t $2:process signal_perms;

     kwallet_domtrans($2)
     kwallet_stream_connect($2)
     # Allow user domain to dbus-chat with kwallet
     optional_policy(‘
```

```
          kwallet_dbus_chat($2)
     ')
')

interface(‘kwallet_domtrans’,‘’)

interface(‘kwallet_read_home_conf_files’,‘’)
interface(‘kwallet_write_home_conf_files’,‘’)
interface(‘kwallet_manage_home_conf_files’,‘’)

interface(‘kwallet_search_home_data’,‘’)
interface(‘kwallet_read_home_data_files’,‘’)
interface(‘kwallet_manage_home_data_files’,‘’)
interface(‘kwallet_manage_home_data_symlinks’,‘’)
interface(‘kwallet_manage_home_data_dirs’,‘’)

interface(‘kwallet_signull’,‘’)
interface(‘kwallet_stream_connect’,‘’)
interface(‘kwallet_dbus_chat’,‘’)
```

# V Modul akonadi.pp

## I Soubor s kontexty (akonadi.fc)

```
# Akonadi executables
/usr/bin/akonadi(.*)? -- \
     gen_context(system_u:object_r:akonadi_exec_t,s0)
# Akonadi .config configuration
HOME_DIR/\.config/akonadi(/.*)? \
     gen_context(system_u:object_r:akonadi_config_home_t,s0)
# Akonadi home configuration
HOME_DIR/\.kde/share/config/kresources(/.*)? \
     gen_context(system_u:object_r:akonadi_home_conf_t,s0)
HOME_DIR/\.kde/share/config/akonadi(.*)? \
     gen_context(system_u:object_r:akonadi_home_conf_t,s0)
HOME_DIR/\.kde/share/config/specialmailcollectionsrc \
     gen_context(system_u:object_r:akonadi_home_conf_t,s0)
# Akonadi data at .local
HOME_DIR/\.local/share/akonadi(/.*)? \
     gen_context(system_u:object_r:akonadi_data_home_t,s0)
HOME_DIR/\.local/share/contacts(/.*)? \
     gen_context(system_u:object_r:akonadi_data_home_t,s0)
```

## II Soubor s pravidly (akonadi.te)

```
policy_module(akonadi,0.2.0)

########################################
#
# Akonadi personal declarations
#

type akonadi_t;
type akonadi_exec_t;
application_domain(akonadi_t, akonadi_exec_t)
ubac_constrained(akonadi_t)

type akonadi_home_conf_t;
userdom_user_home_content(akonadi_home_conf_t)

type akonadi_config_home_t;
userdom_user_home_content(akonadi_config_home_t)

type akonadi_data_home_t;
userdom_user_home_content(akonadi_data_home_t)

type akonadi_tmp_t;
files_tmp_file(akonadi_tmp_t)
ubac_constrained(akonadi_tmp_t)

########################################
#
# Akonadi local policy
#

allow akonadi_t self:fifo_file rw_file_perms;
allow akonadi_t self:process signal_perms;
allow akonadi_t self:process setsched;
allow akonadi_t self:unix_stream_socket connectto;

# Temp access for akonadi
manage_dirs_pattern(akonadi_t, akonadi_tmp_t, akonadi_tmp_t)
manage_files_pattern(akonadi_t, akonadi_tmp_t, akonadi_tmp_t)
manage_lnk_files_pattern(akonadi_t, \
     akonadi_tmp_t, akonadi_tmp_t)
manage_sock_files_pattern(akonadi_t, \
     akonadi_tmp_t, akonadi_tmp_t)
```

```
# Full access to akonadi home
akonadi_manage_config_home_files(akonadi_t)
akonadi_manage_config_home_sockets(akonadi_t)
akonadi_manage_config_home_symlinks(akonadi_t)
akonadi_manage_config_home_dirs(akonadi_t)

akonadi_manage_data_home_files(akonadi_t)
akonadi_manage_data_home_sockets(akonadi_t)
akonadi_manage_data_home_symlinks(akonadi_t)
akonadi_manage_data_home_dirs(akonadi_t)

akonadi_manage_home_conf_files(akonadi_t)

# allow to run bin_t (install_mysql_db also Nepomuk)
corecmd_exec_bin(akonadi_t)
#Execute notrans other akonadi binaries
can_exec(akonadi_t, akonadi_exec_t)

# Access to dbus
dbus_connect_session_bus(akonadi_t)
dbus_session_bus_client(akonadi_t)

dev_read_urand(akonadi_t)
files_read_etc_files(akonadi_t)
files_read_usr_files(akonadi_t)
files_tmp_filetrans(akonadi_t, akonadi_tmp_t, { dir file })

# extended atributes support, aka get_attr for /
fs_getattr_xattr_fs(akonadi_t)
kernel_read_system_state(akonadi_t) #/proc (cpuinfo)
miscfiles_read_localization(akonadi_t) #/etc/localtime

# Akonadi uses MySQL to store metadata
mysql_read_config(akonadi_t)
mysql_can_exec(akonadi_t) #from mysqlpatch

userdom_use_user_terminals(akonadi_t) #run from terminal
userdom_manage_user_tmp_files(akonadi_t) #Mysql tmp files
# Due to contents in .config and .local
userdom_search_config_home_dirs(akonadi_t)
userdom_search_data_home_dirs(akonadi_t)
#.local/share/mime/generic-icons
userdom_read_data_home_files(akonadi_t)
```

```
userdom_data_home_filetrans(akonadi_t, \
     akonadi_data_home_t, dir)

# X access due to helper programs
xserver_user_x_domain_template(akonadi, \
     akonadi_t, akonadi_tmp_t)

#
# Optional policies
#

# Temp acces for akonadi
# Transition so that akonadi_home_files in kde_home_t dir
# wouldn't switch to parent directory type
optional_policy(‘
     kde_manage_tmp_files(akonadi_t)
     kde_manage_tmp_sockets(akonadi_t)
     kde_manage_tmp_symlinks(akonadi_t)
     kde_manage_tmp_dirs(akonadi_t)

     kde_read_config_home_files(akonadi_t) # Qt config
     kde_write_config_home_files(akonadi_t)
     kde_read_home_symlinks(akonadi_t) # links to temp

     kde_read_home_conf_files(akonadi_t) # kdeglobals
     kde_write_home_conf_files(akonadi_t) # kdebugrc
     # standard eg. korganizer resources
     kde_read_home_data_files(akonadi_t)
     kde_write_home_data_files(akonadi_t)
     kde_write_home_data_sockets(akonadi_t) # nepomuk
     kde_home_conf_filetrans(akonadi_t, \
          akonadi_home_conf_t, file)
’)

optional_policy(‘
     kmail_signull(akonadi_t)
’)

# Akonadi is used by KAddressBook as a cache for contacts
optional_policy(‘
     kaddressbook_manage_home_data_files(akonadi_t)
     kaddressbook_manage_home_data_dirs(akonadi_t)
’)
```

### III Soubor s rozhraními (akonadi.if)

```
## <summary>Akonadi KDE PIM service</summary>

########################################
## <summary>
##     Role access for akonadi
## </summary>
## <param name="role">
##     <summary>
##          Role allowed access
##     </summary>
## </param>
## <param name="domain">
##     <summary>
##          User domain for the role
##     </summary>
## </param>
#
interface(`akonadi_role',`
     gen_require(`
          type akonadi_t;
     ')

     role $1 types akonadi_t;

     # Allow the user domain to signal/ps.
     ps_process_pattern($2, akonadi_t)
     allow $2 akonadi_t:process signal_perms;
     allow akonadi_t $2:process signull;

     akonadi_domtrans($2)
     akonadi_stream_connect($2)
     # Allow user domain to dbus-chat with akonadi
     optional_policy(`
          akonadi_dbus_chat($2)
     ')
')

interface(`akonadi_domtrans',`')

interface(`akonadi_search_config_home',`')
interface(`akonadi_read_config_home_files',`')
interface(`akonadi_manage_config_home_files',`')
```

```
interface(‘akonadi_manage_config_home_sockets’,‘’)
interface(‘akonadi_manage_config_home_symlinks’,‘’)
interface(‘akonadi_list_config_home_dirs’,‘’)
interface(‘akonadi_manage_config_home_dirs’,‘’)

interface(‘akonadi_search_data_home’,‘’)
interface(‘akonadi_read_data_home_files’,‘’)
interface(‘akonadi_manage_data_home_files’,‘’)
interface(‘akonadi_read_data_home_sockets’,‘’)
interface(‘akonadi_write_data_home_sockets’,‘’)
interface(‘akonadi_manage_data_home_sockets’,‘’)
interface(‘akonadi_manage_data_home_symlinks’,‘’)
interface(‘akonadi_list_data_home_dirs’,‘’)
interface(‘akonadi_manage_data_home_dirs’,‘’)

interface(‘akonadi_read_home_conf_files’,‘’)
interface(‘akonadi_write_home_conf_files’,‘’)
interface(‘akonadi_manage_home_conf_files’,‘’)
interface(‘akonadi_rw_home_conf_dirs’,‘’)
interface(‘akonadi_manage_home_conf_dirs’,‘’)

interface(‘akonadi_signull’,‘’)
interface(‘akonadi_stream_connect’,‘’)
interface(‘akonadi_dbus_chat’,‘’)
```

# VI Modul kde.pp

## I Soubor s kontexty (kde.fc)

```
# Qt config file
HOME_DIR/\.config/Trolltech\.conf -- \
     gen_context(system_u:object_r:kde_config_home_t,s0)

# KDE home
HOME_DIR/\.kde(/.*)? \
     gen_context(system_u:object_r:kde_home_t,s0)

# KDE apps data
HOME_DIR/\.kde/share/apps(/.*)? \
     gen_context(system_u:object_r:kde_home_data_t,s0)

# KDE apps data nsplugins
```

```
HOME_DIR/\.kde/share/apps/nsplugins(/.*)? \
    gen_context(system_u:object_r:kde_home_data_nsplugin_t,s0)

# KDE apps config
HOME_DIR/\.kde/share/config(/.*)? \
     gen_context(system_u:object_r:kde_home_conf_t,s0)

# KDE KIO_slaves config
HOME_DIR/\.kde/share/config/kio(.*)? \
     gen_context(system_u:object_r:kde_home_conf_kio_t,s0)
```

### II Soubor s pravidly (kde.te)

```
policy_module(kde,0.3.0)

########################################
#
# Declarations
#

# Type declarations

type kde_config_home_t;
userdom_user_home_content(kde_config_home_t)

type kde_home_t;
userdom_user_home_content(kde_home_t)

type kde_home_data_t;
userdom_user_home_content(kde_home_data_t)

type kde_home_data_nsplugin_t;
userdom_user_home_content(kde_home_data_nsplugin_t)

type kde_home_conf_t;
userdom_user_home_content(kde_home_conf_t)

type kde_home_conf_kio_t;
userdom_user_home_content(kde_home_conf_kio_t)

type kde_tmp_t;
files_tmp_file(kde_tmp_t)
ubac_constrained(kde_tmp_t)
```

## III Soubor s rozhraními (kde.if)

```
########################################
## <summary>
##     Manage kde temp symlinks.
## </summary>
## <param name="domain">
##     <summary>
##          Domain allowed access.
##     </summary>
## </param>
#
interface(‘kde_manage_tmp_symlinks’,‘
     gen_require(‘
          type kde_tmp_t;
     ’)

     manage_lnk_files_pattern($1,kde_tmp_t,kde_tmp_t);
     # As these could be inside of user_tmp_t
     # access to it and transition is needed.
     userdom_read_user_tmp_symlinks($1)
     userdom_user_tmp_filetrans($1, kde_tmp_t, lnk_file)
’)

########################################
## <summary>
##     Manage kde temp sockets.
## </summary>
## <param name="domain">
##     <summary>
##          Domain allowed access.
##     </summary>
## </param>
#
interface(‘kde_manage_tmp_sockets’,‘
     gen_require(‘
          type kde_tmp_t;
     ’)

     manage_sock_files_pattern($1,kde_tmp_t,kde_tmp_t);
     # As these could be inside of user_tmp_t
     # access to it and transition is needed.
     userdom_write_user_tmp_sockets($1)
     userdom_user_tmp_filetrans($1, kde_tmp_t, sock_file)
```

```
')

########################################
## <summary>
##     Manage kde temp files.
## </summary>
## <param name="domain">
##     <summary>
##          Domain allowed access.
##     </summary>
## </param>
#
interface(`kde_manage_tmp_files',`
     gen_require(`
          type kde_tmp_t;
     ')

     manage_files_pattern($1,kde_tmp_t,kde_tmp_t);
     # As these could be inside of user_tmp_t
     # access to it and transition is needed.
     userdom_read_user_tmp_files($1)
     userdom_write_user_tmp_files($1)
     userdom_user_tmp_filetrans($1, kde_tmp_t, file)
')

########################################
## <summary>
##     Manage kde temp dirs.
## </summary>
## <param name="domain">
##     <summary>
##          Domain allowed access.
##     </summary>
## </param>
#
interface(`kde_manage_tmp_dirs',`
     gen_require(`
          type kde_tmp_t;
     ')

     manage_dirs_pattern($1,kde_tmp_t,kde_tmp_t);
     # As these could be inside of user_tmp_t
     # access to it and transition is needed.
     userdom_user_tmp_filetrans($1, kde_tmp_t, dir)
```

```
')

interface(`kde_read_config_home_files',`')
interface(`kde_write_config_home_files',`')

interface(`kde_search_home_dirs',`')
interface(`kde_list_home_dirs',`')
interface(`kde_rw_home_dirs',`')
interface(`kde_manage_home_dirs',`')
interface(`kde_read_home_files',`')
interface(`kde_write_home_files',`')
interface(`kde_create_home_files',`')
interface(`kde_manage_home_files',`')
interface(`kde_read_home_sockets',`')
interface(`kde_write_home_sockets',`')
interface(`kde_create_home_sockets',`')
interface(`kde_delete_home_sockets',`')
interface(`kde_manage_home_sockets',`')
interface(`kde_read_home_symlinks',`')
interface(`kde_create_home_symlinks',`')
interface(`kde_manage_home_symlinks',`')
interface(`kde_home_filetrans',`')

interface(`kde_search_home_conf_dirs',`')
interface(`kde_list_home_conf_dirs',`')
interface(`kde_rw_home_conf_dirs',`')
interface(`kde_manage_home_conf_dirs',`')
interface(`kde_read_home_conf_files',`')
interface(`kde_write_home_conf_files',`')
interface(`kde_create_home_conf_files',`')
interface(`kde_manage_home_conf_files',`')
interface(`kde_read_home_conf_sockets',`')
interface(`kde_write_home_conf_sockets',`')
interface(`kde_create_home_conf_sockets',`')
interface(`kde_delete_home_conf_sockets',`')
interface(`kde_manage_home_conf_sockets',`')
interface(`kde_read_home_conf_symlinks',`')
interface(`kde_create_home_conf_symlinks',`')
interface(`kde_manage_home_conf_symlinks',`')
interface(`kde_home_conf_filetrans',`')

interface(`kde_read_home_conf_kio_files',`')
interface(`kde_write_home_conf_kio_files',`')
interface(`kde_create_home_conf_kio_files',`')
```

```
interface(‘kde_manage_home_conf_kio_files’,‘’)

interface(‘kde_search_home_data_dirs’,‘’)
interface(‘kde_list_home_data_dirs’,‘’)
interface(‘kde_rw_home_data_dirs’,‘’)
interface(‘kde_manage_home_data_dirs’,‘’)
interface(‘kde_read_home_data_files’,‘’)
interface(‘kde_write_home_data_files’,‘’)
interface(‘kde_create_home_data_files’,‘’)
interface(‘kde_manage_home_data_files’,‘’)
interface(‘kde_read_home_data_sockets’,‘’)
interface(‘kde_write_home_data_sockets’,‘’)
interface(‘kde_create_home_data_sockets’,‘’)
interface(‘kde_delete_home_data_sockets’,‘’)
interface(‘kde_manage_home_data_sockets’,‘’)
interface(‘kde_read_home_data_symlinks’,‘’)
interface(‘kde_create_home_data_symlinks’,‘’)
interface(‘kde_manage_home_data_symlinks’,‘’)
interface(‘kde_home_data_filetrans’,‘’)

interface(‘kde_read_home_data_nsplugin_files’,‘’)
interface(‘kde_write_home_data_nsplugin_files’,‘’)
interface(‘kde_create_home_data_nsplugin_files’,‘’)
interface(‘kde_manage_home_data_nsplugin_files’,‘’)
```