VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

RODIČOVSKÝ MÓD V KDE

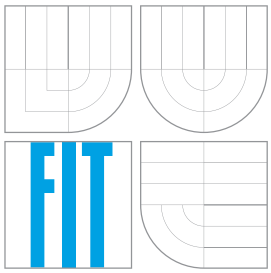
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR MRÁZEK

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

RODIČOVSKÝ MÓD V KDE

PARENTAL MODE IN KDE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR MRÁZEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JOZEF MLÍCH,
SUPERVISOR

BRNO 2010

Abstrakt

Práce popisuje rozhraní KAuthorized a KAuth z knihoven KDE a je proveden pokus o integraci KAuth do KAuthorized. Jsou popsány problémy s touto integrací vzájemně nekompatibilních rozhraní. Během integrace je objeven lokální útok typu Denial of Service na autorizační systém PolicyKit. Integrace není úspěšná. Zbytek práce se zabývá nástrojem KioskTool a návrhem nového uživatelského rozhraní pro něj.

Abstract

This work describes the KAuthorized and KAuth interfaces of the KDE libraries and integration of KAuth into KAuthorized is attempted. Issues with this integration of mutually incompatible interfaces are described. A local Denial of Service attack on the PolicyKit authorization system is discovered during the integration. The integration is not successful. Rest of the work describes the KioskTool utility and proposes a new user interface for it.

Klíčová slova

KDE, Kiosk, KAuth, KAuthorized, KioskTool, autorizace, autentizace, PolicyKit, bezpečnost, lokální odmítnutí služby

Keywords

KDE, Kiosk, KAuth, KAuthorized, KioskTool, authorization, authentication, PolicyKit, security, local Denial of Service

Citace

Petr Mrázek: Rodičovský mód v KDE, bakalářská práce, Brno, FIT VUT v Brně, 2010

Rodičovský mód v KDE

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jozefa Mlícha. Další informace mi poskytli Jaroslav Řezník, Dario Freddi a Radek Nováček. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Mrázek
23. července 2010

Poděkování

Děkuji autorům nástroje KDevelop4, protože bez něj bych se v kódu prostředí KDE tak rychle nezorientoval.

Obsah

Kapitola 1

Úvod

V moderních operačních systémech určených pro desktop (pracovní stanice, notebooky, etc.) se oproti předchozím verzím změnilo mnohé. Jednou takovou změnou je změna bezpečnostní politiky.

MS Windows postupně získal mnohem lepší zabezpečení ve verzích NT, kdy došlo k přechodu na nové jádro, a Vista, kde se poprvé objevila možnost jednoduše povolit obyčejnému uživateli přístup k nastavení systému pomocí UAC[1]. Zamezilo se tak nutnosti „být administrátorem“, což bylo do té doby výchozí nastavení po intalaci. Ve výsledku má uživatel povoleny akce, ke kterým by jinak přístup neměl.

Unixové a unixu podobné systémy se vyvíjely směrem opačným – od práv pevně svázaných s jeho účtem a skupinou, k mnohem volnějšímu pojetí. Za zmínku stojí například program sudo, který umožňuje uživateli spouštět programy s jinými právy než jsou jeho vlastní (převážně superuživatelská práva) případně program su, který je ještě staršího data.

Nově je k dispozici také PolicyKit, který alespoň z pohledu uživatele plní podobnou úlohu jako UAC na OS Windows. Aplikace využívající PolicyKit nemusejí mít pro změnu systémového nastavení jako je třeba datum a čas superuživatelská práva.

Prostředí KDE (po přeznačení KDE Software Compilation) je možné provozovat na více operačních systémech. Aplikace by tak musely používat řešení jako je UAC nebo PolicyKit, což by omezilo jejich přenositelnost. Tyto řešení tedy bylo potřeba nějak obalit. Za tímto účelem vzniklo rozhraní KAuth.

V rámci KDE však bylo možné již dříve měnit práva uživatelů – skrývat položky menu, zamezit použití terminálu a podobně. K tomu slouží Kiosk, což je rozhraní nad konfiguračním systémem KDE KConfig.

Společně s nutnými základy bezpečnosti v systému linux jsou tyto technologie a způsob jakým budou v práci využity blíže popsány ve druhé kapitole. Třetí kapitola popisuje integraci systému KAuth do rozhraní KAuthorized a implementaci nástroje pro migraci dat. Čtvrtá kapitola obsahuje popis implementace testů integrace a jejich výsledky. Pátá kapitola se zabývá portem nástroje KioskTool do KDE4 a úpravami v něm. Šestá kapitola práci uzavírá a navrhuje další možný postup.

Předpokládá se alespoň uživatelská znalost operačního systému Linux a grafického prostředí KDE.

[1] User Access Control

1.1 Co je to KDE Software Compilation?

Původně byl všechen software KDE součástí jedné velké kolekce základních utilit pod názvem „K Desktop Environment“. Protože byly všechny programy a knihovny součástí této jedné kolekce, stačilo ji vydávat pod jednotným názvem synchronizovaně. Jak tým KDE a počet aplikací rostl, mnoho z těchto programů začalo využívat své vlastní vývojové cykly a oddělily se od původní kolekce.

KDE Software Compilation (dále KDE SC) je množina knihoven a programů vytvářených komunitou KDE, které stále používají původní synchronizovaný vývojový cyklus a vytvářejí tak základ pro uživatelsky přívětivé grafické prostředí dostupné pro operační systémy Linux, Windows a Mac OSX.

KDE SC sestává z knihoven KDE-Libs, které tvoří podklad pro všechny programy KDE a mnoha balíků softwaru. KDEBase je balík základních programů KDE a obsahuje mimo jiné i plochu Plasma a prohlížeč souboroveho systému Dolphin (má podobný účel jako program Explorer ve Windows). Součástí KDE je mnoho dalších balíků softwaru.

Zvláštní zmínku si zaslouží repozitáře extragear a kdereview. Extragear obsahuje programy a knihovny nesynchronizované s vývojovým cyklem KDE SC. Kdereview pak software, který je zvažován pro začlenění do některého z balíků KDE SC.

kdereview: http://websvn.kde.org/trunk/kdereview/ extragear: http://extragear.kde.org/

CITACE: http://www.kde.org/community/whatiskde/softwarecompilation.php

1.2 Cíle práce

Práce se z velké části zabývá knihovnami v balíku KDE-Libs a autorizačními rozhraními v nich obsaženými – Kiosk a KAuth. Kiosk je starší rozhraní, postavené nad konfiguračním rozhraním KConfig. KAuth je novějšího data. Obě rozhraní slouží k autorizaci akcí. Akcí se zde rozumí nějaká konkrétní akce proveditelná uživatelem v rámci aplikace, kterou může administrátor zakázat nebo povolit. KAuth je narozdíl od Kiosku schopen také tyto akce provádět spouštěním pomocných programů s jinými právy, než má jeho uživatel. Kiosk má některá další specifika, například zdroje dat nebo povolování přístupu k adresám URL. Zdroji dat rozumíme určitý typ souborů používaný v aplikacích KDE. Například omezením pozadí na plochu (alespoň teoreticky) zamezíme tomu, aby uživatel mohl měnit pozadí plochy na jiné, než instalované v systému.

Hlavním cílem práce je integrovat KAuth do Kiosku, aby mohl být používán pro ukládání povolení akcí a přístupu ke zdrojům dat.

Další část práce se bude soustředit na dokončení portu (přenesení) programu KioskTool z prostředí KDE3 do prostředí KDE SC 4. Nedokončený port je obsažen v repozitáři extragear. ODKAZ:

Kapitola 2

Bezpečnost v systému linux

Zde se jedná zejména o bezpečnost interní – tedy řízení přístupu ısubjektů k ıprostředkům systému. Subjektem mohou být například procesy, uživatelé nebo skupiny uživatelů. Prostředky pak mohou být soubory a adresáře, zařízení (speciální typ souborů), čísla portů, nebo zmíněné akce autorizačních rozhraní Kiosk a KAuth.

Uživatelé se do systému přihlašují, přičemž dochází k ověření jejich identity – autentizaci. V systému Linux lze řešit přihlašování pomocí mechanismu PAM[1]. Ten umožňuje psát programy vyžadující autentizaci nazávisle na způsobu jakým je jí dosaženo. Nezáleží pak, jestli se uživatel přihlašuje heslem nebo například otiskem prstu. Další způsoby autentizace je možné do systému doplnit pomocí zásuvných modulů. Mechanismus však nutné použít – distribuce Slackware jej ve výchozím stavu neobsahuje. Podrobnější popis lze nalézt například v článku, který vyšel na Root.cz[1]. Pokud je to vyžadováno, KAuth využívá mechanismu PAM k ověření totožnosti uživatele.

Dále bude zmíněn standardní způsob řízení přístupu k souborům a adresářům. Jedná se pouze o nejzákladnější poznatky.

2.1 Souborový systém a řízení přístupu k souborům

Unixu podobné operační systémy mají tzv. virtuální systém souborů, kde jsou všechny soubory na všech připojených souborových systémech umístěny v jedné hierarchii. Tato hierarchie souborů a adresářů má jeden kořenový adresář „/“. Adresáře obsahují soubory a platí, že adresář je speciálním typem souboru (existují také další speciální soubory pro zařízení). Adresáře dále obsahují dvě speciální položky: „..“ odkazující na adresář o úroveň výše a „.“ odkazující na adresář samotný (identita).

V hierarchii mohou být dále symbolické a pevné odkazy. Symbolické odkazy mohou odkazovat jak na soubory, tak na adresáře. Při smazání odkazovaného souboru nebo adresáře symbolický odkaz zůstáva. Pevné odkazy pak mohou být pouze na soubory. Takto odkazovaný soubor je smazán společně s posledním pevným odkazem na něj.

Obrázek - jak vypadá souborový systém s odkazy.

Ke každému souboru a adresáři jsou přiřazena metadata, která popisují jejich další vlastnosti. Z hlediska řízení přístupu je podstatný uživatel a skupina vlastnící soubor, práva pro čtení (r), zápis (w) a spuštění (x) uživatelem, skupinou a ostatními subjekty a tzv. setuid a

[1] Pluggable Authentication Modules

setgid bity. U obyčejných souborů označuje právo pro spuštění to, že je to spustitelný program nebo skript. U adresářů spustitelnost označuje fakt, že je možné vypsat jejich obsah. CITACE: http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf

Setuid bit určuje, že spuštěný program má běžet pod účtem vlastníka souboru. Setgid znamená to stejné pro skupiny. Bez nastavených setuid a setgid bitů je program spuštěn pod účtem uživatele, který jej spouští. CITACE: http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.pdf

Obrázek - vysvětlení zápisu -rwxrwxrwx ?

Cesta ve virtuálním souborovém systému je řetězcem po sobě jdoucích názvů adresářů oddělených znakem „/“. Rozlišujeme cestu absolutní, začínající v kořenovém adresáři a relativní, která vede obecně mezi dvěma místy v hierarchii. Příklady cest:

Doplnit

2.2 Procesy a přístup k jejich prostředkům

TODO: Něco o tom, jak UID a GID funguji u procesu, jaky to ma vliv na pristup uzivatelum k pameti procesu, debugging a podobne.

2.3 Kiosk

Zde je čerpáno převážně z úvodu do Kiosku [4] a zdrojového kódu kdelibs CITACE!.

Z pohledu administrátora Kiosk nabízí možnost upravit si KDE pro své vlastní účely. Například schovat některé položky v grafickém rozhraní aplikací, znemožnit uživatelům měnit nastavení a data aplikací nebo zamezit spuštění terminálu. Z pohledu programátora je to jednoduché rozhraní umožňující autorizaci akcí.

Kiosk je vlastně souhrnný pojem pro určité vlastnosti konfiguračního systému KConfig, jeho způsob načítání konfigurace z tzv. Kiosk profilů, rozhraní KAuthorized a specifikace, jakým způsobem jsou autorizace akcí uloženy v KConfigu. Nelze přesně určit jedno konkrétní místo ve zdrojovém kódu KDE, kde by bylo popsáno jakým způsobem funguje, ale podrobným zkoumáním lze vysledovat, jak je docíleno výsledného efektu.

Obrázek: architektura Kiosk, Kconfig

KConfig

Kconfig je v Kioksu použit pro uložení informace o tom, ke kterým akcím a jiným zdrojům má uživatel povoleno přistupovat. Konfigurace je uložena v souborech nápadně připomínajících .INI soubory se kterými se dá pomněrně jednoduše pracovat.

Na výpisu 2.1 vidět, jak je soubor členěn. Obsahuje skupiny a záznamy typu klíč-hodnota. Soubory se během načítání konfigurace slučují do jednoho konfiguračního celku (třída KConfig), podle pořadí ve kterém jsou zpracovány. Jednotlivé skupiny z načítaných souborů se slučují do objektů typu KConfigGroup – lze je získat od KConfig objektu jako podle jejich názvu. Pro pořadí zpracování konfiguračních souborů obecně platí, že globální systémová konfigurace je zpracována jako první.

FIXME: kde jsou defaultní zdroje konfigurace, jaké je defaultní pořadí bed Kiosku

```
1 [$i]
  [Colors]
3 CurrentPalette[$i]=Forty Colors

5 [ColumnMode]
  FontWeight=50
7 PreviewSize=176

9 [DetailsMode]
  FontWeight=50
11
  [General][$i]
13 AutoExpandFolders=true
```

Výpis 2.1: Ukázka konfiguračního souboru KConfig

Celé soubory, skupiny jako je `General` a jednotlivé záznamy jako je např. `AutoExpandFolders` je možné nastavit jako nezměnitelné – „immutable“ pomocí přidání značky [$i]. Jakmile je jednou nastavena na skupinu, soubor nebo hodnotu nezměnitelnost, jsou další objekty stejného typu při načítání ignorovány. V uvedeném příkladu 2.1 je nastavena nezměnitelnost pro klíč `CurrentPalette`, skupinu `General` a celý soubor (první řádek). Máme-li tedy například nastavení programu Akregator (čtečka RSS) z příkladu 2.1 umístěno v některém adresáři s globálním nastavením, budou položky, které jsou ve výsledku nezměnitelné uživatelům vnuceny. Uživatel je pak sice stále může měnit ve svých vlastních konfiguračních souborech ve svém domovském adresáři, ale nezměnitelnost se postará o to, že budou takovéto změny ignorovány. Když nejsou položky v globálním konfiguračním souboru nastaveny jako nezměnitelné, znamená to, že jsou chápány jako výchozí nastavení.

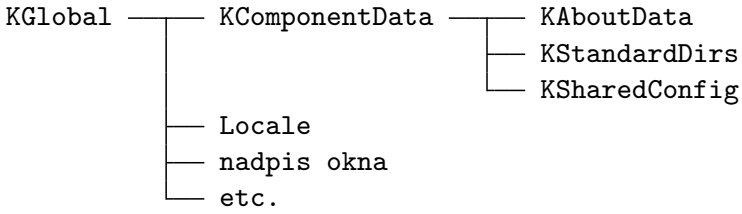
Je zřejmé, že řešení s pouze jedním umístěním pro globální konfiguraci není ideální. Konfigurace se pak totiž vztahuje na všechny uživatele systému. Kiosk proto zavádí tzv. Kiosk profily. Ty jsou pak přiřazeny uživatelům a skupinám. Dále je popsáno načítání Kiosk profilů podrobněji.

Načítání konfigurace během spuštění aplikace

Každá KDE aplikace má několik základních částí, obsažených ve jmenném prostoru `KGlobal` 2.1.

REF: http://api.kde.org/4.x-api/kdelibs/kdecore/namespaceKGlobal.html

TODO: nahradit 2.1!



Obrázek 2.1: Struktura KGlobal

`KGlobal` je jmenný prostor obalující přístup k dalším komponentám a sdíleným zdrojům

v rámci aplikace.

`KComponentData` obsahuje informace relevantní pro jednu komponentu aplikace. Platí, že aplikace může mít jednu hlavní komponentu.

`KAboutData` je soubor základních informace o programu – v případě hlavní komponenty jsou tyto informace použity také pro dialog „O Aplikaci“ (About). Určuje název komponenty. Ten se použije pro načítání konfigurace a pro výběr složky s daty aplikace. Je nutno poznamenat, že aplikace může mít více než jednu komponentu.

`KSharedConfig` obsahuje sloučenou konfiguraci efektivní pro program a zajišťuje, že konfigurace je sdílena mezi komponentami – šetří se tak pamětí a časem nutným k načtení konfigurace.

`KStandardDirs` slouží k určení, které složky budou použity jako zdroj dat a konfigurace pro aplikaci.

Právě `KStandardDirs` a `KSharedConfig` jsou relevantní pro popis načítání konfigurace. Během inicializace komponenty se nejdříve načítá obecná konfigurace. Ta obsahuje nastavení pro celé KDE a zmíněnou aplikaci, ovšem bez začleněných Kiosk profilů.

Potom je volána metoda `KStandardDirs::addCustomized`, která vyhledá Kiosk profily platné pro uživatele a zařadí je mezi zdroje konfigurace s prioritou . Na *nix systémech je soubor s jejich seznamem odkazován z „/etc/kde4rc“. Na rozdíl od cest ke zdrojům určených v konstruktoru `KStandarDirs` jsou umístěny na začátek seznamu složek s konfiguračními zdroji.

Platí, že pokud má uživatel přiřazen svůj vlastní Kiosk profil, nezpracovávají se dále Kiosk profily pro skupiny v nichž je členem. V opačném případě, kdy uživatel nemá vlastní profil, přidají se s prioritou do cest v KStandardDirs všechny aplikovatelné skupinové profily. Pokud se počet cest s konfigurací změnil, je po návratu z addCustomized konfigurace znovu načtena.

Rozhraní KAuthorized

!!!!DOPLNIT NĚJAKÝ PĚKNÝ DIAGRAM, který to trochu oživí.

KAuthorized je rozhraní postavené nad konfiguračním systémem KConfig a využívá všech jeho vlastností. Poskytuje KDE knihovnám a aplikacím možnost autorizace obecných akcí, KAction akcí, konfiguračních modulů KControl, a přistupů k URL adresám. Chybí zde ovšem omezení přístupu ke zdrojům dat a konfigurace.

Akce KAction (třída `KAction` odvozená od `QAction` z knihovny Qt) a obecné akce jsou v rámci KDE činnosti, které může uživatel vykonat v aplikaci. Tyto akce mají název a ten je v Kiosku použit pro jejich autorizaci. Jsou často asociovány s nějakým ovládacím prvkem grafického rozhraní aplikace.

Restrikce akcí jsou nastavovány v konfiguračních souborech ve skupině `KDE Action Restrictions`. Zpravidla se jich používá ke schování nebo vypnutí s nimi provázaných prvků rozhraní. Akce může být například otevření menu s nápovědou, změna pozadí na ploše nebo vypnutí počítače z menu KDE. Pokud je taková akce zakázána pomocí Kiosku, prvky uživatelského rozhraní se nezobrazí (případně nevytvoří během inicializace aplikace).

Rozdíl mezi obecnou akcí a akcí KAction je v rámci `KAuthorized` mizivý a funkce `AuthorizeKAction` je obal nad `authorize`, který před název akce přidá prefix „action/“. Příslušná funkce pak použije globální `KConfig` objekt aplikace k ověření, zda má uživatel tuto akci zakázánu (ve výchozím stavu jsou všechny povoleny).

Omezení přístupu ke zdrojům je nastavováno ve skupině `KDE Resource Restrictions` a je umístěno do globálního konfiguračního souboru jako je `kdeglobals`. Omezením zdrojů lze docílit toho, že aplikace „neuvidí“ zdroje umístěné v uživatelově domovské složce. Lze tak zamezit například tomu, aby si uživatel aplikace rozšiřoval. I když je tato část Kiosku velmi podobná ostatním autorizačním funkcím v `KAuthorized`, není do něj přímo začleněna. Důvodem je, že je úzce provázána s třídou `KStandardDirs`, která tyto omezení zpracovává a používá pro vytvoření seznamů složek se zdroji. Seznam těchto omezení je potřeba znát ještě před tím, než je zcela načtena konfigurace z prostého důvodu – je možné omezit i zdroj pro konfiguraci. Zde bude nutné poupravit způsob načítání konfigurace tak, aby bylo možné umístit omezení na zdroje dat i do Kiosk profilů.

Omezení přístupu k URL je nastavováno ve skupině `KDE URL Restrictions`. Tyto omezení jsou určeny pro zamítnutí přístupu k některým adresám URL. Dalším typem omezení jsou tzv. `KDE Control Module Restrictions`, které se používají ke skrytí nebo zamezení otevření konfiguračních dialogů. KDE používá třídu `KCModule` jako základ pro všechny moduly použité v aplikacích „Nastavení Systému“, `kcmshell` a případně v jiných aplikacích, které používají tyto moduly pro úpravu svého nastavení. Omezení se vztahují k názvům těchto modulů – pokud budou moduly přejmenovány, přestanou omezení fungovat.

2.4 KAuth

KAuth je nové rozhraní pro autorizaci v KDE. Je postaveno na již existujících rozhraních v operačních systémech. V OS na bázi Linuxu je to zpravidla PolicyKit, v OSX pak framework Authorization Services. Podporu pro nová rozhraní je možné přidat připsáním nových zásuvných modulů.

Pomocí KAuthu je možné autorizovat akce uživatele a také je provádět. Ace jsou prováděny za použití pomocného programu, který je spuštěn příslušným autorizačním rozhraním přítomným v systému. Takovýto program se nazývá KAuth pomocník (helper). Pomocník a aplikace která ho využívá mohou do jisté míry komunikovat oběma směry a je také možné sledovat průběh dlouho trvající akce uvnitř pomocníka. KAuth pomocník je rozšířením PolicyKit pomocníka o tyto zjednodušené komunikační funkce. Hlavní výhodou je, že za použití pomocníka lze provádět privilegované akce bez nutnosti být přihlášen pod administrátorským účtem nebo tohoto účtu používat ke spuštění aplikace jako celku.

Další službou KAuthu je registrace pomocníků a akcí v jeho jednotlivých zásuvných modulech. Uživatel KAuthu tedy specifikuje, jaké akce a pomocníky chce použít a při kompilaci budou tyto specifikace převedeny do formy srozumitelné pro jeho zásuvné moduly - např. PolicyKit nebo OSX Authorization Services.

Díky tomu jakým způsobem je KAuth navržen nepodporuje provádění změn v seznamu platných akcí po kompilaci. Také neumožňuje měnit autorizované uživatele a skupiny pro akce – to je zcela přenecháno použitému autorizačnímu systému. Dále pak platí omezení na názvy KAuth akcí, které mohou obsahovat jen malá písmena anglické abecedy, číslice a tečku jako oddělovač.

Formát specifikace akcí

Formát specifikace akcí je podobný jako formát konfiguračních souborů KConfig. Obsahuje skupiny a záznamy klíč-hodnota v těchto skupinách. Uvedený příklad .actions souboru 2.2 byl použit k vygenerování podobného souboru pro PolicyKit, který bude uveden dále: 2.3.

```
[Domain]
Name=Date and Time Control Module
Icon=preferences-system-time

[org.kde.kcontrol.kcmclock.save]
Name=Save the date/time settings
Description=System policies prevent you from saving the date/time settings.
Policy=auth_admin
Persistence=session
```

Výpis 2.2: Ukázka KAuth .actions souboru

Soubor obsahuje skupinu Domain, která popisuje ke které aplikaci patří (V tomto případě KControl modul pro nastavení data a času) a jakou má mít ikonu. Textové položky mohou být lokalizovány (zde vynecháno).

Po skupině Domain následují definované akce. Platí, že název akce je také názvem skupiny v souboru. Název akce se skládá ze dvou částí – jmenného prostoru a názvu akce v něm. Zde je jmenným prostorem část `org.kde.kcontrol.kcmclock`. Název akce je `save`. Skupina dále obsahuje srozumitelný název (`Name`), popis (`Description`) a výchozí chování při pokusu o autorizaci.

Položka Policy určuje průběh autorizace. V případě, že je nastavena na hodnotu `yes`, je autorizace okamžitá. `no` naopak znamená, že akce nemůže být autorizována. Pokud je nastavena na `auth_self`, bude akce autorizována pokud se uživatel přihlásí pod svou vlastní identitou – ověří se tak, že je to opravdu on. Konečně `auth_admin` znamená, že akce bude autorizována pokud se uživatel přihlásí jako administrátor.

Poslední položkou skupiny je `Persistence`. Ta je nepovinná a udává, na jak dlouho bude autorizace udělena. Možnostmi jsou `session`, kdy bude platit dokud se neodhlásí a `always`, kdy nebude omezena.

Specifikace akcí v .actions souborech je také dále omezená tím, že všechny názvy akcí v jednom souboru musejí mít společný základ (jmenný prostor).

Rozhraní KAuth

text, text, text, class diagram? odkaz na dokumentaci KAuth namespace.

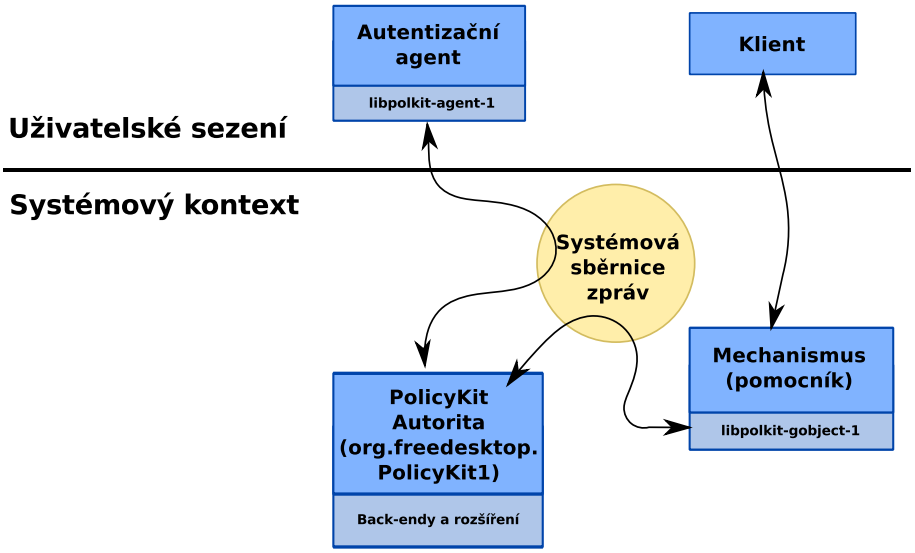
2.5 PolicyKit

V této sekci jsem čerpal z manuálu [8], převážně pak z jeho částí [10] a [9].

PolicyKit poskytuje autorizační rozhraní privilegovaným programům (mechanismy), které nabízejí služby neprivilegovaným programům (klienti) za použití IPC[2] mechanismu jako je D-Bus. Mechanismus považuje požadavky od klienta za nevěrohodné a pro každý požadavek musí mechanismus zjistit, jestli je autorizován nebo jestli má klientovi službu odepřít. Pomocí PolicyKit API může mechanismus tuto část přenechat důvěryhodné třetí straně: PolicyKit autoritě.[10]

Vedle funkce jako autorita umožňuje PolicyKit uživatelům také získat dočasnou autorizaci pomocí autentizace jakožto administrátor nebo vlastník sezení do kterého klientský

[2] Inter-process communication



Obrázek 2.2: Architektura systému PolicyKit, přeloženo z [10]

program patří. To je výhodné v situaci, kfy je nutné ověřit totožnost osoby používající klientský program.[10]

Tedy podobně jako KAuth, PolicyKit umožňuje autorizovat akce uživatelů a pomocí tzv. mechanismů tyto akce provádět. Jádrem PolicyKitu je několik základních komponent. První důležitou částí je démon polkitd, který funguje jako centrální prvek PolicyKitu. Implementuje část PolicyKit Authority, která se stará o čtení databáze možných akcí a povolení nebo zamítnutí požadovaných akcí.

Další částí, implementovanou prostředími jako je Gnome nebo KDE je tzv. autentizační agent. To je program, který například uživateli zobrazí okno pro zadání hesla, pokud je potřeba ověřit jeho totožnost. Pro samotné ověření je možné použít pomocný program běžící se superuživatelskými právy a používající k ověření systém jako je PAM – ten je dobře popsán v článku [1]. To ale není striktně vyžadováno. Lze i jen zobrazit obyčejné okno s dotazem typu Ano/Ne.

Třetí komponentou je „mechanismus“ nebo „pomocník“. zde se jedná o službu pro vykonání privilegovaných akcí místo uživatele, který o provedení akce žádá. Čtvrtou a poslední komponentou je klient. To je program, který využívá služeb systému PolicyKit.

Průběh akcí v systému pro volání funkce pomocníka může vypadat například takto:

- Uživatel se přihlásí, je nastartováno sezení KDE a s ním i autentizační agent
- Autentizační agent se registruje u PolicyKit autority
- Uživatel spustí program, který využívá služeb PolicyKitu
- Program zavolá funkci pomocníka přes systém D-Bus
- DBUS démon nastartuje program pomocníka, pokud již neběží (pomocník musí být v systému D-Bus registrován)
- Spuštěný pomocník se zeptá PolicyKit autority, jestli je volaná akce autorizována

- PolicyKit autorita zkontroluje, jestli je již tato akce autorizována (autorizace může mít například platnost po délku celého sezení). Pokud není zatím autorizována, zjistí jakým způsobem se má autorizace dosáhnout.
- Pokud je to nutné, zavolá PK autorita autentizačního agenta. Je možné ověřovat buď pouze identitu uživatele, kdy stačí, že se přihlásí pod vlastním účtem, nebo je možné vyžadovat přihlášení účtu se superuživatelskými právy.
- Uživatel se přihlásí. Ve většině distribucí Linuxu za pomoci mechanismu PAM.
- Agent oznámí autoritě jestli byla autentizace úspěšná.
- Autorita vrací výsledek autorizace a uloží si ho do vyrovnávací paměti po dobu její platnosti.
- Pokud byl proces autorizace úspěšný, pomocník provede požadovanou akci.
- V tomto bodě může pomocník oznámit programu výsledek akce.

Toto samozřejmě není jediný možný průběh.

Nad PolicyKitem je postavena knihovna polkit-qt, která poskytuje v zásadě stejné funkce jako PolicyKit a lépe je integruje do prostředí knihoven Qt. Polkit-kde je nadstavbou nad knihovnou polkit-qt, která implementuje autentizační agent pro prostředí KDE.

Zatím jediným způsobem uložení povolených akcí v PolicyKitu je tzv. lokální autorita (PolicyKit Local Authority). Je to výchozí implementace PolicyKit Autority a využívá lokálně uložených textových souborů s příponou .policy a .pkla. Nejdříve příklad 2.3 .policy souboru. Tyto soubory používají formát XML a vymezují seznam známých akcí. Instalují se do systému jako součást balíčků.

Značka `vendor` určuje k čemu tento soubor akcí patří. V tomto případě je to KControl modul pro nastavení data a času. Soubor má dále nastavenu ikonu a samotný seznam akcí. Akce má popis (description), zprávu pro případ neúspěšné autorizace (message) a výchozí nastavení autorizace. PolicyKit rozlišuje mezi tzv. aktivním a pasivním sezením. Aktivní je například normálně přihlášený uživatel s vlastním X serverem. Pasivní může být sezení spuštěné dodatečně z login terminálu (su - username). Normálně může být aktivní pouze jedno sezení (to s kterým uživatel zrovna přímo pracuje). Dalším povoleným typem atributu je tzv. anotace. Anotace umožňuje přidat k akci další atributy. Soubor povoluje měnit nastavení času pouze uživateli s aktivním sezením, který se prokáže jako superuživatel.

Takto vymezené akce je dále možné pozměnit v souborech .pkla. Efektivní seznam povolených akcí se získá sloučením záznamů z těchto souborů. Je tedy možné mít například jeden soubor s nastavením instalovaný z distribučního balíčku a druhý s vyšší prioritou vytvořený administrátorem.

Local Authority pro ně zavádí poměrně propracovanou strukturu složek ve dvou umístěních (/var/lib/polkit-1/localauthority a /etc/polkit-1/localauthority). Pořadí načítání je určeno lexikografickým řazením složek a souborů v nich. Pokud je stejně pojmenovaný soubor v obou umístěních, nejdříve se zpracuje ten ve /var/. Detaily a příklad viz. [9].

Na výpisu 2.4 je ukázka .pkla souboru. Uživatelé petr a pavel jsou členy skupiny zamestnanci.

Záznamy se zpracovávají tak jak jdou za sebou a platí, že zpracování pokračuje i po úspěšném porovnání uživatelského jména/skupiny s těmi od žadatele – prochází se vždy

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE policyconfig PUBLIC
"-//freedesktop//DTD PolicyKit Policy Configuration 1.0//EN"
"http://www.freedesktop.org/standards/PolicyKit/1.0/policyconfig.dtd">
<policyconfig>
<vendor>Date and Time Control Module</vendor>
<icon_name>preferences-system-time</icon_name>
   <action id="org.kde.kcontrol.kcmclock.save" >
      <description>Save the date/time settings</description>
      <message>System policies prevent you from saving the date/time
          settings.</message>
      <defaults>
         <allow_inactive>no</allow_inactive>
         <allow_active>auth_admin</allow_active>
      </defaults>
   </action>
</policyconfig>
```

Výpis 2.3: Ukázka souboru s definicí akce (.policy soubor), soubor byl generován KAuthem ze souboru 2.2

```
[Povolene akce pro zamestnance]
Identity=unix-group:zamestnanci
Action=com.example.uzasnyprodukt.*
ResultAny=no
ResultInactive=no
ResultActive=yes

[Zakazy pro par cernych ovci]
Identity=unix-user:petr;unix-user:pavel
Action=com.example.uzasnyprodukt.*
ResultAny=no
ResultInactive=no
ResultActive=auth_admin
```

Výpis 2.4: Ukázka souboru s nastavením PolicyKit Local Authority, Přeloženo z [9]

všechny záznamy. Jako příklad si vezmu uživatele petr, který požádá o autorizaci akce `com.example.uzasnyprodukt.uzasnaakce`. Zaměstnanci mají tuto akci obecně povolenu bez nutnosti se přihlašovat v první skupině souboru. Zpracování však pokračuje a uživatel petr se bude muset pro úspěšnou autorizaci akce přihlásit jako administrátor.

Rozdíl mezi položkami `ResultAny`, `ResultInactive` a `ResultActive` se nemusí zdát zjevný. `ResultActive` a `ResultInactive` jsou výsledky akce pro aktivní a neaktivní sezení – podobně jako značka `allow_active` a `allow_inactive` v .policy souborech. `ResultAny` pak určuje výsledek pro oba typy sezení. Alespoň jedna z těchto položek musí být přítomna. Hodnota položky udává jakým způsobem bude uživatel žádající o autorizaci akce autentizován.

Kapitola 3

Integrace KAuth do KAuthorized

3.1 Analýza problému

První praktickou částí projektu je zjistit jak nejlépe integrovat systém KAuth do rozhraní KAuthorized. Díky celkové rozdílnosti KAuthu a Kiosku bude nejspíše potřeba určitých kompromisů.

Na jedné straně Kiosk, podporující profily přiřazené uživatelům a skupinám a změnu těchto profilů (stačí být administrátor a upravovat profily standardními nástroji). Na druhé straně KAuth, primárně určený k odstranění nutnosti spouštět programy pro systémovou konfiguraci s grafickým uživatelským rozhraním jako administrátor.

Varianta s přímou integrací

První a nejjednodušší varianta je vytvořit statický .actions soubor (poté přeložený KAuthem) v kombinaci s integrací KAuth přímo do rozhraní KAuthorized. KAuth by tak konvertoval tento statický soubor do formy srozumitelné pro PolicyKit během sestavení KDE-Libs.

Problémem je zde to, že neexistuje definitivní seznam možných restrikcí akcí a zdrojů v Kiosk profilech. Toto by šlo řešit tak, že každý program KDE by tyto své akce a zdroje specifikoval v .actions souboru k tomu určeném a zahrnul by jeho překlad do svého sestavení. To ovšem vylučuje jakoukoliv možnost stejné úrovně podpory pro aplikace, které by takto akce nespecifikovaly.

Druhým problémem je neexistence možnosti nastavit autorizované akce pro uživatele a skupiny pomocí KAuthu. Nejen že nejdou nastavit, KAuth navíc nemá ani žádné ponětí o něčem, co by alespoň vzdáleně připomínalo způsob jakým se aplikují Kiosk profily. V případě použití PolicyKitu tyto vcelku základní funkce nemá ve formě knihovny žádná vrstva - PolicyKit, polkit-qt, polkit-kde ani KAuth. Musí se přímo přistupovat ke konfiguračním souborům PolicyKitu.

Tato varianta vyžaduje implementaci nástroje pro překlad Kiosk profilů na nastavení KAuth/PolicyKit. Zde je nutno přihlédout k tomu, že názvy akcí v HAuthu jsou daleko více omezené než ty v Kiosku. Všechny akce v jednom .actions souboru musejí mít společný jmenný prostor a v PolicyKitu soubor musí být podle tohoto jmenného prostoru pojmenován. Navíc mohou názvy obsahovat pouze malá písmena latinky, číslice a tečku jako

oddělovač. Názvy akcí a zdrojů v Kiosku tato omezení samozřejmě nemají a často využívají jiné znaky jako je například pomlčka, podtržítko nebo lomítko. Tyto znaky je potřeba buď odstranit nebo nahradit. Vzniká tak reálná možnost kolize názvů.

Varianta s KConfig objektem v KAuth pomocníkovi

Druhá možná varianta je jednoduše se vzdát mapování akcí mezi Kioskem a KAuthem jedna ku jedné. Mělo by být možné umístit do KAuth pomocníka obsahujícího instanci KConfigu, která načte požadované Kiosk profily a bude přes KAuth/D-Bus umožňovat dotazování se na jednotlivé hodnoty v nich. Bylo by také možné tyto hodnoty měnit. V podstatě by to byl stále pouze KConfig, jen obalen v pomocníkovi. I toto řešení by však mělo problémy.

Za prvé – nezíská se tím vůbec žádná výhoda z pohledu bezpečnosti. Kiosk profily musejí stále zůstat čitelné pro všechny uživatele, jinak by se při spuštění programů nemohly načíst a části které nebudou takto integrovány v pomocníkovi (vše s výjimkou omezení akcí a zdrojů) by tím byly zcela neefektivní. Odpadá tak výhoda KAuthu, kde jak seznam omezení, tak jaká omezení pro koho platí může být před uživateli skryt. Při použití PolicyKit Local Authority je možné nastavit všechny soubory čitelné jenom superuživatelem (dokonce je to výchozí stav).

Za druhé – KAuth pomocníci mají omezenou životnost. Pokud nejsou využíváni po dobu deseti vteřin, jsou ukončeni (viz. zdrojový kód KAuth ADD REF!!). Znamená to znovu načíst celý KConfig objekt. I když je toto načítání vysoce optimalizováno, stále je vcelku zbytečné načítat celou konfiguraci dvakrát. Jednou v programu, který by volal KAuthorized a podruhé v KAuth pomocníkovi. Sdílet zde jeden konfigurační objekt by bylo zbytečně komplikované.

Jedna z vlastností KAuthu, kterou by se také nedalo využít jsou srozumitelné názvy akcí v systému. Místo `org.kde.kiosk.action.help` a mnoha dalších by byly v systému jen akce pomocníka. V tom případě se ovšem nedá mluvit o integraci KAuthu a KAuthorized.

V této variantě by nebylo potřeba implementovat nástroj pro překlad Kiosk profilů na nastavení KAuth/PolicyKit.

Další požadavky na řešení

Použití KAuthu by nemělo být povinné a mělo by být zachováno chování rozhraní `KAuthorized` pokud možno tak, aby se z pohledu aplikací nic nezměnilo. Zde je problém to, že administrátor může, ale také nemusí nastavit v systému `KConfig` kteroukoliv položku jako nezměnitelnou. Pokud bude kontrola oprávnění pomocí KAuthu před kontrolou pomocí KConfigu, je nutné zaručit, že omezení známá v KAuthu mohou být brána jako změnitelná, tak aby mohly být dále modifikována pomocí nastavení v KConfigu. KAuth ani PolicyKit toto neumožňují. Musí se také počítat s tím, že KAuth nemusí být přítomen v systému nebo že konvertor zatím nikdy nebyl spuštěn a v těchto případech použít normální nastavení Kiosk profilu přes `KConfig`.

3.2 Návrh řešení

Rozhodl jsem se implementovat první variantu. To znamená přímo integrovat KAuth do rozhraní KAuthorized. Samotná integrace bude vyžadovat mnoho malých změn v KDE-

Libs a implementaci nástroje pro konverzi Kiosk profilů na nastavení lokálního systému pro autorizaci. Zde se budu držet PolicyKitu a operačního systému Linux.

Konvertor bude fungovat jako KAuth pomocník a bude vytvořen samostatný program spustitelný uživatelem z příkazové řádky, který bude tohoto pomocníka volat. Konverze z Kiosk profilů bude jednosměrná, bude používat polkit-qt pro získání seznamu podporovaných omezení a bude produkovat soubory .pkla, použitelné v PolicyKit Local Authority. Konvertor by mělo být možné, až bude stabilní, integrovat do KAuthu nebo balíku policykit-kde. Konvertor nebude vyžadovat od uživatele žádný vstup.

V konvertoru je nutné zohlednit to, že postup aplikace profilů v Kiosku je jiný než postup ověření autorizace v PolicyKitu. Musí se replikovat Kiosk a jeho zvláštnosti, aby se nezměnilo chování KAuthorized. Problém s omezeními názvů akcí v KAuth v porovnání s Kioskem je nutné řešit buď záměnou znaků v názvech akcí z Kiosku, nebo překódování celých názvů do přijatelné a v KAuthu uložitelné podoby.

Integrace KAuth do KAuthorized je sice otázka několika volání funkcí KAuth, ale autorizační funkce KAuthu na rozdíl od systémů na kterých staví nevrací chybové kódy v případě, že akce neexistuje. Toto bude potřeba do KAuthu doplnit.

Omezení na zdroje `KDE Resource Restrictions` si vyžádají změny ve třídě `KStandardDirs` – bude potřeba doplnit schopnost načítat omezení zdrojů z Kiosk profilů a umístit do jmenného prostoru `KAuthorized` funkci, která bude vracet seznam typů zdrojů s nastaveným omezením.

3.3 Implementace

Změna v KAuth

Implementace spočívá v několika přesně cílených změnách v kódu KDE-Libs. V prvé řadě byla doplněna schopnost KAuthu vracet vedle informace o úspěšnosti autorizace také zvláštní hodnotu pro případ, že akce není použitému systémem pro autorizaci známa. Ve výchozím stavu vypadá funkce pro získání autorizace takto: D.1. Hodnota `Unknown` je vrácena v případě, kdy dojde k chybě během autorizace. Důvod chyby lze zjistit bližším dotazováním autorizačního rozhraní. Úprava metody pak vypadá takto: D.2.

Přesun vyhodnocení omezení na zdroje do KAuthorized

Zjišťování seznamu omezení na zdroje je umístěno v metodě `addCustomized` třídy `KStandardDirs` a tento seznam je vytvářen před tím, než se načtou Kiosk profily. Tato metoda přidává profily k cestám pro načítání konfigurace a třída `KComponentData`, která tuto metodu volá následně způsobí znovunačtení konfigurace. Z takovéto načtené konfigurace ale již nejsou vytaženy seznamy omezení zdrojů.

Rozhodl jsem zjišťování seznamu omezených zdrojů přesunout do rozhraní `KAuthorized`, aby byly změny nutné pro integraci KAuthu pokud možno pouze na jednom místě.

Dále jsem se rozhodl, že oddělím zpracování tohoto seznamu od metody `addCustomized()`. Dá se tak spustit z třídy `KComponentData` poté, co `KStandardDirs` přidá Kiosk profily mezi konfiguraci. Tím se docílí toho, že je možné uložit omezení zdrojů stejným způsobem jako omezení na akce.

Implementoval jsem tedy funkci `restrictResourceTypes` pro získání seznamu Kioskem neomezených zdrojů ve formě objektu `QStringList` a umístil ji do rozhraní KAuthorized. Dále jsem přesunul vyhodnocení těchto omezení do nové metody `evaluateRestrictedResources()` v `KStandarDirs`, která je volána po znovu-načtení konfigurace po přidání profilů. Bylo nutné také změnit konstruktor privátního objektu v KAuthorized, aby neblokoval zpracování profilů. Původně nebyl navržen na to, aby byl použit tak brzo během spouštění programů.

Specifikace akcí a zdrojů

Ke specifikaci akcí v systému KAuth jsou využity statické .actions soubory. Jako základní jmenný prostor jsem se rozhodl použít `org.kde.kiosk`. Pro akce jsem se rozhodl pro `org.kde.kiosk.action` a pro zdroje `org.kde.kiosk.resource`.

Názvy akcí a zdrojů nemohou obsahovat jiné znaky než malá písmena a číslice. Je tedy nutné přeložit jejich skutečné názvy do formy, kterou je KAuth schopen použít. Z akce `action/help` se tak stane `actionhelp`, společně se jmenným prostorem pak `org.kde.kiosk.action.actionhelp`. Kdyby však exitovala jiná akce s názvem `action_help`, došlo by ke kolizi.

Implementace konvertoru

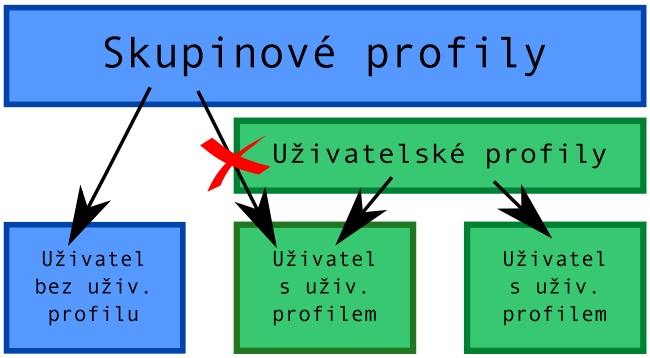
Konvertor se skládá ze dvou částí. Hlavní částí je pomocník, postavený na knihovně KAuth. Ten implementuje veškeré funkce celku (ve zdrojovém kódu je to kdelibs/kdecore/auth/-kioskpklahelper.cpp). Druhou částí je jednoduchý terminálový program, který není až na spuštění pomocníka přes KAuth příliš zajímavý (ve zdrojových kódech je to kioskpklaconvert.cpp ve stejné složce jako pomocník).

Konvertor funguje tak, že nejdříve získá ze souboru /etc/kde4rc nastavení Kiosku, pomocí něj vyhledá Kiosk profily, komu a jakým skupinám jsou přiřazeny a jaké je pořadí skupin při zpracování skupinových profilů. Je také získán seznam akcí známých v PolicyKit Local Authority. Tento seznam se filtruje do dvou skupin podle jmenných prostorů použitých pro omezení na akce a zdroje. Toto jsou základní vstupní parametry programu. Uživatel nemá možnost do procesu přímo zasáhnout – program nemá žádné uživatelské vstupy.

Další částí je vytvoření .pkla souboru ze skupinových Kiosk profilů. V pořadí zjištěném z nastavení Kiosku se z nich pomocí KConfigu načtou soubory globálního nastavení kdeglobals. Z těch jsou dále vytaženy skupiny `KDE Action Restrictions` a `KDE Resource Retrictions`. Klíče jejich položek jsou zpracovány stejně jako specifikace v .actions souborech. Upravené klíče společně s jejich hodnotami a přiřazenou skupinou pak tvoří záznamy ukládané do .pkla souboru pro skupiny. Soubor je uložen tak, aby byl zpracován dříve než soubory pro jednotlivé uživatele (jsou načítány v lexikografické pořadí podle názvu).

Dalším bodem je zpracování profilů přiřazených uživatelům. Postup je z části podobný jako u skupinových profilů, ale platí, že na uživatele s nastaveným uživatelským Kiosk profilem se skupinové profily nevztahují. Znamená to, že každý uživatel musí mít specifikovány všechny akce ve výchozím stavu (povoleno). K tomu se dále přidávají omezení z uživatelského profilu. Ve výsledku se tak vyruší vliv skupinových profilů – pro ilustraci: 3.1. Pro každého uživatele je zvlášť vytvořen .pkla soubor.

Problémem řešení je zejména nemožnost reprezentace vlastnosti nezměnitelnosti z KConfigu. Omezení jsou jednou specifikována jako hodnota ano/ne v nastavení PolicyKit Local



Obrázek 3.1: Aplikace Kiosk profilů

Authority a jsou tak z pohledu KConfigu nezměnitelná. Jedinou možností jak docílit toho, aby byla změnitelná je nezanést ji do PolicyKitu. Pak ale není možné s nimi v něm pracovat. Když je jednou akce jako `action/help` pro ukázání menu s nápovědou zanesena do KAuthu, ztrácí nad ní uživatel kontrolu a nemůže tak menu schovat. Toto je bohužel při způsobu jakým je KAuth navržen nevyhnutelné.

Integrace KAuth do KAuthorized

KAuth nabízí několik možností, jak autorizovat akci. V zásadě se jedná o různé metody třídy `KAuth::Action`, mající mírně odlišný význam. Prvním metodou je `execute()` – tato metoda je uvedena v příkladu [3] a je určena pro jednorázové blokující provedení akce. Pokud je to potřeba, uživatel je požádán o autentizaci. Metoda má podle [3] fungovat i bez přítomnosti pomocníka. `execute()` má také asynchronní verzi.

Další metodou je `authorize()`. Tato metoda je určena pro získání autorizace pro akce před tím, než by byla akce provedena. Má být používána ostatními metodami třídy `KAuth ::Action`. Metoda podobně jako `execute()` může od uživatele vyžadovat autentizaci. Třetí metodou je `status()` – tato metoda je podobná `execute()`, ale v případech, kdy by byla vyžadována autentizace pouze o této skutečnosti informuje.

Takovýto význam by alespoň tyto metody měly mít. Skutečnost je ale odlišná. Třída `KAuth::Action` používá pro autentizaci a použití pomocníků statické zásuvné moduly – to znamená, že jsou určeny při sestavení. Tyto moduly implementují lokálně dostupné autentizační mechanismy. Zde vzniká problém kvality těchto mechanismů a jejich implementace v KAuthu. Použitý PolicyKit vůbec neimplementuje metodu `authorize()`. Ta namísto výsledku vždy vrací hodnotu `Action::Authorized`. Její místo zastává metoda `status()`, která obaluje metodu `checkAuthorizationSync()` z knihoven polkit-qt-1. Toto je synchronní metoda pro získání autorizace.

Ze všech uvedených metod je tedy nejvhodnější `status()`. Po konverzi z Kiosk profilů totiž nevznikají akce pro jejichž autorizaci by bylo nutné se autentizovat.

Rozhraní KAuthorized jsem tedy doplnil o překlad původních názvů akcí z Kiosku do formy přijatelné KAuthem a volání metody status(). Pokud KAuth nezná autorizovanou akci nebo typ dat, je místo něj použit KConfig.

3.4 Testování

Téměř okamžitě po implementaci jsem narazil na první problém. Program, který používá upravené rozhraní KAuthorized totiž po čase přestane reagovat, společně s PolicyKit démonem. Při integraci KAuthu do rozhraní KAuthorized, které je využíváno téměř každým programem v KDE je objem dotazů o několik řádů vyšší než když je KAuth normálně používán a chyba tak vyšla na povrch. Sezení KDE přestane reagovat již během spuštění. Chyba v podstatě umožňuje přihlášenému uživateli provést DoS[1] útok na PolicyKit, protože PolicyKit démon běží jako systémová služba, společná pro všechny uživatele.

Pro zjištění kde je chyba jsem vytvořil jednoduchý test kauthDoS. Ten v nekonečném cyklu volá `KAuth::Action::status()` a v pravidelných intervalech vypisuje hlášení. Když program přestane vypisovat, znamená to, že došlo k chybě. Démon polkitd (autorita PolicyKitu) a program kauthDoS byly spuštěny v debuggeru gdb a byly získány „backtrace“ z obou programů (výpisy C.1 a C.2). Podle těch to vypadá, že problém vzniká během komunikace jednotlivých částí PolicyKitu. Chybu jsem ohlásil jeho autorovi (David Zeuthen) v mailing listu !!!!!REF!!!!! a nedočkal se odpovědi, i když je tento mailing list uveden jako místo kam se mají chyby PolicyKitu hlásit.

Provedl jsem tedy druhý pokus o nahlášení chyby a přitom implementoval test podobný KAuthDoS, ale závisející pouze na PolicyKitu. Chyba byla hlášena do bugtrackeru PolicyKitu !!!!REF!!!!, ale setkala se s nepochopením ze strany autora.

Při hledání chyby jsem narazil na druhý problém v PolicyKitu, kdy každý požadavek na autorizaci způsoboval zahození celé databáze autorizací a jeji opětovné načtení z konfiguračních souborů. Tato chyba byla opravena: !!!!!REF!!!!!

3.4.1 Testování implementace

TODO - na testech se pracuje V zasade se bude jednat o overovani, ze Kiosk s puvodnimi daty a KAuth s konvertovanymi se chovaji stejne.

3.5 Návrh dalšího postupu

Výše byly popsány vybrané části rozhraní KAuthorized, KAuth a PolicyKit. Nyní následuje návrh řešení v nich nalezených nedostatků a také návrh na jejich rozšíření. V prvé řadě je potřeba opravit zmíněnou bezpečnostní chybu v PolicyKitu.

Dále je potřeba rozšířit KAuth:

1. Je potřeba, aby uměl hlásit, pokud autorizovaná akce není známa – a to pro všechny podporované autorizační systémy. Toto jsem pro účely práce udělal a otestoval pro metodu `status()` a PolicyKit1. Ideální by bylo integrovat vyhledání akce do konstruktoru třídy `KAuth::Action` a vracet pak výsledek pomocí volání její metody `isValid()`.

2. KAuth musí umět nejen ověřovat a spouštět akce, ale také měnit oprávnění k těmto akcím pro uživatele a skupiny. Zde je zatím k dispozici pouze KControl modul pro PolicyKit v balíku policykit-kde-1 a sním spojený pomocník na bázi polkit-qt-1, ale ten je zaměřen pouze na uživatele a není nijak do KAuthu integrován.

[1] Denial of Service

3. Mělo by také být možné přidávat nové akce. To znamená, že by se nemusely do systému instalovat statické soubory s definicemi akcí.

Body 2 a 3 by bylo možné realizovat jakožto pomocníky a přidat do systému KAuth rozhraní, které by umožnilo s nimi pracovat.

Zcela korektní integraci KAuthu do KAuthorized považuji kvůli omezením ze strany KAuthu za příliš problematickou. Oba systémy jsou velmi rozdílné v některých kritických bodech. Na pouhé dotazování se na to, jestli je uživateli dovoleno provádět nějakou akci není potřeba KAuth. Stačí jakákoliv databáze schopná uložit údaje ve formě klíč-hodnota. To však nebrání využití KAuthu tam, kde použití pomocníků poskytuje větší bezpečnost. Izolují se části programu vyžadující administrátorská práva od zbytku kódu. Z pohledu administrátora je možnost udělit konkrétním uživatelům a skupinám oprávnění tyto pomocníky spouštět snad největší výhodou řešení.

Dalším možným rozšířením by bylo nějakým způsobem spojit KAuth se systémy jako je SELinux[2]. Zde by byla výhoda v tom, že akce, které by jinak byly uživateli umožněny jde omezit na úrovni operačního systému. Využitelnost KAuthu by se tak rozšířila například na omezení spouštění programů a použití knihoven (příkladem budiž KControl moduly) a skutečné omezení zdrojů dat. Například pokud by měl uživatel zakázáno používat vlastní pozadí na plochu, bylo by zamezeno programu plasma-desktop načítat jiné obrázky, než ty instalované do systému administrátorem - a to bez možnosti to obejít. Omezení na zdroje tak jak je v KDE4 je v porovnání s takovým řešením v podstatě bezzubé. Potom by mělo smysl odstranit některé typy omezení ze systému KConfig, a přímo je integrovat do KAuth.

Co se týče kombinace Kiosk, KAuthorized a KConfig, je rozhodně co dohánět. Bylo by dobré tyto části KDE pročistit a zdokumentovat. Zdvojení funkcí `KAuthorized::authorize()` a `KAuthorized::authorizeKAction()` je poněkud chaotické a vede ke zmatkům, kdy je jeden název akce používán s funkcemi náhodně. `KAuthorized::authorizeKAction()` by měla přijímat jako parametr pouze reference typu KAction. Kiosk jako celek je bohužel jednou z méně udržovaných částí KDE-Libs a určitá část nastavitelných omezení je zcela neefektivní.

[2]Security Enhanced Linux

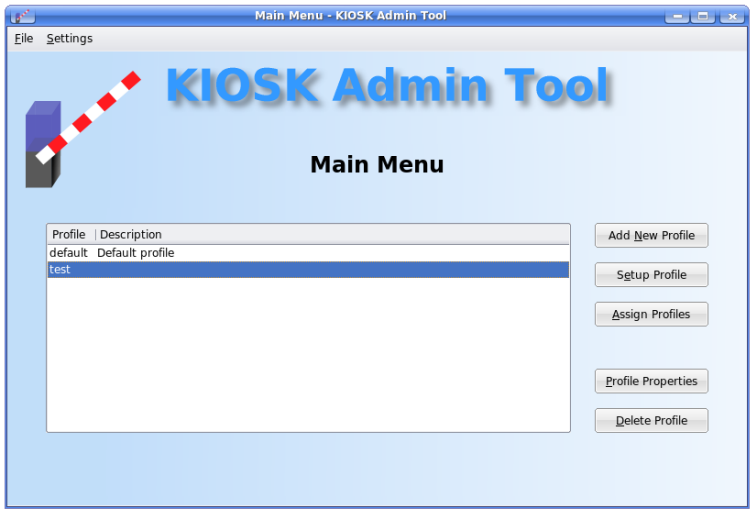
Kapitola 4

Nástroj KioskTool

Při psaní této kapitoly jsem hojně využíval knih [5] a [2]. Obzvláště [2] je vhodným zdrojem informací, protože je volně dostupná.

4.1 KioskTool v KDE 3

Aplikace KioskTool umožňovala v KDE3 spravovat některé specifické části Kiosk profilů pomocí grafického rozhraní. Nejprve popíši její funkčnost ve verzi pro KDE3 za pomocí obrázků, protože tak to bude nejlepší.

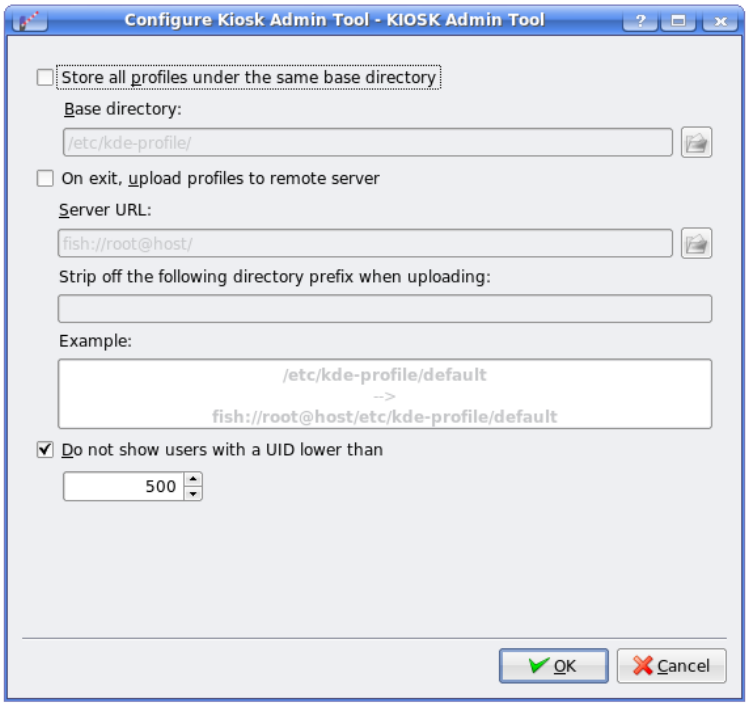


Obrázek 4.1: Úvodní obrazovka programu

Úvodní obrazovka 4.1 ukazuje seznam profilů s jejich popisem, lištu s menu a tlačítka pro manipulaci s profily. Velmi chybí možnost zkopírovat již existující profil.

Mezi nastavení programu 4.2 patří také možnost automaticky uložit profily na vzdálený server.

Dialog 4.3 pro vytvoření nového profilu je poměrně jednoduchý. Umožňuje nastavit název a popis profilu a dále který uživatel bude vlastnit složku s profilem (bude mít přístup pro zápis) a kde bude profil uložen.



Obrázek 4.2: Dialogové okno s nastavením

Při pohledu na obrazovku a dialogy pro přiřazení Kiosk profilů skupinám a uživatelům je nutné si připomenout, že když má uživatel přiřazen svůj vlastní profil, nevztahují se na něj profily pro skupiny kterých je členem. To program nijak nezvýrazňuje. Bylo by dobré, kdyby poskytoval nějaký pohled, který by ukazoval všechny profily efektivní pro uživatele.

Dialog 4.5 pro úpravu profilu je mnohem zajímavější. Je možné z něj spouštět KControl moduly a jiné části systému KDE a tak docílit toho, že KioskTool může využít již existující funkcionalitu. Není proto nutné znovu „vymýšlet kolo“ a uživatel pro nastavení Kiosk profilů používá stejné nástroje jako pro normální nastavení. Tato původní verze nástroje KioskTool definuje všechny moduly a jejich vlastnosti v jednom velkém XML souboru.

Na obrázku 4.6 je vidět jak takový modul vypadá. Uživatel může zamknout obrázek na pozadí plochy a ukázat si náhled tohoto obrázku (náhled ve smyslu, že se pozadí plochy dočasně zamění).

I při krátké době nutné na seznámení s nástrojem (KioskTool 1.0 v Kubuntu 9.04) jsem narazil na fatální chyby, kdy se bez zjevného důvodu zhroutil. Nebude tedy možné se spolehnout na korektnost kódu.

4.2 KioskTool v KDE 4

Port nástroje do KDE4 totiž již existuje, i když je nedokončený a dá se říci opuštěný. Uživatelské rozhraní se příliš nezměnilo: 4.7. Původní nastavení pomocí velkého XML souboru bylo odstraněno a nahrazeno mnoha menšími soubory (samozřejmě používají KConfig). To má za účel umožnit ostatím autorům napsat si pro své aplikace rozšíření. Nástroj však ztratil většinu svých starých vlastností, schopnost spouštět KControl moduly a náhledy, své pěkné (i když zbytečné) obrázkové vzezření a získal několik dalších chyb.

Obrázek 4.3: Dialog pro vytvoření nového profilu

Strukturální popis Aplikace KioskTool se skládá z několika základních komponent a využívá grafické rozhraní navržené pomocí nástroj Qt Designer (části rozhraní jsou specifikovány v .ui souborech, ze kterých se při kompilaci generuje kód). Vzhled je tedy alespoň z pohledu programátora částečně oddělen od funkce programu. Grafické prvky programu však přímo obsahují data se kterými se pracuje – není využito návrhového vzoru MVC[1].

Při startu programu jsou nejdříve vytvořeny základní komponenty `KAboutData` a `KApplication` a hlavní komponenta grafického rozhraní `KioskGui`. Ta je zobrazena. Potom je nastartováno vyhodnocování událostí.

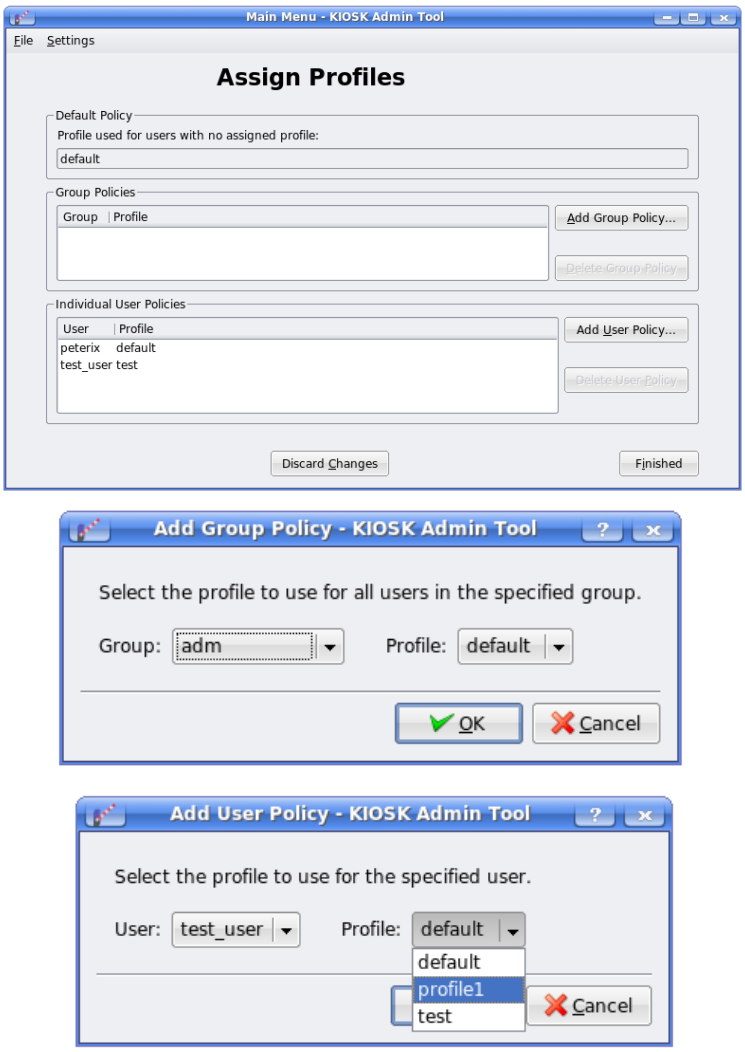
Komponenta `KioskGui` je odvozena od třídy `KXmlGuiWindow` – načítá tedy část svého vzhledu z .rc souboru 4.1 ve formátu XML.

```
<?xml version="1.0"?>
<!DOCTYPE gui SYSTEM "kpartgui.dtd">
<gui name="kioskgui" version="3.1">
<MenuBar>
 <Menu name="file">
    <Action name="upload_all"/>
 </Menu>
</MenuBar>
</gui>
```

Výpis 4.1: kiosktoolui.rc

`KioskGui` potom vytváří instance tříd `KioskRun` a `MainView`. `KioskRun` je pouhou obálkou nad souborem funkcí různého určení (tzv. God object anti-pattern) a je používána pro většinu manipulace s profily a spouštění dalších programů. `MainView` pak určuje vzhled celého programu. Je to třída generovaná z .ui souboru, obsahuje dvě úrovně nadpisů, tři tlačítka která mění význam podle kontextu a objekt typu `QStackedWidget`, který je určen pro zobrazení aktuální stránky. Při startu je to stránka se seznamem profilů (`PAGE_PROFILE_SELECTION`).

[1] Model-View-Controller

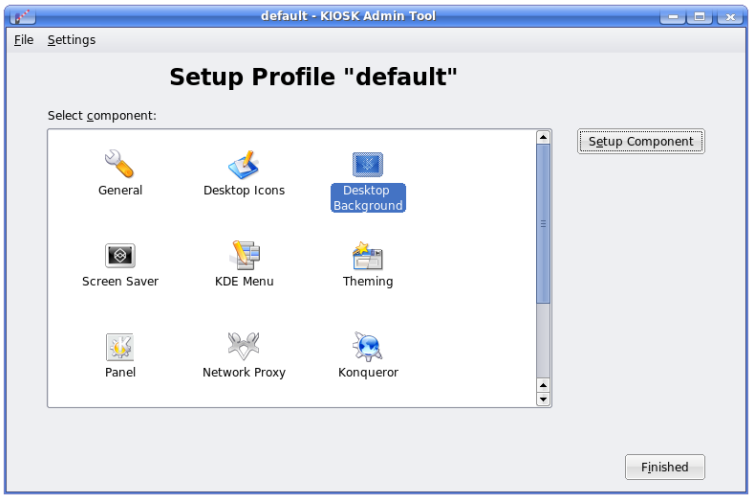


Obrázek 4.4: Dialogy pro přiřazení profilů

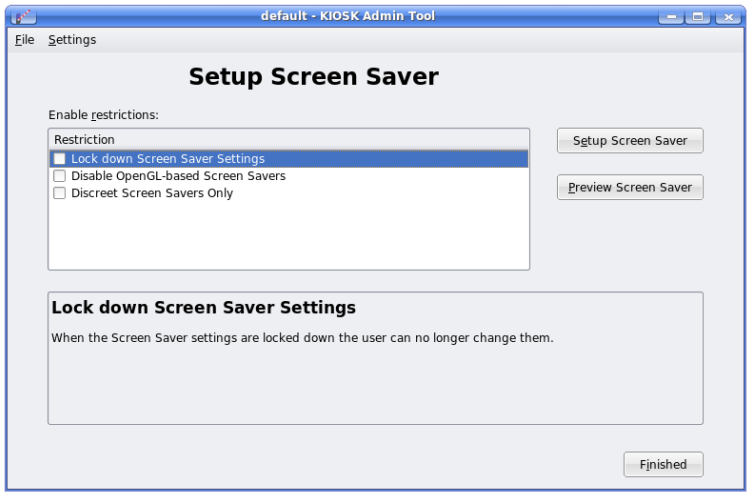
Přechod mezi stránkami je prováděn pomocí metody `selectPage(enum page)`, volané v reakci na akce uživatele a lze v hrubých obrysech popsat pomocí stavového automatu 4.8.

Ve stavech `PAGE_PROFILE_NEW` a `PAGE_PROFILE_PROPS` je používán stejný typ stránky. Jednou pro vytvoření nového profilu, podruhé pro změny v něm. `PAGE_PROFILE_ASSIGN` je stav, ve kterém je aktivní stránka pro přiřazení profilů uživatelům a skupinám. Ve stavu `PAGE_COMPONENTS_SELECTION` je načten profil a je zobrazena stránka se seznamem komponent pro stav `PAGE_COMPONENT` – zde přestává stačit jeden stavový diagram.

Je zřejmé, že způsob jakým je vytvořeno grafické rozhraní přímo určuje funkci programu. Přechod na stránku se seznamem komponent zapříčiní otevření profilu. Návrat na hlavní stránku se seznamem profilů pak způsobí jeho uložení. Platí to i naopak - technická omezení kladená na některé funkce programu se odrážejí v návrhu jeho grafického rozhraní. Například program nemůže upravovat více jak jeden profil, protože ve třídě KioskRun nastavuje pro spuštění KConfig modulů proměnné prostředí a kopíruje profil do dočasného umístění, kde ho může případná spouštěná aplikace měnit. Proto stavový automat a přechody mezi



Obrázek 4.5: Dialog s nastavením profilu



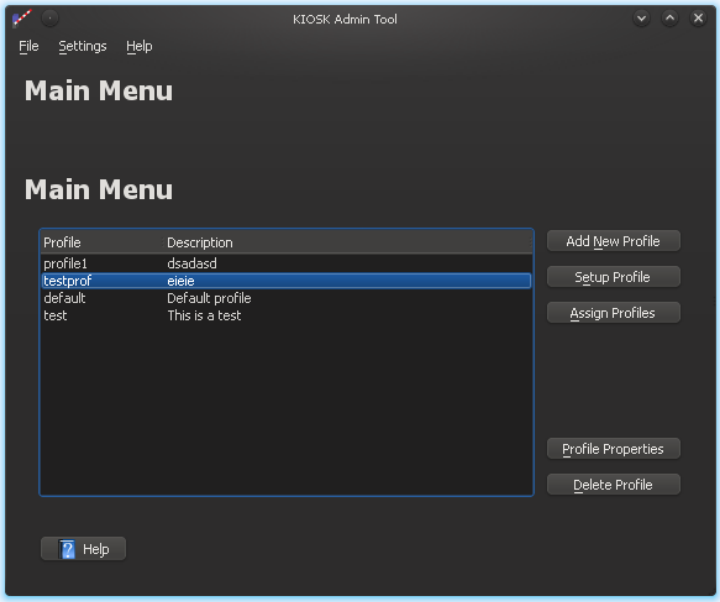
Obrázek 4.6: Dialog jedné komponenty nástroje

stránkami. To by se mělo dát rozšířit – program by měl např. být schopen pracovat s profily na pozadí, zobrazovat okno pro úpravy konkrétního profilu (ať už vlastní nebo z jiného programu) a zároveň třeba i druhé okno se seznamem všech uživatelů a k nim přiřazených profilů. Je tedy potřeba oddělit model pro práci s Kiosk profily a jeho stavy od kódu pro grafické rozhraní.

Vzniká tak také nutnost práce s asynchronními událostmi, protože provedení některých funkcí modelu může trvat déle než je přípustné a blokovat tak uživatelské rozhraní. V aplikaci, kde se pouze přechází mezi přesně vymezenými stavy je toto omluvitelné, ale pokud má uživatel otevřeno několik oken, nebude chtít například čekat, až se profil uloží na server.

Po těchto úpravách bude možné vylepšit uživatelské rozhraní programu. Stanovím si tedy několik cílů:

1. Krátkodobě - otestovat program a opravit v něm chyby, tuto verzi zveřejnit (bude součástí práce).



Obrázek 4.7: Uživatelské rozhraní se od verze z KDE3 příliš nezměnilo

2. Navrhnout a implementovat model pro práci s profily. Měl by být bezpečný a pro přístup k profilům používat KAuth pomocníka. Nabízí se použít Qt API pro stavové automaty [6].

3. Musí být možné upravovat profily zároveň z více aplikací, případně ručně. KioskTool toto nijak neřeší a je možné ztratit data pokud je spuštěn dvakrát.

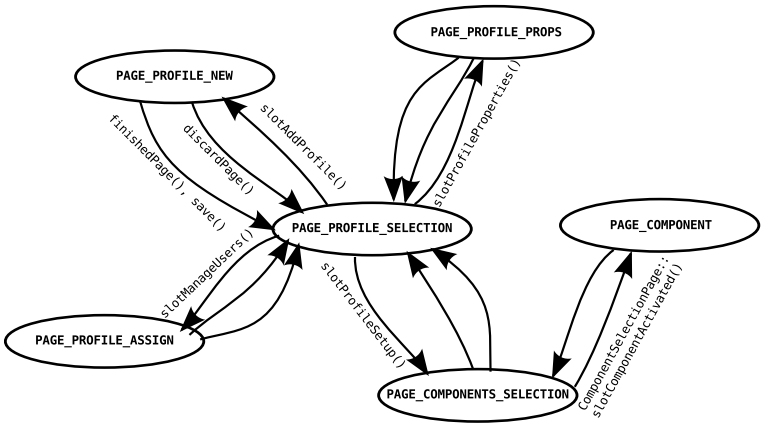
4. Implementovat několik jednoduchých terminálových programů pro práci s profily.

5. Nakonec vytvořit nové grafické rozhraní pro KioskTool.

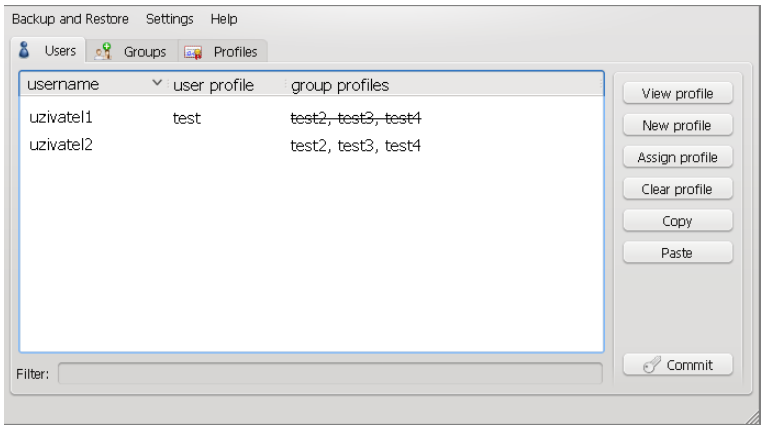
Triviální chyba Při testování jsem narazil pouze na jednu drobnou, ale závažnou chybu. Při přidělování profilů není možné přiřadit více jak jeden profil pro uživatele nebo skupiny. To znamená, že program není vůbec použitelný ke svému účelu. Problém má triviální řešení. Přiřazení se totiž řídilo tímto algoritmem:

```
  for( int idx = 0; idx < listGroups->topLevelItemCount(); ++idx )
2 {
      item = listGroups->topLevelItem(idx);
4     if (item->text(0) == group)
          break;
6 }
  if (item)
8 {
      Error();
```

To je zjevně špatné, protože proměnná item bude obsahovat nenulovou hodnotu, jakmile pude přiřazen alespoň jeden profil. Je tedy potřeba přidat za `break;` další řádek s `item = 0;`. Tímto se stává program použitelným pro běžného uživatele.



Obrázek 4.8: Stavy grafického rozhraní aplikace KioskTool



Obrázek 4.9: Karta pro uživatele

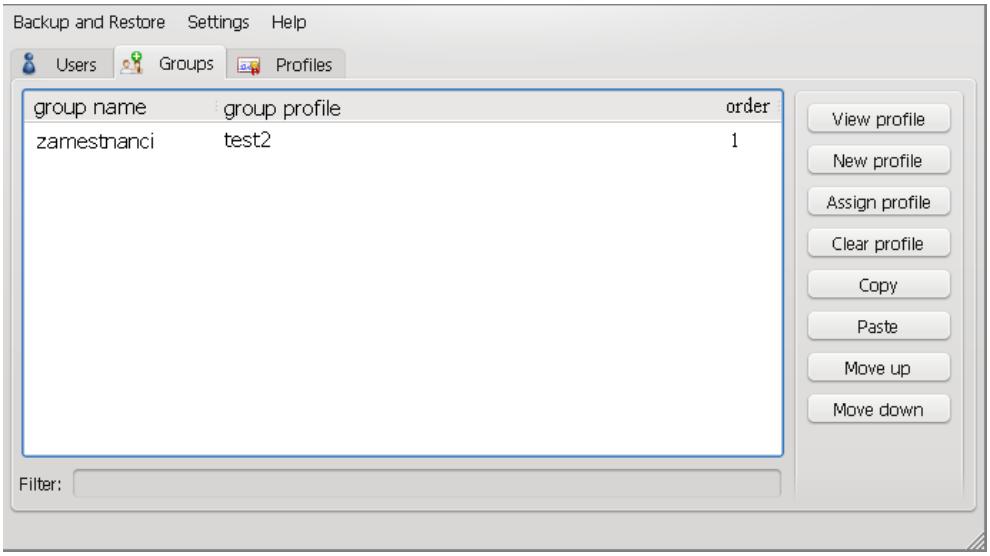
4.3 Návrh uživatelského rozhraní

Zde je použit nástroj QtDesigner pro navržení vzhledu uživatelského rozhraní. Funkční část bude muset být dále implementována a tento vzhled je nutno brát pouze jako první krok.

Namísto stránek a stavových automatů bude mít program několik záložek. Zůstanou tedy tlačítka s akcemi po stranách, ale uživatel bude mít mnohem větší přehled kde se nalézá. Seznam akcí také bude významně rozšířen.

Na obrázku 4.9 je program s otevřenou kartou pro pohled na uživatele. Menu „File“ bylo nahrazeno menu „Backup and Restore“, protože neobsahovalo nic jiného než akci pro zálohování profilů na server. Menu s nastavením a nápovědou zůstává. Většinu plochy programu zabírá pohled na seznam uživatelů a k nim přiřazených uživatelských a skupinových profilů. Pokud má uživatel přiřazen jak uživatelský, tak skupinové profily, je zde fakt, že jsou ty skupinové neefektivní zvýrazněn přeškrtnutím.

V seznamu po pravé straně je několik pro KioskTool nových akcí. Akce „View Profile“ slouží k otevření okna s profilem. Pokud má uživatel pouze skupinové profily, otevře se stejné okno, ale jeho obsah bude výslednicí skupinových profilů a nebude možné jej měnit (toto musí být zvýrazněno aby nebyl uživatel programu uveden v omyl). Akce „View Profile“ přiřadí uživateli nový prázdný profil. Akce „Assign Profile“ slouží k přiřazení existujícího

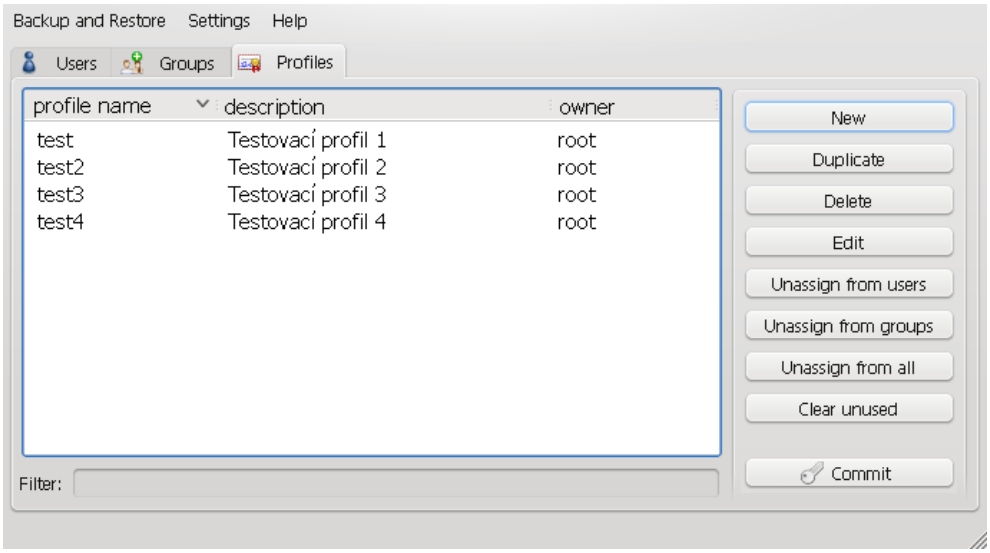


Obrázek 4.10: Karta pro skupiny

profilu uživateli. „Clear Profile“ odebere uživateli jeho profil.

Pod seznamem je pole pro filtrování uživatelů podle zadaného textu. Je tak možné rychle vyhledávat v obsáhlém seznamu uživatelů.

Karta se skupinami 4.10 je podobná kartě pro uživatele, zobrazuje však skupiny, jim přiřazené skupinové profily a jejich pořadí. K seznamu akcí z karty s uživateli jsou přidány akce pro změnu jejich pořadí. Pořadí by také mělo být možné měnit přetažením položek kurzorem na jinou pozici v seznamu.

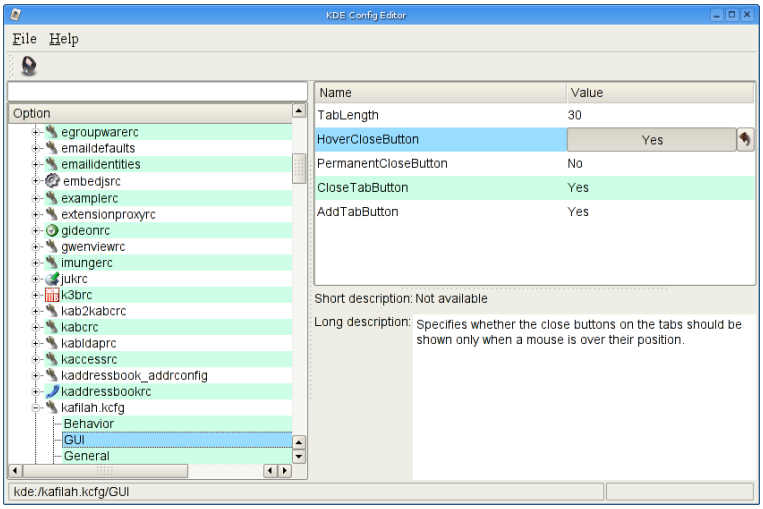


Obrázek 4.11: Karta pro profily

Záložka s profily 4.11 obsahuje jak název napovídá seznam profilů a akce pro práci s nimi. V pořadí odshora vytvoření nového profilu, kopie již existujícího, smazání profilu, úpravu profilu a dále akce pro zrušení přiřazení profilů pro uživatele, skupiny a obojí zároveň. Seznam je zakončen akcí pro smazání všech nepoužívaných profilů.

Všechny záložky budou používat stejný model pro naplnění seznamů. Všechny také mají akci pro uložení změn „Commit“. Při jejím použití by se měl zobrazit seznam všech uživatelem provedených změn a umožnit tak uživateli je ještě jednou zkontrolovat před zapsáním na disk.

Změny v profilech samotných by bylo dobré provádět způsobem podobným jako využívá nástroj KConfigEditor. Ten vypadá zhruba takto: 4.12. KConfigEditor používá rozšíření systému KConfig o deklarace klíčů pomocí systému KConfigXT. Ten je zpravidla používán k vygenerování dialogů pro nastavení programů. V souboru KConfigXT se uvede jaké klíče z konfigurace program používá a také jejich typ. Navíc se tyto soubory instalují společně s programy – lze jich tedy využít v editoru nastavení. Integrace tohoto nástroje a rozšíření a oprava KConfigXT souborů by pak vedly k vcelku elegantnímu řešení problému úpravy profilů.



Obrázek 4.12: Ukázka programu KConfigEditor. Převzato z [7].

Kapitola 5

Závěr

V kapitole 2 byly popsány technologie a rozhraní, na kterých je dále stavěno.

V kapitole 3 byl učiněn neúspěšný pokus o integraci KAuth do KAuthorized. Byly přitom odkryty závažné nedostatky v KAuth a rozhraních, nad kterými je postaven. Nejzajímavější je pak Denial of Service útok na PolicyKit. Ten jsem objevil za pomoci implementovaného testovacího nástroje kauthDoS. KAuth byl integrován do KAuthorized, ale díky zmíněné chybě se při jeho použití stává prostředí KDE naprosto nepoužitelným. Začlenění tohoto celku do kdelibs tedy není z praktických a bezpečnostních důvodů vhodné a proto jsem poslední bod zadání přeskočil.

Dále byl implementován nástroj kioskpklaconvert a KAuth pomocník kioskpklahelper. Ty umožňují konvertovat omezení akcí a zdrojů z KConfigu do nastavení PolicyKit Local Authority.

Kapitola 4 popisuje port nástroje KioskTool do KDE4, navrhuje pro něj nové grafické rozhraní a také popisuje provedené změny nutné pro zprovoznění nástroje. Tím je také splněn čtvrtý bod zadání. V práci započaté na nástroji KioskTool budu pokračovat.

V příloze B jsou informace o umístění zdrojových kódů, obsahu přiloženého datového nosiče a instalaci a zprovoznění KDE4.

Příloha A

Vymezení pojmů

Authorization Services je obdoba PolicyKitu pro operační systém Apple OSX.

KAuth je rozhraní nad autorizačním řešením jako je například PolicyKit.

KAuthorized je tenké rozhraní nad KConfigem/Kioskem umožňující dotazování, zda jsou některé typy akcí povoleny.

KConfig je systém pro ukládání a načítání nastavení v KDE.

KConfigXT je systém pro uložení metadat pro systém KConfig. Popisuje klíče použité v jednotlivých programech a jejich typy.

KControl modul je modulem pro nastavení určité části KDE – například rozložení klávesnice, vzhled oken, apod.

Kiosk je souhrnný pojem pro některé vlastnosti KConfigu a postup načítání konfigurace obecně.

Kiosk profil je přiřazený uživateli nebo skupině uživatelů. Má vyšší prioritu než normální uživatelská nastavení KDE, ale nižší než globální systémové nastavení.

KioskTool je aplikace pro správu Kiosk profilů. V KDE 3 umožňovala jednoduše nastavit některé aspekty Kiosku/KConfigu bez nutnosti měnit profily ručně.

KDE je komunita vývojářů pracujících na tzv. KDE Software Compilation. Zde používám označení KDE ve starém významu – jako uživatelské prostředí.

PolicyKit je starší verze autorizačního rozhraní dostupná v Linuxu.

PolicyKit1, nebo také „polkit“ je jeho novější verze. Právě tu budu převážně používat.

PolicyKit autorita je centrálním prvkem systému PolicyKit. Slouží pro uložení autorizačních dat.

PolicyKit Local Authority je výchozí implementace PolicyKit autority. Používá textové soubory.

polkit-qt je soubor knihoven obalující PolicyKit.

polkit-kde je nadstavba nad polkit-qt pro prostředí KDE. Implementuje autentizačního agenta a KControl modul pro práci s autorizacemi.

polkitd je systémová služba PolicyKitu a implementuje tzv. autoritu.

Qt je balík knihoven v jazyce C++.

QtDesigner je program pro vizuální návrh uživatelského rozhraní programů na bázi Qt knihoven.

Příloha B

Dodatečné informace

Umístění zdrojových kódů Zdrojové kódy KDE4 a jeho knihoven je možné prohlížet na adrese http://websvn.kde.org/trunk/KDE/. Většina práce byla vytvořena jako součást balíku kdelibs a byla vyvíjena proti SVN revizi 1125098.

Přiložené datové médium obsahuje patch pro kdelibs, ve kterém jsou implementovány změny z třetí kapitoly. Kód čtvrté kapitoly je ve složce kiosktool. Další informace k obsahu média jsou v souboru README.

Instalace - Integrace KAuth do KAuthorized Instalace kódu z třetí kapitoly pravděpodobně nebude triviální.

Základem je dokumentace na adrese http://techbase.kde.org/Getting_Started/Build/KDE4. Pokud se KDE instaluje do domovského adresáře k tomu vytvořeného uživatele, je nutné přihlédnout k tomu, že registrační soubory KAuth pomocníků pro systém D-Bus a soubory s definicemi akcí pro PolicyKit je třeba překopírovat do systémových adresářů. Bez toho nemůže KAuth fungovat.

Konkrétně je nutné překopírovat tyto soubory:

Vše z `$prefix/etc/dbus-1/system.d` do `/etc/dbus-1/system.d`

Vše z `$prefix/share/polkit-1/actions` do `/usr/share/polkit-1/actions`

Vše z `$prefix/share/dbus-1/system-services` do `/usr/share/dbus-1/system-services`

`$prefix` je zde složka, kam se instaluje KDE4 zkompilované ze zdrojových kódů. V případě použití doporučovaného uživatele kde-devel to bude `/home/kde-devel/kde`

Kód z třetí kapitoly **VYŽADUJE**, aby byl v systému PolicyKit1, a **POUZE** PolicyKit1. Konvertor pravděpodobně totiž nebude fungovat správně se starší verzí.

Použití KAuth v KAuthorized je ve výchozím stavu vypnuto! K jeho zapnutí je potřeba v kdelibs/kdecore/kernel/kauthorized.cpp odkomentovat blok, který používá metodu status(). Postup je takový, že se zkompiluje kdelibs bez KAuth, nastartuje se sezení KDE tak, aby fungoval KAuth/PolicyKit (je nutné spouštět pomocí KDM a ověřit, že KCmodul pro nastavení času korektně zobrazuje autentizačního agenta z polkit-kde). Poté je potřeba se přepnout do původního sezení, odkomentovat v kauthorized.cpp příslušný blok kódu a zkompilovat kdelibs. Po přepnutí do druhého sezení bude další spuštěný program používat KAuth skrze KAuthorized. V tomto stavu je možné dočasně testovat funkčnost řešení (alespoň než se zasekne polkitd).

Instalace - KioskTool KioskTool by neměl vyžadovat žádné zvláštní zacházení. Není ani potřeba použít oddělený uživatelský účet kde-devel. Stačí vygenerovat buildsystém pomocí cmake, program zkompilovat a nainstalovat.

Příklad postupu ze složky se zdrojovým kódem, umístěné tak, aby se do ní dalo zapisovat:

```
mkdir build && cd build

cmake ..

make

sudo make install
```

Při použití kde-devel stačí umístit zdrojové kódy KioskTool mezi zdrojové kódy ostatních částí KDE a sestavit je pomocí `cmakekde`.

Příloha C

Backtrace z KAuthDoS a PolicyKit déemona

```
  #0  poll () from /lib/libc.so.6
2 #1  in socket_do_iteration () from /usr/lib/libdbus-1.so.3
  #2  in _dbus_transport_do_iteration () from /usr/lib/libdbus-1.so.3
4 #3  in _dbus_connection_do_iteration_unlocked () from /usr/lib/libdbus-1.so
    .3
  #4  in _dbus_connection_block_pending_call () from /usr/lib/libdbus-1.so.3
6 #5  in egg_dbus_connection_pending_call_block (
        connection=0x61a990, pending_call_id=196205)
8       at eggdbusconnection.c:2521
  ...
10 #16 in g_main_context_dispatch ()
        from /usr/lib/libglib-2.0.so.0
12 #17 in g_main_context_iterate ()
        from /usr/lib/libglib-2.0.so.0
14 #18 in g_main_loop_run ()
        from /usr/lib/libglib-2.0.so.0
16 #19 in main ()
```

Výpis C.1: Backtrace z démona polkitd při „zamrznutí“ (zkrácený)

```
  #0  in poll () from /lib/libc.so.6
#2 #1  in socket_do_iteration ()
      from /usr/lib/libdbus-1.so.3
#4 #2  in _dbus_transport_do_iteration ()
      from /usr/lib/libdbus-1.so.3
#6 #5  in egg_dbus_connection_pending_call_block (
      connection=0x6add50, pending_call_id=74401)
#8    at eggdbusconnection.c:2521
  #6  in polkit_authority_check_authorization_sync ()
#10   from /usr/lib/libpolkit-gobject-1.so.0
  #7  in PolkitQt1::Authority::checkAuthorizationSync ()
#12   from /home/kde-devel/kde/lib/libpolkit-qt-core-1.so.0
  #8  in KAuth::Polkit1Backend::actionStatus ()
#14   at kdelibs/kdecore/auth/backends/polkit-1/Polkit1Backend.cpp:87
  #9  0x000000000040162e in main (argc=1, argv=<value optimized out>)
#16   at kdelibs/kdecore/auth/kauthDoS.cpp:40
```

Výpis C.2: Backtrace z testovacího polkitd při „zamrznutí“ (zkrácený)

Příloha D

Fragmenty kódu - úpravy v KAuth

```
Action::AuthStatus Polkit1Backend::actionStatus(const QString &action)
{
    PolkitQt1::UnixProcessSubject subject(QCoreApplication::applicationPid()
        );
    PolkitQt1::Authority::Result r =
        PolkitQt1::Authority::instance()->
        checkAuthorizationSync(action, &subject, PolkitQt1::Authority::None)
            ;
    switch (r) {
    case PolkitQt1::Authority::Yes:
        return Action::Authorized;
    case PolkitQt1::Authority::No:
    case PolkitQt1::Authority::Unknown:
        return Action::Denied;
    default:
        return Action::AuthRequired;
    }
}
```

Výpis D.1: Autorizace akce v PolicyKit1

```
...
case PolkitQt1::Authority::Unknown:
    PolkitQt1::Authority::ErrorCode error =
        PolkitQt1::Authority::instance()->lastError();
    PolkitQt1::Authority::instance()->clearError()
    // E_CheckFailed should indicate that an action doesn't exist
    if(error == PolkitQt1::Authority::E_CheckFailed)
        return Action::Invalid;
    else // other errors. we treat them like before
        return Action::Denied;
...
```

Výpis D.2: Autorizace akce v PolicyKitu po úpravách

```
1 Action::AuthStatus AuthServicesBackend::actionStatus(const QString &action)
  {
3     // check if the action exists first, return error if not
      OSStatus exists = AuthorizationRightGet(action.toUtf8(), NULL);
5     if(exists != errAuthorizationSuccess)
          return Action::Invalid;
7 ...
```

Výpis D.3: Ověření existence akce v OSX Authorization Services

Literatura

[1] Bobčík, B.: PAM - správa autentizačních mechanismů [online]. http://www.root.cz/clanky/pam-sprava-autentizacnich-mechanismu/, 2000-09-19 [cit. 2010-05-16].

[2] Ezust, A.; Ezust, P.: *An Introduction to Design Patterns in C++ with Qt 4*. Prentice Hall; 1 edition (September 10, 2006), 2006, iSBN 0-131-87905-7.

[3] Freddi, D.: Using KAuth actions in your application [online]. http://techbase.kde.org/Development/Tutorials/KAuth/KAuth_Actions, 2010-02-12 [cit. 2010-05-16].

[4] Komunita KDE: Kiosk/Introduction [online]. http://techbase.kde.org/KDE_System_Administration/Kiosk/Introduction, 2008-03-14 [cit. 2010-05-16].

[5] Molkentin, D.: *The book of Qt 4 :the art of building Qt applications*. No Starch Press, 2007, iSBN 1-593-27147-6.

[6] Nokia Corporation: The State Machine Framework [online]. http://doc.qt.nokia.com/4.6/statemachine-api.html, 2010 [cit. 2010-05-16].

[7] Rusin, Z.: Domovská stránka programu KConfigEditor [online]. http://extragear.kde.org/apps/kconfigeditor/, 2010 [cit. 2010-05-16].

[8] Zeuthen, D.: Referenční manuál k PolicyKitu [online]. http://hal.freedesktop.org/docs/polkit/, 2009-07-24 [cit. 2010-05-16].

[9] Zeuthen, D.: Manuálová stránka PolicyKit1 Local Authority [online]. http://hal.freedesktop.org/docs/polkit/pklocalauthority.8.html, 2010 [cit. 2010-05-16].

[10] Zeuthen, D.: Manuálová stránka PolicyKit1 [online]. http://hal.freedesktop.org/docs/polkit/polkit.8.html, 2010 [cit. 2010-05-16].