



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ZDOKONALENÍ INTEGRACE SSSD A SUDO

IMPROVED INTEGRATION OF SSSD AND SUDO

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

MICHAL ŠRUBAŘ

VEDOUcí PRÁCE

SUPERVISOR

Prof. Ing. TOMÁŠ VOJNAR, Ph.D.

BRNO 2014

Abstrakt

Cílem této bakalářské práce je zlepšení integrace mezi SUDO a SSSD se zaměřením na vylepšení podpory SUDO pravidel uložených na serveru FreeIPA. Zabývá se popisem LDAP SUDO provideru a prezentuje návrh a implementaci IPA SUDO providera. Navržený provider eliminuje nadbytečnou režii překládání SUDO pravidel z IPA SUDO schématu do nativního LDAP SUDO schématu na straně FreeIPA serveru.

Abstract

The purpose of this thesis is to improve integration between SUDO and SSSD with a focus on improved support of SUDO rules stored on an FreeIPA server in the native IPA SUDO scheme. It presents documentation of LDAP SUDO provider and also the design and implementation of a native IPA SUDO provider. The designed provider eliminates an unnecessary overhead of exporting SUDO rules from IPA SUDO schema to native LDAP SUDO scheme on an FreeIPA server.

Klíčová slova

SUDO, nativní LDAP SUDO schéma, IPA SUDO schéma, FreeIPA, SSSD, LDAP SUDO provider, IPA SUDO provider

Keywords

SUDO, native LDAP SUDO schema, IPA SUDO schema, FreeIPA, SSSD, LDAP SUDO Provider, IPA SUDO Provider

Citace

Michal Šrubař: Zdokonalení integrace SSSD a SUDO, bakalářská práce, Brno, FIT VUT v Brně, 2014

Zdokonalení integrace SSSD a SUDO

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana ...

.....

Michal Šrubař
20. května 2014

Poděkování

Rád bych poděkoval prof. Tomáši Vojnarovi za veškerou poskytnutou pomoc. Dále mému technickému vedoucímu, ing. Jakubu Hrozkovi za trpělivé a velice ochotné vedení. Poděkování patří také komunitám projektů SSSD a FreeIPA za trpělivé zodpovídání veškerých dotazů a to zejména Pavlu Březinovi.

© Michal Šrubař, 2014.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1	Úvod	3
2	Správa bezpečnostních politik	4
2.1	SUDO	4
2.1.1	Distribuce SUDO pravidel	5
2.1.2	Nativní LDAP SUDO schéma	6
2.1.3	Integrace identit a politik	8
2.2	FreeIPA	9
2.2.1	IPA sudo schéma	9
2.2.2	Definice SUDO pravidel	12
2.3	System Security Services Daemon	13
2.3.1	Aktualizace SUDO pravidel v cache paměti	14
3	SSSD a SUDO provider	15
3.1	Knihovny třetích stran	15
3.2	Architektura SSSD	17
3.3	SSSD z pohledu programu SUDO	18
3.3.1	Zjištění informací o uživateli	19
3.3.2	Autentizace uživatele	19
3.3.3	Získání sudo pravidel	19
3.4	LDAP SUDO Provider	20
3.4.1	Inicializace ldap modulu	20
3.4.2	Plánování aktualizací	22
3.4.3	Aktualizace SUDO pravidel	22
3.4.4	Asynchronní modul	25
3.4.5	Uložení SUDO pravidel do sysdb	27
3.4.6	LDAP SUDO Handler	27
3.5	IPA SUDO provider	27
3.6	Dereferenční dotaz	29
4	Nativní IPA SUDO Provider	30
4.1	Návrh	30
4.1.1	Správný přístup k získávání SUDO pravidel	30
4.1.2	Stažení SUDO pravidel	32
4.1.3	Překlad pravidel	34
4.1.4	Využitelnost LDAP pluginu	37
4.2	Implementace	37
4.3	Testování	39

4.3.1 Metodika	39
5 Závěr	40
A Přiložené CD obsahuje:	42

Kapitola 1

Úvod

Tato bakalářská práce popisuje správu bezpečnostních politik s pomocí použití programu SUDO. Popisuje problémy, na které může administrátor narazit při nasazení tohoto řešení v doménovém prostředí. Představuje možná řešení těchto problémů založená na použití technologií FreeIPA a SSSD, které jsou používány v Red Hat Enterprise prostředích. Tato řešení ovšem vykazují určité nedostatky, které tato práce odkrývá. Smyslem této práce je tedy tyto nedostatky identifikovat, navrhnout jejich řešení a tyto řešení implementovat. Od čtenáře je v práci očekávána uživatelská znalost programu SUDO, základních konceptů online adresářů a LDAP protokolu. Tato práce byla zadána ve spolupráci s firmou Red Hat.

Práce je rozdělena do pěti kapitol. První kapitola popisuje obsah práce. V kapitole 2 jsou popsány problémy, které vznikají při centralizované správě identit a jejich bezpečnostních politik za pomoci programu SUDO. Diskutuje také nad možným řešením v podobě použití serveru FreeIPA a démonu SSSD.

Třetí kapitola se blíže zabývá démonem SSSD. Popisuje jeho architekturu, činnost z pohledu programu SUDO a dokumentuje činnost pluginů ipa a ldap, které SUDO provider používá pro zpracování SUDO pravidel na straně klienta. Tento popis byl sestaven na základě experimentování a čtení zdrojových kódů projektu SSSD. Sekce 3.5 poté detailně popisuje hlavní nedostatek v démonu SSSD, na který se tato práce zaměřuje.

Kapitola 4 následně prezentuje návrh, který posloužil k odstranění nedostatku identifikovaného v sekci 3.5.

Poslední kapitola práce poté zhodnocuje dosažené výsledky a diskutuje nad možnostmi pokračování v této práci. Tato práce používá následující konvence:

1. Všechny důležité pojmy jsou vysázeny **tučným písmem**.
2. Nové pojmy jsou vysázeny *kurzivou*.
3. Hodnoty atributu ipaUniqueID¹ byly záměrně zkráceny kvůli velikosti textu a obrázků.
4. Všechny názvy souborů, jejich obsahy, uživatelské účty, klíčových slova jazyka C nebo jména identifikátorů jsou vysázeny **strojovým písmem**.

¹ Atribut je blíže popsán v sekci 2.2.1.

Kapitola 2

Správa bezpečnostních politik

Tato kapitola popisuje možnost správy bezpečnostních politik pomocí programu SUDO. Popisuje možné problémy, které mohou nastat při distribuci pravidel, kterými se SUDO řídí a dále LDAP schéma, které SUDO používá pro uložení těchto pravidel v LDAP adresářích. Zabývá se nedostatky, které zmíněné schéma provázejí z pohledu domény a představuje nové schéma projektu FreeIPA, které se tyto nedostatky snaží řešit. Dále popisuje klientského démona SSSD, který je nezbytný pro využití tohoto schématu.

2.1 SUDO

Pomocí programu SUDO má administrátor možnost delegovat oprávnění správce¹ ostatním uživatelům bez nutnosti sdílení hesla správce. Tento program umožňuje uživateli operačního systému spustit jeden příkaz jako jiný uživatel. Ve většině případů je tímto uživatelem právě správce linuxového systému, neboli uživatel *root*, tedy uživatel, který má nejvyšší oprávnění. Jestliže uživatel spustí nějaký příkaz za pomoci programu SUDO, pak není vyzván k zadání hesla správce systému, ale k zadání hesla daného uživatele [1, Sekce 4.4.3.2]. Bezpečnostní politiky, kterými se SUDO řídí, se nazývají SUDO pravidla. Někdy jsou tato pravidla označována jako *sudoers*. Tato pravidla se zapisují jako následující čtveřice:

```
user host_list = ( user_list ) command_list
```

- **user** – specifikuje uživatele nebo skupinu, na kterou bude pravidlo aplikováno,
- **host_list** – reprezentuje seznam koncových systémů,
- **user_list** – (nepovinné) specifikuje, pod kterým uživatelem, má být příkaz proveden
- **command_list** – seznam povolených/zakázaných příkazů.

Jestliže chce správce počítače s doménovým jménem `client.example.cz` dát například uživateli `xsruba03` oprávnění spouštět příkaz `/sbin/fdisk` s právy uživatele `root`, pak musí definovat následující pravidlo:

```
xsruba03 client.example.cz = /sbin/fdisk
```

¹uživatele `root`

SUDO pravidla se definují v souboru `/etc/sudoers`. Tento soubor by měl být editován pouze pomocí nástroje **visudo**, který provádí automatickou kontrolu syntaxe. Ve výchozím nastavení se v tomto souboru nachází jediné pravidlo definující práva uživatele root.

```
root ALL = ( ALL ) ALL
```

Toto pravidlo dává uživateli root oprávnění spustit jakýkoliv příkaz jako jakýkoliv uživatel na kterémkoliv systému, kde je výše uvedené pravidlo definováno.

2.1.1 Distribuce SUDO pravidel

V rozsáhlejších sítích má administrátor na starost více než jednu koncovou stanici. Každou stanici může použít více než jeden uživatel, který může potřebovat provést úlohu, pro kterou potřebuje vyšší oprávnění. Proto by chtěl správce distribuovat tyto pravidla mezi více počítačů. SUDO ovšem nemá nativní způsob, jak tato pravidla mezi více počítačů distribuovat [2, Kapitola 20]. Spravovat tato pravidla ručně na všech počítačích, které administrátor spravuje, by bylo ovšem velice pracné a časově náročné. Otázkou tedy zůstává jak uvedená pravidla šířit mezi více počítačů. Existuje několik způsobů jak toho docílit:

1. Ručně distribuovat příslušná pravidla mezi více systémů za použití standardních linuxových nástrojů, jako jsou *cron*, *scp*, *rsync*, ... Tento přístup by ovšem mohl být časově náročný a také by nemusel být spolehlivý.
2. Použít speciální nástroje, jako je například *Puppet*², který sleduje a automaticky šíří konfigurační soubory.
3. Uložit SUDO pravidla do centralizované databáze.

V souvislosti s bodem 3 se jeví jako výhodné použití adresářů LDAP³. Tyto adresáře mohou uchovávat různé typy informací a jsou tak používány organizacemi a institucemi k ukládání informací o jejich zaměstnancích, uživateli nebo zařízeních jejich počítačových sítí. Jsou charakterizovány jako služba typu „jednou-zapsat-a-mnohokrát-číst“⁴. Jsou optimalizovány pro rychlé čtení a vyhledávání, protože předpokládají, že budou uchovávat informace, které nebudou příliš často modifikovány [3]. SUDO pravidla jsou většinou jednou definována správcem systému a mnohokrát použita uživateli. Uložení SUDO pravidel do LDAP adresáře je tedy velice vhodné a nabízí následující výhody:

1. Jestliže jsou SUDO pravidla centralizována, pak se správce systému nemusí starat o jejich distribuci. Spravuje pouze jediný soubor s pravidly.
2. Vyhledávání pravidla v LDAP adresáři je rychlejší. Při použití lokálních sudoers⁵ musí SUDO přečíst celý soubor `/etc/sudoers`. Při uložení pravidel v LDAP adresáři je potřeba pouze několik LDAP dotazů [4].
3. Jestliže udělá správce v sudoers souboru chybu, pak se program SUDO nespustí. Do LDAP adresáře není možné uložit data, která nesplňují dané schéma. Správná syntaxe je tedy zaručena. Ovšem stále je možné udělat chybu v uživatelském jménu, příkazu nebo názvu koncového systému.

²<http://puppetlabs.com>

³Adresář, ke kterému se přistupuje pomocí LDAP protokolu.

⁴write-once-read-many-times

⁵tj. když jsou SUDO pravidla definována v lokální `/etc/sudoers` souboru

Nevýhodou centralizované správy může být situace, kdy dojde k výpadku serveru⁶, na kterém jsou pravidla uložena. V takovém případě nejsou pravidla pro sudo dostupná a není možné je využívat. V praxi se tento problém dá řešit replikací nebo cachováním sudo pravidel.

2.1.2 Nativní LDAP SUDO schéma

LDAP schéma je možno si představit jako množinu pravidel⁷, která definují, jaký typ dat je možno v adresáři uchovat a jak klienti nebo server mají s těmito daty pracovat. Skládá se z definice typů atributů a tříd [3, Kapitola 8]. SUDO definuje vlastní schéma, které je použito při vytváření nebo vyhledávání SUDO pravidel. Schéma tedy definuje, které atributy je nutno použít pro uložení prvků⁸ SUDO pravidla. Definuje také třídu, ze které jsou záznamy, reprezentující SUDO pravidla, odvozeny. Tato třída se nazývá „sudoRole“ a její definice vypadá následovně:

```
objectclass (
  1.3.6.1.4.1.15953.9.2.1
  NAME 'sudoRole' SUP top STRUCTURAL
  DESC 'Sudoer Entries'
  MUST ( cn )
  MAY ( sudoUser $ sudoHost $ sudoCommand $ sudoRunAs $
        sudoRunAsUser $ sudoRunAsGroup $ sudoOption $
        sudoNotBefore $ sudoNotAfter $ sudoOrder $ description
  )
)
```

Atribut `sudoUser` specifikuje uživatele, na kterého bude pravidlo aplikováno. Uživatele je možno specifikovat několika způsoby, které popisuje tabulka 2.1. SUDO pravidlo je také možné aplikovat na všechny uživatele, čehož je možno docílit použitím hodnoty `ALL`.

Tabulka 2.1: Popis možných hodnot atributu `sudoUser`.

Možná hodnota atributu	Příklad
uživatelské jméno	<code>xsruba03</code>
UID	<code>#9843</code>
skupina uživatelů	<code>%3bit</code>
GID	<code>%#43</code>
unixová síťová skupina	<code>+admins</code>
non-unixová síťová skupina	<code>:%finance_dp</code>
všichni uživatelé	<code>ALL</code>

Jestliže víme, na koho bude pravidlo aplikováno, musíme dále specifikovat, na kterých systémech bude toho pravidlo platné. To specifikuje atribut `sudoHost`, jehož možné hodnoty uvádí tabulka 2.2.

⁶Prvek, na kterém jsou závislé ostatní části systému, je v angl. označován jako „*single point of failure*“.

⁷Zde se nejedná o množinu SUDO pravidel.

⁸Na koho bude pravidlo aplikovatelné, kde, jaké obsahuje příkazy, ...

Tabulka 2.2: Popis možných hodnot atributu `sudoHost`.

Možná hodnota atributu	Příklad
jméno počítače	<code>client.example.cz</code>
IP adresa	<code>fe80::8a9f:faff:fe0c:b989</code>
adresa sítě	<code>192.168.1.0/24</code>
síťová skupina	<code>+servers</code>
kterýkoliv počítač	<code>ALL</code>

Posledním atributem pro sestavení SUDO pravidla, jehož možné hodnoty popisuje tabulka 2.3, je atribut `sudoCommand`. Ten specifikuje, který unixový příkaz má uživatel povolen/zakázán spouštět. Je také možné specifikovat konkrétní parametry daného příkazu.

Tabulka 2.3: Tabulka popisující možné hodnoty atributu `sudoCommand`.

Možná hodnota atributu	Popis
povolený příkaz/zakázaný příkaz	<code>/sbin/blkid /dev/sda1</code>
zakázaný příkaz	<code>!/sbin/fdisk</code>
všechny příkazy	<code>ALL</code>

SUDO příkaz je také možné spustit, jako jiný uživatel než `root`. Specifikovat uživatele popř. skupinu, pod kterou bude uživatel moct daný příkaz provést, specifikují atributy `sudoRunAsUser` a `sudoRunAsGroup`. Ve starších verzích programu SUDO byl také používán atribut `sudoRunAs`. Ten už se dnes ovšem nedoporučuje používat a považuje se za „zastaralý“⁹.

K SUDO pravidlu je také možno specifikovat různé volby, které například určují, jak bude vypadat řetězec, který požádá uživatele o zadání hesla, specifikovat zda bude heslo vůbec požadováno, a mnoho dalších. Všechny tyto volby je možné definovat pomocí atributu `sudoOption` [5, Sekce SUDOERS OPTIONS].

Schéma sudo pravidel také definuje atributy `sudoNotBefore` a `sudoNotAfter`, které je možno použít pro časově závislá pravidla. Pomocí těchto atributů je možné specifikovat, po jakou dobu bude pravidlo validní¹⁰. Tyto atributy ovšem aktuálně nejsou FreeIPA serverem používány a proto se jimi dále nebudeme zabývat.

Posledním definovaným atributem sudo pravidel je atribut `sudoOrder`. Ten představuje číselnou hodnotu, která je použita v situaci, kdy bude možno aplikovat více než jedno pravidlo. Vždy bude vybrán záznam s větší hodnotou tohoto atributu. To odráží chování „poslední shody“ lokálního sudoers souboru. Předpokládáme, že soubor `/etc/sudoers` obsahuje pouze následující dvě pravidla:

```
xsruba03 ALL=NOPASSWD: /sbin/blkid /dev/sda1
xsruba03 ALL=NOPASSWD: !/sbin/blkid /dev/sda1
```

⁹deprecated

¹⁰např. po dobu pracovní doby od 7:00 do 15:00

V tomto případě uživatel `xsruba03` nebude mít oprávnění spustit příkaz `blkid`. Jestliže ovšem uvedená pravidla prohodíme, pak uživatel **bude mít** oprávnění tento příkaz spustit. Pořadí, v kterém jsou SUDO pravidla definována, má tedy výrazný vliv na chování programu SUDO. LDAP protokol ovšem negarantuje pořadí, ve kterém jsou přijaty jednotlivé záznamy, tj. SUDO pravidla [6][3, Kapitola 3, Sekce Maintaining Order]. Vychází to z předpokladu, že jak záznamy, tak jednotlivé atributy jsou definovány jako množina. Proto bylo nutné definovat atribut, pomocí kterého je možné řadit jednotlivá pravidla.

Záznam ekvivalentního suda pravidla pro pravidlo `??` by v LDAP adresáři vypadal tak, jak jej popisuje obrázek 2.1.

Záznam sudo pravidla na LDAP serveru

```
dn: cn=rule1,ou=SUDOers,$DC
cn: rule1
description: Simple rule allowing user xsruba03 to run fdisk command.
sudoUser: xsruba03
sudoHost client.example.cz
sudoCommand: /sbin/fdisk
```

Obrázek 2.1: Příklad záznamu SUDO pravidla v LDAP adresáři.

Detailní popis nativního LDAP SUDO schématu je možné nalézt v manuálových stránkách `sudoers.ldap` nebo v oficiálním manuálu [4].

2.1.3 Integrace identit a politik

Pod pojmem *identita* je možno si představit objekt, u kterého nás zajímá jeho jednoznačná identifikace. V linuxovém prostředí se typicky jedná např. o uživatele nebo počítač. Většina administrátorů spravujících prostředí o desítkách a více uživatelů chce spravovat identity centrálně. LDAP adresáře jsou pro tento účel tedy velice využívány. Ve Windows prostředích je možno použít řešení *Active Directory*, *FreeIPA*¹¹ pak v linuxových prostředích. Na tyto servery je nativní LDAP SUDO schéma nahrát a začít tak SUDO pravidla spravovat centrálně. Přístup ke správě identit a bezpečnostních politik pomocí LDAP adresáře a nativního SUDO schématu má ovšem několik nevýhod:

- Nativní LDAP SUDO schéma je pevně dané a není možné jej modifikovat. Záznamy, které neodpovídají tomuto schématu SUDO neumí zpracovat.
- Pravidla je možno definovat pouze pomocí příkazového řádku, popř. pomocí souborů v LDIF¹² formátu.
- V případě, že se uživatel ocitne bez připojení k serveru, který spravuje SUDO pravidla, pak není možné SUDO použít (případně lze pracovat pouze s lokálními pravidly definovanými v souboru `/etc/sudoers`).

Jako největší nevýhodu lze ovšem považovat nulovou integraci mezi existujícími identitami v adresáři a bezpečnostními politikami. Jestliže se administrátor rozhodne spravovat

¹¹Free Identity, Policy and Audit

¹²LDAP Data Interchange Format

jak identity tak bezpečnostní politiky těchto identit centrálně, pak by chtěl, aby tyto entity byly vzájemně propojeny. Toho ovšem není možné, s použitím nativního LDAP SUDO schématu, dosáhnout. Při použití nativního schématu se při definici SUDO pravidel není možné odkazovat na již administrátorem vytvořené identity, tj. například uživatele nebo počítače. Při špatné konfiguraci zde také vzniká problém s překrytím uživatelů. Předpokládejme, že se na unixovém systému nachází uživatel `xsruba03`¹³ a zároveň je stejný uživatel definován i v LDAP adresáři, který administrátor používá pro správu uživatelů. Poté nastává problém zjistit, na kterého uživatele má být sudo pravidlo aplikováno. Toto je jeden z problémů, které se snaží řešit projekt FreeIPA, kterým se zabývá následující sekce.

2.2 FreeIPA

Kvůli efektivitě a jednoduché správě se IT administrátoři snaží spravovat identity centrálně a zároveň spojovat identity s autentizačními a autorizačními politikami. V linuxových prostředích existuje mnoho protokolů pro různé služby. Je možné například využít NIS¹⁴ nebo Kerberos+LDAP¹⁵ pro správu identit a jejich autentizaci nebo pouze LDAP pro správu sudo pravidel. Žádné z těchto služeb ovšem nejsou nijak propojeny a zároveň všechny musí být spravovány lokálně. Administrátor tedy musí mít potřebné znalosti všech těchto služeb a protokolů, aby je dokázal správně používat a dané služby spravovat.

Projekt FreeIPA¹⁶ se zaměřuje na správu identit a jejich politik. Je postaven nad existujícími nativními linuxovými aplikacemi a standardizovanými protokoly. Definuje doménu, ve které se vyskytují servery, klienti a vzájemně mezi sebou sdílí centrálně spravované služby. Poskytuje jakousi obálku nad standardizovanými síťovými službami, jako jsou PAM¹⁷, LDAP, Kerberos, DNS¹⁸, NTP¹⁹ nebo certifikační služby. Jeho hlavní úlohou je spravovat všechny tyto služby na jednom místě. IPA navíc umožňuje synchronizaci dat s Active Directory. Neumí ovšem Windows klienty spravovat.

Jako úložiště veškerých dat IPA používá LDAP adresář²⁰. Pro autentizaci je použit protokol Kerberos, který využívá symetrickou kryptografii pro generování *tiketů*, které jsou používány namísto klasických hesel. Jako certifikační autoritu, která zajišťuje vystavování certifikátů serverům, replikám nebo klientům, IPA používá *Dogtag Certificate System*²¹. Služby jako Kerberos jsou závislé na doménových jménech a časových razítkách. Proto IPA slouží také jako DNS a NTP server [2, Kapitola 1].

S ohledem na vytyčené cíle této bakalářské práce nás dále bude zajímat pouze správa SUDO politik. Tedy jakým způsobem IPA server SUDO pravidla ukládá a jak jsou zpracovány na straně klienta.

2.2.1 IPA sudo schéma

Sekce 2.1.3 popisovala problémy, které mohou nastat při použití nativního LDAP SUDO schématu z pohledu domény. Toto schéma sice může být nahráno na IPA server, ale nepo-

¹³je tedy definován v souboru `/etc/passwd`

¹⁴Network Information Service

¹⁵Lightweight Directory Access Protocol

¹⁶zkráceně jen IPA

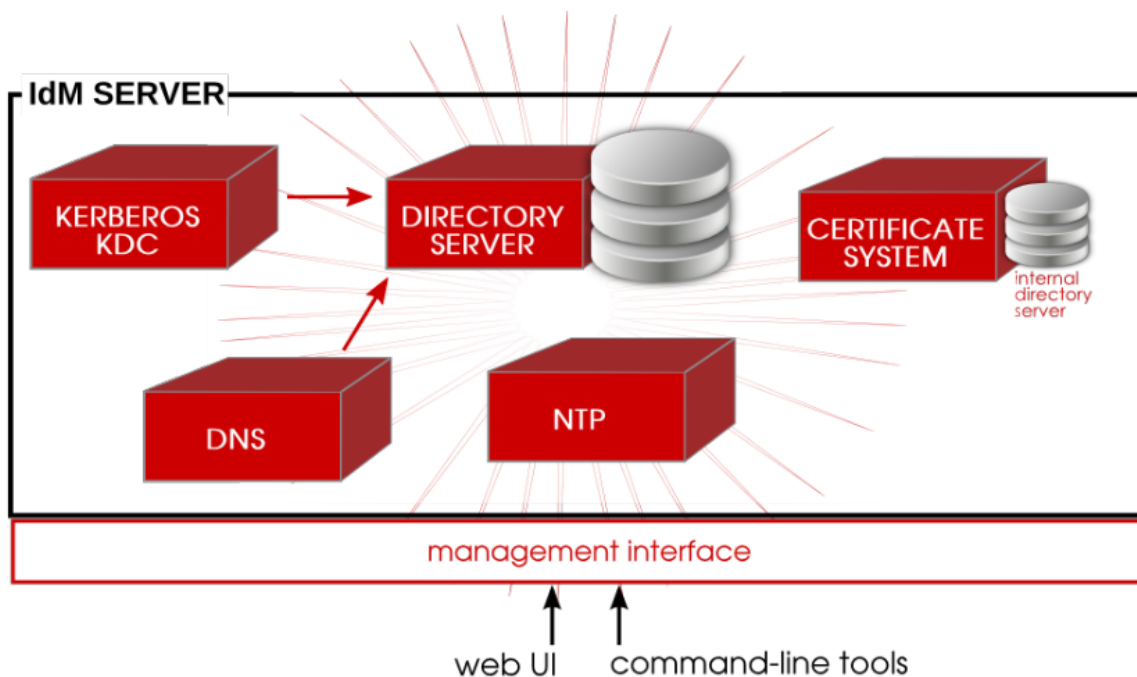
¹⁷Pluggable Authentication Modules

¹⁸Domain Name System

¹⁹Network Time Protocol

²⁰The 389 Directory Server, <http://directory.fedoraproject.org>

²¹http://pki.fedoraproject.org/wiki/PKI_Main_Page



Obrázek 2.2: FreeIPA sjednocuje standardní služby [2].

skytuje žádné propojení s již existujícími objekty dané domény. Z těchto důvodů se vývojáři projektu FreeIPA rozhodli vytvořit vlastní schéma pro ukládání SUDO pravidel. Nově navržené schéma umožňuje lepší integraci s již existujícími objekty s konkrétními pravidly. Umožňuje také daleko flexibilnější a jednodušší konfiguraci. Uživatel zde již dále není dále specifikován pouze jako řetězec znaků, ale jako odkaz na již existující objekt dané domény.

Pro vyjádření vztahu mezi jednotlivými záznamy se v adresářích obecně používají jedinečná jména záznamů adresáře [7, Kapitola 1]. Pro tyto jména bude dále používáno označení **DN**²². Jestliže se tedy atribut `user` odkazuje na záznam, s informacemi o uživateli, bude hodnota tohoto atributu tvořena právě DN daného záznamu. IPA SUDO schéma definuje novou třídu pro SUDO pravidla, jejichž definice vypadá následovně:

```
## Object class for SUDO rules
objectClasses: (
    2.16.840.1.113730.3.8.8.1
    NAME 'ipaSudoRule'
    SUP ipaAssociation
    STRUCTURAL
    MAY ( externalUser $ externalHost $ hostMask $ memberAllowCmd $
        memberDenyCmd $ cmdCategory $ ipaSudoOpt $ ipaSudoRunAs $
        ipaSudoRunAsExtUser $ ipaSudoRunAsUserCategory $
        ipaSudoRunAsGroup $ ipaSudoRunAsExtGroup $
        ipaSudoRunAsGroupCategory $ sudoNotBefore $
        sudoNotAfter $ sudoOrder
    )
    X-ORIGIN 'IPA v2'
```

²²Distinguished Name

```

)

objectClasses: (
  2.16.840.1.113730.3.8.4.6
  NAME 'ipaAssociation'
  ABSTRACT
  MUST ( ipaUniqueID $ cn )
  MAY ( memberUser $ userCategory $ memberHost $ hostCategory $
        ipaEnabledFlag $ description
  )
  X-ORIGIN 'IPA v2'
)

```

Třída `ipaSudoRule` dědí atributy třídy `ipaAssociation`. Každý záznam, popisující konkrétní SUDO pravidlo, obsahuje jedinečný identifikátor (`ipaUniqueID`), který je automaticky generován serverem FreeIPA, své jedinečné jméno (`cn`) a dále může obsahovat atributy specifikující samotné pravidlo.

Obecně v záznamech SUDO pravidel na IPA serveru platí, že `member*` atributy, např. `memberUser` nebo `memberAllowCmd`, obsahují DN již existujících objektů IPA domény. Pro objekty, které jsou definovány mimo IPA doménu, se používají `external*` atributy, např. `externalUser` nebo `externalHost`. Speciální případy, kdy je například SUDO pravidlo aplikovatelné na kteréhokoliv uživatele nebo počítač specifikují `*Category` atributy, např. `userCategory` nebo `cmdCategory`.

Uživatele, na kterého bude pravidlo aplikovatelné, je možné specifikovat pomocí tří atributů. Příklady hodnot těchto atributů popisuje tabulka 2.4. Jak je možné odvodit z prvního řádku této tabulky, záznamy uživatelů IPA domény se nacházejí v kontejneru `cn=users,cn=accounts,$DC`²³. Pomocí atributu `externalUser` je také možné vytvořit SUDO pravidlo aplikovatelné na uživatele, který není členem IPA domény. Tento uživatel ovšem musí být členem jiné domény²⁴ a mezi touto a IPA doménou musí být vytvořena vzájemná důvěra, tzv. *trust*. Popis, jak toto propojení mezi doménami funguje nebo jak jej vytvořit, je ovšem nad rámec této bakalářské práce.

Tabulka 2.4: Atributy specifikující uživatele SUDO pravidla.

Jméno atributu	Hodnota atributu
<code>memberUser</code>	<code>uid=xsruba03,cn=users,cn=accounts,\$DC</code>
<code>userCategory</code>	<code>all</code>
<code>externalUser</code>	<code>xsruba04</code>

Identitu, pod kterou bude příkaz spuštěn, je možné specifikovat pomocí `ipaSudoRunAs*` atributů. Zde je možné specifikovat jak konkrétní identitu, tak i skupinu identit.

Stejným způsobem je poté, pomocí atributů `memberHost`, `hostCategory` a `externalHost`, definován počítač, na který bude pravidlo aplikováno. Zde je navíc definován atribut `hostMask`, pomocí kterého je možné specifikovat počítač, popř. skupinu počítačů, pomocí IP adresy nebo adresy sítě. Tento atribut ovšem v praxi není využíván.

²³Domain Controller

²⁴například doména spravována pomocí Active Directory

Poslední množinou, kterou je nutno specifikovat pro vytvoření kompletního pravidla, jsou příkazy. Popsané schéma přináší novou vlastnost a tou jsou **skupiny příkazů**. Jak záznamy příkazů, tak záznamy skupin příkazů jsou v **samostatných** kontejnerech. Jsou také definovány pomocí jiných tříd. Atributy a příklad jejich možných hodnot popisuje tabulka 2.5.

Tabulka 2.5: Atributy specifikující SUDO příkaz.

Jméno atributu	Hodnota atributu
memberAllowCmd	cn=dics,cn=sudocmdgroups,cn=sudo,\$DC
memberDenyCmd	ipaUniqueID=41e9a39c,cn=sudocmds,cn=sudo,\$DC
cmdCategory	all

Kromě skupin příkazů přináší nové schéma další výhodu a to povolování/zakazování pravidel. Jestliže administrátor spravuje SUDO pravidla pomocí LDAP serveru, pak pravidla, která jsou v adresáři definována, budou vždy použita. Pokud by chtěl administrátor, například z důvodu testování, nějaké pravidlo dočasně zakázat, musel by jej z adresáře odstranit. Třída `ipaAssociation` definuje atribut `ipaEnableFlag`. Pomocí tohoto atributu je možné libovolné pravidlo dočasně povolit nebo zakázat bez nutnosti jeho odstranění z adresáře.

Sudo pravidla jsou tedy na IPA serveru uložena v kontejneru `cn=sudo,$DC`. Zde jsou ovšem dále dělena do dalších tří kontejnerů.

Tabulka 2.6: Rozmístění sudo pravidel na IPA serveru.

Obsah kontejneru	Kontejner
kompletní sudo pravidla	cn=sudo,\$DC
sudo pravidla	cn=sudorules,cn=sudo,\$DC
příkazy	cn=sudocmds,cn=sudo,\$DC
skupiny příkazů	cn=sudocmdgroups,cn=sudo,\$DC

Záznam ekvivalentního suda pravidla pro pravidla ?? a 2.1 v novém IPA SUDO schématu popisuje obrázek 2.3.

Každý záznam na IPA serveru má také **operační** atribut `entryUSN`²⁵. Hodnota tohoto atributu reprezentuje čas poslední změny záznamu. Aktualizaci této hodnoty provádí *USN plugin* při každé změně daného záznamu. Tento plugin poskytuje způsob, který umožňuje informovat IPA klienty o **modifikaci** jakéhokoliv záznamu [8, Sekce 3.4].

LDAP schémata, která používá FreeIPA server je na nainstalovaném IPA serveru možné nalézt v adresáři `/etc/dirsrv/slapp-instance_name/`. Detailní popis tříd a atributů pro definici SUDO pravidel na IPA serveru je možné nalézt v souboru `65sudo.ldif`. Schémata, která používá FreeIPA server, je možné najít také na přiloženém CD, viz Příloha A.

2.2.2 Definice SUDO pravidel

Další výhodou, kterou přináší FreeIPA, je snadnější definice pravidel. SUDO pravidla na IPA serveru je možné definovat dvěma způsoby a to pomocí **webového rozhraní** a jednak pomocí utilit příkazového řádku `ipa sudorule*`. Podrobnější popis s příklady, jak definovat

²⁵Entry Update Sequence Number

Záznam sudo pravidla na IPA serveru

```
dn: ipaUniqueID=e6917a9c,cn=sudorules,cn=sudo,$DC
cn: rule1
ipaEnabledFlag: TRUE
ipaUniqueID: e6917a9c
description: Simple rule allowing user xsruba03 to run fdisk command.
memberUser: uid=xsruba03,cn=users,cn=accounts,$DC
memberHost: fqdn=client.example.cz,cn=computers,cn=accounts,$DC
memberAllowCmd: ipaUniqueID=41e9a39c,cn=sudocmds,cn=sudo,$DC
```

Záznam sudo příkazu pro dané pravidlo

```
dn: ipaUniqueID=41e9a39c,cn=sudocmds,cn=sudo,$DC
description: Manipulate disk partiton table.
sudoCmd: /sbin/fdisk
ipaUniqueID: 41e9a39c
memberOf: cn=dics,cn=sudocmdgroups,cn=sudo,dc=example,$DC
```

Obrázek 2.3: Příklad záznamu SUDO pravidla na IPA serveru.

pravidla ve webovém rozhraní nebo pomocí příkazového řádku, je možné nalézt například v oficiální dokumentaci²⁶. Při správě pravidel v LDAP adresáři bylo nutné pravidla vytvářet textově a ručně je do adresáře přidávat. Webové rozhraní zde tedy administrátorovi velice zjednodušuje práci.

2.3 System Security Services Daemon

Jestliže se administrátor rozhodne spravovat identity a jejich politiky centralizovaně, např. pomocí technologií LDAP+Kerberos, musí provést konfiguraci NSS²⁷ a PAM a použít správné moduly pro tuto podporu. Například `nss_pam_ldapd` pro přístup k identitám a autentizacím na LDAP serveru. Nebo `pam_krb5` pro Kerberos autentizaci. Správná konfigurace může být pro běžného uživatele ovšem velice složitá. Navíc, i po správné konfiguraci, zde zůstávají důležité nedostatky tohoto řešení:

1. Jakmile je uživatelský počítač odpojen od sítě, pak se uživatel nemůže přihlásit. Uživatel to pak řeší separátním uživatelským účtem.
2. Jestliže se uživatel připojuje do více domén zároveň, každá může potencionálně používat jiný způsob autentizace a ověřování oproti jiným serverům. Bylo by tedy nutné provést novou konfiguraci.
3. Při použití mnoha technologií je potřeba provádět konfiguraci více souborů.

Tyto nedostatky řeší projekt SSSD, který poskytuje přístup k různým zdrojům, které spravují identity, jejich autentizaci a politiky. Identity a jejich autentizace tak může být spravována např. pomocí technologií LDAP+Kerberos, FreeIPA nebo Active Directory. Je také možné jej použít pouze pro přístup k SUDO pravidlům spravovaných centrálně

²⁶Red Hat Enterprise Linux 6 Identity Management Guide

²⁷Name Service Switch

pomocí LDAP adresáře. Další výhodou je, že umožňuje uživateli připojení do více domén zároveň. Jestliže klient střídá prostřední např. mezi prací a školou, nemusí stále měnit složitou konfiguraci. SSSD přináší také offline podporu a cachování. Informace o identitách nebo např. sudo pravidla jsou ukládána do cache paměti, která se nachází na disku počítače. Tyto informace je tedy možné využívat i v situacích, kdy se klient ocitne bez připojení k síti. To vše je možné nakonfigurovat pomocí jediného konfiguračního souboru `/etc/sss/sss.conf`.

Jestliže se administrátor rozhodne spravovat SUDO politiky pomocí IPA serveru, pak u všech klientů musí být démon SSSD dostupný. Použití SSSD pro přístup k SUDO politikám, uložených na vzdáleném serveru, má oproti přístupu pomocí ldap pluginu²⁸ následující hlavní výhody:

- **Offline podpora** – SSSD si ukládá všechna přijatá pravidla do své cache paměti. Jestliže dojde k odpojení klienta od sítě, pak může stále používat tyto uložená pravidla.
- **Lepší výkon** – Každé zavolání SUDO utility způsobí jeden nebo více LDAP dotazů na server, kde jsou daná pravidla uložena. V kombinaci s pomalou sítí nebo zatíženým serverem to může znamenat značné zpomalení v případech, kdy uživatel potřebuje použít program SUDO vícekrát za sebou. SSSD si uloží pravidla do cache paměti a při spuštění programu SUDO se pravidla hledají nejprve v cache paměti. Dalším zpomalením u ldap pluginu může být situace, kdy používá SUDO více uživatelů najednou. V takovém případě se vytváří několik LDAP spojení. Zatímco SSSD používá pouze jediné spojení s LDAP serverem

SSSD také podporuje nativní LDAP SUDO schéma. Je tedy možné jej nastavit tak, aby získával SUDO pravidla z LDAP serveru. V takovém případě získá uživatel jednak offline podporu, ale také možnost ukládání těchto pravidel do cache paměti. Konfigurace SSSD pro tento účel je popsána v manuálové stránce `sss-sudo`²⁹ a některé konfigurační volby také v `sss-ldap`³⁰.

2.3.1 Aktualizace SUDO pravidel v cache paměti

SSSD provádí následující tři typy aktualizací SUDO pravidel ve své cache paměti. Smyslem těchto aktualizací je zachovat přesnost pravidel a zároveň redukovat počet dotazů na server. SUDO, ve spojení s LDAP adresářem, pracuje vždy s aktuálními pravidly, protože se dotazuje LDAP adresáře při každém spuštění. SSSD se dotazuje LDAP serveru pouze v případě, kdy dané pravidlo v cache paměti již není validní. Aktualizace se poté snaží reflektovat stav SUDO pravidel na serveru v cache paměti SSSD.

S ohledem na vytyčené cíle této práce nás ovšem dále bude SSSD zajímat pouze z pohledu programu SUDO. Tedy jakým způsobem SSSD sudo pravidla zpracovává na straně klienta.

²⁸Jedná se o plugin, který dodává SUDO. Nezaměňovat tedy s ldap pluginem, který používá SUDO provider démona SSSD.

²⁹<http://jhrozek.fedorapeople.org/sss/1.11.5/man/sss-sudo.5.html>

³⁰<http://jhrozek.fedorapeople.org/sss/1.11.5/man/sss-ldap.5.html>

Kapitola 3

SSSD a SUDO provider

Smyslem této kapitoly je popsat vnitřní architekturu démona SSSD se zaměřením na SUDO providera a zásuvné moduly, které má k dispozici. Tento popis byl sestaven na základě experimentování a čtení zdrojových kódů. Získání zde popsaných informací bylo přitom nezbytné pro úplné pochopení způsobu, jakým SSSD SUDO pravidla zpracovává. V předposlední sekci je také popsán hlavní problém, který je v této práci řešen a to podpora IPA SUDO schéma v démonu SSSD.

3.1 Knihovny třetích stran

SSSD používá pro alokaci dynamické paměti hierarchický alokátor *talloc* [9] namísto standardního systémového volání *malloc*. Nepoužívá imperativní ani OOP¹ paradigma, ale naopak událostmi řízené paradigma² programování.

U imperativního nebo OOP paradigmatu programování se při provádění programu postupuje od shora dolů. To ovšem u událostmi řízeného paradigmatu neplatí. Programy reagují na události a tyto události poté určují, která část programu bude vykonávána. Takový program se poté nachází ve dvou stavech:

1. čeká na událost, nebo
2. provádí obsluhu této události.

Knihovna *tevent* [10] realizuje tyto události za pomoci asynchronních funkcí. Synchronní funkce poskytují programátorovi pouze jedno rozhraní, přes které je možno specifikovat vstupní parametry funkce a získat výsledek dané funkce. Naproti tomu asynchronní funkce poskytují rozhraní dvě. Jedno pro předání parametrů funkci a získání výsledků a druhé pro zaslání samotného požadavku a uvědomění programátora o dokončení této asynchronní události. Funkce, která je zavolána jako výsledek o dokončení události se nazývá tzv. *callback*.

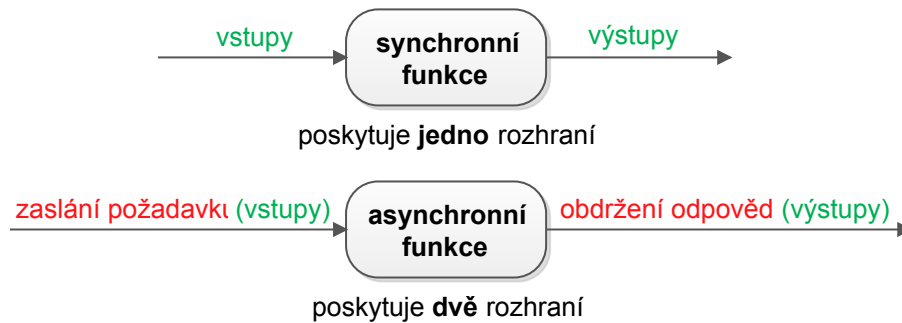
Knihovna *tevent* implementuje tyto asynchronní funkce pomocí tzv. *tevent requests*. Jejich grafické znázornění popisuje schéma 3.2. S tím je také spojena následující konvence:

- S událostí jsou spojeny privátní data, které mají příponu `_state`.
- Funkce, která provádí asynchronní volání má příponu `_send`.

¹Object-oriented programming

²Event-driven Programming Paradigm

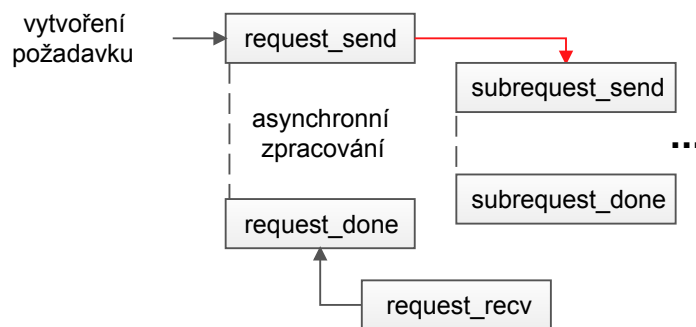
- Funkce, která bude vyvolána v okamžiku dokončení události, má příponu `_done`.
 - V této funkci je možné získat výsledky asynchronní operace pomocí volání funkce s příponou `*_recv`.



Obrázek 3.1: Rozhraní synchronních a asynchronních funkcí.

Tuto konvenci je vhodné striktně dodržovat, protože čitelnost asynchronního kódu je obecně složitější. Použití správného stylu zápisu kódu zlepšuje čitelnost a pomáhá programátorovi si lépe představit, v jakém pořadí se bude kód vykonávat.

Nejprve je danou událost nutno vytvořit a spojit s ní privátní data. Knihovna `tevent` pro alokaci dynamické paměti používá rovněž knihovnu `talloc`. Asynchronní funkci je možné zavolat pomocí funkce „`request_send`“³. Poté probíhá asynchronní provádění programu. V okamžiku kdy bude událost označena jako dokončená⁴, vyvolá se její callback tj. funkce „`request_done`“. V této funkci je možné zpracovat výsledky dané události a následně ji odstranit.



Obrázek 3.2: Příklad grafického zobrazení asynchronní funkce.

Volání asynchronních funkcí je možné do sebe také zanořovat. Jedna asynchronní operace tak může volat jinou. Jelikož jsou s každou událostí spojena privátní data, bylo by nutné tyto data při ukončení zanořené události, uvolnit. Privátními daty ovšem může být

³funkci je možné si pojmenovat libovolně, např. `ldap_query_send`

⁴např. pomocí funkce `tevent_req_done()`

komplexní struktura, potenciálně obsahující ukazatele na další struktury. Z tohoto důvodu používá knihovna `tevent` pro alokaci paměti rovněž alokátor `talloc`, který veškerá data zanořených asynchronních funkcí alokuje hierarchicky. Při dokončení první události, tj. té, která volala další asynchronní funkce, je poté najednou uvolněna veškerá přidělená paměť, tj. také veškerá paměť, která byla přidělena všem zanořeným událostem.

Knihovna `tevent` je také využívána ve spojení s knihovnami `OpenLDAP`⁵ a `DBus`⁶, které obaluje. Tyto knihovny byly obaleny právě proto, aby mohly být využity jako výše zmíněné „`tevent requests`“.

3.2 Architektura SSSD

SSSD se skládá z několika komponent. Hlavním procesem je zde *Monitor*, který je rodičem všech ostatních procesů. Stará se o to, aby ostatní procesy zůstaly aktivní, detekuje síťové změny a informuje o nich ostatní procesy.

Další komponentou jsou tzv. *responderi*. Jsou to procesy, které přijímají požadavky od klientských aplikací, jako jsou např. `sudo`, `id` nebo `getent`, a vracejí odpovědi na tyto požadavky. Smyslem těchto procesů je vrátit, za pomoci cache paměti, co nejrychleji odpověď na dotaz, který jím klientská aplikace zaslala. Proto by měly provádět co nejméně algoritmicky náročných operací a v podstatě pouze číst a předávat data z cache paměti SSSD. Komunikují tedy pouze s cache pamětí nebo poskytovali dat, ale nikdy nekomunikují se vzdáleným serverem. V následujícím textu bude pro každý takový proces používán výraz **responder**.

Veškerou práci spojenou s komunikací se vzdáleným serverem, zpracování výsledků a jejich uložení do cache paměti by měl zajistit konkrétní *provider*. Který provider bude použit pro kterou službu je možné nastavit v konfiguračním souboru `sssd.conf`. Například `id_provider` specifikuje, který provider SSSD použije pro zjišťování informací o identitách. Informace o SUDO pravidlech poté poskytuje SUDO provider. Každý provider může používat jiný modul. Například provider identit může použít *ldap plugin* pro přístup k identitám uloženým v LDAP adresáři, *ad plugin* pro identity spravované technologií Active Directory. Pro SUDO providera je možné použít pluginy *ldap* a *ipa*. Informaci o tom, které pluginy jsou konkrétními providery podporovány, je možné nalézt v manuálové stránce `sssd.conf`⁷. Jestliže mluvíme o LDAP SUDO providerovi, pak máme namysli *ldap plugin*, pomocí kterého přistupuje SUDO provider k SUDO politikám na vzdáleném serveru.

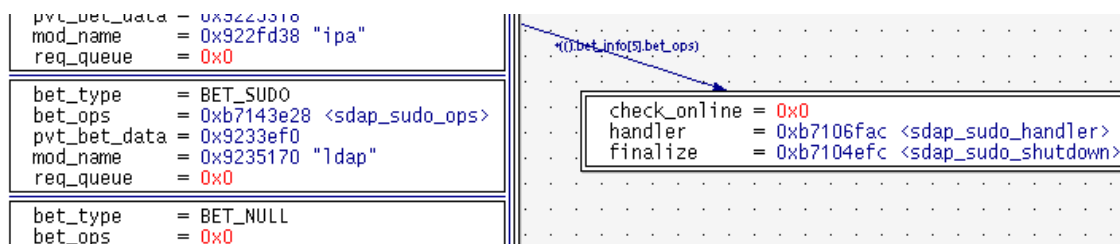
S pomocí SSSD může být jedna stanice připojena do více domén zároveň. Pro každou doménu je vytvořen samostatný proces, který danou doménu reprezentuje. Takový proces je nazýván *backend*. Backend používá providery pro komunikaci se vzdálenými servery.

Každý backend má ve své interní struktuře ukazatel na pole `bet_info`, které obsahuje informace o providerech, které daný backend používá. Každá položka tohoto pole je tvořena strukturou `bet_info`. Ta obsahuje např. jméno pluginu, který se má pro daný provider použít, ukazatel na privátní data nebo ukazatel na strukturu `bet_ops`, kde se nachází ukazatel na funkci, kterou volají responderi, pokud chtějí danému providerovi zaslat nějaký požadavek. Jméno této funkce má v SSSD obecně příponu *_handler* a takto bude označována i v následujícím textu. Položku pole `bet_info`, pro SUDO providera používající *ldap plugin*, zobrazuje obrázek 3.3.

⁵www.openldap.org

⁶dbus.freedesktop.org

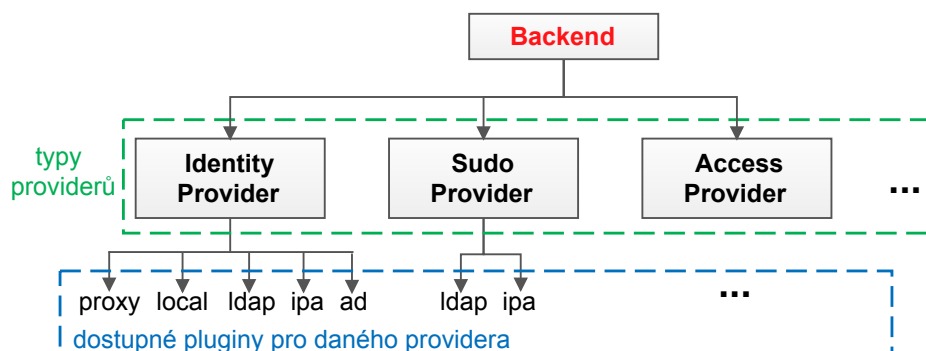
⁷<http://jhrozek.fedorapeople.org/sssd/1.11.5/man/sssd.conf.5.html>



Obrázek 3.3: Položka pole `bet_info` pro SUDO.

Každý backend také používá svou cache paměť. Do této paměti zapisuje pouze daný backend a responderi z ní mohou data pouze číst. Pro tuto cache je použita transakční LDB⁸ databáze, což je zjednodušená verze LDAP adresáře. I když tato databáze přesně neodpovídá LDAP standardu, disponuje podobným API⁹, které například při vyhledávání v této databázi umožňuje využít stejnou syntaxi filtrů jako u LDAP dotazů. Tato cache paměť se často označuje také jako *sysdb*. Toto označení bude také používáno v následujícím textu.

Responderi komunikují s providery skrze backend dané domény zasíláním SBUS zpráv. Jestliže responder nenajde hledaná data v *sysdb*, pak pošle konkrétnímu providerovi zprávu, aby je aktualizoval. Provider tento požadavek provede a upozorní respondera na to, že byla cache paměť aktualizována.



Obrázek 3.4: Typy providerů a jejich pluginů.

3.3 SSSD z pohledu programu SUDO

Předpokládejme, že jak informace o identitách, tak jejich autentizaci a bezpečnostních politikách (sudo pravidlech) jsou uloženy na vzdáleném serveru. Na klientském počítači je nainstalován démon SSSD. Uvažujme, že klient je členem jediné domény. Uživatel, s uživatelským jménem `xsruba03`, chce zobrazit obsah souboru `/etc/shadow`. K tomu ovšem potřebuje vyšší oprávnění a proto se rozhodne použít program SUDO. V tomto okamžiku musí SUDO provést následující tři úlohy:

⁸<http://ldb.samba.org/>

⁹Application programming interface

1. zjištění informací o uživatelském účtu,
2. autentizace daného uživatele,
3. získání sudo pravidel.

3.3.1 Zjištění informací o uživateli

Pro zjištění informací o uživatelském účtu použije SUDO funkce standardní knihovny jazyka C, tj. `getpwnend()`, popř. `getgrouplist()`. Tyto funkce poté volají NSS knihovnu. Ta je dynamicky načítána aplikacemi, které potřebují komunikovat s SSSD. Získávání informací o uživateli je velice častá operace, proto tato knihovna disponuje svojí cache pamětí, která se označuje jako tzv. *fast cache*. Tato cache paměť není součástí SSSD. Jestliže NSS ve své cache paměti daného uživatele nenajde, pošle dotaz, na tohoto uživatele, NSS responderu, což je jeden z procesů SSSD.

NSS responder se nejprve podívá do sysdb, jestliže zde daného uživatele nalezne, vrátí jej zpět knihovně NSS. V opačném případě pošle požadavek provideru identit, aby provedl aktualizaci cache paměti. Ten kontaktuje vzdálený server, aktualizuje cache paměť SSSD a upozorní o tom NSS respondera. Ten poté vrátí informace o uživateli NSS knihovně a ta je předá programu SUDO.

3.3.2 Autentizace uživatele

Jestliže má SUDO informace o uživateli, pak musí provést autentizaci tohoto uživatele. Autentizaci provádí knihovna PAM. Ta nemá vlastní cache paměť a posílá tak požadavek přímo PAM responderu. Ověření poté probíhá podobným způsobem jako u NSS respondera. PAM responder ovšem žádá o aktualizaci cache paměti providera, který zajišťuje autentizaci¹⁰. A navíc znovu provádí ověření daného uživatele oproti vzdálenému serveru.

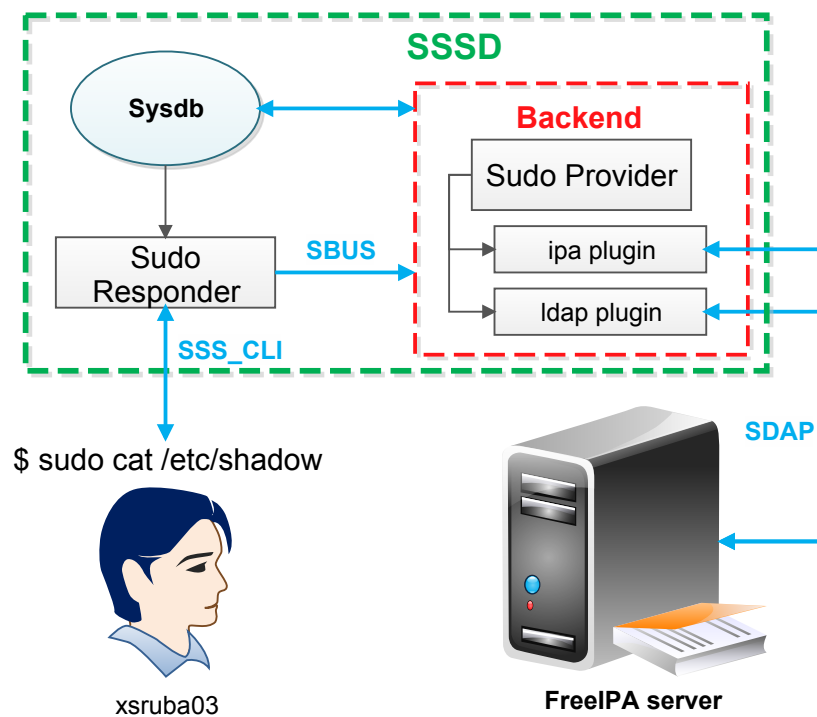
3.3.3 Získání sudo pravidel

Jestliže autentizace uživatele proběhla úspěšně, pak přichází na řadu SUDO responder. Ten nejprve hledá požadované pravidlo v cache paměti. Jestliže se v cache paměti nachází a je stále validní, pak nastává tzv. *cache hit*. V tomto případě je pravidlo vráceno programu SUDO.

Jestliže hledané pravidlo v cache paměti není validní, pak je provedena jeho aktualizace, tj. ověření zda se na serveru stále nachází. Proběhne také aktualizace všech ostatních pravidel, která již nejsou nadále validní. Aktualizaci sysdb provádí SUDO provider, který zajistí vyhledání potřebných pravidel na vzdáleném serveru, jejich zpracování a uložení zpět do sysdb. Po aktualizaci se SUDO responder opět dotáže cache paměti na požadované pravidlo. Tento průběh také zobrazuje schéma 3.5, které předpokládá, že uživatelské jméno již bylo ověřeno a proběhla také autentizace.

Jestliže, po druhém dotazu do cache paměti, není požadované pravidlo nalezeno, pak neprobíhá jeho další vyhledání na vzdáleném serveru. Pokud se ovšem dané pravidlo na serveru opravdu nachází, pak je nutné restartovat démona SSSD, nebo počkat na provedení některé z průběžných aktualizací, které jsou průběžně plánovány. Tuto funkcionalitu blíže popisuje sekce 3.4.2.

¹⁰v `sssd.conf` volba `auth_provider`



Obrázek 3.5: SUOD a SUDO responder.

3.4 LDAP SUDO Provider

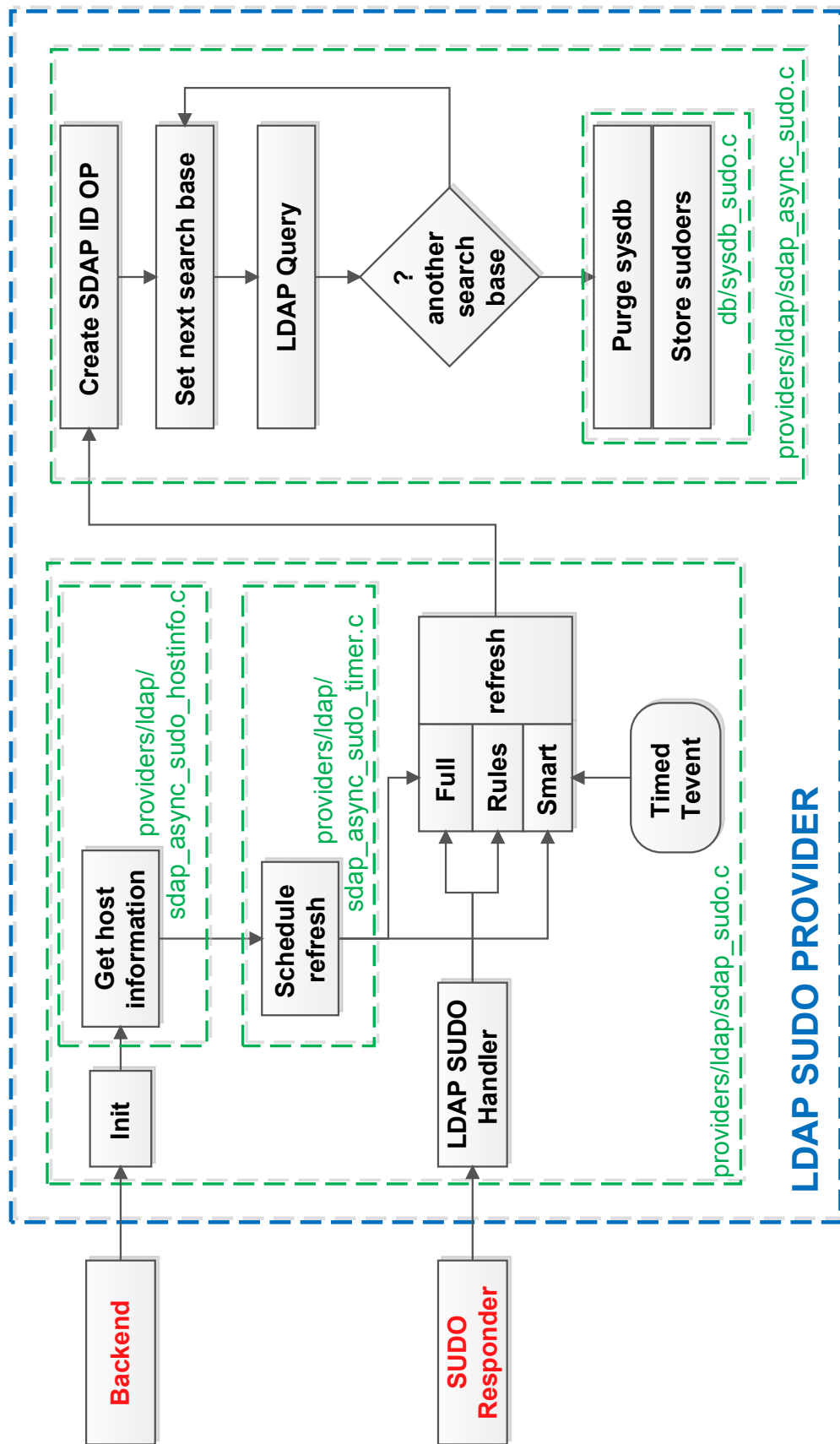
SUDO provider provádí veškerou práci při komunikaci s LDAP serverem. Pomocí volby `sudo_provider` v konfiguračním souboru `sssd.conf` je možno určit, který modul bude tento provider používat. Na výběr jsou moduly „ldap“ nebo „ipa“¹¹. Tato sekce popisuje funkcionality ldap modulu, který je nutno použít, máme-li sudo pravidla uložena na LDAP serveru v nativním LDAP SUDO schématu. Tento modul provádí stažení, zpracování a uložení těchto pravidel do sysdb. Skládá se z několika modulů, které jsou uvedeny na obrázku 3.6.

3.4.1 Inicializace ldap modulu

Inicializaci zajišťuje modul `providers/ldap/sdap_sudo.c` a probíhají při ní následující úlohy:

- alokace a inicializace privátních dat LDAP SUDO providera,
- nastavení handleru pro SUDO respodera, tj. vytvoření struktury `bet_ops`,
- získání všech jmen a IP adres daného počítače,
- nastavení plánování aktualizací SUDO pravidel.

¹¹`sudo_provider=ldap` nebo `sudo_provider=ipa`



Obrázek 3.6: Schéma popisující celkovou funkcionalitu modulů LDAP SUDO providera.

Inicializaci provádí funkce `sdap_sudo_init()`. Jména počítače, pro která budou stahována aplikovatelná SUDO pravidla, jsou získávána automaticky pomocí systémového volání `gethostname()`. Jestliže je již jméno počítače specifikováno jako FQDN¹², pak již další vyhledávání neprobíhá. V opačném případě se LDAP plugin pokusí získat FQDN pomocí asynchronního volání `resolv_gethostbyname_send()`. I když jsou IP adresy a jméno počítače získávána automaticky, je možné je specifikovat přímo v `sssd.conf` pomocí voleb `ldap_sudo_hostnames` a `ldap_sudo_ip`. Jejich hodnoty jsou poté zohledněny při sestavování filtrů pro LDAP dotazy. Použitím těchto voleb je poté možné stahovat i taková SUDO pravidla, která nejsou aplikovatelná na klientský počítač¹³.

3.4.2 Plánování aktualizací

Součástí inicializace je také nastavení plánování průběžných a úplných aktualizací. Jak často budou tyto aktualizace prováděny je možné ovlivnit v `sssd.conf`¹⁴.

Při spouštění SSSD by mohla nastat situace, kdy stihne uživatel spustit program SUDO ještě před provedením aktualizace. V takovém případě by SUDO nefungovalo správně, jelikož by se v sysdb nenacházela žádná pravidla.¹⁵ Proto je při každém spuštění SSSD provedena kompletní aktualizace SUDO pravidel.

Plánování aktualizací je realizováno pomocí časových událostí knihovny `tevent` a implementační detaily je možné nalézt ve funkci `sdap_sudo_setup_periodical_refresh()`.

3.4.3 Aktualizace SUDO pravidel

Všechny tři typy aktualizací využívají modul `providers/ldap/sdap_async_sudo.c`, který provádí samotné stažení a uložení SUDO pravidel do sysdb. Funkcionalitu tohoto modulu je možné ovlivnit pomocí parametrů, které jsou mu předány. Parametry mohou být například LDAP a SYSDB filtry, které se mají použít. Předpřipravení parametrů pro konkrétní aktualizace provádějí následující funkce:

- `sdap_sudo_full_refresh_send()` pro kompletní aktualizaci
- `sdap_sudo_smart_refresh_send()` pro průběžné aktualizace
- `sdap_sudo_rules_refresh_send()` pro aktualizaci pravidel

Úplná aktualizace

Tento typ aktualizace provede smazání všech pravidel z cache paměti a stažení všech pravidel ze serveru. SUDO provider pro ni používá LDAP filtr 3.1 Tento i všechny následující filtry se řídí syntaxí specifikovanou dokumentem RFC2254 [11]. Jsou pouze doplněny bílými znaky pro lepší čitelnost.

```
(&(objectClass=sudoRole)
(|(! (sudoHost=*))
(sudoHost=ALL)
(sudoHost=hostname.domain)
(sudoHost=hostname))
```

¹²Fully Qualified Domain Name

¹³Používáno převážně pro testování.

¹⁴Volby `ldap_sudo_full_refresh_interval` a `ldap_sudo_smart_refresh_interval`.

¹⁵za předpokladu, že se na serveru nějaká pravidla nacházejí

```

(sudoHost=IPv4)
(sudoHost=IPv4/netmask)
(sudoHost=IPv6)
(sudoHost=IPv6/netmask)
(sudoHost=+*)
(|(sudoHost=*\\*)
(sudoHost=*?*)
(sudoHost=*\**))
(sudoHost=*[*]*)
)
)
)

```

LDAP filtr 3.1: Filtr, který vyhledá sudo pravidla na serveru.

Tento filtr vyhledá všechna SUDO pravidla, která jsou aplikovatelná na klientský počítač. Jméno počítače (`hostname` a `hostname.domain`) a IP adresy jsou získány při inicializaci. Filtr také zachytí všechna pravidla aplikovatelná na jakoukoliv síťovou skupinu počítačů (`(sudoHost=+*)`), protože nemá informaci o tom, ve kterých síťových skupinách se počítač nachází.

Před uložením stažených pravidel jsou nejprve ze sysdb odstraněny **všechny** SUDO pravidla pomocí filtru 3.2. Až poté je zahájena transakce pro uložení nově stažených pravidel.

```
(objectClass=sudoRule)
```

SYSDB filtr 3.2: Filtr, který z cache paměti odstraní všechny sudo pravidla.

Úplná aktualizace se neprovádí velmi často, protože při velkém množství pravidel generuje velký síťový provoz. Provádí se tedy pouze při spuštění SSSD, při detekci změny pravidel na serveru nebo periodicky s menší frekvencí než průběžné aktualizace. Při zachování výchozího nastavení se tato aktualizace provádí každých 6 hodin. V manuálních stránkách a v kódu SSSD je možné tento typ aktualizace najít jako „*full refresh*“.

Průběžná aktualizace

Smyslem průběžných aktualizací je periodicky stahovat taková pravidla, která jsou na serveru nově přidána nebo byla od poslední aktualizace modifikována. Pro detekci modifikovaných pravidel se při průběžných aktualizacích využívá USN plugin a jeho `EntryUSN` atribut, viz. USN plugin v sekce 2.2.1.

Předpokládejme příklad kdy je na klientském počítači spuštěno SSSD a tento počítač má následující parametry:

```

Doménové jméno:  client1.example.cz
IPv4 adresa:      192.168.0.2
IPv6 adresa:      fe80::a00:27ff:fe71:1191

```

V sysdb již máme stažená nějaká pravidla a nejvyšší hodnota atributu `EntryUSN`, která se mezi pravidly nachází je **9270**. SUDO provider by vytvořil LDAP dotaz, který by použil filtr 3.3. Tento dotaz by vyhledal všechny pravidla, jejichž hodnota atributu `entryUSN` je větší než 9270.

```
(&(&(objectclass=sudoRole)
  (entryUSN>=9270)
  (!(entryUSN=9270))
)
(|(!(sudoHost=*))
  (sudoHost=ALL)
  (sudoHost=client1.example.cz)
  (sudoHost=client1)
  (sudoHost=192.168.0.2)
  (sudoHost=192.168.0.0/24)
  (sudoHost=fe80::a00:27ff:fe71:1191)
  (sudoHost=fe80::/64)
  (sudoHost=+*)
  (|(sudoHost=*\\\\\\*)
    (sudoHost=*?*)
    (sudoHost=*\\\\***)
    (sudoHost=*[*]*)
  )
)
)
```

LDAP filtr 3.3: Filtr, který vyhledá nově přidaná pravidla.

Při průběžných aktualizacích z cache paměti nejsou mazána žádná pravidla. Dochází pouze k přidávání pravidel, jestliže se na serveru vyskytují novější pravidla. Smyslem tohoto typu aktualizace je tedy stahovat pravidla, která mají na IPA serveru větší hodnoty `entryUSN` atributů než největší známá hodnota v `sysdb`.

V manuálových stránkách a v kódu SSSD je možné tento typ aktualizace vyhledat jako „*smart refresh*“.

Aktualizace pravidel

Tento typ aktualizace je proveden při každém spuštění programu SUDO. Zajistí aby uživatel nedostal větší oprávnění než je definováno. Jestliže se v `sysdb` při spuštění programu SUDO nachází nějaká pravidla, která již nejsou platná¹⁶, pak se provede jejich aktualizace. Předpokládejme příklad, kdy v `sysdb` již nejsou validní pravidla **rule1** a **rule2**. Nejprve se provede vyhledání těchto pravidel na serveru. Pro LDAP dotaz bude použit filtr 3.4. Jakmile budou potřebná pravidla stažena, provede se nejprve odstranění neplatných pravidel z cache paměti. Pro vyhledání těchto pravidel v `sysdb` použije SUDO provider filtr 3.5. Nově stažená pravidla poté budou uložena do `sysdb`.

```
(&(&(objectClass=sudoRole)
  (|(cn=rule1)
    (cn=rule2)
  )
)
)
```

¹⁶Čas po kterém se pravidla v cache paměti stanou neplatná, je možno nastavit v `sssd.conf` pomocí volby `entry_cache_sudo_timeout`, viz. manuálová stránka `sssd.conf`

```

(|(! (sudoHost=*))
  (sudoHost=ALL)
  (sudoHost=client1.example.cz)
  (sudoHost=client1)
  (sudoHost=192.168.0.2)
  (sudoHost=192.168.0.0/24)
  (sudoHost=fe80::a00:27ff:fea1:b983)
  (sudoHost=fe80::/64)
  (sudoHost=+*)
  (| (sudoHost=*\\*\\*\\*)
    (sudoHost=*?**)
    (sudoHost=*\\*\\***)
    (sudoHost=*[*]*)
  )
)
)
)

```

LDAP filtr 3.4: Filtr, který na serveru vyhledá neplatná pravidla z sysdb.

```

(&(objectClass=sudoRule)
  (|(cn=test4)
    (cn=test5)
  )
)

```

SYSDB filtr 3.5: Filtr, pro odstranění neplatných pravidel z cache paměti.

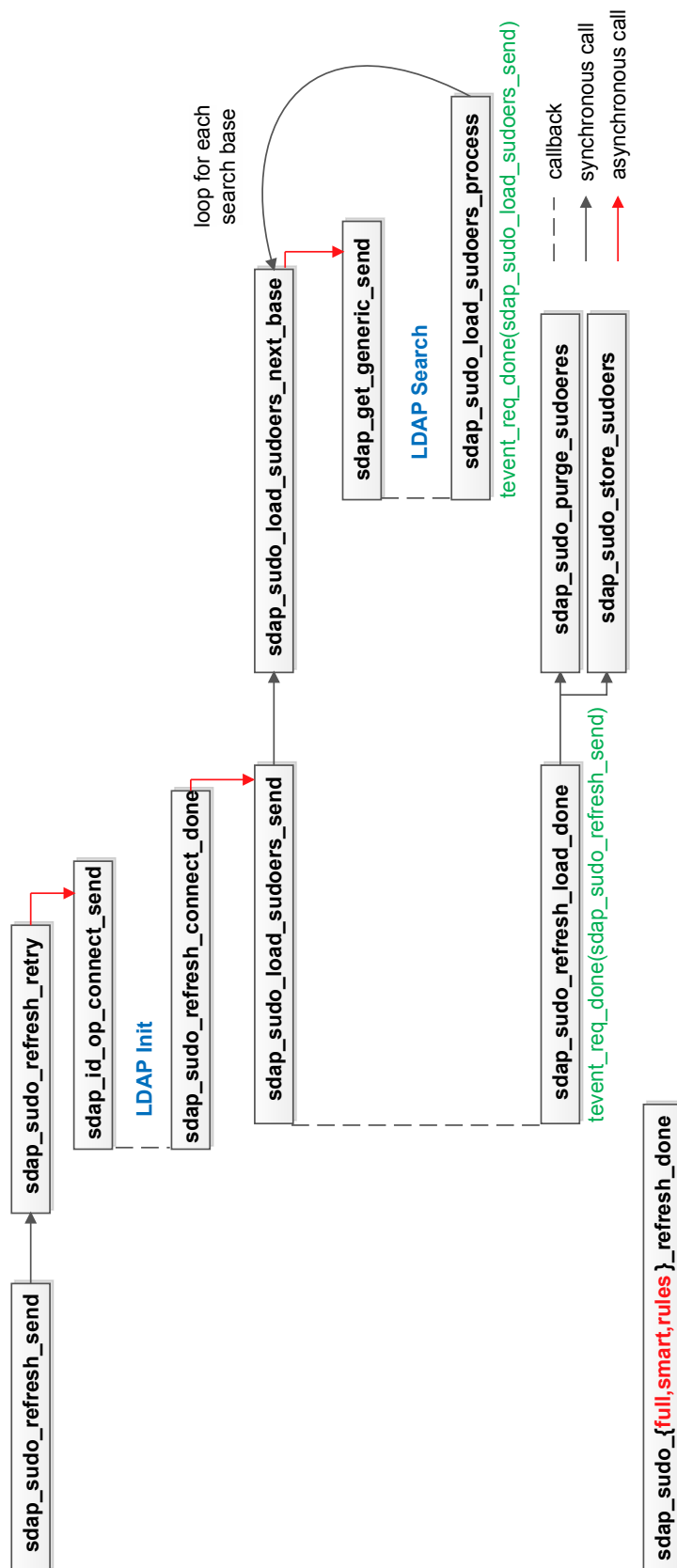
Nejprve by se provedlo vyhledání a stažení pravidel z LDAP serveru. Poté by došlo k vymazání pravidel `rule1` `rule2` ze sysdb a následně uložení nově stažených pravidel. Aktualizace pravidel, která jsou stále platná, se neprovede. Neprovede se ani aktualizace právě vyvolaného pravidla, za předpokladu, je stále platné.

V manuálních stránkách a v kódu SSSD se tento typ aktualizace označuje jako „*rules refresh*“.

3.4.4 Asynchronní modul

Modul `providers/ldap/sdap_async_sudo.c` je nejdůležitější částí LDAP SUDO pluginu, který provádí asynchronní stažení a uložení SUDO pravidel do sysdb. Jeho asynchronní činnost popisuje schéma 3.7. Asynchronní funkce toho modulu poté provádějí následující:

- `sdap_sudo_refresh_retry()` – Provádí kontrola zda je klient stále připojen k síti. Jestliže ano, pak proběhne inicializace LDAP spojení.
- `sdap_sudo_load_sudoers_send()` Zde se vytváří pole atributů pro LDAP dotaz.
- `sdap_sudo_load_sudoers_next_base()` Nastavuje aktuální search base, a posílá LDAP dotaz pomocí rozhraní `sdap_get_generic_send()`.
- `sdap_sudo_load_sudoers_process()` – Zajišťuje přijmutí a zpracování pravidel, tj. přidání pravidel k již staženým. Jakmile jsou všechna pravidla stažena, proběhne modifikace sysdb a poté je zahájena transakce při které se nově stažená pravidla uloží do sysdb.



Obrázek 3.7: Schéma událostí asynchronního modulu providers/ldap/sdap.async_sudo.c.

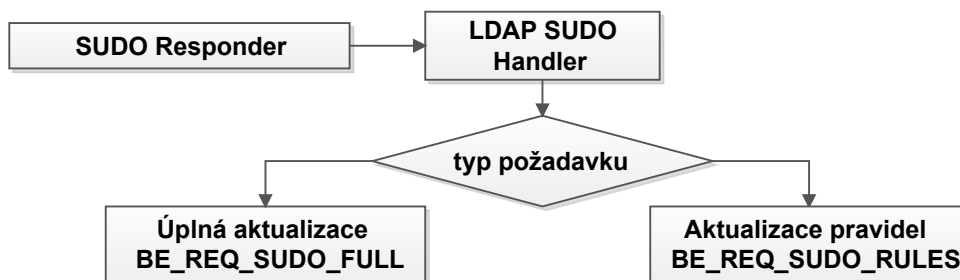
3.4.5 Uložení SUDO pravidel do sysdb

Před samotným uložením pravidel jsou v sysdb nejprve vyhledány a následně odstraněny všechny záznamy, které odpovídají SYSDB filtru pro konkrétní typ aktualizace. Při ukládání SUDO pravidel se každému pravidlu přidá atribut **dataExpireTimestamp**¹⁷. Jeho hodnota reprezentuje čas, kdy daný záznam stane neplatným. Zároveň je získána nejvyšší hodnota USN, která ovšem není uložena v cache paměti společně s pravidly, ale v kontextu `id_ctx->srv_opts` v proměnné `max_sudo_value`.

3.4.6 LDAP SUDO Handler

Jestliže SUDO responder nenajde daná pravidla v sysdb, pak zašle SBUS požadavek přes SUDO handler danému providerovi. Jestliže SUDO používá LDAP jako provider plugin¹⁹, pak se zavolá funkce `sdap_sudo_handler()`. Tato funkce, která se také označuje jako „Handler“ se nachází v interní struktuře backendu, viz. Sekce 3.3. Samotnou definici handleru je možné najít v modulu `sdap_sudo.c`.

LDAP SUDO provideru je přes handler předán typ aktualizace, který po něm SUDO responder žádá a pole pravidel, které chce responder aktualizovat²⁰. SUDO responder může požádat o úplnou aktualizaci nebo o aktualizaci pravidel, která již v sysdb nejsou nadále platná.



Obrázek 3.8: LDAP SUDO Handler.

3.5 IPA SUDO provider

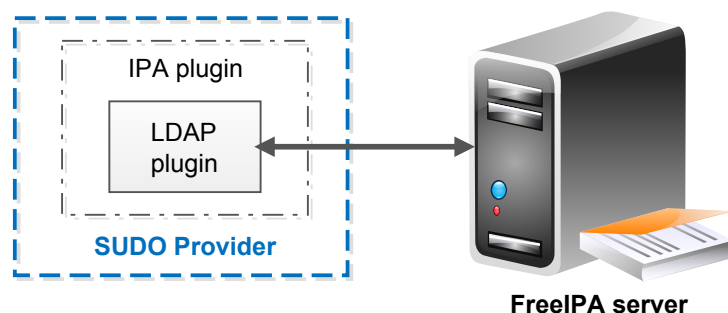
Pro SUDO providera je možné použít také ipa plugin²¹, jestliže jsou SUDO pravidla uložena na IPA serveru. Problém ovšem je, že SSSD nedisponuje IPA pluginem pro SUDO providera, který by podporoval IPA SUDO schéma. Současná implementace tohoto pluginu je pouze obálka pro LDAP SUDO plugin. Samotný LDAP plugin rovněž nepodporuje IPA SUDO schéma a neumí tedy SUDO pravidla v tomto formátu zpracovat. LDAP plugin umí zpracovat pouze pravidla, která odpovídají nativnímu LDAP SUDO schématu.

¹⁷ aktuální čas (v době ukládání) + hodnota volby `entry_cache_sudo_timeout`, viz. `man sssd.conf`¹⁸

¹⁹ `sudo_provider=ldap`

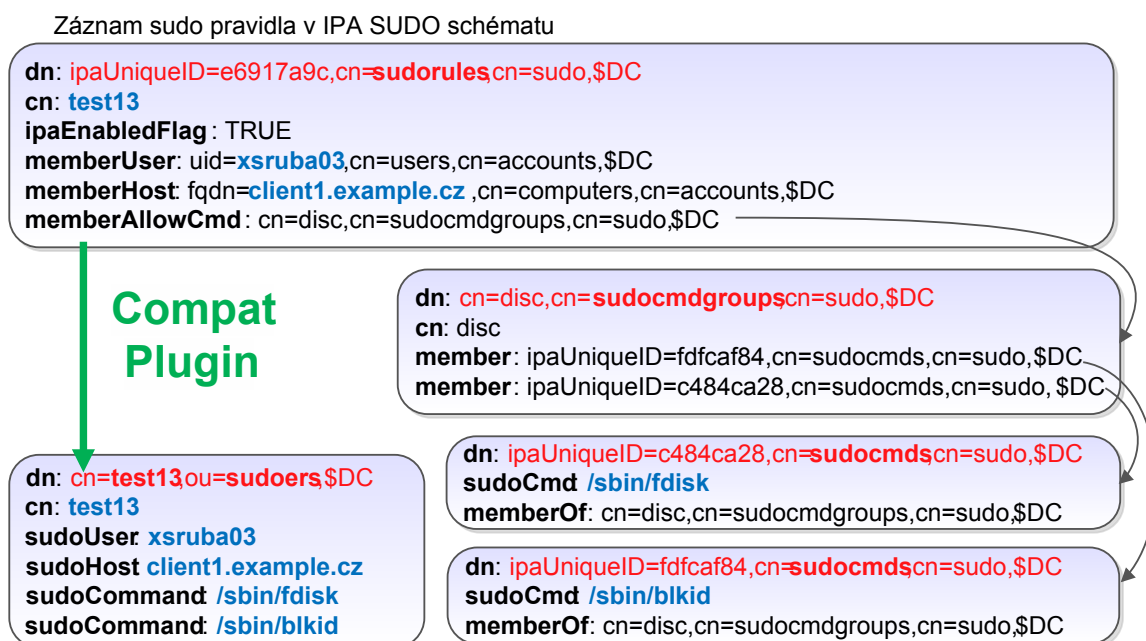
²⁰ pole jmen pravidel, tj. pole hodnot `cn` atributů

²¹ `sudo_provider=ipa`



Obrázek 3.9: Současná implementace IPA SUDO pluginu.

Pro podporu SUDO pravidel v IPA SUDO formátu využívá současná implementace SSSD *compat plugin*. Tento plugin běží na serveru FreeIPA a zajišťuje překlad SUDO pravidel z IPA SUDO schématu do LDAP SUDO schématu. Funkcionalitu pluginu demonstruje schéma 3.10. Překlad samotný probíhá při každém vytvoření nebo modifikaci SUDO pravidla. Takto přeložená pravidla se poté na serveru nacházejí v kontejneru `ou=SUDOers,$DC`.



Obrázek 3.10: Překlad SUDO pravidel provádí na IPA serveru compat plugin.

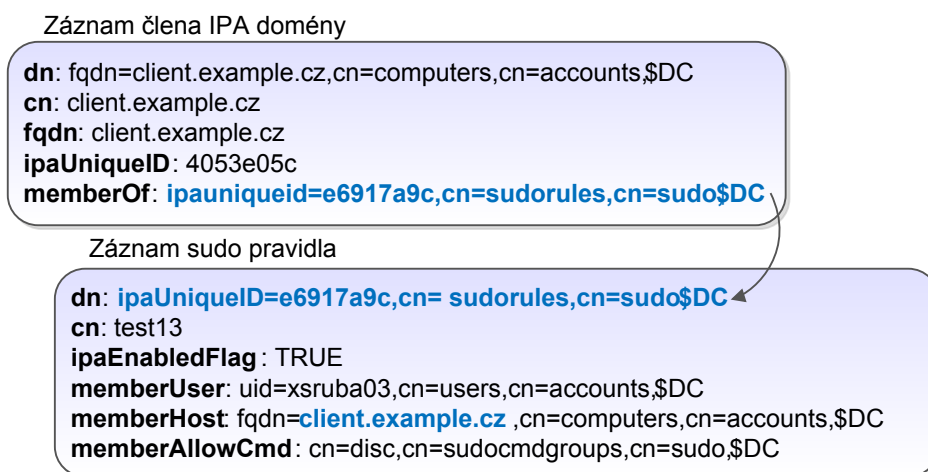
SUDO pravidla jsou tedy na serveru uložena dvakrát, i když v jiných schématech. Za předpokladu, že by SSSD mělo podporu pro nativní IPA SUDO schéma, byla by odstraněna režie compat pluginu. To znamená, že by nadále nebylo nutné provádět překlad **všech** SUDO pravidel. Překlad by byl přenesen ze serveru na klienta. To by znamenalo překlad pouze těch pravidel, která jsou aplikovatelná pro daného klienta.

Smyslem této bakalářské práce je odstranění této režie a implementace IPA SUDO

pluginu, který bude zajišťovat podporu nativního IPA SUDO schématu. Návrhem nativního IPA SUDO providera se zabývá kapitola 4.

3.6 Dereferenční dotaz

LDAP dotaz umožňuje získat záznam, popřípadě množinu záznamů LDAP adresáře. Jestliže se nějaký atribut z těchto záznamů odkazuje na jiné záznamy (hodnota některého z atributů obsahuje DN jiného záznamu v adresáři). Pak pro získání těchto záznamů je nutné vytvořit nový dotaz.



Obrázek 3.11: Příklad záznamu, jehož atributy se odkazují na jiný záznam.

Chování API knihovny OpenLDAP je možné měnit pomocí tzv. „controls“. Je tedy možné modifikovat chování funkce `ldap_search_ext()`. Pak bude možné pomocí této funkce získat nejenom požadované záznamy, ale také atributy ze záznamů, na které se daný záznam odkazuje. SDAP rozhraní využívá této funkcionality a nabízí programátorovi asynchronní funkci `sdap_deref_search_send()`. Tato situace je zachycena na obrázku 3.11.

Server ovšem musí tento typ „controls“ podporovat²². U Active Directory se tato technika označuje jako *Attribute Scope Queries*.²³

²²supportedControl: 1.3.6.1.4.1.4203.666.5.16

²³<http://msdn.microsoft.com/en-us/library/aa746418%28v=vs.85%29.aspx>

Kapitola 4

Nativní IPA SUDO Provider

Tato kapitola se zabývá diskusí nad řešením hlavních problémů, které je nutno zvážit při návrhu nativního IPA SUDO providera pro démona SSSD. Sekce implementace popisuje, jak a které z těchto přístupů byly vybrány pro implementaci nového providera. V sekci testování je poté popsána metodika a nástroje využité k otestování navržených řešení.

4.1 Návrh

Cílem této sekce je popsat možné přístupy ke stažení a zpracování SUDO pravidel, jejichž výhody a nevýhody byly diskutovány s vývojáři LDAP SUDO providera. Některé specifické detaily byly diskutovány také s vývojáři projektu FreeIPA.

4.1.1 Správný přístup k získávání SUDO pravidel

Jedním z problémů, které je nutno vyřešit, je správný přístup k tomu, jakým způsobem získat kompletní SUDO pravidla pro daný počítač a zároveň nestahovat žádná pravidla ani jiné nepotřebné informace navíc. LDAP SUDO provider při úplné aktualizaci stahuje například i všechna pravidla, která jsou aplikovatelná na libovolnou síťovou skupinu počítačů. Ke stahování pravidel z IPA serveru lze přistoupit způsoby popsány níže.

Klient jako člen IPA domény

První možností je využít záznam, obsahující informace o členu IPA domény. Každý člen má na IPA serveru záznam v kontejneru `cn=computers,cn=accounts,$DC`. Každý takový záznam obsahuje například atribut `memberOf`, kde se nacházejí odkazy na záznamy skupin počítačů, síťových skupin a SUDO nebo HBAC¹ pravidel, kterými je daný klient členem. Bylo by tedy možné jedním dotazem získat odkazy na SUDO pravidla, která jsou aplikovatelná na daný klientský počítač a dalšími dotazy už konkrétní záznamy SUDO pravidel. Jestliže bychom chtěli získat DNS² SUDO pravidel aplikovatelných na klientský počítač, jehož doménové jméno je `client.example.cz`. Příslušný LDAP dotaz by mohl mít následující parametry:

- Prohledávaná oblast: `fqdn=client.example.cz,cn=computers,cn=accounts,$DC`,
- Filtr: `"(memberOf=ipauniqueid=*cn=sudorules,cn=sudo,$DC)"`,

¹Host-Based Access Control

²Distinguished Names

- Dotazované atributy: `memberOf`.

S tímto přístupem jsou ovšem spojeny dva problémy. V `memberOf` atributu záznamu pro klienta se nenachází SUDO pravidla aplikovatelná na kterýkoliv počítač. To znamená pravidla, která mají `hostCategory` atribut a pravidla by tedy bylo nutné dodatečně stáhnout. Druhým problémem jsou skupiny počítačů. Jestliže je SUDO pravidlo aplikovatelné na nějakou skupinu počítačů a klient je zároveň členem této skupiny, pak je SUDO pravidlo na daného klienta aplikovatelné. Odkaz na toto pravidlo se ovšem nenachází v záznamu daného počítače, ale v záznamu dané skupiny. Odkazy na tyto pravidla by bylo nutné získat pomocí dereference odkazů³ na dané záznamy skupiny počítačů.

Specifikace klientského počítače

Druhou možností, jak získat kompletní SUDO pravidla, je využít stejný přístup, jako používá současná implementace LDAP SUDO providera. Pro výběr SUDO pravidla aplikovatelného na konkrétní počítač v IPA doméně jsou v tomto případě použity tři atributy:

- `memberHost`,
- `externalHost`,
- `hostCategory`.

Za předpokladu, že je SUDO pravidlo na IPA server přidáno korektně, tj. pomocí webového rozhraní nebo utilit `ipa-sudorule-add-*`, pak atribut `memberHost` bude vždy obsahovat plně specifikované doménové jméno u počítače nebo název skupiny počítačů. Jestliže je pravidlo aplikovatelné na libovolný počítač, pak bude mít atribut `hostCategory` hodnotu `all` a jiné atributy specifikující počítač se již v záznamu pravidla neobjeví. Parametry LDAP dotazu, pomocí kterého bychom získali záznamy všech pravidel aplikovatelných pro klientský počítač s doménovým jménem `client.example.cz`, mohou vypadat následovně:

- Prohledávaná oblast: `cn=sudorules,cn=sudo,dc=example,dc=cz`,
- Filtr: `"(|(memberHost=client.example.cz)(hostCategory=all))"`,
- Dotazované atributy: `memberOf`.

Problémem tohoto přístupu je fakt, že SUDO provider nemá informaci o tom, ve kterých skupinách počítačů se daný počítač nachází. Tato informace je dostupná pouze na IPA serveru. Tento problém by bylo možné odstranit dvěma způsoby:

1. Stažením SUDO pravidel aplikovatelných pro libovolnou skupinu počítačů. V případě rozsáhlé databáze pravidel a mnoho skupin počítačů by to ovšem mohlo způsobit značný síťový provoz navíc.
2. Nejprve získat seznam všech skupin, kterými je daný klient členem, a až poté sestavit LDAP dotaz pro stažení SUDO pravidel.

Pro implementaci byl vybrán právě druhý uvedený přístup. Detaily jsou popsány v úvodu sekce 4.2.

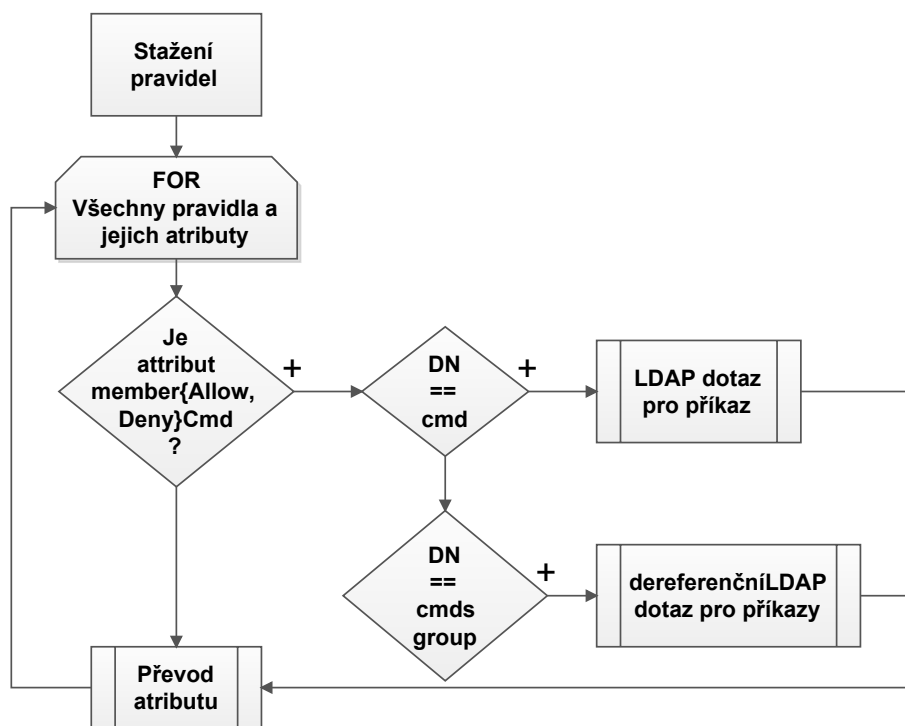
³Tato technika je blíže popsána v sekci 3.6

4.1.2 Stažení SUDO pravidel

Jestliže víme, jak specifikovat SUDO pravidla aplikovatelná na klientský počítač, další krokem je samotné stažení těchto pravidel z IPA serveru. LDAP SUDO provider stahuje SUDO pravidla pomocí jediného LDAP dotazu, protože kompletní záznamy SUDO pravidel se nacházejí většinou v kontejneru `ou=SUDOers,$DC`. Prohledávanou oblast je možné specifikovat v `sssd.conf` pomocí volby `ldap_sudo_search_base`. SUDO pravidla na IPA serveru jsou ovšem rozmístěna ve třech kontejnerech, viz Sekce 2.2.1. Pro stažení kompletních SUDO pravidel se nabízí několik následujících přístupů.

1. Postupné dotazování

První přístup, který se nabízí a který je popsán na obrázku 4.1, je stažení všech pravidel, aplikovatelných na daný počítač, z kontejneru `cn=sudorules,cn=sudo,$DC`. Pro tyto záznamy je ovšem nezbytné ještě stáhnout příkazy. Bylo by tedy nutné iterovat přes stažené záznamy a pro každý odkaz na SUDO příkaz provést LDAP dotaz, jehož výsledkem by již byl konkrétní příkaz. Pokud by se jednalo o skupinu příkazů, pak by bylo nutné provést dereferenční dotaz, jehož výsledkem by byl záznam, kde by se nacházel již konkrétní příkaz.



Obrázek 4.1: Stažení kompletních SUDO pravidel postupným dotazováním.

Tento přístup má tyto výhody:

1. Zpracování pravidel, které představuje překlad pravidel z IPA schématu do nativního LDAP schématu, viz Sekce 4.1.3, je možné provést v jediném cyklu.
2. Nejsou stahována žádná přebytečná SUDO pravidla, příkazy nebo jiné záznamy.

Při rozsáhlé databázi SUDO pravidel by ovšem dotazování na všechny příkazy mohlo znamenat velké množství LDAP dotazů. Pravidla by navíc mezi těmito dotazy mohla být modifikována.

2. Další dotaz pro příkazy

Jiným přístupem, který se nabízí, je stažení všech záznamů o SUDO pravidlech aplikovatelných na klientský počítač. Tyto stažené záznamy poté projít a sestavit filtr pro LDAP dotaz, který stáhne všechny potřebné příkazy pro stažená pravidla. Příklad vytvoření takového filtru ukazuje obrázek 4.2.

Příklad staženého sudo záznamu z IPA serveru

```
dn: ipaUniqueID=7f9192b2,cn=sudorules,cn=sudo,$DC
...
memberDenyCmd: cn=group,cn=sudocmdgroups,cn=sudo,$DC
memberAllowCmd: cn=user,cn=sudocmdgroups,cn=sudo,$DC
memberAllowCmd: ipaUniqueID=6f545188,cn=sudocmds,cn=sudo,$DC
...
```

```
(&(objectClass=ipaSudoCmd)
  (|(memberOf=cn=group,cn=sudocmdgroups,cn=sudo,$DC)
    (memberOf=cn=user,cn=sudocmdgroups,cn=sudo,$DC)
    (ipaUniqueID=6f545188))
)
```

Obrázek 4.2: Sestavení LDAP filtru pro SUDO příkazy.

Záznamy o skupinách SUDO příkazů není nutné stahovat. Tyto skupiny totiž není možné do sebe zanořovat. Každý SUDO příkaz má atribut `memberOf`, který obsahuje DN skupin příkazů, kterých je členem. Jestliže se `memberAllowCmd` nebo `memberDenyCmd` odkazuje na skupinu SUDO příkazů, pak je možné dané příkazy na základě `memberOf` atributu zjistit.

Výhodnou tohoto přístupu je, že redukuje počet LDAP dotazů nutných pro stažení kompletních SUDO pravidel a zároveň nestahuje žádné přebytečné informace. Ve výsledku by tak stačil jeden dotaz pro získání SUDO pravidel a druhý pro stažení příkazů, které jsou potřebné pro tato pravidla.

3. Stažení všech příkazů

Množina SUDO příkazů je obecně malá. Bylo by tedy možné v jediném LDAP dotazu stáhnout SUDO pravidla aplikovatelná na klientský počítač a zároveň **veškeré** SUDO příkazy.

Výhodou tohoto přístupu je, že pro stažení kompletních SUDO pravidel postačí jediný LDAP dotaz. Jejich následný překlad je možné provést v jednom cyklu. Může ovšem nastat situace, kdy se na klienta bude aplikovat např. pouze jedno SUDO pravidlo. Jestliže se na serveru nachází velké množství SUDO pravidel a jejich SUDO příkazů, pak by to znamenalo

přenášení spousty záznamů, které nebudou použity. Bylo by je ovšem nutné zpracovat, což přináší výkonnostní ztráty.

Přístup, který byl vybrán pro implementaci a důvody tohoto výběru, jsou dále popsány v sekci 4.2.

4.1.3 Překlad pravidel

Jelikož SUDO dokáže zpracovat pouze pravidla, která odpovídají nativnímu LDAP SUDO schématu, je nutné provést překlad pravidel, která jsou uložena na IPA serveru v IPA SUDO schématu. Jak bylo popsáno v sekci 3.10, překlad *všech* sudo pravidel v současné době realizuje compat plugin a je prováděn **na serveru** FreeIPA. S podporou nativního IPA SUDO schématu se tento překlad přesune na stranu SSSD a bude tedy probíhat **u klienta**. Na straně klienta se již nebudou překládat veškerá pravidla, ale pouze taková pravidla, která jsou aplikovatelná na klientský počítač.

Některé atributy je možné přeložit pouhým zkopírováním hodnoty a záměnou jména atributu. Mezi tyto atributy patří například: `cn`, `externalUser` nebo `ipaSudoOpt`. Atributy, jako jsou například `memberHost` nebo `memberUser` obsahují DN záznamu daného počítače, popř. uživatele. Jelikož projekt FreeIPA garantuje použitá schémata, je možné dané hodnoty z DN odvodit. Atribut `memberUser`, který na IPA serveru specifikuje uživatele nebo skupinu, by např. mohl mít hodnotu: `memberUser: cn=students,cn=groups,cn=accounts,$DC`. Z této hodnoty je možné odvodit, že je pravidlo aplikovatelné na všechny uživatele skupiny „students“. U `*Category` atributů je nutné provést pouze záměnu malých písmen za velká.

Pro atributy `memberAllowCmd` a `memberDenyCmd` je nutné vyhledat potřebný příkaz, popř. skupinu příkazů, jedná-li se o odkaz na skupinu příkazů. Pro všechny atributy, které IPA SUDO schéma definuje, je tedy nutné určit, jakým způsobem bude přeložena jejich hodnota. Atributy lze rozdělit na několik typů. Jakým způsobem je nutné který typ přeložit popisuje tabulka 4.1. Mohlo by se zdát, že typy `USER.GROUP` a `HOST.GROUP` není nutné rozlišovat, ovšem liší se v prefixu, který je k nové hodnotě nutno přidat: Viz příklady hodnot atributů `sudoUser` a `sudoHost`, které zobrazují tabulky 2.1 a 2.2.

Tabulka 4.1: Typy atributů a způsob jejich překladu.

Typ atributu	Způsobe překladu hodnoty
USER	DN (hodnota kontejneru uid)
USER.GROUP	DN (hodnota kontejneru cn)
HOST	DN (hodnota kontejneru fqdn)
HOST.GROUP	DN (hodnota kontejneru cn)
COMMAND	DN (hodnota kontejneru ipaUniqueID je použita pro filtr)
CMDS.GROUP	hodnota je použita pro filtr pro příkazy
UPPER_CASE	převedení malých písmen hodnoty na velká
COPY	kopie původní hodnoty

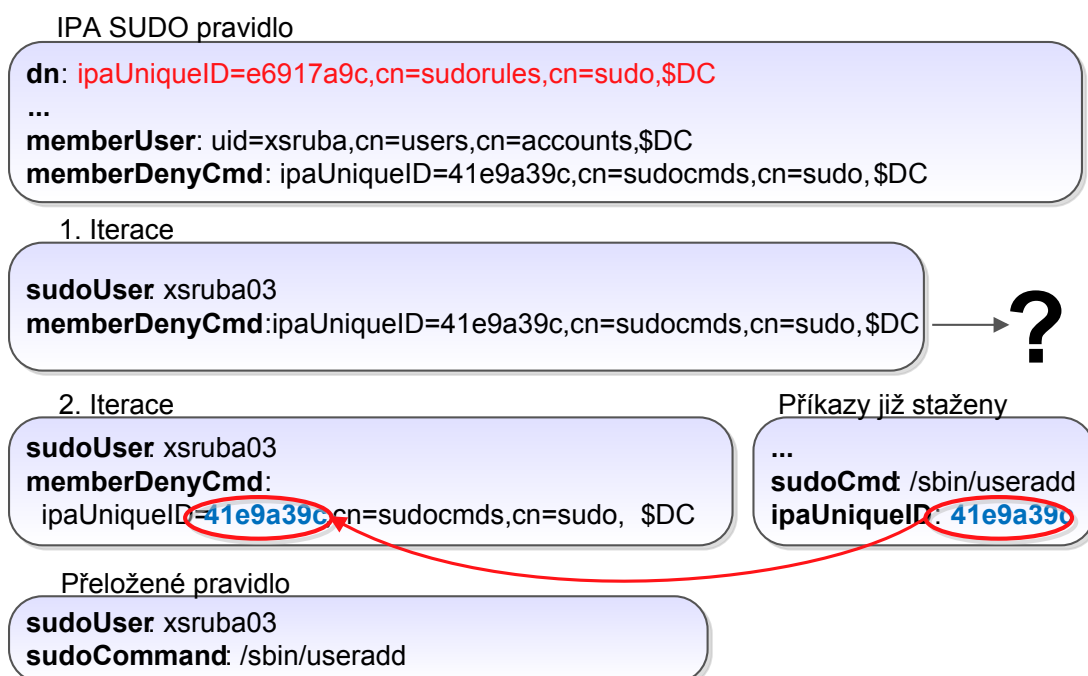
Pro algoritmus překladu pravidel byla vytvořena překladová tabulka, která definuje, jakým způsobem má být proveden překlad jednotlivých jmen a hodnot atributů včetně příkladů, viz Příloha A. Tato tabulka byla vytvořena na základě implementace compat pluginu a byla diskutována s vývojáři projektu FreeIPA. Při jejím sestavování byla také objevena chyba, které se vyskytuje ve webovém rozhraní serveru FreeIPA. Ta umožňuje atributu

`ipaSudoRunAsGroup` přiřadit libovolnou skupinu. Tento atribut ovšem může obsahovat odkaz pouze na POSIX skupinu. Pro tuto chybu byl vytvořen ticket číslo 4314⁴.

Algoritmus překladu pravidel

Uvažujeme variantu stahování SUDO pravidel popsanou v sekci 4.1.2, tj. stažení pravidel pomocí dvou LDAP dotazů. Po prvním dotazu získáme množinu záznamů SUDO pravidel. Druhý dotaz vrátí množinu záznamů SUDO příkazů pro stažená pravidla.

Pro získání filtru, potřebného pro dotaz pro SUDO příkazy, je nutné nejméně jednou iterovat přes množinu stažených SUDO pravidel. Při této iteraci je již možné provést překlad všech atributů kromě těch, které se odkazují na příkazy, tj. `memberAllowCmd` a `memberDenyCmd`. Jakmile obdržíme SUDO příkazy, můžeme provést druhou iteraci množinou SUDO pravidel, při které doplníme potřebné příkazy. Tento postup je zachycen na obrázku 4.3.



Obrázek 4.3: Dvoupřůchodový překlad IPA SUDO pravidel

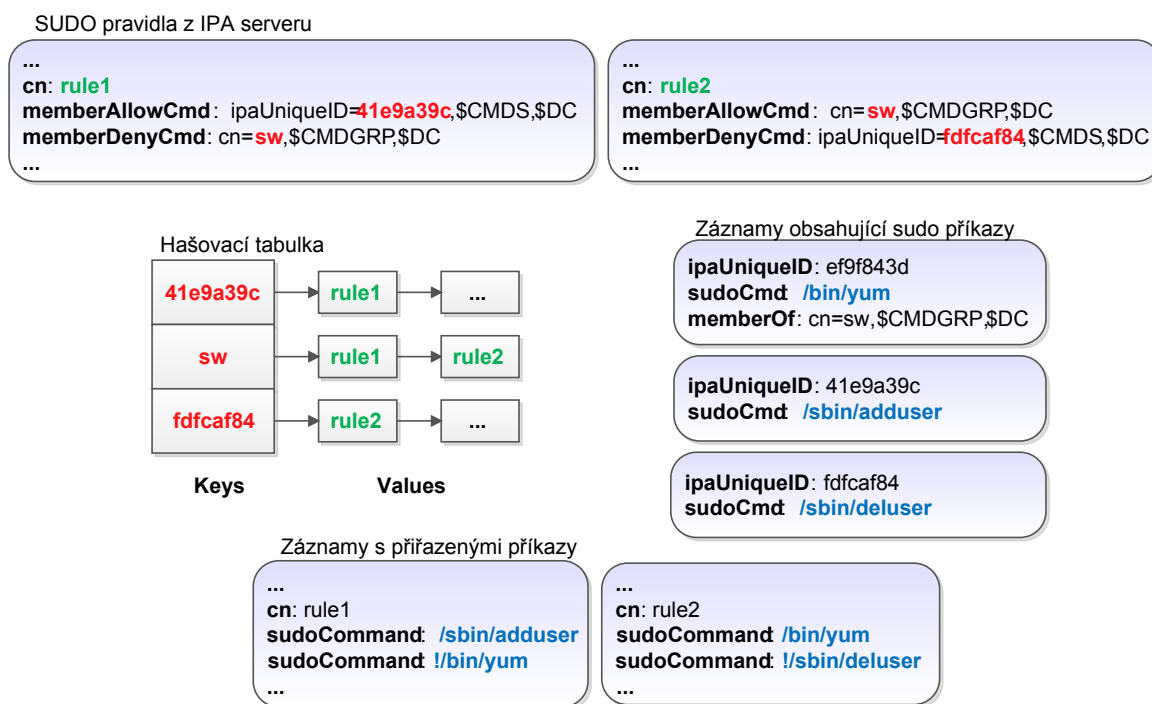
Tento způsob překladu má ovšem následující nevýhody:

- Při druhém průchodu nevíme, kde se nepřeložené atributy `memberAllowCmd` a `memberDenyCmd` nacházejí a proto je nutné je v pravidlech znovu dohledat.
- Pro každý `memberAllowCmd` a `memberDenyCmd` atribut je nutné vyhledat potřebný příkaz, což vede k mnoha iteracím nad **neměnicí** se množinou příkazů.

Optimalizací tohoto přístupu by mohlo být uchování hodnot odkazů na příkazy a pravidla ve kterých se příkaz vyskytl. Jestliže N je počet odkazů na příkazy, ve stažených

⁴<https://fedorahosted.org/freeipa/ticket/4314#no1>

záznamech sudo pravidel, pak je nutné N -krát iterovat množinou příkazů, aby bylo možné sestavit kompletní sudo pravidla. Tyto problémy je možné odstranit za pomoci použití hašovací tabulky. Tento přístup zobrazuje schéma uvedené na obrázku 4.4. Jako klíč může posloužit DN záznamu SUDO příkazu nebo skupiny příkazů. Hodnotou položky této tabulky poté může být seznam všech pravidel, ve kterých se daný příkaz nebo skupina příkazů vyskytuje. Je ovšem nutné si také uchovat informaci o tom, zda má být daný příkaz povolen/zakázán a také pořadí daných příkazů, na kterém záleží, viz Sekce 2.1.2.



Obrázek 4.4: Použití hašovací tabulky pro překlad atributu s příkazy.

Hlavní výhodou tohoto přístupu je snížení počtu iterací množinou záznamů příkazů. Tento přístup je vhodný v případech, kdy pracujeme s velkým počtem záznamů o SUDO pravidlech a příkazech. Typicky tedy při provádění úplných aktualizací. Při aktualizaci pravidel a průběžných aktualizacích budou tyto množiny obecně menší. Použití hašovací tabulky by tedy v těchto případech nebylo nutné. Typ aktualizace je znám před samotným stažením pravidel a počet stažených SUDO pravidel po prvním dotazu. Je tedy možné provádět výběr zvoleného algoritmu na základě těchto informací.

Jestliže máme zvolený algoritmus překladu pravidel, pak je nutné rozhodnout, kdo tento překlad bude realizovat. Nabízí se řešení uložit stažená SUDO pravidla v IPA formátu přímo do cache paměti a nechat SUDO respondera tyto pravidla překládat. Tento přístup sice je realizovatelný, ale není správný z pohledu architektury SSSD. Jak bylo řečeno v sekci 3.2, responderi mají s pomocí cache paměti pouze co nejrychleji odpovídat na dotazy klientských aplikací. Z tohoto důvodu musí být do sysdb uložena již přeložená SUDO pravidla a jejich zpracování tedy musí provést SUDO provider. V tomto případě tedy překlad pravidel musí realizovat navrhovaný IPA SUDO provider.

4.1.4 Využitelnost LDAP pluginu

IPA SUDO provider bude provádět velmi podobnou funkcionalitu jako LDAP SUDO provider. Mohlo by se tedy zdát, že jediná úprava, kterou je nutno provést v LDAP SUDO providerovi, je přidání překladu pravidel před samotným uložením těchto pravidel do cache paměti. Překlad pravidel je ovšem specifický pro IPA providera. V kódu LDAP SUDO providera by se tedy muselo vyskytnout volání funkcí, které by se nacházely v IPA SUDO providerovi. Tento přístup je sice realizovatelný⁵, ovšem neodpovídá návrhu SSSD. Kód zajišťující překlad pravidel by byl součástí dynamické knihovny `libsss_ipa.so` zatímco kód LDAP SUDO providera se nachází v knihovně `libsss_ldap.so`. SSSD je ovšem možno nainstalovat a používat bez podpory IPA. Například je možno využít SSSD pouze ke cachování SUDO pravidel uložených na LDAP serveru, kde se samozřejmě nacházejí v nativním LDAP SUDO schématu. V takovém případě není potřeba SUDO pravidla překládat a SSSD je nakonfigurováno tak, aby využilo LDAP SUDO providera z knihovny `libsss_ldap.so`. Knihovna `libsss_ipa.so` tedy není potřeba. Při načítání `libsss_ldap.so` by poté nastal problém, jelikož by se odkazovala na funkce, které se nacházejí v knihovně, která není a ani by nebyla načtena do paměti. Proto není možné z LDAP SUDO providera volat kód z IPA SUDO providera. Je ovšem možné z IPA SUDO providera volat kód LDAP SUDO providera, LDAP provider je možné si představit jako jakýsi stavební blok, nad kterým mohou stavět ostatní provideři, a toho je možno využít. Větší zásah do LDAP SUDO providera by také mohl vnést mnoho chyb do již fungujícího a odladěného kódu.

Asynchronní modul `sdap_async_sudo.c` provádí stažení a uložení SUDO pravidel v nativním LDAP schématu do sysdb. Je tedy možné využít této asynchronní události pro stažení pravidel v IPA SUDO schématu. Pro tato pravidla je nutné dále stáhnout SUDO příkazy a poté provést překlad těchto pravidel. Jestliže máme SUDO pravidla přeložena z IPA SUDO schématu do nativního LDAP schématu, můžeme opět využít zmiňovaného modulu pro aktualizaci cache paměti.

4.2 Implementace

Pro specifikaci pravidel aplikovatelných na klientský počítač byl vybrán přístup, který je detailně popsán v podsekci 4.1.1. Výběr těchto pravidel je tedy proveden dotazem do kontejneru `cn=sudorules,cn=sudo,$DC`. Filtr pro LDAP dotaz specifikuje klienta pomocí FQDN daného počítače a skupin, ve kterých je členem. Pro získání těchto skupin je použita funkce `ipa_host_info_send()`. Základní filtr, který je použit pro úplnou aktualizaci vypadá následovně:

```
( & ( objectClass=ipasudorule )
  ( ipaEnabledFlag=TRUE )
  ( | ( cn=defaults )
    ( hostCategory=ALL )
    ( memberHost=FQDN )
    ( externalHost=FQDN )
    ( memberHost=DN of a HOSTGROUP )
    . . .
  ) )
```

Obrázek 4.1: Filtr, který ipa plugin používá pro úplnou aktualizaci.

⁵První prototyp, ipa modulu pro SUDO providera, byl na této myšlence založen.

Při aktualizaci pravidel jsou pouze přidána jména pravidel, stejně jako u LDAP SUDO providera. Zde by také bylo možné využít atributu `ipaUniqueID`, který jednoznačně identifikuje SUDO pravidlo. To by ovšem vyžadovalo úpravu SUDO respondera, která by nepřinesla žádnou výhodu. Jelikož jsou hodnoty atributů `cn`, tj. jména SUDO pravidel, jedinečná i v IPA SUDO schématu, není potřeba tuto změnu provádět. Při průběžných aktualizacích je opět využito atributu `EntryUSN`, k detekci nově přidáných nebo modifikovaných SUDO pravidel.

Stažení kompletních SUDO pravidel, tj. pravidla a příkazy, je provedeno pomocí dvou dotazů pro všechny typy aktualizací. Pro stažení pravidel jsou tedy zapotřebí **nejvýše tři** LDAP dotazy a nejméně dva v případě, že stažená pravidla neobsahují odkazy na další záznamy, tj. odkazy na záznamy příkazů.

Zvažována byla také varianta, kde by se pro úplnou aktualizaci postupovalo podle metody 4.1.2, tzn. stažení SUDO pravidel a zároveň všech příkazů v jediném LDAP dotazu. SSSD používá své SDAP API pro provádění LDAP dotazů. Výsledky LDAP a SDAP dotazů ovšem v některých případech nejsou ekvivalentní⁶. Záznamy, přijaté ze serveru, jsou filtrovány předtím, než jsou předány programátorovi. To zajišťuje funkce `sdap_parse_enrty`. Ta vrací pouze záznamy, které spadají do jedné konkrétní třídy. Jestliže přijaté záznamy nejsou odvozeny z této konkrétní třídy, pak jsou zahozeny.

Záznamy o sudo pravidlech a příkazy pro tyto pravidla jsou uloženy v separátních kontejnerech a také jsou definovány pomocí jiné třídy, viz. Tabulka 2.6. Z tohoto důvodu není možné jedním SDAP dotazem získat záznamy sudo pravidel a zároveň záznamy příkazů pro tyto pravidla.

Pro překlad pravidel byl zvolen základní algoritmus s drobnými optimalizacemi, který se při testování osvědčil jako dostatečný pro malou množinu pravidel. V případě výkonnostních problémů je možno přidat podporu hašovací tabulky, která by při velké databázi pravidel mohla zvýšit rychlost překladu.

Pro prvotní stažení pravidel byl využit LDAP SUDO provider. Konkrétně jeho asynchronní modul popsany v sekci 3.4.4. To ovšem vyžadovalo úpravu v synchronní funkci `sdap_sudo_refresh_load_done()`. Jestliže tento modul využívá IPA SUDO prvider. Pak stažená pravidla ještě nemohou být uložena do sysdb. Proto je asynchronní událost stažení pravidel označena jako dokončena před prací s cache pamětí a řízení je vráceno zpět IPA SUDO providerovi. Z tohoto modulu jsou dále využity funkce, které zajišťují práci s cache pamětí, tzn. odstraňování nevalidních a ukládání nově stažených a již **přeložených** pravidel. Pro plánování pravidelných aktualizací byl použit interní plánovač periodických událostí *ptask*. Nově navržený provider se skládá z následujících modulů:

- `ipa_SUDO.c` – provádí počáteční inicializaci IPA SUDO providera, nastavuje SUDO handler a provádí nastavení průběžných aktualizací
- `ipa_async_sudo_hostgroups.c` – asynchronní získání skupin ve kterých je daný počítač členem
- `ipa_sudo_refreshes.c` – provádí nastavení parametrů pro jednotlivé typy aktualizací
- `ipa_async_sudo.c` – asynchronní stažení SUDO pravidel z IPA serveru
- `ipa_async_sudo_cmds.c` – asynchronní stažení záznamů o příkazech pro stažená sudo pravidla

⁶důvodem jsou mapy a záznamy, které nemají společné třídy

- `ipa_sudo_cmd.c` – sestavení LDAP filtru pro získání záznamů o příkazech pro stažená sudo pravidla
- `ipa_sudo_export.c` – překlad samotných pravidel

4.3 Testování

Pro tvorbu jednotkových testů využívá SSSD framework `cmocka`⁷, který podporuje Mock objekty. Pomocí těchto objektů je možné simulovat jakýkoliv reálný objekt. V našem případě byl jako Mock objekt zvolen IPA server, tj. veškerá rozhraní zajišťující komunikaci s IPA serverem. Jednotkové testy je tedy možné provádět bez přítomnosti IPA serveru či připojení k síti. Mock objekty tedy simulují IPA server a SUDO pravidla, která se na něm nacházejí.

4.3.1 Metodika

Reálná SUDO pravidla si každá společnost pečlivě střeží. Proto nebylo možné taková pravidla získat. Pro testování jsem tedy vytvořil sadu pravidel, které pokrývají následující případy použití:

- SUDO pravidla jejichž atributy se neodkazují na žádné objekty (např. uživatele nebo počítače) IPA domény
- komplexní pravidla jejichž atributy se odkazují na jiné objekty IPA domény
- pravidla, která neobsahují odkazy na SUDO příkazy
- výskyt neexistujícího atributu
- výskyt atributu, který se odkazuje na neexistující objekt v IPA doméně
- případ, kdy se na serveru nenacházejí žádná pravidla
- testování správného pořadí přeložených příkazů

⁷<http://cmocka.org/>

Kapitola 5

Závěr

TODO: pochlub se co bylo potřeba nastudovat: talloc, tevent, ldb ke které není dokumentace.

- NSS, pam, ldb ke které není doc

Tato bakalářská práce popisuje LDAP schémata, která používají program SUDO a server FreeIPA k ukládání SUDO pravidel v LDAP adresářích. Srovnává jejich rozdíly a výhody jejich použití. Dále dokumentuje funkcionalitu ldap a ipa pluginů, které má k dispozici SUDO provider démonu SSSD. Tato dokumentace odkrývá nedostatek démonu SSSD, jelikož nepodporuje nativní IPA SUDO schéma. Tento nedostatek se podařilo odstranit navržením a implementací nativního IPA SUDO providera, který IPA SUDO schéma podporuje. Přenáší tak překlad SUDO pravidel ze serveru FreeIPA na klientský démon SSSD. Tento překlad byl také otestován pomocí jednotkových testů. Při návrhu byla také nalezena jedna chyba a bylo poukázáno na některé nedostatky současné implementace.

V práci je dále možno pokračovat například rozšířením jednotkových testů. Mohly by být provedeny výkonnostní testy, na jejichž základě by mohla být stanovena hranice pro výběr, konkrétního algoritmu pro překlad pravidel. Pro plánování aktualizací u LDAP SUDO providera by taktéž mohl být použit ptask plánovač, popř. by toto rozhraní mohlo být navrženo takovým způsobem, aby jej mohli využít oba pluginy pro SUDO providera. Integrace obou pluginů SUDO providera by mohla být provedena ještě lépe, kde by všechny společné části využívaly stejný kód, čímž by se docílilo nulové redundance kódu. To by ovšem vyžadovalo nový návrh celého SUDO providera, což by znamenalo kompletní přepsání jak ldap tak ipa SUDO pluginů pro SUDO providera.

Literatura

- [1] *RedHat Enterprise Linux 4 Security Guide*. Red Hat, Inc., second edition edition, 2008.
- [2] Ella Deon Lackey. Red hat enterprise linux 6 identity management guide.
https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/pdf/Identity_Management_Guide/Red_Hat_Enterprise_Linux-6-Identity_Management_Guide-en-US.pdf, 2013.
- [3] Gordon S. Good Timothy A. Howes Ph.D., Mark C. Smith. *Understanding and Deploying LDAP Directory Services*. Addison Wesley, 2. edition, 2003.
- [4] Sudoers ldap manual [online]. <http://www.sudo.ws/sudoers.ldap.man.html>.
- [5] Sudoers manual [online]. <http://www.sudo.ws/sudoers.man.html>.
- [6] H. Chu. Ordered entries and values in ldap [online].
<http://tools.ietf.org/html/draft-chu-ldap-xordered-00>, 2006.
- [7] P. Masarati. Ldap dereference control [online].
<http://tools.ietf.org/html/draft-masarati-ldap-deref-00>, 2008.
- [8] Rüdiger Landmann. *Red Hat Directory Server 8.2 Administration Guide*. Red Hat, Inc., 2010.
- [9] Pavel BŘEZINA. Talloc - a hierarchical memory allocator [online].
http://is.muni.cz/th/359290/fi_b/, 2012 [cit. 2014-04-05].
- [10] David Koňář. *Developer Support Tools for tevent Library*. FIT VUT v Brně, 2013.
- [11] T. Howes. The string representation of ldap search filters. rfc2254, December 1997.

Příloha A

Přiložené CD obsahuje:

- `freeipa.schemas/`: LDAP schémata používaná FreeIPA serverem.
- `freeipa.schemas/65sudo.ldif`: schéma definující IPA sudo pravidla a příkazy.
- `latex`: kompletní zdrojový text této práce včetně všech schémat.
- `export_table.pdf`: tabulka, která byla použita pro implementaci překladu atributů sudo pravidel.
- `schema_compat.uldif`: schéma, které používá `compat` plugin pro předkad sudo pravidel.
- `sssd`: kompletní zdrojové texty projektu SSSD.
- `literature`: použitá literatura, která je volně dostupná.