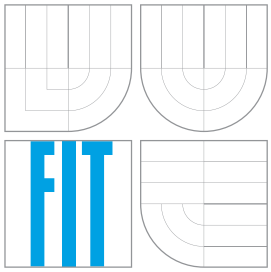
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

Smyslem 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 přináší návrh a implementaci IPA SUDO provideru. 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 the native IPA SUDO provider. Designed provider eliminates unnecessary overhead of exporting SUDO rules from IPA SUDO schema info 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
19. května 2014

Poděkování

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

Obsah

1 Úvod **3**

2 Správa bezpečnostních politik identit **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 . . . 10
2.2.2 Definice SUDO pravidel . . . 13
2.3 System Security Services Daemon . . . 13
2.3.1 Aktualizace SUDO pravidel v cache paměti . . . 14

3 SSSD a SUDO provider **16**
3.1 Knihovny třetích stran . . . 16
3.2 Architektura SSSD . . . 18
3.3 SSSD z pohledu programu SUDO . . . 19
3.3.1 Zjištění informací o uživateli . . . 20
3.3.2 Autentizace uživatele . . . 20
3.3.3 získání sudo pravidel . . . 20
3.4 LDAP SUDO Provider . . . 21
3.4.1 Inicializace . . . 21
3.4.2 Plánování aktualizací . . . 23
3.4.3 Aktualizace SUDO pravidel . . . 23
3.4.4 Asynchronní modul . . . 26
3.4.5 Uložení SUDO pravidel do sysdb . . . 26
3.4.6 LDAP SUDO Handler . . . 28
3.5 IPA SUDO provider . . . 28
3.6 Dereferenční dotaz . . . 30

4 Nativní IPA SUDO Provider **31**
4.1 Návrh . . . 31
4.1.1 Správný přístup k získávání SUDO pravidel . . . 31
4.1.2 Stažení SUDO pravidel . . . 33
4.1.3 Překlad pravidel . . . 35
4.1.4 Využitelnost LDAP pluginu . . . 38
4.2 Implementace . . . 38
4.3 Testování . . . 40

4.3.1 Metodika 40

5 Závěr **41**

A Obsah CD **43**

Kapitola 1

Úvod

Tato bakalářská práce popisuje správu bezpečnostních politik identit s pomocí použití programu SUDO. Popisuje problémy, na které může administrátor narazit při nasazení tohoto řešení v doménovém prostředí. Představuje možná řešení těchto problému použitím technologií FreeIPA a SSSD, které jsou používány v Red Hat Enterprise prostředích. Toto řešení ovšem disponuje určitými nedostatky, které tato práce odkrývá. Smyslem této práce je tedy tyto nedostatky identifikovat, navrhnout jejich řešení a tyto řešení implementovat.

Cílem tedy není diskutovat nad možnostmi a technologiemi, které je možné pro správu politik identit použít. Také nemá sloužit jako návod pro konfiguraci výše zmíněných technologii pro řešení daného problému.

Od čtenáře je ovšem očekávána uživatelská znalost programu SUDO, základních konceptů online adresářů a LDAP protokolu.

Struktura této práce je rozdělena do pěti kapitol. První kapitola popisuje obsah této práce. V kapitole 1 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 a 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.

Zmíněný popis dále posloužil pro sestavení návrhu, který tento nedostatek odstraňuje. Návrh, implementaci a testování navrženého řešení popisuje kapitola 4.

Poslední kapitola poté zhodnocuje dosažené výsledky a diskutuje nad možnostmi pokračování v této práci.

Tato práce dále používá následující konvence:

1. Všechny důležité pojmy jsou vysázeny **tučným písmem**.
2. Nové pojmy jsou vysázeny *kurzivou*.
3. Hodnoty atributu `ipaUniqueID`[1] byly záměrně zkráceny kvůli velikosti textu a obrázků.
4. Všechny názvy souborů, jejich obsahy, 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 identit

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

2.1 SUDO

Pomocí programu SUDO má administrátor možnost delegovat oprávnění správce[1] ostatním uživatelům bez nutnosti sdílení hesla správce. Umožňuje uživateli operačního systému spustit jeden příkaz jako jiný uživatel. Ve většině případů je tímto uživatelem právě správce linuxového systému, tj. uživatel *root*. Tedy uživatel, který má nejvyšší oprávnění. Jestliže uživatel vyvolá nějaký příkaz za pomocí 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 také označovaná jako *sudoers*. Tyto pravidla se zapisují v následujícím formátu:

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

- `user` – specifikuje uživatele nebo skupinu na kterou bude pravidlo aplikováno
- `host_list` – reprezentuje seznam koncových systému
- `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:

[1] uživatele root

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

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 SUDO pravidla mezi více počítačů. SUDO ovšem nemá nativní způsob jak tyto pravidla distribuovat mezi více počítačů [2, Kapitola 20]. Spravovat tyto pravidla ručně na všech počítačích, které administrátor spravuje, by bylo ovšem velice pracné a časově náročné. Otázkou tedy zůstává jak tyto pravidla šířit mezi více počítačů. Existuje několik způsobů jak toho docílit.

1. Ručně distribuovat tyto pravidla mezi více systému 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.

LDAP adresáře[3] mohou uchovávat různé typy informací a jsou tak používány organizacemi a institucemi k ukládání informací o jejich zaměstnancích, uživatelích nebo zařízeních jejich počítačových sítí. Jsou charakterizovány jako „jednou-zapsat-a-mnohokrát-číst“[4] služba. 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 výhodné a nabízí následující výhody:

1. Jestliže jsou SUDO pravidla centralizována, pak se správce systému nemusí starat o jejich distribuci. Spravuje pouze jediný soubor se SUDO pravidly.

2. Vyhledávání SUDO pravidla v LDAP adresáři je rychlejší. Při použití lokálních sudoers[5] musí SUDO přečíst celý SUDOer soubor. Při uložení pravidel v LDAP adresáři je potřeba pouze několik LDAP dotazů. [4]

[2] http://puppetlabs.com
[3] Adresář, ke kterému se přistupuje pomocí LDAP protokolu.
[4] write-once-read-many-times
[5] tj. když jsou SUDO pravidla definována v lokální `/etc/sudoers` souboru

3. Jestliže udělá správce v sudoers souboru chybu, pak se program SUDO nespustí. Do LDAP adresáře není možné uložit data, která nesplňují syntaxi daného LDAP SUDO schématu. Správná syntaxe je tedy zaručena. Ovšem stále je možné udělat chybu v uživatelském jménu, příkazu nebo názvu koncového systému.

Nevýhodou centralizované správy může být situace, kdy dojde k výpadku serveru[6], na kterém jsou pravidla uložena. V takovém případě nejsou pravidla pro sudo dostupná a není možné jej 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 pravidel[7], 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 prvků[8] 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 veškeré uživatele, čehož je možno docílit použitím hodnoty `ALL`.

Možná hodnota atributu	**Příklad**
uživatelské jméno	`xsruba03`
UID	`#9843`
skupina uživatelů	`%3bit`
GID	`%#43`
unixová sítová skupina	`+admins`
non-unixová sítová skupina	`%:finance_dp`
všichni uživatelé	`ALL`

Tabulka 2.1: Popis možných hodnot atributu `sudoUser`.

[6] prvek na kterém jsou závislé ostatní části systému je v angl. označován jako „single point of failure“
[7] zde se nejedná o množinu SUDO pravidel
[8] na koho bude pravidlo aplikovatelné, kde, jaké obsahuje příkazy, ...

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.

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

Tabulka 2.2: Popis možných hodnot atributu `sudoHost`.

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

Možná hodnota atributu	**Popis**
povolený příkazzakázaný příkaz	`blkid /dev/sda1`
zakázaný příkaz	`!blkid`
všechny příkazy	`ALL`

Tabulka 2.3: Tabulka popisující možné hodnoty atributu sudoCommand.

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 jako „zastaralý“[9].

K SUDO pravidlu je také možné 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 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 validní[10]. Tyto atribut ovšem aktuálně nejsou FreeIPA serverem používána a proto je jimi dále nebudeme zabývat.

Posledním definovaným atributem 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. Předpokládejme, že soubor `/etc/sudoers` obsahuje pouze následující dvě pravidla:

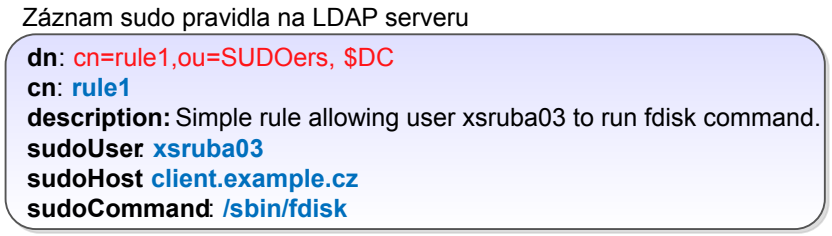
[9] deprecated

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

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

V tomto případě uživatel „xsruba03“ nebude mít oprávnění spustit příkaz `blkid`. Jestliže ovšem prohodíme tyto pravidla, 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 2.1, by v LDAP adresáři vypadal tak, jak jej popisuje obrázek 2.1.



Obrázek 2.1: Příklad záznamu SUDO 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 tedy typicky například uživatel. Většina administrátorů spravujících prostředí o desítkách a více uživatelů chce spravovat identity centrálně. LDAP adresáře jsou pro tento účel tedy velice využívány. Například ve Windows prostředích je možno použít řešení *Active Directory* nebo *FreeIPA* v linuxových prostředích. Na tyto servery je tedy možné 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 LDIF[11] formátu.
- V případě, že se uživatel ocitne bez připojení k serveru, který uchovává SUDO pravidla, pak není možné SUDO použít (případně lze pracovat pouze s lokálními pravidly definovanými v souboru `/etc/sudoers`).

[11] LDAP Data Interchange Format

- může dojít k překrytí uživatele z LDAP adresáře s lokálním uživatelem.

Jako největší nevýhodu lze ovšem považovat nulovou integraci mezi existujícími identitami v adresáři a bezpečnostními politikami. Jestliže se administrátor rozhodne spravovat jak identity tak bezpečnostní politiky těchto identit centrálně, pak by chtěl aby tyto entity byly vzájemně propojeny. Toho ovšem není možné dosáhnout s použitím nativního LDAP SUDO schématu. Při použití nativního schématu, se při definici SUDO pravidel není možné odkazovat na již administrátorem vytvořené identity[12]. 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

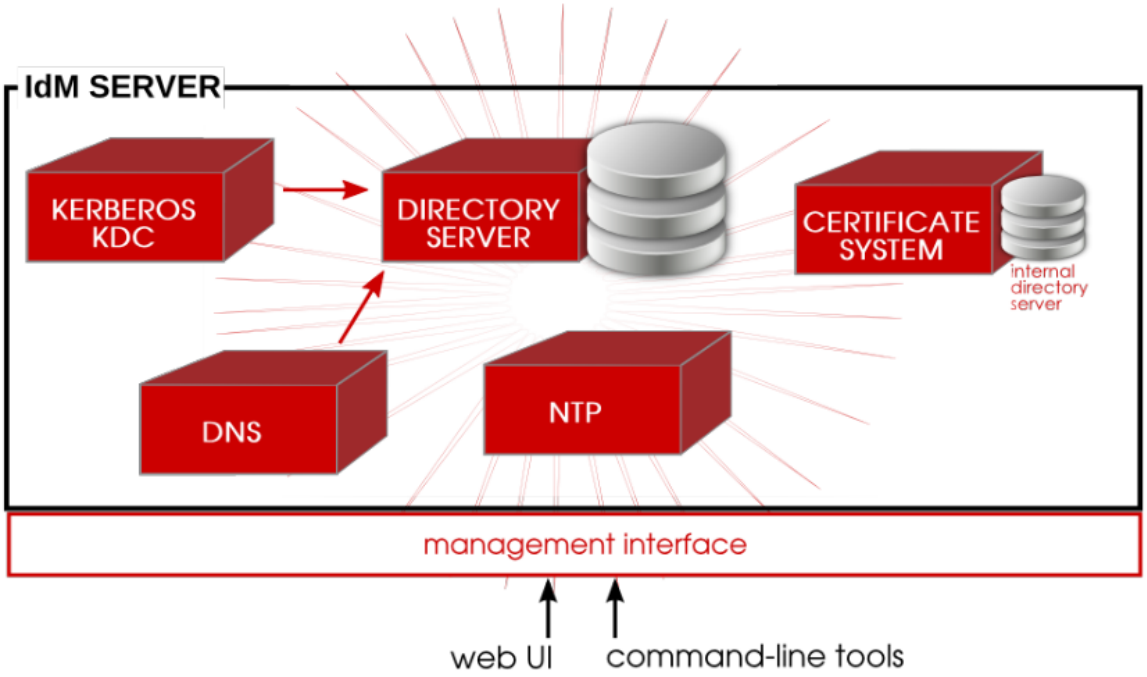
Kvůli efektivitě a jednoduché správě se IT administrátoři snaží spravovat identity centrálně a spojovat identity s autentizačními a autorizačními politikami. V linuxových prostředích existuje mnoho protokolů pro různé služby. Je možné například využití NIS[14] nebo Kerberos+LDAP[15] pro správu identit 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.

FreeIPA[16] je projekt, který 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 jako jsou například Kerberos nebo DNS[17]. Je tedy možné říci, že je to v podstatě doménový kontrolér pro linuxové a unixové stanice. Poskytuje jakousi obálku nad standardizovanými síťovými službami jako jsou PAM, LDAP, Kerberos, DNS, NTP nebo certifikační služby. Jeho hlavní úlohou je tedy 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 adresář[18]. 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 klientů, IPA používá *Dogtag Certificate System*. Služby jako Kerberos jsou závislé na doménových jménech a časových razítkách. Proto IPA slouží také jako DNS a NTP[19] server. [2, Kapitola 1]

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

[12] například uživatele nebo počítače
[13] je tedy definován v souboru `/etc/passwd`
[14] Network Information Service
[15] Lightweight Directory Access Protocol
[16] Free Identity, Policy and Audit, zkráceně IPA
[17] Domain Name System
[18] The 389 Directory Server, http://directory.fedoraproject.org
[19] Network Time Protocol



Obrázek 2.2: FreeIPA sjednocuje standardní služby.

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 neposkytuje žádné propojení s již existujícími objekty dané domény. Z těchto důvodů se vývojáři projektu FreeIPA rozhodli vytvořit vlastní schéma pro uchovávání SUDO pravidel. Nově navržené schéma umožňuje lepší integraci s již existujícími objekty s konkrétními pravidly. Umožňuje také daleko flexibilnější a jednodušší konfiguraci. Uživatel již tedy 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 DN[7, Kapitola 1]. Jestliže se tedy atribut `user` odkazuje na záznam, s informacemi o uživateli, bude hodnota tohoto atributu tvořena právě DN daného záznamu:

```
user: uid=xsruba03,cn=users,cn=accounts,$DC
```

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'
)

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[20] obsahují DN[21] již existujících objektů IPA domény, zatímco `external*`[22] atributy specifikují objekty, které jsou definovány mimo tuto doménu. Speciální případy, kdy je například SUDO pravidlo aplikovatelné na kteréhokoliv uživatele nebo počítač specifikují `*Category`[23] atributy.

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`[24]. Pomocí `externalUser` atributu je také možné vytvořit SUDO pravidlo aplikovatelné na uživatele, který není členem IPA domény. Tento uživatel ovšem musí být členem jiné domény[25] 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.

Jméno atributu	**Hodnota atributu**
`memberUser`	`uid=xsruba03,cn=users,cn=accounts,$DC`
`userCategory`	`all`
`externalUser`	`xsruba04`

Tabulka 2.4: Atributy specifikující uživatele SUDO pravidla.

Identitu, pod kterou bude příkaz spuštěn je možné specifikovat pomocí `ipaSudoRunAs*`

[20] tj. například `memberUser, memberAllowCmd`, ...
[21] Distinguished Name
[22] tj. například `externalUser, externalHost`, ...
[23] tj. například `userCategory, cmdCategory`, ...
[24] Domain Controller
[25] například doména spravována pomocí Active Directory

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.

Poslední množinou, kterou je nutno specifikovat pro vytvoření kompletního pravidla, jsou příkazy. Zde 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.

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

Tabulka 2.5: Atributy specifikující SUDO příkaz.

Kromě skupin příkazů přináší nové schéma další výhodu a to povolování/zakazování pravidel. Jestliže administrátor spravuje SUDO pravidla pomocí LDAP serveru[26], 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, viz. Tabulka 2.6.

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`

Tabulka 2.6: Rozmístění sudo pravidel na IPA serveru.

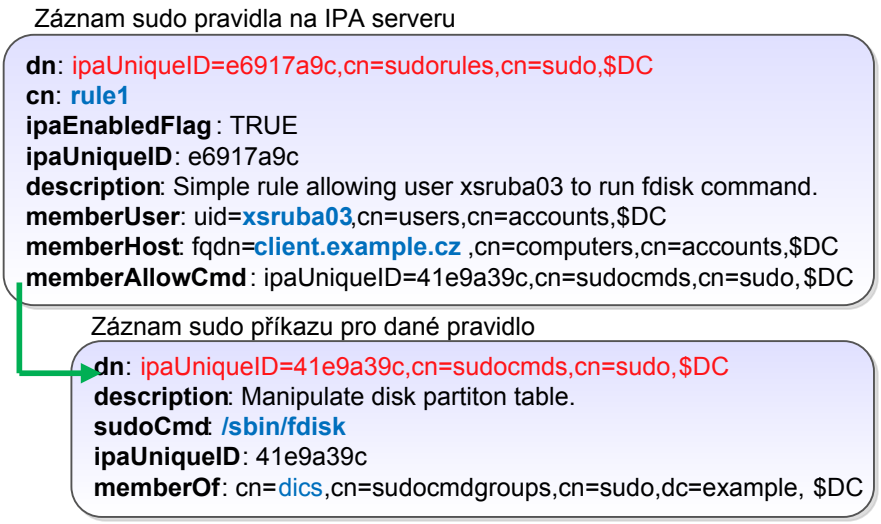
Záznam ekvivalentního suda pravidla pro pravidla 2.1 a 2.1 v novém IPA SUDO schématu popisuje obrázek 2.3.

Každý záznam na IPA serveru má také **operační** atribut entryUSN[27]. Hodnotu tohoto atributu aktualizuje USN plugin při každé změně daného záznamu. Tento plugin tedy 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/slapd-instance_name/`. Detailní popis tříd a atributů pro

[26] tj. sudo pravidla jsou definována pomocí nativního LDAP SUDO schématu

[27] Entry Update Sequence Number



Obrázek 2.3: Příklad záznamu SUDO pravidla na IPA serveru.

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. Jednak pomocí **webového rozhraní** nebo pomocí utilit příkazového řádku `ipa sudorule*`. Podrobnější popis s příklady jak definovat pravidla ve webovém rozhraní nebo pomocí příkazového řádku je možné nalézt například v oficiální dokumentaci[28]. 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 například pomocí technologií LDAP+Kerberos, musí provést nastavení NSS[29] a PAM[30]. Musí použít správně 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 velice složitá. Ovšem 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 většinou řeší separátním uživatelským účtem.

[28] Red Hat Enterprise Linux 6 Identity Management Guide
[29] Name Service Switch
[30] Pluggable Authentication Modules

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 a jejich autentizaci a politiky. Identity a jejich autentizace tak může být spravována pomocí technologií LDAP+Kerberos, FreeIPA nebo Active Directory. Je také možné jej použít pouze pro přístup k sudo pravidlům spravovaných centrálně pomocí LDAP adresáře.

Umožňuje uživateli připojení do více domén zároveň. Pokud například klient střídá prostřední 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 klienta. Tyto informace je tedy možné využívat v situacích kdy se klient ocitne bez připojení k síti. To vše je možné nakonfigurovat pomocí jediného konfiguračního souboru `/etc/sssd/sssd.conf`.

Jestliže se administrátor rozhodně 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 pluginu[31] následující hlavní výhody:

- **offline podpora** - SSSD si ukládá všechna přijatá pravidla do své cache paměti na disku. 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. Další 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 cachování těchto pravidel. Konfigurace SSSD pro tento účel je popsána v manuálové stránce `sssd-sudo`[32] a některé konfigurační volby také v `sssd-ldap`[33].

Pro cíle této bakalářské práce nás ovšem dále bude SSSD zajímat pouze z pohledu programu SUDO. Tedy jakým způsobem SSSD sudo pravidla zpracovává.

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.

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

[32] http://jhrozek.fedorapeople.org/sssd/1.11.5/man/sssd-sudo.5.html

[33] http://jhrozek.fedorapeople.org/sssd/1.11.5/man/sssd-ldap.5.html

SUDO ve spojení s LDAP adresářem pracuje vždy a aktuálními pravidly, protože se dotazuje LDAP adresáře při každém spuštění. SSSD se dotazuje LDAP serveru pouze v případě, kdy dané pravidla v cache paměti již není validní. Aktualizace se poté snaží reflektovat stav SUDO pravidel na serveru v cache paměti SSSD.

Průběžné aktualizace

Periodicky stahují pravidla, která jsou na serveru nově přidány nebo byly od poslední aktualizace modifikovány. V manuálových stránkách a v kódu SSSD je možné tento typ aktualizace vyhledat jako „*smart refresh*“.

Úplná aktualizace

Provede smazání všech pravidel z cache paměti a stažení všech pravidel ze serveru. Tento typ 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[34]. V manuálnových stránkách a v kódu SSSD je možné tento typ aktualizace najít jako „*full refresh*“.

Aktualizace pravidel

Je provedena při každém spuštění programu SUDO. Zajistí aby uživatel nedostal větší oprávnění než je definováno. Tento typ aktualizace stáhne ze serveru pouze ty pravidla, která v cache paměti nadále nejsou platná[35]. Jestliže se některé pravidlo na serveru nenachází, indikuje to změnu na serveru a je provedena úplná aktualizace pravidel. V manuálnových stránkách a v kódu SSSD je možné tento typ aktualizace najít jako „*rules refresh*“.

[34] při zachování výchozího nastavení se provádní každých 6 hodin

[35] Čas po kterém pravidla v cache paměti stanou neplatná je možno nastavit v `sssd.conf` pomocí volby `entry_cache_sudo_timeout`, viz. `manuálová stránka sssd.conf`[36]

Kapitola 3

SSSD a SUDO provider

Smyslem této kapitoly je popsat vnitřní architekturu démona SSSD se zaměřením na SUDO providera a pluginy, které má k dispozici. Tento popis byl sestaven na základě experimentování a čtení zdrojových kódů. Popis byl nezbytný pro úplné pochopení způsobu, jakým SSSD SUDO pravidla zpracovává. V poslední sekci je také popsán hlavní problém, kterým se tato práce zabývá.

3.1 Knihovny třetích stran

SSSD používá pro alokaci dynamické paměti hierarchický alokátor *talloc*[9] namísto standardního systémového volání *malloc*. Nepoužívá imperativní ani OOP[1] paradigma, ale událostmi řízené paradigma[2] 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:

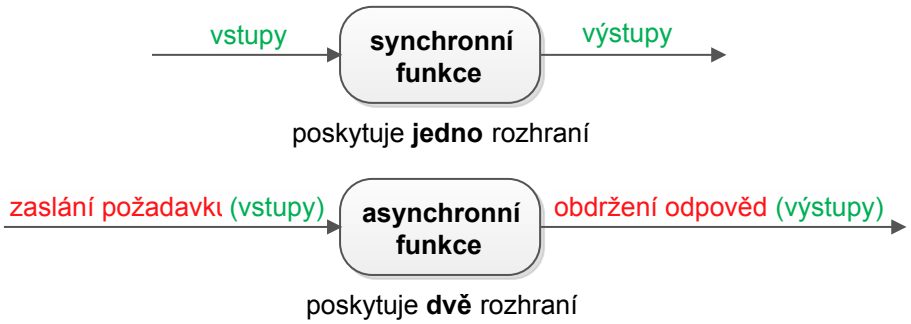
1. čeká na událost
2. provádí obsluhu události

Knihovna tevent[10] realizuje tyto události za pomocí 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*. [10]

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

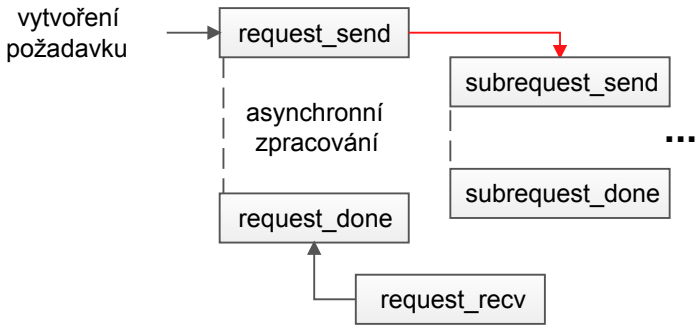
- S událostí jsou spojeny privátní data, které mají příponu `_state`.
- Funkce, která provádí asynchronní volání má příponu `_send`.

[1] Object-oriented programming
[2] Event-driven Programming Paradigm



Obrázek 3.1: Rozhraní synchronních a asynchronních funkcí.

- 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 3.2: Příklad grafického zobrazení asynchronní funkce.

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 rozhraní „request_send“[3]. Poté probíhá asynchronní provádění programu. V okamžiku kdy bude událost označena jako dokončená, vyvolá se její callback tj. funkce „request_done“. V této funkci je možné zpracovat výsledky dané události a následně ji odstranit.

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 komplexní struktura potenciálně obsahující další struktury. Z tohoto důvodu používá knihovna

[3]funkci je možné pojmenovat např. `ldap_query_send`

tevent pro alokaci paměti rovněž alokátor talloc, který veškerá data zanořených asynchronních funkcí alokuje hierarchicky. Při dokončení první události[4] 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 (démonů). 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. *respondeři*. Jsou to procesy, které přijímají požadavky od klientských aplikací[7] 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**.

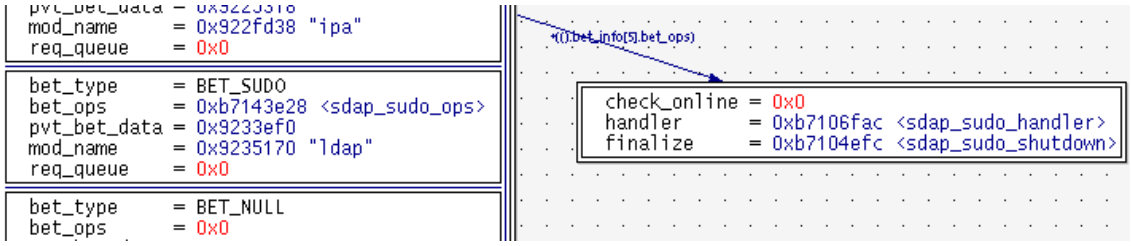
Veškerou práci spojenou s komunikací se vzdáleným serverem, zpracování výsledků a jejich uložení do cache paměti by měl zajistit konkrétní *provider*. Který provider bude použit pro kterou službu je možné nastavit v konfiguračním souboru `sssd.conf`. Například `id_provider` specifikuje, který provider SSSD použije pro zjišťování informací o identitách. Informace o SUDO pravidlech poté poskytuje SUDO provider. Každý provider může používat jiný plugin. Například provider identit může použít *ldap plugin* pro přístup k identitám uložený v LDAP adresáři, *ad plugin* pro identity spravované serverem *Active Direcetory*. 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`[8]. 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 pro komunikaci se vzdálenými servery.

Každý backend má ve své interní struktuře[9] 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 respondeři, pokud chtějí danému providerovi zaslat nějaký požadavek. Jméno této funkce má v SSSD obecně příponu **handler**[10] a takto bude označována i v následujícím textu. Položku pole `bet_info` pro LDAP SUDO providera zobrazuje obrázek

[4] ta která volala další asynchronní funkce a ta další atd
[5] www.openldap.org
[6] dbus.freedesktop.org
[7] např. `sudo`, `id`, `getent`, `ls`, ...
[8] http://jhrozek.fedorapeople.org/sssd/1.11.5/man/sssd.conf.5.html
[9] kontext `struct be_ctx`
[10] například `sdap_sudo_handler` je SUDO handler pro LDAP SUDO providera

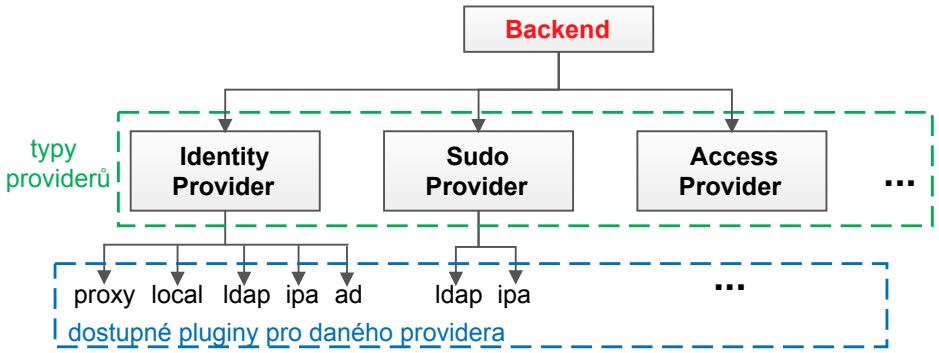
3.3.



Obrázek 3.3: Položka pole `bet_info` pro SUDO.

Každý backend také používá svou cache paměť. Do této paměti zapisuje pouze daný backend a respondeři z ní mohou data pouze číst. Pro tuto cache je použita transakční LDB[11] databáze, což je zjednodušená vezre LDAP adresáře. I když tato databáze přesně neodpovídá LDAP standardu, disponuje podobným API[12], 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.

Respondeři 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 o tom, 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 na nainstalován démon SSSD. Uvažujme, že je klient 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.

[11] http://ldb.samba.org/
[12] Application programming interface

V tomto okamžiku SUDO musí provést následující tři úlohy:

- zjištění informací o uživatelském účtu
- autentizace daného uživatele
- získání sudo pravidel

3.3.1 Zjištění informací o uživateli

Pro zjištění informací o uživatelském účtu použije SUDO funkce standardní knihovny jazyka C, tj. `getpwend()`, popř. `getgrouplist()`. Tyto funkce poté volají NSS[13] knihovnu[14]. Ta je dynamicky načítána aplikacemi[15], které potřebují komunikovat s SSSD. Získávání informací o uživatelích je velice častá operace, proto tato knihovna disponuje svojí cache pamětí, která se označuje jako tzv. *fast cache*[16]. Jesliže NSS ve své cache paměti daného uživatele nenajde, pošle dotaz na tohoto uživatele NSS responderu, což je jeden z procesů SSSD, konkrétně proces sssd_nss.

NSS responder se nejprve podívá do sysdb, jestliže zde daného uživatele nalezne, vrátí jej NSS. V opačném případě pošle požadavek, na aktualizaci cache, provideru identit[17]. 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 SUDO má informace o uživateli, pak musí provést autentizaci tohoto uživatele. Autentizaci prování, opět upravená, knihovna PAM[18]. Ta nedisponuje vlastní cache pamětí a posílá tak požadavek přímo PAM responderu. Ověření poté probíhá podobným způsobem jako u NSS respondera. PAM responder ovšem žádá o aktualizaci sysdb providera zajišťující autentizaci[19]. A navíc znovu provádí ověření daného uživatele oproti vzdálenému serveru.

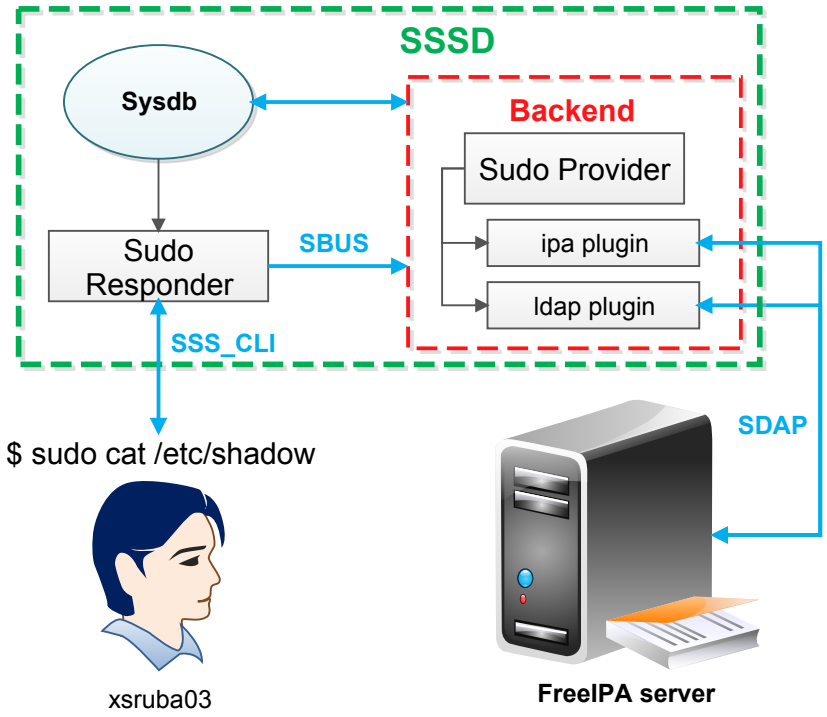
3.3.3 získání sudo pravidel

Jestliže autentizace uživatele proběhla úspěšně, pak přichází na řadu SUDO responder. Ten hledá nejprve požadované pravidlo v cache paměti. Jestliže se dané pravidlo v sysdb 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 pro požadované pravidlo. Tento průběh také zobrazuje schéma 3.5, které předpokládá, že uživatelské jméno již bylo ověřeno a proběhla také autentizace.

[13] Name Service Switch
[14] jedná se o upravenou verzi této knihovny
[15] ls, sudo, getent
[16] jedná se o jinou cache paměť, než kterou používá SSSD
[17] v `sssd.conf` volba `id_provider`
[18] Pluggable Authentication Modules
[19] v `sssd.conf` volba `auth_provider`

Jestliže po druhém dotazu[20] 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.



Obrázek 3.5: SUOD a SUDO responder.

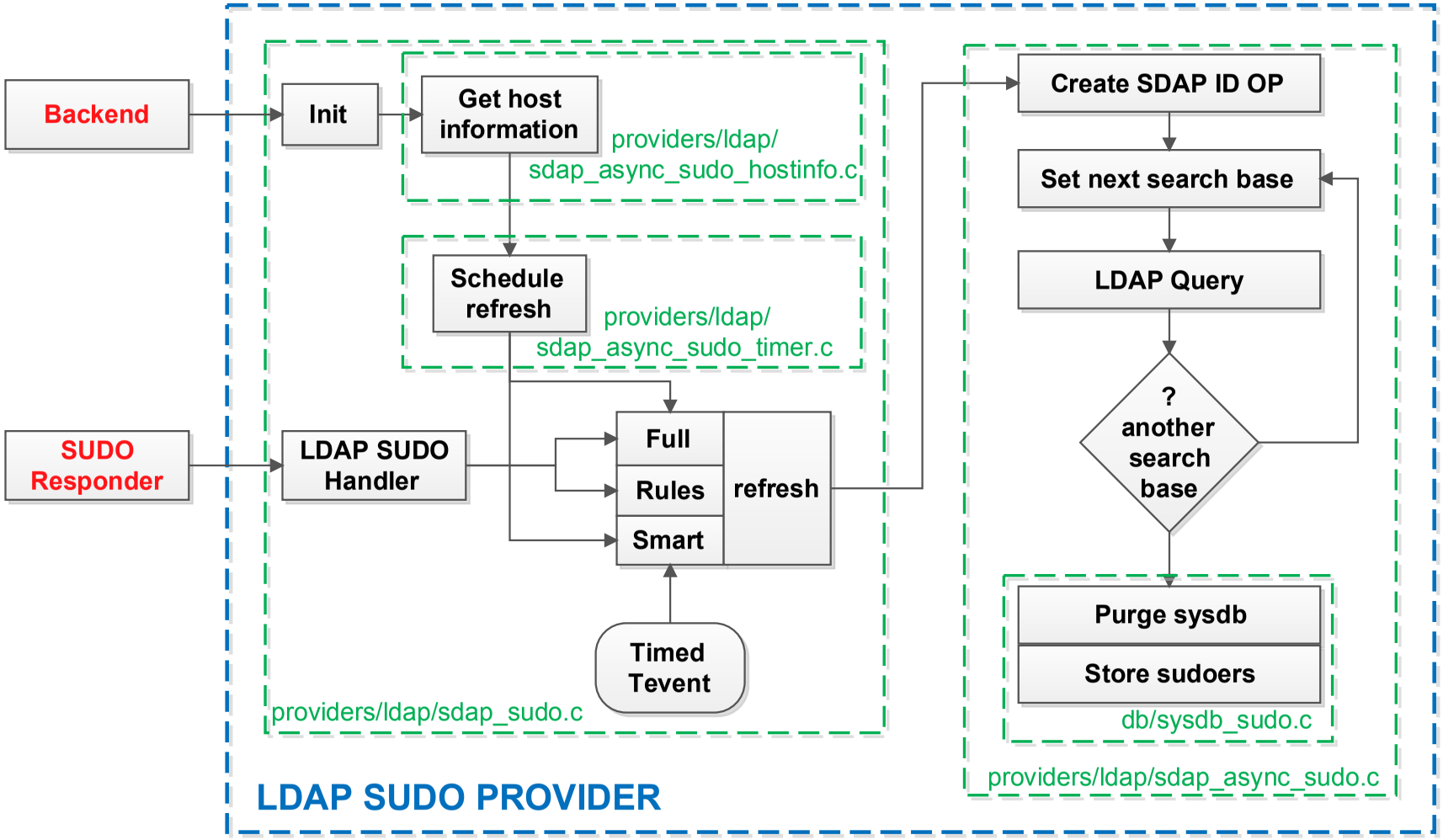
3.4 LDAP SUDO Provider

SUDO provider provádí veškerou práci při komunikaci s LDAP serverem. Pomocí volby `sudo_provider` v konfiguračním souboru `sssd.conf` je možno určit, který plugin tento provider bude používat. Na výběr jsou pluginy „ldap“[21] a „ipa“[22]. Tato sekce popisuje funkcionalitu ldap pluginu, který je nutno použít, máme-li sudo pravidla uložena na LDAP serveru v nativním LDAP SUDO schématu. Tento plugin provádí stažení, zpracování a uložení těchto pravidel do sysdb. Skládá se z několika modulů jejichž celkovou funkcionalitu popisuje schéma 3.6.

3.4.1 Inicializace

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

[20] v prvním dotaze bylo zjištěno, že hledané pravidlo není platné
[21] `sudo_provider=ldap`
[22] `sudo_provider=ipa`



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

- 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

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 FQDN[23]. 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` pod volbami `ldap_sudo_hostnames` a `ldap_sudo_ip`. Jejich hodnoty jsou poté zohledněny při sestavovaní filtrů pro LDAP dotazy. Použitím těchto voleb je poté možné stahovat i taková SUDO pravidla, která nejsou aplikovatelná na klientský počítač[24].

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`[25].

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.[26] 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 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

Pro úplnou aktualizaci SUDO pravidlel používá SUDO provider LDAP filtr 3.1 Tento i všechny následující filtry se řídí syntaxí specifikovaou dokumentem RFC2254 [11]. Jsou pouze doplněny bílými znaky pro lepší čitelnost.

[23] Fully Qualified Domain Name
[24] Používáno převážně pro testování.
[25] Volby `ldap_sudo_full_refresh_interval` a `ldap_sudo_smart_refresh_interval`
[26] za předpokladu, že se na serveru nějaká pravidla nacházejí

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

Průběžná aktualizace

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.

Aktualizace pravidel

Jestliže se v sysdb při spuštění programu SUDO nachází nějaká pravidla, která již nejsou platná, 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 použije SUDO provider filtr 3.5. Nově stažená pravidla poté budou uloženo do sysdb.

```
(&(&(objectClass=sudoRole)
    (|(cn=rule1)
      (cn=rule2)
    )
  )
  (|(!(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 pravide 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.

3.4.4 Asynchronní modul

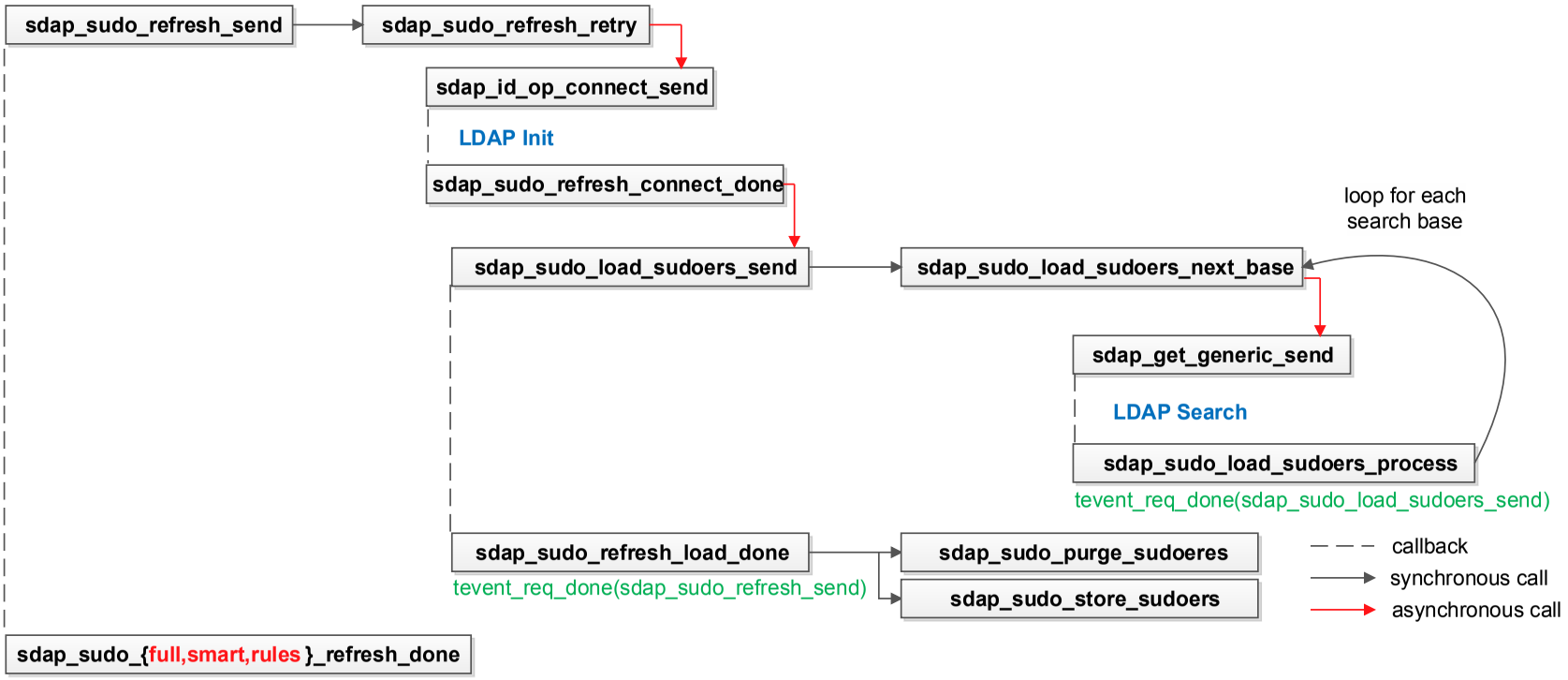
Modul `providers/ldap/sdap_async_sudo.c` je nejdůležitější částí LDAP SUDO pluginu, který provádí asynchronní stažení a uložení SUDO pravidel do sysdb. Jeho asynchronní činnost popisuje schéma 3.7. Asynchronní funkce toho modulu poté provádějí následující:

- `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.

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**[27]. 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`.

[27] aktuální čas (v době ukládání) + hodnota volby `entry_cache_sudo_timeout`, viz. man `sssd.conf`[28]

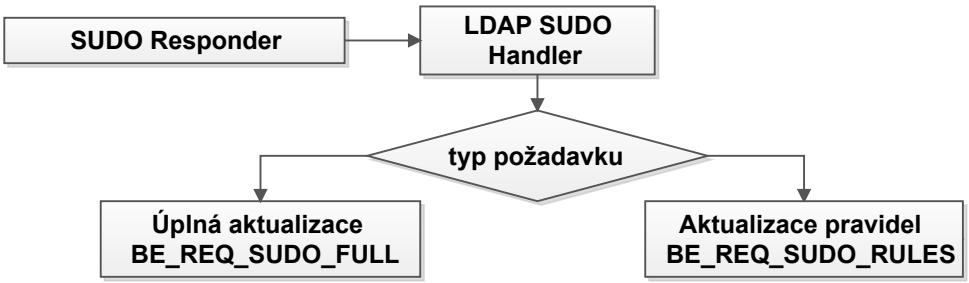


Obrázek 3.7: Schéma událostí asynchroního modulu `providers/ldap/sdap_async_sudo.c`.

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 plugin [29], 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 respoder žádá a pole pravidel, které chce responder aktualizovat[30]. SUDO responder může požádat o úplnou aktualizaci nebo o aktualizaci pravidel, která již v sysdb nejsou nadále platná.

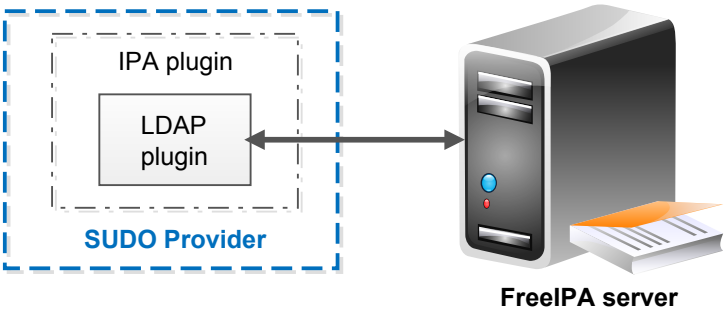


Obrázek 3.8: LDAP SUDO Handler.

3.5 IPA SUDO provider

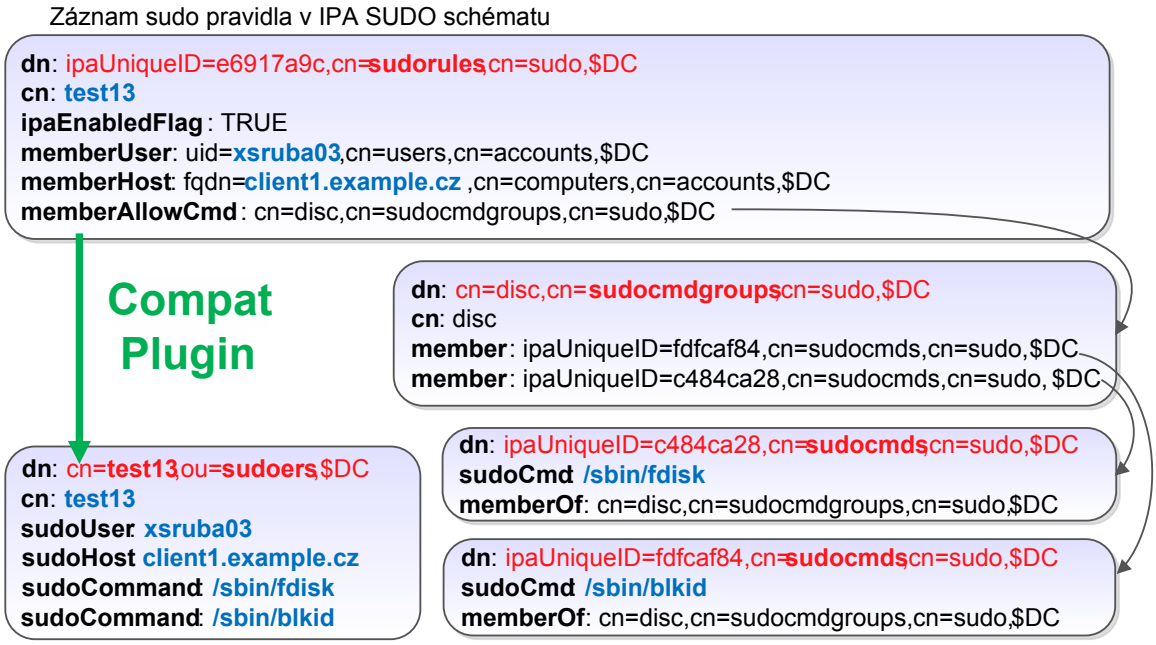
Pro SUDO providera je možné použít také ipa plugin[31], jestliže jsou SUDO pravidla uložena na IPA serveru. Problém ovšem je, že SSSD nedisponuje IPA pluginem pro SUDO providera, který by podporoval IPA SUDO schéma. Současná implementace tohoto pluginu je pouze obálka pro LDAP SUDO plugin. Samotný LDAP 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.

[29] `sudo_provider=ldap`
[30] pole jmen pravidel, tj. pole hodnot `cn` atributů
[31] `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.

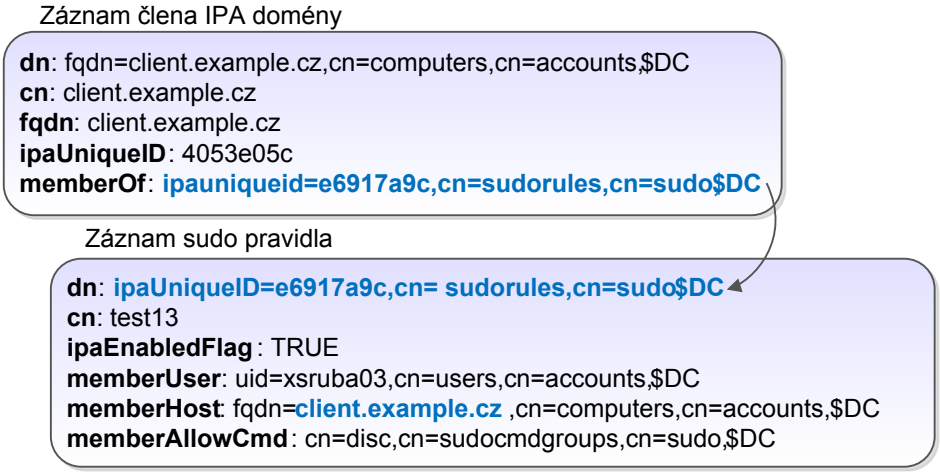
SUDO pravidla jsou tedy na serveru uložena dvakrát, i když v jiných schématech. Za předpokladu, že by SSSD mělo podporu pro nativní IPA SUDO schéma, byla by odstraněna režie compat pluginu. To znamená, že by nadále nebylo nutné provádět překlad **všech** SUDO pravidel. Překlad by byl přenesen ze serveru 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 zabýva kapitola 4.

3.6 Dereferenční dotaz

LDAP dotaz umožňuje získat záznam, popřípadě množinu záznamůz LDAP adreáře. Jestliže se nějaký atribut z těchto záznamu 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“ podporovat[32]. U Active Directory se tato technika označuje jako *Attribute Scope Queries*.[33]

[32] supportedControl: 1.3.6.1.4.1.4203.666.5.16
[33] 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 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 následujícími způsoby.

Klient jako člen IPA domény

Každý klientský počítač, který je členem IPA domény, 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 HBAC[1] 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 DNs[2] SUDO pravidel aplikovatelných na klientský počítač jehož doménové jméno je `client.example.cz`. Poté by LDAP dotaz mohl mít následující parametry:

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

[1] Host-Based Access Control
[2] Distinguished Names

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. Tyto 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 odkazů[3] na dané záznamy skuipin počítačů.

Specifikace klientského počítače

Je možné využít stejný přístup, který používá současná implementace LDAP SUDO providera. Pro výběr SUDO pravidla aplikovatelného na konkrétní počítač v IPA doméně jsou 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 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:

Prohledávána 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 tento druhý přístup. Detaily jsou poté popsány v úvodu sekce 4.2.

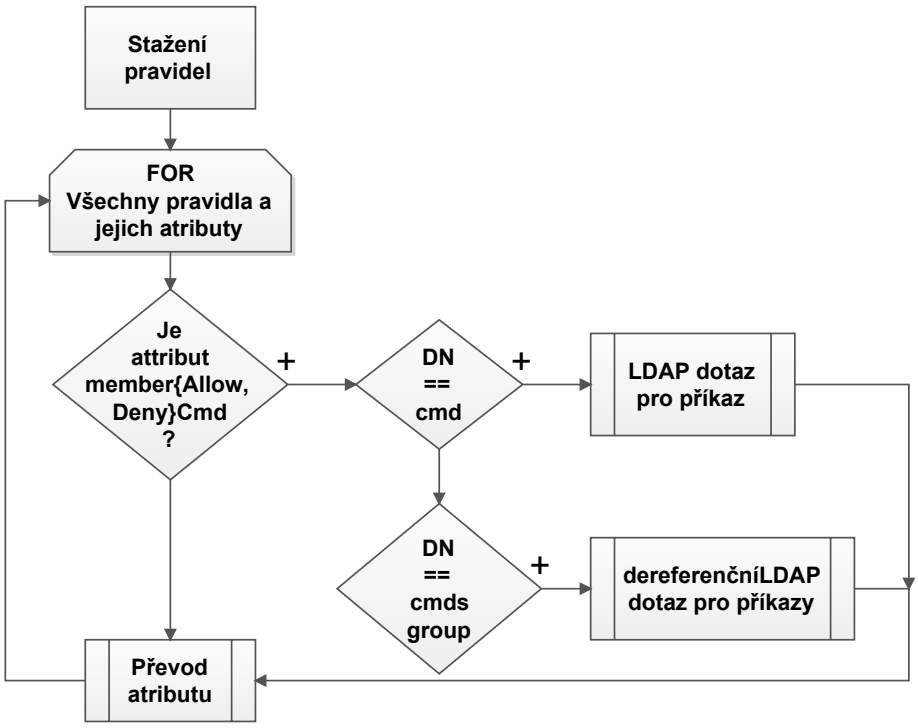
[3] Tato technika je blíže popsána v sekci 3.6

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`[4]. 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 popisuje diagram 4.1, je stažení všech pravidel aplikovatelných na daný počítač. To znamená konkrétní záznamy o SUDO pravidlech z kontejneru `cn=sudorules,cn=sudo,$DC`. Pro tyto záznamy je ovšem nezbytné ještě stáhnout příkazy. Bylo by tedy nutné iterovat před 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 příkazu, 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.

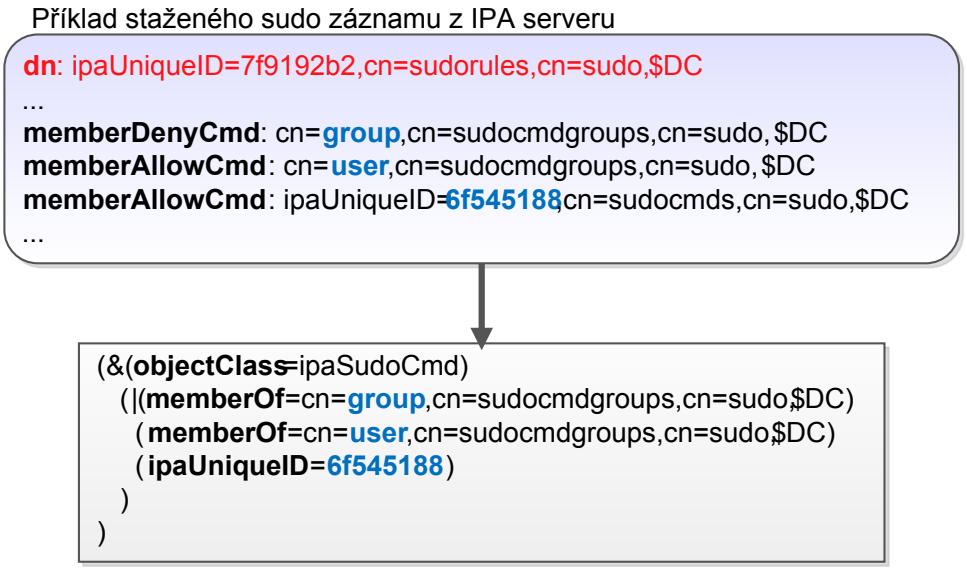
[4] prohledávanou oblast je možné specifikovat v `sssd.conf` pomocí volby `ldap_sudo_search_base`

2. Nejsou stahována žádná přebytečná SUDO pravidla, příkazy nebo jiné záznamy.

Při rozsáhlé databázi SUDO pravidel by to ovšem 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 záznamy projít a sestavit filtr pro LDAP dotaz, který stáhne všechny potřebné příkazy pro stažená pravidla.



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 je redukuje počet LDAP dotazů, nutných pro stažení všech záznamů potřebných pro překlad SUDO pravidel, na dva a zároveň nestahuje žádné přebytečné informace.

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 situaci, 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.

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 podporovou 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. Například z hodnoty atributu `memberUser`:

```
memberUser: cn=students,cn=groups,cn=accounts,$DC
```

je možné odvodit, že je pravidlo aplikovatelné na všechny uživatele skupiny „students“. U `*Category` atribtů je nutné provést pouze záměnu malých písmen za velká.

```
hostCategory: all => sudoCommand: ALL
```

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 typ `USER_GROUP` a `HOST_GROUP` není nutné rozlišovat, ovšem liší se v prefixu, který je k nové hodnotě nutdno přidat. Viz. příklady hodnot atributů `sudoUser` a `sudoHost`, které zobrazují tabulky 2.1 a 2.2.

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

Tabulka 4.1: Typy atributů a způsob jejich překladu.

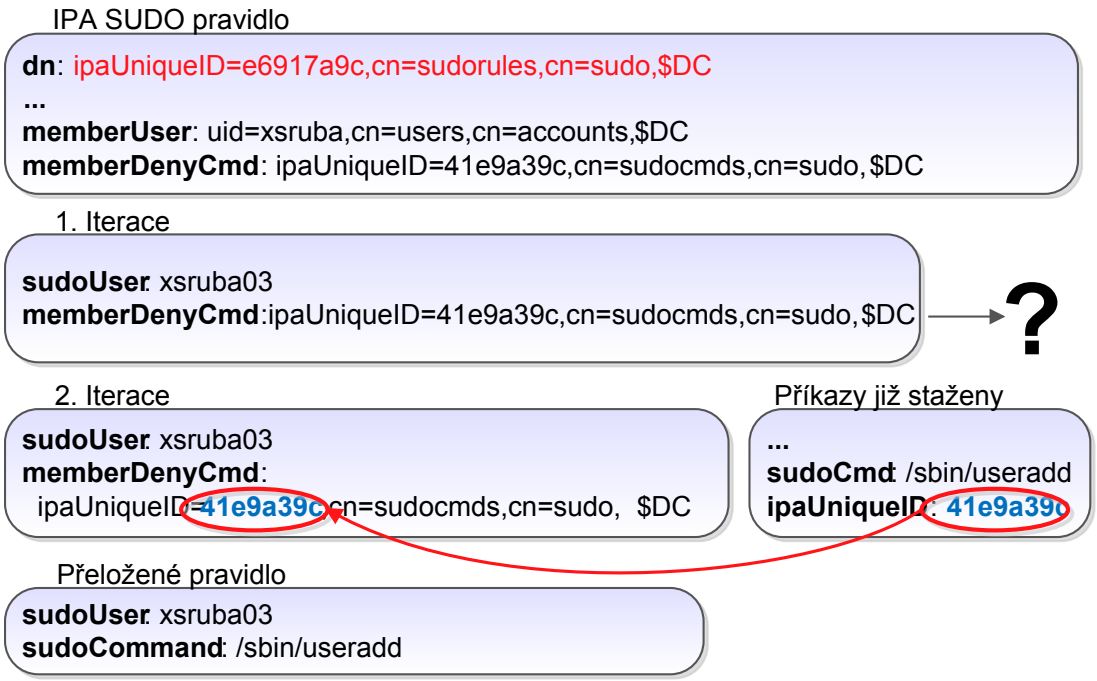
Pro algoritmus překladu pravidel byla vytvořena překladová tabulka, která definuje jakým způsobem má být proveden překlad jednotlivých jmen a hodnot atributů včetně příkladů, viz. Příloha A. Tato tabulka byla vytvořena na základě implementace compat pluginu a byla diskutována s vývojáři projektu FreeIPA. Při jejím sestavovaní byla také

objevena chyba, které se vyskytuje ve webovém rozhraní serveru FreeIPA. Ta umožňuje atributu `ipaSudoRunAsGroup` přiřadit libovolnou skupinu. Správně by mělo být možné tomuto atributu přiřadit pouze POSIX skupinu. Pro tuto chybu byl vytvořen ticket číslo 4314[5].

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.



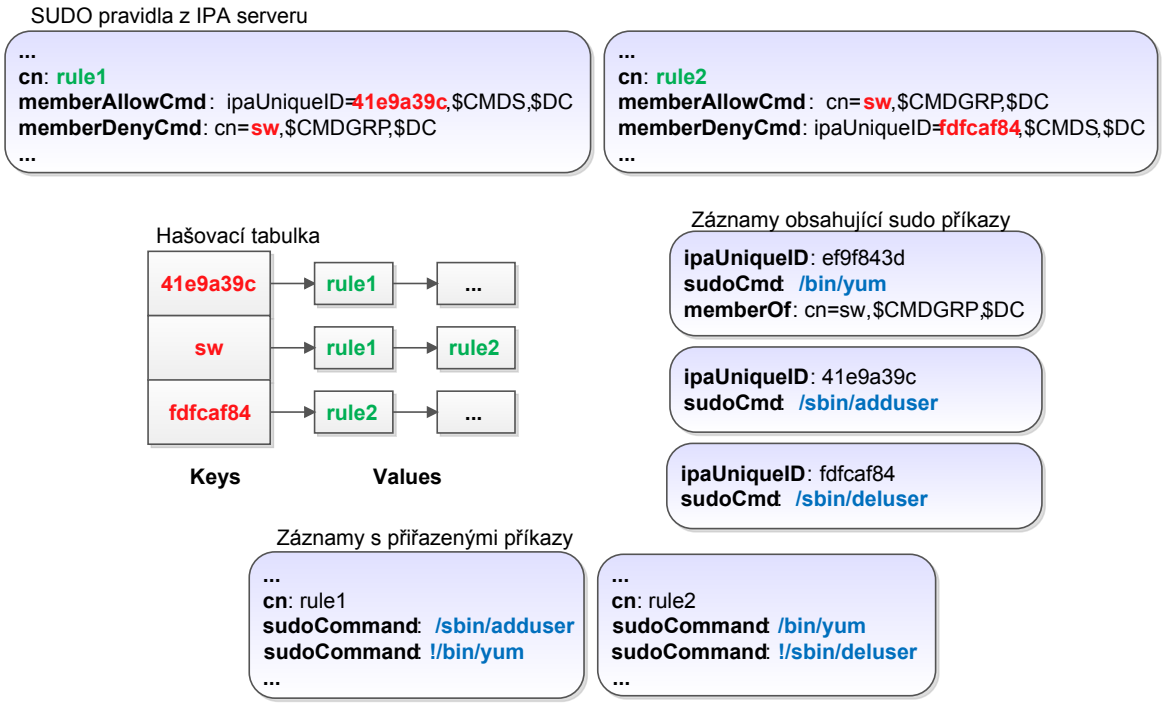
Obrázek 4.3: Dvouprůchodový překlad IPA SUDO pravidel

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 **neměnící se** množinou příkazů.

[5] https://fedorahosted.org/freeipa/ticket/4314#no1

Optimalizací tohoto přístupu by mohlo být uchování hodnot odkazů na příkazy a pravidla ve kterých se příkaz vyskytl. Stále je ovšem nutné N-krát iterovat množinou příkazu, kde N je počet odkazů na příkazy v záznamech SUDO pravidel. Tyto problémy je možné odstranit za pomoci použití hašovací tabulky. Tento přístup zobrazuje schéma 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. 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 vyskytnou volání funkcí, které by se nacházely v IPA SUDO providerovi. Tento přístup je sice realizovatelný[6], 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 tyto 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)
```

[6] První prototyp byl na této myšlence založen.

```
                    ...
  )
)
```

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

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 přidaných 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 ekvivalentní[7]. 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 zárovneň 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 popsaný v sekci 3.4.4. To ovšem vyžadovalo úpravu v synchronní funkci `sdap_sudo_refresh_load_done()`. Jestliže tento modul využívá IPA SUDO prvoder. 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 použít 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

[7] důvodem jsou mapy a záznamy, které nemají společné třídy

- `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
- `ipa_sudo_cmd.c` – sestavení LDAP fitru 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[8], 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

[8] http://cmocka.org/

Kapitola 5

Závěr

TODO: pochlub se co bylo potreba nastudovat: talloc, tevent, ldb ke teré není dokumentace.

- NSS, pam, ldb ke které neni 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 provideru 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 lepé, 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 hiearchical memory allocator [online]. http://is.muni.cz/th/359290/fi_b/, 2012 [cit. 2014-04-05].

[10] David Koňař. *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

Obsah CD

- `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