FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Analýza a nasazení opensource nástroje FreeIPA
Student:	Henrich Le
Vedoucí:	Ing. Jiří Milota
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

Správa přístupových údajů pro jednotlivé nástroje, Linux servery, VPN přístupy a další je jednotlivě obtížná a téměř neudržitelná z dlouhodobého hlediska, proto by bylo vhodné integrovat a připravit centralizovanou správu identit a autentizace. Jako vhodný opensource nástroj se jeví FreeIPA .
1) Prostudujte, navrhněte a implementujte automatizované nasazení FreeIPA.
2) Prostudujte možnosti integrace (a případně připravte funkční ukázku) FreeIPA
- s nástroji pro autentizací(Kerberos, LDAP),
- se správou DNS serveru,
- se správou asymetrických klíčů,
- s přístupovými právy na Linux serveru.
3) Implementujte základní operace nad FreeIPA platformou - přidání uživatele, odebrání uživatele, přidání DNS záznamu, apod.
4) Prostuduje, navrhněte a případně implementujte
- přidání Linux serveru,
- aktualizaci FreeIPA,
- řešení vysoké dostupnosti.
5) Prostudujte možnosti a případně navrhněte a implementujte propojení s externím poskytovatelem OpenID Connect.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 28. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Analýza a nasadenie opensource nástroja FreeIPA

Henrich Le

Katedra počítačových systémů
Vedúci práce: Ing. Jiří Milota

14. mája 2018

Poďakovanie

Rád by som poďakoval svojmu vedúcemu práce Ing. Jiřímu Milotovi za možnosť napísať túto prácu. Ďalej by som chcel poďakovať Jozefovi Mendovi za technické a štrukturálne konzultácie a v neposlednom rade Miriame Bernátovej za konzultácie týkajúce sa štylistickej úpravy textu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 14. mája 2018 ……

Odkaz na túto prácu

Le, Henrich. *Analýza a nasadenie opensource nástroja FreeIPA*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Táto práca je zameraná na nasadenie opensource nástroja FreeIPA. Rozoberá možnosti nasadenia nástroja FreeIPA v single deployment a v škálovanom deployment. V práci je analyzovaná problematika škálovateľnosti, vysokej dostupnosti, logovania, zálohovania a obnovy a zároveň práca ponúka riešenia týchto problematík. Pretože je značná časť práce o konfigurácií, sú v nej rozobrané základné princípy Ansible, čo je vhodná technológia poskytujúca jednoduchú automatizáciu a úpravy pomocou konfiguračných súborov. Ďalší cieľ práce je analýza možnosti prepojenia s externým poskytovateľom služby OpenID Connect, ktorá je často využívaná na poskytovanie identít. Výsledkom práce sú funkčné konfigurácie pre spustenie nástroja FreeIPA s vysokou dostupnosťou a škálovateľnosťou a automatizované operatívne úlohy. Konfiguračné skripty sú jednoducho spravovateľné a spustiteľné aj nie odborne znalým užívateľom.

Kľúčové slová FreeIPA, nasadenie, identita, autentikácia, autorizácia, Ansible, škálovateľnosť, vysoká dostupnosť, Open ID Connect

Abstract

This thesis is focused on the deployment of opensource tool FreeIPA. It analyzes options for deployment of FreeIPA in single deployment and also in multi-master deployment. Problems of scaling, high availability, logging, backup and restore and also their proposed solutions are discussed. Since major part of thesis is about configuration, basic principles of Ansible are described, as this is suitable tool for configuration. Another goal is connecting FreeIPA with an external provider of Open ID Connect service, as this service is widely used for providing identities. The most important outcomes of this thesis are configration scripts for deployment of FreeIPA with high availability, scaling and automating operative tasks. Configuration scripts are easily manageable and can be run without professional knowledge of FreeIPA.

Keywords FreeIPA, deployment, identity, authentication, authorization, Ansible, scalability, high availability, Open ID Connect

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Správa prístupových údajov pre užívateľov, servery a rôzne služby je jednotlivo veľmi obtiažna, a takmer neudržateľná z dlhodobého hľadiska. Preto je vhodné v prostredí, ku ktorému má prístup viacero užívateľov využívajúcich rôzne servery a služby zaviesť centralizovanú správu identít. Ako vhodný opensource nástroj sa javí FreeIPA.

Inšpiráciou na túto tému bakalárskej práce mi bola potreba zaviesť jednoduchú a časovo nenáročnú správu identít vo vývojovom prostredí v mojej práci. Práve z tohto dôvodu sa viac zameriam na prostredie vývoja vo väčšej firme, v ktorej má každý užívateľ svoju vlastnú identitu a zároveň nasadenie FreeIPA nie je vystavené priamo WAN.

Hlavnou motiváciou k tvorbe tejto práce bol fakt, že som sa s technológiami Ansible a FreeIPA ešte nestretol a spracovať ich do bakalárskej práce mi prišlo ako vhodná výzva.

FreeIPA poskytuje vlastnú LDAP directory, Kerberos, správu DNS záznamov, automatické udržiavanie aktuálnosti certifikátov a mnoho iných súčastí zaoberajúcich sa správou identít a udržania jej aktuálnosti. Nasadenie a samotná správa FreeIPA vyžaduje viac ako priemerne znalého užívateľa v správe systémov. Ale užívateľ, ktorý nástroj FreeIPA spravuje, nemusí úplne rozumieť tomu, čo a kde nastavuje, a prípadne chybná konfigurácia môže ústiť k neželanému správaniu serverov a rôznym bezpečnostným rizikám, ktoré sa neodhaľujú jednoducho.

V práci sa zameriavam na návrh nasadenia systému FreeIPA, rozoberám riešenie zálohovania, vysokej dostupnosti, škálovateľnosti a v neposlednom rade automatizáciu repetetívnych úkonov, v ktorých sa väčšina parametrov nemení a mení sa len server, na ktorom sú vykonané.

Ďalej sa v práci sústredím na spárovanie identít poskytovaných externým poskytovateľom identity, konkrétne poskytovateľom identity podľa špecifikácie Open ID Connect, ktorá je nadstavbou OAuth2.0.

Kapitola 1

Ciele práce

Cieľom práce je položenie teoretických základov pre nasadenie a pochopenie základných princípov a technológií používaných v sytéme FreeIPA. Ďalším cieľom je návrh nasadenia systému FreeIPA s možnosťou škálovania, vysokou dostupnosťou a pripravenosťou na rôzne incidenty, kedy bude potrebná obnova dát.

S prácou bude spojená sada automatizačných skriptov, ktoré vďaka jednoduchej konfigurácií budú schopné nasadiť systém FreeIPA v sieti, pripojiť klientské servery do oblasti spravovanej FreeIPA, vytvoriť užívateľov vo FreeIPA zo súboru vo formáte *csv*, priradiť ich do skupín, atď.

Taktiež budem v práci skúmať možnosti a pokúsim sa prepojiť systém FreeIPA s externým poskytovateľom identít pre služby s webovým rozhraním a prihlásením sa priamo do rozhrania FreeIPA na báze protokolu Open ID Connect a položím teoretické základy, s ktorými tento protokol pracuje.

Kapitola 2

FreeIPA

2.1 Čo je FreeIPA

FreeIPA je riešenie manažmentu bezpečnostných informácií a identít, ktoré v sebe spája množstvo technológií ako 389 Directory Server, MIT Kerberos, NTP, DNS, Dogtag (certifikačný systém).[1] Pozostáva z webového rozhrania a množstva administratívnych nástrojov v príkazovom riadku.

2.2 K čomu slúži FreeIPA

Vďaka integrovaným nástrojom vymenovaným v predošlej časti je možné pomocou FreeIPA postaviť bezpečné a relatívne jednoduché riešenie správy identít. Tieto identity je možné využiť naprieč veľkým spektrom služieb ako napríklad užívateľské účty, skupiny užívateľov alebo autorizácia na vykonávanie administrátorských príkazov (sudo) v Linux serveroch. Taktiež na samotných Linux serveroch a webových službách poskytujúcich autentikáciu a autorizáciu pomocou LDAP technológie.

Výhoda FreeIPA je v tom, že všetky administratívne nástroje zlučuje pod jednou strechou, a teda nie je potrebné riešiť vzájomnú kompatibilitu medzi jednotlivými nástrojmi. Zároveň pridáva prostriedky, ako webové rozhranie a príkazový riadok, ktoré uľahčujú prácu s týmito nástrojmi (pridanie DNS záznamov, pridanie užívateľa alebo vytvorenie skupiny v adresárovom serveri, automatické pripojenie vzdialeného disku pri prihlásení užívateľa na Linux server, atď.).

Vo FreeIPA je možné využiť integrovaný doménový server, do ktorého je možno nakonfigurovať klientov FreeIPA tak, aby automaticky odoslali aktualizáciu DNS serveru pri zmene vlastnej IP adresy [2, kapitola 33.5.]. FreeIPA dokáže odosielať aktualizácie aj externému DNS serveru využitím *nsupdate* a *TSIG*. [2, kapitola 33.10.]

A v neposlednom rade má FreeIPA nástroje na správu certifikátov, kde sa nástroj Dogtag [3] stará o správu certifikátov zo strany serveru a nástroj Certmonger [4], ktorý má na starosti aktualizáciu certifikátov zo strany klienta.

Kapitola 3

Kľúčové princípy

3.1 Technológie

V tejto kapitole popíšem technológie využívané vo FreeIPA a technológiu Ansible, ktorú som zvolil pri implementácii konfiguračných skriptov v mojej bakalárskej práci.

Taktiež popíšem priebeh autentikácie pomocou protokolu Open ID Connect, pretože pri snahe prepojiť FreeIPA s externým poskytovateľom identít OIDC je dôležité vedieť ako tento protokol funguje.

3.1.1 Ansible

Ansible slúži k hromadnej konfigurácii viacerých serverov z jedného hlavného serveru pomocou technológie SSH. Je napísaný v jazyku Python[5] a stavia na princípe modulov a úloh.

Ansible zakladá na tom, že každý server je v nejakom stave. Preto sa v Ansible syntaxi uvádza želaný stav, a nie kroky, ktorými tento stav dosiahnuť.

Hlavným programom pre Ansible je tzv. *playbook*, v ktorom sú definované rôzne úlohy, ktoré sa vykonávajú na princípe modulov. Napríklad kopírovanie súborov (lokálne alebo zo vzdialeného zdroja), vyhľadanie riadku kódu v danej konfigurácii podľa regulárneho výrazu a jeho zmena na želaný text, správa firewalld, inštalácia balíčku pomocou package manager a iné. V čase písania tejto práce bolo 1652 oficiálnych modulov[6].

Zoznam serverov, s ktorými Ansible pracuje sa nazýva *inventory*. V *inventory* je možné definovať nielen adresy serverov (či už FQDN alebo IP adresa), ale aj rôzne premenné špecifické pre tento server. Napríklad východzí užívateľ, pomocou ktorého sa Ansible na server pripája alebo port, cez ktorý sa má na SSH server pripojiť. Zároveň je možné definovať meno špecifické pre Ansible a vedľa definovať adresu tohto serveru.

Celý *inventory* je písaný v *ini* alebo *yaml* formáte, v ktorom môžeme definovať skupiny serverov a premenné špecifické pre danú skupinu. Názov

skupiny využíva aj *playbook*, v ktorom definujeme, na ktorú skupinu má byť spustený.

Keďže Ansible dokáže konfigurovať celý systém, občas je potrebné dodatočné heslo alebo iná inak utajovaná informácia. Namiesto toho, aby boli takéto heslá uložené v čistom texte je možné tieto informácie šifrovať. Na šifrovanie slúži *ansible-vault*, čo je možnosť zašifrovať štruktúrovaný text (napríklad vo formáte *yaml*), a tak predísť nechcenému úniku informácií. *Ansible-vault* zašifruje priamo súbor, vďaka čomu môže byť súbor ďalej distribuovaný alebo uložený v správe zdrojových súborov ako napríklad Git. Pri využívaní tohoto súboru stačí zadať heslo pre dešifrovanie pri spustení *playbook*. Viac o *ansible-vault* v jej dokumentácii [7].

Ansible taktiež podporuje Jinja2 šablóny, šablónový jazyk pre jazyk Python, ktorý dokáže na základe premenných vytvoriť textový súbor v želanej štruktúre. Taktiež je možné používať kontrolné štruktúry ako napríklad podmienky a forcykly[8, kapitola 3.8.]. Podrobnejšie sa šablónam v Ansible venuje oficiálna dokumentácia Ansible, dostupná online [9].

Zároveň je možné Ansible rozšíriť o vlastné moduly napísané v ľubovoľnom jazyku, ktorý podporuje JSON formát[10], a tak využiť Ansible na nie úplne štandardné úlohy.

Viac sa Ansible venuje oficiálna dokumentácia Ansible, dostupná online [11] a Hochstein a Moser v „Ansible: Up and Running“ [10].

3.1.2 Apache2

Apache2 je opensource webový server pre moderné operačné systémy. Je založený na moduloch, ktoré zabezpečujú jeho jednoduchú rozšíriteľnosť.

Pre túto prácu je zaujímavá direktíva `<Location>`, ktorá slúži na obalenie jednotlivých direktív, ktoré sa aplikujú pre lokáciu s daným názvom.

Teda je možné definovať rôzne moduly, ktoré budú platné len pre nejakú lokáciu. Táto funkcia je obvzlášť vhodná pre autentikačné moduly.

Podrobnejšie sú direktívy a moduly opísané v oficiálnej dokumentácii Apache2 [12].

3.1.3 Automount

Technológia automount, označovaná aj ako autofs, funguje ako služba, ktorá na požiadanie automaticky pripojí súborový systém na server. Tento súborový systém môže byť pripojený lokálne alebo vzdialene pomocou protokolu NFS.

Automount využíva tzv. *mapy*, pomocou ktorých určuje odkiaľ a kam sa má systém pripojiť. Podrobnejšie túto technológiu popisuje administrátorský manuál od Sun Microsystems [13].

Služba automount je vo FreeIPA priamo integrovaná, teda na serveroch je možné hneď nastaviť rôzne mapy, podľa ktorých sa môžu vzdialené súborové systémy z NFS serveru pripojiť. Jediný potrebný krok od administrátora

je nastavenie služby automount na klientskom serveri tak, aby tieto mapy využíval. [2, kapitola 34.2.1]

3.1.4 Kerberos

Kerberos je systém vyvinutý v MIT ako súčasť projektu Athena, špecifikovaný v „Kerberos: An Authentication Service for Open Network Systems“ [14], a využíva sa na overenie identity *principal*. Principal sa v terminológii Kerberos chápe ako jedinečná identita, ktorej môžeme prideliť *ticket*. Ticket samotný je sada prístupových údajov klienta pre špecifickú službu, napríklad HTTP.

Principal je väčšinou zadaný vo formáte *user/instance@REALM*, kde user je názov, instance je inštancia (slúži na oddelenie jednotlivých služieb, napr. httpd, no každá je na inom stroji) a REALM je sféra alebo sieť, v ktorej sa Kerberos vyskytuje.

Kerberos je založený na princípe dôvery tretej strane, pretože tickety, ktoré vydáva, sú vydávané prostredníctvom **KDC** (Key Distribution Center). To pozostáva z autentikačného serveru a **TGS** (Ticket Granting Server). Prvý ticket, ktorý užívateľ dostane je **TGT** (Ticket Granting Ticket) a slúži na overenie identity principal-u a ďalšiu komunikáciu s **TGS**.

Autentizácia v protokole Kerberos prebieha takto:

1. a) Klient odošle nešifrovaný požiadavok na autentikačný server, v ktorom sa nachádza:
   - identita klienta,
   - identita TGS,
   - IP adresa klienta,
   - požadovaná životnosť TGT.

   Autentikačný server v tomto kroku v databáze overí, že existuje užívateľ s danou identitou. Pokiaľ sa nevyskytne chyba, vygeneruje náhodný kľúč príslušný k relácii.

   b) Autentikačný server následne klientovi odošle dva tickety, kde prvý je **TGT**, ktorý obsahuje:
   - ID klienta,
   - ID TGS,
   - IP adresu klienta,
   - životnosť TGT,
   - **kľúč relácie s TGS**.

   Tento ticket je šifrovaný súkromným kľúčom TGS.

   Druhý ticket obsahuje:

- ID TGS,
- timestamp,
- životnosť - rovnaká ako v TGT,
- **kľúč relácie s TGS**.

Tento ticket je šifrovaný súkromným kľúčom klienta.

ticket šifrovaný kľúčom klienta sa dešifruje pomocou hesla klienta a soli, pozostávajúcej z principal-u (user/instance@REALM). Z toho vyplýva, že heslo sa nikam neodosiela a až v tomto kroku je implicitne overené.

2. Na overenie identít principal-ov sa používa *authenticator*, v ktorom sa nachádza ID principal-a, ktorý správy odosiela, a timestamp, aby sa predišlo útokom z opakovania požiadavkov. Celý tento authenticator je zašifrovaný kľúčom relácie, ktorý nikdy nebol cez sieť posielaný v nešifrovanej podobe.

   Komunikácia s **TGS** prebieha takto:

   a) Klient odošle **TGS** tri správy:
      - **TGT**, šifrovaný súkromným kľúčom TGS.
      - Authenticator, teda ticket, v ktorom je klientove ID a timestamp. Táto správa je šifrovaná pomocou **kľúča relácie s TGS**.
      - Požiadavok s ID služby, ktorú chce klient využívať. Táto správa je nešifrovaná.

      TGS získa TGT, ktorý môže dešifrovať svojím súkromným kľúčom. Tento TGT obsahuje kľúč relácie s TGS, a tým pádom môže dešifrovať aj Authenticator zašifrovaný klientom. Toto určuje klientovu validitu. Následne TGS porovná ID klienta z TGT s tou, ktorá je v Authenticator-e, porovná timestamp z TGT a Authenticator, skontroluje životnosť určenú v TGT a IP adresy povolené v TGT a adresu, z ktorej mu tento požiadavok prišiel. Ďalej v databáze overí, či existuje ID služby, ktorú klient požaduje. Ak všetko prebehne v poriadku, TGS vygeneruje náhodný kľúč relácie k službe, ktorú klient žiadal.

   b) TGS následne odošle klientovi dve správy, z toho jedna je **ticket pre službu** žiadanú klientom, a obsahuje:
      - ID klienta,
      - ID služby,
      - zoznam IP adries (v rovnakom duchu ako už vyššie zmienené),
      - timestamp,
      - životnosť ticketu,
      - **kľúč relácie služby**.

Tento ticket je šifrovaný **súkromným kľúčom služby**.

Druhá správa obsahuje:

- ID služby,
- timestamp,
- životnosť ticketu,
- **kľúč relácie služby**,

a je šifrovaná pomocou **kľúča relácie s TGS**.

3. a) Klient odošle dve správy službe, ku ktorej chce získať prístup. Prvá správa je ticket pre službu, šifrovaný súkromným kľúčom služby, a druhá je authenticator klienta, ktorý pozostáva z ID klienta a timestamp. Tento authenticator je šifrovaný pomocou **kľúča relácie služby**.

   Služba ticket dešifruje pomocou svojho súkromného kľúča, z ktorého získa **kľúč relácie služby**, vďaka ktorému dešifruje authenticator.

   Podobne ako TGS porovná identitu klienta z ticketu a authenticatora, či sa v zozname IP adries z tiketu nachádza IP adresa odkiaľ bol požiadavok odoslaný, porovná aj timestamp a či je ticket stále platný (životnosť).

   b) Služba po úspešnom overení odošle klientovi authenticator pre potvrdenie svojej identity. V tomto authenticator-e sa nachádza ID služby a timestamp, zašifrované pomocou **kľúča relácie služby**.

Všetky ďalšie požiadavky na túto službu používajú ticket, poskytnutý od TGS pokiaľ nevyprší jeho životnosť, ktorá je v ňom definovaná.

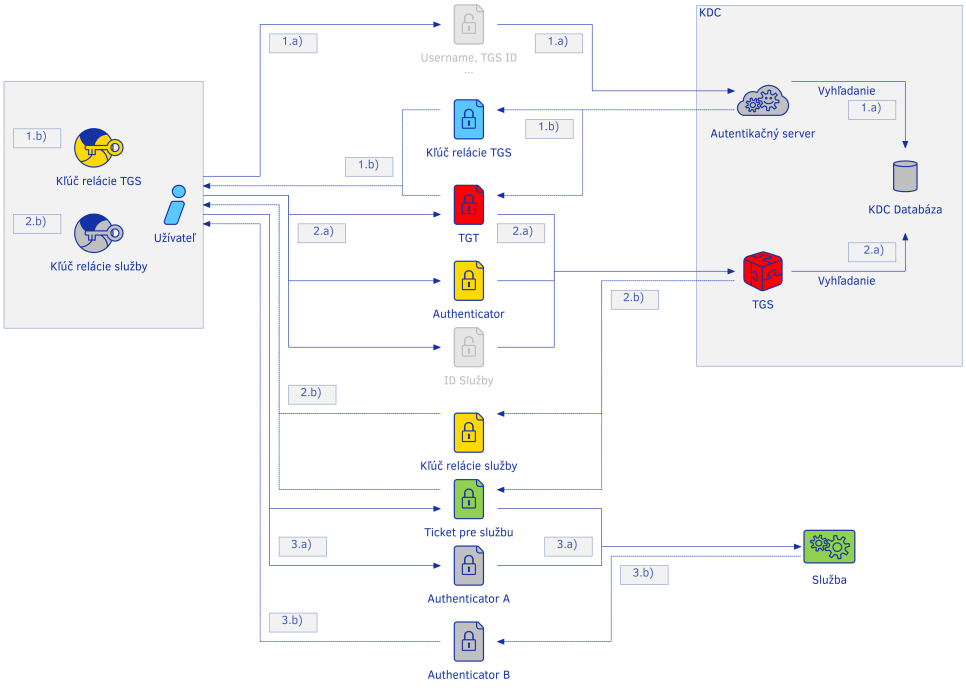
Obrázok 3.1 celý tento proces vizualizuje. V obrázku je farebne rozlíšené, akým kľúčom je správa šifrovaná a v ktorom kroku sa prípadné kľúče relácie získavajú.

Podrobnejšie sa tejto téme venuje špecifikácia RFC4120[15] a Garman v „Kerberos: The Definitive Guide“[16].

3.1.5 LDAP

Adresárový server je server, ktorý umožnuje ukladať rôzne (zvyčajne textové) typy dát, ktoré sú spojené do objektov. Na rozdiel od relačných databáz je optimalizovaný pre časté vyhľadávanie a nie tak časté vkladanie, alebo úpravu už existujúcich záznamov. Stavia na princípe podobnom adresárovej štruktúre (od toho názov „adresárový“), a teda každý záznam môže mať maximálne jedného rodiča a neobmedzene veľa synov (podobne ako adresáre v súborovom systéme). Každý záznam je tvorený sadou atribút.

Ako s adresárovým serverom komunikovať je definované v protokole LDAP. Protokol LDAP taktiež definuje, ako majú dáta v adresárovom serveri vyzerať,



Obrázok 3.1: Autentikácia v Kerberos

požiadavky na komponenty pre tvorbu záznamov v adresárovom serveri a ukazuje ako sa používajú primitívne dátové typy na tvorbu atribútov. Z tohoto možno usúdiť, že protokol LDAP definuje celý adresárový server a ako tento server funguje a je s ním teda úzko spätý.

Protokol LDAP je verejný, teda preň existuje veľa implementácii. FreeIPA vo svojej implementácii používa adresárový server 389 Directory server, ktorý je vyvinutý spoločnosťou Red Hat Inc. pre Linux. [17]

Väčšina dát v LDAP je uložená pomocou atribútov, ktoré sú spojené v zázname (*entry*). Typy atribútov a ich názvy sú definované v *objectClass*.

Príklad záznamu v **LDIF** (LDAP Data Interchange Format) formáte:

```
dn: sn=Doe,ou=people,dc=example,dc=com
objectclass: person
sn: Doe
cn: Joe Doe
```

V jednom zázname môže byť *objectClass* definovaný viackrát. V zásade existujú dva typy ObjectClass jeden je `STRUCTURAL` a druhý je `AUXILIARY`, kde prvý je povinný, pretože pre záznam definuje čo musí a čo môže obsahovať, a druhý definuje rôzne pomocné atribúty, ktoré rozširujú možnosti záznamu.

ObjectClass-y a atribúty môžu byť spolu definované v *schéme*, ktorá má podobný význam ako schéma v relačnej databáze, teda definuje z čoho sa celá

databáza skladá, no pretože dáta v adresárovom serveri sú usporiadané inak ako v relačnej databáze, aj schéma v LDAP je definovaná podstatne inak. Každá schéma musí obsahovať:

- syntax atribút - definuje typy dát, ktoré budú v adresárovom serveri uložené,
- pravidlá porovnávania - definuje, ako sa budú dáta v adresárovom severi porovnávať,
- typy atribút - definuje atribúty a ich názvy, ktoré sa budú v záznamoch používať,
- objectClass-y - definuje, aké objectClass-y budú v záznamoch a aké atribúty sú pre objectClass povinné.

Schéma môže mať aj množstvo nepovinných elementov, ktoré môžu ďalej obmedziť štruktúru dát v adresárovom serveri. Viac o schémach v RFC4512 [18, kapitola 4.].

Samotné záznamy sú uložené v organizačnej štruktúre **DIT** (Data Information Tree), vďaka ktorej môžeme rozlíšiť typy jednotlivých záznamov, teda napríklad záznam typu `person` bude mať rodiča typu `people` a synov typu `customer` a `employee` a záznam typu `item` bude mať rodiča typu `inventory`.

Záznamy využívané k nastaveniu organizačnej štruktúry (v analógii s adresárovým systémom, priečinky) väčšinou využívajú objectClass *organizationalUnit* a môžu byť referencované atribútom „ou=“.

K jednotlivým záznamom v adresárovom serveri pristupujeme pomocou atribútov a to znamená, že každý záznam musí mať jedinečný atribút alebo skupinu atribútov, vďaka ktorým naň môžeme odkázať vo svojej úrovni DIT (analógia s adresárovým systémom je, že nemôžu existovať dva súbory s rovnakým názvom v jednom priečinku). K tomuto účelu slúži **RDN** (Relative Distinguished Name). Pre prístup k záznamu musíme uviesť RDN a RDN jeho rodiča a všetkých ďalších rodičov. Reťaz týchto RDN sa nazýva **DN** (Distinguished Name). Ako analógiu je možno RDN prirovnať k relatívnej ceste k súboru a DN ako cestu absolútnu.

Viac sa tejto téme venuje navrhovaný štandard RFC4511[19].

3.1.6 DNS

DNS (Domain Name System) je protokol, ktorý slúži na preklad IP adries na lahšie zapamätateľné adresy názvov serverov (doménové mená) a naopak.

DNS používa **zóny**, ktoré je možno chápať ako zoznam IP adries a doménových mien pre určitú doménu, subdoménu alebo len jej časť.

V DNS existuje množstvo druhov záznamov. Ako príklad uvediem záznam typu A, čo je záznam v databáze DNS serveru, v ktorom je uložená IPv4 adresa na dotazovaný hostname. Jeho IPv6 ekvivalent je AAAA.

Pri ohľade na nasadenie FreeIPA je zaujímavý typ záznamu **SRV**, ktorý určuje port, adresu (alebo doménové meno) a prioritu v DNS zóne. Vďaka tomuto typu záznamu nie je potrebné serveru alebo službe priamo určiť IP adresy (prípadne doménové mená) a porty, na ktorých jednotlivé služby nájsť.[20]

Toto je obzlášť užitočné pri nasadení FreeIPA, kde pri inštalácií klienta FreeIPA bez explicitného uvedenia serveru (v tomto kontexte je to server, na ktorom sú dáta a služby FreeIPA), dokáže program pre inštaláciu rozpoznať jednotlivé služby vďaka záznamom SRV v DNS serveri.[2]

Využitie tohto typu záznamu je bližšie vysvetlené v sekcii 4.2.1.

Vyhľadávaniu IP adresy pomocou doménového mena sa v tejto práci venovať nebudem, keďže v tomto prípade bude FreeIPA nasadená v privátnom rozsahu, teda na lokálnej sieti.

Obsluha DNS je vo FreeIPA implementovaná pomocou programu BIND9 a jeho pluginu bind-dyndb-ldap[21], ktorý využíva LDAP server FreeIPA ako databázu.

FreeIPA poskytuje podporu **DNSSEC**, čo je protokol, v ktorom sú záznamy podpísané súkromným kľúčom poskytovateľa služby a rozšifrované jeho verejným kľúčom. Toto poskytuje ochranu pred podvrhnutými záznammi a tiež bezpečné preloženie adresy.

3.1.7 SSSD

SSSD (System Security Services Daemon) je systémový démon, ktorého primárna funkcia je poskytnúť prístup službám k lokálnym alebo vzdialeným zdrojom identít - LDAP, IdM, Active Directory, Kerberos. Dokáže si udržovať cache a má taktiež offline podporu, teda nie každý autentikačný požiadavok je posielaný cez sieť. Toto môže značne urýchliť autentikáciu a zároveň vďaka offline podpore nie je problémom ani dočasný výpadok u poskytovateľa identít.

3.1.8 Open ID Connect

Protokol **Open ID Connect** je založený na protokole OAuth2.0 a definuje spôsob, akým delegovať overenie identity na tretiu stranu, a tým pádom zbaviť klienta (v tomto prípade aplikáciu vyžadujúcu identitu) OIDC potreby spravovať identity. Tretia strana sa v tomto prípade nazýva **IdP** (Identity Provider) alebo **OP** (OIDC Provider) a aplikácia výžadujúca identitu je **RP** (Relaying Party) alebo klient. Užívateľ je koncový používateľ, ktorého identita je poskytovaná.

Klientovi poskytuje IdP identitu užívateľa pomocou zakódovaného **JWT** (JSON Web Token), ktorý sa nazýva **ID token** [22, kapitola 1.2]. Vďaka nemu môže RP vyžiadať napríklad **Access token**, ktorý slúži pre prístup k chráneným zdrojom, ako napríklad web API aplikácie [23, kapitola 1.4].

Samotná autentikácia užívateľa prebieha na strane IdP a v protokole nie je uvedený špecifický typ overenia. Teda celá autentikácia je prenechaná na IdP (meno/heslo, pin, hardware, biometrika, …) a protokol OIDC rieši len „transportnú“ vrstvu.

ID token je vyžiadaný pomocou špecifikácie OAuth2.0 (využitie autorizačného serveru) a existuje viacero postupov ako oň požiadať. Konkrétne sú tri a to:

- Authorization Code Flow,
- Implicit Flow,
- Hybrid Flow.

Autorizačný server poskytovateľa OIDC je server, ktorý poskytuje RP autorizačný kód alebo Access token po tom, čo bol užívateľ autentikovaný a autorizovaný [23, kapitola 1.1].

Autorizačný server OP pozostáva z **Authorization Endpoint** a **Token Endpoint**, kde sú oba HTTP zdroje. Authorization Endpoint je koncový bod využitý klientom pre získanie autorizácie užívateľa pomocou presmerovania. Token Endpoint je koncový bod, ktorý je využívaný v Authorization Code Flow a Hybrid Flow, kde klient môže získať ID token alebo Access token výmenou za autorizačný kód [23, kapitola 3]. **Autorizačný kód** je kód, ktorý získa klient po tom, čo bol užívateľ autentikovaný v autorizačnom serveri [23, kapitola 1.3.1].

Authorization Code Flow je postup určený pre tradičné webové aplikácie, ktoré môžu bezpečne udržať **Client Secret** medzi klientom a autorizačným serverom. Autorizačný server odošle autorizačný kód klientovi, ktorý ho môže využiť pre získanie ID tokenu z Token Endpoint a priamo získať Access token. Tento postup poskytuje výhodu v tom, že neodhaľuje žiaden token agentovi, pomocou ktorého sa užívateľ pripája ku klientovi (napr. webový prehliadač). Autorizačný server môže pred výmenou autorizačného tokenu za Access token autentikovať samotného klienta[22, kapitola 3.1.].

Postup Authorization Code Flow:

1. klient pripraví autentikačný požiadavok so želanými parametrami o užívateľovi,
2. klient odošle tento požiadavok autorizačnému serveru,
3. autorizačný server autentikuje užívateľa,
4. autorizačný server získa od užívateľa súhlas o odoslaní identity klientovi/autorizuje klienta,

5. autorizačný server presmeruje užívateľa späť na klienta spolu s autorizačným kódom,

6. klient požiada pomocou autorizačného kódu o získanie ID token a Access token od Token Endpoint,

7. klient zvaliduje ID token, a tým získa identifikátor užívateľa.

Implicit Flow, alebo implicitný postup, je postup určený hlavne pre klientov implementovaných vo webovom prehliadači pomocou skriptovacích jazykov. Pri používaní implicitného postupu sú všetky tokeny odoslané pomocou Authorization Endpoint, teda Token Endpoint je nevyužitý. Access token, aj ID token sú priamo odoslané klientovi (vo webovom prehliadači), a teda môžu byť odhalené koncovému užívateľovi alebo programu, ktorí majú k agentovi (napr. webový prehliadač) prístup. Autorizačný server klienta neautentikuje[22, kapitola 3.2.].

Postup Implicit Flow:

1. klient pripraví autentikačný požiadavok so želanými parametrami o užívateľovi,

2. klient odošle požiadavok autorizačnému serveru,

3. autorizačný server autentikuje užívateľa,

4. autorizačný server získa od užívateľa súhlas o odoslaní identity klientovi/autorizuje klienta,

5. autorizačný server presmeruje užívateľa späť na klienta spolu s ID token a pokiaľ bol vyžiadaný, tak aj Access token,

6. klient zvaliduje ID token, a tým získa identifikátor užívateľa.

Pri použití **Hybrid Flow** je možné, aby niektoré tokeny boli získané z Authorization Endpoint a niektoré z Token Endpoint[22, kapitola 3.3.].

Postup Hybrid Flow:

1. klient pripraví autentikačný požiadavok so želanými parametrami o užívateľovi,

2. klient odošle požiadavok autorizačnému serveru,

3. autorizačný server autentikuje užívateľa,

4. autorizačný server získa od užívateľa súhlas o odoslaní identity klientovi/autorizuje klienta,

5. autorizačný server presmeruje užívateľa späť na klienta spolu s autorizačným kódom a ďalšími parametrami závislými na Response Type (určuje typ získania ID tokenu, môže to byť jeden z vyššie uvedených postupov alebo registrované rozšírenie),

6. klient požiada o odpoveď Token Endpoint použitím autorizačného kódu,

7. klient dostane ID token a Access token,

8. klient zvaliduje ID token, a tým získa identifikátor užívateľa.

3.2 Koncepty

3.2.1 Ansible roles

Ansible roles, alebo role Ansible, je best practice používaný v prostredí Ansible. Zakladá hlavne na tom, že každý server má v sieti nejakú rolu, takže musí byť správne nastavený, aby fungoval podľa očakávaní. Výhoda tohto prístupu spočíva v tom, že prostredie, s ktorým pracujeme je homogénne, takže konfiguračný súbor pre httpd je vždy v jednej zložke a databázové servery sa vždy zálohujú rovnakým spôsobom.

Tieto role sú priamo podporované v Ansible. Stačí pridať do koreňového adresára playbook-u priečinok „roles“ a v ňom pripraviť príslušnú adresárovú štruktúru pre želané role.

Štruktúra pozostáva z priečinku úloh, základných premenných, premenných, ktoré sa môžu meniť v závislosti na deploymente, priečinku so súbormi špecifickými k role, šablón pre rolu, manipulátorov pre rolu a metadát, v ktorých môžu byť napríklad prerekvizity.

Každý z týchto priečinkov musí obsahovať súbor `main.yml`, v ktorom sú definované jednotlivé úlohy, prerekvizity, premenné atď., ktoré sú príslušné k danej role a priečinku, v ktorom sa nachádzajú.

Túto štruktúru aj s príkladmi je možno preštudovať v tabuľke 3.1.

Podrobnejšie sa tejto téme venuje Hochstein a Moser v „Ansible: Up and Running“ [10] v kapitole č. 7 alebo dokumentácia Ansible, dostupná online [24].

3.2.2 Single Deployment

Pod pojom Single deployment sa chápe nasadenie aplikácie na jednom serveri, ktorý obsluhuje všetky požiadavky od klientov. V prostredí FreeIPA to znamená, že všetky možnosti, ktoré FreeIPA ponúka budú výhradne na tomto jednom stroji, ako napríklad NFS server, DNS server alebo Dogtag pre správu certifikátov.

Tabuľka 3.1: Štruktúra priečinkov pre role Ansible

Názov	Obsah	Príklad
tasks	Zoznam úloh, ktoré sa majú vykonať.	Web server potrebuje k svojmu behu napríklad inštaláciu Tomcat-u, no databázový server potrebuje mať nainštalovanú službu MongoDB.
defaults	Základné premenné pre rolu.	Adresa databázového serveru pre webservery.
vars	Dodatočné premenné pre rolu.	Databázový server v Európe sa zálohuje do inej lokalitky ako server v Amerike.
files	Súbory, ktoré možno nasadiť pomocou role.	Základná konfigurácia pre službu httpd.
templates	Šablóny, ktoré rola využíva.	Konfigurácia internetového rozhrania, aby používala statickú IP adresu definovanú napríklad v inventári.
handlers	Manipulátory spojené s rolou.	Reštartovanie služby HTTP pri notifkovaní inou úlohou.
meta	Meta dáta, v ktorých môžu byť zahrnuté napríklad prerekvizity.	Predtým, ako začne nasadenie Tomcat web serveru je možné využiť napríklad rolu Java serveru a pokračovať len s konfiguráciou špecifickou pre Tomcat a aplikáciu samotnú.

3.2.3 Replikácia

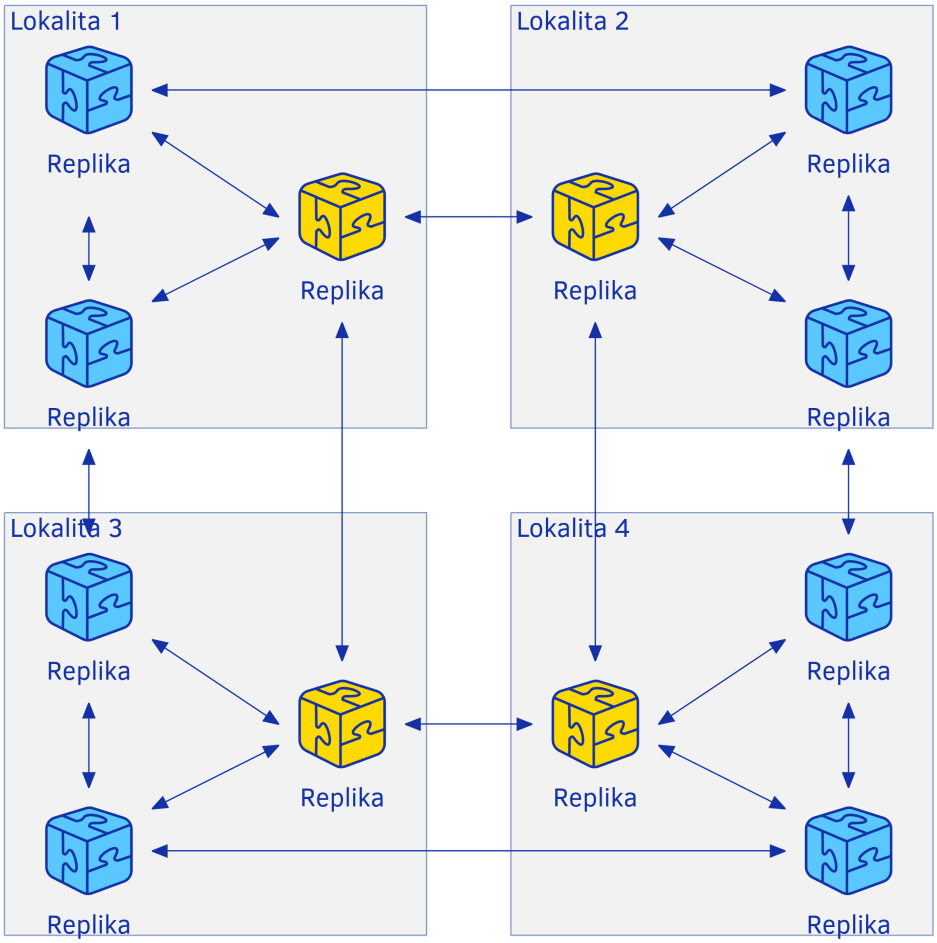
FreeIPA ponúka možnosť replikácie, ktorá je využiteľná najmä v prípade potreby zálohovania, škálovania alebo vysokej dostupnosti.

Replikácia znamená, že dáta definované v hlavnom serveri sa pravidelne, prípadne na požiadanie, skopírujú na server, na ktorom bolo nastavené, že sa má replikovať a naopak. Server, na ktorom je nastavená replikácia sa nazýva *replika*.

Replikácia je obojstranná a pri registrovaní replikácie sa vytvoria dve replikačné dohody medzi serverom definovanom v príkaze na replikáciu a serverom, z ktorého bol príkaz spustený. Prvá replikačná dohoda je o adresárovom serveri a druhá (nepovinná pokiaľ je neželané na replike spravovať certifikáty) je o certifikačnom systéme[2, kapitola 6].

Autori FreeIPA doporučujú minimálne dve no maximálne štyri replikačné dohody z dôvodu negatívnej redundancie, pretože jedna replika sa môže replikovať len z jedného serveru v danom čase. To je príčinou toho, že zvyšné replikačné dohody len čakajú. Zároveň môže mať viacero replikačných dohôd negatívny vplyv na celkový výkon. [2, kapitola 4.2.2]

Navrhované vzory replikačnej topológie sú znázornené v obrázkoch 3.2



Obrázok 3.2: Topológia pre tri segmenty vo zväzku

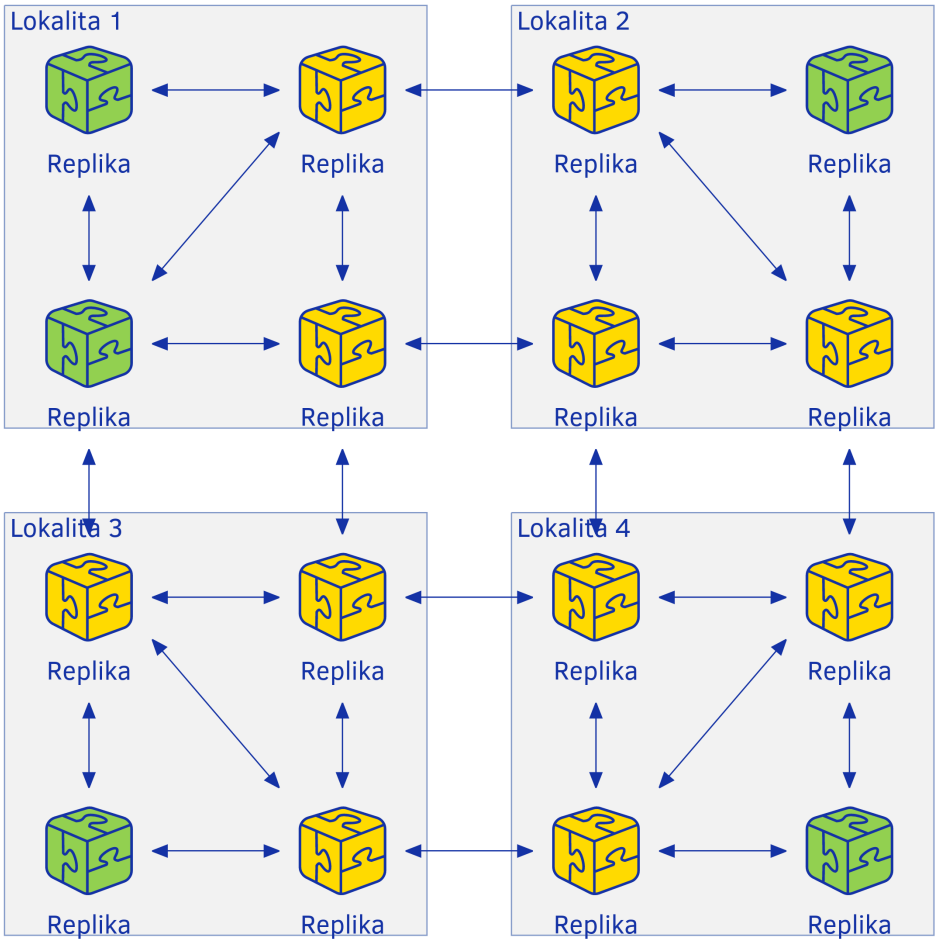
a 3.3. Farby, ktorými sú jednotlivé repliky označené vyjadrujú, koľko replikačných dohôd s ostatnými replikami majú. Zelená označuje dve, modrá tri a žltá štyri.

Samotná replikácia prebieha na nešifrovanom protokole LDAP a pokiaľ je žiadúce použiť SSL, je potreba explicitne zadať certifikát CA pri konfigurovaní replikácie.[2, kapitola 38.4.]

3.2.4 Škálovanie

Pod pojmom škálovanie si možno predstaviť vlastnosť systému, v tomto prípade FreeIPA, s jednoduchou rozšíriteľnosťou.

Existujú dva typy škálovania, a to horizontálne a vertikálne. Pod pojmom horizontálne škálovanie si možno predstaviť rozšírenie systému o jednotlivé



Obrázok 3.3: Topológia pre štyri segmenty vo zväzku

prvky, teda napríklad pridanie ďalšieho serveru. Vo vertikálnom škálovaní ide o vylepšenie vlastností prvku systému, teda napríklad zvýšenie výkonnosti serveru.

V tejto práci sa budem venovať horizontálnemu škálovaniu, a teda rozširovaniu FreeIPA o ďalšie repliky pre rozloženie záťaže.

Podrobne sa tejto téme venuje Bondi v „Characteristics of scalability and their impact on performance. Proceedings of the second international workshop on Software and performance“ [25].

3.2.5 Vysoká dostupnosť

Dostupnosť je vlastnosť systému, ktorou je vyjadrená schopnosť služby (alebo systému) obsluhovať požiadavky v dlhšom časovom úseku (týždne, mesiace,

roky).

Vysoká dostupnosť je pojem, pod ktorým si možno predstaviť dostupnosť systému s nebadateľnými výpadkami, napríklad niekoľko minút v rozsahu dní, týždňov až mesiacov.

Pri systéme autentikácie a autorizácie, čím FreeIPA je, je obzvlášť potrebné vysokú dostupnosť dosiahnuť, pretože pri nedbalom dizajne a nepredvídaní rôznych udalostí môže nedostupnosť systému úplne narušiť prístup k serverom a službám, ktoré tento systém využívajú.

Viac sa téme vysokej dostupnosti venuje Piedad v knihe „High Availability: Design, Techniques, and Processes“ [26].

Kapitola 4

Návrh nasadenia

V tejto kapitole preberiem návrh deploymentu nástroja FreeIPA ako IdM (Identity Management). Zameriam sa na dva rôzne prístupy nasadenia a to single deployment a škálovaný deployment.

Návrh počíta so servermi, na ktorých je nasadený operačný systém CentOS 7 64bit, teda hlavný server a všetky repliky neskôr pripojené bežia na tomto operačnom systéme. FreeIPA je nasadená vo verzii 4.5, ktorá je dostupná v repozitároch CentOS.

Servery potrebujú na svoj správny beh pri minimálnej konfigurácií minimálne 1500 MB operačnej pamäte, no odporúčam aspoň 2 GB pre plynulý beh. Minimálna veľkosť disku pre servery je 5 GB pretože minimálna inštalácia CentOS 7 spolu so serverovou časťou FreeIPA zaberie 2 GB.

Optimálné hardware požiadavky pre približne 10 000 uživateľov sú 4 jadrá, 4 GB RAM a 30 GB úložiska.

Zo softwérového hľadiska je potrebné, aby na serveri neboli bežiace služby ako sú DNS, Kerberos alebo adresárový server. Zároveň je optimálne, aby bola služba `nscd` (name service cache daemon) vypnutá, prípadne nebola využívaná pre zdroje využívané `SSSD`, pretože obe tieto služby využívajú cache. V neposlednom rade musí systém podporovať IPv6 a musí byť na ňom povolený. [2, kapitola 38.1.3.3.]

4.1 Single deployment

Single deployment, alebo samostatné nasadenie predpokladá jeden výkonný server, ktorý bude spracovávať všetky požiadavky od všetkých klientov.

Takýto prístup sa môže vyplatiť v menšom tíme (jednotky až niekoľko desiatok užívateľov), no v prípade väčších tímov (desiatky až stovky užívateľov) je toto nasadenie nedostatočné, pretože spolieha na jeden server, ktorý nemusí byť vždy a všade dostupný. Táto nedostupnosť sa môže prejaviť napríklad pri jednoduchom DNS požiadavku, kde sú všetky služby spustené na jednom ser-

veri a musia sa deliť o procesorový čas a operačnú pamäť, teda službe DNS zaberie viacej času odpovedať na tento požiadavok. Pokiaľ je to aj len pár milisekúnd na jeden požiadavok, pri väčšom množstve požiadavkov sa dokáže čas nakumulovať aj na desiatky sekúnd, najmä pri požiadavku na viacero doménových mien v priebehu krátkeho času. Pri dostatočne výkonnom serveri sa tento nedostatok nemusí prejaviť, no stále sa neodstráni problém jediného bodu zlyhania.

Pri návrhu nasadenia sa nesmie zabudnúť na zálohovanie a prípadnú obnovu dát. Z dôvodu, že samotná FreeIPA plné zálohovanie nad bežiacím serverom nepodporuje [2, kapitola 9.1], single deplyoment nie je optimálny a zameriam sa aspoň na minimálny škálovaný deployment, ktorý počíta s replikáciou, a teda zálohou dát a konfigurácie služby a zároveň minimálny downtime.

4.2 Škálovaný deployment

V škálovanom deployment-e počítam s vytvorením infraštruktúry, ktorá pripraví servery na nainštalovanie FreeIPA. Pri vytvorení tohoto nasadenia by sa malo počítať s redundanciou (aspoň minimálnou) a teda vyhradiť si minimálne jeden server, ktorý bude fungovať ako záloha a v prípade výpadku hlavného serveru preberie jeho rolu.

FreeIPA ma v tomto prípade výhodnú vlastnosť, a tou je replikácia, ktorou som sa zaoberal v predošlej kapitole (3.2.3). Vďaka replikácii môžem zabezpečiť konzistentnosť dát a neustále mať k dispozícii aktuálnu zálohu, v prípade, že by nastal výpadok jedného alebo viacerých, no nie úplne všetkých serverov.

Z hľadiska topológie je dobré použiť maximálne 4 replikačné dohody na jeden server. [2, kapitola 4.2.2] To znamená, že nie je vhodné, aby mal každý server replikačnú dohodu s každým ďalším serverom, preto je potrebné navrhnúť základnú topológiu, ku ktorej je možné následne pridávať ďalšie servery. Samotná dokumentácia k FreeIPA navrhuje nasadenie v zväzkoch serverov, kde majú servery v zväzku mnoho replikačných dohôd medzi sebou, aby sa zaistila integrita jedného zväzku a následné nie tak silno späté replikačné dohody medzi zväzkami samotnými. Zväzky je vhodné umiestniť na základe lokácii, v ktorých sa nachádzajú, teda napríklad jeden zväzok v datacentre, ďalší v kanceláriach a záložný v úplne inej lokalite.

Obnovu dát v rôznych scenároch popíšem v ďalšej sekcii *Záloha a obnovenie* (4.4).

Deployment samotný pozostáva z minimálne jedného hlavného serveru (master) a jednej úplnej repliky (dáta, DNS, správa certifikátov). Podľa potreby je možné pridávať ďalšie repliky, či už so schopnosťou spravovať certifikáty alebo nie.

Optimálny je ďalší server, ktorý bude fungovať ako úložisko užívateľských dát (služba automount).

4.2.1 Vysoká dostupnosť

Pri riešení vysokej dostupnosti bude využitý DNS server a jeho záznamy SRV a round-robin protokol, dostupný v BIND9.[27, kapitola 6.2]

Požívam SRV záznam, pretože v zázname môže byť uvedené doménové meno a využiť túto skutočnost možno tak, že v DNS serveri bude definovaných viacero A záznamov na jedno doménové meno (master). To zabezpečí rotáciu jednotlivých požiadavkov medzi jednotlivé FreeIPA servery a ich rovnomernú záťaž pre všetky služby (LDAP, Kerberos, ntp).

Keď sa server dostane do offline stavu musí byť zo zoznamu A/AAAA záznamov vyradený, pretože by to viedlo k pravidelným výpadkom kvôli rotácii požiadavkov, teda aj na uvedený offline server.

Na toto by bolo vhodné využiť monitorovaciu službu, ktorá pravidelne kontroluje schopnosť serveru odpovedať a v prípade nedostupnosti vykoná predom určenú akciu. V tomto prípade to bude odobranie SRV záznamu pre príslušný server z DNS serveru. Tým sa zaistí plynulé pokračovanie obsluhy požiadavkov.

Pre vysokú dostupnosť DNS serveru budú využité 3 DNS servery. V každom klientovi budú definované 2 záznamy DNS serveru. V prípade, že sa jeden z týchto dvoch serverov stane nedostupný, monitorovacia služba fyzicky nastaví tretiemu serveru IP adresu tohto nedostupného serveru.

4.3 Monitoring

Pre monitoring je možno využiť službu na to určenú (napríklad keepalived, zabix, …). Táto služba vykoná akciu pri predom definovanom incidente. Napríklad odoberie záznamy z DNS serveru pri nedostupnosti serveru, reštartuje službu Apache2 pri špecifickej udalosti, premaže zložku `/tmp` pri dosiahnutí určeného zaplnenia disku atď. Toto je možno využiť pri vysokej dostupnosti a škálovaní, kde je potrebné udržať minimálne jeden server k dispozícii.

V implementačnej časti žiadnu konkrétnu monitorovaciu službu neimplementujem, len podporné skripty. Je to hlavne z dôvodu, že zvyčajne má každý provozovateľ vybranú vlastnú monitorovaciu službu podľa svojich potrieb a znalostí.

4.4 Záloha a obnovenie

Samotní tvorcovia FreeIPA ponúkajú dva rôzne prístupy k zálohám, a to pomocou replík alebo príkazov poskytovaných v balíčku *ipa-server*, a to *ipa-bakcup* a *ipa-restore*. Oba tieto spôsoby slúžia na iný typ zálohy, ktorý sa používa pri inom usecase.

Príkazy poskytované v balíčku *ipa-server* umožňujú vytvoriť zálohu dát alebo aj úplnú zálohu FreeIPA, no celá táto záloha je viazaná ku konkrétnemu

stroju a je doporučované nepoužívať túto zálohu na inom serveri ako na serveri, kde bola vyhotovená. Dôsledkom tejto skutočnosti je to, že tento príkaz je možno využiť pri aktualizácií serveru na novšiu verziu FreeIPA alebo hardware chybe, kde nie sú dostupné iné servery, z ktorých by sa mohol server FreeIPA replikovať. Pokiaľ je vytváraná úplná záloha, služba FreeIPA nemôže byť na serveri spustená. Príkazy vyššie uvedené sa postarajú o to, že FreeIPA nie je spustená. [2, kapitola 9.1.]

Tento prístup je vhodný pri single deployment-e, kde server beží priamo na fyzickom stroji, v prípade virtualizovaného serveru je lepšie použiť možnosť *snapshot*, a teda získať stav serveru predtým, ako prebehne napríklad aktualizácia softwéru. Tento snapshot zachytí celý disk, teda tu nie je potrebné mať obavy z chýbajúceho jedného konfiguračného súboru, ktorý môže zapríčiniť vyčerpávajúce hľadanie chyby a dôvodu, prečo sa server nespráva podľa očakávaní.

Výhoda zálohy a obnovenia z replík spočíva v tom, že dáta obsiahnuté v nej nie sú viazané na žiaden konkrétny server. To znamená, že všetky servery, ktoré je potrebné obnoviť, môžu byť replikované zo serverov vo zväzku, no zároveň táto varianta neposkytuje zálohu a obnovenie konfigurácie serveru samotného. Avšak toto nie je potrebné pri nasadzovaní celého serveru nanovo, keďže dáta, ktoré sú zaujímavé, sú dáta v adresárovom serveri.

4.4.1 Dáta

Na zálohu a obnovenie dát je vhodné použiť servery vo zväzku, do ktorého je tento server nasadený. Teda vytvorí sa replika s replikačnou dohodou určenou podľa dopredu určenej topológie a po inštalácii sa vytvoria replikačné dohody so zvyškom serverov definovaných v topológii. O zálohu dát sa potom starajú replikačné dohody a automatická replikácia medzi servermi určenými pre daný server.

4.4.2 Konfigurácia

Záloha a obnovenie konfigurácie je používané najmä pri nevydarenej aktualizácii FreeIPA alebo neobnoviteľnom zlyhaní hardwareu, kedy je potrebné stroj vypnúť a hardware vymeniť. Pri obnovení zálohy odporúčam spustiť *ipa-restore* s plnou zálohou a následne spustiť vynútenú replikáciu z master serveru alebo zvyšných serverov nachádzajúcich sa vo zväzku.

Pri automatizovanom nasadení FreeIPA, ktoré spracovávam v sekcii *Nasadenie* v kapitole 5 (5.1) sa zálohou konfigurácie a jej obnovením netreba zaoberať, pretože skript pre nasadenie FreeIPA repliky sa postará o to, že prostredie v ktorom je replika inštalovaná je homogénne a pri aktualizácii je vytvorená úplná záloha serveru v prípade neúspechu.

4.5 Autentikácia pomocou Open ID Connect

Ako som v predošlej kapitole opísal, Open ID Connect je protokol fungujúci hlavne pre webové aplikácie. Pre prepojenie FreeIPA s OIDC poskytovateľom existuje služba Keycloak, kde je týmto poskytovateľom samotný Keycloak. Toto prepojenie funguje na báze federácie databáz, teda identity uložené interne vo FreeIPA sú dostupné pre Keycloak. [28, kapitola 3.15]

Tento prístup je popísaný nižšie v podkapitole 4.5.1.

Samotná FreeIPA možnosť externej autentikácie úplne nepodporuje, no v posledných verziách FreeIPA (4.4.0) sa začalo experimentovať s externou autentikáciou pomocou certifikátov využívajúcou moduly autentikácie pre službu `Apache2`. Pre verzie 4.4.0 a nižšie je táto autentikácia dostupná ako experimentálny plugin pre FreeIPA [29] a od verzie 4.5.0 je táto funkcia priamo podporovaná bez nutnosti inštalovania pluginu. Táto autentikácia funguje pre certifikáty nahrané na smartkartách. [2, kapitola 23]

Využíva moduly `mod_lookup_identity` a `mod_auth_gssapi`, kde modul `mod_lookup_identity` vyhľadá certifikát podľa užívateľa a páruje ho s identitou z FreeIPA. Ďalej `mod_auth_gssapi` vyžiada kerberos tiket v mene autentikovaného užívateľa pomocou rozšírenia s4u2self. Autentikačné dáta sú potom uložené do ccache. [30]

4.5.1 Keycloak

Prvou možnosťou ako spojiť protokoly OIDC a Kerberos bolo využitie služby Keycloak, a to tak, že by sa pripravila federácia databáz Keycloak a FreeIPA pomocou SSSD. Užívateľ by sa autentikoval u OIDC poskytovateľa pomocou Keycloak a následne by prebehla lokálna autentikácia na strane Keycloak, ktorá by skontrolovala, či ide o validného užívateľa. [28, kapitola 3.13]

Tento prístup funguje pre služby, kde je potrebná identita FreeIPA pre prihlásenie pomocou OIDC do externej aplikácie, no úplne nespĺňa cieľ mojej práce, a tým je prihlásenie sa priamo do FreeIPA, kde môže užívateľ spravovať svoju identitu (resetovať heslo, upravovať atribúty, pridať verejný SSH kľúč, atď.).

4.5.2 Moduly Apache2

Ďalšia možnosť bola upraviť službu `Apache2` priamo na serveri FreeIPA. Ako som vyššie spomenul, FreeIPA podporuje externú autentikáciu cez smartkarty pomocou dvoch modulov. Tieto moduly by sa dali využiť pri autentikácii cez Open ID Connect poskytovateľa, kde by bol do služby `Apache2` doinštalovaný modul `mod_auth_openidc`, ktorý slúži na odosielanie požiadavkov podľa špecifikácie OIDC [31] .

Autentikácia by prebehla na strane OIDC poskytovateľa, odkiaľ by bol ďalej autentikovaný užívateľ presmerovaný na lokáciu v `Apache2`, na ktorej by

prebehlo spárovanie užívateľa s identitou z FreeIPA pomocou `mod_lookup_identity` a následné získanie tiketu vďaka `mod_auth_gssapi`.

Tento prístup bohužiaľ nefunguje, pretože modul `mod_lookup_identity` bol vytvorený pre vyhľadanie užívateľa pomocou certifikátu [32], a teda ID tokeny a Access tokeny, ktoré odošle OIDC poskytovateľ sú v tomto prípade nevyužité.

4.5.3 Navrhované riešenie

Pre účel prihlásenia priamo do webového rozhrania FreeIPA by bolo vhodné vytvoriť nový modul pre službu `Apache2`, ktorý by fungoval ako klient v kontexte OIDC a na podobnom princípe ako modul `mod_lookup_identity` v kontexte FreeIPA, teda by vyhľadával užívateľov pomocou služby SSSD.

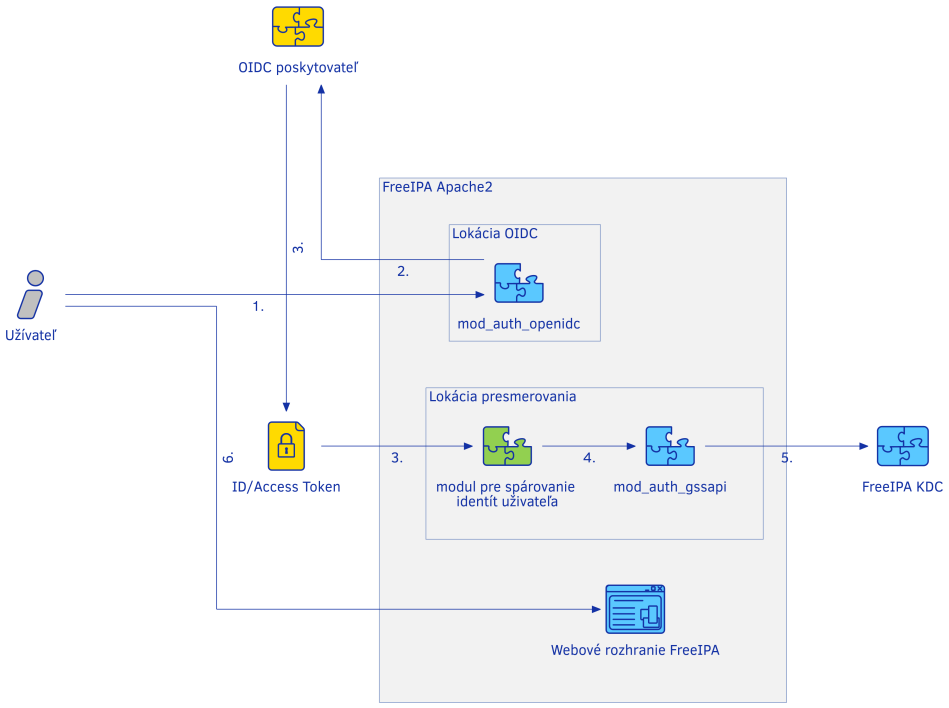
Teda modul `mod_auth_openidc` presmeruje užívateľa k OIDC poskytovateľovi, kde prebehne autentikácia a presmerovanie späť do lokácie obsluhovanej `Apache2` web serverom. V tejto lokácii bude využitý vyššie spomenutý modul, ktorý bude fungovať ako klient OIDC buď na princípe OIDC Authorization Code Flow alebo Implicit Flow. Z hľadiska bezpečnosti je vhodnejšie použiť Authorization Code Flow, pretože ten nevystavuje ID ani Access tokeny priamo užívateľovi (alebo agentovi pomocou ktorého sa užívateľ k nemu pripája), no bola by potreba implementovať komunikáciu medzi týmto modulom a OIDC poskytovateľom.

Po úspešnom získaní ID tokenu by z neho tento modul získal priamo identitu užívateľa (login), pomocou ktorej by mohol modul `mod_auth_gssapi` získať pre tohoto užívateľa ticket pre HTTP službu a užívateľ sa mohol prihlásiť do webového rozhrania FreeIPA pomocou tohoto ticketu.

4.5.3.1 Podrobnejší postup prihlásenia sa pomocou OIDC

Pre znázornenie je použitý Implicit Flow. Pri použití Authorization Code Flow by bolo medzi tretím a štvrtým krokom ešte vytvorené spojenie medzi modulom pre spárovanie identít užívateľa a OIDC poskytovateľom pre výmenu ID/Access tokenu.

1. Užívateľ sa chce prihlásiť,
2. `mod_auth_openidc` presmeruje užívateľa k OIDC poskytovateľovi,
3. OIDC poskytovateľ presmeruje užívateľa spolu s ID/Access tokenom do lokácie, kde sa nachádza modul pre spárovanie identít užívateľa,
4. modul pre spárovanie identít užívateľa využije SSSD pre vyhľadanie užívateľa,
5. `mod_auth_gssapi` vyžiada pomocou s4u2self operácie ticket pre užívateľa k službe HTTP,



Obrázok 4.1: Autentikácia do WebUI FreeIPA pomocou OIDC

6. užívateľ sa môže prihlásiť pomocou ticketu, ktorý získala služba HTTP.

Tento postup zobrazuje obrázok 4.1.

Kapitola 5

Konfiguračné skripty

Celé nasadenie som sa rozhodol implementovať pomocou konfiguračného nástroja Ansible, vďaka ktorému je možno jednoducho a jednotne definovať ako bude FreeIPA nasadená, aké možnosti bude mať (napr. automount, vlastnú DNS) a hlavne, bude vyžadovať minimálne nastavenie od užívateľa (v tomto prípade administrátora).

Pokiaľ budem hovoriť o ceste k súboru, koreňový adresár bude vždy „src/ansible“ a teda všetky cesty budú relatívne k tomuto adresáru.

5.1 Nasadenie

Nasadenie FreeIPA som sa rozhodol riešiť pomocou *Ansible Roles*, ktoré som spomenul v sekcii 3.2.1, pretože každý zo serverov v infraštruktúre bude plniť nejakú rolu. Vďaka využitiu rolí môžem definovať napríklad prerekvizitu nainštalovaného klienta na serveri, kde je želané mať centrálne úložisko.

Vytvoril som role pre hlavný server, v tomto kontexte je to server, ktorý je nasadený ako prvý, repliky, čo sú servery, nasadené pomocou FreeIPA replikačnej technológie a v neposlednom rade klienta FreeIPA, kde táto rola nainštaluje balíčky potrebné balíčky a prebehne registrácia do FreeIPA.

Hlavným súborom, kde je potrebná konfigurácia pre nasadenie je *inventory*, ktorý som popísal v 3.1.1, existujú dve verzie *inventory* a to `hosts.yml` a `hosts.ini`. V nich sú definované všetky servery v topológii, premenné viažúce sa k týmto serverom ako napríklad `hostname`, IP adresa pre priame pripojenie pomocou `ssh` a iné. Podrobnejšie premenné vo vyššie spomenutých súboroch opíšem v 5.1.1.

V hlavnom adresári sa nachádza množstvo spustiteľných skriptov s názvom odpovedajúcemu formátu (`un`)`install_ipa_*.yml`, ktoré nainštalujú, prípadne odinštalujú danú rolu.

Tabuľka 5.1: Štruktúra priečinkov pre role Ansible

`ansible_user`	Užívateľ, pomocou ktorého sa Ansible pripojí k danému serveru.
`ipa_domain`	Názov domény, v ktorej FreeIPA operuje.
`ipa_master_name`	FQDN serveru, ktorý bol nasadený ako prvý.
`ipa_master_ip`	IP adresa serveru, ktorý bol nasadený ako prvý.
`ipa_external_dns`	Príznak, či je FreeIPA nasadzovaná s externým DNS serverom alebo nie.
`ipa_external_dns_key`	Názov kľúča TSIG pre vzdialené aktualizácie DNS serveru.
`ipa_dns_1`	Prvý DNS server, ktorý je definovaný pre klientov. Tiež je použitý ako spojenie s externým DNS serverom.
`ipa_dns_2`	Druhý DNS server, ktorý je definovaný pre klientov.
`ipa_admin`	Užívateľ, ktorý môže vykonávať administrátorskú činnosť nad FreeIPA.
`ipa_automount_home_path`	Cesta k priečinku, ktorý bude vyexportovaný ako adresár s domovskými adresármi užívateľov z NFS serveru.

5.1.1 Inventory

Premenná `ipa_hostname` nie je definovaná ako FQDN z jednoduchého dôvodu, a to, že pri nasadzovaní v inej doméne by bolo potrebné meniť každý záznam pri každom serveri, a preto je doména z `hostname` vynechaná a doplňovaná v každom *playbook-u*.

Parametre `ipa_dns_1` a `ipa_dns_2` sú IP adresy DNS serverov, ktoré sa nastavia ako DNS servery pre klientov.

V inventári sa nachádza aj premenná `ipa_external_dns`, ktorá značí, či sa má FreeIPA nasadiť spolu s integrovanou DNS službou, alebo sa na to využije iný server definovaný v `ipa_dns_1`.

Premenná `ipa_ip` značí IP adresu z rozsahu, ktorá bude spravovaná vo FreeIPA, môže byť iná ako adresa, pomocou ktorej sa Ansible na server pripája.

V nasadení používam aj *ansible-vault* spomenutý v 3.1.1. Zašifrovaný súbor je možno nájsť v ceste „vars/secrets.yml“, a nachádzajú sa tam všetky citlivé informácie potrebné pre nasadenie a operatívu FreeIPA. Počiatočné heslo, pomocou ktorého sú zašifrované, je „atenorth“.

Zoznam jednotlivých premenných platných pre všetky servery z *inventory* spolu s ich popisom možno preštudovať v 5.1.

5.1.2 Master Server

ansible role: **ipaserver**

Master server, alebo aj hlavný server sa v tomto ponímaní chápe ako server, ktorý bude nainštalovaný ako prvý. Zároveň sa vo všetkých ostatných skriptoch pri potrebe definovania spojenia so serverom FreeIPA bude využívať jeho doménové meno z dôvodu jednotnej konfigurácie na všetkých serveroch.

Playbook pre túto rolu nainštaluje potrebné balíčky definované v štandardných premenných v priečinku *defaults*. Následne spustí inštalačný príkaz na inštaláciu FreeIPA na server. Pokiaľ je definovaná externá DNS, inštalátor nenainštaluje integrovaný DNS server. Zároveň skript pripraví DNS SRV záznamy potrebné pre správne fungovanie replík a klientov napríklad pre Kerberos alebo LDAP. Zoznam týchto záznamov sa nachádza v ceste „roles/ipaserver/defaults/main.yml“ v premennej `ipaserver_dns_records`.

Po úspešnej inštalácií server vytvorí reverznú zónu pre IP adresu, ktorú má tento server definovanú ako IP adresu `ipa_ip` a vytvorí v nej záznam pre seba samotného.

5.1.3 Repliky

ansible role: **ipareplica**

Pred inštaláciou repliky je potrebné aby bol server, na ktorom sa inštaluje replika zaradený do FreeIPA, teda aby bol server klientom FreeIPA služby. Na toto využívam Ansible rolu klienta, ktorú popíšem v 5.1.4.

Každá replika má definovanú premennú `ipa_master`, ktorá určuje, s ktorým serverom sa má vytvoriť počiatočná replikačná dohoda.

Niektoré repliky majú definované servery, s ktorými ďalšími servermi sa má pri nasadení vytvorit replikačná dohoda, no nie všetky. To je z dôvodu, že pri nasadení repliky, nemusia existovať servery, ktoré má v predom určenej topológii nastavené, a teda nemôže medzi týmito servermi vytvoriť replikačnú dohodu. To nie je problém, pretože replikačné dohody vytvorené s týmto serverom neskôr sú obojstranné. Teda je replikačná dohoda vytvorená aj so serverom, ktorý nemusí mať definované žiadne servery, s ktorými má vytvoriť replikačnú dohodu, ale je definovaný ako server inej repliky, ktorá s ním má replikačnú dohodu mať.

Pri inštalácii repliky sa automaticky vytvoria záznamy v DNS serveri. Ako v integrovanom DNS serveri, tak aj v externom DNS serveri. Porty a váha jednotlivých SRV záznamov sú definované v „roles/ipareplica/defaults/main.yml“ a v premennej `ipareplica_dns_records`.

5.1.4 Klienti FreeIPA

ansible role: **ipaclient**

Inštalácia klienta nastaví serveru DNS servery definované v *inventory*, pripraví príslušné porty, nainštaluje klientský program pre FreeIPA a vytvorí záznamy v či už integrovanom alebo externom DNS serveri.

5.1.5 Domovské adresáre

ansible role: **ipaautomount**

Táto rola má na starosti nastavenie automatického pripájania vzdialeného úložiska, o ktorého nastavenie sa postará rola opísaná v 5.1.6. Rola nastaví automatické pripájanie želanej lokácie definovanej v automount vo FreeIPA a nastaví SSSD, aby automaticky obnovovala tikety z NFS serveru, na ktorý sa pripája.

5.1.6 Úložisko

ansible role: **ipastorage**

Táto rola slúži k nastaveniu FreeIPA úložisku určenému k ukladaniu užívateľských dát naprieč servermi.

Využíva k tomu technológiu *automount* a *NFSv4*, kde prebieha autentikácia a autorizácia pomocou Kerberos.

Rola vytvorí vytvorí službu NFS v FreeIPA Kerberos a zaistí, že pre ňu existuje automount lokácia a mapa.

5.2 Operatíva

5.2.1 Pridanie užívateľa

názov skriptu: `setup_users.yml`

Pre pridanie viacerých užívateľov a udržaní si prehľadu o tom, aký užívatelia existujú vo FreeIPA som vytvoril ansbile skript, ktorý pridá užívateľov uvedených v *csv* súbore. Týmto užívateľom je možné priradiť atribúty ako sú meno, priezvisko, štandardné heslo pre prvé prihlásenie, verejný ssh kľúč a login shell, ktorý je spustený pri prihlásení.

Súbor `users.csv`, ktorý je možno nájsť v priloženom CD v ceste „src/ansible/operative/lists/users.csv“ je vo formáte *csv* kde ako oddeľovač slúži znak „;“ a má nasledovnú štruktúru:

Login* unikátne meno, ktoré užívateľ používa pre prihlásenie

Stav* značí, či má užívateľ existovať alebo nie, možné voľby sú: „present“ alebo „absent“

Meno* krstné meno užívateľa

Priezvisko* priezvisko užívateľa

UID unikátne číslo užívateľa, pokiaľ je vynechané, vygeneruje sa náhodne

GID unikátne číslo skupiny, do ktorej bude užívateľ patriť, ak je vynechané vytorí sa skupina pre užívateľa

Default shell štandardný shell, ktorý sa spustí pri prihlásení užívateľa

SSH kľúč verejný SSH kľúč, ktorý bude použitý pri prihlasovaní sa užívateľa

Heslo* počiatočné heslo, ktorým sa užívateľ prihlási, aby mohol heslo zmeniť

* Položky označené hviezdičkou sú povinné.

5.2.2 Vytvorenie skupín

názov skriptu: `setup_groups.yml`

Tento skript vytvorí viacero skupín naraz. Dáta čerpá zo súboru `groups.yml` z cesty „src/ansible/operative/lists/groups.yml“. Sú v ňom definované skupiny, ktoré majú existovať a užívatelia, ktorí do tejto skupiny patria. Zoznam užívateľov, ktorí patria do skupiny je nastavený v skupine, teda pokiaľ je nejaký užívateľ, ktorý predtým do skupiny patril zo zoznamu odobraný, je taktiež odobraný zo skupiny. Pokiaľ je zoznam užívateľov prázdny, aj skupina bude prázdna.

5.2.3 Zmena DNS SRV záznamu

názov skriptu: `update_dns_records.yml`

Skript pripravený na odobranie SRV záznamu z DNS serveru. Pokiaľ je v *inventory* definovaná premenná `ipa_external_dns` a je nastavená na hodnotu „yes“ upraví sa záznam v externom DNS. Ak táto premenná definovaná nie je, alebo je definovaná na „no“, záznam je upravený v integrovanom DNS serveri vo FreeIPA.

Skript je potrebné spustiť s ďalšími parametrami (aby sa zaistila automatizovateľnosť), a teda vo formáte `update_dns_records.yml -e "ipa_server=HOST"`, kde HOST je FQDN serveru, ktorého DNS záznamy majú byť odstránené.

Kapitola 6

Využitie implementácie

6.1 Administrátor

6.1.1 Nasadenie

Skripty na nasadenie vytvárajú DNS záznamy ako v integrovanom tak externom DNS serveri. Pri externom DNS serveri je potrebné, aby tento DNS server podporoval aktualizáciu DNS záznamou pomocou *nsupdate*.

Po úspešom dokončení nasadenia je k dispozícií webové rozhranie priamo na adrese serveru, na ktorom bola FreeIPA nasadená. Priamo na serveri FreeIPA je dostupné aj CLI priamo z shellu pomocou príkazu `ipa`.

6.1.1.1 Single deployment

Pri nasadení jedného serveru, popísanom v 4.1 sa využija Ansible rola *ipaserver* opísaná v 5.1.2.

K tomuto slúži spustiteľný skript `install_ipa_server.yml`.

6.1.1.2 Škálovaný deployment

V škálovanom deploymente sa využijú Ansible role *ipaserver* (5.1.2) a *ipareplica* (5.1.3).

Prvý server, ktorý bude nasadený je „master“, tento server sa takto nazýva len z dôvodu, že je nasadený ako prvý.

Pred nasadením ďalších replík je vhodné definovať topológiu, v ktorej budú tieto repliky nasadené. Pri definovaní topológie je potrebné myslieť na to, že optimálny počet replikačných dohôd na repliku sú dve až štyri, iný počet by bol neoptimálny. Táto topológia sa definuje v *inventory* (`hosts.yml` alebo `hosts.ini`) v skupine replík `ipareplicas` pri zázname o serveri v premenných `ipa_replica_master` a `ipa_replica_neighbors`.

Po úspešnom nasadení „master“ serveru môže byť pridaný ľubovoľný počet replík s replikačnými dohodami definovanými v *inventory* podľa predom určenej topológie.

Pre správne nasadenie repliky je potrebné, aby na serveri, na ktorom je replika nasadzovaná bola prítomná Ansible rola *ipaclient*, o to sa postará prerekvizita definovaná v samotnej role *ipareplica*, takže sa administrátor nemusí o túto skutočnosť starať.

Pre spustenie nasadenia replík slúži skript `install_ipa_replica.yml`.

6.1.2 Zdieľané úložisko

Pokiaľ je želaná funkcionalita zdieľaného úložiska pre užívateľov naprieč servermi, je potreba nasadiť Ansible rolu *ipastorage* (5.1.6), ktorá bude slúžiť ako NFS server a Ansible rolu *ipaautomount* (5.1.5).

6.1.2.1 Server

V tomto kontexte je server stroj, na ktorom je spustený NFS server, a teda môže poskytovať ostatným serverom svoje exportované zložky. Tento server je konkrétne definovaný v *inventory* v skupine `ipastorage` a k jeho nasadeniu slúži Ansible rola *ipastorage* (5.1.6). Administrátor nemusí riešiť žiadne prerekvizity a jediné čo potrebuje je spustiť skript `install_ipa_storage.yml`.

Pokiaľ je žiadané, je možno zmeniť lokáciu, do ktorej tento skript pripraví mapu pre automount. K tomu slúži premenná `ipastorage_location`.

6.1.2.2 Klient

Pre automatické pripojenie vzdialeného domovského adresára užívateľa slúži Ansible rola *ipaautomount* (5.1.5).

Klienti, ktorí budú túto službu využívať sú definovaní v *inventory* v skupine `ipaautomount_clients`.

Jediná prerekvizita pre funkčného klienta je funkčný NFS server a lokácia automount. O server sa postará vyššie spomenutá Ansible rola serveru *ipastorage* a lokácia automount je definovaná v *inventory* ako `ipaautomount_location`. Následne stačí spustiť skript `install_ipa_automount.yml`, ktorý sa postará o to, aby klienti definovaní v skupine mali prístup k vzdialenému úložisku.

6.1.3 LDAP login

FreeIPA ukladá všetky svoje dáta v adresárovom serveri, ktorý je dostupný aj mimo FreeIPA. Vďaka tomu je možné ho využiť napríklad pre webové služby, ktoré podporujú overenie identity pomocou LDAP serveru.

Pre správnu funkčnosť je potreba vyplniť správne atribúty. Presné hodnoty sa líšia od implementácie k implementácii webovej služby. Základné atribúty a ich hodnoty pre doménu `domain.local` sú uvedené v tabuľke 6.1,

kde sú uvedené hodnoty ako pre vyhľadávanie a overovanie užívateľa, tak aj pre skupiny.

Tabuľka 6.1: Atribúty pre nastavenie prihlásenia pomocou LDAP

Atribút	**Hodnota**	
	Užívateľ	**Skupina**
server	ldap://domain.local:389	
`base DN`	dc=domain,dc=local	
`search base`	cn=groups,cn=compat	cn=users,cn=accounts
`ID Atribút`	gid	uid
`objectClass`	posixGroup	posixAccount

6.1.4 Záloha a obnovenie

6.1.4.1 Single deployment

Ako som uviedol v kapitole 4.4, FreeIPA nepodporuje plnú zálohu (dát a konfigurácie) kým je spustená a je potrebné službu FreeIPA zastaviť. Preto pri vytvorení zálohy v single deployment je potrebné, aby administrátor službu FreeIPA zastavil, pomocou príkazu `ipa-backup` vytvoril zálohu, vykonal akcie, kvôli ktorým bola vytvorená záloha a v prípade neúspechu obnoví zálohu pomocou príkazu `ipa-restore BACKUP`, kde `BACKUP` je názov súboru, z ktorého obnovovať.

6.1.4.2 Škálovaný deployment

Pokiaľ pri škálovanom deploymente existuje ešte aspoň jeden ďalší server, na ktorom sú dáta v aktuálnom stave, nie je potrebné vytvárať zálohu dát.

Pokiaľ je želané vytvoriť zálohu konfigurácie (špecifické nastavenia pre server), je potreba zastaviť službu FreeIPA a pokračovať ako v scenári opísanom v 6.1.4.1.

Pri potrebe obnovenia dát stačí spustiť na serveri, na ktorom sa majú dáta obnoviť príkaz `ipa-replica-manage --from SEVER re-initialize`. Tento príkaz reinicializuje repliku, teda všetky dáta stiahne zo serveru `SERVER`.

6.1.5 Aktualizácia FreeIPA

6.1.5.1 Single deployment

Pri aktualizovaní FreeIPA v single deployment je potreba vytvoriť zálohu popísanú vyššie v sekcii 6.1.4.1. Následne stačí zavolať inštaláciu `freeipa-server`, napríklad pomocou `yum update freeipa-server -y`.

6.1.5.2 Škálovaný deployment

Pri škálovanom deploymente stačí nainštalovať `freeipa-server` alebo spustiť Ansible rolu pre inštaláciu repliky (pokiaľ sú dostupné balíčky v repozitároch). Samozrejme je potreba skontrolovať poznámky k vydaniu, či nenastala nejaká zásadná zmena. Pokiaľ áno, je vhodné postup premyslieť.

6.1.6 Loggovanie

Pri riešení incidentov a hľadaní chybnej konfigurácie je vhodné skontrolovať služby, ktoré FreeIPA používa.

Zoznam súborov, do ktorých služby FreeIPA zapisujú je nasledovný:
/var/log/audit/audit.log
/var/log/secure
/var/log/httpd/access_log
/var/log/httpd/error_log
/var/log/kadmind.log
/var/log/krb5kdc.log
/var/log/pki/pki-tomcat/ca/transactions
/var/log/dirsrv/slapd-<REALM>/access
/var/log/dirsrv/slapd-<REALM>/audit
/var/log/dirsrv/slapd-<REALM>/errors
/var/log/sssd/sssd.log
/var/log/sssd/krb5_child.log
/var/log/sssd/ldap_child.log
/var/log/sssd/selinux_child.log
/var/log/sssd/gpo_child.log
/var/log/sssd/sssd_nss.log
/var/log/sssd/sssd_pam.log
/var/log/sssd/sssd_pac.log
/var/log/sssd/sssd_autofs.log
/var/log/sssd/sssd_ssh.log
/var/log/sssd/sssd_sudo.log
/var/log/sssd/sssd_ifp.log
/var/log/sssd/sssd_<DOMAIN>.log

Kde <DOMAIN> je doména, ktorú má FreeIPA nastavenú a <REALM> je oblasť pre Kerberos, ktorý FreeIPA využíva.

6.1.7 Monitoring

Ako som opísal v sekcii 4.3, servery FreeIPA je možno monitorovať a v prípade incidentu vykonať nejakú akciu. Pri nedostupnosti serveru je možno využiť skript `update_dns_records.yml`, bližšie opísaný v 5.2.3, ktorý odstráni z DNS

serveru SRV záznamy, teda požiadavky nebudú presmerovávané na zadaný server.

6.1.8 Pridanie a odobranie entít

6.1.8.1 Klient

Pre pridanie a odobranie klienta FreeIPA slúžia skripty `install_ipa_client.yml`, resp. `uninstall_ipa_client.yml`. Prvý skript sa postará o to, aby bol server, na ktorom je spustený, zaradený do FreeIPA, a teda mohol využívať LDAP prihlasovanie pomocou SSSD, získavať tikety od Kerberos pre rôzne služby atď. Druhý slúži pre odobranie serveru z FreeIPA, teda nebude môcť využívať služby, ktoré FreeIPA poskytuje.

6.1.8.2 Užívateľ

Pre pridanie viacerých užívateľov naraz je vytvorený skript `setup_users.yml`, ktorý čerpá dáta z „operative/lists/users.csv“. Tento súbor je detailnejšie opísaný v 5.2.1. V tomto súbore je pri každom užívateľovi definovaný stav „present“ alebo „absent“, ktorý značí, či má daný užívateľ existovať.

6.1.8.3 Skupiny

Pridanie skupín taktiež prebieha hromadne pomocou skriptu `setup_groups.yml`, ktorý čerpá zo súboru „operative/lists/groups.yml“. Tento skript je podrobnejšie opísaný v 5.2.2. Taktiež je pri každej skupine definovaný stav „present“ alebo „absent“, ktorý definuje, či má skupina existovať alebo nie.

6.2 Užívateľ

6.2.1 Prvé prihlásenie

Po zaregistrovaní užívateľa do FreeIPA je vhodné, aby sa tento užívateľ prihlásil heslom, ktoré mu priradil administrátor pri registrácií a následne toto heslo zmenil.

K tomu ho FreeIPA vyzve hneď, ako toto heslo použije (ako pri prihlásení do webového rozhrania, tak pri prihlásení pomocou SSH na server pripojený do FreeIPA), pretože sa tomuto heslu nastaví maximálna platnosť a vyprší hneď ako je nastavené. [33]

6.2.2 LDAP login

Pokiaľ sa chce užívateľ prihlásiť do webovej služby, ktorá je pripojená na FreeIPA LDAP adresárový server, tak využije heslo, ktoré využíva na prihlásenie sa do FreeIPA.

6.2.3 SSH login

Pri prihlásení na server pripojený do FreeIPA sa užívateľ prihlási pomocou hesla, ktoré si definoval, alebo automaticky vďaka verejnému kľúču SSH, ktorý nahral pomocou webového rozhrania FreeIPA alebo ho tam nahral administrátor.

6.2.4 Automount

Pokiaľ sú pri nasadení využité ansible role *ipastorage* a *ipaautomount* z kapitoly 5.1, užívateľovi sa pri prihlásení na Linux server vždy pripojí jeho domovský priečinok a môže ho využívať naprieč servermi, na ktorých je prihlásený, ako zdieľané úložisko.

Keďže je pripájanie vzdialeného súborového systému zabezpečené pomocou Kerberos, je potrebné aby sa užívateľ na prihlásenom stroji autentikoval pomocou príkazu `kinit`.

Kapitola 7

Návrhy na ďalší vývoj

7.1 Open ID Connect modul

Ako som spomenul v sekcii 4.5.3, pre prihlásenie sa pomocou Open ID Connect do webového rozhrania FreeIPA je potrebný dodatočný modul pre službu `Apache2`.

Vývoj autentikačného modulu je nad rámec tejto práce, preto sa jeho implementáciou nezaoberám. Potreba pre tento modul však existuje, a preto nie je zbytočné sa vývojom takéhoto modulu zaoberať, pretože by spájal dva autentikačné protokoly, ktoré sa na prvý pohľad od seba zdajú veľmi vzdialené.

7.2 Monitorovanie

Pre monitorovanie FreeIPA by bolo vhodné vyvinúť Ansible rolu, ktorá by zabezpečila vznik monitorovacieho serveru a rolu, ktorá by doinštalovala do monitorovaných klientov potrebné doplnky.

7.3 Centrálne loggovanie

Pri raste systému sa stáva manažovanie jednotlivých serverov veľmi obtiažne a odhaľovanie problémov náročné. Z tohoto dôvodu by bolo vhodné pripraviť Ansible rolu, ktorá by pripravila server, ktorý zhromažďuje a spracováva logy a rolu, ktorá by pripravila klientov, aby tieto logy odosielala, napríklad pomocou `rsyslog`.

Touto témou sa začali zaoberať aj tvorcovia FreeIPA, zatiaľ iba v PoC [34].

7.4 Ansible modul

Ansible v súčasnej dobe podporuje 13 modulov zoberajúcich sa FreeIPA [6], no FreeIPA poskytuje viac ako 400 príkazov [35]. Z tohoto dôvodu by bolo

vhodné vyvinúť modul, ktorý by vykonával príkazy FreeIPA pomocou FreeIPA API. Zároveň by bolo vhodné konvertovať Ansible role z implementačnej časti do samostnatných modulov, prípadne implementovať moduly, ktoré by s inštaláciou FreeIPA pomohli.

Tvorcovia FreeIPA začali pracovať na Ansible rolách pre FreeIPA, no ich vývoj je nepravidelný a s častými dlhými odmlkami. [36]

Záver

Cieľom tejto práce bolo vysvetliť základy fungovania technológii, na ktorých je FreeIPA postavená a s ktorými pracuje. Ďalej navrhnúť nasadenie systému FreeIPA, vyriešiť otázky škálovateľnosti, vysokej dostupnosti a zálohovania. Tento cieľ sa mi podarilo splniť a v tejto práci ponúkam riešenie nasadenia FreeIPA, jej škálovateľnosť a vysokú dostupnosť. Zálohovanie nie je potrebné, pretože FreeIPA poskytuje službu replikácie, ktorú som podrobnejšie popísal v časti o architektúre FreeIPA. Ďalším cieľom bolo vytvoriť sadu automatizačných skriptov, vďaka ktorým bude možné rýchlo a jednoducho vytvoriť bezpečnú infraštruktúru, v centre ktorej bude stáť nástroj na správu identít - v našom prípade FreeIPA.

Okrem toho som sa v prácii zaoberal zlúčením poskytovania identít od externého poskytovateľa OpenID Connect a jej spárovaním spolu s identitami, ktoré má pod správou samotná FreeIPA pre pohodlné prihlásenie, napríklad aj pomocou Google.

Cieľ vytvorenia automatizačných skriptov sa mi podarilo splniť a súbor obsahuje skripty pre inštaláciu, resp. odinštaláciu prvého serveru FreeIPA, ďalších replík, klienta FreeIPA, NFS serveru, NFS klienta a operatívnych skriptov ako pridanie, resp. odobranie užívateľa.

Prepojenie FreeIPA s externým poskytovateľom identít som analyzoval v sekcii 4.5 a poskytol som návrh pre jeho implementáciu v sekcii 4.5.3.

V rámci práce som taktiež opísal možnosti využitia implementácie.

Do budúcna nebude problém pridávať ďalšie skripty rozširujúce funkčnosť či používať už existujúce aj pri zmene verzie FreeIPA vďaka parametrizovaným volaniam, ktoré pri zmene API možno jednoducho upraviť.

Literatúra

[1] Red Hat Inc.: *FreeIPA*. 2018, [cit. 2018-04-30]. Dostupné z: https://www.freeipa.org/page/Main_Page

[2] Hanzelka, F.; Maňásková, L.; Šteflová Petrová, A.; aj.: *Linux Domain Identity, Authentication, and Policy Guide*. 7 vydání, 2018, [cit. 2018-05-02]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/linux_domain_identity_authentication_and_policy_guide/index

[3] Red Hat Inc.: *PKI*. 2018, [cit. 2018-05-07]. Dostupné z: https://www.freeipa.org/page/PKI

[4] Red Hat Inc.: *Certmonger*. 2018, [cit. 2018-05-07]. Dostupné z: https://www.freeipa.org/page/Certmonger

[5] ansible: *ansible*. 2018, [cit. 2018-04-23]. Dostupné z: https://github.com/ansible/ansible

[6] Red Hat, Inc.: *All Modules*. Dostupné z: http://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html

[7] Red Hat, Inc.: *Ansible Vault)*. 2018, [cit. 2018-05-07]. Dostupné z: http://docs.ansible.com/ansible/latest/user_guide/vault.html

[8] Ronacher, A.: *Jinja2 Documentation*. 2008, [cit. 2018-05-01]. Dostupné z: http://mitsuhiko.pocoo.org/jinja2docs/Jinja2.pdf

[9] Red Hat, Inc.: *Templating(Jinja2)*. 2018, [cit. 2018-04-29]. Dostupné z: http://docs.ansible.com/ansible/latest/user_guide/playbooks_templating.html

[10] Hochstein, L.; Moser, R.: *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*. "O'Reilly Media, Inc.", 2017.

[11] Red Hat, Inc.: *Ansible Documentation*. 2018, [cit. 2018-04-27]. Dostupné z: http://docs.ansible.com/ansible/latest/

[12] The Apache Software Foundation: *Apache HTTP Server Version 2.4 Documentation*. 2018, [cit. 2018-05-08]. Dostupné z: https://httpd.apache.org/docs/2.4/

[13] Sun Microsystems, Inc.: *System Administration Guide, Volume 3*. 2000, [cit. 2018-05-07]. Dostupné z: https://docs.oracle.com/cd/E19455-01/806-0916/6ja8539g7/index.html

[14] Steiner, J. G.; Neuman, B. C.; Schiller, J. I.: Kerberos: An Authentication Service for Open Network Systems. In *Usenix Winter*, 1988.

[15] Neuman, C.; Yu, T.; Hartman, S.; aj.: The Kerberos Network Authentication Service (V5). RFC 4120, RFC Editor, July 2005, [cit. 2018-04-21]. Dostupné z: https://tools.ietf.org/html/rfc4120

[16] Garman, J.: *Kerberos: The Definitive Guide: The Definitive Guide*. " O'Reilly Media, Inc.", 2003.

[17] Red Hat Inc.: *Directory Server*. 2018, [cit. 2018-05-09]. Dostupné z: https://www.freeipa.org/page/Directory_Server

[18] Zeilenga, K.: Lightweight directory access protocol (ldap): Technical specification road map. RFC 4512, RFC Editor, 2006, [cit. 2018-04-28]. Dostupné z: https://tools.ietf.org/html/rfc4512

[19] Sermersheim, J.; Novell, Inc.: Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511, RFC Editor, 2006, [cit. 2018-04-30]. Dostupné z: https://tools.ietf.org/html/rfc4511

[20] Gulbrandsen, A.; Technologies, T.; Vixie, P.; aj.: A DNS RR for specifying the location of services (DNS SRV). RFC 2782, RFC Editor, February 2000, [cit. 2018-05-01]. Dostupné z: https://tools.ietf.org/html/rfc2782

[21] Red Hat Inc.: *DNS - FreeIPA*. 2018, [cit. 2018-04-30]. Dostupné z: https://www.freeipa.org/page/DNS

[22] Sakimura, N.; Bradley, J.; Jones, M. B.; aj.: OpenID Connect Core 1.0 incorporating errata set 1. Technická zpráva, The OpenID Foundation, 2014, [cit. 2018-05-06]. Dostupné z: https://openid.net/specs/openid-connect-core-1_0.html

[23] Hardt, D.: The OAuth 2.0 Authorization Framework. RFC 6749, RFC Editor, October 2012, [cit. 2018-05-05]. Dostupné z: http://www.rfc-editor.org/rfc/rfc6749.txt

[24] ansible: *ansible*. 2018, [cit. 2018-04-26]. Dostupné z: https://docs.ansible.com/ansible/devel/user_guide/playbooks_reuse_roles.html

[25] Bondi, A. B.: Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd international workshop on Software and performance*, ACM, 2000, s. 195–203.

[26] Piedad, F.; Hawkins, M.: *High availability: design, techniques, and processes*. Prentice Hall Professional, 2001.

[27] Internet Systems Consortium: *BIND 9.11 Administrators' Reference Manual (ARM)*. 2017, [cit. 2018-05-09]. Dostupné z: https://ftp.isc.org/isc/bind9/cur/9.11/doc/arm/Bv9ARM.pdf

[28] Red Hat, Inc.: *Keycloak Documentation*. 2018, [cit. 2018-05-07]. Dostupné z: https://www.keycloak.org/docs/3.2/server_admin/topics/identity-broker/overview.html

[29] Red Hat Inc.: *V4/External Authentication/Setup*. 2018, [cit. 2018-05-07]. Dostupné z: https://www.freeipa.org/page/V4/External_Authentication/Setup

[30] Red Hat Inc.: *V4/External Authentication*. 2018, [cit. 2018-05-06]. Dostupné z: https://www.freeipa.org/page/V4/External_Authentication

[31] zmartzone: *mod_auth_openidc*. 2018, [cit. 2018-05-8]. Dostupné z: https://github.com/zmartzone/mod_auth_openidc

[32] Pazdziora, J.: *Apache module mod_auth_openidc*. 2017, [cit. 2018-05-8]. Dostupné z: https://github.com/adelton/mod_lookup_identity

[33] Red Hat Inc.: *New Passwords Expired*. 2018, [cit. 2018-05-10]. Dostupné z: https://www.freeipa.org/page/New_Passwords_Expired

[34] Red Hat Inc.: *Centralized Logging*. 2018, [cit. 2018-05-13]. Dostupné z: https://www.freeipa.org/page/Centralized_Logging

[35] Red Hat Inc.: *IPA Server - API Browser*. 2018, [cit. 2018-05-14]. Dostupné z: https://ipa.demo1.freeipa.org/

[36] freeipa: *FreeIPA Ansible roles*. 2018, [cit. 2018-05-13]. Dostupné z: https://github.com/freeipa/ansible-freeipa

Dodatok **A**

Zoznam použitých pojmov

Autentikácia Proces, v ktorom sa kontroluje, že overovaná identita je tá, za ktorú sa vydáva.

Autorizácia Proces, v ktorom sa overuje, či daná identita má prístupové práva k nejakej funkcionalite/údajom.

Best Practice Zaužívaný spôsob používania softvéru, overený časom alebo propagovaný samotným výrobcom softvéru.

Cache Dočasná pamäť využívaná pre rýchlejšie získanie už skôr získanej informácie.

Deployment Nasadenie (nainštalovanie) aplikácie na nejakom systéme.

Downtime Čas, kedy je server nedostupný z dôvodu údržby alebo výpadku.

IPv4 Internet Protocol version 4.

IPv6 Internet Protocol version 6.

Log Záznam činnosti.

Timestamp Časová značka, ktorá vyjadruje čas vzniku nejakej udalosti.

Token Objekt, ktorý reprezentuje právo vykonať nejakú operáciu.

Škálovateľnosť Vlastnosť systému - schopnosť reagovať na zvýšený počet požiadavkov. Napríklad pomocou rozširovania prvkov systému v prípade väčšej záťaže a prípadne uberať, pokiaľ výpočetná sila nie je potrebná.

Vysoká dostupnosť Vlastnosť systému, vďaka ktorej je systém schopný obsluhovať požiadavky aj v prípade, že jeden z jeho prvkov nie je dostupný.

Dodatok B

Zoznam použitých skratiek

API Application Programming Interface

CA Certifikačná autorita

CLI Command Line Interface

CSV Comma Separated Values

FQDN Fully Quallified Domain Name

HTTP Hypertext Transfer Protocol

ID Identita

IdM Identity Management

IdP Identity Provider

JSON Java Script Object Notation

LDAP Lightweight Directory Access Protocol

MIT Massachusetts Institute of Technology

NFS Network File System

OIDC Open ID Connect

PoC Proof of Concept

SSL Secure Sockets Layer

WAN Wide Area Network

XML Extensible Markup Language

Dodatok C

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `ansible` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF