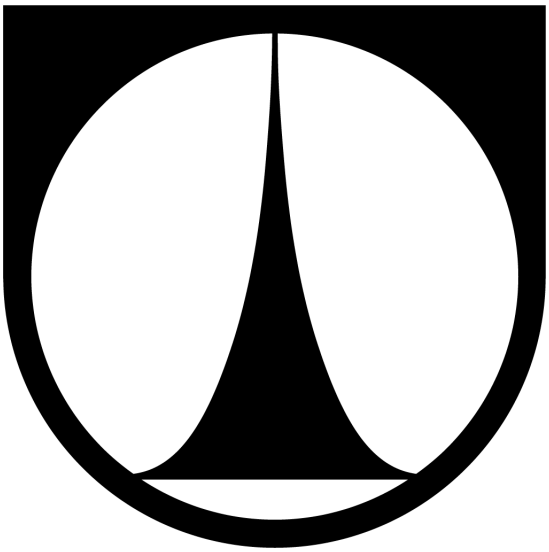
Technická univerzita v Liberci

Fakulta mechatroniky, informatiky a mezioborových studií

DIPLOMOVÁ PRÁCE



Bc. Zdeněk Perutka

Ověřování Verse serveru proti LDAPu a Kerberos serveru

Verse server authentication over LDAP and Kerberos

Ústav Nových technologií a aplikované informatiky

Vedoucí diplomové práce: Ing. Jiří Hnídek, Ph.D.

Studijní program: **N2612 – Elektrotechnika a informatika**

Studijní obor: **1802T007 – Informační technologie**

Liberec 2014

Zadání

1. Nejprve se důkladně seznamte s technologií LDAP a Kerberos.

2. Implementujte do Verse serveru ověřování uživatelských účtů proti LDAPu a Kerberos serveru. Zároveň proveďte kerberizaci knihovny, která se používá k implementaci Verse klientů.

3. Nakonec nové metody ověřování uživatelských účtů důkladně otestujte a vytvořte k nim odpovídající dokumentaci.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych chtěl poděkovat zejména své rodině a přátelům za podporu, sestře Šárce za trpělivou pomoc s pravopisem. Velký dík patří také vedoucímu práce, Ing. Jiřímu Hnídkovi, Ph.D., za konzultace a užitečné rady při řešení zadání.

Kontakt

E-mail: zdenek.perutka@tul.cz, z.perutka@gmail.com

Anotace

Tato diplomová práce má za úkol rozšíření síťového protokolu Verse, zabývajícího se real-timovým sdílením dat o podporu ověřování uživatelských účtů proti LDAPu a jeho kerberizací. Toto rozšíření Verse protokolu je důležité pro zvýšení jeho komplexnosti.

V úvodu práce je stručně popsána podstata fungování protokolů LDAP a Kerberos. Následně je předložen stručný přehled implementací LDAP a Kerberos serverů a knihoven umožňujících implementaci LDAP nebo Kerberos klientů včetně výběru nejvhodnějších knihoven pro implementaci do knihovny implementující protokol Verse.

Hlavními účely práce bylo vytvoření funkcí pro načítání uživatelů z LDAP serveru a jejich autentizaci na straně Verse serveru, autentizace Kerberos uživatele a následná kerberizace knihovny protokolu Verse. Práce se zabývá změnou rozhraní pro odchytávání signálů na Verse serveru na robustnější rozhraní vhodné pro vícevláknové aplikace.

Funkčnost a správnost rozšířeného Verse protokolu je v závěrečné části práce vyzkoušena na vzorové aplikaci, která byla nově rozšířena za účelem testování nových funkcí. Testování probíhalo zadáváním správných i špatných přihlašovacích údajů za užití obou nových autentizačních metod.

Klíčová slova: Verse, LDAP, Kerberos, autentizace, síťový protokol

Annotation

The thesis is focused on extension of the Verse network protocol which is used for real-time data sharing. The extension bases on including support for authentication over LDAP and Kerberos which is very important in context of complexity enhancement.

Introduction of the thesis briefly describes the nature of the LDAP and the Kerberos. Latter, a brief overview of implementations of LDAP, Kerberos servers and libraries which enable LDAP or Kerberos is offered, following by a selection of the two best libraries, to be used for implementation into the Verse protocol library.

The main aims of the study were to create functions for loading user accounts from LDAP server and authenticating them on Verse server side and authentication of Kerberos user and consecutive Kerberizing of Verse protocol library. Change of the signal interface on Verse server to a more robust interface suitable for multithreaded applications is also included.

Functionality of the extended Verse protocol was checked on a sample application in the final parts of the thesis. An already written sample application was also extended in order to test new functions. Testing was proceeded by inserting valid and invalid credentials, using both new authentication methods.

Key words: Verse, LDAP, Kerberos, authentication, network protocol

Obsah

Seznam symbolů a zkratek

AES Advanced Encryption Standard

API Application Programming Interface

AS Authentication Service

CDDL Common Development and Distribution License

CMake Cross-Platform Makefile Generator

CSV Comma-Separated Values

DED Data Exchange Definition

DIT Directory Information Tree

DN Distinguished Name

DNS Domain Name System

GPLv3 General Public License version 3

IETF The Internet Engineering Task Force

KDC Key Distribution Center

LDAP Lightweight Directory Access Protocol

LDIF LDAP Data Interchange Format

MIT Massachusetts Institute of Technology

ITU-T International Telecommunication Union Telecommunication Standardization Sector

PAM Pluggable Authentication Modules

RDN Relative Distinguished Name

SASL Simple Authentication and Security Layer

TCP Transmission Control Protocol

TGS Ticket-Granting Server

TGT Ticket-Granting Ticket

TLS Transport Layer Security

UDP User Datagram Protocol

UTF-8 Universal Character Set Transformation Format — 8-bit

1 Úvod

Úvodem bude stručně popsáno současné řešení problematiky uživatelských účtů a šifrování komunikace. Dále budou popsány cíle práce.

1.1 Současné řešení problematiky

V současné době Verse server načítá uživatelské účty z Comma-Separated Values (CSV) souboru [1]. V něm je definováno uživatelské jméno, heslo, identifikační číslo uživatele a jeho skutečné jméno, viz. následující výpis.

```
username,password,UID,real name
usr@ZDN.LOCAL,abc,2000,Test User
```

Verse server tyto uživatelské účty načítá při startu. Informace vytěžené z CSV souboru ukládá do struktury pro ukládání uživatelů. Tato struktura má atributy pro jednotlivé informace definované v CSV souboru. Navíc je zde atribut určující, zda je možné se k tomuto účtu přihlásit a atributy odkazující na předchozí a následující uživatelský účet. Jednotlivé účty jsou tedy uchovány ve spojovém seznamu.

Při pokusu o přihlášení Verse server obdrží od klienta uživatelské jméno a heslo. Po obdržení přihlašovacích údajů Verse server prochází spojový seznam uživatelských účtů a porovnává uživatelská jména a hesla jednotlivých účtů s obdrženými údaji. Jsou-li zadány správné přihlašovací údaje, server vytvoří uživateli avatar a klientovi odešle zprávu o úspěšné autentizaci, v opačném případě zprávu o neúspěšné autentizaci.

Komunikace mezi Verse serverem a Verse klientem je v současné době šifrována pomocí Transport Layer Security (TLS) [2]. Komunikace může probíhat i nešifrovaně.

1.2 Cíle práce

Prvním cílem práce je rozšířit Verse server, aby byl schopen načíst seznam uživatelů z LDAP serveru. K tomu bude nutné na Verse serveru vytvořit funkci, jejíž pomocí se server připojí k LDAP serveru a odešle mu dotaz požadující seznam uživatelů. Z obdrženého seznamu bude pro jednotlivé uživatele nutné vytěžit uživatelské jméno,

identifikační číslo uživatele jméno a příjmení. Dále je snahou, aby se Verse server při obdržení vhodně zvoleného signálu pokusil z LDAP serveru načíst nově přidané uživatelské účty.

Druhým cílem je vlastní ověření uživatelských účtů proti LDAPu. Verse server se pokusí pomocí přihlašovacích údajů obdržených od Verse klinta přihlásit k LDAP serveru. Pokud bude toto přihlášení úspěšné a zároveň se bude uživatel se zadaným uživatelským jménem nacházet v seznamu uživatelů Verse serveru, Verse server vytvoří uživateli avatar a klientovi odešle zprávu o úspěšné autentizaci, v opačném případě odešle zprávu o neúspěšné autentizaci.

Třetím cílem je ověření Verse serveru proti Kerberos serveru. Verse server si načte uživatelské účty z CSV souboru nebo z LDAPu. Verse klient se přihlásí ke Kerberos serveru a zažádá o přístup ke službě Verse. Pomocí obdržených klíčů naváže spojení s Verse serverem. Verse server poté projde svůj seznam uživatelů (zkontroluje zda uživatel s daným uživatelským jménem v rámci Verse serveru existuje) a odešle klientovi zprávu o úspěšné respektive neúspěšné autentizaci.

Čtvrtým cílem práce je kerberizace Verse protokolu. Ta bude realizována šifrováním dat přenášených mezi Verse serverem a Verse klientem pomocí klíče získaného během autentizace uživatele proti Kerberos serveru.

Pátým cílem je otestování nově implementovaných autentizačních metod. Při tomto testování bude snaha postihnout a otestovat všechny stavy, kterých se mohou nově implementované funkce nacházet.

Nakonec bude sepsána uživatelská dokumentace. Ta bude obsahovat instrukce k provozování Verse serveru a Verse klienta za použití nově implementovaných metod ověřování uživatelských účtů.

2 Popis protokolů Verse, LDAP a Kerberos

V této kapitole bude stručně popsán síťový protokol Verse. Dále bude vysvětlen princip funkce protoklů LDAP a Kerberos. Popsány budou i další pojmy související s LDAPem, Kerberem a jejich implementací do Verse protokolu.

2.1 Verse

Verse je síťový protokol typu klient-server. Je určen pro real-timové sdílení dat především mezi grafickými aplikacemi.

Původní Verse protokol byl vyvíjen Uni-Verse konsorciem v rámci 6. rámcového programu Evropské unie. Do konsorcia patřilo několik významných univerzit a výzkumných institucí jako KTH, Fraunhoufer Institut, Helsinki University of Technology, Interactive Institute a Blender Foundation. Verse protokol byl od počátku navrhován pro efektivní real-timové sdílení dat především mezi grafickými aplikacemi a měl ambici stát se univerzálním protokolem pro komunikaci mezi grafickými aplikacemi. Po ukončení financování Evropskou unií vývoj protokolu ustal a ten se bohužel z mnoha důvodů nakonec nerozšířil [3].

V současné době vyvíjí Ing. Jiří Hnídek, Ph.D. novou verzi Verse protokolu. Snahou je, aby tato verze byla robustnější a především lépe implementovatelná do současných i budoucích aplikací. Protokol je implementován v jazyce C.

2.2 LDAP

Lightweight Directory Access Protocol (LDAP) je odlehčenou verzí protokolu X.500 [4]. Jedná se o protokol určený pro přístup k adresářovým službám. Protokol LDAP je standardizován organizací The Internet Engineering Task Force (IETF) [5] a je to protokol typu klient-server. Jeden nebo více LDAP serverů obsahují data, jež tvoří adresářový strom (Directory Information Tree, DIT). Data na serveru jsou uspořádána hierarchicky.

Klient posílá serveru dotazy. Tyto dotazy slouží například k autentizaci uživatele, k přidávání a modifikaci záznamů nebo jejich vyhledávání či porovnávání. Dotazy de-

finuje funkční model LDAP, viz. 2.8. Server odesílá zpět odpovědi na dotazy, případně odkaz na jiný LDAP server, z něhož lze požadovanou informaci získat. Protokol LDAP je hojně používaný v praxi. Mezi jeho nejčastější uplatnění patří:

- Autentizace uživatelů
- Úložiště uživatelských nastavení
- Řízení přístupu k datům
- Adresář kontaktů
- Úložiště konfigurací aplikací
- Reprezentace struktury organizace

2.3 Adresářová služba

Adresářová služba je databáze optimalizovaná pro procházení a vyhledávání záznamů. Adresáře tvoří hierarchickou strukturu a mohou obsahovat různorodé datové typy. Adresářové služby obecně nepodporují komplikované transakce a nekontrolují referenční integritu.

Dobrým příkladem adresářové služby je Domain Name System (DNS). Je to služba globálně distribuovaná. Jednotlivé DNS servery jsou uspořádány hierarchicky.

2.4 X.500

Jak již bylo řečeno LDAP je odlehčenou verzí X.500. X.500 je skupina standardů popisujících adresářové služby, jež byly schváleny v roce 1988 a byly navrženy organizací ITU-T. V Praxi se X.500 příliš nepoužívá naopak převládá LDAP, který původně sloužil pouze ke komunikaci s X.500.

2.5 Informační model LDAP

Informační model LDAP definuje, jaké informace a datové typy lze ukládat na adresářovém serveru. Data jsou uchovávána coby záznamy, jež jsou hierarchicky

uspořádány do stromové struktury. Jednotlivé záznamy tvoří množinu atributů. Atributy chápeme jako dvojici jméno - hodnota, která určuje stav konkrétního záznamu.

Záznamy musejí odpovídat definovanému schématu. Schéma určuje povolené objektové třídy a k nim náležící atributy. Každý záznam je pak instancí nějaké objektové třídy. Tím pádem záznam musí obsahovat všechny povinné atributy příslušné objektové třídy a může obsahovat nepovinné atributy této třídy.

2.5.1 Objektové třídy

Objektové třídy mohou být tří druhů:

- **STRUCTURAL** - odvozená objektová třída. Právě od těchto tříd se odvozují jednotlivé záznamy. Každý záznam pak musí být odvozený alespoň od jedné této třídy. Pokud je záznam odvozen od více tříd, musí být tyto třídy v dědičném vztahu.
- **ABSTRACT** - abstraktní objektová třída. Nelze od ní odvozovat jednotlivé záznamy, slouží pouze jako rodičovská třída, od níž se odvozují odvozené třídy.
- **AUXILIARY** - doplňkové objektové třídy. Jejich pomocí lze záznamům přidávat další atributy. Každý záznam může být odvozen od libovolného počtu doplňkových tříd.

Na následujícím výpisu je ukázka definice objektové třídy.

```
objectclass ( 2.16.840.1.113730.3.2.2
    NAME 'extendedPerson'
    SUP organizationalPerson
    STRUCTURAL
    MUST (
      uid )
    MAY (
      employeeNumber $ givenName $ jpegPhoto $ userCertificate )
    )
```

Direktiva *SUP* určuje rodičovskou objektovou třídu, tedy třídu od níž daná třída dědí všechny povinné i nepovinné atributy. Direktiva *STRUCTURAL* určuje, že se jedná o odvozenou objektovou třídu, *MAY* je seznam nepovinných atributů. Povinné atributy se určují direktivou *MUST*.

2.5.2 Atributy

Podobně jako objektové třídy jsou přesně definovány i jejich atributy. Ty se pak skládají právě do objektových tříd, viz výše. U atributů funguje, stejně jako u objektových tříd, dědičnost. Následující výpis obsahuje ukázku definice a atributu.

```
attributetype ( 2.5.4.3 NAME ( 'cn' 'commonName' )
  DESC 'RFC2256: common name(s) for which the entity is known by'
  SUP name )
```

U atributů je také možno definovat syntaxi, tedy jakou strukturu budou data mít (např. řetězec znaků UTF-8, celé číslo, JPEG obrázek, atd.).

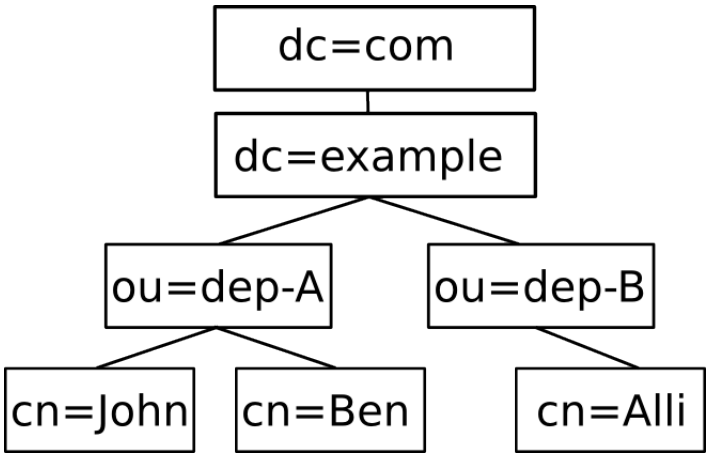
2.6 Jmenný model LDAP

Jmenný model LDAP definuje, jakým způsobem se přistupuje k datům a jakým způsobem jsou data organizována.

2.6.1 Rozlišovací jméno

Rozlišovací jméno (Distinguished Name, DN) slouží k jednoznačné identifikaci jednotlivých záznamů. V rámci jedné větve mluvíme o relativním rozlišovacím jménu (Relative Distinguished Name, RDN), které musí být v dané větvi a úrovni stromu taktéž jednoznačné. RDN se skládá ze jména atributu, jenž jednoznačně identifikuje záznam v jeho větvi a úrovni, a jeho hodnoty.

DN se pak sestává z jednotlivých RDN, které se skládají v pořadí od RDN konkrétního záznamu postupně až ke kořeni. DN je tedy určeno cestou od kořene k záznamu.



Obrázek 1: Příklad DIT

Na obrázku 1 je příklad DIT fiktivního LDAP serveru example.com. Jednotlivé obdélníky představují záznamy a jsou v nich vepsány RDN těchto záznamů. Pro záznam identifikovaný ve své větvi RDN *cn=John* bude DN vypadat následovně: *cn=John,ou=dep-A,dc=example,dc=com*.

2.7 Bezpečnostní model LDAP

Bezpečnostní model LDAP se stará o řízení přístupu k adresářovému serveru. Řeší dva úzce spjaté problémy a to autentizaci uživatelů a jejich autorizaci.

2.7.1 LDAP autentizace

Autentizací rozumíme ověření identity uživatele, přistupujícího k LDAP serveru. Autentizace je úspěšná, pokud zadané DN odpovídá nějakému záznamu na tomto LDAP serveru a pokud tento záznam má atribut *userPassword*, který obsahuje heslo. To se musí schodovat s heslem zadaným. Heslo je samozřejmě možné ukládat v šifrované podobě.

Možnosti autentizace na LDAP serveru jsou variabilní, mezi nejběžnější patří:

- **Anonymní autentizace** - při navazování spojení (operace *bind* viz. kapitola 2.8) nejsou přenášeny autentizační údaje, přístup je tak umožněn komukoliv.

- **Jednoduchá autentizace** - při operaci *bind* se po nechráněném kanále pošle DN a heslo uživatele.

- **Jednoduchá autentizace přes zabezpečený kanál** - stejně jako v předchozím případě je přenášeno DN uživatele a jeho heslo, avšak v tomto případě po kanále, který je zabezpečen SSL nebo TLS [2]. Heslo je přenášeno v nezměněné podobě i v případě, že je na serveru uloženo v podobě haše.

- **Proxy autentizace** - v tomto případě je vytvořen uživatel, který má přiděleno právo číst hesla ostatních uživatelů (viz. kapitola 2.7.2). Tento uživatel pak pomocí operace *compare* (viz. kapitola 2.8) provádí autentizaci uživatelů.

- **Simple Authentication and Security Layer (SASL)** [6] je vrstva, umožňující autentizaci pomocí jiných, na spojení založených protokolů.

2.7.2 LDAP autorizace

Po úspěšné autorizaci uživatele přichází na řadu jeho autorizace. Ta se stará o to, aby měl uživatel přístup pouze k záznamům a atributům, ke kterým přístup mít má.

Přístupová práva je možno nastavit velmi flexibilně. Na následujícím výpisu je příklad nastavení uživatelských práv.

```
access to attr=userPassword
    by self =xw
    by anonymous auth
    by * none

access to *
    by self write
    by users read
    by * none
```

Výpis začíná klíčovými slovy *access to*, za nimi následuje určení kterého záznamu, případně atributu, se bude následující definice přístupových práv týkat. V první části

tohoto předpisu je řešen přístup k atributu *userPassword*, ve druhé je *, zastupující veškeré záznamy na serveru.

Následuje *by*, za nímž určujeme komu přístupová práva přidělujeme. V tomto případě jsou použita klíčová slova: *self*, tedy přístup pro DN patřící tomuto záznamu, *anonymous* pro anonymní uživatele, * pro všechny uživatele a *users* pro autentizované uživatele.

Dále už jsou konkrétní přístupová práva. V našem případě jsou použita *auth* přístupné právo pro autentizaci, *read* lze pouze číst, *write* lze číst i zapisovat, *none* nelze číst ani zapisovat a *=xw*. Právo *=xw* umožňuje daný atribut či objekt měnit, ale neumožňuje jej číst.

2.8 Funkční model LDAP

Funkční model LDAP se stará o přístup k záznamům a o jejich modifikaci. K tomu používá tři druhy operací a to autentizační operace, aktualizační operace a dotazovací operace. Tyto operace bývají přímo obsaženy v jednotlivých implementacích LDAP serverů.

Autentizační operace, jak již napovídá název, slouží k přihlašování uživatelů. Jsou to především operace *bind*, sloužící k přihlášení uživatele a *unbind*, která se stará naopak o odhlášení uživatele.

Aktualizační a dotazovací operace pak slouží k manipulaci s daty. Mezi nejběžnější tyto operace patří:

- *add* - pro přidávání nových záznamů
- *compare* - pro porovnávání záznamů
- *delete* - pro mazání stávajících záznamů
- *modify* - pro změnu stávajících záznamů
- *search* - pro vyhledávání záznamů

2.9 LDIF

LDAP Data Interchange Format, neboli LDIF [7] je standardizovaný textový formát určený k popisování záznamů na adresářovém serveru. Tento textový formát je velmi dobře čitelný pro člověka a tak usnadňuje práci se záznamy. Ať už jde o jejich vytváření, editaci, mazání, nebo přenos mezi různými LDAP servery. Adresářové servery totiž podporují jak import dat z tohoto formátu, tak i export do něj.

Na následujícím výpisu je příklad dat v textovém formátu LDIF.

```
dn: cn=John,ou=dep-A,dc=example,dc=com
cn: John
givenName: John
sn: Higgins
objectClass: inetOrgPerson
objectClass: top
userPassword: {SSHA}36EZHdTOgqbYczBmZe8avOOkNavBcWd+
```

2.10 Kerberos

Kerberos [8] je síťový autentizační protokol. Je určený především pro model klient-server. Oběma stranám umožňuje bezpečně prokázat svou identitu, respektive ověřit identitu protistrany. Kromě toho zabraňuje odposlechnutí komunikace, její zopakování a zaručuje integritu dat.

Kerberos byl vyvinut na Massachusetts Institute of Technology (MIT) v rámci projektu Athena. Pojmenován je po mýtickém řeckém trojhlavém psu střežícím bránu do podsvětí.

Protokol Kerberos je založen na symetrické kryptografii a vyžaduje důvěryhodnou třetí stranu, Key Distribution Center (KDC). KDC se skládá ze dvou celků, jimiž jsou Authentication Service (AS) a Ticket-Granting Server (TGS). K šifrování komunikace používá symetrickou blokovou šifru Advanced Encryption Standard (AES) [9].

Kerberos používá tikety, které slouží k prokázání totožnosti jednotlivých uzlů. Každý účastník komunikace má svůj tajný klíč, který je uložen v databázi KDC. Právě

na základě znalosti tohoto klíče je ověřována identita uzlu. Pro účely komunikace mezi jednotlivými uzly generuje KDC klíč relace *(session key)*, kterým svou komunikaci uzly šifrují.

Pokud chce klient přistoupit k nějaké službě nejprve se autentizuje u AS pomocí sdíleného tajného klíče (zadaného hesla) a obdrží tiket, poté může pomocí toho tiketu požádat o přístup ke službě u TGS, od kterého obdrží další tiket sloužící ke komunikaci se serverem služby. Pomocí tiketu obdrženého od AS může u TGS žádat o přístup k více různým službám, pro každou tuto službu pak obdrží od TGS tiket. Podrobněji jsou jednotlivé fáze popsány v kapitole 2.10.1.

2.10.1 Zahájení komunikace

Zahájení Kerberos komunikace se skládá ze čtyř fází. Ani při jedné z nich není posíláno heslo komunikujícího uzlu po síti. Schéma komunikace je na obrázku 2.

První fází je login uživatele. Ten probíhá pouze lokálně na klientském počítači, kde uživatel zadá identifikátor a heslo. Následně je na vloženém hesle provedena hašovací funkce. Výstup z ní je pak tajným klíčem klienta (K1).

Druhou fází je autentizace uživatele. V této fázi již probíhá komunikace po síti. Nejprve klient odešle AS nezašifrovanou žádost obsahující jeho identifikátor. Podle toho AS vyhledá uživatele v databázi. Pokud jej nalezne, odešle mu dvě zprávy:

- *Zprávu A* obsahující klient-TGS klíč (K2), zašifrovanou K1.
- *Zprávu B* obsahující Ticket-Granting Ticket (TGT), v němž je identifikátor a adresa klienta, dobu platnosti tiketu a K2. TGT je zašifrován tajným klíčem TGS (K3).

Po obdržení zpráv se klient nejprve pokusí rozšifrovat zprávu A a to klíčem K1. V případě že uživatel zadal chybné heslo, se nepodaří zprávu dešifrovat. V opačném případě pak získá K2. TGT ze Zprávy B není klient schopen dešifrovat, ale tento tiket je použit v následující fázi.

Následuje třetí fáze, autorizace přístupu ke službě. Klient žádá TGS o přístup ke službě zasláním dvou zpráv:

- *Zprávy C* která obsahuje TGT a identifikátor požadované služby.
- *Zprávy D* obsahující identifikátor klienta a časovou značku, zpráva je šifrována K2.

TGS pomocí klíče K3 rozšifruje TGT a získá tak K2. Tím rozšifruje zprávu D a získá tak identifikátor klienta a časovou známku. Pak pošle klientovi dvě zprávy:

- *Zprávu E* jenž obsahuje klient-server tiket, obsahující identifikátor a adresu klienta, dobu platnosti tiketu a klient-server klíč (K4). Klient-server tiket je zašifrován tajným klíčem služby (K5).
- *Zprávu F* která obsahuje K4 a je zašifrována klíčem K2.

Poslední fází je žádost klienta o službu. Ten se po obdržení zpráv E a F spojí s aplikačním serverem a tomu odešle opět dvě zprávy:

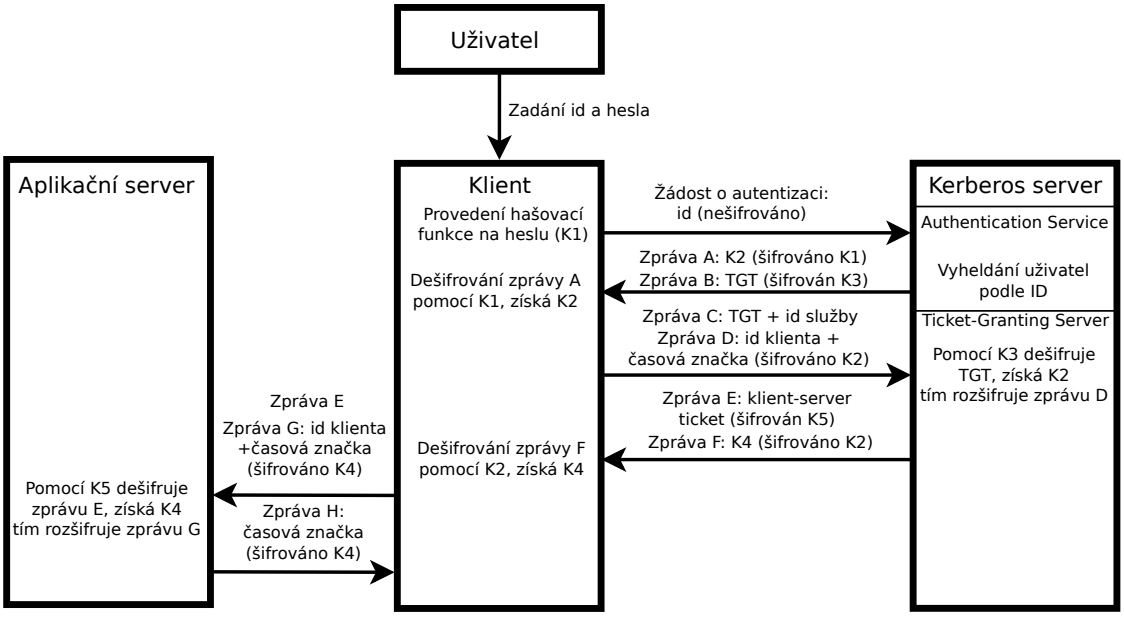
- *Zprávu E*, tedy zprávu kterou předtím obdržel od TGS.
- *Zprávu G* ta je podobná zprávě D, obsahuje identifikátor klienta a časovou značku. Zpráva je šifrována klíčem K4.

Aplikační server dešifruje pomocí K5 zprávu E. Tak získá K4, tím dešifruje zprávu G. Pro potvrzení své identity pak server odešle následující zprávu:

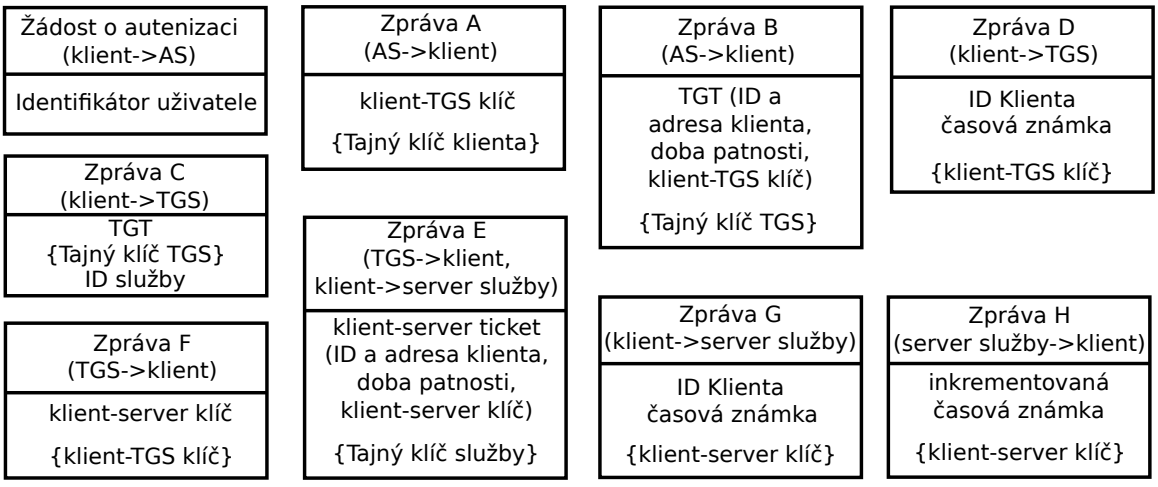
- *Zprávu H* obsahující inkrementovanou časovou značku ze zprávy G. Zpráva H je zašifrována klíčem K4.

Zprávu H klient dešifruje. Pokud časová značka souhlasí, server je důvěryhodný. Mezi klientem a serverem pak probíhá běžná komunikace. Tato komunikace je šifrována klient-server klíčem. Jednotlivé zprávy jsou ilustrovány na obrázku 3.

Aby Kerberos fungoval správně, je nutná časová synchronizace účastníků komunikace. Rozdíl mezi jednotlivými účastníky nesmí být víc než 5 minut. Dále je nutný nepřetržitý provoz KDC. Hlavní KDC proto bývá duplikováno na jiné servery, takzvané ,,slave“ servery, které jej zastoupí v případě výpadku. Jelikož KDC zaručuje identitu komunikujících stran, musí být samo o sobě dobře zabezpečeno.



Obrázek 2: Výměna zpráv při Kerberos autentizaci



Obrázek 3: Schémata jednotlivých zpráv, klíč použitý k šifrování zprávy je ve složených závorkách

3 LDAP a Kerberos servery

Tato kapitola pojednává o OpenSource implementacích LDAP a Kerberos serverů. Je v ní také zdůvodněna volba implementací vybraných pro testování ověřování uživatele Verse serveru proti LDAP serveru a Kerberos serveru. Implementacemi LDAP a Kerberos serverů jsem se již zabýval v rámci svého Magisterského projektu.

3.1 LDAP servery

V této části se nachází rešerše OpenSource implementací LDAP serverů.

3.1.1 389 Directory Server

389 Directory Server je vyvíjen firmou Red Hat [10]. Podporuje SSL verze 3, TLS verze 1 a SASL. Umožňuje také replikaci serverů. Dokáže vykonat řádově tisíce operací za sekundu, obsluhovat desetitisíce uživatelů zároveň a udržovat desítky miliónů záznamů. Je distribuován pod licencí General Public License version 3 (GPLv3) [11].

3.1.2 Apache Directory

Apache Directory je naprogramován v jazyce Java, vyvíjí jej Apache Software Foundation [12]. Podporuje Kerberos V5. Klade si za cíl zavést do světa LDAP triggery, uložené procedury a pohledy známé z relačních databází. Je poskytován s licencí Apache License, Version 2.0 [13].

3.1.3 OpenDS

OpenDS byl vyvíjen firmou Sun Microsystems. Po jejím zániku je vyvíjen komunitně. Je navržen tak, aby dokázal obsáhnout velká nasazení, poskytoval vysoký výkon, byl snadno rozšiřitelný a lehce zaveditelný. OpenDS je rozšiřován s licencí Common Development and Distribution License (CDDL) [14].

3.1.4 OpenLDAP

OpenLDAP je poskytován pod OpenLDAP licencí [15], podobnou BSD licenci. Vyvíjí jej OpenLDAP Foundation [16] a je naprogramován v jazyce C. Jedná se v současné době o nejrozšířenější LDAP server. Nabízí široké možnosti konfigurace a velmi detailní manuál. Je vhodný také proto, že je možné k němu na webu najít velké množství návodů a tutoriálů.

Tato implementace byla vybrána pro praktické testování ověření uživatele Verse serveru proti LDAPu. Důvodů pro výběr této implementace bylo několik, především je to velká komunita uživatelů OpenLDAP. Díky tomu se v případě nějakého problému lépe shání informace. Dalším důvodem je přítomnost této implementace ve standardních repositářích linuxových distribucí.

3.1.5 Komerční LDAP servery

Kromě OpenSource implementací existuje i mnoho komerčních implementací. Jsou jimi např. Open Directory od firmy Apple, Active Directory od Microsoftu, eDirectory od Novellu a Oracle Internet Directory od firmy Oracle.

3.2 Kerberos Servery

Tato část práce se věnuje OpenSource implementacím Kerberos serveru.

3.2.1 MIT Kerberos

MIT Kerberos je vyvíjen na Massachusetts Institute of Technology (MIT). Do jeho vývoje však přispěly i komerční firmy jako: Cygnus Support, Novell, OpenVision Technologies, Oracle, Red Hat, Sun Microsystems a FundsXpress. Přesto má velmi svobodnou licenci, umožňující volné užívání, kopírování i redistribuci [17].

Tato implementace byla vybrána pro praktické testování ověření uživatelů Verse serveru proti Kerberos serveru jakožto nejčastěji používaná implementace. Další výhodou je, že je vyvíjena na MIT stejně jako protokol Kerberos.

3.2.2 Heimdal

Heimdal je implementace Kerberos vyvíjená převážně ve Švédsku. Je poskytován pod BSD-like licencí [18].

3.2.3 Komerční Kerberos servery

Podobně jako v případě LDAP serverů existují i komerční implementace Kerberos serverů. Příkladem mohou být implementace od firem CyberSafe a Microsoft.

4 Knihovny implementující LDAP a Kerberos

V této části práce je obsažena rešerše knihoven umožňujících implementaci klientské aplikace pro ověřování proti LDAP serveru a Kerberos serveru.

4.1 Spring LDAP

Spring LDAP je knihovna, která poskytuje zjednodušený rámec operací LDAP. Je naprogramována v jazyce Java. Vyvíjí ji SpringSource [19], jež je divizí VMware [20]. Na stránkách knihovny je k dispozici kompletní dokumentace [21]. Knihovna je distribuována pod licencí Apache License Version 2.0 [13].

4.2 Python-ldap

Knihovna Python-ldap poskytuje objektově orientované API sloužící k přístupu k LDAP serverům z programů psaných v jazyce Python. Z velké části se jedná o obalovou knihovnu knihovny OpenLDAP viz. 4.4. Knihovna je poskytována pod Python-style licencí [22].

4.3 Perl LDAP

Knihovna Perl LDAP je kolekce Perl modulů, které poskytují objektově orientované API určené k interakci s LDAP serverem. Všechny moduly jsou psány kompletně v Jazyce Perl. To činí Perl LDAP nezávislým na platformě. Na stránkách knihovny se nachází kompletní dokumentace [23]. Perl LDAP je poskytován s licencí GPLv3 [11].

4.4 OpenLDAP

OpenLDAP poskytuje kromě implementace LDAP serveru také klientské API pro komunikace se serverem. Kromě kompletní implementace LDAP obsahuje i funkce umožňující autentizaci proti Kerberos serveru. Jinak o knihovně OpenLDAP platí to samé co o implementaci serveru OpenLDAP viz. 3.1.4. Tato knihovna byla vybrána pro implementaci ověření Verse serveru proti LDAPu. Mezi důvody výběru této

knihovny patří svobodná licence, implementace v jazyce C a přítomnost ve standardních repositářích linuxových distribucí.

4.5 MIT Kerberos

Také MIT Kerberos obsahuje knihovnu, sloužící jako API umožňující kerberizaci klientské aplikace. Knihovna je poskytována pod stejnou licencí jako implementace serveru, viz. 3.2.1. Tato knihovna byla vybrána pro implementaci ověření Verse serveru proti Kerberos serveru. A to ze stejných důvodů jako knihovna OpenLDAP tedy díky svobodné licenci, implementaci v jazyce C a přítomnosti ve standardních repositářích linuxových distribucí.

5 Ověření Verse serveru proti LDAPu

Tato část práce popisuje vlastní implementaci ověřování uživatelů Verse serveru proti LDAPu. Obsahuje také popis některých změn, které bylo nutné ve Verse serveru, v souvislosti s touto implementací udělat.

Samotný verse protokol je možné získat z adresy *https://github.com/verse/verse*, verze implementující podporu autentizace uživatelů proti LDAPu a Kerberos serveru se nachází na přiloženém CD a na *https://github.com/zdenek-perutka/verse*. Postup kompilace je popsán v příloze a také na výše zmíněném webu.

Jelikož je knihovna Verse protokolu implementována v jazyce C, k vlastní implementaci nových autentizačních metod bylo nutné použít tento programovací jazyk. Ještě před započetím implementace bylo nutné změnit soubry *CMakeLists.txt*, tak aby se Cross-Platform Makefile Generator (CMake) pokusil nalézt knihovny OpenLDAP a MIT Kerberos, a aby v případě jejich nalezení byly zdrojové kódy knihovny Verse při překladu slinkovány s těmito knihovnami.

Při implementaci bylo nutné udělat změny pouze na straně serveru. Na straně klienta nehraje roli, je-li použita LDAP autentizace.

5.1 Původní uživatelské účty Verse

Původně Verse server načítal uživatelské účty pouze ze souboru formátu CSV. Soubor byl načten při startu Verse serveru. Informace z něj načtené byly uloženy do struktury *VSUser*. Ta je definována v souboru *vs_user.h* a slouží právě k uchovávání uživatelských účtů. Server si tyto účty uchovává ve spojovém seznamu.

5.2 Úprava datových struktur

Kvůli implementaci ověřování Verse serveru proti LDAPu bylo nutné rozšířit některé datové struktury. Mezi nimi právě strukturu *VSUser*. Do níž byl přidán atribut *ldap_dn*, uchovávající rozlišující jméno uživatele na LDAP serveru, viz. kapitola 2.6.1. Rozšířen byl též kontext Verse serveru, struktura *VS_CTX*, ze souboru *vs_main.h*. Ta byla rozšířena celkem o pět atributů. A to *ldap_hostname* obsahující adresu LDAP serveru,

ldap_user s uživatelským jménem Verse serveru na LDAPu a *ldap_passwd* s jeho heslem. Dále *ldap_search_base*, tento atribut určuje na jaké části serveru bude probíhat vyhledávání uživatelů, a nakonec *ldap_version* určující používanou verzi LDAPu.

5.3 Načtení uživatelů z LDAP serveru

Načtení uživatelských účtů z LDAP serveru probíhá podobně jako z CSV souboru při startu serveru. Verse server se nejprve přihlásí svým uživatelským jménem k LDAP serveru a poté se jej dotáže na seznam uživatelů. Po obdržení seznamu tento seznam projde a informace o uživatelích uloží do příslušných struktur. Uživatel reprezentující Verse server na LDAP serveru musí mít nastaveno právo číst informace o ostatních uživatelích, viz. kapitola 2.7.2. Veškeré nové funkce sloužící k ověřování uživatelů Verse serveru proti LDAPu byly umístěny do souboru *vs_auth_ldap.c*.

```
int vs_load_user_accounts_ldap_server(VS_CTX *vs_ctx)
```

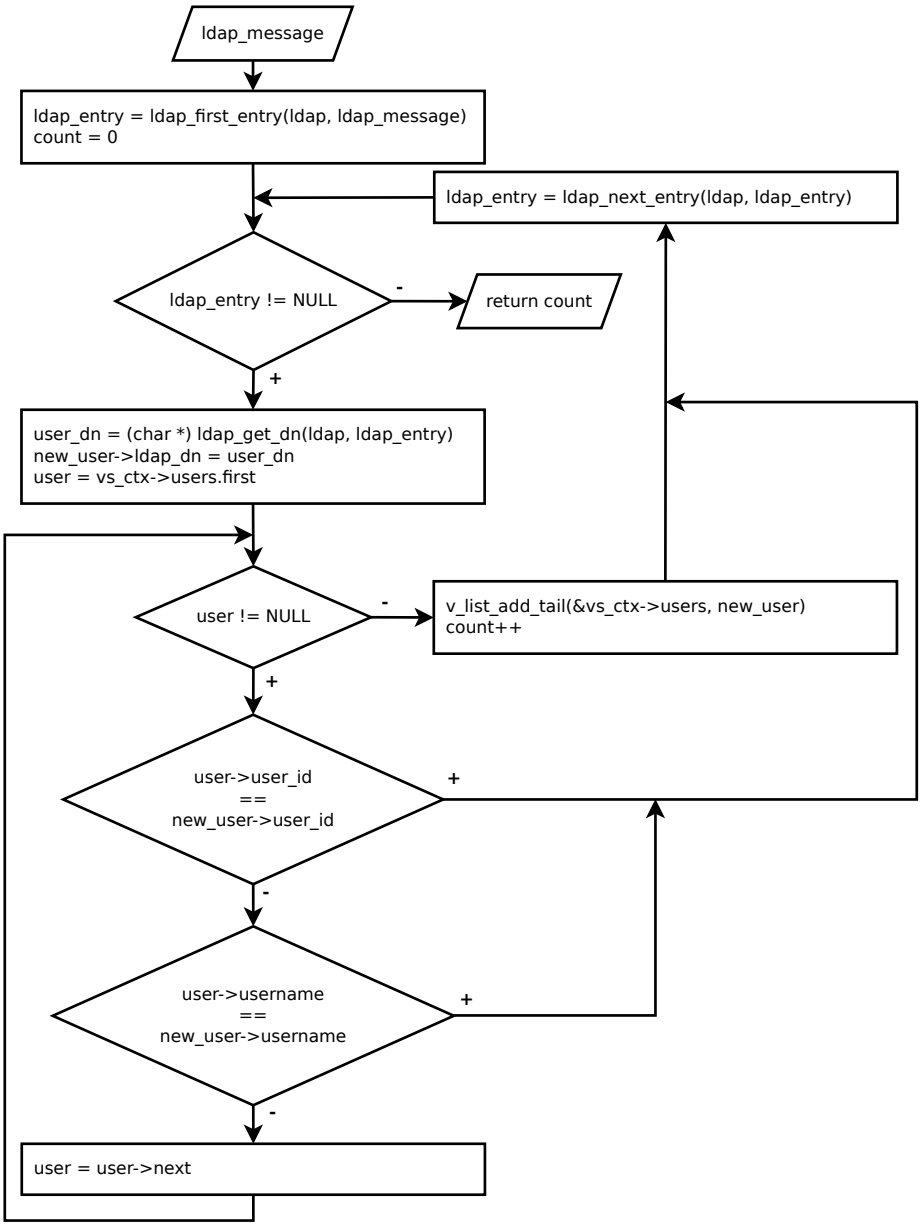
Získání uživatelských účtů z LDAP serveru je implementováno ve funkci *vs_load_user_accounts_ldap_server*, hlavička na výpisu výše. Ta má jako jediný parametr kontext Verse serveru. V případě úspěchu vrací 1, jinak vrací číslo chyby.

Zde je nejprve pomocí funkce *ldap_initialize* zinicializován kontext knihovny LDAP a funkce *ldap_set_option* nastaví požadovanou verzi protokolu LDAP. Pomocí funkce *ldap_sasl_bind_s* se Verse server pokusí přihlásit k LDAP serveru. O odeslání dotazu na LDAP server se postará funkce *ldap_search_ext_s*. Verse server se tak dotáže na seznam uživatelů, který obdrží ve struktuře *LDAPMessage*. Poté je volána funkce *vs_add_users_from_ldap_message*, jež se postará o vytěžení informací o jednotlivých uživatelích a jejich uložení do příslušných struktur.

```
int vs_add_users_from_ldap_message(struct VS_CTX *vs_ctx, LDAP *ldap,
        LDAPMessage *ldap_message)
```

Do funkce vstupuje kontext Verse server, kontext knihovny LDAP a struktura *LDAPMessage*, jež obsahuje zprávu se seznamem uživatelů. Vrácen je počet

uživatelských účtů přidaných do Verse serveru. Činnost funkce je znázorněna na obrázku 4.



Obrázek 4: Funkce vs_add_users_from_ldap_message

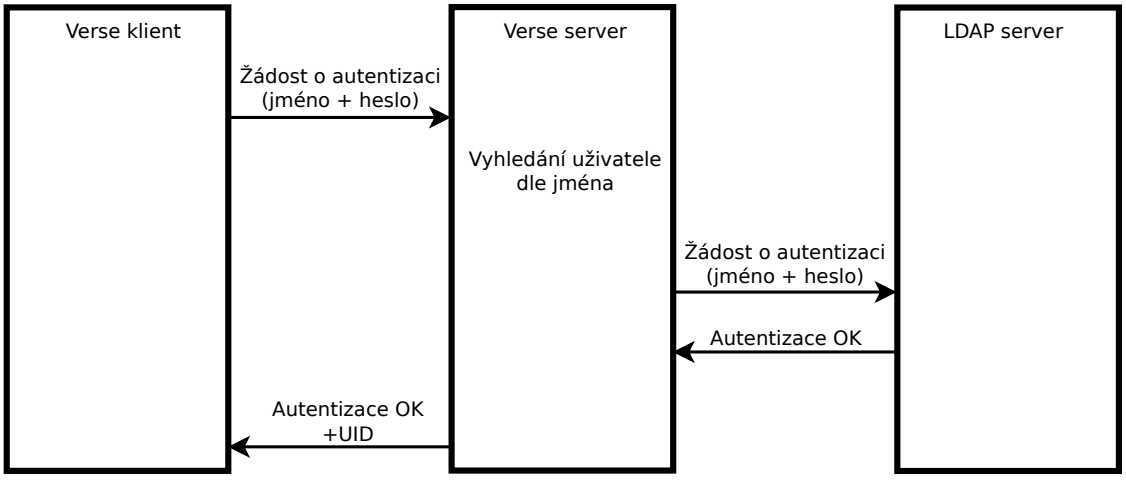
Funkce nejprve prochází jednotlivé záznamy ze zprávy od LDAP serveru. Z každého záznamu vytěží informace o uživatelském účtu jako jsou uživatelské jméno, identifikační číslo, jméno, příjmení a rozlišovací jméno (DN). Poté projde seznam uživatelů Verse

serveru a porovnává získané uživatelské jméno a identifikační číslo s uživatelskými jmény a hesly již existujících uživatelů. Pokud se uživatelské jméno nebo identifikační číslo shoduje se jménem nebo identifikačním číslem již existujícího uživatele, uživatelský účet nemůže být přidán. V opačném případě funkce přidá tohoto uživatele do seznamu uživatelů Verse serveru.

5.4 Ověření uživatele proti LDAPu

Při pokusu o autentizaci klient odešle svoje uživatelské jméno a heslo Verse serveru. Ten se pokusí nalézt uživatele ve svém seznamu. Pokud ho nalezne, pokusí se pomocí daného jména a hesla přihlásit k LDAP serveru. V případě úspěšného přihlášení je klient autentizován k přístupu na Verse server, viz. obrázek 5. Toto obstarává funkce *vs_ldap_auth_user*. Ta má jako parametry *vContext*, uživatelské jméno a heslo. Funkce vrací identifikační číslo uživatele a v případě neúspěchu -1. Na následujícím výpisu je hlavička funkce.

```
int vs_ldap_auth_user(struct vContext *C, const char *username,
        const char *pass)
```



Obrázek 5: Ověření uživatele proti LDAPu

5.5 Obnovení seznamu uživatelů

Na Verse serveru bylo také implementováno načítání nově přidaných uživatelů za běhu serveru. Načítání nových uživatelů je implementováno pro načítání z LDAP serveru i ze souboru CSV. Spustí se, pokud je serveru poslán signál *SIGUSR1*.

O obsluhu signálů se stará funkce *vs_handle_signal* volající funkce obsluhující konkrétní signály. V případě signálu *SIGUSR1* volá nově implementovanou funkci *vs_reload_user_accounts*. Obě tyto funkce se nachází v souboru *vs_main.c*.

Funkce *vs_reload_user_accounts* volá podle zvolené autentizační metody konkrétní funkci načítající uživatelské účty. V případě LDAP autentizace je to funkce *vs_load_user_accounts_ldap_server* popsaná v kapitole 5.3.

5.6 Změna rozhraní pro odchytávání signálů

Aby načtení nových uživatelů po obdržení signálu fungovalo, bylo nutné změnit rozhraní pro odchytávání signálů. V původní implementaci Verse serveru je použito rozhraní *signal*. Toto rozhraní je však nevhodné pro vícevláknové aplikace.

Použito tedy bylo robustnější rozhraní *sigaction*. Odchytávání signálů je nastaveno funkcí *vs_config_signal_handling*. Nejprve jsou pomocí *sigaction* nastaveny odchytávané signály a funkce obsluhující tyto signály, tedy *vs_handle_signal*. Poté je vytvořeno nové vlákno, v němž jsou signály odchytávány. Ty jsou nakonec zablokovány pro ostatní vlákna pomocí *pthread_sigmask*.

Ve vláknu pro zpracování signálů běží funkce *vs_signal_thread*. Tato funkce pouze čeká na signál. V případě jeho obdržení je spuštěna funkce nastavená pomocí rozhraní *sigaction*.

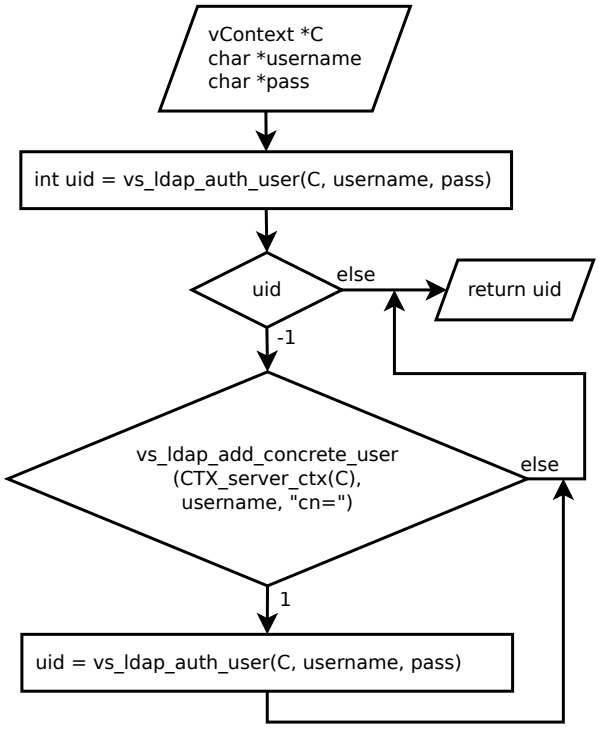
5.7 Načítání uživatele při pokusu o přihlášení

Z důvodu snížení zátěže Verse serveru při startu, byla přidána možnost načítání jednotlivých uživatelských účtů z LDAP serveru až při pokusu o přihlášení uživatele. Uživatelská jména takto vytvořených uživatelů jsou ukládána do CSV souboru. Při dalším spuštění Verse serveru jsou tak načítány pouze uživatelské účty, k nimž

se v minulosti někdo pokusil připojit.

K načtení těchto uživatelských účtů slouží funkce *vs_load_saved_ldap_users*. Prochází CSV soubor a podle uživatelského jména se pokusí vyhledat účet na LDAP serveru. Toto vyhledávání provádí funkce *vs_ldap_add_concrete_user*. Ta je podobná funkci *vs_load_user_accounts_ldap_server* popsané v kapitole 5.3 s tím rozdílem, že vyhledává pouze jednoho konkrétního uživatele.

Při samotném pokusu o přihlášení je volána funkce *vs_ldap_auth_and_add_user*. Ta se nejprve pokusí provést běžnou autentizaci pomocí funkce *vs_ldap_auth_user* popsané v kapitole 5.4. Pokud autentizace selže, pokusí se přidat uživatele pomocí funkce *vs_ldap_add_concrete_user*. Pokud je uživatel úspěšně přidán, znovu se pokusí o běžnou autentizaci, viz. obrázek 6. V případě selhání funkce *vs_ldap_add_concrete_user* a v případě zadání chybných přihlašovacích údajů funkce *vs_ldap_auth_and_add_user* vrací -1. Návratová proměnná *uid* je totiž nastavována pomocí funkce *vs_ldap_auth_user*, která ji v případě zadání chybných přihlašovacích údajů nastaví na -1.



Obrázek 6: Funkce vs_ldap_auth_and_add_user

6 Kerberizace Verse protokolu

V této části je popsána implementace ověřování uživatelů Verse serveru proti Kerberu a dále následná kerberizace knihovny Verse protokolu. Na rozdíl od implementace ověření proti LDAPu bylo potřeba změnit jak serverovou, tak klientskou část protokolu a též upravit vzorovou klientskou aplikaci.

6.1 Úprava datových struktur

Podobně jako v případě LDAPu bylo nutné upravit některé datové struktury. Především strukturu obsahující informace nutné pro přijímání a odesílání paketů. Je to struktura *IO_CTX*, která je definována v souboru *v_network.h*. Na následujícím výpisu jsou atributy, jež byly nově přidány této struktury.

```
typedef struct IO_CTX {
   unsigned short use_kerberos;
   krb5_keytab krb5_keytab;
   krb5_principal krb5_principal;
   krb5_context krb5_ctx;
   krb5_ccache  krb5_cc;
   krb5_auth_context krb5_auth_ctx;
   krb5_ticket  *krb5_ticket;
}
```

K již existujícím atributům byli přidány další atributy potřebné k autentizaci proti Kerberos serveru a následné kerberizaci verse protokolu. Byly to *use_kerberos*, určující zda je Kerberos aktuálně používán. Pokud ano, tento atribut má hodnotu 1. Atribut *krb5_keytab* odkazující na keytab soubor, v němž je uložen tajný klíč služby viz. 2.10.1. Dále *krb5_principal* obsahující informace o Kerberos uživateli, *krb5_ctx* což je kontext Kerberos knihovny. Atribut *krb5_cc* reprezentující lokální úložiště klíčů, *krb5_auth_ctx* kontext aktuálního Kerberos spojení. A nakonec *krb5_ticket* ve kterém jsou informace o tiketu.

Atribut *use_kerberos* byl také přidán jak do struktury *VC_CTX* reprezentující stav Verse klienta definované v souboru *vc_main.h*, tak do obdobné struktury sloužící serveru *VS_CTX* definované v *vs_main.h*.

6.2 Navázání spojení a autentizace uživatele

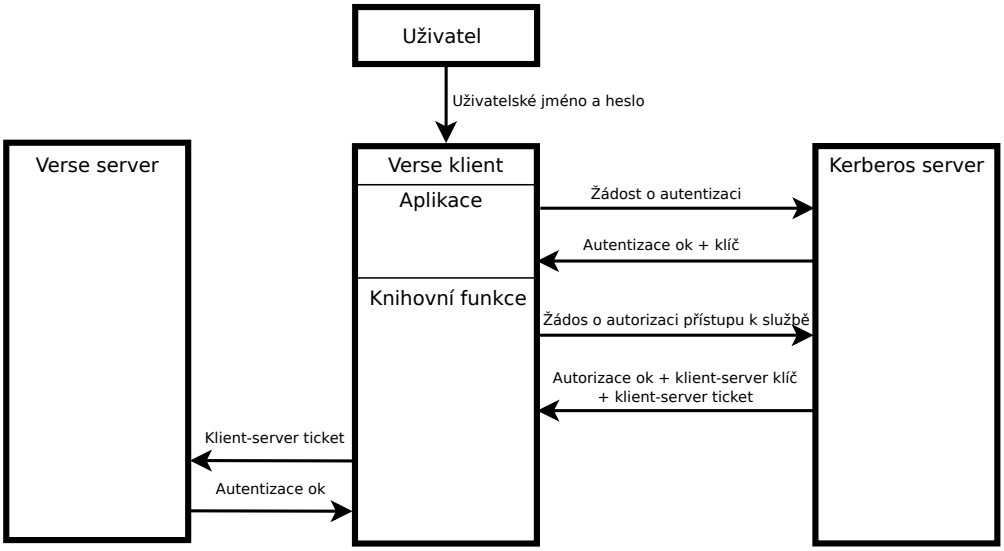
Při navázání spojení je nutné, aby byl na Verse serveru soubor keytab obsahující tajný klíč služby. Verse server pak naslouchá na daném portu a čeká na zprávu od klienta, jež obsahuje klient-server tiket viz. 2.10.1. Poté Verse server projde svůj seznam uživatelů a pokud nalezne uživatele autentizovaného Kerberem, odešle klientovi zprávu o úspěšné autentizaci, která je již šifrována pomocí klient-server klíče. Klient tak může komunikovat s Verse serverem. Pokud uživatel nebyl nalezen v seznamu uživatelů Verse serveru, ten odešle opět šifrovanou zprávu tentokrát o tom, že autentizace proběhla neúspěšně.

Verse klient obstarává veškerou komunikaci s Kerberos serverem. Přímo v rámci klientské aplikace je nutné vykonat první dvě fáze Kerberos autentizace: zadání hesla na němž je vykonána hašovací funkce a odeslání nešifrované žádosti o autentizaci Kerberos serveru. Server mu odešle zprávu obsahující i klíč pro další komunikaci viz. 2.10.1. Pokud bylo zadáno správné heslo, klient zprávu rozšifruje a obdržený klíč uloží do lokálního úložiště klíčů.

Další fáze již probíhá v rámci knihovních funkcí. Verse klient pomocí klíče obdrženého v předchozí fázi žádá o přístup ke službě (tedy k Verse serveru), poté obdrží klient-server klíč sloužící k šifrování komunikace s Verse serverem, a také klient-server tiket, jenž přepošle serveru, opět viz. 2.10.1. Poté čeká na zprávu o úspěšnosti autentizace na Verse serveru. Schéma autentizace zobrazuje obrázek 7.

6.3 Vyjednávání

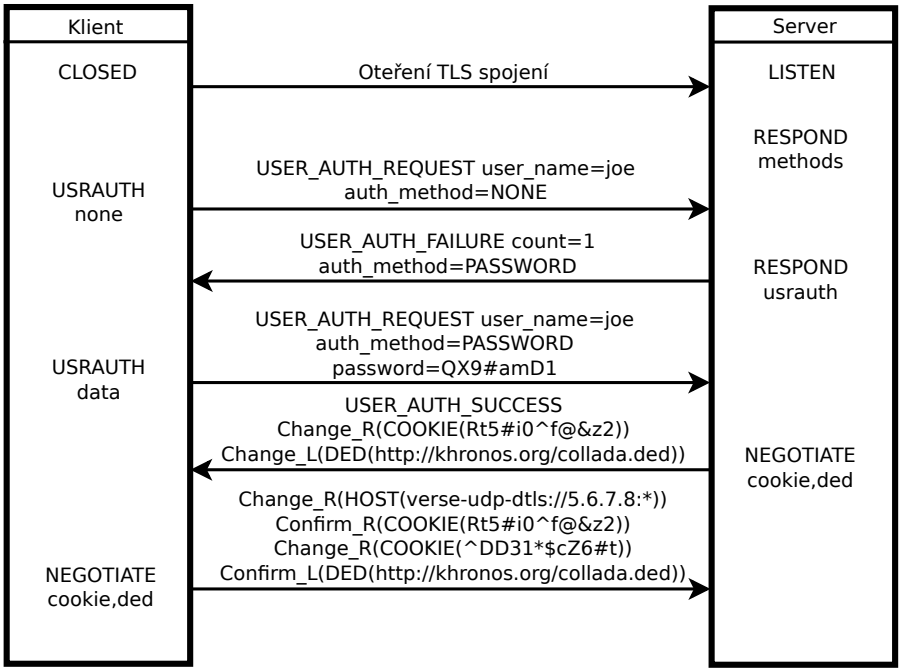
Byla upravena specifikace zahájení komunikace mezi Verse serverem a klientem. Nová specifikace je použita pouze v případě, je-li uživatel autentizován proti Kerberos serveru. V opačném případě je použita původní specifikace.



Obrázek 7: Autentizace uživatele Verse serveru proti Kerberos serveru

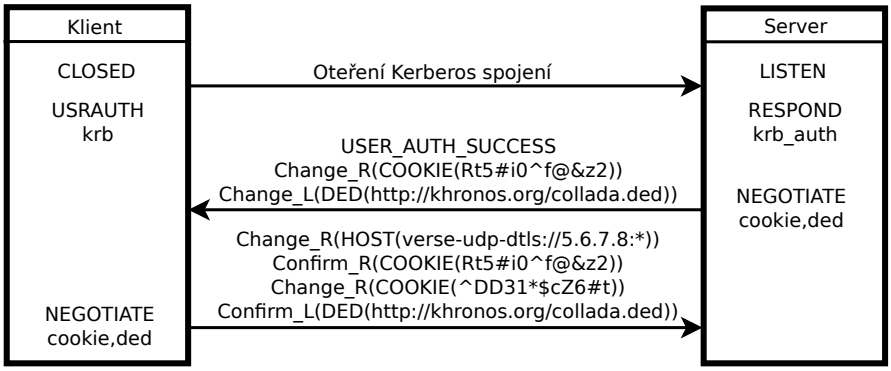
V původní specifikaci (schéma je na obrázku 8) server naslouchá na TCP portu a je ve stavu *LISTEN*, přičemž klient je ve stavu *CLOSED* a zahájí komunikaci otevřením TLS spojení. Server přejde do stavu *RESPOND methods* a dále naslouchá. Klient přejde do stavu *USRAUTH none* a odešle žádost o autentizaci se svým uživatelským jménem. Jako autentizační metoda je zvolena metoda *NONE*, tedy žádná. Server by tuto metodu neměl podporovat. Odešle tedy zprávu o tom, že se autentizace nezdařila, a s ní seznam podporovaných metod. V této specifikaci podporuje pouze metodu *PASSWORD*, tedy autentizaci za pomoci hesla a přejde do stavu *RESPOND usrauth*. Klient, poté co obdrží tuto zprávu, přejde do stavu *USRAUTH data*, a opět odešle žádost o autentizaci. Ta tentokrát obsahuje kromě jména i heslo a jako metoda je nastaveno *PASSWORD*. Server po té přejde do stavu *NEGOTIATE cookie, ded*, odešle zprávu o úspěšné autentizaci a připojí příkazy nastavující cookie a definici výměny dat (Data Exchange Definition, DED). Klient též přejde do stavu *NEGOTIATE cookie, ded* a odešle potvrzení, že obdržel příkazy. Pak se již dohaduje UDP spojení.

Ve specifikaci s Kerberos autentizací (schéma je na obrázku 9) je větší část autentizačního mechanismu řešena již při otevírání Kerberos spojení. Na začátku se server s klientem nachází ve stejných stavech *LISTEN* respektive *CLOSED*. Klient otevře Ker-



Obrázek 8: Původní specifikace vyjednávání mezi serverem a klientem

beros spojení a přejde do stavu *USRAUTH krb*. Server přejde do stavu *RESPOND krb_auth*, v něm ověří zda má uživatele se jménem získaným při navázání Kerberos spojení ve svém seznamu uživatelů. Po nalezení tohoto uživatele přejde do stavu *NEGOTIATE cookie, ded* a odešle zprávu o úspěšné autentizaci a příkazy nastavující cookie a DED. Klient opět přejde také do stavu *NEGOTIATE cookie, ded* a potvrdí přijetí příkazů. Následuje dohadování UDP spojení.



Obrázek 9: Nová specifikace vyjednávání mezi serverem a klientem při úspěšném ověření klienta

6.4 Inicializace struktury reprezentující TCP spojení

Při navazování TCP spojení je potřeba inicializovat strukturu reprezentující toto spojení a to jak na straně klienta, tak na straně serveru. Tato struktura se nazývá *VStreamConn* a je definována v souboru *v_connection.h*. Na straně klienta se toto děje ve funkci *vc_create_client_stream_conn* umístěné v souboru *vc_tcp_connect.c*. Vstupními argumenty jsou struktura reprezentující stav klienta *VC_CTX*, řetězec znaků reprezentující adresu serveru a řetězec znaků reprezentujících jméno služby nebo číslo portu. Výstupním parametrem je číslo chyby. Funkce vrací ukazatel na strukturu reprezentující TCP spojení, viz. následující výpis.

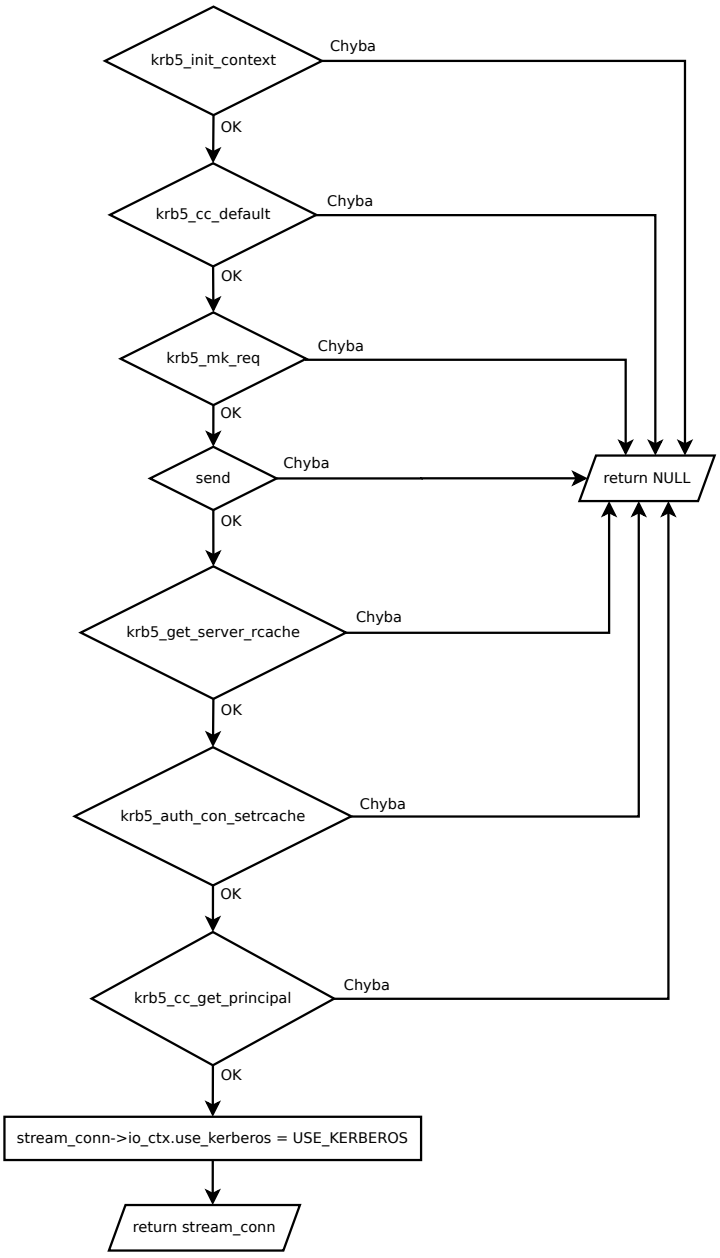
```
struct VStreamConn *vc_create_client_stream_conn(const struct VC_CTX *ctx,
        const char *node, const char *service, uint8 *error)
```

V rámci této funkce také klient otevírá Kerberos spojení. Nejprve je nutné zinicializovat kontext knihovny Kerberos. K tomu slouží funkce *krb5_init_context*. Dále je nastaveno úložiště klíčů pomocí funkce *krb5_cc_default*. Poté je pomocí funkce *krb5_mk_req* získán klient-server tiket a klient-server klíč. Klient-server tiket je pomocí funkce *send* odeslán Verse serveru. Pomocí funkcí *krb5_get_server_rcache* a *krb5_auth_con_setrcache* je nastavena záznamová mezipaměť, která slouží k zamezení duplicitních autentizací. Pro další účely běhu Verse klienta je třeba získat strukturu reprezentující Kerberos uživatele. To umožňuje funkce *krb5_cc_get_principal*. Nakonec je do vstupně výstupního kontextu nastaveno, že je používáno Kerberos spojení, viz. obrázek 10. Pokud v některém z kroků nastane chyba, je vypsáno hlášení o této chybě a funkce vrací *NULL*. V opačném případě funkce vrací ukazatel na právě vytvořenou strukturu reprezentující TCP spojení. Poté klient přejde do stavu *USRAUTH krb*.

Na straně serveru stejnému účelu slouží funkce *vs_init_stream_ctx*, ze které je volána funkce *vs_init_kerberos*. Obě tyto funkce mají jako jediný parametr strukturu reprezentující stav serveru a jsou umístěny v souboru *vs_tcp_connect.c*. Návratová hodnota je celé číslo. V případě že vše proběhne v pořádku vrací 1, jinak vrací -1. Na následujícím výpisu je hlavička funkce *vs_init_kerberos*.

```
int vs_init_kerberos(VS_CTX *vs_ctx)
```

V rámci této funkce je zinicializován kontext knihovny Kerberos funkcí *krb5_init_context*. Poté je vytvořena struktura reprezentující Kerberos uživatele, jež na Kerberos serveru představuje službu Verse. K tomu slouží funkce *krb5_sname_to_principal*.



Obrázek 10: Diagram vytváření Kerberos spojení na straně klienta

6.5 Ověření tiketu

Po inicializaci struktury Verse server obdrží a ověří klient-server tiket. To se děje ve funkci *vs_kerberos_auth*. Funkce je umístěna v souboru *vs_handshake.c* s parametrem *vContext*, což je struktura obsahující všechny nezbytné informace nutné ke běhu Verse serveru i Verse klienta. Výstupním parametrem je ukazatel na řetězec, do nějž se ukládá uživatelské jméno klienta, který poslal klient-server tiket. Hlavička výše zmíněné funkce je na následujícím výpisu.

```
int vs_kerberos_auth(struct vContext *C, char **u_name)
```

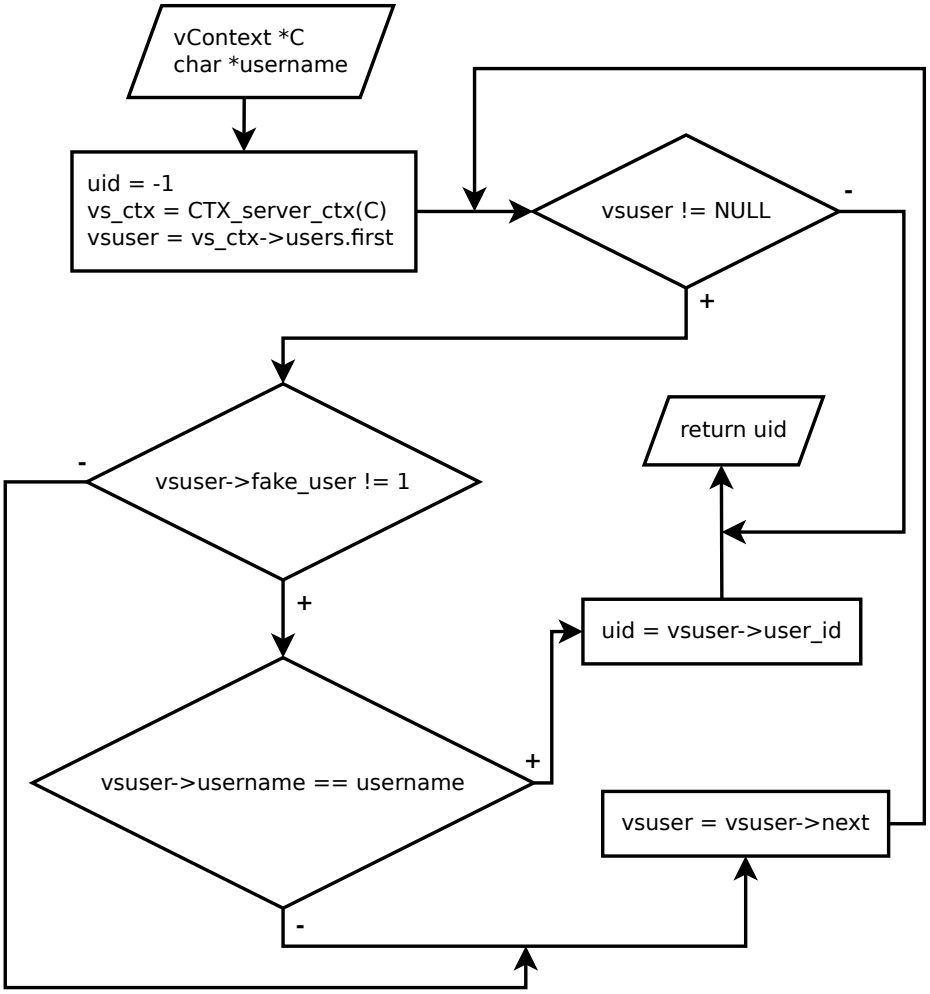
Nejprve je nutné tiket přijmout. K tomu je použita funkce *recvfrom*. Vlastní kontrola přijatého tiketu probíhá pomocí funkce *krb5_rd_req*. Nakonec se server pokusí získat uživatelské jméno pomocí funkce *krb5_unparse_name*. Pokud vše proběhne úspěšně funkce vrací 1, v opačném případě vrací 0.

6.6 Ověření existence uživatele v rámci Verse serveru

Po úspěšném navázání Kerberos spojení se Verse server přepne do stavu *RESPOND usrauth_krb* a pokusí se najít uživatele z klient-server tiketu ve svém seznamu uživatelů, aby zjistil jeho identifikační číslo. Tento seznam uživatelů byl vytvořen při startu Verse serveru načtením z LDAPu nebo CSV souboru. To obstarají funkce v souboru *vs_handshake.c*. Je to funkce *vs_RESPOND_krb_auth_loop*, která zpracovává některé příkazy přijaté v rámci vyjednávání. A také volá funkci *vs_krb_make_user*, jež slouží právě k nalezení daného uživatele mezi uživateli Verse serveru. Pokud se podaří uživatele nalézt, funkce *vs_RESPOND_krb_auth_loop* připraví k odeslání zprávu o úspěšné autentizaci obsahující identifikační číslo uživatele, toto číslo uloží do struktury *VSession* reprezentující aktuální relaci a vytvoří uživateli avatar. Jinak připraví zprávu o neúspěšné autentizaci.

```
static int vs_krb_make_user(struct vContext *C, const char *username)
```

Na výpisu výše je hlavička funkce *vs_krb_make_user*. Ta, jak již bylo zmíněno, slouží přímo k nalezení uživatele. V případě úspěchu vrací identifikační číslo uživatele, v případě neúspěchu vrací -1, viz obrázek 11.



Obrázek 11: Hledání uživatele v seznamu Verse uživatelů

Po úspěšné autentizaci uživatele se server přepne do stavu *NEGOTIATE cookie, ded* a odešle zprávu připravenou v rámci funkce *vs_RESPOND_krb_auth_loop*.

Klient tuto zprávu přijme a zpracuje funkcí *vc_USRAUTH_krb_loop* umístěnou v souboru *vc_tcp_connect.c*. Tato funkce zpracuje a připraví k odeslání vyjednávací příkazy. Především však v případě obdržení zprávy o úspěšné autentizaci nastaví ve struktuře reprezentující aktuální relaci identifikační číslo uživatele a jeho avataru. Klient pak přejde do stavu *NEGOTIATE cookie, ded*. Mezi Verse klientem a Verse serverem pak proběhne ještě několik vyjednávacích zpráv, které dohadují parametry přenosu dat po UDP a TCP spojení je ukončeno.

6.7 Navázání UDP spojení

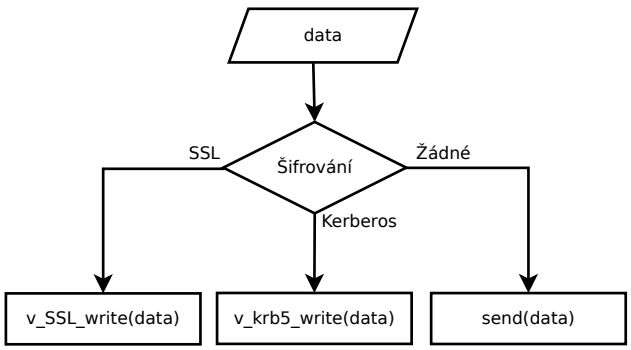
Vlastní přenos dat probíhá přes UDP. Při otevírání tohoto spojení nebylo nutné udělat mnoho změn. Všechny atributy nutné pro kerberizaci přenosu byly získány již během navazování TCP spojení. Bylo pouze nutné tyto atributy překopírovat ze struktury reprezentující TCP spojení (viz. 6.4), do struktury ukládající informace o UDP spojení. Ta se nazývá *VDgramConn* a je definována v souboru *v_connection.h*.

Na straně klienta se o inicializaci této struktury stará funkce *vc_create_client_dgram_conn* implementovaná v souboru *vc_udp_connect.c*. Na straně serveru to je funkce *vs_init_dgram_ctx* ze souboru *vs_udp_connect*. V těchto funkcích bylo přidáno právě ono zkopírování atributů ze struktury *VStreamConn*.

6.8 Kerberizace přenosu přes TCP

Aby mohl být Verse protokol kerberizován, bylo nutné upravit funkce pro odesílání a příjem dat tak, aby byla tato data šifrována klient-server klíčem. Tyto funkce jsou implementovány v souboru *v_network.c*. Jsou to funkce *v_tcp_read* pro příjem dat přes TCP a *v_tcp_write* pro odesílání dat přes TCP.

Ve funkci *v_tcp_write* se rozhoduje zda mají být data šifrována, případně jakým způsobem. V případě nešifrovaného přenosu tato funkce data odešle. Naopak při použití šifrování je volána funkce, jež se postará o zašifrování a přenos dat za použití dané šifrovací technologie, viz. obrázek 12. V případě použití Kerberos šifrování je volána funkce *v_krb5_write*.



Obrázek 12: Funkce v_tcp_write

Tato funkce má jako vstupní parametr vstupně výstupní kontext a jako výstupní parametr číslo chyby. Vrací počet přenesených bytů. V případě chyby vrací 0. Hlavička funkce je na následujícím výpisu.

```
int v_krb5_write(struct IO_CTX *io_ctx, krb5_error_code *error_num)
```

Data jsou šifrována pomocí funkce *krb5_mk_priv*. Pro tuto funkci je důležité, aby byl správně nastaven *krb5_auth_context* (v něm je totiž uložen klient-server klíč). Pomocí tohoto klíče jsou data šifrována. Klient zmíněný klíč získá díky funkci *krb5_mk_req*, kterou volal v rámci inicializace struktury reprezentující TCP spojení, viz. kapitola 6.4. Server jej získá pomocí funkce *krb5_rd_req*, jíž volal v rámci ověření tiketu popsaném v kapitole 6.5. Poté jsou data odeslána pomocí funkce *send*.

Podobným způsobem jako funkce pro odeslání dat po TCP funguje i funkce pro jejich příjem *v_tcp_read*. Data jsou zpracována a předána v závislosti na druhu šifrování. Funkce funguje podobně jako ta na obrázku 12. Pouze místo dat k odeslání je předán ukazatel na místo v paměti, kam mají být zapsána přijatá data. Místo odesílacích a šifrovacích funkcí, jsou volány funkce přijímací a dešifrovací.

V případě užití Kerberos šifrování je to funkce *v_krb5_read*. Ta má stejné parametry jako funkce *v_krb5_write* a vrací velikost dešifrované zprávy. Hlavička funkce je na následujícím výpisu.

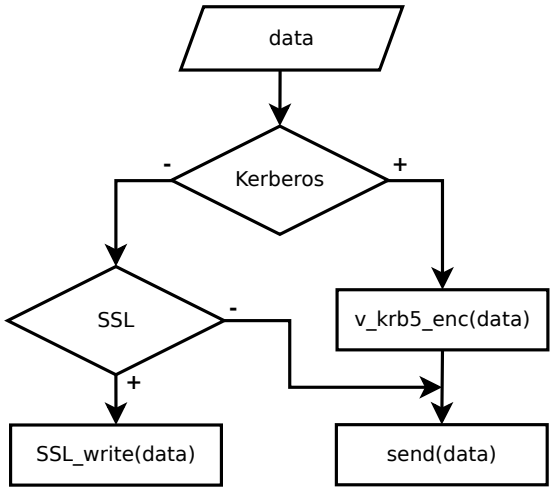
```
int v_krb5_read(struct IO_CTX *io_ctx, krb5_error_code *error_num)
```

Data jsou nejprve přijata pomocí funkce *recvfrom*. Pak jsou předána funkci *krb5_rd_priv*. Ta je opět za pomoci klient-server klíče z *krb5_auth_context* dešifruje. Dešifrovaná data jsou uložena na požadované místo v paměti.

6.9 Kerberizace přenosu přes UDP

Kerberizace protokolu Verse pokračovala úpravou funkcí starajících se o příjem a odesílání dat přes UDP. Jsou to funkce *v_send_packet*, pro odesílání, a *v_receive_packet*, pro příjem dat. Funkce se taktéž nachází v souboru *v_network.c*.

Při odesílání je nejprve posouzeno, zdali má být použito Kerberos šifrování. Pokud ano, je zavolána funkce *v_krb5_enc*, jež data zašifruje. Ta jsou pak odeslána v rámci funkce *v_send_packet*, jež je vyobrazena na obrázku 13.



Obrázek 13: Funkce v_send_packet

Funkce *v_krb5_enc* je velmi podobná funkci *v_krb5_write*. S tím rozdílem, že data jsou pouze zašifrována pomocí *krb5_mk_priv* a nejsou v rámci této funkce odeslána, viz. 6.8.

Data jsou přijímána ve funkci *v_receive_packet*. Ta pro jejich dešifrování volá funkci *v_krb5_dec* opět velmi podobnou funkci popsané v kapitole 6.8 tentokrát *v_krb5_read*. Znovu s rozdílem, že ta pouze dešifruje data, nikoliv přijímá.

6.10 Úprava vzorové aplikace

Nakonec bylo nutné upravit ukázkovou klientskou aplikaci Verse protokolu. Zdrojový kód vzorové aplikace se nachází v souboru *verse_client.c*. V rámci aplikace je zvoleno, zdali bude použita Kerberos autentizace. K tomu byl zvolen přepínač *-k*. Do nápovědy, vypisované funkcí *print_help* byla přidána informace o tomto přepínači.

Verse API bylo rozšířeno o knihovní funkci *vrs_set_krb5_use*. Tato funkce je volána při použití přepínače *-k*. Její pomocí je nastaveno použití Kerberos autentizace v klientském kontextu knihovny Verse. Tato funkce byla přidána do souboru *verse.c*.

Dále je, při použití nově přidaného přepínače, volána funkce *init_krb5_cc*. Ta se stará o autentizaci uživatele na Kerberos serveru. Uživatel je vyzván k zadání uživatelského jména a hesla. Ze zadaného uživatelského jména je pomocí funkce *krb5_parse_name* vytvořena struktura obsahující informace o Kerberos uživateli, *krb5_principal*. Tato struktura a zadané heslo jsou vstupními parametry funkce *krb5_get_init_creds_password*. Tato funkce získá od Kerberos serveru klient-TGS klíč a TGT, viz. kapitola 2.10.1. Vše ostatní již obstarávají výše popsané funkce v rámci knihovny Verse.

7 Testování

Testování probíhalo lokálně na systému Fedora 19. Pro účely testování byl v souboru */etc/hosts* nastaven statický překlad domény *zdn.local*, viz. následující výpis.

```
127.0.0.1   zdn.local kerberos.zdn.local
::1         zdn.local kerberos.zdn.local
```

Při testování byla snaha postihnout všechny možné stavy nově implementovaných funkcí. Bylo tedy ověřeno načtení uživatelských účtů z LDAP serveru, znovu načtení účtů, ověření uživatele proti LDAPu a proti Kerberos serveru.

7.1 Načtení uživatelů z LDAPu

Pro načítaní uživatelů z LDAPu při startu Verse serveru byl vytvořen konfigurační soubor nazvaný *ldap_s.ini*. Podoba souboru je na následujícím výpisu.

```
[Users]
Method = ldap ;
Version = 3 ;
LoadOnStart = yes ;
Hostname = "ldap://zdn.local" ;
UserDN = "cn=root,dc=zdn,dc=local" ;
Pass = "heslo" ;
Base = "dc=zdn,dc=local" ;
[Security]
Certificate = "./pki/certificate.pem" ;
PrivateKey = "./pki/private.key.pem" ;
```

V konfiguračním souboru bylo nastaveno správné DN a heslo pro přihlášení k LDAP serveru. Verse server se tak po načtení konfiguračního souboru úspěšně přihlásí k LDAP serveru, dotáže se na seznam uživatelů a ty přidá do svého seznamu. Spuštění serveru a úspěšné načtení uživatelských účtů je na následujícím výpisu.

```
[root@pandell build]# ./bin/verse_server -d debug -c ./config/ldap_s.ini
DEBUG: user_auth_method: ldap
DEBUG: LDAP version 3
DEBUG: Users will be loaded on startup.
DEBUG: LDAP server hostname: ldap://zdn.local
DEBUG: LDAP user DN: cn=root,dc=zdn,dc=local
DEBUG: LDAP search base: dc=zdn,dc=local
DEBUG: certificate_file_name: ./pki/certificate.pem
DEBUG: private_key: ./pki/private.key.pem
DEBUG: Added: username: usr@ZDN.LOCAL, ID: 2000, realname: Test User
DEBUG: Added: username: usr1@ZDN.LOCAL, ID: 2001, realname: Test User1
DEBUG: Added: username: usr2@ZDN.LOCAL, ID: 2002, realname: Test User2
DEBUG: Added: username: usr3@ZDN.LOCAL, ID: 2003, realname: Test User3
DEBUG: Added: username: usr4@ZDN.LOCAL, ID: 2004, realname: Test User4
DEBUG: 5 user account loaded from LDAP server: ldap://zdn.local,
search base: dc=zdn,dc=local
```

Při zadání chybných přihlašovacích údajů Verse server opět úspěšně načte konfigurační soubor, avšak nepodaří se mu přihlásit k LDAPu. Načtení uživatelských účtů tak selže a Verse server ukončí činnost, viz. následující výpis.

```
[root@pandell build]# ./bin/verse_server -d debug -c ./config/ldap_s.ini
DEBUG: user_auth_method: ldap
DEBUG: LDAP version 3
DEBUG: Users will be loaded on startup.
DEBUG: LDAP server hostname: ldap://zdn.local
DEBUG: LDAP user DN: cn=root,dc=zdn,dc=local
DEBUG: LDAP search base: dc=zdn,dc=local
DEBUG: certificate_file_name: ./pki/certificate.pem
DEBUG: private_key: ./pki/private.key.pem
DEBUG: ldap_sasl_bind_s: 49: Invalid credentials
ERROR: vs_load_user_accounts(): failed
```

7.2 Načtení nově přidaných uživatelů

Pokud je Verse serveru poslán signál *SIGUSR1*, pokusí se načíst nově přidané uživatelské účty. Serveru byl tento signál poslán následujícím příkazem:

```
$ sudo kill -SIGUSR1 ‘pidof verse_server‘
```

Spuštěný Verse server tento signál úspěšně odchytil a pokusil se je přidat k již načteným uživatelům z bodu 7.1. Sever obdržel v seznamu i již existující uživatele. Tyto účty nebyly přidány. Účty přidané na LDAP server až při běhu Verse serveru však byly přidány úspěšně, viz. následující výpis.

```
WARNING: User usr@ZDN.LOCAL could not be added to list of user, because
user usr@ZDN.LOCAL has same user ID: 2000
WARNING: User usr1@ZDN.LOCAL could not be added to list of user, because
user usr1@ZDN.LOCAL has same user ID: 2001
WARNING: User usr2@ZDN.LOCAL could not be added to list of user, because
user usr2@ZDN.LOCAL has same user ID: 2002
WARNING: User usr3@ZDN.LOCAL could not be added to list of user, because
user usr3@ZDN.LOCAL has same user ID: 2003
WARNING: User usr4@ZDN.LOCAL could not be added to list of user, because
user usr4@ZDN.LOCAL has same user ID: 2004
DEBUG: Added: username: usr5@ZDN.LOCAL, ID: 2005, realname: Test User5
DEBUG: Added: username: usr6@ZDN.LOCAL, ID: 2006, realname: Test User6
DEBUG: Added: username: usr7@ZDN.LOCAL, ID: 2007, realname: Test User7
DEBUG: Added: username: usr8@ZDN.LOCAL, ID: 2008, realname: Test User8
DEBUG: Added: username: usr9@ZDN.LOCAL, ID: 2009, realname: Test User9
DEBUG: 5 user account loaded from LDAP server: ldap://zdn.local, search
base: dc=zdn,dc=local
```

7.3 Ověření uživatele proti LDAPu

Testováno bylo taktéž samotné ověření uživatele. Zadány byly postupně správné přihlašovací údaje, chybné uživatelské jméno a chybné heslo.

Klientská aplikace nejprve požádá o uživatelské jméno a posléze o heslo. V případě správně zadaných údajů je přihlášení úspěšné, viz následující výpis.

```
[ZDN@pandell build]$ ./bin/verse_client zdn.local
cb_receive_user_authenticate() username: (null), auth_methods_count: 0,
methods:
Username: usr@ZDN.LOCAL
cb_receive_user_authenticate() username: usr@ZDN.LOCAL, auth_methods_count:
1, methods: 2,
Password:
cb_receive_connect_accept() session_id: 0, user_id: 2000, avatar_id: 65540
```

Pokud je uživatelské jméno či heslo zadáno chybně, přihlášení se nezdaří. V obou těchto případech dostaneme výsledek z následujícího výpisu.

```
[ZDN@pandell build]$ ./bin/verse_client zdn.local
cb_receive_user_authenticate() username: (null), auth_methods_count: 0,
methods:
Username: usr
cb_receive_user_authenticate() username: usr, auth_methods_count: 1,
methods: 2,
Password:
cb_receive_connect_terminate() session_id: 0, error_num: 4
```

7.4 Načtení LDAP uživatele při přihlášení

V další fázi testování bylo otestováno načítání uživatelů při přihlášení. Byl vytvořen konfigurační soubor Verse serveru *ldap_l.ini*, podobný souboru *ldap_s.ini* z 7.1. Pouze v něm jeden řádek v sekci *[Users]* přibyl a jeden byl změněn, viz. následující výpis.

```
LoadOnStart = no ;
File = "cache.csv" ;
```

První řádek určuje, že uživatelé nebudou načteni již při startu serveru. Druhý nastavuje umístění souboru, pro uložení již vytvořených uživatelských účtů.

```
[root@pandell build]# ./bin/verse_server -d debug -c ./config/ldap_l.ini
DEBUG: user_auth_method: ldap
DEBUG: LDAP version 3
DEBUG: Users will be loaded at login.
DEBUG: File for saving user accounts: /home/ZDN/verse/cache.csv
DEBUG: LDAP server hostname: ldap://localhost
DEBUG: LDAP user DN: cn=root,dc=zdn,dc=local
DEBUG: LDAP search base: dc=zdn,dc=local
DEBUG: certificate_file_name: ./pki/certificate.pem
DEBUG: private_key: ./pki/private.key.pem
DEBUG: Loading cached user accounts.
```

Na předcházejícím výpisu je start Verse serveru, při neexistujícím nebo prázdném souboru pro ukládání již vytvořených uživatelských účtů. Žádní uživatelé tedy aktuálně na Verse serveru nejsou vytvořeni. Při pokusu klienta o přihlášení se Verse server pokusí vytvořit uživatelský účet, viz. následující výpis. Na straně klienta je výpis shodný jako při úspěšném přihlášení z bodu 7.3.

```
DEBUG: Receive message: Socket: 4, [::1]:57197 Ver: 1, Len: 25
     USER_AUTH_REQUEST, Username: usr@ZDN.LOCAL, Type: Password: ***
DEBUG: Added: username: usr@ZDN.LOCAL, ID: 2000, realname: Test User
DEBUG: Added to cache: usr@ZDN.LOCAL,2000
DEBUG: 1 user account loaded from LDAP server: ldap://localhost, search
base: dc=zdn,dc=local
```

Při opětovném startu Verse serveru je již ve vyrovnávacím souboru uložen jeden uživatel. Tento uživatel je při dalším startu Verse serveru přidán do jeho seznamu uživatelů, viz. následující výpis.

```
DEBUG: Loading cached user accounts.
DEBUG: Added: username: usr@ZDN.LOCAL, ID: 2000, realname: Test User
DEBUG: 1 user account loaded from LDAP server: ldap://localhost, search
base: dc=zdn,dc=local
```

7.5 Ověření proti Kerberos serveru

V závěrečné fázi testování bylo otestováno ověření Verse serveru proti Kerberos serveru. K tomu byl vytvořen konfigurační soubor *krb.ini* z následujícího výpisu.

```
[Users]
Method = file ;
FileType = csv ;
File = "./config/users.csv" ;
[Security]
UseKerberos = yes ;
```

Testování opět proběhlo zadáním správných i chybných přihlašovacích údajů na straně klienta. Nejprve byl spuštěn Verse server, viz. výpis.

```
[root@pandell build]# ./bin/verse_server -d debug -c ./config/krb.ini
DEBUG: user_auth_method: file
DEBUG: file_type: csv
csv_file_name: ./config/users.csv
DEBUG: Kerberos will be used
DEBUG: Added: username: usr@ZDN.LOCAL, ID: 2000, realname: Test User
DEBUG: Added: username: usr1@ZDN.LOCAL, ID: 2001, realname: Test User1
```

Při zadání správných přihlašovacích údajů v klientské aplikaci, se klient pokusí navázat spojení s Verse serverem. V případě, že daný uživatel na Verse serveru existuje, proběhne vše v pořádku, viz následující výpis.

```
[ZDN@pandell build]$ ./bin/verse_client zdn.local -k
Username: usr
Password:
cb_receive_connect_accept() session_id: 0, user_id: 2000, avatar_id: 65537
```

V případě zadání chybného hesla klient neúspěšně dešifruje zprávu od Kerberos server. Nemůže se tak připojit k Verse serveru a ukončí svoji činnost, viz následující výpis.

```
[ZDN@pandell build]$ ./bin/verse_client zdn.local -k
Username: usr
Password:
krb5_get_init_creds_password: -1765328353: Decrypt integrity check failed
```

Pokud bylo zadáno chybné uživatelské jméno, Kerberos server jej nenajde ve své databázi a Verse klientovi o tom odešle zprávu. Verse klient pak opět končí svoji činnost, viz výpis níže.

```
[ZDN@pandell build]$ ./bin/verse_client zdn.local -k
Username: uss
Password:
krb5_get_init_creds_password: -1765328378: Client not found in Kerberos
database
```

8 Závěr

Tato diplomová práce popisuje rozšíření Verse protokolu o autentizaci proti LDAPu a jeho kerberizaci. Součástí práce je stručná rešerše o knihovnách implementujících LDAP nebo Kerberos. Pro samotnou implementaci pak byly vybrány knihovny OpenLDAP a MIT Kerberos.

Do knihovny Verse protokolu bylo implementováno ověřování uživatelských účtů proti LDAPu a Kerberos serveru, načítání seznamu uživatelů z LDAP serveru a šifrování komunikace mezi Verse klientem a Verse serverem pomocí Keberos Klient-server klíče. Dále bylo změněno rozhraní pro odchytávání signálů a implementováno načtení nových uživatelských, pokud server obdrží signál *SIGUSR1*.

Možnost LDAP autentizace byla do Verse protokolu přidána z důvodu snadnější implementace do již existujícího IT prostředí. LDAP autentizace je totiž běžně používaným mechanismem pro jednotlivé služby (např. elektronická pošta, přihlášení k síti atd.) v podnikových nebo univerzitních sítích.

Ověření uživatelských účtů proti Kerberos serveru a kerberizace Verse protokolu bylo realizováno kvůli zvýšení bezpečnosti Verse protokolu. Kerberizace slouží jako alternativa k TLS šifrování. Hlavní výhodou Kerberos protokolu jsou hesla uložená na jednom důvěryhodném serveru. Jednotlivé služby nepracují přímo s heslem ale pouze s tikety. Nevýhodami Kerberos protokolu jsou jeho poměrně malá rozšířenost a nutnost přesné časové synchronizace všech účastníků komunikace.

Během testování nových metod autentizace byla zkoušena různá nastavení Verse serveru pro LDAP a Kerberos autentizaci. Na straně Verse klienta pak byly postupně zadávány správné a chybné přihlašovací údaje. Všechny tyto testy skončily úspěšně. Jako vzorová aplikace pro testování byla využita již existující klientská aplikace. Ta byla rozšířena o funkci navazující spojení s Kerberos serverem.

Byla také sepsána uživatelská dokumentace popisující instalaci a nastavení Verse serveru a Verse klienta za použití nových autentizačních metod. Tato dokumentace se nachází v příloze práce.

V budoucnu bylo vhodné rozšířit Verse protokol o další autentizační metody. Mezi

tyto metody by mohly patřit např. autentizace pomocí klientského certifikátu a autentizace přes Pluggable Authentication Modules (PAM). PAM je v současné době standardním modulem Unixových systémů. Toto rozšíření by ještě zvýšilo komplexnost protokolu Verse. Dále by bylo vhodné zvýšit pokrytí kódu knihovny protokolu Verse unit testy, to umožní snáze a rychleji testovat případné změny v knihovně Verse protokolu.

Reference

[1] Shafranovich, Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files
RFC 4180, IETF, October 2005. http://www.ietf.org/rfc/rfc4180.txt.

[2] Dierks, T. and Allen, C. The TLS Protocol Version 1.0 RFC 2246, IETF, January 1999. http://www.ietf.org/rfc/rfc2246.txt, Obsoleted by RFC 4346, Updated by RFC 3546, RFC 5746, RFC 6176.

[3] HNÍDEK, Jiří. Síťový protokol pro grafické aplikace. Liberec, 2011. Dizertační práce. Technická univerzita v Liberci.

[4] Barker, P. and KILLE, S. The COSINE and Internet X.500 Schema RFC 1274, IETF, November 1991. http://www.ietf.org/rfc/rfc1274.txt, Obsoleted by RFC 4524.

[5] WAHL, M., HOWES, T. and KILLE, S. Lightweight Directory Access Protocol (v3) RFC 2251, IETF, December 1997. http://www.ietf.org/rfc/rfc2251.txt, Obsoleted by RFC 4510, 4511, 4512, 4513.

[6] Myers, J. Simple Authentication and Security Layer (SASL) RFC 2222, IETF, October 1997. http://www.ietf.org/rfc/rfc2222.txt, Obsoleted by RFC 4422, RFC 4752, Updated by RFC 2444.

[7] Good, G. The LDAP Data Interchange Format (LDIF) - Technical Specification RFC 2849, IETF, June 2000. http://www.ietf.org/rfc/rfc2849.txt.

[8] Kohl, J. and Neuman, C. The Kerberos Network Authentication Service (V5) RFC 1510, IETF, September 1993. http://www.ietf.org/rfc/rfc1510.txt, Obsoleted by RFC 4120.

[9] Raeburn, K. Advanced Encryption Standard (AES) Encryption for Kerberos 5 RFC 3962, IETF, February 2005. http://www.rfc-editor.org/rfc/rfc3962.txt.

[10] Red Hat, Inc. 2012. http://www.redhat.com/.

[11] GNU GENERAL PUBLIC LICENSE Version 3, Free Software Foundation, Inc. June 2007. http://www.gnu.org/licenses/gpl-3.0.txt.

[12] The Apache Software Foundation, 2012. http://www.apache.org/.

[13] Apache License, Version 2.0, The Apache Software Foundation, January 2004. http://www.apache.org/licenses/LICENSE-2.0.

[14] COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0. http://oss.oracle.com/licenses/CDDL.

[15] The OpenLDAP Public License Version 2.8, OpenLDAP Foundation, August 2003. http://www.openldap.org/software/release/license.html.

[16] OpenLDAP Foundation, 2012. http://www.openldap.org/foundation/.

[17] MIT Kerberos License information. http://web.mit.edu/kerberos/krb5-current/doc/mitK5license.html.

[18] Appendix B Copyrights and Licenses. http://www.h5l.org/manual/heimdal-1-5-branch/info/heimdal/Copyrights-and-Licenses.html.

[19] SpringSource, 2012. http://www.springsource.org/.

[20] VMware, Inc. 2012. http://www.vmware.com/cz/.

[21] Mattias Arthursson, Ulrik Sandberg, Eric Dalquist and Keith Barlow Spring LDAP - Reference Documentation. http://static.springsource.org/spring-ldap/docs/1.3.x/reference/html/.

[22] SCM Repositories - python-ldap. http://python-ldap.cvs.sourceforge.net/viewvc/python-ldap/python-ldap/LICENCE.

[23] Perl LDAP. http://ldap.perl.org/.

Příloha - Uživatelská dokumentace

Rozšíření Verse protokolu o podporu autentizace proti LDAPu a Kerberos serveru bylo vyvíjeno na systému Fedora 19.

Požadavky

- GCC http://gcc.gnu.org/ nebo Clang http://clang.llvm.org/
- CMake http://www.cmake.org/
- OpenSSL http://www.openssl.org/
- IniParser http://ndevilla.free.fr/iniparser/
- Check (volitelný) http://check.sourceforge.net/
- Spin (volitelný) http://spinroot.com/
- Python3 (volitelný) http://www.python.org/
- WSLay (volitelný) http://wslay.sourceforge.net/
- OpenLDAP http://www.openldap.org/
- Kerberos V5 http://web.mit.edu/kerberos/

Kompilace

Pro zkompilování Verse serveru, knihovny Verse a ukázkové klientské aplikace otevřete terminál a jděte do kořenového adresáře zdrojových souborů Verse. Zde zadejte:

```
$ mkdir ./build
$ cd ./build
$ cmake ../
$ make
$ sudo make install
```

Pokud používáte Clang, je nutné zadat:

```
$ export CC=/usr/bin/clang
$ export CXX=/usr/bin/clang++
$ mkdir ./build
$ cd ./build
$ cmake ../
$ make
$ sudo make install
```

Adresáře

- ./build je cílový adresář pro zkompilované binární soubory
- ./config je adresář s ukázkovým .csv souborem
- ./doc obsahuje soubor doxyfile pro generovanou dokumentaci doxygen
- ./example obsahuje zdrojové kódy ukázkové aplikace
- ./include obsahuje všechny .h soubory
- ./pki obsahuje příklad certifikátu a privátního klíče
- ./python obsahuje zdrojový kód pro Python modul implementovaný v C
- ./promela obsahuje Promela zdrojový kód použitý pro ověření protokolu
- ./src obsahuje zdrojové kódy

  ./api obsahuje zdrojové API

  ./client obsahuje zdrojové kódy specifické pro Verse klienty

  ./common obsahuje zdrojové kódy sdílené pro Verse server a Verse klienty

  ./server obsahuje zdrojové kódy specifické pro Verse server
- ./unittests obsahuje zdrojové kódy unit testů

Nastavení serveru

Vzorový konfigurační soubor serveru je v adresáři *./config* a jmenuje se *server.ini* Pro úspěšné použití LDAP autentizace je nutné nastavit v sekci *[Users]*:

```
[Users]
Method = ldap ;
Version = 3 ;
LoadOnStart = <yes/no> ;
Hostname = "<adresa ldap serveru>" ;
UserDN = "<DN uživatele>" ;
Pass = "<heslo uživatele>" ;
Base = "<umístění Verse uživatelů na ldap serveru>" ;
```

LoadOnStart rozhoduje zdali budou uživatelé načteni již při startu serveru. *UserDN* a *Pass* určují přihlašovací údaje Verse serveru k LDAP serveru.

Naopak pro použití Kerberizovaného Verse serveru je nutné nastavit v sekci *[Security]*:

```
[Security]
UseKerberos = yes ;
```

Spouštění

Server se spouští z adresáře *./build*:

```
$ ./bin/verse_server -d debug -c <cesta ke konfiguračnímu souboru>
```

Ukázková klientská aplikace je spouštěna ze stejného adresáře:

```
$ ./bin/verse_client <adresa Verse serveru> -d debug
```

Pro použití Kerbera:

```
$ ./bin/verse_client <adresa Verse serveru> -d debug -k
```

Spouštění LDAPu a Kerberos serveru

Uživatelské účty na LDAP serveru musejí být odvozeny od objektové třídy *inetOrgPerson*. Přihlašovací jméno musí být uloženo v atributu *uid*, identifikační číslo uživatele v atributu *employeeNumber* a heslo v atributu *userPassword*. Dále by měli být vyplněny atributy pro jméno a příjmení (*givenName* a *sn*). Pro spuštění LDAP serveru je nutné spustit službu *slapd*:

```
$ sudo service slapd start
```

Pro spuštění Kerberos serveru je nutné spustit službu *krb5kdc* a aby bylo možné Kerberos server konfigurovat, přidávat a měnit uživatelské účty spusťte službu *kadmin*:

```
$ sudo service krb5kdc start
$ sudo service kadmin start
```