ISA Projekt

LDAP Server

dokumentace

Vytvořil: Vorobec Valentyn (xvorob02)

Obsah

Obsah **2**
Teorie **3**
Hierarchická Struktura 3
Základní Principy 3
Příklady Použití LDAP 3
Basic Encoding Rules 4
Kód **5**
TCP spojení 5
Zpracování příchozí zprávy 5
Pomocné funkce 5
Struktury 5
Nedostatky mého programu 5
Testování **6**
Zdroje **13**

Teorie

LDAP, neboli Lightweight Directory Access Protocol, je protokol navržený pro přístup a správu distribuovaných adresářových informací. Jedná se o standardizovaný protokol aplikace s otevřeným protokolem, který umožňuje klientům dotazovat a aktualizovat data v hierarchickém adresáři. LDAP je široce využíván pro správu informací o uživatelích, počítačích, síťových zařízeních a dalších objektech v rámci organizace.

Protokol LDAP je založen na doporučení X.500, které bylo vyvinuto ve světě ISO/OSI, ale do praxe se ne zcela prosadilo, zejména pro svou „velikost“ a následnou „těžkopádnost“. [1]

Hierarchická Struktura

1. Data v LDAP jsou organizována do stromové struktury nazývané adresářový strom.
2. Každý uzel tohoto stromu je identifikován jedinečným názvem - Distinguished Name (DN).
3. DN reprezentuje cestu od kořene stromu k danému uzlu.

Základní Principy

1. Databázový Model
    a. Data v LDAP jsou uložena ve formě záznamů, které obsahují atributy a jejich hodnoty.
2. LDAP Server
    a. Poskytuje služby na čtení a zápis do adresáře.
    b. Spravuje hierarchii dat a odpovídá na dotazy klientů.
3. LDAP Klienti
    a. Komunikují s LDAP serverem pomocí LDAP protokolu.
    b. Mohou provádět různé operace, jako je vyhledávání, přidávání, mazání a modifikace dat.
4. LDAP Distinguished Name (DN)
    a. Unikátní identifikátor uzlu v adresářovém stromu.
    b. Skládá se z názvů jednotlivých uzlů od kořene stromu.

Příklady Použití LDAP

1. Správa Uživatelských Účtů
    a. Uchovávání informací o uživatelích, včetně jejich jména, hesla a přístupových práv.
2. Správa Síťových Zdrojů
    a. Evidence síťových zařízení, jako jsou servery, tiskárny a routery.
3. Centralizovaná Správa Adresářů
    a. Uložení kontaktních informací o zaměstnancích, odděleních a dalších organizačních jednotkách.
4. Implementace SSO (Single Sign-On)

a. Používání LDAP pro autentizaci uživatelů a poskytování jednotného přihlášení.

V projektu se pracovalo s LDAP verze 2 a operace byla pouze čtení. Server běží na portu 389 a pro komunikaci se serverem se zadává příkaz: “ldapsearch”, viz. příklady výpisů. Server tak odpovídá pouze s parametry uid, cn a mail. [4]

Basic Encoding Rules

Pravidla základního kódování (BER) určují, jak by měla být data kódována pro přenos, nezávisle na typu stroje, programovacím jazyce nebo reprezentaci v aplikačním programu.

BER používá pro kódování informací formát Tag-Length-Value (TLV). Typ nebo značka označuje, jaký druh dat následuje, délka označuje délku dat, která následují, a hodnota představuje skutečná data. Každá hodnota se může skládat z jedné nebo více hodnot kódovaných TLV, každá má svůj vlastní identifikátor, délku a obsah.

Kódování TLV sice zvyšuje počet přenesených oktetů, ale usnadňuje zavádění nových polí do zpráv, které mohou zpracovat i starší implementace. Další výhodou je, že předvídatelnost kódování usnadňuje ladění. [3]

Každá příchozí zpráva musí dodržovat předem daný formát. Např. *bindRequest* Má vždy označení sekvence, následuje délka sekvence, *messegeID*, označení *bindRequestu*, délka zprávy a zbytek zprávy.
Podle toho lze postavit na pevno i *bindResponse*.

```
char bind_response[] = {0x30, 0x0c, 0x02, 0x01, 0x01, 0x61, 0x07, 0x0a,
0x01, 0x00, 0x04, 0x00, 0x04, 0x00};
```

První okten označuje, že se jedná o sekvenci a druhý o jeho zbytek. 0x02 0x01 0x01 označuje *messageID* (0x02 označení čísla, 0x01 délka a 0x01 hodnota). 0x61 je označení *bindRespons* a 0x07 je jeho délka. Zbytek je zpráva, která obsahuje věci jako například autentizaci.

Takhle jsou následně sestaveny i *searchResEntry*, jen mnohem složitější.

Kód

TCP spojení

Po zpracování argumentů, je nastaveno tcp spojení. Pro to existuje funkce *tcp*, která byla převzata z NESFIT, IPK projekty. [5] Ve funkci je vytvořen a na bindován soket a následně poslouchá. Jakmile zachytí paket *bindRequest*, odpoví s *bindResponse* a přejde do stavu čekání na *searchRequest*. Po jeho zachycení je vyvolaná funkce *get_message*.

Zpracování příchozí zprávy

Vzhledem k tomu, že všechny příchozí zprávy mají pevně daný formát, tak se dá pozice jednotlivých parametru vypočítat. To provádí funkce *get_message*. Ta zároveň zavolá vunkci search_database, která prohlédá soubor se záznamy, a send_response, která vygeneruje *searchResEntry*. *search_database* otevírá soubor, rozdělí na 3 části pomocí regexu a pomocí parametru filtru uloží (pomocí funkce *append_attributes*) do struktury pouze ty záznamy, kde se filtry shodují. *send_response*, nejdříve zpracuje a zjistí, jaké atributy jsou zadany. Ty pak připojí ke stringu result a připojí i zbytek zprávy.

Pomocné funkce

Program obsahuje nějaké pomocné funkce, jako *concat_hex_array* na spojení, nebo *int_to_string* na převedení čísla do jeho hex stavu (4 -> 0x04) a připojení ke result.

Struktury

První struktura je database. Ta slouží k filtraci požadovaných záznamu ze souboru.
Druhá je attributes. Ta má navíc i pointer na následující strukturu. Při sestavování searchResDone je vytvořen list struktur attributes a ten ve while po jednom pošle klientovi.
Povolené atributy jsou *mail*, *uid* a *cn*.
Je důležité brát v potaz, že celý program neukládá informace ze zadaného souboru do proměnné, nýbrž pokaždé prochází řádky a ukládá potřebné záznamy do struktur a následně i uvolňuje.

Nedostatky mého programu

Nedostatky programu jsou: nepodporuje and, or a not operace, při substringu ‘dn’ část ve výsledku, zobrazuje nesmysly (způsobeno mým nedostatkem pochopení části), zároveň při substringu při nějakém určitém substringu (nevím už jaký a nedokázal jsem napodobit situaci), *searchResEntry* obsahuje byty, které jsou záporné.

Testování

Pro ukázku testování, jde zde 5 příkladů. Můj soubor database.csv obsahuje pouze lehce přes 130 záznamu. Testováno bylo lokálně na port 389, ale na merlinovi mi to šlo spustit až od portu 2001. Snažil jsem se kombinovat parametry.
Program podporuje: nastavení limitu (při nastavení limitu, nebo při jeho vynechání, je limit nekonečno, stejně jak na školním LDAP serveru), nastavení atributů (při zadání nesmyslného atributu, se atribut ignoruje), nastavení filtru (substring a equalMatch).
Příkazy jsou označeny $.

Testoval jsem to tak, že v jedné konzoli jsem spustil ldap server:

sudo ./isa-ldapserver -p 389 -f database.csv (sudo jsem musel použít na svém zařízení jen)

a v druhe jsem zadával příkazy:

1. Vypiš 5 záznamu bez filtru a atributů

   $ ldapsearch -H ldap://localhost:389 -x -z 5 -b "dc=vutbr,dc=cz"

   # extended LDIF
   #
   # LDAPv3
   # base <dc=vutbr,dc=cz> with scope subtree
   # filter: (objectclass=*)
   # requesting: ALL
   #

   # fit.vutbr.cz
   dn: dc=fit,dc=vutbr,dc=cz
   uid: xfeile00
   mail: xfeile00

   # fit.vutbr.cz
   dn: dc=fit,dc=vutbr,dc=cz
   uid: xfelkl00
   mail: xfelkl00

   # fit.vutbr.cz
   dn: dc=fit,dc=vutbr,dc=cz
   uid: xfiala41
   mail: xfiala41

   # fit.vutbr.cz
   dn: dc=fit,dc=vutbr,dc=cz
   uid: xfiala38
   mail: xfiala38

```
# fit.vutbr.cz
dn: dc=fit,dc=vutbr,dc=cz
uid: xfiedo00
mail: xfiedo00

# search result
search: 2
result: 0 Success

# numResponses: 6
# numEntries: 5
```

2. Vypiš 5 záznamu s uid = xvorob02
   $ ldapsearch -H ldap://localhost:389 -x -z 5 -b "dc=vutbr,dc=cz" 'uid=xvorob02'

```
# extended LDIF
#
# LDAPv3
# base <dc=vutbr,dc=cz> with scope subtree
# filter: uid=xvorob02
# requesting: ALL
#

# xvorob02, fit.vutbr.cz
dn: uid=xvorob02,dc=fit,dc=vutbr,dc=cz
uid: xvorob02
mail: xvorob02

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

3. Vypiš záznamy, kde uid = xv*2 (tady je neočekávaný výstup v dn, správně by tam mělo být uid a lokace)

   $ ldapsearch -H ldap://localhost:389 -x -b "dc=vutbr,dc=cz" 'uid=xv*2'

```
# extended LDIF
#
# LDAPv3
# base <dc=vutbr,dc=cz> with scope subtree
# filter: uid=xv*2
# requesting: ALL
#
```

```
#
dn:: dWlkPYACeHaCATIsZGM9Zml0LGRjPXZ1dGJyLGRjPWN6
uid: xvorob02
mail: xvorob02

#
dn:: dWlkPYACeHaCATIsZGM9Zml0LGRjPXZ1dGJyLGRjPWN6
uid: xvrana32
mail: xvrana32

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

4. Vypiš jméno a email, kde uid = xvorob02

```
$ ldapsearch -H ldap://localhost:389 -x -b "dc=vutbr,dc=cz" 'uid=xvorob02' mail cn

# extended LDIF
#
# LDAPv3
# base <dc=vutbr,dc=cz> with scope subtree
# filter: uid=xvorob02
# requesting: mail cn
#

# xvorob02, fit.vutbr.cz
dn: uid=xvorob02,dc=fit,dc=vutbr,dc=cz
cn: Vorobec Valentyn
mail: xvorob02

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

5. Vypiš jméno a email, kde uid = xv*b*2 (tady je neočekávaný výstup v dn, správně by tam mělo být uid a lokace)

```
$ ldapsearch -H ldap://localhost:389 -x -b "dc=vutbr,dc=cz" 'uid=xv*b*2' mail
cn
```

```
# extended LDIF
#
# LDAPv3
# base <dc=vutbr,dc=cz> with scope subtree
# filter: uid=xv*b*2
# requesting: mail cn
#

#
dn:: dWlkPYACeHaBAWKCATIsZGM9Zml0LGRjPXZ1dGJyLGRjPWN6
cn: Vorobec Valentyn
mail: xvorob02

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

6. Vypiš 5 záznamů, kde je atribut mail a nesmyslný atribut ll

```
$ ldapsearch -H ldap://localhost:389 -x -z 5 -b "dc=vutbr,dc=cz" mail ll

# extended LDIF
#
# LDAPv3
# base <dc=vutbr,dc=cz> with scope subtree
# filter: (objectclass=*)
# requesting: mail ll
#

# fit.vutbr.cz
dn: dc=fit,dc=vutbr,dc=cz
mail: xfeile00

# fit.vutbr.cz
dn: dc=fit,dc=vutbr,dc=cz
mail: xfelkl00

# fit.vutbr.cz
dn: dc=fit,dc=vutbr,dc=cz
mail: xfiala41

# fit.vutbr.cz
dn: dc=fit,dc=vutbr,dc=cz
mail: xfiala38
```

```
# fit.vutbr.cz
dn: dc=fit,dc=vutbr,dc=cz
mail: xfiedo00

# search result
search: 2
result: 0 Success

# numResponses: 6
# numEntries: 5
```

7. Vypiš 5 záznamů, bez -b (tady je žádný výstup v dn, správně by tam mělo být uid a lokace)

```
$ ldapsearch -H ldap://localhost:389 -x -z 5

# extended LDIF
#
# LDAPv3
# base <> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

#
dn: dc=fit,
uid: xfeile00
mail: xfeile00

#
dn: dc=fit,
uid: xfelkl00
mail: xfelkl00

#
dn: dc=fit,
uid: xfiala41
mail: xfiala41

#
dn: dc=fit,
uid: xfiala38
mail: xfiala38

#
dn: dc=fit,
uid: xfiedo00
mail: xfiedo00
```

```
# search result
search: 2
result: 0 Success

# numResponses: 6
# numEntries: 5
```

8. Příklad, když limit je 0 (v tom případě limit ignoruj, stejně jak u školního ldap serveru)

```
$ ldapsearch -H ldap://localhost:389 -x -z 0 -b "dc=vutbr,dc=cz" 'uid=xv*2'

# extended LDIF
#
# LDAPv3
# base <dc=vutbr,dc=cz> with scope subtree
# filter: uid=xv*2
# requesting: ALL
#

#
dn:: dWlkPYACeHaCATIsZGM9Zml0LGRjPXZ1dGJyLGRjPWN6
uid: xvorob02
mail: xvorob02

#
dn:: dWlkPYACeHaCATIsZGM9Zml0LGRjPXZ1dGJyLGRjPWN6
uid: xvrana32
mail: xvrana32

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

9. Vypiš záznam, kde je atribut mail, nesmyslný atribut ll a atribut uid

```
$ ldapsearch -H ldap://localhost:389 -x -z 0 -b "dc=vutbr,dc=cz" 'uid=xvorob
02' uid ll mail


# extended LDIF
#
# LDAPv3
# base <dc=vutbr,dc=cz> with scope subtree
# filter: uid=xvorob02
```

```
# requesting: uid ll mail
#

# xvorob02, fit.vutbr.cz
dn: uid=xvorob02,dc=fit,dc=vutbr,dc=cz
mail: xvorob02

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Zdroje

1. **LDAP - Wikipedia.** (2023) [https://cs.wikipedia.org/wiki/LDAP]
2. **DigitalOcean.** (2023). *Understanding the LDAP Protocol, Data Hierarchy, and Entry Components.* [https://www.digitalocean.com/community/tutorials/understanding-the-ldap-protocol-data-hierarchy-and-entry-components]
3. **OSS Nokalva.** (2023). *Basic encoding rules.* [https://www.oss.com/asn1/resources/asn1-made-simple/asn1-quick-reference/basic-encoding-rules.html]
4. **Datatracker. RFC steam IETF.** (1995). Tim Howes, Steve Kille, Wengyik Yeong. [https://datatracker.ietf.org/doc/html/rfc1777]
5. **NESFIT/IPK-Projekty.** (2023). [https://git.fit.vutbr.cz/NESFIT/IPK-Projekty/src/branch/master]