LDAP Server Documentation

Course project to Network Applications and Network Administration

13.11.2023

Jakub Lukáš

Lightweight Directory Access Protocol

LDAP Overview

The Lightweight Directory Access Protocol is used for searching directory entries via network. The entries are located at database on server side and generally contain information about individuals or resources. Each entry has several attributes that differ for every server implementation. The main function of LDAP server is to respond to clients with relevant entries. Clients may specify their requests for entries by applying filters.

LDAP Communication

The server-client communication described below is officially defined in RFC documents. Hexadecimal encoded bytes are used for data transfer and the data are encoded in BER format. Client first sends bind request with simple (anonymously without password, or username and password), or SASL authentication. Then the server responds with Bind Response. Client then may send a Search Request with search filter and wanted entry attributes specified. Server then proceeds to search the database and send found entries back with Search Result Entry containing entry attributes client requested. After all relevant entries are sent or the size limit of sent entries gets exceeded, server sends Search Result Done message with rather success or error message. Client then sends Unbind Request and communication ends.

Supported filters

equalityMatch

This filter contains two octet fields. First goes for matching attribute and second is for matching attribute value. In order for specific entry to be matched, the attribute of the entry specified in the first filed must be exactly the same as the string value specified in the second filed.

Substrings

When working with substrings in LDAP filters, you can employ wildcards to match partial string values. Filter may specify number of substrings and attribute type. In order to match the entry, the value of the attribute of the entry specified in the filter must contain all substrings from the filter with 0...n characters between them. Filter may also specify what substring should be on the start and end of the entry attribute value.

Logical – AND, OR, NOT

These filters perform logical operations on their inner subfilters, which are rather one of the filters described above or nested logical filter. In order for an entry to be matched, all subsilters in the AND filter must match the entry. Unlike in OR filter, where just one subfilter has to match the entry to be matched. Both of these filters can have from one up to n subfilters. The NOT filter has exactly one subfilter and it is matched if the subfilter is not matched.

Implementation

Server is implemented in c++ and is compiled using Makefile. Implementation is divided into 5 modules, each of which has its own .cpp and header file. It heavily relies on strings and vectors. The database server works with should be a local .csv file, every line from the database file should contain one entry per line. Entry attributes are separated by ";" and it is in "uid;cn;mail" format. Server only supports ascii database files, so Czech language is not supported for that reason. Server is parallel and non-blocking. Some of key functions and classes will be explained in detail.

main

This code sets up a basic server that can handle multiple clients concurrently using threads. It parses command-line arguments to specify the database file and, optionally, the port number. The communication with clients involves receiving data and processing it using the "parsePacket" function.

parsePacket

This function provides a mechanism for parsing different types of LDAP packets, handling various components of the requests, and responding appropriately to bind, search, and unbind requests from clients. The error-handling mechanism includes sending error messages with specific result codes to the clients.

SearchTree

The searchTree class serves as the orchestrator for handling LDAP search requests within the server, employing the searchNode class to represent and evaluate the search filter. It encapsulates the logic for parsing search filters, evaluating them against entries in the database, and sending the matching results to the client. Key feature is Search method that opens and reads the database file line by line and evaluates whether the entry matches the search filter criteria.

SearchNode

The searchNode class represents a node in the tree structure used to model LDAP search filters. It encapsulates the logic for parsing and evaluating different types of filters. Key features are parseFilter method that interprets the filter content based on its type and filter evaluation against a ResultEntry object.

ResultEntry

The ResultEntry class is responsible for parsing entries retrieved from the database, organizing them into attributes (name, UID, mail), and facilitating subsequent evaluation against the LDAP search filter.

ByteStream

The ByteStream class provides a simple interface for reading bytes from a buffer and managing the length and message ID associated with the buffer

PacketSender

The packetSender class is responsible for capculating response messages into LDAP packets and sending them to the client.

Testing

Server was tested manually on BUT FIT faculty linux server merlin.fit.vutbr.cz on port 20021. To test sever functionality ldapsearch command was used from faculty server eva.fit.vutbr.cz. Test cases were added during development process. Results were compared with results from faculty LDAP server ldap.fit.vutbr.cz for the same testcases. While the database on which the faculty LDAP server operates is larger than the one used for testing (ldap-lidi-ascii.csv), it's important to note that the test database is a subset of the faculty server database. Therefore, the obtained results are valid as long as all outcomes from the test database are encompassed within the results from the faculty LDAP server. All testcases are located in tests-ldap direcroty.

What should be tested:

- EqualityMatch
- Substrings
    - Initial
    - Initial, any
    - Initial, any, final
    - Initial, final
    - Any, final
    - Final
    - any
- AND
- OR
- NOT
- Combinations of logical filters
- No entry returned
- Size limit
- Returning specific attributes

Tests and expected results

For simplicity, all testcases that return something are designed to return my uid (xlukas18). These tests were performed mainly to test server ability to search results and other functionalities.

1. Simple equalityMatch filter

ldapsearch -x -H ldap://merlin.fit.vutbr.cz:20021 "(cn=Lukas Jakub)"

Expected results: xlukas18

2. And with substring filter

   ldapsearch -x -H ldap://merlin.fit.vutbr.cz:20021 "(&(uid=*18)(cn=*Jakub*))"

Expected results: xlukas18

3. Or and substring filter

ldapsearch -x -H ldap://merlin.fit.vutbr.cz:20021 "(|(uid=xlukas*)(cn=*L*Jakub*))"

Expected results: xlukas18, xlukas15

4. Or inside and filter

```
ldapsearch -x -H ldap://merlin.fit.vutbr.cz:20021
    "(&
        (|(uid=*xlukas*)(uid=*xpoliv*))(|(uid=*18)(uid=*07))
    )"
```

Expected results: xlukas18, xpoliv07

5. And with not filter

```
ldapsearch -x -H ldap://merlin.fit.vutbr.cz:20021 "(&(uid=*lukas*)(!(cn=*Ondrej*)))"
```

Expected results: xlukas18

6. No result

```
ldapsearch -x -H ldap://merlin.fit.vutbr.cz:20021 "(!(mail=*fit*))"
```

Expected results: no results

7. Size limit

```
ldapsearch -x -z 2 -H ldap://merlin.fit.vutbr.cz:20021 "(uid=xves*)"
```

Expected results: xvesel40, xvesel39 (other entries that match this filter are also considered correct, point of this test is that server returns only two entries)

Summary

Given that the program successfully passed the tests, we can reasonably infer, based on my current understanding, that it is functioning as it should. The testing, however, was not as thorough as it could have been. Constraints such as time limitations and the distinct purpose of the project, which focuses on understanding LDAP, contributed to the less extensive testing.

Bibliography

Used for research purposes

- RFC 4511 - https://www.rfc-editor.org/rfc/rfc4511
- Ldap ASN.1 Codec- https://cwiki.apache.org/confluence/display/DIRxSRVx10/Ldap+ASN.1+Codec
- Howes, T., Smith, M., & Good, G. S. (2003). *Understanding and Deploying LDAP Directory Services, 2nd Edition*. Addison-Wesley Professional.