ISA Project - LDAP Server

Author: Šimon Benčík, xbenci01

Introduction

This project aims to develop a simple LDAP Server, adhering to the specifications outlined in relevant RFCs. The server is designed to support simple bind and search requests, accommodating filters such as equalityMatch, substringMatch, AND, OR, NOT. This document details the implementation and provides insights into its functionality.

Theory

LDAP is a compact and efficient binary protocol, utilizing ASN.1 (Abstract Syntax Notation One) for structured data representation. ASN.1 encompasses various encoding methods, each with distinct advantages for specific contexts. LDAP employs the Basic Encoding Rules (BER), offering an optimal balance for its applications. The LDAP protocol's detailed specifications and structures are primarily outlined in RFC 4511[ˆ2].

Design

The LDAP server is implemented in C++17, following object-oriented design principles. The design emphasizes polymorphism and incorporates the factory pattern to enhance modularity and flexibility.

Implementation

The project is organized into two main directories: 'src', containing module implementations, classes, and functions, and 'include', housing the corresponding header files. The program's entry point, **main.cpp**, parses initial arguments, establishes a server socket, and manages parallel TCP communication. Child processes handle incoming bytes, utilizing a type-determining function to create appropriate **LDAPMessage** subclass instances defined in **message.cpp**. These subclasses, contain **BERParser** instances for message parsing as well as functions and variables needed to handle parsing of the message and responding to it. The BERParser is crucial for navigating the buffer and advancing its position, it contains functions to decode ASN.1's primitive types and more complex functions for parsing nested filters into a tree-like structure. Each subclass of LDAPMessage overrides the parse() and respond() methods. This structure allows for future extensions, such as add, modify, and delete functionalities. Filter evaluation and CSV manipulation are handled in **search.cpp**, which contains structures related to filters and functions for individual filter evaluation and entry retrieval. Initially, the filtering was designed to evaluate every entry against each filter, which proved inefficient and incorrect. This approach was later refined to retrieve entries from the CSV file during the search response function and evaluate each one of them against a filter tree, enhancing performance through lazy evaluation. The server concludes each search with a searchResDone response. Currently, the server does not handle incorrect packet structures or unknown message types, which is an area for potential improvement. Further limitations are noted in **README** file. A detailed documentation of individual code components can be reviewed in docs/ folder after generating it using **make doxygen**.

System requirements

- Operating system: Linux or macOS
- Compiler: GCC or Clang with C++17 support
- Libraries: Standard C++ libraries

Usage

After compiling the project with **make**, the server is started as follows:

```
./isa-ldapserver {-p <port>} -f <file>
```

Options:
- -p <port>: Specify port for server to run on, by default it is set to 389.
- -f <file>: Path to ldap database in csv format. Required

Testing

Testing was performed manually throughout the development process. Majority of testing was done in Wireshark - comparing the request hex dump and response hex dump to the reference server ldap.fit.vutbr.cz as well as output testing. Emphasis was placed on ensuring the accuracy of message encoding/decoding and the robustness of filter implementations, ranging from simple to complex nested structures. Tests were done mainly on macOS, with additional reference testing on the merlin.

Literature

1. Wilson, Neil. "LDAPv3 Wire Protocol Reference: The ASN.1 Basic Encoding Rules." https://ldap.com/ldapv3-wire-protocol-reference-asn1-ber/ [^1].
2. Sermersheim, J., Ed. "Lightweight Directory Access Protocol (LDAP): The Protocol." RFC 4511. IETF, June 2006. https://www.rfc-editor.org/rfc/rfc4511.txt [^2].
3. Zeilenga, K., Ed. "LDAP: Technical Specification Road Map." RFC 4510. IETF, June 2006. https://www.rfc-editor.org/rfc/rfc4510.txt.
4. Harrison, R., Ed. "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms." RFC 4513. IETF, June 2006. https://www.rfc-editor.org/rfc/rfc4513.txt.

5. Smith, M., Ed. “Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters.” RFC 4515. IETF, June 2006. https://www.rfc-editor.org/rfc/rfc4515.txt.