# Light Weight Directory Access Protocol (LDAP)

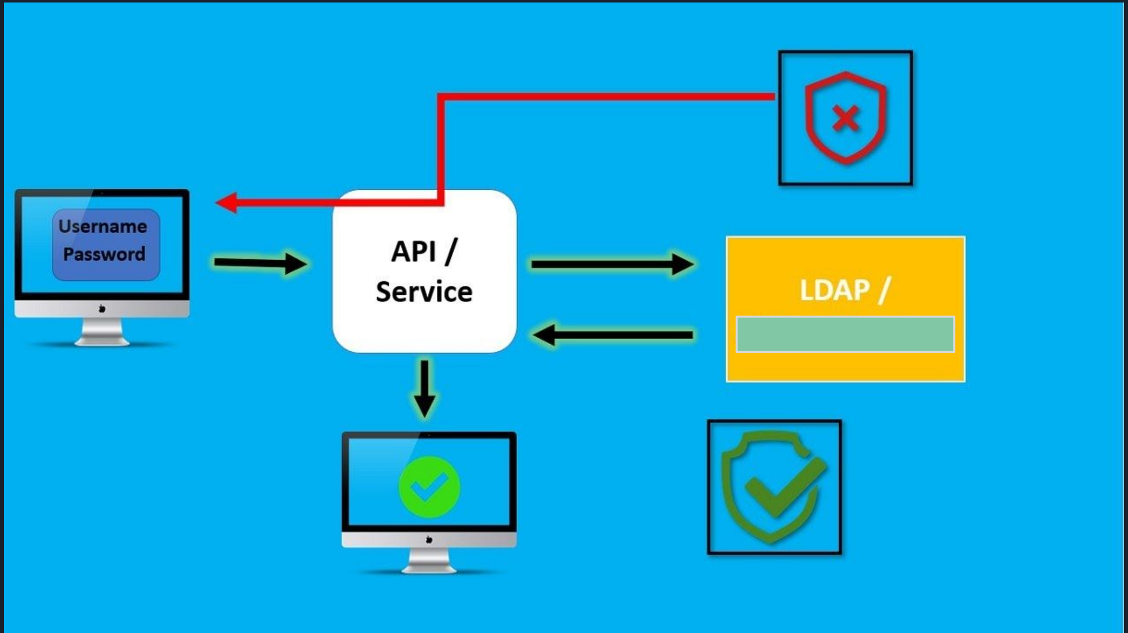
*By,*
*Yadhu Krishna M*
*Sayooj B Kumar*

## What is LDAP?

- Lightweight Directory Access Protocol
- Used for accessing & maintaining distributed directory information services
  - Used for authentication
  - Storing information about users, groups, and applications
  - General purpose data storage
- Based on the X.500 directory-information services
- Used to store and retrieve data from a hierarchical directory structure.
- Open Protocol.

## What is a directory service?

- Store, organize and present data in a key-value type format
- Optimized for lookups, searches, and read operations over write operations.



## What LDAP is NOT

- **LDAP** is not a server / database
- **LDAP** is not a network service / device
- **LDAP** is not an authentication procedure
- **LDAP** is not a user/password repository
- **LDAP** is not a specific open or closed source product
- **LDAP IS A PROTOCOL.**

## Basic Data Components

1. Attributes
2. Entries
3. Data Information Trees - DIT

Refer: https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol#Schema

## Attributes

- Data is stored in elements called attributes
- Attributes are basically key-value pairs.
- Keys have predefined names which are dictated by the objectClasses.
- Other elements within LDAP are used for structure, organization, etc.

- Example: `mail=example.com`

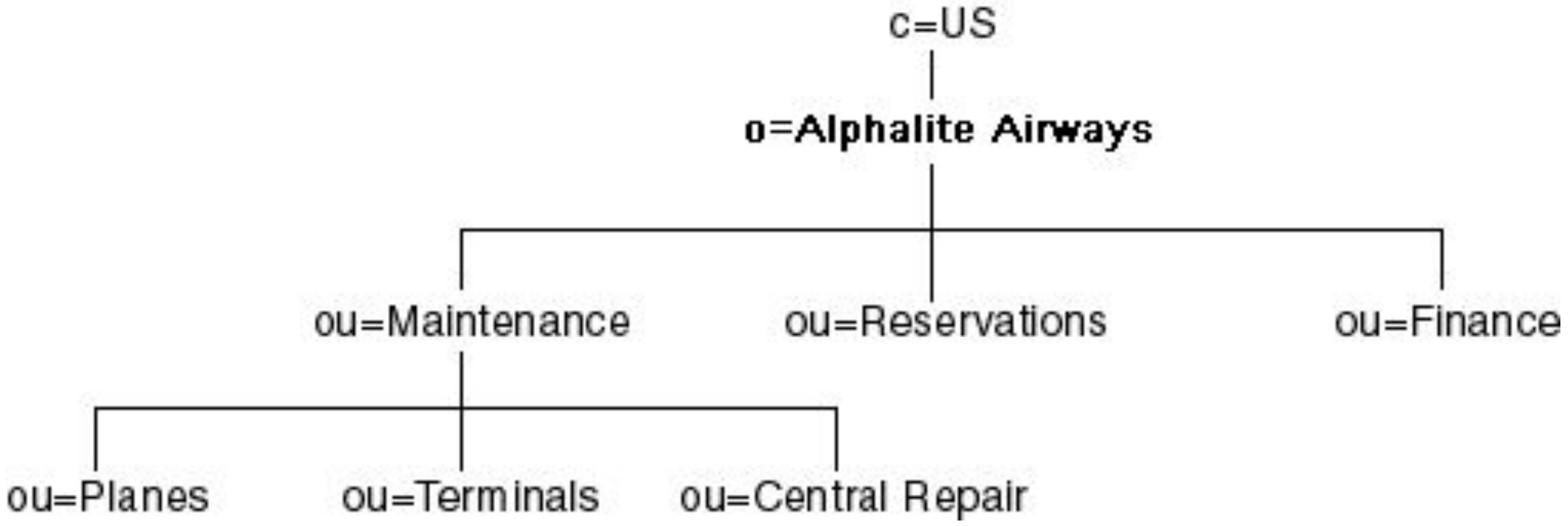
## Entries

- Collection of attributes under a name used to describe something.
- Similar to a row in a relational database system.

- LDAP Data Interchange Format (LDIF)

```
dn: sn=Ellingwood,ou=people,dc=digitalocean,dc=com
objectclass: person
sn: Ellingwood
cn: Justin Ellingwood
```

## DIT- Data Information Trees



## DIT represents an organizational structure.

```
dn: sn=Ellingwood,ou=people,dc=digitalocean,dc=com
objectclass: person
sn: Ellingwood
cn: Justin Ellingwood
```

DN -> Distinguished Name (used to identify entry)
CN -> Common Name
SN -> Surname

## Defining LDAP Data Components

## Attribute Definitions

Strictly according to RFC 4512.

1. Numeric OID.
2. Optional human-readable description.
3. Optional reference to a superior attribute type.
4. Optional reference to the equality matching rule
5. Optional reference to the ordering matching rule.
6. Optional reference to the substring matching rule.
7. Optional string “SINGLE-VALUE”,
8. Optional string “NO-USER-MODIFICATION”

*…And much more*

https://ldap.com/attribute-types/

## Example for Attr. Definition

```
attributetype (
     2.5.4.41 NAME 'name'
     DESC 'RFC4519: common supertype of name attribute'
     EQUALITY caseIgnoreMatch
     SUBSTR caseIgnoreSubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768}
)
```

```
Ln 1: Unique Object ID & Name
Ln 2: Description
Ln 3: How to compare equality.
Ln 4: Defines how substring should be checked.
Ln 5:
```

## ObjectClass Definitions

- Attributes are collected within entities called objectClasses.
- Eg. “person” is an objectClass.

```
dn: sn=Ellingwood,ou=people,dc=digitalocean,dc=com
objectclass: person
sn: Ellingwood
cn: Justin Ellingwood
```

## Schemas

- Collections of related objectClasses and attributes.
- ObjectClass definitions and attribute definitions grouped together.

## Data Organization

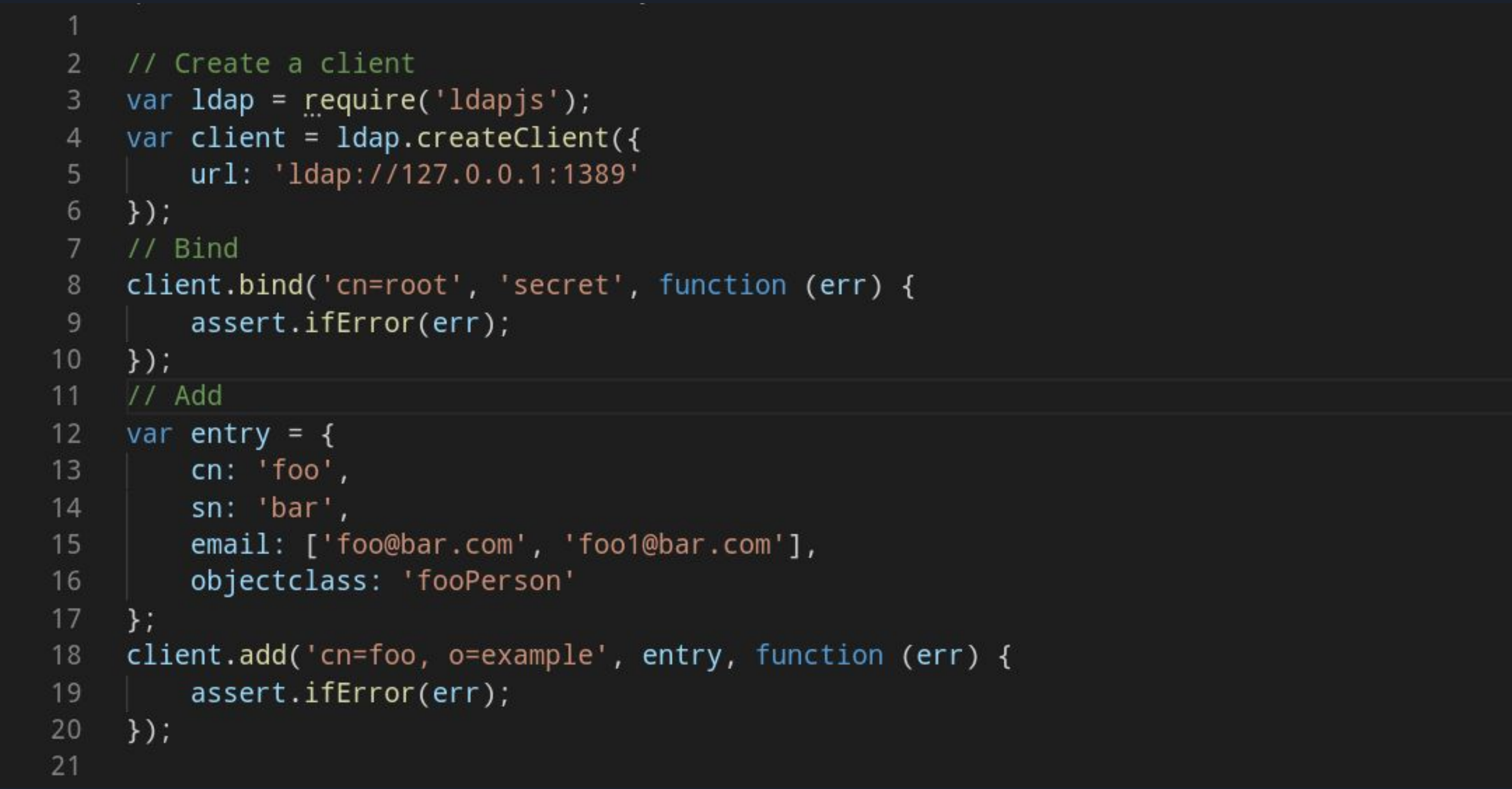
## Placing Entries within the DIT

|  |  |  |
|---|---|---|
| Associated with Domain | : | `(dc=example,dc=com)` |
| Associated with Location | : | `(l=new_york,c=us)` |
| Associated with Organization | : | `(ou=marketing,o=Example_Co)` |

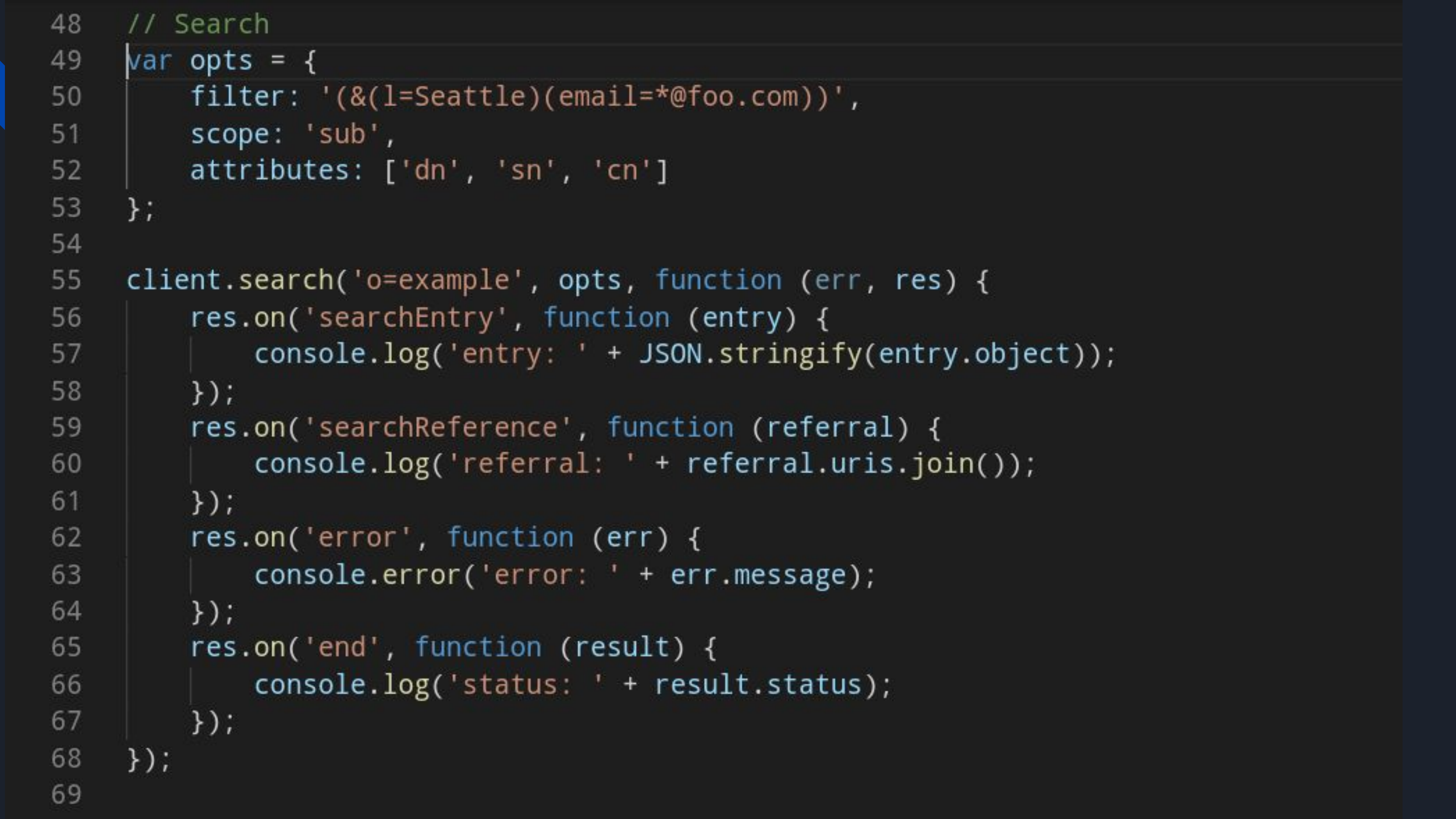
## Protocol Variations

- ldap://
  - basic LDAP protocol
  - structured access to a directory service
- ldaps://
  - indicate LDAP over SSL/TLS
  - Deprecated
- ldapi://
  - LDAP over an IPC. (Inter-Process Communication)
  - Secure

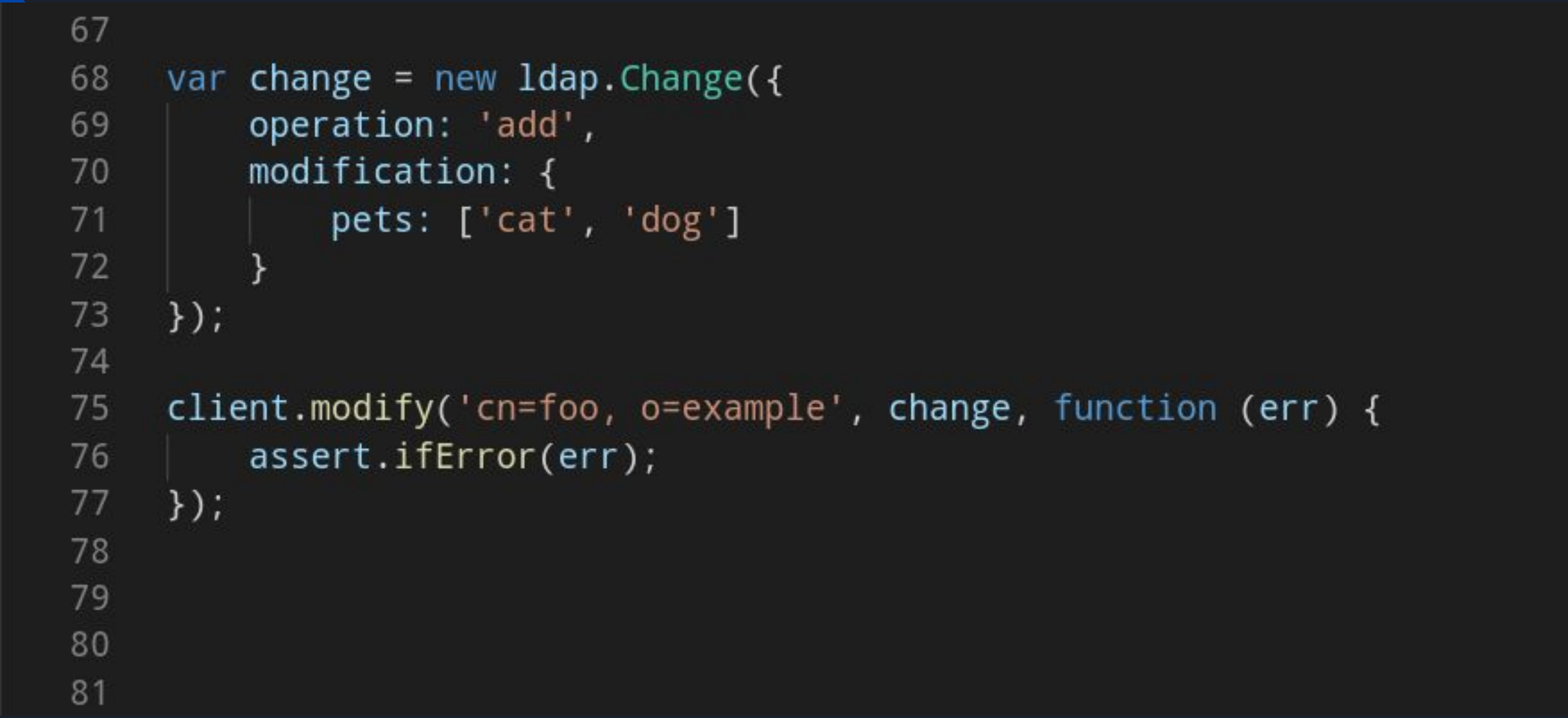
## Basic Operations On LDAP



```javascript

// Create a client
var ldap = require('ldapjs');
var client = ldap.createClient({
    url: 'ldap://127.0.0.1:1389'
});
// Bind
client.bind('cn=root', 'secret', function (err) {
    assert.ifError(err);
});
// Add
var entry = {
    cn: 'foo',
    sn: 'bar',
    email: ['foo@bar.com', 'foo1@bar.com'],
    objectclass: 'fooPerson'
};
client.add('cn=foo, o=example', entry, function (err) {
    assert.ifError(err);
});

```



```javascript
// Search
var opts = {
    filter: '(&(l=Seattle)(email=*@foo.com))',
    scope: 'sub',
    attributes: ['dn', 'sn', 'cn']
};

client.search('o=example', opts, function (err, res) {
    res.on('searchEntry', function (entry) {
        console.log('entry: ' + JSON.stringify(entry.object));
    });
    res.on('searchReference', function (referral) {
        console.log('referral: ' + referral.uris.join());
    });
    res.on('error', function (err) {
        console.error('error: ' + err.message);
    });
    res.on('end', function (result) {
        console.log('status: ' + result.status);
    });
});
```



```
var change = new ldap.Change({
    operation: 'add',
    modification: {
        pets: ['cat', 'dog']
    }
});

client.modify('cn=foo, o=example', change, function (err) {
    assert.ifError(err);
});
```

## Other Operations

- compare(dn, attribute, value, controls, callback)
- del(dn, controls, callback)
- modify(name, changes, controls, callback)
- modifyDN(dn, newDN, controls, callback)
- unbind(callback)

## LDAP Injection

## LDAP INJECTION

- **Something similar like sql injection**
- **Unlike sql we don’t have many ways or functions to exploit**
- **Injections are mainly found in ldap search**
- **Similar to sql blind based we have ldap blind based injection**

## Why injection happens?

- unsanitized input

## Prevention

- Sanitize input
- * ( ) . & - _ [ ] ` ~ | @ $ % ^ ? : { } ! '

## INJECTION

- * (something similar like % in mysql)
- a* (means there are one or more character after a)
- *abc* (means there are character before and after “abc”)
- * is mainly use in blind based injection
- We can also use < > <= >= in certain situations
- Other useful operation is (AND “&”, OR “|” and NOT “!”)

- Just a “*” can work similar to ‘ or 1-- - in sql
- Ie it output  all data  similar to how all row is returned in sql
- We can also add more attributes but make sure parenthesis are balanced

## Example

- Let out query be something like this
- (&( cn = ’test’ )( mail= {our input} ) )
- If {our input}=test@gmail.com)(userPassword=a*
- Resulting query will be
- (&( cn = ’test’ )( mail= test@gmail.com)(userPassword=a* ) )
- From here we can bruteforce for password

## Let’s play around

https://github.com/sayoojbkumar/ldap_injection

Aim: find out anyone of user mail login with the mail capture the flag

# Reference

- https://ldap.com/attribute-types/
- https://www.blackhat.com/presentations/bh-europe-08/Alonso-Parada/Whitepaper/bh-eu-08-alonso-parada-WP.pdf
- https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol#Schema
-

## Thank You !