Если вы видите что-то необычное, просто сообщите мне.

Getting Started With LDAP in Go

Недавно пришлось писать доброе количество кода на Go, который взаимодействует с AD, для одного моего клиента. AD использует легковесный протокол доступа (LDAP) для клиент серверного подключения. LDAP - старый и могущественный протокол для взаимодействия с сервисов, несмотря на это, многие мои друзья спорят, что это реливкия прошлого на данный момент. Я с ними не согласен, но мое объяснение может занять целый отдельный пост.

Два три года назад я столкнулся с LDAP вызовом. Пришлось писать некоторый Go код, который должен использовать LDAP для групп и некоторых других авторизационных вещей. В то время библиотеки были в плачевном состоянии. К сожалению, это не было выходом в то время, но к счатью, я быстро изучил работу в Go с LDAP, что поменялось к лучшему.

Эта статья предоставляет базовое введение в удивительный модуль go-ldap. Что-то вроде ввдения, о котором я мечта, когда я начал работать над LDAP проектом для моего клиента. Кроме того, я хотел иметь некую ссылку на инструкцию к которой я смог бы вернуться в будущем если понадобится. Я надеюсь вы не только найдете этот пост полезным, но так же научитесь чему-то новому. Давайте начнем.

Подключение

Прежде чем мы сможем сделать что-нибудь с AD нужно подключиться к серверу. Давайте, для начала, подключимся к серверу AD с помощью LDAP, мне кажется это естественно и правильно с точки зрения смысла. Еще Go модули описанные в этом посту называются ldap-go, давайте разберемся с LDAP. Модуль go-ldap предоставляет несколько возможностей для подключения к LDAP серверу. Пройдеся по ним подробнее.

Все типи LDAP подключения обрабатывается `DialURL` функцией. Есть несколько других функции доступных в модуле, но документация предполагает что `DialURL` останется единственно рабочей. Как предписывает название функции, вы передаете ей URL и функция пытается подключиться к удаленному LDAP серверу и вернуть подключение если всё прошло успешно.

Пример кода можно найти ниже:

```
ldapURL := "ldaps://ldap.example.com:636"
l, err := ldap.DialURL(ldapURL)
if err != nil {
        log.Fatal(err)
}
defer l.Close()
```

Этот код пытается установить TLS подключение к удаленному серверу. DialURL выводит тип подключения из URL который был передан функции, который в этом случае является ldaps(c s - безопасный).

Если вам нужны подробности TLS конфигурации, функция принимает самоподписанные TLS конфиги через дополнительный параметр:

```
ldapURL := "ldaps://ldap.example.com:636"
l, err := ldap.DialURL(ldapURL, ldap.DialWithTLSConfig(&tls.Config{InsecureSkipVerify: true}))
if err != nil {
        log.Fatal(err)
}
defer l.Close()
```

```
Пример кода выше дан только в качестве демонстрации! Ни когда не пропускайте проверку TLS
когда создаете её.
```

Если вы не хотите использовать TLS, можно просто опустить "s" в URL адресе:

```
ldapURL := "ldap://ldap.example.com:389"
l, err := ldap.DialURL(ldapURL)
if err != nil {
        log.Fatal(err)
```

```
}
defer l.Close()
```

Вы можете так же опустить порт из адреса. Код выше показывает его для краткости. Если вы опустите порт `DialURL` функция автоматически подставит порт 639 для ldaps или 389 для ldap подключений. По умолчанию LDAP порт так же доступен через глобальные переменные DefaultLdapsPort и DefaultLdapPort.

Как вариант, вы можете использовать `NewConn(conn net.Conn, isTLS bool)` функцию которая позволяет вам использовать чистое `net.Conn` подключение, которое вам может понадобиться, в том или ином случае.

Наконец, вы можете так же обновить существующее подключение до TLS используя функцию `StartTLS()`:

```
l, err := DialURL("ldap://ldap.example.com:389")
if err != nil {
    log.Fatal(err)
}
defer l.Close()

// Now reconnect with TLS
err = l.StartTLS(&tls.Config{InsecureSkipVerify: true})
if err != nil {
    log.Fatal(err)
}
```

Теперь зная как подключиться к LDAP серверу мы можем перейти к следующему шагу.

Связывание

Связывание это шаг, где LDAP сервере уже аутентифицирует клиента. Если клиент успешно прошол аутентификацию, сервер предоставляет ему доступ основываясь на его привилегиях.

Есть несколько путей для создания LDAP связываний используя ldap-go. Начнем с простого случая: неаутентифицируемое связываение.

Иногда LDAP сервер дает ограниченный доступ только для чтения для неаутентифицируемых клиентов. Выпонить его можно следующим образом:

```
// Подключяемя к серверу как делали выше

err = l.UnauthenticatedBind("cn=read-only-admin,dc=example,dc=com")
if err != nil {
    log.Fatal(err)
}
```

Если вам, всё равно нужно аутентифицироваться, в вашем распоряжении есть 2 варианта: `SimpleBind` и `Bind`. Последнее - это хорошая обертка вокруг первого, поэтому я люблю использовать её.

```
// Подключяемя к серверу как делали выше

err = l.Bind("cn=read-only-admin,dc=example,dc=com", "p4ssw0rd")
if err != nil {
    log.Fatal(err)
}
```

Наконец вы можете сделать `External` свзяывание которое согласно официальному заявлению позволяет клиентам запрашивать у сервера использование доступов созданных во вне по отношению к механизму клиента.

Это воплощает в реальность, то что клиент связывается в UNIX сокет(используется ldapi://) и происходит SASL/TLS аутентификация "непрямо" через UNIX сокет.

Я никогда не использовать эту форму аутентификация, поэтому я не могу что-то про него рассказать, но я думаю это полезно, в качестве `sidecar` когда вы подключаетесь своим сайдкаром к процессу через UNIX socket в котором ваш процесс сайдкара обрабатывает LDAP аутентификацию(через коммуникацию) от вашего имени.

LDAP CRUD

Теперь, то что мы подключились и аутентифицировались мы можем навредить. Если используемый аккаунт имеет верные доступы, вы можете начать добавлять, изменять, искать и удалять LDAP записи. Давайте посмотрим, на каждую из них в отдельности.

В общем, мы будем работать с тремя базовыми записями: groups, users и машины.

Добавление и изменение

Вы можете создать новую LDAP запись используя `Add` функцию. Она принимает простой параметр `AddRequest`. Вы можете собрать `AddRequest` вручную(структура `AddRequest` экспортируется вместе со всеми своими полями) или вы можете использовать простую функцию помощника внутри библиотеку. Мы рассмотрим оба этих случая.

Я решил сгруппировать оба примера добавление и изменение, так как они связаны очень тесно, о чем я не догадывался, а вы увидите дальше.

Добавление групп

Добавление групп в AD заставило попотеть меня, чтобы выяснить, но после прочтения различной AD страниц документации я закончил с чем-то таким:

```
// Тут идет код подключения

addReq := ldap.NewAddRequest("CN=testgroup,ou=Groups,dc=example,dc=com", []ldap.Control{})
var attrs []ldap.Attribute
attr := ldap.Attribute{
      Type: "objectClass",
      Vals: []string{"top", "group"},
}
attrs = append(attrs, attr)

attr = ldap.Attribute{
```

```
	Type: "name",
	Vals: []string{"testgroup"},
}
attrs = append(attrs, attr)

attr = ldap.Attribute{
	Type: "sAMAccountName",
	Vals: []string{"testgroup"},
}
attrs = append(attrs, attr)

// Делаем группу доступной для изменения
// https://docs.microsoft.com/en-us/windows/win32/adschema/a-instancetype
instanceType := 0x00000004
attr = ldap.Attribute{
	Type: "instanceType",
	Vals: []string{fmt.Sprintf("%d", instanceType},
}
attrs = append(attrs, attr)

// делаем группе домен local и то что это будет группа безопасности
// https://docs.microsoft.com/en-us/windows/win32/adschema/a-grouptype
groupType := 0x00000004 | 0x80000000
attr = ldap.Attribute{
	Type: "groupType",
	Vals: []string{fmt.Sprintf("%d", groupType)},
}
attrs = append(attrs, attr)

addReq.Attributes = attrs

if err := l.AddRequest(addReq); err != nil {
	log.Fatal("error adding group:", addReq, err)
}
```

Теперь этот код выглядит более понятно. Есть более краткий метод делать тоже самое, но я хотел показать код выше для краткости, так как это был мой первоначальный код.

Сопособ пол учше, который делает тоже самое:

```
// Тут идет код подключения

addReq := ldp.NewAddRequest("CN=testgroup,ou=Groups,dc=example,dc=com", []ldp.Control{})

addReq.Attribute("objectClass", []string{"top", "group"})
addReq.Attribute("name", []string{"testgroup"})
addReq.Attribute("sAMAccountName", []string{"testgroup"})
addReq.Attribute("instanceType", []string{fmt.Sprintf("%d", 0x00000004})
addReq.Attribute("groupType", []string{fmt.Sprintf("%d", 0x00000004 | 0x80000000)})

if err := l.AddRequest(addReq); err != nil {
      log.Fatal("error adding group:", addReq, err)
}
```

Выделим пару вещей. Первое, вам нужно быть уверенными в том, что атрибуты `objectClass` правильного типа(`top` и `group`).

Второе, `instanceType` hex-число выглядит пугающим, но это именно, то что ждет AD если вы хотите создать "writable" то есть изменяемую группу записи.

Наконец, атрибут `groupType` выглдяит даже безумнее! Выходит. что если вы хотите, чтобы группа имела local домен масштаб хотелок так же и это так же группа безопасности(как противовес распределенной группой) вам нужно будет делать побитовые операции для флагов описаны в AD документации.

Здесь и сейчас, начинаем. Вы можете проверить, что группа создана используя знакомую ldap команду.

```
ldapsearch -LLL -o ldif-wrap=no -b "OU=testgroup,OU=Group,dc=example,dc=com" \
         -D "${LDAP_USERNAME_DN}" -w "${LDAP_BIND_PASSWD}" -h "${LDAP_HOST}" \
         '(CN=testgroup)' cn
```

Добавление юзера

Добавление пользователя потребует больше при разработке, и меня осинило, потому что не было достаточно очевидно, как это делать. Лучший способ научиться этому делать конкретные примеры.

Представим, вы хотите создать нового пользователя LDAP и назначить ему пароль. Давайте скажем еще вы не хотите, чтобы пароль имел срок действия. Что я обычно думаю в таком случае это хитростью сделать всё в один простой запрос `AddRequest` похожим образом что был ранее.

Я думаю я найду правильные LDAP атрибуты, собрав их в `AddRequest` и эта работа будет выполнена. Я был ужасно не прав и на это потребовалось некоторое время что бы это понять.

Получается ключ - это раздеть весь процесс на три шага:

- Создать отключенный аккаунт
- Установить пароль для него
- Включить аккаунт

Зная это, результирующий код для первого шага будет довольно прост:

```
// Тут идет подключение

addReq = ldp.NewAddRequest("CN=fooUser,OU=Users,dc=example,dc=com", []ldp.Control{})
addReq.Attribute("objectClass", []string{"top", "organizationalPerson", "user", "person"})
addReq.Attribute("name", []string{"fooUser"})
addReq.Attribute("sAMAccountName", []string{"fooUser"})
addReq.Attribute("userAccountControl", []string{fmt.Sprintf("%d", 0x0202})
addReq.Attribute("instanceType", []string{fmt.Sprintf("%d", 0x00000004})
addReq.Attribute("userPrincipalName", []string{"fooUser@example.com"})
addReq.Attribute("accountExpires", []string{fmt.Sprintf("%d", 0x00000000})

addReq.Attributes = attrs

if err := l.AddRequest(addReq); err != nil {
      log.Fatal("error adding service:", addReq, err)
}
```

Теперь аккаунт который был создан можно перенести в второй шаг: настроить пароль пользователю.

Сервер AD хранит пароль в виде кодировки little-endian UTF16 в base64. К счастью, для меня linux предоставляет несколько удобных утилит которая могут сделать это за нас. Чтобы создать новый пароль правильного формата.

```
echo -n "\"password\"" | iconv -f UTF8 -t UTF16LE | base64 -w 0
```

Теперь вы создали праоль для нового пользователя, время добавить его в LDAP сервер. Это можно сделать изменяя у пользователя `unicodePwd` атрибут. Код ниже показывает как это сделать:

```
// Тут идет подключение

// https://github.com/golang/text
// According to the MS docs the password needs to be enclosed in quotes o_O
utf16 := unicode.UTF16(unicode.LittleEndian, unicode.IgnoreBOM)
pwdEncoded, err := utf16.NewEncoder().String(fmt.Sprintf("%q", userPasswd))
if err != nil {
        log.Fatal(err)
}

modReq := ldap.NewModifyRequest("CN=fooUser,OU=Users,dc=example,dc=com", []ldap.Control{})
modReq.Replace("unicodePwd", []string{pwdEncoded})

if err := l.ModRequest(modReq); err != nil {
      log.Fatal("error setting user password:", modReq, err)
}
```

юникод обрабатывает код который идет из Go текстового пакета.

Наконец нам нужно включить пользователя изменяя атрибут.

```
modReq := ldap.NewModifyRequest("CN=fooUser,OU=Users,dc=example,dc=com", []ldap.Control{})
modReq.Replace("userAccountControl", []string{fmt.Sprintf("%d", 0x0200})

if err := l.ModRequest(modReq); err != nil {
      log.Fatal("error enabling user account:", modReq, err)
}
```

Снова, вы можете легко проверить что был создан пользователь.

```
$ ldapsearch -LLL -o ldif-wrap=no -b "OU=fooUser,OU=Users,dc=example,dc=com" \
         -D "{LDAP_USERNAME_DN}" -w "${LDAP_BIND_PASSWD}" -h "${LDAP_HOST}" \
         '(CN=fooUser)' cn
```

Добавление аккаунтов для машин

Вы можете так же создать machine(aka service) аккаунт в LDAP который часто используется в сопряжении с Kerberos для хранения атрибутов сервиса и выдача доступа различным сервисам и ресурсам.

Учетная запись для машины может быть создана так же как и пользовательские аккаунты, но есть разница.

Необходимо добавить значение `computer` в список значений атрибута `objectClass` - главное отличие. Всё остальное может быть тем же самым. Так же некоторые люди не указывают пароля для учетной записи для машины, поэтому вы можете пропустить эту часть полностью и просто создать новую запись похожим путем как создются группы.

Изменение DN

Иногда нужно переместить LDAP запись между различными OU. Ниже можно взглянуть на код, который это делает.

```
// connect code comes here

// move fooUser from OU=Users to OU=SuperUsers
req := ldap.NewModifyDNRequest("CN=fooUser,OU=Users,DC=example,DC=com", "CN=fooUser", true,
"OU=SuperUsers,DC=example,DC=com")
if err = conn.ModifyDN(req); err != nil {
        log.Fatalf("Failed to modify userDN: %s\n", err)
}
```

Первое - второй параметр и это RDN (относительный DN) как замена полному пути DN. LDAP сервер хранит записи(и другие вещи) иерархическим способом (на самом деле это сложно структурированный граф). Каждая запись существует в строгой организационной

иерархии(возможно проекция реального мира AD). Отсюда позиция записи всегда относительна других DN частей или абсолютно если указано что использовать нужно полный путь DN.

Третий параметр говорит удаленному серверу где он должен удалить оригинальную запись как только она будет уделена. Если мы решим оставить запись после копирования, необходимо укзазать значение `false`. Последний парамтер новый родительский каталог.

Если вы хотите переименовать CN(или какие-то атрибут) ты можешь обойти последний параметр указав его пустрой строкой т.о. код который будет переименовывать пользователя `fooUser` в `barUser` без перемещения его вокруг различных оушек будет выглядит следующим образом:

```
// move fooUser to "OU=SuperUsers,dc=example,dc=com"
req := ldap.NewModifyDNRequest("CN=fooUser,OU=Users,DC=example,DC=com", "CN=barUser", true,
"")
if err = conn.ModifyDN(req); err != nil {
        log.Fatalf("Failed to modify DN: %s\n", err)
}
```

Изменение пароля

Другая вещь которая вам будет нужна это изменение существующего пароля. Чтобы его поменять вам нужно что-то похожее на:

```
passwdModReq := ldap.NewPasswordModifyRequest("", "OldPassword", "NewPassword")
if _, err = l.PasswordModify(passwdModReq); err != nil {
        log.Fatalf("failed to modify password: %v", err)
}
```

Если вы не указалис новый пароль сервер будет генерировать случайный пароль и вернет его вам:

```
passwdModReq := ldap.NewPasswordModifyRequest("", "OldPassword", "")
passwdModResp, err := l.PasswordModify(passwdModReq)
if err != nil {
```

```
        log.Fatalf("failed to change password: %v", err)
}

newPasswd := passwdModResp.GeneratedPassword
log.Printf("New password: %s\n", newPasswd)
```

В отличии от изменения пароля учетной записи при создании, вам не нужно выделать кавычками UTF-16 base64 зашифрованную строку.

Удаление

Удаление LDAP раписи очень проста. Всё что вам нужно создать `DelRequest` предоставив DN записи и затем запутстить команду следующим образом:

```
delReq = ldap.NewDelRequest("CN=fooUser,OU=Users,dc=example,dc=com", []ldap.Control{})

if err := l.Delete(delReq); err != nil {
      log.Fatalf("Error deleting service: %v", err)
}
```

Опять же, вы можете легко проверить используя знакомую команду `ldapsearch` показанную выше.

Запро

Давайте попговорим о запросах LDAP записей и их аттрибутов.

Чтобы запросить LDAP запись вам нужное создать `SearchRequest`, который вы пошлете в LDAP сервер используя функцию.

`SearchRequest` предоставляет различные возможность для настройки запроса, но мы отметим первый 3, которые я нахожу важными:

- BaseDN - поиск DN для записи
- Filter - для отсеивания результатов

- Attributes - параметры которые вам интересны. Давайте взглянем на конкретном примере и посмотрим на детали:

```
// код подключения

user := "fooUser"
baseDN := "DC=example,DC=com"
filter := fmt.Sprintf("(CN=%s)", ldap.EscapeFilter(user))

// Filters must start and finish with ()!
searchReq := ldap.NewSearchRequest(baseDN, ldap.ScopeWholeSubtree, 0, 0, 0, false, filter,
[]string{"sAMAccountName"}, []ldap.Control{})

result, err := l.Search(searchReq)
if err != nil {
        return fmt.Errorf("failed to query LDAP: %w", err)
}

log.Println("Got", len(result.Entries), "search results")
```

Начнем с создания нового `SearchRequest` и скормим ему три параметра которые я упомянул ранее. Нужно отметить, что при создании `SearchRequest` есть две вещи.

- Необходим корректный фильтр который будет передаваться в функцию. Запись CN должна быть помещена в скобки `()`. Если вы там не сделаете, то получить ошибку при запуске: LDAP Result Code 201 "Filter Compile Error": ldap: filter does not start with an '('
- При использовании `\` нужно пользоваться `ldap.Escape()` функцией, чтобы исключить случайные ошибки LDAP.

Теперь поговорим об оставшихся параметрах. Используя `ldap.ScopeWholeSubtree` мы говорим LDAP серверу, что хотим искать записи по всему дереву DN.

Есть еще параметры доступные вроде `ldap.ScobeBaseObject` который ищет только внутри `RDN`. Но для этого примера, я хотел показать широкий доменый поиск.

Еще необходимо отметить, что мы передаем срез LDAP параметров в которых мы заинтересованны. Если вы оставите срез аттрибутов пустым поиск вернет все параметры

записей LDAP, который вам понадобится, но я хотел показать как вы можете выбрать список атрибутов. Будте осторожны, размер аттрибутов вы можете запросить их все.

Есть множество других There are plenty of other options to search LDAP at your disposal. Particularly, you should have a look at SearchWithPaging function which as its name suggests lets you page the query results if you expect huge loads of them.

Display results

Now that you know how to query the records you might want to display them in the terminal in some human-readable form. There are two handy functions at your disposal: Print and PrettyPrint.

Personally I think they seem almost the same, though PrettyPrint lets you indent the result(s) so you can see the AD tree structure more clearly. See for yourself the results of using both of the functions:

This is the result of Print():

```
DN: CN=fooUser,OU=Users,DC=example,DC=com
sAMAccountName: [fooUser]
```

This is the result of PrettyPrint(2) (see the attribute 2-space indentation):

```
  DN: CN=fooUser,OU=Users,DC=example,DC=com
    sAMAccountName: [fooUser]
```

Conclusion

We have reached the end of this post. Congrats and thank you if you stayed with me until the end! Hopefully, you learned something new and useful which expands your Go toolbox.

When I started using ldap-go library some things were not quite obvious to me, so hopefully the examples in this blog post help whoever ends up having to interact with AD using Go.

As always, if you have any questions or find any inaccuracies in the post let me know in the comments. Until next time!

- https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
- https://tools.ietf.org/html/rfc4513#section-5.1.2
- https://tools.ietf.org/html/rfc4513#section-6.3.1
- https://tools.ietf.org/html/rfc4422#appendix-A
- https://docs.microsoft.com/en-us/windows/win32/adschema/a-instancetype
- https://docs.microsoft.com/en-us/windows/win32/adschema/a-grouptype
- https://github.com/golang/text
- https://en.wikipedia.org/wiki/Kerberos_(protocol)

Revision #9
Created 2022-03-15 14:51:23 UTC by gasick
Updated 2023-06-23 19:31:45 UTC by gasick