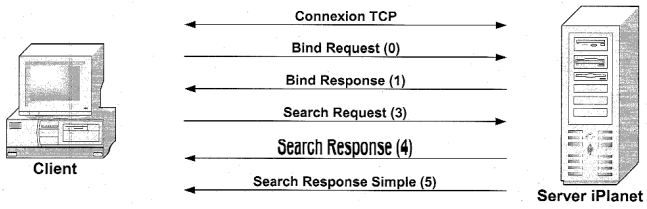
Chapitre 8 : LDAP Protocole

Dans ce chapitre, je vais présenter le protocole LDAP à un niveau beaucoup plus approfondi. Je ne me contenterai pas d'énoncer ces principes comme au chapitre 7, mais j'irai dans l'étude au niveau hexadécimal du protocole.
En effet, la première partie de ce chapitre consistera à comprendre comment le protocole LDAP est transmis. Comme pour TCP ou IP, j'essayerai de faire ressortir la structure de la trame du protocole LDAP.
Par la suite j'étudierai un simple échange LDAP lorsque l'on accède à un annuaire.
Je ferai aussi une étude sur le comportement d'un *referral*.
Pour terminer, je ferai une réplication entre deux serveurs et regarderai toujours au niveau du protocole, ce que cela engendre comme trafic.

Cependant, avant de regarder attentivement les paquets qui sont transmis, regardons comment se passe un échange entre client et serveur LDAP.



Lorsque l'on initialise la connexion avec l'annuaire, une connexion TCP est tout d'abord établie. Ensuite c'est une connexion LDAP qui est établie par le biais des paquets *Bind Request* et *Bind Response*. Une fois connecté, le client fait une recherche (*Search Request*) sur le serveur LDAP et celui-ci lui renvoie les données avec *Search Response*. Une fois que le serveur a envoyé toutes les données, alors celui-ci envoi *Search Response* Simple pour dire qu'il a fini d'émettre.
Le numéro inscrit à côté de ces paquets est le numéro d'application LDAP.
Tout les paquets seront décrits en détail dans les chapitres suivants.

8.1 Structure de la trame du protocole LDAP

Comme énoncé dans le chapitre précédent, le protocole LDAP se situe au dessus de la couche TCP.

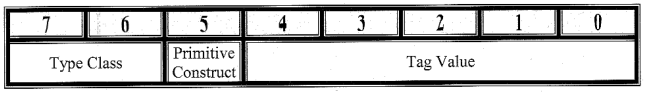
En regardant plus attentivement un paquet LDAP, on s'aperçoit que les données transmisent à l'intérieur de ceux-ci sont en réalité issues de la structure du langage ASN1 (*Abstract Syntax Notation 1*) et ensuite codés en BER (*Basic Encoding Rules*).

Ainsi chaque paquet de type ASN1 (donc chaque paquet LDAP) est composé d'un champ type (*Type*) suivi d'un champ longueur (*Length*), suivi d'un champ contenu (*Content*).

Dans le cas où le champ *Length* est nul (00), le champ Content ne sera pas transmis.

8.1.1 Le champ Type

Le champ *Type* est défini sur 8 bits et de la manière suivante :



Les deux premiers bits de ce champ type contiennent des informations sur le type des informations qui seront transmises. Voici les différents cas possibles :

Bit N°7	Bit N°6	Type Class	Commentaires
0	0	Universal	Ce type de classe peut être Primitif ou construit. C'est ici que l'on défini le type de données à transmettre.
0	1	Application	Ce type est spécifique à une application. Dans mon cas au serveur iPlanet.

Ces deux types sont les plus répandus dans le standard LDAP. Cependant, il en existe encore deux qui sont plus spécifiques aux programmeurs d'une entreprise.

Bit N°7	Bit N°6	Type Class	Commentaires
1	0	Private	Ce type est spécifique à une entreprise. Application entièrement privée.
1	1	Context Specific	Ce type permet d'ajouter une application de type privé.

Le type de classe *Universal* est le seul à être normalisé. Dans le tableau ci-dessous, je vais énoncés quelque unes des valeurs possibles. Pour voir toutes les valeurs possibles, consulter le site http://people.ne.mediaone.net/wasser/ASN1/BasicEncodingRules.html Suivant les types de ces valeurs, ceux-ci seront alors définis comme primitifs (0) ou construits (1). Cela affectera la valeur du bit N°5 du champ *Type*.

Un type primitif contient une seule valeur dans son champ *Content*, tandis que le type Construit, peut contenir plusieurs valeurs à la suite.

Type Tag	Primitif construit	Type des données
Universal 1	Primitif	Boolean
Universal 2	Primitif	Integer
Universal 3	Primitif	Bit String
Universal 4	Primitif	Octet String
Universal 5	Primitif	Null
Universal 6	Primitif	Object Identifier (OID)
Universal 10	Primitif	Enumerated
Universal 16	Construit	Sequence
Universal 17	Construit	Set

Il nous reste maintenant, à définir la valeur du champ *Tag Value*. Cette valeur est simplement la valeur soit *universal* soit de l'application utilisée. Par exemple, si je veux faire une application LDAP de type *Search Request* ces 8 bits seront :

7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	1
Type Application		Primitif	Application N°3			Search Request	

Avec le tableau de la page précédente, nous voyons que LDAP à défini des numéros d'applications pour chacune de ses requêtes ou réponses.
En effet, voici les commandes LDAP les plus courantes.

Voir RFC 1777 (LDAP Version 2) et RFC 2251 (LDAP Version 3)

Numéro de l'application	Nom de l'application
0	Bind Request
1	Bind Response
2	Unbind Request
3	Search Request
4	Search Response
5	Search Response Simple
6	Modifiy Request
7	Modify Response
8	Add Request
9	Add Response
10	Del Request
11	Del Response
14	Compare Request
15	Compare Response
16	Abandon Request
19	Search Response Reference
23	Extended Request
24	Extended Response

8.1.2 Le champ *Length*

le second champ présent dans une structure de type ASN1 est le champ *Length*. Celui-ci à priori nous indique simplement la longueur des données du champ *Content*.
Cependant, si ce champ est plus petit que 128 bits, alors la valeur hexadécimale du champ *Length* sera de la longueur du champ Content. Si le nombre de bits dépassent cette valeur alors le champ *Length* indiquera 81 en hexadécimal et la longueur du champ content sera définie dans l'octet suivant. Il se peut cependant que le champ Content soit d'une longueur indéfinie. Dans ce cas le champ *Length* indiquera toujours la valeur 80 en hexadécimale. Ce cas peut se produire lorsque l'on transmet le sommet d'un arbre sans savoir qui se trouve en dessous ou lorsque le décodage n'arrive pas à ce faire.

Length ≤ 128 Bits	Codage normal en hexadécimal
Length > 128 bits	Valeur de codage = 81 en hexadécimal et suite du codage de la longueur sur le prochain octet.
Length > 256 Bits	Valeur de codage = 82 en hexadécimal et suite du codage de la longueur sur les deux prochains octets.

8.1.3 Le champ *Content*

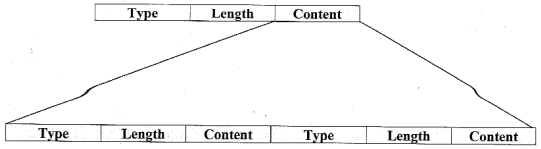
C'est le champ où les données sont transmises réellement. La structure de ce champ est la suivante :

Si l'on a le type primitif comme *Type Class*, alors dans le champ *Content*, il y aura uniquement, la donnée correspondant à ce type.
Au contraire, si on a le type construit, cela veut dire que dans ce champ, plusieurs valeurs seront transmises.

Ainsi avec le type primitif on a :

Type	Length	Content

Tandis qu'avec le type construit on aura :



Grâce à cette encapsulation, plusieurs données de même type (*Application* ou *Universal*), pourront être insérées l'une derrière l'autre. Ceci aura pour effet, un gain de temps et de trafic.

Maintenant que nous avons décrit toutes les possibilités qu'offre LDAP au niveau de son protocole, il reste à étudier un paquet pour que l'explication soit plus parlante.

Pour cela j'ai choisi un paquet *Search Response* que je détaillerai de fond en comble. Il faut bien se rendre compte que la structure de ce paquet est la même que pour un autre. C'est pourquoi je n'en étudierai qu'un seul.

8.2 Etude détaillée du paquet *Search Response*

Ce paquet est la réponse au paquet *Search Request*.
Nous avons dans ce paquet toutes les données transmises par le serveur lors d'un accès à distance sur l'annuaire.

Voici ce que l'analyseur nous donne :
(Je ne détaillerai que le protocole LDAP)

Source Adresse	**Destination Adresse**
Annuaire LDAP 129.194.187.52	Client LDAP 129.194.184.203

TCP :
Length 96 bytes
Source port : 389
Destination port : 4300

LDAP : ProtocolOP : SearchResponse (4)
LDAP : Message ID
LDAP : ProtocolOp = SearchResponse
LDAP : Object Name = cn=te3,ou=telecommunication,o=filiere,o=eig
LDAP : Attribute Type = objectclass
LDAP : Attribute Value = top
LDAP : Attribute Value = groupofuniquenames

Maintenant, si l'on regarde ce qui correspond en hexadécimal, nous aurons :

LDAP : ProtocolOP : SearchResponse (4)
30 5E 02 01
LDAP : Message ID
04
LDAP : ProtocolOp = SearchResponse
64 59 04 2B
LDAP : Object Name = cn=te3,ou=telecommunication,o=filiere,o=eig
63 6E 3D 54 45 33 2C 6F 75 3D 74 65 6C 65 63 6F 6D 6D 75 6E 69 63 61 74 69 6F 6E 2C 6F 3D 66 69 6C 69 65 72 65 2C 6F 3D 65 69 67
LDAP : Attribute Type = objectclass
30 2A 30 28 04 0B 6F 62 6A 65 63 74 63 6C 61 73 73
LDAP : Attribute Value = top
31 19 04 03 74 6F 70
LDAP : Attribute Value = groupofuniquenames
04 12 67 72 6F 75 70 6F 66 75 6E 69 71 75 65 6E 61 6D 65 73

Pour comprendre à quoi correspondent exactement ces valeurs hexadécimales représentées au dessous des champs LDAP, se référer aux pages suivantes qui montrent dans le détail, quelles sont les informations contenues dans chaque paquet.
La toute première valeur en hexadécimal est de 30. Ce n'est pas un hasard, c'est comme cela que l'on reconnaît que les données qui suivent sont du type ASN1.

Regardons plus en détail ces parties de paquets. Il faut toujours tenir d'œil, la forme d'une trame : *Type, Lenght, Content*

LDAP : ProtocolOP : SearchResponse (4)
30 5E 02 01

LDAP : Message ID
04

Structure de la trame		**Data en hexadécimal**	**Décodage**
Type		30	00 1 1000 Univeral (00) de type construit (1) et de type Sequence (10000 = 16)
Length		5E	Length = 94 bytes restant à transmettre
Content	**Type**	02	Donnée de type Integer
	Length	01	De longueur = 1 byte
	Content	04	Message ID

LDAP : ProtocolOp = SearchResponse
64 59 04 2B

LDAP : Object Name = cn=te3,ou=telecommunication,o=filiere,o=eig
63 6E 3D 54 45 33 2C 6F 75 3D 74 65 6C 65 63 6F 6D 6D 75 6E 69 63
61 74 69 6F 6E 2C 6F 3D 66 69 6C 69 65 72 65 2C 6F 3D 65 69 67

Structure de la trame		**Data en hexadécimal**	**Décodage**
Type		64	01 1 00100 Application (01) de type construit (1) et le choix N°4= SearchResponse
Length		59	Length = 89 bytes restant à transmettre
Content	**Type**	04	Donnée de type Octet String
	Length	2B	De longueur = 43 bytes
	Content	63 6E 3D 54 45 33 2C 6F 75 3D 74 65 6C 65 63 6F 6D 6D 75 6E 69 63 61 74 69 6F 6E 2C 6F 3D 66 69 6C 69 65 72 65 2C 6F 3D 65 69 67	cn=te3,ou=telecommunication,o=filiere,o=eig

LDAP : Attribute Type = objectclass
30 2A 30 28 04 0B 6F 62 6A 65 63 74 63 6C 61 73 73

Structure de la trame			**Data en hexadécimal**	**Décodage**
Type			30	00 1 1000 Univeral (00) de type construit (1) et de type Sequence (10000 = 16)
Length			2A	Length = 42 bytes restant à transmettre
Content	**Type**		30	00 1 1000 Univeral (00) de type construit (1) et de type Sequence (10000 = 16)
	Length		28	De longueur = 40 bytes
	Content	**Type**	04	Donnée de type Octet String
		Length	0B	De longueur = 11 bytes
		Content	6F 62 6A 65 63 74 63 6C 61 73 73	objectclass

LDAP : Attribute Value = top
31 19 04 03 74 6F 70
LDAP : Attribute Value = groupofuniquenames
04 12 67 72 6F 75 70 6F 66 75 6E 69 71 75 65 6E 61 6D 65 73

Structure de la trame		**Data en hexadécimal**	**Décodage**
Type		31	00 1 10001 Universal (00) de type construit (1) et de type Set (10001 = 17)
Length		19	Length = 19 bytes restant à transmettre
Content	**Type**	04	Donnée de type Octet String
	Length	03	De longueur = 3 bytes
	Content	74 6F 70	TOP
Content	**Type**	04	Donnée de type Octet String
	Length	12	De longueur = 3 bytes
	Content	67 72 6F 75 70 6F 66 75 6E 69 71 75 65 6E 61 6D 65 73	groupofuniquenames

A l'aide de ces captures, on remarque bien que la suite *Type*, *Length*, *Content* est bel et bien respectée. Ensuite suivant le type de données à transmettre, le codage va être différent (Application Search Request(63), Application *Search Response* (65), etc…). De plus lorsque l'on veut transmettre plusieurs données de même type, il n'est pas nécessaire de renvoyer un nouveau paquet entier avec le champ *type = universal* ou *Application* par exemple. C'est le cas dans le dernier paquet traité.