

1. What is LDAP?

LDAP (Lightweight Directory Access Protocol) is a protocol used to access and manage directory information services. Think of it not as a relational database (tables/rows), but as a **hierarchical tree** (like a file system).

Key Terminology

- **DIT (Directory Information Tree):** The entire hierarchy of data.
- **Entry:** An object in the tree (e.g., a specific user, a printer, a group).
- **DN (Distinguished Name):** The unique full path to an entry.
 - *Example:* `cn=admin,ou=users,dc=company,dc=com`
 - *Analogy:* Like an absolute file path: `/com/company/users/admin`.
- **RDN (Relative Distinguished Name):** The specific name of the entry itself.
 - *Example:* `cn=admin`
- **Attributes:** Data stored inside an entry (e.g., `mail`, `telephoneNumber`, `userPassword`).

Common Attributes

- `cn` = Common Name (often the Full Name or User ID).
 - `ou` = Organizational Unit (like a folder/department).
 - `dc` = Domain Component (parts of the URL, e.g., `google.com` -> `dc=google,dc=com`).
 - `sn` = Surname.
 - `objectClass` = Defines what type of object this is (e.g., `person`, `printer`).
-

2. LDAP Query Syntax

LDAP uses **Prefix Notation** (Polish Notation). Operators come *before* the operands.

Operator	Symbol	Syntax Example	Meaning
Equality	=	(<code>cn=Babs</code>)	<code>cn</code> is equal to "Babs".
AND	&	(<code>&(cn=Babs)(job=Dev)</code>)	<code>cn</code> is "Babs" AND <code>job</code> is "Dev".
OR	,	,	`(
NOT	!	(<code>!(cn=Tim)</code>)	<code>cn</code> is NOT "Tim".
Wildcard	*	(<code>cn=Ba*</code>)	<code>cn</code> starts with "Ba".

Complex Filter Example

A typical login search filter looks like this:

```
(&(cn=USER_INPUT)(userPassword=PASSWORD_INPUT))
```

- **Logic:** "Find an entry where `cn` matches the user input **AND** `userPassword` matches the password input."

3. The Injection Vulnerability

LDAP Injection occurs when user input is concatenated directly into the LDAP filter string without sanitization. This allows an attacker to manipulate the **Logic Tree**.

Scenario A: Authentication Bypass

Target Filter: `(&(cn=USER)(userPassword=PASS))` **Goal:** Log in as `admin` without the password.

Injection:

- Input `USER`: `admin)()|(&` (This is garbage designed to close the loop) -> *Actually, let's look at the classic bypass.*
- Input `USER`: `admin))|((cn=*` -> *Wait, simpler.*
- Input `USER`: `admin`
- Input `PASS`: `*`
- **Result:** `(&(cn=admin)(userPassword=*))` -> "User is admin AND Password is *anything*." (True).

The "Null Byte" / Attribute Injection:

- Input `USER`: `admin)%00`
- **Result:** `(&(cn=admin))%00(userPassword=...)`
- Some LDAP servers stop processing after a NULL byte or closed parenthesis. The filter becomes just `(cn=admin)`, checking only if the user **exists**, ignoring the password entirely.

Scenario B: Blind LDAP Injection

If the application suppresses errors, you use Boolean logic to ask True/False questions.

- **Question:** "Does the admin's telephone number start with 5?"
- **Payload:** `admin)(telephoneNumber=5*`
- **Resulting Filter:** `(&(cn=admin)(telephoneNumber=5*))(userPassword=...)`

- **Logic:** If the login succeeds (or response differs), the phone number starts with 5.
-

4. Discovery & Exploitation Workflow

1. **Fuzzing:** Input characters like `*`, `(`, `)`, `&`, `|` into login forms or search boxes.
 - *Indicator:* Application hangs, returns "syntax error," or displays strange results (all users).
 2. **Bypass:** Attempt to close the filter early to ignore password checks.
 3. **Inference:** Use `*` to extract hidden attributes (e.g., `(attribute=*)`).
 4. **Blind Extraction:** Brute-force sensitive data character by character using wildcards (`a*`, `b*`, `c*`).
-