# LDAP Injection

Lightweight Directory Access Protocol Injection is a type of injection attack where an attacker inserts **malicious LDAP statements** into an application's **query** to **interact with a directory service** (e.g., Active Directory, OpenLDAP) in unintended ways.  
  
The vulnerability occurs when:

* User input is **unsanitized**
* Input is concatenated directly into LDAP search filters or bind queries

This allows attackers to **bypass authentication**, **extract sensitive directory info**, or even **modify LDAP data**.

# Used for:

* **User authentication**
* **Authorization**
* **Storing user/group/org data**
* **Querying directory services**

Typical LDAP filter: (&(uid=ab)(userPassword=secret))  
If input is not sanitized: (&(uid=\*)(userPassword=\*))  
**Wildcard matches all users** = **bypass login**

# Types of LDAP Injection

## Authentication Bypass

Original Filter: (&(uid=admin)(userPassword=$input))

Attacker Input: \*)(&

Resulting Filter: (&(uid=admin)(userPassword=\*)(&))

May bypass password check or cause logic confusion

## Authorization Manipulation

If app fetches user role based on uid, attacker can manipulate input to escalate privileges: (uid=admin)(role=\*)

## 3. Information Disclosure

By injecting wildcards or logical operators, attacker can extract:

* Group memberships
* Email addresses
* Organizational structure
* Payload: \*)(objectClass=\*)

## 4. Blind LDAP Injection

If errors are suppressed, attacker uses **true/false conditions** and **timing** to infer data:

* Try different payloads
* Observe response length, delay, status

# Impact

| Impact Type | Description |
| --- | --- |
| Authentication Bypass | Login as any user |
| Privilege Escalation | Gain admin role |
| Directory Dumping | Enumerate all users, groups |
| Information Leakage | Internal structure of org exposed |
| Denial of Service | Cause crashes or performance degradation |

# Payload Examples

| Scenario | Payload |
| --- | --- |
| Bypass Auth | \*)(uid=\*) |
| Dump Users | \* or `( |
| Escalate Priv | \*)(admin=TRUE) |
| Blind | \*)(!(cn=admin)) (see if result changes) |

# Tools

| Tool | Description |
| --- | --- |
| Burp Suite | Intercept login and inject payloads in username, password |
| Manual Fuzzing | Try payloads in login and search forms |
| LDAPMap | Automated LDAP injection tester |
| Error Observation | Look for verbose LDAP error messages |
| Boolean-Based Tests | Inject filters like: |

)(uid=admin)  
 )(cn=\*)

# Example

**Login Bypass Test**

**Request (POST):**

{

"username": "\*)(uid=\*))(|(uid=\*",

"password": "any"

}

LDAP Filter becomes: (&(uid=\*)(uid=\*))(|(uid=\*))(userPassword=any)  
May return all users = **auth bypass**.

# Mitigation

## 1. Input Sanitization / Escaping

Escape special LDAP characters:  
\*, (, ), \, /, &, |, !, =, ~

**Example in Java: String safeInput = StringEscapeUtils.escapeLdap(input);**

## 2. Use Parameterized LDAP APIs

Avoid string concatenation. Use APIs like:

* DirContext.search() (Java JNDI with controlled filters)
* LDAP libraries that support safe query composition

## 3. Avoid Exposing Raw LDAP Errors

Catch and sanitize error messages before showing them to users.

## 4. Implement Access Controls

Ensure LDAP doesn't return more data than necessary:

* Least privilege for app bind account
* Restrict access to sensitive attributes

## 5. Use Allowlist Filters

Define strict patterns or schemas for allowed values in LDAP queries.

## 6. Logging & Alerting

Log failed queries, especially those with unusual characters or filter logic.

# Secure Coding Example

Insecure: String filter = "(uid=" + userInput + ")";  
Secure: String safeUser = LdapEscaper.escape(userInput);  
 String filter = "(uid=" + safeUser + ")";  
OR USE  
SearchControls controls = new SearchControls();

ctx.search("ou=users,dc=org", "uid={0}", new Object[]{userInput}, controls);

# Points

“LDAP Injection is dangerous in enterprise environments because **Active Directory is everywhere**, and unauthenticated access could reveal internal structure.”

“It’s similar to SQL injection — but instead of databases, exploit **directory services and filters**.”

“Test this by injecting wildcards or boolean logic into LDAP filters and watching for changes in the query response.”

# Real-World Examples

| System | Impact |
| --- | --- |
| Zimbra (CVE-2019-9670) | LDAP injection → RCE |
| SAP / Oracle apps | LDAP injection in SSO integration |
| Enterprise HR platforms | Dumped employee records using wildcard filter injection |