Module 15 - Attacking LDAP Based Implementation

▼ What is LDAP

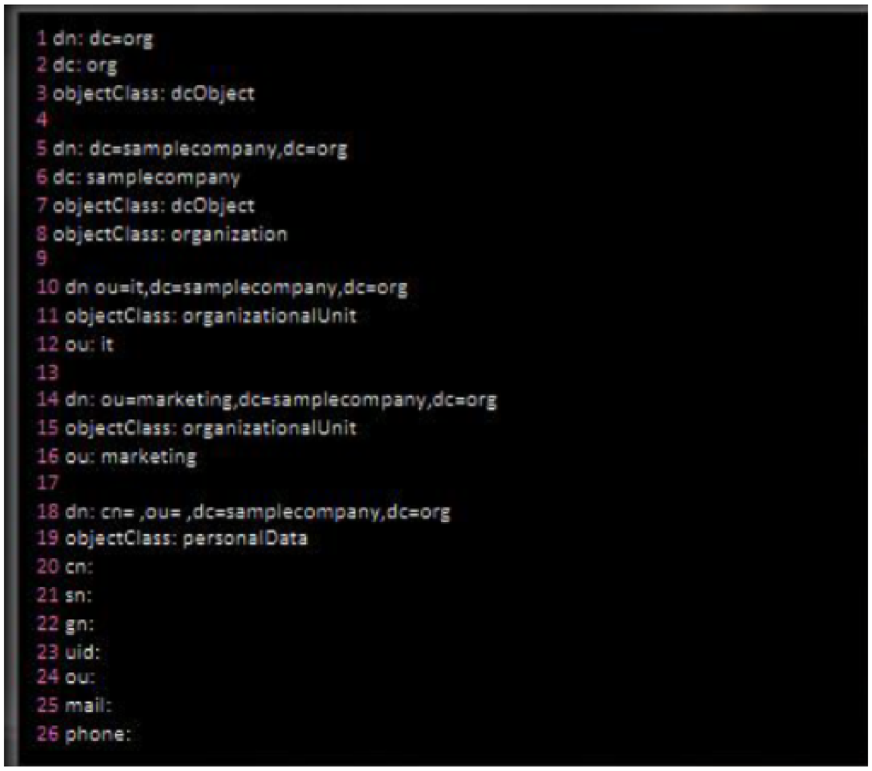
LDAP stands for ***Lightweight Directory Access Protocol***

- used to modify and query directory services over tcp/ip
- Directory service is a database-like virtual storage that holds data in specific hierarchical structure.
- LDAP structure is based on a tree of directory of entries.
- LDAP is used to communicate with Directory Databases, but as a protocol, it does not provide any storage capabilities.
- Sample databases that use directory structure are Microsoft Active Directory (where LDAP is often used in authentication process) or the less known OpenLDAP

▼ Directory Database Structure

- Objects in directory databases accessed via LDAP are stored in LDIF which stands for LDAP Data Interchange Format.
- LDIF defines directory content as a set of records, one record for each object (or entry).
- LDIF also represents update requests, such as Add, Modify, Delete, and Rename, as a set of records, one record for each update request.
- A directory database can support LDIF by defining its assumptions in a LDIF file.
- It can be a plaintext file simply containing directory data representation as well as LDAP commands.
- They are also used to read, write, and update data in a directory.

▼ Sample LDIF File



```
1 dn: dc=org
2 dc: org
3 objectClass: dcObject
4
5 dn: dc=samplecompany,dc=org
6 dc: samplecompany
7 objectClass: dcObject
8 objectClass: organization
9
10 dn ou=it,dc=samplecompany,dc=org
11 objectClass: organizationalUnit
12 ou: it
13
14 dn: ou=marketing,dc=samplecompany,dc=org
15 objectClass: organizationalUnit
16 ou: marketing
17
18 dn: cn= ,ou= ,dc=samplecompany,dc=org
19 objectClass: personalData
20 cn:
21 sn:
22 gn:
23 uid:
24 ou:
25 mail:
26 phone:
```

- Lines 1-3: We are defining the top-level domain "org".
- Lines 5-8: Next, we are defining the subdomain "samplecompany", for example, "samplecompany.org".
- Lines 10-16: We define two organization units (ou): it and marketing.
- Lines 18-26: We then add objects to the domain "samplecompany.org" and assign attributes with values. For example, "sn" stands for surname, "cn" stands for canonical name (or first name), while "mail" is a placeholder for an email address.

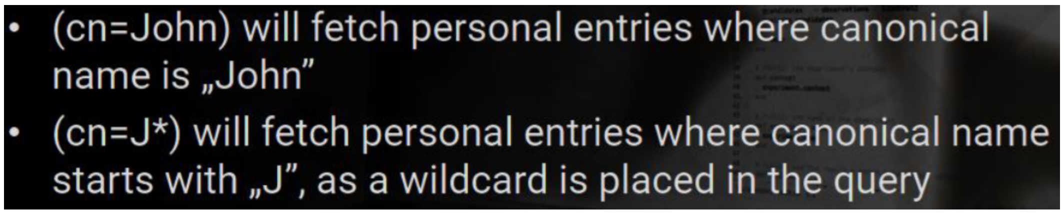
- Each directory services database might have different default attributes.
- For example, in ***OpenLDAP*** implementations you can find a userPassword attribute (which can be interesting from a penetration tester's standpoint) while there's no such attribute in ***Active Directory***.

▼ LDAP Syntax

LDAP as a protocol has its own structure for querying the back-end database. It utilizes operators like the following:

- `=` (equal to)
- `|` (logical or)
- `!` (logical not)
- `&` (logical and)
- `*` (wildcard) – stands for any string or character

Eg:



- (cn=John) will fetch personal entries where canonical name is „John"
- (cn=J*) will fetch personal entries where canonical name starts with „J", as a wildcard is placed in the query

- LDAP query expressions can also be concatenated, resulting in a sample query like the one below:
    - `(|(sn=a*)(cn=b*))`
- In this case, the first **OR** operator is used in order to indicate that we either look for all records which surname starts with "a" **or** canonical name starts with "b".

LDAP Implementations

- The LDAP as a protocol can be a completely independent implementation from the underlying database.
- With that said, we can, for example, configure a web application to serve as a front-end to an Active Directory database.
- In turn, that means that it is possible to use Active Directory (or another directory-based database) with LDAP in order to authenticate web application users.
- This is a convenient method since some roles or user attributes will be shared with domain users, which can be then used for authorization purposes within a web application.

        - This way, a web application can rely on LDAP and the backed directory role attributes when authorizing users to access certain resources.
        - Of course, LDAP can be encountered as a database holding different information, which can include employee data or user account attributes; consider a web interface that can be used to browse employee structure in the company.

Abusing LDAP

- When referring to "abusing" or "exploiting" LDAP in this module, we talk about web-based LDAP implementations.
- However, there could be literally any front-end facade to the LDAP enabled directory database.
- You can often find LDAP services during the scanning of network infrastructure on ***default ports 389*** (for unencrypted connections) or ***636 for LDAP SSL***.

LDAP over TCP

- In order to connect to standalone LDAP services via pure TCP protocol, you can use a tool named JXplorer.
    - `java -jar JXplorer.jar`
- LDAP can be integrated with a web application, which can take user input and implement it into an LDAP query.
- If there is no sanitization of user input, several things can go wrong.

LDAP Vulnerabilities

- LDAP injection
- can inject a wildcard instead of legitimate query to pull all the existing objects.
    - it can lead to DOS bcuz of the huge data requested in case of large databases.
- https://www.sonarsource.com/blog/joomla-takeover-in-20-seconds-with-ldap-injection-cve-2017-14596/

- https://community.f5.com/kb/technicalarticles/joomla-ldap-injection-vulnerability-cve-2017-14596/282973

LDAP Injection

Eg:

- Web app lists LDAP printers without error messages.
- Utilizes filter: `(& (objectclass=printer)(type=Canon*))`.
- Shows icons if Canon printers available; otherwise, none appear.
- Illustrates a true/false scenario.

Blind LDAP Injection:

- Injecting `)(&(objectClass=*))(&(objectClass=void)` alters the query.
    - Resulting query: `(& (objectClass=*) (objectClass=*)) (& objectClass=void) (type=Canon*)`.
    - Only processes the first LDAP query.
    - Result: `(& (objectClass=*) (objectClass=*))` extracted from the backend.
- In case of OR |
    - Technique for gathering information.
    - Queries:
        - `(|(objectClass=void)(objectClass=users))(&(objectClass=void)(type=Canon*))`
        - `(|(objectClass=void)(objectClass=resources))(&(objectClass=void)(type=Canon*))`
    - Purpose: Enumerate the directory structure.

LDAP Python Implementation