**What is LDAP Injection?**

LDAP injection is a type of attack that target vulnerabilities in the implementation of the Lightweight Directory Access Protocol (LDAP). LDAP injection attacks occur when an attacker can manipulate an LDAP query sent to the directory server. This allows the attacker to gain unauthorized access to sensitive information, such as user credentials and other directory data, or even take control of the entire directory server.

Before moving to the LDAP injection, let’s understand a few basic LDAP queries:

An**LDAP query** is a request sent to an LDAP directory server to retrieve or modify information. Queries are typically written in LDAP syntax, a standardized format for specifying search criteria and other options.

Here are some examples of LDAP queries:

To search for all users in the directory with a given name:

(&(objectClass=user)(cn=John))

John represents the name to be searched in the above LDAP query.

To search for all groups in the directory with a given name:

(&(objectClass=group)(cn=Marketing))

Marketing represents the name to be searched in the above LDAP query.

To search for all users in a given group –

(&(objectClass=user)(memberOf=cn=Marketing,ou=Groups,dc=example,dc=com))

This attack is similar to the [SQL Injection](https://www.cobalt.io/blog/a-pentesters-guide-to-sql-injection-sqli) attack. The LDAP injection manipulates an LDAP query sent to the directory server. The attacker can then use the manipulated query to gain access to the server or retrieve sensitive information from the directory. This can be done through any application or system that uses LDAP by injecting a malicious query into the user input.

Suppose the application does not properly validate and sanitize user input. In that case, an attacker could enter a malicious query in the search form (or any other input field) that would be sent to the LDAP server. For example, an attacker could enter a query in a search form of an application that uses LDAP to search a directory. The attacker could enter a search term of "(objectClass=\*)", which would retrieve all objects in the LDAP directory. This could allow the attacker to access sensitive information or potentially take control of the server.

**Understanding the Different Forms of LDAP Injection**

Let’s understand different forms of LDAP injections using example scenarios.

**1. Authentication Bypass**

This type of LDAP injection attack occurs when an attacker can bypass LDAP authentication by manipulating the user input in the login form.

**Example scenario:**

1. The attacker visits the login page of the LDAP server and enters the following as their username:

\*)(uid=\*

1. In the password field, the attacker enters the following as the password:

\*

1. When the attacker submits the login form, the LDAP server receives a query that looks like this:

`(&(uid=\*)(password=\*))`

In this particular payload, the "&" and "(" characters are used to construct a search filter that will return all entries in the directory server where the "uid" and "password" fields are not empty. This allows the attacker to bypass authentication, as the server will interpret the query as requesting access to all entries in the directory without checking for a valid username and password.

The "uid" and "password" fields are commonly used in LDAP directories to store the user's unique identifier (username) and password. Using the "\*" wildcard character in the query, the attacker can match any value in these fields, allowing them to bypass authentication and gain unauthorized access to the directory server.

**2. Information Disclosure**

In this attack, the attacker uses LDAP injection to access sensitive information, such as user login credentials or other sensitive data stored in the directory.

**Example scenario:**

1. The attacker identifies a website that uses LDAP to store and retrieve information from a directory.
2. The attacker crafts a malicious LDAP query, such as (|(cn=\*)(mail=\*)), and injects it into a search form on the website.
3. The website sends the LDAP query to the directory server.
4. The directory server returns all entries in the directory that match the criteria in the query.

The payload (|(cn=\*)(mail=\*)) is a malicious LDAP query that is used in an LDAP injection attack. It attempts to retrieve all entries in the directory with a common name or an email address. The | symbol is a logical operator that means "or," so the query asks the directory server to return all entries that match either the common name or email address criteria.

The \* symbol is a wildcard character that matches any string of characters, so the query asks the directory server to return all entries with a common name or an email address. This can access sensitive information, such as user names, email addresses, and other personal details.

**3. Denial-of-Service (DOS) Attack**

In this attack, the attacker uses LDAP injection to overload the directory server with malicious queries, causing it to crash or become unresponsive.

**Example scenario:**

1. The attacker identifies a website that uses LDAP to store and retrieve information from a directory.
2. The attacker crafts a malicious LDAP query, such as (|(sn=\*)(givenName=\*))\*, and injects it into a search form on the website.
3. The website sends the LDAP query to the directory server.
4. The directory server attempts to process the query. However, because the query contains many wildcard characters, it becomes very resource-intensive and causes the directory server to crash or become unresponsive.

The payload attempts to retrieve all entries in the directory with a surname or a given name and then multiplies the query by many wildcard characters. This can cause a [denial of service](https://www.cobalt.io/blog/what-is-denial-of-service-attack) (DoS) attack by overloading the directory server with resource-intensive queries.

The | symbol is a logical operator that means "or," so the query asks the directory server to return all entries that match the surname or the given name criteria. The \* symbol is the character that multiplies the query.

The \* symbol is a wildcard character that matches any string of characters, so when it is used to multiply the question, it causes the directory server to return many entries that match the criteria in the query.

This causes a denial of service (DoS) attack by overloading the directory server with resource-intensive queries, causing it to crash or become unresponsive.

**4. Blind LDAP Injection**

Blind LDAP injection is a specific type of LDAP injection attack that occurs when the attacker is not able to see the results in the response of the LDAP query that they have injected. This makes it more difficult for the attacker to determine if the injection was successful, as they must rely on other means (such as sleep) to test it.

There are several ways that an attacker can perform a blind LDAP injection attack. For example, they can use a process of trial and error, where the attacker injects different LDAP queries and see if they receive any different responses from the server. This can be a slow and tedious process, but it can be effective if the attacker has enough time and resources.

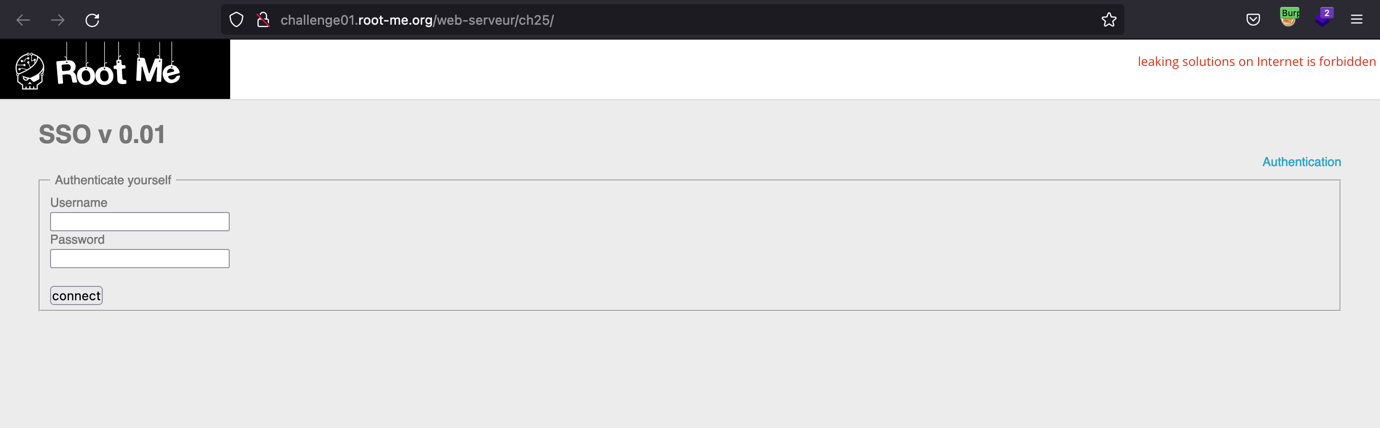
**Understanding LDAP Injection using Practical Demonstration**

We will use root-me.org to understand LDAP injection in depth for the practical demonstration.

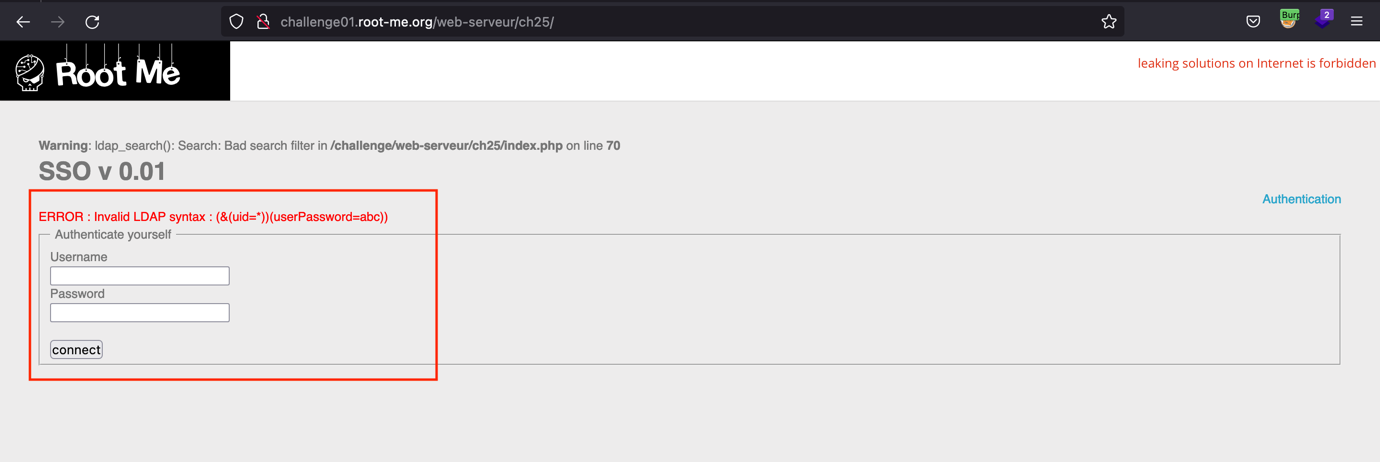
**LDAP Injection – Authentication Bypass**

Performing the attack -

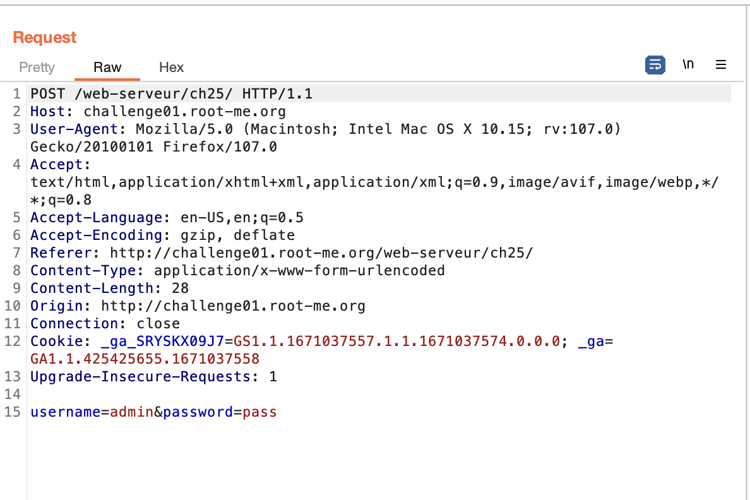
1. Navigate to the login page.



1. Insert the \*) in the Username field to display the error.



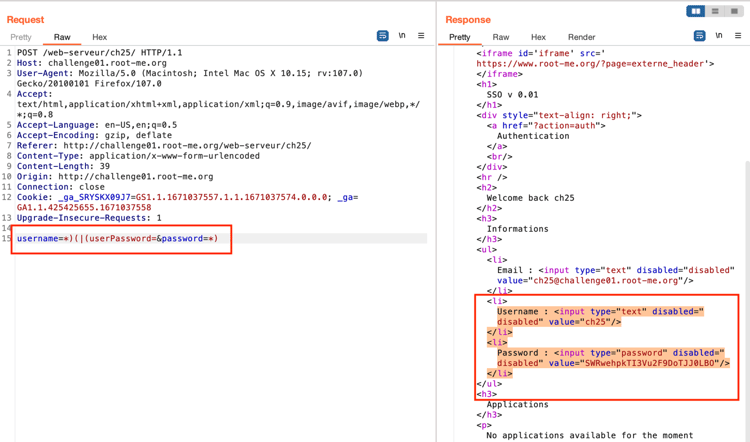
1. The error confirms that the application uses the LDAP server as the directory server.
2. Insert the random values in the Username and Password field, and hit connect while capturing the request in Burpsuite.
3. Send the captured connect request to the repeater tab.



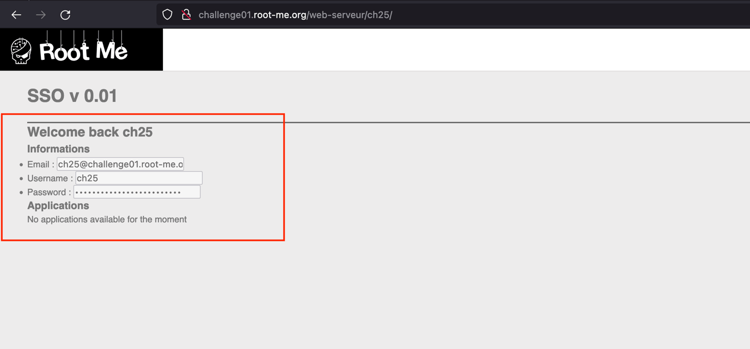
1. In the repeater tab, Insert the malicious payloads in the username and password field:

username=\*)(|(userPassword=&password=\*)

1. Send the request and observe that the Username and Password have been successfully retrieved.



1. Login with the Username and Password to complete the challenge.



LDAP injection is a serious security threat that can expose sensitive user information and compromise the security of an entire organization. Organizations can protect themselves from this attack by implementing the appropriate security measures and safeguarding their sensitive information.

**Remediations**

* Implementing input validation and sanitization to prevent the injection of malicious characters into LDAP queries.
* Enforcing strict access controls and permissions limits unauthorized users' ability to access sensitive information.
* Use a framework such as LINQtoAD. It generates LDAP encoding automatically for LDAP queries.

When creating LDAP queries, avoiding (' or ') is preferable. It will restrict attackers from modifying the content of a query by adding one or more opening-closing brackets.