LDAP Injection Prevention Cheat Sheet¶

cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.html

Introduction¶

This cheatsheet is focused on providing clear, simple, actionable guidance for preventing LDAP Injection flaws in your applications.

LDAP Injection is an attack used to exploit web based applications that construct LDAP statements based on user input. When an application fails to properly sanitize user input, it's possible to modify LDAP statements through techniques similar to SQL Injection.

LDAP injection attacks could result in the granting of permissions to unauthorized queries, and content modification inside the LDAP tree.

For more information on LDAP Injection attacks, visit LDAP injection.

LDAP injection attacks are common due to two factors:

1. The lack of safer, parameterized LDAP query interfaces
2. The widespread use of LDAP to authenticate users to systems.

Primary Defenses:

> Escape all variables using the right LDAP encoding function

Additional Defenses:

> Use a framework (like LINQtoAD) that escapes automatically.

Primary Defenses¶

Defense Option 1: Escape all variables using the right LDAP encoding function¶

The main way LDAP stores names is based on DN (distinguished name). You can think of this like a unique identifier. These are sometimes used to access resources, like a username.

A DN might look like this

```
cn=Richard Feynman, ou=Physics Department, dc=Caltech, dc=edu
```

or

```
uid=inewton, ou=Mathematics Department, dc=Cambridge, dc=com
```

There are certain characters that are considered special characters in a DN.

The exhaustive list is the following: `\ # + < > , ; " =` and leading or trailing spaces.

Some "special" characters that are allowed in Distinguished Names and do not need to be escaped include:

```
* ( ) . & - _ [ ] ` ~ | @ $ % ^ ? : { } ! '
```

Each DN points to exactly 1 entry, which can be thought of sort of like a row in a RDBMS. For each entry, there will be 1 or more attributes which are analogous to RDBMS columns. If you are interested in searching through LDAP for users will certain attributes, you may do so with search filters.

In a search filter, you can use standard boolean logic to get a list of users matching an arbitrary constraint. Search filters are written in Polish notation AKA prefix notation.

Example:

```
(&(ou=Physics)(|
(manager=cn=Freeman Dyson,ou=Physics,dc=Caltech,dc=edu)
(manager=cn=Albert Einstein,ou=Physics,dc=Princeton,dc=edu)
))
```

When building LDAP queries in application code, you MUST escape any untrusted data that is added to any LDAP query. There are two forms of LDAP escaping. Encoding for LDAP Search and Encoding for LDAP DN (distinguished name). The proper escaping depends on whether you are sanitizing input for a search filter, or you are using a DN as a username-like credential for accessing some resource.

Safe Java Escaping Example¶

Safe C Sharp .NET TBA Example¶

.NET AntiXSS (now the Encoder class) has LDAP encoding functions including `Encoder.LdapFilterEncode(string)`, `Encoder.LdapDistinguishedNameEncode(string)` and `Encoder.LdapDistinguishedNameEncode(string, bool, bool)`.

`Encoder.LdapFilterEncode` encodes input according to RFC4515 where unsafe values are converted to `\XX` where `XX` is the representation of the unsafe character.

`Encoder.LdapDistinguishedNameEncode` encodes input according to RFC2253 where unsafe characters are converted to `#XX` where `XX` is the representation of the unsafe character and the comma, plus, quote, slash, less than and great than signs are escaped using slash notation (`\X`). In addition to this a space or octothorpe (`#`) at the beginning of the input string is `\` escaped as is a space at the end of a string.

`LdapDistinguishedNameEncode(string, bool, bool)` is also provided so you may turn off the initial or final character escaping rules, for example if you are concatenating the escaped distinguished name fragment into the midst of a complete distinguished name.

Defense Option 2: Use Frameworks that Automatically Protect from LDAP Injection¶

Safe NET Example

LINQ to Active Directory provides automatic LDAP encoding when building LDAP queries.

Defense Option 3: Additional Defenses¶

Beyond adopting one of the two primary defenses, we also recommend adopting all of these additional defenses in order to provide defense in depth. These additional defenses are:

- **Least Privilege**
- **Allow-List Input Validation**

Least Privilege¶

To minimize the potential damage of a successful LDAP injection attack, you should minimize the privileges assigned to the LDAP binding account in your environment.

Allow-List Input Validation¶

Input validation can be used to detect unauthorized input before it is passed to the LDAP query. For more information please see the Input Validation Cheat Sheet.

Related Articles¶

- OWASP article on LDAP Injection Vulnerabilities.
- OWASP article on Preventing LDAP Injection Prevention.
- OWASP Testing Guide article on how to Test for LDAP Injection Vulnerabilities.