LDAP is a protocol for accessing and modifying the directory services over TCP/IP. This protocol has many implementations. However, we will focus only on the Active Directory Lightweight Directory service in this topic, as it is used by Kentico.

The LDAP injection attacks are similar to SQL injection attacks in principle. The attacker tries to exploit a web application to construct a malicious LDAP statement. If the application does not sanitize the user input, the attacker may be able to execute various commands.

Examples of LDAP injections

Obtaining user information

Original query	http://www.example.com/people_search.aspx?name=Andy)(zone=public)
Malformed query	http://www.example.com/people_search.aspx?name=Andy)(zone=*)

Effect: Using the malformed query, the attacker can obtain all user information of the user Andy, not only the public information as in the original query.

Elevation of privileges

Original query	http://www.example.com/page.aspx?path=Pages&recursive=yes
Malformed query	http://www.example.com/page.aspx?path=Pages|(level=*))&recursive=yes

Effect: Using the malformed query, the attacker can obtain all pages, not only those he has permissions for in the original query.

What can LDAP injection attack do

The attacker can, for example, obtain user logins and employee information, achieve privilege elevation, or modify the content inside the LDAP tree.

Finding LDAP injection vulnerabilities

- Black box testing – try to insert LDAP queries into **sign-in**, **forgotten password**, or **search user** fields. Also, try to insert bogus LDAP queries into these fields (see **Examples of LDAP injections** section).
- Searching in code – try to find usages of the LDAP API in the code (see Directory Services in the .NET Framework for information). Then, make sure that the input passed to the API is sanitized and validated correctly.

How to prevent LDAP injection in Kentico

In Kentico, LDAP is used during the authentication process via forms. The input data is not authenticated against a database, but against Active Directory. Therefore, this spot in the system is vulnerable to LDAP injection attacks. Knowing that, we have made sure the input data is correctly validated to prevent these attacks.

However, if you plan to create a custom web part which operates with the AD data, be sure to check and secure the input before sending the LDAP queries to the server. Good techniques are:

- restrict user input using regular expressions - where a number is expected, validate the input for numbers; where a name is expected, validate for characters.
- escape special characters.

Escaping special characters

The required escape sequence depends on whether the user input is used to create the Distinguished Name or used as a part of the search filter.

Used in DN - escape using / is needed:

&	!	|	=	<	>	,	+	-	"	'	;

Used in filter - escape using {\ASCII} is needed:

(	{\28}
)	{\29}
\	{\5c}
*	{\2a}
/	{\2f}
Null	{\0}

It is also a good practice to include '\\' at the beginning of escaped character listings to prevent recursive replacements.