# LDAP Injection Prevention

After discussing different ways to exploit LDAP injection vulnerabilities, let us discuss how to prevent them.

## General Remarks

While many web developers are aware of SQL injection vulnerabilities due to the common use of SQL databases in web applications, LDAP injection is a much rarer type of vulnerability, and thus there is less awareness about it. Therefore, LDAP injection vulnerabilities potentially exist whenever LDAP integration is used in web applications, even though there are simple countermeasures. To prevent LDAP injection vulnerabilities, the following special characters need to be escaped:

* The parenthesis ( needs to be escaped as \28
* The parenthesis ) needs to be escaped as \29
* The asterisk \* needs to be escaped as \2a
* The backslash \ needs to be escaped as \5c
* The null byte needs to be escaped as \00

## PHP Example

In many languages, there are predefined functions that implement LDAP escaping for us. In PHP, this function is called ldap\_escape. Check out the documentation [here](https://www.php.net/manual/en/function.ldap-escape.php).

As an example, let us consider the following simplified code that is vulnerable to LDAP injection:

Code: php

// ldap connection  
const LDAP\_HOST = "localhost";  
const LDAP\_PORT = 389;  
const LDAP\_DC = "dc=example,dc=htb";  
const LDAP\_DN = "cn=ldapuser,dc=example,dc=htb";  
const LDAP\_PASS = "ldappassword";  
  
// connect to server  
$conn = ldap\_connect(LDAP\_HOST, LDAP\_PORT);  
if (!$conn) {  
 exit('LDAP connection failed');  
}  
  
// bind operation  
ldap\_set\_option($conn, LDAP\_OPT\_PROTOCOL\_VERSION, 3);  
$bind = ldap\_bind($conn, LDAP\_DN, LDAP\_PASS);  
if (!$bind) {  
 exit('LDAP bind failed');  
}  
  
// search operation  
$filter = '(&(cn=' . $\_POST['username'] . ')(userPassword=' . $\_POST['password'] . '))';  
$search = ldap\_search($conn, LDAP\_DC, $filter);  
$entries = ldap\_get\_entries($conn, $search);  
  
if ($entries['count'] > 0) {  
 // successful login  
 <SNIP>  
} else {  
 // login failed  
 <SNIP>  
}

In the search operation, the web application inserts user input without any sanitization, leading to LDAP injection as we have seen and exploited in the last couple of sections. To prevent this, we simply need to call the function ldap\_escape when inserting the user input into the search filter. The corresponding line of code should thus look like this:

Code: php

$filter = '(&(cn=' . ldap\_escape($\_POST['username']) . ')(userPassword=' . ldap\_escape($\_POST['password']) . '))';

## Best Practices

While proper sanitization prevents LDAP injection entirely, there are some further best practices we should follow whenever LDAP is used in a web application. First, we should give the account used to bind to the DS the least privileges required to perform the search operation for our specific task. This limits the amount of data an attacker can access in the event of an LDAP injection vulnerability.

Furthermore, when using LDAP for authentication, it is more secure to perform a bind operation with the credentials provided by the user instead of performing a search operation. Since the DS checks the credentials when performing a bind operation, we delegate the authentication process to the DS to handles it for us. This way, there is no LDAP search filter where LDAP injection can occur. To do this, we need to change our example code above to look like this:

Code: php

// ldap connection  
const LDAP\_HOST = "localhost";  
const LDAP\_PORT = 389;  
const LDAP\_DC = "dc=example,dc=htb";  
  
// user credentials  
$dn = "cn=" . ldap\_escape($\_POST['username'], "", LDAP\_ESCAPE\_DN) . ",dc=example,dc=htb";  
$pw = $\_POST['password'];  
  
// connect to server  
$conn = ldap\_connect(LDAP\_HOST, LDAP\_PORT);  
if (!$conn) {  
 exit('LDAP connection failed');  
}  
  
// bind operation  
ldap\_set\_option($conn, LDAP\_OPT\_PROTOCOL\_VERSION, 3);  
$bind = ldap\_bind($conn, $dn, $pw);  
if ($bind) {  
 // successful login  
 <SNIP>  
} else {  
 // login failed  
 <SNIP>  
}

Lastly, anonymous authentication, also called anonymous binds, should be disabled on the DS so that only authenticated users can perform any operation.