National Cyber Security Centre
Ministry of Security and Justice

Help! My website is vulnerable to SQL injection

Check your website and take precautionary measures

Factsheet FS-2014-05
version 1.0 | 26 January 2015

SQL injection is a popular and frequently used attack on websites, which attackers use to steal large volumes of (client) information. Although there are other types of attacks for capturing this information, SQL injection appears to be a frequently used method.

A website becomes vulnerable to SQL injection when attackers are able to influence the queries sent by a website to a database. This enables the attacker to extract information from the database or to change the contents of the database through, for example, a simple query. In this way, an SQL injection vulnerability can endanger both the integrity as well as the confidentiality of the information behind the website.

This factsheet explains what you must do when your website is vulnerable to SQL injection and an attack has been successful. In addition, the factsheet describes which precautionary measures you can take to ensure that you are safe from SQL injection.

Target audience
This factsheet is geared towards developers and technical administrators of websites. Are you the owner of a vulnerable website, but do you have no knowledge of the technicalities of this website? Send this factsheet to your developer or administrator.

Key facts

- A website becomes vulnerable to SQL injection when user input is used in an unsafe manner in the queries sent by a website to a database.
- SQL injection allows an attacker to extract random information from a database. In certain cases the attacker can also remove information from or simply add information to this database.
- Victims of SQL injection are recommended that they:
  - investigate the damage and cause of the incident;
  - prepare and implement communications about this;
  - take temporary measures during the investigation;
  - restore the integrity of the database;
  - remove the vulnerability by adapting code or installing the latest software patches, and
  - bring the website online again as soon as it is certain that the vulnerability has been effectively repaired.
- SQL injection can be detected by:
  - checking log files;
  - checking the integrity of the database, and
  - monitoring public statements about your website.
- The risk of an SQL injection vulnerability can be minimised by:
  - making use of parameterised queries;
  - normalising, validating and filtering user input for undesirable input;
  - using database accounts with limited user rights;
  - making sensitive information in the database unreadable;
  - installing the latest software updates;
  - being cautious with CMS plug-ins from third parties;
  - regularly carrying out security scans, and
  - allowing for responsible disclosure of any vulnerabilities.

Background

Many websites communicate with underlying databases in a language that is called Structured Query Language (SQL). Websites make use of a database to store all kinds of information. Examples are user names and passwords for closed sections of the website or news items. To enable communication with the database, the website formulates a query to the database as a query in SQL.

What is SQL injection?

SQL injection is the manipulation of the structure of SQL queries which the website sends to the database via illicit user input. Visitors to a website can often influence an SQL query directly or indirectly through, for example, a search term, a form or even the value of a cookie or the string identified by the browser (the 'user agent'). This is not a problem if the visitor can only influence the contents of the query. However, if he can also alter the structure of the query, that will give the visitor unauthorised access to the database.

Suppose that an organisation offers the possibility via a website to search news items based on a search term. For this purpose, the website developer included the following SQL query in the code:

```
SELECT  title, description
FROM    news
WHERE   description LIKE '%search term%'
```

This SQL query finds the title and description of news items in the database. Here, the query restricts the result based on the search term entered by the visitor. The problem, however, is that by manipulating the search term, the visitor can also manipulate the structure of the SQL query. Suppose that a attacker offers the search term '`search term' UNION SELECT user name, password FROM users; --`' to the website. The SQL query would suddenly look like this:

```
SELECT  title, description
FROM    news
WHERE   description LIKE '%search term'
UNION
SELECT  username, password
FROM    users; --%'
```

This SQL query does two things. Firstly, it ensures that news items can still be retrieved. Secondly, it results in the website requesting a list of user names and passwords from another part of the database. In brief, the attacker has ensured that the website eventually sent a different query to the database than intended. So, in this example, it is possible to leak all user names and passwords from the database through a simple query. Depending on the information that your website stores in the database, this could also give attackers access to other information, such as credit card numbers, medical information and documents.

What could happen?

SQL injection offers the possibility to retrieve, manipulate and remove information from the database, and in some cases, add information to it. The preceding paragraph provides an example of this. Additionally, SQL injection can be used to influence the logic of the website, thereby bypassing an authentication mechanism, for example.

How would I know if I have been hacked?

You can find out if your website has suffered an SQL injection attack by monitoring your systems and by monitoring public statements about your website. Furthermore, there's a possibility that an external organisation or developer informs you about the presence of the vulnerability or its misuse. The box 'Detecting SQL injection' contains more information on setting up monitoring mechanisms.

Detecting SQL injection

- *Check log files for suspect messages*
  Check whether the log files of the web server, the website, the application server, the database, the IPS/IDS, or the firewall contain any suspicious items. Examples are:
  - the presence of SQL commands in URLs from the web server log; examples of these commands include SELECT, INSERT, DROP, UNION and UPDATE[1];
  - messages from the IDS about any SQL injection attempts detected, and
  - error messages from the database, such as failed SQL queries carried out by the website.
- *Check the integrity of the database*
  Check if the database contains any suspicious items, such as new, unknown or strange user names or tables.
- *Monitor public statements about your website*
  Make use of mechanisms such as Google Alerts[2] and Twitter searches to ensure that you automatically receive a notification of any statements about your website in combination with e.g. 'dump', 'hack' or 'sql injection'.

What can I do if I have been hacked?

An attack on your website has been successful if information from the database has been accessed by attackers. Of course, it is not possible to undo this, but it is important to take immediate steps to limit the impact of the attack as much as possible and to avoid any future misuse.

In case of an attack or suspected attack based on SQL injection we recommend that the following measures be taken:

1. Investigate whether there was an actual incident. For this, you should use the detection options mentioned in the box 'Detecting SQL injection' and carry out an automated security scan of the website[3] to find any SQL injection vulnerabilities by yourself.

[1] Attention should also be paid to coded versions of these key words, such as '&#68&#82&#79&#80', as a hex-coded version of the key word 'DROP'. See for example: "Character Encoding Calculator" at http://ha.ckers.org/sqlinjection/

[2] https://www.google.com/alerts

[3] There are various tools available to automatically establish whether your website contains an SQL injection vulnerability. A popular example of this is sqlmap (http://sqlmap.org/).

2. Determine the impact of the incident. Did the leak 'only' pertain to e-mail addresses for example, or to passwords and other sensitive information as well?

The previous steps may show that there was an actual incident. If this is the case, you should take the following steps:

3. Decide who to inform, how you will do so, and what you are going to communicate. Examples are internal parties such as the legal department and the help desk, but also external parties such as visitors of your website, clients, suppliers, customers, supervisory authorities and the press.

4. Decide what to do with your website while dealing with the incident. Obviously, you cannot simply keep the website online. Possible temporary solutions are:
   - bringing a temporary static website online as a replacement of the normal website, and
   - deactivating part of the functionality of the website.

   Never simply restore a backup of the entire system or the database. For chances are that this will replace a vulnerable system that will be hacked again afterwards.

5. Restore the integrity of the database if it appears that attackers have not only stolen information from the database, but have also made unauthorised alterations to the database. For example, restore a backup of which you are certain that this backup was made prior to the hack.

6. Repair the vulnerability:

   - If you use a standard software product (like a CMS), you should at least install the latest updates of this software. If this does not solve the problem, you should report this problem to the supplier as soon as possible. The supplier can then release an update later.

   - If you developed the website yourself (or had it developed), you must make adjustments to your website yourself (or have this done for you) in order to remove the vulnerability. Evaluate the entire website, not only the attacked parts. In so doing, you should make use of the measures under the heading 'How to prevent SQL injection in my website?'.

7. Have a thorough penetration test carried out on the website. In any case, you should have the SQL injection vulnerabilities checked out, and preferably other types of vulnerabilities as well. Repair the vulnerabilities found.

8. Restore the functionalities of your website as soon as it is clear that the vulnerabilities have been rectified effectively.

Block and restore access

If the login data of your users were leaked as well, it is important that you do not only notify the users about this, but also prevent any misuse of these data. You should therefore consider blocking user access initially. After having restored the functionalities, you can offer the option to change login data in a safe manner (for example through e-mail verification) to allow users to access your website again.

How to prevent SQL injection in my website?

Of course, it is always better to prevent the presence of any SQL injection vulnerabilities in your website. Apart from measures to prevent SQL injection vulnerabilities, the NCSC 'ICT Security Guidelines for Web Applications' also contain measures for the prevention of all kinds of other vulnerabilities. The below measures, most of which are included in these guidelines, are important to prevent SQL injection vulnerabilities (references to the specific measures in the guidelines are between brackets):

- Make use of parameterised queries when setting up an SQL query (guideline B3-5). This means that an SQL query does not come about by dynamically pasting all kinds of strings together, but by defining a static SQL query and 'pasting' parameters in this query later. All modern programming languages support this concept.

- Normalise and validate all user input (guidelines B3-3 and B3-6). This means that the website checks whether the input it receives corresponds with the expected input. For instance, any (Dutch) postcode entered must always consist of four figures and two letters. The website should not accept any input which is not provided in this format.

- Filter for undesirable input (guideline B3-1). After normalising and validating the input, the website checks whether any undesirable input, e.g. certain 'hazardous' key words, remains. For instance, the use of the words SELECT and DROP in the input could indicate a possible attempt of SQL injection.

- Ensure that the website uses a database account with limited rights (guideline B0-12). If, for example, the website may only consult information and not make any changes, it will not be possible to carry out changes in the database using SQL injection. Take note that these measures have an effect on the integrity of the database but not on the confidentiality.

- Encrypt data in the database (guideline B5-3). Encrypting data in the database or otherwise making them unreadable (hashing) does not reduce the chance of an SQL injection vulnerability, but does reduce the damage any such vulnerability may cause.

List of most important steps after an SQL injection

1. Investigate whether there was an actual incident.
2. Determine the impact of the incident. What information has been leaked?
3. Decide who to inform, how you will do so, and what you are going to communicate.
4. Decide what to do with your website while dealing with the incident. Temporarily switch off functionalities, for example.
5. Restore the integrity of the database.
6. Repair the vulnerability by installing updates or by altering software you developed yourself.
7. Have a thorough penetration test carried out in order to identify any other vulnerabilities in your website.
8. Restore the functionalities of your website as soon as it is clear that the vulnerabilities have been rectified effectively.

» Always install the latest versions of the software used by the website (guideline Bo-7). This not only concerns the web server and the website, but also for example, databases, libraries and plug-ins.

» Be careful with the use of all kinds of plug-ins when you use a content management system (CMS). Third parties (other than CMS developers) often develop all kinds of plug-ins for this kind of systems and do not always pay adequate attention to their security features. Be aware that such plug-ins often also have direct access to the database and can thus introduce an SQL injection vulnerability which could have consequences for the entire website.

» Regularly carry out penetration tests and security scans of the website to identify vulnerabilities (guidelines Bo-8 and Bo-9).

» Finally, allow for responsible disclosure of any vulnerabilities in your website. By publishing a responsible disclosure policy on your website, you will show how people and organisations can report any vulnerabilities such as SQL injection vulnerabilities, to you, and what the process entails after such reporting. You could make use of the NCSC 'Responsible Disclosure Guidelines'[4] when drawing up such a policy.

In conclusion

SQL injection is only one of the vulnerabilities that could be present in a website. A safe website is therefore not only protected against this vulnerability, but also against all kinds of other vulnerabilities. The NCSC has drawn up the "ICT Security Guidelines for Web Applications" (*only available in Dutch*) in order to help organisations with this. More information about these guidelines and its contents can be found on the NCSC website via https://www.ncsc.nl[5].

[4] https://www.ncsc.nl/actueel/nieuwsberichten/leidraad-responsible-disclosure.html
[5] https://www.ncsc.nl/dienstverlening/expertise-advies/kennisdeling/whitepapers/ict-beveiligingsrichtlijnen-voor-webapplicaties.html

Publication by **National Cyber Security Centre**
Turfmarkt 147 | 2511 DP The Hague
PO Box 117 | 2501 CC The Hague
www.ncsc.nl | info@ncsc.nl | T +31 (0)70-751 55 55 | F +31 (0)70-322 25 37
Publication no: FS-2014-05 | No rights can be derived from this information.