



WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikimedia Shop

- Interaction
 - Help
 - About Wikipedia
 - Community portal
 - Recent changes
 - Contact page

- Tools
- Print/export

- Languages
 - العربية
 - Català
 - Čeština
 - Dansk
 - Deutsch
 - Español
 - فارسی
 - Français
 - 한국어
 - Bahasa Indonesia
 - Italiano
 - עברית

Latviešu

- Bahasa Melayu
- Nederlands
- 日本語
- Norsk bokmål
- Polski
- Português
- Русский
- Српски / srpski
- Suomi
- Svenska

Create account Log in

Article Talk Read Edit View history

SQL injection

From Wikipedia, the free encyclopedia

SQL injection is a [code injection](#) technique, used to [attack](#) data driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).^[1] SQL injection must exploit a [security vulnerability](#) in an application's software, for example, when user input is either incorrectly filtered for [string literal escape characters](#) embedded in [SQL](#) statements or user input is not [strongly typed](#) and unexpectedly executed. SQL injection is mostly known as an attack [vector](#) for websites but can be used to attack any type of SQL database.

In a 2012 study, security company [Imperva](#) observed that the average web application received 4 attack campaigns per month, and retailers received twice as many attacks as other industries.^[2]

Contents [hide]

- 1 Forms
- 2 Technical implementations
 - 2.1 Incorrectly filtered escape characters
 - 2.2 Incorrect type handling
 - 2.3 Blind SQL injection
 - 2.3.1 Conditional responses
 - 2.4 Second Order SQL Injection
- 3 Mitigation
 - 3.1 Parameterized statements
 - 3.1.1 Enforcement at the coding level
 - 3.2 Escaping
 - 3.3 Pattern check
 - 3.4 Database permissions
- 4 Examples
- 5 In popular culture
- 7 References
- 8 External links

Classification parameters	Methods	Techniques/Implementations
Intent	Identifying injectable parameters	see "type of attack"
	Escaping Data	
	Adding or Modifying Data	
	Performing Denial of Service	
	Evaluating detection	
	Requesting Authentication	
	Escalating access/privileges	
Input Source	Injection through user input	Malicious strings: URL, GET-Method, as Web form, Input field(s): POST-Method
	Injection through cookies	Modified cookie fields containing SQLiA
	Injection through server variables	Headers are manipulated to contain SQLiA
	Second-order injection	Frequency-based Primary Application
		Frequency-based Secondary Application
Input type of attack, technical aspect	Classic SQLiA	Concatenated Substring Application
		Regex-Based Queries
		Techniques
		Alternate Encodings
		Obfuscate Logically Injected Queries
	Inference	Classic SQLiA
		Conditional Responses
		Conditional Errors
		Denial of Service (DoS)
		Denial of Service (DoS) (Time-of-Check/Time-of-Use)
DBMS specific SQLiA	DBMS specific SQLiA	DBMS specific SQLiA
		DBMS specific SQLiA
		DBMS specific SQLiA
Compound SQLiA	Compound SQLiA	Compound SQLiA
		Compound SQLiA

A Classification of SQL injection attacking vector until 2010.

Forms [edit]

SQL injection (SQLI) is considered one of the top 10 web application vulnerabilities of 2007 and 2010 by the [Open Web Application Security Project](#).^[3] In 2013, SQLI was rated the number one attack on the OWASP top ten.^[4] There are five main sub-classes of SQL injection:

- Classic SQLI
- Blind or Inference SQL injection
- [Database management system](#) -specific SQLI
- Compounded SQLI
 - SQL injection + insufficient authentication ^[5]
 - SQL injection + [DDoS](#) attacks ^[6]^[7]

Українська
Tiếng Việt
中文

 Edit links

- SQL injection + [DNS hijacking](#)
- SQL injection + [XSS](#) ^[8]

The [Storm Worm](#) is one representation of Compounded SQLI. ^[9]

This classification represents the state of SQLI, respecting its evolution until 2010—further refinement is underway. ^[10]

Technical implementations ^[edit]

Incorrectly filtered escape characters ^[edit]

This form of SQL injection occurs when user input is not filtered for [escape characters](#) and is then passed into a SQL statement. This results in the potential manipulation of the statements performed on the database by the end-user of the application.

The following line of code illustrates this vulnerability:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

This SQL code is designed to pull up the records of the specified username from its table of users. However, if the "userName" variable is crafted in a specific way by a malicious user, the SQL statement may do more than the code author intended. For example, setting the "userName" variable as:

```
' or '1'='1
```

or using comments to even block the rest of the query (there are three types of SQL comments):^[11]

```
' or '1'='1' -- '  
' or '1'='1' ( { '  
' or '1'='1' /* '
```

renders one of the following SQL statements by the parent language:

```
SELECT * FROM users WHERE name = '' OR '1'='1';
```

```
SELECT * FROM users WHERE name = '' OR '1'='1' -- ';
```

If this code were to be used in an authentication procedure then this example could be used to force the selection of a valid username because the evaluation of '1'='1' is always true.

The following value of "userName" in the statement below would cause the deletion of the "users" table as well as the selection of all data from the "userinfo" table (in essence revealing the information of every user), using an [API](#) that allows multiple statements:

```
a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't
```

This input renders the final SQL statement as follows and specified:

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM  
userinfo WHERE 't' = 't';
```

While most SQL server implementations allow multiple statements to be executed with one call in this way, some SQL APIs such as [PHP](#)'s `mysql_query()` function do not allow this for security reasons. This prevents attackers from injecting entirely separate queries, but doesn't stop them from modifying queries.

Incorrect type handling [\[edit\]](#)

This form of SQL injection occurs when a user-supplied field is not [strongly typed](#) or is not checked for [type](#) constraints. This could take place when a numeric field is to be used in a SQL statement, but the programmer makes no checks to validate that the user supplied input is numeric. For example:

```
statement := "SELECT * FROM userinfo WHERE id =" + a_variable + ";"
```

It is clear from this statement that the author intended `a_variable` to be a number correlating to the "id" field. However, if it is in fact a [string](#) then the [end-user](#) may manipulate the statement as they choose, thereby bypassing the need for escape characters. For example, setting `a_variable` to

```
1;DROP TABLE users
```

will drop (delete) the "users" table from the database, since the SQL becomes:

```
SELECT * FROM userinfo WHERE id=1;DROP TABLE users;
```

Blind SQL injection [\[edit\]](#)

Blind SQL Injection is used when a web application is vulnerable to an SQL injection but the results of the injection are not visible to the attacker. The page with the vulnerability may not be one that displays data but will display differently depending on the results of a logical statement injected into the legitimate SQL statement called for that page. This type of attack can become time-intensive because a new statement must be crafted for each bit recovered. There are several tools that can automate these attacks once the location of the vulnerability and the target information has been established.^[12]

Conditional responses [\[edit\]](#)

One type of blind SQL injection forces the database to evaluate a logical statement on an ordinary application screen. As an example, a book review website uses a [query string](#) to determine which book review to display. So the [URL](#) `http://books.example.com/showReview.php?ID=5` would cause the server to run the query

```
SELECT * FROM bookreviews WHERE ID = 'Value(ID)';
```

from which it would populate the review page with data from the review with [ID](#) 5, stored in the [table](#) `bookreviews`. The query happens completely on the server; the user does not know the names of the database, table, or fields, nor does the user know the query string. The user only sees that the above URL returns a book review. A [hacker](#) can load the URLs `http://books.example.com/showReview.php?ID=5 OR 1=1` and `http://books.example.com/showReview.php?ID=5 AND 1=2`, which may result in queries

```
SELECT * FROM bookreviews WHERE ID = '5' OR '1'='1';
SELECT * FROM bookreviews WHERE ID = '5' AND '1'='2';
```

respectively. If the original review loads with the "1=1" URL and a blank or error page is returned from the "1=2" URL, the site is likely vulnerable to a SQL injection attack. The hacker may proceed with this query string designed to reveal the version number of [MySQL](#) running on the server:

`http://books.example.com/showReview.php?`

`ID=5 AND substring(@@version,1,1)=4`, which would show the book review on a server running MySQL 4 and a blank or error page otherwise. The hacker can continue to use code within query strings to glean more information from the server until another avenue of attack is discovered or his or her goals are achieved.^{[13][14]}

Second Order SQL Injection [\[edit\]](#)

Second Order SQLi happens when submitted values contain SQL injection attacks that are stored instead of executed immediately. In some cases, the application may correctly encode a SQL statement and store it as valid SQL. Then, another part of that application without controls to protect against SQL Injection might execute that stored SQL statement. This attack requires more knowledge of how submitted values are later used. Automated web application security scanners would not easily detect this type of SQL Injection and may need to be manually instructed where to check for evidence of second order SQLi.

Mitigation [\[edit\]](#)

Parameterized statements [\[edit\]](#)

Main article: [Prepared statement](#)

With most development platforms, parameterized statements that work with parameters can be used (sometimes called placeholders or [bind variables](#)) instead of embedding user input in the statement. A placeholder can only store a value of the given type and not an arbitrary SQL fragment. Hence the SQL injection would simply be treated as a strange (and probably invalid) parameter value.

In many cases, the SQL statement is fixed, and each parameter is a [scalar](#) , not a [table](#) . The user input is then assigned (bound) to a parameter.^[15]

Enforcement at the coding level [\[edit\]](#)

Using [object-relational mapping](#) libraries avoids the need to write SQL code. The ORM library in effect will generate parameterized SQL statements from object-oriented code.

Escaping [\[edit\]](#)

A straightforward, though error-prone, way to prevent injections is to escape characters that have a special meaning in SQL. The manual for an SQL DBMS explains which characters have a special meaning, which allows creating a comprehensive [blacklist](#) of characters that need translation. For instance, every occurrence of a single quote (') in a parameter must be replaced by two single quotes (' ') to form a valid SQL string literal. For example, in [PHP](#) it is usual to escape parameters using the function `mysql_real_escape_string()` ; before sending the SQL query:

```
$mysqli = new mysqli('hostname', 'db_username', 'db_password',
'db_name');
$query = sprintf("SELECT * FROM `Users` WHERE UserName='%s' AND
Password='%s'",
                $mysqli->real_escape_string($Username),
                $mysqli->real_escape_string($Password));
$mysqli->query($query);
```

Note: 'mysql' is deprecated, and should be avoided. `mysqli` [↗](#) is preferred.^[16] This function, i.e. `mysql_real_escape_string()`, calls MySQL's library function `mysql_real_escape_string`, which prepends backslashes to the following characters: `\x00`, `\n`, `\r`, `\,`, `'`, `"` and `\x1a`. This function must always (with few exceptions) be used to make data safe before sending a query to [MySQL](#).^[17]

There are other functions for many database types in PHP such as `pg_escape_string()` for [PostgreSQL](#) . There is, however, one function that works for escaping characters, and is used especially for querying on databases that do not have escaping functions in PHP. This function is: `addslashes(string $str)`. It returns a string with backslashes before characters that need to be quoted in database queries, etc. These characters are single quote ('), double quote ("), backslash (\) and NUL (the NULL byte).^[18]

Routinely passing escaped strings to SQL is error prone because it is easy to forget to escape a given string. Creating a transparent layer to secure the input can reduce this error-proneness, if not entirely eliminate it.^[19]

In .Net Framework just use parameterized query or stored procedure using parameters

```
SqlCommand cmd1= new SqlCommand("Select * from users where username
='"+@paramUser+"' and password='"+@paramPass+"' ", conn1)
```

```
cmd1.CommandType= CommandType.Text cmd1.Parameters.Add("@paramUser",
SqlCommandType.NVarChar,20).Value=Username1 cmd1.Parameters.Add("@paramPass",
SqlCommandType.NVarChar,20).Value=pass1
```

In this case if for example Username1="" or 1=1--" this will not succeed.

Pattern check [\[edit\]](#)

Integer, float or boolean parameters can be checked if their value is valid representation for the given type. Strings that must follow some strict pattern (date, UUID, alphanumeric only, etc.) can be checked if they match this pattern.

Database permissions [\[edit\]](#)

Limiting the permissions on the database logon used by the web application to only what is needed may help reduce the effectiveness of any SQL injection attacks that exploit any bugs in the web application.



For example, on [Microsoft SQL Server](#) , a database logon could be restricted from selecting on some of the system tables which would limit exploits that try to insert JavaScript into all the text columns in the database.

```
deny select on sys.sysobjects to webdatabaselogon;
deny select on sys.objects to webdatabaselogon;
deny select on sys.tables to webdatabaselogon;
deny select on sys.views to webdatabaselogon;
deny select on sys.packages to webdatabaselogon;
```

Examples [\[edit\]](#)

- In September 1995, Andrew Plato, a technical writer for Microsoft discovered that he could send SQL queries through URL string of an early e-commerce site and directly query the database (SQL Server 6.0). Unaware of what this meant, Plato approached developers who dismissed the issue as irrelevant.^{[\[citation needed\]](#)}
- In February 2002, Jeremiah Jacks discovered that Guess.com was vulnerable to an SQL injection attack, permitting anyone able to construct a properly-crafted URL to pull down 200,000+ names, credit card numbers and expiration dates in the site's customer database.^{[\[20\]](#)}
- On November 1, 2005, a teenage hacker used SQL injection to break into the site of a [Taiwanese](#) information security magazine from the Tech Target group and steal customers' information.^{[\[21\]](#)}
- On January 13, 2006, [Russian](#) computer criminals broke into a [Rhode Island government](#) web site and allegedly stole credit card data from individuals who have done business online with state agencies.^{[\[22\]](#)}
- On March 29, 2006, a hacker discovered an SQL injection flaw in an official [Indian government](#) 's [tourism](#) site.^{[\[23\]](#)}
- On June 29, 2007, a computer criminal defaced the [Microsoft](#) UK website using SQL injection.^{[\[24\]](#)[\[25\]](#)} UK website *The Register* quoted a Microsoft [spokesperson](#) acknowledging the problem.
- In January 2008, tens of thousands of PCs were infected by an automated SQL injection attack that exploited a vulnerability in application code that uses [Microsoft SQL Server](#) as the database store.^{[\[26\]](#)}
- In July 2008, [Kaspersky](#) 's [Malaysian](#) site was hacked by a [Turkish](#) hacker going by the handle of "m0sted", who said to have used an SQL injection.
- In February 2013, a group of Maldivian hackers, hacked the website " UN-Maldives" using SQL Injection.
- In May 28, 2009 [Anti-U.S. Hackers Infiltrate Army Servers](#) Investigators believe the hackers used a technique called SQL injection to exploit a security vulnerability in Microsoft's SQL Server database to gain entry to the Web servers. "m0sted" is known to have carried out similar attacks on a number of other Web sites in the past—including against a site maintained by Internet security company Kaspersky Lab.
- On April 13, 2008, the [Sexual and Violent Offender Registry](#) of [Oklahoma](#) shut down its website for "routine maintenance " after being informed that 10,597 [Social Security numbers](#) belonging to [sex offenders](#) had been downloaded via an SQL injection attack^{[\[27\]](#)}
- In May 2008, a [server farm](#) inside [China](#) used automated queries to [Google's search engine](#) to identify [SQL](#)

[server](#) websites which were vulnerable to the attack of an automated SQL injection tool.^{[26][28]}

- In 2008, at least April through August, a sweep of attacks began exploiting the SQL injection vulnerabilities of Microsoft's [IIS web server](#) and [SQL Server database server](#) . The attack does not require guessing the name of a table or column, and corrupts all text columns in all tables in a single request.^[29] A HTML string that references a [malware JavaScript](#) file is appended to each value. When that database value is later displayed to a website visitor, the script attempts several approaches at gaining control over a visitor's system. The number of exploited web pages is estimated at 500,000.^[30]
- On August 17, 2009, the [United States Department of Justice](#) charged an American citizen, [Albert Gonzalez](#) , and two unnamed Russians with the theft of 130 million credit card numbers using an SQL injection attack. In reportedly "the biggest case of [identity theft](#) in American history", the man stole cards from a number of corporate victims after researching their [payment processing systems](#). Among the companies hit were credit card processor [Heartland Payment Systems](#) , convenience store chain [7 Eleven](#), and supermarket chain [Hannaford Brothers](#) .^[31]
- In December 2009, an attacker breached a [RockYou](#) plaintext database containing the [unencrypted](#) usernames and passwords of about 32 million users using an SQL injection attack.^[32]
- On July 2010, a South American security researcher who goes by the [handle](#) "Ch Russo" obtained sensitive user information from popular [BitTorrent](#) site [The Pirate Bay](#) . He gained access to the site's administrative control panel and exploited a SQL injection vulnerability that enabled him to collect user account information, including [IP addresses](#) , [MD5 password hashes](#) and records of which torrents individual users have uploaded.^[33]
- From July 24 to 26, 2010, attackers from [Japan](#) and [China](#) used an SQL injection to gain access to customers' credit card data from Neo Beat, an [Osaka](#) -based company that runs a large online supermarket site. The attack also affected seven business partners including supermarket chains Izumiya Co, Maruetsu Inc, and Ryukyu Jusco Co. The theft of data affected a reported 12,191 customers. As of August 14, 2010 it was reported that there have been more than 300 cases of credit card information being used by third parties to purchase goods and services in China.
- On September 19 during the [2010 Swedish general election](#) a voter attempted a code injection by hand writing SQL commands as part of a [write in](#) vote.^[34]
- On November 8, 2010 the British [Royal Navy](#) website was compromised by a Romanian hacker named TinKode using SQL injection.^{[35][36]}
- On February 5, 2011 [HBGary](#) , a technology security firm, was broken into by [LulzSec](#) using a SQL injection in their CMS-driven website^[37]
- On March 27, 2011, [mysql.com](#) , the official homepage for [MySQL](#), was compromised by a hacker using SQL blind injection^[38]
- On April 11, 2011, [Barracuda Networks](#) was compromised using an SQL injection flaw. [Email addresses](#) and usernames of employees were among the information obtained.^[39]
- Over a period of 4 hours on April 27, 2011, an automated SQL injection attack occurred on [Broadband Reports](#) website that was able to extract 8% of the username/password pairs: 8,000 random accounts of the 9,000 active and 90,000 old or inactive accounts.^{[40][41][42]}
- On June 1, 2011, " [hacktivists](#) " of the group [LulzSec](#) were accused of using SQLI to steal [coupons](#) , download keys, and passwords that were stored in plaintext on [Sony](#) 's website, accessing the personal information of a million users.^{[43][44]}
- In June 2011, [PBS](#) was hacked, mostly likely through use of SQL injection; the full process used by hackers to execute SQL injections was described in this [Imperva](#)  blog.^[45]
- In May 2012, the website for [Wurm Online](#), a [massively multiplayer online game](#), was shut down from an SQL injection while the site was being updated.^[46]
- In July 2012 a hacker group was reported to have stolen 450,000 login credentials from [Yahoo!](#) . The logins were stored in [plain text](#) and were allegedly taken from a Yahoo [subdomain](#) , [Yahoo! Voices](#). The group breached Yahoo's security by using a "[union](#) -based SQL injection technique".^{[47][48]}
- On October 1, 2012, a hacker group called "Team GhostShell" published the personal records of students, faculty, employees, and alumni from 53 universities including [Harvard](#) , [Princeton](#) , [Stanford](#) , [Cornell](#) , [Johns Hopkins](#) , and the [University of Zurich](#) on [pastebin.com](#) . The hackers claimed that they were trying

to “raise awareness towards the changes made in today’s education”, bemoaning changing education laws in Europe and increases in [tuition in the United States](#) .^[49]

- On June 27, 2013, hacker group " [RedHack](#) " breached Istanbul Administration Site.^[50] They claimed that, they’ve been able to erase people's debts to water, gas, Internet, electricity, and telephone companies. Additionally, they published admin user name and password for other citizens to log in and clear their debts early morning. They announced the news from twitter.^[51]

In popular culture [\[edit\]](#)

- Unauthorized login to web sites (i.e. hacking) by means of SQL injection forms the basis of one of the subplots in [J.K. Rowling](#) 's novel "[The Casual Vacancy](#) " , published in 2012.
- The minor [xkcd](#) character "Robert"); DROP TABLE students;--" (also known as "Little Bobby Tables") was named to carry out a SQL injection.^{[52][53]}

See also [\[edit\]](#)

- [Web application security](#)
- [Code injection](#)
- [Cross-site scripting](#)
- [Metasploit Project](#)
- [OWASP](#) Open Web Application Security Project
- [w3af](#)
















References [\[edit\]](#)

1. [^] Microsoft. "[SQL Injection](#)" . Retrieved 2013-08-04. "SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker."
2. [^] Imperva (July 2012). "[Imperva Web Application Attack Report](#)" (PDF). Retrieved 2013-08-04. "Retailers suffer 2x as many SQL injection attacks as other industries. / While most web applications receive 4 or more web attack campaigns per month, some websites are constantly under attack. / One observed website was under attack 176 out of 180 days, or 98% of the time."
3. [^] "[Category:OWASP Top Ten Project](#)" . OWASP. Retrieved 2011-06-03.
4. [^] "[Category:OWASP Top Ten Project](#)" . OWASP. Retrieved 2013-08-13.
5. [^] "[WHID 2007-60: The blog of a Cambridge University security team hacked](#)" . Xiom. Retrieved 2011-06-03.
6. [^] "[WHID 2009-1: Gaza conflict cyber war](#)" . Xiom. Retrieved 2011-06-03.
7. [^] [\[1\]](#) ^[*dead link*]
8. [^] "[Third Wave of Web Attacks Not the Last](#)" . Dark Reading. Retrieved 2012-07-29.
9. [^] Danchev, Dancho (2007-01-23). "[Mind Streams of Information Security Knowledge: Social Engineering and Malware](#)" . Ddanchev.blogspot.com. Retrieved 2011-06-03.
10. [^] Deltchev, Krassen. "[New Web 2.0 Attacks](#)" . B.Sc. Thesis. Ruhr-University Bochum. Retrieved February 18, 2010.
11. [^] [IBM Informix Guide to SQL: Syntax. Overview of SQL Syntax > How to Enter SQL Comments](#) , IBM
12. [^] "[Using SQLBrute to brute force data from a blind SQL injection point](#)" . Justin Clarke. Archived from the original on June 14, 2008. Retrieved October 18, 2008.
13. [^] macd3v. "[Blind SQL Injection tutorial](#)" . Retrieved 6 December 2012.
14. [^] Andrey Rassokhin; Dmitry Oleksyuk. "[TDSS botnet: full disclosure](#)" . Retrieved 6 December 2012.
15. [^] "[SQL Injection Prevention Cheat Sheet](#)" . Open Web Application Security Project. Retrieved 3 March 2012.
16. [^] "[mysql Introduction - PHP Manual](#)" . PHP.net.
17. [^] "[mysqli->real_escape_string - PHP Manual](#)" . PHP.net.
18. [^] "[Addslashes - PHP Manual](#)" . PHP.net.
19. [^] "[Transparent query layer for MySQL](#)" . Robert Eisele. November 8, 2010.
20. [^] "[Guesswork Plagues Web Hole Reporting](#)" . [SecurityFocus](#). March 6, 2002.

- 21. ^ "WHID 2005-46: Teen uses SQL injection to break to a security magazine web site" . [Web Application Security Consortium](#). November 1, 2005. Retrieved December 1, 2009.
- 22. ^ "WHID 2006-3: Russian hackers broke into a RI GOV website" . [Web Application Security Consortium](#). January 13, 2006. Retrieved May 16, 2008.
- 23. ^ "WHID 2006-27: SQL Injection in incredibleindia.org" . [Web Application Security Consortium](#). March 29, 2006. Retrieved March 12, 2010.
- 24. ^ Robert (June 29, 2007). "Hacker Defaces Microsoft U.K. Web Page" . [cgisecurity.net](#). Retrieved May 16, 2008.
- 25. ^ Keith Ward (June 29, 2007). "Hacker Defaces Microsoft UK Web Page" . [Redmond Channel Partner Online](#). Retrieved May 16, 2008.
- 26. ^ ^a ^b Sumner Lemon, IDG News Service (May 19, 2008). "Mass SQL Injection Attack Targets Chinese Web Sites" . [PCWorld](#). Retrieved May 27, 2008.
- 27. ^ Alex Papadimoulis (April 15, 2008). "Oklahoma Leaks Tens of Thousands of Social Security Numbers, Other Sensitive Data" . [The Daily WTF](#). Retrieved May 16, 2008.
- 28. ^ Michael Zino (May 1, 2008). "ASCII Encoded/Binary String Automated SQL Injection Attack" .
- 29. ^ Giorgio Maone (April 26, 2008). "Mass Attack FAQ" .
- 30. ^ Gregg Keizer (April 25, 2008). "Huge Web hack attack infects 500,000 pages" .
- 31. ^ "US man 'stole 130m card numbers'" . [BBC](#). August 17, 2009. Retrieved August 17, 2009.
- 32. ^ O'Dell, Jolie (December 16, 2009). "RockYou Hacker - 30% of Sites Store Plain Text Passwords" . [New York Times](#). Retrieved May 23, 2010.
- 33. ^ "The pirate bay attack" . July 7, 2010.
- 34. ^ "Did Little Bobby Tables migrate to Sweden?" . [Alicebandmallory.com](#). Retrieved 2011-06-03.
- 35. ^ [Royal Navy website attacked by Romanian hacker](#)  [BBC News](#), 8-11-10, Accessed November 2010
- 36. ^ Sam Kiley (November 25, 2010). "Super Virus A Target For Cyber Terrorists" . Retrieved November 25, 2010.
- 37. ^ "We Are Anonymous: Inside the Hacker World of LulzSec" . Little, Brown and Company.
- 38. ^ "MySQL.com compromised" . [sucuri](#).
- 39. ^ "Hacker breaks into Barracuda Networks database" .
- 40. ^ "site user password intrusion info" . [Dslreports.com](#). Retrieved 2011-06-03.
- 41. ^ "DSLReports says member information stolen" . [Cnet News](#). 2011-04-28. Retrieved 2011-04-29.
- 42. ^ "DSLReports.com breach exposed more than 100,000 accounts" . [The Tech Herald](#). 2011-04-29. Retrieved 2011-04-29.
- 43. ^ "LulzSec hacks Sony Pictures, reveals 1m passwords unguarded" , [electronista.com](#), June 2, 2011
- 44. ^ Ridge Shan (June 6, 2011), "LulzSec Hacker Arrested, Group Leaks Sony Database" , [The Epoch Times](#)
- 45. ^ "Imperva.com: PBS Hacked - How Hackers Probably Did It" . Retrieved 2011-07-01.
- 46. ^ "Wurm Online is Restructuring" . May 11, 2012.
- 47. ^ Chenda Ngak. "Yahoo reportedly hacked: Is your account safe?" . [CBS News](#). July 12, 2012. Retrieved July 16, 2012.
- 48. ^ <http://www.zdnet.com/450000-user-passwords-leaked-in-yahoo-breach-7000000772/> 
- 49. ^ Perlroth, Nicole (3 October 2012). "Hackers Breach 53 Universities and Dump Thousands of Personal Records Online" . [New York Times](#).
- 50. ^ "RedHack Breaches Istanbul Administration Site, Hackers Claim to Have Erased Debts" .
- 51. ^ "Redhack tweet about their achievement" .
- 52. ^ Munroe, Randall. "XKCD: Exploits Of A Mom" . Retrieved 26 February 2013.
- 53. ^ "Bobby Tables: A guide to preventing SQL injection" . Retrieved 6 October 2013.

External links [\[edit\]](#)

- [Complete Reference Guide to SQL Injection, Attack and Prevention Method of SQL Injection](#)  by WorldofHacker.
- [SQL Injection Knowledge Base](#) , by Websec.
- [Blind Sql injection with Regular Expression](#) 
- [WASC Threat Classification - SQL Injection Entry](#) , by the Web Application Security Consortium.
- [Why SQL Injection Won't Go Away](#) , by Stuart Thomas.
- [SQL Injection Attacks by Example](#) , by Steve Friedl 

- [SQL Injection Prevention Cheat Sheet](#) , by OWASP.
- [SQL Injection Tutorial](#) , by BTS.
- [sqlmap: automatic SQL injection and database takeover tool](#) 
- [SDL Quick security references on SQL injection](#)  by Bala Neerumalla.
- [Backdoor Web-server using MySQL SQL Injection](#)  By Yuli Stremovsky
- [Defacing website with SQL injection](#)  by sploitwiki

Categories: [Data management](#) | [Injection exploits](#) | [SQL](#) | [Computer security exploits](#)

This page was last modified on 11 February 2014 at 05:20.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#).

Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

