

Lancaster University

Network Security Lab 03 Instruction

0. Open Lancaster.waynechiu.cc

Lab 01 - PCAP

Lab 02 - HashCat

Lab 03 - SQL

Lab 04 - Buffer Overflow

The Lab of Network Security lecture of
Lancaster University

1. Do some readings of SQL injection

SQL injection

41 languages

Contents hide

(Top)

History

Root cause

Ways to exploit

Getting direct output or action

Blind SQL injection

Conditional responses

Second-order SQL injection

Prevention/Mitigation

String escaping

Object relational mappers

Parameterized statements

Pattern check

Database permissions

Article Talk

Read Edit View history Tools

From Wikipedia, the free encyclopedia

In computing, **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][2]} 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.

SQL injection attacks allow attackers to [spoof](#) identity, tamper with existing [data](#), cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server. Document-oriented [NoSQL](#) databases can also be affected by this security vulnerability.^[3]

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

History [edit]

Classification parameters	Methods	Techniques/Implementation
Intent	Identifying susceptible parameters	see "Input type of attack"
	Extracting Data	
	Adding or Modifying Data	
	Performing Denial of Service	
	Escalating detection	
	Bypassing Authentication	
Input Source	Executing remote commands	Malicious strings in Web forms Input (file(s)) POST-Method Modified cookie fields containing SQLiAs Blindness are manipulated to contain SQLiAs Frequency-based Primary Application Frequency-based Secondary Application Secondary Support Application Cascaded Submissions Application
	Performing privilege escalation	
	Injection through user input	
	Injection through cookies	
	Injection through server variables	
	Second-order injection	
Input type of attack, technical aspect	Classic SQLiAs	Piggy-Backed Queries Tautologies Alternate Encodings Illegal Logically Incorrect Queries UNION SQLiAs Stored Procedures SQLiAs
		Conditional Responses
		Conditional Errors
		Out-Of-Band Channeling
		(Double-Blind SQLiAs/Time delays/ Benchmark attacks)
	Inference	Timing SQLiAs
		Deep Blind SQLiAs
		Multiple statements SQLiAs
	DBMS specific SQLiAs	DBF Programming DB Mapping
	Compounded SQLiAs	Fast Flaming SQLiAs

A classification of SQL injection attacking vector as of 2010

Appearance hide

Text

☐ Small
☒ Standard
☐ Large

Width

☒ Standard
☐ Wide

Color (beta)

☐ Automatic
☒ Light
☐ Dark

2. Reading the SQL query hint

Web Login

This Lab03 SQL of Network Security
lecture of Lancaster University

The SQL statement is

```
SELECT * FROM users WHERE  
(username = 'YOUR_USERNAME_HERE')  
AND (password =  
'YOUR_PASSWORD_HERE')
```

[I need instructions.](#)

Tips: Making the outcome of
where clause true.

3. Try injecting

Web Login

Username
TRUE

Password
••••

Login

This Lab03 SQL of Network Security
lecture of Lancaster University

The SQL statement is
SELECT * FROM users WHERE
(username = 'YOUR_USERNAME_HERE')
AND (password =
'YOUR_PASSWORD_HERE')

[I need instructions.](#)

4. Congratulation!

Login successful!

The SQL statement is : `SELECT * FROM users WHERE`

Which results in following query result :

Username =		Password =	
Username =		Password =	
Username =		Password =	