

## Vulnerabilities

Remote Code Execution (RCE):  
 Payloads, Commands &  
 Vulnerabilities

OS Command Injection | Payloads,  
 prevention, & attacks

Server-Side Request Forgery (SSRF)  
 | Attack Types & Risks

What is an SQL Injection (SQLi)?  
 Prevention, Examples, Attacks ^

In-band SQL injection

**Blind SQL injection**

Out-of-band SQL injection

Local file inclusion (LFI)

Remote file inclusion (RFI)

Directory traversal

Cross-Site Scripting (XSS)  
 Vulnerability Guide v

Cross-site request forgery (CSRF)

XML external entity (XXE)  
 Injection | Attack & Examples v

CRLF injection

Email injection

HTML injection

NoSQL injection

JSON Injection

Insecure Direct Object References  
 (IDOR) Vulnerability References

Unvalidated redirects and  
 forwards v

Directory Listing | Risks &  
 Vulnerabilities Explained

Password reset poisoning

# Blind SQL injection



## What is blind SQL injection?

*Blind SQL injection* is a type of **SQL injection** where the attacker does not receive an obvious response from the attacked database and instead reconstructs the database structure step-by-step by observing the behavior of the database server and the application. Blind SQL injection is also called *inferential SQL injection*.

There are two types of blind SQL injections: *boolean-based* and *time-based*.

## Consequences of blind SQL injection

Performing an attack using blind SQL injections takes much longer than in the case of in-band SQL injections but can yield the same results. Based on the behavior of the database server and the application, the attacker may be able to do the following:

- Check if other types of SQL injections are possible

Buffer overflow

Integer Overflow Guide: Attacks, Examples, Vulnerabilities

Attacks

Session Hijacking: Detection, Defense & Prevention

What is a Session Fixation Attack? Vulnerabilities, Prevention, and More

Session prediction

Cookie hijacking

Cookie poisoning

Clickjacking

Forced browsing

XSS filter evasion

Source code disclosure

Host header attacks

Web Cache Poisoning | Cache Poisoning Attack Explained

Privilege escalation

What is a Web Shell | Types, Attacks, Detection & Protection

Reverse shell

Cross-frame scripting (XFS)

Broken link hijacking

SEO poisoning

Low Orbit Ion Cannon (LOIC)

High Orbit Ion Cannon (HOIC)

Slowloris attack

R.U.D.Y. attack

Man-in-the-middle attacks (MITM)

Protection

Same-origin policy (SOP)

HTTP Strict Transport Security (HSTS): Example & Fix

- Get information about the structure of the database
- Get data out of the database

# What is boolean-based blind SQL injection?

*Boolean-based blind SQL injection* is a subtype of blind SQL injection where the attacker observes the behavior of the database server and the application after combining legitimate queries with malicious data using boolean operators.

## Example of boolean-based blind SQL injection

As an example, let’s assume that the following query is meant to display details of a product from the database.

```
1 SELECT * FROM products WHERE id = product_id
```

At first, a malicious hacker uses the application in a legitimate way to discover at least one existing product ID – in this example, it’s product 42. Then, they can provide the following two values for *product\_id*:

```
1 42 AND 1=1
2 42 AND 1=0
```

If this query is executed in the application using simple string concatenation, the query becomes respectively:

```
1 SELECT * FROM products WHERE id = 42 and 1=1
2 SELECT * FROM products WHERE id = 42 and 1=0
```

If the application behaves differently in each case, it is susceptible to boolean-based blind SQL injections.

## Tools

What is DAST? A Guide to  
Dynamic Application Security  
Testing

Static Application Security Testing  
(SAST) | Tools and Solutions

Interactive Application Security  
Testing (IAST) Tools

Software composition analysis  
(SCA)

Web application firewall (WAF)

Web asset discovery

Vulnerability management

Vulnerability assessment

## About Invicti Learn

If the database server is Microsoft SQL Server, the attacker can now supply the following value for *product\_id*:

```
1  42 AND (SELECT TOP 1 substring(name, 1, 1)
2    FROM sysobjects
3    WHERE id=(SELECT TOP 1 id
4      FROM (SELECT TOP 1 id
5        FROM sysobjects
6        ORDER BY id)
7    AS subq
8    ORDER BY id DESC)) = 'a'
```

As a result, the sub-query in parentheses after **42 AND** checks whether the name of the first table in the database starts with the letter *a*. If true, the application will behave the same as for the payload **42 AND 1=1**. If false, the application will behave the same as for the payload **42 AND 1=0**.

The attacker can iterate through all letters and then go on to the second letter, third letter, etc. As a result, the attacker can discover the full name of the first table in the database structure. They can then try to get more data about the structure of this table and finally – extract data from the table. While this example is specific to MS SQL, similar techniques exist for other database types.

## What is time-based blind SQL injection?

*Time-based blind SQL injection* is a subtype of blind SQL injection where the attacker observes the behavior of the database server and the application after combining legitimate queries with SQL commands that cause time delays.

## Example of time-based blind SQL injection

Let's say we have the same query as in the example above:

```
1 SELECT * FROM products WHERE id = product_id
```

A malicious hacker may provide the following *product\_id* value:

```
1 42; WAITFOR DELAY '0:0:10'
```

As a result, the query becomes:

```
1 SELECT * FROM products WHERE id = 1; WAITFOR DELAY '0:0:10'
```

If the database server is Microsoft SQL Server and the application is susceptible to time-based blind SQL injections, the attacker will see a 10-second delay in the application.

Now that the attacker knows that time-based blind SQL injections are possible, they can provide the following *product\_id*:

```
1 42; IF(EXISTS(SELECT TOP 1 *
2 FROM sysobjects
3 WHERE id=(SELECT TOP 1 id
4 FROM (SELECT TOP 1 id
5 FROM sysobjects
6 ORDER BY id)
7 AS subq
8 ORDER BY id DESC)
9 AND ascii(lower(substring(name, 1, 1))) = 'a'))
10 WAITFOR DELAY '0:0:10'
```

If the name of the first table in the database structure begins with the letter *a*, the second part of this query will be true, and the application will react with a 10-second delay. Just like for boolean-based blind SQL injections above, the attacker can use this method repeatedly to discover the name of the first table in the database structure, then try to get more data

about the table structure of this table and finally extract data from the table.

## How to prevent blind SQL injection vulnerabilities?

The only fully effective way to prevent all types of SQLi vulnerabilities in web applications, including blind SQLi, is to use parameterized queries (also known as prepared statements) to access SQL databases. If your programming language does not support parameterized queries but your database engine supports stored procedures, you may use stored procedures with prepared statements instead. Relying purely on other prevention methods such as whitelists, blacklists, or input filtering/escaping, is not recommended. Malicious hackers may find a way around such sanitization.

## Frequently asked questions

**What is blind SQL injection?**

**What is boolean-based SQL injection?**

**What is time-based SQL injection?**

## Related blog posts

- [What is blind SQL injection?](#)



Invicti Security Corp  
1000 N Lamar Blvd Suite 300  
Austin, TX 78703, US

© Invicti 2025

## RESOURCES

[Features](#)  
[Integrations](#)  
[Plans](#)  
[Case Studies](#)  
[Changelogs](#)  
[Invicti Learn](#)

## WEB SECURITY

[The Problem with False Positives](#)  
[Why Pay for Web Scanners](#)  
[SQL Injection Cheat Sheet](#)  
[Getting Started with Web Security](#)  
[Vulnerability Index](#)  
[Content Security Policy \(CSP\)](#)  
[Directives, Examples, Fixes](#)

## COMPANY

[About Us](#)  
[Contact Us](#)  
[Support](#)  
[Careers](#)  
[Resources](#)  
[Partners](#)

## USE CASES

[Penetration Testing Software](#)  
[Website Security Scanner](#)  
[Ethical Hacking Software](#)  
[Web Vulnerability Scanner](#)  
[Comparisons](#)  
[Online Application Scanner](#)

## COMPARISON

[Acunetix vs. Invicti](#)  
[Burp Suite vs. Invicti](#)  
[Checkmarx vs. Invicti](#)  
[Probely vs. Invicti](#)  
[Qualys vs. Invicti](#)  
[Tenable Nessus vs. Invicti](#)