Application Security

by Sophia

WHAT'S COVERED

This lesson explores security concerns with applications that connect to databases, in three parts. Specifically, this lesson will cover:

1. Confidentiality, Integrity, and Availability (CIA)

When planning for robust database security, there are three main concerns to address: confidentiality, integrity, and availability. These are often referred to as the **CIA triad**.

Confidentiality focuses on ensuring that the data is protected against unauthorized access or disclosure of private information. Many organizations have to follow various laws and guidelines around data confidentiality, like the Health Insurance Portability and Accountability Act (HIPAA) in medicine or the Sarbanes-Oxley Act (SOX) in the business world. As such, the data stored within databases needs to be classified as highly restricted, where very few individuals would have access (credit card information, as an example); confidential, where certain groups would have information (pay information for employees, as an example); or unrestricted, where anyone can have access.

Integrity focuses on ensuring that the data is consistent and free from errors. The database itself plays a key role in data integrity, as we have seen in prior lessons. Protecting sensitive data often involves using encryption. Through encryption, the underlying data is scrambled, using a key to a format so that it cannot be read as is.

There are many forms of encryption with various algorithms. With weaker encryption, you may see attackers trying to decrypt the data by brute force, which means trying to guess the decryption key through iterative trial and error.

Availability focuses on the accessibility of the data when authorized users want access to it.

With applications, security vulnerabilities can occur due to many different factors. Individuals can sometimes take advantage of bugs within the application that connects to the database. Issues can occur when code is poorly developed or focused purely on functionality and not security. One example is session hijacking, when an individual takes over a web session from another individual. By doing so, the individual can access certain personal information from another user that they may not have been able to access otherwise.

TERMS TO KNOW

CIA Triad
The combination of confidentiality, integrity, and availability that ensures data and systems are fully protected from security threats.

Confidentiality
The assurance that private data is not accessible to unauthorized persons.

Availability
The assurance that data will be available when it is needed.

Integrity
The assurance that data is consistent and free from errors.

2. SQL Injections

SQL injections are one of the most common web hacking techniques used with applications. They can damage your database or provide complete unauthorized access to the database if things are not well protected. SQL injections typically work by attempting to add malicious code into SQL statements through the use of web page input.

This can occur if the page asks for input like a userid and concatenates the input to the SQL string. For example, from a program, we may have something that looks like this:

```
my input = readFromInput('userid');
mysql = 'SELECT * FROM users where user_id =' + myinput;
```

If the user enters in 5 for the userid, this will create a SQL statement like this:

```
SELECT *
```

```
FROM users
WHERE user_id = 5;
```

That would be the expected result. However, if the “hacker” entered in “5 or 1=1”, the following SQL statement would look like this:

```
SELECT *
FROM users
WHERE user_id = 5 or 1=1;
```

If we look at the statement, the 1=1 will always return true, which means that the query would return every single row within the table. Imagine if the table had usernames, passwords, and other user information. All of that would now be compromised in this successful SQL injection. Or perhaps the input could look like “5; DROP TABLE customer;”. The resulting query would look like this:

```
SELECT *
FROM users
WHERE user_id = 5; DROP TABLE customer;
```

If this SQL injection is successful, it could potentially drop the table customer, which is also quite problematic.

Different databases handle SQL injection issues slightly differently. To avoid these types of scenarios, what is important is that the application first ensures that the input data has been validated before sending the query to the database. We also want to filter input data to avoid the user bypassing our validation. By filtering the user input, we can ensure that we're checking for any special characters that should not be included. In many applications, the use of SQL parameters can help.

TERM TO KNOW

SQL Injection

A hacking technique that inserts malicious SQL statements into a data entry field in an attempt to execute the statements on the database.

3. Other Types of Security Attacks on Databases

In addition to SQL injection, there are many other types of attacks that cybercriminals use to try to get unauthorized access to a database's data. These include:

3a. Cross-Site Scripting (XSS)

This is an attack on a website that inserts dangerous Javascript code into the page's HTML. It isn't specific to pages that provide database interfaces, but the scripting can affect the security of the database that the webpage accesses. Input validation can help minimize the risk of XSS. So can implementing a content-security-policy (CSP), which can restrict the sources from which scripts can be loaded.

3b. Weak Authentication

Database systems that allow weak, easy-to-guess passwords are at risk from hacking software that guesses passwords.

3c. Weak Authorization

Database systems that grant unnecessary privileges to user accounts can be easier to exploit after gaining entry via stolen or guessed user credentials. Following the principle of least privilege—that is, not allowing users any more privileges than they need—can restrict the amount of damage an attacker can do if they gain access through stolen credentials.

3d. Insecure Direct Object References (IDOR)

This occurs when a web-based application exposes internal implementation details, such as database keys or filenames, in URLs or other parameters. Attackers can manipulate those references to access unauthorized data.

3e. Insecure Data Transmission

If data is transmitted between the application and the database without encryption, it can be intercepted and tampered with by attackers. For example, an attacker might capture the stream of data and glean usernames and passwords from it.

3f. Denial of Service (DoS) Attacks

Attackers overwhelm the application with a large volume of requests or queries, which degrades the database's performance and can even render it unavailable.

To mitigate these risks, developers should:

- Follow secure coding practices
- Implement proper input validation
- Enforce least privilege access controls
- Enforce strong authentication and authorization mechanisms
- Encrypt sensitive data
- Regularly update the database application
- Conduct security assessments and audits

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

In this lesson, you learned that **CIA** stands for **confidentiality, integrity, and availability**. It represents the fundamental principles that guide secure systems design and implementation. Confidentiality ensures that data is only accessible to authorized individuals or entities, preventing unauthorized disclosure. Integrity guarantees data accuracy and consistency, safeguarding against unauthorized modifications or tampering. Availability ensures that data and services are accessible when needed, preventing downtime and ensuring uninterrupted operations. These three principles collectively form the foundation for creating secure and resilient information systems.

SQL injection is a severe cybersecurity vulnerability in PostgreSQL and other database systems. This vulnerability occurs when malicious actors exploit poor input validation and execute arbitrary SQL code. This vulnerability arises when an application fails to properly sanitize user inputs before incorporating them into SQL queries. Attackers can manipulate input fields, such as login forms or search bars, to inject malicious SQL statements. The database server executes these statements. In PostgreSQL, this could lead to unauthorized access, data theft, data manipulation, or even a complete database compromise. By understanding SQL injection risks and implementing preventative measures, developers can help ensure the security and integrity of PostgreSQL databases and the applications that interact with them.

In addition to SQL injection, you learned there are many **other types of security attacks on databases**, involving **cross-site scripting (XSS)**, **weak authentication**, **weak authorization**, **insecure direct object references (IDOR)**, **insecure data transmission**, and **denial of service (DoS) attacks**, and briefly covered ways that developers can mitigate these risks.

TERMS TO KNOW

Availability	The assurance that data will be available when it is needed.
CIA Triad	The combination of confidentiality, integrity, and availability that ensures data and systems are fully protected from security threats.
Confidentiality	The assurance that private data is not accessible to unauthorized persons.
Integrity	The assurance that data is consistent and free from errors.

SQL Injection

A hacking technique that inserts malicious SQL statements into a data entry field in an attempt to execute the statements on the database.