SQL & NoSQL Vulnerabilities

By:

Ali Mehrani

Ali Eftekhari

Taha Ebrahimzadeh

Introduction to SQL Databases

- Structured Databases
- Using SQL for data definition and manipulation
- Relational data model
- Schema-based design
- ACID compliance

Introduction to SQL Databases

- Available on cloud services
- Plenty of options to choose
- MySQL, PostgreSQL, Oracle Database, SQLite and more

Introduction to SQL Databases

- Ensuring data integrity with ACID Properties:
  - Atomicity, Consistency, Isolation, and Durability
- Use cases including:
  - Transactional applications
  - Web & Enterprise applications
  - Data analytics
  - Data warehousing

Introduction to NoSQL Databases

- Non-relational data model
- Flexible schema design
- less structured compared to relational SQL databases
- suitable for storing distributed data
- High scalability

Introduction to NoSQL Databases

- Highly used in large-scale systems
- Use cases including:
    - Big data applications
    - Real-time web applications
    - Content and media streaming systems

Introduction to NoSQL Databases

NoSQL databases have different structures, with each type handling data in its unique way. Some examples include:

Document Stores

- Storing data in documents
- MongoDB, Apache CouchDB
- Documents can have different structures

Key-Value Stores

- Storing data in key-value pairs
- Redis, Amazon DynamoDB
- High performance

Introduction to NoSQL Databases

Some other examples for NoSQL databases structure include

Graph Stores

- Represents data in nodes and edges
- Interconnected data
- Flexible schema
- Neo4j, Amazon Neptune-AWS

Wide-column Stores

- Storing data in columns instead of rows
- Columns grouped into families
- Flexible schema
- Cassandra, ScyllaDB, HBase

Database Vulnerabilities

Database vulnerabilities can arise for various reasons. Here are some of the most common vulnerabilities in SQL/NoSQL database systems:

- SQL/NoSQL Injection
- Data Inconsistency (NoSQL)
- Privilege Escalation
- Inappropriate Encryption
- Insecure Direct Object Reference (IDOR)
- Database system Misconfiguration
- Weak Authentication and Authorization
- Buffer Overflow

SQL/NoSQL Injection

SQL Injection

- Inserting malicious SQL code into queries
- Prevented by using Object-relational mapping (ORM) for RDBMS
- NoSQL injection is another variant for NoSQL databases
- NoSQL variant similarly targets NoSQL queries (e.g. cassandra CQL)

SQL Injection Example

Consider a login page in a web app:

- Entered field values are sent directly to the query with no preprocessing.

```
SELECT * FROM users WHERE username = 'username_in' AND password = 'pass_in';
```

Attacker enters:

- Username: `'admin' OR 1=1 --'` and password: `'something'`
- The query is now:

```
SELECT * FROM users WHERE username = 'admin' OR 1=1 --' AND password = 'something';
```

- The `--'` comments the rest of the query.

SQL Injection Prevention

- **Using prepared statements:** ensures that queries are compiled first and the parameters are passed later
- **Validating inputs:** manually check for malicious inputs.
- **Web App Firewall (WAF):** protects web apps from numerous attacks including SQL injection.
- **Object-relational mapping (ORM):** checks for malicious inputs, protects the DB.

NoSQL Injection Example

Consider this script for finding user data in MongoDB

```
function findUser(request) {
username=request.username;
password=request.password;
user = db.collection('users').findOne({ username: username, password: password });
user ? return (user, 'login success') : ('login failure');
}
```

NoSQL Injection Example

The attacker will enter the following JSON as the request input for the function

```
{
  "username": "admin",
  "password": { "$ne": null }
}
```

The following will be executed and will bypass the password checking procedure.

```
db.collection('users').findOne({ username: "admin", password: { "$ne": null } })
```

NoSQL Injection Prevention

- **Validating Inputs:** checking input validity either manually or by using libraries or frameworks
- **Web application firewall:** protects the web app from malicious traffic
- **Prepared Statements:** NoSQL Databases might allow using prepared statements for execution
- **Object-Document mapping:** Document-store databases might allow using ODMs which will handle input verification automatically

Insecure Direct Object Reference (IDOR)

An access control vulnerability arising when users directly access system objects and resources.

- Not directly about the DB but about the access control
- Attacker mostly attacks by manipulating the URL
- Among owasp vulnerabilities
- Predictable URL behaviour which references DB objects directly

IDOR Examples

Suppose the following URL:

https://university.com/stdinfo/810100000.txt

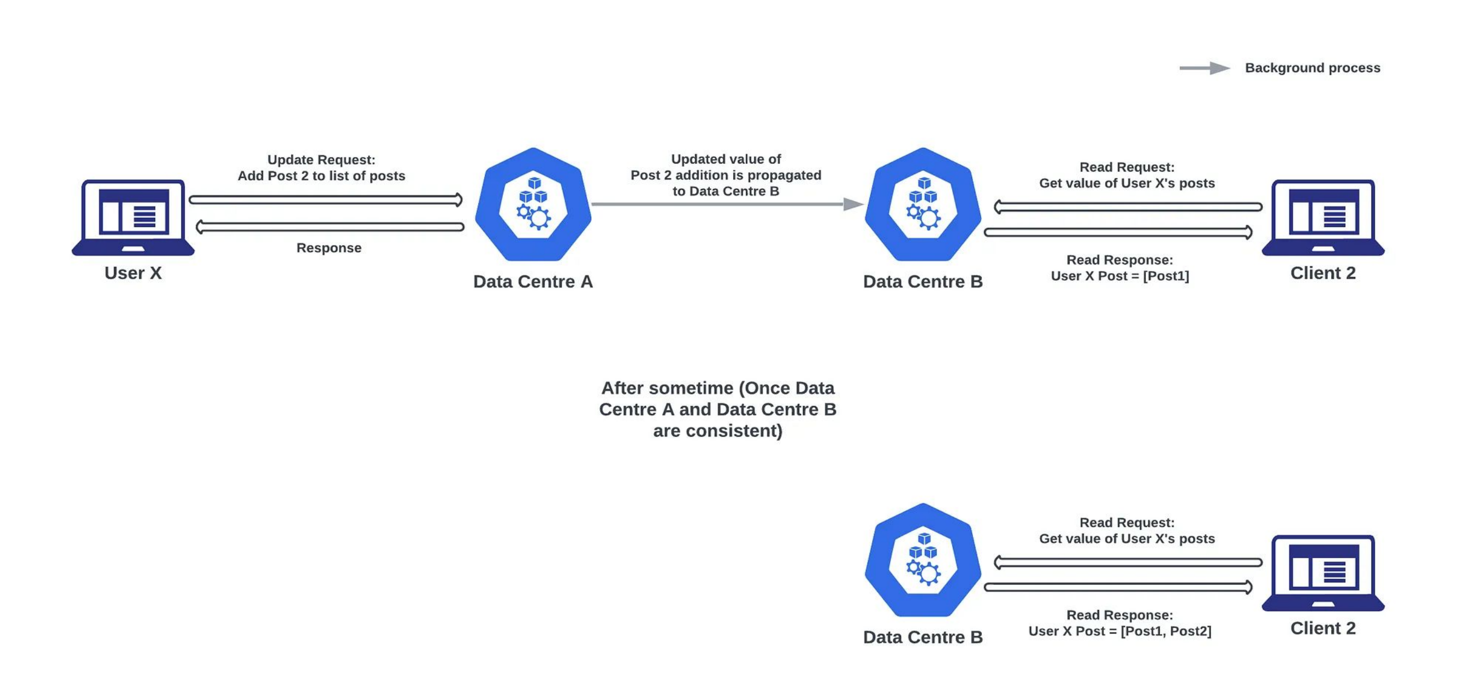
- A student data is stored in a text file accessed by a static URL
- Attacker can access any student data by changing the StdNo. in the URL.
- Can be prevented by implementing proper access controls.

Data Inconsistency (NoSQL)

Causes:

- Eventual Consistency
- Distribution
  - Network issues
  - Node failures
- Lack of ACID Transactions

Data Inconsistency (NoSQL)



Database System Misconfiguration

Common vulnerabilities:

- Open Network Access
  - Exposing the database to the public internet without proper firewalls
  - Unrestricted IP access
- Missing Security Patches and Updates
- Misconfigured Backup Settings
- Improper Database Isolation

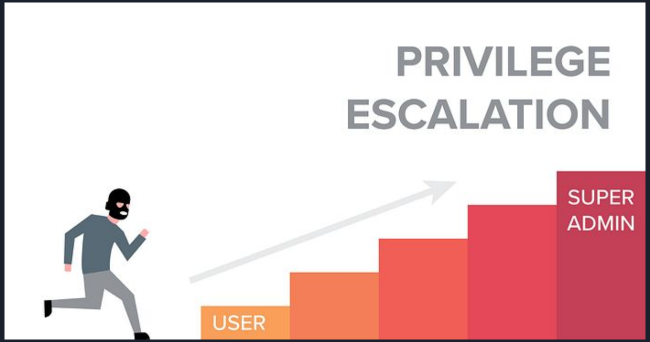
Database System Misconfiguration

Solutions:

- Secure Network Access
    - Restrict database access to trusted IP addresses
    - Use firewalls and VPNs to protect database access
- Keep Software Up-to-Date
- Implement Robust Backup Solutions
    - Configure regular, automated backups of the database
- Regular Audits and Penetration Testing

Privilege Escalation

- **Vertical Privilege Escalation**
  - user gains higher privileges



- **Horizontal Privilege Escalation**
  - user accesses another user's resources with similar privileges

Privilege Escalation

Causes:

- Improper Validation
- Weak Authentication
  - Can allow attackers to bypass authentication
- Software Bugs
  - Attackers can use an exploit to gain access
- Misconfigured Database Permissions
  - Unauthorized users can gain access

Privilege Escalation

Solutions:

- Input Validation
- Multi-factor Authentication
- Principle of Least Privilege
- Patch Management

Weak Authentication and Authorization

- **Authentication**
  - verifying the identity of a user or a system.
- **Authorization**
  - determines the level of access granted to authenticated users.

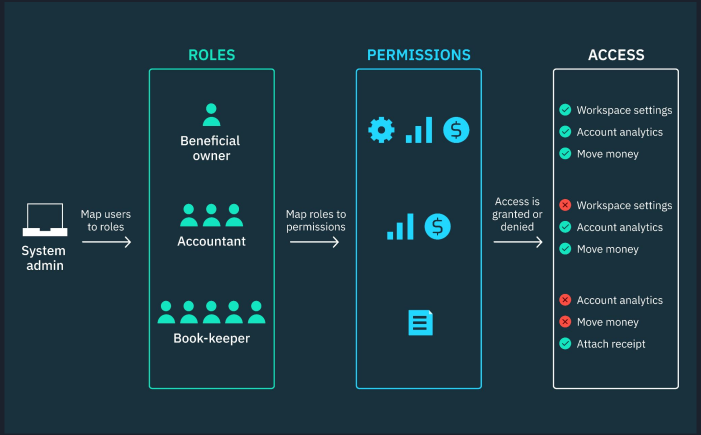
Weak Authentication and Authorization

Common weaknesses for both SQL/NoSQL:

- Default Credentials
- Lack of Multi-Factor Authentication (MFA)
- Weak Password Policies
- Improper Access Controls

Weak Authentication and Authorization

- Plaintext Storage of Credentials
- Lack of Role-Based Access Control
- Poor Session Management

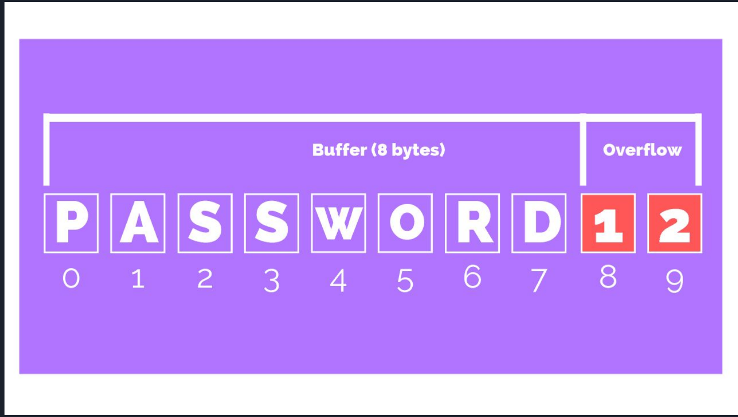


Buffer Overflow

1. **Stack-based Buffer Overflow**: This is the most common type. It occurs when the stack, a memory area that stores local variables and controls the order of execution, is overflowed.
2. **Heap-based Buffer Overflow**: This occurs in the heap, a memory area used for dynamic allocation.

Preventions:

- Maintenance
- Input validation
- Memory management
- Filtering the data



Buffer Overflow

Example:

```
char buffer[256];

strcpy(buffer, userInput);   // Vulnerable to overflow if userInput exceeds 256 bytes
```

Solution:

```
strncpy(buffer, userInput, sizeof(buffer) - 1);

buffer[sizeof(buffer) - 1] = '\0';   // Ensure null-termination
```

Inappropriate Encryption

- Weak Encryption Algorithms
  - Using algorithms like DES which can be easily broken
- Short Encryption Keys
  - Using short keys such as 56-bit keys instead of stronger keys like 256-bit keys.
- Poor Implementation
  - Incorrect use of cryptographic libraries, or failing to use secure modes of operation (like using ECB instead of CBC for block ciphers).
- Encryption Performance Issues
  - Overhead of strong encryption impacting database performance

References

1. https://ibm.com/topics/database-security
2. https://owasp.org
3. https://www.proofpoint.com/us/threat-reference/privilege-escalation
4. https://janmuhammadzaidi.medium.com/vertical-privilege-escalation-the-user-can-takeover-an-admin-account-via-response-manipulation-9237c8b2fefa

Thank you

for your attention!