|  |  |
| --- | --- |
| Database Security | NoSQL Security | NoSQL Database - RavenDB NoSQL Database  NoSql injection - mongodb | Abstract  Databases are an essential part of our everyday life: almost everything in the 21st century relies heavily upon some kind of a database (e.g. shopping online or withdrawing money from the bank). The recent advance in technology has created the need to store large amount of data, allowing for high availability and scalability; thus the IT world saw the rise of the NoSQL databases that would compensate the shortcomings of the traditional model (relational - SQL). Unfortunately, this type of DB also has some vulnerabilities, that can be easily exploited by malicious parties if the configuration of the database is improper. Below, the paper will focus mainly on the NoSQL injection (NoSQLi) as the topic for discussion.  DLarisa  Database Security Course 2022-2023 |

Contents

[Introduction 2](#_Toc123412149)

[NoSQL 2](#_Toc123412150)

[MongoDB 4](#_Toc123412151)

[NoSQL Injection 4](#_Toc123412152)

[Bibliography 12](#_Toc123412153)

# Introduction

Databases are a part of our everyday life: from shopping, to banking and online social media platforms, all companies rely on databases in order to retain big amounts of data. Data persistence and analysis, in order to draw conclusions about the company's activity, are essential to predict future performance and ensure long-term survival.

The best well-known databases, are the relational ones, defined by E.F.Codd in his research paper from 1970 „*A Relational Model of Data for Large Shared Data Banks*” (Codd, 2002). The increased use of social media has resulted in an explosion of user-generated data that must be properly managed, analysed, and archived. Other data sources, such as GPS, sensors, automated trackers, and monitoring systems, also generate a large amount of data on a regular basis. This advancement in technology has created the need to store large amounts of data (just to form a genereal idea, „*the big 4*”, aka Google, Amazon, Microsoft and Facebook store at least 1200 petabytes between them – 1 petabyte = 1000 terabytes (Mitchell, 2022)) that the traditional model cannot support. SQL databases fail in this new scenario because they use complex hierarchies, becoming slower and slower with the escalation of data they have to maintain. Although SQL dominated the market for a long time, solutions have begun to appear that solve this issue and offer the possibility of storing data in a much simpler way – and we are talking about NoSQL databases.

NoSQL databases are not new in the IT world (they have been around since the 1960s (Wikipedia, 2022)), but they started to gain fame in the early 2009 and since then, they developed rapidly (NoSQL-Database.org, 2022). Unlike traditional SQL databases, NoSQL was created in part as a response to the adoption of more agile design methodologies, allowing for dynamic schema definitions.

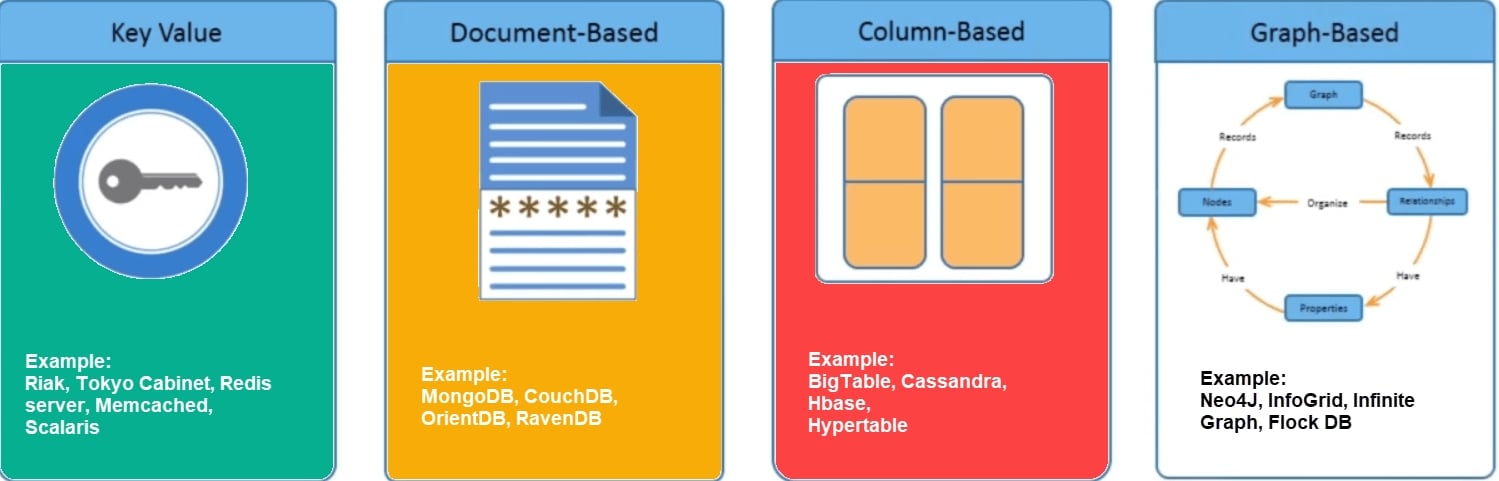
# NoSQL

Per definition, the main features of a NoSQL (a.k.a. „non-SQL” or „not only SQL”) database are: non-relational, distributed, open-source and horizontally scalable. Often more characteristics apply such as: schema-free, easy replication support, simple API, eventually consistent, a huge amount of data and more (NoSQL-Database.org, 2022).

Because NoSQL Databases do not store data in a relational schema, data attributes can be added on the fly without affecting the structure of the entire table or adding redundant elements to the remaining rows. On the other hand, NoSQL Databases are great at scaling horizontally (servers are low-cost commodity hardware in comparison with RDBMS which scales vertically and requires more RAM, more CPU, more HDD, etc..) (Gupta, 2022). Basically, they are more flexible than relational databases; can handle schemas that are not strict, with large amount of data, prioritizing availability and offering only eventual consistency.

**Types** (Samuel, 2022)**,** (Bhardwaj, 2022):

1. *Document DB*: stores data as documents (XML, JSON, YAML, etc...), seen as independent units, that consist of nested objects (tree structures). They enable a smooth transition from the object world of programming languages to data storage. In essence, they are a type of key-value DB. MongoDB is one of the most famous examples.
2. *Key–Value DB*: stores data as a collection of key-value pairs (dictionary ot hash table DB – keys must be unique). The values do not follow a fixed schema – can be anything like an object, string, JSON, etc... It works best for shopping cart content. E.g.: Redis.
3. *Column–Oriented DB*: stores data as a collection of columns, each one treated separately. It works best for aggregation queries like SUM, COUNT, MAX, MIN, AVG, etc... E.g.: Cassandra.
4. *Graph DB*: stores data as nodes and relationship (similar to the concept of „*graph*”). They are commonly used for social connection networks. E.g: Neo4j.

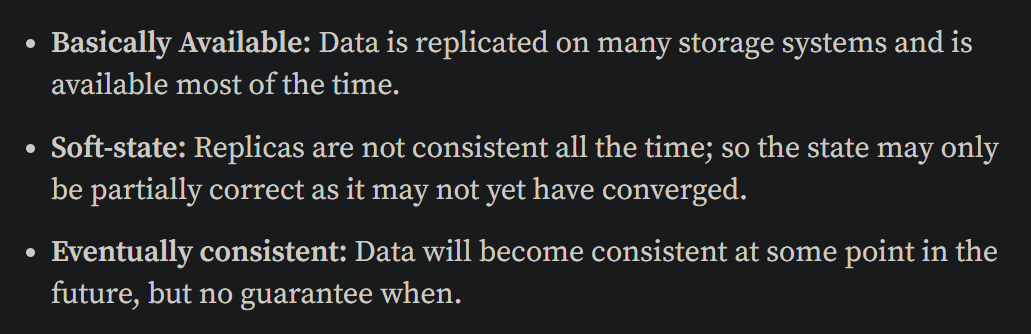
(Bhardwaj, 2022)

Some of the advantages of the NoSQL databases include (Gupta, 2022):

1. They can deal with large amounts of data and can be scaled quickly to manage voluminous data in a fast manner (query optimization). Also, the data models are flexible.
2. A NoSQL database never has a single point of failure and is constantly accessible. Even if one machine is offline, the cluster as a whole can still read requests. Within a data center, it can replicate data between various physical machines. It also prevents hardware failures (horizontal scaling).
3. Most NoSQL DB are cloud ready, especially with the rise of cloud infrastructure in the enterprise world. They are easily portable in this new setting and they are capable of supporting the hybrid solution where a portion of the database is hosted in the cloud and the other part is hosted within the company.
4. High Performance and Scalability - Performance in NoSQL databases can be improved by expanding the cluster with additional nodes. With more nodes added to a cluster, a database system's performance typically degrades. However, when new nodes are added, a good NoSQL database improves performance for both read and write operations, and performance gains are linear in nature.
5. A NoSQL database is easier to implement for developers.

Of course, there are some drawback to this kind of databases as well (Tozzi, 2022):

1. The biggest issue is the fact that they cannot perform ACID (Atomicity, Consistency, Isolation, Durability) transactions, referring to data integrity. Most NoSQL follow the BASE model (Basically Available, Soft-state, Eventually consistent).



1. Lack of standarization – each NoSQL database comes with its own set of rules, design and query language.
2. There is a lack of compatibility with SQL and also the JOIN operator cannot be used.

# MongoDB

MongoDB is a document database developed by 10gen (MongoDB, 2022) and is currently one of the most popular NoSQL database products. It manages collections of JSON-like documents, each with an unique id, and so, applications can have data represented in a more natural way, because data can be nested in complex hierarchies while remaining queryable and indexable. Among other benefits, this allows developers to build full-stack applications using only JavaScript. Documents are stored in collections (similar to a table in relational DB, but without a pre-defined schema), and collections are in turn stored in a database. It also has its own query language and it uses RESTful API.

Unfortunately, there are some security concerns in the design of MongoDB. We will mention just a few of them (Yaron Gonen, 2011):

1. Mongo data-files are unencrypted and Mongo does not provide an automatic encryption method, meaning that we need to rely on the application to ensure that sensitive information is first encrypted before is stored in the database.
2. Because MongoDB relies heavily on JavaScript (internal commands are short js scripts), it raises the risk for potential injection attacks. It is even possible to store javascript functions in the DB.
3. MongoDB does not have facilities for auditing actions performed in the database.

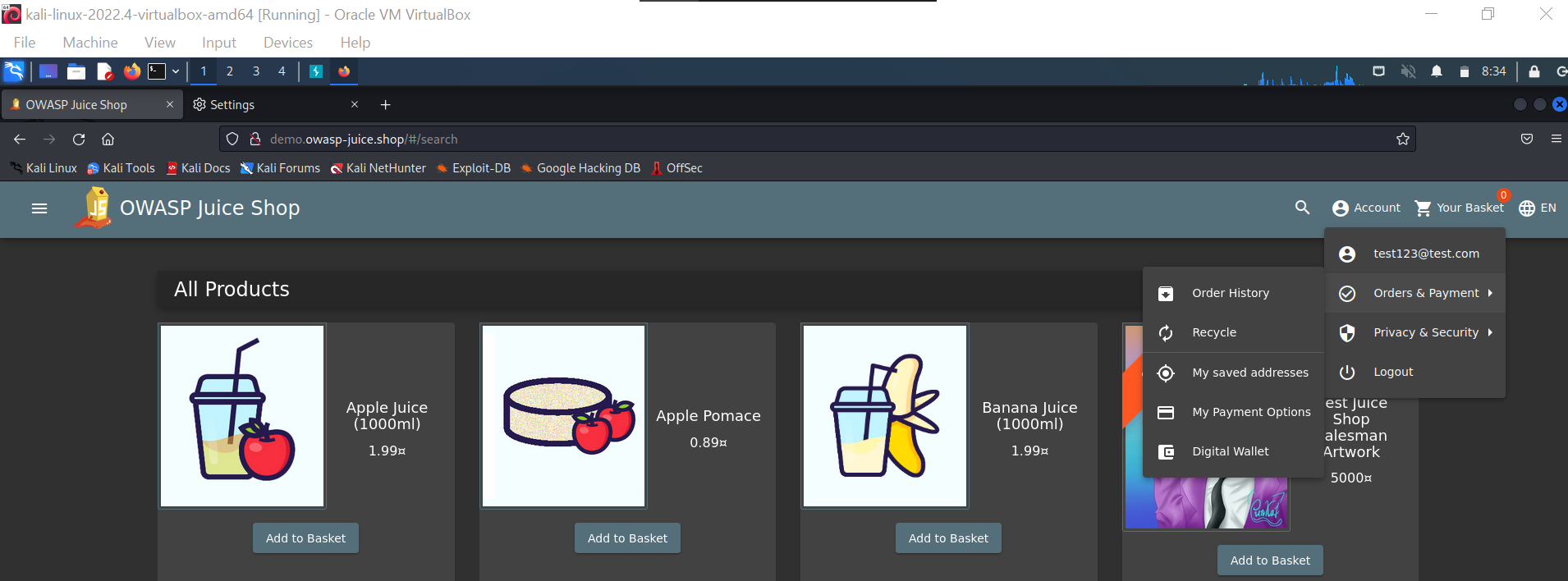
# NoSQL Injection

NoSQL injection is very similar to its conterpart, the SQL injection. NoSQL injection enables an attacker to read or modify backend data that they are not authorized to access by taking advantage of flaws revealed by inadequate input validation. It occurs at the application layer, and if successfully exploited, it might provide a hacker complete access to the database's contents. Additionally, an attacker could conduct malicious queries that compromise the database's capacity to maintain data confidentiality, integrity, and availability (SecureFlag, 2022).

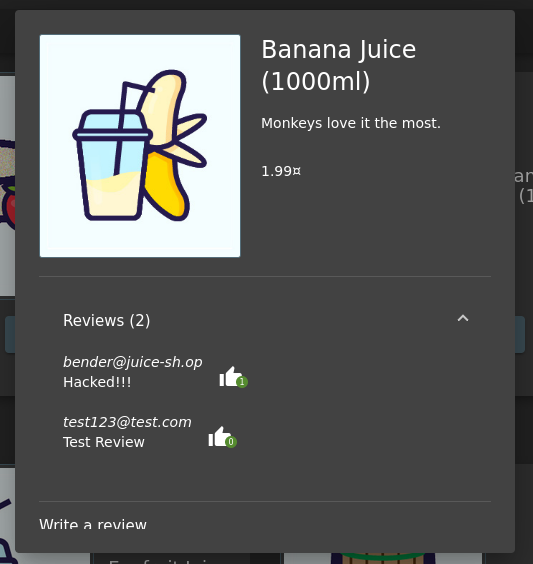
Consistency requirements are less rigorous for NoSQL databases than for traditional SQL databases. Scalability and performance benefits come from fewer relational constraints. Even though NoSQL databases don’t use the SQL language, they are nevertheless susceptible to injection. NoSQL injection is part of the „*OWASP top 10 Application Security Risks*” (OWASP, 2022).

For the practical part, I will use *Kali Linux VM* that comes with the *BurpSuite* application and demonstrate the attack.

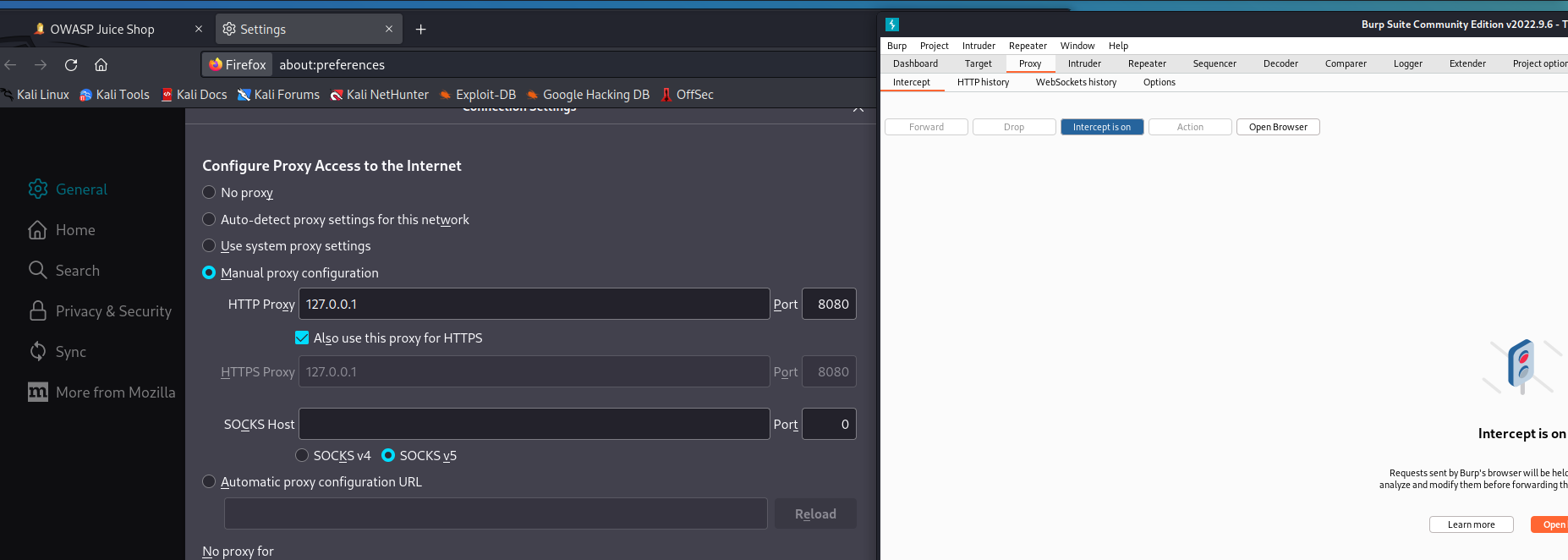
First, we access <http://demo.owasp-juice.shop/#/search> (OWASP, 2022), we create a new user and then we login.



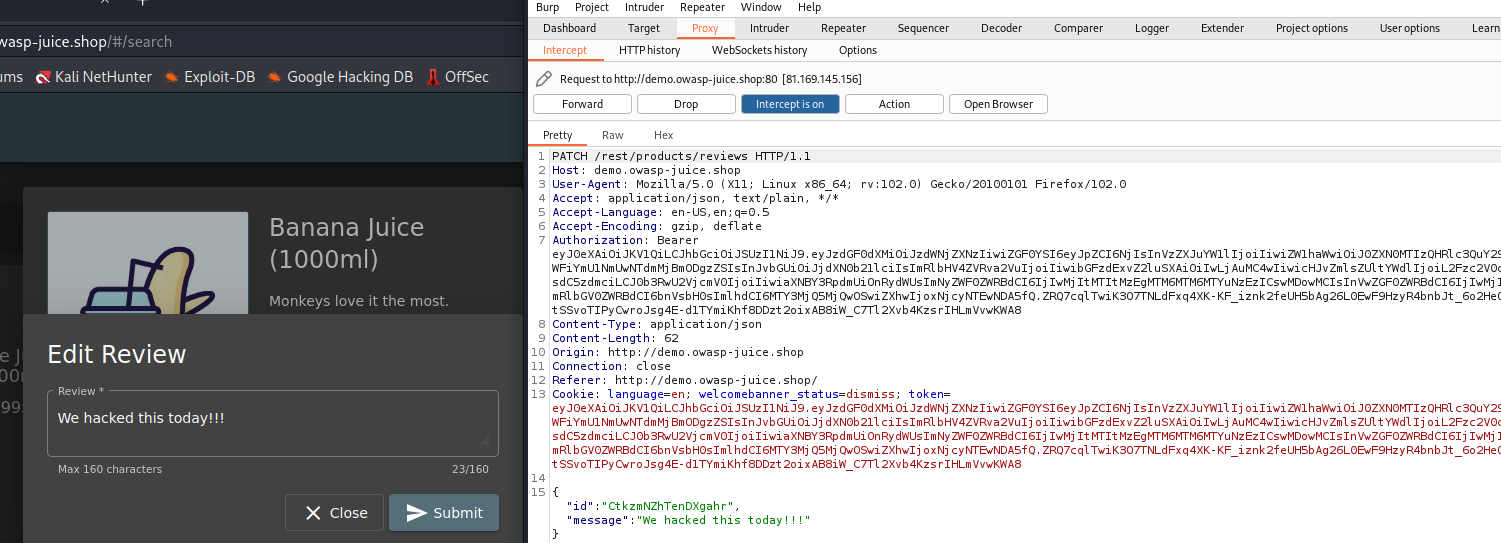
We enter a review:



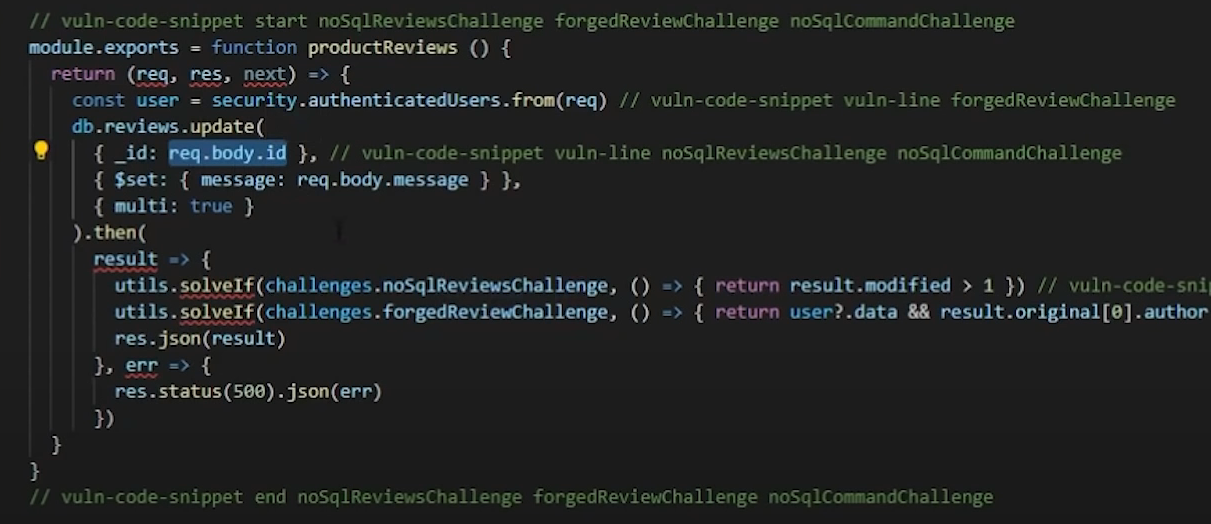
Now, we open burpSuite, make sure that the interceptor in ON and enable the proxy for our web page.



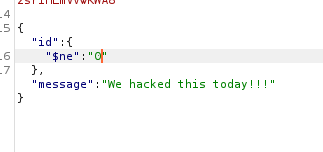
We edit our review in order to get the POST request:



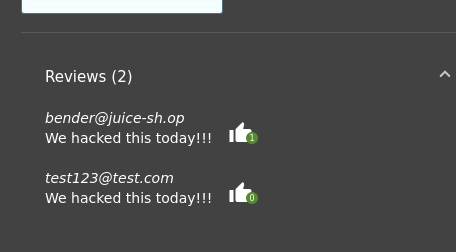
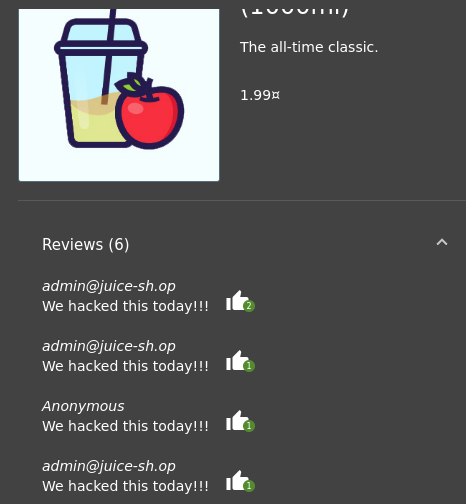
If we take a look at the code, we can notice the vulnerability – the id is exposed:



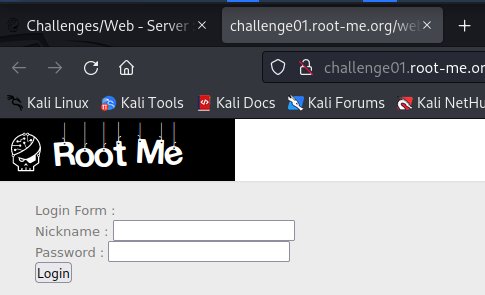
MongoDB uses JavaScript for different commands, as we mentioned above, so we can attack the shop as follows:



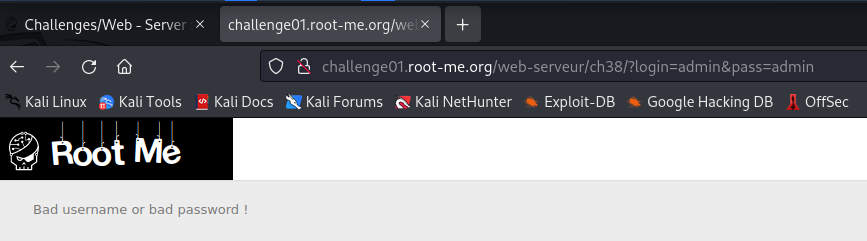
Basically, we update the id that is not equal with 0 (*„$ne”: „0”*) and all the ids from the database are different than 0 and so, the reviews will all change.

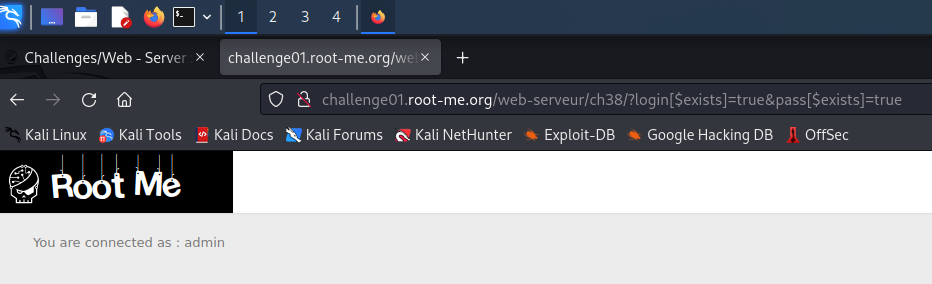
For another example, we will use <https://www.root-me.org/en/Challenges/Web-Server/NoSQL-injection-Authentication> (first, you need to create an account and authenticate) and start the challenge (Root Me, 2022).



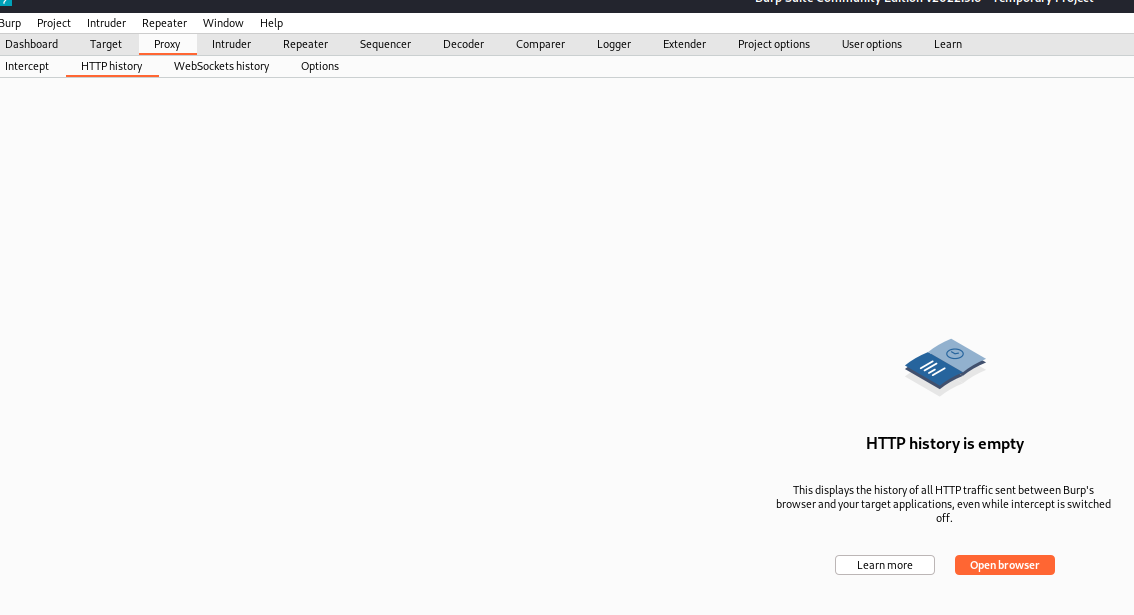
If we introduce admin/admin, we will be prompted to the following page:



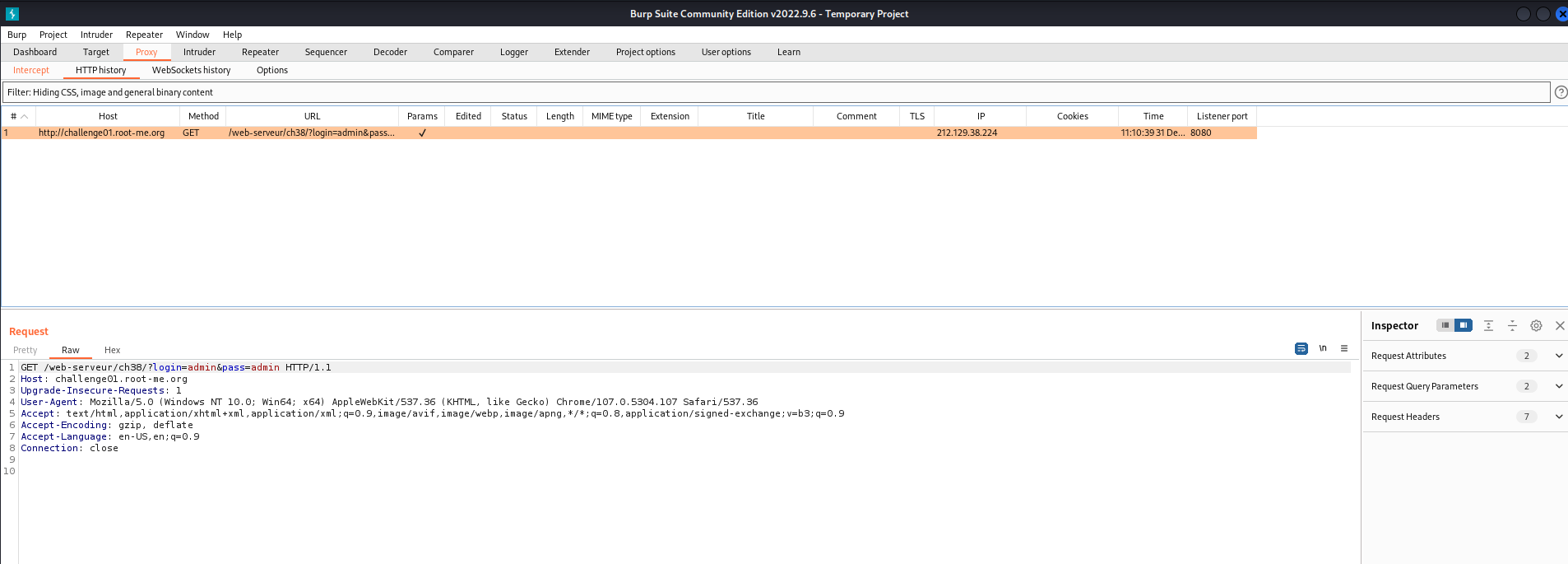
We can see the query strings (major security flaw). We can use this to our advantage. This [page](https://book.hacktricks.xyz/pentesting-web/nosql-injection) has all types of exploits for the NoSQL Injector (HackTricks, 2022). So we can alter the URL in order to gain access:



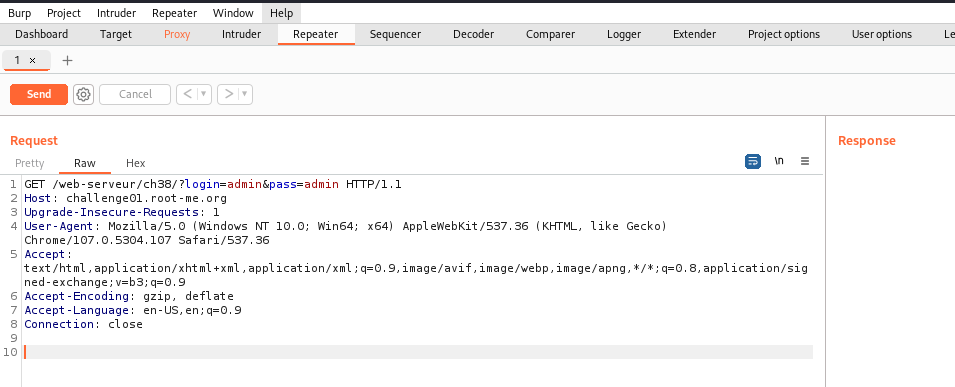
The challenge asks to find the username of the hidden user, not only to gain access, so let’s see how we can obtain it. Again, we use burpSuite and make sure that the interceptor is on (we don’t need to use proxy for this website).



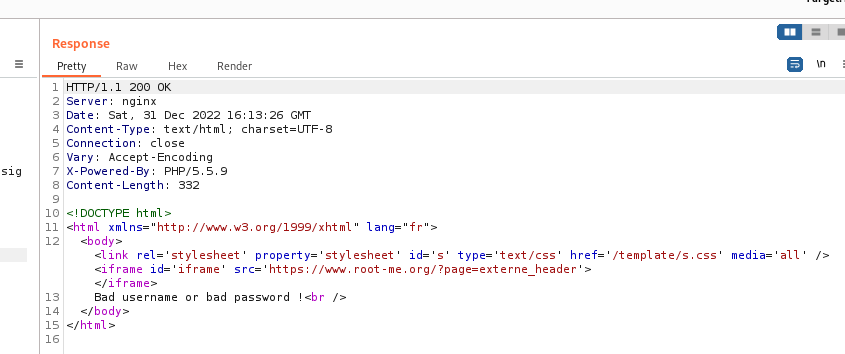
Click on the „*Open Browser*”. We enter this URL <http://challenge01.root-me.org/web-serveur/ch38/?login=admin&pass=admin> and then we will obtain this:



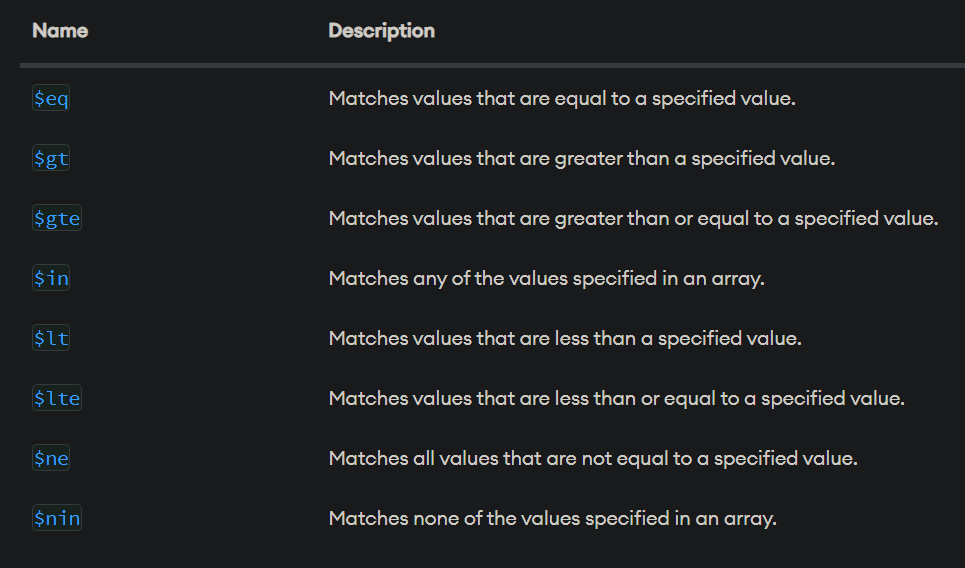
Select the full Raw request (CTRL+A), then CTRL+R and then we click on the „*Repeater*” tab.

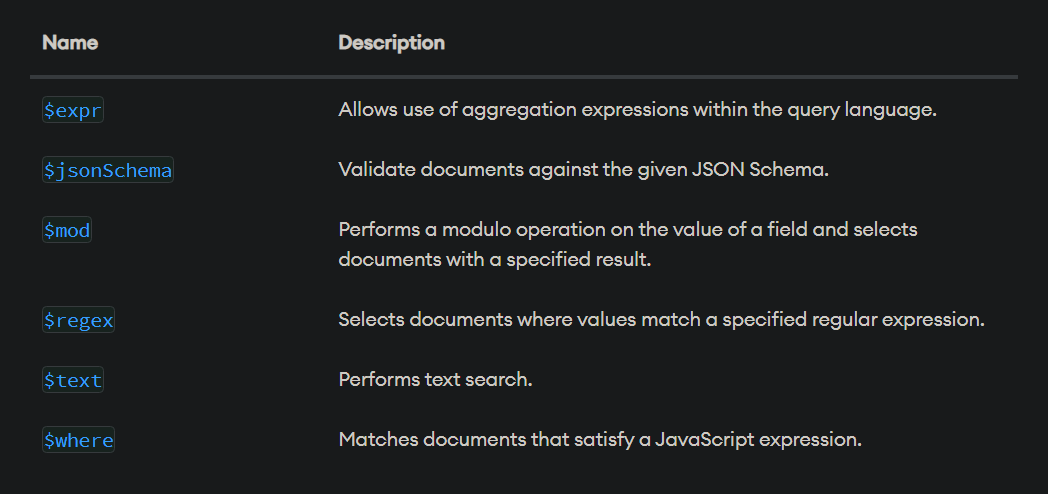


If click „*Send*”, we will get an error:

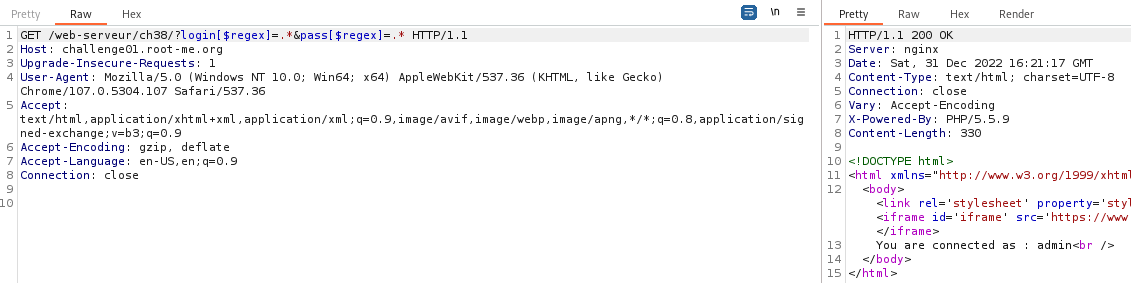


We need to edit this line „*GET /web-serveur/ch38/?login=admin&pass=admin HTTP/1.1*” in order to find the password. We take a look at the MongoDB selectors (MongoDB, 2022):

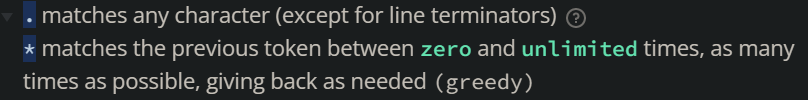




We should try to discover a nickname for the authentication (we saw above that one user is *admin*, but let’s pretend that we didn’t discovered that yet).



Why „*login[$regex]=.\*&pass[$regex]=.\**” works? We need to understand what the regex means (regex101, 2022):



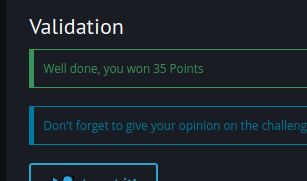
So one username is *admin*. Let’s see if we can find another one? We use „*login[$gt]=admin&pass[$regex]=.\**” (a user that is greater than *admin*).



So another username is *test*. So, the hidden user is between *admin* and *test*. So we send „*login[$gt]=admin&login[$lt]=test&pass[$regex]=.\**” (a user that is greater than *admin*, but smaller than *test*):



We check our answer and...



It is correct.

These are just a few examples of NoSQL Injections that use MongoDB. To further learn more, these resources are very useful: [1](https://book.hacktricks.xyz/pentesting-web/nosql-injection), [2](https://www.mongodb.com/docs/manual/reference/operator/query/), [3](https://nullsweep.com/a-nosql-injection-primer-with-mongo/), [4](https://steps0x29a.wordpress.com/2021/07/07/nosql-injection-in-mongodb/), [5](https://learn.snyk.io/lessons/nosql-injection-attack/javascript/).

To conclude, here are some ways of preventing NoSQL injections:

1. Instead of using the interpreter, use a secure API or library. You can use *mongo-sanitize* or *mongoose*, for instance, with MongoDB.
2. Use server-side input validation to prevent harmful and incorrect data. Verify any expressions and characters that shouldn't be present in the input data at all times. Also, use sanitized data instead of special characters, which can be exploited to generate NoSQL injection payloads.
3. If possible, use database controls that limit the amount of data a query can retrieve.
4. Run applications at the lowest privilege possible. For instance, run the web server with a least privileged user and connect to the database with a user that doesn’t have access to administration data. This security control limits application attacks.
5. In MongoDB: don’t use *where*, *mapReduce* or group operators (they are used in NoSQL injection payloads); or disable JavaScript in server-side execution.

# Bibliography

Bhardwaj, P. (2022, Dec 30). *A Comprehensive NoSQL Tutorial Guide for Beginner: Learn Step-by-Step*. Preluat de pe JanBask Training: https://www.janbasktraining.com/blog/nosql-tutorial/

Codd, E. (2002). A Relational Model of Data for Large Shared Data Banks. În *Software Pioneers* (pg. 263–294). https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf: https://doi.org/10.1007/978-3-642-59412-0\_16. Preluat de pe Software Pioneers (pp. 263–294): https://doi.org/10.1007/978-3-642-59412-0\_16

Gupta, S. C. (2022, Dec 30). *SQL vs. NoSQL Database: When to Use, How to Choose*. Preluat de pe Towards Data Science: https://towardsdatascience.com/datastore-choices-sql-vs-nosql-database-ebec24d56106#904d

HackTricks. (2022, Dec 30). *NoSQL injection*. Preluat de pe HackTricks: https://book.hacktricks.xyz/pentesting-web/nosql-injection

Mitchell, G. (2022, Dec 30). *How much data is on the internet?* Preluat de pe BBC Science Focus: https://www.sciencefocus.com/future-technology/how-much-data-is-on-the-internet/

MongoDB. (2022, Dec 30). Preluat de pe MongoDB: https://www.mongodb.com/

MongoDB. (2022, Dec 30). *Query and Projection Operators*. Preluat de pe MongoDB: https://www.mongodb.com/docs/manual/reference/operator/query/

NoSQL-Database.org. (2022, Dec 30). *NoSQL*. Preluat de pe NoSQL-Database.org: https://hostingdata.co.uk/nosql-database/

OWASP. (2022, Dec 30). *A03:2021 – Injection*. Preluat de pe OWASP Top 10:2021: https://owasp.org/Top10/A03\_2021-Injection/

OWASP. (2022, Dec 30). *OWASP Juice Shop: Probably the most modern and sophisticated insecure web application*. Preluat de pe Github: https://github.com/juice-shop/juice-shop

regex101. (2022, Dec 30). *regular expressions 101*. Preluat de pe https://regex101.com/

Root Me. (2022, Dec 30). *NoSQL injection - Authentication*. Preluat de pe Root Me: https://www.root-me.org/en/Challenges/Web-Server/NoSQL-injection-Authentication

Samuel, N. (2022, Dec 30). *SQL vs NoSQL Databases: 5 Critical Differences*. Preluat de pe HEVO: https://hevodata.com/learn/sql-vs-nosql/

SecureFlag. (2022, Dec 30). *NoSQL Injection*. Preluat de pe knowledge-base.secureflag: https://knowledge-base.secureflag.com/vulnerabilities/nosql\_injection/nosql\_injection\_vulnerability.html

Tozzi, C. (2022, Dec 30). *The Limitations of NoSQL Database Storage: Why NoSQL’s Not Perfect*. Preluat de pe ChannelFutures: https://www.channelfutures.com/cloud-2/the-limitations-of-nosql-database-storage-why-nosqls-not-perfect

Wikipedia. (2022, Dec 30). *NoSQL*. Preluat de pe Wikipedia - The Free Encyclopedia: https://en.wikipedia.org/wiki/NoSQL

Yaron Gonen, N. G.-O. (2011). Security Issues in NoSQL Databases. *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 541-547.