Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра

студента ***Рогалевої Марини Володимирівни***

академічної групи ***125м-17-1***

спеціальності ***125 Кібербезпека***

спеціалізації[1]

за освітньо-професійною програмою ***Кібербезпека***

на тему ***Забезпечення безпеки від атак-ін'єкцій нереляційної бази даних MongoDB NoSQL***

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	д.т.н., проф. Корнієнко В. І.			
розділів:				
спеціальний	ст. викл. Святошенко В.О.			
економічний	к.е.н., доц. Пілова Д.П.			

Рецензент				

Нормоконтролер	ст. викл. Мєшков В.І.			

Дніпро

2018

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня магістра

студенту ***Рогалевій М. В.*** **академічної групи** ***125м-17-1***
(прізвище та ініціали) (шифр)

спеціальності ***125 Кібербезпека***

спеціалізації[1]

за освітньо-професійною програмою ***Кібербезпека***

на тему ***Забезпечення безпеки від атак-ін'єкцій нереляційної бази даних MongoDB NoSQL***

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від __29.11.18№ 2025-л

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень ***Процес впровадження nosql-ін'єкції під час запиту користувача до бази даних***

Предмет досліджень ***Стійкість бази даних MongoDB до атаки через NoSQL-ін'єкції***

Мета ***Підвищення захисту конфіденційності інформації баз даних від атак типу NoSQL-ін'єкцій***

Вихідні дані для проведення роботи ***PHP, MONGODB, NoSQL-ін'єкції***

3 ОЧІКУВАНІ РЕЗУЛЬТАТИ

Наукова новизна ***Полягає у тому, що вперше створена методика на основі синтезу найефективніших та доступних рішень***

Практична цінність ***Полягає у можливості використовувати методику для різних баз даних для підвищення захисту інформації.***

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати повинні надати методику захисту інформації в базах даних від NoSQL-атак

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт** (початок-кінець)
Огляд джерел за темою та напрям досліджень	03.09.18-06.10.18
Методи досліджень	07.10.18-31.10.18
Результати досліджень	01.11.18-24.11.18
Виконання економічного розділу	25.11.18-04.12.18
Оформлення пояснювальної записки	05.12.18-10.12.18

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект ***доведено економічну ефективність та розраховано строк окупності затрат***

Соціальний ефект

7 ДОДАТКОВІ ВИМОГИ

Завдання видано _____ д.т.н., проф. Корнієнко В. І. В.
(підпис керівника) (прізвище, ініціали)

Дата видачі: 03.09.18р.

Дата подання до екзаменаційної комісії: 14.12.18р.

Прийнято до виконання _____ Рогалева М. В.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 91 с., 37 рис., 7 табл., 4 додатки, 21 джерел.

Об’єкт дослідження: процес впровадження NoSQL-ін’єкції під час запиту користувача до бази даних.

Мета роботи (проекту): підвищення захисту конфіденційності інформації баз даних від актуальних атак типу NoSQL-ін’єкцій.

Методи дослідження: методи порівняння, індукції та дедукції, системного підходу, аналізу і синтезу.

У спеціальній частині дана характеристика методам захисту інформації в базах даних.

У роботі досліджено методи захисту інформації в базах даних на рівнях баз даних та web-додатку. Проведено аналіз методів захисту інформації в базах даних. Запропоновано рішення захисту інформації в базі даних MongoDB. Розроблено методику захисту конфіденційності інформації в базах даних MongoDB.

В економічному розділі визначено економічну ефективність та строк окупності затрат.

Практичне значення роботи полягає у можливості використовувати методику для різних баз даних для підвищення захисту інформації.

Результати здійснених у дипломній роботі (проекті) досліджень можуть бути використані в конфігурації серверу для підвищення безпеки інформації в базах даних.

Наукова новизна дослідження полягає у тому, що вперше створена методика на основі синтезу найефективніших та доступних рішень на різних рівнях захисту інформації в базах даних.

NoSQL-ІН’ЄКЦІЇ, MongoDB, КОНФІДЕНЦІЙНІСТЬ.

Реферат

Пояснительная записка: 91 с., 37 рис., 7 табл., 4 приложений, 21 источников.

Объект исследования: процесс внедрения NoSQL-инъекции во время запроса пользователя к базе данных.

Цель работы (проекта): повышение защиты конфиденциальности информации баз данных от актуальных атак типа NoSQL-инъекций.

Методы исследования: методы сравнения, индукции и дедукции, системного подхода, анализа и синтеза.

В специальной части дана характеристика методам защиты информации в базах данных.

В работе исследованы методы защиты информации в базах данных на уровнях баз данных и web-приложения. Проведен анализ методов защиты информации в базах данных. Предложено решение защиты информации в базе данных MongoDB. Разработана методика защиты конфиденциальности информации в базах данных MongoDB.

В экономическом разделе определена экономическая эффективность и срок окупаемости затрат.

Практическое значение работы состоит в возможности использовать методику для различных баз данных для повышения защиты информации.

Результаты проведенных в дипломной работе (проекте) исследований могут быть использованы в конфигурации сервера для повышения безопасности информации в базах данных.

Научная новизна исследования заключается в том, что впервые создана методика на основе синтеза эффективных и доступных решений на различных уровнях защиты информации в базах данных.

NoSQL-инъекции, MongoDB, КОНФИДЕНЦИАЛЬНОСТЬ.

ABSTRACT

Explanatory note: 91 p., 37 figures, 7 tables, 4 supplements, 21 sources.

The object of the research: the process of implementing sql-injection at the user's request to the database.

The purpose of the project (project): to increase the protection of confidentiality of information databases from actual attacks such as NoSQL injection.

Methods of research: methods of comparison, induction and deduction, system approach, analysis and synthesis.

In the special section, this is a description of the methods for protecting information in databases.

The methods of data protection in databases at the levels of databases and webapplications are investigated in the work. The analysis of data protection methods in databases is carried out. The decision to protect information in the MongoDB database is proposed. The technique of protection of confidentiality of information in MongoDB databases is developed.

The economic section defines economic efficiency and the payback period.

The practical value of the work is the ability to use a methodology for different databases to enhance information security.

The results of the thesis (project) research can be used in the server configuration to improve the security of information in databases

The scientific novelty of the research is that for the first time a method has been created based on the synthesis of the most effective and available solutions at different levels of information security in databases.

SQL INJECTION, ORACLE, SYBASE, CONFIDENTIALITY.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

БД – база даних;

СУБД – система управління базами даних;

РСУБД – реляційна система управління базами даних;

ОЗУ – оперативна пам'ять;

НСД – несанкціонований доступ;

ЦП – центральний процесор;

SSD – твердотільний жорсткий диск.

ЗМІСТ

ВСТУП

Актуальність. Безпека баз даних залишається одним з найбільш важливих аспектів інформаційної безпеки. Доступ до корпоративної бази даних надає зловмисникові контроль над найбільш важливими даними. Атаки з використанням SQL-ін'єкцій вставляють шкідливий код в оператори, що передаються додатком на рівень бази даних. Це дозволяє зловмисникові робити з даними практично все, включаючи доступ до несанкціонованих даних, а

також зміна, видалення і вставку нових даних. Незважаючи на те, що використання SQL - ін'єкцій неухильно знижувалося протягом багатьох років завдяки безпечним структурам і підвищення обізнаності, воно як і раніше залишається ключовим засобом використання вразливостей систем. Останнім часом бази даних NoSQL стають все більш популярними. Деякі з цих баз даних NoSQL використовують різні мови запитів, що робить традиційні методи впровадження SQL неактуальними. На сьогоднішній день найбільша кількість атак, яке припадають на нереляційні бази даних є - ін'єкційні атаки. Цей тип атаки був визнаний найнебезпечнішим в області зв'язків з базами даних. Те, що NoSQL не використовує мову SQL в запитах, не означає, що він застрахований від загрози ін'єкційних атак. Ці бази даних як і раніше потенційно вразливі для атак з використанням ін'єкцій, навіть якщо вони не використовують традиційний синтаксис SQL.

Метою дипломної роботи є підвищення захисту конфіденційної інформації баз даних від актуальних атак типу NoSQL - ін'єкцій.

У роботі були поставлені наступні завдання:

- дослідити вразливости бази даних MongoDB на рівні баз;

- дослідити вразливости бази даних MongoDB на рівні web - додатку;

- дослідити методи та практики захисту інформації в базах на рівні баз даних;

- дослідити методи та практики захисту інформації в базах на рівні webдодатку;

- створити додаткові методи захисту.

Об’єкт дослідження – процес впровадження NoSQL-ін’єкції під час запиту користувача до бази даних; Предмет дослідження – стійкість бази даних до атаки через NoSQL-ін’єкції.

При формуванні методики захисту конфіденційності інформації від NoSQL-ін’єкцій використовувався метод синтезу.

У спеціальній частині дана характеристика методам захисту інформації в базах даних.

В економічному розділі визначено економічну ефективність та строк окупності затрат.

Наукова новизна дослідження полягає у тому, що вперше створена методика на основі синтезу найефективніших та доступних рішень на різних рівнях захисту інформації в базах даних.

Практичне значення роботи полягає у можливості використовувати методику для різних нереляційних баз даних для підвищення захисту інформації.

РОЗДІЛ 1. БАЗОВІ ПОНЯТТЯ ТА ПРОБЛЕМАТИКА НЕРЕЛЯЦІЙНИХ БАЗ ДАНИХ

На сьогодні технології захисту інформації розвиваються дуже швидко. Яскравий приклад — прогрес NoSQL (Notionally structured query language)-технологій, що з’являються на зміну відомим реляційним базам даних. Синонімом NoSQL стали величезні обсяги даних, лінійна масштабованість,

кластери, відмовостійкість, нереляційність. Це стосується сфери NoSQL, де безліч технологій є не стільки прямою заміною більш традиційних механізмів зберігання інформації, скільки вирішенням спеціальних проблем, додаючи те, що можна очікувати від традиційних систем. Водночас, виробники більшості баз даних історично намагалися позиціонувати свій софт, як рішення «все в одному», NoSQL прагне до меншого рівня відповідальності — коли для певних завдань може бути обраний такий інструмент, який би вирішував саме це завдання щонайкраще. Приміром, NoSQL-стек може ефективно використовувати реляційні бази даних, наприклад MySQL, проте він також може містити в себе Redis — для організації зберігання записів key-value або Hadoop — для інтенсивної обробки даних. Інакше кажучи, NoSQL — це відкрита технологія, що складається з альтернативних, існуючих і додаткових шаблонів управління даними. Однією з найбільш розповсюджених NoSQLБД (база даних) є MongoDB, яка є документорієнтованою СУБД. Її можна розглядати, як альтернативу реляційним СУБД. Подібно реляційним СУБД, вона також може виграшно доповнюватися більш спеціалізованими NoSQL рішеннями. У MongoDB є як переваги, так і недоліки.

1.1 Постановка проблеми

6 червня 2012 р., коли компанія-розробник MongoDB 10gen розпочала тривале співробітництво з корпорацією Microsoft, яка надала MongoDB в обслуговування хмару Microsoft Azure. На додаток до розширення опцій хмари і хостингу, доступних розробникам MongoDB, цей крок об’єднав можливості бази даних NoSQL з технологіями Microsoft, включно з Microsoft Azure, NET та інші технології з відкритим вихідним кодом, що підтримує Microsoft. Після такого розширення MongoDB здобула масове розповсюдження серед користувачів. За даними офіційного сайту компанії на даний час має більш ніж 40 мільйонів завантажень, тисячи клієнтів, і більш ніж 1000 партнерів, а саме:

EA Sports, BuzzFeed, Adobe, MTV, CISCO, NBC Universal, MetLife, Google, Nokia, NewYorkTimes, Bosch, Facebook, Ebay, WashingtonPost, Forbes та багато інших. У результаті кількість збереженої та оброблюваної інформації збільшилась у десятки тисяч разів, що змусило розробників поліпшити програмний код. Такі обсяги інформації привернули увагу зловмисників (хакерів). Фахівці з безпеки інформації компанії Binary Edge довели, що у мережі виявлено 39134 сервера із незахищеними базами MongoDB. Загальний розмір баз даних — 619,8 Тбайт. Серед тисяч баз даних без аутентифікації виявлено 347 сервера з назвою «Інформація видалена, тому що ти не захистив базу паролем». Інший випадок теж вказує на «мінуси» БД. Троє студентів з Центру IT-безпеки, конфіденційності та звітності (CISPA) Саарського університету виявили 39890 баз даних MongoDB, доступних через Інтернет. Студенти використовували для пошуку відомий пошуковик Shodan, який сканує порти та індексує інформацію, недоступну через інші пошукові системи. Зокрема, шукали сервери з відкритим портом TCP 27017, який вказаний за замовчуванням у конфігурації MongoDB. Найбільші знахідки — БД одного з французьких Інтернет-провайдерів і оператора стільникового зв'язку з адресами і телефонами мільйонів клієнтів, а також база німецького Інтернет-магазину, яка містить платіжну інформацію. Ці випадки показують, що інформація в MongoDB не є стовідсотково захищеною, що зумовлено недбалістю системних адміністраторів, які нехтують усіма методами захисту інформації в БД, що створила компанія 10gen. Також варто зазначити порушення цілісності та можливості несанкціонованого доступу до MongoDB, це:

- використання ін'єкцій у регулярних виразах через параметри запиту;
- можливість виконувати «скрипти» на сервері, якщо на ньому встановлена NoSQL-СУБД (MongoDB дозволяє запускати JavaScript-код, тобто впровадження JavaScript-ін'єкцій) також підвидом є обхід аутентифікації;
- JSON-ін'єкції;

Враховуючи перелічені вище фактори та методи НСД (несанкціонованого доступу), постає яскраво виражена проблема правильного використання захисних механізмів MongoDB, а також впровадження додаткових методів захисту.

1.2 Статистика використання реляційних і нереляційних баз даних

DB - Engines охоплює понад 200 систем баз даних, незмінно оцінює MongoDB як найпопулярнішу базу даних NoSQL з місяця в місяць.

Gartner, шановна дослідницька фірма, нещодавно визнала MongoDB лідером в [11] своєму звіті Magic Quadrant по системам оперативного управління базами даних з точки зору повноти бачення і здатності до виконання.

На додаток цим стороннім джерелам популярність MongoDB очевидна в кількості завантажень програмного забезпечення: в даний час 40 мільйонів завантажень, які продовжують рости.

Серед реляційних баз даних, на сьогоднішній день, найпопулярнішою залишається MySQL.

Stack Overflow опублікував результати свого щорічного опитування серед розробників. В цьому році в опитуванні взяло участь більше 100 000 [4] респондентів, в той час як в 2017 році це число складало: 64 000. На даний момент - це найбільше в світі опитування розробників. За результатами опитування розробників StackOverflow наочно видно відсоток використання реляційних і нереляційних баз даних.

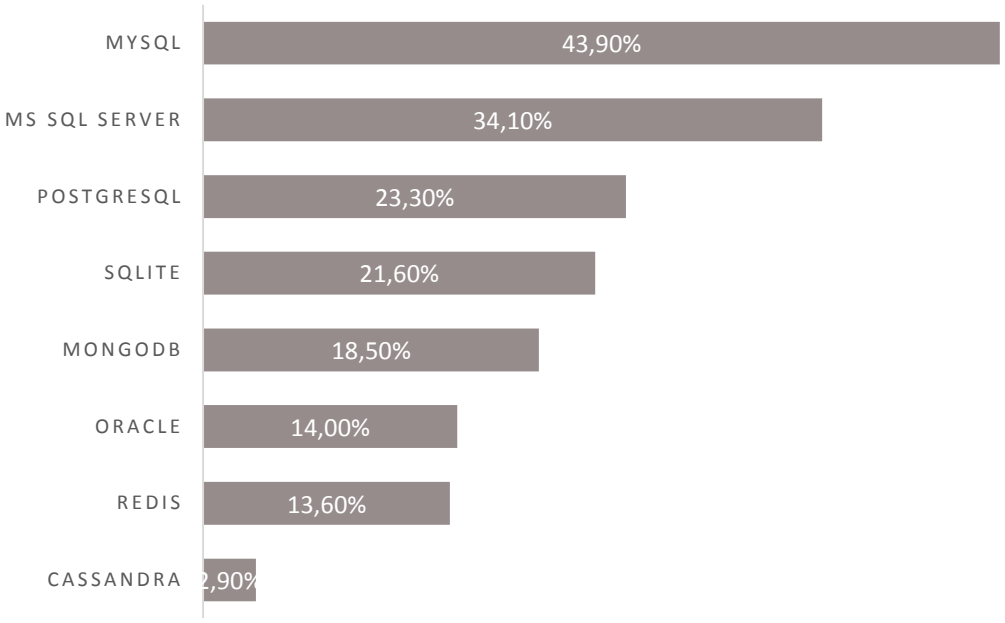


Рисунок 1.1 – Графік популярності баз даних за результатами опитування розробників StackOverflow за 2017 рік

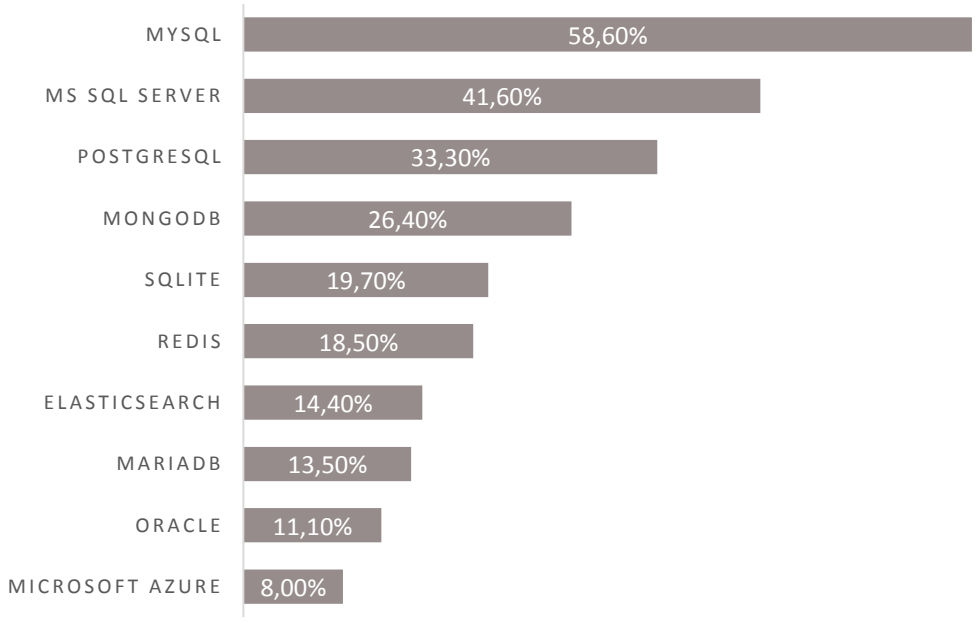


Рисунок 1.2 – Графік популярності баз даних за результатами опитування розробників StackOverflow за 2018 рік

Грунтуючись на результатах опитування можна сказати що:

1) Нові технології починають завойовувати частку ринку в світі баз даних - MongoDB (перший випуск у 2009 році), Cassandra (перший випуск у 2008 році), Redis (перший випуск у 2009 році).

2) Найпопулярніша база даних - MySQL. В цьому році більше половини респондентів (58,7%) використовують MySQL. Схоже, що бази даних RDBMS та, зокрема, MySQL збережуться тут як мінімум на декілька років.

3) SQL Server також має велику частку ринку (41,6%). Це також збільшення в порівнянні з минулим роком (34,1%). Microsoft просуває SQL Server зі своїми Windows-серверів, і таким чином завойовує значну частку ринку.

4) PostgreSQL набирає обертів в останні кілька років. Розробники, що працюють з Postgres, задоволені продуктом, як з точки зору можливостей, так і продуктивності.

5) Майже 1/4 всіх програмістів (19,7%) використовують SQLite, яка представляє собою полегшену базу даних SQL, засновану на одному файлі. Ця база даних користується великою популярністю в невеликих настільних додатках і додатках, де потрібно вбудована база даних (наприклад, додатки для мобільних телефонів).

1.3 Порівняння SQL і NoSQL рішень

Бази даних SQL використовують мову структурованих запитів (SQL) для визначення і обробки даних. З одного боку, SQL є одним з найбільш універсальних і широко використовуваних варіантів, що робить його безпечним вибором і особливо добре підходить для складних запитів. З іншого боку, це може бути обмеженням. SQL вимагає, щоб використовували зумовлені схеми для визначення структури даних, перш ніж працювати з ними. Крім того, всі дані повинні мати однакову структуру. Це може зажадати значної попередньої підготовки, а також, це може означати, що зміна структури

може бути складною і руйнівною для всієї системи. База даних NoSQL, з іншого боку, має динамічну [18] схему для неструктурованих даних, дані зберігаються різними способами: вони можуть бути орієнтовані на стовпці, орієнтовані на документи, засновані на графіку або організовані як сховище KeyValue. Ця гнучкість означає, що:

- Можна створювати документи без попереднього визначення їх структури;
- Кожен документ може мати свою унікальну структуру;
- Синтаксис може варіюватися від бази до бази даних;
- Можна додавати поля, по ходу подій.

Мова запитів. SQL (MySQL) [20] та NoSQL (MongoDB) [19] мають багату мову запитів. Порівняємо мову запитів двох популярних представників різних типів баз даних.

Таблиця 1.1 – Порівняння мови запитів SQL і NoSQL баз даних

Запрос	SQL (MySQL)	NoSQL (MongoDB)
Вставить объект	INSERT INTO users (user_id, age, status) VALUES ('bcd001', 45, 'A')	db.users.insert({ user_id: 'bcd001', age: 45, status: 'A' })
Найти объект	SELECT * FROM users	db.users.find()
Обновить объект	UPDATE users SET status = 'C' WHERE age > 25	db.users.update({ age: { $gt: 25 } }, { $set: { status: 'C' } }, { multi: true })

Масштабованість. У більшості випадків бази даних SQL є вертикально масштабованими, що означає, що можливо збільшити навантаження на один сервер, збільшивши таким чином, як ЦП, ОЗУ або SSD. З іншого боку, бази даних NoSQL масштабируются по горизонталі. Це означає, що можна [15] обробляти більший трафік шляхом поділу або додавання інших серверів в базу даних NoSQL. В кінцевому підсумку може стати більше і потужніше, що робить бази даних NoSQL кращим вибором для великих або постійно мінливих наборів даних.

Бази даних Structure SQL засновані на таблицях, [6] а бази даних NoSQL - на основі документів, пар ключ-значення, графових баз даних або сховищ з широкими стовпцями. Це робить реляційні бази даних SQL кращим варіантом для додатків, що вимагають багаторядкових транзакцій, таких як система обліку, або для застарілих систем, створених для реляційної структури. Прикладами баз даних SQL є - MySQL, Oracle, PostgreSQL і Microsoft SQL Server. Прикладами баз даних NoSQL є - MongoDB, BigTable, Redis, RavenDB Cassandra, HBase, Neo4j і CouchDB.

Функціональні відмінності між MySQL (SQL) і MongoDB (NoSQL)

MySQL: реляційна база даних SQL. Нижче наведені деякі переваги і переваги MySQL:

- Зрілість: MySQL - надзвичайно зріла база даних, що представляє величезне співтовариство, велике тестування і чималу стабільність.

- Сумісність: MySQL доступний для всіх основних платформ, включаючи Linux, Windows, Mac, BSD і Solaris. Він також має коннектори з такими мовами, як Node. js, Ruby, C #, C ++, Java, Perl, Python і PHP, що означає, що він не обмежується мовою запитів SQL.

- Економічно ефективний: безкоштовні бази даних з відкритим вихідним кодом.

- Реплицируемой: база даних MySQL може бути реплицирована на кілька вузлів, що означає, що робоче навантаження може бути зменшено, а масштабованість і доступність додатка може бути збільшена.

- Поділ: хоча поділ неможливо в більшості баз даних SQL, це можливо зробити на серверах MySQL, що є вигідним.

MongoDB: нереляційних база даних NoSQL. Нижче наводяться деякі переваги і переваги MongoDB:

- Динамічна схема. Дає гнучкість в зміні схеми даних без зміни будь-яких існуючих даних.

- Масштабованість: MongoDB масштабується по горизонталі, що допомагає знизити навантаження.

- Керованість: база даних не вимагає адміністратора бази даних. Так як це досить зручно для користувача, його можуть використовувати як розробники, так і адміністратори.

- Швидкість: висока продуктивність для простих запитів.

- Гнучкість: є можливість додавати нові стовпці або поля в MongoDB, не впливаючи на існуючі рядки або продуктивність програми.

Організації всіх розмірів впроваджують MongoDB, тому що це дозволяє їм швидше створювати додатки, обробляти різноманітні типи даних і більш ефективно управляти додатками в масштабі. Розробка спрощується, оскільки документи MongoDB природним чином відображаються на сучасних об'єктно-орієнтованих мовах програмування. Використання MongoDB видаляє шар складного об'єктно-реляційного відображення (ORM), який переводить об'єкти в коді в реляційні таблиці. Гнучка модель даних MongoDB також означає, що схема бази даних може розвиватися відповідно до бізнес-вимог. MongoDB також можна масштабувати в межах декількох розподілених центрів обробки даних, забезпечуючи нові рівні доступності та масштабованості, раніше недоступні для реляційних баз даних, таких як MySQL. У міру зростання обсягу даних MongoDB легко масштабується без простоїв і без зміни програми.

І напроти, для досягнення масштабу з MySQL часто потрібні значні, нестандартні інженерні роботи.

1.4 Базові поняття NoSQL

Термін NoSQL вперше був використаний в 90-х роках, проте, популярність він набрав в 2009 році. Під цим терміном малася [14] на увазі база даних з відкритим вихідним кодом, де дані зберігалися в ASCII файлах, а замість специфічної мови запиту до даних використовувала скрипти командної оболонки. На сьогоднішній день в середовищі NoSQL рішень все змінилося.

На зустрічі, що відбулася в Сан-Франциско в 2009 році метою було обговорення новинок в сфері IT. Тоді термін «NoSQL» запропонований Еріком Евансом вжився і став часто використовуваним в сфері зберігання та обробки даних по сьогоднішній день. Хоча, варто зауважити, що новий термін не планували виносити за рамки тієї зустрічі, йому не надавали смислового навантаження, однак, він стрімко поширився по інтернет мережі через Twitter - співтовариство і став назвою цілого напряму в IT-індустрії.

NoSQL - термін, який ще й досі не знайшов загальновизнаного визначення. Помилково вважається, що NoSQL розшифровується як No SQL, тобто ряд підходів зовсім не використовує кошти мови SQL - це твердження не вірне. Під розшифровкою прийнято розуміти Not Only SQL, що означає - не тільки SQL.

1.5 Причини виникнення NoSQL

Незважаючи на те, що реляційні бази даних є хорошим рішенням для проектування будь-яких моделей даних, це рішення, як правило, погано масштабується, а з ростом кількості даних, потреба в розподілених СУБД [1] різко зросла і це є однією з ключових причин виникнення NoSQL рішень.

Іншою причиною виникнення нереляційних СУБД стало потреба у швидкій роботі з даними. Реляційна база даних, як правило, має безліч компонентів для того щоб зручно було писати запит і було зручно працювати з будь-якої файлової системою і операційною системою, проте всі ці компоненти забезпечують зручність і універсальність накладають обмеження на швидкість доступу до даних і особливо це помітно на простих операціях, тому другою ключовою причиною поширення нереляційних СУБД є відмова від компонентів забезпечуючих універсальний доступ до даних, за допомогою мови SQL, для того щоб забезпечити швидкий доступ до даних і роботу з ними для певних класів досить, як правило, простих операцій. Незважаючи на те, що практично будь-яке завдання можна змоделювати за допомогою реляційної моделі даних, не завжди ця модель є досить простою для сприйняття людиною і нереляційні бази дані, як правило, можуть надати для специфічних завдань більш прості інструменти моделювання. Хоча специфічні завдання можуть зустрічатися зараз набагато частіше ніж раніше.

Прикладом такого завдання може послужити зберігання документа з деякими списками [8]. Для нереляційних баз нам знадобилося б основна сутність в якій зберігаються документи і сутності для кожного зі списку котрі містять всі їх можливі значення, в разі якби розглядался зв'язок «багато до багатьох» входження елементів списку в сам документ, то треба було ще додаткові таблиці, які забезпечували б зв'язок «багато до багатьох». У разі нереляційних баз даних можна змоделювати їх таким чином, щоб списки були вбудованими елементами конкретного документами, таким чином можна спростити саму схему моделі даних.

1.6 Особливості NoSQL-систем

Прості і гнучкі нереляційні моделі, призначені для вирішення специфічних завдань. Ці моделі підтримують масштабування по горизонталі.

Звичайні реляційні СУБД часто використовують вертикальне масштабування. Уявімо, що у нас є дуже велика таблиця, вона може бути розділена на деякі партіції або частини і кожні з цих частин може зберігатися на окремому пристрої зберігання. NoSQL системи часто використовують інший вид масштабування - по горизонталі, коли окремо зберігаються цілі стовпці. Також NoSQL системи позиціонують себе як забезпечують високу ступінь доступності даних.

NoSQL включає в себе безліч різних технологій баз даних, які були розроблені у відповідь на вимоги, що пред'являються при створенні сучасних додатків:

- Розробники працюють з додатками, які створюють масивні обсяги нових, швидко змінюються типів даних - структуровані, напівструктуровані, неструктуровані і поліморфні дані.

- Додатки, які колись обслуговували кінцеву аудиторію, тепер поставляються як послуги, які повинні бути завжди доступні, доступні з різних пристроїв і масштабуються глобально для мільйонів користувачів.

- В даний час організації звертаються до масштабування архітектурам з використанням програмного забезпечення з відкритим вихідним кодом і хмарних обчислень замість великих монолітних серверів.

Реляційні бази даних не були розроблені для того щоб справлятися з проблемами масштабованості і гнучкості, які стикаються з сучасними програмами, а також не були побудовані для використання наявної сьогодні технології зберігання і обробки товарів, з чим успішно справляються нереляційні бази даних.

До переваг відносять, в порівнянні з реляційними базами даних, масштабованість і забезпечення продуктивність, модель даних зачіпає кілька проблем, які реляційна модель, на сьогоднішній день, не вирішує:

- Великі обсяги швидко мінливих структурованих, напівструктурованих і неструктурованих даних

- Гнучкі спринти, швидка ітерація схеми і часті кодові натискання

- Гнучко використовується об'єктно-орієнтоване програмування

- Географічно розподілена масштабируемая архітектура замість дорогої монолітної архітектури

Особливістю NoSQL рішень є неструктурованість, її особливість полягає в тому, що документи або певні елементи можуть зберігати поля унікального типу без зміни схеми всієї таблиці. Також переважним властивістю є - відсутність обмеження з боку бази даних при зміні коду програми. А також NoSQL оперує цілісними об'єктами, на відміну від реляційних БД, орієнтованих на логічну бізнес-сутність і нормалізацію даних:

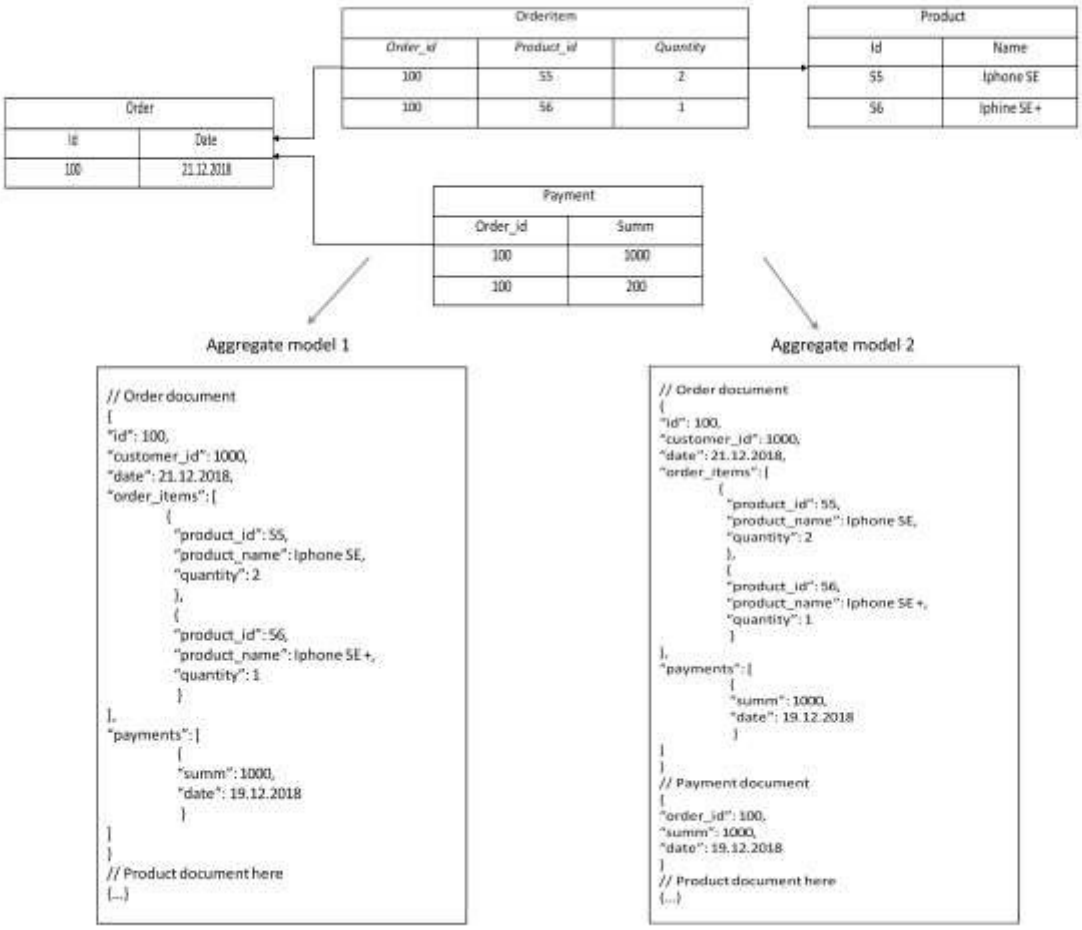


Рисунок 1.3 – Реляційна і нереляційних модель зберігання даних

У прикладі розглянуто дві агрегатні моделі для реляційної моделі "замовлення - позиції замовлення - платежі - продукт". У підсумку виходить один логічний об'єкт, різниця між ними тільки в способі об'єднання. Приклад ілюструє поширену практику проектування NoSQL БД - база повинна бути орієнтована на найчастіші запити до неї, а також максимально підкорятися вимогам програми. Якщо розмір денормалізірованних об'єктів великий, це може спровокувати виникнення проблем під час роботи з ними. Припустимо, є необхідність порахувати продану кількість товарів за рік, раніше замовлення було закріплено за позиціями і платежами одночасно, то хоч нам вся інформація і не буде потрібна, об'єкти цілком все одно доведеться витягти для підрахунку.

Таблиця 1.2 - Переваги і недоліки нормалізованих і даних у вигляді агрегатів

Нормализация данных	Данные в виде агрегатов
Целостность информации при обновлении (меняется информация в одной таблице, а не в нескольких); Ориентированность на широкий спектр запросов к данным.	Оптимизация только под определенный вид запросов; Сложности при обновлении денормализированных данных.
Неэффективна в распределенной среде; Низкая скорость чтения при использовании объединений (joins); Несоответствие объектной модели приложения физической структуре данных.	Лучший способ добиться большой скорости на чтение в распределенной среде; Возможность хранить физически объекты в том виде, в каком с ними работает приложение (легче кодировать и меньше ошибок при преобразовании);

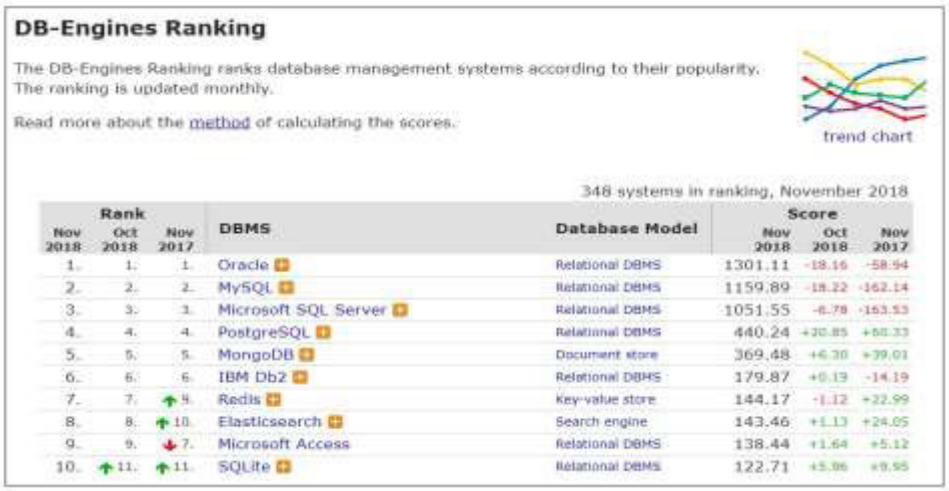
Бази даних NoSQL не потребують фіксованою структурою таблиць і не забезпечують повну підтримку ACID. Вони забезпечують в кінцевому підсумку узгодженість, а це означає, що дані будуть узгоджені протягом певного періоду часу.

1.7 Типы NoSQL баз даних

Існують різні підходи до класифікації баз даних NoSQL, кожна з яких має різні категорії і підкатегорії. Проте, основна класифікація, - це та, яка заснована на моделях даних. Можна класифікувати бази даних NoSQL відповідно до моделі даних в стовпець, документ, K - значення і графік [12]. Бази даних NoSQL засновані на нефункціональних категоріях і оцінці їх охоплення функцій. У таблиці наведено ці функції. Прийнято поділяти NoSQL СУБД на 4 види:

- Сховища типу «ключ-значення»
- Документні сховища
- Колоночного сховища
- Граф-орієнтовані сховища

Для того щоб знайти найбільш популярних представників кожного класу моделей можна скористатися рейтингом [21].



DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

trend chart

348 systems in ranking, November 2018

Rank Nov 2018	Rank Oct 2018	Rank Nov 2017	DBMS	Database Model	Score Nov 2018	Score Oct 2018	Score Nov 2017
1.	1.	1.	Oracle	Relational DBMS	1301.11	-18.16	-58.94
2.	2.	2.	MySQL	Relational DBMS	1159.89	-18.22	-162.14
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1051.55	-6.78	-163.53
4.	4.	4.	PostgreSQL	Relational DBMS	440.24	+20.85	+60.33
5.	5.	5.	MongoDB	Document store	369.48	+6.30	+39.01
6.	6.	6.	IBM Db2	Relational DBMS	179.87	+0.13	-14.19
7.	7.	↑ 9.	Redis	Key-value store	144.17	-1.12	+22.99
8.	8.	↑ 10.	Elasticsearch	Search engine	143.46	+1.13	+24.05
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	138.44	+1.64	+5.12
10.	↑ 11.	↑ 11.	SQLite	Relational DBMS	122.71	+5.06	+9.95

Рисунок 1.4 – Найбільш популярні типи баз даних
(https://db-engines.com/en/ranking)

На цьому рейтингу наведені оцінки за деякими формальними критеріями реально існуючих СУБД для кожного виду NoSQL систем.

Сховища типу «ключ-значення»

Найбільш популярний представник Redis. Основні характеристики цієї системи:

- Легка масштабованість

- Ефективний пошук за значенням ключа (ключем володіє кожен елемент даних, що зберігаються)

При цьому об'єкти, які ми зберігаємо, в такій моделі можуть мати досить складну структуру. Розробники таких систем декларують їх високу продуктивність. Недоліком є повна відсутність зв'язків між об'єктами звичне для реляційної моделі БД. Відповідно перевірка цілісності в БД переноситься на відповідальність користувача або додатки роботи з даними. Цей клас додатків може бути ефективно використаний для певного кола завдань (для зберігання зображень, відеороликів, файлів, програм і т. д).

Документні сховища

Вони зберігають об'єкти в спеціалізованому форматі, який називається JSON (Java Script Object Notation) або BSON (Binary JSON).

Зберігання таких об'єктів влаштовано таким чином. Рядок JSON містить або масив значень або об'єкт містить неврегульовану безліч пар ключ-значеннь. Масив полягає в квадратних дужках ([i]), об'єкт в фігурних дужках ({i}), самі елементи розділяються комами, пара ім'я / значення складається з імені поля, укладеного в лапки, після нього ставиться двокрапка і наводиться

саме значення. Значення в масиві або об'єкта може бути числом, рядком, а може мати рівні вкладеності, тобто містити вкладені масиви або об'єкти.

Приклад. Як можна зберігати інформацію про студента. На прикладі JSON.

```
{
" StudentId ": "54874",
" StudentName ": "Іванова Катерина",
"GroupNumber": 125,
"PhoneList": [
{ "Type": " будинок ", "phone":
                "24 733-45-24"},
        { " Type ": " mob ", " phone ":
"099 943-35-22"
]
}
```

У кожного студента зберігається ідентифікатор у вигляді номера заліковки, ПІБ, номера групи, також він містить вкладений масив, що складається з телефонів, для кожного телефона зберігається тип телефон та його номер.

Основні характеристики цієї системи полягають в тому, що вони сприймають атрибути простих типів а також вкладені об'єкти, можна будувати індекси по окремих полях, що дозволяє будувати досить складні запити, треба відзначити, що такі структури не підтримують основні принципи транзакцій характерних для реляційних СУБД, проте обробка кожного окремого документа зазвичай є атамарною і такі системи можуть ефективно застосовуватися в системах управління контентом, у видавничій справі, в документному пошуку і т.д.

Колоночні сховища (Wide column store)

Засновані на подання даних у вигляді таблиць, дроблення таблиці відбувається не за рахунок поділу по рядках, як прийнято в звичайних реляційних СУБД, а по стовпцях, для багатьох систем цього класу характерно SQL подібних мов досить високого рівня.

Граф-орієнтовані сховища (Graph DBMS)

Вони призначені для зберігання вузлів зв'язку та можуть зручно представляти зв'язку між ними, більшість таких систем дозволяють задавати вузли і пов'язувати між певним набором атрибутів. Можна вибирати вузли та зв'язку задаючи значення цих атрибутів. Вони підтримують алгоритми обходу графів і побудови маршрутів і ефективно використовуються для задач пов'язаних з аналізом соціальних мереж, вибором транспортного маршруту і т. д.

1.8 Традиційні РСУБД і властивості ACID

Реляційна база даних являє собою систему, яка зберігає дані, що зберігаються в стосунках, мають відносини один з одним через первинні і зовнішні ключі. Відносини типових програм називаються таблицями. Прикладами такого додатка є Microsoft SQL Server, СУБД Oracle і IBM DB 2 [3]. Мова SQL використовується для управління даними в системі реляційних баз даних. Область SQL Language включає в себе дані запиту з декількох з'єднань, вставки, видалення, оновлення та інших операцій. Через їх багатого набору функцій, таких як можливості запитів та управління транзакціями, вони, здавалося, були придатні практично для будь-якого можливого завдання. Одна з важливих особливостей РСУБД є надання властивостей ACID для виконання транзакцій. РСУБД підтримує властивості ACID, які важко підтримувати по розподілених даних.

РСУБД забезпечують ACID для паралельних транзакцій; крім того, вони забезпечують узгодженість і доступність по масштабованості згідно теоремі

CAP. З іншого боку, РСУБД не є найбільш підходящим рішенням в деяких сценаріях, таких як великі дані і широкомасштабні веб-додатки. Великі дані тягнуть за собою масштабованість і гнучкість, які не надаються РСУБД. Зокрема, важко здійснювати операції і спільні операції в розподіленої системі з використанням РСУБД. Як експоненціального зростання даних система повинна масштабироваться горизонтально. РСУБД важко отримати горизонтальну масштабованість. Для РСУБД потрібна фіксована структура таблиці, від якої не потрібно зміст великих даних.

1.9 CAP – теорема

Так як багато нереляційних СУБД пов'язані з розподіленими БД, для них справедлива CAP - теорема. В якій йдеться про те, що для нереляційних БД існує 3 характеристики, де можна вибрати тільки 2 з них на шкоду третій характеристиці [5].

Consistency. Консистентность даних. На різних вузлах знаходиться узгоджене між вузлами інформація. Наприклад, якщо провести транзакцію, то вона з'явиться відразу на всіх серверах.

Availability. Доступність системи. Відповідає за то, може система відповідати на всі запити, які отримує від користувачів. Наприклад, при збоях система може не відповідати на деякі запити.

Partition tolerance. Стійкість до поділу системи. У разі, якщо є розподілена система з декількох вузлів, чим більше вона, тим більша ймовірність того, що якийсь із вузлів буде відмовляти.

Якщо говорити про реляційну СУБД [13], то для них, 2 характеристики які вибираються із запропонованих. Це Consistency і Availability, тоді в разі, якщо один з вузлів розподіленої реляційної БД відмовляє, то це призведе до відмови всієї системи. Якщо ж розглядати нереляційні СУБД, то для них 2 характеристики Partition tolerance і Consistency, при цьому відмовив вузол, як

правило, стає недоступним і система не буде відповідати на запити користувачів, які звернулися із сайтом, але СУБД обробить це виняток і не буде враховувати даний вузол, в разі, якщо він довго не відповідає.

Згодом, характеристики CAP - теореми привели до BASE - архітектурі, тобто до правил за якими діють більшість сучасних нереляційних СУБД. Ці правила сформульовані більш м'яко ніж ті характеристики, які використовуються в CAP - теоремі.

1.10 Base-властивості

На відміну від РСУБД, яка забезпечує властивості ACID, NoSQL вводить те, що відомо, як властивості BASE. Підхід BASE, за словами Брюєра, який довів теорему CAP, втрачає властивості ACID узгодженості та ізоляції на користь «доступності та продуктивності». Абревіатура BASE складається з наступних характеристик: Basically Available, Soft-state, Eventual consistency. Продекларовані ними принципи формулюються так:

Soft state - система може знаходиться в нестійкому стані. Нестійким станом, можна називати, наприклад, незавершені транзакції або випадок, якщо в наборі з реплікації втрачено зв'язок з керуючим вузлом. Дані можуть в процесі обробки перебувати в неузгоджені стані, в цей момент їх можуть бачити користувачі.

Eventually consistent - дані, які будуть схильні до зміни коли-небудь узгоджуютись на всіх серверах. В кінцевому рахунку всі зміни повинні бути застосовані на всіх серверах.

На відміну від Consistency, Eventual Consistency дозволяє краще масштабувати базу даних оскільки немає необхідності чекати відповіді від усіх серверів для того щоб внести зміни відразу на всіх.

Разом додаток працює в основному весь час, але не обов'язково має бути послідовним весь цей час (м'яке стан (Soft - state)), в кінцевому підсумку буде в

деякому відомому стані (можлива узгодженість (Eventual consistency)). Властивості ACID забезпечують сувору узгодженість. Сувора узгодженість означає, що всі операції читання повинні повертати одні і ті ж дані з останньої завершеної операції запису. Така сувора узгодженість не може бути досягнута разом з доступністю і терпимістю в теоремі CAP.

Кінцева узгодженість означає, що всі операції читання можуть повертати різні дані з останньої завершеної операції запису, але з часом: «У стійкому стані» система, в кінцевому підсумку, поверне останнє записане значення. Тому клієнти можуть зіткнутися з неузгодженим станом даних по мірі відновлення оновлень. Наприклад, в реплікованих оновленнях бази даних можна перейти до одного вузла, який реплицирует останню версію до всіх інших вузлів, які містять репліку модифікованого набору даних, так що вузли репліки в кінцевому підсумку будуть мати останню версію. У таблиці наведено подібності та відмінності між RDBM і NoSQL.

Таблиця 1.3 - Відмінності між СУБД та базою даних NoSQL

Элемент	RDBMS	NoSQL
ACID / BASE	Предоставление свойств ACID	Предоставление свойств BASE
Структура таблицы	Требуется фиксированная структура таблицы	Не требуется фиксированная структура таблицы
Масштабирование	Вертикальная масштабирование	Горизонтальное масштабирование
консистенция	Сильная непротиворечивость	Слабая согласованность
Доступность	Имеется в наличии	Имеется в наличии

Возможности запроса	Большие возможности запросов	Низкие возможности запросов

Висновки до першого розділу

У першому розділі обгрунтована актуальність NoSQL-рішень в наш час. Статистично показано, що популярність нереляційних баз даних з року в рік зростає, а кількість таких баз даних тільки збільшується. Були проаналізовані реляційні і нереляційні бази даних їх відмінності, можливості і застосування. На сьогоднішній день найбільш частіш використовуваною СУБД є MongoDB за простоту у використанні, надійність і доступність.

Відомості про базові поняття NoSQL-рішень готують до практичної розв'язки питання про методику захисту конфіденційної інформації бази даних MongoDB від NoSQL-ін'екцій.

РОЗДІЛ 2. РОЗРОБКА МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ В БАЗІ ДАНИХ MONGODB ВІД NOSQL-ІН'ЄКЦІЙ

MongoDB - це система управління базами даних з відкритим вихідним кодом (СУБД), яка використовує модель бази даних, орієнтовану на документи, яка підтримує різні форми даних. Це одна з численних технологій нереляційних баз даних, яка виникла в середині 2000-х років під маркером NoSQL для використання у великих додатках даних і інших завданнях обробки, включаючи дані, які не підходять для жорсткої реляційної моделі. Замість використання таблиць і рядків, як в реляційних базах даних, архітектура MongoDB складається з колекцій і документів.

MongoDB була створена Дуайтом Мерріманом і Еліотом Горовицем, які зіткнулися з проблемами розвитку масштабованості з традиційними підходами до реляційних баз даних при створенні веб-додатків в DoubleClick, інтернет-рекламної компанії, яка тепер належить Google Inc. Назва бази даних була отримана з слова, що представляє ідею підтримки великого обсягу даних.

Мерриман і Горовіц допомогли сформувати 10Gen Inc. в 2007 році для комерціалізації MongoDB і супутнього програмного забезпечення. Компанія була перейменована в MongoDB Inc. в 2013 році і опублікована в жовтні 2017 року за символом тікера MDB.

СУБД була випущена в якості програмного забезпечення з відкритим вихідним кодом в 2009 році і доступна відповідно до умов версії 3.0 безкоштовної державної ліцензії GNU Affero від Free Software Foundation на додаток до комерційних ліцензіями, пропонованим MongoDB Inc.

На момент написання цієї статті серед інших користувачів страхова компанія MetLife використовує MongoDB для додатків для обслуговування клієнтів, сайт Craigslist використовує її для архівації даних, фізична лабораторія CERN використовує її для збору і виявлення даних, а The New York Times використовує MongoDB для підтримки програми для створення форм для фотографій.

2.1 Як влаштована база даних MongoDB

Запис в MongoDB - це документ, який являє собою структуру даних, що складається з пар полів і значень. Документи MongoDB схожі на об'єкти JavaScript Object Notation, але використовують варіант Binary JSON (BSON), який підтримує більше типів даних. Поля в документах на кшталт стовпцями в реляційної базі даних, а значення, які вони містять, можуть бути різними типами даних, включаючи інші документи, масиви і масиви документів, відповідно до керівництва користувача MongoDB.

Документи, які також повинні включати первинний ключ в якості унікального ідентифікатора, є базовою одиницею даних в MongoDB. Колекції містять набори документів і функції в якості еквівалента реляційних таблиць бази даних. Колекції можуть містити будь-які типи даних, але обмеження даних в колекції не може бути поширене в різних базах даних.

Mongo оболонка являє собою інтерактивний JavaScript інтерфейс для MongoDB, який дозволяє користувачам запитувати і оновлювати дані, а також здійснювати адміністративні операції. Оболонка є стандартною складовою дистрибутивів з відкритим вихідним кодом MongoDB. Як тільки MongoDB встановлений, користувачі підключають оболонку mongo до своїх екземплярів MongoDB.

Формат зберігання документів BSON і обмін даними, який використовується в MongoDB, надає двоичное уявлення JSON-подібних

документів. автоматична окантовка - ще одна ключова функція, яка дозволяє розподіляти дані в колекції MongoDB за кількома системами для горизонтальної масштабованості по мірі збільшення обсягів даних і збільшення пропускної здатності.

СУБД NoSQL використовує єдину основну архітектуру для узгодженості даних з вторинними БД, які підтримують копії первинної БД. Операції автоматично реплікуються в ці вторинні БД для автоматичного перемикання на резервний ресурс.

2.1.2 Переваги і недоліки MongoDB

Як і інші БД NoSQL, MongoDB не вимагає визначених схем і зберігає дані будь-якого типу. Це дає користувачам гнучкість у створенні будь-якої кількості полів в документі, що спрощує масштабування баз даних MongoDB порівняно з реляційними БД.

Однією з переваг використання документів є те, що ці об'єкти зіставляються з власними типами даних на декількох мовах програмування. Крім того, наявність вбудованих документів зменшує необхідність об'єднання баз даних, що може знизити витрати.

Основною функцією MongoDB є її горизонтальна масштабованість, що робить її корисною базою даних для компаній, що працюють з великими додатками даних. Крім того, обрис дозволяє базі даних розподіляти дані по кластеру машин.

MongoDB підтримує ряд механізмів зберігання даних і надає API-інтерфейс для зберігання даних, які дозволяють третім сторонам розробляти власні механізми зберігання для MongoDB.

СУБД також має вбудовані можливості агрегування, які дозволяють користувачам запускати код MapReduce безпосередньо в базі даних. MongoDB також включає в себе власну файлову систему GridFS, схоже

розподіленої файлової системи Hadoop (HDFS), в першу чергу для зберігання файлів, розмір яких перевищує розмір BSON розміром 16 МБ на документ. Ці подібності дозволяють використовувати MongoDB замість Hadoop, хоча програмне забезпечення бази даних інтегрується з Hadoop, Spark і іншими структурами обробки даних.

Хоча переваг багато, є деякі мінуси для MongoDB. З його автоматичної стратегією переходу на інший ресурс користувач налаштовує тільки один головний вузол в кластері MongoDB [9]. Якщо майстер виходить з ладу, підлеглий вузол автоматично перетвориться в новий майстер. Цей перемикач обіцяє безперервність, але це не миттєво - це може зайняти до хвилини. Для порівняння, база даних Cassandra NoSQL підтримує кілька основних вузлів, тому, якщо один з майстрів йде вниз, інший працює для високодоступних інфраструктури бази даних.

Єдиний майстер-вузол MongoDB також обмежує швидкість запису даних в БД. Запис даних повинна записуватися на головному пристрої, а запис нової інформації в базу даних обмежена пропускною здатністю цього головного вузла.

Ще одна потенційна проблема полягає в тому, що MongoDB не забезпечує повну кількість посилань цілісність з використанням обмежень зовнішнього ключа, що може вплинути на узгодженість даних. Крім того, аутентифікація користувача за замовчуванням не включена в базі даних MongoDB, що є ознакою популярності технології у розробників. Однак зловмисні хакери націлені на велику кількість нових систем MongoDB, що призвело до установки за замовчуванням, яка блокує мережеві підключення до БД, якщо вони не були налаштовані адміністратором БД.

2.1.3 Платформи MongoDB

MongoDB доступний в комерційних версіях через постачальника MongoDB Inc. MongoDB Community Edition – це версія з відкритим вихідним кодом, в той час як MongoDB Enterprise Server пропонує додаткові функції безпеки, механізм зберігання в пам'яті, функції адміністрування та аутентифікації і можливості моніторингу через Ops Manager.

Графічний користувальницький інтерфейс (GUI), так званий MongoDB Compass, дає користувачам можливість працювати зі структурою документа, вести запити, індексувати дані і багато іншого. Конектор MongoDB дозволяє користувачам підключати БД NoSQL до своїх інструментами бізнес-аналітики для візуалізації даних і створення звітів з використанням SQL-запитів.

Слідуючи по слідах інших постачальників баз даних NoSQL, MongoDB Inc. запустила хмарну базу даних в якості служби під назвою MongoDB Atlas в 2016 році. Atlas працює на AWS, Microsoft Azure і Google Cloud Platform. Зовсім недавно MongoDB випустила платформу Stitch для розробки додатків на Atlas MongoDB з планами поширити її на локальні бази даних.

Компанія також додала підтримку транзакцій ACID з декількома документами як частина MongoDB 4.0 в 2018 році. Відповідність властивостям ACID - атомарность, узгодженість, ізоляція і довговічність - в декількох документах розширюються типи транзакційних робочих навантажень, які MongoDB може обробляти з гарантованою точністю і надійністю.

2.2 NoSQL-ін'єкція

NoSQL-ін'єкція відноситься до ін'єкційної атаці через розміщення шкідливого коду в поля введення веб-сторінок. Атакуючий використовує перевагу погано відфільтрованих або неправильно екранованих символів в частині операторів NoSQL і вставляє довільні дані в рядок, яка в кінцевому

підсумку запускає механізми бази даних NoSQL (наприклад, форма входу в систему), як показано на рисунку 2.1.

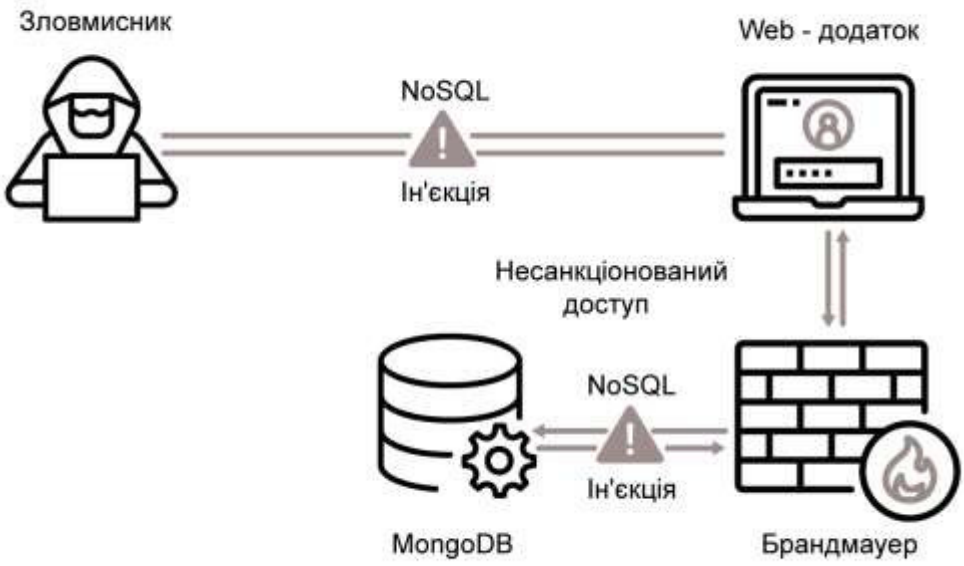


Рисунок 2.1 – Атака з використанням NoSQL в веб-додатках

Через вразливі веб-додатки зловмисник може отримати несанкціонований доступ до бази даних NoSQL і може змінити або видалити дані. В даний час майже всі бази даних NoSQL потенційно вразливі для атак-ін'єкцій NoSQL. NoSQL атака може проходити в веб-додатки через деякі методи, такі як ін'єкція, через елементи управління введенням веб-сторінки і файли cookie. Веб-форми надають деякий доступ до бек-енду БД NoSQL, дозволяю додавати або змінювати збережені дані. Будь-яка веб-форма, навіть проста форма входу в систему, форма реєстрації або вікно пошуку (де користувач може вводити або змінювати дані), може забезпечити доступ до внутрішньої бази даних NoSQL. Це означає, що існує висока ймовірність розгортання шкідливого коду і обходу брандмауера зловмисником.

Загальна причина того, що веб-додаток вразливий для впровадження NoSQL - неправильна фільтрація і погана перевірка користувальницького введення. Веб-форми досить поширені для збору даних від користувачів. Щоб

запобігти атакам, веб розробники повинні застосовувати належну фільтрацію / перевірку всіх форм.

Припустимо, що деякі запити PHP веб-додатків [7] виводять через екран ім'я користувача і пароль для доступа до приватної зоні. Додаток вибере ці значення і збере запит для відправки в базу даних NoSQL. Колекція «regusers» містить два документа авторизованих користувачів, як показано на рисунку 2.2.

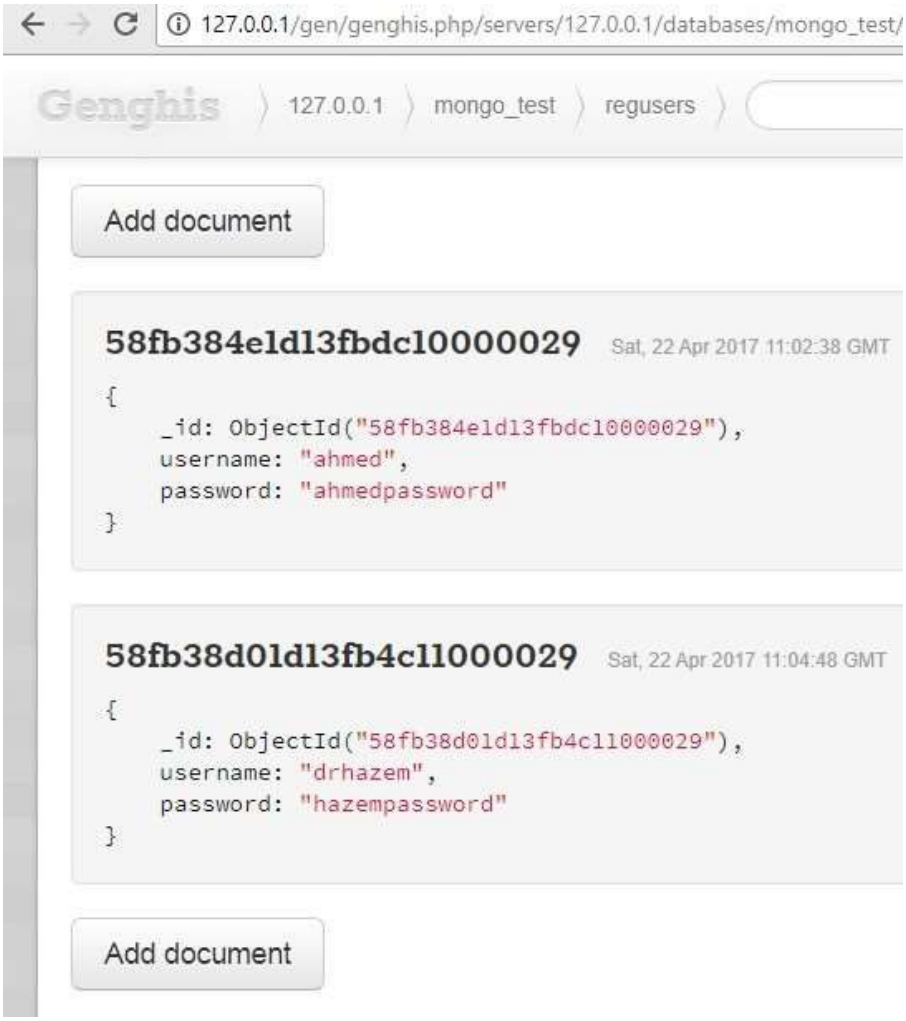


Рисунок 2.2 – Контейнер для об'єктів “regusers”

Веб-сторінка PHP може виглядати як на ричинку 2.3.

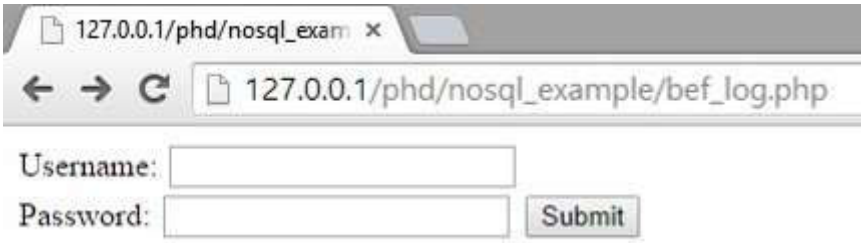


Рисунок 2.3 – Форма авторизації

Припустимо, що скрипт PHP вибирає документ з БД. Наступний рядок запиту NoSQL перевіряє комбінацію імені користувача та пароля, дійсні вони чи ні:

$ Collection-> find (array ("username" => $ _GET ['username'],

"Password" => $ _GET ['password']));

В цьому випадку, зловмисник може написати кілька текстів, які будуть відправлятися в базу даних NoSQL без будь-яких перевірок. У разі зловмисника, він міг написати в поле пароля рядок «$ ne» = 1, як показано на рисунку 2.4.

Рисунок 2.4 – Впровадження NoSQL в поле пароля.

В цьому випадку результуючий запит буде виглядати наступним чином:

$ Collection-> find (array ("username" => "drhazem",

"Password" => array ("$ ne" => 1)));

«$ Ne» вибирає документи, в яких значення поля не дорівнює «1». Таким чином, цей запит дасть той же результат, що якби користувач з правами адміністратора ввів свій пароль правильно. Згідно з цим, веб-додаток дозволить доступ до адміністративної області користувачеві, який не знає правильний пароль.

2.2.1 Query Selector Injection в MongoDB

MongoDB несприйнятлива до звичайним атакам SQL-ін'екцій, але вразлива для аналогічної атаки під назвою Query Selector Injection або NoSQL Injection, яка використовує вбудовані логічні оператори для зміни запитів. Для демонстрації скористаємося авторизаційної формою входу.

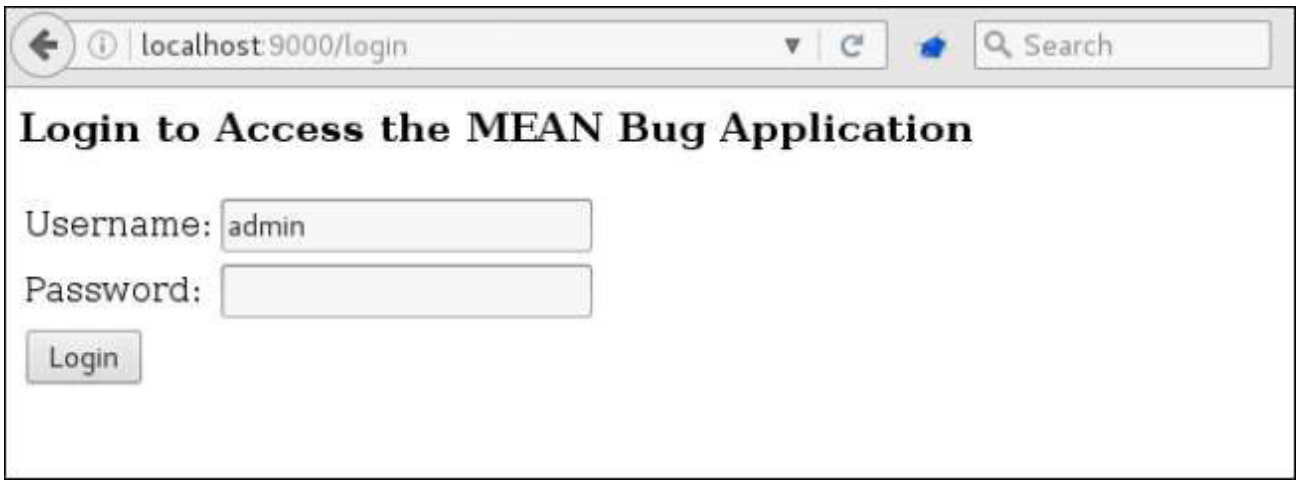


Рисунок 2.5 – Форма входу в систему

Використовуючи веб - проксі BurpSuite, перехопивши запит на вхід і змінивши POST пароль параметр для передачі [$ ne] = де $ N e є оператором запиту MongoDB.

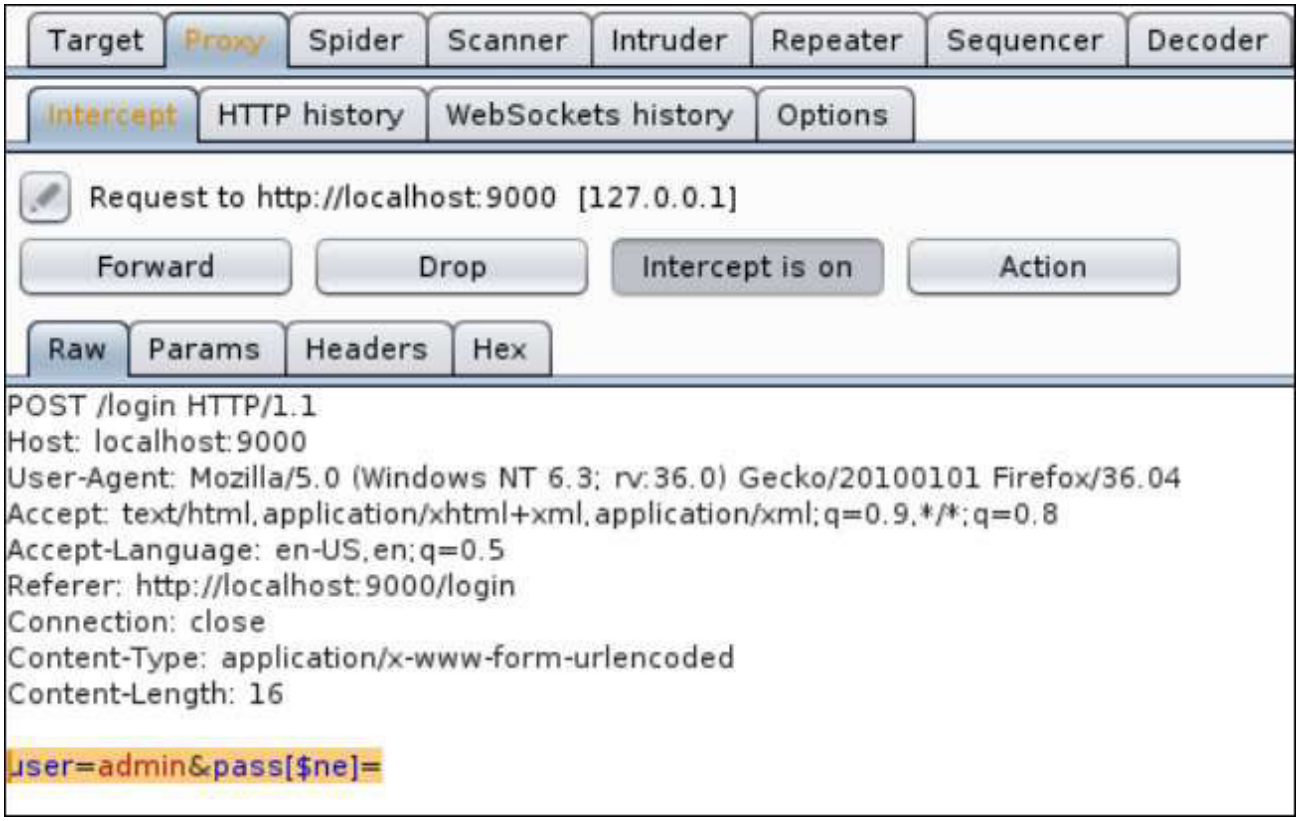


Рисунок 2.6 – Параметр форми, змінений за допомогою HTTP-проксі - інструменту BurpSuite

Після обробки тіла запиту, буде розглядатися об'єкт {User: 'admin', pass: {$ ne: null} і передаватися в запит MongoDB в блоці коду нижче.

```
db.collection('collection').findOne({username: user, password: pass, isActive:
true},function(err, result){
if(err){
    console.log('Query error...');
    return err;
}
if(result !== null){
    req.session.authenticated = true;
          res.redirect('/');
}
else
     res.redirect('/login?user='+user);
});
```

Рисунок 2.7 – Фрагемент коду запиту звернений до MongoDB

З вищевказаної корисним навантаженням запит відповідатиме першому імені користувача, яке відповідає «admin» з паролем, який не дорівнює нулю,

що дозволяє нам аутентифицироваться в додатку в якості адміністратора без надання дійсного пароля.

Якщо спробувати застосувати ту ж ін'єкцію в API/ Secure/ query API, можна буде побачити, що запит містить тіло JSON, яке, на відміну від простого тіла POST, не схильне до вразливості атрибутів при аналізі Express.

Рисунок 2.8 – Вихідний запит який містить тіло JSON

Однак, якщо ми змінимо заголовок Content - Type на x - www - form - urlencoded і змінимо тіло, щоб воно містило корисне навантаження, можна буде продовжити використовувати парсер Express, щоб вставити селектор запитів в об'єкт запиту. Це дозволяє виконати ще одну атаку на запит вибору запиту і отримати всі об'єкти з бази даних.

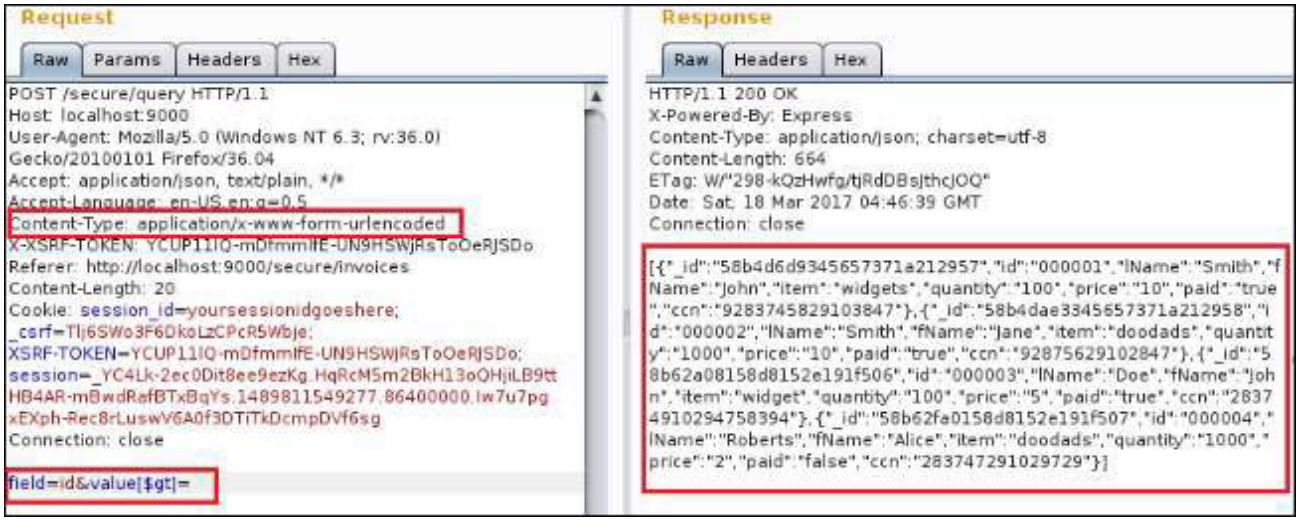


Рисунок 2.9 – Модифіковані тема Content-Type і тіло запиту для використання парсера Express

Найпростіший спосіб зменшити впровадження селектора запитів - передати значення в String перед запуском запиту. Запит вразливою аутентифікації з першого прикладу був змінений блоком коду нижче.

```
db.collection('collection').findOne({
       username: String(user),
       password: String(pass),
       isActive: true
},function(err, result){…});
```

Рисунок 2.10 – Фрагмент коду зміненогу запиту вразливої аутентіфікації

Якщо повторно відправити шкідливий запит, пароль буде підтверджено String '$ ne: null', а не з оцінкою фактичного оператора запиту, що не рівного. В результаті аутентифікація обхідний атаки більше не працює.

Рисунок 2.11 – Невдала активація атаки селектора запитів після введення в String

В якості альтернативи, інструмент перевірки введення, такий як mongo - sanitize, може використовуватися для відхилення будь-якого ключа запиту, який починається з символу $, Використовуваного для позначення операторів запиту.

2.2.2 Ін'єкції в регулярних виразах

Бази даних NoSQL дозволяють здійснювати пошук за допомогою регулярних виразів, при неправильному використанні якого можна завдати відчутної шкоди.

Для цього в MongoDB передбачений оператор $ regex [2]. Наприклад, запит на повернення всіх користувачів чий логін починається з приставки «ko», буде виглядати так:

db. users. find ({login: {$ regex: "^ ko"}})

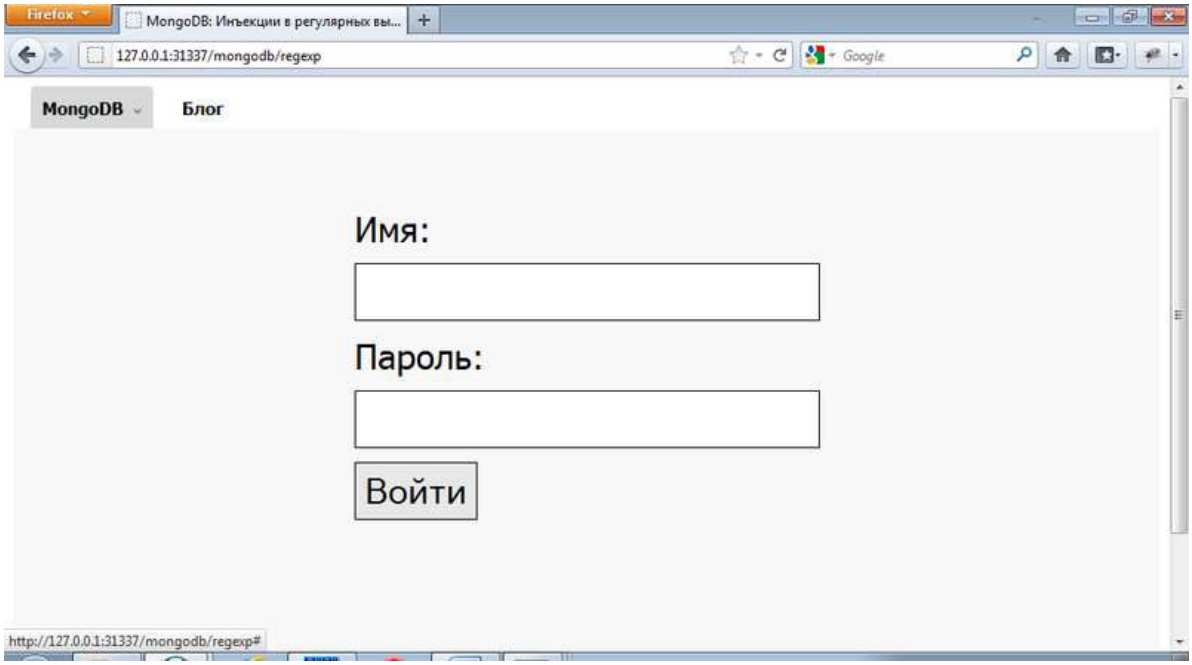


Рисунок 2.12 – Форма авторизації

Код авторизації з даної ін'єкцією виглядає так:

var regexpPwd = new RegExp ("^" + password, "i"); var loginParam = {login: login, password: regexpPwd};

У авторизаційної поля пароля досить ввести регулярний вираз, наприклад, [\ s \ S] *, змінна password не фільтрується, і логін root. В результаті БД виконає запит:

«Db. users. findOne ({login: 'root', password: / ^ [\ s \ S] * / i}) »

Успішна аутентифікація пройшла. Захиститися від подібної атаки досить просто. По-перше, завжди необхідно перевіряти вхідні дані, звідки б вони не надійшли - безпосередньо від користувача, або з зовнішнього файлу, або з бази даних. Перед використанням даних в програмі їх слід перевіряти. Подруге, використовувати регулярні вирази тільки в тих випадках, коли це дійсно

необхідно. Наприклад, наведений вище запит можна переписати ось таким чином:

db. users. findOne ({login: 'root', password: 'p @ ssw 0 rd'})

2.2.3 JSON -ін'екції

MongoDB не підтримує SQL, однак, БД не може не підтримувати своєї мови запиту. Замість SQL було вирішено використовувати JSON (BSON) [16] - текстовий формат обміну даними. При обліку відсутності фільтрації стає можливим здійснення атак - ін'єкцій, які називаються JSON -ін'екціями. Код авторизації з даної ін'єкцією виглядає так [2]:

var loginParam = eval ("({login: '" + login + "', password: '" + password + "'})");

Даний код перетворює текстове представлення об'єкта (запиту до MongoDB) в об'єкт. Об'єкт передається на сервер, після чого відбувається аутентифікація. Вразливістю цього коду є те, що вхідні дані не контролюються. Виходить, для успішного входу в систему досить ввести в поле логіна ім'я користувача «root '}) //» без введення пароля. Захиститися від подібної атаки можна якщо перевіряти всі дані з зовнішніх джерел. Наприклад, логін повинен відповідати регулярному виразу "^ [a - zA - Z] + $". Не застосовувати препарат функцію eval для роботи з JSON. У Node. js доступна функція JSON. parse, яка парсит вхідні рядок і створює об'єкт на її основі.

2.2.4 JavaScript -ін'єкція

В даному прикладі вразливість знаходиться в команді db. eval і в операторі $ where. Код авторизації з даної ін'єкцією виглядає так [2]:

var js = "this.login === '" + login + "' && this.password === '" + password + "'"; var loginParam = { "$ where": js};

Поле пароль і логін не перевіряються, тому можна виконати будь-який скрипт на сервері. Якщо ввести ім'я користувача « root '//», то після генерації скрипта і відправлення запиту на сервер можна буде успішно увійти в систему, це сталося через те що на сервері був сформований запит до БД даного типу:

'$ Where': 'this.login === \' root \ '// \' && this.password === \ '\' '}

«//» - позначення коментаря в JavaScript, підсумковий запит набуває вигляду « this. login === ' root ' ». В даному способі зловмисник, який увійшов в систему таким чином буде мати права «тільки на читання», тому скрипт модифікує дані не зможе бути виконаним. Однак, це можливо шляхом реалізації атаки через функцію db. eval.

var js = "function () {return db.users.findOne ({login: '" + login + "', password: '" + password + "'});}" db.eval (js);

Створення нового користувача :

'}), Db.users.insert ({login:' pen_test ', password:' pen_test '}), 1} //

Тільки що був згенерований новий користувач через якого був реалізований вхід в систему.

2.3 Рекомендовані методу захисту

У MongoDB реалізовано 5 етапів безпеки: аутентифікація, авторизація, шифрування, аудит, управління.

Аутентифікація LDAP

У базі даних спочатку присутні призначені для користувача ролі, за замовчуванням, які відключені. Недоліків є те, що тут немає таких механізмів як складність пароля, запит на зміну призначеного для користувача доступу під час редагування статусу, ротації за віком. Однак, це вирішується шляхом застосування LDAP. У більш пізньої версії MongoDB 3.2, призначені для користувача ролі зберігаються на окремих машинах, самих же користувачів база даних зберігає в LDAP. У версії MongoDB 3.4 надається можливість зберігання призначених для користувача ролей в LDAP.

Ролі користувачів

У MongoDB є можливість створювати призначені для користувача ролі власноруч прописуючи їм макроси, що конкретно їм дозволено робити. Основні впроваджені ролі:

Читання (read) - цей доступ, який присвоюється за замовчуванням - доступ в режимі читання даних;

Читання-редагування (readWrite) - доступ в режимі редагування даних;

Власник (dbOwner) - доступ в режимі додавання / видалення користувачів, створення призначених для користувача ролей, видача прав

доступу користувачам, крім цього включає режим readWrite і dbAdmin. Описані права поширюються виключно на один сервер бази даних;

Адміністрування будь-якої бази даних (dbAdminAnyDatabase) - режим dbAdmin поширений на всі бази даних без можливості адміністрування користувачів;

Супер-режим (root) - доступ в режимі суперкористувача без можливості змінювати системну колекцію.

Ручне настроювання користувальницьких ролей. В рамках ручної настройки можливо закріплювати конкретні дії на вибіркових ресурсах. Це дозволяє здійснити глобальний контроль дій в суспільстві MongoDB. Для настройки призначених для користувача ролей існує наступні типи ресурсів:

Db - визначити обрану базу даних.

Collection - визначити обраний набір документів

Cluster - визначає джерела метаданих або шардірованний кластер.

A nyResource - визначає повний доступ до всіх ресурсів

Наприклад, на випадок надзвичайних ситуацій створимо нову роль супер-користувача.

```
db = db.geSiblDB("admin");
db.createRole({
     role: "root",
     privileges:[{
          resource: {anyResource:true},
          actions: ['anyAction']
     }]
     roles:[]
});
db.createUser({
     user: "userRoot",
     pwd: "sfhjH78HDkw29^dga2",
     roles: ["root"]
})
```

Рисунок 2.13 – Створення ролі супер-користувача

Хід дій. У базі даних geSiblDB створимо призначену для користувача роль root на будь-який ресурс і дія. Далі, в цій же базі даних створимо користувача userRoot з паролем sfhjH78HDkw29 ^ dga2 і роллю root.

Застосування SSL

SSL застосовується в незахищених мережах для захисту даних. SSL забезпечує конфіденційність і аутентифікацію - дві ключові компоненти для захисту бази даних. З цими компонентами з'являється допоміжний рівень захисту.

Рекомендовано використовуват SSL для з'єднань между кластерами MongoDB та между Клієнтом та екземпляр MongoDB. Використання протоколу SSL не впливає на продуктивність та може захищати від атак.

Шифрування диска

У MongoDB є можливість шифрування даних в стані " in transit " і " at rest ." Дані, які зберігаються на диску, називаються - дані at - rest . Під шифруванням розуміється збереження цих даних в зашифрованому сховище. До переваг даного рівня шифрування варто віднести єдину процедуру шифрування сховища. Шифрування можна здійснити в такий спосіб:

- Зашифрувати весь диск;
- Зашифрувати виключно файли БД;
- Зашифрувати в додатку.

Слід шіфруваті та зберігаті конфіденційні дані.

Рекомендовано розміщуваті конфіденційні дані, такі як облікові записи, паролі, адреси електронної пошти, номери мобільних телефонів та ідентіфікаційні номери для зберігання у зашифрованному форматі. Використовувати міжнародні загальні алгоритми шифрування.

Завдяки цьому, навіть якщо хакери отримають дані, вони не зможуть зчітати конфіденційну інформацію. Збиток, таким чином, зводиться до мінімуму, оскількі зашифровані дані абсолютно незрозумілі.

Управління

Під управлінням розуміється можливість вводити складні стандарти в БД за рахунок перевірки документів. Вона, в свою чергу, визначає допустимі області документа для конкретної колекції. Перевірка документів здійснюються наступними командами:

CollMod - для перевірки визначає ключ;

Validator - для перевірки визначає параметри;

ValidationLevel -встановлює частоту включення валідатора;

ValidationAction - встановлює макрос при помилку перевірки.

Зменшення площі атаки мережі

Для зменшення затримок в середовищі БД пропонується розміщення монго-екземплярів на всіх хостах додатка. Це рішення переважно для продуктивності, але і є недоліком для безпеки, так як з кожним монго-екземпляром з'являється ще n -з'єднання укладають мережу. Тоді у зловмисника з'являється на n -З'єднання більше, які він може використовувати. Для цього запобігання, рекомендовано утримувати групу серверів БД поруч з додатками, яким дозволено посилатися до інших систем. Краще налаштувати звернення додатків виключно до серверів БД, що в свою чергу, виключить надмірну кількість з'єднань в мережі.

Інші методи захисту

Служба безпеки Harden MongoDB. Щоб зміцнити безпеку служб MongoDB рекомендовано дотримуватись наступних вказівок, виправляти вразливі місця та захищати безпеку додатків.

Якщо не додати жодних параметрів після активації служби MongoDB, автентифікація дозволу не виконується за замовчуванням. Користувачі, які зареєструвалися в службі, можуть виконувати будь-які дії (включаючи додавання, видалення, редагування, запити та інші дії високого ризику).

Причини вразливості. Після встановлення служби MongoDB, за замовчуванням створюється адміністраторська база даних. Адміністративна база даних порожня і інформація про дозвіл не зареєстрована. Навіть якщо параметр "-auth" додається на початку MongoDB, якщо користувач admin.system.users не містить користувачів [10], можливо виконувати будь-які дії без автентифікації, доки не буде додано користувача до admin.system.users. Метод затвердження перевіряє послуги аутентифікації та авторизації MongoDB лише після того, як користувач додається до admin.system.users.

Метод усилення проти вразливостей несанкціонованого доступу MongoDB

1) Змінити порт за замовчуванням. Змініть стандартний порт MongoDB (TCP 27017 за замовчуванням) на інший порт.

2) Не розгортати сервери MongoDB безпосередньо в Інтернеті чи DMZ. Використовувати групу безпеки або брандмауер локальної операційної системи, щоб керувати IP-адресами, яким дозволен доступ. Якщо база даних забезпечує лише послуги для серверів мережі, рекомендовано вимкнути публікацію служби MongoDB в Інтернеті. Група безпеки еквівалентна брандмауеру. Політика групової політики вхідного Інтернету за умовчанням - це доступ до всіх портів в Інтернеті. Видалити групу безпеки за умовчанням та заблокувати службу, додавши правило, що відхиляє весь доступ.

3) Використовувати параметр "-bind_ip".

Ця опція обмежує IP-адресу інтерфейсу прослуховування. Коли запускається MongoDB, використовувати --bind_ip 192.168.0.1, щоб запустити зв'язані IP-адреси, так щоб екземпляр бази даних прослуховував лише запити з 192.168.0.1.

4) Увімкнути автентифікацію входу на основні ролі.

Наприклад, ви можете створити користувача з ім'ям "супер" в базі даних адміністратора та встановити пароль "supWDxsf67% H".

Увійдіть у базу даних з автентифікацією disabled:

```
[mongodb@rac3 bin]$ ./mongo 127.0.0.1:27028 (Here the default port is modified.)
MongoDB shell version: 2.0.1
connecting to: 127.0.0.1:27028/test
```

Рисунок 2.14 – Підключення до бази даних

Перейдіть на базу даних адміністратора:

```
> use admin
switched to db admin
>
```

Рисунок 2.15 – Підключення до бази даних адміністратора

Перевірити або створити обліковий запис адміністратор

```
> db.addUser("supper", "supWDxsf67%H")или же
>db.createUser({user:"supper",pwd:"supWDxsf67%H",roles:["root"]})

{ "n" : 0, "connectionId" : 4, "err" : null, "ok" : 1 }
{
"user" : "supper",
"readOnly" : false,
"pwd" : "51a481f72b8b8218df9fee50b3737c44",
"_id" : ObjectId("4f2bc0d357a309043c6947a4")
}
```

Рисунок 2.16 – Перевірка та створення облдікового запису адміністратора

Обліковий запис адміністратора знаходиться у файлі system.users.

```
> db.getCollectionNames()
[ "system.indexes", "system.users", "system.version" ]
```

Рисунок 2.17 – Місцезнаходження записа адміністратора

Метод addUser не може використовуватися в версіях MongoDB пізніше V3. Замість цього слід використовувати db.createUser для створення нових користувачів. В цілях безпеки ім'я облікового запису не повинно бути загальним словом, а пароль повинен бути складним. Пароль може містити великі та малі літери, цифри та спеціальні символи. Не слід використовувати дату народження, ім'я, ідентифікаційний номер або загальні паролі.

Admin.system.users зберігає інформацію для користувачів з вищими правами користувачів, ніж користувачі в інших базах даних, і має суперпривілегії. Тому користувачі, створені в базі даних адміністратора, можуть виконувати операції з даними в інших базах даних MongoDB. У системі MongoDB база даних створюється суперкористувачем. Він може містить декілька користувачів, але кожен користувач може існувати лише в одній базі даних одночасно. Однак користувачі в різних базах даних можуть мати однакове ім'я. User1 в певній базі даних (наприклад, DB1) не може отримати доступ до бази даних DB2, але може отримати доступ до даних, створених іншими користувачами в DB1. Користувачі з однаковим ім'ям в різних базах даних не можуть входити в інші бази даних. Наприклад, якщо для DB1 та DB2 є user1, після того, як user1 входить до DB1, він не може одночасно входити в DB2. Користувачі, створені в базі даних адміністрування, мають суперпривілеї та можуть виконувати операції на будь-якому об'єкті даних у будь-якій базі даних в системі MongoDB. Для перевірки користувачів у базі даних використовувати db.auth (). Якщо перевірка була успішною, повернеться 1; інакше повернеться 0. Метод db.auth () може перевіряти лише інформацію

користувача бази даних, де він належить, і не може перевірити інформацію користувача в інших базах даних.

5) Вимкнути HTTP і RESTful порти.

MongoDB поставляється з HTTP-сервісом і підтримує RESTful інтерфейси. За замовчуванням MongoDB використовує порт за замовчуванням для прослуховування веб-служб, але зазвичай не вимагає віддаленого керування через Інтернет. Рекомендовано вимкнути цю опцію. Змініти файл конфігурації або вибрати параметр "-nohttpinterface" під час запуску.

6) Активізувати перевірку журналів.

Функція аудиту може використовуватися для запису всіх операцій, що виконуються користувачем в базі даних. Ці записи дозволяють системному адміністратору аналізувати, які події відбулися в базі даних в будь-який конкретний момент часу.

7) Резервне копіювання даних локально та віддалено.

Останній рівень захисту від загроз - це надійна та послідовна політика резервування.

```
>mongodump -h dbhost -d dbname -o dbdirectory
-h:
The address of the MongoDB server, such as 127.0.0.1.
You can also specify the port: 127.0.0.1:27017.
-d:
The database instance to be backed up, for example: test.
-o:
The storage location for the backup data, such as: c:\data\dump.
The directory must be created in advance. After the backup is
complete, the system automatically creates a test directory under
the dump directory to store the backup data for the database instance.
```

Рисунок 2.18 – Резервне копіювання MongoDB

Команда mongorestore для відновлення резервної копії data.mongodb.

```
Syntax
Script syntax for the mongorestore command is as follows:
>mongorestore -h dbhost -d dbname --directoryperdb dbdirectory
-h:
Address of the MongoDB server
-d:
The database instance to be restored, such as: test. The name can
be different from that of the backup, for example test2.
--directoryperdb:
Location of the backup data, for example c:\data\dump\test.
--drop:
During data restoration, the current data is deleted first, then
the backup data is restored. Therefore, once the backup is complete,
any data added or modified after the backup is deleted. Proceed
with caution when using this option.
```

Рисунок 2.19 – Відновлення даних MongoDB

Таблиця 2.4 - Параметри команди Mongodump

Синтаксис	Описание	Пример
mongodump —host HOST_NAME —port PORT_NUMBER	Ця команда архівує всі дані MongoDB.	mongodump —host w3cscholl.cc —port 27017
mongodump —dbpath DB_PATH —out BACKUP_DIRECTORY	-	mongodump —dbpath /data/db/ —out /data/backup/
mongodump —collection COLLECTION —db DB_NAME	Ця команда створює резервує вказану колекцію баз даних.	mongodump —collection mycol —db test

Резервна політика. Повна резервна копія: це найшвидший спосіб відновити всі дані, але витрати на ресурси є високими, і процес займає багато часу. Повна резервна копія + поступове резервне копіювання: швидший спосіб відновити всі дані, однак, якщо під час відновлення додаткових даних виникають проблеми, неможливо відновити все.

2.4 Випробувальний інструмент «NoSQL Racket»

В даний час є понад 225 баз даних NoSQL з веб-додатками. Кожен пропонує різні функції і обмеження. Отже, з'являється велика проблема, тому що немає спільної мови між веб-додатками і NoSQL БД. З цієї причини випробувальний інструмент пропонує загальне тестування механізму виявлення атак з використанням NoSQL без залежності від конкретного синтаксису і моделі даних. Створимо просту таблицю бази даних з ім'ям «Driverstbl». Таблиця «Driverstbl» містить всі рядки запиту форми і їх типи, такі як зарезервовані ключові слова, логічні оператори і реляційні оператори, як показано в таблиці 1.

Таблиця 2.5 - Driverstbl

NoSQL Database Type	String Type	Syntax
MongoDB	Reserved keywords (RK)	db
MongoDB	Reserved keywords (RK)	find
MongoDB	Reserved keywords (RK)	update
MongoDB	Operator(OP)	$and
MongoDB	Operator(OP)	$exists
MongoDB	Operator(OP)	$ne
CouchDB	Reserved keywords (RK)	getDatabaseInfos
CouchDB	Reserved keywords (RK)	getDoc
CouchDB	Reserved keywords (RK)	storeDoc
CouchDB	Operator(OP)	&&
CouchDB	Operator(OP)	in

Кожен оператор запиту коду і оператор запиту часу виконання перетворюються в формат порівняльних шаблонів в залежності від таблиці «Driverstbl», як показано на рисунку 2.20.

Інструмент тестування «NoSQL Racket» повертає масив, який містить кількість повторень для кожного рядка, що зберігається в таблиці «Driverstbl». Припустимо, що наступний PHP-скрипт в статичному стані коду є S1, а той же оператор коду в динамічному стані - S2:

S1: $ collection-> find (array ("username" => $ _GET ['username'],
"Password" => $ _GET ['password']));

S2: $ collection-> find (array ("username" => "drhazem",
"Password" => array ("$ ne" => 1)));

Згідно «Driverstbl», інструмент тестування «NoSQL Racket» генерує наступні шаблони для S1, S2:

S1 pattern: Array ([0] => Array ([0] => PK [1] => find [2] => 1) [1] => Array ([0] => PK [1] => array [2] => 1) [2] => Array ([0] => OP [1] => => [2] => 2)).

S2 pattern: Array ([0] => Array ([0] => PK [1] => find [2] => 1) [1] => Array ([0] => OP [1] => $ne [2] => 1) [2] => Array ([0] => PK [1] => array [2] => 2) [3] => Array ([0] => OP [1] => => [2] => 3)).

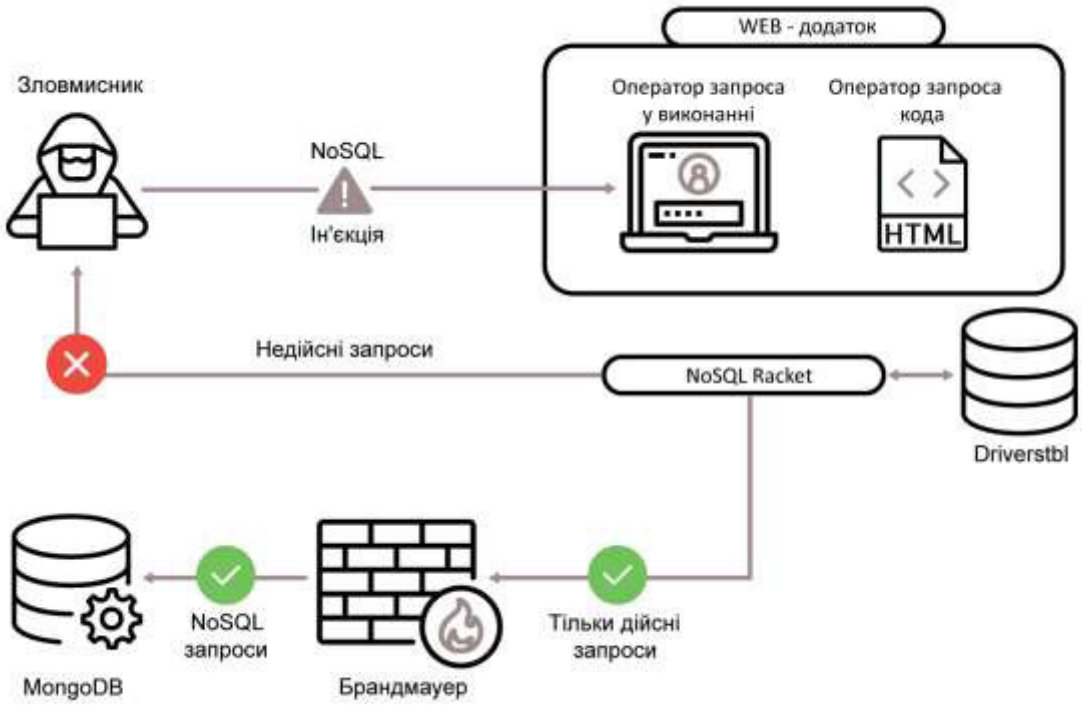


Рисунок 2.20 – Впровадження NoSQL в поле пароля

Після генерації шаблонів, для кожного коду оператора запиту (S1) і оператор запиту в робочому стані (S2), «NoSQL Racket» перевірить відповідність між згенерували шаблонами, як показано в наступному алгоритмі:

Крок 1: Отримати оператор запиту коду (S1) і оператор запиту часу виконання (S2).

Крок 2: Нехай DBT = тип бази даних NoSQL.

Крок 3: Підключіться до nosqldbs і виберіть всі «String Type» і «Syntax» з таблиці «Driverstbl», де NoSQL Database Type = DBT.

Крок 4: Згрупуйте і порахуйте слова в S1 і S2 відповідно до обраних даними в Кроці 3.

Крок 5: Генерація шаблонів для кожного оператора S1, S2.

Крок 6: встановіть Inj = 0

Крок 7: Для кожного елемента в шаблонах S1 і S2 повторюйте до кінця елементів.

Крок 7.1: Якщо S1 [i] НЕ дорівнює S2 [i], тоді встановити Inj = 1

Крок 7.2: Перейти до кроку 7.

Крок 8: Якщо Inj = 1, відобразити сторінку з помилкою і припинити роботу, інакше продовжити виконання і виконати запит.

Згідно з результатами зіставлення шаблонів і вхідних значень, існує два рішення: якщо шаблони збігаються, веб-додаток продовжить роботу, якщо шаблони не збігаються, веб-додаток буде перервано і запропонований алгоритм відобразить сторінку з помилкою.

Эксперименты и результаты

Щоб дослідити ефективність випробувального інструменту «NoSQL Racket», розглянемо здатність виявлення атак за допомогою порівняльного дослідження з найпотужнішими тестерами, наприклад, Netsparker, Vega і Skipfish. З іншого боку, будуть використовуватися чотири версії веб-сторінок в цьому порівняльному дослідженні, яке охоплює всі типи баз даних NoSQL, які засновані на документах, орієнтовані на стовпці і мають значення ключа. Також перевіримо продуктивність запропонованого інструменту «NoSQL Racket» за допомогою інструменту тестування продуктивності під назвою «LoadComplete».

A. Порівняльне дослідження здатності виявлення

Для вивчення використовуються чотири веб-скрипта PHP, і всі вони уразливі для атаки з використанням NoSQL. Ці сценарії виконуються після відправки після натискання кнопки входу користувача. Коли користувач відправляє правильне ім'я користувача і пароль для кожної бази даних NoSQL, висновком буде «Авторизований користувач», якщо будь-яка з полів або обидва поля невірні, висновок буде «Ви не авторизовані».

1) Результати визначення можливості MongoDB:

У MongoDB можна ввести ключові слова NoSQL в представлені дані з веб-сторінки входу в систему. Це може виглядати, наприклад, так

http://127.0.0.1/phd/MongoDB/after_log.php?user=ahmed & pass [$ ne] = 1 & sbumit1 = Відправити

«$ Ne» вибирає документи, в яких значення поля не дорівнює (тобто! =) Значенню «1». Таким чином, цей запит дасть той же результат, як якщо б користувач-адміністратор правильно ввів свій пароль. Згідно з цим наприклад, веб-додаток надасть доступ до області адміністрування користувачеві, який не знає правильного пароля. Веб-сторінка входу сканується шляхом надання URL-адреси для кожного наступного інструменту сканера. Результати тестування Netsparker обчислені і показані на рис. 2.20, результати випробувань Skipfish наведені на рис. 2.21, результати тестування Vega обчислені і показані на рисунку 2.22.

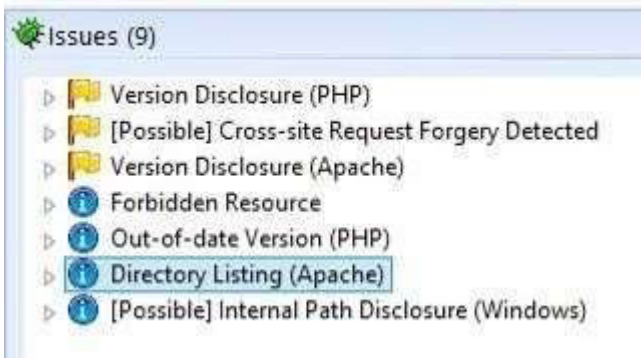


Рисунок 2.20 – Тестування MongoDB через Netsparker

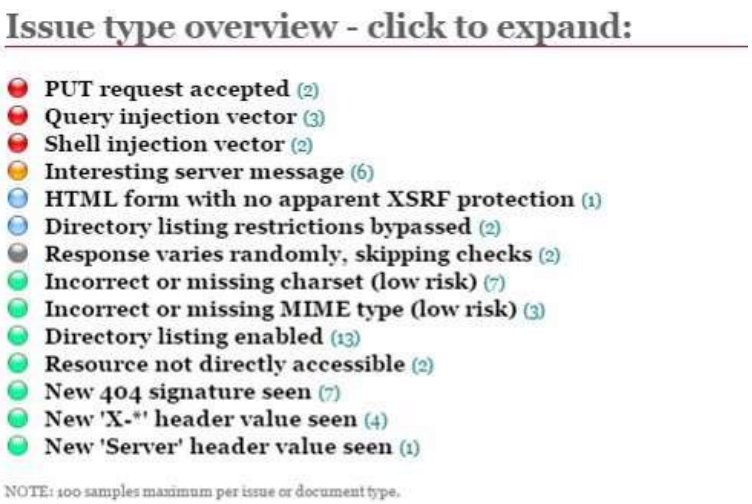


Рисунок 2.21 – Тестування MongoDB через Skipfish

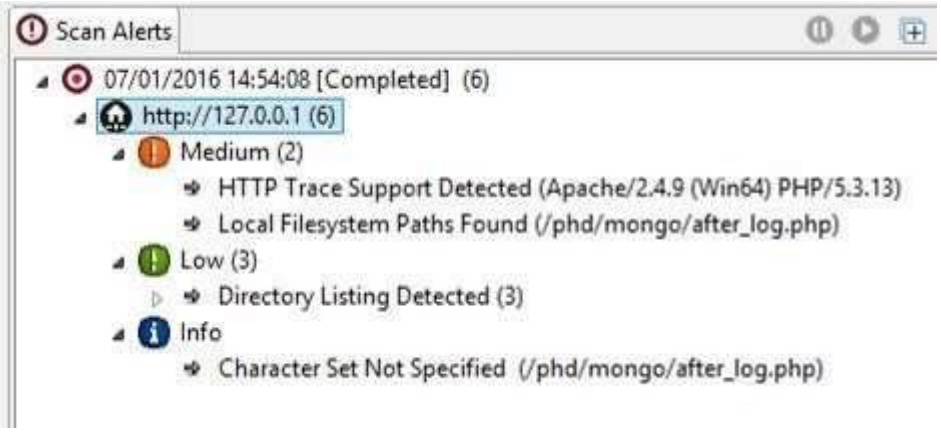


Рисунок 2.22 – Тестування MongoDB через Vega

Результати тестування «NoSQL Racket» представлені на рисунку 2.23.

Рисунок 2.23 – Результати тестування «NoSQL Racket»

Веб-сторінка входу сканується, вводячи URL-адреси по Netsparker, Vega і Skipfish, жодна з них не виявляє жодних проблем, пов'язаних з NoSQL Injection. Але коли використовувався «NoSQL Racket», атака з використанням NoSQL-ін'єкції була виявлена, і результати тестування були обчислені і показані на рис. 9.

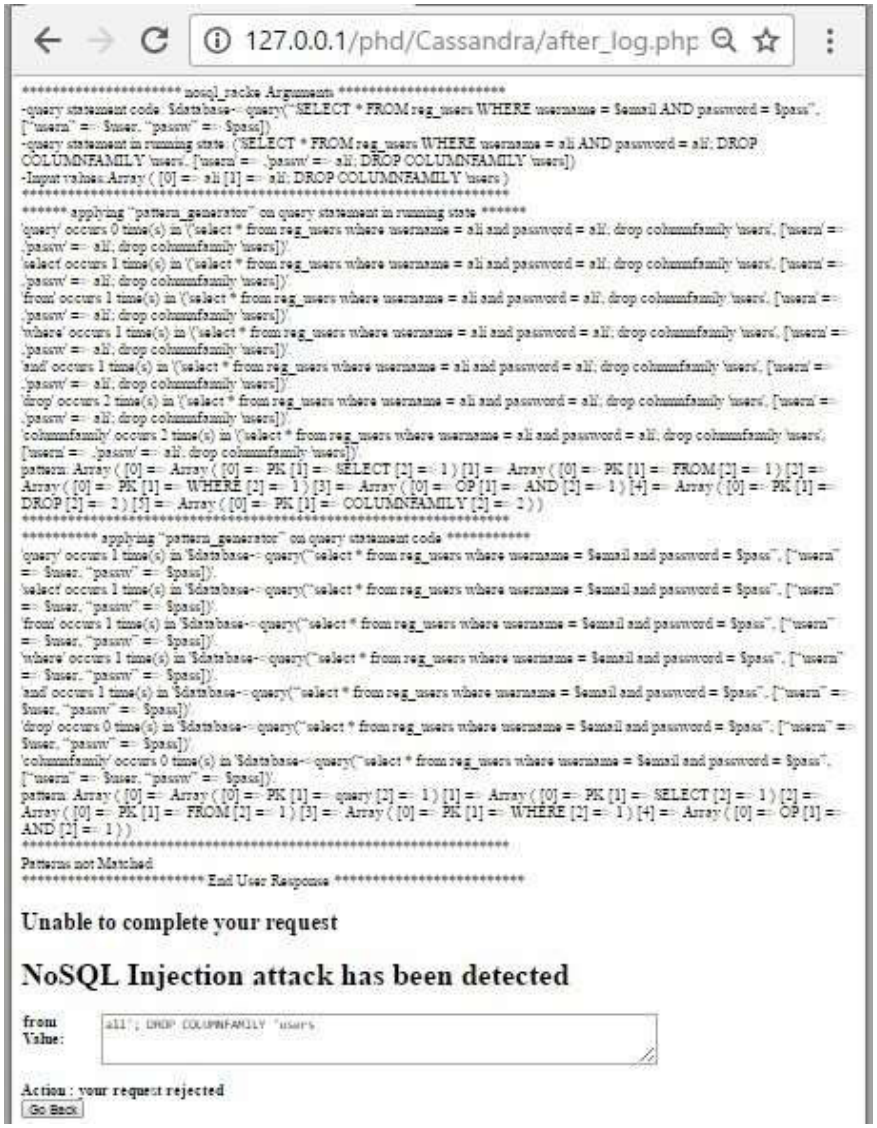


Рисунок 2.24 – Результати тестування «NoSQL Racket» для Cassandra

2) Результати виявлення Кассандри. У Кассандру зловмисник може ввести будь-яке ім'я користувача та пароль: ali '; DROP COLUMNFAMILY ' Це призводить до CQL-замовлення:

('Select * from reg_users where username = ali and password = ali'; drop columnfamily 'users', ['usern' =>, 'passw' => Ali '; drop columnfamily ' users])

Веб-сторінка входу сканується, вводячи URL-адреси по Netsparker, Vega і Skipfish, і жодна з них не виявляє жодних проблем, пов'язаних з NoSQL Injection. Але коли використовувався «NoSQL Racket», атака з використанням NoSQL-ін'єкції була виявлена, і результати тестування були обчислені і показані на рисунку 2.24.

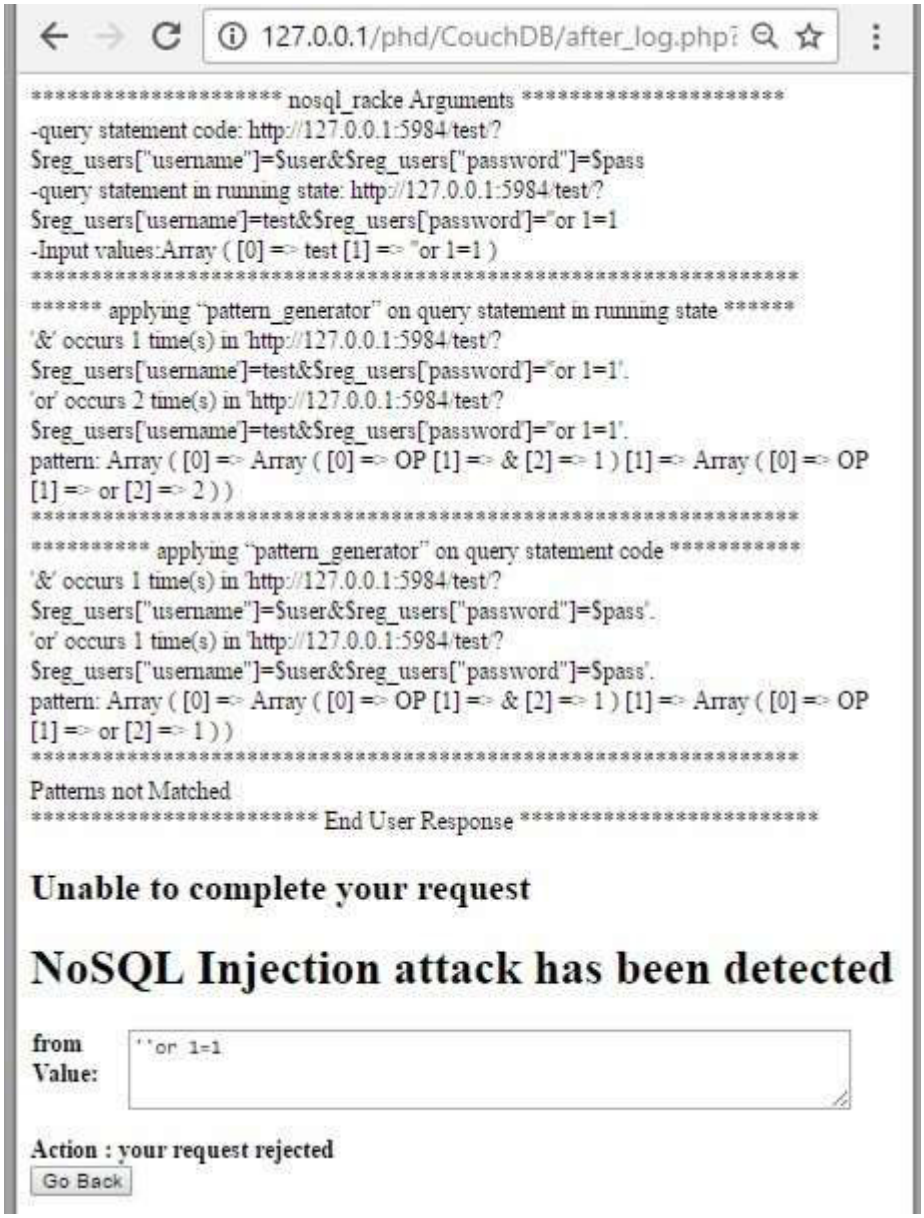


Рисунок 2.25 – Результати тестування «NoSQL Racket» для CouchDB

3) Результати виявлення CouchDB. У CouchDB зловмисник може ввести будь-яке ім'я користувача та пароль. Це призводить до URL-запиту: http://127.0.0.1/phd/CouchDB/after_log.php?user=test&pacc =% 27% 27or + 1% 3D1 & sbumit1 = Submit

Веб-сторінка входу сканується, вводячи URL-адреси по Netsparker, Vega і Skipfish, і жодна з них не виявляє жодних проблем, пов'язаних з NoSQL Injection. Але коли використовується «NoSQL Racket», виявлена атака з використанням NoSQL і результати тестування показані на рисунку 2.25.

4) Результати визначення Amazon DynamoDB. В Amazon DynamoDB можна вводити ключові слова NoSQL в представлені дані з веб-сторінки входу в систему. Наприклад, це може виглядати так: http://127.0.0.1/phd/AmazonDynamoDB/after_log.php?user = Ahmed & пройти [$ GT] = 1 & sbumit1 = Submit

«$ Gt» вибирає ті документи, ключі та, де значення поля більше (тобто>) вказаного значення. Таким чином, наведене вище твердження порівнює пароль в базі даних з нового рядка, яка повертає true. Згідно з цим наприклад, веб-додаток надасть доступ до області адміністрування користувачеві, який не знає правильного пароля. Веб-сторінка входу сканується, вводячи URL-адреси по Netsparker, Vega і Skipfish, і жодна з них не виявляє жодних проблем, пов'язаних з NoSQL Injection. Але коли використовується «NoSQL Racket», виявлена атака з використанням NoSQL і результати тестування показані на рисунку 2.26.

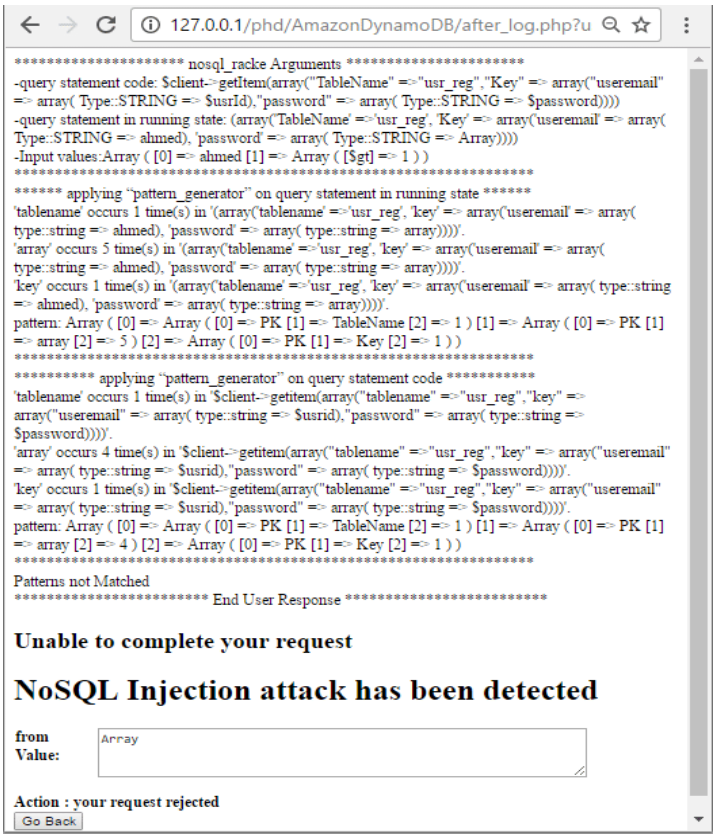


Рисунок 2.26 – Результати тестування «NoSQL Racket» для Amazon DynamoDB

Наступна таблиця 2 показує порівняння виявлення можливості для всіх інструментів тестування з запропонованим інструментом «NoSQL Racket» над чотирма базами даних NoSQL, які використовуються в процесі тестування.

Таблиця Порівняння здібностей по виявленню атак випробовуваних інструментів

Таблиця 2.6 - Порівняння виявлення можливості для всіх інструментів тестування

Тестируемые инструменты / NoSQL БД	MongoDB	Cassandra	CouchDB	Amazon DynamoDB
Netsparker	-	-	-	-
Vega	-	-	-	-
Skipfish	-	-	-	-
NoSQL Racket	+	+	+	+

Сканери додатків, такі як Netsparker, Vega і Skipfish не здатні виявити атаки NoSQL- ін'єкції. Однак, запропонований реалізований підхід виявляє атаку NoSQL Injection у всіх тестованих БД.

В. Тестування продуктивності для «NoSQL Racket»

Тестування продуктивності виконується використовуючи інструмент тестування LoadComplete. Процес тестування проходить шляхом збільшення одночасної кількості користувачів кожну секунду. У цій роботі тест, в першу чергу, буде проводиться для одного користувача. Після чого, число одночасних користувачів збільшується на 50 користувачів кожну секунду. Тестування середовище складається з машини під керуванням Windows 8.1-64 з процесором Intel Core i7 і 8 ГБ оперативної пам'яті. Результати тестування показані на наступних графіках. Графік навантаження: графік, показаний на рисунку 2.27, показує зв'язок між кількістю одночасно працюючих

користувачів і тестом часу виконання. Як показано на графіку, спостерігається, що пропонований інструмент «NoSQL Racket» може завантажувати 50 змодельованих одночасно користувачів кожну секунду.

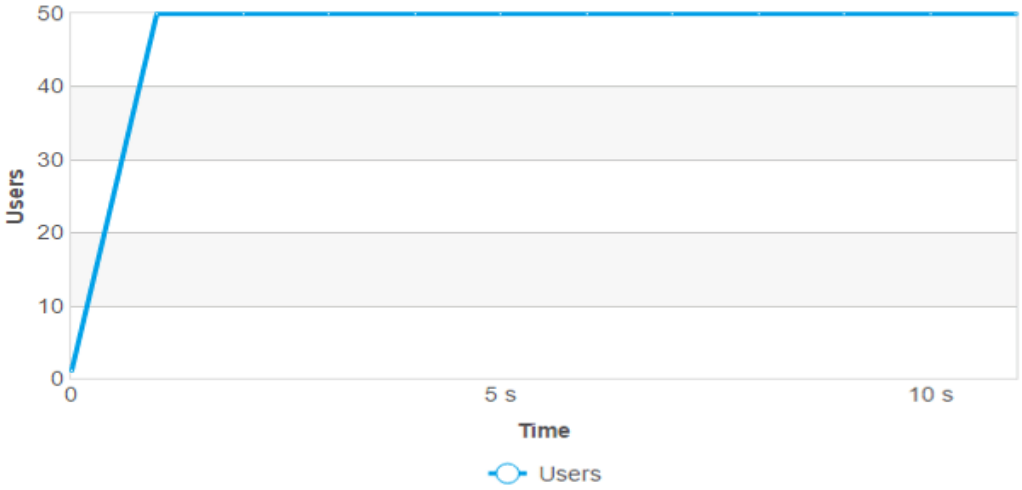


Рисунок 2.27 – Графік навантаження

Графік виконаних запитів: графік, показаний на рисунку 2.28 показує співвідношення між кількістю одночасно працюючих користувачів, кількість успішно пройдених запитів і виконання тесту час. Як показано на графіку, спостерігається, що пропонований інструмент «NoSQL Racket» може успішно передати 47 запитів від 50 запитів кожну секунду.

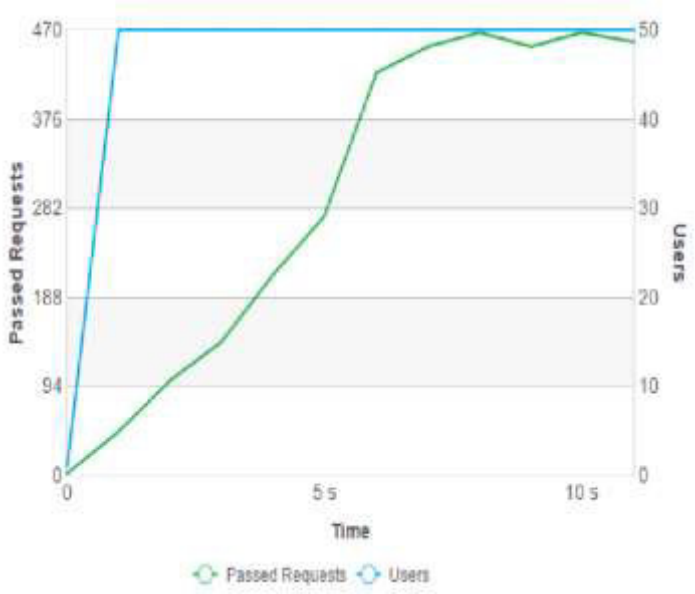


Рисунок 2.28 – Графік минулих запитів

Графік попереджень і помилок: 15 графік показує співвідношення між числом одночасних користувачів, кількістю веб-сторінок, змодельованих з попередженнями і помилками і часом виконання тесту. Як показано на графіку, попередження не виявлено, помилки виявлені.

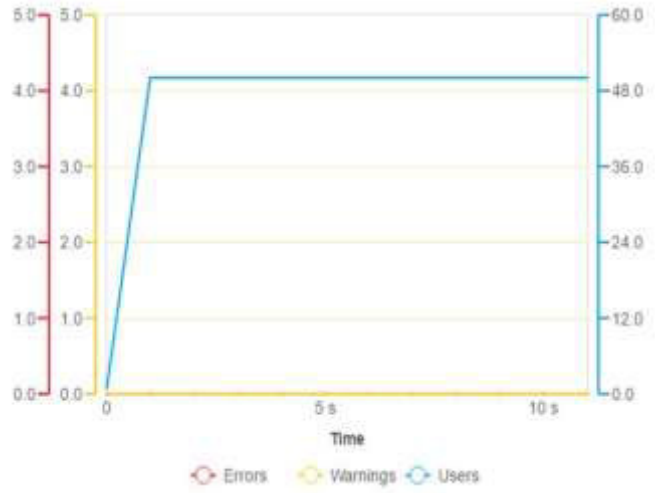


Рисунок 2.29 – Графік попереджень і помилок

Графік завантаження сторінки: час завантаження сторінки - це час за який обробляється вміст веб-сторінки, включаючи всі HTML-теги, зображення, скрипти, CSS-файли і так далі. Графік показаний на рисунку 2.30 показує співвідношення між завантаженням сторінку, часом і кількістю одночасних користувачів. Як показано в графіку, максимальний час завантаження сторінки становить 850 мс, а середній час завантаження сторінки становить 75 мс.

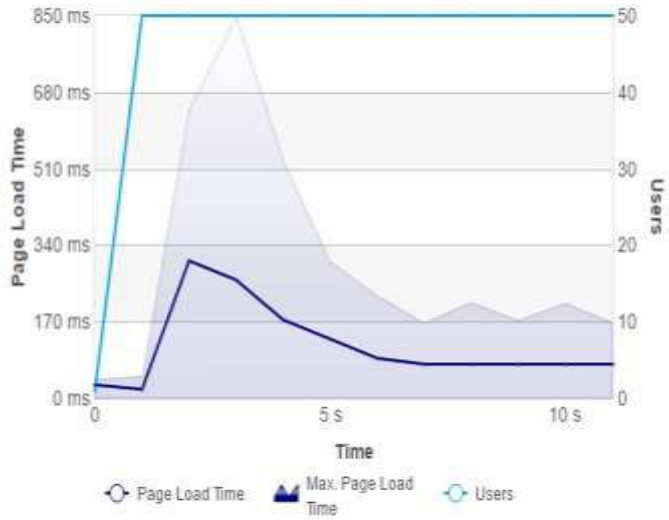


Рисунок 2.30 – Графік завантаження сторінки

Графік швидкості передачі запиту: запит передачі швидкості відноситься до швидкості передачі даних, коли запит був відправлений на сервер. Графік, показаний на рисунку 2.31, показує співвідношення між кількість одночасно працюючих користувачів, запитом швидкості передачі метрики і часом виконання тесту. Як показано на графіку, найповільніша швидкість передачі запитів - 200 кБ / с.

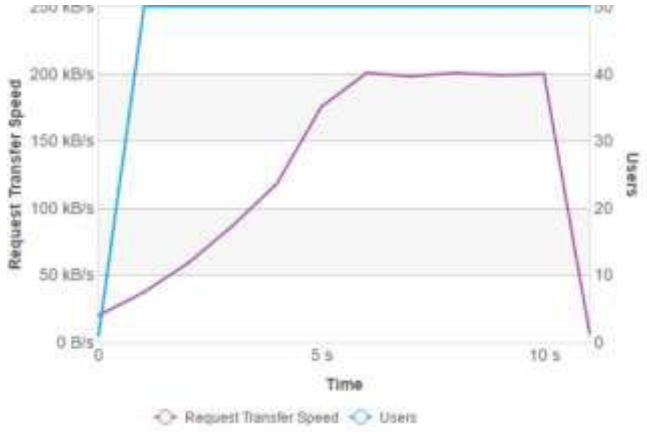


Рисунок 2.31 – Графік швидкості передачі запиту

Графік швидкості передачі відповіді: відповідь на швидкості відноситься до швидкості передачі даних, коли сервер відправив відповідь назад. Графік, показаний на рисунку 2.32, показує співвідношення між числом одночасних користувачів, показником швидкості передачі відповіді і часом виконання тесту. Як показано на графіку, найповільніша швидкість передачі відповіді - 1,52 МБ / с.

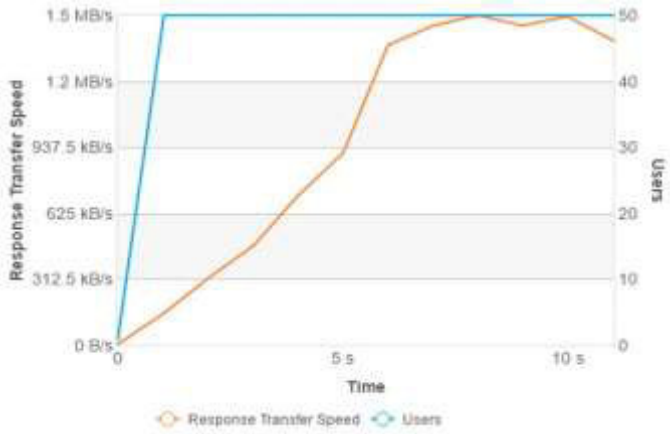


Рисунок 2.32 – Графік швидкості передачі відповіді

Висновки до другого розділу

У другому розділі була розглянута сучасна СУБД MongoDB. Були наочно розглянуті процеси реалізації атак-ін'єкцій в регулярних виразах, JSON і JavaScript-ін'екціі. Були запропоновані основні рекомендації по налаштуванню конфігурації системи і запропонована методика по захисту від розглянутих атак. Для впровадження якої було запропоновано попереднє тестування програми на предмет ін'єкцій, яка називається «NoSQL Racket», цей інструмент реалізований функцією PHP. Якщо NoSQL виявляє ін'єкційні атаки, він продовжить працювати; в разі збою однієї або декількох атак з використанням NoSQL, він відобразить сторінку помилки і припинить роботу для NOSQL запиту. Випробувальний інструмент «NoSQL Racket» був застосован на чотирьох різних базах даних NoSQL - MongoDB, Cassandra, CouchDB і Amazon DynamoDB. Крім того, його здатність до виявлення порівнянна з найпотужнішими інструментами тестування веб-додатків Netsparker, Vega і Skipfish. За даними сканування результатів, жоден з розглянутих інструментів не зміг виявити NoSQL.

3 РОЗДІЛ. ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок (фіксованих) капітальних витрат

Капітальні інвестиції:

- вартість розробки проекту інформаційної безпеки (розробка схем пристроїв, політики функціонування системи тощо);
- вартість створення основного й додаткового програмного забезпечення
(ПЗ);
- витрати на первісні закупівлі апаратного забезпечення;
- витрати на навчання технічних фахівців і обслуговуючого персоналу.

Спершу розрахуємо час, який буде витрачено на створення ПЗ:

$$t = tmз + tв + ta + tпр + tопр + tд, \text{ годин,} \quad (3.1)$$

де $tmз$ – тривалість складання технічного завдання на розробку ПЗ;

$tв$ – тривалість вивчення ТЗ, літературних джерел за темою тощо;

ta – тривалість розробки блок-схеми алгоритму;

$tпр$ – тривалість програмування за готовою блок-схемою;

$tопр$ – тривалість опрацювання програми на ПК;

$tд$ – тривалість підготовки технічної документації на ПЗ.

Умовна кількість оперантів у програмі:

$$Q = q \cdot c\,(1 + p), \text{ штук,} \quad (3.2)$$

де q – очікувана кількість операторів – 30;

с – коефіцієнт складності програми – 1.3;

р – коефіцієнт корекції програми в процесі її опрацювання – 0.07.

$$Q = 30 \cdot 1.3(1 + 0.07) = 28.6, \text{ штук.}$$

Оцінка тривалості складання технічного завдання на розробку ПЗ **tтз** – 3 год. Тривалість вивчення технічного завдання:

$$t_в = \frac{Q \cdot B}{(75...85) \cdot k} = \frac{28.6 \cdot 1.2}{80 \cdot 1} = 0.429 \text{, годин.} \quad (3.3)$$

де B – коефіцієнт збільшення тривалості етапу внаслідок недостатнього опису завдання, B = 1,2...1,5; k – коефіцієнт, що враховує кваліфікацію програміста і визначається стажем роботи за фахом: від 2 до 3 років – 1,0;

Тривалість розробки блок-схеми алгоритму:

$$ta = \frac{Q}{(20...25) \cdot k} = \frac{28.6}{20 \cdot 1} = 1.43 \text{, годин.} \quad (3.4)$$

Тривалість складання програми за готовою блок-схемою:

$$tпр = \frac{Q}{(20...25) \cdot k} = \frac{28.6}{20 \cdot 1} = 1.43 \text{, годин.} \quad (3.5)$$

Тривалість опрацювання програми на ПК:

$$tonp = \frac{1{,}5Q}{(4...5)\cdot k} = \frac{1.5\cdot 28.6}{4\cdot 1} = 10.725$$, годин. (3.6)

Тривалість підготовки технічної документації на ПЗ:

$$t_д = \frac{Q}{(15...20)\cdot k} + \frac{Q}{(15...20)}\cdot 0{,}75 = \frac{28.6}{15\cdot 1} + \frac{28.6}{15}\cdot 0.75 = 2.87$$, годин. (3.7)

t = 3 + 0.429 + 1.43 + 1.43 + 10.725 + 2.87=19.884 годин.

Розрахунок витрат на створення програмного продукту

$$К_{пз} = З_{зп} + З_{мч}. \text{ грн}$$ (3.8)

Заробітна плата виконавця враховує основну і додаткову заробітну плату, а також відрахування на соціальне потреби (пенсійне страхування, страхування на випадок безробіття, соціальне страхування тощо) и визначається за формулою:

$$З_{зп} = t\cdot З_{пр} = 19.884\cdot 22.02 = 437.85$$, грн, (3.9)

де t – загальна тривалість створення ПЗ, годин;

Зпр – средньогодинна заробітна плата програміста з нарахуваннями, грн/годину.

$$З_{пр} = \frac{З_м}{168} = \frac{3700}{168} = 22.02$$, грн/годину. (3.10)

де $З_м$ – середня заробітна плата на місяць – 3700 грн.

Вартість машинного часу для налагодження програми на ПК визначається за формулою:

$$З_{мч} = t_{опр} \cdot С_{мч} + t_{д} = 10.725 \cdot 1.2 + 2.87 = 15.74 \text{, грн.} \quad (3.11)$$

де $t_{опр}$ – трудомісткість налагодження програми на ПК, годин;

$t_{д}$ – трудомісткість підготовки документації на ПК, годин;

$С_{мч}$ – вартість 1 години машинного часу ПК, грн./година.

Вартість 1 години машинного часу ПК визначається за формулою:

$$С_{мч} = P \cdot t_{нал} \cdot С_{е} + \frac{Ф_{зал} \cdot Н_{а}}{F_{р}} + \frac{К_{лпз} \cdot Н_{апз}}{F_{р}}$$

$$С_{мч} = 0.5 \cdot 2.1 + \frac{3000 \cdot 0.1}{1920} = 1.2 \text{, грн/год.} \quad (3.12)$$

де P – встановлена потужність ПК, 0.5 кВт;

$С_{е}$ – тариф на електричну енергію, 2.1 грн/кВт·година;

$Ф_{зал}$– залишкова вартість ПК на поточний рік, 3000 грн.;

$Н_{а}$ – річна норма амортизації на ПК, 0.1 частки одиниці;

$Н_{апз}$ – річна норма амортизації на ліцензійне програмне забезпечення, частки одиниці;

$К_{лпз}$ – вартість ліцензійного програмного забезпечення, грн.;

$F_{р}$ – річний фонд робочого часу (за 40-годинного робочого тижня $F_{р}$ = 1920 год).

$$К_{пз} = 437.85 + 15.74 = 453.59 \text{ грн.} \quad (3.8)$$

Таким чином, капітальні (фіксовані) витрати на проектування та впровадження проектного варіанта системи інформаційної безпеки складають:

$$К = К_{пз} + К_{аз} + К_{навч} + К_{н}, \text{ тис. грн.} \quad (3.13)$$

де $К_{пз}$ – вартість створення програмного продукту, тис. грн;

$К_{аз}$ – вартість закупівлі апаратного забезпечення та допоміжних матеріалів, тис. грн;

$К_{навч}$ – витрати на навчання технічних фахівців і обслуговуючого персоналу, тис. грн;

$К_{н}$ – витрати на встановлення обладнання та налагодження системи інформаційної безпеки, тис. грн.

Таблиця 3.1 Вартість закупівлі апаратного забезпечення та допоміжних матеріалів

Назва комплектуючих	Вартість, грн.
Процесор: Процессор AMD Athlon X4 870K 3.9GHz/4MB	1269
Материнская плата: ASRock H81M-VG4 R3.0	1259
ОЗУ для Intel: AMD SODIMM DDR3-1600 4096MB PC3-12800	705
Жесткий диск: i.norys 500GB 5900prm 8MB INO-IHDD0500S2-D1-5908 3.5 SATAII	745
Корпус з блоком живлення: Gamemax ET-209-NP	418
Разом	4396

$К_{аз}$ = 4.396 тис. грн.

Витрати на навчання технічних фахівців і обслуговуючого персоналу, це є підготовчі курси з адміністрування та обслуговування системи виявлення вторгнень що складають 1.5 тис. грн;

$К_{навч}$ =1.5 тис. грн.

Витрати на встановлення обладнання та налагодження системи інформаційної безпеки складають, 0.5 тис. грн.

$К_н$ = 0.5 тис. грн.

$$К = 0.45359 + 4.396 + 1.5 + 0.5 = 6.849 \text{ тис. грн.} \quad (3.13)$$

3.2 Експлуатаційні витрати:

$$С_к = С_н + С_а + С_з + С_{єв} + С_е + С_{ел} + С_{тос} \quad (3.14)$$

де витрати на навчання адміністративного персоналу й кінцевих користувачів($С_н$). визначаються за даними організації з проведення тренінгів персоналу, курсів підвищення кваліфікації – 1.5 тис. грн.

Річний фонд амортизаційних відрахувань ($С_а$) визначається у відсотках від суми капітальних інвестицій за видами основних фондів і нематеріальних активів (ПЗ) – 20% або 1317 грн.

Річний фонд заробітної плати інженерно-технічного персоналу, що обслуговує систему інформаційної безпеки ($С_з$), складає:

$$С_з = З_{осн} + З_{дод} = 3723 \cdot 12 + 3723 \cdot 0.22 \cdot 12 = 54\,504{,}72 \text{ грн.} \quad (3.15)$$

де $З_{осн}$, $З_{дод}$ – основна мінімальна заробітна плата на 01.01.2018 [17], грн на рік.

Единий соціальний внесок – 0.22, частки одиниці;

Вартість електроенергії, що споживається апаратурою системою інформаційної безпеки протягом року ($С_е$), визначається за формулою:

$$С_{ел} = Р \cdot F_р \cdot Ц_е = 0.5 \cdot 365 \cdot 24 \cdot 2.1 = 9\ 198 \text{ грн}, \quad (3.16)$$

де Р – встановлена потужність апаратури інформаційної безпеки, кВт;

$F_р$ – річний фонд робочого часу системи інформаційної безпеки (визначається виходячи з режиму роботи системи інформаційної безпеки);

$Ц_е$ – тариф на електроенергію, грн/кВт·годин

Витрати на технічне й організаційне адміністрування та сервіс системи виявлення вторгнень визначаються у відсотках від вартості капітальних витрат 2%. А саме:

$$С_{тос} = К \cdot 0.2 = 1\ 369 \text{ грн}$$

$$Ск = 1 + 1.317 + 54.504 + 9.198 + 1{,}369 = 67{,}388 \text{ , тис. грн.} \quad (3.14)$$

3.3 Оцінка можливого збитку від атаки (злому) на вузол або сегмент корпоративної мережі

Кінцевим результатом впровадження й проведення заходів щодо забезпечення інформаційної безпеки є величина відвернених втрат, що розраховується, виходячи з імовірності виникнення інциденту інформаційної безпеки й можливих економічних втрат від нього. По суті, ця величина відображає ту частину прибутку, що могла бути втрачена.

Загалом можливо виділити такі види збитку, що можуть вплинути на ефективність комп'ютерної системи інформаційної безпеки (КСІБ):

- порушення конфіденційності ресурсів КСІБ (тобто неможливість доступу до них неавторизованих суб'єктів або несанкціонованого використання каналів зв'язку);
- порушення доступності ресурсів КСІБ (тобто можливість доступу до них авторизованих суб'єктів (завжди, коли їм це потрібно);
- порушення цілісності ресурсів КСІБ (тобто їхня неушкодженість);
- порушення автентичності ресурсів КСІБ (тобто їхньої дійсності, непідробленості).

Вихідні дані:

$t_п$ =48 годин – час простою вузла або сегмента корпоративної мережі внаслідок атаки, годин;

$t_в$=24 годин – час відновлення після атаки персоналом, що обслуговує корпоративну мережу, годин;

$t_{ви}$=12 годин – час повторного введення загубленої інформації співробітниками атакованого вузла або сегмента корпоративної мережі, годин;

$З_о$=3723 грн – місячна заробітна плата обслуговуючого персоналу (адміністраторів та ін.) з нарахуванням єдиного соціального внеску, грн на місяць;

$З_с$=4300 грн – місячна заробітна плата співробітника атакованого вузла або сегмента корпоративної мережі з нарахуванням єдиного соціального внеску, грн на місяць;

$Ч_о$=4 – чисельність обслуговуючого персоналу (адміністраторів та ін.), осіб.;

$Ч_с$=10 – чисельність співробітників атакованого вузла або сегмента корпоративної мережі, осіб.;

О = 150 000 грн – обсяг чистого прибутку/дохід від реалізації/ атакованого вузла або сегмента корпоративної мережі, грн у рік, або

оподаткований прибуток атакованого вузла або сегмента корпоративної мережі;

$П_{зч}$ = 1700 грн – вартість заміни встаткування або запасних частин, грн;

I=1 – число атакованих вузлів або сегментів корпоративної мережі;

N = 16 – середнє число можливих атак на рік.

Упущена вигода від простою атакованого вузла або сегмента корпоративної мережі становить:

$$U = П_п + П_в + V, \text{ грн.} \quad (3.15)$$

де $П_п$ – оплачувані втрати робочого часу та простої співробітників атакованого вузла або сегмента корпоративної мережі, грн;

$П_в$ – вартість відновлення працездатності вузла або сегмента корпоративної

мережі (переустановлення системи, зміна конфігурації та ін.), грн;

U – втрати від зниження обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі, грн.

Втрати від зниження продуктивності 3 співробітників з ЗП атакованого вузла або сегмента корпоративної мережі являють собою втрати їхньої заробітної плати (оплата непродуктивної праці) за 60 годин простою внаслідок атаки:

$$П_п = \frac{\sum Зс \cdot Чс}{F} \cdot t_п, \quad (3.16)$$

де F – місячний фонд робочого часу (при 40-а годинному робочому тижні становить 160-176 ч).

$$П_п = \frac{\sum 4300 \cdot 10}{160} \cdot 48 = 12900 \text{, грн.}$$

Витрати на відновлення працездатності вузла або сегмента корпоративної мережі включають кілька складових:

$$П_в = П_{ви} + П_{пв} + П_{зч}, \text{ грн.} \quad (3.17)$$

де $П_{ви}$ – витрати на повторне уведення інформації, грн;

$П_{пв}$ – витрати на відновлення вузла або сегмента корпоративної мережі, грн;

$П_{зч}$ – вартість заміни устаткування або запасних частин, грн.

Витрати на повторне введення інформації $П_{ви}$ розраховуються виходячи з розміру заробітної плати 4300 грн 3 співробітників атакованого вузла або сегмента корпоративної мережі $З_с$, які зайняті повторним введенням втраченої інформації, з урахуванням необхідного для цього часу $t_{ви}$=12:

$$П_{ви} = \frac{\sum 4300}{160} \cdot 12 = 322.5 \text{, грн.} \quad (3.18)$$

Витрати на відновлення вузла або сегмента корпоративної мережі $П_{пв}$ визначаються часом відновлення після атаки $t_в$ = 24 і розміром середньогодинної заробітної плати обслуговуючого персоналу (адміністраторів):

$$П_{пв} = \frac{\sum 3723}{160} \cdot 24 = 558.5 \text{, грн.} \quad (3.19)$$

$$П_в = 322.5 + 558.5 + 1760 = 2641 \text{ грн.} \quad (3.17)$$

Втрати від зниження очікуваного обсягу продаж в 200 000 грн за 90 годин простою атакованого вузла або сегмента корпоративної мережі виходячи із середньогодинного обсягу продажів і сумарного часу простою атакованого вузла або сегмента корпоративної мережі:

$$V = \frac{O}{Fг} \cdot (t_п + t_в + t_{ви}) = \frac{150000}{9340} \cdot (48 + 24 + 12) = 1349.04 \text{, грн} \quad (3.20)$$

де $F_г$ – річний фонд часу роботи організації (прийом заказів інтернетмагазином) становить близько 9340 ч.

$$U = П_п + П_в + V = 12900 + 2641 + 1349.04 = 16890.04 \text{ грн.} \quad (3.15)$$

Таким чином, загальний збиток від атаки на вузол або сегмент корпоративної мережі організації складе

$$B = \sum_i \sum_n U = 16890.04 \cdot 16 \cdot 1 = 270240.64 \text{ грн.} \quad (3.21)$$

3.4 Загальний ефект від впровадження системи інформаційної безпеки

Загальний ефект від впровадження системи інформаційної безпеки визначається з урахуванням ризиків порушення інформаційної безпеки і становить:

$$E = B \cdot R – C = 270240.64 \cdot 0.7 - 67.388 = 121780.448 \text{ грн,} \quad (3.22)$$

де В – загальний збиток від атаки на вузол або сегмент корпоративної мережі, грн;

R – очікувана імовірність атаки на вузол або сегмент корпоративної мережі, частки одиниці;

C – щорічні витрати на експлуатацію системи інформаційної безпеки, тис. грн.

3.4 Визначення та аналіз показників економічної ефективності системи інформаційної безпеки

Коефіцієнт повернення інвестицій ROSI:

$$ROSI = \frac{E}{K} = \frac{121.78}{6.85} = 17.78 \text{, частки одиниці,} \quad (3.23)$$

де E – загальний ефект від впровадження системи інформаційної безпеки, грн;

K – капітальні інвестиції за варіантами, що забезпечили цей ефект, грн. Термін окупності:

$$T_O = \frac{K}{E} = \frac{1}{ROSI} = 0.05 \text{ років.} \quad (3.24)$$

Висновки до третього розділу

В економічному розділі в результаті обчислюваних витрат на впровадження методів захисту бази даних MongoDB, було підтверджено його економічну ефективність та термін окупності витрат. Проект економічно доцільний та його можно використовувати на підприємстві.

ВИСНОВКИ

Бази даних NoSQL схильні до тих же ризиків безпеки, що і їх аналоги SQL. Деякі низькорівневі методи і протоколи змінилися, але ризики впровадження, неналежного управління доступом і небезпечного доступу до мережі високі і схожі між системами SQL і NoSQL. В області NoSQL - рішень СУБД MongoDB є хорошим рішенням з вбудованими заходами безпеки. Однак навіть використання найбільш безпечного сховища даних не запобігає ін'єкційні атаки, які використовують уразливості в веб-додатках, які звертаються до сховища даних. Один із способів запобігти цьому - ретельний аналіз коду і статичний аналіз.

В ході роботи були запропоновані способи захисту від атак і пом'якшення їх наслідків для забезпечення безпеки використання. А також був модифікован і розглянут інструмент тестування для виявлення атак NoSQL Injection, який називається NoSQL Racket - інструмент, реалізований як функція PHP. Його здатність до виявлення і профілактики порывняна з найпотужнішими інструментами тестування веб-додатків, такими як Netsparker, Vega і Skipfish.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mongodb - Security Weaknesses in a typical NoSQL database. [Электронный ресурс]/ trustwave.com – Режим доступа: https://www.trustwave.com/Resources/SpiderLabs-Blog/Mongodb---SecurityWeaknesses-in-a-typical-NoSQLdatabase/

2. Гузь Э. Ю. Азбука NoSQL-инъекций [Электронный ресурс]/ habr.com – Режим доступа: https://habr.com/company/xakep/blog/143909/

3. Купцов С. М. Разбираем ACID по буквам в NoSQL [Электронный ресурс]/ habr.com – Режим доступа: https://habr.com/post/228327/

4. Результаты опроса разработчиков 2018 [Электронный ресурс]/ insights.stackoverflow.com – Режим доступа: https://insights.stackoverflow.com/survey/2018/

5. CAP-ТЕОРЕМА БРЮЕРА [Электронный ресурс]/ softwaremaniacs.org – Режим доступа: http://softwaremaniacs.org/blog/2010/01/31/brewers-cap-theorem/

6. NoSQL vs SQL, и при чем тут теорема CAP [Электронный ресурс]/ dou.ua – Режим доступа: https://dou.ua/lenta/articles/nosql-vs-sql/

7. Козырев Ю. А. PHP и различные виды NoSQL [Электронный ресурс]/ habr.com – Режим доступа: https://habr.com/post/214647/

8. Lior Okman, Nurit Gal-Oz, Yaron Gonen, Ehud Gudes & Jenny Abramov. “Security Issues in NoSql Databases”, IEEE, 2011.

9. Jens Heyens, Kai Greshake, & Eric Petryka, “MongoDB databases at risk-Several Thousand Mongo DBs without access control on the internet”, 2015.

10. Elisa Bertino & Ravi Sandhu. “Database Security- Concepts, Approaches and Challenges”, IEEE, 2005.

11. MongoDB overview [Электронный ресурс]/ mongodb.com – Режим доступа: http://www.mongodb.com/mongodb.overview

12. Kaur, H., Kaur, J., Kaur, K.: A review on non-relational databases, their types, advantages and disadvantages. IJERT (February 2013)

13. S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services,” SIGACT News, vol. 33, pp. 51–59, June 2002 [Электронный ресурс]/ doi.acm.org – Режим доступа: http://doi.acm.org/10.1145/564585.564601

14. Jeremy Ronk, "Introduction to NoSQL" [Электронный ресурс]/ jeremyronk.wordpress.com – Режим доступа: http://jeremyronk.wordpress.com/2014/06/23/introduction-to-no-sgl.

15. Kristina Chodorow. "scaling MongoDB". O' Reilly Media, January 2011. pI3.

16. BSON [Электронный ресурс]/ bsonspec.org – Режим доступа: http://bsonspec.org/, 2014.11.

17. Минимальная заработная плата Статистика [Электронный ресурс]/ index.minfin.com.ua – Режим доступа: https://index.minfin.com.ua/labour/salary/min

18. СУБД NoSQL – сильные и слабые стороны [Электронный ресурс]/ jetinfo.ru – Режим доступа: http://www.jetinfo.ru/stati/silnye-i-slabye-storony-nosql

19. Швечиков Д. С. MongoDB: Запросы [Электронный ресурс]/ habr.com – Режим доступа: https://habr.com/post/134590/

20. Основы языка SQL [Электронный ресурс]/ opennet.ru – Режим доступа: https://www.opennet.ru/docs/RUS/sql/

21. DB-Engines Ranking [Электронный ресурс]/ db-engines.com – Режим доступа: https://db-engines.com/en/ranking

ДОДАТОК А. Відомість матеріалів дипломного проекту

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	3	
2	А4	Зміст	2	
3	А4	Вступ	2	
4	А4	1 Розділ	21	
5	А4	2 Розділ	39	
6	А4	3 Розділ	12	
7	А4	Висновки	1	
8	А4	Перелік посилань	2	
9	А4	Додаток А	1	
10	А4	Додаток Б	1	
11	А4	Додаток В	1	
12	А4	Додаток Г	1	

ДОДАТОК Б. Перелік файлів на електронному носії

1. Пояснювальна Записка.docx
2. Презентація_Диплом.pttx
3. racket-master.7z

ДОДАТОК В. Відгук керівника економічного розділу

Керівник: _____ к.е.н, доц. Пілова Д. П.
(підпис)

ДОДАТОК Г. ВІДГУК

на дипломну роботу магістра на тему:
Забезпечення безпеки від атак-ін'єкцій
нереляційної бази даних MongoDB NoSQl
студентки групи 125м-17-1
Рогалевої Марина Володимирівни

Пояснювальна записка розташована на ___ сторінках та містить __ рисунків, 20 джерел та 4 додатка. Тема і зміст дипломної роботи повністю відповідає освітньо-професійній програмі 125 Кібербезпека.

Розробка методики щодо забезпечення безпеки від атак NoSQL-ін'єкцій є перспективною задачою, від вирішення якої залежить захищенность конфіденційної інформації на підприємствах. Актуальною проблемою нереляційної бази даних MongoDB є ряд реалізуємих на неї атак-ін'єкцій з метою крадіжки даних, зловмисної модифікації або знищення, а також однією з проблем виступає не освідченність адміністраторів баз даних у данному питанні, що ставить під загрозу зберігання даних. Аналіз атак і вразливостей нереляційної бази даних MongoDB створює ряд питаннь вирішення яких дає змогу підвищити безпеку зберігання даних.

Зміст та структура дипломної роботи дозволяють розкрити данне питання, що полягає у створенні методики щодо забезпечення безпеки від атак NoSQL-ін'єкцій.

Студентка показала добрий рівень володіння теоретичними положеннями з обраної теми, показав здатність формувати власну точку зору (теоретичну позицію).

Робота виконана самостійно. В дипломному проекті відображені типові атаки на нереляційні бази даних, що ставлять під загрозу безпечне зберігання конфіденційної інформації.

Робота оформлена та написана відповідно до вимог щодо написання дипломної роботи магістра. Містить необхідний ілюстрований матеріал. Автор добре знає проблему, уміє формулювати наукові та практичні завдання і знаходить адекватні засоби для їх вирішення.

В цілому дипломна робота задовольняє усім вимогам і заслуговує на оцінку _____, а її автор Рогалева Марина Володимирівна присвоєння їй звання магістра та кваліфікації професіонал із організації інформаційної безпеки.

Керівник дипломної роботи
д.т.н., професор В.І. Корнієнко
Керівник спеціальної частини
старший викладач БІТ В. О. Святошенко