**PHÂN TÍCH VÀ PHÒNG NGỪA NOSQL INJECTION: SO SÁNH CÁC CHIẾN LƯỢC BẢO MẬT GIỮA SQL VÀ NOSQL**

Nguyen Tuan Kiet 1, Nguyen Ngoc Khanh 2, Trần Thị Kim Chi 3

*1 IUH - IS,*

*2 IUH - IS,*

*3 IUH - IS*

[*kiet5454aa@gmail.com*](mailto:kiet5454aa@gmail.com)*,* [*khanhise2002@gmail.com*](mailto:khanhise2002@gmail.com)*,* [*tranthikimchi@iuh.edu.vn*](mailto:tranthikimchi@iuh.edu.vn)

**ANALYSIS AND PREVENTION OF NOSQL INJECTION: A COMPARATIVE STUDY OF SECURITY STRATEGIES BETWEEN SQL AND NOSQ**

**Tóm tắt.**

Trong bối cảnh các hệ thống cơ sở dữ liệu phát triển mạnh mẽ, an ninh cho cơ sở dữ liệu SQL và NoSQL trở thành nhu cầu thiết yếu. Tấn công SQL Injection đã được nghiên cứu rộng rãi, trong khi NoSQL Injection nổi lên như một mối đe dọa mới và phức tạp do sự linh hoạt của các hệ cơ sở dữ liệu phi quan hệ. Nghiên cứu này tập trung so sánh các chiến lược phòng chống hai loại tấn công này, bao gồm các phương pháp như xác thực đầu vào, kiểm tra truy vấn, và sử dụng tường lửa ứng dụng web (WAF). Bài báo áp dụng cách tiếp cận phân tích kỹ thuật kết hợp thử nghiệm thực tế, sử dụng các công cụ như Burp Suite và SQLmap để đánh giá hiệu quả của từng biện pháp bảo mật trên MySQL với PHP và MongoDB với NodeJS. Kết quả cho thấy việc kết hợp kiểm tra truy vấn với xác thực đầu vào có thể giảm thiểu đáng kể rủi ro từ Injection.

**Từ khóa.** NoSQL Injection, SQL Injection, bảo mật cơ sở dữ liệu, phòng chống tấn công, chiến lược bảo mật

1. **Giới thiệu**

Trong kỷ nguyên số, dữ liệu là một trong những tài sản quan trọng nhất đối với các tổ chức và doanh nghiệp. Việc quản lý và bảo vệ dữ liệu trở nên ngày càng phức tạp, đặc biệt với sự phát triển của các hệ thống cơ sở dữ liệu lớn và đa dạng, từ các hệ thống cơ sở dữ liệu quan hệ (SQL) truyền thống đến các hệ thống cơ sở dữ liệu phi quan hệ (NoSQL). Sự chuyển dịch này mang lại nhiều lợi ích trong việc lưu trữ và truy xuất dữ liệu phức tạp và linh hoạt, nhưng cũng đồng thời mở ra các lỗ hổng bảo mật mới. Một trong những hình thức tấn công bảo mật đáng chú ý nhất liên quan đến dữ liệu là SQL Injection, một kỹ thuật mà tin tặc sử dụng để chèn mã độc vào các truy vấn SQL nhằm truy cập hoặc thay đổi dữ liệu mà không được phép.

Mặc dù đã có những tiến bộ đáng kể trong việc phòng ngừa SQL Injection, sự xuất hiện của cơ sở dữ liệu NoSQL lại đưa ra một thách thức mới: NoSQL Injection. Khác với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS), các hệ cơ sở dữ liệu NoSQL như MongoDB, Couchbase, và Cassandra không sử dụng ngôn ngữ truy vấn chuẩn SQL mà thay vào đó là các cấu trúc dữ liệu và API truy vấn độc lập phụ thuộc nhiều vào ngôn ngữ lập trình của website. Sự khác biệt này tạo ra một số điểm mạnh nhưng cũng làm tăng tính phức tạp trong việc bảo vệ khỏi các cuộc tấn công injection, đặc biệt là khi các công cụ phòng ngừa truyền thống không còn phát huy hiệu quả. Do vậy, nghiên cứu về cách phòng chống NoSQL Injection là vô cùng cần thiết nhằm bảo vệ dữ liệu trong các hệ thống NoSQL.

Trong bài nghiên cứu này, nhóm chúng tôi hướng đến mục tiêu so sánh các chiến lược phòng ngừa đối với hai loại tấn công injection này trong các hệ quản trị cơ sở dữ liệu SQL và NoSQL. Phần lớn các chiến lược hiện nay tập trung vào việc sử dụng các phương pháp như xác thực, phân quyền, kiểm tra tính hợp lệ của đầu vào, và quản lý quyền truy cập. Tuy nhiên, các biện pháp này không phải lúc nào cũng có thể áp dụng đồng nhất cho cả SQL và NoSQL do sự khác biệt trong kiến trúc và cách thức truy vấn. Chính vì vậy, nghiên cứu này sẽ tập trung vào việc phân tích từng phương pháp phòng ngừa cho cả hai loại cơ sở dữ liệu, so sánh hiệu quả, ưu và nhược điểm của từng phương pháp, từ đó đề xuất những giải pháp tối ưu cho các hệ thống NoSQL.

Để đạt được mục tiêu trên, nghiên cứu sẽ bao gồm các nội dung sau: (1) tổng quan về cơ sở dữ liệu SQL và NoSQL cũng như các dạng tấn công injection phổ biến trong từng hệ thống; (2) phân tích kỹ lưỡng các chiến lược phòng ngừa SQL Injection đã được áp dụng hiệu quả và thách thức khi triển khai các chiến lược này trên NoSQL; (3) đánh giá và so sánh các phương pháp phòng ngừa NoSQL Injection mới; và (4) đề xuất các phương pháp phòng ngừa tối ưu cho các hệ thống NoSQL. Qua nghiên cứu này, chúng tôi mong muốn đóng góp vào lĩnh vực bảo mật hệ thống thông tin bằng cách cung cấp những kiến thức và đề xuất thiết thực giúp giảm thiểu rủi ro do các cuộc tấn công injection gây ra, đặc biệt trong bối cảnh các hệ thống NoSQL ngày càng được sử dụng rộng rãi.

1. **BACKGROUND**
2. **So sánh khác nhau giữa SQL và NoSQL**

Tại đây chúng tôi đưa ra một vài sự khác nhau cơ bản của 2 loại cơ sở dữ liệu này để làm tiền đề cho mục sau:

|  |  |  |
| --- | --- | --- |
| Đặc điểm | SQL | NoSQL |
| Cấu trúc | Có cấu trúc cố định, tổ chức theo bảng với sơ đồ cố định. Mỗi bảng có các hàng (bản ghi) và cột (thuộc tính). | Không có sơ đồ cố định hoặc có sơ đồ động, tổ chức theo nhiều định dạng khác nhau (key-value, tài liệu (document), cột (collumn), đồ thị (graph)) giúp linh hoạt hơn. |
| Khả năng mở rộng | Chủ yếu mở rộng theo chiều dọc (thêm tài nguyên vào một máy chủ). | Mở rộng theo chiều ngang (thêm nhiều máy chủ để xử lý tải). |
| Tính nhất quán | Đảm bảo các thuộc tính ACID (Tính nguyên tử, Nhất quán, Cách ly, Bền vững) để đảm bảo độ tin cậy và toàn vẹn của dữ liệu. | Sử dụng các thuộc tính BASE (Cơ bản sẵn có, Trạng thái mềm, Cuối cùng nhất quán) cho tính linh hoạt, đặc biệt trong các môi trường phân tán. |
| Ngôn ngữ truy vấn | Sử dụng SQL, một ngôn ngữ truy vấn chuẩn với nhiều tính năng phong phú, bao gồm cả các phép nối và truy vấn phức tạp. | Không có ngôn ngữ chuẩn hóa; mỗi loại NoSQL có thể có phương pháp truy vấn riêng, thường thiếu các phép nối và khả năng truy vấn phức tạp. |
| Bảo mật | Cung cấp các cơ chế bảo mật mạnh mẽ với dữ liệu có cấu trúc, cho phép quản lý bảo mật dễ dàng hơn. | Thường có ít biện pháp bảo mật tích hợp hơn, đặc biệt là với dữ liệu không có cấu trúc, đòi hỏi quản lý bảo mật bổ sung trong các môi trường chia nhỏ dữ liệu​. |
| Quan hệ dữ liệu | Dữ liệu được chuẩn hóa cao để tránh dư thừa; các quan hệ giữa các bảng được xác định rõ ràng thông qua các khóa và ràng buộc. | Thường không xác định quan hệ, dữ liệu được lưu trữ theo định dạng không chuẩn hóa, có thể dẫn đến dư thừa nhưng cải thiện hiệu suất truy xuất trong hệ thống phân tán. |

1. **Tổng quan về SQL Injection**

SQL Injection (SQLi) là một trong những kiểu tấn công phổ biến nhất nhằm khai thác các lỗ hổng trong ứng dụng web có sử dụng cơ sở dữ liệu quan hệ (SQL databases). SQL Injection xảy ra khi tin tặc chèn mã độc SQL vào các truy vấn được thực thi trong cơ sở dữ liệu để truy xuất hoặc thay đổi dữ liệu bất hợp pháp. Các cuộc tấn công SQL Injection có thể dẫn đến rò rỉ dữ liệu nhạy cảm, thay đổi dữ liệu hoặc thậm chí làm gián đoạn hệ thống.

Phương thức tấn công SQL Injection có nhiều biến thể như Union-based SQL Injection, Error-based SQL Injection, Time-base SQL Injection và Blind SQL Injection. Mỗi phương thức tấn công khai thác các điểm yếu khác nhau trong việc xác thực và kiểm tra tính hợp lệ của đầu vào. Trong các phương pháp Union-based SQL Injection, tin tặc sử dụng mệnh đề UNION để kết hợp kết quả truy vấn bất hợp pháp vào kết quả hợp lệ, Error-based SQL Injection lợi dụng thông tin trả về từ các lỗi cơ sở dữ liệu, Time-base SQL injection lại dựa vào tốc độ phản hồi của câu truy vấn để xác định tính hợp lệ của mã độc, còn Blind SQL Injection khai thác những nơi không có phản hồi trực tiếp từ cơ sở dữ liệu.

Các chiến lược phòng chống SQL Injection bao gồm việc sử dụng câu truy vấn chuẩn bị trước (Prepared Statements) và Stored Procedures, kiểm tra và xác thực đầu vào (Input Validation), mã hóa và loại bỏ các ký tự đặc biệt, và phân quyền (Access Control). Các biện pháp này đã cho thấy hiệu quả trong việc ngăn ngừa SQL Injection trên các cơ sở dữ liệu SQL truyền thống, nhưng chúng không thể hoàn toàn loại bỏ rủi ro khi không được triển khai đồng nhất hoặc không tuân thủ các nguyên tắc bảo mật một cách chặt chẽ. Nhiều nghiên cứu cũng đã được thực hiện nhằm đánh giá hiệu quả của các phương pháp phòng chống, từ đó cho thấy việc sử dụng Prepared Statements và Input Validation là hai chiến lược quan trọng nhất trong phòng chống SQL Injection.

1. **Tổng quan NoSQL Injection**

NoSQL Injection là một dạng tấn công tương tự SQL Injection nhưng xảy ra trong các hệ thống cơ sở dữ liệu phi quan hệ (NoSQL). Do các hệ quản trị NoSQL không sử dụng cấu trúc truy vấn SQL mà dựa trên các dạng cấu trúc dữ liệu linh hoạt như JSON, XML hoặc các API đặc thù, nên phương thức tấn công NoSQL Injection cũng có những khác biệt đáng kể. Một số hệ quản trị NoSQL phổ biến như MongoDB, CouchDB, và Cassandra có cách xử lý truy vấn và kiểm tra đầu vào khác nhau, dẫn đến các dạng tấn công NoSQL Injection cũng không hoàn toàn giống nhau.

Trong MongoDB, một trong những hệ NoSQL phổ biến nhất, NoSQL Injection xảy ra khi các truy vấn JSON chứa các cấu trúc JSON độc hại hoặc khi các toán tử truy vấn như $ne, $gt, $lt được sử dụng để sửa đổi điều kiện truy vấn. Trong CouchDB, NoSQL Injection cũng có thể xảy ra khi các mã truy vấn bằng JavaScript không được kiểm tra chặt chẽ. Đối với Cassandra, mặc dù ít gặp NoSQL Injection hơn do hệ thống sử dụng CQL (Cassandra Query Language) gần giống SQL, nhưng vẫn tồn tại các nguy cơ khi các lệnh truy vấn không được xác thực đầu vào đầy đủ.

Để phòng chống NoSQL Injection, một số phương pháp đã được đề xuất và áp dụng như: kiểm tra tính hợp lệ của truy vấn (Query Validation), xác thực và mã hóa đầu vào (Input Validation & Encoding), và sử dụng các thư viện hỗ trợ (Middleware Libraries) để lọc và chuẩn hóa truy vấn. Tuy nhiên, do NoSQL Injection là mối đe dọa tương đối mới và chưa có nhiều nghiên cứu chuyên sâu, nên các biện pháp phòng chống vẫn đang trong giai đoạn thử nghiệm và cải tiến.

1. **So sánh các biện pháp phòng chống SQL và NoSQL Injection trong các nghiên cứu hiện tại**

Trong những năm gần đây, nhiều nghiên cứu đã so sánh các chiến lược phòng chống SQL Injection và NoSQL Injection. Một số nghiên cứu cho thấy các biện pháp phòng chống SQL Injection có thể được áp dụng cho NoSQL với một số điều chỉnh nhất định. Chẳng hạn, việc xác thực đầu vào (Input Validation) là một biện pháp chung có thể áp dụng cho cả SQL và NoSQL để ngăn chặn các mã độc. Tuy nhiên, các nghiên cứu cũng chỉ ra rằng một số biện pháp dành riêng cho SQL, như Prepared Statements, không thể được triển khai trong các hệ thống NoSQL vì khác biệt về ngôn ngữ truy vấn và cấu trúc dữ liệu.

Các nghiên cứu trước đây cũng nhấn mạnh vai trò của việc hạn chế quyền truy cập và sử dụng các công cụ bảo mật tự động như Web Application Firewalls (WAF) để giảm thiểu rủi ro từ NoSQL Injection. Bên cạnh đó, một số nghiên cứu đã đề xuất các phương pháp như Query Whitelisting (cho phép các truy vấn hợp lệ) và Policy Enforcement (tuân thủ chính sách bảo mật) cho các hệ NoSQL, nhưng tính khả thi và hiệu quả của các phương pháp này vẫn chưa được chứng minh đầy đủ.

Nghiên cứu của chúng tôi sẽ tiếp tục dựa trên nền tảng của các công trình trước đây, đồng thời mở rộng các phương pháp phòng chống để đảm bảo tính an toàn cho các hệ thống NoSQL hiện đại.

1. **Phương pháp nghiên cứu**
2. **Phương pháp đánh giá chiến lược bảo mật**

Để đánh giá tính hiệu quả của các biện pháp phòng chống, chúng tôi sẽ áp dụng cách tiếp cận so sánh giữa hai loại cơ sở dữ liệu SQL và NoSQL. Quá trình đánh giá bao gồm:

1. Phân tích lý thuyết: Đánh giá từng phương pháp phòng ngừa như Prepared Statements (cho SQL), Query Validation (cho NoSQL), và các biện pháp chung như kiểm tra đầu vào. Chúng tôi sẽ phân tích các nghiên cứu trước đây và so sánh các khía cạnh như tính khả thi, hiệu quả, và tính linh hoạt của từng chiến lược.
2. Phân tích thực nghiệm: Thử nghiệm trực tiếp các kỹ thuật phòng chống trên các cơ sở dữ liệu SQL (MySQL) và NoSQL (MongoDB). Mục tiêu là đánh giá khả năng ngăn chặn và giảm thiểu tấn công, đồng thời xem xét tác động của các biện pháp này đối với hiệu năng và trải nghiệm người dùng.
3. **Công cụ và môi trường thử nghiệm**

Nghiên cứu sẽ sử dụng các nền tảng và công cụ sau để thực hiện thử nghiệm:

* Cơ sở dữ liệu SQL: Sử dụng MySQL do tính phổ biến và khả năng hỗ trợ SQL Injection.
* Cơ sở dữ liệu NoSQL: Sử dụng MongoDB vì đây là hệ NoSQL phổ biến và có nhiều khả năng bị tấn công NoSQL Injection.
* Công cụ tấn công: Sử dụng Burp Suite và SQLmap với 50 câu lệnh injection cho mỗi loại.
* Môi trường thử nghiệm: Thiết lập một môi trường thử nghiệm bao gồm các dịch vụ web tích hợp với cả MySQL và MongoDB, đảm bảo cách ly các thử nghiệm để tránh ảnh hưởng đến dữ liệu sản xuất hoặc các hệ thống khác.

1. **Quy trình thử nghiệm**

Quy trình thử nghiệm được xây dựng để đo lường mức độ hiệu quả của từng chiến lược bảo mật trong các tình huống tấn công SQL và NoSQL Injection:

Bước 1: Cấu hình môi trường thử nghiệm và xác định các điểm đầu vào không an toàn cho cả MySQL và MongoDB. Các điểm đầu vào này sẽ được sử dụng để thử nghiệm các kỹ thuật tấn công injection.

Bước 2: Thực hiện các tấn công SQL Injection và NoSQL Injection vào từng loại cơ sở dữ liệu mà không có bất kỳ biện pháp bảo mật nào để ghi nhận 50 câu lệnh chắc chắn thành công.

Bước 3: Triển khai từng biện pháp phòng chống lần lượt, bao gồm:

* Prepared Statements và Stored Procedures cho SQL.
* Schema Validation cho NoSQL.
* Các biện pháp chung như Input Sanitization và Web Application Firewall (WAF) cho cả hai hệ.

Bước 4: Thực hiện lại các tấn công sau khi đã triển khai từng biện pháp phòng ngừa và ghi nhận kết quả, bao gồm:

* Khả năng ngăn chặn cuộc tấn công của từng chiến lược.
* Tác động đến hiệu năng hệ thống khi áp dụng các biện pháp bảo mật.
* Mức độ phức tạp và khả năng áp dụng của từng chiến lược phòng chống.

Bước 5: Lập lại bước 1, 2 và 3 thêm 2 lần và hoàn thiện từng chiến lược

Bước 6: Phân tích và đánh giá các kết quả, từ đó rút ra kết luận về mức độ hiệu quả của từng chiến lược trong việc phòng chống SQL Injection và NoSQL Injection.

Quy trình này sẽ cung cấp các dữ liệu thực nghiệm cụ thể và có căn cứ để đưa ra so sánh toàn diện, giúp đưa ra khuyến nghị về những chiến lược bảo mật tối ưu cho từng loại cơ sở dữ liệu trong các tình huống thực tế.

1. **Kết quả và phân tích**
2. **Kết quả**

|  |  |  |
| --- | --- | --- |
| Biện pháp bảo vệ | Tỉ lệ phòng chống SQL Injection | Tỉ lệ phòng chống NoSQL Injection |
| Prepared Statement | 85% - 90% | Không hỗ trợ |
| Store Procedures | 40% - 60% (nếu kiểm soát tốt quyền truy cập) | Không hỗ trợ |
| Schema Validation | 0% | 40% (đối với các injection nhắn vào Id) |
| Input validation | 50 - 80% (không sử dụng prepared statement) | 90% - 100% |
| WAF | 90% - 100% | 80% - 100% |

1. **Phân tích**

Dựa trên kết quả thử nghiệm, các biện pháp bảo vệ như Prepared Statements, Schema Validation, Input Validation, và Web Application Firewall (WAF) đều mang lại các mức độ hiệu quả khác nhau đối với SQL Injection và NoSQL Injection. Tuy nhiên, sự khác biệt trong kiến trúc và truy vấn của SQL và NoSQL làm cho hiệu quả của các chiến lược này thay đổi đáng kể:

**Prepared Statements:**

* SQL Injection: Prepared Statements là phương pháp hiệu quả và dễ triển khai nhất trong việc ngăn chặn SQL Injection, với khả năng đạt tỷ lệ bảo vệ từ 85-90%, nhờ vào việc tách biệt dữ liệu đầu vào khỏi mã truy vấn. Stored Procedures tuy cũng có hiệu quả nhưng thấp hơn (khoảng 40-60%) vì phụ thuộc vào cách triển khai và quyền truy cập. Tuy nhiên, trong một số trường hợp, Prepared Statements không thể sử dụng, chẳng hạn như khi truy vấn sử dụng điều kiện ORDER BY với một biểu thức không phải chuỗi (string). Vì Prepared Statements sẽ chuyển đổi tất cả các giá trị đầu vào thành chuỗi, điều này gây ra vấn đề khi sử dụng trong ORDER BY. Ví dụ, ORDER BY "id" sẽ không thể thực hiện được, trong khi ORDER BY id lại có thể thực hiện bình thường.
* NoSQL Injection: Không hỗ trợ vì các hệ thống NoSQL không sử dụng ngôn ngữ truy vấn SQL chuẩn.

**Stored Procedures**

* SQL Injection: hiệu quả tương đối (40-60%) vì phụ thuộc vào cách triển khai và quyền truy cập.

**Schema Validation:**

* SQL Injection: Vì SQL sử dụng cấu trúc bảng cố định và không xác thực dữ liệu schema.
* NoSQL Injection: Hiệu quả ở mức 40%, đặc biệt với các trường hợp injection nhắm vào \_id. Tuy nhiên, việc phụ thuộc vào cấu trúc dữ liệu linh hoạt khiến giải pháp này không toàn diện.

**Input Validation:**

* SQL Injection: Đạt hiệu quả 50-80% khi không sử dụng Prepared Statements, nhưng khả năng ngăn chặn phụ thuộc vào mức độ nghiêm ngặt của quy tắc kiểm tra đầu vào.
* NoSQL Injection: Hiệu quả cao và được xem là phương pháp chính trong việc phòng chống loại tấn công này, đạt 90-100%, vì trực tiếp ngăn chặn các mã độc JSON hoặc các toán tử truy vấn nguy hiểm như $ne, $gt. Ngoài ra, trong một số trường hợp cụ thể khi kết hợp sử dụng với white list (danh sách các phần tử được phép xuất hiện trong câu lệnh) hoặc black list (danh sách các phần tử không được phép xuất hiện trong câu lệnh) thì việc ngăn chặn một số loại NoSQL Injection là 100%..

**Web Application Firewall (WAF):**

* SQL Injection: Hiệu quả cao, đạt 90-100%, nhờ khả năng chặn các mẫu truy vấn độc hại từ phía máy chủ. Nhưng hiệu quả phụ thuộc vào cấu hình và độ chính xác của các quy tắc bảo vệ.
* NoSQL Injection: Hiệu quả trong khoảng 80-100%, nhưng hiệu quả phụ thuộc vào cấu hình và độ chính xác của các quy tắc bảo vệ

1. **Đánh giá ưu và nhược điểm của từng phương pháp**

|  |  |  |
| --- | --- | --- |
| **Biện pháp** | **Ưu điểm** | **Nhược điểm** |
| Prepared Statements | Hiệu quả cao với SQL Injection, dễ triển khai. | Không áp dụng được cho NoSQL.  Có thể có một số trường hợp ngoại lệ. |
| Stored Procedures | Kết hợp tốt với quyền truy cập, giảm thiểu truy vấn trực tiếp. | Hiệu quả thấp hơn Prepared Statements, khó mở rộng.  Thiếu tính linh hoạt. |
| Schema Validation | Bảo vệ tốt cho NoSQL khi áp dụng đúng cách.  Giảm đáng kể các loại injection có tính nguy hiểm cao liên quan. | Hạn chế với dữ liệu động hoặc không có schema cố định. |
| Input Validation | Hiệu quả với cả SQL và NoSQL, dễ dàng mở rộng và tùy chỉnh. | Yêu cầu quy trình phát triển nghiêm ngặt, phụ thuộc vào mức độ kiểm tra.  Dễ bỏ sót một số phần quan trọng. |
| WAF | Cung cấp lớp bảo vệ bổ sung, giảm thiểu rủi ro từ các truy vấn độc hại. | Chi phí thiết lập cao nếu sử dụng các nguồn của bên thứ ba như AWS.  Phụ thuộc nhiều vào khả năng cấu hình và khả năng nhận diện các rủi ro. |

1. **THẢO LUẬN**

Qua nghiên cứu, có thể thấy rằng SQL Injection và NoSQL Injection tuy có chung bản chất là khai thác các lỗ hổng trong việc xử lý truy vấn cơ sở dữ liệu, nhưng sự khác biệt trong kiến trúc và phương pháp truy vấn đã dẫn đến các yêu cầu và chiến lược phòng chống khác nhau.

Với SQL Injection, các giải pháp như Prepared Statements và Stored Procedures đã chứng minh tính hiệu quả cao nhờ khả năng cách ly dữ liệu đầu vào và truy vấn. Tuy nhiên, những phương pháp này đòi hỏi sự nhất quán trong triển khai chúng không áp dụng được cho các hệ NoSQL vốn không sử dụng ngôn ngữ truy vấn SQL chuẩn.

Đối với NoSQL Injection, các chiến lược như Schema Validation và Input Validation đóng vai trò quan trọng, đặc biệt là trong việc ngăn chặn các truy vấn độc hại nhắm vào cấu trúc JSON hay các toán tử đặc thù như $ne, $gt. Tuy nhiên, sự linh hoạt trong cấu trúc dữ liệu NoSQL cũng tạo ra thách thức lớn, đòi hỏi các chiến lược này phải được tùy chỉnh và tối ưu liên tục. Ví dụ, Schema Validation hiệu quả trong các hệ thống NoSQL có định dạng dữ liệu tương đối cố định nhưng lại không đáp ứng tốt đối với dữ liệu phi cấu trúc hoặc linh hoạt hoàn toàn.

Web Application Firewall (WAF) nổi bật với khả năng cung cấp một lớp bảo vệ bổ sung cho cả hai loại cơ sở dữ liệu. Tuy nhiên, hiệu quả của WAF phụ thuộc nhiều vào cách cấu hình và khả năng nhận diện các mẫu truy vấn độc hại. Chi phí triển khai WAF có thể là rào cản đối với các tổ chức nhỏ hoặc có ngân sách hạn chế.

Một vấn đề cần thảo luận thêm là tác động của các biện pháp bảo mật lên hiệu năng và trải nghiệm người dùng. Các biện pháp như Input Validation hay Prepared Statements có thể làm tăng độ trễ truy vấn nếu không được tối ưu hóa, trong khi việc triển khai WAF có thể dẫn đến chi phí vận hành tăng cao. Do đó, việc lựa chọn và kết hợp các chiến lược cần dựa trên đặc thù hệ thống và mức độ ưu tiên giữa bảo mật, hiệu suất và chi phí.

1. **KẾT LUẬN**

Bài nghiên cứu đã phân tích và so sánh các chiến lược phòng ngừa SQL Injection và NoSQL Injection, hai hình thức tấn công nghiêm trọng đối với cơ sở dữ liệu. Qua việc đánh giá các biện pháp như Prepared Statements, Schema Validation, Input Validation và Web Application Firewall (WAF), chúng tôi nhận thấy rằng mỗi phương pháp đều có ưu và nhược điểm nhất định, phụ thuộc vào loại cơ sở dữ liệu và kiến trúc hệ thống.

SQL Injection đã được nghiên cứu sâu và có nhiều giải pháp hiệu quả, trong đó Prepared Statements là lựa chọn hàng đầu. Tuy nhiên, với NoSQL Injection, việc thiếu chuẩn hóa và sự đa dạng của các hệ cơ sở dữ liệu NoSQL đặt ra thách thức lớn hơn. Các biện pháp như Schema Validation và Input Validation được xác định là có hiệu quả cao, nhưng cần được tối ưu và áp dụng phù hợp với đặc thù của từng hệ thống đặt ra nhiều thách thức đối với lập trình viên hay người vận hành hệ thống.

Nghiên cứu này không chỉ nhấn mạnh tầm quan trọng của việc lựa chọn biện pháp bảo mật phù hợp mà còn đề xuất sự kết hợp giữa nhiều chiến lược để tăng cường khả năng bảo vệ. Trong bối cảnh các hệ thống dữ liệu NoSQL ngày càng phổ biến, việc tiếp tục nghiên cứu và cải tiến các phương pháp phòng chống NoSQL Injection là rất cần thiết, góp phần bảo vệ dữ liệu và tối ưu hóa an ninh thông tin cho các tổ chức, doanh nghiệp.

1. **Reference**

[1] Sahatqija, K. et al. (2018) ‘Comparison between relational and NOSQL databases’, 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 0216–0221. doi:10.23919/mipro.2018.8400041.

[2] Gupta, A. et al. (2017) ‘NoSQL databases: Critical Analysis and comparison’, 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), pp. 293–299. doi:10.1109/ic3tsn.2017.8284494.

[3] Kumar, J. and Garg, V. (2017) ‘Security analysis of unstructured data in NOSQL mongodb database’, 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), pp. 300–305. doi:10.1109/ic3tsn.2017.8284495.

[4] Okman, L. et al. (2011) ‘Security issues in nosql databases’, 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 541–547. doi:10.1109/trustcom.2011.70.

[5] V O, J. and Subramanian, N. (2022) ‘Secure-D:framework for detecting and preventing attacks in SQL and NoSQL databases’, 2022 10th International Symposium on Digital Forensics and Security (ISDFS), pp. 1–5. doi:10.1109/isdfs55398.2022.9800805.

[6] Vonitsanos, G. et al. (2020) ‘Security and privacy solutions associated with NoSQL Data Stores’, 2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMA, pp. 1–5. doi:10.1109/smap49528.2020.9248442.

[7] NoSQL injection Web Security Academy. Available at: <https://portswigger.net/web-security/nosql-injection>.

[8] A03:2021 – injection A03 Injection - OWASP Top 10:2021. Available at: <https://owasp.org/Top10/A03_2021-Injection>.

[9] SQL injection prevention cheat sheet SQL Injection Prevention - OWASP Cheat Sheet Series. Available at: <https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html>.

[10] JDriven (2017) SQL Injection: When a prepared statement is not enough..., JDriven Blog. Available at: <https://blog.jdriven.com/2017/10/sql-injection-prepared-statement-not-enough>.