BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

BÁO CÁO ĐỒ ÁN

TÌM HIỂU VỀ CÁC KỸ THUẬT MÃ HOÁ CHO HỆ QUẢN TRỊ CSDL MYSQL

Ngành: **CÔNG NGHỆ THÔNG TIN**

Tên học phần: **BẢO MẬT THÔNG TIN**

Giảng viên hướng dẫn : Nguyễn Trọng Minh Hồng Phước

Sinh viên thực hiện : Nguyễn Quang Hưng

MSSV: 2380600933 Lớp: 23DTHE1

TP. Hồ Chí Minh, Tháng 12 năm 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

BÁO CÁO ĐỒ ÁN

TÌM HIỂU VỀ CÁC KỸ THUẬT MÃ HOÁ CHO HỆ QUẢN TRỊ CSDL MYSQL

Ngành: **CÔNG NGHỆ THÔNG TIN**

Tên học phần: **BẢO MẬT THÔNG TIN**

Giảng viên hướng dẫn : Nguyễn Trọng Minh Hồng Phước

Sinh viên thực hiện : Nguyễn Quang Hưng

MSSV: 2380600933 Lớp: 23DTHE1

TP. Hồ Chí Minh, Tháng 12 năm 2025

Mục lục

Lời mở đầu

Nội dung….

Lời cam kết

Tôi xin cam đoan đề tài nghiên cứu “Tìm hiểu về các kỹ thuật mã hóa cho hệ quản trị CSDL mySQL” mà tôi thực hiện là công trình nghiên cứu của riêng tôi, dưới sự hướng dẫn trực tiếp của giảng viên Nguyễn Trọng Minh Hồng Phước.

Toàn bộ thông tin, nội dung được nêu trong bài báo cáo môn học này tuyệt đối không có việc sao nội dung từ những công trình nghiên cứu trước đó. Nếu phát hiện bất cứ gian dối nào trong bài báo cáo, tôi xin chịu hoàn toàn trách nhiệm và nhận mọi sự xử lý, kỷ luật từ nhà trường.

Lời cảm ơn

Trong quá trình học tập và hoàn thành báo cáo này, tôi đã nhận được rất nhiều sự giúp đỡ quý báu. Trước tiên, tôi xin bày tỏ lòng biết ơn sâu sắc đến Ban Giám hiệu Nhà trường, Khoa Công nghệ Thông tin và toàn thể quý Thầy Cô Trường Đại học Công nghệ TP. HCM đã tạo điều kiện học tập tốt nhất và trang bị cho tôi nền tảng kiến thức vững chắc trong suốt thời gian qua.

Tôi xin gửi lời cảm ơn chân thành và sâu sắc nhất đến Thầy Nguyễn Trọng Minh Hồng Phước - giảng viên hướng dẫn môn Bảo mật Thông tin. Thầy đã tận tình hướng dẫn, định hướng nghiên cứu, cung cấp những tài liệu quý giá và đưa ra những góp ý sâu sắc, thiết thực trong suốt quá trình tôi thực hiện đồ án "Tìm hiểu về các kỹ thuật mã hóa cho hệ quản trị CSDL MySQL". Những chỉ dẫn tận tâm của Thầy là yếu tố then chốt giúp tôi có thể hoàn thành báo cáo này.

Mặc dù đã có nhiều nỗ lực, nhưng do hạn chế về kinh nghiệm, báo cáo không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được sự đóng góp ý kiến từ quý Thầy Cô và các bạn để đề tài nghiên cứu của tôi được hoàn thiện hơn.

Một lần nữa, tôi xin chân thành cảm ơn.

Tóm tắt nội dung

Tóm tắt …

Chương 1: Tổng quan về Bảo mật trong Cơ sở dữ liệu MySQL

1.1. Khái niệm bảo mật cơ sở dữ liệu

Bảo mật cơ sở dữ liệu (Database Security) là tập hợp các biện pháp, kỹ thuật và chính sách nhằm bảo vệ dữ liệu khỏi truy cập trái phép, lạm dụng, phá hoại, thất thoát hoặc rủi ro từ các tác nhân bên trong và bên ngoài hệ thống.

Trong bối cảnh chuyển đổi số, dữ liệu được xem là "Tài sản số" quan trọng nhất của mọi tổ chức. Các cuộc tấn công mạng nhằm vào cơ sở dữ liệu ngày càng tinh vi, từ tấn công SQL Injection, khai thác lỗ hổng bảo mật, đến các mối đe dọa nội gián. Do đó, việc bảo đảm an toàn cho dữ liệu không còn là một lựa chọn mà trở thành yêu cầu sống còn đối với mọi hệ thống thông tin.

Một hệ thống CSDL an toàn phải đảm bảo 3 mục tiêu cốt lõi theo mô hình **CIA**:

- **Confidentiality (Tính bí mật):** Chỉ những đối tượng được ủy quyền mới có quyền truy cập dữ liệu.
- **Integrity (Tính toàn vẹn):** Dữ liệu không bị thay đổi, chỉnh sửa, sao chép một cách trái phép.
- **Availability (Tính sẵn sàng):** Dữ liệu luôn có thể truy xuất, sao lưu khi cần thiết.

Là một trong những hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến nhất thế giới, MySQL là mục tiêu của nhiều cuộc tấn công. Do đó, việc hiểu rõ và triển khai các biện pháp bảo mật cho MySQL là vô cùng cấp thiết. Các biện pháp này có thể được áp dụng ở nhiều cấp độ, từ việc quản lý người dùng và phân quyền, bảo vệ kết nối, cho đến việc mã hóa dữ liệu – trọng tâm của đồ án này – nhằm trực tiếp bảo vệ tính bí mật (Confidentiality) của dữ liệu ngay cả khi nó bị đánh cắp.

1.2. Vai trò của bảo mật trong hệ quản trị cơ sở dữ liệu

Hệ quản trị cơ sở dữ liệu *(HQTCSDL)* đóng vai trò là trái tim của hầu hết các ứng dụng, nơi lưu trữ và quản lý toàn bộ dữ liệu trọng yếu của tổ chức. Do đó, bảo mật không còn là một tính năng phụ thêm, mà là **chức năng nền tảng** và **trách nhiệm cốt**

lõi của mọi HQTCSDL hiện đại. Vai trò của bảo mật trong HQTCSDL thể hiện qua các khía cạnh then chốt sau.

1.2.1. Bảo vệ tài sản số trọng yếu

CSDL là nơi tập trung các dữ liệu nhạy cảm nhất, từ thông tin cá nhân, bí mật kinh doanh, đến hồ sơ tài chính, bí mật lõi và giao dịch của hệ thống. Một sự cố bảo mật xảy ra với CSDL có thể dẫn đến những hậu quả nghiêm trọng và đa tầng:

- **Tổn thất kinh tế trực tiếp:** Chi phí khắc phục sự cố, bồi thường cho khách hàng, và mất doanh thu từ gián đoạn kinh doanh.
- **Tổn hại danh tiếng và sự tin cậy:** Uy tín của tổ chức bị ảnh hưởng nghiêm trọng, dẫn đến mất lòng tin từ khách hàng và đối tác.
- **Hệ quả pháp lý:** Vi phạm các quy định về bảo vệ dữ liệu như GDPR, Luật An ninh mạng Việt Nam, hoặc các chuẩn ngành như PCI-DSS (với thanh toán) và HIPAA (với y tế), kéo theo các khoản phạt nặng nề.
- **Mất cân bằng cạnh tranh:** Rò rỉ bí mật thương mại, chiến lược kinh doanh có thể làm mất lợi thế cạnh tranh.

1.2.2. Đảm bảo tính toàn vẹn của hệ thống thông tin

Dữ liệu bị xuyên tạc hoặc phá hoại sẽ dẫn đến các quyết định chiến lược kinh doanh sai lầm, các giao dịch lỗi và làm mất ý nghĩa của toàn bộ hệ thống hoặc có thể bị khai tác tạo các giao dịch bất hợp pháp. Bảo mật HQTCSDL đảm bảo:

- **Tính toàn vẹn của dữ liệu:** Ngăn chặn việc thay đổi dữ liệu trái phép.
- **Tính sẵn sàng của dịch vụ:** Chống lại các cuộc tấn công từ chối dịch vụ (DDoS) nhằm vào CSDL, đảm bảo hệ thống vận hành liên tục.

1.2.3. Nền tảng cho việc tuân thủ các chuẩn bảo mật

Các quy định và tiêu chuẩn bảo mật toàn cầu ngày càng nghiêm ngặt và chặt chẽ (như GDPR, PCI-DSS) đều có những yêu cầu cụ thể về bảo vệ dữ liệu. Đối với một HQTCSDL được bảo mật tốt, với các cơ chế như **mã hóa dữ liệu**, **kiểm soát truy cập chi tiết (RBAC)** và **ghi nhật ký kiểm toán (Auditing)**, cung cấp nền tảng kỹ thuật để tổ chức chứng minh việc tuân thủ một cách rõ ràng và hiệu quả. Hầu hết các tổ chức hay các doanh nghiệp đều tuân thủ theo các chuẩn bảo mật.

1.2.4. Vai trò then chốt của HQTCSDL trong kiến trúc bảo mật

Trong kiến trúc bảo mật tổng thể, bảo mật ứng dụng và bảo mật mạng đóng vai trò như những "Bức tường thành" bên ngoài. Tuy nhiên, một khi các lớp phòng thủ này bị xuyên thủng, **bảo mật ở cấp độ CSDL chính là "Lớp phòng thủ cuối cùng"** để bảo vệ dữ liệu. Các kỹ thuật như mã hóa dữ liệu ngay trong CSDL (mà chúng ta sẽ tìm hiểu trong đồ án này) đảm bảo rằng ngay cả khi kẻ tấn công có tiếp cận được đến tầng CSDL thì chúng cũng không thể đọc được thông tin có ý nghĩa.

1.3. Các rủi ro bảo mật phổ biến đối với cơ sở dữ liệu

Các rủi ro bảo mật đa dạng từ tấn công kỹ thuật đến lỗi vận hành đã cho thấy một thực tế: không có một "Bức tường lửa" duy nhất nào có thể bảo vệ toàn diện cho CSDL. Một chiến lược bảo mật hiệu quả phải là sự kết hợp của nhiều lớp phòng thủ, từ ứng dụng, mạng lưới, quản lý truy cập, cho đến các giải pháp bảo vệ dữ liệu ở cấp độ sâu như **mã hóa** – giải pháp then chốt để giảm thiểu thiệt hại từ các rủi ro liên quan đến việc đánh cắp dữ liệu trực tiếp.

Một hệ thống CSDL phải đối mặt với nhiều mối đe dọa khác nhau từ bên ngoài lẫn bên trong. Hiểu rõ các rủi ro này là bước đầu tiên để xây dựng các biện pháp phòng thủ hiệu quả. Các rủi ro có thể được phân loại thành các nhóm chính sau:

1.3.1. Tấn công khai thác lỗ hổng ứng dụng và dịch vụ

Đây là nhóm tấn công nhắm vào các điểm yếu trong kiến trúc hoặc mã nguồn của ứng dụng và lỗ hổng của chính HQTCSDL.

- **SQL Injection (Tấn công tiêm mã SQL):**
    - **Cơ chế:** Kẻ tấn công chèn các đoạn mã SQL độc hại vào các biến đầu vào của ứng dụng (như form đăng nhập, thanh tìm kiếm, …). Nếu ứng dụng không kiểm soát và xác thực đầu vào kỹ càng, đoạn mã này sẽ được thực thi trực tiếp trên CSDL.
    - **Hậu quả:** Cho phép kẻ tấn công đọc, sửa, xóa toàn bộ dữ liệu, vượt qua cơ chế xác thực, hoặc thậm chí chiếm quyền điều khiển máy chủ CSDL.
    - **Ví dụ trong MySQL:** Một câu truy vấn như SELECT * FROM users WHERE username = '[input]' nếu [input] là ' OR ' 1 '=' 1 sẽ trở thành điều kiện

luôn đúng, trả về thông tin của người dùng mà không cần kiểm tra điều kiện xác thực username.

- **Khai thác lỗ hổng phần mềm HQTCSDL:**
  - **Cơ chế:** Bản thân MySQL hoặc các plugin đi kèm có thể tồn tại các lỗ hổng bảo mật chưa được cập nhật bản sửa lỗi (zero-day hoặc các bản cũ).
  - **Hậu quả:** Kẻ tấn công có thể sử dụng các exploit (mã khai thác) để leo thang đặc quyền, thao tác xem, chỉnh sửa mã từ xa, hoặc làm sập dịch vụ.

1.3.2. Truy cập và đặc quyền trái phép

Rủi ro xuất phát từ việc quản lý quyền truy cập lỏng lẻo và kiểm tra điều kiện chưa được chặt chẽ. Điển hình như các lỗi sau:

- **Thiết lập quyền (Privilege) quá rộng rãi:** Cấp cho người dùng hoặc ứng dụng các quyền vượt quá mức cần thiết (ví dụ: quyền GRANT ALL cho một tài khoản ứng dụng thông thường). Khi tài khoản đó bị xâm phạm, thiệt hại sẽ được nhân lên gấp nhiều lần.
- **Leak thông tin xác thực:** Mật khẩu được lưu trữ dạng plaintext, gửi qua kênh không mã hóa, hoặc bị lấy cắp từ các nơi khác và dùng thử trên CSDL.
- **Đe dọa nội gián (Insider Threat):** Nhân viên, quản trị viên có quyền truy cập hợp pháp nhưng lạm dụng nó để sao chép, sửa đổi hoặc xóa dữ liệu trái phép. Đây là một trong những mối đe dọa khó phát hiện và phòng ngừa nhất.

1.3.3. Rủi ro từ việc lưu trữ và sao lưu dữ liệu

Dữ liệu không chỉ tồn tại trong bộ nhớ chính mà còn ở dạng "tĩnh" (data-at-rest).

- **Đánh cắp file dữ liệu vật lý:** Kẻ tấn công có thể truy cập trực tiếp vào các file dữ liệu của MySQL (như *.ibd, *.frm) thông qua lỗ hổng hệ điều hành hoặc truy cập vật lý vào ổ cứng.
- **Đánh cắp bản sao lưu (Backup):** Các file sao lưu (dump) thường không được bảo vệ và có thể bị lấy cắp, dẫn đến việc toàn bộ CSDL bị phơi bày. Nếu dữ liệu không được mã hóa, việc đánh cắp file dữ liệu hoặc file sao lưu đồng nghĩa với việc mất toàn bộ thông tin.

1.3.4. Tấn công mạng và nghe lén

Kiểu 1: Tấn công trung gian (Man-in-the-Middle): Kẻ tấn công chặn các gói tin trong đường truyền mạng giữa ứng dụng và máy chủ MySQL. Sử dụng các kĩ thuật trong mạng đường truyền để nhặt các thông tin nhạy cảm. Hậu quả cũng rất đáng lo ngại nếu kết nối không được mã hóa (bằng SSL/TLS), kẻ tấn công có thể đọc được toàn bộ truy vấn, kết quả trả về, và đặc biệt là thông tin đăng nhập.

Xem hình minh họa về tấn công Man-in-the-Middle tại Phụ lục, Hình 1

Kiểu 2: T**ấn công từ chối dịch vụ (DoS/DDoS):** Nhằm làm tràn lượng xử lý của máy chủ MySQL bằng một khối lượng truy vấn khổng lồ, khiến nó không thể phục vụ các yêu cầu hợp lệ, phá vỡ **tính sẵn sàng (Availability)** của hệ thống.

1.3.5. Cấu hình không an toàn

Đây là rủi ro phổ biến do lỗi của con người, mà hầu hết lỗi này sẽ bị khai thác bởi tin tặc. Dưới đây là các lỗi phổ biến nhất:

- **Sử dụng cấu hình mặc định:** Các cài đặt mặc định của MySQL có thể không an toàn (ví dụ: cho phép kết nối từ xa với quyền root, sử dụng cổng mặc định 3306 mà không có biện pháp bảo vệ).
- **Không cập nhật bản sửa lỗi mới nhất:** Không thường xuyên cập nhật phiên bản MySQL mới nhất để vá các lỗ hổng bảo mật.

1.4. Vai trò then chốt của mã hóa trong bảo mật cơ sở dữ liệu

Trong một kiến trúc bảo mật phân tầng, mỗi lớp (như tường lửa, hệ thống xác thực, kiểm soát truy cập) đều đóng một vai trò quan trọng. Tuy nhiên, khi các lớp phòng thủ bên ngoài này bị xuyên thủng, **mã hóa (Encryption)** nổi lên như "lớp phòng thủ cuối cùng" và là một trong những biện pháp bảo vệ mạnh mẽ nhất. Vai trò chiến lược của mã hóa được thể hiện qua các luận điểm cốt lõi sau:

1.4.1. Lớp bảo vệ cuối cùng cho dữ liệu "ở trạng thái tĩnh" (Data-at-Rest)

Đây là giá trị cốt lõi nhất của mã hóa trong CSDL. Nó trực tiếp giải quyết các kịch bản tấn công mà các biện pháp bảo mật khác không thể ngăn chặn được:

- B**ảo vệ ngay cả khi hệ thống thất thủ:** Ngay cả khi kẻ tấn công đã chiếm được toàn quyền truy cập vào máy chủ, đánh cắp trực tiếp các file dữ liệu vật lý của CSDL (như file .ibd, .frm trong MySQL) hoặc các bản sao lưu (backup),

chúng cũng không thể đọc được nội dung thực sự của dữ liệu. Dữ liệu đã được mã hóa sẽ chỉ là một chuỗi ký tự vô nghĩa nếu không có khóa giải mã chính xác.

- Giảm thiểu thiệt hại từ các mối đe dọa nội gián: Mã hóa giúp giảm rủi ro từ những nhân viên hoặc quản trị viên có quyền truy cập trực tiếp vào hệ thống lưu trữ nhưng không được ủy quyền xem dữ liệu nhạy cảm.

1.4.2. Bảo vệ dữ liệu "trên đường truyền" (Data-in-Transit)

Mã hóa không chỉ bảo vệ dữ liệu khi nó được lưu trữ, mà còn khi nó di chuyển giữa máy chủ CSDL và các ứng dụng.

Chống nghe lén và tấn công trung gian (Man-in-the-Middle): Bằng cách sử dụng các giao thức mã hóa như TLS/SSL (việc mà MySQL hỗ trợ), mọi truy vấn và kết quả trả về giữa client và server đều được mã hóa. Điều này ngăn chặn kẻ tấn công trên mạng chặn đọc thông tin nhạy cảm, đặc biệt là thông tin xác thực của người dùng.

1.4.3. Nền tảng không thể thiếu cho việc tuân thủ chính sách và pháp luật

Hầu hết các tiêu chuẩn bảo mật và quy định về bảo vệ dữ liệu cá nhân trên thế giới (như GDPR, PCI-DSS, HIPAA) đều yêu cầu hoặc khuyến nghị mạnh mẽ việc mã hóa dữ liệu nhạy cảm.

Minh chứng kỹ thuật cho việc tuân thủ: Việc triển khai mã hóa là một bằng chứng rõ ràng cho thấy tổ chức đã thực hiện các biện pháp kỹ thuật cần thiết để bảo vệ dữ liệu, từ đó giảm thiểu rủi ro pháp lý và các khoản phạt nặng nề.

1.4.4. Tăng cường tính toàn vẹn của mô hình bảo mật tổng thể

Mã hóa bổ sung và củng cố cho các biện pháp bảo mật khác:

- **Bổ sung cho Kiểm soát truy cập (Access Control):** Kiểm soát truy cập quyết định *"ai được phép truy cập dữ liệu"* thông qua hệ thống. Mã hóa đảm bảo rằng ngay cả khi quy tắc kiểm soát truy cập bị bẻ khóa, dữ liệu mà kẻ tấn công lấy được *vẫn không thể sử dụng được*.
- **Tạo các "** vùng **an toàn" bên trong CSDL:** Các kỹ thuật mã hóa ở cấp độ cột cho phép tạo ra các "vùng an toàn" ngay trong cùng một bảng dữ liệu, nơi chỉ những người dùng ứng dụng có khóa mới có thể đọc được thông tin nhạy cảm (như số thẻ tín dụng, mật khẩu đã băm), trong khi các dữ liệu khác (như ID, tên) vẫn có thể đọc được bình thường.

1.5. Kết luận

Chương 1 đã thiết lập một nền tảng lý thuyết toàn diện về bảo mật cơ sở dữ liệu, từ đó làm nổi bật sự cấp thiết và tính tất yếu của việc nghiên cứu các giải pháp bảo mật chuyên sâu, đặc biệt là mã hóa, trong bối cảnh hiện nay.

Tổng kết các luận điểm trọng tâm:

- **Bảo mật CSDL** đã được khẳng định là một yêu cầu sống còn, không đơn thuần là một tính năng kỹ thuật. Nó là tập hợp các biện pháp nhằm bảo vệ "tài sản số" quan trọng nhất của tổ chức trước một loạt các mối đe dọa từ bên trong lẫn bên ngoài. Mô hình CIA (Tính bí mật, Toàn vẹn, Sẵn sàng) đã cung cấp một khuôn khổ vững chắc để đánh giá một hệ thống CSDL an toàn.
- **Vai trò của HQTCSDL** cũng như **MySQL,** trong kiến trúc bảo mật là then chốt. Là trái tim của hệ thống thông tin, một HQTCSDL được bảo mật tốt không chỉ bảo vệ dữ liệu mà còn là nền tảng cho uy tín, hoạt động liên tục và khả năng tuân thủ pháp luật của tổ chức.
- **Phân tích chi tiết các rủi ro bảo mật phổ biến** (từ SQL Injection, khai thác lỗ hổng, đe dọa nội gián, cho đến đánh cắp dữ liệu trực tiếp) đã chỉ ra một thực tế quan trọng: không có một "phép màu" hay một lớp phòng thủ duy nhất nào có thể bảo vệ toàn diện cho CSDL. Các biện pháp truyền thống như tường lửa hay kiểm soát truy cập, dù quan trọng, vẫn có thể bị vượt qua.

Chính từ sự phân tích các rủi ro này, vai trò chiến lược của **mã hóa** đã được làm sáng tỏ. Mã hóa không thay thế các lớp bảo mật khác, mà nó bổ sung và củng cố chúng, đóng vai trò là **"lớp phòng thủ cuối cùng"** có hiệu quả thực tế cao nhất:

- Khi mọi lớp phòng thủ khác thất bại, mã hóa chính là rào chắn cuối cùng vô hiệu hóa giá trị của dữ liệu đã bị đánh cắp.
- Nó trực tiếp giải quyết các kịch bản tấn công nghiêm trọng nhất: đánh cắp file dữ liệu vật lý, đánh cắp bản sao lưu, và nghe lén đường truyền.
- Việc triển khai mã hóa biến một thảm họa rò rỉ dữ liệu tiềm tàng thành một sự cố đơn thuần, vì dữ liệu lúc này trong tay kẻ tấn công chỉ là một chuỗi ký tự vô nghĩa.

Chương 2: Các kỹ thuật mã hóa trong MySQL

Tiếp nối vai trò then chốt của mã hóa đã được phân tích ở Chương 1, Chương 2 sẽ đi sâu vào các kỹ thuật mã hóa cụ thể trong MySQL ...

2.1. Tổng quan về mã hóa trong MySQL

Mã hóa (Encryption) trong MySQL là tập hợp các cơ chế và kỹ thuật nhằm bảo vệ dữ liệu khỏi việc bị truy cập trái phép bằng cách biến dữ liệu gốc (Plaintext) thành dạng không thể đọc được (Ciphertext) nếu không có khóa giải mã tương ứng. MySQL không chỉ hỗ trợ mã hóa ở một cấp độ duy nhất mà cung cấp nhiều lớp và phương thức mã hóa khác nhau, phù hợp với từng kịch bản bảo mật cụ thể.

Về mặt tổng thể, các kỹ thuật mã hóa trong MySQL có thể được phân loại theo ba hướng chính như sau:

- Mã hóa dữ liệu lưu trữ (Data-at-Rest Encryption)
- Mã hóa dữ liệu truyền tải (Data-in-Transit Encryption)
- Mã hóa ở cấp độ logic (Mã hóa cột, Mã hóa ứng dụng)

Việc lựa chọn kỹ thuật phù hợp phụ thuộc vào mức độ nhạy cảm của dữ liệu, yêu cầu hiệu năng và khả năng quản lý khóa của hệ thống.

2.2. Mã hóa dữ liệu lưu trữ (Data-at-Rest Encryption)

Mã hóa dữ liệu lưu trữ là kỹ thuật mã hóa các tập tin dữ liệu vật lý của MySQL khi chúng được lưu trữ trên đĩa cứng hoặc các thiết bị lưu trữ khác. Đây là lớp bảo vệ cuối cùng ngăn chặn việc đánh cắp dữ liệu thông qua truy cập vật lý hoặc đánh cắp file Backup của hệ thống.

2.2.1. Mã hóa bảng không gian (Tablespace Encryption)

Từ phiên bản MySQL 5.7.11 trở đi, MySQL Enterprise Edition hỗ trợ InnoDB Tablespace Encryption, còn thường được gọi là "Transparent Data Encryption" (TDE). Tính năng này cho phép mã hóa toàn bộ hoặc một phần các Tablespace của InnoDB:

- Mã hóa toàn bộ Tablespace: Toàn bộ dữ liệu trong Tablespace được mã hóa tự động, bao gồm các bảng và chỉ mục.
- Mã hóa File-per-table: Mỗi bảng được lưu trong file riêng và có thể được mã hóa độc lập.

Cơ chế hoạt động:

- Sử dụng thuật toán AES (Advanced Encryption Standard) với độ dài khóa 128, 192 hoặc 256 bit.
- Sử dụng chế độ mã hóa khối ECB (Electronic Codebook) hoặc CBC (Cipher Block Chaining).
- Khóa mã hóa dữ liệu (DEK - Data Encryption Key) được lưu trữ trong phần header của tablespace và được bảo vệ bằng khóa chính (MEK - Master Encryption Key).
- MEK được lưu trữ trong file keyring bên ngoài MySQL.

Triển khai:

```sql
-- Tạo bảng với mã hóa
Create Table thongtin_nhansu (
    id int primary key,
    ten varchar(100),
    luong decimal(10, 2)
) Encryption = 'Y';

-- Bật mã hóa cho bảng
Alter Table thongtin_nhansu Encryption = 'Y';
```

Code Minh Hoạ 1: Triển khai Mã hóa TablespaceEncryption

2.2.2. Mã hóa file nhật ký (Redo Log và Undo Log Encryption)

Ngoài dữ liệu chính, MySQL còn cung cấp khả năng mã hóa các file nhật ký:

- Redo Log Encryption: Mã hóa nhật ký ghi lại các thay đổi dữ liệu.
- Undo Log Encryption: Mã hóa nhật ký để khôi phục các thay đổi chưa commit.

Điều này đảm bảo tính nhất quán và bảo mật toàn diện cho hệ thống.

2.2.3. Mã hóa file sao lưu (Backup Encryption)

Bên cạnh việc mã hóa trực tiếp dữ liệu đang vận hành, các file sao lưu (Backup) cũng là một mục tiêu tấn công phổ biến. Trên thực tế, nhiều sự cố rò rỉ dữ liệu nghiêm trọng không xuất phát từ hệ thống đang chạy, mà từ các file sao lưu bị đánh cắp hoặc lưu trữ không an toàn.

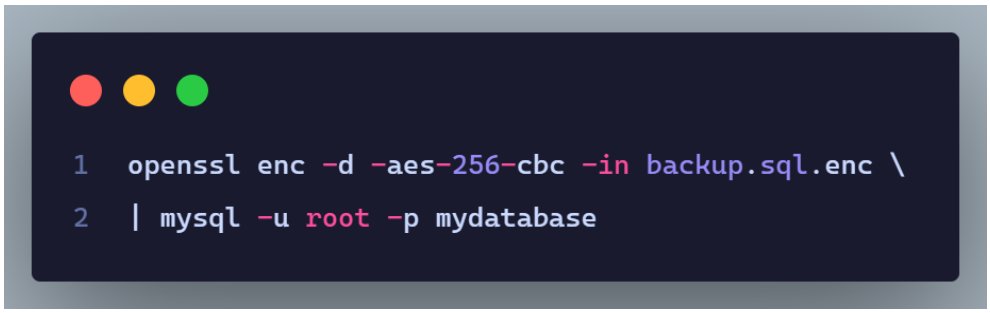
Các công cụ sao lưu của MySQL như *mysqldump* (logical backup) và *mysqlbackup* (physical backup – MySQL Enterprise Backup) đều hỗ trợ hoặc có thể kết hợp cơ chế mã hóa để bảo vệ dữ liệu sao lưu.

a) Mã hóa file sao lưu với mysqldump

mysqldump không mã hóa dữ liệu một cách trực tiếp ở mức CSDL, nhưng cho phép kết hợp với các công cụ mã hóa bên ngoài như OpenSSL hoặc GPG để mã hóa đầu ra.

Ví dụ sử dụng OpenSSL để mã hóa file dump:

Khi cần khôi phục dữ liệu:



Trong trường hợp này:

- File backup.sql.enc được mã hóa hoàn toàn
- Kẻ tấn công không thể đọc nội dung backup nếu không có khóa giải mã

b) Mã hóa file sao lưu với mysqlbackup (MySQL Enterprise Backup)

Đối với phiên bản MySQL Enterprise, công cụ mysqlbackup hỗ trợ mã hóa backup ở mức hệ thống với các tùy chọn tích hợp sẵn:

- Mã hóa bằng thuật toán AES

- Quản lý khóa thông qua Keyring

Ví dụ:

```
mysqlbackup --backup \
--encrypt \
--keyring-file-data=/var/lib/mysql-keyring/keyring \
--target-dir=/backup/full
```

c) Đánh giá và vai trò bảo mật

Mã hóa file sao lưu mang lại các lợi ích bảo mật quan trọng:

- Bảo vệ dữ liệu khỏi rò rỉ khi backup bị đánh cắp
- Đáp ứng yêu cầu tuân thủ các chuẩn bảo mật (GDPR, PCI-DSS)
- Giảm thiểu rủi ro từ việc lưu trữ backup trên cloud hoặc thiết bị ngoài

Tuy nhiên, hiệu quả của mã hóa backup phụ thuộc rất lớn vào quản lý khóa. Nếu khóa mã hóa được lưu cùng vị trí với file backup, toàn bộ cơ chế bảo mật sẽ mất tác dụng.

2.3. Mã hóa dữ liệu truyền tải (Data-in-Transit Encryption)

Mã hóa dữ liệu trên đường truyền bảo vệ thông tin khi di chuyển giữa máy chủ MySQL và các ứng dụng client, ngăn chặn các cuộc tấn công nghe lén (eavesdropping) và tấn công trung gian (Man-in-the-Middle).

2.3.1. SSL/TLS trong MySQL

MySQL hỗ trợ mã hóa kết nối thông qua giao thức SSL/TLS (Secure Sockets Layer/Transport Layer Security):

- Cấu hình SSL trên MySQL Server:
  - ❖ Tạo chứng chỉ SSL:

```
# Tạo private key và certificate
openssl genrsa 2048 > ca-key.pem
openssl req -new -x509 -nodes -days 3650 -key ca-key.pem -out ca-cert.pem
```

❖ Cấu hình MySQL:

```
[mysqld]
ssl-ca=/path/to/ca-cert.pem
ssl-cert=/path/to/server-cert.pem
ssl-key=/path/to/server-key.pem
require_secure_transport = ON  # Bắt buộc kết nối SSL
```

❖ Kiểm tra trạng thái SSL:

```
Show Variables Like '%ssl%';
Show Status Like 'Ssl_cipher';
```

Thiết lập kết nối SSL từ Client:

- Sử dụng tham số --ssl-mode = REQUIRED trong mysql client
- Trong ứng dụng (ví dụ PHP, Python, Java, …), cấu hình kết nối SSL thông qua các thư viện tương ứng.

2.3.2. Mã hóa kết nối Replication

Trong kiến trúc MySQL Replication, dữ liệu được truyền liên tục từ máy chủ Master (Source) sang Slave (Replica) thông qua kết nối mạng. Nếu kết nối này không được mã hóa, kẻ tấn công có thể nghe lén hoặc can thiệp vào luồng dữ liệu replication, dẫn đến rò rỉ toàn bộ dữ liệu CSDL.

Do đó, việc mã hóa kết nối replication là yêu cầu bắt buộc đối với các hệ thống MySQL triển khai trong môi trường mạng không tin cậy (LAN dùng chung, WAN, Cloud).

a) Nguyên lý mã hóa Replication

- Replication sử dụng tài khoản riêng (replication user)
- Kết nối giữa Master và Slave có thể được bảo vệ bằng SSL/TLS
- Toàn bộ binary log events được mã hóa trong quá trình truyền

b) Cấu hình mã hóa Replication bằng SSL/TLS

Bước 1: Chuẩn bị chứng chỉ SSL

MySQL sử dụng các file chứng chỉ:

- CA certificate (ca.pem)
- Server certificate (server-cert.pem)
- Client certificate (client-cert.pem)

Bước 2: Cấu hình Master

Bước 3: Tạo user replication yêu cầu SSL

```
CREATE USER 'repl'@'%' IDENTIFIED BY 'replpass' REQUIRE SSL;
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
```

Bước 4: Cấu hình Slave kết nối Replication có SSL

c) Lợi ích bảo mật

- Ngăn chặn nghe lén dữ liệu replication
- Bảo vệ thông tin xác thực replication user
- Đảm bảo tính bí mật và toàn vẹn của dữ liệu đồng bộ

d) Hạn chế và lưu ý

- Có ảnh hưởng nhỏ đến hiệu năng do chi phí mã hóa
- Cần quản lý chứng chỉ SSL cẩn thận
- Nên kết hợp với firewall và giới hạn IP cho replication

2.4. Mã hóa ở cấp độ logic

2.4.1. Mã hóa cột (Column-level Encryption)

MySQL cung cấp các hàm mã hóa cấp cột cho phép mã hóa và giải mã dữ liệu theo từng trường:

Các hàm mã hóa tích hợp:

- AES_ENCRYPT() / AES_DECRYPT(): Sử dụng thuật toán AES
- DES_ENCRYPT() / DES_DECRYPT(): Sử dụng thuật toán
- MD5(), SHA1(), SHA2(): Các hàm hash (mã hóa một chiều)

Ưu điểm và hạn chế:

❖ Ưu điểm: Linh hoạt, chỉ mã hóa các trường nhạy cảm, hiệu suất tốt hơn so với mã hóa toàn bộ tablespace

❖ Hạn chế: Không thể sử dụng chỉ mục trên các cột đã mã hóa, tăng độ phức tạp truy vấn

2.4.2. Mã hóa ở tầng ứng dụng (Application-level Encryption)

Mã hóa ở tầng ứng dụng là phương pháp bảo mật dữ liệu nhạy cảm ngay tại lớp ứng dụng trước khi chúng được gửi đến Cơ sở dữ liệu. Trong mô hình này, ứng dụng đóng vai trò chủ động trong việc mã hóa và giải mã dữ liệu, thay vì dựa hoàn toàn vào các tính năng mã hóa của MySQL.

Đặc điểm và nguyên lý hoạt động:

- Dữ liệu được mã hóa ngay tại ứng dụng bằng các thư viện mã hóa như OpenSSL, Libsodium, hoặc các SDK hỗ trợ sẵn trong ngôn ngữ lập trình.
- Chỉ dữ liệu đã mã hóa (Ciphertext) mới được lưu trữ trong CSDL, thường dưới dạng chuỗi nhị phân (BLOB) hoặc chuỗi mã hóa Base64.
- Khóa mã hóa không lưu trong CSDL, mà được quản lý riêng biệt bởi ứng dụng hoặc hệ thống quản lý khóa bên ngoài (HSM, Key Vault).
- MySQL chỉ đóng vai trò lưu trữ, không tham gia vào quá trình mã hóa/giải mã.

Ưu điểm:

- Bảo mật cao nhất: Ngay cả quản trị viên CSDL (DBA) cũng không thể đọc được dữ liệu gốc nếu không có khóa.

- Linh hoạt: Ứng dụng có thể lựa chọn thuật toán mã hóa phù hợp (AES-GCM, ChaCha20-Poly1305, v.v.).
- Độc lập với phiên bản MySQL: Có thể áp dụng trên cả MySQL Community Edition.
- Bảo vệ dữ liệu ngay cả khi CSDL bị rò rỉ toàn bộ.

Nhược điểm:

- Tăng độ phức tạp phát triển: Ứng dụng phải xử lý mã hóa/giải mã, quản lý khóa, và xử lý lỗi liên quan.
- Không thể tìm kiếm hoặc lập chỉ mục trực tiếp trên dữ liệu đã mã hóa (trừ khi sử dụng kỹ thuật mã hóa có thể tìm kiếm được).
- Ảnh hưởng hiệu năng: Tăng tải xử lý cho ứng dụng, đặc biệt với lượng dữ liệu lớn.

Trường hợp áp dụng:

- Lưu trữ thông tin cực kỳ nhạy cảm: Số thẻ tín dụng, mật khẩu, khóa cá nhân.
- Ứng dụng đa nền tảng cần nhất quán về mã hóa giữa các client.

Hệ thống yêu cầu tuân thủ nghiêm ngặt (PCI-DSS, GDPR) với mô hình "zero-trust".

2.4.3. Mã hóa hàm băm (Hashing)

Hàm băm (Hash Function) là quá trình chuyển đổi dữ liệu đầu vào (input) thành một chuỗi ký tự có độ dài cố định (hash value) bằng thuật toán toán học một chiều (không thể đảo ngược). Khác với mã hóa đối xứng hay bất đối xứng, việc băm không nhằm mục đích phục hồi dữ liệu gốc.

Mục đích sử dụng trong bảo mật CSDL:

- Bảo vệ mật khẩu: Lưu trữ giá trị băm của mật khẩu thay vì mật khẩu gốc.
- Kiểm tra tính toàn vẹn dữ liệu: Phát hiện thay đổi trái phép.
- Xác thực tập tin hoặc bản ghi: Tạo chữ ký số đơn giản.

Các hàm băm phổ biến trong MySQL:

- MD5(): Tạo hash 128-bit (32 ký tự hex). Không nên dùng cho mật khẩu do dễ bị tấn công va chạm.
- SHA1(): Tạo hash 160-bit (40 ký tự hex). Cũng đã lỗi thời và không an toàn.

- SHA2(): Họ hàm an toàn, hỗ trợ độ dài 224, 256, 384, 512 bit. Được khuyến nghị sử dụng.

Ngoài ra còn có Cơ chế "muối" (Salting) để tăng cường bảo mật: Để chống lại tấn công rainbow table, cần thêm một giá trị ngẫu nhiên (salt) vào mật khẩu trước khi băm.

So sánh giữa Mã hóa và Băm:

Tiêu chí	Mã hóa (Encryption)	Băm (Hashing)
Mục đích	Bảo mật, có thể giải mã	Xác thực, toàn vẹn
Tính đảo ngược	Có (với khóa)	Không
Đầu ra	Ciphertext (có thể giải mã)	Hash value (cố định)
Ứng dụng	Bảo vệ dữ liệu nhạy cảm	Mật khẩu, chữ ký, checksum

Bảng 1: *Bảng so sánh giữa Mã hóa và Hàm Băm*

2.5. Quản lý khóa mã hóa trong MySQL

Quản lý khóa mã hóa (Key Management) là yếu tố then chốt quyết định hiệu quả của toàn bộ hệ thống mã hóa. Một khóa bị rò rỉ sẽ khiến dữ liệu mã hóa trở nên vô dụng.

2.5.1. MySQL Keyring

MySQL Keyring là một cơ chế lưu trữ và quản lý khóa mã hóa an toàn, được tích hợp trong MySQL Enterprise Edition. Keyring cho phép lưu trữ khóa bí mật, khóa mã hóa, và chứng chỉ một cách tập trung.

Các loại Keyring plugin hỗ trợ:

- keyring_file: Lưu khóa trong file trên đĩa (đơn giản nhưng kém an toàn).
- keyring_encrypted_file: Tương tự nhưng mã hóa file keyring bằng mật khẩu.
- keyring_okv: Tích hợp với Oracle Key Vault (doanh nghiệp lớn).
- keyring_aws: Tích hợp với AWS Key Management Service (KMS).
- keyring_hashicorp: Tích hợp với HashiCorp Vault.

Ưu điểm của Keyring:

- Tách biệt khóa và dữ liệu: Khóa không lưu chung với file dữ liệu.

- Hỗ trợ nhiều backend: Linh hoạt với các hệ thống quản lý khóa chuyên nghiệp.
- Tích hợp sâu với MySQL: Hỗ trợ TDE, mã hóa log, backup.

Hạn chế:

- Chỉ có trong MySQL Enterprise Edition.
- Cần cấu hình và bảo mật backend keyring cẩn thận.

2.5.2. Best Practices cho quản lý khóa

Nguyên tắc chung:

1. Tách biệt khóa và dữ liệu: Không lưu khóa trong cùng CSDL, ổ đĩa, hoặc server với dữ liệu mã hóa.
2. Luân chuyển khóa định kỳ (Key Rotation): Thay đổi khóa mã hóa theo chu kỳ (ví dụ: 90 ngày) để giảm thiểu rủi ro khóa bị xâm phạm.
3. Sử dụng hệ thống quản lý khóa chuyên nghiệp (HSM/KMS): AWS KMS, Azure Key Vault, Google Cloud KMS, hoặc HashiCorp Vault.
4. Áp dụng nguyên tắc tối thiểu đặc quyền (Least Privilege): Chỉ những người/vai trò cần thiết mới được truy cập khóa.

Quy trình quản lý khóa an toàn:

1. Tạo khóa (Key Generation):
   - Sử dụng bộ sinh số ngẫu nhiên an toàn (CSPRNG).
   - Độ dài khóa tối thiểu 256-bit cho AES.
2. Lưu trữ khóa (Key Storage):
   - Mã hóa khóa chính (MEK) bằng khóa chủ (KEK).
   - Lưu KEK trong HSM hoặc dịch vụ quản lý khóa.
3. Phân phối khóa (Key Distribution):
   - Sử dụng giao thức an toàn (TLS, SSH) khi truyền khóa.
   - Không gửi khóa qua email hoặc tin nhắn không mã hóa.
4. Sao lưu khóa (Key Backup):
   - Sao lưu khóa ra nhiều nơi an toàn.
   - Mã hóa bản sao lưu khóa.
5. Hủy khóa (Key Destruction):
   - Hủy khóa cũ an toàn khi không còn dùng.
   - Đảm bảo khóa bị xóa vật lý khỏi bộ nhớ.

Giám sát và kiểm toán:

- Ghi nhật ký truy cập khóa: Ai truy cập, khi nào, mục đích.
- Cảnh báo bất thường: Nhiều lần truy cập thất bại, truy cập ngoài giờ.
- Định kỳ kiểm toán: Đánh giá tính toàn vẹn của hệ thống quản lý khóa.

Khuyến nghị cho dự án nhỏ và vừa:

- Sử dụng keyring_encrypted_file với mật khẩu mạnh.
- Lưu file keyring trên ổ đĩa riêng biệt, được mã hóa toàn bộ (VeraCrypt, LUKS).
- Luân chuyển khóa ít nhất mỗi 6 tháng.
- Sao lưu file keyring và mật khẩu vào nơi an toàn (không cùng với backup CSDL).

2.6. So sánh các kỹ thuật mã hóa

Tiêu chí	Mã hóa Tablespace	Mã hóa cột	Mã hóa ứng dụng	SSL/TLS
Phạm vi	Toàn bộ tablespace	Từng cột cụ thể	Dữ liệu trước khi gửi đến DB	Kết nối mạng
Hiệu năng	Trung bình	Cao	Phụ thuộc	Trung bình
Độ phức tạp	Thấp (Auto)	Trung bình	Cao	Trung bình
Bảo vệ DBA	Không	Một phần	Cao	Không
Version	Enterprise 5.7.11+	Tất cả	Tất cả	Tất cả
Use Case	Tuân thủ PCI-DSS, GDPR	Dữ liệu nhạy cảm riêng lẻ	Bảo mật tối đa	Bảo vệ đường truyền

2.7. Kết luận

MySQL cung cấp một hệ sinh thái mã hóa đa dạng và linh hoạt, cho phép bảo vệ dữ liệu ở nhiều cấp độ khác nhau. Việc lựa chọn kỹ thuật mã hóa phù hợp phải dựa trên:

1. Yêu cầu bảo mật: Mức độ nhạy cảm của dữ liệu và các quy định phải tuân thủ

2. Hiệu năng hệ thống: Ảnh hưởng của mã hóa đến tốc độ xử lý

3. Khả năng quản lý: Độ phức tạp trong triển khai và vận hành

4. Chi phí: Sử dụng MySQL Enterprise Edition hay Community Edition

Chiến lược mã hóa hiệu quả thường kết hợp nhiều lớp bảo vệ:

- Sử dụng SSL/TLS cho kết nối mạng
- Triển khai mã hóa tablespace cho dữ liệu lưu trữ
- Áp dụng mã hóa cột cho các trường đặc biệt nhạy cảm
- Quản lý khóa tập trung và an toàn

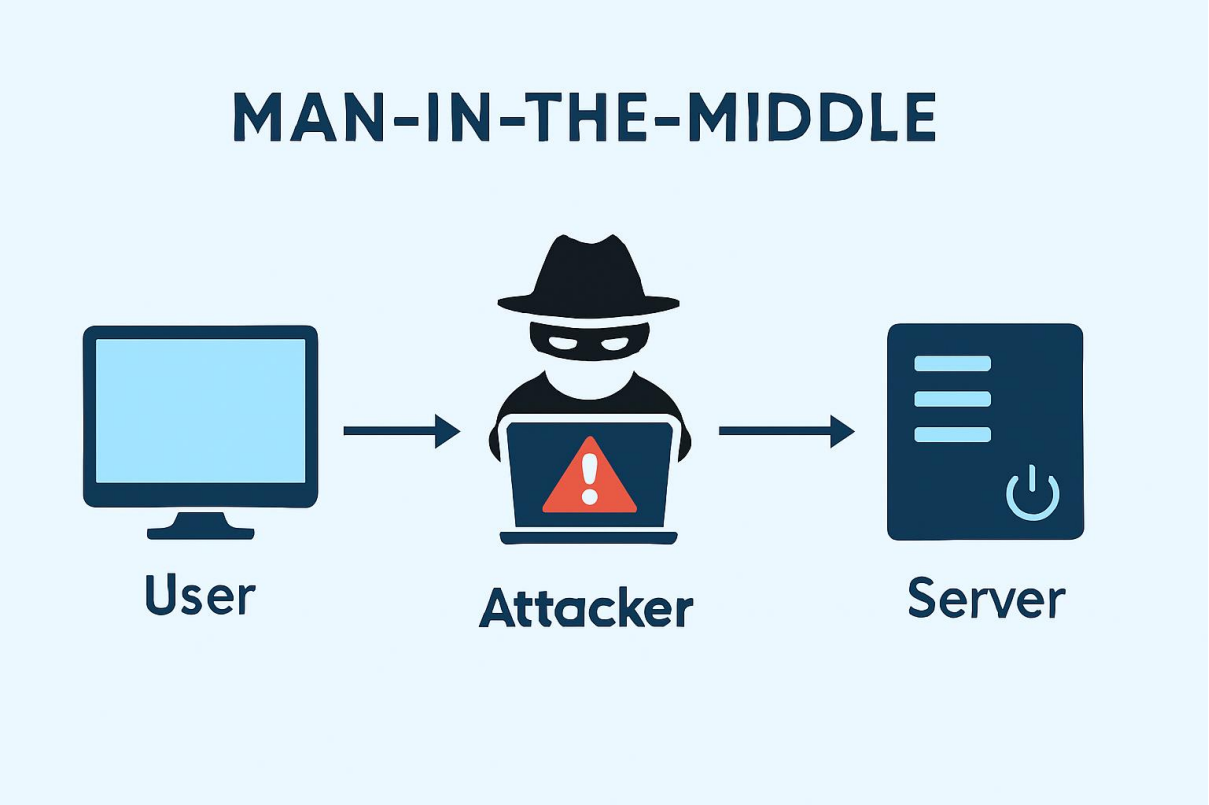
Trong chương tiếp theo, chúng ta sẽ tiến hành thực nghiệm đánh giá hiệu năng và độ bảo mật của các kỹ thuật mã hóa này trong các tình huống thực tế khác nhau.

Chương 3: Triển khai hệ thống

Chương 4: Kết luận và hướng phát triển

Tài liệu tham khảo

Phụ lục



Hình 1: Tấn công Man-in-the-Middle