BẢO MẬT DỮ LIỆU NHẠY CẢM TRONG QUÁ TRÌNH GHI LOG ỨNG DỤNG JAVA SPRING BOOT: NGHIÊN CỨU VÀ ỨNG DỤNG

Trần Xuân Trường[1]

[1] Viện Sư phạm Kỹ thuật, Trường Đại Học Bách Khoa Hà Nội

tran@travis.guru

TÓM TẮT— *Bài báo này nghiên cứu và triển khai các phương pháp để bảo mật dữ liệu nhạy cảm trong quá trình ghi log của ứng dụng Java sử dụng Spring Boot. Bằng cách áp dụng các công nghệ và phương pháp tiên tiến, bài báo đề xuất một giải pháp hiệu quả để ngăn chặn việc tiết lộ thông tin nhạy cảm trong quá trình ghi log. Các kết quả từ nghiên cứu này không chỉ giúp cải thiện bảo mật cho các ứng dụng Java sử dụng Spring Boot, mà còn đóng góp vào việc nâng cao nhận thức về vấn đề bảo mật trong quá trình phát triển phần mềm.*

Từ khóa— *software engineering, security, logging, sensitive data.*

I. GIỚI THIỆU (INTRODUCTION)

Trong thế giới số hóa hiện nay, việc bảo mật dữ liệu trở nên cực kỳ quan trọng. Các ứng dụng Java sử dụng Spring Boot được rộng rãi sử dụng trong nhiều lĩnh vực khác nhau. Tuy nhiên, trong quá trình ghi log, có thể có sự tiết lộ thông tin nhạy cảm, dẫn đến các vấn đề về bảo mật. Điều này đặt ra nhu cầu cấp thiết về việc nghiên cứu và triển khai các phương pháp bảo mật dữ liệu nhạy cảm trong quá trình ghi log. Mục tiêu của bài báo này là nghiên cứu và triển khai các phương pháp bảo mật dữ liệu nhạy cảm trong quá trình ghi log của các ứng dụng Java sử dụng Spring Boot.

II. CƠ SỞ LÝ THUYẾT (THEORETICAL BACKGROUND)

A. Java và Spring Boot

Java là một ngôn ngữ lập trình hướng đối tượng phổ biến, được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau. Spring Boot là một dự án con của Spring, giúp giảm thiểu yêu cầu cấu hình mã nguồn và tạo ra các ứng dụng Spring độc lập.

B. Ghi log trong ứng dụng

Ghi log là một phần quan trọng của việc phát triển và duy trì ứng dụng. Các bản ghi log cung cấp thông tin chi tiết về các hoạt động của ứng dụng, giúp phát hiện và khắc phục lỗi, tối ưu hóa hiệu suất và thực hiện phân tích.

C. Bảo mật dữ liệu nhạy cảm

Dữ liệu nhạy cảm như thông tin cá nhân, thông tin tài chính, dữ liệu y tế, và các loại thông tin khác nên được bảo vệ để ngăn chặn truy cập trái phép. Việc bảo mật dữ liệu nhạy cảm trong quá trình ghi log đòi hỏi các phương pháp và công nghệ đặc biệt.

D. Thiết kế và triển khai bảo mật trong ghi log

Trong quá trình thiết kế và triển khai bảo mật cho quá trình ghi log, cần xem xét các yếu tố như: loại dữ liệu nhạy cảm cần bảo vệ, môi trường hoạt động của ứng dụng, và các yêu cầu pháp lý hoặc quy định.

E. Các phương pháp bảo mật thông tin nhạy cảm trong ghi log

Có nhiều phương pháp có thể được sử dụng để bảo mật thông tin nhạy cảm trong quá trình ghi log, bao gồm: mã hóa dữ liệu, ẩn dữ liệu nhạy cảm, sử dụng các hệ thống quản lý truy cập, và các phương pháp khác.

F. Các loại log phổ biến

1. Log hệ thống (System Logs)

Những log này thường bao gồm thông tin về hoạt động của hệ thống, như thời gian khởi động, các sự kiện liên quan đến hệ thống, lỗi hệ thống, và cảnh báo.

2. Log ứng dụng (Application Logs)

Những log này cung cấp thông tin về hoạt động của ứng dụng, bao gồm các sự kiện ứng dụng, trạng thái hoạt động, lỗi, và cảnh báo. Thông tin này thường rất hữu ích trong việc phân tích hiệu suất ứng dụng và khắc phục lỗi.

3. Log giao dịch (Transaction Logs)

Những log này theo dõi các giao dịch diễn ra trong ứng dụng, như các yêu cầu và phản hồi HTTP, các giao dịch cơ sở dữ liệu, và các hoạt động liên quan đến thanh toán.

4. Log bảo mật (Security Logs)

Những log này theo dõi các sự kiện liên quan đến bảo mật, như đăng nhập và đăng xuất, các cố gắng truy cập không hợp lệ, và các hoạt động liên quan đến quản lý truy cập.

5. Log lỗi (Error Logs)

Những log này chứa thông tin chi tiết về các lỗi phát sinh trong ứng dụng. Thông tin này có thể bao gồm loại lỗi, thời gian phát sinh, và ngữ cảnh liên quan.

6. Log gỡ lỗi (Debug Logs)

Những log này chứa thông tin chi tiết giúp lập trình viên gỡ lỗi ứng dụng. Thông tin này thường chỉ được ghi khi ứng dụng đang chạy ở chế độ gỡ lỗi.

G. Các cách thức ghi log phổ biến trong ứng dụng Java Spring Boot

Trong Spring Boot, Logback là thư viện ghi log mặc định và SLF4J là API ghi log mặc định. Bạn có thể tùy chỉnh cấu hình ghi log thông qua file **application.properties** hoặc **application.yml**, hoặc thông qua code bằng cách sử dụng các annotation và interface của Spring và SLF4J.

1. Log Console

Đây là cách đơn giản nhất để ghi log, thông tin sẽ được in ra console (cửa sổ terminal hoặc command line) của môi trường chạy ứng dụng. Điều này rất hữu ích trong quá trình phát triển, nhưng không phải là lựa chọn tốt cho môi trường sản xuất vì khả năng lưu trữ và tìm kiếm thông tin kém.

2. Logback

Logback là thư viện ghi log mạnh mẽ và linh hoạt cho Java, được xem như là sự thay thế cho log4j. Logback có nhiều cấu hình tùy chỉnh và hỗ trợ nhiều định dạng xuất, bao gồm cả log console và log file. Điều này giúp bạn có thể lưu log theo từng mức độ (level) như ERROR, WARN, INFO, DEBUG, TRACE.

3. SLF4J (Simple Logging Facade for Java)

Đây là một API ghi log giúp tạo ra một lớp trừu tượng (facade) trên nhiều thư viện ghi log khác nhau (như Logback, log4j). Điều này cho phép bạn có thể thay đổi thư viện ghi log mà không cần thay đổi code của ứng dụng.

4. Log4j

Đây là một thư viện ghi log phổ biến cho Java, có nhiều cấu hình tùy chỉnh và hỗ trợ nhiều định dạng xuất. Tuy nhiên, Log4j thường được thay thế bằng Logback do Logback có hiệu suất và tính năng tốt hơn.

III. PHƯƠNG PHÁP NGHIÊN CỨU (METHODS)

Chúng tôi đã lựa chọn phương pháp "ẩn dữ liệu nhạy cảm" để bảo mật thông tin trong quá trình ghi log. Thực tế cho thấy, trong nhiều trường hợp, thông tin nhạy cảm không cần thiết cho việc phân tích log, vì vậy việc ẩn các thông tin này không ảnh hưởng đến việc khắc phục lỗi hay phân tích hiệu suất ứng dụng. Đồng thời, phương pháp này cũng giúp ngăn chặn việc tiết lộ thông tin nhạy cảm, giảm rủi ro bị tấn công và đảm bảo tuân thủ các quy định về bảo mật.

A. Ẩn dữ liệu nhạy cảm của log

Phần này tập trung vào việc sử dụng phương thức **maskSensitiveData** của lớp **PiBaseHelper**. Phương thức này có tác dụng tìm và che giấu thông tin nhạy cảm trong log.
Phương thức này hoạt động bằng cách sử dụng một loạt các biểu thức chính quy (được định nghĩa trong mảng **MASKING_PATTERNS**) để tìm kiếm các mẫu thông tin nhạy cảm trong log, như mật khẩu, mã OTP, mã PIN, token, khóa riêng tư, khóa công khai, và số tiền. Khi tìm thấy một mẫu khớp, phương thức sẽ thay thế thông tin nhạy cảm bằng các dấu "*".

B. Ẩn dữ liệu nhạy cảm khi ghi Log Console

Phần này tập trung vào việc sử dụng lớp **MaskingRequestLoggingFilter**. Lớp này kế thừa từ **CommonsRequestLoggingFilter** của Spring và được sử dụng để ghi log yêu cầu HTTP.
Trong lớp **MaskingRequestLoggingFilter**, phương thức **createMessage** được ghi đè để thêm tính năng che giấu thông tin nhạy cảm. Phương thức này sẽ tạo ra một thông điệp log từ thông tin yêu cầu HTTP. Nếu yêu cầu là một **ContentCachingRequestWrapper**, nó sẽ lấy nội dung yêu cầu dưới dạng chuỗi, sau đó sử dụng **PiBaseHelper.maskSensitiveData** để che giấu thông tin nhạy cảm trong nội dung yêu cầu. Thông điệp log cuối cùng sẽ chứa nội dung yêu cầu đã được che giấu thông tin nhạy cảm.

C. Ẩn dữ liệu nhạy cảm khi ghi Log sử dụng LogBack

Phần này tập trung vào việc sử dụng file cấu hình logback là **logback-spring.xml** và lớp **MaskingPatternLayout**.

File cấu hình **logback-spring.xml** định nghĩa cách thức ghi log vào file. Trong đó, **MaskingPatternLayout** được sử dụng như một encoder cho **FILE** appender.
Lớp **MaskingPatternLayout** kế thừa từ **PatternLayout** của Logback. Trong phương thức **writeLoopOnConverters**, nó gọi **PiBaseHelper.maskSensitiveData** để che giấu thông tin nhạy cảm trong log.
Nhờ cấu hình này, khi một sự kiện log được ghi vào file bởi **FILE** appender, thông tin nhạy cảm trong sự kiện log sẽ được che giấu bởi **PiBaseHelper.maskSensitiveData**, giúp bảo vệ thông tin nhạy cảm khỏi việc bị tiết lộ thông qua log.

IV. KẾT QUẢ (RESULTS)

Qua quá trình nghiên cứu và triển khai, phương pháp "ẩn dữ liệu nhạy cảm" trong quá trình ghi log đã đem lại những kết quả đáng kể:

- **Hiệu quả che giấu thông tin nhạy cảm**: Nhờ sử dụng phương thức **maskSensitiveData** từ lớp **PiBaseHelper**, chúng tôi đã thành công trong việc che giấu thông tin nhạy cảm như mật khẩu, mã OTP, mã PIN, token, khóa riêng tư, khóa công khai và số tiền trong các log. Những thông tin này đã được thay thế bằng các dấu "*", giúp ngăn chặn việc tiết lộ thông tin nhạy cảm qua log.
- **Ứng dụng trong ghi Log Console và LogBack**: Lớp **MaskingRequestLoggingFilter** và lớp **MaskingPatternLayout** đã được thiết kế để ứng dụng phương thức **maskSensitiveData** trong quá trình ghi Log Console và LogBack. Điều này giúp chúng tôi áp dụng phương pháp "ẩn dữ liệu nhạy cảm" một cách linh hoạt và hiệu quả.
- **Tuân thủ các quy định về bảo mật**: Việc ẩn dữ liệu nhạy cảm trong log giúp ứng dụng của chúng tôi tuân thủ tốt hơn với các quy định về bảo mật, đồng thời giảm thiểu rủi ro liên quan đến vi phạm bảo mật dữ liệu.
- **Không ảnh hưởng đến việc phân tích log**: Như đã nêu trong phần [Phương pháp nghiên cứu (Methods)], việc ẩn dữ liệu nhạy cảm không ảnh hưởng đến việc phân tích log, vì thông tin nhạy cảm thường không cần thiết cho việc phân tích hiệu suất ứng dụng hay khắc phục lỗi.

Tổng kết lại, qua quá trình nghiên cứu và triển khai, phương pháp "ẩn dữ liệu nhạy cảm" đã cho thấy hiệu quả đáng kể trong việc bảo vệ thông tin nhạy cảm trong quá trình ghi log của các ứng dụng Java Spring Boot.

V. THẢO LUẬN (DISCUSSION)

Việc che giấu thông tin nhạy cảm trong quá trình ghi log đã trở thành một phần quan trọng trong việc bảo mật ứng dụng. Tuy nhiên, không phải mọi loại thông tin nhạy cảm đều có thể được nhận biết và che giấu một cách tự động. Trong phạm vi của nghiên cứu này, chúng tôi đã tập trung vào một số loại thông tin nhạy cảm phổ biến như mật khẩu, mã OTP, mã PIN, token, khóa riêng tư, khóa công khai và số tiền. Tuy nhiên, có thể có những loại thông tin nhạy cảm khác mà phương pháp này chưa đề cập đến. Bạn đọc khi áp dụng có thể mở rộng thêm các thông tin nhạy cảm cần bảo vệ.
Ngoài ra, cần lưu ý rằng, việc che giấu thông tin nhạy cảm trong log chỉ là một phần nhỏ trong việc bảo mật ứng dụng. Bên cạnh đó, việc bảo mật dữ liệu nhạy cảm trong quá trình truyền tải, lưu trữ, và xử lý cũng cần được chú trọng. Đối với các ứng dụng Java Spring Boot, có nhiều cách tiếp cận khác như sử dụng HTTPS, mã hóa dữ liệu, sử dụng JWT (JSON Web Token) để bảo vệ dữ liệu nhạy cảm.

VI. KẾT LUẬN (CONCLUSION)

Qua quá trình nghiên cứu và triển khai, chúng tôi nhận thấy rằng việc ẩn dữ liệu nhạy cảm trong quá trình ghi log là một phương pháp hiệu quả để bảo vệ thông tin nhạy cảm khỏi việc bị tiết lộ. Đặc biệt, trong môi trường sản xuất, việc này không chỉ giúp cải thiện bảo mật ứng dụng mà còn giúp tuân thủ các quy định về bảo mật dữ liệu.
Tuy nhiên, như đã thảo luận ở trên, việc này chỉ là một phần của việc bảo mật ứng dụng và cần được kết hợp với các biện pháp bảo mật khác để tạo nên một hệ thống bảo mật toàn diện.
Chúng tôi hi vọng rằng, với nghiên cứu này, chúng tôi đã góp phần vào việc cải thiện bảo mật trong quá trình ghi log của các ứng dụng Java Spring Boot và đề xuất một hướng đi mới trong việc xử lý thông tin nhạy cảm trong ứng dụng.

VII. TÀI LIỆU THAM KHẢO (REFERENCES)

[1] Spring Boot, "Spring Boot Documentation", Spring Boot, 2023.
[2] Logback, "Logback Manual", QOS.ch, 2023.
[3] P.J. Deitel, H.M. Deitel, "Java How to Program, Early Objects", Pearson, 12th Edition, 2019.
[4] C. Walls, R. Breidenbach, "Spring in Action", Manning Publications, 2011.

VIII. PHỤ LỤC (APPENDIX)

Để giúp người đọc hiểu rõ hơn về cách triển khai bảo mật dữ liệu nhạy cảm trong quá trình ghi log trong thực tế, chúng tôi đã đưa toàn bộ mã nguồn của ứng dụng vào kho lưu trữ công cộng trên GitHub. Mã nguồn này bao gồm chi tiết cụ thể của các tệp đã được đề cập trong mục [Phương pháp nghiên cứu (Methods)].
Dưới đây là các liên kết đến mã nguồn chi tiết:

A. Ẩn dữ liệu nhạy cảm của log

[1] PiBaseHelper.java
https://github.com/travistran1989/securing-sensitive-data-in-logging-process-of-java-spring-boot-applications/blob/f3d26abb7aa6d36424cb545bd890dacee2f25c1c/src/PiBaseHelper.java

B. Ẩn dữ liệu nhạy cảm khi ghi Log Console

[2] MaskingRequestLoggingFilter.java
https://github.com/travistran1989/securing-sensitive-data-in-logging-process-of-java-spring-boot-applications/blob/f3d26abb7aa6d36424cb545bd890dacee2f25c1c/src/MaskingRequestLoggingFilter.java

C. Ẩn dữ liệu nhạy cảm khi ghi Log sử dụng LogBack

[3] logback-spring.xml
https://github.com/travistran1989/securing-sensitive-data-in-logging-process-of-java-spring-boot-applications/blob/f3d26abb7aa6d36424cb545bd890dacee2f25c1c/src/logback-spring.xml
[4] MaskingPatternLayout.java
https://github.com/travistran1989/securing-sensitive-data-in-logging-process-of-java-spring-boot-applications/blob/f3d26abb7aa6d36424cb545bd890dacee2f25c1c/src/MaskingPatternLayout.java

SECURING SENSITIVE DATA IN LOGGING PROCESS OF JAVA SPRING BOOT APPLICATIONS: A STUDY AND IMPLEMENTATION

Tran Xuan Truong

ABSTRACT*— This paper investigates and implements methods to secure sensitive data during the logging process of Java applications using Spring Boot. By employing advanced technologies and methods, the paper proposes an effective solution to prevent the disclosure of sensitive information during logging. The results from this study not only help to improve the security of Java applications using Spring Boot, but also contribute to raising awareness about security issues in software development.*