**6.** **Một cách tiếp cận để kiểm tra bảo mật của phần mềm ứng dụng cơ sở dữ liệu được đề xuất**

SINIŠA S. ILIĆ, LJUBOMIR LAZIĆ, PETAR SPALEVIĆ

Khoa Khoa học Máy tính

Đại học Priština

Knjaza Miloša 7, 38220 Kosovska Mitrovica

SERBIA

sinisa.ilic@pr.ac.rs, http://www.ftn.pr.ac.rs

*Trừu tượng:-*Bài viết này trình bày khái niệm về cấu hình và phát triển cơ sở dữ liệu xem xét các vấn đề bảo mật, đặc biệt là khi kết nối với internet. Bất kể các biện pháp phòng ngừa về khả năng bảo mật được thực hiện trên các cấp độ khác của môi trường cơ sở dữ liệu, chẳng hạn như: mạng, hệ điều hành, ứng dụng khách, điều quan trọng là phải bảo vệ chính cơ sở dữ liệu bằng cách tránh các vấn đề bảo mật cơ sở dữ liệu nổi tiếng. Để chứng minh rằng cấu hình được đề xuất có mức độ bảo vệ bảo mật cao, kiểm tra bảo mật phải được thực hiện. Mục tiêu tổng thể của kiểm thử bảo mật là giảm các lỗ hổng trong hệ thống phần mềm và chúng tôi đã đề xuất phương pháp kiểm thử bao gồm xem xét mã và đánh giá lỗ hổng đại diện cho các phương pháp hay nhất phổ biến nhất để đảm bảo bảo mật phần mềm.

*Từ khóa: -*cấu hình cơ sở dữ liệu, lỗ hổng bảo mật, kiểm tra bảo mật

Contents

[**1Giới thiệu** 1](#_Toc156136814)

[**2Công việc liên quan** 2](#_Toc156136815)

[**3Xây dựng vấn đề** 4](#_Toc156136816)

[**3.1 Tấn công SQL Injection** 4](#_Toc156136817)

[**4Đề xuất giải pháp vấn đề** 5](#_Toc156136818)

[**5Kết thúc** 7](#_Toc156136819)

[**Lời cảm ơn** 8](#_Toc156136820)

**1Giới thiệu**

Để cung cấp các hệ thống phần mềm đáng tin cậy về mặt chức năng có chất lượng cao hơn trong thời gian và ngân sách, các lỗ hổng tiềm ẩn trong bảo mật hầu hết bị bỏ qua. Các nhà phát triển phần mềm cố gắng thiết kế cơ sở dữ liệu và ứng dụng theo các thông số kỹ thuật chức năng mà không xem xét bảo mật, bởi vì mục tiêu chính trong phát triển phần mềm là đáp ứng các yêu cầu chức năng. Vì bảo mật là một yêu cầu phi chức năng, nó không phải là mối quan tâm hàng đầu đối với các nhà phát triển hệ thống. Trong quá trình phát triển phần mềm, không có "công thức" làm thế nào để xử lý các vấn đề tiềm ẩn liên quan đến bảo mật. Thường thì rất khó để thấy trước chúng. Vì việc lập trình các quy trình xử lý các rủi ro bảo mật thường xuyên nhất rất tốn thời gian so với thời gian cần thiết để lập trình các yêu cầu chức năng, nên rõ ràng tại sao thiếu bảo mật xảy ra.

Lỗi bảo mật cũng phụ thuộc vào kiến trúc của việc triển khai phần mềm. Nếu phần mềm đáng tin cậy trong một số môi trường, điều tương tự có thể không đáng tin cậy trong môi trường khác. Đó là lý do tại sao rất nhiều mối đe dọa bảo mật phải được xử lý trong phát triển phần mềm ngay cả khi đôi khi nó có vẻ không cần thiết. Trong môi trường đa cấp, người ta có thể tin tưởng vào khả năng cao kẻ tấn công sẽ không vượt qua các rào cản bảo mật của các cấp độ giữa kẻ tấn công và hệ thống phần mềm. Nhưng nó cũng có thể xảy ra rằng lỗ hổng bảo mật của một cấp độ dẫn đến nguy hiểm bảo mật của các cấp độ khác.

Thông thường, việc xác thực dữ liệu đầu vào phải được thực hiện cả hai: trong ứng dụng khách (bằng cách sử dụng các tập lệnh hoặc hàm xác thực) và trong cơ sở dữ liệu bằng cách sử dụng các ràng buộc, trình kích hoạt hoặc quy trình được lưu trữ. Bằng cách kiểm tra dữ liệu ở cả hai cấp, khả năng dữ liệu có giá trị sai sẽ được chèn vào cơ sở dữ liệu là tối thiểu.

Thiết kế một cơ sở dữ liệu sẽ đạt được tất cả các yêu cầu bảo mật là rất khó, vì một hệ thống cơ sở dữ liệu xử lý một lượng lớn dữ liệu theo những cách phức tạp. Kết quả là hầu hết các hệ thống cơ sở dữ liệu thông thường đều có rò rỉ mà kẻ tấn công có thể sử dụng để xâm nhập cơ sở dữ liệu.

Trước khi bất kỳ hệ thống phần mềm nào sẽ được triển khai, kiểm thử phải được thực hiện. Thật không may, hầu hết các trường hợp trong các kịch bản thử nghiệm của hệ thống cơ sở dữ liệu kiểm tra xem các yêu cầu chức năng có được đáp ứng hay không, các bài kiểm tra bảo mật được thực hiện kém. Nó gần như là thực tế rằng thử nghiệm nói chung được thực hiện dưới áp lực thời gian rất lớn. Thông thường, những người kiểm tra lịch trình chặt chẽ chỉ bị hạn chế để kiểm tra chức năng phần mềm và thiếu các khía cạnh chất lượng khác. Mặt khác, kiểm thử chức năng xác nhận xem hệ thống có hoạt động như dự định hay không và kiểm tra bảo mật nhằm mục đích phát hiện hành vi không xác định trong hệ thống. Nhiệm vụ kiểm thử bảo mật là rất khó khăn khi xem xét rằng các mối đe dọa bảo mật đang phát triển nhanh chóng và rất khó để mô phỏng các điều kiện này.

May mắn thay, có rất nhiều công cụ miễn phí có thể được sử dụng để kiểm tra các lỗ hổng bảo mật. Một số trong số chúng là một phần của HĐH (như ssh, netcat, wireshark, v.v.); những người khác là một phần của trình duyệt web (Firefox, tiện ích bổ sung Firebug, tiện ích bổ sung View Source Chart, tiện ích bổ sung Tamper Data, v.v.) và các phần khác. Ngoài ra còn có một số công cụ miễn phí (HP Scrawlr, SQLiX) cũng như thương mại (HP WebInspect, IBM Rational AppScan) có thể tự động kiểm tra các lỗi bảo mật.

Kiểm tra bảo mật trong các ứng dụng này đặt ra những thách thức nghiêm trọng cho các kỹ sư.

**2Công việc liên quan**

Bảo mật là một quá trình duy trì mức độ rủi ro chấp nhận được. Vì vậy, bảo mật là một quá trình, không phải là trạng thái cuối cùng, tức là bảo mật không phải là sản phẩm cuối cùng.

Khi hệ thống bảo mật dữ liệu được xem xét, nó đòi hỏi một cách tiếp cận có hệ thống phải bao gồm: nhân viên, bảo mật vật lý và kỹ thuật, bảo mật thủ tục kinh doanh, an ninh mạng (LAN và / hoặc WAN), trách nhiệm và xử phạt [1]. Đây là một cách tiếp cận rất chung về bảo mật, vì hầu hết mọi người đều xử lý các vấn đề bảo mật liên quan đến cấu hình tường lửa. Tuy nhiên, để bảo vệ dữ liệu được thu thập, lưu trữ và chuyển giao khỏi: mối nguy vật lý, lỗi phần cứng, lỗi hoạt động, lỗi phần mềm, trộm cắp phương tiện và các thao tác khác nhau, mọi công ty đàng hoàng phải có chính sách bảo mật dữ liệu. Công việc của các nhà quản lý chịu trách nhiệm về an ninh ở tất cả các cấp, là phân tích: rủi ro tiềm ẩn, ảnh hưởng của chúng, tác động nếu rủi ro xảy ra, xác suất xuất hiện rủi ro, phương pháp xử lý cần thực hiện để đảm bảo rủi ro sẽ không xảy ra và ai chịu trách nhiệm thực hiện các biện pháp thích hợp.

Để đạt được các hệ thống an toàn được đánh giá cao, một thử nghiệm thâm nhập hoàn hảo phải được thực hiện [2]. Kiểm thử thâm nhập không thể chứng minh hoặc thậm chí chứng minh rằng một hệ thống là hoàn hảo. Nó có thể đặt một ràng buộc hợp lý về kiến thức và yếu tố công việc cần thiết để một người thâm nhập thành công. Không nên thực hiện loại thử nghiệm này trên các hệ thống sản xuất vì có thể tạm dừng, đổ hoặc treo hệ thống. Thật không may, kết quả của các thử nghiệm thâm nhập cho thấy hầu hết các hệ thống cơ sở dữ liệu thông thường đều có rò rỉ mà kẻ tấn công có thể sử dụng để xâm nhập cơ sở dữ liệu.

Có một số cách được đề xuất trong các giấy tờ để xử lý rủi ro bảo mật. Chúng ta có thể nhóm chúng thành các lớp sau: Kỹ thuật phát hiện xâm nhập, kỹ thuật bảo vệ cơ sở dữ liệu và tự động hóa kiểm tra bảo mật.

Sự xâm nhập có thể được định nghĩa là "bất kỳ tập hợp các hành động nào cố gắng thỏa hiệp tính toàn vẹn, tính bảo mật hoặc tính khả dụng của tài nguyên. Trong quá trình phát hiện xâm nhập, một số hoạt động có thể được coi là đáng ngờ theo các tiêu chí được xác định trước (hành vi độc hại đi chệch khỏi các mẫu bình thường đã thiết lập). Vấn đề với hiện đại hiện nay là giảm tỷ lệ âm tính giả và dương tính giả. Đồng thời, cần xem xét một hệ thống phát hiện xâm nhập theo thời gian thực. Thật khó để đạt được cả hai.

Các hoạt động độc hại có thể được phát hiện bởi Support Vector Machines (SVM) - một trong những thuật toán phân loại thành công nhất trong khu vực khai thác dữ liệu [3]. Giới hạn của việc sử dụng SVM là thời gian đào tạo dài và các tác giả đã trình bày một nghiên cứu để tăng cường thời gian đào tạo của SVM, đặc biệt khi xử lý các tập dữ liệu lớn, sử dụng phân tích phân cụm phân cấp.

Vì "các hoạt động độc hại" không cần phải đến từ kẻ tấn công, nên phải đảm bảo đủ thời gian để (các) kỹ sư bảo mật điều tra xem hoạt động có thực sự nguy hiểm hay không. Trong thời gian đó, kẻ tấn công tiềm năng phải được thuyết phục rằng giao dịch của mình thành công. Phương pháp để giải quyết vấn đề này là xây dựng càng nhiều "bản sao" cơ sở dữ liệu càng nhiều người dùng đáng ngờ được kết nối với cơ sở dữ liệu [4]. Nếu một người dùng "đáng ngờ" được chứng minh là kẻ tấn công, nó sẽ bị chặn và nếu không, thuật toán hợp nhất được xây dựng để thay thế phiên bản (giá trị) đáng tin cậy của nó bằng phiên bản đáng ngờ (giá trị), sau đó xóa phiên bản đáng ngờ.

Nhiều hệ thống phần mềm được thiết kế dựa trên Web và có sẵn cho công chúng thông qua Internet. Bằng cách này, họ trở nên tiếp xúc với một loạt các cuộc tấn công dựa trên Web. Có tới 78% các lỗ hổng được báo cáo gần đây ảnh hưởng đến các ứng dụng Web [5]. Hai loại tấn công đặc biệt phổ biến và gây sát thương. Trong SQL injection, kẻ tấn công thực hiện các câu lệnh cơ sở dữ liệu độc hại bằng cách khai thác xác nhận không đầy đủ của dữ liệu chảy từ người dùng đến cơ sở dữ liệu. Trong cross-site scripting, kẻ tấn công thực thi mã độc trên máy của nạn nhân bằng cách khai thác xác thực không đầy đủ của dữ liệu chảy đến các câu lệnh xuất ra HTML.

Để bảo vệ tính toàn vẹn và bảo mật dữ liệu, các kỹ thuật xác thực khác nhau được đề xuất. Mỗi mục dữ liệu mà người dùng gửi từ máy khách đến máy chủ đều được kiểm tra và lọc trước khi đến cơ sở dữ liệu. Quá trình này được gọi là vệ sinh.

Dữ liệu đầu vào có thể được phân tích cú pháp thành các phân đoạn theo các từ khóa của cú pháp SQL và so sánh với cấu trúc dự kiến [6]. Nếu cấu trúc dự kiến khác với cấu trúc thực tế (phân tích cú pháp từ trường đầu vào), dữ liệu đầu vào không được chuyển đến cơ sở dữ liệu.

Một cách tiếp cận khác là lọc dữ liệu đầu vào bằng cách sử dụng công cụ tìm kiếm biểu thức chính quy [7]. Công cụ tìm kiếm biểu thức chính quy cho phép tìm các mẫu trong chuỗi văn bản (bằng cách: từ khóa, văn bản trong ngoặc, thẻ, mẫu số, v.v.). Các từ khóa của câu lệnh SQL, tautologies, mã hóa URL và các ký tự mã hóa khác có thể được đặt tại các chuỗi tìm kiếm biểu thức chính quy. Các ký tự / từ độc hại này có thể bị xóa khỏi các trường nhập liệu và không được chuyển sang cơ sở dữ liệu. Cách tiếp cận tương tự có thể đạt được với hàm java HashMap có thể được sử dụng để chuyển đổi các ký tự được mã hóa (kẻ tấn công có thể sử dụng thay vì các ký tự bình thường để bỏ qua một số kỹ thuật xác thực tiêu chuẩn) thành các ký tự tiêu chuẩn trong thời gian tối thiểu [8].

Như đã nêu trong phần giới thiệu, tính bảo mật của các hệ thống cơ sở dữ liệu thường phụ thuộc vào tính bảo mật của các lớp khác giữa kẻ tấn công và cơ sở dữ liệu. Bảo mật tốt hơn được áp dụng cho chính cơ sở dữ liệu thì càng ít có khả năng làm tăng nguy cơ xâm nhập vào cơ sở dữ liệu.

Dữ liệu trong cơ sở dữ liệu có thể được mã hóa và ngay cả trong trường hợp xâm nhập, kẻ tấn công không thể sử dụng dữ liệu đó [9]. Dữ liệu có thể được chia thành công khai, phân loại và riêng tư. Dữ liệu được phân loại có thể được truy cập bởi người dùng được ủy quyền (ví dụ: kế toán viên có thể xem dữ liệu bảng lương cho tất cả nhân viên) và dữ liệu riêng tư chỉ có thể được truy cập bởi người dùng đã chèn nó (mỗi người dùng có dữ liệu riêng tư). Dữ liệu được bảo vệ bằng các khóa đối xứng (do hệ thống tạo ra) và được phân phối trong các phong bì được chứng nhận bằng cách sử dụng các cặp khóa công khai / riêng tư cho người dùng. Nhiều kỹ thuật mã hóa thay thế cũng có thể được sử dụng bằng cách xem xét tính bảo mật và hiệu suất [10].

Việc bảo vệ cơ sở dữ liệu cũng có thể được nhúng trong cơ sở dữ liệu. Người ta có thể sử dụng các chính sách nhúng vào chính cơ sở dữ liệu và cho phép các chính sách này chặn mọi nỗ lực thỏa hiệp trạng thái của cơ sở dữ liệu hoặc thay đổi cấu hình của nó theo cách mâu thuẫn với những gì đã được chủ sở hữu hệ thống thiết lập và đưa vào chính sách [11]. Nó đạt được bằng cách sử dụng quyền truy cập của người dùng vào cơ sở dữ liệu và bằng cách sử dụng các quy trình được lưu trữ ngay cả để thay đổi các tham số của cơ sở dữ liệu bởi người quản trị cơ sở dữ liệu (không phải chủ sở hữu cơ sở dữ liệu). Khi người dùng thành thạo hoặc tin tặc bắt đầu nỗ lực thay đổi cấu hình bảo mật (tham số cơ sở dữ liệu), yêu cầu sẽ trải qua quá trình xác minh trước khi có thể được xử lý. Bước này được thực hiện bởi các thủ tục lưu trữ cơ sở dữ liệu có logic tích hợp để kiểm tra yêu cầu đối với các chính sách. Nếu yêu cầu tuân thủ các chính sách đã đặt chi phối phạm vi áp dụng của nó, thì yêu cầu sẽ được áp dụng. Sau đó, các bảng/dạng xem hệ thống cơ sở dữ liệu được cập nhật để phản ánh sự thay đổi và biên bản kiểm tra được ghi lại. Nếu không, yêu cầu sẽ bị từ chối và chủ sở hữu hệ thống được cảnh báo, người dùng được thông báo và dấu vết kiểm tra được ghi lại.

Bằng cách sử dụng các mô hình MAC (Kiểm soát truy cập bắt buộc), RBAC (Kiểm soát truy cập dựa trên vai trò) và DAC (Kiểm soát truy cập tùy ý) đã được sửa đổi, có thể thiết kế một hệ thống bảo mật cơ sở dữ liệu có thể kiểm soát riêng quyền truy cập của người dùng vào các nhóm dữ liệu có kích thước khác nhau và phù hợp với tình huống đặc quyền truy cập của người dùng đối với dữ liệu tùy ý được thay đổi thường xuyên [12]. Trong các mô hình này, người dùng có thể truy cập bất kỳ dữ liệu nào có mức độ bảo mật thấp hơn hoặc bằng nhau và có thể truy cập được bởi các vai trò mà người dùng được chỉ định.

Nó không đủ để xây dựng các mô-đun để phát hiện sự xâm nhập và bảo vệ cơ sở dữ liệu. Các mô hình được xây dựng nên được kiểm tra về rủi ro bảo mật. Thường thì rất tốn thời gian để xây dựng các kịch bản thử nghiệm và thực hiện chúng.

Có hai cách tiếp cận trong kiểm thử các lỗ hổng bảo mật: bằng cách phân tích mã nguồn tĩnh và động và bằng cách xử lý hệ thống được kiểm tra như hộp đen.

Mã nguồn của một ứng dụng có thể được phân tích theo cách để tìm một tập dữ liệu mà việc thực thi chương trình có thể đến một nút cụ thể, được gọi là nút đích trong Biểu đồ luồng điều khiển (CFG), đại diện cho mục tiêu của phân tích thử nghiệm. Đây là cách tiếp cận chuỗi [13]. Cách tiếp cận chuỗi bảo mật sẽ dẫn đến việc tạo ra chỉ những chuỗi sự kiện cần được thực thi để phát hiện các lỗ hổng bảo mật. Nếu cách tiếp cận không thể tìm ra giải pháp để đi qua cây, thì cây này được coi là không thể, và do đó không có lỗ hổng nào được phát hiện hoặc báo cáo, loại bỏ tất cả các kết quả dương tính giả. Bằng cách biết mã nguồn, cây được xây dựng và một tập hợp dữ liệu được tạo tự động để tìm các nút dễ bị tấn công.

Công cụ kiểm tra tự động để kiểm tra lỗ hổng SQL injection – Ardilla [14] theo dõi luồng dữ liệu bị nhiễm độc được tạo ra thông qua cơ sở dữ liệu. Khi dữ liệu bị nhiễm độc được lưu trữ trong cơ sở dữ liệu, thông tin bị nhiễm độc được lưu trữ cùng với nó. Khi dữ liệu sau đó được truy xuất từ cơ sở dữ liệu, nó được đánh dấu bằng vết bẩn được lưu trữ. Độ chính xác này giúp Ardilla có thể phát hiện chính xác các cuộc tấn công Cross-Site Scripting bậc hai. Ardilla cũng sử dụng các mẫu tấn công SQL Injection được phát triển bởi các chuyên gia bảo mật và phát hiện các cuộc tấn công bằng cách tìm kiếm sự khác biệt trong cách chương trình hoạt động khi chạy trên hai đầu vào: một vô hại và một có khả năng độc hại.

**3Xây dựng vấn đề**

Theo các bài báo được liệt kê, có rất nhiều lỗ hổng bảo mật có thể gây hại cho việc chạy các hệ thống cơ sở dữ liệu. Kẻ tấn công sẽ cố gắng để có được quyền quản trị trên cơ sở dữ liệu. Mục tiêu của chúng tôi là cố gắng bảo vệ cơ sở dữ liệu nhiều nhất có thể bằng cách tự bảo vệ ở cấp cơ sở dữ liệu.

Nếu giả định rằng kẻ tấn công sẽ cố gắng thực thi mã độc từ việc ghi nhật ký ứng dụng khách với tư cách khách, anh ta / cô ta có thể sẽ cố gắng thực hiện các cuộc tấn công sau.

**3.1 Tấn công SQL Injection**

Lỗ hổng SQL Injection là kết quả của việc ứng dụng sử dụng đầu vào của người dùng trong việc xây dựng các câu lệnh cơ sở dữ liệu. Nếu đầu vào của người dùng sẽ được gán cho các biến *input1*và*input2* và nếu nội dung của biến đó sẽ được sử dụng để tự động tạo *truy vấn* câu lệnh SQL như:

*truy vấn* = "CHỌN UserId TỪ NGƯỜI DÙNG Ở ĐÂU

uname ='" +input1+"' và passw= '"*+input2+*"'"

Rõ ràng là nếu input1=admin, và *input2*=trust, *truy vấn* sẽ là:

*truy vấn* = "CHỌN UserId FROM Người dùng WHERE uname = 'admin' và passw = 'trust'"

và truy vấn sẽ được gửi đến cơ sở dữ liệu để thực thi. Nếu người dùng được phép đặt bất cứ thứ gì vào trường biểu mẫu web thích hợp (đầu vào biến của chúng tôi), họ có thể thử đặt như sau [15]:

1) *input1*=anyuser, *input2*=pass' hoặc '1'='1

2) *input1*= anyuser' hoặc 1=1 LIMIT 1;#, *input2*=pass

3) *input1* = anyuser, *input2* = pass 'VÀ 1 = 0

CÔNG ĐOÀN CHỌN

(case+when+(USER()='root@localhost')+th en+ 2+else+1+end) VÀ '1' = '1

Trong trường hợp đầu tiên, *truy vấn* sẽ bao gồm lệnh SQL sẽ trả về UserId của tất cả người dùng cho kẻ tấn công (vì tautology '1' = '1'). Trong trường hợp thứ hai, lệnh SQL sẽ trả về người dùng đầu tiên (hầu như luôn luôn là quản trị viên) vì từ khóa LIMIT sẽ lọc người dùng đầu tiên và # (dấu băm) sẽ nhận xét tất cả các ký tự còn lại trong chuỗi *truy vấn*. Trong trường hợp thứ ba, lệnh SQL sẽ bao gồm hai lệnh phụ, lệnh đầu tiên sẽ trả về không có gì (vì 1 = 0 là sai) và lệnh thứ hai (sau các từ khóa UNION ALL) sẽ trả về 2 nếu người dùng cơ sở dữ liệu root@localhost và 1 nếu người dùng cơ sở dữ liệu không root@localhost (trường hợp cụ thể được lấy cho MySQL Server, nhưng nó rất giống với các DBMS khác).

Người ta có thể tự kết luận về việc tạo các lệnh SQL tương tự bằng cách sử dụng các giá trị khác nhau cho biến *đầu vào*, đặc biệt là bằng cách biết rằng nhiều DBMS cung cấp cho người dùng các chức năng đọc dữ liệu từ các tệp trong HĐH và ghi dữ liệu vào tệp.

**3.2 Tấn công Cross-Site Scripting (XSS)**Đây là một loại lỗ hổng bảo mật được tìm thấy trong các ứng dụng web cho phép kẻ tấn công tiêm tập lệnh phía máy khách vào các trang web được người dùng khác xem. Giả sử rằng người dùng có thể gửi đến cơ sở dữ liệu một số đầu vào ở dạng văn bản. Nếu văn bản đã gửi được lưu trong cơ sở dữ liệu, người dùng khác có quyền xem văn bản đó có thể xem được.

Kẻ tấn công có thể sử dụng cơ hội để đưa mã web độc hại vào trường đó và gửi nó vào cơ sở dữ liệu. Khi người dùng khác mở trang web với nội dung đó, tập lệnh sẽ được thực thi tại trình duyệt của người dùng và có khả năng gây hại cho máy tính của người dùng hoặc xâm phạm dữ liệu cá nhân của một số người dùng.

**3.3 Nghe lén và đánh cắp mật khẩu**Đôi khi kẻ tấn công sẽ cố gắng lắng nghe lưu lượng mạng giữa máy chủ và máy khách cố gắng đăng nhập vào hệ thống bằng cách sử dụng thông tin đăng nhập hệ thống (tên người dùng, mật khẩu). Nếu thông tin đăng nhập không được mã hóa, kẻ tấn công sẽ cố gắng đăng nhập vào ứng dụng web bằng cách sử dụng thông tin đăng nhập bị đánh cắp. Nhưng nếu thông tin đăng nhập được mã hóa, kẻ tấn công sẽ cố gắng đăng nhập bằng cách lặp lại các gói tin mạng bị đánh cắp - giống như người dùng được ủy quyền đã gửi đến cơ sở dữ liệu để xác thực.

**4Đề xuất giải pháp vấn đề**

Như đã đề cập, tính bảo mật của cơ sở dữ liệu không thể dựa vào bảo mật của các cấp độ hệ thống phần mềm khác. Cơ sở dữ liệu phải được tự bảo vệ càng nhiều càng tốt. Chúng tôi đã tạo cơ sở dữ liệu với các thông số kỹ thuật sau:

1) người dùng có quyền và đặc quyền truy cập vào ứng dụng cơ sở dữ liệu không được kiểm soát thông qua bảng Người dùng, nhưng họ được đăng ký là người dùng cơ sở dữ liệu thông thường mà không có bất kỳ vai trò quản trị nào,

2) Không có một người dùng thông thường nào (ngoại trừ chủ sở hữu CSDL) có thể có bất kỳ quyền nào (chọn, chèn, cập nhật, xóa) trên bất kỳ bảng nào,

3) cách duy nhất để xem, chèn, sửa đổi hoặc xóa dữ liệu là thông qua các quy trình được lưu trữ, trong đó tất cả người dùng đều có quyền thực hiện các thủ tục,

4) trong các thủ tục được lưu trữ không có truy vấn được tạo động nào có thể được thực thi thông qua các lệnh execute\_sql (chuỗi); Truy vấn được xây dựng với các tham số thủ tục được lưu trữ (xem

Hình 1)

5) các thủ tục được lưu trữ có logic tích hợp để kiểm tra quyền và quyền của người dùng trên các chức năng kinh doanh khác nhau (thông qua các bảng có thuộc tính được mã hóa),

6) các thủ tục được lưu trữ có logic tích hợp để kiểm tra các thuộc tính của các tham số được gửi cho chúng (chiều rộng, từ trong danh sách đen, các ký tự được mã hóa trong danh sách đen, v.v.),

7) các cuộc gọi đến các thủ tục được lưu trữ với các giá trị tham số đáng ngờ được ghi vào các bảng đặc biệt,

8) Quản trị viên cơ sở dữ liệu cục bộ chỉ có thể thay đổi quyền và quyền của người dùng thông qua các quy trình được lưu trữ để xác định xem yêu cầu có đến từ mạng LAN hay không.

Ngoài các thông số kỹ thuật cơ sở dữ liệu, đối với ứng dụng dựa trên web, điều rất quan trọng là thông tin đăng nhập phải được chuyển từ người dùng sang máy chủ bằng cách sử dụng Lớp cổng bảo mật (SSL) được chứng nhận bởi bên đáng tin cậy.

Hãy giải thích các thông số kỹ thuật cơ sở dữ liệu được đề xuất không khó thực hiện.

Thông thường ứng dụng được kết nối với cơ sở dữ liệu thông qua người dùng cơ sở dữ liệu siêu mạnh và người dùng ứng dụng sau đó được kiểm soát gián tiếp thông qua bảng "Người dùng", có thể dễ dàng truy cập từ ứng dụng. Bằng cách thực hiện đặc tả số 1) kết nối giữa ứng dụng và cơ sở dữ liệu được thiết lập cho người dùng cơ sở dữ liệu cụ thể. Thông qua bảo mật DBMS tích hợp, bất kỳ người dùng cơ sở dữ liệu thông thường nào cũng sẽ chỉ có quyền rõ ràng để thực hiện quy trình được lưu trữ (đặc tả 3) và sẽ không có quyền truy cập trực tiếp vào các bảng cơ sở dữ liệu (đặc tả 2). Điều đó có nghĩa là không một người dùng nào có thể tạo bất kỳ loại truy vấn nào - không tĩnh, cũng không động. Logic nghiệp vụ được thực hiện thông qua các lệnh SQL được lưu trữ trong các thủ tục và giao tiếp duy nhất giữa ứng dụng và cơ sở dữ liệu là gọi các thủ tục được lưu trữ và gửi các giá trị của các tham số của thủ tục được lưu trữ (đặc tả số 4).

Trong các bảng đặc biệt chỉ có thể được truy cập thông qua các thủ tục được lưu trữ, quyền người dùng được xác định ở định dạng được mã hóa. Trước khi thực hiện chức năng nghiệp vụ của thủ tục, quyền của người dùng cơ sở dữ liệu được kiểm tra đối với vai trò kinh doanh đã xác định. Nếu người dùng không có quyền đối với một số chức năng kinh doanh, thông báo lỗi sẽ được tạo (thông số kỹ thuật số 5). Các quy trình được thiết kế để kiểm tra (xác thực) các giá trị của các tham số và nếu kẻ tấn công cố gắng chèn mã đáng ngờ vào các tham số này, người dùng, địa chỉ IP người dùng, tên máy tính, tên thủ tục và giá trị tham số được ghi lại trong bảng nhật ký và báo động được nâng lên (thông số kỹ thuật 6 và 7). Thiết kế của thủ tục lưu trữ được trình bày tại Hình 1.

|  |
| --- |
| Tạo thủ tục sp\_UpdateEmpAge(par1 int, par2 int)  BẮT ĐẦU  nếu không fnCheckPermission (người dùng, UpdateEmpAge) bắt đầu  error\_message('Người dùng không được phép thực hiện hành động này') trả về  kết thúc    nếu không fnValidateParam(par1, int, UpdateEmpAge) trở về    nếu không fnValidateParam(par2, int, UpdateEmpAge) trở về    CẬP NHẬT nhân viên  SET age = par2  WHERE employeeid = par1    Kết thúc |

Hình 1 – Thiết kế quy trình lưu trữ Database

Người quản trị có quyền thay đổi quyền của người dùng trên logic nghiệp vụ – không phải đối với các bảng cơ sở dữ liệu. Sẽ rất nguy hiểm nếu kẻ tấn công có thể xác định bằng cách nào đó là quản trị viên. Đó là lý do tại sao quản trị viên chỉ có thể tự xác định từ mạng LAN (đặc điểm kỹ thuật 8).

Khi người dùng kết nối với ứng dụng cơ sở dữ liệu thông qua Lớp cổng bảo mật (SSL), thông tin đăng nhập được mã hóa, vì vậy kẻ tấn công không thể nhìn thấy văn bản rõ ràng của tên người dùng và mật khẩu. Nếu kẻ tấn công cố gắng lưu dữ liệu được truyền qua mạng từ người dùng này sang máy chủ khác trong thời điểm người dùng đăng nhập và truyền lại (tự nhận mình là người dùng đã đăng nhập thành công), giao thức SSL sẽ đánh bại cuộc tấn công này bằng cách sử dụng nonce, một số duy nhất một lần - id kết nối không thể lặp lại (máy chủ sẽ nhận ra rằng nó được lặp lại khối mạng và sẽ bỏ qua nó).

Kiểm tra lỗ hổng bảo mật của cơ sở dữ liệu đó có thể được thực hiện theo cách sau:

-        phân tích mã nguồn của thủ tục được lưu trữ và xem xét phân tích lan truyền Lỗi cuối cùng bằng cách sử dụng kỹ thuật Hộp trắng,

-        cài đặt các công cụ để kiểm tra bảo mật, như đã đề cập trong phần giới thiệu, để quét lỗ hổng tự động theo kỹ thuật Hộp đen,

-        trong cả hai kỹ thuật được đề cập kiểm tra phần lớn các mẫu SQL injection và XSS [16],

-        kiểm tra các lỗ hổng bằng cách xác định là người dùng cơ sở dữ liệu thông thường và cố gắng có được quyền quản trị,

-        kiểm tra xem người dùng thông thường có thể kiểm soát HĐH hay không bằng cách tạo một số truy vấn SQL độc quyền có thể truy cập các tệp (cấu hình) trong HĐH.

Nói chung, đó là một thực hành tốt để tạo ra các kế hoạch kiểm thử với mô tả chính xác của từng kịch bản thử nghiệm và liên kết các phương pháp giảm thiểu nếu một số kịch bản thử nghiệm không thành công. Trong trường hợp này, chúng tôi không thể cung cấp các mẫu thử nghiệm mà là các kịch bản tấn công được đề cập. Thật không may, kiểm thử bảo mật được thúc đẩy bằng cách thăm dò các giả định và các lĩnh vực phức tạp cụ thể để xác định cách một chương trình có thể bị phá vỡ

**5Kết thúc**

Để bảo vệ phần mềm với cơ sở dữ liệu khỏi kẻ tấn công, cần liên tục theo dõi và điều tra tất cả các trường hợp lỗ hổng phần mềm được công bố. Một trong những cách để cải thiện tính bảo mật của cơ sở dữ liệu là nhúng lá chắn vào dữ liệu đầu vào có thể gây ra các lỗi bảo mật tiềm ẩn, bằng cách sử dụng các quy trình được lưu trữ với xác thực tích hợp trên dữ liệu đầu vào và để bảo vệ việc chuyển thông tin đăng nhập để xác thực người dùng thông qua các kỹ thuật mã hóa web tiêu chuẩn. Vì bảo mật đầy đủ không thể được đảm bảo, cách tiếp cận có hệ thống để kiểm tra cơ sở dữ liệu đó phải được thực hiện bằng cách sử dụng các kịch bản tấn công. Chúng tôi đã chuẩn bị mô hình cơ sở dữ liệu được mô tả trong ba kiến trúc: Java-Oracle, VB .NET - MS SQL Server và PHP-MySQL để kiểm tra mô hình được đề xuất.

**Lời cảm ơn**

Công việc này đã được thực hiện trong dự án

'Quản lý chất lượng phần mềm tối ưu

Khuôn khổ ', được hỗ trợ một phần bởi Bộ Khoa học và Phát triển Công nghệ Cộng hòa Serbia theo Dự án số TR-35026*.*

*Tham khảo:*

[1] S. Obradović, S.S.Ilić, V. Marković, *Trách nhiệm quản lý liên quan đến bảo mật dữ liệu*, Kỷ yếu hội nghị quốc tế UNITECH, Gabrovo, Bulgaria 2009.

[2] C. Weissman, *Kiểm tra thâm nhập, Tiểu luận 11*, Bộ Quốc phòng, "Máy tính đáng tin cậy

Tiêu chí đánh giá hệ thống," DoD 5200.28STD, tháng 12 năm 1985 (The Orange Book).

www.acsac.org/secshelf/book001/11.pdf

[3] L. Khan, M. Awad, B. Thuraisingham, *Một hệ thống phát hiện xâm nhập mới sử dụng các máy vectơ hỗ trợ và phân cụm phân cấp*, Tạp chí VLDB, Tập 16, 2007. 507-521.

[4] Peng Liu, *DAIS:* *Hệ thống cách ly tấn công dữ liệu thời gian thực cho các ứng dụng cơ sở dữ liệu thương mại*, Kỷ yếu Hội nghị ứng dụng bảo mật máy tính thường niên lần thứ 17 - ACSAC 2001. 219-229

[5] Cenzic. Báo cáo xu hướng bảo mật ứng dụng Q1 2009. http://www.cenzic.com/downloads/ Cenzic\_AppSecTrends\_Q1 Quý 2 năm 2009.pdf

[6] G. T. Buehrer, B. W. Weide, P. A. G. Sivilotti, *Sử dụng xác thực cây phân tích cú pháp để ngăn chặn các cuộc tấn công SQL injection*, Hội thảo quốc tế lần thứ năm về kỹ thuật phần mềm và phần mềm trung gian - SEM 2005 Tháng Chín 2005 Lisbon, Bồ Đào Nha

[7] K.V.N.Sunitha và M.Sridevi, *Hệ thống phát hiện tự động cho tấn công SQL injection*, Tạp chí Quốc tế về Khoa học Máy tính và Bảo mật (IJCSS), Tập (4): Số (4), trang 426-435

[8] E. Adi, I. Salomo, *Detect and Sanitise Encoded Cross-Site Scripting and SQL Injection Attack Strings Using a Hash Map*, Australian Information Security Management Conference, 2010.

[9] Z. Yang, S. Sesay, J. Chen and Du Xu, *A* *Secure Database Encryption Scheme*,

American Journal of Applied Sciences 1 (4): 327-331, 2004

[10] S. Burnett, S. Paine, RSA Security's *Official Guide to Cryptography*, Osborne/McGraw-

Hill, 2001

[11] G. Jabbour, D. A. Menasce, Policy-*Based Enforcement of Database Security*

*Configuration through Autonomic Capabilities*, Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems ICAS'08

[12] Min A Jeong, Jung-Ja Kim, Y. Won, *A Flexible Database Security System Using multiple Access Control Policies*, International Conference on Database and Expert Systems Applications - DEXA 2003, LNCS 2736, pp. 876–885, 2003.

[13] A. Hanna, H. Z. Ling, J. Furlong, M. Debbabi, *Towards Automation of Testing High-Level Security Properties*, The Eighth IAPR International Workshop on Document Analysis Systems DAS 2008, Nara, Japan

[14] A. Kieyzun, P. J. Guo, K. Jayaraman, M. D. Ernst, "Automatic Creation of SQL Injection and Cross-Site Scripting Attacks", Proceedings of the 31st International Conference on Software Engineering ICSE '09

[15] Justin Clarke, *SQL Injection attacks and defence*, Syngress Publishing, Inc. Elsevier, Inc., 2009

[16] P. Hope, B. Walther, Web Security Testing Cookbook, 1st Edition, O'Reilly Media, Inc., 2008