TẠP CHÍ KHOA HỌC ĐẠI HỌC ĐÀ LẠT Tập 8, Số 2, 2018 24–44

HỆ THỐNG CẢNH BÁO TẤN CÔNG THAY ĐỔI GIAO DIỆN WEBSITE

Trần Đắc Tốt[a*], Đặng Lê Nam[a], Phạm Nguyễn Huy Phương[a]

[a]Khoa Công nghệ Thông tin, Trường Đại học Công nghiệp Thực phẩm TP. Hồ Chí Minh, TP. Hồ Chí Minh, Việt Nam
[*]Tác giả liên hệ: Email: tottd@cntp.edu.vn

Lịch sử bài báo
Nhận ngày 25 tháng 01 năm 2018
Chỉnh sửa ngày 18 tháng 04 năm 2018 | Chấp nhận đăng ngày 14 tháng 05 năm 2018

Tóm tắt

Hậu quả của việc tấn công làm thay đổi giao diện, nội dung của Website là đặc biệt nghiêm trọng. Vì vậy, phải có những phương pháp phát hiện kịp thời những hình thức tấn công này nhằm hạn chế tối đa hậu quả. Trong bài báo này, tác giả trình bày một phương pháp mới cho phép phát hiện sự thay đổi giao diện, nội dung của Website. Phương pháp đề xuất được phát triển dựa trên hàm băm và các kỹ thuật đối sánh chuỗi để tìm sự thay đổi nội dung dựa trên sự khác biệt giữa hai tài liệu HTML (Hyper Text Markup Language) của cùng một trang Web tại hai thời điểm khác nhau. Đồng thời, áp dụng thuật toán C4.5 (Quinlan, 1993) để tăng độ chính xác của các cảnh báo bảo mật. Hơn thế nữa, tác giả đã triển khai thành một ứng dụng với giao diện hài hòa, dễ sử dụng. Kết quả thực nghiệm cho thấy hệ thống phát hiện chính xác và cảnh báo kịp thời cho quản trị viên Website khi có sự thay đổi nội dung.

Từ khóa: Cảnh báo tấn công; Đối sánh chuỗi; Giám sát Website; Kiểm tra tính toàn vẹn; Tấn công Defacement.

Mã số định danh bài báo: http://tckh.dlu.edu.vn/index.php/tckhdhdl/article/view/412

Loại bài báo: Bài báo nghiên cứu gốc có bình duyệt

ANTI-WEBSITE DEFACEMENT SYSTEM

Tran Dac Tot[a*], Dang Le Nam[b], Pham Nguyen Huy Phuong[b]

[a]The Faculty of Information Technology, Hochiminh City University of Food Industry, Hochiminh City, Vietnam
[*]Corresponding author: Email: tottd@cntp.edu.vn

Article history
Received: January 25th, 2018
Received in revised form: April 18th, 2018 | Accepted: May 14th, 2018

Abstract

Recently the impacts of hackers' attacks which change the interface and content of Website are particularly serious. Therefore, there should be methods to allow real-time detection of these changes to reduce the consequences of these attacks. This article presents a new method to detect the changes of the interface and content of Website. Our proposed method is developed based on the HTML Boyer-Moore algorithm combined with the MD5 hash function and has been built into an application with a user-friendly interface. Moreover, we applied C4.5 algorithm to enhance the accuracy of the warning messages. Changes to the Website such as inserting new content, deleting or editing old content, changing the format, color, size, and type of content will be immediately recorded and notified to the Website administrator. The application will also highlight the locations of these changes and send a warning message and recommendations to the webmaster for processing. Experiment results show that the proposed method can accurately locate and produce spontaneous warnings to the Website administrator.

Keywords: Defacement attack; Defect attack alert; Integrity checking; String matching; Website monitoring.

Article identifier: http://tckh.dlu.edu.vn/index.php/tckhdhdl/article/view/412

Article type: (peer-reviewed) Full-length research article

1. ĐẶT VẤN ĐỀ

Trong các năm gần đây, việc ứng dụng công nghệ thông tin (CNTT) ngày càng được tăng cường trong các cơ quan, doanh nghiệp, tổ chức đặc biệt đối với các cơ sở giáo dục, đó là triển khai rộng rãi các hệ thống quản lý, cung cấp các dịch vụ hành chính công trực tuyến, từng bước thiết lập chính quyền điện tử... Cùng với sự phát triển ứng dụng CNTT, mối nguy về an toàn, bảo mật đối với các hệ thống thông tin quản lý cũng ngày càng gia tăng. Các hoạt động tấn công mạng nhằm vào các cổng/trang thông tin điện tử (TTĐT) của các cơ quan giáo dục và các tổ chức tại Việt Nam có dấu hiệu gia tăng, luôn đứng trước nguy cơ thường trực về mất an toàn, an ninh thông tin (ATANTT).

Một trong những hình thức tấn công (*defacement*) được biết rộng rãi nhất là tấn công thay đổi nội dung, giao diện của Website (Gaurav, Newley, & Jason, 2015). Hình thức tấn công này thường sử dụng các mã độc (*virus, worm, trojan…*) để xóa bỏ, sửa đổi, hoặc thay thế nội dung các trang Web trên hệ thống (*Webserver*) (Stalling, 1999). Lỗ hổng Website là mục tiêu tiềm tàng của việc tấn công vì các mục đích khác nhau. Những kẻ phá hoại có các công cụ để tìm kiếm các lỗ hổng Website một cách sâu rộng và nhanh chóng, tiếp theo sẽ tiến hành khai thác những điểm yếu đó (Amanda, 2003; OWASP, 2017).

Những cuộc tấn công thay đổi Website đã được thực hiện để xâm phạm tính toàn vẹn của Website bằng một trong những hình thức sau: Thay đổi nội dung của trang Web; Thay đổi bất kỳ phần nào của nội dung trang Web; Thay thế toàn bộ trang Web; Chuyển hướng trang Web; và Phá hủy hoặc xóa bỏ trang Web (Charles, Shari, & Jonathan, 2015; British Broadcasting Company, 2014; & Cục An toàn Thông tin, 2016). Các hệ thống kiểm soát an ninh mạng như Firewall, VPN (*Virtual Private Network*), PKI (*Public Key Infrastructure*),… là những công cụ quan trọng để bảo vệ cho hệ thống Website được an toàn hơn, nhưng các hệ thống nói trên không đầy đủ để đảm bảo an ninh Website, do đó cần những cơ chế đảm bảo an ninh tốt hơn (Hiệp hội An toàn Thông tin Việt Nam, 2017). Trong thực tế có nhiều phương pháp được đề xuất để bảo vệ hệ thống Website chống lại các cuộc tấn công. Tuy nhiên các phương pháp này cũng có nhiều ưu và nhược điểm, cụ thể như các mô tả sau.

Cashin (2000) đề xuất chương trình xác minh tính toàn vẹn của tập tin. Chương trình này giúp xác định liệu một kẻ xâm nhập đã sửa đổi gì trong máy tính, bằng cách tạo ra một cơ sở dữ liệu mới chính là một bản sao của những dữ liệu cần bảo vệ. Bản cơ sở dữ liệu mới này được lưu trữ ở nơi nào đó an toàn và sau đó có thể được sử dụng để so sánh nhằm phát hiện bất kỳ sửa đổi bất hợp pháp trong máy tính. Kỹ thuật này chưa có phương án tự bảo vệ mình khi bản thân bị tấn công. Xiang và Hongtao (2008) đề xuất hai kỹ thuật sử dụng sơ đồ Watermarking để phát hiện các trang Web giả mạo. Phương pháp này sử dụng kỹ thuật HMAC (*Keyed-Hashing for Message Authentication*) hoặc kỹ thuật NNSC (*Nonnegative Sparse Coding*) để tạo ra một hình mờ từ một trang Web và sau đó nhúng vào trong trang Web. Khi có sự cố giả mạo xảy ra, hình mờ trong trang Web giả mạo sẽ bị xóa hoặc không phù hợp với nội dung, dựa vào đó ta sẽ cảnh báo. Davanzo,

Medvet, và Bartoli (2010) đề xuất một công nghệ cho phép phát hiện defacement tự động bằng cách xây dựng một hồ sơ mẫu cho trang Web. Trang Web được theo dõi và mỗi khi có điều gì đó "bất thường" xảy ra, sẽ gởi cảnh báo cho người dùng. Tushar, Vineet, và Vivek (2011) đã phát triển một trình duyệt Web mẫu, có thể được sử dụng để kiểm tra, phát hiện defacement Website với cơ chế hoạt động như sau: Sau khi mở Website bằng trình duyệt, trang web sẽ được theo dõi và sử dụng hàm băm để lấy mã băm của Website, lưu trữ lại, sau một khoảng thời gian Website được băm lại. Một tùy chọn kiểm tra Website có bị tấn công được trao cho người quản trị. Người quản trị kiểm tra mã băm hiện tại của Website so sánh với mã băm đã lưu trước đó. Nếu hai mã băm giống nhau có nghĩa là Website bình thường, nếu không giống thì nghĩa là đã bị tấn công. Người quản trị có thể phục hồi Website về mặc định ban đầu. Phương pháp này chỉ hiệu quả đối với các trang tĩnh.

Bên cạnh đó, Tushar, Vineet, và Vivek (2012) cũng đề xuất cơ chế phát hiện tấn công Website và phát hiện chính xác vị trí của sự tấn công bằng cách sử dụng thuật toán Diff. Bằng cách tích hợp các kỹ thuật phát hiện tấn công trên một trình duyệt Web. Với kỹ thuật này, họ tính toán mã băm cho các Website và so sánh với mã băm đã lưu trước đó tương ứng với Website đó. Nếu kết quả so sánh mã băm là giống nhau thì đánh dấu nó bình thường, nếu khác sẽ được đánh dấu là bị tấn công. Với các trang bị đánh dấu là bị tấn công sẽ sử dụng thuật toán Diff, tính toán so sánh giữa hai trạng thái của một trang để phát hiện ra vị trí bị tấn công. Các tác giả cũng đánh giá cách tiếp cận của họ hiệu quả hơn so với các cách tiếp cận khác với bốn tham số: Độ phức tạp không gian; Độ phức tạp thời gian; Số lần phát hiện tấn công; và Giá thành. Tuy nhiên cơ chế này chỉ áp dụng cho các Website tĩnh.

Ramniwas, Nikhil, và Deepak (2014) đề xuất một cách tiếp cận mới để phát hiện Website bị tấn công bằng cách phân tích các tập tin nhật ký (*log file*). Các tập tin nhật ký được trích xuất từ máy chủ Web và được xử lý để lấy thông tin cần thiết. Thông tin này sẽ được chuyển sang định dạng XML (*eXtensible Markup Language*), vì các thông tin ở định dạng XML sẽ giúp việc tìm kiếm hiệu quả hơn so với định dạng văn bản thuần. Sau khi xử lý xong, tập tin nhật ký sẽ được phân tích để tìm ra các mẫu và các mẫu nãy sẽ được sử dụng để phát hiện ra các Website bị tấn công. Phương pháp này khá hiệu quả, tuy nhiên việc phát hiện tấn công hoàn toàn phụ thuộc vào tập tin nhật ký, vì một lý do nào đó nếu bị sửa đổi thì sẽ bỏ qua các trường hợp bị tấn công dẫn đến không cảnh báo kịp thời. Rashmi và Shahzia (2015) đề xuất cơ chế phát hiện tấn công bằng cách sử dụng hàm băm. Hệ thống được đề xuất bao gồm phát triển một mô-đun tích hợp vào máy chủ Web Apache để phát hiện tấn công trong các Website và tự động cấu hình để trang bị tấn công không hiển thị khi máy chủ Web nhận được yêu cầu từ người dùng hợp pháp. Phương pháp này khá hay khi đề xuất ngăn cản người dùng hợp pháp truy cập các Website đã bị tấn công. Tuy nhiên nó cũng chỉ phù hợp cho các trang tĩnh.

Với mong muốn xây dựng một phương pháp hiệu quả hơn, chúng tôi nghiên cứu phương pháp mới kết hợp giám sát trong mạng nội bộ (*Local Area Network - LAN*) và giám sát từ xa. Phương pháp này kết hợp nhiều yếu tố như giám sát máy chủ, cơ sở dữ

liệu, sử dụng hàm băm và các kỹ thuật đối sánh chuỗi để tìm sự thay đổi nội dung dựa trên sự khác biệt giữa hai tài liệu HTML của cùng một Website tại hai thời điểm khác nhau. Đồng thời áp dụng thuật toán C4.5 để gia tăng độ chính xác của các cảnh báo. Từ đó xây dựng hệ thống giám sát Website nhằm phát hiện kịp thời các cuộc tấn công để đảm bảo tính toàn vẹn của trang, đồng thời tạo ra thông điệp cảnh báo có ý nghĩa khi đã bị tấn công. Đặc biệt, hệ thống này đã khắc phục được tối đa những hạn chế đã được đề cập ở trên.

Nội dung của bài báo được tổ chức như sau: Mục 2 trình bày các kiến thức cơ sở lý thuyết; Mục 3 đề xuất phương pháp phát hiện tấn công; Mục 4 trình bày các kết quả thực nghiệm khi triển khai hệ thống; và Mục 5 là kết luận và hướng nghiên cứu tương lai.

2. CƠ SỞ LÝ THUYẾT

2.1. Hàm băm

Hàm băm mật mã là hàm toán học chuyển đổi một bản tin có độ dài bất kỳ thành một dãy bit có độ dài cố định (tùy thuộc vào thuật toán băm). Dãy bit này được gọi là bản tin rút gọn (*message digest*) hay giá trị băm (*hash value*), biểu diễn cho bản tin ban đầu. Các thuật toán về hàm băm dòng MD (*Message Digest*) gồm MD2, MD4, MD5 do Rivest (1992) đề xuất. Giá trị băm theo các thuật toán này có độ dài cố định là 16 Byte. Phương pháp Secure Hash Standard (SHS) là chuẩn gồm tập hợp các thuật toán băm mật mã an toàn (*Secure Hash Algorithm - SHA*) như SHA-1 (20 Byte), SHA-256 (32 Byte), SHA-512 (64 Byte) do NIST và NSA xây dựng. Hàm băm an toàn SHA phức tạp hơn nhiều, dựa trên các phương pháp tương tự. Ngoài ra còn có một số thuật toán khác như: RIPEMD, HAVAL, Whirlpool, và Tiger. Các hàm băm đề cập ở trên đều có tính bảo mật cao, trong đó họ SHA được coi là có tính bảo mật cao nhất. Tuy nhiên, xét về tốc độ mã hoá thì MD5 là hàm băm có tốc độ mã hoá cao nhất (Piyush & Sandeep, 2014). Vì vậy, tác giả chọn MD5 để xây dựng các cơ chế nhằm phát hiện sự thay đổi của Website.

2.2. Đối sánh chuỗi

Thuật toán Boyer-Moore (Boyer & Moore, 1977) là thuật toán thực hiện tìm kiếm chuỗi rất hiệu quả trong thực tiễn. Thuật toán kiểm tra các ký tự của mẫu từ phải sang trái và khi phát hiện sự khác nhau đầu tiên, thuật toán sẽ tiến hành dịch theo hai cách:

- *Cách 1*: Dịch sao cho những phần đã so sánh trong lần trước khớp với những phần giống nó trong lần sau;
- *Cách 2*: Coi ký tự đầu tiên không khớp trên văn bản là b = T[j+i-1], ta sẽ dịch sao cho nó có một ký tự giống *b* trên xâu mẫu khớp vào vị trí đó (nếu có nhiều vị trí xuất hiện *b* trên xâu mẫu chọn vị trí phải nhất).

Với T[0 .. n-1] là văn bản có *n* ký tự và P[0 .. m-1] là cụm từ cần tìm có *m* ký tự

với *m* <= *n*. Thuật toán duyệt các ký tự trong *P* từ phải sang trái, trong trường hợp không khớp (hoặc tìm thấy *P* trong *T*) nó sử dụng hai hàm tính lại giá trị để dịch chuyển *P*. Hai hàm dịch chuyển được dùng trong thuật toán gọi là phép dịch chuyển hậu tố (*good-suffix shift*) hay là phép dịch chuyển trung khớp và dịch chuyển ký tự tồi (*bad character shift*).

2.3. Thuật toán cây quyết định C4.5

Thuật toán C4.5 (Quinlan, 1993) là một thuật toán được cải tiến cho phép xử lý trên tập dữ liệu có các thuộc tính số, tập dữ liệu bị thiếu và bị nhiễu. Thuật toán này thực hiện phân lớp tập mẫu theo chiến lược ưu tiên theo chiều sâu, xét tất cả các phép thử có thể để phân chia tập dữ liệu và chọn ra một phép thử có giá trị *GainRatio* tốt nhất (*GainRatio* là đại lượng đánh giá độ hiệu quả của thuộc tính dùng để thực hiện phép tách khi phát triển cây quyết định).

- *Thuộc tính phân loại*: Shannon (1948) đưa ra lý thuyết thông tin cung cấp một khái niệm để đo tính thuần nhất của một tập hợp gọi là *Entropy*. Giả sử các ví dụ của tập *S* thuộc *i* loại và có *C* giá trị phân loại thì công thức *Entropy* tổng quát được biểu diễn như trong công thức (1).

$$Entropy(S) = -\sum_{i=1}^{C} (P_i \log p_i) \quad (1)$$

- *Độ lợi thông tin*: Độ lợi thông tin (*Information Gain*) là một phép đo hiệu suất phân loại các ví dụ của một thuộc tính. Ví dụ, *Gain(S, A)* của thuộc tính *A*, trên tập *S*, được định nghĩa như công thức (2) sau đây:

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (2)$$

Trong đó: *Values(A)* là tập hợp có thể có giá trị của thuộc tính *A; S_v* là tập con của *S* chứa các ví dụ có thuộc tính *A* mang giá trị *v*.

- *Tỷ suất lợi ích* (*GainRatio*): Thuật toán C4.5 mở rộng cách tính độ lợi thông tin thành tỉ suất lợi ích để cố gắng khắc phục sự thiên lệch. Tỉ suất lợi ích được xác định bởi công thức (3).

$$GainRatio(S,P) = \frac{Gain(S,P)}{SplitInfo(S,P)} \quad (3)$$

Trong đó: *SplitInfo(S,P)* là thông tin do phân tách *P* trên cơ sở giá trị của thuộc tính phân loại *S*. Công thức tính như trong (4).

$$SplitInfo(S,P) = -\sum_{i=1}^{c} \frac{|P_i|}{|P|} log \frac{|P_i|}{|P|} \quad (4)$$

Trong đó: P là tập các giá trị thuộc tính của S; P_i là tập con của tập P ứng với thuộc tính S giá trị v_i; và C là số giá trị phân loại.

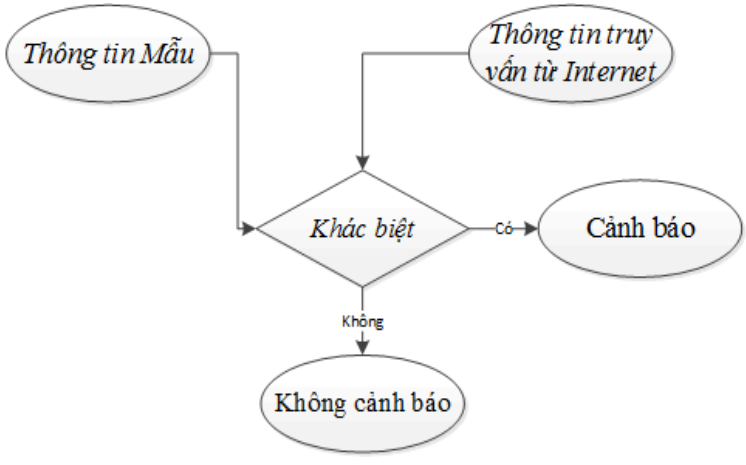
2.4. Xây dựng cơ chế phân tích và phát hiện sự thay đổi của Website

Một Website bao gồm các thành phần chính như Domain, Host, Source, và Database. Tương ứng với các thành phần này là những kỹ thuật tấn công khác nhau, chẳng hạn như: Kiểm soát truy cập website, chiếm hữu phiên làm việc, lợi dụng các thiếu sót trong việc kiểm tra dữ liệu nhập hợp lệ hay để lộ thông tin, từ chối dịch vụ… Sự phân chia như vậy giúp xem xét vấn đề rõ ràng hơn để xây dựng tám cơ chế phân tích và phát hiện sự thay đổi của website, chi tiết trình bày như dưới đây.

2.4.1. Cơ chế phát hiện thay đổi thông số của tên miền

Để phòng ngừa máy chủ tên miền (*Domain Name System-DNS*) bị tấn công (David, Manos, Paul, Tatuya, & Wenke, 2008) làm sai lệch thông tin ban đầu hoặc bị chuyển hướng truy cập vào Website giả mạo người quản trị website sẽ khai báo thông tin (tên miền, địa chỉ IP (*Internet Protocol*) của máy chủ Webserver) và sẽ được lưu trữ lại. Định kỳ kiểm tra bằng cách sử dụng thuật toán đối sánh chuỗi (đã trình bày ở Mục 2.2) để so sánh thông tin đã khai báo và thông tin được phân giải từ Internet (thông qua nhiều DNS Server). Nếu có sự khác biệt sẽ đưa ra cảnh báo, sơ đồ thuật toán minh họa như trong Hình 1. Các ý tưởng thuật toán như sau:

- Lấy thông tin đã lưu trữ trên hệ thống và thông tin nhận được khi truy vấn trên môi trường Internet;
- Sau đó so sánh các thông tin trên, nếu giống nhau tức là chưa có sự thay đổi, nên Website là an toàn. Nếu khác nhau tức thì thông số DNS đã bị sửa đổi, hệ thống sẽ gửi cảnh báo.



Hình 1. Sơ đồ thuật toán kiểm tra thông số DNS

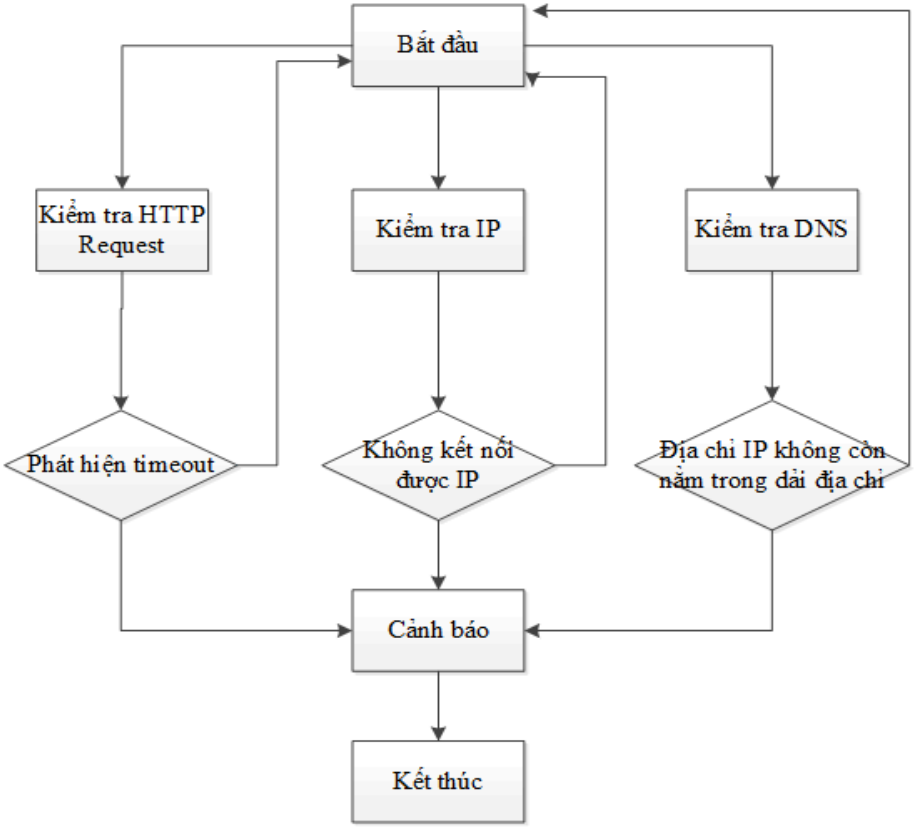
2.4.2. Cơ chế kiểm tra hoạt động/không hoạt động của Website

Để biết tình trạng hoạt động/không hoạt động của một Website, ta dựa vào ba tiêu chí để kiểm tra (DNS, HTTP Request, và IP) nhằm biết tình trạng hoạt động của Website, được trình bày chi tiết như trong Hình 2.

- *Kiểm tra DNS*: Kiểm tra xem địa chỉ IP đang xét có nằm trong danh sách địa chỉ IP mà DNS phân giải hay không. Nếu phát hiện thấy địa chỉ IP đang xét còn trong nằm trong danh sách là bình thường, ngược lại thì cảnh báo;

- Kiểm tra *HTTP Request*: Khi ta gửi yêu cầu mà Website trả về trạng thái hồi đáp (*response*) là 200 nghĩa là bình thường, ngược lại thì đã xảy ra vấn đề trong kết nối với Webserver nên cần cảnh báo;

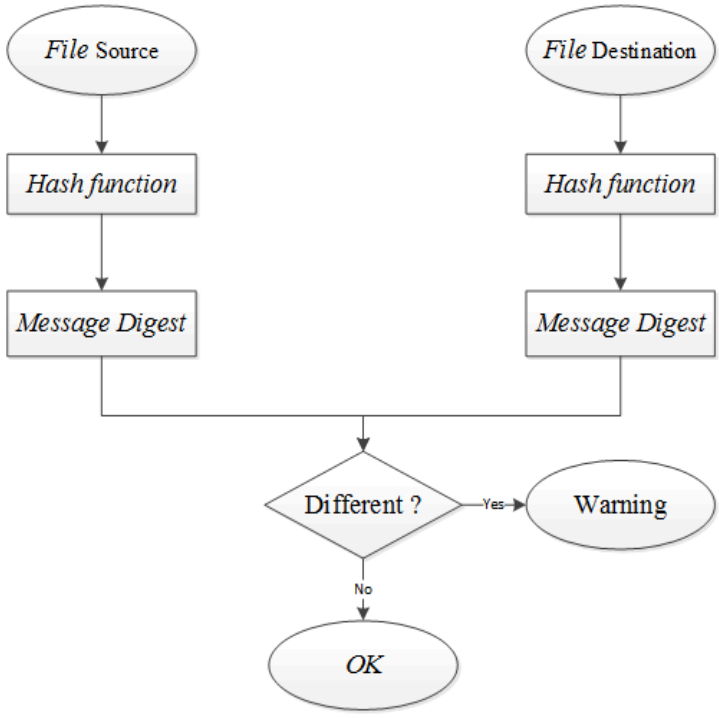
- *Kiểm tra IP Public*: Ta gửi gói tin truy vấn ICMP (*Internet Control Message Protocol*) dạng "echo-request" đến cho máy đích và lắng nghe gói tin hồi đáp ICMP "echo-response" Nếu gói tin hồi đáp trả về là "thành công" thì kết nối đến máy chủ Web bình thường, ngược lại thì ta sẽ cảnh báo.



Hình 2. Sơ đồ thuật toán kiểm tra hoạt động/không hoạt động của Website

2.4.3. Cơ chế phát hiện thay đổi tính toàn vẹn

Một Website muốn hoạt động tốt và an toàn thì việc đầu tiên chúng ta cần phải làm là bảo vệ thư mục chứa mã nguồn của Website trên máy chủ Web. Mọi thư mục, tập tin cần bảo vệ liên tục 24/24. Vì chỉ cần một thay đổi nhỏ trên tập tin cấu hình có thể dẫn đến Website hoạt động sai lệch, hoặc các tập tin bị chèn thêm hoặc xóa nội dung. Thậm chí có nhiều tập tin độc hại được các kẻ tấn công đưa lên và được thay đổi liên tục nhằm qua mặt các phần mềm chống virus. Vì vậy, phương pháp này áp dụng hàm băm và đối sánh chuỗi để kiểm tra sự thay đổi, thêm xóa tập tin, sơ đồ thuật toán như trong Hình 3



Hình 3. Sơ đồ thuật toán kiểm tra sự thay đổi

Với cơ chế này, tác giả xây dựng cơ chế phát hiện thay đổi mã nguồn, giám sát thư mục hoặc giám sát tập tin.

2.4.4. Cơ chế phát hiện thay đổi cơ sở dữ liệu (CSDL)

Để biết được sự thay đổi thông tin trong CSDL ta theo dõi những hành động của Hệ quản trị CSDL để biết thông tin gì đã được thay đổi trong CSDL, từ đó có thể kiểm soát được những vấn đề có thể ảnh hưởng tới thông tin đã lưu trữ trong CSDL. Trong SQL Trace, các sự kiện được thu thập nếu chúng là thể hiện của các lớp sự kiện được đặc tả trong định nghĩa truy vết. Các sự kiện này được lọc theo vết hoặc được đưa vào hàng đợi để xem xét. Các nguồn sự kiện (*event source*) có thể là nguồn tạo ra các sự kiện truy vết như các câu lệnh Transact-SQL (*select, insert, update, delete*...) Sau khi một sự kiện

xảy ra, nếu chúng thuộc các lớp sự kiện được định nghĩa trước đó thì chúng sẽ được thu thập. Trong bài báo này, tác giả tạo tập tin truy vết (*file trace*) để theo dõi Database (dùng trong phần mềm Microsoft SQL Server 2008) của Website bao gồm các sự kiện sau:

- Phân tích thay đổi trên tài khoản (*user*) gồm: Tạo tài khoản, xoá tài khoản, cấp/huỷ quyền của tài khoản;
- Phân tích thay đổi trên bảng (*table*) gồm: Tạo bảng, xoá bảng, thêm cột, xoá cột, thay đổi kiểu dữ liệu, thêm/xoá khoá chính, khoá ngoại;
- Phân tích thay đổi trên *Trigger/Store Procedure* gồm: Tạo/xoá, sửa;
- Phân tích thay đổi kích thước/cấu trúc Database: Thay đổi dung lượng tập tin (dữ liệu, log), thêm tập tin.

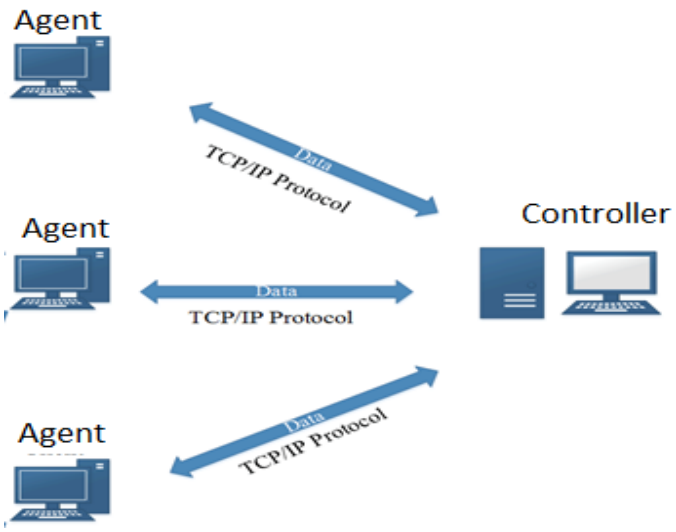
Sau một khoảng thời gian định trước, chúng tôi đọc dữ liệu từ tập tin truy vết để theo dõi, lọc và chuẩn hóa các thông tin thao tác đến dữ liệu của Database trong khoảng thời gian trên. Mô hình như trong Hình 4.

Hình 4. Mô hình giám sát Database trong SQL Server

2.4.5. Cơ chế phát hiện tấn công làm tê liệt Website

Khi bị xâm nhập, kẻ tấn công có thể kiểm soát được Website hoặc máy chủ Web, khi đã chiếm được quyền quản trị thì kẻ tấn công (*hacker*) sẽ vô hiệu hóa các ứng dụng có liên quan nhằm che đậy hành vi, dễ dàng can thiệp sâu vào hệ thống. Một số trường hợp được bảo mật tốt thì khi không xâm nhập được chúng sẽ tìm cách đánh sập hoặc làm tê liệt hoạt động của Website/Webserver. Một kỹ thuật được sử dụng phổ biến là tấn công từ chối dịch vụ (*Denial of Service attack - DDoS/DoS*) mà sơ khai nhất là hình thức DoS (*Denial of Service*), tiếp đến là tấn công từ chối dịch vụ phân tán (*Distributed Denial of Service - DDoS*) và tấn công theo phương pháp phản xạ phân tán (*Distributed Reflection Denial of Service - DRDoS*). Để giải quyết trường hợp kẻ tấn công phá hủy chương trình giám sát hay bị tấn công từ chối dịch vụ làm tê liệt máy chủ Web. Tác giả đề xuất sử dụng mô hình *Agent – Controller* (Hình 5). *Agent* chính là một ứng dụng được cài đặt trên các máy chủ. *Controller* là trung tâm giám sát được cài đặt trên một máy độc lập với máy chủ Web, *Controller* này sẽ tiếp nhận các thông tin từ *Agent* gởi về. Với mô hình này giữa *Agent* và *Controller* sẽ duy trì kênh liên lạc riêng sử dụng *Advanced Encryption Standard* (AES) để mã hóa các thông điệp trao đổi và định kỳ *Controller* sẽ gởi thông tin liên lạc

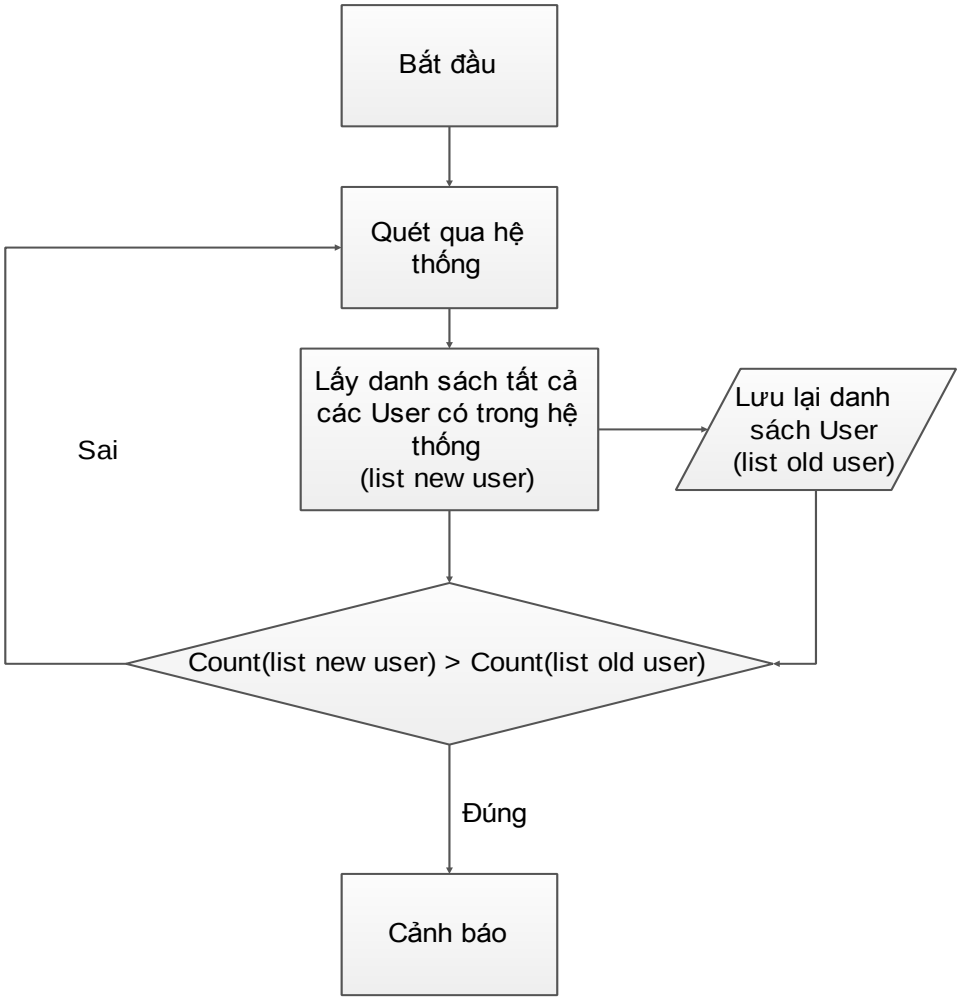
đến *Agent*, nếu mất thông tin liên lạc trong một khoảng thời gian nhất định thì sẽ cảnh báo.



Hình 5. Mô hình Agent - Controller

2.4.6. Cơ chế phát hiện thêm mới người dùng trên hệ thống

Hacker thường dùng phương pháp thêm mới người dùng để có thể chiếm quyền trên máy tính. Chính vì vậy phải có phương án phát hiện thêm mới người dùng trên máy tính để kịp thời cảnh báo. Để phát hiện thêm mới người dùng, hệ thống sẽ quản lý danh sách các người dùng đang có trên máy tính. Hệ thống sẽ quét qua máy tính lấy danh sách người dùng trên hệ thống, sau đó lưu lại. Lần tiếp theo lại quét qua máy tính lấy danh sách người dùng lên và so sánh với danh sách đã lưu trước đó. Nếu phát hiện có bất kỳ người dùng nào mới lấy không có trong danh sách trước đó nghĩa là đã được thêm mới. Hệ thống sẽ gửi thông tin cho người quản lý hệ thống biết để có thể kịp thời ngăn chặn hành vi xâm nhập hệ thống (Hình 6).



Hình 6. Sơ đồ thuật toán kiểm tra sự thay đổi của người dùng

2.4.7. Cơ chế giám sát thay đổi nội dung

Đối với một số Website không cho phép cài đặt *Agent* giám sát vào máy chủ. Do đó nhóm tác giả đề xuất cần có cơ chế giám sát thay đổi nội dung từ xa (Hình 7). Để phát hiện sự thay đổi văn bản trong một trang Web thì cần phải thu thập tất cả các văn bản trong trang Web đó rồi tiến hành phân tích, đánh giá văn bản bị thay đổi như thế nào. Chi tiết được trình bày như sau:

- *Bước 1*: *Khai báo thông tin Webpage*: Hệ thống nhận dữ liệu đầu vào là Webpage cần được so khớp (*Webpage check*);

- *Bước 2*: *Lấy tài liệu HTML của Webpage:* Toàn bộ mã nguồn của trang Web được lấy về định kỳ do người dùng xác lập sẵn;

- *Bước 3*: *Lưu tất cả văn bản xuống cơ sở dữ liệu*: Mục đích lưu văn bản xuống cơ sở dữ liệu là để đối sánh cho lần kiểm tra tiếp theo. Dữ liệu sau khi được lấy ở Bước 2 sẽ được lưu xuống cơ sở dữ liệu trong lần lấy đầu tiên (dữ liệu mẫu), các lần sau việc lưu dữ liệu do người dùng chỉ định;

- *Bước 4*: *So sánh đối chiếu văn bản*: Kiểm tra độ dài của văn bản bằng cách so sánh độ dài nội dung văn bản vừa lấy được với nội dung văn bản đã được lưu trước đó, nếu tỉ lệ giữa độ dài văn bản mới lấy so với độ dài đã lưu vượt qua một hằng số ngưỡng đã cho trước (<50%, hoặc do người dùng chỉ định) thì đưa ra cảnh báo, ngược lại, chuyển sang Bước 5. Kiểm tra cụm từ bắt buộc có trong Website. Mỗi Website đều có các cụm từ đặc trưng của cơ quan, công ty, tổ chức (như copyright (tên công ty, cơ quan,…)). Bằng cách tìm kiếm nội dung của các cụm từ này trong nội dung hiện tại của Website, nếu bất cứ cụm từ nào mà không tìm thấy thì đưa ra cảnh báo, ngược lại, chuyển sang Bước 5. Hàm kiểm tra được minh họa như sau:

  isCheck ← *FindNotExistText(subString, parentString)*

  Trong đó hàm *FindNotExistText(subString, parentString)* là hàm sử dụng thuật toán *BoyerMoore* để tìm sự tồn tại của chuỗi cần tìm (*subString*). Nếu *isCheck = true* nghĩa là không có sự mất mát nào (bình thường) ngược lại là cảnh báo.

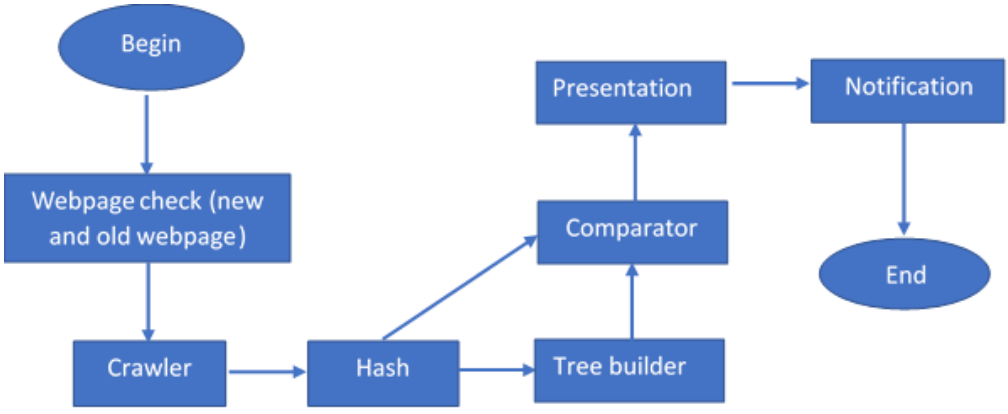
  Kiểm tra cụm từ không bao giờ cho phép xuất hiện trong trang Web. Các cụm từ này thường là các cụm từ mà tin tặc thường để lại sau khi tấn công vào một trang Website nào đó, như "Hack by…". Bằng cách tìm kiếm nội dung của các cụm từ này trong nội dung hiện tại của Website, nếu bắt cứ cụm từ nào mà được tìm thấy thì đưa ra cảnh báo, ngược lại, chuyển sang Bước 5; Kiểm tra sự tồn tại của cụm từ khả nghi tương tự như cách kiểm tra sự mất mát cụm từ nhưng khác về giá trị trả về của nó là ngược lại. Tức là, isCheck = true nghĩa là xuất hiện cụm từ khả nghi và sẽ cảnh báo, ngược lại thì bình thường;

- *Bước 5*: Từ kết quả xử lý của *Module Crawler*, thay vì sử dụng trực tiếp thuật toán đối sánh chuỗi để so sánh ngay và tìm ra sự thay đổi của Webpage, phương pháp được đề xuất sẽ sử dụng thuật toán MD5 (*Hash*) để băm kết quả thành một chuỗi để lưu trữ và so sánh với kết quả băm trước đó. Nếu kết quả băm ở thời điểm hiện tại có sự thay đổi so với kết quả băm đã lưu trữ trước đó sẽ đồng nghĩa với việc Webpage đã thay đổi về nội dung. Lúc này hệ thống mới tiến hành áp dụng thuật toán đối sánh chuỗi để so khớp (*Comparator*) cụ thể trên toàn bộ Webpage. Để làm tăng tốc độ của phương pháp đề xuất, hệ thống bổ sung thêm *Module Tree Builder* nhằm chia nhỏ Webpage thành nhiều phần theo cấu trúc HTML của cây trước khi so sánh. Vì vậy, khi phát hiện Webpage bị thay đổi ở phần cấu trúc nào, hệ thống sẽ

chỉ so khớp phần nội dung của cấu trúc đó bằng đối sánh chuỗi thay vì so khớp toàn bộ Webpage;

- *Bước 6*: Sau khi có kết quả so sánh, kết quả sẽ được lưu trữ và chuyển qua bộ phận cảnh báo.

Như vậy, trong phương pháp này, hệ thống sẽ giảm được thời gian vì đã không cần phải luôn so khớp bằng đối sánh chuỗi ở mọi giai đoạn của phương pháp mà chỉ áp dụng khi chắc chắn Webpage có sự thay đổi. Thêm vào đó, cấu trúc cây sẽ giới hạn phạm vi thay đổi của Webpage nên khi áp dụng đối sánh chuỗi sẽ hiệu quả hơn.Với cơ chế này, có thể kiểm tra sự mất mát cụm từ và sự xuất hiện từ khả nghi để nhận biết Website có bình thường hay không.



Hình 7. Sơ đồ thuật toán phát hiện thay đổi nội dung

2.4.8. Cơ chế cảnh báo

Người quản trị hệ thống không phải lúc nào trực hệ thống 24/7 để có thể xem chương trình thông báo lỗi mà thông tin cảnh báo cần gửi đến thiết bị hoặc phương tiện gần gũi với người quản trị. Việc gửi thông tin cảnh báo sẽ được gửi một cách tự động bởi chương trình. Để tránh gửi quá nhiều cảnh báo, các thông tin sẽ được xử lý và được gửi đến thành phần xử lý lọc dữ liệu (sử dụng thuật toán C4.5), thành phần này sẽ tiến hành lọc để biết được thông tin nào cần thiết phải cảnh báo, lưu vào CSDL, xuất ra giao diện người dùng. Các thông tin nghiêm trọng cần cảnh báo sẽ được gửi qua E-mail và SMS theo thời gian thực. Cách thức xử lý trong cơ chế cảnh báo được minh họa như sau:

Một tập dữ liệu khoảng 10,000 bản ghi các thuộc tính (*HTTP, Redirect, IP, DNS, Timecheck, Lengthtext, Existstext, Notexiststext, Changestext,* và *Result*) đã được kiểm tra về độ chính xác của kết quả *defaced* hay không *defaced*. Tập dữ liệu này sẽ được dùng để huấn luyện và 5000 bản ghi sẽ dùng để làm tập dữ liệu kiểm thử. Trong quá trình giám sát khi nhận được các cảnh báo gởi về, nếu các cảnh báo có các dấu hiệu bất thường, các thông tin đó sẽ được bộ phận cảnh báo đưa vào thành phần xử lý lọc dữ liệu. Sau khi kiểm tra, bộ phận này sẽ quyết định mức độ nếu cần thiết sẽ gởi email và SMS để cảnh báo đến

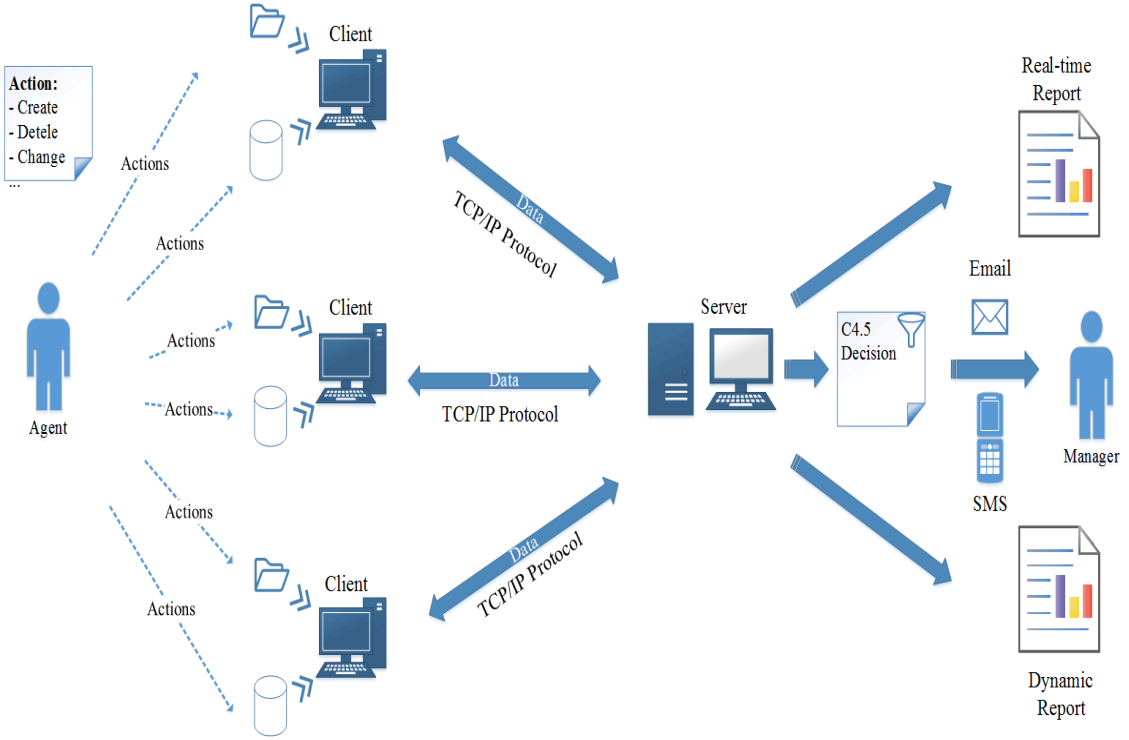
người dùng hoặc bỏ qua cảnh báo, chỉ ghi nhật ký. Việc sử dụng thành phần xử lý lọc dữ liệu (sử dụng thuật toán C4.5) sẽ giúp hạn chế bớt các cảnh báo trùng lắp.

3. PHƯƠNG PHÁP ĐỀ XUẤT

3.1. Kiến trúc hệ thống

Hệ thống được thiết kế bao gồm giao diện bộ giao tiếp (*User Interface-UI*) và hai phân hệ: *Anti Website Deface Server* (AWDS) và *Anti Website Deface Clients* (AWDC) như mô tả trong Hình 8, trong đó:

- *UI*: Thực hiện vai trò tiếp nhận thông tin từ người dùng chuyển tiếp đến hai phân hệ AWDS hoặc AWDC để hiển thị thông tin kết quả;
- *AWDS*: Giám sát tập trung, nằm trong hoặc cùng hệ thống mạng của hệ thống Webserver để tổng hợp thông tin và thực hiện cảnh báo khi có các thay đổi;
- *AWDC*: Được cài đặt trên các Webserver để thu thập thông tin, giám sát Website, Database và gửi về cho AWDS phân tích, phát hiện các dấu hiệu bất thường. AWDS và AWDC giao tiếp trực tiếp với nhau qua giao thức TCP (*Transmission Control Protocol*) và quá trình giao tiếp này được mã hóa bằng *Advanced Encryption Standard* (AES).

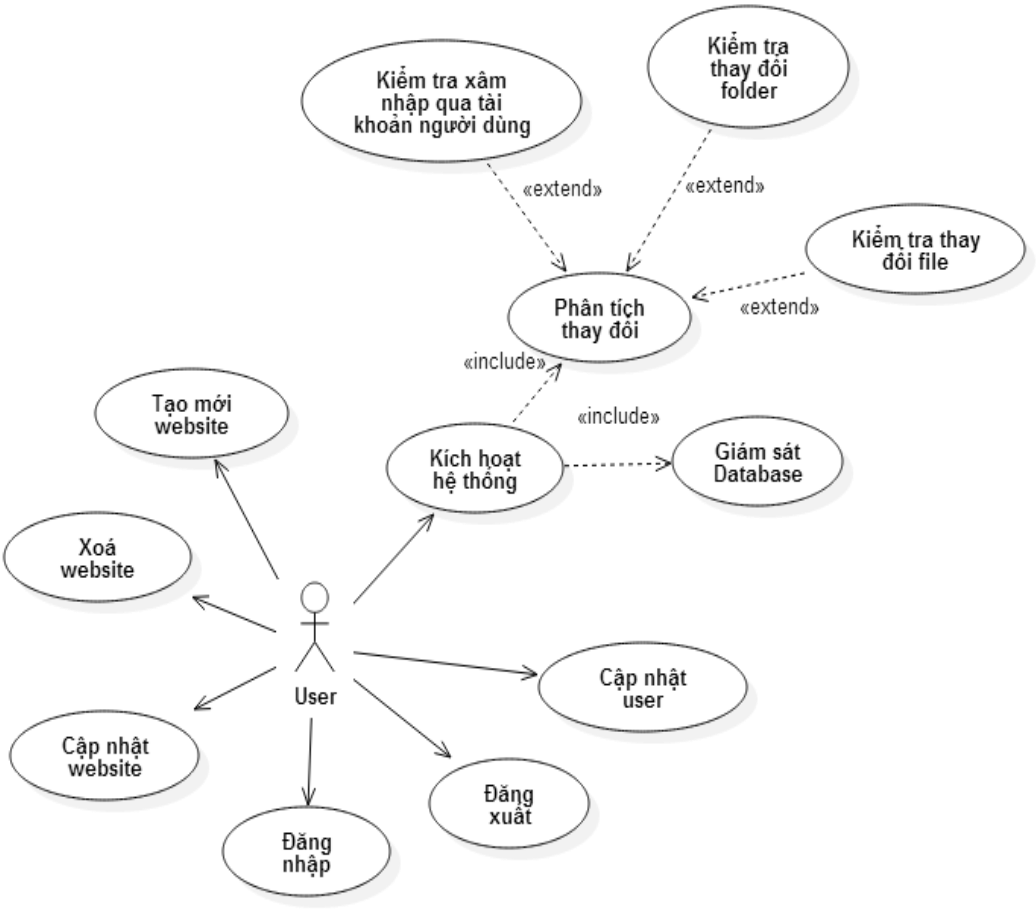


Hình 8. Mô hình kiến trúc hệ thống

3.2. Mô hình xử lý cơ chế phân tích và phát hiện sự thay đổi của Website

3.2.1. Mô hình xử lý AWDC

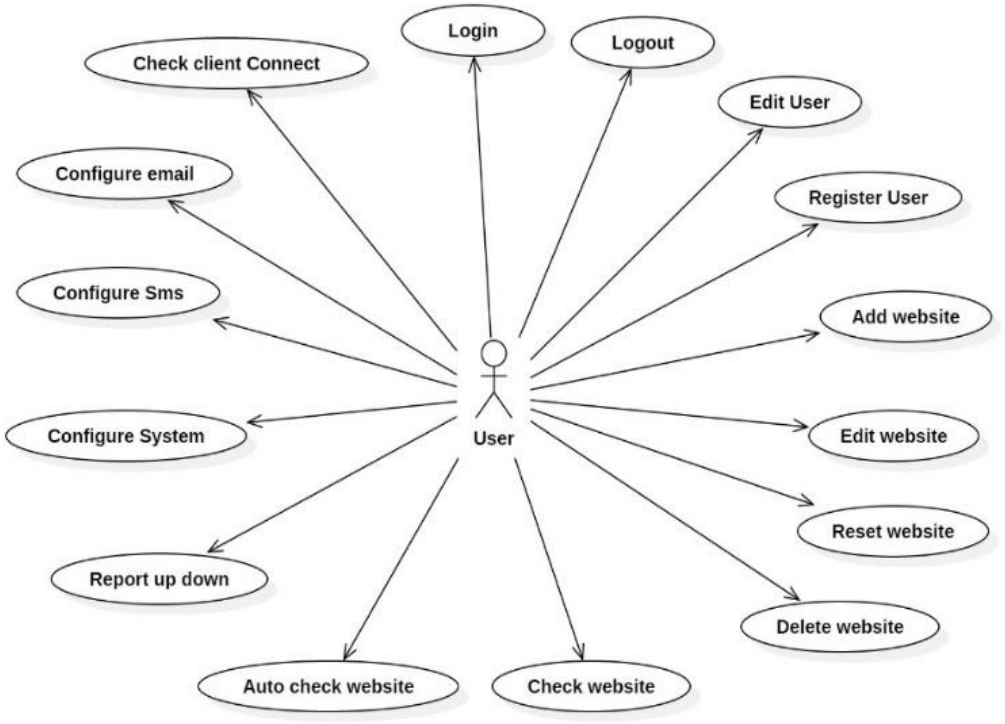
Mô hình Use-Case của AWDC được biểu diễn như Hình 9.



Hình 9. Mô hình Use-Case của AWDC

3.2.2. Mô hình xử lý AWDS

Mô hình Use-Case của AWDS được biểu diễn như Hình 10.



Hình 10. Mô hình Use-Case của AWDS

3.3. Chức năng của hệ thống

3.3.1. Phân hệ AWDS

Phân hệ AWDS thực hiện một số chức năng như: (1) Theo dõi trạng thái các máy chủ Web có cài đặt AWDC, tổng hợp báo cáo từ các AWDC gửi về để thông báo cho người giám sát bằng hình thức Email hoặc tin nhắn SMS khi phát hiện Website bị tấn công; (2) Thông báo cho các AWDC chuyển trạng thái Website bị tấn công sang trạng thái báo động; (3) Phát hiện những thay đổi hoặc ngắt kết nối giữa AWDC với AWDS; (4) Giám sát việc hoạt động/dừng của Website; và (5) Giám sát chính xác các thông số cấu hình của Website DNS, IP. Đối với các Website giám sát mà không có cài đặt được AWDC hệ thống tự phân tích và đưa ra các cảnh báo. Tuy nhiên chức năng giám sát chỉ thực hiện được các chức năng đã nêu tại các Mục 2.4.1; 2.4.2; và 2.4.7.

3.3.2. Phân hệ Anti AWDC

Phân hệ AWDC thực hiện một số chức năng như: (1) Phát hiện sự thay đổi về cấu trúc, thành phần số lượng và kích thước, tên các cấu phần của Website; (2) Phát hiện sự thay đổi về cấu trúc, thành phần, thuộc tính, đối tượng, tên các cấu phần của Database Website; và (3) Thêm người dùng trên hệ thống.

3.4. Môi trường phát triển ứng dụng

Hiện nay có một số ngôn ngữ lập trình như Java, .Net, ... với những ưu và nhược điểm khác nhau. Trong đó, công nghệ .Net là phổ biến trong xây dựng phần mềm nhờ tính bảo mật, chuyên nghiệp, khả năng xử lý trực tiếp và linh hoạt tại các máy tính khách, và khả năng tương thích tốt khi tích hợp với hệ thống khác. Vì vậy, tác giả chọn công nghệ .Net để cài đặt phần mềm "*Anti Website Deface*" với các yêu cầu chức năng hệ thống đã đặt ra.

4. KẾT QUẢ THỬ NGHIỆM

4.1. Danh sách các Website thử nghiệm

Các Website thử nghiệm đều thỏa mãn yêu cầu về công nghệ như Hệ điều hành Windows, Hệ quản trị Cơ sở dữ liệu Microsoft SQL Server 2008 (Bảng 1).

Bảng 1. Địa chỉ các Website thử nghiệm

STT	Thời gian	Địa chỉ Website thử nghiệm
1	16/7/2017 – 19/8/2017	http://hufi.edu.vn
2	16/7/2017 – 6/10/2017	http://fit.oktot.com
3	22/8/2017 – 15/9/2017	http://dnpu.edu.vn

4.2. Các kết quả thử nghiệm

Chúng tôi đã triển khai tám kịch bản thử nghiệm khác nhau và số lần thử nghiệm là 50. Các kết quả thử nghiệm được trình bày trong Bảng 2.

Bảng 2. Các kịch bản thử nghiệm

STT	Kịch bản thử nghiệm	Số lần thử nghiệm	Đạt yêu cầu	Không đạt yêu cầu	Tỷ lệ thành công
1	Thay đổi hình ảnh trên trang Web	50	50	00	100%
2	Đánh sập Website	50	50	00	100%
3	Thay đổi thư mục	50	50	00	100%
4	Thay đổi tính toàn vẹn của File/page	50	50	00	100%
5	Up Shell	50	50	00	100%
6	Thay đổi trên Database MS SQL Server 2008	50	50	00	100%
7	Redirect Website	50	50	00	100%
8	Thay đổi người dùng trên máy Webserver	50	50	00	100%

Kết quả cảnh báo sẽ được gửi đến các quản trị viên của ba Website qua hai hình thức là gửi Email và tin nhắn SMS. Sau đây là mô tả các kịch bản thử nghiệm:

- *Kịch bản 1*: *Thay đổi hình ảnh trên trang Web*: Các hình ảnh của Website được lưu trữ trên thư mục chứa mã nguồn Website. Giả sử vì một lý do nào đó các hình ảnh này bị thay đổi, yêu cầu phải có cảnh báo;

- *Kịch bản 2*: *Đánh sập Website*: Là các trường hợp Website không truy cập được, máy chủ bị tắt, bị tấn công từ chối dịch vụ, dịch vụ máy chủ Web bị tắt;

- *Kịch bản 3*: *Thay đổi thư mục*: Thư mục mã nguồn Website bị xóa, đổi tên;

- *Kịch bản 4*: *Thay đổi tính toàn vẹn của File/Page*: Các File/Page bị thêm, xóa, sửa, chèn nội dung;

- *Kịch bản 5*: *Up Shell*: Các tập tin được đưa trái phép lên thư mục mã nguồn Website;

- *Kịch bản 6*: *Thay đổi trên Database MS SQL Server 2008*: Thay đổi các thông tin trên cơ sở dữ liệu như mô tả trong Mục 2.4.4;

- *Kịch bản 7*: *Redirect Website*: Trang Web bị chuyển hướng truy cập do bị chèn mã nguồn, bị thay đổi IP đến trang Web giả mạo;

- *Kịch bản 8*: *Thay đổi người dùng trên máy Webserver*: Thêm, xóa tài khoản trên máy chủ Web.

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Nghiên cứu này đã trình bày một hướng tiếp cận bài toán hoàn toàn khác so với các phương pháp cũ trước đây. Phương pháp đề xuất mới này dựa trên sự kết hợp hàm băm MD5 và thuật toán đối sánh chuỗi và thuật toán cây quyết định đã mang lại kết quả rất khả quan trong việc giám sát, có thể phát hiện sự thay đổi giao diện, nội dung của Website một cách nhanh chóng, kịp thời theo thời gian thực. Kết quả thực nghiệm cho thấy các thay đổi như chèn thêm nội dung mới, xóa hay sửa nội dung cũ của mã nguồn, giao diện Website… đều được phát hiện và cảnh báo chính xác. Tuy nhiên, phương pháp đề xuất cũng mới chỉ phù hợp với các Website có tần suất thay đổi nội dung ít như Website của sở ban ngành, các trường đại học... Trong các nghiên cứu tiếp theo, chúng tôi sẽ cải tiến để ứng dụng có thể phù hợp với nhiều loại Website hơn nữa.

TÀI LIỆU THAM KHẢO

Amanda, A. (2003). *Surviving security: How to integrate people, process, and technology*. Florida, USA: CRC Press.

Boyer, R. S., & Moore, J. S. (1977). A fast string searching algorithm. *Magazine Communications of the ACM, 20*(10), 762-772.

British Broadcasting Company (BBC). (2012). *Chinese websites 'defaced in Anonymous attack'*. Retrieved from http://www.bbc.co.uk/news/technology-17623939.

British Broadcasting Company (BBC). (2014). *Nottinghamshire police Website hacked by Anonghost*. Retrieved from http://www.bbc.com/news/uk-england-nottinghamshire-29951605.

Cashin. E. L. (2000). *Integerit file verification system*. Retrieved from http://integrit. sourceforge.net.

Charles, P., Shari, L. P., & Jonathan, M. (2015). *Security in computing*. New Jersey, USA: Prentice Hall Press.

Cục An toàn Thông tin. (2016). *Bản tin An toàn thông tin tháng 7 năm 2016*. Hà Nội, Việt Nam: Bộ Thông tin và Truyền thông.

Davanzo, G., Medvet, E., & Bartoli, A. (2010). A comparative study of anomaly detection techniques in Website defacement detection. *IFIP International Federation for Information Processing, 278*(1), 711-716.

David, D., Manos, A., Paul, V., Tatuya, J., & Wenke, L. (2008). *Increased DNS forgery resistance through 0x20-bit encoding: Security via leet queries*. Paper presented at The 15th ACM Conference on Computer and Communications Security, USA.

Dương, A. Đ., & Trần, M. T. (2005). *Mã hoá và ứng dụng.* TP. Hồ Chí Minh, Việt Nam: NXB. Đại học Quốc gia TP. Hồ Chí Minh.

Gaurav, R., Newley, P., & Jason, N. (2015). *Malaysia airlines Website hacked by group calling itself "Cyber Caliphate"*. New York, USA: Wall Street Journal. Retrieved from https://www.wsj.com/articles/malaysia-airlines-website-hacked-by-group-calling-itself-cyber-caliphate-1422238358.

Hiệp hội An toàn thông tin Việt Nam (VNISA). (2017). *Tài liệu báo cáo ngày An toàn thông tin Việt Nam 2017*. Retrieved from https://vnisa.org.vn/thu-vien/tai-lieu-hoi-thao/tai-lieu-bao-cao-ngay-an-toan-thong-tin-viet-nam-2017.html.

OWASP. (2017). *Top 10 most critical web application security risks*. Retrieved from https://www.owasp.org/index.php/Category:OWASP_top_ten_project.

Piyush, G., & Sandeep, K. (2014). A comparative analysis of SHA and MD5. *International Journal of Computer Science and Information Technologies, 5*(3), 4492-4495.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* Massachusetts, USA: Morgan Kaufmann Publishers.

Ramniwas, K., Nikhil, K. S., & Deepak, S. T. (2014). A novel approach to detect Webpage tampering. *International Journal of Computer Science and Information Technologies, 5*(3), 4604-4607.

Rashmi, K. V., & Shahzia, S. (2015). Implementation of Web defacement detection technique. *International Journal of Innovations in Engineering and Technology, 6*(1), 134-140.

Richard, B. (2003). *C# network programming.* California, USA: John Wiley & Sons Publishing.

Rivest, R. (1992). *The MD5 message-digest algorithm, RFC 1321.* Retrieved from https://www.ietf.org/rfc/rfc1321.txt.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal, 27*, 379-423.

Shar, L., & Tan, H. B. (2013). Defeating SQL injection. *Computer, 46*(3), 68-77.

Stalling, W. (1999). *Cryptography and network security: Principles and practice.* New Jersey, USA: Prentice Hall Publishing.

Tushar, K., Vineet, R., & Vivek, R. (2011). Implementing a Web browser with Web defacement detection techniques. *World of Computer Science and Information Technology Journal, 1*(7), 307-310.

Tushar, K., Vineet, R., & Vivek, R. (2012). Implementation of an efficient Web defacement technique and spotting exact defacement location using Diff algorithm. *International Journal of Emerging Technology and Advanced Engineering, 2*(3), 252-256.

Xiang, Y. L., & Hongtao, L. (2008). *Fragile watermarking schemes for tamper-proof Webpages*. Paper presented at The WASE International Conference on Information Engineering, China.