HỆ THỐNG PHÁT HIỆN TẤN CÔNG BOTNET SỬ DỤNG WEB PROXY VÀ CONVOLUTIONAL NEURAL NETWORK

Trần Đắc Tốt[a*], Phạm Tuấn Khiêm[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@hufi.edu.vn

Lịch sử bài báo
Nhận ngày 10 tháng 02 năm 2020
Chỉnh sửa ngày 20 tháng 5 năm 2020 | Chấp nhận đăng ngày 15 tháng 6 năm 2020

Tóm tắt

Botnet đang ngày càng trở thành những mối đe dọa nguy hiểm nhất trong lĩnh vực an ninh mạng, nhiều hướng tiếp cận khác nhau để phát hiện tấn công bằng botnet đã được nghiên cứu. Tuy nhiên, dù bất kì hướng tiếp cận nào được sử dụng, sự tiến hóa về bản chất của botnet cùng tập các quy luật được định nghĩa sẵn để phát hiện ra botnet có thể ảnh hưởng đến hiệu suất của hệ thống phát hiện botnet. Trong bài báo này, chúng tôi đề xuất một họ kiến trúc tổng quát sử dụng thuộc nhóm Convolutional Neural Network để biến đổi từ đặc trưng thô do các công cụ ghi nhận và phân tích network flow cung cấp thành đặc trưng cấp cao hơn, từ đó tiến hành phân lớp (nhị phân) để đánh giá một flow tương ứng với tình trạng bị botnet tấn công hay không. Chúng tôi thử nghiệm trên tập CTU-13 với các cấu hình khác nhau của convolutional neural network để đánh giá tiềm năng dùng deep learning với convolutional neural network vào bài toán phát hiện botnet. Đặc biệt là đề xuất hệ thống phát hiện Botnet sử dụng Web proxy. Đây là một kỹ thuật giúp triển khai hệ thống phát hiện botnet với chi phí thấp mang lại hiệu quả cao.

Từ khóa: AntiBotDDOS; Botnet; Convolutional neural network; Tấn công từ chối dịch vụ; Web proxy.

DOI: http://dx.doi.org/10.37569/DalatUniversity.10.3.652(2020)

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

DETECTING WEB-BASED BOTNETS USING A WEB PROXY AND A CONVOLUTIONAL NEURAL NETWORK

Tran Dac Tot[a*], Pham Tuan Khiem[a], Pham Nguyen Huy Phuong[a]

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

Article history
Received: February 10th, 2020
Received in revised form: May 20th, 2020 | Accepted: June 15th, 2020

Abstract

Botnets are increasingly becoming the most dangerous threats in the field of network security, and many different approaches to detecting attacks from botnets have been studied. Whatever approach is used, the evolution of the botnet's nature and the set of defined rules for detecting botnets can affect the performance of botnet detection systems. In this paper, we propose a general family of architectures that uses a convolutional neural network group to transform the raw characteristics provided by network flow recording and analysis tools into higher-level features, then conducts a (binary) class to assess whether a flow corresponds to a botnet attack. We experimented on the CTU-13 dataset using different configurations of the convolutional neural network to evaluate the potential of deep learning on the botnet detection problem. In particular, we propose a botnet detection system that uses a web proxy. This technique can be helpful in implementing a low-cost, but highly effective botnet detection system.

Keywords: AntiBotDDOS; Botnet; Botnet detection; Convolutional Neural Network; Web proxy.

DOI: http://dx.doi.org/10.37569/DalatUniversity.10.3.652(2020)

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

1. ĐẶT VẤN ĐỀ

Botnet–một mạng các máy chủ bị xâm hại dưới sự điều khiển từ xa của Botmaster–có tiềm năng thực thi ở quy mô lớn những tác vụ độc hại khác nhau. Một số ví dụ điển hình là những cuộc tấn công phân tán từ chối dịch vụ (DDoS), ăn cắp thông tin cá nhân và spam. Cùng với sự phát triển nhanh chóng các công nghệ máy tính và tốc độ truyền tải Internet, botnet đã phát triển mạnh mẽ kể từ đầu năm 2000. Sự phát triển này đòi hỏi những hệ thống phát hiện botnet phải thích ứng với việc nâng cấp và mở ra nhu cầu về hệ thống phát hiện botnet dựa trên khám phá các mẫu cấu trúc một cách tự động. Trong lĩnh vực này, kĩ thuật gom nhóm và phân loại–được sử dụng trong tự động hóa việc phân tích lưu lượng truyền tải–yêu cầu mạng truyền tải phải được biểu diễn một cách có ý nghĩa để có thể cho phép việc nhận dạng mẫu. Vì vậy, một thành phần quan trọng cho những hệ thống như thế này chính là rút trích những đặc trưng (thuộc tính) từ traffic trên đường mạng.

Những gói dữ liệu bao gồm hai phần chính, tiêu đề gói (*header*) và nội dung truyền tải (*payload*). Phần tiêu đề lưu giữ thông tin điều khiển các giao thức trong khi phần payload lưu giữ thông tin ứng dụng được sử dụng trong mạng. Vì lý do đó, việc phân tích network traffic có thể được thực hiện theo từng gói (*per-packet*, dùng một trong hai phần nêu trên) hoặc theo từng luồng (chỉ sử dụng những gói tiêu đề tổng hợp).

Một số công trình thực hiện đánh giá cả hai hướng tiếp cận phát hiện botnet dựa trên gói payload và luồng (Haddadi, Le, Porter, & Zincir-Heywood, 2015). Nhận thấy rằng tập các đặc trưng đã được dùng và tầm quan trọng đã được kiểm chứng của phương pháp rút trích đặc trưng trong các hướng tiếp cận trên (Haddadi & Zincir-Heywood, 2014). Một số công trình đánh giá hai hướng tiếp cận trên một luồng (*one flow based*) dựa trên một gói payload (*one packet payload based*) (Haddadi & ctg., 2015). Vì những botnet gần đây có xu hướng sử dụng mã hóa để che giấu thông tin và phương thức của chúng khỏi những hệ thống phát hiện botnet nên hệ thống phát hiện botnet dựa trên một luồng (*one flow based detection system*) chiếm ưu thế hơn hệ thống dựa trên gói dữ liệu (*packet-based system*) vì có thể được dùng để phát hiện tấn công ngay cả khi nội dung traffic bị mã hóa.

Do tập luật phải được định nghĩa sẵn dựa trên tri thức sẵn có, kết quả khi phân tích, so sánh, và đánh giá hiệu năng của các hệ thống phát hiện dựa trên luật có thể bị ảnh hưởng bởi việc chọn tập dữ liệu và nâng cấp tập luật. Snort (Snort, n.d.) và BotHunter (Gu, Porras, Yegneswaran, Fong, & Lee, 2007) là các hệ thống phát hiện dựa trên luật thường được dùng để so sánh và đánh giá. Snort là một hệ thống ngăn ngừa và phát hiện những xâm nhập thông dụng (IDS/IPS). Đây là công cụ mã nguồn mở nên tập luật của công cụ này có thể được sửa đổi một cách dễ dàng. BotHunter cũng là một hệ thống mở khác, tận dụng module đánh giá của Snort và sửa đổi tập luật của Snort để dành riêng biệt cho việc phát hiện botnet.

Phát hiện botnet ngày càng trở nên khó khăn hơn khi chúng sử dụng các giao thức thông dụng như HTTP, các cấu trúc phân tán và các kĩ thuật như mã hóa. Nhiều hệ thống phát hiện botnet đã được đề xuất nhằm đối phó với những sự thay đổi trên. Đứng từ góc

độ dữ liệu, một số các kĩ thuật tập trung chủ yếu vào phân tích mã nguồn và file thực thi của malware, trong khi đó các kĩ thuật khác lại sử dụng các dữ liệu đến từ máy chủ lưu trữ (*host*) và dữ liệu mạng (*network*). Ngoài ra, phân tích lưu lượng dựa trên luật và phát hiện dữ liệu bất thường là một trong những hướng tiếp cận được sử dụng nhiều nhất. Trong hướng tiếp cận dựa trên luật và sự bất thường trong dữ liệu, các luật và sự bất thường có thể xác định thông qua phát tích dữ liệu bằng việc đánh giá bởi chuyên gia hoặc được làm một cái tự động bằng hệ thống sử dụng các thuật toán máy học.

Gu và ctg. (2007) đã phát triển một hệ thống như thế với tên BotHunter, kết hợp với cảnh báo Snort IDS để phát hiện botnet. Sự kết hợp này dựa vào việc các botnet có những phần hoạt động giống nhau trong vòng đời tồn tại của chúng. Phân tích payload (*Payload analysis*) cũng là một phần của hệ thống BotHunter này. Wurzinger, Bilge, Holz, Goebel, Kruegel, và Kirda (2009) đã đề ra một hướng tiếp cận để phát hiện botnet dựa trên sự tương quan của các lệnh (*command*) và hồi đáp (*response*) trong dữ liệu truy vết được ghi lại trong quá trình giao tiếp trong mạng. Các đặc tính trong traffic như số lượng các bytes không phải là ASCII trong *payload* được phân tích để xác định tính chất của bot. Celik, Raghuram, Kesidis, và Miller (2011) đã đề xuất một hệ thống phát hiện hoạt động C&C của botnet (*flow-based botnet C&C activity*) sử dụng header của các gói. Họ đã điều tra về sự ảnh hưởng của hiệu chuẩn của đặc tính luồng dựa trên thời gian (*time-based flow features*). Wang, Huang, và Lin (2011) đề ra một phương pháp nhận diện các botnet HTTP và IRC dựa trên các mẫu hành vi (*behavioral pattern*) của chúng. Trong hướng tiếp cận này, họ đã phân tích đặc tính của các truy vấn DNS (như số lượng truy xuất DNS thất bại) và luồng TCP để phát hiện các tên miền và địa chỉ IP độc hại. Zhao, Traore, Ghorbani, Sayed, Saad, và Lu (2012) phát minh ra hệ thống phát hiện botnet dựa trên luồng phân đoạn (*flow intervals*). Các đặc tính luồng của các gói tin lưu lượng dữ liệu được sử dụng cùng với một số thuật toán Machine Learning tập trung vào P2P botnet như Waledac.

Trong phần tiếp theo của bài báo này, Phần 2 trình bày nội dung chính của Hệ thống phát hiện botnet (AntiBotDDOS) với web proxy và chúng tôi đề xuất một họ kiến trúc tổng quát sử dụng thuộc nhóm Convolutional Neural Network để biến đổi từ đặc trưng thô do các công cụ ghi nhận và phân tích network flow cung cấp thành đặc trưng cấp cao hơn, từ đó tiến hành phân lớp (nhị phân) để đánh giá một flow tương ứng với tình trạng bị botnet tấn công hay không. Phần 3 là phần thực nghiệm. Phần 4 là là phần kết quả và thảo luận. Phần 5 là phần kết luận và hướng nghiên cứu tiếp theo.

2. HỆ THỐNG PHÁT HIỆN BOTNET VỚI WEB PROXY VÀ CNN

Hệ thống phát hiện botnet (AntiBotDDOS) với web proxy được xây dựng theo mô hình application proxy có bổ sung thêm một số tính năng để giảm thiểu tấn công bằng DDoS đến web server như:

- Khả năng tự kiểm tra, phân biệt người dùng và PC-Bot:
    - Challenge HTTP;

- Challenge Java;
- Phát hiện fake IP.

- Khả năng tự học, tự cấu hình để điều chỉnh các thông số nhằm tối ưu hoạt động hệ thống.
- Khả năng xác thực người dùng thông qua cơ chế Captcha.

Ngoài ra, về việc xác định ngưỡng hoạt động của AntiBotDDOS, sẽ có hai ngưỡng thiết lập như sau:

- Thiết lập bị động: Web server được cấu hình một ngưỡng hoạt động mà ở đó các tham số xử lý số lượng HTTP request/response được thiết lập cố định.
- Cơ chế chủ động: Hệ thống tự động học (CNN) và thiết lập ngưỡng xử lý HTTP request/response.

2.1. Convolutional Neural Network trong phát hiện tấn công Botnet

2.1.1. Tính đa dạng trong nguồn dữ liệu thô

Công cụ phát sinh luồng (*Flow Generation*) tóm tắt thông tin traffic sử dụng các header của các packet trong mạng. Các công cụ này thu thập thông tin các packer với các đặc tính chung, ví dụ như địa chỉ IP, port, nhóm các thông tin này lại và thực hiện một số tính toán thống kê, ví dụ như số lượng packer trong mỗi flow…

Trong RFC 2722, một traffic flow được xem là tương ứng với một kết nối liên kết với một nhóm tài nguyên cụ thể. Phương pháp chung thường được sử dụng để xác định một traffic flow là sử dụng tổ hợp của năm thuộc tính từ header của packet, bao gồm cả header ở tầng network và tầng transport trong TCP/IP network protocol stack. Các thông tin này là: Địa chỉ IP nguồn, địa chỉ IP đích, port nguồn, port đích, và giao thức.

Trên thực tế, có nhiều công cụ để thu thập (*collect*), xuất thông tin (*export*) và giúp phân tích (*analyse*) traffic mạng. Một số công cụ hỗ trợ cả chế độ online (ghi nhận trực tiếp dữ liệu từ hoạt động thực tế của mạng) hay offline (phân tích dữ liệu đã được ghi nhận–precaptured từ file).

Tùy theo từng công cụ hay giải pháp được sử dụng để thu thập và rút trích thông tin thuộc tính cho các flow trên mạng, tập đặc trưng (*feature set*) có thể rất khác nhau.

Một số công cụ trích xuất thông tin từ luồng bao gồm:

- Maji (Maji, n.d.) là công cụ mã nguồn mở cài đặt IPFIX, do nhóm nghiên cứu WAND tại Đại học Waikato, New Zealand hỗ trợ. Công cụ này trích xuất

luồng đơn hướng từ traffic thời gian thực với giao tiếp Packet CAPture (PCAP) và hầu hết các định dạng file trace phổ biến.

- YAF (YAF, n.d.) là công cụ trích xuất thông tin luồng hai hướng do nhóm NetSA tại CERT thiết kế. Công cụ này thu thập và trích xuất các luồng dựa trên IPFIX. Tương tự với Maji, YAF có thể xử lý dữ liệu packet từ các file đã ghi lại traffic hay ghi nhận trực tiếp từ môi trường mạng.

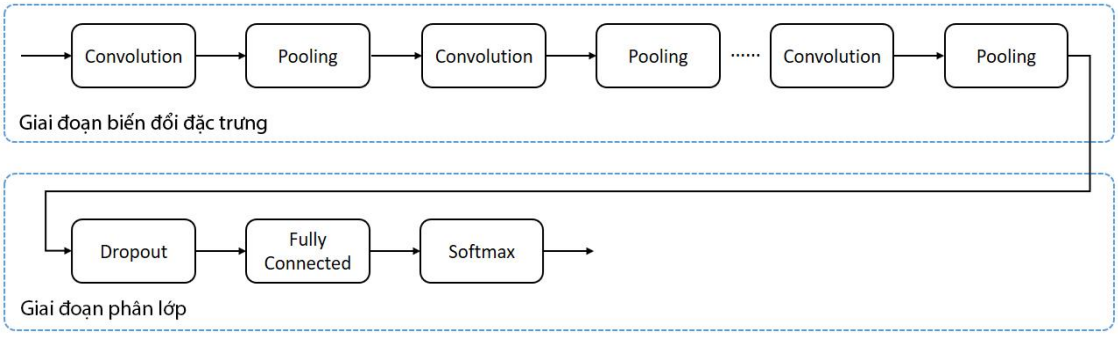
- Softflowd (Softflowd, n.d.) là công cụ nhỏ gọn cho phép trích xuất luồng đơn hướng và hỗ trợ các phiên bản khác nhau của NetFlow. Công cụ này trích xuất dữ liệu NetFlow sử dụng dữ liệu đã được ghi lại vào file hoặc ghi nhận thời gian thực từ môi trường mạng.

- Tranalyzer (Tranalyzer, n.d.) là công cụ nhỏ gọn cho phép trích xuất luồng đơn hướng và hỗ trợ phiên bản mở rộng của tập đặc trưng NetFlow. Công cụ này cũng hỗ trợ xử lý dữ liệu được ghi lại trong tập tin hoặc xử lý thời gian thực từ traffic của mạng.

- Netmate (Netmate, n.d.) là công cụ trích xuất và phân tích luồng hai hướng. Công cụ này cũng hỗ trợ xử lý dữ liệu được ghi lại trong tập tin hoặc xử lý thời gian thực từ traffic của mạng.

Do tập hợp các đặc trưng được mỗi công cụ cung cấp có thể khác nhau cả về số lượng và ý nghĩa của từng thành phần trong vector đặc trưng, trong phạm vi tìm hiểu khả năng ứng dụng Convolutional Neural Network vào phân tích và đánh giá để phát hiện tấn công botnet, chúng tôi đề xuất một họ kiến trúc tổng quát sử dụng thuộc nhóm Convolutional Neural Network để biến đổi từ đặc trưng thô do các công cụ ghi nhận và phân tích network flow cung cấp thành đặc trưng cấp cao hơn, từ đó tiến hành phân lớp (nhị phân) để đánh giá một flow tương ứng với tình trạng bị botnet tấn công hay không.

2.1.2. Mô hình đề xuất



Hình 1. Mô hình tổng quan về kiến trúc CNN được khảo sát

Hình 1 thể hiện mô hình tổng quan của kiến trúc CNN được chúng tôi chọn khảo sát. Trong kiến trúc này có hai thành phần chính:

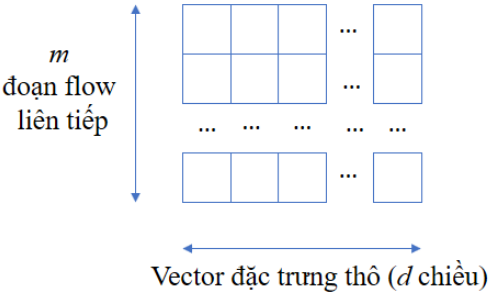
- Giai đoạn biến đổi đặc trưng: Với mục tiêu để biến đổi và tạo ra đặc trưng biểu diễn cấp cao từ tập thuộc tính thô của flow do các công cụ ghi nhận và phân tích traffic mạng cung cấp. Dữ liệu thô của mỗi flow được biến đổi qua nhiều layer để tạo ra đặc trưng cấp cao, chuẩn bị cho giai đoạn phân lớp để đánh giá flow có phải bị botnet tấn công hay không.

- Giai đoạn phân lớp: Sử dụng đặc trưng cấp cao của flow được tạo ra trong giai đoạn biến đổi đặc trưng, các bước xử lý ở giai đoạn phân lớp giúp đánh giá flow có phải bị botnet tấn công hay không.

Chi tiết về cách xây dựng giai đoạn biến đổi đặc trưng và giai đoạn phân lớp được trình bày trong phần tiếp theo.

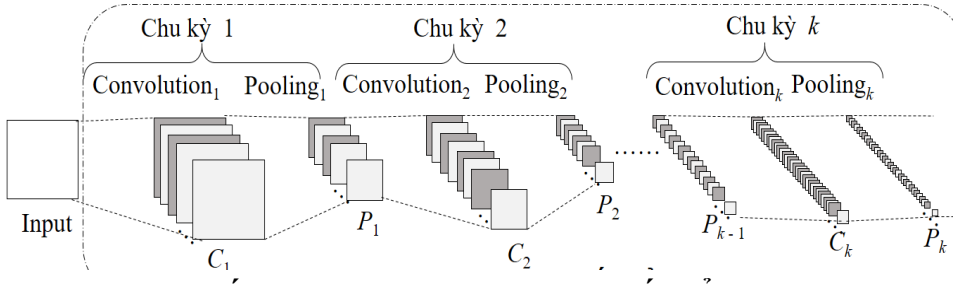
2.1.3. Giai đoạn biến đổi đặc trưng

Dữ liệu đầu vào bao gồm *m* đoạn flow liên tiếp, mỗi đoạn flow được biểu diễn bằng vector gồm *d* chiều chính là đặc trưng thô được cung cấp từ công cụ phân tích mạng (Hình 2). Vector đầu vào được tổ chức dưới dạng hai chiều, gồm *m* dòng (tương ứng với *m* đoạn flow liên tiếp) và *d* cột (biểu diễn *d* thành phần trong vector đặc trưng thô của công cụ phân tích mạng). Ý tưởng chính của chúng tôi là tận dụng cách biểu diễn 2D thường gặp của dữ liệu hình ảnh trong các công trình về Convolutional Neural Network trên ảnh vào bài toán phân tích và phát hiện botnet.



Hình 2. Biểu diễn vector đầu vào

Thành phần xử lý chính của kiến trúc bao gồm *K* chu kỳ, mỗi chu kỳ gồm một *layer convolution* và một *layer pooling* (Hình 3).



Hình 3. Cấu trúc chung của giai đoạn biến đổi đặc trưng

Mỗi layer convolution sử dụng một filter bank bao gồm các filter có kích thước bằng nhau, nhằm tạo ra các kết quả sau khi được lọc khác nhau từ vector đặc trưng đầu vào. Do kết quả đầu ra tại mỗi node trong layer convolution là tổ hợp tuyến tính của các giá trị đầu vào của layer này, chúng tôi luôn sử dụng thêm layer biến đổi phi tuyến ReLU (*Rectified Linear Unit*) ngay sau layer convolution để bổ sung tính chất phi tuyến vào neural network.

Sau layer convolution, chúng tôi bổ sung layer pooling bằng max pooling với kích thước kernel là 2 × 1. Việc sử dụng layer pooling giúp cho các đặc tính nổi bật rút ra được từ layer convolution có khả năng xuất hiện linh hoạt trong biên độ nhất định. Kích thước kernel 2 × 1 cho phép liên kết để tổng hợp đặc trưng từ các đoạn flow.

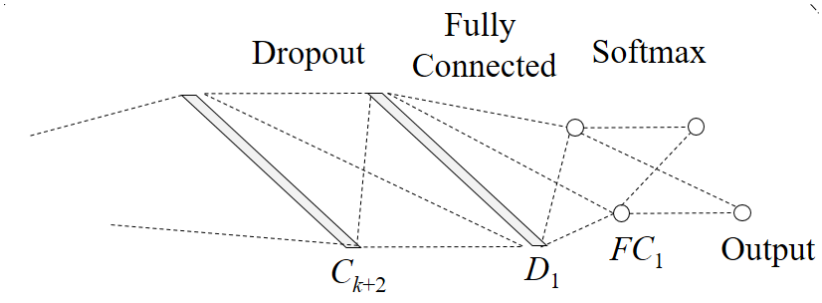
Chúng tôi định nghĩa các tham số cho layer convolution thứ i ($1 \le i \le K$) bao gồm:

- $kernel_i$ là độ cao của kernel được sử dụng trong các filter của filter bank. Như vậy, tất cả filter được dùng trong layer convolution thứ i đều có kích thước là $kernel_i \times d$. Nói cách khác, chúng tôi muốn tạo ra khả năng tương tác trên thông tin đặc trưng của mỗi nhóm gồm $kernel_i$ dòng liên tiếp trong vector feature map.

- n_i^C là số lượng filter trong filter bank.

Sau mỗi chu kỳ, do việc sử dụng layer pooling, độ dài của mỗi đặc trưng được học khi áp dụng một filter cụ thể trong filter bank sẽ giảm đi 50%. Do đó, để có thể giữ lại những thông tin từ đặc trưng từ cấp thấp hơn, chúng tôi sử dụng số lượng filter trong filter bank tăng dần qua mỗi chu kỳ xử lý: $n_i^C < n_{i+1}^C$ với $1 \le i < K$. Trong thử nghiệm, chúng tôi chọn giá trị độ cao $3 \le kernel_i \le 5$.

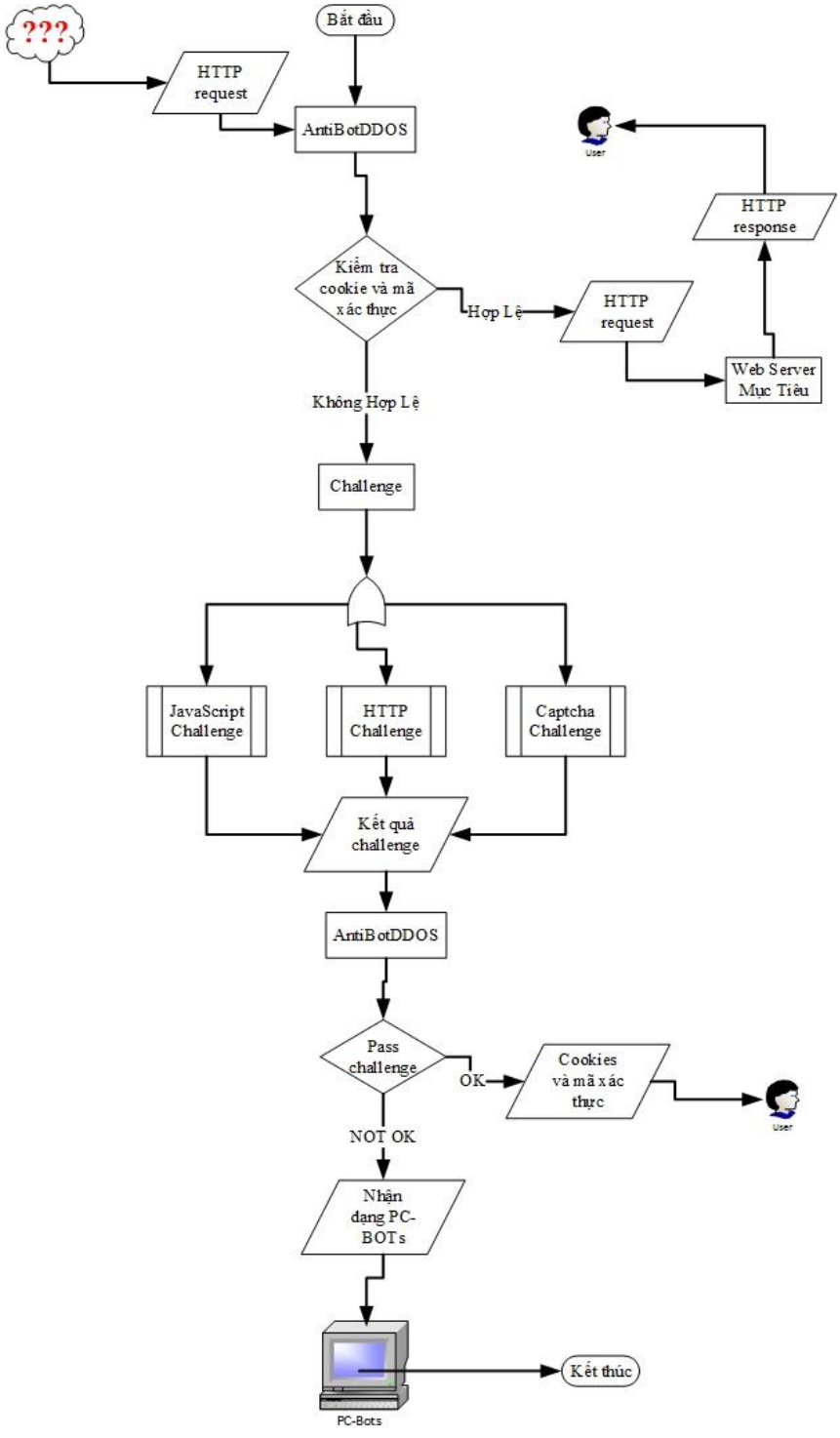
2.1.4. Giai đoạn phân lớp



Hình 4. Cấu trúc chung của giai đoạn phân lớp

Hình 4 trình bày cấu trúc chung của giai đoạn phân lớp. Để hạn chế việc quá khớp khi huấn luyện neural network, chúng tôi áp dụng kỹ thuật Dropout. Ý tưởng chính của Dropout là một số node (cùng với các cạnh nối với node này) sẽ được chọn ngẫu nhiên để bỏ qua với xác suất nhất định khi huấn luyện neural network.

2.2. Cơ chế xác thực người dùng



***Hình 5.* Cơ chế xác thực người dùng của AntiBotDDOS**

Trong mô hình sử dụng Web Server Reverse Proxy, toàn bộ truy cập đến web server mục tiêu sẽ được ứng dụng AntiBotDDOS (Hình 5) kiểm tra theo cơ chế:

- Toàn bộ HTTP request sẽ được AntiBotDDOS tiếp nhận.

- Nếu HTTP request có chứa cookies và mã xác thực hợp lệ thì request này sẽ được chuyển đến web server mục tiêu và HTTP response sẽ được trả về người dùng.

- Trong trường hợp HTTP request không hợp lệ, AntiBotDDOS sẽ tiến hành challenge theo một trong ba cách: HTTP challenge, JavaScript challenge, và Captcha challenge… (số lượng các module challenge có thể mở rộng theo từng phiên bản của AntiBotDDOS).

- Nếu vượt qua được challenge thì trình duyệt sẽ nhận được cookies và mã xác thực. Ngược lại, AntiBotDDOS sẽ ghi nhận HTTP request đó được gửi từ PC-Bots và loại bỏ HTTP request này.

2.2.1. HTTP Challenge

Đối với giao thức HTTP được quy định tại RFC 2616, thì code 3xx được sử dụng trong việc chuyển hướng truy cập (*redirection*). Khi HTTP request được client gửi tới AntiBotDDOS. AntiBotDDOS sẽ gửi trả về cho client HTTP return code 302. Nếu client là trình duyệt, khi nhận được HTTP return code 302 sẽ chuyển hướng truy cập đến một URL do AntiBotDDOS chỉ định và khi truy cập vào URL này, AntiBotDDOS sẽ gửi tiếp cho client một đoạn JavaScript để tạo cookies và mã xác thực hợp lệ.

Ngược lại, nếu client không phải là trình duyệt, HTTP return code 302 sẽ không được xử lý đúng quy trình. Đối với các PC-bot, các HTTP request được gửi trực tiếp đến webserver mà không cần thông qua trình duyệt, các hành vi này được lập trình sẵn và sẽ không đủ thông minh để xử lý các code return 302, hoặc không tạo ra được các cookies hợp lệ để truy cập vào tài nguyên.

2.2.2. JavaScript Challenge

Trong tình huống bị tấn công DDoS có dấu hiệu tham gia của các mạng botnets thì tính năng HTTP challenge được sử dụng đầu tiên nhằm hạn chế và phân loại các yêu cầu xuất phát từ các bots. Trong một số trường hợp các C&C servers giải mã được cookies được tạo ra cho HTTP challenge thì AntibotDDOS sẽ bật chế độ Java challenge.

Trong chế độ Java Challenge khi request được gửi tới AntiBotDDOS và được trả về một đoạn mã JavaScript (khác với JavaScript trong HTTP challenge) đã được xáo trộn để tăng độ phức tạp. Trong đó đoạn mã này có chứa một key mã hóa. Nếu client là trình duyệt có bật chức năng thực thi JavaScript thì nó sẽ thực thi đoạn mã JavaScript này để tạo cookies và mã xác thực hợp lệ. Sau đó, trình duyệt sẽ dùng thông tin này để gửi lại AntiBotDDOS. Trong trường hợp ngược lại Bot sẽ không xử lý challenge này và không thể tạo Cookies hợp lệ.

2.2.3. Captcha Challenge

Khi HTTP request được gửi tới AntiBotDDOS. AntiBotDDOS trả về một trang web có chứa form CAPTCHA và câu trả lời (Hình 6). Hai thông tin này kết hợp với secrect key (một đoạn mã được cấu hình sẵn trong AntiBotDDOS) để tạo thành mã kiểm tra. Khi client nhập câu trả lời và submit kết quả, AntiBotDDOS dùng kết quả này kết hợp với secrect key để tạo ra mã trả lời. Nếu mã trả lời và mã kiểm tra giống nhau, AntiBotDDOS sẽ trả về cho client một trang web có chứa JavaScript và yêu cầu client tạo cookies và mã xác thực hợp lệ.

Trong các phương thức chống DDoS thì đây được xem là biện pháp tương đối hữu hiệu để ngăn chặn tấn công từ các botnet. Tuy nhiên việc bật chế độ Captcha sẽ làm ảnh hưởng đến người sử dụng hợp pháp thì hình thành thao tác xác thực mỗi khi kết nối đến Webserver. Phương án sử dụng Captcha được sử dụng là biện pháp cuối cùng, khi tất cả các challenges khác đều không có hiệu quả hoặc năng lực của Proxy Server là AntibotDDOS không đủ để xử lý các request từ mạng botnet quá lớn.

***Hình 6.* Giao diện Captcha Challenge của AntiBotDDOS**

3. THỬ NGHIỆM

3.1. Bộ dữ liệu CTU-13

CTU-13 là một bộ dữ liệu lưu lượng botnet được ghi nhận tại Đại học CTU, Cộng hòa Séc, công bố năm 2011. Trong tập dữ liệu này, traffic được gán nhãn có botnet, hoạt động bình thường và background traffic. Mục tiêu của tập dữ liệu này là cung cấp dataset thực tế có kích thước lớn, trong đó traffic được ghi nhận bao gồm traffic có botnet tấn công hòa lẫn với traffic thông thường và traffic nền.

Bộ dữ liệu CTU-13 bao gồm 13 lần ghi nhận (gọi là kịch bản–*scenario*) của các mẫu botnet khác nhau. Trên mỗi kịch bản, nhóm tác giả của CTU-13 cho thực thi một malware cụ thể, sử dụng nhiều giao thức và thực hiện các hành động khác nhau (Bảng 1).

Bảng 1. Trình bày các đặc tính của 13 kịch bản botnet

Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	√	√	√							
2	√	√	√							
3	√			√				√		
4	√				√			√		UDP and ICMP DDoS.
5		√		√					√	Scan web proxies.
6				√						Proprietary C&C. RDP.
7									√	Chinese hosts.
8				√						Proprietary C&C. Net-BIOS, STUN.
9	√	√	√	√						
10	√				√			√		UDP DDoS.
11	√				√			√		ICMP DDoS.
12							√			Synchrinization.
13		√		√					√	Captcha. Web mail.

Ghi chú: CF: ClickFraud; PS: Port Scan; FF: FastFlux; US: do nhóm tác giả CTU-13 tạo ra và điều khiển.

Mỗi kịch bản được ghi lại trong một tệp tin pcap có chứa tất cả các gói tin trong ba loại lưu lượng truy cập. Các tập tin pcap này đã được xử lý để có được các loại thông tin khác, chẳng hạn như NetFlows, WebLogs… Các phân tích đầu tiên của bộ dữ liệu CTU-13 được mô tả và công bố trong bài báo của nhóm tác giả (Garcia, Grill, Stiborek, Zunino, 2014) sử dụng NetFlows theo chiều hướng để đại diện cho lưu lượng truy cập và gán nhãn. Trên thực tế, NetFlows đơn hướng này không nên sử dụng vì kết quả nhanh chóng bị vượt qua đáng kể bởi phân tích thứ hai của bộ dữ liệu, sử dụng NetFlows hai chiều. Các NetFlow hai chiều có một số lợi thế hơn có hướng:

- NetFlows hai chiều giải quyết vấn đề phân biệt giữa client và server;
- NetFlows hai chiều bao gồm nhiều thông tin hơn;
- NetFlows bao gồm nhiều nhãn chi tiết hơn.

Bảng 2. Thống kê thời gian, số gói tin, số NetFlows, và kích thước của file pcap

Id	Duration(hrs)	# Packets	# NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52.0GB	Neris	1
2	4.21	71,851,300	1,808,123	60.0GB	Neris	1
3	66.85	167,730,395	4,710,639	121.0GB	Rbot	1
4	4.21	62,089,135	1,121,077	53.0GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1
6	2.18	38,764,357	558,920	30.0GB	Menti	1

Bảng 2. Thống kê thời gian, số gói tin, số NetFlows, và kích thước của file pcap (tt)

Id	Duration(hrs)	# Packets	# NetFlows	Size	Bot	#Bots
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.50	155,207,799	2,954,231	123.0GB	Murlo	1
9	5.18	115,415,321	2,753,885	94.0GB	Neris	10
10	4.75	90,389,782	1,309,792	73.0GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34.0GB	Virut	1

Bảng 3. Thống kê số lượng flow có botnet, flow thông thường và flow nền trong mỗi kịch bản của CTU-13

Scen.	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2,824,636	39,933 (1.410%)	30,387 (1.070%)	1,026 (0.030%)	2,753,290 (97.470%)
2	1,808,122	18,839 (1.040%)	9,120 (0.500%)	2,102 (0.110%)	1,778,061 (98.330%)
3	4,710,638	26,759 (0.560%)	116,887 (2.480%)	63 (0.001%)	4,566,929 (96.940%)
4	1,121,076	1,719 (0.150%)	25,268 (2.250%)	49 (0.004%)	1,094,040 (97.580%)
5	129,832	695 (0.530%)	4,679 (3.600%)	206 (1.150%)	124,252 (95.700%)
6	558,919	4,431 (0.790%)	7,494 (1.340%)	199 (0.030%)	546,795 (97.830%)
7	114,077	37 (0.030%)	1,677 (1.470%)	26 (0.020%)	112,337 (98.470%)
8	2,954,230	5,052 (0.170%)	72,822 (2.460%)	1,074 (2.400%)	2,875,282 (97.320%)
9	2,753,884	179,880 (6.500%)	43,340 (1.570%)	5,099 (0.180%)	2,525,565 (91.700%)
10	1,309,791	106,315 (8.110%)	15,847 (1.200%)	37 (0.002%)	1,187,592 (90.670%)
11	107,251	8,161 (7.600%)	2,718 (2.530%)	3 (0.002%)	96,369 (89.850%)
12	325,471	2,143 (0.650%)	7,628 (2.340%)	25 (0.007%)	315,675 (96.990%)
13	1,925,149	38,791 (2.010%)	31,939 (1.650%)	1,202 (0.060%)	1,853,217 (96.260%)

Mối quan hệ giữa thời gian của kịch bản, số lượng gói tin, số NetFlows và kích thước của tệp pcap được trình bày trong Bảng 2. Bảng này cũng cho biết phần mềm độc hại được sử dụng và số máy tính bị nhiễm trên mỗi kịch bản.

Đặc điểm khác biệt của bộ dữ liệu CTU-13 là đã được nhóm tác giả phân tích và gán nhãn từng kịch bản một cách thủ công. Quá trình ghi nhãn đã được thực hiện bên trong các tập tin NetFlows. Bảng 3 cho thấy mối quan hệ giữa số lượng nhãn cho traffic flow nền, có botnet và traffic flow thông thường trong mỗi kịch bản.

3.2. Tiêu chí đánh giá độ chính xác

Quy ước:

- True Positive: Số mẫu botnet được nhận biết chính xác là botnet
- False Negative: Số mẫu botnet bị nhận nhầm là bình thường
- False Positive: Số mẫu bình thường bị nhận nhầm là botnet
- True Negative: Số mẫu bình thường được nhận biết chính xác.

Để đánh giá tính chính xác của việc phát hiện botnet, chúng tôi sử dụng độ đo DR (*Detection Rate*) và FPR (*False Positive Rate*) vẫn thường được sử dụng trong việc đánh giá phát hiện botnet (Haddadi & Zincir-Heywood, 2014; Haddadi & ctg., 2015).

- Độ đo DR được định nghĩa là tỉ lệ giữa True Positive với (True Positive + False Negative), cho biết khả năng phát hiện đủ các mẫu botnet.
- Độ đo FPR được định nghĩa là tỉ lệ giữa False Positive với (False Positive + True Negative), cho biết tỉ lệ các mẫu bình thường bị nhận biết nhầm thành botnet.
- Độ đo TNR (*True Negative Rate*) được định nghĩa là tỉ lệ giữa True Negative với (True Negative + False Positive), cho biết khả năng phát hiện đủ các mẫu bình thường.
- Độ đo FNR (*False Negative Rate*) được định nghĩa là tỉ lệ giữa False Negative với (False Negative + True Positive), cho biết tỉ lệ các mẫu botnet bị nhận nhầm thành bình thường.

3.3. Kết quả thực nghiệm

Nhóm tác giả đã chọn ra tập dữ liệu thử nghiệm là tập con của các flow trong tập CTU-13 gốc. Bảng 4 trình bày số lượng mẫu (botnet và bình thường) trong từng kịch bản của bộ CTU-13 rút gọn. Trong mỗi kịch bản, số lượng mẫu botnet và mẫu bình thường được chọn bằng nhau để tránh ảnh hưởng không công bằng khi huấn luyện mô hình máy học để phân lớp. Kết quả thử nghiệm trên tập dữ liệu rút gọn này được công bố trên 80% (cho mỗi kịch bản).

Do số lượng mẫu trong tập dữ liệu rút gọn tương đối ít, chỉ có kịch bản 1, 2, 9, và 13 có trên 4000 mẫu, nên chúng tôi không chọn sử dụng tập dữ liệu rút gọn để huấn luyện thử nghiệm mô hình CNN để phát hiện botnet.

Bảng 4. Số lượng mẫu trong bộ CTU-13 rút gọn

Kịch bản	Số mẫu botnet	Số mẫu bình thường	Tổng số mẫu (flow)
CTU-1	3,233	3,233	6,466
CTU-2	2,374	2,374	4,748
CTU-3	19	19	38
CTU-4	2	2	4
CTU-5	159	159	318
CTU-6	27	27	54
CTU-7	49	49	98
CTU-8	53	53	106
CTU-9	3,803	3,803	7,606
CTU-10	71	71	142
CTU-11	10	10	20
CTU-12	428	428	856
CTU-13	3,803	3,803	7,606

Với tập CTU-13 đầy đủ, chúng tôi sử dụng thử nghiệm với ba tập đặc trưng sau: Tập đặc trưng Argus cơ bản (Argus, n.d.), tập đặc trưng Argus mở rộng, và tập đặc trưng Tranalyzer (dựa theo khuyến nghị trong Haddadi, Phan, và Zincir-Heywood (2016)). Bảng 5, 6, và 7 lần lượt trình bày kết quả thử nghiệm trên tập dữ liệu CTU-13 (bản đầy đủ) với các tập đặc trưng này. Để tránh việc phụ thuộc vào dữ liệu huấn luyện và khảo sát khả năng của hệ thống thích nghi với nhiều tình huống khác nhau, chúng tôi sử dụng phương pháp *k*-fold với số lượng phần (*fold*) là 10. Với mỗi giá trị K của số lượng chu kỳ trong giai đoạn biến đổi đặc trưng, chúng tôi lần lượt xem xét các cấu hình khác nhau của mô hình CNN theo kiến trúc được khảo sát. Mỗi cấu hình cụ thể tương ứng với bộ tham số gồm: (1) Số lượng filter trong filter bank và kích thước của filter trong mỗi chu kỳ; (2) Số lượng đoạn flow liên tiếp d được dùng. Với mỗi cấu hình, chúng tôi lần lượt huấn luyện 9/10 số lượng mẫu và sử dụng 1/10 số lượng mẫu để kiểm chứng. Độ chính xác của cấu hình tốt nhất mà chúng tôi tìm được cho mỗi giá trị K–số lượng chu kỳ, được thể hiện trong Bảng 5.

Qua kết quả thử nghiệm chúng ta có thể thấy là nếu giai đoạn biểu diễn đặc trưng có ít chu kỳ ($K = 1$ hay $K = 2$) để rút trích đặc trưng thì việc nhận biết botnet chưa thật sự tốt. Tuy nhiên, khi tăng số lượng chu kỳ lên, kết quả nhận biết botnet được cải thiện (với $K = 3$ hay $K = 4$). Chúng tôi không tiếp tục xét với giá trị $K \geq 6$ vì lúc này cấu trúc neural network tương đối phức tạp, độ chính xác nếu cải thiện cũng không đáng kể so với việc chi phí tính toán sẽ khá cao và có nguy cơ rơi vào hiện tượng quá khớp. Kết quả thực nghiệm cho thấy tập đặc trưng Argus mở rộng có khuynh hướng cho kết quả tốt hơn tập đặc trưng Argus cơ bản và có kết quả tốt tương đương với tập đặc trưng Tranalyzer. Điều này cũng phù hợp với nhận xét khi thử nghiệm các phân lớp truyền thống (C4.5, SVM) trên tập dữ liệu CTU-13 rút gọn (Haddadi & ctg., 2016).

Bảng 5. Kết quả thử nghiệm trên CTU-13 (bộ đầy đủ) khi sử dụng dữ liệu đầu vào là tập đặc trưng Argus cơ bản

	Botnet		Bình thường	
	DR	FPR	TNR	FNR
CNN được đề xuất (K = 1)	85.8%	14.2%	87.9%	12.1%
CNN được đề xuất (K = 2)	87.5%	12.5%	89.1%	10.9%
CNN được đề xuất (K = 3)	92.3%	7.7%	90.3%	9.7%
CNN được đề xuất (K = 4)	91.7%	8.3%	90.6%	9.4%
CNN được đề xuất (K = 5)	90.4%	9.6%	92.6%	7.4%

Bảng 6. Kết quả thử nghiệm trên CTU-13 (bộ đầy đủ) khi sử dụng dữ liệu đầu vào là tập đặc trưng Argus mở rộng

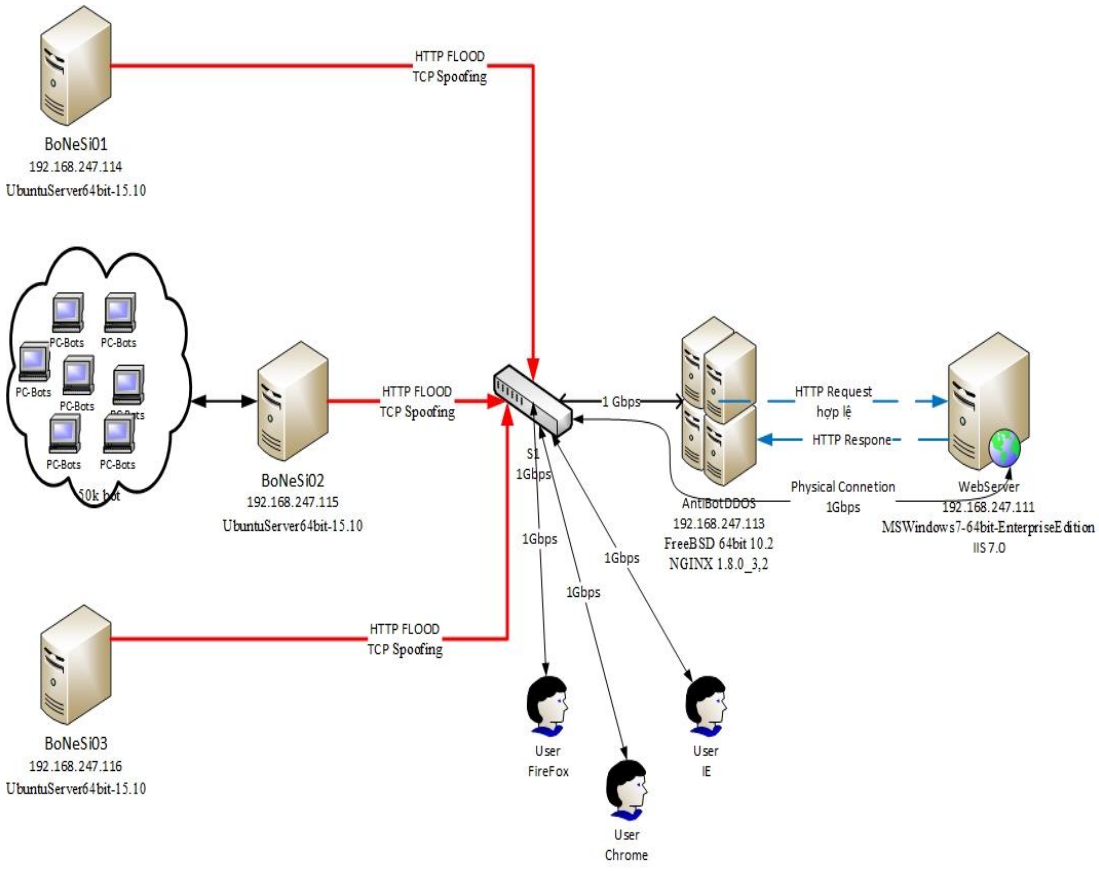
	Botnet		Bình thường	
	DR	FPR	TNR	FNR
CNN được đề xuất (K = 1)	85.9%	14.1%	88.7%	11.3%
CNN được đề xuất (K = 2)	87.8%	12.2%	89.2%	10.8%
CNN được đề xuất (K = 3)	93.2%	6.8%	91.7%	8.3%
CNN được đề xuất (K = 4)	92.4%	7.6%	92.5%	7.5%
CNN được đề xuất (K = 5)	91.3%	8.7%	91.9%	8.1%

Bảng 7. Kết quả thử nghiệm trên CTU-13 (bộ đầy đủ) khi sử dụng dữ liệu đầu vào là tập đặc trưng Tranalyzer

	Botnet		Bình thường	
	DR	FPR	TNR	FNR
CNN được đề xuất (K = 1)	85.9%	14.1%	88.7%	11.3%
CNN được đề xuất (K = 2)	87.8%	12.2%	89.2%	10.8%
CNN được đề xuất (K = 3)	93.2%	6.8%	91.7%	8.3%
CNN được đề xuất (K = 4)	92.4%	7.6%	92.5%	7.5%
CNN được đề xuất (K = 5)	91.3%	8.7%	91.9%	8.1%

Việc thử nghiệm trên tập CTU-13 với 13 kịch bản của bảy loại botnet khác nhau cho thấy tiềm năng sử dụng giải pháp phân lớp bằng máy học ứng dụng convolutional neural network với nhiều lớp ẩn. Chúng tôi đã tiến hành thử nghiệm với nhiều cấu hình cụ thể khác nhau của nhóm các convolutional neural network có kiến trúc gần giống nhau để chọn ra một cấu hình phù hợp, có khả năng phát hiện với tỉ lệ chính xác cao các flow botnet.

3.4. Mô hình thử nghiệm AntiBotDDOS



Hình 7. Mô hình kiểm thử trên môi trường mạng LAN

Web Server

- Thông tin cấu hình phần cứng: DELL OptiPlex 6th Generation Intel Corei3 processor, RAM 4 GB, NIC 1 Gbps.
- Hệ điều hành sử dụng là Microsoft Windows 7 64bit Enterprise Edition.
- Phần mềm đóng vai trò làm dịch vụ web server là IIS 7.0.
- Địa chỉ IP Address: 192.168.247.111/24.

AntiBotDDOS Server

- Thông tin cấu hình phần cứng: Dell Precision Tower 3420, Intel Core i5 processor, RAM 8 GB, NIC 1 Gbps.
- Hệ điều hành sử dụng là FreeBSD 64bit 10.2.

- Phần mềm đóng vai trò làm web application proxy server: NGINX 1.8.0_3,2.
- Địa chỉ IP Address: 192.168.247.113/24.

BoNeSi:

BoNeSi là công cụ giả lập tấn công DDoS có sử dụng mạng Botnet. Công cụ này giả lập Botnet Traffic để tạo ra hiệu ứng giống như tấn công DDoS thật sự.

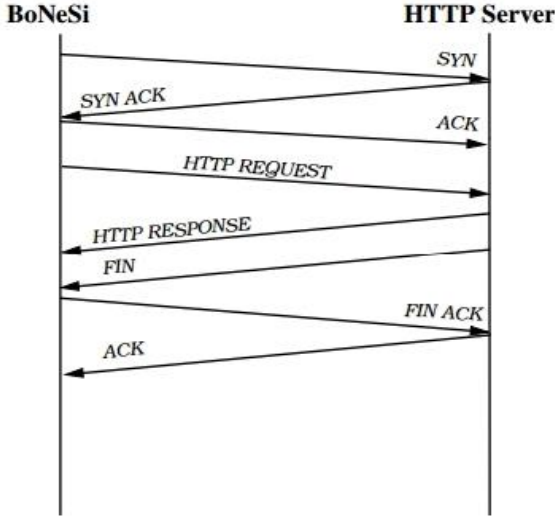
- Sử dụng ba máy tính có cấu hình Dell Precision Tower 3620, Intel Corei5processor, RAM 8 GB, NIC 1 Gbps, NIC 1 Gbps.
- Hệ điều hành Ubuntu Server 64bit 15.10.
- IP Address: 192.168.247.114/24, 192.168.247.115/24, 192.168.247.114/24.

Người dùng hợp lệ

Có ba người dùng hợp lệ sử dụng Dell Precision Tower 3620, Intel Core i5processor, RAM 8 GB, NIC 1 Gbps, NIC 1 Gbps lần lượt với các trình duyệt Internet Explorer, Google Chrome, FireFox để truy cập.

3.5. Kịch bản tấn công

BoNeSi Servers (Hình 8) tạo ra HTTP Flood từ bộ địa chỉ IP đã được thiết lập sẵn để kết nối đến danh sách URL. BoNeSi sử dụng thư viện libnet và libpcap trên Linux để nhận IP Packets từ lớp mạng của hạt nhân Linux, sau đó nó tiêm IP Packets này vào gói tin để gửi tới web server mục tiêu. Trong một HTTP Flood, BoNeSi thiết lập kết nối TCP và gửi HTTP request tới web server mục tiêu (Hình 8).

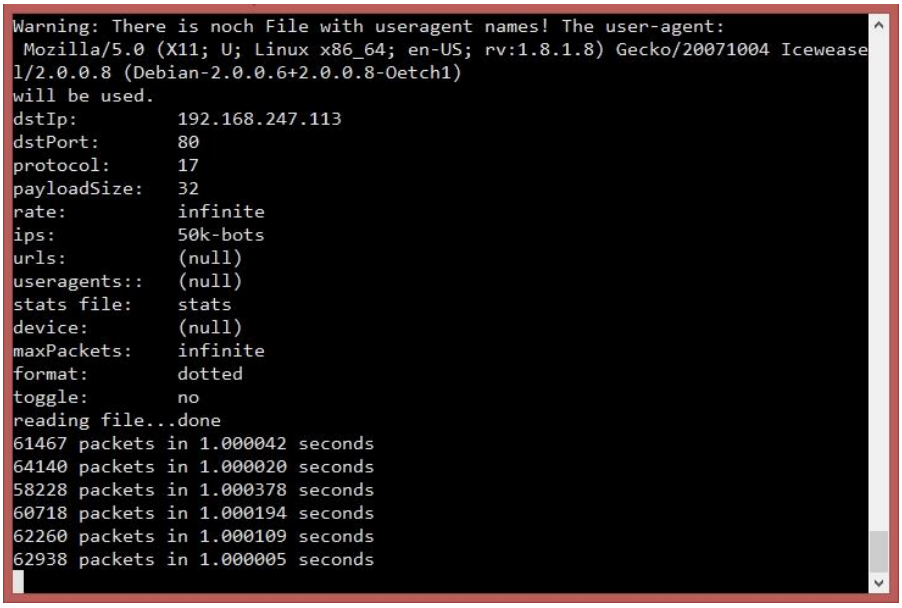


Hình 8. Cách thức kết nối của BoNeSi-PC-Bots tới web server mục tiêu

Trong HTTP request mà BoNeSi gửi đi sẽ có option “Connection: Close” trong header line. Với header line như vậy, web server sẽ đóng kết nối TCP lại ngay lập tức sau khi nó gửi HTTP reponse. Ba máy tính sử dụng BoNeSi (được gọi là BoNeSi Servers), mỗi máy tạo ra 50,000 PC-Bots với payload size 1,470 bytes, lưu lượng mạng của cuộc tấn công này khoảng 1,764 Gbps.

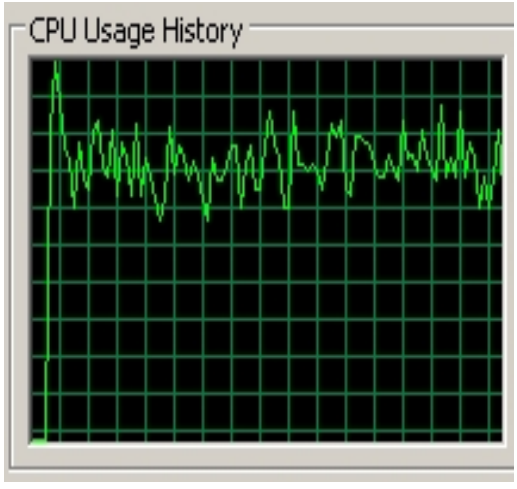
4. KẾT QUẢ VÀ THẢO LUẬN

Khi thực hiện tấn công với BoNeSi (Hình 9) vào hệ thống.



Hình 9. Trạng thái tấn công của BoNeSi

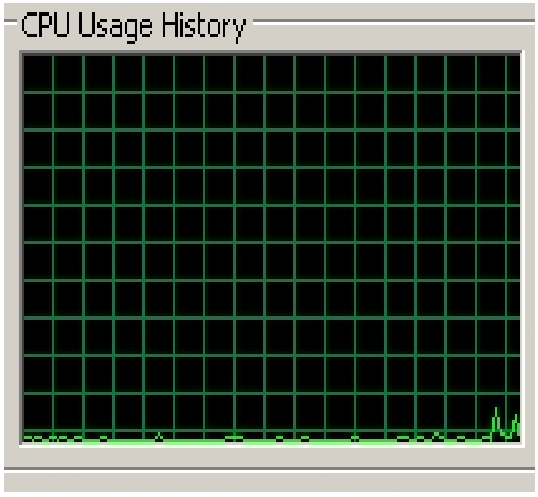
Trong trường hợp không sử dụng AntiBotDDOS thì trạng thái của webserver mục tiêu được ghi nhận (Hình 10).



Hình 10. CPU của web server mục tiêu trong tình huống bị tấn công DDoS và không sử dụng AntiBotDDOS

Truy xuất từ các máy tính người sử dụng hợp lệ, các request đều bị time out. Điều đó cho thấy tác hại của DDoS đủ lớn để làm vô hiệu hóa hoạt động của Webserver.

Khi kích hoạt hệ thống AntibotDDOS (Hình 11).



Hình 11. CPU của web server mục tiêu trong tình huống bị tấn công DDoS và sử dụng AntiBotDDOS

Một trong những biểu hiện hiệu quả của AntibotDDOS (Hình 11), đó là khi bị tấn công DDoS bằng công cụ Bonesi, dưới sự bảo vệ của AntiBotDDOS, người dùng vẫn có thể truy cập bình thường vào webserver mục tiêu.

So sánh kết quả đạt được với một số sản phẩm thương mại:

- Trong cơ chế phân biệt người dùng của sản phẩm thương mại Defense Pro, do Radware phát triển, để nhận dạng người dùng và PC-Bots khá giống với module HTTP Challenge của AntiBotDDOS. Cơ chế challenge PC-Bots bằng cách sử dụng code HTTP 302 cũng gần giống với AntiBotDDOS (Bảng 8).

- Không giống như công nghệ của các nhà cung cấp khác, tấn công DDoS được F5 Network phân chia như sau: [Mitigating DDoS Attacks with F5 Technology–White paper]: Network attacks (layer 3 và layer 4–Mô hình OSI), session attacks (layers 5 và 6), application attacks (layer 7). Mỗi loại tấn công F5 Network sử dụng một loại công nghệ được mô tả sơ lược tại [Mitigating DDoS Attacks with F5 Technology–White paper]. Đối với kiểu tấn công DDoS HTTP GET Flood và Recursive GET Flood, nền tảng BIG-IP của F5 ngăn chặn bằng cách sử dụng JavaScript để kiểm tra đâu là trình duyệt web thật sự của người dùng. [Mitigating DDoS Attacks with F5 Technology–White paper, mục Recursive GET floods], cách thức này gần giống với hoạt động của AntiBotDDOS–module JavaScript challenge.

Bảng 8. Bảng so sánh AntibotDDOS với các sản phẩm thương mại

Thiết bị	Max Mitigation Capacity/ Throughput	Max Legit Concurrent Sessions	Max Attack Concurrent Sessions	Network Operation	Block Actions
AntiBotDDOS	10 Gbps	65,536,000	65,536,000	Layer 4, 7	Drop connection
Radware-DefensePro x06 Series		2,000,000	Không giới hạn	Layer 2, 7	Drop packet, reset (source, destination, both), suspend (source, src port, destination, dest port or any combination), Challenge-Response for TCP, HTTP and DNS suspicious traffic
F5-VIPRION 2100 Blade	40 Gbps L4, 18 Gbps L7	1M L7 requests per second 400K L4 connections per second 7M-HTTP requests per second	12M max L4 CCU	L2, L4, L7	

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong bài báo 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 sử dụng Convolutional Neural Network vào phân tích và đánh giá để phát hiện tấn công botnet, chúng tôi đề xuất một họ kiến trúc tổng quát sử dụng thuộc nhóm Convolutional Neural Network để biến đổi từ đặc trưng thô do các công cụ ghi nhận và phân tích network flow cung cấp thành đặc trưng cấp cao hơn, từ đó tiến hành phân lớp (nhị phân) để đánh giá một flow tương ứng với tình trạng bị botnet tấn công hay không. Qua kết quả thực nghiệm chúng tôi thử nghiệm trên tập CTU-13 với các cấu hình khác nhau của convolutional neural network là để đánh giá tiềm năng dùng deep learning với convolutional neural network vào bài toán phát hiện botnet. Đây là giải pháp tiềm năng và có thể được sử dụng trong tương lai với khối lượng flow botnet được ghi nhận đủ nhiều và đa dạng. Chúng tôi đã đề xuất hệ thống phát hiện Botnet sử dụng Web proxy. Với khả năng tự kiểm tra và phân biệt người dùng và PC-Bot thông qua các kỹ thuật; Challenge HTTP; Challenge Java; Phát hiện fake IP; Hệ thống có khả năng tự học, tự cấu hình để điều chỉnh các thông số nhằm tối ưu hoạt động phát hiện Botnet. Và khả năng xác thực người dùng thông qua cơ chế Captcha. Đây là một kỹ thuật giúp triển khai hệ thống phát hiện Botnet với chi phí thấp mang lại hiệu quả cao.

Trong hướng nghiên cứu tiếp theo, chúng tôi sẽ tiếp tục thu thập đủ dữ liệu (với khối lượng đủ nhiều) các flow thực tế, đặc biệt là các flow botnet của nhiều loại botnet

khác nhau để huấn luyện mô hình convolutional neural network để hoàn thiện các sản phẩm có khả năng phát hiện sớm Botnet.

TÀI LIỆU THAM KHẢO

Argus. (n.d.). Retrieved from https://openargus.org/.

Celik, Z. B., Raghuram, J., Kesidis, G., & Miller, A. J. (2011). *Salting public traces with attack traffic to test flow classifiers*. Paper presented at The USENIX 4th CSET Workshop, California, USA.

Garcia, S., Grill, M., Stiborek, H., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers and Security Journal*, *45*, 100-123.

Gu, G., Porras, P., Yegneswaran, V., Fong, M., & Lee, W. (2007). *BotHunter: Detecting malware infection through ids-driven dialog correlation*. Paper presented at The 16th USENIX Security Symposium, Massachusetts, USA.

Haddadi, F., Le, C. D., Porter, L., & Zincir-Heywood, A. N. (2015). On the effectiveness of different botnet detection approaches. In J. Lopez & Y. Wu (Eds), *Information security practice and experience* (pp. 121-135). Berlin, German: Springer Publishing.

Haddadi, F., Phan, D. T., & Zincir-Heywood, A. N. (2016). *How to choose from different botnet detection systems?* Istanbul, Turkey: Institute of Electrical and Electronics Engineers Publishing.

Haddadi, F., & Zincir-Heywood, A. N. (2014). Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. *IEEE Systems Journal*, *10*(4), 1390-1401.

Maji. (n.d.). Retrieved from https://research.wand.net.nz/software/maji.php.

Netmate. (n.d.). Retrieved from https://github.com/DanielArndt/netmate-flowcalc.

Softflowd. (n.d.). Retrieved from http://www.mindrot.org/projects/softflowd.

Snort. (n.d.). Retrieved from Snort: https://www.snort.org.

Tranalyzer. (n.d.). Retrieved from https://tranalyzer.com/.

Wang, K., Huang, C., & Lin, S. (2011). A fuzzy pattern-based filtering algorithm for botnet detection. *Computer Networks*, *55*, 3275-3286.

Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., & Kirda, E. (2009). Automatically generating models for botnet detection. In M. Backers & P. Ning (Eds), *Computer science–ESORICS* (pp. 232-249). Berlin, German: Springer Publishing.

YAF. (n.d.). Retrieved from https://tools.netsa.cert.org/yaf/index.html.

Zhao, D., Traore, I., Ghorbani, A., Sayed, B., Saad, S., & Lu, W. (2012). Peer to peer botnet detection based on flow intervals. In D. Gritzalis, S. Furnell, & M. Theoharidou (Eds), *Information security and privacy research* (IFIP Advances in Information and Communication Technology) (pp. 87-102). Berlin, German: Springer Publishing.