TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN THANH HUY - 51900800**

**NGUYỄN TRƯỜNG KHẢ - 51900752**

**PHÁT HIỆN TẤN CÔNG DDOS**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

**MẠNG MÁY TÍNH**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TRẦN THANH HUY - 51900800**

**NGUYỄN TRƯỜNG KHẢ- 51900752**

**PHÁT HIỆN TẤN CÔNG DDOS**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

**MẠNG MÁY TÍNH**

Người hướng dẫn

**TS. TRƯƠNG ĐÌNH TÚ**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Lời đầu tiên em xin chân thành cảm ơn quý thầy, cô khoa Công Nghệ Thông Tin đã tận tình giảng dạy, chỉ bảo em trong những năm học vừa qua, những kiến thức mà em nhận được trên giảng đường đại học là nền tảng cho quá trình nghiên cứu đề tài này. Em xin chân thành cảm ơn Khoa Công nghệ thông tin đã tạo điều kiện để em có thể thực hiện đề tài nghiên cứu.

Em xin chân thành cảm ơn thầy Trương Đình Tú – giảng viên hướng dẫn em môn học Dự án Công nghệ thông tin, đã luôn theo sát, nhiệt tình giúp đỡ em trong suốt quá trình thực hiện đề tài **Phát Hiện tấn công DDoS**.

Mặc dù bản thân em đã nỗ lực hết mức, nhưng bài báo cáo này không thể tránh khỏi còn nhiều thiếu sót. Vì thế, em mong rằng sẽ nhận được những lời nhận xét, góp ý từ quý thầy, cô để em có thể tiếp tục phát triển và hoàn thiện hơn nữa.

Một lần nữa, em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày ... tháng … năm 2024*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS. Trương Đình Tú. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày … tháng … năm 2024*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

**PHÁT HIỆN TẤN CÔNG DDoS**

**TÓM TẮT**

Các cuộc tấn công từ chối dịch vụ phân tán (DDoS) đã trở thành vũ khí ưa thích của tin tặc, tống tiền mạng và khủng bố mạng. Những cuộc tấn công này có thể nhanh chóng vô hiệu hóa nạn nhân, gây thiệt hại lớn về doanh thu. Mặc dù có nhiều giải pháp truyền thống để giảm thiểu, các cuộc tấn công DDoS vẫn tiếp tục gia tăng về tần suất, quy mô và mức độ nghiêm trọng. Điều này đòi hỏi một mô hình mạng mới để đối phó với các mối đe dọa an ninh đầy thách thức ngày nay. Mạng điều khiển bằng phần mềm (SDN) là một mô hình mạng mới nổi, đã thu hút được sự quan tâm đáng kể từ nhiều nhà nghiên cứu để đáp ứng yêu cầu của các trung tâm dữ liệu hiện nay.

Phương pháp học máy đang ngày càng được ứng dụng rộng rãi trong việc phát hiện tấn công từ chối dịch vụ phân tán (DDoS). So với các phương pháp truyền thống, như dựa trên các quy tắc tĩnh hoặc ngưỡng phát hiện cố định, phương pháp học máy có khả năng phân tích và nhận diện các mẫu hành vi phức tạp và bất thường trong lưu lượng mạng. Các mô hình học máy có thể học từ dữ liệu lịch sử, cải thiện độ chính xác và khả năng phát hiện các cuộc tấn công mới hoặc chưa từng gặp trước đây. Điều này làm cho phương pháp học máy trở nên linh hoạt và hiệu quả hơn trong môi trường mạng liên tục biến đổi, giúp bảo vệ hệ thống tốt hơn trước các mối đe dọa DDoS ngày càng tinh vi.

**DDoS ATTACK DETECTION**

**ABSTRACT**

Distributed denial of service (DDoS) attacks have become the preferred weapon of hackers, cyber extortion, and cyber terrorism. These attacks can quickly disable victims, causing major revenue losses. Despite many traditional solutions for mitigation, DDoS attacks continue to increase in frequency, scale, and severity. This requires a new networking paradigm to deal with today's challenging security threats. Software-driven networking (SDN) is an emerging networking paradigm that has attracted significant attention from many researchers to meet the requirements of today's data centers.

Machine learning methods are increasingly being widely applied in detecting distributed denial of service (DDoS) attacks. Compared to traditional methods, such as based on static rules or fixed detection thresholds, machine learning methods are capable of analyzing and identifying complex and unusual behavioral patterns in network traffic. Machine learning models can learn from historical data, improving accuracy and the ability to detect new or never-before-seen attacks. This makes machine learning more flexible and effective in constantly changing network environments, helping to better protect systems against increasingly sophisticated DDoS threats.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ vii](#_Toc173254036)

[DANH MỤC BẢNG BIỂU ix](#_Toc173254037)

[DANH MỤC CÁC CHỮ VIẾT TẮT x](#_Toc173254038)

[CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI 1](#_Toc173254039)

[1.1 Giới thiệu đề tài 1](#_Toc173254040)

[1.2 Lý do chọn đề tài 1](#_Toc173254041)

[CHƯƠNG 2. CÁC PHƯƠNG PHÁP PHÁT HIỆN DDOS 3](#_Toc173254042)

[2.1 Sơ lược về tấn công DDoS 3](#_Toc173254043)

[2.2 Các phương pháp phát hiện DDoS dựa trên học máy 5](#_Toc173254044)

[CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT 9](#_Toc173254045)

[3.1 Công nghệ SDN 9](#_Toc173254046)

[3.1.1 Định nghĩa về SDN 9](#_Toc173254047)

[3.1.2 Kiến trúc SDN 10](#_Toc173254048)

[3.1.3 Giao thức OpenFlow 12](#_Toc173254049)

[3.1.4 Ryu Controller 13](#_Toc173254050)

[3.2 Xây dựng mô hình mạng 15](#_Toc173254051)

[3.3 Thu thập dữ liệu 16](#_Toc173254052)

[3.3.1 Mô tả các bước trong quá trình thu thập dữ liệu 16](#_Toc173254053)

[3.3.2 Mô tả dữ liệu 16](#_Toc173254054)

[3.4 Phân tích dữ liệu, xử lý dữ liệu, trích xuất các đặc trưng 18](#_Toc173254055)

[3.4.1 Phân tích dữ liệu 19](#_Toc173254056)

[3.4.2 Xử lý dữ liệu 19](#_Toc173254057)

[3.4.3 Trích xuất các đặc trưng cho việc training các mô hình học máy 19](#_Toc173254058)

[CHƯƠNG 4. THỰC NGHIỆM 21](#_Toc173254059)

[4.1 Các công nghệ cần chuẩn bị 21](#_Toc173254060)

[4.2 Thực nghiệm phát hiện DDos 21](#_Toc173254061)

[4.2.1 Trên máy Ubuntu ryu 21](#_Toc173254062)

[4.2.2 Trên máy Mininet VM 21](#_Toc173254063)

[4.3 Thực nghiệm phát hiện tấn công và kết quả 22](#_Toc173254064)

[4.3.1 Kiểm tra truy cập tới Web server 22](#_Toc173254065)

[4.3.2 Thực hiện tấn công HTTP Flood tới Web Server 24](#_Toc173254066)

[4.3.3 Thực hiện tấn cống ICMP Flood tới h2 25](#_Toc173254067)

[4.3.4 Thực hiện tấn công UDP Flood tới h3 26](#_Toc173254068)

[4.3.5 Thực hiện tấn công TCP-SYN Flood tới h4 27](#_Toc173254069)

[4.4 Thực nghiệm biện pháp ngăn chặn và kết quả 28](#_Toc173254070)

[4.4.1 Trên máy Ubuntu RYU 28](#_Toc173254071)

[4.4.2 Thực nghiệm tấn công và ngăn chặn block port khi phát hiện DDos 29](#_Toc173254072)

[CHƯƠNG 5. KẾT LUẬN 32](#_Toc173254073)

[5.1 Kết luận 32](#_Toc173254074)

[5.2 Hướng phát triển 32](#_Toc173254075)

[TÀI LIỆU THAM KHẢO 33](#_Toc173254076)

# DANH MỤC HÌNH VẼ

[Hình 1: Các phương pháp phát hiện dựa trên học máy [1] 5](#_Toc173254077)

[Hình 2: Các bước chính trong phương pháp đề xuất 9](#_Toc173254078)

[Hình 3: Kiến trúc cơ bản của SDN[4] 11](#_Toc173254079)

[Hình 4: Các thành phần chính của OpenFlow [4] 12](#_Toc173254080)

[Hình 5: Kiến trúc mô hình mạng 15](#_Toc173254081)

[Hình 6: Các bước trong quá trình thu thập dữ liệu 16](#_Toc173254082)

[Hình 7: Biểu đồ thể hiện tỷ lệ phần trăm giữa lưu lượng bình thường và DDos trong Dataset 19](#_Toc173254083)

[Hình 8: Khởi động controller thành công 21](#_Toc173254084)

[Hình 9: Khởi động hệ thống mạng bằng python 22](#_Toc173254085)

[Hình 10: Truy cập tới trang web trên Web Server 23](#_Toc173254086)

[Hình 11: Ping giữa các thiết bị 23](#_Toc173254087)

[Hình 12: Kết quả dự đoán lưu lượng bình thường 24](#_Toc173254088)

[Hình 13: Biểu đồ lưu lượng trên Web Server khi bị tấn công HTTP Flood 24](#_Toc173254089)

[Hình 14: Kết quả dự đoán Web Server bị tấn công 25](#_Toc173254090)

[Hình 15: Biểu đồ lưu lượng trên host h2 khi bị tấn công ICMP Flood 26](#_Toc173254091)

[Hình 16: Kết quả dự đoán host h2 bị tấn công 26](#_Toc173254092)

[Hình 17: Biểu đồ lưu lượng trên host h3 khi bị tấn công UDP Flood 27](#_Toc173254093)

[Hình 18: Kết quả dự đoán host h3 bị tấn công 27](#_Toc173254094)

[Hình 19: Biểu đồ lưu lượng trên host h4 khi bị tấn công TCP-SYN Flood 28](#_Toc173254095)

[Hình 20: Kết quả dự đoán host h4 bị tấn công 28](#_Toc173254096)

[Hình 21: Khởi động thành công controller có tích hợp ngăn chặn tấn công. 29](#_Toc173254097)

[Hình 22: Kết quả phát hiện h2 bị tấn công và tự chặn cổng đi vào của h2 30](#_Toc173254098)

[Hình 23: Kết quả phát hiện h3 bị tấn công và tự chặn cổng đi vào của h3 30](#_Toc173254099)

[Hình 24: Kết quả phát hiện h4 bị tấn công và tự chặn cổng đi vào của h4 31](#_Toc173254100)

# DANH MỤC BẢNG BIỂU

[Bảng 1: Các loại tấn công DDoS phổ biến [2] 5](#_Toc173264254)

[Bảng 2: Phương thức và ưu điểm của phương pháp học máy [1] 8](#_Toc173264255)

[Bảng 3: Tóm tắt kết quả đạt được của các phương pháp dựa trên học máy [1] 8](#_Toc173264256)

[Bảng 4: Mô tả các đặc trưng của dữ liệu 18](#_Toc173264257)

[Bảng 5: Kết quả quấn luyện các mô hình học máy với 15 đặc trưng 20](#_Toc173264258)

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| DDoS | Distributed denial of service |
| DoS | Denial of service |
| SDN | Software defined Networking |
| KNN | K-Nearest Neighbor |
| SVM | Support Vector Machines |
| DT | Decision Tree |
| ANN | Artificial Neural Network |

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, các cuộc tấn công mạng ngày càng trở nên phức tạp và khó lường. Một trong những dạng tấn công phổ biến và nguy hiểm nhất là tấn công từ chối dịch vụ phân tán (DDoS - Distributed Denial of Service). Tấn công DDoS nhắm vào mục tiêu làm gián đoạn hoặc làm suy yếu các dịch vụ trực tuyến bằng cách gửi lượng lớn yêu cầu tới máy chủ hoặc mạng mục tiêu, làm cho hệ thống không thể xử lý kịp thời và dẫn đến việc dịch vụ bị gián đoạn.

Mạng điều khiển bằng phần mềm (SDN - Software Defined Networking) là một kiến trúc mạng mới nổi, cho phép quản lý và điều khiển mạng dễ dàng hơn thông qua các giao diện lập trình. SDN tách biệt mặt phẳng điều khiển khỏi mặt phẳng dữ liệu, giúp tối ưu hóa và linh hoạt hơn trong việc quản lý mạng. Tuy nhiên, cùng với những lợi ích mà SDN mang lại, cũng tồn tại nhiều thách thức, đặc biệt là về an ninh mạng. Việc phát hiện và ngăn chặn các cuộc tấn công DDoS trong môi trường SDN là một vấn đề quan trọng và cần được nghiên cứu kỹ lưỡng.

Mô hình học máy (Machine Learning) là một phương pháp mạnh mẽ và hiệu quả trong việc phát hiện các mối đe dọa an ninh mạng. Sử dụng các thuật toán học máy, chúng ta có thể phân tích dữ liệu mạng, phát hiện các mẫu bất thường và cảnh báo về các cuộc tấn công tiềm năng một cách nhanh chóng và chính xác.

Các chương sau của báo cáo sẽ có cấu trúc như sau. Chương 2 đề cập đến các phương pháp phát hiện DDoS hiện có. Trong chương 3, nhóm sẽ đề xuất phương pháp mà nhóm nghiên cứu. Chương 4 là phần thực nghiệm cho phương pháp đã đề ra. Chương 5 là phần kết luận của báo cáo.

## Lý do chọn đề tài

Tính cấp thiết: Các cuộc tấn công DDoS đang gia tăng cả về số lượng và mức độ phức tạp, gây ra tổn thất lớn về kinh tế và uy tín cho các doanh nghiệp và tổ chức. Việc tìm ra các phương pháp hiệu quả để phát hiện và ngăn chặn các cuộc tấn công này là nhu cầu cấp bách.

Tính mới lạ: Việc sử dụng mô hình học máy để phát hiện tấn công DDoS trong môi trường SDN là một hướng nghiên cứu mới mẻ và chưa được khám phá nhiều. Sử dụng học máy cho phép phát hiện các cuộc tấn công một cách tự động và hiệu quả hơn so với các phương pháp truyền thống. Nghiên cứu này mang đến cách tiếp cận hiện đại và đổi mới trong việc giải quyết vấn đề an ninh mạng.

Tính thực tế: Kết quả nghiên cứu có thể được áp dụng trực tiếp để tăng cường khả năng phát hiện và ngăn chặn các cuộc tấn công DDoS trong môi trường SDN.

# CÁC PHƯƠNG PHÁP PHÁT HIỆN DDOS

## Sơ lược về tấn công DDoS

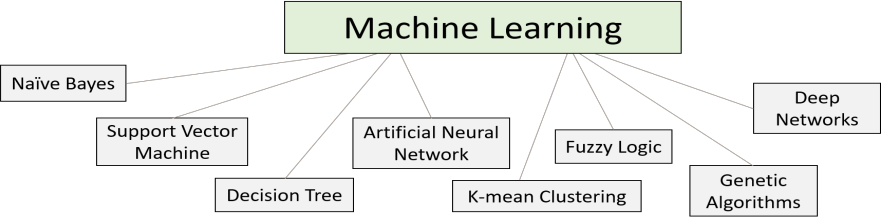
Mục tiêu của một cuộc tấn công DDoS là làm gián đoạn các dịch vụ của mục tiêu bằng cách sử dụng nhiều nguồn phân tán. Một ví dụ điển hình của cuộc tấn công này là cuộc tấn công làm ngập hệ thống mạng (flooding), trong đó nạn nhân bị áp đảo bởi lượng lớn lưu lượng mạng được gửi đến. Ý tưởng của các cuộc tấn công DDoS là sử dụng một số lượng lớn các nguồn phân tán ở nhiều địa điểm khác nhau để tấn công nạn nhân. Các mạng bot (botnets) thường hữu ích để khởi động các cuộc tấn công DDoS vì một lượng lớn các máy chủ bị chiếm quyền điều khiển (còn được gọi là zombie) thường có sẵn. Bảng 1 phân loại các cuộc tấn công DDoS phổ biến thành ba loại. [2]

|  |  |
| --- | --- |
| **Tấn công ngập lụt dựa trên phản xạ (Reﬂection and ampliﬁcation-based ﬂooding attacks)** | |
| Tấn công Smurf | Trong tấn công Smurf, một số lượng lớn gói tin ICMP (Internet Control Message Protocol) với địa chỉ IP nguồn giả mạo của nạn nhân được phát tán tới một mạng máy tính sử dụng địa chỉ IP Broadcast. Điều này tạo ra một lượng lớn lưu lượng truy cập không hợp lệ trên mạng, gây tắc nghẽn mạng. |
| Tấn công mảnh vỡ (Fraggle attack) | Tấn công mảnh vỡ tương tự như tấn công Smurf, nhưng nó sử dụng lưu lượng UDP không hợp lệ thay vì lưu lượng ICMP để đạt được cùng một mục tiêu. |
| **Tấn công ngập lụt khai thác giao thức (Protocol exploitation ﬂooding attacks)** | |
| Tấn công SYN Flood | Trong tấn công TCP SYN flood, kẻ tấn công gửi gói tin với bit SYN được đặt để khởi động quá trình ba bước bắt tay TCP. Nạn nhân phản hồi bằng cách gửi gói tin lại địa chỉ nguồn với bit SYN-ACK được đặt. Kẻ tấn công không bao giờ phản hồi lại gói tin trả lời, có thể là cố ý hoặc vì địa chỉ nguồn của gói tin bị giả mạo. Do đó, hàng đợi nhận TCP của nạn nhân sẽ bị đầy, ngăn cản các kết nối TCP mới từ các khách hàng hợp pháp. |
| Tấn công phân mảnh UDP | Trong cuộc tấn công Phân mảnh UDP, kẻ tấn công gửi các gói UDP lớn (1500+ byte) để tiêu thụ nhiều băng thông hơn với ít gói hơn. Tài nguyên của nạn nhân bị tiêu hao khi tập hợp lại các gói giả mạo này và không có khả năng tập hợp lại |
| **Tấn công ngập lụt dựa trên phản xạ và khuếch đại (Reﬂection and ampliﬁcation-based ﬂooding attacks)** | |
| Tấn công khuếch đại DNS | Trong tấn công khuếch đại DNS (DNS amplification flooding), các máy zombie tạo ra các truy vấn DNS nhỏ với địa chỉ IP nguồn giả mạo, dẫn đến việc tạo ra một khối lượng lớn lưu lượng mạng hướng về phía nạn nhân. |
| Khuếch đại NTP Tấn công lũ lụt | Tấn công khuếch đại NTP (Network Time Protocol) tương tự như tấn công khuếch đại DNS, nhưng nó sử dụng các máy chủ NTP thay vì máy chủ DNS. |

Bảng 1: Các loại tấn công DDoS phổ biến [2]

## Các phương pháp phát hiện DDoS dựa trên học máy

Phần này sẽ trình bày chi tiết về các phương pháp học máy được áp dụng trong việc phát hiện DDoS đã được nghiên cứu và công bố. Bảng 2 liệt kê các phương pháp và ưu điểm của các phương pháp ML một cách chi tiết. Bảng 3 tóm tắt kết quả đạt được của các phương pháp dựa trên học máy.



Hình 1: Các phương pháp phát hiện dựa trên học máy [1]

**Naive Bayes (NB) [1]**

Phương pháp phân loại này dựa trên mạng Bayesian và được gọi là Naive Bayes. Đây là một trong những kỹ thuật đơn giản và dễ hiểu nhất để xây dựng các bộ phân loại. Các bộ phân loại sẽ xác định các nhãn lớp cho các tình huống cụ thể và sau đó hiển thị các vector giá trị đặc trưng. Các nhãn lớp được rút ra dựa trên một số tập hợp cụ thể. Fadhil et al. [5] đã đề xuất việc sử dụng NB để phát hiện tấn công DDoS thông qua phân tích thống kê lưu lượng mạng. Một mô hình được thiết kế và triển khai thực tế để phát hiện tấn công DDoS cũng đã sử dụng phân loại NB [6].

**Support Vector Machine (SVM) [1]**

Phương pháp này được Vapnik phát triển lần đầu và đã chứng tỏ hiệu quả vượt trội, từ đó thu hút nhiều sự quan tâm trong nghiên cứu học máy. SVM có khả năng thực hiện cả hồi quy và phân loại bằng cách sử dụng học có giám sát [10]. Subbulakshmi et al. [11] đã tạo ra một tập dữ liệu chứa các cuộc tấn công DDoS và sau đó sử dụng máy vector hỗ trợ cải tiến (ESVM) để xác định các cuộc tấn công này. Ye et al. [7] cũng đã phát triển một mô hình phát hiện tấn công DDoS vào năm 2018 bằng cách kết hợp các kỹ thuật phân loại SVM.

**Decision Tree (DT) [1]**

Cây quyết định là một trong những phương pháp cơ bản và phổ biến nhất trong khai thác dữ liệu và học máy. Phương pháp này sử dụng các quan sát về danh mục để xác định giá trị mục tiêu và phân loại dữ liệu dựa trên tập dữ liệu đã học trước đó [13]. Zekri et al. [14] đã phát triển một phương pháp dựa trên cây quyết định để tự động và hiệu quả nhận diện các cuộc tấn công ngập lụt DDoS dựa trên chữ ký. Bài báo [7] cũng đã tạo ra một mô hình học máy có khả năng học từ các mẫu tấn công, kết hợp cả phát hiện tấn công DDoS dựa trên bất thường và dựa trên chữ ký.

**Artificial Neural Network (ANN) [1]**

Mạng nơron nhân tạo (ANN) được giới thiệu bởi McCulloc và Pitts vào năm 1943 với một tập hợp các neuron cơ bản, tương tự như các neuron sinh học và cấu trúc của các mạng sinh học [15]. Để nhận diện và giảm thiểu các cuộc tấn công DDoS đã biết và chưa biết trong thời gian thực, Saied et al. [16] đã phát triển một mô hình ANN. Balarengadurai và Saraswathi [17] cũng đã phát triển một mô hình đánh giá mối nguy cho IoT bằng cách sử dụng ANN trong khuôn khổ nghiên cứu của họ.

**K-mean Clustering [1]**

Phân cụm K-mean là một phương pháp phổ biến để chia một tập dữ liệu thành K nhóm. Phương pháp này tinh chỉnh các trung tâm cụm ban đầu K bằng cách cho mỗi trường hợp gia nhập vào trung tâm cụm gần nhất sau khi xác định các trung tâm cụm ban đầu. Hao et al. [18] đã phát triển một thuật toán phát hiện các cuộc tấn công DDoS của các phiên chưa biết bằng cách sử dụng thuật toán phân cụm K-means, đạt được tỷ lệ chính xác 97,83% [8].

**Fuzzy Logic [1]**

Logic mờ, được phát triển dựa trên lý thuyết tập mờ, sử dụng lý thuyết gần đúng thay vì chính xác dựa trên logic định lượng truyền thống. Để phân biệt các gói tin độc hại khỏi lưu lượng hợp pháp và thực hiện các hành động thích hợp nhằm ngăn chặn các cuộc tấn công DDoS, Iyengar và Ganapathy [19] đã phát triển một mô hình logic mờ dựa trên các quy tắc đã định. Balarengadurai và Saraswathi [20] đã phát triển một cơ chế để dự đoán và phát hiện các cuộc tấn công DDoS trong IEEE 802.15.4 bằng cách sử dụng thuật toán logic mờ.

**Genetic Algorithms [1]**

Thuật toán di truyền là một phương pháp học máy phổ biến theo các khái niệm tiến hóa. Phương pháp này tiếp cận giải quyết vấn đề tương tự như một cuộc khảo sát sinh học [21]. Chaudhary và Shrimal [8] đã đề xuất một phương pháp phát hiện các cuộc tấn công DDoS trong các mạng ad hoc di động dựa trên các thuật toán tiến hóa vào năm 2019. Một phân tích lưu lượng thời gian thực có thể mở rộng dựa trên các thuật toán tiến hóa đã được phát triển để phát hiện và giảm thiểu các cuộc tấn công DDoS trên cơ sở hạ tầng xử lý phân tán Hadoop.

|  |  |  |
| --- | --- | --- |
|  | Phương thức | Ưu điểm |
| Fadlil et al. [5] | |  | | --- | | Tạo ra một phương pháp mới để phát hiện các cuộc tấn công DDoS. |  |  | | --- | |  | | Dự đoán sẽ hoạt động phối hợp với hệ thống phát hiện xâm nhập (IDS) để dự báo sự xuất hiện của các cuộc tấn công DDoS. |
| Ye et al. [7] | Trích xuất các giá trị đặc trưng 6-tuple từ flow table của switch và tạo ra mô hình tấn công DDoS. | Có ích trong việc xác định các cuộc tấn công DDoS trong mạng định nghĩa bằng phần mềm (SDN). |
| Lucky et al. [8] | Được triển khai trong các môi trường chi phí thấp để phát hiện và giảm thiểu các cuộc tấn công DDoS một cách hiệu quả và nhanh chóng. | Thiết kế đã được xem xét và kết quả cho thấy kiến trúc mới không tạo thêm gánh nặng cho mạng được giám sát. |
| Putri et al. [9] | Trên tập dữ liệu ISCX trong môi trường thử nghiệm, Snort phát hiện lên tới 42 cảnh báo về các cuộc tấn công DoS. | Do sự khác biệt về độ chính xác giữa giá trị và công cụ phân cụm WEKA, các gói dữ liệu mneg-cluster được chọn ngẫu nhiên từ một gói giá trị dữ liệu và được sử dụng để tính toán giá trị của centroid. |
| Chaudhary and  Shrimal [10] | Mục tiêu của nghiên cứu này là tạo ra một hệ thống IDS dựa trên thuật toán di truyền để phát hiện các cuộc tấn công DDoS trong mạng MANETs. | Theo kết quả thực hiện, hệ thống IDS được đề xuất, dựa trên các thuật toán tiến hóa, có thể phát hiện hiệu quả các cuộc tấn công DDoS trên MANETs. |

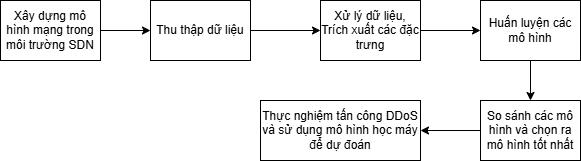
Bảng 2: Phương thức và ưu điểm của phương pháp học máy [1]

|  |  |  |  |
| --- | --- | --- | --- |
|  | Kĩ thuật | Độ chính xác(%) | Dataset |
| Fadlil et al. [5] | NB | - | MITRLADUY |
| Ye et al. [6] | SVM | 95,24 | - |
| Zekri et al. [14] | Decision tree | - | Tự thu thập |
| Lucky et al. [7] | Decision tree | 99.93 | CIC 2017 and 2019 |
| Putri et al. [9] | K-mean clustering | 99.69 | ISCX |
| Chaudhary and Shrimal [10] | Genetic algorithm | 85 | Tự thu thập |

Bảng 3: Tóm tắt kết quả đạt được của các phương pháp dựa trên học máy [1]

# PHƯƠNG PHÁP ĐỀ XUẤT

Trong chương này, nhóm đề xuất phương pháp phát hiện DDoS dựa trên học máy kết hợp với xây dựng mô hình mạng trong môi trường SDN. Đầu tiên cần xây dựng mô hình mạng giả lập trong môi trường SDN. Tiến hành thu thập dữ liệu và gắn nhãn để xây dựng dataset cho mô hình học máy. Sau khi có dataset, tiến hành phân tích, xử lý dữ liệu và trích xuất các đặc trưng quan trọng. Tiến hành huấn luyện và so sánh các mô hình dựa trên các tiêu chí khác nhau (Accuracy, Recall, Precision F1 Score, …), sau đó chọn ra mô hình tốt nhất. Cuối cùng áp dụng mô hình học máy vào thực nghiệm phát hiện tấn công DDoS.



Hình 2: Các bước chính trong phương pháp đề xuất

## Công nghệ SDN

### Định nghĩa về SDN

SDN (Software-Defined Networking) là một phương pháp tiếp cận quản lý mạng dựa trên phần mềm. Nó tách biệt các chức năng điều khiển mạng khỏi các thiết bị phần cứng truyền thống như router và switch, thay vào đó sử dụng phần mềm để quản lý và điều khiển mạng.

Về cơ bản, SDN chia tách độc lập hai cơ chế hiện đang tồn tại trong cùng một thiết bị mạng: Cơ chế điều khiển (Control Plane), cơ chế chuyển tiếp dữ liệu (Data Plane) để có thể tối ưu hoạt động của hai cơ chế này [2].

Control Plane (Mặt phẳng điều khiển)

* Chức năng: Control Plane chịu trách nhiệm ra quyết định về cách các gói dữ liệu sẽ được chuyển tiếp qua mạng. Nó bao gồm các chức năng như định tuyến, định tuyến lại (rerouting), xây dựng bảng định tuyến (routing tables), và các giao thức điều khiển như OSPF, BGP, và RIP.
* Hoạt động: Các thiết bị mạng như router và switch sử dụng Control Plane để trao đổi thông tin về trạng thái mạng, từ đó đưa ra quyết định về đường đi của các gói dữ liệu. Ví dụ, router sử dụng giao thức định tuyến để học biết về các mạng khác và xác định đường đi tốt nhất cho các gói dữ liệu.
* Vị trí: Control Plane thường được tích hợp trong các thiết bị mạng (router, switch). Mỗi thiết bị mạng có Control Plane riêng của nó để quản lý và ra quyết định.

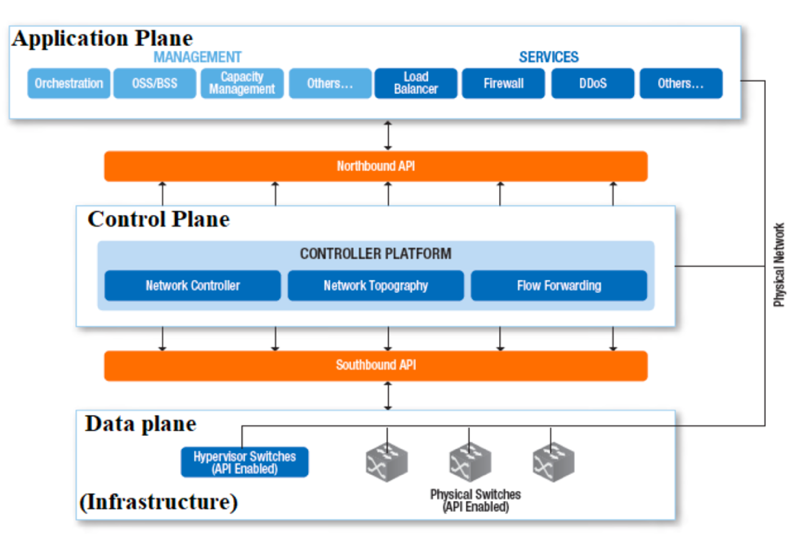
Data Plane (Mặt phẳng dữ liệu)

* Chức năng: Data Plane chịu trách nhiệm chuyển tiếp các gói dữ liệu theo các quyết định được đưa ra bởi Control Plane. Nó thực hiện các hoạt động như truyền tải, định tuyến và lọc gói tin.
* Hoạt động: Khi một gói dữ liệu đến một thiết bị mạng, Data Plane sẽ dựa vào bảng định tuyến (được xây dựng bởi Control Plane) để xác định đường đi của gói tin và chuyển tiếp nó đến đích. Data Plane thực hiện các hoạt động này ở tốc độ cao để đảm bảo hiệu suất mạng.
* Vị trí: Data Plane cũng được tích hợp trong các thiết bị mạng, nơi nó hoạt động song song với Control Plane nhưng với vai trò khác biệt.

Trong các mạng truyền thống, Control Plane và Data Plane thường được tích hợp trong cùng một thiết bị mạng. Tuy nhiên, với sự ra đời của SDN (Software-Defined Networking), hai mặt phẳng này được tách biệt rõ ràng:

* Control Plane: Được chuyển lên các bộ điều khiển trung tâm (controllers), quản lý và ra quyết định cho toàn bộ mạng.
* Data Plane: Vẫn được duy trì trong các thiết bị mạng nhưng hoạt động dưới sự chỉ đạo của Control Plane từ các bộ điều khiển.

### Kiến trúc SDN



Hình 3: Kiến trúc cơ bản của SDN[4]

Kiến trúc SDN bao gồm ba lớp chính:

* Lớp hạ tầng mạng (Infrastructure):
  + Bao gồm các thiết bị mạng như switch và router.
  + Chịu trách nhiệm truyền tải dữ liệu.
* Lớp điều khiển (Control plane):
  + Gồm các bộ điều khiển (controller), đây là nơi diễn ra việc ra quyết định và quản lý toàn bộ mạng.
  + Controller giao tiếp với các thiết bị mạng qua giao diện phía Nam (Southbound Interface), thường sử dụng giao thức OpenFlow.
* Lớp ứng dụng (Application plane):
  + Là lớp ứng dụng dành cho người dùng, người dùng có thể sử dụng các API được cung cấp từ lớp Control Layer để có thể lập trình hoạt động của mạng, ngoài ra cho phép giám sát được lưu lượng ra vào mạng và hiệu năng.
  + Các ứng dụng này cung cấp các chức năng và dịch vụ mạng cụ thể, như quản lý lưu lượng, bảo mật, và ảo hóa mạng.

Các lớp giao tiếp thông qua một tập hợp các giao diện được gọi là northbound APIs và southbound APIs.

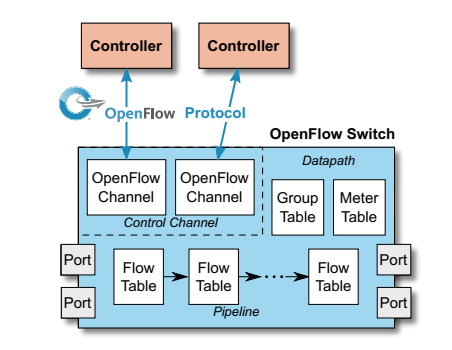
### Giao thức OpenFlow

#### Giới thiệu OpenFlow

Giao thức OpenFlow là một thành phần quan trọng trong kiến trúc Mạng điều khiển bằng phần mềm (SDN - Software-Defined Networking). OpenFlow cho phép các bộ điều khiển SDN trực tiếp tương tác với tầng dữ liệu của các thiết bị mạng như switch và router, để quản lý luồng dữ liệu trên mạng.

OpenFlow là giao thức mở chuẩn hóa được thiết kế bởi nhóm Open Networking Foundation (ONF). Nó cho phép sự tách rời của tầng điều khiển (control plane) và tầng dữ liệu (data plane), một khái niệm cốt lõi của SDN. Tầng điều khiển quyết định đường đi của dữ liệu, trong khi tầng dữ liệu chịu trách nhiệm chuyển tiếp dữ liệu dựa trên những quyết định đó.

#### Kiến trúc OpenFlow



Hình 4: Các thành phần chính của OpenFlow [4]

Kiến trúc OpenFlow bao gồm các thành phần chính sau:

* Controller (Bộ điều khiển): Là một hệ thống phần mềm trung tâm quản lý và điều khiển các thiết bị mạng bằng cách sử dụng giao thức OpenFlow.
* OpenFlow Switch: Là thiết bị phần cứng hoặc phần mềm hỗ trợ giao thức OpenFlow, có chức năng chuyển tiếp dữ liệu dựa trên các quy tắc (flow rules) được thiết lập bởi Controller.
* Flow Table (Bảng luồng): Mỗi switch OpenFlow có một hoặc nhiều bảng luồng, chứa các quy tắc để xác định cách xử lý các gói tin.
* Secure Channel (Kênh bảo mật): Là kênh giao tiếp an toàn giữa switch và Controller, thường sử dụng TLS (Transport Layer Security) để bảo mật.

#### Hoạt động của OpenFlow

Quy trình hoạt động của OpenFlow có thể được mô tả như sau:

* Thiết lập Kết nối: Switch OpenFlow khởi tạo một kết nối bảo mật với Controller thông qua kênh bảo mật.
* Trao đổi Thông tin: Switch gửi thông tin về các cổng (ports) và khả năng của mình tới Controller. Controller có thể sử dụng thông tin này để xây dựng các bảng luồng và cấu hình switch.
* Quản lý Luồng: Khi một gói tin mới đến switch mà không có quy tắc nào trong bảng luồng để xử lý nó, gói tin sẽ được gửi đến Controller. Controller sẽ quyết định cách xử lý gói tin đó và cài đặt một quy tắc mới vào bảng luồng của switch.
* Chuyển tiếp Gói Tin: Dựa trên các quy tắc trong bảng luồng, switch sẽ chuyển tiếp các gói tin theo các hành động được xác định trước (ví dụ: chuyển tiếp, bỏ qua, thay đổi tiêu đề, v.v.).

### Ryu Controller

Ryu là một framework mã nguồn mở được thiết kế để phát triển các ứng dụng điều khiển mạng SDN (Software-Defined Networking). Được viết bằng ngôn ngữ lập trình Python, Ryu cung cấp một bộ công cụ và thư viện phong phú để phát triển các ứng dụng điều khiển mạng một cách nhanh chóng và dễ dàng.

Ryu hỗ trợ nhiều giao thức điều khiển mạng khác nhau, bao gồm OpenFlow (từ phiên bản 1.0 đến 1.5), Netconf, OF-config và nhiều giao thức khác. Điều này giúp Ryu trở thành một công cụ linh hoạt và mạnh mẽ trong việc triển khai và quản lý mạng SDN.

#### Kiến trúc của Ryu Controller

Ryu Controller được thiết kế với kiến trúc module hóa, cho phép các nhà phát triển dễ dàng thêm, xóa hoặc tùy chỉnh các chức năng của nó. Dưới đây là các thành phần chính của Ryu:

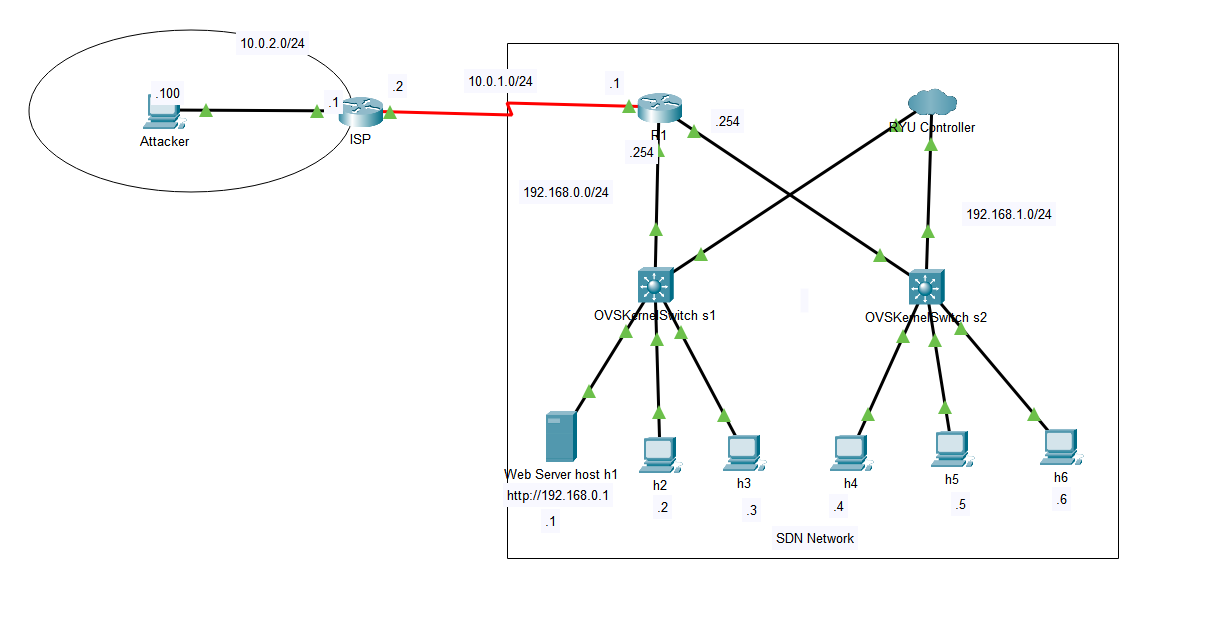
* Core Components (Thành phần lõi): Bao gồm các thư viện và công cụ cơ bản để quản lý và điều khiển mạng. Thành phần lõi của Ryu chịu trách nhiệm xử lý các yêu cầu từ các thiết bị mạng và điều khiển chúng theo các quy tắc và chính sách được định nghĩa.
* Application Components (Thành phần ứng dụng): Đây là nơi mà các ứng dụng mạng cụ thể được triển khai. Các ứng dụng này có thể bao gồm các chức năng như định tuyến, quản lý băng thông, phát hiện tấn công DDoS, v.v. Các ứng dụng này tương tác với các thành phần lõi của Ryu để thực hiện các tác vụ điều khiển mạng.
* API (Giao diện lập trình ứng dụng): Ryu cung cấp các API phong phú cho phép các nhà phát triển tương tác với controller và các thiết bị mạng. API của Ryu được thiết kế theo hướng RESTful, giúp dễ dàng tích hợp với các ứng dụng và dịch vụ khác.

#### Hoạt động của Ryu Controller

Ryu Controller hoạt động bằng cách nhận và xử lý các sự kiện mạng từ các thiết bị mạng, sau đó thực hiện các hành động điều khiển dựa trên các sự kiện này. Quá trình hoạt động của Ryu có thể được mô tả qua các bước sau:

* Kết nối và đăng ký: Các thiết bị mạng (như OpenFlow switch) kết nối với Ryu Controller thông qua giao thức OpenFlow. Khi kết nối được thiết lập, các thiết bị này đăng ký với Ryu Controller.
* Nhận sự kiện: Ryu Controller lắng nghe và nhận các sự kiện mạng từ các thiết bị đã đăng ký. Các sự kiện này có thể bao gồm các gói tin mới, thay đổi trạng thái của các cổng, và các sự kiện khác.
* Xử lý sự kiện: Ryu Controller sử dụng các ứng dụng và các module để xử lý các sự kiện nhận được. Các ứng dụng này có thể áp dụng các quy tắc và chính sách điều khiển để quyết định hành động cần thực hiện.
* Thực hiện hành động: Dựa trên kết quả xử lý sự kiện, Ryu Controller gửi các lệnh điều khiển đến các thiết bị mạng. Các lệnh này có thể bao gồm cài đặt các quy tắc mới trong flow table, thay đổi cấu hình của các cổng, hoặc các hành động khác.

## Xây dựng mô hình mạng



Hình 5: Kiến trúc mô hình mạng

Hệ thống mạng gồm một Router R1 được kết nối với Router ISP. Router ISP được kết nối bởi một host Attacker mô phỏng các client từ bên ngoài của hệ thống mạng. Router R1 kết nối với 2 OVSKernelSwitch s1, s2 tương ứng với 2 VLAN:

* VLAN 10: 192.168.0.0/24 (Switch s1)
* VLAN 20: 192.168.1.0/24 (Switch s2)

Trong đó Switch s1 kết nối với 3 host h1, h2, h3. Switch s2 kết nối với 3 host h4, h5, h6. Host h1 được cấu hình như một Web Server trong hệ thống mạng có chạy trên địa chỉ ip 192.168.0.1 ở port 80. Việc cấu hình định tuyến trên Router cũng như các thiết bị khác được định nghĩa bằng chương trình Python.

## Thu thập dữ liệu

### Mô tả các bước trong quá trình thu thập dữ liệu



Hình 6: Các bước trong quá trình thu thập dữ liệu

Hình 4 mô tả quá trình thu thập dữ liệu để xây dựng dataset, dữ liệu được thu thập hai loại lưu lượng là bình thường và lưu lượng trong trường hợp tấn công DDoS. Dữ liệu thu thập từ hai loại lưu lượng được tiến hành trích xuất đặc trưng, nhãn 0 sẽ được gán cho lưu lượng bình thường, nhãn 1 sẽ được gán cho trường hợp có DDoS. Dataset thu thập được thông qua việc giám sát traffic trên RYU-controller. Các đặc trưng được tham thảo và phát triển từ [3].

### Mô tả dữ liệu

Mô tả:

* + Tên file: DDos\_and\_Normal\_Traffic\_Dataset.csv.
  + Kích thước: 113MB.
  + Cấu trúc: 553968 dòng, 28 cột.
  + Các đặc trưng được mô tả trong bảng 4

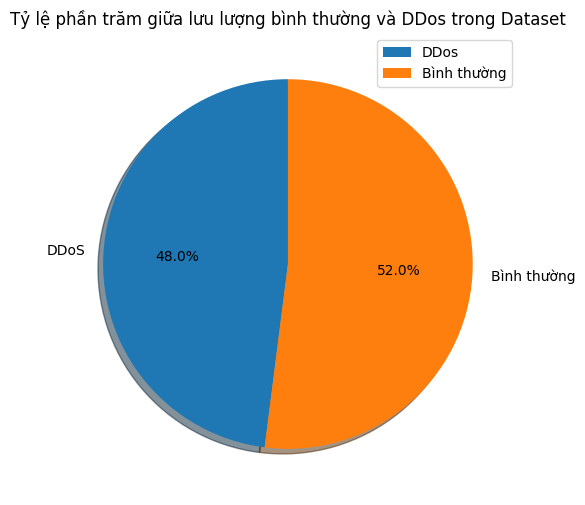
|  |  |
| --- | --- |
| **Tên trường** | **Mô tả** |
| timestamp | Thời gian thu thập thống kê, ở định dạng timestamp (số giây từ thời điểm bắt đầu của epoch). |
| datapath\_id | ID của datapath (switch OpenFlow) mà thống kê được thu thập từ đó. |
| flow\_id | ID được tính toán dựa trên thông tin của luồng, bao gồm địa chỉ IP nguồn, cổng nguồn, địa chỉ IP đích, cổng đích và giao thức IP. |
| ip\_src | Địa chỉ IP nguồn của luồng. |
| tp\_src | Cổng nguồn của luồng (TCP hoặc UDP). |
| ip\_dst | Địa chỉ IP đích của luồng. |
| tp\_dst | Cổng đích của luồng (TCP hoặc UDP). |
| ip\_proto | Giao thức IP của luồng (ví dụ: 1 cho ICMP, 6 cho TCP, 17 cho UDP). |
| icmp\_code | Mã ICMP (nếu giao thức IP là ICMP). |
| icmp\_type | Loại ICMP (nếu giao thức IP là ICMP). |
| flow\_duration\_sec | Thời gian tồn tại của luồng (tính bằng giây). |
| flow\_duration\_nsec | Thời gian tồn tại của luồng (tính bằng nano giây). |
| idle\_timeout | Giá trị idle timeout của luồng (thời gian tối đa luồng không hoạt động trước khi bị xóa). |
| hard\_timeout | Giá trị hard timeout của luồng (thời gian tối đa luồng tồn tại trước khi bị xóa). |
| flags | Các cờ liên quan đến luồng. |
| packet\_count | Số lượng gói tin trong luồng. |
| byte\_count | Lượng dữ liệu (tính bằng byte) trong luồng. |
| packet\_count\_per\_second | Tốc độ gói tin mỗi giây của luồng. |
| packet\_count\_per\_nsecond | Tốc độ gói tin mỗi nano giây của luồng. |
| byte\_count\_per\_second | Tốc độ dữ liệu mỗi giây của luồng (tính bằng byte/giây). |
| byte\_count\_per\_nsecond | Tốc độ dữ liệu mỗi nano giây của luồng (tính bằng byte/nano giây). |
| avg\_packet\_size | Kích thước trung bình của các gói tin trong một luồng dữ liệu, tính bằng byte. |
| flow\_duration\_total | Tổng thời gian từ khi luồng bắt đầu đến khi luồng kết thúc, tính bằng giây |
| idle\_mean | Thời gian nhàn rỗi trung bình |
| idle\_std | Độ lệch chuẩn thời gian nhàn rỗi |
| idle\_max | Thời gian nhàn rỗi lớn nhất |
| idle\_min | Thời gian nhàn rỗi nhỏ nhất |
| label | Được sử dụng để gán nhãn cho các luồng khi sử dụng dữ liệu cho mục đích huấn luyện mô hình ML. |

Bảng 4: Mô tả các đặc trưng của dữ liệu

## Phân tích dữ liệu, xử lý dữ liệu, trích xuất các đặc trưng

### Phân tích dữ liệu

Dataset gồm 288139 traffic nhãn DDoS và 265838 traffic nhãn bình thường. Biểu đồ (Hình 7) cho thấy tỉ lệ nhãn tương đối cân bằng, 48% DDoS so với 52% bình thường.



Hình 7: Biểu đồ thể hiện tỷ lệ phần trăm giữa lưu lượng bình thường và DDos trong Dataset

### Xử lý dữ liệu

Vì dữ liệu được thu thập theo từng đợt nên các nhãn sẽ phân bố không đều, làm cho mô hình có thể gặp tình trạng overfitting, thiên vị, v.v. Do đó, cần xử lý để nhãn phân bố đều hơn. Bên cạnh đó, cần loại bỏ những đặc trưng không có giá trị, kiểm tra các dữ liệu bị null và loại bỏ nếu có, biến đổi các đặc trưng dạng string sang số để phù hợp với đầu vào của mô hình học máy.

### Trích xuất các đặc trưng cho việc training các mô hình học máy

Để lựa chọn được một mô hình học máy có độ chính xác cao và tối ưu nhất. Nhóm em lần lượt trích xuất 4, 10, 15 và tất cả các đặc trưng trong bộ dữ liệu. Sau đó tiến hành training với các giải thuật KNN, Logistic Regression, Neural Network, SVM, GaussianNB. Sau khi so sánh độ chính xác giữa các mô hình với từng đặc trưng được trích xuất thì thấy mô hình KNN với 15 đặc trưng cho độ chính xác cao nhất 98.43%, kết quả thế hiện chi tiết trong bảng 5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên mô hình | Accuracy | Precision | Recall | F1 Score | AUC |
| KNN | 98.433156 | 98.433138 | 98.433156 | 98.433130 | 0.994997 |
| Logistic Regression | 90.362107 | 90.362819 | 90.362107 | 90.359664 | 0.960399 |
| Neural Network | 51.996823 | 27.036696 | 51.996823 | 35.575344 | 0.500000 |
| SVM | 39.915520 | 23.606813 | 39.915520 | 29.667620 | 0.085243 |
| GaussianNB | 94.072710 | 94.286569 | 94.072710 | 94.056426 | 0.993968 |

Bảng 5: Kết quả quấn luyện các mô hình học máy với 15 đặc trưng

Mười lăm đặc trưng được trích xuất cho kết quả tốt nhất là: 'ip\_src', 'ip\_dst', 'flow\_duration\_nsec', 'flags', 'packet\_count', 'flow\_duration\_sec', 'byte\_count', 'packet\_count\_per\_second', 'byte\_count\_per\_second', 'avg\_packet\_size', 'flow\_duration\_total', 'idle\_mean', 'idle\_std', 'idle\_max', 'idle\_min'.

# THỰC NGHIỆM

## Các công nghệ cần chuẩn bị

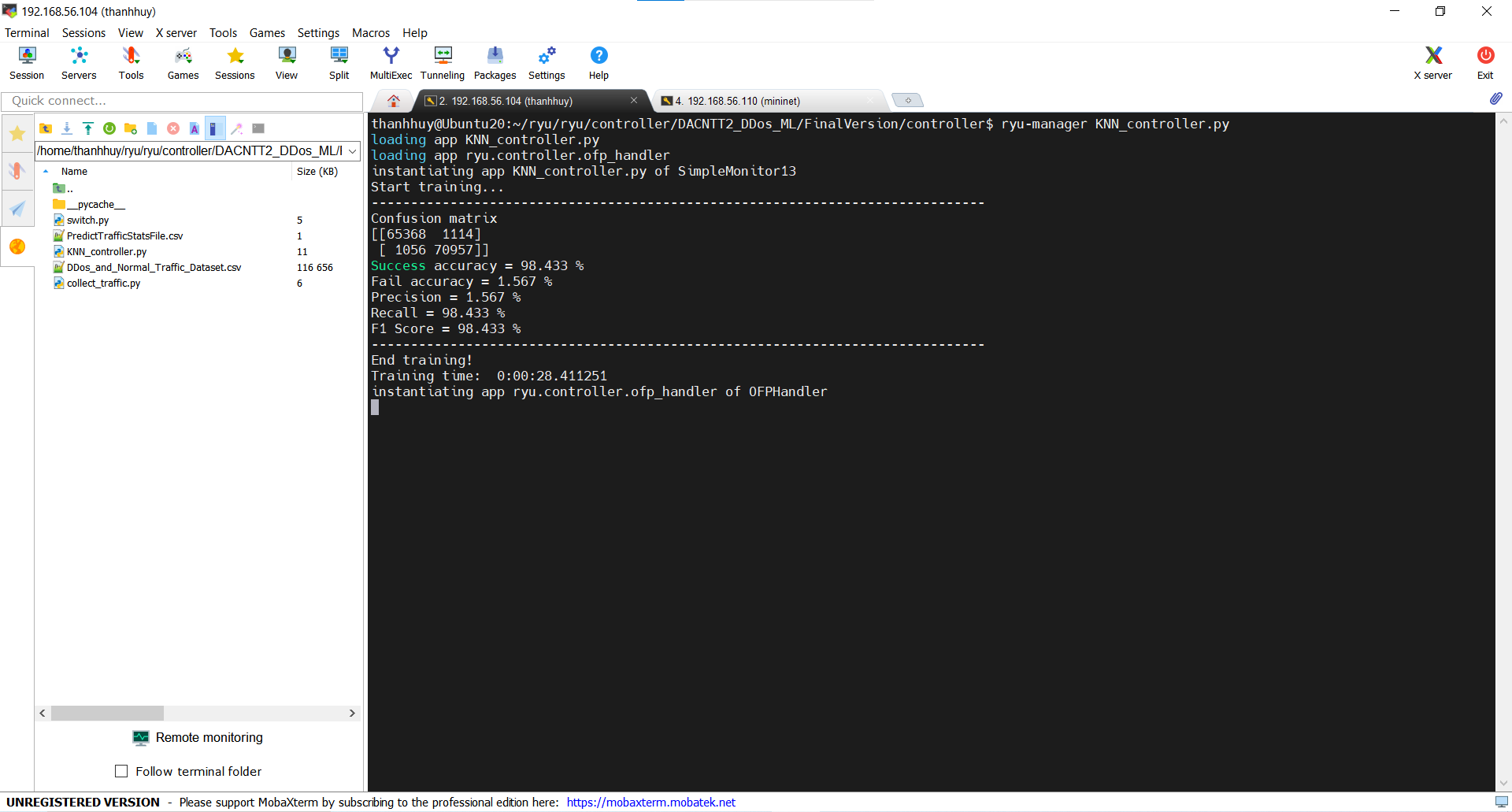
* Một máy ảo Ubuntu (20.04.8 LTS) được cài đặt RYU controller
* Một máy ảo Mininet VM
* Một phần mềm hỗ trợ kết nối shh tới hai máy ảo cho thuận tiện thao tác thực nghiệm. Ở đây nhóm sử dụng MobaXterm.

## Thực nghiệm phát hiện DDos

### Trên máy Ubuntu ryu

ryu-manager KNN\_controller.py

Chạy lệnh trên khởi động controller đã được tích hợp mô hình học máy KNN. KNN\_controller đảm nhiệm giám sát, điều khiển và phát hiện khi có tấn công DDos cho hệ thống mạng. Hình 8 thể hệ controller đã được khởi động thành công và đang tiến hành điều khiển hệ thống mạng.

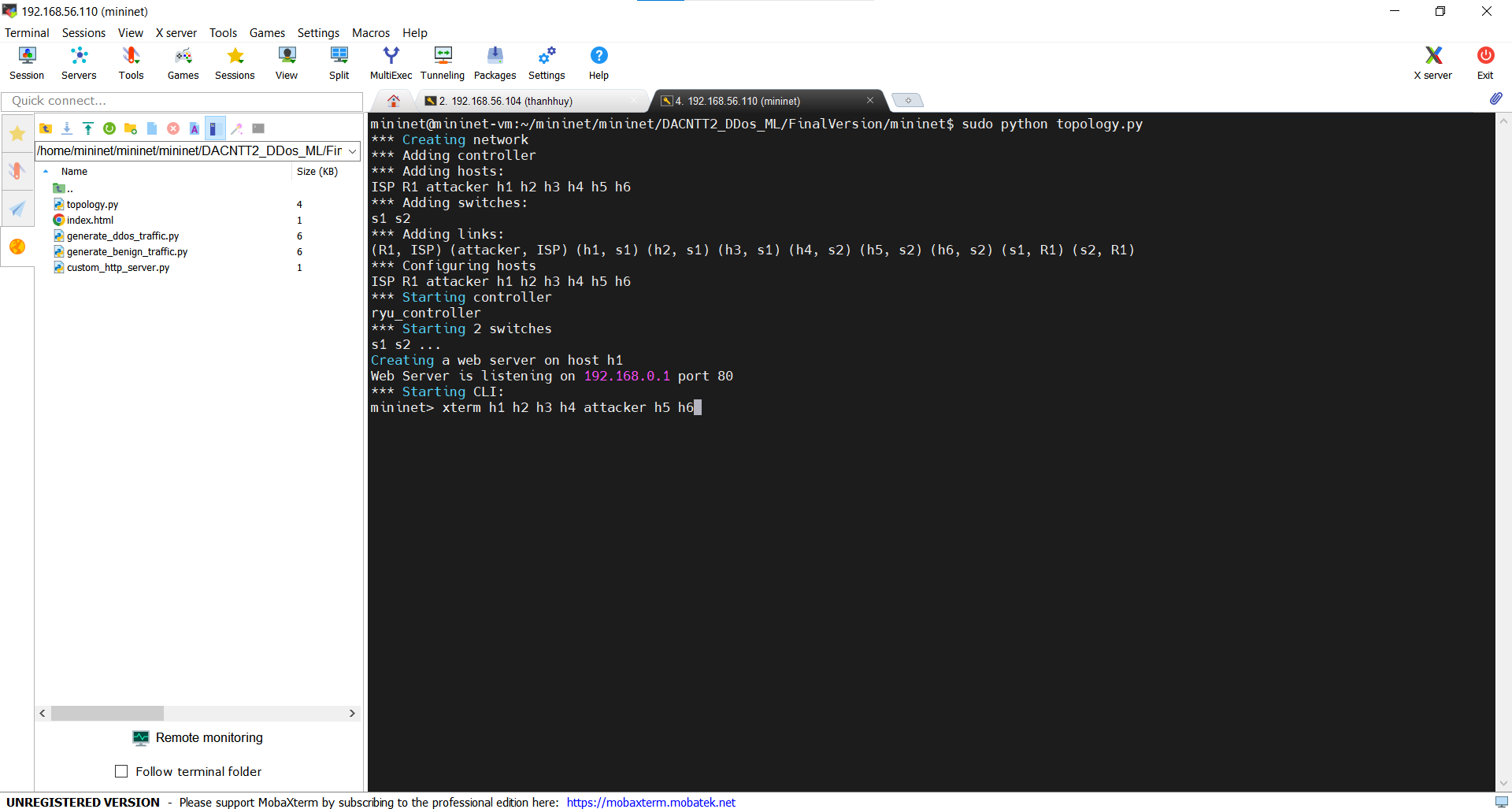


Hình 8: Khởi động controller thành công

### Trên máy Mininet VM

sudo python topology.py

Do hệ thống mạng được cấu hình và xây dựng bằng chương trình Python, nên cần chạy lệnh trên để khởi động hệ thống mạng. Hình 9 cho thấy các thiết bị, dịch vụ trong hệ thống mạng đã được khởi tạo thành công.



Hình 9: Khởi động hệ thống mạng bằng python

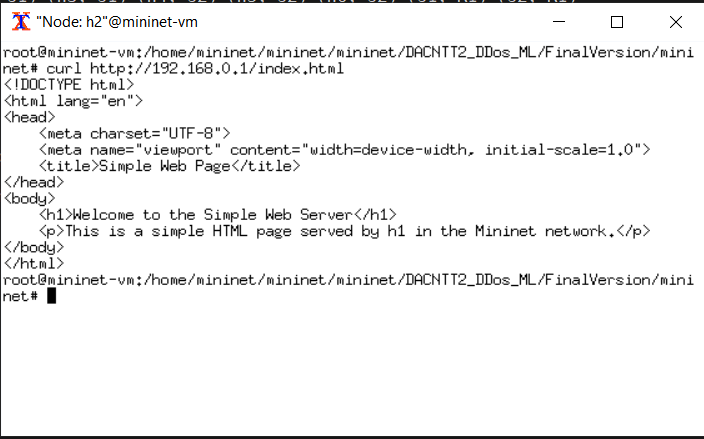
Lệnh `xterm h1 h2 h3 h4 h5 h6 attacker` dùng để mở các terminal của các host. Chạy các lệnh tấn công trên terminal của host attacker để mô phỏng quá trình tấn công DDos vào hệ thống mạng.

## Thực nghiệm phát hiện tấn công và kết quả

### Kiểm tra truy cập tới Web server

curl http://192.168.0.1/index.html

Lệnh trên gửi một request tới trang web <http://192.168.0.1/index.html> với Method GET. Kết quả truy cập tới trang web thành công thể hiện trên hình 10.



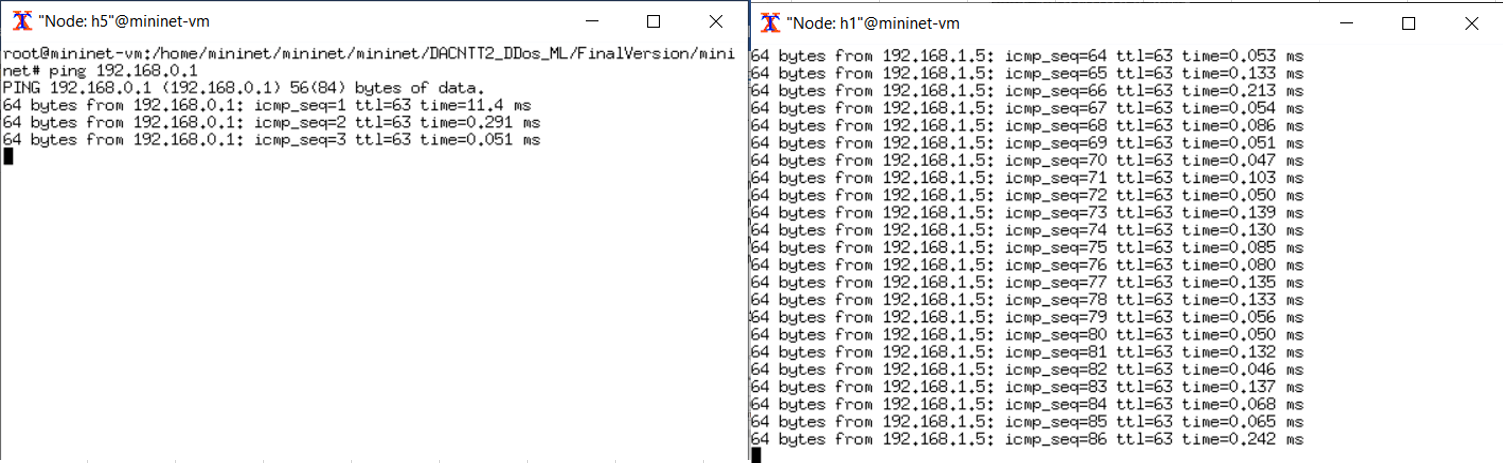
Hình 10: Truy cập tới trang web trên Web Server

Thực hiện các lệnh ping bình thường để kiểm tra traffic bình thường

ping 192.168.0.1

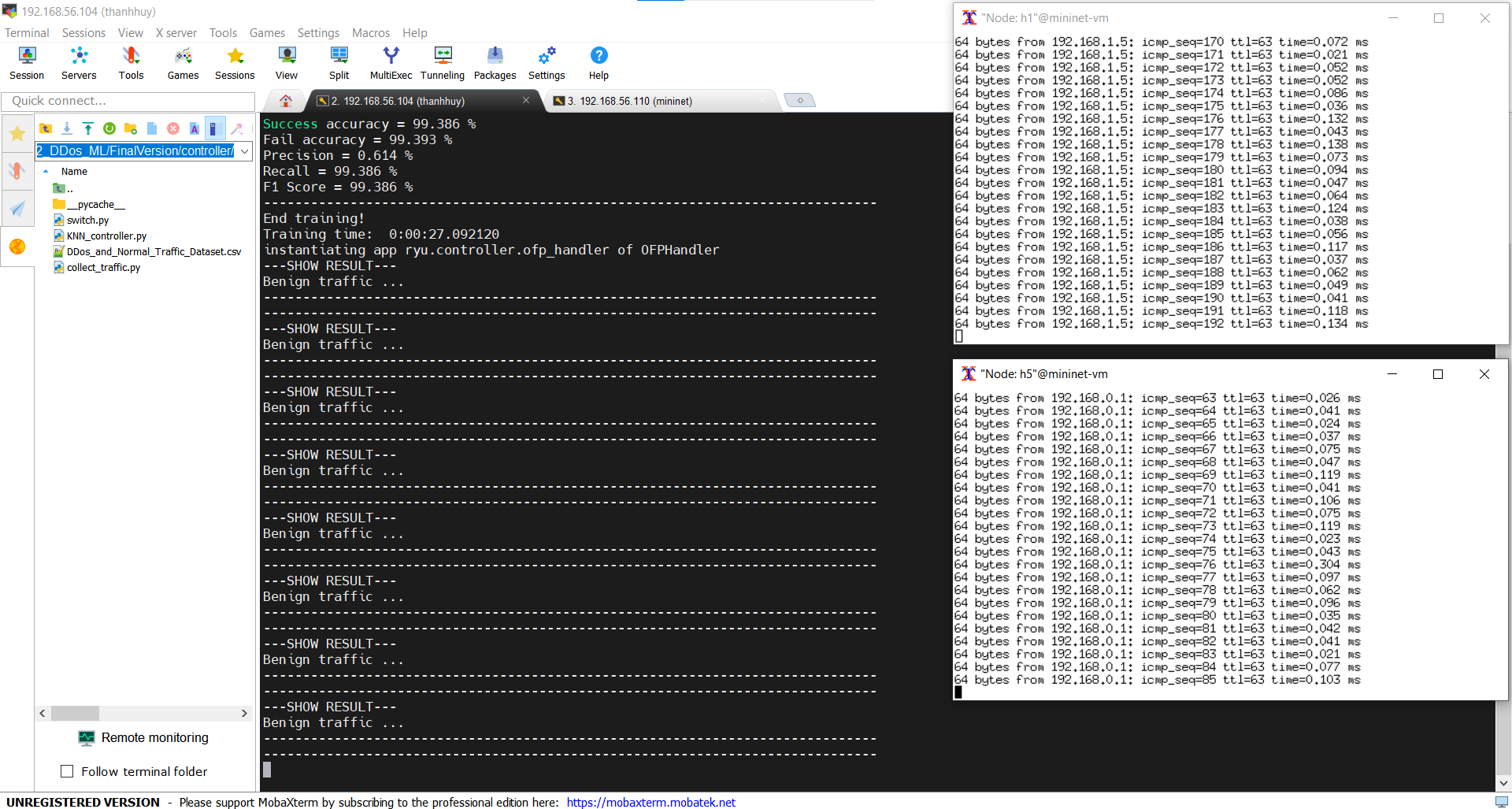
ping 192.168.1.5

Thực hiện ping kiểm tra các host trong mạng có thể truy cập được với nhau thông qua lệnh ping. Hình 11 cho thấy host h1 đã ping được tới h1 và ngược lại.



Hình 11: Ping giữa các thiết bị

Sau khi đã thực một một số lệnh ping giữa các thiết bị, gửi các request tới Web Server thì mô hình KNN được tích hợp trong ryu-controller đã giám sát và đựa ra dự đoán các lưu lượng là bình thượng, kết quả như hình 12.



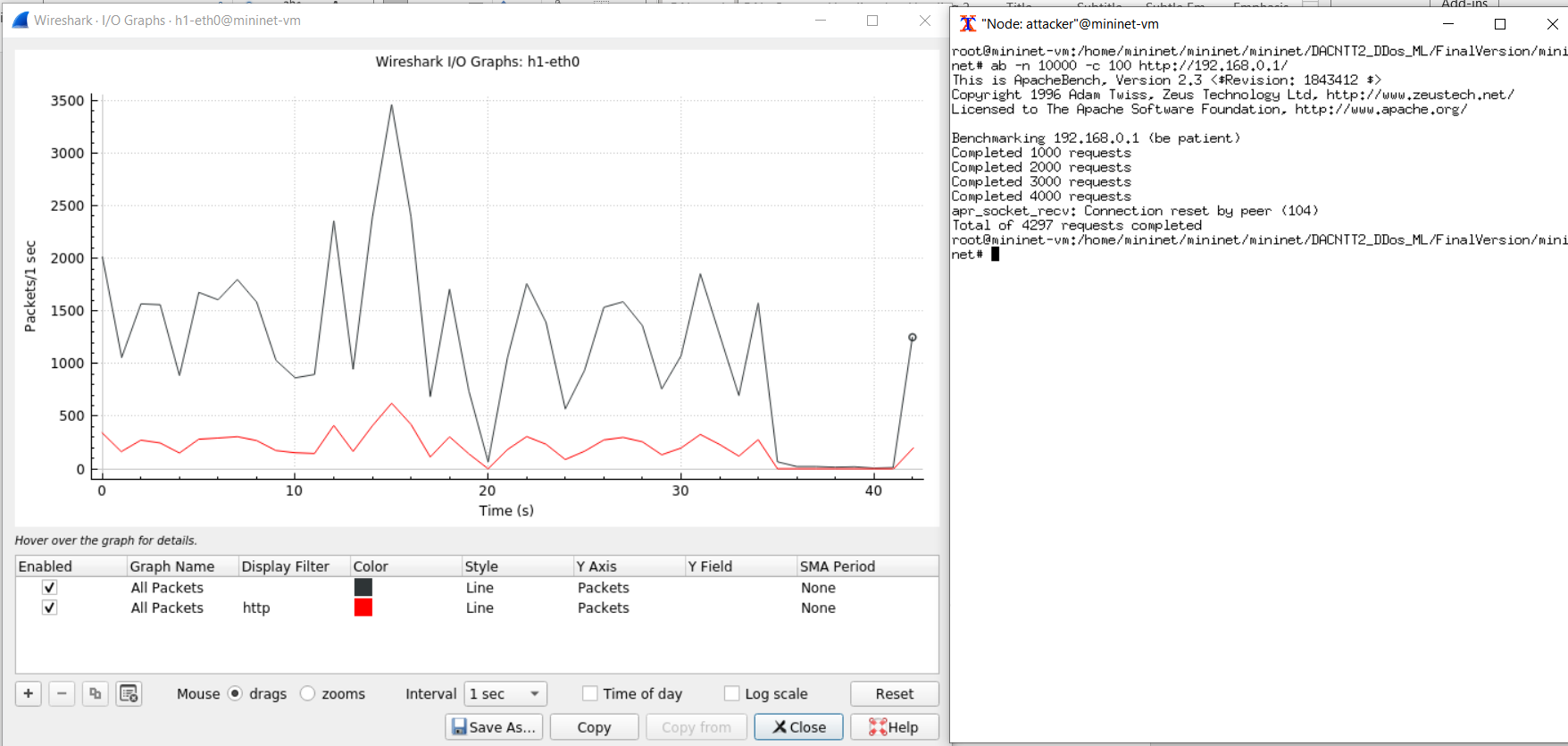
Hình 12: Kết quả dự đoán lưu lượng bình thường

### Thực hiện tấn công HTTP Flood tới Web Server

ab -n 10000 -c 100 <http://192.168.0.1/>

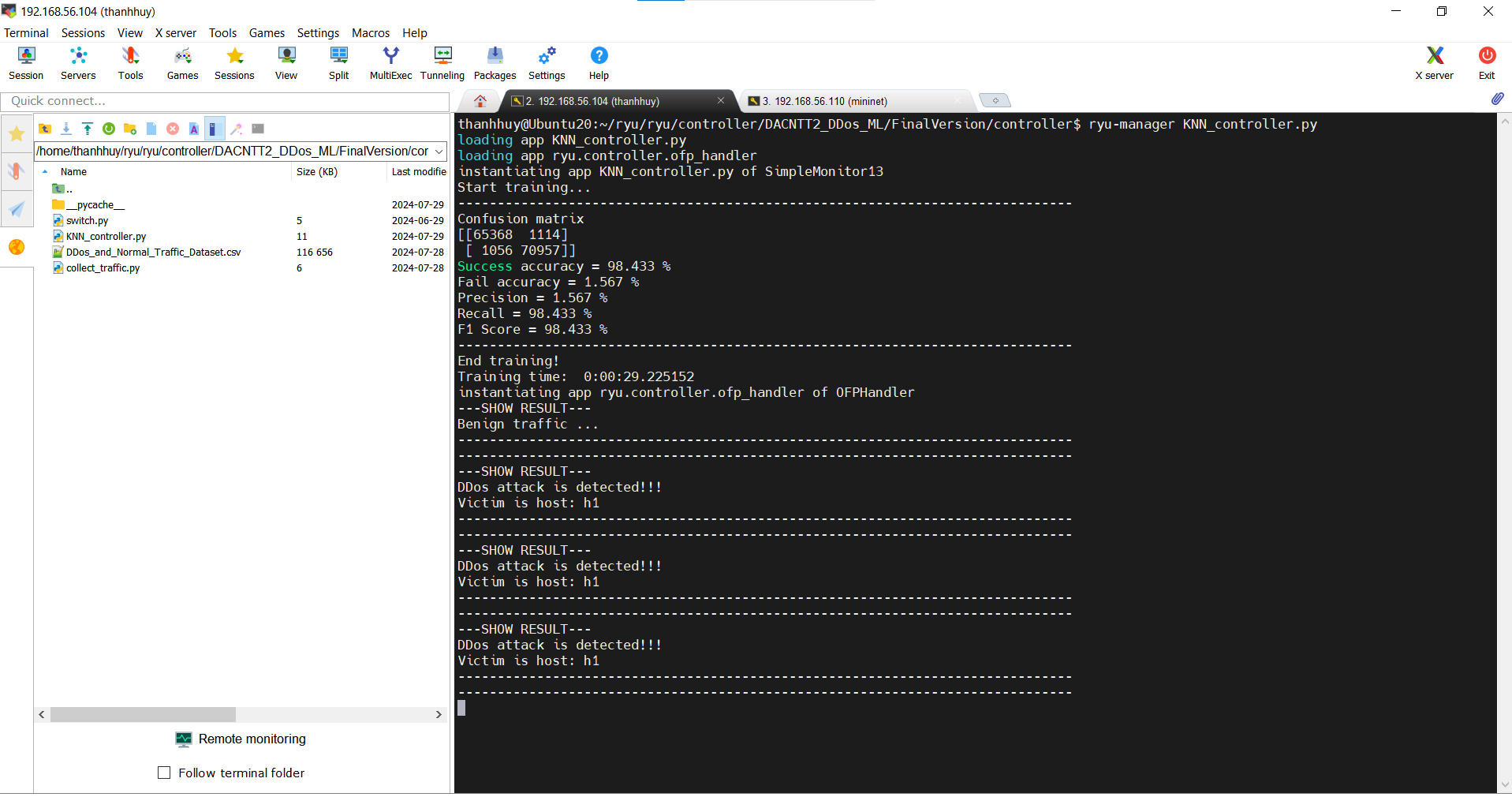
Lệnh trên tạo ra 10000 request với 100 kết nối đồng thời được gửi tới Web server (có thể tùy chỉnh các thông số)

Biểu đồ dưới hiển thị các requests đã được gửi đến Web Server trong thời gian lệnh HTTP Flood được thực thi. Theo biểu đồ cho thấy mốc đỉnh cao nhất là 3500 gói tin/giây và khoảng 700 gói tin http/giây.



Hình 13: Biểu đồ lưu lượng trên Web Server khi bị tấn công HTTP Flood

Như kết quả dự đoán hiển thị trên hình 15, Model KNN xây dựng trong ryu-controller đã dự đoán chính xác Web Server h1 đang bị tấn công khi lệnh HTTP Flood được thực thi.



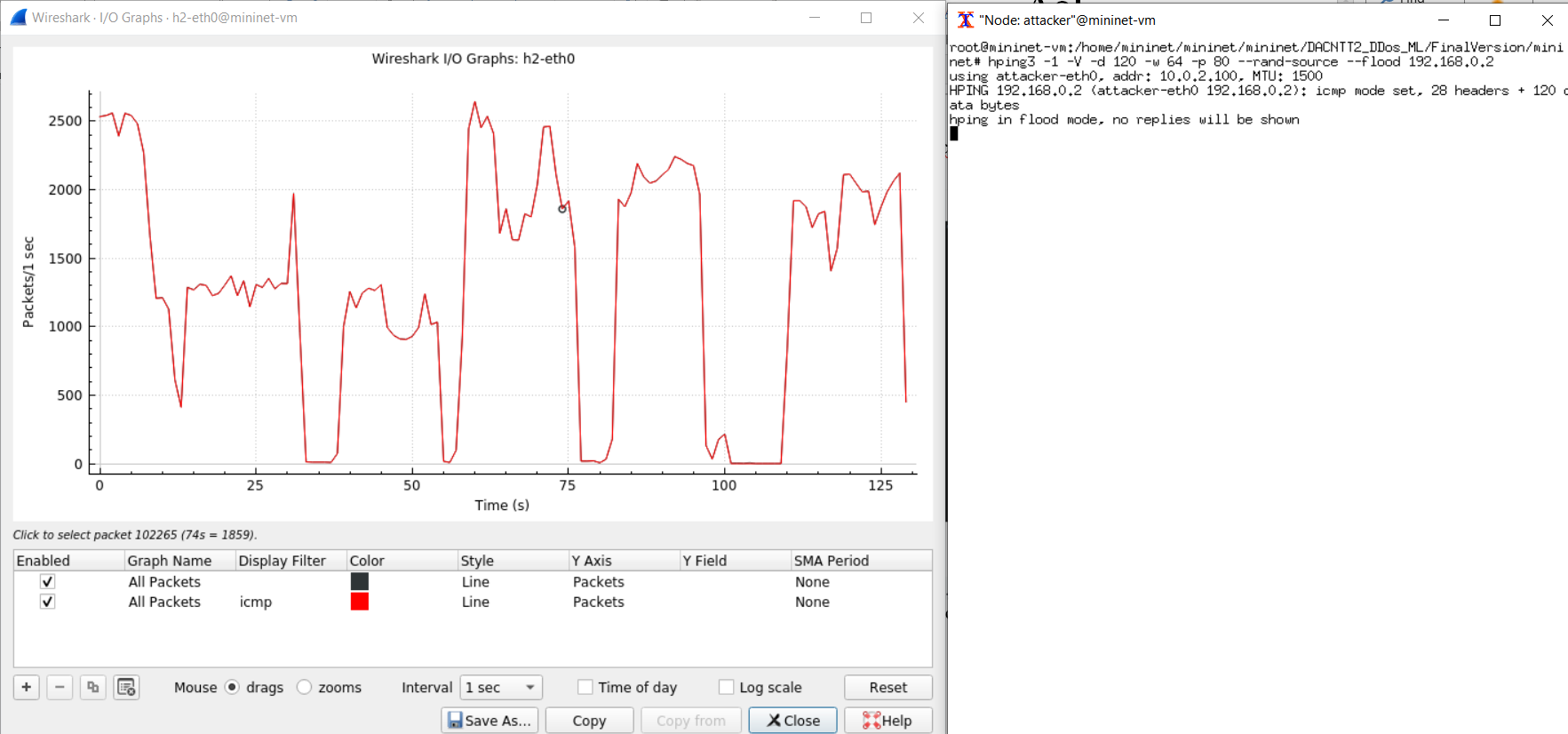
Hình 14: Kết quả dự đoán Web Server bị tấn công

### Thực hiện tấn cống ICMP Flood tới h2

hping3 -1 -V -d 120 -w 64 -p 80 --rand-source –flood 192.168.0.2

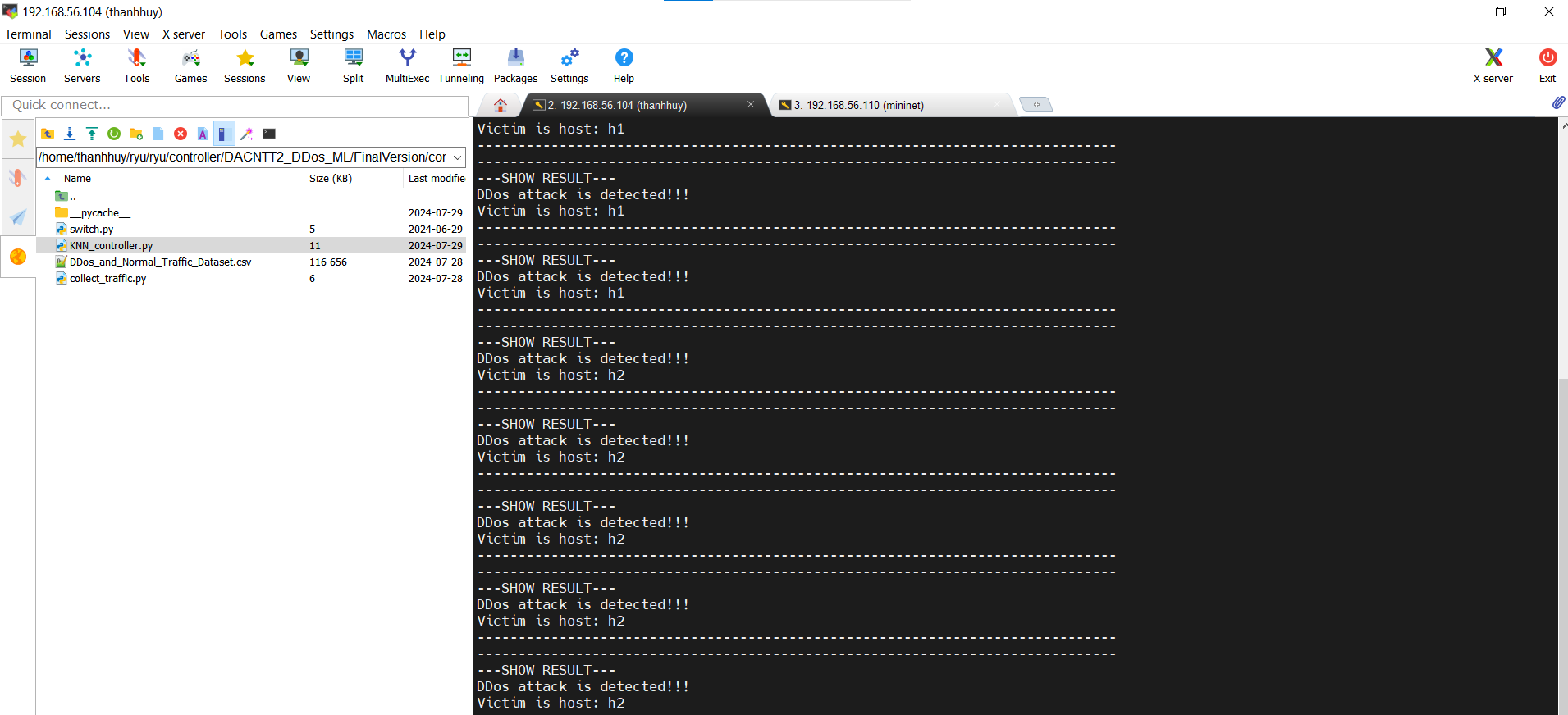
Lệnh trên dùng để tạo ra một lượng lớn các gói tin ICMP Echo Request tới host h2.

Biểu đồ hình 16 hiển thị các request ICMP đã được tới host h2 trong khi thời gian lệnh ICMP Flood được thực thi.



Hình 15: Biểu đồ lưu lượng trên host h2 khi bị tấn công ICMP Flood

Như kết quả dự đoán hiển thị trên hình 17, Model KNN xây dựng trong ryu-controller đã dự đoán chính xác host h2 đang bị tấn công.



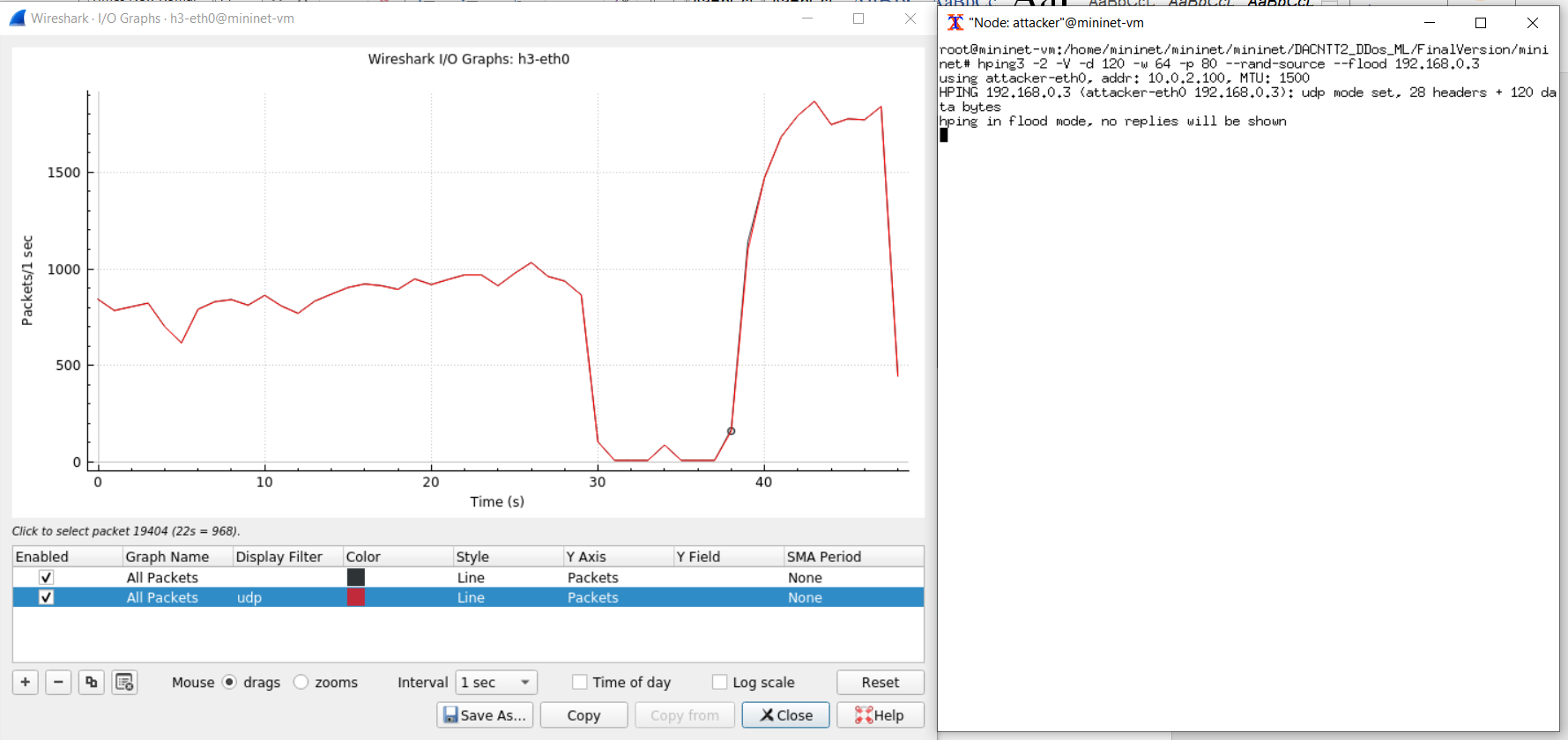
Hình 16: Kết quả dự đoán host h2 bị tấn công

### Thực hiện tấn công UDP Flood tới h3

hping3 -2 -V -d 120 -w 64 -p 80 --rand-source –flood 192.168.0.3

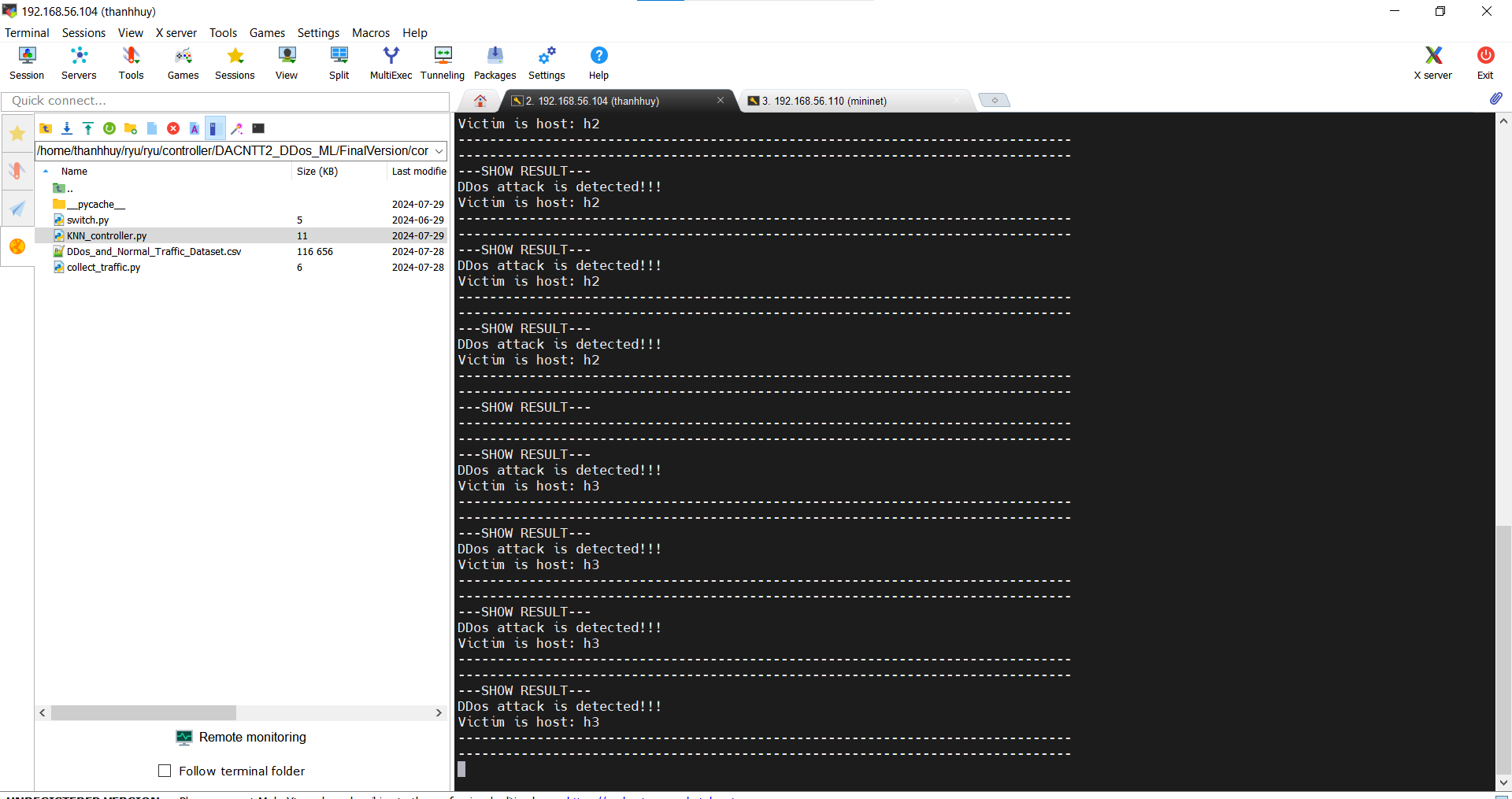
Lệnh trên dùng để tạo ra một lượng lớn các gói tin UDP tới host h3.

Biểu đồ bên dưới hiển thị các traffic UDP đã được gửi tới host h3 trong khi thời gian lệnh UDP Flood được thực thi.



Hình 17: Biểu đồ lưu lượng trên host h3 khi bị tấn công UDP Flood

Như kết quả dự đoán hiển thị trên hình 19, Model KNN xây dựng trong ryu-controller đã dự đoán chính xác host h3 đang bị tấn công.



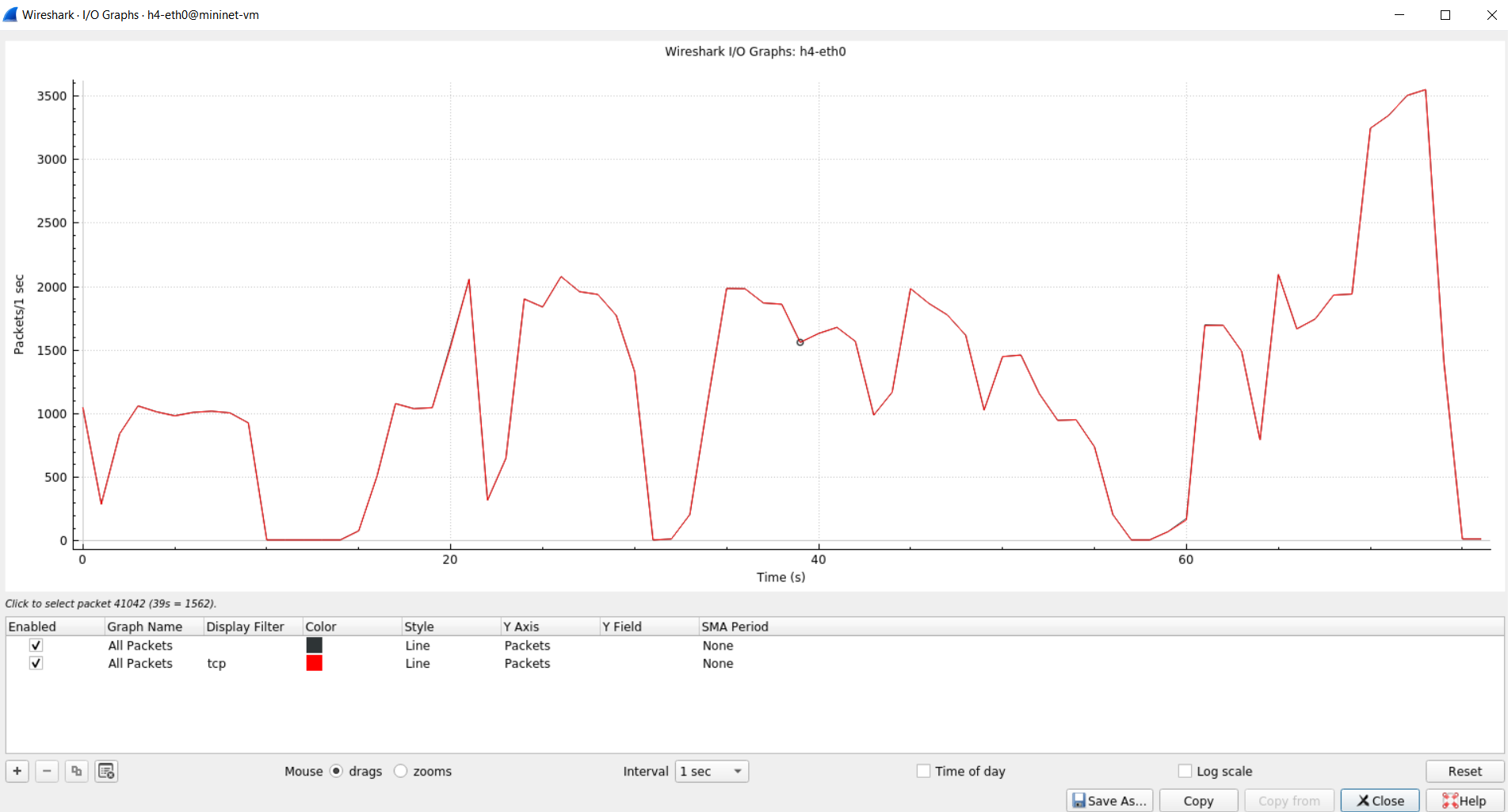
Hình 18: Kết quả dự đoán host h3 bị tấn công

### Thực hiện tấn công TCP-SYN Flood tới h4

hping3 -S -V -d 120 -w 64 -p 80 --rand-source –flood 192.168.1.4

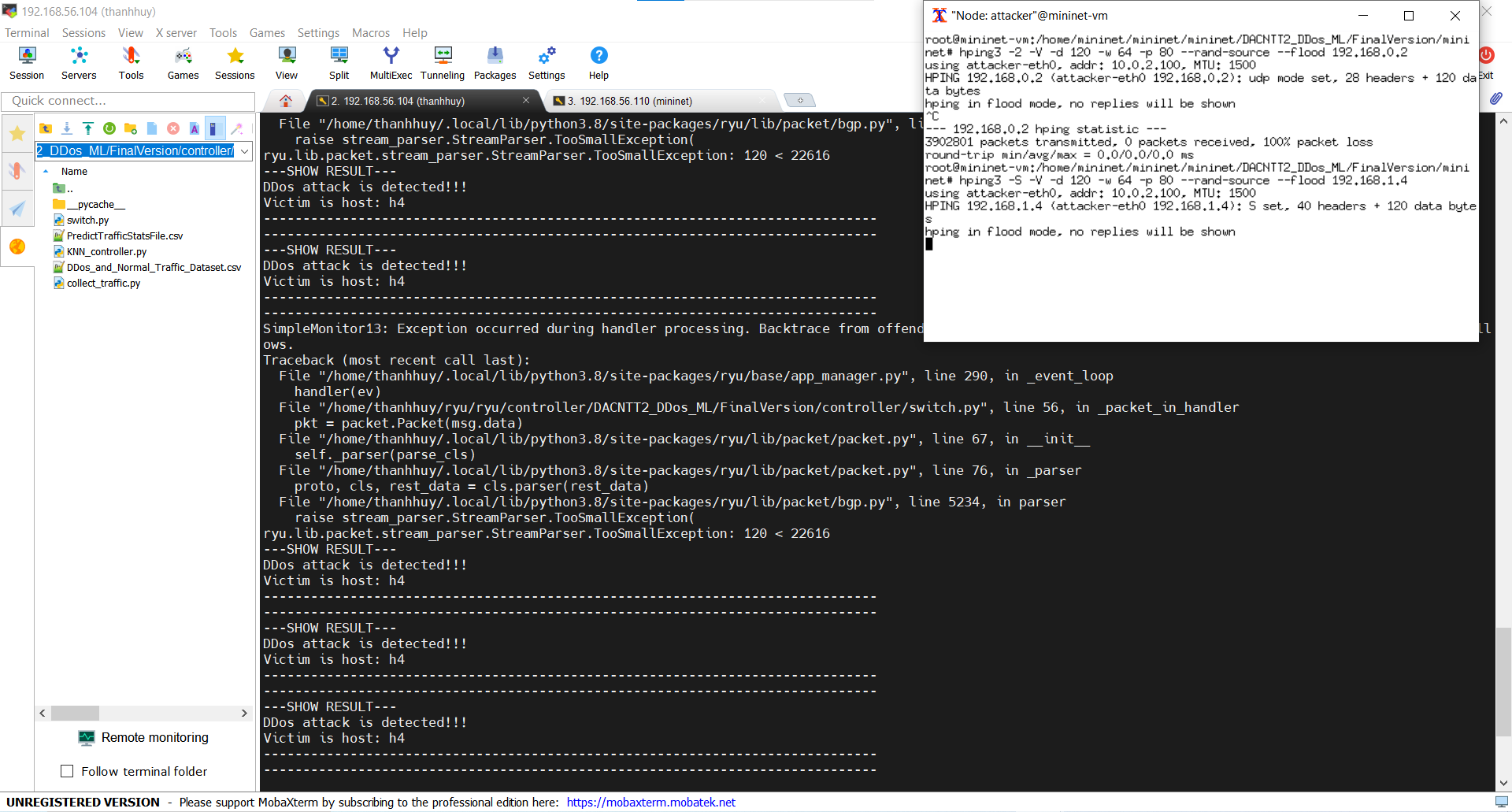
Lệnh trên dùng để tạo ra một lượng lớn các gói tin TCP-SYN tới host h4.

Biểu đồ hình 20 hiển thị các traffic TCP-SYN đã được tới host h4 trong khi thời gian lệnh TCP-SYN Flood được thực thi.



Hình 19: Biểu đồ lưu lượng trên host h4 khi bị tấn công TCP-SYN Flood

Như kết quả dự đoán hiển thị trên hình 21, Model KNN xây dựng trong ryu-controller đã dự đoán chính xác host h4 đang bị tấn công



Hình 20: Kết quả dự đoán host h4 bị tấn công

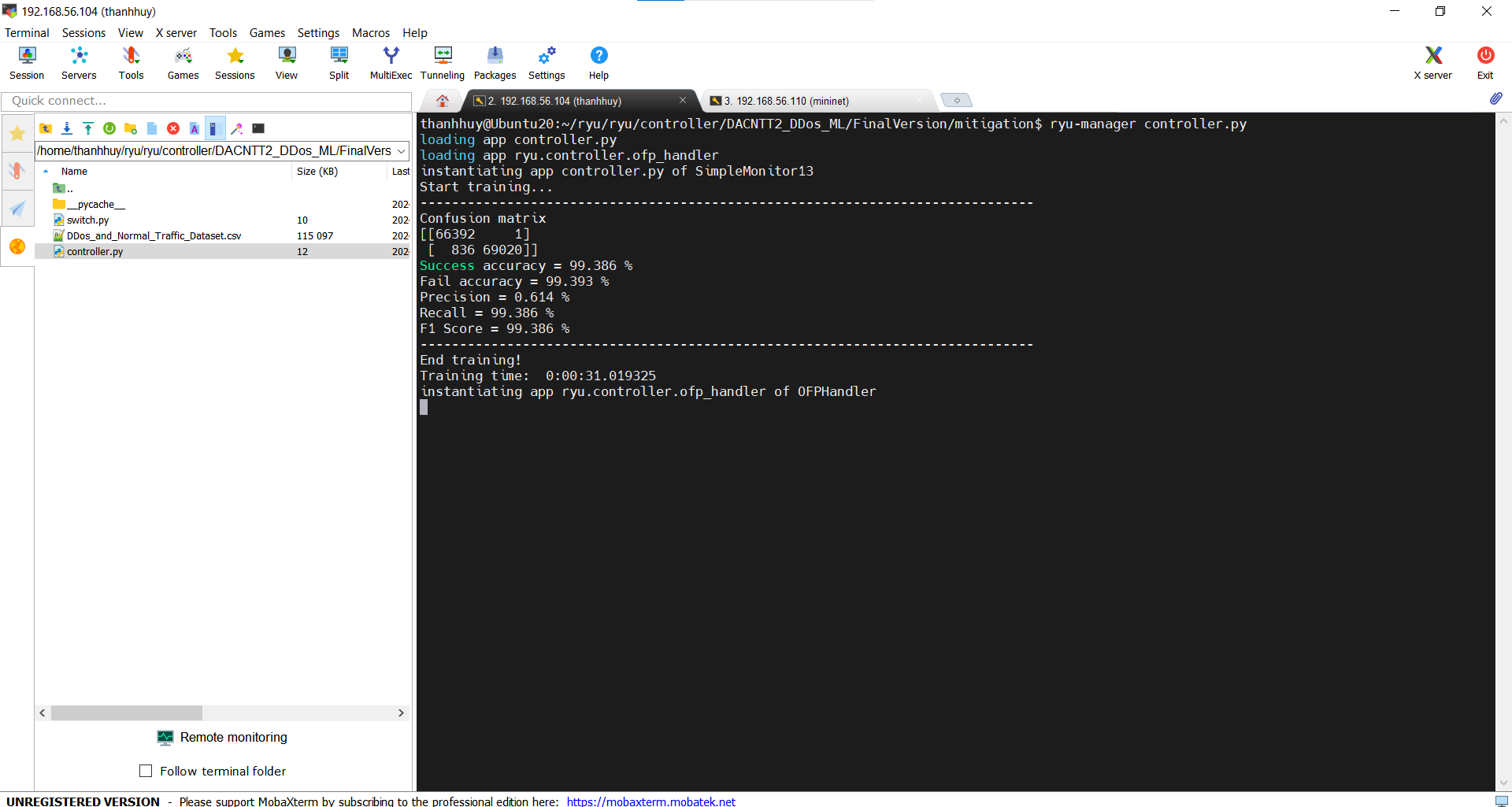
## Thực nghiệm biện pháp ngăn chặn và kết quả

### Trên máy Ubuntu RYU

Chuyển đường dẫn tới phần mitigation, chạy lệnh bên dưới để khởi động ryu-controller có tích hợp mô hình KNN và biện pháp ngăn chặn tấn công DDos.

ryu-manager controller.py

Hình 22 hiển thị đã chạy thành công controller đã tích hợp mô hình KNN và mô hình ngăn chặn block port khi phát hiện host bị tấn công.



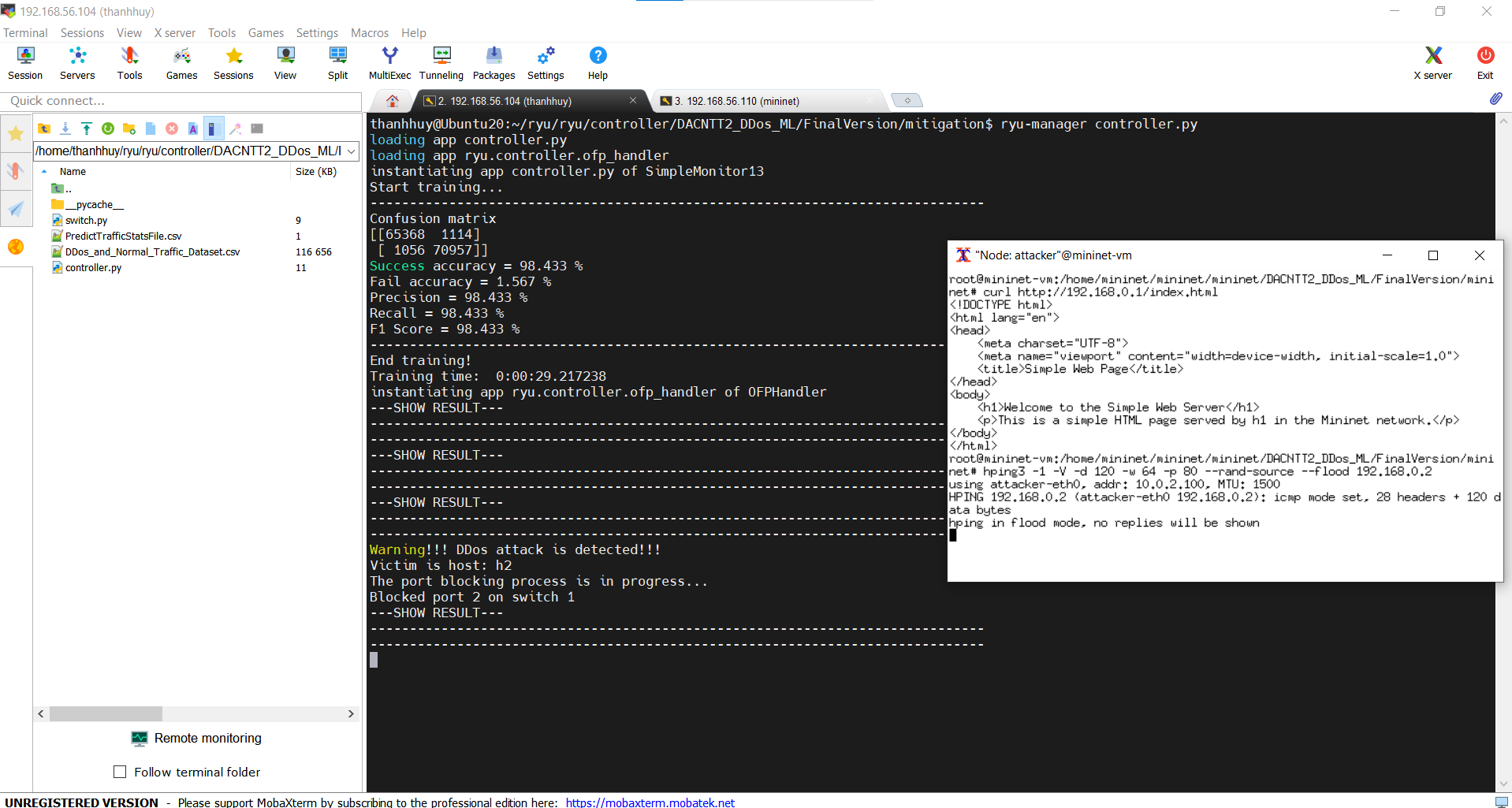
Hình 21: Khởi động thành công controller có tích hợp ngăn chặn tấn công.

### Thực nghiệm tấn công và ngăn chặn block port khi phát hiện DDos

Khởi động lại hệ thống mạng và kiếm tra lại các lưu lượng bình thường trên hệ thống mạng để đảm bảo hệ thống hoạt động bình thường. Chạy lại các lệnh tấn công ở phần 4.3 và kiểm tra sau khi phát hiện bị tấn công DDos thì port của host đó bị block.

#### ICMP Flood tới host h2

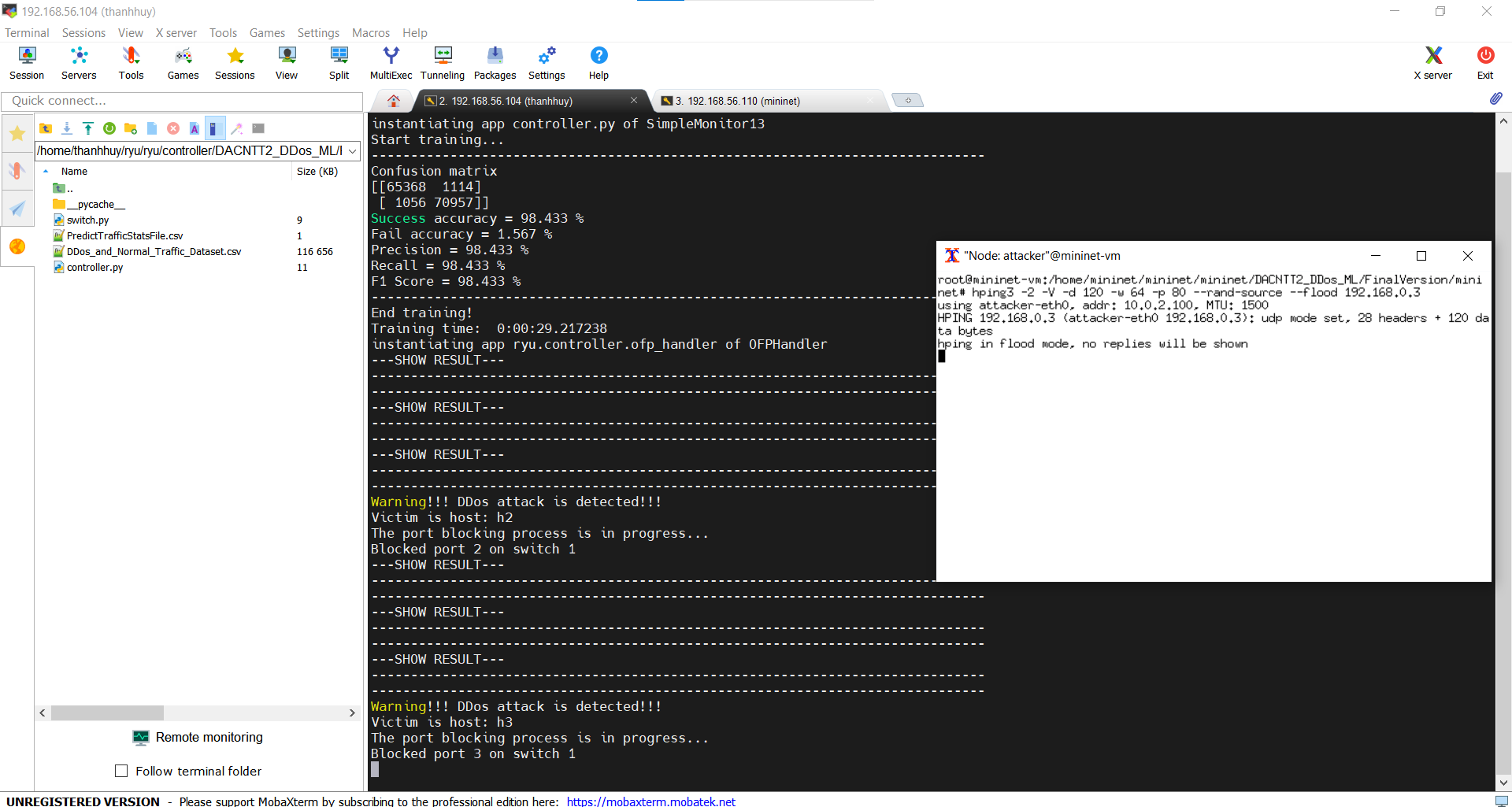
Hình 23 cho thấy hệ thống đã phát hiện được host h2 bị tấn công và tự động chặn cổng đi vào của h1 trên switch 1.



Hình 22: Kết quả phát hiện h2 bị tấn công và tự chặn cổng đi vào của h2

#### Thực hiện tấn công UDP Flood tới host h3

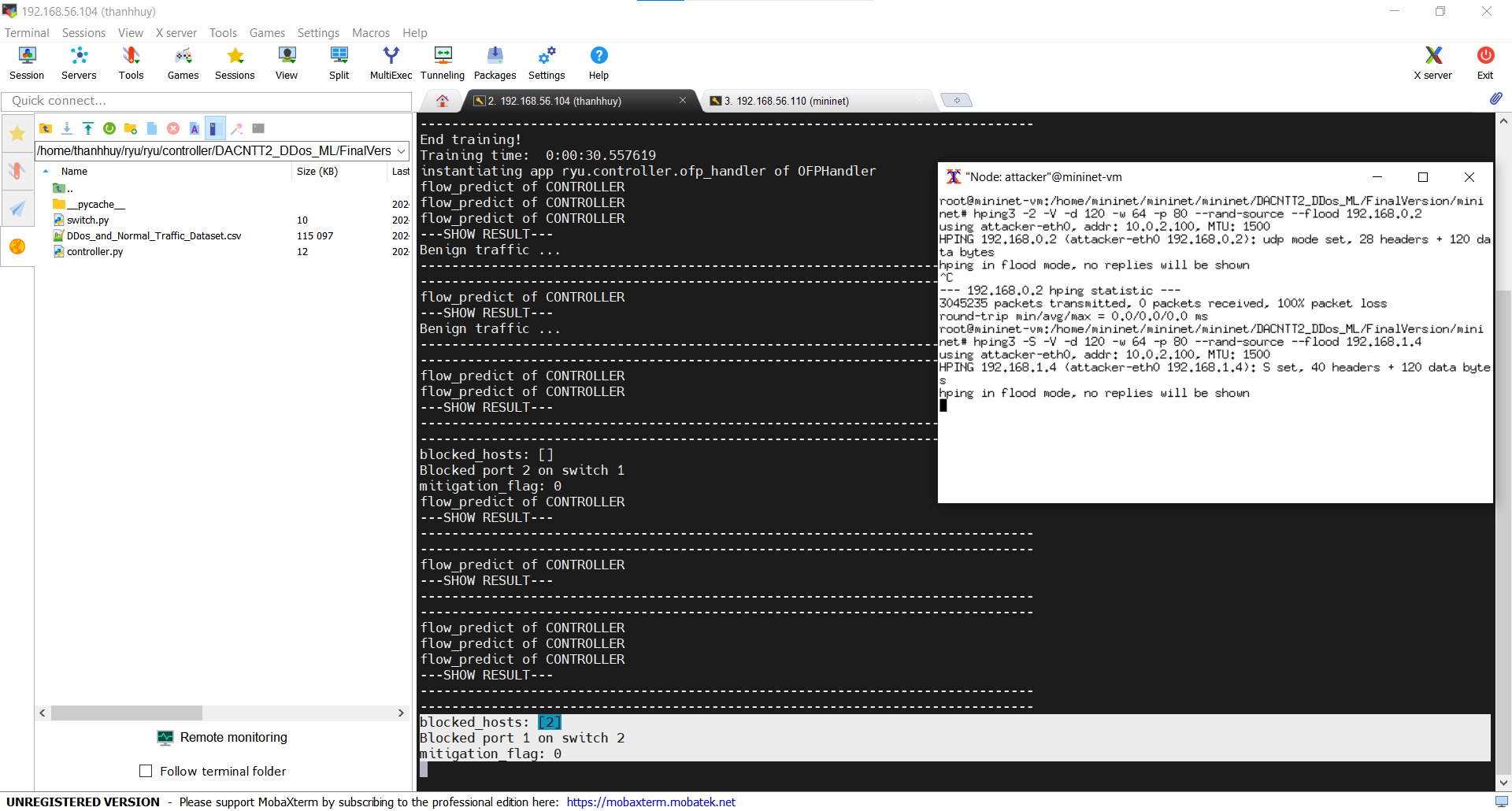
Hình 23 cho thấy hệ thống đã phát hiện được host h2 bị tấn công và tự động chặn cổng đi vào của h3 trên switch 1.



Hình 23: Kết quả phát hiện h3 bị tấn công và tự chặn cổng đi vào của h3

#### Thực hiện tấn công TCP-SYN Flood tới h4

Hình 24 cho thấy hệ thống đã phát hiện được host h4 bị tấn công và tự động chặn cổng đi vào của h4 trên switch 2.



Hình 24: Kết quả phát hiện h4 bị tấn công và tự chặn cổng đi vào của h4

# KẾT LUẬN

## Kết luận

Đánh giá kết quả thu được:

* Triển khai thành công hệ thống mạng SDN trên nền tảng mininet. Xây dựng và cấu hình được hệ thống mạng bằng chương trình Python.
* Tạo ra được một dataset DDos dùng cho việc huấn luyện các mô hình học máy. Huấn luyện và đánh giá các mô hình học máy.
* Giả lập thành công các loại tấn công DDoS, với các loại tấn công phổ biến là HTTP, SYN, IMCP, UDP FLOOD.
* Tích hợp mô hình KNN vào ryu-controller để phát hiện DDos chính xác khi hệ thống bị tấn công.
* Thực hiện giải pháp tự động ngăn chặn cổng đi vào của máy nạn nhân khi hệ thống phát hiện máy đó đang bị tấn công.

## Hướng phát triển

Phát triển hệ thống mạng với quy mô lớn, tương thích với hệ thống mạng thực tế.

Thêm các loại tấn công khác: Giả lập và phát hiện các loại tấn công DDoS khác như DNS Amplification, và các kỹ thuật tấn công khác để đảm bảo hệ thống có khả năng phát hiện rộng rãi và linh hoạt.

Phát triển các biện pháp giảm thiểu nâng cao: Nghiên cứu và triển khai các biện pháp giảm thiểu tấn công thông minh hơn, như tự động chuyển đổi lưu lượng, điều chỉnh chất lượng dịch vụ (QoS), và sử dụng các kỹ thuật lọc lưu lượng tiên tiến.

Tích hợp với các hệ thống bảo mật khác: Kết hợp hệ thống phát hiện và giảm thiểu tấn công với các hệ thống bảo mật khác như tường lửa (Firewall), hệ thống ngăn chặn xâm nhập (IPS), và hệ thống quản lý sự kiện bảo mật (SIEM).

# TÀI LIỆU THAM KHẢO

Tiếng Anh

[1] Al-Shareeda, M. A., Manickam, S., & Ali, M. (2023). DDoS attacks detection using machine learning and deep learning techniques: Analysis and comparison. Bulletin of Electrical Engineering and Informatics, 12(2), 930-939.

[2] Bawany, N. Z., Shamsi, J. A., & Salah, K. (2017). DDoS attack detection and mitigation using SDN: methods, practices, and solutions. Arabian Journal for Science and Engineering, 42, 425-441.

[3] Iman Sharafaldin, Arash Habibi Lashkari and Ali A. Ghorbani. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. 2018

[4] Alaa Taima Albu-Salih. Performance Evaluation of Ryu Controller in Software Defined Networks. 2019

[5] A. Fadlil, I. Riadi, and S. Aji, “Review of detection DDOS attack detection using naive bayes classifier for network forensics,” Bulletin of Electrical Engineering and Informatics, vol. 6, no. 2, pp. 140–148, Jun. 2017, doi: 10.11591/eei.v6i2.605.

[6] R. Vijayasarathy, S. V. Raghavan, and B. Ravindran, “A system approach to network modeling for DDoS detection using a NaiveBayesian classifier,” in 2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011), Jan. 2011, pp. 1–10, doi: 10.1109/COMSNETS.2011.5716474.

[7] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, “A DDoS attack detection method based on SVM in software defined network,” Security and Communication Networks, pp. 1–8, 2018, doi: 10.1155/2018/9804061.

[8] G. Lucky, F. Jjunju, and A. Marshall, “A lightweight decision-tree algorithm for detecting DDoS flooding attacks,” in 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Dec. 2020, pp. 382–389, doi: 10.1109/QRS-C51114.2020.00072.

[9] N. A. Putri, D. Stiawan, A. Heryanto, T. W. Septian, L. Siregar, and R. Budiarto, “Denial of service attack visualization with clustering using K-means algorithm,” in 2017 International Conference on Electrical Engineering and Computer Science (ICECOS), Aug. 2017, pp. 177–183, doi: 10.1109/ICECOS.2017.8167129

[10] A. Chaudhary and G. Shrimal, “Intrusion detection system based on genetic algorithm for detection of distribution denial of service attacks in MANETs,” in Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India, 2019, pp. 370–377, doi: 10.2139/ssrn.3351807.

[11] V. N. Vapnik, “An overview of statistical learning theory,” IEEE Transactions on Neural Networks, vol. 10, no. 5, pp. 988–999, 1999, doi: 10.1109/72.788640.

[12] T. Subbulakshmi, P. Parameswaran, C. Parthiban, M. Mariselvi, J. A. Anusha, and G. Mahalakshmi, “A unified approach for detection and prevention of ddos attacks using enhanced support vector machines and filtering mechanisms,” ICTACT Journal on Communication Technology, vol. 4, no. 2, pp. 737–743, Jun. 2013, doi: 10.21917/ijct.2013.0105.

[13] J. Singh and M. J. Nene, “A survey on machine learning techniques for intrusion detection systems,” International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 11, pp. 4349–4355, 2013.

[14] M. Zekri, S. El Kafhali, N. Aboutabit, and Y. Saadi, “DDoS attack detection using machine learning techniques in cloud computing environments,” in 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), Oct. 2017, pp. 1–7, doi: 10.1109/CloudTech.2017.8284731.

[15] A. Nazir, “A comparative study of different artificial neural networks based intrusion detection systems,” International Journal of Scientific and Research Publications, vol. 3, no. 7, pp. 2250–3153, 2013, doi: 10.1.1.415.1490.

[16] A. Saied, R. E. Overill, and T. Radzik, “Detection of known and unknown DDoS attacks using Artificial Neural Networks,” Neurocomputing, vol. 172, pp. 385–393, Jan. 2016, doi: 10.1016/j.neucom.2015.04.101.

[17] E. Hodo et al., “Threat analysis of IoT networks using artificial neural network intrusion detection system,” in 2016 International Symposium on Networks, Computers and Communications (ISNCC), May 2016, pp. 1–6, doi: 10.1109/ISNCC.2016.7746067.

[18] X. Hao, B. Meng, and K. Gu, “Detecting DDoS attack based on PSO clustering algorithm,” in Proceedings of the 2016 3rd International Conference on Materials Engineering, Manufacturing Technology and Control, 2016, pp. 670–674, doi: 10.2991/icmemtc-16.2016.133.

[19] N. C. S. N. Iyengar and G. Ganapathy, “Chaotic theory based defensive mechanism against distributed denial of service attack in cloud computing environment,” International Journal of Security and Its Applications, vol. 9, no. 9, pp. 197–212, Sep. 2015, doi: 10.14257/ijsia.2015.9.9.18.

[20] C. Balarengadurai and S. Saraswathi, “Fuzzy based detection and prediction of DDoS attacks in IEEE 802.15. 4 low rate wireless personal area network,” International Journal of Computer Science Issues (IJCSI), vol. 10, no. 6, pp. 293–301, 2013.

[21] P. G. Majeed and S. Kumar, “Genetic algorithms in intrusion detection systems: A survey,” International Journal of Innovation and Applied Studies, vol. 5, no. 3, pp. 233–240, 2014