Thiết kế và triển khai hệ thống phát hiện xâm nhập dựa trên kĩ thuật máy học

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Nguyễn Đức Thông
Phạm Ngọc Hiếu Minh
Trần Thanh Tín

Ngày 25 tháng 6 năm 2018

Mục lục

Danh sách hình vẽ

Danh sách bảng

Chương 1

Giới thiệu

1.1 Đề tài

Thiết kế và triển khai hệ thống phát hiện xâm nhập dựa trên kĩ thuật máy học

1.2 Đặt vấn đề

Ngày nay, mạng máy tính đang trở nên phổ biến hơn và được sử dụng rộng rãi trong hầu hết các lĩnh vực trên toàn thế giới. Tuy nhiên, đi kèm với sự phát triển và phổ biến của các mạng này là những rủi ro và thách thức liên quan đến chúng, đặc biệt là các vấn đề an ninh mạng và bảo mật dữ liệu.

Theo Mạng lưới Bảo mật Việt Nam (VSEC), 70% trang web ở Việt Nam có thể xâm nhập và 80% hệ thống mạng có thể được kiểm soát bởi tin tặc. Trong văn bản này, sự phát triển của các hệ thống phát hiện xâm nhập (IDS) quan trọng hơn bao giờ hết, phát triển IDS trở nên phổ biến và đóng một vai trò vô cùng quan trọng trong bất kỳ chính sách, an ninh và an toàn nào của bất kỳ hệ thống thông tin nào. Đối với những điều đã đề cập ở trên, chúng tôi nghiên cứu về phát hiện xâm nhập và thuật toán K-means làm thuật toán phân cụm chính cho việc học máy. Ngoài thuật toán K-means, NSL-KDD là một tập dữ liệu hiệu quả được phát triển để giúp các nhà nghiên cứu so sánh các phương pháp phát hiện xâm nhập khác nhau.

1.3 Mục tiêu

Hệ thống ứng dụng IDS của chúng tôi được sử dụng làm bộ lọc, lưu lượng mạng được truyền qua bộ lọc của chúng tôi sẽ được phân tích và tính toán. Kết quả sẽ được báo cáo cho người giám sát nếu có bất thường về dữ liệu trong thời gian nhanh nhất, để giúp người giám sát kịp thời xử lý và ngăn chặn bất kỳ sự xâm nhập nào.

Chương 2

Lí thuyết

2.1 Hệ thống phát hiện xâm nhập

2.1.1 Giới thiệu

IDS (Intrusion Detection System – hệ thống phát hiện xâm nhập) là một hệ thống giám sát lưu lượng mạng, có khả năng phát hiện các hoạt động khả nghi và cảnh báo cho hệ thống hoặc người quản trị mạng. IDS phát hiện dựa trên các dấu hiệu đặc biệt về các nguy cơ đã biết hoặc dựa trên việc so sánh lưu lượng mạng hiện tại với baseline. Trong đó baseline cho những hành vi bình thường được thiết lập bởi hồ sơ người dùng, máy chủ, hoặc hoạt động mạng trong quá trình học. Sau khi quá trình học kết thúc, những phát hiện bất thường được tìm ra sau quá trình thăm dò của IDS bị chệch đi so với baseline này. Phát hiện xâm nhập là một công việc khó khăn do sự phát triển nhanh chóng các của mạng lưới mạng, quá nhiều môi trường máy tính dẫn đến tính bất đồng bộ, nhiều giao thức mạng và sự phân loại đáng kể của các ứng dụng thông dụng và độc quyền. Hầu hết IDS sử dụng các dấu hiệu đặc biệt về nguy cơ xâm nhập đã biết(tương tự đối với cách mà các chương trình diệt virus hiện nay sử dụng để phát hiện và diệt virus), và sự khác biệt ứng xử của dấu hiệu xâm nhập so với những dấu hiệu đến từ người dùng thông thường.

IDS thường được đặt ở nhiều vị trí trong hệ thống mạng, trong đó, vị trí phía sau Firewall được nhiều nhà quản trị tin dùng nhất. ở vị trí này, IDS có nhiệm vụ phân tích các gói tin đã được Firewall thông qua, xác định các dấu hiệu đã được định nghĩa mà Firewall không thể kiểm tra hoặc ngăn chặn. Từ các dấu hiệu này, IDS cung cấp thông tin và đưa ra các cảnh báo cho người quản trị viên.

Với việc bảo vệ an toàn thông tin mạng ở một mức độ cao. Nhiều chuyên gia cho rằng IDS có giá trị giống như Firewall và VPN là ngăn ngừa các cuộc tấn công mà trong đó IDS cung cấp sự an toàn bằng cách trang bị cho bạn thông tin về các cuộc tấn công. Chính vì điều này, IDS có thể đáp ứng nhu cầu về an toàn hệ thống bằng cách cảnh báo về khả năng có thể xảy ra của các cuộc tấn công và đôi khi bên cạnh các cảnh báo đúng thì chúng cũng mắc phải một số nhầm lẫn dẫn đến các cảnh báo chưa chính xác.

2.1.2 Chức năng

Nhìn chung, bản thân IDS không có khả năng tự động ngăn chặn các cuộc tấn công, tuy nhiên, các phiên bản hiện đại của IDS như IPS (sẽ được nhắc đến ở những chương sau) đã có thể thực hiện nhiều vai trò hơn và có thể ngăn chặn các cuộc tấn công khi nó xảy ra. Thực tế, IDS dường như chỉ thông báo cho chúng ta biết rằng mạng đang có dấu hiệu bị tấn công và đang ở trong giai đoạn nguy hiểm.

Hệ thống phát hiện xâm nhập cho phép các tổ chức bảo vệ hệ thống mạng khỏi những đe dọa

với việc gia tăng kết nối mạng và sự tin cậy của hệ thống thông tin, bổ sung cho những điểm yếu của hệ thống khác... Sau đây là một vài lý do mà một hệ thống nên sử dụng IDS:

- Bảo vệ tính toàn vẹn dữ liệu, đảm bảo sự nhất quán của dữ liệu trong hệ thống. Các biện pháp đưa ra có khả năng ngăn chặn được sử thay đổi bất hợp pháp hoặc phá hoại dữ liệu.
- Bảo vệ tính riêng tư, nghĩa là đảm bảo cho người sử dụng khai thác tài nguyên của hệ thống theo đúng chức năng nhiệm vụ đã được phân quyền, ngăn chặn được sự truy cập thong tin bất hợp pháp.
- Bảo vệ tính bí mật, giữ cho thông tin không bị để lộ ra ngoài phạm vi cho phép.
- Bảo vệ tính khả dụng, nghĩa là hệ thống luôn sẵn sàng thực hiện yêu cầu truy cập thông tin của người dùng hợp pháp.
- Cung cấp thông tin về sự truy cập, đưa ra các chính sách đối phó, khôi phục, sửa chữa...

Về cơ bản, hệ thống IDS có thể giúp chúng ta ngăn ngừa các sự kiện tấn công trước khi nó xảy ra, cung cấp một số các giải pháp cho mạng và máy chủ, thậm chí cũng có thể hoạt động như một chuông báo động. Tuy nhiên chức năng chính của nó là thông báo cho người quản trị biết về các sự kiện có liên quan đến an ninh hệ thống đang sắp sửa xảy ra trong mạng và hệ thống mà người quản trị hiện đang kiểm soát.

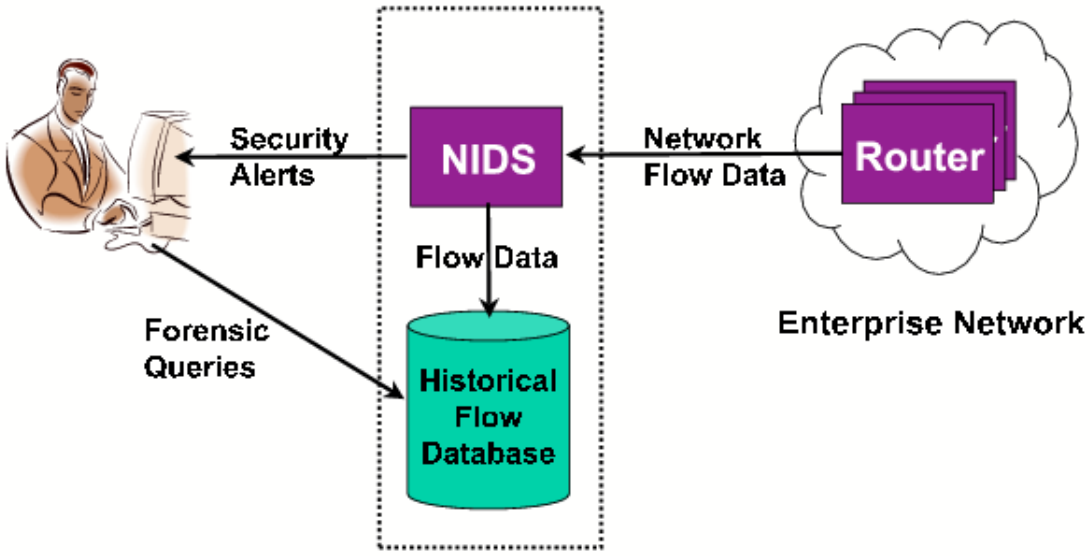
Thông thường, để phân loại IDS (IPS), người ta thường dựa vào đặc điểm của nguồn dữ liệu thu thập được. Trong trường hợp này, các hệ thống IDS được chia thành hai loại phổ biến như sau:

- Host-Based IDS (HIDS): Sử dụng dữ liệu kiểm tra từ một máy trạm đơn để phát hiện xâm nhập.
- Network-Based IDS (NIDS): Sử dụng dữ liệu trên toàn bộ lưu thông mạng, cùng với dữ liệu kiểm tra từ một hoặc một vài máy trạm để phát hiện xâm nhập.

2.1.3 NIDS

Hệ thống IDS dựa trên mạng sử dụng bộ dò và bộ cảm biến được cài đặt trên toàn mạng. Những bộ dò này theo dõi trên mạng với mục đích tìm kiếm những lưu lượng mạng khớp với những dấu hiệu được mô tả, định nghĩa từ trước. Những bộ cảm biến thu nhận và phân tích lưu lượng trong hệ thống thời gian thực. Khi nhận được mẫu lưu lượng hay dấu hiệu, bộ cảm biến gửi cảnh báo đến trạm quản trị và có thể được cấu hình nhằm tìm ra biện pháp ngăn chặn những xâm nhập xa hơn. NIDS là tập hợp nhiều cảm biến được cài đặt ở trên toàn mạng nhằm theo dõi những gói tin trong mạng, so sánh với mạng được định nghĩa để phát hiện đó là tấn công hay không.

NIDS được đặt giữa hệ thống mạng bên trong và hệ thống mạng bên ngoài để giám sát toàn bộ lưu lượng vào ra. Có thể là một thiết bị phần cứng riêng biệt được thiết lập sẵn hay phần mềm cài đặt trên máy tinh, chủ yếu dùng để đo lưu lượng mạng được sử dụng.



Hình 2.1: Luồng hoạt động của NIDS

NIDS giám sát toàn bộ mạng con của nó bằng cách lắng nghe tất cả các luồng dữ liệu trên mạng con đó. (Nó thay đổi chế độ hoạt động của card mạng NIC vào trong chế độ promisuous). Bình thường, một NIC hoạt động ở chế độ nonpromisuous nghĩa là nó chỉ nhận các gói tin mà có địa chỉ MAC khớp với địa chỉ của nó, các gói tin khác sẽ không nhận, hoặc không xử lý và bị loại bỏ. Để giám sát tất cả các đường truyền trong mạng con, NIDS phải chấp nhận tất cả các gói tin và chuyển chúng tới ngăn xếp để xử lý. Do vậy nó sẽ phải cài đặt chế độ hoạt động cho card mạng là promisuous.

Ưu điểm

- Chi phí triển khai thấp
- Phát hiện được các cuộc tấn công mà HIDS bỏ qua.
  Khác với HIDS, NIDS kiểm tra header của tất cả các gói tin vì thế hầu như không bỏ sót các dấu hiệu xuất phát từ đây.
- Khó xóa bỏ dấu vết (evidence)
- Phát hiện và đối phó kịp thời
- Có tính độc lập cao

Nhược điểm

- Bị hạn chế với Switch
- Hạn chế về hiệu năng
- Tăng băng thông mạng
- Một hệ thống NIDS thường gặp khó khăn trong việc xử lý các cuộc tấn công trong một phiên được mã hóa
- Một hệ thống NIDS cũng gặp khó khăn khi phát hiện các cuộc tấn công mạng từ các gói tin phân mảnh

2.1.4 HIDS

Host-based IDS tìm kiếm dấu hiệu của xâm nhập trong một máy chủ cục bộ; thường sử dụng các cơ chế kiểm tra và phân tích các thông tin được lưu lại. Chủ yếu tìm kiếm các hoạt động bất thường như đăng nhập, truy cập tập tin không được cấp phép, bước leo thang các đặc quyền không được chấp nhận. Kiến trúc IDS này thường dựa trên các luật (rule-based) để phân tích các hoạt động. Ví dụ, đặc quyền của người sử dụng ở quyền bậc cao chỉ có thể thành công thông qua lệnh su-select user, như vậy những hành vi đăng nhập liên tục vào tài khoản root có thể được coi là một cuộc tấn công.

Ưu điểm

- Xác định được kết quả của cuộc tấn công
- Khả năng giám sát các hoạt động cụ thể của hệ thống
- Phát hiện các hoạt động xâm nhập mà NIDS không phát hiện được: chẳng hạn kẻ tấn công sử dụng bàn phím xâm nhập vào một máy chủ sẽ không bị NIDS phát hiện.
- Không yêu cầu thêm phần cứng

Nhược điểm

- Khó quản trị
- Nguồn thông tin phân tích không an toàn
- Hệ thống host-based khá đắt
- Chiếm tài nguyên hệ thống

2.2 Khai phá dữ liệu

2.2.1 Khái niệm

Là quá trình tính toán để tìm ra các mẫu trong các bộ dữ liệu lớn liên quan đến các phương pháp tại giao điểm của máy học, thống kê và các hệ thống cơ sở dữ liệu. Đây là một lĩnh vực liên ngành của khoa học máy tính. Mục tiêu tổng thể của quá trình khai thác dữ liệu là trích xuất thông tin từ một bộ dữ liệu và chuyển nó thành một cấu trúc dễ hiểu để sử dụng tiếp. Ngoài bước phân tích thô, nó còn liên quan tới cơ sở dữ liệu và các khía cạnh quản lý dữ liệu, xử lý dữ liệu trước, suy xét mô hình và suy luận thống kê, các thước đo thú vị, các cân nhắc phức tạp, xuất kết quả về các cấu trúc được phát hiện, hiện hình hóa và cập nhật trực tuyến.

2.2.2 Các bước trong quá trình khai phá

Quá trình được thực hiện qua 9 bước:

1. **Tìm hiểu lĩnh vực của bài toán (ứng dụng)**: Các mục đích của bài toán, các tri thức cụ thể của lĩnh vực.
2. **Tạo nên (thu thập) một tập dữ liệu phù hợp.**
3. **Làm sạch và tiền xử lý dữ liệu.**

4. **Giảm kích thức của dữ liệu, chuyển đổi dữ liệu**: Xác định thuộc tính quan trọng, giảm số chiều (số thuộc tính), biểu diễn bất biến.

5. **Lựa chọn chức năng khai phá dữ liệu**: Phân loại, gom cụm, dự báo, sinh ra các luật kết hợp.

6. **Lựa chọn/ Phát triển (các) giải thuật khai phá dữ liệu phù hợp.**

7. **Tiến hành khai phá dữ liệu.**

8. **Đánh giá mẫu thu được và biểu diễn tri thức**: Hiển thị hóa, chuyển đổi, bỏ đi các mẫu dư thừa,...

9. **Sử dụng tri thức được khai phá.**

Quá trình khám phá tri thức theo cách nhìn của giới nghiên cứu về các hệ thống dữ liệu và kho dữ liệu về quá trình khám phá tri thức

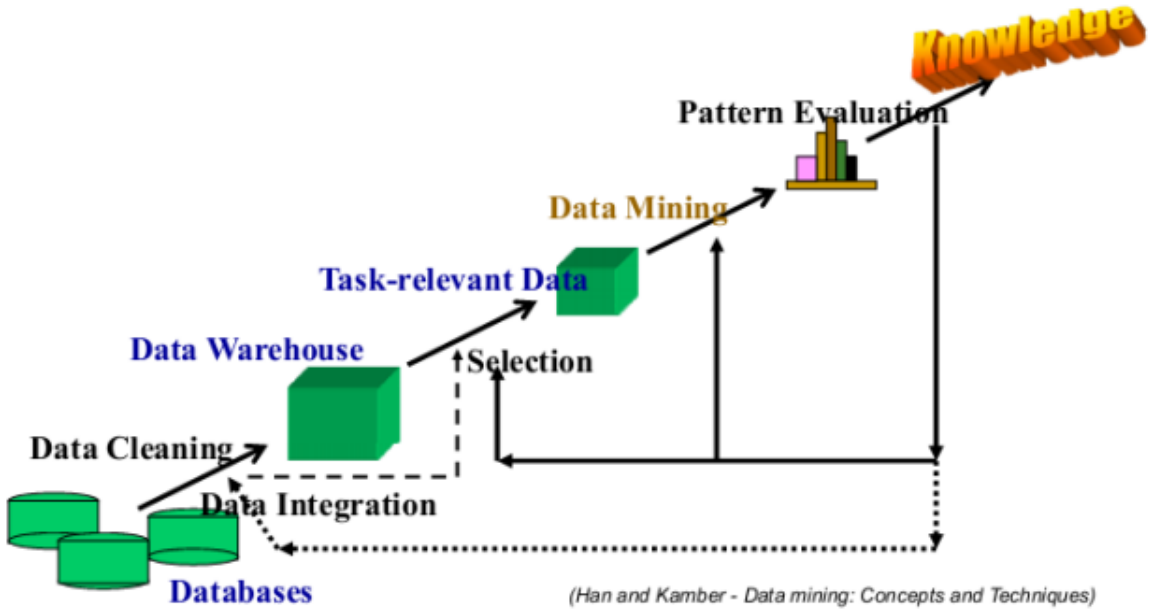
- Chuẩn bị dữ liệu *(data preparation)*, bao gồm các quá trình làm sạch dữ liệu *(data cleaning)*, tích hợp dữ liệu *(data integration)*, chọn dữ liệu *(data selection)*, biến đổi dữ liệu *(data transformation)*.

- Khai thác dữ liệu *(data mining)*: xác định nhiệm vụ khai thác dữ liệu và lựa chọn kỹ thuật khai thác dữ liệu. Kết quả cho ta một nguồn tri thức thô.

- Đánh giá *(evaluation)*: dựa trên một số tiêu chí tiến hành kiểm tra và lọc nguồn tri thức thu được.

- Triển khai *(deployment)*.

- Quá trình khai thác tri thức không chỉ là một quá trình tuần tự từ bước đầu tiên đến bước cuối cùng mà là một quá trình lặp và có quay trở lại các bước đã qua.



Hình 2.2: Quá trình khai phá tri thức

2.2.3 Ứng dụng của khai phá dữ liệu

- Kinh tế - ứng dụng trong kinh doanh, tài chính, tiếp thị bán hàng, bảo hiểm, thương mại, ngân hàng, ... Đưa ra các bản báo cáo giàu thông tin; phân tích rủi ro trước khi đưa ra các chiến lược kinh doanh, sản xuất; phân loại khách hàng từ đó phân định thị trường, thị phần; ...
- Khoa học: Thiên văn học – dự đoán đường đi các thiên thể, hành tinh, ...; Công nghệ sinh học – tìm ra các gen mới, cây con giống mới, ...; ...
- Web: các công cụ tìm kiếm.

2.3 Máy học

2.3.1 Giới thiệu

Những năm gần đây, AI - Artificial Intelligence (Trí Tuệ Nhân Tạo), và cụ thể hơn là Machine Learning (Học Máy hoặc Máy Học) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư (1 - động cơ hơi nước, 2 - năng lượng điện, 3 - công nghệ thông tin). Trí Tuệ Nhân Tạo đang len lỏi vào mọi lĩnh vực trong đời sống mà có thể chúng ta không nhận ra. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, ..., chỉ là một vài trong vô vàn những ứng dụng của AI/Machine Learning.

Machine Learning là một tập con của AI. Theo định nghĩa của Wikipedia, Machine learning is the subfield of computer science that "gives computers the ability to learn without being explicitly programmed". Nói đơn giản, Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.

Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, Machine Learning đã tiến

thêm một bước dài và một lĩnh vực mới được ra đời gọi là Deep Learning (Học Sâu - thực sự tôi không muốn dịch từ này ra tiếng Việt). Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc

2.3.2 Phân nhóm các thuật toán

Theo phương thức học, các thuật toán Machine Learning thường được chia làm 4 nhóm:

- Supervise learning (học có giám sát)
- Unsupervised learning (học không có giám sát)
- Semi-supervised lerning (Học bán giám sát)
- Reinforcement learning (Học củng cố)

1. **Học có giám sát (Supervised Learning)**

   Supervised learning là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là (data, label), tức (dữ liệu, nhãn). Supervised learning là nhóm phổ biến nhất trong các thuật toán Machine Learning

   (a) **Classification (Phân loại)**: Một bài toán được gọi là classification nếu các label của input data được chia thành một số hữu hạn nhóm. Ví dụ: Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không. Ba ví dụ phía trên được chia vào loại này

   (b) **Regression (Hồi quy)** Nếu label không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một căn nhà rộng x m2x m2, có yy phòng ngủ và cách trung tâm thành phố z kmz km sẽ có giá là bao nhiêu?

   Gần đây Microsoft có một ứng dụng dự đoán giới tính và tuổi dựa trên khuôn mặt. Phần dự đoán giới tính có thể coi là thuật toán Classification, phần dự đoán tuổi có thể coi là thuật toán Regression. Chú ý rằng phần dự đoán tuổi cũng có thể coi là Classification nếu ta coi tuổi là một số nguyên dương không lớn hơn 150, chúng ta sẽ có 150 class (lớp) khác nhau

2. **Học không giám sát (Unsupervise learning)**

   Trong thuật toán này, chúng ta không biết được outcome hay nhãn mà chỉ có dữ liệu đầu vào. Thuật toán unsupervised learning sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

   Một cách toán học, Unsupervised learning là khi chúng ta chỉ có dữ liệu vào XX mà không biết nhãn YYtương ứng. Những thuật toán loại này được gọi là Unsupervised learning vì không giống như Supervised learning, chúng ta không biết câu trả lời chính xác cho mỗi dữ liệu đầu vào. Giống như khi ta học, không có thầy cô giáo nào chỉ cho ta biết đó là chữ A hay chữ B. Cụm không giám sát được đặt tên theo nghĩa này.

   Các bài toán Unsupervised learning được tiếp tục chia nhỏ thành hai loại:

(a) **Clustering (phân nhóm)**

Một bài toán phân nhóm toàn bộ dữ liệu XX thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm. Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. Điều này cũng giống như việc ta đưa cho một đứa trẻ rất nhiều mảnh ghép với các hình thù và màu sắc khác nhau, ví dụ tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.

(b) **Association (kết hợp)**

Là bài toán khi chúng ta muốn khám phá ra một quy luật dựa trên nhiều dữ liệu cho trước. Ví dụ: những khách hàng nam mua quần áo thường có xu hướng mua thêm đồng hồ hoặc thắt lưng; những khán giả xem phim Spider Man thường có xu hướng xem thêm phim Bat Man, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm.

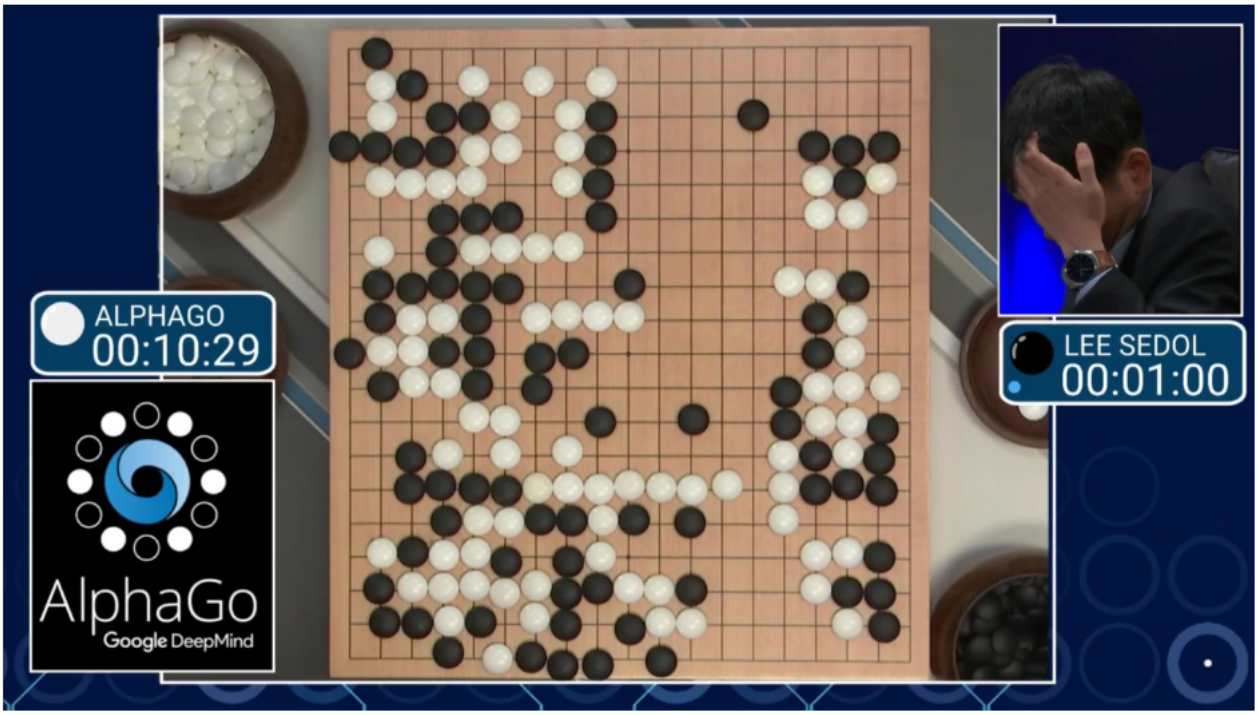
3. **Học bán giám sát (Semi-Supervise Learning)**

Các bài toán khi chúng ta có một lượng lớn dữ liệu XX nhưng chỉ một phần trong chúng được gán nhãn được gọi là Semi-Supervised Learning. Những bài toán thuộc nhóm này nằm giữa hai nhóm được nêu bên trên.

Một ví dụ điển hình của nhóm này là chỉ có một phần ảnh hoặc văn bản được gán nhãn (ví dụ bức ảnh về người, động vật hoặc các văn bản khoa học, chính trị) và phần lớn các bức ảnh/văn bản khác chưa được gán nhãn được thu thập từ internet. Thực tế cho thấy rất nhiều các bài toán Machine Learning thuộc vào nhóm này vì việc thu thập dữ liệu có nhãn tốn rất nhiều thời gian và có chi phí cao. Rất nhiều loại dữ liệu thậm chí cần phải có chuyên gia mới gán nhãn được (ảnh y học chẳng hạn). Ngược lại, dữ liệu chưa có nhãn có thể được thu thập với chi phí thấp từ internet.

4. **Học củng cố (Reinforcement Learning)**

Reinforcement learning là các bài toán giúp cho một hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất (maximizing the performance). Hiện tại, Reinforcement learning chủ yếu được áp dụng vào Lý Thuyết Trò Chơi (Game Theory), các thuật toán cần xác định nước đi tiếp theo để đạt được điểm số cao nhất



Hình 2.3: AlphaGo chơi cờ vây với Lee Sedol. AlphaGo là một ví dụ điển hình của Reinforcement Learing

Chương 3

Công nghệ tiếp cận

3.1 Snort 3

3.1.1 Giới thiệu

Snort là phần mềm IDS mã nguồn mở, được phát triển bởi Martin Roesh. Snort đầu tiên được xây dựng trên nền Unix sau đó phát triển sang các nền tảng khác. Snort được đánh giá là IDS mã nguồn mở đáng chú ý nhất với những tính năng rất mạnh.

Snort là một NIDS được Martin Roesh phát triển dưới mô hình mã nguồn mở. Tuy Snort miễn phí nhưng nó lại có rất nhiều tính năng tuyệt vời mà không phải sản phẩm thương mại nào cũng có thể có được. Với kiến trúc thiết kế theo kiểu module, người dùng có thể tự tăng cường tính năng cho hệ thống Snort của mình bằng việc cài đặt hay viết thêm mới các module. Cơ sở dữ liệu luật của Snort đã lên tới 2930 luật và được cập nhật thường xuyên bởi một cộng đồng người sử dụng. Snort có thể chạy trên nhiều hệ thống nền như Windows, Linux, OpenBSD, FreeBSD, NetBSD, Solaris, HP-UX, AIX, IRIX, MacOS. Bên cạnh việc có thể hoạt động như một ứng dụng thu bắt gói tin thông thường, Snort còn có thể được cấu hình để chạy như một NIDS. Snort hỗ trợ khả năng hoạt động trên các giao thức sau: Ethernet, 802.11,Token Ring,FDDI,Cisco HDLC, SLIP, PPP, và PF của OpenBSD.

3.1.2 Tính năng

Snort 3.0 là một phiên bản cập nhật của Hệ thống Ngăn chặn xâm nhập Snort (IPS) có tính năng thiết kế mới được cung cấp từ Snort 2.X với thông lượng lớn hơn, phát hiện tốt hơn, khả năng mở rộng và khả năng sử dụng tốt hơn. Một số tính năng chính của Snort 3.0 là:

- Hỗ trợ nhiều luồng xử lý gói
- Sử dụng cấu hình được chia sẻ và bảng thuộc tính
- Dịch vụ tự động phát hiện cho cấu hình không cần cổng
- Thiết kế mô-đun
- Nền tảng plugin với hơn 200 plugin
- Cấu hình bộ nhớ mở rộng hơn
- Cấu hình LuaJIT, logger, và các tùy chọn quy tắc

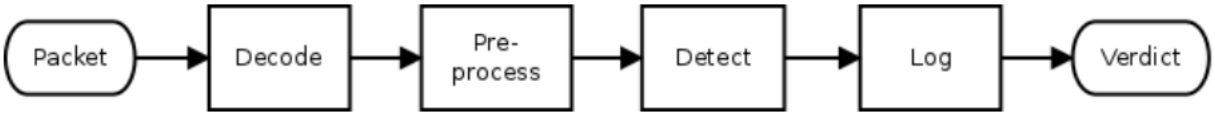
- Hỗ trợ Hyperscan
- Xử lý TCP được ghi đè
- Trình phân tích cú pháp và cú pháp quy tắc mới
- Các quy tắc dịch vụ như http cảnh báo
- Quy tắc Bộ đệm "dính"
- Quy tắc SO tốt hơn
- Kiểm soát HTTP mới
- Hiệu suất giám sát mới
- Hồ sơ thời gian và không gian mới
- Độ trễ giám sát và thực thi mới
- Piglets để tạo điều kiện thử nghiệm thành phần
- Kiểm tra các sự kiện
- Automake và Cmake
- Tự động tạo tài liệu tham khảo

Các tính năng đang được phát triển

- Sử dụng mạng lưới network đã chia sẻ
- Hỗ trợ phần cứng giảm tải để tăng tốc mẫu nhanh
- Cung cấp hỗ trợ cho DPDK và ODP
- Hỗ trợ pipelining xử lý gói tin
- Hỗ trợ chế độ proxy
- Hỗ trợ đa tennant
- Gia tăng tải lại
- Tuần tự hóa hiệu suất của dữ liệu và sự kiện mới
- Xử lý quy tắc nâng cao
- Hỗ trợ Windows
- Phát hiện bất thường
- và hơn thế nữa!

3.1.3 Kiến trúc

Snort 2.x



Hình 3.1: Kiến trúc Snort 2.x

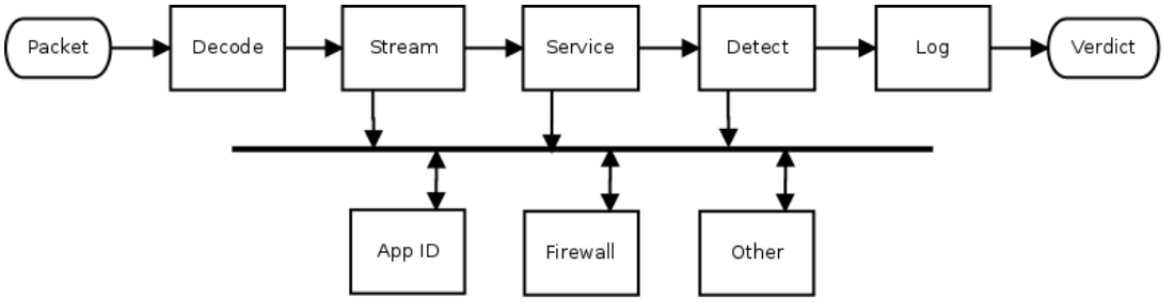
Bước tiền xử lý trong Snort 2 có khả năng cấu hình cao. Các bộ tiền xử lý tùy ý có thể được nạp tự động khi khởi động, được cấu hình trong snort.conf, và sau đó được thực thi khi chạy. Về cơ bản, các bộ tiền xử lý được đưa vào danh sách được lặp lại cho mỗi gói. Các phiên bản gần đây đã tinh chỉnh một số danh sách xử lý, nhưng kiến trúc cơ bản giống nhau đã cho phép Snort 2 phát triển từ một trình thám thính, không có tiền xử lý,cho đến một IPS chính thức, với rất nhiều tiền xử lý.

Trong khi phương pháp "list of plugins"("danh sách plugin") này có tính linh hoạt đáng kể, nó cản trở sự phát triển trong tương lai khi luồng dữ liệu từ một bộ tiền xử lý tới bộ xử lý tiếp theo phụ thuộc vào điều kiện lưu lượng dữ liệu, tình huống chung với các tính năng nâng cao như nhận dạng ứng dụng. Trong trường hợp này, một bộ tiền xử lý như HTTP có thể trích xuất và chuẩn hóa dữ liệu mà cuối cùng không được sử dụng, hoặc appID có thể liên tục kiểm tra dữ liệu không có sẵn.

Callbacks giúp thoát ra khỏi bộ tiền xử lý đang quá tải. Đây là nơi mà một bộ tiền xử lý cung cấp một bộ xử lý khác có chức năng gọi khi có sẵn một số dữ liệu nhất định. Snort đã bắt đầu thực hiện phương pháp này để chuyển một số dữ liệu tiền xử lý HTTP và SIP tới appID. Tuy nhiên, nó vẫn là một tính năng ngoại vi và vẫn yêu cầu dữ liệu mà không được xử lý

Khi Snort hoạt động nó sẽ thực hiện việc lắng nghe và thu bắt tất cả các gói tin nào di chuyển qua nó. Các gói tin sau khi bị bắt được đưa vào Môđun giải mã gói tin. Tiếp theo gói tin sẽ được đưa vào môđun Tiền xử lý, rồi môđun Phát hiện. Tại đây tùy theo việc có phát hiện được xâm nhập hay không mà gói tin có thể được bỏ qua để lưu thông tiếp hoặc được đưa vào môđun Log và cảnh báo để xử lý. Khi các cảnh báo được xác định môđun Kết xuất thông tin sẽ thực hiện việc đưa cảnh báo ra theo đúng định dạng mong muốn.

Snort 3.x



Hình 3.2: Kiến trúc Snort 3.x

Một trong những mục tiêu của Snort 3 là cung cấp một khuôn khổ linh hoạt hơn để xử lý gói tin bằng cách thực hiện một phương pháp hướng sự kiện. Khác là để sản xuất dữ liệu chỉ khi cần thiết để giảm thiểu chuẩn hóa đắt tiền. Tuy nhiên, xử lý gói cơ bản cung cấp chức năng rất giống nhau.

Các bước xử lý cơ bản Snort 3 tương tự như Snort 2 như đã thấy trong sơ đồ. Bước tiền xử lý sử dụng các loại kiểm duyệt cụ thể thay vì một danh sách tổng quát, nhưng thủ tục cơ bản bao gồm giải mã gói không trạng thái, khôi phục luồng TCP và phân tích dịch vụ cụ thể trong cả hai trường hợp. (Snort 3 cung cấp các móc cho các kiểm duyệt tùy ý, nhưng chúng không tập trung vào xử lý luồng cơ bản và không được hiển thị.)

Tuy nhiên, Snort 3 cũng cung cấp một cơ chế linh hoạt hơn các hàm gọi lại. Bằng cách sử dụng các sự kiện kiểm tra, một kiểm duyệt viên có thể cung cấp dữ liệu mà các kiểm duyệt viên khác có thể xử lý. Điều này được gọi là mẫu quan sát hoặc mẫu đăng ký xuất bản.

Lưu ý rằng dữ liệu không thực sự được xuất bản. Thay vào đó, quyền truy cập vào dữ liệu được xuất bản và điều đó có nghĩa là người đăng ký có thể truy cập (các) phiên bản thô hoặc chuẩn hóa nếu cần. Việc chuẩn hóa chỉ được thực hiện trên lần truy cập đầu tiên và các truy cập tiếp theo nhận được dữ liệu đã chuẩn hóa trước đó. Điều này dẫn đến việc xử lý kịp thời (JIT).

3.1.4 Cấu hình

Cấu hình Snort hiệu quả được thực hiện thông qua môi trường, dòng lệnh, tệp cấu hình Lua và một bộ quy tắc.

Lưu ý rằng khả năng tương thích ngược với Snort 2 đã được hy sinh để có được chức năng mới và được cải thiện. Trong khi Snort 3 sử dụng một số cơ sở mã Snort 2, rất nhiều đã thay đổi. Cấu hình của Snort 3 được thực hiện bằng Lua, vì vậy ký tự cũ của bạn sẽ không hoạt động như cũ. Quy tắc vẫn dựa trên văn bản nhưng với các chỉnh sửa cú pháp, vì vậy các quy tắc 2.X của bạn phải được sửa. Tuy nhiên, snort2lua sẽ giúp bạn chuyển đổi conf và quy tắc của bạn sang định dạng mới.

Môi trường

LUA_PATH phải được đặt dựa trên cài đặt của bạn:

```
LUA_PATH=$install_prefix/include/snort/lua/\?.lua\;\;
```

SNORT_LUA_PATH phải được thiết lập để tải các tệp cấu hình phụ nếu bạn sử dụng snort.lua mặc định. Ví dụ:

```
export SNORT\_LUA\_PATH=$install\_prefix/etc/snort
```

Dòng lệnh

Một dòng lệnh đơn giản có thể trông giống như sau:

```
snort -c snort.lua -R cool.rules -r some.pcap -A cmg
```

Để hiểu điều đó, bạn có thể bắt đầu bằng cách chạy snort mà không có đối số bằng cách chạy snort –help. Trợ giúp cho tất cả các tùy chọn cấu hình và quy tắc có sẵn thông qua một dòng lệnh phù hợp. Trong trường hợp này:

-c snort.lua là tệp cấu hình chính. Đây là một kịch bản lệnh Lua được thực hiện khi được nạp.

-R cool.rules chứa một số quy tắc phát hiện. Bạn có thể viết của riêng bạn hoặc có được chúng từ Talos (nguyên tắc 3.0 nguyên bản chưa có sẵn từ Talos, do đó bạn phải chuyển đổi chúng với snort2lua). Bạn cũng có thể đặt các quy tắc của mình trực tiếp trong tệp cấu hình của mình.

-r some.pcap yêu cầu Snort đọc lưu lượng mạng từ tệp chụp gói tin đã cho. Thay vào đó, bạn có thể sử dụng -i eth0 để đọc từ giao diện trực tiếp. Có nhiều tùy chọn khác có sẵn quá tùy thuộc vào DAQ bạn sử dụng.

-A cmg cho biết các sự kiện xâm nhập đầu ra ở định dạng "cmg", có các chi tiết tiêu đề cơ bản theo sau bởi tải trọng trong hex và văn bản.

Tập tin cấu hình

File cấu hình cho phép bạn hoàn toàn kiểm soát cách Snort xử lý các gói tin. Bắt đầu với snort.lua mặc định được bao gồm trong bản phân phối vì có chứa một số thành phần chính. Lưu ý rằng hầu hết các cấu hình trông giống như:

```
stream = { }
```

Điều này có nghĩa là cho phép mô-đun luồng sử dụng các giá trị mặc định nội bộ. Để xem những thứ đó là gì, bạn có thể chạy:

```
$ snort --help-config stream
```

Snort được tổ chức thành một bộ sưu tập các mô-đun dựng sẵn và plugin. Nếu một module có các tham số, nó được cấu hình bởi một bảng Lua có cùng tên. Ví dụ, chúng ta có thể thấy những gì module đang hoạt động cung cấp với lệnh này:

```
$ snort --help-module active
```

Quy tắc

Quy tắc xác định những gì Snort đang tìm kiếm. Chúng có thể được đặt trực tiếp trong tệp cấu hình Lua của bạn với mô đun ips, trên dòng lệnh với –lua hoặc trong các tệp bên ngoài. Nói chung, bạn sẽ có nhiều quy tắc thu được từ nhiều nguồn khác nhau như Talos và tải các tập tin bên ngoài là cách để đi vì vậy chúng tôi sẽ tóm tắt ở đây. Thêm vào cấu hình Lua của bạn:

```
ips = { include = 'rules.txt' }
```

để tải tệp quy tắc bên ngoài có tên rules.txt. Bạn chỉ có thể chỉ định một tệp theo cách này nhưng quy tắc tệp có thể bao gồm các tệp quy tắc khác với câu lệnh include. Ngoài ra, bạn có thể tải các quy tắc như:

```
$ sort -c snort.lua -R rules.txt
```

3.1.5 Mở rộng tính năng

Plugin

Các plugin có một API được liên kết được xác định cho từng loại, tất cả đều có chung một tiêu đề, được gọi là BaseApi. Một thư viện động làm cho các plugin của nó có sẵn bằng cách xuất khẩu biểu tượng đó là snort_plugins, một mảng null kết thúc của các con trỏ BaseApi.

BaseApi bao gồm loại, tên, phiên bản API, phiên bản plugin và các con trỏ hàm để xây dựng và hủy một Mô-đun. API cụ thể thêm nhiều dữ liệu và chức năng khác cho vai trò đã cho của chúng.

Mô đun

Nếu chúng ta định nghĩa một Inspector, gadget nó có thể được cấu hình trong snort.lua như sau:

```
gadget =
{
    brain = true,
    claw = 3
}
```

Khi bảng tiện ích được xử lý, Snort sẽ tìm kiếm một mô-đun được gọi là tiện ích. Nếu Module đó có một API liên quan, nó sẽ được sử dụng để cấu hình một thể hiện mới của plugin. Trong trường hợp này, một GadgetModule sẽ được khởi tạo, não và móng sẽ được thiết lập, và cá thể Module sẽ được truyền cho hàm tạo GadgetInspector.

Module có ba phương thức ảo chính:

- **begin()** - được gọi khi Snort bắt đầu xử lý bảng Lua được liên kết. Đây là một nơi để phân bổ bất kỳ dữ liệu cần thiết và thiết lập mặc định.
- **set()** - được gọi để đặt từng thông số sau khi xác thực.

- **end()** - được gọi khi Snort kết thúc xử lý bảng Lua được liên kết. Đây là nơi kiểm tra tính toàn vẹn bổ sung của các tham số liên quan nên được thực hiện.

Mô-đun được cấu hình sẽ chuyển đến trình tạo plugin để lấy dữ liệu cấu hình từ Mô-đun. Đối với các cấu hình không quan trọng, mô hình làm việc là Mô-đun đưa một con trỏ tới dữ liệu đã được cấu hình cho cá thể của plugin có quyền sở hữu.

Lưu ý rằng có tối đa một phiên bản của một Mô-đun cụ thể, ngay cả khi nhiều phiên bản plugin được tạo sử dụng Mô-đun đó. (Nhiều trường hợp yêu cầu cấu hình ràng buộc Snort.)

Inspector

Có một số loại kiểm duyệt, xác định kiểm duyệt nào được thực hiện khi:

- **IT_BINDER** - xác định kiểm duyệt nào áp dụng cho các luồng đã cho
- **IT_WIZARD** - xác định kiểm duyệt dịch vụ nào sẽ sử dụng nếu không ràng buộc rõ rang
- **IT_PACKET** - được sử dụng để xử lý tất cả các gói trước khi xử lý phiên và dịch vụ (ví dụ: chuẩn hóa)
- **IT_NETWORK** - xử lý các gói dịch vụ w / o (ví dụ: arp_spoof, back_orifice)
- **IT_STREAM** - để theo dõi lưu lượng, chống phân mảnh ip và khôi phục tcp
- **IT_SERVICE** - cho http, ftp, telnet, v.v.
- **IT_PROBE** - xử lý tất cả các gói sau tất cả các gói ở trên (ví dụ: perf_monitor, port_scan)

Codec

Snort Codec giải mã các gói dữ liệu thô. Các Codec này bây giờ hoàn toàn có thể cắm được; hầu như mọi Snort Codec đều có thể được xây dựng động và được thay thế bằng Codec tùy biến, thay thế. Bản chất có thể cắm được cũng giúp việc xây dựng các Codec mới cho các giao thức dễ dàng hơn mà không cần phải chạm vào mã Snort.

Bước đầu tiên trong việc tạo một Codec là định nghĩa lớp và giao thức của nó. Mỗi Codec phải kế thừa từ lớp Snort Codec được định nghĩa trong "framework / codec.h".

IPS action

Các plugin thực thi một hành động dựng sẵn trong API được sử dụng để xác định phán quyết. (Ngược lại, hành động được tạo sẵn không có sự liên kết với chức năng plugin.)

Piglet Test Harness

Để hỗ trợ phát triển plugin, chế độ thử nghiệm được gọi là chế độ "piglet"được cung cấp. Với chế độ Piglet, bạn có thể gọi các phương thức riêng cho một plugin cụ thể. Các xét nghiệm heo con được xác định là kịch bản Lua. Mỗi kịch bản thử nghiệm heo con xác định một thử nghiệm cho một plugin cụ thể.

3.2 Scikit

3.2.1 Giới thiệu

Scikit-learn (viết tắt là **sklearn**) là một thư viện mã nguồn mở trong ngành machine learning, rất mạnh mẽ và thông dụng với cộng đồng Python, được thiết kế trên nền NumPy và SciPy. Scikit-learn

chứa hầu hết các thuật toán machine learning hiện đại nhất, đi kèm với comprehensive documentations. Điểm mạnh của thư viện này là nó được sử dụng phổ biến trong giáo dục và công nghiệp, do đó nó luôn được cập nhật và có một cộng đồng người dùng đông đảo.

Hình 3.3: Scikit

3.2.2 Tại sao nên dùng Scikit?

Hiện nay có nhiều thư viện mã nguồn mở phục vụ cho nghiên cứu machine learning. Bên cạnh Scikit-learn, có 2 thư viện nổi bật khác là

- **LibSVM**: Được viết trên C bởi Chih-Chung Chang và Chih-Jen Lin. Như tên gọi của nó, thư viên này chứa các thuật toán SVM (Support Vector Machine), nhóm thuật toán mạnh mẽ hỗ trợ cả các tác vụ regression và classification.
- **TensorFlows**: Do các nhà khoa học của viện nghiên cứu Google Brain phát triển. TensorFlows được viết trên Python và là thư viện mở.

Trong khi TensorFlows có vẻ ở mức độ tiếp cận thấp hơn hơn thì Scikit-learn cho phép ta sử dụng ngay các thuật toán quan trọng một cách đơn giản và hiệu quả. Nói vậy không có nghĩa Scikit-learn là một thư viện "nông cạn", Scikit-learn là nền tảng để xây dựng các ML implementations khác (Nilearn, Pylearn2,...). Scikit-learn còn là một trong những lựa chọn hàng đầu của các nhà nghiên cứu và phát triển. Đứng sau Scikit-learn là các viện nghiên cứu hàng đầu thế giới, gồm có Inria, Télécom Paristech, Paris-Saclay (Pháp), NYU Moore-Sloan Data Science Environment và Columbia University.

3.3 Thuật toán KMeans

3.3.1 Giới thiệu

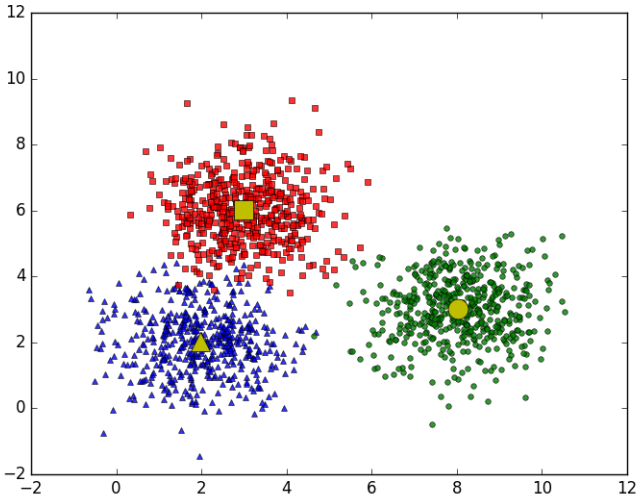
Thuật toán **K-means clustering** do MacQueen giới thiệu trong tài liệu "J. Some Methods for Classification and Analysis of Multivariate Observations" năm 1967. K-means Clustering là một thuật toán dùng trong các bài toán phân loại/nhóm n đối tượng thành k nhóm dựa trên đặc tính/thuộc tính của đối tượng (k £n nguyên, dương). Về nguyên lý, có n đối tượng, mỗi đối tượng có m thuộc tính, ta phân chia được các đối tượng thành k nhóm dựa trên các thuộc tính của đối tượng bằng việc áp dụng thuật toán này. Coi mỗi thuộc tính của đối tượng (đối tượng có m thuộc tính) như một toạ độ của không gian m chiều và biểu diễn đối tượng như một điểm của không gian m chiều.

Trong thuật toán K-means clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích là làm thể nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.

Ví dụ: Một công ty muốn tạo ra những chính sách ưu đãi cho những nhóm khách hàng khác nhau dựa trên sự tương tác giữa mỗi khách hàng với công ty đó (số năm là khách hàng; số tiền khách

hàng đã chi trả cho công ty; độ tuổi; giới tính; thành phố; nghề nghiệp; . . .). Giả sử công ty đó có rất nhiều dữ liệu của rất nhiều khách hàng nhưng chưa có cách nào chia toàn bộ khách hàng đó thành một số nhóm/cụm khác nhau. Nếu một người biết Machine Learning được đặt câu hỏi này, phương pháp đầu tiên ta nghĩ đến sẽ là K-means Clustering. Sau khi đã phân ra được từng nhóm, nhân viên công ty đó có thể lựa chọn ra một vài khách hàng trong mỗi nhóm để quyết định xem mỗi nhóm tương ứng với nhóm khách hàng nào. Phần việc cuối cùng này cần sự can thiệp của con người, nhưng lượng công việc đã được rút gọn đi rất nhiều.

Ý tưởng đơn giản nhất về cluster (cụm) là tập hợp các điểm ở gần nhau trong một không gian nào đó (không gian này có thể có rất nhiều chiều trong trường hợp thông tin về một điểm dữ liệu là rất lớn). Hình bên dưới là một ví dụ về 3 cụm dữ liệu.

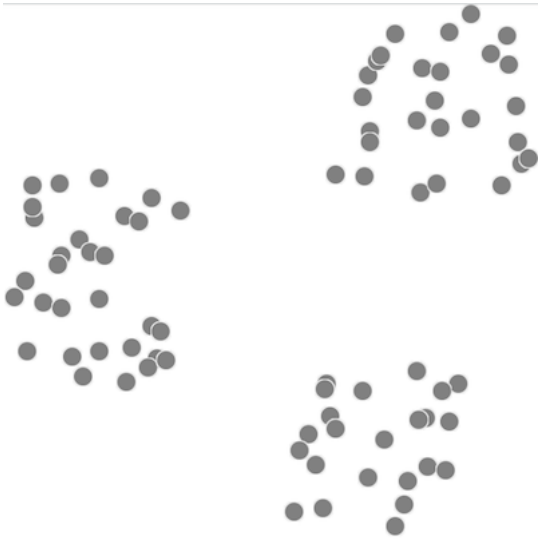


Hình 3.4: Bài toán với 3 clusters.

Giả sử mỗi cluster có một điểm đại diện (center) màu vàng. Và những điểm xung quanh mỗi center thuộc vào cùng nhóm với center đó. Một cách đơn giản nhất, xét một điểm bất kỳ, ta xét xem điểm đó gần với center nào nhất thì nó thuộc về cùng nhóm với center đó. Tới đây, chúng ta có một bài toán thú vị: *Trên một vùng biển hình vuông lớn có ba đảo hình vuông, tam giác, và tròn màu vàng như hình trên. Một điểm trên biển được gọi là thuộc lãnh hải của một đảo nếu nó nằm gần đảo này hơn so với hai đảo kia . Hãy xác định ranh giới lãnh hải của các đảo*

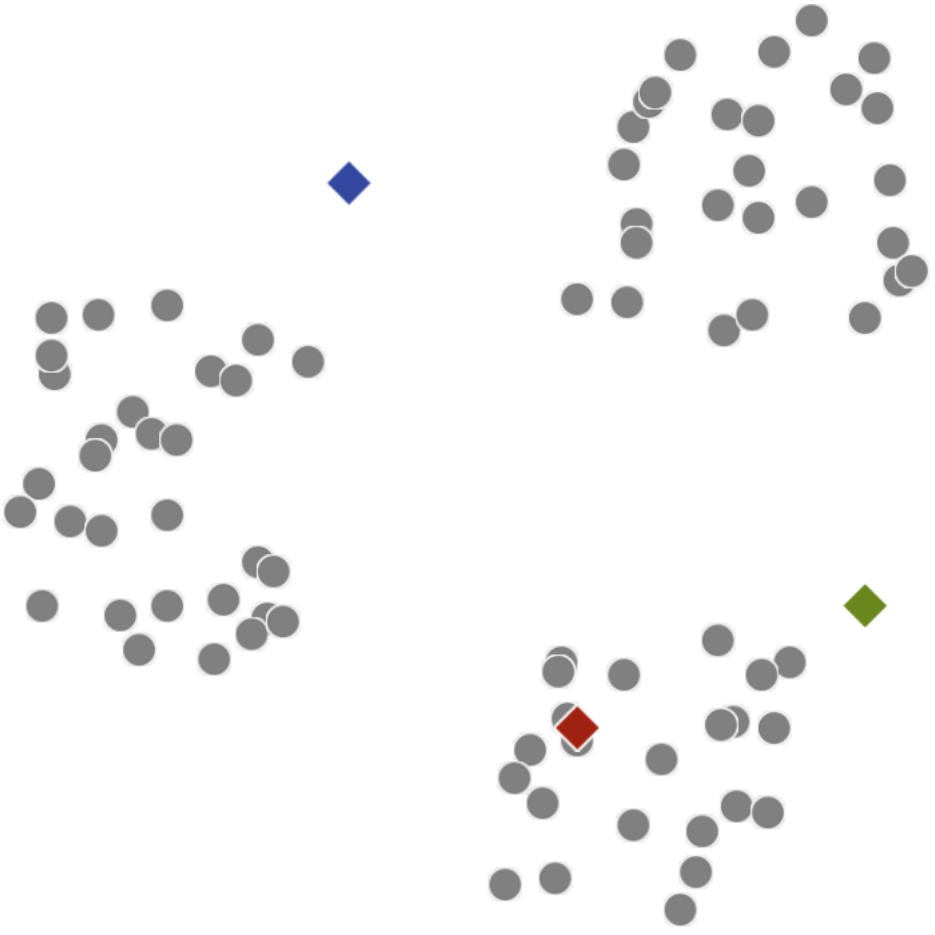
3.3.2 Phân tích chi tiết

Đầu tiên là chuẩn bị dữ liệu cần phân cụm. Tiếp theo quyết định số lượng cụm (cluster) cần phân chia. Ở ví dụ này thử chọn số cluster là 3. Ở đây data được thể hiện dưới dạng các điểm cho dễ quan sát. Cự ly của các dữ liệu được hiểu là độ dài đoạn thẳng nối giữa 2 điểm với nhau.



Hình 3.5: Cụm ban đầu.

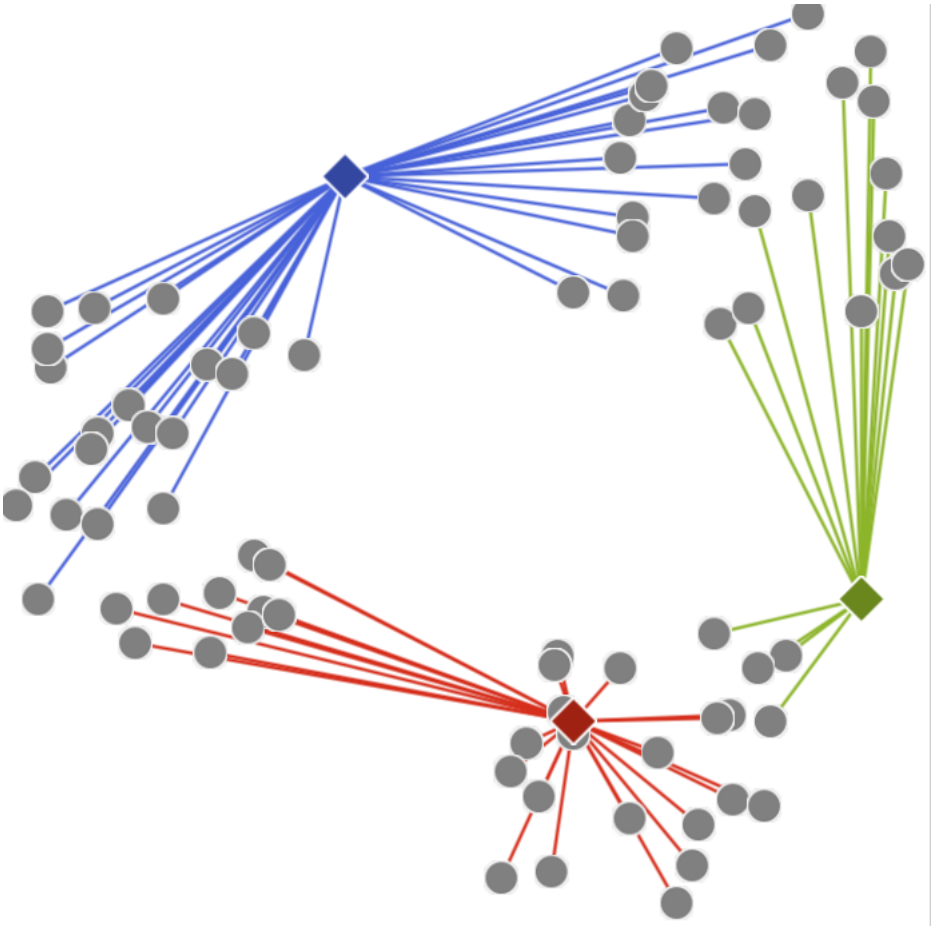
Bước 2
Chọn ngẫu nhiên 3 điểm làm điểm trung tâm của cluster.



Hình 3.6: Chọn ngẫu nhiên trung điểm.

Bước 3

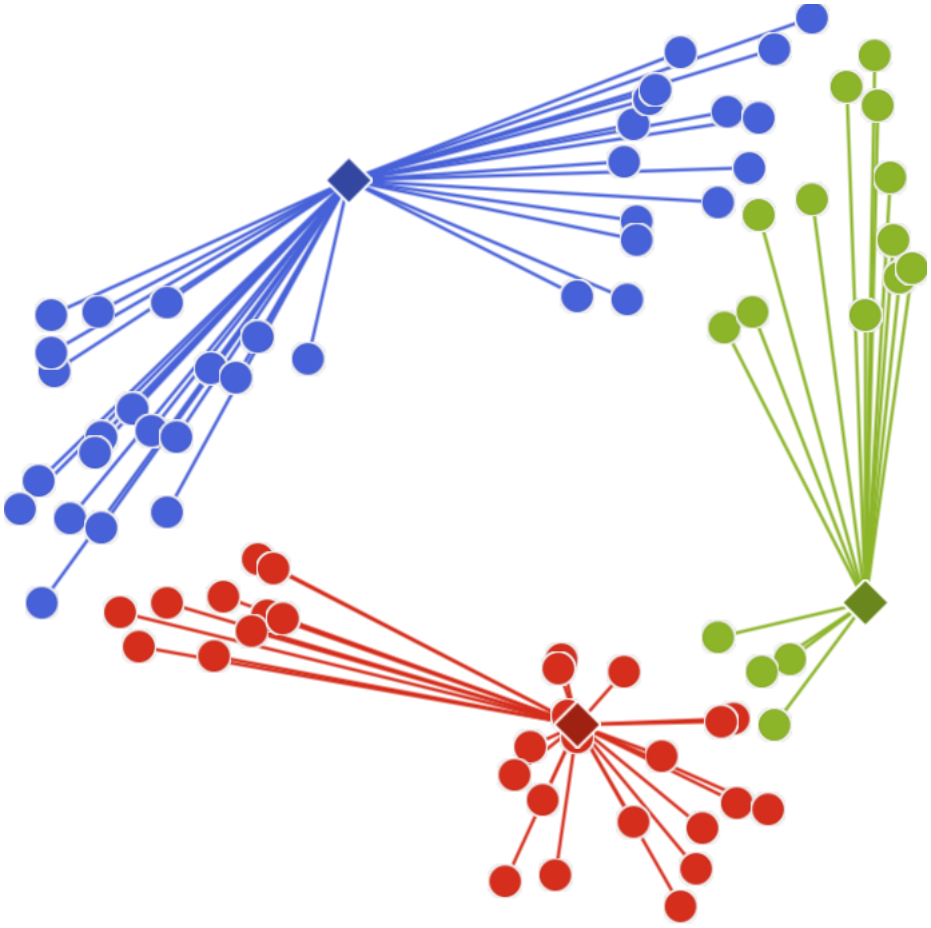
Với các điểm dữ liệu không được chọn là điểm trung tâm thì tính toán khoảng cách từ chính điểm đó đến các cluster và quyết định cluster nào gần với mình nhất.



Hình 3.7: Tính toán khoảng cách tới điểm.

Bước 4

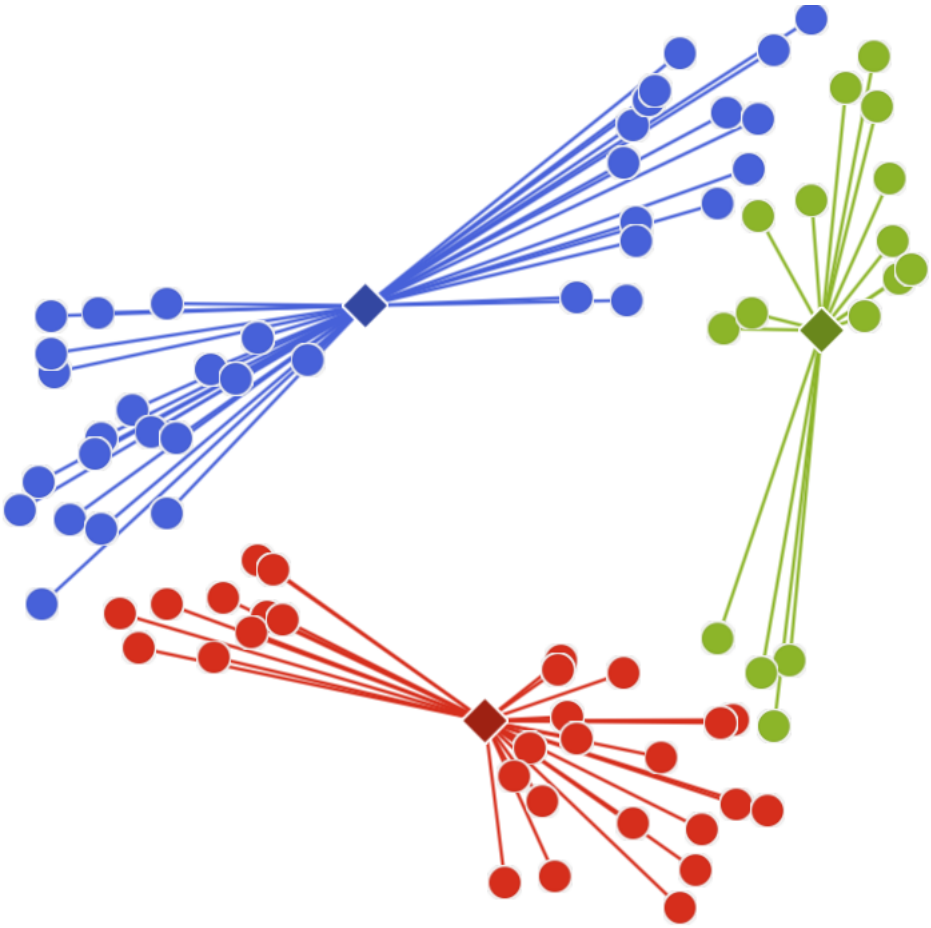
Từ bước tính toán trên, tiến hành phân loại các điểm về các cluster đã quyết định(cluster gần nó nhất). Vậy là đã phân ra được 3 cụm.



Hình 3.8: Phân thành 3 clusters.

Bước 5

Bước trên chúng ta đã thu được 3 cụm, bây giờ tiến hành tính trọng tâm của các điểm dữ liệu của từng cụm. Sau đó di chuyển điểm trung tâm của cụm sang vị trí vừa tính được.

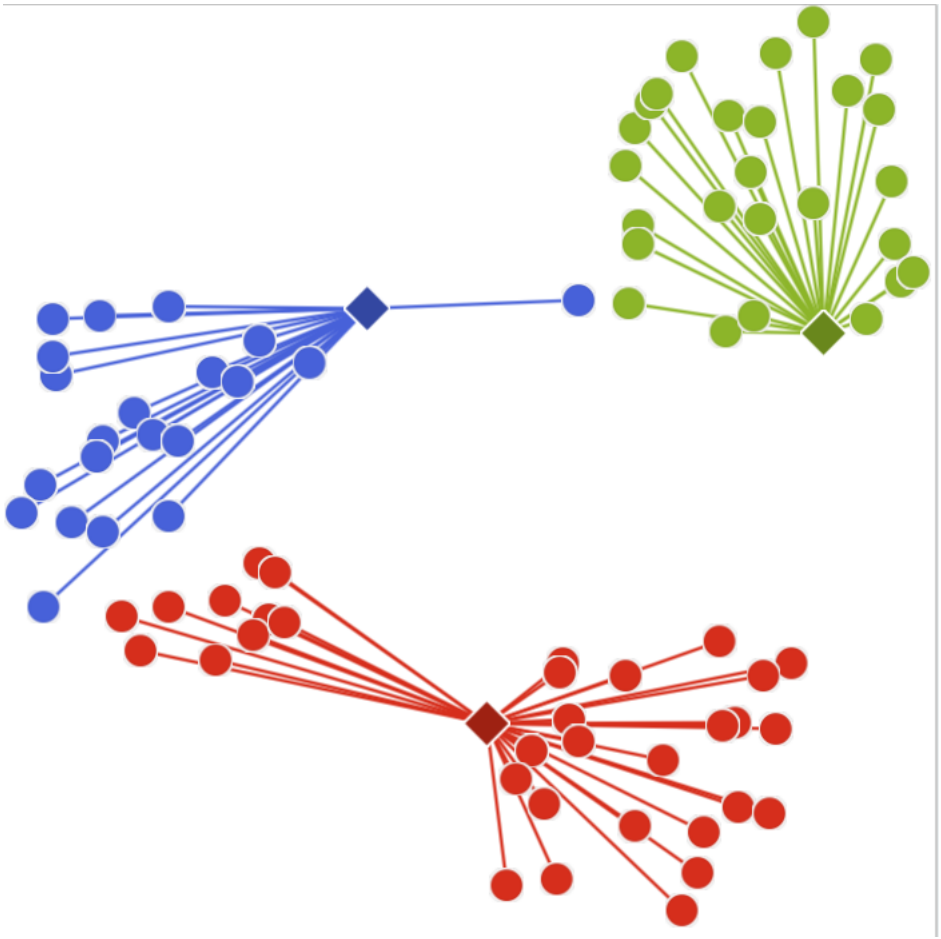


Hình 3.9: Tính trọng điểm.

Vị trí mà 3 điểm trung tâm của cluster vừa di chuyển đến được hiểu ngắn gọn chính là điểm trung tâm đang di chuyển đến vị trí chính xác hơn.

Bước 6

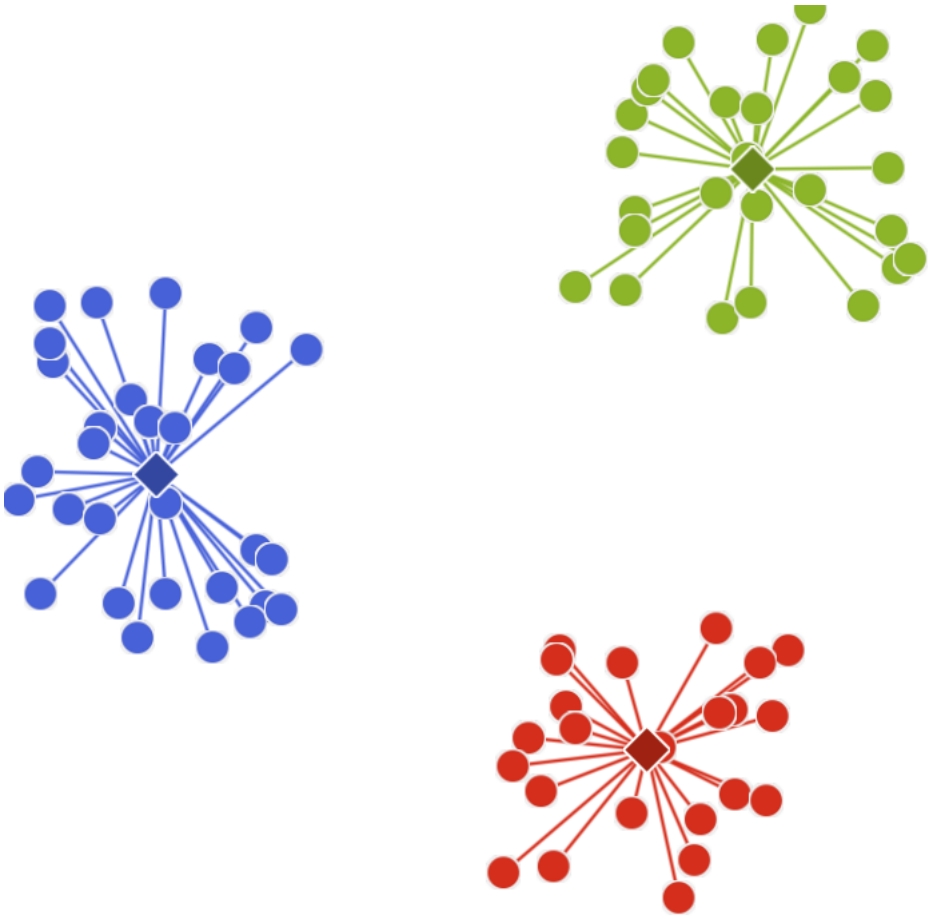
Một lần nữa tiến hành bước 3, tính toán lại khoảng các các điểm đến các điểm trung tâm. Sau đó phân loại lại các điểm dữ liệu về các cụm.



Hình 3.10: Tính toán lại khoảng cách.

Bước 7

Sau đó lặp lại quá trình di chuyển cluster trung tâm và phân loại lại các điểm về các cụm gần nhất. Quá trình này sẽ dừng khi sau khi dữ liệu sau khi phân cụm lại không thay đổi gì so với lần trước.



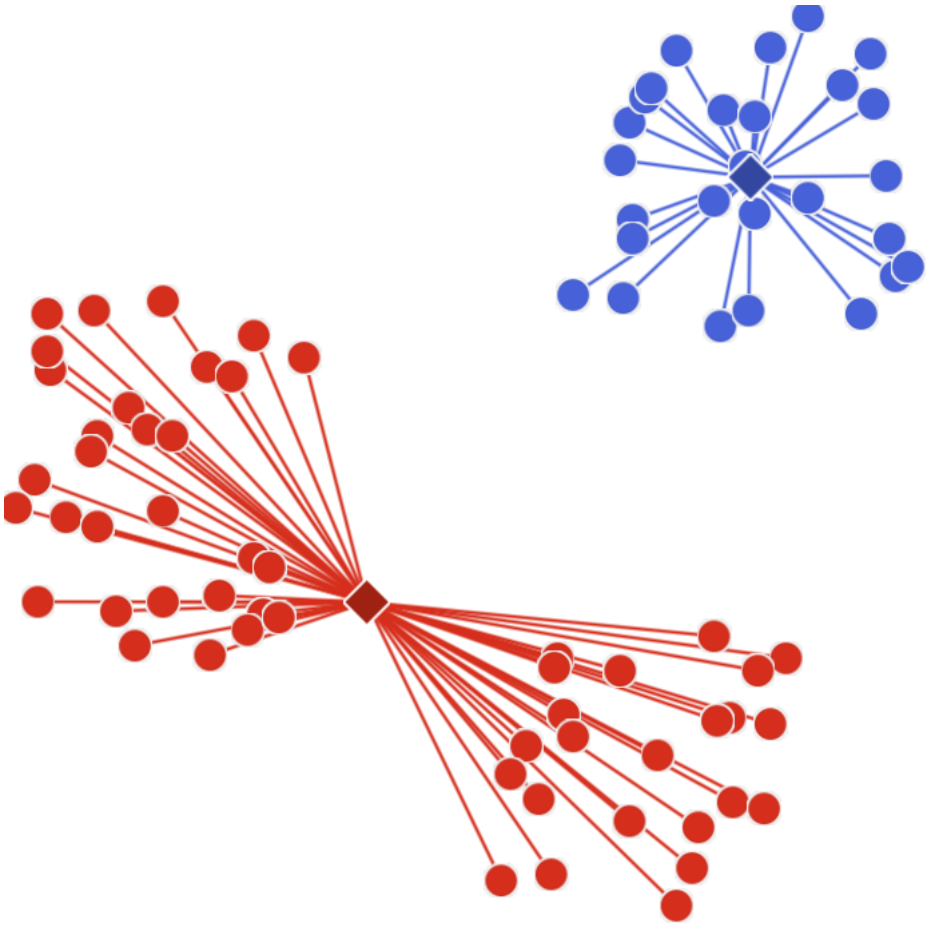
Hình 3.11: Kết quả.

3.3.3 Lưu ý

Trước khi sử dụng phương pháp này, chúng ta phải quyết định trước số lượng cluster, tuy nhiên trong quá trình tính toán số lượng cluster có thể khác với số lượng cluster mình dự đoán nên kết quả sẽ không chính xác.

Vì vậy để giải quyết vấn đề này, để có thể chọn ra số lượng cluster thích hợp thì cần phải phân tích dữ liệu cẩn thận, chạy thử k-means với nhiều biến số số lượng cluster.

Cùng ví dụ trên nếu thay số lượng cluster thành 2, kết quả phân loại sẽ thành ra như sau:



Hình 3.12: Khi chia thành 2 cụm.

3.4 Tập dữ liệu NSL-KDD

3.4.1 Giới thiệu

Bộ dữ liệu NSL-KDD được đề xuất để giải quyết một số vấn đề cố hữu của tập dữ liệu KDD CUP'99.

KDD CUP'99 là bộ dữ liệu được sử dụng rộng rãi nhất để phát hiện bất thường trong máy học. Nhưng Tavallaee và cộng sự đã tiến hành một phân tích thống kê về tập dữ liệu này và tìm thấy hai vấn đề quan trọng ảnh hưởng rất lớn đến hiệu năng của các hệ thống được đánh giá và kết quả đánh giá rất kém về các phương pháp phát hiện bất thường. Để giải quyết những vấn đề này, họ đề xuất một bộ dữ liệu mới, NSL-KDD, bao gồm các bản ghi đã chọn của bộ dữ liệu KDD hoàn chỉnh.

Sau đây là những ưu điểm của NSL-KDD so với bộ dữ liệu KDD gốc:

- Đầu tiên, nó không bao gồm các bản ghi dự phòng trong huấn luyện, vì vậy các bộ phân loại sẽ không bị thiên vị đối với các bản ghi thường xuyên hơn.
- Thứ hai, số lượng các bản ghi được chọn từ mỗi nhóm mức độ khó tỷ lệ nghịch với tỷ lệ phần trăm của các bản ghi trong bộ dữ liệu KDD gốc. Kết quả là, tỷ lệ phân loại của các phương pháp học máy riêng biệt khác nhau ở phạm vi rộng hơn, làm cho nó nhiều hiệu quả hơn để có một đánh giá chính xác về các kỹ thuật học tập khác nhau.
- Thứ ba, số lượng hồ sơ trong tập huấn và kiểm tra là hợp lý, điều này làm cho nó hợp lý để chạy thử nghiệm trên bộ hoàn chỉnh mà không cần phải chọn ngẫu nhiên một phần nhỏ. Do đó, kết quả đánh giá của các công trình nghiên cứu khác nhau sẽ nhất quán và có thể so sánh được.

3.4.2 Đặc tính

Số thứ tự	Tên	Mô tả	Dữ liệu
1	Duration	Độ dài thời gian của kết nối	0
2	Protocol_type	Giao thức được sử dụng trong kết nối	TCP
3	Service	Dịch vụ mạng đích được sử dụng	ftp_da
4	Flag	Trạng thái của kết nối – Normal hoặc Error	SF
5	Src_bytes	Số byte dữ liệu được chuyển từ nguồn đến đích trong một kết nối	491
6	Dst_bytes	Độ dài thời gian của kết nối	0

Bảng 3.1: Bảng đặc tính

Loại	Đặc tính
Nominal	protocol_type(2) service(3) flag(4)
Binary	land(7) logged_in(12) root_shell(14) su_attempted(15) is_host_login(21) is_guest_login(22)
Numeric	duration(1) src_bytes(5), dst_bytes(6) wrong_fragment(8) urgent(9) hot(10) num_failed_logins(11) num_compromised(13) num_root(16) num_file_creations(17) num_shells(18) num_access_files(19) num_outbound_cmds(20) count(23) srv_count(24) serror_rate(25) srv_serror_rate(26) rerror_rate(27) srv_rerror_rate(28) same_srv_rate(29) diff_srv_rate(30) srv_diff_host_rate(31) dst_host_count(32) dst_host_srv_count(33) dst_host_same_srv_rate(34) dst_host_diff_srv_rate(35) dst_host_same_src_port_rate(36) dst_host_srv_diff_host_rate(37) dst_host_serror_rate(38) dst_host_srv_serror_rate(39) dst_host_rerror_rate(40) dst_host_srv_rerror_rate(41)

Bảng 3.2: Bảng phân loại các đặc tính

3.4.3 Phân nhóm

Các lớp tấn công có trong bộ dữ liệu NSL-KDD được nhóm thành bốn loại:

1. **DOS**: Từ chối dịch vụ là một thể loại tấn công, làm cạn kiệt tài nguyên của nạn nhân qua đó

khiến cho nó không thể xử lý các yêu cầu hợp pháp - ví dụ: syn flood. Các mục có liên quan: “byte nguồn” và “tỷ lệ phần trăm của gói có lỗi ”

2. **Probing**: Mục tiêu của cuộc tấn công giám sát và thăm dò khác là thu thập thông tin về nạn nhân từ xa, ví dụ: quét cổng. Các mục có liên quan: “thời lượng kết nối” và “byte nguồn”

3. **U2R**: quyền truy cập trái phép vào đặc quyền người dùng (root) cục bộ là một kiểu tấn công, mà kẻ tấn công sử dụng tài khoản thông thường để đăng nhập vào hệ thống nạn nhân và cố gắng giành quyền root / quản trị viên bằng cách khai thác một số lỗ hổng trong nạn nhân. Các cuộc tấn công tràn bộ đệm. Các tính năng liên quan: “số lượng tệp được tạo” và “số lời nhắc shell được gọi,”

4. **R2L**: truy cập trái phép từ một máy từ xa, kẻ tấn công xâm nhập vào một máy từ xa và truy cập nội bộ của máy nạn nhân. Ví dụ: đoán mật khẩu. Các tính năng liên quan: Tính năng ở cấp mạng - "thời lượng kết nối"và "dịch vụ được yêu cầu"và các tính năng ở cấp máy chủ - "số lần đăng nhập không thành công"

3.5 Flatbuffer

FlatBuffers là một thư viện serialization đa nền tảng hiệu quả cho C++, C#, C, Go, Java, JavaScript, TypeScript, PHP và Python. Ban đầu nó được tạo ra tại Google để phát triển trò chơi và các ứng dụng quan trọng khác về hiệu suất.

3.5.1 Tại sao sử dụng FlatBuffers?

- **Truy cập vào dữ liệu được tuần tự hóa mà không cần phân tích cú pháp / giải nén** - Những gì đặt ngoài FlatBuffers là nó đại diện cho dữ liệu phân cấp trong một bộ đệm nhị phân phẳng theo cách mà nó vẫn có thể được truy cập trực tiếp mà không cần phân tích cú pháp / giải nén, trong khi vẫn hỗ trợ tiến hóa cấu trúc dữ liệu (chuyển tiếp / khả năng tương thích ngược).

- **Tốc độ và bộ nhớ hiệu quả** - Bộ nhớ duy nhất cần để truy cập dữ liệu của bạn là bộ đệm. Nó yêu cầu 0 phân bổ bổ sung (bằng C ++, các ngôn ngữ khác có thể thay đổi). FlatBuffers cũng rất thích hợp để sử dụng với mmap (hoặc streaming), chỉ yêu cầu một phần của bộ đệm có trong bộ nhớ. Truy cập gần với tốc độ truy cập cấu trúc thô chỉ với một thêm một hướng (một loại vtable) để cho phép phát triển định dạng và các trường tùy chọn. Đó là nhằm vào các dự án mà chi tiêu thời gian và không gian (phân bổ bộ nhớ nhiều) để có thể truy cập hoặc xây dựng dữ liệu tuần tự là không mong muốn, chẳng hạn như trong trò chơi hoặc bất kỳ ứng dụng nhạy cảm hiệu suất nào khác.

- **Linh hoạt** - Các trường không bắt buọc có nghĩa là không chỉ có khả năng tương thích tốt và tương thích ngược (ngày càng quan trọng đối với các ứng dụng tồn tại lâu dài: không phải cập nhật tất cả dữ liệu với mỗi phiên bản mới!). Nó cũng có nghĩa là bạn có nhiều lựa chọn trong dữ liệu nào bạn viết và dữ liệu nào bạn không sử dụng và cách bạn thiết kế cấu trúc dữ liệu.

- **Lượng mã nhỏ** - Một lượng nhỏ mã được tạo ra, rất dễ tích hợp.

- **Kiểu dữ liệu cứng** - Lỗi xảy ra tại thời gian biên dịch thay vì phải viết kiểm tra thời gian chạy lặp đi lặp lại và dễ bị lỗi. Mã có thể được sinh tự động.

- **Sử dụng thuận tiện** - Mã C++ được tạo cho phép truy cập và xây dựng mã. Sau đó, có chức năng tùy chọn để phân tích các lược đồ và các biểu diễn văn bản giống JSON khi chạy hiệu quả nếu cần (bộ nhớ nhanh hơn và hiệu quả hơn các trình phân tích cú pháp JSON khác).
  Mã Java và Go hỗ trợ tái sử dụng đối tượng. C# có các trình truy cập dựa trên cấu trúc hiệu quả.
- **Đa nền tảng và không có phụ thuộc** - mã C ++ sẽ hoạt động với bất kỳ gcc / clang và VS2010 gần đây nào. Đi kèm với các tệp xây dựng cho các thử nghiệm và mẫu (tệp .mk Android và cmake cho tất cả các nền tảng khác).

Chương 4

Triển khai và thử nghiệm

4.1 Mô hình hệ thống

Hệ thống được chia làm 2 phần:

1. Plugin cho Snort có nhiệm vụ bắt và xử lý gói tin và gửi đến máy chủ
2. Máy chủ đóng vai trò dự đoán gói tin là tốt hay xấu và đưa ra kết quả

4.1.1 Snort Plugin

```
static const InspectApi dpx_api{
        {PT_INSPECTOR,
        sizeof(InspectApi),
        INSAPI_VERSION,
        0,
        API_RESERVED,
        API_OPTIONS,
        s_name,
        s_help,
        mod_ctor,
        mod_dtor},
       IT_STREAM,
       PROTO_BIT__ANY_IP,
       nullptr, // buffers
       nullptr, // service
       nullptr, // pinit
       nullptr, // pterm
       nullptr, // tinit
       nullptr, // tterm
       dpx_ctor,
       dpx_dtor,
       nullptr, // ssn
       nullptr  // reset
};

SO_PUBLIC const BaseApi *snort_plugins[] =
        {
            &dpx_api.base,
            nullptr};
```

Hình 4.1: Đăng ký plugin với snort

Khi có gói tin mới đến, snort sẽ gọi đến plugin thông qua hàm eval để xử lý gói tin. Ở đây ta chỉ chấp nhận các gói tin UDP, TCP và ICMP. Nếu gói tin bất thường sẽ được gọi đến DetectionEngine của Snort

```
void Dpx::eval(Packet *packet) {
    if (packet->is_ip4() && (packet->is_tcp() || packet->is_udp() || packet->is_icmp())) {
        FeatureExtractor::IpFragment *frag = getIpFragment(packet);
        FeatureExtractor::Packet *datagr = nullptr;

        FeatureExtractor::Timestamp now = frag->get_end_ts();
        datagr = reasm->reassemble(frag);
        if (datagr) {
            conv_reconstructor->add_packet(datagr);
        } else {
            conv_reconstructor->report_time(now);
        }
        ++dpxstats.total_packets;
    }

    FeatureExtractor::Conversation *conv;
    while ((conv = conv_reconstructor->get_next_conversation()) != nullptr) {
        FeatureExtractor::ConversationFeatures *cf = stats_engine->calculate_features(conv);
        conv = nullptr;        // Should not be used anymore, object will commit suicide

        std::string result = predict(cf);
        if (result != "normal") {
            DetectionEngine::queue_event(DPX_GID, DPX_SID);
        }
        delete cf;
    }
}
```

Hình 4.2: Hàm xử lý khi có gói tin mới

Phân tích gói tin để lấy các thông tin cơ bản riêng biệt chuẩn bị để dự đoán

```
FeatureExtractor::IpFragment *Dpx::getIpFragment(Packet *packet) {
    FeatureExtractor::IpFragment *f = new FeatureExtractor::IpFragment();
    FeatureExtractor::Timestamp ts(packet->pkth->ts);
    f->set_start_ts(ts);
    f->set_length(packet->pkth->pktlen);

    if (!packet->is_eth()) {
        return f;
    }
    f->set_eth2(true);
    if (!packet->is_ip4()) {
        return f;
    }

    f->set_src_ip(packet->ptrs.ip_api.get_src()->get_ip4_value());
    f->set_dst_ip(packet->ptrs.ip_api.get_dst()->get_ip4_value());
    f->set_ip_proto((FeatureExtractor::ip_field_protocol_t) packet->ptrs.ip_api.proto());
    f->set_ip_id((uint16_t) (packet->ptrs.ip_api.id()));
    f->set_ip_flag_mf(packet->is_fragment());
    f->set_ip_frag_offset(packet->ptrs.ip_api.off());
    f->set_ip_payload_length(packet->ptrs.ip_api.pay_len());

    if (f->get_ip_frag_offset() > 0)
        return f;

    switch (f->get_ip_proto()) {
        case FeatureExtractor::TCP:
            f->set_src_port(packet->ptrs.tcph->src_port());
            f->set_dst_port(packet->ptrs.tcph->dst_port());
            f->set_tcp_flags(packet->ptrs.tcph->th_flags);
            break;
```

Hình 4.3: Phân tích các thông tin cơ bản của gói tin

```
        case FeatureExtractor::UDP:
            f->set_src_port(packet->ptrs.udph->src_port());
            f->set_dst_port(packet->ptrs.udph->dst_port());
            break;

        case FeatureExtractor::ICMP:
            f->set_icmp_type((FeatureExtractor::icmp_field_type_t) packet->ptrs.icmph->type);
            f->set_icmp_code(packet->ptrs.icmph->code);
            break;

        default:
            break;
    }
    return f;
}
```

Hình 4.4: Phân tích các thông tin cơ bản của gói tin

Sau khi trích xuất các thông tin cần thiết từ gói tin, dữ liệu được tách thành 41 features cho dự đoán, được mã hóa bằng Flatbuffer

```
std::string Dpx::predict(FeatureExtractor::ConversationFeatures *cf) {
    flatbuffers::FlatBufferBuilder builder(1024);
    auto protocolType = builder.CreateString(cf->get_conversation()->get_protocol_type_str());
    auto service = builder.CreateString(cf->get_conversation()->get_service_str());
    auto flag = builder.CreateString(cf->get_conversation()->get_state_str());

    kmeans::PacketBuilder packetBuilder(builder);
    packetBuilder.add_duration(cf->get_conversation()->get_duration_ms() / 1000);
    packetBuilder.add_protocol_type(protocolType);
    packetBuilder.add_service(service);
    packetBuilder.add_flag(flag);
    packetBuilder.add_src_bytes(cf->get_conversation()->get_src_bytes());
    packetBuilder.add_dst_bytes(cf->get_conversation()->get_dst_bytes());
    packetBuilder.add_land(cf->get_conversation()->land());
    packetBuilder.add_wrong_fragment(cf->get_conversation()->get_wrong_fragments());
    packetBuilder.add_urgent(cf->get_conversation()->get_urgent_packets());
    packetBuilder.add_hot(0);
    packetBuilder.add_num_failed_logins(0);
    packetBuilder.add_logged_in(0);
    packetBuilder.add_num_compromised(0);
    packetBuilder.add_root_shell(0);
    packetBuilder.add_su_attempted(0);
    packetBuilder.add_num_root(0);
    packetBuilder.add_num_file_creations(0);
    packetBuilder.add_num_shells(0);
    packetBuilder.add_num_access_files(0);
    packetBuilder.add_num_outbound_cmds(0);
    packetBuilder.add_is_host_login(0);
    packetBuilder.add_is_guest_login(0);
```

Hình 4.5: Mã hóa thông tin gói tin dùng Flatbuffer

```
packetBuilder.add_count(cf->get_count());
packetBuilder.add_srv_count(cf->get_srv_count());
packetBuilder.add_serror_rate(cf->get_serror_rate());
packetBuilder.add_srv_serror_rate(cf->get_srv_serror_rate());
packetBuilder.add_rerror_rate(cf->get_rerror_rate());
packetBuilder.add_srv_rerror_rate(cf->get_srv_rerror_rate());
packetBuilder.add_same_srv_rate(cf->get_same_srv_rate());
packetBuilder.add_diff_srv_rate(cf->get_diff_srv_rate());
packetBuilder.add_srv_diff_host_rate(cf->get_srv_diff_host_rate());
packetBuilder.add_dst_host_count(cf->get_dst_host_count());
packetBuilder.add_dst_host_srv_count(cf->get_dst_host_srv_count());
packetBuilder.add_dst_host_same_srv_rate(cf->get_dst_host_same_srv_rate());
packetBuilder.add_dst_host_diff_srv_rate(cf->get_dst_host_diff_srv_rate());
packetBuilder.add_dst_host_same_src_port_rate(cf->get_dst_host_same_src_port_rate());
packetBuilder.add_dst_host_srv_diff_host_rate(cf->get_dst_host_srv_diff_host_rate());
packetBuilder.add_dst_host_serror_rate(cf->get_dst_host_serror_rate());
packetBuilder.add_dst_host_srv_serror_rate(cf->get_dst_host_srv_serror_rate());
packetBuilder.add_dst_host_rerror_rate(cf->get_dst_host_rerror_rate());
packetBuilder.add_dst_host_srv_rerror_rate(cf->get_dst_host_srv_rerror_rate());
auto orc = packetBuilder.Finish();
builder.Finish(orc);
uint8_t *buf = builder.GetBufferPointer();
int size = builder.GetSize();

std::string result = request(buf, size);
return result;
```

Hình 4.6: Mã hóa thông tin gói tin dùng Flatbuffer

Sau khi đã phân tích và mã hóa dùng Flatbuffer, chuỗi đã mã hóa được đến máy chủ dự đoán dùng libcurl

```cpp
std::string Dpx::request(uint8_t *buf, int size) {
    std::string readBuffer;

    curl_global_init(CURL_GLOBAL_ALL);
    CURL *ctx = curl_easy_init();
    curl_easy_setopt(ctx, CURLOPT_URL, host.c_str());

    curl_easy_setopt(ctx, CURLOPT_POST, 1);
    curl_easy_setopt(ctx, CURLOPT_POSTFIELDS, buf);
    curl_easy_setopt(ctx, CURLOPT_POSTFIELDSIZE, size);
    curl_easy_setopt(ctx, CURLOPT_WRITEFUNCTION, WriteCallback);
    curl_easy_setopt(ctx, CURLOPT_WRITEDATA, &readBuffer);

    struct curl_slist *headers = NULL;
    headers = curl_slist_append(headers, "Content-Type: application/octet-stream");
    curl_easy_setopt(ctx, CURLOPT_HTTPHEADER, headers);

    CURLcode ret = curl_easy_perform(ctx);
    if (ret != CURLE_OK)
        fprintf(stderr, "curl_easy_perform() failed: %s\n",
                curl_easy_strerror(ret));

    curl_slist_free_all(headers);
    curl_easy_cleanup(ctx);

    return readBuffer;
}
```

Hình 4.7: Gọi đến máy chủ dự đoán

4.1.2 Máy chủ dự đoán

Máy chủ dự đoán có nhiệm vụ huấn luyện mạng dự đoán, khởi tạo endpoint /predict để plugin snort gửi yêu cầu http đến kiểm tra gói tin. Máy chủ cũng cung cấp trang thống kê đơn giản để người dùng quan sát

```python
if __name__ == '__main__':
    kmean.load_training_data('datasets/KDDTrain+.csv')
    kmean.train_clf()

    from gevent import pywsgi
    from geventwebsocket.handler import WebSocketHandler
    server = pywsgi.WSGIServer(('', 5000), app, handler_class=WebSocketHandler)
    server.serve_forever()
```

Hình 4.8: Máy chủ dự đoán

```python
def load_training_data(self, filepath):
    data, labels = self.load_data(filepath)
    self.cols = data.columns
    self.training = [data, labels]

def load_test_data(self, filepath):
    data, labels = self.load_data(filepath)
    map_data = pd.DataFrame(columns=self.cols)
    map_data = map_data.append(data)
    data = map_data.fillna(0)
    self.testing = [data[self.cols], labels]

@staticmethod
def load_data(filepath):
    data = pd.read_csv(filepath, names=COL_NAMES, index_col=False)
    # Shuffle data
    data = data.sample(frac=1).reset_index(drop=True)
    NOM_IND = [1, 2, 3]
    BIN_IND = [6, 11, 13, 14, 20, 21]
    # Need to find the numerical columns for normalization
    NUM_IND = list(set(range(40)).difference(NOM_IND).difference(BIN_IND))

    # Scale all numerical data to [0-1]
    data.iloc[:, NUM_IND] = minmax_scale(data.iloc[:, NUM_IND])
    labels = data['labels']
    del data['labels']
    data = pd.get_dummies(data)
    return [data, labels]
```

Hình 4.9: Nhập và xử lý tập dữ liệu

```python
def train_clf(self):
    self.clf = KMeans(n_clusters=4, init='random').fit(self.training[0])
    self.set_categories()
```

Hình 4.10: Dùng thuật toán KMeans tạo 4 clusters

Endpoint sẽ phân tích request lấy và giải mã Flatbuffer để lấy được cái thông tin gói tin để đưa vào mạng dự đoán

```python
@app.route('/predict', methods=['POST'])
def predict_packet():
    raw_data = request.get_data()
    data = Packet.GetRootAsPacket(raw_data, 0)
    packet = {
        "duration": data.Duration(),
        "protocol_type": data.ProtocolType(),
        "service": data.Service(),
        "flag": data.Flag(),
        "src_bytes": data.SrcBytes(),
        "dst_bytes": data.DstBytes(),
        "land": data.Land(),
        "wrong_fragment": data.WrongFragment(),
        "urgent": data.Urgent(),
        "hot": data.Hot(),
        "num_failed_logins": data.NumFailedLogins(),
        "logged_in": data.LoggedIn(),
        "num_compromised": data.NumCompromised(),
        "root_shell": data.RootShell(),
        "su_attempted": data.SuAttempted(),
        "num_root": data.NumRoot(),
        "num_file_creations": data.NumFileCreations(),
        "num_shells": data.NumShells(),
        "num_access_files": data.NumAccessFiles(),
        "num_outbound_cmds": data.NumOutboundCmds(),
        "is_host_login": data.IsHostLogin(),
        "is_guest_login": data.IsGuestLogin(),
        "count": data.Count(),
```

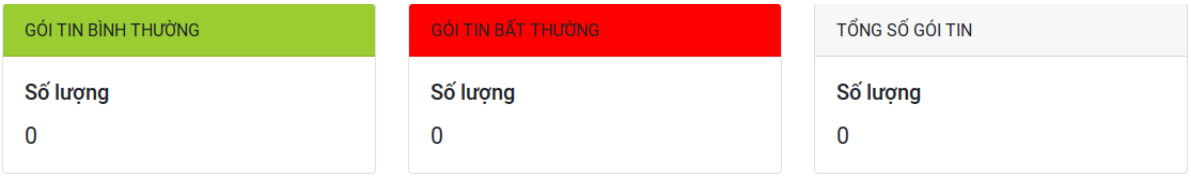
Hình 4.11: Nhận và giải mã Flatbuffer

Sau khi đã giải mã thành công, sẽ đưa vào mạng dự đoán Kmeans để dự đoán và trả về kết quả cho Snort plugin

```
result = kmean.predict(packet)
if (result != "normal"):
    global anomaly
    anomaly = anomaly + 1
else:
    global normal
    normal = normal + 1
return result
```

Hình 4.12: Dự đoán gói tin

4.2 Thử nghiệm và nhận xét



Hình 4.13: Màn hình thống kê ban đầu

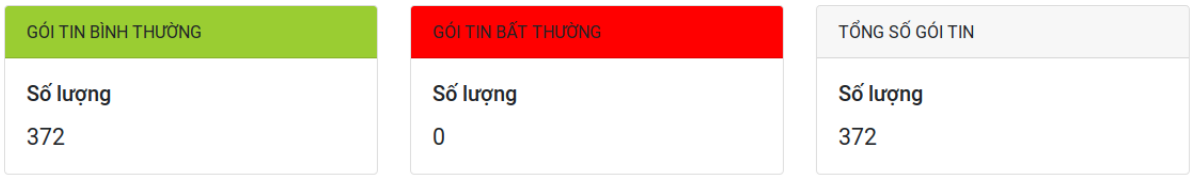
```
dthongvl@thongku:~/workspace/ids-machine-learning$ docker-compose up
Starting ids-machine-learning_snort_1           ... done
Recreating ids-machine-learning_predict-server_1 ... done
Attaching to ids-machine-learning_snort_1, ids-machine-learning_predict-server_1

predict-server_1  | /usr/local/lib/python2.7/site-packages/pandas/core/frame.py:6211: FutureWarning: Sorting because non-
concatenation axis is not aligned. A future version
predict-server_1  | of pandas will change to not sort by default.
predict-server_1  |
predict-server_1  | To accept the future behavior, pass 'sort=False'.
predict-server_1  |
predict-server_1  | To retain the current behavior and silence the warning, pass 'sort=True'.
predict-server_1  |
predict-server_1  |   sort=sort)
```

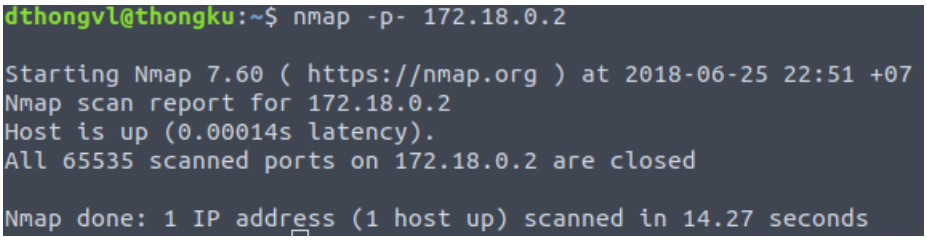
Hình 4.14: Khởi tạo Docker Compose để thử nghiệm

```
dthongvl@thongku:~$ sudo hping3 --flood --rand-source --udp -p 445 172.18.0.2
HPING 172.18.0.2 (br-fecc119967b8 172.18.0.2): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 172.18.0.2 hping statistic ---
734480 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

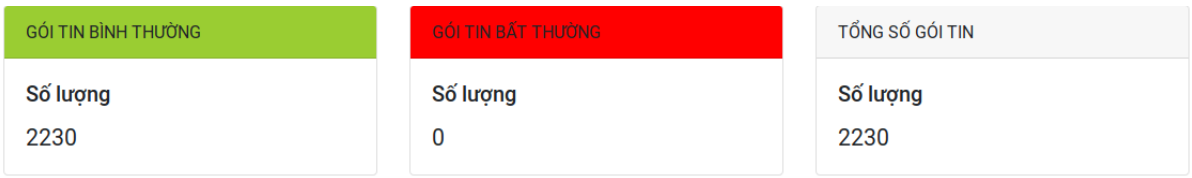
Hình 4.15: Thử tấn công Ping Flood sử dụng công cụ hping3



Hình 4.16: Không phát hiện được tấn công



Hình 4.17: Thử tấn công Scan port bằng công cụ nmap



Hình 4.18: Không phát hiện được tấn công

Chương 5

Kết luận

5.1 Nhận định

Nhóm vẫn chưa thể triển khai thành công hệ thống phát triển xâm nhập dựa trên kĩ thuật máy học, tuy nhiên nhóm đã hoàn thành được bộ khung hoạt động. Nhóm đã đút kết được nhiều kiến thức từ quá trình phát triển.

Sử dụng thuật toán KMeans vẫn chưa mang lại độ chính xác cao, chỉ ở mức 74%. Kèm theo đó, tập dữ liệu NSL-KDD đã cũ và khó có thể nhận biết được các hình thức tấn công mới. Khả năng hoạt động thực tế khi máy chủ dự đoán dùng Python vẫn còn khá chậm.

5.2 Hướng phát triển

- Tăng tốc độ dự đoán và trả lời của máy chủ
- Cải thiện độ chính xác của mạng bằng các thuật toán khác hoặc kết hợp nhiều thuật toán. Ví dụ: Kmean kết hợp Random Forest.
- Sử dụng các tập dữ liệu mới và thực tế hơn.
- Cải thiện giao diện thống kê và các chức năng cảnh báo.

Tài liệu tham khảo