ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Lê Việt Hoàng

CẢI THIỆN HIỆU NĂNG THUẬT TOÁN PHÁT HIỆN BẤT THƯỜNG MIDAS-R TRONG AN NINH MẠNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành : Điện Tử Viễn Thông

HÀ NỘI, 2023

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Lê Việt Hoàng

CẢI THIỆN HIỆU NĂNG THUẬT TOÁN PHÁT HIỆN BẤT THƯỜNG MIDAS-R TRONG AN NINH MẠNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành : Điện Tử Viễn Thông

Cán bộ hướng dẫn: PGS.TS.Nguyễn Nam Hoàng

HÀ NỘI, 2023

Lời cảm ơn

Lời đầu tiên, tôi xin chân thành cảm ơn các thầy cô giáo trong khoa Công nghệ thông tin, trường Đại Học Công Nghệ - Đại học Quốc Gia Hà Nội đã tận tình chỉ bảo, giảng dạy và truyền đạt những kinh nghiệm, kiến thức vô cùng quý báu cho tôi trong thời gian học tập tại trường.

Tôi xin gửi lời cảm ơn đến thầy giáo, PGS.TS. Nguyễn Nam Hoàng – Khoa ĐTVT đã định hướng đề tài, nhận xét, giúp đỡ và hướng dẫn cho tôi trong suốt quá trình làm đồ án. Cuối cùng, tôi xin được cảm ơn sâu sắc đến gia đình và bạn bè đã luôn giúp đỡ, hỗ trợ và đóng góp ý kiến để giúp tôi hoàn thành đồ án.

Do thời gian, kiến thức và kinh nghiệm của tôi còn hạn chế nên luận văn không thể tránh khỏi những sai sót. tôi hy vọng sẽ nhận được những ý kiến nhận xét, góp ý của các thầy cô giáo và các bạn để đồ án được hoàn thiện hơn. tôi xin chân thành cảm ơn!

Lời cam đoan

Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án **Cải thiện hiệu năng thuật toán phát hiện bất thường MIDAS-R trong an ninh mạng** là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ, các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

Hà Nội, ngày tháng năm 2023

Tác giả

Nguyễn Lê Việt Hoàng

TÓM TẮT

An toàn không gian mạng là một trong những vấn đề được quan tâm nhiều nhất hiện nay. Cùng với đó, đại dịch COVID-19 đã làm bùng nổ nhu cầu sử dụng Internet khiến cho các cuộc tấn công mạng trở nên phổ biến hơn bao giờ hết. Sự gia tăng nhanh chóng của tội phạm mạng đã thúc đẩy sự cần thiết của việc cảnh báo các cuộc tấn công đa dạng cũng như phát hiện các bất thường chính xác và kịp thời. Điều này đặt ra thách thức cho các hệ thống an ninh mạng khi mà những phương pháp phát hiện hiện nay không thể đáp ứng cả hai tiêu chí trên. Gần đây, một thuật toán phát hiện bất thường mới có tên MIDAS đã được đề xuất với tiềm năng là lời giải cho vấn đề này. Tuy nhiên, MIDAS vẫn còn tồn tại những hạn chế về độ chính xác và biến thể MIDAS-R đã ra đời nhằm khắc phục những nhược điểm này nhưng có sự đánh đổi giữa độ chính xác và hiệu năng của thuật toán, ngoài ra hai thuật toán nêu trên chưa có khả năng thích ứng trong môi trường có lưu lượng mạng cao và thay đổi nhanh.

Trong đồ án này, tôi đề xuất các điều chỉnh cho thuật toán MIDAS-R nhằm giải quyết những vấn đề trên bằng cách thực hiện hàng loạt phân tích và đưa ra cải tiến phù hợp cho từng thành phần của thuật toán mà không làm thay đổi nền tảng lý thuyết của nó. Cụ thể, tôi sẽ sửa đổi cấu trúc dữ liệu phác thảo CMS trong MIDAS-R dựa trên các ý tưởng của NitroSketch, đưa ra một phương pháp tạo giá trị băm hiệu quả hơn và áp dụng các kỹ thuật lập trình như SIMD, Branchless Programming, Caching,.. nhằm nâng cao hiệu suất trong khi vẫn giữ được độ chính xác của thuật toán gốc. Kết quả thử nghiệm cho thấy thuật toán mới có hiệu suất tổng thể cao hơn khoảng 50% so với thuật toán ban đầu đồng thời cũng thích nghi tốt hơn với đa dạng lưu lượng mạng. Điều này đã nâng cao hơn nữa khả năng sử dụng của MIDAS-R trong thực tế.

Từ khóa: An ninh mạng, Phát hiện bất thường

Mục lục

Danh sách hình vẽ

Danh sách bảng

Thuật ngữ viết tắt

AIDS	Anomaly-based Intrusion Detection System	Hệ thống phát hiện xâm nhập dựa trên bất thường
AUC	Area Under ROC Curve	Diện tích bên dưới đường cong ROC
AVX	Advanced Vector Extensions	Phần mở rộng Vector nâng cao
CMS	Count-Min Sketch	Bản phác thảo đếm tối thiểu
CPU	Center Processing Unit	Đơn vị xử lý trung tâm
IDS	Intrusion Detection System	Hệ thống phát hiện xâm nhập
MIDAS	Microcluster-Based Detection of Anomalies in Edge Streams	Phát hiện bất thường trong luồng cạnh dựa trên cụm vi mô
NS	NitroSketch based base structure	Cấu trúc dữ liệu phác thảo dựa trên NitroSketch
ROC	Receiver Operating Characteristic	Đường cong đặc trưng của bộ thu nhận
SIDS	Signature-based Intrusion Detection System	Hệ thống phát hiện xâm nhập dựa trên chữ ký
SIMD	Single Instruction, Multiple Data	Xử lý đa dữ liệu đơn lệnh

Chương 1

TỔNG QUAN

Chương này nêu định nghĩa về ngành an ninh mạng, đánh giá thực trạng về an toàn không gian mạng hiện nay và khảo sát các phương pháp được sử dụng trong IDS(Intrusion Detection System) để phát hiện các cuộc tấn công mạng.

1.1 Giới thiệu về an ninh mạng

An ninh mạng là một tập hợp các quy tắc và cấu hình được thiết kế để bảo vệ tính toàn vẹn, bảo mật và khả năng truy cập của mạng máy tính và dữ liệu bằng cả công nghệ phần mềm và phần cứng. Ngành an ninh mạng đang phát triển nhanh chóng do nhu cầu ngày càng tăng đối với các giải pháp và dịch vụ có thể ngăn chặn, phát hiện và giảm thiểu các cuộc tấn công mạng vào các doanh nghiệp, cơ quan chính phủ và cá nhân. Các yếu tố chính thúc đẩy sự tăng trưởng của thị trường của ngành này bao gồm số lượng các mối đe dọa mạng ngày càng tăng, việc áp dụng công nghệ đám mây và di động và nhu cầu làm việc từ xa an toàn.

1.2 Thực trạng an ninh mạng hiện nay

Theo báo cáo của AAG[7] năm 2023, bối cảnh an ninh mạng toàn cầu đã chứng kiến các mối đe dọa gia tăng trong những năm gần đây. Trong đại dịch, tội phạm mạng đã lợi dụng các mạng bị sai lệch khi các doanh nghiệp chuyển sang

môi trường làm việc từ xa. Năm 2020, các cuộc tấn công bằng phần mềm độc hại tăng 358% so với năm 2019.

Từ đây, các cuộc tấn công mạng trên toàn cầu đã tăng 125% cho đến năm 2021 và số lượng các cuộc tấn công mạng ngày càng tăng tiếp tục đe dọa các doanh nghiệp và cá nhân vào năm 2022.

Dưới đây là một số thông tin đáng chú ý:

- Lừa đảo(Phishing) vẫn là hình thức phổ biến nhất của tội phạm trực tuyến. Vào năm 2021, 323.972 người dùng internet được cho là nạn nhân của các cuộc tấn công lừa đảo. Điều này có nghĩa là một nửa số người dùng bị vi phạm dữ liệu đã rơi vào một cuộc tấn công lừa đảo.
- Gần 1 tỷ email đã bị lộ trong một năm, ảnh hưởng đến 1/5 người dùng internet.
- Vi phạm dữ liệu khiến các doanh nghiệp thiệt hại trung bình 4,35 triệu đô la vào năm 2022.
- Khoảng 236,1 triệu cuộc tấn công ransomware đã xảy ra trên toàn cầu trong nửa đầu năm 2022.
- Cứ 2 người dùng internet ở Mỹ thì có 1 người bị xâm phạm tài khoản vào năm 2021.
- 39% doanh nghiệp ở Vương quốc Anh cho biết đã bị tấn công mạng vào năm 2022.
- Khoảng 1 trong 10 tổ chức của Hoa Kỳ không có bảo hiểm chống lại các cuộc tấn công mạng.
- 53,35 Công dân Hoa Kỳ bị ảnh hưởng bởi tội phạm mạng trong nửa đầu năm 2022.
- Tội phạm mạng khiến các doanh nghiệp ở Vương quốc Anh thiệt hại trung bình £4200 vào năm 2022.
- Năm 2020, các cuộc tấn công bằng phần mềm độc hại đã tăng 358% so với năm 2019.

1.3 IDS và các phương pháp phát hiện tấn công

1.3.1 IDS - Intrusion Detection System

Hệ thống phát hiện xâm nhập(IDS) là một ứng dụng phần mềm giám sát các hoạt động của mạng hoặc hệ thống và phân tích chúng để tìm các dấu hiệu vi phạm chính sách, cách sử dụng được chấp nhận hoặc các biện pháp bảo mật tiêu chuẩn. Sau đó, nó sẽ báo cáo mọi hoạt động độc hại hoặc vi phạm chính sách cho quản trị viên hệ thống. IDS có thể giúp bảo vệ một tổ chức khỏi các cuộc tấn công mạng bằng cách phát hiện và cảnh báo về các mối đe dọa tiềm ẩn trước khi chúng gây ra thiệt hại.

1.3.2 Các phương pháp phát hiện tấn công trong IDS

Có thể phân loại các IDS theo phương pháp phát hiện như sau:

- IDS dựa trên chữ ký(SIDS) - Signature-based IDS: Hệ thống phát hiện xâm nhập dựa trên chữ ký(SIDS) sử dụng các kỹ thuật khớp mẫu để tìm ra một cuộc tấn công đã biết; chúng còn được gọi là Phát hiện dựa trên tri thức - Knowledge-based Detection[8]. Trong SIDS, các phương thức đối sánh được sử dụng để tìm ra sự xâm nhập trước đó. Nói cách khác, khi một chữ ký xâm nhập phù hợp với chữ ký của một lần xâm nhập trước đó đã tồn tại trong cơ sở dữ liệu chữ ký, tín hiệu báo động được kích hoạt. Đối với SIDS, nhật ký của máy chủ được kiểm tra để tìm chuỗi các lệnh hoặc hành động trước đây đã được xác định là phần mềm độc hại.

- IDS dựa trên sự bất thường(AIDS) - Anomaly-based IDS: AIDS đã thu hút sự quan tâm của rất nhiều học giả do có khả năng khắc phục hạn chế của SIDS. Trong AIDS, một mô hình bình thường của hành vi của một hệ thống máy tính là được tạo bằng cách sử dụng máy học(machine learing), dựa trên thống kê(statistical-based) hoặc dựa trên tri thức(knowledge-based). Bất kỳ sai lệch đáng kể nào giữa hành vi được quan sát và mô hình đều được coi là như một sự bất thường, có thể được hiểu là một sự xâm nhập(intrusion). Giả định cho nhóm kỹ thuật này là hành vi ác ý khác với hành vi thông thường

của người dùng. Các hành vi của người dùng bất thường không giống với các hành vi tiêu chuẩn được phân loại là xâm nhập. Quá trình của AIDS bao gồm hai giai đoạn: giai đoạn đào tạo và giai đoạn thử nghiệm. Trong giai đoạn đào tạo, hồ sơ lưu lượng truy cập bình thường được sử dụng để tìm hiểu một mô hình hành vi là bình thường và sau đó trong giai đoạn thử nghiệm, một bộ dữ liệu mới được sử dụng để thiết lập khả năng khái quát hóa của hệ thống đối với các cuộc xâm nhập chưa từng thấy trước đây. AIDS có thể được phân loại thành một số loại dựa trên phương pháp được sử dụng cho đào tạo như: dựa trên Thống Kê, dựa trên Tri Thức và dựa trên Học Máy [9].

Mỗi phương pháp đều có những ưu và nhược điểm riêng, điều này được tóm gọn trong Bảng 1.1 dưới đây:

Bảng 1.1: So sánh các phương pháp phát hiện tấn công trong IDS

		Ưu điểm	**Nhược điểm**
Phương pháp phát hiện	SIDS	- Rất hiệu quả trong việc xác định xâm nhập với báo động sai tối thiểu (FA). - Phát hiện kịp thời các hành vi xâm nhập. - Vượt trội để phát hiện các cuộc tấn công đã biết. - Thiết kế đơn giản.	- Cần cập nhật chữ ký mới thường xuyên. - SIDS được thiết kế để phát hiện các cuộc tấn công đối với các chữ ký đã biết. Khi trước đó xâm nhập đã được thay đổi một chút thành một biến thể mới, thì hệ thống sẽ không thể xác định độ lệch mới này của cuộc tấn công tương tự. - Không thể phát hiện cuộc tấn công zero-day. - Không phù hợp để phát hiện các cuộc tấn công nhiều bước. - Ít hiểu biết sâu sắc về các cuộc tấn công
	AIDS	- Có thể được sử dụng để phát hiện các cuộc tấn công mới. - Có thể được sử dụng để tạo chữ ký xâm nhập.	- AIDS không thể xử lý các gói được mã hóa, vì vậy cuộc tấn công có thể không bị phát hiện và có thể đưa ra một mối đe dọa. - Báo động dương tính giả cao. - Khó xây dựng một hồ sơ bình thường cho một hệ thống máy tính rất năng động. - Cảnh báo chưa được phân loại. - Cần đào tạo ban đầu.

Gần đây, một thuật toán phát hiện bất thường mới được đề xuất được gọi là MIDAS hứa hẹn mang ưu điểm của cả hai nhóm trên đồng thời hạn chế nhược điểm của các phương pháp được sử dụng trong AIDS.

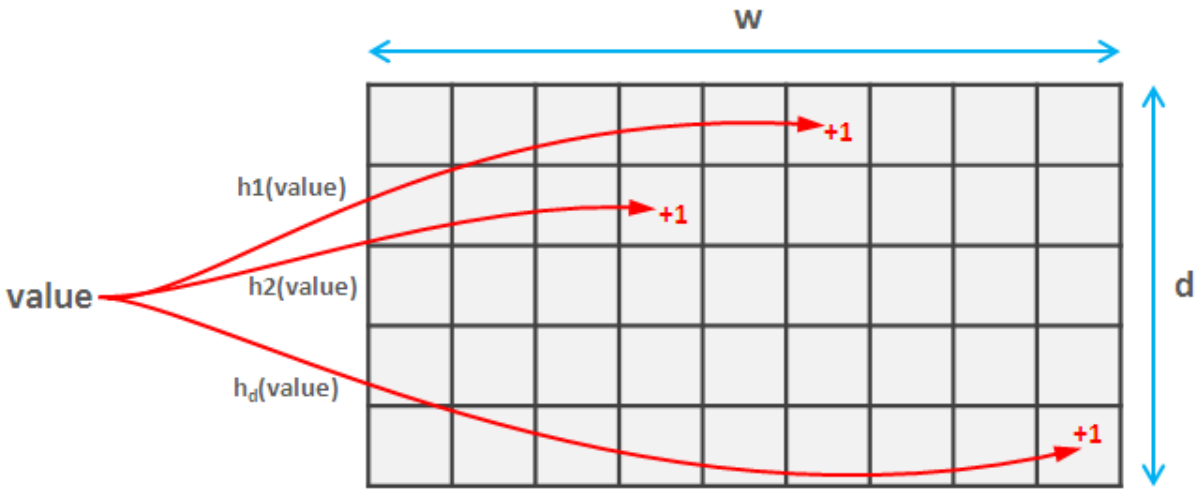
1.4 Thuật toán MIDAS

Phần này sẽ giới thiệu về CMS, cấu trúc dữ liệu phác thảo được sử dụng trong MIDAS. Tiếp đến là tổng quan về MIDAS và biến thể của nó, MIDAS-R. Cuối cùng là đánh giá hiệu quả của chúng.

1.4.1 Count-Min Sketch - Bản phác thảo đếm tối thiểu

Câu hỏi đặt ra là: Cho một chuỗi dữ liệu liên tục, làm thế nào để biết số lần xuất hiện của một giá trị bất kỳ trong chuỗi đó?

CMS - Count-Min Sketch[10] là một cấu trúc dữ liệu trả lời câu hỏi trên với kích thước bộ nhớ cố định, thời gian truy vấn không đổi và lưu trữ gần đúng số lần xuất hiện của các giá trị trong chuỗi dữ liệu. Ý tưởng cơ bản của Count-Min Sketch khá đơn giản, nó chỉ là một mảng hai chiều $(d \times w)$ của các bộ đếm số nguyên. Khi một giá trị đến, nó được ánh xạ tới một vị trí tại mỗi d hàng bằng cách sử dụng d hàm băm khác nhau. Bộ đếm trên mỗi vị trí được tăng lên 1 đơn vị. Quá trình này được thể hiện trong hình dưới đây:



Hình 1.1: Minh họa Count-Min Sketch[1]

Để truy vấn, ta chỉ cần trả về giá trị nhỏ nhất của các bộ đếm tại các vị trí được ánh xạ tới giá trị đó. Sự phụ thuộc giữa kích thước bản phác thảo và độ chính xác[11] được thể hiện trong công thức bên dưới với chiều rộng w và độ sâu d:

$$w = \left\lceil \frac{2}{\epsilon} \right\rceil \qquad \epsilon : \text{Lỗi ước tính} \tag{1.1}$$

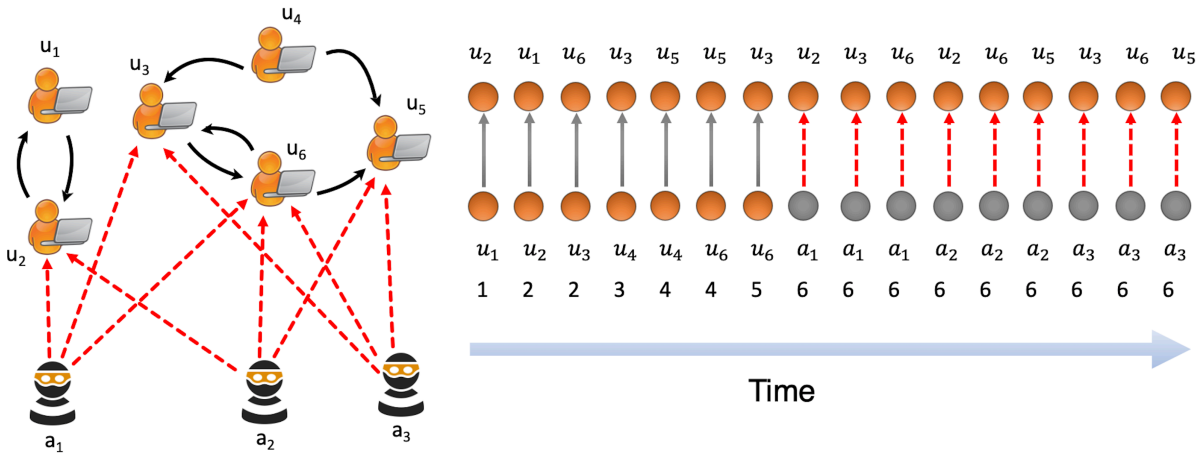
$$d = \left\lceil \frac{\ln(1-\delta)}{\ln \frac{1}{2}} \right\rceil \qquad \delta : \text{Độ tin cậy} \tag{1.2}$$

Tóm lại hoạt động của CMS gồm:

- Khởi tạo(Init):
  - Bước 1: Tạo một bản phác thảo CMS với kích thước $(d \times w)$.
  - Bước 2: Đặt tất cả các bộ đếm(hay phần tử trong mảng) về 0.
- Thêm giá trị(Update):
  - Bước 1: Tạo các giá trị băm sử dụng d hàm băm khác nhau.
  - Bước 2: Ánh xạ các giá trị băm đó tới một vị trí tại mỗi hàng.
  - Bước 3: Tăng bộ đếm tại các vị trí đó lên 1 đơn vị.
- Truy vấn giá trị(Query):
  - Bước 1: Tạo các giá trị băm sử dụng d hàm băm khác nhau.
  - Bước 2: Ánh xạ các giá trị băm đó tới một vị trí bộ đếm tại mỗi hàng.
  - Bước 3: Trả về giá trị nhỏ nhất của các bộ đếm tại các vị trí đó.

1.4.2 Sơ lược về thuật toán MIDAS

MIDAS được thiết kế nhằm trả lời câu hỏi sau: Đưa ra một luồng các cạnh đồ thị từ một đồ thị động, làm cách nào chúng ta có thể gán điểm bất thường cho các cạnh theo cách trực tuyến, với mục đích phát hiện hành vi bất thường, sử dụng thời gian và bộ nhớ không đổi?



Hình 1.2: Minh họa cuộc tấn công mạng[2]

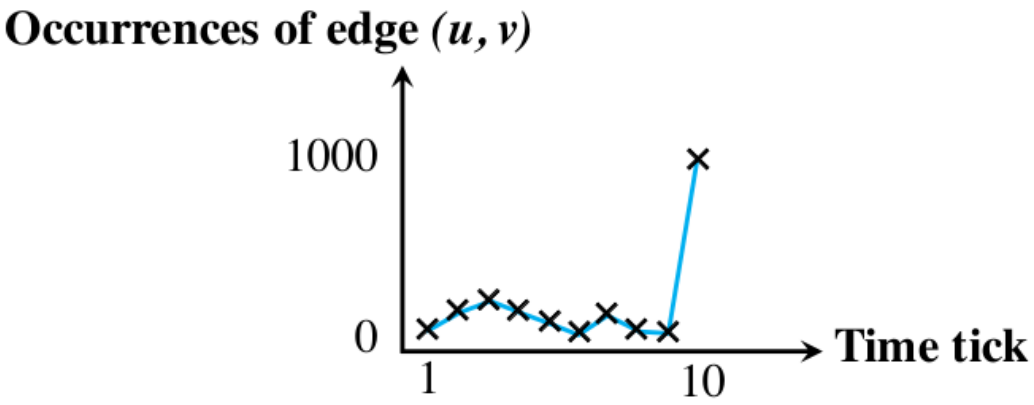
Trong trường hợp này, các tác giả lập mô hình các nút(Node) như máy tính xách tay, máy tính để bàn là nút của đồ thị, một kết nối giữa chúng được xem như là một cạnh(Edge). Ví dụ như trong Hình 1.2 thể hiện một cuộc tấn công mạng, các nút là các máy tính của kẻ tấn công cũng như người dùng, các cạnh được tạo thành từ 2 nút. Khi máy tính của người dùng bị tấn công, sẽ xuất hiện nhiều cạnh đến thiết bị(hay nút) của người đó.

Các sự kiện gian lận hoặc bất thường trong nhiều ứng dụng xảy ra trong các cụm vi mô(Microcluster) hoặc đột nhiên đến các nhóm có cạnh giống nhau đáng ngờ, ví dụ: tấn công từ chối dịch vụ(Dos) trong dữ liệu lưu lượng mạng. Các phương thức hiện có xử lý các luồng cạnh trong một cách trực tuyến nhằm mục đích phát hiện các cạnh bất ngờ riêng lẻ, không phải các cụm vi mô và do đó có thể bỏ lỡ một lượng lớn hoạt động đáng ngờ.

Mô tả vấn đề: Cho $E = \{e_1, e_2, \dots\}$ là một dòng các cạnh từ đồ thị tiến hóa theo thời gian G. Mỗi cạnh tới là một bộ $e_i = (u_i, v_i, t_i)$ bao gồm nút nguồn $u_i \in \mathcal{V}$, nút đích $v_i \in \mathcal{V}$ và thời điểm xuất hiện t_i là thời điểm mà tại đó cạnh được thêm vào biểu đồ. Ví dụ, trong một luồng lưu lượng mạng, một cạnh e_i có thể đại diện cho một kết nối được tạo từ địa chỉ IP nguồn u_i đến địa chỉ IP đích v_i tại thời điểm t_i . Chúng ta không giả sử rằng tập nút $\mathcal{V}$ đã biết trước. Ví dụ: địa chỉ IP hoặc ID người dùng mới có thể được tạo trong suốt quá trình. chúng ta mô hình hóa G dưới dạng đồ thị có hướng. Đồ thị vô hướng có thể được xử lý đơn giản bằng coi một cạnh vô hướng đến $e_i = (u_i, v_i, t_i)$ là hai cạnh có hướng đồng thời, một trong hai hướng. chúng ta cũng cho phép G là một đa đồ thị: các cạnh có thể

được tạo nhiều lần giữa cùng một cặp nút. Các cạnh được phép đến đồng thời(tức là $t_i + 1 \geq t_i$), vì trong nhiều ứng dụng, ti được đưa ra dưới dạng các dấu thời gian rời rạc.

Cấu trúc dữ liệu trực tuyến: Xem xét ví dụ trong Hình 1.3 về một cặp nguồn-đích (u, v), cho thấy một đợt hoạt động lớn tại thời điểm 10. Đợt hoạt động này là ví dụ đơn giản nhất của một cụm vi mô, vì nó bao gồm một nhóm lớn các cạnh rất giống với một khác(trên thực tế giống hệt nhau), cả về mặt không gian (tức là về các nút mà chúng kết nối) và tạm thời.



Hình 1.3: Chuỗi thời gian của một cặp nguồn-đích (u, v), với một đợt hoạt động lớn tại thời điểm 10.[3]

Trong cài đặt ngoại tuyến, có nhiều phương pháp chuỗi thời gian có thể phát hiện bùng nổ hoạt động như vậy. Tuy nhiên, trong cài đặt trực tuyến, hãy nhớ rằng chúng ta muốn sử dụng bộ nhớ được giới hạn, vì vậy không thể theo dõi dù chỉ một chuỗi thời gian như vậy. Hơn thế nữa, có nhiều cặp nguồn-đích như vậy và tập hợp các nguồn và đích không cố định trước.

Để khắc phục những vấn đề này, MIDAS duy trì hai loại cấu trúc dữ liệu phác thảo Count-Min(CMS). Giả sử chúng ta đang ở một thời điểm cố định cụ thể t vào luồng; coi thời gian là một biến rời rạc để đơn giản. Gọi s_{uv} là tổng số cạnh từ u đến v tính đến thời điểm hiện tại. Sau đó sử dụng một dữ liệu CMS duy nhất cấu trúc để duy trì xấp xỉ tất cả số lượng như vậy s_{uv}(cho tất cả các cạnh uv) bộ nhớ không đổi: bất cứ lúc nào, chúng ta có thể truy vấn cấu trúc dữ liệu để lấy gần đúng đếm $\hat{s}_{uv}$. Thứ hai, gọi a_{uv} là số cạnh từ u đến v trong khoảng thời gian

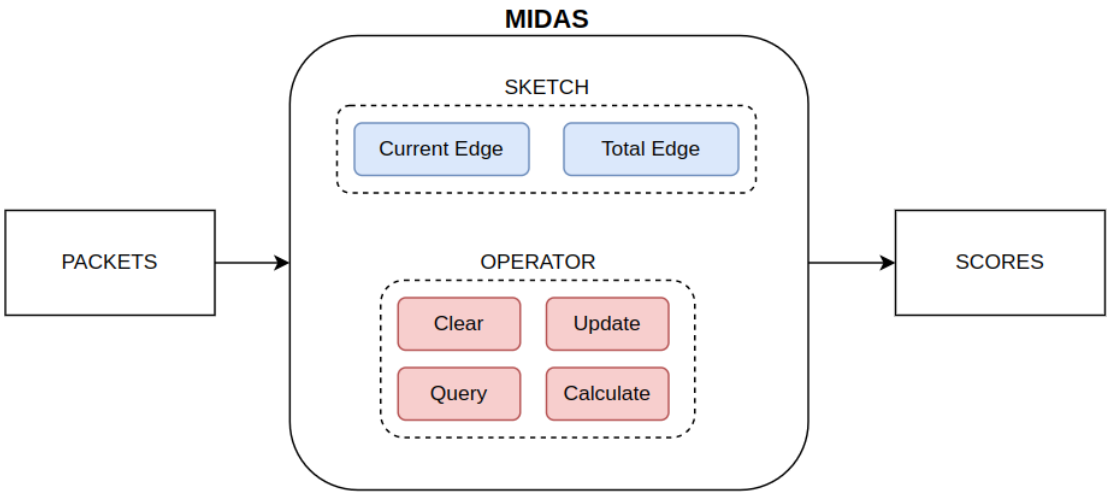
hiện tại(nhưng không bao gồm các dấu tích thời gian trong quá khứ). Ta theo dõi a_{uv} bằng dữ liệu CMS tương tự cấu trúc, sự khác biệt duy nhất là chúng ta đặt lại cấu trúc dữ liệu CMS này mỗi thời gian chúng ta chuyển sang đánh dấu thời gian tiếp theo. Do đó, cấu trúc dữ liệu CMS này cung cấp đếm gần đúng $\hat{a}_{uv}$ cho số cạnh từ u đến v trong khoảng thời gian hiện tại t.

Tính điểm bất thường: MIDAS sử dụng một giả định rằng mức trung bình (tức là tỷ lệ trung bình tại các cạnh xuất hiện) trong dấu thời gian hiện tại (ví dụ: $t = 10$) giống như mức trung bình trước thời điểm hiện tại($t < 10$). Lưu ý rằng điều này tránh giả định bất kỳ phân phối cụ thể nào cho mỗi lần đánh dấu thời gian và cũng tránh giả định nghiêm ngặt về sự tĩnh tại theo thời gian. Do đó, chúng ta có thể chia các cạnh quá khứ thành hai lớp: đánh dấu thời gian hiện tại($t = 10$) và tất cả các tích tắc trong quá khứ ($t < 10$). Nhắc lại ký hiệu trước đây của chúng ta, các số sự kiện tại ($t = 10$) là a_{uv} ,trong khi số cạnh trong thời gian qua thời điểm($t < 10$) là $s_{uv} - a_{uv}$. Theo bài kiểm tra mức độ phù hợp chi bình phương, thống kê chi bình phương được định nghĩa là 2 tổng trên các hạng mục(được quan sát−dự kiến). Trong trường hợp này, danh mục của chúng ta là $t = 10$ hy vọng và $t < 10$. Theo giả định mức trung bình của chúng ta, vì chúng ta có tổng số cạnh s_{uv}(đối với trường hợp này cặp nguồn-đích), số dự kiến tại $t = 10$ là s_{uv}, và dự kiến t số cho $t < 10$ là số còn lại, tức là $\frac{t-1}{t}s_{uv}$. Lưu ý rằng cả a_{uv} và s_{uv} đều có thể được ước tính bằng cấu trúc dữ liệu CMS, thu được xấp xỉ $\hat{a}_{uv}$ và $\hat{s}_{uv}$ tương ứng. Điều này dẫn đến điểm bất thường sau đây của chúng ta, sử dụng mà chúng ta có thể đánh giá một cạnh mới đến với cặp nguồn-đích (u, v). Với một cạnh mới đến (u, v, t), điểm bất thường của chúng ta được tính theo công thức 1.3 sau:

$$score(u, v, t) = (\hat{a}_{uv} - \frac{\hat{s}_{uv}}{t})^2(\frac{t^2}{\hat{s}_{uv}(t-1)}) \tag{1.3}$$

Tóm lại, thuật toán MIDAS sử dụng bản phác thảo đếm tối thiểu (CMS) để đếm số lần xuất hiện của cạnh trong mỗi dấu thời gian, sau đó sử dụng kiểm tra chi bình phương để đánh giá mức độ sai lệch và tạo ra một số điểm đại diện cho sự bất thường, điều này được thể hiện trong Thuật toán 1. Điểm càng cao thì độ bất thường của cạnh càng lớn. Phương pháp được đề xuất sử dụng bộ nhớ không đổi và có độ phức tạp thời gian không đổi khi xử lý từng cạnh. Ngoài ra, bằng cách sử dụng một nguyên tắc khuôn khổ kiểm định giả thuyết[3], MIDAS cung cấp các

giới hạn lý thuyết về giả thuyết xác suất dương mà những phương pháp tương tự khác([12],[13]) không cung cấp. Sơ đồ 1.4 mô tả cấu tạo của MIDAS và cách nó tính điểm bất thường cho một các gói tin.



Hình 1.4: Sơ đồ hoạt động của MIDAS

Trong đó:

- PACKETS: Danh sách các gói tin mô phỏng một mạng thực.
- SKETCH: Nhóm gồm các cấu trúc dữ liệu CMS để ước lượng số lần xuất hiện của từng cạnh.
  - *CurrentEdge*: Đếm số lượng cạnh trong thời điểm hiện tại.
  - *TotalEdge*: Đếm tổng số cạnh từ khi bắt đầu theo dõi.
- OPERATOR: Nhóm chứa các thao tác với các cấu trúc trong khối SKETCH
  - *Clear*: Xóa *CurrentEdge*, tức là đặt các giá trị trong đó về 0.
  - *Update*: Thêm một cạnh vào các cấu trúc dữ liệu SKETCH.
  - *Query*: Truy vấn một cạnh trong các cấu trúc dữ liệu SKETCH.
  - *Calculate*: Tính toán điểm bất thường cho từng gói tin bằng kiểm định giả thiết Chi-Square để xác định sự tương quan giữa các mạng.
- SCORES: Điểm bất thường của các gói tin.

Algorithm 1: MIDAS: Streaming Anomaly Scoring[3]

Data: Stream of graph edges over time

Output: Anomaly scores per edge

▷ **Initialize CMS data structures:**

Initialize CMS for total count s_{uv} and current count a_{av}

while *new edge* $e = (u, v, t)$ *is received* : **do**

  ▷ **Update Counts:**

  Update CMS data structures for the new edge uv

  ▷ **Query Counts:**

  Retrieve updated counts $\hat{s}_{uv}$ and $\hat{a}_{uv}$

  ▷ **Anomaly Score:**

  output score$(u, v, t) = (\hat{a}_{uv} - \frac{\hat{s}_{uv}}{t})^2(\frac{t^2}{\hat{s}_{uv}(t-1)})$

end

Thuật toán 1 mô tả cách hoạt động của MIDAS. Theo đó, các bộ đếm trong CMS được xóa sau mỗi lần thay đổi dấu thời gian. Tuy nhiên, một số bất thường vẫn tồn tại trong nhiều dấu thời gian. Các tác giả của MIDAS đã đề xuất một biến thể được gọi là MIDAS-R có thể khắc phục điều này.

1.4.3 Thuật toán MIDAS-R

Phần này mô tả cách tiếp cận của MIDAS-R[14], phương pháp này xem xét các cạnh liên quan: nghĩa là, nó nhằm mục đích nhóm các cạnh gần nhau lại với nhau, hoặc theo thời gian hoặc không gian.Thuật toán MIDAS-R được tóm gọn trong Thuật toán 2.

MIDAS-R duy trì một phần thông tin trong dấu thời gian trước đó cho phép tiếp theo thuật toán để nhanh chóng tạo ra điểm cao khi cạnh xuất hiện trở lại. Nó cũng coi các nút nguồn và đích là thông tin bổ sung giúp xác định các cạnh bất thường.

Quan hệ tạm thời: Thay vì chỉ đếm các cạnh trong cùng một khoảng thời gian (như đã làm ở MIDAS), các cạnh trong quá khứ gần đây cũng sẽ được tính vào dấu thời gian hiện tại, nhưng được sửa đổi bằng cách giảm cân nặng. Một cách đơn giản và hiệu quả để thực hiện việc này bằng cấu trúc dữ liệu CMS như sau: vào cuối mỗi lần đánh dấu, thay vì đặt lại cấu trúc dữ liệu CMS đối với a_{uv}, ta sẽ nhân tất cả giá trị của nó với một hệ số số cố định $factor \in (0, 1)$. Điều này cho phép các cạnh trong quá khứ được tính vào dấu thời gian t hiện tại, với trọng số giảm dần.

Quan hệ không gian: Nắm bắt các nhóm lớn không gian gần đó các cạnh: ví dụ: một địa chỉ IP nguồn duy nhất đột nhiên tạo ra một số lượng lớn các cạnh đến nhiều đích hoặc một nhóm nhỏ các nút đột nhiên tạo ra một số lượng lớn các cạnh giữa chúng. Một trực giác đơn giản là ở một trong hai hai trường hợp này, quan sát các nút có sự xuất hiện đột ngột của một lượng lớn số cạnh. Do đó, chúng ta có thể sử dụng cấu trúc dữ liệu CMS để theo dõi cạnh đếm như trước, ngoại trừ đếm tất cả các cạnh liền kề với bất kỳ nút u. Cụ thể, tạo bộ đếm CMS cho $\hat{a}_u$ và $\hat{s}_u$ để tính gần đúng số lượng cạnh hiện tại và tổng số liền kề với nút u. Với mỗi cạnh tới (u, v), chúng ta có thể tính toán ba điểm bất thường: một cho cạnh (u, v), như trong thuật toán trước đây; một cho nút nguồn u và một cho nút đích v. Cuối cùng là kết hợp ba điểm bằng cách lấy giá trị lớn nhất của chúng.

Algorithm 2: MIDAS-R: Incorporating Relations[14]

Data: Stream of graph edges over time

Output: Anomaly scores per edge

▷ **Initialize CMS data structures:**

Initialize CMS for total count s_{uv} and current count a_{uv}

Initialize CMS for total count s_u, s_v and current count a_u, a_v

while *new edge* $e = (u, v, t)$ *is received* : **do**

  ▷ **Update Counts:**

  Update CMS data structures for the new edge uv, source node u and destination node v

  ▷ **Query Counts:**

  Retrieve updated counts $\hat{s}_{uv}$ and $\hat{a}_{uv}$

  Retrieve updated counts $\hat{s}_u, \hat{s}_v$ and $\hat{a}_u, \hat{a}_v$

  ▷ **Compute Edge Score:**

  $\text{score}((u, v, t)) = (\hat{a}_{uv} - \frac{\hat{s}_{uv}}{t})^2(\frac{t^2}{\hat{s}_{uv}(t-1)})$

  ▷ **Compute Node Scores:**

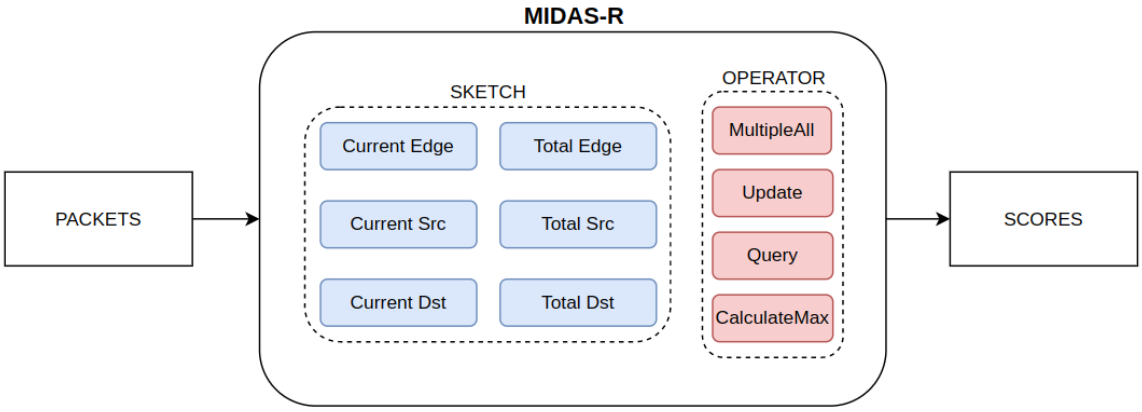
  $\text{score}((u, t)) = (\hat{a}_u - \frac{\hat{s}_u}{t})^2(\frac{t^2}{\hat{s}_u(t-1)})$

  $\text{score}((v, t)) = (\hat{a}_v - \frac{\hat{s}_v}{t})^2(\frac{t^2}{\hat{s}_v(t-1)})$

  ▷ **Final Score:**

  output $\max\{\text{score}(u, v, t), \text{score}(u, t), \text{score}(v, t)\}$

end



Hình 1.5: Sơ đồ hoạt động của MIDAS-R

Hình 1.5 minh họa cách MIDAS-R tính điểm bất thường, các khối hoàn toàn tương tự như trong MIDAS được đề cập ở Hình 1.4 với các sửa đổi sau:

- Khối SKETCH được bổ sung thêm 4 cấu trúc dữ liệu:
  - $CurrentSrc$: Đếm số lượng nút nguồn hiện tại.
  - $TotalSrc$: Đếm tổng số nút nguồn từ khi bắt đầu theo dõi.
  - $CurrentDst$: Đếm số lượng nút đích hiện tại.
  - $TotalDst$: Đếm tổng số nút đích từ khi băt đầu theo dõi.
- Khối OPERATOR gồm các thao tác:
  - $MultipleAll$: Thay thế cho thao tác $Clear$ trong MIDAS, nó thực hiện nhân tất cả giá trị trong cấu trúc dữ liệu $Current$ SKETCH với trọng số $factor$ như đề cập ở trên.
  - $CalculateMax$: Tính toán độ tương quan từ các cấu trúc dữ liệu $Edge$, Src và Dst SKETCH rồi lấy giá trị lớn nhất để làm điểm bất thường thay vì chỉ sử dụng $Edge$ SKETCH như MIDAS.

1.4.4 Đánh giá và nêu vấn đề

Các tác giả của MIDAS đã tiến hành đo lường[3], kết quả được mô tả trong bảng sau:

Bảng 1.2: Độ chính xác(độ lệch chuẩn)

Dataset	PEN miner	F-FADE	SEDAN SPOT	MIDAS	MIDAS-R
DARPA	0.8267	0.8451	0.6442	0.9042(0.0032)	**0.9514**(0.0012)
CTU-13	0.6041	0.8028	0.6397	0.9079(0.0049)	**0.9703**(0.0009)
UNSW -NB15	0.7028	0.6858	0.7575	0.8843(0.0079)	**0.8952**(0.0028)

Bảng 1.3: Thời gian chạy

Dataset	PEN miner	F-FADE	SEDAN SPOT	MIDAS	MIDAS-R
DARPA	20423s	325.1s	67.54s	**0.09**s	0.30s
CTU-13	10065s	844.2s	38.73s	**0.05**s	0.21s
UNSW -NB15	12857s	2267s	48.03s	**0.06**s	0.15s

Từ các khái niệm và hai bảng nêu trên, có thể đưa ra những ưu điểm sau của MIDAS-R:

- Có độ chính xác vượt trội so với các thuật toán khác.
- Chỉ cần sử dụng dữ liệu địa chỉ IP và dấu thời gian để phát hiện các bất thường thay vì cần rất nhiều trường như trong các phương pháp dựa trên học máy, thống kê, tri thức...
- Thời gian chạy nhanh hơn nhiều so với những thuật toán còn lại(trừ MIDAS).
- Tìm sự bất thường trong biểu đồ động/phát triển theo thời gian(Phát hiện xâm nhập, xếp hạng giả mạo, gian lận tài chính).
- Phát hiện sự bất thường cụm vi mô(đột nhiên đến các nhóm cạnh giống nhau đáng ngờ, ví dụ: tấn công DoS).

- Đảm bảo lý thuyết về xác suất dương tính giả.
- Bộ nhớ không đổi(không phụ thuộc vào kích thước đồ thị).
- Thời gian cập nhật liên tục (phát hiện bất thường theo thời gian thực để giảm thiểu tác hại).
- Chính xác hơn đáng kể so với các phương pháp tiếp cận hiện đại khác.

Tuy nhiên, MIDAS-R còn tồn tại các vấn đề sau:

- Thời gian xử lý chậm hơn MIDAS tới 3 lần.
- Chỉ có thể xử lý tuần tự tất cả các gói tin, điều này là không khả thi trong môi trường có lưu lượng mạng lớn và thay đổi đột ngột, nhất là trong giai đoạn bùng nổ truy cập Internet như hiện nay.

1.5 Nội dung đồ án

1.5.1 Mục tiêu và phương pháp

- Mục tiêu:
  - Cải thiện hiệu năng thuật toán MIDAS-R ít nhất 40% so với thuật toán ban đầu trong khi vẫn đảm bảo độ chính xác.
  - Bổ sung khả năng thích ứng trong điều kiện lưu lượng mạng thay đổi theo thời gian.
- Phương pháp: Để giải quyết vấn đề này, đầu tiên cần tiến hành phân tích điểm nghẽn hiệu năng tại từng bước xử lý trong MIDAS-R và đưa ra cách cải tiến phù hợp.

Lưu ý: Trong khuôn khổ đồ án này, tôi chỉ tập trung cải tiến thuật toán MIDAS-R. Tuy nhiên các phương pháp cải thiện được đề xuất hoàn toàn có thể áp dụng cho các thuật toán MIDAS và các biến thể của nó.

1.5.2 Phạm vi của đồ án

- Toàn bộ việc triển khai thuật toán MIDAS-R được thực hiện hoàn toàn trên máy tính.
- Dữ liệu tấn công đã được tiền xử lý dưới dạng file CSV thu thập từ các các tổ chức an ninh mạng.

1.5.3 Phần mềm và công cụ

- Ngôn ngữ lập trình C được sử dụng để triển khai thuật toán.
- Ngôn ngữ script Python dùng cho tổng hợp và trực quan hóa số liệu.
- Công cụ Make để tự động hóa quá trình biên dịch và liên kết.
- Đo lường hiệu năng: perf, flamegraph.
- Môi trường triển khai trên hệ điều hành Linux.

Chương 2

PHÂN TÍCH VÀ CẢI THIỆN THUẬT TOÁN

Trong chương này, tôi sẽ tìm cách cải thiện thuật toán Phần 2.1 tiến hành đo đạc các chỉ số về hiệu năng của thuật toán MIDAS-R. Từ đó đưa ra các đánh giá và đề xuất hướng cải tiến. Dựa trên những đề xuất này, các Phần 2.2, 2.3, 2.4, 2.5 lần lượt đưa ra các cái thiện hiệu năng cho từng thành phần của MIDAS-R. Phần 2.6 cuối cùng tổng hợp các cải tiến trên và đưa ra nhận xét về thuật toán mới.

2.1 Đo lường, phân tích và phương hướng

2.1.1 Chuẩn bị

1. Dataset: Dữ liệu được sử dụng là CIC-DDOS2019[15], đây là tập dữ liệu tấn công chứa hơn 20 triệu gói tin đã được tiền xử lý gồm các trường: IP nguồn, IP đích, Timestamp, Nhãn(cho biết đó có phải là gói tin tấn công hay không). Cụ thể sẽ được trình bày ở Mục 3.1.

2. Cài đặt thuật toán: Tôi sẽ triển khai thuật toán MIDAS-R bằng ngôn ngữ lập trình C với các thành phần như đã mô tả ở sơ đồ trong Hình 1.5 gồm:

    - Các cấu trúc trong khối SKETCH: Sử dụng cấu trúc dữ liệu CMS với d

$= 8$, $w = 4096$. Thuật toán hash được chọn là xxHash[16] vì nó là nhanh nhất về tổng thể và đã được áp dụng rộng rãi[17].

- Thao tác *Calculate*: Dòng 10-16 trong Thuật toán 2.

Triển khai 1 cho thấy trình tự hoạt động cụ thể của thuật toán: Biến **midasR** gồm khối SKETCH như đã đề cập ở Mục 1.4.3 với kích thước $(d \times w)$, trọng số **factor** được đặt thành 0.9 để có độ chính xác tốt nhất[14], **input** chứa IP nguồn/đích và Timestamp của gói tin. Duyệt lần lượt tất cả gói tin, cập nhật bộ đếm về số lượng cạnh và nút tạo bởi địa chỉ IP của gói tin vào khối SKETCH bằng hàm **cms_add()**(dòng 9-18). Khi duyệt đến gói tin ở thời điểm tiếp theo, tất cả bộ đếm trong các cấu trúc *Current* sẽ được nhân với trọng số **factor** với hàm **multipleAll()**(dòng 2-8). Tính điểm bất thường cho mỗi cấu trúc trong SKETCH bằng hàm **ComputeScore()** nhận đầu vào là số lượng cạnh và nút của gói tin đó sử dụng hàm **cms_check()** để truy vấn, cuối cùng là tính điểm bất thường cho gói tin bằng cách lấy giá trị lớn nhất của chúng(dòng 19-24).

```
double midasROperator(MidasR *midasR, Input input) {
  if (input.ts > midasR->current_ts) {
    double factor = midasR->factor;
    multipleAll(&(midasR->numCurrentEdge), factor);
    multipleAll(&(midasR->numCurrentSrc), factor);
    multipleAll(&(midasR->numCurrentDst), factor);
    midasR->current_ts = input.ts;}
  char src[32], dst[32], edge[32];
  sprintf(edge, "%d", combine(input.src, input.dst));
  cms_add(&(midasR->numCurrentEdge), edge);
  cms_add(&(midasR->numTotalEdge), edge);
  sprintf(src, "%d", input.src);
  cms_add(&(midasR->numCurrentSrc), src);
  cms_add(&(midasR->numTotalSrc), src);
  sprintf(dst, "%d", input.dst);
  cms_add(&(midasR->numCurrentDst), dst);
  cms_add(&(midasR->numTotalDst), dst);
  return max(ComputeScore(cms_check(&(midasR->numCurrentEdge), edge),
                          cms_check(&(midasR->numTotalEdge), edge), input.ts),
             ComputeScore(cms_check(&(midasR->numCurrentSrc), src),
                          cms_check(&(midasR->numTotalSrc), src), input.ts),
             ComputeScore(cms_check(&(midasR->numCurrentDst), dst),
                          cms_check(&(midasR->numTotalDst), dst), input.ts));}
```

Listing 1: Triển khai thực tế của MIDAS-R

2.1.2 Đánh giá và đề xuất

Những hoạt động tiêu tốn nhiều thời gian nhất mô tả ở Bảng 2.1 và 2.2 được tính toán sử dụng các công cụ benchmark Flamegraph và Perf. Ở đây, tôi sẽ phân loại mức tiêu thụ CPU theo các hàm và các đơn vị tính toán, điều này sẽ cho biết chi tiết hơn về các hoạt động logic lẫn tính toán phần cứng. Cụ thể:

- Phân loại theo hàm chỉ đơn giản là đọc kết quả benchmark với các hàm tương ứng.
- Phân loại theo đơn vị tính toán là phân rã các thao tác của hàm đến đơn vị nhỏ nhất(tức là đến cấp độ mà nếu tiến đến sâu hơn sẽ không có ý nghĩa về

mặt hiệu năng tính toán); ví dụ: trong hàm **cms_add()** chứa thao tác tạo giá trị hash sử dụng hàm băm, ở đây hàm băm là đơn vị nhỏ nhất vì bản thân hàm băm chỉ là một tập hợp các phép tính, tiếp tục chia nhỏ các phép tính này là không có ý nghĩa.

Bảng 2.1: Điểm nóng CPU trong MIDAS-R theo đơn vị tính toán

Đơn vị	**Mô tả**	**Tỉ lệ tiêu thụ CPU**
Hash	Tạo giá trị hash	**55.46**%
Heap	Thao tác truy cập và tính toán với Heap Memory	40.36%
Compte score	Tính điểm bất thường	0.19%
Khác	Những hoạt động còn lại như Kernel Call, thư viện,...	9.52%

Bảng 2.2: Điểm nóng CPU trong MIDAS-R theo hàm

Hàm	**Mô tả**	**Tỉ lệ tiêu thụ CPU**
cms_add()	Thêm một giá trị vào CMS	45.98%
cms_check()	Truy vấn một giá trị trong CMS	**47.05**%
CompteScore()	Tính điểm bất thường	0.19%
MultipleAll()	Nhân tất cả phần tử trong CMS với tham số $factor$	5.52%
Khác	Những hàm còn lại	4.62%

Nhận xét: Từ Bảng 2.1 và 2.2 có những đánh giá sau:

1. Chi phí cho việc tính toán điểm bất thường là không đáng kể, điều này là dễ hiểu vì nó chỉ là một hàm toán học thuần túy.
2. Việc tính toán tạo giá trị băm là rất tốn kém. Mặc dù hàm bằm được chọn rất hiệu quả nhưng các hàm **cms_add()** và **cms_check()** đều cần tạo giá trị băm dẫn đến gia tăng số lần băm.
3. Các thao tác với Heap tiêu tốn rất nhiều thời gian. Lý do là bộ nhớ Heap chậm hơn rất rất nhiều so với CPU[18].

Đề xuất: Qua các nhận xét trên, tôi đưa ra các đề xuất cải tiến sau:

A. (1) $\longrightarrow$ Chỉ cần trung cải thiện các cấu trúc trong khối SKETCH đề cập ở sơ đồ trong Hình 1.5.

B. (2) $\longrightarrow$ Sử dụng kỹ thuật Caching(phân bổ riêng một vùng nhớ để tái sử dụng giá trị băm nhiều lần).

C. (2)(3) $\longrightarrow$ Cần một thuật toán mới để giảm thiểu truy xuất Heap và sử dụng hàm băm.

D. (2) $\longrightarrow$ Tìm cách tạo nhiều giá trị băm với ít hàm băm hơn.

Ở các phần tiếp theo, tôi sẽ lần lượt đưa ra các đề xuất cải tiến dựa theo các phương án trên.

2.2 Bản phác thảo dựa trên NitroSketch

Phần này sẽ giới thiệu về bản phác thảo NitroSketch và cách áp dụng những ý tưởng của nó nhằm đáp ứng Đề xuất A và C.

2.2.1 Giới thiệu về NitroSketch

NitroSketch[4] là một khung phác thảo giải quyết một cách có hệ thống các tắc nghẽn về hiệu năng của các bản phác thảo mà không làm mất đi tính mạnh mẽ

và tính tổng quát. Nó được thiết kế và triển khai để giảm số lượng hoạt động của CPU và bộ nhớ trên mỗi gói. Điểm nổi bật của NitroSketch là nó có thể áp dụng vào nhiều bản phác thảo khác như CMS, Count Sketch[19], UnivMon[20]. Các thử nghiệm[4] cho thấy nó có thể đạt được độ chính xác tương đương các bản phác thảo gốc trong khi vượt trội về tốc độ trong khi giảm 40% tiêu thụ CPU.

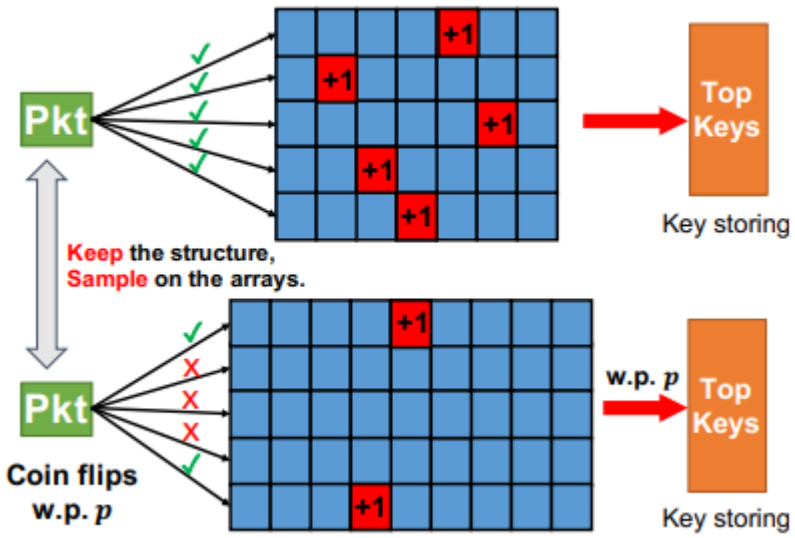
Ý tưởng chính của NitroSketch:

A. **Lấy mẫu mảng bộ đếm để giảm tắc nghẽn:** Như đã đề cập ở Mục 1.4.1, với mỗi gói tin, CMS tạo giá trị băm và cập nhật tất cả các mảng bộ đếm. Được minh họa trong Hình 2.1, ý tưởng là thay vì vậy, ta sẽ "tung đồng xu"xem cần cập nhật mảng bộ đếm nào cho gói tin với xác suất p và cập nhật bộ đếm thêm p^{-1}. Điều này dẫn đến giảm số lần tạo giá trị băm và thao tác Heap. Tuy nhiên, nhược điểm là cần "tung đồng xu"cho mỗi mảng bộ đếm.

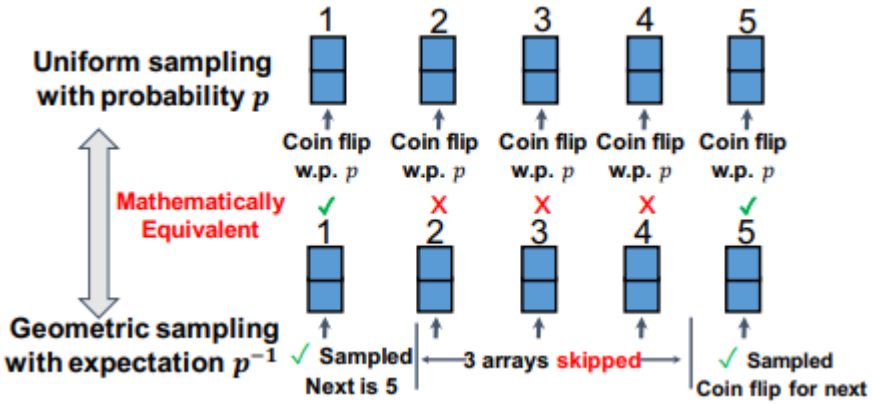
B. **Lấy mẫu sử dụng phân phối hình học để tránh chi phí "tung đồng xu":** Thay vì "tung đồng xu", NitroSketch sử dụng phân phối hình học[21] để chọn mảng bộ đếm nào sẽ được cập nhật. Ví dụ ở Hình 2.2, bản phác thảo có 5 mảng bộ đếm và giả sử rằng mảng 1 vừa được cập nhật. Ta sử dụng một biến hình học, với xác suất thành công p cho chúng ta biết có bao nhiêu mảng sẽ bỏ qua. Nếu mẫu cho biết cập nhật tiếp theo cách 4 mảng, ta bỏ qua 3 mảng sau đó và cập nhật mảng 5 (với cùng gói gói tin). Nếu phân phối yêu cầu bỏ qua 6 mảng, thì ta sẽ bỏ qua các cập nhật tiếp theo cho gói hiện tại và cập nhật mảng 3 ở gói tin tiếp theo.

C. **Lấy mẫu thích ứng dựa trên tốc độ gói tin đến để giảm thời gian hội tụ:** Có sự đánh đổi giữa thời gian hội tụ và tốc độ cập nhật gói tin. Càng nhiều gói bỏ qua, chúng ta càng phải đợi lâu hơn để đảm bảo độ chính xác. Khi sử dụng xác suất lấy mẫu mảng p cố định, chúng ta nên xác định p là đủ nhỏ cho tốc độ gói cao nhất có thể. Tuy nhiên, trong trường hợp đó, chúng tôi không cần thiết phải chịu thời gian hội tụ lâu vì hầu hết các khối lượng công việc đều có tốc độ gói trung bình thấp hơn và các đợt bùng nổ lưu lượng không thường xuyên. Khi mà tốc độ đến của gói thấp, không cần phải bỏ qua nhiều gói (để đạt được tốc độ này) và do đó chúng ta có thể tăng p để nhanh chóng đạt được hội tụ như trong Hình 2.3.

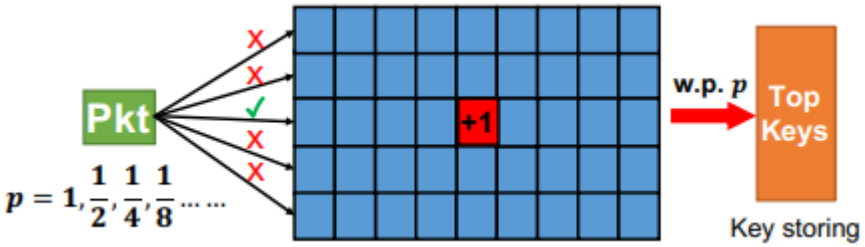
D. **Cập nhật bộ đệm và song song hóa tính toán với SIMD:** Vì các hoạt động đếm(hay phác thảo) là độc lập, sau khi áp dụng các ý tưởng đã đề cập ở trên, chúng ta có thể đệm các bộ đếm được lấy mẫu và các khóa đang chờ xử lý của chúng để tính toán các hàm băm và cập nhật bộ đếm trong chế độ Đa dữ liệu một lệnh (SIMD[22]), cụ thể là sử dụng Advanced Vector Extensions(AVX[23]). Chi tiết về kỹ thuật này sẽ được trình bày ở Mục 2.4.



Hình 2.1: Ý tưởng A, NitroSketch[4]



Hình 2.2: Ý tưởng B, NitroSketch[4]



Hình 2.3: Ý tưởng C, NitroSketch[4]

2.2.2 Áp dụng

Có thể thấy các ý tưởng của NitroSketch vừa có thể tăng tốc thực thi cho CMS, vừa mang lại khả năng thích ứng với lưu lượng mạng thay đổi. Tuy nhiên, nó chỉ phù hợp với luồng dữ liệu có kích thước lớn nên không thể áp dụng các cấu trúc *Current* SKETCH vì chúng chỉ dùng để đếm các gói tin trong cùng một thời điểm do đó kích thước luồng rất nhỏ. Vì vậy, trong sơ đồ trong Hình 1.5, ta sẽ chỉ tiến hành sửa đổi các cấu trúc *Total* SKETCH. Để cho ngắn gọn ta sẽ gọi cấu trúc dữ liệu mới này là NS.

Algorithm 3: Sketch based on NitroSketch

Data: Stream of packet over time

▷ **Init: with** d, w

$S[1,1] \dots S[d,w] \leftarrow 0$

▷ **Update:** for key x and probability p

while $i \leq d$ **do**

  if *is next row* **then**

    $h_i(x) \leftarrow H_i(x)$

    $S[i, h_i] \leftarrow S[i, h_i] + p^{-1}$

    $next \leftarrow next + Geo(p)$; `/* update next row to update */`

  else

    skip until next row(even packets)

  end

end

▷ **Query:** for x key

return $median_{i \in d}\{S_{i,h_i(x)}\}$

Như mô tả ở Thuật toán 3, giai đoạn Khởi tạo(Init) chỉ đơn giản là tạo một ma trận S với kích thước (d, w) sau đó đặt tất cả giá trị trong nó về 0. Giai đoạn Cập nhật(Update) sẽ chỉ tính giá trị băm h sử dụng hàm băm H cho mảng bộ đếm i với giá trị khóa x, sau đó cập nhật bộ đếm tại vị trí h trong S lên p^{-1} đơn vị rồi tính hàng bộ đếm tiếp theo cần cập nhật sử dụng phân phối hình học với xác suất p, tất cả các mảng còn lại được bỏ qua. Giai đoạn Truy vấn(Query) ta chỉ cần lấy trung vị của bộ đếm tại tất cả các mảng với d giá trị băm khác nhau.

2.3 Băm hiệu quả

Đề cập ở Tiểu mục 2.2.2, các ý tưởng của NitroSketch chỉ có thể áp dụng cho các cấu trúc *Total* SKETCH. Với mỗi gói tin đến, các cấu trúc *Current* SKETCH đều được cập nhật và truy vấn do đó chiếm một phần đáng kể thời gian chạy.

Như đã biết, khối SKETCH mỗi lần Cập nhật và Truy vấn đều cần tạo giá trị băm tương ứng với mỗi đầu vào cho tất cả d mảng bộ đếm Mục này trả lời cho

Đề xuất D bằng cách đưa ra một phương pháp tạo các giá trị băm hiệu hơn với Double hashing sau đó phân tích ưu nhược điểm của nó và tìm cách điều chỉnh cho phù hợp với bài toán.

2.3.1 Giới thiệu về Băm kép

Phương pháp Băm kép là một kỹ thuật tạo nhiều giá trị băm với chỉ với hai hàm băm([24], [25]). Nó đã được áp dụng rộng rãi trong triển khai Bộ lọc Bloom[26].

Để mô tả các kỹ thuật này, chúng tôi giả sử cần d giá trị băm $g_1, \ldots, g_d$, để xác định các chỉ số liên kết với một phần tử trong mảng có độ rộng w. Thông thường, đây là các giá trị băm sử dụng các hàm băm độc lập: $g_1(x) = h_1(x), \ldots, g_d(x) = h_d(x)$. Băm kép là một giải pháp thay thế gần như hiệu quả nhưng chỉ yêu cầu hai lần băm để tính toán một chuỗi các giá trị băm cho khóa x. Giá trị băm đầu tiên a được tính bằng cách sử dụng hàm bằm h_1. Giá trị băm thứ hai sử dụng h_2. Giờ có thể tính bất kỳ vị trí tiếp theo nào trong chuỗi:

$$\begin{aligned} a &= h_1(x) \\ b &= h_2(x) \\ g_i(x) &= (a + b \times i) \mod w \end{aligned} \tag{2.1}$$

Cách thực hiện này rất đơn giản. Tuy nhiên nó có một số vấn đề[27]: Có khả năng cho $b = h_2(x)$ có thể dẫn đến nhiều lần lặp lại cùng một giá trị trong quá trình tính toán. Rõ ràng là nếu $b = 0$, tất cả các giá trị đều giống nhau. Và nếu b và w có các thừa số chung thì cũng có thể xảy ra lặp lại trong tính toán d chỉ số. Ví dụ, xét $b = w/2$ thì các giá trị băm g_i sẽ lặp lại tuần hoàn. Ngoài ra còn 2 vấn đề nữa nhưng nó liên quan đến bộ lọc Bloom nên không được đề cập ở đây.

Vấn đề trên có thể được giải quyết như đề cập ở Mục 6.5.4 trong [28]. Thuật toán 4 mô tả cách nó hoạt động.Gần đây, chính các tác giả của thuật toán này đã đề xuất một phương pháp mới chỉ cần một hàm băm[29] trong khi vẫn đảm bảo số lượng xung đột chấp nhận được. Như cho thấy ở Thuật toán 5.

Algorithm 4: Enhanced double hashing

Data:

x : the element being added or queried

w : size of array

d : number of hash value to compute

h_1, h_2 : hash functions

Result: g[0 .. d - 1] : array of hash

1 $a \leftarrow h_1(x)$

2 $b \leftarrow h_2(x)$

3 $g[0] \leftarrow a$

4 **for** $i \leftarrow 1 \dots d-1$ **do**

5 $\quad a \leftarrow (a+b) \mod w$

6 $\quad b \leftarrow (b+i) \mod w$

7 $\quad g[i] \leftarrow a$

8 **end**

Algorithm 5: New enhanced double hashing

Data:

x : the element being added or queried

w : size of array

d : number of hash value to compute

H : hash function

Result: g[0 .. d - 1] : array of hash

1 $h \leftarrow H(x)$

2 $delta = (h >> 17) \vee (h << 47)$

3 **for** $i \leftarrow 0 \dots d$ **do**

4 $\quad delta \leftarrow delta + i$

5 $\quad g[i] \leftarrow h$

6 $\quad h \leftarrow h + delta$

7 **end**

2.3.2 Áp dụng

Thuật toán 5 có thể áp dụng với CMS(hay các cấu trúc *Current* trong khối SKETCH) như Triển khai 2. Hàm **new_hashes()** trả về **depth** giá trị băm với khóa **key**. Ở đây, ta tạo giá trị băm **hash** bằng hàm băm xxHash **XXH64()** với giá trị khởi tạo được chọn bất kỳ là 13.

```
uint64_t *new_hashes(uint64_t *hashes, const char *key, int depth) {
  uint64_t hash = XXH64(key, strlen(key), 13);
  uint64_t delta = (hash >> 17) | (hash << 47);
  for (int i = 0; i < depth; i++) {
    delta += i;
    hashes[i] = hash;
    hash += delta;
  }
  return hashes;
}
```

Listing 2: Tạo nhiều giá trị băm cho CMS

Đối với NS, ta có thể chỉ đơn giản là tạo các giá trị băm liên tiếp như trong Triển khai 3 vì như đã nêu ở Mục 2.2, giai đoạn Cập nhật của nó chỉ tăng duy nhất bộ đếm mỗi lần, đặc điểm này khiến cho việc chỉ cần tạo các giá trị băm liên tiếp mà không cần lo lắng làm gia tăng xung đột.

```
uint64_t *new_hashes(uint64_t *hashes, const char *key, int depth) {
  uint64_t hash = XXH64(key, strlen(key), 13);
  for (int i = 0; i < depth; i++) {
    hashes[i] = hash + i;
  }
  return hashes;
}
```

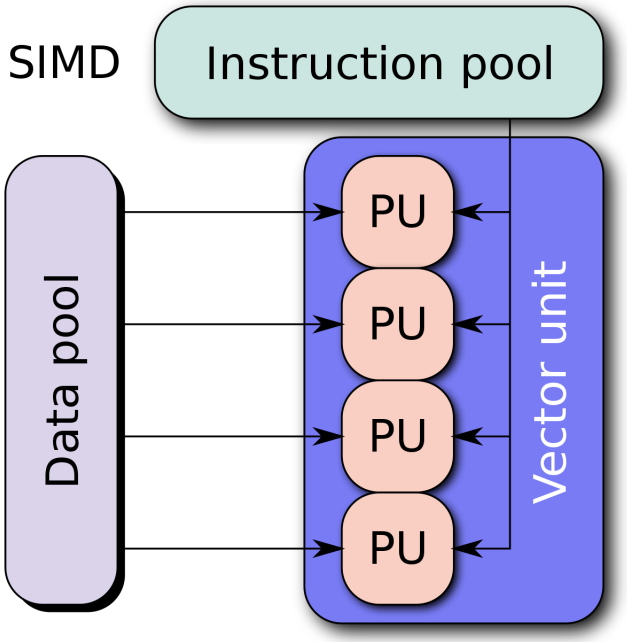
Listing 3: Tạo nhiều giá trị băm cho NS

2.4 SIMD - Single Instruction/Multiple Data

2.4.1 Giới thiệu về SIMD

SIMD (đơn hướng dẫn nhiều dữ liệu) là một kỹ thuật tính toán song song theo đó một CPU (hoặc xử lý phần tử được tích hợp trong CPU) thực hiện một thao tác đơn lẻ bằng cách sử dụng đồng thời nhiều mục dữ liệu. Ví dụ: CPU có khả năng SIMD có thể thực hiện một phép toán số học đơn lẻ bằng cách sử dụng một số phần tử của một mảng dấu phẩy động đồng thời. Hoạt động SIMD thường được sử dụng để tăng tốc hiệu năng của các thuật toán và chức năng cường độ cao tính toán trong học máy, xử lý hình ảnh, mã hóa và giải mã âm thanh/video, khai thác dữ liệu và đồ họa máy tính. Như trong Hình 2.4, ta có thể tải đồng thời nhiều dữ liệu vào một đơn vị Vector và tính toán với chúng một cách đồng thời.

Vào năm 2011, Intel và AMD đã giới thiệu bộ vi xử lý hỗ trợ công nghệ SIMD x86 mới có tên là Phần mở rộng Vector nâng cao(AVX[23]). AVX thêm các hoạt động dấu phẩy động được đóng gói sử dụng thanh ghi rộng 256-bit. AVX cũng hỗ trợ cú pháp hướng dẫn hợp ngữ ba toán hạng mới, giúp giảm số lần truyền dữ liệu từ thanh ghi này sang thanh ghi khác mà một chức năng phần mềm phải thực hiện. Bắt đầu từ năm 2017, bộ xử lý hướng đến máy chủ và máy tính để bàn cao cấp do Intel tiếp thị bao gồm một bộ vi xử lý mới. Phần mở rộng SIMD có tên là AVX-512. Cải tiến kiến trúc này hỗ trợ số nguyên đóng gói và dấu phẩy động hoạt động bằng cách sử dụng thanh ghi rộng 512-bit.



Hình 2.4: Minh họa hoạt động của SIMD[5]

2.4.2 Áp dụng

Trong trường hợp này, ta có thể áp dụng AVX cho thao tác *MultipleAll* trong Sơ đồ 1.5, nhắc lại rằng nhiệm vụ của nó là nhân tất cả giá trị trong cấu trúc CMS với trọng số *factor*. Cụ thể hàm **multipleAllAVX()** được mô tả trong triển khai 4, đầu tiên ta khởi tạo một vector **by256** với các giá trị **by**(đây chính là trọng số *factor*), tiếp đến là duyệt qua toàn bộ phần tử trong CMS, mỗi lần lặp ta sẽ tải 4 giá trị ra, nhân chúng với *factor* rồi tải trở lại CMS. So với triển khai ban đầu **multipleAll()**, cách làm này sẽ cho hiệu năng cao hơn vì xử lý cùng lúc 4 giá trị tại mỗi lần lặp.

```
void multipleAll(CountMinSketch *cms, double by) {
  for (int i = 0; i < cms->depth; i++) {
    for (int j = 0; j < cms->width; j++) {
      cms->bins[i * cms->width + j] *= by;
    }
  }
}

void multipleAllAVX(CountMinSketch *cms, double by) {
  __m256d by256 = _mm256_set1_pd(by);
  for (int i = 0; i < cms->depth; i++) {
    for (int j = 0; j < cms->width; j += 4) {
      __m256d bins256 = _mm256_loadu_pd(cms->bins + i * cms->width + j);
      bins256 = _mm256_mul_pd(bins256, by256);
      _mm256_storeu_pd(cms->bins + i * cms->width + j, bins256);
    }
  }
}
```

Listing 4: Tạo nhiều giá trị băm cho NS

2.5 Các phương pháp khác

Ngoài các cách kể trên, áp dụng những phương pháp sau đây cũng làm tăng đáng kể hiệu năng tổng thể của thuật toán MIDAS-R:

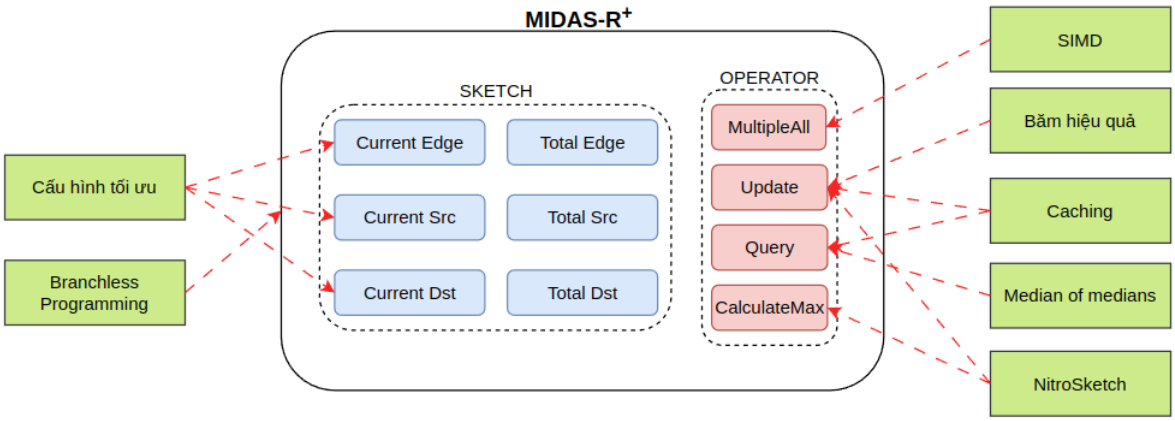
- Caching: Trong triển khai 1, có thể thấy thông tin mỗi gói tin đều được cập nhật và truy vấn rồi tính điểm bất thường. Ta có thể khai thác đặc điểm này bằng chỉ tính toán các giá trị băm ở bước cập nhật và lưu lại, sau đó tại bước truy vấn có thể sử dụng lại chúng. Cách làm này sẽ giảm thiểu số lần băm và chi phí phân bổ và giải phóng bộ nhớ, đáp ứng đề xuất B.
- Median of medians[30]: Là một thuật toán lựa chọn thời gian tuyến tính xác định. Nó hoạt động bằng cách chia danh sách thành các danh sách con rồi xác định giá trị trung vị gần đúng trong mỗi danh sách con. Sau đó, nó lấy các trung vị đó và đặt chúng vào một mảng và tìm trung vị của danh sách đó. Nó sử dụng giá trị trung bình đó làm trục và so sánh các phần tử khác của

mảng với trục. Nếu một phần tử nhỏ hơn giá trị trục, thì phần tử đó được đặt ở bên trái của trục và nếu phần tử có giá trị lớn hơn trục, thì phần tử đó sẽ được đặt ở bên phải. Thuật toán lặp lại trong mảng, tiến đến giá trị cần tìm. Thuật toán này có độ phức tạp thuật toán tốt hơn so với cách sắp xếp mảng rồi tìm giá trị chính giữa nên sẽ được sử dụng để tính trung vị cho NS.

- Cấu hình tối ưu cho khối SKETCH: Triển khai của các tác giả MIDAS-R sử dụng cùng kích thước cho tất cả cấu trúc trong khối SKETCH. Điều này làm lãng phí bộ nhớ và thời gian xử lý vì các cấu trúc *Current* chỉ cần xử lý số lượng dữ liệu nhỏ hơn rất nhiều so với *Total* . Do đó, cần định cấu hình riêng cho mỗi nhóm *Current* và *Total*.

- Branchless Programming: Là kỹ thuật lập trình giảm thiểu rẽ nhánh chương trình. Khi CPU gặp phải một bước nhảy có điều kiện hoặc bất kỳ kiểu phân nhánh nào khác, nó không chỉ ngồi yên cho đến khi điều kiện của nó được tính toán. Thay vào đó, nó bắt đầu thực thi một cách suy đoán nhánh có vẻ như sẽ được thực hiện ngay lập tức. Trong quá trình thực thi, CPU tính toán số liệu thống kê về các nhánh được thực hiện trên mỗi lệnh và sau một thời gian, chúng bắt đầu dự đoán chúng bằng cách nhận dạng các mẫu chung[31], điều này là tiên quyết để CPU có thể thực thi đường ống(Pipleline). Vì lý do này, chi phí thực sự của một nhánh phần lớn phụ thuộc vào mức độ dự đoán của CPU. Nếu đó là trò tung đồng xu 50/50 thuần túy, ta phải chịu rủi ro kiểm soát và loại bỏ toàn bộ đường ống, mất thêm 15-20 chu kỳ để xây dựng lại. Và nếu nhánh luôn luôn hoặc không bao giờ được thực hiện, ta hầu như không phải đánh đổi gì ngoài việc kiểm tra điều kiện. Như chúng ta đã thiết lập trong phần trước, các nhánh mà CPU không thể dự đoán một cách hiệu quả sẽ rất tốn kém vì chúng có thể gây ra tình trạng ngừng trệ đường ống dài để tìm nạp các lệnh mới sau khi một nhánh dự đoán sai. Kỹ thuật này sẽ được áp dụng toàn bộ thuật toán MIDAS-R.

2.6 Kết hợp các đề xuất



Hình 2.5: Kết hợp tất cả đề xuất

Đến đây, ta sẽ áp dụng tất cả các đề xuất cải tiến vào MIDAS-R và gọi thuật toán mới là MIDAS-R$^+$. Hình 2.5 mô tả thuật toán MIDAS-R$^+$ được tạo thành từ MIDAS-R với các thành phần được sửa đổi tương ứng với các đề xuất nêu trên. Việc áp dụng các đề xuất này không làm thay đổi nền tảng lý thuyết của MIDAS-R vì:

- Các kỹ thuật lập trình như SIMD, Branchless Programming, Caching không làm thay đổi hành vi của thuật toán.
- Lựa chọn cấu hình tối ưu chỉ thay đổi kích thước các cấu trúc SKETCH nên chỉ ảnh hưởng đến độ chính xác của thuật toán.
- Ý tưởng về Băm hiểu quả và NitroSketch vẫn đảm bảo đầu vào cho bước tính điểm bất thường hoàn toàn tương đương thuật toán ban đầu([4],[29]).

Chương 3

KIỂM THỬ VÀ ĐÁNH GIÁ

3.1 Chuẩn bị

3.1.1 Tập dữ liệu

Có 6 tập dữ liệu được sử dụng là CIC-DDOS2019[15], CIC-IDS2018[32], CTU-13[33], DARPA[34], ISCX-IDX2012[35] và UNSW-NB15[36] đều được cung cấp bởi các tổ chức an ninh mạng và các nhà nghiên cứu. Bảng 3.1 liệt kê kích thước hay số lượng gói tin trong các tập dữ liệu. Dữ liệu được tổ chức thành các tệp như sau:

- *Meta.txt* chứa kích thước hay tổng số gói tin.
- *Data.csv* chứa thông tin các gói tin gồm IP nguồn, IP đích và Timestamp.
- *Label.csv* chứa nhãn các gói tin, cho biết gói tin nào là gói tin tấn công.

Ngoài ra, để mô phỏng lưu lượng mạng phục vụ cho kiểm nghiệm thuật toán MIDAS-R$^+$ mới, tôi đã tạo một tệp *Density.csv*. Tệp này chứa số lượng gói tin trong cùng một thời điểm.

3.1.2 Cấu hình

Thông số kỹ thuật của phần cứng và môi trường như bảng 3.2

Bảng 3.1: Số lượng gói tin trong tập dữ liệu

Dataset	Size
CIC-DDOS2019	20Mp
CIC-IDS2018	7.9Mp
CTU-13	2.5Mp
UNSW-NB15	2.5Mp
DARPA	4.5Mp
ISCX-IDX2012	1.1Mp

Bảng 3.2: Cấu hình

Processor	AMD Ryzen 5 5500U@2.1GHz
RAM	8GB
Platform	Xubuntu 22.04.2 LTS x86_64

Lưu ý: Tiểu mục 2.5 đã đưa ra đề xuất đặt kích thước cho khối SKETCH một cách riêng biệt để tránh lãng phí tài nguyên không cần thiết. Hiển nhiên nếu ta đặt các thông số cho MIDAS-R$^+$ theo đề xuất này còn trên MIDAS-R vẫn dùng chung thông số cho khối SKETCH thì tốc độ của thuật toán mới sẽ nhanh hơn thuật toán ban đầu theo cấp số nhân khi tăng dần các thông số. Do đó, không cần thiết phải thử nghiệm đề xuất này.

Thuật toán MIDAS-R và MIDAS-R$^+$ được đặt cùng một thông số như sau:

- *Current* SKETCH: $d = 4$, $w = 1024$
- *Total* SKETCH: $d = 8$, $w = 8092$

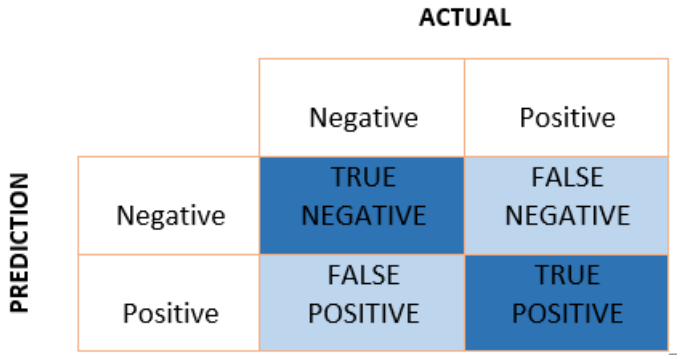
Riêng với MIDAS-R$^+$, xác suất p được cấu hình như Bảng 3.3. Việc lựa chọn các thông số này cần có sự phân tích lưu lượng, tùy thuộc vào tình huống mà đưa ra cài đặt hợp lý.

Bảng 3.3: Thích ứng lưu lượng mạng

Packets per time unit(x)	p
$x < 5$	1
$5 \geq x < 10$	2^{-2}
$10 \geq x < 20$	2^{-3}
$x \geq 20$	2^{-4}

3.1.3 Phương pháp đo độ chính xác

Ma trận nhầm lẫn: Giống như tên gọi, nó cho ta một biểu diễn kết quả nhận dạng của thuật toán. Ma trận nhầm lẫn cho ta cái nhìn đầy đủ nhất về dự đoán của thuật toán với dữ liệu kiểm định. Hình 3.1 thể hiện cấu trúc của ma trận nhầm lẫn đối với bài toán nhị phân (gồm 2 lớp Positve và Negative). Với giá trị mỗi hàng là nhãn đúng của mẫu, cột là kết quả dự đoán. Giá trị True Positive(TP) và True Negative(TN) thể hiện dự đoán đúng của mô hình. False Negative(FN) và False Positve(FP) thể hiện dự đoán sai. Ma trận nhầm lẫn sẽ cho ta biết mô hình đang bị nhầm lẫn giữa những lớp nào trong bộ dữ liệu kiểm thử.



Hình 3.1: Ma trận nhầm lẫn[6]

Độ chính xác: Người ta thường sử dụng đường cong ROC để đo lường độ chính xác. Một đường cong ROC được xây dựng bằng cách vẽ biểu đồ tỷ lệ dương thực (TPR) so với tỷ lệ dương giả (FPR) ở các cài đặt ngưỡng khác nhau. Đường cong càng gần góc trên bên trái của biểu đồ, thuật toán càng phân loại dữ liệu tốt hơn. Trong đó:

TPR(True Positive Rate): Tỷ lệ dương tính thực sự là tỷ lệ dương tính thực tế được phân loại xác định chính xác

$$TPR = \frac{TP}{TP + FN} \tag{3.1}$$

FPR(Fasle Positive Rate): tỷ lệ dương tính giả là tỷ lệ âm tính thực tế được xác định không chính xác là dương tính

$$FPR = \frac{FP}{FP + TP} \tag{3.2}$$

AUC là viết tắt của "Area Under the ROC Curve", bản chất là diện tích bên dưới đường cong ROC, đại diện cho mức độ hoặc thước đo khả năng phân tách. Nó cho biết thuật toán có khả năng phân biệt bao nhiêu giữa các lớp. AUC càng cao, hiệu năng thuật toán càng tốt trong việc phân biệt giữa các nhóm dương tính và âm tính.

Ở đây, để tính toán ROC-AUC, cần lấy dữ liệu đã được gán nhãn trong các tập dữ liệu và so sánh nó với điểm bất thường mà MIDAS-R/MIDAS-R$^+$ sinh ra. Do điểm bất thường từ 0 đến giá trị bất kỳ nên ta cần chuẩn hóa để tính ROC-AUC. Thuật toán MIDAS-R$^+$ được chạy 20 lần nhằm thu được kết quả chính xác nhất vì có yếu tố ngẫu nhiên trong đó. Trong khi MIDAS-R chỉ cần chạy một lần do độ chính xác mỗi lần chạy là như nhau.

3.1.4 Phương pháp đo hiệu năng

Bằng cách tính thời gian tiêu tốn của thuật toán khi xử lý tập dữ liệu. Thời điểm bắt đầu đo từ lúc xứ lý gói tin đầu tiên và thời điểm kết thúc là gói tin cuối cùng, bỏ qua giai đoạn khởi tạo. Các thuật toán được thực thi 20 lần và lấy giá trị trung bình của các thời gian chạy thu được nhằm tránh sai số.

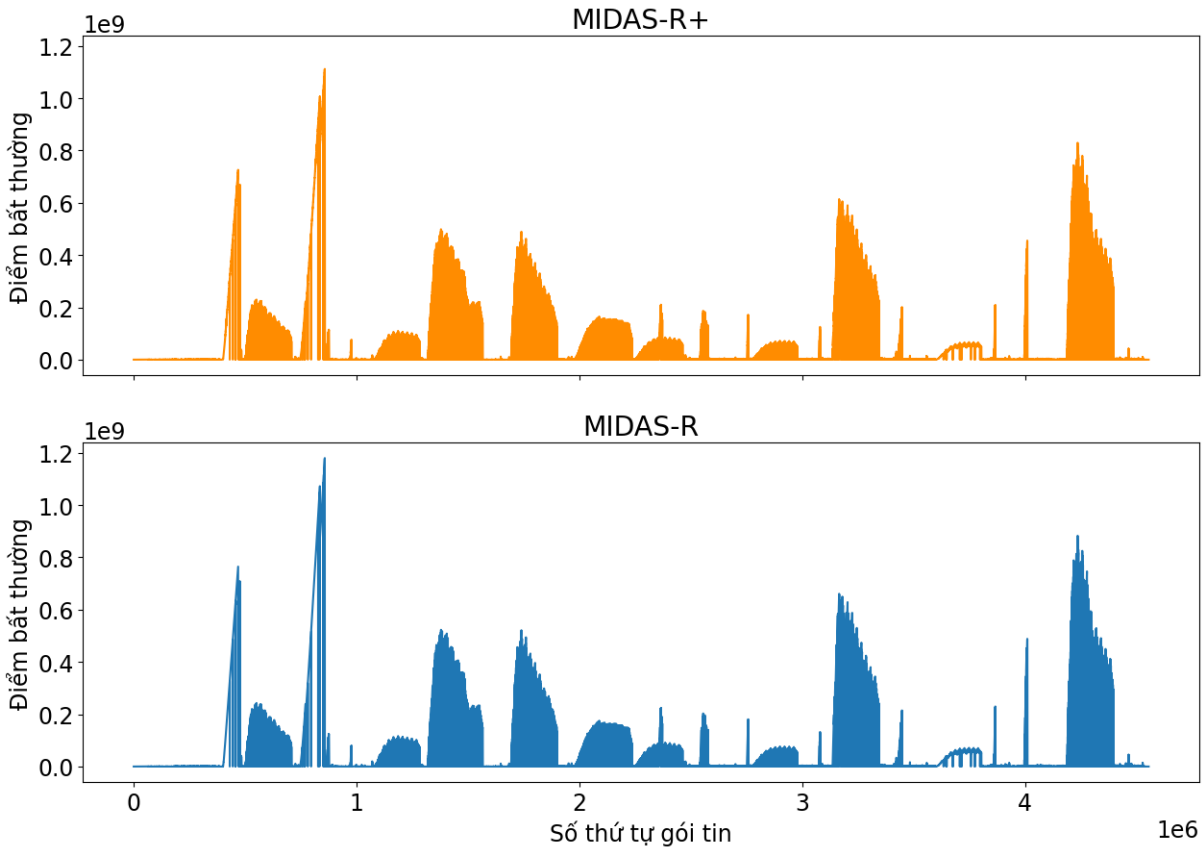
3.2 Kết quả và đánh giá

Ta sẽ tiến hành đo độ chính xác theo AUC-ROC. Đường cong ROC được trực quan ở Hình 3.2, Bảng 3.4 hiển thị giá trị AUC tương ứng với các trọng số $factor$ khác nhau và đều sử dụng DARPA vì đây là tập dữ liệu chứa đa dạng các cuộc tấn công và sát với thực tế nhất trong các tập dữ liệu được sử dụng. Hình 3.3 cho biết AUC giữa các tập dữ liệu. Cả ba đo lường trên đều được tiến hành để theo dõi độ chính xác của MIDAS-R so với MIDAS-R$^+$. Hình 3.4 mô tả lượng thời gian cần thiết để xử lý hết các tập dữ liệu khác nhau. Độ lệch chuẩn thu được là không đáng kể($< 0.5\%$) nên không cần thiết trực quan hóa trong biểu đồ.

Từ các kết quả này, ta có một số nhận xét sau:

- Như trong Hình 3.1, cuộc tấn công xảy ra ở những gói tin có điểm bất thường cao đột ngột, chính là những nơi tạo thành tháp nhọn. Cả 2 thuật toán đều có khả năng phát hiện các vị trí bất thường giống nhau.
- Độ chính xác của MIDAS-R và MIDAS-R$^+$ là tương đương nhau với các trọng số $factor$. Đáng chú ý, MIDAS-R$^+$ thậm chí còn chính xác hơn MIDAS-R một chút như trong Bảng 3.4. Điều này có thể do khả năng thích nghi với lưu lượng mạng của MIDAS-R$^+$ đã dẫn đến bỏ qua các gói tin gây nhiễu.
- Đường cong ROC trong Hình 3.2 gần như khớp hoàn toàn giữa MIDAS-R và MIDAS-R$^+$ cho biết khả năng phân loại bất thường của hai thuật toán là giống nhau.
- AUC của hai thuật toán trên các tập dữ liệu khác nhau không có khác biệt đáng kể được biểu diễn trong Hình 3.3.

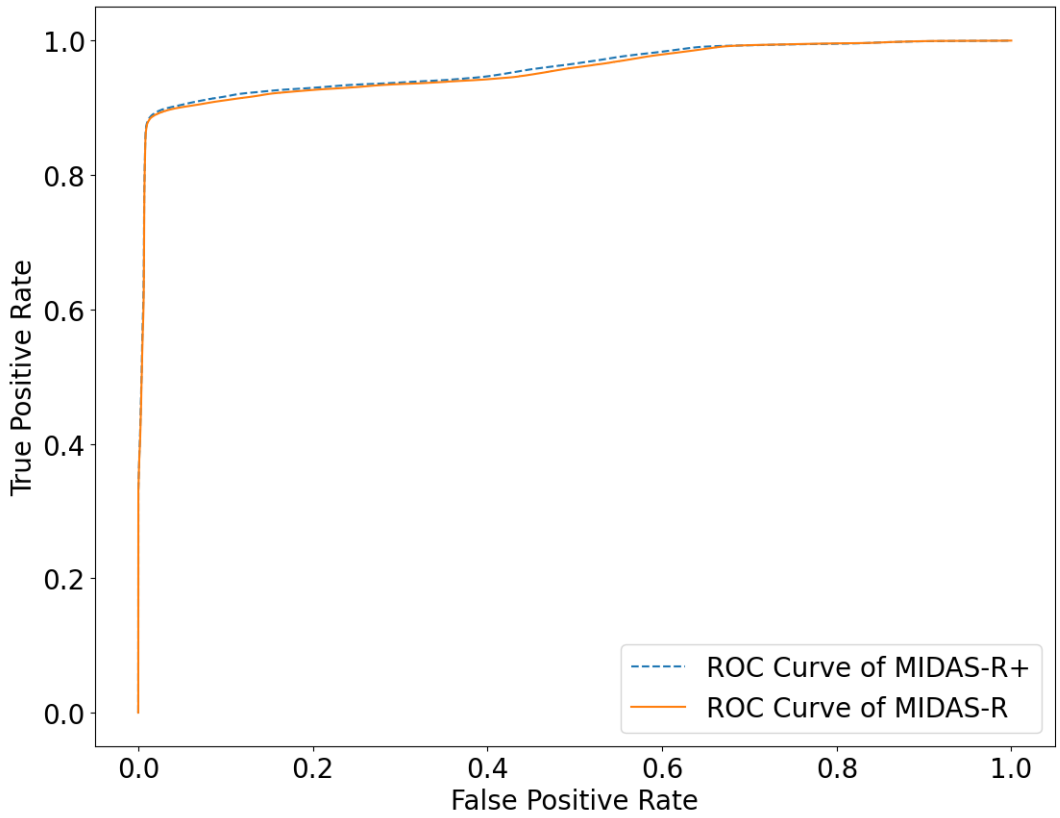
- Từ các nhận xét trên có thể khẳng định độ chính xác của thuật toán vẫn giữ nguyên sau khi cải tiến MIDAS-R thành MIDAS-R$^+$.
- Thời gian thực thi của MIDAS-R$^+$ nhanh hơn MIDAS-R từ 49% đến 55% như cho thấy trong Hình 3.4.



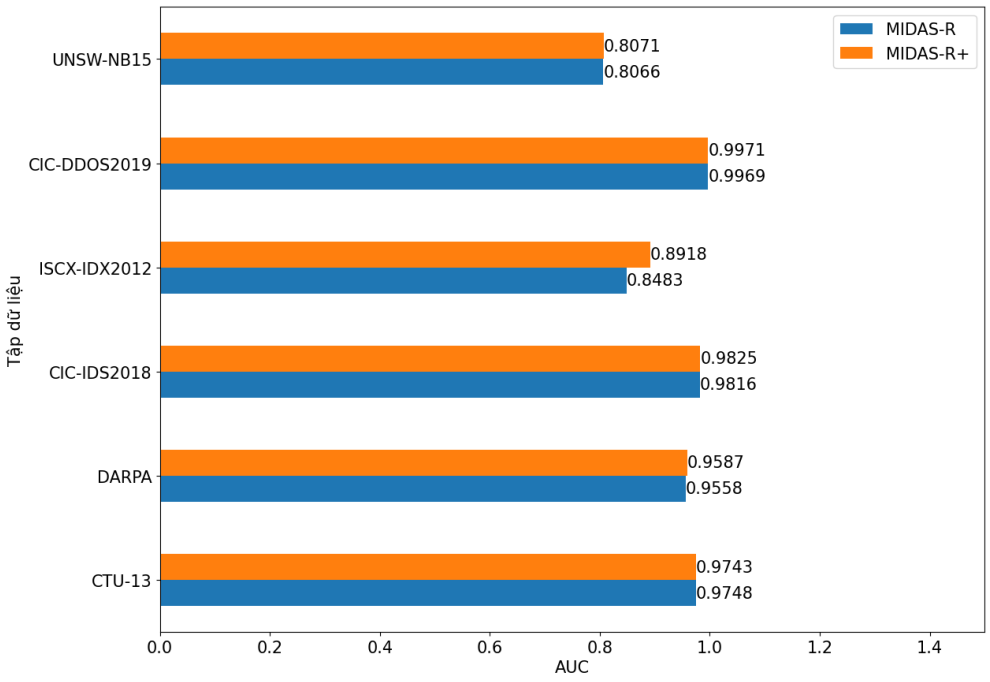
Hình 3.1: Điểm bất thường

Bảng 3.4: Ảnh hưởng của trọng số *factor* đến ROC-AUC của MIDAS-R và MIDAS-R$^+$

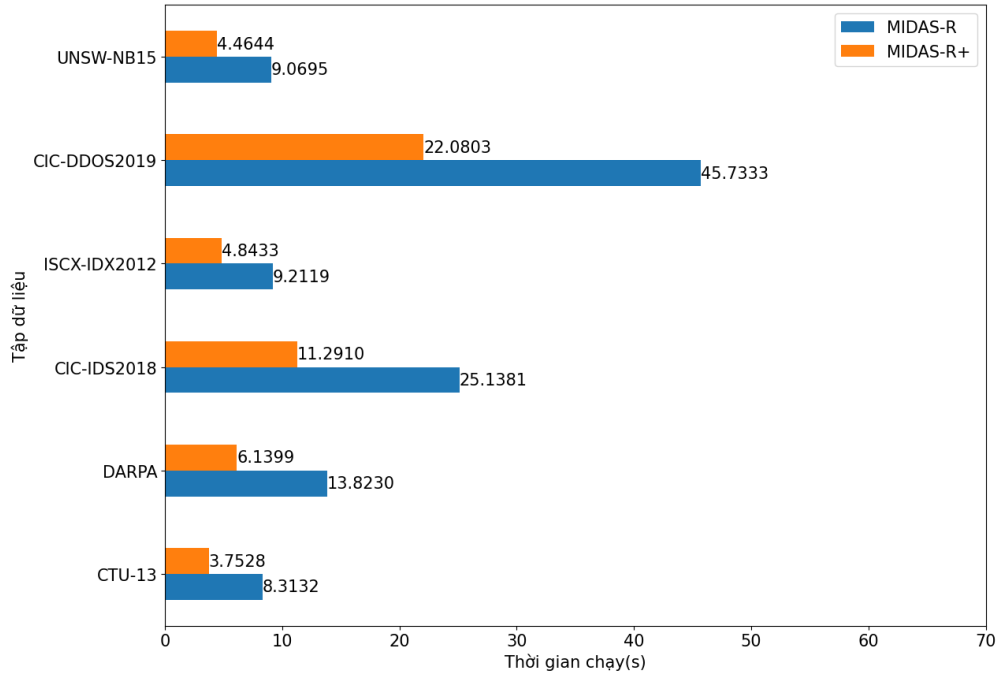
factor	MIDAS-R	MIDAS-R$^+$
0.1	0.8981	0.9106
0.2	0.9150	0.9241
0.3	0.9262	0.9325
0.4	0.9337	0.9381
0.5	0.9385	0.9420
0.6	0.9416	0.9448
0.7	0.9443	0.9475
0.8	0.9482	0.9515
0.9	0.9557	0.9587



Hình 3.2: Đường cong ROC của MIDAS-R và MIDAS-R$^+$ trên với tập dữ liệu DARPA, $factor = 0.9$



Hình 3.3: So sánh AUC giữa MIDAS-R và MIDAS-R$^+$ trên các tập dữ liệu, $factor = 0.9$



Hình 3.4: Thời gian chạy của MIDAS-R và MIDAS-R$^+$ trên các tập dữ liệu, $factor = 0.9$

KẾT LUẬN

Đồ án đã đề xuất các cải tiến cho thuật toán phát hiện bất thường MIDAS-R kết hợp nhiều phương pháp từ góc độ phần cứng, kỹ thuật lập trình lẫn độ phức tạp thuật toán. Được kiểm nghiệm về độ chính xác và hiệu năng với nhiều tập dữ liệu tấn công khác nhau và cho thấy thời gian thực thi nhanh hơn 49% đến 55% và có độ chính xác không đổi so với thuật toán ban đầu, do đó đã đạt được mục tiêu ban đầu. Tuy nhiên, vẫn còn một số hạn chế về lựa chọn tham số tối ưu cho thuật toán, các tham số được chọn dựa trên suy luận tương đối, chưa có cơ sở lý thuyết chặt chẽ cho vấn đề này. Từ đó làm tiền đề để nghiên cứu cải thiện thuật toán hơn nữa.

Hướng phát triển tiếp theo sẽ là thực hiện một cuộc thống kê toàn diện trên các cuộc tấn công mạng và đưa ra kích thước hợp lý cho cấu trúc dữ liệu SKETCH, phát triển một nền tảng toán học vững chắc về giới hạn lỗi trên lý thuyết với các kích thước này. Ngoài ra, cần thực hiện kiểm tra hiệu quả của thuật toán trong một trường mạng thực tế thay vì chỉ sử dụng các tập dữ liệu mô phỏng, điều này là rất cần thiết cho việc ứng dụng thuật toán vào thực tế.

Tài liệu tham khảo

[1] Ikatsov, "Probabilistic data structures for web analytics and data mining." https://highlyscalable.wordpress.com/2012/05/01/probabilistic-structures-web-analytics-data-mining/, 2012.

[2] R. L. Siddharth Bhatia, "Anomaly detection on dynamic (time-evolving) graphs in real-time and streaming manner. detecting intrusions (dos and ddos attacks), frauds, fake rating anomalies.." https://github.com/Stream-AD/MIDAS, 2022.

[3] M. Y. Siddharth Bhatia, Bryan Hooi, "Midas: Microcluster-based detector of anomalies in edge streams," *AAAI Conference on Artificial Intelligence*, 2020.

[4] G. E. Zaoxing Liu, Ran Ben-Basat, "Nitrosketch: Robust and general sketch-based monitoring in software switches," *ACM Special Interest Group*, pp. 334–350, 2019.

[5] Wikipedia, "Single instruction, multiple data." https://en.wikipedia.org/wiki/Single_instruction,_multiple_data, 2023.

[6] U. Blogs, "Assessing the effectiveness of classification models in machine learning." https://blog.unicloud.com.vn/ml/ai_performance_metrics/, 2022.

[7] C. Griffiths, "The latest 2023 cyber crime statistics (updated march 2023)," tech. rep., AAG, 2023.

[8] P. V. Ansam Khraisat, Igbal Gondal, "An anomaly intrusion detection system using c5 decision tree classifier. in: Trends and applications in knowledge discovery and data mining," *Trends and Applications in Knowledge Discovery and Data Mining*, vol. 154, pp. 149–155, 2018.

[9] R. S. Ismail Butun, Salvatore D. Morgera, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 266 – 282, 2014.

[10] S. M. Graham Cormode, "An improved data stream summary:the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, pp. 58–75, 2005.

[11] S. Graham Cormode, "Approximating data with the count-min data structure," *IEEE Software*, vol. 29, pp. 266 – 282, 2012.

[12] S. S. Dimitrije Jankov, "Real-time high performance anomaly detection over data streams: Grand challenge," *the 11th ACM International Conference*, pp. 3–6, 2017.

[13] M. L. Audrey Wilmet, Tiphaine Viard, "Degree-based outliers detection within ip traffic modelled as a link stream," *Network Traffic Measurement and Analysis Conference(TMA)*, pp. 2–11, 2018.

[14] B. H. Siddharth Bhatia, "Real-time anomaly detection in edge streams," *ACM Transactions on Knowledge Discovery from Data*, vol. 16, no. 4, pp. 2–11, 2022.

[15] S. H. I. Sharafaldin, A. H. Lashkari, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019.

[16] Cyan4973, "Extremely fast non-cryptographic hash algorithm." https://github.com/Cyan4973/xxHash, 2014.

[17] Y. Collet, "xxhash benchmark and applications." https://cyan4973.github.io/xxHash/, 2020.

[18] C. Carvalho, "The gap between processor and memory speeds," *Departamento de Informática, Universidade do Minho*, 2002.

[19] K. C. Moses Charikar, "Finding frequent items in data streams," *ICALP*, pp. 693–703, 2002.

[20] A. M. Zaoxing Liu, "One sketch to rule them all: Rethinking network flow monitoring with univmon," *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 101–114, 2016.

[21] Wikipedia, "Geometric distribution." https://en.wikipedia.org/wiki/Geometric_distribution, 2023.

[22] J. G. F. C. João M.P. Cardoso, "High-performance embedded computing," *Embedded Computing for High Performance*, vol. 2, 2017.

[23] Intel, "Intel advanced vector extensions." https://software.intel.com/en-us/isa-extensions/intel-avx, 2012.

[24] D. E. Knuth, *The Art of Computer Programming.* Addison-Wesley, 1997.

[25] G. H. Gonnet, *Handbook of Algorithms and Data Structures.* Addison-Wesley, 1984.

[26] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, p. 422–426, 1970.

[27] P. S. Almeida, "A case for partitioned bloom filters," *IEEE Transactions on Computers*, pp. 1–11, 2022.

[28] P. C. Dillinger, "Adaptive approximate state storage." http://peterd.org/pcd-diss.pdf, 2010.

[29] P. C. Dillinger, "Bloom filters using a flawed double hashing scheme." https://github.com/facebook/rocksdb/issues/4120, 2021.

[30] V. P. Manuel Blum, Robert W. Floyd, "Time bounds for selection," *Journal of Computer and System Sciences*, vol. 7, pp. 448–461, 1973.

[31] Danluu, "Branch prediction." https://danluu.com/branch-prediction/, 2017.

[32] A. H. Sharafaldin, Lashkari, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *4th International Conference on Information Systems Security and Privacy*, pp. 108–116, 2018.

[33] S. G. Garc´ıa, "An empirical comparison of botnet detection methods," *Comput. Secur*, vol. 45, pp. 100–123, 2014.

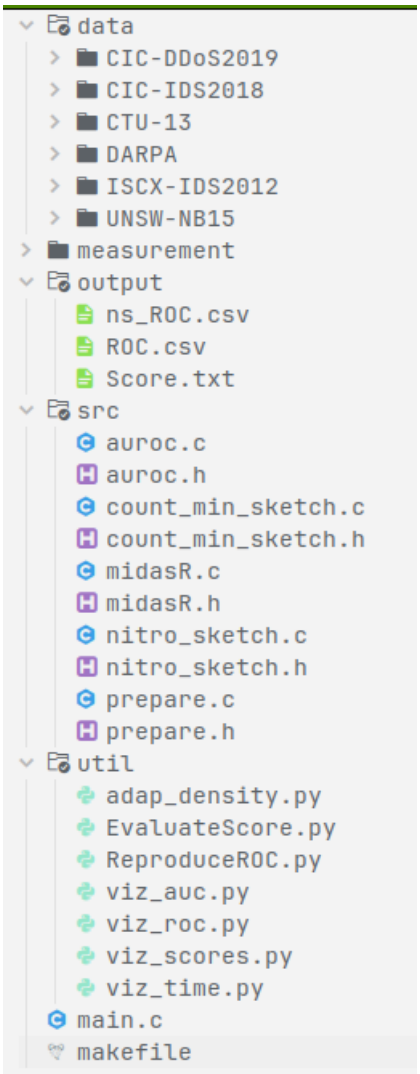
[34] R. C. Lippmann, "Results of the darpa 1998 offline intrusion detection evaluation," *Recent Advances in Intrusion Detection, Second International Workshop*, 1999.

[35] Shiravi, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur*, vol. 31, pp. 357–374, 2012.

[36] Moustafa, "Unsw-nb15: a comprehensive data set for network intrusion detection systems(unsw-nb15 network data set)," *MilCIS 2015*, pp. 1–6, 2015.

Triển khai chi tiết

Toàn bộ mã nguồn của đồ án được lưu trữ tại https://github.com/Lucifer-02/my_midas. Hình 3.5 dưới đây mô tả cách tổ chức của mã nguồn.



Hình 3.5: Cây thư mục của đồ án

- Thư mục *src*: Chứa các thư viện sẽ được sử dụng
  - Tệp *auroc.**: Chuẩn hóa điểm bất thường và tính toán giá trị AUC
  - Tệp *count-min-sketch.**: Bản phác thảo CMS và các thao tác với nó như truy vấn, cập nhật, khởi tạo...
  - Tệp *midasR.**: Các thành phần của MIDAS-R/MIDAS-R$^+$ và các hàm khởi tạo, tính điểm bất thường...
  - Tệp *nitro_sketch.**: Bản phác thảo dựa trên NitroSketch
  - Tệp *prepare.**: Đọc dữ liệu từ các tập dữ liệu
- Thư mục *output*: Kết quả sau khi chạy MIDAS-R và MIDAS-R$^+$ sẽ được ghi vào đây
  - Tệp *Score.txt*: Chứa điểm bất thường của tập dữ liệu tấn công
  - Tệp *ROC_plus.csv*: Chứa đường cong ROC sinh ra từ việc chạy MIDAS-R$^+$
  - Tệp *ROC.csv*: Chứa đường cong ROC sinh ra từ việc chạy MIDAS-R
- Thư mục *data*: Chứa các tập dữ liệu tấn công đã đề cập sở Mục 3.1.
- Thư mục *util*:
  - Tệp *adap_density.py*: Xử lý Timestamp trong các tập dữ liệu để mô phỏng khả năng thích ứng lưu lượng của MIDAS-R$^+$
  - Tệp *EvaluateScore.py*: Tính toán giá trị AUC cho các điểm bất thường
  - Tệp *ReproduceROC.py*: Tạo đường cong ROC từ điểm bất thường
  - Tệp *viz_auc.py*: Trực quan hóa giá trị AUC của MIDAS-R và MIDAS-R$^+$
  - Tệp *viz_roc.py*: Trực quan hóa đường cong ROC
  - Tệp *viz_time.py*: Trực quan hóa thời gian chạy của MIDAS-R và MIDAS-R$^+$
- Thư mục *measurement*: Chứa các kết quả đo lường về độ chính xác và hiệu năng.
- Tệp *makefile*: Tự động hóa các công việc như biên dịch, chạy với các tập dữ liệu, Benchmark, trực quan hóa, tính độ chính xác,...
- Tệp *main.c*: Đọc dữ liệu từ tập dữ liệu, tính toán điểm bất thường với MIDAS-R/MIDAS-R$^+$ rồi lưu ra tệp.