BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÙI VĂN CÔNG

NGHIÊN CỨU PHƯƠNG PHÁP PHÁT HIỆN LỖ HỔNG MÃ NGUỒN DỰA TRÊN ĐỒ THỊ ĐẶC TRƯNG MÃ

Chuyên ngành: Kỹ thuật máy tính
Mã số: 9.48.01.06

TÓM TẮT LUẬN ÁN TIẾN SĨ KỸ THUẬT

HÀ NỘI - 2025

Công trình hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học:

PGS. TS. Đỗ Xuân Chợ

PGS. TS. Đỗ Trung Tuấn

Phản biện 1:

Phản biện 2:

Phản biện 3:

Luận án được bảo vệ trước Hội đồng chấm Luận án cấp Học viện họp tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Km10, Đường Nguyễn Trãi, Quận Hà Đông, Hà Nội.

Vào hồi: **giờ 00, ngày tháng năm 2025**

Có thể tìm hiểu luận án tại:

1. **Thư viện Quốc gia Việt Nam**
2. **Thư viện Học viện Công nghệ Bưu chính Viễn thông**

MỞ ĐẦU

1. GIỚI THIỆU

Đảm bảo chất lượng phần mềm đòi hỏi đánh giá và phát hiện lỗ hổng hiệu quả, chủ yếu qua hai phương pháp: dựa trên tập luật và dựa trên đặc trưng bất thường. Để nâng cao độ chính xác, cần cải thiện hai bước: trích xuất và phân loại đặc trưng mã nguồn. Về trích xuất, luận án lựa chọn phương pháp dựa trên đồ thị CPG do khả năng biểu diễn mối quan hệ phức tạp và độ chính xác cao. Về phân loại, các phương pháp hiện tại chưa hiệu quả với các bộ dữ liệu mất cân bằng. Do đó, luận án tập trung đề xuất cải tiến cả hai quá trình nói trên nhằm phát hiện lỗ hổng mã nguồn tốt hơn trên các bộ dữ liệu mất cân bằng.

2. MỤC TIÊU, ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU CỦA LUẬN ÁN

Mục tiêu của luận án: Tập trung phát triển phương pháp phát hiện lỗ hổng dựa trên phân tích đồ thị đặc trưng mã nguồn, nhằm đề xuất mô hình mới giúp nâng cao độ chính xác và khắc phục khó khăn trong trích xuất, lựa chọn, phân loại mã nguồn.

Đối tượng nghiên cứu:

- Các mã nguồn C/C++ và các lỗ hổng phổ biến như tràn bộ đệm, lỗi cú pháp, lỗi con trỏ,...
- Các công nghệ, kỹ thuật, quy trình phân tích mã nguồn.
- Một số hướng tiếp cận trong phát hiện lỗ hổng mã nguồn dựa trên thuật toán học máy, học sâu,…

Phạm vi nghiên cứu: Tập trung vào lý thuyết nền tảng, các phương pháp biểu diễn, phân tích, phát hiện lỗ hổng trong mã nguồn C/C++, dựa trên các NC đã công bố trong và ngoài nước.

3. PHƯƠNG PHÁP NGHIÊN CỨU

Luận án kết hợp nghiên cứu lý thuyết và thực nghiệm, trong đó lý thuyết phục vụ các nhiệm vụ cụ thể sau:

- NC nền tảng lý thuyết về phát hiện lỗ hổng mã nguồn;
- NC nền tảng lý thuyết về học máy, học sâu cho luận án;
- Khảo sát, đánh giá các đề xuất, giải pháp đã có cho phát hiện lỗ hổng mã nguồn, trên cơ sở đó tổng hợp các ưu điểm, nhược điểm làm cơ sở cho đề xuất của luận án;
- Lựa chọn, đề xuất các đặc trưng, xây dựng các mô hình phát hiện lỗ hổng mã nguồn dựa trên CPG.

Phương pháp thực nghiệm được sử dụng trong luận án:

- Khảo sát các tập dữ liệu về lỗ hổng và lựa chọn tập dữ liệu phù hợp cho thực nghiệm;
- Thử nghiệm và so sánh các mô hình phát hiện lỗ hổng được đề xuất với các mô hình hiện có.

4. CÁC ĐÓNG GÓP CỦA LUẬN ÁN

Luận án có hai đóng góp chính đó là:

Đóng góp thứ nhất của luận án là *đề xuất phương pháp kết hợp RoBERTa và GCN trong trích xuất các đặc trưng quan trọng từ CPG, cùng kỹ thuật cân bằng dữ liệu, nhằm nâng cao hiệu quả phân tích và phát hiện lỗ hổng mã nguồn.*

Đóng góp thứ hai của luận án là *đề xuất hướng tiếp cận mới giữa mạng đồ thị học sâu GGNN trong trích xuất đặc trưng*

mã nguồn và phương pháp học tương phản, giúp nâng cao độ chính xác trong phát hiện lỗ hổng mã nguồn, mở ra tiềm năng áp dụng cho các bài toán phân loại khác.

5. BỐ CỤC CỦA LUẬN ÁN

Chương 1. Tổng quan về lỗ hổng mã nguồn và vấn đề phát hiện lỗ hổng mã nguồn

Chương 1 trình bày tổng quan về lỗ hổng mã nguồn, gồm khái niệm, phân loại, ảnh hưởng, các phương pháp biểu diễn và phát hiện, đồng thời khảo sát nghiên cứu trong và ngoài nước làm cơ sở đề xuất cho luận án.

Chương 2. Đề xuất phương pháp phát hiện lỗ hổng mã nguồn dựa trên mô hình kết hợp

Chương 2 trình bày mô hình kết hợp mới gồm hai nhiệm vụ: trích xuất đặc trưng mã nguồn bằng mạng đồ thị và xử lý ngôn ngữ tự nhiên, cùng kỹ thuật sinh dữ liệu; cuối chương là phần thực nghiệm và đánh giá mô hình.

Chương 3. Cải thiện hiệu quả mô hình phát hiện lỗ hổng mã nguồn dựa trên kỹ thuật học biểu diễn

Chương 3 đề xuất hướng tiếp cận mới nhằm cải thiện trích xuất đặc trưng mã nguồn bằng GGNN và phát hiện lỗ hổng qua học tương phản. Nội dung gồm: kiến trúc mô hình, lý thuyết học biểu diễn, kỹ thuật cân bằng dữ liệu qua phân phối Bernoulli trong tầng Dropout của mạng CNN, và phần thực nghiệm đánh giá.

Phần cuối cùng của luận án chính là một số kết luận và định hướng phát triển nghiên cứu tiếp theo.

CHƯƠNG 1. LỖ HỔNG MÃ NGUỒN VÀ VẤN ĐỀ PHÁT HIỆN LỖ HỔNG MÃ NGUỒN

Tổng quan về lỗ hổng mã nguồn C/C++

Định nghĩa 1.1. Lỗ hổng phần mềm *là những điểm yếu trong quá trình phân tích, thiết kế mã nguồn hoặc vận hành hệ thống mà kẻ tấn công có thể lợi dụng để gây ra sự cố mất an toàn trong triển khai phần mềm.*

Những kẻ tấn công sẽ lợi dụng những sai sót này để làm mất an toàn của hệ thống.

Định nghĩa 1.2. Lỗ hổng mã nguồn *là những điểm sai sót trong mã lập trình của phần mềm, có thể bị khai thác để gây hại cho hệ thống hoặc người dùng.*

Phần mềm lớn thường do nhiều nhóm phát triển với công cụ khác nhau, dễ dẫn đến sai sót tiềm ẩn – nguyên nhân gây ra lỗ hổng phần mềm.

Theo Grampp và Morris cùng McGraw, có thể phân loại lỗ hổng phần mềm theo hai đặc tính kỹ thuật hay mức mã nguồn:

1. Phân theo đặc tính kỹ thuật: Dựa theo ba tiêu chí: (i) khu vực phát sinh lỗ hổng; (ii) các khiếm khuyết của hệ thống thông tin; và (iii) vị trí xuất hiện các lỗ hổng.
2. Phân loại lỗ hổng theo mức mã nguồn: Tập trung phân tích, phát hiện các lỗ hổng dựa trên mã nguồn ứng dụng.

Một số lỗ hổng phổ biến thường gặp trong C/C++ dưới đây

Bảng 1.1. Danh sách lỗ hổng mã nguồn phổ biến trong C/C++

TT	Lỗ hổng C/C++	Giải thích
1	Buffer OverFlow	Lỗi tràn bộ đệm
2	Insecure Data Storage	Lưu trữ dữ liệu không an toàn.

3	Dynamic memory errors	Quản lý không chính xác bộ nhớ động và con trỏ động
4	Invalid conversion	Ép kiểu sai
5	Sotfware Bugs	Lỗi trong phần mềm
6	Cannot convert 'type1' to 'type2'	Truyền sai kiểu tham số cho hàm
7	Format string vulnerabilities	Lỗi về định dạng của chuỗi
8	Insecure cryptography	Mật mã không an toàn.

Vấn đề phát hiện lỗ hổng mã nguồn C/C++

Thường dựa trên hai phương pháp chính:

1. **Phát hiện lỗ hổng C/C++ dựa trên dấu hiệu**: Bằng cách so khớp với dữ liệu lỗ hổng đã biết, qua các phương pháp như đối chiếu mẫu, phân tích từ vựng, phân tích luồng dữ liệu, cùng các công cụ Flawfinder, KLEE,...
2. **Phát hiện dựa trên đặc trưng bất thường:** Thường phân tích, dự đoán bằng một số thuật toán học máy, học sâu

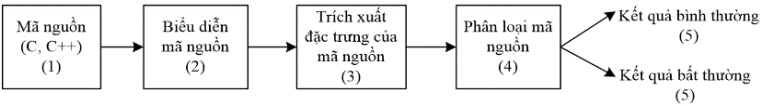
Các nghiên cứu liên quan

Dưới đây là bảng danh sách thống kê các công trình nghiên cứu được công bố trong 5 năm trở lại đây:

Bảng 1.5. Danh sách các nghiên cứu về phát hiện lỗ hổng mã nguồn

Đề xuất	Phương pháp trích xuất	Cân bằng dữ liệu	Thuật toán phân loại
Chakraborty và cộng sự [59]	GGNN	SMOTE	MLP và triplet loss
Chunyong Zhang cùng cộng sự [65]	Graph Attention Network	SMOTE	Metric Learning
Xiansheng Cao và các cộng sự [56]	GCN cho CPG BiLSTM cho AST	Không	Softmax
Wenjing Cai và các cộng sự [102]	Chuyển đổi đồ thị CPG	Không	TextCNN
Ruitong Liu cùng đồng nghiệp [51]	CodeLM + GNN	Không	Sigmoid
Amanpreet Singh Saimbhi [54]	CNN cho CPG	Không	Fully-connected

Tổng quan kiến trúc mô hình phân loại lỗ hổng mã nguồn



Hình 1.2. Mô hình tổng quát bài toán phát hiện lỗ hổng mã nguồn

Dựa trên hình 1.2 có thể thấy từ khối (1), khối (2) là quá trình chuẩn hoá và phân tích mã nguồn thông qua việc biểu diễn mã nguồn dưới dạng CPG. Điều này giúp cho việc biểu diễn các thông tin thuộc tính của mã nguồn như biến, hàm,… được đầy đủ hơn. Qua đó loại bỏ được các đoạn mã dư thừa như các khoảng trắng, các ký tự đặc biệt, các dòng chú thích,… Khối (3) là trích xuất đặc trưng mã nguồn, có thể sử dụng một số phương pháp phổ biến như Word2Vec, RoBERTa, DGCNN, GCN, GGNN,…Khối (4) là phân loại mã nguồn, có thể sử dụng một số hàm như Fully Connected, Softmax,…

Biểu diễn mã nguồn

Xét ví dụ dưới đây về đoạn mã ban đầu:

Ex_Graphembedde.cpp

```
1   #include <iostream>
2   using namespace std;
3   void foo()
4   {
5       int x = source();
6       if (x < MAX) {
7           int y = 2 * x;
8           sink(y);
9       }
10  }
```

Hình 1.3. Đoạn mã ban đầu của một chương trình

Từ đoạn mã (hình 1.3), biểu diễn mã nguồn về dạng đồ thị:

1. Biểu diễn mã nguồn với AST

Định nghĩa 1.3. Cây cú pháp trừu tượng AST là cây biểu diễn cấu trúc cú pháp của mã nguồn, được xây dựng sau khi chương trình được phân tích cú pháp nhưng bỏ qua các chi tiết không cần thiết như dấu ngoặc, dấu chấm phẩy…

2. Biểu diễn mã nguồn với CFG

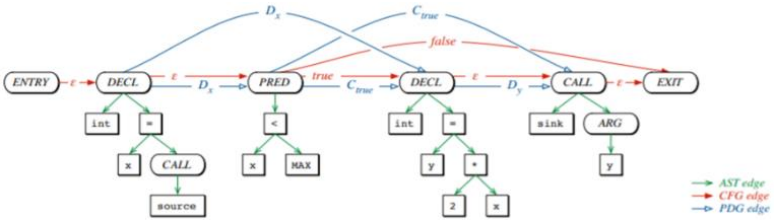
Định nghĩa 1.4. Đồ thị luồng điều khiển CFG *là một đồ thị có hướng, trong đó:mỗi đỉnh đại diện cho một khối mã, một chuỗi các câu lệnh liên tiếp không có nhánh rẽ; mỗi cạnh đại diện cho khả năng chuyển hướng thực thi từ khối này sang khối khác.*

3. Biểu diễn mã nguồn với PDG

Định nghĩa 1.5. Đồ thị phụ thuộc chương trình PDG *là đồ thị biểu diễn các phụ thuộc dữ liệu và điều khiển giữa các thành phần mã nguồn, giúp hiểu rõ luồng thông tin và điều kiện thực thi trong chương trình.*

4. Biểu diễn mã nguồn với CPG

Định nghĩa 1.6. Đồ thị đặc trưng mã nguồn CPG *là một đồ thị có hướng, biểu diễn các thực thể cú pháp, dữ liệu và điều khiển trong chương trình phần mềm, được thiết kế để phân tích bảo mật, học máy và truy vấn mã nguồn.*



Hình 1.7. Biểu diễn mã nguồn về dạng CPG

Một CPG bao gồm các thành phần biểu diễn chính:

- Các nút và kiểu nút: Các nút trong CPG đại diện một thành phần trong mã nguồn như lớp, phương thức,...
- Cạnh có hướng đánh nhãn: Biểu diễn quan hệ giữa các thành phần chương trình thông qua các nút tương ứng.
- Các cặp key-value: Các nút chứa các cặp thuộc tính key-value nơi các key hợp lệ phụ thuộc vào kiểu nút.

Trích xuất đặc trưng mã nguồn dưới dạng CPG

Để trích xuất đặc trưng từ CPG, các nghiên cứu thường dùng mạng đồ thị học sâu như GCN, DGCNN, GGNN,...

Cân bằng dữ liệu

Một số kỹ thuật như: kỹ thuật giảm mẫu lớp đa số; tăng mẫu lớp thiểu số; hay tăng mẫu ngẫu nhiên; kỹ thuật SMOTE.

Phân loại mã nguồn

Một số phương pháp, hàm phân loại phổ biến như: phân loại dựa trên học máy, học tương phản, hàm tương phản,...

Kết luận chương 1

Chương 1 đã trình bày tổng quan về lỗ hổng mã nguồn, đặc biệt mã nguồn C/C++, làm rõ tính cấp thiết của việc phát hiện sớm các lỗ hổng. Kết quả phân tích giúp xác định mục tiêu, định hướng nghiên cứu, là cơ sở cho đề xuất ở Chương 2 và 3 tiếp theo

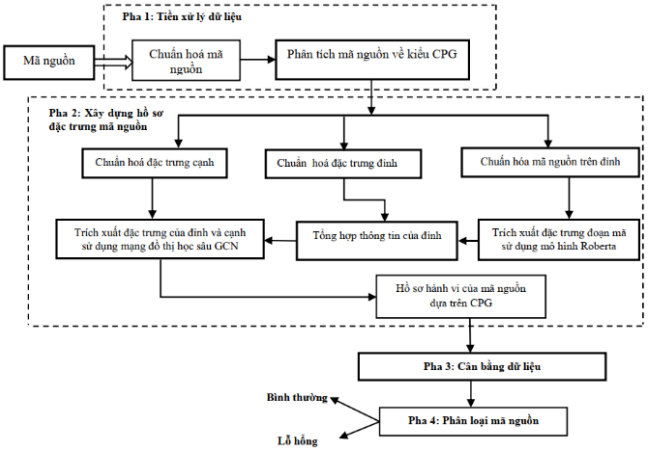
CHƯƠNG 2. ĐỀ XUẤT PHƯƠNG PHÁP PHÁT HIỆN LỖ HỔNG MÃ NGUỒN DỰA TRÊN MÔ HÌNH KẾT HỢP

Cơ sở khoa học của đề xuất

Luận án đề xuất kết hợp RoBERTa và GCN để khai thác hiệu quả đặc trưng trên CPG, đồng thời sử dụng kỹ thuật BDC (BDC là kỹ thuật sinh dữ liệu mới cho lớp thiểu số thông qua phân phối Bernoulli trong tầng Dropout của mạng CNN) thay thế SMOTE nhằm sinh dữ liệu thực tế và giảm nhiễu.

Đề xuất mô hình GRB trong phát hiện lỗ hổng mã nguồn

Kiến trúc tổng quát của mô hình GRB



Hình 2.1. Các thành phần và nguyên tắc hoạt động của mô hình GRB

Nguyên tắc trích xuất thông tin của mô hình GRB

Luận án đề xuất chuẩn hóa mã nguồn thành CPG để thuận tiện trích xuất đặc trưng. Đỉnh và cạnh được mã hóa bằng one-hot; đoạn mã trên đỉnh được nhúng bằng RoBERTa. Các đặc

trưng sau đó được tổng hợp và đưa vào GCN để trích xuất hồ sơ hành vi mã nguồn. Đây là hướng tiếp cận mới giúp trích xuất được đầy đủ hơn các thành phần đặc trưng từ CPG.

Xây dựng hồ sơ đặc trưng của mã nguồn dựa trên CPG

Qua hình 2.1, việc xây dựng hồ sơ đặc trưng của mã nguồn chính là sự kết hợp chuẩn hoá đặc trưng cạnh và tổng hợp thông tin của đỉnh. Từ đó đưa qua GCN để trích xuất các đặc trưng.

1. Xây dựng hồ sơ đặc trưng từ tổng hợp thông tin đỉnh
- Sử dụng RoBERTa để trích xuất đặc trưng ngữ nghĩa và cú pháp từ mã trên đỉnh CPG gồm: (1) tiền xử lý và huấn luyện lại BPE phù hợp; (2) mã hóa mã nguồn thành vector đặc trưng qua tầng nhúng và encoder.
- Đặc trưng đỉnh (IfStatement, Variable, Return,...) được one-hot hóa thành ma trận kích thước $N \times K$.

2. Chuẩn hoá đặc trưng cạnh: Các cạnh CPG (AST, ControlFlow,...) được one-hot hóa với chỉ số riêng, biểu diễn quan hệ đỉnh bằng vector nhị phân để GCN nhận diện luồng thông tin.
3. Trích xuất đặc trưng CPG bằng GCN: Mỗi lớp GCN lan truyền và tổng hợp thông tin từ các đỉnh lân cận, sử dụng phép "tích chập" tương tự CNN để trích xuất đặc trưng theo công thức sau:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) + H^{(l)} \qquad (2.1)$$

Kỹ thuật cân bằng dữ liệu dựa trên phân bố Bernoulli

Định nghĩa 2.1. Phân phối nhị phân Bernoulli *mô tả kết quả của một phép thử chỉ có hai khả năng xảy ra: 1 (thành công) với xác suất p, và 0 (thất bại) với xác suất 1- p.*

Định nghĩa 2.2. Phân bố Bernoulli trong phân loại lỗ hổng mã nguồn *là một mô hình xác suất mô tả khả năng một đoạn mã cụ thể có chứa lỗ hổng hay không.*

Với biến ngẫu nhiên a_k đại diện trạng thái của đoạn mã:

- $a_k = 1$: đoạn mã có chứa lỗ hổng
- $a_k = 0$: đoạn mã không có lỗ hổng

Việc sử dụng *BDC* cho cân bằng dữ liệu, biểu diễn vector $x = (x_1, x_2, \dots x_d)$, mỗi thành phần x_k $(k = 1, 2, \dots d)$ sẽ là:

$$\widehat{x_k} = a_k \,.\, x_k \tag{2.2}$$

Với $a_k \sim P$ là một biến ngẫu nhiên có phân phối Bernoulli:

$$P(a_k) = \begin{cases} 1 - p, & a_k = 0 \\ p \ , & a_k = 1 \end{cases} \tag{2.3}$$

Ở mỗi lần thử nghiệm trong bài toán sinh dữ liệu ở lớp thiểu số (tức lớp có lỗ hổng), phân phối Bernoulli sẽ:

1. Sinh ngẫu nhiên một giá trị theo phân bố Bernoulli
2. Kết quả trả về 0 (không có lỗ hổng), 1 (có lỗ hổng)
    - Với xác suất p, nhận giá trị 1
    - Với xác suất $1 - p$, nhận giá trị 0

Quá trình sinh dữ liệu theo Bernoulli chia làm hai bước:

Bước 1: Chọn một xác suất $p \in [0,1]$

Bước 2: Sinh ngẫu nhiên một số $x_k \in [0,1]$, từ phân bố đều với điều kiện

- $x_k \leq p$ cho kết quả là 1
- $x_k > p$ cho kết quả là 0

Thuật toán 2.1: Cân bằng dữ liệu sử dụng BDC

Input: Đặc trưng -dac_trung
Nhãn dán -$nhan$
Tỷ lệ -α
Số lượng mẫu mới được sinh ra trên lớp thiểu số -n
$D = \{\}$
Output: D_{canbang}
Function: $BDC(dac\_trung, nhan, \alpha = 0.1, n = 10)$:
for $f_i, l_i \in dac\_trung, nhan$ do:
if $l_i \; of \; lop\_thieu\_so$ do:
for $t := 1$ to n do:
$f'_i \leftarrow tao\_du\_lieu\_lop\_thieu\_so\_theo\_ty\_le(\alpha)$
$Them \;\; f'_i \;\; vào \;\; D$
end
end
end
return $D_{canbang}$

Từ thuật toán 2.1 có thể thấy:

Bước 1: Tìm kiếm đặc trưng của lớp thiểu số

Bước 2: Lặp các đặc trưng của lớp thiểu số, áp dụng *BDC* sinh dữ liệu với lớp thiểu số.

Bước 3: Thêm dữ liệu sau khi sinh vào dữ liệu gốc.

Thuật toán phân loại

Luận án sử dụng hàm kết nối đầy đủ để phân loại.

Môt số kết quả thực nghiệm

1. Đánh giá mô hình đề xuất trên bộ dữ liệu Verum

Bảng 2.3. Kết quả thực nghiệm trên mô hình đề xuất GRB

Mô hình	Kích thước	Acc	Pre	Rec	F1
GCN + RoBERTa + *BDC*	**512**	**87.58**	**39.42**	**67.07**	**49.65**
GCN + RoBERTa + *BDC*	256	87.0	37.79	65.68	47.97
GCN + RoBERTa + *BDC*	1024	85.23	39.69	65.59	48.95

2. Kết quả thực nghiệm 2

Đánh giá vai trò của GCN đối trong mô hình đề xuất

Bảng 2.4. Kịch bản đánh giá vai trò của GCN trong mô hình đề xuất

Hướng tiếp cận	Mô hình	Kích thước	Acc	Pre	Rec	F1
Đề xuất	**GCN + RoBERTa + *BDC***	**512**	**87.58**	**39.42**	**67.07**	**49.65**
Thay thế GCN bằng DGCNN	DGCNN+RoBERTa+*BDC*	512	85.67	39.29	65.24	49.04
	DGCNN+RoBERTa+*BDC*	256	85.24	39.06	65.12	48.83
	DGCNN+RoBERTa+*BDC*	1024	85.23	39.15	64.84	48.82
Không sử dụng GCN	RoBERTa+*BDC*	512	87.26	38.18	63.81	47.78
	RoBERTa+*BDC*	256	86.74	36.84	63.45	46.62
	RoBERTa+*BDC*	1024	87.24	38.21	64.65	48.04

Đánh giá vai trò của RoBERTa đối với mô hình đề xuất

Bảng 2.5. Kết quả thực nghiệm đánh giá vai trò RoBERTa

Hướng tiếp cận	Mô hình	Acc	Pre	Rec	F1
Đề xuất luận án	**GCN+RoBERTa+*BDC***	**87.58**	**39.42**	**67.07**	**49.65**
Thay thế RoBERTa bằng BERT	GCN+BERT+*BDC*	86.86	37.61	66.83	48.13

Hướng tiếp cận	Mô hình	Acc	Pre	Rec	F1
Thay thế RoBERTa bằng Word2Vec	GCN+Word2Vec+*BDC*	86.22	37.48	66.3	48.03
Thay thế RoBERTa bằng Doc2Vec	GCN+Doc2Vec+*BDC*	85.79	37.62	65.66	47.83
Không sử dụng RoBERTa	GCN+*BDC*	82.16	26.72	54.82	35.93

Đánh giá vai trò của mô hình kết hợp GCN và RoBERTa

Bảng 2.6. Kết quả khi không sử dụng mô hình GCN kết hợp RoBERTa

Mô hình	Acc	Pre	Rec	F1
Đề xuất của luận án	**87.58**	**39.42**	**67.07**	**49.65**
GGNN + *BDC*	83.38	29.40	58.62	39.17
GGNN	83.05	28.76	58.02	38.46

Đánh giá vai trò BDC trong mô hình đề xuất

Bảng 2.7. Kết quả thực nghiệm đánh giá vai trò BDC trong mô hình

Hướng tiếp cận	Mô hình	Kích thước	Acc	Pre	Rec	F1
Đề xuất	**GCN+RoBERTa+*BDC***	**512**	**87.58**	**39.42**	**67.07**	**49.6**
Thay thế *BDC* bằng SMOTE	GCN+RoBERTa+SMOTE	512	87.05	37.89	65.5	48.01
	GCN+RoBERTa+SMOTE	256	87.12	38.22	66.01	48.41
	GCN+RoBERTa+SMOTE	1024	87.52	39.25	66.67	49.24
Không sử dụng *BDC*	GCN + RoBERTa	512	87.24	38.36	64.4	48.08
	GCN + RoBERTa	256	86.6	37.15	63.73	47.55
	GCN + RoBERTa	1024	87.21	38.13	64.41	47.9

3. Kết quả thực nghiệm 3: So sánh mô hình đề xuất với 1 số hướng tiếp cận khác trên cùng bộ dữ liệu

Bảng 2.8. Kết quả so sánh đề xuất mô hình luận án với các hướng tiếp cận khác trên cùng bộ dữ liệu Verum và FFmpeg+Qume

Bộ dữ liệu	Hướng tiếp cận	Acc	Pre	Rec	F1
Verum	**Đề xuất của luận án**	**87.58**	**39.42**	**67.07**	**49.65**
	REVEAL	86.94	34.03	64.24	44.49
	Russell	90.98	24.63	10.91	15.24
	VulDeePecker	89.05	17.68	13.87	15.7
	SySeVR	84.22	24.46	40.11	30.25
	Devign	88.41	34.61	26.67	29.87
FFmpeg+Qume	**Đề xuất của luận án**	**62.78**	**57.15**	**72.98**	**64.1**
	Russell	58.13	54.04	39.50	45.62
	VulDeePecker	53.58	47.36	28.70	35.20
	SySeVR	52.52	48.34	65.96	56.03
	Devign	58.57	53.60	62.73	57.18
	REVEAL	62.51	56.85	74.61	64.42

Kết luận chương 2

Các kết quả chính trong Chương 2 gồm:

(1). Đề xuất mô hình GRB kết hợp xử lý ngôn ngữ tự nhiên và mạng đồ thị học sâu nhằm nâng cao hiệu quả trích xuất đặc trưng mã nguồn C/C++ dưới dạng đồ thị CPG;

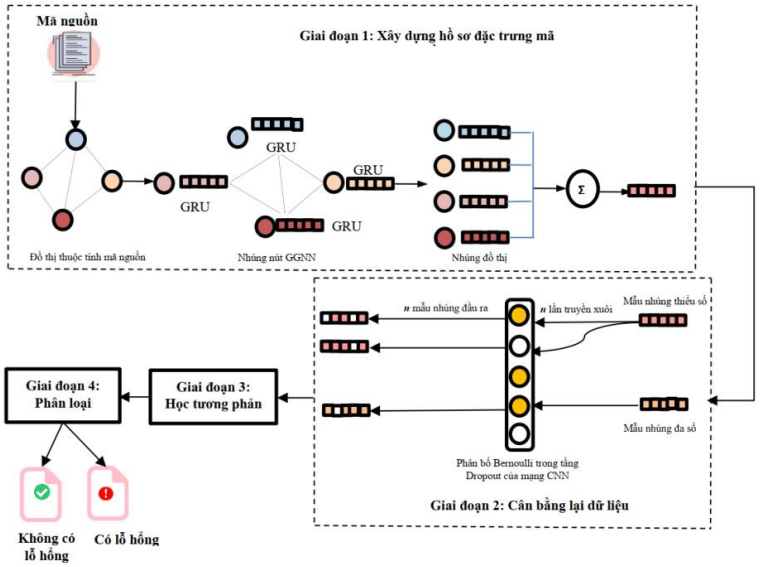
(2). Đề xuất kỹ thuật sinh dữ liệu mới hiệu quả hơn phương pháp truyền thống, mở ra hướng tiếp cận mới cho bài toán cân bằng dữ liệu, đặc biệt với mã nguồn C/C++.

CHƯƠNG 3. CẢI THIỆN HIỆU QUẢ MÔ HÌNH PHÁT HIỆN LỖ HỔNG MÃ NGUỒN DỰA TRÊN KỸ THUẬT HỌC BIỂU DIỄN

Cơ sở khoa học của đề xuất

Luận án đề xuất mô hình DrCSE, kết hợp GGNN và học tương phản với hàm mất mát Entropy chéo để trích xuất và phân loại đặc trưng mã nguồn C/C++. GGNN khắc phục hạn chế của GCN và tích hợp GloVe để cải thiện biểu diễn ngữ nghĩa. Kỹ thuật học tương phản giúp xử lý nhiễu do sinh dữ liệu, nâng cao hiệu quả phân loại lỗ hổng.

Kiến trúc tổng quát của mô hình DrCSE



Hình 3.1. Kiến trúc của mô hình DrCSE

Nguyên tắc trích xuất đặc trưng của mô hình

Trích xuất đặc trưng CPG trên đỉnh dùng hàm GloVe

Sau khi biểu diễn mã nguồn dưới dạng CPG, luận án trích xuất đỉnh và nội dung liên quan, trong đó mỗi đỉnh đại diện cho các thành phần như câu lệnh, biểu thức, biến, hàm, định nghĩa hoặc sử dụng biến. Nội dung từ mỗi đỉnh (ví dụ: tên hàm, giá trị biến) được chuyển thành chuỗi văn bản và mã hóa bằng GloVe.

Quy trình mã hóa: (1) tiền xử lý văn bản đỉnh, (2) chuyển văn bản thành vector đặc trưng E_v bằng GloVe, (3) kết hợp E_v với vector one-hot biểu diễn loại đỉnh T_v, tạo thành vector đặc trưng tổng hợp X_v cho đỉnh v. Tính theo công thức GloVe.

$$\omega_i^T \tilde{\omega}_j + \mathrm{a}_i + \tilde{\mathrm{a}}_j = \log(X_{ij}) \quad (3.1)$$

Trong đó:

- ω_i: Vector biểu diễn từ mục tiêu của từ i
- $\tilde{\omega}_j$: Vector biểu diễn từ ngữ cảnh của từ j
- a_i, $\tilde{\mathrm{a}}_j$: bias của từ mục tiêu và từ ngữ cảnh
- X_{ij}: Số lần từ j xuất hiện trong ngữ cảnh của từ i
- $\log(X_{ij})$: logarit của tần suất đồng xuất hiện

Để huấn luyện mô hình, luận án dùng hàm mất mát sau đây:

$$J = \sum_{i,j=1}^{V} f(X_{ij}).(\omega_i^T \tilde{\omega}_j + \mathrm{a}_i + \tilde{\mathrm{a}}_j - \log(X_{ij}))^2 \quad (3.2)$$

Trong đó: $f(X_{ij})$: Là hàm trọng số để giảm ảnh hưởng của các cặp từ có tần suất cực lớn hoặc cực nhỏ.

Sau khi huấn luyện xong: Vector biểu diễn cuối cùng cho mỗi từ là tổng của 2 vector được thể hiện qua công thức sau:

$$Embedding(\omega) = \omega_i + \tilde{\omega}_i \quad (3.3)$$

Trích xuất đặc trưng CPG trên cạnh sử dụng GGNN

Để trích xuất được những thông tin của những đỉnh này, luận án sử dụng GGNN. Công thức 3.4 mô tả cách thức áp dụng mô hình GGNN cho nhiệm vụ trích xuất thông tin của cạnh.

$$X_g = \sum_{v \in V} (GRU(X_v, \sum_{u \in E} g(X_u))) \quad (3.4)$$

Trong đó:

$GRU(.)$ là hàm hồi tiếp có cổng,

X_v là vector đại diện của đỉnh v.

$g(.)$ hàm biến đổi giúp tổng hợp thông tin các đỉnh u chính là các đỉnh lân cận của đỉnh v cùng vector đại diện X_u tương ứng.

Thuật toán 3.1 dưới đây mô tả cách xây dựng hồ sơ đặc trưng mã nguồn từ phương pháp phân tích CPG trong DrCSE.

Thuật toán 3.1. Xây dựng hồ sơ đặc trưng mã nguồn

Input: Mã nguồn - C
Output: Vector đặc trưng x_g đại diện cho C
Procedure:
Function $graphEmbed(C)$:
    (V, E) ← Code_property_graph (C)
    for $v \in V$ do:
        $T_v \leftarrow onehot(v.type())$
        $C_v \leftarrow glove(v.code())$
        $x_v \leftarrow concat(T_v, C_v)$
        $X \leftarrow x_v \cup X$
    end
    $\boldsymbol{X'} \leftarrow \boldsymbol{GGNN(X, E)}$
    $\boldsymbol{x_g} \leftarrow \boldsymbol{Aggregate(X')}$
return $\boldsymbol{x_g}$

Nâng cao hiệu quả phân loại sử dụng học tương phản

Cách thức hoạt động của phương pháp này như sau:

Giả sử ta có tập các điểm dữ liệu $D = \{(x_i, x_i^+, x_i^-)\}$. Trong đó, x_i và x_i^+ là hai điểm dữ liệu tương đồng hay có cùng nhãn, x_i và x_i^- là hai điểm dữ liệu khác biệt hay khác nhãn. Gọi h_i, h_i^+, h_i^- lần lượt là các vector đại diện của x_i, x_i^+, x_i^-, khi đó mục tiêu huấn luyện một mini-batch *N* định nghĩa như sau:

$$L_{cl} = \sum_{i=1}^{n} L_i \tag{3.5}$$

Với L_i được định nghĩa theo công thức (3.6) như sau:

$$L_i = -\frac{1}{N_{y_i}} \cdot \sum_{k=1}^{N} 1_{i \neq k} \cdot 1_{y_i = y_k} \cdot log \frac{e^{sim(h_i, h_k)/\tau}}{\sum_{j=1}^{N} 1_{i \neq j} \cdot 1_{y_i \neq y_j} e^{sim(h_i, h_j)/\tau}} \tag{3.6}$$

Trong đó:

- $\boldsymbol{\tau}$ là mức độ cân bằng giữa mẫu âm và mẫu dương,
- $sim\ (h_1, h_2)$ là sự tương đồng cosine $\frac{h_1^T\ h_2}{||h_1||\ .||h_2||}$,
- $1_B = 1$ khi *B* đúng, ngược lại $1_B = 0$,
- N_y là tổng số mẫu trong mini-batch có cùng nhãn y,
- i là chỉ số của ví dụ trong mini-batch,
- k là chỉ số của các ví dụ khác trong mini-batch có cùng nhãn với chỉ số của ví dụ i hay $x_k = x_i^+$,
- j là chỉ số của các ví dụ có chỉ số i trong mini-batch.

Phương trình trên, h_k chính vector đại diện của x_i^+. Đồng thời, nghiên cứu sử dụng nhúng đồ thị để trích xuất vector đại diện $h = f_\Theta(x) = graphEmbed(x)$ sau đó huấn luyện mô hình sử dụng mục tiêu học tương phản (phương trình 3.6).

Hàm mất mát Entropy chéo (Cross-Entropy Loss)

Với một bài toán phân loại, hàm mất mát Entropy chéo thể hiện qua công thức (3.7):

$$L_{ce} = -\frac{1}{N}\sum_{k=1}^{N} y_i . log\,(\sigma(h_i)) + (1 - y_i) . log(1 - \sigma(h_i)) \quad (3.7)$$

Trong đó,

- h_i là vector biểu diễn của x_i,
- $\sigma(x)$ là hàm sigmoid.

Hàm mất mát Entropy chéo hoạt động hiệu quả trong tối ưu hóa các mô hình phân loại nhờ các đặc điểm sau:

- Nhấn mạnh vào sai số lớn: Mô hình dự đoán xác suất cho nhãn đúng gần 0, hàm mất mát lớn khuyến khích mô hình điều chỉnh mạnh để cải thiện.
- Thường kết hợp với hàm Softmax, còn hàm mất mát Entropy chéo tận dụng trực tiếp đầu ra Softmax để tính toán mức độ sai lệch.

Thuật toán 3.3. Hàm mất mát Entropy chéo (Cross-Entropy)

Input: Representation Model - RM
Balanced Dataset - $D_{balanced}$
Output: Cross Entropy Loss
Procedure:
Function $crossEntropyLossFunction(RM, D_{balanced})$:
  $L_{ce} \leftarrow 0$
  for $x_i, l_i \in D_{balanced}$ **do**:
    $h_i \leftarrow RM.encode(x_i)$
    $y_i \leftarrow RM.predict(h_i)$
    $L_i \leftarrow y_i . log\,(\sigma(h_i)) + (1 - y_i) . log(1 - \sigma(h_i))$
    $L_{ce} \leftarrow L_{ce} + L_i$
  end
    $L_{ce} \leftarrow -L_{ce} / N$

return L_{ce}

Thử nghiệm và đánh giá mô hình đề xuất

Kết quả thực nghiệm kịch bản 1 với đề xuất DrCSE

Bảng 3.2. Kết quả thực nghiệm mô hình DrCSE

τ	α											
	0.1				0.2				0.3			
	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
0.05	85.96	36.41	63.35	46.24	85.89	36.75	62.21	46.20	85.39	37.62	60.30	46.33
0.1	**87.72**	39.35	**69.07**	**50.14**	87.34	37.81	66.33	48.16	86.72	38.57	63.24	47.91
0.2	86.16	**40.08**	66.72	50.01	85.92	36.75	64.11	46.72	85.03	36.41	62.01	45.88

Kết quả thực nghiệm kịch bản 2

Bảng 3.3 dưới đây trình bày các kết quả của các mô hình được đề xuất trong ARQ1, ARQ2 và ARQ3 thuộc kịch bản 2.

Bảng 3.3. Kết quả thực nghiệm kịch bản 2

RQ	ARQ	Hướng tiếp cận	Acc	Pre	Rec	F1
RQ1	ARQ1	SMOTE và Contrastive Loss	**87.72**	**39.35**	**69.07**	**50.14**
	ARQ2	BDC và Triplet Loss	87.23	34.51	66.47	45.43
		BDC và MLP	86.89	31.12	46.55	37.30
		BDC và RF	85.25	29.34	47.09	36.15
RQ2	ARQ3	SMOTE + Triplet Loss	86.94	34.03	64.24	44.49

Dựa trên kết quả thực nghiệm tại bảng 3.3, thể hiện vai trò khi sử dụng BDC, phương pháp học tương phản cho kết quả tốt

Kết quả thực nghiệm kịch bản 3: So sánh DrCSE với các hướng tiếp cận khác trên cùng bộ dữ liệu Verum

Bảng 3.4. Kết quả thực nghiệm kịch bản 3

Hướng tiếp cận	Accuracy	Precision	Recal	F1-score
Mô hình đề xuất DrCSE	**87.72**	**39.35**	**69.07**	**50.14**
Mô hình đề xuất GRB	87.58	39.42	67.07	49.65
REVEAL	86.94	34.03	64.24	44.49
Russell	90.98	24.63	10.91	15.24
VulDeePecker	89.05	17.68	13.87	15.7
SySeVR	84.22	24.46	40.11	30.25
Devign	88.41	34.61	26.67	29.87

Kết quả thực nghiệm 4: So sánh DrCSE với các hướng tiếp cận khác trên cùng bộ dữ liệu FFmpeg+Qemu

Bảng 3.5. Kết quả thực nghiệm kịch bản 4

Hướng tiếp cận	Accuracy	Precision	Recal	F1-score
Mô hình đề xuất DrCSE	**64.37**	**58.19**	**76.29**	**66.02**
Mô hình đề xuất GRB	87.58	39.42	67.07	49.65
Russell	58.13	54.04	39.50	45.62
VulDeePecker	53.58	47.36	28.70	35.20
SySeVR	52.52	48.34	65.96	56.03
Devign	58.57	53.60	62.73	57.18
REVEAL	62.51	56.85	74.61	64.42

Kết luận chương 3

Chương 3 trình bày mô hình DrCSE nhằm nâng cao hiệu quả phân loại lỗ hổng mã nguồn C/C++, với cơ sở khoa học rõ ràng và kết quả thực nghiệm tích cực. Luận án cũng phân tích quy trình hoạt động của mô hình và đánh giá hiệu quả qua nhiều kịch bản, đồng thời hướng dẫn lựa chọn tham số phù hợp.

KẾT LUẬN

Phát hiện lỗ hổng mã nguồn là vấn đề cấp thiết. Luận án đề xuất mô hình cải tiến trích xuất đặc trưng mã nguồn dựa trên xử lý ngôn ngữ tự nhiên và mạng đồ thị học sâu, tích hợp kỹ thuật BDC và học tương phản để nâng cao hiệu quả phân loại.

Hai đóng góp chính của luận án bao gồm:

Thứ nhất: Luận án đề xuất mô hình phát hiện lỗ hổng mới dựa trên hai nhiệm vụ chính: (i) trích xuất đặc trưng mã nguồn bằng cách kết hợp RoBERTa và GCN trên CPG; (ii) áp dụng kỹ thuật sinh dữ liệu. Được chứng minh tại [CT1, CT2, CT4].

Thứ hai: Đề xuất phương pháp nâng cao độ chính xác phát hiện lỗ hổng bằng cách kết hợp GGNN để trích xuất đặc trưng và học tương phản để cải thiện biểu diễn. Kết quả thực nghiệm cho độ chính xác cao, có giá trị thực tiễn. Minh chứng tại [CT3, CT5].

Trong tương lai, một số phương pháp, kỹ thuật mới có thể áp dụng để cải thiện các mô hình đề xuất trong luận án bao gồm: (i) phương pháp tổng hợp và trích xuất thuộc tính mới, như sử dụng học tăng cường trong việc trích xuất được các đặc trưng trên các bộ dữ liệu chưa được gán nhãn,…(ii) áp dụng các mô hình ngôn ngữ lớn để tổng hợp và trích xuất đặc trưng mã nguồn thay vì sử dụng kỹ thuật nhúng đồ thị như hiện tại.

DANH MỤC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN ÁN

[CT1].**Van-Cong-Bui**, Xuan-Do-Cho, *Detecting software vulnerabilities based on source code analysis using GCN Transformer*, in 2023 RIVF Int. Conf. Comput. Commun. Technol. (RIVF), pp.112–117, Hanoi, Vietnam, 2023, https://doi.org/10.1109/RIVF60135.2023.10471834.

[CT2].Cho Do Xuan, **B.V.Cong**, *An advanced computing approach for software vulnerability detection*, Multimedia Tools and Applications, vol. 83, pp. 86707-86740, June 2024, DOI: 10.1007/s11042-024-19682-y (**ISI Q1**).

[CT3].**Bui Van Cong**, Cho Do Xuan, *A New Framework for Software Vulnerability Detection Based on an Advanced Computing*, the Journal Computers, Materials & Continua, vol. 79, no. 3, pp. 3699-3723, June 2024, *https://doi.org/10.32604/cmc.2024.050019 (***ISI Q2***).*

[CT4].**Bùi Văn Công**, Đỗ Xuân Chợ, Đỗ Trung Tuấn, *Hướng tiếp cận phát hiện lỗ hổng phần mềm dựa trên mô hình kết hợp Graph Transformer và Rebalancing Data*, kỷ yếu Hội nghị Khoa học công nghệ Quốc gia lần thứ XVII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), PTIT, Hà Nội, ngày 08-09/8/2024. DOI: 10.15625/vap.2024.0225.

[CT5].**Bui Van Cong**, Do Xuan Cho, Do Trung Tuan, *Detection of source code vulnerabilities using Nature language processing and deep graph network*, the Journal of Science and Technology on Information security, vol. 23, no. 3, pp. 27-42, 2024, DOI: 10.54654/isj.v3i23.1057.