ĐỀ XUẤT DỰ ÁN NGHIÊN CỨU

Mở rộng khả năng phát hiện lỗ hổng bảo mật SedSVD cho ngôn ngữ Java

Động lực

Phát hiện lỗ hổng bảo mật trong mã nguồn là một thách thức quan trọng trong phát triển phần mềm. Công trình "SedSVD: Statement-level software vulnerability detection based on Relational Graph Convolutional Network with subgraph embedding" đề xuất phương pháp hiệu quả để phát hiện lỗ hổng bảo mật ở mức độ câu lệnh, đạt chỉ số F1-measure 95,15%. Tuy nhiên, mô hình này hiện chỉ được áp dụng cho C/C++, như đã được chỉ ra trong phần "Threats to validity": "Vì mô hình chủ yếu nhắm vào các chương trình C/C++, kết quả phát hiện có thể bị hạn chế."

Dự án này tập trung vào việc mở rộng SedSVD cho ngôn ngữ Java - một trong những ngôn ngữ phổ biến nhất trong phát triển phần mềm hiện nay. Chúng em sẽ tận dụng khả năng mở rộng sẵn có của mô hình mà không phải phát triển công cụ phân tích mới, như tác giả đã đề cập: "vì mô hình chúng tôi thiết kế không phụ thuộc vào ngôn ngữ lập trình, nên có thể mở rộng và linh hoạt".

Phương pháp

Để mở rộng SedSVD cho ngôn ngữ Java, chúng em sẽ:

1. **Sử dụng công cụ có sẵn để tạo CPG từ mã Java**: Áp dụng công cụ như Joern để tạo đồ thị thuộc tính mã (CPG) cho mã nguồn Java, với cấu trúc tương tự như đồ thị được tạo bởi Joern cho C/C++.
2. **Điều chỉnh bộ tiêu chí chọn nút trung tâm**: Ánh xạ các loại nút trong bảng 1 của bài báo (Call, indirectIndexAccess, addressOf, ControlStructure, v.v.) sang các khái niệm tương đương trong Java, ví dụ như Method Invocation, Array Access, Reference Expression và Control Statements.
3. **Điều chỉnh biểu diễn mã**: Cập nhật phương pháp nhúng mã (Word2vec) để xử lý cú pháp Java, đảm bảo các token có được biểu diễn vector phù hợp với đặc điểm của ngôn ngữ này.
4. **Sử dụng lại mô hình RGCN**: Giữ nguyên kiến trúc mô hình RGCN và MLP như trong bài báo gốc, nhưng huấn luyện lại trên dữ liệu Java.

Thí nghiệm dự kiến

Chúng em sẽ thực hiện các thí nghiệm sau:

1. **Thu thập dữ liệu Java**: Xây dựng bộ dữ liệu Java từ các nguồn như NVD (National Vulnerability Database) và các kho mã nguồn mở có chứa các vấn đề bảo mật đã được ghi nhận, tương tự như cách bộ dữ liệu C/C++ được xây dựng trong bài báo gốc.

2. **Tạo nhãn chuẩn**: Áp dụng phương pháp tạo nhãn chuẩn được cải tiến trong bài báo (Algorithm 1) cho mã nguồn Java, với các điều chỉnh phù hợp để xác định chính xác các thành phần mã dễ bị tổn thương.
3. **Đánh giá so sánh**: So sánh hiệu suất của mô hình mở rộng trên Java với các phương pháp hiện có như SpotBugs và FindSecurityBugs, sử dụng các chỉ số FPR, FNR, ACC và F1-measure.
4. **Phân tích chéo ngôn ngữ**: Đánh giá khả năng áp dụng các đặc trưng học được từ C/C++ sang Java để xác định tính hiệu quả của việc chuyển giao kiến thức giữa các ngôn ngữ.

Tập dữ liệu tham khảo: OWASP Benchmark là một bộ dữ liệu chuẩn cho việc đánh giá các công cụ phát hiện lỗ hổng bảo mật trong Java, chứa nhiều loại lỗ hổng bảo mật phổ biến.

Nghiên cứu tham khảo: "Deep Learning Based Vulnerability Detection: Are We There Yet" (Chakraborty et al., IEEE Transactions on Software Engineering, 2021) cung cấp một tổng quan về các phương pháp học sâu trong phát hiện lỗ hổng, bao gồm các thách thức khi áp dụng cho nhiều ngôn ngữ.