Trường Đại học Công nghệ Thông tin – ĐHQG Tp. HCM
Khoa Mạng máy tính và truyền thông

# BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

**Môn học:** Lập trình an toàn và khai thác lỗ hổng phần mềm

**Tên chủ đề: SySeVR: Framework for Using Deep Learning to Detect Software Vulnerabilities**

*Mã nhóm: G05 Mã đề tài: CK16*

**Lớp**: NT521.N11.ANTT

## 1. THÔNG TIN THÀNH VIÊN NHÓM:

*(Sinh viên liệt kê tất cả các thành viên trong nhóm)*

| STT | Họ và tên | MSSV | Email |
|---|---|---|---|
| 1 | Võ Anh Kiệt | 20520605 | 20520605@gm.uit.edu.vn |
| 2 | Nguyễn Bùi Kim Ngân | 20520648 | 20520648@gm.uit.edu.vn |
| 3 | Nguyễn Bình Thục Trâm | 20520815 | 20520815@gm.uit.edu.vn |

## 2. TÓM TẮT NỘI DUNG THỰC HIỆN:[1]

### A. Chủ đề nghiên cứu trong lĩnh vực An toàn phần mềm:

☑ Phát hiện lỗ hổng bảo mật phần mềm

☐ Khai thác lỗ hổng bảo mật phần mềm

☐ Sửa lỗi bảo mật phần mềm tự động

☐ Lập trình an toàn

☐ Khác: ……

### B. Tên bài báo tham khảo chính:

| |
|---|
| Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., & Chen, Z. (2021). Sysevr: A framework for using deep learning to detect software vulnerabilities. IEEE Transactions on Dependable and Secure Computing. |

### C. Dịch tên Tiếng Việt cho bài báo:

| |
|---|
| SySeVR – Framework cho việc ứng dụng Deep Learning để việc phát hiện lỗ hổng phần mềm |

[1] Ghi nội dung tương ứng theo mô tả

## D. Tóm tắt nội dung chính:

<mô tả nội dung tóm tắt của bài báo/chủ đề trong vòng 350 từ>

Phát hiện lỗ hổng phần mềm là vấn đề rất quan trọng và vẫn chưa được giải quyết triệt để bởi nhiều lỗ hổng luôn được báo cáo hằng ngày. Do đó mở ra hướng đi ứng dụng machine learning vào phát hiện lỗ hổng.

Deep learning trở nên hấp dẫn cho mục tiêu trên bởi nó có thể giúp giảm bớt các chức năng yêu cầu thực hiện thủ công. Mặc dù đã đạt được nhiều thành tựu, nhưng deep learning chưa được tìm hiểu một cách hệ thống trong lĩnh vực phát hiện lỗ hổng. Do vậy, nhóm tác giả đề xuất framework có hệ thống đầu tiên dùng deep learning để phát hiện lỗ hổng.

SySeVR: Syntax-based, Semantics-based, and Vector Representations (biểu diễn vector dựa trên cú pháp và ngữ nghĩa)

## E. Tóm tắt các kỹ thuật chính được mô tả sử dụng trong bài báo:

Về tổng quan model :

Bước 1: Thu thập Input

+ Chương trình cho learning (training)

+ Mục tiêu (testing)

Bước 2: Xử lý: qua các giai đoạn sau

+ Xử lý dựa trên Cú pháp SyVCs

+ Xử lý dựa trên Ngữ nghĩa SeVCs từ SyVCs

+ Biểu diễn Vector từ SeVCs

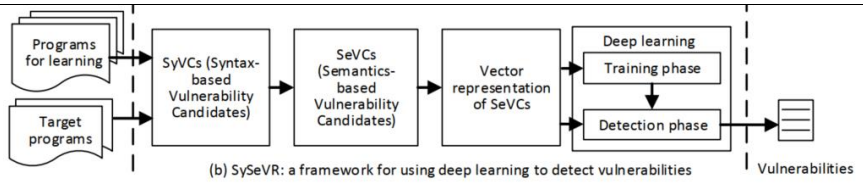
+ Deep Learning:

- Nếu đi qua pha training thì dữ liệu sẽ hỗ trợ cho phần training model

- Nếu đi qua pha phát hiện lỗi thì sẽ tích hợp model đã được training để thực hiện hỗ trợ trong kiểm tra lỗi và đưa ra báo cáo

Bước 3: Xuất Output:

+ Đưa ra báo cáo về các lỗi đã được tìm thấy



(b) SySeVR: a framework for using deep learning to detect vulnerabilities

Model gồm 3 thuận toán ứng với cho 3 giai đoạn chính trong bước xử lí

Về chi tiết thuật toán

- Thuật toán 1: Trích xuất đặc điểm cú pháp lỗ hổng từ chương trình

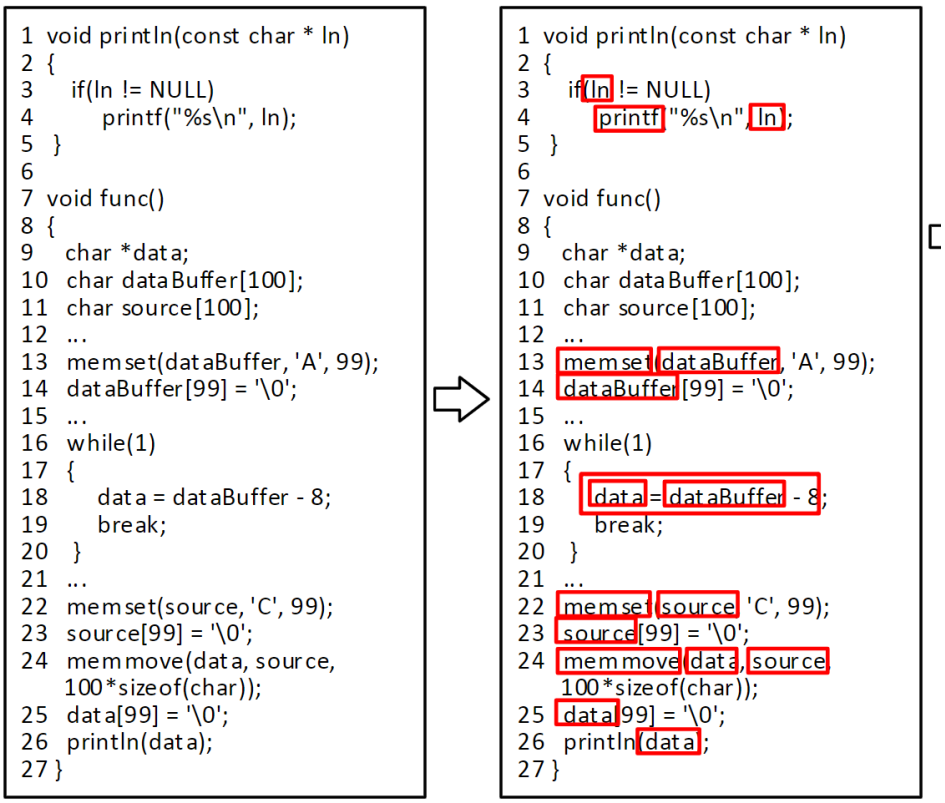
+ Các đặc điểm cú pháp có vai trò xác định các đoạn mã là ứng viên ban đầu của lỗ hổng

**Algorithm 1** Extracting SyVCs from a program

**Input:** A program $P = \{f_1, \ldots, f_\eta\}$; a set $H = \{h_k\}_{1 \le k \le \beta}$ of vulnerability syntax characteristics
**Output:** A set $Y$ of SyVCs

```
1: Y ← ∅;
2: for each function f_i ∈ P do
3:    Generate an abstract syntax tree T_i for f_i;
4:    for each code element e_{i,j,z} in T_i do
5:       for each h_k ∈ H do
6:          if e_{i,j,z} matches h_k then
7:             Y ← Y ∪ {e_{i,j,z}};
8:          end if
9:       end for
10:   end for
11: end for
12: return Y; {the set of SyVCs}
```

Ví dụ mình hoạ:




- Thuật toán 2: Chuyển đổi từ SyVCs sang SeVCs

+ Bước này nhằm cung cấp các câu lệnh có liên quan về mặt ngữ nghĩa với SyVCs từ trên

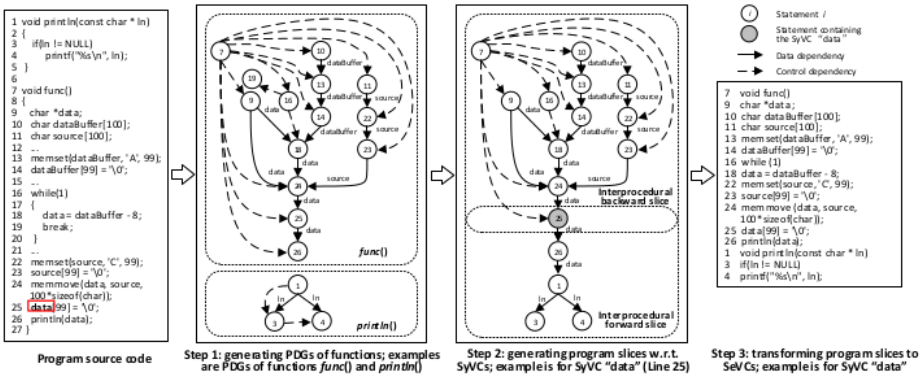
+ Để thực hiện, tác giả sử dụng kỹ thuật cắt lát chương trình. Trong đó, cần dùng Program Dependency Graph (PDG), với yêu cầu dùng data dependency và control dependency (được xác định trên Control Flow Graph CFG)

**Algorithm 2** Transforming SyVCs to SeVCs

**Input:** A program $P = \{f_1, \ldots, f_\eta\}$; a set $Y$ of SyVCs generated by Algorithm 1
**Output:** The set of SeVCs

$C \leftarrow \emptyset$;
**for** each $f_i \in P$ **do**
  Generate a PDG $G'_i = (V_i, E'_i)$ for $f_i$;
**end for**
**for** each $e_{i,j,z} \in Y$ in $G'_i$ **do**
  Generate forward slice $\mathsf{fs}_{i,j,z}$ & backward slice $\mathsf{bs}_{i,j,z}$ of $e_{i,j,z}$;
  Generate interprocedural forward slice $\mathsf{fs}'_{i,j,z}$ by interconnecting $\mathsf{fs}_{i,j,z}$ and the forward slices from the functions called by $f_i$;
  Generate interprocedural backward slice $\mathsf{bs}'_{i,j,z}$ by interconnecting $\mathsf{bs}_{i,j,z}$ and the backward slices from both the functions called by $f_i$ and the functions calling $f_i$;
  Generate program slice $\mathsf{ps}_{i,j,z}$ by connecting $\mathsf{fs}'_{i,j,z}$ and $\mathsf{bs}'_{i,j,z}$ at $e_{i,j,z}$;
  **for** each statement $s_{i,j} \in f_i$ appearing in $\mathsf{ps}_{i,j,z}$ as a node **do**
    $\delta_{i,j,z} \leftarrow \delta_{i,j,z} \cup \{s_{i,j}\}$, according to the order of the appearance of $s_{i,j}$ in $f_i$;
  **end for**
  **for** two statements $s_{i,j} \in f_i$ and $s_{a_p,b_q} \in f_{a_p}$ $(i \neq a_p)$ appearing in $\mathsf{ps}_{i,j,z}$ as nodes **do**
    **if** $f_i$ calls $f_{a_p}$ **then**
      $\delta_{i,j,z} \leftarrow \delta_{i,j,z} \cup \{s_{i,j}, s_{a_p,b_q}\}$, where $s_{i,j} < s_{a_p,b_q}$;
    **else**
      $\delta_{i,j,z} \leftarrow \delta_{i,j,z} \cup \{s_{i,j}, s_{a_p,b_q}\}$, where $s_{i,j} > s_{a_p,b_q}$;
    **end if**
  **end for**
  $C \leftarrow C \cup \{\delta_{i,j,z}\}$;
**end for**
**return** $C$; {the set of SeVCs}

+ Chi tiết quá trình chuyển đổi SyVCs -> SeVC của SyVC ‘data’. Kết quả là SeVC tương ứng với SyVC ‘data’, cụ thể là là tập hợp thứ tự các câu lệnh có liên quan về mặt ngữ nghĩa với SyVC ‘data’.




- Thuật toán 3: Mã hóa SeVCs thành vector

**Algorithm 3** Encoding SeVCs into vectors

**Input:** A set $Y$ of SyVCs generated by Algorithm 1;
a set $C$ of SeVCs corresponding to $Y$ and generated by Algorithm 2;
a threshold $\theta$
**Output:** The set of vectors corresponding to SeVCs

$R \leftarrow \emptyset$;
**for** each $\delta_{i,j,z} \in C$ (corresponding to $e_{i,j,z} \in Y$) **do**
  Remove non-ASCII characters in $\delta_{i,j,z}$;
  Map variable names in $\delta_{i,j,z}$ to symbolic names;
  Map function names in $\delta_{i,j,z}$ to symbolic names;
**end for**
**for** each $\delta_{i,j,z} \in C$ (corresponding to $e_{i,j,z} \in Y$) **do**
  $R_{i,j,z} \leftarrow \emptyset$;
  Divide $\delta_{i,j,z}$ into a set of symbols $S$;
  **for** each $\alpha \in S$ in order **do**
    Transform $\alpha$ to a fixed-length vector $v(\alpha)$;
    $R_{i,j,z} \leftarrow R_{i,j,z} || v(\alpha)$, where $||$ means concatenation;
  **end for**
  **if** $R_{i,j,z}$ is shorter than $\theta$ **then**
    Zeroes are padded to the end of $R_{i,j,z}$;
  **else if** the sub-vector (of $\delta_{i,j,z}$) up to the position of the SyVC $e_{i,j,z}$ is shorter than $\theta/2$ **then**
    Delete the rightmost portion of $R_{i,j,z}$ to make the resulting vector of length $\theta$;
  **else if** the sub-vector (of $\delta_{i,j,z}$) next to the the position of the SyVC $e_{i,j,z}$ is shorter than $\theta/2$ **then**
    Delete the leftmost portion of $R_{i,j,z}$ to make the resulting vector of length $\theta$;
  **else**
    Keep the sub-vector (in $\delta_{i,j,z}$) immediately left to the position of the SyVC of length $\lfloor(\theta-1)/2\rfloor$, the sub-vector corresponding to the SyVC, and the sub-vector immediately right to the position of the SyVC of length $\lceil(\theta-1)/2\rceil$ {the resulting vector has length $\theta$;}
  **end if**
  $R \leftarrow R \cup R_{i,j,z}$;
**end for**
**return** $R$; {the set of vectors corresponding to SeVCs}

- Cuối cùng ta dán nhãn cho vector là có lỗ hổng (1) hoặc không có lỗ hổng (0) để tiến hành learning cho một deep neural network, từ đó nó có thể phát hiện xem SeVCs được cho có chứa lỗ hổng không

## F. Môi trường thực nghiệm của bài báo:

• Cấu hình máy tính: NVIDIA GeForce GTX 1080 GPU và Intel Xeon E5-1620 CP 3.50Hz

• Các công cụ hỗ trợ sẵn có:

+ Checkmarx: giai đoạn extracting SyVCs để phân tích syntax lỗ hổng

• Ngôn ngữ lập trình để hiện thực phương pháp: C/C++

• Đối tượng nghiên cứu (chương trình phần mềm dùng để kiểm tra tính khả thi của phương pháp/tập dữ liệu – nếu có):

Vulnerability dataset gồm 2 nguồn từ:

+ NVD: 19 sản phẩm open-source phổ biến, từ đó thu được 1,591 chương trình C/C++, với 874 là có lỗ hổng

+ SARD: 14,000 chương trình C/C++, với 13,906 là có lỗ hổng

4 phần mềm: Libav, Seamonkey, Thunderbird và Xen

• Tiêu chí đánh giá tính hiệu quả của phương pháp: <vd: số lượng lỗ hổng được phát hiện, thời gian chạy, coverage….>
+ False position rate (FPR): tỉ lệ xác định sai mẫu ko có lỗ hổng thành có lỗ hổng (tỉ lệ cảnh báo sai)
+ False nagative rate (FNR): tỉ lệ xác định sai mẫu có lỗ hổng thành không có lỗ hổng (tỉ lệ không cảnh báo sai)
- + Accuracy (A): tỉ lệ chính xác (số mẫu đoán đúng/toàn bộ)

## G. Kết quả thực nghiệm của bài báo:

Khả năng:

- SySeVR cho phép nhiều loại neural networks phát hiện các loại lỗ hổng bảo mật
- Việc cung cấp thông tin về ngữ nghĩa giúp tăng độ hiệu quả của SySeVR, ví dụ giảm FNR
- Có thể ứng dụng trong các phần mềm thực tế

Ưu:

- Kiểm nghiệm trên 4 phần mềm đã phát hiện được 15 lỗ hổng chưa được báo cáo trong NVD, trong đó 7 cái đã được báo cáo với nhà phát triển, 8 lỗ hổng còn lại đã được lặng lẽ vá lại trong các phiên bản mới
- Thủ nghiệm trên nhiều loại neural network model
- Đạt kết quả tốt hơn mô hình trước đó của tác giả là VulDeePecker và các công cụ phát hiện lỗ hổng hiện có.

Nhược:

- Không cho biết thời gian chạy các giai đoạn, chi phí
- Không so sánh với các phương pháp có cùng đề tài ứng dụng deep learning (VulDeePecker là của tác giả) nên thiếu khách quan
- Coverage của đặc điểm cú pháp được xử lí (4 loại) là 93,6% từ dữ liệu của SARD, và dữ liệu SARD không đại diện được cho toàn bộ phần mềm thực tế
- Thí nghiệm chỉ sử đơn model để phát hiện lỗ hổng

## H. Công việc/tính năng/kỹ thuật mà nhóm thực hiện lập trình và triển khai cho demo:

Các công việc của nhóm:

Đọc hiểu được ý tưởng, thông tin truyền đạt của bài báo.

Setup môi trường cài đặt.

Chạy file chương trình và đánh giá.

Thực hiện implement và phát triển một số hướng còn hạn chế của bài báo.

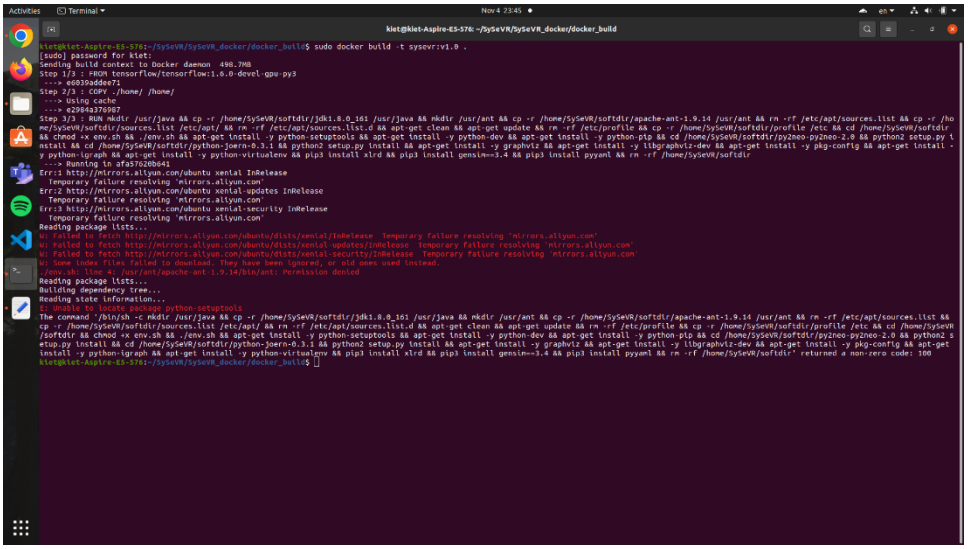
Tổng kết và đánh giá

Công việc tương lai:

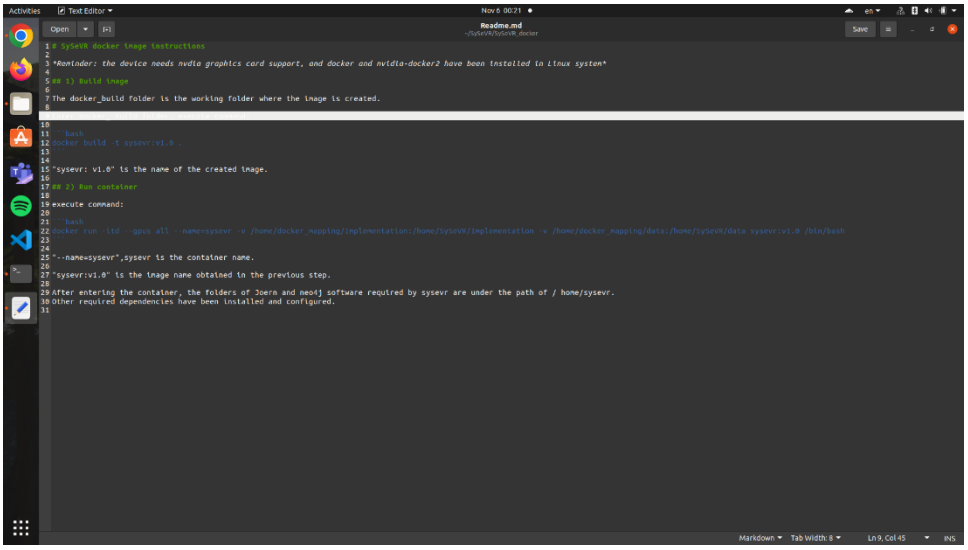
Cải thiện demo, tìm ra hướng mới cho bài báo

## I. Các khó khăn, thách thức hiện tại khi thực hiện:

- Nhóm gặp khó khăn trong đọc hiểu thuật toán
- Thực hiện chạy file từ github của nhóm tác giả có nhiều vấn đề ở cả 2 môi trường window và linux



- Phần code yêu cầu phần cứng hỗ trợ Nvdia



## 3. TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:

| |
|---|
| 100% |

## 4. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

| STT | Công việc | Phân công nhiệm vụ |
|---|---|---|
| 1 | Nghiên cứu và đọc hiểu thuật toán | Nguyễn Bình Thục Trâm<br>Nguyễn Bùi Kim Ngân<br>Võ Anh Kiệt |
| 2 | Hiểu luồng dữ liệu, các thành phần và nhiệm vụ của chúng | Nguyễn Bùi Kim Ngân |
| 3 | Nghiên cứu các thí nghiệm được trình bày nhằm giải quyết các câu hỏi nghiên cứu được đề ra | Nguyễn Bình Thục Trâm<br>Nguyễn Bùi Kim Ngân<br>Võ Anh Kiệt |
| 4 | Triển khai source code, demo | Võ Anh Kiệt<br>Nguyễn Bình Thục Trâm |
| | | |

# BÁO CÁO TỔNG KẾT CHI TIẾT

**Phần bên dưới của báo cáo này là tài liệu báo cáo tổng kết - chi tiết của nhóm thực hiện cho đề tài này.**

*Qui định: Mô tả các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, số liệu thống kê trong bảng biểu, có giải thích)*

## A. Phương pháp thực hiện

*Kiến trúc, thành phần của hệ thống trong bài báo:*

Về tổng quan model :

Bước 1: Thu thập Input

+ Chương trình cho learning (training)

+ Mục tiêu (testing)

Bước 2: Xử lý: qua các giai đoạn sau

+ Xử lý dựa trên Cú pháp SyVCs

+ Xử lý dựa trên Ngữ nghĩa SeVCs từ SyVCs

+ Biểu diễn Vector từ SeVCs

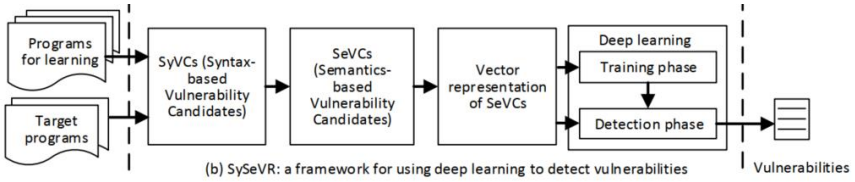
+ Deep Learning:

- Nếu đi qua pha training thì dữ liệu sẽ hỗ trợ cho phần training model

- Nếu đi qua pha phát hiện lỗi thì sẽ tích hợp model đã được training để thực hiện hỗ trợ trong kiểm tra lỗi và đưa ra báo cáo

Bước 3: Xuất Output:

+ Đưa ra báo cáo về các lỗi đã được tìm thấy



(b) SySeVR: a framework for using deep learning to detect vulnerabilities

Model gồm 3 thuận toán ứng với cho 3 giai đoạn chính trong bước xử lí

Về chi tiết thuật toán

- Thuật toán 1: Trích xuất đặc điểm cú pháp lỗ hổng từ chương trình

+ Các đặc điểm cú pháp có vai trò xác định các đoạn mã là ứng viên ban đầu của lỗ hổng

**Algorithm 1** Extracting SyVCs from a program

**Input:** A program $P = \{f_1, \ldots, f_\eta\}$; a set $H = \{h_k\}_{1 \le k \le \beta}$ of vulnerability syntax characteristics
**Output:** A set $Y$ of SyVCs

```
Y ← ∅;
for each function f_i ∈ P do
    Generate an abstract syntax tree T_i for f_i;
    for each code element e_{i,j,z} in T_i do
        for each h_k ∈ H do
            if e_{i,j,z} matches h_k then
                Y ← Y ∪ {e_{i,j,z}};
            end if
        end for
    end for
end for
return Y; {the set of SyVCs}
```

Ví dụ mình hoạ:

```
1  void println(const char * ln)
2  {
3     if(ln != NULL)
4        printf("%s\n", ln);
5  }
6
7  void func()
8  {
9     char *data;
10    char dataBuffer[100];
11    char source[100];
12    ...
13    memset(dataBuffer, 'A', 99);
14    dataBuffer[99] = '\0';
15    ...
16    while(1)
17    {
18       data = dataBuffer - 8;
19       break;
20    }
21    ...
22    memset(source, 'C', 99);
23    source[99] = '\0';
24    memmove(data, source,
      100*sizeof(char));
25    data[99] = '\0';
26    println(data);
27 }
```

Program source code

```
1  void println(const char * ln)
2  {
3     if(ln != NULL)
4        printf("%s\n", ln);
5  }
6
7  void func()
8  {
9     char *data;
10    char dataBuffer[100];
11    char source[100];
12    ...
13    memset(dataBuffer, 'A', 99);
14    dataBuffer[99] = '\0';
15    ...
16    while(1)
17    {
18       data = dataBuffer - 8;
19       break;
20    }
21    ...
22    memset(source, 'C', 99);
23    source[99] = '\0';
24    memmove(data, source,
      100*sizeof(char));
25    data[99] = '\0';
26    println(data);
27 }
```

SyVCs (highlighted by boxes)

- Thuật toán 2: Chuyển đổi từ SyVCs sang SeVCs

+ Bước này nhằm cung cấp các câu lệnh có liên quan về mặt ngữ nghĩa với SyVCs từ trên

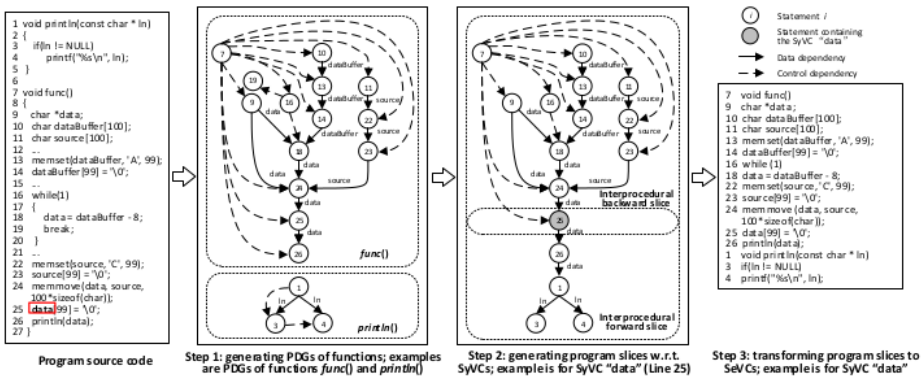
+ Để thực hiện, tác giả sử dụng kỹ thuật cắt lát chương trình. Trong đó, cần dùng Program Dependency Graph (PDG), với yêu cầu dùng data dependency và control dependency (được xác định trên Control Flow Graph CFG)

**Algorithm 2** Transforming SyVCs to SeVCs

**Input:** A program $P = \{f_1, \ldots, f_\eta\}$;
a set $Y$ of SyVCs generated by Algorithm 1
**Output:** The set of SeVCs

1: $C \leftarrow \emptyset$;
2: **for** each $f_i \in P$ **do**
3: Generate a PDG $G'_i = (V_i, E'_i)$ for $f_i$;
4: **end for**
5: **for** each $e_{i,j,z} \in Y$ in $G'_i$ **do**
6: Generate forward slice $\mathrm{fs}_{i,j,z}$ & backward slice $\mathrm{bs}_{i,j,z}$ of $e_{i,j,z}$;
7: Generate interprocedural forward slice $\mathrm{fs}'_{i,j,z}$ by interconnecting $\mathrm{fs}_{i,j,z}$ and the forward slices from the functions called by $f_i$;
8: Generate interprocedural backward slice $\mathrm{bs}'_{i,j,z}$ by interconnecting $\mathrm{bs}_{i,j,z}$ and the backward slices from both the functions called by $f_i$ and the functions calling $f_i$;
9: Generate program slice $\mathrm{ps}_{i,j,z}$ by connecting $\mathrm{fs}'_{i,j,z}$ and $\mathrm{bs}'_{i,j,z}$ at $e_{i,j,z}$;
10: **for** each statement $s_{i,j} \in f_i$ appearing in $\mathrm{ps}_{i,j,z}$ as a node **do**
11: $\delta_{i,j,z} \leftarrow \delta_{i,j,z} \cup \{s_{i,j}\}$, according to the order of the appearance of $s_{i,j}$ in $f_i$;
12: **end for**
13: **for** two statements $s_{i,j} \in f_i$ and $s_{a_p,b_q} \in f_{a_p}$ $(i \neq a_p)$ appearing in $\mathrm{ps}_{i,j,z}$ as nodes **do**
14: **if** $f_i$ calls $f_{a_p}$ **then**
15: $\delta_{i,j,z} \leftarrow \delta_{i,j,z} \cup \{s_{i,j}, s_{a_p,b_q}\}$, where $s_{i,j} < s_{a_p,b_q}$;
16: **else**
17: $\delta_{i,j,z} \leftarrow \delta_{i,j,z} \cup \{s_{i,j}, s_{a_p,b_q}\}$, where $s_{i,j} > s_{a_p,b_q}$;
18: **end if**
19: **end for**
20: $C \leftarrow C \cup \{\delta_{i,j,z}\}$;
21: **end for**
22: **return** $C$; {the set of SeVCs}

+ Chi tiết quá trình chuyển đổi SyVCs -> SeVC của SyVC 'data'. Kết quả là SeVC tương ứng với SyVC 'data', cụ thể là là tập hợp thứ tự các câu lệnh có liên quan về mặt ngữ nghĩa với SyVC 'data'.



- Thuật toán 3: Mã hóa SeVCs thành vector

**Algorithm 3** Encoding SeVCs into vectors

**Input:** A set $Y$ of SyVCs generated by Algorithm 1; a set $C$ of SeVCs corresponding to $Y$ and generated by Algorithm 2; a threshold $\theta$

**Output:** The set of vectors corresponding to SeVCs

$R \leftarrow \emptyset$;
**for** each $\delta_{i,j,z} \in C$ (corresponding to $e_{i,j,z} \in Y$) **do**
  Remove non-ASCII characters in $\delta_{i,j,z}$;
  Map variable names in $\delta_{i,j,z}$ to symbolic names;
  Map function names in $\delta_{i,j,z}$ to symbolic names;
**end for**
**for** each $\delta_{i,j,z} \in C$ (corresponding to $e_{i,j,z} \in Y$) **do**
  $R_{i,j,z} \leftarrow \emptyset$;
  Divide $\delta_{i,j,z}$ into a set of symbols $S$;
  **for** each $\alpha \in S$ in order **do**
    Transform $\alpha$ to a fixed-length vector $v(\alpha)$;
    $R_{i,j,z} \leftarrow R_{i,j,z} || v(\alpha)$, where $||$ means concatenation;
  **end for**
  **if** $R_{i,j,z}$ is shorter than $\theta$ **then**
    Zeroes are padded to the end of $R_{i,j,z}$;
  **else if** the sub-vector (of $\delta_{i,j,z}$) up to the position of the SyVC $e_{i,j,z}$ is shorter than $\theta/2$ **then**
    Delete the rightmost portion of $R_{i,j,z}$ to make the resulting vector of length $\theta$;
  **else if** the sub-vector (of $\delta_{i,j,z}$) next to the the position of the SyVC $e_{i,j,z}$ is shorter than $\theta/2$ **then**
    Delete the leftmost portion of $R_{i,j,z}$ to make the resulting vector of length $\theta$;
  **else**
    Keep the sub-vector (in $\delta_{i,j,z}$) immediately left to the position of the SyVC of length $\lfloor(\theta-1)/2\rfloor$, the sub-vector corresponding to the SyVC, and the sub-vector immediately right to the position of the SyVC of length $\lceil(\theta-1)/2\rceil$ {the resulting vector has length $\theta$;}
  **end if**
  $R \leftarrow R \cup R_{i,j,z}$;
**end for**
**return** $R$; {the set of vectors corresponding to SeVCs}

Cuối cùng ta dán nhãn cho vector là có lỗ hổng (1) hoặc không có lỗ hổng (0) để tiến hành learning cho một deep neural network, từ đó nó có thể phát hiện xem SeVCs được cho có chứa lỗ hổng không

*Kiến trúc, thành phần nhóm đã thực hiện:*

- Thuật toán 1: Trích xuất đặc điểm cú pháp lỗ hổng từ chương trình

Bước 1: Đầu tiên sử dụng joern để phân tích mã nguồn: đầu vào là tệp mã nguồn và đầu ra là tệp có tên .joernIndex.

Bước 2: Tiếp theo sử dụng code get_cfg_relation.py để lấy đồ thị hàm CFG bằng công cụ joern. Đầu vào là kết quả của bước 1 và đầu ra được lưu trữ với các thư mục trong cfg_db

Bước 3: Thực hiện để lấy đồ thị hàm PDG. Đầu vào là các tệp trong cfg_db và đầu ra được lưu trữ với các thư mục trong pdg_db

Bước 4: Thực hiện để lấy đồ thị gọi hàm. Đầu vào là các tệp trong pdg_db và đầu ra được lưu trữ với các thư mục trong dict_call2cfgNodeID_funcID

Bước 5: Thực hiện để nhận bốn loại SyVC. Đầu vào là các tệp trong dict_call2cfgNodeID_funcID và đầu ra là bốn loại SyVC.

Bước 6: Thực hiện để trích xuất các lát. Đầu vào là các tệp được tạo bởi point_get.py và đầu ra là các tệp lát

Bước 7: Thực hiện để lấy số dòng của các dòng dễ bị tấn công trong bộ dữ liệu nvd. Đầu vào là tệp mã nguồn hoặc tệp func và tệp khác

Bước 8: Thực hiện để trích xuất số dòng của các dòng dễ bị tấn công từ SARD_testcaseinfo.xml. "000" là tệp mã nguồn. Đầu ra là SARD_testcaseinfo.txt, sau đó được đổi tên thành contain_all.txt

Bước 9: Tạo nhãn NVD và SARD

Bước 10: Thực hiện để add label vào slide

**B. Chi tiết cài đặt, hiện thực**

*Cài đặt, lập trình trên máy tính, cấu hình máy tính sử dụng, chuẩn bị dữ liệu:*

Môi trường:

Hệ điều hành: Ubuntu 18.04

Môi trường cho thuật toán 1:

Joern 0.3.1:

+ JDK 1.7

+ Neo4J 2.1.8 Community Edition

+ Gremlin for Neo4J 2.X

+ Apache Ant build tool

Python 2.7

Môi trường cho thuật toán 2:

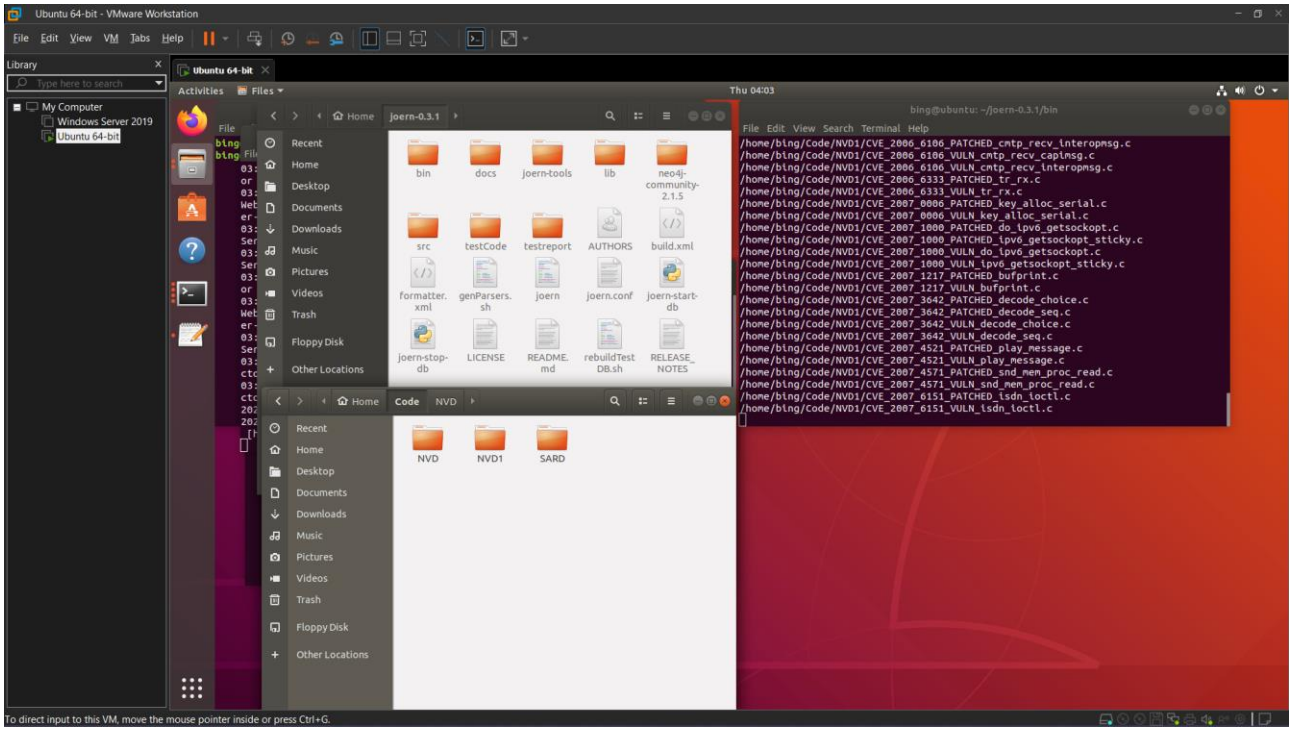
Python 3.6

Tensorflow 1.6

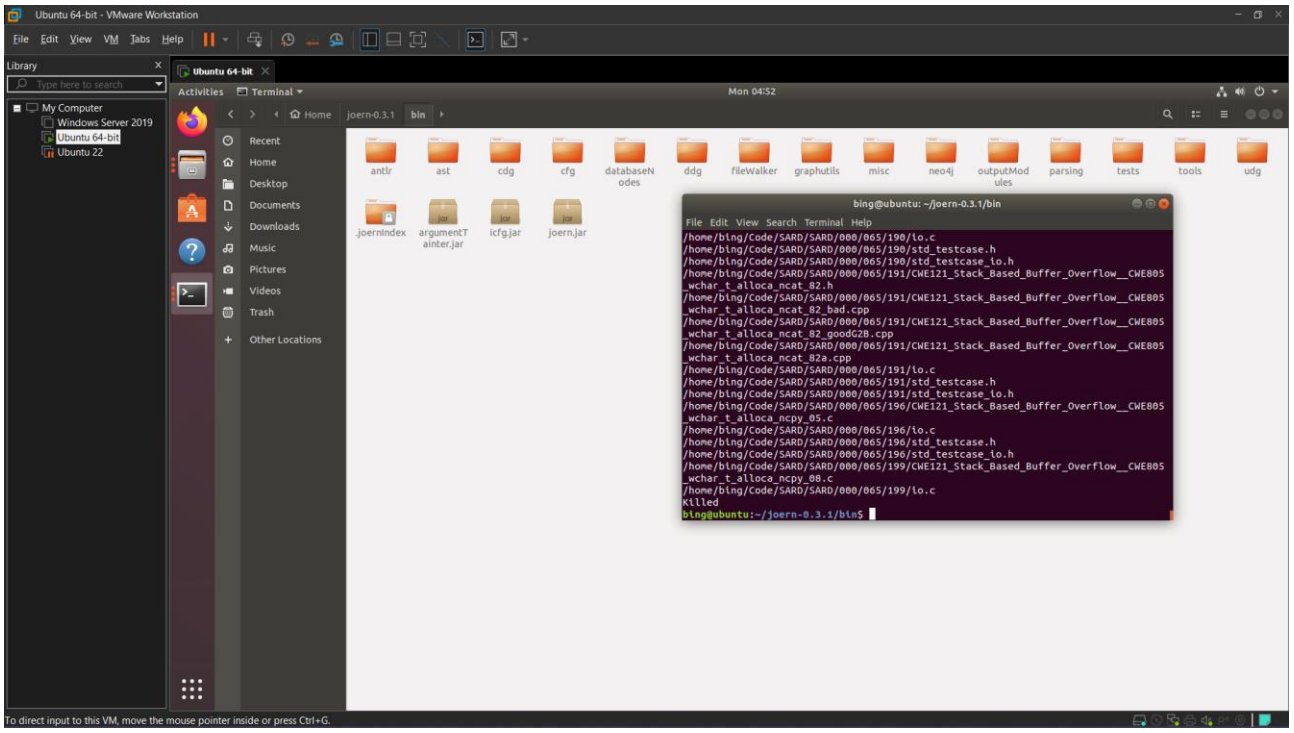
Gensim 3.4

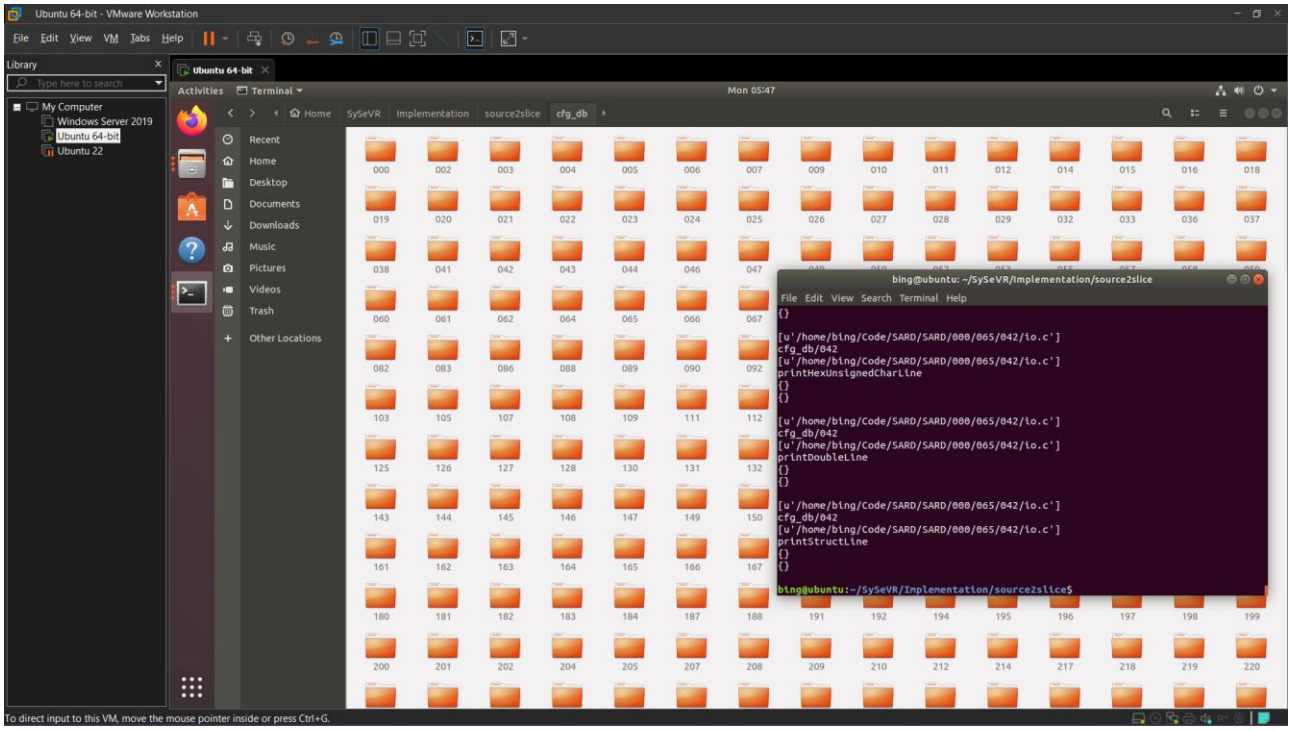
**C. Kết quả thực nghiệm**

Step 1 bước 1


Ubuntu 64-bit - VMware Workstation
/home/bing/Code/NVD1/CVE_2006_6106_PATCHED_cmtp_recv_interopmsg.c
/home/bing/Code/NVD1/CVE_2006_6106_VULN_cmtp_recv_capimsg.c
/home/bing/Code/NVD1/CVE_2006_6106_VULN_cmtp_recv_interopmsg.c
/home/bing/Code/NVD1/CVE_2006_6333_PATCHED_tr_rx.c
/home/bing/Code/NVD1/CVE_2006_6333_VULN_tr_rx.c
/home/bing/Code/NVD1/CVE_2007_0006_PATCHED_key_alloc_serial.c
/home/bing/Code/NVD1/CVE_2007_0006_VULN_key_alloc_serial.c
/home/bing/Code/NVD1/CVE_2007_1000_PATCHED_do_ipv6_getsockopt.c
/home/bing/Code/NVD1/CVE_2007_1000_PATCHED_ipv6_getsockopt_sticky.c
/home/bing/Code/NVD1/CVE_2007_1000_VULN_do_ipv6_getsockopt.c
/home/bing/Code/NVD1/CVE_2007_1000_VULN_ipv6_getsockopt_sticky.c
/home/bing/Code/NVD1/CVE_2007_1217_PATCHED_bufprint.c
/home/bing/Code/NVD1/CVE_2007_1217_VULN_bufprint.c
/home/bing/Code/NVD1/CVE_2007_3642_PATCHED_decode_choice.c
/home/bing/Code/NVD1/CVE_2007_3642_PATCHED_decode_seq.c
/home/bing/Code/NVD1/CVE_2007_3642_VULN_decode_choice.c
/home/bing/Code/NVD1/CVE_2007_3642_VULN_decode_seq.c
/home/bing/Code/NVD1/CVE_2007_4521_PATCHED_play_message.c
/home/bing/Code/NVD1/CVE_2007_4521_VULN_play_message.c
/home/bing/Code/NVD1/CVE_2007_4571_PATCHED_snd_mem_proc_read.c
/home/bing/Code/NVD1/CVE_2007_4571_VULN_snd_mem_proc_read.c
/home/bing/Code/NVD1/CVE_2007_6151_PATCHED_isdn_ioctl.c
/home/bing/Code/NVD1/CVE_2007_6151_VULN_isdn_ioctl.c

Step 1 bước 2


Ubuntu 64-bit - VMware Workstation
/home/bing/Code/SARD/SARD/000/065/190/lo.c
/home/bing/Code/SARD/SARD/000/065/190/std_testcase.h
/home/bing/Code/SARD/SARD/000/065/190/std_testcase_io.h
/home/bing/Code/SARD/SARD/000/065/191/CWE121_Stack_Based_Buffer_Overflow__CWE805_wchar_t_alloca_ncat_82.h
/home/bing/Code/SARD/SARD/000/065/191/CWE121_Stack_Based_Buffer_Overflow__CWE805_wchar_t_alloca_ncat_82_bad.cpp
/home/bing/Code/SARD/SARD/000/065/191/CWE121_Stack_Based_Buffer_Overflow__CWE805_wchar_t_alloca_ncat_82_goodG2B.cpp
/home/bing/Code/SARD/SARD/000/065/191/CWE121_Stack_Based_Buffer_Overflow__CWE805_wchar_t_alloca_ncat_82a.cpp
/home/bing/Code/SARD/SARD/000/065/191/lo.c
/home/bing/Code/SARD/SARD/000/065/191/std_testcase.h
/home/bing/Code/SARD/SARD/000/065/191/std_testcase_io.h
/home/bing/Code/SARD/SARD/000/065/196/CWE121_Stack_Based_Buffer_Overflow__CWE805_wchar_t_alloca_ncpy_05.c
/home/bing/Code/SARD/SARD/000/065/196/lo.c
/home/bing/Code/SARD/SARD/000/065/196/std_testcase.h
/home/bing/Code/SARD/SARD/000/065/196/std_testcase_io.h
/home/bing/Code/SARD/SARD/000/065/199/CWE121_Stack_Based_Buffer_Overflow__CWE805_wchar_t_alloca_ncpy_08.c
/home/bing/Code/SARD/SARD/000/065/199/lo.c
Killed
bing@ubuntu:~/joern-0.3.1/bin$



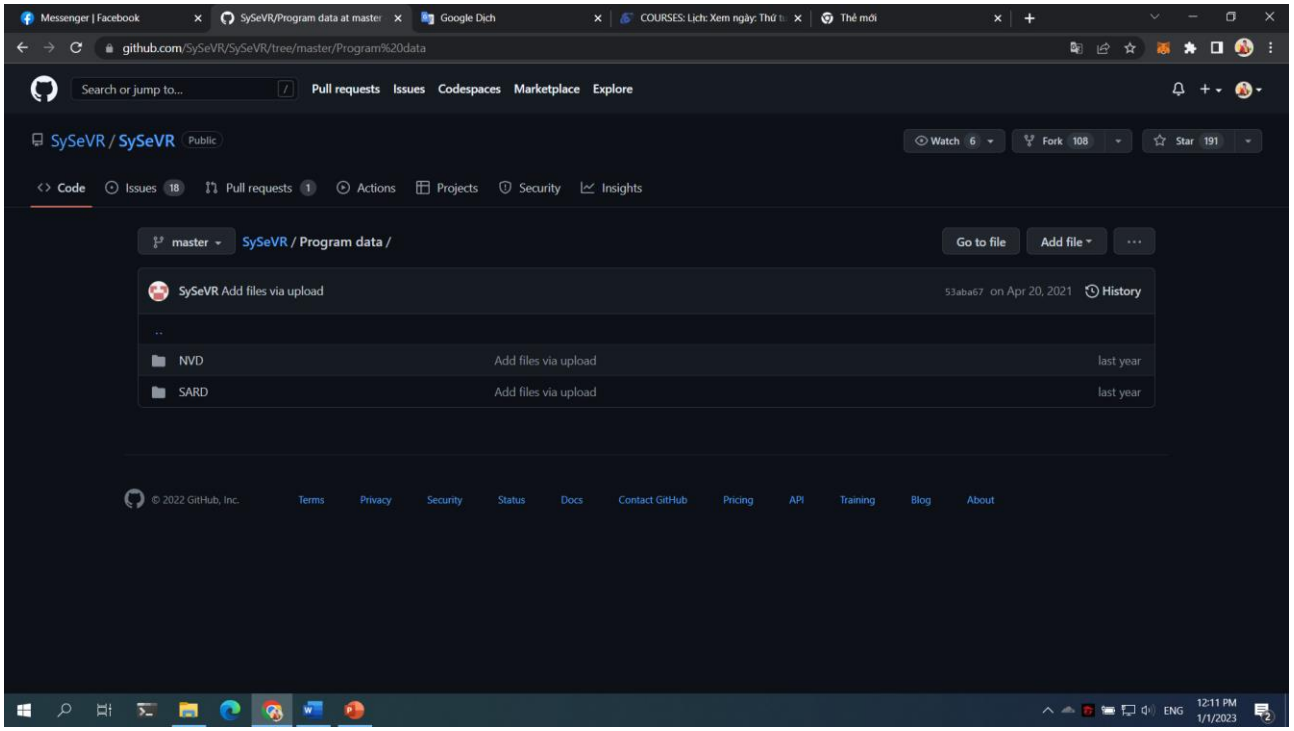

Step 1 bước 3


Step 1 bước 4


Step 1 bước 5


Step 1 bước 6 7

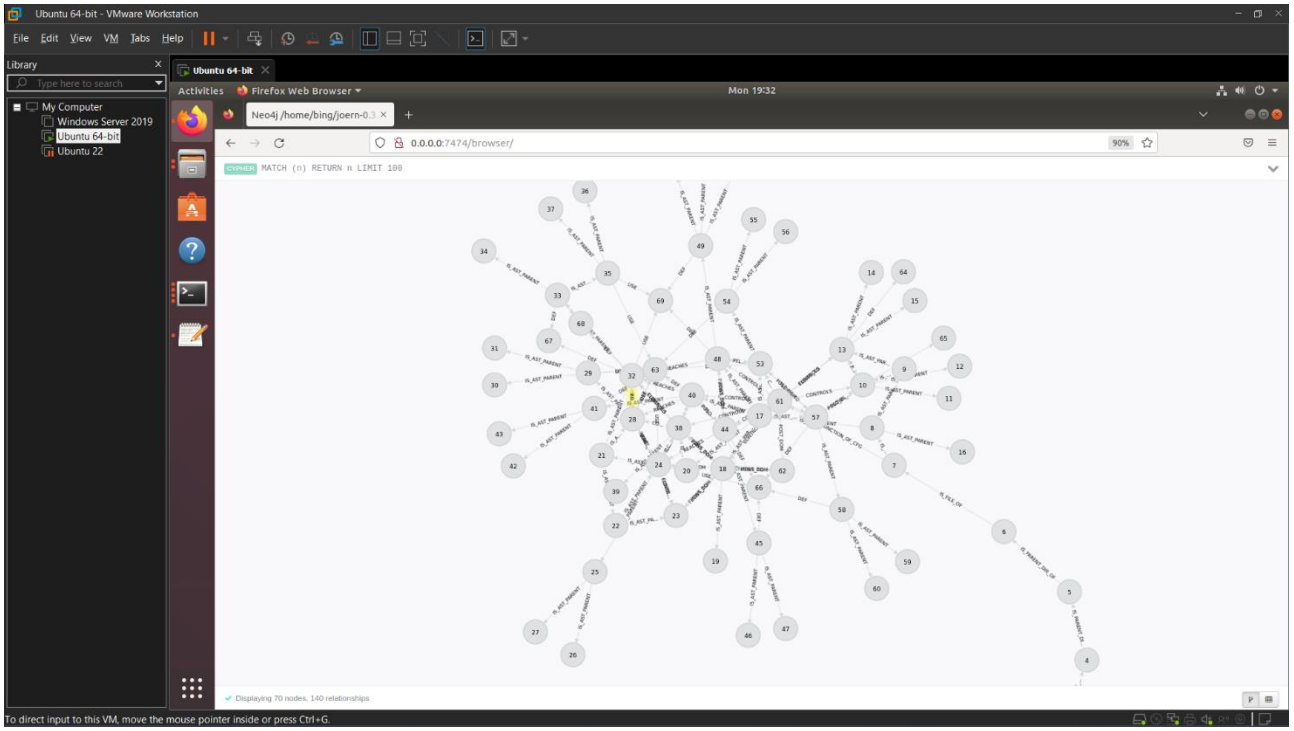

Step 1 bước 8 sử dụng lại kết quả của bài




Step 1 bước 9


Step 1 bước 10


Check database




## D. Hướng phát triển

Đề tài có nhiều tiềm năng phát triển trong tương lai khi việc thực hiện tự động hóa quá trình thực hiện tìm kiếm lỗ hỏng phần mềm, có tính ứng dụng thực tiễn cao trong thực tế. Có thể phát triển đồ án để tiếp tục thực hiện đồ án chuyên ngành hoặc khóa luận tốt nghiệp

*Sinh viên báo cáo các nội dung mà nhóm đã thực hiện, có thể là 1 phần hoặc toàn bộ nội dung của bài báo. Nếu nội dung thực hiện có khác biệt với bài báo (như cấu hình, tập dữ liệu, kết quả,...), sinh viên cần chỉ rõ thêm khác biệt đó và nguyên nhân.*

---

***Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này***

# YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

**Báo cáo:**

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- Đặt tên theo định dạng: [Mã lớp]-Project_Final_NhomX_Madetai. (trong đó X và Madetai là mã số thứ tự nhóm và Mã đề tài trong danh sách đăng ký nhóm đồ án).

  *Ví dụ: [NT521.N11.ANTT]-Project_Final_Nhom03_CK01.*
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

**Đánh giá**:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

*Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**