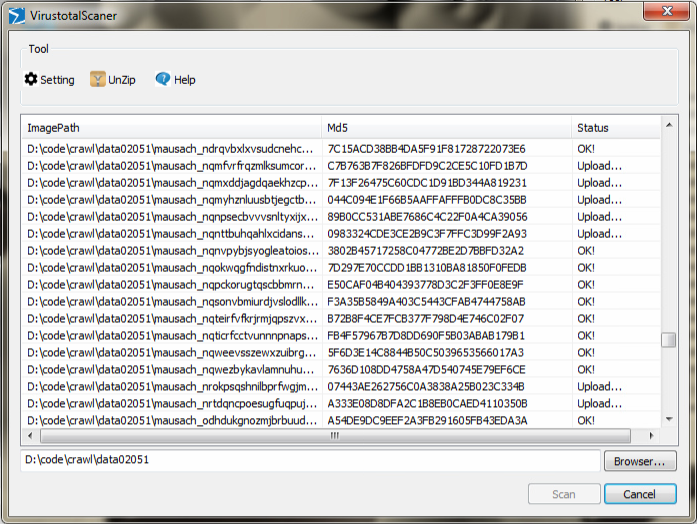
# Mô tả chi tiết các công cụ sử dụng

## VirusTotal Scanner

Yêu cầu: Tập mẫu sạch được crawl từ internet về

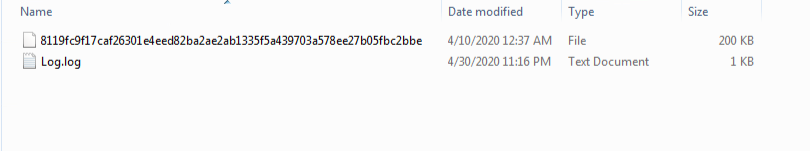
Quy trình: Lựa chọn tập mẫu đầu vào, ấn nút Scan trên giao diện Tool và đợi kết quả trả về

Giao diện Chính:

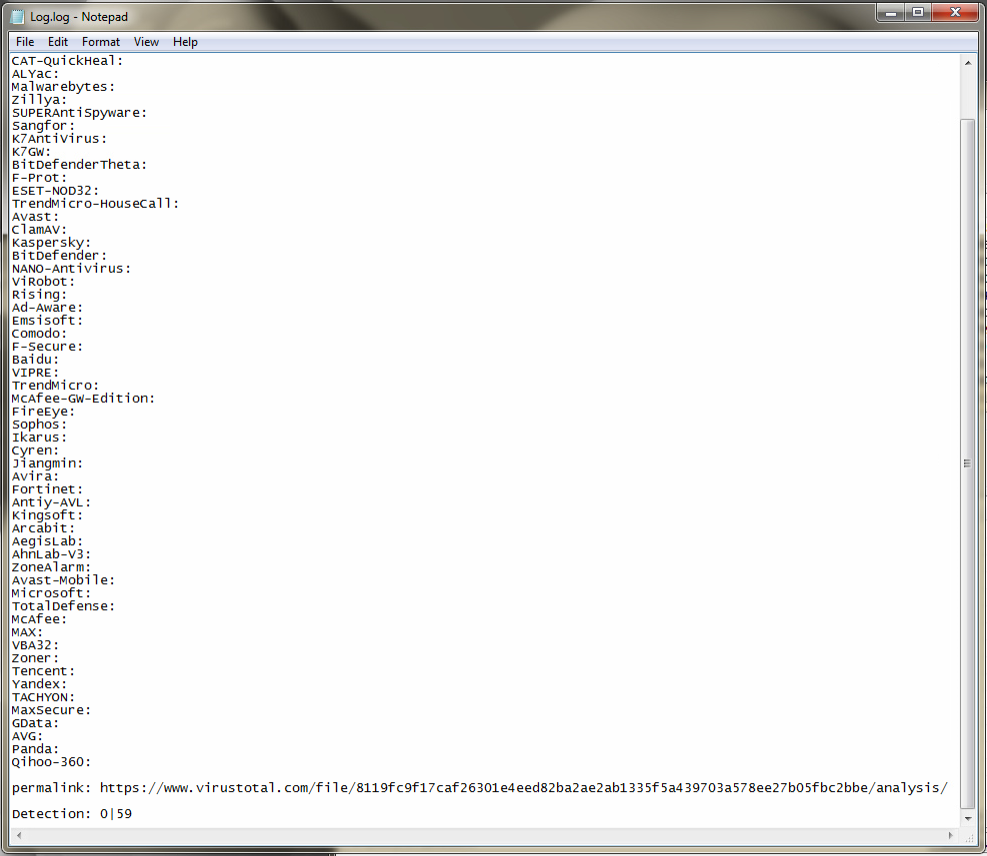


Kết quả trả về: Trong thư mục input đầu vào, các mẫu up lên xong, khi chuyển sang trạng thái OK! sẽ được phân chia vào các thư mục có tên tương ứng với MD5 của mẫu. Trong thư mục này sẽ chứa 2 file: File thứ nhất là file đầu vào được quét, File thứ 2 là file Log.log trả về chứa thông tin kết quả quét của các engine Antivirus trên VirusTotal.

Nội dung kết quả trả về:



Nội dung file Log.log



Nếu kết quả Detection: khác 0 thì nghĩa là mẫu được up lên (tạm thời được coi) là virus. Còn nếu là File Sạch thì sẽ trả về 0/<số engine quét>

## Kiểm tra log trả về từ VirusTotal

Công cụ: ***CheckLogVirusTotalSample.py***

Yêu cầu: Tập mẫu sạch được crawl từ internet về được up lên VirusTotal để Check Virus. Nhận được file Log trả về chứa kết quả Check

Mô tả về file Log trả về và Tool Up mẫu lên VirusTotal được miêu tả trong phần Tool VirusTotal Scanner

Quy trình: một tập mẫu HTML hoặc JavaScript được crawl về từ internet, chưa được kiểm tra xem có phải là virus hay không. Tập mẫu được crawl về sẽ sử dụng công cụ VirusTotal Scanner để up tập mẫu lên VirusTotal để tiến hành quét file. Kết quả thu được sẽ là một folder có tên là MD5 của file được up, trong đó có file gốc và file “Log.log”. Tập result thu được làm input đầu vào cho câu lệnh thực thi file python CheckLogVirusTotalSample.py để kiểm tra File

Kết quả trả về: Trong cùng một thư mục cha được cho làm input đầu vào sẽ sinh ra 2 thư mục con: mau\_sach (chứa tất cả các mẫu sạch được VirusTotal trả về) và mau\_virus (chứa các mẫu mà log trả về từ VirusTotal có nội dung Detection khác 0)

## Kiểm tra mẫu trùng

Công cụ: ***CheckDuplicateMD5In2Folder.py*** và ***CheckDuplicateMD5InFolder.py***

Quy trình: 2 Tool này được sử dụng trong bước kiểm tra các tập text làm input đầu vào trong mạng CNN.

* Với Tool CheckDuplicateMD5In2Folder sẽ check trong 2 thư mục, và copy tất cả những mẫu không có trong thư mục 1 ở thư mục 2 vào một thư mục res được tạo trong thư mục 2.
* Với Tool CheckDuplicateMD5InFolder sẽ check trong 1 thư mục và lọc ra tất cả các mẫu (virus hoặc file sạch) có MD5 khác nhau từng đôi một

Mô tả: 2 Tool viết bằng Python kiểm tra các File có MD5 trùng nhau. Mục đích là chỉ lấy tập những file không trùng nhau làm tập train hoặc test cho mô hình

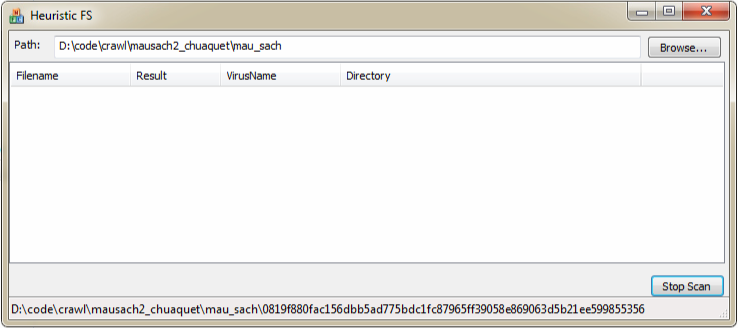
## Tool Text Extraction

Công cụ: HeuristicFS và EmulJS

Mô tả:

* Công cụ HeuristicFS cung cấp giao diện cho dll EmulJS có tác dụng phân tích file mã nguồn đầu vào và xuất ra file text có chứa mã nguồn JS được chuẩn hóa. File text này sẽ được sử dụng làm đầu vào cho mô hình CNN
* EmulJS giống như trình giả lập của JavaScript, nó được xây dựng để thực thi một đoạn code JavaScript giống như JavaScript Engine. Tuy nhiên, trong quá trình thực thi sẽ kiểm tra với một số luật đã được xây dựng từ đó kết luận có là virus hay không.

Giao diện Heuristic FS



Chỉnh sửa công cụ EmulJS để xử lý tĩnh đoạn code JS. Lợi dụng việc EmulJS xử lý từng câu lệnh 1 thì luồng xử lý sẽ là cho nó chạy từng câu lệnh, ở mỗi câu lệnh thay vì thực thi câu lệnh đó thì lưu trữ câu lệnh đọc được ra file chuẩn hóa. Đồng thời, trong khi thực thi sẽ chuẩn hóa lại một số giá trị được qui định dưới đây

## Quá trình phân tích mẫu js và chuẩn hóa

**Chuẩn hóa trong EmulJS**

Từ khóa: ***new, if, else, for, in, function, do, while, var, return, try, catch, finally, throw, break, continue, delete,*** được giữ nguyên

Các dấu ngoặc: ***{ }, ( ), [ ]*** thì giữ nguyên

Các toán tử thì giữ nguyên: ***=, += , -=, |= , ^=, &=, <<=, >>=, &, |, ^, &&, ||, ==, != , ===, !==, <=, >=, <, >, >>, <<, >>>, +, -, \*, /, %, ++, --, ~, !, typeof()***

Dấu điều kiện ***‘?’, ‘:’*** được giữ nguyên

Các giá trị đặc biệt: ***true, false, null, undefined*** được giữ nguyên

Các từ đặc biệt: ***this, self, length*** được giữ nguyên

Khai báo ***regex*** được giữ nguyên

* Sửa lại phần tên biến trong hàm: không reset index của tên biến

Tên biến:

* các biến chuẩn hóa thành ***bien1, bien2, bien3***,…
* các biến hàm chuẩn hóa thành ***bien\_ham1, bien\_ham2, bien\_ham3***
* biến sau câu lệnh catch chuẩn hóa thành ***bien\_catch***
* Các tên biến và tên biến chuẩn hóa được lưu vào stack để đối chiếu cho những câu lệnh tiếp theo

Tên hàm:

* Hàm có tên chuẩn hóa thành ***ham1, ham2, ham3, …***
* Hàm không có tên chuẩn hóa thành ***ham\_noname1, ham\_noname2, …***
* Các tên hàm được chuẩn hóa và tên hàm thực được lưu vào stack tương ứng để đối chiếu với nhau

Gọi hàm sử dụng dấu ***‘.’*** Thì giữ nguyên, hoặc dấu ngoặc ***[“ten\_ham\_api”]*** thì chuẩn hóa thành ***[“string”] ví dụ window[“document”][“write”] chuẩn hóa thành window[“string”][“string”].***

* ***Làm sao phải giữ nguyên được window[“document”][“write”] -> tìm cách???***

Tên các object được khởi tạo từ đầu: ***window, String, Array, Object***, .. và một số **hàm api** được giữ nguyên tên.

***Gọi hàm API nhưng chưa được định nghĩa trong danh sách ban đầu hoặc cũng không định nghĩa tại danh sách các hàm người dùng đã khai báo: ghi tên hàm ra và không chuẩn hóa.***

Tham số hàm chuẩn hóa thành: ***thamso1, thamso2, thamso3, …***

* ***Tham số cũng không được reset***

Giá trị kiểu ***string***, chuẩn hóa thành ***“string”***

Giá trị kiểu ***int***, chuẩn hóa thành ***int***

Giá trị kiểu ***float***, chuẩn hóa thành ***float***

***Khai báo mảng dạng : tenmang = [1, 2, 3, …] thì chuẩn hóa thành bien1 = [int, int, int, …], tương tự với kiểu string***

* Sửa lại cấu trúc khai báo biến bien1 = [so\_luong:kieu\_du\_lieu]

Dấu chấm phẩy: ***‘;’*** ngăn cách trong lệnh for, ***‘;’*** kết thúc lệnh tạm thời không lấy

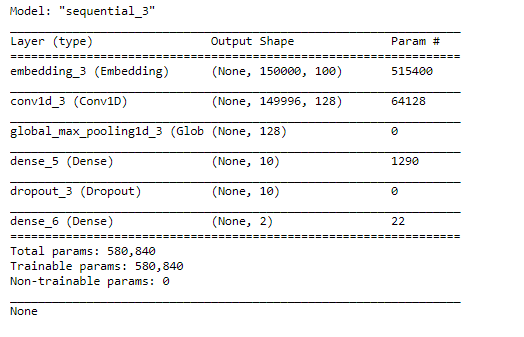
* Thêm dấu chấm phẩy vào

Các đối tượng không được khai báo trước đó, không có trong stack khởi tạo trước đó, chuẩn hóa thành: ***\_Undefined\_Object -> chuyển tên ngắn đi: \_UO***

* Nội dung này được thực hiện trên **EmulJS**
* **Đường dẫn link Github chứa mã nguồn và document:** <https://github.com/wnos278/DoAnTotNghiep>

## Tuần này

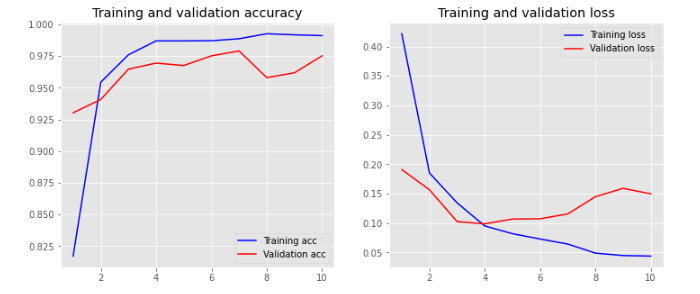
* Xử lý dữ liệu:
* Hoàn thành sửa code EmulJS để trả ra file text chuẩn hóa được check trùng nhau theo MD5, các mẫu là khác nhau.
* Train thử với Mô hình CNN xử lý ngôn ngữ
* Thông tin cấu trúc mạng CNN:



* + Số chiều input đầu vào: 5153
  + Số chiều input đầu ra: 100
  + Độ dài mỗi input: 50000



* Đồ thị hóa quá trình huấn luyện:



* Kết quả đạt được:
  + tổng số mẫu train: 5222, trong đó 2500 mẫu sạch, 2722 mẫu virus
  + kết quả train mô hình cnn với 15 epoc
  + accuracy 98,4%, loss 4,42%
  + validate 97,71%, loss 14,41%
  + tập test: 1531 mẫu sạch: mô hình nhận diện được 1371 mẫu sạch: 89,54%
  + tập test2: 2122 mẫu virus: mô hình nhận diện được 1994 mẫu virus: 93.96%
* Khó khăn: Hiện tại do số lượng các mẫu sau khi chuẩn hóa lại bị trùng MD5 nhiều, do vậy đang tìm cách để khắc phục