

**HỌC VIỆN KỸ THUẬT MẬT MÃ**

**------------------------------**

****

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC**

***ĐỀ TÀI:***

**ỨNG DỤNG DEEP LEARNING TRONG BÀI TOÁN PHÁT HIỆN MÃ ĐỘC**

**Giảng viên hướng dẫn: Trần Anh Tú**

**Sinh viên thực hiện:**

* **Trần Hùng Mạnh – AT15A**
* **Ngô Quang Sang – CT3C**

# MỤC LỤC

[MỤC LỤC 3](#_Toc5967848)

[MỤC LỤC HÌNH ẢNH 4](#_Toc5967849)

[BẢNG CHỮ VIẾT TẮT 5](#_Toc5967850)

[I. Tổng quan về mã độc 6](#_Toc5967851)

[1. Phân loại và nguyên tắc của phần mềm độc hại 6](#_Toc5967852)

[1.1. Xâm nhập và chiếm điều khiển từ xa của hệ thống đích 6](#_Toc5967853)

[1.1.1. Exploit ( Chương trình khai thác) 6](#_Toc5967854)

[1.1.2. Trojan Horse (Ngựa thành troy) 7](#_Toc5967855)

[1.2. Duy trì quyền điều khiển từ xa cho hệ thống đích 12](#_Toc5967856)

[1.2.1. Backdoor (cửa sau) 12](#_Toc5967857)

[1.2.2. Rootkit 12](#_Toc5967858)

[1.3. Thực hiện mục đích cụ thể trên hệ thống đích 13](#_Toc5967859)

[2. Quan hệ giữa hai loại mã độc hại và các cơ chế hoạt động 14](#_Toc5967860)

[3. Bài toán phát hiện mã độc 14](#_Toc5967861)

[4. Phân tích cấu trúc định dạng tệp thực thi PE (Portable Execute) 15](#_Toc5967862)

[4.1. PE File 15](#_Toc5967863)

[4.2. Section Names và Characteristics 17](#_Toc5967864)

[4.3. Section Entropy 18](#_Toc5967865)

[4.4. Các PE Import 18](#_Toc5967866)

[II. Giới thiệu Deep Learning (Học sâu) 20](#_Toc5967867)

[1. Tổng quan về Deep Learning 20](#_Toc5967868)

[1.1. Deep Learning là gì? 20](#_Toc5967869)

[1.2. Các khái niệm cơ bản của Deep Learning 21](#_Toc5967870)

[2. Mô hình thuật toán Deep Learning 23](#_Toc5967871)

[2.1. Đặc điểm chung của các thuật toán deep learning 23](#_Toc5967872)

[2.2. Một số mô hình Deep Learning cơ bản 26](#_Toc5967873)

[III. Ứng dụng Deep Learning vào bài toán phát hiện mã độc 29](#_Toc5967874)

[1. Giới thiệu tổng quát 29](#_Toc5967875)

[2. Sản phẩm “Ứng dụng Deep learning vào bài toán phát hiện mã độc từ file thực thi trên hệ điều hành Windows” 29](#_Toc5967876)

[2.1. Hacker, tin tặc và hệ điều hành Windows, định dạng Portable Execute (PE - EXE) 29](#_Toc5967877)

[2.2. Giải pháp phát hiện mã độc từ tệp thực thi PE 30](#_Toc5967878)

[2.3. Các công cụ sử dụng 30](#_Toc5967879)

[2.4. Phương pháp thực hiện 30](#_Toc5967880)

[2.4.1. Sưu tầm data huấn luyện 30](#_Toc5967881)

[2.4.2. Trích xuất thuộc tính file thực thi PE 30](#_Toc5967882)

[2.4.3. Xây dựng mô hình Deep learning 32](#_Toc5967883)

[2.5. Xây dựng sản phẩm 34](#_Toc5967884)

# MỤC LỤC HÌNH ẢNH

[I‑1 virus lây nhiễm phần đầu tệp tin 9](#_Toc5967825)

[I‑2 Virus lây nhiễm phần cuối của tệp tin 10](#_Toc5967826)

[I‑3 Mô tả hoạt động của virus macro 11](#_Toc5967827)

[II‑1 Neural Network và nơ-ron 21](file:///C:\Users\Manh\Desktop\Ứng-dụng-deep-learning-trong-bài-toán-phát-hiện-mã-độc-Tran_Hung_Manh_vs_Ngo_Quang_Sang.docx#_Toc5967828)

[II‑2 Minh họa Neural Network tương tự não người 22](file:///C:\Users\Manh\Desktop\Ứng-dụng-deep-learning-trong-bài-toán-phát-hiện-mã-độc-Tran_Hung_Manh_vs_Ngo_Quang_Sang.docx#_Toc5967829)

[II‑3 Minh họa hoạt động của Gradient descent. 24](#_Toc5967830)

[II‑4 Minh họa 2D thuật toán Gradient descent 25](#_Toc5967831)

[II‑5 Mạng Deep Neural Network cơ bản 26](file:///C:\Users\Manh\Desktop\Ứng-dụng-deep-learning-trong-bài-toán-phát-hiện-mã-độc-Tran_Hung_Manh_vs_Ngo_Quang_Sang.docx#_Toc5967832)

[II‑6 Mô tả nguyên tắc thị giác 27](#_Toc5967833)

[II‑7 Minh họa cơ chế convolution 27](#_Toc5967834)

[II‑8 Ví dụ minh họa 2D thuật toán convolution 28](#_Toc5967835)

[II‑9 Minh họa thuật toán Max Pooling 28](#_Toc5967836)

[II‑10 Cấu trúc đầy đủ của một mạng Convolution Neural Network 28](#_Toc5967837)

[III‑1 Minh họa phân tích và phân lớp dữ liệu sử dụng Deep Learning (Machine Learning) 29](#_Toc5967838)

[III‑2 Sử dụng thư viện pefile để xem được các thuộc tính của file PE 31](#_Toc5967839)

[III‑3 Code của trích xuất data và gán nhãn cho data 31](#_Toc5967840)

[III‑4 Data sau khi trích xuất vào file CSV (56 features). 31](#_Toc5967841)

[III‑5 phân chia dữ liệu training và test 32](#_Toc5967842)

[III‑6 Code Python xây dựng mô hình DNN 32](#_Toc5967843)

[III‑7 Kết quả trên training data 33](#_Toc5967844)

[III‑8 Kết quả trên testing data 33](#_Toc5967845)

[III‑9 Code Python xây dựng mô hình CNN 33](#_Toc5967846)

[III‑10 Cấu trúc cơ bản của sản phẩm 34](#_Toc5967847)

# BẢNG CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| **Chữ viết tắt** | **Mô tả** |
| PE | Portable Execute |
| IRC | Internet Relay Chat |
| AI | Artificial Intelligence |
| SMBv1 | Server Message Block version 1 |
| USB | Universal Serial Bus |
| DoS | Denial of Service |
| DDoS | Distributed Denial of Service |
| RAM | Random Access Memory |
| RAT | Remote Administration Tools |
| PUP | Potentially Unwanted Program |
| BSS | Base Station Subsystem |
| UPX | Ultimate Packer for Executables |
| API | Application Programming Interface |
| ID | Identification |
| ReLU | Rectified Linear Unit |
| DNN | Deep Neural Network |
| CNN | Convolution Neural Network |

# I. Tổng quan về mã độc

Với ứng dụng và phổ biến rộng rãi, các kênh liên lạc đa dạng và môi trường ứng dụng phức tạp trong môi trường mạng internet mang lại thuận tiện cho lây lan của phần mềm độc hại, điều này gây ra mối đe dọa lớn cho an ninh của hệ thống mạng và máy chủ trên mạng.

Mã độc hại có ý nghĩa tương tự như phần mềm độc hại, ngoại trừ mức độ chi tiết được mô tả là khác nhau.  Mã độc hay phần mềm độc hại là các chương trình máy tính có chứa bên trong nó nội dung các mã độc hại được tạo ra với mục đích thực hiện các hành vi bất hợp pháp như: truy cập trái phép, đánh cắp thông tin người dùng, lây lan thư rác, thậm chí thực hiện các hành vi tống tiền (ransomware), tấn công và gây tổn thương cho các hệ thống máy tính… nhằm chuộc lợi cá nhân, hoặc các lợi ích về kinh tế, chính trị hay đơn giản chúng có khi được tạo ra chỉ là một trò đùa ác ý nào đó. Bất kỳ một phần mềm nào là lý do làm tổn thương, phá vỡ đến tính bí mật, tính toàn vẹn và tính sẵn sàng của dữ liệu người dùng, máy tính hoặc môi trường mạng đều có thể được xem như các mã độc.

## **1. Phân loại và nguyên tắc của phần mềm độc hại**

Có nhiều phần mềm độc hại chẳng hạn như virus, worm (sâu), bot, trojan Horse (ngựa thành Troy), exploit (chương trình khai thác), backdoor (cửa sau), rootkit, spyware (phần mềm gián điệp)，spamware (phần mềm gửi thư giác), adware (phần mềm quảng cáo rác),… Đôi khi tên các phần mềm độc hại trên được gọi lẫn lộn. Ví dụ: tất cả phần mềm độc hại khiến máy tính hoạt động bất thường được gọi là virus và phần mềm độc hại không được cấp phép và ẩn trong máy tính được gọi là trojan… Báo cáo này xin được phân loại mã độc ra làm 3 loại chính dựa theo mục đích cụ thể của chúng.

Toàn bộ quá trình xâm nhập hệ thống đích và đạt được mục đích cụ thể bao gồm ba giai đoạn:

1. Chiếm quyền điều khiển từ xa của hệ thống đích
2. Duy trì điều khiển từ xa đối với hệ thống đích
3. Thực hiện mục đích cụ thể trên hệ thống đích thông qua điều khiển từ xa. Hệ thống đích được đề cập ở đây có thể là hệ thống máy chủ đích hoặc thiết bị mạng đích.

### **1.1. Xâm nhập và chiếm điều khiển từ xa của hệ thống đích**

Tính năng cơ bản của phần mềm độc hại là khả năng có được điều khiển từ xa hệ thống đích bằng nhiều cách khác nhau trong các điều kiện trái phép, đó là chức năng của giai đoạn đầu tiên trong quy trình xâm nhập.

#### **1.1.1. Exploit ( Chương trình khai thác)**

Là hình thức cơ bản lớp đầu tiên của phần mềm độc hại. Khai thác lỗi trong hệ điều hành hoặc ứng dụng để điều khiển từ xa hệ thống đích hoặc thêm quyền người dùng cục bộ một cách trái phép. Quy trình cụ thể là: xây dựng dữ liệu đầu vào và gửi nó đến chương trình hoặc ứng dụng hệ điều hành bị lỗi, để chương trình bị lỗi thay đổi theo dữ liệu đầu vào được xây dựng và luồng chương trình bình thường bị thay đổi, do đó giúp người dùng trái phép điều khiển từ xa kiểm soát hệ thống đích hoặc cung cấp cho người dùng địa phương các đặc quyền cao hơn. Sau khi sử dụng exploit để có được điều khiển từ xa của hệ thống đích, việc duy trì kiểm soát tiếp theo và hoàn thành mục đích cụ thể trong quy trình xâm nhập là độc lập với exploit. Phần mềm độc hại Exploit điển hình bao gồm Exploit tràn bộ đệm, Exploit SQL,…

Khi số lượng mã nguồn của hệ điều hành hoặc ứng dụng đạt đến một kích thước nhất định, các lỗi trong chương trình là không thể tránh khỏi, do đó, các loại lỗ hổng khác nhau đang xuất hiện và phần mềm độc hại cũng như Exploit cũng gia tăng. Bởi vì, hệ điều hành hoặc ứng dụng có thể được vá với các lỗi tương ứng, khi đó phần mềm độc hại không thể sử dụng Exploit để điều khiển hệ thống đích từ xa.

#### **1.1.2. Trojan Horse (Ngựa thành troy)**

Trong cuộc chiến của người Hy Lạp cổ đại, họ đã tặng cho kẻ thù của mình con ngựa khổng lồ. Đối thủ của người Hy Lạp đã không nghi ngờ gì chấp nhận món quà và đưa con ngựa ấy vào trong thành. Tuy nhiên chiến binh của Hy Lạp đã ẩn bên trong con ngựa và sẵn sàng hành động khi vào sâu trong thành. Mã độc trojan có tên như vậy cũng bởi phương thức hoạt động của nó khá giống với cuộc chiến thành Troy kia. Trojan đánh lừa người dùng cách khiến người dùng nghĩ mình đang cài đặt một chương trình, ứng dụng một cách hợp pháp và an toàn nhưng thay vào đó sẽ có một mục đích ẩn mà người dùng không thể biết được

Đây là một loại phần mềm độc hại giả vờ là một chương trình hợp pháp để lừa người dùng thực thi. Quá trình cụ thể của việc sử dụng Trojan để xâm chiếm hệ thống đích là: trước tiên, trojan lây lan cho người dùng thông qua mạng hoặc các phương tiện lưu trữ khác nhau. Vì nó được ngụy trang và lừa đảo, nó thường được phát hành cho những người dùng thiếu kinh nghiệm, nhận thức. Nó cũng có thể tạo backdoor (cửa sau, xem mục 1.2.1) để có được quyền điều khiển từ xa của máy chủ đích, nếu cần, cũng có thể giải phóng exploit mà nó mang theo để thêm quyền của người dùng.

Có thể thấy từ quá trình sử dụng trojan để xâm nhập, không giống như tính năng exploit chỉ bao gồm điều khiển từ xa, trojan cũng giống backdoor, ngoài trojan header được sử dụng để lừa người dùng thực thi trên hệ thống đích, exploit và các payload khác, các payload này được phát hành sau khi thực hiện trojan để duy trì kiểm soát từ xa hệ thống đích và hoàn thành mục đích cụ thể.

Trojan trên máy tính cũng vậy, được cài vào hệ thống của chúng thông qua các hình thức “tặng quà” hay những cách tương tự. Ví dụ ta cần kiếm một chương trình nào đó phục vụ công việc và tìm chúng qua các mạng chia sẻ, các diễn đàn hay tìm kiếm torrent của ứng dụng này. Các hacker biết rõ điều này nên họ đã tạo sẳn các chương trình trên với tập tin crack đã được “khuyến mãi” thêm mã độc (trojan/backdoor). Nếu bất cẩn người dùng có thể bị nhiễm trojan theo hình thức này, một khi bị nhiễm thì các tín hiệu bàn phím chúng ta gõ vào hay những hành động trên máy tính của mình sẽ được thông báo đến hộp thư của hacker hay đẩy lên một máy chủ FTP nào đó trên mạng internet. Đối với các trojan phức tạp và tinh vi sẽ được trang bị thêm các cơ chế nhận lệnh từ kênh IRC để các hacker dễ dàng điều khiển và phát động các cuộc “tổng tấn công” gây ra tình trạng từ chối dịch vụ của website hay máy chủ của cơ quan hay tổ chức.

Bên cạnh việc lây lan Trojan thông qua sao chép và thực thi tệp thực thi PE, nhiều loại trojan lừa đảo như là trojan đính kèm thư, trojan mã độc web và trojan virus macro. Những Trojan này thường bị người dùng thực thi do sự thiếu hiểu biết và thiệt hại là rất lớn.

#### 1.1.3. Worm (sâu)

 Nó có khả năng tự tái tạo để tự động phát tán một loại phần mềm độc hại trong môi trường mạng mà không cần sự can thiệp của người dùng. Worm lợi dụng mật khẩu yếu hoặc khiếm khuyết trong hệ thống đích để giành quyền kiểm soát từ xa hệ thống đích và thu thập thông tin liên quan trong hệ thống đích để lây nhiễm chính worm vào các hệ thống khác được nối mạng với hệ thống đích. . Có hai dạng tồn tại của worm, có thể ở dạng tệp hoặc tiến trình/luồng trong bộ nhớ. Khi nó tồn tại dưới dạng tiến trình/luồng, Worm không liên quan đến các hoạt động tệp trong hệ thống đích và sẽ được che giấu trong quá trình lan truyền.

 Worm (sâu) thường dựa vào các hành vi, lỗ hổng trong các giao thức mạng để lan truyền. Ví dụ, sâu ransome WannaCry đã khai thác lỗ hổng trong phiên bản đầu tiên của giao thức chia sẻ tài nguyên của Server Message Block (SMBv1) được triển khai trong hệ điều hành Windows. Sau khi hoạt động trên máy tính mới bị nhiễm, phần mềm độc hại WannaCry sẽ bắt đầu tìm kiếm mạng cho các nạn nhân tiềm năng mới: các hệ thống đáp ứng yêu cầu SMBv1 do sâu tạo ra. Con sâu có thể tiếp tục nhân giống trong một hệ thống theo cách này. Khi nạn nhân mang thiết bị cá nhân bị nhiễm đi làm, đi chơi hay đâu đó, sâu có thể lây lan sang các mạng khác.

Stuxnet, một trong những loại sâu máy tính khét tiếng nhất cho đến nay, bao gồm một thành phần sâu để lan truyền phần mềm độc hại thông qua việc chia sẻ các thiết bị USB bị nhiễm, cũng như phần mềm độc hại nhắm vào các hệ thống thu thập dữ liệu và kiểm soát giám sát, được sử dụng rộng rãi trong môi trường công nghiệp , bao gồm các tiện ích điện, dịch vụ cấp nước, nhà máy xử lý nước thải và các nơi khác.

Tương tự như Trojan, Worm cũng bao gồm phần Payload để hoàn thành mục đích cụ thể sau khi xâm nhập thành công hệ thống đích, như CodeRed Worm, có phần Payload là DoS (Denial of Service, từ chối dịch vụ).

#### 1.1.4. Bot

Khái niệm của Bot tương tự như Worm. Nếu phần Payload của Worm chứa Backdoor, thì Worm được gọi là Bot hoặc Worm được điều khiển từ xa là Bot.

Bot so với worm nguy hiểm hơn 1 bậc, mặc dù worm tự nhân bản độc lập, có thể lây lan, nhưng sau đó nó lan rộng ra nó không còn kiểm soát, không thể điều chỉnh hành vi theo mong muốn của người phát hành worm; trong khi Bot vẫn có thể được kiểm soát và điều chỉnh sau khi nó được tuyền đi. Có thể thấy rằng bot có hại hơn worm và Bot được chế tạo tốt thậm chí có thể nâng cấp phần payload của nó để thích nghi với môi trường mới hoặc hoàn thành các chức năng mới theo ý định của nhà phát hành.

Các máy chủ sau khi bị bot xâm nhập thành công là Zombie (máy chủ bù nhìn), một nhóm được gọi là Zombie Botnet. Cuộc tấn công DDoS (Distributed Denial of Service, Từ chối dịch vụ phân phối), một trong những mối đe dọa bảo mật chính trên Internet, được thực hiện bằng cách truyền các lệnh tấn công DoS cho Botnet cho một mục tiêu cụ thể.

#### 1.1.5. Virus máy tính

Virus máy tính (hay còn gọi là virus) là một chương trình phần mềm có khả năng tự sao chép chính nó từ đối tượng lây nhiễm này sang đối tượng khác (đối tượng có thể là các file chương trình, văn bản, máy tính…). Virus có nhiều cách lây lan và cũng có nhiều cách để gây thiệt hại cho máy tính. Nó có thể làm hỏng dữ liệu, thay đổi dữ liệu hay làm giảm hiệu suất hệ thống bằng cách việc lợi dụng tài nguyên bộ nhớ hay không gian lưu trữ của đĩa. Virus yêu cầu ai đó cố tình hoặc vô tình lây nhiễm mà không có kiến ​​thức hoặc sự cho phép của người dùng hoặc chuyên viên quản trị hệ thống.

Virus có cấu trúc tương tự như worm và bao gồm phần payload để hoàn thành mục đích cụ thể, ngoài ra virus header được sử dụng để tự nhân bản.

Virus và worm rất dễ nhầm lẫn giữa hai khái niệm này. Sự khác biệt chính giữa hai loại này là sự lan truyền của worm không yêu cầu các tệp máy chủ và có thể truyền trực tiếp chính worm đến hệ thống đích thông qua mạng, trong khi việc truyền virus đòi hỏi các tệp máy chủ, virus chỉ có thể được ký sinh trong tệp máy chủ. Nhân bản và truyền bệnh của worm không cần can thiệp thủ công, nó được worm thực hiện tự động và hoàn thành tự động; trong khi sự lây lan của virus đòi hỏi phải can thiệp thủ công. Đầu tiên, sự lây lan của virus phụ thuộc vào vị trí của tệp máy chủ thay đổi và nơi tệp máy chủ đi, nơi virus có thể lây lan. Thứ hai, nếu tệp máy chủ không được thực thi, virus bị ký sinh sẽ không lây nhiễm vào hệ thống đích.

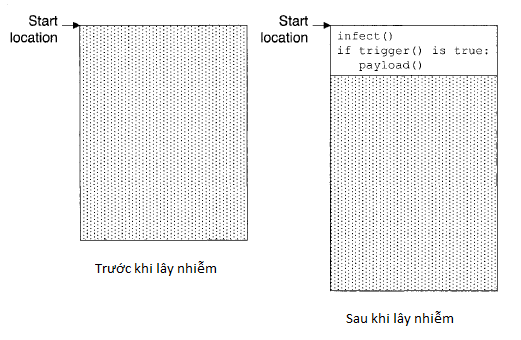
Virus có thể lây lan bằng cách mở tệp đính kèm email, nhấp vào tệp thực thi, truy cập trang web bị nhiễm hoặc xem quảng cáo trang web bị nhiễm. Nó cũng có thể lây lan qua các thiết bị lưu trữ di động như USB chẳng hạn.

##### 1.1.5.1. Virus lây nhiễm tập tin

Virus file hay virus lây truyền qua tập tin là loại virus có đặc điểm lây nhiễm qua các tệp, nó tự đính kèm vào các tệp chương trình, thường được chọn là các tệp .com, .exe, .sys, .ovl, .prg và .mnu. Khi chương trình được tải, virus cũng được tải. Các virus lây nhiễm tệp khác đến dưới dạng các chương trình hoặc tập lệnh chứa toàn bộ được gửi dưới dạng tệp đính kèm vào một ghi chú email. Có hai vấn đề cần quan tâm trong virus lây nhiễm tệp tin đó là

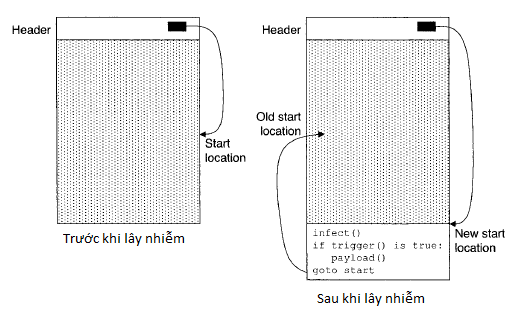
* Virus nằm ở đâu?
* Virus được thực thi như thế nào khi mà các tập tin lây nhiễm chạy?

Thông thường virus được tiến hành đặt ở hai nơi, nơi bắt đầu một tệp tin và nơi kết thúc  
hoặc một vị trí nào đó trong tệp tin, các kĩ thuật thực hiện của virus được trình bày như sau:



0‑I virus lây nhiễm phần đầu tệp tin

Lây nhiễm phần đầu của tập tin: các tập tin đơn giản thường có định dạng như .EXE, .COM, MS-DOS toàn bộ các tập tin được kết hợp bởi phần mã và phần dữ liệu. Khi được thực thi toàn bộ tập tin sẽ được nạp lên bộ nhớ và thực thi sẽ được bắt đầu bởi lệnh nhảy đến phần đầu tiên nạp tập tin đó. Trong trường hợp này virus được đặt ở vị trí bắt đầu của tập tin nơi mà các tập tin nhận quyền điều khiển đầu tiên khi các tập tin bị lây nhiễm chạy nó sẽ thực thi các đoạn mã độc đầu tiên.



0‑II Virus lây nhiễm phần cuối của tệp tin

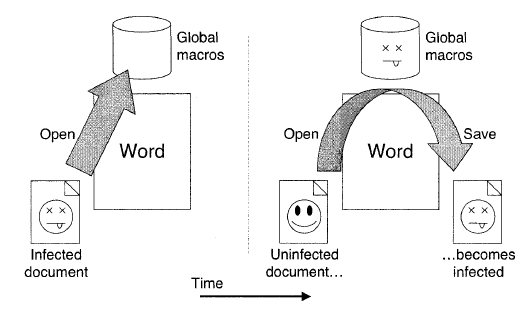
Lây nhiễm phần cuối của tập tin: gắn thêm các mã vào phần cuối của một tập tin thì cực kì đơn giản, một virus đặt ở vị trí kết thúc một tập tin bằng cách nào nó có thể chiếm đoạt quyền điều khiển, có 2 phương pháp cơ bản như sau:

1. Các mã lệnh chương trình gốc có thể được lưu lại, và được thay thế bởi một lệnh nhảy đến các mã của virus, sau khi thực thi xong virus sẽ sử dụng quyền điều khiển trỏ đến các mã ban đầu mà nó đã lây nhiễm.
2. Virus có thể sẽ cố gắng chạy trực tiếp các mã lệnh chương trình gốc ngay tại khu vực nó đã lưu hoặc virus có thể phục hồi các mã lệnh gốc này về trạng thái ban đầu và chạy nó.

##### 1.1.5.2. Virus macro

Một số ứng dụng cho phép chứa các tập tin dữ liệu, như các trình xử lý văn bản cho phép nhúng các hàm macro bên trong chúng, macro là một đoạn mã nhỏ được viết bởi một ngôn ngữ chuẩn được biên dịch bởi ứng dụng, ngôn ngữ này cung cấp đầy đủ các hàm để viết một virus. Do vậy virus macro chiếm được nhiều lợi thế hơn là virus lây nhiễm tập tin.

Trong hệ điều hành Windows thì macro chủ yếu là các đoạn mã được viết bởi ngôn ngữ lập trình Visual Basic đây là một ngôn ngữ được sử dụng trong phần mềm thuộc bộ Microsoft Office như Word, PowerPoint, Excel. Đặc điểm của những virus này là nhắm vào các lệnh ngôn ngữ macro trong các ứng dụng như Microsoft Word và các chương trình khác. Trong Word, macro là các chuỗi đã lưu cho các lệnh hoặc tổ hợp phím được nhúng trong tài liệu. Virus macro có thể thêm mã độc của chúng vào chuỗi macro hợp pháp của tệp Word. Microsoft đã tắt macro theo mặc định trong các phiên bản Word mới hơn. Nhưng tội phạm mạng cũng sử dụng trang web hay blog đưa ra thông tin không chính xác cho người dùng nhằm kích hoạt macro và khởi chạy virus. Vì đã thấy sự hồi sinh của các virus macro trong những năm gần đây, Microsoft đã thêm một tính năng mới trong Office 2016 cho phép các nhà bảo mật chọn lọc sử dụng macro cho các quy trình công việc đáng tin cậy, cũng như chặn các macro trên toàn hệ thống.



0‑III Mô tả hoạt động của virus macro

##### 1.1.5.3. Virus overwrite

Đối với người dùng cuối, virus overwrite là một trong những virus gây phiền toái nhất, ngay cả khi nó không đặc biệt nguy hiểm cho toàn bộ hệ thống.

Virus overwrite (ghi đè) được thiết kế đặc biệt để xóa tệp hoặc ứng dụng, nó sẽ xóa nội dung của bất kỳ file nào mà nó lây nhiễm. Cách duy nhất để loại bỏ virus là xóa file, và do đó, người dùng bị mất toàn bộ nội dung của file đó. Sau khi lây nhiễm một hệ thống, một virus overwrite bắt đầu ghi đè lên các tệp bằng mã riêng của nó. Những virus này có thể nhắm mục tiêu vào các tệp hoặc ứng dụng cụ thể hoặc ghi đè một cách có hệ thống cho tất cả các tệp trên thiết bị bị nhiễm. Một virus overwrite có thể cài đặt mã mới trong các tệp và ứng dụng lập trình chúng để phát tán virus sang các tệp, ứng dụng và hệ thống bổ sung.

##### 1.1.5.4. Virus đa hình

Virus đa hình là một loại phần mềm độc hại có khả năng thay đổi hoặc biến đổi mã cơ bản mà không thay đổi các chức năng hoặc tính năng cơ bản của nó. Quá trình này giúp virus tránh được phát hiện từ nhiều sản phẩm chống phần mềm độc hại và phát hiện mối đe dọa dựa trên việc xác định chữ ký của phần mềm độc hại; một khi chữ ký của virus đa hình được xác định bởi một sản phẩm bảo mật, thì virus đó có thể tự thay đổi để nó sẽ không còn bị phát hiện khi sử dụng chữ ký đó.

Như vậy có thể thấy một virus đa hình là một virus thực hiện các phương pháp che dấu dựa vào việc mã hóa các thân mã chương trình của nó bằng việc thay đổi các cách thức mã hóa hay thay đổi phương thức giải mã nhiều lần, chúng có khả năng tinh vi lẩn trốn trước sự tìm kiếm của các phần mềm diệt virus bởi các biến thể sinh ra ngẫu nhiên theo thời gian và các đối tượng lây nhiễm.

##### 1.1.5.5. Virus resident (thường trú)

Loại virus này tự nhúng vào bộ nhớ của một hệ thống (lây nhiễm qua file). Điều này cho phép virus tiếp tục hoạt động ngay cả khi nguồn gốc của việc lây nhiễm virus đã được loại bỏ. Tức là ngay cả khi virus gốc bị xóa, phiên bản được lưu trong bộ nhớ có thể được tái kích hoạt khi hệ điều hành tải một ứng dụng hoặc chức năng cụ thể. Virus thường trú là một vấn đề khó giải quyết vì nó có thể lẩn trốn phần mềm diệt virus và phần mềm phát hiện virus bằng cách ẩn trong RAM của hệ thống.

Tùy thuộc vào việc virus được lập trình ra sao, việc phát hiện chúng có thể rất phức tạp và thậm chí việc loại bỏ chúng còn khó khăn hơn nữa. Có thể tách virus resident thành hai nhóm: Fast infector (lây lan nhanh) và slow infector (lây lan chậm). Fast infector gây ra sự phá hủy càng nhanh càng tốt và do đó dễ phát hiện hơn; còn slow infector khó nhận biết hơn vì các dấu hiệu của chúng phát triển rất chậm.

### 1.2. Duy trì quyền điều khiển từ xa cho hệ thống đích

Sau khi có quyền điều khiển từ xa của hệ thống mục tiêu, phần mềm độc hại này sẽ chạy trên hệ thống mục tiêu để duy trì quyền điều khiển từ xa của hệ thống đích. Loại phần mềm độc hại này được sử dụng trong giai đoạn thứ hai của quá trình xâm nhập hoàn toàn.

#### **1.2.1. Backdoor (cửa sau)**

Nó là một loại chạy trong hệ thống mục tiêu, để cung cấp trái phép dịch vụ điều khiển từ xa phần mềm độc hại.

Cần lưu ý rằng Backdoor khác với loại phần mềm độc hại đầu tiên. Chức năng của Backdoor là cung cấp dịch vụ để kiểm soát từ xa trái phép hệ thống đích. Loại phần mềm độc hại đầu tiên cần nhiều cách khác nhau để đạt được điều này. Nghĩa là, Backdoor phải được chạy trên hệ thống đích để cung cấp dịch vụ tương ứng, vì vậy loại phần mềm độc hại đầu tiên phải được sử dụng trước tiên để có được quyền thực thi chương trình trên hệ thống đích. Loại phần mềm độc hại đầu tiên là tiền đề và cơ sở để Backdoor hoạt động. Sau khi Backdoor chạy, việc điều khiển từ xa của hệ thống đích được hoàn thành bởi các dịch vụ do Backdoor cung cấp.

Backdoor thường là một đoạn mã nằm trong phần mềm, hoặc một phần mềm nằm trong một phần cứng cho phép truy cập từ xa để lấy thông tin, hỗ trợ, phân tích hoặc dùng cho các mục đích khác. Backdoor thường không được ghi chú hay thông báo cho người dùng, vậy nên người dùng không hề biết đến sự tồn tại của nó cho đến khi backdoor bị phát hiện.

Một số backdoor được dùng bởi các công ty để cập nhật phần mềm của họ từ xa theo cách nhanh nhất có thể, tránh việc người dùng lười hay không biết update phần mềm qua giao diện được cung cấp. Một số backdoor khác thu thập thông tin nặc danh về việc sử dụng thiết bị giúp nhà sản xuất hiểu hơn về cách người dùng tương tác với thiết bị để tối ưu sản phẩm. Một số khác thì tự động phát hiện lỗi để nhà sản xuất có thể trợ giúp khách hàng tốt hơn. Đây là backdoor tốt.

Một số khác thì phục vụ cho mục đích đen tối, ví dụ như lén coi trộm thông tin khách hàng, lén lấy thông tin thẻ tín dụng, lén đọc tin nhắn, thông tin nhạy cảm hay mở cửa để lén cài thêm các phần mềm khác để chiếm quyền kiểm soát thiết bị của người dùng. Đây là backdoor xấu, và đôi khi nó được gọi là Remote Access Trojan (RAT). Hacker đôi khi cũng tự cài backdoor vào máy tính, thiết bị của người dùng khi khai thác lỗ hổng bảo mật để phục vụ mục đích xấu như: lén download thêm những công cụ hack hay mã độc khác.

Một loại backdoor không xấu cũng không tốt phục vụ cho mục đích giám sát của chính phủ. Loại backdoor này thu hút sự chú ý của dư luận kể từ năm 2013 khi Edward Snowden tiết lộ những tài liệu cho thấy chính phủ Mỹ hợp tác với cơ quan tình báo Anh đã mở ra nhiều backdoor để phục vụ cho mục đích điều tra, theo dõi của họ. Nếu dùng để chống khủng bố, ngăn chặn các cuộc tấn công thì nó tốt, nhưng cũng đặt ra vấn đề rất lớn về quyền ẩn danh, quyền bí mật trong thư tín điện thoại, điện tín…

#### **1.2.2. Rootkit**

Khái niệm này có nguồn gốc trong hệ điều hành UNIX / Linux, ban đầu dùng để chỉ tập hợp các công cụ hệ thống UNIX / Linux có được và duy trì một tập hợp các đặc quyền Root. Cho đến nay, khái niệm rootkit được chấp nhận rộng rãi đề cập đến một công cụ được sử dụng để giúp những kẻ xâm nhập duy trì các đặc quyền quản trị viên đó càng lâu càng tốt sau khi có được quyền quản trị viên máy chủ đích.

Trong khái niệm rootkit hiện tại, quá trình lấy đặc quyền của quản trị viên không được thực hiện bởi rootkit, nghĩa là việc sử dụng rootkit dựa trên giả định rằng quyền quản trị viên đã đạt được. Theo khái niệm backdoor, backdoor chỉ cung cấp một "kênh" để truy cập và kiểm soát trái phép hệ thống đích, nhưng nó không liên quan đến việc bảo vệ kênh này, vì vậy kênh này được quản trị viên dễ dàng quản lý trên hệ thống đích hoặc thiết bị bảo mật hoặc cái gì đó phát hiện ra nó. Vai trò của rootkit là duy trì điều khiển từ xa hệ thống đích càng lâu càng tốt. Do đó, nhiệm vụ cơ bản là ẩn kênh do backdoor cung cấp, để làm cho quản trị viên hoặc thiết bị bảo mật trên hệ thống đích không thể phát hiện ra sự tồn tại của kênh.

Như vậy đặc điểm chính của rootkit là có khả năng che dấu nên nếu dùng các chương trình từ hệ thống như: “Registry Editor”, “Find Files”, “Task Manager” thì không thể phát hiện. Thậm chí dù có phát hiện ra rootkit đi nữa thì xóa được nó cũng không hề đơn giản chút nào. Không thể sử dụng các công cụ bình thường mà phải dùng các chương trình anti rootkit đặc biệt. Rootkit thường hoạt động ở 2 mức là mức ứng dụng (User–mode) và mức nhân hệ điều hành (Kernel–mode) nên phát hiện được chúng vô cùng khó khăn.

Các rootkit trong các nền tảng hệ điều hành chính hiện tại đã tương đối hoàn thiện. Các rootkit ở trình độ cao có thể ẩn sâu các tiến trình, luồng, kết nối mạng, dữ liệu mạng và giao tiếp mạng trong hệ điều hành để bảo vệ phần mềm độc hại đang chạy không được phát hiện trong hệ thống đích.

Trong các ứng dụng thực tế, rootkit thường bao gồm chức năng của Backdoor trực tiếp. Do đó, rootkit có thể được hiểu là một Backdoor với các tính năng ẩn.

### 1.3. Thực hiện mục đích cụ thể trên hệ thống đích

Loại phần mềm độc hại thứ ba được sử dụng để hoàn thành các hoạt động cuối cùng mà hệ thống xâm nhập cần thực hiện, như đánh cắp thông tin tình báo, phá hủy hệ thống, khởi động các cuộc tấn công và truyền dữ liệu. Vai trò của loại phần mềm độc hại thứ nhất và thứ hai là cung cấp nền tảng vận hành an toàn và thuận tiện cho loại phần mềm độc hại thứ ba.

#### 1.3.1. Spyware (phần mềm gián điệp)

Đây là loại phần mềm độc hại thứ ba điển hình được sử dụng để thu thập thông tin khác nhau từ hệ thống mục tiêu, chẳng hạn như kinh doanh, tình báo quân sự, số thẻ tín dụng người dùng, thông tin / tài liệu riêng tư cá nhân, tên người dùng trang web / hộp thư, mật khẩu, v.v. .

Sau khi thu thập thông tin này, phần mềm gián điệp sẽ gửi nó cho kẻ xâm nhập qua mạng. Giao tiếp mạng được tạo bởi chính phần mềm gián điệp và dưới sự bảo vệ của rootkit phần mềm gián điệp được ẩn đi, để phần mềm gián điệp có thể tồn tại an toàn trong hệ thống đích.

Spyware đang chạy trên cách tiếp cận hệ thống mục tiêu có ba giai đoạn chính:

* Khi trên worm/bot/virus/trojan thực hiện phần mềm gián điệp worm/bot/virus/trojan trong payload, được phát hành và thực hiện;
* Sử dụng khai thác get sau khi điều khiển từ xa hệ thống đích, phần mềm gián điệp được truyền đến máy chủ đích thông qua mạng và được thực thi.
* Sau khi rootkit/backdoor được cài đặt và chạy trên hệ thống đích, phần mềm gián điệp được truyền đến máy chủ đích thông qua dịch vụ điều khiển từ xa do hệ thống đích cung cấp.

Ví dụ như cookie theo dõi và ghi lại thông tin cá nhân của người dùng và thói quen duyệt internet là một trong những loại phần mềm quảng cáo phổ biến nhất. Nhà quảng cáo có thể sử dụng cookie để theo dõi những trang web nào người dùng truy cập để nhắm mục tiêu quảng cáo trong chiến dịch tiếp thị. Ví dụ: nhà quảng cáo có thể theo dõi lịch sử trình duyệt và tải xuống của người dùng với mục đích hiển thị quảng cáo bật lên hoặc biểu ngữ để thu hút người dùng mua hàng. Vì dữ liệu được thu thập bởi phần mềm gián điệp thường được bán cho bên thứ ba, vậy nên các quy định như GDPR (Quy định bảo vệ dữ liệu chung) đã được ban hành để bảo vệ thông tin nhận dạng cá nhân (PII) của khách truy cập trang web.

Keylogger thường là một phần mềm nhỏ gọn – hoặc đôi lúc nguy hiểm hơn thậm chí là một thiết bị phần cứng – với khả năng ghi lại mọi phím bấm mà người dùng đã nhấn trên bàn phím. Tổng hợp kết quả của các tổ hợp phím này, người cài đặt keylogger có thể thu được tin nhắn cá nhân, nội dung email, số thẻ tín dụng và dĩ nhiên nguy hiểm nhất là mọi loại mật khẩu của người dùng. Keylogger cũng có thể được sử dụng bởi chủ lao động để quan sát hoạt động máy tính của nhân viên, phụ huynh để giám sát việc sử dụng internet của con cái họ, người dùng để theo dõi hoạt động trái phép có thể có trên thiết bị của họ hoặc cơ quan thực thi pháp luật để phân tích các sự cố liên quan đến sử dụng máy tính.

Các keylogger là một thiết bị phần cứng giống với ổ flash USB và đóng vai trò là đầu nối vật lý giữa bàn phím máy tính và máy tính, trong khi các chương trình keylogging là phần mềm không yêu cầu truy cập vật lý vào máy tính của người dùng để cài đặt. Người nào đó có thể tải xuống keylogger theo mục đích bởi ai đó muốn theo dõi hoạt động trên một máy tính cụ thể hoặc họ vô tình tải xuống và thực thi như một phần của rootkit hoặc Trojan quản trị từ xa (RAT).

#### 1.3.2. Adware (phần mềm quảng cáo rác)

Adware là phần mềm không mong muốn được thiết kế để xuất hiện quảng cáo trên màn hình thiết bị của người dùng, thường trong trình duyệt web. Một số chuyên gia bảo mật xem nó là tiền thân của những chương trình không mong muốn PUP (Potentially Unwanted Program).

Adware mang lại lợi nhuận cho nhà phát triển bằng cách tự động hiển thị quảng cáo online trong giao diện người dùng của phần mềm hoặc trên màn hình trong quá trình cài đặt. Và khi đó người dùng sẽ thấy các chương trình giảm cân siêu tốc, cung cấp bí quyết làm giàu nhanh chóng và thậm chí là những cảnh báo virus giả mạo “mời” họ click vào. Ngoài ra người dùng có thể thấy các tab mới tự động mở, thay đổi trang chủ trình duyệt, kết quả từ công cụ tìm kiếm người dùng chưa thấy bao giờ hoặc thậm chí chuyển hướng đến trang khác.

Cũng giống như spyware, adware cũng có 3 giai đoạn chính và tương tự như spyware.

## **2. Quan hệ giữa hai loại mã độc hại và các cơ chế hoạt động**

Loại phần mềm độc hại thứ nhất và thứ hai của là một điều kiện tiên quyết cho các loại thứ ba, chúng đóng một vai trò cung cấp và duy trì cho các loại thứ ba trên hệ thống đích điều khiển từ xa một cách trái phép, việc thực hiện một mục đích cuối cùng của sự xâm nhập được thực hiện bởi một loại phần mềm độc hại thứ ba. Trong loại phần mềm độc hại đầu tiên, exploit, trojan, worm, bot và virus đều có thể bao gồm phần payload. Do đó, ngoài chức năng có được quyền kiểm soát từ xa đối với hệ thống đích, trojan, worm, bot và virus cũng có thể bao gồm loại phần mềm độc hại thứ hai và loại phần mềm độc hại thứ ba trong payload và phát hành chúng sau khi thực thi.

Một mối đe dọa hiện nay là có loại tấn công mang tên Blended Attck, sử dụng nhiều công nghệ liên quan đến phần mềm độc hại để tối đa hóa hiệu quả tấn công.

1. Trong giai đoạn đầu tiên để xâm nhập và giành được quyền điều khiển từ xa của hệ thống đích, các phương tiện xâm nhập như trojan, worm, bot và virus được kết hợp trong một chương trình thực thi cụ thể để có thể kết hợp giữa xâm nhập chủ động và xâm nhập thụ động, từ đó tăng cường đáng kể khả năng xâm nhập vào hệ thống đích.
2. Trong giai đoạn duy trì kiểm soát từ xa đối với hệ thống đích, việc sử dụng thành thạo các kỹ thuật liên quan đến rootkit cho phép rootkit và các phần mềm độc hại khác được bảo vệ để tồn tại trong hệ thống đích một cách đáng tin cậy và trong thời gian dài.
3. Khi hoàn thành mục đích xấu cụ thể, một loạt các chức năng phần mềm độc hại loại ba được tích hợp, để có thể thu thập thông tin, có thể gây thiệt hại cho hệ thống đích nếu muốn.

## **3. Bài toán phát hiện mã độc**

Vào cuối những năm 80, phần mềm chống mã độc (hay phần mềm chống virus) đã ra đời với sự ra đời của mã độc. Công nghệ được sử dụng trong giai đoạn mã độc này tương đối đơn giản, giúp máy tính dễ dàng phát hiện các mã độc đã biết trước. Hầu hết các phần mềm chống virus ban đầu đều áp dụng phương pháp so sánh tính năng và chỉ cần sử dụng chuỗi tính năng để hoàn tất phát hiện. Với sự phát triển của công nghệ mã độc, mã độc bắt đầu bị biến dạng trong quá trình lan truyền để tránh bị giết. Vào thời điểm này, số lượng biến thể của cùng một mã độc đã tăng lên đáng kể và đã thay đổi rất nhiều từ hình thức cấu trúc. Rất khó để phần mềm chống virus trích xuất một đoạn mã làm chữ ký cho mã độc. Trong tình hình đó, chữ ký phổ rộng được ra đời, chữ ký phổ rộng chia chữ ký thành nhiều đoạn và sử dụng mã hóa cặp byte để phân chia các đoạn cần so sánh và các đoạn không cần so sánh. Tuy nhiên, cho dù đó là quét chữ ký hay chữ ký phổ rộng, cần phải trích xuất các tính năng sau khi lấy các mẫu mã độc, và sau đó mới có thể phát hiện. Điều này làm cho việc phát hiện mã độc có độ trễ nhất định và luôn đi theo mã độc. Để đối phó với các loại virus đã biết và cả chưa biết, quét heuristic đã ra đời. Quét heuristic sử dụng kinh nghiệm và kiến ​​thức đã có để tiến hành phát hiện mã nhị phân không xác định. Kỹ thuật này ghi lại mã độc có các tính năng độc hại mà các tệp nhị phân thông thường không có. Các hành vi, chẳng hạn như các tệp đọc và ghi khác thường, tự động kết thúc, v.v. Trọng tâm và khó khăn của quét heuristic là làm thế nào để trích xuất các đặc trưng hành vi độc hại của mã độc. Quét chữ ký, tìm kiếm các chữ ký phổ rộng và quét heuristic, cả 3 phương pháp tra diệt đều không có tệp nhị phân đang chạy thực tế, có thể được phân loại là phương pháp phát hiện tĩnh mã độc. Với sự phát triển dần dần của công nghệ chống phần mềm độc hại và công nghệ điện toán đám mây đã được sử dụng ngày càng nhiều bởi các nhà cung cấp bảo mật, nhưng phương pháp phát hiện tĩnh mã độc vẫn là hiệu quả nhất và được sử dụng rộng dãi nhất để phát hiện mã độc.

## 4. Phân tích cấu trúc định dạng tệp thực thi PE (Portable Execute)

Hiện nay việc phân tích và phát hiện mã độc chủ yếu dựa vào 2 phương pháp: **phân tích động** và **phân tích tĩnh**. Trong đó phương pháp phân tích động, thông qua việc thực thi các mã chương trình và quan sát các hành vi của nó để quyết định xem đoạn mã đó có phải là mã độc hay không. Ưu thế của phương pháp này là quá trình phân tích nhanh, dễ dàng ngay cả với những mã độc bị mã hóa phức tạp tuy nhiên lại không hiệu quả khi gặp phải những mã độc chạy theo thời gian hoặc các mã độc có khả năng nhận ra sự theo dõi và sẽ không thực thi các chức năng của nó. Phương pháp phân tích tĩnh dựa vào việc thực hiện dịch ngược mã độc về dạng assembly hay dạng mã hex và phân tích các dấu hiệu của mã độc, phương pháp này tuy gặp khó khăn với những mã độc bị mã hóa hay đóng gói phức tạp nhưng có ưu điểm là có thể phát hiện ra mã độc ngay cả khi không cần thực thi nó và các đặc trưng của mã độc cũng như cấu trúc của nó cho phép phát hiện mã độc một cách chính xác. Như đã trình bày ở mục 3 phương pháp phân tích tĩnh mã độc vẫn được sử dụng rộng rãi nhất trong hiện tại và tương lai.

Khi phân tích áp dụng phương pháp phân tích tĩnh, các file dạng PE sẽ được dịch ngược về các mã hex và dữ liệu mẫu sau khi được dịch ngược sẽ được xử lý để lấy các mã hex quan trọng chủ yếu chúng nằm ở các phần PE header và Section nơi chứa các mã chương trình (Executable Code Section), các dữ liệu (Data Section), các tài nguyên (Resources Section), các thư viện (Import Data ,Export Data) … Nhìn chung các tính năng trích xuất sẽ được chia làm 4 loại:

### 4.1. PE File

Bao gồm những thông tin cần thiết được sử dụng bởi loader, các thông tin về tệp PE và các hệ điều hành thực thi nó. PE Header được sử dụng để trích xuất một tập hợp các tính năng đã biết và dưới đây là các tính năng đã được trích xuất

|  |  |
| --- | --- |
| Tên thuộc tính | Mô tả |
| Machine | giá trị xác định PE File này được biên dịch cho dòng máy nào |
| SizeOfOptionalHeader | Kích thước của Optional header |
| Characteristics | là bit cờ, xác định định dạng PE File |
| MajorLinkerVersion | Số phiên bản chính của linker. |
| MinorLinkerVersion | Số phiên bản phụ của linker. |
| SizeOfCode | Kích thước của section .text hoặc tổng kích thước của các section .text. |
| SizeOfInitializedData | Kích thước của section khởi tạo hoặc tổng kích thước. |
| SizeOfUninitializedData | Kích thước của phần dữ liệu chưa khởi tạo (BSS), hoặc tổng của tất cả các phần nếu có nhiều phần BSS. |
| AddressOfEntryPoint | Địa chỉ của điểm bắt đầu chương trình, chính là điểm mà chương trình bắt đầu được thực thi. |
| BaseOfCode | Địa chỉ liên quan tới image base của điểm bắt đầu của code section khi nó được nạp lên bộ nhớ. |
| BaseOfData | chỉ có trong file PE32 mà không có trong PE32+ có độ dài 4 bit |
| ImageBase | Địa chỉ của byte đầu tiên của image khi được load lên bộ nhớ. Giá trị này phải chia hết cho 64K. Với mỗi hệ điều hành sẽ có giá trị mặc định khác nhau. |
| SectionAlignment | Phần liên kết các section trong bộ nhớ, tức là một section luôn luôn được bắt đầu bằng bội số của sectionAlignment |
| FileAlignment | Phần liên kết các section trong file. Tương tự như SectionAlignment nhưng áp dụng với file. |
| MajorOperatingSystemVersion | Số phiên bản chính của hệ điều hành yêu cầu. |
| MinorOperatingSystemVersion | Số phiên bản phụ của hệ điều hành yêu cầu. |
| MajorImageVersion | Số hiệu phiên bản chính của image. |
| MinorImageVersion | Số hiệu phiên bản phụ của image. |
| MajorSubsystemVersion | Số phiên bản chính của hệ thống subsystem. |
| MinorSubsystemVersion | Số phiên bản phụ của hệ thống subsystem. |
| SizeOfImage | Kích thước theo bytes của image, bao gồm tất cả headers khi image được load lên bộ nhớ. Giá trị này phải là bội số của SectionAlignment. |
| SizeOfHeaders | Kích thước của MS-DOS stub, PE header và section header làm tròn ra số chia hết cho  FileAlignment. |
| CheckSum | Checksum của file. |
| Subsystem | Sybsystem được yêu cầu để thực thi file. |
| DllCharacteristics |  |
| SizeOfStackReserve | Kích thước của stack được dự trữ. |
| SizeOfStackCommit | Kích thước của stack dùng để cam kết. |
| SizeOfHeapReserve | Kích thước khai báo không gian bộ nhớ heap |
| SizeOfHeapCommit | Kích thước không gian bộ nhớ heap dùng để cam kết. |
| LoaderFlags |  |
| NumberOfRvaAndSizes | Số lượng đầu vào của data-directory. |

### 4.2. Section Names và Characteristics

Mỗi tệp PE thường có một hoặc nhiều section ở phần thân của PE. Các section có thể bao gồm các đoạn mã nhị phân, chuỗi, tính năng, hình ảnh và nhiều hơn thế nữa. Các tệp thực thi tiêu chuẩn thường có các cấu trúc cơ bản tương tự giống nhau về tên, thứ tự và đặc điểm. Các file PE độc hại đôi khi lại có cấu trúc khác để lẩn tránh khỏi sự phát hiện và phân loại. Sự khác biệt về cấu trúc và tên của section là do một vài yếu tố như trình biên dịch tạo ra tệp PE. Các tệp PE độc hại thường được tạo bởi các trình biên dịch tùy chỉnh, nó cho phép chúng thực hiện các thao tác trên các tệp PE mà không thể được hỗ trợ trên các trình biên dịch thông thường. Ví dụ về các hoạt động đó là: đóng gói, thao tác chuỗi, mã hóa, tính năng chống gỡ lỗi, tước thông tin nhận dạng, …

Một số tên các section đã được liên kết với Builder, Compiler và Packing Infrastructure là:

|  |  |
| --- | --- |
| Tên section | Builders/Compilers/ Packing Infrastructure |
| Mpres | Mpres Packer |
| Upx | UPX |
| Tsuarch | TSULoader |
| Petite | Petit packer |
| Rmnet | Ramnit Packer |
| Nsp | NsPack packer |
| Aspack | Aspack packer |
| Themeda | Themida |
| Adata | Armadillo packer |
| Vmp | VMProtect |
| Mew | MEW Packer |
| Neolit | ImpRec-created section |
| Newsec | LordPE Builder |
| Taz | PESPIN |
| Gfids | Visual Studio (14.0) |
| Stab | Haskell compiler (GHC) |

### 4.3. Section Entropy

Kỹ thuật đóng gói được phát triển nhằm tối ưu hóa bộ nhớ và băng thông trong quá trình lưu trữ và truyền tệp tin. Sử dụng trong kỹ thuật đóng gói trở thành công cụ mạnh mẽ được kẻ viết mã độc cải tiến để tránh bị phát hiện. Kỹ thuật đóng gói (bao gồm kỹ thuật nén và mã hóa) với mục đích che giấu mã độc thường được sử dụng để chuyển đổi mã thực thi nhị phân sang một định dạng khác nhằm thu gọn mã độc và làm biến dạng khác biệt hẳn so với ban đầu, tránh sự phát hiện của các chương trình diệt virus dựa trên cơ sở mẫu nhận dạng. Trong nhiều trường hợp, mã độc hại được tổ hợp đệ quy từ các kỹ thuật nén và mã hóa khác nhau, để cùng một mã độc nhưng có thể nhanh chóng tạo ra một lượng lớn biến thể nhị phân thực thi nhằm phát tán.

Các PE section chứa data thường có một entropy đã biết. Entropy cao hơn có thể biết được dữ liệu đã bị đóng gói. Các tệp độc hại thường được đóng gói để tránh phân tích tĩnh do mã thực tế thường được lưu trữ được mã hóa ở một trong các phần và sẽ chỉ được trích xuất khi chạy. Vì thế nhóm nghiên cứu đã tiến hành tính lại entropy cho các section trong file PE để tiến hành so sánh với entropy có sẵn trong các PE section.

Công thức tính entropy:

Trong đó p(i) là khả năng của khối thông tin thứ i trong chuỗi n biểu tượng của sự kiện x, n là tổng số các giá trị có thể nhận của tín hiệu.

Trong phân tích thông tin, chúng ta quan tâm đến các byte dữ liệu (mỗi byte có 256 giá trị xảy ra khác nhau). Bởi vậy, giá trị entropy của một tệp sẽ là giá trị trong khoảng từ 0 đến 8:

0: tuần tự 8: ngẫu nhiên

<------------------------------------------------------>

Thông thường, mức entropy trên 6,7 được coi là một dấu hiệu tốt cho thấy một phần được đóng gói. Tuy nhiên, có một số trình đóng gói tùy chỉnh sử dụng phần đệm như một phần của quy trình mã hóa để tạo ra ít entropy hơn trong việc nén. Nhưng những trình đóng gói như vậy không phải là phổ biến.

### 4.4. Các PE Import

Một PE có thể import code từ các PE khác. Để làm như vậy, nó chỉ định tên tệp PE và các chức năng cần nhập. Điều quan trọng là phân tích các import để có được một hình ảnh mạch lạc về những gì PE đang làm. Một số chức năng được nhập là biểu thị cho các hoạt động độc hại tiềm ẩn như mã hóa API được sử dụng để giải nén / mã hóa hoặc API được sử dụng để chống debugging. Ví dụ về import mã độc tiềm năng:

|  |  |
| --- | --- |
| **Tên Import** | **Có khả năng là mã độc?** |
| KERNEL32.DLL!MapViewOfFile | Code Injection (mã tiêm) |
| KERNEL32.DLL!IsDebuggerPresent | Chống debugging |
| KERNEL32.DLL!GetThreadContext | Code Injection (mã tiêm) |
| KERNEL32.DLL!ReadProcessMemory | Code Injection (mã tiêm) |
| KERNEL32.DLL!ResumeThread | Code Injection (mã tiêm) |
| KERNEL32.DLL!ResumeThread | Code Injection (mã tiêm) |
| KERNEL32.DLL!WriteProcessMemory | Code Injection (mã tiêm) |
| KERNEL32.DLL!SetFileTime | Tàng hình |
| USER32.DLL!SetWindowsHookExW | API Hooking |
| KERNEL32.DLL!MapViewOfFile | Code Injection (mã tiêm) |
| ADVAPI32.DLL!CryptGenRandom | Mã hóa |
| ADVAPI32.DLL!CryptAcquireContextW | Mã hóa |
| KERNEL32.DLL!CreateToolhelp32Snapshot | Process Enumeration |
| ADVAPI32.DLL!OpenThreadToken | Token Manipulation |
| ADVAPI32.DLL!DuplicateTokenEx | Token Manipulation |
| CRYPT32.DLL!CertDuplicateCertificateContext | Mã hóa |

# II. Giới thiệu Deep Learning (Học sâu)

## 1. Tổng quan về Deep Learning

### 1.1. Deep Learning là gì?

Bắt đầu với khái niệm **Artificial Intelligence (AI)** nghĩa là **trí thông minh nhân tạo**, là mục tiêu của con người nhằm đưa trí thông minh của con người lên các hệ thống, cỗ máy nhằm thực hiện những công việc cần đến trí thông minh của con người như dự đoán, phân loại, chơi cờ vua hay thực hiện những công việc mà con người khó làm được với mức độ chính xác cao hơn như phẫu thuật, chữa bệnh bằng robot, giải đáp thắc mắc như chatbot,...

**Machine Learning (Học máy)** là một ngành khoa học nghiên cứu trong việc đưa trí thông minh của con người lên các hệ thống, cỗ máy đó. Machine Learning đưa trí thông minh của con người vào hệ thống dựa trên lượng dữ liệu đầu vào lớn (dataset), sử dụng sức mạnh tính toán ngày càng cao của các cỗ máy hệ thống (máy tính) để giúp máy tính “học” (trainning) những dữ liệu đó từ đó máy tính có thể thực hiện những công việc dựa trên lượng kinh nghiệm đã học được, về cơ bản Machine Learning giúp cho máy tính có thể học dựa trên kinh nghiệm.

**Deep Learning** là một trong nhiều thuật toán trong lĩnh vực Machine Learning trong việc giúp cho máy tính học được từ dữ liệu đầu vào, Deep Learning mô hình hóa dữ liệu bằng cách sử dụng nhiều lớp phân tích dữ liệu gần giống mô hình nơ-ron trong não người và từ những kết quả đầu ra cuối cùng đưa ra quyết định cho máy tính.

Deep Learning giúp hệ thống đào sâu và phân tích dữ liệu đồng thời xây dựng cho hệ thống kinh nghiệm về thông tin đã phân tích và ứng dụng các kinh nghiệm đã tích lũy đó vào giải quyết các vấn đề mà con người đặt ra, Deep Learning thường được dùng trong việc dự đoán thời tiết, phân loại sản phẩm, nhận diện vật thể, giọng nói, gợi ý đề xuất, dịch ngôn ngữ,...

Deep Learning giúp con người giải quyết những vấn đề mang tính chuyên môn cao hoặc những công việc tự động nhàm chán nhưng cần đến trí thông minh của con người. Mục đích cuối cùng là để giảm thiểu thời gian làm việc của con người, giúp con người có thời gian giải quyết những việc khác, nâng cao hiệu suất làm việc, hỗ trợ các nhà nghiên cứu, phát triển thương mại, quảng cáo, các ngành công nghiệp, dịch vụ,…

Deep Learning cho con người thấy khả năng phân tích dữ liệu tuyệt vời và khả năng đưa ra quyết định với độ chính xác rất cao (với những mô hình được huấn luyện bài bản với accuracy (độ chính xác) có thể đạt trên 90%) do đó Deep Learning có độ tin cậy rất cao. Vì vậy ngày nay Deep Learning được sử dụng trong các hệ thống lớn như hệ thống nhận diện và gán nhãn cho ảnh của Facebook, hệ thống mở khóa Face ID của Apple, Hệ thống gợi ý quảng cáo của Google, Deep Learning còn xuất hiện nhiều trên smartphone với nhiều công dụng như tùy chỉnh màu và lấy nét dựa trên quang cảnh, xe tự lái,… Có thể thấy Deep Learning nói riêng và lĩnh vực AI nói chung có tầm ảnh hưởng lớn và phổ biến trong cuộc sống thời đại công nghệ 4.0.

Có 2 cách học phổ biến cho Deep Learning (và cả Machine Learning) là **Học có giám sát (Supervised learning)** và **Học không giám sát (Unsupervised learning).** Trong đó:

* **Học có giám sát** là khi huấn luyện một AI sử dụng phương pháp học giám sát, ta phải đưa vào một đầu vào và chỉ cho nó chính xác đầu ra là gì. Nếu AI cho đầu ra sai, nó sẽ tự điều chỉnh sự tính toán của nó. Quá trình lặp lại trong tập dự liệu sẽ hoàn thành cho đến khi AI không đưa ra kết quả sai nữa. Một ví dụ cho học có giám sát là AI dự báo thời tiết. AI này học cách dự báo thời tiết dựa vào các dự liệu cũ đã có. Dữ liệu huấn luyện đó có các đầu vào (độ ẩm, tốc độ gió, áp suất) và các đầu ra (nhiệt độ). Học có giám sát bao gồm việc sử dụng các tập dữ liệu có dán nhãn (labelled dataset) mà có các đầu vào và đầu ra được dự đoán.
* **Học không giám sát** là khi huấn luyện AI sử dụng học không giám sát, để AI tự phân tích, phân loại dữ liệu. Một ví dụ của học không giám sát là AI dự đoán hành vi cho website thương mại. Nó sẽ không học bằng cách sử dụng các tập dữ liệu dán nhãn, thay vào đó nó sẽ tự tạo ra các phân lớp của nó cho dữ liệu đầu vào. Nó sẽ chỉ ra cho từng loại người dùng sẽ mua từng loại sản phẩm khác nhau nào.

### 1.2. Các khái niệm cơ bản của Deep Learning

#### noron

0‑I Neural Network và nơ-ron

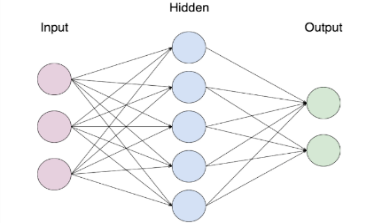
#### 1.2.1. Neural Network và nơ-ron

Một cách ngắn gọn nhất thì **Neural Network** là mô hình toán học mô phỏng **nơ-ron** trong hệ thống thần kinh con người. Mô hình đó biểu hiện cho một số chức năng của nơ-ron thần kinh con người.

Tính chất truyền đi thông tin trong nơ-ron não người bắt đầu khi **nơ-ron (node)** nhận thông tin từ các dây thần kinh (x1, x2,…xN – **đặc trưng đầu vào (input data)**), để thông tin đi vào **soma** thì phải vượt qua **dendrite** (w1, w2,…wN – **những ngưỡng (threshholds)**) dendrite sẽ quyết định những thông tin cần thiết và quan trọng sẽ đi qua dễ dàng hơn nhờ threshhold cao và ngược lại các thông tin không quan trọng sẽ bị loại bỏ dần nhờ các threshhold thấp. Sau khi thông tin qua dendrite và vào đến soma thông tin sẽ được xử lí và kết quả cuối cùng sẽ theo đường **axon (output data)** để đến **nơ-ron (node trong layer)** tiếp theo. **Θ (theta)** ở trung tâm soma đóng vai trò như **bias threshold**, là threshhold đặc biệt quyết định độ quan trọng của toàn bộ thông tin của nơ-ron đó. Công thức tổng quát mô phỏng quá trình xử lí của soma trong Neural Network được biểu diễn như sau:

Trong đó:

* y: là thông tin cuối cùng sau khi xử lí (output data) sẽ được đưa sang node trong layer tiếp theo để xử lí tiếp
* a: là hàm kích hoạt có chức năng chuẩn hóa output data, tùy theo nhu cầu từng bài toán mà a sẽ khác nhau (sẽ đề cập bên dưới)



0‑II Minh họa Neural Network tương tự não người

**Neural Network** là sự tập hợp của nhiều node khác nhau thành một hệ thống, chia thành nhiều **lớp (layer)** theo thứ tự với lớp đầu tiên là **input layer (lớp tập hợp các node nhận thông tin đầu vào)**, **các hidden layer (các lớp ẩn có chức năng xử lí thông tin)**, và **output layer (lớp cuối cùng trả về thông tin sau khi xử lí xong)**.

## 2. Mô hình thuật toán Deep Learning

### 2.1. Đặc điểm chung của các thuật toán deep learning

Đều có 3 thành phần chính là input layer, hidden layers và output layer:

* **Input layer** là tham số đầu vào của dữ liệu bằng các kĩ thuật trích xuất đặc trưng, chuẩn hóa và số hóa dữ liệu để đưa vào mô hình Deep Learning.
* **Hidden layers** là một hay nhiều layer tuần tự có tác dụng phân tích dữ liệu dựa trên các node trong từng layer hết hợp với các hàm tuyến tính (linear function) và phi tuyến tính (non-linear activation) để kích hoạt các node đó trả về giá trị nhất định và dùng giá trị đó phân tích cho những lớp tiếp theo.
* **Output layer** là kết quả cuối cùng của hidden layers, kết quả cuối cùng này nằm trong một trong những kết quả đã được huấn luyện từ trước.
* Các hàm số tuyến tính như **logistic regression**, **linear regression** và các hàm phi tuyến như **tanh, sigmoid, ReLU**, … kết hợp với các node để chuẩn hóa giá trị trả về và đưa giá trị vào lớp tiếp theo.
* Hệ thống huấn luyện như hàm **Loss function, Gradient Descent, Support Vector Machine, Kernel, Backpropagation, Regularization**, …
* Các thông số đánh giá như **Accuracy, F1 score, Precision, Recall**, …

Hệ thống huấn luyện của mô hình Deep Learning: Bao gồm các hệ thống tính toán độ lỗi từ các trọng số của mô hình Deep learning (chạy thử mô hình Deep learning và so sánh kết quả tính toán của mô hình với kết quả thực tế) và các hệ thống thực hiện điều chỉnh tham số mô hình sao cho mô hình Deep learning trả về kết quả gần chính xác nhất so với kết quả thực thế. Điển hình của hệ thống tính toán độ lỗi là Loss function và hệ thống điều chỉnh tham số là Gradient Descent.

#### 2.1.1. Loss function (Hàm mất mát)

Loss function là một hàm cơ bản để tính toán độ lỗi của trọng số mô hình Deep learning. Loss function xây dựng bằng cách cho mô hình tính toán với data huấn luyện, sau đó tính tổng trung bình độ sai khác của toàn bộ kết quả tính được so với kết quả thực tế. Độ sai khác càng thấp thì độ chính xác của mô hình càng cao.

Ta có công thức tính độ lỗi cơ bản của một mẫu thử là:

Trong đó: y là kết quả dự đoán của mô hình, là kết quả thực tế

Nhưng do công thức trên thì kết quả có thể là số âm hoặc số dương, do đó để tính tổng độ lỗi của toàn bộ mẫu thử ta cần loại bỏ dấu của kết quả, và biến đổi công thức sao cho dễ đạo hàm (để dễ làm việc với hệ thống điều chỉnh tham số như Gradient Descent).

Từ đó ta có công thức Loss function:

Trong đó: m là tổng số mẫu thử trong một lần cập nhật

#### 2.1.2. Gradient descent (Xuống dốc)

Gradient descent là một hàm cập nhật các thông số mới cho các tham số, sao cho khi tính toán với các tham số đó thì sẽ cho ra độ lỗi nhỏ nhất có thể. Tổng quát hơn, mục đích của Gradient Descent là đi tìm global optimal (giá trị nhỏ nhất) của các hàm tính độ lỗi mô hình Deep learning (như Loss function).

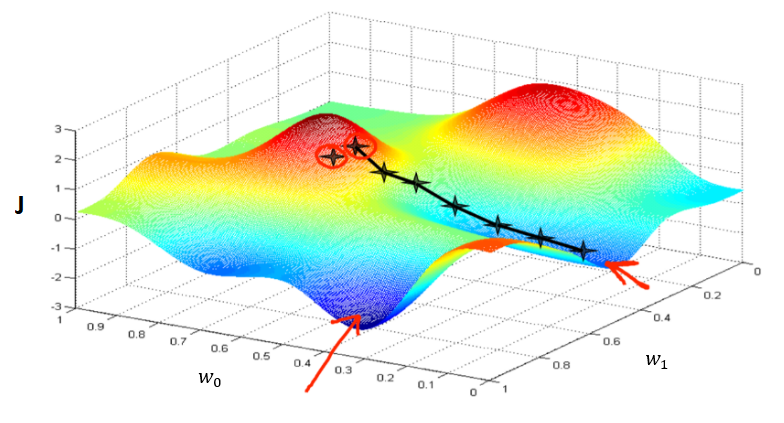
Gradient descent thực hiện tính toán dựa trên đạo hàm của các hàm tính độ lỗi theo từng trọng số và từ đạo hàm đó xác định chiều giá trị đi lên hay đi xuống của hàm tính độ lỗi mà cập nhật các tham số sao cho giá trị của hàm tính độ lỗi giảm xuống.

Công thức cập nhật tham số của Gradient descent:

Trong đó:

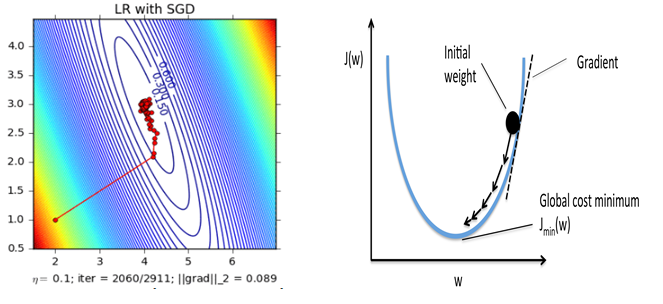
* j là thứ tự trọng số trong mô hình Deep learning
* là tham số quyết định tốc độ học của mô hình (learning rate).

Lưu ý: cần cập nhật giá trị tham số song song với nhau



0‑III Minh họa hoạt động của Gradient descent.

Mô hình đồi núi là những nơi mà với các trọng số khác nhau mà cho ra độ lỗi khác nhau. Đường đi của Gradient descent (đường đen) là từ nơi có độ lỗi cao (vùng màu đỏ) đến nơi mà độ lỗi thấp (vùng màu xanh). Trong hình thì đường Gradient Descent đi xuống local optimal (vùng trũng cục bộ, nơi mà độ lỗi tuy nhỏ nhưng chưa phải nhỏ nhất) thay vì global optimal (vùng trũng toàn bộ, nơi mà độ lỗi có giá trị nhỏ nhất).



0‑IV Minh họa 2D thuật toán Gradient descent

Nhưng do Gradient Descent chỉ phù hợp với những mô hình Neural Network đơn giản (chỉ gồm 1 lớp đầu vào và 1 lớp đầu ra). Đối với các mô hình Deep Neural Network với nhiều hidden layer thì tốc độ sẽ rất chậm vì phải tính toán đạo hàm của hàm mất mát theo từng trọng số của từng layer. Do đó trong mô hình DNN còn có cách cập nhật tham số nhanh hơn đó là Backpropagation.

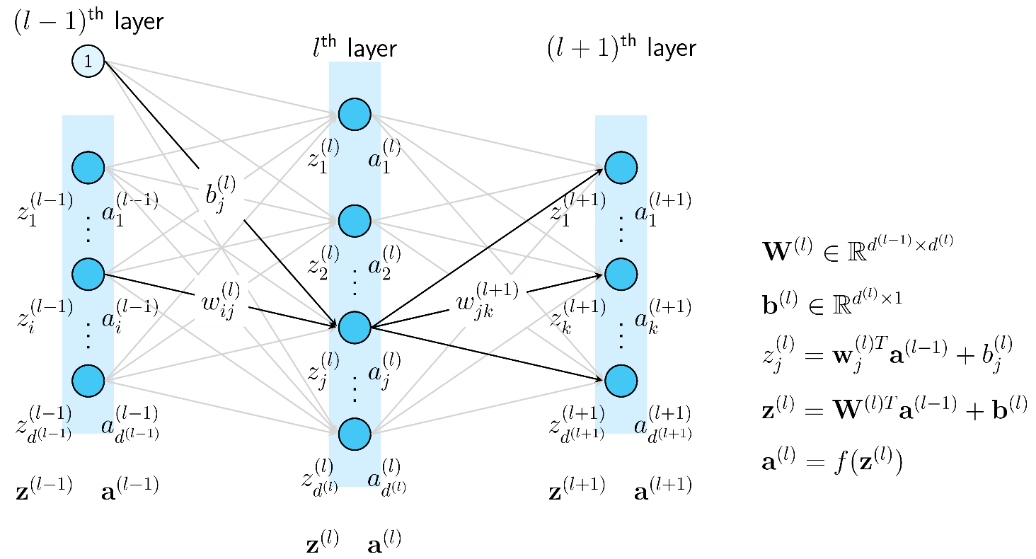
#### 2.1.3. Backpropagation (Đạo hàm ngược)

Backpropagation là một phương pháp cập nhật tham số nhanh hơn Gradient Descent được sử dụng phổ biến trong các mô hình Deep Neural Network. Vì nó thực hiện tính toán đạo hàm dựa trên những giá trị đã được tính của các node trong quá trình feed-forward (quá trình tính giá trị các node tuần tự từ layer đầu đến layer cuối cùng) nên tốc độ tính toán đạo hàm sẽ nhanh hơn.

Công thức của Backpropagation là:

Trong đó:

* (L): layer hiện tại đang xét
* i, j: trọng số thứ i trong node thứ j của layer hiện tại.
* là một đại lượng dễ tính toán
* vì



Mô phỏng cách tính Backpropagation

#### 2.1.4. Accuracy, F1 score, recall, precision

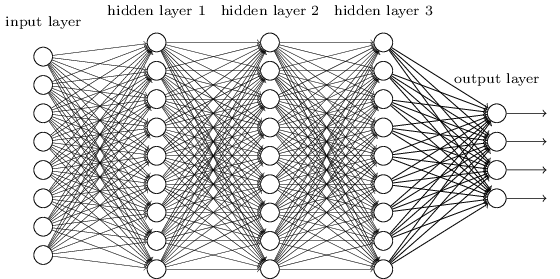
Các thông số đánh giá độ chính xác của một mô hình Deep learning như accuracy, F1 score, recall, precision thường được dùng trong mô hình Deep learning để đánh giá chi tiết độ chính xác của mô hình Deep learning, từ đó đưa ra hướng tối ưu cho mô hình. Ví dụ với mô hình phân 2 lớp với lớp dương là **positive** và lớp âm là **negative**.

* **Accuracy**: Thông số cơ bản để đánh giá độ chính xác mô hình, được tính bằng tổng số dự đoán chính xác chia cho tổng số dự đoán thực tế. Nhược điểm của accuracy là không thể tính được số lượng dự đoán nhầm của từng lớp.
* **Recall**: Sinh ra để bù đắp nhược điểm của accuracy, thông số recall cho biết tỉ lệ chính xác khi phân lớp âm là bao nhiêu. Được tính bằng số lượng dự đoán đúng lớp dương (true positive) chia cho chính nó cộng với số lượng kết quả dương thực tế nhưng bị phân nhầm vào lớp âm (false negative). Với mô hình phân lớp dương tốt nhất thì recall sẽ bằng 1 (false negative=0).
* **Precision**: Giống như recall, precision phản ánh chi tiết độ chính xác của accuracy, nhưng ngược lại với recall là thông số này cho ta biết tỉ lệ chính xác khi phân vào lớp dương của mô hình. Được tính bởi số dự đoán đúng khi phân lớp dương (true positive) chia cho chính nó cộng với số lượng kết quả âm thực tế nhưng bị phân nhầm vào lớp dương (false positive). Với mô hình phân lớp tốt nhất thì precision sẽ bằng 1 (false positive=0).
* **F1 score**: Thông số F1 score được dùng để đánh giá cả 2 thông số recall và precision, từ đó cho thấy khía cạnh chính xác hơn của mô hình. Vì một số mô hình vì chênh lệch lượng data huấn luyện giữa các loại mẫu khác nhau mà khi training thì accuracy sẽ khó đưa ra kết quả chính xác. Do đó F1 score sẽ đánh giá cả 2 thông số recall và precision để đưa ra kết quả thực tế hơn cho mô hình.

### 2.2. Một số mô hình Deep Learning cơ bản

#### 2.2.1. Deep Neural Network (Mạng nơ-ron sâu)

**Deep Neural Network (DNN)** là mạng nơ-ron cơ bản của Deep Learning. DNN bao gồm các cấu trúc cơ bản của mạng Neural Network với các lớp layer **kết nối toàn vẹn (fully-connected – chỉ những lớp layer mà các node trong layer đó được kết nối đầy đủ với các node của layer trước và sau liền kề nó)**, input đầu vào là các mẫu training đã được trích xuất đặc trưng, chuẩn hóa và đã được gán nhãn, sau khi qua xử lí sẽ cho ra kết quả dự đoán cuối cùng nằm trong một trong những kết quả đã cho trước. DNN thường được cùng trong các mô hình cơ bản, bước xử lí cuối cùng của dữ liệu sau khi đã được trích xuất đặc trưng. Được ứng dụng vào các mô hình phân loại theo nhãn, dự đoán, phân loại cây, hoa quả, phát hiện ung thư, dự báo thời tiết, …



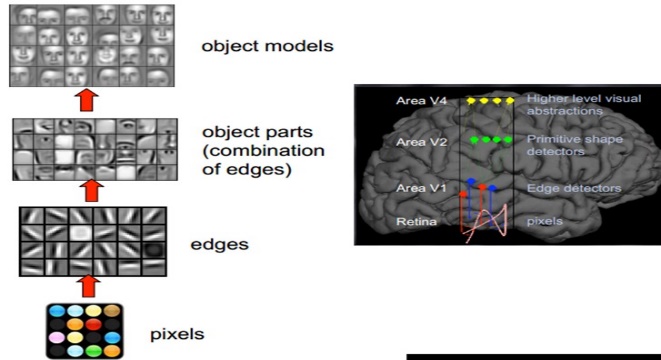
0‑V Mạng Deep Neural Network cơ bản

#### 2.2.2. Convolution Neural Network (Mạng nơ-ron tích chập)

**Convolution Neural Network (CNN)** cũng được xem là một trong những mạng nơ-ron cơ bản. Thông tin đầu vào thường là những dữ liệu chưa được trích xuất và cùng đơn vị (hình ảnh, đoạn thông tin nhiễu). Nhờ vào các lớp **convolution (tích chập)** và các **lớp pooling** có khả năng trích xuất đặc trưng từ lượng thông tin đầu vào mà CNN được sử dụng phổ biến trên hình ảnh. Thường được sử dụng để phân tích hình ảnh, nhận dạng khuôn mặt, vật thể, con người, gán nhãn cho các vật thể trong ảnh, … Cơ bản thì cấu trúc CNN sau khi nhận các thông tin đầu vào đã được chuẩn hóa thì sẽ sử dụng các lớp convolution và pooling để trích xuất các đặc trưng quan trọng nhất trên thông tin, sau cùng các thông tin sẽ được trải phẳng “flatten” thành vector và được đưa vào mô hình DNN để xử lí và cho ra kết quả cuối cùng. CNN được xây dựng dựa trên “**Nguyên tắc thị giác”** của con người.

##### 2.2.2.1. Nguyên tắc thị giác

Nguyên tắc thị giác của con người như sau: bắt đầu từ tín hiệu ban đầu (các điểm ảnh đi vào đồng tử, các pixels), sau đó là xử lý sơ bộ (một số tế bào ở vỏ não tìm thấy các cạnh và hướng - convolution), sau đó trừu tượng hóa (não phán đoán rằng hình dạng của vật thể trước mắt là một hình tròn - pooling) và sau đó trừu tượng hóa thêm nữa (phát hiện các đặc trưng như núm thắt bóng bay, dây buộc, và cuối cùng từ những đặc trưng đấy được não xử lí, bộ não dự đoán từ hình dạng vật thể trước mắt là một quả bóng bay).



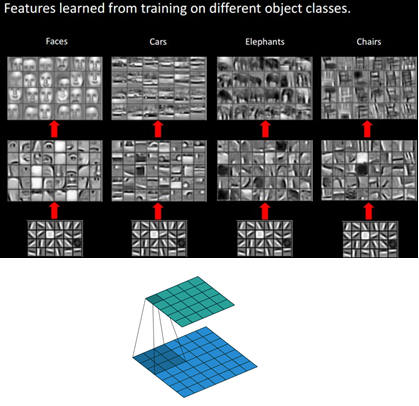
0‑VI Mô tả nguyên tắc thị giác

Từ những pixels đầu vào, qua các lớp convolution và pooling dần nhận ra các góc cạnh, rồi nhận dạng ra các bộ phận khuôn mặt, và từ đấy nhận ra khuôn mặt người.

Đối với các đối tượng khác nhau, thị giác của con người cũng tự nhận thức bằng cách phân loại theo các bộ phận khác nhau.

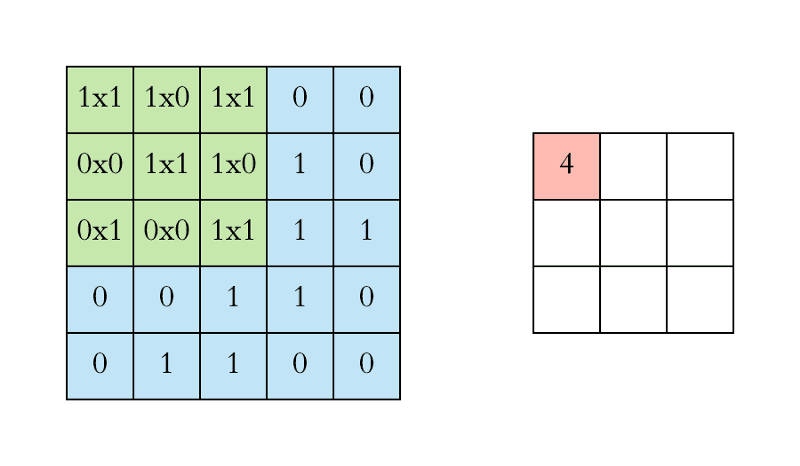
##### 2.2.2.2. Thuật toán Convolution và Pooling

**Convolution** gồm 2 khái niệm khác là **Convolution Filter** và **Convolutional Layer**. Trong mạng Neural Network thông thường, từ input, ta cho qua các hidden layer ra được output. Với CNN, Convolutional Layer cũng chính là hidden layer, khác ở chỗ, Convolutional Layer là một tập các **feature map** và mỗi feature map này là một bản scan của input ban đầu, nhưng được trích xuất ra các feature/đặc tính cụ thể. Scan như thế nào thì lại dựa vào Convolution Filter hay kernel. Đây là một ma trận sẽ quét qua ma trận dữ liệu đầu vào, từ trái qua phải, trên xuống dưới, và nhân tương ứng từng giá trị của ma trận đầu vào mà ma trận kernel rồi cộng tổng lại, đưa qua activation funciton (sigmoid, relu, elu, ...), kết quả sẽ là một con số cụ thể, tập hợp các con số này lại là 1 ma trận nữa, chính là feature map.



0‑VII Minh họa cơ chế convolution

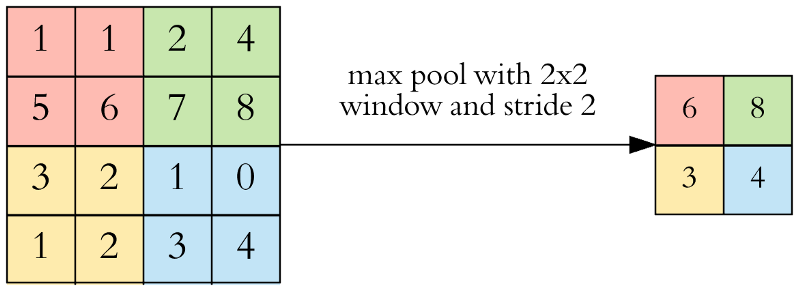
Ma trận phía trên là feature map, kết quả của tích chập của ma trận phía dưới (input data), mỗi feature map có trọng số khác nhau sẽ lấy ra những đặc trưng khác nhau từ input data.



0‑VIII Ví dụ minh họa 2D thuật toán convolution

Công thức thuật toán convolution

**Pooling** có chức năng làm giảm số **hyperparameter (tham số)** mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh overfitting (mô hình hoạt động tốt trên tập traning nhưng hoạt động kém hiệu quả trên tập testing). Loại pooling ta thường gặp nhất là **max pooling**, lấy giá trị lớn nhất trong một pooling window. Pooling hoạt động gần giống với convolution, nó cũng có 1 cửa sổ trượt gọi là pooling window, cửa sổ này trượt qua từng giá trị của ma trận dữ liệu đầu vào (thường là các feature map trong convolutional layer), chọn ra một giá trị từ các gía trị nằm trong cửa sổ trượt, với max pooling sẽ lấy giá trị lớn nhất.



0‑IX Minh họa thuật toán Max Pooling



0‑X Cấu trúc đầy đủ của một mạng Convolution Neural Network

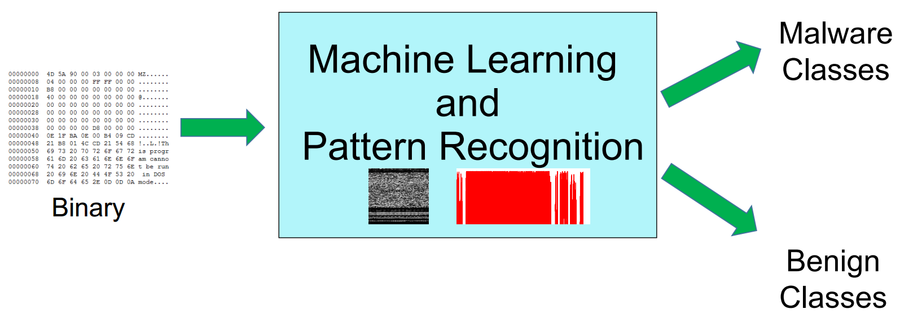
# III. Ứng dụng Deep Learning vào bài toán phát hiện mã độc

## 1. Giới thiệu tổng quát

Trước đây, các phần mềm chống virus và malware có một nhược điểm là chỉ có khả năng quét, phát hiện và tiêu diệt những mối nguy hiểm đã được biết đến (thông qua các bản vá từ nhà phát triển). Do đó, tin tặc có thể lợi dụng điều này để “lách” phần mềm chống virus chỉ bằng cách tạo ra các biến thể, chứ chưa cần tạo ra những loại virus mới. Nhưng trong tương lai, việc này sẽ sớm được giải quyết, bởi những nhà phát triển phần mềm chống virus hiện nay đang đi theo xu hướng cho phần mềm hoạt động dựa trên trí tuệ nhân tạo. Kỹ thuật này sử dụng các mạng lưới thần kinh nhân tạo (như Deep learning) để nhận biết dấu hiệu của những đoạn mã độc, thông qua việc “học” hàng triệu mẫu dữ liệu chứa malware và không phải malware.

Khả năng nhận biết chính xác của Deep learning khiến nó được nghiên cứu ứng dụng trong rất nhiều lĩnh vực, và các trình nhận diện virus cũng nằm trong số đó. Những năm gần đây, các chuyên gia trong lĩnh vực an ninh mạng dành khá nhiều công sức nghiên cứu để đưa kỹ thuật Deep learning vào các sản phẩm thực tế.

Thay vì cố gắng viết một chương trình, lập trình viên sẽ phát triển một thuật toán mà máy tính có thể sử dụng để xem xét hàng trăm hoặc hàng ngàn mẫu thử khác nhau (bao gồm cả file hợp pháp và virus, malware). Sau đó, các máy tính sử dụng “kinh nghiệm” đã học được để nhận diện trên các file mới không có trong data huấn luyện (có thể là virus hoặc không).



0‑I Minh họa phân tích và phân lớp dữ liệu sử dụng Deep Learning (Machine Learning)

## 2. Sản phẩm “Ứng dụng Deep learning vào bài toán phát hiện mã độc từ file thực thi trên hệ điều hành Windows”

## 2.1. Hacker, tin tặc và hệ điều hành Windows, định dạng Portable Execute (PE - EXE)

**Hệ điều hành Windows** được **Microsoft** cho ra mắt phiên bản đầu tiên là **Windows 1.0** (1985). Qua nhiều năm phát triển hiện nay phiên bản được sử dụng nhiều nhất đó là **Windows 10** (800 triệu người dùng, thị phần 40% vào 2019), theo sau là **Windows 7** (700 triệu người dùng, thị phần 38% vào năm 2019). Windows cũng là hệ điều hành được sử dụng nhiều nhất trên thế giới, từ những chuyên gia cho đến người không chuyên, được dùng trong rất nhiều công việc từ máy tính cá nhân, quản lí ngân sách, khách hàng cho đến vận hành các hệ thống lớn… Do đó đây là miếng mồi ngon cho các hacker, tin tặc viết ra những con virus, malware nhắm vào hệ điều hành này.

Một trong những định dạng file có khả năng chứa virus nhất đó là định dạng **PE** (Portable Execute – file thực thi tại chỗ) mà trong đó phổ biến nhất là định dạng **EXE**, là một định dạng file thực thi PE có thể hoạt động trên nhiều nền tảng như DOS, OpenVMS, Microsoft Windows, Symbian hoặc OS/2. Do đó sản phẩm nghiên cứu sẽ đào sâu vào phân tích file PE.

## 2.2. Giải pháp phát hiện mã độc từ tệp thực thi PE

Do tính chất cấu trúc mà các định dạng khác của PE như DLL, SYS,… không có quá nhiều khác biệt giữa cả 2 loại malware và benign (mã độc và hợp pháp) nhưng chúng đều phải thực thi qua EXE do đó sản phẩm sẽ đào sâu vào phân tích file EXE.

Trong Deep learning có thể thấy rằng khi số lượng **features** (đặc trưng) lớn sẽ làm giảm hiệu suất và đôi khi là chất lượng của mô hình học. Lượng features quá nhiều sẽ khiến cho quá trình huấn luyện và phân lớp dữ liệu tốn kém về mặt tài nguyên cũng như thời gian xử lý thậm chí nếu nhiều features phổ biến sẽ đẫn đến dư thừa gây nhiễu và ảnh hướng đến chất lượng khi xây dựng mô hình, chính vì vậy bài toán đặt ra và cần thiết là làm sao loại bỏ được các features gây nhiễu và chọn được một tập đặc trưng đại diện tốt nhất mà vẫn đảm bảo độ chính xác hiệu quả của mô hình dự đoán. Do đó mô hình lần này sử dụng cách trích xuất các thuộc tính, thông số của file EXE và lọc ra những thuộc tính có tính chất thống kê, sau đó chuẩn hóa và đưa vào mô hình Deep learning để tiến hành phân lớp, mục đích của giải pháp này là để giảm thiểu lượng features đầu vào, giảm thời gian huấn luyện nhưng vẫn đạt được hiệu suất cao.

## 2.3. Các công cụ sử dụng

Về môi trường thực hiện, sản phẩm được làm tương thích với hệ điều hành Windows, chạy trực tiếp trên cửa sổ **Command Prompt (cmd.exe)**, IDE thực hiện là **Sublime Text**.

Yêu cầu môi trường: Cài đặt **Python 3.6.8**, các gói thư viện Python: **tensorflow, scikit-learn, matplotlib, pandas, numpy, tqdm, jupyter, pefile**.

Code demo sẽ được thực hiện trên **Jupyter notebook** (nằm trong gói thư viện jupyter của Python chức năng hỗ trợ viết code, đánh dấu và thực thi code Python trên nền trình duyệt web) và **Google Colab** (Jupyter notebook phiên bản chạy trên nền đám mây sử dụng phần cứng của Google).

## 2.4. Phương pháp thực hiện

### 2.4.1. Sưu tầm data huấn luyện

Sưu tầm data huấn luyện:

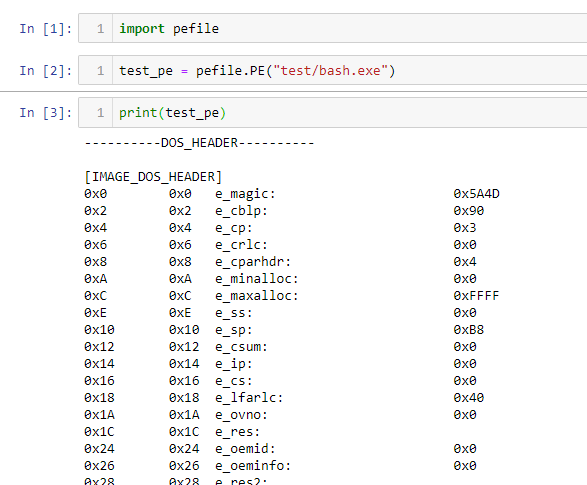
* **Malware** data được sưu tầm từ trang web VirusShare (virusshare.com) package 134th, được lọc ra những file thực thi PE (96724 files).
* **Benign** data được sưu tầm từ tất cả file thực thi PE từ hệ điều hành Windows 2008, Windows XP và Windows 7 (32/64 bit) (41323 files).

Tổng: 138048 files.

Data được sưu tầm và tổng hợp bởi Tek (randhome.io).

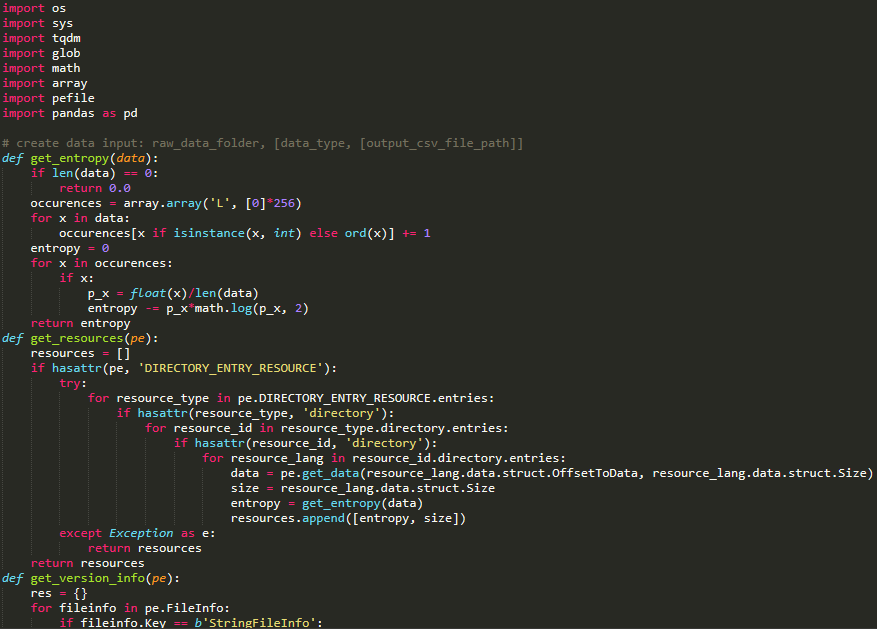
### 2.4.2. Trích xuất thuộc tính file thực thi PE

Ngôn ngữ thực hiện là Python. Do đó sản phẩm sẽ sử dụng thư viện **pefile** của Python để phân tích cấu trúc file PE, nhằm lấy ra những thuộc tính của file PE.

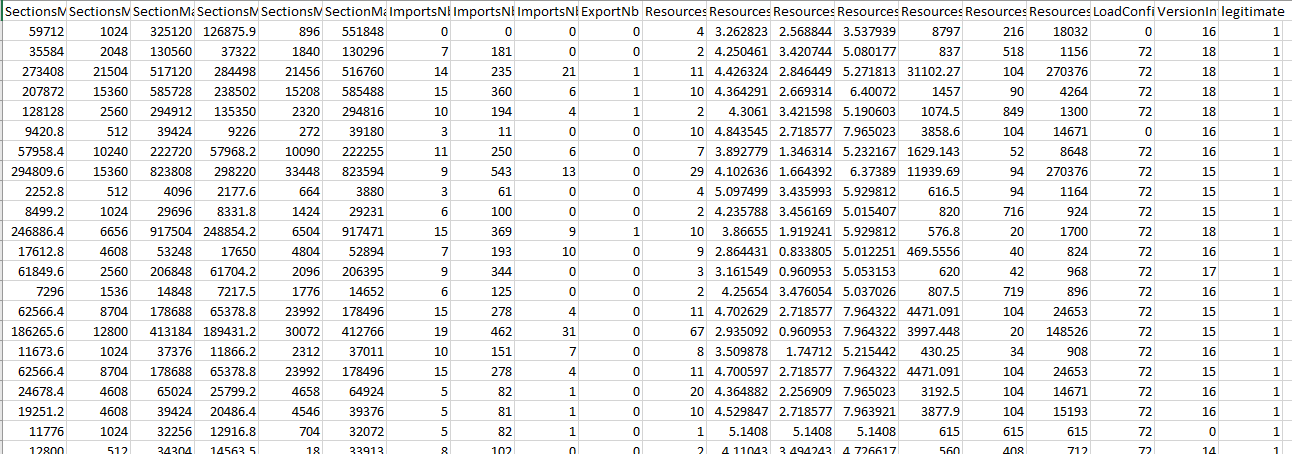


0‑II Sử dụng thư viện pefile để xem được các thuộc tính của file PE

Áp dụng pefile nhóm đã viết code để trích xuất toàn bộ thông tin của data huấn luyện vào một file CSV (code được viết trong file data.py).



0‑III Code của trích xuất data và gán nhãn cho data



0‑IV Data sau khi trích xuất vào file CSV (56 features).

Mỗi dòng chứa thông tin của một file, thuộc tính cuối cùng chỉ nhãn của file đó (file hợp pháp thì legitimate=1, malware thì legitimate=0)

Sau khi có file CSV được gán nhãn. Nhóm thực hiện việc lấy data từ file CSV đưa vào bộ nhớ, data được chia thành 2 tập: tập training (80% data) và testing (20% data). Mỗi tập gồm 2 phần: X (ma trận chứa các thông số thuộc tính của data) và y (ma trận chứa nhãn của data tương ứng):



0‑V phân chia dữ liệu training và test

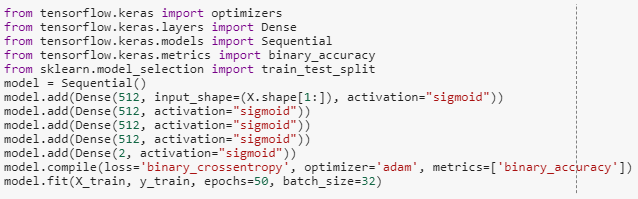
Sau khi có data cho việc huấn luyện, bước tiếp theo là xây dựng mô hình Deep learning.

### 2.4.3. Xây dựng mô hình Deep learning

Framework: Mô hình được xây dựng trên thư viện **Keras** với backend là **Tensorflow**.

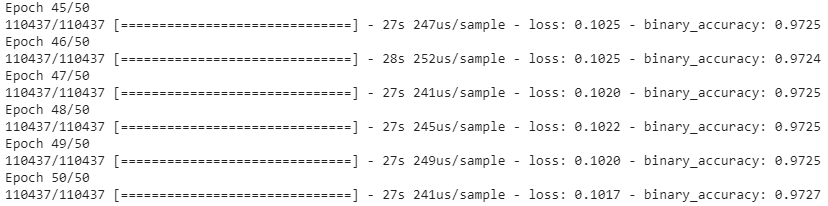
Do đặc trưng dữ liệu nên nhóm sử dụng hai mô hình cơ bản đó là Deep Neural Network (DNN) và Convolution Neural Network (CNN).

Deep Neural Network: Nhóm nghiên xây dựng một mô hình DNN cơ bản với 5 lớp (1 input layer, 3 hidden layers, 1 output layer) với số lượng node của từng hidden layer là 512 (nhằm cho việc phân tích data chính xác hơn). Vì đây là mô hình phân 2 lớp cơ bản nên đã sử dụng activation function là hàm sigmoid, hàm tính độ lỗi là **binary\_crossentropy** (tương tự Loss function) và hàm cập nhật tham số mô hình là **adam** (thuật toán mới được đưa vào năm 2014 giúp cho mô hình tiến về global optimal nhanh hơn). Mô hình được training 50 lần trên tập training data với **batch\_size** là 32 (xử lí 32 mẫu trong một lần cập nhật nhằm giảm ảnh hưởng của các mẫu nhiễu và tăng tốc độ training).

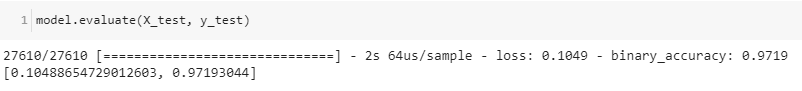


0‑VI Code Python xây dựng mô hình DNN

Kết quả: Mô hình đạt được độ chính xác **97.27%** trên tập training data và **97.19%** trên tập testing data (gần như không bị hiện tượng overfitting).

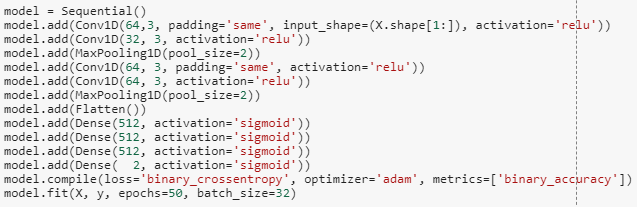


0‑VII Kết quả trên training data



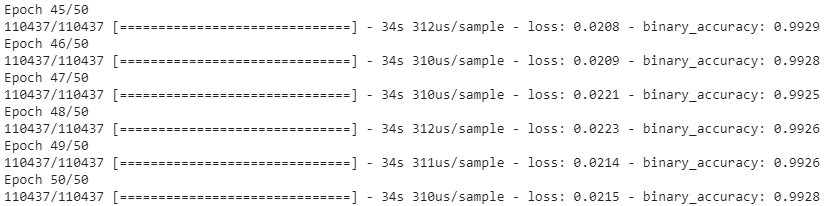
0‑VIII Kết quả trên testing data

Convolution Neural Network: CNN đòi hỏi việc phải chuẩn hóa dữ liệu về cùng đơn vị trước khi đưa vào xử lí. Do đó nhóm đã scale dữ liệu của từng feature về khoảng (0, 1) theo đơn vị của từng feature. Mô hình bao gồm 4 convolution layers (1 dimensional – mỗi đầu vào là vector thay vì ma trận), 2 max pooling layers, 1 **flatten layer** (trải phẳng ma trận thành vector để đưa vào mạng DNN), cuối cùng được đưa vào DNN với 3 hidden layers (512 nodes mỗi layer) và 1 output layer. Mọi thuật toán compile được giữ nguyên.

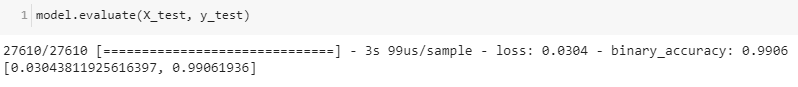


0‑IX Code Python xây dựng mô hình CNN

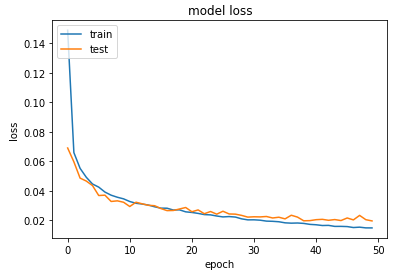
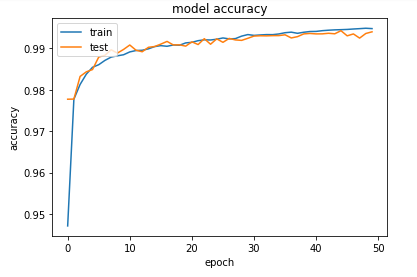
Kết quả: Mô hình đạt được độ chính xác cao nhất **99.29%** trên tập training data và **99.06%** trên tập testing data (gần như không bị hiện tượng overfitting).



Kết quả trên training data



Kết quả trên testing data

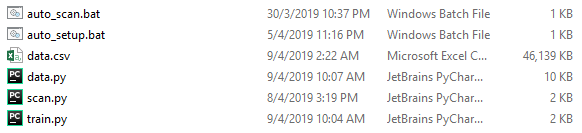


Kết quả accuracy và loss trên mô hình CNN (cho thấy gần như không bị overfitting)

Đánh giá: Dựa vào accuracy trên cả tập training data và testing data thì CNN là mô hình chiếm ưu thế hơn, nhưng DNN lại có thời gian training thấp hơn trên tập data huấn luyện.

Kết luận: Do thời gian mô hình tiến đến global optimal khá thấp (chỉ sau khoảng 40 epochs) nên nhóm quyết định chọn CNN làm mô hình chính để phân lớp file thực thi với 2 loại malware và benign.

## 2.5. Xây dựng sản phẩm



0‑X Cấu trúc cơ bản của sản phẩm

Trong đó:

* **data.py**: code Python có chức năng trích xuất thuộc tính file PE từ thư mục và tự động gán nhãn cho data.
* **train.py**: code Python chứa mô hình CNN dùng để training và lưu mô hình vào một file riêng để dùng nhận diện malware.
* **scan.py**: code Python có chức năng nhận diện mã độc theo thư mục
* **auto\_setup.bat**: file thực thi các câu lệnh trên nền DOS (Command Prompt) có chức năng tự động cài đặt các gói Python cần thiết cho mô hình (yêu cầu cài đặt Python sẵn).
* **auto\_scan.bat**: file thực thi các câu lệnh trên nền DOS (Command Prompt) có chức năng tự khởi động python và chạy scan.py (xây dựng cho người dùng cuối).
* **data.csv**: file text CSV chứa data training.