Cải thiện khả năng phát hiện tấn công mạng bằng kỹ thuật học sâu

Tô Trọng Tín[1], Trần Văn Lăng[2, 3]

[1]Học viện Công nghệ Bưu chính viễn thông, [2]Viện Cơ học và Tin học ứng dụng, VAST, [3]Đại học Nguyễn Tất Thành
tiznto@gmail.com, langtv@vast.vn

Tóm tắt

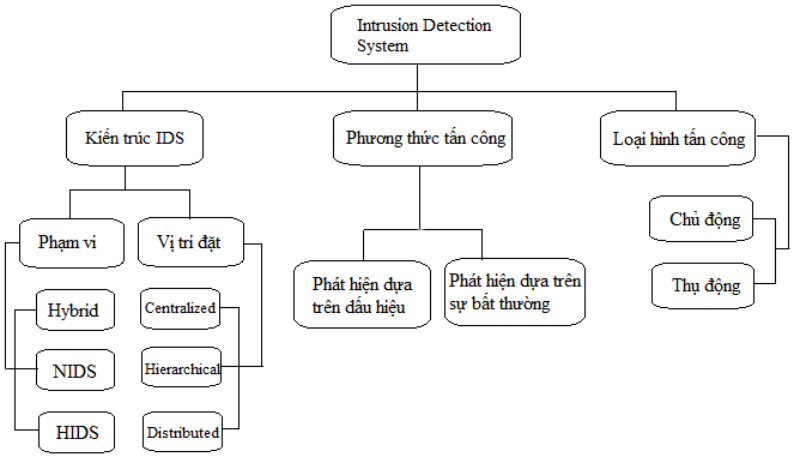
Hệ thống phát hiện tấn công mạng (Intrusion Detection System - IDS) là một phần mềm bảo mật được thiết kế để cảnh báo một cách tự động cho các quản trị viên khi có ai đó hoặc cái gì đó đang cố gắng xâm nhập hệ thống thông qua các hoạt động nguy hiểm hoặc vi phạm chính sách bảo mật. Nhiều nghiên cứu đã áp dụng thành công các thuật toán máy học để hệ thống IDS có khả năng tự học và cập nhật các cuộc tấn công mới. Nhưng để hạn chế báo động nhầm và tăng khả năng dự đoán các cuộc tấn công, thì ngoài khả năng tự quyết định, IDS cần phải có tư duy phân tích. Một khả năng mà các nhà nghiên cứu gọi là học sâu. Bài viết này đề cập đến học sâu như một hướng tiếp cận mới có thể giúp hệ thống IDS cải thiện độ chính xác và tăng tốc độ phân tích khi đầu vào quá lớn. Với việc áp dụng mạng thần kinh sâu như mạng đa lớp ẩn (Multilayer Perceptron - MLP) và mạng neural hồi quy (Recurrent Neural Network – RNN) trên tập dữ liệu KDD99 được sử dụng để đánh giá độ chính xác (Accuracy), độ lỗi phân lớp (MSE – Mean Squared Error) và ma trận hỗn loạn (Confusion Matrix). Hiệu quả đạt được là 98,2% với MLP và 99,04% với RNNs, so với 92,6% của SVM và 88.46% của Naïve Bayes..

Nhận 19.12.2017
Được duyệt 21.01.2018
Công bố 01.02.2018

Từ khóa
IDS, mạng máy tính, mạng thần kinh, học sâu, máy học

1. Giới thiệu

Trước sự tiến bộ của thông tin và truyền thông, những mối đe dọa an ninh mạng cũng tăng lên rất nhiều, hệ thống phát hiện tấn công mạng (IDS) là một trong những vấn đề bảo mật rất đáng quan tâm, IDS hoạt động bằng cách theo dõi hoạt động của hệ thống thông qua việc kiểm tra các lỗ hổng bảo mật, tính toàn vẹn của các tệp tin và tiến hành phân tích các mẫu dựa trên các cuộc tấn công đã biết, nó cũng tự động theo dõi lưu lượng mạng để tìm kiếm các mối đe dọa mới nhất có thể dẫn đến một cuộc tấn công trong tương lai.



Hình 1. Phân loại hệ thống IDS

Hình 1 là các loại hệ thống IDS được Pathan (2014) [1] phân theo ba tiêu chí lần lượt là kiến trúc hệ thống, phương thức phát hiện xâm nhập và các loại hình tấn công.

Hầu hết các nhà nghiên cứu đều tập trung vào nghiên cứu kỹ thuật phát hiện của IDS. Họ đã cố gắng áp dụng các kỹ thuật máy học với hệ thống này và đạt được những thành công nhất định. Peter Scherer et al. (2011) [2] đã ứng dụng kỹ thuật SVMs và các thuật toán clustering vào việc cải thiện các thông số dự đoán; thí nghiệm đã đạt những kết quả khả quan, nhưng vấn đề khi sử dụng đơn lớp SVM thì rất khó phản ánh được độ tương quan giữa các lớp tấn công. Các tác giả Hoàng Ngọc Thanh, Trần Văn Lăng, Hoàng Tùng (2016) [3] đã đề xuất một cách xây dựng bộ phân lớp lai đa tầng trên cơ sở kiến trúc của mô hình phân đa lớp truyền thống One-vs-Rest trong đó luồng dữ liệu đi qua sẽ được sàn lọc qua các tầng thuật toán như SVM, ANN..Mỗi tầng của thuật toán chuyên dụng được dùng để phân tích một loại tấn công tương ứng. Họ đã xác nhận rằng sử dụng mô hình đa lớp sẽ cho ra kết quả tốt hơn mô hình đơn lớp. Qua các thí nghiệm và nghiên cứu trên chúng ta có thể hình dung một mô hình có thể là tối ưu để cải thiện khả năng phát hiện xâm nhập bao gồm nhiều lớp xử lý và trong mỗi lớp chứa một công cụ để quyết định từng dấu hiệu của dữ liệu đầu vào.

Bài báo này mở rộng nghiên cứu sang các kỹ thuật học tập sâu (Deep learning); đây là một kỹ thuật mới đang có rất nhiều ưu điểm và tính năng cần nghiên cứu khai thác với 2 điểm chính: Thứ nhất, kết quả từ các thuật toán học sâu không chịu sự chi phối của việc định nghĩa các đặc trưng; điều đó có nghĩa là các dữ liệu đầu vào không cần phải qua công đoạn tiền xử lý và trích chọn đặc trưng, chúng ta có thể đưa vào gần như là dữ liệu thô. Thứ hai, bản thân của các mạng học tập sâu vẫn sử dụng các thuật toán thống kê với qui mô siêu lớn, khi đưa vào càng nhiều dữ liệu thì độ chính xác càng cao. Xuejun Gu et al. [4] đã chỉ ra hiệu quả của mạng neural (thần kinh) sâu trong xử lý dữ liệu phi tuyến thời gian thực; theo đó mạng được chú ý đến gồm ba mô hình: 1) Multilayer-Perceptrons (MLP), 2) Mạng neural tái phát (RNN), 3) Mạng neural tích chập (CNN); trong đó mô hình MLP và RNN là rất hiệu quả trong việc phân tích chuỗi dữ liệu tuần tự, liên tục và mang nhiều đặc trưng dữ liệu [5]. Vì vậy trong bài viết này áp dụng một mô hình lai của hai mạng trên và huấn luyện với bộ dữ liệu KDD99 để kiểm tra hiệu suất. Thông qua việc huấn luyện tìm ra một bộ tham số đạt hiệu quả cao nhất và xác nhận được tỉ lệ phát hiện chính xác cũng như tỉ lệ phân lớp lỗi.

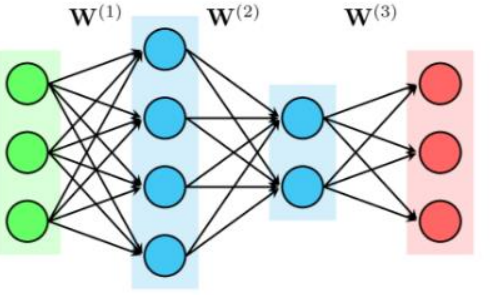
Bài viết này gồm có 4 phần, các phần còn lại của bài báo như sau: trong phần II trình bày mô hình mạng MLP, mạng RNNs, thuật toán học lan truyền ngược, cách bố trí thí nghiệm, các phương pháp đánh giá và kết quả trình bày trong phần III; đánh giá và kết luận nêu trong phần IV

2. Mô hình học sâu

Nếu nói học máy là một phạm trù của trí tuệ nhân tạo (AI), chúng lấy một số ý tưởng cốt lõi của AI và tập trung vào giải quyết các vấn đề thực tế với các mô hình được thiết kế để bắt chước việc ra quyết định của con người thì học sâu tập trung vào các vấn đề trọng tâm hơn về một tập hợp các công cụ và kỹ thuật máy học, cũng như việc áp dụng chúng để giải quyết các vấn đề đòi hỏi tư duy. Về cơ bản, học tập sâu liên quan đến việc nhập vào một hệ thống máy tính rất nhiều dữ liệu, chúng có thể sử dụng để đưa ra các quyết định về các dữ liệu khác thông qua việc học ở nhiều cấp độ tương ứng với các mức độ trừu tượng khác nhau với các lớp, qua đó hình thành một hệ thống các tính năng phân cấp từ thấp đến cao.

2.1 Mạng Multilayer-Perceptron (MLP)

Mạng neural sâu (DNN) là một mạng neural nhân tạo với nhiều lớp ẩn giữa lớp đầu vào và đầu ra. Khác với các mạng neural thường; các mạng neural sâu có thể mô hình mối quan hệ phi tuyến một cách phức tạp, chẳng hạn như phát hiện và phân tích đối tượng để tạo ra các mô hình hỗn hợp; mà các đối tượng này xem như thành phần được xếp lớp của các dữ liệu ban đầu. Các lớp ẩn cho phép lấy các thành phần của các đặc điểm từ các lớp thấp hơn, mô hình hóa dữ liệu phức tạp hơn so với mạng lưới nông khi thực hiện việc tương tự. Một mạng Multilayer-Perceptron (MLP) chính là mạng neural sâu.



Hình 2. Kiến trúc mạng MLP với 2 lớp ẩn

Hình 2 ví dụ mạng neural gồm ba lớp trong đó hai lớp ẩn và một lớp đầu ra (khi tính số lớp trong mạng neural ta lấy số lớp ẩn cộng cho *1)*, các ma trận $W^{(L)}$ đại diện cho các trọng số (weight) trong từng lớp, mỗi lớp có một hệ số tự do gọi là bias - ký hiệu là $b^{(L)}$; bias và weight là hai đại lượng quan trọng cần tìm khi cần tối ưu mạng MLP cho một công việc nào đó. Output của các input được tính theo công thức:

$$z^{(L)} = \left(\boldsymbol{w}^{(L)T} z^{(L-1)} + \boldsymbol{b}^{(L)}\right)$$
$$a^{(L)} = f\left(z^{(L)}\right) \quad (1)$$
$$\hat{y} = a^{(L)}$$

Trong đó *a* kí hiệu cho output và *f* là hàm kích hoạt; hàm số được sử dụng nhiều nhất là hàm sigmoid và hàm tanh vì đạo hàm của chúng rất đẹp; nhưng những năm gần đây người ta phát hiện ra các hàm số này bị hạn chế vì chúng

không thể hiện hết được miền giá trị của các unit. Nếu input là một trị tuyệt đối của một số rất lớn thì gradient của nó rất gần với 0 hoặc -1; vì vậy các hệ số của unit sẽ không được cập nhật. Theo Krizhevsky et al. [6] hàm Rectified Linear Unit (ReLU) đang là một hàm số đơn giản và giúp tăng tốc độ huấn luyện của các thuật toán học tập sâu lên rất nhiều, công thức của nó là $f(s) = Max(0, s)$ nên gradient được tính toán rất nhanh với giá trị là 1, nếu đầu vào lớn hơn 0 và bằng 0 nếu s = 0.

Giả sử tính được một điểm dữ liệu y_t sau vòng lập thứ *t*, cần tính độ mất mát J(W, b, X, Y) của y_t và dùng một thuật toán huấn luyện để đưa y_t về càng gần giá trị y thực tế. Phương pháp phổ biến nhất để tối ưu MLP vẫn là Gradient Descent (GD) nhưng trong điều kiện tập dữ liệu lớn, liên tục với điểm dữ liệu nhiều chiều thì GD kém hiệu quả và quá cồng kềnh khi phải liên tục tính toán lại đạo hàm của hàm mất mát tại tất cả các điểm dữ liệu. Vì vậy cần dùng Root Mean Square Error (RMSE) để tính độ mất mát *J* trên từng điểm dữ liệu sau đó dùng phương pháp học Backpropagation cho hàm số Stochastic Gradient Descent (SGD) để tính đạo hàm theo ma trận $\boldsymbol{W}^{(L)}$, $\boldsymbol{b}^{(L)}$.

Các bước thực hiện như sau:

1. Với giá trị đầu vào *X*, tính giá trị đầu ra *Y*, với mỗi layer phải lưu lại một giá trị output là $a^{(L)}$.
2. Với ouput layer ta có

$$J(W, b, X, Y) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left\|y_n - a_n^{(L)}\right\|_2^2}$$

$$e^{(L)} = \frac{\partial J}{\partial z^{(L)}} \quad (2)$$

3. Từ (1) (2) suy ra:

$$\frac{\partial J}{\partial w^{(L)}} = a^{(L-1)} e^{(L)T}$$

$$\frac{\partial J}{\partial b^{(L)}} = e^{(L)}$$

4. Lan truyền ngược với L là L-1, L-2…1 ta có:

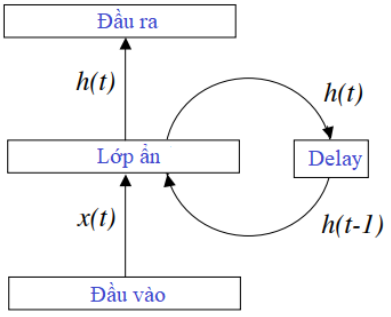
$$e^{(L)} = \left(w^{(L+1)} e^{(L+1)}\right) \Theta\, f'\left(z^{(L)}\right)$$

Trong đó Θ là hàm hadamard product là hàm lấy từng thành phần của hai vector nhân với nhau để được vector kết quả.

5. Cập nhật đạo hàm cho ma trận trọng số và bias.

2.2 Recurrent neural network (RNN)

Mạng neural tái phát, mạng neural hồi quy hay recurrent neural network (RNN) là một loại mạng neural nhân tạo được bổ sung một số trọng số để tạo ra các chu trình trên đồ thị mạng, qua đó cố gắng duy trì trạng thái cục bộ. Hình thức đơn giản nhất của mạng RNN chính là mạng MLP với các đơn vị kích hoạt trong các lớp ẩn được đưa trở lại mạng cùng với đầu vào.



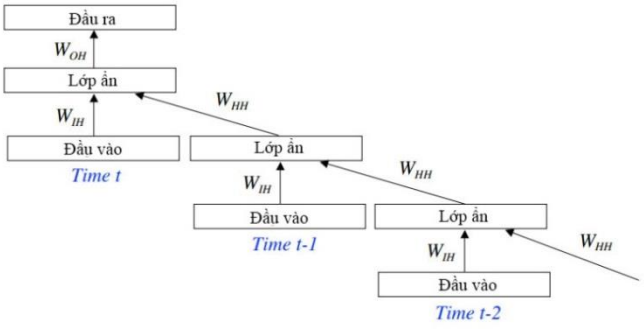
Hình 3. Mô hình fully recurrent neural network

Giả định rằng đầu vào và đầu ra của mạng RNN lần lượt là vectors *x(t)* và *y(t)*, ba ma trận trọng số là W_{xh}, W_{hh} và W_{hy} như Hình 3. Hàm kích hoạt unit ở lớp ẩn và lớp đầu ra là f_H và f_O, hành vi của mạng RNN có thể được mô tả như là một hệ thống động bằng cặp phương trình ma trận phi tuyến:

$$h_t = f_h\left(w_{xh}x(t) + W_{hh}h(t-1)\right)$$

$$y(t) = f_0\left(w_{h_y}h(t)\right)$$

Trong đó *f* là hàm phi tuyến, *h(t)* là tập các đơn vị kích hoạt ẩn được dùng để xác định trạng thái của mô hình. Trạng thái của một hệ thống động là một tập hợp các giá trị tóm tắt tất cả các thông tin về hành vi trong quá khứ của hệ thống cần thiết, để cung cấp mô tả duy nhất về hành vi tương lai của nó. Hình 4 mô tả mạng RNN được dàn trải ra theo từng bước thời gian.



Hình 4. Mạng RNN được dàn trải theo bước thời gian

Huấn luyện mạng RNN cũng tương tự như các mạng neural truyền thống, trong đó vẫn sử dụng RMSE để tính toán hàm mất mát theo mỗi bước thời gian và dùng phương pháp học lan truyền ngược để cập nhật các trọng số liên quan tuy nhiên giải thuật lan truyền ngược (backpropagation). Trong RNN có một sự thay đổi đó là đạo hàm tại mỗi đầu ra phụ thuộc không chỉ vào các tính toán tại bước đó mà còn phụ thuộc vào các bước trước đó; bởi vì các tham số trong mạng RNN được sử dụng chung cho tất cả các bước trong mạng.

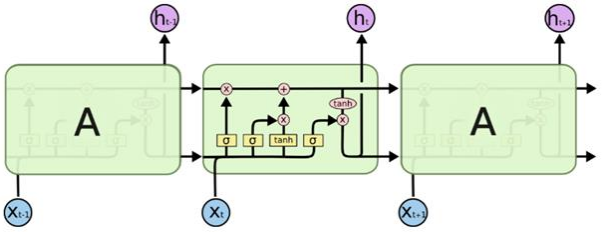
Nếu mạng huấn luyện một chuỗi dữ liệu với thời gian bắt đầu là t_0 và kết thúc ở bước thời gian t_1 thì tổng chi phí sẽ là tổng độ lệch chuẩn theo mỗi bước thời gian:

$$E_{total}(t_0, t_1) = \sum_{t=t_0}^{t_1} E_{sse/ce}(t)$$

và các trọng số được cập nhật theo hàm SGD:

$$\Delta w_{iJ} = -\eta \frac{\partial E_{total}(t_0, t_1)}{\partial w_{iJ}} = -\eta \sum_{t=t_0}^{t_1} \frac{\partial E_{sse/ce}(t)}{\partial w_{iJ}}$$

trong đó η là learning rate, Δw_{iJ} là tổng trọng số của tất cả các bước thời gian trước đó. Với công thức như vậy, có thể thấy được phần nào sự khó khăn khi huấn luyện mạng RNN, vì với các chuỗi dài ta cần phải truyền ngược lại thông qua rất nhiều tầng mạng. Ngoài ra khi huấn luyện bằng phương pháp lan truyền ngược liên hồi theo mỗi bước thời gian thì sẽ làm cho các gradient bùng nổ hoặc biến mất, Bengio et al. đã đề cập và giải quyết vấn đề này năm 1994 [7]. Một biến thể của mạng RNN có thể giải quyết vấn đề phụ thuộc xa được giới thiệu bởi Hochreiter & Schmidhuber (1997) [8], gọi là mạng bộ nhớ dài ngắn hạn (Long-short tearm memory – LSTM). Mạng LSTM được thiết kế theo kiến trúc dạng chuỗi tương tự như mạng RNN nhưng kiến trúc bên trong của LSTM có 4 tầng tương tác với nhau thay vì 1 tầng như mạng RNN (Hình 5). Việc nhớ thông tin trong thời gian dài là đặc tính của mạng với tầng trạng thái nên không cần phải huấn luyện với bất kỳ phương pháp nào.



Hình 5. 4 tầng mạng LSTM trong một bước thời gian *(nguồn: https://dominhhai.github.io)*

Các công thức ứng với các tầng mạng được thể hiện như sau:

$$i_t = \sigma(w_{x_i} x(t) + w_{hi} h(t-1) + w_{ci} c(t-1) + b_i) \quad (3)$$

$$f_t = \sigma\left(w_{x_f} xt + w_{hf} h(t-1) + w_{cf} c(t-1) + b_f\right) \quad (4)$$

$$c_t = f_t c(t-1) + i_c \tan h\left(w_{x_c} x(t) + w_{hc} h(t-1) + b_c\right) \quad (5)$$

$$O_t = \sigma\left(w_{x_0} X(t) + w_{h_0} h(t-1) + w_{co} c_t + b_0\right) \quad (6)$$

$$h_t = O_t \tan h(c_t) \quad (7)$$

Hàm σ là một hàm sigmoid, i, f, o và c tương ứng là cổng đầu vào, cổng quên, cổng đầu ra và đơn vị trạng thái. Ba cổng (*I, f, o*) là các cổng kiểm soát luồng thông tin, W_{ci}, W_{cf} và W_{co} biểu thị cho các ma trận trọng số của các kết nối. Bước đầu tiên LSTM sẽ quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào; nó lấy đầu vào là *ht−1* và x_t rồi đưa ra kết quả là một số trong khoảng [0,1] cho mỗi số trong trạng thái tế bào *Ct−1* ở phương trình (4); tiếp theo là quyết định xem thông tin mới nào sẽ lưu vào trạng thái tế bào bằng cách kết hợp phương trình (3) và (5). Cuối cùng giá trị đầu ra sẽ dựa vào trạng thái tế bào phương trình (6) và phương trình (7) nhân đầu ra với một cổng sigmoid để cho ra một giá trị đầu ra mong muốn.

3. Kết quả thử nghiệm

Trong phạm vi nghiên cứu, mô hình phân loại dựa trên mạng MLP và mạng RNN-LSTM được lựa chọn. Hai mô hình này được huấn luyện trên bộ dữ liệu KDD Cup 1999. Thuật toán được xây dựng trên ngôn ngữ Python và thư viện Keras, Sklearn, chạy trên nền tảng Tensorflow và môi trường Spyder của Anaconda.

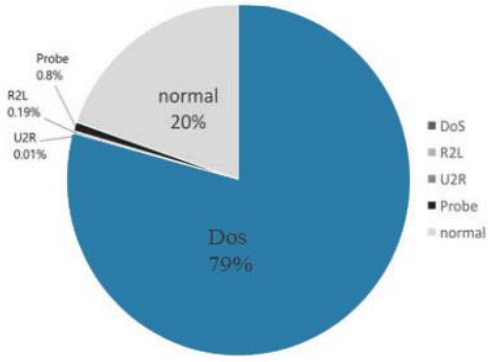
3.1 Tập dữ liệu KDD99

Tập dữ liệu KDD99 đã được sử dụng để đo lường hiệu suất của IDS trong rất nhiều nghiên cứu mặc dù tập dữ liệu này cũ nhưng nó có rất nhiều kết quả đo hiệu năng, rất thích hợp để so sánh với các mô hình khác. Tập dữ liệu này có tất cả 4.898.431 traffic mạng; mỗi traffic có 42 chiều, các chiều bao gồm các loại giao thức, dịch vụ và cờ:
'duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment','urgent','hot','num_failed_logins','logged_in,num_compromised','root_shell','su_attempted','num_root','num_file_creations','num_shells','num_access_files','num_outbound_cmds','is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_rate','rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rate','dst_host_count','dst_host_srv_count','dst_host_same_srv_rate','dst_host_diff_srv_rate','dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate','dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate','outcome'.
Có tất cả 23 kiểu tấn công được phân loại theo số chiều như trên; 23 loại tấn công này được phân làm 4 danh mục là DoS, R2L, U2R và Probe (Hình 6).

Loại	Tấn công
DoS	back, land, neptune, pod, smurf, teardrop
R2L	ftp-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster
U2R	buffer-overflow, loadmodule, perl, rootkit
Probe	ipsweep, nmap, portsweep, satan

Hình 6. Phân loại tấn công trong KDD99 [3]

Công đoạn đầu tiên phải thực hiện là xác định chia tập dữ liệu thành 2 thành phần, trong đó dùng 80% dữ liệu để huấn luyện và 20% dữ liệu dùng để kiểm tra. Cũng không cần sử dụng tập dữ liệu chưa được gán nhãn nào khác để kiểm tra khả năng phát hiện của mạng neural, sau quá trình huấn luyện 80% dữ liệu, mạng neural dùng chức năng fit model để cố gắng gán nhãn 20% dữ liệu còn lại sau đó cập nhật các trọng số để mạng neural đạt hiệu quả cao nhất (Hình 7). Đa số trong tập dữ liệu này là tấn công DoS và tập dữ liệu bình thường nên thuật toán sẽ được đào tạo một cách chênh lệch.



Hình 7 Tỉ lệ các cuộc tấn công trong tập dữ liệu

3.2 Các phương pháp đánh giá

Trước tiên sử dụng Accuracy (độ chính xác) để tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử. Tiếp theo lấy tỉ lệ phát hiện (DR – Detection Rate) và tỉ lệ phát hiện sai (FAR – False Alarm Rate) làm thước đo đánh giá độ nghiêm trọng của một lớp nào đó trong một hệ thống phân lớp. DR biểu thị tỉ lệ các trường hợp tấn công đã bị phát hiện bởi thuật toán và FAR là tỉ lệ các trường hợp thường đã bị phát hiện sai. Dựa trên một ma trận sai số, cách tính các chỉ số được thực hiện như sau:

DR = TP / (TP + FN)

FAR = FP / (TN + FP)

Trong đó (TP - True Positive) là số mẫu phân lớp đúng được chấp nhận , (TN – True Negative) là số mẫu phân lớp đúng bị từ chối, (FP – False Positive) là số mẫu phân lớp sai được chấp nhận, (FN – False Negative) là số mẫu phân lớp sai bị từ chối. Khi chỉ số DR tăng và chỉ số FAR giảm thì hiệu suất phát triển được đánh giá tốt hơn.

3.3 Cài đặt mô hình

Trước khi sử dụng tập dữ liệu huấn luyện cần phải chuẩn hóa tất cả các trường hợp từ 0 đến 1; đầu vào có tất cả 41 trường đặc điểm và đầu ra có 4 loại tấn công và 1 loại là dữ liệu bình thường. Do mạng neural yêu cầu đầu vào phải là các cột có giá trị số cố định, như một dữ liệu bảng tính phải có đầu vào hoàn toàn là số nên phải mã hóa các vector đặc điểm cho các loại dữ liệu khác nhau. Trong thư viện Tensorflow và Scikit-learn có một số đoạn mã dùng để mã hóa vector và tăng số chiều vector như sau:

- Encode_text_dummy: dùng để mã hóa các trường văn bản, giống như 4 loại tấn công là một trường duy nhất trong một lớp, 4 lớp có thể mã hóa thành “1, 0 , 0, 0”, “0, 1, 0, 0”, “0, 0, 1, 0”, “0, 0, 0, 1”. Đây là phương pháp mã hóa các trình dự đoán non-target.
- Encode_text_index: Giống như Encode_text_dummy, mã này mã hóa trường văn bản thành các số đại diện trong các lớp “0”, “1”, “2”, “3”; đây là phương pháp mã hóa các trình dự đoán có mục tiêu.
- Encode_numeric_zscore: Mã hoá các giá trị số dưới dạng z-score.

Ở thử nghiệm cài đặt mạng MLP với số lớp là 4, 1 lớp đầu vào có 10 node, lớp ẩn có 3 lớp và một lớp đầu ra.

Ở thử nghiệm cài đặt mạng RNN- LSTM vào các lớp ẩn với các tham số time-step, batch-size và epoch.

3.4 Kết quả

Kết quả được so sánh hiệu suất với các thuật toán máy học là SVM và bayesian, tỉ lệ Accuracy thể hiện hiệu quả của thuật toán trong quá trình huấn luyện.

Bảng 1 Kết quả so sánh hiệu suất giữa các thuật toán học

	DR	FAR	Accuracy
Bayesian	77,65%	17,57%	88,46%
SVM	87,65%	6,12%	92,6%
MLP	96,33%	3,34%	98,22%
LSTM-RNN	98,8%	10,05%	99,04%

4. Kết luận

Từ kết quả thực nghiệm cho thấy mô hình MLP và RNN-LSTM có thể đáp ứng được yêu cầu phát hiện tấn công nêu ra ở trên. Tuy hiệu suất khá trực quan nhưng không thể hiện hết được sức mạnh của thuật toán do được xây dựng để nghiên cứu chứ không phải vì mục đích thương mại, không đáp ứng đủ yêu cầu về phần cứng, thời gian huấn luyện cũng như kích thước tập huấn luyện. Bên cạnh đó, ưu điểm của mô hình học sâu là có thể phát hiện các cuộc tấn công mạng nhanh hơn và cho tỉ lệ chính xác cao; đặc biệt với dữ liệu càng nhiều và thời gian huấn luyện càng lâu. Như vậy việc sử dụng mô hình học sâu vào việc phát hiện tấn công mạng là hoàn toàn phù hợp. Ngoài ra hướng tiếp cận mới là áp dụng mạng MLP với hàm kích hoạt là ReLU (so với các hàm Sigmoid và tanh) và mạng RNN-LSTM trên tập dữ liệu KDD99, áp dụng hàm đo độ mất mát là RMSE với hiệu quả đánh giá trên tổng bình phương của toàn bộ độ mất mát trên các lớp ẩn.

Từ các kết quả trên cũng đặt ra các vấn đề nghiên cứu còn bỏ ngõ như sau:

- Cần nghiên cứu các mô hình với bộ tham số và các hàm số khác nhau để tìm ra các bộ số thích hợp làm tăng hiệu suất của thuật toán.
- Năng lực xử lý dữ liệu cũng như tính toán của hệ thống máy đóng vai trò quan trọng trong việc khai thác thuật toán; đặc biệt là các thuật toán học sâu yêu cầu lượng lớn bộ nhớ để huấn luyện.
- Ứng dụng thuật toán vào các tập dữ liệu tấn công khác bao gồm tập dữ liệu có gán nhãn và không có gán nhãn.

Tài liệu tham khảo

1. Al-Sakib Khan Pathan, *The State of the Art in Intrusion Prevention and Detection*, Taylor & Francis Group, LLC, NewYork, 2014, p. 117-139.
2. Peter Scherer, Martin Vicher, Jan Martinovic, Using SVM and Clustering Algorithms in IDS Systems, *Proceedings of the Dateso 2011: Annual International Workshop on DAtabases, TExts, Specifications and Objects*, Pisek, Czech Republic, (2011)
3. Hoàng Ngọc Thanh, Trần Văn Lăng, Hoàng Tùng, Một tiếp cận máy học để phân lớp các kiểu tấn công trong hệ thống phát hiện xâm nhập, *Kỷ yếu Hội nghị Quốc gia lần thứ IX về Nghiên cứu cơ bản và Ứng dụng Công nghệ thông tin*, – FAIR'2016, Cần Thơ, 04-05/8/2016, ISBN: 978-604-913-472-2, NXB. Khoa học tự nhiên và Công nghệ, DOI: 10.15625/vap.2016.00061, (2016) 502-508.
4. Guo-BingZhou, JianxinWu, Chen-LinZhang, Zhi-HuaZhou, Minimal Gated Unit for Recurrent Neural Networks, *International Journal of Automation and Computing*, June 2016, DOI: 10.1007/s11633-016-1006-2, (**13)**3 226–234,
5. Olalekan Ogunmolu, Xuejun Gu, Steve Jiang, Nicholas Gans, Nonlinear Systems Identification Using Deep Dynamic Neural Networks, *American Control Conference (ACC)*. Seattle, WA, 2017.
6. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, *Communications of the ACM*, ISSN: 0001-0782, EISSN: 1557-7317, DOI: 10.1145/3065386 , (**60**) 6, (2017), 84-90.
7. Bengio, Yoshua, S. Patrice, F.Paolo, Learning long-term dependencies with gradient descent is difficult, *Neural Networks, IEEE Transactions on Neural Networks*, DOI: 10.1109/72.279181, (5)2 (1994) 157-166
8. Hochreiter, Sepp, Jrgen Schmidhuber, Long short-term memory, *Neural computation*, (**9**)8, (1997) 1735-1780..

Improvement detection abbility of network attacks by deep learning

To Trong Tin[1], Tran Van Lang[2, 3]

[1]Posts and Telecommunications Institute of Technology, [2]Institute of Applied Mechanics and Informatics, VAST
[3]Nguyen Tat Thanh University

Abstract The Intrusion Detection System (IDS) is a security software designed to alert automatically when someone or something is trying to infiltrate the system, but this invasion may cause the system to be in danger or violate the privacy policy. Many studies have successfully applied machine learning algorithms to IDS systems that have the ability to self-study and update new attacks. But to limit false alarms and increase the likelihood of predicting attacks, the IDS should have more analytical thinking. This is deep learning. This paper addresses the deep learning as a new approach that can help the IDS system improve accuracy and speed up analysis when input data is too large. With the application of deep neural networks such as the Multilayer Perceptron (MLP) and the Recurrent Neural Network (RNN) on the KDD99 dataset to evaluate Accuracy, Mean Squared Error and Confusion Matrix. The efficiency gains were 98.2% for MLP and 99.04% for RNNs, compared to 92.6% for SVM and 88.46% for Naïve Bayes.

Keywords IDS, computer network, neural network, deep learning, machine learning