Deep Learning Techniques to Detect Botnet

Doan Trung Son, Nguyen Thi Khanh Tram, Pham Minh Hieu

Abstract— Over the past time, the world has witnessed an unprecedented explosion of Deep Learning. Besides the development of Information Technology, security and safety threats are also increasing, one of which is the Botnet network. Botnet network is increasingly complex and difficult to detect, and traditional techniques are no longer effective, so one of the urgent problems today is to find an effective solution to detecting botnets [2]. Based on the characteristics of deep learning such as scalability, performance, execution time, interpretability, etc., therefore, in this paper, the author proposes to use deep learning techniques to detect Botnet networks.

Keywords—botnet; deep learning; BotShark-SA; BoTShark-CNN.

I. INTRODUCTION

A BotNet is the shorthand term for "Bots Network". Just a network of infected computers (Bots/Zombies) dominated by another computer. The larger the Botnet network, the higher the danger. Botnets are actually a group of compromised Internet devices that are controlled remotely by cybercriminals. Cybercriminals use Botnets to launch coordinated attacks and perform other malicious activities. The word “botnet” is a combination of two words, “robot” and “network”. Here, a cybercriminal performs the role of a “botmaster” using a Trojan virus to compromise the security of several computers and connect them to the network for malicious purposes. Each computer on the network acts as a "bot" and is controlled by the bad guys to spread malware, spam, or malicious content to launch the attack. The number, scale, level of danger, and especially the hidden ability of Botnet networks is increasingly sophisticated and complex. In Vietnam, according to information from the Vietnam Computer Emergency Response Center (VNCERT), in the first two quarters of 2019, nearly 100,000 Vietnamese network (IP) addresses were queuing and connecting to Internet sites every day. computer network (Botnet) and up to 6,219 incidents of cyberattacks on Vietnamese websites.

BotHunter is one of the earliest behavior- based Botnet detection systems, it works based on the use of SNORT software to generate alerts about the behavior of each individual machine. However, the fact that it works by observing packet payloads makes this system not very effective in detecting Botnet networks that have encrypted connections. So BotMiner appeared, BotMiner works by grouping the behavior of different machines in the same Botnet.

On the other hand, in this early time, there was some research on how to detect malicious network flows by Botnet based on machine learning, but these network flows are only from internal machines in the LAN. After that, machine learning became a popular technique in Botnet detection. The two-stage system includes a feature extraction phase and a machine learning phase. This system is used to detect Botnet networks based on IRC (Internet Relay Chat - a protocol designed for real-time chat communication based on Client-Server architecture), it works by using the Bayesian classification method to detect network traffic from Botnet’s C&C (Command-and-Control) machine, this system results in 90% detection rate and 15.4%
Several other Botnet detections work based on DNS queries, and it has achieved a malicious DNS query detection rate of up to 92.5% [3]. However, detection based on DNS queries makes the systems workable only with Botnets that use the DNS system to search their C&C servers (mostly these networks are Botnets). The system for detecting P2P Botnet network flows works by assuming that the network traffic generated by users will fluctuate wildly unlike the network traffic of P2P Botnet. This system achieves detection rates up to 98%, but false detection rates are high (30%) [4].

The author has evaluated 3 methods of Botnet detection based on deep learning: BoTShark-SA (Botnet Traffic Shark - Stacked Autoencoders); BoTShark-CNN (Botnet Traffic Shark - Convolutional Neural Networks) and Botnet detection method based on anomalous data flow. The results show that BoTShark-SA and BoTShark-CNN both achieve the ratio of correctly predicted positivity to true positive rate TPR (True Positive Rate) greater than or equal to 0.91 on the ISCX dataset. BoTShark works on all basic properties and none of the properties are filtered by experts. BoTShark-SA achieved a TPR of 0.91 and a False Positive Rate (FPR) of 0.15, while BoTShark-CNN achieved TPR higher than 0.92. The comparison shows that the proposed deep learning-based approach achieves higher Botnet detection rates (false detection rate is still high) [4].

Botnets are commonly used, frequently in Distributed Denial of Service (DDoS) attacks. An attacker can control a large number of hijacked computers at a remote station, exploit their bandwidth and send connection requests to the target machine. Many networks suffer terrible consequences after suffering attacks of this type [5].
The results of the evaluation of the three methods above show that these methods are superior to other related methods, and it can detect Botnets with an accuracy of up to 99%.

Along with that, the model training time is acceptable when applied to a fairly large set of Botnets, the detection results are very accurate and have great potential for practical application. Besides, the Botnet detection model built based on the Artificial Neural Network (ANN) with the output of the Softmax output activation function is also the most applied and most effective model.

**B. DEEP LEARNING TECHNIQUE TO DETECT BOTNET**

In this paper, the authors build a classification model based on deep learning, in which: The model is built based on the activation function which is the Softmax function, the model is trained and tested with train and test sets. 10 best features have been extracted previously. The model is built on top of Keras with TensorFlow support. The application used is the Google Collaboration application by Google.

**Introduction of dataset:** The data set used by the author is a Bot-IoT built by Nickolaos Koroniatis, Nour Moustafaa, Elena Sitnikova, Benjamin Turnbull [6] from the University of New South Wales Canberra, Austria. The dataset is a combination of often simulated IoT network flows, along with diverse attack methods. The dataset is huge with 72000000 records, with 16.7 GB for CSV files and 69.3 GB for Pcap files.

The Bot-IoT dataset is a data set that has been built and analyzed thoroughly and completely, through which machine learning and deep learning methods can be easily applied. This dataset has shown its advantages when compared with other datasets as follows:

**TABLE I. Bot-IoT DATASET AND OTHER DATASET**

<table>
<thead>
<tr>
<th>Data set</th>
<th>Experiment</th>
<th>Actual traffic</th>
<th>Labelled data</th>
<th>IoT device traces</th>
<th>Diverse attack scenarios</th>
<th>Captured full packets</th>
<th>Extra new properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARPA98</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>KDD99</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

The dataset is built on three components, namely: Networking platform, simulation of IoT devices, and finally feature extraction and investigation, and analysis. First, the networking platform consists of normal and attack virtual machines (VMs). Second, about simulating IoT devices, IoT devices will be simulated through the Node-red tool. Finally, it is about extracting and investigating and analyzing features. Here, the Argus tool is used to extract features about the data so that machine learning techniques can then be applied.

**Data preprocessing:** First, it is necessary to import the dataset. First, we initialize the path to the dataset, and then use Python's CSV-formatted data reading function.

Next, we will get the data features, here the feature columns used to identify the data stream saddr, daddr, proto, sport, dport will be removed. In addition, we also remove the label columns in the data such as an attack, category, and subcategory. These are the labels of the data streams so we keep them separate.

In addition, the label used to distinguish it is the category. We need to convert the labels of the category column to numeric values. It is similar to a dictionary of 5 values: \{0:A, 1:B, 2:C, 3:D, 4:E.\} where label A is replaced with 0, label B is replaced with 1. Then we need to convert the numbers corresponding to the labels into vectors. For example, if the first column label is 0, it will convert to a 5-dimensional vector of \[1,0,0,0,0\], the second column label of 4 will convert it to \[0,0,0,1,0\]. Because the explosive lattice model requires the input of the label to be a vector, we need to convert its label to a vector.
For the extracted dataset, we proceed to classify the columns in the dataset into alphanumeric columns.

```python
all_keys = X_train.keys()
num_keys = X_train.iloc[:, :-2].select_dtypes(exclude=['object']).keys()
cat_keys = X_train.iloc[:, :-2].select_dtypes(include=['object']).keys()
print(all_keys)
```

In which, the variable num_keys contains columns of numeric values and the variable cat_keys contains columns of literal values, the iloc command is used to browse each row in the column (keys) and the select_dtypes command is used to determine the data type to choose. For columns with numeric values, we need to proceed to adjust the data of the characteristics to a common scale with a small enough range of values, the purpose of which is to help classifiers work properly, most accurate and effective. And the technique applied here is MinMax Scaling.

\[
z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}
\]

The above formula will normalize the properties to values in the range [0, 1]. However, because the values in the data set are too large, MinMax Scaling cannot be applied. Therefore, we first need to use the base 10 logarithmic formula to reduce the data size.

Similarly, for columns with literal values, we will also use MinMax Scaling to return data in the range [0,1]. However, before that, we also need to have a lexicographic method to convert alphanumeric values.

Building neural network model: The neural network model is built based on the output activation function, which is the Softmax function. The model consists of 3 layers as follows:

```
Figure 2. Neural Network model
```

The input layer is a 14-valued attribute, followed by the first hidden layer with 128 nodes, then the second hidden layer with 64 nodes, and the last is an output label with 5 labels. Here, to increase model efficiency and reduce overfitting during training we use dropout technique.

After initializing the neural network, we proceed to train the model:

```python
history = model.fit(X_train, y_train, validation_split=0.2, shuffle=True, batch_size=256, epochs=10, callbacks=[stopping()])
```

Where epoch is the number of repetitions, and batch_size is the number of samples, these two values are calculated according to the given formula. After each iteration, we will drop out, edit the model to achieve the lowest loss and highest accuracy, and use the graph to display the training results based on the accuracy.

```
Figure 3. Show training results
```

III. RESULTS AND DISCUSSION

A. MODEL EVALUATION

To make the assessment simple and effective, the author has evaluated the model based on 4 common metrics: Precision, Recall, F1-Score and Accuracy, these measures have scores equal to the value average of each label. The evaluation will be done by running the algorithm 3 times and calculating the results, which are:

```
Figure 3. Show training results
```

```
```

Where epoch is the number of repetitions, and batch_size is the number of samples, these two values are calculated according to the given formula. After each iteration, we will drop out, edit the model to achieve the lowest loss and highest accuracy, and use the graph to display the training results based on the accuracy.
Thus, we can see that the model has built accurate predictions with the ratio at 4 very similar measurements, and through 4 measurements we draw conclusions: With Precision 0.9997, it shows that every 10000 fields positive cases (Botnet) are predicted, then 9997 cases are true positives; Recall of 0.9996 shows that out of every 10,000 true positives, the model predicts 9996 of them. The remaining two measures, F1 and Accuracy, are both very high (0.9998 and 0.9996), also showing the effectiveness of the model in distinguishing between Botnet and normal.

To make the assessment more intuitive, the author calculated the measured value for each label: Precision, Recall and F1-score, and gave the results for each part as follows:

**TABLE II. RESULTS FOR EACH PART**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Số mẫu</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDoS</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td>385309</td>
</tr>
<tr>
<td>DoS</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td>330112</td>
</tr>
<tr>
<td>Normal</td>
<td>0.85</td>
<td>0.58</td>
<td>0.69</td>
<td></td>
<td>107</td>
</tr>
<tr>
<td>Reconnaissance</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td>18163</td>
</tr>
<tr>
<td>Theft</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td>14</td>
</tr>
</tbody>
</table>

**TABLE III. RESULTS OF THE REMAINING MEASUREMENTS**

<table>
<thead>
<tr>
<th></th>
<th>TNR</th>
<th>NPV</th>
<th>TS</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDoS</td>
<td>0.999966</td>
<td>0.999871</td>
<td>0.999852</td>
<td>385309</td>
</tr>
<tr>
<td>DoS</td>
<td>0.999975</td>
<td>0.99997</td>
<td>0.999933</td>
<td>330112</td>
</tr>
<tr>
<td>Normal</td>
<td>0.999937</td>
<td>0.99999</td>
<td>0.535088</td>
<td>107</td>
</tr>
<tr>
<td>Reconnaissance</td>
<td>0.999943</td>
<td>0.999918</td>
<td>0.994512</td>
<td>18163</td>
</tr>
<tr>
<td>Theft</td>
<td>0.999981</td>
<td>1</td>
<td>0</td>
<td>14</td>
</tr>
</tbody>
</table>

**Figure 5. Modeling of partial results**

The parameters of table 3 are determined as follows: TNR (Selectivity - Ratio of correctly predicted negative values to true negatives), NPV (Rate of correctly predicted negative values to value), predictive negative) and TS (Threat Score - just like F1-score would be the composite of Precision (PPV) and Recall (TPR)).

**Figure 6. Modeling the remaining measures**

From the measures for each component, we can draw the comment that when the number of positive samples of that type is large, the prediction accuracy of negative and positive values for that type will be more accurate. When the number of positive samples is small, for example, the Normal-type, the prediction will no longer be accurate, but because
The difference between the number of positive and negative samples is too large, the measures of TNR and NPV are still give high results, similar to Theft's case. Therefore, the evaluation of the model's operability will be most accurate based on the following metrics: Precision, Recall, and F1-score.

B. COMPARE METHODS

In this paper, the author compares other machine learning methods including SVM (Support Vector Machine), RNN (Recurrent Neural Network), and LSTM (Long Short Term Memory) that have been introduced. In the paper "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset" by Nickolaos Koroniotis, Nour Moustafaa, Elena Sitnikova and Benjamin Turnbull [9] with DeepLearning method.

<table>
<thead>
<tr>
<th>TABLE IV. COMPARE METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>702</td>
</tr>
<tr>
<td>Precision</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Recall</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>Training Time</td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

Figure 7. Compare methods

Thus, when compared with other methods, the applied deep learning method gives better results, along with a shorter training time. And also from the table, we can see that 2 other methods using neural networks, RNN and LSTM, also give high results compared to the SVM method. However, the Precision parameter of the SVM is 1, which shows that the SVM method predicts positive cases very accurately, not mistakenly predicting negative to positive, but it has the disadvantage that it is easy to miss a lot. positive case.

Therefore, it can be concluded that SVM method will be better than deep learning methods in predicting small data sets or data sets with few positive cases. However, within the scope of Network Forensics data, it is clear that the optimization of deep learning methods will be more clearly demonstrated.

IV. CONCLUSION

The article has drawn the role of deep learning in Botnet detection and realized Botnet network detection is based on deep learning. The article has outlined two methods of Botnet detection based on Deep Learning, which are BoTShark-SA and BoTShark-CNN based on 2 techniques and Autoencoder and CNN. The obtained results show the effectiveness and superiority of the Botnet detection deep learning model. In the future, the author wishes to directly execute on the raw data set after intercepting network data, thereby reducing data processing time and enhancing the applicability of the model. Building a deep learning application to detect Botnets can contribute to the task of ensuring information security, and protecting state secrets in the People's Public Security force in the coming time.

ACKNOWLEDGMENT

The authors would like to thank VINIF for their financial support to Nguyen Thi Khanh Tram as master student in VNU University of Engineering and Technology.

REFERENCES


