Using Object Detection Network for Malware Detection and Identification in Network Traffic Packets

Chunlai Du[1], Shenghui Liu[1], Lei Si[2], Yanhui Guo[2, *] and Tong Jin[1]

Abstract: In recent years, the number of exposed vulnerabilities has grown rapidly and more and more attacks occurred to intrude on the target computers using these vulnerabilities such as different malware. Malware detection has attracted more attention and still faces severe challenges. As malware detection based traditional machine learning relies on exports' experience to design efficient features to distinguish different malware, it causes bottleneck on feature engineer and is also time-consuming to find efficient features. Due to its promising ability in automatically proposing and selecting significant features, deep learning has gradually become a research hotspot. In this paper, aiming to detect the malicious payload and identify their categories with high accuracy, we proposed a packet-based malicious payload detection and identification algorithm based on object detection deep learning network. A dataset of malicious payload on code execution vulnerability has been constructed under the Metasploit framework and used to evaluate the performance of the proposed malware detection and identification algorithm. The experimental results demonstrated that the proposed object detection network can efficiently find and identify malicious payloads with high accuracy.

Keywords: Intrusion detection, malicious payload, deep learning, object detection network.

1 Introduction

According to the analysis of the number of Common Vulnerabilities and Exposures (CVEs) released from 1999 to 2019, the number of exposed vulnerabilities increased rapidly from the gentle fluctuation of 4,000-6,000 in 2005-2016 to more than 12,000 in 2017-2019 as shown in Fig. 1 below [Vulnerabilities by Date (2019)]. The vulnerabilities of CVE are counted by type as shown in Fig. 2 below [Vulnerabilities by Type (2019)]. From them, the total number of vulnerabilities exceeds 70,000 including Executable Code type, Overflow type, Gain privilege type, Memory corruption type, and XSS attacks type. The number and type of vulnerabilities are so rich that attackers can easily launch an attack by selecting and applying the appropriate vulnerabilities. With the chosen vulnerabilities which are unfixed in target host, attacker builds and sends data packets carrying a malicious payload to trigger the corresponding vulnerabilities. Once the remote hosts receive and then process those data packets with unfixed software, shellcode

[1] School of Information Science and Technology, North China University of Technology, Beijing, 100144, China.

[2] Department of Computer Science, University of Illinois Springfield, Springfield, USA.

[*] Corresponding Author: Yanhui Guo. Email: yguo56@uis.edu.

Received: 10 February 2020; Accepted: 24 April 2020.

CMC. doi:10.32604/cmc.2020.010091

can be triggered and then control these remote hosts. With the idea that everything can be accessed in the internet of things coming true, whether the communication packets between nodes are malicious or not is especially important [Medhane, Sangaiah, Hossain et al. (2020); Tian, Gao, Su et al. (2020); Wang, Gao, Liu et al. (2019)].

At present, the network attack using the vulnerabilities of the operating system or application software becomes the main means. The confrontation between network attack and defense faces a severe situation because the traditional intrusion detection based on access feature and data packet statistics has been unable to work effectively.

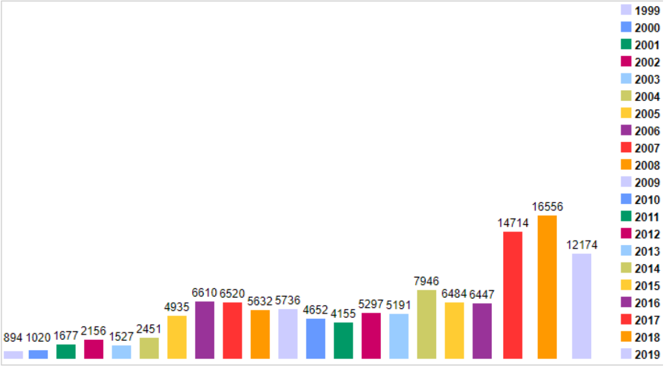


Figure 1: Vulnerabilities by year

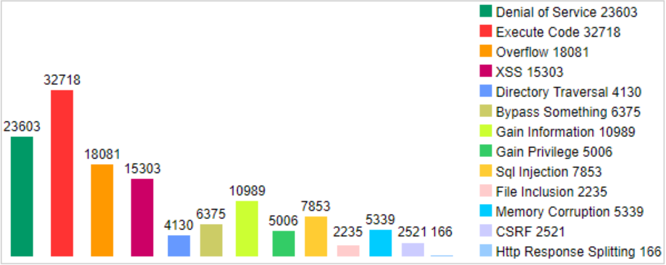


Figure 2: Vulnerabilities by type

In recent years, malicious payload detection using machine learning and deep learning has become a hot topic in the field of network intrusion detection [Karatas, Demir and Sahingoz (2018); Naseer, Saleem, Khalid et al. (2018); Tian, Shi, Wang et al. (2019); Yin, Luo, Zhu et al. (2020)]. A variety of features have been extracted ranging from grammatical features or session features to semantic features and then employed to distinguish the malicious payload using machine learning algorithms. The semantic features extracted in web script, such as SQL injection, XSS, web Trojan, has achieved good results. However, they are not efficient in non-web script and need further research.

In order to solve the bottleneck of feature extraction in malicious payload detection, this paper proposes a novel deep learning-based method to automatic extract feature and identify and classify the malicious payload in the network intrusion environment. The key contributions of our work are as follows:

(1) Build a malicious payload dataset of a non-web script attack under the Metasploit

framework.

(2) Propose a malicious payload detection model based on object detection deep learning network.

(3) Automatic extract features without expert interaction.

The rest of the paper is organized as follows. The related literature is summarized in Section 2. In Section 3, the proposed model is described for detecting malicious payload, Section 4 shows the experimental results and Section 5 presents the conclusions and future work.

2 Related work

Network intrusion detection is to detect the intrusion behavior of attackers via monitoring the network data packets and their associated attribute characteristics, such as frequency, connection source, packet length, etc. Intrusion detection is divided into signature-based intrusion detection and anomaly-based intrusion detection [Duhan and Khandnor (2016)].

The signature-based intrusion detection is a rule matching method, which pays more attention to the session and the characteristics of the packet, such as packet length, session entity's IP and session start time, etc. This method can only detect the network intrusion registered in the rules, and can not detect the unknown or unregistered network intrusion. In order to overcome this limitation, Guruprasad proposed an evolution-based automatic generation rule model for the snort system to improve the efficiency of detection [Guruprasad and D'Souza (2016)].

The anomaly-based intrusion detection does not need to be ruled by experts' experience, and it can automatically extract features to form detection rules through such methods as a neural network, deep learning, etc. At present, deep learning has become a research hotspot in intrusion detection. Unlike machine learning which needs expert's experience to design to extract features, deep learning can automatically extract features and select significant features [Karatas, Demir and Sahingoz (2018)]. Therefore, two famous deep learning models, CNN and RNN, have been used prominently in intrusion detection and get rid of the drawback of feature extraction relying on expert's experience. Niyaz implemented a sparse autoencoder and soft-max regression-based NIDS which use the Self-taught Learning (STL) [Niyaz, Sun, Javaid et al. (2016)]. Shone proposed a nonsymmetric deep auto-encoder (NDAE) with unsupervised feature learning. Its multiple asymmetric hidden layers can better adapt to multi-dimensional inputs [Shone, Ngoc, Phai et al. (2018)]. Using a newly defined loss function, Vinayakumar et al. proposed a highly scalable hybrid DNNS framework [Vinayakumar, Alazab, Soman et al. (2019)]. Ashfaq proposed a semi-supervised learning method based on fuzziness. The scheme trains a single hidden layer forward neural network (SLFN) to output a fuzzy vector and then uses a fuzzy vector to fully classify unlabeled samples [Ashfaq, Wang, Huang et al. (2017)]. Tang proposed a flow-based deep neural network model in an SDN environment. The model only selects 6 basic features from 41 features of the NSL-KDD dataset and the detection accuracy reaches 75.75%. However, the model does not make full use of the automatic feature extraction by deep learning and still has a lot of space for improvement on accuracy [Tang, Mhamdi, McLernon et al. (2016)]. Yan et al. proposed a stacked sparse autoencoder (SSAE) to realize the automatic extraction of high-dimensional features. The original classification features are imported into SSAE to realize the automatic learning of deeply sparse features, and then different basic classifiers are constructed by using low-dimensional sparse features to achieve the intrusion detection of high-dimensional features [Yan and Han (2018)]. Yu et al. proposed a detection model based on feature graph which filters the normal connections with grid partitions and records the patterns of various attacks with graph structure [Yu, Tian, Qiu et al. (2019)].

Deep learning has been used to extract the text's semantic features of SQL-injection, XSS and web Trojan horse. Liu splits the payloads into words and converts each word to a 20-dimensional vector via word embedding and used CNN and RNN for classification [Liu, Lang, Liu et al. (2019)]. Andresini et al. not only use the residual error of auto-encoders as feature engineering and combine deep unsupervised neural networks with supervised neural networks but also formulate a deep learning method that resorts to a feature augmentation to elicit the data distribution of unforeseen attacks [Andresini, Appice, Mauro et al. (2019)]. Swarnkar et al. proposed to consider the minimum and maximum frequency range of a specific NGram in the grouping of the training dataset. According to the number of deviations from the range, the intrusion is detected and the unknown intrusions are detected from the deviation of the range. This model only tests the detection effect of HTTP type [Swarnkar and Hubballi (2015)]. Based on distributed deep learning to analyze URLs, Tian proposed a web attack detection system that is deployed on edge devices of IoT [Tian, Luo, Qiu et al. (2020)].

In the stage of network intrusion, the attackers will send data packets with malicious payload to trigger the unfixed vulnerabilities of target hosts. The location and content of the same malicious payload in the data packet are different from the different target platforms and encoding methods used to bypass virus detection. Thus, we focus on the payload area of the data package and use a deep learning network to detect the malicious payload.

3 Proposed model

3.1 Motivation

KDDCUP'99 and NSL-KDD are used in most of the existing intrusion detection research. Recently, with the development of network attacks, there are some new test datasets, such as cidds-001, cic-ids2017 and cse-cic-ids2018. KDDCUP'99 and NSL-KDD only contain DOS, probe, r2l, u2r and normal features [Tavallaee, Bagheri, Lu et al. (2009)]. CIDDS-001 includes characteristics of the session such as communication, data capture time, number of bytes, number of data packets in session [Abdulhammed, Faezipour, Abuzneid et al. (2019)]. CIC-IDS2017 and CSE-CIC-IDS2018 include DoS attack, Web attack, BOT, port scan, brute force attack and other features [Karatas, Demir and Sahingoz (2018)]. However, these datasets include manually designed features from the original data.

In this study, our detection aim is focused on the exploitable attack scenario in which the attacker sends the malicious payload of binary executable code by using the vulnerability of the target system or software. Therefore, in order to better describe the detection scenario, we do not use the previous public datasets and establish our own dataset to build the proposed method and test its effectiveness in the real network traffic environment.

3.2 Malicious payload detection model

Our model consists of building a dataset of malicious payload, preprocessing, and deep learning framework for automatic feature learning and extraction, classification shown in Fig. 3 below.

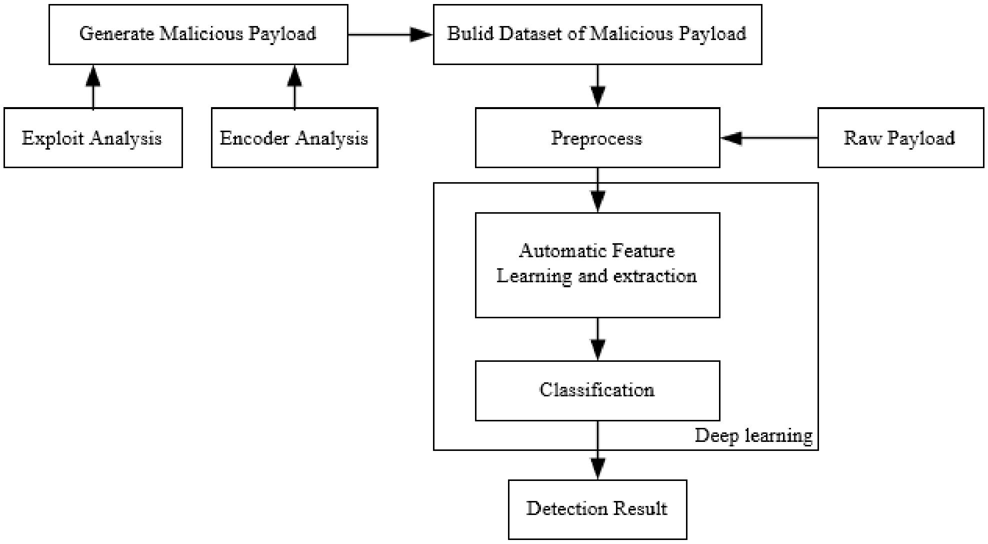

Generate Malicious Payload
Bulid Dataset of Malicious Payload
Exploit Analysis
Encoder Analysis
Preprocess
Raw Payload
Automatic Feature Learning and extraction
Classification
Deep learning
Detection Result

Figure 3: Proposed model

To build a dataset of malicious payload, we analyze the source code of the Metasploit framework [Metasploit-framework (2019)] and focus on the procedure to generate and send the malicious payload. In the preprocessing stage, we capture valid data packets carrying a malicious payload and transfer the data packet into a two-dimensional image. Then, the images with malicious payload are used to train a deep learning network, named YOLO [Redmon, Divvala, Girshick et al. (2016)]. In the detection phase of malicious payload, raw payloads are preprocessed as the previous step and then are fed as inputs of trained YOLO network. Final, the classification results by YOLO is used as the detection results and payload categories.

3.3 Dataset

Metasploit is a well-known attack framework, which contains rich exploit modules corresponding to different vulnerabilities and exploitable malicious payload modules. We establish a pair ***<exploit, payload>*** to name the malware, and packets with different malicious payloads are used as different samples in the dataset for training and its corresponding pair name as their categories.

Through making the real attack on the target hosts, a series of attack packets are captured and then the key malicious payload packets are located. The format of the content contained in the payload of packets in each attack is not uniform, and the length of the payload is not fixed. After observation of the lengths of payloads, we extract a total of three data packets including before and after the malicious payload of binary attack as effective data that can cover most of the payloads.

The procedure is described in algorithm 1 to build the payload samples as follows:

Algorithm 1: Build malicious payload samples

(1) Analyze the attacking process in Metasploit, track the generation of payload, and store the payload before sending;

(2) Locate the corresponding payload in the packets, take three consecutive data packets among the payload, and calibrate and record the position of the payload in the packets;

(3) Arrange three consecutive packets into a data frame with a fixed length and fill the insufficient parts using a constant value.

3.4 Deep learning for malicious payload detection and classification

Deep learning [LeCun, Bengio and Hinton (2015)] provided a distinguished ability to automatically extract multiple levels of abstraction and features from the raw data. One of the efficient deep learning architectures is the convolutional neural network (CNN) that has been commonly used for image classification. Currently, CNN based deep learning networks have achieved success in object detection and image classification, and many studies and projects have been conducted to identify and classify different kinds of images.

YOLO is a well-known object detection algorithm based on CNN, which divides the image into different regions and predicts the detected objects' bounding boxes and probabilities for each region [Redmon, Divvala, Girshick et al. (2016)]. It employs a single network to the full image and has a fast speed in object detection and identification.

In the proposed payload detection and identification method, the packages are captured in the network traffic, and three continuous packages are ensemble into as an image as shown in Algorithm 1. Then, the malicious payload detection problem is transferred into a task of object detection and classification. A deep convolutional network is designed, and malicious payload samples are used to train the network and then the payloads are identified via the detection results.

Rather than building a model from scratch for each application, we improved a pre-build deep learning model to identify new image classes. Known as transfer learning, it reduces the time in training and improves the network's generalization ability. The proposed model uses a pre-build YOLO v3 network as a backbone to extract the features and the loss function is redefined according to different malicious payload categories. The detailed information about the network structures of the YOLO v3 model can be viewed in detail [Redmon, Divvala, Girshick et al. (2016)].

Fig. 4 below shows an example of a package image with a malicious payload, and Fig. 5 below demonstrates the detection and classification results using the trained YOLO network.

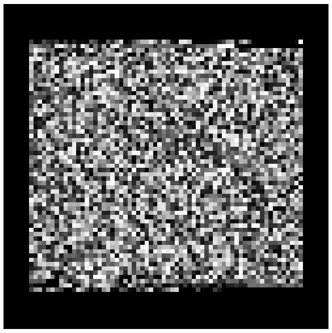


Figure 4: Sample of a package image

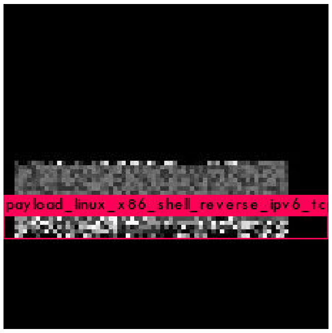


Figure 5: Prediction result on the package image

4 Experiment and discussion

To validate the efficiency of the proposed approach, we selected 246 unique <*exploit, payload*> pairs from our malicious payload dataset. Those pairs are collected from two common operating systems, and they are composed of 23 different payloads. Same payload with different exploits is considered as different samples in the same category. The number of different exploits for each payload is shown in Figs. 6 and 7 below. It is worth to noticed that, in the Fig. 6 below, the payload, payload-windows-reverse_ord_tcp, has the most available exploits. It means that the payload is very popular for exploits targeting windows platform. All the pair images are normalized at the size of 228×228 pixels, and an augment method is used to increase the number of images in the training stage of deep learning network. Fig. 8 below shows examples of the augment results for one image.

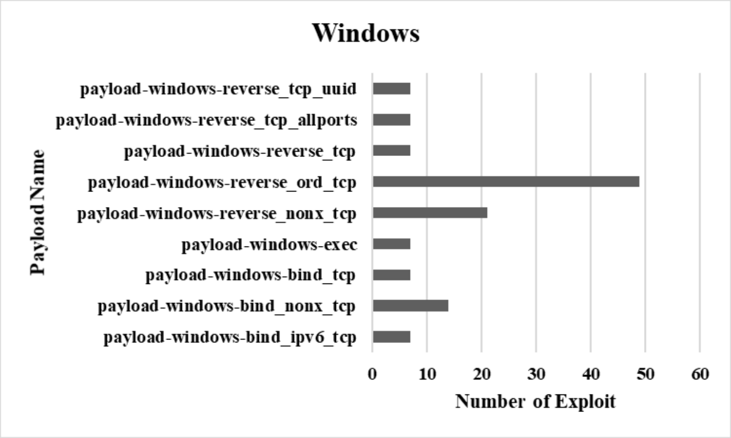


Figure 6: Number of Exploit of windows platform with the same valid payload

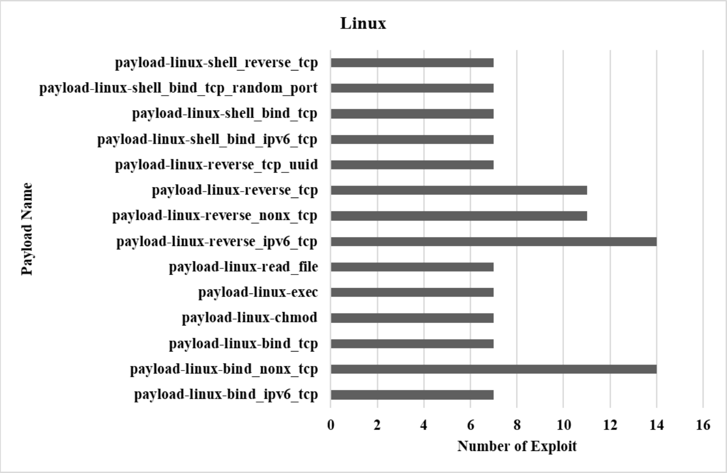


Figure 7: Number of Exploit of Linux platform with the same valid payload

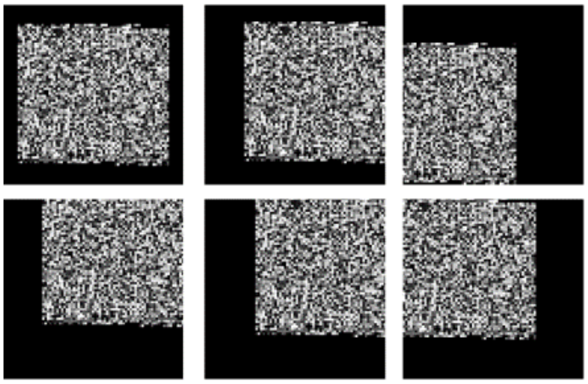


Figure 8: Augment results of a packet image

In the training stage, 12597 images were selected randomly from the whole data set after image augmentation. Each category has the same number of packet images. Then they are split with 90% of the images for training, 10% for testing. Therefore, 11328 were selected from 12597 categories balanced as the training set, 1269 for the test set, respectively.

The performance of the YOLO v3 on the training process is shown in Fig. 9 below, including the changing on accuracy and loss values during training. The model after training was employed in the testing set and the Fig. 10 below shows the confusion matrix of the classification results. Some typical examples of result images are shown in Fig. 11 below. The result demonstrates that the proposed method has great detection capability and high accuracy.

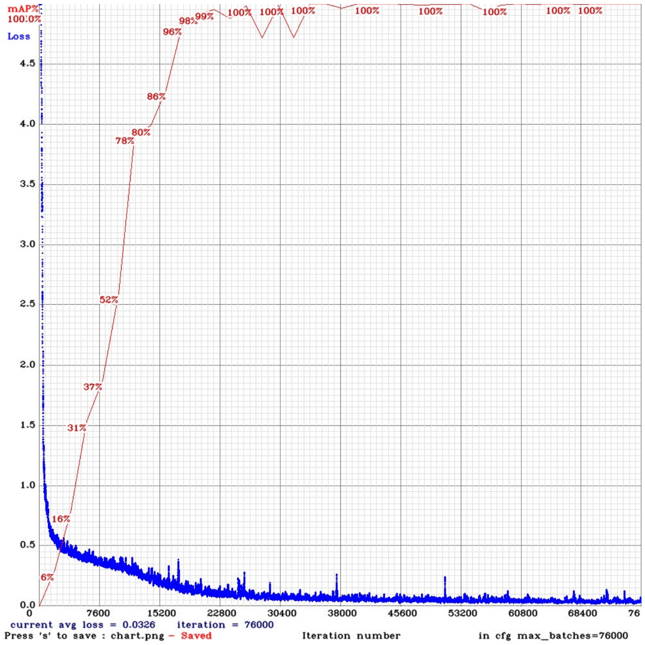
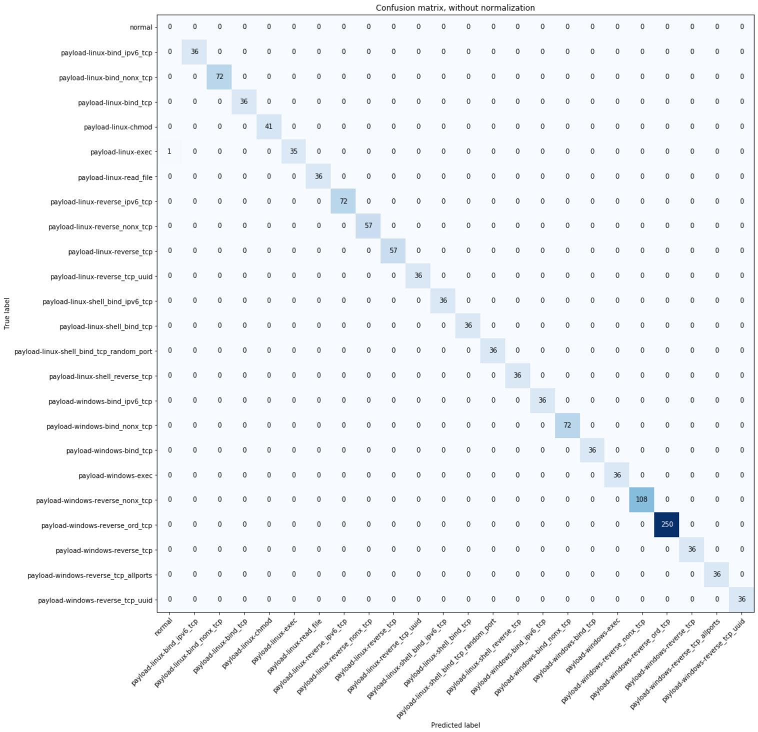


Figure 9: Accuracy and loss values in the training process



	normal	payload-linux-bind_ipv6_tcp	payload-linux-bind_nonx_tcp	payload-linux-bind_tcp	payload-linux-chmod	payload-linux-exec	payload-linux-read_file	payload-linux-reverse_ipv6_tcp	payload-linux-reverse_nonx_tcp	payload-linux-reverse_tcp	payload-linux-reverse_tcp_uuid	payload-linux-shell_bind_ipv6_tcp	payload-linux-shell_bind_tcp	payload-linux-shell_bind_tcp_random_port	payload-linux-shell_reverse_tcp	payload-windows-bind_ipv6_tcp	payload-windows-bind_nonx_tcp	payload-windows-bind_tcp	payload-windows-exec	payload-windows-reverse_nonx_tcp	payload-windows-reverse_ord_tcp	payload-windows-reverse_tcp	payload-windows-reverse_tcp_allports	payload-windows-reverse_tcp_uuid
normal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-bind_ipv6_tcp	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-bind_nonx_tcp	0	0	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-bind_tcp	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-chmod	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-exec	1	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-read_file	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-reverse_ipv6_tcp	0	0	0	0	0	0	0	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-reverse_nonx_tcp	0	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-reverse_tcp	0	0	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-reverse_tcp_uuid	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-shell_bind_ipv6_tcp	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0
payload-linux-shell_bind_tcp	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0
payload-linux-shell_bind_tcp_random_port	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0
payload-linux-shell_reverse_tcp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0
payload-windows-bind_ipv6_tcp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0
payload-windows-bind_nonx_tcp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	72	0	0	0	0	0	0	0
payload-windows-bind_tcp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0
payload-windows-exec	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0
payload-windows-reverse_nonx_tcp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	108	0	0	0	0
payload-windows-reverse_ord_tcp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	250	0	0	0
payload-windows-reverse_tcp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0
payload-windows-reverse_tcp_allports	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0
payload-windows-reverse_tcp_uuid	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36

Figure 10: Confusion matrix of the classification results

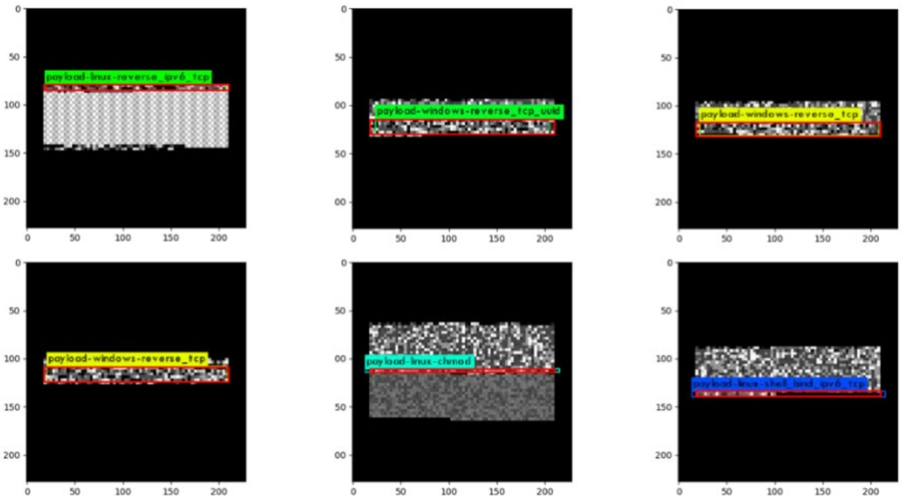


Figure 11: Six typical result images and the red rectangle is ground true label

The results on the accuracy, mAP and detection speed are shown in Tab. 1. The results demonstrate that the proposed method has both high accuracy and fast detection speed, and can be used for malware detection and classification in a real environment.

Table 1: Statistics results for proposed method

Threshold	Precision	mAP	Detection time
50%	0.99	99.45%	19.8 ms

5 Future work and conclusion

This study developed a deep learning network for malware detection and classification in network traffic packets. A YOLO v3 model was modified and tuned for malware detection and classification in packet images. The experimental results demonstrate that 99% accuracy was obtained on the current data set. In the future, a large data set is planned to be collected with more classes and more variation on the environment of packet acquisition. Due to that the conditions for acquired packets are close to the real applications, a similar performance could be achieved in practice after it is tuned using a more diversified data.

Funding Statement: This work was supported by Natural Science Foundation of China (61702013, 61572492), the National Key research and Development Plan (Grant No. 2018YFB0803504), Joint of Beijing Natural Science Foundation and Education Commission (KZ201810009011), Science and Technology Innovation Project of North China University of Technology (19XN108).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Abdulhammed, R.; Faezipour, M.; Abuzneid, A.; AbuMallouh, A. (2019): Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. *IEEE Sensors Letters*, vol. 3, no. 1, pp. 1-4.

Andresini, G.; Appice, A.; Mauro, N. D.; Loglisci, C.; Malerba, D. (2019): Exploiting the auto-encoder residual error for intrusion detection. *Proceedings of IEEE European Symposium on Security and Privacy Workshops*, pp. 281-290.

Ashfaq, R. A. R.; Wang, X.; Huang, J. Z.; Abbas, H.; He, Y. (2017): Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences*, vol. 378, pp. 484-497.

Duhan, S.; Khandnor, P. (2016): Intrusion detection system in wireless sensor networks: a comprehensive review. *Proceedings of International Conference on Electrical, Electronics, and Optimization Techniques*, pp. 2707-2713.

Guruprasad, S.; D'Souza, R. (2016): Development of an evolutionary framework for autonomous rule creation for intrusion detection. *Proceedings of IEEE 6th International Conference on Advanced Computing*, pp. 534-538.

Karatas, G.; Demir, O.; Sahingoz, O. K. (2018): Deep learning in intrusion detection systems. *Proceedings of International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism*, pp. 166-169.

LeCun, Y.; Bengio, Y.; Hinton, G. (2015): Deep learning. *Nature*, vol. 521, no. 7553, pp. 436-444.

Liu, H.; Lang, B.; Liu, M.; Yan, H. (2019): CNN and RNN based payload classification methods for attack detection. *Knowledge-Based Systems*, vol. 163, pp. 332-341.

Medhane, D. V.; Sangaiah, A. K.; Hossain, M. S.; Muhammad, G.; Wang, J. (2020): Blockchain-enabled distributed security framework for next generation IoT: an edge-cloud and software defined network integrated approach. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2020.2977196.

Metasploit-framework (2019): https://github.com/rapid7/metasploit-framework.

Naseer, S.; Saleem, Y.; Khalid, S.; Bashir, M. K.; Han, J. et al. (2018): Enhanced network anomaly detection based on deep neural networks. *IEEE Access*, vol. 6, pp. 48231-48246.

Niyaz, Q.; Sun, W.; Javaid, A. Y.; Alam, M. (2016): A deep learning approach for network intrusion detection system. *Proceedings of International Conference on Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, pp. 21-26.

Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. (2016): You only look once: unified, real-time object detection. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788.

Shone, N.; Ngoc, T. N.; Phai, V. D.; Shi, Q. (2018): A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41-50.

Swarnkar, M.; Hubballi, N. (2015): Rangegram: A novel payload-based anomaly detection technique against web traffic. *Proceedings of IEEE International Conference on Advanced Networks and Telecommunications Systems*, pp. 1-6.

Tang, T. A.; Mhamdi, L.; McLernon, D.; Zaidi, S. A. R.; Ghogho, M. (2016): Deep learning approach for network intrusion detection in software defined networking. *Proceedings of International Conference on Wireless Networks and Mobile Communications*, pp. 1-6.

Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A. A. (2009): A detailed analysis of the KDD cup 99 data set. *Proceedings of IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1-6.

Tian, Z.; Gao, X.; Su, S.; Qiu, J. (2020): Vcash: A novel reputation framework for identifying denial of traffic service in internet of connected vehicles. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2019.2951620.

Tian, Z.; Luo, C.; Qiu, J.; Du, X.; Guizani, M. (2020): A distributed deep learning system for web attack detection on edge devices. *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963-1971.

Tian, Z.; Shi, W.; Wang, Y.; Zhu, C.; Du, X. et al. (2019): Real-time lateral movement detection based on evidence reasoning network for edge computing environment. *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4285-4294.

Vinayakumar, R.; Alazab, M.; Soman, K. P.; Poornachandran, P.; Al-Nemrat, A. et al. (2019): Deep learning approach for intelligent intrusion detection system. *IEEE Access*, vol. 7, pp. 41525-41550.

Vulnerabilities by Date (2019): https://www.cvedetails.com/browse-by-date.php.

Vulnerabilities by Type (2019): https://www.cvedetails.com/vulnerabilities-by-types.php.

Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A. K.; Kim, H. J. (2019): An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *International Journal of Distributed Sensor Networks*, vol. 15, no. 3, pp. 1-9.

Yan, B.; Han, G. (2018): Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. *IEEE Access*, vol. 6, pp. 41238-41248.

Yin, L.; Luo, X.; Zhu, C.; Wang, L.; Xu, Z. et al. (2020): Connspoiler: disrupting C & C communication of IoT-based botnet through fast detection of anomalous domain queries. *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1373-1384.

Yu, X.; Tian, Z., Qiu, J.; Su, S.; Yan, X. (2019): An intrusion detection algorithm based on feature graph. *Computers, Materials & Continua*, vol. 61, no. 1, pp. 255-273.