Sensors and Materials, Vol. 30, No. 4 (2018) 857–867
MYU Tokyo

S & M 1544

Detection and Analysis of Distributed Denial-of-service in Internet of Things—Employing Artificial Neural Network and Apache Spark Platform

Ting-Yuan Chang* and Chang-Jung Hsieh

Department of Information Management, National Chin-Yi University of Technology,
No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung City, Taiwan (R.O.C.)

(Received October 29, 2017; accepted January 18, 2018)

Keywords: DDoS attack, IoT, sensor, artificial neural network, Apache Spark

In the development of Internet of Things (IoT), network security has received increasing attention. Many network attacks are performed through the sensors. Among the various cyberattacks, distributed denial-of-service (DDoS) attacks represent one of the most serious types. DDoS attacks should not be underestimated in terms of the loss they may cause. In this study, we integrated Apache Spark, a big-data computing framework, with a detection model based on a back-propagation artificial neural network. Thanks to the capacity of a big-data computing framework for mass historical data, computation of the characteristics required for the learning of the detection model can be performed in a real-time manner. An artificial neural network model is perfect for DDoS detection, owing to its good scalability and its advantage of restricting the expansion of computing resources consumed as data volume increases. This eliminates the problems related to traditional approaches to DDoS signature computing, where mass data can take considerable time to compute and can even overwhelm a system. The results of this study show that the trained artificial neural network achieved a detection rate as high as 99.80% and the real-time detection system achieved a detection rate of 87.18%. Compared with other studies in this field, it is clear that the proposed approach provides effective detection of DDoS attacks, and that the incorporation of Apache Spark, the open-source clustering big-data computing framework, allows more computers and more Kafka Producers receiving packets to be used to form an even larger detection system capable of dealing with increasingly huge and diverse DDoS attacks.

1. Introduction

The rapid development of information technology has led to people's significantly increased use of the Internet, and as a result, a number of cyber security issues have emerged. New IoT items are not just a target in themselves for attackers. A more dangerous option exists in such devices. The growing popularity of wireless sensor networks increases the risk of security attacks. One of the most common and dangerous types of attack that takes place these days

*Corresponding author: e-mail: c07408@ncut.edu.tw
http://dx.doi.org/10.18494/SAM.2018.1789

ISSN 0914-4935
http://myukk.org/

in any electronic society is a distributed denial-of-service (DDoS) attack. Due to the resource constraint nature of mobile sensors, DDoS attacks have become a major threat to its stability.

DDoS attacks, a cyberattack technique,[1] involve occupying the resources of a targeted host by a large number of accessing operations, and they can slow down services if not disrupt them. Currently, there are no effective defenses that provide 100% detection and blockage of DDoS attacks. These types of attacks are usually conducted through zombie networks consisting of many virus-infected zombie computers, and use IP spoofing to disguise the IPs from which the attacks are launched, making it difficult to track the sources of these attacks. At present, without an effective mechanism for identifying where attacks originate, organizations using an intrusion detection system (IDS), an intrusion prevention system (IPS), or firewall as a barrier can often erroneously deny users who are not attackers. Some other organizations may decide to increase their hosts' performance and their networks' bandwidth as a passive alternative that mitigates the impact of a possible attack. However, it is a reality that upgrading hardware and networks can never keep up with an attacks' growing power, inflicting organizations with great losses.[2]

Currently, to mitigate the impact caused by denial-of-service (DoS) attacks and DDoS attacks, these attacks are mainly dealt with by source tracking and intercepting.[2] However, since DDoS attacks may come from any corner of the world and are launched from tens of thousands of zombie computers, identifying and intercepting them one by one is an extremely time- and resource-consuming task. To address this issue pragmatically, it is crucial to endow organizations with a real ability to discern normal flow from abnormal flow.

A competent anomaly-based detection system can anticipate unknown attacks, if it is provided with sufficient training data. The conventional methods of anomaly-based detection aimed at DDoS attacks, however, either involve complicated algorithms or are inadequate for learning from mass historical data because a single machine is unable to finish the required computation in a reasonable timeframe or to provide sufficient computing resources. Therefore, in this study we tried to integrate detection with a big-data computing framework, Spark Apache. The proposed platform performs in-memory computing so as to prevent computational bottlenecks by significantly reducing hard-drive accessing operations for iterative computation, with the hopes of addressing the above-mentioned problems. This uses the advantages of the big-data computing framework and increases practicability by incorporating additional relevant elements.

In this study we integrate Apache Spark, a big-data computing framework, with an artificial neural network for attack detection. This approach is different from traditional detection because it has improved efficiency in integrating and analyzing historical data, thanks to its big-data computing ability, and incorporates an artificial neural network for learning, so that the resulting detection system is able to identify attack-associated flows. Finally, the Center for Applied Internet Data Analysis (CAIDA) DDoS Attack 2007 Dataset was used to simulate attack-associated network flows, and the 1999 Defense Advanced Research Projects Agency (DARPA) Intrusion Detection Evaluation Data Set was used to simulate normal network flows, in order to evaluate the approach of this study in experimental settings.

2. Related Works

2.1 DDoS attack

DoS attacks, a type of cyberattack, attempt to use a large number of requests to occupy a network and a host's resources, thereby slowing down or disrupting its services. These attacks are often launched from zombie computers that have been affected by viruses and are uncontrolled, making it difficult for the attacked parties to track or intercept them. DoS attacks conducted through a zombie network consisting of two or more zombie computers are referred to as DDoS attacks.[3] These attacks are usually launched from a large number of virus-infected zombie computers, and their power is proportional to the number of zombie computers involved. In recent years, in response to the popularity of the Internet, modern computers are greatly superior to their precursors in terms of performance. Thus, current DoS attacks are mostly distributed ones, in order to have sufficient power to paralyze their targets.

2.2 Methods for detecting DDoS attacks

Intrusion detection may be divided into two major types, namely, signature-based detection and anomaly-based detection.[3] Signature-based detection is mainly based on abnormal signature databases that are built using past attack-associated behaviors and allow it to identify malconduct according to the signatures recorded in these databases. Meanwhile, anomaly-based detection involves machine learning where data sets are used for training so as to enable machines to recognize attack-associated behaviors. Anomaly-based detection is particularly powerful when detecting zero-day attacks and unknown attacks.[4–7]

2.3 Artificial neural networks

An artificial neural network is a parallel computing model similar to a human nervous system, and it is supported by an information processing technique inspired by the human brain and nervous system. Such a network is also known as a parallel distributed processing model or a connectionist model.[6] Artificial neural networks are composed of processing elements (PEs), which are their basic components. Inputs in an artificial neural network are processed by summation functions, activity functions, and transfer functions into outputs. These transfer functions and activity functions are what differentiate different neural network models.

Back-propagation neural networks are currently the most representative and most extensively used type of neural network. A back-propagation artificial neural network features additional hidden layers and the use of smooth, differentiable nonlinear transfer functions, which address the failure of sensors in solving the exclusive-OR problem. A back-propagation artificial neural network has a similar architecture to a supervised learning network and a feedforward network. It primarily relies on computation using the gradient steepest descent method, from which weights are fine-tuned, thereby minimizing deviation between its inputs and outputs. It may be termed as a mapping between inputs and outputs. A back-propagation artificial neural network

usually uses a sigmoid function as its activity function, which is

$$p(x) = \frac{1}{1+e^{-x}}. \tag{1}$$

During operation, a back-propagation artificial neural network performs both forward propagation and backward propagation. Forward propagation relates to a process where data are inputted from the input layer, followed by being weighted, before being propagated to the hidden layer, and then transfer functions are used to calculate outputs and deviations corresponding to individual neurons. In the event that forward propagation fails to receive the expected outputs, the backward propagation stage begins to apply the actual outputs and the expected outputs to the energy function, and the gradient steepest descent method is used for correction of the weighted values. These two directions of propagation can never take place at the same time. Throughout the training, forward and backward propagations are alternately performed for correction of the weights, so as to obtain an optimal set of weighted values, whereas only forward propagation is used to produce outputs during testing.

2.4 Apache Spark

Apache Spark is an open-source big-data computing framework developed by AMPLab of the University of California, Berkeley. Implementing in-memory computing, Apache Spark is 100 times faster than the traditional approach of Hadoop MapReduce that stores data in a hard drive, and it allows repeated inquiry of data in the memory, making it suitable for machine learning algorithms.[8]

Apache Spark is developed from Hadoop MapReduce but it does not need to write data generated during computation to a hard drive like MapReduce does. MapReduce must save data from computation to a hard drive, which is often a bottleneck in terms of I/O performance. Apache Spark can register the data generated during computation in a memory. Since memory operates much faster than a hard drive, the overall computing speed can be significantly improved. This improvement in speed is even more evident in operations that need to be performed repeatedly, as with iterative computation in machine learning, and this advantage makes Apache Spark more suitable for machine learning.[9]

3. Methods

3.1 Architecture of detection system

The detection system proposed in this study is composed of five computers, as shown in Fig. 1. Therein, one Master computer controls the cluster while the four Slave computers perform distributed computing. These computers play their respective roles in the cluster as follows:

A. Master: Hadoop Distributed File System (HDFS) Name Node, HDFS Secondary Name Node, Kafka Server, Apache Spark, ZooKeeper Server

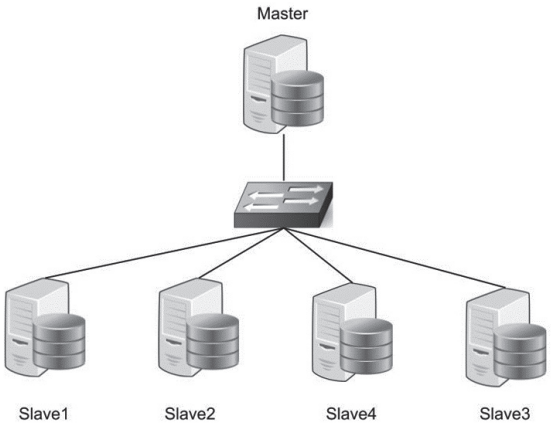

Fig. 1. Network architecture of the detection system.

B. Slave1: HDFS Data Node, Worker
C. Slave2: HDFS Data Node, Worker
D. Slave3: HDFS Data Node, Worker
E. Slave4: HDFS Data Node, Worker

3.2 Workflow of detection system

In this study, we employed a system architecture as shown in Fig. 2. Our detection analysis included six stages: (1) packet capture, (2) signature calculation, (3) data normalization, (4) data set building, (5) neural network check model, and (6) HDFS operation.

Experiments were conducted based on the seven signatures used for identifying DDoS attacks as defined by researchers[5] using the UCLA Dataset. During the experiments, these seven parameters showed significant variation when attacks occurred, and the accuracy was as high as 97%, demonstrating that the system effectively detected DDoS attacks.

DDoS attacks are attacks coming from distributed sources, and one master computer may receive attacks from various sources. In view of this, during detection, flows with identical targets and sources must be integrated so as to facilitate the subsequent calculation of signatures. On the basis of the research of Oshima *et al.* in 2009,[10] where window sizes for computation were set at 500, 1000, 2500, and 5000, we first found that in our experiments with these settings, when signature computation was conducted on the data set Normal using a window size of 500, more than half of the total flow-integration sessions received only one datum or no datum with a consistent source and target IP addresses, making calculation of the seven signatures impossible. Given the fact that more than half of the signatures became meaningless and consumed computing resources in vain, we decided to remove the window size of 500 from the experiments. Furthermore, in view of the fact that training the artificial neural network would be time-consuming, in this study, we also removed the window size of 2500 and only kept window sizes of 1000 and 5000 to save time, and experiments were conducted to identify the optimal window size.

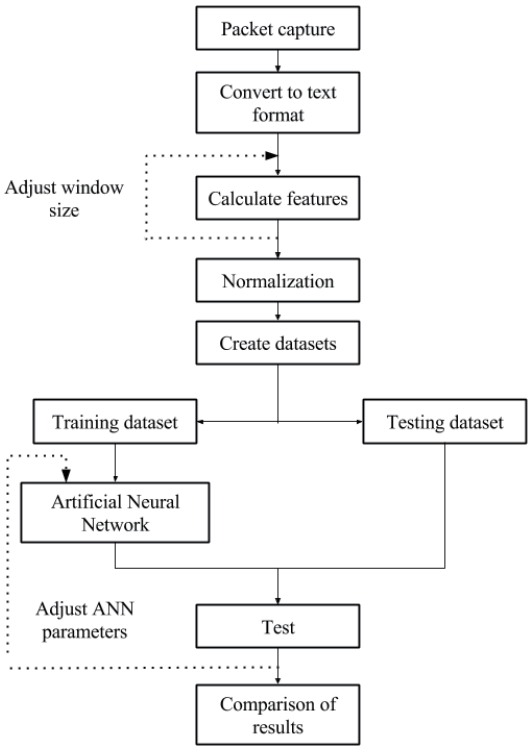

Fig. 2. Detection workflow.

4. Experiments and Analysis

4.1 Experimental settings

In this study, we used a data set combining the 1999 DARPA Intrusion Detection Evaluation Data Set and the CAIDA DDoS Attack 2007 Dataset. The former has only nonattacking flows in Week 1 and Week 3, while the latter only has attack-associated flows. Unlike other data sets that usually have mixed flows, these two selected datasets allow a clear distinction between attacking and non-attacking flows in experiments, which is favorable for assessment in our test stage.

4.1.1 Normal

The 1999 DARPA Intrusion Detection Evaluation Data Set, compiled by Lincoln Lab, Massachusetts Institute of Technology, was used for evaluating the efficiency of intrusion detection. It includes network flows collected over five weeks. The first three weeks provide training data, and the latter two weeks provide test data. Week 1 and Week 3 have normal network flows without any attacks. In this study, we used data from Week 1 and Week 3 as normal network flows involving no attacks.

4.1.2 Attack

The CAIDA DDoS Attack 2007 Dataset contains one-hour network flows from August 4, 2007 (from 20:50:08 UTC to 21:56:16 UTC), and has a size of 21 GB. All non-attacking components have been removed and only components related to DDoS attacks and the attacked host's responses are left, making it a data set containing only flows related to DDoS attacks.

4.2 Experiment methods

In this study, we included two stages of experiments. For the first stage, data were divided into two parts based on time, that is, 80% as training data and 20% as test data. The training data came from Monday to Thursday in Week 1, Week 3 of DARPA 1999, and the first 48 min of CAIDA2007. The test data came from Friday in Week 1 and Week 3 of DARPA1999, and the last 12 min of CAIDA2007. The optimal window size and network model were found by adjusting the window size, the neural network's learning rate, the number of neurons in the hidden layer and the max iterations. For the second stage, the optimal parameters identified in the first stage were taken and recombined into a 1-minute data set including normal and attack-associated flow packets for use in the experiments. Seven experiments were conducted. Experiments 1, 3, 5, and 7 used only normal flows, each with a duration of 13.5 s and Experiments 2, 4, and 6 used attack-associated flows, each with a duration of 2 s. The objective was to determine whether a trained neural network could forecast unknown situations.

4.3 Results of experiments

4.3.1 Experiments in Stage 1

In first stage of experiments, the average detection results for individual network parameters with a window size of 1000 are shown in Table 1, and the average detection results for individual network parameters with a window size of 5000 are shown in Table 2. The average detection results with window sizes of 1000 and 5000 are shown in Table 3.

Table 1
Average detection results for individual network parameters with a window size of 1000.

Results	Parameters								
	Neurons in hidden layer			Learning rate			Max iterations		
	5	9	18	0.1	0.5	1.0	3000	5000	7000
Accuracy	93.22%	85.60%	98.71%	90.66%	92.27%	94.60%	92.27%	92.27%	92.99%
True positive rate	93.35%	85.75%	98.93%	90.81%	92.45%	94.76%	92.42%	92.47%	93.13%
True negative rate	85.44%	75.55%	85.08%	81.31%	80.66%	84.11%	82.64%	79.46%	83.97%
False positive rate	14.56%	24.45%	14.92%	18.69%	19.34%	15.89%	17.36%	20.54%	16.03%
False negative rate	6.65%	14.25%	1.07%	9.19%	7.55%	5.24%	7.58%	7.53%	6.87%

Table 2
Average detection results for individual network parameters with a window size of 5000.

Results	Parameters								
	Neurons in hidden layer			Learning rate			Max iterations		
	5	9	18	0.1	0.5	1.0	3000	5000	7000
Accuracy	99.80%	98.82%	95.26%	99.14%	96.01%	98.73%	98.98%	98.35%	96.54%
True positive rate	99.81%	98.84%	95.28%	99.16%	96.02%	98.75%	99.00%	98.37%	96.56%
True negative rate	91.06%	86.15%	87.68%	88.62%	89.56%	86.71%	88.54%	87.91%	88.44%
False positive rate	8.94%	13.85%	12.32%	11.38%	10.44%	13.29%	11.46%	12.09%	11.56%
False negative rate	0.19%	1.16%	4.72%	0.84%	3.98%	1.25%	1.00%	1.63%	3.44%

Table 3
Average detection results with window sizes of 1000 and 5000.

Window size	Accuracy	True positive rate	True negative rate	False positive rate	False negative rate
1000	92.51%	92.68%	82.02%	17.98%	7.32%
5000	97.28%	97.33%	87.55%	12.45%	2.67%

4.3.2 Experiments in Stage 2

In second stage of experiments, Figs. 3–9 are real-time detection results of Experiments 1–7, respectively. The data for real-time detection in the seven experiments are summarized in Table 4.

5. Discussion

Table 4 summarizes data obtained from seven experimental trials in real-time detection. Compared with the results of the experiments in Stage 1 that also used a window size of 5000 and the same artificial neural network parameters, it is observed that the accuracy, true positive rate, and true negative rate decreased by 12.669, 12.253, and 18.882%, while the false positive rate and false negative rate increased by 18.882 and 12.253%, respectively.

The experiments in the second stage used the same artificial neural network model, but their results are substantially inferior to those from the experiments in the first stage. Possible reasons for this difference include: (A) the randomly recombined network flows were unknown to the artificial neural network detection system, so the detection rates decreased, and (B) owing to the limit imposed by the Timeout value of the Apache Spark real-time detection system, the experiments using some specific window sizes could not meet the predetermined number of packets, and the detection rates were thus adversely affected.

As can be seen in the results of the experiments, either in Stage 1 or 2, most of the false positive rates are greater than the false negative rates. This means that the accuracy was higher when detecting attacks than when discerning nonattacking signatures. The possible reasons for this include the following. (A) There was a wide difference between the numbers of samples of attacking and nonattacking signatures. For example, in the training data using a window size of 5000, there were 74920941 samples related to attacking signatures and 151987 samples related to nonattacking signatures. If errors occurred in the nonattacking situations, then the increase

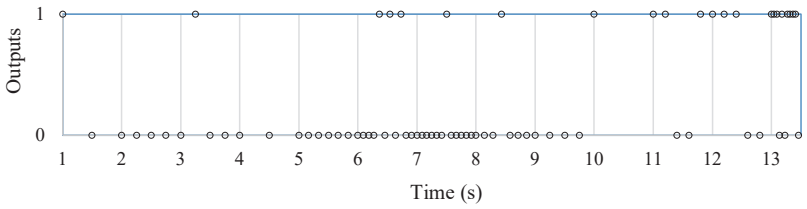

Fig. 3. (Color online) Real-time detection results of Experiment 1 (Nonattacking).

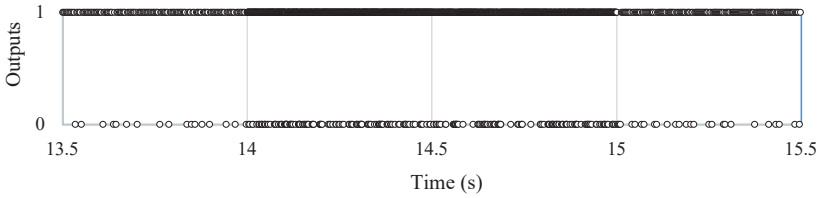

Fig. 4. (Color online) Real-time detection results of Experiment 2 (Attacking).

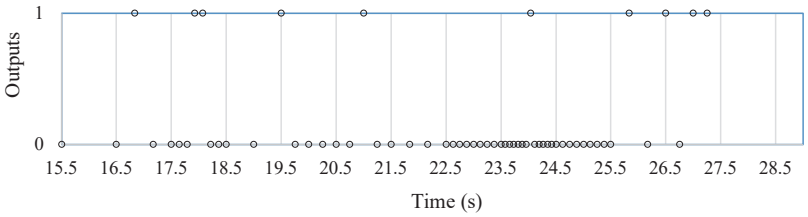

Fig. 5. (Color online) Real-time detection results of Experiment 3 (Nonattacking).

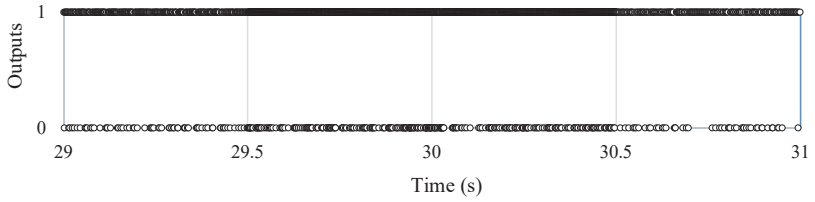

Fig. 6. (Color online) Real-time detection results of Experiment 4 (Attacking).

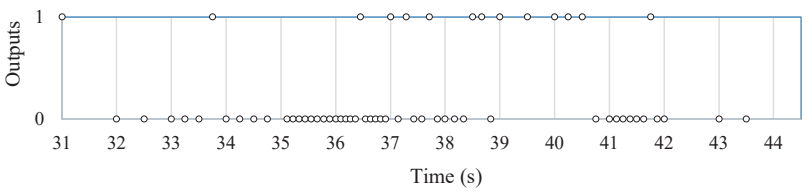

Fig. 7. (Color online) Real-time detection results of Experiment 5 (Nonattacking).

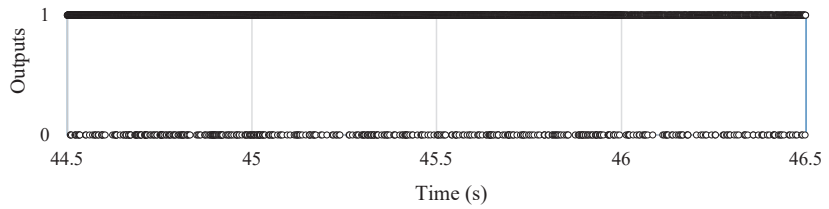

Fig. 8. (Color online) Real-time detection results of Experiment 6 (Attacking).

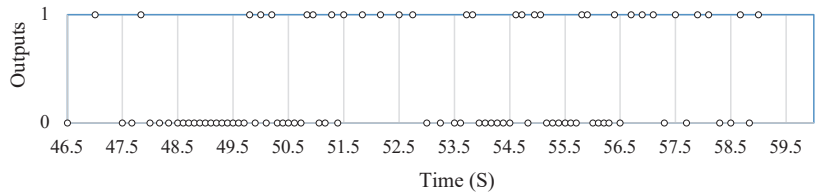

Fig. 9. (Color online) Real-time detection results of Experiment 7 (Nonattacking).

Table 4
Summary of real-time detection results.

	Accuracy	True positive rate	True negative rate	False positive rate	False negative rate
1 (Nonattacking)	70.27%	—	70.27%	29.73%	—
2 (Attacking)	88.02%	88.02%	—	—	11.98%
3 (Nonattacking)	83.33%	—	83.33%	16.67%	—
4 (Attacking)	85.33%	85.33%	—	—	14.67%
5 (Nonattacking)	76.67%	—	76.67%	23.33%	—
6 (Attacking)	88.98%	88.98%	—	—	11.02%
7 (Nonattacking)	65.12%	—	65.12%	34.88%	—
Overall Results	87.18%	87.61%	72.86%	27.14%	12.39%

in the error rate would be greater than that in the attacking situations. (B) In this study, we performed detection based on anomalies but not set rules and conducted classification according to signatures because this approach would be favorable to me detection of zero-day attacks of unknown types and using unknown methods. This means, however, that in certain situations, misjudgments could easily happen when the identified signatures of a nonattacking flow are similar to those of an attacking flow, and vice versa. For example, remote monitoring requires the regular transmission of a large number of packets with a fixed size and this could be identified as an attack. In contrast, packets related to a real attack are unlikely to be considered as normal packets. This is mainly because if an attack has a rate similar to that of normal packets, its power and impact would be significantly reduced.

6. Conclusions

In this study, we aimed to provide a fast and effective way to detect DDoS attacks. The detection of these attacks is difficult because their sources are hard to track and their forms

are diverse. It is impossible to accurately identify attacks using only one or a few signatures, and traditional approaches like an IPS, IDS, and firewalls have been proven to be ineffective in protecting against these attacks.

We used a back-propagation artificial neural network for this study, and obtained a detection rate as high as 99.80% in our Stage 1 experiments, while true positive, true negative, false positive, and false negative rates were 99.81, 91.06, 8.94 and 0.19%, respectively. The experiments in the second stage were conducted by recombining the parameters for Stage 1 into a 1-min-long data set of network flows and using a real-time detection system. The best accuracy obtained was 87.18%, while true positive, true negative, false positive, and false negative rates were 87.61, 72.86, 27.14, and 12.39%, respectively. As the results from Stage 2 were inferior to those from Stage 1, it is clear there is still room for improvement of the artificial neural network model in terms of replicability and the Timeout value setting of the detection system.

As demonstrated by the results from the two stages, most of the false positive rates were higher than the false negative rates. Although the high false positive rates would not open the system to attacks, they blocked normal users outside, causing inconvenience. On the other hand, the false negative results mean that attacks were likely to pass the detection system, and although the false negative rates were relatively low, their consequences should not be underestimated. Like accuracy, these two kinds of error detection can be improved by adjusting the artificial neural network's parameters and increasing the training data volume.

There have been many studies on using artificial neural networks for DDoS detection. While the earlier studies obtained overall accuracies similar to those of this study, this study differentiated itself by integrating an artificial neural network with a big-data computing framework, and since the framework is easy to build and has good scalability, it allows more computers and more Kafka Producers receiving packets to be incorporated to form an even larger detection system capable of dealing with increasingly huge and diverse DDoS attacks. In addition, the use of such a big-data computing framework promises fast processing of mass historical data. This means that more training data can be added in the future to further improve its accuracy and replicability.

References

1 Top Ten Reviews: Homepage of Top Ten Reviews, http://www.toptenreviews.com/business/internet/best-ddos-protection-services/prolexic-ddos-mitigation-services-review/ (accessed October 2017).
2 Forbes Cyber Security: Homepage of Forbes, https://www.forbes.com/sites/gartnergroup/2017/08/28/3-ways-to-defeat-ddos-attacks/#6295efb0da78 (accessed October 2017).
3 S. M. Lee, D. S. Kim, J. H. Lee, and J. S. Park: Comput. Math. Appl. **63** (2012) 501.
4 Z. Teng, C. T. Dan, and K. Q. Lo: IEEE 7th Int. Symp. Cyberspace Safety and Security (IEEE, 2015).
5 K. Reyhaneh and F. Ahmad: Int. Conf. Network and Electronics Engineering IPCSIT 11 (2011).
6 S. Alan, E. O. Richard, and R. Tomasz: Neuro Comput. **172** (2016) 385.
7 M. L. Sang, S. K. Dong, J. H. Lee, and S. P. Jong: Comput. Math. Appl. **63** (2012) 501.
8 R. Sandy, L. Uri, O. Sean, and W. Josh: Patterns for Learning from Data at Scale (O'Reilly Media, Inc., California, 2015).
9 K. Holden, K. Andy, W. Patrick, and Z. Matei: Learning Spark, Lightning-Fast Big Data Analysis (O'Reilly Media, Inc., California, 2015).
10 S. Oshima, N. Takuo, and S. Toshinori: 2nd Int. Conf. Computer Science and Its Applications (2009) 10.