This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

NetFlow Monitoring and Cyberattack Detection Using Deep Learning with Ceph

CHAO-TUNG YANG[1](Member, IEEE), JUNG-CHUN LIU[1], ENDAH KRISTIANI[2,3], MING-LUN LIU[1], ILSUN YOU[4], and GIOVANNI PAU[5](Member, IEEE).

[1]Department of Computer Science, Tunghai University,
No.1727, Sec.4, Taiwan Boulevard, Xitun District, Taichung City, Taiwan (R.O.C.) 40704
(e-mail: Chao-Tung Yang ctyang@thu.edu.tw, Jung-Chun Liu jcliu@thu.edu.tw, Ming-Lun Liu s90407p@gmail.com)
[2]Department of Industrial Engineering and Enterprise Information,
Tunghai University, Taichung City, Taiwan, (R.O.C.) 40704 (email: endahkristi@gmail.com)
[3]Department of Informatics, Krida Wacana Christian University, Jakarta, Indonesia 11470
[4]Department of Information Security Engineering Soonchunhyang University
646 Eupnaeri, Sinchangmyeon Asansi, Chungcheongnamdo, Republic of Korea 31538 (email: ilsunu@gmail.com)
[5]Faculty of Engineering and Architecture,
Kore University of Enna, 94100 Enna - Italy (email: giovanni.pau@unikore.it)

Corresponding author: Ilsun You (e-mail: ilsunu@gmail.com).

ABSTRACT Figuring the network's hidden abnormal behavior can reduce network vulnerability. This paper presents a detailed architecture in which the collected log data of the network can be processed and analyzed. We process and integrate on-campus network information from every router and store the integrated NetFlow log data. Ceph is used as an open-source distributed storage platform that offers high efficiency, high reliability, scalability, and preliminary preprocessing of raw data with Python, removing redundant areas and unification. In the subanalysis, we discover the anomaly event and absolute flow by three times of standard deviation rule. Keras has been used to classify in-time data collected via a cyber-attack and to construct an automatic identifier template through the Recurring Neural Network (RNN) test. The identification accuracy of the optimization model is around 98% in attack detection. Finally, in the MySQL server, the results of the real-time evaluation can be obtained, and the results of the assessment can be displayed via ECharts.

INDEX TERMS Data Storage, Ceph, Deep Learning, Cyberattack, NetFlow Log

I. INTRODUCTION

There is no question that in the rapidly moving age of information the Internet has become an integral component of human life. Nevertheless, it also masks an unfair network behavior in the Internet world. Figuring the network's hidden abnormal behavior can reduce network vulnerability [1]. In the past, network traffic information records have been maintained in the databases but a range of network connections have been developed through technological development. The regular database was hard to cope with the increasing information [2]. To store these materials, it is critical to creating a high-performance, reliable, and scalable storage environment [3].

The increase in hardware machinery and deep learning have affected cyberattack behavior. In order to tackle this danger, advancement techniques and the development of new policies are required. Monitoring cyber attacks using deep learning technology will currently be a major problem [4].

In this experiment, we used information from NetFlow which function includes no packet content and only the necessary traffic configuration data. The benefit of NetFlow is the lightweight and fastness of the entire data packet. Such features could be ideal for extraordinary detection in a crowded network environment [5].

In this research, our goal is to implement Ceph as a storage environment, i.e., save the generated NetFlow data, and practice Keras to analyze the stored NetFlow data, visualize the NetFlow to monitor, and identify the threat. Specific goals are listed below:

1) Set up a decentralized Ceph storage environment and

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

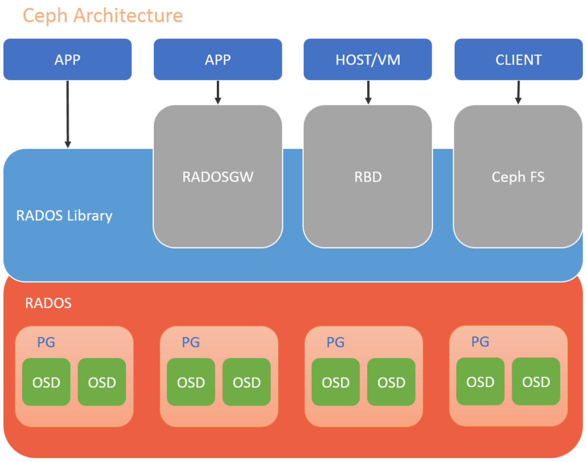


FIGURE 1: Ceph Architecture.

use CRUSH algorithm to distribute information in a balanced way;
2) Instantly store the produced NetFlow information in the Ceph setting as a historical database;
3) Use deep learning to analyze NetFlow information in real-time and assess various algorithms and designs;
4) Visualize NetFlow data analytical results on the webpage.

II. BACKGROUND REVIEW AND RELATED WORKS

In this section, some background knowledge is reviewed for later use of system design and implementation.

A. BACKGROUND REVIEW

1) Ceph

Ceph [6] is open source software that is extendable to large-scale storage fields specified by software. It can store large amounts of data at a lower price, provide the balancing system, and information reliability. It supports three types of installations: block storage, object storage, and file system storage on the same platform to enhance future storage environments. The lower layer of Ceph is a clustered storage area. In the future, when needed, it can only add the nodes to the cluster using a scale. Ceph has automated repair and management features that can improve the efficiency of storage data placing on system by copying information simultaneously to different nodes using a CRUSH [7] algorithm. If a cluster node crashes, it does not affect the entire operation of the storage system.

Figure 1 depicts the three-layer underlying architecture of Ceph. The first layer provides the storage of objects, blocks, and files. The second layer extracts the underlying data through the RADOS function library. The third layer comprises a storage space composed of a plurality of RADOS nodes.

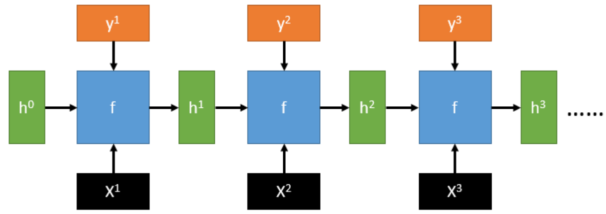


FIGURE 2: RNN Architecture.

2) Keras

Keras [8] is a deep learning open-source library in Python that runs on TensorFlow, CNTK, or Theano. In the development of Keras, accelerated experimentation is promoted. It is designed to be easy to use, modular and expandable. Keras has already trained the model's input layer, hidden layer, and output level, so it can perform quick and easy operations, and only the right parameters need to be added. Keras supports both convolution and recursion neural networks and can be trained on CPUs and GPUs by using deep learning models.

3) Recurrent Neural Networks

For the data highly correlated in space and time series, RNN(Recurrent Neural Network) [9] is used. In conventional neural networks, all inputs and outputs are independent from each other. This method may not be suitable for all time series data circumstances. In cases where previous data must be retained to inform the results, this sort of problem can be resolved by using RNN. The concept of RNN is to cyclically transfer its own network data. The output of the network is based on previous calculations so that a wider range of time series input structures can be handled. Figure 2 describes the Recurrent Neural Networks Architecture.

4) Long Short Term Memory Network

Long Short Term is a unique form of RNN that adds data about "long-term addiction" via memory function [10], primarily to fix the issue of gradient disappearance and gradient explosion during long-sequence practice. There is an additional Cell state updated with the moment compared to the overall RNN. To determine memory storage and use, the Forget Gate, Input Gate, and Output Gate are used. Figure 3 shows the Long Short Term Memory Network architecture, while Figure 4 represents the LSTM internal structure.

When controlling the transmission status through three Gates, it is necessary to remember the essential long-term data and forget the insignificant data. It can fix the issue in training of gradient disappearance and explosion. Nonetheless, the training difficulty is increased due to the enhanced amount of introduced parameters. The equation of LSTM is described in the equation 1-3 as follows:

$$c^t = z^f \odot c^{t-1} + z^i \odot z \tag{1}$$

$$h^t = z^o \tanh(c^t) \tag{2}$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

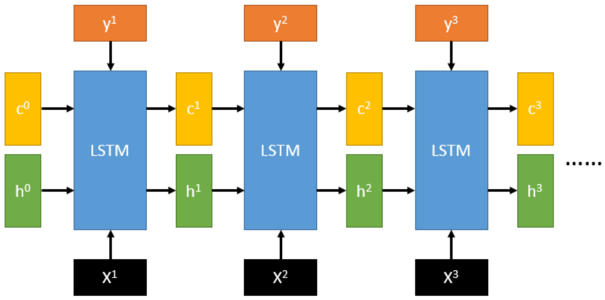


FIGURE 3: LSTM Architecture.

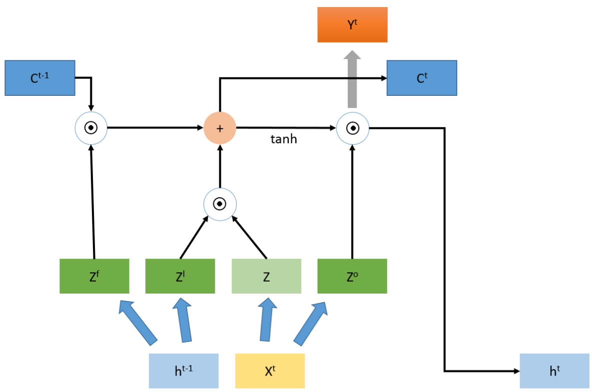


FIGURE 4: LSTM Internal Structure.

$$y^t = \sigma(W' h^t) \tag{3}$$

5) Gate Recurrent Unit

A sort of cyclic neural network is also the Gate Recurrent Unit (GRU) [11] [12]. During long-term sequence practice, the same as LSTM, it is necessary to address the issue of gradient explosion and gradient disappearance. The GRU simplifies an Update Gate for the LSTM Input Gate and Forget Gate. Compared to LSTM, one parameter is used that can accelerate the execution during practice and decrease memory utilization. GRU can also produce outcomes comparable to LSTM regarding the outcomes. GRU's practicality will be higher, given the hardware's computing power and time cost. Figure 5 presents the Gate Recurrent Unit Network architecture, while Figure 6 represents the GRU internal structure.

The equation of GRU is defined in the equation 4-7 as follows:

$$z = \sigma(x_t U^z + s_{t-1} W^z) \tag{4}$$

$$r = \sigma(x_t U^r + s_{t-1} W^r) \tag{5}$$

$$h = \tanh(x_t U^h + (s_{t-1} \circ r) W^h) \tag{6}$$

$$s_t = (1 - z) \circ h + z \circ s_{t-1} \tag{7}$$

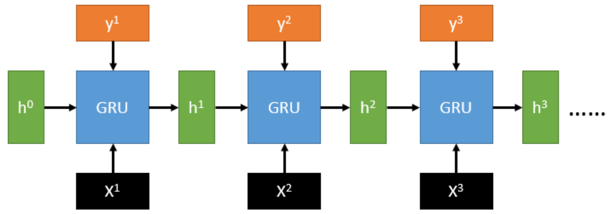


FIGURE 5: GRU Architecture.

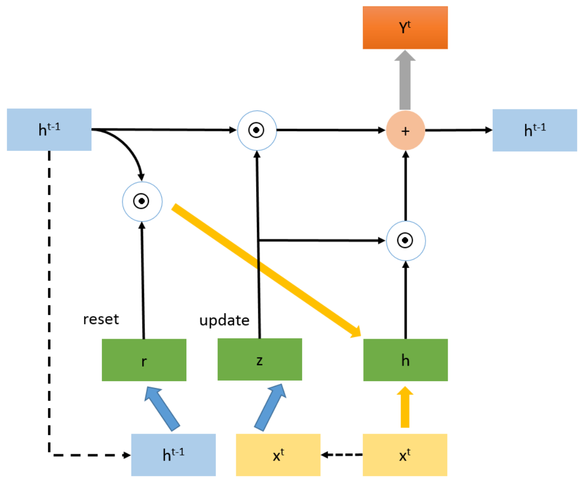


FIGURE 6: GRU Internal Structure.

6) Grafana

Grafana is an open-source visualization tool that is most often used with Graphite, InfluxDB, Elasticsearch, and Logz.io. Essentially, it is a mobile graphite-web substitute that helps users to build and edit dashboards. This tool provides a single parser that allows easy metric and function editing. Considering the Grafana quick customer-side render, the users can design detailed charts with smart axis formats (such as lines and points) even over long periods using Flot as a default option.

7) Bootstrap

Bootstrap [13] is a collection of open-source front-end frameworks for the creation of web pages and applications, including frameworks for HTML, CSS, and JavaScript, offering typography, forms, buttons, navigation and numerous other elements and extensions for JavaScript. The graphical content and database of the same page can be presented on the phone or screen via the CSS3+JQuery website technology.

8) ECharts

ECharts [14] is a free Javascript chart library, smoothly running on PCs and mobile devices. ECharts is today compatible with most browsers, and the underlying layer is based on ZRender's lightweight class Canvas library. It provides visualization charts of information that are intuitive and extremely customizable.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

B. RELATED WORKS

In a document released by Knowledge-Base Systems [15], Hongyu Liu et al. suggested an immediate port-to-port detection technique using PL-CNN (i.e., a classification technique based on a convolutional neural network payload) and PL.-RNN (Neural Network-Based Payload Classification Method) to detect attacks. Without feature engineering, the two methods learn feature representation from the original payload and support end-to-end detection. In this paper, we recognize that in complexity and time-consuming, more precise, and faster detection, deep learning can be distinct from traditional machine learning function engineering. In [16], Tae-Young Kim et al. suggested a C-LSTM neural network to detect anomalies in network traffic data efficiently. This feature is a technique to extract time and space information from the initial data automatically. It can attain very excellent anomaly detection efficiency in network traffic data by extracting the CNN's spatial characteristics and the LSTM model's temporal characteristics. In [17], Markus Ring et al. suggested a new technique to produce pseudo-NetFlow information based on the generation of an anti-neural network (GAN), which can produce excellent outcomes for detection and generation. The primary challenge is that GAN can handle only ongoing characteristics, and NetFlow generally has various characteristics of classification. Therefore, the authors suggests three distinct techniques of preprocessing and applies this technique to the information set of CIDDS-001. Experiments demonstrate that high quality can be produced by two of the three techniques. Tuan A Tang et al. proposed a flow-based anomaly detection system that uses deep learning [18]. Jihyun Kim et al. built a deep learning model that applied long-term memory architecture (LSTM) to recurrent neural networks (RNN) and trained using the 1999 KDD Cup dataset [19]. The obatined results confirm that the deep learning technique is valid for IDS by performance testing. In [20], Rui Fu et al. applied the LSTM and GRU techniques to predict traffic flow and assessed the efficiency of both methods. They discovered that LSTM and GRU NNs perform better than ARIMA and that GRU NNs perform a little better than LSTM NNs and generally converge quicker than LSTM.

Considering that the findings of techniques for botnet identification are not generally contrasted, Garcia et al. provided two techniques of botnet detection, i.e., BClus and CAMNEP [21]. They compare three botnets using actual information sets using several methods of detection, such as BClus, CAMNEP, and BotHunter. Moreover, the authors analyze the effect on each technique of botnet activity and if each technique can be best suited to separate information sets of the botnet stage. It is useful to note that the findings obtained in [21] have been taken into account in our botnets testing concepts. In the solution introduced in [22], Zhang et al. suggested a new technique for identifying anomalous conduct in network performance information, consisting of two machine learning algorithms: Boosted Decision Tree (BDT) and easy Feedforward neural network structure. The authors evaluate and compare each algorithm's efficiency. It is crucial to perceive that, in the tests carried out in [22], the information set's conduct does not fulfill expectations.

Many scientists use machine learning techniques to detect cyber-attacks by classifying payloads. For instance, in [23], Wang K et al. proposed a payload-based intrusion detection method. On the contrary, Rafał Kozik et al. used the flexibility of cloud-based architecture, as well as the latest advances in the field of large-scale machine learning, for shifting computationally more expensive and demanding operations [24] [5]. Going to the cloud to conduct traffic classification effectively using edge computing capacities, based on complex ELMs (extreme learning machine models) pre-built on the cloud, can decrease the overhead efficiency on edge devices. The detection of anomalies is the practice of recognizing objects or occurrences that do not conform to the anticipated conduct or are not linked to other items in the dataset. For these reasons, the authors of [24] present a technique for combining NetFlow with an Extreme Learning Machine (ELM) classifier trained in the distributed setting of the Apache Spark framework. The approach proposed by the authors uses the Map-Reduce model to extend the ELM classifier's training process and to conduct malware detection algorithms based on NetFlow. The findings reported on the benchmark data set show that the suggested ELM-based NetFlow assessment can be regarded as a reliable tool for network event detection.

The preprocessing of data is commonly acknowledged as a significant phase in anomaly detection. Davis et al. [25] reviewed the data preprocessing techniques used by the anomaly-based Network Intrusion Detection System (NIDS), focusing on what network traffic features are being used and what specific constructs and selection methods are being used. In [26], Hofstede et al. explain all stages of NetFlow's traffic output and typical traffic surveillance settings, covering all stages of NetFlow traffic surveillance. A new unsupervised technique of identification of anomalies was suggested by Duygu Sinanc Terzi et al.in [27]. This solution aims to determine the anomaly on a particular IP created by a UDP flood attack. This strategy is applied in case studies on government NetFlow information.

Our experimental set up split into two sub-features analysis, anomaly detection and cyberattacks identification. In this case, deep learning was used to build a model of recognition attacks.

III. SYSTEM DESIGN AND IMPLEMENTATION

A. SYSTEM ARCHITECTURE

In the Ceph storage setting, the Python language was used to automate the download and storage of full NetFlow campus information. With the use of Python for evaluation and judgment and through Matplotlib to visualize the information, we can evaluate and optimize the proposed model. Moreover, it is possible to save MySQL live analysis outcomes. Finally, the proposed system uses ECharts to show outcomes of customized assessment and combine Bootstrap to generate

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

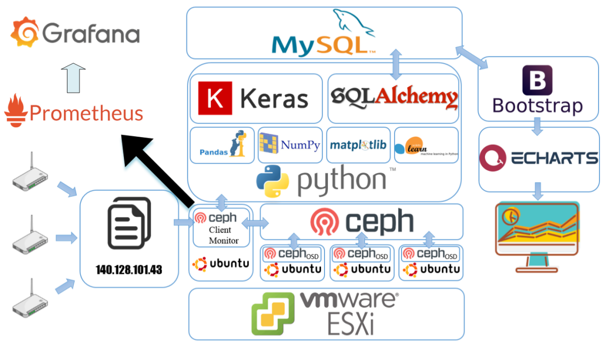


FIGURE 7: System Analysis Process.

responsive web pages. Figure 7 shows the system analysis process of the proposed system.

Four virtual machines were built as Monitor, OSD, and MDS for the Ceph storage environment in the vSphere ESXi environment [28]. As nodes for distributed storage of Ceph, a Monitor and three OSDs are used. Python reads the information in real-time for pre-processing information directly. The Keras library is used through the Python language to analyze the pre-processed data, and the results of the analysis are read and written through the sqlalchemy package into the MySQL database. The findings are finally presented on the webpage.

B. SYSTEM SERVICES

The services supplied by our scheme include information collection, data storage, preprocessing of information, data analysis, and visualization of information.

1) Data Collection

A collection of information gathered by campus routers is used to generate the entire NetFlow campus information. The Python program frequently captures and stores NetFlow information updated every 5 minutes in the Ceph storage setting.

2) Data Storage

NetFlow information is split into two components in the Ceph storage setting. One component is the present time's real-time information region, and the other portion is the past period's historical information region. When updating NetFlow information, Ceph Monitor will immediately crawl new information. The captured real-time information will be maintained for information pre-processing and analysis in the fast-paced Ceph Monitor system. After the end, the data will be moved to the distributed storage historical data area consisting of three Ceph OSDs. The time cost of storing and handling information in the historical information region can be significantly decreased by prioritizing information processing in Ceph Monitor.

The CRUSH algorithm is used to calculate the PG-ID to be stored in each data in the section where Ceph data is stored and then to calculate the stored OSD position. OSDs of different sizes will have different weights, and when stored, the data will be allocated taking into account the weight.

3) Data Preprocessing

This information can not be used straight for data analysis when a fresh piece of real-time data is downloaded. In order to enhance the data quality, we need to filter out essential information areas and remove noise that can influence the outcomes of the assessment. We need to transform the information into a standardized format because the units used for the information are not identical. With LabelEncoder [29] or OneHotEncoder [30], some information formats need to be converted to a format that can be analyzed. Lastly, the information is sorted into the input type for evaluation of the in-depth assessment model.

4) Data Analysis

The data analysis section is split into two sub-features:

- **Anomaly detection** Since the flow fluctuates periodically in days, the present time unit flow will be near to the past day's unit flow [31]. Moreover, a decent range will fulfill the rise in the amount of consecutively broken traffic. So it will be split into two components when performing anomaly detection. The first portion compares the past day's unit flow rate, and if the flow rate rise is higher than 100%, it will be considered abnormal. The second part calculates the difference between the current time and the previous time and determines the flow difference average and standard deviation within 30 days. In the event of very general distribution, a three-sigma rule will include at least 88.8% of the information in the range of three standard deviations. As a mild abnormal flow rate, we determined the unit flow rate of more than three standard deviations and determined that the unit flow rate of more than five standard deviations was a significant abnormal flow rate by the empirical rule. Through the above two techniques, when abnormal traffic happens, we can rapidly discover the moment.
- **Cyberattack Identification** The Attack Identification section will be used to train processed information using RNN model. The loss price, the precision value, and the training time of the three techniques evaluate the output of the distinct kinds of assault.

5) Data Visualization

Concerning data visualization, we connect to the MySQL database via PHP to get the JSON format data, then process and load the data via JQuery and Ajax. Finally, the loaded data is displayed via the ECharts custom chart. Through the above method, the front and back information can be readily visualized. Lastly, to obtain Responsive Web Design that is consistent with different devices, the webpage is coupled with the Bootstrap framework.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

TABLE 1: Computing Environment.

Name	CPU	RAM	Disk	OS
ESXi	6 CPUs x Intel(R) Core(TM) i7-3970X 3.50GHz	64G	2T * 6	ESXi-6.5.0

TABLE 2: Virtual Machine Environment.

Name	CPU	RAM	Disk	OS
master	6 vCPUs	40G	1.5T	Ubuntu 16.04
ceph-osd1	2 vCPUs	4G	2T	Ubuntu 16.04
ceph-osd2	2 vCPUs	4G	2T	Ubuntu 16.04
ceph-osd3	2 vCPUs	4G	2T	Ubuntu 16.04

C. DATA IMPLEMENTATION

In this work, under one physical host running ESXi, we set up four virtual machines, one as the primary computing node and the other three as the Ceph environment storage nodes. This cluster represents a full NetFlow analysis scheme. It includes Ceph, Tensorflow, Keras, and LAMP internal software. ESXi offers a remote monitoring interface such as the physical host's CPU, memory, and hard disk usage. Moreover, the use of virtual machines and activities is recorded.

Ceph mgr offers the tracking of clusters in Ceph, such as health clusters, current usage, OSD practice, and data related to I/O. A relational database for MySQL as a data store for outcomes of real-time data analysis is developed. The figure below indicates complete flows for successive periods, complete packets, and complete bytes.

IV. EXPERIMENTAL RESULTS

A. EXPERIMENTAL ENVIRONMENT

Using a physical machine fitted with ESXi, this experiment shows the physical machine environment in table1. This physical host involves as Ceph clusters with four virtual machines, one for Ceph Monitor and three for Ceph OSD2.

B. HISTORICAL FLOW CHANGES

We obtained the gathered real-time information for the processing of statistics for the implementation of historical information. Moreover, the total number of traffic sent for each period, the total amount of traffic used and the total size of the transport packet and the average number of items have been calculated. Figure 8 displays the complete flow change graph for 30 successive days.

It can be discovered the difference in traffic between each period and the previous time through historical information. By plotting the flow difference over time, we can see that, at certain moments, there are unusual flow changes. They are the point in time in which it is necessary to pay attention to when these traffic variations happen. Figure 9 demonstrates the variation of the flow difference for 30 consecutive days.

C. ABNORMAL ANALYSIS RESULT

From previous experiments, we found that the change in total flow will be consistent with the periodicity under normal conditions, and the change in the difference in flow will meet the specific interval. Through the two points above, we can discover the time points that do not fulfill the regular stream

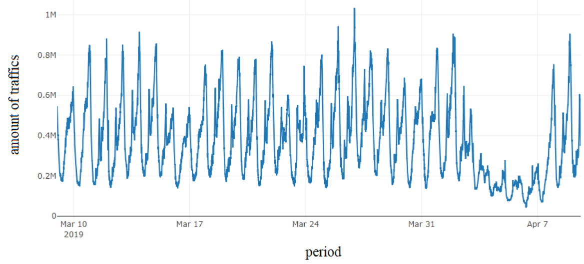


FIGURE 8: Total flow change graph for 30 consecutive days.

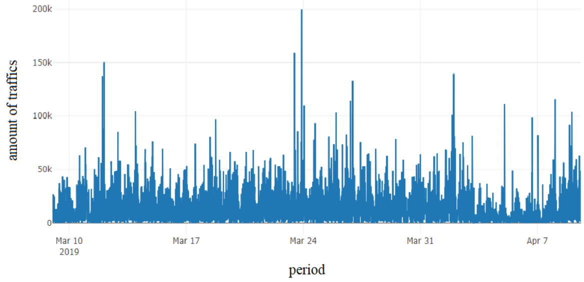


FIGURE 9: flow difference variation for 30 consecutive days.

adjustments and the significant flow difference shifts and mark these time points for the manager to perform subsequent inquiries. By comparing the total number of traffic at the same time as the previous day, we check whether the total number of traffic at the current time is abnormal. If the present time's complete traffic volume is higher than 1.5 times the previous day's total traffic volume, the present time point and complete traffic volume will be marked. We will discover the median and standard deviation for 30 successive days in the abnormal flow difference portion. According to the three-time standard deviation rule, outliers with more than three standard deviations can be found rapidly and marked as medium flow difference modifications, higher than five. The default deviation is labeled as a shift in elevated flow. Figure 10 shows the result of the abnormal analysis.

D. CYBERATTACK IDENTIFICATION RESULT

Cyberattack identifications are classified through Python's possible attack data and note the types of possible attacks. The accumulated data is integrated as a training set for attack identification, as shown in Figure 11. It has been used Keras to establish the RNN deep learning model, as shown in Figure 12. Subsequently, it has been trained and verified the finished training set through the Mean-Square Error (MSE) [32] as the evaluation criteria for the training loss value, as shown in Figure 13 and Figure 14. Parameter adjustment and optimization are performed without affecting the accuracy based on the equation 8, and the training cost is reduced, as shown in Figure 15 for 1 ms per step.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true}^{(i)} - y_{pred}^{(i)})^2 \quad (8)$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

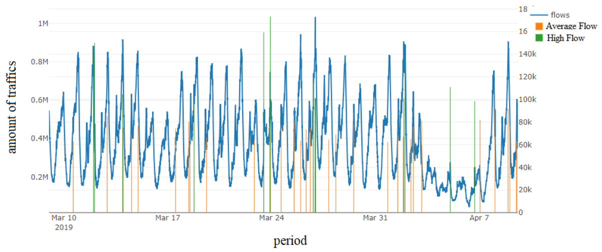


FIGURE 10: Abnormal Analysis Result.

	3	4	5	6	7	8	9	10	11	12	13
0	203.205.138.238	80.0	2.83	144.0	3.0	25.0	42.0	2.0	10.59.2.14	59370.0	CodeRed attack
1	199.93.56.125	80.0	0.00	642.0	8.0	25.0	46.0	2.0	10.30.2.12	55450.0	Nimda
2	5.136.241.233	445.0	0.00	52.0	1.0	25.0	46.0	2.0	10.30.3.119	61678.0	Worm attack
3	13.113.157.96	80.0	3.24	144.0	3.0	25.0	46.0	2.0	10.72.3.48	60774.0	CodeRed attack
4	119.46.206.250	80.0	9.01	144.0	3.0	25.0	45.0	2.0	10.67.2.24	59350.0	CodeRed attack
5	119.46.206.251	80.0	9.00	144.0	3.0	25.0	45.0	2.0	10.67.2.24	59377.0	CodeRed attack
6	213.147.126.18	80.0	1.43	144.0	3.0	25.0	41.0	2.0	10.6.0.14	1989.0	CodeRed attack
7	104.250.139.218	80.0	1.01	144.0	3.0	25.0	46.0	2.0	10.30.5.32	53033.0	CodeRed attack
8	104.250.139.218	80.0	1.01	144.0	3.0	25.0	46.0	2.0	10.30.5.32	53031.0	CodeRed attack
9	104.250.139.218	80.0	1.01	144.0	3.0	25.0	46.0	2.0	10.30.5.32	53022.0	CodeRed attack
10	104.250.139.218	80.0	1.01	144.0	3.0	25.0	46.0	2.0	10.30.5.32	53030.0	CodeRed attack

FIGURE 11: Attack Classification Result.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               1152
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 771
=================================================================
Total params: 34,947
Trainable params: 34,947
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 12: RNN Model Architecture.

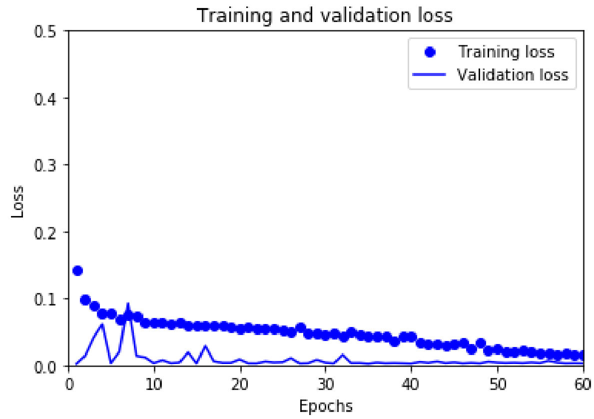


FIGURE 13: Training Loss Result.

E. GRAFANA METRIC PERFORMANCE

In this section, we present the metric measurement using Grafana. Figure 16 describes the Ceph monitoring status, while Figure 17 shows the time-series of Network Log.

F. VISUALIZATION OF RESULTS

Finally, the analysis results and attack identification were stored the real-time data in the MySQL database. It has been used PHP to load MySQL data and visualize the results through ECharts, and the Bootstrap framework to render on web pages of various mobile devices. Through this method we can monitor the instantaneous traffic changes, as depicted in Figure 18. Moreover, the network usage of colleges and dormitories is shown in Figure 19.

V. CONCLUSIONS AND FUTURE WORKS

This work provides a fully open-source software architecture that incorporates processing, evaluation, and monitoring of NetFlow logs. By using the Ceph architecture to store ever-increasing historical information, it can achieve the ability to obtain massive storage at a comparatively low cost. In terms of detecting cyberattacks, we use deep learning to build a model of recognition of attacks that can achieve recognition acceleration faster than traditional machine learning. Also, by updating the model through an updated continuously attack database, it can achieve greater precision. Finally, through the ECharts design page, help the decision-maker quickly recognize the network problem.

Some exploits lack vital features and cannot be recognized correctly, subject to the initial information constraints, and multiple cyberattacks are challenging to define. In the future, we hope to gain more in-depth information and use the ever-increasing role and equipment to find out more critical features to detect further cyber attacks. Besides, in the historical information segment, the reading assessment is slow, limited by hardware and network speed. We hope to be able to apply standardized machine requirements and decentralized computing architecture to achieve faster efficiency with high-speed networks.

ACKNOWLEDGMENTS

This work was sponsored by the Ministry of Science and Technology (MOST), Taiwan, under Grant No. 108-2221-E-029 -010 as well as the Soonchunhyang University Research Fund.

REFERENCES

[1] C.-T. Yang, S.-T. Chen, J.-C. Liu, Y.-Y. Yang, K. Mitra, and R. Ranjan, "Implementation of a real-time network traffic monitoring service with network functions virtualization," Future Generation Computer Systems, vol. 93, pp. 687–701, 2019.

[2] I. Kotenko, M. Kolomeets, A. Chechulin, and Y. Chevalier, "A visual analytics approach for the cyber forensics based on different views of the network traffic," Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 9, no. 2, pp. 57–73, 2018.

[3] C.-T. Yang, S.-T. Chen, W.-H. Cheng, Y.-W. Chan, and E. Kristiani, "A heterogeneous cloud storage platform with uniform data distribution by software-defined storage technologies," IEEE Access, vol. 7, pp. 147 672–147 682, 2019.

[4] C. Atapour, I. Agrafiotis, and S. Creese, "Modeling advanced persistent threats to enhance anomaly detection techniques," Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 9, no. 4, pp. 71–102, 2018.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

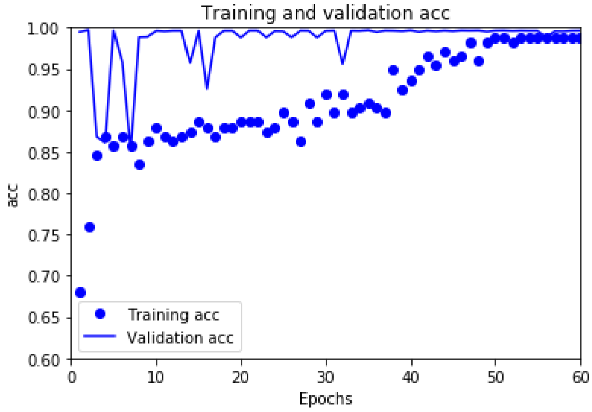
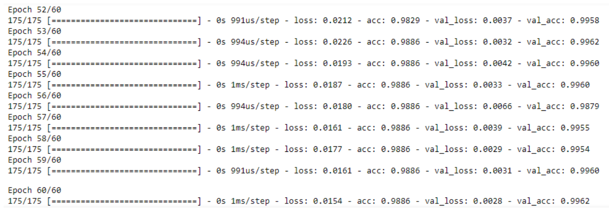


FIGURE 14: Training Accuracy Result.



```
Epoch 52/60
175/175 [==============================] - 0s 991us/step - loss: 0.0212 - acc: 0.9829 - val_loss: 0.0037 - val_acc: 0.9958
Epoch 53/60
175/175 [==============================] - 0s 994us/step - loss: 0.0226 - acc: 0.9886 - val_loss: 0.0032 - val_acc: 0.9962
Epoch 54/60
175/175 [==============================] - 0s 994us/step - loss: 0.0193 - acc: 0.9886 - val_loss: 0.0042 - val_acc: 0.9960
Epoch 55/60
175/175 [==============================] - 0s 1ms/step - loss: 0.0187 - acc: 0.9886 - val_loss: 0.0033 - val_acc: 0.9960
Epoch 56/60
175/175 [==============================] - 0s 994us/step - loss: 0.0180 - acc: 0.9886 - val_loss: 0.0066 - val_acc: 0.9879
Epoch 57/60
175/175 [==============================] - 0s 1ms/step - loss: 0.0161 - acc: 0.9886 - val_loss: 0.0039 - val_acc: 0.9955
Epoch 58/60
175/175 [==============================] - 0s 1ms/step - loss: 0.0177 - acc: 0.9886 - val_loss: 0.0029 - val_acc: 0.9954
Epoch 59/60
175/175 [==============================] - 0s 991us/step - loss: 0.0161 - acc: 0.9886 - val_loss: 0.0031 - val_acc: 0.9960

Epoch 60/60
175/175 [==============================] - 0s 1ms/step - loss: 0.0154 - acc: 0.9886 - val_loss: 0.0028 - val_acc: 0.9962
```

FIGURE 15: Model Training Process.

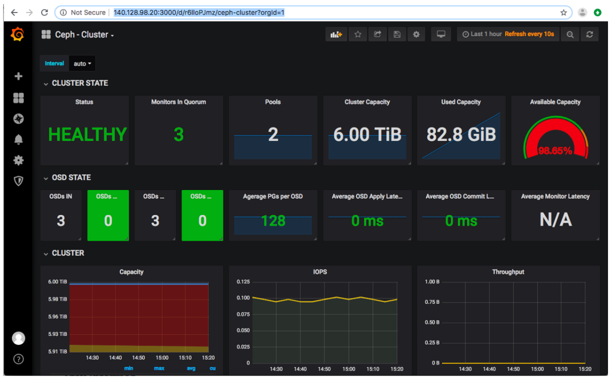


FIGURE 16: Grafana Ceph Monitoring.

FIGURE 17: NetFlow in time series stream.

[5] C.-T. Yang, E. Kristiani, Y.-T. Wang, G. Min, C.-H. Lai, and W.-J. Jiang, “On construction of a network log management system using elk stack with ceph,” The Journal of Supercomputing, pp. 1–17, 2019.
[6] “Ceph,” 2019. [Online]. Available: https://ceph.com/
[7] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, “Crush: Controlled, scalable, decentralized placement of replicated data,” in SC ’06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Nov 2006, pp. 31–31.
[8] “Keras,” 2019. [Online]. Available: https://keras.io/
[9] “Recurrent neural network,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network
[10] “Long short-term memory,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory
[11] “Gated recurrent unit,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Gated_recurrent_unit
[12] N. Ramakrishnan and T. Soni, “Network traffic prediction using recurrent neural networks,” in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec 2018, pp. 187–193.
[13] “Bootstrap,” 2019. [Online]. Available: https://getbootstrap.com/
[14] “Echarts,” 2019. [Online]. Available: https://echarts.baidu.com/
[15] H. Liu, B. Lang, M. Liu, and H. Yan, “Cnn and rnn based payload classification methods for attack detection,” Knowledge-Based Systems, vol. 163, pp. 332 – 341, 2019.
[16] T.-Y. Kim and S.-B. Cho, “Web traffic anomaly detection using c-lstm neural networks,” Expert Systems with Applications, vol. 106, pp. 66 – 76, 2018.
[17] M. Ring, D. Schlor, D. Landes, and A. Hotho, “Flow-based network traffic generation using generative adversarial networks,” Computers & Security, vol. 82, pp. 156 – 172, 2019.
[18] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep learning approach for network intrusion detection in software defined networking,” in 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Oct 2016, pp. 258–263.
[19] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, “Long short term memory recurrent neural network classifier for intrusion detection,” 2016 International Conference on Platform Technology and Service (PlatCon), pp. 1–5, 2016.
[20] R. Fu, Z. Zhang, and L. Li, “Using lstm and gru neural network methods for traffic flow prediction,” in 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Nov 2016, pp. 324–328.
[21] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” Computers & Security, vol. 45, pp. 100 – 123, 2014.
[22] J. Zhang, R. Gardner, and I. Vukotic, “Anomaly detection in wide area network meshes using two machine learning algorithms,” Future Generation Computer Systems, vol. 93, pp. 418 – 426, 2019.
[23] R. Kozik, M. Choras, M. Ficco, and F. Palmieri, “A scalable distributed machine learning approach for attack detection in edge computing environments,” Journal of Parallel and Distributed Computing, vol. 119, pp. 18 – 26, 2018.
[24] R. Kozik, “Distributing extreme learning machines with apache spark for netflow-based malware activity detection,” Pattern Recognition Letters, vol. 101, pp. 14 – 20, 2018.
[25] J. J. Davis and A. J. Clark, “Data preprocessing for anomaly based network intrusion detection: A review,” Computers & Security, vol. 30, no. 6, pp. 353 – 375, 2011.
[26] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, “Flow monitoring explained: From packet capture to data analysis with netflow and ipfix,” IEEE Communications Surveys Tutorials, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014.
[27] D. S. Terzi, R. Terzi, and S. Sagiroglu, “Big data analytics for network anomaly detection from netflow data,” in 2017 International Conference on Computer Science and Engineering (UBMK), Oct 2017, pp. 592–597.
[28] “Vmware esxi,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/VMware_ESXi
[29] “Labelencoder,” 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[30] “Onehotencoder,” 2019. [Online]. Available: https://scikit-

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

FIGURE 18: Instant Flow Change.

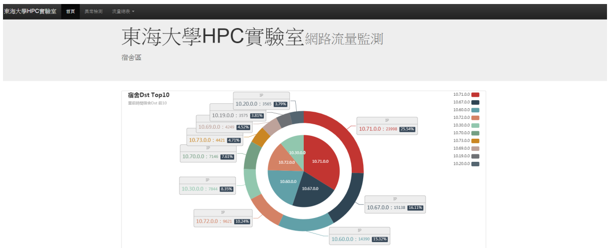


FIGURE 19: Comparison of the network usage of the dormitory.

learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
[31] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, "Anomaly detection methods in wired networks: a survey and taxonomy," Computer Communications, vol. 27, no. 16, pp. 1569 – 1584, 2004.
[32] "Mean squared error," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error

CHAO-TUNG YANG Chao-Tung Yang is Distinguished Professor of Computer Science at Tunghai University in Taiwan. He received his Ph.D. in computer science from National Chiao Tung University in July 1996. In August 2001, he joined the Faculty of the Department of Computer Science at Tunghai University. He is serving in a number of journal editorial boards, including Future Generation Computer Systems, International Journal of Communication Systems, KSII Transactions on Internet and Information Systems, Journal of Cloud Computing. He has published more than 300 papers in journals, book chapters and conference proceedings. His present research interests are in cloud computing, big data, parallel computing, and deep learning. He is a member of the IEEE Computer Society and ACM.

JUNG-CHUN LIU received the B.S. degree in electrical engineering from National Taiwan University in 1990. He received M.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering at the University of Texas at Austin, in 1996 and 2004, respectively. He is currently an assistant professor in the Department of Computer Science at the Tunghai University, Taiwan. His research interests include cloud computing, embedded systems, wireless networking, artificial intelligence, digital signal processing, and VLSI design.

ENDAH KRISTIANI received M.S. degree in Electrical Engineering (Information Technology) from Universitas Gadjah Mada, Yogyakarta in 2007, Indonesia. In August 2007 joined the Faculty of Engineering and Computer Science, Department of Informatics Engineering at Krida Wacana Christian University (UKRIDA) Jakarta. Now she is a PhD student at Department of Industrial Engineering and Enterprise Information and join the High Performance Computing Laboratory, Tunghai University, Taichung, Taiwan.

MING-LUN LIU received the B.S. degree in Computer Science from Tunghai University, Taichung in 2017 and M.S. degree in Computer Science from Tunghai University, Taichung in 2019.

ILSUN YOU received the M.S. and Ph.D. degrees in computer science from Dankook University, Seoul, South Korea, in 1997 and 2002, respectively, and the second Ph.D. degree from Kyushu University, Japan, in 2012. From 1997 to 2004, he was with the Thin multimedia Inc., Internet Security Company Ltd., and Hanjo Engineering Company Ltd., as a Research Engineer. He is currently an Associate Professor with the Department of Information Security Engineering, Soonchunhyang University. His current research interests include Internet security, authentication, access control, and formal security analysis. He has served or is currently serving as a Main Organizer of international conferences and workshops, such as MobiWorld, MIST, SeCIHD, AsiaARES, and IMIS. He is the Editor-in-Chief of the Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). He is on the Editorial Board of Intelligent Automation and Soft Computing (AutoSoft), Journal of Network and Computer Applications (JNCA), International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Computing and Informatics (CAI), Journal of High Speed Networks (JHSN), and Security and Communication Networks (SCN). He is a Fellow of the IET. (Based on document published on 17 September 2019).

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2963716, IEEE Access

GIOVANNI PAU is a professor at the Faculty of Engineering and Architecture, Kore University of Enna, Italy. Prof. Pau received his Bachelor's degree in Telematic Engineering from the University of Catania, Italy, and both his Master's degree (cum Laude) in Telematic Engineering and Ph.D. from Kore University of Enna, Italy. Prof. Pau is the author/co-author of more than 60 refereed articles published in journals and conferences proceedings. He is a Member of the IEEE (Italy Section) and has been involved in the organization of several international conferences as a session co-chair and technical program committee member. Prof. Pau serves/served as a leading Guest Editor in special issues of several international journals and is an editorial board member as Associate Editor of IEEE Access (IEEE), Wireless Communications and Mobile Computing (Hindawi), EURASIP Journal on Wireless Communications and Networking (Springer), Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), Journal of Computer Networks and Communications (Hindawi), Wireless Networks (Springer), Human-centric Computing and Information Sciences (Springer), and Future Internet (MDPI). His research interests include Wireless Sensor Networks, Home Automation, Fuzzy Logic, Internet of Things, and Network Security.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.