Theme Article: Special Section on Consumer, Commercial, and Industrial IoT Security and Resilience

Secure and Resilient Artificial Intelligence of Things: A HoneyNet Approach for Threat Detection and Situational Awareness

Liang Tan
Sichuan Normal University, Chengdu, China
Chinese Academy of Sciences, Beijing, China

Keping Yu
Waseda University, Japan

Fangpeng Ming and Xiaofan Cheng
Sichuan Normal University, Chengdu, China

Gautam Srivastava
Brandon University, Brandon, MB, Canada
China Medical University, China

***Abstract*—Artificial Intelligence of Things (AIoT) is emerging as the future of Industry 4.0 and will be widely applied in consumer, commercial, and industrial fields. In AIoT, intelligent objects (smart devices), smart gateways, and edge/cloud nodes are subject to a large number of security threats and attacks. However, the traditional network security approaches are not fully suitable for AIoT. To address this issue, this article proposes a HoneyNet approach that includes both threat detection and situational awareness to enhance the security and resilience of AIoT. We first design a HoneyNet based on Docker**

Digital Object Identifier 10.1109/MCE.2021.3081874
Date of publication 19 May 2021; date of current version 3 May 2022.

technology that collects data to detect adversaries and monitor their attack behaviors. The collected data are then converted into images and used as samples to train a deep learning model. Finally, the trained model is deployed in AIoT to perform threat detection and provide situational awareness. To validate our scheme, we conduct HoneyNet deployment and model training on the SiteWhere AIoT platform and construct a simulation environment on this platform for threat detection and situational awareness. The experimental results demonstrate the feasibility and effectiveness of our solution.

Artificial Intelligence of Things (AIoT) is a collaborative application of Artificial Intelligence (AI)[12] and the Internet of Things (IoT).[3] Specifically, AIoT indicates a system that collects information in real time through various types of sensors and analyzes the data intelligently via machine learning in end devices, gateways, edge/cloud computing platforms, which includes positioning, comparing, prediction, and scheduling operations. AIoT currently has broad application prospects for home management, buildings management and occupancy, industrial manufacturing, smart cities, and so on. Many traditional open source IoT platforms have gradually been upgraded to AIoT platforms, including the Kaa IoT Platform, SiteWhere, ThingSpeak, and DeviceHive. In addition, AIoT is emerging as the future of Industry 4.0 and plays an essential role in consumer, commercial, and industrial applications.

In fact, due to the pivotal role that the AIoT plays in consumer, commercial, and industrial areas, the security threats and attacks that it faces are becoming more prominent and should be a priority concern.[4,5] In AIoT, either terminal devices, gateways or edges and cloud centers are all subject to external security threats and attacks. Moreover, the security threats are severe. For example, in January 2019, security researchers found that all 12 different AIoT devices sold by major retailers (e.g., Walmart and Best Buy) had serious security vulnerabilities and privacy issues; these devices included iHome, Merckry, Momentum, Oco, Practecol, TP-Link, Vivitar, Wyze, and Zmodo. At the end of 2019, a security surveillance scandal involving Amazon's Ring smart doorbell shocked the USA. Hackers could monitor users' homes, expose their WiFi passwords, and so on. The traditional network security protection methods are not fully suitable to prevent attacks against AIoT for the following reasons. First, it is difficult for the traditional network security protection to work directly on AIoT devices, which are composed of large numbers of terminals. Second, AIoT terminals have limited computing and storage resources while the traditional network security protection methods such as firewalls, intrusion detection systems, and antivirus software cannot function adequately when faced with this large device scale and resource limitations; thus, their security protection effects are limited. Therefore, it is urgent to design a secure and resilient AIoT that both performs threat detection and provides situational awareness. To meet this need, in this study, we designed a HoneyNet threat detection and situational awareness scheme to enhance the security and resilience of AIoT. The solution employs a security framework consisting of "HoneyNet detection—deep learning analysis—threat detection and situational awareness" to detect AIoT threats and respond to them in a timely manner to either completely prevent cyberattacks or reduce their damage. The main contributions of this article are the following.

1) We designed honeynets in AIoT to collect adversary behavioral data, convert them into images, and used the images as samples to train a convolutional neural network (CNN) in a data center (DC) close to the AIoT controller center.

2) We deployed the trained CNN models in multiple locations, including the control terminal of the field network, the control and enterprise networks, smart terminal devices, and smart IoT gateways to perform automated threat detection and provide situational awareness.

RELATED WORK

In recent years, with the development of artificial intelligence technology, researchers have found that training attack data information through machine learning algorithms can effectively improve the accuracy of network attack intrusion detection. Pan and Li[6] trained a support vector machine (SVM) to extract intrusion detection rules, and deployed those rules to device nodes in the IoT for intrusion detection. Based on a clustering feature set, data from the data source can be compared numerically to activate intrusion detection alarm based on a threshold value. Reddy *et al.*[7] used neurofuzzy and SVM algorithms that achieved good detection rates when applied to a specific dataset. Fuqun[8] applied a least-squares SVM to the network intrusion detection task, and achieved improved intrusion detection efficiency. Sahu and Mehtre[9] constructed network intrusion detection based on the decision tree, and experiments showed that this algorithm is suitable for intrusion detection location attacks. Zhang *et al.*[10] proposed an intrusion detection model based on an improved genetic algorithm and deep belief network. Their experiment effectively improves the recognition rate of intrusion attacks on the NSL-KDD dataset. Most of the above algorithms use public datasets (e.g., KDD99 and DARPA). Unfortunately, these types of data are no longer suitable as test data for modern AIoT security.

Deep learning models have made rapid progress and have gradually begun to be applied to network intrusion inspection. Alom *et al.*[11] applied deep belief networks (DBNs) to intrusion detection. A construction process for a DBN network model was proposed by processing a benchmark dataset using a DBN and then combining the result with neural networks for data classification. This approach was experimentally shown to improve the intrusion detection rate. Abolhasanzadeh[12] proposed a method for detecting attacks in big data using a deep autoencoder. Yin *et al.*[13] designed a deep learning method using a recurrent neural network for intrusion detection, that is suitable for high-precision classification modeling. Akter *et al.*[14] trained a deep learning-based intrusion detection framework and applied it to improve household appliance security.

Overall, threat detection and situational awareness for AIoT have become important aspects of IoT security and resilience[15] and will extensively impact consumer, commercial, and industrial IoT security and resilience.

HONEYNET THREAT DETECTION AND SITUATION AWARENESS FOR AIoT

In this section, we discuss about our proposed honeynet-based threat detection and situational awareness to enhance secure and situation awareness of AIoT. To start with, we designed the overall system architecture as shown in Figure 1. Our design adds three new parts to the typical Industrial Internet of Things (IIoT) architecture. The details are as follows.

1) AIoT Security Control Center (ASCC): The main functions of the ASCC are Honeynet control, security threat monitoring, and situational awareness. The ASCC includes three nodes as described below.

a) Honeynet Control Master (HCM): The HCM is mainly responsible for managing and interacting with HoneyNet nodes: The honeypot is composed of independent devices integrated into a complete HoneyNet system. The HCM is responsible for the environmental configuration of the HoneyNet, deploying the honeypot, launching or terminating the honeypot, and uploading logs.

b) Honeypot Image Warehouse (HIW): The HIM is mainly responsible for storing and managing honeypot images.

c) Monitor Server (MS): The MS is mainly responsible for two aspects: communicating with control servers in the enterprise, control, and field networks in AIoT and configuring and monitoring these control servers to detect threats and situational awareness in a timely manner. The other aspect involves threat detection and situational awareness. The server uses a trained deep learning model from the model training server in the DC to detect threats on actual data.

2) HoneyNet: The main function of HoneyNet is to discover a variety of malicious programs and malicious behaviors in AIoT. HoneyNet

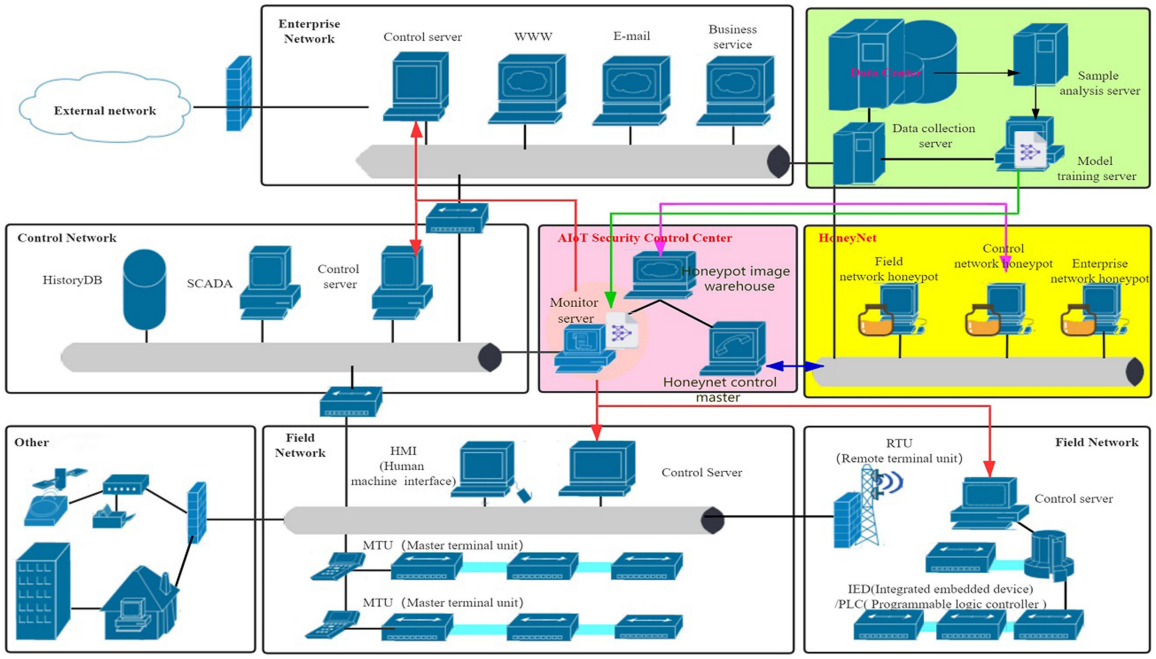

Figure 1. Overall architecture.

includes the three types of honeypots described below.

a) Enterprise Network Honeypot (ENH): The ENH is mainly responsible for finding and capturing malicious code and malicious behaviors in the enterprise network.

b) Control Network Honeypot (CNH): The CNH is mainly responsible for discovering and capturing malicious code, malicious behaviors, and malicious instructions in the control network.

c) Field Network Honeypot (FNH): The FNH is mainly responsible for discovering and capturing malicious behavior and malicious instructions in the field network.

3) Data Center (DC): The main function of the DC is to collect data, including HoneyNet malicious code data, malicious behavior data, and malicious instruction data, and then preprocess these data to form training samples. The resulting samples are used to train a deep learning CNN model. The DC includes two main nodes as described below.

a) Data Collection Server (DCS): The DCS is mainly responsible for collecting data and backing up source logs deleted by the honeypot. The collected data are stored on a Hadoop Distributed File System.

b) Sample Analysis Server (SAS): The SAS is mainly responsible for summarizing and merging the logs collected by various honeypots. It extracts important fields such as ports, numbers, attacker IP addresses, attack paths, requested content, vulnerability fingerprints, and malicious sample server IP addresses, and for extracting data features.

c) Model training server (MTS): MTS is mainly responsible for training modules that we use CNN for deep learning.

In Figure 1, the red line indicates the information interaction between the MS and the enterprise network, the control network, and the field network. The MS needs to issue the monitoring commands to the control servers at all levels. Simultaneously, it also needs to report statuses in the enterprise network, the control network, and the field network to the MS. The green line represents the information interaction between the MTS and the MS. Here, the MS needs to download the trained deep learning model parameters from the MTS. The purple line

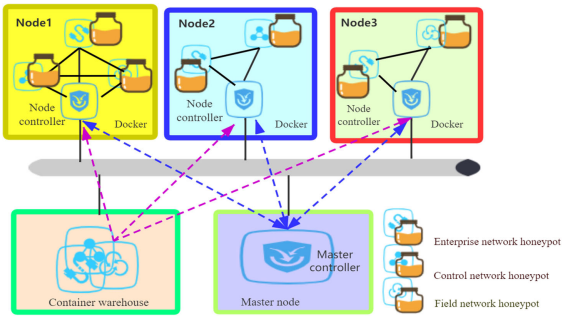

Figure 2. Honeynet design and deployment diagram.

denotes the information exchange between HIW and the Honeypot, and that Honeypot needs to download the mirror image from HIW. The blue line denotes the control information interaction between the HCM and the honeypot. The HCM issues the control instructions to the honeypot, and the honeypot reports the sample collection status to the HCM.

HoneyNet Design

In AIoT, we designed and deployed honeynets to detect threats. The HoneyNet is an active security defense system composed of multiple honeypots and business hosts in the IIoT. It can compensate for the poor defensive effect of a single honeypot facing complex and changeable network attack methodologies and multiple attack targets in the attack-defense game. The HoneyNet possesses strong interaction and collaboration characteristics.

The detailed Honeynet design and deployment diagram is shown in Figure 2. We deploy honeypots to multiple physical nodes, and each physical node deploys one ENH honeypot, one CNH honeypot, and one FNH honeypot. Also, we deploy the node controllers on the each physical nodes so that the three honeypots and the node controllers form a small honeynet. To coordinate the honeynet on each physical node, we deploy a master physical node. The master physical node has a master controller, which is responsible for communicating with each physical node controller and can distribute files and commands, realize tasks such as honeypot opening and closing, node monitoring, obtain honeypot response messages, and record them in the output log, etc. As well, each physical node connects to a remote container warehouse to download and run the required honeypot images. Compared with traditional honeypots, the honeypots in our scheme are based on Docker technology, which greatly reduces the deployment costs and offer better security, isolation, flexibility, and scalability. Moreover, the honeypots can form a dynamic HoneyNet system with data control, data acquisition, and data analysis functions. Using the Docker container configuration file and shell scripts, it is easy to set up the IP address of the honeypot and open the corresponding port to build the normal HoneyNet service.

Data Collection and Preprocessing

AIoT malicious samples refer to program files or behaviors that can infect IoT devices and carry out attacks such as distribution denial of service (DDoS), mining, or steal device information. First, we use the honeynet designed above to capture malicious programs and behaviors in a Sichuan power grid. We obtained 7000 malicious instances and 5500 normal instances for comparison. Second, considering the development history, code characteristics, and the number of samples actually captured, we divide the IIoT malicious samples into nine representative family categories, namely Rbot, ExploitKit, Stuxnet, Qbot, Xor and Mirai, LuaBot, Elknot, GoScanSSH, and Benign. When we classify these samples into families, part of them is done manually by reverse engineering and part of them is scanned by vulnerability test. The judgment results of each engine are integrated for marking. Finally, to make the samples trainable, we use Radare2, which is an open source reverse engineering tool, to extract information from malicious samples, including 38 different attributes, such as file format, file type, system bitness, and so on. For improving the recognition rate of the model for cyberattacks, we want to use the CNN model to automatically extract features for training. We believe that it is necessary to convert sample data into images, the specific reasons are as follows: First, the data we obtained through reverse engineering have noise in the process of converting to an image, we can remove the noise; second, it is more suitable to use images as samples for CNN model training.

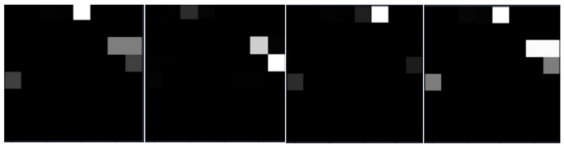

Figure 3. Data grayscale.

The different features of the sample data include numerical data and character data. For numerical data, we need to normalize the data to convert it into the same dimension for CNN model training; for character data, we first perform numeralization and then use one-hot encoding to convert it into a 2-D matrix for the CNN model training. The specific process is as follows.

1) Discard extracted data with null attribute values.
2) Normalize continuity attributes such as file size first; then, map the normalized value in the interval to the interval number to achieve discretization, and finally, use one hot encoding to encode the interval number.
3) Perform one-hot encoding of the discrete values in the data based on type.

After the above processing, the experimental data format is converted into a binary number. Then, each 8 bits is converted into a gray value. These gray values are converted into grayscale images and the blank positions are filled with zeros. Each grayscale image represents a row of data, and these rows are resized into 28×28 grayscale images, as shown in Figure 3. It is a grayscale image of a 2-D matrix used for neural network training, which was obtained by converting the malicious samples of the IoT during the preprocessing phase. The grayscale image of the data represents the preprocessing result. The grayscale distributions of samples can verify the usability of our data after preprocessing.

Improved LeNet5 for Malicious Sample Classification Detection

Our scheme adopts convolutional neural networks (CNNs), which usually consist of an input layer, several convolutional layers, a pooling layer, an activation layer, a fully connected layer, and an output layer. The convolutional layer is composed of many convolutional units, whose convolution operations extract relatively low-level features. The pooling layer is down-sampled, and the training parameters are generally reduced by halving the feature map. The fully connected layer maps the features learned by the preceding convolutional layer and pooling layer to the sample label space, and each neural unit of its input data is connected with the output neural unit. The reason we chose a CNN is to capitalize on its shared convolution kernel to automatically extract features for training the classification model.

To improve the efficiency of detecting and classifying unknown samples, we analyze the classification task. Because this classification task has inputs similar to those in traditional image recognition, we improve on the LeNet5 network based on CNNs and propose a new deep learning model that is more suitable for detecting attacks on unknown samples. The original LeNet5 network has a relatively shallow number of layers; however, the characteristics of network attack data are more complex. Thus, it is not particularly effective to use this shallow network to extract features for classification. We need to increase the number of network layers to extract more complex malicious samples. However, related experiments show that simply increasing the depth of the network not only makes it prone to overfitting but also increases the system overhead required for training. Based on the above considerations, we borrowed the Inception module structure proposed in GoogLeNet and added it to the original network structure along with batch normalization and dropout layers to prevent overfitting. A diagram of the overall network structure is shown in Figure 4.

Threats Detection and Situational Awareness

The detailed process of threats detection and situational awareness is as below. First, the MS obtains a trained deep learning model from the DC. Second, the MS monitors the control servers for the enterprise, control, and field networks. When suspicious service access occurs, the control servers immediately collect the access data and sends it back to the MS. After preprocessing the data, the MS uses the deep learning model to

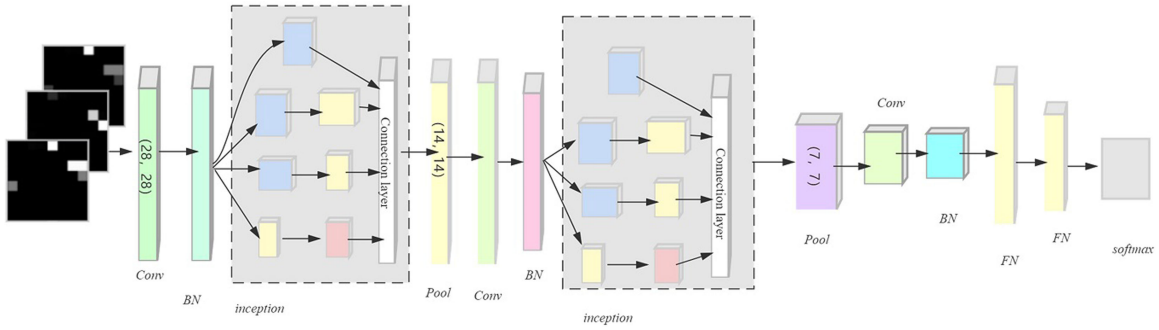

Figure 4. Network structure.

obtain a special classification for the suspicious activity. Third, the MS performs situation assessment based on the specific classifications of prior suspicious visits; it conducts different situation assessments for different threats.

The situation assessment system performs a strategy-based threat rating and fusion on the different states of the four threats, including threat path, threat port, attack fingerprint, and time series to give an overall threat index for AIoT.

First, we have made statistics on the threat path of AIoT. For example, files ending in xml such as $sdesc.xml$ are device configuration files, and it is speculated that the attacker's intention is to steal device information; $/api.php$ and other files ending in php are web page files, and it is speculated that the attacker's intention is ordinary scanning. Domain information such as $dnspod.qcloud.com:443$ is generated by attackers using open ports for proxy access. $/kindeditor/php/upload\_json.php$ and other words with $upload\_json$ represent the attacker's intention to inject web vulnerabilities; $/UD/,/$ and other paths are the resource paths of IoT devices; $/ctrlt/DeviceUpgrade\_1$ is the injection path of firmware SOAP service vulnerabilities; $/ctrlt/DeviceUpgrade\_1$ is the injection path of firmware SOAP service vulnerabilities. Combine attack intention analysis to establish path strategy rating. CVE device vulnerability injection is level 5; WEB vulnerability injection is level 4; device file scanning is level 3; front-end page scanning is level 2; proxy utilization is level 1; other paths are level 0. When scoring fusion, all the paths that appear in the log on the day are analyzed and rated according to the rating strategy, and all the scores are added to obtain the path threat value, which is capped at 30.

Then, we make statistics on the threat port of AIoT. Port 5555 is related to the PHP page and access times are 656; port 52869 is related to picsdesc.xml configuration file access and access times are 597; port 37 215 is related to $/ctrlt/DeviceUpgrade\_1$ injection, and access times are 379. Based on this, the port strategy rating is established: port 37 215 is 4; port 52 869 is 3, port 5555 is 2, the remaining ports are 1, and the commonly used service ports (80, 8080, 443, etc.) are rated as 0. In the scoring integration, the threat port value is obtained by adding up all the port rating values that appear in the log of the day, and the value is capped at 20.

Third, we make statistics on the fingerprints of attacks on AIoT. Attack fingerprint refers to whether specific vulnerability information or attack method information is included in the malicious request content. For example, for IoT firmware injection, there will be a $u:Upgrade$ string in the body field of the request packet; for port mapping attacks used for intranet penetration, the request packet will have the $NewInternalPort$ field; for general botnet attacks, the request packet will generally be $wget$ and $curl$ fields. We have established the AIoT attack fingerprint strategy rating, which is level 4 for malicious family fingerprints; level 3 for download command fingerprints; level 3 for device upgrade fingerprints; level 3 for new port fingerprints; level 0 for no fingerprints. When performing scoring fusion, the body field of all records appearing in the log of the day is parsed, and all the scores are added to obtain the attack fingerprint threat value, which is capped at 40.

Finally, we make statistics on the time series of the AIoT. The so-called time series is to

analyze the sudden changes in the number of requests over a period of time. Specifically, if the number of requests on a certain day exceeds the average of the previous week by a certain percentage, then there is reason to believe that a large-scale attack has broken out on the network. The specific rating strategy is as below. Let us set the rate as 10, 8, 5, 3, and 0 if the traffic mutation ratio is "big than 10," "5–10," "3–5," "1.5–3," and "less than 1.5," respectively. This value is capped at 10.

To evaluate the current network status through the above four threats, we adopt the following security strategies:

1) No matter which threat exceeds the peak index, the early warning threshold should be set. For example, the value of the threat path exceeds 30, the early warning must be issued immediately.

2) Set the comprehensive situation assessment index as the sum of the above four threats (threat path value, threat port value, attack fingerprint value, and time series value) to achieve network security status alert. The early warning threshold is also set empirically, usually 60. If it exceeds the threshold, then an immediate warning must also be issued.

EXPERIMENTAL RESULTS AND DISCUSSION

Experimental Environment

The experimental platform hardware configuration consisted of an Intel i7-6700 CPU with a GTX1080Ti graphics card and 32 GB of memory running on a Windows 10 operating system. Our model was implemented in the Python programming language and executed on the TensorFlow framework. In this study, the malicious samples collected with the designed HoneyNet structure were labeled into nine categories, namely, Rbot, ExploitKit, Stuxnet, Qbot, Xor, Mirai, LuaBot, Elknot, and GoScanSSH. Then, we selected 5500 normal samples for comparison purposes and divided all the data into training and test sets at a ratio of $7:3$. Finally, we used these datasets to train and verify the proposed model.

Environmental experiment of honeynets installed with different honeypots on five PCs and three Raspberry PIs. The detailed design is shown in the section "HoneyNet Design." For the designed honeynet, the PC operating system is Ubuntu18.04, and the Raspberry PI use the Raspberry PI 3B+ operating system. We use Docker17.03 to deploy the T-pot open source image to realize the work of training log extraction and attack sample collection. We used the SiteWhere2.0 environment to simulate the AIoT, which is an open source IoT platform that supports device data ingestion, storage, processing, and integration and runs on the core server provided by Apache Tomcat. In this study, we deployed SiteWhere2.0 on Ubuntu 8.04.

Experimental Procedures and Evaluation Indicators

To evaluate the model's performance and comprehensively measure the classification accuracy of each malicious sample, we selected the following metrics: macro average precision rate, recall rate, and F1-score. The macro average precision rate represents the proportion of the predicted positive category among all predicted positive categories; the recall rate represents the proportion of correct true positive predictions among all the true positive categories; and the F1-score formula is the ratio of the arithmetic mean of the two to the geometric mean, which is a comprehensive performance indicator that considers both precision and recall; a larger F1-score indicates a better model performance. Moreover, the "accuracy" is also utilized with precision, recall, and F1 to verify the results when the model is balanced. Accuracy refers to the number of correct predictions over the total number of predictions.

Analysis of Experimental Results

In the classification of malicious samples, Bayes, *k*-nearest neighbor (KNN), back propagation neural network (BPNN) and our proposed method are used to train the model on the data of the collection system and then evaluated with the test set. Bayes is an algorithm based on Bayes' formula to find the posterior probability when the prior probability is known; KNN is a method of classifying each record in a dataset. The so-called KNN means that each sample can be represented by its KNNs. BPNN is a back-propagation neural network that is based on the forward propagation neural network. The cost

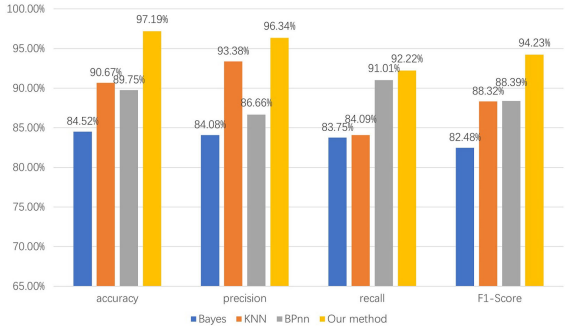

Figure 5. Comparison of different indicators.

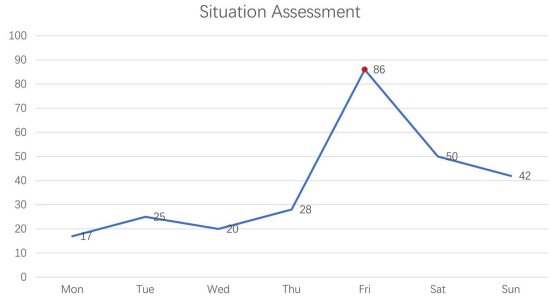

Figure 6. Situation assessment score.

function is obtained, and then backpropagation is performed by calculating the gradient. Finally, the weight of each neuron in the network is optimized. The specific values of each indicator are shown in Figure 5. From the analysis of the above figure, compared with the traditional machine learning method, the proposed method has a good improvement in the accuracy, precision, recall rate, and F1 value, which can reach 97.19%, 96.34%, 92.22%, and 94.23%, respectively, which can prove the validity of our proposal.

For situation assessment, the data collected by the SiteWhere platform classify unknown samples through models and understand the current IoT situation to obtain the specific situation of malicious samples for situation assessment. Figure 6 shows the situation assessment of the platform in a week. It can be seen that on Friday the situation assessment score of 86 is higher than the set threshold value of 60 (based on experience), and the system will alarm. We can find that our scheme can effectively monitor the state of AIoT, which effectively proves the feasibility of the scheme.

CONCLUSION

With increasingly complex network security issues, it is necessary to develop security measures for AIoT. Hence, in this article, we have proposed honeynet-based threat detection and situational awareness for secure and resilient AIoT. In this scheme, we designed a Docker-based honeynet to collect a certain amount of attack log data. Then, a new judgment model was constructed by the deep learning method. Finally, the stable model has been deployed in AIoT to realize threat detection and situational awareness. Through experimental evaluation, our solution conforms to the AIoT environment. Moreover, the scheme has certain scalability. In future work, we have plan to work on completing the real-time update of the model and the design of a larger honeynet to enhance the security and resilience of the AIoT for consumers, commercials, and industry.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61373162, in part by the Sichuan Provincial Science and Technology Department Project under Grant 2019YFG0183, in part by the Sichuan Provincial Key Laboratory Project under Grant KJ201402, and in part by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) under Grant JP18K18044 and Grant JP21K17736.

REFERENCES

1. K. Yu, Z. Guo, Y. Shen, W. Wang, J. C.-W. Lin, and T. Sato, "Secure artificial intelligence of things for implicit group recommendations," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2021.3079574.
2. J. Zhang, K. Yu, Z. Wen, X. Qi, and A. K. Paul, "3D reconstruction for motion blurred images using deep learning-based intelligent systems," *Comput., Mater. Continua*, vol. 66, no. 2, pp. 2087–2104, 2021.
3. L. Zhen, A. K. Bashir, K. Yu, Y. D. Al-Otaibi, C. H. Foh, and P. Xiao, "Energy-efficient random access for leo satellite-assisted 6 g internet of remote things," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5114–5128, Apr. 2021.

4. C. Feng *et al.*, "Efficient and secure data sharing for 5G flying drones: A blockchain-enabled approach," *IEEE Netw.*, vol. 35, no. 1, pp. 130–137, Jan./Feb. 2021.
5. L. Tan, H. Xiao, K. Yu, M. Aloqaily, and Y. Jararweh, "A blockchain-empowered crowdsourcing system for 5G-enabled smart cities," *Comput. Standards Interfaces*, vol. 76, 2021, Art. no. 103517.
6. J. Pan and H. Li, "Intrusion detection method for Internet of Things based on practical Byzantine fault tolerance" *Comput. Applications*, vol. 39, no. 6, pp. 1742–1746, 2019.
7. R. R. Reddy, B. Kavya, and Y. Ramadevi, "A survey on SVM classifiers for intrusion detection," *Int. J. Comput. Appl.*, vol. 98, no. 19, pp. 34–44, 2014.
8. Z. Fuqun, "Detection method of LSSVM network intrusion based on hybrid kernel function," *Modern Electron. Techn.*, vol. 21, 2015, Art. no. 027.
9. S. Sahu and B. M. Mehtre, "Network intrusion detection system using j48 decision tree," in *Proc. Int. Conf. Adv. Comput., Commun. Inform.*, 2015, pp. 2023–2026.
10. Y. Zhang, P. Li, and X. Wang, "Intrusion detection for IoT based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31711–31722, 2019.
11. M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proc. Nat. Aerosp. Electron. Conf.*, 2015, pp. 339–344.
12. B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," in *Proc. 7th Conf. Inf. Knowl. Technol.*, 2015, pp. 1–5.
13. C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
14. M. Akter, G. D. Dip, M. S. Mira, M. A. Hamid, and M. Mridha, "Construing attacks of Internet of Things (IoT) and a prehensile intrusion detection system for anomaly detection using deep learning approach," in *Proc. Int. Conf. Innov. Comput. Commun.*, 2020, pp. 427–438.
15. H. Li, K. Yu, B. Liu, C. Feng, Z. Qin, and G. Srivastava, "An efficient ciphertext-policy weighted attribute-based encryption for the Internet of health things," *IEEE J. Biomed. Health Inform.*, to be published, doi: 10.1109/JBHI.2021.3075995.

Liang Tan is currently a Professor with the College of Computer Science, Sichuan Normal University, Chengdu, China, and with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His research interests include cloud computing, big data, trusted computing, and network security. Contact him at jkxy_tl@sicnu.edu.cn.

Keping Yu is currently a Researcher (Assistant Professor) with Global Information and Telecommunication Institute, Waseda University, Japan. He is the corresponding author of this article. Contact him at keping.yu@aoni.waseda.jp.

Fangpeng Ming received the bachelor's degree in engineering from Sichuan Normal University, Chengdu, China, in 2019, where he is currently working toward the master's degree in computer application technology. Contact him at frank.ming@foxmail.com.

Xiaofan Cheng received the bachelor's degree in engineering from Sichuan Normal University, Chengdu, China, in 2019, where he is currently working toward the master's degree in computer application technology. Contact him at sail967642@gmail.com.

Gautam Srivastava has been in a tenure-track position with Brandon University, Brandon, MB, Canada, in 2014. He received the B.Sc. degree from Briar Cliff University, Sioux City, IA, USA, in 2004, and the M.Sc. and Ph.D. degrees from the University of Victoria, Victoria, BC, Canada, in 2006 and 2011, respectively. Contact him at srivastavag@brandonu.ca.