Hindawi
Security and Communication Networks
Volume 2019, Article ID 5695021, 10 pages
https://doi.org/10.1155/2019/5695021

Research Article

Integrating Traffics with Network Device Logs for Anomaly Detection

Jiazhong Lu,[1] Fengmao Lv,[2] Zhongliu Zhuo,[1] Xiaosong Zhang,[1] Xiaolei Liu,[1] Teng Hu,[1] and Wei Deng[2]

[1]*Center for Cyber Security, University of Electronic Science and Technology of China, Chengdu 611731, China*
[2]*School of Statistics, Southwestern University of Finance and Economics, Chengdu 611130, China*

Correspondence should be addressed to Fengmao Lv; fengmaolv@126.com and Xiaosong Zhang; cdgbsjfzx@126.com

Received 25 December 2018; Revised 30 April 2019; Accepted 15 May 2019; Published 13 June 2019

Guest Editor: Pelin Angin

Advanced cyberattacks are often featured by multiple types, layers, and stages, with the goal of cheating the monitors. Existing anomaly detection systems usually search logs or traffics alone for evidence of attacks but ignore further analysis about attack processes. For instance, the traffic detection methods can only detect the attack flows roughly but fail to reconstruct the attack event process and reveal the current network node status. As a result, they cannot fully model the complex multistage attack. To address these problems, we present Traffic-Log Combined Detection (TLCD), which is a multistage intrusion analysis system. Inspired by multiplatform intrusion detection techniques, we integrate traffics with network device logs through association rules. TLCD correlates log data with traffic characteristics to reflect the attack process and construct a federated detection platform. Specifically, TLCD can discover the process steps of a cyberattack attack, reflect the current network status, and reveal the behaviors of normal users. Our experimental results over different cyberattacks demonstrate that TLCD works well with high accuracy and low false positive rate.

1. Introduction

Cyberattacks usually leave footprints on network devices. Typically, an attacker's attack path jumps through multiple routers or servers and then uploads malicious code (e.g., XSS script), implants virus (e.g., botnet), and submits Trojaned software or unofficial patch containing malicious payloads [1–7]. Generally, the footprints left by cyberattacks are spatiotemporally dispersed across logs of different victims' machines [8]. For instance, XSS script attack may leave evidence in server's weblog. However, as the logs are dispersed across diverse disconnected sources, piecing together the contextual information of each malicious footprint still needs human involvement. Therefore, directly leveraging the logs for anomaly detection is ineffective. On the other hand, network traffic can also provide complementary evidence for attack-related activities, such as anomalous data about connections from IRC/HTTP/DNS servers to botnet. However, it is insufficient to precisely detect attack behaviors and grasp a complete view of attacks with only the network traffic data.

To date, most existing log-based or traffic-based intrusion detection methods have the following limitations: (1) They only focus on a single or a few logs, lacking context information (especially the contacts in internal network). (2) The traffic characteristics are not diverse enough to achieve good detection performance. (3) Both the log-systems and the traffic-systems heavily rely on hefty equipment, which incurs very large cost overhead [9]. (4) The false positives and false negatives are not satisfactory in realistic detection process [10]. In this paper, we propose to integrate traffics with network device logs for detecting cyberattacks. Specifically, we collect logs and traffics from switch, router, firewall, and servers. Then we use fuzzy association rules to integrate the device logs with traffics to reconstruct the cyberattack.

Overall, the main contributions of this paper are listed as follows: (1) We propose a novel combined detection method to reconstruct the attack process. (2) Our method

can effectively integrate multiple network device logs with traffics by leveraging fuzzy association rules. (3) We conduct extensive evaluation of TLCD over diverse cyberattacks (e.g., phishing, XSS, and botnet). The experimental results clearly demonstrate the effectiveness of our method.

2. Related Work

In general, network intrusion detection mainly includes signature-based detection and anomaly-based detection [11, 12]. Specifically, signature-based detection relies on existing signature databases to detect malware infections. By using signature-based detection methods, malwares can be effectively identified through pattern matching. However, signature-based detection techniques have the fatal disadvantage that a new malware infection cannot be detected if its signature is not contained in the signature database.

Anomaly detection is a technique for detecting abnormal behaviors that deviate from normal behaviors [13, 14]. Specifically, it aims to detect events in the monitored domain different from the pattern defined by normal behaviors [13, 15]. Basically, the normal behavior of the network needs to be identified first. Compared to the signature-based detection, the main advantage of anomaly-based intrusion detection is the ability to detect new or unknown attacks, since abnormal behaviors can also occur when the signature of new malware is not available. However, due to the complexity of the behaviors from different networks and applications, it is difficult to accurately identify the normal behaviors. The existing anomaly detection methods are usually based on device logs or traffic flows alone [16]. In general, their methods are too simple to achieve satisfactory results [17]. Additionally, they fail to effectively reconstruct the attack conditions [18]. On the contrast, our detection method is multinetwork device interrelated and verified and can further improve the accuracy and reflect the state of the network environment at the time.

3. Method

This work mainly proposes to implement anomaly detection through integrating multiple network device logs with traffics. Specifically, the device logs and traffics are integrated through association rules. Our method can effectively improve the detection performance and reconstruct the network attack process, which enables us to grasp a complete view of the entire network environment.

3.1. Method Overview. Due to the diversity of network attacks, the network environment is quite complex. For instance, botnets must first send control commands to each C&C server and then to the controlled-host, while worm needs first to upload malware code to target-host and then infect others computers by the target-host. Therefore, the network device logs and traffic flows can play very important roles in cyberattack detection. To collect our data for anomaly detection, we first obtain the traffic flows with port mirroring and adopt TCPDUMP to extract useful traffic attributes (such as the source port, destination port, protocol number, source IP, destination IP, the number of packet size, and send time). Additionally, we also extract log information (such as Data, Time, Module, Level, PID, Type, Action, Application, Reason, etc.) from the gateway's internal routers, switches, firewalls, and servers. After that, we attempt to extract the mapping relations between logs and traffics with association rules. Finally, the extracted relationships can be used to generate the time stamps of log records and reconstruct the attack process.

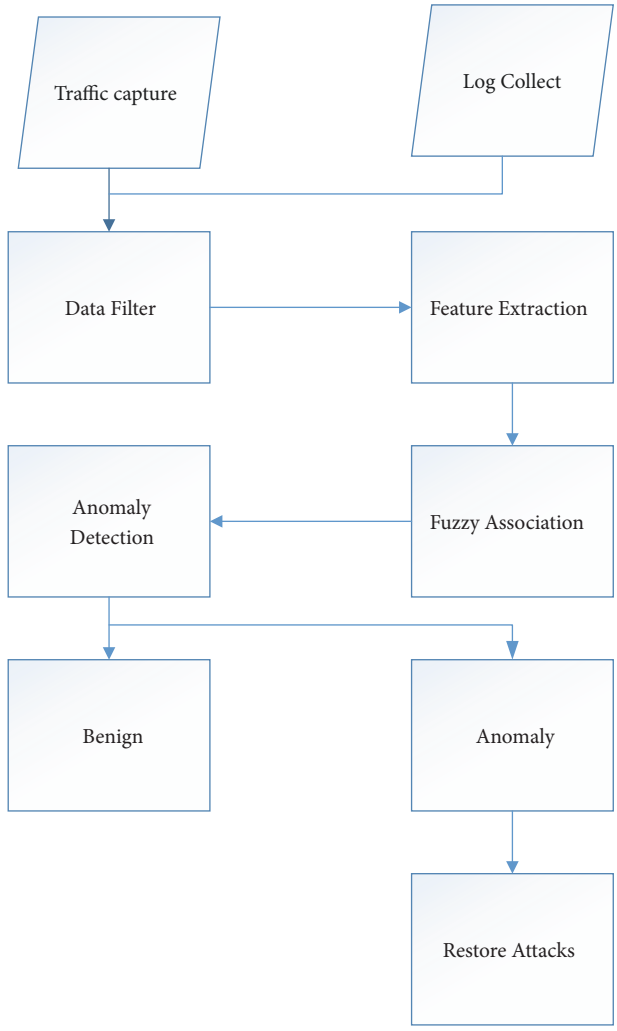

Figure 1: The overview of TLCD.

In Figure 1, we display the overview of TLCD. The locations of the traffic captures and log collectors are shown in Figure 2. Traffic capture modules are placed at the university servers and enterprise servers, which are connected outside the Internet. In this way, we can capture both the inbound and outbound data in real time. The inputs to TLCD are multiple raw data from network devices (e.g., router, switch, firewall, and servers). The detection is specifically designed to be format agnostic for both the traffics and logs. Through a parser plugin, TLCD can handle any input format of traffics and logs. As the detection task is usually featured by large-scale data, filtering is necessary for reducing the detection cost and time

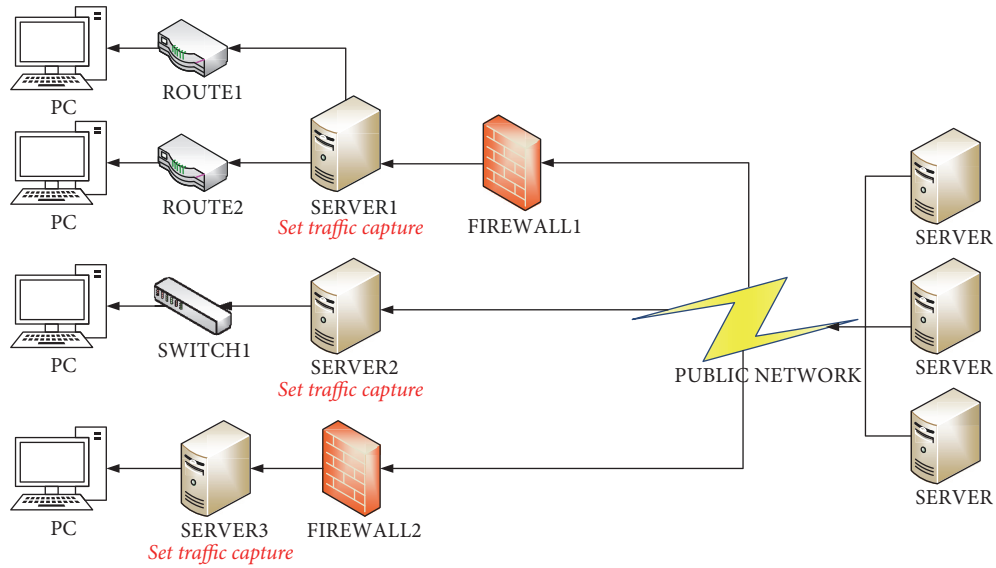

Figure 2: The deployment of the traffic captures and log collectors (server-1, firewall-1, router-1, and router-2 are deployed in enterprise. Servers-2, switch-1, firewall-1, and servers-3 are deployed in campus).

overhead. Therefore, we will filter out the irrelevant data in both logs and traffics. In the feature extraction module, we extract 63 traffic features (including 5 new TCP flags) and 16 log features. The log features are mainly used to reveal what happened before and after the cyberattack and auxiliarily detect the anomaly behaviors. The fuzzy association module aims at modeling strong associations between traffics and logs. After obtaining the confidence intervals of candidate rules, the ones among high confidence intervals are used to construct strong association rules. Note that there are many kinds of mappings here. It is possible that multiple traffics relate to one single log or one traffic relates to several logs.

To present the mechanisms of TLCD, we list the details as follows:

(1) Phase I, reprocessing: the traffic captures and log collectors first collect the original traffics and logs, which are considered as the inputs to TLCD. Then, these traffic-log inputs are reprocessed via the parser plugin and the filter module. Specifically, the filter module filters out the irrelevant data in both logs and traffics.

(2) Phase II, feature extraction: the feature extraction module consists of five components, including the traffic correlation (to analyze the traffic packets for malware), temporal correlation (to obtain the time characteristics of malware), combination correlation (to model the strong relevance of malware traffic), TCP flag (to record the sending and responding of traffic data), and log-information (to record the log information for attack reconstruction) components.

(3) Phase III, fuzzy association module: this module aims to integrate the traffics with logs through association rules.

Table 1: Details of the TCP flags.

TCP flags	TCP handshake situation
	ACK, URG, FIN, RST values
	The destination IP repeatedly responds with ACK = 1
	The destination IP only has ACK = 1, SYN = 1 and FIN=1
	The source IP only has SYN = 1

(4) Phase IV, anomaly detection module: advanced machine learning techniques are adopted to recognize malicious data as anomalies.

(5) Phase V, attack reconstruction module: the reconstruction module leverages the associations between the logs and traffics to generate the time stamps of log records, which can be finally used to reconstruct the attack process.

3.2. Feature Extraction

3.2.1. Network Traffics. To collect the traffic-log data, we have monitored the university-enterprise network for one month. The features used in our framework include temporal correlation features [19], TCP flag features (displayed in Table 1), and log features (displayed in Table 2). Usually, some cyberattacks, such as botnets and phishing emails, need to automatically send commands through programs. These automatic attack commands, more or less, contain inherent patterns. Specifically, we capture the traffics from 4 common cyberattacks (XSS, HTTP botnet, P2P botnet, and phishing) and find that different network attacks behave differently in the TCP handshake stage. For instance, phishing mail transmission process adopts POP3 and IMP4 protocol, allowing attackers to send different types of files. Also, the

Table 2: Detailsof the network device logs.

	Firewall logs	Traffic logs	Event logs	Network logs	Security logs	System logs	Cron logs	Mail logs	Messages logs	Mysqld logs
Data	*	*	*	*	*	*	*	*	*	*
Time	*	*	*	*	*	*	*	*	*	*
Module	*		*			*				
Level	*		*		*		*	*	*	*
PID			*			*	*	*	*	*
Type	*			*		*	*	*	*	*
Action		*					*	*	*	*
Source		*								
Destination		*								
Translated Source		*								
Translated Destination		*								
Duration		*								
Bytes Sent		*								
Bytes Received		*								
Application		*				*				
Reason	*	*	*	*	*	*	*	*	*	*

corresponding data volume can be very small. Therefore, the phishing mails can result in the same TCP handshake states to the normal ones, and it can be kinda easy to establish a connection. However, in a botnet, an attacker needs to control the C&C server and send a large number of commands, which will inevitably cause a handshake failure in TCP handshake. In Table 1, we display the details of TCP flags. Specifically, SYN denotes whether a connection is established, FIN and ACK denote the corresponding responses, and RST denotes the connection reset. Note that the ACK information can be used together with SYN and FIN as evidences for attack detection. For instance, if both SYN and ACK are activated, it means that the connection is established with confirmation. On the contrast, if only SYN is activated, we can conclude that that the connection is established without confirmation. Usually, most of the unreachable attacks can only activate SYN. Additionally, for the situation with FIN and RST activated and SYN unactivated, the firewalls may still detect the SYN/FIN packet. When such a packet appears in the situation, it is most likely that the network has been attacked. As the ACK/FIN packet represents a completed TCP connection, a normal FIN packet is always marked by ACK. A "NULL" packet is the packet not marked by any TCP flags (URG, ACK, PSH, RST, SYN, and FIN are all set to 0). For normal network activities, the TCP stack can never generate packets featured by unreasonable TCP flag combinations; otherwise the networks have been attacked. Therefore, the TCP flag features can provide useful information about the network status [19].

3.2.2. Network Device Logs. In Table 2, we display the details of network device logs. As we can see, different types of network device logs have different characteristics. Specifically, the firewall logs record the events between the inside and outside the network, such as port filtering, hazard level, and authentication; the traffic logs record current traffic conditions, such as packet size, IP address, and duration; the event logs record events that occur during the execution of the system, in order to provide traces for activity monitor and problem diagnosis; the network logs record the process of network access, such as data packet request or uploading; the security logs mainly record the operations of network devices and the system errors; the system logs record the hardware and software errors, as well as events that occur in the monitoring system, allowing the user to check the cause of errors and find traces left by the attacker; the Cron logs record periodic tasks in Linux (Cron reads the configuration files and writes them in memory when Linux starts. As there exist some cyberattacks featured by cyclicality, Cron logs are effective for identifying this type of attack); the mail logs allow the administrators to get the copies of messages processed by the Domino system router (when the mail log is enabled, Domino will check the messages as they go through MAIL.BOX, and save their copies to MAILJRN.NSF for future recovery); the message logs are plain text files that will be first checked for error messages when a problem occurs; the MySQL logs contain information of log-err, query log, log-slow-queries, log-update, and log-bin. By default, all logs are created in the MySQL directory. In this work, we extract attributes from ten types of logs from different network devices. Each type of log has its own attributes. For instance, the firewall log has attributes of data, time, module, level, type, and reason, while the traffic logs have attributes of data, time, action, source, destination, translated source, translated destination, duration, bytes sent,

bytes received, application, and reason. Although different logs reflect different characteristics of the device status, the shared attributes, such as time, date, and reason, can be effectively used to infer the status of one event in different logs.

3.3. Anomaly Detection

3.3.1. Feature Integration through Association Rules. In daily networks, there are no direct correlations between the log data and the traffic data. However, they can be correlated through the shared attributes like time and date. Therefore, we need to model the mutual mappings between traffics and logs by effectively leveraging the correlated attributes. With that, we can obtain the classification boundaries of the log attributes based on the corresponding attributes of traffics.

The discretization of traffic features, which is useful for boundary division, plays an important role in detecting anomaly traffics. Specifically, we use the Fuzzy-C Means (FCM) algorithm to divide the traffic characteristics (including quantitative attributes and Boolean attributes) into several fuzzy sets. Note that the elements and nonelements of each fuzzy set can be mutually transformed, in order to achieve the goal of softening features. In the process of highly skewed data, FCM algorithm can effectively model the actual distribution of data and clearly reveal the boundary between normal data and anomalies.

In our method, we first extract 29 basic attributes of the traffics, including the five-tuple (source IP address, destination IP address, source port, destination port, and protocol number), the total number of uplink and downlink packets, the total number of uplink and downlink payload packets, the total amount of uplink and downlink load, flow duration, average load, the maximum load, the minimum load, average time interval between the uplink and downlink data packets, the minimum time interval, the maximum time interval, and so on. Then, we extract 16 basic attributes of the logs, including data, time, module, PID, type, action, source, destination, translated source and destination, duration, bytes sent and received, application, reason, and so on. We assume that each feature comes from a Gaussian distribution. Then according to the membership function of fuzzy recognition, we can determine the fuzzy numbers of the maximum fuzzy set. Denoting the center of the maximum fuzzy set as μ, the membership degree as r_i (i = 1, 2, 3, ..., n), and σ as the parameter, the Gaussian fuzzy expression can be represented as follows:

$$\mathrm{y} = \exp\left[-\frac{(x-\mu)^2}{\sigma^2}\right]. \tag{1}$$

To approximate the maximum membership degree, we design the objective function as

$$g(\sigma) = \sum_{i=1}^{n}\left\{\exp\left[\frac{-(x_i-\mu)^2}{\sigma^2}\right] - r_i\right\}^2. \tag{2}$$

The corresponding membership function is expressed as

$$A_{ij}\left(x_j\right) = \begin{cases} 0, & \left|x_j - \overline{x}_j\right| > 2s_j \\ 1 - \left(\dfrac{x_j - \overline{x}_j}{2s_j}\right)^2, & \left|x_j - \overline{x}_j\right| \le 2s_j, \end{cases} \tag{3}$$

where $\overline{x_j}$ is the center and $2s_j$ is the standard deviation σ. Finally, we can identify whether a sample is an anomaly based on the principle of the maximum membership.

3.3.2. Anomaly Detection. The key point in anomaly detection is to detect anomalies from benign data according to the extracted features. To achieve this, we adopt supervised learning methods, such as K-Nearest Neighbor (KNN), Support Vector Machine (SVM), neural networks, or decision trees, to design the detection module. Basically, supervised learning first needs to establish a training set and then train a classification model over the training set. For this anomaly detection task, our goal is to learn a classifier that can effectively detect out the anomalies. In this work, we adopt Gradient Boosting Decision Tree (GBDT), which is an advanced machine learning technique and models the data with an ensemble of decision trees. To finally evaluate the performance of our method, 10-fold cross validation is adopted in our work.

4. Experiments

4.1. Dataset. In our experiments, we evaluate our method over 4 types of network attack (XSS, HTTP botnet, P2P botnet, and phishing). These cyberattacks are carefully injected into the normal business and will not bring undesirable effects for other business. Both the traffic data and the log data are collected from university servers and enterprise servers. To obtain the traffic data, we have monitored the university-enterprise network for one month. Specifically, we simulate the P2P botnet and HTTP botnet attack according to the Contagio blog [21] and white paper [22], which provide guidance about how to make botnet evade intrusion detection techniques. To simulate the XSS attacks, we inject malicious code into the web pages of university servers. The simulated phishing emails are sent to both university servers and enterprise servers. Note that in our simulation, the anomaly traffics only account for 0.1% of the total traffic flows, which is close to the real situations. As displayed in Table 3, the collected data include 30 normal traffic datasets, 6 traffic datasets for XSS injection attacks, 5 traffic datasets for phishing emails, and 20 ones for botnets (13 P2P botnets and 7 HTTP botnets). On the other hand, as displayed in Table 4, the log data are collected from 1 switch, 2 routers, 2 firewalls, and 3 servers.

4.2. Experimental Results. To validate the performance of integrating traffics with logs for anomaly detection, we conduct comparison experiments through only leveraging the traffic data (or the log data). As displayed in Tables 5–8 and Figures 3–6, it is clear that neither traffics nor logs can independently achieve desirable results in detecting

Table 3: The collection details for traffic data.

Type	Traffic Amount	Name
Normal	30	N/A
XSS	6	N/A
Phishing	5	N/A
HTTP botnets[20]	7	Virut, Sogou
P2P botnets [21]	13	NSIS.ay, SMTP Spam, Zeus (C&C), UDP Storm, Zeus, Zero access, Weasel

Table 4: The collection details for log data.

Device name	Quantity	Brand
Switch	1	Huawei
Router	2	Cisco,Huawei
Firewall	2	Juniper
Server	3	Cisco

Table 5: The detection results over XSS attack.

XSS	FP	FN
10-fold KNN for traffics	8.2%	5.6%
10-fold SVM for traffics	8.6%	5.8%
10-fold KNN for logs	9.1%	9.9%
10-fold SVM for logs	9.0%	8.6%
10-fold SVM for logs-and-traffics	5.2%	6.3%
10-fold KNN for logs-and-traffics	6.2%	3.6%
TLCD (GBDT)	4.3%	2.5%

Table 6: The detection results over phishing email.

Phishing	FP	FN
10-fold KNN for traffics	7.1%	7.3%
10-fold SVM for traffics	6.5%	7.3%
10-fold KNN for logs	8.8%	8.3%
10-fold SVM for logs	7.9%	8.2%
10-fold SVM for logs-and-traffics	5.0%	6.0%
10-fold KNN for logs-and-traffics	5.5%	4.8%
TLCD (GBDT)	5.3%	4.9%

Table 7: The detection results over HTTP botnet.

Http Botnet	FP	FN
10-fold KNN for traffics	5.5%	4.8%
10-fold SVM for traffics	5.3%	5.0%
10-fold KNN for logs	6.3%	5.9%
10-fold SVM for logs	6.3%	5.8%
10-fold SVM for logs-and-traffics	3.6%	2.9%
10-fold KNN for logs-and-traffics	3.8%	2.7%
TLCD (GBDT)	2.5%	2.8%

cyberattacks (both the False Negative (FN) and False Positive (FP) values decrease significantly), which is consistent with [16]. On the contrast, when we integrate the traffic flows with network device logs, the detection performance can be significantly improved. Additionally, we also compare

Table 8: The detection results over P2P botnet.

P2P botnet	FP	FN
10-fold KNN for traffics	4.5%	4.6%
10-fold SVM for traffics	5.2%	5.0%
10-fold KNN for logs	6.4%	6.0%
10-fold SVM for logs	6.0%	5.9%
10-fold SVM for logs-and-traffics	2.9%	2.9%
10-fold KNN for logs-and-traffics	3.3%	2.9%
TLCD (GBDT)	2.8%	2.6%

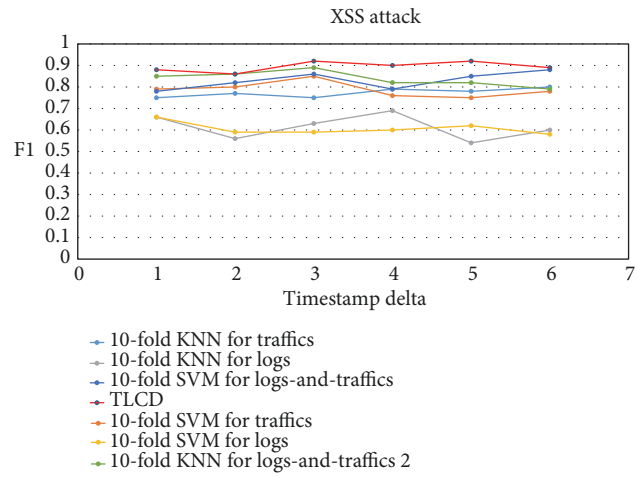

Figure 3: The F1 value obtained by each method over the XSS attack.

the detection performance of different supervised learning methods, including SVM, KNN, and GBDT. As we can see, these compared methods can achieve very similar results, with GDBT slightly better than the others. This effectively demonstrates that the features obtained through integrating traffics with logs are robust for our cyberdetection task.

4.3. Attack Reconstructions. In our experiments, we also evaluate the performance of TLCD on attack reconstruction. In particular, for these detected attacks, we first obtain their time horizon and communication address according to the information of the corresponding anomaly traffics, such as data, time, IP, and so on. With that, we can get the corresponding log features, and then the concrete network devices are determined. Finally, we reconstruct the original attack paths based on the abnormal information above.

Figures 7–10 display our attack reconstruction results for the four simulated cyberattacks. Generally, the XSS attack

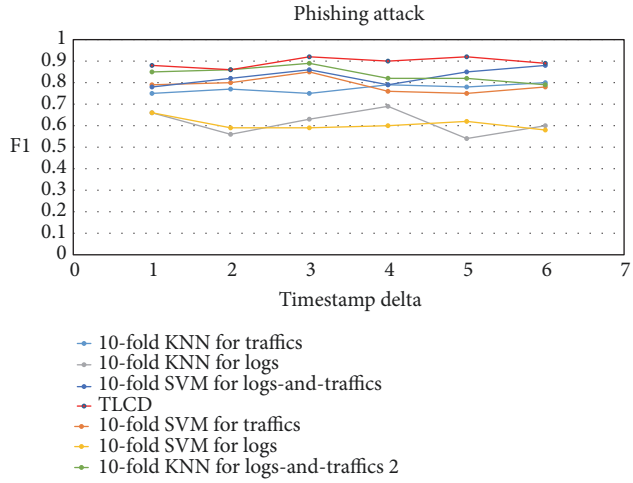

Figure 4: The F1 value obtained by each method over the phishing email.

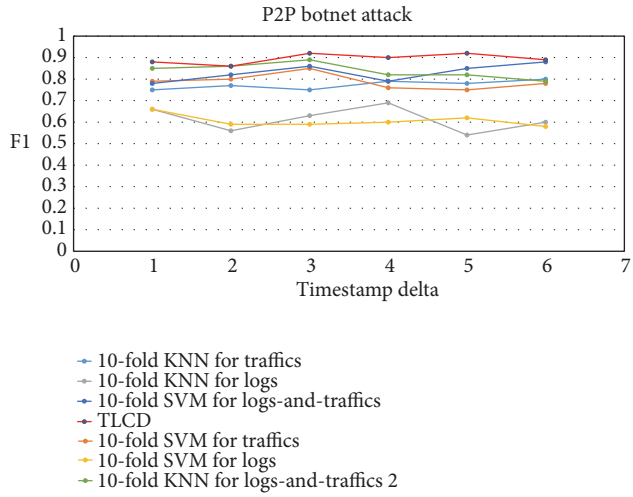

Figure 5: The F1 value obtained by each method over the P2P botnet.

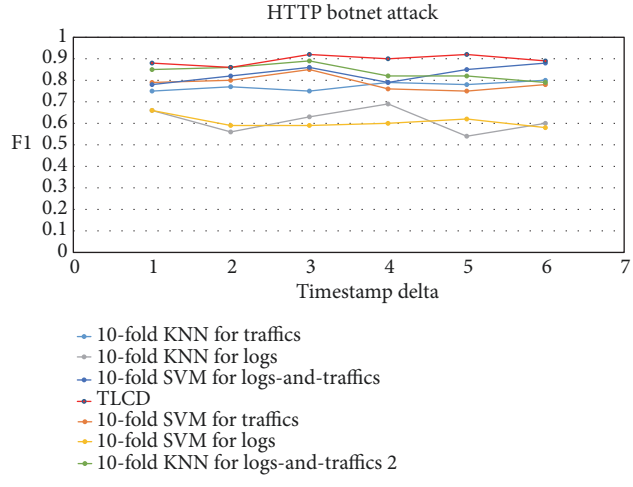

Figure 6: The F1 value obtained by each method over the HTTP botnet.

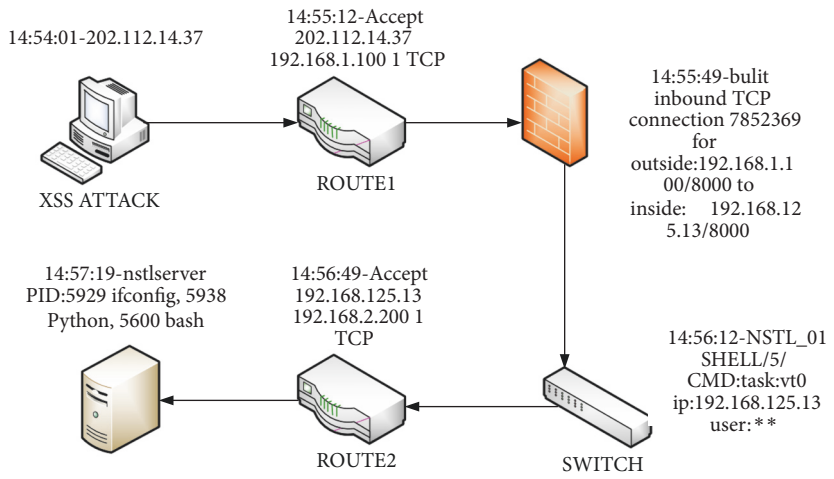

Figure 7: XSS attack reconstruction.

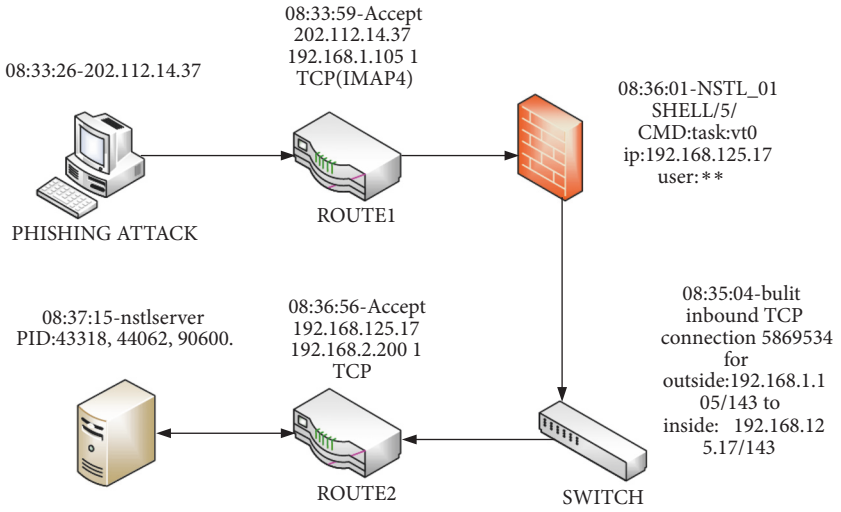

Figure 8: Phishing attack reconstruction.

damages the web server through passing several network devices, including routers, firewalls, switches, and so on. The phishing attack shares similar attack process to XSS, except that it adopts the IMP4 protocol and a fixed port. Different from XSS and phishing, the HTTP botnet does not aim to attack the servers, but the hosts through passing the servers. Note that the HTTP botnet attack is completed by web pages and the ICMP protocol. The P2P botnet attack is implemented not only through the servers, but also directly over the hosts. As displayed in Figures 7–10, our reconstructed results have accurately revealed the attack paths of the corresponding cyberattacks, which demonstrates that our method can effectively reconstruct the attack process.

5. Conclusion

In this paper, we propose to integrate traffics with network device logs for detecting cyberattacks. Specifically, we use fuzzy association rules to integrate the device logs with traffics to obtain the features for attack detection and reconstruction. The experiments over four common network attacks clearly demonstrate that our TLCD method can effectively detect diverse cyberattacks and reconstruct their event process.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by National Natural Science Foundation of China under grant no. 6157211.

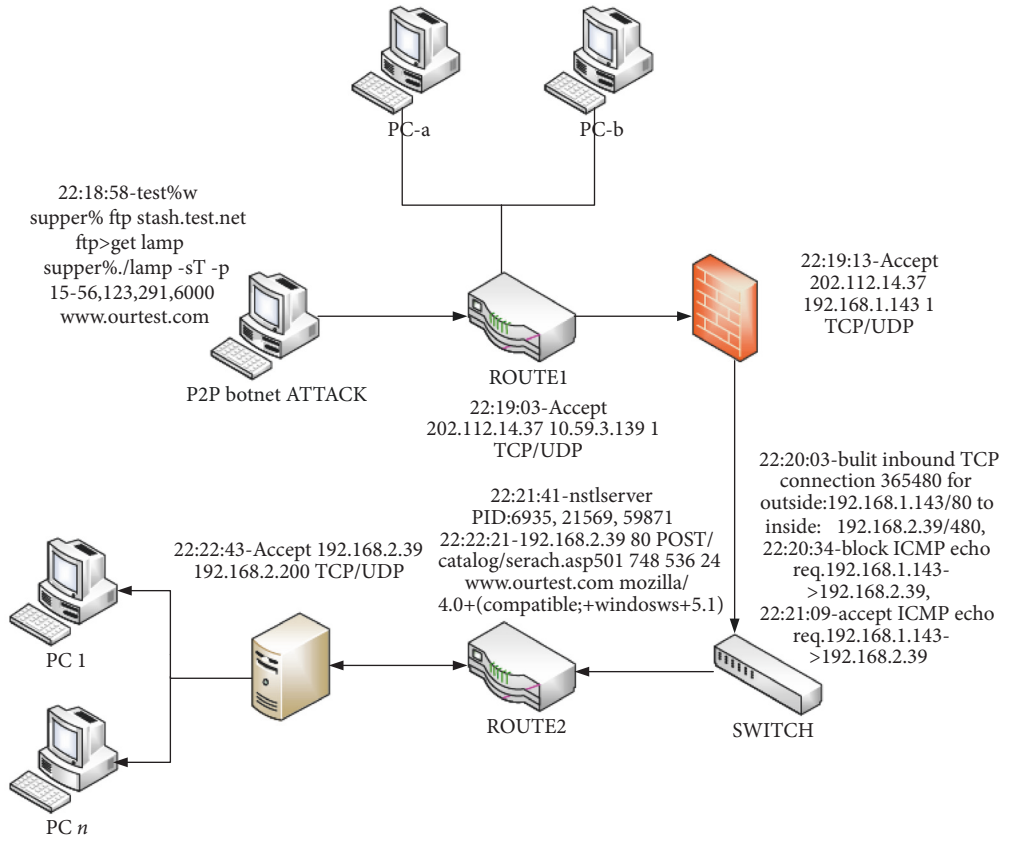

Figure 9: HTTP botnet attack reconstruction.

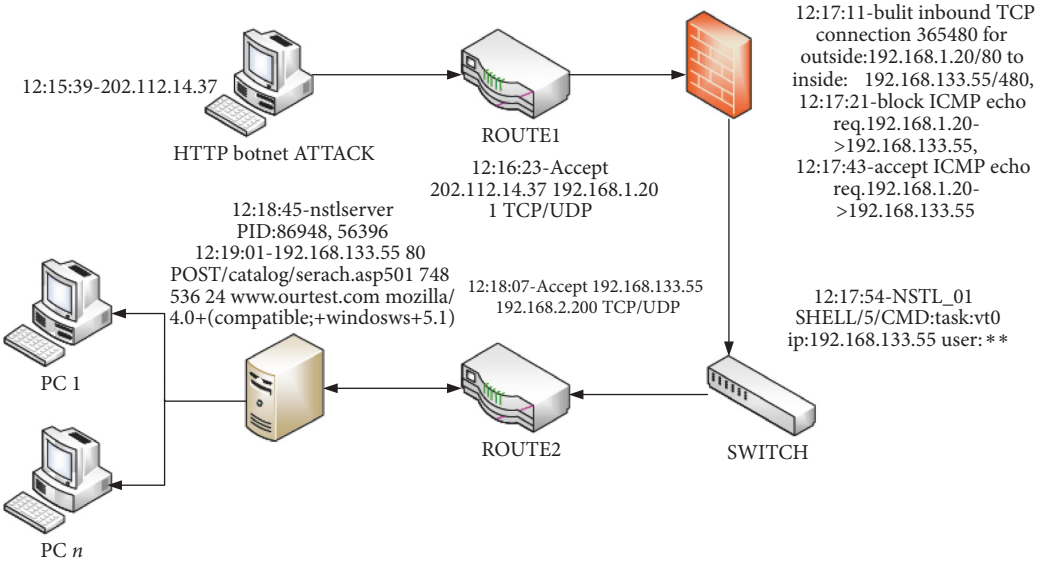

Figure 10: P2P botnet attack reconstruction.

References

[1] A. Sood and R. Enbody, *Targeted Cyber Attacks: Multi-Staged Attacks Driven by Exploits and Malware*, Syngress, 2014.

[2] K. L. Chiew, K. S. C. Yong, and C. L. Tan, "A survey of phishing attacks: Their types, vectors and technical approaches," *Expert Systems with Applications*, vol. 106, pp. 1–20, 2018.

[3] B. Caswell, J. Beale, and A. Baker, *Snort Intrusion Detection and Prevention Toolkit*, Syngress, 2007.

[4] A. Yaar, A. Perrig, and D. Song, "Pi: a path identification mechanism to defend against DDoS attacks," in *Proceedings of the 2003 Symposium on Security and Privacy, SP 2003*, pp. 93–107, USA, May 2003.

[5] H. Wenhua and Y. Geng, "Identification method of attack path based on immune intrusion detection," *Journal of Networks*, vol. 9, no. 4, pp. 964–971, 2014.

[6] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: protecting connection setup from denial-of-capability attacks," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 289–300, 2007.

[7] K. Kleemola, M. Crete-Nishihata, and J. Scott-Railton, Targeted Attacks against Tibetan and Hong Kong Groups Exploiting CVE-2014-4114, Citizen Lab, June 2015.

[8] J. R. Crandall, G. Wassermann, D. A. de Oliveira, Z. Su, S. F. Wu, and F. T. Chong, "Temporal search: detecting hidden malware timebombs with virtual machines," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 5, pp. 25–36, 2006.

[9] Z. Gu, K. Pei, Q. Wang, L. Si, X. Zhang, and D. Xu, "LEAPS: detecting camouflaged attacks with statistical learning guided by program analysis," in *Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 57–68, Rio de Janeiro, Brazil, June 2015.

[10] K. Pei, Z. Gu, B. Saltaformaggio et al., "Hercule: attack story reconstruction via community discovery on correlated log graph," in *Proceedings of the the 32nd Annual Conference on Computer Security Applications*, pp. 583–595, ACM, Los Angeles, Calif, USA, December 2016.

[11] H.-S. Chen, W. Gao, and D. G. Daut, "Signature based spectrum sensing algorithms for IEEE 802.22 WRAN," in *Proceedings of the 2007 IEEE International Conference on Communications, ICC'07*, pp. 6487–6492, IEEE, UK, June 2007.

[12] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *Proceedings of the 2006 IEEE International Conference on Communications, ICC 2006*, pp. 2388–2393, IEEE, Turkey, July 2006.

[13] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández et al., "Anomaly-based network intrusion detection: techniques, systems and challenges," *Journal of Computers and Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[14] F. Gong, "Deciphering detection techniques: Part ii anomaly-based intrusion detection," White Paper, McAfee Security, 2003.

[15] Y. Yu, "A survey of anomaly intrusion detection techniques," *Journal of Computing Sciences in Colleges*, vol. 28, no. 1, pp. 9–17, 2012.

[16] F. Sönmez, M. Zontul, O. Kaynar, and H. Tutar, "Anomaly detection using data mining methods in it systems: a decision support application," *Sakarya University Journal of Science*, vol. 22, no. 4, pp. 1109–1123, 2018.

[17] N. Kamiyama and T. Mori, "Simple and accurate identification of high-rate flows by packet sampling," in *Proceedings of the Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pp. 1–13, IEEE, Barcelona, Spain, April 2006.

[18] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet Computing*, vol. 10, no. 1, pp. 82–89, 2006.

[19] J. Lu, K. Chen, Z. Zhuo, and X. Zhang, "A temporal correlation and traffic analysis approach for APT attacks detection," *Cluster Computing*, pp. 1–12, 2017.

[20] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Journal of Computers and Security*, vol. 45, pp. 100–123, 2014.

[21] M. Parkour, Contagio malware database, https://www.mediafire.com/folder/c2az029ch6cke/TRAFFIC_PATTE%20RNS_COLLECTION, 2013.

[22] S. Siddiqui, M. S. Khan, K. Ferens, and W. Kinsner, "Detecting advanced persistent threats using fractal dimension based machine learning classification," in *Proceedings of the 2nd ACM International Workshop on Security and Privacy Analytics (IWSPA '16)*, pp. 64–69, ACM, 2016.