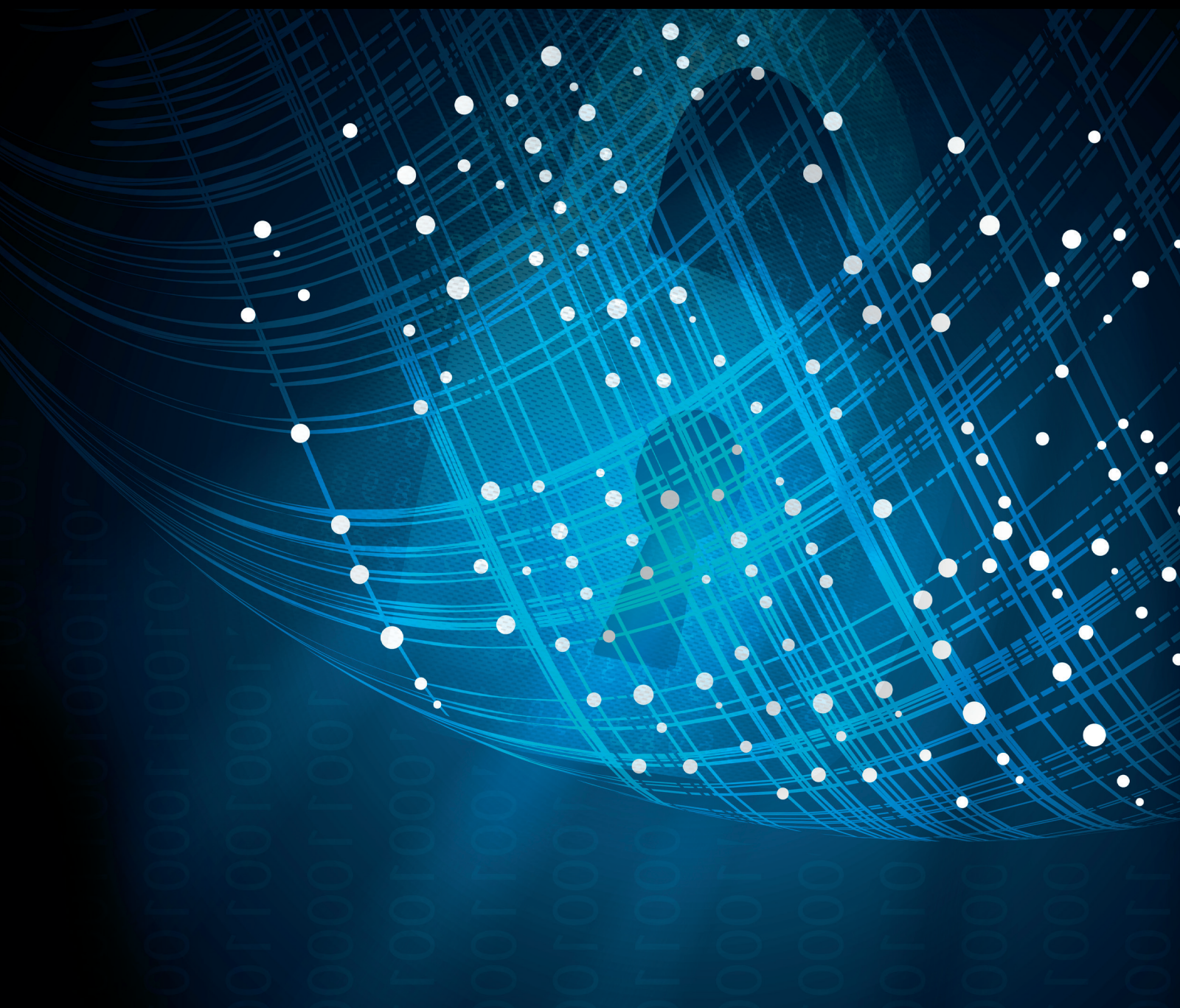


Security Measurements of Cyber Networks

Special Issue Editor in Chief: Zheng Yan

Guest Editors: Yuqing Zhang, Raymond Choo, and Yang Xiang





Security Measurements of Cyber Networks

Security and Communication Networks

Security Measurements of Cyber Networks

Special Issue Editor in Chief: Zheng Yan

Guest Editors: Yuqing Zhang, Raymond Choo, and Yang Xiang



Copyright © 2018 Hindawi. All rights reserved.

This is a special issue published in “Security and Communication Networks.” All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Editorial Board


Mamoun Alazab, Australia
Cristina Alcaraz, Spain
Angelos Antonopoulos, Spain
Frederik Armknecht, Germany
Benjamin Aziz, UK
Alessandro Barenghi, Italy
Pablo Garcia Bringas, Spain
Michele Bugliesi, Italy
Pino Caballero-Gil, Spain
Tom Chen, UK
Kim-Kwang Raymond Choo, USA
Alessandro Cilardo, Italy
Stelvio Cimato, Italy
Vincenzo Conti, Italy
Salvatore D'Antonio, Italy
Paolo D'Arco, Italy
Alfredo De Santis, Italy
Angel M. Del Rey, Spain
Roberto Di Pietro, France
Jesús Díaz-Verdejo, Spain
Nicola Dragoni, Denmark
Carmen Fernandez-Gago, Spain
Clemente Galdi, Italy

Dimitrios Geneiatakis, Italy
Bela Genge, Romania
Debasis Giri, India
Prosanta Gope, UK
Francesco Gringoli, Italy
Jiankun Hu, Australia
Ray Huang, Taiwan
Tao Jiang, China
Minho Jo, Republic of Korea
Bruce M. Kapron, Canada
Kiseon Kim, Republic of Korea
Sanjeev Kumar, USA
Maryline Laurent, France
Jong-Hyouk Lee, Republic of Korea
Huaizhi Li, USA
Zhe Liu, Canada
Pascal Lorenz, France
Leandros Maglaras, UK
Emanuele Maiorana, Italy
Vincente Martin, Spain
Fabio Martinelli, Italy
Barbara Masucci, Italy
Jimson Mathew, UK



David Megias, Spain
Leonardo Mostarda, Italy
Qiang Ni, UK
Petros Nicopolitidis, Greece
David Nuñez, USA
A. Peinado, Spain
Gerardo Pelosi, Italy
Gregorio Martinez Perez, Spain
Pedro Peris-Lopez, Spain
Kai Rannenberg, Germany
Francesco Regazzoni, Switzerland
Khaled Salah, UAE
Salvatore Sorce, Italy
Angelo Spognardi, Italy
Sana Ullah, Saudi Arabia
Ivan Visconti, Italy
Guojun Wang, China
Zheng Yan, China
Qing Yang, USA
Kuo-Hui Yeh, Taiwan
Sherali Zeadally, USA
Zonghua Zhang, France

Contents






Security Measurements of Cyber Networks

Zheng Yan , Yuqing Zhang, Kim-Kwang Raymond Choo, and Yang Xiang
Editorial (3 pages), Article ID 6545314, Volume 2018 (2018)



An Imbalanced Malicious Domains Detection Method Based on Passive DNS Traffic Analysis

Zhenyan Liu , Yifei Zeng, Pengfei Zhang, Jingfeng Xue , Ji Zhang, and Jiangtao Liu
Research Article (7 pages), Article ID 6510381, Volume 2018 (2018)



OFFDTAN: A New Approach of Offline Dynamic Taint Analysis for Binaries

Xiajing Wang , Rui Ma , Bowen Dou , Zefeng Jian , and Hongzhou Chen 
Research Article (13 pages), Article ID 7693861, Volume 2018 (2018)

Security Measurement for Unknown Threats Based on Attack Preferences

Lihua Yin, Yanwei Sun , Zhen Wang, Yunchuan Guo , Fenghua Li, and Binxing Fang
Research Article (13 pages), Article ID 7412627, Volume 2018 (2018)

HAC: Hybrid Access Control for Online Social Networks

Fangfang Shan , Hui Li, Fenghua Li, Yunchuan Guo , and Ben Niu
Research Article (11 pages), Article ID 7384194, Volume 2018 (2018)



Data Fusion for Network Intrusion Detection: A Review

Guoquan Li, Zheng Yan , Yulong Fu , and Hanlu Chen
Review Article (16 pages), Article ID 8210614, Volume 2018 (2018)

Uncovering Tor: An Examination of the Network Structure

Bryan Monk, Julianna Mitchell , Richard Frank, and Garth Davies
Research Article (12 pages), Article ID 4231326, Volume 2018 (2018)


An Approach for Internal Network Security Metric Based on Attack Probability

Chun Shan , Benfu Jiang, Jingfeng Xue , Fang Guan, and Na Xiao
Research Article (11 pages), Article ID 3652170, Volume 2018 (2018)

A Dynamic Hidden Forwarding Path Planning Method Based on Improved Q-Learning in SDN Environments

Yun Chen , Kun Lv , and Changzhen Hu 
Research Article (12 pages), Article ID 2058429, Volume 2018 (2018)



A New Type of Graphical Passwords Based on Odd-Elegant Labelled Graphs

Hongyu Wang , Jin Xu, Mingyuan Ma, and Hongyan Zhang
Research Article (11 pages), Article ID 9482345, Volume 2018 (2018)

Analysis on Influential Functions in the Weighted Software Network

Haitao He, Chun Shan , Xiangmin Tian , Yalei Wei, and Guoyan Huang
Research Article (10 pages), Article ID 1525186, Volume 2018 (2018)

Security Feature Measurement for Frequent Dynamic Execution Paths in Software System

Qian Wang , Jiadong Ren, Xiaoli Yang, Yongqiang Cheng , Darryl N. Davis, and Changzhen Hu
Research Article (10 pages), Article ID 5716878, Volume 2018 (2018)

Editorial

Security Measurements of Cyber Networks

Zheng Yan ^{1,2}, Yuqing Zhang,³ Kim-Kwang Raymond Choo,⁴ and Yang Xiang⁵

¹The State Key Lab of ISN, School of Cyber Engineering, Xidian University, Xi'an, China

²The Department of Communications and Networking, Aalto University, Espoo, Finland

³School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing, China

⁴Department of Information Systems and Cyber Security, University of Texas at San Antonio, TX 78249, USA

⁵School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, Australia

Correspondence should be addressed to Zheng Yan; zheng.yan@aalto.fi

Received 7 May 2018; Accepted 7 May 2018; Published 10 October 2018

Copyright © 2018 Zheng Yan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cyber networks facilitate and expedite the development of information systems, communication technologies, and digital economies. However, as the cyber networks become large, heterogeneous, and pluralistic, evaluating the security of cyber networks becomes increasingly challenging. Successful attacks and intrusions against cyber networks can result in significant impacts, ranging from personal risk to national-level security. Thus, there is an increasing need to understand and measure network security by considering a range of requirements and standards, and in the context of different network scenarios. Although many prior studies have focused on network intrusion detection, malware detection, and security threat defense, a generally accepted security measurement framework is still lacking. Such a framework is urgently required for quick identification of security holes, assessment of potential threats, and implementation of efficient countermeasures.

Security measurement theories and methods concern several pertinent questions in terms of security-related data collection, composition, analytics, and processing. This is especially crucial for detecting security threats and measuring cyber network security in a quantified, precise, and efficient manner. (i) How can we adaptively collect related data for security measurement for large-scale heterogeneous networks in a generic and pervasive way? (ii) How can we compose and fuse collected big data without incurring expensive data storage and transmission costs, as well as ensuring efficient data processing to facilitate precise security threat detection and judgment? (iii) How can we protect

valuable data, preserve data privacy, and effectively control access to key data, as well as manage its storage? (iv) How can we aggregate and mine security-related data to measure the security of the whole network in a quantifiable manner and with high trustworthiness? These open and interesting issues are now attracting attention from both the research community and the practitioner community.

This special issue brings together recent advances on security-related data processing and analytics, detection of malware, virus and network intrusion, and network system protection, in the context of network security assessment and measurement. We selected 12 research papers from more than 30 submissions after rigorous peer-reviews. Next, we categorize these 12 accepted papers into three categories and briefly discuss each paper.

1. Intrusion Detection and Network Security Measurement

In the survey entitled “Data Fusion for Network Intrusion Detection: A Review”, G. Li et al. conducted a comprehensive review on massively high dimensional data fusion techniques for network intrusion detection in order to accurately detect complex or synthetic network attacks.

Based on attack prediction, L. Yin et al. proposed a method of security measurement in their paper entitled “Security Measurement for Unknown Threats Based on Attack Preferences”. They computed optimal attack timing

from the perspective of attackers. They used a long-term game to estimate the risk of being found in terms of choosing optimal timing according to risk and profit. On the basis of game theoretical analysis, the likelihood of being attacked for each node is estimated as a security metric result.

Z. Liu et al. attempted to build a malicious domains detection model oriented to imbalanced data. They proposed an imbalanced malicious domain detection method based on passive DNS traffic analysis in the paper entitled “An Imbalanced Malicious Domains Detection Method Based on Passive DNS Traffic Analysis”. It can deal with not only between-class imbalance problem but also within-class imbalance problem.

Y. Xu et al. explored the metrics of network security in the paper entitled “Security Metrics Methods of the Network System”. They considered network security from three aspects: environment security, reliability security, and vulnerability security. Based on the discussion on fragility distribution, the authors proposed a technical tool, vulnerability graph, to characterize the properties of vulnerabilities and introduced a metric function to measure the degree of damage of vulnerabilities.

In the paper entitled “Uncovering Tor: An Examination of the Network Structure”, B. Monk et al. used social network analysis to examine hyperlink connections and the structure of website communities they form on Tor and how characteristics of these communities could have implications for criminal activity on Tor as understood through the lens of social disorganization theory.

In the paper entitled “An Approach for Internal Network Security Metric Based on Attack Probability”, C. Shan et al. proposed an internal network security metric based on attack probability. It simplifies network attack graph for a large-scale network and assists network security detection.

2. Software Fault Location and Maintenance

In the paper entitled “Security Feature Measurement for Frequent Dynamic Execution Paths in Software System”, Q. Wang et al. proposed a security feature measurement algorithm of frequent dynamic execution paths in software to provide a basis for improving the security and reliability of software. By using a complex network model and a sequence model and combining them with the invocation and dependency relationships between function nodes, the authors can analyze fault cumulative effect and spread effect. The function node security features of the software complex network are also defined and measured. In addition, frequent software execution paths are mined and weighted in order to obtain security metrics of the frequent paths.

For understanding software design patterns and controlling their development and maintenance process, H. He et al. proposed an approach to define node importance for mining influential software nodes based on invoking dependency relationships among the nodes in the paper entitled “Analysis on Influential Functions in the Weighted Software Network”.

The authors constructed a weighted software network to represent software execution dependency structure and proposed an algorithm to evaluate node importance to further obtain the most influential nodes in the software network based on it.

In the paper entitled “OFFDTAN: A New Approach of Offline Dynamic Taint Analysis for Binaries”, X. Wang et al. proposed an approach to offline dynamic taint analysis for binaries through four stages: dynamic information acquisition, vulnerability modeling, offline analysis, and backtrace analysis.

3. Network Security Countermeasures

In the paper entitled “A Dynamic Hidden Forwarding Path Planning Method Based on Improved Q-Learning in SDN Environments”, Y. Chen et al. proposed an improved Q-learning algorithm to automatically plan an optimal attack path. They adopted Software-Defined Network (SDN) to adjust routing paths and dynamically create hidden forwarding paths to filter vicious attack requests.

F. Shan et al. proposed a hybrid access control model (HAC) that leverages attributes and relationships to control access to resources in the paper entitled “HAC: Hybrid Access Control for Online Social Networks”. They designed a new policy specification language to express both the relationships and attributes of users and proposed a path checking algorithm to figure out if a path between two users can satisfy with a hybrid policy.

H. Wang et al. studied graphical password (GPW) used in information communications in the paper entitled “A New Type of Graphical Passwords Based on Odd-Elegant Labelled Graphs”. The authors designed new Topsnut-GPWs by means of a graph labelling, called odd-elegant labelling. The new Topsnut-GPWs are constructed using Topsnut-GPWs (a type of GPWs) with smaller vertex numbers and more robust to deciphering attacks, compared with traditional Topsnut-GPWs.

4. Summary

Editing this special issue has been an invaluable experience for us. We hope this special issue will provide a useful reference to its readers and contribute to advances in network security, as well as stimulate additional innovation.

Acknowledgments

This work is sponsored by the National Key Research and Development Program of China (Grant 2016YFB0800700), the NSFC (Grants 61672410 and U1536202), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program no. 2016ZDJC-06), and the 111 project (Grants B08038 and B16037). We would like to express our appreciation to all authors for their contributions and the reviewers for their valuable review comments. We

also sincerely thank the Editorial Board of ***Security and Communication Networks*** for approving this special issue and their continuous support on its final publication.

Zheng Yan
Yuqing Zhang
Kim-Kwang Raymond Choo
Yang Xiang

Research Article

An Imbalanced Malicious Domains Detection Method Based on Passive DNS Traffic Analysis

Zhenyan Liu , Yifei Zeng, Pengfei Zhang, Jingfeng Xue , Ji Zhang, and Jiangtao Liu

Beijing Key Laboratory of Software Security Engineering Technology, School of Software, Beijing Institute of Technology, Beijing 100081, China

Correspondence should be addressed to Zhenyan Liu; zhenyanliu@bit.edu.cn

Received 9 January 2018; Accepted 19 April 2018; Published 20 June 2018

Academic Editor: Yuqing Zhang

Copyright © 2018 Zhenyan Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Although existing malicious domains detection techniques have shown great success in many real-world applications, the problem of learning from imbalanced data is rarely concerned with this day. But the actual DNS traffic is inherently imbalanced; thus how to build malicious domains detection model oriented to imbalanced data is a very important issue worthy of study. This paper proposes a novel imbalanced malicious domains detection method based on passive DNS traffic analysis, which can effectively deal with not only the between-class imbalance problem but also the within-class imbalance problem. The experiments show that this proposed method has favorable performance compared to the existing algorithms.

1. Introduction

With the rapid development of the Internet and information technology, network security threats are escalating, the security of cyberspace is becoming more and more complex and hidden, the risk of network security is increasing, and various network malicious attacks emerge endlessly. In these network malicious attacks, most of them are based on DNS (Domain Name System). The reason why DNS can provide an available infrastructure for attackers is that it is open and ease of use.

The core of the network malicious attack based on DNS is C&C (Command and Control) server. By means of the C&C server, the attackers can order remote hosts to perform malicious activities, such as spamming, phishing, DDOS (Distributed Denial of Service), and distributing malware which may be used to steal information, disrupt computer, extort money, etc. Therefore, it is urgent to detect this kind of malicious domain of C&C server and further take corresponding countermeasure.

It is very popular to employ the classification algorithm in machine learning to detect malicious domains in the current research [1, 2]. However, these existing studies pay no or little attention to the problem of imbalanced data. In fact, the actual DNS traffic is inherently imbalanced, in which most of the cases are benign and far fewer cases are

malicious. As a result, this tends to construct an imbalanced training dataset in which there are many more samples of some categories than others. When learning from an imbalanced dataset, class information must be considered; otherwise the classifier will be overwhelmed by the majority classes and ignores the minority ones, and then the overall classification performance will undoubtedly be degraded. To address this shortfall, this paper will propose an imbalanced malicious domains detection method which can build malicious domains detection model by learning imbalanced dataset based on passive DNS traffic analysis.

In this paper we make the following contributions:

- (1) We especially focus on learning from the imbalanced data in the malicious domains detection field. And the latest research progress of learning from the imbalanced data in other fields is invited in the malicious domains detection field.
- (2) We construct the stronger discriminative features to profile malicious domains based on passive DNS traffic analysis.
- (3) We propose an improved imbalanced malicious domains detection method which is an extension of EasyEnsemble and demonstrate its favorable performance by the comparative experiments.

The remainder of this paper is organized as follows. In Section 2, we briefly review related work. Section 3 describes how to profile malicious domains based on passive DNS traffic analysis. We elaborate on an imbalanced malicious domains detection method in Section 4. Section 5 presents our comparative experiments of this new method. Finally, we conclude the paper in Section 6.

2. Related Work

2.1. Learning from Imbalanced Data. Although rarely in network security, learning from the imbalanced data has already made considerable progress in other fields. In general, there are three ways to tackle the imbalanced learning problem. The first one is from the data perspective, which mainly uses resampling approaches to modify the class distribution of the data. The second one is from the algorithm perspective, which mostly focuses on optimizing various algorithms, such as SVM (Support Vector Machine), Decision Tree and Neural Network, based on cost-sensitive learning which considers the costs associated with misclassifying samples [3]. In addition, some researches also utilize one-class learning [4] which is particularly useful when used on extremely imbalanced data sets. The third one is from data feature perspective, which can build a fair feature space attaching much weight to the minority classes by means of some improved feature selection methods. This third approach is applied in many applications, including fraud/churn detection, text categorization, medical diagnosis, detection of software defects, and many others [5].

Most researches have been focused on the first approach, resampling which is more practical than the other two approaches. The resampling includes undersampling, oversampling, and the integration of undersampling and oversampling [6]. The key idea of undersampling is to remove the majority class samples from the original data set, and the key idea of oversampling is to append the minority class samples to the original data set.

The simplest resampling technique is random. But random undersampling can potentially remove certain important samples, and random oversampling can lead to overfitting. Various improved undersampling algorithms, including EasyEnsemble and BalanceCascade, have been proposed [7]. Both methods utilize ensemble learning to overcome the deficiency of information loss introduced in the traditional random undersampling, since ensemble learning is based on multiple subsets which contain more information than a single one [8]. The famous improved oversampling algorithms are SMOTE (Synthetic Minority Oversampling Technique) [9] and its variants, such as Borderline-SMOTE [10] and ADASYN (Adaptive Synthetic Sampling) [11]. They devote to create the excellent artificial minority class samples using different strategies.

In practical application, when the samples of the minority classes are absolutely rare, oversampling is generally employed to increase the samples of the minority classes. Or else when the samples of the minority classes are relatively rare, undersampling is generally employed to decrease the samples of the majority classes.

2.2. Malicious Domain Detection Based on Passive DNS Traffic Analysis. The majority of detection methods based on DNS traffic are data-driven, most commonly having machine learning algorithms at their core. These methods require accurate ground truth of both malicious and benign DNS traffic for model training as well as for the performance evaluation [12]. The methods of DNS data collection can be generally divided into two subcategories: active and passive. *Active* method obtains DNS data by deliberately sending DNS queries and record the corresponding DNS responses, while *passive* method is passively to backup real DNS queries and responses.

Compare with *active* DNS data collection, *passive* DNS data collection is more representative and more comprehensive. As a result, the detection of malicious domain based on passive DNS traffic analysis has received increasing attention from the research community over the past decade. "Passive DNS" was invented by Weimer [13] in 2004. After that, many researchers have an insight into the important value of passive DNS when doing incident response investigations. And many passive DNS systems have developed, in which the most famous and popular one is DNSDB from Farsight Security. Farsight collects passive DNS data from its global sensor array, and then filters and verifies the DNS transactions before inserting them into the DNSDB [14]. The trends within this set are believed to be representative of Internet-wide trends and therefore provide valuable insight.

Antonakakis et al. [1] proposed a dynamic reputation system for DNS, called Notos, to automatically assign a low reputation score to a malicious domain. To measure a number of statistical features of a domain, Notos used historical DNS information collected passively from multiple recursive DNS resolvers distributed across the Internet. Bilge et al. [2] introduced a passive DNS analysis approach and a detection system, EXPOSURE, to detect domain names that are involved in malicious activities. The data that EXPOSURE used for the initial training consist of DNS traffic from the real-time response data from authoritative Name Servers located in North America and in Europe.

Perdisci et al. [15] presented FluxBuster, a novel detection system that used a purely passive approach for detecting and tracking malicious flux networks. FluxBuster is based on large-scale passive analysis of DNS traffic generated by hundreds of local recursive DNS (RDNS) servers located in different networks and scattered across several different geographical locations. Zhou et al. [16] proposed a model which can detect Fast-Flux Domains using random forest algorithm. It used passive DNS to log domain name query history of real campus network environment.

Analyzing these existing related works, we discovered that most of them are to collect DNS traffic in a period time to form a passive DNS set. This kind of passive DNS set is only a DNS data fragment and needs more collection cost. While DNSDB is relatively comprehensive, as a result, we determined to use the passive DNS traffic from DNSDB in this paper.

3. Profiling Malicious Domains Based on Passive DNS Traffic Analysis

To profile malicious domains, based on passive DNS traffic analysis we extract two groups features of malicious domains: static lexical features and dynamic DNS resolving features. Static lexical features mainly origin from the lexical information of domain name. Dynamic DNS resolving features are constructed based on DNS response attributes. Table 1 gives an overview of these features.

The results of statistical analysis of some features are selected to show in Figure 1. From these, we can find that these features have the stronger ability to distinguish the malicious domains from the benign ones.

In this section, we will present 12 static lexical features and 4 dynamic DNS resolving features and the motivation that we construct these features to profile malicious domains.

3.1. Static Lexical Features. To avoid detection, the attackers generally employ domain generation algorithms (DGA) to dynamically produce a large number of random domain names. The lexical features of these malicious domain names are largely different from benign domain names. We construct 12 static lexical features to profile malicious domains.

So far the short domain names have been almost registered; therefore the majority of malicious domain names generated by DGA are longer than benign domain names. And max length of labels (i.e., parts delimited by dots) in subdomain of malicious domain names is also commonly longer. So we construct two features based on the length measure: first, length of domain name (Feature 1), and second, max length of labels in subdomain (Feature 2).

The most distinctive property of domain names generated by DGA is that the distribution of characters is random. We know that information entropy is defined as the average amount of information produced by a stochastic source of data [17]. So, we employ information entropy to measure the disorder of characters.

Let d be a domain name and m be the number of distinct characters in d . We define entropy (d) as character entropy of d (Feature 3).

$$Entropy(d) = -\sum_{i=1}^m \left(\frac{\text{count}(a_i)}{\text{length}(d)} \right) \log_2 \left(\frac{\text{count}(a_i)}{\text{length}(d)} \right) \quad (1)$$

where a_i ($i = 1 \dots m$) means a character in d , $\text{count}(a_i)$ is the number of a_i in d , and $\text{length}(d)$ is the length of d .

If the character entropy value of d is greater, then more likely d will be identified to be malicious.

In addition, malicious domain names are used by malwares not by human, so they are not easy-to-remember or human pronounceable. Thus the appearance of numerical and alphabetic characters in malicious domain names is also very important indicative signs. With this insight, we construct five features as follows: number of numerical characters (Feature 4), ratio of numerical characters (Feature 5), conversion frequency of numerical and alphabetic character (Feature 6), max length of continuous numerical characters (Feature 7), max length of continuous alphabetic characters

(Feature 8), and max length of continuous same alphabetic characters (Feature 9).

As we all know, the consonant letters in the English alphabet are much more than the vowel letters. Therefore, in random malicious domain names, the ratio of vowels (Feature 10) is smaller, the length of continuous consonants (Feature 11) is longer, and conversion frequency of vowel and consonant (Feature 12) is very higher.

3.2. Dynamic DNS Resolving Features. The Internet-scale attacks using DNS leave unavoidably a trail of footmarks which are hidden into the DNS resolving records, so we may mine these footmarks (i.e., DNS resolving features) to profile malicious domains. In this section, we will present 4 dynamic resolving features origin from the DNS resolving records.

In order to evade blacklists and resist takedowns, the DNS answer that is returned by the server for a malicious domain generally consists of multiple DNS A records (i.e., Address records) or NS records (i.e., Name Server records). And the slippery attackers do not usually target specific Name Server or IP ranges. Therefore, we construct four statistical features as follows: number of distinct A records (Feature 13), IP entropy of domain name (Feature 14), number of distinct NS records (Feature 15), and similarity of NS domain name (Feature 16).

Number of distinct A records (Feature 13) records the total number of IP addresses resolved in DNSDB. Furthermore, IP entropy of domain name (Feature 14) is constructed to measure the dispersion of these IP addresses resolved. Let d be a domain name, S be the set of these IP addresses resolved, and n be the number of distinct IP/16 prefixes in S . We define $IP_Entropy(d)$ as IP entropy of domain name (Feature 14).

$$IP_Entropy(d) = -\sum_{i=1}^n \left(\frac{\text{count}(\text{ipx}_i)}{|S|} \right) \log_2 \left(\frac{\text{count}(\text{ipx}_i)}{|S|} \right) \quad (2)$$

where ipx_i ($i = 1 \dots n$) means an IP/16 prefix in S , $\text{count}(\text{ipx}_i)$ is the number of ipx_i in S , and $|S|$ is the size of S .

If the IP entropy value of d is greater, then more likely d will be identified to be malicious.

Number of distinct NS records (Feature 15) records the total number of Name Servers resolved in DNSDB. Furthermore, Similarity of NS domain name (Feature 16) is constructed to measure the difference of these Name Servers resolved. We calculate the Edit Distance between every pair of Name Server names of a domain, and then the average of these distances is defined as the similarity of NS domain name. If the similarity of NS domain name of d is bigger, then more likely d will be identified to be malicious.

4. An Imbalanced Malicious Domains Detection Method

Almost all classification algorithms seem to be powerless to learn from an extremely imbalanced training data set. In consideration of the actual imbalanced distribution of

TABLE 1: An overview of domain features.

Feature group	No.	Feature Name	Malicious domain profile
Static lexical features	1	Length of domain name	Longer
	2	Max length of labels in subdomain	Longer
	3	Character entropy	Greater
	4	Number of numerical characters	Higher
	5	Ratio of numerical characters	Higher
	6	Conversion frequency of numerical and alphabetic character	Higher
	7	Max length of continuous numerical characters	Shorter
	8	Max length of continuous alphabetic characters	Longer
	9	Max length of continuous same alphabetic characters	Shorter
	10	Ratio of vowels	Lower
	11	Max length of continuous consonants	Longer
	12	Conversion frequency of vowel and consonant	Higher
Dynamic DNS resolving features	13	Number of distinct A records	Higher
	14	IP entropy of domain name	Higher
	15	Number of distinct NS records	Higher
	16	Similarity of NS domain name	Bigger

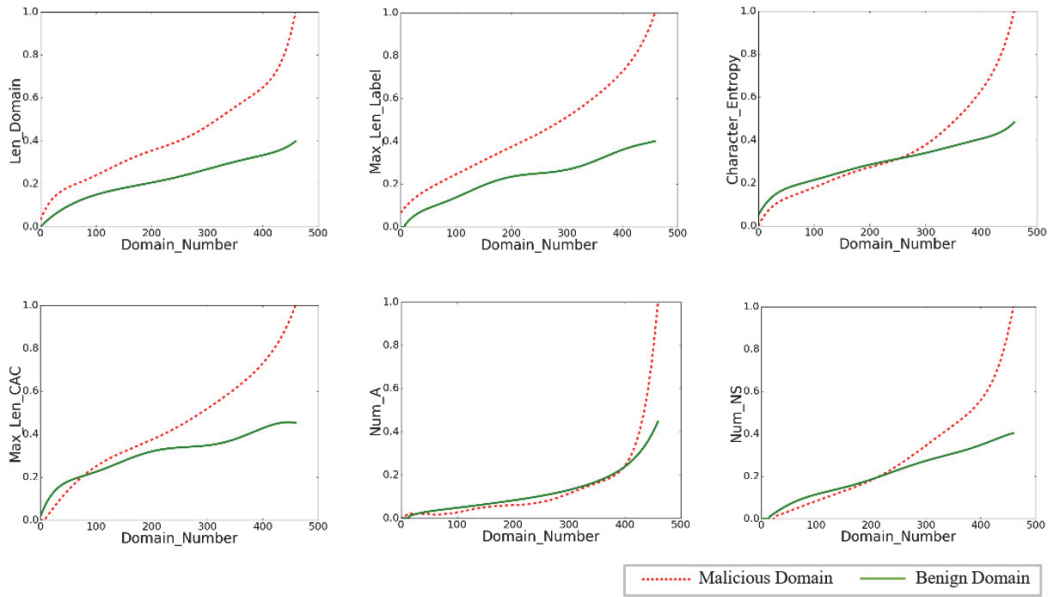


FIGURE 1: The results of statistical analysis of some features.

DNS traffic data (i.e., malicious domains are relatively rare), inspired by existing methods, our research focuses on the combination of undersampling and ensemble learning.

In existing methods, EasyEnsemble [7] is a typical improved algorithm combining undersampling with ensemble learning. As we know, the main deficiency of undersampling is that potentially useful information contained in the unselected examples is neglected. To remedy this deficiency, EasyEnsemble incorporates ensemble learning into undersampling.

The idea behind EasyEnsemble is quite simple. Given the majority class instances set N_{maj} and the minority

class instances set N_{min} , this method independently samples several subsets N_1, N_2, \dots, N_T from N_{maj} , where $|N_i| = |N_{min}|$ ($i = 1, 2, \dots, T$). For each subset N_i , a base classifier is trained using N_i and N_{min} . All base classifiers are combined for the final decision. Remarkably, many learning algorithms can be employed to generate the base classifier.

EasyEnsemble make better use of the majority class than undersampling by ensemble learning, so it is very helpful for between-class imbalance learning. However, EasyEnsemble ignores within-class imbalance, especially for the majority class. That is, in the majority class some instances are highly similar which may form several clusters, and more other

- (1) {Input: A set of minority class examples N_{\min} , a set of majority class examples N_{\max} , $|N_{\min}| < |N_{\max}|$, the number of subsets T to sample from N_{\max} }
- (2) N_{\max} are clustered into several small groups G_1, G_2, \dots by HAC
- (3) $i \leftarrow 0$
- (4) **repeat**
- (5) $i \leftarrow i + 1$
- (6) Select randomly $(\alpha_j |G_j|)$ instances from each cluster G_j ($j = 1, 2, \dots$) with a total of K
- (7) Select randomly $|N_i| - K$ instances from $N_{\max} - \Sigma G_j$
- (8) Combine the dataset sampled from step (6) and (7) to form a subset N_i , where $|N_i| = |N_{\min}|$
- (9) Learn H_i using N_i and N_{\min} , H_i is a base classifier employed Decision Tree
- (10) **until** $i = T$
- (11) Output: An ensemble $H(\mathbf{x}) = \operatorname{argmax}_c \sum_{i=1}^T I(H_i(\mathbf{x}) = c)$

ALGORITHM 1: The HAC_EasyEnsemble algorithm.

instances are almost unique. This kind of phenomena is commonly called “long-tailed distribution” in the statistical sense.

We should select a representative subset from each cluster and combine them with a subset selected randomly from the other unique instances set to form a preliminary subset. According to this idea, we proposed an improved EasyEnsemble method to learn imbalanced DNS traffic data.

In this novel method, firstly the instances in the majority class are clustered together in several small groups G_1, G_2, \dots by Hierarchical Agglomerative Clustering (HAC). For each cluster G_j ($j = 1, 2, \dots$), according to the size of G_j , we select randomly several instances with a total of K . And then we select randomly $|N_i| - K$ ($i = 1, 2, \dots, T$) instances from $N_{\max} - \Sigma G_j$ to form a subset N_i , where $|N_i| = |N_{\min}|$. Base classifier H_i is trained using N_i and N_{\min} . All T base classifiers are combined for the final decision. Note that Decision Tree algorithm is employed to generate the base classifier.

The pseudocode of the improved EasyEnsemble named HAC_EasyEnsemble is shown in Algorithm 1.

Noted that here I is an indicative function, and c is the class label, if the parameter of I is true, then return 1, or else return 0. In HAC, we may employ various cluster proximity measures which are typically complete link, group average, Ward’s method [18], etc. For the complete link, the proximity of two clusters is defined as the maximum of distance (minimum of the similarity) between any two points in the two different clusters. For the group average, the proximity of two clusters is defined as the average pairwise proximity among all pairs of points in the different clusters. For Ward’s method, the proximity of two clusters is defined as the increase in the squared error that results when two clusters are merged [19].

5. Experiment

In order to verify the novel HAC_EasyEnsemble algorithm used to learn imbalanced DNS traffic data, we do a series of experiments to compare the performance of HAC_EasyEnsemble and EasyEnsemble based on the same dataset. And we use three different cluster proximity

measures in HAC: complete link, group average, and Ward’s method.

Originally, we construct an imbalanced training set which contains 6400 benign domains (from alexa.com) and 3000 malicious domains (from cybercrime-tracker.net, malware-domains.com, and hosts-file.net etc.). The reason for this ratio of malicious domains is that the HAC_EasyEnsemble algorithm is more effective for relatively rare malicious domains, not absolutely. The DNS resolving records of these domains are obtained by DNSDB API, and then 12 static lexical features and 4 dynamic DNS resolving features listed in Section 3 are constructed based on these records.

Commonly the evaluation measures for the imbalanced classification are macroaveraged precision, macroaveraged recall, macroaveraged F1 [20]. Since macroaveraged scores are averaged values over the number of categories, then the performance of classifier is not dominated by major categories. Let P be the precision, R be recall, and m denote the total number of categories, then macroaveraged precision is $(1/m) \sum_{i=1}^m P_i$, macroaveraged recall is $(1/m) \sum_{i=1}^m R_i$, macroaveraged F1 is $(1/m) \sum_{i=1}^m F1_i$, where $F1 = 2PR/(P + R)$.

In order to get the number of base classifiers T mentioned in Section 4 of HAC_EasyEnsemble classification model, we firstly do a series of experiments. In these experiments we set different T for HAC_EasyEnsemble classification model, then we observe the error rate of classification in different T . Figure 2 shows the relationship between the number of base classifiers of HAC_EasyEnsemble classification model and the error rate of classification.

From Figure 2, we can find that when the number of base classifiers equals approximately 10, the error rate of classification tends to be unchanged. Consequently, in the next comparing experiments, the number of base classifiers is set as 10.

Tenfold cross validation is performed on the experiment dataset. For this purpose, the corpus is initially partitioned into tenfold. In each experiment, ninefold data are used to train while onefold data are used to test. Ten experiment results are showed in Figure 3 and the average value of ten experiment results is reported in Table 2.

Figure 3 gets further insight about the comparison of complete link clustering, group average clustering, Ward’s

TABLE 2: The macroaveraged P, R, and F1 score comparison of four schemes.

	Complete Link			HAC_EasyEnsemble Group Average			Ward's Method			EasyEnsemble non-clustering		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
<i>Benign</i>	0.9712	0.9500	0.9605	0.9774	0.9591	0.9682	0.9837	0.9622	0.9728	0.9534	0.9375	0.9454
<i>Malicious</i>	0.9552	0.9533	0.9542	0.9491	0.9567	0.9529	0.9651	0.9667	0.9659	0.9181	0.9300	0.9240
Macro-ave	0.9632	0.9517	0.9574	0.9633	0.9579	0.9605	0.9744	0.9645	0.9694	0.9358	0.9338	0.9347

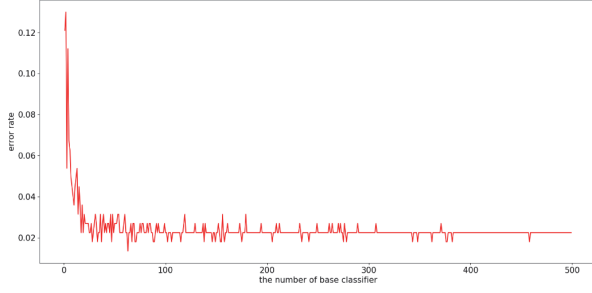


FIGURE 2: The relationship between the number of base classifiers of HAC_EasyEnsemble and the error rate of classification.

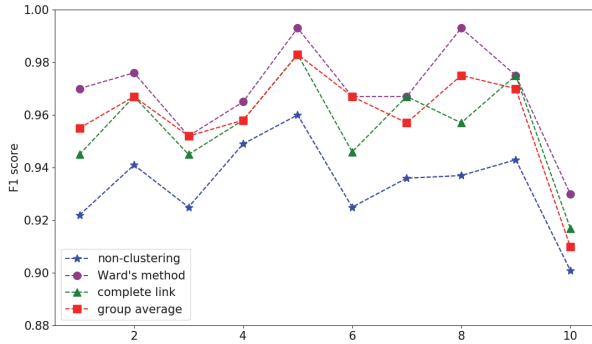


FIGURE 3: The F1 scores comparison of four schemes.

method clustering, and nonclustering with line chart form, from which it can be seen that the scores with clustering are nearly higher than ones with nonclustering overall in each experiment. And Ward's method is the best among of them in performance, while complete link and group average are almost in same level.

Table 2 shows the macroaveraged P, R, and F1 score of each scheme. For example, compared to nonclustering, the macroaverage F1 scores of Ward's method clustering, of group average clustering, and of complete link clustering are approximately improved 3.5%, 2.6%, and 2.3%, respectively, and then we can draw a conclusion that sampling with HAC will be very helpful to improve the performance of classifier.

Finally, to find out whether the HAC_EasyEnsemble is able to show its advantage in different ratio of malicious domains, we do the other 6 experiments to compare the detection performance of HAC_EasyEnsemble. In the 6 experiments, the number of benign domains in training set

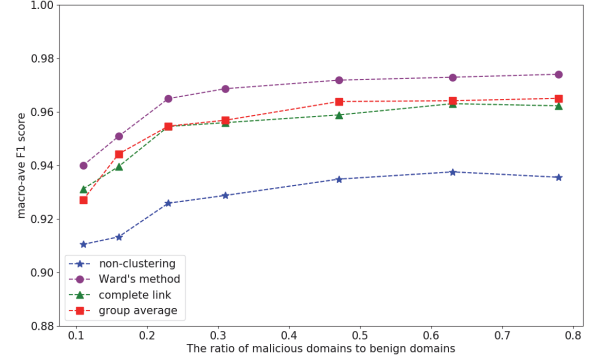


FIGURE 4: The macroaveraged F1 score in different ratio of malicious domains to benign domains.

is 6400, and the number of malicious domains is 700, 1000, 1500, 2000, 4000, and 5000, respectively. Figure 4 shows the experimental results.

From Figure 4, we can see that the detection performance of HAC_EasyEnsemble is almost in line with the previous 3000:6400 (see Figure 3 and Table 2) in any other ratio greater than 1000:6400 ($\approx 16\%$). So, if properly used, HAC_EasyEnsemble can be used to detect malicious domains by learning from imbalanced DNS traffic data.

6. Conclusions

In this paper, we proposed an improved version of EasyEnsemble for detecting malicious domains named HAC_EasyEnsemble, which can effectively deal with the within-class imbalance problem in tandem with the between-class imbalance problem, while EasyEnsemble can only deal with the between-class imbalance problem. The key idea of this improvement is to incorporate HAC into undersampling of EasyEnsemble, and three typical cluster proximity measures which are complete link, group average, and Ward's method are also compared by experiments. Moreover, to profile malicious domains, we construct 12 static lexical features and 4 dynamic DNS resolving features based on passive DNS data from DNSDB. The comparative experiments show that the HAC_EasyEnsemble is superior for the malicious domains detection oriented to imbalanced DNS traffic. And it is worth emphasizing that this novel method is extremely suitable for the tasks in which enough malicious domains cannot be obtained in a limited amount of time.

We believe that HAC_EasyEnsemble is an effective method that can help us to cope with cybercrime. As future work, we plan to construct more discriminative features to profile malicious domains and further to enhance the performance of the HAC_EasyEnsemble algorithm.

Data Availability

The authors declare that the data used in our manuscript can be accessed by the following method. Firstly, the benign domain names are downloaded from alexa.com and the malicious domain names are downloaded from cybercrime-tracker.net, malwaredomains.com, hosts-file.net, etc. And then the DNS resolving records of these domains are obtained by DNSDB API with additional charge.

Disclosure

The funders (Dr. Xue and Dr. Shan) were involved in the manuscript editing, approval, or decision to publish.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was financially supported by National Key R&D Program of China (2016YFB0801304) and Scientific Research Project of Beijing Institute of Technology (2017CX02029).

References

- [1] M. Antonakakis, R. Perdisci, D. Dagon et al., "Building a dynamic reputation system for DNS," *Usenix Security Symposium*, pp. 273–290, 2010.
- [2] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "EXPOSURE: a passive DNS analysis service to detect and report malicious domains," *ACM Transactions on Information and System Security*, vol. 16, no. 4, p. 14, 2014.
- [3] N. Nikolaou, *Cost-Sensitive Boosting: A Unified Approach*, The University of Manchester, 2016.
- [4] S. Wang and X. Yao, "Relationships between diversity of classification ensembles and single-class performance measures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 206–219, 2013.
- [5] S. Maldonado, R. Weber, and F. Famili, "Feature selection for high-dimensional class-imbalanced data sets using support vector machines," *Information Sciences*, vol. 286, pp. 228–246, 2014.
- [6] Y. Peng and J. Yao, "AdaOUBoost: adaptive over-sampling and under-sampling to boost the concept learning in large scale imbalanced data sets," in *Proceedings of the ACM SIGMM International Conference on Multimedia Information Retrieval (MIR '10)*, pp. 111–118, March 2010.
- [7] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 2, pp. 539–550, 2009.
- [8] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, no. 11, pp. 113–141, 2013.
- [9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.
- [10] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," in *Proceedings of the International Conference on Intelligent Computing*, vol. 3644 of *Lecture Notes in Computer Science*, pp. 878–887, Springer, 2005.
- [11] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: adaptive synthetic sampling approach for imbalanced learning," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '08)*, pp. 1322–1328, June 2008.
- [12] M. Stevanovic, J. M. Pedersen, A. D'Alconzo, S. Ruehrup, and A. Berger, "On the ground truth problem of malicious DNS traffic analysis," *Computers & Security*, vol. 55, pp. 142–158, 2015.
- [13] F. Weimer, "Passive DNS replication," in *Proceedings of the FIRST Conference on Computer Security Incident*, pp. 1–13, 2004.
- [14] <https://www.farsightsecurity.com/solutions/dnsdb/>.
- [15] R. Perdisci, I. Corona, and G. Giacinto, "Early detection of malicious flux networks via large-scale passive DNS traffic analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 5, pp. 714–726, 2012.
- [16] C. Zhou, K. Chen, X. Gong, P. Chen, and H. Ma, "Detection of fast-flux domains based on passive DNS analysis," *Acta Scientiarum Naturalium Universitatis Pekinensis*, vol. 52, no. 3, pp. 396–402, 2016.
- [17] [https://en.wikipedia.org/wiki/Entropy_\(information_theory\)](https://en.wikipedia.org/wiki/Entropy_(information_theory)).
- [18] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.
- [19] S. Miyamoto, R. Abe, Y. Endo, and J.-I. Takeshita, "Ward method of hierarchical clustering for non-Euclidean similarity measures," in *Proceedings of the 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR '15)*, pp. 60–63, IEEE, November 2015.
- [20] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2010.

Research Article

OFFDTAN: A New Approach of Offline Dynamic Taint Analysis for Binaries

Xiajing Wang , Rui Ma , Bowen Dou , Zefeng Jian , and Hongzhou Chen 

Beijing Key Laboratory of Software Security Engineering Technology, School of Software, Beijing Institute of Technology, Beijing 100081, China

Correspondence should be addressed to Rui Ma; mary@bit.edu.cn

Received 11 January 2018; Revised 10 March 2018; Accepted 8 April 2018; Published 30 May 2018

Academic Editor: Raymond Choo

Copyright © 2018 Xiajing Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dynamic taint analysis is a powerful technique for tracking the flow of sensitive information. Different approaches have been proposed to accelerate this process in an online or offline manner. Unfortunately, most of these approaches still have performance bottlenecks and thus reduce analytical efficiency. To address this limitation, we present OFFDTAN, a new approach of offline dynamic taint analysis for binaries. OFFDTAN can be described in terms of four stages: dynamic information acquisition, vulnerability modeling, offline analysis, and backtrace analysis. It first records program runtime information and models the stack buffer overflow vulnerabilities and controlled jump vulnerabilities. Then it performs offline analysis and backtrace analysis to locate vulnerabilities. We implement OFFDTAN on the basis of QEMU virtual machine and apply it to off-the-shelf applications. In order to illustrate how our approach works, we first employ a case study. Furthermore, six applications have been verified so as to evaluate our approach. Experimental results demonstrate that our approach is correct and effective. Compared with other offline analysis tools, OFFDTAN has much lower application runtime overhead.

1. Introduction

Software vulnerabilities are the root of various cyber-attacks. Common attacks, such as XSS and buffer overflows, all exploit software vulnerabilities; thus vulnerability analysis attracts extensive research during the past decade. One of the research hotspots is vulnerability analysis for binary program, which can be performed dynamically or statically. In the dynamic analysis, the taint analysis techniques have been studied recently, and numerous taint analysis tools have been implemented and applied extensively to the field of vulnerability analysis for binary program [1].

Taint analysis was first proposed by funnywei at the summit of XCon2003 [2]. The basic idea is to label data originating from or arithmetically derived from untrusted sources, such as network input or user input, as tainted, then keep track of the propagation of tainted data, and further detect whether tainted data is used in dangerous ways. At present, taint analysis could be divided into static taint analysis and dynamic taint analysis on the basis of running state of analysis object. The former keeps track of tainted data

by performing semantic and grammatical analysis of the source code. That has some advantages, like higher path coverage and better analysis efficiency, but there still exist false positives. However, the latter marks and tracks certain data during program runtime. That has better accuracy, but there also exist some problems such as lower path coverage, larger space overhead, and lower analysis efficiency.

To address these limitations, in this paper, we present OFFDTAN, an approach of offline dynamic taint analysis for binary program, implement it on top of QEMU [3] to support fine-grained real-time monitoring for target program, and evaluate it in six realistic applications. In order to illustrate how our approach works, we employ an extended case study. Specifically, in the proposed method, OFFDTAN uses Hook technology to mark taint source and acquires the program runtime information such as register and memory information. Additionally, we summarize the applicable model of stack buffer overflow vulnerabilities and controlled jump vulnerabilities, which can be further used for offline analysis. Moreover, to obtain accurate tainting results, we improve existing taint propagation policy, mainly considering the

effect of the tainted instruction operation on flag registers and associated registers, which makes taint propagation policy more accurate than previous approach. In particular, taint propagation flow graph has been established to backtrack taint data and locate its specific offset within taint source file.

The contributions of this paper can be summarized as follows:

- (i) *Presentation of framework*: we present OFFDTAN, a generic offline dynamic taint analysis framework that uses KVM acceleration on QEMU virtual machine to detect vulnerabilities.
- (ii) *Enhancement of propagation policy*: we enhance taint propagation policy of flag register and related register and summarize two types of specific vulnerability models applicable to this method.
- (iii) *Construction of propagation flow graph*: we propose the construction method of taint propagation flow graph to backtrack taint source, which can precisely locate the specific offset of taint data.
- (iv) *Evaluation of framework*: we evaluate OFFDTAN by applying it to large off-the-shelf applications such as FeiQ and Microsoft Word 2010. The experimental results show that our approach is effective and has better performance.

The remainder of this paper is organized as follows. Related work was discussed in Section 2. Section 3 presents the details of proposed approach. Section 4 discusses the experimental results and validation, and our conclusions were finally proposed in Section 5.

2. Related Work

Dynamic taint analysis can be performed online or offline. Online analysis handles taint propagation during the program execution, while offline analysis first records the trace of program execution and then performs the taint analysis by replaying program.

Online analysis usually leverages instrumentation technology to monitor the taint propagation. Common instrumentation tools, such as Valgrind [4], Pin [5], and Dynamo [6], have been used extensively to implement most of online analysis tools. James Newsome et al. released TaintCheck [7] on the basis of Valgrind, which provides taint analysis for data flow and enables the detection of buffer overflow vulnerabilities. However, this tool needs larger space overhead and ignores the analysis for control flow. Considering the effect of control flow, Clause et al. propose Dytan [8], which achieves the analysis for control flow, but this tool still presents the limitation of time overhead. LIFT [9] is developed by Qin et al. based on StarDBT, which sharply shortens the duration of taint analysis by screening for unnecessary data flow information whereas the problem of memory consumption has not been properly addressed.

Due to the rise of symbolic execution, some researchers attempt to provide a combination method of dynamic taint analysis and symbolic execution, such as DTA++ [10], BitBlaze [11], and DECAF [12], which can improve the path

coverage of dynamic taint analysis. Lai et al. [13] mark each byte of external input data to perform fine-grained taint analysis, which improves the granularity of dynamic taint analysis. Wang et al. [14] propose a method to bypass the checksum mechanism, which combines with symbolic execution and fine-grained dynamic taint analysis, to develop TaintScope. Zhuge et al. [15] present a method of type-based dynamic taint analysis, according to the type information of instructions and functions, which provides better semantic support. In addition, this method presents the combination of taint analysis and symbolic execution at variable level. Although above methods have improved the performance of online analysis, many problems are still not fundamentally solved, especially the limitation of high runtime overhead.

To address these limitations, several researchers propose the method of offline analysis, which is attractive at present. Jee et al. propose a dynamic taint analysis method based on shadow memory, which separates taint analysis from program execution and develops ShadowReplica [16]. Dan Caselden et al. introduce a hybrid information and control-flow graph (HI-CFG) and give algorithm to infer it from an instruction-level trace. Then they use the Tracecap tool of BitBlaze to record instruction traces [17]. Manolis Stamatogiannakis et al. [18] leverage full-system execution trace logging and replaying to decouple analysis from the original execution. Shi et al. [19] propose a combination of coarse-grained and fine-grained dynamic taint analysis (DTA) method. It executes online coarse-grained DTA to screen out effective instruction and then uses offline fine-grained DTA to calculate taint information. Wang et al. [20] introduce the propagation policy of multi-tainted label and implement the prototype system FlowWalker. Their taint propagation strategy makes a further support for the extended instruction set of MMX/SSE family. Ma et al. [21] present a taint analysis method based on trace offline indices which are byte-grained and utilize taint tags. Their approach fixes the problem of taint loss which resulted from just-in-time translation first time. Ming et al. [22] propose a full-featured offline taint analysis tool StraightTaint, which completely decouples the program execution and taint analysis, resulting in much lower execution slowdown. Dolan-Gavitt et al. [23] present a full-system analysis tool PANDA that is based on QEMU emulator and has the ability to record and replay executions.

The above-mentioned offline analysis methods alleviate the problem of lower analysis efficiency to a certain extent. However, most analysis tools or methods are running on the same operating system as the target program and thus cannot eliminate the impact of vulnerability analysis tool on the target program. PANDA eliminates above impact, but its performance is a bit slow. In particular, the propagation policies of these methods are still not complete.

To address the first problem, we managed to use QEMU virtual machine that supports KVM acceleration to create a simulated computer environment isolated from the host and precede fine-grained observation for relevant target program in the client. In addition, this paper develops appropriate propagation policy and vulnerability checking strategy to improve the accuracy of analysis. In the aspect of propagation policy, we focus on improving the taint update policy of flag

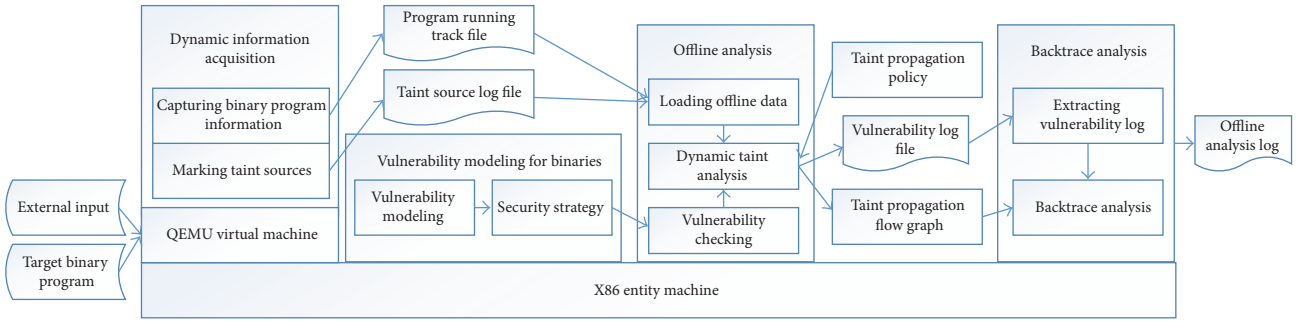


FIGURE 1: The OFFDTAN framework.

register and related register. To check vulnerabilities, this paper summarizes two types of specific vulnerability models applicable to this method and establishes the vulnerability checking strategy by studying the cause of released vulnerabilities. Moreover, the approach of backtrace analysis has been presented to locate taint data's specific offset within taint source file. This method further reduces the proportion of manual analysis and improves the efficiency of vulnerability analysis.

3. OFFDTAN

In this section, we first outline the framework of OFFDTAN and then detail each stage.

3.1. General Framework. This paper proposes OFFDTAN, an approach of offline dynamic taint analysis for binary program. We employ KVM acceleration on QEMU to implement OFFDTAN, because it is isolated from the operating system of the host. In order to better describe this method, it can be divided into the following four stages: dynamic information acquisition, vulnerability modeling, offline analysis, and backtrace analysis, as shown in Figure 1.

More precisely, dynamic information acquisition stage is used for dynamically recording the trace of program execution and taint source log, which can be used for offline analysis stage. The basic condition of this stage is that QEMU simulates the CPU and memory state. Vulnerability modeling stage establishes the vulnerability model by summarizing existing vulnerability characteristics and then generates the vulnerability checking strategy. Offline analysis stage loads the trace file of program execution and taint source log file to perform offline analysis on the basis of taint propagation policy and vulnerability checking strategy. Additionally, offline analysis stage builds taint propagation flow graph and generates vulnerability log file. Finally, backtrace analysis stage takes vulnerability log file and taint propagation flow graph as input to backtrack taint data to locate vulnerability.

3.2. Dynamic Information Acquisition. This stage is mainly to extract the runtime information of target program and mark the taint source. Moreover, the generated trace file of program execution and taint source file are used for offline analysis. The key processes of this stage are the capture of program runtime information and the marking of taint sources.

3.2.1. Capturing Program Runtime Information. The capture of program runtime information is mainly to obtain four aspects of information, namely, the information of CPU running state, memory information, process information, and the trace file of program execution.

(1) *Acquiring the Information of CPU Running State.* The indispensable CPU state information mainly includes general register, instruction register, flag register, segment register, control register, and the internal variable of QEMU. This information can be obtained by capturing the current CPU pointer.

(2) *Acquiring Memory Information.* Clearly, there are four types of memory addresses in the QEMU, namely, client virtual address, client physical address, host virtual address, and host physical address.

Reading and writing the QEMU memory should actually be working in the physical address of host. Therefore, the key point is performing the address translation from client address to host address to acquire current memory information.

(3) *Acquiring Process Information.* The processes store much program information, and offline analysis is accompanied by the virtual replaying of program which requires a large number of data types and semantic information. Therefore, the acquisition of process information is the core during the capture of program runtime information.

In the Windows operating system, data structures associated with the process include five structures: KPROCESS, EPROCESS, KTHREAD, KPRCB, and KAPC_STATE. That almost describes all the information about processes and threads. To obtain current process, our method first needs the memory address of KPRCB, which can be acquired by the reverse analysis of the Windows kernel. On the basis of KPRCB, our approach can further calculate KTHREAD, KAPC_STATE, KPROCESS, and EPROCESS, respectively, to acquire process information.

(4) *Generating the Trace File of Program Execution.* Actually, the generating process of trace file is to integrate captured program runtime information in accordance with the organization and management way of kernel data structure. The trace file mainly includes obtained CPU running state

information, memory information, and process information. The generated trace file of program execution records instruction running number, current process number, thread number, instruction, operand, register, and other information. Specifically, it can be divided into the following five steps:

- (a) Through CPU running state information to obtain instructions, registers, and other information.
- (b) Taking instruction information and relevant register information as input, the instruction opcode and operand can be obtained through memory information.
- (c) Obtaining process-related information by acquired register information, memory information, and process information.
- (d) The acquired information is integrated into the trace file of program execution, including register information, instruction opcode, instruction operand, and process information.
- (e) Generate the trace file of program execution.

3.2.2. Marking Taint Sources. Taint markings are prerequisite for taint propagation, and the way of taint markings and taint operation recordings has a great influence on the efficiency of dynamic taint analysis. In the dynamic taint analysis, the external data, such as file input or network input, are generally marked as taint data. In order to mark taint source, the corresponding system services that are responsible for reading external data need to be monitored. In the Windows operating system, some system services, such as *NtReadFile*, *NtCreateFile*, *NtWriteFile*, *NtClose*, *WSARecv*, and *recv*, are responsible for file input and network input. Thus, in this paper, OFFDTAN marks taint sources of network input and file input by coding the Hook function for above system services to monitor the parameters and return value of those system services. As a result, our method obtains and records the offset of taint data in the taint source file, the length of taint data, and the head address of memory where taint data is stored.

3.3. Vulnerability Modeling. This paper focuses on the research of stack buffer overflow vulnerabilities and controlled jump vulnerabilities, so corresponding models have been established according to the characteristic of offline analysis.

3.3.1. Stack Buffer Overflow Vulnerabilities Model. Since the *strcpy*, *memcpy*, and other string manipulation functions are optimized in the process of compiling, this paper summarizes the assembly code of string library functions to analyze the program path that could trigger buffer overflows. Moreover, this paper establishes the dependency among the execution paths and combines offline analysis with modeling buffer overflow vulnerabilities.

The stack buffer overflow vulnerabilities model of this paper mainly concerns whether the operand of instruction *rep movsd* can override function return address or *EBP* to further affect the control flow of program.

In order to better represent this model, firstly, we explain various terms used in below definition. The symbol *EDI* represents destination address operated by crucial instruction. The symbols *EBP* and *ESP* denote stack base address and stack top pointer, respectively, and symbol *RA* represents function return address. The program counter register is denoted as *ECX*. Then the following definitions are given.

Definition 1. The symbol *I* denotes whether there is a crucial instruction, $I \in \{0, 1\}$. The value of *I* is 1 if and only if instruction *rep movsd* exists in the trace of program execution.

Definition 2. The symbol *F* denotes whether the instruction *I* writes on stack memory, $F \in \{0, 1\}$. The value of *F* is 1 if and only if $ESP < EDI < EBP$ (or $ESP < EDI < RA$), which can determine that instruction *I* is operating the stack memory.

Definition 3. The symbol *P* denotes the safe distance of stack buffer, $P \in [0, 2097152]$. The safe distance is defined as $P = EBP - EDI$ or $P = RA - EDI$.

Definition 4. The symbol *D* denotes whether *EBP* (or *RA*) is overwritten, $D \in \{0, 1\}$. In this case, the length of source data operated by instruction *I* is expressed as *L*, which is $4 * ECX$. The value of *D* is 1 if and only if $L > P$.

Definition 5. The symbol *E* represents whether *ECX* is tainted, $E \in \{0, 1\}$. The value of *E* is 1 if and only if *ECX* is tainted.

Definition 6. The symbol *C* denotes whether *ECX* has a comparison instruction before instruction *I*, $C \in \{0, 1\}$. The value of *C* is 1 if and only if there is a comparison instruction to *ECX* before instruction *I*.

Definition 7. The symbol *T* denotes whether the parameters compared with *ECX* are tainted, $T \in \{0, 1\}$. The value of *T* is 1 if and only if the parameters are tainted.

Definition 8. The symbol *V* expresses whether the program has stack buffer overflows and *V* is byte type. The values of *I*, *F*, *D*, *E*, *C*, *T* are the lower six bits of *V* in turn. For example, the value of *V* is 0011010 if the values of *I*, *F*, *D*, *E*, *C*, *T* are 1, 1, 0, 1, 0, 1, respectively.

In summary, this model considers two modes of stack buffer overflows primarily.

- (1) *The triggered mode of stack buffer vulnerabilities:* in this mode, $L > P$, i.e., $D = 1$; thus the data written to stack could overwrite *EBP* or *RA*. The value of *V* is 0011XXXX.
- (2) *The nontriggered mode of stack buffer vulnerabilities:* in this mode, *ECX* is tainted, thus triggering potential stack buffer vulnerabilities. This model is divided into two cases on the basis of the value of *C*. (i) When $C = 0$, the value of *V* is 0011010X. (ii) When $C = 1$ and $T = 1$, the value of *V* is 00110111.

3.3.2. Controlled Jump Vulnerabilities Model. Controlled jump usually refers to the fact that taint data is used for return address, function pointer, and destination address, etc. and makes the program hijacked to the shellcode code, which causes the program to be controlled by the attacker. First, this model locates this type of jump instruction by the sequence of instruction during the program execution. Second, according to offline analysis, the coincident conditions that triggered vulnerabilities can be concluded. For any of jump instructions, it can trigger vulnerabilities if and only if its destination address is tainted.

This model mainly considers three types of jump modes.

Mode 9 (call/jump register mode). In this mode, the operand of the instruction is only one register. If the register is tainted, this instruction operation is tainted.

Mode 10 (call/jump [register + offset] mode). In this mode, the operand adds the offset on the basis of register. If the operand or register of instruction is tainted, this instruction operation is tainted.

Mode 11 (Ret/Retf mode). This instruction is used for function return, and usually ESP will be assigned to EIP (the instruction address of CPU execution) before return. If the data pointed by ESP is tainted, this instruction operation is tainted.

3.4. Offline Dynamic Taint Analysis. Due to the logic complexity of internal system call, the traditional dynamic taint analysis has the disadvantages of lower analysis efficiency and higher runtime overhead while analyzing large-scale programs. OFFDTAN presents an approach of offline dynamic taint propagation, which combines instruction and the system call, and separates dynamic analysis from program execution. More precisely, our approach leverages the trace file of program execution and taint log file to implement the offline analysis through virtually replaying program. That ultimately improves the accuracy and efficiency of dynamic taint analysis.

3.4.1. Overview of Offline Dynamic Taint Analysis. The offline dynamic taint analysis can be mainly divided into three key points.

- (1) *Program virtual replaying*: by loading instructions and contexts recorded by the trace file of program execution, it is possible to simulate program execution according to the order of instruction.
- (2) *Dynamic taint analysis*: by loading taint source log file, it performs taint analysis during the virtual replaying. This analysis is guided by propagation policy.
- (3) *Program vulnerability analysis*: it checks the program vulnerabilities in the process of dynamic taint analysis, which is guided by security policy.

The (1) is mainly to simulate CPU's running process, and in this context, the semantics of different instructions are parsed to implement program virtual execution. Specifically,

OFFDTAN first loads the trace file of program execution, reads line by line, and stores the necessary information such as instruction and register information. With the help of *udis86*, a third-party disassembly engine, read instruction information then needs to be disassembled. At the same time, some relevant program runtime information will be copied to the simulated CPU. Finally, we perform semantic and syntactic analysis for each instruction within the context of simulated CPU.

Since (3) is described in Section 3.3, this section will focus on dynamic taint analysis, which mainly includes taint data recording and taint propagation. The former records the taint state of memory and register during the taint propagation. The latter primarily provides proper propagation policy for taint analysis. According to the taint data recording and corresponding taint propagation, this stage ultimately generates the taint propagation flow graph, which provides data input for backtrack analysis and forward analysis.

3.4.2. Taint Data Recording. Taint data recording primarily includes the introduction of taint source and the state update of taint data. The introduction of taint source is the first step in taint analysis. Unlike traditional dynamic taint analysis, offline analysis introduces taint source by analyzing taint source log file. Specifically, according to the instruction number of virtual replaying, the taint source information with the same number would be loaded into the taint state space. Next, this method uses shadow memory to store and maintain taint state of memory address and register in taint state space. It updates taint state in real time based on taint propagation policy during the dynamic taint analysis.

Taint state space records the taint state of memory address and register, as shown in Figure 2. However, most of previous approaches have not recorded taint operation; thus taint propagation path cannot be stored. In order to support the backtrack analysis of taint data, taint operation needs to be recorded according to different storage carrier.

- (1) *Taint operation recording for memory*: the tainted memory of each byte is stored in the form of data structure *TM*, which is arrayed in a form of linked list based on the order of taint operation when taint is propagated. Data structure *TM* mainly contains memory address, taint state, memory address of taint source, and pointer to the node of taint propagation flow graph, as shown in the upper part of Figure 2.
- (2) *Taint operation recording for register*: by learning from the value of register enumerated variable *ud_type*, which is the third-party disassembler *udis86*, serialized storage management would be taken for the taint state information of register. In the *udis86*, the value of register enumerated has 141 types; thus this paper defines the structure array *TRArray*[141] of *TR* to store the taint state of register, where array index corresponds to the specific register. The data structure *TR* mainly contains taint state, pointer to the node of taint propagation flow graph, and taint source operation, as shown in the lower part of Figure 2.

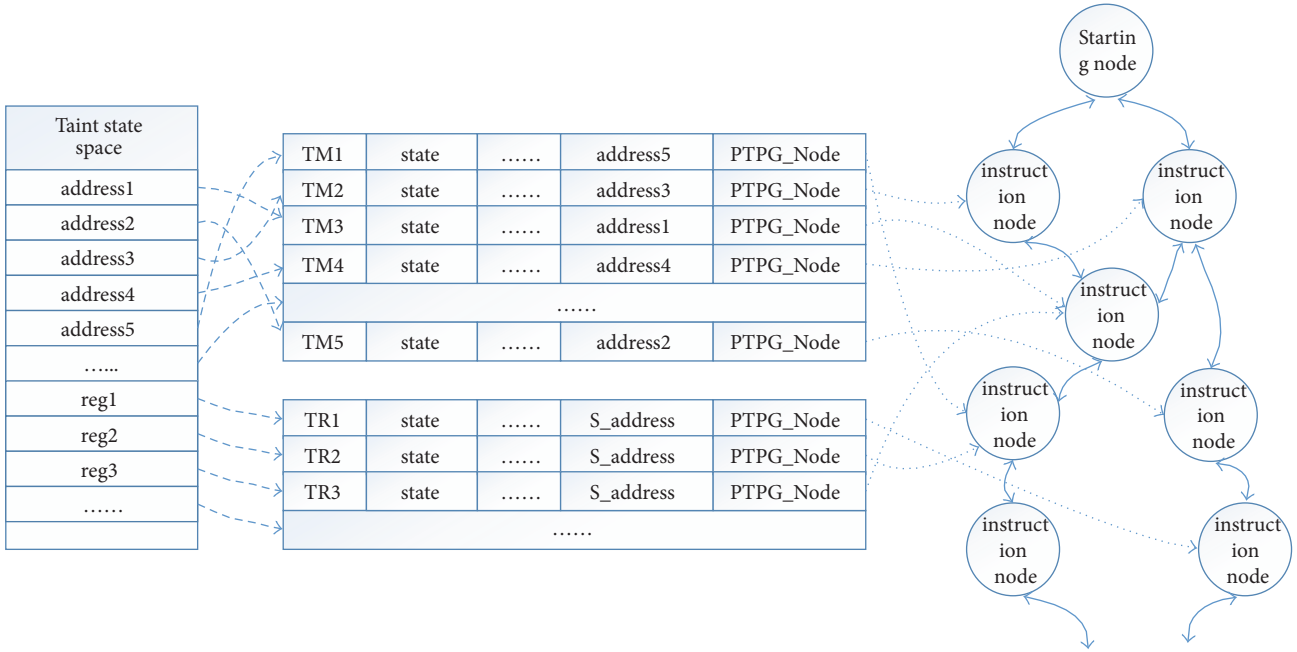


FIGURE 2: Taint state space and corresponding taint propagation flow graph.

3.4.3. Taint Propagation Policy. Taint propagation is the key to taint analysis. Furthermore, a complete and proper propagation policy, which describes how taint data should be propagated during program replaying, is crucial for taint propagation. In order to develop taint propagation policy, we first extract and analyze common instruction set in the complex instruction space and instruction semantics through many program tests. Then we study six categories of instruction, such as data transfer class, operation class, shifting instruction class, string operation class, and control transfer class. According to its instruction semantics, source operands, destination operands, and addressing mode, taint propagation policy is developed. Finally, we set query interface, tainting interface, and sanitization interface, which is used for updating the taint state in real time.

A corresponding taint propagation policy is made for above six instruction categories.

(1) Data transfer class: *MOV, PUSH, XCHG, etc.*

- (i) If one of the source operands is tainted, the destination operands will be tainted.
- (ii) If source operands are untainted, the destination operands need to be sanitized.

(2) Operation class: *ADD, ADC, AND, XOR, etc.*

- (i) If any of the operands of an instruction is tainted, this result of operation is tainted and destination operands are also marked as tainted.
- (ii) If the conditional flag of processor is affected by tainted operand, the affected conditional flag is also marked as taint data.

- (iii) For instruction *XOR*, if the source operand and destination operand have identical register or memory address, the operand of instruction needs to be sanitized.

(3) Shifting instructions: *SAL, SHL, SHLD, RCL, etc.*

- (i) If the arbitrary operands of this instruction are tainted, this result of operation is tainted and destination operands are also marked as taint data.
- (ii) If the conditional flag of the processor is affected by tainted operand during the shifting, the affected conditional flag is also tainted.

(4) String operation class: *MOVS, LOAD, STOS, etc.*

- (i) If the source operands of this instruction are tainted, the destination operands are tainted.
- (ii) If an instruction with *REP* exists, the *ECX* needs to be sanitized after this instruction has been executed.

(5) Control transfer class: *JMP, CALL, RET, etc.*

- (i) If these instruction operands are tainted, the *EIP* is tainted.
- (ii) If instruction operands are indirect addressing, the *EIP* is marked as taint data as long as the internal register of operand is tainted.

(6) Others: *CLD, CLC, STC, CBW, etc.*

- (i) For these instructions of clear direction flag register, such as *CLD* and *CLC*, the corresponding flag registers need to be sanitized.
- (ii) The instruction *STC* does not need to be processed.
- (iii) For extended instructions such as *CBW*, the extended high register is marked as taint data if the low register is tainted.

In summary, our propagation policy focuses on improving the update ways for the taint state of flag register and associated register. For example, when the taint state of accumulator *AX* changes, it is necessary to modify the taint state of associated registers, such as the *AL* register, *AH* register, and *EAX* register.

3.4.4. Constructing Taint Propagation Flow Graph. Dynamic taint analysis is the fundamental of the constructing of taint propagation flow graph. During the offline dynamic taint analysis, taint operation would be recorded to provide the data dependency between instruction operands and function parameters for building taint propagation flow graph. More precisely, OFFDTAN first loads the trace file of program execution and analyzes those instructions during the program virtual replaying. According to the propagation policy and the taint state space of operand, OFFDTAN then determines the effect of each instruction on the taint state space. If there is a change in the taint state space, OFFDTAN will record this trace of taint operation and point to the corresponding nodes in the taint propagation flow graph.

Dynamic taint analysis can achieve the forward analysis and backtrace analysis to taint data by using taint propagation flow graph. The traditional taint propagation flow graph is composed of instruction nodes and directed edges. In this method, the taint state recording structure can reflect the distribution of taint data in the taint state space. In order to locate taint source, our method employs a bidirectional edge as connection edge and adds a serial number for each node in the taint propagation flow graph to distinguish the upper and lower relations between the nodes.

In a word, a taint propagation flow graph is composed of instruction nodes and bidirectional edges, as shown in Figure 2, which also demonstrates the relationship between taint state space and the node of taint propagation flow graph.

The node, which is composed a triple, has two types: the starting node and the intermediate node. Taint source is taken as starting node, and intermediate node is established by determining whether the direct or indirect operand of relevant operation instruction is tainted. If it is, a node is generated for this instruction. In particular, a pointer of the taint state recording structure corresponding to tainted operand points to the node of taint propagation flow graph. Then intermediate nodes would be created sequentially until the end of program execution.

The edges are used for connecting these nodes according to the dependencies between the operands of instruction nodes. Based on the real-time mapping relationship between the recording structure of taint state and the corresponding node of taint propagation flow graph, our approach can find

the corresponding node from the recording structure of taint state and then connect the current node and the queried node through bidirectional edge.

3.5. Backtrace Analysis for Taint Data. The backtrace analysis of taint data can be used for locating taint source. In addition, the backtrace analysis has a significant effect on improving vulnerability analysis and assisting manual analysis.

This paper first takes the taint data in the recorded program vulnerabilities as the source of backtrace analysis. According to its address, the corresponding node of taint propagation flow graph can be found from taint state space. Then, according to the taint source information of this node, the prior node of taint propagation flow graph can be further inquired from taint state space. Finally, our approach successively backtracks taint propagation flow graph until it located to the source of taint information. In a word, the process of backtrace analysis is to continuously find a feasible path from program vulnerabilities to taint source.

Backtrace analysis could be formally described as follows. Firstly, taint source information can be defined as S , which is the n -tuple of all the set of taint source information and is denoted as $S = (s_1, s_2, \dots, s_n)$. We assume that any of the propagation paths can be represented by the elements in the n -tuple $V = (v_1, v_2, \dots, v_n)$, where v_i ($i \in [0, n]$) is the node of taint propagation flow graph and n is the number of nodes. The data structure of v_i is $\langle src_i, ins_i, dst_i \rangle$, where src_i represents its source operand, ins_i is the current operation instruction, and dst_i refers to the destination operand. The goal of backtrace analysis is to obtain the src_i in the taint operation v_i of program vulnerabilities. Then propagation path i -tuple $V = (v_1, v_2, \dots, v_i)$ will be traversed reversely, so that the src_i in v_i belongs to any elements of S ; i.e., $v_i -> src_i \in S$.

4. Evaluation

This section describes the verification process of proposed approach. We first analyze the implementation step by step through a small case and illustrate the correctness and feasibility of our method. Then, we apply it to six large applications, such as FeiQ 2.5 and Word 2010, to further verify the correctness and effectiveness of proposed approach. Finally, we set a comparative experiment to evaluate OFFDTAN's performance.

4.1. Experimental Setup. Our approach is implemented on QEMU 1.2.0 that supports KVM acceleration. Host hardware is Dell 8900 with Intel Core i7-4770 processor and 32 GB memory, and host operating system is 64-bit CentOS 7. Client hardware is the x86 architecture simulated by QEMU with a virtual CPU and 512 M memory, and client operating system is 32-bit Windows 7.

4.2. Case Study

4.2.1. Case Design. In this case analysis, we manually construct C program with vulnerability, which is *test.cpp*, and then generate *test.exe* by compiling it. Algorithm 1 is the

```

(1) void myMemcpy(char si[], int count){
(2)     char dest[10];
(3)     memcpy(dest, si, count);           //building program
(4) }                                     //vulnerability point
(5) int main(int argc, char *argv[]){
(6)     HANDLE hOpenFile = (HANDLE)CreateFile(argv[1], //reading taint
        GENERIC_READ,           //source file test.txt
        FILE_SHARE_READ, NULL,
        OPEN_EXISTING, NULL, NULL);

(7)     .....
(8)     count = readCount(buf);           //reading the count
(9)     newBuf = readNewBuf(buf);         //reading the string
(10)    myMemcpy(newBuf, count);
(11)    return 0
(12) }

```

ALGORITHM 1: Source code of *test.cpp*.

source file of target program *test.exe*. In this program, the *test.txt* is taken as input file, which is taint source file, and its contents are "16aaaaaaaa..aaaa". From the *test.cpp* and *test.txt*, we can draw a conclusion that the test program reads "16" in taint source file as the value of *count* in the *memcpy(dest, si, count)*, and reads "aaaaaaaa..aaaa" as the value of string *newBuf*. Moreover, *count* is the counter that tracks how many times function *memcpy* writes data to destination address *dest*, where *dest* is an array whose size is 10. Therefore, this case must cause an overflows.

We first use debugging tool *Ollydbg* to analyze *test.exe* in the local Windows 7 operating system, and then the address of stack buffer can be obtained, which is *0x401046*. At this time, because the value of *ECX* is too large, the function return address *0x12FE30* is covered by taint data when the *rep movsd* is executed. The taint data that covers the function return address is "aaaa".

4.2.2. Case Analysis. This part will analyze this case in detail in accordance with the implementation process of OFFDTAN.

(1) *Capturing Program Runtime Information.* This case executes target program *test.exe* to acquire the program dynamic information. Specifically, the data structure *TraceNode* is defined to integrate the CPU, memory, and process information. The trace file of program execution *record.log* ultimately will be generated. Figure 3 demonstrates the information contained in the trace file of program execution, mainly including instruction running number, thread number, instruction register, assembly instruction, and general register. These will be used for program virtual replaying.

(2) *Recording Taint Sources.* This method writes Hook function for some services that is related to file input and network input (such as *NtCreateFile*, *NtReadFile*, and *recv*) to mark taint source. At last, the taint source log file *taintsource.log* will be generated. We take a taint source information as an example to illustrate the information contained in the taint source log file, as shown in Table 1. This information will

Records Count : 178301696 the total number of instructions

id : 1, tid : 2068, eip : 0x4010d8, asm : push esi instruction number
thread number instruction register
eax : 0x1, ecx : 0x76389754, edx : 0x775570b4, ebx : 0x7ffda000
esp : 0x12fe3c, ebp : 0x0, edi : 0x0, esi : 0x38
op[0] : 0x38, op[1] : 0x0, op[2] : 0x0 general register
the value of operand

id : 2, tid : 2068, eip : 0x4010d9, asm : call dword [0x406000]
eax : 0x1, ecx : 0x76389754, edx : 0x775570b4, ebx : 0x7ffda000
esp : 0x12fe38, ebp : 0x0, edi : 0x0, esi : 0x38
op[0] : 0x7638ca7c, op[1] : 0x0, op[2] : 0x0

FIGURE 3: The trace file of program execution.

TABLE 1: The content of taint source log file.

Name	Content
No.	1
The head address of taint data in memory	0X12FE54
The length of taint data	200
The offset of taint data in taint source file	0

be used for marking taint source in the process of offline dynamic taint analysis.

(3) *Offline Dynamic Taint Analysis.* This step takes the taint source log file *taintsource.log* and the trace file of program execution *record.log* as input to perform offline dynamic taint analysis in accordance with the taint propagation policy and program vulnerability checking strategy. It will ultimately generate program vulnerability log file *sink.log* (Figure 4(a)) and corresponding taint propagation flow graph (Figure 4(b)).

As can be seen from Figure 4(a), our method detects five program vulnerabilities. The information of each vulnerability includes instruction number, the memory or register address of taint data, and the length of taint data. All these will be used for the backtrace analysis of taint data.

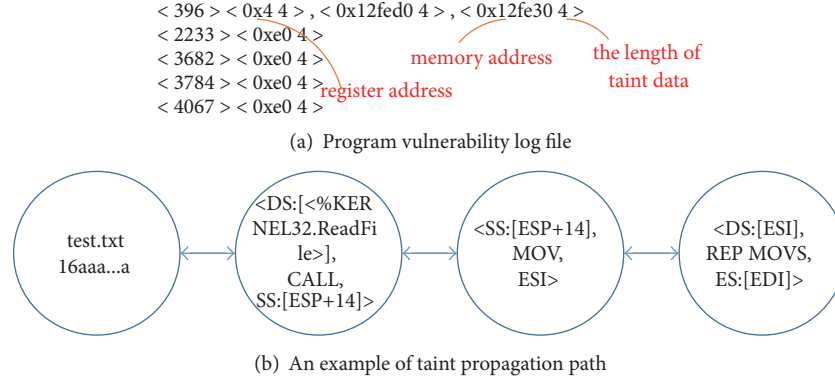


FIGURE 4: The results of offline dynamic taint analysis.

```

< 396 > < 0x4 4 > , < 0x12fed0 4 > , < 0x12fe30 4 >
< 0x401046 rep movsd 0 Function return address is overwritten and stack overflow occurs, the address of ECX is tainted, the
address of ESI is tainted. The address of ECX is 0x4; the address of ESI is 0x12fed0; the function return address is 0x12fe30 >
< 0xe0000000 , 0xe0000001 > | < 0xe0000004 , 0xe0000007 > | < 0xe0000000 , 0xe0000001 > < 0xe0000010 , 0xe0000013 >

```

the offset of taint data in ECX the offset of taint data in ESI the offset of taint data that overrides function return address

FIGURE 5: Offline analysis log file.

```

00000010h : 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 ; aaaaaaaaaaaaaaaaaa
00000020h : 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 ; aaaaaaaaaaaaaaaaaa
00000030h : 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 ; aaaaaaaaaaaaaaaaaa

```

taint data that overrides function return address

FIGURE 6: Taint source file.

(4) *Backtrace Analysis*. This step uses program vulnerability log file as input and then backtracks taint data according to the corresponding relationship between taint propagation flow graph and taint state space. In Figure 4(b), we take the first program vulnerability as an example to show the corresponding taint propagation path V and illustrate how to backtrack taint source. Each intermediate node has three elements, namely, src_i , ins_i , and dst_i . We start with the src_i of last node, which is $DS : [ESI]$, and inquire it from taint state space to get the prior node in which the dst_i is ESI . Then, the src_i of obtained node is inquired, which is $SS : [ESP + 14]$, to acquire the prior node related to $SS : [ESP + 14]$. Next, we successively traverse acquired node to inquire its src_i until it belongs to taint source S , and $DS : [< \%KERNEL32.ReadFile >]$ is the entry address of function *ReadFile*. The memory block $SS : [ESP + 14]$ is the buffer address read by function *ReadFile*, and this memory block stores the string loaded from taint source file *test.txt*. This step will ultimately generate the offline analysis log file *result.log*, as shown in Figure 5.

As can be seen from Figure 5, each offline analysis log is corresponding to the program vulnerability log file, indicating that our method backtracks each of program vulnerabilities. Taking the first record in Figure 5 as an example, the address *0x12FE30*, which is detected by proposed method, is covered by taint data when the instruction *rep movs* is executed. This is consistent with the result of *Olydbg* analysis. In addition, the result of backtrace analysis is that the offset

of taint data within taint source file, which covers return address, is *0x10 ~ 0x13*. The taint source file (see Figure 6) shows that the “aaaa” covers return address, which also coincides with the analysis in Figure 5.

According to above case analysis, we elaborate the analysis procedure of our approach and verify the correctness and feasibility of this method as well.

4.3. *Off-The-Shelf Applications*. To evaluate our method’s ability to detect vulnerabilities for real applications, we take *FeiQ* and *Microsoft Office Word* as an example and analyze the vulnerability information of these two projects in detail.

4.3.1. *Experimental Objective*. This experiment is mainly to verify the correctness and effectiveness of taint source marking, the vulnerability model, and the final experimental results of this method.

4.3.2. *Experimental Design*. The design of this experiment is to use realistic application as test object. Through analyzing application program, the correctness and effectiveness of proposed method can be verified.

Two programs have been selected first. *FeiQ 2.5* is a network communication program and widely used in the enterprise. *Microsoft Office Word 2010* is a word processing program, which is the mainstream of current word processing software.

TABLE 3: Program description and evaluation results.

Program	Version	Vulnerability	Crash Address	Crash Instruction	Offset of Taint
Adobe Reader	9.3.4	CVE-2010-2883	0x0803DDAB	call strcat	0x12C
Microsoft Office Excel	2003	CVE-2011-0104	0x300DE834	rep movs	0x300
Firefox	3.6.16	CVE-2011-0073	0x1046659B	call dword ptr [ECX + 70h]	0x4A
Microsoft Office Word	2003	CVE-2012-0158	0x275C8A0A	rep movs	0xA0F

TABLE 4: The overhead of OFFDTAN and PANDA when running FeiQ and Word.

Applications	PANDA				OFFDTAN			
	Record Time (sec.)	Replay Time (sec.)	CPU%	Mem%	Record Time (sec.)	Replay Time (sec.)	CPU%	Mem%
FeiQ	53.67	86.09	17.4%	35.3%	47.04	69.58	12.4%	34.5%
Word 2010	74.02	127.95	18.3%	36.2%	65.92	94.81	14.6%	34.9%

As is depicted in Figure 8(a), OFFDTAN successfully marks the POC file read by Microsoft Office Word 2010. In Figure 8(b), OFFDTAN analyzes the vulnerability address of Microsoft Office Word 2010, which is *0x66E9195D*, and the corresponding assembly instructions is *call dword[EAX + 4]*. It also found that the reason of program controlled jump is that the memory address of *[0x275A48E8]* is tainted, which is consistent with the analysis result provided by CVE previously. In addition, this method uses backtrace analysis to analyze the taint data in *[0x275A48E8]* derived from the POC file of *CVE-2014-1761* whose offset is *0x1D3F*, *0x1D4A*, *0x1D54*, and from *0x1E18* to *0x1E1A*. Therefore, we verify the correctness and effectiveness of marking taint source and controlled jump model proposed by OFFDTAN.

4.4. Further Verification. We proceed to evaluate our approach on four other applications to further verify the correctness and effectiveness of OFFDTAN. In addition, we compared the performance overhead of this method with other tools to verify the analysis efficiency of proposed method.

4.4.1. Effectiveness. We analyze the security vulnerabilities of four programs: Adobe Reader, Excel 2003, Firefox, and Word 2003. For each of these projects, our method generates the valid description of vulnerability analysis. Table 3 provides an overview of these results, showing the program and its version, crash address, crash instruction, and the offset of taint source. Moreover, the vulnerabilities are shown and denoted by their CVE-identifiers.

In Table 3, the crash address and crash instruction are the memory address and corresponding instruction that crashed during program execution, respectively. The offset of taint refers to the offset address of taint data in taint source file.

We take the first vulnerability as an example to briefly illustrate analysis result. We use the POC file of *CVE-2010-2883* to analyze Adobe Reader and generate taint source log file. The result shows that the program crashed when the instruction *call strcat* at *0x0803DDAB* is called, and the

reason for crash is that the string length of field *uniqueName* is not judged, causing stack overflows, which is consistent with the result of CVE. Moreover, our method uses backtrace analysis to obtain the offset of taint data in taint source file, which is *0x12C*.

4.4.2. Performance. In this experiment, we describe the performance overheads introduced in two real applications by our approach to taint analysis and compare them with PANDA, which is another state-of-the-art whole-system offline taint analysis tool built on QEMU 2.1.0. Similar to OFFDTAN, PANDA also has the ability to record and replay executions. We evaluate both tools on four items that represent time and space overheads.

The comparison results of performance overheads are shown in Table 4. The record time refers to the time of information recording during program execution, and replay time includes program simulation replay time and taint analysis time. The CPU% and Mem% mean the percentage of CPU and memory used during program execution, respectively.

As shown in Table 4, in addition to memory usage, our tool has improved significantly in terms of other performance. OFFDTAN is about 1.13x faster than PANDA during the recording, and its offline analysis is faster than PANDA with a factor of 1.29. Overall, OFFDTAN operates about 1.23x faster than PANDA. Besides, OFFDTAN's CPU usage is about 24.4% smaller than PANDA, and the memory usage is 2.9% lower than PANDA. We attribute this to the employ of KVM acceleration on QEMU.

4.5. Experimental Summary. OFFDTAN can detect two kinds of vulnerabilities, i.e., stack buffer overflows and controlled jump, which fully validate that the two-vulnerability model combined with offline dynamic taint analysis can better achieve analysis effects.

In addition, whether with small program or large-scale program, such as FeiQ and Microsoft Office Word 2010, OFFDTAN can correctly analyze the address of program vulnerability, the cause of program crash, and the specific offset

of taint data within taint source file, thus ensuring that our approach is strictly correct and effective. Compared with PANDA, our method runs about 1.23x faster than PANDA. Moreover, OFFDTAN enhances the automation of vulnerability analysis as well.

5. Conclusion

The analysis efficiency of online dynamic taint analysis is a challenging problem, which usually occupies a considerable number of resources. In this paper, we generally eliminate this limitation by using the research approach of offline dynamic taint analysis, and on this point we propose OFFDTAN, an approach of offline taint analysis for binary program, which includes four stages: dynamic information acquisition, vulnerability modeling, offline analysis, and backtrace analysis. Our evaluation shows the effectiveness and correctness of this approach, as well as better performance.

However, OFFDTAN still has some deficiencies. First, program path coverage heavily depends on test cases; thus test cases will affect the accuracy of our method. Second, our experimental verification is still not sufficient, which needs further verification. Moreover, our approach can only detect stack buffer overflow vulnerabilities and controlled jump vulnerabilities. As future work, we are confident that the similar concepts can also be applied to model other types of vulnerabilities to adapt more scenes.

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 work was supported by the National Key R&D Program of China (no. 2016QY07X1404).

References

- [1] W. Shizhong, G. Tao, D. Guowei, Z. Puhan et al., *Software Vulnerability Analysis Technology*, The Science Publishing Company, Beijing, China, 2014.
- [2] Funnywei, "Buffer Overflow Vulnerability Mining Model [Z/OL]," 2003, <http://xcon.xfocus.net/XCon2003/archives/Xcon2003.funnywei.pdf>.
- [3] F. B. Qemu and F. B. Qemu, "A Fast and Portable Dynamic Translator," in *Proceedings of the Atec 05: Conference on Usenix Technical Conference*, 2005.
- [4] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," *ACM SIGPLAN Notices*, vol. 42, no. 6, pp. 89–100, 2007.
- [5] C. K. Luk, R. Cohn, R. Muth et al., "Pin: building customized program analysis tools with dynamic instrumentation," *Programming Language Design & Implementation*, vol. 9, no. 8, pp. 190–200, 2005.
- [6] V. Bala, E. Duesterwald, and S. Banerjia, "Dynamo: a transparent dynamic optimization system," *SIGPLAN Notices*, vol. 35, no. 5, pp. 1–12, 2000.
- [7] J. Newsome and D. Song, "Dynamic taint analysis for automatic dedection, analysis, and signature generation of exploits on commodity software," *Network and Distributed System Security Symposium (NDSS)*, 2005.
- [8] J. Clause, W. Li, and A. Orso, "Dytan: a generic dynamic taint analysis framework," in *Proceedings of the 2007 ACM International Symposium on Software Testing and Analysis (ISSTA '07) and PADTAD-V Workshop*, pp. 196–206, London, UK, July 2007.
- [9] F. Qin, C. Wang, Z. Li, H.-S. Kim, Y. Zhou, and Y. Wu, "LIFT: a low-overhead practical information flow tracking system for detecting security attacks," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '39)*, pp. 135–146, December 2006.
- [10] M. Kang G, S. Mccamant, P. Poosankam et al., "DTA++: dynamic taint analysis with targeted control-flow," in *Proceedings of the Network and Distributed System Security Symposium (NDSS '11)*, San Diego, Calif, USA, February 2011.
- [11] D. Song, D. Brumley, H. Yin et al., "A new approach to computer security via binary analysis," in *Proceedings of the 4th International Conference on Information Systems Security*, pp. 1–25, Springer-Verlag, 2010.
- [12] A. Henderson, A. Prakash, L. K. Yan et al., "Make it work, make it right, make it fast: building a platform-neutral whole-system dynamic binary analysis platform," in *Proceedings of the 23rd International Symposium on Software Testing and Analysis (ISSTA '14)*, pp. 248–258, July 2014.
- [13] L. Zhiquan, *Study of Fuzzing for Implementation of Stateful Network Protocol Based on Dynamic Taint Analysis*, National University of Defense Technology, 2010.
- [14] T. Wang, T. Wei, G. Gu, and W. Zou, "TaintScope: a checksum-aware directed fuzzing tool for automatic software vulnerability detection," in *Proceedings of the 31st IEEE Symposium on Security and Privacy (SP '10)*, pp. 497–512, IEEE Computer Society, May 2010.
- [15] Z. Jianwei, C. Libo, and F. Tian, "Type-based dynamic taint analysis technology," *Journal of Tsinghua University (Science and Technology)*, vol. 10, pp. 1320–1328, 2012.
- [16] K. Jee, V. P. Kemerlis, A. D. Keromytis, and G. Portokalidis, "ShadowReplica: efficient parallelization of dynamic data flow tracking," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13)*, pp. 235–246, November 2013.
- [17] D. Caselden, A. Bazhanyuk, M. Payer, S. McCamant, and D. Song, "HI-CFG: Construction by binary analysis and application to attack polymorphism," in *Computer Security—ESORICS*, pp. 164–181, Springer Berlin Heidelberg, 2013.
- [18] S. Manolis, P. Groth, and H. Bos, "Decoupling provenance capture and analysis from execution," in *Proceedings of the 7th USENIX Workshop on the Theory and Practice of Provenance (TaPP '15)*, 2015.
- [19] S. Dawei and Y. Tianwei, "A dynamic taint analysis method combined with coarse-grained and fine-grained," *Computer Engineering*, vol. 40, no. 3, pp. 12–17, 2014.
- [20] W. Fuwei, *Research on Taint Analysis-Oriented Binary Program Analysis and Vulnerability Mining*, Beijing University of Posts and Telecommunications, 2015.
- [21] J.-X. Ma, Z.-J. Li, T. Zhang, D. Shen, and Z.-K. Zhang, "Taint analysis method based on offline indices of instruction trace," *Journal of Software*, vol. 28, no. 9, pp. 2388–2401, 2017.

- [22] J. Ming, D. Wu, J. Wang, G. Xiao, and P. Liu, "StraightTaint: decoupled offline symbolic taint analysis," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE '16)*, pp. 308–319, September 2016.
- [23] B. Dolan-Gavitt, J. Hodosh, P. Hulin, T. Leek, and R. Whelan, "Repeatable reverse engineering with PANDA," *Computer Science*, 2014.

Research Article

Security Measurement for Unknown Threats Based on Attack Preferences

Lihua Yin,¹ Yanwei Sun ,^{2,3} Zhen Wang,⁴ Yunchuan Guo ,²
Fenghua Li,^{2,3} and Binxing Fang⁵

¹Cyberspace Institute of Advanced Technology (CIAT), Guangzhou University, Guangzhou, China

²State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

³School of Cyberspace Security, University of Chinese Academy of Sciences, Beijing, China

⁴School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China

⁵Institute of Electronic and Information Engineering of UESTC in Guangdong, Dongguan, China

Correspondence should be addressed to Yunchuan Guo; guoyunchuan@iie.ac.cn

Received 12 January 2018; Revised 14 March 2018; Accepted 10 April 2018; Published 20 May 2018

Academic Editor: Huaizhi Li

Copyright © 2018 Lihua Yin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Security measurement matters to every stakeholder in network security. It provides security practitioners the exact security awareness. However, most of the works are not applicable to the unknown threat. What is more, existing efforts on security metric mainly focus on the ease of certain attack from a theoretical point of view, ignoring the “likelihood of exploitation.” To help administrator have a better understanding, we analyze the behavior of attackers who exploit the zero-day vulnerabilities and predict their attack timing. Based on the prediction, we propose a method of security measurement. In detail, we compute the optimal attack timing from the perspective of attacker, using a long-term game to estimate the risk of being found and then choose the optimal timing based on the risk and profit. We design a learning strategy to model the information sharing mechanism among multiattackers and use spatial structure to model the long-term process. After calculating the Nash equilibrium for each subgame, we consider the likelihood of being attacked for each node as the security metric result. The experiment results show the efficiency of our approach.

1. Introduction

Security measurement matters to every stakeholder in network security and involves all the stages and aspects of the entire life cycle. There would be no effective security awareness and actions without accurate security measurement. The existing security measurements mainly focus on the relationship between exploits and system vulnerabilities, and their security measurements of unknown threats like zero-day loophole are very limited. In addition, zero-day attacks targeting governments and corporates are growing with time. An increasing number of hackers, motivated by their persistent love for technology or tempted by profits, are attempting to discover and propagate zero-day exploits. In 2016, 10822 vulnerabilities were found in China, and 2203 of them were zero-day vulnerabilities (<http://www.cert.org.cn/publish/main/upload/File/2016CNVDannual.pdf>), which

may cause serious consequences. According to *The Hacker News*, hackers exploited the zero-day vulnerability to attack Bangladesh's central bank in 2016 and stole over \$80 million from the Federal Reserve Bank (<http://thehackernews.com/2016/03/bank-hacking-malware.html>). In such case, it poses a great challenge as to carry out effective measurements of threats posed by zero-day vulnerabilities to help system administrators better understand and guard against them.

Current security measurements are mostly around known vulnerabilities. They get the result of the measurement after analyzing the attacks and coming up with corresponding rules. Such measurement can be a distortion from real-life situation. For example, some vulnerabilities proven highly threatening according to CVSS' measurement are not actually exploited too much by the attackers. The work proposed by Wang et al. [1] has similar problems; this is one of the few measurement works targeting hidden vulnerability. The

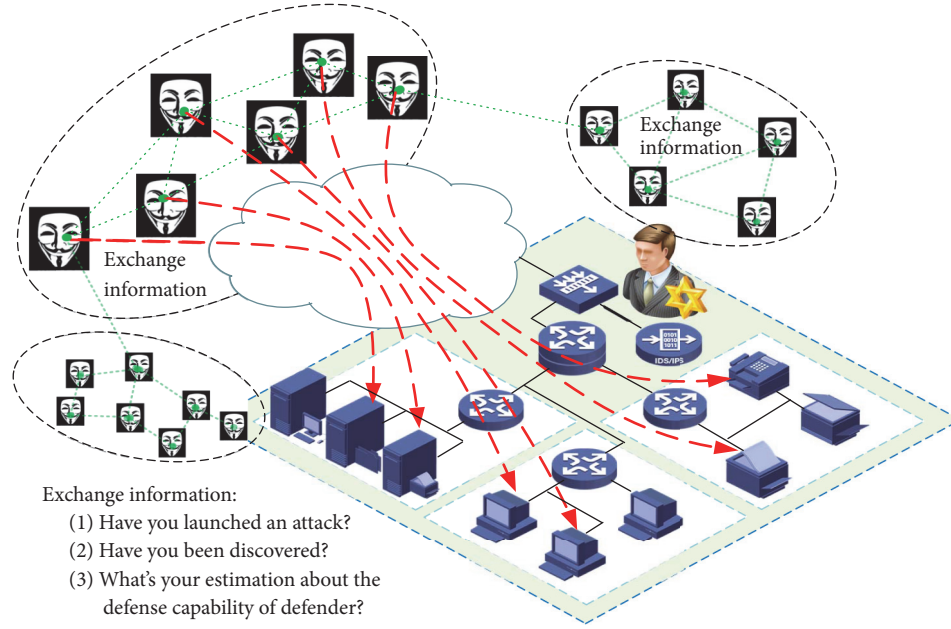


FIGURE 1: Multiattackers versus a certain company.

paper uses attack graph to analyze and decide the minimal number of vulnerabilities needed to achieve a set goal, assuming that there is a hidden vulnerability on any of the nodes. The number will be used as a reference for security measurements. This study measures only from the dimension of complexity, not taking the differences in the attackers' preference, and the result is only a theoretical one, so there is not too much reference value for the result. For example, if there is a network with weak protection and only one hole is needed to break into the network, then the measurement result should be highly risky according to the method above. However, the attackers will not attack a network that they deem of no value; Honeynet is a case in point. Therefore, when measuring unknown threats, we need to include the attacking decision of the attacker as a significant metric and take the degree of complexity, possible risks, benefits, and so on into consideration. This way, we can analyze the decision the attacker is most likely to take and get a more realistic measurement result. That is to say, if, during a certain period, the attackers are eager to attack a node, then we will think this node is facing serious threats at this time; otherwise, the result should not be labeled as highly threatening.

The security measurement of unknown threats with the attack behavior preferences in mind faces the following three major challenges:

- (i) Modeling zero-day attacks from the time dimension: current studies express unknown vulnerabilities using attack graph, which are mainly from the spatial dimension. But zero-day attack is a long-term process. The zero-day exploit (also called cyber resources) will always be available unless the vulnerability is fixed by the defender. As the process goes on, the attacker needs to make a tradeoff between risk and profit for each time point: if the vulnerability is

exploited, the attacker may get some profit but the risk of being noticed may also increase. On the contrary, if the zero-day vulnerability is exploited too late or not exploited, there is also a chance for the defender to fix it, leaving no chances for the attacker. So the first challenge is how to model this process properly from the dimension of time.

- (ii) Identify and calculate the factors that change over time: in the course of the zero-day attack and defense, the factors that influence the attacker's decision-making include the potential profits, the attacker's knowledge of the defending party, and the risks related to the attack. A comprehensive and reasonable expression and calculation of these factors are the basis for predicting attack decision.
- (iii) Predict the attack decision and measure the unknown threat: after setting the relevant parameters, how to use these parameters to accurately predict the attacker's possible decision at each time point determines the accuracy and reliability of the entire security measurement.

To overcome these challenges, this paper, which is based on our previous work [2], uses long-term game theory to predict the behavior of the attacker (the attack-defense scenario is shown in Figure 1) and then propose a new security metric based on the prediction. The main contributions are as follows:

- (i) We present a multiple game model: we consider zero-day attacks as a long-term process. The attacker's decision (attack or wait) at any point in time will have a corresponding effect on the later game process. Therefore, we discuss the continuous game process in a certain period of time.

- (ii) We discuss the factors that affect the attack and defense: we discuss the factors that affect the attack and defense, particularly the change of the attacker's understanding of the defensive ability of the defender. At the beginning of the game, attackers know little about the defender. But with the game process, their understandings become more and more accurate due to their observation and information sharing. We design a learning mechanism to simulate the correction process.
- (iii) We propose a new security metric based on attack prediction: from the attacker's point of view, we calculate the Nash equilibria for each subgame. Using the result as an important reference, we propose a new security metric for unknown threat.

The rest of the paper is organized as follows. Section 2 surveys the related work. Section 3 introduces the preliminaries. Section 4 introduces long-term game formulation. Section 5 discusses the details about the security measurement method. Section 6 reports experimental results, and Section 7 gives the conclusions.

2. Related Work

There have been plenty of research studies on network security metrics. They focus on different aspects, such as metrics of system vulnerabilities, metrics of defense power, metrics of situations, and metrics of attack or threat severity [3]. In detail, when assessing the risk of malware threat, Hardy et al. [4] proposed the targeted threat index combining the social engineering and technical sophistication. Thakore [5] provided a set of metrics such as coverage, redundancy, confidence, and cost to quantitatively evaluate monitor deployments. Kührer et al. [6] focused on the effectiveness of malware blacklists and showed that the current blacklist is insufficient to protect against the variety of malware threats. There are other studies focused on evaluating the strength of IDS or other security products [7, 8], strength of user password [9, 10], and so on. However, the effects of all the metrics mentioned above are not ideal when faced with unknown threats.

In order to make a better understanding of zero-day attack, some researches focus on analyzing the attack itself such as detecting and identifying the attack. To identify the unknown files, Avasarala et al. [11] introduced the class-matching approach. Mishra and Gupta [12] proposed a hybrid solution which uses the concept of CSS matching and URI matching to defend against zero-day phishing attacks. Wang et al. proposed some representative works on measuring the zero-day attack [1, 13, 14]. To evaluate the robustness of networks, [13, 14] modeled network diversity as a security metric and then proposed two complementary diversity metrics. The paper [1] conducted the evaluation process based on how many zero-day vulnerabilities are required to compromise a network asset. However, all these works are conducted based on attack graph and do not consider the attacker behavior.

TABLE 1: Payoff in the two-player one-shot game.

	Protect	Not protect
Attack	$-C_a, -C_d$	$G - C_a, -G$
Not attack	$0, -C_d$	$0, 0$

Analyzing the attacker behavior is of great importance when measuring the network security. Ekelhart et al. [15] developed a simulation-driven approach which took attack strategies and attacker behavior into consideration. Al-Jarrah and Arafat [16] used the time delay neural network which embedded the temporal behavior of the attacks to maximize the recognition rate of network. Mitchell and Chen [17] proposed specification-based IDS which can adapt to different attacker types such as reckless, random, and opportunistic attackers. In this way, it could get a higher detection accuracy. Allodi and Massacci first pointed out that not all the vulnerabilities were equally exploited by the attacker [18] and then focused on the choice of attackers [19]. By validating the actual “traces” attacks left on real systems, they claimed that the real attacker would behave less powerful than we thought and would not exploit every vulnerability. The attackers would strategically choose the busy periods and some certain vulnerabilities, while the efforts of security professionals were diffused across many vulnerabilities [20, 21]. Based on this observation, Dumitraş [22] proposed a novel metrics that enabled a more accurate assessment of the risk of cyberattacks. Bozorgi et al. [23] used machine learning method with high dimensional feature vectors input to predict the vulnerability which was most likely to be exploited by the attacker. All these analyses, however, are from the perspective of defender, ignoring the information sharing mechanism among attackers where the mechanism is the most important part during the attack and can guide attackers to change their strategies dynamically.

3. Preliminaries

Before introducing the long-term game, we begin with the simple attack-defense game. The players of the game are attacker and defender. Liang and Xiao [24] divide the game applications into two subclasses: general analysis and the specialized analysis. In general analysis, the networks are often not specific but abstract, and the strategy set of attacker $SA = \{\text{attack}, \text{not attack}\}$; meanwhile, the set of defender $SD = \{\text{protect}, \text{not protect}\}$. Let C_a represent the attack cost, C_d represent the defense cost, and G represent the profit of a successful attack. In this section, G is a fixed value. Their payoffs are as shown in Table 1. In [25], a *Remainder cost* was defined to indicate the damage that the attack brought to the system after the defender implemented the defense strategy, and *Remainder cost* = $G \times \epsilon$ where $\epsilon \in [0, 1]$. For simplicity, we assume that $\epsilon = 0$ in this section; that is, if the defender implemented the defense strategy, there is no damage to the system and no reward to the attacker.

According to Table 1, we can see that when the attack cost is fixed, if the attacker and the defender are completely rational, both of them will make their decisions by calculating

the Nash equilibrium, and the Nash equilibrium is related to the parameter discussed below:

- (1) If $G \leq C_a$, the attacker will not attack and the defender will not protect. In this case, we can say the network is pretty safe.
- (2) If $C_d \geq G > C_a$, the attacker will attack, but the defender still not protect. In this case, we can say the network is of great danger.
- (3) If $G > C_a$ and $G > C_d$, no pure Nash equilibrium exists, but there is a mixed Nash equilibrium; that is the attacker will choose to launch an attack with the probability of $P_A = C_d/G$, and the protecting probability $P_D = (G - C_a)/G$. In this case, we can say the network is a little bit safer than case two, but more dangerous than case one.

It can be seen from the above model that if the defender decides to protect the target, the attacker cannot finish his attack successfully, consequently getting no reward. As time goes on, this one-shot game is repeated time after time, and there is no necessary correlation between each of them. This is a simple case for security metrics, but it does not apply to the attacker who holds the zero-day exploits of certain target, mainly for the following two reasons. First, the payoff is much different. According to the stealth of zero-day exploits [26], most of the software and the security products cannot detect the existence of it and thus difficult to resist the zero-day attacks effectively. So, the attacker can get the corresponding profit only if he/she launches an attack. Second, the strategy is much different. Compared with the one-shot game, attacker with zero-day exploits is more concerned about the persistence of the resource, he/she has to make sure if this resource will still be useful after this attack and then makes a tradeoff between risk and profit. Therefore, the key points to the entire game process are the attacker's decision and the defense capability. In next section, we will discuss the detail of the long-term game.

4. Long-Term Game Formulation

We analyze the attack-defense scenario between multiple attackers and a single target. The target could be a company or an organization, and it contains a lot of nodes. And we assume that each node has at least one cyber resource (where the node has no resource is out of our discussion). All of these vulnerable nodes are protected by the same administrator, so we assume that all the nodes share the same defense capability. We define an attack-defense game as a combination of one resource and one node. The attacker who owns more than one resource means that he will be involved in more than one attack-defense game. So the relationship is a many-to-many mapping. As mentioned above, our discussion takes place over a certain period of time since the zero-day attack is a long-term process. We assume that it takes one-time tick to complete an attack, and if the vulnerability was not discovered by the administrator this time, it will still be useful to the attacker next time. For each time tick, we compute the probability of being attacked

for every node and take these results as the security metric. In this section, we first introduce the game formulation and some key parameters and then we focus on the attackers learning strategy.

4.1. Attack-Defense Game

Cyber Resource. We call a zero-day exploit or a set of zero-day exploits a cyber resource to the attacker [27]. A cyber resource could help attacker to finish a certain attack. If one or more exploits are fixed or expired, which could cause the failure of the attack, then we say the resource is expired.

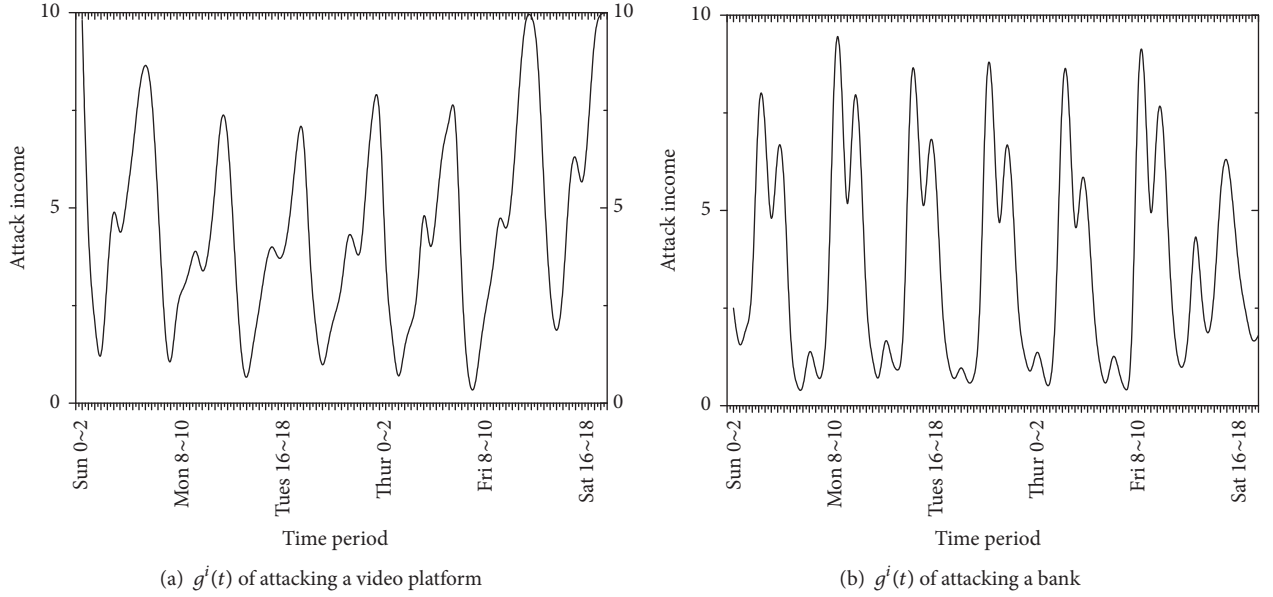
The Number of Long-Term Game. In this attack-defense scenario, there are total m attackers, n cyber resource, and q nodes. During the game, q is a constant, but m and n will change from time to time. Compared with the number of attackers, we are more concerned about the number of resources, because n resource means n attack-defense games. We use $n(t)$ to represent the total number at time tick t . There are three main aspects that will influence the number. First, at the beginning of time tick t , $n_{\text{new}}(t)$ is used to denote the newly joined resource. Second, at the end of time point t , $n_{\text{exp}}(t)$ is used to denote the number of expired resources. Third, according to the defense capability, some of the resources are randomly eliminated at the end of time point t , and the number is notated as $n_{\text{dis}}(t)$. Then the total number of attack-defense games at time point $t + 1$ is

$$n(t + 1) = n(t) + n_{\text{new}}(t) - n_{\text{dis}}(t) - n_{\text{exp}}(t). \quad (1)$$

The Duration of Each Long-Term Game. Different from the one-shot game, zero-day attack-defense game is a long-term game. It consists of multiple subgames over a period of time. The game will go on as long as the resource is still useful. There are mainly three possibilities to terminate the game:

- (1) The resource is expired. We use notation L to represent the lifecycle of certain resource. Different vulnerabilities have different lifecycles. For example, compared with buffer overflow and executable code, other vulnerabilities such as PHP vulnerability or SQL injection often have longer lifecycles [28]. If one of these vulnerabilities expired, we say the resource is expired.
- (2) The resource was found and patched due to the attack action. We call this the passive-defense capability of the administrator, denoted as P_a , indicating the probability of discovering the vulnerability *after* being attacked.
- (3) The resource was found and patched when nothing happens. We call this the initiative-defense capability of the administrator, denoted as P_b , indicating the probability of discovering the vulnerability *before* being attacked.

P_a and P_b are determined by the capability of the administrator itself, and both are fixed values which are known to the administrator and unknown to the attackers during the

FIGURE 2: Sample of $g^i(t)$.

game process. Each attacker has its own assessment about these two values, denoted as $P_a^i(t)$ and $P_b^i(t)$. After each subgame, the attacker will revise the assessment by observing the state of other neighbors and exchanging information with each other. This is consistent with the values of most hackers who believe that all the information should be free. They think everyone has the rights to access information, so, many hacker groups form a unique “black ecosystem” to share information more effectively. According to the director of Baidu security laboratory (Baidu is the predominant search engine in China), the information sharing mechanism among such black ecology is much better than those among white ecosystem. The details about the update rules will be discussed at the next subsection.

The Strategy Set for Each Subgame. The same as the one-shot game mentioned above, the strategy set of attacker SA = {attack, not attack}; meanwhile, the set of defender SD = {defend, not defend}.

The Payoff for Each Subgame. For any resource i , at any time t , the payoff of the subgame consists of three parts: the attack cost denoted as C_a^i , the one-time attack income $g^i(t)$, and the long-term profit expectancy from time t to the end of the long-term game, denoted as $E^i(t)$. $E^i(t)$ is dependent on four factors including $g^i(t)$, $P_a^i(t)$, $P_b^i(t)$, and the attacker’s action last subgame. When making a decision, what the attacker really cares about is not only the one-time attack income for the attack, but also the long-term profit expectancy during the whole lifecycle. The specific parameter settings are discussed in the next subsection. For the administrator, the loss also includes two parts, the defense cost denoted as C_d and the loss caused by the attack. In order to reduce the complexity of this model, it is assumed that the attacking loss equals the negative of the attacker’s attack revenue.

Game Rules. For each subgame, given the payoffs, both attacker and administrator determine their decisions by calculating the Nash equilibrium. At the end of each subgame, if the resource is still useful, attacker will revise the assessment by observing the state of other neighbors and exchanging information with each other in order to recalculate the payoffs for next subgame.

4.2. Some Key Parameters. Notations lists the key parameters used in our long-term game model. In this paper, we assume that C_a^i , C_d , P_a , P_b , and L are fixed value, and we mainly discuss the following three parameters.

Gain Function of Time $g^i(t)$. For certain resource i , gain function of time $g^i(t)$ represents the one-time attack income that the attacker can get when he attacks the node at the certain time t . This $g^i(t)$ could be a fixed value or a function of time, and that depends on the target type. For example, if the attacker wants to attack some e-commerce platforms, he can get better rewards in some specific days such as Black Friday. For the sake of discussion, Figure 2 shows the instantaneous yield curves of attacks against a live video platform and a bank over different time periods. It can be seen that, for a bank attack, the attack revenue on weekdays is higher than that on weekends, and the attacks on working hours are higher than the off-hours. On the contrary, the attack revenue is relatively higher on weekends and late nights when the attack target is changed to the video platform. In general, we assume that $g^i(t)$ is affected by the target’s visiting traffic, business process, and customer’s work schedule. The specific values are beyond the scope of this article. This article assumes that both attackers and administrator have a good understanding of the target of the attack, so the value of $g^i(t)$ is known.

Profit Expectancy $E^i(t)$. For the certain resource i , we use $E^i(t)$ to represent the profit expectancy from time t to the end of the lifecycle, so $E^i(t) = \sum_{\sigma=t}^L P_A(\sigma)g'(\sigma)$, where $g'(\sigma) = \max\{g(\sigma) - C_a, 0\}$ and $\sigma \in [t, L]$. So, we need to compute all the $P_A(\sigma)$ for $\sigma = t, t+1, \dots, L$. If notation Q is used to indicate that the resource is still available at time point σ and notation R is used to indicate that the attacker will choose to attack, then $P_A(\sigma) = P(Q) * P(R)$. So we have

$$P_A(\sigma) = [P_A(\sigma-1) * (1 - P_a) + (1 - P_A(\sigma-1)) * (1 - P_b)] * P_A(t-1). \quad (2)$$

$P_A(\sigma-1) * (1 - P_a)$ means that the attacker has attacked last time but has not been discovered, and $(1 - P_A(\sigma-1)) * (1 - P_b)$ means that the attacker has not attacked last time and has not been discovered. To simplify the computation, it is assumed that the defender will always be protected when calculating $P_A(\sigma)$. And $P_A(t-1)$ used in (4) is an approximate value. Because the exact value of $P_A(t)$ which is determined by calculating the Nash equilibrium of the subgame at time point σ could not be known, the probability $P_A(t-1)$ is used instead. According to the above recursive formula, we can get

$$P_A(\sigma) = \left[P_A(t-1) + \frac{(1 - P_b) * P_A(t-1)}{(P_b - P_a) * P_A(t-1) - 1} \right] \cdot [(P_b - P_a) * P_A(t-1)]^{t-1} - \frac{(1 - P_b) * P_A(t-1)}{(P_b - P_a) * P_A(t-1) - 1}. \quad (3)$$

The Estimation of Administrators Defense Capability $P_a^i(t)$ and $P_b^i(t)$. During the game, the attacker does not know the real P_a and P_b , but each attacker has its own assessment about these two values, denoted as $P_a^i(t)$ and $P_b^i(t)$. After each subgame, the attacker will update the assessment by observing the state of other neighbors and exchanging information with each other. The update rule includes the following main steps.

- (i) Initializing the assessment randomly at the beginning of the game: the initial assessments are random because the attacker knows little about the defender.
- (ii) Calculating the observed result of P_a and P_b : at the end of the time point t , the attacker observes his neighbors and counts the numbers of them (1) who had attacked this time and been discovered and (2) who had not attacked this time and been discovered and then calculates the observed result of P_a and P_b .
- (iii) Combining the observed result, neighbors' assessment with his previous assessment to be his new assessment: when combining these three results, the reference value difference should be considered. For the neighbor who has survived longer, its assessment has a higher reference value. What is more, at the beginning of the game, the observed result plays an important role. However, as the game goes on, this importance diminished. Because the observed

samples, that is, attacker's neighbors, are limited, the observed result has a strong randomness.

Some new parameters and notations are introduced as follows. Let P_{aD} denote the probability of the attacker being eliminated after attack, so $P_{aD} = P_a \times P_D$. Similarly, P_{bD} is used to denote the probability of being eliminated without attack, so $P_{bD} = P_b \times P_D$. For any attacker $i \in AT$ at the end of time point $t \in [0, L_i]$, $P_a^i(t)$ and $P_b^i(t)$ represent the attacker's assessment of P_a and P_b , $P_a^i(0)$, and $P_b^i(0)$ are the initial estimates. We use $s = \{s_1^t, s_2^t, \dots, s_{k_t}^t\}$ to denote the neighbors' strategy and $f = \{f_1^t, f_2^t, \dots, f_{k_t}^t\}$ to denote whether these neighbors are found by the defender or not, where k_t is the total number of neighbors of i , $s_j^t \in Sa$, $f_j^t \in 0, 1$. AD is used to denote the neighbors who had attacked this time and had been found, so $AD = \{j \mid s_j^t = 1 \wedge f_j^t = 1, j \in [0, k_t]\}$. ND is used to denote the neighbors who were not attacked and had been found, so $ND = \{j \mid s_j^t = 0 \wedge f_j^t = 1, j \in [0, k_t]\}$. $P_D^t(j)$ is used to denote the P_D calculated by neighbor j . Let P_{obaD}^t denote the observed value of P_{aD} , so $P_{obaD}^t(i) = |AD| / \sum s_j^t$; let P_{obbD}^t denote the observed value of P_{bD} so $P_{obbD}^t(i) = |ND| / (k - \sum s_j^t)$. After introducing above notations, this paper presents three plans for revising the parameters P_a and P_b ; we will make a brief introduction.

Plan 1: Average Summation. Record all the P_a^t and P_b^t for each neighbor and resource i itself, and then take the average with $P_{obaD}^t(i)$ to calculating P_{aD}^t , donated as

$$\overline{P_{aD}^t} = \frac{\sum_{j=1}^{k_t} P_a^t(j) P_D^t(j) + P_a^t(i) P_D^t(i) + P_{obaD}^t(i)}{k_t + 2} \quad (4)$$

and then calculate

$$\overline{P_D^{t+1}} = \frac{\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i)}{k_t + 1} \quad (5)$$

so

$$P_a^{t+1} = \frac{\overline{P_{aD}^{t+1}}}{\overline{P_D^{t+1}}} = \frac{(\sum_{j=1}^{k_t} P_a^t(j) P_D^t(j) + P_a^t(i) P_D^t(i) + |AD| / s_j^t) (k_t + 1)}{(k_t + 2) (\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, \quad (6)$$

$$P_b^{t+1} = \frac{\overline{P_{bD}^{t+1}}}{\overline{P_D^{t+1}}} = \frac{(\sum_{j=1}^{k_t} P_b^t(j) P_D^t(j) + P_b^t(i) P_D^t(i) + |ND| / (k - s_j^t)) (k_t + 1)}{(k_t + 2) (\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}.$$

Plan 2: Staged Average Summation. In the first half of the resource life cycle, revise P_a and P_b according to Plan 1, and remove $P_{obaD}^t(i)$ and $P_{obbD}^t(i)$ when calculating P_{aD}^{t+1} and P_{bD}^{t+1} in the second half. That is because when this long-term game

goes to a certain stage, the estimated value $P_a^{t+1}(i)$ and $P_b^{t+1}(i)$ are already close to the actual value, but there is a large

uncertainty in the observations, so we removed the $P_{oabD}^t(i)$ and $P_{obbD}^t(i)$ in the second half.

$$P_a^{t+1}(i) = \begin{cases} \frac{(\sum_{j=1}^{k_t} P_a^t(j) P_D^t(j) + P_a^t(i) P_D^t(i) + |AD|/s_j^t)(k_t + 1)}{(k_t + 2)(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 \leq \frac{L}{2}, \\ \frac{\sum_{j=1}^{k_t} P_a^t(j) P_D^t(j) + P_a^t(i) P_D^t(i)}{(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 > \frac{L}{2}, \end{cases}$$

$$P_b^{t+1}(i) = \begin{cases} \frac{(\sum_{j=1}^{k_t} P_b^t(j) P_D^t(j) + P_b^t(i) P_D^t(i) + |ND|/(k - s_j^t))(k_t + 1)}{(k_t + 2)(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 \leq \frac{L}{2}, \\ \frac{\sum_{j=1}^{k_t} P_b^t(j) P_D^t(j) + P_b^t(i) P_D^t(i)}{(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 > \frac{L}{2}. \end{cases} \quad (7)$$

Plan 3: Staged and Weighted Average Summation. Based on plan two, we introduce the concept of neighbor weight; that is, the longer the neighbor exists, the greater reference value it can provide. It should be noted that the notation t in $P_a^t(i)$ and

$P_b^t(i)$ is the existing time of resource i but not the existing time of certain neighbor. So we use $T = \{t_1, t_2, \dots, t_k\}$ to represent the existing time of k neighbors. Then the formula of plan three is as follows:

$$P_a^{t+1}(i) = \begin{cases} \frac{(\sum_{j=1}^{k_t} P_a^t(j) P_D^t(j) t_j + P_a^t(i) P_D^t(i) t + |AD|/s_j^t)(k_t + 1)}{(\sum_{j=1}^{k_t} t_j + t + 1)(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 \leq \frac{L}{2}, \\ \frac{(\sum_{j=1}^{k_t} P_a^t(j) P_D^t(j) t_j + P_a^t(i) P_D^t(i) t)(k_t + 1)}{(\sum_{j=1}^{k_t} t_j + t)(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 > \frac{L}{2}, \end{cases}$$

$$P_b^{t+1}(i) = \begin{cases} \frac{(\sum_{j=1}^{k_t} P_b^t(j) P_D^t(j) t_j + P_b^t(i) P_D^t(i) t + |AD|/s_j^t)(k_t + 1)}{(\sum_{j=1}^{k_t} t_j + t + 1)(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 \leq \frac{L}{2}, \\ \frac{(\sum_{j=1}^{k_t} P_b^t(j) P_D^t(j) t_j + P_b^t(i) P_D^t(i) t)(k_t + 1)}{(\sum_{j=1}^{k_t} t_j + t)(\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i))}, & t + 1 > \frac{L}{2}. \end{cases} \quad (8)$$

5. Measure the Network Security

5.1. Calculate the Nash Equilibrium for Each Subgame. After calculating the above parameters, we can fill the payoffs matrix for each subgame (see Table 2) where

$$\begin{aligned} G_A^D &= g'(t) + (1 - P_A)E(t + 1), \\ G_{NA}^D &= (1 - P_b)E(t + 1), \\ G_A^{ND} &= g'(t) + E(t + 1), \\ G_{NA}^{ND} &= E(t + 1). \end{aligned} \quad (9)$$

Here is a brief discussion of the game's Nash equilibrium, also the optimal timing selection guidelines:

- (1) At some time t , when $g'(t) \leq 0$ and $C_d < P_b E(t + 1)$, attacker will not attack but the defender will protect. When $g'(t) \leq 0$ and $C_d > P_b E(t + 1)$, attacker will not attack and the defender will not protect.

Because $g'(t) \leq 0$ and $P_a > P_b$, so $G_A^{ND} < G_{NA}^{ND}$; the best choice for attackers is not attacking whatever the defending choice is. But in terms of the defenders, if $C_d < P_b E(t + 1)$, that means there is some probability of discovering the vulnerability by expending a little defending cost, so the defender will defend. But if the cost is high, that is, $C_d > P_b E(t + 1)$, the defender will not defend.

- (2) At some time t , when $g'(t) > (P_a - P_b)E(t + 1)$ and $C_d < P_a E(t + 1)$, attacker will attack and the defender will protect. When $g'(t) > (P_a - P_b)E(t + 1)$ and $C_d \geq P_a E(t + 1)$, attacker will attack and the defender will not protect.

Because $g'(t) > (P_a - P_b)E(t + 1)$, that means the profit of this attack is higher than the loss caused by the discovery of the attack, so the best choice for attackers is attacking whatever the defending choice is. But in terms of the defender, under this circumstance, if

TABLE 2: Payoff in the multiplayer evolutionary game.

	Protect	Not protect
Attack	$G_A^D, -G_A^D - C_d$	$G_A^{ND}, -G_A^{ND}$
Not attack	$G_{NA}^D, -G_{NA}^D - C_d$	$G_{NA}^{ND}, -G_{NA}^{ND}$

$C_d < P_a E(t+1)$, the defender will defend, but if $C_d \geq P_a E(t+1)$, that means the defending cost is higher than the loss caused by being attacked, so defender will not protect.

- (3) At some time t , when $0 < g'(t) \leq (P_a - P_b)E(t+1)$ and $C_d \geq P_a E(t+1)$, the defender will not protect and the attacker will attack.

Because $C_d < P_a E(t+1)$ and $P_a > P_b$, so $-G_A^D - C_d < -G_A^{ND}$ and $-G_{NA}^D - C_d < -G_{NA}^{ND}$; due to the high cost of protection, the defender will not defend whatever the attacking choice is. Under this circumstance, the attacker will choose attack.

- (4) At some time t , when $0 < g'(t) \leq (P_a - P_b)E(t+1)$ and $C_d < P_b E(t+1)$, the defender will protect and the attacker will not attack.

Because $C_d < P_b E(t+1)$ and $P_a > P_b$, so $-G_A^D - C_d > -G_A^{ND}$ and $-G_{NA}^D - C_d > -G_{NA}^{ND}$; due to the low cost of protection, the defender will defend whatever the attacking choice is. Under this circumstance, the attacker will not choose attack.

- (5) At some time t , when $0 < g'(t) \leq (P_a - P_b)E(t+1)$ and $P_b E(t+1) < C_d < P_a E(t+1)$, there is no pure Nash equilibrium, but only mixed Nash equilibrium; that is, the attacker will attack with the probability of $(C_d - P_b E(t+1)) / ((P_a - P_b)E(t+1))$, and the defender will defend with the probability of $g'(t) / ((P_a - P_b)E(t+1))$.

We use X to denote the probability of attack for the attacker and Y to denote the probability of defending. So the expected utility function of the attacker is

$$\begin{aligned}
 U_A &= X [Y G_A^D + (1 - Y) G_A^{ND}] \\
 &\quad + (1 - X) [Y G_{NA}^D + (1 - Y) G_{NA}^{ND}], \\
 U_D &= Y [X (-G_A^D - C_d) + (1 - X) (-G_{NA}^D - C_d)] \\
 &\quad + (1 - Y) [X (-G_A^{ND}) + (1 - X) (-G_{NA}^{ND})].
 \end{aligned} \tag{10}$$

Differentiate the above-mentioned function:

$$\begin{aligned}
 \frac{\partial U_A}{\partial X} &= [Y G_A^D + (1 - Y) G_A^{ND}] \\
 &\quad - [Y G_{NA}^D + (1 - Y) G_{NA}^{ND}].
 \end{aligned} \tag{11}$$

Let $\partial U_A / \partial X = 0$; we get

$$Y = \frac{G_{NA}^{ND} - G_A^{ND}}{G_A^D + G_{NA}^{ND} - G_A^{ND} - G_{NA}^D} = \frac{g'(t)}{(P_a - P_b)E(t+1)}. \tag{12}$$

Similarly,

$$\begin{aligned}
 \frac{\partial U_D}{\partial Y} &= [X (-G_A^D - C_d) + (1 - X) (-G_{NA}^D - C_d)] \\
 &\quad - [X (-G_A^{ND}) + (1 - X) (-G_{NA}^{ND})].
 \end{aligned} \tag{13}$$

Let $\partial U_D / \partial Y = 0$; we get

$$\begin{aligned}
 X &= \frac{G_{NA}^D - G_{NA}^{ND} + C_d}{-G_A^D - G_{NA}^{ND} + G_A^{ND} + G_{NA}^D} \\
 &= \frac{C_d - P_b E(t+1)}{(P_a - P_b)E(t+1)}.
 \end{aligned} \tag{14}$$

5.2. Measure the Network Security Using the Nash Equilibrium. After calculating the Nash equilibrium for each subgame, we use this result as a reference for the safety measurement of unknown threat.

At any time t , for the j th node in the target, let non-negative vector $p_j(t) = [p_{j,1}(t), \dots, p_{j,K_j}(t)]^T$ denote the probability distribution of K_j unknown vulnerabilities, where $p_{j,m}(t) \in [0, 1]$, $m = 1, \dots, K_j$, and $p_{j,m}(t)$ represents the probability of m th resource being used by the attacker. So we can compute the probability that the node j is being attacked at time t :

$$P_j(t) = 1 - \prod_{n=1}^{K_j} (1 - p_{j,n}(t)). \tag{15}$$

Assume $P(t) = [P_1(t), P_2(t), \dots, P_q(t)]^T$ which models the attack probability distribution of all nodes in the target; $W(t) = [w_1(t), w_2(t), \dots, w_q(t)]$ represent the weight of each node in the target. We use $S(t)$ to denote the final result of the security measurement:

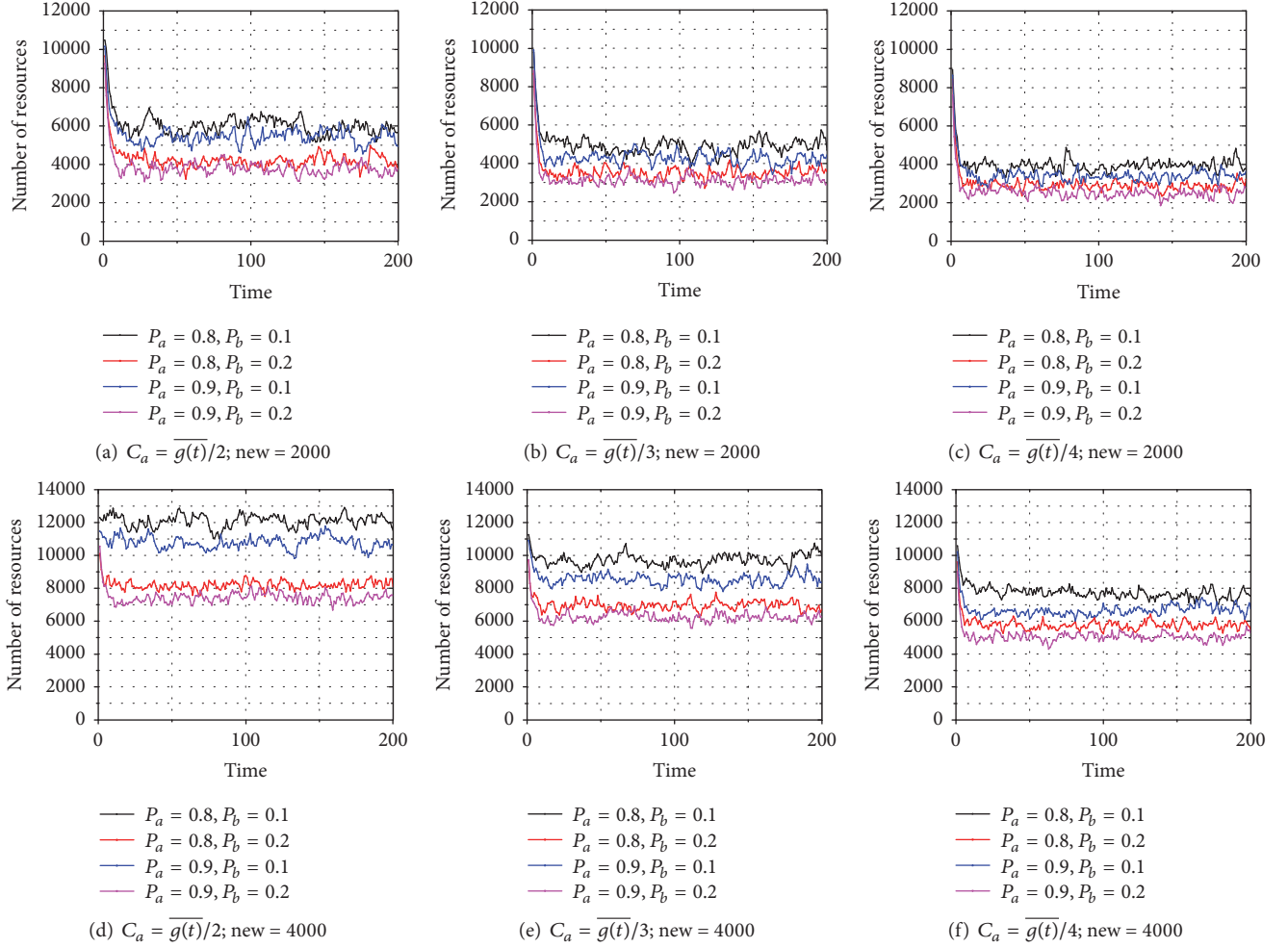
$$S(t) = P_j(t) \cdot W(t) = \sum_{n=1}^q P_n(t) \cdot w_n(t). \tag{16}$$

From the above equation we can see that the higher the value of $S(t)$, the greater the threat level. In next section, we will show how to calculate this score in detail.

6. Experiment

We evaluate the effectiveness of the proposed evolutionary games by synthetic data set. All experiments are conducted on a Windows 7 system with Intel Core i7-6700 3.4 GHz CPUs and 8 G memory.

Exp1: Numbers of Subgames. We first conduct our experiment and observe the total number of subgames. We provide ten types of gain functions with vary monotonicity and codomain. We set the lifecycle of each resource as 20 (for each subgame, it equals 1), and the total number of subgames for the beginning is 12000 (different attackers may keep the same resource), distributed at a 200×100 matrix. The numbers of new joins can be regarded as a statistic process obeying Gaussian distribution. We pick different P_a and P_b

FIGURE 3: Numbers of resources with varying parameters C_a and new_{dis} .

to see whether the number could get stable with different number of new joins. Figure 3 shows the total number of subgames where the average of new joins equals 2000 and 4000; meanwhile, we set different attack cost.

We can see that the number of subgames can get stable in all these six experiments. We also can see there are so many factors that affect the number of subgames during the game. For example, different attack cost values lead to different experimental results. The average number in Figure 3(a) is higher than that in Figures 3(b) and 3(c). This is because when the cost of the attack is too high, the attacker's probability of attack will drop drastically, so the probability of being discovered is relatively lower. We can also find that when P_b equals 0.1 (notated as the black line and blue line), the number is greater than those where P_b equals 0.2 (the red line and pink line). This is because the greater P_b is, the greater the number of discovered vulnerabilities is. However, compared with P_b , the difference of P_a does not have a major impact on the change of total number. That is reasonable because during the whole game process, the probability of attack is much less than the probability of waiting, so the impact of P_a is much less.

Exp2: Measure the Network Security Using the Attack Probabilities. After calculating the attack probability for each vulnerability at each time, we use these results as important references to measure the network security. This experiment aims to show how to measure the network security using the attack probabilities. Due to the space limitation, it is impossible to list all the nodes and the attack probabilities of the game, so we use a lite version to explain our method. We assume that the target has seven nodes and each node has at least one vulnerability. Table 3 lists the probabilities of these nodes at the certain time period. The attack probability for each vulnerability is computed through the game Nash equilibrium, and the attack probability for each node, recorded as "tot" in the table, is computed through (15). From the table we can see that, for some vulnerability, the attack probability decreased to zero and never increase, such as the vulnerability 2 in node 1, which means this vulnerability had been fixed at certain time. And for some other vulnerability, the attack probability stayed at zero for a period and then increased such as vulnerability 2 in node 2, which means the vulnerability is newly discovered at time t_6 . We take node 1 as an example; there are total 2 vulnerabilities, and at

TABLE 3: The attack probability for 7 nodes.

(a)

Time	Node1			Node2			Node3			Node4		
	vul1	vul2	tot	vul1	vul2	vul3	tot	vul1	tot	vul1	vul2	tot
t_1	0.25	0.06	0.3	0.16	0	0.13	0.27	0.34	0.34	0.22	0.13	0.43
t_2	0.15	1	1	0.75	0	0.16	0.79	0	0	0.35	0.41	0.9
t_3	0.34	0	0.34	0.18	0	0.09	0.25	0	0	0.49	0.27	0.67
t_4	0.17	0	0.17	0.26	0	0.05	0.3	0.15	0.15	0.59	0.34	0.79
t_5	0.22	0	0.22	0.57	0	0.16	0.64	0.05	0.05	0.61	0.02	0.86
t_6	0.3	0	0.3	0.63	0.15	0	0.69	0	0	0	0.17	0.47
t_7	0.51	0	0.51	0.67	0.26	0.03	0.76	0	0	0	0.06	0.23
t_8	0.06	0	0.06	0.18	0.21	0.15	0.45	0.16	0.16	0	0.19	0.41
t_9	0	0	0	0.38	0.44	0.12	0.69	0.22	0.22	0	0.25	0.58
t_{10}	0.14	0	0.14	0.47	0.37	0.17	0.72	0.13	0.13	0	0.08	0.66

(b)

Time	Node5			Node6			Node7		
	vul1	vul2	vul3	tot	vul1	vul2	tot	vul1	tot
t_1	0.44	0.34	0.11	0.7	0.66	0.34	0.97	0.74	0.85
t_2	0.23	0.19	0.25	0.61	0.34	0.64	0.79	0.16	0.46
t_3	0.12	1	0.65	1	0.62	0.35	0.84	0.26	0.52
t_4	0.29	0.05	0.21	0.58	0.15	0.84	0.87	0.69	0.81
t_5	0.56	0.12	0.76	0.94	0.08	0.71	0.78	0.14	0.58
t_6	0.18	0.2	0.43	0.85	0	0.34	0.54	0.71	0.89
t_7	0.34	0.14	0.23	0.75	0.15	0	0.52	0.29	0.72
t_8	0.12	0.08	0.05	0.54	0.35	0	0.74	0.03	0.55
t_9	0.74	0.66	0.01	0.92	0.28	0	0.41	0.47	0.66
t_{10}	0	0	0	0	0.91	0	0.93	0.18	0.37

TABLE 4: The final scores.

Time	Node1		Node2		Node3		Node4		Node5		Node6		Node7		score
	p	w	p	w	p	w	p	w	p	w	p	w	p	w	s
t_1	0.3	0.2	0.27	0.1	0.34	0.1	0.43	0.2	0.7	0.1	0.97	0.1	0.85	0.2	0.544
t_2	1	0.2	0.79	0.1	0	0.1	0.9	0.2	0.61	0.1	0.79	0.1	0.46	0.2	0.691
t_3	0.34	0.2	0.25	0.1	0	0.1	0.67	0.2	1	0.1	0.84	0.1	0.52	0.2	0.515
t_4	0.17	0.2	0.3	0.1	0.15	0.1	0.79	0.2	0.58	0.1	0.87	0.1	0.81	0.2	0.544
t_5	0.22	0.2	0.64	0.1	0.06	0.1	0.86	0.2	0.94	0.1	0.78	0.1	0.58	0.2	0.574
t_6	0.3	0.2	0.69	0.1	0	0.1	0.47	0.2	0.85	0.1	0.54	0.1	0.89	0.2	0.54
t_7	0.51	0.2	0.76	0.1	0	0.1	0.23	0.2	0.75	0.1	0.52	0.1	0.72	0.2	0.495
t_8	0.06	0.2	0.45	0.1	0.16	0.1	0.41	0.2	0.54	0.1	0.74	0.1	0.55	0.2	0.393
t_9	0	0.2	0.69	0.1	0.22	0.1	0.58	0.2	0.92	0.1	0.41	0.1	0.66	0.2	0.472
t_{10}	0.14	0.2	0.72	0.1	0.13	0.1	0.66	0.2	0	0.1	0.93	0.1	0.37	0.2	0.412

time t_1 , we compute each probability of vulnerability being exploited, that is, 0.25 and 0.06. After that we compute the probability of node 1 being attacked, that is, 0.3. Similarly, we can get the probability of being attacked for each node at time t_1 . Finally, we measure the network security through (16) and the final score of the system at time t_1 to t_{10} is shown in Table 4.

7. Conclusion

This paper focuses on measuring the network security with unknown threats. Although there are a lot of research studies on network security metrics, most of them are not ideal when faced with unknown threats. To help administrator have a better understanding about the potential zero-day attack, we analyzed the behavior of attackers and predict their attack timing. Due to the stealth and persistence feature, we modeled the zero-day attack as a long-term game. We specified and computed all the key parameters during the game and then got the Nash equilibrium for each subgame. We use these results as important references to measure the network security. The experiment showed the efficiency of our approach.

Main Parameters and Descriptions

C_a^i :	Attacking cost
C_d^i :	Defending cost
P_a :	Real passive-defending capability of the administrator
P_b :	Real initiative-defending capability of the administrator
$P_a^i(t)$:	The estimation of administrator's passive-defense capability at time t
$P_b^i(t)$:	The estimation of administrator's initiative-defense capability at time t
L :	Resource lifecycle
$g_n^i(t)$:	The gain function of time for certain attacker n
$E_n^i(t)$:	Profit expectation from time t to the end of lifecycle.

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 work was supported by the National Key R&D Program of China (no. 2016YFB0800702) and DongGuan Innovative Research Team Program (no. 201636000100038).

References

- [1] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "K-zero day safety: a network security metric for measuring the risk of unknown vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 1, pp. 30–44, 2014.
- [2] Y. Sun, L. Yin, Y. Guo, F. Li, and B. Fang, "Optimally selecting the timing of zero-day attack via spatial evolutionary game," in *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*, pp. 313–327, Springer, 2017.
- [3] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, "A survey on systems security metrics," *ACM Computing Surveys*, vol. 49, no. 4, article no. 62, 2016.
- [4] S. Hardy, M. Crete-Nishihata, K. Kleemola et al., "Targeted threat index: Characterizing and quantifying politically-motivated targeted malware," in *Proceedings of the in USENIX Security Symposium*, pp. 527–541, 2014.
- [5] U. Thakore, *A quantitative methodology for evaluating and deploying security monitors [Ph.D. thesis]*, 2015.
- [6] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, vol. 8688, pp. 1–21, Springer, 2014.
- [7] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: a survey of common practices," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–41, 2015.
- [8] N. Boggs, S. Du, and S. J. Stolfo, "Measuring drive-by download defense in depth," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, vol. 8688, pp. 172–191, Springer, 2014.
- [9] X. De Carné De Carnavalet and M. Mannan, "A large-scale evaluation of high-impact password strength meters," *ACM Transactions on Information and System Security*, vol. 18, no. 1, 2015.
- [10] B. Ur, S. M. Segreti, L. Bauer et al., "Measuring real-world accuracies and biases in modeling password guessability," in *Proceedings of the USENIX Security Symposium*, pp. 463–481, 2015.
- [11] B. R. Avasarala, J. C. Day, D. Steiner et al., "System and method for automated machine-learning, zero-day malware detection," US Patent 9,292,688, March 2016.
- [12] A. Mishra and B. B. Gupta, "Hybrid solution to detect and filter zero-day phishing attacks," in *Proceedings of the In Proceedings of the Second International Conference on Emerging Research in Computing, Information*, pp. 373–379, 2014.
- [13] L. Wang, M. Zhang, S. Jajodia, A. Singhal, and M. Albanese, "Modeling network diversity for evaluating the robustness of networks against zero-day attacks," in *Proceedings of the European Symposium on Research in Computer Security*, pp. 494–511, 2014.
- [14] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016.
- [15] A. Ekelhart, E. Kiesling, B. Grill, C. Strauss, and C. Stummer, "Integrating attacker behavior in IT security analysis: a discrete-event simulation approach," *Information Technology and Management*, vol. 16, no. 3, pp. 221–233, 2015.
- [16] O. Al-Jarrah and A. Arafat, "Network intrusion detection system using attack behavior classification," in *Proceedings of the*

5th International Conference on Information and Communication Systems, ICICS 2014, pp. 1–6, April 2014.

- [17] R. Mitchell and I.-R. Chen, “Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 5, pp. 593–604, 2014.
- [18] L. Allodi and F. Massacci, “Comparing vulnerability severity and exploits using case-control studies,” *ACM Transactions on Information and System Security*, vol. 17, no. 1, article no. 1, 2014.
- [19] L. Allodi, F. Massacci, and J. M. Williams, “The work-averse cyber attacker model,” 2016.
- [20] K. Nayak, D. Marino, P. Efstathopoulos, and T. Dumitraş, “Some Vulnerabilities Are Different Than Others,” in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, vol. 8688, pp. 426–446, Springer International Publishing.
- [21] S. Mitra and S. Ransbotham, “Information disclosure and the diffusion of information security attacks,” *Information Systems Research*, vol. 26, no. 3, pp. 565–584, 2015.
- [22] T. Dumitraş, “Understanding the vulnerability lifecycle for risk assessment and defense against sophisticated cyber attacks,” in *Cyber Warfare*, pp. 265–285, Springer, 2015.
- [23] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond heuristics: Learning to classify vulnerabilities and predict exploits,” in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD-2010*, pp. 105–113, ACM, July 2010.
- [24] X. Liang and Y. Xiao, “Game theory for network security,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 472–486, 2013.
- [25] W. Jiang, B.-X. Fang, Z.-H. Tian, and H.-L. Zhang, “Evaluating network security and optimal active defense based on attack-defense game model,” *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 32, no. 4, pp. 817–827, 2009.
- [26] L. Bilge and T. Dumitras, “Before we knew it: An empirical study of zero-day attacks in the real world,” in *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012*, pp. 833–844, ACM, October 2012.
- [27] R. Axelrod and R. Iliev, “Timing of cyber conflict,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 4, pp. 1298–1303, 2014.
- [28] M. Shahzad, M. Z. Shafiq, and A. X. Liu, “A large scale exploratory analysis of software vulnerability life cycles,” in *Proceedings of the 34th International Conference on Software Engineering, ICSE 2012*, pp. 771–781, IEEE Press, June 2012.

Research Article

HAC: Hybrid Access Control for Online Social Networks

Fangfang Shan ^{1,2}, Hui Li,¹ Fenghua Li,^{3,4} Yunchuan Guo ³ and Ben Niu³

¹State Key Laboratory of Integrated Service Network, School of Cyber Engineering, Xidian University, Xi'an 710071, China

²School of Computer Science, Zhongyuan University of Technology, Zhengzhou 450000, China

³State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

⁴School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China

Correspondence should be addressed to Yunchuan Guo; guoyunchuan@iie.ac.cn

Received 23 October 2017; Accepted 2 April 2018; Published 17 May 2018

Academic Editor: Raymond Choo

Copyright © 2018 Fangfang Shan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The rapid development of communication and network technologies including mobile networks and GPS presents new characteristics of OSNs. These new characteristics pose extra requirements on the access control schemes of OSNs, which cannot be satisfied by relationship-based access control currently. In this paper, we propose a hybrid access control model (HAC) which leverages attributes and relationships to control access to resources. A new policy specification language is developed to define policies considering the relationships and attributes of users. A path checking algorithm is proposed to figure out whether paths between two users can fit in with the hybrid policy. We develop a prototype system and demonstrate the feasibility of the proposed model.

1. Introduction

Online social networks (OSNs) have attracted widespread popularity nowadays. Users can conveniently share personal information with friends via OSNs. More than 300 hours of videos are uploaded to YouTube and nearly 25 million photos are posted to Instagram every minute [1]. Considering the fact that sensitive information may be leaked, the protection of users' privacy becomes a challenging task. To address this problem, researchers have proposed the relationship-based access control (ReBAC) mechanism [2–5]. Resource owners specify access control policies on the basis of relationships between individual users. By maintaining the balance between ease of use and flexibility, ReBAC has been commonly applied in real OSN systems. It has also been recognized as one of the most straightforward and useful ways in protecting user's privacy.

With the development of mobile technologies, plenty of smart devices are connected to the network. These devices may generate a large amount of private information, such as location and health status, and then share the information through OSNs [6]. In general, mobile technology contributes the following features to the online social networks.

(i) More and more privacy information collected through smart mobile devices may be uploaded to online social networks.

(ii) Privacy information collected by smart mobile devices can be used for access control scheme of OSNs.

These features have brought about new challenges for access control schemes of online social networks. For example, Alice is shopping at Bergdorf Goodman in New York. She has recorded a video of the megamall with her Google glasses and published it in OSN to see if any female friends can give her some pieces of advice in choosing cosmetics. Perhaps some of her friends nearby may come to have lunch together. She does not want every friend to know what she is doing now, so friends who do not live in New York will not be granted access to this video. The widely used relationship-based access control methods cannot describe the attribute "location: in New York" and cannot meet Alice's needs. To provide finer-grained access control over private data generated by wearable devices or m-health, researchers should take attributes such as location, profession, and gender into consideration.

The access control mechanism named UURAC_A [4] was the first mechanism that took attributes of users into

consideration. Better expression ability and finer-grained access control policies are characteristic of it. However, the attribute-based policy is separated from the relationship-based one. In other words, the policy of $UURAC_A$ falls into two parts: the relationship-based policy and the attribute-based policy. For this reason, $UURAC_A$ can merely figure out common attributes of one or several users on the relationship path instead of specifically identifying different attribute of different users.

In this paper, we propose a hybrid access control model based on both attribute and relationship. It designs a new language of policy specification to specify policies based on attributes and relationships. Compared with the study of Cheng et al. [4], the policy specification language is characteristic of better expression effectiveness and easier usage. It presents detailed instructions on the policy specification language and several application examples. A path checking algorithm is proposed to find out whether paths between two users involved in OSNs would fit in with the attribute-based policy. The path checking algorithm is implemented to conduct experiments to validate the feasibility and performance of the scheme.

2. Related Work

As large amounts of private personal data are created by Web 2.0 applications, Carrie [7] summarized four technical requirements of access control mechanisms for social networks based on Web 2.0 technologies. He emphasized that the access control mechanisms for Web 2.0-based social networks should have characteristics of relationship-based, fine-grained, interoperability, and sticky policies and named it relationship-based access control (ReBAC).

To meet these requirements and protect privacy of social network users, researchers have proposed a variety of access control mechanisms for OSNs. These mechanisms are broadly divided into three categories. Methods of the first type leverage relationships between users and resources to constrain the access of privacy information. Some researchers introduce modal and hybrid logic into their access control model of OSNs. Others make use of cryptography to prevent unwanted access.

Most access control schemes made use of various relationships between users and resources to protect sensitive information in OSNs. In [8], the authors tried to define access control policies based on the type, depth, and trust level of relationships between web-based social networks (WBSNs) users. They proposed an access control model for WBSNs which is characterized by using certificates for verifying the authenticity of relationships and enforcing a rule-based access control approach at the client side. Carminati et al. [9] extended the mechanism presented in [8] by providing details on the enforcement of the access control model. They defined two protocols to verify the authenticity of relationships and analyzed the security of them. In [10, 11], Carminati et al. leveraged OWL, SWRL, and semantic web technologies to express filtering, administration, and authorization policies. They proposed an access control model to describe the relationships between users and resources. A relationship-based

access control model was proposed by Cheng et al. [12] which utilized user-to-user, user-to-resource, and resource-to-resource relationships to define access control policies. It can be used to capture controls of administrative activities of users together with other normal usage activities. In [13], the authors presented an object-to-object relationship-based access control model (OOReBAC) which leveraged relationships between objects to control access of them. In [14], the authors presented a graph-based access control model which can be used in various systems, not just social networks. It introduced concepts of path conditions and principal matching and has better policy expression ability and request evaluation efficiency.

With the development of semantic web technology, some researchers considered using modal and hybrid logic in their access control schemes of OSNs. Masoumzadeh and Joshi [15] presented a scheme of access control based on ontology that can be used on semantic web-based social networks to support both system and user policies. Fong et al. [16] formalized the privacy preservation mechanism of Facebook-style social network systems and proposed an access control model for them. The policies of this model are able to express access control requirements such as common friends and clique. Fong [17] defined a modal logic-based language to specify and do composition of ReBAC policies. He presented a case study of EHR systems to prove that ReBAC can be used in application domains other than social computing. In [3], the authors demonstrated that policy language proposed in [17] was incomplete and it was unable to express all ReBAC policies. They extended the policy language of [17] to identify vertex and support for disjoint intermediaries and proved it to be representationally complete. As an extension of modal logic, Bruns et al. [18] utilized hybrid logic to specify policies and enforce access control decisions in relationship-based access control approach. A fragment of hybrid logic was used to express complex relationship-based access control policies such as “at least three friends”. Several other works [19–21] also utilized the hybrid logic to support better expression capacity of access control needs.

Researchers then considered adopting cryptography and other technologies to the access control mechanisms of OSNs. Anwar and Fong [22] designed a visualization tool to show the result of policy configurations. In [23], the authors presented an access control mechanism to protect textual contents in online social networks which is enforced to be transparent to content publishers and readers. The proposed system leveraged automatic semantic annotation to analyze the semantics of the contents in order to generate different versions according to types of readers. Apart from relationship-based access control mechanisms, other works concentrated on security protocols that leverage cryptographic techniques to achieve access control goals [24–30].

3. HAC Model Foundation

This section presents the foundation of HAC, including the attributes in OSN, the social graph with attributes, and the model components.

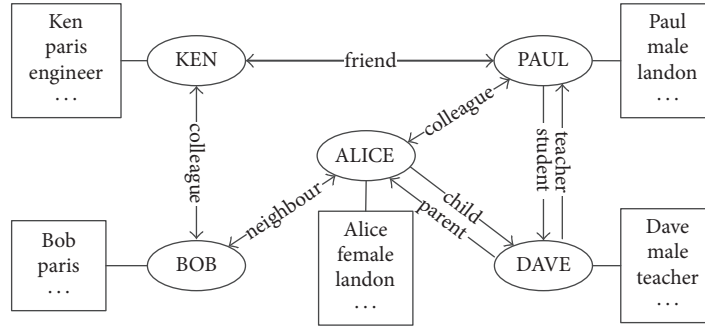


FIGURE 1: A sample social graph.

3.1. Attributes in OSN. Most of the recent access control schemes for OSN make access control decision based on relationships. By considering the relationships only, data owners cannot make proper access control policies based on location and time. Recent studies have shown that attribute-based access control (ABAC) can provide flexible and fine-grained access control in dynamic distributed systems [31–34]. As only the attributes of the subject, object, and environment are considered, most current solutions of typical ABAC schemes cannot be directly used in OSNs. Relationship of users should also be checked here. When registering an OSN account, users are always required to submit personal information, such as name and gender. This personal information is recognized as profile attribute, which can be used to define policies.

Attributes are categorized into profile attributes and relationship attributes in HAC.

Profile Attribute. Profile attribute contains information of environment and identity, or characteristics of users. In HAC, profile attributes fall into two types: user-defined attribute and objective attribute. The user-defined attributes are specified by the profile owner, such as gender, name, job, hometown, and hobbies. In contrast, the objective attributes are gained or defined by the OSN systems, such as time, location, and IP address.

Relationship Attribute. Relationship attribute is used to describe type, weight, and other information of relationships in OSNs.

3.2. Social Graph with Attributes. As shown in Figure 1, the researchers use a directed labeled simple graph G to abstract an OSN. The nodes of G represent users while edges represent the relationships between them. Each user is associated with a profile containing his profile attributes. Relationships between users are noted as relationship attributes. The social graph of HAC contains two types of information: (1) users and their profile attributes and (2) relationship attributes between the users.

The researchers use a triple $G = \langle N, \Sigma, E \rangle$ to describe the social graph in an OSN.

$N = \{\langle u, \text{pro_attr_group} \rangle \mid u \in U, \text{pro_attr_group} \in P\}$ denotes the set of nodes (or vertices) of the graph, containing

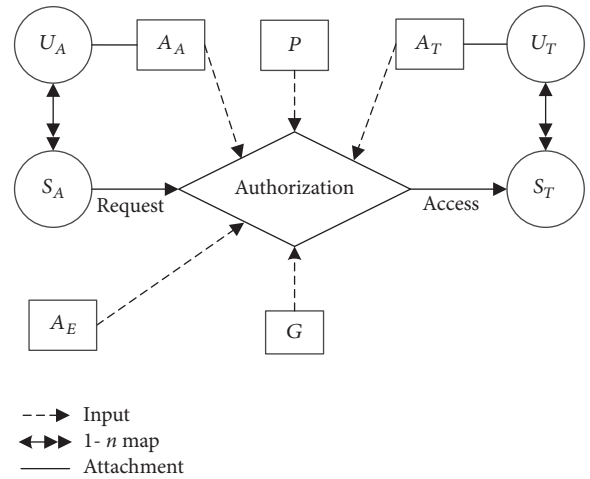


FIGURE 2: Model overview.

profile attribute information of users in an OSN. U represents user set while P is the set of profile attributes.

$\Sigma = \{r_1, r_2, \dots, r_n\}$ is the set of relationship attributes, each element of which represents a relationship attribute.

$E = N \times N \times \Sigma$ is the set of edges in the social graph, representing relationship attributes between users in the OSN.

3.3. HAC Model Components. Figure 2 shows the conceptual diagram of HAC. Model components are access requester, target user, access requester attribute, target user attribute, environment attribute, sessions of access requester and target user, policy, authorization, and social graph.

Access requester (u_a) is a registered human being who may initiate an access request to a profile or a resource of a target user in the OSN. Each access requester has a set of *attributes* (a_a) to describe his personal information, such as gender, name, job, and hometown. **Target user (u_t)** is a registered human being whose profile or resource is the recipient of access. Each target user is also related to a set of *attributes* (a_t). **Environment attribute (a_e)** represents environment information used in access control procedure, such as time, location, and IP address. **Session of access requester (s_a)** is an active instance of an access requester logged into the OSN system while **session of target user (s_t)** represents

the active instance of a target user. *Policy* (p) is defined by the target user based on various attributes governing the access of his profile or resources. According to different kinds of attributes, policies are categorized into profile attribute-based policy and relationship attribute-based policy. *Social graph* (G) describes the relationships between users on social networks. It is denoted as a directed labeled simple graph. *Authorization* is an abstract function with attributes, request, social graph, and policy as inputs. It makes a decision to grant or deny the access of the target user. *Request* represents an access requester's initiation of access. It is described as a tuple $\langle u_a, operation, target \rangle$, where u_a is the access requester, $target$ may be a profile or resource of the target user, and $operation$ represents the access that can be performed on targets.

4. HAC Policy Specification and Evaluation

In this section, the researchers present policy language, policy specification, and the policy evaluation of HAC.

4.1. Policy Language. Policy of HAC is defined by the target user. It constrains the profile and relationship attributes of users along the relationship path. The policy language is defined as follows.

(i) N and E are user (or node) set and relationship (or edge) set, respectively.

(ii) $PN = \{n_1, n_2, \dots, n_s\}$ ($1 \leq s \leq S$) is the profile attribute name set for users (or nodes), where S is the number of profile attribute names.

(iii) $PV = \{v_1, v_2, \dots, v_m\}$ ($1 \leq m \leq M$) is the predefined profile attribute value set for users (or nodes), where M is the number of profile attribute values.

(iv) $RA = \{r_1, r_2, \dots, r_k\}$ ($1 \leq k \leq K$) is the predefined attribute set for relationships (or edges), where K denotes the number of relationship attributes.

(v) $AT(n)$ and $AT(e)$ are attributes of node n and edge e , respectively. $AT(n) \in PN \times PV$ is a binary relation on sets PN and PV , and $AT(e) \in RA$. The node attribute is a profile attribute while the edge attribute is a relationship attribute.

Profile attribute of a node is a binary relation on profile attribute name set and profile attribute value set. A profile attribute-based policy rule is composed of a profile attribute name, a relationship specifier, and a profile attribute value as shown below.

(i) *profile attribute name*, $AT(n)$, *profile attribute value*.

Note that the profile attribute-based policy rule is specified by the data owner. For example, policy rule $R1$ says, "the current user's profession must be teacher". Policy rule $R2$ requests the current user to be an adult.

$R1: profession = "teacher"$

$R2: age > 18$

A complete attribute-based policy rule is composed of one relationship attribute and several profile attributes as shown below.

(i) $[relationship_attribute_1, (profile_attribute_{11}; \dots; profile_attribute_{1n})] \dots [relationship_attribute_h, (profile_attribute_{h1}; \dots; profile_attribute_{hn})]$

For example, $R3$ indicates that "the requester must be a teacher living in New York, and he should be a friend of the

TABLE 1: Grammar for path sentences.

$path_sentence ::= path_word \mid path_word \text{ connector } path_word$
$connector ::= \vee \mid \wedge$
$path_word ::= "(" path \text{ "," } hop_count ")"$
$hop_count ::= num$
$Path ::= attr_specs \mid attr_specs \text{ path}$
$attr_specs ::= "[" rel_attr \text{ "," } pro_attr_group "]"$
$rel_attr ::= r_1 \mid r_2 \mid \dots \mid r_k \mid RA \text{ where } RA = \{r_1, r_2, \dots, r_k\}$
$pro_attr_group ::= pro_attr_pair \mid pro_attr_pair \text{ pro_attr_group}$
$pro_attr_pair ::= "(" pro_attr_name \text{ "," } pro_attr_value ")"$
$pro_attr_name ::= n_1 \mid n_2 \mid \dots \mid n_l \mid PN \text{ where } PN = \{n_1, n_2, \dots, n_l\}$
$pro_attr_value ::= v_1 \mid v_2 \mid \dots \mid v_m \mid PV \text{ where } PV = \{v_1, v_2, \dots, v_m\}$
$num ::= [0 - 9]^+$

data owner". $R4$ specifies a rule saying that "only the adult male colleagues of the owner can access the resource". In $R5$, the relationship attribute is not used and it is shown as "-", which indicates that the relationship is not constrained. $R5$ requires that the location of the requester should be London and the access time must between 2017-09-05 and 2017-10-05.

$R3: [f, (occupation = "teacher"; howetown = "New York")]$

$R4: [c, (gender = "male"; age > 18)]$

$R5: [-, (time \in "2017-09-05: 2017-10-05"; location = "London")]$

4.2. Policy Specification. Policies are evaluated according to the paths between the access requester and the target user in social graph. The access control policy is composed of an operation and a path sentence. As shown in Table 1, the syntax of the path sentence is defined with Backus-Naur Form (BNF).

A path sentence consists of several path words that are connected by connectors. Every *path word* is composed of *path* and *hopcount*. The path specifies the relationships from the access requester to the target user. The *hopcount* represents the maximal number of edges along the path.

Unlike UURAC and UURAC_A, our path is aiming at expressing policies based on attributes. The researchers use relationship attribute and profile attribute to express the restriction on users and the relationship between them.

Several examples are given to show how to use hybrid rules to express the access control need in OSNs.

Example 1 (relationship attribute and profile attribute policy). If Jim wants to allow some users to access his photos, those users should share a common friend named "Jack" with him and their occupation must be doctor. He can specify a policy like this:

$PI: \langle photo_access, ([f, (name = "Jack")]) [f, (occupation = "Doctor");], 2) \rangle$

If Jim wants to show his photos to his friend Jack or his colleagues who are interested in medicine, the policy can be specified as below:

```

(1) for all (attr_specs) of path
(2)   path_reg_exp = path_reg_exp + attr_spec.rel_attr
(3) if (attr_specs) is the last one on the path then
(4)   return path_reg_exp
(5) else
(6)   break

```

ALGORITHM 1: *RegularExpressionTrans(path)*.

P2: $\langle \text{photo_access}, ([f, (\text{name} = \text{"Jack"})], 1) \vee ([c, (\text{interest} = \text{"medicine"})], 1) \rangle$

For P1, the system has to figure out the basic path (*ff*, 2) according to the social graph and examine profile attributes of users along the path. If the access requester is a doctor and he has a common friend named “Jack” with Jim, he may get the permission. For P2, two kinds of access requesters can get the permission. First, the user’s name is equal to “Jack”, and an (*f*, 1) path is found between him and Jim. Second, the user is interested in medicine and there is a (*c*, 1) path in the social graph between him and Jim.

For each policy, the last attribute spec restrains the attributes of the access requester.

Example 2 (relationship attribute policy). Profile attributes of the following policy are empty. The policy specifies that coworkers of Jim’s friends can access his profile. Policies like this can capture UURAC policies.

P3: $\langle \text{profile_access}, ([f, (-)][c, (-)], 2) \rangle$

4.3. Policy Evaluation. Algorithms of policy evaluation are presented in this section. The algorithms are used to evaluate whether the access requests should be granted. The algorithms have to find a required path between the access requester and the target user according to the social graph. The required path found in the social graph may ensure that the relationships between the access requester and the target user can satisfy the hybrid policy.

Regular Expression Transformation Procedure. Relationship attributes can be extracted from the path to form a regular expression as shown in Algorithm 1. For a path (*attr_specs* | *attr_specs path*), the algorithm traverses each attribute specification in the path and gets the relation attribute. All relation attributes are then catenated to be a regular expression which is used to match the paths in the social graph.

Path Checking Algorithm. Algorithm 2 shows the path checking method of HAC. It takes the social graph *G*, the *path*, the number of relationship attributes’ limit *hopcount*, the source node *start*, and the target node *end* as input. It returns a Boolean value, of which the output T means the access request will be granted and F means denied. Here, the source node *start* and the target node *end* represent the target user and the access requester, respectively.

Similar to [8], the path checking method leverages a depth first search (DFS) to find the proper path in the social

```

(1) currentPath  $\leftarrow$  NIL; h  $\leftarrow$  0
(2) nodeHistory  $\leftarrow$  start
(3) path_reg_exp  $\leftarrow$  RegularExpressionTrans(path)
(4) if hopcount  $\neq$  0 then
(5)   return ADFS(start)

```

ALGORITHM 2: *PathCheck (G, path, hopcount, start, end)*.

graph. Without the limit of *hopcount*, DFS may search along a path in the social graph too deep to find the proper path. Operations in OSNs always occur between the people with close relationships. Limited with *hopcount*, DFS is suitable for our model. The researchers improve the DFS to cope with profile attribute check and name it ADFS.

The variable *currentPath* is used to hold the node sequence traversed from the source node *start* to the current node. Variable *h* is used to tell if the *currentPath* exceeds the limit of *hopcount*. All nodes traversed are recorded by variable *nodeHistory*. These variables are initialized with NIL, 0, and the source node *start*, respectively. The main procedure gets regular expression of relationship attributes through *RegularExpressionTrans(path)*. Then, it launches traversal function ADFS() with parameter *start* to find out if the proper path exists in the social graph.

The function ADFS() is shown in Algorithm 3. It takes *u* as the only parameter. If the algorithm takes a further step from the node *u* and makes *h* + 1 bigger than the *hopcount*, it returns F. Otherwise, the further step is legal. ADFS() picks up one edge (*u*, *v*, σ) starting with node *u* in the social graph. Then, the algorithm faces five cases. For *if* 1, the current target node *v* belongs to the variable *currentPath*. This means that the edge (*u*, *v*, σ) has been visited. The algorithm breaks from the loop. For *if* 2, the node *v* is unvisited and it is exactly the target node *end*. ADFS() first checks whether the relationship attribute of the current edge (*u*, *v*, σ) is equal to the *d*th regular expression of relationship attributes extracted from the attribute-based access control policy. If not, the algorithm breaks. Otherwise, it means that the path between *start* and *end* matching the regular expression is found. If the profile attribute of node *v* fits the requirement of the attribute-based access control policy, the algorithm sets *h* to be *h* + 1 and concatenates the edge to *currentPath*. Then, it saves the node *v* in *nodeHistory*. In *if* 3, the node *v* is unvisited, and it is exactly the target node *end*. But the relationship attribute of the current edge (*u*, *v*) is not equal to the *h*th regular expression of relationship attributes extracted from the attribute-based access control policy. The algorithm will break from the current loop. In *if* 4, node *v* is unvisited and it is not the target node *end*. The relationship σ is not *path_reg_exp[h]*, and the algorithm breaks from *if* 4. In *if* 5, node *v* is unvisited, and it is not the target node *end*. The relationship attribute of the current edge (*u*, *v*, σ) is equal to the *d*th regular expression of relationship attributes. The algorithm sets *h* to be *h* + 1, concatenates (*u*, *v*, σ) to *currentPath*, sets *v* to be *currentNode*, and saves the current node to *nodeHistory*. Then, the function ADFS() is called recursively from node *v*. If a matching path is found, the algorithm will return T from


```

(1) if  $h + 1 > \text{hopcount}$  then
(2)   return F
(3) else
(4)   for all  $(v, \sigma)$  where  $(u, v, \sigma)$  in  $G$  do
(5)     if  $1 (v \in \text{currentPath})$ 
(6)       break
(7)     if  $2 ((v \notin \text{currentPath}) \ \&\& \ (v = \text{end}))$ 
(8)       if  $(\text{path\_reg\_exp}[h] \neq \sigma)$  then
(9)         break
(10)      if  $(! \text{match}(\text{path.attr\_spec}[h].\text{pro\_attr\_group}, v.\text{pro\_atr\_group}))$  then
(11)        break
(12)       $h \leftarrow h + 1$ ;  $\text{currentPath} \leftarrow \text{currentPath} \cup (u, v, \sigma)$ 
(13)       $\text{currentNode} \leftarrow v$ 
(14)       $\text{nodeHistory} \leftarrow \text{nodeHistory} \cup (\text{currentNode})$ 
(15)      return T
(16)    if  $3 ((v \notin \text{currentPath}) \ \&\& \ (v = \text{end}) \ \&\& \ (\text{path\_reg\_exp}[h] \neq \sigma))$ 
(17)      break
(18)    if  $4 ((v \notin \text{currentPath}) \ \&\& \ (v \neq \text{end}) \ \&\& \ (\text{path\_reg\_exp}[h] \neq \sigma))$ 
(19)      break
(20)    if  $5 ((v \notin \text{currentPath}) \ \&\& \ (v \neq \text{end}) \ \&\& \ (\text{path\_reg\_exp}[h] = \sigma))$ 
(21)       $h = h + 1$ ;  $\text{currentPath} \leftarrow \text{currentPath} \cup (u, v, \sigma)$ 
(22)       $\text{currentNode} \leftarrow v$ 
(23)       $\text{nodeHistory} \leftarrow \text{nodeHistory} \cup (\text{currentNode})$ 
(24)      if  $(\text{ADFS}(v))$  then
(25)        return T
(26)      else
(27)        break
(28)    if  $h = 0$  then
(29)      return F
(30)    else
(31)       $h = h - 1$ ;  $\text{currentPath} \leftarrow \text{currentPath} \setminus (u, v, \sigma)$ 
(32)       $\text{nodeHistory} \leftarrow \text{nodeHistory} \setminus \text{currentNode}$ 
(33)    return F

```

ALGORITHM 3: ADFS(u).

if 5. Otherwise, the algorithm will set h to be $h - 1$, remove (u, v, σ) from currentPath , remove v from nodeHistory , and move to another edge from node u .

The algorithm will test all paths from start to end . If any of them fits the access control policy, the algorithm will return T. Otherwise, it may return F as the depth of each path which will be checked is constrained by hopcount . We use min and max to represent the minimum and maximum out degree of node on the social graph, respectively. The time complexity of this algorithm will fall into the range of $O(\text{min}^{\text{hopcount}})$ and $O(\text{max}^{\text{hopcount}})$. The check of profile attribute will bring about extra overhead to the algorithm. In the first experiment, we evaluated this overhead which proved it to be acceptable. This experiment is presented in Section 5. So, the path checking algorithm of HAC is effective.

5. Implementation

This section presents the implementation of the path checking algorithm. Five sets of experiments are arranged to test the usability and performance of the algorithm. The researchers implement the algorithm in Java and store the social graph and sample access control policies in MySQL databases. All

the experiments are conducted on a machine with 4 GB memory and an Intel quad-core CPU at 3.6 GHz which runs the operating system of an Ubuntu 12.04 image.

5.1. Datasets. When selecting datasets in the organization of the experiments, there are two choices as reported in [35]: public available real datasets and synthetic datasets. As collected from real-world OSN systems, most of the public available datasets do not consider multiple relationship types or attribute information [36]. In order to meet the requirements, this necessary information should be added manually. However, if the dataset is modified, it may no longer present user behaviors [36]. Hence, synthetic dataset is a better choice for us in this evaluation. The researchers generate a random regular graph with n nodes, and each node has a fixed number of edges. Graphs with different nodes and edges can be created by changing parameters n and d according to the need of the experiments.

In the first experiment, to evaluate the effect of attribute evaluation on performance of the algorithm, the researchers test the time of ADFS to make an access control decision and compare it with the one without attribute support described in [36] (DFS). To do this, the researchers comment out the

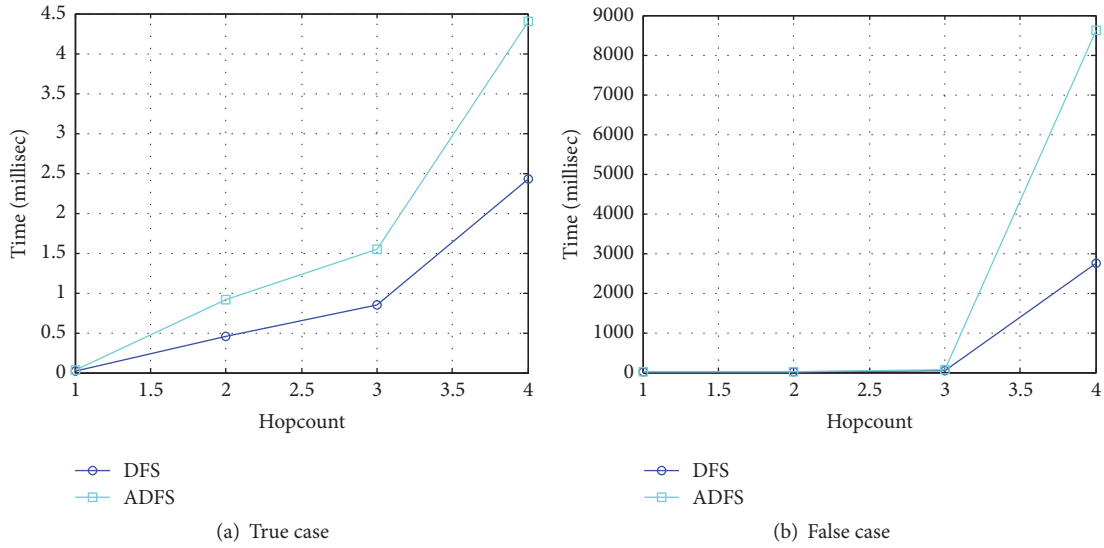


FIGURE 3: Time of path checking.

code of the profile attribute match. As discussed in [36], n is set to be 1000. On average, main users of Facebook have 173.6 friends as described in [37]. Then, the researchers set that each node has 174 randomly selected neighbors. Only one relationship attribute is considered in this experiment. Five profile attributes, such as user name, gender, career, date of birth, and hometown, are assigned to each node. The user names are set to be different from each other. The gender of each user is randomly selected from male and female. The researchers collected twenty different careers in the career set which are randomly assigned to each user in the social graph. The date of birth is randomly chosen between 1927 and 2007. The hometown of each node is randomly selected from a location set which includes twenty cities. Two sets of experiments are arranged. One set returns the true and the other returns the false. Each 4-hopcount policy runs five times over the two algorithms on 1000 nodes randomly selected from the node set. The average value of those 5000 runs is the final result.

Before the path checking algorithm is invoked, the relationship attributes should be extracted from the path to form a regular expression by the regular expression transformation procedure. This procedure is also called preprocessing. In order to confirm that the policy language is appropriate to be used in an attribute-based access control model, the researchers should make sure that the preprocessing would not take too much time compared with the ADFS algorithm. So, the time of preprocessing is evaluated in the second experiment where the parameters are set to be the same as those in the first experiment.

In the near future, OSNs may support more than one type of relationship. To evaluate if HAC can meet the access control needs of multiple types of relationships in OSNs, in the third experiment, the researchers discuss the variety of time with the increase of the relationship attribute types. The number of relationship attribute types varies from 1 to 8. Other parameters are the same as in the first experiment.

To evaluate how the scale of OSN will impact the performance of the algorithm, in the fourth experiment, the researchers test the variety of time with the number of nodes in the social graph. The number of nodes is set to be 1000, 2000, 5000, and 20000, respectively. The rest of the parameters are the same as in the first experiment.

To evaluate how the density of OSN will impact the performance of the algorithm, in the last experiment, the researchers examine the variety of time with the number of neighbors. The number of neighbors is set to be 100, 174, 200, and 500, respectively. The social graph becomes denser as the number grows. Other parameters are kept consistent with the first experiment.

5.2. Performance. DFS and ADFS algorithms are compared in the first experiment. The researchers consider four policies with different numbers of relationship attributes (hopcount) which varied from 1 to 4. Figure 3 presents the result of the experiment. It takes slightly more time for ADFS to make an access control decision than DFS does, as ADFS takes profile attribute check after finding a qualified relationship attribute. More relationship attributes mean more profile attributes should be checked. So, the time gap of those two algorithms increases as the number of relationship attributes grows. The slight increase of time in ADFS is acceptable, as the use of the profile attribute makes access control policy more powerful. In the false case experiments, it takes significantly more time to finish the path evaluation as more paths in the social graph have to be checked.

Figure 4 presents the result of the second experiment. It shows a comparison of the time of preprocessing and ADFS algorithm. Both of them increase along with the number of hopcounts. For each hopcount, the time of preprocessing is nearly a tenth of ADFS. This result is acceptable which indicates that the policy language is appropriate to be used in an attribute-based access control model.

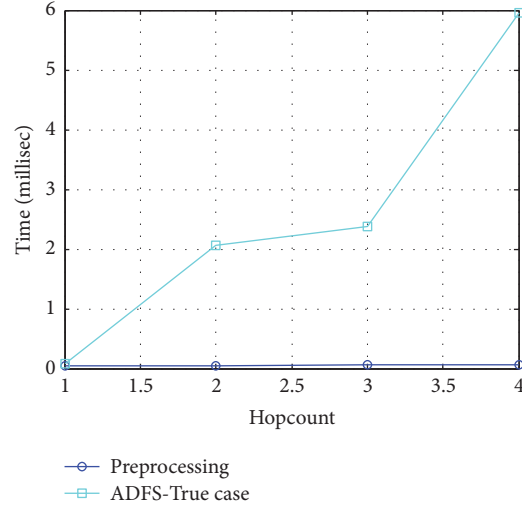


FIGURE 4: Time of preprocessing.

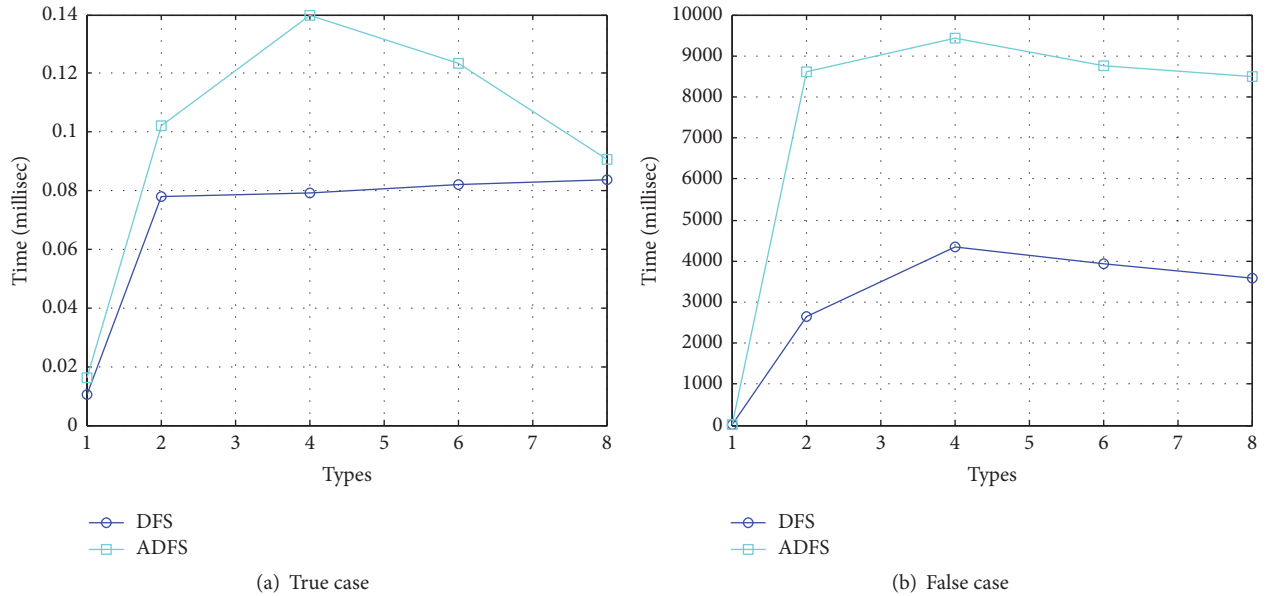


FIGURE 5: Time of path checking versus types (hopcount = 4).

The result of the third experiment is shown in Figure 5. Since people always tend to share information with friends within a close distance, 4-hopcount policies are used here. In both true and false cases, the time of ADFS to make access control decision increases linearly with the number of relationship attribute types. It reaches the peak value as 4 types of relationship attributes exist in the social graph. A larger number of relationship attributes may not affect the time of policy check as 4-hopcount policies are used in the experiment. The result means that our algorithm will work well with the future social networks as more relationship attribute types will be supported.

Figure 6 shows the result of the fourth experiment. Time of path checking grows with the increase of nodes from 1000 to 20000. In the true case, it takes no more than 1 millisecond to make the access control decision. For the false case, the

time it takes becomes longer because all possible paths should be checked.

The result of the last experiment is presented in Figure 7. In both true and false cases, the time of path checking increases as the social graph gets denser. The time grows sharply as the number of degrees exceeds 200. In most social network systems, the number of friends may be approximately 200 [37], so ADFS algorithm is feasible.

6. Comparison

This section discusses several related works of relationship-based access control schemes and compares HAC with [4, 5, 16, 36] (see Table 2).

The first column of Table 2 represents nine characteristics discussed in this section. The next three columns represent

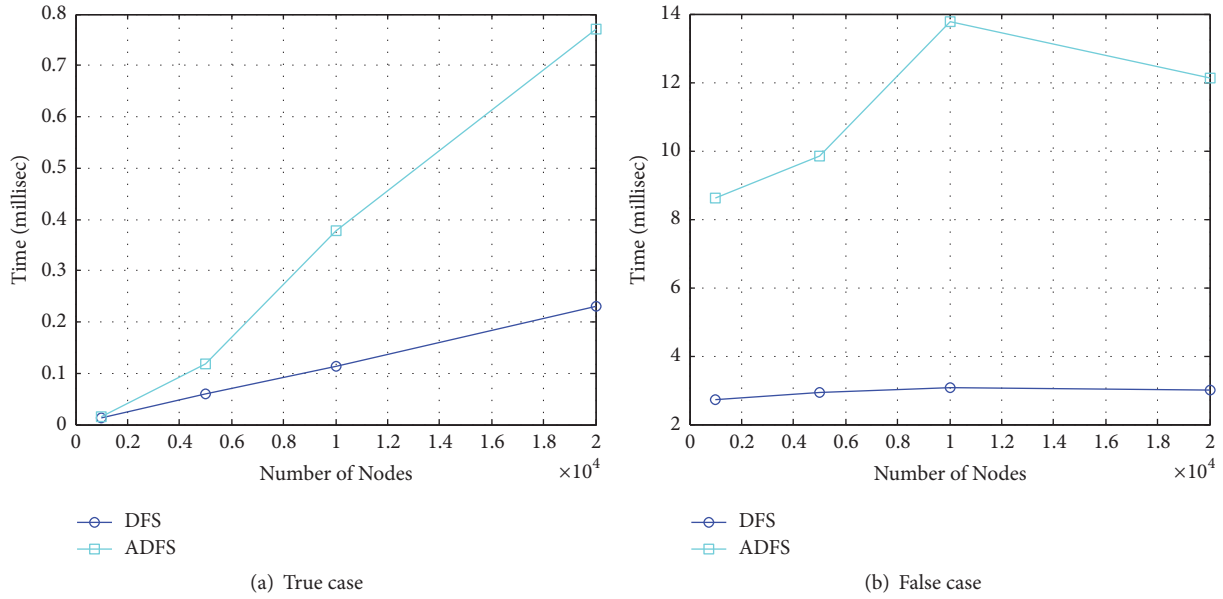


FIGURE 6: Time of path checking versus number of nodes (hopcount = 4).

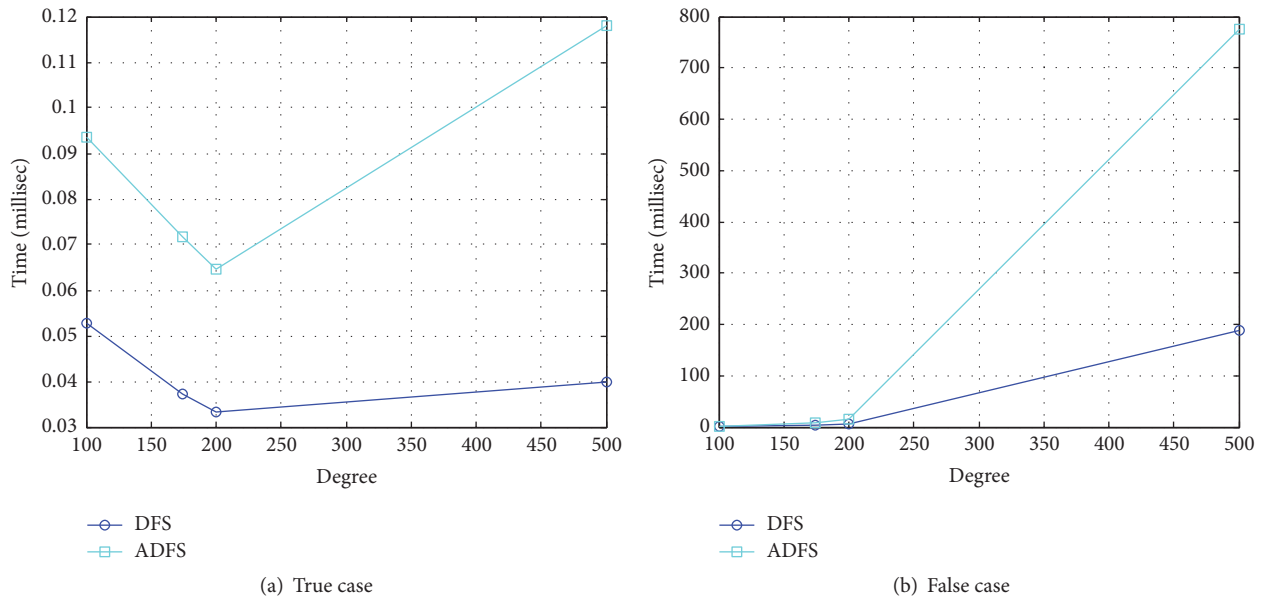


FIGURE 7: Time of path checking versus degree (hopcount = 4).

the characteristics of the access control schemes discussed above. Characteristics of HAC are listed in the last column.

The scheme in [16] is a formal algebraic access control model for Facebook-style systems. But user attributes and relationships beyond friendship are not supported in this model. OSNs access control models presented in [5, 36] have similar user graphs to HAC. However, these models do not explicitly take into account user attributes.

This work is similar to [4]. Despite its flexibility, UURAC_A [4] is still far from perfect. It does not support specific user attribute. More concretely, its policy specification language can merely figure out common attribute requirements of one or several users on the relationship path, lacking specification

ability of different attribute requirements of different users along the path. Additionally, it cannot describe some policies, such as “the adult colleagues of my friend Tom can access the resource”, which requires the attribute of my friend “name is Tom” and the attribute of the colleagues of my friend Tom “age > 18”. Besides, compared with UURAC_A, HAC is simple and easy to understand. It is easier for users in the OSNs to set up access control policies with HAC.

7. Conclusion

This research proposes an attribute and relationship-based hybrid access control model HAC for OSNs based on two

TABLE 2: Comparison.

	Fong et al. [16]	UURAC [5, 36]	UURAC _A [4]	HAC
Multiple Relationship Types		✓	✓	✓
User Profile Attributes			✓	✓
Specific User Attribute				✓
User-user Relationship	✓	✓	✓	✓
Directional Relationship		✓	✓	✓
Relationship Depth	✓	✓	✓	✓
Policy Individualization	✓	✓	✓	✓
Attribute Composition	none	none	attributes of user set	attributes of exact user
Relationship Description	ff	path pattern of different types	path pattern of different types	exact type sequence

aspects, including policy language and path checking. The policy language contributes to the literature on ReBAC by allowing users to specify spatial, temporal, and historical based policies with better expressiveness and flexibility. This research also presents several attribute and relationship-based hybrid policies and formally expresses them in the proposed policy language. Path checking algorithm enables users to figure out whether an access request can be satisfied. A prototype is implemented, and several experiments are evaluated to validate the feasibility of the scheme. HAC is advantageous compared with existing OSN access control models in terms of the expressiveness ability of policy language and the evaluation algorithm of access request.

In the future, researchers plan to improve the hybrid policy language to gain better expressiveness ability and support for more relationship types including one-to-many relationship and temporary relationship.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The research activities described in this paper have been conducted within the Research Project “the National Key Research and Development Program of China (2016YFB0801001)” and “General Program of National Natural Science Foundation of China (61672515)”.

References

- [1] J. Xiong, Y. Zhang, X. Li, M. Lin, Z. Yao, and G. Liu, “RSE-PoW: a role symmetric encryption PoW scheme with authorized deduplication for multimedia data,” *Mobile Networks and Applications*, pp. 1–14, 2017.
- [2] P. W. L. Fong, “Relationship-based access control: protection model and policy language,” in *Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, CODASPY '11*, pp. 191–201, February 2011.
- [3] P. W. L. Fong and I. Siahaan, “Relationship-based access control policies and their policy languages,” in *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, SACMAT '11*, pp. 51–60, June 2011.
- [4] Y. Cheng, J. Park, and R. Sandhu, “Attribute-aware relationship-based access control for online social networks,” in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 292–306, Springer, Berlin, Germany, 2014.
- [5] Y. Cheng, J. Park, and R. Sandhu, “A user-to-user relationship-based access control model for online social networks,” in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 8–24, Springer, Berlin, Germany, July 2012.
- [6] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, “Social attribute aware incentive mechanism for device-to-device video distribution,” *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1908–1920, 2017.
- [7] E. G. Carrie, “Access control requirements for web 2.0 security and privacy,” in *Proceedings of the IEEE Web 2.0 privacy and security workshop (W2SP '07)*, 2007.
- [8] B. Carminati, E. Ferrari, and A. Perego, “Rule-based access control for social networks,” in *Proceedings of the Move to Meaningful Internet Systems Workshops, OTM '06*, pp. 1734–1744, Springer, Berlin, Germany, 2006.
- [9] B. Carminati, E. Ferrari, and A. Perego, “Enforcing access control in Web-based social networks,” *ACM Transactions on Information and System Security*, vol. 13, no. 1, pp. 1–38, 2009.
- [10] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, “A semantic web based framework for social network access control,” in *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT 2009*, pp. 177–186, June 2009.
- [11] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, “Semantic web-based social network access control,” *Computers & Security*, vol. 30, no. 2-3, pp. 108–115, 2011.
- [12] Y. Cheng, J. Park, and R. Sandhu, “Relationship-based access control for online social networks: Beyond user-to-user relationships,” in *Proceedings of the Proceedings of the Privacy, Security, Risk and Trust*, pp. 646–655, 2012.
- [13] T. Ahmed, F. Patwa, and R. Sandhu, “Object-to-object relationship-based access control: Model and multi-cloud demonstration,” in *Proceedings of the 17th IEEE International Conference on Information Reuse and Integration, IRI '16*, pp. 297–304, July 2016.
- [14] J. Crampton and J. Sellwood, “Path conditions and principal matching: a new approach to access control,” in *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies, SACMAT '14*, pp. 187–198, 2014.
- [15] A. Masoumzadeh and J. Joshi, “OSNAC: an ontology-based access control model for social networking systems,” in *Proceedings of the 2nd IEEE International Conference on Social*

- Computing, SocialCom 2010, 2nd IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT '10*, pp. 751–759, August 2010.
- [16] P. W. Fong, M. Anwar, and Z. Zhao, “A privacy preservation model for facebook-style social network systems,” in *Proceedings of the European Symposium on Research in Computer Security*, pp. 303–320, Springer, Berlin, Germany, September 2009.
 - [17] P. W. Fong, “Relationship-based access control: Protection model and policy language,” in *Proceedings of the 1st ACM Conference on Data and Application Security and Privacy, CODASPY '11*, pp. 191–201, February 2011.
 - [18] G. Bruns, P. W. Fong, I. Siahaan, and M. Huth, “Relationship-based access control: its expression and enforcement through hybrid logic,” in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, pp. 117–124, February 2012.
 - [19] E. Tarameshloo and P. W. Fong, “Access control models for Geo-Social Computing systems,” in *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies, SACMAT '14*, pp. 115–126, June 2014.
 - [20] E. Tarameshloo, P. W. L. Fong, and P. Mohassel, “On protection in federated social computing systems,” in *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, CODASPY '14*, pp. 75–86, March 2014.
 - [21] M. Cramer, J. Pang, and Y. Zhang, “A logical approach to restricting access in online social networks,” in *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, SACMAT '15*, pp. 75–86, June 2015.
 - [22] M. Anwar and P. W. L. Fong, “A visualization tool for evaluating access control policies in facebook-style social network systems,” in *Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12*, pp. 1443–1450, Italy, March 2012.
 - [23] M. Imran-Daud, D. Sánchez, and A. Viejo, “Privacy-driven access control in social networks by means of automatic semantic annotation,” *Computer Communications*, vol. 76, pp. 12–25, 2016.
 - [24] B. Carminati and E. Ferrari, “Privacy-aware collaborative access control in web-based social networks,” in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 81–96, Springer, Berlin, Germany, July 2008.
 - [25] J. Domingo-Ferrer, A. Viejo, F. Sebé, and Ú. González-Nicolás, “Privacy homomorphisms for social networks with private relationships,” *Computer Networks*, vol. 52, no. 15, pp. 3007–3016, 2008.
 - [26] B. Carminati and E. Ferrari, “Enforcing relationships privacy through collaborative access control in web-based social networks,” in *Proceedings of the Proceedings of the Collaborative Computing: Networking, Applications and Worksharing*, pp. 1–9, November 2009.
 - [27] K. B. Frikken and P. Srinivas, “Key allocation schemes for private social networks,” in *Proceedings of the Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pp. 11–20, November 2009.
 - [28] G. Mezzour, A. Perrig, V. Gligor, and P. Papadimitratos, “Privacy-preserving relationship path discovery in social networks,” in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 189–208, Springer, Berlin, Germany, December 2009.
 - [29] M. Xue, B. Carminati, and E. Ferrari, “P3D-privacy-preserving path discovery in decentralized online social networks,” in *Proceedings of the Computer Software and Applications*, pp. 48–57, July 2011.
 - [30] M. Backes, M. Maffei, and K. Pecina, “A Security API for Distributed Social Networks,” in *Proceedings of the NDSS*, vol. 11, pp. 35–51, February 2011.
 - [31] X. Jin, R. Krishnan, and R. Sandhu, “A unified attribute-based access control model covering DAC, MAC and RBAC,” in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 41–55, Springer, Berlin, Germany, July 2012.
 - [32] H.-B. Shen and F. Hong, “An attribute-based access control model for web services,” in *Proceedings of the 7th International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT '06*, pp. 74–79, December 2006.
 - [33] E. Yuan and J. Tong, “Attributed based access control (ABAC) for web services,” in *Proceedings of the Web Services*, pp. 561–569, July 2005.
 - [34] X. Li, S. Tang, L. Xu, H. Wang, and J. Chen, “Two-Factor Data Access Control With Efficient Revocation for Multi-Authority Cloud Storage Systems,” *IEEE Access*, vol. 5, pp. 393–405, 2017.
 - [35] B. Carminati, E. Ferrari, and J. Girardi, “Performance analysis of relationship-based access control in OSNs,” in *Proceedings of the 13th International Conference on Information Reuse and Integration, IRI '12*, pp. 449–456, August 2012.
 - [36] Y. Cheng, J. Park, and R. Sandhu, “An access control model for online social networks using user-to-user relationships,” *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 4, pp. 424–436, 2016.
 - [37] S. W. Lee and J. Lee, “A comparative study of KakaoStory and facebook: focusing on use patterns and use motives,” *Telematics and Informatics*, vol. 34, no. 1, pp. 220–229, 2017.

Review Article

Data Fusion for Network Intrusion Detection: A Review

Guoquan Li,¹ Zheng Yan ^{1,2}, Yulong Fu ¹ and Hanlu Chen¹

¹State Key Laboratory of ISN, School of Cyber Engineering, Xidian University, Xi'an, China

²Department of Communications and Networking, Aalto University, Espoo, Finland

Correspondence should be addressed to Yulong Fu; ylfu@xidian.edu.cn

Received 20 December 2017; Revised 26 March 2018; Accepted 11 April 2018; Published 15 May 2018

Academic Editor: Jesús Díaz-Verdejo

Copyright © 2018 Guoquan Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Rapid progress of networking technologies leads to an exponential growth in the number of unauthorized or malicious network actions. As a component of defense-in-depth, Network Intrusion Detection System (NIDS) has been expected to detect malicious behaviors. Currently, NIDSs are implemented by various classification techniques, but these techniques are not advanced enough to accurately detect complex or synthetic attacks, especially in the situation of facing massive high-dimensional data. Besides, the inherent defects of NIDSs, namely, high false alarm rate and low detection rate, have not been effectively solved. In order to solve these problems, data fusion (DF) has been applied into network intrusion detection and has achieved good results. However, the literature still lacks thorough analysis and evaluation on data fusion techniques in the field of intrusion detection. Therefore, it is necessary to conduct a comprehensive review on them. In this article, we focus on DF techniques for network intrusion detection and propose a specific definition to describe it. We review the recent advances of DF techniques and propose a series of criteria to compare their performance. Finally, based on the results of the literature review, a number of open issues and future research directions are proposed at the end of this work.

1. Introduction

Network Intrusion Detection System (NIDS) is a new generation of network security equipment following the traditional security measures such as firewall and data encryption [1], which has been rapidly developed in recent years. It successfully resists many attacks and malicious actions and is called the second line of defense in the Internet. However, in the current big data era, the large amount of traffic data makes NIDS face critical challenges. First, large amounts of high-dimensional data increase processing complexity and need huge computing and storage resources. Second, many redundant and unrelated data could adversely affect network security detection. Third, some new attacks are difficult to detect due to big data process and analytics. Besides, the inherent weakness of NIDSs, such as high false positives (FP) and high false negatives (FN), raises urgent requests on effective solutions. Data Fusion (DF), as a promising technology of big data, has been applied into the domain of network intrusion detection to overcome the above-mentioned challenges in recent years.

The concept of DF originated from the US Air Force project; the US Department of Defense first proposed a Joint Directors of Laboratories (JDL) DF model based on national defense monitoring needs in 1987 [2]. Subsequently, DF was gradually studied and applied in other fields, such as automatic control, image recognition, target detection, and cyber security, and many scholars have proposed definition of DF based on their own studies and researches [3]. In order to clearly show the role of DF technology in network intrusion detection, an expression of DF in the field of NIDS is presented in this article.

In general, DF can be applied into three layers according to where fusions are needed, namely, data layer, feature layer, and decision layer. The data layer is the lowest system layer, playing the role of processing and integrating raw network data; the feature layer is the middle layer, fusing and reducing features of the preprocessed data; the decision layer is the highest layer, fusing and combining the inferences or decisions of various processing units. In the field of NIDS, most researches of data fusion only focus on the feature layer and the decision layer. It is because the network data they

need to fuse comes from public datasets that have already been fused at the data layer. The use of DF technology at the feature level can greatly reduce the size of data processing, thereby enhancing the efficiency of NIDSs. Besides, useful and refined data generated by feature fusion can support decision-making and further improve the robustness and accuracy of the system. As for using of DF technology at the decision level, the decision fusion center fuses the decisions of multiple local detectors to obtain more accurate and reliable identifications of network behaviors.

Currently, a lot of research work has been carried out on DF for intrusion detection in order to improve the performance of NIDS. However, we found that the open datasets, the number of experimental data samples, and the fusion techniques used in many literatures are diverse. It is difficult to understand and analyze the strengths and weaknesses of different fusion techniques. Thus, it becomes essential to specify uniform criteria to evaluate them in view of a large number of references and give performance statistics of the current literature. This work is meaningful because it can make it easier for researchers and practitioners to understand the characteristics of the current DF techniques and methods.

In this article, we provide a thorough review on DF techniques in NIDS. We first describe DF for NIDS by representing the process and role of fusion for motivating this research work. We review existing DF techniques used in intrusion detection and propose evaluation criteria to analyze and compare the characteristics and performance of different fusion techniques. Besides, we simply analyze different open network datasets that can be used for testing intrusion detection techniques. Based on our review, we put forward current main challenges and point out promising research directions in this field.

The main contributions of this survey are listed as follows.

- (1) We give a description of DF for NIDS in order to motivate related research in this field.
- (2) We propose a number of evaluation criteria for evaluating fusion techniques for network intrusion detection.
- (3) We further employ the proposed criteria to review the performance of different fusion techniques, which offers a good reference for scholars in the fields of network security and information fusion.
- (4) We propose the challenges and promising research directions of DF for network intrusion detection based on our review.

The remainder of this article is organized as follows. Section 2 gives a brief introduction about the background knowledge of NIDS and DF. Several commonly used fusion techniques are elaborated in Section 3. Section 4 puts forward the evaluation criteria of data fusion techniques based on a large amount of literatures. The power of different fusion techniques is analyzed and compared in Section 5. In Section 6, the existing issues of DF are discussed, and some promising research directions are proposed. Section 7 summarizes the whole article.

2. Background Knowledge

In order to better understand this article, this section introduces some basic theory, including network intrusion detection and DF. Network intrusion detection is an old topic that has been repeatedly studied. We mainly present two kinds of intrusion detection techniques, anomaly-based and misuse-based, and explain their advantages and disadvantages, separately. As regards DF, we introduce it from its source, definitions, levels, and applications and put forward a general DF framework for intrusion detection to facilitate intuitive understanding.

2.1. Network Intrusion Detection. NIDS is a kind of network security scheme that can monitor the network transmission in real time and alert or take corresponding measures when detecting some behaviors that threaten network security. Actually, NIDS can be regarded as a pattern of recognition system that can distinguish malicious attacks from normal network behaviors. Intrusion detection technology plays an important role in the process of identifying malicious behaviors. The intrusion detection techniques based upon data mining generally fall into two categories: misuse detection and anomaly detection [4, 5]. The misuse-based detection, also called signature-based detection, is based on known attack signatures. It usually uses the well-known attack signatures to match and identify attacks. The advantages and disadvantages of the misuse-based detection are as follows [6].

(1) Advantages

- (i) Fast and efficient detection of known attacks or specific attack tools.
- (ii) Detecting attacks without generating an overwhelming number of false alarms.
- (iii) Allowing system administrators, regardless of their security skills, to track their system security issues and run exception handlers.

(2) Disadvantages

- (i) Hard to detect novel or unknown attacks.
- (ii) Hard to detect the variants of known attacks.

Due to the efficient detection and low false positive rate (FPR), the misuse-based IDSs are widely used in commercial networks. Furthermore, much excellent open-source software has also been implemented, typically represented by Snort. The Snort IDS is one of the commonly used misuse-based NIDSs, which performs real-time traffic analysis, content searching, and content matching to discover attacks using preidentified attack signatures [7]. It is popular with many researchers because of its open source and adaptability to various platforms. In [1], Tian et al. fused the alerts through Snort to test the performance of their proposed detection fusion system.

Although the misuse-based detection is efficient, it can only detect known attacks and cannot detect novel or zero-day attacks [38]. To detect novel attacks, the anomaly-based NIDS have been proposed. In many related literatures,

most of the network behaviors acquired by researchers are normal, so NIDSs usually use the anomaly-based detection techniques. Anomaly detection is a recognition model based on normal behaviors of the network connections. Any deviation from the established pattern of normal behaviors is considered to be a suspicious action. The anomaly detection seems to be able to detect all types of attacks, including unknown attacks. However, it indicates that some activities are suspicious but not malicious, resulting in high FP [39]. The advantages and disadvantages of the anomaly-based detection are as follows [6].

(1) Advantages

- (i) It can detect novel or unknown attacks.
- (ii) It Produces information that can in turn be used to define signatures for misuse detectors.

(2) Disadvantages:

- (i) It requires extensive training data of network connections and behaviors.
- (ii) FPR is not ideal.

The misuse-based detection is efficient in detecting known attacks but cannot detect novel attacks, while the anomaly-based detection can detect unknown attacks but usually has a high FPR. Therefore, NIDS used only one of these two which could be limited in performance and scope of application. To avoid the above defects, many hybrid approaches have been proposed, which combine the advantages of both misuse and anomaly detection [40]. Hybrid intrusion detection technology can be divided into three categories as follows.

- (1) Anomaly-based detection on top of misuse-based detection
- (2) Misuse-based detection followed on top of anomaly-based detection
- (3) Misuse-based and anomaly-based detection in parallel

Zhang et al. [15] implemented a hybrid system through the following first approach. This hybrid system can be used to detect known intrusions in real time and to detect unknown intrusions offline. Generally, in the past two decades, NIDSs have been fully studied. Intrusion detection technologies continue to improve and update. The performance of NIDSs has been greatly optimized accordingly, but NIDSs still face many challenges. The use of DF technology in the field of NIDS is a very promising research direction, which holds great potential to deal with these challenges.

2.2. Data Fusion

2.2.1. Data Fusion Definition. The concept of DF first appeared and applied in the military field in the 1980s, with strong military characteristics, which was called “intelligence synthesis.” Joint Directors of Laboratories (JDL) defines DF

from the perspective of military applications as follows: DF is a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, complete and timely assessments of situations, threats, and their significance. Waltz and Llinas [41] supplemented and modified the above definition in their work, replaced the “position estimate” with the “state estimate,” and added the detection function, which gave the definition: data fusion is a multilevel and multifaceted process and mainly completes the detection, integration, correlation, estimation, and combination of data from single and multiple data sources. Its purpose is to achieve an accurate estimate of the status and identity of the target and to make a complete and timely assessment of the situation and threats. Many other DF definitions are presented by some scholars based on their own researches and analysis. Although these definitions give us inspiration and guidance to some extent, they are not exhaustive in a particular area. A more specific expression of DF in the field of intrusion detection is beneficial to researchers within the field and motivates their own work. Therefore, based on these facts, we presented a specific description of DF in NIDS: “single source or multisource data collected from the network is preprocessed to obtain a uniform data format. More refining data of greater quality is obtained through feature fusion and association, which greatly improves the identification of malicious network behaviors. The initial decisions generated from multisource data are integrated in a decision fusion center to achieve more accurate and comprehensive inferences or decisions.” This expression is based on network intrusion detection; the goal of DF is to improve efficiency, accuracy rate (ACC) and robustness while reducing FNR and FPR, saving computing resources of system. We believe that the proposed definition is helpful to practitioners and researchers in the field of intrusion detection.

2.2.2. Data Fusion Levels. The data fusion is mainly applied at three levels with respect to the processing stage of the fusion [42]. Normally, three main levels are discerned: data, feature, and decision. At different levels, the representation of information is different: the outputs of the data level fusion and the feature level fusion are the “states,” “characteristics,” and “attributes,” and the outputs of the decision level fusion are “inferences” or “decisions.” Different fusion techniques and methods are usually used in different levels to improve overall performance of data processing.

The brief description of fusion levels is shown as below.

- (1) Data level fusion: it is also called low level fusion, which combines several different raw data sources to produce refined data that is expected to be more informative and synthetic.
- (2) Feature level fusion: it combines many data features and is also known as intermediate level fusion. The objective of feature fusion is to extract or select a limited number of important features for subsequent data analysis through feature reduction methods, which can reduce computation and memory resources.

- (3) Decision level fusion: it is also called high level fusion, which fuses decisions coming from multiple detectors. Each detector completes basic detection locally including preprocess, feature reduction, and identification to establish preliminary inferences on observed objectives. And then these inferences are fused into a comprehensive and accurate decision through the decision fusion techniques.

2.2.3. Data Fusion Applications. As a technology, DF is a multidisciplinary research field with a wide range of potential applications in such areas as automatic control, image recognition, target detection, and intrusion detection. The following is a brief introduction to DF applications based on the review of some related literatures.

In [43], Cao et al. presented a fire automation control system based on DF by applying it into intelligent building. The control system consists of six layers (sensor layer, sensor subsystem layer, primary fusion subsystem layer, decision management subsystem layer, actuator subsystem layer, and actuator layer). It can be applied into intelligent building to automatically realize accurate fire alarm and fire protection.

Zhang et al. proposed a DF based smart home control system [44]. The proposed smart home control framework includes the Internet access module, information acquisition module, and internal network service module with Bluetooth connection, data fusion controller that uses fuzzy logic and fuzzy neural network, and embedded computer in household appliances. It integrates information from multiple sources to control household appliances to create an intelligent home environment.

In [45], DF system based on D-S (Dempster-Shafer) evidence reasoning was proposed, in which two Charge Coupled Device (CCD) cameras and an Infrared Radiation (IR) sensor are used to extract the characteristics for identifying a missile target. Based on the D-S evidence reasoning, the authors recognized missile target and jamming light on region square feature and clutter and fire pile on position feature, respectively. The probability of identification obtained by integrating the three sensors with D-S evidence is greatly improved comparing with the method of using a single sensor.

Hu and Wang applied DF fuzzy theory to develop a fire alarm system based on a wireless sensor network [46]. This system not only offers detection correctness, but also improves the intelligence of monitoring. The proposed method has excellent performance and it is superior to traditional diagnostic methods with a single sensor.

In [47], a deep model for remote sensing DF and classification was proposed. The Convolutional Neural Network (CNN) is used to efficiently extract abstract information characteristics from Hyperspectral Image/Multispectral Image (MSI/HSI) and Light Detection and Ranging (LIDAR) data, respectively. Then, Deep Neural Networks (DNN) was used to fuse the heterogeneous characteristics obtained by CNN. The proposed depth fusion model provides competitive results in terms of classification accuracy. In addition, the proposed deep learning idea opens a new window for future remote sensing data fusion.

In [48], Yan et al. applied DF to reputation generation and proposed a reputation generation method based on opinion fusion and mining. The opinions were fused and classified into a number of major opinion sets containing opinions with similar or identical attitudes. Based on these opinion sets, the rating is aggregated to normalize the reputation of the entity. The experimental results from actual data analysis of several popular Chinese and English commercial websites demonstrated the versatility and accuracy of the method.

Liu et al. collected four articles to study the application of DF in the Internet of Things (IoT) [49]. With a large number of wireless sensor devices, IoT generates a large amount of data, which are massive, multisourced, heterogeneous, dynamic, and sparse. In the special issue, they believed that DF was an important tool for processing and managing these data to improve processing efficiency and provide advanced intelligence. By exploiting the synergy among the datasets, DF can reduce the amount of data, filter noise measurements, and make inferences at any stage of data processing in IoT.

A DF model for intrusion detection was presented in [42], based on clustering. The model uses a centralized approach to fuse data from different analyzers and then make a final analysis decision. The main strength of the proposed approach lies in its accuracy to fuse information from different detection modules and its adaptability to scalability. In addition, the DF module takes into account the efficiency of each analyzer in the process of fusion and can predict upcoming network threats.

2.2.4. A General Fusion Framework for Network Intrusion Detection. Herein, we specify a general fusion framework for network intrusion detection, as shown in Figure 1. The framework is comprised of the following parts.

(a) *Input/Data Source.* In order to monitor network status and detect and prevent attacks, we need to collect data from multiple sources in the network. These data include different types of packets and the statistical logs of network devices, for example, hosts, routers, and switches. They have different types and formats and cannot be processed directly.

(b) *Data Preprocessing.* The function of data preprocessing is to eliminate obviously wrong, invalid, or duplicate data and to get the valid data that can be used. The raw data is normalized and digitized through data preprocess, which is then converted into a unified format for analysis and processing.

(c) *Feature Fusion.* The network data has the characteristics of big data. Massive network data not only overly consumes computing and storage resources, but also cause dimensional disasters. Feature fusion occurs at the feature level and can reduce a large number of features to few features. The more streamlined data after feature fusion play a more important role in decision-making than the original features while accelerating data processing and increasing the detection accuracy of NIDS.

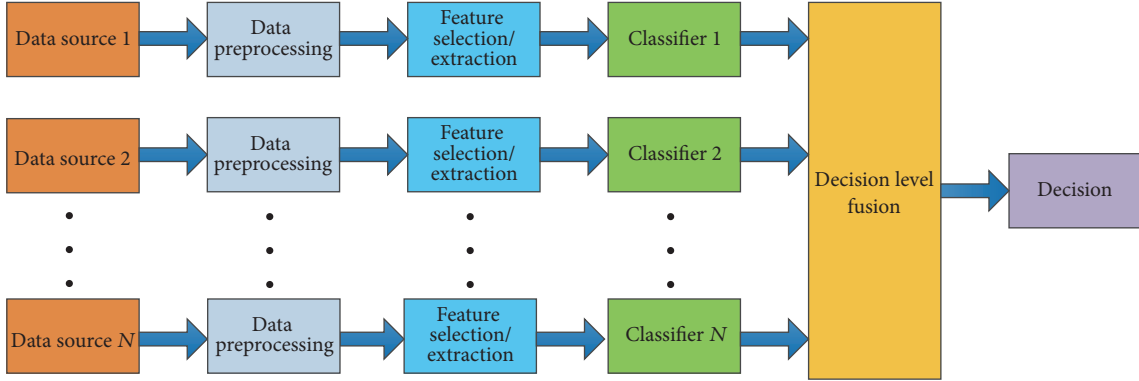


FIGURE 1: A general fusion framework for network intrusion detection.

(d) *Classification*. Intrusion detection can be seen as a pattern recognition system. Its performance is determined by the classifiers. Classifier models are obtained through training to identify abnormal network behaviors and make timely responses to the network attacks.

(e) *Decision Fusion*. Decision fusion is the integration of multiple results of basic detectors. The so-called decisions in the intrusion detection can be understood as the detection results of network behaviors. Decision fusion can achieve improved accuracy and more specific inference than the way of using a single detector alone. Besides, decision fusion can effectively detect complex attacks by integrating multiple decisions.

(f) *Output/Decision*. Output is the final decision, which usually is a judgment in the NIDS, either an abnormal behavior (e.g., an attack) or a normal behavior.

3. Data Fusion Techniques for NIDS

This section introduces the data fusion techniques, mainly focusing on feature fusion and decision fusion. We classify the fusion techniques shown in Figure 2 and describe the commonly used fusion techniques.

As mentioned above, DF techniques in NIDS can be classified into the data layer fusion, the feature layer fusion, and the decision layer fusion. To the best of our knowledge, the majority of researches on NIDS are based on open datasets, which leads to the result that the data level fusion is omitted in the related literatures. Therefore, we mainly review the DF techniques at the feature layer and the decision layer.

There are two main categories for feature fusion in NIDS: filters and wrappers [50]. The filters are applied through statistical methods, information theory based methods, or searching techniques [51], such as Principal Component Analysis (PCA), Latent Dirichlet Allocation (LDA), Independent Component Correlation Algorithm (ICA), and Correlation-Based Feature Selection (CFS). The wrapper uses a machine learning algorithm to evaluate and fuse features to identify the best subset representing the original dataset. The

wrapper is based on two parts: feature search and evaluation algorithms. The wrapper approach is generally considered to generate better feature subsets but costs more computing and storage resources than the filter [27]. The filter and the wrapper are two complementary modes, which can be combined. A hybrid method is usually composed of two stages. First, the filter method is used to eliminate most of the useless or unimportant features, leaving only few important ones, which can effectively reduce the size of data processing. In the second stage, the remaining few features representing the original data are used as input parameters to send into the wrapper to further optimize the selection of important features.

The decision fusion methods are divided into two classes: winner-take-all and weighted sum, by considering how to combine decisions from basic classifiers [32]. Majority vote, weighted majority vote, Naïve-Bayes, RF (Random Forest), Adaboost, and D-S evidence theories are classified as the type of winner-take-all because they all have measured values for each basic classifier and the final decision depends on the classifier with the highest measured value. In case of the weighted sum, the weight of each basic classifier depends on its own capabilities. The weights of basic classifiers are calculated, and then their outputs with the weights are added to give a final decision. The method of weighted sum mainly includes average and neural network. Figure 2 gives the categories of fusion techniques. In what follows, we briefly described several commonly used feature fusion and decision fusion techniques, respectively.

3.1. Feature Fusion Techniques. There are many types of feature fusion methods in the literature. We introduce some of them due to space limitations. Some classic fusion techniques are described below.

3.1.1. PCA. Principal Component Analysis (PCA) is a multivariate statistical technique used for feature reduction [12, 52]. The goal of PCA in intrusion detection is to extract n (small integer) most important features representing the dataset. It can achieve dimensionality reduction while removing noise from the data and improving the performance of

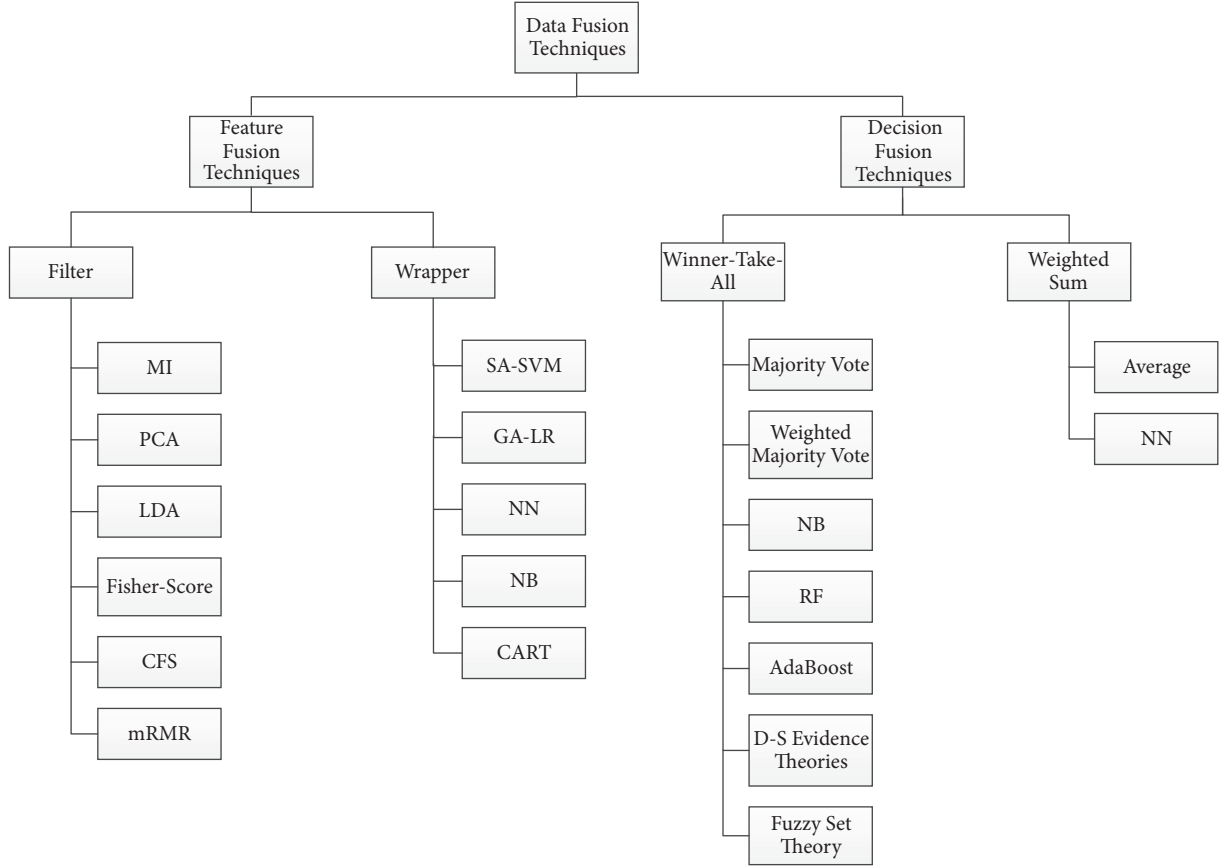


FIGURE 2: Categories of fusion techniques.

the system. In order to achieve these goals, PCA needs to extract new variables, that is, the main components. The first principal component has the largest variance that is the most representative of the entire dataset. The second principal component is computed under the constraint of being orthogonal to the first component and to have the largest possible variance. The other principal elements are calculated in the same way. These principal components form the new features of the original data. Before applying PCA, the data must be averaged and normalized to avoid the imbalance between the data values. PCA is popular in feature fusion because its simplicity and high precision. Nonetheless, in fact, each principal component can be represented by a linear combination of primitive features, which leads to a lack of interpretability for these principal components, especially when a large number of features are involved.

3.1.2. CFS. Correlation-Based Feature Selection (CFS) evaluates and ranks feature subsets rather than individual features [27]. It tends to have a set of attributes highly correlated with the class but with low intercorrelation. CFS often uses a variety of heuristic search strategies (such as hill climbing and best-first) to search a feature subset space within a reasonable time period. It first calculates the matrix of feature-class and the feature-feature correlation from the training data and

then uses best-first to search the feature subset space [50]. The equation for CFS is

$$M_s = \frac{kr_{cf}^-}{\sqrt{k + k(k-1)r_{ff}^-}}, \quad (1)$$

where M_s is the heuristic of the feature subset S containing k features, r_{cf}^- is the average value of all feature-classification correlations, and r_{ff}^- is the average value of all feature-feature correlations. The molecular kr_{cf}^- means the predictive ability of features, and the denominator $\sqrt{k + k(k-1)r_{ff}^-}$ indicates the redundancy between features.

3.1.3. GA. Genetic Algorithm (GA) is a search heuristic model for simulating natural selection processes [53]. This heuristic approach is often used to generate useful solutions for optimization and search problems. GA is a kind of Evolutionary Algorithm (EA), which uses natural evolution-inspired techniques (such as genetic, mutation, selection, and crossover) to generate solutions for optimizing results. We can use the evaluation function to calculate the goodness of each chromosome. This operation begins with the initial population of randomly generated generations of chromosomes,

and the quality of each individual is gradually increased. Each individual chooses three basic GA operators, namely, selection, crossover, and mutation. In intrusion detection, in the face of a large number of features of original data, the GA can search for a subset of the raw features through Support Vector Machine (SVM), Neural Networks (NN), or other classifiers as evaluation functions. The advantage of this approach is that it has a flexible and powerful global search capability that converges from multiple directions without regard to previous knowledge of system behaviors. The main drawback is the high consumption of computing resources.

3.2. Decision Fusion Techniques. Comparing with feature fusion, the level of decision fusion is higher, and the data to be merged is more abstract. The decision fusion further improves the performance of the detection system, especially when a single detector is difficult to identify complex network behaviors. In what follows, we introduce several common decision fusion techniques.

3.2.1. Weighted Majority Vote. Weighted majority vote can assign weights to each basic classifier, which indicates the importance of the outputs of different classifiers for a final decision [32]. The weight varies according to the ability of the basic classifier to separate the samples. The formula is as below.

$$\sum_{i=1}^L b_i d_{i,k}(x) = \max_{1 \leq j \leq c} \sum_{i=1}^L b_i d_{i,j}(x), \quad (2)$$

where $d = [d_{i,1}, \dots, d_{i,c}]^T \in \{0, 1\}^c$, $i = 1 \dots L$ is the outputs of the classifiers from the decision vector d , where L is the number of classifiers and $d_{i,j} = 1$ is 1 or 0 depending on whether classifier i chooses j , or not, respectively. The final decision to fuse multiple classifiers is determined by the base classifier's output $d_{i,j}(x)$ and corresponding weights b_i . This method assigns a higher weight to the basic classifier with higher accuracy, but it ignores other inaccurate base classifiers. The weights for the base classifiers are difficult to obtain and adjust. Therefore, it is difficult to detect new network attacks.

3.2.2. Bayesian Estimation. Bayesian estimation is applied to DF for a long time. It is an excellent method if prior probability is known. In order to obtain the most accurate and comprehensive information, this method first analyzes the compatibility of various sensors, removes false information with low confidence, and makes the Bayesian estimate of useful information under the assumption that the corresponding prior probabilities are known. The advantages of Bayesian approach include explicit uncertainty characterization and fast and efficient computation. Moreover, Bayesian networks offer good generalization with limited training data and easy maintenance when adding new features or new training data [23]. The disadvantage of Bayesian estimation is that it cannot distinguish unaware and uncertain information, and it can only handle the related events. In particular, it is difficult to know the prior probabilities in practical applications.

When the hypothetical prior probabilities are contradictory to reality, the results of the inference will be undesirable and will become quite complicated when dealing with multiple hypotheses and multiple conditions. In fact, the Bayesian inference methods are now rarely applied in DF because of these defects.

3.2.3. D-S Evidence Theory. The Dempster-Shafer evidence theory, abbreviated as D-S theory, is a complete theory of dealing with uncertainty. Its most notable feature is the usage of "interval estimates" rather than "point estimates" for the description of uncertainty information. It shows great flexibility in distinguishing between unknown and uncertain. These advantages make it widely applicable to information fusion, expert systems, intelligence analysis, and multiattribute decision analysis.

In the NIDS using the DS evidence theory, the results of each basic classifier are considered to be different "evidences." Different pieces of evidence of the same hypothesis (e.g., network connection categories, such as normal or attack) are integrated to obtain the supporting degree of the hypothesis. On the basis of the supporting degree, whether the network connection is normal or intrusion can be finally judged [31]. Zhao et al. used D-S theory to fuse several basic classifiers [33]. The correct rates of fused results in terms of every kind of intrusions are all close to, or even higher than, the highest correct rates of all basic detectors, which achieves a high correct rate to all intrusions. D-S Evidence Theory is considered as the generalization of the Bayesian theory. It can well represent "uncertainty" and does not need to know prior probabilities, compared with the Bayesian theory. Besides, it also has some drawbacks, such as the fact that the evidence is required to be independent and there is a potential exponential explosion in computation.

3.2.4. Neural Network. Neural Network (NN) is a supervised learning method that consists of input neurons, output neurons, and hidden neurons. In order to represent the relationship between the input neuron and the output neuron, the neural network needs a large amount of labelled data to train and obtain an accurate model. NN has the characteristics of self-learning, self-adaptation, self-organization, and fault-tolerant, which enable it to solve complex nonlinear problems. Furthermore, the advantage of NN is that it can automatically adjust the connection weights without any domain-specific knowledge, while other methods use preselected weights to combine outputs [32]. Therefore, its strong capabilities can be well adapted to the requirements of multisource DF in NIDS. In network intrusion detection, the classification results of multiple detectors are used as input neurons, and the output neurons are integrated classification results. The output of the neural network is used as feedback to adjust the training parameters. With the improved parameters, the detectors can be fused to produce an improved resultant output. The main drawback of NN is the lack of valid criteria for creating, selecting, and combining the results of the base classifiers. For example, one may use a Multilayer Perceptron (MLP) or a radial basis function to find fusion weights with different structure.

Please note that the DF techniques are not limited to the above-mentioned ones. Other techniques are no longer described in detail. These techniques can be applied to fuse network data. The performance comparison of different fusion techniques is given in Section 5 based on the criteria proposed in Section 4.

4. Evaluation Criteria of DF Techniques

The application of DF techniques in intrusion detection has received particular attention in the field of network security. Many studies on DF have been conducted to improve the performance of NIDS. However, DF in NIDS still faces many serious challenges, such as how to reduce the complexity of massive data, how to ensure data security, and how to overcome the complexity and improve the efficiency of the fusion. Therefore, in order to facilitate the analysis and comparison of different fusion techniques, we propose a number of criteria for evaluating the performance of fusion techniques in NIDS based on the traditional criteria of IDS performance. Herein, we introduce specific evaluation criteria. Since most of the experiments for NIDS performance testing are based on a few public datasets, we firstly introduce the commonly used datasets for intrusion detection.

4.1. Datasets. Since real-time network data brings personal or organizational privacy issues and cannot be used for comparison of different algorithms, most of researches conduct experiments based on open datasets. Fusion techniques may show different performance based on different datasets. Herein, we introduce some classic datasets and new but more realistic datasets that are used in the field of intrusion detection research.

4.1.1. DARPA Dataset. In order to evaluate difficult intrusion detection techniques, the United States MIT Lincoln Laboratory successfully constructed a complete dataset in 1998, namely, DARPA 1998. The dataset is a 9-week network connection data collected from a simulated US Air Force LAN, dividing into training data and testing data. The testing data contains some types of attacks that do not appear in the training data, which makes the dataset more realistic. The KDD99 dataset was generated for the KDD cup competition, which extracts 41 features from the DARPA 1998 dataset. It is one of the most popular and comprehensive intrusion detection datasets and is widely applied to evaluate the performance of NIDSs [54]. It includes a complete training set, 10% training set, and a testing set. Each connection record in the KDD99 training dataset contains 41 feature attributes and an attack type label. The type of attack in KDD99 training dataset mainly includes Denial-of-Service (DOS) attacks, Probe attacks, User-to-Root (U2R) attacks, and Remote-to-Local (R2L) attacks. The KDD99_10% packet is a 10% sample of KDD99 packets, with approximately 490,000 data records, which is used in most of the literatures. However, there are many problems in KDD99; for example, the number of different types of attacks is not balanced and some data records are duplicate or invalid. To address these problems

in the KDD99 dataset, as a new revision of the KDD99, NSL-KDD was proposed by Tavallaee et al. [55]. The training and testing datasets of the NSL-KDD consist of approximately 125,973 and 22,544 connection records, respectively. Similar to the KDD99 dataset, each record in this dataset has 41 quantitative and qualitative features.

4.1.2. Kyoto 2006+ Dataset. There is a fatal problem in the existing dataset benchmark (KDD99) for network security, which does not reflect the current network security situation and the latest attack characteristics. This is because it generated from a simulated network nearly 20 years ago. To overcome its limitations, the Kyoto 2006+ dataset was presented by Song et al. [56]. It is a dataset based on actual traffic data from 2006 to 2009, which comes from different types of honeypots installed in the Kyoto University. The dataset consists of 14 conventional features captured by honeypots based on the KDD99 dataset and 10 additional features. Conventional features include the duration of the session, service, source byte, and destination byte, which is meaningful and important for subsequent data processing or decision-making. In addition to 14 statistical features, additional features were extracted, which may enable us to investigate effectively what kinds of attacks happened in networks. It can be used for further analysis and evaluation of NIDSs. The Kyoto 2006+ dataset includes about 50,033,015 normal sessions and 434,343,255 attacks, in which 425,719 attacks are unknown. Each connection in the dataset has 23 features. Compared to the KDD99 dataset, the Kyoto 2006+ dataset is generated in the real network. By using the Kyoto 2006+ dataset, researchers can access more realistic and practical network security attacks.

4.1.3. UNSW-NB15 Dataset. The above-mentioned datasets cannot meet the needs of research on the current network security situation, especially KDD99 and NSL-KDD. The UNSW-NB15 [57] was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a dataset that consists of real modern normal activities and synthetic contemporary attacks. The data collection period was 16 hours on January 22, 2015, and 15 hours on February 17, 2015. Tcpdump tool is used to capture 100 GB of the raw traffic. This dataset contains nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Moreover, the Argus and Bro-IDS tools are used and twelve algorithms are developed to generate in total 49 features with class labels. There are 175,341 records in the training set and 82,332 records in the testing set. The key characteristics of the UNSW-NB15 dataset are a hybrid of the real modern normal behaviors and the synthetic attack activities. Thus, this dataset is considered as a new benchmark dataset that can be used for evaluating NIDSs by the NIDS research community [57]. It is worth noting that the IXIA tool contains all the information about the new attacks that are continuously updated from CVE site 4. This site is a public information security vulnerability and exposure dictionary. However, it is undeniable that the UNSW-NB15 dataset is more complex than the KDD99 dataset [58].

TABLE 1: The formulas of the metrics.

Measures	Equations
ACC	$(TP + TN)/(TP + FP + TN + FN)$
PR	$TP/(TP + FP)$
RR	$TP/(TP + FN)$
F-Measure	$(2 * PR * RR)/(PR + RR)$
FPR	$FP/(TN + FP)$
FNR	$FN/(FN + TP)$
FAR	$(FPR + FNR)/2$

4.2. Validity. The validity is the key to measuring the quality of the NIDS. The purpose of the application of fusion technology is to improve the performance of intrusion detection. Therefore, the validity can still be used to measure the fusion technology.

The elements of the validity evaluation metrics include TP (the number of positive samples predicted to be positive), FP (the number of negative samples predicted to be positive), FN (the number of positive samples predicted to be negative), and TN (the number of negative samples predicted to be negative). Based on these measurement elements, the accuracy (ACC), precision rate (PR), recall rate (RR), F-Measure, FPR, and FNR are applied to evaluate the performance of the fusion techniques. These metrics' formulas are listed in Table 1.

4.3. Efficiency. In the big data era, communications and activities between people generate high volume and high-dimensional network data that require real-time classification. In NIDS, not only the network behavior classification technology needs to be efficient, but also the efficiency of data fusion is crucial [59], which determines the efficiency of NIDS. Training time and testing time can be used to measure the efficiency of fusion technology. Besides, the number of features produced by feature fusion also measures the efficiency of the fusion technique.

4.4. Data Security. In actual network monitoring, DF and classification techniques concern data security issues in order to provide trustworthy data fusion results, such as data confidentiality, integrity, and creditability. We must consider that the privacy of individuals or organizations cannot be compromised when we analyze and fuse network data. Therefore, data security and data privacy also need to be considered in data fusion.

4.5. Scalability. Digital communications will enter the era of 5G with the rapid technology development. Large-scale heterogeneous networks have become the trend of network development, and mass data and heterogeneous DF technologies are increasingly important. Fusion techniques and frameworks should take scalability into consideration, such as compatibility with different data formats and scalability of memory and CPU, which, therefore, becomes a measure of fusion technologies.

5. Comparisons and Discussions

Based on the above evaluation criteria, we conduct a rigorous review and analysis on 31 related studies, of which 23 are feature fusion techniques and the remaining 8 are decision fusion techniques. The results of research and analysis are listed in Tables 2 and 3, respectively. The experiments reported in the above work were conducted based on published datasets, including KDD99, NSL-KDD, Kyoto 2006+, and UNSW-NB15. We analyzed and compared the performance of different fusion techniques in terms of the feature fusion and the decision fusion based on the proposed criteria and specified metrics. It must be mentioned that the following comparisons are made based on different datasets. In addition, the experimental details in the literature are different, which may affect the performance evaluation of data fusion techniques.

5.1. Comparison of Feature Fusion Techniques. In this part, we review feature fusion techniques based on our proposed criteria and show our evaluation results in Table 2.

The original intention of feature fusion is to reduce the size of data and improve the operation efficiency of NIDS. Therefore, the efficiency is the key to measure the quality of feature fusion. We concern with training time compared with testing time in evaluating the efficiency of feature fusion. This is because the training time is usually far longer than the corresponding testing time. We first analyze and compare the training time of classifiers using different feature fusion techniques based on different datasets. For the KDD dataset series (DARPA99, KDD99, and KDD99.10%), we can find that the training time of network intrusion classifier using the following feature fusion techniques is shorter than others, such as GFR, FRM-SFM [18], and CART [23]; CFS-GA [25] is very efficient for the NSL-KDD dataset; based on the Kyoto 2006+ dataset, PLS [12] helps to reduce time consumption of classifier training. In summary, these mentioned fusion techniques are outstandingly efficient in the training time of network behavior classifier. What these fusion techniques have in common is that fewer features are generated regardless of the dataset, with a minimum of 4 features in [25]. The filter is more efficient than the wrapper among these feature fusion techniques, and the hybrid methods usually have excellent efficiency.

In addition to efficiency, the validity is also an important measure of feature fusion techniques. For the KDD dataset series, SA-SVM [20], GA-LR [16], (Filter-MISF, FMIFS) [17], PCA [11], MIFS [24], (FRM-SFM, GFR) [18], SVM [9, 10, 19], (GeFS-mRMR, GeFS-CFS) [19], and NN [9] achieved very high accuracy, exceeding 99.20%, and the highest was 99.96% of SA-SVM. In addition, the FPR of Filter-MISF, GA-LR, Filter, MIFS, MLCFS [24], and SVM are less than 0.50%. We found out that GA-LR, SVM, Filter-MISF, and MISF perform very well in terms of validity in the KDD dataset series. As for the NSL-KDD dataset, (FMISF, MIFS, FLCFS) [24], Chi-Square [21], FVBRM [27], and CFS [30] performed excellently in accuracy, both exceeding 96.75% and up to 99.91% of FMIFS. The FPR of FMISF, MIFS, and FLCFS are all lower than 0.53%, and Chi-Square's FAR is 0.13%. These

TABLE 2: The performance of different feature reduction algorithms.

(a)

Feature fusion techniques	Article	Dataset	Number of training/testing data	Number of features	Classifier	Identified attack types	Metrics					Validity	F-Score	FPR	FNR	FAR	Training time (s)	Efficiency Testing time (s)	Data security	Scalability
							ACC	PR	RR	Score		RR	Score							
NN	[8]	DARPA99	6819/3679	84/37	MLF	All	*	*	*	*	*	*	*	*	*	*	*	*	×	×
	[9]	KDD99	7000/7000	41/13	NN	Attack/Normal	99.41%	*	*	*	*	*	*	*	*	*	*	*	×	×
	[10]	KDD99.10%	*	41/34	NN	Attack/Normal	81.57%	*	*	*	*	*	*	18.19%	0.25%	9.22%	*	*	×	×
PCA	[11]	KDD99.10%	5000/5000	41/10	SVM	Probe	99.78%	99.85%	99.70%	99.77%	*	*	*	*	*	*	276	*	×	×
	[12]	KDD99.10%	5000/5000	41/10	SVM	R2L	99.70%	99.50%	99.39%	99.53%	*	*	*	*	*	*	237	*	×	×
	[12]	Kyoto 20606+	31360/47040	18/5	MLP	Attack/Normal	97.12%	*	*	*	*	*	*	4.29%	1.44%	2.87%	22.14	*	×	×
	[13]	NSL_KDD	125971/22000	41/23	NN	All	*	86.49%	83.95%	83.78%	*	*	*	*	*	*	*	*	×	×
Fisher-Score	[14]	KDD99.10%	1500/1500	41/13	RBF-NN	Attack/Normal	*	85.33%	*	*	*	*	*	5.40%	*	*	6.71	0.13	×	×
				41/19			*	91.27%	*	*	*	*	*	5.20%	*	*	8.38	0.13	×	×
				41/41			*	95.70%	*	*	*	*	*	6.31%	*	*	9.07	0.18	×	×
RF	[15]	KDD99.10%	16919/49838	41/34	RF	Attack/Normal	*	*	94.20%	*	*	*	*	1.10%	*	*	*	*	×	×
	[16]	KDD99.10%	1000/1000	41/18	RF	Attack/Normal	99.90%	*	99.81%	*	*	*	*	0.11%	*	*	*	*	×	×
		UNSW-NB15	2000/2000	49/20	C4.5	Attack/Normal	81.42%	*	*	*	*	*	*	6.39%	*	*	*	*	×	×
Filter- MISF Filter	[17]	KDD99.10%	15246/478775	41/6	LS-SVM	Attack/Normal	99.90%	*	99.93%	99.53%	0.07%	*	*	*	*	*	63.19	2732	×	×
				41/19			99.75%	*	99.43%	99.34%	0.17%	*	*	*	*	*	87.83	30.64	×	×
				41/25			99.70%	*	99.38%	99.34%	0.23%	*	*	*	*	*	*	*	×	×
GFR	[18]	KDD99.10%	550/115705	41/19	Multi-class SVM	All	98.62%	*	*	*	*	*	*	*	*	*	0.12	4.63	×	×
				41/10			98.68%	*	*	*	*	*	*	*	*	*	0.16	7.8	×	×
SVM	[10]	KDD99.10%	*	41/30	Multi-class SVM	All	99.61%	*	*	*	*	*	*	*	*	*	81.18	6.36	×	×
	[19]	KDD99.10%	424/106	41/17	SVM	Dos and Probe	99.30%	*	*	*	*	*	*	*	*	*	*	*	×	×
	[9]	KDD99	7000/7000	41/13	SVM	Attack/Normal	99.52%	*	*	*	0.50%	*	*	*	*	*	163	1.06	×	×
SA-SVM	[20]	KDD99	93969/10441	41/23	SA-DT	All	99.96%	*	*	*	*	*	*	*	*	*	*	*	×	×
Chi-Square	[21]	NSL_KDD	8325/24975	41/31	Multi-class SVM	All	98.00%	*	*	*	*	*	*	*	*	0.13%	*	*	*	*

LR: logistic regression; MISF: Mutual Information-Based Feature Selection; GFR: gradually feature removal method; FRM: feature removal method; SFM: sole feature method; SA: simulated annealing; MLF: multilayer feed-forward; LS: least square; and DT: decision tree. *Not given. × Not mentioned. Number of features (m/n): m and n represent the number of features before and after fusion, respectively.

(b)

Feature fusion techniques	Article	Dataset	Number of training/testing data	Number of features	Classifier	Identify attack types	Metrics					Validity			Efficiency			Data security	Scalability
							ACC	PR	RR	F-Score	FPR	FNR	FAR	Training time (s)	Testing time (s)				
GeFS-CFS				41/4.5			99.20%	*	*	*	*	*	*	*	*	*	×	×	
GeFS-mRMR	[19]	KDD99_10%	424/106	41/18	C4.5	Dos and Probe	99.60%	*	*	*	*	*	*	*	*	*	×	×	
Markov-Blanket				41/17	BN		98.70%	*	*	*	*	*	*	*	*	*	×	×	
BN	[22]	KDD99	4700/4700	41/30	BN	All	83.13%	*	*	*	*	*	*	37.44	*	×	×		
	[23]	KDD99	5092/6890	41/17	BN	All	91.06%	*	*	*	*	*	*	112.11	59.4	×	×		
CART	[23]	KDD99	5092/6890	41/12	CART	All	88.52%	*	*	*	*	*	*	3.86	0.19	×	×		
	[19]	KDD99_10%	424/106	41/12	CART	Dos and Probe	94.30%	*	*	*	*	*	*	*	*	×	×		
FMIFS		KDD99	*				99.79%	*	99.46%	*	0.13%	*	*	*	*	×	×		
	[24]	NSL_KDD	*	41/19	LS-SVM	Attack/Normal	99.91%	*	98.76%	*	0.28%	*	*	*	*	×	×		
		Kyoto 2006+	*				99.77%	*	99.64%	*	0.13%	*	*	*	*	×	×		
MIFS		KDD99	*				99.70%	*	99.38%	*	0.23%	*	*	*	*	×	×		
	[24]	NSL_KDD	*	41/25	LS-SVM	Attack/Normal	97.96%	*	95.96%	*	0.53%	*	*	*	*	×	×		
		Kyoto 2006+	*				99.32%	*	98.59%	*	0.16%	*	*	*	*	×	×		
FLCFS		KDD99	*				97.63%	*	89.26%	*	0.34%	*	*	*	*	×	×		
	[24]	NSL_KDD	*	41/17	LS-SVM	Attack/Normal	96.75%	*	93.26%	*	0.47%	*	*	*	*	×	×		
		Kyoto 2006+	*				99.12%	*	98.10%	*	0.58%	*	*	*	*	×	×		
CFS-GA	[25]	NSL_KDD	125973/22544	41/4	J48	Attack/Normal	91.86%	*	*	*	*	*	*	1.37	0.22	×	×		
BBAL-NB		NSL_KDD	9566/4500	41/15	NB	Attack/Normal	*	*	91.62%	*	5.73%	*	*	*	0.76	×	×		
BBAL-SVM	[26]	NSL_KDD	9566/4500	41/16	SVM	Attack/Normal	*	*	95.87%	*	2.89%	*	*	*	68.77	×	×		
FVBRM	[27]	NSL_KDD	56687/6299	41/24	NB	All	97.78%	*	*	*	*	*	*	*	9.42	×	×		
		KDD99	90000/10000	41/16		All	90.36%	*	*	*	*	*	*	*	*	×	×		
ML	[28]	NSL_KDD	90000/10000	41/16	SVM	Attack/Normal	89.35%	*	*	*	*	*	*	*	*	×	×		
		Kyoto 2006+	90000/10000	23/8		Attack/Normal	87.12%	*	*	*	*	*	*	*	*	×	×		
ARM	[29]	KDD99	*	41/11	NB	All	62.02%	*	*	*	*	*	*	*	*	×	×		
		UNSW-NB15	*	49/11			37.50%	*	*	*	*	*	*	*	*	×	×		
HVS				18/5	MLP	Attack/Normal	98.28%	*	*	*	3.05%	0.35%	1.70%	64.47	0.02	×	×		
PLS	[12]	Kyoto 2006+	*	18/5	PLS	Attack/Normal	94.72%	*	*	*	6.52%	4.02%	5.27%	0.02	0.03	×	×		
CFS	[30]	NSL_KDD	25192/11850	41/8	RandomTree	All	97.49%	*	*	*	*	*	*	2.50%	*	×	×		

GefS: generic feature selection; mRMR: minimal redundancy maximal relevance; BN: Bayesian networks; CART: classification and regression tree; FMIFS: flexible mutual information feature selection; MIFS: mutual information feature selection; FLCFS: flexible mutual information feature selection; BBAL: binary bat algorithm with levy flights; FVBRM: feature vitality based reduction method; ML: feature vitality based reduction method; ARM: association rule mining; HVS: heuristic for variable selection; and PLS: partial least squares regression. * Not given. * m/n : m and n represent the number of features before and after fusion, respectively.

TABLE 3: The performance of different decision reduction algorithms.

Decision fusion techniques	Article	Dataset	Number of training/testing data	Classifier	Identified attack types	Metrics							
						ACC	PR	RR	F-Score	FPR	FNR	Data security	Scalability
D-S Evidence Theory	[31]	KDD99	*	Multiclass SVM	Attack/normal	*	*	95.10%	*	0.19%	4.74%	×	×
	[32]	KDD99	*	RBF-NN	Dos	99.08%	*	*	*	0.71%	*	×	×
				C4.5		98.90%	*	*	*	*	*	×	×
				BN		96.70%	*	*	*	*	*	×	×
	[33]	KDD99	30000/30000	NN	Attack/normal	99.20%	*	*	*	*	*	×	×
				MDT		*	*	*	*	*	*	×	×
RF	[15]	KDD99_10%	16919/49838	D-S fusion		99.10%	*	*	*	*	*	×	×
				RF	Attack/normal	*	*	94.20%	*	1.10%	*	×	×
Adaboost	[34]	KDD99	494021/311029	Decision stumps	Attack/normal	*	*	90.02%	*	1.68%	*	×	×
NN				PHAD		99%	35%	28.00%	31%	*	*	×	×
				ALAD		99%	38%	32.00%	35%	*	*	×	×
	[35]	DARPA99	*	Snort	All	99%	9%	51.00%	15%	*	*	×	×
				Data-dependent fusion		99%	39%	68.00%	50%	*	*	×	×
RBF-NN	[32]	KDD99	*	RBF-NN	Dos	99.59%	*	*	*	0.63%	*	×	×
Majority voting rule				BN		*	93.10%	91.90%	92.20%	*	*	×	×
				IBK		*	99.60%	99.60%	99.60%	*	*	×	×
	[36]	NSL-KDD	8105/11695	J48	Attack/normal	*	98.50%	98.50%	98.50%	*	*	×	×
				SVM		*	98.50%	92.90%	92.60%	*	*	×	×
				Classifier fusion		*	99.10%	99.40%	99.20%	*	*	×	×
MLP				MLP-4		*	*	*	*	*	*	×	×
				intrinsic features		*	*	*	*	3.19%	*	×	×
				MLP-7		*	*	*	*	*	*	×	×
	[37]	KDD99	833/7436	content features	Attack/normal	*	*	*	*	2.25%	*	×	×
				MLP-19 traffic features		*	*	*	*	23.94%	*	×	×
				MLP-30 features		*	*	*	*	3.57%	*	×	×

PHAD: packet header anomaly detection system; ALAD: application layer anomaly detector; MDT: Multirandom Decision Tree; and IBK: lazy classifier. * Not given. [×]Not mentioned. Number of features (m/n): m and n represent the number of features before and after fusion, respectively.

feature fusion techniques have outstanding characteristics in NIDSs based on NSL-KDD datasets. In the Kyoto 2006+ dataset, the accuracy of (FMIFS, MIFS, FLCFS) [24] and (HVS, PCA) [12] was all higher than 97.12%, and the FPR of FMIFS, MIFS, and FLCFS are all below 0.58%.

A notable fact is that the accuracy of the classification in the new dataset (UNSW-NB15) is not as good as the old datasets mentioned earlier (such as KDD dataset series). The major reason is that the UNSW-NB15 dataset is considered complex due to the similar behaviors of the modern attack and normal network traffic compared to the KDD99 dataset [55]. So far, the effectiveness of network intrusion detection is not good based on the UNSW-NB15 dataset. The accuracy in [16] reached the highest accuracy 81.42% based on our statistics, and the corresponding feature fusion technique and classifier are GA-LR, C4.5, respectively. Decision Tree (DT) classifier has indeed performed better in the UNSW-NB15 dataset [55] than other methods. The misfortune is not alone. The FAR of NIDSs in the UNSW-NB15 dataset is also bad. Therefore, advanced classification techniques and feature fusion techniques need further study. In general, GA-LR, SVM, Filter-MISF, and MISF show excellent validity in the KDD dataset series; FMISF, MIFS, FLCFS, and Chi-Square are more valid in the NSL-KDD dataset; the feature fusion techniques with high-validity are FMIFS, MIFS, and FLCFS in the Kyoto 2006+ dataset. Because the performance of network intrusion detection based on UNSW-NB15 dataset is not very good, more advanced fusion and classification techniques should be further investigated in order to identify the anomalies from this complex dataset.

Unfortunately, the fusion techniques in the literature we have reviewed have not considered the security of data fusion. The data privacy issues were not covered because existing experiments were based on the public datasets. In addition, the scalability of fusion technologies and frameworks were normally not mentioned in the past work. However, these properties of data fusion are particularly important in the big data era. More efforts are needed in order to solve these issues.

5.2. Comparison of Decision Fusion Techniques. In this subsection, we analyze the performance of different decision fusion techniques based on the proposed criteria and show our evaluation results in Table 3.

According to Table 3, we can find that the training and testing time of the classifiers are not recorded. The reason is that decision fusion techniques fuse the recognition results of basic classifiers. Although the training and testing time of classifiers can reflect the efficiency of classifiers, it cannot reflect the merits of decision fusion techniques. Besides, the KDD dataset series are used in the most statistical literature. So herein, we mainly analyze the validity of decision fusion techniques based on the KDD dataset series. The accuracy of D-S Evidence Theory [32, 33] and NN [33] is over 99%, which is usually higher than the accuracy of a single basic classifier. The FPR is also reduced through the integration of basic classifiers. The FPR in [31] (D-S Evidence Theory) is as low as 0.19%. As a group, D-S Evidence Theory, Data-Dependent Fusion, NN, RF, and Adaboost show good fusion performance in combining multiple basic decisions.

Like the feature fusion techniques, the existing decision fusion techniques did not consider the credibility of basic decisions and data security in the process of integration, which will affect the reliability of the final results or cause privacy leakage. Besides, most of the literatures also fail to analyze the scalability of decision fusion. We believe that these aspects are very important and should attract special attention.

6. Open Issues and Future Research Directions

In recent years, DF has achieved special attention and has developed rapidly in many fields. In the field of network intrusion detection, scholars have conducted extensive researches in DF and have made significant progress. However, the current data fusion techniques still face some serious challenges or open issues, which are summarized as below according to our literature review.

First, most of the existing researches were conducted based on open datasets and the practicability of these fusion algorithms or techniques needs further validation. Few researches used real network data because it is easy to expose privacy and cannot measure or compare with other existing works, which is not conducive to the development of data fusion technology. In fact, this is a difficult contradiction, which hinders the further development of network intrusion detection.

Second, in the era of big data, the network security monitoring and prevention may need real-time fusion and processing of massive network data. However, large data communication overhead and long computation delay are obviously a big challenge to overcome.

Third, existing DF technologies do not consider data security, including confidentiality and credibility. The feature fusion techniques could reveal the privacy of individuals or organizations, and the decision fusion techniques need to identify the credibility of local decisions. All above are not considered in the past work.

Fourth, since most of the researches conducted their work over some public datasets and these datasets are preprocessed, there are few data level fusion techniques used in intrusion detection. However, we are facing a large number of different types of raw data in actual networks. Thus, the data layer fusion becomes indispensable for intrusion detection. Special efforts are expected on data fusion with regard to network intrusion detection.

Fifth, there is a lack of studies on the visualization of data fusion. Through utilizing the visualization algorithm, we can not only deeply understand the features and effectiveness of the fusion technology, but also easily identify the distribution characteristics of the fused data. Few articles use a visual method to analyze classical datasets. In [60], Ruan et al. performed a visual analysis of the KDD99 dataset using MDC and PCA techniques to clearly identify normal and attack clusters. Based on this research, we believe that it is also necessary to provide a beautiful and comprehensive data fusion expression.

In addition, based on the above open issues, we further proposed a number of promising research directions in the field of data fusion for network intrusion detection.

First, the improvement of data fusion technology depends on new datasets to evaluate and verify. Most of the fusion techniques and intrusion detection technology show excellent performance on some old datasets, such as KDD99 and NSL-KDD. However, these datasets are out of date and do not represent the current network security situation, which deviates from the actual network security detection. More research needs to be done on new dataset collection, such as UNSW-NB15. The existing problem is that the performance of feature fusion based on the UNSW-NB15 dataset is not good. We should further study more advanced or appropriate fusion techniques to better identify abnormalities from complex network data.

Second, big network data fusion techniques should be investigated. The current fusion techniques are difficult to effectively and adaptively integrate network data of high-velocity, varieties of formats and types. In the era of big data, in addition to the large amount of data, the network data that needs to be collected come from different sources in different types of networks. Therefore, the collection of heterogeneous network data is required to research more advanced fusion methods.

Third, universal, flexible, and extensible fusion framework should be studied. There are many kinds of data fusion technologies, and the principles and mathematical theories of some fusion technologies are not easy to understand. Therefore, the simple, easy-to-use, universal, and easy-to-expand network data fusion architecture is worth studying. It can modularize mature fusion techniques and provide open interfaces for new fusion methods and architectures; thus, it greatly promotes the development of data fusion in the field of intrusion detection.

Fourth, data security in data fusion should be ensured. Most of the existing researches are based on public datasets, and security issues were not considered at all. In an actual network, network data includes personal or organizational information, which is easily revealed during the integration process, and the credibility and integrity of network data are difficult to guarantee. Data security and privacy should be protected and ensured in order to achieve trustworthy data fusion.

Finally, data layer fusion is an essential part of study towards efficient and practical data fusion in real-time network intrusion detection. The data layer fusion has not been seriously studied by relevant literature because of the widespread use of public datasets. The study of data layer fusion is also very significant, especially for practical applications. However, it is very difficult to collect and evaluate the original network data containing various modern attacks.

7. Conclusion

In this article, we categorically presented a detailed review on the feature fusion techniques and the decision fusion techniques used in NIDSs. A specific description of DF in

the field of intrusion detection was presented in order to motivate this work. Based on the literature study, we proposed the evaluation criteria of data fusion techniques in terms of NIDS. The performance of different data fusion techniques is measured using the proposed criteria. We found that, in the feature fusion, in addition to some excellent fusion techniques, such as SVM and MIFS, the improved types of fusion techniques and hybrid fusion techniques are generally efficient and valid. For the decision fusion techniques, D-S Evidence Theory, NN, RF, and Adaboost can combine multiple decisions more precisely than other methods regarding the studies based on KDD dataset series. In addition, we found many effective classification algorithms in NIDS, namely, RF, C4.5, NN, and SVM, as well as their variants. Unfortunately, the current fusion techniques normally did not consider the security and the scalability of DF.

DF has been regarded as one of the most important technologies in improving the performance of the NIDSs. The use of DF can well alleviate the defects of network intrusion detection and improve the comprehensive performance of NIDSs. However, there are still many deficiencies in current DF techniques. Based on our review, we pointed out the main challenges and promising future research directions in this field of research. In summary, this article provides a good reference for researchers and practitioners in the field of network intrusion detection.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is sponsored by the National Key Research and Development Program of China (Grant 2016YFB0800700), the NSFC (Grants 61602359, 61672410, and U1536202), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program no. 2016ZDJC-06), the Fundamental Research Funds for the Central Universities (JB181503), the 111 Project (Grants B08038 and B16037), and Academy of Finland (Grant no. 308087).

References

- [1] J. Tian, W. Zhao, R. Du, and Z. Zhang, "A New Data Fusion Model of Intrusion Detection-IDSFP," in *Parallel and Distributed Processing and Applications*, vol. 3758 of *Lecture Notes in Computer Science*, pp. 371–382, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [2] F. E. White, "Data Fusion Lexicon," Defense Technical Information Center, 1991.
- [3] H. Boström, S. F. Andler, M. Brohede et al., "On the definition of information fusion as a field of research," *Neoplasia*, vol. 13, pp. 98–107, 2007, IN1.
- [4] B. R. Raghunath and S. N. Mahadeo, "Network Intrusion Detection System (NIDS)," in *Proceedings of the 2008 First International Conference on Emerging Trends in Engineering and Technology*, pp. 1272–1277, Nagpur, Maharashtra, India, July 2008.

- [5] Y. Fu, Z. Yan, J. Cao, O. Koné, and X. Cao, "An Automata Based Intrusion Detection Method for Internet of Things," *Mobile Information Systems*, vol. 2017, Article ID 1750637, 13 pages, 2017.
- [6] L. Wang and H. Xiao, "An integrated decision system for intrusion detection," in *Proceedings of the 1st International Conference on Multimedia Information Networking and Security, MINES 2009*, pp. 417–421, chn, November 2009.
- [7] M. A. Aydin, A. H. Zaim, and K. G. Ceylan, "A hybrid intrusion detection system design for computer network security," *Computers and Electrical Engineering*, vol. 35, no. 3, pp. 517–526, 2009.
- [8] H. Wang, X. Liu, J. Lai, and Y. Liang, "Network security situation awareness based on heterogeneous multi-sensor data fusion and neural network," in *Proceedings of the International Multi-Symposiums on Computer and Computational Sciences*, pp. 352–359, 2007.
- [9] S. Mukkamala, G. Janoski, and A. Sung, "Audit data reduction for intrusion detection," Training, 2008.
- [10] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *Proceedings of the International Symposium on Applications and the Internet*, pp. 209–216, IEEE, Orlando, Fla, USA, January 2003.
- [11] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of PCA and optimized SVM," in *Proceedings of the 2014 International Conference on Contemporary Computing and Informatics, IC3I 2014*, pp. 879–884, ind, November 2014.
- [12] A. Ammar, "Comparison of Feature Reduction Techniques for the Binominal Classification of Network Traffic," *Journal of Data Analysis Information Processing*, vol. 03, pp. 11–19, 2015.
- [13] N. A. Biswas, F. M. Shah, W. M. Tammi, and S. Chakraborty, "FP-ANK: An improvised intrusion detection system with hybridization of neural network and K-means clustering over feature selection by PCA," in *Proceedings of the 18th International Conference on Computer and Information Technology, ICCIT 2015*, pp. 317–322, bgd, December 2015.
- [14] J. Zhou, J. Wang, and Z. Qun, *The Research on Fisher-RBF Data Fusion Model of Network Security Detection*, Springer, Berlin, Heidelberg, Germany, 2012.
- [15] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 5, pp. 649–659, 2008.
- [16] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Computers & Security*, vol. 70, pp. 255–277, 2017.
- [17] M. A. Ambusaidi, X. He, Z. Tan, P. Nanda, L. F. Lu, and U. T. Nagar, "A novel feature selection approach for intrusion detection data classification," in *Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 82–89, 2015.
- [18] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert Systems with Applications*, vol. 39, no. 1, pp. 424–430, 2012.
- [19] H. T. Nguyen, S. Petrović, and K. Franke, "A comparison of feature-selection methods for intrusion detection," in *Lecture Notes in Computer Science*, I. Kottenko and V. Skormin, Eds., vol. 6258, pp. 242–255, 2010.
- [20] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," *Applied Soft Computing*, vol. 12, no. 10, pp. 3285–3290, 2012.
- [21] I. Sumaiya Thaseen and C. Aswani Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 4, pp. 462–472, 2017.
- [22] Y. Xu and W.-B. Zhang, "A novel IDS model based on a Bayesian fusion approach," in *Proceedings of the 1st International Conference on Multimedia Information Networking and Security, MINES 2009*, pp. 546–549, chn, November 2009.
- [23] S. Chebroli, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & Security*, vol. 24, no. 4, pp. 295–307, 2005.
- [24] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.
- [25] K. S. Desale and R. Ade, "Genetic algorithm based feature selection approach for effective intrusion detection system," in *Proceedings of the International Conference on Computer Communication and Informatics*, pp. 1–6, 2015.
- [26] A.-C. Enache, V. Sgarciu, and A. Petrescu-Niță, "Intelligent feature selection method rooted in Binary Bat Algorithm for intrusion detection," in *Proceedings of the Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI 2015*, pp. 517–521, 2015.
- [27] S. Mukherjee and N. Sharma, "Intrusion Detection using Naive Bayes Classifier with Feature Reduction," *Procedia Technology*, vol. 4, pp. 119–128, 2012.
- [28] M. A. Ambusaidi, X. He, and P. Nanda, "Unsupervised Feature Selection Method for Intrusion Detection System," in *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA*, pp. 295–301, 2015.
- [29] N. Moustafa and J. Slay, "The significant features of the UNSW-NB15 and the KDD99 data sets for Network Intrusion Detection Systems," in *Proceedings of the International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, 2017.
- [30] S. Thaseen and C. A. Kumar, "An analysis of supervised tree based classifiers for intrusion detection system," in *Proceedings of the International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pp. 294–299, 2013.
- [31] F. Xie, H. Yang, Y. Peng, and H. Gao, "Data fusion detection model based on SVM and evidence theory," in *Proceedings of the 2012 IEEE 14th International Conference on Communication Technology, ICCT 2012*, pp. 814–818, chn, November 2012.
- [32] A. P. Chan, D. S. Yeung, E. C. Tsang, and W. W. Ng, "Empirical study on fusion methods using ensemble of RBFNN for network intrusion detection," in *Lecture Notes in Artificial Intelligence*, S. Yeung, Z. Q. Liu, X. Z. Wang, and H. Yan, Eds., vol. 3930, pp. 682–690, 2006.
- [33] X. Zhao, H. Jiang, and L. Jiao, "A Data Fusion Based Intrusion Detection Model," in *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science*, pp. 1017–1021, Wuhan, Hubei, China, March 2009.
- [34] W. Hu and S. Maybank, "Adaboost-Based Algorithm for Network Intrusion Detection," *IEEE Transactions on Systems Man Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man Cybernetics Society*, vol. 38, pp. 577–83, 2008.
- [35] C. Thomas and N. Balakrishnan, "Advanced sensor fusion technique for enhanced intrusion detection," in *Proceedings of*

- the *IEEE International Conference on Intelligence and Security Informatics (ISI '08)*, pp. 173–178, Taipei, Taiwan, June 2008.
- [36] K. Saleem Malik Raja and K. Jeya Kumar, “Diversified intrusion detection using Various Detection methodologies with sensor fusion,” in *Proceedings of the 2014 International Conference On Computation of Power , Energy, Information and Communication (ICCPEIC)*, pp. 442–448, Chennai, India, April 2014.
 - [37] G. Giacinto, F. Roli, and L. Didaci, “Fusion of multiple classifiers for intrusion detection in computer networks,” *Pattern Recognition Letters*, vol. 24, no. 12, pp. 1795–1803, 2003.
 - [38] J. Song, H. Takakura, and Y. Kwon, “A generalized feature extraction scheme to detect 0-day attacks via IDS alerts,” in *Proceedings of the 2008 International Symposium on Applications and the Internet (SAINT'08)*, pp. 55–61, fin, August 2008.
 - [39] M. Beheshti, J. Han, K. Kowalski, J. Ortiz, J. Tomelden, and D. Alvillar, “Packet information collection and transformation for network intrusion detection and prevention,” in *Proceedings of the 2008 International Symposium on Telecommunications (IST'08)*, pp. 42–48, irn, August 2008.
 - [40] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, “An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks,” *Expert Systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005.
 - [41] E. Waltz and J. Llinas, *Handbook of Multisensor Data Fusion*, CRC Press, 2001.
 - [42] B. A. Fessi, S. Benabdallah, Y. Djemaiel, and N. Boudriga, “A clustering data fusion method for intrusion detection system,” in *Proceedings of the 11th IEEE International Conference on Computer and Information Technology, CIT 2011 and 11th IEEE International Conference on Scalable Computing and Communications, SCALCOM 2011*, pp. 539–545, cyp, September 2011.
 - [43] L. Cao, J. Tian, and W. Jiang, “Information fusion technology and its application to fire automatic control system of intelligent building,” in *Proceedings of the International Conference on Information Acquisition (ICIA'07)*, pp. 445–450, Seogwipo-si, South Korea, July 2007.
 - [44] L. Zhang, H. Leung, and K. C. C. Chan, “Information fusion based smart home control system and its application,” *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1157–1165, 2008.
 - [45] Y. Xiao and Z. Shi, “Application of multi-sensor data fusion technology in target recognition,” in *Proceedings of the 3rd IEEE International Conference on Advanced Computer Control (ICACC'11)*, pp. 441–444, chn, January 2011.
 - [46] X. Hu and X. Wang, “Application of fuzzy data fusion in multi-sensor fire monitoring,” in *Proceedings of the 2012 International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA '12)*, vol. 1, pp. 157–159, August 2012.
 - [47] Y. Chen, C. Li, P. Ghamisi, X. Jia, and Y. Gu, “Deep fusion of remote sensing data for accurate classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 8, pp. 1253–1257, 2017.
 - [48] Z. Yan, X. Jing, and W. Pedrycz, “Fusing and mining opinions for reputation generation,” *Information Fusion*, vol. 36, pp. 172–184, 2017.
 - [49] J. Liu, Z. Yan, and L. T. Yang, “Fusion - An aide to data mining in Internet of Things,” *Information Fusion*, vol. 23, pp. 1–2, 2015.
 - [50] G. H. John, R. Kohavi, and K. Pfleger, “Irrelevant Features and the Subset Selection Problem,” in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, pp. 121–129, 1994.
 - [51] E. De La Hoz, E. De La Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez, “Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps,” *Knowledge-Based Systems*, vol. 71, pp. 322–338, 2014.
 - [52] H. Chen, Y. Fu, and Z. Yan, “Survey on big data analysis algorithms for network security measurement,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 10394, pp. 128–142, 2017.
 - [53] D. Paudel, *A hybrid network intrusion detection system using SVM and GA*, 2016.
 - [54] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, “Intrusion detection by machine learning: a review,” *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994–12000, 2009.
 - [55] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *Proceedings of the International Conference on Computational Intelligence for Security and Defense Applications*, pp. 33–58, 2009.
 - [56] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, “Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation,” in *Proceedings of the Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pp. 29–36, 2011.
 - [57] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in *Proceedings of the Military Communications and Information Systems Conference*, pp. 1–6, 2015.
 - [58] N. Moustafa and J. Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set,” *Information Security Journal*, vol. 25, no. 1–3, pp. 18–31, 2016.
 - [59] D. Wu, B. Yang, and R. Wang, “Scalable privacy-preserving big data aggregation mechanism,” *Digital Communications and Networks*, vol. 2, no. 3, pp. 122–129, 2016.
 - [60] Z. Ruan, Y. Miao, L. Pan, N. Patterson, and J. Zhang, “Visualization of big data security: a case study on the KDD99 cup data set,” *Digital Communications and Networks*, vol. 3, no. 4, pp. 250–259, 2017.

Research Article

Uncovering Tor: An Examination of the Network Structure

Bryan Monk, Julianna Mitchell , Richard Frank, and Garth Davies

International CyberCrime Research Center (ICCRC), School of Criminology, Simon Fraser University, Burnaby, BC, Canada

Correspondence should be addressed to Julianna Mitchell; jdm17@sfu.ca

Received 3 November 2017; Revised 21 March 2018; Accepted 2 April 2018; Published 9 May 2018

Academic Editor: Zheng Yan

Copyright © 2018 Bryan Monk et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The dark web is a concealed portion of the Internet that can only be accessed through specialized software. Although multiple dark web technologies exist, with a common trait of using encryption to enforce anonymity, the Tor network remains the most prominent dark web network. To visit websites on the network, the user must use a heavily modified Firefox browser. The use of encryption to achieve anonymity poses a significant challenge for law enforcement that wishes to monitor users and content for illicit activity. This study examines Tor by focusing on the network structures created between websites via hyperlinks. Examining hyperlinks can provide insight into how virtual communities form on a network. We explore traditional social disorganization principles as a basis to draw comparisons between these virtual communities and real-life crime-prone neighborhoods. Automated data collection techniques were used to leverage the interconnected nature of domains on Tor. Using social network analysis, website hyperlinks are examined and core sites are identified. The analysis shows that these core sites form a significant portion of all connections made on the network with a density of 0.132. This core serves a critical function and has implications for detecting how users connect on Tor.

1. Introduction

The Internet has been shown to be a very powerful communication tool, enabling individuals to connect globally to access and exchange material with very few obstacles, including governmental or jurisdictional interference. While the Internet has been a significant driving force in the globalization of knowledge, it has simultaneously created an environment where nefarious users can disseminate illicit content, connect users with similar interests, and facilitate the proliferation of virtual illicit communities. Today, the majority of global Internet users still only access a fraction of the Internet, the portion known as “the surface web” [1]. As opposed to the surface web, which is open and reachable by anyone with an Internet connection, the “dark web” uses the infrastructure of the surface web but, through encryption, creates a subnetwork that is anonymous, concealed, largely unindexed, and only accessible through encrypted Internet browsers [1, 2]. A true dark web has three central characteristics [2]: (i) it uses peer-to-peer technology rather than centralized servers that information can be tracked back to; (ii) it uses the infrastructure of the Internet; and (iii) it operates through nonstandard protocols and ports. As a

result of these characteristics, the structure of the dark web is constantly evolving. At the same time, these characteristics can foster a potentially risky environment within which illicit behaviors can more easily occur.

Dark webs are employed for a multitude of objectives, including keeping Internet activities and identities anonymous, evading censorship, and communicating sensitive information securely. Simultaneously, dark webs also increase opportunities and support for individuals conducting illicit activities [3]. Online connections can be created instantaneously and those interested in pursuing malicious activities or accessing illicit content can network with little effort [4]. This has led to growing concerns over the extent to which dark webs may be facilitating and fostering serious criminal activity, as well as assisting the operations of terrorists and violent extremist groups [5]. Law enforcement and researchers have realized the need to understand the extent of criminal activity on dark webs but have faced three related challenges. First, the extent of criminal activity remains unclear due to the sheer size and dynamic structure of dark web networks. Second, research is hindered by a lack of adequate tools and methods to examine the immensity and evolving structure of dark webs. Finally, criminological

theories have yet to be applied to the dark web as a means of understanding how crime can flourish in this environment.

This study responds to these challenges by examining dark web-based networks through the use of advanced data collection tools and analytic methods. While many dark web browsers exist, the Tor network remains one of the most well-known and frequently operated networks used to access and operate on the dark web [1] and, as such, is well suited to provide an in-depth understanding of network structures. Tor is a peer-to-peer network that operates by connecting users through specialized software designed to anonymize and encrypt data sent between those users. The Tor network allows users to access the dark web through a specially developed web browser. Since its inception in 2002, Tor has become one of the most universally used dark web technologies due to its anonymity features and its ability to efficiently access dark web content and information [6]. Although Tor developers maintain that the network's primary mission is to provide users with the technology to evade intrusive surveillance and data collection by governments and corporations while simultaneously fostering innovative research in online anonymity and privacy, there are increasing concerns about the illicit opportunities that could be provided to malicious users.

1.1. Content on Tor. A key research interest has been the nature of the content available on the Tor network. Content analyses have revealed that an extremely high volume of unethical content is available through the network [5], where unethical content was defined as negative behavior and included "anti-social behavior, drugs, weapons, hacking, cannibalism, bomb making, hit man services, black markets, and child pornography" [5]. Some researchers have concluded that the prominence and pervasiveness of this unethical content far outweigh the benefits of Tor services [5]. However, such findings have been based exclusively on content found on forum discussions from three main databases, potentially limiting their generalizability [5]. Past research has also demonstrated a varying degree of illegal activity on the network where virtual interaction and illegal file sharing were found to be the most prominent illegitimate uses of dark webs such as Tor [7]. File sharing includes sharing copyright-infringing files, such as music, movies, and TV [5]. Cybercriminals also have been known to utilize the network in order to exchange information and transfer data for hacking, identity fraud, and buying and selling illegal goods [8].

1.2. Malicious Uses of Tor. Tor and similar dark webs allow users to access websites that sell illegal goods and services, similar in functionality and structure to eBay, although carrying illicit material. Such sites, known as dark markets, carry a variety of illegal commodities and appear to cater to users looking for specific materials. Most of these markets require Bitcoin as virtual trading currency. Bitcoin is an increasingly popular peer-to-peer decentralized payment system [9]. Various types of malware have also been identified on the Tor network. Such malware presents a highly dangerous threat to individuals and uses ransomware to encrypt individual's files

and prohibit access to these files until a payment is provided [2]. It has been speculated that terrorist organizations operate on these markets to finance their activities through the selling of illegal weapons and drugs. This is possible for two important reasons: first, unlike the regular Internet, these domains are not registered to a central authority; and second, Tor is highly anonymized, which means users feel safe posting this content.

1.3. Tor Hyperlinks. Tor websites, like those found on the regular Internet, do not exist in a vacuum; rather, they are hyperlinked both with themselves and to each other. Without hyperlinks, websites could only be found if the exact URL was known to the user. Hyperlinks thus form the basis for how users traverse this network, connect to domains, explore content, and connect with other users. Social network analysis allows for the analysis of these hyperlinks and illuminates website connectivity. Websites with more incoming and outgoing hyperlinks may be considered to be more popular and more important to the dissemination of information within the Tor network. Network analysis can help to determine not only how information is being distributed and shared among network structures but also the importance of particular websites to the network. Like other Internet structures, there may be a small dense cluster of nodes that form the center of the network and guide the way information travels. This would have significant implications for how the structures of communities develop and function and could improve strategies used by law enforcement agencies to detect and remove illicit users and content.

In this study, we examine Tor's hyperlink connections through social network measures to gain insight into the network structures that form on Tor and how these structures create conditions favorable to deviant or criminal activity. An automated data collection tool, known as The Dark Crawler (TDC), was employed to collect and capture a sample of dark web content. TDC was able to navigate and collect hyperlink information for 1,220 unique dark web websites on the Tor network. This study also explores whether the core principles of social disorganization theory can explain the prevalence of "bad" communities on Tor. Traditional social disorganization theory hypothesizes that community structures characterized by instability, heterogeneity, and weak social ties can foster crime and disorder [10]. We suggest that Tor social network structures, formed through hyperlinks, have a similar nature in that sparse and unstable networks composed of weak connections could potentially foster illicit online activity and content. Although no known studies have yet to determine the time Tor domains are online and offline, the transitivity and instability of Tor have been widely speculated by researchers [11]. Additionally, like crime-prone neighborhoods, networks that form on Tor lack regulation or monitoring by law enforcement and government. Social disorganization theory proposes that the absence of a governing system can limit community members' ability to formally or informally control the behavior of others and increase opportunities for individuals to engage in criminal activities. If connections on Tor are found to be unstable and weak, coupled with a lack of

regulation, it is possible that these characteristics present unique opportunities for users to leverage the network to engage in devious activity similar to that of socially disorganized communities. There is currently a void in research that has specifically examined this link between dark web online communities and offline communities and, as such, the application of this criminological theory is only used as an exploratory lens in the current study. Previous work on Tor has examined individual websites such as The Silk Road, an online black market, [9, 12] or as a larger construct providing a content analysis of the domains found within the network [2, 5]. Using social network analysis, this study aims to examine (a) hyperlink connections and the structure of website communities they form on Tor and (b) how characteristics of these communities could have implications for criminal activity on Tor as understood through the lens of social disorganization theory. From a law enforcement perspective this may have implications for disruption strategies or target selection for policing interventions.

2. Theoretical Background

Past research has examined how Internet communities created through hyperlink connections can be understood as individuals connecting and developing ties, thereby forming small online communities within a large network [4]. These virtual communities are not constrained by geographic barriers and have provided an avenue for users to connect worldwide [13]. This has resulted in the development of large, globally connected social networks that allow users to connect with like-minded individuals, gain social support, and share information and material. Researchers have suggested that communities that form online have a similar nature to offline communities, where complex social networks are formed through similar interests or purposes and are sustained through continuous interactions among users [7, 14]. Further, researchers have proposed that, like the offline world, the Internet also contains “bad neighborhoods” with crime distributions resembling the offline world. These virtual neighborhoods have been considered to have greater concentrations of crime to the extent that they are largely composed of IP addresses hosting illicit content or performing malicious activities. Malicious users can also acquire criminal capital through formed connections by sharing information and material within online criminal communities [4]. As part of this study, we draw inspiration from a major criminological approach to explore whether this could be true of dark web network connections. While parallels can be drawn between regular Internet structures and dark web structures in how they connect websites, they are distinct enough to warrant a separate examination of Tor’s hyperlinking structures. That said, it is probable that Tor is used in the same manner by malicious users. As such, the primary assumptions of social disorganization theory are used as basis to understand how the structure of Tor itself, and how users connect on the network, could potentially increase the opportunities for deviant or criminal activity.

2.1. Social Disorganization Theory. Many criminologists have focused on the relationship between urban and community organization and crime [15–17]. Social disorganization theory specifically draws attention to the reciprocal connection between communities and crime and has become one of the most influential models of crime within criminology over time. Central to the social disorganization approach is the idea that community organization and social ties are important mechanisms through which communities can control crime. As such, the theory suggests urban organization can influence patterns of crime. In particular, the research of [10] proposed that four structural factors, instability, heterogeneity, weak social ties, and lack of supervision, can increase the likelihood of crime and delinquency in a community. It is argued that such factors disrupt the social organization in a community, where social organization is measured by the prevalence and interconnectedness of social networks. As such, they theorized that weak social structures decrease the ability of the community to maintain informal social control over members’ behavior. Informal control in a community has been known to occur when members can control crime through informal surveillance of the neighborhood and intervene in problematic or suspicious activity. Strong and cohesive social ties within community networks have, therefore, been shown to increase the effectiveness of social control in reducing crime because community members are more willing to engage in monitoring and guardianship behaviors against crime [16, 18]. Thus, the prevalence of community organization or community disorganization are regarded as separate traits that have an influence on crime rates.

Social disorganization theory has generally been used to analyze urban crime geographies where researchers have focused on demographic, economic, social, familial, and urban factors when assessing criminal activity. The three variables most often assessed are those related to poverty, ethnic heterogeneity, stability, and population mobility in a community, as these are viewed as factors that can weaken a community’s ability to manage the prevalence of crime [17]. In examining and testing these factors, researchers have found that such characteristics present in a community do appear to have a relationship with increased crime rates. With regard to the role of poverty, studies have found that communities of low socioeconomic status lack money and resource to organize and mobilize a community [19, 20]. In a study by [21] ethnic heterogeneity present among communities was found to be associated with social disorganization, where the degree of ethnic mix and population density to violent crime was analyzed. With regard to population mobility, meaning the rate of incoming and outgoing community members, [17] found that residential instability had a negative effect on network connections in a community, which in turn was related to an increase in crime. Additionally, [22] found that instability, measured by residents living in a community less than 2 years, had a reciprocal relationship with community disorder where the more instability that was present in a community led to increased disorder, and vice versa.

Network density has also been proposed as an influential characteristic by indicating the extent to which individuals

are connected to each other by direct relations [17]. For instance, high network density within a community has been found to increase its ability to control criminal behavior as community members are more apt to monitor and respond to such behavior. Alternatively, low network density and weak social ties can result in the inability to exert social control [17]. Social disorganization theorists have proposed that supervision is an important component within communities [23]. Crime and disorder can develop where there is lack of supervision over activities and these activities are not held in check through social control. A lack of supervision or regulation of activities within a community can lead to higher rates of crime, whereas cohesive communities are able to exert control and intervene in criminal or delinquent activities. Indeed, in studies on teenage delinquency, researchers found that gang developed in communities when teenagers were left unsupervised whereas more cohesive communities were better able to collectively supervise teenagers and control potentially delinquent behaviors [10, 17].

Overall, criminological studies of social disorganization principle have consistently demonstrated how characteristics of a disorganized community, sparse networks, unsupervised groups, instability, and heterogeneity, are important in explaining variation in crime rates across geographies. However, it is noted that these studies measured concepts related to disorganization and are thus subject to measurement error, with the potential that other underlying variables could be influencing crime rates. Despite such limitations, the results of these studies have proved to be consistent with the key principles proposed by social disorganization theory.

Drawing on social disorganization theory, we propose that Tor structures exhibit similar characteristics to offline crime-prone communities. In this study, these characteristics will be adapted in a more general sense through the results of social network analysis and what such measures reveal about the Tor network. Weak connectivity, low network density, and large diversity within Tor could indicate disorganization within these structures. Structural disorganization could be a contributing factor to the presence of illicit activity and material on Tor. Due to its anonymization features, Tor is also inherently void of regulation and guardianship, which is likely to increase or attract malicious users operating within the network. We propose that examining hyperlink structures is an important step to understanding Tor networks, as hyperlinks play a crucial role in facilitating the transmission of information and content, connecting malicious users and websites, and providing opportunities to engage in criminal activities.

2.2. Current Study. Examining the Tor network, how it is deployed, and who comprises the Tor community is an imperative undertaking to better understand the dark web. The current research aims to achieve an in-depth understanding of the network's structure through the use of innovative collection and analytic techniques. Using a specialized web crawler enables efficient collection and the ability to filter a vast amount of information from the Tor network. Subsequent analysis of this information can help determine the extent to which Tor websites are connecting

to other websites within the network through hyperlinks. Social network analysis can assess these connections and provide a macro-level understanding of the network structures within Tor. The social network measures serve as a way to analyze how legitimate and illegitimate users are able to navigate and connect on Tor. These connections are implicit in understanding the structures within the network and determining the impact of each website within this dark web. This can generate insights into what content exists within Tor and how the structures of the network may foster online illicit activity. The overall network form will be examined along with social network measures including cohesion, homophily, and core-periphery to provide insight into whether the network structures are characterized by sparse, weak, and heterogeneous social connections between domains.

Networks can take many forms but generally exhibit two dichotomous structures: random or scale-free. Random networks are composed of a disconnected set of nodes that are paired with a certain probability [24]. These networks often have low heterogeneity, in contrast to real world observed networks where edge formation is likely a product of choice [24]. Scale-free networks often follow a power-law distribution where nodes are more likely to connect to a central few nodes rather than forming connections to only unique nodes [25]. New nodes in the network are more likely to select edges to these central actors rather than follow a random pattern. These scale-free networks characterize the regular Internet, which may provide insight into the connections between nodes on the dark web (Tor). Scale-free networks may contain a central core of hubs that contain most of the connections within the network. These hubs then are important to how information and users traverse the network. Identifying these hubs through a core-peripheral model will provide insight into how a user may traverse the network with an additional focus on how criminal activity may proliferate within Tor. Legal domains acting as hubs may provide access and opportunities to illicit ones, which would be largely isolated otherwise. The presence of a core then reduces the distance that a user will have to travel to find illicit content increasing criminal opportunities.

3. Data and Methods

The Dark Crawler (TDC) is a modified version of the webcrawler presented in previous research [26–28] and is shown in detail in Appendix. It operates by collecting and downloading webpages into an offline database. Prior to data collection, a list of seed websites was compiled to provide the crawler points of departure. Seed websites are universal resource locators (URLs) that are manually selected and can be as few as one website or more than 1,000 websites. In the current study, Tor domains were used as seed websites. These Tor domains were found through a Google search of the most widely known.onion directory called the Hidden Wiki, with additional URLs found on Reddit and other regular Internet sites. The Hidden Wiki categorizes and shares known Tor domains, allowing users to search for various types of legal and illicit content. Overall, a collection of 150 seed websites

were identified to start the search process. These 150 websites were selected at random using a random number generator and through content analysis the domain was categorized (Drugs, Child Exploitation, Directory, Hosting, Weapons, etc.) The categories were modeled from previous studies [2, 5] with some categories collapsed into others (guns and hitman services were coded as weapons) while others were expanded. Often the “Other” or “Misc.” categories were used as a catchall for hard to identify websites and those were disaggregated for this study into appropriate subcategories such as politics.

Starting with the seed websites, the crawler downloaded each top level domain and added any found hyperlinks containing a .onion address to its internal queue. It recursively followed these links from the queue until either no .onion domains were found or it reached the termination criteria. For the purposes of this study, the termination criterion was the downloading of 1 million webpages across any number of domains. Among the 1 million Tor webpages that comprise the sample there were 1,220 unique domains.

3.1. Social Network Analysis

3.1.1. Homophily. Homophily looks to explain why nodes with similar features are more likely to share social relationships or ties [29]. This relationship has not been explored through the analysis of the links between top level domains in an online network. If domains are more likely to share hyperlinks with similar domains, this can be compared to homophily observed through website content. Network topography may play a more prominent role in which webpages hyperlink to other webpages rather than simply connecting to webpages with similar content. This has implications about the nature of node hyperlinkages operating within the dark web. Tor nodes are not necessarily looking to increase visibility and might only be interested in remaining part of the underground community. The presence of homophily within the network would indicate the possibility that websites were more likely to form connections with similar others. Alternatively, a lack of homophily would indicate that the network structure was not driven by homogeneous website connections but instead was driven by a motley crew of users and content. If network connections are dissimilar and diverse, this could be a contributing factor to the presence of illicit activity on Tor. As social disorganization theory proposes, diversity within a community can lead to higher delinquency as it interferes with members’ ability to develop strong social connections and effectively work together to communicate and provide surveillance within the community [10].

3.1.2. Cohesion. A scale-free network features highly centralized nodes or hubs connecting to other nodes forming large clusters [30]. Traditionally, a network that is characterized by the small-world phenomenon has an average path length of 6 [31]. The Internet is characterized as having small-world features despite containing an average path length of 19 [30]. This is due to the sheer size of the Internet which follows logarithmic scaling, where the degree centralization remains

relatively high despite the sheer volume of domains present [32]. Similar to the regular Internet, Tor will likely follow these properties, which has important considerations regarding network traversal, information flow, and law enforcement disruption.

Given the novel nature of examining Tor through social network analysis, examining these properties was essential before comparing results and the generalizability of this network to the regular Internet. If Tor does not resemble the regular Internet, then comparing the importance of the nodes within the network would be ill-advised. Determining the nature of the Tor network would also allow additional network measures to be examined on the overall network. A core-periphery analysis was conducted to determine the importance that some hubs might be having on the overall network. If there was no identified core, this would indicate that the websites were connected to each other and had a similar number of network ties. Alternatively, the presence of a core would suggest that the Tor network was operated by a few centralized hubs that serve as crucial links to the entire network. As the adoption of Tor by users remains relatively low, accessing content on Tor can be difficult without prior knowledge of where to look. To remedy this problem, it is likely that central hubs have emerged which are responsible for linking new users to onion domains. Without hubs directing users to various domains, Tor users would have difficulty accessing desired content. The Tor network takes advantage of the high density of users to operate on a peer-to-peer system which increases anonymity for the entire community. Social disorganization theory would suggest that this anonymity within a dense network increases criminal activity as it interferes with accountability to other Tor users. Further, opportunities to engage in illicit activity on Tor are not constrained by isolation. While some studies have suggested that living in a sparsely populated area can reduce opportunities for offending due to the offender’s distance from targets [33], this is unlikely to impact deviant users in the online realm. Users wishing to access or share illicit content can do so without being constrained by geographical barriers and likely desire to remain hidden and disconnected on the network to avoid law enforcement.

3.2. Core-Periphery Analysis. Core-periphery analysis has been used successfully on smaller pockets of the Internet. Researchers hypothesize that the Internet, due to its architecture, retains an inherent core structure [34]. Although researchers suggest that the core is formed by connections based upon network traffic, hyperlinks serve as another way to measure connectivity. Most of the research focusing on online communities uses core-periphery to look at user social relationships [35, 36], but few use websites to examine these connections. While the existence of a core is briefly discussed in [10], no social network analysis regarding the structure was conducted. We propose that while a core may be present in the network and even necessary for users to find content, hyperlink connections outside of the core are likely to be infrequent and weak. Tor users interested in accessing or distributing illicit content can be expected to remain isolated in order to avoid detection. Assuming website linkages are

TABLE 1: Comparison of categorical and continuous core-periphery models.

One mode categorical $N = 61$	One mode continuous <i>minres</i> $N = 8$	One-mode continuous score (Corene)	Website content	Website type
N531	N531	0.76	Offline	Offline
N514	N514	0.31	Links	Directory
N1007	N1007	0.26	Wiki	Hosting
N899	N899	0.20	Politics	Directory
N938	N938	0.20	Politics	Directory
N937	N937	0.19	Politics	Directory
N898	N898	0.19	Politics	Directory
N515	N515	0.16	Links	Directory
Fit = 0.22		Fit = 0.07		

a valid form of network construction, and the nature of the Tor network is to be covert, then examining offline covert networks appears to be the logical choice from which to draw comparisons. The Tor network most closely resembles criminal networks, where the actors are attempting to avoid detection. The types of offline networks that share these characteristics would most likely be gang affiliation networks, given that actors are actively trying to avoid detection and are interested in conducting illicit activities.

Core-periphery analysis has been shown to be effective in determining who the central actors are within numerous networks, both offline and online [34, 37, 38]. Core-periphery analysis can potentially reveal websites that provide highly important positions within the network, even though they would not appear to be influential based upon network scores or using the naked eye. A person may appear highly influential in a network while actually not being a central component of the core. Some nodes that may appear as peripheral entities actually have significance to the core of the network and would be overlooked. Previous studies utilizing the various forms of the TDC have discussed the biases and limitations associated with using this data collection technique [26]. The current sample of the Tor network relied upon the manual data collection of seed websites through content analysis which may influence what Tor nodes are represented within the network [26, 28].

4. Results

The analysis of the 1,220 nodes in the Tor sample revealed a sparse network with an average degree score of 2.27. Each node was connected to only a couple of other websites, showing the scale-free nature of the network. There were 2,763 ties between the nodes, leading to a network density of 0.002. This indicated that less than one percent of all possible connections exist within the network. Websites within the network were only connected to a few select nodes. These ties were directed, as not all hyperlinks are reciprocated back between websites. Of the 2,763 edges within the network only 136 were reciprocal between nodes, while 2,627 were not. Node reciprocity then only happened in 4.9% of all connections within the network. The average distance of this network was 4.95, meaning that it took, on average, about

5 connections to get from one end of the network to the other when considering all possible links between pairs of two nodes. The surface web, on average, has an average distance of 16–19 connections [25], suggesting that Tor exhibits a different structure. The average score additionally suggested that the network contained properties associated with small-world networks. The degree centralization for the network is 41.88%, which indicated there were scale-free features present within the network. We suggest the identification of these sparse connections and a lack of reciprocity between the hyperlinks contributes to the presence of crime among the Tor community as compared to the surface web. This allows domains to remain hidden from unwanted attention, such as law enforcement, while still retaining the ability to attract interested users.

4.1. Categorical versus Continuous Models. Multiple tests were conducted using core-periphery analysis to determine the best model fit for the dataset. The final results, reported in Table 1, compared the categorical model and the one-mode continuous model using the *minres* algorithm. The *minres* algorithm is a form of factor analysis that can be used if the diagonal values within the relational matrices are not valued and only detect the presence or absence of a tie [39]. The *minres* algorithm works best with binary networks where the presence of a tie is captured. Model fit is represented as a Pearson's correlation coefficient between the current model and an ideal model with a maximized core structure [39]. The procedure inherently maximized nodes with connections into one block in a matrix (the core) while the nodes with few connections (the periphery) were minimized and placed into a second block [39]. This immediately biased any significance testing and thus model fit was largely a descriptive process, rather than reaching an arbitrary cut-point. The categorical model suggested a core of 61 websites with a periphery of 1,159 nodes and had a model fit of 0.22. The one-mode continuous analysis was conducted using the *minres* algorithm and a core of eight nodes with 1,212 periphery nodes was suggested with a model fit of 0.07. Comparison between the categorical model and the continuous model demonstrated the categorical model was a better fit for the data given the large differences between the scores in model fit. Table 1 contains the top 8 websites as

TABLE 2: Group densities between core and periphery across categorical and continuous models.

Categorical (Fit = 0.22)	Core	Periphery	Continuous (Fit = 0.07)	Core	Periphery
Core	0.132	0.019	Core	0.250	0.160
Periphery	0.001	0.001	Periphery	0.001	0.001

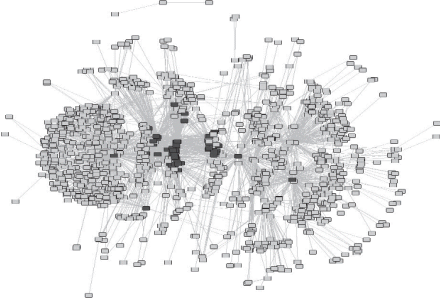


FIGURE 1: Core-periphery categorical model. Note: dark grey squares represent core nodes; light grey squares represent periphery nodes.

ranked by the continuous model and their associated core scores.

Determining the variability of subject matter through content analysis on all sixty-one core domains indicated by the categorical model proved cumbersome, therefore only the nodes present in both models were analyzed; this was a limitation with the data capture which could be solved by automated methods in future. Eight nodes from the continuous model were used to highlight the types of websites found within the core and it is important to note that these eight nodes were identified as part of the core structures of both models. The structural position of the core within the network is shown in Figure 1. With the addition of more attributes, the core 61 nodes could be assessed to determine if there were any similarities between them. There may be a correlation between domain membership and content, with certain typologies more likely to exist in the core.

A content analysis of the core eight nodes was conducted to determine if any particular type of website or content was contributing to the core. The top eight nodes all appeared to have similar domain structures (directory sites), while the content varied between the sites. Notably, the website with the highest core score was offline when the content analysis was conducted two months after data capture. While the continuous model was not selected to represent the network visually, the core scores demonstrated which core nodes were the most central to the network. The core is represented by the dark grey squares, while the periphery is represented by the light grey squares in Figure 1. Visually, these 61 nodes, given their centralization and large number of ties, represented what would be expected of a core as demonstrated by their position in the network. The large cluster of dark grey squares, shown centre left in Figure 1, represented approximately 60% of the core. Many nodes overlapped, which suggested they held a similar place within the network and may be part of the same community.

4.2. Core-Periphery Analysis. Group densities were also compared to determine the best model to use for the core-periphery analysis and are shown in Table 2. These comparisons helped to identify how connected the core is to itself and the periphery. If the core connections were not above the network average it could have indicated the model was a poor fit for the data. The density measured the proportion of connections that existed as a ratio of all possible connections available within the network [39]. In the categorical model the core density was 0.132, as compared to the overall network density of 0.002. The large discrepancy indicated that there was in fact a core present within the network. The density from the core to the periphery was .019 and from the periphery back to the core it was 0.001. The connection of the core nodes to the peripheral nodes was low, having only formed connections to 1.9% of possible nodes. From periphery to periphery the density was also 0.001, less than the total network. The peripheral connectivity was the same for both the core and other peripheral nodes making connections with only 1.16 nodes. The core nodes connected to an average of eight other members of the core. The core also connected, on average, to approximately 22 periphery nodes. This was in comparison to the periphery which linked to other peripheral nodes at just over one connection per node. These large differences indicated that there was a core present within this network and was consistent with other online communities containing scale-free properties.

In the continuous model the core density was 0.25; meaning 25% of all the possible ties between core members were present. Each core node on average connected to at least two other members in the core. The core to periphery was 0.16, which was also quite large given that the periphery consisted of 1,212 nodes. On average, the core nodes connected to 194 periphery nodes within the network. The periphery back to the core was 0.001 and the periphery to the periphery nodes remained the same at 0.001. While the continuous model had a much denser core, the categorical model retained connectivity while including more nodes. This was ideal for identifying the nodes that possessed a significant number of the connections within the network. Adding the ties from all dyadic pairs within the core, and ties from the core to the periphery in the continuous model, 57% of all the ties in the network were present. In the categorical model this rises to 66%, indicating that the core was responsible for almost two-thirds of all present ties.

4.3. Network Structure. The fundamental assumption underlying a network using websites as nodes was that domains can have social relationships or meaningful connections to other websites [26]. These assumed relationships suggest that websites have properties that can be considered analogous to the connections made by people. Websites at their core

are still operated by users, so while this remains a possible limitation, the hyperlinks are still chosen by individuals, an action no different than choosing friendships offline. Comparing networks across group typologies then should be considered in this context, such as those containing child exploitation materials [4, 30], terrorist networks [40], or abstract structures [25, 32]. Domains hyperlinking to other domains are a product of this decision-making process and do not significantly differ from how edges are formed in other contemporary groups in different contexts. In this study, examining the network characteristics revealed that Tor shared some similarities with these other observed networks. Without an extensive meta-analysis of all network types, it was impossible to place Tor into a specific category.

Comparing results across online networks can help to clarify how online networks operate. For example, a comparison of the cohesion measures indicated that the density of 0.002 found in the Tor network was lower than those found in two regular Internet child exploitation networks (0.05 and 0.11, resp.) [30]. It is possible this observed difference was due to network size; however, networks containing CE materials may be the best comparison available due to criminal domain saturation. Comparing the features of the network to licit alternatives provides little benefit except to simply say that Tor is in fact different. The average distance of 4.95 hops (the sum of all dyadic pairs between nodes) within the Tor network is consistent with the literature regarding online networks exhibiting small-world properties [32]. As defined by Milgram, small-world properties are networks with an average path length of six or less [41]. In addition to containing properties associated with small-world networks the Tor network also had features consistent with a power-law distribution that defines scale-free networks [32]. This was measured through the degree centralization score, which in the Tor network was 42%; this was quite large and surpasses other online networks [32, 42]. Although it was difficult to speculate as to the exact network structure of Tor without a more exhaustive review, the Tor network appeared to share similar properties with other online networks although definitively more centralized.

5. Discussion

5.1. Core-Periphery. The results of the core-periphery model were consistent with the findings regarding the overall degree centralization of the network. As noted above, the reported degree centralization of 42% indicated the network contained properties associated with a scale-free network: few hubs with many connections. This indicated there were some parallel attributes between the regular Internet and the dark web, as both contained features of a scale-free network. A core of 61 nodes (or 5% of the network) was identified and had an overall group density of 0.132. At 18,019, the core to periphery densities were also larger than those of the overall network (0.002). This core contingent accounted for a significant portion of all ties in the network and connections to other core nodes, indicating this data reduction technique may be useful for examining the Tor network. Core-periphery analysis works best in an environment that cannot be divided

reasonably into cohesive subgroups [39]. Without a dataset containing attributes to interpret these subgroups or factions, core-periphery allowed for the identification of a subset of nodes containing a significant majority of the connections [39]. The Internet was also typically considered to reflect a core-periphery model as conceptualized by [34, 43] when they describe how links are formed through specific network paths, as opposed to random chance. These authors suggested the Internet was inherently designed in such a way that information travels along designated paths to form a core [34]. The idea that new nodes within the network were more likely to form connections with existing popular nodes. For example, a newly registered domain will link to Google™ at a significantly higher rate than cbc.ca. Again, the Tor network appeared to operate in a similar manner, with websites connected to a small number of central nodes rather than similar peripheral websites. This demonstrated that new nodes, or websites, were more likely to form connections to older, more well-established websites than to newer sites. New domains on Tor were more likely to link to this centralized core than form connections to peripheral nodes. This could be due to the relative anonymity associated with domain creation within Tor, where users are simply unaware of the other nodes' existence. Either way it has implications for law enforcement regarding authorship and domain generation, nodes which connected outside of the norm to more peripheral nodes may be authored or created by the same individual.

The results also deviate from those found in relation to an offline gang network [44], insofar as the current study suggested a much denser core. The similarities between the two types of networks both being elusive and illicit in nature did not seem to link these structures. The core-periphery analysis in the current model is more conducive to Internet-based networks, suggesting that network topography plays a larger role than network typology. Online networks may not be comparable to offline networks despite the symmetries in the content of those networks due to the scale-free nature of online networks. The presence of overlapping nodes indicated they belong to the same community on the network. This finding suggested that connections between websites were not driven by homogeneous characteristics and content. The core-periphery analysis also suggested that connections on the network are dispersed and cannot be tied into reasonably cohesive subgroups. It was also more difficult to determine any key members within the network – obfuscating the entities that produced the majority of illicit activity or content. This made sense given the Tor network is employed by users aiming to avoid detection by law enforcement.

The absence of homophily in the network structures also showed that websites were dissimilar in content and that ties were not created or limited by shared characteristics. However, this can be seen as a benefit for Tor users. Forming connections with dissimilar websites can increase embeddedness and the ability of users to operate without being tracked or monitored, a key objective of Tor users. These ties were also more likely to dissolve, which may contribute to the transient nature of the Tor network [29]. As social disorganization theory suggests, the diversity of users can interfere with the Tor

community's ability to form relationships between users and develop effective communication. Without communication, surveillance of activities on the network is greatly inhibited and dissimilarity between users can instead breed mistrust among the community. Thus, the weak ties and heterogeneity found within network communities lend credence to the possible roles these characteristics have in fostering illicit activity on the Tor network.

Online illicit networks provide an additional advantage for individuals seeking to engage in malicious or illicit activities as compared to offline illicit networks. Unlike offline networks, websites do not need to spend costly resources to maintain social capital and sustain relevancy within the network. Thus, core membership was likely less intensive in online networks, allowing these key sites to maintain their structural positioning within the network. This was different from offline networks where core-periphery structures provided different network contexts. Larger cores allowed the network to sustain its structure if users left, while larger peripheries allowed more adaptability within networks [44]. For law enforcement removing nodes within a network with a large core would have little disruption effect as the pathways through the network are easily attained through other avenues. Targeting of nodes for removal then should not be based upon the principles of disruption or fragmentation but instead should be aimed towards deterrence or desistance where the impact would affect the largest number of users possible. Cryptomarkets and large forums would make ideal targets where users have made investments into accounts whether financially or through gaining trust and reputation. Additionally, dispersed cores are not impacted when peripheral nodes continually enter and exit the network. A single core node does not serve as the only linkage between the cluster of peripheral nodes and the rest of the network. Conversely, removing core nodes within larger peripheries, as in the case of the current study where the core is characterized by scale-free features, will have a substantially larger impact on network traversal. For example, the targeted law enforcement efforts on the "Silk Road" by the FBI saw a large disruption effect for Tor users who were not simply displaced to competing cryptomarkets [12].

In determining if the Tor network shared similar properties to the regular Internet, it is likely that the connections between domains are formed using the same principles [25]. Edge formation was not random and not all nodes within the network have the same probabilities of forming connections [25]. There was, to some degree, a form of preferential selection taking place, where new nodes were more likely to link back to these already established core nodes. The encrypted nature of Tor means information is not routed in exactly the same way, as data still travels through a designated path. Core modeling of the regular Internet further reinforced the use of this analysis in assessing these models within a social network context [34]. The 61 core nodes identified through the categorical model should be compared further with attribute data to speculate about the types of websites comprising this crucial component. The results of the content analysis for the core eight domains indicated that over half are primarily focused on supplying

hyperlinks to other domains on Tor. Due to the anonymity and obscurity of Tor, it is likely these directory sites are necessary for users to find content. If the dissemination of information on Tor is reliant upon these directory sites to such a large degree, it has implications for how crime occurs within this context. Future research is necessary to assess if directory sites are linking illegal or criminal websites to other criminal sites or if these sites are forming their own communities.

Following a power-law distribution, the network properties of Tor indicate that it is less likely to recover following domain failures of the hub nodes [25, 43]. Two important considerations are that this leaves the network vulnerable to attack from groups looking to disrupt the network (law enforcement, hacktivists, governments, etc.) or internal causes such as the lack of infrastructure may cause significant disruptions to accessibility and network fluidity. While removing one node remains a challenge and has shown to be a costly endeavour by law enforcement [12], targeting of a select few nodes representing the core structure may have larger and more impactful disruption effects than focusing on the criminal activities within a single domain. The core structure will also regulate how new users can move from domain to domain on Tor likely having to pass through a hub before specific URLs are identified. A hub which is offline for a significant amount of time may vastly restrict the access of users to content which may funnel them into following a different path. Consider the example of a user looking to purchase a firearm in the context of the results. They download the Tor browser and follow the steps and find the Hidden Wiki. This domain has links to a few domains which reportedly sell firearms; however the links are dead or broken. The person decides to try one of the marketplaces and see if someone is selling there. On the marketplace someone may be advertising a different domain which sells firearms because it may go against the policies of the marketplace. The person then follows the hyperlink to the final domain and is able to purchase the gun. Alternatively, in a less centralized network, the domain which sells firearms may be directly hyperlinked by many websites and is accessible by simply clicking it directly from the Hidden Wiki. The addition of one domain does not seem like a preventative factor in stopping a motivated user from purchasing the firearm but due to the factors listed above may be critical. The user needs to find the content on the marketplace, law enforcement has another avenue for intervention and both need to remain stable long enough for this user to make the critical leaps.

6. Conclusions

The Tor network is a relatively unknown and unexplored region of the dark web allowing users to anonymously communicate and browse content through encrypted means. Past research has speculated on the content of Tor through limited studies focused on small sections of the network [5, 7]. A more thorough and systematic process was necessary to conduct analysis of the Tor network. In the current study, an automated tool was used to collect data and social network analysis was applied to examine how websites formed

connections through hyperlinks to other websites. This type of analysis adds a further dimension to how users access content on Tor.

The results of the social network analysis provided insight into the characteristics of the virtual communities formed through hyperlink connections. Support for a homophily effect regarding the way Tor nodes were connecting to others within the network was found. This was also supported by the notion that the Tor nodes had, on average, more incoming ties than the link sites. The core-periphery model identified the main 61 central nodes within the network and guide users through this dark network. Although there was a slight homophily effect regarding the core nodes, websites within the networks appeared to be largely composed of heterophilous connections. As such, similar content did not appear to be the driving force connecting websites. Reference [4] found a similar finding when analyzing online child exploitation communities and suggested that connections with similar content may have a negative impact; similar connected websites may experience decreased traffic to their domains as users can access the information on competitor's sites. When considered under a social disorganization perspective, it is also probable that dissimilar and heterogeneous connections are contributing to the Tor community's inability to effectively surveil deviant users and activity.

Social disorganization principles can provide insight into how network structures may foster illicit activity on Tor. The sparse online networks appear to share similar characteristics to real world crime-prone neighborhoods; they are largely composed of weak and heterophilous ties between domains. Although it was not specifically examined in this current study, Tor is also known for its inherently transient nature; a vast number of domains are created and taken down on a daily basis. Such instability may amplify the facilitation of illicit activity; users specifically choose to operate on Tor and other dark webs to avoid detection and obfuscate their activities. The extension of a criminological theory and concepts should be further explicated to understand how dark web community structures could be fostering illicit activity. The integration of criminology theory is useful to the extent that it can potentially help inform police strategies and reduce criminal activity by targeting formed network communities. Further, it would be advantageous to take a later sample of the data to determine if the characteristics of Tor network structures remain consistent. Due to the sheer volume and size of Tor, measuring change over time within the network structures is needed to refine and corroborate our findings.

As the data was not collected through a randomized sample and may be reflective of the seed websites, the results are not without limitations. For example, if the seed websites were entirely focused on drug markets, without attribute data it would not be possible to tell if the entire sample consists of drug related websites. The seed sites may be biasing the sample based upon content rather than overall network position. The lack of attribute data hampers the ability of the data to be generalizable to a broader sample without first examining what types of websites were collected. Future research should include as many attributes as possible from

each website such as content, legality, page views, visitors, registered users, and Bitcoin wallets to allow comparison between the network measures and website characteristics as was previously done in child exploitation research [4]. The attributes would also allow for community detection to be used to examine how these websites were grouping together to form connections. The core-periphery analysis used in the study examined only one set of these communities and it is likely that many others exist. Moreover, applying criminological concepts and measures to the online realm has many challenges and this study did not empirically test social disorganization theory or examine how other structural factors can mediate or influence the relationship between communities and crime [23].

Dark web networks have received significant attention in the past decade, in both the public and political realm, and are being viewed as the only way to protect one's information and privacy in a world where the collection and flow of personal data are nearly inescapable for Internet users. It remains an important task for researchers and law enforcement to pursue a further understanding of how these networks are exploited by individuals and entities for criminal purposes, as this may better inform strategies to disrupt illicit activities on Tor and other dark webs. By understanding the characteristics that foster criminal structures on the network, this could aid enforcement in targeting malicious users and removing illicit content. Still, only a portion of the Tor network was explored in this study, leaving a significant portion still in the dark. Employing social network analysis to examine more domains and the networks they form will allow for a better picture of how users are traversing the Tor network and can better indicate whether social disorganization principles can provide an understanding of crime on Tor.

Appendix

In the seed data collection 1000 domains were found using Google, Reddit, and the Hidden Wiki. Of which 150 were selected using a random number generator to seed the crawler to start the data collection period. This was done to eliminate potential selection bias where if a core structure was identified it would simply represent the seeds chosen. If the core was made up of primarily seed domains the network structure would more accurately represent the ego networks of the seeds rather than a representative sample of the Tor network as a whole. The results of a QAP regression (UCInet 6 v. 6.644) showed that ($R^2 = 0.13$) seed domains did not significantly predict core membership ($\exp(B) = 1.05, p > .05$); however, seed domains did have significantly greater indegree ($\exp(B) = 0.31, p < .001$) and average geodesic distances ($\exp(B) = 0.13, p < .01$). The crawler recursively followed found hyperlinks on each collected webpage subsequently stored in a queue. A webpage was scraped and the XML document which comprises the underlying structure of the webpage was stored in the crawler database. The webpage could then be viewed offline from the database and visually verified by a researcher at a later date for content analysis. Data collection could be terminated by manually cancelling the TDC after a certain time period

or by specifying a criterion. For this study TDC ran until it had collected 1 million webpages and lasted ~90 days or 3 months. The 1 million webpages comprise 1,220 unique domains. Each hyperlink found on a webpage then was collapsed into representing a specific domain. If a hyperlink was found on page 1 or page 820 (the mean number of webpages per website) of Website A both represented a tie to Website B and C and was coded as such. The hyperlinks were then aggregated for each domain and used to generate the networks.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

- [1] S. Lu, "What is the dark web and who uses it? The Globe and Mail," <https://www.theglobeandmail.com/technology/tech-news/what-is-the-dark-web-and-who-uses-it/article26026082/>.
- [2] S. Mansfield-Devine, "Darknets," *Computer Fraud & Security*, vol. 12, pp. 4–6, 2009.
- [3] S. Dredge, "What is tor? A beginners guide to the privacy tool," *The Guardian*, 2013, <http://www.theguardian.com/technology/2013/nov/05/tor-beginners-guide-nsa-browser>.
- [4] B. G. Westlake and M. Bouchard, "Liking and hyperlinking: Community detection in online child sexual exploitation networks," *Social Science Research*, vol. 59, pp. 23–36, 2016.
- [5] C. Guitton, "A review of the available content on Tor hidden services: The case against further development," *Computers in Human Behavior*, vol. 29, no. 6, pp. 2805–2815, 2013.
- [6] K. Misata, "The Tor Project," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 20, no. 1, p. 45, 2013.
- [7] A. J. Kim, *Community building on the web: Secret strategies for successful online communities*, Addison-Wesley Longman Publishing Co., Inc, Boston, MASS, USA, 2000, <http://dl.acm.org.proxy.lib.sfu.ca/citation.cfm>.
- [8] G. Moura, R. Sadre, and A. Pras, "Bad neighborhoods on the internet," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 132–139, 2014.
- [9] N. Christin, "Traveling the Silk Road: A Measurement of a Large Anonymous Online Marketplace," Defense Technical Information Center, 2012.
- [10] C. R. Shaw and H. D. McKay, *Juvenile delinquency and urban areas*, University of Chicago Press, Chicago, Ill, USA, 1942.
- [11] D. Moore and T. Rid, "Cryptopolitik and the darknet," *Survival*, vol. 58, no. 1, pp. 7–38, 2016.
- [12] D. Décary-Héty and L. Giommoni, "Do police crackdowns disrupt drug cryptomarkets? A longitudinal analysis of the effects of Operation Onymous," *Crime, Law and Social Change*, vol. 67, no. 1, pp. 55–75, 2017.
- [13] I. Goodwin, "Book Review: H. Rheingold. 1993. The Virtual Community: Homesteading on the Electronic Frontier. Reading, Massachusetts: Addison-Wesley. ISBN 0-201-60870-7 H. Rheingold. 2000. The Virtual Community: Homesteading on the Electronic Frontier (2nd Edition). C," *Westminster Papers in Communication and Culture*, vol. 1, no. 1, p. 103, 2015.
- [14] R. P. Bagozzi and U. M. Dholakia, "Intentional social action in virtual communities," *Journal of Interactive Marketing*, vol. 16, no. 2, pp. 2–21, 2002.
- [15] R. J. Sampson and R. B. Groves, "Community structure and crime: testing social disorganization theory," *American Journal of Sociology*, vol. 94, no. 4, pp. 774–802, 1989.
- [16] J. Bursik, "Ecological theories of crime and delinquency since Shaw and McKay," *Annals of the American Academy of Political and Social Science*, vol. 338, pp. 119–136, 1984.
- [17] R. J. Sampson, "Neighborhood and crime: The structural determinants of personal victimization," *Journal of Research in Crime and Delinquency*, vol. 22, no. 1, pp. 7–40, 1985.
- [18] F. E. Markowitz, P. E. Bellair, A. E. Liska, and J. Liu, "Extending social disorganization theory: Modeling the relationships between cohesion, disorder, and fear," *Criminology*, vol. 39, no. 2, pp. 293–319, 2001.
- [19] R. Kornhauser, *Social sciences of delinquency*, University of Chicago Press, Chicago, Ill, USA, 1978.
- [20] J. Byrne and R. J. Sampson, "Key issues in the social ecology of crime," in *The Social Ecology of Crime*, J. Byrne and R. J. Sampson, Eds., pp. 1–22, Springer-Verlag, New York, NY, USA, 1986.
- [21] M. E. Cahill and G. F. Mulligan, "The determinants of crime in Tucson, Arizona," *Urban Geography*, vol. 24, no. 7, pp. 582–610, 2003.
- [22] S. Wouter and J. R. Hipp, "A longitudinal test of social disorganization theory: Feedback effects among cohesion, social control, and disorder," *Criminology*, vol. 49, no. 3, pp. 833–871, 2011.
- [23] C. E. Kubrin and R. Weitzer, "New directions in social disorganization theory," *Journal of Research in Crime and Delinquency*, vol. 40, no. 4, pp. 374–402, 2003.
- [24] A. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
- [25] A. L. Barabási and E. Bonabeau, "Scale-free networks," *Scientific American*, vol. 288, no. 5, pp. 50–59, 2003.
- [26] B. Westlake, M. Bouchard, and R. Frank, "Finding the key players in online child exploitation networks," *Policy and Internet*, vol. 3, pp. 1–25, 2011.
- [27] M. Bouchard, K. Joffres, and R. Frank, "Preliminary analytical considerations in designing a terrorism and extremism online network extractor," *Intelligent Systems Reference Library*, vol. 53, pp. 171–184, 2014.
- [28] B. Monk, R. Allsup, and R. Frank, "LECENing places to hide: Geo-mapping child exploitation material," in *Proceedings of the 13th IEEE International Conference on Intelligence and Security Informatics (ISI'15)*, pp. 73–78, usa, May 2015.
- [29] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: homophily in social networks," *Annual Review of Sociology*, vol. 27, pp. 415–444, 2001.
- [30] K. Joffres and M. Bouchard, "Vulnerabilities in online child pornography networks," in *Using Network Analysis to Prevent Crime. Crime Prevention Studies*, A. Malm and G. Bichler, Eds., Criminal Justice Press, New York, NY, USA, 2015.
- [31] G. Kadianakis and K. Loesing, "Extrapolating network totals from hidden-service statistics," Tor Tech Report 01(001), 2015, <https://research.torproject.org/techreports/extrapolating-hidserv-stats-2015-01-31.pdf> Retrieved from.
- [32] A.-L. Barabási, "The physics of the Web," *Physics World*, vol. 14, no. 7, pp. 33–38, 2001.
- [33] L. Cohen and M. Felson, "Social change and crime rate trends: A routine activity approach," *American Sociological Review*, vol. 44, no. 4, pp. 588–608, 1979.
- [34] R. Pastor-Satorras and A. Vespignani, *Evolution and structure of the internet: A statistical physics approach*, Cambridge University Press, Cambridge, UK, 2004.

- [35] J. C. Wang and C. C. Chiu, "Recommending trusted online auction sellers using social network analysis," *Expert Systems with Applications*, vol. 34, no. 3, 2008.
- [36] S. L. Toral, M. R. Martínez-Torres, and F. Barrero, "Analysis of virtual communities supporting OSS projects using social network analysis," *Information and Software Technology*, vol. 52, no. 3, pp. 296–303, 2010.
- [37] M. Bouchard and R. Konarski, "Assessing the core membership of a youth gang from its co-offending network," in *Crime and Networks*, C. Morselli, Ed., Criminology and Justice Series, Routledge, New York, NY, USA, 2014.
- [38] H. Jeong, B. Tombor, R. Albert, Z. N. Oltval, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, pp. 651–654, 2000.
- [39] S. P. Borgatti and M. G. Everett, "Models of core-periphery structures," *Social Networks*, vol. 21, no. 4, pp. 375–395, 2000.
- [40] J. Xu and H. Chen, "The topology of dark networks," *Communications of the ACM*, vol. 51, no. 10, pp. 58–65, 2008.
- [41] S. Milgram, "The small world problem," *Psychology Today*, vol. 1, no. 1, pp. 61–67, 1967, <http://snap.stanford.edu/class/cs224w-readings/milgram67smallworld.pdf>.
- [42] L. A. Adamic, "The Small World Web," in *Research and Advanced Technology for Digital Libraries*, pp. 443–452, Springer, Berlin, Heidelberg, Germany, 1999.
- [43] A. L. Barabási, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: the topology of the world-wide web," *Physica A*, vol. 281, no. 1, pp. 68–77, 2000.
- [44] B. Hoppe and C. Reinelt, "Social network analysis and the evaluation of leadership networks," *The Leadership Quarterly*, vol. 21, no. 4, pp. 600–619, 2010.

Research Article

An Approach for Internal Network Security Metric Based on Attack Probability

Chun Shan ¹, Benfu Jiang,¹ Jingfeng Xue ¹, Fang Guan,¹ and Na Xiao^{1,2}

¹Beijing Key Laboratory of Software Security Engineering Technique, Beijing Institute of Technology, 5 South Zhongguancun Street, Haidian District, Beijing 100081, China

²State Grid Jibei Information & Telecommunication Company, Beijing 100053, China

Correspondence should be addressed to Jingfeng Xue; xuejf@bit.edu.cn

Received 2 November 2017; Revised 10 February 2018; Accepted 15 March 2018; Published 24 April 2018

Academic Editor: Zheng Yan

Copyright © 2018 Chun Shan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A network security metric may provide quantifiable evidence to assist security practitioners in securing computer networks. However, research on security metrics based on attack graph is not applicable to the characteristics of internal attack; therefore we propose an internal network security metric method based on attack probability. Our approach has the following benefits: it provides the method of attack graph simplification with monitoring event node which could solve the attack graph exponential growth with the network size, while undermining the disguise of internal attacks and improving the efficiency of the entire method; the method of attack probability calculation based on simplified attack graph can simplify the complexity of internal attacks and improve the accuracy of the approach.

1. Introduction

With the rapid development of network and information technology, the role of information system in enterprise becomes more and more important. At the same time, the number of attacks from internal network has also increased. Therefore, it is necessary to build an effective security metric technology for the internal network.

According to the definition and analysis of internal attacks provided by Computer Emergency Response Team (CERT) [1], the internal attacks have the transparency to defense intercepts, such as access control or firewalls. Internal attacks also have the camouflage system privileges, high risk to access the core confidential resources easily, and the complexity of gradual attacks. The security metric as a proactive defense technology, whose role is actively analyzing and evaluating what is existing in the current security risks or potential security risks before the attacks. When the attack action occurred, the security metric method needs to analyze and assess the threat of attack incidents, then predict the attack paths, and take appropriate measures to defend [2].

The analysis method in network security can be divided into two types: one is the unknown vulnerabilities in a network, mainly considering the prevention measures; the other one is the known vulnerabilities in a network, repairing the weak parts of the network and improving the security of the whole network. As for the unknown vulnerabilities, the information security experts have already carried out a lot of research; the main methods are as follows:

- (i) Analysis protocol vulnerabilities, such as ARP address resolution protocol: researchers try to find out the protocol vulnerabilities, sum up the vulnerability in some areas, give the solutions for the lack of agreement, and achieve the purpose of prevention.
- (ii) Analyze the source code of the software: mistakes are unavoidable when programming, such as buffer overflow vulnerabilities. By studying some important codes, researchers take necessary precautions against possible errors and give patches of software, so as to improve the overall safety of software.

Although these methods are effective, they are very abstract and not easy to implement, and the results are

relatively few. But if we start from the known network security vulnerabilities, it is relatively easy, for example, all kinds of graph theory based model checking methods, such as attack graph. The attack graph is a kind of graph theory method to judge the network security by studying the nodes and the relationship between nodes in the network. By building the actual network into a theoretical graph theory model, the attack graph can give us many places to think deeply, sometimes with unexpected results. For example, constructing a model from the known aspects to simplify or idealize the actual elements allows us to focus on the most important or important aspects of cybersecurity, ignoring the secondary and quickly determining the security of the network. The attack graph model has great advantages over other assessment models, becoming one of the most widely used and most studied security metrics models.

Although the attack graph can visually indicate the origin and destination of the network attack, it cannot quantitatively describe the network security. In order to conduct quantitative analysis of the possibility of attacks, we introduce the cumulative reachable probability for each node. Above all, we proposed an internal network security metric method based on attack probability to solve the problem of the existing security metrics with attack graph for the internal network.

2. Related Work

The numerous existing researches on network security metrics based on attack graph mainly focus on the representation of attack graph models, the metrics of indicators, and the conclusions of network security metric. Those early researchers conducted research mainly including the following aspects.

The representation of attack graph models. Xie et al. [3] firstly explored three sources of uncertainty in the attack graph, but the attack graph model they established is carrying on probability derivation only when the attack behaviors are determined, resulting in the fact that the probability of uncertainty testing data is not calculated in the final derivation process. Wang et al. [4] proposed the probabilistic attribute description of the attack graph based on the probability of attacks and the cost of the network deployments, using the method of cumulative reachable probability to evaluate the safety of the whole network, but they did not take into account the impacts of other uncertain factors.

The metrics of indicators: Li et al. [5] used CVSS to evaluate vulnerabilities and proposed a general approach for the network security metrics based on vulnerabilities, but they only considered the probability of a single vulnerability node, while ignoring the vulnerability of the vulnerability node in the whole system, especially the indicators between the vulnerability nodes.

The conclusions of network security metric: in terms of attack probability calculation, Wang et al. [6] use Bayesian network algorithm to calculate the risk probability for internal nodes and quantify the node variables, the node variable values, and the conditional probability distribution. Based on the improved likelihood weighting algorithm, the calculation of Bayesian network node parameter is more convenient; the internal threat forecast also is more accurate. However,

this approach did not take into account the vulnerability of their own indicators. Zhang et al. [7] proposed satisfying the temporal order of attack evidence, using the Bayesian network algorithm to analyze the security for all attack paths. However, the probability confidence of nodes in the attack graph is complicated and lacks mathematical theory, and the computational model is also too complicated to work efficiently.

We proposed an internal network security metric of the attack probability based on the attack graph model [8] in this paper. Because of the internal attacks' characteristics of camouflage and complexity, we decided to add the monitoring event node and the key-value pair in the attack graph. Compared with other security metrics, our internal network security metric improved the efficiency and the accuracy obviously with the help of attack graph simplification method and cumulative reachable probability calculation method.

3. An Approach for Internal Network Security Metric Based on Attack Probability

3.1. Method Overview. In order to understand the characteristics and the occurrence environment for the internal attacks, first of all, according to the original attack graph and the attack evidence provided by the security monitoring system, we could get the temporal difference relationship of the monitoring event nodes and simplify the attack graph with the temporal difference relationship. Second, we divide the simplified attack graph into key-value pairs and then calculate the probability of the key-value pairs. Third, we calculate the cumulative reachable probability by the method of attack probability calculation we proposed. Finally, the quantitative evaluation of the current internal network is represented by the cumulative reachable probability of target node. The specific steps are shown in Figure 1.

3.2. The Attack Graph Model. Based on the complexity of internal attacks, internal attack events are mostly multistep and continuous attack behaviors. An attack event includes multiple attack subtargets and a series of related subattack events; that is, from a resource node to the next resource, the event requires a minimum set of basic attack actions. In the initial stage, the attacker has a certain system access or operating authority and through an atomic attack can help the attacker to reach the next state node, so as to obtain more resources and permissions to achieve the next attack subtarget. Therefore, in the attack graph model constructed in this paper, the resource state node is mapped to the original attribute node; the attack action node is mapped to the original atomic attack action node. The attack graph model includes the following contents.

(1) *The Atomic Attack.* An attack action is a different instruction or a set of operations, which could divide into different basic actions. For example, to open a word file, we need two methods. Method 1: open the file by double-clicking it; Method 2: click on the file, and then click "Enter" to open it. These two methods can open the file but are composed of different operations; we will set those different

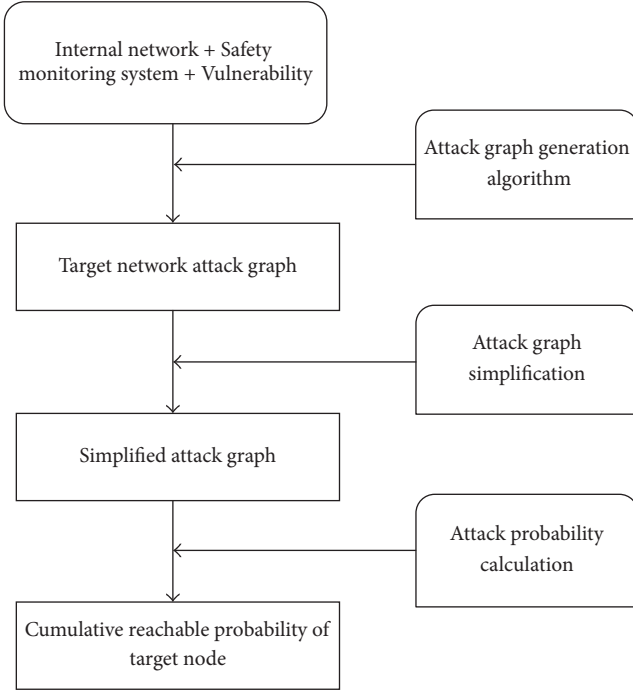


FIGURE 1: The schematic diagram of internal network security metric based on attack probability.

but similar function operations or instructions as one basic action.

And from one resource state node to another resource state node, the attacker needs a combination of multiple basic actions to achieve; such a series of basic actions is called the atomic attack. An atomic attack is the minimum set of basic actions that an attacker needs from a resource state node to another resource state node [4].

(2) *The Monitoring Event Node.* In order to simplify the attack graph, the monitoring event node is introduced in the attack graph model. Because each atomic attack contains a series of the basic attack actions, although the internal attack has camouflage, some basic actions more or less will trigger the security monitoring system, and attacks and other related information will be recorded in alarm log. The set of information recorded in the alarm log is called attack evidence. The monitoring event node refers to the attacker from one resource state to the next resource state, and the implementation of atomic attack triggers the monitoring system, recorded in the alarm log as attack evidence. We use one monitoring event node to record one kind of atomic attack's set of basic actions execute time sequence.

(3) *The Attack Graph Model.* In order to construct an attack graph model applicable to the internal network security metric, an improved probability attack graph is proposed based on the attribute attack graph [6, 7, 9]. Among them, the nodes represent the conditions of using vulnerability (the necessary resources and permissions of exploiting the vulnerability) and the use of the vulnerability of the atomic

attack action; the directed edge represents the dependency between nodes, clearing the probability of attack process distribution conditions can be intuitive to show all the attack paths that may exist in the internal network. The formal definitions of the attack graph and its constraints are as follows.

Definition 1 (attack graph model [7, 10]). $AG = (S, A, O, E, P)$ is a directed acyclic graph.

- (i) S represents the system resource state node set, $S = s_0 \cup s_i \cup s_e$ ($i = 1, 2, \dots, e$), where s_0 represents the initial node, describing the resource state that the attacker has occupied at the first time. s_i represents a single resource state node that describes the resources which an attacker gets during an attack. s_e represents the target resource state node that describes the attacker's final attack target.
- (ii) A represents the set of attack action nodes, and a_i represents an atomic attack.
- (iii) O denotes the set of monitoring event nodes. The value of node o_i can be T or F , which indicates whether the attack behaviors of a_i have been detected. When an attack action a_i occurs, the security monitoring system can capture these actions and provide the appropriate evidence of the attack recorded in the alarm log.
- (iv) E represents the set of directed edges between nodes. The attack graph defined in this paper is a directed acyclic graph; the edges between the various nodes are directed edges. $E = E_a \cup E_b \cup E_c$, E_a is $S \times A$, represents the attacker has some resources before they can initiate an attack; $E_b = A \times S$ represents the attacker could get some resources in the condition of the attack action success; $E_c = A \times O$ represents the corresponding attack evidence of the attack action captured by the security monitoring system.
- (v) P represents the probability of attack, $P = P_A \cup P_{AS}$, where P_A represents the probability that the attack behavior a_i occurs after having some resources; that is, the probability of attacking the attack action node, P_{AS} , represents the probability that a_i succeeds into the next resource state s_j , that is, the success probability of attack action.

(4) *The Directed Edges Relationship.* Considering the attack graph is a directed acyclic graph, it is necessary to effectively define the relationship between the edges which point to the same node, including "AND" and "OR" relationship. The specific content is Definition.

Definition 2 (directed edges relationship).

- (i) The "AND" relationship between two state nodes s_i and s_j , indicating that the attacker needs to have both of the resources in order to carry out the next attack.
- (ii) The "OR" relationship between two state nodes s_i and s_j , indicating that if the attacker has any of these two resources, he can proceed to the next attack.

Procedure IsAvailableMonitor**Input:** AO, EO

//AO - Sequence of attack evidence of a monitoring event node in the alarm log;

//EO - Attack evidence sequence of the corresponding atomic attack action

Output: The confidence value of the monitoring event node**Method:**

```

(01) String function(AO)
(02) initialize EO
(03) A= getS(AO)
(04) t=0
(05) S={}
(06) for i=1:EO.size()
(07)   count=0;
(08)   for j=1:A.size()
(09)     n=j, m=i
(10)     while (A(n)==EO(m) && m<=A.size()&&n<=EO.size())
(11)       count++, m++, n++;
(12)   t++;
(13)   S{t}=count
(14) f=max(S)
(15) if (f>= EO.size()) return True;
(16) else return False;

```

PSEUDOCODE 1: The pseudocode of confidence analysis.

- (iii) The “AND” relationship between two attack action nodes a_i and a_j , indicating the attacker needs two attack actions to occupy the next state node.
- (iv) The “OR” relationship between two attack action nodes a_i and a_j , indicating that the attacker can send any attack to occupy the next state node.
- (v) There is a “OR” relationship between two attack action nodes pointing to the same monitoring event node o_i ; that is, any attack action can independently trigger o_i .

(5) *Temporal Difference Relationship*. The temporal difference relationship means that if the monitoring event node a_i is detected by the security monitoring system earlier than the monitoring event node a_j all the time, then we could say the monitoring events a_i and a_j have a temporal difference relationship.

3.3. *The Method of Attack Graph Simplification*. Although the time complexity of the current attack graph generation algorithm [10] can be controlled in $O(n^2)$ (n is the number of hosts in the network) [11], the network structure becomes more and more complex and the connection between the various terminal nodes becomes more and more close, resulting in an increasingly complex attack graph structure. But the happening probability of some attack paths is very low or does not satisfy the current actual situation. The removal of these paths does not influence the whole metric model, and even their existence only increases the amount of subsequent work. Therefore, we propose the monitoring event node to prune the attack path in attack graph to simplify the whole structure of attack graph.

(1) *Pruning Attack Nodes with Confidence Analysis*. The alarm log as input, we reference the attack evidence confidence analysis algorithm defined by Wang et al. [6]; the probability of basic action of attack evidence covers the basic action set in atomic attack determining the confidence of monitoring event. For example, assume the basic attacks contained in the atomic attack are $a = \{ba_1, ba_2, ba_3, ba_4\}$. If the attack evidence provided by the security monitoring system is $a_1 = \{ba_1, ba_5, ba_7, ba_2, ba_3\}$, the coverage rate is 0.75, and the attack evidence value of a_1 is F . Assuming another evidence is $a_2 = \{ba_7, ba_1, ba_2, ba_3, ba_5, ba_4, ba_6\}$, the coverage rate is 1, and the attack evidence confidence value of a_2 is T . Then we should remove all the attack action nodes of the attack evidence confidence value F . The pseudocode implementation of the algorithm is shown in Pseudocode 1.

The complexity of confidence analysis is $g(n) = O(kn)$, n is the number of atomic attack nodes, and k is the number of basic attacks. Because the number of basic attacks is a constant, we can get $g(n) = O(kn) = O(n)$.

(2) *Pruning Paths with Temporal Difference Relationship*. According to the alarm log provided by the security monitoring system, we can obtain the temporal difference relationship of the monitoring event node. Then prune the path of the existing attack graph with temporal difference relationship; the attack paths would be deleted which do not match the temporal difference relationship; otherwise the attack paths will be preserved. Finish all these jobs, and the simplified work of the attack graph is completed.

For example, in Figure 2, the cycle a_i is the basic action of attack evidence, the square s_j is the state node of hosts [6], and the cycle o_k is the monitoring event nodes. We could know

Procedure IsTemporalDifferenceRelationship
Input: G,Otd,O1,O2,Wo1,Wo2
 //Attack graph G; Otd --The temporal difference relationship of the
 //monitoring event nodes; O1,O2 -- Two monitoring event nodes of
 //temporal difference relationship; Wo1, Wo2 -- Two sets of sequence
 //that let the values of O1,O2 are T.
Output: Whether there is a timing difference between Wo1 and Wo2,
 “Yes” or “No”.
Method:
 (01) $b \leftarrow G$ //The topological sequence of attack action nodes
 (02) if $((O1 \rightarrow O2) \in Otd)$
 (03) foreach $(A1 \in Wo1, A2 \in Wo2)$
 (04) if $((A2 \rightarrow A1) \in b)$ return No;
 (05) end for (03)
 (06) return Yes;
 (07) else (02)
 (08) return Yes;
 (09) end if (02)

PSEUDOCODE 2: The pseudocode of confidence analysis.

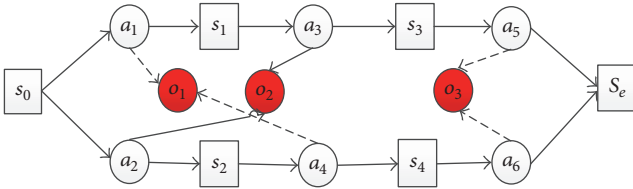


FIGURE 2: The temporal difference relationship.

there are two paths from the initial state node s_0 to the final target node s_e .

Path 1: $s_0 \rightarrow a_1 \rightarrow s_1 \rightarrow a_3 \rightarrow s_3 \rightarrow a_5 \rightarrow s_e$

Path 2: $s_0 \rightarrow a_2 \rightarrow s_2 \rightarrow a_4 \rightarrow s_4 \rightarrow a_6 \rightarrow s_e$.

If we do not consider the temporal difference relationship between the monitoring event nodes, then the conclusion is that the attack is likely to have two attack paths. But when we consider the temporal difference relationship of the monitoring event nodes, the temporal difference relationship of the monitoring event nodes in Figure 2 is $o_1 \rightarrow o_2 \rightarrow o_3$. Because Path 2 triggers the monitoring event o_2 at the first time, not matching the temporal difference relationship of the monitoring event nodes provided by the security monitoring system, Path 2 should be deleted. Similarly, if the temporal difference relationship is $o_2 \rightarrow o_1 \rightarrow o_3$, delete Path 1 that needs to be deleted. The pseudocode implementation of the algorithm is shown in Pseudocode 2.

The complexity of the temporal difference relationship is decided by the number of attack paths in attack graph which shows that the problem is NP-hard. According to attack graph generation algorithm [10], it can be controlled in $O(n^2)$ (n is the number of hosts in the network).

According to the above definition, we designed the schematic diagram of attack graph simplification in Figure 3.

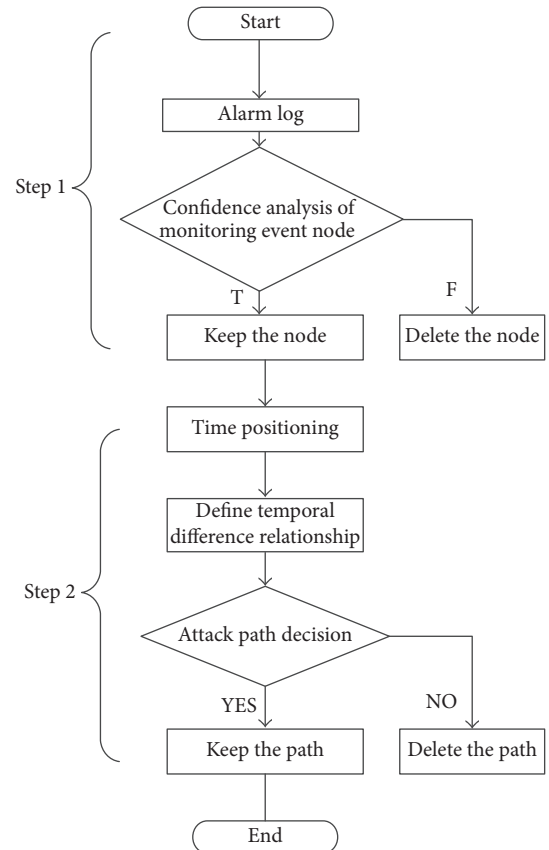


FIGURE 3: The schematic diagram of attack graph simplification.

The two steps in Figure 3 would simplify attack graph. By the monitoring event nodes confidence analysis and the temporal difference relationship of monitoring events, we can remove the interference attack graph nodes and attack

TABLE 1: The weight value of hosts.

Name	Host location	Indicator value	Description
H_i	Office network	2	Host is located on the office network, easy to be used
	Core network	1	Host is located on the core network, difficult to be used

paths and improve the efficiency and accuracy of our security metric.

3.4. The Method of Attack Probability Calculation. After the attack graph simplification, each subpath is a subprocess in which the attackers initiate an atomic attack by using the vulnerability state of the current resource state and obtain more resource states after attacking. Therefore, we propose an approach for dividing the attack graph nodes into key-value pairs composed of “resource state node, attack action, and resource state node.” The probability of key-value pairs is defined as $P = (P_A, P_{AS})$. P_A is the probability of attack action, which represents the probability from the first resource state node to the attack action node; P_{AS} means the probability from the attack action node to the next resource state node, also called the probability of attack success.

This paper proposes the cumulative reachable probability calculation method for target node with the simplified attack graph. The specific steps are as follows.

(1) *Divide the Key-Value Pairs.* According to the dependency relationship between the simplified attack graphs, divide all nodes and edges into the key-value pairs in the form of “resource state node, attack action, and resource state node.”

(2) *Calculate the Initial Attack Probability.* (a) The probability of an attack action is P_A .

The formula is

$$P_A = \frac{(V_s + H_i)}{12}. \quad (1)$$

H_i represents the weight where the host is located. The location of the host in the internal network is different, so the weight will be different, as shown in Table 1. The internal network is divided into the core network and the office network.

V_s represents its own probability, which indicates the probability of being used vulnerability. According to the CVSS metric method of individual vulnerabilities: Access Vector (AV), Authentication (AU), and Access Complexity (AC) [9] are shown in Table 2.

The formula is

$$V_s = AV + AU + AC. \quad (2)$$

(b) The attack action success probability is P_{AS} , which could assess the success probability for a vulnerability that the attackers use it to attack. The influencing factors include the information of vulnerability (K), the method of atomic

TABLE 2: The CVSS metric method of individual vulnerabilities.

Name	Degree of difficulty	Indicator value	Description
AC	High	1	The vulnerability is very difficult to use
	Medium	2	The vulnerability is a little difficult to use
	Low	3	The vulnerability is easy to use
AU	Multiple	1	The vulnerability is difficult to use
	Single	2	Vulnerability is a little difficult to use
	None	3	The vulnerability is easy to use
AV	Local	1	The vulnerability only can be used locally and difficulty
	Adjacent network	2	The vulnerability can be used and is harder to use
	Network	3	The vulnerability can be exploited remotely and easily

attack (M), and whether corresponding attack tool is used (N). Based on the relevant research papers, with reference to the calculation method of the success rate of independent vulnerability from CVSS and Wu et al. [12], consider the following.

The formula is

$$P_{AS} = K + M + N, \quad (3)$$

where K is in the range of $\{0, 0.1\}$, indicating whether the vulnerability information is published. The vulnerability has been issued; K value is 0.1; otherwise, the value is 0.

Here, M is in the range of $\{0, 0.2, 0.4\}$, indicating whether the atomic attack method or step is currently available. If the vulnerability has a detailed attack step scheme, then the value of M is 0.4. If there is a simple attack scheme, then M is 0.2. Otherwise, M is 0.

N is in the range of $\{0, 0.2, 0.4\}$, indicating whether the attack tools are required in the atomic attack. If the vulnerability is not required to use the attack tools, N is 0.4. If the vulnerability exploited needs to use the attack tools and the corresponding attack tools are available, N is 0.2; and if you need to use the attack tools, but there are no available attack tools, the value of N is 0.

(3) *Calculating the Cumulative Reachable Probability of Attack Action Node P_{AC} .* P_{AC} means the cumulative reachable probability of attack action except for the initial node. Due to the complexity of the internal network, we will discuss the classification of the key-value pairs with the directional edges.

(a) The cumulative occurrence probability of common key-value pairs: normally, there is only one directed edge of an action node; that is, after obtaining the resources of one resource node, you can use the vulnerability to attack.

Therefore, when an atomic attack is initiated, the attack probability of all the front nodes is evaluated, and the value

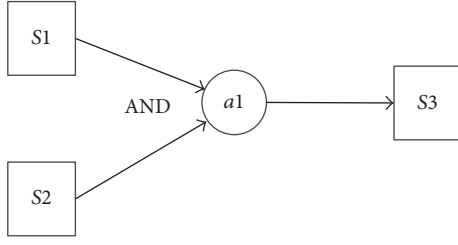


FIGURE 4: The “AND” relationship between two resource state nodes.

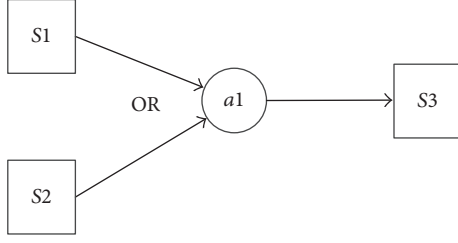


FIGURE 5: The “OR” relationship between two resource state nodes.

is the cumulative reachable probability of the first resource state node in the key-value pair. Then, we can calculate the cumulative reachable probability of current action node.

The formula is

$$P_{AC} = P_S \times P_A. \quad (4)$$

P_S represents the cumulative reachable probability of the first resource state node in a key-value pair and P_A represents the probability of the attack action carrying out the attack.

(b) The directed edges to the same attack action with “AND” relationship.

The form is shown in Figure 4.

The formula is

$$P_{AC} = P_{Si} \times P_{Sj} \times P_A. \quad (5)$$

P_{Si} , P_{Sj} represent the cumulative reachable probability of two resource state nodes pointing to the same attack action.

(c) The directed edges to the same attack action with “OR” relationship.

The form is shown in Figure 5.

The formula is

$$P_{AC} = (P_{Si} \oplus P_{Sj}) \times P_A. \quad (6)$$

(4) Calculating the Cumulative Reachable Probability P_s for the Target Node. (a) The cumulative reachable probability of common resource state node. In the usual case, there are only one directed edge points to the resource state node. That means to obtain another resource state node only one vulnerability is needed.

The formula is

$$P_{Si} = P_{AC} \times P_{AS}. \quad (7)$$

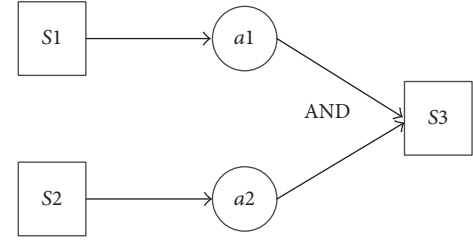


FIGURE 6: The “AND” relationship between two attack action nodes.

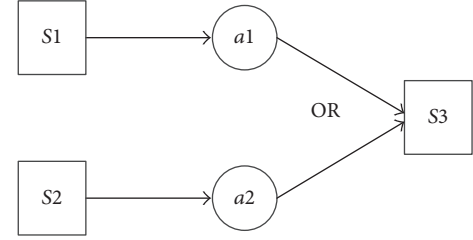


FIGURE 7: The “OR” relationship between two attack action nodes.

(b) The directed edges to the same resource state node with “AND” relationship, as shown in Figure 6.

The formula is

$$P_{Si} = (P_{AC1} \times P_{AS1}) \times (P_{AC2} \times P_{AS2}). \quad (8)$$

(c) The directed edges to the same resource state node with “OR” relationship.

The concrete form is shown in Figure 7.

The formula is

$$P_{Si} = (P_{AC1} \times P_{AS1}) \oplus (P_{AC2} \times P_{AS2}). \quad (9)$$

The complexity of the method of attack probability calculation is decided by the number of atomic attack nodes. According to attack graph generation algorithm [10], we can get a certain attack graph, so it is countable but is not predictable.

4. Experiment and Analysis

4.1. Experiment Environment. To verify the method proposed in this paper, we build a representative virtual simulation environment of an internal network which comprised hosts with VMware and network devices with GNS3. The key network topology is shown in Figure 8.

The firewall isolates the network structure into two parts, the external network and the internal network. The hosts in office network are as follows: Host 0 is an ordinary computer with Windows system for office users; Host 1 is the DNS server, which provides DNS services for all internal network hosts; Host 2 is the Web server and provides HTTP service for all internal network hosts. The hosts in the core network are as follows: Host 3 and Host 4 provide SSH services, with Linux system; Host 5 is the FTP server; Host 6 is the database SQL server.

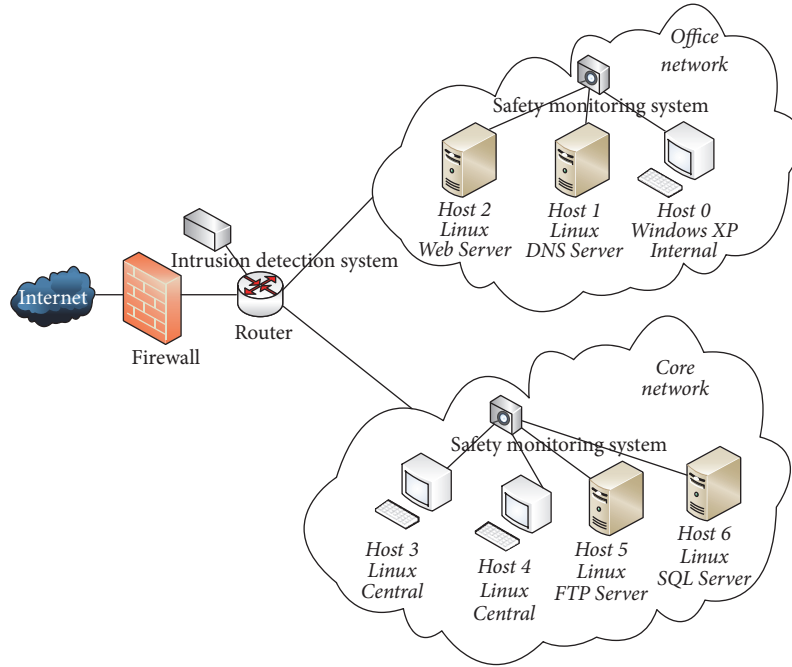


FIGURE 8: The network topology diagram.

TABLE 3: The vulnerability information for each terminal.

Host	Name of software	Vulnerability description	CVE ID
H1	BIND 9	Stack buffer overflow vulnerability	CVE-2015-7547
H2	IIS 7.0	IIS Buffer Overflow Vulnerability	CVE-2008-0075
H3	OPENSSEH (SSH2)	Mode information disclosure vulnerability	CVE-2008-5161
		Local privilege elevation vulnerability	CVE-2007-2063
H4	OPENSSEH (SSH2)	Mode information disclosure vulnerability	CVE-2008-5161
		Local privilege elevation vulnerability	CVE-2007-2063
H5	Ser-U 10.5.0.19	Read the vulnerability	CVE-2015-7601
		FTP buffer overflow vulnerability	CVE-2015-7768
H6	SQLServer 2005	Buffer Overflow Vulnerability	CVE-2008-0086
		Information disclosure and buffer overflow vulnerability	CVE-2008-0106

The security monitoring system is deployed in the two subnetworks to monitor the hosts; then we could obtain the appropriate attacking alarm log. The specific security monitoring systems include the OSSEC intrusion detection system, which monitors the abnormal activities of the host PC and the Trojan horse and Tripwire security monitoring tools deployed on the file server for system integrity check.

The attacker originally owns the resources as normal staff user on Host 0 in the office network. The final target is to get the root privilege of Host 5 or Host 6 in the core network. At the first time, the corresponding security policies are as follows: (1) the office network Web server Host 2 and DNS server Host 1 provide internal network service for internal users; all internal hosts can connect to the office network by accessing the services on Host 1 and Host 2. (2) Host 4 in the

office network is allowed to access the SQL services on Host 6 for specific data but cannot browse all information and could not modify or download; (3) Host 3 and Host 4 in the core network are allowed to access the rest of the terminals and the corresponding services and own the permissions to modify specific data.

We scanned the vulnerabilities on each host with Nessus. The vulnerability scanned results and related information of each host are shown in Table 3.

The security monitoring system was deployed to monitor the hosts; we could obtain the appropriate attacking alarm log. The specific security monitoring systems included the OSSEC intrusion detection system, which monitored the abnormal activities of hosts and the Trojan horse, and Tripwire security monitoring tools are deployed on FTP server and SQL server for system integrity check.

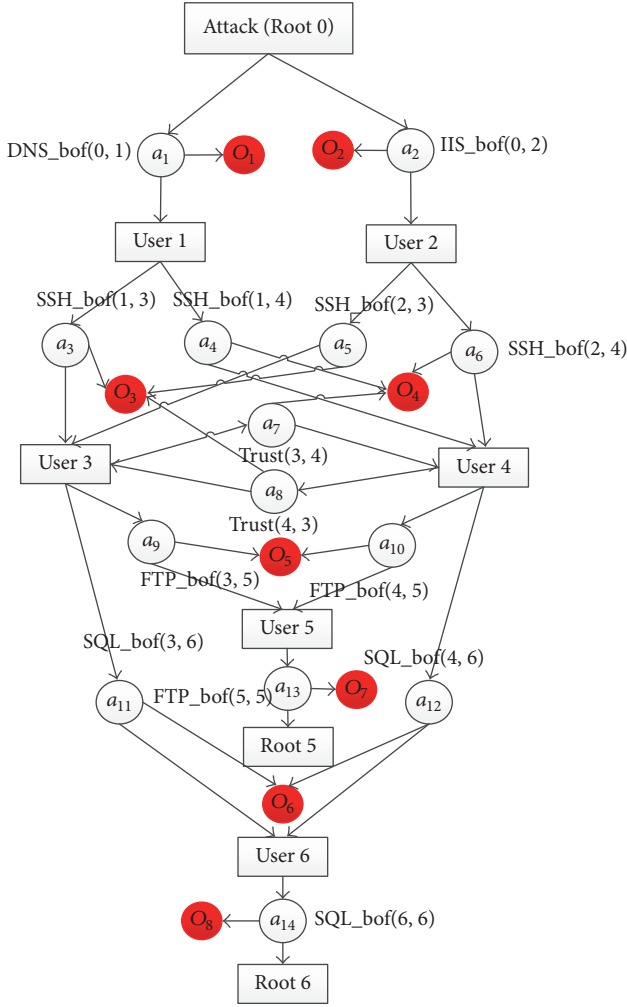


FIGURE 9: The initial attack graph.

4.2. Experiment Verification. After determining the topology of the internal network and the information of each terminal, attack graph automatic generation algorithm would help us to generate the attack graph of the internal network, and the atomic attack edges pointing to the same resource state node all present “OR” relationship. The concrete structure is shown in Figure 9.

In Figure 9, the rectangle represents the resource state node and the resource rights that can be obtained after each atomic attack; the hollow circle represents the atomic attack action node and marked the corresponding terminal vulnerability on the left or right side of the atomic attack; the red solid circle is the monitoring event node.

According to the method of attack graph simplification, we got the simplified attack graph in Figure 10. We can simplify the attack graph with the temporal difference relationship of monitoring event node, removing the attack paths that do not match the temporal difference relationship.

After simplifying the attack graph, we can divide the simplified attack graph into key-value pairs and calculate the cumulative reachable probability for target nodes. The key-value pairs and attack probability are shown in Table 4.

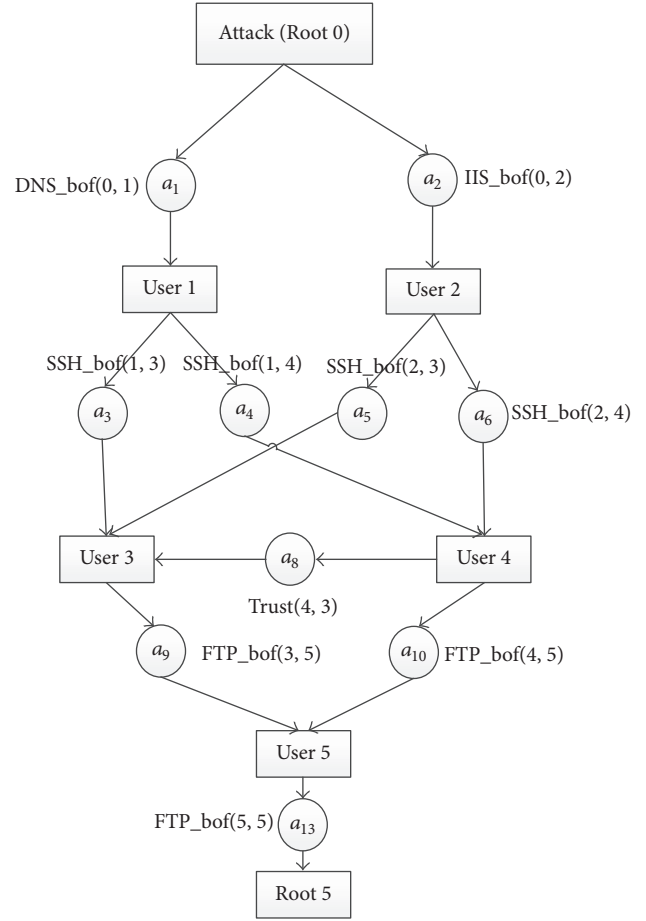


FIGURE 10: The simplified attack graph.

TABLE 4: The key-value pairs with attack probability.

Key-value pair	Attack probability $((P_A, P_{AS}))$
(Root 0, a_1 , User 1)	(0.83, 0.7)
(Root 0, a_2 , User 2)	(0.92, 0.5)
(User 1, a_3 , User 3)	(0.67, 0.3)
(User 1, a_4 , User 4)	(0.67, 0.3)
(User 2, a_5 , User 3)	(0.67, 0.3)
(User 2, a_6 , User 4)	(0.67, 0.3)
(User 4, a_8 , User 3)	(0.58, 0.5)
(User 3, a_9 , User 5)	(0.83, 0.7)
(User 4, a_{10} , User 5)	(0.83, 0.7)
(User 5, a_{13} , Root 5)	(0.83, 0.7)

The cumulative reachable probability values for subresource state nodes and target node are shown in Table 5.

The cumulative reachable probability from H0 to H5 is 3.95% by using the other terminal's vulnerabilities. Compared with the frequency of internal attacks and the safety reports of the enterprise system from daily inspection, the two values are basically in line. Based on such an internal network, where the attacker only has privilege in the office network, the

TABLE 5: The cumulative reachable probability for target nodes.

Subresource state node	Cumulative reachable probability
User 1	0.581
User 2	0.46
User 3	0.117
User 4	0.117
User 5	0.068
Root 5	0.0395

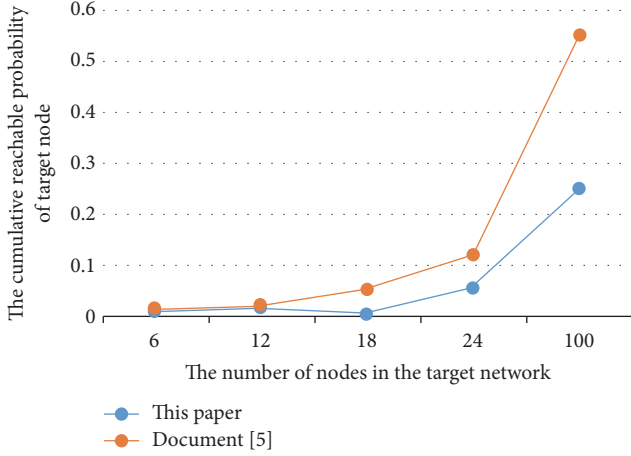


FIGURE 11: The comparison of cumulative reachable probability.

probability of stealing the core data successfully is relatively low.

4.3. Experiment Analysis. In order to assess the accuracy of our method, we performed five times' experiments with different cumulative reachable calculation method of target node on the same internal network. We compared the data based on the attack probability of the attack graph proposed by Li et al. [5] with the data obtained by the method proposed in our paper. The results are shown in Figure 11, the abscissa indicates the number of target network nodes and the ordinate indicates the cumulative reachable probability of target node.

Comparing the data in Figure 11, the cumulative reachable probability of target node changed more smooth and more stable with our method than Li et al. [5] that our security metric should be more accurate than Li et al.'s [5].

At the same time, the approach we proposed could prune the attack graph paths and not only reduce the calculation greatly but also improve the accuracy of the attack graph obviously. Therefore, the computational effort is absolutely less than Document [5], as shown in Figure 12.

5. Conclusion

In our paper, we propose an internal network security metric method based on attack probability to solve the problem of the existing security metrics based on attack graph lacking the

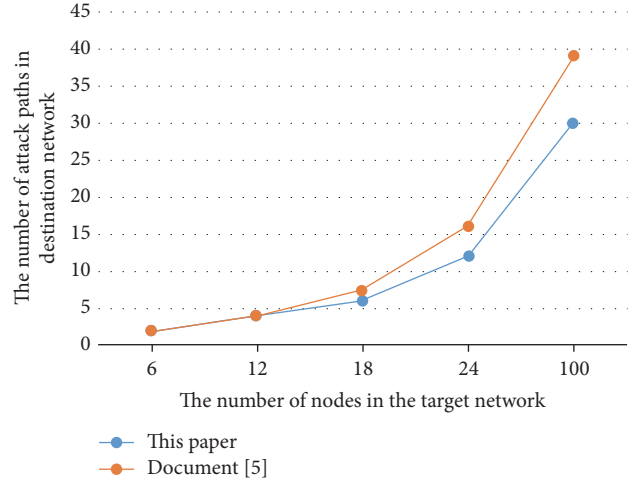


FIGURE 12: The comparison of attack paths.

applicability of the internal network. We use the monitoring event node and the temporal difference relationship to simplify the attack graph, put forward the concept of the key-value pair to analyze the attack graph, and propose the calculation method of cumulative reachable probability for different kind of target nodes based on vulnerabilities with CVSS metric indicators and the directed edges relationship. The simulation results show that the method of attack graph simplification has a significant improvement in efficiency, and the method of attack probability calculation improves the quantitative analysis accuracy obviously. The next step of the work will focus on the refinement attack probability calculation, finding a more comprehensive internal network to improve the accuracy of the final probability value.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This article is supported by National Key R&D Program of China (Grant no. 2016YFB0800700) and National Natural Science Foundation of China (Grant no. U1636115).

References

- [1] G. Yang, J. Ma, and A. Yu, "Study on internal threat detection," *Journal of Information Security*, vol. 1, no. 3, 2016.
- [2] X. Lu, "Research on information system security metrics theory and method," *Computer Science*, vol. 35, no. 11, pp. 42–44, 2008.
- [3] P. Xie, J. H. Li, X. Ou et al., "Using Bayesian networks for cyber security analysis," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks*, vol. 23, pp. 211–220, 2010.
- [4] L. Wang, T. Islam, T. Long et al., "An Attack Graph-Based Probabilistic Security Metrics," in *Proceedings of the Conference on Data & Applications Security XXII*, 5094, pp. 283–296, 2008.

- [5] Q. Li, B. Wang, X. Wang et al., "Network security measurement method based on probability of attack graph node," *Application Research of Computers*, vol. 30, no. 3, pp. 906–908, 2013.
- [6] H. Wang, G. Yang, and D. Han, "Study on internal threat prediction based on bayesian networks," *Application Research of Computers*, vol. 30, no. 9, pp. 2767–2771, 2013.
- [7] S. Zhang, G. Li, S. Song et al., "Application of Bayesian Reasoning in the Confidence Calculation of Attack Graph Node," *Journal of Software*, vol. 21, no. 9, pp. 2376–2386, 2010.
- [8] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012.
- [9] X. Chen, B. Fang, Q. Tan et al., "Study on inference algorithm of internal attack intention based on probability attack graph," *Journal of Computers*, vol. 37, no. 1, pp. 62–72, 2014.
- [10] Y. Ye, X.-S. Xu, Y. Jia, and Z.-C. Qi, "An attack graph-based probabilistic computing approach of network security," *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 33, no. 10, pp. 1987–1996, 2010.
- [11] X. Ou, S. Govindavajhala, and A. W. Appel, "MULVAL: a logic-based network security analyzer," in *Proceedings of the 14th Usenix Security Symposium*, pp. 113–117, Baltimore, MD, USA, 2005.
- [12] D. Wu, D.-G. Feng, Y.-F. Lian, and K. Chen, "Efficiency evaluation model of system security measures in the given vulnerabilities set," *Journal of Software*, vol. 23, no. 7, pp. 1880–1898, 2012.

Research Article

A Dynamic Hidden Forwarding Path Planning Method Based on Improved Q-Learning in SDN Environments

Yun Chen , Kun Lv , and Changzhen Hu 

School of Software, Beijing Institute of Technology, Beijing, China

Correspondence should be addressed to Kun Lv; kunlv@bit.edu.cn

Received 10 January 2018; Accepted 12 March 2018; Published 23 April 2018

Academic Editor: Zheng Yan

Copyright © 2018 Yun Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Currently, many methods are available to improve the target network's security. The vast majority of them cannot obtain an optimal attack path and interdict it dynamically and conveniently. Almost all defense strategies aim to repair known vulnerabilities or limit services in target network to improve security of network. These methods cannot response to the attacks in real-time because sometimes they need to wait for manufacturers releasing corresponding countermeasures to repair vulnerabilities. In this paper, we propose an improved Q-learning algorithm to plan an optimal attack path directly and automatically. Based on this path, we use software-defined network (SDN) to adjust routing paths and create hidden forwarding paths dynamically to filter vicious attack requests. Compared to other machine learning algorithms, Q-learning only needs to input the target state to its agents, which can avoid early complex training process. We improve Q-learning algorithm in two aspects. First, a reward function based on the weights of hosts and attack success rates of vulnerabilities is proposed, which can adapt to different network topologies precisely. Second, we remove the actions and merge them into every state that reduces complexity from $O(N^3)$ to $O(N^2)$. In experiments, after deploying hidden forwarding paths, the security of target network is boosted significantly without having to repair network vulnerabilities immediately.

1. Introduction

A defense strategy represents a series of defense methods in the target information system network that can reduce the attack success rate of attackers. Currently, many methods are available to generate a defense strategy. The most important problem is the game between cost and performance. The defense strategy may own excellent performance, but defenders scan and recapture the information system in most instances, which is very uneconomic.

Generally speaking, whether it is SDN or traditional network, we can plan defense strategy through locate optimal attack path. Regarding this method, a majority of previous papers specify generating a complete attack graph [1–3]; however, in a very large computer cluster, the state explosion problem tends to affect the attack graph generation. Thus, the optimal attack path cannot be modeled quickly, and in extreme cases, it may not be possible to determine the optimal attack path. In [4], authors use the ant colony optimization (ACO) approach to search the optimal attack path based

on the minimal attack path [5], but ACO can easily fall into a local optimum. Reference [6] proposes a HMM-based attack graph generation method, and then authors use ACO-based algorithm to compute the optimal attack path. Based on this path, evaluating the security of target network can be evaluated and corresponding countermeasures can be planned. But this method primarily handles the known vulnerabilities. Reference [7] proposes a malicious nodes-based security model enacting method, but its performance on handling zero-day vulnerability is not strong enough.

Reinforcement learning [8] is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize a cumulative reward. It differs from standard supervised learning in that correct input/output pairs are never presented, nor are suboptimal actions explicitly corrected. Further, the focus is on online performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

To determine a method that can model the optimal defense strategy in many conditions, the algorithm of the method must not depend on all states of the target network and it should be able to decide which atomic attack should be picked to be the next state dynamically. These abilities depend on the characteristics of the specific algorithm.

In this paper, the optimal attack path between source node and target node is computed by the improved Q-learning algorithm. Concretely, in the first stage, we collect information of known vulnerabilities and corresponding type of hosts from national vulnerability database (NVD) [9]. Then, a fuzzy neural network will be used to train these samples to gain the host weight. After getting host weight, the reward function of improved Q-learning can be built. Using this function, the optimal attack path can be located between source node and target node.

In Section 2, we will introduce Q-learning, which is followed by an overview of the main contributions of this paper. In Section 3, the definition of a network model will be discussed. We propose a reward function, and the optimal forwarding path will be built based on this function. In Section 4, we will discuss how to implement this method in a real information system network.

Section 5 provides the experimental results for the optimal protective path and discusses how to improve the Q-learning algorithm. The paper concludes with a summary and future work in Section 6.

1.1. Related Work. There are several works in defense strategy planning in recent years. Currently, many methods are available to generate a defense strategy and these methods can be classified into three categories. First, we can compute the optimal attack path using attack graph and enact policies to destroy the path. Second, we can locate the not-trust nodes in target network and plan countermeasures to prevent these nodes from being exploited by attackers. Third, special strategies can be designed to aim to specific network environments specific attack types. But all of these methods have inherent defects.

Regarding the first method, the intrinsic of this method is that system generates attack graph of target network and then finds and interdicts the optimal attack path in the attack graph. Wang et al. [6] propose a HMM-based attack graph generation method and ACO-based algorithm to evaluate the security of target network and plan corresponding countermeasures. This method can compute the transition probability between each two states. Based on the probability and ACO-based algorithm, the shortest attack path can be found, which can be used to evaluate security metrics. But this method owns some defects. Firstly, the complexity of ACO algorithm is $O(Nc * n^2 * m)$ (Nc is the iteration number, n is the number of vertices, and m is the number of ants in ant colony) that is too high to computation if we process computer cluster. Secondly, this technique's performance is not good enough when it deals with APT and zero-day vulnerabilities, because the interval of time series of HMM is slight less than the interval of APT and this method uses Common Vulnerability and Exposures (CVE) [10]. Ghosh et al. [4] proposes an ACO-based defense strategy planning

method. This method is similar as [6]. It uses minimal attack graph to locate the optimal attack path, but this path may not be the global optimal and this attack graph will also show state explosion issue if it is used in very large computer clusters.

For the second technique, the core of this technique is to find the malicious nodes. Akbar et al. [7] propose a Support Vector Machine (SVM) and rough set-based security model building method. In that paper, authors use SVM and rough set to classify the nodes in target network as trust nodes, strange nodes, and malicious nodes. This technique can also acquire the transaction success rate. This method can handle zero-day vulnerabilities in some conditions, but it needs a large number of sample data to training SVM that is impossible to obtain enough data set in some network environments because the data need to spend a lot of time to collect.

Regarding the third method, the key of this method is to handle specific attack types or vulnerabilities. Hu et al. [11] characterize the interaction between defender and APT attacker and an information-trading game among insiders as a two-layer game model. Through their analysis, the existence of Nash Equilibrium for both games is certified and the security metric can be evaluated. But this method can only process APT; the generalization of it is limited. Same as [11], Wang et al. [12] propose a k -zero-day safety method. It starts with the worst case assumption that this is not measurable which unknown vulnerabilities are more likely to exist in the target network and ends to the number of zero-day vulnerabilities that can destroy the network asset. But the complexity of computing this metric is exponential in the size of the zero-day attack graph. Furthermore, the zero-day attack graph cannot reflect the condition of known vulnerabilities related work.

1.2. Contribution. In this paper, we use an improved Q-learning algorithm to generate the optimal attack path. In Q-learning [8], which action will be selected is based on a reward function. In other words, a large number of sample data are not required, as is the case in many other machine learning algorithms. Compared to temporal difference learning, Q-learning can directly iterate an optimal policy, which in this paper is the optimal attack path. Defining the reward function is the key issue in Q-learning. In this paper, we use the host weight and attack success rate of atomic attacks to build a reward function. Specifically, the host weight is decided by the position the host stays in and services the host offers. Besides, we improve the structure of state matrix in Q-learning. The dimension of the matrix is reduced, which can lower the space complexity. Furthermore, the network model that reflects the configuration of the target network will be used to analyze the result of Q-learning.

Our ultimate goal is to build a hidden forwarding path. In this path, we create virtual hosts that provide specific defense strategies in SDN to filter specific attacks. These hosts can be created or deleted dynamically, which can ensure the computation of hidden forwarding paths will occupy the SDN controller's minimal memory space when we want to change the routing path. Furthermore, through using the hidden forwarding paths, vulnerabilities are filtered,

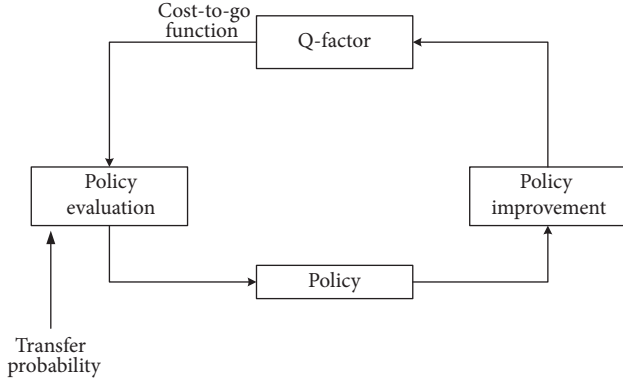


FIGURE 1: Illustration of policy iteration in reinforcement learning.

which can guarantee the system's security without repairing vulnerabilities or limiting services on hosts.

The other contribution of this paper is to render the defense strategy to be economical. Our method does not need scan or monitor hosts at all time. The hosts will be scanned only if our algorithm thinks it is not-trust node.

2. Preliminary

2.1. Description of Q-Learning Algorithm. Q-learning is a model-free reinforcement learning technique and it derives from policy iteration. The flow diagram of policy iteration is shown in Figure 1. Specifically, Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP). Therefore, the definition of Q-learning is given in

$$\begin{aligned} g_n &= (i_n, a_n, j_n) \\ s_n &= (i_n, a_n, j_n, g_n). \end{aligned} \quad (1)$$

This formula denotes that the state i_n transfers to $j_n = i_{n+1}$ using action a_n , and the cost of this process is g_n . Above, n represents the discrete time sequence.

In Section 2.1, Q-factor is considered as the sum of the immediate cost and all of the successor states' discount costs which are followed by the current state in policy μ .

However, different from policy iteration, Q-learning is an incremental dynamic programming process, and it is very suitable to solve MDP which does not have an apparent transfer probability. Furthermore, Q-learning works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. A policy is a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. Thus, the update formula of Q-factor is given in

$$\begin{aligned} J_n(j) &= \max_{b \in A} Q_n(j, b) \\ Q_{n+1}(i, a) &= (1 - \delta_n(i, a)) Q_n(i, a) \\ &\quad + \delta_n(i, a) [g(i, a, j) + \gamma J_n(j)]. \end{aligned} \quad (2)$$

j is the successor state and $\delta_n(i, a)$ is the learning rate of state-action (i, a) in time n . A is the action set. γ is the discount factor. Generally, in order to ensure the algorithm converges to the optimal value, the learning rate can be set based on

$$\delta_n = \frac{\alpha}{(\beta + n)}, \quad n = 1, 2, 3, \dots \quad (3)$$

In this formula, α, β are positive numbers. After we evaluated a series of values of α and β , we found that, if α/β is approximately 0.1, the convergence speed and the accuracy of the Q-learning agent are suitable.

But the traditional Q-learning algorithm owns some problems. Firstly, the Q-learning agent should choose an action when a state transfers to other state, but in the information system, we can fuse the action into state. Secondly, the traditional Q-learning may generate redundant terms, although the total reward of this path is the highest one. In Section 3, we will discuss and solve these problems.

2.2. Software-Defined Network. Software-defined networking (SDN) is an approach to computer networking that allows network administrators to programmatically initialize, control, change, and manage network behavior dynamically via open interfaces [13] and abstraction of lower-level functionality. SDN is meant to address the fact that the static architecture of traditional networks does not support the dynamic, scalable computing and storage needs of more modern computing environments such as data centers. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (the SDN controller or control plane) from the underlying systems that forward traffic to the selected destination (the data plane). The SDN structure is shown in Figure 2.

3. Building the Optimal Attack Path

The optimal attack path is the most cost-effective attack path from source node to target node in an information system network. Using the optimal attack path, the attacker can achieve his/her goal at minimum cost. In this paper, we use the optimal attack path to build the hidden forwarding path.

3.1. Network Model. A network model reflects host information in the target network, including software applications, host name, host's IP address and operating system, and communication link. Using a network model, we can produce a network topology graph and installed software applications configuration. In this paper, our network model refers to [14], but we have improved this model to suit our system. The structure of our network model is shown in Figure 3.

3.2. Evaluating Weights of the Hosts. In reinforcement learning, the reward function is the core facilitator. If and only if we have a reward function, the policy iteration or value iteration can be executed. Currently, two methods are available to obtain the reward function.

The first is to obtain the state set and the action set and determine the relationship between them. A reward

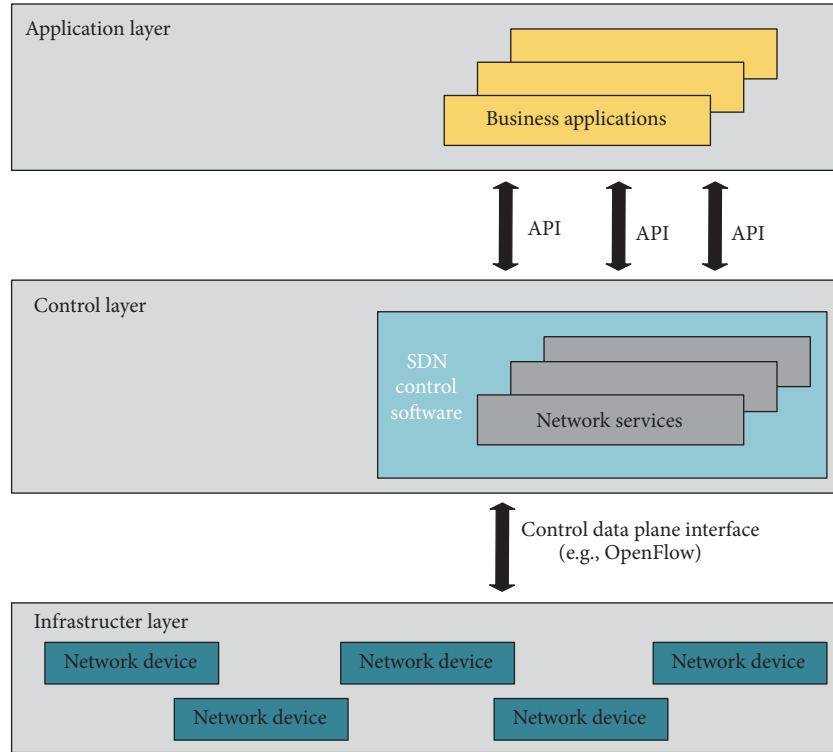


FIGURE 2: Model of software-defined network.

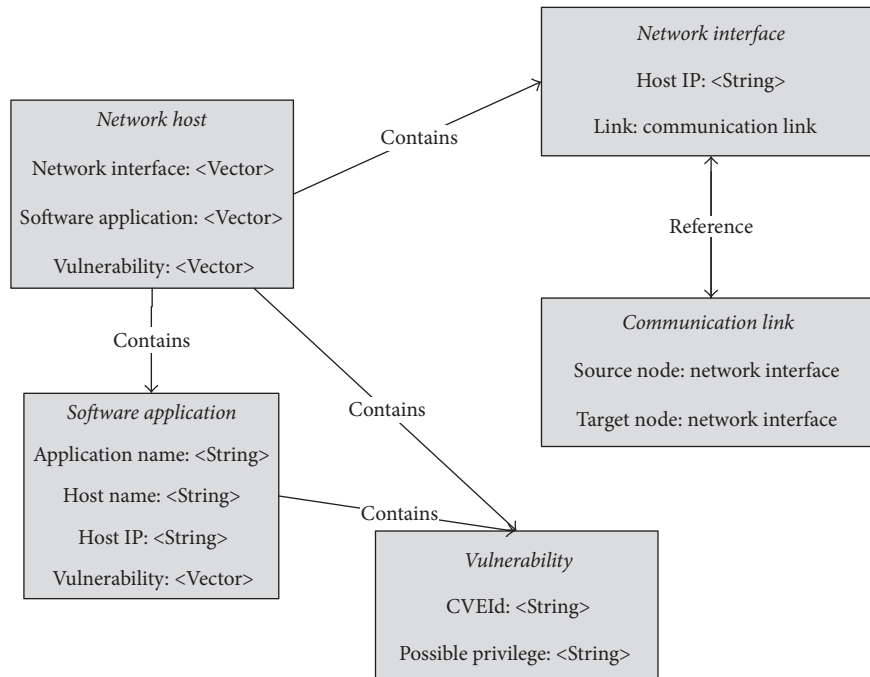


FIGURE 3: Network model.

function can then be fitted, which is named the state-action function or Q-factor. Furthermore, about several decades ago, another method that can also obtain the reward function was proposed, named inverse reinforcement learning

[15, 16]. But to use the inverse reinforcement learning, the optimal policy should be specified beforehand. Thus, in this paper, we use the first method to obtain the reward function.

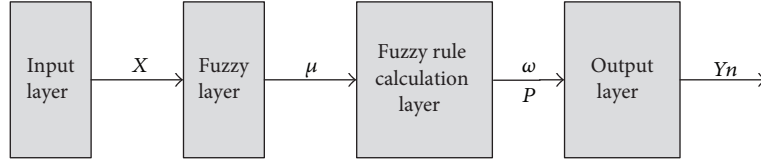


FIGURE 4: T-S fuzzy neural network structure.

Definition 1. Attack success rate of single weakness represents the rate at which a weakness in the information system is successfully exploited by attackers and the degree of difficulty in carrying out the attack. This element is denoted as Pr .

The attack success rate of a single weakness can be affected by a large number of factors, for instance, available information on the weakness, the attack method, the attack tool, and whether there is a communication link between the source host and target node. The level of detail known about a weakness will affect the attack success rate. The value of the success rate of a single weakness attack can be found from the Common Vulnerability Scoring System (CVSS) [17–19].

Definition 2. Host weight denotes a host's weight in the target network. It is the function defined by

$$Hw = f(\text{pos}, \text{sev}). \quad (4)$$

In this equation, Hw is the host weight, pos represents the host's position (Internet, DMZ, Intranet), and sev denotes which services the host offers (MAIL, WEB, DNS, SQL, FTP, Management, etc.). Therefore, this equation indicates that host weight is decided by the host's position and the services it provides. Unfortunately, all services and position are not known, nor is the relationship between the position and a service. However, we can estimate the importance of these attributes based on basic knowledge of computer networks. For instance, the importance of different positions in an information system is Intranet > DMZ > Internet, and for the services, Management > SQL > FTP > DHCP > MAIL > DNS > WEB. Furthermore, we know that pos and sev are positively correlated. Thus, we can build a fuzzy set based on this knowledge and use fuzzy computation to determine the value of the host weight. In this paper, we use a T-S fuzzy neural network (FNN) to fit the weight of host. The structure of the neural network is shown in Figure 4.

In this figure, X_i is an input vector. In this paper, in order to simplify the calculation and fit our network, it is denoted as a ten-element tuple $\langle \text{Pos}_i, \text{Pos}_j, \text{Pos}_k, \text{Sev}_1, \text{Sev}_2, \text{Sev}_3, \text{Sev}_4, \text{Sev}_5, \text{Sev}_6, \text{Sev}_7 \rangle$. pos and sev are the type of position and service, respectively. μ represents the membership function, ω is the result of the fuzzy computation, and P denotes the coefficient of the neural network. Yn is the neural network's output and is a single-element tuple $\langle \text{Weight} \rangle$. Weight denotes the host weight. Ye is the expectation output. It represents the host weight. At first, we just categorize all hosts into five different degrees based on the impact of their confidentiality, integrity, and availability on the target network, *Danger*, *High*, *Medium*, *Low*, and *Very*

Low, but these only reflect the weight of a host roughly. In the next step, we use our algorithm to fit more accurate results.

In this paper, the proposed host weight is computed based on a fuzzy self-adaption weight correction algorithm. In the first stage of the algorithm, we quantify the sample data. Because accurate results will be obtained by the fuzzy computation, we just need an initial value that reflects the importance of different attributes. We then normalize the sample and initialize the T-S fuzzy neural network, including the number of neurons in every layer, the initial learning rate and P , and the evolution times of the network. After training the network, for every input's attribute, $\hat{\mu}$, a coefficient vector of the parameters in (4), which is defined in (5), can be obtained using the algorithm

$$\hat{\mu} = \sum_j \mu_j, \quad j = 1, 2, \dots, k. \quad (5)$$

In this equation, k is the number of inputs parameters. Thus, (4) can be redefined as

$$Hw = \hat{\mu} X_t. \quad (6)$$

X_t is the set of pos , sev in the target network. Therefore, the host weight can be solved from (6).

3.3. Modeling Optimal Attack Path Using Improved Q-Learning Algorithm. After we obtain the information on vulnerabilities and hosts in the target network, the process of Q-learning begins.

Definition 3. Optimal attack path is an attack path which has the highest probability to be chosen by intruders.

Definition 4. The not-trust nodes are the nodes in the optimal attack path.

After we acquire the state set and action set, if we can fit the reward function (cost function), the optimal attack path can be determined.

3.3.1. The Proposed Reward Function. The reward function (cost-to-go function) decides which action will be chosen in the current state. Using the reward function, we can obtain an optimal policy, which is the optimal attack path. Because a state node represents a host-vulnerability pair, the information of host and vulnerability will decide the reward function. In this paper we formulate the reward function based on host weight (Hw) and the attack success rate on

Require:

- (1) Host weight Hw
- (2) The attack success rate of vulnerability Pr
- (3) Vulnerability V
- (4) Host Name HN

Ensure: Optimal policy (attack path) π^*

```

(5) function IQL(Hw, Pr, V, HN)
(6)    $n \leftarrow \text{size}(V)$  obtain the number of vulnerabilities
(7)    $m \leftarrow \text{size}(HN)$  obtain the number of hosts
(8)    $S \leftarrow \text{getstate}(V, HN)$  gain state set
(9)    $\gamma \leftarrow \text{getNumber}()$  initialize discount factor
(10)   $\text{initial}Q = 0$  initialize value matrix
(11)   $\text{Reward} \leftarrow \mathbf{R}(S, HN, Hw, S \cdot V \cdot P_r)$  build reward matrix
(12)  for  $t = 1 : n$   $n$  is iteration step do  $Q_{t+1}(i_{t+1}) \leftarrow$ 
      funcQ( $Q_t(i_t), \delta_t, \gamma, R(i_t, i_{t+1}), J_t(i_{t+1})$ ) obtain the optimal policy
(13)    if  $Q_{t+1}(i) = Q_t(i)$  then break
(14)    end if
(15)  end for  $\pi^* \leftarrow i_0, i_1, \dots, i_n$ 
(16)  return  $\pi^*$ 
(17) end function

```

ALGORITHM 1: Planning optimal attack path based on improved Q-learning algorithm.

a single weakness (Pr). Therefore, the reward function is defined by

$$R = f(Hw, Pr). \quad (7)$$

From this equation, we know that the reward function is decided by Hw and Pr, but the relationship between the host weight and attack success rate of a single weakness is not known. Generally, there is a positive relationship between Hw and Pr, and they are interdependent. Thus, we use a multiplication sign to connect Hw and Pr. We should also consider the impact of a host on the information system network (see the intermediate item in (8)). For the relationship between attack success rate and host weight, we believe the host weight is less important than the attack success rate, because the calculation aims to vulnerabilities; thus, the square root of Pr is used in (8), which ensures the variation range of the value of attack success rate is bigger than host weights. The reward function is rewritten as

$$R = \sqrt{P_r} \times \frac{Hw_i}{\sum_j Hw_j} \times \overline{\sum_m T_m^i}. \quad (8)$$

In this equation, Hw_i is the weight of host H_i which owns a specific vulnerability. T_m^i denotes the m th privilege which can be captured on H_i based on a specific vulnerability. Hw_j is the weight of a host in the target network. c is the total number of hosts in the target network.

3.3.2. Improved Q-Learning Algorithm Structure. After obtaining the reward function, we determine g_n in (2) and calculate the Q-factor. In this paper, we use deterministic state transition model. Our contribution is that we merge actions into state set, because we actions and states are bounder together. Thus, the dimension of state transition matrix in Q-learning is reduced, that means the space complexity of

improved Q-learning is lower than traditional Q-learning algorithm. In addition, to avoid the attack circle, we set P_r to a small number if the target host is the source host; thus the reward matrix is recalculated based on the new P_r . Using this Q-factor, the optimal policy/attack path from source node to target node can be acquired. The improved Q-learning algorithm to identify the optimal attack path is shown in Algorithm 1.

In this algorithm, i_t represents a state in time n . In this paper, a state is a two-element tuple $\langle \text{HostName}, \text{Vulnerability} \rangle$. Because the states in the Q-learning system are host-vulnerability pairs, attaining the optimal policy means obtaining a set of host-vulnerability pairs that define the optimal attack path.

4. Building Dynamic Forwarding Paths in a SDN Environment

Nowadays, for an information system network, the most common defense strategy is to divide the target network into Intranet and DMZ. In the Intranet, some hosts are prohibited to visit other hosts or services, but the restrictions are very inconvenient for users. If the administrators find new vulnerabilities in a target network, some services or applications must be stopped to repair these vulnerabilities, which is uneconomic. Besides, if firewalls and/or IDS do not have a specific property, they cannot defend from specific attacks. To solve these problems, we propose the hidden forwarding path in this paper.

4.1. Building Hidden Forwarding Paths Using SDN. Using SDN, we can build forwarding nodes arbitrarily. In Figure 5(a), all hosts in the target network can connect with each other. In this case, the security of the target network cannot be very high because the forwarding

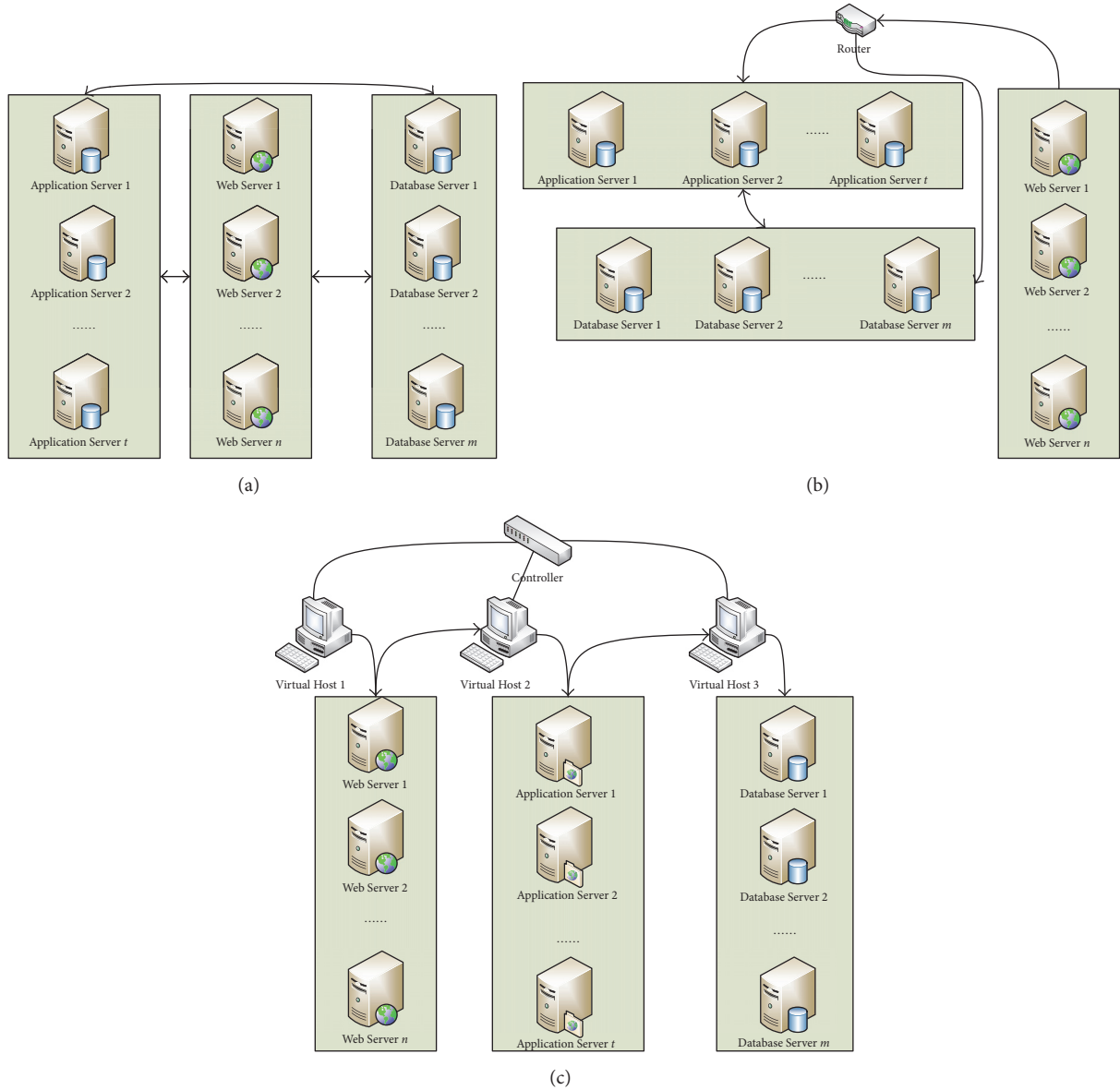


FIGURE 5: (a) An information system network's forwarding structure without any router. (b) The forwarding structure using traditional three-layer switching. (c) The forwarding path using SDN.

structure is too complex to monitor. To solve this issue, traditional networks use routers and switches to control the forwarding path (see Figure 5(b)). Generally, the core switch divides the information system network into two zones, DMZ and Intranet. If we change the forwarding path using the optimal attack path, the network structure should be altered. For example, if the web server visits application server, the application server responds to the requests of the web server by accessing data in the database server. If the application server is located in optimal attack path, the initial routing path should be changed or prohibited to protect system's security; for example, theoretically speaking, the web server can send packets to other servers, and these servers will visit database server. But this is impractical scheme, because in traditional network, part of network

topology is fixed. Thus, we cannot modify the forwarding path directly, meaning we cannot avoid the not-trust nodes without repairing vulnerabilities or limiting some services. In this paper, we use SDN to create the specific virtual hosts being used as forwarding nodes. In the first step, according to the not-trust nodes, we create virtual hosts that offer specific services, defense strategies, or interfaces to keep the original network structure in SDN environments. A sample of a hidden forwarding path is shown in Figure 5(c). In this network, the optimal attack path is

$$\text{WebServer} \rightarrow \text{ApplicationServer} \rightarrow \text{DatabaseServer}. \quad (9)$$

According to this attack path, we can add virtual hosts to be forwarding nodes between each two nodes and specific

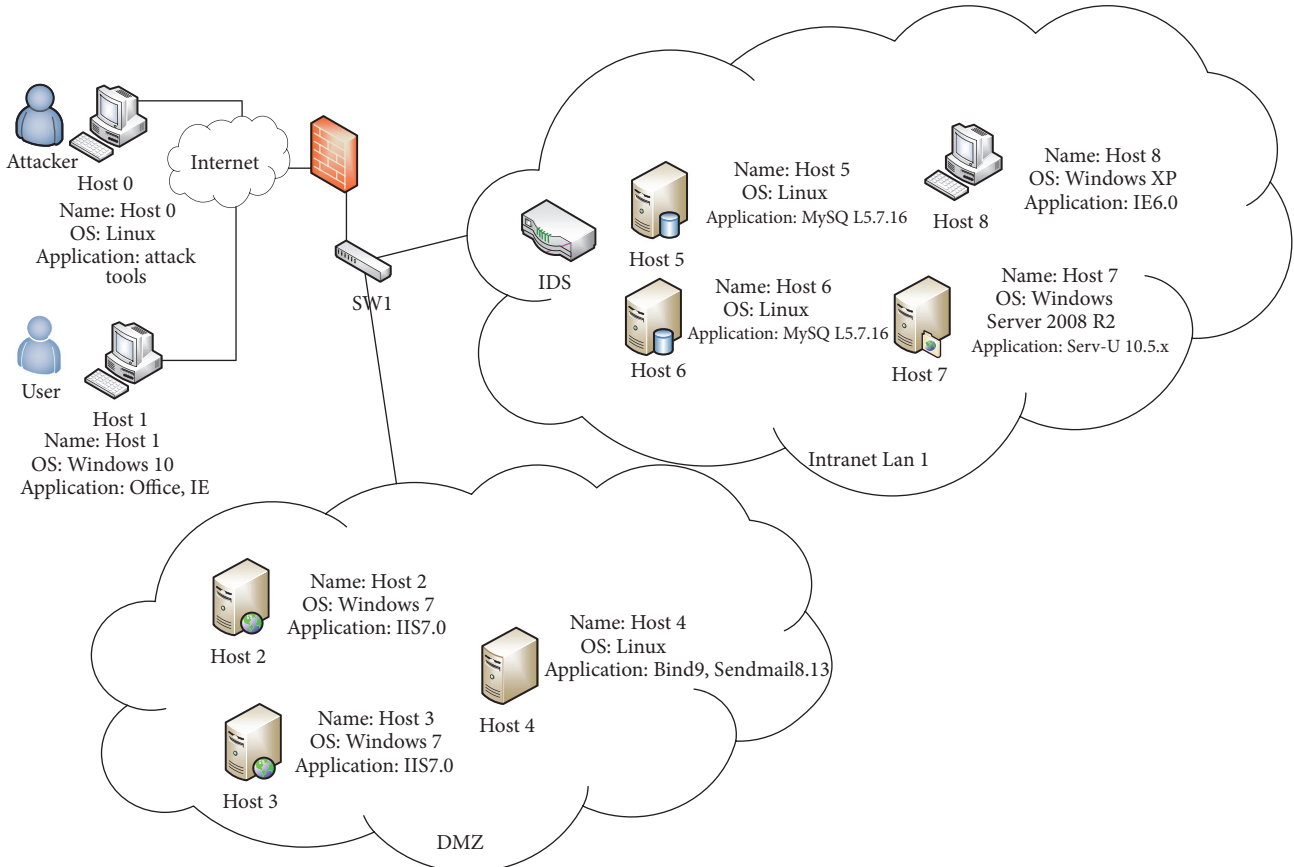


FIGURE 6: Network topology graph.

security strategies can be enacted on these virtual hosts, and the hidden forwarding path is shown as follows:

$$\begin{aligned} &(\text{VirtualHost1} \rightarrow) \text{WebServer} \rightarrow (\text{VirtualHost2} \rightarrow) \\ &\text{ApplicationServer} \rightarrow (\text{VirtualHost3} \rightarrow) \text{DatabaseServer} \end{aligned} \quad (10)$$

In virtual Host 1, we just analyze the packets that represent an HTTP request, and the function of Host 2 and Host 3 is similar to Host 1. Thus, every virtual host can defend against specific attacks, which is more effective and cost-effective than placing firewalls between every not-trust node. Concretely, in Figure 5(b), if the web server wants to communicate with application server, it will send the request packets to router and the router will send packets to application server through its routing table. But in Figure 5(c), if the web server sends packets, they will be sent to virtual Host 2 first. Then, virtual Host 2 will send packets application server. For the defense strategy in this optimal attack path, we should add filters that can filter attacks aimed at web, application, and database servers. We have two methods to add filters. First, we add web, ftp, and database filters into the first node (virtual Host 1). Second, because different servers are divided into different blocks in SDN environments, we can add specific filters on corresponding virtual hosts. In the first method, we must change the defense strategy if the optimal attack path changes. In the second method, we just need to add

or delete corresponding virtual hosts in SDN environments, which is much more convenient and efficient than the first method.

After implementing these specific functions on the virtual hosts, they form the hidden forwarding path. We can easily monitor the security logs of these virtual hosts. Furthermore, because the route is computed by the controller, we can change the hidden forwarding path dynamically and update the mapping table in the DNS server, which will not influence the physical components of the information system network. By using this method, administrators need not repair known and 0-day vulnerabilities in the target network immediately.

5. Experiment

In this experiment, we offer an example of building a hidden forwarding path. The network topology graph which is shown in Figure 6 is based on the network model mentioned in Section 2.1. Some security policies are implemented in this network: the firewall divides the information system network into three zones which represent Internet, DMZ, and Intranet, respectively. The web servers are configured in DMZ to offer web service to users. Hosts in the Intranet are not allowed to access the Internet. In addition, there is an intrusion detection system (IDS) in the Intranet to supervise

TABLE 1: Information of software and vulnerabilities on the terminals in the target network.

Host	Network segment	Service	CVE number	Success rate
H_0	Internet	Attack Tools	None	0
H_1	Internet	Office	None	0
H_2	DMZ	IIS7.0(HTTP)	CVE-2015-1635	0.9
H_3	DMZ	IIS7.0(HTTP)	CVE-2015-1635	0.9
H_4	DMZ	BIND9(DNS)	CVE-2015-5477	0.6
		Sendmail(Mail)	CVE-2009-4565	0.5
		OpenSSH 5.4(SSh)	CVE-2016-0778	0.3
H_5	Intranet	OpenSSH 5.4(SSh)	CVE-2016-0778	0.3
		MySQL 5.7.12(SQL)	CVE-2016-3521	0.6
			CVE-2016-3614	0.3
H_6	Intranet	MySQL 5.7.12(SQL)	CVE-2016-3521	0.6
			CVE-2016-3614	0.3
H_7	Intranet	Serv-U 10.5.0.19(FTP)	CVE-2011-4800	0.7
H_8	Intranet	Outlook(Mail)	CVE-2015-6172	0.4
		System	CVE-2003-0352	0.6

TABLE 2: Grade of pos and sev in host weight.

	Internet	DMZ	Intranet	Web	DNS	MAIL	DHCP	SQL	FTP	Manager
Initial value	1	2	3	1	2	3	4	5	6	7
Coefficient	0.279	0.362	0.363	0.345	0.378	0.337	0.355	0.375	0.399	0.384
Result	0.28	0.72	1.09	0.25	0.76	1.01	1.42	1.88	2.39	2.69

the entire target network. Also, Internet users can only access the IIS web services on Host 2 and Host 3 and the domain name service on Host 4. At the same time, Host 2 and Host 3 can access Host 4's Sendmail service and the SQL service on Host 5 and Host 6. Host 7 is a FTP server which can also be accessed by Host 2 and Host 3. But Host 2, Host 3, and Host 4 are prohibited from accessing Host 8 (Intranet management terminal) directly. Host 8 can access and download data from Hosts 2 to 7. The software applications and vulnerabilities on every terminal are shown in Table 1.

In this paper, we extract 3000 fuzzy information items from the hosts to build the sample data. Because we only need the initial value of the input's attributes (X in Section 3.2) to represent the importance of different attributes, if a host does not offer a service or is not installed in the position, the attribute's value is set to 0. In contrast, if the host offers a service or is installed in the position, the value of the attribute is set to a number based on the importance of the attribute. Initially, an attribute with least importance is set to 1, and other attributes' values will in turn increment at an interval of 1 based on the least important attribute's value. Thus, the initial values of sev and pos are shown in the first line in Table 2.

Based on this table, we obtain the value of input data. The neural network's output is the host weight. Because the expectation output (Y_e) only reflects the host weight roughly, it is only necessary for the network's output to display a similar trend to the expectation results. Thus, the result of testing instance prediction is shown in Figure 7.

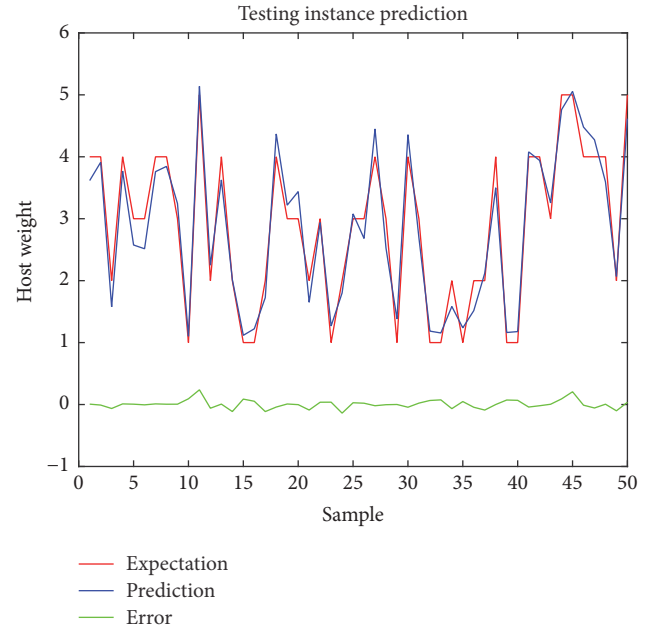


FIGURE 7: Results of training instance of host weight using FNN.

From Figure 7, the prediction of testing instance follows the same trend as the expectation result of testing instance. If we round off the prediction result (Figure 8), the prediction is equal to the expectation. Therefore, the parameters trained by the fuzzy neural network are assumed to be correct.

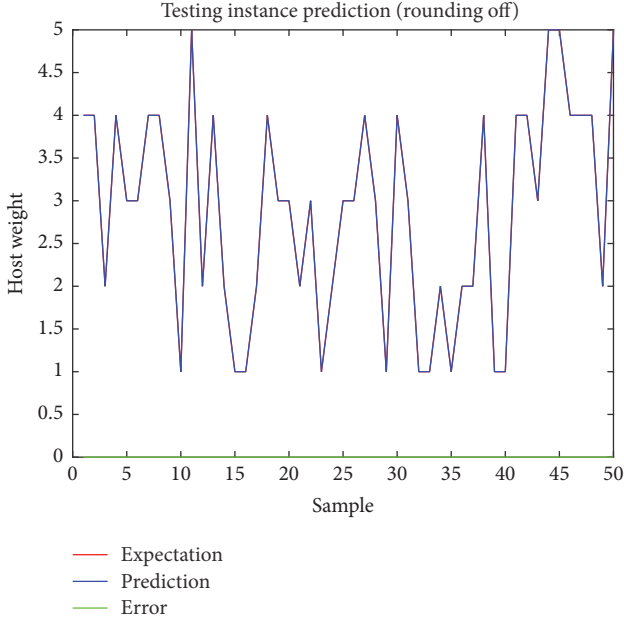


FIGURE 8: Results of training instance of host weight using FNN (round off).

The coefficients and the grading standards of pos and sev defined in (4) are shown in the second and third lines in

$$H_{1,Root} \rightarrow CVE - 2015 - 1635 \rightarrow H_{3,Root} \rightarrow CVE - 2011 - 4800 \rightarrow H_{7,Root} \quad (11)$$

Thus, we can find the not-trust nodes. Based on these nodes' information and SDN, the hidden forwarding path can be built. Thus, the new network topology is shown in Figure 10.

5.1. Testing Results. In this section, we test the performance of the hidden forwarding path (Figure 10) and the traditional forwarding path (Figure 5) in a real information system network. We assume that the firewall cannot prevent the attacker. We use CVE - 1025 - 1635 and CVE - 2011 - 4800 to be the test vulnerabilities, because they have the highest attack success rate in the DMZ and Intranet, respectively. The test results are shown in Table 3.

In Table 3, the reason for "Maybe" in the third line is that the firewall and IDS may not provide an effective

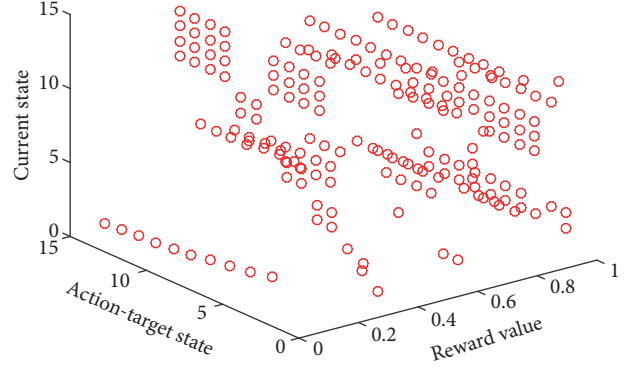


FIGURE 9: Reward matrix of improved Q-learning algorithm.

Table 2. In this table, the first line shows the membership of every pos and sev, and the second shows every element's factor.

According to Table 2, the initial reward matrix is obtained and is shown in Figure 9.

In this paper, we set Pr to 0.1 if the target host is the source host and the reward matrix is recalculated based on the new P_r . Besides, we assume that the attackers' goal is to destroy the database of the target network; thus the optimal attack path is

defense against a specific type of attack, that is, a potentially dangerous function. But in the hidden forwarding path, the defense strategy is designed to ensure that the suspicious packet is filtered by the virtual host. Furthermore, because we implement fewer defense strategies on a virtual host, the cost of the operation will not be excessive. On the other hand, adding some virtual forwarding nodes in the target network will add memory utilization and CPU utilization in the target network, although the total cost of this defense strategy is much lower than stopping some servers until the system vulnerabilities are repaired.

Besides, we also offer the optimal attack path using traditional Q-learning algorithm:

$$H_{1,Root} \rightarrow CVE - 2015 - 1635 \rightarrow H_{3,Root} \rightarrow CVE - 2015 - 1635 \rightarrow H_{3,User} \rightarrow CVE - 2011 - 4800 \rightarrow H_{7,Root} \quad (12)$$

Comparing the optimal attack path obtained from improved Q-learning algorithm with the path gained from traditional Q-learning algorithm, we can see that the result of improved Q-learning algorithm is terser than traditional Q-learning.

We also compare our method with other defense strategies. The results are shown in Table 4. The computing method

of value of "Cost" of an algorithm refers to security metrics and defense cost mentioned in [6]. The "Cost" consists of two parts: one is defense cost, and the other is attack cost. The "Complexity" and "Cost" determine "Performance."

In Table 4, IQL is the method proposed in this paper, TQL is traditional Q-learning algorithm, and DG-HMM is the algorithm proposed in [6]. From this table, we can find that

TABLE 3: Test result of attack target node.

Network structure	Vulnerability name	Attack vector	Filter or not	Attack successfully or not
Original routing path	CVE-1025-1635	Network	No	Yes
Hidden routing path	CVE-2015-1635	Network	Yes	No
Original routing path	CVE-2011-4800	Network	Maybe	Maybe
Hidden routing path	CVE-2011-4800	Network	Yes	Not

TABLE 4: Result of comparing test.

Method	Time complexity	Space complexity	Cost	Performance
IQL	$O(e * N)$	$O(N^2)$	(10.4, 29)	High
TQL	$O(e * N)$	$O(N^3)$	(10.4, 29)	Middle
AG-HMM	$O(T * N^2)$	$O(N^2)$	(14.3, 20.9)	Middle

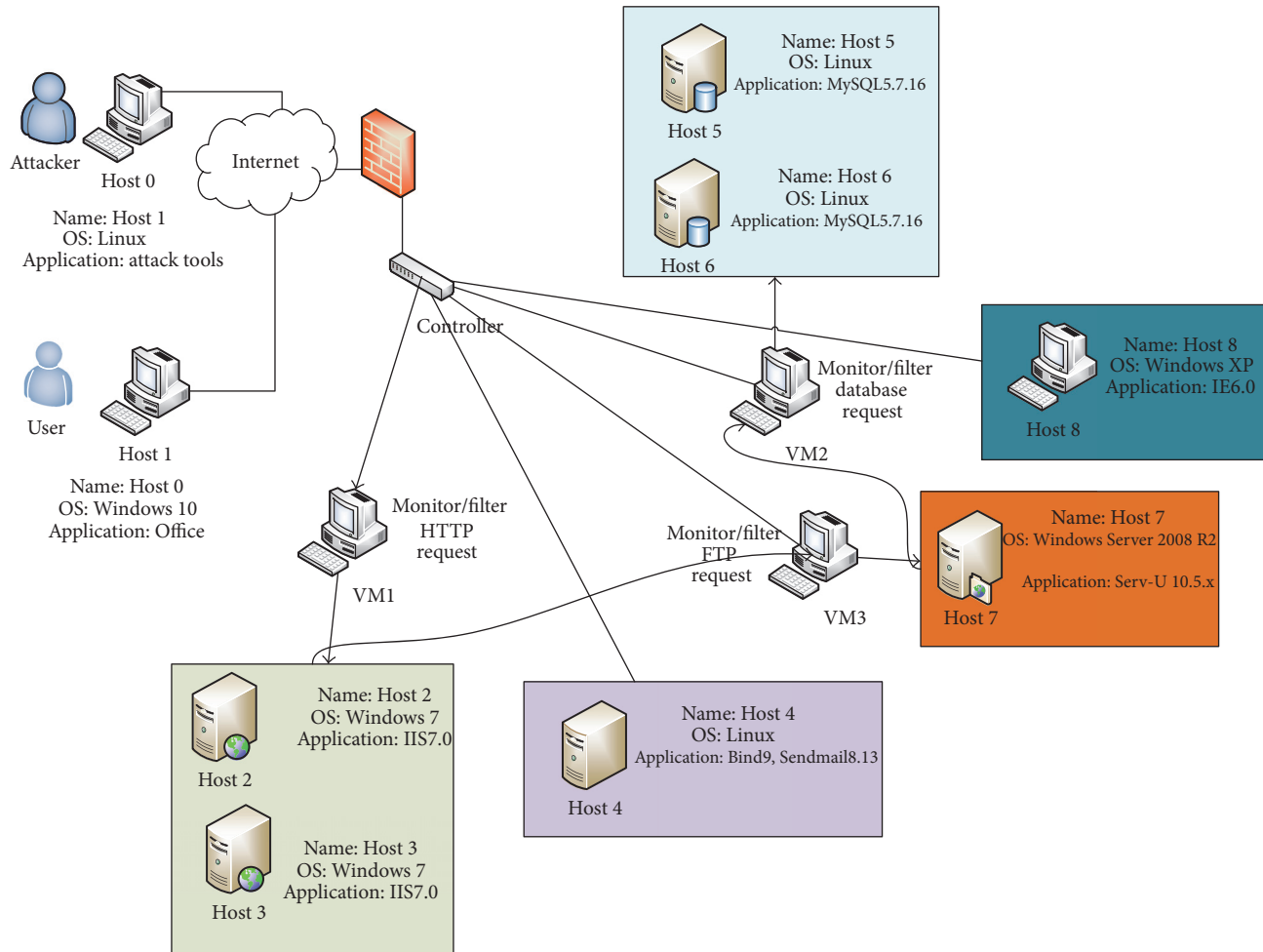


FIGURE 10: Implementing hidden routing path.

our approach is superior to the other two methods, whether in time complexity or at cost.

6. Conclusion

In this paper, we propose a dynamic hidden forwarding path planning method using improved Q-learning (IQL). The IQL improves running speed of Q-learning agent. Using IQL, the optimal attack path can be obtained quickly and does not rely on the complete or minimal attack graph. The defense strategy obtained from optimal attack path, hidden forwarding path, can be implemented in many different kinds of network environments. SDN can allocate the bandwidth based on the network traffic condition, that means although a large number of packets may be sent to one virtual host, these data will not trigger a DoS attack indirectly.

In the next stage, we aim to optimize the cost function to improve the speed of convergence. In addition, we are going to implement this method in a traditional office network without triggering a DoS attack.

Conflicts of Interest

There are not any conflicts of interest related to this paper.

Acknowledgments

This work is supported by funding from the Basic Scientific Research Program of Chinese Ministry of Industry and Information Technology (Grant no. JCKY2016602B001).

References

- [1] R. Ritchey, B. O'Berry, and S. Noel, "Representing TCP/IP connectivity for topological analysis of network security," in *Proceedings of the 18th Annual Computer Security Application*, 2002.
- [2] R. W. Ritchey and P. Ammann, "Using model checking to analyze network vulnerabilities," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 156–165, IEEE, May 2000.
- [3] O. Sheynar, *Scenario Graphs and Attack Graphs [Ph.D. thesis]*, Carnegie Mellon University, Pittsburgh, Penn, USA, 2004.
- [4] N. Ghosh, S. Nanda, and S. K. Ghosh, "An ACO Based Approach for Detection of an Optimal Attack Path in a Dynamic Environment," in *Proceedings of the 11th International Conference Distributed Computing and Networking*.
- [5] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs, "Efficient minimum-cost network hardening via exploit dependency graphs," in *Proceedings of the 19th Annual Computer Security Applications Conference, ACSAC 2003*, pp. 86–95, December 2003.
- [6] S. Wang, Z. Zhang, and Y. Kadobayashi, "Exploring attack graph for cost-benefit security hardening: a probabilistic approach," *Computers & Security*, vol. 32, pp. 158–169, 2013.
- [7] S. Akbar, J. A. Chandulal, K. Nageswara Rao, and G. Sudheer Kumar, "Improving network security using machine learning techniques," in *Proceedings of the 2012 3rd IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2012*, December 2012.
- [8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, Mass, USA, 2016.
- [9] Natl Institute of Standards and Technology, *National Vulnerability Database Version 2.2*, 2008, <http://www.nvd.org>.
- [10] Common vulnerabilities and exposures. <http://cve.mitre.org/>.
- [11] P. Hu, H. Li, H. Fu, D. Cansever, and P. Mohapatra, "Dynamic defense strategy against advanced persistent threat with insiders," in *Proceedings of the 34th IEEE Annual Conference on Computer Communications and Networks, IEEE INFOCOM 2015*, pp. 747–755, May 2015.
- [12] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "K-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 1, pp. 30–44, 2014.
- [13] Software Defined Networking (SDN): Layers and Architecture Terminology. <https://www.rfc-editor.org/rfc/rfc7426.txt>.
- [14] Y. Chen, K. Lv, and C. Hu, "Optimal Attack Path Generation Based on Supervised Kohonen Neural Network," *Network and System Security*, pp. 399–412, 2017.
- [15] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," in *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2016*, pp. 1060–1068, May 2016.
- [16] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement," in *Proceedings of the 17th International Conference on Machine Learning*, pp. 663–670, 2000.
- [17] Common vulnerability scoring system. <https://www.first.org/cvss>.
- [18] Y. Zhang, X. Yun, and M. Hu, "Research on privilege-escalating based vulnerability taxonomy with multidimensional quantitative attribute," *Journal of China Institute of Communications (in Chinese with English abstract)*, pp. 107–114, 2004.
- [19] G. Stoneburner, A. Goguen, and A. Feringa, "Risk management guide for information technology systems :," National Institute of Standards and Technology NIST SP 800-30, 2002.

Research Article

A New Type of Graphical Passwords Based on Odd-Elegant Labelled Graphs

Hongyu Wang ¹, Jin Xu,¹ Mingyuan Ma,¹ and Hongyan Zhang²

¹School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
²School of Management Science and Engineering, Shandong Normal University, Jinan 250014, China

Correspondence should be addressed to Hongyu Wang; why5126@pku.edu.cn

Received 11 January 2018; Accepted 1 March 2018; Published 11 April 2018

Academic Editor: Zheng Yan

Copyright © 2018 Hongyu Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Graphical password (GPW) is one of various passwords used in information communication. The QR code, which is widely used in the current world, is one of GPWs. Topsnut-GPWs are new-type GPWs made by topological structures (also, called graphs) and number theory, but the existing GPWs use pictures/images almost. We design new Topsnut-GPWs by means of a graph labelling, called odd-elegant labelling. The new Topsnut-GPWs will be constructed by Topsnut-GPWs having smaller vertex numbers; in other words, they are compound Topsnut-GPWs such that they are more robust to deciphering attacks. Furthermore, the new Topsnut-GPWs can induce some mathematical problems and conjectures.

1. Introduction and Preliminary

1.1. Researching Background. Graphical passwords (GPWs) have been investigated for over 20 years, and many important results can be found in three surveys [1–3]. GPW schemes have been proposed as a possible alternative to text-based schemes. However, the existing GPWs have (i) no mathematical computation; (ii) more storage space; (iii) no individuality; (iv) geometric positions; (v) slow running speed; (vi) vulnerable to attack; and (vii) no transformation from lower safe level to high security. However, QR code is a successful example of GPW’ applications in mobile devices by fast, relatively reliable and other functions [4, 5]. GPWs may be accepted by users having mobile devices with touch screen [6, 7].

Wang et al. show an idea of “topological structures plus number theory” for designing new-type GPWs (abbreviated

as Topsnut-GPWs, [8–10]). All topological structures used in Topsnut-GPWs can be stored in a computer through ordinary algebraic matrices. And Topsnut-GPWs have no requirement of geometric positions for users and allow users to make their individual passwords rather than learning more rules they do not like and so on.

How to quickly build up a large scale of Topsnut-GPWs from those Topsnut-GPWs having smaller vertex numbers? How to construct a one-key versus more-locks (one-lock versus more-keys) for some Topsnut-GPWs? And how to compute Topsnut-GPWs’ space by the basic computing unit 2^n ? Obviously, we need enough graphs and lots of graph coloring/labellings, and we can turn more things into Topsnut-GPWs. Let G_p be the number of graphs having p vertices. From [11], we know

p	G_p	bits
18	1787577725145611700547878190848	100
19	24637809253125004524383007491432768	114
20	645490122795799841856164638490742749440	129
21	32220272899808983433502244253755283616097664	145
22	3070846483094144300637568517187105410586657814272	161
23	559946939699792080597976380819462179812276348458981632	179
24	195704906302078447922174862416726256004122075267063365754368	197

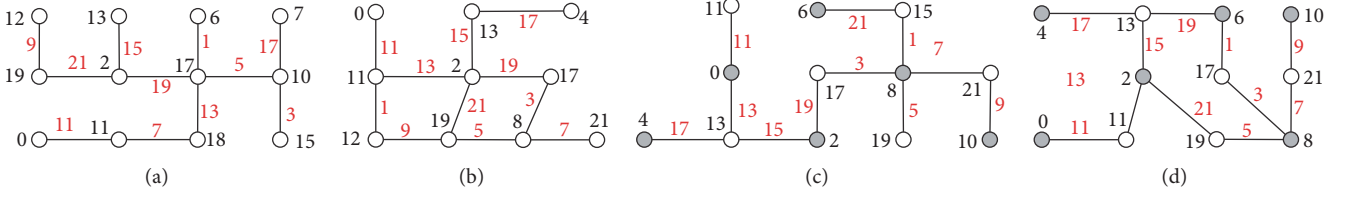


FIGURE 1: (a) An odd-elegant tree; (b) an odd-elegant graph; (c) a set-ordered odd-elegant tree; (d) a set-ordered odd-elegant graph.

where $G_p \approx 2^{\text{bits}}$ for $p = 18, 19, \dots, 24$. It means that adding various graph labellings enables us to design tremendous Topsnut-GPWs with huge topological structures and vast of graph coloring/labellings, since there are over 150 graph

labellings introduced in [12]. As a fact, Topsnut-GPWs can generate alphanumeric passwords with longer units. As an example, we take a path $v_1 v_{10} v_{11} v_{20}$ in Figure 6(d) to produce an alphanumeric password

$$W = 1'1816141210201'10'11517211110'11'10202011'20'111579320' \quad (1)$$

by selecting the neighbors of each vertex of these four vertices v_1, v_{10}, v_{11} , and v_{20} . Clearly, such password W may have longer unit in a large scale of Topsnut-GPW for meeting the need of high level security.

In this article, we will apply a graph labelling called odd-elegant labelling [13]. And we will define some construction operations under odd-elegant labelling for designing our compound Topsnut-GPWs.

1.2. Preliminary. We use standard notation and terminology of graph theory. Graphs mentioned are loopless, with no multiple edges, undirected, connected, and finite, unless otherwise specified. Others can be found in [14]. Here, we will use $A(p, q)$ -graph G which is one with p vertices and q edges; the symbol $[m, n]$ stands for an integer set $\{m, m+1, \dots, n\}$ for integers m and n with $0 \leq m < n$; $[s, t]^o$ indicating an odd-set $\{s, s+2, \dots, t\}$, where s and t both are odd integers with $1 \leq s < t$; and $[k, \ell]^e$ represents an even-set $\{k, k+2, \dots, \ell\}$, where k and ℓ are both even integers with respect to $0 \leq k < \ell$.

Definition 1 (see [13]). Suppose that a (p, q) -graph G admits a mapping $f : V(G) \rightarrow [0, 2q-1]$ such that $f(u) \neq f(v)$ for distinct vertices $u, v \in V(G)$, and the label $f(uv)$ of every edge $uv \in E(G)$ is defined as $f(uv) = f(u) + f(v) \pmod{2q}$ and the set of all edge labels is equal to $[1, 2q-1]^o$. One considers f to be an *odd-elegant labelling* and G to be an *odd-elegant*.

Definition 2 (see [15]). Suppose that a bipartite graph G receives a labelling f such that $\max\{f(x) : x \in X\} < \min\{f(y) : y \in Y\}$, where (X, Y) is the bipartition of vertex set $V(G)$ of G . We call f a *set-ordered labelling* (So-labelling for short).

As shown in Figure 1, there are four different examples of Definitions 1 and 2.

Definition 3. Let G_j be a (p_j, q_j) -graph with $j = 1, 2$. A graph G obtained by identifying each vertex $x_{i,1}$ of G_1 with a vertex $x_{i,2}$ of G_2 into one vertex $x_i = x_{i,1} \circ x_{i,2}$ with

$i \in [1, m]$ is called an *m-identification graph* and denoted as $G = \odot_m \langle G_1, G_2 \rangle$; the vertices x_1, x_2, \dots, x_m are called the *identification-vertices*.

Moreover, the *m-identification graph* $G = \odot_m \langle G_1, G_2 \rangle$ defined in Definition 3 has $p_1 + p_2 - m$ vertices and $q_1 + q_2$ edges. One can split each identification-vertex $x_i = x_{i,1} \circ x_{i,2}$ into two vertices $x_{i,1}$ and $x_{i,2}$ (called the *splitting-vertices*) for $i \in [1, m]$, such that G is split into two parts G_1 and G_2 . For the purpose of convenience, the above procedure of producing an *m-identification graph* $G = \odot_m \langle G_1, G_2 \rangle$ is called an *m-identification operation*; conversely, the procedure of splitting $G = \odot_m \langle G_1, G_2 \rangle$ into two parts G_1 and G_2 is named as the *m-splitting operation*.

Definition 4. Let G_i be a connected (p_i, q_i) -graph with $i = 1, 2$, and let $p = p_1 + p_2 - 2$. If the 2-identification (p, q) -graph $G = \odot_2 \langle G_1, G_2 \rangle$ has a mapping $f : V(G) \rightarrow [0, q-1]$ holding the following: (i) $f(x) \neq f(y)$ for each pair of vertices $x, y \in V(G)$, (ii) f is an odd-elegant labelling of G_i with $i = 1, 2$, and (iii) $|f(V(G_1)) \cap f(V(G_2))| = 2$ and $f(V(G_1)) \cup f(V(G_2)) \subseteq [0, q-1]$, then one calls G a *twin odd-elegant graph* (a TOE-graph), f a *TOE-labelling*, G_1 a *TOE-source graph*, G_2 a *TOE-associated graph*, and (G_1, G_2) a *TOE-matching pair*.

We illustrate Definition 4 with Figure 2. In other words, a twin odd-elegant graph $G = \odot_2 \langle G_1, G_2 \rangle$ with its TOE-source graph G_1 and TOE-associated graph G_2 , where (G_1, G_2) is a TOE-matching pair.

Furthermore, if each G_i with $i = 1, 2$ is a connected graph in Definition 4, and the TOE-source G_1 is a bipartite connected graph having its own bipartition (X_1, Y_1) and a labelling f satisfying Definition 2, we call the 2-identification graph $G = \odot_2 \langle G_1, G_2 \rangle$ a *set-ordered twin odd-elegant graph* (So-TOE-graph) and f a *set-ordered twin odd-elegant labelling* (So-TOE-labelling). Notice that the source graph G_1 is a set-ordered odd-elegant graph by Definitions 1 and 2. In vivid speaking, a source graph and its associated graph

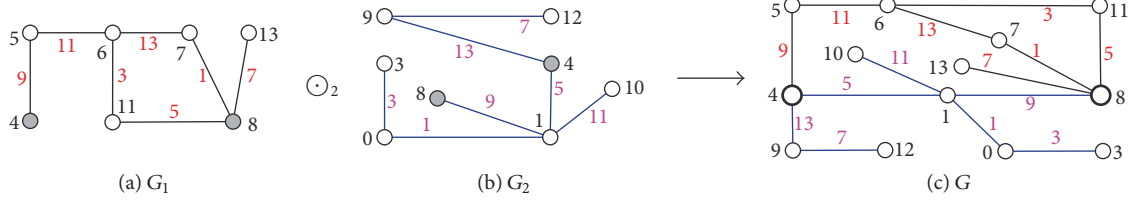


FIGURE 2: The formation process of Definition 4.

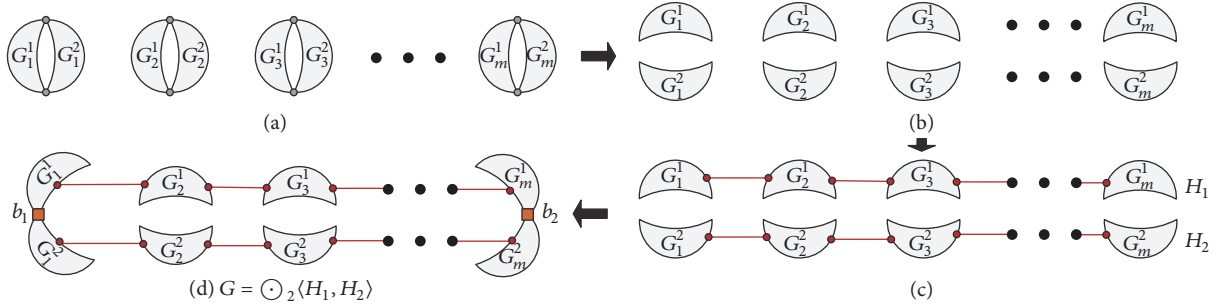


FIGURE 3: A scheme of the edge-series operation.

defined in Definition 4 can be called a *TOE-lock-model* and a *TOE-key-model* ([10]), respectively.

1.3. Techniques for Constructing 2-Identification Graphs. The following three operations, *CA-operation*, *edge-series operation*, and *base-pasted operation*, will be used in this article.

(O-1) CA-Operation. Suppose each graph G_k has an odd-elegant labelling f_k and $V(G_k) = \{x_l^k : l = 1, 2, \dots, |V(G_k)|\}$ with $k \in [1, m]$. Clearly, for $a \neq b$ with $a, b \in [1, m]$, there are vertices $x_a^i \in V(G_a)$ and $x_b^j \in V(G_b)$ such that $f_a(x_a^i) = f_b(x_b^j)$. For example, some G_k has a vertex x_i^k such that the label $f_k(x_i^k) = 0$ with $k \in [1, m]$. We can combine those vertices that have the same labels into one vertex, which gives us a new graph, denoted by $G = \odot_\epsilon \langle G_1, G_2, \dots, G_m \rangle$. This process is called a *CA-operation* on G_1, G_2, \dots, G_m .

(O-2) Edge-Series Operation. Given two groups of disjoint trees $G_1^r, G_2^r, \dots, G_m^r$ with $r = 1, 2$ there are vertices $x_k^r, y_k^r \in V(G_k^r)$ with $k \in [1, m]$. Joining the vertex y_j^r with the vertex x_{j+1}^r by an edge for $j \in [1, m-1]$ produces a tree H_r (denoted by $H_r = \odot_{\epsilon}^m G_k^r$) with $r = 1, 2$; next we let one vertex $u_s^1 \in V(H_1)$ coincide with one vertex $v_s^2 \in V(H_2)$ into one vertex $a_s = u_s^1 \circ v_s^2$ with $s = 1, 2$. The resulting graph $\odot_2 \langle H_1, H_2 \rangle$ is just a 2-identification graph.

(O-3) Base-Pasted Operation. Given two disjoint trees T_r (called *base-trees*) having vertices $x_1^r, x_2^r, \dots, x_m^r$ and two groups of disjoint trees $G_1^r, G_2^r, \dots, G_m^r$ with $r = 1, 2$, we let a vertex $u_k^r \in V(G_k^r)$ coincide with the vertex $x_k^r \in V(T_r)$ into one vertex $u_k^r \circ x_k^r$ for $k \in [1, m]$ such that the resulting tree F_r (i.e., $F_r = T_r \odot_{\epsilon}^m G_k^r$) has $V(F_r) = \bigcup_{k=1}^m V(G_k^r)$, $E(F_r) = (\bigcup_{k=1}^m E(G_k^r)) \cup E(T_r)$ for $r = 1, 2$. We overlap one vertex $w_s^1 \in V(F_1)$ with one vertex $z_s^2 \in V(F_2)$ into one vertex

$b_s = w_s^1 \circ z_s^2$ with $s = 1, 2$ to build up a 2-identification graph $F = \odot_2 \langle F_1, F_2 \rangle$ holding $V(F_1) \cap V(F_2) = \{a_1, a_2\}$ and $E(F) = E(F_1) \cup E(F_2)$.

In the following, we give the diagrams with $m = 2$ for edge-series operation and base-pasted operation, shown in Figures 3 and 4, respectively.

2. Main Results and Their Proofs

Lemma 5. Each star $K_{1,n}$ is a *TOE-source tree* of a *So-TOE-tree*.

Proof of Lemma 5 is shown in Figure 5. It describes the construction process of the *So-TOE-tree* $\odot \langle K_{1,n}, K_{1,n} \rangle$ by any *TOE-source tree* $K_{1,n}$.

Theorem 6. Every set-ordered odd-elegant graph being not a star is a *So-TOE-source graph* of at least two *So-TOE graphs*.

Proof. Suppose that (p_1, q) -graph G_1 having vertex bipartition is (X, Y) , where $X = \{x_i : i \in [1, s]\}$, $Y = \{y_j : j \in [1, t]\}$, $s + t = p_1$, and $\min\{s, t\} \geq 2$. By the hypothesis of the theorem, G_1 has a set-ordered odd-elegant labelling f_1 defined by $f_1(x_i) + 2 \leq f_1(x_{i+1})$, $i \in [1, s-1]$; $f_1(y_1) = f_1(x_s) + 1$, $f_1(y_j) + 2 \leq f_1(y_{j+1})$, $j \in [1, t-1]$; $f_1(y_t) \leq 2q - 1$. Hence, $f_1(E(G_1)) = \{f_1(xy) = f_1(x) + f_1(y) \pmod{2q} : xy \in E(G_1)\} = [1, 2q - 1]^o$. It is not difficult to observe that $f_1(V(G_1)) \subset \{0, 2, \dots, f_1(x_s), f_1(y_1), \dots, 2q - 1\}$; that is, $f_1(X)/2 \subset \mathbb{N}$ and $(f_1(Y) + 1)/2 \subset \mathbb{N}$.

Case 1. We construct a labelling f_2 of a new tree T_2 having $q+1$ vertices by the labelling f_1 such that $f_2(V(T_2)) = [1, f_1(x_s) + 1]^o \cup [f_1(y_1) - 1, 2q - 2]^e$, such that $f_2(E(T_2)) = \{f_2(uv) = f_2(u) + f_2(v) \pmod{2q} : uv \in E(T_2)\} = [1, 2q - 1]^o$, where $f_2(u) \neq f_2(v)$ for $u, v \in V(T_2)$. This tree T_2 can be built up in the following way: a bipartition (U_1, V_1) with $U_1 = \{u_i :$

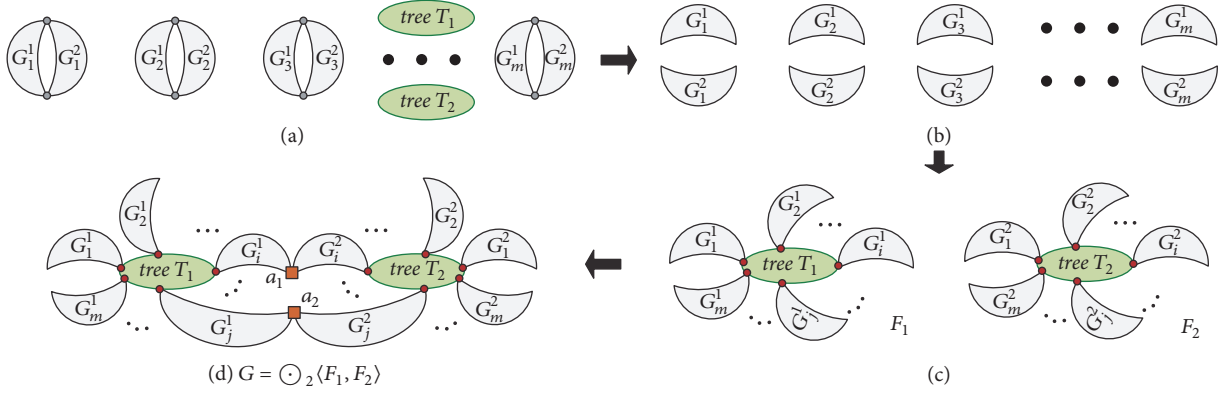


FIGURE 4: A scheme of the base-pasted operation.

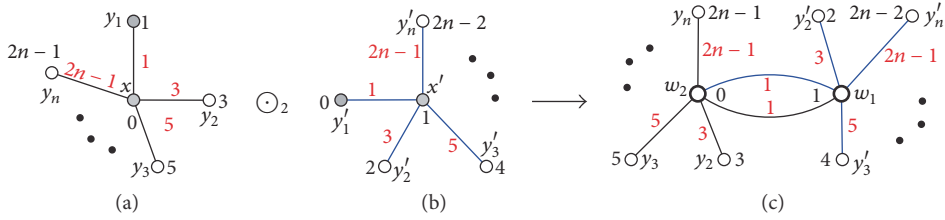


FIGURE 5: An example of illustrating Lemma 5.

$i \in [1, s_1]$ and $V_1 = \{v_j : j \in [1, t_1]\}$, where $s_1 + t_1 = q + 1$, such that $f_2(u_i) = 2i - 1$, $i \in [1, s_1]$; $f_2(v_j) = 2(s_1 - 2 + j)$, $j \in [1, t_1]$. Any edge $u_i v_j \in E(T_2)$ satisfies $f_2(u_i v_j) = f_2(u_i) + f_2(v_j) \pmod{2q}$ with $i \in [1, s_1]$ and $j \in [1, t_1]$. We construct the edge set of T_2 as $\{u_i v_j, u_i v_{t_1} : i \in [1, s_1], j \in [1, t_1 - 1]\}$ such that the edge labels are $f_2(u_i v_{t_1}) = 2i - 3$, $f_2(u_i v_j) = 2j + 2s_1 - 3 \pmod{2q}$ for $i \in [1, s_1]$ and $j \in [1, t_1 - 1]$. Observe that $f_2(E(T_2)) = [1, 2q - 1]^0$, $f_1(y_1) = f_2(u_{s_1})$, and $f_1(x_s) = f_2(v_1)$.

Now, we can combine the vertex y_1 and x_s of G_1 with the vertex u_{s_1} and v_1 of T_2 into one (two identification-vertices) w_1 and w_2 , respectively, so we obtain the desired graph $G = \odot_2 \langle G_1, T_2 \rangle$. And G has a labelling f defined as $f(x_i) = f_1(x_i)$, $i \in [1, s - 1]$; $f(y_j) = f_1(y_j)$, $i \in [2, t]$; $f(u_k) = f_2(u_k)$, $k \in [1, s_1 - 1]$, $f(v_l) = f_2(v_l)$, $l \in [2, t_1]$, $f(w_1) = f_1(y_1)$, and $f(w_2) = f_1(x_s)$. Clearly, any pair of two vertices of G are assigned different numbers. According to Definition 4, G is an So-TOE-graph having the source graph G_1 . Examples that illustrate Case 1 of Theorem 6 are shown by Figures 6(a), 6(b), and 6(d).

Case 2. Similarly to Case 1, we can get the following results: let $f_2(V(T'_2)) = [1, f_1(x_s) - 1]^0 \cup [f_1(y_1) - 1, 2q - 2]^e \cup \{0\}$, $f_2(E(T'_2)) = [1, 2q - 1]^0$, and furthermore $f_2(u) \neq f_2(v)$ for $u, v \in V(T'_2)$. This tree T'_2 can be built up in the following way: a bipartition (U_2, V_2) with $U_2 = \{u_i : i \in [1, s_1 - 1]\}$ and $V_2 = \{v_j : j \in [1, t_1 + 1]\}$, such that $f_2(u_i) = 2i - 1$, $i \in [1, s_1 - 1]$; $f_2(v_j) = 2(s - 2 + j)$, $j \in [1, t_1]$, $f_2(v_{t_1+1}) = 0$. Any edge $u_i v_j \in E(T'_2)$ satisfies $f_2(u_i v_j) = f_2(u_i) + f_2(v_j) \pmod{2q}$ with $i \in [1, s_1 - 1]$ and $j \in [1, t_1 + 1]$. We construct the edge set of T'_2 as $\{u_1 v_j, u_i v_{t_1+1} : i \in [2, s_1 - 1], j \in [1, t_1]\}$ such that

the edge labels are $f_2(u_i v_{t_1+1}) = 2i - 1$, for $i \in [1, s_1 - 1]$, and $f_2(u_1 v_j) = 2(s + j) - 3$, for $j \in [1, t_1]$. Observe that $f_2(E(T'_2)) = [1, 2q - 1]^0$, $f_1(x_1) = f_2(v_{t_1+1})$, and $f_1(x_s) = f_2(v_1)$.

Now, we can combine the vertex x_1 and x_s of T_1 with the vertex v_{t_1+1} and v_1 of T'_2 into one (the identified vertex) w_1 and w_2 , so we obtain the desired tree $G' = \odot_2 \langle G_1, T'_2 \rangle$. And G' has a labelling f defined as $f(x_i) = f_1(x_i)$, $i \in [2, s - 1]$; $f(y_j) = f_1(y_j)$, $i \in [1, t]$; $f(u_k) = f_2(u_k)$, $k \in [1, s_1 - 1]$, $f(v_l) = f_2(v_l)$, $l \in [2, t_1]$, $f(w_1) = 0$, and $f(w_2) = f_1(x_s)$. Clearly, any pair of two vertices of G' are assigned different numbers. According to Definition 4, G' is a So-TOE-graph having the source graph G_1 . An example for illustrating Case 2 of Theorem 6 is given by Figures 6(a), 6(c), and 6(e). \square

Theorem 7. Suppose that $G_k = \odot_2 \langle G_k^1, G_k^2 \rangle$ is a So-TOE-graph, where each G_k^1 is a source tree for $k \in [1, m]$. Then $G = \odot_2 \langle H_1, H_2 \rangle$ obtained by the edge-series operation has a So-TOE-labelling.

Proof. By the hypothesis of the theorem, every $(p_k^1 + p_k^2 - 2, 2q_k)$ -graph G_k has a set-ordered odd-elegant source- (p_k^1, q_k) -graph G_k^1 and an associated- (p_k^2, q_k) -graph G_k^2 for $k \in [1, m]$. Let $V(G_k^1) \cap V(G_k^2) = \{w_k^1, w_k^2\}$; the vertex set of each graph G_k^r can be partitioned into (X_k^r, Y_k^r) with $r = 1, 2$, where $X_k^r = \{x_{k,i}^r : i \in [1, s_k^r]\}$, $Y_k^r = \{y_{k,j}^r : j \in [1, t_k^r]\}$, and $s_k^r + t_k^r = p_k^r$ for $k \in [1, m]$ and $r = 1, 2$. By Definition 4, every G_k has a So-TOE-labelling θ_k with $k \in [1, m]$ such that $\theta_k(x_{k,i}^r) \geq r - 1$; $\theta_k(x_{k,i}^r) + 2 \leq \theta_k(x_{k,i+1}^r)$ with $i \in [1, s_k^r]$; $\theta_k(y_{k,j}^r) = \theta_k(x_{k,s_k^r}^r) - (-1)^r$; $\theta_k(y_{k,j}^r) + 2 \leq \theta_k(y_{k,j+1}^r)$ for $i \in [1, t_k^r]$; and $\theta_k(y_{k,i}^r) \leq 2q_k - r$.

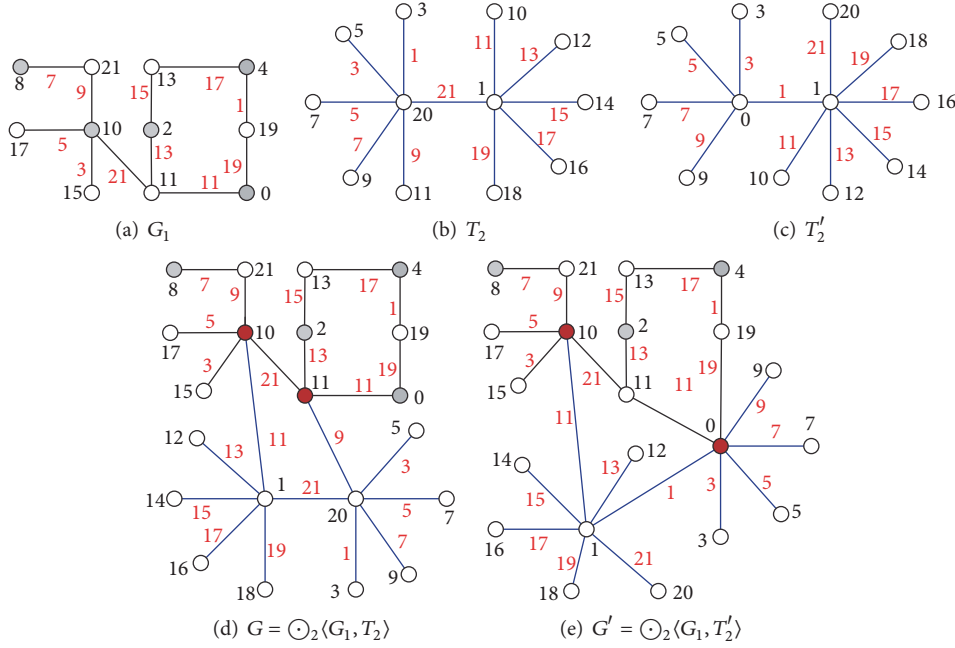


FIGURE 6: Examples of Theorem 6.

Therefore, $\theta_k(E(G_k^r)) = \{\theta_k(xy) = \theta_k(x) + f_k^r(y) \pmod{2q_k} : xy \in E(G_k^r)\} = [1, 2q_k - 1]^0$, where $\theta_k(x) \neq \theta_k(y)$ for distinct vertices $x, y \in V(G_k)$, which means $\theta_k(x_{k,s_k}^1) = \theta_k(y_{k,1}^2) = \theta_k(w_k^1)$ and $\theta_k(y_{k,1}^1) = \theta_k(x_{k,s_k}^2) = \theta_k(w_k^2)$. Clearly, the labels of other vertices of $G_k^1 \cup G_k^2$ differ from each other.

Firstly, we split G_k into two parts G_k^1 and G_k^2 , that is, doing a 2-splitting operation on every G_k with $k \in [1, m]$. Secondly, our discussion focuses on G_k^1 and G_k^2 with $k \in [1, m]$. We construct a graph by joining the vertex y_{k,t_k}^r with the vertex $x_{k+1,1}^r$ by an edge, where $k \in [1, m-1]$ and $r = 1, 2$, called H_r . For the purpose of convenience, we set $S(a, b) = \sum_{l=a}^b \theta_l(x_{l,s_l}^1) + 2$, $Q(1, t) = 2 \sum_{l=1}^t (q_l + 1)$, $Q_m = 2 \sum_{l=1}^m (q_l + m - 1)$, $S(1, 0) = 0$, and $Q(1, 0) = 0$. For $r = 1, 2$, $i \in [1, s_k^r]$, and $j \in [1, t_k^r]$, we define a new labelling f as follows:

- (T-1) $f(x_{k,i}^r) = \theta_k(x_{k,i}^r) + S(1, k-1)$;
- (T-2) $f(y_{k,j}^r) = \theta_k(y_{k,j}^r) + Q(1, k-1) + S(k+1, m)$;
- (T-3) $f(x_{k,i}^r y_{k,j}^r) = f(x_{k,i}^r) + f(y_{k,j}^r) \pmod{Q_m}$;
- (T-4) $f(y_{k,t_k}^r x_{k+1,1}^r) = f(x_{k+1,1}^r) + f(y_{k,t_k}^r) \pmod{Q_m}$.

By the labelling forms (T-1) and (T-2) above, we can verify $f(x_{k,i}^1) \in [0, f(x_{m,s_m}^1)]^e = [0, S(1, m) - 2]^e$ with $k \in [1, m]$ and have the following properties: (i) $f(x_{k,i}^2) \in [1, f(x_{m,s_m}^2)]^o = [1, S(1, m) - 1]^o$; (ii) $f(y_{k,j}^1) \in [f(y_{1,1}^1), f(y_{m,t_m}^1)]^o = [S(1, m) - 1, Q_m - 1]^o$; and (iii) $f(y_{k,j}^2) \in [f(y_{1,1}^2), f(y_{m,t_m}^2)]^e = [S(1, m) - 2, Q_m - 2]^e$.

Computing the labelling forms (T-3) and (T-4) enables us to obtain $f(E(H_r)) = [1, Q_m - 1]^o$ for $r = 1, 2$. Now, we combine the vertex x_{m,s_m}^1 with the vertex $y_{1,1}^2$ into one

vertex and then combine the vertex $y_{1,1}^1$ with the vertex x_{m,s_m}^2 into one vertex. (i.e., do the 2-identification operation). Thus the labelling f is a So-TOE-labelling of $G = \odot_2(H_1, H_2)$; therefore, G is a So-TOE-graph too. \square

See Figures 7, 8 and 9 for understanding Theorem 7.

In experiments, for each arrangement $G_{k_1}^r, G_{k_2}^r, \dots, G_{k_m}^r$ of $G_1^r, G_2^r, \dots, G_m^r$, there are many possible constructions of $G = \odot_2(H_1, H_2)$ for holding Theorem 7 (as shown in Figure 9).

Theorem 8. Suppose that $G_k = \odot_2(G_k^1, G_k^2)$ is a So-TOE-graph, where each G_k^1 is a source graph for $k \in [1, p]$. Then $G = \odot_2(S_1, S_2)$ obtained by the base-pasted operation has a So-TOE-labelling if two base-trees T_1 and T_2 are set-ordered.

Proof. By the hypothesis of the theorem, every $(p_k^1 + p_k^2 - 2, 2q)$ -graph $G_k = \odot_2(G_k^1, G_k^2)$ has a set-ordered odd-elegant source- (p_k^1, q) -graph G_k^1 and an associated- (p_k^2, q) -graph G_k^2 for $k \in [1, p]$. Let $G_k^1 \cap G_k^2 = \{w_k^1, w_k^2\}$; the vertex set of each graph G_k^r can be partitioned into (X_k^r, Y_k^r) with $r = 1, 2$, where $X_k^r = \{x_{k,i}^r : i \in [1, s_k^r]\}$, $Y_k^r = \{y_{k,j}^r : j \in [1, t_k^r]\}$, $s_k^r \leq t_k^r$, and $s_k^r + t_k^r = p_k^r$ for $k \in [1, p]$ and $r = 1, 2$. Every G_k , by Definition 4, has a So-TOE-labelling π_k with $k \in [1, p]$, and π_k has the following properties: $\pi_k(x_{k,1}^r) = r - 1$; $\pi_k(x_{k,i}^r) + 2 \leq \pi_k(x_{k,i+1}^r)$ for $i \in [1, s_k^r]$; $\pi_k(x_{k,s_k}^r) = M - 1 + r$; $\pi_k(y_{k,1}^r) = \pi_k(x_{k,s_k}^r) - (-1)^r = M - 1 + r - (-1)^r$; $\pi_k(y_{k,j}^r) + 2 \leq \pi_k(y_{k,j+1}^r)$ with $i \in [1, t_k^r]$; $\pi_k(y_{t_k}^r) = 2q - r$; and $\pi_k(x_{k,i}^r y_{k,j}^r) = \pi_k(x_{k,i}^r) + \pi_k(y_{k,j}^r) \pmod{2q}$.

Thus, the properties of each So-TOE-labelling π_k induce $\pi_k(E(G_k^r)) = \{\pi_k(xy) = \pi_k(x) + f_k^r(y) \pmod{2q} : xy \in E(G_k^r)\}$, and also

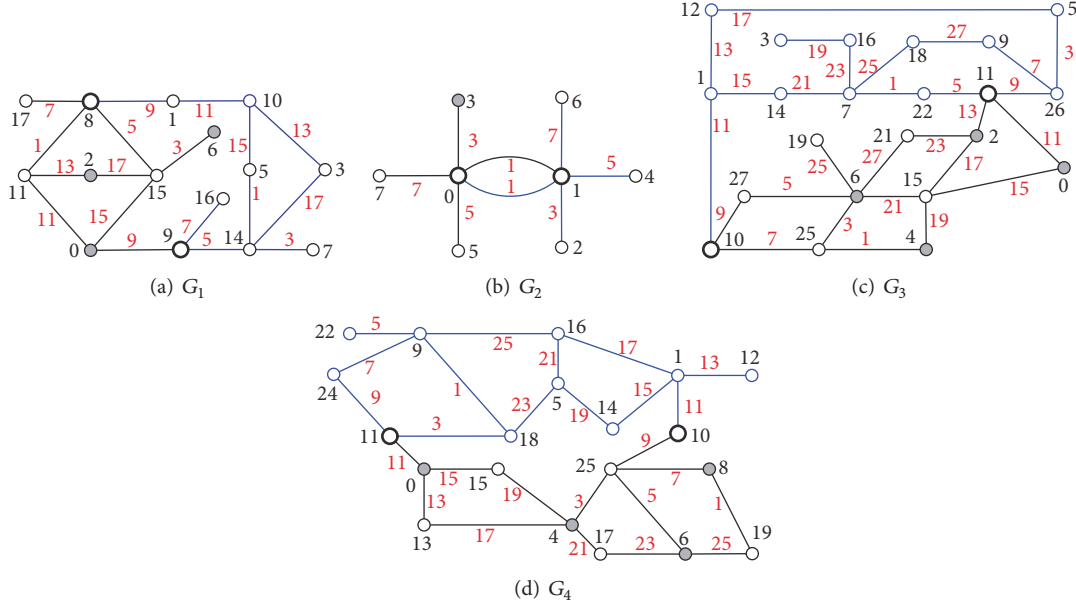
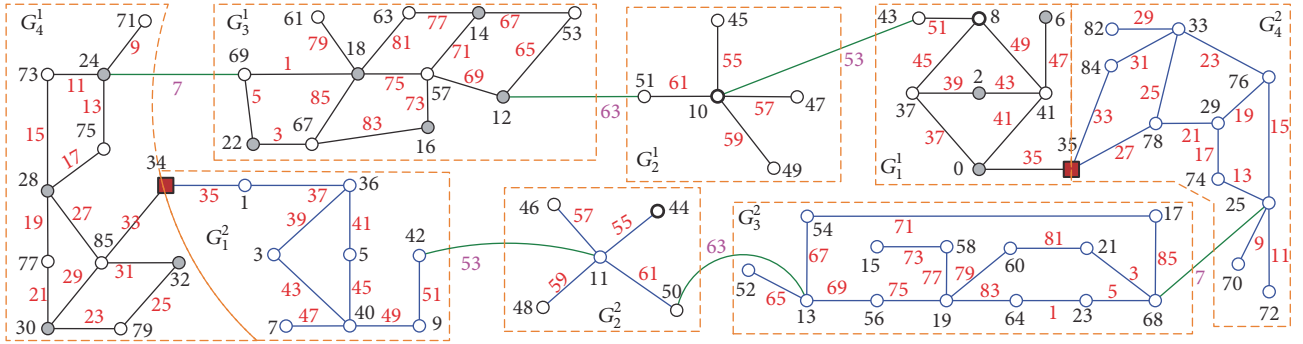
FIGURE 7: Four So-TOE-graphs G_k with $k \in [1, 4]$ described in the proof of Theorem 7.

FIGURE 8: A So-TOE-graph made by the graphs shown in Figure 7 for illustrating Theorem 7.

$$\begin{aligned}
 & \pi_k(E(G_k^r)) \\
 &= [\pi_k(x_{k,1}^r) + \pi_k(y_{k,1}^r), \pi_k(x_{k,s_k}^r) + \pi_k(y_{k,t_k}^r)] \quad (2) \\
 & \cdot (\text{mod } 2q) = [1, 2q - 1]^o,
 \end{aligned}$$

where $\pi_k(x) \neq \pi_k(y)$ if $x \neq y$ and $x, y \in V(G_k)$. In other words, we have $\pi_k(x_{k,s_k}^1) = \pi_k(y_{k,1}^2) = \pi_k(w_k^1)$ and $\pi_k(y_{k,1}^1) = \pi_k(x_{k,s_k}^2) = \pi_k(w_k^2)$. The labels of other vertices of $G_k^1 \cup G_k^2$ differ from each other.

Let $V(T_r) = \{z_1^r, z_2^r, \dots, z_p^r\}$, such that there exists a set-ordered odd-elegant labelling $f_{T_r}^{oe}$, satisfying $f_{T_r}^{oe}(z_i^r) < f_{T_r}^{oe}(z_{i+1}^r)$ with $i \in [1, p-1]$, and the bipartition (U_r, V_r) of vertex set of T_r satisfies $|U_r| \leq |V_r|$ for $|U_r| = l$ and $r = 1, 2$.

Next, we discuss all graphs G_k^1 and G_k^2 with $k \in [1, p]$ by the parity of positive integer p in the following two cases.

Case 1. For considering the case $p = 2\beta + 1$ and $r = 1, 2$, we define a new labelling f with $i \in [1, s_k^r]$ and $j \in [1, t_k^r]$ in the following way:

$$(C-1) \quad f(x_{2k-1,i}^r) = \pi_{2k-1}(x_{2k-1,i}^r) + 2(q+1)(k-1) \text{ with } k \in [1, \beta+1];$$

$$(C-2) \quad f(x_{2k,i}^r) = \pi_{2k}(x_{2k,i}^r) + 2(q+1)(\beta+k) - 2 - (-1)^r \text{ with } k \in [1, \beta];$$

$$(C-3) \quad f(y_{2k-1,j}^r) = \pi_{2k-1}(y_{2k-1,j}^r) + 2(q+1)(\beta+k-1) \text{ with } k \in [1, \beta+1];$$

$$(C-4) \quad f(y_{2k,j}^r) = \pi_{2k}(y_{2k,j}^r) + 2(q+1)(k-1) + 2 + (-1)^r \text{ with } k \in [1, \beta];$$

$$(C-5) \quad f(x_{k,i}^r y_{k,j}^r) = f(y_{k,j}^r) + f(x_{k,i}^r) \pmod{2p(q+1)-2}.$$

Based upon the labelling forms (C-1)–(C-4), we compute

$$\begin{aligned}
 & \left(\bigcup_{k=1}^{\beta+1} f(X_{2k-1}^1) \right) \cup \left(\bigcup_{k=1}^{\beta} f(Y_{2k}^1) \right) \\
 &= [0, 2(q+1)\beta + M]^e;
 \end{aligned}$$

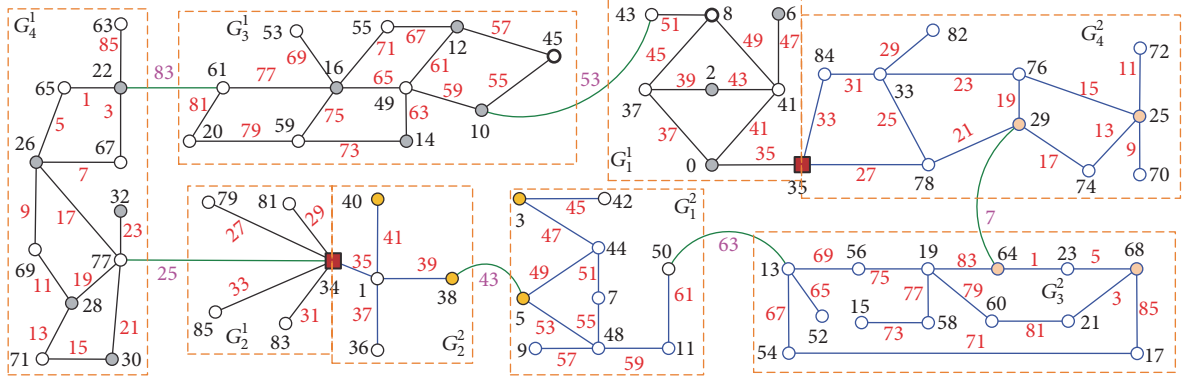


FIGURE 9: Another So-TOE-graph made by the graphs shown in Figure 7 for illustrating Theorem 7.

$$\left(\bigcup_{k=1}^{\beta+1} f(Y_{2k-1}^1) \right) \cup \left(\bigcup_{k=1}^{\beta} f(X_{2k}^1) \right)$$

$$= [2(q+1)\beta + M + 1, 2p(q+1) - 3]^o,$$

$r = 1,$

$$\left(\bigcup_{k=1}^{\beta+1} f(X_{2k-1}^2) \right) \cup \left(\bigcup_{k=1}^{\beta} f(Y_{2k}^2) \right)$$

$$= [1, 2(q+1)\beta + M + 1]^o;$$

$$\left(\bigcup_{k=1}^{\beta+1} f(Y_{2k-1}^2) \right) \cup \left(\bigcup_{k=1}^{\beta} f(X_{2k}^2) \right)$$

$$= [2(q+1)\beta + M, 2p(q+1) - 4]^e,$$

$r = 2.$

(3)

Thereby, we have shown that $\bigcup_{r=1}^2 \bigcup_{k=1}^p f(V(G_k^r)) \subset [0, 2p(q+1) - 3]$ and

$$\left(\bigcup_{r=1}^2 \bigcup_{k=1}^p f(E(G_k^r)) \right) \cup \left(\bigcup_{r=1}^2 f(E(T_r)) \right)$$

$$= [1, 2p(q+1) - 3]^o, \quad (4)$$

and furthermore the labels of vertices, except $f(x_{p,s}^1) = f(y_{1,1}^2)$ and $f(x_{p,s}^2) = f(y_{1,1}^1)$, differ from each other, and the labels of edges differ from each other.

Next, after computing the labelling forms (C-5) with $k \in [1, p]$, we obtain

$$f(E(G_{2k-1}^r)) = [A(2), B(1)]^o, \quad k \in [1, \beta+1];$$

$$f(E(G_{2k}^r)) = [A(1), B(0)]^o, \quad k \in [1, \beta]. \quad (5)$$

$$(\text{mod } 2p(q+1) - 2),$$

where $A(x) = M + 1 + 2(q+1)(\beta + 2k - x)$ and $B(y) = M - 3 + 2(q+1)(\beta + 2k - y)$. By the above deduction, we can know that

$$\bigcup_{k=1}^p f(E(G_k^r)) = [1, 2p(q+1) - 3]^o \setminus F, \quad (6)$$

where $F = \{M - 1 + 2(q+1)(\beta + 1 + k), M + 1 + 2(q+1)k : k = 0, 1, 2, \dots, \beta - 1\}$. Next, for each vertex $z_k^r \in V(T_r)$ with $k \in [1, p]$ and $r = 1, 2$, we set

$$f(z_k^1) = f(x_{2k-1, s_{2k-1}}^1),$$

$$f(z_k^2) = f(y_{2(\beta+k-l)+1, 1}^2),$$

$k \in [1, l];$

$$f(z_{l+k}^1) = f(x_{2k, 1}^1),$$

$$f(z_{\beta+1+k}^2) = f(y_{2k, t_{2k}^2}^2), \quad (7)$$

$k \in [1, \beta];$

$$f(z_{l+\beta+k}^1) = f(x_{2k-1, 1}^2),$$

$$f(z_{l+k}^2) = f(y_{2(l+k)-1, t_{2(l+k)-1}^2}^2),$$

$k \in [1, \beta + 1 - l].$

According to formula (7), we obtain $f(z_i^r z_j^r) = f(z_i^r) + f(z_j^r) \in F$ with $i \in [1, l], j \in [l+1, p]$, and $r = 1, 2$, which means

$$f(E(T_r)) = F. \quad (8)$$

Doing a CA-operation on G_k^r and T_r having labelling f for $k \in [1, p]$ produces a new graph S_r with $r = 1, 2$. Now, we combine the vertex x_{p, s_p}^1 with the vertex $y_{1, 1}^2$ into one vertex $w_1 = x_{p, s_p}^1 \circ y_{1, 1}^2$ and moreover identify the vertex $y_{1, 1}^1$ with the vertex x_{p, s_p}^2 into one vertex $w_2 = y_{1, 1}^1 \circ x_{p, s_p}^2$ (i.e., do a 2-identification operation).

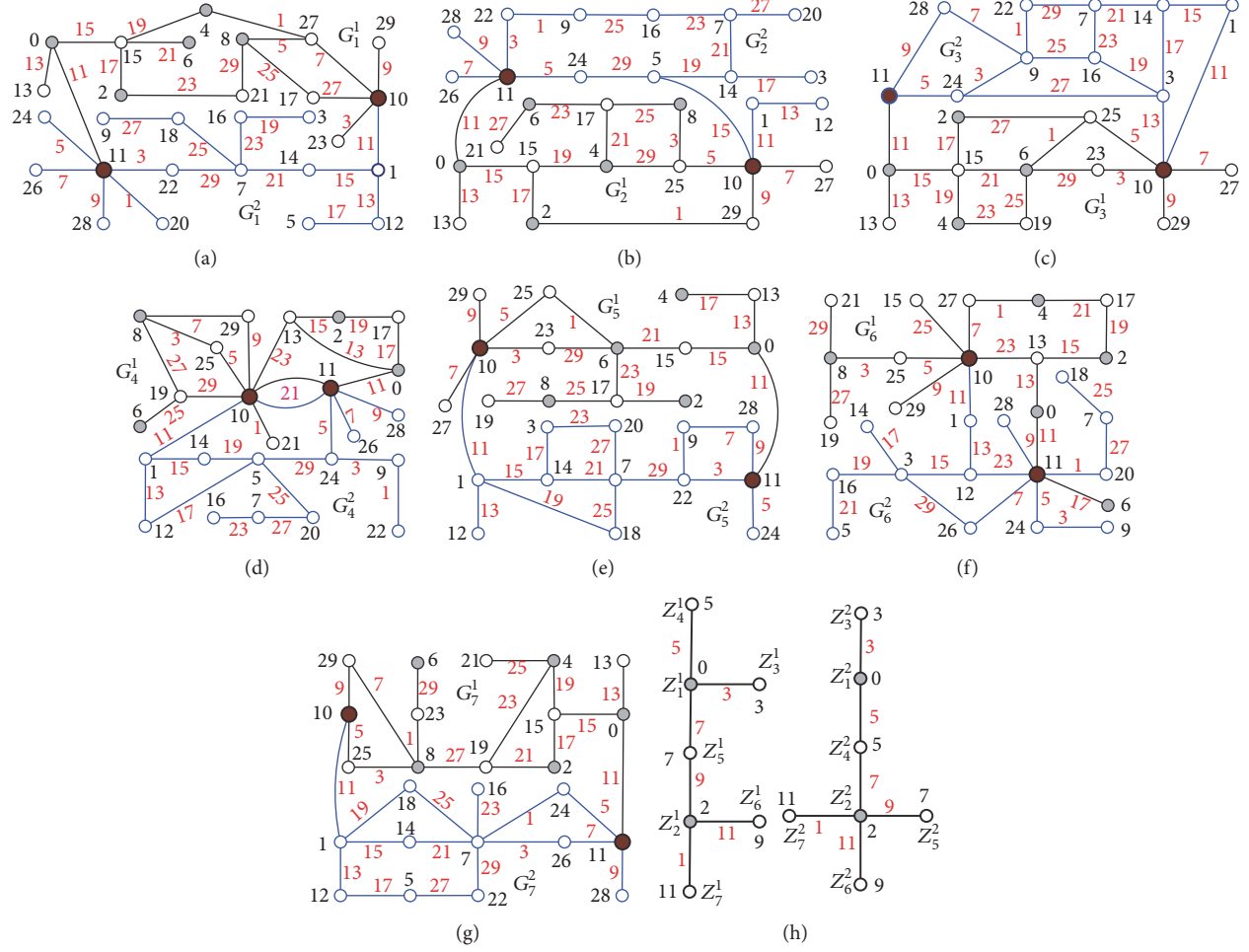


FIGURE 10: Seven So-TOE-graphs G_k with $k \in [1, 7]$ and two base-trees T_1 and T_2 described in the proof of Case 1 of Theorem 8.

By Definitions 2 and 4 and formulae (3)–(8), the labelling f is a So-TOE-labelling of $G = \odot_2 \langle S_1, S_2 \rangle$. Hence, G is a So-TOE-graph. Here, we have proven Case 1. For understanding Case 1, see Figures 10 and 11.

Case 2. We, for the case $p = 2\beta$ and $r = 1, 2$, define a new labelling f for $i \in [1, s'_k]$ and $j \in [1, t'_k]$ in the following way:

$$(L-1) \quad f(x_{2k-1,i}^r) = \pi_{2k-1}(x_{2k-1,i}^r) + 2(q+1)(k-1) \text{ with } k \in [1, \beta];$$

$$(L-2) \quad f(x_{2k,i}^r) = \pi_{2k}(x_{2k,i}^r) + 2(q+1)(\beta+k) - M - 4 - (-1)^r \text{ with } k \in [1, \beta];$$

$$(L-3) \quad f(y_{2k-1,j}^r) = \pi_{2k-1}(y_{2k-1,j}^r) + 2(q+1)(\beta+k-1) - M - 2 \text{ with } k \in [1, \beta];$$

$$(L-4) \quad f(y_{2k,j}^r) = \pi_{2k}(y_{2k,j}^r) + 2(q+1)(k-1) + 2 + (-1)^r \text{ with } k \in [1, \beta];$$

$$(L-5) \quad f(x_{k,i}^r y_{k,j}^r) = f(y_{k,j}^r) + f(x_{k,i}^r) \pmod{2p(q+1)-2}.$$

From the above labelling forms (L-1)–(L-4), we can compute

$$\begin{aligned} & \left(\bigcup_{k=1}^{\beta} f(X_{2k-1}^1) \right) \cup \left(\bigcup_{k=1}^{\beta} f(Y_{2k}^1) \right) \\ &= [0, 2(q+1)\beta - 2]^e; \end{aligned}$$

$$\begin{aligned} & \left(\bigcup_{k=1}^{\beta} f(Y_{2k-1}^1) \right) \cup \left(\bigcup_{k=1}^{\beta} f(X_{2k}^1) \right) \\ &= [2(q+1)\beta - 1, 2p(q+1) - 3]^o, \end{aligned}$$

$r = 1,$

$$\begin{aligned} & \left(\bigcup_{k=1}^{\beta} f(X_{2k-1}^2) \right) \cup \left(\bigcup_{k=1}^{\beta} f(Y_{2k}^2) \right) \\ &= [1, 2(q+1)\beta - 1]^o; \end{aligned}$$

$$\left(\bigcup_{k=1}^{\beta} f(Y_{2k-1}^2) \right) \cup \left(\bigcup_{k=1}^{\beta} f(X_{2k}^2) \right)$$

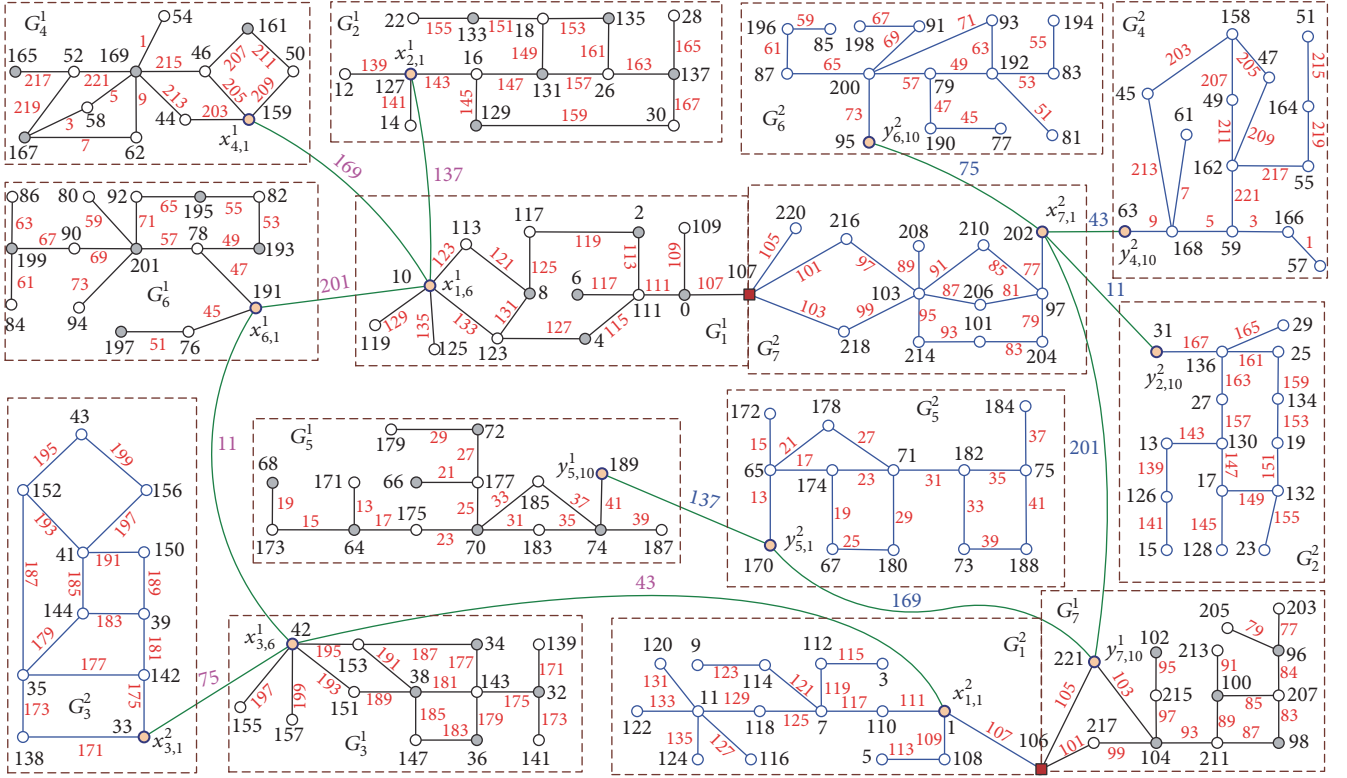


FIGURE 11: A So-TOE-graph $\odot_2 \langle S_1, S_2 \rangle$ made by the graphs shown in Figure 10 for understanding the proof of Case 1 of Theorem 8.

$$= [2(q+1)\beta - 2, 2p(q+1) - 4]^e,$$

$$r = 2.$$

(9)

Thereby, we conclude that $\bigcup_{r=1}^2 \bigcup_{k=1}^p f(V(G_k^r)) = [0, 2p(q+1) - 3]$ and

$$\left[\bigcup_{r=1}^2 \bigcup_{k=1}^p f(E(G_k^r)) \right] \cup \left[\bigcup_{r=1}^2 f(E(T_r)) \right] = [1, 2p(q+1) - 3]^o, \quad (10)$$

in which the labels of vertices and edges, except $f(y_{p,t_p}^1) = f(y_{1,1}^2)$ and $f(y_{p,t_p}^2) = f(y_{1,1}^1)$, differ from each other, respectively.

Again, by computing the labelling form (L-5) for each $k \in [1, p]$, we obtain

$$\begin{aligned} f(E(G_{2k-1}^r)) &= [\alpha(2), \beta(1)]^o; \\ f(E(G_{2k}^r)) &= [\alpha(1), \beta(0)]^o, \end{aligned} \quad (11)$$

$$(\text{mod } 2p(q+1) - 2), \quad k \in [1, \beta],$$

where $\alpha(x) = 2(q+1)(\beta + 2k - x) - 1$ and $\beta(y) = 2(q+1)(\beta + 2k - y) - 5$.

Synthesizing the above argument, we get $\bigcup_{k=1}^p f(E(G_k^r)) = [1, 2p(q+1) - 3]^o \setminus F'$, where the set $F' = \{2(q+1)(\beta +$

$k) - 3 \pmod{2p(q+1) - 2} : k \in [1, 2\beta - 1]\}$. For each vertex $z_k^r \in T_r$ with $r \in [1, p]$ and $r = 1, 2$, we set

$$f(z_k^1) = f(x_{2k-1,1}^1),$$

$$f(z_k^2) = f(x_{2(\beta+k-l), s_{2(\beta+k-l)}^2}),$$

$$k \in [1, l];$$

$$f(z_{l+k}^1) = f(x_{2k, s_{2k}^1}),$$

$$f(z_{\beta+k}^2) = f(x_{2k-1,1}^2), \quad (12)$$

$$k \in [1, \beta];$$

$$f(z_{l+\beta+k}^1) = f(y_{2k, t_{2k}^2}),$$

$$f(z_{l+k}^2) = f(y_{2(l+k)-1,1}^1),$$

$$k \in [1, \beta - l].$$

The above formula (12) enables us to obtain $f(z_i^r z_j^r) = f(z_i^r) + f(z_j^r) \in F'$ with $i \in [1, l]$, $j \in [l+1, p]$, and $r = 1, 2$. Thereby, we have shown

$$f(E(T_r)) = F'. \quad (13)$$

After performing a CA-operation on G_k^r and T_r having labelling f for $k \in [1, p]$, then we obtain a new graph S_r with

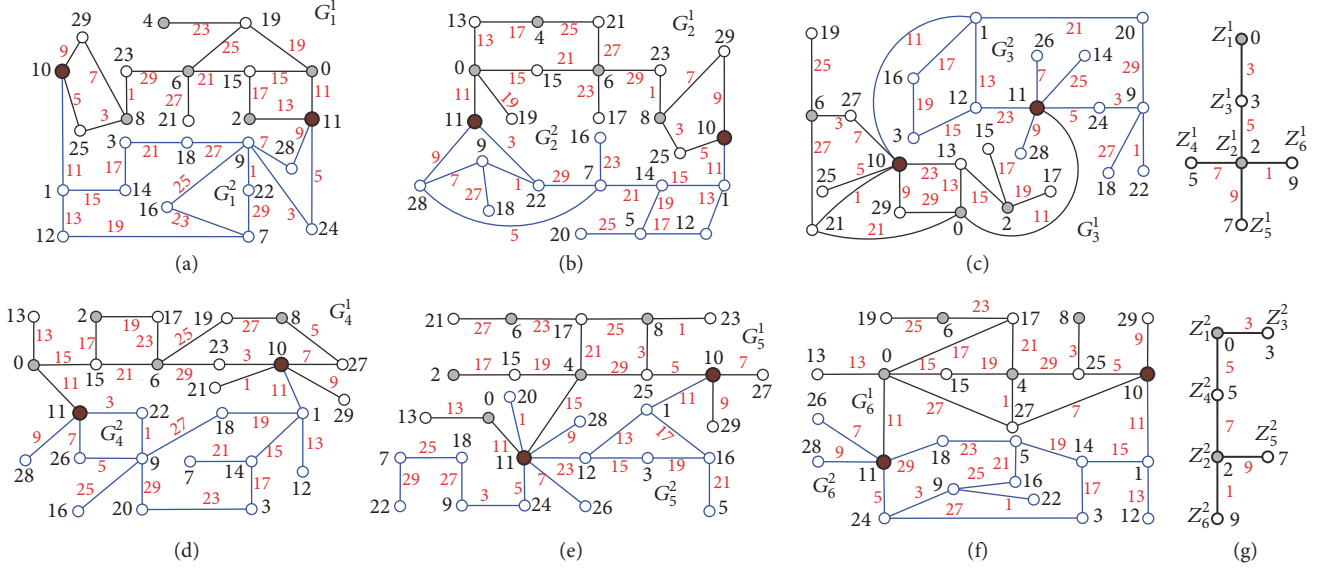


FIGURE 12: Six So-TOE-graphs G_k with $k \in [1, 6]$ and two base-trees T_1 and T_2 described in the proof of Case 2 of Theorem 8.

$r = 1, 2$. Now, we overlap the vertex y_{p,t_p}^1 with the vertex $y_{1,1}^2$ into one vertex $w'_1 = y_{p,t_p}^1 \circ y_{1,1}^2$ and overlap the vertex $y_{1,1}^1$ with the vertex y_{p,t_p}^2 into one vertex $w'_2 = y_{1,1}^1 \circ y_{p,t_p}^2$ (i.e., do a 2-identification operation) in order to obtain $G' = \odot_2 \langle S_1, S_2 \rangle$. Furthermore, by Definitions 2 and 4 and formulae (9)–(13), the labelling f is a So-TOE-labelling of G' , which implies that G' is a So-TOE-graph.

Hence, the proof of Case 2 is finished, and illustrating this case is given in Figures 12 and 13.

The proof of the theorem is complete. \square

3. Conclusion and Further Research

There are new Topsnut-GPWs having twin odd-elegant labellings introduced here. We define the twin odd-elegant labelling and investigate the 2-identification graph $G = \odot_2 \langle G_1, G_2 \rangle$, called twin odd-elegant graph. We think that finding all possible TOE-matching pairs (G, H) defined in Definition 4 may be interesting for a given TOE-graph G with an odd-elegant labelling g . Let

$$\begin{aligned} M_{\text{TOE}}(G, g) \\ = \{H : (G, H) \text{ is a TOE-matching pair}\} \end{aligned} \quad (14)$$

be the set of all TOE-associated graphs H , so, we have a TOE-book $B(G, g) = \bigcup_{H \in M_{\text{TOE}}(G, g)} \odot_2 \langle G, H \rangle$ with book-back G and book-pages $H \in M_{\text{TOE}}(G, g)$.

We should pay attention to the following problems:

(i) Since $G = \odot_2 \langle G, H \rangle \cap \odot_2 \langle G, H' \rangle$ for any pair of book-pages $H, H' \in M_{\text{TOE}}(G, g)$, does $V(G) = \bigcup_{H \in M_{\text{TOE}}(G, g)} (V(G) \cap V(H))$?

(ii) Suppose that G has m pairwise different odd-elegant labellings g_1, g_2, \dots, g_m . Find some possible relationships among TOE-books $B(G, g_i)$ with $i \in [1, m]$.

For the future researching work on Topsnut-GPWs, we propose the following.

Conjecture 9. Let each $\odot_2 \langle G_k^1, G_k^2 \rangle$ be a TOE-graph for $k \in [1, m]$ with $m \geq 2$. The 2-identification graph $G = \odot_2 \langle H_1, H_2 \rangle$ obtained by the edge-series operation (resp., the base-pasted operation) admits a TOE-labelling r .

Conjecture 10. Every simple and connected TOE-graph admits an odd-elegant labelling.

Conjecture 11. Each connected graph is the TOE-source graph of a certain TOE-graph.

A more interesting problem is to design super Topsnut-GPWs such that each super Topsnut-GPW will not be deciphered by attacks of nonquantum computers, since (i) our methods introduced here can construct quickly large scale of Topsnut-GPWs having hundreds vertices; (ii) the space of the Topsnut-GPWs given in Theorem 8 is quite tremendous; (iii) the 2-identification graphs $\odot_2 \langle H_1, H_2 \rangle$ of Theorem 7 and $\odot_2 \langle S_1, S_2 \rangle$ of Theorem 8 are the compound type of Topsnut-GPWs based on smaller scale of Topsnut-GPWs $G_k = \odot_2 \langle G_k^1, G_k^2 \rangle$ with $k \in [1, m]$, and they induce the TOE-books $B(H_1, f)$ and $B(S_1, g)$; it may be guessed that there is no polynomial algorithm for determining the TOE-books; and (iv) no polynomial algorithm was reported for finding all odd-elegant labellings of a given graph.

Thereby, we hope to discover such super Topsnut-GPWs which can be used in the era of quantum information.

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.

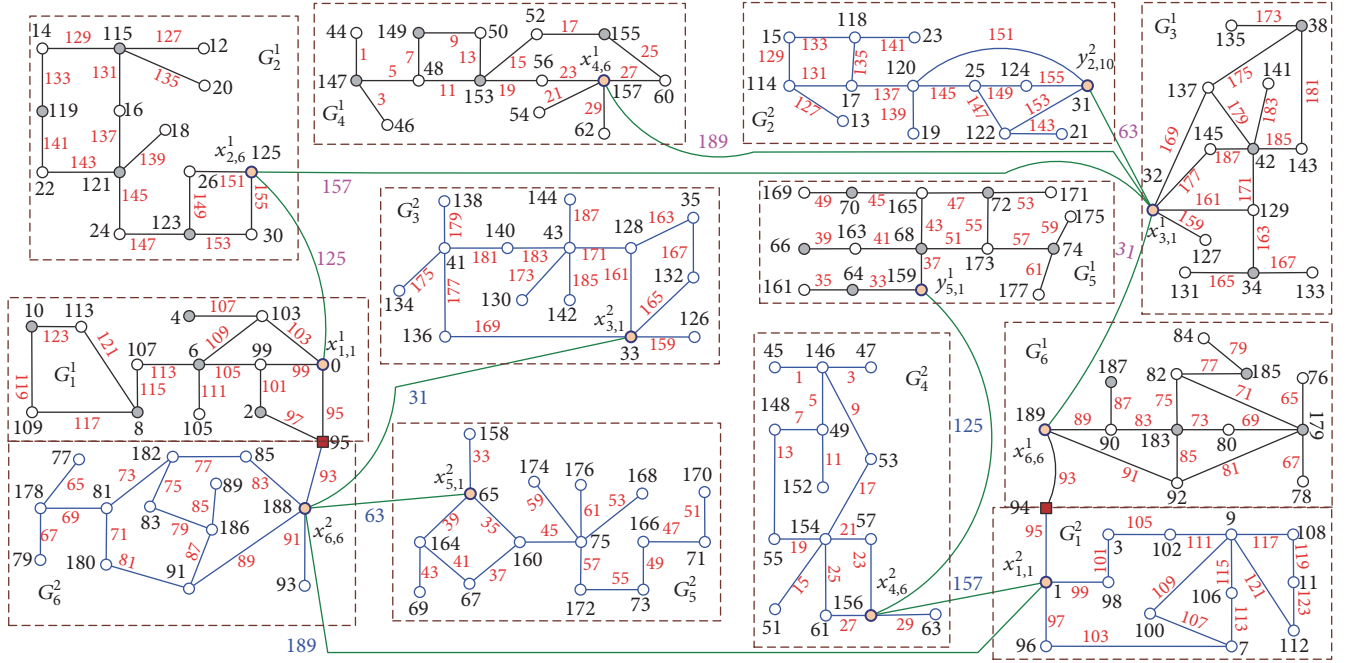


FIGURE 13: A So-TOE-graph $\odot_2\langle S_1, S_2 \rangle$ made by the graphs shown in Figure 12 for understanding the proof of Case 2 of Theorem 8.

Acknowledgments

This work was supported by the National Key R&D Program of China (no. 2016YFB0800700) and the National Natural Science Foundation of China (nos. 61572046, 61502012, 61672050, 61672052, 61363060, and 61662066).

References

- [1] X. Suo, Y. Zhu, and G. S. Owen, "Graphical passwords: a survey," in *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC '05)*, pp. 463–472, Tucson, Ariz, USA, December 2005.
- [2] R. Biddle, S. Chiasson, and P. C. Van Oorschot, "Graphical passwords: learning from the first twelve years," *ACM Computing Surveys*, vol. 44, no. 4, Article 19:1-41. Technical Report TR-09-09, School of Computer Science, Carleton University, Ottawa, Canada. 2009. (25 pages, 145 reference papers), 2012.
- [3] H. Gao, W. Jia, F. Ye, and L. Ma, "A survey on the use of graphical passwords in security," *Journal of Software*, vol. 8, no. 7, pp. 320–329, 2013.
- [4] "QR Code Essentials". Denso ADC. 2011. Retrieved 12 March 2013.
- [5] "QR Code features". Denso-Wave. Archived from the original on 2013-01-29. Retrieved 3 October 2011.
- [6] X. Suo, "A Study of Graphical Password for Mobile Devices," in *Mobile Computing, Applications, and Services*, vol. 130 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 202–214, Springer International Publishing, Cham, 2014.
- [7] B. Yao, H. Sun, X. Zhang, H. Wang, J. Li, and G. Yan, "Graph theory towards designing graphical passwords for mobile devices," in *Proceedings of the 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1640–1644, Chengdu, China, December 2017.
- [8] H. Wang, J. Xu, and B. Yao, "Exploring new cryptographical construction of complex network data," in *Proceedings of the 1st IEEE International Conference on Data Science in Cyberspace, DSC 2016*, pp. 155–160, June 2016.
- [9] H. Wang, J. Xu, and B. Yao, "The key-models and their lock-models for designing new labellings of networks," in *Proceedings of the 2016 IEEE Advanced Information Management, Communication, Electronic and Automation Control Conference, IMCEC 2016*, pp. 565–568, October 2016.
- [10] H. Wang, J. Xu, and B. Yao, "Twin Odd-graceful Trees Towards Information Security," in *Proceedings of the 7th International Congress of Information and Communication Technology, ICICT 2017*, pp. 15–20, Elsevier Science Publishers B. V., February 2017.
- [11] F. Harary and E. M. Palmer, *Graphical enumeration*, Academic Press, 1973.
- [12] J. A. Gallian, "A Dynamic Survey of Graph Labeling," *The Electronic Journal of Combinatorics*, vol. 17, DS6. (440 pages, 2265 reference papers), 2013.
- [13] X. Zhou, B. Yao, and X. Chen, "Every lobster is odd-elegant," *Information Processing Letters*, vol. 113, no. 1-2, pp. 30–33, 2013.
- [14] J. A. Bondy and U. S. R. Murty, *Graph Theory*, Springer, London, UK, 2008.
- [15] B. Yao, H. Cheng, M. Yao, and M. Zhao, "A note on strongly graceful trees," *Ars Combinatoria*, vol. 92, pp. 155–169, 2009.

Research Article

Analysis on Influential Functions in the Weighted Software Network

Haitao He,^{1,2,3} Chun Shan ,⁴ Xiangmin Tian ,^{1,2,3} Yalei Wei,^{1,2,3} and Guoyan Huang^{1,2,3}

¹College of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei 066004, China

²The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao, Hebei 066004, China

³The Key Laboratory for Software Engineering of Hebei Province, Qinhuangdao, Hebei 066004, China

⁴Beijing Key Laboratory of Software Security Engineering Technique, Beijing Institute of Technology, 5 South Zhongguancun Street, Haidian District, Beijing 100081, China

Correspondence should be addressed to Chun Shan; sherryshan@bit.edu.cn

Received 24 October 2017; Revised 5 January 2018; Accepted 4 February 2018; Published 1 April 2018

Academic Editor: Zheng Yan

Copyright © 2018 Haitao He et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Identifying influential nodes is important for software in terms of understanding the design patterns and controlling the development and the maintenance process. However, there are no efficient methods to discover them so far. Based on the invoking dependency relationships between the nodes, this paper proposes a novel approach to define the node importance for mining the influential software nodes. First, according to the multiple execution information, we construct a weighted software network (WSN) to denote the software execution dependency structure. Second, considering the invoking times and outdegree about software nodes, we improve the method PageRank and put forward the targeted algorithm FunctionRank to evaluate the node importance (NI) in weighted software network. It has higher influence when the node has larger value of NI. Finally, comparing the NI of nodes, we can obtain the most influential nodes in the software network. In addition, the experimental results show that the proposed approach has good performance in identifying the influential nodes.

1. Introduction

Measuring accurately the importance of the node in the software networks is the premise to improve the security and robustness of software [1, 2]. Moreover, with the development of the software, measuring the importance of nodes in the network has practical significance for defending and protecting the influential nodes in the software network [3], if these nodes are suffered by deliberate attacks, maybe cascading failure occurs [4, 5]. Accordingly, how to mine the potential characteristics of software to control the evolution process of the software structure has become a hot spot for researching [6–9].

Many researchers introduced the idea of complex networks to the field of software structure and abstracted software to a network from different granularity point [10]. With the network structure, many potential characteristics can be discovered directly. Ma et al. [11] abstracted interaction relationship between packages into a software network, and

they defined functions in package as nodes and dependencies among functions as edges. Wang et al. [12] proposed an approach to study the evolution of special software kernel components, which adopted the theory of complex networks. They also proposed a generic method to find major structural changes that happened during the evolution of software systems. Li et al. [13] proposed a modular attachment mechanism of software network evolution. Their approach treated object-oriented software system as a modular network, which was more realistic. A new definition of asymmetric probabilities was given to acquire links in directed networks when new nodes attached to the existing network. With the directed network, both of the “scale-free” and “small-world” properties were verified to be present in the software network. In [14], David proposed a method to simplify the complexity of the software network. With the method, some valuable characteristics in the network could be obtained easily. From the researches above, the complex network was proved to be applicative in the software engineering and it brought

us a new perspective to research the software structure. However, these methods of modeling the software mentioned above were based on the static structure of the source code. The execution characteristic during the software running process was neglected in these methods. For the software, most of the characteristics are exhibited during the execution process.

The characteristics of the software execution can help us to understand the software better. It is obvious that the node is an important part of the network and it has enormous influence on the stability, reliability, and robustness of the network [15]. For a software network, the software function plays a critical role in the stability and robustness of the software during the execution process. In the structure of the software, the functions carry most of the feature characteristics and topology information and they can affect each other. In most cases, the fault of a function is not only caused by itself but also infected by the other functions. Recently, the importance of the node in the network was defined from different aspects. Bhattacharya et al. [2] defined a measure to evaluate relative importance of the nodes in software graph. By the value of the betweenness and the clustering coefficient, Zhang et al. [16] measured the importance of each node to analyze the influence of each node to the entire network. According to the propagation field of the classes, Li et al. [17] put forward an indicator to measure the importance classes in the software network at class level. Based on the value of the indegree and the outdegree of each node, Wang and Lü [18] proposed a method to mine the influential nodes. With the method, they proved that the fault appeared with a large probability in those nodes with large degree value. In the researches above, the node was proved to play a key role to analyze the network. However, the node was regarded as an individual unit, as well as the relationship between the node and the entire network was ignored. In practical application, the network should be considered as a whole, in which the nodes can interact with each other.

Considering the above-mentioned shortcomings, the dependency relationship between the function nodes, and the absence of efficient analysis methods, we construct the WSN to show the software structure according to the information of multiple execution. Based on the dependency relationship between the function nodes, we present a targeted method FunctionRank to evaluate the importance of the software nodes. With the analysis result of each node, we rank the influence of each node to mine the top- k nodes. These function nodes have played an important part in ensuring software reliability and stability. So they should be paid more attention in the process of software updating and software maintenance.

The primary contributions of this paper can be summarized as follows:

- (i) A novel method is proposed to construct weighted software network (WSN). So we make the understanding and recognition of software structure more accurate.

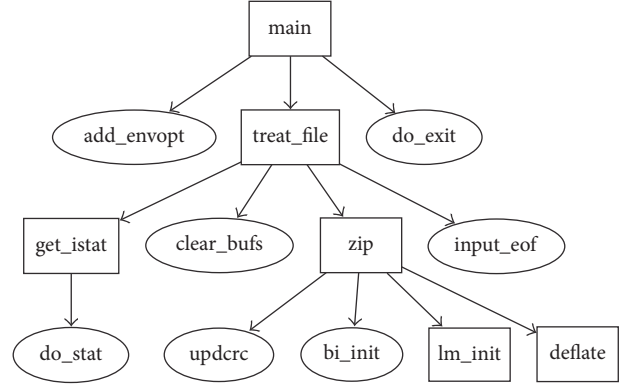


FIGURE 1: The software network.

- (ii) A measurement node importance (NI) is put forward to evaluate the importance of each node in the network.
- (iii) The IC (independent cascade) model as an attack model is used to evaluate the influential functions for software system.
- (iv) The proposed algorithm is an effective method for security measurements of cybernetwork and provides basis for software security and reliability improvement.

The rest of this paper is organized as follows. The construction process of the weighted software network (WSN) is described in Section 2. The node importance of each function node is given definition in Section 3. Then, in Section 4, the method FunctionRank is given to mine the most influential nodes. In Section 5, the performances of the proposed algorithm are showed by experiments. Finally, conclusions and future works of the paper are presented in Section 6.

2. Definitions of Weighted Software Network

Complex networks are suitable to show the invoking relationships between the software functions. Based on the information of the multiple execution processes, we define the software execution dependency structure with a directed-weighted network.

2.1. Software Network. In this section, according to the multiple execution information, we define a software network to demonstrate the software execution dependency structure. Figure 1 shows a real example of software network.

Where each node represents a software function and each edge is the invoking relationship between the functions. In the software network, most of the characteristics can be exhibited during the software execution process.

2.2. Weighted Software Network. Next, in order to guarantee the completeness of the experimental data and make the understanding and recognition of software structure more accurate, we define a weighted software network. Compared with the software network, we consider invoking times

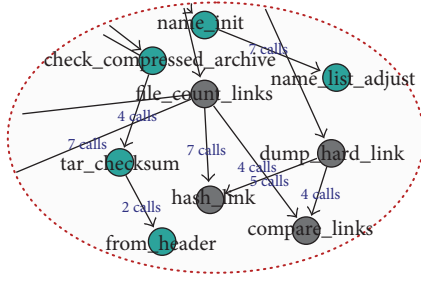


FIGURE 2: Weighted software network.

between the software functions in multiple execution processes as the weight of each edge. The weighted software network is suitable to demonstrate the complex invoking relationships between the software functions. The definition of weighted software network (WSN) is given as follows:

$$\text{WSN} = \{\text{Node}, \text{Edge}, W\}, \quad \text{Node}, \text{Edge} \in \text{software}. \quad (1)$$

Figure 2 shows a weighted software network, where Node is a software functions set and Edge is an invoking relationship set between the software functions. W_e that stands for the weight of edge e is calculated by the following formula:

$$W_e = \sum_j w_e, \quad e \in \text{Edge}, \quad (2)$$

where j is the times of the trials with different experiment cases. w_e is a value of 1 or 0. If the edge e of one calling relationship appears in an execution trace, no matter how many times of it, let w_e be 1; otherwise it is 0.

Figure 3 presents a simple process of the WSN established. As shown in Figure 3(a), s_i ($1 \leq i \leq 5$) is a function invoking trace in one-time execution of the software. The trace s_i contains a series of function calling relationships which can reflect the software execution process. Figure 3(b) shows a structure of WSN, in which the node and the edge of the network are defined as the function and the calling relationship between the functions appearing in $s_1 \sim s_5$ in Figure 3(a), the weight of edges represents the number of each calling process executed in the 5 times' execution, and the times of a calling relationship in some execution processes were ignored.

Based on multiple execution information under the different experimental cases of software, we guarantee the completeness of the experimental data. Function nodes which have appeared during the software multiple execution processes are considered as a set of nodes of the network structure, calling relationship between the software functions is considered as a set of edges, the weight of the edge, we consider the weight c to stand for the edge appearing c times in the N execution traces of the software, and we ignore the times appearing in an execution trace s_i . In this way, WSN is built.

3. Node Importance

According to the complex invoking relationships for software system, we show the most common topology structures of

the weighted software network in Figure 4 to explain the importance of the function node.

Definition 1 (IN (indegree nodes)). For a node vi , IN is a set of functions which call node vi directly. The IN of node vi is gotten by only one call step.

As shown in Figure 4(a), $\text{IN}(A) = \{B, C\}$. The influence of node vi is based on $\text{IN}(vi)$ which call vi directly.

Definition 2 (ON (outdegree nodes)). For a node vi , ON is a set of functions which are called by node vi directly. The number of $\text{ON}(vi)$ is vi 's outdegree, CO.

As shown in Figure 4(a), $\text{ON}(A) = \{B, C, D\}$ and $\text{CO}(A) = 3$.

Definition 3 (TN (terminal nodes)). The nodes that have no outdegree and have no contribution to the influence of other nodes are defined as terminal nodes.

As shown in Figure 4(b), C is a terminal node.

Definition 4 (LTN (loop terminal nodes)). The nodes that only have an outlink to their own are defined as loop terminal nodes.

As shown in Figure 4(c), C only has an outlink to its own. So C is a loop terminal node.

Definition 5 (OD (output degree)). The weight sum of each edge for a node vi to its outdegree nodes, $\text{OD}(vi)$, is named as output degree of the node vi .

In Figure 4(a), the weight of each edge for A to $\text{ON}(A)$ is 2, 2, and 5, respectively. $\text{OD}(A)$ is the sum of these weights, namely, A 's output degree.

Definition 6 (WC (weighted contribution)). The ratio of the weight for node vi to node vj and vi 's output degree, $\text{WC}(vj)$, is the weighted contribution of vi to vj .

In Figure 4(a), the weight of A to B is 2. The weighted contribution of A to B is given as follows:

$$\text{WC}(B) = \frac{2}{\text{OD}(A)} = \frac{2}{2+2+5} = \frac{2}{9}. \quad (3)$$

Based on the above definitions, the node importance (NI) of node vj is given as follows:

$$\text{NI}(V_j) = \left[\alpha + (1 - \alpha) \sum_{\text{IN}(V_j)} \text{WC}(V_j) * \text{NI}(V_i) \right] \cdot (\text{CO}(V_j) + 1), \quad (4)$$

where α is the certain probability of calling a random node for LTN, and the probability of invoking each node is the same. It is set as 0.15 with experimental verification.

4. Important Nodes Mining

In this section, we first provide an algorithm outdegree nodes to get the outdegree node list of all nodes, according to the outdegree nodes of each node in the software network, and then we provide another algorithm FunctionRank

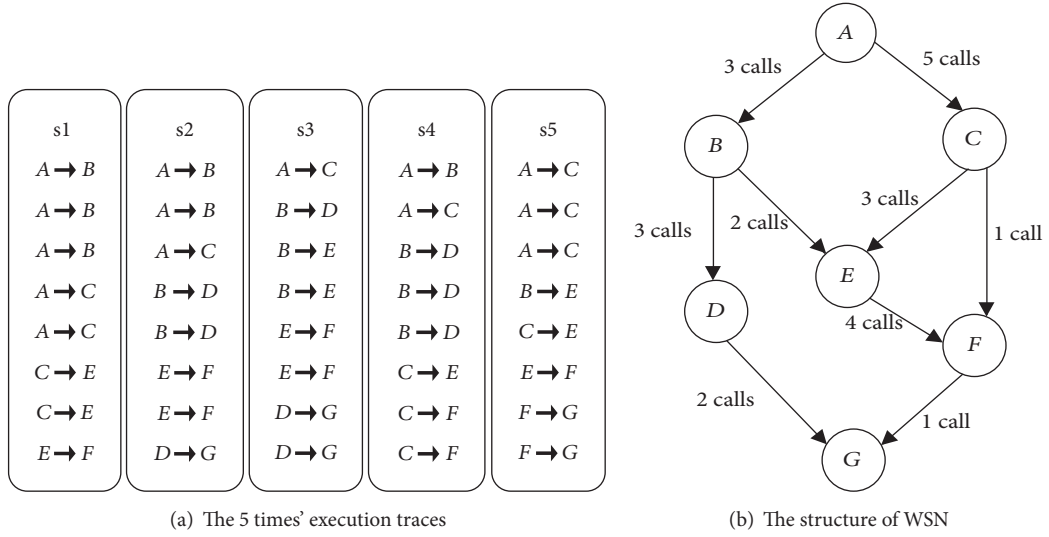


FIGURE 3: The constructing process of WSN.

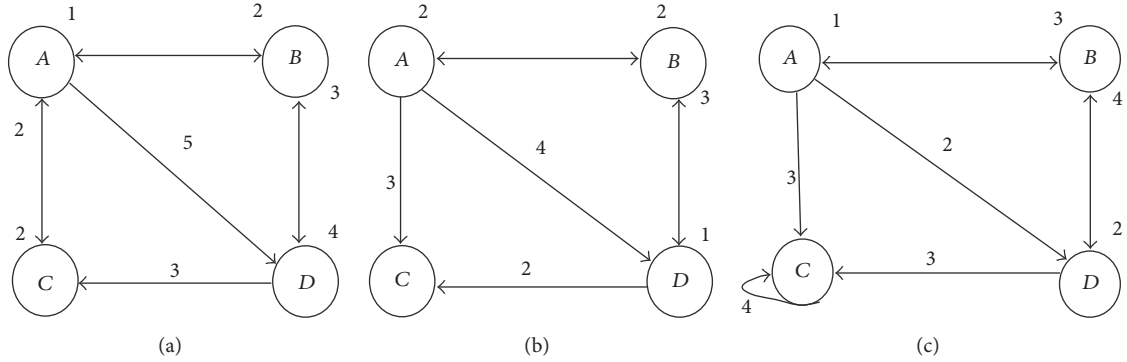


FIGURE 4: Three kinds of topology structures.

Input: node set V , edge set E
Output: childStr //the out-degree node list of all nodes

```

(01) for (each  $vi \in V$ ) {
(02)   for ( $\langle vsi, vej \rangle \in E$ ) {
(03)     if ( $vi = vsi$ )
(04)       childStr += " " + vej;
(05)   }
(06)   print ( $vi$  + childStr);
(07) }
  
```

ALGORITHM 1: Outdegree nodes.

to calculate NI of each node. In the method Function-Rank, we evaluate the importance of nodes iteratively (see Algorithms 1 and 2).

As shown in algorithm outdegree nodes, for each node in set V we traverse the edges in set E in line (1) and line (2). We define the nodes of an edge as start node vsi and end node vej , respectively. In line (3) to line (4) we add the end node

vej of an edge to the childStr of node vi , when node vi equals the start node vsi of the edge. Finally, we print the childStr of vi in line (6).

We evaluate the importance of each node in the network by an iterative process, as shown in Algorithm 2. In line (1), we initialise $NI(vi)$ as importance of nodes and α as the certain probability to call a random node, respectively. Line (2) to (19) is the iterative process to compute the importance coming from outdegree of the current node and other nodes which call the current node. The computational formula of node importance (NI) is given in line (18); it has higher influence when the node has larger value of NI. Ultimately, the importance for a node (NI) is obtained when error of current NI value and previous NI value is less than a given threshold for all nodes.

With the measuring results obtained from Algorithm 2, we choose the top- k nodes as the influential nodes for the software network. In Algorithm 3, we illustrate the process of top- k nodes (KN).

In Algorithm 3, we initialise list as the measurement list for all the nodes in line (1). Lines (2) to (4) are a looping

Input: node vi , childStr (vi)
Output: the NI of node vi //evaluate the importance of nodes
 Process:
 (01) **Initialize** $NI(vi) = 1f, \alpha = 0.15f$
 (02) **if** (childStr[vi] != null) {
 (03) outdegree = childStr.size ();
 (04) **for** (each $vj \in$ childStr [vi]) {
 (05) **if** (vj is equal vi) {
 (06) outdegree - -;
 (07) **else**{
 (08) weighMap.put (vj ,weight ($vi \rightarrow vj$));
 (09) weigh += weight ($vi \rightarrow vj$);
 (10) }
 (11) }
 (12) **for** (each $vj \in$ childStr [vi]) {
 (13) **if** (vj is not equal vi) {
 (14) tempNI (vj) += NI (vi) * weighMap.get (vj)/weigh;
 (15) }
 (16) }
 (17) }
 (18) tempNI (vi) = ($\alpha + (1 - \alpha) * tempNI (vi)$) * (outdegree (vi) + 1);
 (19) NI (vi) = tempNI (vi);

ALGORITHM 2: FunctionRank.

Input: node set N , NI of each node
Output: the top - k influential nodes
 Process:
 (01) **Initialize** list //store the importance of nodes
 (02) **for** (each node $v \in N$)
 (03) list.add (NI(v));
 (04) **end for**
 (05) Collections.sort (list);
 (06) Collections.reverse (list);
 (07) print list.get (k)

ALGORITHM 3: Top- k nodes (KN).

process to store the NI value for each node. The sorting process is given in line (5) and line (6), the top- k nodes are chosen from the list in line (7).

5. Experimental Analysis

A series of experiments were conducted to compare the performance of the proposed algorithm (named as FunctionRank) with different parameter values. They were implemented in JDK1.6.0 and executed on a PC with 3.30 GHz CPU and 5 GB memory.

5.1. Experimental Datasets. Firstly, several dynamic software datasets are used to evaluate the performance of the algorithms. The classical software is obtained from the open-source community. These software programs are coded in C or C++, including program software tar and cflow.

In the experiment, we chose different versions of tar and cflow, respectively, for experiment. tar is a decompression software for Linux, and cflow is an analysis tool for C program to extract the relationship of function calls (download from the open-source software library: <https://sourceforge.net>).

5.2. Evaluation on the FunctionRank. We run the algorithm on each version of tar and cflow. By the algorithm FunctionRank, we calculate the NI of each function node. Here we mine top-10 nodes in each version about software tar and cflow. It is shown in Tables 1 and 2, respectively.

As it is shown in Table 1, for versions tar-1.21 and tar-1.23, the NI of the top-10 are almost the same. The reason is that the difference between the three versions only reflects the number of function calls. In other words, there is no change of the component function of these two versions. In the latest three versions, developers changed the logical contents of some functions or insert new functions into the software to enrich the features of software; on the other hand, the software was simplified or some features were removed to improve the robustness, which results in the ranking variation. For example, in the prior versions node `_gnu_flush_read` ranked 2nd or 3rd but it ranked 7th and 8th in versions tar-1.25, tar-1.27, and tar-1.28. Table 2 shows the top-10 influential functions of software cflow in different versions. The ranking of some functions in each version of cflow varies but with little range. For example, function `print_symbol's` ranking ranges from 1 to 2. So we can make a prediction that it may still be more influential than most others in the next new version. Meanwhile, there is no function `alloc_cons` for the latest versions cflow-1.3 and cflow-1.4 results in the ranking variation. In other words, there is change of the component function of these two versions.

TABLE 1: Top-10 influential nodes for each version of software tar.

Function name	V1.21	V1.23	V1.25	V1.27	V1.28
	Rank/value	Rank/value	Rank/value	Rank/value	Rank/value
dump_file0	1/3.020	1/2.808	1/2.804	1/2.604	1/2.603
flush_archive	2/2.245	5/1.969	4/1.949	4/1.943	4/1.939
_gnu_flush_read	3/2.212	2/2.130	7/1.593	7/1.592	8/1.591
update_archive	4/2.139	4/2.108	2/2.270	2/2.270	2/2.246
to_chars	5/2.124	3/2.124	3/2.127	3/2.124	3/2.124
_gnu_flush_write	6/1.865	6/1.738	12/1.152	13/1.151	14/1.149
start_header	7/1.672	7/1.673	6/1.698	6/1.674	7/1.674
_open_archive	8/1.589	8/1.581	8/1.578	8/1.578	6/1.803
dump_regular_file	9/1.460	9/1.468	10/1.305	9/1.478	9/1.478
find_next_block	10/1.317	13/1.102	13/1.105	15/1.100	15/1.097

TABLE 2: Top-10 influential nodes for each version of software cflow.

Function name	cflow-1.0	cflow-1.1	cflow-1.2	cflow-1.3	cflow-1.4
	Rank/value	Rank/value	Rank/value	Rank/value	Rank/val
print_symbol	1/4.195	1/4.195	1/4.195	1/4.196	2/4.104
yylex	2/3.822	2/3.822	2/3.822	2/3.823	3/4.074
nexttoken	3/2.985	3/2.985	3/2.985	3/2.987	4/3.334
parse_variable_declaration	4/2.309	4/2.309	4/2.309	5/2.310	10/2.310
parse_dcl	5/2.200	5/2.200	5/2.200	7/2.200	11/2.200
gnu_output_handler	6/2.068	6/2.068	6/2.068	8/2.068	12/2.011
yyrestart	7/2.042	7/2.042	7/2.042	9/2.050	8/2.593
alloc_cons	8/2.037	9/42.010	8/2.010	—	—
tree_output	9/1.985	10/1.985	9/1.985	10/1.993	5/3.101
lookup	10/1.906	11/1.892	11/1.880	11/1.831	14/1.822

In addition, the number of nodes which have high NI is rather small in each version. These high value nodes have taken a great part in ensuring software reliability and stability. It means that there are little functions that should be paid more attention in software updating and software maintenance. We calculate the count for different range of NI values. The results of software tar and cflow are shown in Figures 5 and 6, respectively.

As we can see in Figure 5, most of nodes are ordinary functions. We would not pay more attention to them. Meanwhile, a handful of nodes that have high NI should be paid more attention. They play important roles in the process of software updating and software maintenance. For cflow, the number of nodes in each scope is shown in Figure 6. It has the same characteristic with tar. The number of nodes with high NI is much less than that of low NI . By paying more attention to these influential nodes in future versions, we can improve software reliability and stability. Thereby we can greatly reduce the amount of work and improve work efficiency.

At the same time, NI of the same ranking nodes within different versions has slight wave, as shown in Figures 7 and 8.

As it is shown in Figure 7, the NI distribution of software tar is similar extremely in the six versions. With the increasing of node ranking, the NI of each node shows a decrease trend.

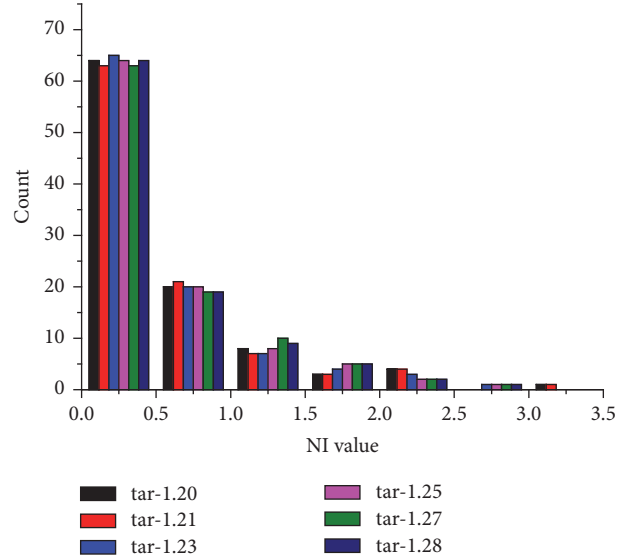


FIGURE 5: Number of nodes in value scope of tar.

As the lower rank, the value shows a trend of increase. The higher NI ranges from 0.7 to 3.0; most nodes' values are around 0.4. The development of versions follows the same

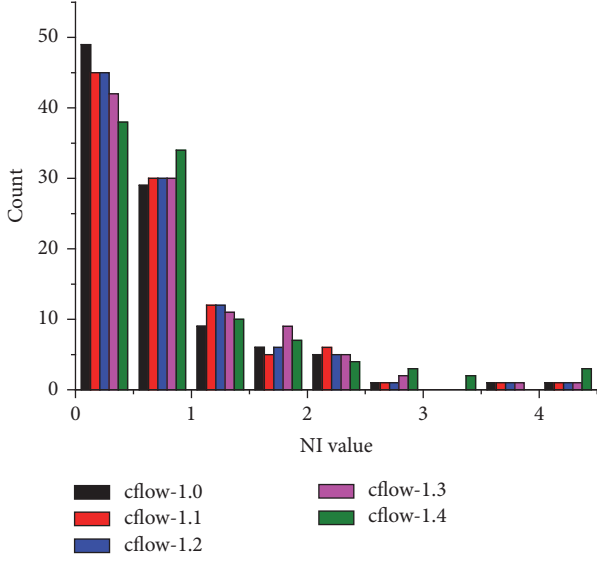


FIGURE 6: Number of nodes in value scope of cflow.

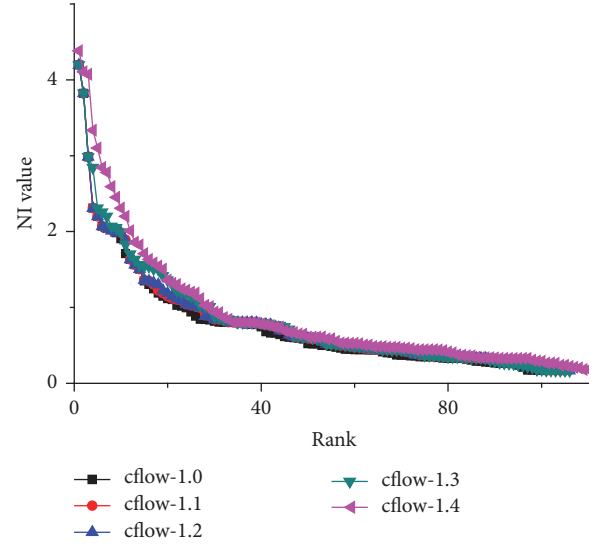


FIGURE 8: NI distribution of cflow.

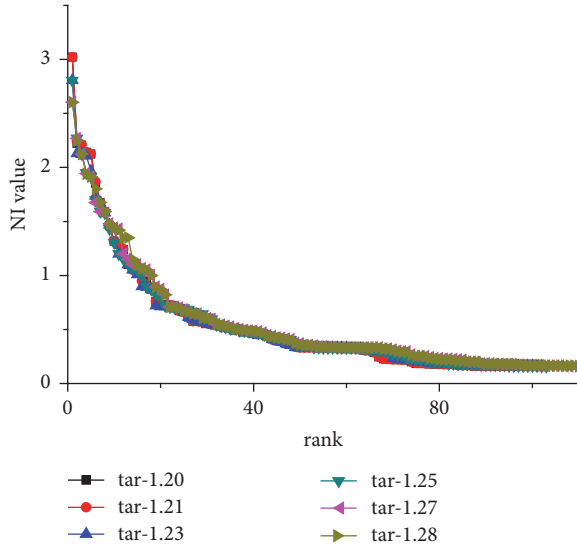


FIGURE 7: NI distribution of tar.

laws, the NI of a certain ranking remains stable and the NI distribution of different software versions is nearly the same. So, we can predict the future versions' trends on this. Meanwhile, Figure 8 shows the NI distribution of software cflow, the higher NI ranges from 0.8 to 4.0, and most nodes' values are around 0.5. The curve of each version has the same tendency; namely, the NI distribution of software cflow follows the same trend.

5.3. Performance Evaluation. In the study of complex network, we often examine the effectiveness of a method [19, 20] through the analysis of spreading influence about top- k nodes. Therefore, this paper will introduce IC (independent cascade) model. The IC model derived from the SIR (Susceptible-Infected-Recovered) model, the SIR model is a theory about virus spreading and has to be researched widely

in complex networks, such as the marketing, advertising, early warning, and social stability. In software engineering, the similar algorithms were used to analyze the change impact [21] and error propagation [22].

The IC model is a probability model; when a node v is activated, it will attempt to activate its inactive outdegree nodes with probability p only once [23]. Whether node v can activate its neighbor nodes successfully, v is still active, but it has no influence later. The communication process is over when there are no influential active nodes in the network, while, in the actual execution process of software, the running fault can affect the other function running due to the invoking relationship. When running fault, all of the invoked functions would affect the normal execution of the parent function. So the faults can widely spread among the function nodes during the running process. So we take IC model as a software attack model to evaluate the effectiveness of our method. A software attack instance is shown in Figure 9.

We assume the node a and node b are attacked as Figure 9(a) shows, and then a and d will attack its inactive outdegree nodes with probability p only once, where b and c are attacked successfully by a ; meanwhile e and h are attacked successfully by d in Figure 9(b), next a and d have no aggressivity, and the nodes attacked by a and d can attack their inactive outdegree nodes with probability p in the same way. Finally, the number of attacked nodes represents the influence of original attacked nodes.

When calculating the influence of the top- k important nodes obtained by different methods, we will separately run IC model about 10 times and then consider the average of active nodes as the performance evaluation of the method.

The software key entities typically account for a small proportion and only account for one point five percent to two percent in the study of class size [24]. At the same time, it is not acceptable for the cost of checking most of the key entities. So an appropriate number of key entities is needed to be selected. By ranking all functions as descending order

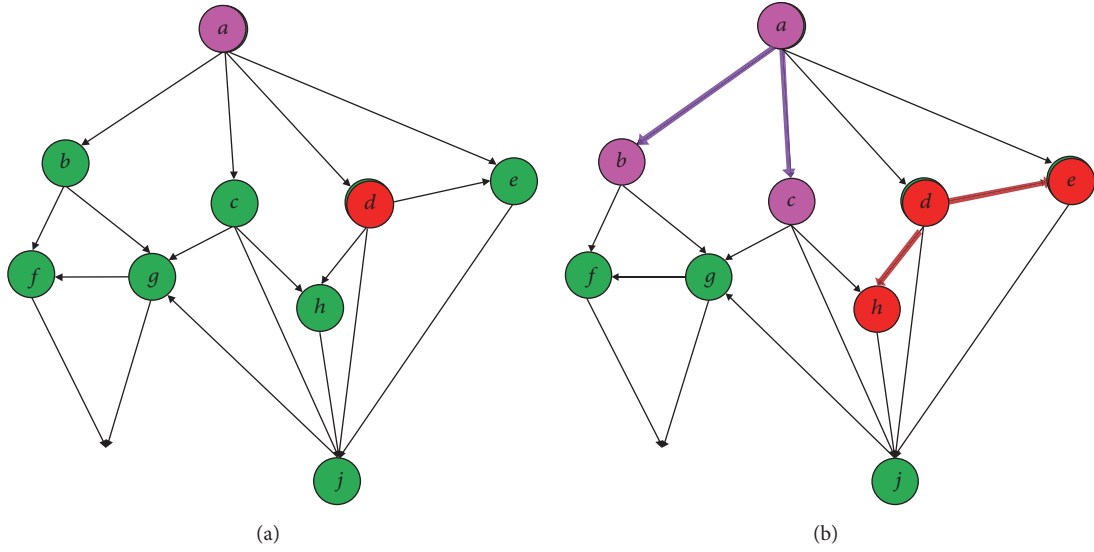


FIGURE 9: A software attack instance.

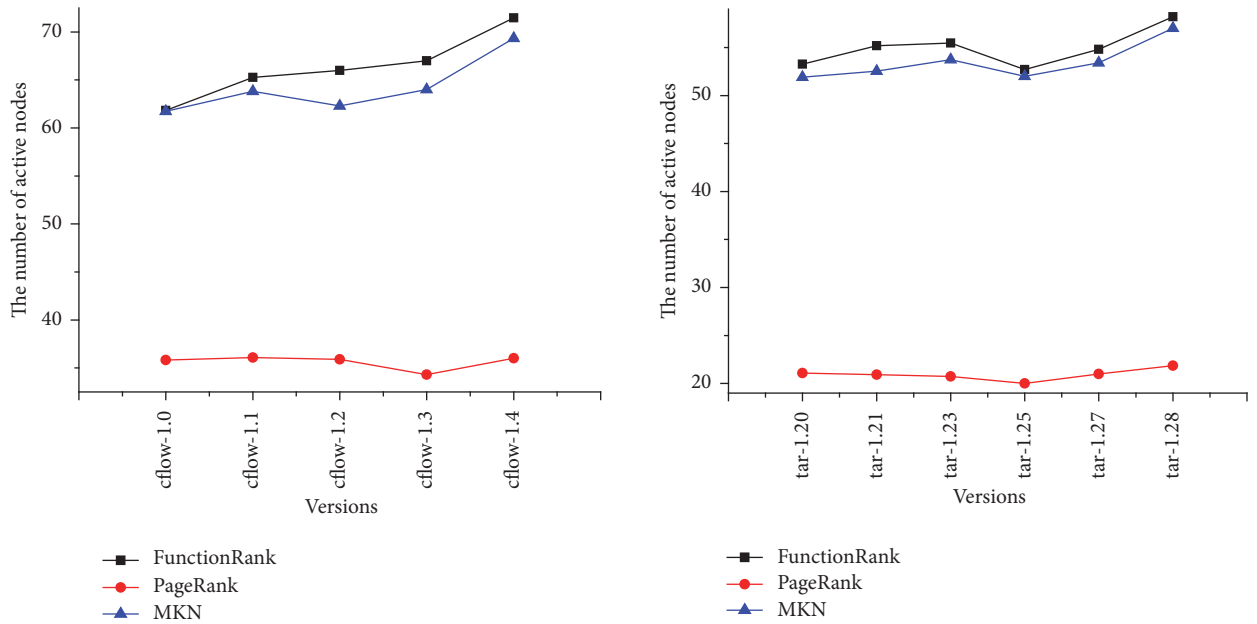


FIGURE 10: The curves of the number of active nodes.

according to the measurements, we chose little key functions for different systems: top 20 for tar and top 30 for cfow.

Figure 10 shows the average of active nodes for different software versions. In all the different versions of the software systems, key functions identified by NI can activate more nodes than that identified by the method PageRank and MKN [25] as Figure 10 shows. Visibly, compared with another two methods, NI is more effective in the identification of the key functions. The key functions play an important role in software system in terms of reducing the numbers of test data, detecting the vulnerabilities of software structure, and analyzing software reliability, and they should be paid more attention in the process of software updating and software

maintenance. Measuring accurately the importance of the node in the software networks is the premise to improve the security and robustness of software. Moreover, with the development of the software, measuring the importance of nodes in the network has practical significance for protecting the influential nodes from deliberate attacks in the software network.

6. Conclusions and Future Work

In order to understand and recognize software structure better, a novel method is proposed in this paper to mine the influential nodes in weighted software network. Firstly, taking

into account the invoking times, we construct a directed-weighted network structure to make the understanding and recognition of software structure more accurate. Then, a measurement of NI is put forward to evaluate the node importance, where we provide an idea of importing PageRank and WSN to Software engineering domain. Furthermore, we also consider the outdegree value as a key parameter to the node importance. The outdegree value can reflect the complexity of the node. Finally, the algorithm named FunctionRank is presented to calculate the NI and the change trends of nodes' importance are analyzed by different software versions. In addition, the experimental results show that the proposed feasible approach has good performance in identifying the influential software nodes.

Although the approach we proposed shows some feasibilities in identifying influence nodes in complex software network, the broad validity of our approach should be demonstrated further. Our future work is using more open-source software network to evaluate the validity to improve our approach.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Key R&D Program of China (2016YFB0800700), the National Natural Science Foundation of China under Grants no. 61472341, no. 61572420, and no. 61772449, the Natural Science Foundation of Hebei Province of China under Grants no. F2015203326 and no. F2016203330, and the Advanced Program of Postdoctoral Scientific Research under Grant no. B2017003005.

References

- [1] W.-F. Pan, B. Li, Y.-T. Ma, Y.-Y. Qin, and X.-Y. Zhou, "Measuring structural quality of object-oriented softwares via bug propagation analysis on weighted software networks," *Journal of Computer Science and Technology*, vol. 25, no. 6, pp. 1202–1213, 2010.
- [2] P. Bhattacharya, M. Iliofotou, I. Neamtii, and M. Faloutsos, "Graph-based analysis and prediction for software evolution," in *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*, pp. 419–429, IEEE, Zürich, Switzerland, June 2012.
- [3] D. Chen, L. Lü, M. Shang, Y. Zhang, and T. Zhou, "Identifying influential nodes in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 4, pp. 1777–1787, 2012.
- [4] E. Zio, L. R. Golea, and G. Sansavini, "Optimizing protections against cascades in network systems: a modified binary differential evolution algorithm," *Reliability Engineering and System Safety*, vol. 103, pp. 72–83, 2012.
- [5] S. M. Chen, X. Q. Zou, H. Lv, and Q. G. Xu, "Research on robustness of interdependent network for suppressing cascading failure," *Acta Physica Sinica*, vol. 63, no. 2, 2014.
- [6] M. Kitsak, L. K. Gallos, S. Havlin et al., "Identification of influential spreaders in complex networks," *Nature Physics*, vol. 6, no. 11, pp. 888–893, 2010.
- [7] N. Masuda and H. Kori, "Dynamics-based centrality for directed networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 82, no. 5, Article ID 056107, 2010.
- [8] G. Huang, B. Zhang, R. Ren, and J. Ren, "An algorithm to find critical execution paths of software based on complex network," *International Journal of Modern Physics C*, vol. 26, no. 9, Article ID 1550101, 2015.
- [9] G. Huang, P. Zhang, B. Zhang, T. Yin, and J. Ren, "The optimal community detection of software based on complex networks," *International Journal of Modern Physics C*, vol. 27, no. 8, Article ID 1650085, 2016.
- [10] S. M. Chen, X. Q. Zou, H. Lu, and Q. G. Xu, "Research on robustness of interdependent network for suppressing cascading failure," *Acta Physica Sinica*, 2013.
- [11] J. Ma, D. Zeng, and H. Zhao, "Modeling the growth of complex software function dependency networks," *Information Systems Frontiers*, vol. 14, no. 2, pp. 301–315, 2012.
- [12] L. Wang, P. Yu, Z. Wang, C. Yang, and Q. Ye, "On the evolution of Linux kernels: a complex network perspective," *Journal of Software: Evolution and Process*, vol. 25, no. 5, pp. 439–458, 2013.
- [13] H. Li, H. Zhao, W. Cai, J.-Q. Xu, and J. Ai, "A modular attachment mechanism for software network evolution," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 9, pp. 2025–2037, 2013.
- [14] F. Thung, D. Lo, M. H. Osman, and M. R. V. Chaudron, "Condensing class diagrams by analyzing design and network metrics using optimistic classification," in *Proceedings of the 22nd International Conference on Program Comprehension, ICPC '14*, pp. 110–121, ACM, Hyderabad, India, June 2014.
- [15] K. Zhuang, H. Shen, and H. Zhang, "User spread influence measurement in microblog," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3169–3185, 2017.
- [16] X. Zhang, G. Zhao, T. Lv, Y. Yin, and B. Zhang, "Analysis on Key Nodes Behavior for Complex Software Network," in *Information Computing and Applications*, vol. 7473 of *Lecture Notes in Computer Science*, pp. 59–66, Springer, Berlin, Germany, 2012.
- [17] D. W. Li, B. Li, P. He, and W. F. Pan, "Ranking the importance of classes via software structural analysis," in *Future Communication, Computing, Control and Management*, vol. 141 of *Lecture Notes in Electrical Engineering*, pp. 441–449, Springer, Berlin, Germany, 2012.
- [18] B.-Y. Wang and J.-H. Lü, "Software networks nodes impact analysis of complex software systems," *Journal of Software*, vol. 24, no. 12, pp. 2814–2829, 2013.
- [19] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani, "The role of the airline transportation network in the prediction and predictability of global epidemics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 7, pp. 2015–2020, 2006.
- [20] A. Garas, P. Argyrakis, C. Rozenblat, M. Tomassini, and S. Havlin, "Worldwide spreading of economic crisis," *New Journal of Physics*, vol. 12, Article ID 113043, 2010.
- [21] L. Zhang, G.-Q. Qian, and L. Li, "Software stability analysis based on change impact simulation," *Chinese Journal of Computers*, vol. 33, no. 3, pp. 440–451, 2010.
- [22] W. F. Pan and B. Li, "Software quality measurement based on error propagation analysis in software networks," *Journal of Central South University (Science and Technology)*, vol. 43, no. 11, pp. 4339–4347, 2012.

- [23] Y. Zhao, S. Li, and F. Jin, "Identification of influential nodes in social networks with community structure based on label propagation," *Neurocomputing*, vol. 210, pp. 34–44, 2016.
- [24] A. Zaidman and S. Demeyer, "Automatic identification of key classes in a software system using webmining techniques," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 20, no. 6, pp. 387–417, 2008.
- [25] G. Huang, P. Zhang, Y. Li, and J. Ren, "Mining the important nodes of software based on complex networks," *ICIC Express Letters*, vol. 9, no. 12, pp. 3263–3268, 2015.

Research Article

Security Feature Measurement for Frequent Dynamic Execution Paths in Software System

Qian Wang ^{1,2} **Jiadong Ren**^{1,2} **Xiaoli Yang**^{1,2} **Yongqiang Cheng** ³,
Darryl N. Davis³ and **Changzhen Hu**⁴

¹College of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei 066000, China

²Computer Virtual Technology and System Integration Laboratory of Hebei Province, Hebei 066000, China

³Computer Science, University of Hull, Hull HU6 7RX, UK

⁴Beijing Key Laboratory of Software Security Engineering Technique, Beijing Institute of Technology,
5 South Zhongguancun Street, Haidian District, Beijing 100081, China

Correspondence should be addressed to Qian Wang; wangqianysu@163.com

Received 12 October 2017; Accepted 19 February 2018; Published 22 March 2018

Academic Editor: Zheng Yan

Copyright © 2018 Qian Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The scale and complexity of software systems are constantly increasing, imposing new challenges for software fault location and daily maintenance. In this paper, the Security Feature measurement algorithm of Frequent dynamic execution Paths in Software, SFFPS, is proposed to provide a basis for improving the security and reliability of software. First, the dynamic execution of a complex software system is mapped onto a complex network model and sequence model. This, combined with the invocation and dependency relationships between function nodes, fault cumulative effect, and spread effect, can be analyzed. The function node security features of the software complex network are defined and measured according to the degree distribution and global step attenuation factor. Finally, frequent software execution paths are mined and weighted, and security metrics of the frequent paths are obtained and sorted. The experimental results show that SFFPS has good time performance and scalability, and the security features of the important paths in the software can be effectively measured. This study provides a guide for the research of defect propagation, software reliability, and software integration testing.

1. Introduction

The increase in complexity of software requirements makes software developers unsure of the development quality of software system; in effect the “software crisis” still has not been completely solved. How to effectively excavate the inherent characteristics of the software system structure, to recognize, measure, manage, and control the complexity of software structure, becomes a key problem for solving the development bottleneck in the software industry.

Research on the complexity of software network structure can combine the methods of complex system science and statistical physics. Depending on the granularity, software systems can be composed of different types of software entities, such as functions, classes, subroutines, packages, and artifacts. With these entities interacting with each other, software systems can achieve specific functional

requirements. If the software entities are viewed as nodes and the relationship between the nodes is abstracted as edges, the software execution process presents a nonlinear network structure according to the relationship of the entities [1] and also a linear sequence structure according to the sequential characteristics of the execution order. Then, the software system can be expressed as an abstracted complex network model and a sequence model, which provides a new train of thought [2] for the description of the software system.

The root cause of the security danger hidden in software lies in the vulnerability of the entity itself. The vulnerability is the measurement of the potential danger of a software entity to be used as an attack and can be discussed from the perspective of computer network [3, 4] or software static code analysis, but the integrity (whole structure) and the dynamic execution (behavior characteristic) of software system are ignored. In addition, the degree to which software system

security is threatened depends not only on the severity of the fault, but also on the fault propagation capacity of the entity. If one or more functions fail, the fault may be propagated to other functions by invocation relationships and further lead to a part of or the whole software system crashing, known as “cascading failure” [5]. Therefore, the software security feature measurement should take into account the vulnerability and propagation of software entities.

How to quantitatively measure the security features of nodes from the software complex network is the premise and basis for further analysis of the software behavior trajectory path. At present, there are lots of methods for discovering the important nodes in complex networks. The classic methods based on centrality contain degree centrality [6], closeness centrality [7], betweenness centrality [8], eigenvector centrality [9], subgraph centrality [10], and so on. The classic methods based on random walk model include PageRank [11], LeaderRank [12], and their improved algorithm NodeRank [13]. Wang and Lü [14] by means of the influence node mining method prove that the defect propagation capacity of a node is stronger if the in-degree and out-degree of the node are bigger. Huang et al. [15] based on the invocation and dependency relationships between functions with the fault probability of nodes calculate the fault accumulation degree of upper nodes by the iteration from the leaf nodes. These methods attempt to describe the relevance of software node importance to fault generation and propagation, but fail to form a measurement of software security.

Sequence or path is the most basic and important way for the description of dynamic software execution process. The full execution path of the whole software can reflect the occurrence order and frequency of the software internal entities. However, the method of path extraction and mining is restricted by the nested, circulatory, iteration and the continuous invocation relationships of entities. Most software path mining algorithms are extracted on the basis of complex networks. For example, Tang et al. [16] propose an algorithm for shortest path mining between any two vertices in complex network. Zhang et al. [17] minimize the length of the extracted path and reduce the unnecessary time overhead by further processing the repetitive structure. The GP method proposed by Nguyen et al. [18] can automatically detect and fix software vulnerabilities according to the software execution path. Murtaza et al. [19] predict future software possible defects by analyzing the historical vulnerability sequence data with characteristics of Markov to provide adequate response time. Zou et al. [20] analyze the reliability of Digital Instrumentation and Control software system based on the flow network model by finding sensitive paths in the complexity software. These algorithms are based on the network to extract path, which can lead to the phenomenon of repeated reading and approximate connection; also, these software security analyses cannot work without existing vulnerability information or real faults as their training data.

In this paper, the Security Feature measurement algorithm of Frequent dynamic execution Paths in Software, SFFPS, is proposed. A complex network model and a sequence model are formed based on software dynamic execution behavior. It is for early security feature measurement,

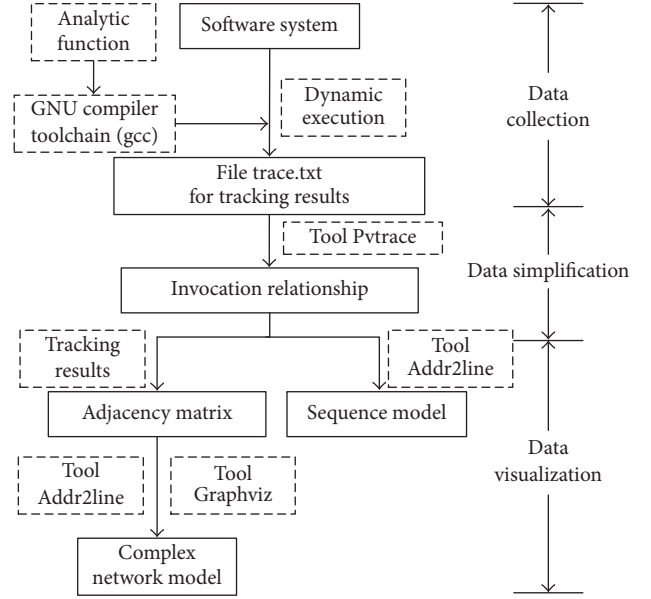


FIGURE 1: Theory process of model construction.

before there are real vulnerabilities or faults generated, which can provide the premise for the software quality and reliability evaluation. The main contributions are as follows.

(1) The software system is mapped to a complex network model and sequence model, from the nonlinear perspective to effectively express the characterization of complex correlation between software entities and from the linear perspective to capture sequential characteristics of the dynamic execution.

(2) The behavior nature of fault accumulation and propagation is analyzed based on the system structure of software dynamic execution and standard measurement of security features (vulnerability and propagation) being defined.

(3) Frequent paths in software dynamic execution are mined and weighted by the node security features. The key paths which are worthy of attention are ensured by both their frequency and security features.

The remainder of the paper is organized as follows. Section 2 gives the model construction. Sections 3 and 4 develop the definition of the security features and the SFFPS algorithm. Section 5 provides some examples. Section 6 presents the performance study of SFFPS and shows the rank of the important paths. Section 7 contains the concluding remarks.

2. Constructions of Complex Network Model and Sequence Model

The dynamic execution trace of software systems contains three phases, which are data collection, tracking data simplification, and data visualization as shown in Figure 1. The modeling process of simple functions is shown in Figure 2.

Phase 1. Match the entry and exit configuration functions of the GNU compiler toolchain (gcc), and insert the analysis function into the entry and exit of the application functions

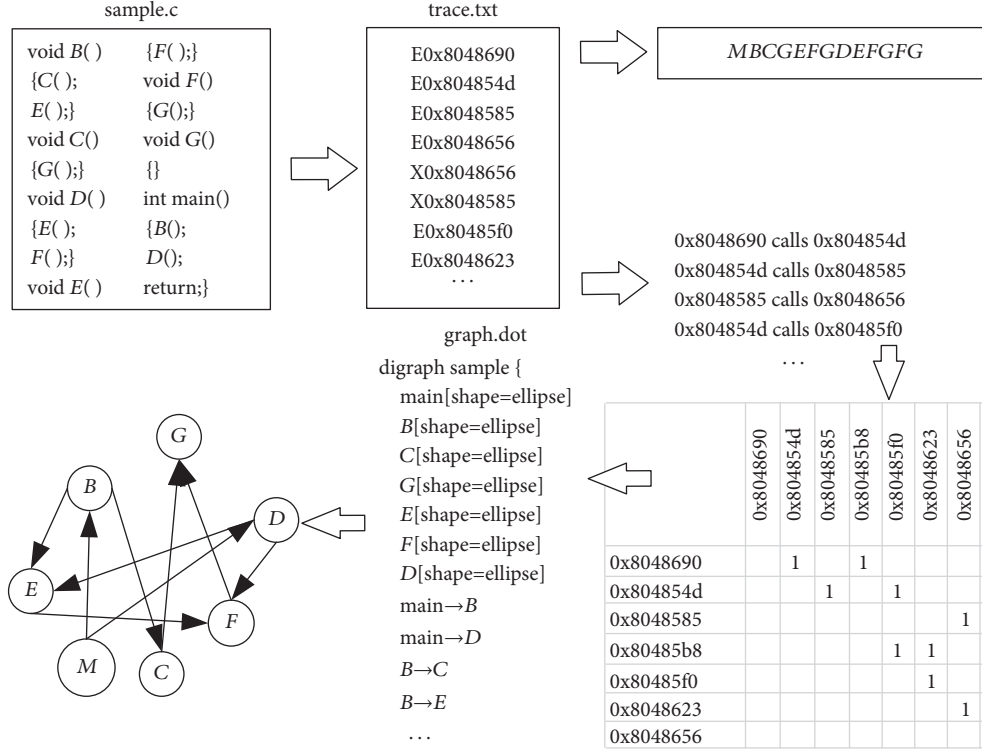


FIGURE 2: Modeling process of invocation relationship between simple functions.

to trace the function execution process. The tracking results are recorded in the file trace.txt.

Phase 2. The letters “E” and “X” before the tracking addresses represent the entry and exit of a function, respectively. A simplification tool Pvtrace is used to analyze the function invocation according to the letters “E” and “X.” An address transformation tool Addr2line is used and the address is transformed to function name.

Phase 3. Map the function invocation order to sequence model and a visualization tool Graphviz is used to form the complex network, which defines the global relationship between all the functions.

According to Figure 2, the corresponding relationships of function address and function name are as follows:

0x8048690 → M(main); 0x804854d → B;
0x8048585 → C; 0x80485b8 → D.
0x80485f0 → E; 0x8048623 → F, 0x8048656 → G.

Only the addresses with the letter “E” are used for sequence model construction.

3. The Security Feature Definition and Measurement of Function Nodes

The security feature measurement of a function node is based on the software structure; the analysis of vulnerability and propagation is according to cumulative effect and the

spread effect caused by the mechanism of fault production and propagation. The global accessibility and fault tolerance with step attenuation effect are fully considered, so the node security features are calculated according to the degree distribution and step attenuation factor.

Definition 4 (software complex network). In a software complex network, functions are defined as the nodes; the invocation relationships between functions are defined as edges.

Definition 5 (vulnerability). Vulnerability of a function node is the characteristic that a function node may break down because of the effect of its invoked fault node through invocation relationship.

Typically, if a node invokes more other nodes, it is more functional and vulnerable. That is to say, it is more likely to be affected and be faulted. The calculation of V (vulnerability) is as follows:

$$V(u) = \text{outDegree}(u) + \sum_{w \in \text{OS}(u)} \theta * V(w), \quad (1)$$

where u, w represent function nodes, $V(u)$ represents the vulnerability of node u , $\text{OutDegree}(u)$ represents the out-degree of node u , θ represents the step attenuation factor, which satisfies $\theta \in (0, 1)$, and $\text{OS}(u)$ represents the direct out-neighbor set of node u .

Definition 6 (propagation). Propagation of a function node is the characteristic that a function node may propagate its

Input: Complex network CN, step attenuation factor θ
Output: Node list with security features NList
 for each node u in CN
 { $V(u) = \text{calculation_}V(u)$;
 $P(u) = \text{calculation_}P(u)$;
 NList.add ($u, V(u), P(u)$); }
Procedure calculation_ $V(u)$
 { $V(u) = \text{outDegree}(u)$;
 For each node $w \in \text{OS}(u)$
 $V(u) += \text{calculation_}V(w)$;
 return $V(u)$; }
Procedure calculation_ $P(u)$
 { $P(u) = \text{inDegree}(u)$;
 for each node $w \in \text{IS}(u)$
 $P(u) += \text{calculation_}P(w)$;
 return $P(u)$; }

ALGORITHM 1: Calculation of node security features.

fault to the nodes by which it is invoked. The calculation of P (propagation) is as follows:

$$P(u) = \text{inDegree}(u) + \sum_{w \in \text{IS}(u)} \theta * P(w), \quad (2)$$

where $P(u)$ represents the propagation capacity of node u , $\text{inDegree}(u)$ represents the in-degree of node u , and $\text{IS}(u)$ represents the direct in-neighbor set of node u .

Algorithm 1 describes the calculation process of vulnerability and propagation.

4. Mining Frequent Paths from Dynamic Execution with Security Feature Measurement

The importance of a software dynamic execution path takes into account two aspects: one is the occurrence frequency of the path and the other one is the security feature coming from the nonrepetitive nodes contained in the path. These two aspects are complementary. For example, if there are lots of loop bodies in the software execution, loop body and its subset are always frequent. But because most of its contained nodes are the same, the fault influence range is small. Similarly, if a path contains many different nodes with a lower occurrence frequency, its impact range is large, but its occurrence possibility is small. That is to say, if the frequency of a path is very high and the path contains more nonrepetitive nodes, the path is worthy of more attention.

4.1. Relative Definitions of Frequent Path. Let $F = \{f_1, f_2, f_3, \dots, f_n\}$ be a set of function symbols. S is a software execution path, and it is composed of function symbols with time-ordered occurrence. Minimal support count (mincount) can be calculated by $\text{mincount} = \text{minsup} * |S|$, where minsup is a given threshold and $|S|$ is the number of function symbols in S . If there are k symbols in S , S is a k -path.

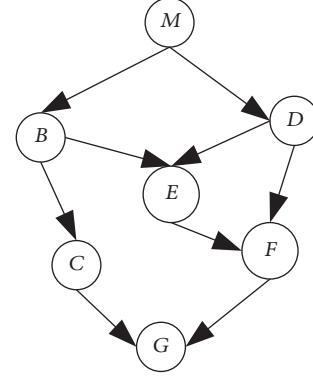


FIGURE 3: Complex network model of simple function invocation relationship.

Definition 7 (subpath and superpath). A path $S_1 = \langle a_1, a_2, \dots, a_m \rangle$ is a subpath of another path $S_2 = \langle b_1, b_2, \dots, b_n \rangle$, denoted as $S_1 \subseteq S_2$, if there are numbers i_1, i_2, \dots, i_m , such that $1 \leq i_1 < i_2 < \dots < i_m \leq n$ and $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_m \subseteq b_{i_m}$. It can also be said that S_2 is a superpath of path S_1 .

Definition 8 (support number). S is a path; the support number of S , denoted as $\text{sup}(S)$, is defined as its occurrence number in the software execution.

Property 9 (frequent path). A path S is frequent if its support number $\text{sup}(S)$ is equal to or more than mincount .

Property 10 (antimonotone). If path A is not a frequent path, any path B containing A , which is a superpath of A , cannot be a frequent path.

4.2. Weighting the Frequent Path Based on the Security Features of Function Node. SFFPS algorithm is for mining the security features of frequent paths based on the dynamic execution sequence model and the node security features in the complex network model. It contains two phases: one is frequent path mining and the other one is security feature weighting. First, the function nodes in the sequence model are read to form the function position set. Then, the position index is used for pattern growth; this self-growth strategy can avoid candidate generation and ensure the continuity of function execution. Finally, path frequency is validated by minimum support count mincount , and path is weighted according to the security feature of the nonrepetitive nodes contained in it. The security features of the frequent paths are measured. Algorithm 2 describes the mining and weighting process.

5. An Illustrative Example

The complex network in Figure 2 is a variant of the tree-like structure in Figure 3, which is redrawn for easier understanding.

Without losing generality, the coordination factor is set to 0.5. Security features of each node are calculated as follows.

Input: Function execution path S , minimal support threshold minsup
Output: Path list with security features Slist
 $\text{mincount} = \text{minsup} * |S|;$
for each node S_i in S
 $\{ \text{Pos}(S_i).add(S_i.\text{pos}); \}$
for each $\text{Pos}(S_i)$
 $\{ \text{sup}(S_i) = |\text{Pos}(S_i)|;$
 if $(\text{sup}(S_i) < \text{mincount})$
 Delete $\text{Pos}(S_i);$
 else
 $L_1 = L_1.add(S_i, \text{sup}(S_i)); \}$
for $(k = 2; L_{k-1} \neq \emptyset; k++)$
 $\{ \text{gen_mine}(L_{k-1});$
 for each $l_m \in L$
 $\{ \text{for each different function symbol } u \text{ in } l_m$
 $\{ V(l_m) += V(u);$
 $P(l_m) += P(u); \} \}$
Sort each $l_m \in L$ by $V(l_m), P(l_m)$ and form $\text{Slist};$
Procedure $\text{gen_mine}(L_{k-1})$
 $\{ \text{for each } l_i \in L_{k-1}$
 $\{ \text{for each position pos in Pos}(l_i)$
 $\{ S_j = S[\text{pos} + 1];$
 for each position pos in $\text{Pos}(l_i)$
 $\{ \text{if } (\text{pos} + 1 \text{ exists in Pos}(S_j))$
 $\{ \text{Pos}(l_i S_j).add(\text{pos} + 1); \} \}$
 $\text{sup}(l_i S_j) = |\text{Pos}(l_i S_j)|;$
 if $(\text{sup}(l_i S_j) < \text{mincount})$
 delete $\text{Pos}(l_i S_j);$
 else
 $L_k = L_k.add(l_i S_j); \}$

ALGORITHM 2: Security feature measurement of frequent paths in software.

As the “main” function is special (vulnerability is always large and propagation is 0), it is excluded for measurement.

Vulnerability

$$\begin{aligned}
 V(G) &= \text{outDegree}(G) = 0. \\
 V(C) &= \text{outDegree}(C) + \theta * V(G) = 1 + 0.5 * 0 = 1. \\
 V(F) &= \text{outDegree}(F) + \theta * V(G) = 1 + 0.5 * 0 = 1. \\
 V(E) &= \text{outDegree}(E) + \theta * V(F) = 1 + 0.5 * 1 = 1.5. \\
 V(D) &= \text{outDegree}(D) + \theta * \{V(E) + V(F)\} = 2 + 0.5 * (1.5 + 1) = 3.25. \\
 V(B) &= \text{outDegree}(B) + \theta * \{V(C) + V(E)\} = 2 + 0.5 * (1 + 1.5) = 3.25.
 \end{aligned}$$

Propagation

$$\begin{aligned}
 P(B) &= \text{inDegree}(B) = 1; P(D) = \text{inDegree}(D) = 1. \\
 P(C) &= \text{inDegree}(C) + \theta * P(B) = 1 + 0.5 * 1 = 1.5. \\
 P(E) &= \text{inDegree}(E) + \theta * \{P(B) + P(D)\} = 2 + 0.5 * (1 + 1) = 3. \\
 P(F) &= \text{inDegree}(F) + \theta * \{P(E) + P(D)\} = 2 + 0.5 * (3 + 1) = 4.
 \end{aligned}$$

$$P(G) = \text{inDegree}(G) + \theta * \{P(C) + P(F)\} = 2 + 0.5 * (1.5 + 4) = 4.75.$$

According to the sequence model of the example, $S = (M)BCGEFGDEFGFG$, if the minsup is set to 0.15, $\text{mincount} = 0.15 * 12 \approx 2$.

$$\text{Pos}(B) = \{1\}; \text{Pos}(C) = \{2\}; \text{Pos}(D) = \{7\}; \text{Pos}(E) = \{4, 8\}.$$

$$\text{Pos}(F) = \{5, 9, 11\}; \text{Pos}(G) = \{3, 6, 10, 12\}.$$

Frequent 1-Path

$$E, \text{sup}(E) = 2, \text{Pos}(E) = \{4, 8\}.$$

$$F, \text{sup}(F) = 3, \text{Pos}(F) = \{5, 9, 11\}.$$

$$G, \text{sup}(G) = 4, \text{Pos}(G) = \{3, 6, 10, 12\}.$$

The mining method of frequent 2-path is based on the position set of the frequent 1-path by using the adjacent position value as index to find the extended paths. For example, the position set of node E is $\{4, 8\}$, and its extended position set is $\{5, 9\}$. The function nodes in positions 5 and 9 both correspond to node F . So, $\text{Pos}(EF) = \{5, 9\}$ is obtained, $\text{sup}(EF) = 2$, and path EF is a frequent 2-path.

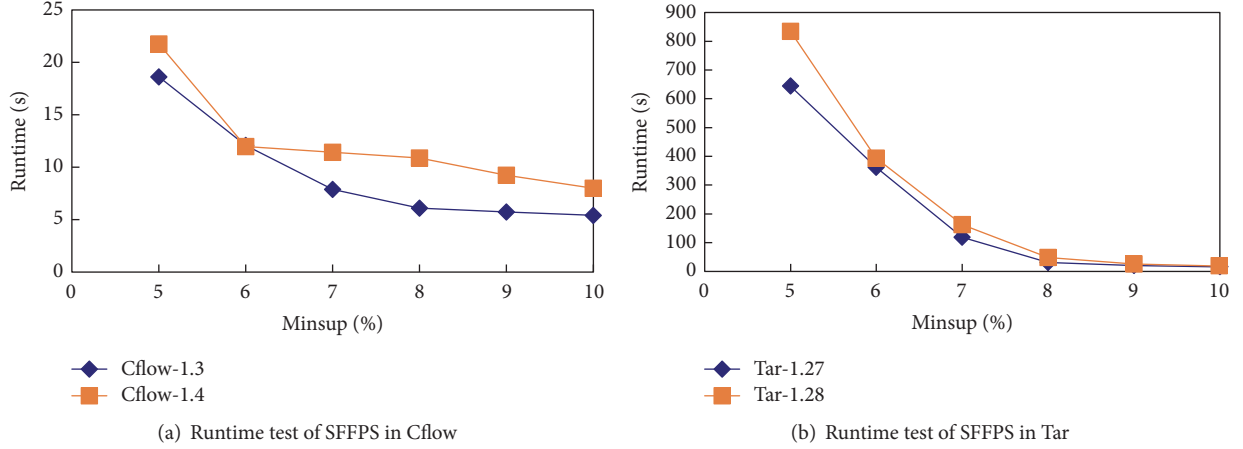


FIGURE 4: Runtime test of SFFPS with different support thresholds.

TABLE 1: The security features of frequent paths.

Frequent paths	sup	V	P
E	2	1.5	3
F	3	1	3
G	4	0	4.75
EF	2	2.5	7
FG	4	1	8.75

Frequent 2-Path

$EF, \text{sup}(EF) = 2, \text{Pos}(EF) = \{5, 9\}; FG, \text{sup}(FG) = 3.$
 $\text{Pos}(FG) = \{6, 10, 12\}.$

The security features of frequent 1-path included in the function nodes are calculated as before, and the security features of frequent 2-path are calculated as follows. Table 1 shows the security features of all the frequent paths.

$$V(EF) = V(E) + V(F) = 1.5 + 1 = 2.5;$$

$$P(EF) = P(E) + P(F) = 3 + 4 = 7. \quad (3)$$

6. Experimental Results

Experiments are performed on a PC with Intel® Core™ 3.6 GHz CPU and 16 G main memory, running on Windows 8. We evaluate the runtime and scalability of the algorithm SFFPS and calculate the fault feature ranks of nodes and important paths. To test the algorithms in the same coding environment, all the programs are written in Java using MyEclipse. Datasets used in the experiment are open-source software programs of Cflow and Tar obtained from open-source software library (<https://sourceforge.net>).

6.1. Runtime and Scalability Tests of SFFPS. By testing the runtime and scalability of SFFPS, two newest versions of each Cflow and Tar are selected. The support threshold is from 0.005 to 0.01 for runtime test, and the upper threshold 0.01 is used for scalability test. The total runtime is composed of

three parts, node fault feature calculation, frequent pattern mining, and weight appending. Figure 4 is the runtime test of SFFPS with different support thresholds and Figure 5 is the scalability test with different length percentages of the sequence when the support threshold is set to 0.01.

From Figure 4, SFFPS performs well in the support threshold range $[0.005, 0.010]$. This is due to the adjacency table which is for the storage of the complex network model. The calculation of the out-degree and in-degree of the nodes is made easier, which improves the calculation of node security feature. Furthermore, as the sequence model is based on the start order of each function, the detailed invocation and end time of a node are ignored, and the length of the sequence model is simplified. Also, position value index is used for the mining and pattern growth of the paths, which avoids candidate generation, and index methods are always effective. Finally, the weight appending process achieves efficiency because fewer nodes are involved by the strategy of nonrepetition.

From Figure 5, SFFPS shows good scalability on the software Cflow. With the increase of the length of the sequence, the execution time of SFFPS is essentially a linear growth. From the experimental data, the number of frequent sequences is also increasing. This indicates that the functions of Cflow are uniformly distributed. However, the time overhead of software Tar is quite expensive around 40% of sequence length; the number of frequent sequences increases rapidly from 194 when the percentage is 20% to 1123. After that, the time overhead and the number of frequent sequences reduces. This indicates that there are more core functions in software Tar and there are more invocations of core functions in the early stage of the program.

6.2. The Security Features of the Function Nodes. Tables 2 and 3 show the security feature rank and value of the function nodes in the newest versions of Cflow and Tar.

From Tables 2 and 3, the security features of the same function nodes are relatively stable for different versions of the same software. So, in the process of version evolution, it can be inferred and predicted that the same function should

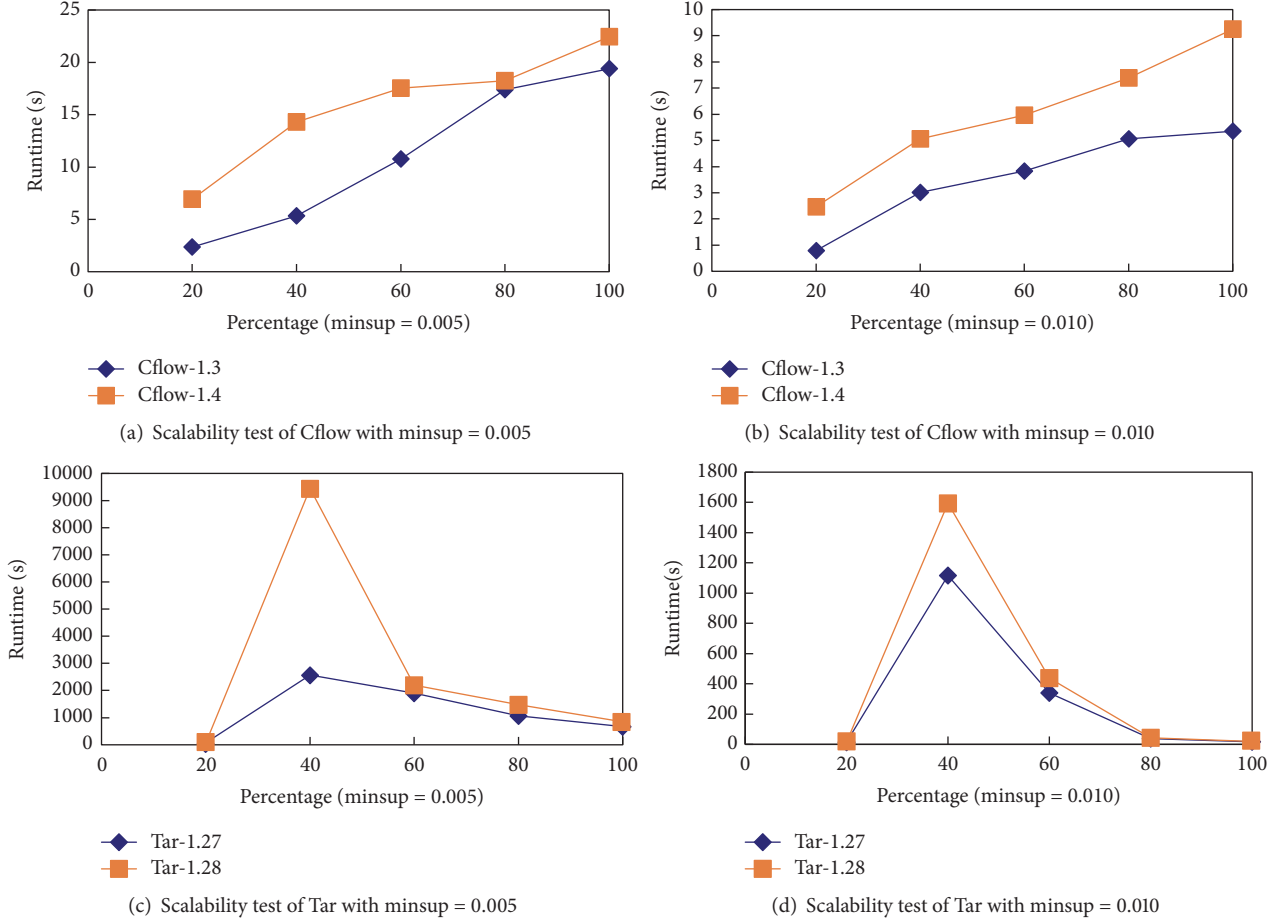


FIGURE 5: Scalability test of SFFPS with different percentages of sequence length.

TABLE 2: Rank and value of function node security features in Cflow.

Function name	Vulnerability		Function name	Propagation	
	Cflow-1.3	Cflow-1.4		Cflow-1.3	Cflow-1.4
	(rank/value)			(rank/value)	
parse_variable_declaration	1/45.05	1/47.07	nexttoken	1/33.73	1/36.17
parse_declaration	2/40.77	2/41.85	tokpush	2/19.74	3/20.96
yyparse	3/37.50	4/38.82	putback	3/19.32	4/19.32
parse_typedef	4/19.33	6/20.78	get_token	4/17.86	5/19.08
tree_output	5/18.50	5/25.56	linked_list_append	5/15.47	2/21.06
func_body	6/18.15	7/18.28	lookup	6/13.79	6/13.94
parse_function_declaration	7/17.80	8/17.99	hash_symbol_hasher	7/12.90	8/13.04
parse_dcl	8/16.92	9/17.41	hash_symbol_compare	7/12.90	8/13.04
expression	9/16.17	10/16.42	yy_load_buffer_state	8/12.86	7/13.27
initializer_list	10/12.53	13/12.72	yylex	9/9.93	11/10.54

have approximate rank in a new software version. Also, the function rank in the old version can be used as a basis for the version upgrade process with function nodes remove, merger, or update. The nodes with larger rank changes should be given more attention.

Tables 4 and 5 show the frequent paths of Cflow-1.4 in the top 10 security feature ranks of vulnerability and propagation.

There are double meanings of the paths listed in Tables 4 and 5. One is that the paths are frequent, which first affirms that the occurrence possibility of the path is relatively large. The other one is that the security feature values of the paths are larger, which evaluates the security risk of the path. Only when both of them work together can we make a persuasive security measurement.

TABLE 3: Rank and value of function node security features in Tar.

Vulnerability			Propagation		
Function name	Tar-1.27 (rank/value)	Tar-1.28	Function name	Tar-1.27 (rank/value)	Tar-1.28
dump_file0	1/42.84	1/44.09	to_chars	1/17.28	1/17.28
create_archive	2/32.02	2/33.21	assign_string	2/11.67	2/11.67
dump_file	3/25.92	4/27.04	to_octal	3/9.64	3/9.64
dump_regular_file	4/19.37	5/19.37	tar_copy_str	4/7.13	4/7.13
dump_hard_link	5/17.37	6/17.37	set_next_block_after	5/6.64	5/6.64
start_header	6/15.62	8/15.62	find_next_block	6/6.16	6/6.16
dump_dir0	7/15.37	7/16.37	start_header	7/5.84	7/5.84
dump_dir	8/10.68	9/11.18	finish_header	7/5.84	7/5.84
close_archive	9/7.50	11/8.00	current_block_ordinal	7/5.84	7/5.84
_open_archive	10/7.25	10/8.75	flush_archive	8/5.83	8/5.83

TABLE 4: Vulnerability rank and value of frequent paths (minsup = 0.01).

Paths	Rank/value
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler, set_active	1/16.75
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline	1/16.75
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler	1/16.75
direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline	2/15.75
include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler, set_active	2/15.75
include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler	2/15.75
include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline	2/15.75
direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler	2/15.75
direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler, set_active	2/15.75
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol	3/14.75

In addition, the frequency of the path can be used to predict the function nodes that are going to be affected, and the security features of the path can be used to evaluate the possible impact scale of the abnormal path. For example, the main consideration of random fault detection is the vulnerability. According to the first path of Table 4, from the perspective of frequency, if the first three functions of a fault path are “is_printable,” “include_symbol,” and “direct_tree,” then the next functions which are likely to be affected are “include_symbol,” “print_symbol,” “gnu_output_handler,” and so on. From the perspective of security features, the path displays higher rank and value in vulnerability, which indicates the fault location is relatively accurate. If it is a hostile attack detection, the attacker expects a wider range effect, so the propagation should be considered more. In this case, the analysis method is similar.

7. Conclusion

In this paper, a novel algorithm, SFFPS, is proposed to define and measure the security feature of dynamic execution path in software. Complex network model and sequence model are constructed for the record of invocation relationship and function execution order. The node degree in the complex network is used for security feature analysis from a structural perspective before real fault occurrence. The paths extracted from the sequence model are used for frequency test and weighted by the node security features. Finally, frequent dynamic execution paths with top security feature rank are mined as important paths which should be of greater concern. With the experiment, SFFPS can effectively mine the important paths from the newest versions of software programs Cflow and Tar. SFFPS can be applied as a basis for software evolution, a tool for software internal structure

TABLE 5: Propagation rank and value of frequent paths (minsup = 0.01).

Paths	Rank/value
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler, set_active	1/36.35
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline	2/34.48
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler	2/34.48
include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler, set_active	3/34.42
direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler, set_active	3/34.42
include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler, set_active	4/32.67
is_printable, include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name	5/32.60
direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline	6/32.54
include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline, gnu_output_handler	6/32.54
include_symbol, direct_tree, include_symbol, print_symbol, gnu_output_handler, print_symbol, print_level, print_function_name, newline	6/32.54

analysis, and a guidance to fault location and attack detection, which are helpful for software quality assurance.

Conflicts of Interest

There are no conflicts of interest related to this paper.

Acknowledgments

This work is supported by the National Key R&D Program of China (2016YFB0800700), the National Natural Science Foundation of China under Grant nos. 61472341, 61772449, and 61572420, the Natural Science Foundation of Hebei Province, China, under Grant nos. F2016203330 and F2015203326, the Advanced Program of Postdoctoral Scientific Research under Grant no. B2017003005, and the Doctoral Foundation of Yanshan University under Grant no. B1036.

References

- [1] Y.-T. Ma, K.-Q. He, B. Li, and J. Liu, "Empirical study on the characteristics of complex networks in networked software," *Ruan Jian Xue Bao/Journal of Software*, vol. 22, no. 3, pp. 381–407, 2011.
- [2] X. Wang and W. Yichen, *Research Progress on Error Propagation Model in Software System[J]*, vol. 43, Computer Science, 2016.
- [3] H. L. Vu, K. K. Khaw, T. Y. Chen, and F.-C. Kuo, "A new approach for network vulnerability analysis," *IEEE Conference on Local Computer Networks*, vol. 58, no. 4, pp. 200–206, 2008.
- [4] X. J. Qin, L. Zhou, Z. N. Chen, and S. . Gan, "Software vulnerable trace's solving algorithm based on lazy symbolic execution," *Chinese Journal of Computers. Jisuanji Xuebao*, vol. 38, no. 11, pp. 2290–2300, 2015.
- [5] J. Wang, Y.-H. Liu, and X.-L. Liu, "Model for cascading faults in complex software," *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 34, no. 6, pp. 1137–1147, 2011.
- [6] D. Wei, Y. Li, Y. Zhang, and Y. Deng, "Degree centrality based on the weighted network," in *Proceedings of the 2012 24th Chinese Control and Decision Conference, CCDC 2012*, pp. 3976–3979, China, May 2012.
- [7] K. Okamoto, W. Chen, and Y. Li X, "Ranking of Closeness Centrality for Large-Scale Social Networks," in *International Workshop on Frontiers in Algorithmics*, pp. 186–195, Springer-Verlag, 2008.
- [8] M. Kitsak, S. Havlin, G. Paul, M. Riccaboni, F. Pammolli, and H. E. Stanley, "Betweenness centrality of fractal and nonfractal scale-free model networks and tests on real networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 75, no. 5, Article ID 056115, 2007.
- [9] X. Wu, M. Zhang, and Y. Han, "Research on centrality of node importance in scale-free complex networks," in *Proceedings of the 31st Chinese Control Conference, CCC 2012*, pp. 1073–1077, chn, July 2012.
- [10] X. Yan, C. Li, L. Zhang, and Y. Hu, "A new method optimizing the subgraph centrality of large networks," *Physica A: Statistical Mechanics and its Applications*, vol. 444, pp. 373–387, 2016.
- [11] L. Page, "The PageRank citation ranking: Bringing order to the web," *Stanford Digital Libraries Working Paper*, vol. 9, no. 1, pp. 1–14, 1998.
- [12] S. Xu and P. Wang, "Identifying important nodes by adaptive LeaderRank," *Physica A: Statistical Mechanics and its Applications*, vol. 469, pp. 654–664, 2017.
- [13] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos, "Graph-based analysis and prediction for software evolution," in *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*, pp. 419–429, IEEE, Zürich, Switzerland, June 2012.

- [14] B.-Y. Wang and J.-H. Lü, "Software networks nodes impact analysis of complex software systems," *Ruan Jian Xue Bao/Journal of Software*, vol. 24, no. 12, pp. 2814–2829, 2013.
- [15] G. Huang, P. Zhang, Y. Li, and J. Ren, "Mining the important nodes of software based on complex networks," *ICIC Express Letters*, vol. 9, no. 12, pp. 3263–3268, 2015.
- [16] J.-T. Tang, T. Wang, and J. Wang, "Shortest path approximate algorithm for complex network analysis," *Journal of Software*, vol. 22, no. 10, pp. 2279–2290, 2011.
- [17] B. Zhang, G. Huang, H. He, and J. Ren, "Approach to mine influential functions based on software execution sequence," *IET Software*, vol. 11, no. 2, pp. 48–54, 2017.
- [18] T. Nguyen, W. Weimery, C. Le Goues, and S. Forrest, "Using execution paths to evolve software patches," in *Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2009*, pp. 152–153, USA, April 2009.
- [19] S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, and A. B. Bener, "Mining trends and patterns of software vulnerabilities," *The Journal of Systems and Software*, vol. 117, pp. 218–228, 2016.
- [20] B. Zou, M. Yang, E.-R. Benjamin, and H. Yoshikawa, "Reliability analysis of Digital Instrumentation and Control software system," *Progress in Nuclear Energy*, vol. 98, pp. 85–93, 2017.