IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 23, 2020, accepted June 28, 2020, date of publication July 1, 2020, date of current version July 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006361

A General Framework to Understand Vulnerabilities in Information Systems

XIONG ZHANG[1], HAORAN XIE[2], (Senior Member, IEEE), HAO YANG[1], HONGKAI SHAO[1], AND MINGHAO ZHU[1]

[1]School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China
[2]Department of Computing and Decision Sciences, Lingnan University, Hong Kong

Corresponding author: Minghao Zhu (mhzhu@bjtu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 71801014, in part by the Beijing Social Science Foundation under Grant 17GLC069, and in part by the Ministry of Finance of the People's Republic of China under Grant CJ [2018] No. 281.

ABSTRACT Firms and organizations are increasingly facing security issues related to vulnerabilities in their information systems. Firms, especially small and medium-sized enterprises, usually have very limited security resources and thus have difficulty understanding vulnerabilities and fixing them accordingly. This study aims to build a general framework that can help firms understand the characteristics of vulnerabilities in information systems: for instance, what category a specific vulnerability belongs to, what potential risks it poses, and what the key clues are to addressing it. To this end, we collect data on real vulnerabilities that have emerged in firms' information systems from a popular vulnerability report platform. Features are extracted at four different levels, namely, the word, phrase, topic, and record levels. The experimental results show that the general framework helps characterize the modes and patterns of various types of vulnerabilities. This study contributes to the security literature by providing a deeper understanding of the characteristics of vulnerabilities and their related suggested solutions. Firms can apply this framework to ensure information security.

INDEX TERMS Classification, information security, risk-level prediction, topic analysis, vulnerability.

I. INTRODUCTION

A vulnerability is typically a flaw in the source code, a defect, or even a logic error in the design of software or information systems, which may lead to potentially compromised security for an endpoint or network [1]. Vulnerabilities might be exploited by hackers, leading to unauthorized access to information systems, theft of important data, or even destruction of the entire system.

As firms increasingly use information systems to fulfill business needs, such as product and service offerings, interactions with consumers, online payments, deliveries, and other basic business functionalities, they face greater risks of vulnerabilities in relation to such systems. A security failure in one of these information systems could lead to huge losses for firms. For example, it is reported that over 60% of online banks have a poor or extremely poor level of protection, and fraud and theft of funds are possible in over 50% of online banking applications [2].

Vulnerabilities are being reported in increasing numbers. Figure 1 shows the annualized number of vulnerabilities reported in the China National Vulnerability Database of Information Security in the last few years.

Firms face increased vulnerabilities in their information systems for two main reasons: (1) the design and implementation of software and information systems are becoming increasingly complicated. Higher complexity leads to a greater likelihood of encountering vulnerabilities [3]. (2) Vulnerabilities are very complicated by nature because there will be different security vulnerabilities in different types of software and hardware devices, different versions of the same device, different systems composed of different devices, and different settings of the same system.

Given this complexity, addressing vulnerabilities in a timely manner to avoid security threats is a serious challenge for firms. Studies on vulnerabilities are urgently needed.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhitao Guan.

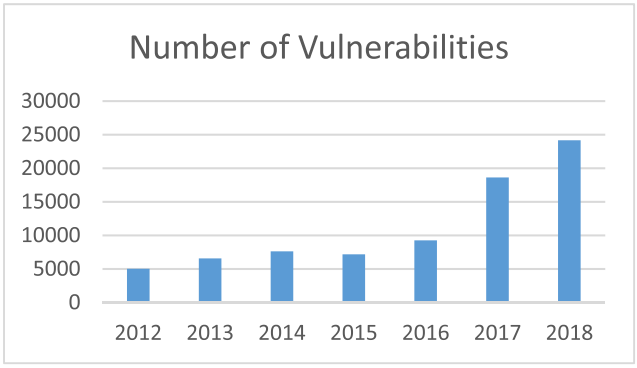

FIGURE 1. Annualized number of vulnerabilities in the China National Vulnerability Database of Information Security (2012–2018).

Understanding vulnerabilities and addressing them if they occur have become critical issues for firms.

Existing studies have addressed the subjects of vulnerability identification in information systems [4], [5], threat analysis for secure software design [6], topic analysis in software repositories [7], management issues related to vulnerabilities in information systems [8], economic issues in the black market of information security threats [9], security vulnerability analysis in different applications, e.g., power systems [10], [11], fog computing-enabled robust demand response with consideration of collusion attacks in the Internet of Energy [12], cross-lingual multi-keyword ranked search with semantic extensions over encrypted data [13], and posters' behavior in online forums [14]. However, so far, there have been few in-depth studies on vulnerabilities dealing with the automated classification and risk-level prediction of vulnerabilities and the automated generation of solutions for various types of vulnerabilities in software and information systems. Owing to the urgent need of firms to deal with vulnerabilities, these practical issues should be addressed in research. Accordingly, this work aims to fill this gap.

The risk levels represented by vulnerabilities and the solutions required to address them are different for different types of vulnerabilities. Common types of vulnerabilities that occur in information systems include cross-site scripting (XSS), denial of service, SQL injections, weak passwords, and application configuration errors. Owing to the differences in their nature, different vulnerabilities present different risk levels. For example, despite not directly harming web servers, XSS attacks could spread through websites, resulting in the theft of website user accounts, which would cause serious harm to websites. If not checked carefully, a malicious SQL instruction may be mistakenly run as a normal SQL instruction. This may cause the database to be attacked, resulting in data theft, modification, and deletion. This could further cause the website to be attacked by malicious codes, backdoor programs, and other hazards. A backdoor program is a small executable which could be dropped into users' computer systems by using SQL injection. This backdoor program could listen in on unused ports in users' computer systems. Thus, attackers could use backdoor programs to manage files, install software, or even control the whole computer system.

In-depth investigations of various types of vulnerabilities could provide guidance for firms in implementing appropriate and timely responses to avoid significant losses. Specifically, studying vulnerabilities in software and information systems can (1) help elucidate the characteristics, patterns, and modes of different types of vulnerabilities, (2) help firms evaluate the potential risk levels of various vulnerability patterns, (3) help classify vulnerabilities into the right categories, and (4) help clarify how to avoid or address certain types of vulnerabilities. From a data-driven perspective, this study builds a general framework to better understand the real vulnerabilities that have appeared in the information systems of firms.

Crowdsourcing testing, as a new testing service mode, is gradually becoming a trend in information security testing. Unlike the traditional testing mode, crowdsourcing testing relies on a vulnerability coordination platform on which personnel with different types of technology expertise from different regions can contribute to vulnerability detection. Through a vulnerability reward platform incorporating enterprises and cybersecurity experts, firms can pay cybersecurity experts rewards for discovering vulnerabilities. This testing mode is commonly used by firms with limited security resources, especially small and medium-sized enterprises.

Cybersecurity experts can post detected vulnerabilities and their suggested solutions on a well-known vulnerability coordination platform. This platform can serve as a database of various types of vulnerabilities in information systems. We implement a web crawler to collect the raw data on real vulnerabilities that have appeared in the information systems of firms from this platform.

We attempt to understand such vulnerabilities and suggested solutions from the perspective of machine learning. In this study, we propose a general framework integrating the following functionalities: (1) automatic classification of vulnerabilities into the right categories; (2) automatic prediction of the risk levels of different types of vulnerabilities; and (3) automatic identification of solutions to vulnerabilities in software and information systems. We first analyze the descriptions, proofs, and suggestions related to vulnerabilities using text analysis and topic modeling. Then, we classify vulnerabilities using popular classification algorithms by considering text, linguistic statistic, and key topic features. Based on the topic analysis of vulnerabilities, we then build a model to automatically predict the risk levels of various vulnerabilities. Finally, by analyzing the details and solutions of vulnerabilities using a topic modeling approach, this framework can automatically generate potential solutions to new vulnerabilities that appear in the information systems of enterprises.

Our experiments are designed and conducted based on vulnerability reports from a well-known crowdsourcing testing platform. The models are trained using features at different levels: linguistic statistics, text, and key topics. The models

are evaluated using the standard AUC metric. The experimental results show that the general framework achieves good performance in vulnerability classification and risk-level prediction for most types of vulnerabilities. The boosting model, logistic regression, and linear discriminant analysis achieve relatively higher overall classification performance than other techniques for most types of vulnerabilities. On the other hand, the bagging model and boosting model achieve higher performance in risk-level prediction than other techniques for most types of vulnerabilities. In addition, the key topics of vulnerabilities and the suggested solutions provide a deep understanding of vulnerabilities. By using this framework, firms can respond to vulnerabilities in software and information systems in a timely manner and reduce potential losses.

On the basis of real vulnerability data, this article proposes to deeply investigate how to automatically understand vulnerabilities and how to fix them in a timely manner using artificial intelligence techniques. The dataset in this research is quite challenging to analyze since we consider source codes in multiple programming languages and comments in both Chinese and English, as well as other textual data. This dataset is noisy and unstructured due to the existence of URLs, misspellings, typos, and unconventional acronyms, among other features.

This research contributes to the security literature by building a general framework to understand in depth the vulnerabilities that firms face: the class a specific type of vulnerability belongs to, its risk level, and some clues to fix it. The proposed framework fulfills this need by using step-by-step key functional modules. Thus, this article goes beyond most existing literature, which mainly investigates the identification of vulnerabilities. This research also has significant practical implications since the proposed framework provides general guidelines to help firms address security issues arising from vulnerabilities in their information systems.

The rest of this paper is organized as follows: Section II reviews related literature on vulnerability, topic modeling, and classification. Section III introduces the data sources used in this study. Section IV presents the general framework. Section V introduces the experimental design, presents the experimental results, and finally discusses these results. Section VI concludes the paper.

II. LITERATURE REVIEW

This work is closely related to the following research streams: (1) vulnerability and (2) topic modeling analysis.

A. VULNERABILITY

Vulnerability has received considerable research attention as an important issue in software and information systems.

Studies have focused on the main reasons for vulnerabilities in software and information systems. Revnivykh and Fedotov [3] concluded that complexity is one of the main reasons for the appearance of vulnerabilities in information systems. These authors suggested that improving the quality of the testing components of information systems could be a possible method to mitigate vulnerabilities. Similarly, Cong and Romero [15] empirically showed that vulnerability is positively correlated with the complexity of the information system.

Other researchers have aimed to investigate how to predict vulnerabilities in software and information systems. Hovsepyan *et al.* [4] employed an support vector machine to predict vulnerabilities in mobile applications on the Android platform. To this end, the authors filtered out all comments, strings, and numbers. In contrast, we consider all source code, comments, and other textual data posted by cybersecurity experts in the vulnerability descriptions and solutions; thus, the data are more unstructured and noisier in our work. In addition, Hovsepyan *et al.* only considered vulnerabilities in mobile applications. Instead, we consider various types of vulnerabilities. Ghoujdi [16] explored how to use information hidden in texts in the Common Vulnerabilities and Exposures (CVE) system and the Open Source Vulnerability Database (OSVDB) for vulnerability prediction and clustering.

Researchers have also attempted to build a risk management framework for vulnerabilities in software and information systems. For instance, Liu *et al.* [17] proposed an artificial immunity-based model for risk management in information system security, in which vulnerabilities in information systems could be identified using math-modeled detectors. These authors showed that the proposed model could quantify the risk level in real time. Farahmand *et al.* [8] proposed a five-step approach for risk management in information systems that involved estimating the expected cost of a security incident.

Other studies have used various techniques to obtain a better understanding of files related to software and information systems. Samtani *et al.* [18] proposed an examination of the functions and characteristics of tutorials, source codes, and related attachments in an online forum using classification algorithms and topic analysis. Similarly, Ruohonen [19] attempted to categorize web-related and other exploits of known software vulnerabilities using topic modeling and a random forest classifier. Abbasi *et al.* [20] built an approach to identifying technology experts in an online forum by analyzing texts posted by experts.

However, the analysis of unstructured textual data on vulnerabilities, such as source codes and related files, is still immature. Textual data, especially those posted by cybersecurity experts on vulnerability coordination platforms, are usually noisy due to misspellings and typographical errors, unconventional acronyms, multiple phrases used for the same concept, etc. In addition, automated techniques to process unstructured textual data on information systems are still only nascent [7]. Tuma *et al.* systematically summarized the existing literature on software threat analysis and observed that most techniques include misuse cases, threat patterns, and rule-based graph matches; however, none of them take a data-driven perspective [6].

Our study is related to, but different from, the abovementioned studies. This study aims to automatically identify the

patterns and characteristics of various types of vulnerabilities and evaluate their risks from a data-driven perspective. Furthermore, we investigate the associated solutions to provide guidance for firms and organizations. The dataset in our work is considerably more challenging than those employed in previous studies due to the various programming languages (e.g., C, C#, and SQL), multiple commenting languages (e.g., English and Chinese), and the existence of many URLs, misspellings, typos, and unconventional acronyms, which are commonly used in online forums.

B. TOPIC MODELING

Topic modeling is a type of statistical algorithm that can automatically extract abstract topics from a collection of text documents. Abstract topics are usually collections of words cooccurring in documents. These cooccurring words are semantically related according to the nature of the language. Thus, abstract topics can be represented by cooccurring words, while documents can be represented by the topics within them.

Before they can be processed and analyzed using computer programs, text data must be converted into numerical data. Bag of words (BOW), term frequency–inverse document frequency (TF–IDF), and Word2Vec are three commonly used techniques for this conversion. In BOW, only the presence of words in the document is noted, and the order of words is discarded [21]. TF–IDF is the product of two statistics: the TF and the IDF. The TF measures how frequently a word appears in a document. The IDF measures how much information can be provided by the word. Relatively less frequent words are considered more informative and important. Therefore, a specific word is more important if it has a higher TF–IDF value [22]. Word2Vec, in turn, can reconstruct the linguistic contexts of words using a two-layer neural network model. A large collection of documents is used as the input for the two-layer neural network model, and a vector space of several hundred dimensions may be generated as the output. All the words in the collection of documents could be assigned corresponding vectors in this space [23].

Commonly used algorithms for topic modeling include latent Dirichlet allocation (LDA), latent semantic analysis (LSA), and nonnegative matrix factorization (NMF). LDA is a general statistical model in which each document in the collection is a mixture of an underlying set of topics, and each topic is a mixture of an underlying set of words in the entire collection of documents [24]. LSA extends the vector space model by reducing the dimensionality of the term-document matrix using singular value decomposition (SVD). LSA first constructs a matrix using the counts of each unique word in each document in the collection. While the dimensionality of this matrix is reduced using SVD, the similarity structure among columns is preserved. Thus, related words can be grouped into topics. Finally, documents can be analyzed by comparing the cosine distance between different vectors [25]. In NMF, the matrix can be factorized into two matrices, such that no negative elements exist in all three matrices [26]. In this process, the document-term matrix can be constructed by using the weights of different words from the collection of documents. Then, this matrix can be factorized into a term-feature matrix and a feature-document matrix, which consists of clusters of related documents.

Topic modeling techniques have been applied to various research domains. For instance, in a bibliographic analysis, Xu *et al.* [27] utilized an attention mechanism to capture the relations among documents and learn semantic information on document discourse levels to judge the importance of sentences for the summarization of scientific literature. Hanif *et al.* [28] adopted topic modeling to refine similarities among topics in a bibliographic analysis. In an empirical analysis of social media and news articles, Yang *et al.* [29] proposed a language model-based topic clustering framework to analyze news articles. Shi [30] studied emotional analysis using a deep confidence neural network and a dual attentional model. The experimental results showed that the proposed model achieved the best results among the considered alternatives. Hong and Davison [31] trained a topic model using aggregate Twitter data to achieve a higher quality of learned model. In software engineering, Asuncion *et al.* [32] proposed an automated technique to combine traceability with topic modeling. In recommender systems, Wang and Blei [33] developed an algorithm to recommend scientific articles by combining collaborative filtering and probabilistic topic modeling. Hu and Ester [34] proposed a social topic model to capture both the social and topic aspects of user check-ins. This model achieved better performance than other state-of-the-art models in social network-based recommender systems. Jang *et al.* [35] proposed an algorithm to predict novel drug–phenotype associations and drug side-effect associations using topic modeling and natural language processing. Li *et al.* [36] predicted stock performance using a sentiment analysis of financial news shared on social media. Chen *et al.* [7] summarized the existing literature using topic modeling to mine software repositories.

III. DATA

A. CROWDSOURCED TESTING

Crowdsourced testing has become a new testing service mode for information security testing. Using an online platform that facilitates feedback and releases of information on security issues between enterprise information systems and cybersecurity researchers, cybersecurity users can submit information on security vulnerabilities that they detect in information systems, and enterprise users can learn about the vulnerabilities in their own information systems. This vulnerability detection mode utilizes the intelligence of a crowd of cybersecurity experts instead of a small group of security experts to detect vulnerabilities in information systems. Such platforms provide venues for software developers to exchange information on technology bugs. Interactive discussions about technology bugs on crowdsourced testing platforms promote software-development technologies. Compared with

traditional security vulnerability detection methods, the crowdsourced testing mode has the following advantages:

(1) Unlike traditional testing models that utilize only a small group of cybersecurity experts hired in enterprises, crowdsourced testing can explore security vulnerabilities in information systems from different levels and therefore can generate a dynamic, large, and comprehensive vulnerability feature library. This library is helpful for improving the efficiency and accuracy of vulnerability detection and characterization in a distributed environment.

(2) In the crowdsourced testing mode, enterprises can obtain high-value vulnerability information about their systems in a short time, which reduces expenses. Usually, it takes longer to detect vulnerabilities in the traditional vulnerability testing mode because R&D researchers and professional testing engineers need to be employed.

(3) Crowdsourced testing breaks through the limitations of limited testers and tools in a closed environment. Therefore, crowdsourced testing can effectively provide different testing environments, increase the number and efficiency of testers, and improve the effects of testing, leading to shorter detection times and lower costs of environment construction.

(4) Security issues are regularly inspected in the traditional security testing mode. In contrast, crowdsourced testing can continuously monitor and exploit information systems for enterprises. Thus, new security vulnerabilities can be detected in a timely manner.

(5) Crowdsourced testing utilizes the intelligence of many participating cybersecurity experts; thus, the test results should be more objective than those produced in the traditional testing mode, where hired researchers may report biased results.

(6) Cybersecurity experts participating in crowdsourced testing are from different geographical locations, use different language systems, and thus can provide support for location-based testing.

B. DATA SOURCE AND DATA COLLECTION

The data in this study are from a vulnerability coordination platform that utilizes crowdsourced penetration testers and cybersecurity experts. Through this platform, cybersecurity experts can report detected network vulnerabilities and suggested solutions for enterprise information systems.

When cybersecurity experts and penetration testers detect vulnerabilities, they post all the details and proofs of the vulnerabilities, the associated firms, and the suggested solutions on this platform.

Upon receiving a notification of potential vulnerabilities via this platform, the associated firms evaluate the vulnerabilities and determine their subsequent responses. After this evaluation, firms may attempt to address the vulnerabilities and then release all the details to the public. Alternatively, firms may ignore the vulnerabilities after completing their evaluation. This may be because the vulnerability is considered unimportant or the risk it poses is deemed low. Firms may ignore such notifications even without evaluating them because of a lack of resources.

Figure 2 shows the general sequence of actions in this platform.

We implement a web crawler using Python to collect all the data from this platform. The general framework of this web crawler is presented in Figure 3.

Considering the data structure and characteristics of the target website, we directly access the data interface of the website to collect the required data. We first check the types of target data, locate the respective APIs through packet grabbing, and simultaneously obtain the parameters for these APIs. We obtain or generate parameter values for this crawler and construct a URL containing the required resources. Thus, we can obtain the JSON strings, parse them, and write them to a data file. The web crawler sequentially requests the next URL until all the required URLs have been requested. The ''archive once interrupted'' function is designed and implemented to ensure the robustness of this web crawler. If a certain URL is requested more times than the upper bound, this crawler records the progress of the current request and the data file collected and automatically loads these files at next startup to retain the previous work.

Using this web crawler, we collected all the data attributes in this platform. Each vulnerability record has approximately 20 attributes. Table 1 shows the data structure of the most important attributes for each vulnerability record with sample values.

The vulnerability ID is the unique ID for each vulnerability record. The vulnerability title briefly describes the problem occurring in an information system. The associated firm indicates the firm whose information systems contain vulnerabilities. The vulnerability poster is the cybersecurity expert who detected the vulnerability. The post time, fix time, and release time refer to the time at which the vulnerability is posted on the platform, the time at which it is fixed, and the time at which details of the vulnerability are released to the public once it is fixed, respectively. The vulnerability type is the category of vulnerability specified by the poster. The risk level is the potential degree of risk specified by the poster. The vulnerability status is the current state of a certain vulnerability. Possible statuses include *checked by associated firms*, *fixed by associated firms*, *ignored by associated firms*, and *details of vulnerability released to the public*. The vulnerability disclosure status shows detailed timestamps indicating when this vulnerability was reported online, when the details were released to associated firms only, when the vulnerability was checked by the firms, when it was fixed by the firms, when it was released to the public, etc. The vulnerability brief description and vulnerability detailed description fields describe the vulnerability briefly and in detail, respectively. The vulnerability proof field shows why there is such a vulnerability in the information system of a firm. The suggested solution is the solution proposed by posters to potentially fix the vulnerability. Firm response indicates how

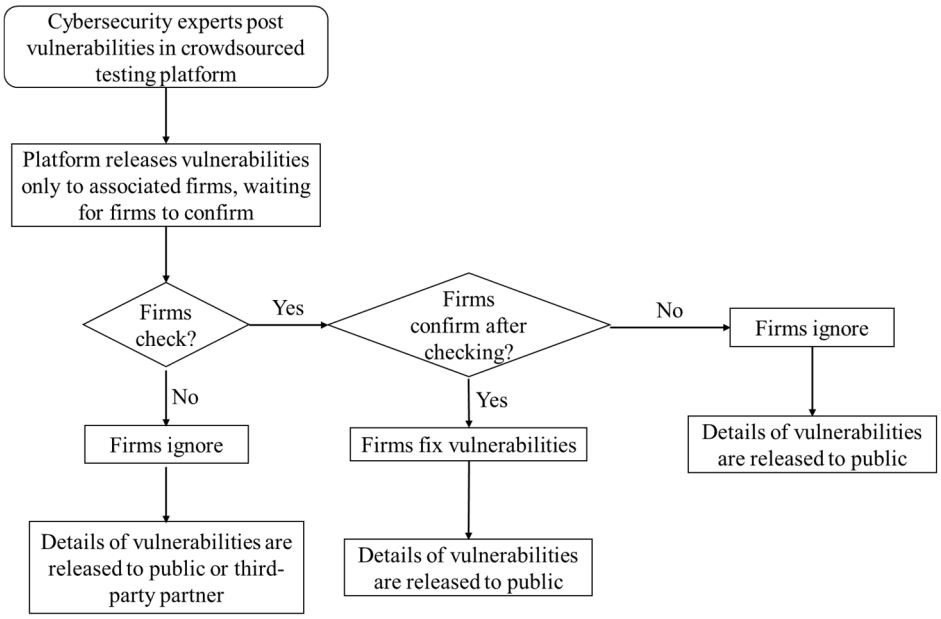

FIGURE 2. General sequence of actions on the vulnerability coordination platform.

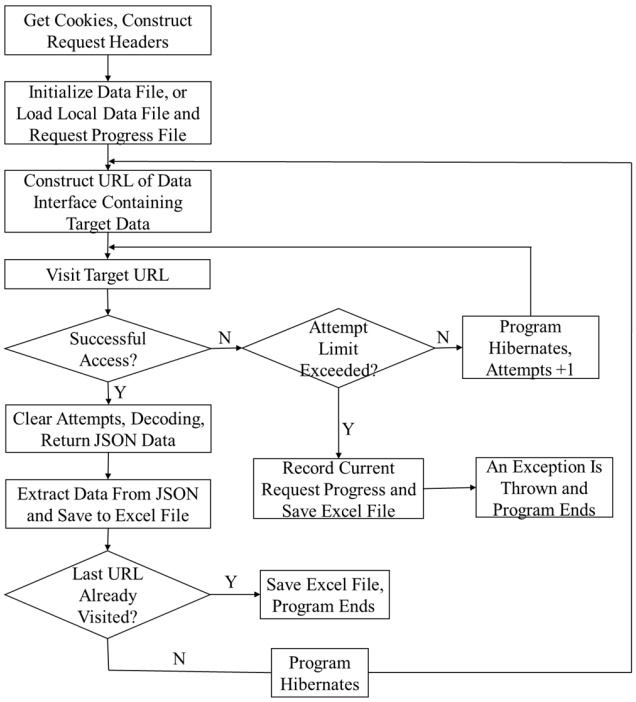

FIGURE 3. General framework of the web crawler.

TABLE 1. Data structure and sample values for vulnerability records.

Data Attribute	Sample Value[a]
Vulnerability ID	2014-62212
Vulnerability Title	Account leakage from Virtual currency OKCoin
Associated Firm	OKCoin
Vulnerability Poster	Lunlun
Post Time	May 25, 2014
Fix Time	June 28, 2014
Release Time	June 28, 2014
Vulnerability Type	Design Defect/Logic Error
Risk Level	High
Vulnerability Status	Vulnerability has been fixed by OKCoin
Vulnerability Disclosure Status	May 25, 2014: Vulnerability posted, waiting for firms' confirmation, May 25, 2014: Firm confirmed, details released to firms, June 4, 2014: details released to area expert, Jun 28, 2014, fixed by firms, details released to the public
Vulnerability Brief Description	OKCoin, Account problem in largest Chinese bitcoin trading platform
Vulnerability Detailed Description	Not shown due to space limits
Vulnerability Proof	Not shown due to space limits
Suggested Solution	Not shown due to space limits
Firm Response	Thanks, will fix it as soon as possible

firms respond to this vulnerability. Possible values include indicators that firms have evaluated the vulnerability and will handle it quickly or have transferred it to a third-party partner.

We collected the raw dataset containing the vulnerabilities within the information systems of firms using the web crawler and archived them in our database in August 2019. Each record in the database is a vulnerability having a unique vulnerability ID, vulnerability title, brief vulnerability description, detailed vulnerability description, vulnerability proof, suggested solution, vulnerability category, and risk level.

TABLE 2. Dataset summary.

Data Attribute	High Risk	Medium Risk	Low Risk	Total Count
SQL Injection	7127	2075	429	9631
Design Defect/Logic Error	3120	1186	679	4985
Sensitive Information Disclosure	2162	1011	661	3834
Cross-site Scripting (XSS)	1118	1306	1124	3548
Remote Code Execution	2583	548	129	3260
Unauthorized Access/Permission Bypass	1791	1042	379	3212
Weak Password	1894	676	187	2757
File Operation Vulnerability	1747	503	138	2388
Configuration Error	1425	490	325	2240
Successful Intrusion Event	1013	145	104	1262
Others	539	216	131	886
Untimely System/Service Patches	417	99	21	537
Cross-site Request Forgery	138	140	56	334
Malicious Information Dissemination	69	65	91	225
Denial of Service	44	72	53	169
URL Redirection	7	35	107	149

C. DATASET SUMMARY

Initially, we collected 39503 records of vulnerabilities in information systems, with 20 different attributes for each vulnerability record.

Most data are unstructured in this dataset. Both source codes and comments appear in the detailed vulnerability description, vulnerability proof, and suggested solutions. Given the complexity of different information systems, the source codes may be in different types of programing languages, e.g., SQL, Java, C, and C#. The comments are mainly in Chinese, but some may be in English. Moreover, the vulnerability descriptions, proofs, and suggested solutions in the dataset are of a complex and noisy nature (e.g., large numbers of misspellings, various linguistic patterns, and posted URL links). Figures may also appear in the detailed description, vulnerability proof, and suggested solution fields. These figures are mainly screenshots of source codes, execution outputs, or even user interfaces. The figures are not included in this study.

The raw data are preprocessed by removing duplicate and missing values. The noisy data are also cleaned and checked manually if necessary.

All vulnerability records are classified into 16 categories. Some categories, such as SQL injection and design defect/logic error, have more vulnerability records, whereas other categories, such as URL redirection and denial of service, have fewer records. The final dataset includes 39417 records in 16 categories. Table 2 summarizes the categories of all the vulnerability records, with counts at high-, medium-, and low-risk levels.

TABLE 3. Length of detailed descriptions, vulnerability proofs, and suggested solutions.

Statistics	Length of Detailed Description	Length of Vulnerability Proof	Length of Suggested Solution
Min	0	0	0
First Quartile	23	0	2
Mean	61	8	5
Third Quartile	149	53	14
Max	23358	21781	10666
Variance	605039.5	639077.9	4342.9

For each vulnerability record, multiple URLs and username/password pairs exist in the detailed descriptions, vulnerability proofs, and suggested solutions. Thus, the character count of these fields might be higher, as summarized in Table 3.

IV. FRAMEWORK

In this section, we present the general framework to characterize vulnerabilities in information systems via the textual data available on crowdsourced testing systems. Figure 4 shows the general approach. We first describe the overview of this framework and then explain each step in detail in this section.

A. FRAMEWORK OVERVIEW

The goal of this framework is to characterize in-depth the vulnerabilities firms face and possible solutions they could deploy to reduce losses from security issues as much as possible.

There are six main function modules in this framework. First, the raw data are preprocessed and prepared through, for instance, data cleansing, removal of missing values, duplicate values, and punctuation, and replacement of uppercase letters, among others. Common stop word lists in Chinese and English are adopted. Stop words are also generated specifically for this dataset using TF–IDF. Second, the categorization of vulnerabilities is standardized because the categorization is the ground truth in this research and the terms used by posters to describe vulnerability types are sometimes not standard. Third, features are extracted at three different levels: the word level (term frequency), the phrase level (TF–IDF with N-grams), and the record level (topics and record-level statistics, e.g., document part-of-speech counts). The Chinese and English texts are processed separately. The Chinese texts are processed through procedures including data cleansing, word segmentation, stop word removal, tokenization, and feature extraction. The English texts are processed through procedures including data cleansing, replacement of uppercase letters, removal of punctuation and stop words, tokenization, and feature extraction. Then, the models are trained and tested for vulnerability classification in module 4 and for risk-level prediction in module 5. Finally, in module 6, key clues for

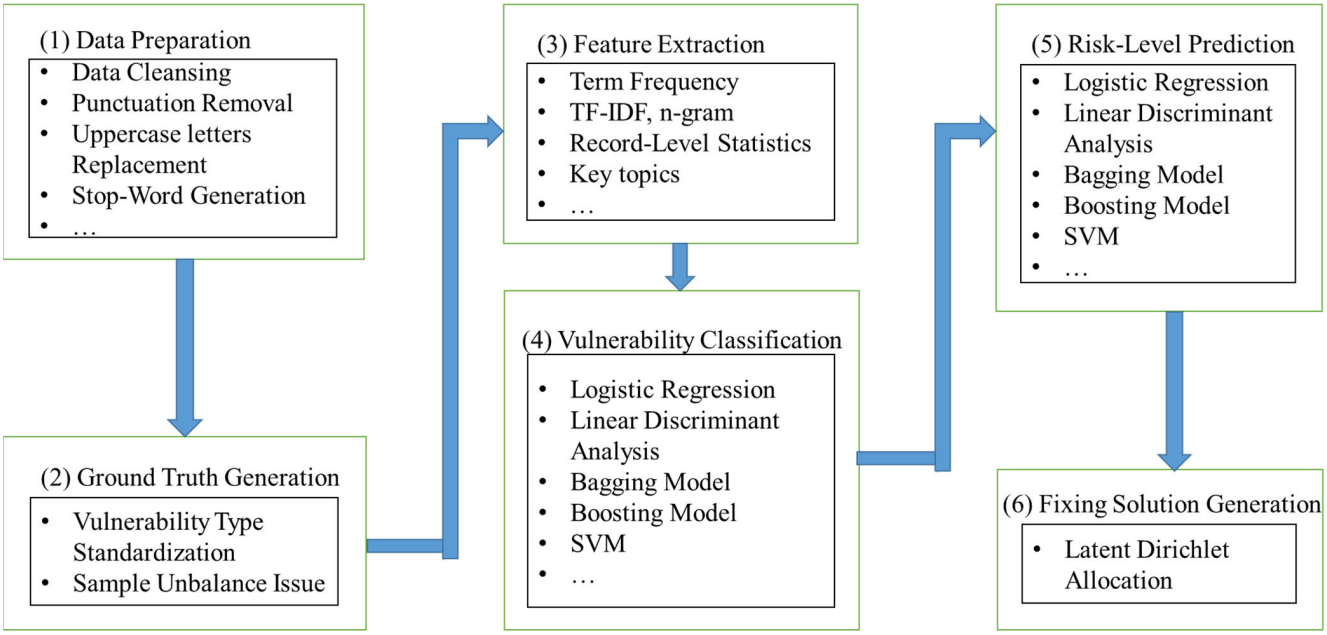

FIGURE 4. General framework.

dealing with vulnerabilities are automatically generated using topic analysis.

The details of all key procedures for processing the data, extracting the features, and training the models in this framework are explained in the following subsections.

B. DATA PREPARATION

Text preprocessing is a very important procedure for text analysis. The vulnerability dataset in this study contains large numbers of misspellings, special characters, various linguistic patterns, punctuation marks, posted URL links, etc. The raw data are preprocessed as follows. We present the general procedures and highlight key difference between Chinese and English during data processing.

1) CONVERSION OF TEXT DATA INTO LOWER CASE

The raw textual data in the vulnerability brief description, vulnerability detailed description, vulnerability proof, and suggested solution fields are converted to lower case to prevent word differences.

2) TOKENIZATION

The text data are then converted into words, e.g., tokens, which could identify interesting words from the dataset. The Chinese textual data are segmented using the Jieba module. The English textual data are segmented using spaces. Word segmentation is significantly different between Chinese and English. These tokens are the input for the following steps.

3) REMOVAL OF PUNCTUATION

This dataset has both Chinese and English punctuation marks. These punctuation marks are removed from the text dataset.

4) REMOVAL OF STOP WORDS

Due to the uniqueness of these data on vulnerabilities in information systems, we need to build our own stop-word list for this dataset. We first build a general stop-word list by combining four widely used Chinese stop-word lists (from the Harbin Institute of Technology, Baidu, Si Chuan University, and CN), the English long stop-word list, and the stop-word list for SQL from Rank.nl. For the specific dataset in this study, we then calculate the TF for each word (in both Chinese and English) and treat the words with extremely low and extremely high TFs as stop words. We then calculate the TF–IDF weights for each word and treat the words with a low TF–IDF weight as stop words. Finally, we combine these words with the commonly used Chinese stop-word lists and the English stop-word list into the final stop-word list. All stop words are removed from the dataset.

5) WORD NORMALIZATION

English words display rich transformations. The Porter stemmer algorithm is used to remove common morphological and inflectional endings from words in English.

C. FEATURE EXTRACTION

To extract features for the vulnerability detailed descriptions, we adopt the bag of words model and integrate it with TF–IDF and N-grams. We consider the order of words in vulnerability records. We also consider other record-level statistics, such as the word density of each record and the punctuation count. The detailed process of feature extraction is presented as follows.

1) BAG-OF-WORDS MODEL

The BOW model is commonly used to represent text features in information retrieval and natural language processing

tasks. In BOW, the occurrences of words in all documents are noted, but the order of words is ignored. Thus, the BOW model can generate a list of words with a word count per document. Finally, the BOW model generates a matrix representation of the vulnerability dataset, with each row indicating each vulnerability record and each column indicating a term in the dataset. Furthermore, each cell in the matrix represents the frequency of a specific term in the specified vulnerability record.

2) TF–IDF STATISTICS

In text analysis, the importance of a word increases if the word appears more frequently in the document, but it decreases if the word appears more frequently in the entire document collection. TF–IDF is a statistical measure used to evaluate the importance of a word to a document in a document collection. To avoid distortions from longer documents, the TF is normalized by the document length as in (1), where $tf_{i,j}$ is the normalized TF of word i in document j, and $n_{i,j}$ is the number of appearances of word i in document j.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

The IDF of a specific word can be calculated by dividing the total number of records by the number of records containing the word, as in (2), where idf_i is the IDF of word i, $|D|$ is the total number of documents in the document collection, and $\left|\left\{j : t_i \in d_j\right\}\right|$ is the total number of documents in which word i appears. The denominator is adjusted $\left|\left\{j : t_i \in d_j\right\}\right|+1$ to avoid the potential division-by-zero problem.

$$idf_i = \log \frac{|D|}{\left|\left\{j : t_i \in d_j\right\}\right| + 1} \tag{2}$$

Finally, the TF–IDF of word i can be calculated as the product of the TF and the IDF, as in (3).

$$tfidf_{i,j} = tf_{i,j} \times idf_i \tag{3}$$

3) N-GRAMS

The N-gram model is based on a sequence of n words from a given sequence of text. The model considers the order of words occurring in documents. For a certain sequence of text, a list of N-grams can be constructed by finding pairs of words that occur adjacent to each other. In this research, we adopt a bigram approach.

4) RECORD-LEVEL STATISTICS

Incorporation of other document-level statistics can improve the model performance. In this study, we calculate the following record-level statistics as features for the vulnerability document.

- Word count of the document (total number of words in the document);
- Part-of-speech count of the document (total numbers of the various parts of speech in the document);
- Average word density of the document (average length of the words used in the document);
- Punctuation count of the document (total number of punctuation marks in the document); and
- Uppercase letter count of the document (total number of upper case letters in the document).

D. VULNERABILITY UNDERSTANDING

This study adopts the following algorithms to classify vulnerabilities and predict their risk levels. The performance is compared among these algorithms.

1) LOGISTIC REGRESSION ANALYSIS (LR)

LR is a predictive model for assigning observations to a discrete set of classes. The cost function in LR can be defined as in (4).

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)\right] \tag{4}$$

where $\hat{y}_i = h_\theta(x_i)$ is the predicted value and y_i is the true value. The model parameters should be updated by minimizing the error to achieve the best regression performance. The model parameters are usually updated using the gradient-descent approach, which starts with random parameter values and iteratively updates them to attain the minimum cost.

2) LINEAR DISCRIMINANT ANALYSIS (LDA)

The general idea of linear discriminant analysis is to project high-dimensional samples into the best vector space, such that samples in the new vector space have the greatest distance between categories, i.e., $S_{between}$, but the smallest distance within categories, i.e., S_{within}. $S_{between}$ can be defined as in (5), and S_{within} can be defined as in (6), where n_i is the sample size of category i, x_k is the kth sample, and c is the number of categories.

$$S_{between} = \sum_{i=1}^{c} n_i (u_i - u)(u_i - u)^T \tag{5}$$

$$S_{within} = \sum_{i=1}^{c} \sum_{x_k \in classi} (u_i - x_k)(u_i - x_k)^T \tag{6}$$

Fisher's linear discriminant rule is used to maximize the ratio between $S_{between}$ and S_{within}, which can be defined as in (7), where φ is a column vector.

$$J_{fisher}(\varphi) = \frac{\varphi^T S_{between} \varphi}{\varphi^T S_{within} \varphi} \tag{7}$$

3) K-NEAREST NEIGHBOR (KNN)

KNN is a nonparametric approach for classification and regression. In classification, the category of each object can be defined by the most common category among its k nearest neighbors. Thus, KNN is an instance-based learning approach that is sensitive to the structure of data samples.

4) CLASSIFICATION AND REGRESSION TREE (CART)

CART uses a decision tree to move from observations stored in branches to categories stored in leaves. This decision tree can be trained by splitting the dataset based on classification features. Dataset splitting is conducted recursively such that

all the data in the same data subset have the same category; otherwise, the splitting does not improve classification performance. The Gini impurity indicator can be used to measure the data purity: the indicator is lowest when the dataset is purest. The Gini impurity is computationally efficient because no log calculation is required.

5) SUPPORT VECTOR MACHINE (SVM)

SVM is a supervised learning algorithm that can recognize patterns for classification tasks. Each data point can be considered a point in the feature space, and the points of different categories are divided by the gap between the categories. Thus, the test data are mapped into this feature space and classified into the same category to which the data points belong.

A given training dataset with two categories $D = \{(x_i, y_i) | x_i \in \mathbb{R}^p, y_i \in (-1, 1)\}, i = 1:n$ contains n data points in a p dimensional space. y_i indicates the category to which data point x_i belongs. The target is to find the hyperplane $wx - b = 0$ on which the data points are separated as widely as possible. The optimization problem with a software margin can be defined as in (8).

$$\max_{y_i(wx_i-b)\geq 1-\xi_i, \xi_i \geq 0, i=1:n} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \tag{8}$$

Different kernel functions, e.g., polynomial, Gaussian radial basis, and hyperbolic, could be used.

6) BAGGING MODEL

The general idea of a bagging model is to create an ensemble of multiple meta-algorithms to improve performance. Such a model could improve stability and decrease the generalization error and variance. For a given dataset with a size of n, the model could generate m new datasets with a size of n' by uniformly sampling with replacement. m models could be trained using m new datasets. The final classification could be performed by voting.

7) BOOSTING MODEL

The general idea of the boosting model is to convert weak classifiers into strong ones by adjusting the weights of each weak classifier depending on its accuracy. After adding a new weak classifier, higher weights are assigned to misclassified data, and lower weights are assigned to correctly classified data. Thus, new models focus on misclassified observations.

E. TEXT ANALYSIS

Latent Dirichlet Allocation (LDA) is adopted in this study to analyze the details of vulnerabilities and the suggested solutions. LDA assumes that each document is a random mixture of topics in a multinomial distribution. Each topic is a mixture of vocabulary words in a multinomial distribution. Therefore, a document collection D containing M documents of length N_i for $i \in \{1, \ldots, M\}$ could be generated as follows: from a Dirichlet distribution with parameter α, choose the topic distribution for document i: $\theta_i \sim \text{Dir}(\alpha)$ with $i \in \{1, \ldots, M\}$, and from a Dirichlet distribution with parameter β, choose the word distribution for topic k: $\varphi_k \sim \text{Dir}(\beta)$ with $k \in \{1, \ldots, K\}$.

Then, for the j^{th} word in document i, with $i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, N_i\}$, a topic is first chosen as in (9).

$$z_{i,j} \sim Multinomial(\theta_i) \tag{9}$$

Then, a word is chosen as in (10).

$$w_{i,j} \sim Multinomial(\varphi_{z_{i,j}}) \tag{10}$$

The parameters α and β are first randomly initialized to train the LDA model. Then, we calculate the topic distribution and word distribution for all the documents. Next, the optimal α can be derived using the Newton–Raphson method. β can be derived via normalization. This process continues until α and β converge.

V. EXPERIMENTS AND RESULTS

This section presents the results related to (1) the vulnerability classification, (2) the risk-level prediction, and (3) potential fixes for various types of vulnerabilities. All experimental results are assessed in terms of the AUC. Detailed discussions are provided to illustrate the benefits of this framework.

A. DATA PREPARATION AND FEATURE EXTRACTION

The raw data are cleaned by removing noise, e.g., missing values, outliers, and records with extremely short or long content. Some records may include extremely long detailed descriptions due to multiple URLs of attacked webpages, many username/password pairs, and so on. Other records may include extremely short detailed descriptions, such as ''SQL injection in webpage'' or ''See it when access''. These descriptions do not include key information about the nature of the vulnerabilities. Thus, records with extremely long or extremely short detailed descriptions are removed in this research. More specifically, records with less than 20 characters or with more than 3000 characters are removed. Records with detailed descriptions of medium length are considered in this work.

The sample size is different among different types of vulnerabilities. Vulnerability types with few records, e.g., fewer than 1000 records, are removed to prevent sample imbalance. The data may lack representativeness if the sample size is too small. We end up with 10 types of vulnerabilities in the experiments. After data preprocessing, there are 32199 records related to 10 different types of vulnerabilities. The SQL injection category has the most records (8106), and the successful intrusion event category has the fewest records (1051). The average record count for each type of vulnerability is approximately 3220.

The stop-word list is constructed by combining the commonly used Chinese and English stop-word lists and the list specifically generated for this dataset by considering TF–IDF values. Features are extracted from three dimensions: TD–IDF with bigrams, key topics, and record-level statistics, as discussed in detail in Section IV. We conduct

TABLE 4. Vulnerability classification performance.

	LR	LDA	KNN	CART	Bagging Model	Boosting Model	SVM
SQL Injection	87.36%	86.88%	66.10%	73.39%	86.35%	89.20%	69.29%
Cross-site Scripting (XSS)	90.25%	89.51%	60.34%	77.45%	88.88%	91.09%	89.19%
Weak Password	86.69%	85.93%	67.48%	72.94%	85.17%	86.41%	87.28%
Successful Intrusion Event	74.87%	69.42%	55.77%	59.86%	73.80%	74.03%	71.06%
Sensitive Information Disclosure	72.13%	70.91%	52.76%	58.15%	70.08%	70.79%	70.67%
File Operation Vulnerability	80.33%	78.26%	56.98%	66.12%	78.53%	79.89%	79.51%
Configuration Error	68.26%	66.25%	58.05%	58.13%	65.74%	69.47%	69.30%
Design Defect/Logic Error	73.73%	73.03%	62.45%	62.27%	72.59%	76.36%	73.14%
Remote Code Execution	82.45%	79.85%	61.88%	68.07%	81.94%	84.17%	79.21%
Unauthorized Access/Permission Bypass	66.27%	66.44%	54.65%	58.58%	65.99%	65.88%	67.11%
Average	80.45%	78.92%	60.53%	67.14%	79.08%	81.11%	74.63%

experiments for different topic numbers (10, 20, 30, 40, and 50) in this research. We present the classification and prediction results when the topic number is 30 for the following reasons: (1) The classification and prediction performance are similar for different topic numbers. (2) The choice of 30 topics is standard in this line of research, according to domain experts. All of the key topics are included when we choose 30 topics. For these reasons, the experimental results considering 10, 20, 40 and 50 topics are not presented due to space limits. However, these results are available upon request. A 10-fold cross-validation approach is applied.

B. VULNERABILITY CLASSIFICATION

The experimental results obtained from several popular algorithms are reported in Table 4. The seven columns correspond to the seven classification algorithms. We observe that most of the popular algorithms achieve good average classification performance for all vulnerability types, with the boosting model showing the best performance in this regard. The boosting model builds on weak classifiers to yield a final strong classifier by adjusting the weights so that future weak classifiers can focus on examples that have been misclassified by previous weak classifiers. The LR and boosting models both achieve an average classification performance higher than 80% in AUC among all the vulnerability categories. Linear discriminant analysis and the bagging model achieve an average classification performance close to 80% in AUC among all the vulnerability categories. The average classification performance of SVM is approximately 75% among all the vulnerability categories. The average classification performance of KNN and CART is relatively low in AUC.

In general, the overall average classification performance in AUC is acceptable. For instance, the average classification performance in AUC is higher than 80% for the XSS and weak password categories. The average classification performance in AUC is higher than or close to 75% for the SQL injection, remote code execution, file operation vulnerability, and design defect/logic error categories. The classification performance in AUC is relatively lower but still close to or higher than 65% for the successful intrusion event, sensitive information disclosure, configuration error, and unauthorized access/permission bypass categories.

This relatively low classification performance is because of the complexity of these four types of vulnerabilities. Successful intrusion events are a very broad concept including multiple types of intrusion events, leading to a blurry pattern and low classification rate. The low classification rate for the unauthorized access/permission bypass category relates to the fact that vulnerability records include patterns of multiple types, and thus, they could also be categorized into other types, which can be shown by the similarity of the results of the topic analysis among different vulnerability types. For instance, some vulnerabilities are categorized as unauthorized access/permission bypass vulnerabilities because files are accessed without permission; thus, these records could also be categorized as file operation vulnerabilities. Some instances of unauthorized access are due to design defects in information systems; thus, these records could also be categorized as design defects/logic errors. There exist multiple reasons for the disclosure of sensitive information. For instance, SQL injections, XSS or weak passwords may lead to the disclosure of sensitive information in certain cases. Configuration errors usually involve human factors. Thus, the patterns in the configuration error category are not obvious, leading to low classification performance.

C. RISK-LEVEL PREDICTION

The performance of the algorithms in terms of risk-level prediction of the various types of vulnerabilities is reported in Table 5. The seven columns correspond to seven popular prediction models.

The same dataset is used to train the models to predict the risk level of each vulnerability record. The bagging model and boosting model both achieve a prediction performance higher than 75%. The prediction performance is higher than 60% for LR, linear discriminant analysis, CART and SVM. The performance of KNN is lower.

The overall prediction performance is not as high as that presented in a well-structured dataset owing to the sample imbalance among different risk levels for most vulnerability categories. For instance, the prediction performance is low for the unauthorized access/permission bypass category. This is because of the extremely serious sample imbalance, i.e., 56.79% of records are high-risk vulnerabilities, 31.8%

TABLE 5. Risk-level prediction.

	LR	LDA	KNN	CART	Bagging Model	Boosting Model	SVM
SQL Injection	68.58%	66.87%	55.90%	66.14%	84.03%	86.47%	66.31%
Cross-site Scripting (XSS)	63.34%	60.06%	62.12%	64.02%	77.70%	79.11%	66.92%
Weak Password	62.70%	58.93%	56.55%	60.07%	78.25%	77.53%	62.28%
Successful Intrusion Event	65.91%	56.10%	60.74%	66.17%	84.12%	82.94%	67.33%
Sensitive Information Disclosure	64.99%	61.83%	59.33%	57.36%	66.32%	65.23%	65.40%
File Operation Vulnerability	62.45%	57.16%	52.81%	59.05%	73.95%	76.13%	62.32%
Configuration Error	73.34%	70.41%	56.75%	58.32%	74.52%	75.02%	72.97%
Design Defect/Logic Error	63.87%	63.35%	55.69%	60.10%	68.79%	72.66%	65.32%
Remote Code Execution	69.27%	65.81%	61.25%	70.16%	86.90%	87.41%	71.43%
Unauthorized Access/Permission Bypass	63.70%	60.72%	58.59%	52.03%	62.43%	66.80%	64.05%
Average	66.02%	63.18%	57.56%	61.88%	76.13%	77.83%	66.22%

are medium-risk, and 11.4% are low-risk. Serious sample imbalance issues also exist in the sensitive information disclosure, design defect/logic error, configuration error, and other categories.

In terms of prediction performance among different vulnerability types, the performance is relatively high for the remote code execution, SQL injection, successful intrusion event, XSS, and configuration error categories. The overall prediction rate is close to or higher than 70%. In contrast, the prediction rate is relatively low for the sensitive information disclosure, unauthorized access/permission bypass, file operation vulnerability, and design defect/logic error categories. The prediction rate is medium for the weak password category.

The low prediction performance for sensitive information disclosure is because sensitive information may be in different forms, such as passwords, keys, certificates, authorization credentials, personal data, configuration files, log files, and backup files. Thus, sensitive information disclosure might occur in various cases, leading to lower prediction performance. Design defects/logic errors are a broad concept that includes several cases. Significant patterns in logic vulnerabilities, which generally appear in functions and business processes, cannot be identified. For instance, a negative value for prices and quantities of products would be a design defect in information systems. There exist multiple patterns in the unauthorized access/permission bypass category; thus, overlap may exist between vulnerabilities of this and other types. For example, unauthorized access/permission bypass may occur when files are accessed without permission, leading to overlap with file operation vulnerability. Design defects in information systems may also result in unauthorized accesses. This may explain the low prediction rate for the unauthorized access/permission bypass category. For file operation vulnerability, different file operations, e.g., file inclusion, reading, writing, modification, and upload, among others, lead to different vulnerability characteristics. Some vulnerabilities may closely relate to other types of vulnerabilities, e.g., SQL injections.

In addition, the diversity of programming languages and comment languages increases the difficulty of accurate risk-level prediction. Generally, our dataset, especially the detailed description, vulnerability proof, and suggested solution data attributes, are relatively complex and noisy due to large numbers of misspellings, various linguistic patterns, posted URL links, the mixture of both Chinese and English sentences, the use of multiple programming languages, etc. Our prediction results are not expected to be as accurate as the results of studies in the existing literature associated with well-structured forums. In addition, this study considers several types of vulnerabilities, which increases the prediction difficulty.

D. AUTOMATIC SOLUTION GENERATION

In this subsection, we conduct text analysis on the vulnerabilities and their associated suggested solutions. The topic analysis of these associated solutions helps illuminate potential fixes of various vulnerabilities. We extract English and Chinese texts separately into different corpuses because of significant differences between these languages. We build general stop-word lists by combining four commonly used Chinese stop-word lists and the long stop-word list from Rank.nl. We also specifically generate stop words for our dataset according to the TF–IDF weights. We also adopt the stop-word list for SQL from Rank.nl.

Through topic analysis using LDA, we can automatically extract key topics for vulnerabilities and the suggested solutions for each type of vulnerability. In the experiments, we choose 10 topics for each type of vulnerability and 10 keywords for each topic. Due to space limits, we cannot present all the keywords for all types of vulnerabilities. We present topic analyses for three types of vulnerabilities only and discuss them as follows. The topic analyses for other types of vulnerabilities are available upon request.

Table 6 shows the details of the topic analysis for the SQL injection category. We observe that the main reason for SQL injections is lax filtering of query inputs, based on which additional queries may be constructed, leading to unauthorized information leakage. In Table 6, topics 0 to 5 and topics 8 and 9 indicate common keywords and common injection points, e.g., PHP and ASP pages, used in the construction of injection instructions. Topic 6 indicates the key reason for SQL injections: the lax filtering of user inputs. Topic 7 indicates the most common targets of SQL injections: user

TABLE 6. Topic analysis of SQL injections.

topic 0	list	php	news	asp	product	sql	class	注入	article	search
topic 1	select	php	type	user	cat	code	schema	count	union	区域
topic 2	jsp	login	info	bo	var	code	区域	game	user	char
topic 3	data	document	script	email	length	subst	node	function	child	attribute
topic 4	null	info	based	sql	type	blind	char	payload	mysql	time
topic 5	code	区域	注入	aspx	参数	存在	web	php	type	city
topic 6	注入	sql	存在	php	漏洞	数据库	一个	测试	过滤	参数
topic 7	注入	密码	admin	信息	漏洞	后台	sql	存在	系统	登录
topic 8	accept	user	content	application	html	agent	zh	code	host	post
topic 9	sql	db	code	post	str	key	区域	php	string	user

TABLE 7. Topic analysis of solutions for SQL injections.

topic 0	data	document	length	subst	function	email	node	child	attribute	script
topic 1	过滤	加强	严格	字符	关键字	特殊字符	敏感	waf	做好	权限
topic 2	注入	null	指教	修改	密码	sql	口令	漏洞	修补	code
topic 3	sql	语句	注入	使用	数据库	变量	用户	查询	防御	编译
topic 4	应该	知道	代码	问题	一个	user	管理员	漏洞	管理	开发
topic 5	专业	add	slashes	数据库	限制	转换	权限	mysql	函数	判断
topic 6	一下	安全	补丁	程序	网站	cms	更新	不会	删除	升级
topic 7	参数	过滤	懂得	进行	转义	查询	提交	强制	类型转换	相关
topic 8	过滤	输入	用户	进行	检查	验证	建议	数据	使用	注入
topic 9	修复	礼物	rank	注入	厂商	升级	漏洞	sql	版本	waf

TABLE 8. Topic analysis of design defects/logic errors.

topic 0	文件	dll	一个	exe	asp	aaaaaa	测试	程序	qq	网站
topic 1	data	document	email	function	script	subst	length	child	attribute	node
topic 2	user	code	区域	password	login	用户	admin	php	登录	action
topic 3	content	accept	code	type	user	application	区域	zh	form	post
topic 4	密码	验证码	重置	输入	修改	邮箱	手机号	找回	用户	手机
topic 5	file	php	key	code	post	data	string	str	文件	path
topic 6	用户	信息	修改	一个	发现	登录	密码	订单	账号	问题
topic 7	php	web	文件	区域	code	xml	漏洞	执行	shell	url
topic 8	code	区域	html	false	url	script	li	frame	alert	var
topic 9	qq	手机	支付	进行	一个	使用	数据	没有	用户	问题

information and passwords. For instance, a common reason for SQL injection is the use of an admin account with a weak password.

Through the topic analysis of the suggested solutions for SQL injection, we observe that for topics 1, 7, and 8, the suggestions involve filtering queries from users. Characters, keywords, special characters, and possible escape characters should be processed and possibly mapped to certain types of vulnerabilities. These are important for avoiding SQL injection vulnerabilities. For topics 3 and 7, the suggested solutions also include parameterizing query requests from users. In addition, the analysis of topic 2 indicates that weak passwords and exposed passwords should be modified immediately in the case of an SQL injection. Passwords should be saved in a non-plaintext way to reduce possible losses after SQL injections. The analysis of topics 4 and 5 suggests that admin and normal user accounts should be significantly differentiated. For topics 6 and 9, suggestions include the use of an upgraded security patch to increase the security level.

Table 8 presents the topic analysis of vulnerabilities related to design defects/logic errors. Most design defect/logic error vulnerabilities are closely related to lax verification. Topics 4 and 6 are related to verification, e.g., passwords, verification codes, and ways of sending or resetting verification codes. The analysis of topic 2 suggests that design defect/logic error vulnerabilities may occur due to inappropriate login designs. The analysis of topics 0 and 7 show that design defects may lead to problems with uploads and operation of the database. The other topics are mainly related to keywords, codes, and functions in various programming languages.

Table 9 suggests that the key to addressing vulnerabilities related to design defects/logic errors is to strengthen and improve user authentication and verification mechanisms. The analysis of topics 7, 8, and 9 suggests enforcing a limit on user permissions, e.g., the need to use mobile phones, security issues, passwords, and authentication codes, and enforcing a limit on login frequency. For topics 0, 3, and 5, suggestions include enforcing user access control. Within topic 2,

TABLE 9. Topic analysis of solutions for design defects/logic errors.

topic 0	修复	权限	漏洞	问题	知道	控制	一下	没有	简单	应该
topic 1	data	document	email	function	script	方法	length	child	subst	node
topic 2	验证	过滤	进行	校验	用户	判断	参数	权限	服务端	token
topic 3	专业	逻辑	修改	安全	url	代码	验证	加个	密码	设置
topic 4	不要	支付	金额	数据	信息	订单	返回	校验	直接	进行
topic 5	礼物	验证	rank	加强	添加	完善	提交	机制	逻辑	安全
topic 6	code	区域	删除	应该	测试	md	文件	null	正则	user
topic 7	用户	密码	问题	修改	需要	网站	一个	手机	绑定	建议
topic 8	增加	验证	限制	登录	厂商	后台	口令	验证码	联系	修改
topic 9	验证码	限制	次数	设置	密码	短信	重置	懂得	错误	数字

TABLE 10. Topic analysis of file operation vulnerabilities.

topic 0	file	upload	php	type	editor	code	上传	区域	files	path
topic 1	php	文件	漏洞	下载	pass	wd	code	区域	任意	存在
topic 2	data	document	script	function	email	length	string	uc	child	return
topic 3	login	sb	var	bin	user	bash	daemon	root	lib	spool
topic 4	code	区域	user	password	local	jsp	site	tomcat	server	jd
topic 5	app	map	link	jsp	test	file	config	web	code	区域
topic 6	上传	shell	文件	直接	漏洞	一个	发现	密码	后台	admin
topic 7	file	web	path	jsp	download	filename	xml	文件	string	code
topic 8	jsp	upload	有限公司	系统	公司	admin	action	页面	集团	区域
topic 9	content	form	data	accept	disposition	type	application	php	zh	user

TABLE 11. Topic analysis of solutions for file operation vulnerabilities.

topic 0	参数	php	白名单	path	file	上传	文件夹	修改	upload	使用
topic 1	限制	上传	文件	进行	路径	下载	后缀	判断	验证	类型
topic 2	配置	web	正确	console	mx	data	document	server	string	length
topic 3	厂商	code	修复	代码	官方	联系	知道	区域	xml	upload
topic 4	过滤	上传	参数	加强	下载	后台	注入	严格	代码	jsp
topic 5	升级	礼物	漏洞	修复	版本	安全	系统	rank	建议	问题
topic 6	权限	目录	上传	控制	设置	执行	禁止	文件	访问	web
topic 7	专业	懂得	应该	验证	修改	问题	添加	增加	危险	允许
topic 8	删除	shell	文件	漏洞	升级	修补	网站	测试	php	信息
topic 9	修复	漏洞	密码	修改	补丁	口令	不要	安全	用户	一个

authorization and tokens appear as fixes to prevent possible ultra vires originating from logic errors. The analysis of topic 4 suggests conducting multiple verifications.

Table 10 presents the topic analysis for file operation vulnerability. File operation vulnerability includes the upload and download of arbitrary files and file inclusion vulnerabilities. Topics 0 and 8 list the common file types, e.g., PHP and JSP, that are subject to file operation vulnerabilities. Attackers can insert malicious codes into PHP and JSP files, upload them to servers, and cause damage to victims once the codes are executed. Topics 1 and 6 are related to the download of arbitrary files. Attackers can exploit such vulnerabilities, e.g., illegally obtain passwords, to download files that should not have been downloaded from any drive on the servers. Topics 2 and 5 show the possible files that might be exploited. Topics 4, 7, and 9 show multiple cases in which file operation vulnerabilities occurred.

In terms of solutions for file operation vulnerabilities, the analysis of topic 1 reveals suggestions for limiting the types of files to upload, carefully verifying download request links, and filtering out upload requests for sensitive file types and inappropriate download requests. Certain types of files, e.g., PHP and JSP, may include malicious codes. The suggested fixes related to topic 1 also seek to prevent the possibility of file traversal and arbitrary file downloads by saving files in a database and only accepting the file ID for downloads. On topics 1 and 6, suggestions include setting limits to user access to certain directories, verifying user access before downloading files, denying download permission for files from sensitive directories, preventing file traversal or arbitrary file download, and preventing the execution of malicious code from uploaded files. The analysis of topic 4 also suggests filtering parameters during file uploads by users. The analysis of topic 0 indicates that upload file types should be managed on a white list. Suggested fixes related to topic 8 include upgrading middleware, because a possible reason for arbitrary file downloads is the use of outdated middleware on websites. The analysis of topic 9 suggests changing passwords that are already exposed or less secure. Suggestions for topics 2, 3, and 5 include setting appropriate limits on server configuration and updating security patches and middleware. Suggestions under topic 7 include conducting verification using a white list.

VI. CONCLUSIONS

This study aimed to understand the real vulnerabilities that occur in firms' information systems and the associated fixes for these vulnerabilities. To this end, we built a general framework that can automatically categorize vulnerabilities, analyze their risk levels, and deeply evaluate the key topics related to each vulnerability category and the associated solutions. Data were collected from a vulnerability report platform, where real vulnerabilities that have actually appeared and their associated solutions were reported. We extracted features related to the vulnerabilities at three levels: the word, phrase, and record levels (with topic- and record-level statistics). We combined commonly used stop-word lists for both Chinese and English and generated stop words specific to our dataset as well. The experimental results showed that the general framework achieves an overall good performance in vulnerability classification, risk-level prediction, and extraction of key clues for addressing vulnerabilities.

This study extends the vulnerability literature by applying an artificial intelligence approach to vulnerabilities that have emerged in information systems. Taking a data-driven perspective, this research contributes to the existing literature by designing a general framework to characterize in-depth the real vulnerabilities that firms are facing. Unlike existing work, which mainly examines vulnerability identification, this research investigates different types of vulnerabilities, the potential risks they may pose, and possible immediate fixes. This study also has practical implications. Firms can apply this framework to better understand the vulnerability issues they face in terms of, e.g., their modes, patterns, risk levels, and possible solutions. Thus, firms can defend their information systems and ensure the security of their assets using the proposed framework.

This study is not without limitations. For instance, data on other types of vulnerabilities could be collected and analyzed in the future. Other state-of-the-art techniques could be integrated, and their performance could be tested.

REFERENCES

[1] M. Rouse. (2019). Vulnerability (information technology). TechTarget. Accessed: Mar. 25, 2020. [Online]. Available: https://whatis.techtarget.com/definition/vulnerability

[2] *Vulnerabilities in Online Banking Applications*, Positive Technologies, London, U.K., 2019.

[3] A. V. Revnivykh and A. M. Fedotov, "Main reasons of information systems vulnerability," *Global J. Pure Appl. Math.*, vol. 12, no. 3, pp. 2133–2142, 2016.

[4] A. Hovsepyan, R. Scandariato, W. Joosen, and J. Walden, "Software vulnerability prediction using text analysis techniques," in *Proc. 4th Int. workshop Secur. Meas. Metrics (MetriSec)*, 2012, pp. 7–10.

[5] F. G. Bulut, H. Altunel, and A. Tosun, "Predicting software vulnerabilities using topic modeling with issues," in *Proc. 4th Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2019, pp. 739–744.

[6] K. Tuma, G. Calikli, and R. Scandariato, "Threat analysis of software systems: A systematic literature review," *J. Syst. Softw.*, vol. 144, pp. 275–294, Oct. 2018.

[7] T.-H. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Softw. Eng.*, vol. 21, no. 5, pp. 1843–1919, Oct. 2016.

[8] F. Farahmand, S. B. Navathe, P. H. Enslow, and G. P. Sharp, "Managing vulnerabilities of information systems to security incidents," in *Proc. 5th Int. Conf. Electron. Commerce (ICEC)*, 2003, pp. 348–354.

[9] J. Radianti and J. Gonzalez, "Understanding hidden information security threats: The vulnerability black market," in *Proc. 40th Annu. Hawaii Int. Conf. Syst. Sci. (HICSS)*, 2007, pp. 1–10.

[10] B. Gao and L. Shi, "Modeling an attack-mitigation dynamic game-theoretic scheme for security vulnerability analysis in a cyber-physical power system," *IEEE Access*, vol. 8, pp. 30322–30331, 2020.

[11] K. Ding, Y. Qian, Y. Wang, P. Hu, and B. Wang, "A data-driven vulnerability evaluation method in grid edge based on random matrix theory indicators," *IEEE Access*, vol. 8, pp. 26495–26504, 2020.

[12] G. Li, J. Wu, J. Li, Z. Guan, and L. Guo, "Fog computing-enabled secure demand response for Internet of energy against collusion attacks using consensus and ACE," *IEEE Access*, vol. 6, pp. 11278–11288, 2018.

[13] Z. Guan, X. Liu, L. Wu, J. Wu, R. Xu, J. Zhang, and Y. Li, "Cross-lingual multi-keyword rank search with semantic extension over encrypted data," *Inf. Sci.*, vol. 514, pp. 523–540, Apr. 2020.

[14] X. Zhang, A. Tsang, W. T. Yue, and M. Chau, "The classification of hackers by knowledge exchange behaviors," *Inf. Syst. Frontiers*, vol. 17, no. 6, pp. 1239–1251, Dec. 2015.

[15] Y. Cong and J. Romero, "On information systems complexity and vulnerability," *J. Inf. Syst.*, vol. 27, no. 2, pp. 51–64, Dec. 2013.

[16] A. K. Ghoujdi, *Exploit Prediction and Vulnerability Clustering Using Text Mining*. Saarbrücken, Germany: LAP LAMBERT Academic Publishing, 2013.

[17] C. Liu, M. Guo, L. Peng, J. Guo, S. Yang, and J. Zeng, "Artificial immunity-based model for information system security risk evaluation," in *Proc. Int. Conf. E-Health Netw. Digit. Ecosyst. Technol. (EDT)*, vol. 1, Apr. 2010, pp. 39–42.

[18] S. Samtani, R. Chinn, and H. Chen, "Exploring hacker assets in underground forums," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, May 2015, pp. 31–36.

[19] J. Ruohonen, "Classifying Web exploits with topic modeling," in *Proc. 28th Int. Workshop Database Expert Syst. Appl. (DEXA)*, Aug. 2017, pp. 93–97.

[20] A. Abbasi, W. Li, V. Benjamin, S. Hu, and H. Chen, "Descriptive analytics: Examining expert hackers in Web forums," in *Proc. IEEE Joint Intell. Secur. Informat. Conf.*, Sep. 2014, pp. 56–63.

[21] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, nos. 1–4, pp. 43–52, Dec. 2010.

[22] J. Ramos, "Using TF-IDF to determine word relevance in document queries," in *Proc. 1st Instructional Conf. Mach. Learn.*, vol. 242, 2003, pp. 133–142.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[24] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[25] S. T. Dumais, "Latent semantic analysis," *Annu. Rev. Inf. Sci. Technol.*, vol. 38, no. 1, pp. 188–230, 2004.

[26] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*. Cambridge, MA, USA: MIT Press, 2001, pp. 556–562.

[27] H. Xu, Z. Wang, and X. Weng, "Scientific literature summarization using document structure and hierarchical attention model," *IEEE Access*, vol. 7, pp. 185290–185300, 2019.

[28] O. Hanif, Z. Donghua, W. Xuefeng, and M. S. Nawaz, "Refining the measurement of topic similarities through bibliographic coupling and LDA," *IEEE Access*, vol. 7, pp. 179997–180011, 2019.

[29] P. Yang, W. Li, and G. Zhao, "Language model-driven topic clustering and summarization for news articles," *IEEE Access*, vol. 7, pp. 185506–185519, 2019.

[30] M. Shi, "Research on parallelization of microblog emotional analysis algorithms using deep learning and attention model based on spark platform," *IEEE Access*, vol. 7, pp. 177211–177218, 2019.

[31] L. Hong and B. D. Davison, "Empirical study of topic modeling in Twitter," in *Proc. 1st Workshop Social Media Analytics (SOMA)*, 2010, pp. 80–88.

[32] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. (ICSE)*, vol. 1, 2010, pp. 95–104.

[33] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 448–456.

[34] B. Hu and M. Ester, "Social topic modeling for point-of-interest recommendation in location-based social networks," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 845–850.
[35] G. Jang, T. Lee, S. Hwang, C. Park, J. Ahn, S. Seo, Y. Hwang, and Y. Yoon, "PISTON: Predicting drug indications and side effects using topic modeling and natural language processing," *J. Biomed. Informat.*, vol. 87, pp. 96–107, Nov. 2018.
[36] X. Li, H. Xie, R. Y. K. Lau, T.-L. Wong, and F.-L. Wang, "Stock prediction via sentimental transfer learning," *IEEE Access*, vol. 6, pp. 73110–73118, 2018.

XIONG ZHANG received the bachelor's degree from the University of Science and Technology of China, the master's degree from Louisiana State University, USA, and the Ph.D. degree in information systems from the City University of Hong Kong. He is currently an Assistant Professor with the School of Economics and Management, Beijing Jiaotong University. He has published in journals, including the *Journal of the Association for Information Systems*, *Information Systems Frontiers*, *Information Technology and People*, *Internet Research*, and *Electronic Commerce Research and Applications* and has presented at conferences, including the International Conference on Information Systems, Workshop on Information Technologies and Systems, Pacific Asia Conference on Information Systems, Cross Strait Conference of Information Management Development and Strategy, and others.

HAORAN XIE (Senior Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong. He is currently an Associate Professor with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong. His research interests include artificial intelligence, big data, and educational technology. He has published 200 research publications, including 86 journal articles, including 66 SCI/SSCI indexed and 14 SCOPUS indexed publications. He is a Senior Member of ACM. He has obtained ten research awards, including the Golden Medal and the Special Award from the International Invention Innovation Competition in Canada, the Silver Award from Geneva's Invention Expo, and others. He has ranked among the world's top 25 researchers in Artificial Intelligence in Education on Google Scholar. He is the Editor-in-Chief of *Computers and Education: Artificial Intelligence* and an Associate Editor of the *International Journal of Mobile Learning and Organisation*. He has served as an Academic Editor for *PLOS One* and *Education Research International*, an Editorial Member of IJDET and JCE, in 11 guest editorships of special issues of journals, an Editor for 6 books, an Organization Committee Chair/Member of 57 conferences, and a Reviewer for 40 international journals. He has successfully obtained more than 45 research grants, the total amount of these grants is more than HK$27 million.

HAO YANG is currently a Senior Student with Beijing Jiaotong University. His major research is in management information systems. His research interests include natural language processing, image identification using deep learning, network information security protection technology, and industrial big data analysis.

HONGKAI SHAO is currently pursuing the bachelor's degree with Beijing Jiaotong University. His major research is in information management and information systems. He has a wide range of interests in programming and machine learning.

MINGHAO ZHU received the B.E. degree in transportation and the Ph.D. degree in management science from Beijing Jiaotong University, Beijing, China, in 2007 and 2012, respectively. From 2013 to 2015, he was a Postdoctoral Research Fellow with the Institute of Computing Technology, Chinese Academy of Sciences. He is currently an Associate Professor of information management with Beijing Jiaotong University. His research interests include manufacturing policy, urban traffic, and rail traffic using system dynamics and deep learning. He has received honors and awards, including the Science and Technology Award of the China Railway Society and the Scientific Research Award of Beijing Jiaotong University.