Classification of Microsoft Office Vulnerabilities: A Step Ahead for Secure Software Development

Supriya Raheja and Geetika Munjal

Abstract Classification of software vulnerability no doubt facilitates the understanding of security-related information and accelerates vulnerability analysis. In the absence of proper classification, it hinders its understanding and also renders the strategy of developing mitigation mechanism for clustered vulnerabilities. Now, software developers and researchers have agreed on the fact that incorporating security in software engineering phases including requirement and design may yield maximum benefits. In this paper, we have attempted to classify software vulnerabilities of Microsoft Office, so that this can help in building secure software. Vulnerabilities are firstly classified on well-established security properties like authentication and authorization. Vulnerability data is collected from various authentic sources, including Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE). From these databases, only those vulnerabilities are included whose mitigation is possible at the design phase. Then, this vulnerability data is preprocessed to handle missing values and noise removal; further, the data is classified using various supervised machine learning techniques. Classification models are compared for three security metrics: integrity, confidentiality and availability. All the classifiers achieved highest accuracy for integrity.

Keywords Vulnerabilities · Software flaws · Classification · Microsoft Office vulnerabilities · Analysis

1 Introduction

Software development is a complex process. Due to the high customer demand, design and program complexity of a software increase which creates difficulties for a software developer. Program complexity subsequently increases the security flaws.

S. Raheja (✉) · G. Munjal
The Amity University, Noida, India
e-mail: supriya.raheja@gmail.com

G. Munjal
e-mail: munjal.geetika@gmail.com

A. K. Bhoi et al. (eds.), *Bio-inspired Neurocomputing*, Studies in Computational Intelligence 903, https://doi.org/10.1007/978-981-15-5495-7_21

The use of common and familiar code in the software development also increases the probability of exploitation by the attackers as they have the expertise and tools to exploit the bugs present in the code. Sometimes, the developer leaves a software bug which may allow an attacker to exploit an application. Even, software vulnerabilities have adverse concerns that can aggravate the common software system misadventure. The exploitation of these software vulnerabilities may affect the survival of people which can have possibly terrible effects. In addition, the execution of vulnerable software can violate the security policy of an organization either implicitly or explicitly. So, in the current era, security flaws should be the major concern for the developers. But, most of the software developers compromise with the software vulnerabilities to achieve the organization's goal.

1.1 Vulnerability

Vulnerability is a flaw in the security of system. This fragility exists because of procedures followed in designing and implementation of security system or internal controls which permit an attacker to intentionally exploit the system. It compromises the security of the system or violates the system's security policy. A famous example of vulnerability is the Java vulnerability which occurred due to a coding error. In this, the software developer decided the package membership with the first component of the package name, and developer has restricted the package name with the first period in the full name, rather than the last period in the full name [1, 2].

Vulnerability consists of three basic elements:

i. Defect in the design of the system
ii. Attacker's access to the flaw
iii. Attacker's capability to exploit the flaw.

If an attacker wants to exploit vulnerability, then they must have at least one technique that can connect to weakness of the system [3].

1.1.1 Classification of Vulnerabilities

Vulnerabilities are classified into different classes based on the asset class; they are belongings such as personnel, physical site, organizational, hardware, network and software.

- Personnel class: includes vulnerabilities due to improper recruitment process and inadequate security knowledge.
- Physical site class: includes vulnerabilities based on the area subject to natural disasters and unpredictable power outages.
- Organizational class: includes vulnerabilities due to lack of improper audits and the absence of continuous planning about security.

- Hardware class: includes vulnerabilities which can be prone to humidity, easily prone to dust, soiling and malicious storage.
- Network class: contains vulnerabilities due to unprotected communication lines and insecure network architecture.
- Software class: includes vulnerabilities due to insufficient testing, lack of audit trail and many causes due to which these vulnerabilities occur like complexity, familiarity, connectivity, password management flaws, software bugs, unchecked user input, OS design flaws and many more [4].

In this work, the focus is on software-based vulnerabilities. The main sources of these vulnerabilities are software bugs, insufficient testing, unchecked user inputs, lack of audit trail, etc. [2]. Finding of vulnerabilities becomes an everlasting issue in all kind of software products. In 2018 only, a total of 16,555 vulnerabilities are in the record of CVE [5], in which more than 10,000 were found due to software bugs only. Even, in the last few years, there is an increase in data of undisclosed or zero-day [5] vulnerabilities. Software developers cannot ignore the publicly published vulnerabilities as they are still creating the highest threat for software companies. As stated by the SANS report, most of the security breaches were exploited because of known vulnerabilities [3].

The security of user data majorly depends on the security of product which they are using. Attackers may compromise the user data by exploiting the vulnerabilities. There are various software products which are mainly preferred by the users as well as by developers. The Microsoft software products are one of them. So, the exploitation of these software products may affect the mass users. Like, if a user works with a susceptible version of Microsoft Office, malicious program present in the file may exploit the vulnerability which further infects the user's document. Microsoft security development [6] team highlighted the following security-related issues: (i) choice of granularity, which is actually involved in data collection and making prediction based on it. It also focuses on multi-level prediction as compared to binary-level because binary-level prediction model adds little knowledge thus by adding more security defects. (ii) Statistical learner choice, it helps in choosing learning method which is even more important than 'data.' (iii) Classification performance, it suggests that performance in terms of precision and recall with threshold of '0.7' is fair for prediction models. This chapter addresses the software vulnerabilities of the most usable product of Microsoft, i.e., Microsoft Office.

The most preferable base for vulnerability database is the Common Vulnerabilities and Exposures (CVE) by researchers. This database is maintained by the National Institute of Standards and Technology (NIST) [7]. The usage of such databases like CVE makes the software developers, security experts, security vendors and research-based organizations to use the vulnerability data proficiently. It helps the software developers to recognize the infected software products in terms of particular vulnerability from the CVE database. But it is a tremendous task for the developer to identify and search the vulnerability from the list.

A software developer must give emphasis on the secure code which can reduce the chances of exploitation. Analysis of these vulnerabilities can provide a base to

the software developers for enhancing the security of software product. In this work, we are emphasizing the Microsoft Office. Through the analysis and classification, developers can prioritize the vulnerabilities. The severity of any vulnerability can be acknowledged with the help of a vulnerability score defined by common vulnerability severity score (CVSS). These vulnerability scores are majorly influenced by three impact metrics, including integrity, confidentiality and availability. Being directly proportional to given metrics, reducing these metrics, one can reduce the vulnerability score. Therefore, knowledge of which vulnerability increases these factors may provide the scope of improvement to developers and also to prioritize their responses.

The prime objective of this work is to categorize different vulnerabilities to define their accuracy in terms of these three metrics. Authors have been considering the last twelve years database of Microsoft Office vulnerabilities (January 2006–June 2018). The other objective of this work is to provide the comparative evaluation of various classification techniques in analyzing the vulnerable data so that appropriate classification techniques can be used for the required task.

The rest of the chapter is organized as follows: In Sect. 2, we discuss the background and related work of software vulnerabilities. Section 3 discusses the Microsoft Office vulnerabilities and database used in detail. Section 4 discusses different types of classification techniques. The results of different classifiers are demonstrated in Sect. 5, and finally, conclusions and future scope are discussed in Sect. 6.

2 Background and Related Work

While developing a software program, change may introduce security vulnerabilities. Software development teams have made use of various strategies to verify if code contains security vulnerabilities; for example, the Microsoft has also introduced security development life cycle to enhance its product's security; however, for large software products, a single approach to fill security feature is not sufficient. Thus, the vulnerability detection and removal techniques using more effective techniques must be ordered to the most skeptical areas of the product. Generally, vulnerabilities have been classified into broad categories such as buffer overflows, format string vulnerabilities, code execution and memory corruption. [3]. There are two major issues associated with these broad categories: First, it is difficult for a developer to assign a vulnerability to a single category; second, the dissimilarities are too common to be useful for any analysis. Correct classification of vulnerabilities can lead to improved relationship between the incidents and their exploitations. This can help developers to determine the effectiveness of countermeasures; however, such analysis has not been often applied.

In one of the studies, authors [7] have proposed an ontology-based approach to retrieve vulnerability-related data and establish a correlation between them; the data is taken from the National Vulnerability Database [8]. Authors also provided

machine recognizable CVE vulnerability knowledge and reusable security vulnerabilities interoperability. Authors proposed a classification framework which transforms the vulnerability dictionary into CVE. Authors have used clustering algorithm for analysis purpose [4]. Li et al. have explored other possibilities of classifying vulnerabilities based on the CVE using self-organizing maps [9].

A system is introduced to observe vulnerabilities of software used in organization [10]. This paper also explains the CPE dictionary which is used to identify the CPE identifier. These identifiers are consigned to software products based on the collected information. The CVE database is also used to recognize vulnerabilities in software based on the allotted CPE identifiers. The projected approach was not human interactive. Searching for CVEs by a software developer may lead to a poor efficacy.

Several researchers have worked on the identification and classification of software faults. Carlstead et al. conducted a research on operating system. Authors have worked on the protection errors in operating system [11]. Later, Landwher et al. [9] in 1993 have suggested how to enhance the security in the existing operating systems. Authors have published a list of security flaws in different operating systems and classified each flaw based on its genesis or on other factors. Marick [12] have presented a survey on different studies on software faults. Aslam et al. [13] in 1995 and then in 1996 have developed a classification scheme to understand the software faults that can challenge the security mechanisms used during development of software products. The proposed classification scheme has divided the software faults mainly into two broad categories: coding faults and emerging faults. Coding faults are due to syntactical errors, design errors or missing requirements, whereas emergent faults mainly occur because of improper installation of software. In this case, faults may present even when the software functions properly according to specifications. Bishop [14] and Bailey have shown that this classification technique does not provide the accurate results when a fault exists in more than one classification category. Leveson [15] has suggested that by understanding the nature of vulnerabilities, the system design can be improved which can further reduce the risk of running critical or sensitive systems.

Authors have discussed the characteristic-based classification for bugs [14], but this bug classification method was not effective to classify software vulnerabilities. In [11], authors have classified the vulnerabilities when they are introduced on the basis on software development life cycle. But the scheme was not effective; when the vulnerability belongs to more than one phase, then it was difficult to determine the specific class to which that belongs. In [16], authors have classified the vulnerabilities according to genesis to categorize them, if they are done for purpose or inadvertent. They further classified the intentional vulnerabilities into malicious or non-malicious. In [17], authors have classified the vulnerabilities on the basis of their immediate impact on software. Akyol presented the same work but specific for the network protocols.

In [16], authors have classified vulnerabilities based on their disclosure procedure. In authors have classified vulnerabilities in terms of impact, type of attack

and source of vulnerability. Authors have classified vulnerabilities of mobile operating system based on elevation of privilege notions. Authors have applied machine learning models to classify vulnerabilities. In [18], authors have applied the Naïve Bayes algorithm to classify vulnerabilities in terms of text info. Further, in [19], LDA and SVM algorithms have been applied to classify vulnerabilities from National Vulnerability Database (NVD). Authors have proposed an approach for classification of software vulnerabilities based on different vulnerability characteristics including resources consumption, strict timing requirement, etc. They classify and give statistical analysis of the vulnerabilities. Massaccis specifically reported by Google Project Zero.

Various studies [20–22] compared the effectiveness of distinct modeling methods for fault prediction, and they have demonstrated that different modeling methods result in different prediction accuracies where accuracy results are dependent on the dataset. To find the impact of different modeling methods on Microsoft Office, we have applied several of these modeling techniques in our experiments. It will also help to evaluate if an algorithm choice makes a difference in terms of class prediction or not.

From the last some years, the software engineering community has increased the utilization of machine learning technology in software analysis [23, 24]. This will benefit the researchers in taking advantage from structure present in the data beyond the linear combinations modeled by regression. This also includes discovering new vulnerabilities by auditing; however, still lots of vulnerabilities are unidentified. For example, in [25], author has tried to related commits to CVEs. In [26], the author has worked on relatedness of commit message and issue reports where machine learning approach is used for natural language processing. The approach uses machine learning techniques for natural languages. Sabetta and Bezzi [27] identified usage of source code change along with commit message. In all the above studies [25, 26], the results have demonstrated high precision and low recall related to the commit message, thus highlighting the commit parameter in software vulnerability analysis.

Alhazmi and Malaiya [5] have built a model for future vulnerabilities where they targeted operating systems. The study is based on the logistic model that is tested on the existing data; the results are validated on the parameters including average error and average bias.

In one of the studies, several existing databases of vulnerability are compared on the basis of vulnerability features in each of them. In various databases, many important features are missing. For example, ‘discovery date’ is hard to find [28] in database.

There are quite a number of researches which are focusing on the classification of vulnerabilities to help out the software developers and even no one has explored particularly about the classification of Microsoft products. There are many more vulnerability classification approaches existing in the literature, but they are very general with respect to software vulnerabilities. Supriya et al. [27] have mentioned the classification of CVE-based vulnerabilities of Linux kernel. As per our literature survey, no one has ever classified the vulnerabilities of Microsoft Office which is

majorly used by the users for their day-to-day task. Therefore, in this research work, authors are focusing on the classification of Microsoft Office (MSO) vulnerabilities using machine learning technique. For this work, authors have precisely considered the CVE database of Microsoft Office vulnerabilities. Accuracy of classification is defined in terms of three metrics: integrity, confidentiality and availability. In the next section, a detailed description of vulnerabilities and different classification techniques are discussed.

3 MS Office Vulnerabilities Dataset and Their Attributes

Database is the backbone of any system. Similarly, vulnerability databases are an important part of any software security knowledgebase. There are numerous vulnerability databases available which exchange software vulnerability information. However, the impact of categorization depends on the used vulnerability database. Extracting valuable and important information from these available databases is a tremendous task for a developer [10, 29–32]. Classifying these vulnerability databases assists developers to detect a mitigation method and also to prioritize and remediate vulnerabilities in real time. This research work classifies the Microsoft Office vulnerability based on three major security metrics using machine learning techniques; these metrics directly impact the vulnerability score. Severity of any software vulnerability depends on the vulnerability score, and these metrics are directly proportional to the vulnerability score. The vulnerability score can help the developers to identify the risk level of the vulnerability in the early stage of software development life cycle (SDLC).

3.1 Vulnerabilities in MS Office

MS Office documents are exposed to different types of vulnerabilities which can be exploited by attackers and intruders. From the database, authors have identified the main vulnerabilities of MS Office which are exploited in the last twelve years. Different exploited vulnerabilities are illustrated in Fig. 1. This section describes the each vulnerability as follows:

1. *Code Execution*: It is used to define the ability of an attacker to execute the instructions on a target machine. The attacker executes the commands of their own choice. It is a kind of software bug which provides a way to execute an arbitrary code. In most of the cases of exploitations, attackers inject and execute the shellcode to run arbitrary commands manually.
2. *Buffer overflow*: It is also known as buffer overrun. It is an anomaly program that allows a program to overrun the buffer's boundary in parallel to writing data in a buffer, and it also overwrites the adjacent memory locations. As a special case,

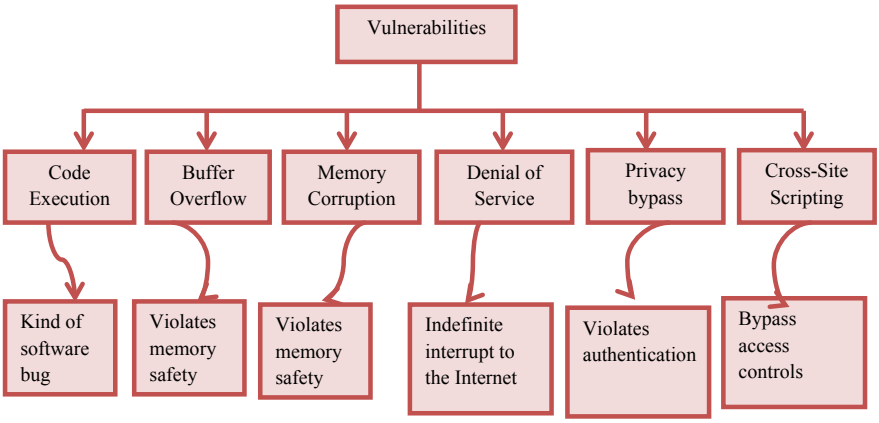

Fig. 1 Important vulnerabilities in MS Office since the last 12 years

buffer overflow is violation of memory safety. This vulnerability is exploited by inputs that are designed to execute code or to change normal execution of the program. This may result in incorrect results, memory access errors, a crash or a breach of system security. Therefore, this software vulnerability is majorly found in software products and can easily be maliciously exploited by an attacker [33].

3. *Memory Corruption*: The contents of memory locations are unintentionally modified due to the programming errors which lead the corruption of the memory in a computer program. Such type of errors violates memory safety. In future, if the infected memory is used in the software, it may cause either to software crash or can reflect some kind of abnormal action. Around 10% of the software application on the Windows system fails due to memory corruption [34].
4. *Privacy Bypass*: Most of the applications require authentication to gain access on confidential data or private information. Based on studies, it can be concluded that there is not a single authentication procedure exists which provides adequate security. There are different ways by which authentication process can be easily bypassed by ignoring or neglecting the security threats. It can be bypassed by ignoring the login page or can directly call a page which allows to be accessed only after the valid authentication [3].
5. *Denial of Service*: It is an attempt made by attacker that makes a machine or network resource unavailable to the intended users such as to interrupt or suspend services of a host who is connected to the Internet. It is accomplished by flooding the targeted machine or resource to overload systems and to prevent all legitimate requests from being fulfilled [8].
6. *XXS*: Cross-site scripting vulnerability is used by attackers to bypass access controls. This scripting is carried out on Web sites. Their effect may range to significant security risk, depending on the sensitivity of the data which is handled by the vulnerable site and the nature of any security mitigation implemented by site's owner [35].

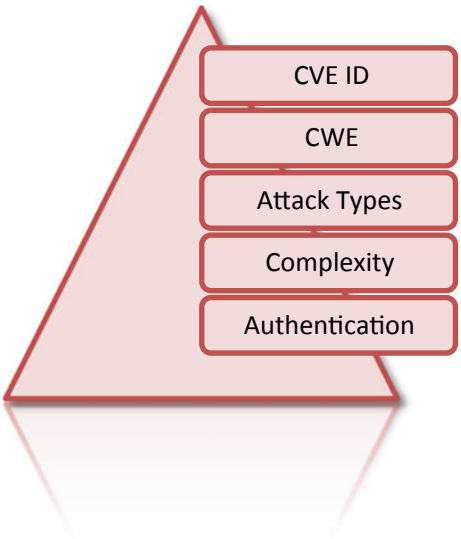

Fig. 2 Attributes used in dataset

All these types of vulnerabilities are exploited by the attackers and may result into risk for software. In the dataset, authors have represented these MS Office vulnerabilities under one attribute, i.e., attack type. This paper considers all important attributes which affect the impact of integrity, confidentiality and availability metrics as given in Fig. 2.

3.2 Dataset and Attributes Used

CVE is a process which is used to allocate unique numbers to widely well-known vulnerabilities present in software applications and to offer the vulnerability statistics in detail. It also provides information about the affected products. Nowadays, CVE is the standard among software companies, antivirus sellers, researchers and security experts to share the information about the known vulnerabilities. Assignment of unique identifiers, i.e., 'CVE ID,' is managed by the MITRE [19, 36, 37]. Figure 3 shows a CVE record acquired from the CVE database (CVE-2018-0950) for Microsoft Office 2018. NIST provides the CVE documents in NVD xml format which contains additional information like severity metrics, vulnerability type and affected software list. Figure 4 shows the severity and metrics of CVE feed (CVE-2018-0950). CVSS Base score is provided in NVD feeds to the vulnerability [33–35]. From Fig. 4, we can see that CVSS Base score is provided in NVD feeds to the vulnerability.

In this work, authors have taken the records of Microsoft Office vulnerabilities from the NVD feeds from the period 2006–2018 to analyze and classify the Microsoft Office vulnerabilities. In this work, authors have explored the vulnerability details of the vulnerabilities found in MS Office document. A sample snapshot of dataset is shown in Fig. 5. These attributes are briefly described in this section:

CVE-2018-0950 Detail

Current Description

An information disclosure vulnerability exists when Office renders Rich Text Format (RTF) email messages containing OLE objects when a message is opened or previewed, aka "Microsoft Office Information Disclosure Vulnerability." This affects Microsoft Word, Microsoft Office. This CVE ID is unique from CVE-2018-1007.

Source: MITRE

Fig. 3 CVE feed for the Microsoft Office 2018 [35]

Impact

CVSS v3.0 Severity and Metrics:

Base Score: 6.5 MEDIUM

Vector: AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N (V3.0 legend)

Impact Score: 3.6

Exploitability Score: 2.8

Attack Vector (AV): Network

Attack Complexity (AC): Low

Privileges Required (PR): None

User Interaction (UI): Required

Scope (S): Unchanged

Confidentiality (C): High

Integrity (I): None

Availability (A): None

Fig. 4 Severity and metrics for CVE feed (CVE-2018-0950) [35]

	CVE ID	CWE ID	Attack Type(s)	Published Date	Update Date	Score	Gained Access	Access	Complexity	Authentication	Conf.	Integ.
2	CVE-2016-0145	119	Exec Code Overflow Mem. Corr.	4/12/2016	4/14/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
3	CVE-2016-0134	119	Exec Code Overflow Mem. Corr.	3/9/2016	3/11/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
4	CVE-2016-0127	119	Exec Code Overflow Mem. Corr.	4/12/2016	4/14/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
5	CVE-2016-0057	264	Priv Bypass	3/9/2016	3/11/2016	7.2	None	Local	Low	Not required	Complete	Complete
6	CVE-2016-0056	119	Exec Code Overflow Mem. Corr.	2/10/2016	2/16/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
7	CVE-2016-0055	119	Exec Code Overflow Mem. Corr.	2/10/2016	2/16/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
8	CVE-2016-0053	119	Exec Code Overflow Mem. Corr.	2/10/2016	2/12/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
9	CVE-2016-0052	119	Exec Code Overflow Mem. Corr.	2/10/2016	2/17/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
10	CVE-2016-0012	200	Bypass +Info	1/13/2016	1/14/2016	4.3	None	Remote	Medium	Not required	Partial	None
11	CVE-2016-0010	119	Exec Code Overflow Mem. Corr.	1/13/2016	1/14/2016	9.3	None	Remote	Medium	Not required	Complete	Complete
12	CVE-2015-6172	20	Exec Code	12/9/2015	12/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
13	CVE-2015-6124	119	Exec Code Overflow Mem. Corr.	12/9/2015	12/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
14	CVE-2015-6118	119	Exec Code Overflow Mem. Corr.	12/9/2015	12/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
15	CVE-2015-6108	119	Exec Code Overflow Mem. Corr.	12/9/2015	12/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
16	CVE-2015-6107	119	Exec Code Overflow Mem. Corr.	12/9/2015	12/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
17	CVE-2015-6106	119	Exec Code Overflow Mem. Corr.	12/9/2015	12/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
18	CVE-2015-6093	119	Exec Code Overflow Mem. Corr.	11/11/2015	11/12/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
19	CVE-2015-6092	119	Exec Code Overflow Mem. Corr.	11/11/2015	11/12/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
20	CVE-2015-6091	119	Exec Code Overflow Mem. Corr.	11/11/2015	11/12/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
21	CVE-2015-2545	20	Exec Code	9/8/2015	9/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
22	CVE-2015-2510	119	Exec Code Overflow	9/8/2015	9/9/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
23	CVE-2015-2477	119	Exec Code Overflow Mem. Corr.	8/14/2015	8/18/2015	9.3	None	Remote	Medium	Not required	Complete	Complete
24	CVE-2015-2470	189	Exec Code	8/14/2015	8/18/2015	9.3	None	Remote	Medium	Not required	Complete	Complete

Fig. 5 A snapshot of few entries of MSO vulnerability dataset

Table 1 Different CVSS scores of vulnerabilities with their state condition

Score range	State
1–3	Low
4–6	Medium
7–9	High
9–10	Critical

i. CVE ID: Common Vulnerability and Exposure. This is a unique identifier for public-known cybersecurity vulnerabilities.
ii. CWE ID: Common Weakness Enumeration. This includes the set of software weaknesses that can lead to software vulnerability and errors. This enumeration aims to stop the vulnerabilities at the source by educating software acquirers, designers, architects, etc.
iii. CVSS Score: Common Vulnerability Scoring System (CVSS) provides standardized vulnerability scores, i.e., when an algorithm is used for scoring the vulnerability as mentioned in Table 1. CVSS leverages a single vulnerability allowable time to validate and remediate a given vulnerability. It is also used to prioritize risk [19].
iv. Attack complexity: This shows that how much the conditions are beyond the attacker's control to exploit the vulnerability. This complexity has two possible values:
    (a) Low: This says that no such conditions exist. Successfully, vulnerability can be exploited.
    (b) High: Successful attack cannot be done. It might be target specific, and reconnaissance is required.
v. Confidentiality impact: This gives the measure of confidentiality information managed by software component due to successfully exploited vulnerability. The following are the values showing the impact of confidentiality:
    (a) Complete: total loss of confidentiality.
    (b) Partial: some loss of confidentiality but do not have full control.
    (c) None: There is no loss of confidentiality.
vi. Integrity Impact: This gives the measure of trustworthiness or veracity of information.
    (a) Complete: total loss of protection. Attacker can modify all files protected by component.
    (b) Partial: Modification is possible, but attacker does not have control over the consequences of modification. Data modification does not have serious impact on component.
    (c) None: There is no loss of integrity.
vii. Availability impact: Accessibility of information resources measures the impact of availability of the component.

(a) Complete: Total loss of an availability resulting in attacks is now completely able to deny access to resources.
(b) Partial: reduced interruption in resource availability
(c) None: There is no loss of availability.

4 Classification of Vulnerabilities

Categorizing vulnerabilities will help to explain different possible common vulnerability vectors and the features responsible for them. It can also help in detection and remediation of vulnerabilities. This section briefly discusses various classifiers used in the literature for various domains.

Classification is the task of obtaining the correct class label for an unknown sample [8, 38]. The set of labels is defined in advance, and each sample is considered in separation from the other samples. To match the test sample from the pre-stored labels from the samples in the dataset, many similarity measures are used such as the distance measure, information gain and entropy.

4.1 Bayes Classifiers

Some classifiers follow Bayes theory which is based on statistical classification. Using class belongingness probabilities, they can help in predicting class of data sample, which will further help in the classification process. This classifier follows Bayes' theorem. Naive Bayesian classifiers believe in contribution of class information about remaining in sustainment of participating attribute [38]. This property is known as class conditional independence. The computation involved in the process simplifies the task, in this sense, is considered 'naive.' Various Bayes classifiers are Bayes net and Naive Bayes [18, 21]. The probabilistic strategies which are counted under linear classifier are logistic regression and Naïve Bayes classifier. Naïve Bayes classifier acts as linear classifier, if the values in likelihood factor $p\,(X_i|c)$ are exponents, where p is the probability of a single attribute 'i' from a set X attributes. $X_m = [x_1, x_2, x_3, \ldots, x_m]$ with 'm' representing count of attributes, c is one of the class or target labels from the set of 'k' class labels $\{c_1, c_2, c_3, \ldots, c_k\}$.

Naïve Bayes classifier assumes class conditional independence. The value of a feature about a class is independent of the values of the other features. It is based upon the principle of Bayes theorem. Bayes classifier uses the probabilistic approach to assign the class/target label to the new sample. Posterior conditional probability $p(c_i|X_m)$ in Bayesian classifier is computed using Bayes rule.

$$p(c_i|X_m) = \frac{p(X_m|c_i)\,p(c_i)}{p(X_m)} \tag{1}$$

Posterior probabilities are calculated in Eq. (1) for $i = 1, 2, 3, \ldots, m$ using the training data. Naïve Bayes classifier is easy and fast to predict class on test data even when the data is multi-class. Due to its simplicity, this method may outperform on complex models. However, it is quite sensitive to the presence of redundant and irrelevant predicted attributes.

4.2 K-Nearest Neighbor

Another type of classifiers widely used and tested in literature is the nearest neighbor classifier which employs normalized Euclidean distance to search the nearest sample (instance) to the sample in a test data from training samples. It predicts the same class as that given in a training sample. If there are many instances in the training data having the same distance, then the instance found first is used to predict the class. The classifiers which fall into this category are KNN and IB1. They both are instance-based learners.

K-nearest neighbor (KNN) is also said to be ***lazy learner*** which is sensitive to the local structure of the data and does not build model explicitly. Here, the value of k needs to be specified in advance. This approach becomes infeasible in case of high-dimensional data, and unknown samples become expensive [12].

Another nearest neighbor classifier is instance-based (IB1) classifier; they use specific instances during the prediction time of the classifier. To match the samples, they use similarity functions. Instance-based learning algorithms are based on the assumption that the similar samples (instances) have similar classifications. It leads to their local bias for classifying novel samples according to their most similar neighbor's classification. IB1 classifier can be used for two-class or multi-class datasets. It fails to classy suitably and gives less accurate results if the features are logically inadequate for describing the target class. IB1 classifier is useful for classification especially in feature selection problems, as it gives a better accuracy of the reduced data where features are adequate in describing the target class than the accuracy of complete data.

The performance of the nearest neighbor classifier is comparable of decision tree classifiers. It can interpret, generate and classify new instance from the model rapidly. In lazy classifiers, generalization of training data is delayed until a query is made to the system. The advantage of lazy classifier is that it performs approximation for target function locally, as the system needs to solve multiple problems simultaneously. After that, the system deals with the changes made in the problem domain. This classifier is successful for large datasets. IBK, Kstar and LWL are some algorithms of lazy classifier.

Tree-based classifiers are decision tree and random forest; both have proven to be robust and successful tools for solving many problems in the field of machine learning. Random forest can be used for two classes as well multi-class problems where the number of classes is more than two [22]. In this situation, there is little need to fine-tune the algorithmic parameters involved in random forest. However, random

forest algorithm does not work well when there is extreme data imbalance in data. ***Decision tree*** generates a tree with a set of rules to represent different classes from the given dataset. Classifiers utilizing decision tree to classify an unknown sample generate the tree in two phases. One is the tree construction phase and other is tree pruning. In the construction phase, the tree keeps all the training examples at the root and then partitions them recursively based upon some criterion. Different approaches to construct a decision tree are Iterative Dichotomiser 3 (ID3), C4.5 a successor of ID3 presented by Quinlan and classification and regression trees (CART). All of them follow a top to bottom greedy approach to construct a tree. Training set attributes are recursively partitioned into smaller subsets as the tree keeps on building. The goal of the decision tree is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the available attributes. This method is able to handle both numerical and categorical data and easily able to handle multi-output problems.

Meta-classifier uses automatic learning algorithms on meta-data. This meta-data is further used to know how the automatic learning can introduce flexibility in solving different learning problems which can further improve the performance of the existing learning algorithms [39]. Meta-classifiers has are good for on datasets with different levels of class imbalance. A meta-classifier 'bagging' is developed using several training datasets; these datasets are built from the original training dataset. It finds various models and averages them to produce a final ensemble design. Another meta-classifier is under-sampled stratified bagging where the total training set size is double the number of the minority class. Some meta-classifiers are cost sensitive.

Some meta-classifier algorithms in WEKA [40] are on the name Adaboost, bagging, stacking, vote, logic boost, etc. Another important classifier discussed in the literature is rule based. It follows technique of assigning each element of population set to one of the defined classes. If every element in the population is assigned to a class it really belongs, it will be considered as a perfect test. However, in case of errors, statistical techniques will be applied to validate the classification. Various rule-based algorithms are decision table, decision trees and random forest [12]. In this work, we have applied various types of classifier on MSO vulnerability dataset to analyze an impact of various metrics.

4.3 Evaluating Classifiers

The classification models are evaluated based on its accuracy, speed, robustness, scalability and interpretability [41].

The output of a classifier is taken in the form of a confusion matrix shown in Table 2 depicting the predicted class with true and false numbers.

False positives are the samples that are erroneously predicted as positive, but actually belong to a negative class. Similarly, false negatives are the samples that are wrongly predicted as negative, but truly belong to a positive class. True positives are the samples that are accurately predicted as positive and actually belong to the

Table 2 Confusion matrix

	Predicted class	
Actual class	Positive	Negative
Positive	True positive	False negative
Negative	False positive	True negative

positive class. Likewise, true negatives are the samples which are accurately predicted as negative and truly belong to the negative class. All these measures of the confusion matrix are further helpful in evaluating the accuracy, specificity and sensitivity of a classifier. Higher values of true positive and true negative show the effectiveness of the algorithm. All these parameters are validated based on a k cross-validation where the data is divided into k folds, where $k - 1$ folds are selected as the training data and the reserved fold is used to calculate accuracy, specificity and sensitivity.

Accuracy is the most common used performance measure. It evaluates the effectiveness by taking the ratio of correct prediction divided by total number of predictions made as in an Eq. (2).

$$\text{Acc} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} \tag{2}$$

Other performance measure is **error rate** which is the complement of accuracy rate. It evaluates the effectiveness by taking the ratio of incorrect prediction divided by total number of predictions made. **Recall** is also called as the sensitivity or true positive rate. It is defined as the ratio of the samples belonging to positive class which was predicted correctly as positive. **Specificity is** defined as the ratio of the samples belonging to negative class which was predicted correctly as negative. **Precision** is defined as the proportion of the samples belonging to positive class which was predicted correctly as positive.

Cross-validation technique is applied to validate the results; this technique splits the original data into one or more subsets [42, 43]. During training of the classifier, one of the subsets is used as test set to test the accuracy of the classifier and other subsets are used for parameter estimation. It keeps on rotating the test set based upon an applied cross-validation technique. Several types of cross-validation techniques can be used such as k-fold cross-validation and leave-one-out cross-validation.

All the experiments are done in Waikato Environment for Knowledge Analysis (WEKA) [21] which is a landmark in data mining including classification. It provides a comprehensive collection of machine learning algorithms. It allows users to explore various machine learning methods over naive datasets.

5 Performance Evaluations of Classifiers on MSO Vulnerabilities

Various classifiers are applied on MSO vulnerabilities dataset. Some classifiers possess similar accuracy in all of three security services and some differentiate. However, the highest accuracy is held by integrity. Here, accuracy representation for various classifiers individually depicts the accuracy for integrity, confidentiality and availability.

5.1 Using Bayes Classifiers

The Bayes classifier minimizes the probability of misclassification in statistical classification. This classifier possesses high scalability, which requires a number of parameters linear in the number of variables in a learning problem. Various versions of Bayes classifier are used from WEKA [21] including BayesNet, NaveBayes and Nave Bayes Updatable classifier. These classifiers depict the accuracy in terms of integrity, confidentiality and availability. Bayes net gave a highest accuracy for all three parameters as compared to nave Bayes and Nave Bayes updatable version as illustrated in Fig. 6.

5.2 Using Lazy Classifiers

The lazy classifiers have depicted accuracy as shown in Fig. 7. IBK algorithm of lazy classifier has shown 99% of accuracy in integrity, 98% for availability and

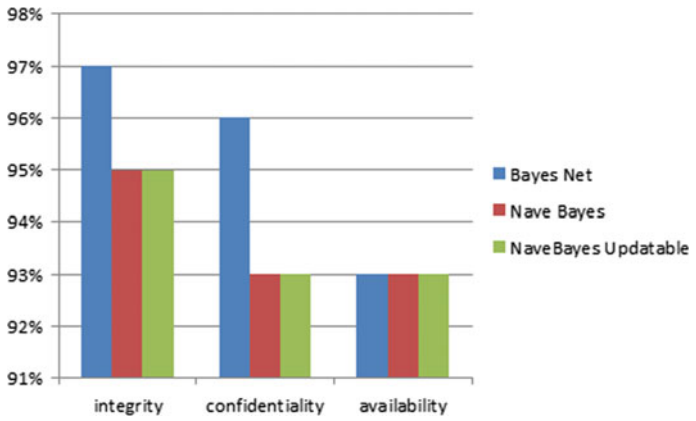

Fig. 6 Accuracy by Bayes classifier

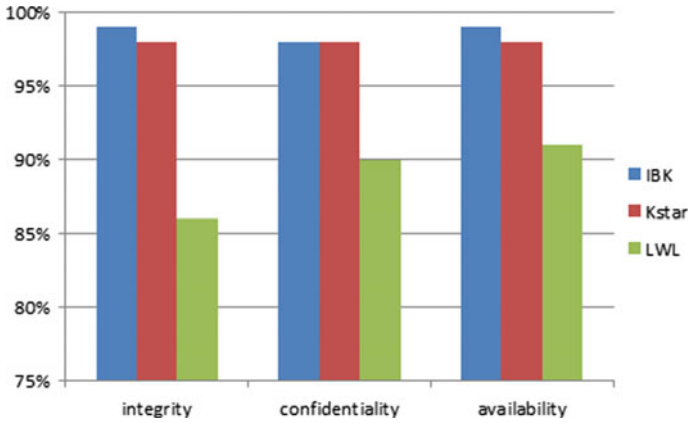

Fig. 7 Accuracy by lazy classifier

98% for confidentiality. KStar, performance is same for all three parameters, i.e., for integrity, confidentiality and availability. Locally weighted learning (LWL) is giving 91% accuracy for availability and 90% for confidentiality and 86% for integrity. These three algorithms are applied under cross-validation (CV) testing option to get validated results.

5.3 Using Meta-Classifiers

Various kinds of application are applied for meta-classifier as illustrated in Fig. 8. The algorithms applied are Adaboost M1, attribute selected, bagging, classification

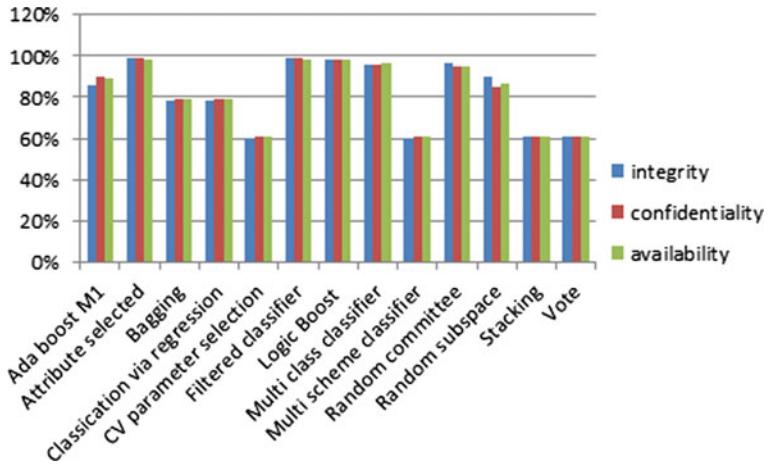

Fig. 8 Accuracy by meta-classifier

via regression, CV parameter selection, filtered classifier, logic boost, multi-class classifier, multi-scheme classifier, random committee, random subspace, stacking and vote. All these classifier algorithms are applied in this vulnerability. Results of these algorithms are represented in terms of accuracy for integrity, confidentiality and availability impact.

Filtered classifier and attribute selected possess 99% of accuracy in integrity and confidentiality. The logic boost depicts same amount of accuracy, i.e., 98% for integrity, confidentiality and availability. The bagging, classification via regression and CV parameter selection have 79% accuracy for confidentiality and availability and 79% for confidentiality and availability and 61% accuracy for confidentiality and availability. The random committee possesses 99% accuracy for integrity. Stacking and vote have 61% for all integrity, confidentiality and availability impact. This shows that there are various results for the classifier for meta, and majority is still with integrity.

5.4 Using Rules Classifiers

The classifier algorithms applied in rules are decision tables, JRip, OneR, PART and ZeroR as illustrated in Fig. 9. These depict the amount of accuracy for integrity, confidentiality and availability. In the rules classifier, decision table and PART have 99% of accuracy for integrity and JRip shows 99% of accuracy for confidentiality. ZeroR has 61% of accuracy for both confidentiality and availability. Still, the majority is with integrity.

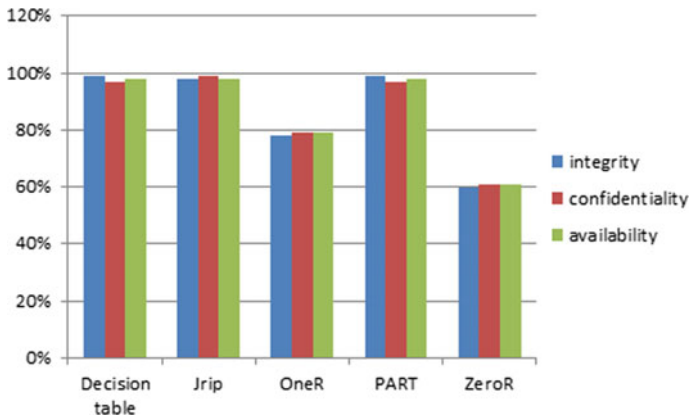

Fig. 9 Accuracy by rules classifier

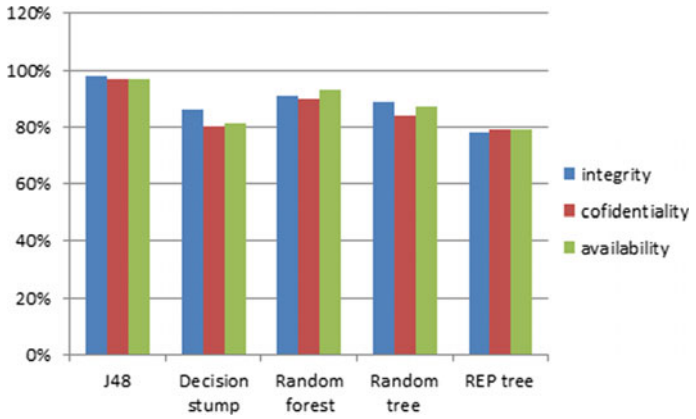

Fig. 10 Accuracy by trees classifiers

5.5 Using Tress-Based Classifiers

The tree classifiers used are J48, decision stump, random forest, random tree and REP tree to give the accuracy for integrity, confidentiality and availability. J48 have the highest accuracy which is about 98% for integrity. Decision stump gives 86% accuracy in integrity. Random forest gives 93% for availability. Random tree has 89% for integrity, and REP tree shows 79% for confidentiality and availability. These results are shown in Fig. 10. These classifiers give the accuracy results by all algorithms in them. Each classifier gives accuracy in terms of integrity, confidentiality and availability.

6 Conclusion and Future Work

In this work, authors studied the Common Vulnerability and Exposures list for different vulnerabilities from the National Vulnerability Database. They further explored the CVE database for Microsoft Office-related vulnerabilities. This work classified the Microsoft Office vulnerabilities using various supervised machine learning techniques. Classification results are evaluated on the basis of three metrics: integrity impact, availability impact and confidentiality impact which enable the software developers for the secure development of a software product. This classification results help the software developers to identify the particular impact of vulnerability on software which in advance can decide assessment and mitigation techniques for that particular vulnerability of Microsoft Office.

Our results depend on the dataset we use and parameter chosen. This suggests that performance can be enhanced with further experimentation and also combining metrics or type of attack which may be: integrity, confidentiality or availability and

learning methods. In some of the models, we discover only marginal improvements in all of the assessment parameters. These improvements will further reduce manual work in identifying vulnerability in software and help software developers in taking intelligent decisions while building software. Such studies help in understanding trends and patterns existing in software vulnerabilities and also managing the security of computer systems. In our study, we have also observed that in many databases, most of the entries are blank which may give misguiding results.

As a future work, code-specific parameters can be explored in improving software security. Other technique which may also help in enhancing software security is data resampling techniques and ensemble models of learning. Our current study is limited to classify licensed software vulnerabilities; in future, it can also be extended to classify vulnerabilities in open-source software. The work can also be extended by conducting more experiments using other machine learning models including unsupervised techniques to find similar patterns in vulnerabilities and attacks

References

1. Krsul, I.V: Software vulnerability analysis. Ph.D. dissertation, Purdue University (1998)
2. Krsul, I., Spafford, E.: A Classification of Software Vulnerabilities That Result From Incorrect Environmental Assumptions, Report Purdue University (2015)
3. Alqahtani, S.S., Eghan, E.E., Rilling, J.: Tracing known security vulnerabilities in software repositories–A semantic web enabled modeling approach. Sci. Comp. Prog. pp. 153–175 (2016)
4. Howard, M., LeBlanc, D., Viega, J.: 19 Deadly Sins of Software Security. McGrawHill/Osborne, Emeryville, CA (2005)
5. Alhazmi, O.H., Malaiya, Y.K.: Prediction capabilities of vulnerability discovery models. In: Proceedings of Annual Reliability and Maintainability Symposium (RAMS), pp. 1–10 (2006)
6. Howard, M., Lipner, S.: The Security Development Lifecycle. Microsoft Press (2006)
7. Guo, M., Wang, J.A.: An ontology-based approach to model common vulnerabilities and exposures in information security. In: Proceedings of ASEE 2009 Southeast Section Conference, Marietta, GA, USA, pp. 5–7 (2009)
8. Munjal, G., Kaur, S.: Comparative study of ANN for pattern classification. WSEAS Trans. Comput. **6**, 236–241 (2007)
9. Li, W., Yi, P., Wu, Y., Pan, L., Li, J.: A new intrusion detection system based on KNN classification algorithm in wireless sensor network. J. Electr. Comput. Eng. (2014). https://doi.org/10.1155/2014/240217
10. Syed, R., Zhong, H.: Cybersecurity vulnerability management: An ontology-based conceptual model. In: Twenty-Fourth Americas Conference on Information Systems, New Orleans, LA, USA, pp. 16–18 (2018)
11. Carlstead, J., Bibsey, II, R., Popek, G.: Pattern-Directed Protection Evaluation, Tech. Report., Information Sciences Institute, University of Southern California (1975)
12. Marick, B.: A survey of software fault surveys. Tech. Rep. UIUCDCS-R-90-1651, University of Illinois at Urbana-Champaign (December 1990)
13. Aslam, T., Krsul, I., Spafford, E.: Use of A Taxonomy of Security Faults, Tech. Report Number: 96–051, Department of Computer Science Engineering, Purdue University (1996)
14. Bishop, M., Bailey, D.: A Critical Analysis of Vulnerability Taxonomies. Tech. Rep. CSE-96–11, Department of Computer Science at the University of California at Davis (1996)
15. Leveson, N.: High-pressure steam engines and computer software. In: Computer 27, 10 (October), Keynote Address IEEE/ACM International Conference in Software Engineering Melbourne Australia (1992)

16. Christey, S., Wysopal, C.: Responsible Vulnerability Disclosure Process. INTERNET-DRAFT "draft-christey-wysopal-vuln-disclosure-00.txt". The Internet Society (2002)
17. D'Ambros, M., Lanza, M., Robbes, R.: Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empirical Software Eng. **17**, 531–577 (2012). https://doi.org/10.1007/s10664-011-9173-9
18. Zimmermann, T., Nagappan, N., Williams, L.: Searching for a needle in a haystack: Predicting security vulnerabilities for windows vista. In: Proceedings of Third International Conference on Software Testing, Verification and Validation (ICST), SVM, pp. 421–428 (2010)
19. Joshi, C., Singh, K.U., Tarey, K.: A review on taxonomies of attacks and vulnerability in computer and network system. Int. J. Adv. Res. Comput. Sci. Software Eng. **5**, 742–747 (2015)
20. Sabetta, A., Bezzi, M.: A practical approach to the automatic classification of security-relevant commits. In: 34th International Conference on Software Maintenance and Evolution. IEEE Computer Society, Sept. 2018, pp. 1–5 (2018)
21. Weber, S., Karger, P.A., Paradkar, A.: A software flaw taxonomy: Aiming tools at security. In: Proceedings of the 2005 Workshop on Software Engineering for Secure Systems—Building Trustworthy Applications, St. Louis, Missouri, pp. 1–7 (2005)
22. Li, X., Chang, X., Board, J.A., Kishor, S.: A novel approach for software vulnerability classification. In: IEEE Annual Reliability and Maintainability Symposium (RAMS), (2017). https://doi.org/10.1109/ram.2017.7889792
23. Weka 3–Data Mining With Open Source Machine Learning Software in Java. Available: http://www.cs.waikato.ac.nz/ml/weka/. Accessed Aug 2019
24. Zhang, J., Zulkernine, M., Haque, A.: Random-forests-based network intrusion detection systems. IEEE Trans. Syst., Man, Cybernetics Part-C, Appl. Rev. **38**(5), 649–659 (2008)
25. Neuhaus, S., Zimmermann, T.: Security trend analysis with CVE topic models. In: IEEE International Symposium on Software Reliability Engineering, pp. 111–120 (2010). https://doi.org/10.1109/issre.2010.53
26. Perl, H., Dechand, S., Smith, M., Arp, D., Yamaguchi, F., Rieck, R., Fahl, S., Acar, Y.: VCCFinder: Finding potential vulnerabilities in open source projects to assist code audits. In: 22nd CCS'15, Denver, Colorado, USA, ACM, pp. 426–437 (2015). https://doi.org/10.1145/2810103.2813604
27. Rangwala, M., Zhang, P., Zou, X., Li, F.: A taxonomy of privilege escalation attacks in Android applications. Int. J. Secure. Network **9**, 40–55 (2014). https://doi.org/10.1504/IJSN.2014.059327
28. Raheja, S., Munjal, G., Shagun: Analysis of linux kernel vulnerabilities. Ind. J. Sci. Technol. **9**, 12–29 (2016). https://doi.org/10.17485/ijst/2016/v9i48/138117
29. Haibo, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**, 1263–1284 (2009). https://doi.org/10.1109/TKDE.2008.239
30. Piessens, F.: A taxonomy of causes of software vulnerabilities in Internet software. In: Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering, pp. 47–52 (2002)
31. Pothamsetty, V., Akyol, B.A.: A vulnerability taxonomy for network protocols: Corresponding engineering best practice countermeasures. In: Communications, Internet, and Information Technology, IASTED/ACTA Press, pp. 168–175 (2004)
32. Takahashi, T., Miyamoto, D., Nakao, K.: Toward automated vulnerability monitoring using open information and standardized tools. In: IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pp. 1–4 (2016). https://doi.org/10.1109/percomw.2016.7457049
33. Massacci, F., Nguyen, V.H.: Which is the right source for vulnerability studies? An empirical analysis on mozilla firefox. I:n Proceedings of the 6th International Workshop on Security Measurements and Metrics (MetriSec 2010), pp. 15–15 (2010). https://doi.org/10.1145/1853919.1853925
34. Michael, G., Kishore, S.T.: Software faults, software aging and software rejuvenation. J. Reliab. Eng. Assoc. Jpn **27**, 425–438 (2005)
35. National Vulnerability Database. https://nvd.nist.gov/

36. Igure, V.M., Ronald, D.W.: Taxonomies of attacks and vulnerabilities in computer systems. IEEE Commun. Surv. Tutorials **10**, 6–19 (2008)
37. Khazai, B., Kunz-Plapp, T., Büscher, C.: Int. J. Disaster Risk Sci. **5**, 55 (2014). https://doi.org/10.1007/s13753-014-0010-9
38. Wijayasekara, D., Manic, M., McQueen, M.: Vulnerability identification and classification via text mining bug databases. In: IECON 2014-40th Annual Conference of the IEEE Industrial Electronics Society, pp. 3612–3618 (2014). https://doi.org/10.1109/iecon.2014.7049035
39. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc., San Francisco, CA (2005)
40. Torkura, K.A., Meinel, C.: Towards cloud-aware vulnerability assessments. In: 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS), pp. 746–751 (2015). https://doi.org/10.1109/sitis.2015.63
41. Zhou, Y., Sharma, A.: Automated identification of security issues from commit messages and bug reports. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp. 914–919. ACM, New York (2017). https://doi.org/10.1145/3106237.3117771
42. Bowes, D., Gray, D: Recomputing the Confusion Matrix for Prediction Studies Reporting Categorical Output, Technical Report 509, Univ. of Hertfordshire (2011)
43. Wijayasekara, D., Manic, M., McQueen, M.: Vulnerability identification and classification via text mining bug databases. In: IECON—40th Annual Conference of the IEEE Industrial Electronics Society, pp. 3612–3618 (2014)
44. Ghaffarian, S.M., and Shahriari, H.R: Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. ACM Comput. Surv. **50**(4), 56:1–56:36 (2017). https://doi.org/10.1145/3092566
45. Morrison, P., Herzig, K., Murphy, B., Williams, L.: Challenges with applying vulnerability prediction models. In: Proceedings of the 2015 Symposium and Bootcamp on the Science of Security, pp. 1–9. ACM, New York (2015)