Software Vulnerability Classification Based On Deep Neural Network

Markad Ashok Vitthalrao , Mukesh Kumar Gupta

Abstract: ***Software vulnerability is most common issues in software engineering, many applications has suffering vulnerability, information leakage, and data hijacking such kind of problems facing since couple of years. Sometimes developers should be making some mistakes during code making which generate vulnerability issues for entire application. In this research work, we carried out an approach to software vulnerability detection using deep learning approach behalf of metadata processing. The system carried software vulnerability detection based on the Deep Neural Network (DNN). a new dynamic vulnerability classification approach has suggested. The model basic build based on TF-IDF as well density based feature selection approach for DNN. basically TF-IDF has used to measured the frequency and weight of specific word of vulnerability description; the Vector Space Model (VSM) is used for feature selection to achieve an finest set of feature term, and; the DNN neural network model is used to built an dynamic weakness classifier to achieve effectiveness into the bug detection. The overall system has categorized into four phases in first phase we detect the code clone to eliminate the data redundancy and execution time complexity, in second we apply Vector Space Model (VSM) recommend the re-factor possibility in entire code while in third section we build DNN module for software vulnerability detection and finally recommend the vulnerability for entire code. The system partial implementation has evaluated in java environment which provide satisfactory results for heterogeneous code modules .***

Keywords: Deep neural networks, computer security, data mining, machine learning.

I. INTRODUCTION

Software bug classification is very essential due to the software security during the beta testing. Basically the Quality Assurance (QA) validation phase insure the respective software is a bug free or it does not contain any vulnerability. It is hard to detect various kind of defects of software in real time environment because of different kind of new attacks. The two kind of techniques already introduced in existing authors like a static analysis and dynamic analysis, basically static analysis is the technique where QA analyze for identify the vulnerability based on open background knowledge while dynamic analysis is works like supervised learning approach. In such techniques introduces various machine learning based classification which first train the system and identify the bug during the data testing.

Revised Manuscript Received on October 05, 2019

Markad Ashok Vitthalrao, Research Scholar, Department of Computer Science & Engineering, School of Engineering & Technology, Suresh Gyan Vihar University, Jagatpura, Jaipur.

Dr. Mukesh Kumar Gupta, Principle, Department of Computer Science & Engineering, School of Engineering & Technology, Suresh Gyan Vihar University, Jagatpura, Jaipur.

The quality of software has measure based on (size of code, number of dependencies, code cloning etc), if the software code generate user log files based on current activity, it gives less possibility to external attacker for data piracy. Some authentication, authorization as well as security techniques also boost security parameters of software. The database connection and runtime port open/close is also generate the vulnerability of software during the education. In this research we introduce automated software bug classification using natural language processing and deep learning approaches. The DNN has used to classify the bug and finally measure the accuracy of proposed system which is explore and validate with existing research works.

II. RELATED WORK

To develop a secure software it is hard in today's environment for software developers, still it is too much hard to identifying the vulnerability of existing software's. It is much important to identify the effectiveness of software vulnerability and eliminate search bugs during the software development life cycle (SDLC). Various automatic software has been introduces in last couple of years which automatically identify the software work. Many testing systems already introduced by multiple researchers like a code clone detection, software triage classification etc. According to consideration of OWAPS best software air and service attacks it is much important to boost our current software to defense such attacks. There are different level of bugs in a developed software which is identify is after the effective classification. In the first level we determine the access control level issues where someone an authenticated access our application information with specific network attacks. Many times developers open the connection port but did not close after the process execution which enhances data leakage issue into the software. In second level the access control has been categorized into the three different phases, such violations basically related with Role Base Access Control (RBAC) like process level, communication level and sensitive data disclosure etc. Such vulnerability should we harm sensitive information of application as well as generate a data leakage issue. Third Level bugs should be categorized on security, authentication and authorization while fourth level bugs considered as security session management, bypass authentication, session hijacking and unauthorized access of entire system. During the development of specific software to understand the importance of security and taste the system with penetration testing phase which provides drastic supervision to the system.

III. LITERATURE SURVEY

Guoyan Huang, et. al. [1], proposed a new unthinking vulnerability categorization model based on TFI-DNN. The model is made upon term frequency- inverse document frequency (TF-IDF), info gain

(IG) associate degreed deep neural network (DNN): the TF-IDF is employed to calculate the frequency associate degreed weight of every word from vulnerability description; the human gamma globulin is employed for feature choice to get an optimum set of feature word; and also the DNN neural network model is employed to construct an automatic vulnerability classifier to realize effective vulnerability classification. Basically National Vulnerability Dataset of the US Government has been wont to validate the effectiveness of the planned model.

According to Marian Gawron et. al. [2], there is an automatic different to the manual classification, as a result of the number of known vulnerabilities per day can't be processed manually any longer. They enforced totally different approaches that area unit able to mechanically classify vulnerabilities supported the vulnerability description. They evaluated our approaches that use Neural Networks and therefore the Naive mathematician strategies severally, on the bottom of in public known vulnerabilities.

Andrei Quiroz, ET. al. [3], proposes a Support Vector Machine (SVM) classification model mistreatment Twitter posts (tweets) as a supply for filtering relevant info associated with software system vulnerabilities. During this paper, tweets thought-about relevant are going to be those alerting concerning new susceptibilities in software system (being misused or not), furthermore as posts warning software system users concerning security patches and informs. The non relevant info are going to be thought-about as those that haven't any warning characteristic, i.e.: tweets concerning opinion, general voice communication and topics that haven't any sense of prepared. The projected model achieved associate grade accuracy of ninety four by mistreatment straightforward options like the frequency of words (unigram and bigram). Affordable rates of recall and preciseness into the fascinating category values were recorded as, sixty eight and forty sixth severally for similar straightforward options. This experiment opens a path for future studies concerning the link between however alerts and discoveries in laptop security are expressed by the security community on social media posts.

Jacob A. Harer, ET. al. [4], presented an approach automatic bug detection using machine learning, this system basically introduces in three different layers first it deals with data preprocessing which describe data acquisition, misclassified instances removal, data normalization etc. The preprocessing face gives surety to eliminate such instances which is completely irrelevant with appropriate features. After that system deals with feature extraction and the feature selection phase execute parallel. According to the selected features the train model has build to generate the Background Knowledge (BK) and this BK has used during the classification of vulnerability assessment. This paper also produces using this approach system can achieve around 90% heterogeneous dataset..

Jeesoo Jurn, ET. al. [5], introduced a trend of systems and tools associated with machine-driven vulnerability detection and correction. we tend to propose an automatic vulnerability detection technique supported binary complexness analysis to stop a zero-day attack. They conjointly introduce AN unthinking patch generation technique through PLT/GOT table modification to reply to zero-day vulnerabilities.

According to Zhen Li et al [6], the finding of software susceptibilities (or vulnerabilities for short) are a vital downside that has notwithstanding to be tackled, as recognized by multiple bugs according to a routine. This incorporate machine learning ways that to automatic vulnerability detection. Deep learning is attractive for this purpose as a result of it does not want human consultants to manually define choices. The experiments analysis has done of system with four computer code merchandise demonstrate the utility of the framework: we tend to notice fifteen vulnerabilities that aren't according within the National Vulnerability info. Among these fifteen vulnerabilities, seven are unknown and are according to the vendors, and therefore the alternative eight are "silently" patched by the vendors' once emotional newer versions of the products.

According to Antonino Sabetta, ET. al. [7], a method that uses machine-learning to investigate ASCII text file repositories and to mechanically determine commits that area unit security-relevant (i.e., that area unit probably to mend vulnerability). They treat the ASCII text file changes introduced by commits as documents written in language, classifying them victimization commonplace document classification strategies. By combining freelance classifiers that use info from completely different sides of commits, our methodology will yield high preciseness (80%) whereas guaranteeing acceptable recall (43%). the utilization of data extracted from the ASCII text file changes yields a considerable improvement over the most effective identified approach within the state of the art, whereas requiring a considerably smaller quantity of coaching knowledge and using a simpler architecture.

Tamas Abraham, ET. al. [8], find that the first focus isn't solely on discovering new approaches, however on serving to Foreign Intelligence Service practitioners by simplifying and automating their processes. Considering the variability of applications already obvious, we have a tendency to believe mil can still offer help to Foreign Intelligence Service within the future as new area unitas of use are explored and improved algorithms to boost existing functionality become available.

HoaKhanh Dam, et. al. [9], described a new approach, built upon the powerful deep learning Long Short Term Memory model, to mechanically learn each linguistics and syntactical options of code. Our analysis on eighteen humanoid applications and therefore the Firefox application demonstrates that the prediction power obtained from our learned options is best than what's achieved by state of the art vulnerability prediction models, for each within-project prediction and cross-project prediction.According to Rebecca L. Russell et. al. [10], growing numbers of software package vulnerabilities unit of measurement invent every year whether or not the unit of measurement reportable widely or describes inside in detected code. These susceptibilities can cause danger of exploit and

finish in system compromise, data leaks, or denial of service. we tend to leverage the wealth of C and C++ ASCII computer file code accessible to develop oversized scale function-level condition detection system victimization machine learning. To addition vacant labeled exposure datasets, they compiled a huge dataset of uncountable ASCII computer file role different classification methods has used to generate the test labels of system. Victimization of these datasets, they developed a fast and ascendible vulnerability detection tool supported deep feature illustration learning that directly interprets flexed ASCII computer files. we tend to gauge our tool on code from every real package packages and thus the agency soaks up four benchmark dataset. The outcome has demonstrate proposed learning phase learning on ASCII text file could be a promising approach for automated software vulnerability detection.

Jiadong Ren, ET. al. [11], proposed a package buffer overflow vulnerability prediction methodology by exploitation software metrics including an alternative tree formula. First, the software metrics were far from the software ASCII document, associate degreed data from the dynamic data stream at the helpful level was extracted by a data mining technique. Second, a model supported an alternative tree formula that was created to measure multiple varieties of buffer overflow vulnerabilities at a helpful level. Finally, the experimental results showed that our technique ran in less time than SVM, Bayes, AdaBoost, and random forest algorithms and earned eighty 2.53% and 87.51% accuracy in a pair of fully completely different data sets. The technique given during this paper earned the impact of accurately predicting package buffer overflow susceptibilities in C/C++ and Java programs.

IV. SYSTEM ARCHITECTURE

The proposed system deals with software that detect the software code clones, with or without alteration. System grows a self-Learning machine that categorizes similar and dissimilar code wreckages, the algorithm for Code mining and Code retrieval. Proposed Hybrid Model for Software Code Clone Detection by unverified learning, discount techniques, and code resemblance comparator and fins the semantic or useful similar codes. After complete execution of this phase it will produces the code snippet which contains the bugs. Finally describe non-functional methods using reduction that reduces comparison, recuperates system presentation, decreases complexity, and upsurges system maintainability

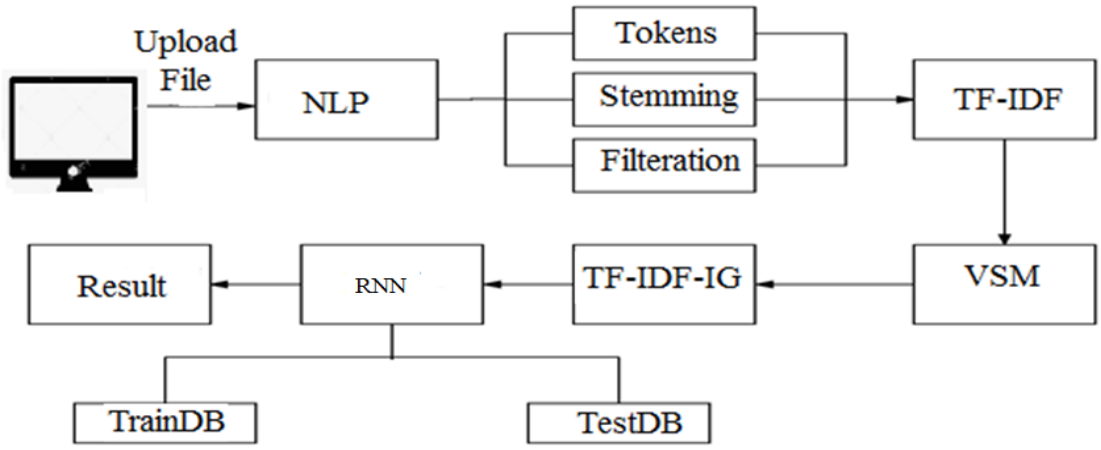

Figure 1: Proposed System Overview

Data Reduction:

- Reducing the Data Scale – Sometime input data should be contains some redundant information or features, in reduction we need to eliminate such features or misclassified instances, in this section we reduce the actual data scale.
- The processed called normalized data should be balanced, it is easy to handle for moderately than the imaginative data set.

Train data and Preprocessing:

- In this phase we construct the dynamic Background Rules called BK as a desired input dataset.
- System extract the metadata from given input object and apply preprocessing and normalization etc.
- Then tokenization, porter's stemmer and filtration will execute.
- Finally, TF-IDF will provide the accessibility of present vector and accumulate it into feature database.
- Once train execution has done, it dynamically generate the feature set for each object.

Testing phase with Preprocessing and TF-IDF:

- Need to upload the data object which does not have existing labels for desired class item.
- It also work like train module, based

on density of each terms.

- Then features have been extracted & DNN will determine the relationship vector with extracted training set.

Feature Selection:

- This section contains two different techniques features extraction as well as features selection.
- In features extraction we extract numerous unique features using extraction technique. TF-IDF, Co-relation coefficient techniques has used for extract the features.
- Select the features using binary feature selection technique.

Clone Detection:

- First detect the multiple clones
- Find the re-factor possibility of all clones

Bug Triage:

- Find the different type exception which generates at runtime environment.
- Create the bug report for system and notify to developer.
- Design and develop an approach for NLP as well as data reduction, train module TF-IDF.
- Implementation of DCNN training phase which generates the rules and simultaneously generate VSM.
- Implement the testing phase of DCNN and find the bugs of given input module as well as system
- Validation of the developed system as compared to existing system through extensive experiments.

V. RESULTS AND DISCUSSION

In this work briefly we explained the how to detect software code v with respect to its different types at class level single file. Within single java file we detected vulnerability of code which will be show complexity of software. Package level code clone detection and Refactoring of code clone will be also possible by applying syntactic and semantic rules as per methodology of Statement Mapping, Preconditions Examination and Abstract Syntax Tree mechanism. In this mechanism we will apply in future scope using same methods.

Table 1 : System performance with accuracy

RNN	ANALYSS
ACCURACY	90.60
PRECISION	89.40
RECALL	88.70
F1 SCORE	87.50

The above table 1 shows classification accuracy of RNN with extracted features of code respectively. Basically around 3000 account initial input data has given for classification, execute the train and test module respectively. It provides average accuracy 90.60 for proposed research.

VI. CONCLUSION

Basically system presented a framework for software vulnerability detection based on deep learning to improve the detection of bugs in source code. The system provided an overview of the static analysis tools and techniques and subsequently detailed the proposed vulnerability detection based on the deep neural network. It will much helpful in checking the small scale software applications, rather than high configuration software's. It also focus on to identify code clone as well as bug triage with assessment sheet, it is much essential for software developer to trace the actual bug place in large scale application for real time environments. Initial findings suggest that DNN is an effective tool for bug detection in large Java applications.

REFERENCES

1. Huang, Guoyan, Yazhou Li, Qian Wang, Jiadong Ren, Yongqiang Cheng, and Xiaolin Zhao. "Automatic classification method for software vulnerability based on deep neural network.", IEEE Access (2019).
2. Gawron, Marian, Feng Cheng, and Christoph Meinel. "Automatic vulnerability classification using machine learning." In International Conference on Risks and Security of Internet and Systems, pp. 3-17. Springer, Cham, 2017.
3. Queiroz, Andrei, Brian Keegan, and Fredrick Mtenzi. "Predicting Software Vulnerability Using Security Discussion in Social Media." (2017).
4. Harer, Jacob A., Louis Y. Kim, Rebecca L. Russell, Onur Ozdemir, Leonard R. Kosta, Akshay Rangamani, Lei H. Hamilton et al. "Automated software vulnerability detection with machine learning." arXiv preprint arXiv:1803.04497 (2018).
5. Jurn, Jeesoo, Taeeun Kim, and Hwankuk Kim. "An Automated Vulnerability Detection and Remediation Method for Software Security." Sustainability 10, no. 5 (2018): 1652.
6. Li, Zhen, Deqing Zou, Shouhuai Xu, Hai Jin, Yawei Zhu, Zhaoxuan Chen, Sujuan Wang, and Jialai Wang. "SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities." arXiv preprint arXiv:1807.06756 (2018).
7. Sabetta, Antonino, and Michele Bezzi. "A Practical Approach to the Automatic Classification of Security-Relevant Commits." In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 579-582. IEEE, 2018.
8. Han, Yi, Benjamin IP Rubinstein, Tamas Abraham, Tansu Alpcan, Olivier De Vel, Sarah Erfani, David Hubczenko, Christopher Leckie, and Paul Montaguea. "Reinforcement learning for autonomous defence in software-defined networking." In International Conference on Decision and Game Theory for Security, pp. 145-165. Springer, Cham, 2018.
9. Dam, Hoa Khanh, Truyen Tran, Trang Thi Minh Pham, Shien Wee Ng, John Grundy, and Aditya Ghose. "Automatic feature learning for predicting vulnerable software components." IEEE Transactions on Software Engineering (2018).
10. Russell, Rebecca, Louis Kim, Lei Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul Ellingwood, and Marc McConley. "Automated vulnerability detection in source code using deep representation learning." In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 757-762. IEEE, 2018.

11. Ren, Jiadong, Zhangqi Zheng, Qian Liu, Zhiyao Wei, and Huaizhi Yan. "A Buffer Overflow Prediction Approach Based on Software Metrics and Machine Learning." Security and Communication Networks 2019 (2019).

APPENDIX

Mr. Markad Ashok Vitthalrao did B.E in Information Technology from Pune University in 2006 and M.E in Computer Engineering from Pune University in 2013. He is Currently pursuing Ph.D in Computer Science & Engineering from Suresh Gyan Vihar University Jaipur (Rajasthan). His research area is Machine Learning and Deep Learning.

Name: Dr. Mukesh Kumar Gupta
Designation: Professor & Associate Dean (Research) Department: Electrical Engineering
Qualification: Ph.D., M.E. (Power System), B.E. Area of Interest: Power System & Renewable Energy