Open Access
Full Length Article

ORIGINAL ARTICLE

Automated Vulnerability Detection for Software using NLP Techniques

Hafiza Iqra Shamim[1], Hafiza Sana Shamim[2], Asghar Ali Shah[3]

[1, 2, 3] Department of Computer Science, Bahria University, Lahore Campus

ABSTRACT

For information security, certain vulnerabilities can lead to major effects which later can cause consequences for economic, social, and national security. It's predicted that through the use of top-notch NLP technologies in the field of vulnerability detection, a model can be created for the accomplishment of the automatic analysis which will help in the detection of particular text files such as source code. An in-depth learning-based vulnerability detector is proposed that pulls out some features through the help of an RNN composite neutral network. They employ a SARD and NVD dataset of lots of open source roles, which will be tagged with results after three static analyzers that hint at possible activities. Through the implementation of this data set, a fast and accessible detection system will come into being with built-in deep feature sign knowledge that will directly translate the source code. Based on our research, we found out that through deep feature representation learning on source code the software vulnerabilities can be automatically unrevealed. In this study, the model vulnerability detection is used as a natural language processing (NLP) problem where source codes are handled as texts. Furthermore, advanced deep learning NLP models are there which offer transfer learning on written English to address/look into automated software vulnerability detection. In short, we have used different algorithms i-e, machine learning, and deep learning.

Keywords: NLP, RNN, Open-source, Software vulnerability

Author`s Contribution
[1] Data analysis, interpretation, and manuscript writing, Active participation in data collection, [2] Conception, synthesis, planning of research, [3.] Interpretation and discussion

Address of Correspondence
Hafiza Iqra Shamim
Email: iqrashamim1996@gmail.com

Article info.
Received: October 03. 2021
Accepted: December 26, 2021
Published: December 30, 2021

Cite this article: *Shamim HI, Shamim HS, Shah AA. Automated Vulnerability detection for software using NLP techniques. J. inf. commun. technol. robot. appl.2021; 12(2):48-57.*

Funding Source: *Nil*
Conflict of Interest: *Nil*

INTRODUCTION

Vulnerability detection was always most important in the world of safety software. Computerized inspections and weaknesses identification have been innovated and changed to stream research centers with huge source code. One of the most active studies acting weaknesses is to use the absolute of the absolute most of the modern innovation of NLP art, and achieve programmed surveys and detect source code for specific text documents. This page provides a variety of new records and achievements and an old technical summary such as Codebert [1]. For pretty an extended time, figuring out protection weaknesses in the application earlier than they're taken gain of has been a difficult errand. Conventional

strategies for code exams were introduced, however, they're usually fruitless and wasteful.

In this study, the version programming shortcoming disclosure as an NLP trouble with supply code treated as textual content, and the utilization superior substantial getting to know NLP fashions with circulate getting to know on created English to deal with robotized programming shortcoming area. The preprocessed NIST NVD/SARD dataset then made a dataset of 100,000 information in the C application layout language via way of means 123 styles of shortcomings for making plans and testing. To the quantity monitoring down protection deficiencies, the large assessments produce first-rate results, with over 93% precision [2]. A huge part of associated publications of motion depends on clearly defined shortcoming viewpoints and paintings problem to recommendations or code similarities. To decide this trouble, multiple scholastics have encouraged that neural institutions with unfastened element extraction limits be used to moreover foster ID information. Regardless, there are one-of-a-kind styles of neural institutions, and the records preprocessing methodologies used will vastly affect version execution [3]. "How properly nation of the artwork DL-primarily based techniques act in a real shortcoming gauge circumstance?" The ask on this audit. Their display brings someplace close to over half, lots incredibly [4].

Picking the proper neural affiliation and records preprocessing method for particular trouble is a tricky undertaking for creators and skilled professionals. To decide this trouble, The analysts directed a large exploration of programming imperfection acknowledgment issues, differentiating the exhibit of maximum ordinary neural institutions (i.e., Bi-LSTM and RVFL) via way of means of maximum ordinary information preprocessing approaches (i.e., vector depiction additionally software program engineer strategies of symbolization), then concocted numerous fascinating exam results. The professionals determined that: 1) RVFL`s readiness pace is in each case faster than Bilstm's, but Biestimate LSTM's precision is better than Rvfl's; 2) making use of doc2vec for vector portrayal can similarly broaden the version's making plans pace and speculation restrict over word2vec, and 3) staggered symbolization can help with running at the exactness of neural affiliation fashions [5]. Computer programmers are regularly making use of Natural Language Processing (NLP) approaches to address mechanize the assessment of Software Vulnerabilities (SVs) in mild portrayals in open repositories [3]. The used AI to foster a massive extension of paintings degree shortcoming revelation device the use of the abundance of open-supply C and C++ code available. The researchers accumulated a large dataset of plenty of open supply works referred to as the use of discoveries from 3 wonderful constant analyzers that display expected efforts to enhance contemporary named weak spot datasets [13]. This article discusses a developing fashion in mechanized weak spot detection and remedy methodologies and advancements. This inquiry shows a mechanized weak spot differentiating evidence method primarily based totally on parallel intricacy research to save you a zero-day attack. Foster a programmed restore improvement method by the use of PLT/GOT desk extrade to reply to zero-day flaws [14]. This task aims to illustrate how keeping apart textual content highlights from competencies in C supply code and deconstructing them through the use of an AI classifier can bring about a substantial discount in difficulties [15].

LITERATURE REVIEW

In the world of programming security, vulnerability discovery has constantly been the most introductory responsibility. Robotized disquisition and identification of sins has turned into an eclipse and inflow exploration center as invention propels and despite colossal source law. One of the most anticipated investigations in the field of weakness discovery is the application of the absolute most state of the art NLP advances to make models and achieve the programmed examination and identification of source law for specific textbook documents, for illustration, source law. This runner gives a speedy figure of different new records and advancements, like CodeBERT, just as a rundown of further seasoned inventions (6). Post-arrangement, security defects in transportation systems may bring about unanticipated contrivance breakdown, frame crashes, or malignant abuse by wafers. It's pleasurable when these defects are planted and amended beforehand before the item is delivered. CWE is for Common Weakness Estimation,

and it's the wording for portraying normal sins in C law. The scientists present a profound literacy model for feting presumably the most well-known kinds of safety sins in source law in this study. To disentangle law weakness areas of interest, also tone- consideration associations. By taking advantage of law AST construction, our methodology fosters an exact handle of law semantics with far lower learnable variables (7). As of in the not-so-distant history, there had been close to no assessment into conveying the 6 studies of move sorting out some way to the field of NLP. In any case, new developments including getting ready and tweaking fashion are being made (8). In (9), the makers present VulDeePecker, a significant literacy-grounded failing ID structure. VulDeePecker gathers tests by taking outlaw tools (i.e., law sections that live semantically connected to the fault) after imperfect computer programmers and subsequently changing over them into vector descriptions. In the literacy calculation, the Long Short-Term Memory is employed (LSTM). VulDeePecker outperforms the public with the skill-failing disclosure styles to the extent of both delicacy and acceptability, as shown by tests. (10) Has made different AI models for distinguishing goofs in C/ C law that could provoke security handovers. Recently, with the grim advancement of programming development, precipitously more programming has been made by people. While people share in the solace brought by programming, they're also undermined by programming failings. It could be said that item failings are presumably the most significant issue that compromises the customary movement of programming. For programming guests, the prompt and unusual plutocrat-related rigors achieved by programming failings worldwide have outperformed numerous bones. It's easily a fact that there are colorful failings in utmost programming. There are numerous kinds of programming failings, for case, CVE-2015-8558 (11). Alongside an ever-evolving system for effectively perceiving law failing, this model also can pinpoint the law corridor that was declared vulnerable by the model. Accordingly, a specialist may zero in indeed more snappily on the unsafe law locales, which transforms into the" sensible" part of the failing acknowledgment. The recommended AI accomplishes an F1- score of98.40 percent on definite CWEs since the benchmarked NIST SARD dataset, which looks fine to the top league (5). Our contrivance was taken a stab at law from authentic programming packs also as the NIST SATE IV standard dataset (13). One explanation is that colorful item clones of a relative failing may live, making it delicate to screen, in reality (e.g., different performances of libraries and operations) (17).

Systematic review studies

In total, we found 15 relevant studies in the sources we looked at, as shown in this table. Total studies were classed as scalar articles, including one meta-analysis [5]. Ten of the papers looked at technology evaluation concerns, while the other five looked at research trends. Software Vulnerability Detection guidelines were mentioned in all papers. As a result, the research directly linked itself to Software Vulnerability Detection. In terms of where SLRs are published, IEEE, ACM, and Google.

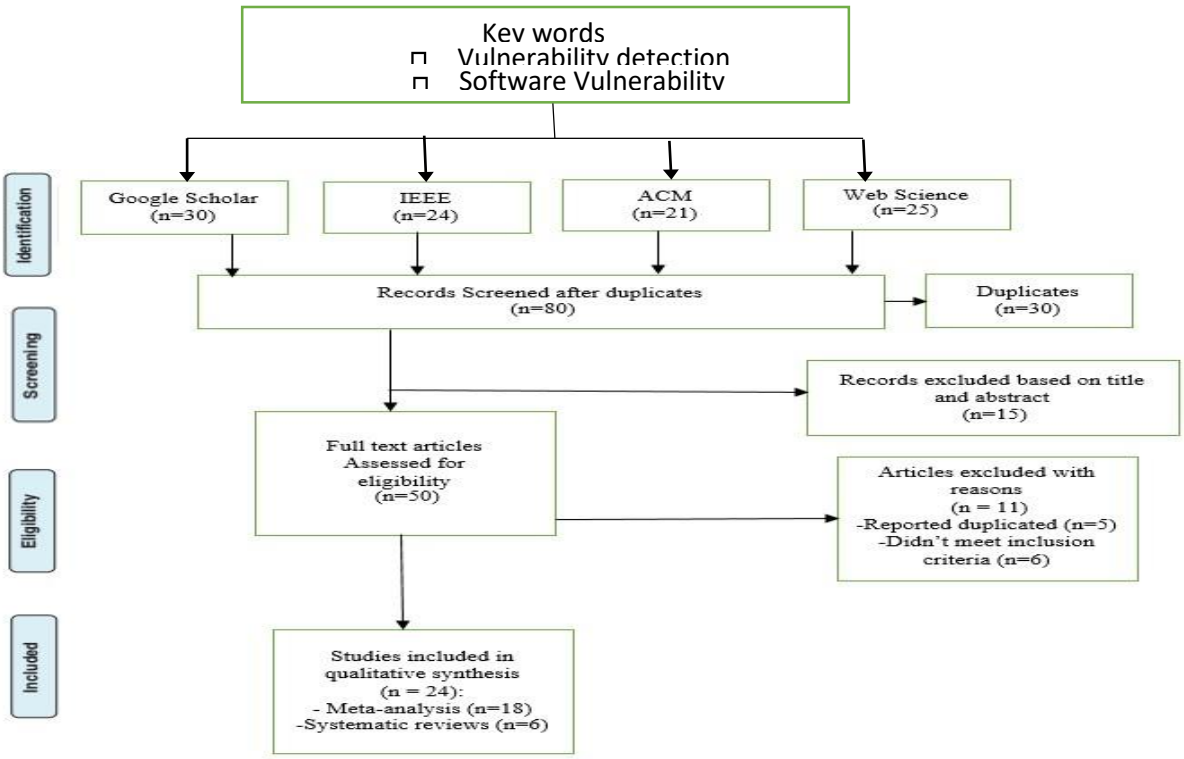
Table1: Systematic Review Studies

ID	Author	Year	Topic Area	Article type	Num. primary
R1	J. Wu [1]	2021	Software Vulnerability	Scholarly articles	107
R2	N. Ziems, S. Wu [2]	2021	Software Vulnerability	Scholarly articles	24
R3	G. Tang, L. Meng [3]	2020	Software Vulnerability	Scholarly articles	32
R4	G. Tang, L. Meng [4]	2020	Software Vulnerability	Scholarly articles	32
R5	A Tanwar, H Manikandan[5]	2021	Software Vulnerability Detection	Scholarly articles	76
R6	J. Howard, S. Ruder [6]	2018	Text Classification	Scholarly articles	10
R7	Z. Li, D. Zou [7]	2018	Software Vulnerability	Scholarly articles	49
R8	J. A. Harer, L. Y. Kim [8]	2018	Software Vulnerability Detection	Scholarly articles	31

R9	J Akram, P Luo [9]	2021	Software Vulnerability Detection	Scholarly articles	34
R10	Le, B. Sabir [10]	2019	Software Vulnerability	Scholarly articles	56
R11	R. Russell, L. Kim [11]	2018	Software Vulnerability	Scholarly articles	24
R12	J. Jurn, T. Kim [12]	2018	Software Vulnerability	Scholarly articles	16
R13	B. Chernis and R. Verma [13]	2018	Software Vulnerability Detection	Scholarly articles	22
R14	S. Chakraborty, R. Krishna [14]	2021	Matrics	Scholarly articles	10
R15	Z. Li [15]	2016	Software Vulnerability	Scholarly articles	30

RESEARCH METHODOLOGY

The hunt procedure and final composition selection are depicted in Figure 1. There were 80 papers originally planted, and 30 papers rejected the cause of duplication. Of the 80 papers reviewed, 50 were planted to be conceivably respectable for addition, while the remaining 15 were rejected grounded on the title and abstract. After carrying full textbooks and applying eligibility criteria to the remaining 24 papers.



Search Sources

The present advanced assortments are stylish places to search for books, magazines, and papers. Because of Defined means and the huge number of papers regarding this matter, we pick three motorized libraries for this jotting check. Three distinctive motorized libraries were employed to execute an examination

- ACM
- IEEE
- Google Scholar
- Web Science

Since there have been similar in numerous reports near picture inscribing, we limited our jotting inspection to papers led over the most recent four times â€" 2016 to 2019. We sifted through papers that were posted under the software engineering subject during our examination in the motorized library.

Inclusion Criteria

For the last five times, experimenters have been looking at natural language issues. The effectiveness of models has been extensively bettered due to recent developments in artificial intelligence (AI). Still, the issues are rightly satisfactory. Since machines cannot mimic mortal minds or the way they interact, it will still be a challenge. It's extremely delicate to keep up with the growing quantum of knowledge on this content. It's delicate to keep up with the current findings and results in the world of image captioning. A rigorous Methodical Literature Review (SLR) offers a summary of image captioning developments over the last four times in this composition. The paper's crucial end is to clarify the most common strategies and difficulties of image captioning, as well as to summarize the findings. During this analysis, inconsistencies in the findings attained in image captioning were discovered, and as a result, this paper raises mindfulness of shy data collection. As a consequence, it's critical to equate the issues of a recently developed model with the most recent data, not just with state-of-the-art approaches. For experimenters, this SLR is a source of certain material. This SLR is a depository of similar data for experimenters so that they can compare results directly before reporting new accomplishments in the field of image caption generation.

Exclusion Criteria

This systematic literature review (SLR) examines the various deep learning frameworks for image captioning in depth. To conduct the study, we combed through papers from three scholarly libraries, applied inclusion and exclusion criteria to all of them, and chose 12 primary studies for a literature review. We collected the data and processed it using a data extraction mechanism. We collected the data and extensively analyzed it using data extraction mechanisms

RESULT AND ANALYSIS

Quality Assessment Criteria

Table2. Categories and grading to access the quality of the selected study

Items	Description	Grads
A	Title	0=inaccurate 1=possibly accurate 2=clearly accurate
B	Abstract	0=inaccurate 1=possibly accurate 2=clearly accurate
C	Introduction Background-objective, experimental approach	0=inaccurate 1=possibly accurate 2=clearly accurate
D	Introduction Objective-primary and secondary	0=inaccurate 1=possibly accurate 2=clearly accurate
E	Methods Nature of the review permission	0=inaccurate 1=possibly accurate 2=clearly accurate
F	Method Study Design number of the experimental and control group	0=inaccurate 1=possibly accurate 2=clearly accurate
G	Method Experimental procedure-precise detail	0=inaccurate 1=possibly accurate 2=clearly accurate

Quality Assessment Results

Table3. Quality Assessment Results								
References	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Total**
Literature review on vulnerability detection using NLP technology, 2021[1]	2	1	1	2	1	0	2	9
Security Vulnerability Detection Using Deep Learning Natural Language Processing, 2021[2]	1	0	2	0	1	1	0	6
A Comparative Study of Neural Network Techniques for Automatic Software Vulnerability Detection, 2020[3]	1	1	0	2	1	0	2	7
Multi-context Attention Fusion Neural Network for Software Vulnerability Identification, 2021[4]	2	1	1	2	0	1	1	8
Universal Language Model Fine-tuning for Text Classification, 2018[5]	2	1	1	0	0	1	2	7
VulDeePecker: A Deep Learning-Based System for Vulnerability Detection, 2018[6]	1	1	0	0	1	1	2	6
Automated software vulnerability detection with machine learning, 2018[7]	2	2	1	1	0	1	2	9
SQVDT: A Scalable Quantitative Vulnerability Detection Technique for Source Code Security Assessment, 2021[8]	1	1	0	2	2	0	2	8
Automated Software Vulnerability Assessment with Concept Drift, 2019[9]	2	0	2	2	1	0	0	7
Automated Vulnerability Detection in Source Code Using Deep Representation Learning, 2018[10]	1	0	0	0	2	1	0	4
An Automated Vulnerability Detection and Remediation Method for Software Security, 2018[11]	2	1	0	0	1	1	0	5
An Automated Vulnerability Detection and Remediation Method for Software Security Machine Learning Methods for Software Vulnerability Detection, 2018[12]	1	2	1	1	0	0	0	5
Deep Learning based Vulnerability Detection: Are We There Yet," IEEE Transactions on Software Engineering, 2021[13]	0	0	1	1	2	1	0	5
Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016[14]	0	1	1	0	1	0	0	3

Heat Map

Table4. Quality Assessment Heat Map								
References	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Total**
Literature review on vulnerability detection using NLP technology, 2021[1]	2	1	1	2	1	0	2	9
Security Vulnerability Detection Using Deep Learning Natural Language Processing, 2021[2]	1	0	2	0	1	1	0	6
A Comparative Study of Neural Network Techniques for Automatic Software Vulnerability Detection, 2020[3]	1	1	0	2	1	0	2	7

Multi-context Attention Fusion Neural Network for Software Vulnerability Identification, 2021[4]	2	1	1	2	0	1	1	8
Universal Language Model Fine-tuning for Text Classification, 2018[5]	2	1	1	0	0	1	2	7
VulDeePecker: A Deep Learning-Based System for Vulnerability Detection, 2018[6]	1	1	0	0	1	1	2	6
Automated software vulnerability detection with machine learning, 2018[7]	2	2	1	1	0	1	2	9
SQVDT: A Scalable Quantitative Vulnerability Detection Technique for Source Code Security Assessment, 2021[8]	1	1	0	2	2	0	2	8
Automated Software Vulnerability Assessment with Concept Drift, 2019[9]	2	0	2	2	1	0	0	7
Automated Vulnerability Detection in Source Code Using Deep Representation Learning, 2018[10]	1	0	0	0	2	1	0	4
An Automated Vulnerability Detection and Remediation Method for Software Security, 2018[11]	2	1	0	0	1	1	0	5
An Automated Vulnerability Detection and Remediation Method for Software Security Machine Learning Methods for Software Vulnerability Detection, 2018[12]	1	2	1	1	0	0	0	5
Deep Learning based Vulnerability Detection: Are We There Yet," IEEE Transactions on Software Engineering, 2021[13]	0	0	1	1	2	1	0	5
Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016[14]	0	1	1	0	1	0	0	3

Forest Plot

Odds ratios (squares proportional to weights used in meta-analysis); summary measure (centerline of the diamond); associated confidence intervals (lateral tips of the diamond); names of fictional studies on left; odds ratios and confidence intervals on right; odds ratios (squares proportional to weights used in meta-analysis); odds ratios (squares proportional to weights used in meta-analysis); odds ratios (squares proportional to weights used in meta-analysis); odds ratios (squares proportional a vertical line that is solid and has no effect.

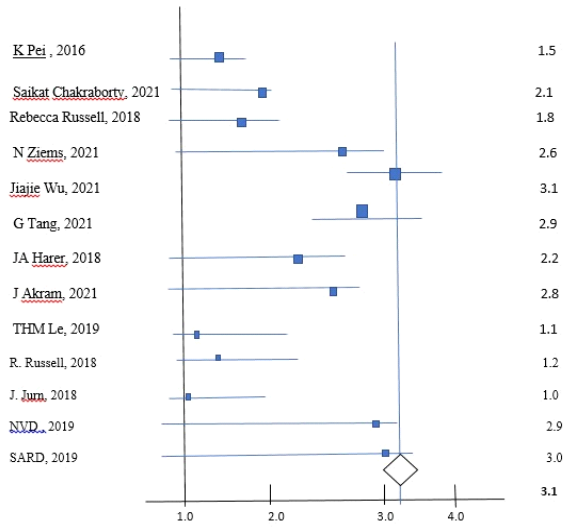


Figure 2. Forest Plot

DISCUSSION

Dataset

National Vulnerability Database (NVD) [16] and software guarantee Reference Dataset (SCRD) [17] are two appreciably used vulnerability statistics resources.

NVD: The national Vulnerability Database

(NVD) became established in 2000 by way of American authorities as a repository for requirements-based vulnerability control. This dataset is primarily based on the not unusual vulnerabilities and Exposures (CVE) listing and is fully synced with it. It commonly analyses CVEs that have been published in the CVE Dictionary. As a result, any CVE updates will appear in NVD right away.

SCRD: This dataset can deliver a listing of recognized protection troubles to users and researchers. more than one test instance from diverse assets, such as production programs, artificial, and educational, are included in the SARD collection. SARD packages are broken up into three classes: packages, which don't have any vulnerabilities, applications, which have flaws, and applications, which have vulnerabilities and the associated patch variations.

A dataset is a logically prepared series of statistics this is generally connected to a certain frame of work. We can use the SARD and NVD dataset inside the studying segment by extracting code metrics from a massive variety of source code files, some of which are prone and some of which can be no longer. The dataset consists of 1,591 NVD open-source C/C++ packages and 14,000 SARD open-source C/C++ programs.

When it comes to time and budget, manually detecting vulnerable code is very hard and costly. To minimize the cost, developers started using automatic vulnerabilities tools (AVP). Today, developers started using deep learning techniques on AVP. All the suggested perspectives are established on the technique of feature extraction pressed by previous applications of deep learning (automatic language processing) [25]. This revolt is expedited by big code from open sources projects to provide an amazing performance using a different model of machine learning. In the context of the data-driven paradigm, this paper enlightens the recent analytic research on the cyber code of venomous and common software using different concepts of similarity, correlation, and collective indication [26].

This has inspired the researchers of different communities such as cyber security to implement deep learning techniques so that it can help in learning and understanding vulnerable code patterns and semantics indicates [27]. In the research, we used ML classifier algorithms such as Linear Support classification, Random Forest, and Naive Bayes Classifiers, and to check their outcomes, we uses three different metrics (precision, recall, and F1-score evaluation metrics) [28]. So far, reports have been manually categorized using risk specifiers, which has led to errors in human motivation and scalability due to a lack of security experts [29].

The results of our mapping research can be used to identify research opportunities in the areas of software risk ratings and automated vulnerability remediation technologies [30]. This research evaluated the uses of deep neural network models, and traditional models like random forest and found that learning with tree-based models generated the best results [31]. Deep learning approaches for anomaly-based network intrusion detection are studied in which a comparison is made between observed anomalies and traditional machine learning techniques like the random forest, SVM and Ada boosting. Other optimizers can be used to discover weaknesses such as the arithmetic optimization approach [32] and the Aquile optimizer [33]. Consequently, educated highlights are more vivid in [34].In comparison to existing comparative methodologies dispersed tactics, the suggested strategy produced a positive result.

For abnormality identification altered in line with Apache Spark in-memory processing stage, a fake brain network-based solution is proposed in this research [35]. Closest neighbor, choice trees, and backing vector machines are used to compare the suggested model against three common AI algorithms. By building out relations between units [36] and natural language [37] processing recurrent neural networks (RNNs) can recall irregular length succession of designs. We designed a product-based location Framework that can execute both source code and bytecode to overcome the lack of

bytecode security. Finally, we direct basic studies toward constructing a reliable weak data set, and a concise examination is offered [38].

The article [39] examines Three Types of weakness location strategies to more easily assess the presence of profound component learning weakness recognition innovation. To reduce the exploration of pointless approaches, writing begins with the age and selection of smart seedlings. The solution provided in this article is NeuFuzz, Which employs brain organizations to extract hidden vulnerability designs from a large number of defenseless and clean program execution methods. Use expectation model to determine if inconspicuous where is vulnerable during online directed fluffing, and seed is set apart as suggested by forecast outcome and then added to the seed line. Finally, in the subsequent seed determination cycle and seed transformation, weak seeds will be targeted and given additional change energy [40]. On this Foundation, the writing uses the multi-head pointer cute simultaneous arrangement, Positioning, and repair allowing for fine-grained positioning of explicit factors and completion of maintaining cycle [41]. Writing provides the Deep Repair learning structure, which incorporates repeated or rehashed codes in the program and focuses on and converts assertion in the code library to build program fix parts based on the rule of code compatibility [42].

To build a relapse model rather than a classifier model researchers used 3 entirely connected backpropagation brain organizations. Furthermore the Model Neural Network Regression approach (NNR) Text and mathematical measures of code changes and feed them into a brain network, with the outcome indicating the likelihood that the code modification under test includes problems [43]. In 2017 vulnerability in Apache Struts was discovered resulting in the compromising of 143 million buyers' financial information [44].

CONCLUSION

The proposed a profound mastering-based weakness discovery calculation that concentrates on highlights using an RNN composite brain agency. They utilize a SARD and NVD dataset of masses of open source jobs so that it will be categorized with consequences after three static analyzers that clue to capacity exercises. using this dataset, foster a brief and open weak point discovery framework based totally on profound issue sign records that straightforwardly decipher source code. Our discoveries suggest that a profoundly detailed portrayal of mastering supply code may be applied to discover programming weaknesses. in this review, the version programming weakness identification as a feature language managing (NLP) trouble with supply code took care of as texts, and make use of progressed profound learning NLP models with flow learning on composed English to deal with mechanized programming weakness reputation. Diverse calculations might be analyzed (AI and profound getting to know).

REFERENCES

1. . Wu, “Literature review on vulnerability detection using NLP technology,” arXiv.org, 23-Apr-2021. [Online]. Available: https://arxiv.org/abs/2104.11230v1. [Accessed: 09-Nov-2021].
2. N. Ziems and S. Wu, “Security Vulnerability Detection Using Deep Learning Natural Language Processing,” arXiv.org, 06-May-2021. [Online]. Available: https://arxiv.org/abs/2105.02388. [Accessed: 09-Nov-2021].
3. G. Tang, L. Meng, H. Wang, S. Ren, Q. Wang, L. Yang, and W. Cao, “A Comparative Study of Neural Network Techniques for Automatic Software Vulnerability Detection,” 2020 International Symposium on Theoretical Aspects of Software Engineering (TASE), 2020.
4. G. Tang, L. Meng, H. Wang, S. Ren, Q. Wang, L. Yang, and W. Cao, “A Comparative Study of Neural Network Techniques for Automatic Software Vulnerability Detection,” 2020 International Symposium on Theoretical Aspects of Software Engineering (TASE), 2020.
5. “Multi-context Attention Fusion Neural Network for Software Vulnerability Identification” [Online]. Available: https://arxiv.org/pdf/2104.09225. [Accessed: 12-Nov-2021].
6. J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” arXiv.org, 23-May-2018. [Online]. Available: https://arxiv.org/abs/1801.06146. [Accessed: 12-Nov-2021].
7. Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, and Y. Zhong,
8. “VulDeePecker: A Deep Learning-Based System for Vulnerability Detection,” Proceedings 2018 Network and Distributed System Security Symposium, 2018.
9. J. A. Harer, L. Y. Kim, R. L. Russell, O. Ozdemir, L. R. Kosta, A. Rangamani, L. H. Hamilton, G. I. Centeno, J. R. Key, P. M. Ellingwood, E. Antelman, A. Mackay, M. W. McConley, J. M. Opper, P. Chin, and T. Lazovich, “Automated software vulnerability detection with machine learning,” arXiv.org, 02-Aug-2018. [Online]. Available: https://arxiv.org/abs/1803.04497. [Accessed: 12-Nov-2021].
10. “SQVDT: A Scalable Quantitative Vulnerability Detection ...” [Online]. Available:https://www.researchgate.net/publication/345973173_SQVDT_A_Scalable_Q uantitative_Vulnerability_Detection_Technique_for_Source_Code_Security_Assessmen t. [Accessed: 12-Nov-2021]. [10] T. H. M. Le, B. Sabir, and M. A. Babar, “Automate Software Vulnerability

Assessment with Concept Drift," 2019 IEEE/ACM16th International Conference on Mining Software Repositories (MSR), 2019.
11. R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer, O. Ozdemir, P. Ellingwood, and M. Mcconley, "Automated Vulnerability Detection in Source Code Using Deep Representation Learning," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018.
12. J. Jurn, T. Kim, and H. Kim, "An Automated Vulnerability Detection and Remediation Method for Software Security," Sustainability, vol. 10, no. 5, p. 1652, 2018.
13. B. Chernis and R. Verma, "Machine Learning Methods for Software Vulnerability Detection," Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics, 2018.
14. S. Chakraborty, R. Krishna, Y. Ding, and B. Ray, "Deep Learning-based Vulnerability Detection: Are We There Yet," IEEE Transactions on Software Engineering, pp. 1–1, 2021.
15. Z. Li, D. Zou, S. Xu, H. Jin, H. Qi, and J. Hu, "VulPecker," Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016.
16. (NVD2019), "National vulnerability database." [Online]. Available: https://nvd.nist.gov/
17. (SARD2019), "Software assurance reference dataset." [Online]. Available: https://samate.nist.gov/SRD/index.php
18. T. Mikolov, M. Karafiát, L. Burget, J.Cernock`y, and S. Khudanpur, "Recurrent neural network-based language model," in Eleventh Annual Conference of the International Speech Communication Association, p. 1045.
19. "5 Natural Language Processing Techniques for Extracting ..." [Online]. Available:https://blog.aureusanalytics.com/blog/5natural-language-processing-techniques-forextracting-information. [Accessed: 04-Jan2022].
20. "BERT (Language model)," Wikipedia, 10-Oct-2019. [Online]. Available: https://en.wikipedia.org/wiki/BERT_(Langu age_model). [Accessed: 04-Jan-2022].
21. "Simple recurrent neural networks" Wikipedia, 08-July-2019. [Online]. Available:https://keras.io/guides/working_with_rnns/
22. "Long Short-Term Memory" Wikipedia, 07-July-2021 [Online]. Available:https://machinelearningmastery.com/gentleintroduction-long-short-term-memorynetworks-experts/
23. "Bidirectional recurrent neural networks" Wikipedia, 01-Dec-2021
24. [Online]. Available: https://en.wikipedia.org/wiki/Bidirectional_r HYPERLINK "https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks"ecurrent_neural_networks
25. "Gated recurrent unit" Wikipedia, 24-Nov-2021 [Online]. Available:"https://en.wikipedia.org/wiki/Gated_recurrent_unit"nt_unit
26. Zagane M, Abdi M K, Alenezi M. Deep Learning for Software Vulnerabilities Detection Using Code Metrics [J]. IEEE Access, 2020, 8: 74562-74570.
27. R. Coulter, Q.-L. Han, L. Pan, J. Zhang, and Y. Xiang. " Code Analysis for Intelligent Cyber Systems: A Data-Driven Approach." In: Information Sciences 524 (2020), pp. 46-58.
28. G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang. " Software Vulnerability Detection Using Deep Neural Networks: A Survey." In: Proceedings of the IEEE (2020), pp. 1-24.
29. MITRE. Common Vulnerabilities and Exposures (CVE). URL: https: //cve.mitre.org/ (visited on May 28, 2021).
30. Aota M, Kanehara H, Kubo M, Murata N, Sun B, Takahashi T (2020) Automation of vulnerability classification from its description using machine learning. 2020 IEEE Symposium on Computers and Communications (ISCC), pp 1–7
31. Bhuiyan FA, Sharif MB, Rahman A (2021) Security bug report usage for software vulnerability research: a systematic mapping study. IEEE Access 9:28471–28495
32. Siewruk G, Mazurczyk W (2021) Context-aware software vulnerability classification using machine learning. IEEE Access
33. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609
34. Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-qaness MA, Gandomi AH (2021) Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput Ind Eng. https://doi.org/10.1016/j.cie.2021.107250
35. Dam HK, Tran T, Pham T, Ng SW, Grundy J, Ghose A (2021) Automatic feature learning for predicting vulnerable software components. IEEE Trans Softw Eng 47(1):67–85. https://doi.org/10.1109/TSE.2018.2881961
36. Alnafessah A, Casale G (2020) Artificial neural networks based techniques for anomaly detection in Apache Spark. Cluster Comput. https://doi.org/10.1007/s10586-019-02998-y
37. Sherstinsky A (2020) Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Physica D 404:132306
38. W. Wang, Y. Li, X. Wang, J. Liu, X. Zhang, "Detecting Android Malicious Apps and Categorizing Benign Apps with Ensemble of Classifiers,"
39. Feist, J., Greico, G., Groce, A.: Slither: a static analysis framework for smart contracts. Paper presented at the WETSEB 2019: 2nd international workshop on emerging trends in software engineering for blockchain (2019)
40. X. Ban, S. Liu, C. Chen, and C. Chua, "A performance evaluation of deep-learned features for software vulnerability detection," Concurrency and Computation: Practice and Experience, vol. 31, no. 19, p. e5103, 2019.
41. Y. Wang, Z. Wu, Q. Wei et al., "NeuFuzz: efficient fuzzing with a deep neural network," IEEE Access, vol. 7, pp. 36340–36352, 2019.
42. M. Vasic, A. Kanade, P. Maniatis, et al., "Neural program repair by jointly learning to localize and repair," 2019, https://arxiv.org/abs/1904.01720.
43. M. White, M. Tufano, M. Martínez, M. Monperrus, and D. Poshyvanyk, "Sorting and transforming program repair ingredients via deep learning code similarities," in Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 479–490, IEEE, Hangzhou, China, February 2019.
44. L. Qiao and Y. Wang, "Effort-aware and just-in-time defect prediction with a neural network," PLoS One, vol. 14, no. 2, Article ID e0211359, 2019.
45. X. Chen et al., "Android HIV: A study of repackaging malware for evading machine-learning detection," IEEE Trans. Inf. Forensics Security, vol. 15, pp. 987–1001, Jul. 2020.