European Journal of Computer Science and Information Technology
Vol.10, No.1, pp.23-37, 2022
Print ISSN: 2054-0957 (Print),
Online ISSN: 2054-0965 (Online)

A SYSTEMATIC LITERATURE REVIEW OF SOFTWARE VULNERABILITY DETECTION

Adanma Cecilia Eberendu[1], Valentine Ikechukwu Udegbe[1], Edmond Onwubiko Ezennorom[1], Anita Chinonso Ibegbulam[1], Titus Ifeanyi Chinebu[2]
[1]Department of Computer Science, Madonna University, Nigeria, Elele, Rivers State, Nigeria
[2]Department of Physical Sciences, Federal College of Dental Technology and Therapy, P. M. B. 01473 Trans Ekulu Enugu, Nigeria.

Citation: Adanma Cecilia Eberendu, Valentine Ikechukwu Udegbe, Edmond Onwubiko Ezennorom, Anita Chinonso Ibegbulam, Titus Ifeanyi Chinebu (2022) A Systematic Literature Review of Software Vulnerability Detection, *European Journal of Computer Science and Information Technology*, Vol.10, No.1, pp.23-37

ABSTRACT: *This study provided a systematic literature review of software vulnerability detection (SVD) by searching ACM and IEEE databases for related literatures. Using the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) flowchart, a total of 55 studies published in the selected journals and conference proceeding of IEEE and ACM from 2015 to 2021 were reviewed. The objective is to identify, select and critically evaluate research works carried out on software vulnerability detection. The selected articles were grouped into 7 categories across various vulnerability detection evaluation criteria such as neural network – 5 papers, machine learning – 11 papers, static and dynamic analysis – 8 papers, code clone – 3 papers, classification – 4 papers, models – 3 papers, and frameworks – 6 papers. There are 15 articles that could not fall into any of these 7 categories, thus, they were place in others row that used different criteria to implement vulnerability detection. The result showed that many researchers used machine learning strategy to detect vulnerability in software since large volume of data can be reviewed easily with machine learning. Although many systems have been developed for detecting software vulnerability, none is able to show the type of vulnerability detected.*

KEYWORDS: vulnerability, software vulnerability, vulnerability detection, software vulnerability detection

INTRODUCTION

Software Vulnerability Detection (SVD) consists of flaws, bugs, errors, faults, weakness, defects, malicious codes, or system errors which hackers used to alter the performance or normal behaviour of the system (Barabanov et al., 2018). As the number of software systems increases so also the number of vulnerabilities. Considering that most devices are visible to multiple users on the internet, it will not take time for any launched attack to result to unpredictable damages and cost.

According to Jurn et al (2018), software vulnerabilities involve coding errors in program that cause undesirable actions to occur in software. These coding errors can cause the system to crash, connectivity to fail, prevent users from login, upgrade user privileges, grant access to unauthorized user, and printing errors. All software is vulnerable especially operating systems, web browsers, word processors and spreadsheets, video and audio players, embedded systems, and firmware according to Ibrahim et al (2019). Amankwah et al (2017) said that software vulnerability can be local or remote. Local vulnerabilities occur where access is local while remote vulnerabilities executes code on a remote machine and maliciously send it to network traffic or files.

Software vulnerability detection, according to Su et al (2016) is a process of discovering code snippets that have weaknesses which attackers might use to gain unauthorized access into the system thereby compromising the software or the platform on which the system runs. Software vulnerability is a thing of concern for software development organizations and end users of the product. Already, report on software vulnerability is on the increase, as such focus has been shifted to software vulnerability detection, monitoring and prevention. Many organizations have taken time to report vulnerabilities in software annually. For instance, US-CERT (2020) showed that the number of vulnerabilities in 2018 (17,252) and 2019 (17,382) is lower than 18,362 vulnerabilities reported in the National Vulnerability Database (NVD) in 2020. It was also noted that most companies encountered enormous losses due to vulnerability in software. For instance, more than 500,000 passwords were stolen from Zoom in April 2020 (Winder, 2020). In the same 2020, Leswing (2020) reported that twitter suffered a scam that swindled 121,000 US dollars in 400 Bitcoin transaction. Already, over 12,500 vulnerabilities were reported during the second quarter of 2021 (Greig, 2021).

Buffer flow is one of the causes of software vulnerability in which a program receives excess data that corrupts memory space and alter other data causing the program to flag errors or behave abnormally (Jurn et al 2018). This might give a hacker full control of the computer system (Jang & Choi, 2018). Also, Gupta and Gupta (2017) saw cross-site scripting as error that causes vulnerability by injecting malicious scripts into the websites or web application of unsuspected user with the intention of executing it on the user's system. The execution of the software is triggered from the hacker's browser. SQL injection occurs when an attacker gains unauthorized access with the queries send by an application to the database so that he (an attacker) can be able to create, insert, modify or delete records in the database. (Alwan & Younis, 2017). Braz et al 2021 discovered that unvalidated input also triggers vulnerability in software system once a program receives input from untrusted sources that causes the program to misbehave or corrupt the entire system. IP fragmentation attack instigates vulnerability in systems too (Leite & Albuquerque, 2018). The packets are fragmented in such a way that they are impossible to reassembly giving attacker loophole to carry out their nefarious attacks. Race condition vulnerability occurs when multiple users access a software resource concurrently. Farah et al, (2017) discovered that an attacker can take advantage of the gap between the Time-Of-Check and Time-Of-Use to gain access to the system, thus making it vulnerable.

Typically, the major purpose of an attacker is to gain access to a system and take control of its valuable information for personal benefit (Jang & Choi, 2018). In the work of Barabanov et al. (2016), they discovered that the development of new hacking techniques by hackers has resulted to increased number of vulnerabilities detection. It will be worthwhile for software developers and the users to be aware of how to detect and prevent vulnerabilities in software systems. Many research works have been carried out in this regard but none has comprehensively carried out a systematic literature review (SLR) of software vulnerability detection. This work tries to bridge this gap by systematically reviewing the research works based on the following research questions:

1. What is the number of research studies that deal with software vulnerability detection?
2. What are the contents the authors addressed in the study of software vulnerability detection?
3. What is the trend of software vulnerability detection from 2015 to 2021?
4. What are the main approaches used for detecting software vulnerability?

METHODOLOGY

Search strategy

For this systematic search, we developed a search strategy to identify relevant literature and this was tailored to two databases: IEEE and ACM and also a manual search in google search engine. This search was carried out on April 11, 2021. The search term used are “software vulnerability” and “software vulnerability detection”. All search concentrated on papers published between 2015 and 2021, giving a 5 years span and this included peer-review papers in journal articles and conference proceedings published in English. Only the AND operator was used during the search because it will generate the required result. Example of the keyword used in the search is: “software AND vulnerability AND detection” and also “software AND vulnerability”. OR operator was avoided because it will generate large result which might not be necessary.

Selection criteria

The selection criteria were based on the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) statement (Moher et al 2009), which was used for the literature screening to form the basis for the review and the search mainly focus on the mapping existing literatures on software vulnerability in the field of Computer Science, Software Engineering, Information Systems and Information Technology. The search then narrowed down to software development and security spanning articles from 2015 to 2021. All articles before 2015 were excluded from the search and the interest is on those published in English language and those in other languages were also excluded. A total of 174 papers were considered and 119 of them were excluded while 55 articles were extracted using this PRISMA method. Exclusion involved 93 research studies due to duplicate appearance, while 14 were screened out due to irrelevant title and abstract. Ongoing through the full text, it was also discovered that 12 of the remaining studies are illegible due to the content.

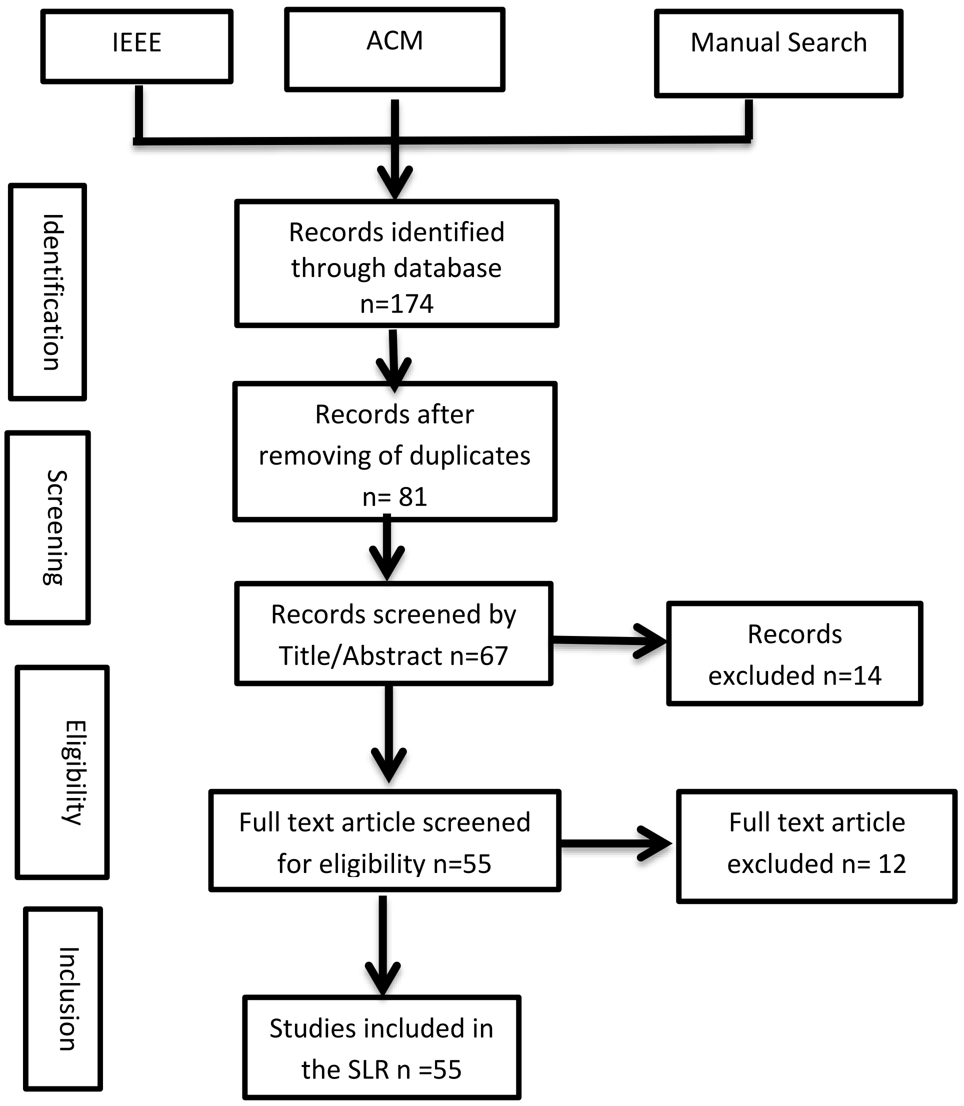
IEEE
ACM
Manual Search
Identification
Records identified through database n=174
Screening
Records after removing of duplicates n= 81
Records screened by Title/Abstract n=67
Records excluded n=14
Eligibility
Full text article screened for eligibility n=55
Full text article excluded n= 12
Inclusion
Studies included in the SLR n =55

Fig 1: Overview of Systematic review using PRISMA flow chart (Moher et al,2009)

Quality Assessment

To maintain the quality of the review, all duplicates were checked thoroughly and removed. Abstract were deeply crosschecked for the analysis and purification of the articles to ensure the quality and relevance of the academic literatures included in the review process. A careful evaluation of each research paper was carried out at a later stage. The next exclusion criterium was to limit the papers in the review process for those published in English language only. After the filtration of the duplicate records, 119 articles were removed from the study and 55 articles were selected after assessing each article on the aforementioned inclusion and exclusion criteria. The number of selected research papers for analysis and their year of publication is graphically depicted in figure 2 below.

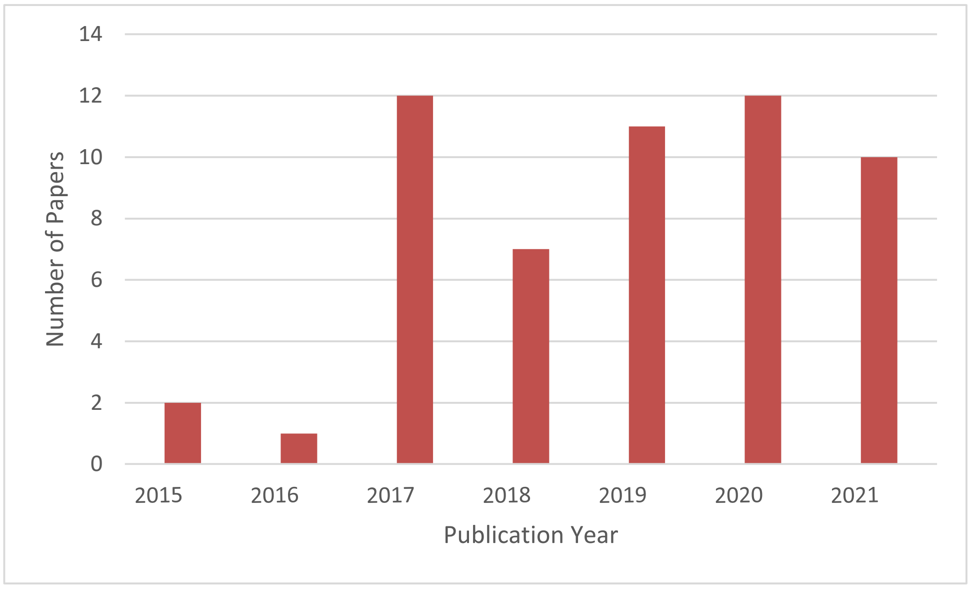

Fig 2: Number of selected papers and year of publication

It worth noting that the research papers selected are those published in renowned databases and discovered that the authors have used one of the methods, techniques, or tools to prove the result of their research.

Data Extraction

In the data extraction phase, 55 articles were selected and the characteristics for the selection are as follows:

1. Articles must be original papers, peer-reviewed papers or conference papers, but published reports and case studies were excluded.
2. The article must be in English language and from the field Computer Science, Software Engineering, Information Systems, or Information Technology.
3. Extracted articles were published between 2015 and 2021 from the IEEE and ACM databases only

The extraction of data is developed to solve the research questions based on the research type, research contents, and approaches for software vulnerability detection. The synthesis of the data is carried as tabulated in table 1.

Table 1: Pattern for Data extraction and synthesis

Data/Article	Description	Objective
Source	IEEE and ACM	Document
Type	Journal or Conference Proceedings	
Research Type	Experiment, Development, Evaluation, Proposal, and Validation	Document
Content addressed in Vulnerability Detections	Modelling, design	RQ2
Trend of Vulnerability Detection	Metrics, tools, and algorithms	RQ3
Approaches for vulnerability detection	Neural network, machine learning, code clone, static & dynamic analysis, model & Framework, etc.	RQ4
Taxonomy of Software vulnerability	Characteristics, causes	Extracts

Analysis and Results

The analysis and results section explain the literature evolution and the contents analysis. The literature evolution discussed the journals published between 2015 and 2021 while the content analysis focused on the approaches as categorized in table 2 below.

Table 2: Content analysis based on authors and approaches for detecting vulnerability in software

S/no	Category	References	Number
1	Neural Network	Majumder et al 2019; Tang et al 2020; Zheng et al 2019; Mao et al, 2020; Liu et al 2020; Damien et al 2019	5
2	Code Clone	Hu et al 2017; Liu et al 2019a; Kim et al 2017	3
3	Static and Dynamic analysis	Ruggahakotuwa et al 2019; Pereira et al 2020; Spoto et al 2019; Russell et al 2018; Ibrahim et al 2019; Nong and Cai 2020; Chernis and Verma 2018; Li et al 2018	8
4	Machine Learning	Liu et al 2020; Liu et al, 2019b; Zou et al 2021; Zeng et al 2020; Lin et al 2018; Zheng et al 2021; Nguyen et al 2019; Ji et al 2018; Russell et al 2018; Li and Shao 2019; Kumar et al 2019; Zagane et al 2020; Liu et al 2018; Wu et al 2021; Grieco and Dinaburg (2018); Lin et al. (2019)	16
5	Models & Frameworks	Wang et al 2015; Bagri & Gupta, 2019; Min et al 2019; Wang et al 2020; Zarakovitis et al 2021; Li et al January, 2017; Li et al August 2017; Li et al., September 2017; Jinan et al 2017.	9
6	Uncategorized	Braz et al 2021; Ruohonen et al 2016; Qi 2021; Obaida et al 2017; Trabelsi et al 2015; Kapur 2017; Zhang et al 2020; Kostromitin et al 2020; Li et al 2021; Zhang et al 2017; Wibowo et al 2017; Sultana and Williams 2017; Ziems and Wu 2021; Paradis et al, 2018; Han et al 2019; Choi et al 2020	16

Forty-three papers were published in conference, symposium, or workshop proceedings while only twelve were published in monthly or quarterly journal. The research works are distributed into seven categories to aid the analysis and achieve the objectives. The references are included to help future research works that may carry out further investigation. Thus, considering the forms of software vulnerability detection, a mind map of SVD is depicted in figure 1. The subsections showed the taxonomy of the available software vulnerability detection approaches based on the analysis of the existing literature.

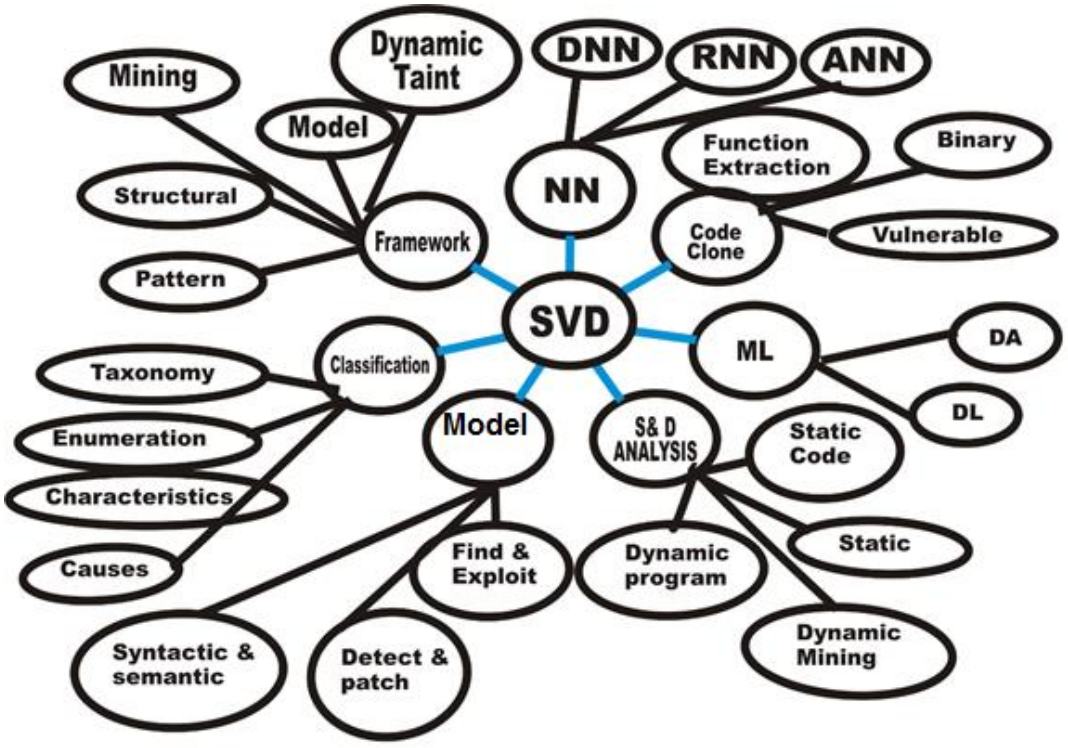

Fig. 1: Mind Map Of Different Ways Of Detecting Vulnerabilities In Software

The automated software vulnerability detection using neural networks was addressed by five researchers. They proposed artificial neural network (Majumder et al 2019; Tang et al 2020), recurrent neural network (Zheng et al 2019; Mao et al, 2020), or deep neural network (Liu et al 2020). Damien et al (2019) designed a tool to automatically perform application code mutations that imitate the activity of malicious code in an application. Three researchers used binary code clone to detect vulnerability in software. Hu et al (2017) used a semantics-based approach while Liu et al 2019a used deep learning-based approach and Kim et al (2017) used vulnerable code clone approach. Ruggahakotuwa et al (2019) and seven other researchers used static and dynamic analysis to perform vulnerability detection in software. Pereira (2020) combined static code analysis methods with software metrics to enhance the capability of detecting vulnerability in software. Spoto et al (2019) used static analyzers for cybersecurity to identify overt flows of infected data in Java code. Russell et al (2018) leveraged the capability of C and C++ open-source code but Ibrahim et al (2019) used static analysis vulnerability scanner to analyze open source PHP applications in GitHub. Nong and Cai (2020) based their arguments on (static and/or dynamic) for memory vulnerability detectors. Chernis and Verma (2018) used static analysis to trap significant percentage of flaws from functions in C source code. Li et al (2018) designed integrated testing framework for static analysis methods and dynamic mining tool.

The potentials of machine learning and deep learning technologies are the most popular technique used to develop systems for SVD. Liu et al (2020) used deep learning technology to design a system for Cross Domain Software Vulnerability Discovery, while Liu et al, (2019a) proposed deep learning approach to design automatic vulnerability detection in binary code. Zou et al (2021) also developed a deep learning-based system for multiclass vulnerability detection to identify vulnerabilities in code. Zeng et al (2020) in their study reviewed research works that employed deep learning to detect software vulnerability. Those that used machine learning features are Lin et al (2018) who addressed issues when high quality training data is in deficit. Zheng et al (2021) evaluated vulnerability detection on source codes while Nguyen et al (2019) used machine learning to design SCDAN (Semi-supervised Code Domain Adaptation Network) that will predict the vulnerability detection performance. Ji et al (2018) investigated the capability of machine learning but Russell et al (2018) used machine learning and the features of C and C++ to develop vulnerability detection system. Li and Shao (2019) made use of the features of machine learning to analyze the problems and challenges of software vulnerability detection. Kumar et al (2019) used machine learning algorithms to develop a system for detecting software vulnerability. Zagane et al (2020) linked the ideas of using deep learning to machine learning features and discussed a deep-learning-based approach that used code metrics for detecting software vulnerabilities. Liu et al (2018) and Wu et al (2021) developed automated methods for detecting vulnerabilities in software while Grieco and Dinaburg (2018) used integrated tools to develop a system that is capable of detecting vulnerabilities in source code. Lin et al. (2019) used machine learning approach that used cross-domain data source to develop a framework to improve vulnerability detection performance.

Six of the research works developed framework to automate vulnerability detection in software product (Wang et al 2015; Bagri & Gupta, 2019; Min et al 2019; Li et al 2017; Wang et al 2020; Zarakovitis et al 2021). Min et al (2019) designed an Android software vulnerability mining framework based on dynamic taint analysis technology. Li et al (January, 2017) based their studies on software vulnerability classification, causes, and characteristics. The work of Li et al (August 2017) proposed using data mining techniques to develop vulnerability detection model and in their further study, they (Li et al., September 2017) used extended chemical abstract to design a new vulnerability model. Jinan et al (2017) reviewed research works carried out on vulnerability enumeration, taxonomy, models, and detection methods.

The other 15 researchers that dealt on different issues of vulnerability detection are Braz et al (2021) that investigated the main causes of Improper Input Validation (IIV) and its late detection, Ruohonen et al (2016) examined time delays through network analysis between software vulnerability disclosure notifications and acknowledgments, and Qi (2021) implemented the adaptive vulnerability information detection protocol. Obaida et al (2017) proposed Secure Sensitive Data (SSD) Eclipse IDE plug-in to bridge the gap in sensitive data leaks while Trabelsi et al 2015 investigated the Twitter feed. Kapur (2017) used dis-adoption process to model vulnerability patch and Zhang et al (2020) developed integrated testing and evaluation system (iTES) to evaluate the techniques of software vulnerability detection. Kostromitin et al (2020) discovered that hardware implementation of undocumented instructions causes unstable functioning of critical objects. Security threats are prominent in software reused libraries due to its propagation throughout development phases as such Li et al (2021) developed PDGraph to expose the risk. Zhang et al (2017) utilized semi-symbolic Fuzz testing method to uncover hard-to-reach vulnerabilities in programs. Wibowo et al (2017) investigated the Architectural Vulnerability Factor (AVF) of all major in-core memory structures of an out-of-order superscalar processor while Sultana and Williams (2017) used micro patterns detect vulnerability in software. Ziems and Wu (2021) and Paradis et al, (2018) modelled test as a source code and used it for software vulnerability detection in natural language processing (NLP), Han et al (2019) proposed a static detection model, while Choi et al (2020) developed Cyber-Physical Inconsistency to target vulnerability detection in Robotic Vehicles (RVs).

CONCLUSION

Systematic review of works done on software vulnerability detection used different techniques such as machine learning and deep learning, neural network, binary code clone, static and dynamic analysis, methods and framework analysis to detect vulnerability in software products. Based on this systematic literature review, machine learning and deep learning approaches were mostly used to detect vulnerability in software because every domain is driven nowadays by machine learning application and many researchers are venturing into it. Static and dynamic analysis was also used extensively to detect vulnerability in software. Some developed techniques for detecting vulnerability were unable to show the type of vulnerability detected and this is an issue for discussion in subsequent study. The result of this systematic literature review was evaluated using PRISMA guidelines (Page et al, 2021).

References

Alwan, Z. S., and Younis, M. F. (2017). Detection and prevention of SQL injection attack: A survey. *International Journal of Computer Science and Mobile Computing*, *6*(8), 5-17.

Amankwah, R., Kudjo, P.K. and Antwi, S. Y. (2017) Evaluation of Software Vulnerability Detection Methods and Tools: A Review. *International Journal of Computer Applications 169(8),* Pp 22-27.

Bagri, B., and Gupta, G. (2019, October). Automation Framework for Software Vulnerability Exploitability Assessment. In *2019 Global Conference for Advancement in Technology (GCAT)* (pp. 1-7). IEEE.

Barabanov, A. V, Markov, A. S. and Tsirlov, V. L. (2018) Statistics of software vulnerability detection in certification testing. International Conference Information Technologies in Business and Industry IOP Conf. Series: Journal of Physics: Conf. Series **1015** 042033

Barabanov, A. V., Markov, A. S. and Tsirlov, V. L. (2016) Methodological framework for analysis and synthesis of a set of secure software development controls *Journal of Theoretical and Applied Information Technology 88*(1), **77**-88

Braz, L., Fregnan, E., Çalikli, G., and Bacchelli, A. (2021, May). Why Don't Developers Detect Improper Input Validation?'; DROP TABLE Papers; --. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 499-511). IEEE.

Chernis, B., and Verma, R. (2018, March). Machine learning methods for software vulnerability detection. In *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics* (pp. 31-39).

Choi, H., Kate, S., Aafer, Y., Zhang, X., and Xu, D. (2020, October). Cyber-physical inconsistency vulnerability identification for safety checks in robotic vehicles. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (pp. 263-278).

Damien, A., Feyt, N., Nicomette, V., Alata, E., and Kaâniche, M. (2019, September). Attack injection into avionic systems through application code mutation. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)* (pp. 1-8). IEEE.

Farah, T., Shelim, R., Zaman, M., Hassan, M. M., and Alam, D. (2017). Study of race condition: A privilege escalation vulnerability. In *WMSCI 2017-21st World Multi-Conference Syst. Cybern. Informatics, Proc* (Vol. 2, pp. 100-105).

Grieco, G., and Dinaburg, A. (2018, January). Toward Smarter Vulnerability Discovery Using Machine Learning. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security* (pp. 48-56).

Greig, J. (2021). More than 12,500 vulnerabilities disclosed in first half of 2021: Risk Based Security. Retrieved on 30 august, 2021 from https://www.zdnet.com/article/

Gupta, S., and Gupta, B. B. (2017). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, *8*(1), 512-530.

Han, L., Zhou, M., Qian, Y., Fu, C., and Zou, D. (2019). An Optimized Static Propositional Function Model to Detect Software Vulnerability. *IEEE Access*, *7*, 143499-143510.

Hu, Y., Zhang, Y., Li, J., and Gu, D. (2017, May). Binary code clone detection across architectures and compiling configurations. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)* (pp. 88-98). IEEE.

Ibrahim, A., El-Ramly, M., and Badr, A. (2019, November). Beware of the Vulnerability! How Vulnerable are GitHub's Most Popular PHP Applications? In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)* (pp. 1-7). IEEE.

Jang, Y. S., and Choi, J. Y. (2018). Automatic prevention of buffer overflow vulnerability using candidate code generation. *IEICE TRANSACTIONS on Information and Systems*, *101*(12), 3005-3018.

Ji, T., Wu, Y., Wang, C., Zhang, X., and Wang, Z. (2018, June). The coming era of alpha-hacking: A survey of automatic software vulnerability detection, exploitation and patching techniques. In *2018 IEEE third international conference on data science in cyberspace (DSC)* (pp. 53-60). IEEE.

Jinan, S., Kefeng, P., Xuefeng, C., and Junfu, Z. (2017, May). Security Patterns from Intelligent Data: A Map of Software Vulnerability Analysis. In *2017 IEEE 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference* *on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)* (pp. 18-25). IEEE.

Jurn, J., Kim T. and Kim H. (2018) An Automated Vulnerability Detection and Remediation Method for Software Security *Sustainability 1652*(10), 1-12

Kapur, P. K. (2017, December). User based fault detection, vulnerability discovery and patching: An interdisciplinary research. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)* (pp. 27-33). IEEE.

Kim, S., Woo, S., Lee, H., and Oh, H. (2017, May). Vuddy: A scalable approach for vulnerable code clone discovery. In *2017 IEEE Symposium on Security and Privacy (SP)* (pp. 595-614). IEEE.

Kostromitin, K. I., Dokuchaev, B. N., and Kozlov, D. A. (2020, September). Analysis of the Most Common Software and Hardware Vulnerabilities in Microprocessor Systems. In *2020 International Russian Automation Conference (RusAutoCon)* (pp. 1031-1036). IEEE.

Kumar, A., and Lim, T. J. (2019, April). EDIMA: early detection of IoT malware network activity using machine learning techniques. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)* (pp. 289-294). IEEE.

Leite, G. S., and Albuquerque, A. B. (2018, September). An Approach for Reduce Vulnerabilities in Web Information Systems. In *Proceedings of the Computational Methods in Systems and Software* (pp. 86-99). Springer, Cham.

Leswing, K. (2020). Twitter hackers who targeted Elon Musk and others received $121,000 in bitcoin: analysis shows. Retrieved on 30 August, 2021 from https://www.cnbc.com/2020/07/16/

Li, J., Chen, J., Huang, M., Zhou, M., Xie, W., Zeng, Z., ... and Zhang, Z. (2018). An integration testing framework and evaluation metric for vulnerability mining methods. *China Communications*, *15*(2), 190-208.

Li, Q., Song, J., Tan, D., Wang, H., and Liu, J. (2021, June). PDGraph: A Large-Scale Empirical Study on Project Dependency of Security Vulnerabilities. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 161-173). IEEE.

Li, X., Chang, X., Board, J. A., and Trivedi, K. S. (2017, January). A novel approach for software vulnerability classification. In *2017 Annual Reliability and Maintainability Symposium (RAMS)* (pp. 1-7). IEEE.

Li, X., Chen, J., Lin, Z., Zhang, L., Wang, Z., Zhou, M., and Xie, W. (2017, August). A mining approach to obtain the software vulnerability characteristics. In *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)* (pp. 296-301). IEEE.

Li, X., Chen, J., Lin, Z., Zhang, L., Wang, Z., Zhou, M., and Xie, W. (2017, September). A new method to construct the software vulnerability model. In *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA)* (pp. 225-229). IEEE.

Li, Z., and Shao, Y. (2019, February). A survey of feature selection for vulnerability prediction using feature-based machine learning. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing* (pp. 36-42).

Lin, G., Zhang, J., Luo, W., Pan, L., De Vel, O., Montague, P., and Xiang, Y. (2019). Software vulnerability discovery via learning multi-domain knowledge bases. *IEEE Transactions on Dependable and Secure Computing*.

Lin, G., Zhang, J., Luo, W., Pan, L., Xiang, Y., De Vel, O., and Montague, P. (2018). Cross-project transfer representation learning for vulnerable function discovery. *IEEE Transactions on Industrial Informatics*, *14*(7), 3289-3297.

Liu, D., Wang, J., Rong, Z., Mi, X., Gai, F., Tang, Y., and Wang, B. (2018, August). Pangr: A Behavior-Based Automatic Vulnerability Detection and Exploitation Framework. In *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)* (pp. 705-712). IEEE.

Liu, S., Dibaei, M., Tai, Y., Chen, C., Zhang, J., and Xiang, Y. (2019). Cyber vulnerability intelligence for Internet of Things binary. *IEEE Transactions on Industrial Informatics*, *16*(3), 2154-2163.

Liu, S., Lin, G., Han, Q. L., Wen, S., Zhang, J., and Xiang, Y. (2019). DeepBalance: Deep-learning and fuzzy oversampling for vulnerability detection. *IEEE Transactions on Fuzzy Systems*, *28*(7), 1329-1343.

Liu, S., Lin, G., Qu, L., Zhang, J., De Vel, O., Montague, P., and Xiang, Y. (2020). CD-VulD: Cross-domain vulnerability discovery based on deep domain adaptation. *IEEE Transactions on Dependable and Secure Computing*.

Majumder, R., Som, S., and Gupta, R. (2017, December). Vulnerability prediction through self-learning model. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)* (pp. 400-402). IEEE.

Mao, Y., Li, Y., Sun, J., and Chen, Y. (2020, December). Explainable Software vulnerability detection based on Attention-based Bidirectional Recurrent Neural Networks. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 4651-4656). IEEE.

Min, Z., Haimin, Y., Ping, C., and Zhengxing, Y. (2019, March). Android software vulnerability mining framework based on dynamic taint analysis technology. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (pp. 2112-2115). IEEE.

Moher, D., Liberati, A., Tetzlaff, J., Altman, D. G., and Prisma Group. (2009). Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *PLoS medicine*, *6*(7), e1000097.

Nguyen, V., Le, T., Le, T., Nguyen, K., DeVel, O., Montague, P., ... and Phung, D. (2019, July). Deep domain adaptation for vulnerable code function identification. In *2019 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.

Nong, Y., and Cai, H. (2020, February). A preliminary study on open-source memory vulnerability detectors. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 557-561). IEEE.

Obaida, M. A., Nelson, E., Ee, R. V., Jahan, I., and Sajal, S. Z. (2017, May). Interactive sensitive data exposure detection through static analysis. In *2017 IEEE International Conference on Electro Information Technology (EIT)* (pp. 270-275). IEEE.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., ... and Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *Bmj*, *372*.

Paradis, C., Kazman, R., and Wang, P. (2018, December). Indexing text related to software vulnerabilities in noisy communities through topic modelling. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 763-768). IEEE.

Pereira, J. D. A. (2020, October). Techniques and Tools for Advanced Software Vulnerability Detection. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 123-126). IEEE.

Qi, X. I. O. N. G. (2021, March). Generation technology of multimedia application software defect data for group users. In *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)* (pp. 18-23). IEEE.

Ruggahakotuwa, L., Rupasinghe, L., and Abeygunawardhana, P. (2019, December). Code Vulnerability Identification and Code Improvement using Advanced Machine Learning. In *2019 International Conference on Advancements in Computing (ICAC)* (pp. 186-191). IEEE.

Ruohonen, J., Holvitie, J., Hyrynsalmi, S., and Leppänen, V. (2016, November). Exploring the clustering of software vulnerability disclosure notifications across software vendors. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)* (pp. 1-8). IEEE.

Russell, R., Kim, L., Hamilton, L., Lazovich, T., Harer, J., Ozdemir, O., ... and McConley, M. (2018, December). 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.

Spoto, F., Burato, E., Ernst, M. D., Ferrara, P., Lovato, A., Macedonio, D., and Spiridon, C. (2019). Static identification of injection attacks in Java. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *41*(3), 1-58.

Sultana, K. Z., & Williams, B. J. (2017, November). Evaluating micro patterns and software metrics in vulnerability prediction. In *2017 6th International Workshop on Software Mining (SoftwareMining)* (pp. 40-47). IEEE.

Tang, G., Meng, L., Wang, H., Ren, S., Wang, Q., Yang, L., and Cao, W. (2020, December). A comparative study of neural network techniques for automatic software vulnerability detection. In *2020 International Symposium on Theoretical Aspects of Software Engineering (TASE)* (pp. 1-8). IEEE.

Trabelsi, S., Plate, H., Abida, A., Ben Aoun, M.M., Zouaoui, A., Missaoui, C., Gharbi, S., and Ayari, A. (2015) Monitoring software vulnerabilities through social networks analysis. In proceeding of the 12th International Joint Conference on e-Business and Telecommunications (ICETE). *4* (pp. 236-242)

US-Cert (2020). Vulnerabilities Exploited in 2020. From https://www.cisa.gov/uscert/ncas/alerts/aa20-133a accessed on July 27,2021.

Wang, H., Ye, G., Tang, Z., Tan, S. H., Huang, S., Fang, D., ... and Wang, Z. (2020). Combining graph-based learning with automated data collection for code vulnerability detection. *IEEE Transactions on Information Forensics and Security*, *16*, 1943-1958.

Wang, X., Ma, H., Yang, K., and Liang, H. (2015, November). An Uneven Distributed System for Dynamic Taint Analysis Framework. In *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing* (pp. 237-240). IEEE.

Wibowo, B., Agrawal, A., and Tuck, J. (2017, October). Characterizing the impact of soft errors across microarchitectural structures and implications for predictability. In *2017 IEEE International Symposium on Workload Characterization (IISWC)* (pp. 250-260). IEEE.

Winder, D. (2020) Zoom Gets Stuffed: Here's How Hackers Got Hold Of 500,000 Passwords. Retrieved on August 30, 2021 from https://www.forbes.com/sites/daveywinder/2020/04/28/

Wu, Y., Lu, J., Zhang, Y., and Jin, S. (2021, January). Vulnerability Detection in C/C++ Source Code with Graph Representation Learning. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 1519-1524). IEEE.

Su, Y., Li, M., Tang, C., and Shen, R. (2016) An Overview of Software Vulnerability Detection. International *Journal of Computer Science and technology* 7(3)

Zagane, M., Abdi, M. K., and Alenezi, M. (2020). Deep learning for software vulnerabilities detection using code metrics. *IEEE Access*, *8*, 74562-74570.

Zarakovitis, C., Klonidis, D., Salazar, Z., Prudnikova, A., Bozorgchenani, A., Ni, Q., ... and Mallouli, W. (2021, August). SANCUS: Multi-layers Vulnerability Management Framework for Cloud-native 5G networks. In *The 16th International Conference on Availability, Reliability and Security* (pp. 1-10).

Zeng, P., Lin, G., Pan, L., Tai, Y., and Zhang, J. (2020). Software Vulnerability Analysis and Discovery using Deep Learning Techniques: A Survey. *IEEE Access*.

Zhang, B., Ye, J., Feng, C., and Tang, C. (2017, December). S2F: discover hard-to-reach vulnerabilities by semi-symbolic fuzz testing. In *2017 13th International Conference on Computational Intelligence and Security (CIS)* (pp. 548-552). IEEE.

Zhang, C., Chen, J., Cai, S., Liu, B., Wu, Y., and Geng, Y. (2020, December). iTES: Integrated Testing and Evaluation System for Software Vulnerability Detection Methods. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 1455-1460). IEEE.

Zheng, J., Pang, J., Zhang, X., Zhou, X., Li, M., and Wang, J. (2019, December). Recurrent Neural Network Based Binary Code Vulnerability Detection. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence* (pp. 160-165).

Zheng, W., Semasaba, A. O. A., Wu, X., Agyemang, S. A., Liu, T., and Ge, Y. (2021, March). Representation vs. Model: What Matters Most for Source Code Vulnerability Detection. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 647-653). IEEE.

Ziems, N., and Wu, S. (2021). Security Vulnerability Detection Using Deep Learning Natural Language Processing. *arXiv preprint arXiv:2105.02388*.

Zou, D., Wang, S., Xu, S., Li, Z., and Jin, H. (2019). μVulDeePecker: A deep learning-based system for multiclass vulnerability detection. *IEEE Transactions on Dependable and Secure Computing*. 18(5) (pp.2224-2236)