A SYSTEMATIC LITERATURE REVIEW OF SOFTWARE VULNERABILITY DETECTION

Adanma Cecilia Eberendu¹, Valentine Ikechukwu Udegbe¹, Edmond Onwubiko Ezennorom¹, Anita Chinonso Ibegbulam¹, Titus Ifeanyi Chinebu²

¹Department of Computer Science, Madonna University, Nigeria, Elele, Rivers State, Nigeria
²Department of Physical Sciences, Federal College of Dental Technology and Therapy, P. M. B. 01473 Trans Ekulu Enugu, Nigeria.

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 placed 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.
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.
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

<table>
<thead>
<tr>
<th>Source</th>
<th>Type</th>
<th>Research Type</th>
<th>Content addressed in Vulnerability Detections</th>
<th>Trend of Vulnerability Detection</th>
<th>Approaches for vulnerability detection</th>
<th>Taxonomy of Software vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE and ACM</td>
<td>Journal or Conference Proceeding</td>
<td>Experiment, Development, Evaluation, Proposal, and Validation</td>
<td>Modelling, design</td>
<td>Metrics, tools, and algorithms</td>
<td>Neural network, machine learning, code clone, static &amp; dynamic analysis, model &amp; Framework, etc.</td>
<td>Characteristics, causes</td>
</tr>
<tr>
<td>Objective</td>
<td>Document</td>
<td></td>
<td></td>
<td>RQ2</td>
<td>RQ3</td>
<td>Extracts</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>S/no</th>
<th>Category</th>
<th>References</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Code Clone</td>
<td>Hu et al 2017; Liu et al 2019a; Kim et al 2017</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Static and Dynamic analysis</td>
<td>Ruggahakotuwla 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</td>
<td>8</td>
</tr>
</tbody>
</table>

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 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, Ruohonien 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

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


on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids) (pp. 18-25). IEEE.


@ECRTD-UK- https://www.eajournals.org/
https://doi.org/10.37745/ejcsit.2013