An Overview on Software Vulnerabilities

[1]Pawan Kumar, [2]Yudhvir Singh
[1]M.Tech (2nd year), [2]Professor and Dean
Computer Science and Engineering Department,
U.I.E.T MDU Rohtak.

Abstract : As emerging technology evolves, it affects the requirement of users and organization. Every day new software buzz is developing for the public and commercial use. As simple, attractive and smoothly the software is working, their codes are that much complex and with the complex cording the chance of having vulnerability in the software increases. If attackers find the vulnerability in the software, it can cause serious consequences. The purpose of this report is to provide an overview on software vulnerabilities. The report discusses the work of various researchers, and their techniques. The main focus of this report is 'Detection of software vulnerabilities'. The various techniques and methods of detecting software vulnerabilities and comparison of the tools used have argued in this report.

***IndexTerms* - Software vulnerabilities, detection techniques.**

I. INTRODUCTION

With the increase in modern technology requirement of software increases rapidly. Person and organization use software for own purposes. On the requirement of person and organization new software are developed every day. The software is developed by different programmers. Unintentional mistakes or undesired condition in code of the program makes the vulnerable software. Many scholars have given different definitions of the approach as Kuang et al. [1] Describe software vulnerability as the "fault that can be viciously used to harm the security of software systems". Krsul [2] defines it "a defect that allows an attacker to violate an explicit or implicit security policy to achieve some impact". In another study, Jimenez et al. [3] described software vulnerability as "a flaw, weakness or even an error in the system that can be exploited by an attacker in order to alter the normal behavior of the system". Schultz et al. [4] Described software vulnerability as ''a malicious defect, which facilitate an assailant to neglect security perspective". Moreover the OIS an organization of safety measures of the web defines security vulnerability as "a flaw within a software system that can cause it to work contrary to its documented design and could be exploited to cause the system to violate its documented security policy" [5]. As conclude from definition a software vulnerability is an imperfection in the evolution of software that can be exploited by an assailant with a specific end goal to get a few benefits in the information system . It implies the vulnerability opens the entry points in the system. Generally, an attacker's aim is to gain some benefits of the information system by controlling or alter the data or finding the profitable data from information system for its own advantage [6]. An Information system is a system that has network system, numerous classification of utilization oriented operating systems and operating system. Vulnerabilities in software causes huge negative effects on the information system.

Now days, information is the main and significant point to develop the economic society. Software is used excessively for developing the social image of an individual citizen and organizations. Information security has turned into the vital issue in the whole region of social and economic development. It affects the person's liberty, national security, public interest and economic development. Different sorts of internet crimes likes, cyberbulling and harassment, financial extortion, copyright violation, phishing and personal data hacking are rapidly increasing. It can root severe damage to national economic progress and financial situation [7]. So the awareness of software vulnerability is important for organizations and the general public. It's essential to all knowing detection and prevention technique of vulnerability.

In the content of European project SHIELDS, whose primary goal is to overcome any issue between security expert and software programmer and this manner decreases the rate of software vulnerabilities [8]. They present different method to detect software vulnerabilities and some of the familiar vulnerabilities. A minimum number of events, vulnerabilities are overworked due to inaccurate data enrolled by system users. The attackers can inject fault and malicious code in the system where the condition is undesired that give them access to run their own code applications.

There are a lot of international security organizations that tied up in Vulnerability research. NIST, Common Weakness Research and Open Web Application Security Project (OWASP) are some official institutions that administer vulnerabilities. "An integrated technology that needs analyzers has a collection of the detection technique to make common use and complementary advantages is known as Vulnerability analysis technology". Detection methods are divided into static, dynamic and Hybrid detection method.

II. RELATIVE WORK:-

Past many years, researchers work on detection and prevention method of software vulnerabilities. This paper contains some of the research work of different researchers.

In 1987, 10th NCSC journal has an article "What do you feed a Trojan Horse? Techniques to solve Hacking problem" [9], this article addresses how to detect an attack and track the intruder. In this article monitoring software and equipment can be programmed to alarm whenever intruder access the system. To track the source of code or the location of the intruder, many organizations coordinate together. This is the limitation of the experiment, i.e. the problem of coordination among different organizations. It is time taking and expensive method. An advantage is that an intruder is trapped and legal action is taken place in him.

In 2008, 2nd International Conference on Emerging Security Information, Systems and Technologies has a paper "Vulnerability dependencies in Antivirus Software". This paper's main objective is "to examine Antivirus (AV) software vulnerabilities and the risks they bring to the critical information infrastructure system" [10]. They utilize an automation of MATINE terminology for searching dependencies available and arises in an AV operating system. Previously, for discovery of vulnerabilities in network protocol this method is used effectively. AV software has great impact on security, so they used this approach to find vulnerabilities. It concludes that until the year 2006 main reason to raise AV vulnerability is to access the file-format. As compared to any other software vulnerabilities AV vulnerabilities are rarely reported and discussed. The AV main focus is on new malware and fusion of enterprises. Awareness about the drawbacks of AV software spread among the users. AV software is chosen on the bases of the system.

In 2009, 'World Congress on Software Engineering' has published a paper on "Fault Injection Technology for Software Vulnerability Testing Based on Xen" [11]. In their paper Zeng et.al conferred 'a Xen-based fault injection automation of operating system vulnerability analysis in regulation to frame a capable and general-purpose operating system analysis model, which implants malicious faults into a bilateral layer between software functions and their related environments'. Their method completes some features effectively like the backup, recovery and debugging. For setting breakpoint for fault injection the model examine and evaluate the program code statically, for disturbing the environment the fault is inserted at interface among the operating system and the related environment in dynamic testing. By this way they create the objectionable situation to prompt the software vulnerability mark.

In 2009, '33rd Annual IEEE International Computer Software and Applications Conference' has a paper on "GUI-Based Testing of Boundary Overflow Vulnerability". Their paper is focused on "input validation of graphical user interfaces" [12] [13]. They generate test case for SUC, decision tables are used as input validation. For handling error in SUC like boundary overflow an algorithm is introduced. The result of their approach shows that error boundary overflow is effectively found in SUC mechanism. Their approach automatically fixed the error they find during the process. Their tool is viewed as a novel extension tool for finding boundary overflow errors [14].

In 2010, '3rd International Conference on Software Testing, Verification, and Validation Workshops' has a paper on "Towards Security Vulnerability Detection by Source Code Model Checking" [15]. The main objective of their paper is 'program code miniature monitor technique to verify whether few security automation guidance are pursued and respectively to disclose analogous security vulnerabilities'. For example, SAP security protocol for programming linked to cross-site scripting attack and logging sensitive information are used. They present the working of BANDERA specification language discusses secure programming guidelines [16]. Java program code and Bandera checker both mingled together and check security programming ground rule are chased and interrelated vulnerabilities are present [17]. The limitation of this approach is that Bandera 0.3 is a automation model and resources discussed in the paper are less complex than real program.

In 2010, 'IIEEE Symposium on Computers and Communications (ISCC)' has a paper on "A Key-Agreement Protocol Based on the Stack-Overflow Software Vulnerability". Fatayer et.al. Talk about a key agreement protocol that show "a common stack-overflow attack, namely the return-to-libc attack, coupled with a common defense, namely the Address Space Layout Randomization (ASLR), together allow for constructing a key agreement protocol that allows two entities to agree on a shared key, whereas the shared key can then be used to encrypt further communication" [18]. That shared key can used as encryption for communication. For evaluating the performance and feasibility of a key agreement protocol they develop a prototype. The prototype result is based on the protocol implementation concepts that are proven. The result showed that generating protocol has no fixed lengths with less computation overhead and message, time overhead is direct in the key length. Due to low computation overhead, it is best for CPU-Constrained platforms. This protocol can be used for malicious purpose by attackers. Gain access to unauthorized system attackers prefers exploiting software vulnerabilities like stack, heap and format string overflow [19].

In 2013, '7th International Conference on Software Security and Reliability Companion' publish an article "Vulcloud: Scalable and Hybrid Vulnerability Detection in Cloud Computing" [20]. The purpose of this article is giving an overview of Vulcloud technique. The workload is a cloud computing technique that combines fuzzing, dynamic and static technique. In first

step object is statically analyzes and vulnerable items are reported. In a second step for fuzzing cases items are created semi automated and supervise under dynamic monitoring. Finally, the item is analyzed statically and determines they are vulnerable or not.

In 2013, 2nd ISCCCA journal has an article "Research on vulnerability detection for software based on Taint Analysis" [21]. The aim of their Paper is to discover the vulnerabilities and actualizes a model system for the Java Web program using Taint dependence analysis method. In their paper for actualizing the Java web program they propose the Taint dependency scrutiny algorithm and its associated graph. To accomplish the program vulnerability detection method they deliberated the imitation for the rate set of the infected string gadget in aspect of the finite state automata mingled with united attack pattern matching and the promising Taint Analysis methodology. The limitation of their approach is that an assailant can violate by added script tag inside the script code and then enclosed it. In algorithm bad() function doest no filter the data effectively. The advantage is that the model systems can recognize those programs which have reflection cross site scripting vulnerabilities for the user input data.

In 2013, 'JOURNAL OF SOFTWARE, ACADEMY PUBLISHER' publishes a paper "Vulnerability Discovery Technology and Its Applications" [22]. They classify the method of previous find vulnerabilities and discuss their weakness, strength and extend of each method. They discuss the vulnerability finding method and issue of detecting network vulnerabilities. There search is very systematic. They improve the current discovery techniques. Researchers keep trying and give low false alarm rate [23] [24].

In 2014, 'IEEE Symposium on Security and Privacy' publish an article "Modeling and Discovering Vulnerabilities with code property graphs" [25]. The purpose of this article gives a method using code property graph (CPG) to productively excerpt gigantic amount of program code for vulnerabilities. In this article to find the large amount of vulnerabilities from source code they build a CPG. That enables modeling patterns into graph traversals for common vulnerabilities. By this approach they present much previous vulnerability as a graph. In Linux kernel source code they find out 18 previously unknown vulnerabilities that are fixed by the vendor. Limitations of this approach are: The process is static and have the limitations of static program analysis. Same Code is used in control and data flow tracking. The vulnerability catch out is undesirable in the familiar scenario. This approach will only detect probably vulnerable program code. The advantage of this approach is: To find common types of vulnerabilities CPG and graph traversals are suitable best association of them can be custom fit well to classify vulnerability definitely to a program code hub. The developer has entire authority on false- positive and false-negative standards by utilization of traversals.

In 2014, IJCSNS Journal has a paper on "A performance Evaluation of Vulnerability Detection Net Clarity Audito, Nessus and Retina" [26]. The topic of research in this paper is the comparison between hardware and Software tools. They used Net Clarity Auditor in hardware, wide program Nessus and Retina system connection protective scanner with the aid of a software. The main aspect and comparison is penetrating power, browsing time frame and capability of vulnerability detection schema. The author's job on two sites which are classified as 'Rangsit University Network' and 'Royal Thai Army Network' to accomplish the goal. By Rangsit University Network two zones are chosen that is demilitarized zones and intranet zones. For detecting vulnerability Net Clarity Auditor and open source Nessus Honefeed installed on a computer notebook called Nessus Notebook. Royal Thai Army Network at zone MTC (Military Technology Centre) 1, they use Net Clarity Auditor and open source Nessus Honefeed installed on a computer notebook called Nessus Notebook. At zone 2 Net Clarity Auditor, Nessus Honefeed and Retina Network Security Scanner are installed on Retina Computer. It is seen by experiment that Net Clarityauditor provides the finest in searching achievement, the browsing time frame of Nessus is the lowest and the Net Clarity Auditor is best fitted in the strength of vulnerability detection. The disadvantage of their experiment is that the investment cost of Net Clarity Auditor is very higher as compared to Nessus and Retina Network Security Scanner.

In 2015, IJOART publish a paper on "Web Security Attacks and Injection- A Survey". This paper gives us an overview about the web security attack and injection [27]. Web application is the most common platform for hackers to exploit because they can exploit the web browser and hide themselves from others. Web application stores sensitive data of persons and organization that data is like financial, medical, personal data. By attacking web application hackers can steal the identity of the person or money from users. Hackers can make malicious sites and affect a large number of web clients. They discuss various injection techniques like script injection and attack injection.

In 2017, Queen's University Belfast - Research Portal have a research paper on "Vulnerability Detection in open source software the cure and the cause" [28]. This paper examines the reason for OSS (Open Source Software) vulnerability, why they are a noteworthy issue, and how they might be moderated. They study that OSS is freely distributed software and it can be further developed by developers. OSS is a new research area so there are only three methods to detect vulnerabilities Static analysis, dynamic analysis and code review. These methods have limitations that static analysis creates excessively false positive dynamic analysis does not scale and code review are exceptionally tedious. A new method for OSS vulnerability detection are distributed demand driven security testing are focused on the path they taken and stops attackers attack that are unreported vulnerabilities at a

victim side. Use of execution complexity metrics, this can diminish the code examination exertion as prioritization can take part in view of the measurements. IDE plugins are developed for detecting vulnerabilities using code pattern techniques that allow developer to remove the vulnerabilities. Limitation of OSS is that the Combination of many OSS arises the rate of vulnerabilities. The main advantage of OSS is to improve code quality and time-to-release, testing is focused only on using a selection of detection methods.

III. Software Vulnerabilities Detection Method

Vulnerabilities detection refers using detection technologies for exploring and locating the unexposed vulnerabilities of the system. The vulnerabilities detection method can be broken down into three approaches which are illustrated as static analytics, dynamic and hybrid analysis [19].

3.1 Static Analysis Technique

Techniques applied directly on source code without executing the program are known as static analysis technique, purpose of static technique is getting specific information directly from the source code. In static analysis technique we examine the source code and structure system and try to find out the logical problem, grammatical mistake and implementation issues without running the source code. The traditional technique is a manual code inspection, but it takes too much time and analyzers have deep understanding about vulnerabilities. With the help of computer researchers develop some tools to perform the static analysis technique. Static detection tools are not fully automated, they're output still need human to verify and estimate.

3.1.1 Pattern Matching

In the technique we have to simply match the string or pattern the string pattern that cause vulnerability. The Linux command tool “grep” uses the pattern matching for string. Flawfinder uses it for finding possible vulnerability and arrange them by risk. The value of the parameters of the functions decided the risk level. This method gives much false-positive because the result has no evaluation.

3.1.2 Lexical Analysis:-

In lexical analysis the source code is converted into a number of tokens and then identifies the vulnerability tokens are compared with a vulnerability database. It does not check the grammar or syntax of the program, so that the false-positive number is high. The ITS4 tool is an example of lexical analysis.

3.1.3 Parsing:-

In parsing technique program is represented in the form of parse tree for analyzing the semantics and the syntax of the program. For detecting SQL injection attack Parsing technique is used.

3.1.4 Type Qualifier:-

In the programming language qualifies types and modify the property of a variable they make utilization of emerging type qualifiers. The altered program code is investigated to catch vulnerabilities.

3.1.5 Data flow Analysis:-

It is estimated the all values that a variable can get during the running time of the program, especially preferred for the detection of buffer overflow. This process is taking place in three phases: Control and Data flow, Patter matching and Flow analyzer.

3.1.6 Taint Analysis:-

When any data coming from un-trusted sources, it is taken as a thread in the system and it is marked as tainted. The data is processed and changed into untainted because without changing the data into taint data can not reach to undesirable functions.

3.1.7 Model checking:-

Model checking technique is complex, but once achieve, testing the properties of the system is easy. Model checking is an automatic testing technique. It can detect vulnerabilities only when the system matches its specifications.

3.2 Dynamic Analysis Technique:-

In dynamic analysis technique the execution of program code is necessary and then the behavior of the system is analyzed and gives a conclusion. It monitors the source code during the execution time and find out the defects by detection and analyzing the undesirable condition. In this techniques rate of false alarm is relatively low.

3.2.1 Fault Injection:-

Fault injection is attesting technique in which a fault is injected into the system to analyze the system. For generating the fault some knowledge about the system is required, i.e. white box technique. Inject fault in the system help to seek the flaw in security.

3.2.1 Fuzzing testing:-

No previous knowledge about the system is needed in facing testing, i.e. Black box technique. It takes arbitrary data as inputs for the application and check if the application works correctly. As compared to fault injection this testing is easier to implement.

3.2.2 Dynamic Taint:-

Before entering sensitive function, proper validation of taint data is checked during the execution of code.

3.3Hybrid Analysis Technique:-

Static analysis technique can be used to recognize vulnerabilities in code without code execution. This technique is fast and scalable, used to scan millions of lines of a program during analysis. It suffers the high false positives. Dynamic analysis technique detects vulnerabilities in codes during execution. It suffers state-explosion, high cost and low coverage and efficiency problem, but it gets low false positive. It is a natural idea to join them to complement each other. Hybrid Analysis technique is a technique which combines static and dynamic analysis technique. It overcomes the limitation of both static and dynamic techniques. Rawat et al. Present a hybrid method for c program to detect buffer overflow vulnerability. Flinder_SCA tools are an example of the hybrid analysis technique.

IV. Comparison of Tools in Vulnerability Detection:-

The main tools that are used in the vulnerability detection are listed in the table and mark the technique they used for code detection. LA,DFA,TQ,MC,F,TA,P,SC,BC are stands for Lexical Analysis, Data Flow Analysis, Type Qualifier, Modeling Checking, Fuzzing, Taint Analysis, Parsing, source code, Binary code respectively [29].

Tools	LA	DFA	TQ	MC	F	TA	P	SC	BC
ITS4	✓							✓	
SPLINT		✓						✓	
PMD	✓							✓	
FindBugs	✓	✓						✓	
RATS	✓							✓	
FlawFinder	✓							✓	
Jlint		✓						✓	
Parfait		✓				✓		✓	
Astree			✓					✓	
Frama-c			✓					✓	
CBMC				✓			✓	✓	
JPF				✓				✓	
SLAM			✓	✓				✓	
BLAST			✓	✓	✓			✓	
JPIKE					✓				✓
Silley					✓				✓
Peach					✓				✓
Bestrom					✓				✓
Spider Pig					✓				✓
BitBlaze						✓	✓		✓
BuzzFUzz					✓	✓		✓	
DART					✓		✓	✓	
CUTE					✓		✓	✓	
EXE							✓	✓	
SAGE					✓		✓		✓
KLEE							✓	✓	
AEG							✓	✓	
MAYHEM						✓	✓		✓
S2E							✓		✓
SANTE			✓				✓	✓	
FLINDER-SCA			✓			✓	✓	✓	
SMASH			✓				✓	✓	
Driller					✓		✓		✓

V. Conclusion:-

technology continues to evolve, people and organization require new software to develop their growth in society. Security analyzers give priority to the security of information system. They try to eliminate the vulnerability from the system. So developers, pay more attention to developing vulnerability free software. As conclude detection method for finding the software vulnerabilities is important. A single technique is not so effective, so more techniques and tools are used to find out the vulnerability of software. These technologies and tools can help analyzers to find out and study unknown vulnerabilities.

REFERENCES

[1] C. Kuang, Q. Miao, and H. Chen, "Analysis of software vulnerability," WSEAS Transactions on Computers Research, vol. 1, p. 45, 2006.

[2] I. V. Krsul, "Software vulnerability analysis," Purdue University, 1998.

[3] W. Jimenez, A. Mammar, and A. Cavalli, "Software Vulnerabilities, Prevention and Detection Methods: A Review1," Security in Model-Driven Architecture, p. 6, 2009.

[4] E. E. Schultz Jr, D. S. Brown, and T. A. Longstaff, "Responding to computer security incidents: Guidelines for incident handling," Lawrence Livermore National Lab., CA (USA)1990.

[5] Richard Amankwah, Patrick Kwaku Kudjo and Samuel Yeboah Antwi, "Evaluation of Software Vulnerability Detection Methods and Tools: A Review" International Journal of Computer Applications (0975 – 8887) Volume 169 – No.8, July 2017.

[6] Bingchang Liu, Liang Shi† , Zhuhua Cai, Min Li, "Software Vulnerability Discovery Techniques: A Survey" Fourth International Conference on Multimedia Information Networking and Security " 2012.202 pg:- 152-156

[7] Gábor Vanderer, "Security Awareness and Social Responsibility" 10th Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics 2015

[8] Willy Jimenez , Amel Mammar, Ana Cavalli, "Software Vulnerabilities, Prevention and Detection Methods: A Review" http://www.shields-project.eu/ European Community's Seventh Framework Program (FP 7/2007-203).

[9] Cliff Stoll "What do you Feed a Trojan Horse? Techniques to Solve Hacking Problems" National Bureau of Standards/National Computer Security Center 1987 pg-231-237.

[10] Kreeta Askola, Rauli Puupera, Pekka Pietikainen, juhani Eronen, Marko Laakso, Kimmo Halumen, Juha Roning, "Vulnerability dependencies in Antivirus Software"The Second International Conference on Emerging Security Information, Systems and Technologies 2008, vol-10 pg:-273-278.

[11] Fanping Zeng Juan Li Ling Li Xufa Wang, "Fault Injection Technology for Software Vulnerability Testing Based on Xen" World Congress on Software Engineering 2009 vol-172 pg:-206-210.

[12] T. Tuglular, C. A. Muftuoglu, O. Kaya, "GUI-Based Testing of Boundary Overflow Vulnerability" 33rd Annual IEEE International Computer Software and Applications Conference 2009.189 pg:-539-544.

[13] F. Belli. Finite state testing and analysis of graphical user interfaces. In Proceedings of the 12th International Symposium on Software Reliability Engineering, Washington, DC, USA, 2001, pp. 34-43. IEEE.

[14] Pc-lint. In http://www.gimpel.com/html/pcl.htm, 2008

[15] Keqin Li, "Towards Security Vulnerability Detection by Source Code Model Checking" Third International Conference on Software Testing, Verification, and Validation Workshops 2010.23 pg:-381-387

[16] J. Corbett, M. Dwyer, J. Hatcliff, and Robby, "Expressing checkable properties of dynamic systems: the Bandera specification language," International Journals on Software Tools for Technology Transfer, Springer, Vol. 4, No. 1, October 2002, pp. 1433-2779

[17] J. Hatcliff and M. Dwyer, "Using the Bandera tool set to model-check properties of concurrent Java software," Proceedings of the 12th International Conference on Concurrency Theory, LNCS, Vol. 2154, Springer-Verlag, 2001, pp. 39–58.

[18] Tamer S. Fatayer, Sherif Khattab and Fatma A. Omara, "A Key-Agreement Protocol Based on the Stack-Overflow Software Vulnerability"[IEEE 2010 IEEE Symposium on Computers and Communications (ISCC) - Riccione, Italy (2010.06.22-2010.06.25)].

[19] R. W. Gardner, M. Bishop, and T. Kohno, "Are patched machines really fixed?" IEEE Security and Privacy, vol. 7, pp. 82–85, 2009

[20] Jingzheng Wu∗, Yanjun Wu ∗, Zhifei Wu ∗, Mutian Yang ∗, Yongji Wang, "ulcloud: Scalable and Hybrid Vulnerability Detection in Cloud Computing" Seventh International Conference on Software Security and Reliability Companion " 2013.17 pg:-225-226.

[21] Beihai Liang, Binbin Qu, Sheng Jiang,Chutian Ye1, "Research on Vulnerability Detection for Software Based on Taint Analysis" Proceedings of the 2nd International Symposium on Computer, Communication, Control and Automation 2013.

[22] Zhunyang Pan, Caixia Liu, Shuxin Liu and Shuming Guo, "Vulnerability Discovery Technology and Its Applications" JOURNAL OF SOFTWARE, VOL. 8, NO. 8, AUGUST 2013.

[23] Frank Piessens, "A Taxonomy of causes of Software Vulnerabilities in Internet Software," Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering. IEEE Computer Society Press, Los Alamitos, CA, 2002.

[24] Jay-Evan J. Tevis, and John A. Hamilton, "Methods for the Prevention, Detection and Removal of Software Security Vulnerabilities," Proceedings of the 42nd annual Southeast regional conference. ACM, 2004.

[25] Fabian Yamaguchi, Nico Golde, Daniel Arp and Konrad Rieck, "Modeling and Discovering Vulnerabilities with Code Property Graphs" 2014 IEEE Symposium on Security and Privacy 2014 vol-44 pg:-590-604.

[26] Sanon Chimmanee, Thanyada Veeraprasit, and Chetneti Srisa-An, "A Performance Evaluation of Vulnerability Detection: NetClarity Audito, Nessus, and Retina" IJCSNS International Journal of Computer Science and Network Security, VOL.14 No.3, March 2014.

[27] Swarnaprabha Patil, Prof. Nitin Agrawal, "Web Security Attacks and Injection- A Survey" International Journal of Advancements in Research & Technology, Volume 4, Issue 2, February -2015 pg:-62-67.

[28] Millar, S. (2017). Vulnerability Detection in Open Source Software: The Cure and the Cause. Queen's University Belfast. (https://pure.qub.ac.uk/portal/en/publications/vulnerability-detection-in-open-source-software-the-cure-and-the-cause(94ec148c-80e4-448e-a267-c9ffb992b285).html).

[29] Yunfei Su, 2Mengjun Li, 3Chaojing Tang, 4Rongjun Shen, "An Overview of Software Vulnerability Detection" IJCST Vol. 7, Issue 3, July - Sept 2016.