ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 5, Issue 3, pp: (43-52), Month: July - September 2017, Available at: www.researchpublish.com

A Hybrid Multi-Agent Vulnerability Scanner for Detecting SQL Injections in Web Applications Systems

Hillary Mutai[1], Dr. Robert Oboko[2]

[12]School of Computing& Informatics, University of Nairobi, Nairobi, Kenya

Abstract: **Businesses, as well as individuals interact with web applications on a daily basis due to their flexibility, appropriateness, availability, usability and interoperability. This makes web applications highly vulnerable to cyber-attacks. While web applications have strengthened most organizations by mapping their businesses globally, and facilitating information exchange, the major concern in web applications is mitigating security breaches. This due to the fact that in the recent past, there has been a dramatic increase in web application vulnerabilities being reported as attackers improves their skill and competencies to defeat the existing techniques. The main objective of this study was to design a hybrid multi-agents system for detecting SQL web applications vulnerabilities and formulate system requirements. Multiagent Systems Engineering (MaSE) was used as a methodology for system design in this study. MaSE provided a guide in analysis and design. The Hybrid system designed was subjected to a desktop review as a test bed to determine the time taken to scan vulnerabilities across webgoat, vicnum and genhoud web applications. When compared to Vega, Wapiti and Zap vulnerability scanners, the hybrid multi-agents system performed better in detecting SQL injections. The authors concluded that a hybrid multi-agents system provides a better coverage with no false positive and false negative limited time to scan compared with already existing vulnerabilities scanners.**

Keywords: **Web vulnerability scanners, Multi-agents, SQL injection attacks, and web-based applications.**

I. INTRODUCTION

Most web applications and the underlying databases often contain confidential or sensitive information. This information includes customer details, financial records, credit cards details and user credentials. This form of sensitive information is highly valuable to an organization; making web application an ideal target for attacks. Web applications have become a desirable target for cyber–criminals. SQL injection attacks on web applications have experienced a significant rise in recent years with Owasp ranking SQL injection among the top ten vulnerability attack globally[1]. Past research[2] have shown that most studies have confirmed the shortcomings of SQL web scanners. However, some IT practitioners and researchers have proposed different methods as a solution to SQL injection problem[3]. Most studies have shown that the existing techniques have not been 100% accurate, they suffer from various weaknesses. [4]confirmed that these shortcomings include incomplete implementations, multiple frameworks, a longer span of time taken to scan and False positive and false negative. [5]Studied safety and integrity of web applications contents. In his study, he concluded that numerous attacks on the web application servers exploit weaknesses of a system. Further, he pointed out that compromised web contents by intruders and attackers is an area of concern, not only for the government institutions but to all organizations, web application owners as well as individuals who access web applications.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 5, Issue 3, pp: (43-52), Month: July - September 2017, Available at: www.researchpublish.com

II. LITERATURE REVIEW

A. Overview of Web Application Vulnerabilities:

Web applications have been developed to perform practically every useful function online [6]. These include but not limited to; shopping online, interactive web pages, web search, auctions, banking, gambling and social networking. Web applications are however faced with some weaknesses which vary regarding complexity, detection, and recovery. For example in a report published by [7] , 86% of all websites tested by whitehat, Sentinel had at least one significant vulnerability. The most severe security risks organization face presently is linked to open web applications. These vulnerabilities are weak authentication, SQL injection, Cross-Site Scripting (CSS), session management, sensitive data disclosure, security misconfiguration, and cross-site request forgery among other vulnerabilities (Fig. 1). Besides, making use of components which have known vulnerabilities, as well as unvalidated forwards and redirects, constitute the top most web vulnerability.

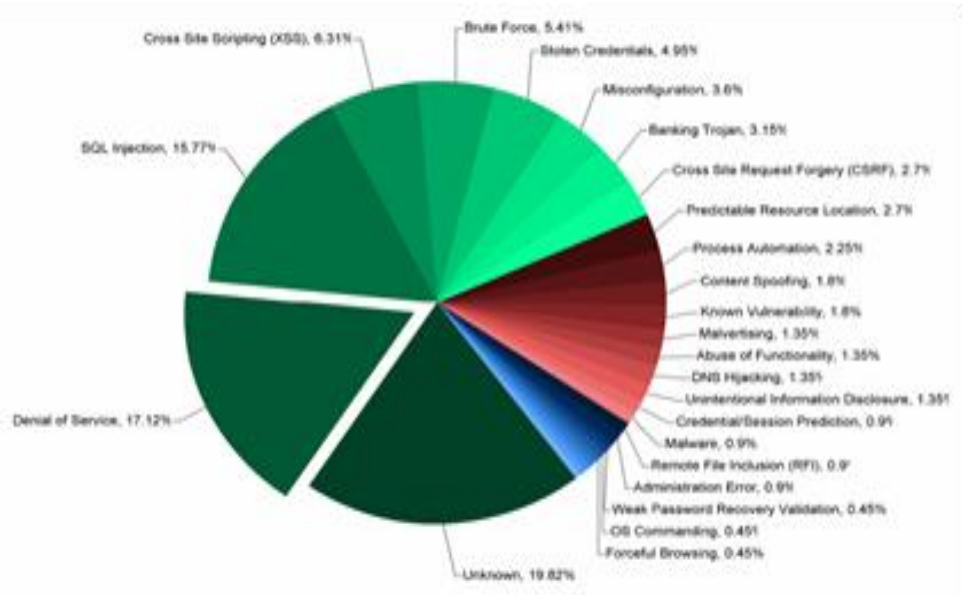


Fig. 1: Web Application Vulnerabities ([1])

[8] Illustrated that manually analysing all the applications for loopholes and prioritizing their importance for remediation can be a daunting task without organized efforts and using automated tools to improve accuracy and efficiency. [9]Shows that the most effective way to detect vulnerabilities in web applications is by manually reviewing the code. However, this procedure suffered various challenges which include; takes much time to consume, expert skills is required, and prone to errors that are overlooked. This led to various security experts developing automated approaches to detect various web security vulnerabilities. The approaches are categorized into three broad testing categories. They are black box, white box and grey box.

The black-box approach was designed to analyse user generated actions on web applications. In this analysis, the process assumes there is no source code for the web application. The technique behind this method is that a user submits many but various patterns of SQL in a web application forms or pages. Analysis is then done to assess the results. When an application shows errors, it is assumed that the application shows some traces of vulnerability. However, the black box technique does not guarantee completeness and accuracy based on the result it captures.

On the other hand, the white-box technique anchors its analysis on the server side when assessing web based applications. When using this method, the application's source code is assumed to be available. Analysis techniques such as using static or dynamic procedures can be invoked. In his paper, [6] made a complete survey of these techniques and made several statements, however this approach suffers from susceptibility to false positives and false negative which is necessitated by imprecisions in analysis. Moreover, this challenge is further compounded by the dynamic and amplified use of scripting languages. While static analysis methods often execute analysis with precision, its emphasis is anchored on the control path.

The white box approach also suffers from dynamic analysis which is performed on paths that are already executed. This poses a challenge regarding the covered paths during an execution [10]. While this is a major weakness of this approach, internal access to web application through dynamic analysis makes the technique precise.

Target of attacker:

According to [2] the attacker searches fields for user input and the parameters linked to it that are exposed to SQL injections in a web application. Different types of databases responds uniquely to attacks and queries directed to them. The attacker "Fingerprint" these information to the database thus able to know the database's type and version housing the web application. Armed with this information, an attacker can perform targeted attack on the database.

B. Web application vulnerability testing tools:

[6] stated that web application vulnerability scanners are programmed software that examines web applications to determine security vulnerabilities. These scanners are able to identify security breaches such as command execution, server configuration errors, SQL injection, and directory traversal among other breaches. Testing tools can be acquired commercially. However, there are many other tools available as open source. Whether it is a commercial or an open source tool, each of these tools have their strengths and weaknesses. Majority of these tools crawl a web application and identify application layer vulnerabilities either by inspecting them for suspicious attributes or manipulating Hypertext Transfer Protocol (Http) messages. For very complex cases these tools emulate attacks originating from peripheral hackers. They also provide advance techniques for depicting different forms of vulnerabilities. Majority of testing tools which work by penetration use fuzz testing method. Fuzzing is the most advanced testing approach that covers wide range of cases. The technique allows the application to accept invalid data as an input. These processes limit the chances of vulnerabilities.

[11] explained that the penetration testing tools are efficient and fast. They are able to detect security breaches quickly. Moreover, unlike conventional black box testing, any person with little technical know-how on security can use penetration. However, despite being efficient, penetration testing tools have disadvantages. This tools cannot find all vulnerabilities. Penetration testing tools are poor at finding vulnerabilities like access control flaws, identification of hard coded backdoors, multi-vector attack, information disclosure and encryption flaws. Further, the use of random data also fails to uncover vulnerabilities unless the fuzzy process is repeated several times. Penetration testing tools do not have any specific goal to work toward, and, therefore, try to attack any possible risk [10].

B. Web Vulnerability Scanning Tools:

The following open source web scanners were used by the researcher in this study, Vega, Zap and Wapiti.

Vega is an open source scanner. It is also used as a testing platform. Vega is a powerful program with capability of finding and validating SQL injections as well as ensuring safety of sensitive information when scanning for vulnerabilities. Developers designed Vega with an automatic scanner suited for accelerated tests.

Vega is designed with an intercepted proxy for tactical inspection. The Vega scanner spots SQL injection and other security flaws in a database. Similarly, the program have an Application Program Interface (API) which makes it easier to extend based on the language being used by the web application on the internet. Vega classifies the scan alert summary into four categories namely high, medium, and low or Info. It provides a report with each of the categories as mentioned in the groupsabove. The report consists of all vulnerabilities found, and the quantity.

Wapiti is an open source web scanning tool that performs a security audit of a web application. It employs a black box approach i.e. it does not study the code. Instead, it checks all the web pages and identifies forms found on a web page where it can insert or reject data. Based on the information derived from its website, it is clear that Wapiti can detect SQL injection among others web vulnerabilities.

Wapiti has an architectural which support POST, Http and GET techniques of web application attack. It's characterized by its ability to provide comprehensive reports after completion of a scan, authentication using various methods such as NTLM, Kerberos or Basic, ability to define or limit the scope of the scan to a folder, web page or a domain, update to understand recently release web development technologies such as HTML5.

Zap Attack Proxy (ZAP) is an open source web scanning tools that use a Graphical User Interface (GUI) interface. The application suits both new developers and experienced programmers. To utilize the application, simply input the URL of the application you would like to scan and wait for the scan to be completed then review the generated report.

Multi-Agent System (MAS):

MAS designate multiple entities (agents) in a distinctive environment. The system is composed of autonomous collection of agents capable of defining their objectives and actions. Further, they interact and collaborate with one another by passing messages (communication). In a multi-agent system environment, the agents collectively work to address a particular problem in context. The system provide a perfect platform for negotiation, competition, coordination and cooperation among diverse functional units. Communication is made possible by using an interaction protocol to accomplish their roles.

Role of Agents in Web Application Security:

Multi-agent systems has positively impacted web applications. Being a distributed systems, it has a number of advantages. Multiple agents operates in parallel, hence this results in an increased overall speed. Agents operate in an asynchronous which increases its efficiency. When a single or several agents fail it does not interfere with the whole system. This is because other agents in the system undertake the role, thus increasing agent robustness and reliability. Agent system has the capability of being scaled. This can be done based on the magnitude of the problem to be solved. Scaling can be achieved by adding new agents; adding new agents does not hinder the operations of other agents in the system. Scaling increases agent's flexibility. Compared to centralized systems, agent system are far much more cost-effective since agent system is consists of simple subsystems with low unit cost. Agents are designed autonomously. Individual agents are distinctively designed and developed by developers.

D. Agent Development Methodology:

Multi-agent System Engineering Methodology (MaSE)

The MaSE methodology is used in the distributed agent paradigm. The methodology has analysis and design phases. The analysis phases comprises of detailed stages that include; capturing system goals, use cases, refining roles of agents among others. In design phase, four phases are applied. These phases include building classes for agents, creating conversations, assembling classes for agents and designing the system.

This study used MaSE methodology for the design and analysis of the system.

E. Related Work:

Manual approaches:

In this section various techniques about the manual approaches including defensive programming used to detect and prevent input manipulation of web application vulnerabilities in particular SQL injection [4]. [4]Stated that the input text or data manipulations could be avoided by designing the system to prevent user input from comprising malicious characters or keywords. Moreover, they stated that input filters could be achieved using white or black list approach by programmers. However, many developers of web applications pay little attention on risk linked to SQL injections on applications that incorporate back-end databases which gives the attackers an opportunity to perform their attacks. Code review approach detects bug at low cost. However, it consumes a lot of time in comparison with automated static analysis. There are high chances that it may be implemented by developers because of tight timelines and race to ship an application according to [12].

In this research, a Hybrid multi-agent technique and approach was used to help advance validation of user's input by delivering data on a system.

Static and Dynamic Code Analysis Approach:

The rise of web application vulnerabilities has attracted extensive discussion in practice and research. This is because SQL injection has contributed to a significant security threat in web application databases [1]vulnerability list report). Traditionally, developers use data validation by checking through the system entry points. They use commands such as GET and POST[13]. However, a major rise and demand for more secure web applications and significant complex applications create a challenge on the security of web applications. [13][14], in their paper, presented an improved

approach for detecting vulnerabilities such as SQL injection attacks in web application, this technique is implemented using Ardilla. Ardilla automatically creates model input that; expose SQL Injection, symbolically traces taints through implementation including database access and transforms the inputs to generateconcrete results. However, this approach has various limitations: the approach is based on the source code of the web application, its design is limited to specific application like PHP applications, and therefore, this approach requires a developer availability. Also, it requires learning skills because a developer will need to adjustthe source code. This approach faces a major challenge in a case the pioneer developer abandons the project, it would be cumbersome patching the vulnerabilities.

In an attempt to come up with a more efficient approach, [15] presented a technique that uses combination of dynamic and static assessment to inhibit dissemination of SQL query by the user in SQL inquiry attributes standards. The process is achieved by using a utility that has the capability of detecting SQL static query element in web driven application. This is achieved at run time by detecting SQL queries. This approach compares SQL derived from normal users with SQL query produced dynamically by a potential attacker. Moreover, this approach has challenges in regard to learning and adjusting the source code.

These pieces of studies inform this study by providing an understanding of various components to be incorporated in the current approach to enhance efficiency in SQL injection detection, reliability, lower rate of false positive and false negative and reduction in the span of time taken to scan web applications for SQL injections. Furthermore, lack of cross platform application has also been noted extensively in literature enabling clarity in defining the multi-platform of interest in this study.

AMNESIA, according to [16]is an approach that amalgamate runtime monitoring and static in a web application in detecting and preventing SQL injection attacks. The static part is used to put together a legal query using program analysis and the other dynamically generates queries automatically using monitoring during runtime.

A database server detects structural query injected by a form support approach. If queries resist the approach, then the technique prevents operation of the queries on the database server using these steps; identifying the hotspot, building SQL query models, instrumenting application and runtime monitoring. The major shortcoming of this approach is it is only implemented on Java supported application.

Use of API Approach:

[17]found the concept behind SQL DOM which is simpler than others depending on developer's ability to perform the complex defensive coding techniques in building dynamic SQL queries using strings. However, an API was used to enhance the security. The Sqldomgen is used to perform analysis at compile time on the database schema; hence constructing various set of SQL query classes integrated with IDE which developer's calls directly to construct SQL queries. This approach maps various variations of SQL queries according to tables and columns. Moreover, [17] stated that this approach was categorized into various classes with strong-typed methods including SQL statements, table columns and where conditions. Validation of data types is automatically achieved by mapping the data types in the database. In this approach, development of column classes strings replaces quote with doublequote in strings at runtime to sanitize them. However this approach suffers from various weaknesses and limitations which include writing new query, rewriting query generated codes, overheads for developers training and code rewriting, its full-object policy comes at a cost and stored procedures remain unprotected.

Hybrid Approach:

[15] proposed an approach which is an extension of replica based hybrid technique to curb SQL Injection attack (MHAPSIA) works in production web environment which provides protection from SQL injection. This approach is categorized into a production environmental mode and safe mode. In safe mode, a secure query model for SQL is created while in the production environmental, dynamic queries are validated against a secure query model in safe mode, and approves typical input request against sanitizer mode.

Other studies such as [18] have investigated SQL Injection with an intention of improving the security of web applications using an algorithm. These studies have broadly examinedalgorithm and describe the leaks depends on analysing the web application source code. The study gave 12 steps given each performs specific kind of leaks where SQL injection vulnerabilities are discussed.

[18]contributed significantly to theHybrid Approach by developing a program that inspects various forms of writing; incode, no standard, and the availability of some alternatives commands. The proposed algorithm combines only two compiled languages for use, i.e., VB and C#. The study is limited in that other languages like Java are left out.

[12] presented a hybrid approach which combines an audit and signature based method. In the signature based technique, the authors used a token wise string to present a detection mode for SQL injection. They used a technique known as the Hirschberg algorithm. The algorithm was based on the principle of divide and conquers. The technique was adopted to minimize the complexity of time and space. In audit technique, the authors analyzed transactions to determine presence of malicious access. However, this technique harboured several challenges such as inefficiency in detecting attacks. Further, it generated false positives and false negatives.

A significant gap in this study is focus on one single approach to SQL injection detection rather than taking a combination of various methods. Besides, the literature points out that most existing web application SQL injection scanners have not been 100% in detecting web vulnerabilities. Therefore, this study seeks to contribute to these bodies of knowledge in SQL injection by investigating existing approaches and proposing a better approach to efficiency detection of SQL injection with a minimal rate of false positive and false negative.

Proposed architecture:

The study proposed an architecture in which situated agents in various modules continuously detects SQL injection web vulnerabilities (Fig. 4). The agents used a statement from the web application to detect SQL injections. On receiving the statement from web application, the agent analyses the statement for any suspicious activity and either make a decision for the statement to be executedif no suspicious activity is found, then the statement is sent to database for execution. Upon existence of a suspicious activity, the statement is compared against the existing specification and it is send to the audit agent module. If the analysis and the auditing module processes have been fulfilled, it will provide a complete transaction. Hence, and SQL injection alert is generated.

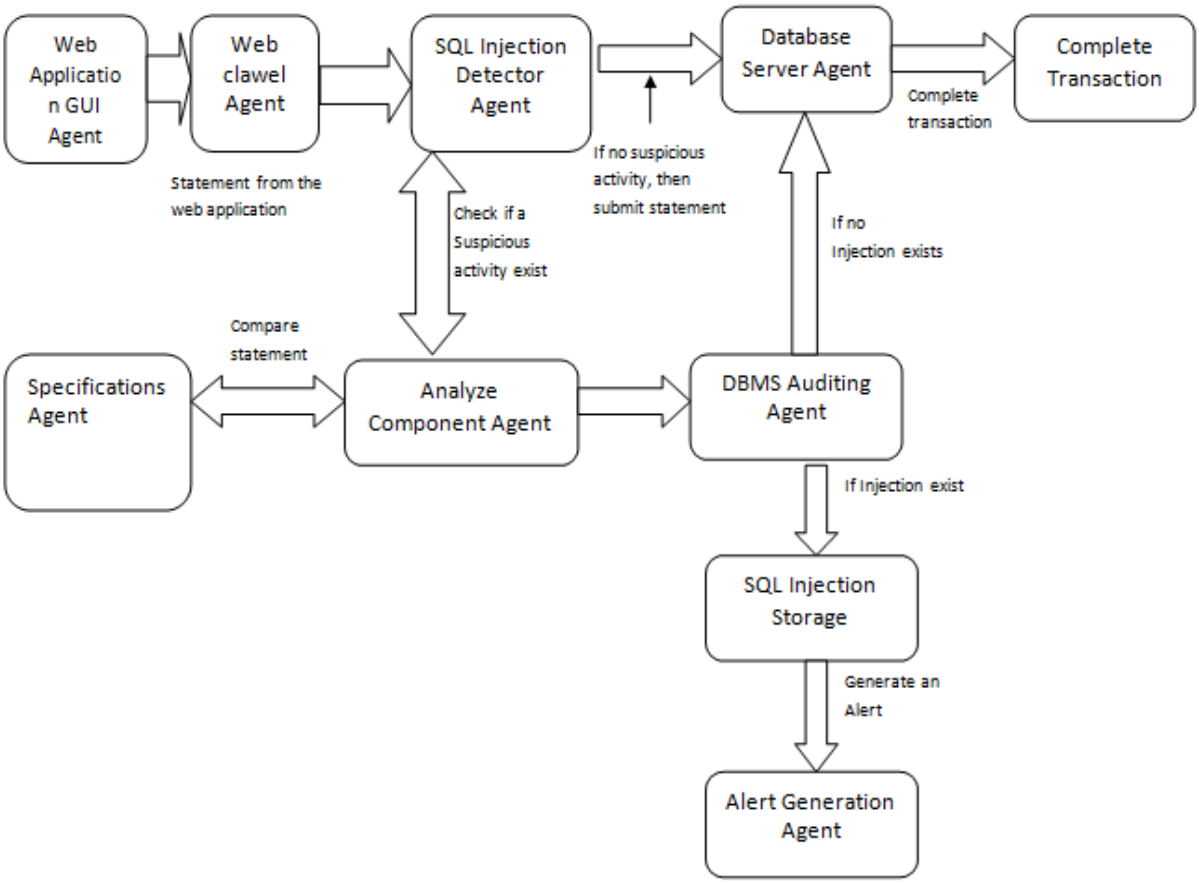


Fig. 2: Representation of the proposed architecture ([12])

III. METHODOLOGY

Target Population:

The study targeted opens source web vulnerability testing tools. Any open source web application vulnerability scanning tool was eligible to be used in the study.

Sampling Procedure:

Purposive sampling based on various categories or classifications was used to select the tools used in the study. The tools selected from lists available on various online portals that classify open source web scanning tools in reference to various factors such as their capability and detecting vulnerabilities were selected. The web site had all well known vulnerabilities in advance which the tools was benchmarked with. Website applications from WAVSEP by [19] were developed to analyse and detect the degree of accuracy of scanners being used. The goal was to increase broad understanding of detection barriers, and determine how each scanning tool could navigate diverse vulnerabilities discovered. The metrics used in this research included, detection accuracy, number of vulnerabilities detected, time taken to scan web application, consistency and stability, features available.

Sample Size:

Purposive sampling was used to select web application with known vulnerabilities as well as the tools for scanning the chosen applications. This was because purposive sampling accords the researcher the leeway to target cases that had the required information. However all the selected tools were benchmarked with OWASP top ten lists of vulnerabilities. Analysis was performed against the set metrics to chosen tools which was a representative of the sample. The algorithms of these tools were analysed and used to inform on the required improvement.

Tools selected for this research:

The open source vulnerable tools selected were Wapiti, Zed Attack Proxy and Vega, while Ron Scanner was developed by the researcher to test and validate the hybrid multi-agent system.

System design:

Multiagent Systems Engineering (MaSE) was used as a methodology for system design in this study. MaSE provided a guide in analysis and design. Two steps were involved; analysis and design.

Analysis phase:

In the analysis phase, the first step entailed capturing the system goals which involved initial specification of the system and transformation to a set of system goals. These goals were analyzed and mapped to a Hierarchical diagram. The second step used cases to build a set of sequence. This would assist in identifying the initial roles together with message routes by the system analyst. The third step was refining roles; here constructed goals are transformed with sequence diagrams into roles. When roles have been created, each role is associated with tasks that describe the behaviour exhibited to achieve its goals successfully.

Design phase:

The second step was design phase. The design phase involved building agent classes using defined roles in the analysis phase. Design stage provided a class diagram for the agent at the end. This step illustrated the general system organization that involved classes of agent and the conversations. The second step in the design phase, was construction of conversations. The conversation is constructed from messages and agents states for individual path communicating using a Con-current Task Method, increasing messages and agent states for increased performance. Assembling agents step, creates the parts of the agent classes. The steps specified the architecture of the agent and defined the features and components that constitute the architecture.

Hybrid multi-agent System:

This system was designed based with an objective to reduce scanning time in web application to achieve increased detection accuracy. In this study, accuracy of the Hybrid multi-agent system was fully optimized. Additionally, the fuzzing and crawling components were engineering to work efficiently and deliver acceptable results.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 5, Issue 3, pp: (43-52), Month: July - September 2017, Available at: www.researchpublish.com

Test application:

webgoat, vicnum and genhoud

Variables tested:

i. Detection accuracy – vulnerabilities detected by the web vulnerability tools. This is expressed in terms of percentage.

ii. Time – the time taken by the any of the tools under study to scan a given web application

iii. Consistency and reliability – this was arrived at after running the same tool several times against the same web application under the same conditions and configurations and comparing the results.

Data Collection:

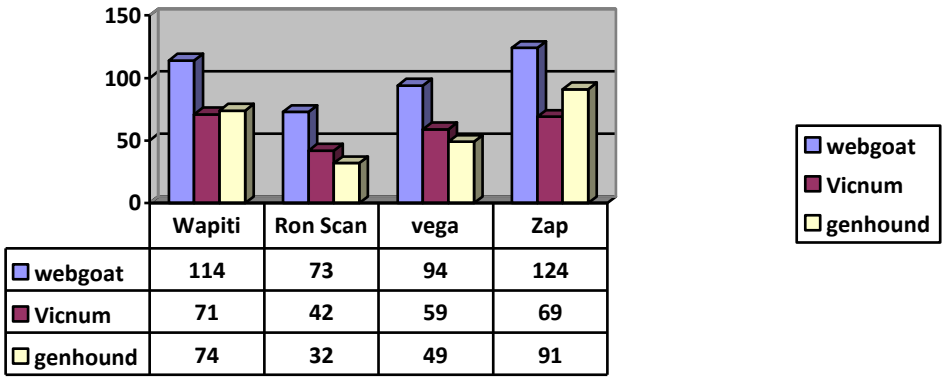
This study relied primarily on quantitative data. Data was collected on the detection accuracy, the number of vulnerabilities detected, reliability, consistency and stability, features available of the tools. Data was collected through observation and examination of the reports displayed at the end of the scanning process. These metrics have been used in a study that was conducted by [20]. Each web scanning tool was tested against the web application with all the relevant settings configured.

IV. RESULTS

The simulation results were evaluated by comparing the performance of the open source scanner and the Ron scanner under the set metrics.

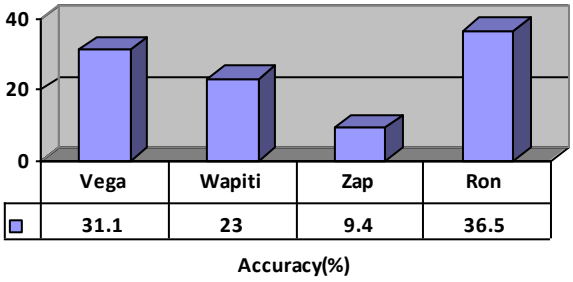
Time taken to scan various applications:

To maintain reliability of the study the researcher administered 15 test on the web vulnerability scanners, the assumption was that all the web vulnerabilities were at similar conditions during the test. The results are as shown in the following figure(s)



	Wapiti	Ron Scan	vega	Zap
webgoat	114	73	94	124
Vicnum	71	42	59	69
genhound	74	32	49	91

Fig.3: Web Tools Vs Scan Time



	Vega	Wapiti	Zap	Ron
	31.1	23	9.4	36.5

Fig. 4: Scanning Tools Accuracy

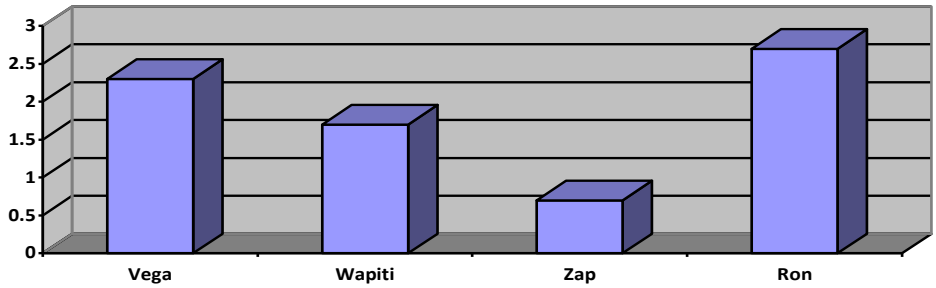


Fig. 1: Scanning Tools Consistency

V. DISCUSSION

From the results, different patterns of behaviour were observed in scan time taken and the number of vulnerabilities detected. Ron scan recorded a mean scan time of 16.5 % which is shown to be the lowest as compared to other vulnerabilities. The results demonstrate that our proposed hybrid multi-agent system is able to perform a scan on a web application faster than other selected vulnerability tools and more accurate in detecting SQL vulnerabilities. The mean scan time is 2.2 sec lower and the mean vulnerabilities detected is 0.4sec higher in our proposed hybrid multi-agent system.

In a study conducted by [21] to test vulnerability, using tools such burp scanner and the IBM's rational app scan noted crawling contemporary applications is a major problem for many WVS. The finding highlighted by [21] reflects this research. Thus, a hybrid system should be considered to increase performance for scanning vulnerabilities.

[22]showed that open source WVS is weak in identifying vulnerabilities. They also take longer to scan for vulnerabilities. A study by [22] is in tandem with this study. This study designed a sophisticated algorithm to mitigate this concern and increase vulnerability identification and detection. [23]analysed various web vulnerabilities scanners and their results showed that time take to take to perform scan varies with different tools and many shown to take longer time, the researcher was able to perform SQL injections in less time by fuzzing web applications using hybrid multi-agent system.

[8]carried out identical study by employing incursion tools to evaluate effectiveness of available WVS. Both open source and commercial tools were used in the study. These tools included W3AF, Wapiti, N-stalker and W3AF. The tools were subjected to a customized edition of Buggy Bank web applications. The apparatus employed were tested for XSS, SQL and other vulnerabilities. Researchers found out that testing WVS in secure and non-secure applications is a favourable technique for discovering web vulnerabilities. Additionally, the study observed that in discovering non-traditional instances of SQLI, further, studies need to be undertaken to increase detection techniques used by these tools. In this study, the researcher used permutation and heuristic in the detection process.

The hybrid multi-agent system is capable of mitigating concerns highlighted by other previous studies. This is achieved through using multiple multi agents during the process of vulnerability and subsequent improvement of the existing vulnerability detection techniques. For example, this study observed that WVS use GET and POST strategies to detect weaknesses in an application. These two methods require sufficient time to scan. Despite the time factor, they provide accurate results.

VI. CONCLUSION

The hybrid system presented in this study is extensive. This is in regard to execution and detection method against vulnerabilities found in web application. The hybrid multi agent system executes faster and scan web application for vulnerabilities. It produces a report for an evaluator to identify the vulnerabilities that have been discovered. However, since the hybrid multi-agents designed in this study did not achieve 100% when scanning for existing vulnerabilities, a robust crawling algorithm component should be increased. This will enable "deep" crawling to identify vulnerabilities. Additionally, the result shows that the hybrid system designed should be optimized to shorten scanning time. Studies aimed at creating and implementing a sophisticated multi agent should be pursued to increase capacity of detecting more vulnerabilities.

REFERENCES

[1] Owasp, "OWASP Top 10 - 2013," *OWASP Top 10*, p. 22, 2013.

[2] K. Phalguna Rao, A. BSasankar, and V. Chavan, "Analysis of Detection and Prevention Techniques Against SQL Injection Vulnerabilities," vol. 4, pp. 50–55, 2013.

[3] R. Joseph, "Sql-Injection Tool for finding the Vulnerability and Automatic Creation of Attacks on JSP," vol. 1, no. 9, pp. 60–68, 2012.

[4] A. Kumar Singh and S. Roy, "A network based vulnerability scanner for detecting SQLI attacks in web applications," *2012 1st Int. Conf. Recent Adv. Inf. Technol. RAIT-2012*, pp. 585–590, 2012.

[5] K. Chaitali and V. Kala, "Report on Vulnerabilities in Web Applications," vol. 4, no. 9, 2014.

[6] D. Stuttard and M. Pinto, *The Web Application: Hacker's Handbook*, 2nd ed. Indiana: Wiley Publlishing, Inc., 2011.

[7] V. Dwivedi, "Web Application Vulnerabilities : A Survey," vol. 108, no. 1, pp. 25–31, 2014.

[8] D. A. Shelly, "Using a Web Server Test Bed to Analyze the Limitations of Web Application Vulnerability Scanners Using a Web Server Test Bed to Analyze the Limitations of Web Application Vulnerability Scanners," Virginia Polytechnic Institute and State University, 2010.

[9] F. E. Eassa, M. Zaki, A. M. Eassa, and T. Aljehani, "An integrated multi-agent testing tool for security checking of agent-based web applications," *WSEAS Trans. Comput.*, vol. 13, no. 2, pp. 9–19, 2014.

[10] M. Mirjalili, A. Nowroozi, and M. Alidoosti, "A survey on web penetration test," no. November, 2014.

[11] A. Jaiswal, "Security Testing of Web Applications : Issues and Challenges," vol. 88, no. 3, pp. 26–32, 2014.

[12] M. N. Bhat, N. Veeranjaneyulu, and A. Raghunath, "International Journal of Advanced Research in Computer Science and Software Engineering A Hybrid Approach for handling SQLI Vulnerabilities in Web Applications," vol. 3, no. 12, pp. 604–609, 2013.

[13] R. Rawat, C. Singh Dangi, and J. Patil, "Safe Guard Anomalies against SQL Injection Attacks," *Int. J. Comput. Appl.*, vol. 22, no. 2, pp. 11–14, 2011.

[14] M. J. A and P. Savaridassan, "Detection and Fixing of web application vulnerabilities on field data," vol. 4, no. 2, 2015.

[15] R. Johari and P. Sharma, "A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection," *Proc. - Int. Conf. Commun. Syst. Netw. Technol. CSNT 2012*, pp. 453–458, 2012.

[16] W. G. J. Halfond, A. Orso, D. A. Kindy, and A. S. K. Pathan, "AMNESIA: Analysis and Monitoring for NEutralizing SQL-injection Attacks," *Int. J. Commun. Networks Inf. Secur.*, vol. 5, pp. 80–92, 2013.

[17] C. Khairnar, "Detection and Automatic Prevention against SQL Injection Attacks and XSS Attacks perform on Web Applications," vol. 5, no. 11, 2015.

[18] H. Al-Amro and E. El-Qawasmeh, "Discovering security vulnerabilities and leaks in ASP.NET websites," *Proc. 2012 Int. Conf. Cyber Secur. Cyber Warf. Digit. Forensic, CyberSec 2012*, pp. 329–333, 2012.

[19] H. Shahriar, S. North, and W. Chen, "Early Detection of Sql Injection Attacks," *Int. J.*, vol. 5, no. 4, pp. 53–65, 2013.

[20] K. Mcquade, "Open Source Web Vulnerability Scanners : The Cost Effective Choice ?," pp. 1–13, 2014.

[21] A. Doup, M. Cova, and G. Vigna, "Why Johnny Can ' t Pentest : An Analysis of Black-box Web Vulnerability Scanners."

[22] M. S. Patole and S. D. Kothimbire, "A Review on Web Security Mechanism Performance Evaluation Using Vulnerability and Attack Injection," vol. 3, no. 11, pp. 2585–2588, 2014.

[23] D. Zlatkovski and A. Mileva, "EVALUATION AND TESTING OF SEVERAL FREE / OPEN SOURCE WEB," no. Ciit, pp. 221–224, 2013.