International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 8, pp: 504-510 www.ijaem.net ISSN: 2395-5252

Malicious Link Detector on Website: Review

*Muhammad Saidu Aliero[1], Zulkifilu Muhammad[1,] Abdulrashid Allami[1], Basiru Umar Aliero[2], Salisu Adamu Aliero[3],

[1]*Department of ICT Science, Faculty of Engineering, Kebbi State University of Science and Technology Aliero, Nigeria*

[2]*Department of Guidance and Counseling, Faculty of Education, Kebbi State University of Science and Technology Aliero, Nigeria.*

[3]*University Library, Kebbi State University of Science and Technology Aliero, Nigeria.*

msaidua2000@gmail.com

Date of Submission: 15-10-2020 Date of Acceptance: 15-11-2020

ABSTRACT: Malicious link is one of the most common web-based application vulnerabilities that can be invaded to redirect users from trusted web application to malicious application result in unauthorized access and unauthorized data modification. Researchers have proposed many methods to tackle this problem, however these methods fail to address the whole problem of malware injection attack, because most of the approaches are vulnerable in nature, cannot resist sophisticated attack or limited to scope of subset particular malware type. In this paper we provide a detailed background of malware, and discuss most commonly used method by programmers to defend against malware using analytical evaluation approach lastly we conclude and recumbent future improvements .

Keywords: Malicious link, web application, web application security, web application vulnerably dynamic approach, analytical evaluation

I. INTRODUCTION

Malicious URLs have been widely used to mount various cyber-attacks including spamming, phishing and malware. Detection of malicious URLs and identification of threat types are critical to thwart these attacks. Automatic security assessment tools are used to automatically detect existence of defect, weakness or security flaws that can be exploited by potential attackers (MS Aliero et al, 2015).

These tools provide automatic way of security assessment either by examining the source code of applications or through penetration testing. Security assessment report published by Software Administration, Network and Security and Institute of Computer Security/FBI, show almost 500 computer security analyst concludes that 55% of web penetration tester use automatic approach for testing and evaluating effectiveness of their applications (Antunes & Vieira, 2010).

This present a major concern that needs to be investigated as web application is today home to billions of users over the globe and Adversaries have used the Web as a vehicle to deliver malicious attacks such as phishing, spamming, and malware infection. For example, phishing typically involves using a fake website seemingly from a trustworthy source to trick people to click a link that takes you to a counterfeit webpage.

To address this challenge many researches in academia and industries have been working on malicious links detection typically through inspecting link to detect their malicious content (McAfee, 2011). Today, many malicious links are found on the web. And because of the high rate of joblessness in Nigeria, job seeking websites are becoming popular as they are convenient and economical. One can easily get a space with a publisher such as Jobbeman.com and place adverts for job vacancies. On the other hand, a lot of people rely heavily on those websites to have information about available job vacancies in other to apply. Unfortunately this bless can also turn to a curse: hackers and attackers have found web links to be low- cost and highly effective means to conduct malicious activities.

This research will focus on analytical analysis on proposed solution to detect malicious links this analysis would help administrator in selecting the best tool in the market or research who wants improve existing solutions.

II. EXISTING SOLUTION

Today's age, it is almost mandatory to have an online presence to run a successful venture. As a result, the importance of the World Wide Web has continuously been increasing especially in job seeking websites.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 8, pp: 504-510 www.ijaem.net ISSN: 2395-5252

Unfortunately, the technological advancements come coupled with new sophisticated techniques to attack and scam users in job seeking websites. Such attacks include rogue websites that provide work counterfeit goods, financial fraud by tricking users into revealing sensitive information which eventually lead to theft of money or identity, or when installing malware in the user's system.

This section represent review of the state-of-the-art of malicious URL detection in literature. The research specifically focus on the contributions made for crawling activities used in detecting malicious links. The work also present categories of different various types of crawlers representation used for creating the training data for this task, and also categorize various learning algorithms used to learn a good prediction model.
Most of common way by which user can be compromised is through spreading compromised URLs (or the spreading of such URLs forms a critical part of the attacking operation) (Hong, J. 2012).

Compromised URLs that are used for cyber-attacks are termed as malicious URLs. In fact, it was noted that close to one-third of all websites are potentially malicious in nature (Patil, D. R., & Patil, J. B. 2015) demonstrating rampant use of malicious URLs to perpetrate cyber-crimes.

2.1 Review on Malicious links detector

A Malicious URL or a malicious web site hosts a variety of unsolicited content in the form of spam, phishing,or drive-by-exploits in order to launch attacks. Unsuspecting users visit such web sites and become victims of various types of scams, including monetary loss, theft of private information (identity, credit-cards, etc.), and malware installation.Popular types of attacks using malicious URLs include:Drive-by Download, Phishing and Social Engineering, andSpam (Cova, M., Kruegel, C., & Vigna, G. 2010). Drive-by-download (Heartfield, R., & Loukas, G. 2016)refers to the unintentionaldownload of malware upon just visiting a URL. Such attacksare usually carried out by exploiting vulnerabilities in pluginsor inserting malicious code through JavaScript.

Phishing and Social Engineering attacks (OpenDNS, L. L. C. 2016) trick the users into revealingprivate or sensitive information by pretending to be genuineweb pages. Spam is the usage of unsolicited messages forthe purpose of advertising or phishing. These types of attacksoccur in large numbers and have caused billions of dollars' worth of damage every year. Effective systems to detect such malicious URLs in a timely manner can greatly help to counter large number of and a variety of cyber-security threats.

Consequently, researchers and practitioners have worked to design effective solutions for Malicious URL Detection. The most common method to detect malicious URLs deployed by many antivirus groups is the black-list method. Black-lists are essentially a database of URLs that havebeen confirmed to be malicious in the past. This database iscompiled over time (often through crowd-sourcing solutions, e.g. PhishTank (Sinha, S., Bailey, M., & Jahanian, F. 2008) as and when it becomes known that aURL is malicious. Such a technique is extremely fast due to asimple query overhead, and hence is very easy to implement.Additionally, such a technique would (intuitively) have a verylow false-positive rate (although, it was reported that oftenblacklisting suffered from non-trivial false-positive rates (Garera, S., Provos, N., Chew, M., & Rubin, A. D. 2007).

However, it is almost impossible to maintain an exhaustive listof malicious URLs, especially since new URLs are generatedevery day. Attackers use creative techniques to evade blacklistsand fool users by modifying the URL to "appear" legitimatevia obfuscation. (Garera, S., Provos, N., Chew, M., & Rubin, A. D. 2007) identified four types ofobfuscation: Obfuscating the Host with an IP, Obfuscating theHost with another domain, obfuscating the host with large hostnames, and misspelling. All of these try to hide the maliciousintentions of the website by masking the malicious URL.Recently, with the increasing popularity of URL shorteningservices, it has become a new and widespread obfuscationtechnique (hiding the malicious URL behind a short URL) (Garera, S., Provos, N., Chew, M., & Rubin, A. D. 2007),Alshboul, Y., Nepali, R., & Wang, Y. 2015).

Once the URLs appear legitimate, and users visit them,an attack can be launched. This is often done by maliciouscode embedded into the JavaScript. Often the attackers willalso try to obfuscate the code so as to prevent signature basedtools from detecting them. Attackers use many other simpletechniques to evade blacklists including: fast-flux, in whichproxies are automatically generated to host the web-page; algorithmicgeneration of new URLs; etc. Additionally, attackerscan often simultaneously launch more than one attack, whichalters the attack-signature, making it undetectable by tools thatfocus on specific signatures. Blacklisting methods, thus havesevere limitations, and it appears almost trivial to bypass them,especially

due to the fact that blacklists are useless for makingpredictions on new URLs.

To overcome these issues, in the last decade, researchershave applied machine learning techniques for Malicious URLDetection (Patil, D. R., & Patil, J. B. 2015), Chhabra, S., Aggarwal, A., Benevenuto, F., & Kumaraguru, P. 2011),, (Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. 2009 and (Hoi, S. C., Wang, J., & Zhao, P. 2014. Machine Learning approaches,use a set of URLs as training data, and based on the statisticalproperties, learn a prediction function to classify aURL as malicious or benign. This gives them the ability togeneralize to new URLs unlike blacklisting methods. Theprimary requirement for training a machine learning modelis the presence of training data. In the context of maliciousURL detection, this would correspond to a set of large numberof URLs. Machine learning can broadly be classified into supervised, unsupervised, and semi-supervised, which correspond to having the labels for the training data, not having thelabels, and having labels for limited fraction of training data. Labels correspond to the knowledge that a URL is maliciousor benign.

After the training data is collected, the next step is toextract informative features such that they sufficiently describethe URL and at the same time, they can be interpretedmathematically by machine learning models. For example,simply using the URL string directly may not allow us tolearn a good prediction model (which in some extreme casesmay reduce the prediction model to a blacklist method).

Thus, one would need to extract suitable features based on someprinciples or heuristics to obtain a good feature representationof the URL. This may include lexical features (statisticalproperties of the URL string, bag of words, n-gram, etc.), host basedfeatures (WHOIS info, geo-location properties of thehost, etc.), etc. These features after being extracted have to be processed into a suitable format (e.g. a numerical vector), suchthat they can be plugged into an off-the-shelf machine learningmethod for model training.

The ability of these features toprovide relevant information is critical to subsequent machinelearning, as the underlying assumption of machine learning(classification) models is that the feature representations ofthe malicious and benign URLs have different distributions.Therefore, the quality of feature representation of the URLs iscritical to the quality of the resulting malicious URL predictivemodel learned by machine learning.Finally, using the training data with the appropriate featurerepresentation, the next step in building the predictionmodel is the actual training of the model. (Sahoo, D., Liu, C., & Hoi, S. C. 2017)

There are plentyof classification algorithms that can be directly used over thetraining data (Naive Bayes, Support Vector Machine, LogisticRegression, etc.). However, there are certain properties ofthe URL data that may make the training difficult (both interms of scalability and learning the appropriate concept). Forexample, the number of URLs available for training can bein the order of millions (or even billions).

As a result, thetraining time for traditional models may be too high to bepractical. Consequently, Online Learning (Canali, D, et al 2011), March) a family of scalable learning techniques have been heavily applied forthis task. Similarly, for this task, URLs are represented usingthe bag-of-words (BoW) features. These features basicallyindicate whether a particular word (or string) appears in aURL or not - as a result every possible type of word that mayappear in any URL becomes a feature. This representationmay result in millions of features which would be very sparse(most features are absent most of the time, as a URL willusually have very few of the millions of possible words presentin it). Accordingly, a learning method should exploit thissparsely property to improve learning efficiency and efficacy.

Despite the promising generalizing ability of machine learningapproaches, one potential shortcoming of these approachesfor malicious URL detection may be their resource intensivenature (especially while extracting features that are non-trivialand expensive to compute), reducing their practical value whenrequiring real-time security assurance compared to blacklistingmethods.

2.2 Review onWeb Application Crawling

In this section, the study provides a brief introduction to crawling and review the previous works in the literature. Furthermore, the section alsoemphasis on web crawling activities which is one of the major phases in detection of malicious links in web applications together with the relationships among document groups; more specifically, classes, to improve the performance of crawling activities. In particular, the study consider factors that the linked to low coverage of crawling activities by most of the previously proposed crawlers.

With the very fast growth of WWW, the quest for locating the most relevant answers to

users' information needs becomes more challenging. In addition to general purpose Web directories and search engines, several domain specific Web portals and search engines also exist, which essentially aim to cover a specific domain/topic (e.g., education), product/material (e.g., product search for shopping), region (e.g., transportation, hotels etc. at a particular country Chakrabarti,(S., Van den Berg, M., & Dom, B., 1999) or media/file type (e.g., mp3 files or personal homepages (Pant, G., & Srinivasan, P. 2006).

Such specialized search tools may be constructed manually-by also benefiting from possible assistance of the domain experts or automatically. Some examples of the automatic approaches simply rely on intelligent combination and ranking of results obtained from traditional search tools (just like meta search engines), whereas some others first attempt to gather the domain specific portion of the Web using web crawling techniques and then apply other operations (e.g., information extraction, integration, etc.).

(Chakrabarti, S., Van den Berg, M., & Dom, B.,1999)were the first to propose a soft-focus crawler, which obtainsa given page's relevance score (i.e., relevance to the target topic) from a classifierand assigns this score to every URL extracted from that page. We refer tothis soft-focus crawler as the baseline focused crawler. In a more recent work, they have proposed using a secondaryclassifier to refine the URL scores and increase the accuracy of this initial softfocused crawler (Chakrabarti, S., Punera, K., & Subramanyam, M. 2002). An essential weakness of the baseline focused crawler is its inability to modeltunneling; that is, it cannot tunnel toward the on-topic pages by following a pathof off-topic pages(Barfourosh, A. et al, 2002)..

Two other remarkable projects, the context-graph-basedcrawler (Diligenti, M. et al, 2000) and Cora's focused crawler (A. Mccallum, 1999) achieve tunneling.The context-graph based crawler [56] also employs a best-search heuristic, butthe classifiers used in this approach learn the layers which represent a set ofpages that are at some distance to the pages in the target class (layer).

Morespecifically, given a set of seeds, for each page in the seed set, pages that directlyrefer to this seed page (i.e., parents of the page) constitute layer-1 train set, pagesthat are referring to these layer-1 pages constitute the layer-2 train set, and soon; up to some predefined depth limit. The overall structure is called the contextgraph, and the classifiers are trained so that they assign a given page to oneof these layers with a likelihood score.

The crawler simply makes use of theseclassifier results and inserts URLs extracted from a layer-i page to the layer-i queue, i.e., it keeps a dedicated queue for each layer. URLs in each queue arealso sorted according to the classifier's score. While deciding the next page tovisit, the crawler prefers the pages nearest to the target class that is, the URLspopped from the queue that correspond to the first nonempty layer with thesmallest layer label. This approach clearly solves the problem of tunneling, butit requires constructing the context graph, which in turn requires finding pageswith links to a particular page (back links). In contrast, our rule-based crawleruses forward links while generating the rules and transitively combines these rulesto effectively imitate tunneling behavior.

In particular, reinforcement learning is used in CORA's focused crawler. CORA'scrawler basically searches for the expected future reward by pursuing a pathstarting from a particular URL. The training stage of classifier(s) involves learning the paths that may lead to on-topic pages in some number of steps.

In contrast, our rule-based crawler does not need to see a path of links during training, but constructs the paths using the transitive combination and chaining of simple rules of length 1.

The focused crawler of Web Topic Management System (WTMS) fetches onlypages that are close (i.e., parent, child, and sibling) to on-topic pages (Mukherjea, S. 2000). InWTMS, the relevancy of a page is determined by only using IR-based methods.In another work, Aggarwal et al. attempt to learn the Web's linkage structure todetermine a page's likelihood of pointing to an on-topic page. However, theydo not consider interclass relationships in the way we do in this study. Bingo!Is a focused-crawling system for overcoming the limitations of initial trainingby periodically retraining the classifier with high quality pages (Diligenti, M., 2000).

Recently,Menczer et al. present an evaluation framework for focused crawlers and introducean evolutionary crawler (Sizov, S., Graupmann, J., & Theobald, M. 2003). In another work, Pant and Srinivasan provide asystematic comparison of classifiers employed for focused crawling task (Menczer, F., Pant, G., & Srinivasan, P. 2004).

Two recent methods that exploit link context information are explored in (Menczer, F., Pant, G., & Srinivasan, P. 2004). In the first approach, so called text-window, only a number of

words around eachhyperlink is used for determining the priority of that link. The second one, tagtreeheuristic, uses the words that are in the document object model (DOM) treeimmediately in the node that a link appears, or its parents, until a threshold issatisfied.

In (Craven, et.at 1998) we proposed a similar but slightly different technique, so-calledpage segmentation method, which fragments a Web page according to the use ofHTML tags.

Focused crawling paradigm is employed in a number of prototype systems for gathering topic/domain specific Web pages. For instance, in (Pirkola, A. 2012), focused crawling is used for obtaining high quality pages on a mental health topic (depression).In (Pant, G., & Srinivasan, P. 2006), a prototype system is constructed that achieves focused crawling and multilingual information extraction on the laptop and job offers domains.

Black Box Testing Scanner (BBTS) was developed in (Chen and Wu, 2010) to find SQL injection vulnerabilities using dynamic approach. The BBST uses state aware crawler to identify forms and links with injection parameters to In this approach the authors' uses state aware scanner that will able to recognized webpage that contain injection parameter.

This improvement enable scanner to save time by not downloading pages that does not contain injection parameter and also avoid false positive that result in from attacking pages that does not have injection parameter. Author experimental evaluation show the scanner achieved 100% accuracy on tested application in short period of time. However author did not evaluate his approach with available existing technique to evaluate both accuracy and efficiency of the proposed scanner.

Viper is black box scanner proposed in (Ciampa et al., 2010) that detects SQLI vulnerability through dynamic testing. Scanners that implement dynamic approach are required to have different number of attacks type as well as way of analyzing server response. Viper uses different number of predefined attack pattern to trigger hidden vulnerability in application but it is capable of performing single analysis on server response to attacks. Experiment shows that Viper was able to carry successful attacks that trigger blind SQL injection vulnerability but Viper was not able to report it, this is because analysis was not address to analyzed blind SQL injection vulnerability.

Static Scanner was proposed in (Shar and Tan, 2012) that detect vulnerable point by characterizing input function into pattern of code attributes. Static code attributes are collected from backward static program slices of sensitive program points, so that to mine both input sanitization code patterns and input validation code patterns, from such static code attributes. This scanner uses vulnerabilities prediction model to enable scanner to predict vulnerable code for SQLI and cross site scripting (XSS) vulnerabilities. Authors evaluated their scanner with different PHP web-based application source code, which shows the effectiveness of their approach. However their approach can only be effective in PHP web-based applications.

4SQLi is the scanner that combines both static and dynamic approach to detect SQLI vulnerabilities (Appelt et al., 2014). 4SQLi uses a single or multiple mutation operators of different types that can be used as a single input parameter to generate desired inputs which will use latter for detecting subtle vulnerabilities that can only be triggered with an input generated by combining multiple mutation operators. For example, consider an application that alters inputs by searching for known attack patterns that can be generated using one of the behavior-changing operators. To form a successful attack, it is necessary to apply a behavior-changing operator and then apply one or more obfuscation operators.

III. DISCUSSION

This study presented explored existign solutions on malicious link detector and result of our analysis reveals that the existing solution faces two common challenges. One is bypassing login authentication; bypassing login application using SQL injection attack is highly dependent on how developer designed authentication query and type of backend database used in target application. Because it is possible single attack pattern that bypass login application in mysql version 5 may failed to bypass mysql version 5.1. A query that is perfoming check on numbers of return is more easier to bypass than query that is perfoming checking on single rows returned. Therefore, our malicious link detector perform brute force attack on attempt to bypass login authentication since it has no knowledge of knowing the type and version database is using on backend application.

Another challege faced by our malicious link detector is inability to recognize maliciouslinks underneath valid link. This is one of the reasons why our malicios link detector failed to identify any malicious link underneath valid link.

RECOMMENDATION

In short, the our recommendation on how to improve the current state of art is as follows:

i. Enhancing crawler intelligence to perform full indexing of links in target application even if the page requires partial page refreshment.
ii. Enhancing crawler intelligence to recognize the page as potential to malicious link attack as long as link is directly commutating with database and end users.
iii. Shaping of pattern used in tautology attaks to limit the number of return record into one to bypass the safety check applied by the developer so as to bypass login application even in the presence of the safety check.
iv. Updating database of SQL qeury related error messages return by different database server including Oracle, MySQL server and Sybase so as to avoid false positive on platforms other thant PHP applications.

REFERENCES

[1]. Antunes, Nuno, & Vieira, Marco. (2011). Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services. Paper presented at the Services Computing (SCC), 2011 IEEE International Conference on.
[2]. Antunes, Nuno, & Vieira, Marco. (2012). Evaluating and improving penetration testing in web services. Paper presented at the Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on.
[3]. Antunes, Nuno, & Vieira, Marco. (2015). Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples. Services Computing, IEEE Transactions on, 8(2), 269-283.
[4]. Appelt, Dennis, Nguyen, Cu Duy, Briand, Lionel C, & Alshahwan, Nadia. (2014). Automated testing for SQL injection vulnerabilities: an input mutation approach. Paper presented at the Proceedings of the 2014 International Symposium on Software Testing and Analysis.
[5]. Bandhakavi, Sruthi, Bisht, Prithvi, Madhusudan, P, & Venkatakrishnan, VN. (2007). CANDID: preventing sql injection attacks using dynamic candidate evaluations. Paper presented at the Proceedings of the 14th ACM conference on Computer and communications security.
[6]. Bau, Jason, Bursztein, Elie, Gupta, Divij, & Mitchell, John. (2010). State of the art: Automated black-box web application vulnerability testing. Paper presented at the Security and Privacy (SP), 2010 IEEE Symposium on.
[7]. Boyd, Stephen W, & Keromytis, Angelos D. (2004). SQLrand: Preventing SQL injection attacks. Paper presented at the Applied Cryptography and Network Security.
[8]. Buehrer, Gregory, Weide, Bruce W, & Sivilotti, Paolo AG. (2005). Using parse tree validation to prevent SQL injection attacks. Paper presented at the Proceedings of the 5th international workshop on Software engineering and middleware.
[9]. Cenzic's. (2014). Application Vulnerability Trends Report: 2014. Retrieved 29/09/2015, from https://www.info-point-security.com/sites/default/files/cenzic-vulnerability-report-2014.pdf
[10]. Chen, Jan-Min, & Wu, Chia-Lun. (2010). An automated vulnerability scanner for injection attack based on injection point. Paper presented at the Computer Symposium (ICS), 2010 International.
[11]. Cheon, Eun Hong, Huang, Zhongyue, & Lee, Yon Sik. (2013). Preventing SQL Injection Attack Based on Machine Learning. International Journal of Advancements in Computing Technology, 5(9).
[12]. Cho, Ying-Chiang, & Pan, Jen-Yi. (2015). Design and Implementation of Website Information Disclosure Assessment System. PloS one, 10(3), e0117180.
[13]. Ciampa, Angelo, Visaggio, Corrado Aaron, & Di Penta, Massimiliano. (2010). A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications. Paper presented at the Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems.
[14]. Cook, William R, & Rai, Siddhartha. (2005). Safe query objects: statically typed objects as remotely executable queries. Paper presented at the Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on.
[15]. Djuric, Zoran. (2013). A black-box testing tool for detecting SQL injection vulnerabilities. Paper presented at the Informatics and Applications (ICIA), 2013 Second International Conference on.
[16]. Halfond, William GJ, & Orso, Alessandro. (2007). Detection and prevention of sql

injection attacks Malware Detection (pp. 85-109): Springer.
[17]. Huang, Shih-Kun, Lu, Han-Lin, Leong, Wai-Meng, & Liu, Huan. (2013). Craxweb: Automatic web application testing and attack generation. Paper presented at 7th International Conference on the Software Security and Reliability(SERE), IEEE.
[18]. Kumar, Praveen. (2013). The multi-tier architecture for developing secure website with detection and prevention of sql-injection attacks. International Journal of Computer Applications, 62(9), 30-35.
[19]. Lawal, MA, Sultan, Abu Bakar Md, & Shakiru, Ayanloye O. (2016). Systematic Literature Review on SQL Injection Attack. International Journal of Soft Computing, 11(1), 26-35.
[20]. Liban, Abdilahi, & Hilles, Shadi. (2014). Enhancing Mysql Injector vulnerability checker tool (Mysql Injector) using inference binary search algorithm for blind timing-based attack. Paper presented at the Control and System Graduate Research Colloquium (ICSGRC), 2014 IEEE 5th.
[21]. Liu, Anyi, Yuan, Yi, Wijesekera, Duminda, & Stavrou, Angelos. (2009). SQLProb: a proxy-based architecture towards preventing SQL injection attacks. Paper presented at the Proceedings of the 2009 ACM symposium on Applied Computing.
[22]. Livshits, V Benjamin, & Lam, Monica S. (2005). Finding Security Vulnerabilities in Java Applications with Static Analysis. Paper presented at the Usenix Security.
[23]. McClure, Russell A, & Kruger, Ingolf H. (2005). SQL DOM: compile time checking of dynamic SQL statements. Paper presented at the Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on.
[24]. Medhane, Munqath H Alattar SP. R-WASP: Real Time-Web Application SQL Injection Detector and Preventer.