UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

Automating Penetration Testing Within An Ambiguous Testing Environment

Lim Kah Seng, Norafida Ithnin & Syed Zainudeen Mohd Shaid
Faculty of Computing, Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Johor, Malaysia
lim0709@gmail.com, afida@utm.my, szainudeen@utm.my

Submitted: 12/01/2018. Revised edition: 25/05/2018. Accepted: 27/05/2018. Published online: 21 November 2018

***Abstract*—Automated web application penetration testing has emerged as a trend. The computer was assigned the task of penetrating web application security with penetration testing technique. Relevant computer program reduces time, cost, and resources required for assessing a web application security. At the same time, scaling down tester reliance on human knowledge. Web application security scanner is such kind of program that is designed to assess web application security automatically with penetration testing technique. The downside is that computer is not well-formed as human. Consequently, web application security scanner often found generating the false alarms, especially in a testing environment, which web application source codes are unreachable. Thus, in this paper, the state-of-the-art of black box web application security scanner is systematically reviewed, to investigate the approaches for detecting web application vulnerability in an ambiguous testing environment. This survey is critical in providing insights on how to design efficient algorithms for assessing web application security with penetration testing technique in the ambiguous environment.**

***Keywords*—Penetration testing, Web application security scanner, False alarm, Ambiguous testing environment**

I. INTRODUCTION

Expansion in computer's computation power plus creativity of researchers in creating the efficient algorithms to simulate the task of web application penetration testing has resulted in an introduction of a computer program known as web application security scanner. In addition to that, automated web application penetration testing is becoming ubiquitous with increases in usage of web application security scanner in web application penetration testing's methodology.

Assorted algorithms were deriving from the fundamental white box, black box, or grey box testing technique for instructing computer assessing web application security automatically with penetration testing technique. Derived white box testing techniques statically parsed web application sources code to locate web application vulnerability. Alternatively, derived black box testing techniques dynamically examines web application execution behaviours to detect anomaly that proving existence of web application vulnerability. [1], [2]. On the other hand, derived hybrid testing techniques leveraged both white box and black box testing techniques stated in [3], [4]. The intention of integrating both white box and black box testing techniques is to improve the test coverage, while reduces white box and black box web application security scanners false alarms as advertised in [5]–[7].

The automated web application penetration testing is achievable by translating related testing technique into an executable algorithm. However, the corresponding activity is challenging, considering that a computer is merely a dummy machine that performs the calculation based on predefined set of instructions. The computer neither able to self-learning nor responds heuristically to unexpected events. Therefore, whenever web application security scanner is assessing a web application security within the ambiguous environment, false alarms are produced.

Unreachable web application source code, and unpredictable web application events, actions, and responses are what create the ambiguous environment, or so-called black box testing environment. Moreover, a web application always behaves differently to diverse data that entering the web application's data entry point. Dealing with such ambiguity is challenging. Therefore, false alarms often found reported by

the black box web application security scanner. Black box web application security scanner produces false positives that mistakenly interpret valid execution behaviours as vulnerability or false negatives that describes the missed vulnerability [2], [8].

In [1], [9]–[12], quality of black box web application security scanners was quantified. Experimental outcomes of [13]–[15] shown problem of false positive and false negative is severe in existing black box web application security scanners. Consequently, this paper reviews the state-of-the-art of black box web application security scanner for clarifying their strengths and limitations in detecting web application vulnerability within the ambiguous testing environment. In this paper, following items are deliverable:

- Strengths and limitations of web application security scanners in assessing web application security with penetration testing technique within the ambiguous testing environment.
- Factors that caused black box web application security scanners generate false alarms.
- State-of-the-art of black box web application security scanners.

The remaining part of the manuscript comprised of following sections. Section two defines web application security scanner and its general architecture. In Section three, the state-of-the-art of black box web application security is systematically reviewed. The related works are elaborate in Section four. Subsequently, Section five discusses the outcome of the literature review. Finally, Section six concluded the paper.

II. WEB AAPLICATION SECURITY SCANNER

Web application security scanner is a computer program that automatically scans a web application for web application vulnerability detection. This computer program simulates penetration tester's activity, penetrating web application attack vectors with selected attack payload to detect web application vulnerabilities [16]–[19],

Presently, three classes of web application security scanner namely white box, black box, and hybrid web application security scanners are available. White box web application security scanner parsing web application source code, statically tracking the propagation of malicious data from source to sink to detect web application vulnerability by detects changes in the semantics of web application source code. [20], [21]. On the other hand, the black box web application security scanner detects web application vulnerability by dynamically executing under-test web application on a web browser, analysing web application responses for the existence of anomaly [18], [22]. In the meanwhile, hybrid web application security scanner detects web application vulnerability by integrating both testing techniques of white box and black box web application security scanners [23].

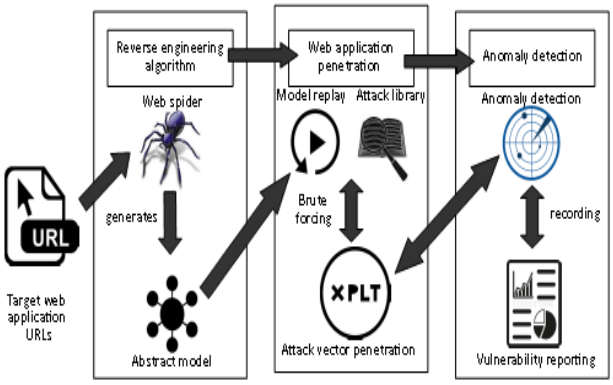

Fig. 1. General architecture of black box web application security scanner

Web application security scanner dynamically executes under-test web application on a web browser to solve the challenge of assessing web application security, under a circumstance that web application source code is unreachable. Implemented reverse engineering algorithms mined web applications' DOM document for data entry point discovery. Subsequently, selected malicious texts are injecting into identified data entry points. If the malicious data was successfully triggering the web application abnormal execution behaviours, vulnerability is deemed present, or vice versa. Therefore, a web application security scanner that can assess web application security in the ambiguous testing environment generally comprised of three set of algorithms as stated in [24]–[26]. The first set of algorithms reverse engineers web application for reconnaissance purpose. The second set of algorithms penetrating web application security with designated malicious data. Lastly, the third set of algorithms detects web application vulnerability by analysing web application responses for the occurrence of anomalies, as depicted in Fig. 1.

III. STATE-OF-THE-ART OF AUTOMATED BLACK BOX WEB APPLICATION PENETRARTION TESTING

Addressing the challenge of automated web application vulnerability assessment with penetration testing technique, in the ambiguous testing environment, had led to the invention of algorithms like [22], [27]. This section is about reviewing the state-of-the-art of the corresponding algorithms.

A. Reverse Engineer Web Application

Invented reverse engineering algorithms have a critical role in dynamically executing a web application for reconnaissance purpose. These reverse engineering algorithms load under-test web application on a web browser to mine DOM documents for data entry points, submitting data to the web server, and interact with active web elements for navigation purpose. Introduced reverse engineering algorithm generally is a web crawler that usually contains a DOM parser, a client-side executor, and a robot. DOM parser

responsible for mining DOM document to retrieve active web elements. The client-side executor has client-side scripts executed. Lastly, the robot is critical in interacting with found web elements and inputting web forms with valid data [28]–[31].

Since some events, web pages, or web contents are retrievable only if valid data is inputting to the web application. Therefore, algorithms that capable of submitting legitimate data to the back-end server is critical in addressing a research problem known as hidden web crawling. An initial version of the corresponding algorithms attempts to mitigate hidden web crawling problem by inputting web application with random data, or halted web crawling process for inquiry of input value from the tester. However, because of randomly generated data was discovered often failed in bypassing the input sanitization function. Moreover, halting the crawling process for inquiring of an input value from the tester is computationally inefficient. Researchers had suggested more sophisticated algorithms, which select appropriate input value from a custom-made library, after extracting semantics of the data entry points. These algorithms extract semantics of web forms. Afterward, retrieve suitable input value from a library by searching for the matched keywords using string distance calculation algorithm. An algorithm knowing as LITE (Layout-based information algorithm) was suggested by [32] for extracting web form semantics. In the meanwhile, [19], [22] propose IKM (Information Knowledge Manager) for inputting web forms with legitimate data. Besides this, SmartProfiles and Google Fusion Table are the two custom-made libraries proposed by [32] and [33] respectively.

Model checking technique had adopted by web application security scanner for modelling web application execution behaviours. This model checking technique records web application navigations using an abstract model likes finite state machine, tree graph, or flow diagram [34]–[36], to achieve the state-aware web crawling. Nevertheless, state explosion can be a severe limitation for model checking technique. Hence, state pruning algorithms are proposed to prevent state explosion by reducing the size of an abstract model, by distinguishing previously visited states from those newly discovered. Classification of web application states is computed by calculates string distances or tree structures of visited DOM documents, using string distance calculation algorithms such as Levenshtein's algorithm or SimHash [37], [38]. A new state is deemed existed if DOM documents of visited web pages are found different, with a new node is added to the generated abstract model. [39] had builds relevant formulas for calculating occurrence of state change, with colouring technique was introduce for systematically adding a newly discovered state to the abstract model. In addition to that, an algorithm that for pruning the size of an abstract model is introduced by researchers to address state explosion problem, while handling modern web application dynamic execution behaviours. The corresponding algorithm eliminates infinite sections of retrieved DOM documents, to look for idle sections [36].

B. Penetrating the Attack Vector

Brute forcing techniques had been widely practiced by existing black box web application security scanner for penetrating the web application security. Fuzzing technique populates web application attack vector with randomly generated data, with an assumption the attack vector is comprisable with generated random data. However, because of random data is easily sanitized by the defensive mechanism of a web application [27], [40]. Therefore, alternate fuzzing technique that generates attack codes based on attack vector semantics is applied to increase the likelihood of successful exploitation.

Given that some web application vulnerabilities like SQL injection, cross-site scripting, cross-site request forgery are detectable only with selected malicious data. For instance, SQL injection attack is achievable only with malicious SQL query. In the meanwhile, cross-site scripting attack requiring injection of malicious client-side scripts. Therefore, another brute forcing technique knowing as fault injection is proposed by [1], [22], [41]. This fault injection technique brute forcing web application attack vectors with selected attack codes of an attack library for penetrating web application security. Unfortunately, attack codes of an attack library always limited in number and low in variety. Moreover, major attack library does not receive often update as stated in [42], [43]. Consequently, web application security scanners often found failed in penetrating the attack vector security. This lead to proposing of an algorithm for evolving or mutating attack codes of an attack library with mutating and crossover operators of the genetic algorithm [3], [40], [44]. In the meanwhile, a learning-based algorithm was introduced by [3] for expanding the attack coverage, while improving usage of computer's computation resource, by eliminating the need of brute forcing attack vector with attack codes of attack library exhaustively.

C. Detecting the Anomaly

The black-box web application security scanner detects web application vulnerability by locating the anomaly. Since anomalies are always some string of texts. Assorted string matching or string distance calculation algorithms are used in existing black box web application security scanners to detect web application vulnerability.

Signature or learning based vulnerability detection algorithms are what implemented in existing web application security scanner for locating the malicious string of texts. The learning-based algorithm compares string distance of URLs, system commands, or DOM documents learned with innocent and malicious input to define the availability of anomaly. Assorted string distance algorithms like SimHash, Levenshtein's algorithm, Jarod's algorithm was implemented to identify the anomaly. In addition to that, an experimentation was conducted for studying the effectiveness of relevant string distance algorithms. Related experiment outcomes are

retrievable in [45]. Besides this, a set of security rules is deriving from learned execution behaviours, to support the statement that under-test web application is vulnerable to malicious data injected. Several security rules were defined by [2], [46], [47] for showcasing successful SQL injection or cross-site scripting attacks.

In the meanwhile, signature-based algorithms detect web application vulnerability by searching web application responses to locate a specific string of malicious text. Most of the time, string pattern matching algorithms like Boyer's string matching algorithm [48], [49] are used to detect the malicious data. Detecting vulnerability likes cross-site scripting requiring the algorithm searching web application responses for malicious client-side scripts. On the other hand, SQL injection is detected by analysing generated SQL query for the existence of attack code, or to define whether there is a raised of exceptions by database management system on web application's web page [50]–[52].

IV. RELATED WORKS

Experiments were conducted by researchers for quantifying black box web application security scanner quality. In these experiments, black box web application security scanners were configured to scan selected test-beds, which are very vulnerable web applications equipped with known web application vulnerabilities. Quality of a black box web application security scanner is defined by calculates the number of detected web application vulnerability. Experiment outcomes of [53]–[55] shown black box web application security scanners are perform well in detecting simple injection-based vulnerabilities like reflected SQL injection and cross-site scripting. Nevertheless, black box web application security scanners contain weaknesses like low in test coverage and tend to generate false alarms such as false positives and false negatives [6], [56]–[58].

V. DISCUSSION

Assorted algorithms were invented by practitioners to enable black box web application security scanner conducts automated web application penetration testing without having to access to web application source code. Suggested reverse engineering algorithms crawl web application to interpret web application execution behaviours while inputting web application with data to mitigate hidden web crawling problem. However, because hidden web contents were revealed only with valid data were inputted. Moreover, designing a sophisticated algorithm to input each data entry point with appropriate data is challenging due to the ambiguity of semantic of data entry points. Major hidden data entry points and attack vectors of the under-test web application are hard to reach. Consequently, some part of web applications not successful included in the automated web application penetration testing, with the generation of false negatives by black box web application security scanners.

Brute forcing techniques like fault injection and fuzzing are leverage by web application security scanners for penetrating web application security. The web application security scanners brute force found attack vectors with selected attack codes of an attack library for compromising a web application confidentiality, integrity, or availability. But because writing an efficient attack code that manages to bypass the input sanitization function is as challenging as creating an algorithm for inputting the web forms. In addition to that, attack codes of an attack library always limited in number and variety. As a result, most exploitation performed on discovered attack vectors is fail.

The presence of web application vulnerabilities as a string of text has promoted usage of learning-based and signature-based vulnerability detection for detecting the web application vulnerabilities. These algorithms search or compare web application responses for detecting web application vulnerabilities. Weaknesses of existing vulnerability detection algorithms are that they are too conservative. For instance, the appearance of exceptions on DOM documents does not necessary means web application is vulnerable to SQL injection. Raised exceptions could be caused by improper database configuration or failure of establishing the secure connection to a database. Therefore, signature-based and learning-based algorithms were found having difficulty in detecting successful exploitations, especially those web application vulnerabilities, which embedded within complicated computation steps.

In summary, the challenge of achieving automated web application penetration testing within the ambiguous testing environment is at designing sophisticated algorithms for interpreting and understanding the semantics of under-test web application with an absence of source codes. These algorithms are playing a critical role in enabling web application security scanner locates the data entry points or attack vectors, select suitable attack codes from attack library, as well as knowing the location to look for successful exploitations [3], [59]–[61]. Table I summarized algorithms proposed by researchers for addressing challenges of performing the automated web application penetration testing on the ambiguous testing environment.

TABLE I. THE SUMMARY OF STATE-OF-THE-ART OF BLACK BOX WEB APPLICATION SECURITY SCANNERS

Components	Research Problem	Proposing approach	Description	Authors
Reverse engineering.	Hidden web crawling.	Authorization Authentication Data Table.	An algorithm for bypassing web application authentication scheme.	[2]
		Information Knowledge Manager.	Algorithms for inputting web forms with valid data.	[22], [52]
		Layout-based Information Extraction Technique (LITE).	An algorithm for extracting web forms semantic.	[32]
		SmartProfiles/ Google Fusion.	Libraries that providing the valid input values.	[33], [62]
		Interactive web crawling	An algorithm for inputting web form by interrupting web crawling process to request input value from tester.	
	Web application modelling.	State-aware crawling.	Algorithms for modelling web application navigations or event with model checking technique.	[27], [39]
		State change detection algorithms.	Algorithms to detect occurrence of state change.	[36], [39]
	State explosion prevention.	Elimination of infinite section.	An algorithm for excluding web application infinite sections.	[36]
Exploitation	Penetrating attack vectors security.	Brute forcing.	Algorithm to penetrating web application attack vector with random data.	[63], [64]
		Fault injection.	An algorithm for penetrating web application attack vector with attack codes of an attack library.	[22]
		Exploit mutations.	Algorithms for evolving or mutating attack codes with genetic algorithm.	[40], [65], [66]
Vulnerability detection	Detecting the successful exploitation	Signature-based vulnerability detection	Algorithms for detecting web application vulnerability by examining web application response for specific string of text.	[50], [51]
		Learning -based vulnerability detection	Algorithms for detecting web application vulnerability by identifying the violation of defined innocent execution behaviours.	[67], [68]

VI. CONCLUSION

Leveraging the computer for automated web application penetration testing is becoming a trend. Despite automated web application penetration testing reduces time, cost, and knowledge required. The automated web application penetration testing is helpful in preserving the knowledge of web application penetration testing. Given that web application source code is not always accessible during an automated web application penetration testing. Algorithms are proposed by practitioners for assessing web application security with penetration testing technique within the ambiguous testing environment. Proposed reverse engineering algorithms interpret contexts and semantics for data entry points or attack vectors identification. In the meanwhile, brute forcing algorithms are used to penetrate web application security. Lastly, Proposed learning-based and signature-based vulnerability detection algorithms detect web application vulnerability by examining web application responses to trace the anomaly. But, because of web application source code is not accessible, interpreting and understanding of contexts and semantics of an under-test web application are becoming challenging. Consequently, security of some part of the under-test web application is not precisely accessed, selected attack codes are failed in penetrating web application security, with successful exploitations are not detectable. These weaknesses elaborate why web application security scanners are often found generating false alarms. Nowadays, sophisticated algorithms are still needed to solve the challenge of automatically assessing web application security with penetration testing technique within the ambiguous testing environment.

REFERENCES

[1] J. Fonseca, M. Vieira, and H. Madeira. (2007). Testing and Comparing Web Vulnerability Scanning Tools for SQL

Injection and XSS Attacks. *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, 365-372.

[2] X. Wang, L. Wang, G. Wei, D. Zhang, and Y. Yang. (2010). Hidden Web Crawling for SQL Injection Detection. *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, 14-18.

[3] O. Tripp, O. Weisman, and L. Guy. (2013). Finding Your Way in the Testing Jungle: A Learning Approach to Web Security Testing. *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, 347-357.

[4] I. Medeiros, N. F. Neves, and M. Correia. 2014. Automatic Detection and Correction of Web Application Vulnerabilities Using Data Mining to Predict False Positives. *Proceedings of the 23rd international conference on World wide web*, 63-74.

[5] L. Suto. (2010). Analyzing the Accuracy and Time Costs of Web Application Security Scanners. *San Franc. Febr*.

[6] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell. (2010). State of the Art: Automated Black-box Web Application Vulnerability Testing. *Security and Privacy (SP), 2010 IEEE Symposium on*, 332-345.

[7] J. Fonseca, M. Vieira, and H. Madeira. (2007). Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks. *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, 365-372.

[8] Crawling the Content Hidden Behind Web Forms | SpringerLink. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-74477-1_31. [Accessed: 13-Dec-2017].

[9] L. Suto. (2007). Analyzing the Effectiveness and Coverage of Web Application Security Scanners. *San Franc. Oct.*

[10] N. Antunes and M. Vieira. (2010). Benchmarking Vulnerability Detection Tools for Web Services. *Web Services (ICWS), 2010 IEEE International Conference on*, 203-210.

[11] Y. Makino and V. Klyuev. (2015). Evaluation of web vulnerability scanners. *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on*, 1, 399-402.

[12] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell. (2010). State of the Art: Automated Black-Box Web Application Vulnerability Testing. *Security and Privacy (SP), 2010 IEEE Symposium on*, 332-345.

[13] Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan. (2013). A Cost-effective Approach to Evaluating Security Vulnerability Scanner. *Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific*, 1-3.

[14] H. Holm, T. Sommestad, J. Almroth, and M. Persson. (2011). A Quantitative Evaluation of Vulnerability Scanning. *Inf. Manag. Comput. Secur.*, 19(4), 231-247.

[15] M. Parvez, P. Zavarsky, and N. Khoury. (2015). Analysis of Effectiveness of Black-box Web Application Scanners in Detection of Stored SQL Injection and Stored XSS Vulnerabilities. *Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for*, 186-191.

[16] E. Fong and V. Okun. (2007). Web Application Scanners: Definitions and Functions. *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, 280b-280b.

[17] P. Baral. 2011. Web Application Scanners: A Review of Related Articles [Essay]. *IEEE Potentials*, 30(2), 10-14.

[18] M. Curphey and R. Arawo. (2006). Web Application Security Assessment Tools. *IEEE Secur. Priv.*, 4(4), 32-41.

[19] Y.-W. Huang and D. T. Lee. (2005). Web Application Security—Past, Present, and Future. *Computer Security in the 21st Century*, Springer, 183-227.

[20] F. Alssir and M. Ahmed. (2012). Web Security Testing Approaches: Comparison Framework. *Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science*, 163-169.

[21] J. Thomé, A. Gorla, and A. Zeller. (2014). Search-based Security Testing of Web Applications. *Proceedings of the 7th International Workshop on Search-Based Software Testing*, 5-14.

[22] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai. (2003). Web Application Security Assessment by Fault Injection and Behavior Monitoring. *Proceedings of the 12th international conference on World Wide Web*, 148-159.

[23] Z. ĐURIĆ. (2014). WAPTT-Web Application Penetration Testing Tool. *Adv. Electr. Comput. Eng.*, 14(1).

[24] J.-M. Chen and C.-L. Wu. (2010). An Automated Vulnerability Scanner for Injection Attack Based on Injection Point. *Computer Symposium (ICS), 2010 International*, 113-118.

[25] M. Balduzzi, C. T. Gimenez, D. Balzarotti, and E. Kirda. (2011). Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications. *NDSS*, 2011.

[26] D. Gol and N. Shah. (2015). Detection of Web Appication Vulnerability based on RUP Model. *Recent Advances in Electronics & Computer Engineering (RAECE), 2015 National Conference on*, 96-100.

[27] F. Duchene, S. Rawat, J.-L. Richier, and R. Groz. (2013). LigRE: Reverse-engineering of Control and Data Flow Models for Black-box XSS Detection. *Reverse Engineering (WCRE), 2013 20th Working Conference on*, 252-261.

[28] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai. (2003). Web Application Security Assessment by Fault Injection and Behavior Monitoring. *Proceedings of the 12th international conference on World Wide Web*, 148-159.

[29] Y.-W. Huang and D. T. (2005). Web Application Security—Past, Present, and Future. *Computer Security in the 21st Century*, Springer, 183-227.

[30] Z. Djuric. (2013). A Black-box Testing Tool for Detecting SQL Injection Vulnerabilities. *Informatics and Applications (ICIA), 2013 Second International Conference on*, 216-221.

[31] G. Pellegrino, C. Tschürtz, E. Bodden, and C. Rossow, "jÄk. (2015). Using Dynamic Analysis to Crawl and Test Modern Web Applications. *International Workshop on Recent Advances in Intrusion Detection*, 295-316.

[32] S. Raghavan and H. Garcia-Molina. (2000). Crawling the Hidden Web. Stanford.

[33] F. R. Muñoz and L. J. G. Villalba. (2015). Web from Preprocessor for Crawling. *Multimed. Tools Appl.*, 74(19), 8559-8570.

[34] M. E. Dincturk, G.-V. Jourdan, G. V. Bochmann, and I. V. Onut. (2014). A Model-based Approach for Crawling Rich Internet Applications. *ACM Trans. Web TWEB*, 8(3), 19.

[35] K. Benjamin, G. v Bochmann, G.-V. Jourdan, and I.-V. Onut. (2010). Some Modeling Challenges When Testing Rich Internet Applications for Security. *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*, 403-409.

[36] S. Choudhary, M. E. Dincturk, G. V. Bochmann, G.-V. Jourdan, I. V. Onut, and P. Ionescu. (2012). Solving Some Modeling Challenges When Testing Rich Internet Applications for Security. *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, 850-857.

[37] E. van Eyk, W. van Leeuwen, M. A. Larson, and F. Hermans. (2014). *Performance of Near-duplicate Detection Algorithms for Crawljax*. Citeseer, 2014.

[38] O. Lounis, S. E. B. Guermeche, L. Saoudi, and S. E. Benaicha. (2014). A New Algorithm for Detecting SQL Injection Attack in Web Application. in *Science and Information Conference (SAI), 2014*, 589-594.

[39] A. Doupé, L. Cavedon, C. Kruegel, and G. (2012). Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner. *USENIX Security Symposium*, 14.

[40] F. Duchene, S. Rawat, J.-L. Richier, and R. Groz. (2014). KameleonFuzz: Evolutionary Fuzzing for Black-box XSS Detection. *Proceedings of the 4th ACM conference on Data and application security and privacy*, 37-48.

[41] J. Fonseca and F. Matarese. (2013). Using Vulnerability Injection to Improve Web Security. *Innovative Technologies for Dependable OTS-Based Critical Systems*, Springer, 145-157.

[42] P. Xiong, B. Stepien, and L. Peyton. (2009). Model-based Penetration Test Framework for Web Applications using TTCN-3. *E-Technol. Innov. Open World*, 141-154.

[43] M. Balduzzi, M. Egele, E. Kirda, D. Balzarotti, and C. Kruegel. (2010). A Solution for the Automated Detection of Clickjacking Attacks. *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, 135-144.

[44] O. Tripp, M. Pistoia, P. Cousot, R. Cousot, and S. Guarnieri. (2013). Andromeda: Accurate and Scalable Security Analysis of Web Applications. *FASE*, 7793, 210-225.

[45] M. Zachara and D. Pa\lka. (2016). Comparison of Text-Similarity Metrics for the Purpose of Identifying Identical Web Pages During Automated Web Application Testing. *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology–ISAT 2015–Part II*, 25-35.

[46] E. Reshef, Y. El-Hanany, G. Raanan, and T. Tsarfati. (2007). System for Determining Web Application Vulnerabilities, 7237265, Jun-2007.

[47] M. I. P. Salas and E. Martins. (2014). Security Testing Methodology for Vulnerabilities Detection of Xss in Web Services and Ws-security. *Electron. Notes Theor. Comput. Sci.*, 302, 133-154.

[48] A. Z. M. Saleh, N. A. Rozali, A. G. Buja, K. A. Jalil, F. H. M. Ali, and T. F. A. Rahman. (2015). A Method for Web Application Vulnerabilities Detection by Using Boyer-Moore String Matching Algorithm. *Procedia Comput. Sci.*, 72, 112-121.

[49] T. F. A. Rahman, A. G. Buja, K. Abd, and F. M. Ali. (2017). SQL Injection Attack Scanner Using Boyer-Moore String Matching Algorithm. *JCP*, 12(2), 183–189.

[50] A. Z. M. Saleh, N. A. Rozali, A. G. Buja, K. A. Jalil, F. H. M. Ali, and T. F. A. Rahman. (2015). A Method for Web Application Vulnerabilities Detection by Using Boyer-Moore String Matching Algorithm. *Procedia Comput. Sci.*, 72, 112-121.

[51] T. F. A. Rahman, A. G. Buja, K. Abd, and F. M. Ali. (2017). SQL Injection Attack Scanner Using Boyer-Moore String Matching Algorithm. *JCP*, 12(2), 183-189.

[52] Y.-W. Huang, C.-H. Tsai, T.-P. Lin, S.-K. Huang, D. T. Lee, and S.-Y. Kuo. (2005). A Testing Framework for Web Application Security Assessment. *Comput. Netw.*, 48(5), 739-761.

[53] J. Bau, F. Wang, E. Bursztein, P. Mutchler, and J. C. Mitchell. Vulnerability Factors in New Web Applications: Audit Tools, Developer Selection & Languages.

[54] F. van der Loo. (2011). Comparison of Penetration Testing Tools for Web Applications. PhD Thesis, Master's thesis, University of Radboud, Netherlands.

[55] N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl. (2011). An Analysis of Black-box Web Application Security Scanners against Stored SQL Injection. *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. 1095-1101.

[56] L. Suto. (2007). Analyzing the Effectiveness and Coverage of Web Application Security Scanners. San Franc. Oct., 2007.

[57] N. Antunes and M. Vieira. (2009). Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services. Dependable Computing. *PRDC'09. 15th IEEE Pacific Rim International Symposium on*, 301-306.

[58] M. Vieira, N. Antunes, and H. Madeira. 20090. Using Web Security Scanners to Detect Vulnerabilities in Web Services. Dependable Systems & Networks, 2009. *DSN'09. IEEE/IFIP International Conference on*, 566-571.

[59] N. Antunes and M. Vieira. (2017). Designing Vulnerability Testing Tools for Web Services: Approach, Components, and Tools. *Int. J. Inf. Secur.*, 16(4), 435-457.

[60] T.-B. Dao and E. Shibayama. (2010). Coverage Criteria for Automatic Security Testing of Web Applications. *ICISS*, 111-124.

[61] T. B. Dao and E. Shibayama. (2011). Security sensitive Data Flow Coverage Criterion for Automatic Security Testing of Web Applications. *International Symposium on Engineering Secure Software and Systems*, 101-113.

[62] M. Benedikt, J. Freire, and P. Godefroid. (2002). VeriWeb: Automatically Testing Dynamic Web Sites. *Proceedings of 11th International World Wide Web Conference (WW W'2002*.

[63] N. Antunes and M. Vieira. (2013). SOA-Scanner: An Integrated Tool to Detect Vulnerabilities in Service-Based Infrastructures. Services Computing (SCC). *2013 IEEE International Conference on*, 280-287.

[64] N. Antunes and M. Vieira. (2009). Detecting SQL Injection Vulnerabilities in Web Services. Dependable Computing, 2009. *LADC'09. Fourth Latin-American Symposium on*, 17-24.

[65] N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl. (2011). Testing and Assessing Web Vulnerability Scanners for Persistent SQL Injection Attacks. *Proceedings of the First International Workshop on Security and Privacy Preserving in e-Societies*, 12-18.

[66] A. Avancini and M. Ceccato. (2013). Comparison and Integration of Genetic Algorithms and Dynamic Symbolic Execution for Security Testing of Cross-Site Scripting Vulnerabilities. *Inf. Softw. Technol.*, 55(12), 2209-2222.

[67] N. Li, T. Xie, M. Jin, and C. Liu. (2010). Perturbation-based User-Input-Validation Testing of Web Applications. *J. Syst. Softw.*, 83(112263–2274).

[68] Z. Djuric. (2013). A Black-box Testing Tool for Detecting SQL Injection Vulnerabilities. *Informatics and Applications (ICIA), 2013 Second International Conference on*, 216-221.